A Versatile and Flexible Chiplet-based System Design for Heterogeneous Manycore Architectures

Hao Zheng, Ke Wang, and Ahmed Louri
Department of Electrical and Computer Engineering, George Washington University, Washington D.C.
{*haozheng, cory, louri*}@*gwu.edu*

***Abstract*—Heterogeneous manycore architectures are deployed to simultaneously run multiple and diverse applications. This requires various computing capabilities (CPUs, GPUs, and accelerators), and an efficient network-on-chip (NoC) architecture to concurrently handle diverse application communication behavior. However, supporting the concurrent communication requirements of diverse applications is challenging due to the dynamic application mapping, the complexity of handling distinct communication patterns and limited on-chip resources. In this paper, we propose *Adapt-NoC*, a versatile and flexible NoC architecture for chiplet-based manycore architectures, consisting of adaptable routers and links. Adapt-NoC can dynamically allocate disjoint regions of the NoC, called subNoCs, for concurrently-running applications, each of which can be optimized for different communication behavior. The adaptable routers and links are capable of providing various subNoC topologies, satisfying different latency and bandwidth requirements of various traffic patterns (e.g. all-to-all, one-to-many). Full system simulation shows that Adapt-NoC can achieve 31% latency reduction, 24% energy saving and 10% execution time reduction on average, when compared to prior designs.**

I. Introduction

Today's heterogeneous manycore architectures are running multiple and diverse applications, requiring various computing (CPUs, GPUs and accelerators) and communication (cache and memory coherence) capabilities. However, a majority of these architectures deploy static network-on-chip (NoC) configurations, which are inefficient in supporting various traffic patterns of applications running at the same time. This often results in sub-optimal on-chip communication, therefore negatively affecting the application performance.

Reconfigurable NoCs [1]–[7] have been introduced to remedy the sub-optimal on-chip communication through providing application-specific NoC topologies. For example, prior work either inserts express links into mesh topology [1] or fully customizes network connectivity [2], [3] based on communication task graphs. Moreover, Polymorphic NoC [4] statically configures different router architectures (e.g. buffer size and crossbar) and network connections. While these static reconfigurable schemes are beneficial to single application execution, they have limited applicability for modern manycore architectures where multiple applications are running. These applications are often dynamically allocated into different regions of compute and memory resources [8]–[11], thus leading to frequently changed application mapping and diverse regional communication behavior. As a result, a flexible NoC architecture is envisioned to support the diverse and regional communication requirements of dynamically-allocated applications.

Designing such a flexible NoC architecture is challenging, as it requires a variety of network topologies to cover different traffic patterns (e.g. one-to-many, all-to-all), and bandwidth and latency requirements of diverse applications. The problem is further exacerbated by the dynamically sized and allocated application mapping, thus leading to substantially possible network connections within different mapped regions. An intuitive architectural design could exhaust limited on-chip power, area and wiring budgets, offsetting the performance benefits.

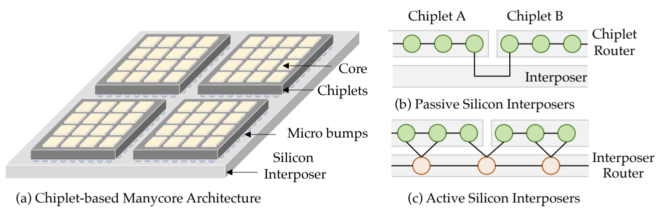

Core
Chiplets
Micro bumps
Silicon Interposer
(a) Chiplet-based Manycore Architecture
Chiplet A
Chiplet B
Chiplet Router
Interposer
(b) Passive Silicon Interposers
Interposer Router
(c) Active Silicon Interposers

Fig. 1. (a) Chiplet-based manycore architecture, (b) passive interposer, and (c) active interposer.

To this end, we propose **Adapt-NoC**, a versatile and flexible NoC design consisting of adaptable routers and links for chiplet-based manycore architectures. The Adapt-NoC can dynamically allocate the NoC into different regions, called subNoCs[1], each of which can be optimized in different and diverse subNoC topologies by exploiting additional routing resources of the interposer. Consequently, there could be several mesh, cmesh, torus, and tree topologies running simultaneously, supporting distinct traffic patterns with different latency and throughput demands.

The main contributions of Adapt-NoC are:

- Adapt-NoC is capable of providing multiple disjoint subNoC topologies with different configurations of size, bi-section bandwidth, and diameter.
- We propose a methodology to synergize various subNoC topologies to satisfy a variety of application categories and NoC requirements such as flow control and routing, thereby maximizing the performance benefits of the Adapt-NoC.

We evaluate the proposed Adapt-NoC using full system simulation with Parsec and Rodinia benchmark suites. Our simulation results show that the Adapt-NoC achieves 31% latency reduction, 24% energy saving, and 10% overall execution time reduction, as compared to prior NoC designs.

