NoC Symbiosis

(Special Session Paper)

Daniel Petrisko, Chun Zhao, Scott Davidson, Paul Gao,
Dustin Richmond and Michael Bedford Taylor
Bespoke Silicon Group
University of Washington

***Abstract*—Conventional wisdom states that Network-on-Chip router area grows quadratically with the channel width, and this perception has fundamentally shaped the assumptions of thousands of NoC papers that have been written to date, and many chip designs. However, this assumption is not entirely true. Simple analysis and empirical data from this paper shows that, in modern standard cell technology, a router's *standard cell logic area* actually grows only linearly; it is solely the *wire routing area* that grows quadratically.**

If we think of a NoC as a standalone block as is done in standard hierarchical VLSI design, then the overall area growth is indeed quadratic. But this approach either vastly under-utilizes logic area, or, in designs that match wire and logic area, leads to small network links. At the same time, many standard non-NoC logic blocks like processors or accelerator blocks typically use the standard cell logic area but need only a fraction of available wiring resources.

We propose an alternative approach, *NoC Symbiosis*, in which router logic and the node logic it services are jointly placed together. The router absorbs excess wiring resources from the node logic, and the node logic absorbs excess standard cell area from the router. Current-day automatic place and route (APR) tools already automatically distribute the router logic across the node logic, in order to provide enough space for the wiring resources. With this approach, future SoC's can leverage vastly larger amounts of wiring bandwidth than ever before, or alternatively, reduce the area overhead of existing routers.

We describe how we first encountered this phenomena, perform experiments to demonstrate its behavior, and provide design tips to help teams realize the potential of NoC Symbiosis.

I. Introduction

Network-on-Chip (NoC) is a common design pattern in modern SoCs, which scales to hundreds or thousands of nodes in a single chip while maintaining performance. They are a widely-used workhorse of multicore processor architectures [1], [2], [3], [4], [5], [6], [7], parallel architectures [8], and accelerators [9], [10], [11], [12], [13], [14].

Common practice envisions NoC routers as standalone blocks in SoC architectures. Network components, wiring, and other architectural logic can be designed independently of the architectural logic that they connect to and facilitate data transport to and from, then be implemented inside of independent bounding boxes, and replicated across a floorplan.

This separation of concerns in design practices is widely modeled in the many tools that estimate NoC area, power, and performance metrics [15], [16], [17], [18], [19], and many papers that analyze NoC area and power [20], [21], [22], [19].

978-1-4673-9030-9/20/$31.00 ©2020 IEEE

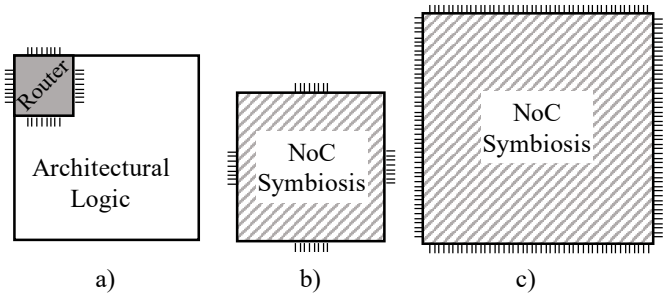

Fig. 1: (a) Conventional design practices suggests that NoC and architectural logic should be placed and routed in separate hierarchical boxes. We propose an alternate approach, *NoC Symbiosis*, where the components are placed and routed together (b) to save area for the same bandwidth, or (c) to allow greatly enhanced bandwidth and throughput with minimal area impact.

Because of this practice, conventional wisdom states that Network-on-Chip router area grows quadratically with the channel width, and this perception has fundamentally shaped the assumptions of thousands of NoC papers that have been written to date, and many chip designs. However, this assumption is not entirely true. Simple analysis and empirical data from this paper shows that, in modern standard cell technology, a router's *standard cell logic area* actually grows only linearly; it is solely the *wire routing area* that grows quadratically.

If we think of a NoC as a standalone block as is done in standard hierarchical VLSI design, then the overall area growth is indeed quadratic. But this approach either vastly under-utilizes logic area, or, in designs that match wire and logic area, leads to small network links. At the same time, many standard non-NoC logic blocks like processors or accelerator blocks typically use the standard cell logic area but need only a fraction of available wiring resources.

We propose an alternative approach, *NoC Symbiosis*, in which router logic and the node logic it services are jointly placed together. The router absorbs excess wiring resources from the node logic, and the node logic absorbs excess standard cell area from the router. Unlike in the past, current-day automatic place and route (APR) tools can now automatically distribute the router logic across the node logic, in order to provide enough space for the wiring resources. With this

approach, future SoC's can leverage vastly larger amounts of wiring bandwidth than ever before, or alternatively, reduce the area overhead of existing routers.

In this paper, we describe our experiences with NoC Symbiosis. We have seen NoC Symbiosis in our own work and have designed to exploit it. We also perform experiments in a 12 nm process, sweeping across a variety of network topologies, widths, and wiring constraints. These experiments explore the parameters that determine when NoC Symbiosis is possible, and expose the benefits of leveraging and the costs of ignoring symbiosis. Our experiments show that other architectures could benefit from a Symbiosis-aware design flow and we provide a methodology to recognize, leverage, and model the behavior of Symbiosis.

The remainder of this paper is organized as follows: In Section II we describe the concept of NoC Symbiosis. In Section III we relate where we discovered NoC Symbiosis. In Section IV we use our experience to develop experiments that demonstrate when NoC Symbiosis occurs. In Section V we distill our findings into a set of concrete design principles and we outline how NoC Symbiosis can be realized in future work. We conclude in Section VI.

