Optimization of Wire Pipelining and Channel Parallelism for 2D-Mesh NoC Physical Design

Pei-Huan Tsai, Maico Cassel dos Santos, Joseph Zuckerman, Kuan-Lin Chiu, Luca P. Carloni
Department of Computer Science, Columbia University
New York, NY
{tph, mcassel, jzuck, chiu, luca}@cs.columbia.edu

***Abstract*—Modern systems-on-chip (SoCs) increasingly rely on high-bandwidth networks-on-chip (NoCs) to support communication among their many heterogeneous components. Meanwhile, as technology nodes advance, physical design (PD) has a growing impact on NoC performance, particularly for high-bandwidth NoCs. However, few published works focus on NoC optimization from a PD perspective. In this work, we study the problem of optimizing the PD of 2D-mesh NoCs by focusing on two prominent techniques: wire pipelining and channel parallelism. Our study is based on experimental results obtained from multiple tape-in NoC designs in a 12nm technology process. We develop models to approximate the power and area effects of different NoC design approaches and analyze the underlying trends. Our findings show that pipelining does not affect die area but increases NoC power consumption by 1.6× compared to increasing parallelism. In contrast, increasing parallelism can result in a NoC area up to 2× larger than one achieving the same bandwidth through pipelining. Building on these insights, we formulate a mathematical optimization problem, which could be solved by optimization solvers to balance the trade-offs between these two techniques. Our study provides a general framework for analyzing NoC physical design trade-offs and optimizing NoC configurations.**

***Index Terms*—system-on-chip, network-on-chip, physical design, case study, analysis and optimization.**

I. Introduction

With the ever-increasing complexity of system-on-chip (SoC) designs, the network-on-chip (NoC) has become the primary architecture employed for on-chip communication. The inherent scalability of NoCs, combined with the ability to decouple computation and communication, maps well to SoC designs with tens to hundreds of components [1]. Meanwhile, the rise of compute-intensive applications, such as deep learning, and increasingly performant I/O interfaces, such as high-bandwidth memory, are driving NoC architectures to deliver maximum bandwidth under stringent SoC resource constraints.

Prior work on increasing NoC bandwidth mostly focuses on optimizing the NoC architecture itself (topology and protocols for routing and flow-control) and its key elements (primarily the router micro-architecture). Few research papers, however, have presented the *physical design (PD)* implications of NoC design choices, despite the recent rise in academic SoC tape-out efforts targeting machine learning applications [2]–[5]. Among various SoC designs, whether at the tape-in or tape-out stage, NoC architectures with 2D-mesh are widely adopted by SoC designers in both industry [6] and academia [2]–[4], [7], [8]. Considering a 2D-mesh NoC for a *tile-based SoC architecture*, as shown in Fig. 1, *channels* are created between tile implementations, wherein the NoC (i.e., routers and wire connections) can be implemented. This approach decouples the implementation of the tiles from the NoC, which is advantageous for several reasons: 1) bugs in tile implementations are less likely to affect the entire chip; 2) the implementation and optimization of the NoC and tiles can proceed in parallel; and 3) PD tasks and strategies, such as clocking and power delivery, are easier to implement.

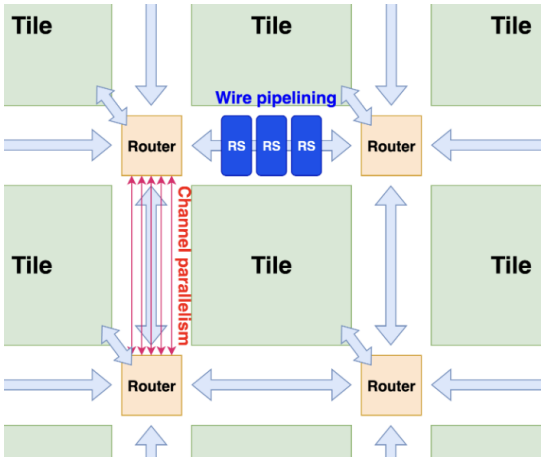


Fig. 1. 2D-mesh NoC: channel parallelism vs. wire pipelining.

In a 2D-mesh NoC architecture, we identify two optimization techniques to increase bandwidth. First, the number of wires between routers, which we denote as *channel parallelism*, can be increased. Although this is a relatively simple architectural change, the communication interfaces of tiles and routers need to be modified accordingly. Besides, increasing the number of wires can reshape the entire physical resource allocation strategy on an SoC. Moreover, this method has limitations in advanced technology nodes. While transistor performance continues to improve, the impact of interconnect RC delays grows with technology scaling [9]. As a result, the wire delay of long router-to-router links in NoC PD implementations has increasingly become the determining factor of a NoC's maximum frequency, necessitating link buffering techniques to compensate. *Wire pipelining* can be used to increase the operating clock frequency of the NoC. Pipeline stages, however, must comply with the NoC's *flow control*

This work is supported in part by an award from IBM corporation.

protocol, adding additional complexity to the pipelining flip-flop designs. Moreover, the introduction of extra standard cells can increase the overall cost of SoC physical implementation, further restricting the applicability of this approach.