II. Background

Chiplet-based manycore architecture: The silicon interposer enables chip miniaturization, breaking one monolithic chip into smaller individual chips, called chiplets. As a result, a collection of chiplets can be placed side by side and stitched together on a silicon interposer via micro-bumps (μbumps) in a face-down manner, as illustrated in Fig. 1(a). To connect these chiplets, wires have to be routed through the interposer, as shown in Fig. 1(b), called passive interposer. Aside from this limited wiring, most of the interposer area and wiring resources are underutilized though have been paid. To exploit the full benefits of the interposer, recent work [12]–[14] has studied cost-effective ways of implementing active interposer designs, where both wires and transistors are deployed in the interposer using maturated and low-cost process

[1]Each region to which an application is mapped is called a sub-NoC in this paper.

978-1-7281-1085-1/20/$31.00 ©2020 IEEE

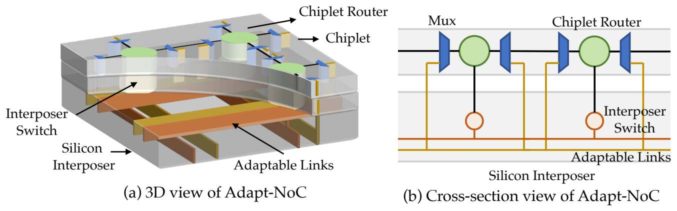


Fig. 2. An example of 2 × 2 Apdat-NoC consisting of adaptable routers and links.

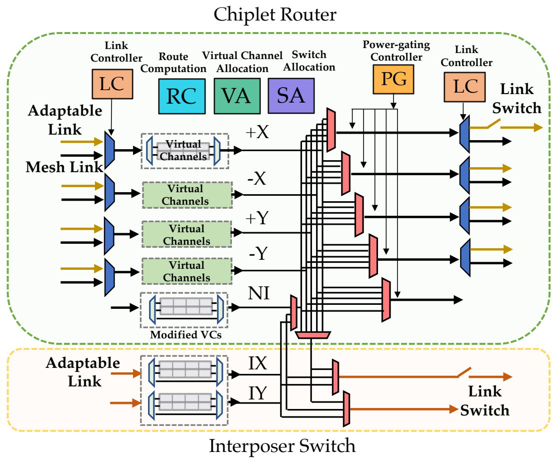


Fig. 3. The micro-architecture of adaptable router.

technologies (e.g. 65nm technology). Consequently, the miniaturized chiplet and "minimally active" interposer can offer more economic and performance benefits than the current monolithic chip designs [14]. Furthermore, the active interposer extends current NoC design to a two-layer NoC architecture, as shown in Fig. 1(c). The additional routing capability could efficiently accommodate the growing communication demands of high-bandwidth in-package memory (e.g. stacked high bandwidth memory). However, the connectivity between interposer and chiplet NoCs requires additional router radix at both ends while inevitably increasing hop count for those packets traversing between the interposer and the chiplet.

Reconfigurable NoCs: Runtime reconfiguration has also been proposed to improve NoC performance, reliability and energy saving. SMART [5] deploys a low-swing clock-less repeated link embedded within the router crossbar that can bypass packets to several hops away within a single cycle. A reconfigurable link design [6] can dynamically allocate channel bandwidth between adjacent routers, thereby efficiently improving network throughput and reliability. Panthre [7] reconfigures NoC topology to detour traffic away from the power-gated routers, improving NoC energy efficiency. All these techniques are orthogonal to the Adapt-NoC design, which could be simultaneously deployed to further improve NoC performance and energy benefits.

III. Adapt-NoC Architecture

A. Overview of Adapt-NoC

In this work, we use a 64-core chiplet-based manycore architecture as a baseline, which consists of four 4 × 4 mesh-based manycore chiplets stacked on a silicon interposer, as illustrated in Fig. 1(a,b). The Adapt-NoC is built on the baseline manycore architecture, with the goal of supporting the concurrent communication of diverse applications in different subNoCs. Specifically, the Adapt-NoC can dynamically provide multiple disjoint subNoCs, each of which can be configured as different subNoC topologies such as mesh, cmesh, torus, and tree. Fig. 2 depicts a 2 × 2 allocated subNoC, consisting of four adaptable routers and eight adaptable links. Each adaptable router consists of a chiplet router and an interposer switch. Adaptable links connect the adaptable routers in each row or column through the interposer. As a result, both chiplet routers and interposer switches are connected by adaptable links (depicted in different colors), horizontally or vertically.