II. Network-on-Chip Symbiosis

We introduce the concept of *Network-on-Chip Symbiosis*. Symbiosis is defined as a cooperative relationship between two dissimilar organisms. NoC Symbiosis occurs when the network components of a chip are placed and routed in conjunction with dissimilar logic, producing better Quality-of-Results (QoR) than independently placing and routing each component within its own bounding box.

Figure 1 demonstrates two applications of NoC Symbiosis. Instead of (a) instantiating a NoC router and non-network architectural logic separately, merging the logic can unlock two opportunities. First, as shown in (b), the total area can be potentially shrink. Second, as shown in (c), for surprisingly small incremental area, a designer can greatly increase bandwidth into and out of the tile, whether by widening existing channels or add more NoC channels. This bandwidth can be applied to decreasing congestion, reducing serialization latency, or adding new traffic classes or functionality (e.g. how about security or QoS features?) in a NoC.

Figure 2 illustrates why the bounding box for a 2D mesh router becomes under-utilized as the network link width increases. For a small network link width, the area of the bounding box (A_{BB}) is determined by the area of the router cells A_{SC}, which scale linearly with the network link width (N). As the network link width increases, the design reaches an inflection point where the bounding box is determined by the number of wires and the required wire pitch forces the perimeter to expand, quadratically growing the bounding box area (A_{BB}).

The relationship for router bounding box area in Figure 2 is summarized in Equation 1:

$$A_{BB} = max(\frac{A_{SC}}{U}, (D * N * P)^2) \quad (1)$$

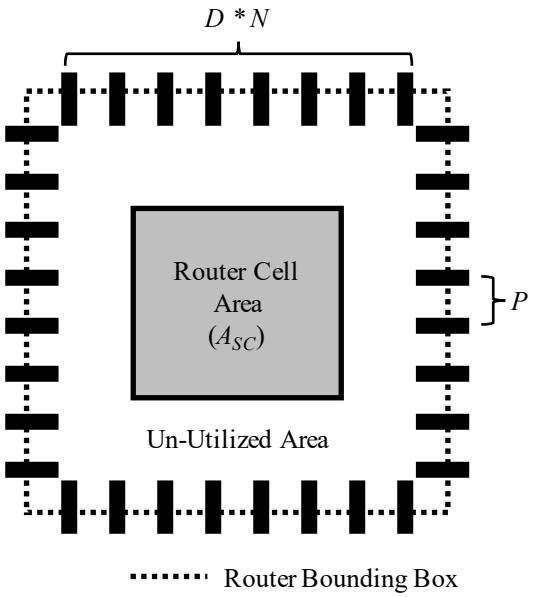

Fig. 2: In a NoC router, conventional wisdom holds that the area of the router bounding box (A_{BB}) scales quadratically with the number of wires on an edge ($N * D$). Instead, we demonstrate that the router cell area (A_{SC}) scales linearly with the network link width, while the bounding box perimeter scales linearly with the number of wires and the wire pitch (P). As the number of network link width increases, the area of the bounding box will first grow linearly with the router cell area, until perimeter requirements force the area to grow quadratically. Beyond this point, the bounding box will become underutilized by the router cells and provide an opportunity for NoC Symbiosis.

Where A_{BB} is the estimated area of the bounding box, A_{SC} is the sum of the area of the synthesized standard cells, U is the target area utilization of the bounding box, D is the duplex factor, N is the number of network link width and P is the effective wire pitch. In this paper we analyze full-duplex networks, so D is 2.

NoC Symbiosis is possible when the router bounding box is larger than the area required to fit its standard cells at a given utilization. This occurs when $(D * N * P)^2 > \frac{A_{SC}}{U}$. When this holds, the bounding box area scales quadratically with the number of wires, and the area becomes underutilized. In this Symbiotic Region, combined place and route of the network and non-network components will yield a better QoR.

III. NoC Symbiosis in the Wild

We first encountered NoC Symbiosis in BlackParrot (BP): a tiled, cache-coherent, Linux-capable RISC-V multicore introduced in [3]. BlackParrot implements the RISC-V 64-bit RV64G ISA, which includes the integer (I), multiplication and division (M), atomics (A), as well as single and double precision floating-point (F/D) instructions. It supports three privilege levels—machine, supervisor, and user-as well as SV39

virtual memory. These extensions are sufficient to run a full-featured operating system such as Linux. Development efforts have prioritized the use of intentional interfaces, modularity, silicon validation as first-order design metrics. Rather than employing a bus-based architecture, high-efficiency NoCs are used to provide scaling and flexibility so that BlackParrot can service a wide variety of design points. The three BlackParrot NoCs are the Coherence Network, I/O Network and Memory Network. For further implementation details, see [3].

We describe the evolution of BlackParrot across three versions, illustrating distinct points on the NoC Symbiosis spectrum. We then discuss implications of NoC Symbiosis on a variety of other tiled architectures which were not explicitly designed to be Symbiotic. These systems are summarized in Table I.

BlackParrot v0. BlackParrot v0 was designed with 5 wide network links that could shuttle entire 512-bit cache lines in a single cycle among cores and coherence directories, resulting in 4-10x greater bandwidth than contemporary NoCs in Table I, as well as 6-8x shorter serialization latency compared to BlackParrot v1 or v2. In addition to reducing memory access latency, this quick inter-tile communication provides order of magnitude speedup for coherence operations, which can cripple heavily cooperative multi-threaded programs.

We first attempted to tape in BlackParrot v0 as part of a VLSI course. Following conventional practice, we partitioned the physical design of the core into the Front End, the Back End, and the Memory End. Each portion was allocated to a different student. The Front and Back Ends quickly converged. The Memory End, with NoC routers, was unable to route without DRC errors and had incredibly low area utilization. Students were extremely unhappy with their assignment. At that point, we started to realize how insanely provisioned the NoC was!

