Asymmetric NoC Architectures for GPU Systems

Amir Kavyan Ziabari[†], José L. Abellán [‡], Yenai Ma[♮], Ajay Joshi[♮], David Kaeli[†]

[†]Department of Electrical and Computer Engineering
Northeastern University
{aziabari,kaeli}@ece.neu.edu

[‡]Computer Science Dept.
Universidad Católica San Antonio de Murcia
jlabellan@ucam.edu

[♮]Department of Electrical and Computer Engineering
Boston University
{yenai,joshi}@bu.edu

ABSTRACT

While both Chip MultiProcessors (CMPs) and Graphics Processing Units (GPUs) are many-core systems, they exhibit different memory access patterns. CMPs execute threads in parallel, where threads communicate and synchronize through the memory hierarchy (without any coalescing). GPUs on the other hand execute a large number of independent thread blocks and their accesses to memory are frequent and coalesced, resulting in a completely different access pattern.

NoC designs for GPUs have not been extensively explored. In this paper, we first evaluate several NoC designs for GPUs to determine the most power/performance efficient NoCs. To improve NoC energy efficiency, we explore an asymmetric NoC design tailored for a GPU's memory access pattern, providing one network for L1-to-L2 communication and a second for L2-to-L1 traffic. Our analysis shows that an asymmetric multi-network Cmesh provides the most energy-efficient communication fabric for our target GPU system.

Categories and Subject Descriptors

B.4.3 [**Input/Output and Data Communications**]: Interconnections (subsystems)—*physical structures, topology*; C.1.2 [**Processor Architectures**]: Multiple Data Stream Architectures—*Single-Instruction Stream, Multiple-Data Processors (SIMD)*

1. INTRODUCTION

Graphics Processing Units (GPUs) have been used for general purpose computation for more than a decade [16]. GPU computing has made it possible to exploit massive degrees of parallelism. GPU programming frameworks, such as OpenCL [21], support parallelism in GPUs by dividing an application workload into groups of threads (work-groups) that can be independently executed on separate GPU processing elements, known as Compute Units (CUs). General-purpose applications that exhibit parallel behavior can use GPUs effectively. This has encouraged researchers to use GPUs to explore larger data sets [8, 11] and pursue a range of challenging problems in various domains (e.g., chemical-physics [13] and genetics [7]).

The programming model of GPUs involves the loading of a workgroup from memory to a CU through the GPU-to-memory network and the network-on-chip (NoC). The workload will be processed in a single-instruction-multiple-thread (SIMT) fashion, and then written back to the memory through the two networks. Depending on the scale of the workload, these steps will be repeated iteratively until the processing of the work-groups in the workload are complete.

With the expected increase in the number of CUs (with each new generation of GPU systems) and the growth in size of dataset, we need to increase NoC bandwidth to be able to support this execution model. This increase in bandwidth will correspondingly increase the static power (due to an increase in the number of channels) and the dynamic power (due to the increase in channel utilization running larger workloads). Hence, we need to explore NoC designs that provide energy-efficient communication to optimize the energy efficiency of the overall GPU system.

In this paper, we investigate the design of an energy-efficient NoC that is tailored for GPUs and their underlying programming model. To explore the design space of NoC designs for GPUs, we target a contemporary AMD Southern Islands GPU architecture [3] and choose applications with a range of workload intensities from the AMD SDK benchmark suite [1]. We first compare various electrical NoCs, using performance and power metrics, to identify the best topologies for our target GPU system. We present an analysis of the traffic patterns resulting from executing multiple data-parallel applications on the evaluated NoCs. Leveraging our analysis considering both performance and power metrics, we propose an asymmetric NoC design that can achieve comparable performance at reduced power consumption.

2. TARGET SYSTEM AND EVALUATION METHODOLOGY

2.1 AMD Southern Islands GPUs

In this paper, we consider the AMD Radeon HD 7970 architecture from AMD's Southern Islands family as our target system. This GPU system is tailored to provide general purpose computing. It is manufactured in 28 nm technology. The CUs are clocked at 925 MHz [3]. Table 1 provides the details of the AMD Radeon HD 7970 GPU architecture.

The Radeon HD 7970 has 32 CUs. Each CU is able

NOCS'15 September 28–30,2015, Vancouver, BC, Canada

http://dx.doi.org/10.1145/2786572.2786596 .

to execute one independent block of threads (i.e., a work-group) at a time. Work-groups consist of multiple wave-fronts (consisting of 64 threads or work-items). Each CU can execute an entire wave-front in 4 cycles. Each wave-front can start execution on the SIMD pipelines of the compute unit. Execution begins as soon as data becomes available in the memory hierarchy.

Each CU is equipped with a Load-Data-Share (LDS) unit as local memory. CUs are connected individually to the global memory hierarchy through two cache units; a L1 vector data cache (vL1) unit and a read-only L1 scalar data cache (sL1) unit. The global memory hierarchy also consists of L2 cache units. Each L2 cache unit is coupled with only one memory controller that can access a unique memory address range. This distribution of address ranges across the L2 banks reduces the load on any single memory controller, while exploiting spatial locality [3]. We explore the design space of a NoC that connects the scalar and vector L1 data cache units to L2 cache banks.

There are two types of messages that is transmitted between the L1 and L2 cache units in the memory hierarchy. The first type is an 8-byte control message. This control message contains information specific to cache requests (reads, writes, invalidation, etc.), and the address of the destination cache unit. The second type is a 72-byte message, which includes an 8-byte control message and a 64-byte cache line. For any channel with a bandwidth smaller than 72 bytes, the control message will be transferred in a separate cycle first, and the cache line is packeted and transferred immediately in the following cycles. For example, if we use a 32-byte channel to transmit a 72-byte message, then we will use the first cycle to transmit the 8-byte control message, and then the next 2 cycles to transmit the 64-byte cache line.