Both channel parallelism and wire pipelining come with associated power, performance, and area (PPA) trade-offs. The two techniques can also be combined to take advantage of their benefits while compensating for their drawbacks. In this work, we perform an extensive analysis of the trade-offs between these two techniques and, in particular, their implications on PD. Adopting a *latency-insensitive (LI) design* approach to NoC optimization [10], [11], we insert *relay stations (RSs)* between NoC routers to pipeline wires, while complying with the principles of LI design. We carry out multiple NoC tape-ins in a $12nm$ technology, ensuring they are placed, routed, and design-rule-check (DRC) clean, while varying both channel parallelism and the number of *RSs* (i.e., pipeline stages) inserted. We analyze key parameters including power, timing closure, and channel size, and develop estimation functions accordingly. Based on these observations, we propose an optimization formulation and present a case study demonstrating how balancing these factors can improve NoC performance. Our main contributions are as follows:

1) We conducted a comprehensive PPA analysis on many 2D-mesh NoC designs with *RSs* through tape-in results in a $12nm$ technology node.
2) We modeled the trends of area and power overhead associated with both techniques based on the collected data, demonstrating the effectiveness of our models.
3) Based on the insights and models we derived, we formulate and solve an optimization problem to determine the optimal NoC configuration, considering both power and area constraints for chip designers.

To the best of our knowledge, this is the first analysis of the various trade-offs involved in optimizing the performance of 2D-mesh NoC from a PD perspective that is supported by real tape-in results with a commercial electronc design automation (EDA) flow.

II. Background

2D-Mesh NoC. For several reasons, the 2D-mesh NoC is a highly-appealing topology. Its regularity and simple geometry make it well-suited for SoC design and implementation, as discussed in Section I. When combined with turn restriction routing [12], the 2D-mesh architecture is guaranteed to be free from *routing deadlock*. This means that NoC routers do not need to implement any kind of deadlock avoidance, detection, or recovery mechanisms. Because of these nice properties, the 2D-mesh topology is widely adopted, including in SoC design platforms [7], [8], [13] and silicon prototypes [2]–[4].

Multiple Channels. For many applications, NoCs must employ *multiple channels* to avoid *protocol deadlock*. These channels can be either virtual or physical (implemented as separate planes). For example, in a classic MESI coherence protocol, requests, responses, and forwards must use separate channels [14]. *Virtual channels* enable multiple virtual networks within a single physical NoC by using distinct input queues and managing them based on message types. Alternatively, *multiple physical NoC planes* form independent networks with simpler routers [11]. The number of NoC planes required depends on the SoC architecture. For example, Celerity [2] employs a single plane to efficiently support its large mesh of simple cores for parallel programming. FlooNoC [8] utilizes a three-plane NoC to handle AXI traffic. Both OpenPiton [7] and ESP [3] adopt three planes for their cache-coherence protocols; additionally, ESP allocates two more planes for accelerator DMA and one for miscellaneous functions, totaling six planes. In our experiments, we consider NoC designs with 1, 2, 3, and 6 planes to cover this range.

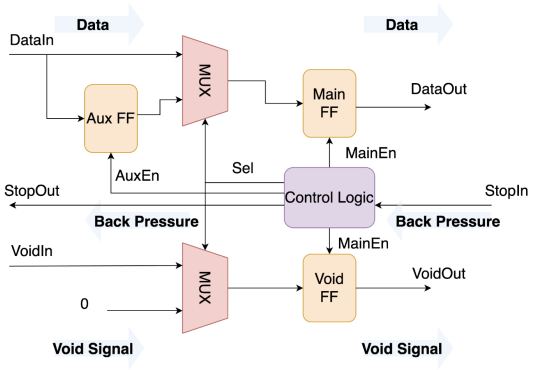
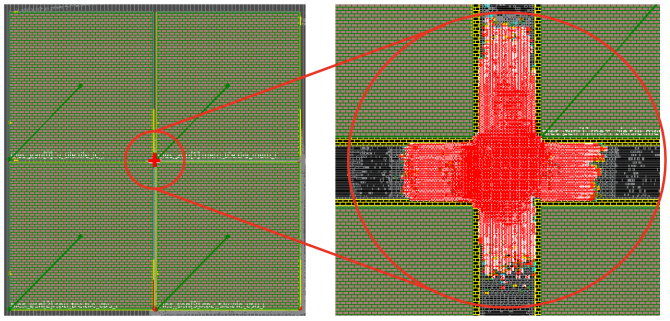


Fig. 2. Relay station for LID: block diagram [15].

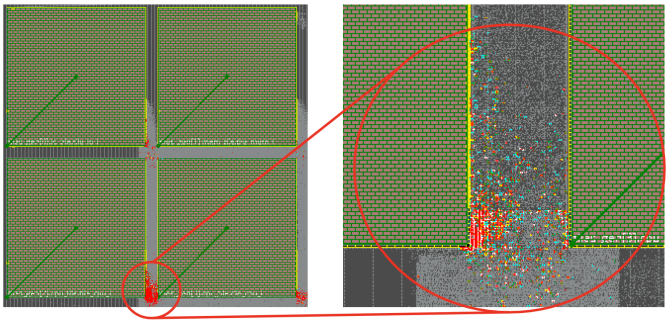
Latency-Insensitive Design. As predicted by Matzke [16], technology scaling has resulted in a smaller portion of chips being reachable within a single clock cycle. Latency-insensitive design (LID) was introduced to cope with increasing communication delays in the synchronous design of integrated circuits [10]. To handle arbitrary delays between components, *stallable modules* can be wrapped with a LI *shell* that handles inter-module communication and stalls the execution of a module at each clock cycle when a complete set of valid data is not available at its input ports. Shells are connected by *LI channels* that carry, along with the data, a *void* and a *stop* signal to indicate when an upstream module has not produced valid data and when a downstream module is not ready to receive data, respectively. *RSs* can be used to pipeline LI channels, while maintaining the correctness of the control protocol. Fig. 2 shows the design of the *RS*, which consists of a finite state machine that controls whether data should be latched in *main* or *auxiliary* flip-flops and correctly propagates the void and stop signals [15]. While the number of flip-flops of a *RS* must be equal to two times the data width, the control logic and void-signal flip-flop remain constant.