We discovered NoC Symbiosis when a postdoc on our team mentioned that he had no problems placing and routing BP. It seemed impossible. But he, taking the easy route and wanting to avoid partitioning, had placed and routed the entire tile, instead of the individual components. The tools easily routed the entire tile, with a *smaller* bounding box than the three components implemented separately. Figure 1 depicts this phenomenon. This discovery impacted our internal tape-in builds, and led to the subsequent development of BlackParrot v1 and the study of NoC Symbiosis.

BlackParrot v1. While BlackParrot v0 reaped latency benefits from its wide NoCs, without a prefetcher or multithreaded memory system the core could not provide enough requests to use a reasonable portion of the average bandwidth available. As a relatively small core, it simply was not powerful enough to take advantage of the Symbiotic potential of 2500+ bits per cycle of bandwidth. Additionally, several other nodes on the NoC were incapable of sinking a full cache line per cycle, requiring excessive buffering storage and serialization penalties. To solve these issues, BlackParrot v1 consolidates the on-chip networks, reducing the network link width (and bandwidth!) using wormhole routers.

Wormhole routing decreased the link width significantly (at the cost of increased serialization latency and decreased bandwidth). The v0 I/O Network separated into two new networks: a 1D I/O network which passes through special purpose I/O tiles at the top of the chip, and a Memory Network, which is a half-duplex 1D network that flows from each core tile to the on-chip DRAM controller. Decomposing the 2D I/O network into 1D networks results in dramatic PPA benefits, as the router crossbars shrink from 5 ports to 3.

The changes in v1 moved the design to the edge of the Symbiotic Region described in Equation 1. While this resulted in PPA savings, BlackParrot v1 loses the advantages that v0 enjoyed from NoC Symbiosis. In BlackParrot v2, our current version, we leverage the implications of NoC symbiosis with a design that lies between v0 and v1.

BlackParrot v2. We developed BlackParrot v2 with additional parameterization to explore the NoC symbiosis design space. This version introduces additional RTL parameters to control the size of the core logic and router clocks, in addition to the wormhole routers introduced in v1. This allows the design to be tailored to particular application bandwidth or performance requirements. In particular, the parameterizations provided by v2 allow BlackParrot developers to compose its NoCs anywhere between area-optimized v1 and maximum-performance v0 configurations.

The Coherence Network widened so that an entire packet header fits in a single flit. This eliminates serialization latency from payload-free packets, such as loads and acknowledgements. The Memory networks expanded to a half-duplex 128b to reduce serialization latency while filling an L2 cache line.

Despite increasing the number of ports on each side of the tile by 70%, there was negligible impact on utilization. NoC Symbiosis allowed us to adjust system level bandwidth requirements with trivial backend adjustments.

Summary. Future iterations of BlackParrot will account for the extra flexibility that NoC Symbiosis grants, allowing for surprisingly wide topologies and router configurations. While BlackParrot has encountered NoC Symbiosis in a variety of designs, we will demonstrate in the next section that it is a general concept applicable to nearly all modern SoCs.

Symbiosis Gallery. Table I shows the Symbiotic capabilities of a variety of tiled architectures. Most designs do not take advantage of NoC Symbiosis, and are either firmly in the Standalone Router Region or near the inflection point (see Figure 3).

One reason designers may not have leveraged NoC Symbiosis in the past may is a lack of tooling support. For instance, Raw and Tilera were designed before timing-driven placement, so most blocks painstakingly positioned by hand. Such a flow does not allow for counter-intuitive experimentation like distributing router cells throughout the tile. This disconnect helps explain the relatively low pin utilization in these designs.

HammerBlade is a massively parallel manycore focusing on ML and graph applications, which will happily consume any available bandwidth. The Execution Migration Machine on the other hand aims to reduce on-chip bandwidth consump-

	Tilera [6]	**Raw** [23]	**OpenPiton** [4]	**HammerBlade** [24]	**EMM** [5]	**BP v0** [25]	**BP v1** [26]	**BP v2** [3] [27]
Process Node	90nm	180nm	32nm	12nm	45nm	40nm	12nm	12nm
Tile Area (mm^2)	9.6	16	1.175	0.025	0.784	0.832	0.360	0.360
Est. 2X Wire Pitch (nm)*	540	1080	192	128	270	240	128	128
Networks	5x34b	4x34b	3x66b	1x56b 1x97b	6x66b	1x130b 2x578b 2x642b	5x66b	3x98b 2x130b (Half-Duplex)
Max Wires Per Side	340	272	396	300	792	5140	648	848
Effective Link Width	170	136	198	150	396	2570	324	424
Pin Utilization+	5.9%	7.3%	7.0%	24.3%	24.2%	90.2%	13.8%	18.1%

TABLE I: Network specifications for various tiled architectures. Tilera and Raw are early examples of tiled manycores. OpenPiton and HammerBlade are more recent projects focusing on massive parallelism. The Execution Migration Machine is a tiled multicore without support for instruction-granularity thread migration. BlackParrot is a tiled, cache-coherent, application-class multicore. * All 2X wire pitches are estimated as 6x node size, except for 12nm, taken from [28]. + All pin utilizations are with DLDT pins except for BP v0, which used 3 pin layers to support its large number of pins.

tion of tiled manycores, instead migrating threads to cached data. However, its novel protocols employed many separate physical networks to ensure deadlock avoidance, resulting in a moderate pin utilization.