2.2 Evaluation Methodology

To evaluate the NoC for a GPU system, we used the Multi2sim 4.2 simulation framework [22]. We evaluated the crossbar, mesh, concentrated mesh (Cmesh), butterfly, concentrated crossbar, and Clos topologies. Our NoC designs utilize state-of-the-art single-cycle routers, with the extra required lookahead signals embedded in the 8-byte control messages (see Section 2.1), which are transferred to the cache units [20]. The routers in the designs operate at 1 GHz frequency.

Table 2 provides the set of applications used in our evaluation. These applications are taken from the AMD APP SDK [1]. The benchmarks represent a wide range of workload sizes, bandwidth demands, and memory intensities (*URNG* is a compute-intensive application, *CONV* and *DCT* have very large workloads and high memory intensity. *FWT* includes irregular access patterns).

The power of the NoC is estimated using a detailed transistor-level circuit modeling, the physical layout, the flow control mechanism and network traffic workloads. The wires in the crossbar, butterfly and clos topologies are designed to be implemented in the global metal layers using pipelining and repeater insertion in 28 nm Predictive Technology Models [2]. All of inter-router channels in all of the mesh-based topologies are implemented in the semi-global metal layers using standard repeater wires. The power dissipated in the SRAM array and crossbar of the routers is calculated by adapting the methodology described in [15] and [24], respectively.

Table 1: **AMD Radeon HD 7970 GPU specification.**

Processor Cores		Memory System	
Fabrication process	28nm	Size of L1 Vector Cache	16K
Clock Frequency	925	Size of L1 Scalar Cache	16K
Compute Units	32	L2 Caches/Mem. Cntrls.	6
MSHR per Compute Unit	16	Block Size	64B
SIMD Width	16	Size of L2 Cache	128K
Threads per Core	256	Memory Page Size	4K
Wavefront Size	64	LDS Size	64K

Table 2: **Workloads from the AMD APP SDK.**

Abbreviation	Application
CONV	Simple Convolution
DCT	Discrete Cosine Transforms
DWTHAAR	One-dimensional Haar Wavelet Transform
FWSHALL	Floyd-Warshall Shortest Path Calculation
FWT	Fast Walsh Transfrom
HIST	Histogram
MATMUL	Matrix Multiplication
MT	Matrix Transpose
RED	Reduction
RG	Recursive Gaussian Filter
SOBEL	Sobel Edge Detection Algorithm
URNG	Uniform Random Noise Generator

3. SYMMETRIC NOC DESIGNS FOR GPUS

In this section we explore the trade-offs associated with various NoC topologies that can be used for interconnection between the cache units in the GPU memory hierarchy. We analyze a crossbar, a mesh, a concentrated mesh (Cmesh), a Butterfly, a concentrated crossbar, and a Clos network (see Figure 1), to cover the entire spectrum from high-diameter, low-radix networks, to low-diameter, high-radix networks at the other end. Table 3 includes the key architecture parameters of these NoC topologies. The bisectional bandwidth is matched across all topologies for a fair comparison of performance of network topologies.

The first topology considered is the well-known global crossbar. The global crossbar is generally considered to be the most efficient network topology in terms of programmability. A 46×46 global crossbar is considered to provide all-to-all connectivity between the 40 L1 (8 scalar and 32 vector) cache units and 6 L2 cache units. A crossbar provides non-blocking connectivity between each pair of nodes, but the crossbar design is very challenging to layout. Since a crossbar requires a large number of global buses across the length of the chip, this can lead to significant power consumption in the wires. Crossbars also require global arbitration, which can add significant latency and power dissipation. We place the arbiter in the center of the chip to minimize the arbitration overhead [18].

Figure 1(a) shows a 2D 8×5 mesh network. In Southern Island GPUs, each CU is connected to a single vector cache unit and a group of four CUs share a single scalar unit. The number of nodes along each dimension of the mesh is chosen based on this adjacency of the cache units. The diagonal placement [19] is considered for the L2 cache banks to minimize the average packet latency and request-response variance. The mesh design utilizes the X-Y routing and distributed flow control [23]. The high hop count in the mesh, however, results in long latencies and high energy consumption in both routers and channels.

The hop count of a mesh network could be reduced if

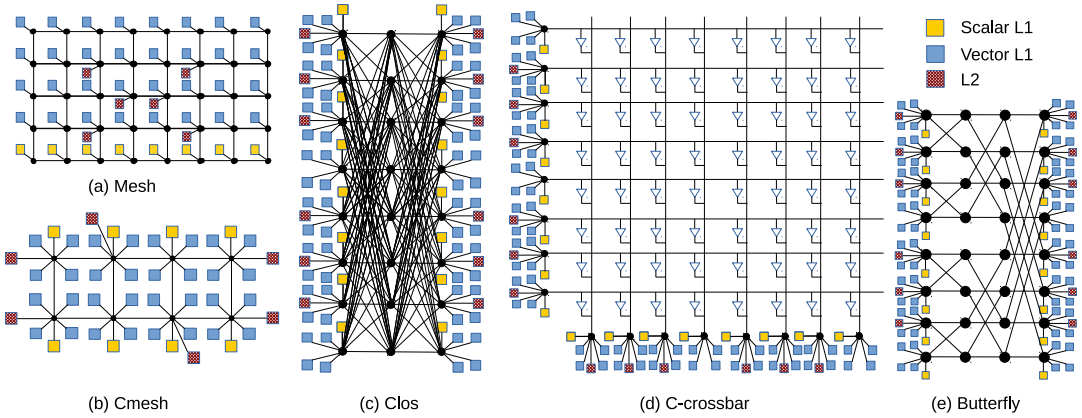


Figure 1: **Topologies of the NoCs evalutated for the 32-CU GPU**: (a) a 2D 8x5 mesh, (b) a concentrated mesh (Cmesh) with 5x and 6x concentration, (c) a 6-ary, 3-stage, Clos network with 8 intermediate routers, (d) a concentrated tri-state global crossbar (C-crossbar), (e) a 6-ary 4-fly butterfly. In all of our figures: dots = routers, triangles = tristate buffers. In (a) and (b) inter-dot lines = bi-direction channels. In (c), (d) and (e) inter-dot lines = uni-directional channels. In (c), (d) and (e) the source and destination nodes are separated for the clarity, otherwise the number of cache units are the same as (a) and (b). In (c) nodes are connected to the concentration switch by bi-directional channels.

