An Architectural Power Model for Networks on Chip

Animesh Agrawal

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2023-168
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2023/EECS-2023-168.html

May 12, 2023

An Architectural Power Model for Networks on Chip

by

Animesh Agrawal

A thesis submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Borivoje Nikolic, Chair
Professor Sophia Shao

Spring 2023

The thesis of Animesh Agrawal, titled An Architectural Power Model for Networks on Chip, is approved:

Chair ______________________ Date May 11, 2023.

______________________ Date 5/10/2023

______________________ Date ______________

University of California, Berkeley

An Architectural Power Model for Networks on Chip

Abstract

An Architectural Power Model for Networks on Chip

by

Animesh Agrawal

Master of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Professor Borivoje Nikolic, Chair

Motivated by the breakdown of Dennard scaling and current slowdown in CMOS scaling, SoC designers have increasingly embraced heterogeneity to meet power, performance, and area constraints. As SoC designs increase in scale, networks-on-chip (NoCs), have grown in prominence and complexity, becoming responsible for a significant fraction of an SoC's power consumption. However, NoCs are highly parameterizable and support a diverse set of constructions, making it crucial for designers to have access to feedback about a NoC design's power consumption early in the design phase.

In this thesis, we present an ML-based, workload aware, architectural power model for networks on chip (NoCs). We identify NoC architectural parameters most relevant to power consumption and construct a framework to generate a diverse training dataset of NoC configurations covering a range of power responses. We use the identified parameters and generated dataset to train a machine learning model capable of estimating NoC router power consumption and evaluate our model on realistic NoC designs routing simulated network traffic. Finally, we also present our progress towards constructing a NoC power model for NoC routers with heterogeneous channel configurations.

Contents

Contents **i**

List of Figures **iii**

List of Tables **v**

1 Background and Project Introduction **1**
1.1 Background 1
1.1.1 Networks on Chip (NoCs) 1
1.1.2 Constellation 4
1.2 Prior Work 6
1.2.1 Power Estimation Tools 6
1.2.2 Architectural Power Models 6
1.3 Project Goals 7

2 Implementation **9**
2.1 Building NoC Router Architectures 9
2.1.1 Training Architectures 9
2.1.2 Testing Architectures 10
2.2 Generating Traces on NoCs 10
2.2.1 Prior Approaches to Dynamic Activity 11
2.2.2 Trace-Generation with Traffic Simulation 11
2.2.3 Trace-Generation with Formal Tools 12
2.3 Power Estimation 12
2.4 Dataset Generation 13
2.5 Learning Model Construction 13
2.5.1 An Initial Model 14

3 A Power Model for Constellation NoCs **15**
3.1 Evaluating The Initial Model 15
3.1.1 Evaluation with Train-Test Split 15
3.1.2 Evaluation with Realistic Topologies 16

3.2 Identifying Additional Model Parameters . . . 16
3.2.1 Utilization Rate . . . 16
3.2.2 Flit Hamming Distance . . . 19
3.3 A Power Model for Constellation NoCs . . . 21
3.3.1 Model Evaluation . . . 21
3.3.2 Model Usability . . . 23

4 Heterogeneous NoC Router Power Modeling **24**
4.1 Router Power Consumption Decomposition . . . 25
4.1.1 Input and Ingress Unit Power Characteristics . . . 26
4.2 Constructing a Heterogeneous Router Power Model . . . 27
4.3 Component-specific Model Performance . . . 28

5 Conclusion **30**

Bibliography **31**

List of Figures

1.1 A NoC for a large, heterogeneous, SoC [20]. 2
1.2 NoC Router Node Architecture [4]. 3
1.3 Tree Topology [20] 4
1.4 2D Mesh Topology [20] 4
1.5 NoC traffic for Λ [4]. 5
1.6 Traffic Matrix Λ. 5
1.7 A NoC router (node 2) with 1 input channel and 1 ingress channel. 5