The remaining systems are all cache-coherent multicores: systems which tend to be latency-constrained rather than bandwidth-constrained. As opposed to EMM which requires complete transfers of state before resuming execution, techniques such as critical word first cache fills can alleviate serialization latency penalties incurred by narrower network links. As an aggressively scalable manycore, OpenPiton prefers narrower network links, allowing a designer to pack more small tiles into a chip. BlackParrot targets moderate scalability, between 1-16 cores, and so provides a wider range of pin utilization configurations, optimizing by specializing networks.

BlackParrot v0 is a notable outlier. It is a coherent multicore, yet it uses over 90% of its maximum pinout. As described above, cache line wide links allow for extremely low latency coherence operations. While conventional wisdom dictates that such large routers would be infeasible, we demonstrate that NoC Symbiosis enables this design space without any significant manual effort.

IV. NoC Symbiosis in Captivity

In this section we study NoC Symbiosis through a series of directed experiments. These experiments demonstrate the parameters where NoC Symbiosis exists and show how to recapture the un-utilized area that NoC Symbiosis provides.

We perform three experiments: First, we measure the relationship between network width, and bounding box utilization to demonstrate verify our hypothesis in Figure 2. Next, we determine how much of the un-utilized area in the bounding box can be reused by dissimilar logic to verify that the un-utilized area can be recycled. Finally, we examine the interaction between the number of networks, aggregate network width, and router area utilization to demonstrate that our experiments generalize to more than one network.

For our experiments we use silicon-validated wormhole routers from BaseJump STL[29]. The router is a 2D mesh router with 5 ports (North, East, South, West, and Local). Our network is input buffered, and stores two flits per router direction. The cardinal directions are routed to the edge of the bounding box.

For each experiment we place and route the wormhole router inside of a bounding box determined by Equation 1 and an aspect ratio of 1:1. We set the target utilization (U) to 80% and target a multi-corner 800 MHz operating frequency. To emulate the routing environment of a large SoC, we configure input and output delays for an 825 ps arrival time.

Our experiments are performed using the 12nm GlobalFoundries PDK. We use Design Compiler O-2018.06-SP4 to perform synthesis and IC Compiler II O-2018.06-SP5 to perform place and route.

Utilization vs. Network Link Width. To demonstrate why NoC Symbiosis arises, we first study how the utilization of the router bounding box scales as the network link width increases. This experiment aims to confirm our hypothesis in Figure 8. We place a router within a hierarchical bounding box and sweep the network link width (N) from 32 to 2048. We measure and plot the utilization of the bounding box ($\frac{A_{SC}}{A_{BB}}$), and the un-utilized area ($A_{BB} - A_{SC}$).

We examine two wire pitch configurations for network link wires to demonstrate how P from Equation 1 affects symbiosis. First, we space alternating metal layers two tracks apart (Double-Layer-Double-Track, or DLDT). This strategy is the default for newer process nodes where routing is plentiful, as it approximates a single track spacing while preserving signal integrity at increased density. Older nodes with fewer metal layers may have fewer options for accommodating NoC pins. For this case, we analyse layouts using a single pin layer with double track spacing (Single-Layer-Single-Track, or SLDT). Switching from DLDT to SLDT has the effect of doubling the value of P.

Utilization in DLDT. Figure 3 plots the utilization ($\frac{A_{SC}}{A_{BB}}$)

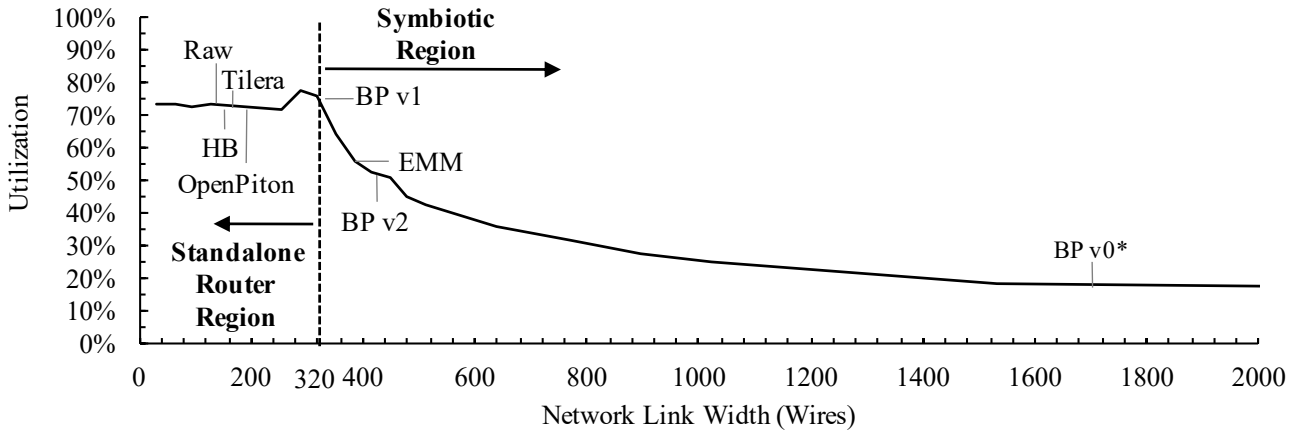

Fig. 3: Plot of router bounding box area utilization ($\frac{A_{SC}}{A_{BB}}$) as a function of DLDT network link width (N). Design points are annotated for related work in Table I. The utilization is initially constant, but when the bounding box becomes wire-limited the utilization decreases proportional to the inverse square, contrary to conventional wisdom. The annotations illustrate design points along the NoC symbiosis spectrum in Table I. * BlackParrot v0 actually used way more wires but on three layers; we normalize to DLDT.