Table 3: *Network Configuration* - Networks are sized to support two types of messages that create the GPU traffic. N_C = number of channels, b_C = bits/channel, N_{BC} = number of bisection channels, N_R = number of routers, H = number of hops along data path, T_R = router latency, T_{GR} = average latency of global connections (if used), T_C = channel latency, T_{TC} = latency from cache units to the concentration switch, T_S = serialization latency, T_0 = zero-load latency. Both types of messages are considered in the latency calculations. The latency values are separated by ",".

	Channels				Routers		Latency						
Topology	N_C	b_C	N_{BC}	$N_{BC} \cdot b_C$	N_R	Common Radix	H	T_R	$T_{GR}(avg)$	T_C	T_{TC}	T_S	T_0
Butterfly	48	32×8	8	2048	32	2×2	4	1	n/a	1	1, 3	0, 2	9, 19
Clos	128	8×8	64	4096	24	8×8	3	1	2	n/a	1, 9	0, 8	8, 36
C-crossbar	8	32×8	8	2048	16	8×8	3	1	4	n/a	1, 3	0,2	8, 15
Cmesh	18	72×8	4	2304	8	9×9	1 – 5	1	n/a	1	1	0	3 – 11
Crossbar	46	8×8	46	2944	1	46×46	1	n/a	4	n/a	0	0,8	5, 13
Mesh	146	32×8	10	2560	40	5×5	2 – 12	1	n/a	1	0	0,2	5 – 51

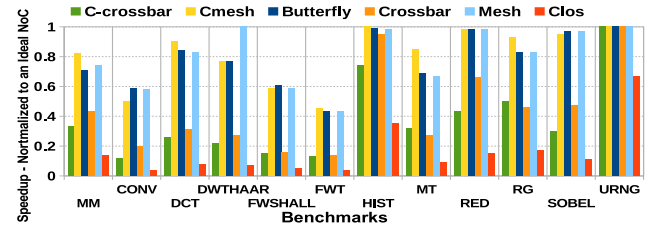
concentration is used to combine the traffic of multiple cache units at one router and reduce the diameter of the network [6]. This reduces load imbalance across the channels without increasing wiring complexity. Figure 1(b) illustrates a Cmesh for the GPU architecture. Here four neighboring CUs in the AMD Radeon 7970 HD GPU layout are concentrated on a single concentration switch [17].

Concentrated crossbars (C-crossbar) and Clos networks are intermediate points between the conventional high-radix, low-diameter, crossbar topology and the low-radix, high-diameter, mesh topology. Figure 1(c) and 1(d) illustrate the two topologies that utilize the concentration of the neighboring cache units to reduce the required wiring. However, this leads to high-radix routers. Both designs have lower hop counts in comparison to mesh topologies, but require longer point-to-point channels. The Clos network is reconfigurably non-blocking. Similar to the crossbar, for C-crossbar design, we pipelined point-to-point channels to improve throughput and placed the arbiter in the center of the chip.

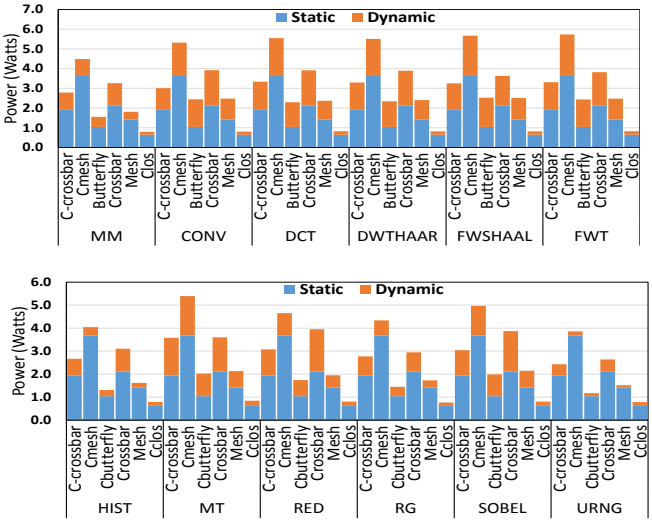
We also considered a conventional 6-ary 4-fly butterfly in this study, as shown in Figure 1(e). A conventional butterfly has no path diversity (and therefore no load-balancing in the network) as compared to a flattened butterfly [12], but requires low-radix routers. We favor the conventional butterfly over a flattened butterfly, since one side of each communication is typically a L2 cache unit. Since the L2 cache units are partitioned the network load becomes balanced (see Section 4 for more details).

Figure 2 presents a comparison of the various NoC designs for the targeted GPU architecture in terms of performance, power and energy-delay-product (EDP) metrics. We normalized the performance of the GPU with different NoCs for each application using an ideal NoC with a fixed 3-cycle latency. The 3-cycle latency is the lowest latency we can have in our design, which is the minimum zero-load latency of the Cmesh. As shown in Figure 2(a), the mesh, Cmesh and butterfly NoCs exhibit comparable performance across all the applications. The highest performance is achieved by a Cmesh network. On average, the performance of the Cmesh is 0.71% the ideal network, while the mesh and butterfly both achieve a performance (on average) equal to 0.69% of an ideal network.