2.1 Example topology for power evaluation. 10
2.2 The dataset generation workflow. 13

3.1 Actual power consumption versus predicted power consumption for a gradient-boosted regressor evaluated on train-test split. 15
3.2 Power consumption as a function of utilization rate for a router in a large mesh-topology NoC. 17
3.3 Power consumption as a function of utilization rate for a single router node from a power evaluation topology. 18
3.4 Trace for data point 109 (high power consumption) 18
3.5 Trace for data point 179 (low power consumption) 18
3.6 Router power consumption when adjacent flits have a Hamming distance of 0. . 20
3.7 Router power consumption when adjacent flits have a Hamming distance of 2. . 20
3.8 Router power consumption when adjacent flits have a Hamming distance of 32. 20
3.9 Router power consumption when adjacent flits have a Hamming distance of 64. 20
3.10 Training dataset, evaluation dataset, and model predictions versus ingress utilization rate. 22
3.11 Training dataset, evaluation dataset, and model predictions versus input utilization rate. 22

4.1 Example heterogeneous NoC router 24
4.2 Power per-component for a router in a large torus network 25
4.3 Power per-component for a router in a large 2d mesh network 25
4.4 Power per-component for a router in a butterfly network 26
4.5 Power per-component for a router from the training dataset 26

4.6 Input and ingress unit power consumption in a line topology. 27
4.7 Input and ingress unit power consumption in a large mesh topology. 27
4.8 Actual vs predicted power consumption for NoC router ingress units. 28
4.9 Actual vs predicted power consumption for NoC router input units. 28

List of Tables

3.1 Performance of prior-work-based architectural learning models on realistic NoCs 16
3.2 Performance of architectural learning models on realistic NoCs after input port and ingress port utilization rates are separated. 19
3.3 Performance of our architectural learning model on realistic NoCs. 21
3.4 Model training and prediction time. 23

4.1 Performance of our architectural learning model built on a gradient-boosted regressor on NoC input and ingress units. 29

Acknowledgments

The work presented in this thesis would not have been possible without the support of many graduate students in the SLICE and BWRC labs. This project required the use of a significant number of tools, many of which I wasn't familiar with, but the support of other graduate students – and their breadth of expertise – made this project possible. I'd like to thank Jerry Zhao for his assistance with using Constellation to build and simulate NoCs, Daniel Grubb for setting up Joules for power estimation, and Tianrui Wei for helping me use Jasper for formal trace generation. I'd also like to thank Jerry Zhao, Vighnesh Iyer, and Bora Nikolic for advising this work, and Sophia Shao for reading and approving this thesis. The work presented here would not have been possible without their direction.

Research, as a process, can sometimes feel like a tunnel without an end. At times like this, it helps to have friends who understand the processes's trials and tribulations. Thank you to Ella Schwarz, Franklin Huang, Jennifer Zhou, Raghav Gupta, Reza Sajadiany, and all the other students who made SLICE a supportive environment by sharing their own struggles and collaboratively celebrating success.

Finally, I'd like to thank Apple for generously funding the Apple Masters Scholarship program.

Chapter 1

Background and Project Introduction

Power efficiency is critical for modern systems-on-chip (SoCs). With the end of Moore's law, architects are increasingly relying on heterogeneity to balance performance and power targets, resulting in ever-growing SoCs with dozens of specialized functional units. In building these high-complex and interconnected SoCs, data transfer between components quickly becomes a limiting factor, with traditional bus-based interconnects unable to meet performance requirements. Networks-on-chip (NoCs) have emerged as a solution to this challenge, providing scalable, low-latency, and high-bandwidth communication.

However, NoCs are often responsible for a significant fraction of an SoC's power consumption. For application-specific SoCs running communication-heavy applications, NoCs can comprise nearly 40% of the SoC's overall power consumption [5]. Further, as fabrication processes improve, NoC power consumption decreases slower than compute and memory components [3]. Thus, accurate power characterization and estimation for networks-on-chip is crucial.