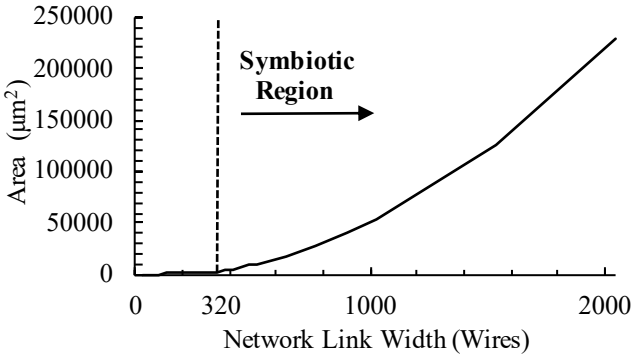

Fig. 4: Plot of un-utilized bounding box area ($A_{BB} - A_{SC}$) as a function of DLDT network link width (N). The available area is initially approximately 0, and remains constant as N increases, but as the width increases the un-utilized area begins to grow quadratically when it reaches the Symbiotic Region.

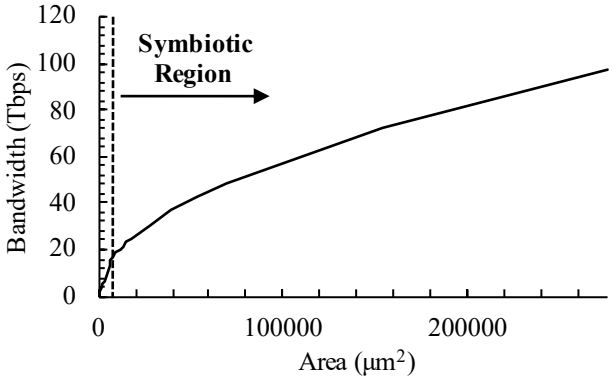

Fig. 5: Plot of router bandwidth, as a function of bounding box area (A_{BB}), for DLDT pins. Initially, the bandwidth grows linearly with the area, but when the symbiotic region is reached, the area grows quadratically and provides diminishing bandwidth returns.

as a function of the network link width (N) for DLDT. The data show that there is a range of network widths where the utilization decrease is linear, followed by a portion where the utilization decreases quadratically. This inflection point is highlighted in the plot at a network link width of 320. The region to the left is called the Standalone Router Region because this is where independent place and route of the router will not affect design QoR. The region to the right is called the Symbiotic Region because this is where symbiotic place and route of NoC router and logic components will yield better QoR.

Figure 3 is annotated with network link widths that correspond to the network configurations in related work.

Figure 4 plots the un-utilized area ($A_{BB} - A_{SC}$) as a function of the DLDT network link width (N). In the Symbiotic Region, the un-utilized area increases quadratically as hypothesized in Figure 2.

Figure 5 shows the relationship between bounding box area (A_{BB}) and router bandwidth. The bandwidth is derived from the network link width (N) and the target frequency of 800 MHz to compute bandwidth for each area result in our experiments.

Utilization with SLDT. As described in our experimental setup, in SLDT we have only a single layer of pins, resulting in an effective pitch P that is two times that of DLDT. Since the wire pitch is larger, we expect that the Symbiotic Region will begin at $\frac{320}{4}$ network link width, according to Equation 1.

Figure 6 plots the utilization ($\frac{A_{SC}}{A_{BB}}$) as a function of the

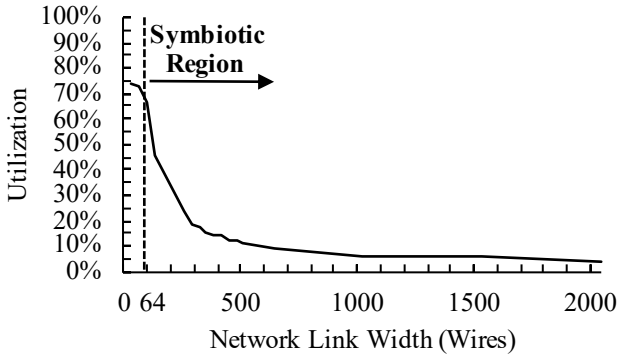

Fig. 6: Plot of router bounding box area utilization ($\frac{A_{SC}}{A_{BB}}$) as a function of SLDT network link width (N). The utilization is initially constant but when the bounding box becomes wire-limited, the utilization decreases proportional to the inverse square and provides an opportunity for NoC Symbiosis. This transition happens $4\times$ earlier than in Figure 3 (64 vs 320)

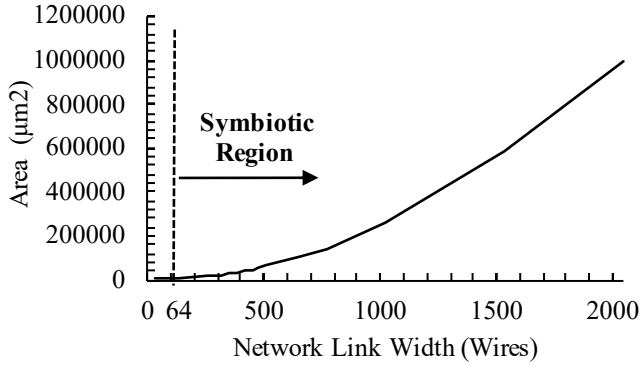

Fig. 7: Plot of un-utilized bounding box area ($A_{BB} - A_{SC}$) as a function of SLDT network link width (N). The available area is initially low, and remains constant as N increases, but as the network link width increases the un-utilized area begins to grow quadratically and can be filled with symbiotic logic.

SLDT network link width (N). As in Figure 3, the Symbiotic Region is annotated at the place where the bounding box switches from linear to quadratic growth.

As hypothesized, this occurs with approximately $\frac{1}{4}$ compared to DLDT designs. This is particularly relevant to older designs that were implemented in previous generation design nodes with fewer metal layers.