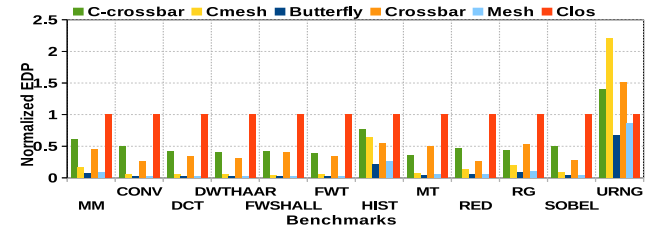
As mentioned in Section 2, wavefronts of a work-group that reside in a single CU can start their execution as soon as their data becomes available through the memory hierarchy. The latency associated with retrieving the required data will determine the start of execution of the wavefront. Since L2 caches in the GPU architecture are partitioned, the required



(a) **Performance Comparison of various NoC designs**



(b) **Power breakdown of various NoC design**



(c) **Energy-per-Delay Product of various NoC Designs**

Figure 2: **Evaluation of Performance and Power for a GPU with various NoC designs using AMD APP SDK benchmarks. The performance is normalized to an ideal network with 3-cycle latency.**

data for the wavefronts may reside in a number of L2 units (and not just a single). Any network that combines the traffic of L1s toward L2s (and vice versa) can hurt the performance of, not a single, but multiple CUs. Because the mesh, Cmesh, and butterfly NoCs provide diverse paths between L1s and individual L2s, they lead to higher utilization of the CU. On the other hand, the Crossbar, Clos and C-Crossbar, which at some point combine the traffic to/from the L2s, achieve lower performance.

Figure 2(b) illustrates the power breakdown of the different NoC topologies. Among these topologies, Cmesh has the highest static and dynamic power consumption. The highest static power can be attributed to the fact that Cmesh has the widest channel width (72 bytes) and it has high-radix routers. The power dissipated by the wires in the 72-byte links is the source of this high power dissipation. At the same time, the power consumed by the router's crossbar is large since power is directly correlated with the flit size, which in this case is 72 bytes. Clos has the lowest performance in comparison with other topologies (16% of an ideal network – Figure 2(a)), and hence it exhibits the lowest dynamic power consumption compared to all other topologies. While Clos uses routers with the same radix as a C-crossbar, it is designed with the lowest channel width (8 Bytes). Moreover, in a Clos layout, we placed all the middle-stage routers together in the center of the chip to provide shorter global channels, which in turn results in lower power consumption, as suggested in [10].

Figure 2(c) shows an overall comparison of all the topologies using the EDP metric, which jointly accounts for changes in performance and energy. We have normalized the EDP of each design based on the EDP of the Clos, which has the highest EDP among all the network topologies which we evaluated. As shown in the figure, a butterfly network achieves the lowest EDP across all the applications (0.10% of the EDP of the Clos, averaged across all the benchmarks). The mesh and Cmesh are the next best topologies that exhibit low EDP, 0.12% and 0.31% the EDP of the Clos, respectively. In the next section, we propose asymmetric NoC architectures (designed based on application analysis) and we use the butterfly, mesh and Cmesh networks topologies for evaluating the asymmetric NoCs.

4. ASYMMETRIC NOC DESIGN FOR GPU

In the previous section we compared the use of different NoC topologies for our target system while running a variety of the applications. In this section, we analyze the trends in OpenCL applications and optimize the NoC design based on the traffic pattern exhibited by these applications.

As mentioned in Section 1, the OpenCL data parallel programming model achieves scalable performance through independent execution of work-groups. Applications written in OpenCL exhibit unique memory access patterns, and hence the NoC can be designed based on these expected memory access patterns to maximize the NoC energy efficiency.

The amount of communication between cores (L1-to-L1) in a GPU is limited. This limited communication includes interaction due to false-sharing (where two cores share different sub-blocks of the cache-lines), and limited cache coherency signals (if any), which are enforced by the AMD's *relaxed* consistency model [3, 22]. At the same time, the latency of L1-to-L2 traffic is typically not critical, as the GPU programming model enforces that no other CU should be stalled awaiting completion of a write-back transfer. Moreover, GPU workloads exhibit lighter traffic for L1-to-L2 accesses due to architectural enhancements such as coalescing. Figure 3 breaks down the percentage of different types of communication between the cache units (L1-to-L1, L1-to-L2, and L2-to-L1) for our selected applications.

We consider the knowledge of the expected network traffic, and the results presented in Figure 3, to enhance our NoC design. As a first step for shaping our proposed topologies, we consider two separate networks, one for each direction of the communication, i.e., L1 to L2 and L2 to L1. We consider this strategy for two reasons: 1) We can improve the performance of both networks in terms of latency by restricting the traffic to one direction 2) We can design asymmetric networks optimized for the direction with heavier load and higher delay sensitivity.

As a second step, we eliminate the paths between the L1 cache units and replace any L1-to-L1 communication with a

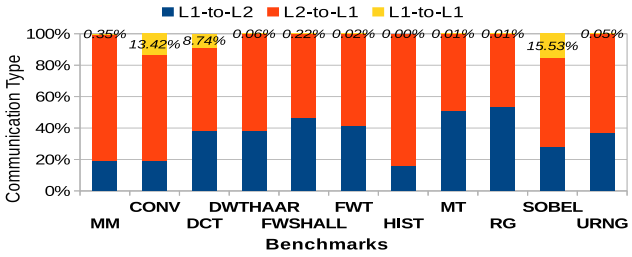


Figure 3: Breakdown of the different types of communications between cache units.