III. Physical Design Analysis

Experimental Setup. The experimental results presented in this paper are based on multiple tape-in designs that have successfully undergone place-and-route (P&R) and have passed DRC verification. To carry out these experiments, we used state-of-the-art commercial EDA tools with a 12nm FinFET technology process. The experiments were based on a tile-



(a) Routing congestion in the router section.



(b) Routing congestion due to channel parallelism.

Fig. 3. Examples of wiring congestion in different regions.

based SoC design with a tile size of $2mm \times 2mm$. Liberty views were extracted during the tile-level physical design and used for the top-level SoC physical design.

Methodology. To evaluate the PPA trade-offs, we focus on two key parameters: the *minimum channel size* of the NoC and the *maximum operating frequency* achievable through wire pipelining. To determine the minimum channel size, we reduce its size during the floorplanning phase until the routing EDA tool cannot handle congestion, thus causing unacceptably long runtimes and poor quality of results.[1] In our evaluation, we deem the design failed during P&R if the tool's runtime exceeds twice its typical duration. To identify the maximum frequency, we adjust the clock frequency in the constraint files and run the complete PD flow. If timing violations are limited to a few picoseconds of setup time, we consider the implementation still valid since these violations can be resolved by reducing the clock frequency. As frequency increases, the implementation can suffer from an excessively long tool runtime or a significant amount of hold violations. In this scenario, we define that P&R fails when the tool runtime doubles or when the design presents hold timing violations.

A. The Impact of Channel Parallelism

An increase in channel parallelism expands the channel size of an SoC. In our experiments, we discovered and defined two types of channel size constraints: *router constraint* and *channel-parallelism constraint*. The router constraint ensures

[1]In this paper we use the term "router" for the hardware component of the NoC, while we use the term "routing" to refer to the action of the router EDA tool. Also, we use the term "congestion" to denote limited available resources to the router EDA tool, not for traffic in the NoC's runtime operation.

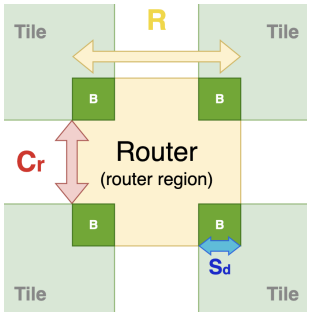


Fig. 4. Symbols and equations for the router constraint.

$$S = \frac{S_d}{C_r} = \frac{R - C_r}{2C_r} \quad (1)$$

$$B = (\frac{R - C_r}{2})^2 \quad (2)$$

$$\frac{A}{D} = R^2 - 4B \quad (3)$$

$$\frac{A}{D} = 2RC_r - C_r^2 \quad (4)$$

that the router region has sufficient area to accommodate the router's standard cells and wires. Fig. 3a shows an example of a violation of this constraint, where the router area is congested, as shown by the red marks in the channel region. The channel-parallelism constraint, on the other hand, ensures there are enough routing resources inside channels for the wire connections between routers. Fig. 3b illustrates the violation of this constraint where the wires from the north and local ports are squeezed together in the channel and cause congestion.

Fig. 4 shows the router region in yellow and tiles in green. The router topology extends into the channels, forming a cross-shaped configuration. The main advantage of this topology is to allow narrower channels compared to a square-shape router. Because of the inter-dependency of the channel size and the router region, it is critical to estimate the minimum size of a channel size that respects router constraints. Equation (1) calculates the *stretching factor* S of the router by taking the ratio of the *router extension* S_d and the *channel size* C_r. Equation (2) calculates the area occupied by each tiles' corner, where R is the *total length of the router*. Equation (3) calculates the router region area which is equal to dividing the router's *standard cell area* A by a *density factor* D. Finally, Equation (4) expresses the C_r as a function of A, D, and R.

A large stretching factor combined with high cell density can lead to long wiring detours, resulting in P&R failures. In our experiments, a stretching factor larger than 1.2 and a pre-placement estimation of the standard cell density larger than 0.7 cause routing detours leading to congestion and time degradation. Hence, we used 1.2 and 0.7 as empirical values for S and D, respectively. By replacing these two values in (1) and (4) and isolating R and C, respectively, we reach the following lower bound for the channel size:

$$C_r > \sqrt{\frac{A}{0.7 \cdot 5.8}} = \sqrt{k_1 \cdot data\_width}, \quad (5)$$

where k_1 is a constant, as the router's standard cell area is positively correlated with the data width.

The channel-parallelism constraint can be modeled as a linear function of the data width due to the linear relationship of the available routing tracks with the channel size:

$$C_c = k_2 \cdot data\_width. \quad (6)$$

Fig. 5 illustrates how (5) and (6), represented by the blue and green lines respectively, follow the actual lower

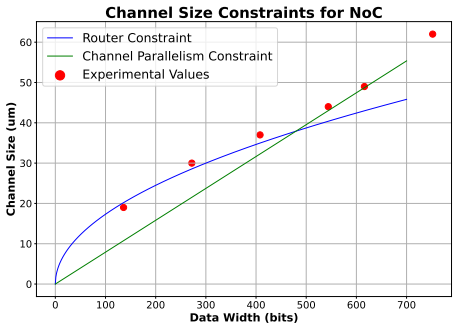


Fig. 5. Channel size model and validated results.

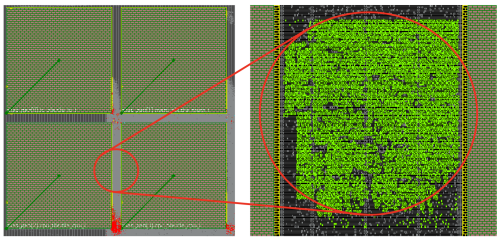


Fig. 6. *RSs* do not contribute to congestion.

bound obtained from our experiments (red dots) on NoCs with various data widths. The router constraint grows sub-linearly, as increasing channel size makes the router region grow quadratically. On the other hand, the channel-parallelism constraint grows linearly with respect to the number of wires inside the channel. Values obtained from our experimental analysis closely follow the greater of these two constraints, showing the accuracy of the estimation function.