1.1 Background

1.1.1 Networks on Chip (NoCs)

An interconnection network, broadly, provides a programmable system that transports data between end points, formally known as terminals. For on-chip interconnection networks, these terminals can be memory arrays, system registers, CPU/GPU cores, and programmable accelerators [4].

SoC interconnects have historically been a system of shared buses that connect all terminals with a shared channel. Each message transmitted over the bus is received by all terminals in serial order, even when the message is intended only for a single recipient terminal. Replies from the recipient are again sent over the shared channel, creating a serial ordering of all interconnect communication. Shared bus interconnects have the benefits of being easy to implement and inexpensive to manufacture. However, they do not scale to large SoCs. In addition to physical limitations that prevent large buses from operating at high-speeds, aggregate bandwidth demand scales with the number of SoC components. Further, messages issued on the interconnect are often intended for a single terminal, for example when a processor requests a value from a memory bank. Since shared buses provide only a single channel, only one such point-to-point message can be sent at a time, posing an unnecessary serial constraint on a large system. At the same time, however, dedicated wiring between each pair of terminals would unnecessarily waste limited wiring resources as terminals communicate only sporadically.

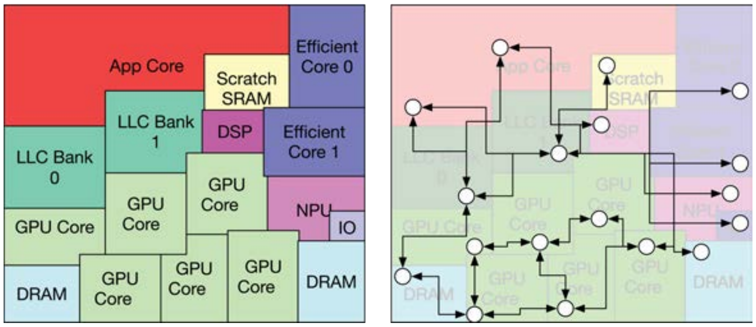

Figure 1.1: A NoC for a large, heterogeneous, SoC [20].

Networks-on-chip aim to address these limitations by compromising between the two approaches. NoCs consist of a collection of router nodes that connect to terminals and other router nodes via channels. Unlike dedicated wiring, the network is shared by terminals, freeing up already scarce space on the chip.

Compared to the shared bus interconnection architecture, NoCs only deliver messages to the intended destination terminals, removing the need for terminals to process and discard messages not intended for them. Further, multiple messages from different terminals can be resident within the NoC at the same time, increasing communication bandwidth and improving scaling.

To communicate over the NoC, terminals send messages as packets of data. To ensure fair allocation of its limited resources, NoCs may also offer quality-of-service (QoS) features to ensure packets meeting specific requirements are guaranteed a certain level of performance. Conversely, NoCs may also offer certain classes of packets only best-effort guarantees; packets in these service classes can experience arbitrary delay and may even be dropped by the network. Within a NoC, packets are decomposed into small, fixed-size, flits to simplify routing logic.

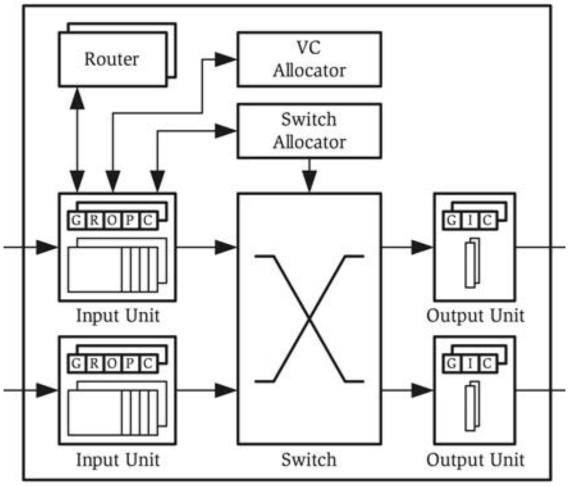

Figure 1.2: NoC Router Node Architecture [4].