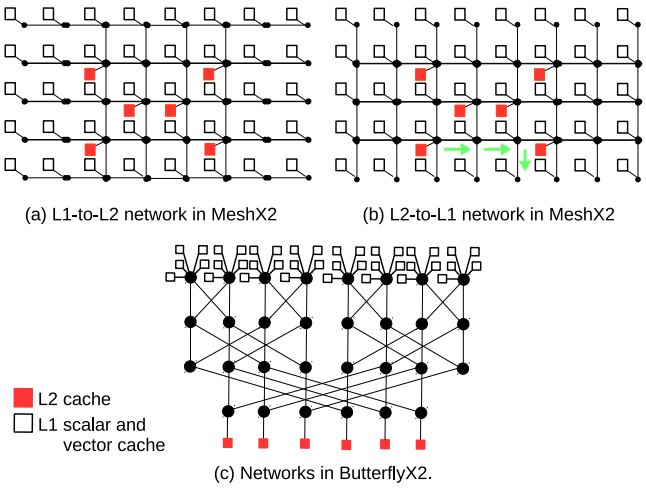


Figure 4: (a) The L1-to-L2 network for MeshX2, (b) The L2-to-L1 network for MeshX2. The use of X-Y routing leads to several unused links. These links were removed to reduce the static power consumption and switch radices, (c) The network for both direction in a ButterflyX2. Since the number of L2s is limited to 6, the design uses two less routers and wider bisection channels than the conventional butterfly.

L1-to-L2 transfer followed by a L2-to-L1 transfer. While this slightly increases the traffic between the L1-to-L2 and the L2-to-L1, it allows us to reduce the power consumption in the network. This reduction in power is due to lowering of the number of physical links and router radix. For example, in the L1-to-L2 network in Cmesh, 5 L1s and one L2 (in 6 out of 8 routers) are connected to each concentration switch unidirectionally instead of bidirectionally, which leads to a reduction in the number of wires (32×8 wires are eliminated for *each*. Links also become unidirectional – 1280 wires per concentration switch). We also see a reduction in the router radix, from 10×10 (or 9×9) to 8×4 (or 7×4).

Based on these considerations, we propose using parallel networks, one for L1-to-L2 communication and one for L2-to-L1 communication. We eliminate the links that create paths between the L1 units. The new parallel designs for the mesh (MeshX2) and the butterfly (ButterflyX2) networks are shown in Figure 4. The CmeshX2 design uses two parallel Cmesh networks, similar to the design presented in Section 3, with the link and router radix reductions, as discussed above.

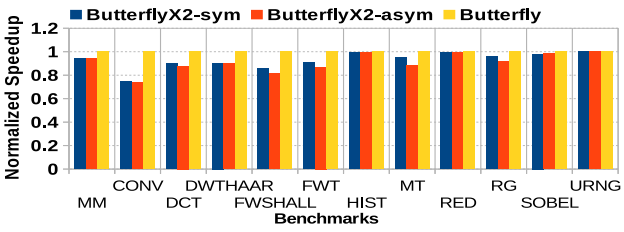
Table 4: Bisection bandwidth for symmetric and asymmetric designs of target topologies, i.e., butterfly, mesh, and Cmesh.

	L1-to-L2		L2-to-L1		Bisection Bandwidth
Topology	b_C	N_{BC}	b_C	N_{BC}	$N_{BC} \cdot b_C$
ButterflyX2-sym	22×8	6	22×8	6	2112
ButterflyX2-asym	16×8	6	22×8	6	1824
MeshX2-sym	22×8	10	22×8	6	2816
MeshX2-asym	16×8	10	22×8	6	2336
CmeshX2-sym	32×8	4	32×8	4	2048
CmeshX2-asym	22×8	4	32×8	4	1728

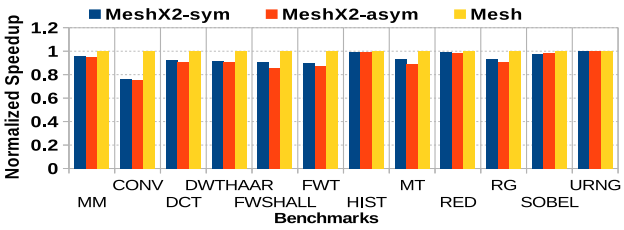
Based on these modifications, channel widths (see Table 4) are calculated using the bisection bandwidth criteria for the same network throughput as used in previous designs. For each baseline topology, two variations of the NoCs have been proposed; symmetric and asymmetric. The difference between these variations is in the channel width. In the symmetric designs, channel width of the L1-to-L2 and L2-to-L1 networks are the same. In the asymmetric designs, channels in the L1-to-L2 networks are chosen to be narrower since, as explained earlier, the L1-to-L2 communication latency has minimal effect on the overall system performance. The traffic in the L1-to-L2 direction is also lighter, as shown in Figure 3.

Overall we propose and analyze 6 different networks for the GPUs; MeshX2-sym, MeshX2-asym, ButterflyX2-sym, ButterflyX2-asym, CmeshX2-sym, and CmeshX2-asym. All of our proposed designs for mesh and Cmesh (MeshX2-sym, MeshX2-asym, CmeshX2-sym and CmeshX2-asym) use X-Y Routing. In the MeshX2-sym and MeshX2-asym designs, the links in the first and last row are unutilized because the L2 caches are placed diagonally in the center of the mesh. An example of the routing is shown in the Figure 4(b). As can be seen (and generalized), the messages from L2 never use the links in the last (and the first) row of these mesh-based designs. By removing these links, we reduce both static power consumption in the NoC and the number of bisectional links. The Cmesh-based designs (CmeshX2-sym and CmeshX2-asym) have the same logical organization as the baseline Cmesh (introduced in Section 3) for both directions, i.e. L1-to-L2 and L2-to-L1. The L1-to-L2 and L2-to-L1 networks in a butterfly-based design (ButterflyX2-sym and ButterflyX2-asym) are slightly different as compared to the baseline butterfly (introduced in Section 3). Since the number of L2s is 6, the last stage of the butterfly is modified (two routers are removed) to accommodate for this variation.