By combining (5) and (6), we obtain the following equation to determine a channel size that satisfies both router and channel constraints:

$$C = \max(C_r, C_c). \tag{7}$$

The router constraint is primarily determined by the area of the standard cells. Changes to the router type or micro-architecture can result in scaling variations. The channel-parallelism constraint, on the other hand, can be affected by the reduction of routing-track resources due to area-reuse techniques. For example, if an over-the-cell routing technique is applied, the channel-parallelism constraint curve will exhibit a steeper slope compared to the original curve.

B. The Impact of Wire Pipelining

In our experiments, we adopted a channel-dedicated NoC architecture that reserves channel resources specifically for the NoC, as this approach remains mainstream in SoC NoC design [2], [3], [7], [17]. With this architecture, *RSs* do not contribute to routing congestion during PD implementation, as they effectively reutilize the space within the channel that is already allocated for routers and inter-router wiring. Throughout our experiments, wire pipelining did not result in any increase in channel size, as illustrated in Fig. 6.

Another architecture, over-the-cell NoCs [8], repurposes the channel area in channel-dedicated NoC designs to expand the

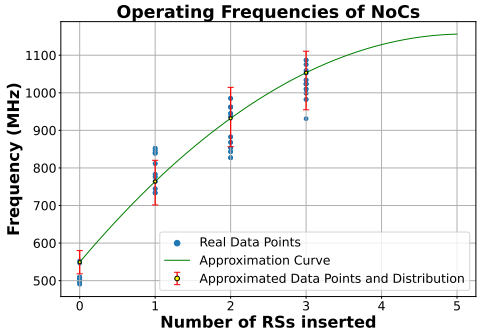


Fig. 7. NoC operating frequencies after *RS* insertion

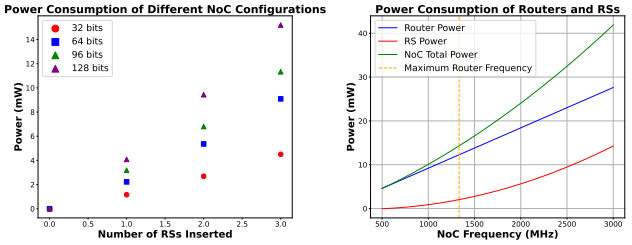
TABLE I
POWER SPLIT (MW) OF ROUTERS AND RELAY STATIONS UNDER DIFFERENT NOC CONFIGURATIONS.

	0 RS		1 RS		2 RS		3 RS	
	Router	RS	Router	RS	Router	RS	Router	RS
32 Bits	4.87	0	8.02/87.4%	1.16/12.6%	9.59/78.1%	2.69/21.9%	10.76/70.5%	4.50/29.5%
64 Bits	10.19	0	14.30/86.6%	2.22/13.4%	17.92/77.0%	5.36/23.0%	21.00/70.0%	9.09/30.0%
96 Bits	11.64	0	20.07/86.3%	3.18/13.7%	21.86/76.3%	6.80/23.7%	24.43/68.3%	11.33/31.7%
128 Bits	15.85	0	28.57/87.5%	4.08/12.5%	30.86/76.6%	9.43/23.4%	32.61/68.2%	15.20/31.8%

tile region, thereby reducing the available space for *RS* placement. However, the standard cell area of *RSs* is significantly smaller than that of the channel, rendering the impact on area utilization negligible. In all of our experiments, the area occupied by *RSs* averaged less than 2% of the channels. As a result, *RSs* can still be easily embedded within the tile area originally designated for channels in a channel-dedicated NoC. Moreover, over-the-cell NoC architectures impose stricter routing constraints, potentially leading to greater wiring overhead compared to the negligible cost of standard-cell *RS* placement.

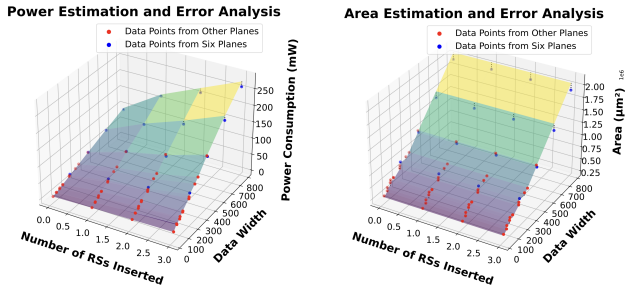
Fig. 7 illustrates the trend of the maximum operating frequency across different NoC configurations, varying in data width and number of planes, along with an approximate function. In an ideal scenario, the delay between components is proportional to their physical distance after optimal buffer insertion. When wire delays have been optimized but the NoC still fails to achieve the desired frequency, *RSs* can be introduced to reduce the effective communication distance by pipelining data transmission. Although *RS* insertion improves timing by shortening wire segments, it does not lead to a strictly linear delay reduction due to the additional wiring logic associated with each *RS*. This effect can be modeled by the equation $F = (\alpha R+1)F_{base}$, where F_{base} is the original NoC operating frequency before *RS* insertion, and α represents the degradation factor. We also observe that the positive impact of *RS* insertion on the timing closure of a NoC comes with diminish returns: each newly inserted *RS* is less effective than the previous one. Therefore, a correction factor β is introduced, resulting in the revised formula $F = (\alpha(1-\beta R)R+1)F_{base}$. We solve the Least Squares Problem to estimate the parameters α and β, which define our approximation function.