Router Nodes

As shown in Figure 1.2, each channel into a router node connects to an input unit within the router node. When an input unit receives a flit sent along the channel into the router node, the flit is buffered in a FIFO queue. There, it waits for, among other things, routing logic to compute the packet's next hop in the network and the downstream network components to become available. To prevent an input unit from stalling when a flit's downstream node is not available, multiple virtual channels (VCs) can be multiplexed onto a single physical channel. When the flit at the head of a virtual channel's buffer is ready and able to be routed to its next destination, the virtual channel requests allocation of the physical channel. If the request is granted, the flit is sent to the switch.

The switch connects input units to output units, which forward the flit to the next router node along the path to its destination terminal. Similar to input units, output units contain logic to manage virtual channels and allocation [4].

Building a NoC

When designing a NoC, architects can vary a number of parameters to customize the NoC's performance, power consumption, and capabilities. Below, we present some major parameters in a non-comprehensive fashion.

At the network level, NoC designers can customize the NoC topology, which specifies the static arrangement of router nodes and channels within the interconnection network (examples in Figures 1.3 and 1.4). More interconnected topologies have greater routing bandwidth and can transmit packets to their destination with fewer hops through the network, but also consume more space on the SoC and have a greater power consumption.

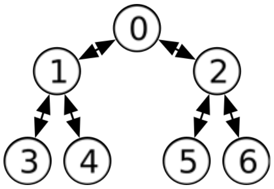

Figure 1.3: Tree Topology [20]

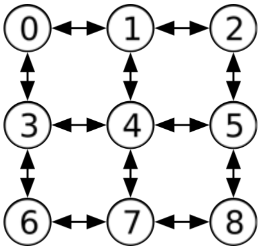

Figure 1.4: 2D Mesh Topology [20]

At the router level, NoC designers can customize the number of virtual channels per input or output channel, the number of packets the switch can route per cycle, and whether requests from a virtual channel to acquire the physical channel and switch can happen in parallel. While designers can select the number of entries in each input and output unit's buffers, the number of physical input and output units are implicitly decided by the topology, which governs the number of channels into and out of each router node.

Heterogeneous NoCs

Heterogeneous NoCs feature differing design parameters per sub-graph or per-router within the network [7]. At the network level, certain sub-graphs of router nodes could be configured with deeper buffers and more virtual channels per physical channel to, for example, better service connected terminals that are higher-bandwidth than the rest of the SoC. At the router level, certain NoC routers may contain channels and input units with differing numbers of virtual channels and buffer sizes. For example, such a router may be used to connect the more performant network sub-graph discussed earlier with the rest of the NoC.

1.1.2 Constellation

Constellation is a Chisel-based [2] NoC generator framework that is capable of synthesizing highly-irregular and heterogeneous NoCs based on a high-level specification provided by the SoC architect. Constellation includes a diverse set of pre-constructed network topologies and routing algorithms and produces synthesizable RTL. To evaluate a constructed NoC, Constellation includes a traffic injection and measurement framework capable of simulating user-specified traffic and reporting granular statistics on the NoC's throughput and latency. While the generated NoCs are communication protocol agnostic, Constellation provides protocol converters that allow Constellation-generated NoCs to be integrated into larger systems on chip [20].

Chipyard is a comprehensive SoC design, simulation, and implementation environment [1]. Constellation features plug-and-play integration with Chipyard, allowing for the simulation of entire SoCs containing Constellation-generated NoCs.

Constellation's Performance Evaluator

To benchmark and evaluate NoC performance, Constellation includes a C++ performance evaluator. To test a specific NoC configuration with a specific packet traffic pattern, users specify a flow matrix Λ where matrix entry λ_{ij} indicates the rate of traffic, or number of packets per cycle on average, from source terminal i to destination terminal j.