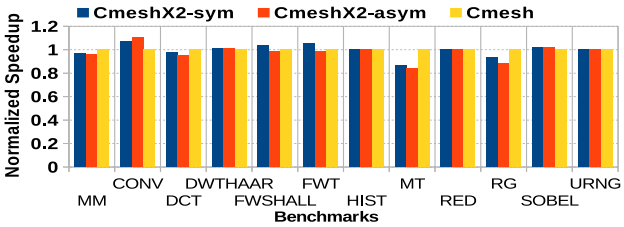
Figure 5 presents the performance of the proposed topologies, normalized to the respective baseline designs, i.e., butterfly, mesh and Cmesh. In Figure 5(a), the performance of the ButterflyX2-sym and ButterflyX2-asym is compared with the baseline butterfly NoC, as shown in Section 3. Both the ButterflyX2-sym and ButterflyX2-asym designs provide similar relative performance (92% and 91% as compared to the Butterfly network, respectively). This is while the asymmetric design (ButterflyX2-asym) has a smaller number of bisectional wires in the layout (as shown in Table 4). Figure 5(b) compares the performance of our proposed MeshX2-sym and MeshX2-asym designs with a conventional mesh design. The MeshX2-sym and MeshX2-asym designs provide similar performance (93% and 92% relative to the baseline mesh, respectively) while the L1-to-L2 links are narrower in the MeshX2-asym. The main reason for this slight performance loss in both ButterflyX2 and MeshX2 designs (symmetric



(a) **Speedup of asymmetric and symmetric ButterflyX2 against the baseline butterfly.**



(b) **Speedup of asymmetric and symetric MeshX2 against Mesh**



(c) **Speedup of asymmetric and symetric CmeshX2 against Cmesh**

Figure 5: Performance comparison of various parallel designs against baseline NoCs.

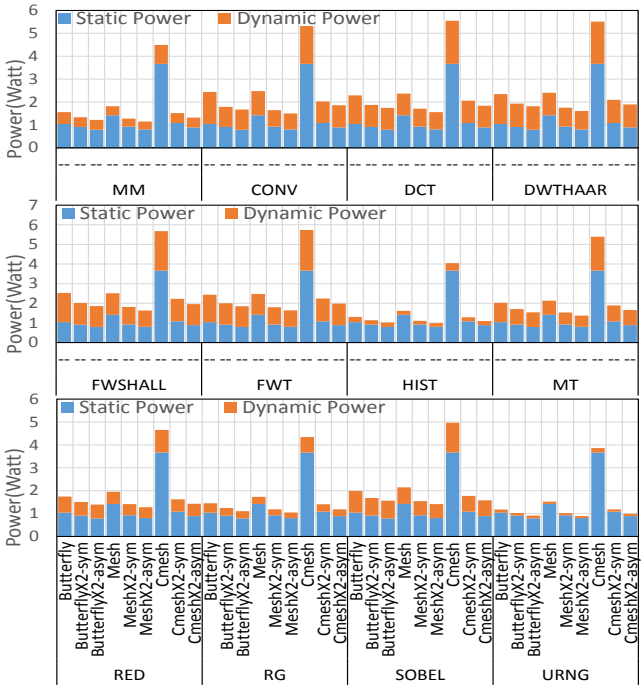


Figure 6: Dynamic and Static power breakdown for various parallel symmetric and asymmetric designs.

and asymmetric) is due to the lower channel width, which results in more serialization delay for access to the cache. However, this performance degradation is compensated by the reduction in congestion (resulting from using two separate networks for the L1-to-L2 and the L2-to-L1 communication).

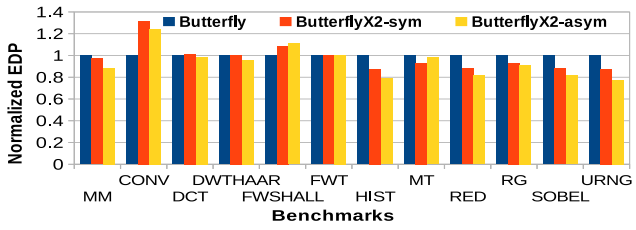
We present the speedup achieved by CmeshX2-sym and CmeshX2-asym as compared to the baseline Cmesh network in Figure 5(c). In both of our proposed designs the performance is almost equal to the baseline Cmesh network (99% and 97% of baseline performance), while the channel width is half of the channel width of the baseline Cmesh. The main reasons why we can obtain similar performance is due to a reduction in arbitration latency (CmeshX2 networks have switches with lower radices, since the connection between concentration switch and injecting nodes are unidirectional), and routing the L1-to-L2 and L2-to-L1 traffic on two different physical networks.

Figure 6 shows a comparison of conventional, symmetric and asymmetric NoCs, in terms of power consumption. The ButterflyX2-sym and ButterflyX2-asym designs do not exhibit significant power reduction in comparison to the baseline butterfly network (18% and 25% power reduction on average for symmetric and asymmetric design, respectively) because the reduction in power, due to reduction in channel width, has been mostly compensated by the increase in the number of low-radix routers.

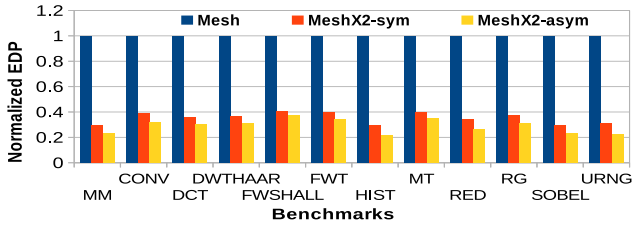
The MeshX2-sym and MeshX2-asym designs dissipate 35% and 44% less *static* power than the baseline mesh, respectively. This is due to the narrower bandwidth and lower channel counts of our MeshX2 designs, versus the higher component count of the mesh. While the router count is doubled in the design, most of these routers have low radices. The total power savings achieved by the MeshX2-sym and MeshX2-asym design are 30% and 37% on average in comparison to baseline mesh, respectively.