The performance gain achieved by inserting *RSs* introduces additional power consumption, as reported in TABLE I. The power consumption of both the routers and the *RSs* increase after *RS* insertion due to dynamic power consumption, which



(a) Power consumption of *RSs*. (b) NoC power distribution.

Fig. 8. Power overhead of *RS* insertion.



(a) (b)

Fig. 9. Estimation functions and real data points.

is directly proportional to the operating frequency. This effect is analogous to increasing data parallelism: both doubling the frequency and doubling data parallelism result in a $2\times$ increase in bandwidth, accompanied by a $2\times$ increase in power. Therefore, the real power overhead of wire pipelining comes from the additional standard cells of the *RSs*. Fig. 8a illustrates the quadratic-like trend of *RS* power overhead. While not perfectly quadratic due to timing degradation in *RS*, the trend is primarily driven by the simultaneous increase in both the number of *RSs* and the operating frequency.

Fig. 8b details the power consumption of routers and *RSs* as a function of the NoC frequency, using the 32-bit NoC as an example. While the power consumption of the routers scales linearly with frequency due to additional dynamic power consumption, the power overhead from RS insertion grows super-linearly with frequency, as more *RSs* need to be inserted to achieve higher frequencies.

C. Physical Resource Estimation

The power and area impacts of channel parallelism and wire pipelining can be analyzed as follows. For power estimation, we observe a similar relationship for routers and *RSs*: their power consumption is generally proportional to the operating frequency F and the data width D. Therefore, their power consumption can be approximated as $const \times F \times D$. On the other hand, the area estimation can be derived based on two channel size constraints mentioned in Section III-A. For a 2D-mesh NoC, it can be expressed as $const \times (2L_{chip}C + C^2)$, where C denotes the channel size and L_{chip} is the semiperimeter of the chip excluding the channel region. These estimation functions greatly reduce design space exploration time for the NoC PD by eliminating the need to run EDA tools for

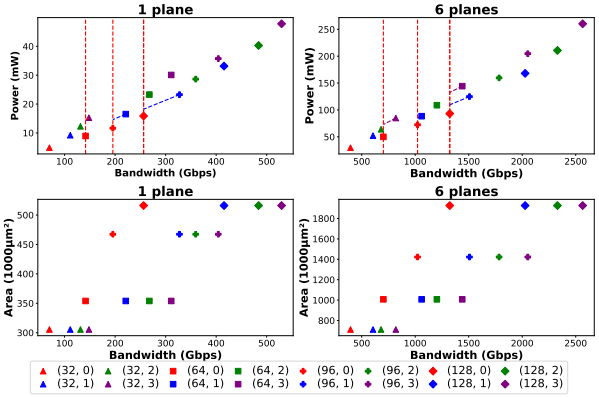


Fig. 10. Power and area consumption of different NoCs.

each configuration. For example, completing the PD of a six-plane NoC with a 512-bit data width can take over five days on a server equipped with Intel Xeon Gold 6258R CPUs, utilizing 24 cores and 196 GB of memory. Such a task would be daunting if one aims to explore or optimize NoC configurations extensively.

As a case study, we collected PD data from one-, two-, and three-plane NoC configurations with varying parameters, averaging the results to construct the estimation functions of NoC physical resources. Fig. 9 shows the accuracy of the estimation functions for power and area, evaluated against data obtained from actual PD measurements. The planes in the figure represents the estimation function, while the data points denote the measured values. The average error between the power estimation and the measured data is 3.6%, whereas the average error for the area estimation is 6.4%. The size of the 32-bit router is slightly larger than estimated due to components that are not scalable with data width. Note that none of the six-plane NoC data was used in deriving these estimation functions. Despite this, the power and area estimations show average errors of 5% and 6% respectively, which are within tolerance. These data points highlight the accuracy and scalability of the estimation function.

D. Trade-Offs between Power and Area

We use one-plane and six-plane NoCs as illustrative examples to highlight the trade-offs between wire pipelining and channel parallelism, as shown in Fig. 10. The first row illustrates the power consumption of NoC systems, while the second row represents their area consumption. The legend presents the NoC configurations in tuple format, where the first value indicates the data width of the NoC, and the second value represents the number of *RSs* inserted. NoC configurations with the same data width share the same marker shape. For example, NoCs with a 32-bit data width are represented by triangles. Configurations with the same number of *RSs* inserted are distinguished by color; for instance, all NoCs with one *RS* inserted are colored blue.

For a single-plane NoC configuration, we can achieve similar bandwidth of a 64-bit NoC by using a 32-bit data width NoC with three *RSs* in each link between routers. This configuration incurs a 63% power overhead while saving 16%

TABLE II
OPTIMIZATION PROBLEM PARAMETERS.

Parameter	Variable type	Description
C	Decision variable	Channel size (μm) of the chip design.
D	Decision variable	Data width ($bits$).
F	Decision variable	Operating frequency of the NoC (*MHz*).
R	Decision variable	Number of *RSs* inserted.
A_u, P_u	User defined variable	Power and area budgets specified by users.
B_{req}	User defined variable	Target NoC bandwidth specified by users.
L_{chip}	Constant	Semiperimeter of the chip excluding the channel size.
F_{base}	Constant	Maximum operating frequency of the NoC before *RS* insertion.
F_{router}	Constant	Maximum operating frequency of the router.
α, β	Constant	The coefficients for *RS* effect on the NoC maximum frequency.
γ	Constant	The coefficient of the router constraint.
δ	Constant	The coefficient of the channel-parallelism constraint.
a, b, c	Constant	Adjusting factors for power consumption and area of the NoC.
k	Constant	Adjusting factor for the bandwidth calculation of the NoC.

in area overhead. Similarly, achieving the bandwidth of a 96-bit NoC is possible with a 64-bit data width NoC by inserting one *RS* in each link, resulting in a 26% power overhead but avoiding a 32% area overhead associated with increasing the NoC's parallelism. To match the bandwidth of a 128-bit NoC, two options are available: adding one *RS* to a 96-bit NoC or inserting two *RSs* into a 64-bit NoC. These configurations incur 15% and 41% additional power overhead, while saving 10% and 45% in area overhead, respectively.