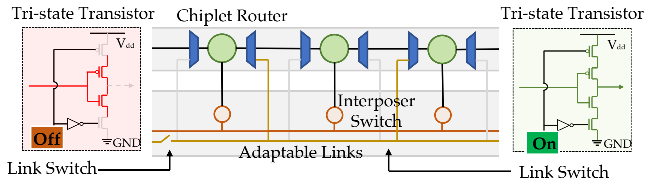


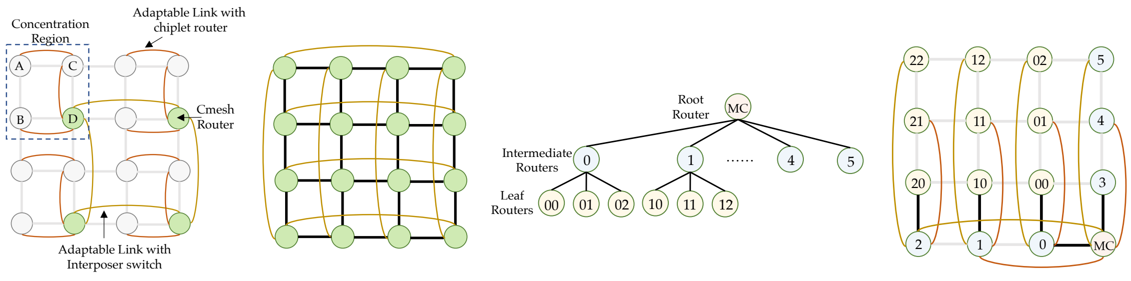
Fig. 4. Adaptable link is comprised of tri-state transistors.

B. Adaptable Router

The dynamically-allocated subNoC topology often results in significantly different router configurations (i.e. router radix and connection). We, therefore, propose an adaptable router with the aim of providing desired router radix (up to 7 radices) and connections with other routers. Specifically, Fig. 3 depicts the proposed micro-architecture of an adaptable router, which consists of a 5 × 5 chiplet router and 2 × 2 interposer switch. As compared to prior works [12], [13], we decompose the high-radix interposer router (i.e. 8 × 8 router) to four low-radix switches (i.e. 2 × 2 switch), each of which is connected with the crossbar of the chiplet router by two additional muxes (i.e. 3:1 and 5:1 muxes). The rationale behind this design is three-fold: (1) to reduce chiplet router radix, (2) to mitigate hop count at interposer, and (3) to compose a high-radix router when required. When the interposer switch operates independently, it functions as a bypass switch to support the core-router, inter-router data transmissions. In these cases, we further add bypass links to the virtual channels of the inteterposer switch and network interface to avoid buffer delays when packets bypass the router. In addition to a conventional router (chiplet router) [15], we implement a power-gating controller to power off unused ports and crossbars, reducing static power consumption. Moreover, the link controller is deployed to control the connection between the chiplet router and different links, and turn on/off link switches of adaptable links (discussed in the next section).

C. Adaptable Link

The potential for subNoC size and connections to dynamically change require a variety of link connections between different types of adaptable routers. For example, in an 8 × 8 Adapt-NoC, each row/column requires at least 28 (i.e. C_8^2) bi-directional links to cover all possible sub-NoC connections, thus exhausting limited wiring resources. We propose an adaptable link design to provide the sufficient amount of links for multiple subNoCs. The adaptable link design can be dynamically segmented to several short wires of different length. Fig. 4 shows the proposed adaptable link design, which comprises a number of tri-state transistors [16]. The goal of segmenting a wire is to avoid signal interference when multiple signals are propagating on a single wire. The tri-state transistor can cut off the signal propagation between two routers to avoid signal interference. Specifically, the link switch (tri-state transistor) is controlled by the link controller of the router. Upon receiving the off signal (1-bit) from the link controller, the transistor will be disconnected from GND and V_{vdd}, thus terminating the data transmission between the routers. The on signal reconnects the transistor with the GND and V_{vdd}. By doing so, the tri-state transistor resumes the data transmission between the



(a) Cmesh in a 4 × 4 SubNoC (b) Torus in a 4 × 4 SubNoC (c) Logic diagram of the tree connection (d) Tree in a 4 × 4 SubNoC

Fig. 5. SubNoC topology examples: (a)4 × 4 cmesh, (b) 4 × 4 torus, (c)logic diagram of the tree connection, and (d) 4 × 4 tree.

two routers and functions as a link repeater. It should be noted that the link controller is not on the critical path of data transmission.

D. SubNoC Formation

A straightforward deployment or combination of subNoC topologies could incur flow control and routing issues, which in turn affects NoC performance. This requires a synergy of subNoC architecture, flow control and routing algorithm to take full benefits of the Adapt-NoC. In this section, we select cmesh, torus and tree as case studies in a 4×4 allocated subNoC, to illustrate the methodology of forming different sub-NoC topologies.

1) CMesh (Concentrated Mesh): Concentration is to connect multiple nodes to a single router, which can effectively reduce the network diameter by using fewer number of routers. Previous designs [17], [18] relied on high-radix routers or external arbitration logic to realize the network concentration. However, these designs are difficult to implement in the Adapt-NoC due to insufficient router radix and inflexibility of external arbitration (fixed grant and request signals). Therefore, we propose to implement the concentration by utilizing the interposer switch in the Adapt-NoC. Specifically, Fig.5(a) shows a 2×2 Cmesh topology, where four cmesh routers are connected to sixteen nodes. Each cmesh router (i.e. node D) connects to four nodes (A,B,C and itself) in a concentrated region. Within each region, we first connect the injection ports of nodes B and C, which are adjacent to node D, to the interposer switch of node D using adaptable links. Since interposer switch of node D has been fully connected by nodes B and C, node A cannot directly connect to node D. In this case, node A connects to node D through the interposer switch of node C. When packet contention happens between nodes A and C, the node A buffers the packets to the interposer switch of node C. This avoids fixed grant and request signals required by the external arbitration [18]. We use the cmesh to demonstrate the capability of Adapt-NoC to reduce network latency (reduced diameter) and power consumption (fewer number of routers).