The largest power savings are observed when using parallel asymmetric CmeshX2 (CmeshX2-asym) and symmetric CmeshX2 (CmeshX2-sym). The CmeshX2-asym dissipates 69% less power, in comparison to the baseline concentrated mesh, while the CmeshX2-asym dissipates 65% lower power. This significant reduction in power is directly related to the narrowing of the channel width.

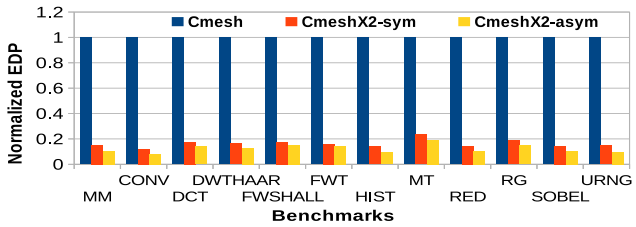
The EDP for each application is calculated based on the execution time of the application and energy consumed by this NoC during execution. Even though the proposed NoC designs have slightly lower performance on average in comparison to the baseline designs, they exhibit larger power savings that result in lower EDP than the baseline topologies. As shown in Figure 7(a), The ButterflyX2-asym has 7% lower EDP on average (lowest EDP among the butterfly designs) as compared to baseline butterfly network In Figure 7(b), the EDP of the proposed MeshX2 designs (MeshX2-sym and MeshX2-asym) is compared with the EDP for the baseline mesh network. We see significant power savings with the MeshX2-asym, while achieving comparable performance to the baseline mesh. This leads to a 72% reduction in EDP, as compared against the baseline. The significant power savings and comparable performance provided by the CmeshX2-asym



(a) **EDP comparison of proposed ButterflyX2-sym and ButterflyX2-asym against the baseline butterfly.**



(b) **EDP comparison of proposed MeshX2-sym and MeshX2-asym against baseline Mesh.**



(c) **EDP comparison of proposed CmeshX2-sym and CmeshX2-asym against baseline Cmesh.**

Figure 7: Design comparison based on the Energy-Delay Product metric.

versus the baseline Cmesh results in an 88% reduction in EDP (see Figure 7(c)).

Figure 8 compares the range of topologies proposed in this section in terms of EDP. The EDP values are normalized to the topology with the highest EDP, i.e., Cmesh. Cmesh also had the highest performance among all topologies that we evaluated in Section 3. As can be seen, our proposed CmeshX2-asym design has the lowest EDP among all the topologies (88% reduction against Cmesh), while it provides comparable performance as Cmesh (97%). MeshX2-asym designs are a close second, with a 86% reduction in EDP. High performance and low power consumption in CmeshX2-asym and MeshX2-asym designs make them suitable options for GPUs.

5. RELATED WORK

A large amount of work has been done in the area of network-on-chip designs for many-core architectures, targeting energy-efficient on-chip communication. A broad spectrum of network design topologies for on-chip communication in CMPs has been explored. The NoC designs for CMPs has matured to the extent that several commercial designs are now available [23, 25]).

The area of on-chip networks on a GPU has not been widely explored. In [5], the authors evaluate GPU performance

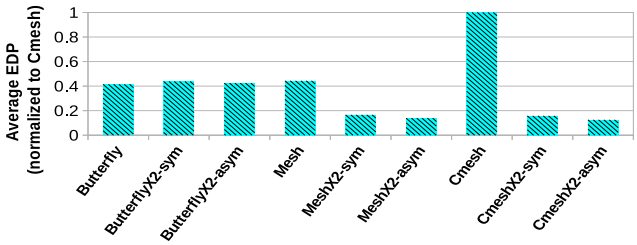


Figure 8: Overall comparison of baseline and proposed designs using the EDP metric.

across different micro-architecture and NoC design choices using different benchmarks. Our baseline evaluations differ from their work in the following aspects. In [5], the design layout is not considered, nor is power estimation. Also their hypothetical GPU assumed a smaller number of streaming multiprocessors and a higher number of memory controllers. This hides the effect of path diversity and congestion in the network.

The work in [4] exploits the many-to-few traffic patterns observable in manycore accelerators by alternating full routers in congested areas of a mesh with half routers that reduce cost. In [9] the authors propose a 3-D stacked GPU design using optical on-chip crossbar interconnects and explore the power consumption implications in throughput-oriented architectures. The work in [26] advocates using silicon-photonic link technology for on-chip communication in state-of-the-art commerical GPUs and presents an efficient GPU-specific photonic NoC design. It also examines the scalability of the NoC design for forward-looking GPUs with 128 compute units.

In the past, several research papers have suggested the use of parallel networks and channel slicing to improve the utilization of the NoC in CMPs [6, 14]. The main difference between these prior studies and our work is that our network is designed with two uni-directional asymmetric networks, tailored for the needs of a GPU.

6. CONCLUSION

In this work, we evaluated a number of network-on-chip designs, comparing power and performance metrics, targeting the memory subsystem of a contemporary state-of-the-art GPU architecture.

We first analyzed the memory access patterns of GPU applications, and used this to motivate the design of a range of asymmetric NoCs that are specifically tailored for GPUs. These asymmetric NoC designs use two different NoC architectures for L1-to-L2 and L2-to-L1 communications. This strategy reduces contention in the NoC, and in turn, improves application performance. We compared various asymmetric NoC designs based on performance, power and EDP metrics. Our analysis shows that CmeshX2-asym provides comparable performance to the best baseline design, Cmesh, but consumes 65% lower power. The MeshX2-asym topology also consumes 37% lower power than the baseline mesh while providing comparable performance to this topology. Based on these evaluations we conclude that the CmeshX2-asym and MeshX2-asym are the most suitable electrical NoC designs for GPU systems.

For future work, we plan to consider adding a dedicated NoC for smaller messages (i.e., cache control messages) and applying run-time power saving techniques on the sub-networks of CmeshX2-asym to further reduce power consumption and improve energy efficiency of the NoC, and the system as a whole.