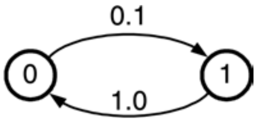

Figure 1.5: NoC traffic for Λ [4].

$$\begin{bmatrix} 0 & 0.1 \\ 1 & 0 \end{bmatrix}$$

Figure 1.6: Traffic Matrix Λ.

The network depicted in Figure 1.5 has 2 flows: one from node 0 to node 1 and another from node 1 to node 0. According to the traffic matrix in Figure 1.6, flow 0 routes 0.1 packets per cycle (injected by node 0) on average, while flow 1 routes 1 packet per cycle (injected by node 1) on average.

Constellation's framework reports the NoC's throughput, or percentage of requested traffic that can be fulfilled at steady-state, and average latency for user-specified specified traffic patterns. If the NoC can service all of the traffic requested by the traffic matrix, Constellation reports a throughput of 1. Constellation's performance evaluation framework is primarily run within Chipyard, leveraging Chipyard's RTL simulation environment.

QoS Guarantees in Constellation NoCs

Constellation NoCs, at time of writing, do not offer performance classes or performance guarantees for routed packets. However, Constellation does guarantee that packets will eventually reach their destination once accepted into the network.

Flow Control

Router nodes in Constellation NoCs have two distinct connections to channels passing data into the router: Ingress ports and Input ports.

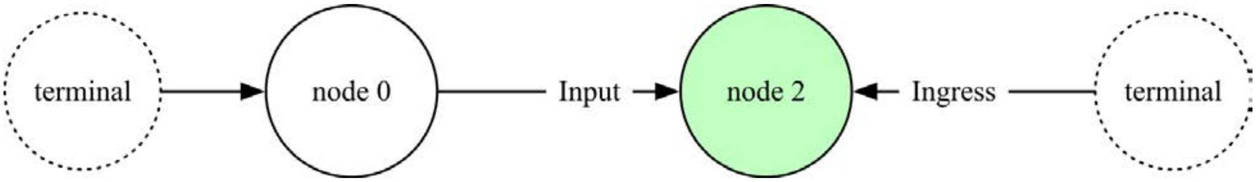

Figure 1.7: A NoC router (node 2) with 1 input channel and 1 ingress channel.

Ingress ports connect routers directly to network input terminals. The router communicates with the terminal via a ready-valid interface, where the terminal drives a `valid` signal, indicating a packet for the NoC is available, and the router drives a `ready` signal indicating the router is ready to receive a new packet – data is transferred when both `ready` and `valid` are driven. This allows the NoC to defer new traffic until the network has the capacity to buffer and route additional packets.

Input ports connect routers to other routers in the network, and flits transmitted along input channels are part of packets currently being routed by the network. Therefore, when an upstream node (for example node 0 in Figure 1.7) transmits a flit to a downstream node (node 2 in Figure 1.7), it is important to ensure the downstream node has the capacity to receive and buffer the transmitted flit so it is not dropped by the network. Constellation NoCs accomplish this with a credit-based flow control system, where upstream nodes receive a credit for every buffer entry found in the downstream node's receiving input unit. When an upstream node transmits a flit, it spends a credit, and when the downstream node routes said flit to its next hop in the network, a credit is returned to the upstream node. Thus, upstream nodes can never transmit more packets than the downstream node has the capability to receive.

1.2 Prior Work

1.2.1 Power Estimation Tools

Industry-standard state-of-the-art power estimation tools, such as Joules, replay simulation traces on gate-level representations of the hardware design and calculate the capacitive load and switching activity of each component [10, 15]. While these tools are highly accurate, they are also expensive, proprietary, and slow, largely due to the unavoidably large amount of memory and compute necessary to track the state of every gate in a large design.