Similarly, in a six-plane NoC configuration, a 32-bit NoC can achieve the bandwidth of a 64-bit NoC by inserting three *RSs* into the channel wires. This results in an additional 46% power consumption but avoids a 42% area overhead. To achieve the bandwidth of a 96-bit NoC, we can add one *RS* to a 64-bit NoC, incurring a 18% power overhead and avoid an additional 41% area overhead. Finally, to achieve the bandwidth of a 128-bit NoC, we can incur either a 17% or 42% additional power overhead, depending on whether we insert *RSs* into a 96-bit or 64-bit NoC, respectively, to avoid 36% and 91% additional area overhead.

In the single-plane NoC configuration, the maximum area savings is 45%, in contrast to the 91% reduction achieved in the six-plane configuration. This difference arises because, unlike the six-plane configuration, where a linear relationship between data width and channel size is observed, the channel size in the single-plane case follows the router constraint due to limited data parallelism. The trade-off between power and area becomes more pronounced in NoCs with larger bandwidth, particularly as the area increases linearly with channel size. The power penalty of wire pipelining is slightly higher for NoCs with fewer planes or narrower data widths. This is in part due to other logic elements in the SoC, such as clock trees, which do not scale with bandwidth.

IV. OPTIMIZATION OF NOC CONFIGURATIONS

Channel parallelism and wire pipelining can be applied either as alternatives or in a complementary manner to optimize NoC design. In this section, we first formulate a NoC optimization problem and then present a case study to demonstrate the benefits of combining both techniques.

A. The Optimization Problem

In Section III-B we determine that *RS* insertion introduces zero silicon area overhead. Hence, to minimize the silicon area, we increase the channel parallelism only after inserting as many *RSs* as possible. In contrast, to minimize power consumption, maximizing channel parallelism is the preferred strategy because the NoC achieves its highest power efficiency when no *RSs* are used.

However, when architects have specific PPA requirements, the optimization problem must be formulated to accommodate the constraints of the SoC design. In our formulation, architects can define their *power budget* $\mathbf{P_u}$ and *silicon area budget* $\mathbf{A_u}$. The non-convex *Mixed-Integer Quadratically Constrained Programming* (MIQCP) problem is defined as follows, with parameters and variables listed in TABLE II.

$$\text{minimize} \quad \frac{a \times R \times F \times D + b \times F \times D}{P_u} + \frac{c \times (2L_{chip}C + C^2)}{A_u}. \tag{8}$$

$$\text{subject to} \quad k \times F \times D \geq B_{req}, \tag{9}$$

$$(\alpha(1 - \beta R)R + 1)F_{base} \leq F_{router}, \tag{10}$$

$$C^2 \geq \gamma \times D, \tag{11}$$

$$C \geq \delta D, \tag{12}$$

$$C \geq 0, D \geq 0, R \geq 0, F \geq 0, \tag{13}$$

$$c \times (2L_{chip}C + C^2) \leq A_u, (aR + b) \times F \times D \leq P_u, \tag{14}$$

$$F \leq (\alpha(1 - \beta R)R + 1)F_{base}. \tag{15}$$

Decision variables. There are four decision variables C, D, F, R in the optimization problem. D and R, which are related to the architecture of the system, are integer variables. C and F, which pertain to the resulting PD values, are continuous variables. If the system requires a specific data width setting, Special Ordered Sets of Type 1 [18] can be introduced into the problem formulation.

Objective function (8)**.** The objective function considers both the power and area overhead introduced by wire pipelining and channel parallelism. As discussed in Section III-C, the power consumption of both *RSs* and routers is positively correlated to $F \times D$, while the area overhead introduced by the channel parallelism is positive related to $2L_{chip}C + C^2$.

Bandwidth constraint (9)**.** When designing NoCs, architects typically define the required bandwidth to ensure optimal performance. To meet the requirements, the achieved bandwidth must exceed the specified requirement.

Router frequency constraint (10)**.** In Section III-B, we derived an approximate function for the maximum NoC operating frequencies with *RS* insertion.[2] As the NoC maximum operating frequency increases, it eventually becomes limited by the maximum frequency of the routers (F_{router}).

Channel constraints with respect to the data width (11)**,** (12)**.** As derived in Section III-A, the channel size is constrained by both router and channel-parallelism constraints.

Nonnegative constraints (13)**.** All decision variables are nonnegative.

Budget constraints (14)**.** The power and area consumption of the NoC must remain within the allocated budgets.

Frequency constraint (15)**.** The operating frequency of the NoC is constrained by the maximum frequency limits set by

[2]To make the function tractable for the optimization solver, we applied piecewise linearization to the function.

TABLE III
COMPARISON OF POWER AND AREA RESULTS FOR NOC DESIGNS UNDER DIFFERENT SCENARIOS AND APPROACHES.