2) Torus: Express links [19] are used to connect non-adjacent routers, which allows packets to bypass the intermediate routers. Torus is an example of deploying express links to the mesh topology, where peripheral routers are connected by multiple wrap-around links, horizontally and vertically. The wrap-around links not only reduce the network diameter but also increase the number of links across the network (bi-section bandwidth). In the Adapt-NoC, the adaptable links are segmented to desired length and connect the peripheral routers of the allocated subNoC. The torus topology is used to demonstrate the capability of the Adpat-NoC to simultaneously improve network latency (reduced diameter) and throughput (improved bi-section bandwidth), as required by the applications.

3) Tree: The heavy reply traffic (e.g. one-to-many traffic) from the memory controller (MC) has been identified as a bottleneck of NoC performance in the throughput processor [20], [21]. The intensive reply traffic results in packet congestion at the injection port which significantly increases the queuing latency. Such undesirable congestion results from insufficient injection bandwidth and poor load balance in grid-like topologies, such as the mesh topology. We, therefore, explore the use of Adapt-NoC to configure a tree topology as a reply network, providing high injection bandwidth at the MC (i.e. root node) and balanced load distribution. Specifically, we maximize the fanout of the root router (i.e. MC) to provide high injection bandwidth, where the reply packets from root router are directly injected to the input buffers of the intermediate routers. Furthermore, the root and intermediate routers are connected with their downstream routers, vertically and horizontally to evenly distribute the reply traffic. Fig. 5(c,d) illustrates the logical and physical connections of a fully connected tree topology. As shown in Fig. 5(d), the root, intermediate and leaf routers are fully connected by a set of links (mesh and adaptable links), horizontally and vertically. Such link connections (5(d)) are designed to couple with dimensional routing (generic routing algorithm). Consequently, the reply traffic from the MC can be delivered to all routers within two hops in the illustrated example. We note that the request traffic still goes through the mesh topology. This implementation could demonstrate the capability of Adapt-NoC to efficiently handle one-to-many traffic patterns in the throughput processors.

Tree Scalability: While the example (Fig. 5(d)) shows a fully connected tree topology, the tree suffers from scalability issues when the subNoC size increases. In the case of large subNoC size, we follow the same design principle to maximize injection bandwidth and distribute reply packets. As a result, we still maximize the fanout of root router, but connect root and intermediate routers with their downstream routers at an evenly-spaced distance in each row/column.

4) Possible subNoC topologies: The subNoC formation methodology can be generalized to more topologies, including different configurations of network concentration, express link placements, and combined topologies. For example, the torus (Fig. 5(b)) and tree (Fig. 5(d)) could be combined together to simultaneously optimize both request and reply networks for memory-intensive applications. Moreover, the wrap-around torus links can be segmented to several short express links to bypass routers. While a number of topologies can be generalized, in this work we concentrate on four popular topologies — mesh, cmesh, torus and tree — to demonstrate the performance and energy benefits of simultaneously deploying different subNoC topologies.

IV. Implementation Details

In this section, we specifically discuss the implementation details of subNoC partitioning and configuration, memory controller implementation, and deadlock avoidance in the Adapt-NoC.

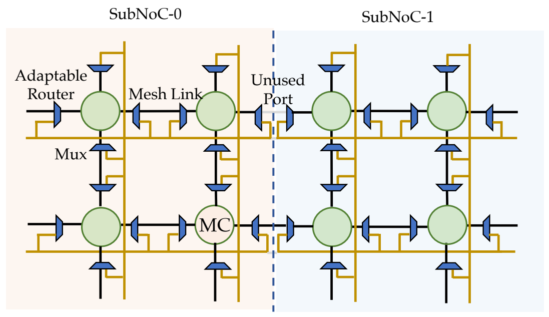


Fig. 6. Memory Controller (MC) connection between subNoCs.

A. Dynamic SubNoC Partitioning and Configuration

The nature of dynamic subNoC allocation is to allocate a set of cache, memory, core and NoC fabric within a region of manycore architecture. Significant research efforts have been devoted to allocating cache, memory and core within a region by modifying cache coloring, page replacement, and mapping policy [11], thereby taking the full advantage of data locality and mitigating inter-application interference. The remaining challenge of allocating subNoCs is to timely configure the subNoC topology while maintaining the dynamic application mapping. To demonstrate the capability of runtime configuration, we studied the timing overhead of configuring subNoCs. The configuration overhead consists of three parts, draining all in-flight packets, powering on router and setting up router connections. We conducted an experiment to calculate the worst-case timing overhead of draining packets in a 32-core system. We found, in the worst-case scenario, the subNoC to be taking about 200 cycles to drain all in-flight packets due to the intensive traffic of rodinia benchmark. Turning on a powered-off router (12 cycles/hop) and setting up link connections (2 cycles/hop) need to take an additional 250 cycles in the 32-core system. The result showed that Adapt-NoC has great potential to support dynamic configurations.