7. REFERENCES

[1] AMD Accelerated Parallel Processing (APP) Software Development Kit (SDK). http://developer.amd.com/sdks/amdappsdk/.

[2] Predictive Technology Model. http://ptm.asu.edu/.

[3] AMD Graphics Cores Next (GCN) Architecture, June 2012. White paper.

[4] A. Bakhoda, J. Kim, and T. M. Aamodt. Throughput-effective on-chip networks for manycore accelerators. In *Proceedings of the 2010 43rd annual IEEE/ACM international symposium on microarchitecture*, pages 421–432. IEEE Computer Society, 2010.

[5] A. Bakhoda, G. L. Yuan, W. W. Fung, H. Wong, and T. M. Aamodt. Analyzing CUDA Workloads Using a Detailed GPU Simulator. In *Prof. of the Int'l Symposium on Performance Analysis of Systems and Software*, April 2009.

[6] J. Balfour and W. J. Dally. Design tradeoffs for tiled cmp on-chip networks. In *Proceedings of the 20th annual international conference on Supercomputing*, pages 187–198. ACM, 2006.

[7] J. Cole, S. Newman, F. Foertter, I. Aguilar, and M. Coffey. Breeding and genetics symposium: Really big data: Processing and analysis of very large data sets. *Journal of animal science*, 90(3):723–733, 2012.

[8] X. Cui, J. S. Charles, and T. Potok. Gpu enhanced parallel computing for large scale data clustering. *Future Generation Computer Systems*, 29(7):1736–1741, 2013.

[9] N. Goswami, Z. Li, R. Shankar, and T. Li. Exploring silicon nanophotonics in throughput architecture. *Design & Test, IEEE*, 31(5):18–27, 2014.

[10] A. Joshi, B. Kim, and V. Stojanovic. Designing energy-efficient low-diameter on-chip networks with equalized interconnects. In *High Performance Interconnects, 2009. HOTI 2009. 17th IEEE Symposium on*, pages 3–12. IEEE, 2009.

[11] D. R. Kaeli, P. Mistry, D. Schaa, and D. P. Zhang. *Heterogeneous Computing with OpenCL 2.0*. Morgan Kaufmann, 2015.

[12] J. Kim, W. J. Dally, and D. Abts. Flattened butterfly: a cost-efficient topology for high-radix networks. *ACM SIGARCH Computer Architecture News*, 35(2):126–137, 2007.

[13] M. Krone, J. E. Stone, T. Ertl, and K. Schulten. Fast visualization of gaussian density surfaces for molecular dynamics and particle system trajectories. *EuroVis-Short Papers*, 2012:67–71, 2012.

[14] P. Kumar, Y. Pan, J. Kim, G. Memik, and A. Choudhary. Exploring concentration and channel slicing in on-chip network router. In *Proceedings of the 2009 3rd ACM/IEEE International Symposium on Networks-on-Chip*, pages 276–285. IEEE Computer Society, 2009.

[15] X. Liang, K. Turgay, and D. Brooks. Architectural power models for sram and cam structures based on hybrid analytical/empirical techniques. In *Proc. of the Int'l Conference on Computer Aided Design*, 2007.

[16] M. Macedonia. The gpu enters computing's mainstream. *Computer*, 36(10):106–108, 2003.

[17] M. Mantor. Amd hd7970 graphics core next (gcn) architecture. In *HOT Chips, A Symposium on High Performance Chips*, 2012.

[18] J. Meng, C. Chen, A. K. Coskun, and A. Joshi. Run-time energy management of manycore systems through reconfigurable interconnects. In *Proceedings of the 21st Edition of the Great Lakes Symposium on Great Lakes Symposium on VLSI*, GLSVLSI '11, pages 43–48, New York, NY, USA, 2011. ACM.

[19] A. K. Mishra, N. Vijaykrishnan, and C. R. Das. A case for heterogeneous on-chip interconnects for cmps. In *Computer Architecture (ISCA), 2011 38th Annual International Symposium on*, pages 389–399. IEEE, 2011.

[20] S. Park, T. Krishna, C.-H. Chen, B. Daya, A. Chandrakasan, and L.-S. Peh. Approaching the Theoretical Limits of a Mesh NoC with a 16-Node Chip Prototype in 45nm SOI. In *Proc. of the 49th Design Automation Conference*, June 2012.

[21] J. E. Stone, D. Gohara, and G. Shi. OpenCL: A Parallel Programming Standard for Heterogeneous Computing Systems. *IEEE Design and Test*, 12(3), May 2010.

[22] R. Ubal, B. Jang, P. Mistry, D. Schaa, and D. Kaeli. Multi2Sim: A Simulation Framework for CPU-GPU Computing. In *Proc. of the 21st Int'l Conference on Parallel Architectures and Compilation Techniques*, Sept. 2012.

[23] S. R. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, A. Singh, T. Jacob, S. Jain, V. Erraguntla, C. Roberts, Y. Hoskote, N. Borkar, and S. Borkar. An 80-Tile Sub-100W TeraFLOPS Processor in 65nm CMOS. *IEEE Journal of Solid-State Circuits*, 43(1), Jan. 2008.

[24] H. Wang, L.-S. Peh, and S. Malik. Power-Driven Design of Router Microarchitectures in On-Chip Networks. In *Proc. of the 36th Int'l Symposium on Microarchitecture*, 2003.

[25] D. Wentzlaff, L. Bao, B. Edwards, P. Griffin, H. Hoffmann, A. Agarwal, J. F. Brown III, C. Ramey, C.-C. Miao, and M. Mattina. On-Chip Interconnection Architecture of the Tile Processor. *IEEE Micro*, 27(5), Sept. 2007.

[26] A. K. Ziabari, J. L. Abellan, R. Ubal, C. Chen, A. Joshi, and D. Kaeli. Leveraging silicon-photonic noc for designing scalable gpus. In *ACM International Conference on Supercomputing*. ACM, 2015.