Figure 7 shows the un-utilized area ($A_{BB} - A_{SC}$) as a function of the network link width (N). It is annotated with the Symbiotic Region. As in Figure 6 the Symbiotic Region starts earlier. The unused area also increases more quickly, increasing to due to the increased value of P

Comparing the plots of the DLDT experiments and the SLDT experiments in the previous two sections, we can see that the wiring resources have a significant impact on NoC Symbiosis. This is important to consider when choosing a design node, metal stack, or track allocation strategy, as these

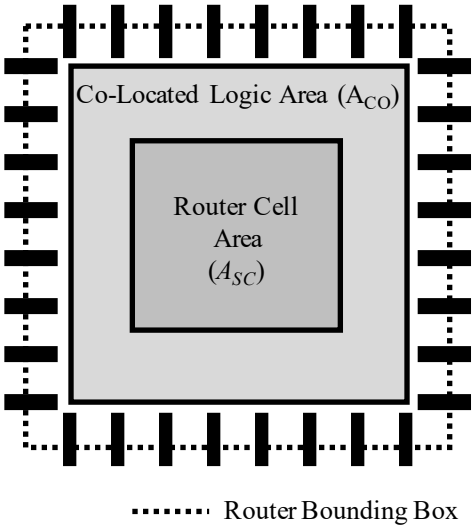

Fig. 8: The router bounding box from Figure 2 with co-located logic (A_{CO}) inserted to recover un-utilized area within the bounding box.

decisions affect Symbiotic behavior.

Attainable Symbiotic Area. In this experiment we will measure the resources made available through NoC Symbiosis by attempting to recover the un-utilized area. First, we determine un-utilized area for each network link width configuration using the data from DLDT experiments in Section IV. Then, we pack the remaining cell area with co-located logic by synthesizing shift registers between the inputs and outputs of the router. We repeat the process until until the estimated utilization reaches 80%, and no DRC errors occur. We then measure and report the area of the co-located logic, A_{CO} from Figure 8.

Figure 9 records the relationship between attainable co-located logic area (A_{CO}) and network link width (N). All points outside of the symbiotic region are ignored because the router bounding box is cell limited and introducing additional logic produced DRC errors. This figure demonstrates that the Attained co-located logic area increases along with the Theoretical un-utilized area. Thus, much of the un-utilized cell area can be recovered.

Networks vs Area. In this section we will measure how the number of networks affect the router standard cell area (A_{SC}) area as the aggregate network link width (N) increases. In the previous sections we demonstrated results for single 2D mesh router inside of a bounding box. In these experiments, we will repeat the data collection shown in the previous experiments, but we will also vary the number of routers in the bounding box.

Figure 10 plots the standard cell area of the routers in the bounding box (A_{SC}) as the the aggregate network link width varies from 128 to 512. The sweep is performed three times: with 1, 2, and 4 routers in the bounding box. As shown in the figure, the number of networks has no effect on A_{SC}. We

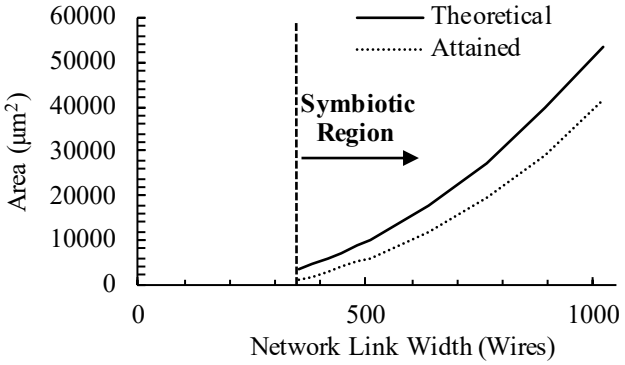

Fig. 9: Plot of co-located logic area vs. network link width (N). The Theoretical limit is the un-utilized area from Figure 4. Attained area shows the maximum co-located logic area that can be inserted into the un-utilized region.

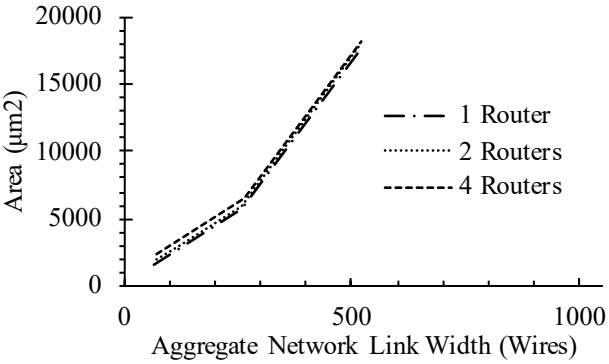

Fig. 10: Plot of router standard cell area (A_{SC}) vs. aggregate link width for 1, 2, and 4 routers within the same bounding box. The aggregate link width is the total width of links entering the side of a bounding box, with links distributed evenly between routers, with the links distributed equally between the routers. For a given point on the X-axis, the number of links entering the box remains constant.

conclude that *aggregate* network link width has the first order impact on physical design for wormhole routed networks.

V. Symbiotic Design Methodology

In this section we summarize our experiments and experiences into a set of helpful design tips. Our goal is to help teams *Recognize Opportunities for NoC Symbiosis*; *Design for NoC Symbiosis* when it is available; and build *Models for NoC Symbiosis* that can predict symbiotic impacts early in the design cycle.

Recognize NoC Symbiosis: NoC Symbiosis is possible when a network bounding block becomes underutilized. In Figure 3 and Figure 6 we demonstrated that there are many designs in, or near the Symbiotic Region. It is important to realize when a design reaches this region so that an optimal design is produced.