B. Memory Controller Implementation

The MCs should be available to each subNoC, supporting cache-memory communication. We, therefore, implement one MC to each 2×4 subNoC. While each subNoC is assigned with an independent MC, it may not be sufficient for memory intensive applications. To address this issue, we propose a MC sharing design, in which each MC can be shared by up to four subNoCs. Fig. 6 illustrates the MC sharing between subNoC-0 and subNoC-1. When the application running on subNoC-1 demands additional memory bandwidth, the subNoC-1 connects with the MC of subNoC-0 using the unused ports of the peripheral routers in both subNoCs. Moreover, the interposer switches can also provide inter-subNoC communication.

Memory Controller Scalability: The growing number of MCs could be lagging behind the increase in core count. As a result, each MC has to be shared by multiple subNoCs using the mentioned sharing policy. In this case, eight MCs can be shared by 32 2×4 subNoCs, which account for 256 cores. Furthermore, we can allocate one MC to a larger subNoC size (i.e. 4×4 subNoC), which can scale to 512 cores. This shows the scalability of Adapt-NoC for handling hundreds of cores projected in the future.

C. Deadlock Avoidance

The network deadlock incurs due to improper subNoC switching, protocol deadlock or circular channel dependence. To avoid the misrouting of packets, we do not alter the subNoC size and topology until all in-flight packets were drained. Furthermore, we avoid the protocol deadlock by using multiple virtual networks. Although we use dimensional-order routing for mesh, cmesh and torus, the circular channel dependence is inevitable in each tori. Despite the fact that a number of techniques have been proposed to solve the deadlock in ring and tori [15], [22]–[24], we select the simple yet effective dateline [15] to avoid such circular channel dependence. It should be noted that the prerequisite of forming circular channel dependence is that a given router is connected to at least two other routers. As each MC is only allowed to connect to one router of any subNoC, this precludes the formation of cyclic dependency within any subNoC.

V. Evaluation

We evaluate the proposed architecture under a full system simulation, with the combined use of architecture-level and circuit-level simulators. The cycle-accurate gem5-GPU simulator [25], and GARNET [26] were used for detailed timing simulation of the memory and NoC. We use DSENT [27] to evaluate the power consumption and use a Synopsys Design Compiler with the 45 nm FreePDK45 open cell library to evaluate the area overheads.

To examine the diverse application behavior, we evaluate the performance of our proposed design using both the Parsec and Rodinia applications, in an 8×8 chiplet-based heterogeneous system. We assume that three applications are dynamically mapped to different regions of the heterogeneous manycore system. The Rodinia application is mapped to a region with the mix of 4 CPUs, 4 MCs and 24 GPUs, where each 2×4 subNoC consists of 1 CPU, 1 MC, and 6 GPUS. Each GPU core contains 8-wide SIMD lanes and is equipped with a 64KB private L1 cache and scratch memory. Parsec applications are mapped to 28 CPUs and 4 MCs, where each 2×4 subNoC consists of 7 CPUs and 1 MC. Each CPU core also contains private a 64KB L1 instruction/data cache, and 1 MB shared L2 cache. Each core or MC is attached to a router. The baseline mesh router has 2 pipeline stages (look-ahead routing and speculative optimizations), 2 VCs per virtual network and 4 buffers per VC.

1) **Baseline:** We use a mesh topology as the baseline design, where the chiplets are connected by a passive interposer.
2) **Shortcut [1]:** We extend the concept of short-cut links to chiplet-based NoC designs, where the chiplet routers can be connected by a set of reconfigurable links in the active interposer.
3) **ButterDount [13]:** We use a state-of-the-art active interposer design, where the chiplet routers are connected in a mesh topology, and interposer routers are connected in a Butterdount topology.
4) **Adapt-NoC:** The proposed Adapt-NoC design with multiple disjoint subNoC topologies.

A. Application Category and Mapping Policy

TABLE I
Application Categories.

Categories	Applications
CPU	Blackscholes (BS), Swaptions (SW), X264, Ferret (FR), Bodytrack (BT), Canneal (CA)
GPU-CPU	Guassian (GA), Breath-First-Search (BFS), Needleman-Wunsch (NW)
GPU	Kmeans (KM), Back-propagation (BP), Heart-Wall (HW)

To cover a wide range of application behavior, we selected a number of applications from the Parsec and Rodinia suites, as shown in Table I. We profiled all applications and grouped them under three different categories, namely CPU, GPU-CPU, and GPU. CPU applications only include inter-CPU communications; GPU-CPU

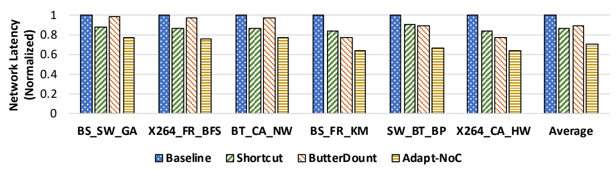


Fig. 7. Network latency analysis of mixed workloads, normalized to baseline.