Scenario	Power constrained	Area constrained	Power and area constrained	Power and area sufficient
Power and area budgets	(P_u : 85, A_u : 1,920,000)	(P_u : 130, A_u : 960,000)	(P_u : 100, A_u : 1,440,000)	(P_u : 160, A_u : 2,560,000)
Channel-parallelism approach	(P : 83, A : 1,533,190)	(P : −, A : −)	(P : −, A : −)	(P : 83, A : 1,533,190)
Wire-pipelining approach	(P : −, A : −)	(P : 121, A : 791,513)	(P : −, A : −)	(P : 121, A : 791,513)
Hybrid approach	(P : 83, A : 1,533,190)	(P : 121, A : 791,513)	(P : 96, A : 1,096,200)	(P : 108, A : 895,371)
Channel-parallelism approach	(P : 97.6%, A : 79.9%)	(P : −, A : −)	(P : −, A : −)	(P : 51.9%, A : 59.9%)
Wire-pipelining approach	(P : −, A : −)	(P : 93.1%, A : 82.4%)	(P : −, A : −)	(P : 75.6%, A : 30.9%)
Hybrid approach	(P : 97.6%, A : 79.9%)	(P : 93.1%, A : 82.4%)	(P : 96%, A : 76.1%)	(P : 67.5%, A : 35.0%)

TABLE IV
OPTIMIZATION RESULTS OF NOC CONFIGURATIONS UNDER DIFFERENT SCENARIOS.

Scenario	Power constrained	Area constrained	Power and area constrained	Power and area sufficient
Channel-parallelism approach	(D : 98, R : 0)	(D : −, R : −)	(D : −, R : −)	(D : 98, R : 0)
Wire-pipelining approach	(D : −, R : −)	(D : 52, R : 3)	(D : −, R : −)	(D : 52, R : 3)
Hybrid approach	(D : 98, R : 0)	(D : 52, R : 3)	(D : 72, R : 1)	(D : 58, R : 2)

the router microarchitecture and the delays introduced by long wiring links within the NoC.

B. Optimization Result

We demonstrate the use of this model under four different scenarios: 1) power-constrained, 2) area-constrained, 3) balanced but limited power and area budgets, and 4) balanced with sufficient budgets. For each scenario, we compare three approaches: 1) channel-parallelism-only, 2) wire-pipelining-only, and 3) hybrid approach. As a case study, we consider a six-plane NoC system, targeting a bisection bandwidth of 1300 Gbit/sec. *Gurobi* [19] was used to solve the optimization problems, and the results under different power and area constraints are presented in TABLE III and TABLE IV.

For the channel-parallelism approach, increasing the data width is the only way to improve performance. In area-constrained environments, the approach is restricted in achieving the target bandwidth. In contrast, wire pipelining is well-suited for area-constrained scenarios since *RS* insertion does not increase silicon area. However, the power overhead of *RSs* grows approximately quadratically with operating frequency. Consequently, this approach is not suited for power-constrained scenarios.

In both power-constrained and area-constrained scenarios, the hybrid approach consistently identifies the most efficient NoC configurations, which align with those selected by the wire-pipelining and channel-parallelism methods, as they represent the most effective means of achieving the target bandwidth. When both power and area are constrained, the hybrid approach demonstrates clear advantages by combining both strategies to meet the bandwidth target, whereas the individual approaches fail to do so.

Finally, when area and power budgets are sufficient, all three approaches are capable of achieving the desired bandwidth. Among them, the hybrid approach offers the advantage of minimizing overall resource usage by jointly considering power and area constraints. As shown in the last column of TABLE III, all approaches meet the bandwidth target: the channel-parallelism approach consumes approximately 51.9% of the power budget and 59.9% of the area budget, averaging 55.9% in overall resource utilization; the wire-pipelining approach uses 75.6% of the power budget and 30.9% of the area budget, resulting in 53.25% utilization on average. In comparison, the hybrid approach requires 67.5% of the power budget and 35.0% of the area budget, achieving the lowest average resource utilization at 51.25%. These results highlight the hybrid method's ability to balance resources efficiently while meeting performance requirements.

In conclusion, the hybrid approach provides the best solution with two key advantages:

1) It enables the NoC design to achieve higher bandwidth, particularly when both power and area are constrained.
2) It enables better trade-offs between resource usage and performance for a given power and area budget.

V. RELATED WORK

SoC physical design and optimization. In contrast to the NoC performance studies in earlier years [20], [21], which focus on simulation-based results, numerous works [2]–[4] have recently been more focused on tape-out results, primarily adopting 2D-mesh topologies. However, these studies primarily focus on system implementation and optimization, rather than exploring the trade-offs from a physical design perspective. In contrast, our work complements these efforts by highlighting the physical design trade-offs involved.

NoC pipelining. Pipelining data transmission is a commonly used technique to improve the system throughput. Most prior research on NoC pipelining focused on router microarchitecture pipelining to increase operating frequency. [22]–[27]. However, in advanced technologies, the critical path of the NoC shifts from the routers to the links between them during the downstream physical design flow. Our work complements these studies by focusing on wire pipelining to achieve higher frequencies until the NoC frequency becomes

limited by the router. For link-level wire pipelining in NoCs, Gebhardt et al. [28] proposed an optimization methodology based on modeling and simulation techniques, with a focus on asynchronous NoCs. In contrast, our work emphasizes the trade-offs between wire pipelining and channel parallelism in the physical design of synchronous latency-insensitive NoCs.

NoC channel parallelism. Increasing channel parallelism improves NoC performance, particularly in latency-critical scenarios such as configuration or register planes. Fischer et al. [8] demonstrate the effectiveness of this technique. Our work complements theirs by introducing wire pipelining as an additional dimension for NoC throughput optimization and by analyzing the trade-offs between the two approaches.