Equation 1, summarized below, can detect when a design enters the symbiotic region:

$$\frac{A_{SC}}{U} < (D * N * P)^2 \tag{2}$$

The Symbiotic Region of a specific design depends on system parameters like topology and link width as we demonstrated in our experiments. It also depends on process technology, metal stack and hierarchical strategy as we demonstrated in Section IV. Designers should be aware of these parameters and use them throughout the design process.

Design for NoC Symbiosis: NoC Symbiosis can be harnessed as a beneficial design parameter to improve QoR. Our experiences in Section III demonstrated that symbiosis can reduce design times as well as area. As we showed in Section IV, the unused area inside of a symbiotic router can be recaptured as useful design space.

With the right techniques, designers can move a design into the Symbiotic Region and tune their design appropriately. For example, designers can employ techniques such as wormhole routing, which allows link width to be varied without disrupting the architecture.

There are several ways to move a design into the Symbiotic Region:

- Increase the router link width.
- Increase the number of networks.
- Constrain the number of routing layers.

If a design is Symbiotic but wire limited, there are many ways to tune to take advantage of the extra area including:

- Increase sizes of local memory (cache, scratchpad, etc.).
- Combine multiple tiles and attach to a single router.
- Increase the datapath width of tile components.

Some of these can be tuned without major architectural implications, or architecture-level Quality-of-Result. Designers should consider high-level architectural goals and modeling when making these decisions.

Model NoC Symbiosis: Accurate estimation of chip metrics early in the design process is critical for reducing cost [30]. NoC estimation tools [15], [16], [17], [18], [19] are an essential tool for performing early estimation of chip metrics.

We believe that current generation non-parametric modeling tools [15] can be trained to estimate chip metrics in the presence of NoC Symbiosis. Previous work estimates NoC area in isolation and has not been tested with NoC Symbiosis. As we recounted in Section IV, an isolated router can produce underutilized area. In Section III we recounted how it increased design time. Using our technique described in Section IV, non-parametric tools should be able to model the transition into the Symbiotic Region, and the attainable area of co-located logic.

VI. Conclusion

In this paper, we introduced the concept of *Network-on-Chip Symbiosis*. NoC Symbiosis occurs when the network components of a chip are placed and routed in conjunction

with dissimilar logic components to produce better Quality-of-Result (QoR) than independent place and route of each component.

We recounted our experiences designing BlackParrot [3] and how our discovery of Symbiosis affected our design methodology. We used our knowledge to build experiments that demonstrated the symbiotic design space, and how to leverage NoC Symbiosis in a design. Finally, we distilled our experiences into a set of best practices for NoC Symbiosis.

ACKNOWLEDGMENTS

This material is based on research sponsored by Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) under agreement numbers FA8650-18-2-7856 and FA8650-18-2-7852. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) or the U.S. Government.

This work was partially supported by NSF SaTC Award 1563767, NSF SaTC Award 1565446, and by the DARPA/SRC JUMP ADA Center.

REFERENCES

[1] E. Waingold, M. Taylor, D. Srikrishna, V. Sarkar, W. Lee, V. Lee, J. Kim, M. Frank, P. Finch, R. Barua, J. Babb, S. Amarasinghe, and A. Agarwal, "Baring it all to Software: Raw Machines," in IEEE Computer, September 1997.

[2] M. B. Taylor, W. Lee, S. Amarasinghe, and A. Agarwal, "Scalar Operand Networks: On-Chip Interconnect for ILP in Partitioned Architectures," in High Performance Computer Architecture (HPCA), February 2003.

[3] D. Petrisko, F. Gilani, M. Wyse, D. C. Jung, S. Davidson, P. Gao, C. Zhao, Z. Azad, S. Canakci, B. Veluri, T. Guarino, A. Joshi, M. Oskin, and M. B. Taylor, "Blackparrot: An agile open-source risc-v multicore for accelerator socs," IEEE Micro, vol. 40, no. 4, pp. 93–102, July 2020.

[4] J. Balkind, M. McKeown, Y. Fu, T. Nguyen, Y. Zhou, A. Lavrov, M. Shahrad, A. Fuchs, S. Payne, X. Liang, M. Matl, and D. Wentzlaff, "Openpiton: An open source manycore research framework," in Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems, ser. ASPLOS '16. New York, NY, USA: ACM, 2016, pp. 217–232. [Online]. Available: http://doi.acm.org/10.1145/2872362.2872414

[5] K. S. Shim, M. Lis, M. H. Cho, I. Lebedev, and S. Devadas, "Design tradeoffs for simplicity and efficient verification in the execution migration machine," in 2013 IEEE 31st International Conference on Computer Design (ICCD), 2013, pp. 145–153.

[6] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C. Miao, J. F. Brown III, and A. Agarwal, "On-chip interconnection architecture of the tile processor," IEEE Micro, vol. 27, no. 5, pp. 15–31, Sep. 2007.

[7] S. Bell, B. Edwards, J. Amann, R. Conlin, K. Joyce, V. Leung, J. MacKay, M. Reif, L. Bao, J. Brown, M. Mattina, C. Miao, C. Ramey, D. Wentzlaff, W. Anderson, E. Berger, N. Fairbanks, D. Khan, F. Montenegro, J. Stickney, and J. Zook, "Tile64 - processor: A 64-core soc with mesh interconnect," in IEEE International Solid-State Circuits Conference, Feb 2008, pp. 88–598.