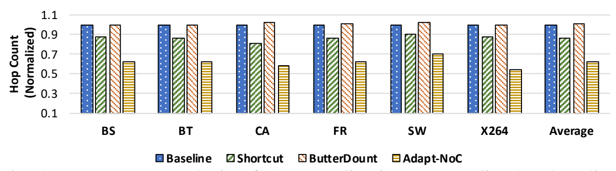


Fig. 8. Hop count analysis of CPU applications, normalized to baseline.

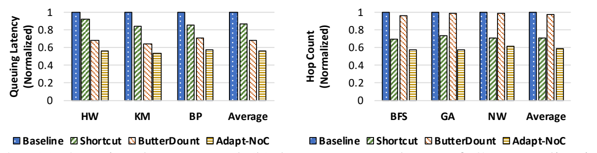


Fig. 9. (a) Queuing latency, and (b) hop count analyses of GPU applications, normalized to baseline.

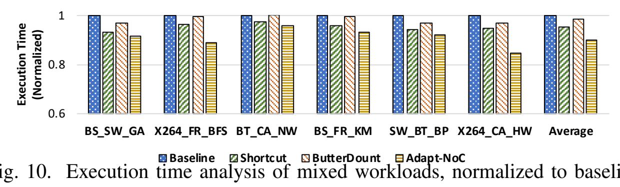


Fig. 10. Execution time analysis of mixed workloads, normalized to baseline.

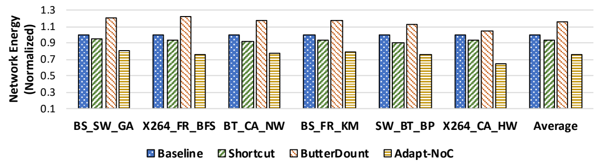


Fig. 11. Energy analysis of mixed workloads, normalized to baseline.

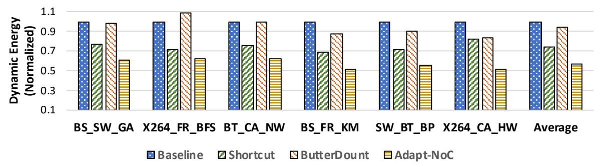


Fig. 12. Dynamic energy analysis of mixed workloads, normalized to baseline.

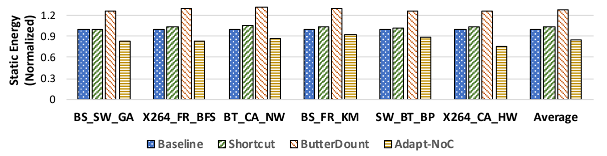


Fig. 13. Static energy analysis of mixed workloads, normalized to baseline.

applications include a large portion of GPU and CPU co-executions; GPU applications mainly consist of GPU executions and demand high memory bandwidth. We use cmesh, torus, and a hybrid topology (mesh + tree) to support these different communication demands, thereby examining the benefits of allocating different subNoC topologies. We mixed these applications as different workloads, where PARSEC applications are mapped to a 4×4 subNoC, and Rodinia applications are mapped to a 4×8 subNoC.

B. Performance Analysis

1) Network Latency Analysis: Fig. 7 shows the network latency analysis of the Adapt-NoC as compared to baseline, shortcut and butterdount. The Adapt-NoC reduces the network latency of mixed workloads by 31%, 16%, and 19% on average, when compared to the baseline, shortcut and butterdount. The reduced latency results from the hop count and queuing latency reductions in both GPU and CPU applications. Specifically, Fig. 8 shows the hop count analysis of CPU applications. The Adapt-NoC reduces the average hop count of packets by 39% and 24% as compared to baseline and shortcut. While the Adapt-NoC achieves significant hop count reduction in all CPU applications, some of CPU applications, Canneal and Swaptions, only convert such hop reduction to 18-20% network latency reduction due to their traffic-intensive application phases. The butterdount is designed for handling intensive memory traffic, and therefore it has limited latency reduction because of the sparse memory traffic in CPU applications. We further studied the queuing latency and hop count reductions of GPU applications as shown in Fig. 9. Both the Adapt-NoC and butterdount efficiently alleviate the packet congestion at injection port and reduce the queuing latency in memory-intensive applications (HW, KM, and BP), as they provide additional network bandwidth to improve the poor load balance of mesh topology. The enhanced load balance eventually results in 33% and 45% queuing latency reductions as compared to baseline. For those applications involve CPU-GPU co-executions, we observed that the Adapt-NoC simultaneously optimizes load balancing and reduce hop count. This turns to 13% and 41% of hop count reduction when compared to shortcut and the baseline.