VI. Conclusion

In this work, we explored the physical design implications of increasing bandwidth in 2D-mesh NoC implementations. We focused on two techniques: 1) wire pipelining with *RSs*, and 2) increasing channel parallelism. We implemented multiple NoC tape-in designs with varying configurations in a commercial $12nm$ technology process using industry-standard EDA tools, enabling accurate modeling and analysis of the trade-offs of each approach. The experimental results show that when targeting the same bandwidth, we can save up to 91% in silicon area overhead by choosing wire pipelining over channel parallelism, while increasing power consumption overhead by 42%. In contrast, increasing channel parallelism can save up to 63% in power overhead compared to wire pipelining, while increasing area overhead by 16%. To address this trade-off, we formulate an optimization problem where these two techniques can be strategically combined based on user requirements. To the best of our knowledge, this is the first paper that provides a comprehensive PPA analysis of both techniques supported by real tape-in results and that offers guidance on selecting the most suitable configuration based on specific design constraints.

References

[1] W. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proc. of the Design Automation Conference (DAC)*, 2001, pp. 684–689.

[2] S. Davidson *et al.*, "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, 2018.

[3] T. Jia *et al.*, "A 12nm agile-designed SoC for swarm-based perception with heterogeneous IP blocks, a reconfigurable memory hierarchy, and an 800MHz multi-plane NoC," in *European Solid State Circuits Conference (ESSCIRC)*, 2022, pp. 269–272.

[4] D. Jung *et al.*, "Scalable, programmable and dense: The hammerblade open-source RISC-V manycore," in *Proc. of the Int'l Symp. on Computer Architecture (ISCA)*, 2024, pp. 770–784.

[5] A. Gonzalez *et al.*, "A 16mm2 106.1 GOPS/W heterogeneous RISC-V multi-core multi-accelerator SoC in low-power 22nm FinFET," in *European Solid State Circuits Conference (ESSCIRC)*, 2021, pp. 259–262.

[6] B. Dally, "Reflections on 21 years of NoCs," Keynote at Int'l Network-on-Chip Symp. (NOCS), 2022.

[7] B. Jonathan *et al.*, "OpenPiton: an open source manycore research framework," in *Proc. of the Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2016, p. 217–232.

[8] T. Fischer *et al.*, "FlooNoC: A Multi-Tb/s Wide NoC for Heterogeneous AXI4 Traffic," *IEEE Design & Test*, vol. 40, no. 6, pp. 7–17, 2023.

[9] G. Yeap, "Smart mobile SoCs driving the semiconductor industry: Technology trend, challenges and opportunities," in *Int'l Electron Devices Meeting (IEDM)*, 2013, pp. 1.3.1–1.3.8.

[10] L. P. Carloni *et al.*, "Theory of latency-insensitive design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 9, pp. 1059–1076, 2001.

[11] Y. Yoon *et al.*, "Virtual channels and multiple physical networks: Two alternatives to improve noc performance," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 32, no. 12, pp. 1906–1919, 2013.

[12] C. Glass and L. Ni, "The turn model for adaptive routing," in *Proc. of the Int'l Symp. on Computer Architecture (ISCA)*, 1992, pp. 278–287.

[13] P. Mantovani *et al.*, "Agile SoC development with Open ESP," in *Proc. of the Int'l Conf. on Computer Aided Design*, 2020.

[14] V. Nagarajan *et al.*, *A Primer on Memory Consistency and Cache Coherence: Second Edition*. Morgan & Claypool, 2020.

[15] L. P. Carloni, "From latency-insensitive design to communication-based system-level design," *Proceedings of the IEEE*, vol. 103, no. 11, pp. 2133–2151, Nov. 2015.

[16] D. Matzke, "Will physical scalability sabotage performance gains?" *Computer*, vol. 30, no. 9, pp. 37–39, 1997.

[17] Y. Shao *et al.*, "Simba: Scaling deep-learning inference with multi-chip-module-based architecture," in *Proc. of the Int'l Symp. on Microarchitecture (MICRO)*, 2019, p. 14–27.

[18] C. Guéret *et al.*, "Applications of optimization with Xpress-MP," *contract*, 1999.

[19] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2025. [Online]. Available: https://www.gurobi.com

[20] P. P. Pande *et al.*, "Performance evaluation and design trade-offs for network-on-chip interconnect architectures," *IEEE Trans. Comput.*, vol. 54, no. 8, pp. 1025–1040, 2005.

[21] R. Marculescu *et al.*, "Outstanding research problems in NoC design: System, microarchitecture, and circuit perspectives," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 28, no. 1, pp. 3–21, 2009.

[22] P. Zhou *et al.*, "NoC frequency scaling with flexible-pipeline routers," in *Proc. of the Int'l Symp. on Low Power Electronics and Design (ISLPED)*, 2011, pp. 403–408.

[23] Z. Zhang and X. Hu, "A novel pipelining scheme for network-on-chip router," in *Proc. of the Int'l Symp. on Intelligent Information Technology Application (IITA)*, vol. 2, 2009, pp. 372–375.

[24] B. Chemli and A. Zitouni, "Design and evaluation of optimized router pipeline stages for network on chip," in *Proc. of the Int'l Image Processing, Applications and Systems (IPAS)*, 2016, pp. 1–5.

[25] A. Psarras *et al.*, "Shortpath: A network-on-chip router with fine-grained pipeline bypassing," *IEEE Trans. Comput.*, vol. 65, no. 10, pp. 3136–3147, 2016.

[26] B. Daya *et al.*, "Quest for high-performance bufferless NoCs with single-cycle express paths and self-learning throttling," in *Proc. of the Design Automation Conference (DAC)*, 2016, pp. 1–6.

[27] A. Ejaz *et al.*, "FreewayNoC: a DDR NoC with pipeline bypassing," in *Int'l Symp. on Networks-on-Chip (NOCS)*, 2018, pp. 1–8.

[28] D. Gebhardt *et al.*, "Link pipelining strategies for an application-specific asynchronous NoC," in *Int'l Symp. on Networks-on-Chip (NOCS)*, 2011, p. 185–192.