[8] A. Rovinski, C. Zhao, K. Al-Hawaj, P. Gao, S. Xie, C. Torng, S. Davidson, A. Amarnath, L. Vega, B. Veluri, A. Rao, T. Ajayi, J. Puscar, S. Dai, R. Zhao, D. Richmond, Z. Zhang, I. Galton, C. Batten, M. B. Taylor, and R. G. Dreslinski, "Evaluating celerity: A 16-nm 695 giga-risc-v instructions/s manycore processor with synthesizable pll," IEEE Solid-State Circuits Letters, vol. 2, no. 12, pp. 289–292, 2019.

[9] M. Khazraee, L. Vega, I. Magaki, and M. Taylor, "Specializing a Planet's Computation: ASIC Clouds," IEEE Micro, May 2017.

[10] N. Goulding, J. Sampson, G. Venkatesh, S. Garcia, J. Auricchio, J. Babb, M. Taylor, and S. Swanson, "GreenDroid: A Mobile Application Processor for a Future of Dark Silicon," in HOTCHIPS, 2010.

[11] M. B. Taylor, "Is Dark Silicon Useful? Harnessing the Four Horsemen of the Coming Dark Silicon Apocalypse," in Design Automation Conference (DAC), 2012.

[12] S. Davidson, S. Xie, C. Torng, K. Al-Hawai, A. Rovinski, T. Ajayi, L. Vega, C. Zhao, R. Zhao, S. Dai, A. Amarnath, B. Veluri, P. Gao, A. Rao, G. Liu, R. K. Gupta, Z. Zhang, R. Dreslinski, C. Batten, and M. B. Taylor, "The celerity open-source 511-core risc-v tiered accelerator fabric: Fast architectures and design methodologies for fast chips," IEEE Micro, vol. 38, no. 2, pp. 30–41, Mar 2018.

[13] Y. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," in International Symposium on Computer Architecture (ISCA), 2016, pp. 367–379.

[14] Z. Du, R. Fasthuber, T. Chen, P. Ienne, L. Li, T. Luo, X. Feng, Y. Chen, and O. Temam, "Shidiannao: Shifting vision processing closer to the sensor," in International Symposium on Computer Architecture (ISCA), June 2015, pp. 92–104.

[15] A. B. Kahng, B. Lin, and S. Nath, "Orion3.0: A comprehensive noc router estimation tool," IEEE Embedded Systems Letters, vol. 7, no. 2, pp. 41–45, June 2015.

[16] ——, "Explicit modeling of control and data for improved noc router estimation," in DAC Design Automation Conference 2012, June 2012, pp. 392–397.

[17] A. B. Kahng, Bin Li, L. Peh, and K. Samadi, "Orion 2.0: A fast and accurate noc power and area model for early-stage design space exploration," in 2009 Design, Automation Test in Europe Conference Exhibition, April 2009, pp. 423–428.

[18] C. Tan, Y. Ou, S. Jiang, P. Pan, C. Torng, S. Agwa, and C. Batten, "Pyocn: A unified framework for modeling, testing, and evaluating on-chip networks," in International Conference on Computer Design (ICCD), 2019, pp. 437–445.

[19] J. Balfour and W. J. Dally, "Design tradeoffs for tiled cmp on-chip networks," in ACM International Conference on Supercomputing 25th Anniversary Volume. New York, NY, USA: Association for Computing Machinery, 2006, p. 390–401. [Online]. Available: https://doi.org/10.1145/2591635.2667187

[20] K. Sewell, R. G. Dreslinski, T. Manville, S. Satpathy, N. Pinckney, G. Blake, M. Cieslak, R. Das, T. F. Wenisch, D. Sylvester, D. Blaauw, and T. Mudge, "Swizzle-switch networks for many-core systems," IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 2, no. 2, pp. 278–294, June 2012.

[21] R. Dreslinski, K. Sewell, T. Manville, S. Satpathy, N. Pinckney, G. Blake, M. Cieslak, R. Das, T. Wenisch, D. Sylvester, D. Blaauw, and T. Mudge, "Swizzle switch: A self-arbitrating high-radix crossbar for noc systems," in 2012 IEEE Hot Chips 24 Symposium (HCS), Aug 2012, pp. 1–44.

[22] S. Satpathy, Z. Foo, B. Giridhar, R. Dreslinski, D. Sylvester, T. Mudge, and D. Blaauw, "A 1.07 tbit/s 128 × 128 swizzle network for simd processors," in Symposium on VLSI Circuits, June 2010, pp. 81–82.

[23] M. B. Taylor, J. Psota, A. Saraf, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, A. Agarwal, W. Lee, J. Miller, D. Wentzlaff, I. Bratt, B. Greenwald, H. Hoffmann, P. Johnson, and J. Kim, "Evaluation of the raw microprocessor: an exposed-wire-delay architecture for ilp and streams," in International Symposium on Computer Architecture, 2004., 2004, pp. 2–13.

[24] M. Ruttenberg and M. Taylor, "The hammerblade risc-v manycore," FOSDEM 2020, 2020.

[25] Bespoke Silicon Group, "Blackparrot: A linux-capable accelerator host multicore," https://github.com/black-parrot/black-parrot/tree/archive/pparrot_ee477_wi19, 2020.

[26] ——, "BlackParrot: A Linux-Capable Accelerator Host Multicore," https://github.com/black-parrot/black-parrot/tree/tapeout_bp_rc2, 2020.

[27] ——, "BlackParrot: A Linux-Capable Accelerator Host Multicore," https://github.com/black-parrot/black-parrot/tree/tapeout_bp_rc3, 2020.

[28] "14 nm lithography process," https://en.wikichip.org/wiki/14_nm_lithography_process.

[29] M. B. Taylor, "Basejump STL: Systemverilog Needs a Standard Template Library for Hardware Design," in Design Automation Conference, 2018.

[30] A. B. Kahng, "New directions for learning-based ic design tools and methodologies," in 2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC), 2018, pp. 405–410.