2) Execution Time Analysis: Fig. 10 shows the execution time analysis of the Adapt-NoC. Overall, the Adapt-NoC reduces the execution time by 10%, 5%, and 8% on average, as compared to baseline, shortcut, and butterdount. Specifically, the Adapt-NoC reduces the execution time of CPU applications up to 14% (x264) and 8% on average, as compared to baseline. Moreover, the Adapt-NoC reduces the execution time of GPU applications up to 25% (Heartwall) and 12% on average, as compared to baseline.

3) Energy Analysis: Fig. 11 shows the normalized energy analysis of the Adapt-NoC. The Adapt-NoC improves overall energy savings by 24%, 18% and 40%, when compared to baseline, shortcut and butterdount on average. The energy benefits of the Adapt-NoC are from the reduced execution time, reduced hop count and the use of power-gating. Specifically, Fig. 12 shows the dynamic energy analysis, in which the Adapt-NoC reduces the dynamic energy by 42%, 17%, 37% as compared to baseline, shortcut and butterdount. The dynamic energy saving benefits from the reduced hop count of packets (39% and 43% in CPU and GPU applications), because packets traverse fewer number of routers and reduce the switching activity of those skipped routers. Fig. 13 shows the static energy analysis, in which the Adapt-NoC reduces the static energy by 15%, 18%, 42% on average, as compared with baseline, shortcut, and butterdount. The static energy saving is from powering off the unused routers and ports in peripheral routers. The overall energy savings efficiently compensate the energy overhead caused by additional adaptable links (3.9 mW/link), which ultimately converts to 24% energy saving when compared to baseline.

C. Area Overhead

We evaluate the area overhead through Synopsis Design Vision using 45 nm and 65nm technologies for chiplet and interposer respectively. The baseline chiplet router consists of a crossbar of 17806 um^2, a switch allocator of 4589 um^2, a virtual channel allocator of 9066 um^2, and buffers of 98740 um^2. As a result, the overall chiplet NoCs account for 8.3 mm^2 area. The Adapt-NoC requires 0.3 mm^2 additional area in chiplet, which is much smaller than shortcut and butterdount (1.66 mm^2) as they require additional router radix in chiplet routers. We further evaluate the area of active interposer design, which shows that Adapt-NoC and butterdount requires 4.41 mm^2 and 5.83 mm^2 in the interposer. The decomposed low-radix switches require 24% less area than high-radix interposer router in the interposer. Both of Adapt-NoC and butterdount consume less than 2% area overhead of interposer, which has minimal side-effects on the interposer yield.

D. Timing Analysis and Optimization

While the chiplet router radix is reduced in this work, the additional logic delay of muxes and complicated crossbar switch could have a negative impact on the chiplet router timing. We use Synopsys to verify the critical delays of adaptable router. The simulation result shows that route computation (RC), virtual channel allocation (VA), switch allocation (SA), and switch traversal (ST) of a conventional 5×5 chiplet router take 164 ps, 370 ps, 243 ps, and 256 ps respectively. Each mux requires 102 ps critical delay. Therefore, we deploy two optimizations to solve the potential timing issues. First, we merge mux logic of input and output ports into RC and ST stages respectively. As the router frequency is limited by the delay dominant router stage (VA), the delays of merged RC and ST (266 ps, 350 ps) are still shorter than that of VA stage (370 ps). This satisfies the timing of router, thus does not affect router frequency. Second, we optimize VA delays by reducing the number of input ports at the arbitration stage, as the number of input ports mainly determines delays of arbitration at VA stage [28]. To keep the input port number constant in the VA stage, we connect the interposer switch with injection port of chiplet router by a 3:1 mux. Thus, the optimized VA stage would not require additional delays.

E. Physical Constraints of Interposer

As the adaptable routers and links require additional routing resources (wire and microbump), in this section we verify the proposed design within the interposer wiring and micro-bump constraints.

Wiring Constraint: We assume four metal layers in the silicon interposer, and the wire width is 1 μm, and wire height is 1.5 μm, wire pitch is 2 μm. Typically, half of the wiring resources are reserved for designing interconnect. For a 2mm tile, the interposer can provide up to eight 128-bit bi-directional links per tile direction, which satisfies wiring demands of Adapt-NoC (two 128-bit bi-directional links per tile direction).

Micro-bump Constraint: We assume 36 μm micro-bump pitch in this work, and each adaptable router requires seven 128-bit bi-directional links between interposer and chiplet. This consumes about 33% of μbump area of an 18×24 mm^2 interposer, which will be reduced to 1-6% μbump area when the pitch reduces to the range of 5-15μm.

VI. Conclusions

In this paper, we propose *Adapt-NoC*, a versatile and flexible NoC architecture for chiplet-based manycore architectures, consisting of adaptable routers and links. Adapt-NoC can dynamically allocate disjoint regions of the NoC, called subNoCs, with the objective of supporting concurrent communication requirements of diverse applications. The adaptable routers and links of each subNoC are capable of providing various topologies such as mesh, cmesh, torus, and tree, supporting various traffic patterns (e.g. one-to-many) with different latency and bandwidth demands. Full system simulation shows that subNoC topologies can significantly improve application performance by reducing 31% overall latency, 10% execution time, and 24% energy when compared to prior NoC designs.

VII. Acknowledgment

This research was partially supported by NSF grants CCF-1513923, CCF-1547036 and CCF-1702980. We sincerely thank the anonymous reviewers for their excellent feedback.

References

[1] U. Ogras and R. Marculescu. Application-specific network-on-chip architecture customization via long-range link insertion. In *Proc. of ICCAD*, pages 246–253. IEEE, 2005.
[2] M. Stensgaard et al. Renoc: A network-on-chip architecture with reconfigurable topology. In *Proc. of NoCs*, pages 55–64. IEEE, 2008.
[3] M. Modarressi et al. Application-aware topology reconfiguration for on-chip networks. *IEEE Transactions on VLSI Systems*, 19(11), 2010.
[4] M. Kim, J. Davis, M. Oskin, and T. Austin. Polymorphic on-chip networks. In *Proc. of ISCA*, pages 101–112. IEEE, 2008.
[5] O. Chen et al. Smart: a single-cycle reconfigurable noc for soc applications. In *Proc. of the DATE*, pages 338–343. IEEE, 2013.
[6] Mohammad Al F. et al. Configurable links for runtime adaptive on-chip communication. In *Proc. of DATE*, pages 256–261. IEEE, 2009.
[7] P. Ritesh, D. Reetuparna, and B. Valeria. Power-aware nocs through routing and topology reconfiguration. In *Proc. of DAC*, June 2014.
[8] B. Beckmann and D. Wood. Managing wire delay in large chip-multiprocessor caches. In *Proc. of MICRO*, pages 319–330. IEEE, 2004.
[9] C. Kim et al. An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches. In *Proc. of ASPLOS*, volume 37, pages 211–222. ACM, 2002.
[10] F. Triviño et al. Virtualizing network-on-chip resources in chip-multiprocessors. *Microprocessors and Microsystems*, 35(2):230–245, 2011.
[11] R. Das et al. Application-to-core mapping policies to reduce memory system interference in multi-core systems. In *Proc. of HPCA*, pages 107–118. IEEE, 2013.
[12] NE. Jerger, A. Kannan, Z. Li, and GH. Loh. Noc architectures for silicon interposer systems. In *Proc. of MICRO*, pages 458–470. IEEE, 2014.
[13] A. Kannan et al. Enabling interposer-based disintegration of multi-core processors. In *Proc. of MICRO*, pages 546–558. IEEE, 2015.
[14] D. Stow et al. Cost-effective design of scalable high-performance systems using active and passive interposers. In *Proc. of ICCAD*, pages 728–735. IEEE, 2017.
[15] W. Dally and B. Towles. *Principles and practices of interconnection networks*. Elsevier, 2004.
[16] A. Kodi, A. Sarathy, and A. Louri. ideal: Inter-router dual-function energy and area-efficient links for network-on-chip (noc) architectures. In *Proc. of ISCA*, pages 241–250, 2008.
[17] J. Balfour and W. Dally. Design tradeoffs for tiled cmp on-chip networks. In *Proc. of ICS*, pages 390–401. ACM, 2006.
[18] P. Kumar et al. Exploring concentration and channel slicing in on-chip network router. In *Proc. of NoCs*, pages 276–285. IEEE, 2009.
[19] J. Kim et al. Flattened butterfly topology for on-chip networks. In *Proc. of MICRO*, pages 172–182. IEEE Computer Society, 2007.
[20] A. Bakhoda et al. Throughput-effective on-chip networks for manycore accelerators. In *Proc. of MICRO*, pages 421–432. IEEE, 2010.
[21] H. Jang et al. Bandwidth-efficient on-chip interconnect designs for gpgpus. In *Proc. of DAC*, page 9. ACM, 2015.
[22] C. Carrion et al. A flow control mechanism to avoid message deadlock in k-ary n-cube networks. In *Proc. of HiPC*, pages 322–329. IEEE, 1997.
[23] L. Chen and T. Pinkston. Worm-bubble flow control. In *Proc. of HPCA*, pages 366–377. IEEE, 2013.
[24] S. Ma et al. Leaving one slot empty: Flit bubble flow control for torus cache-coherent nocs. *IEEE Transactions on Computers*, 64(3):763–777, 2013.
[25] N. Binkert and et al. The gem5 simulator. In *ACM SIGARCH Computer Architecture News*, May 2011.
[26] N. Agarwal et al. Garnet: A detailed on-chip network model inside a full-system simulator. In *Proc. of ISPASS*, pages 33–42. IEEE, 2009.
[27] C. Sun and et al. Dsent a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling. In *Proc. of NoCs*, May 2012.
[28] LS Peh and W. Dally. A delay model and speculative architecture for pipelined routers. In *Proc. of the HPCA*, pages 255–266. IEEE, 2001.