Automatic proxy selection models for runtime introspection reduce the number of tracked signals by observing that signals with similar switching activity have similar effects on power consumption and selecting a small subset of signals to serve as proxies representing the entire design. Proxy selection models are highly accurate, with errors within 9% of industry-standard tools, while using as few as 0.05% of the available RTL signals [13, 18]. However, these models still require cycle-by-cycle simulation of all RTL signals, limiting their usefulness for pre-RTL design stage exploration.

1.2.2 Architectural Power Models

Architectural power models address these shortcomings by predicting a design's power consumption solely on its architectural parameters and high-level runtime statistics. Architectural models are component-specific, as they rely on assumptions about a design's microarchitectural construction to select appropriate input architectural parameters. Despite being less accurate than runtime-based power models and tools, architectural models allow designers to quickly verify their designs are approximately within power constraints and evaluate the power consumption implications of high-level design decisions. This allows designers to quickly narrow large design spaces and eliminate performant designs that don't meet power constraints [8].

Existing power models for NoCs primarily focus on modeling the power consumption of NoC routers because the power consumption of NoC channels varies significantly based on distance between the NoC router nodes they connect. This distance is often SoC-specific and determined at P&R time, making it difficult to accurately model.

Analytical architectural power models for NoC router nodes present mathematical equations to calculate the capacitance of individual NoC router components based on architectural parameters and aggregate runtime statistics, such as average Hamming distance between consecutive packets [12]. Learning-based architectural power models, in contrast, use trained models to predict post-synthesis power traces from architectural features that correlate with power consumption. When tested on datasets extracted from the training dataset, and on datapoints out of the domain of the training dataset, learning models have exhibited average errors of approximately 10% in their power estimations [11, 16]. Compared to analytical models, learning models can be automatically updated on new training data as improving manufacturing processes change the power relationship between architectural features. Learning models also do not require model developers to possess an intimate understanding of the component's physical implementation.

Prior work in architectural learning models for NoC router nodes identifies four key router architectural parameters in estimating power consumption: the number of input units, the number of virtual channels per physical channel, the number of buffer entries per channel, and flit width (the flit's size in bits). Prior work also identifies switching activity, or the frequency with which a signal toggles, as the relevant dynamic factor in estimating the router's power consumption [11].

1.3 Project Goals

Understanding the power consumption of NoCs is crucial when designing SoCs, and early approximation of NoC power consumption saves time both in design-space exploration and in working to meet physical constraints.

While existing architectural learning power models for NoC routers indicate acceptable performance on testing datasets derived from their training dataset, our exploration indicates significantly higher average errors when estimating the power of NoC nodes in larger networks or in irregular topologies. Existing power models also do not support heterogeneous routers with channels of varying specifications, limiting their coverage of the available design space. Finally, some existing power models for NoC routers are not workload-aware, returning a static power prediction regardless of the dynamic activity run on the NoC.

In this thesis, we present our work towards an improved architectural learning power model for NoCs, focusing on NoC routers in specific for reasons similar to those presented in prior work. We first discuss our work towards building a framework to generate NoC router training data off of RTL for entire NoCs and industry-standard state-of-the-art power estimation tools in an automated fashion. Then, we discuss cases where existing models struggle to provide accurate predictions and present modifications to existing work to improve the performance of models and make them workload-aware. Finally, we share our approach for building learning power models for irregular, highly heterogeneous, NoC routers.

Chapter 2

Implementation

Building an accurate learning-based architectural model requires a large training dataset of highly accurate power data. For NoCs, this data needs to capture both the domain of architectural parameters found in different NoCs and the dynamic factors found in the packets injected into and delivered by the NoC.

2.1 Building NoC Router Architectures

Static NoC power consumption varies significantly based on the NoC's architecture. More complex topologies require additional channels (and therefore more wiring) in between NoC routers, while more performant NoC routers contain additional virtual channels and buffer entries.

2.1.1 Training Architectures

Prior work identifies certain router-level architectural parameters as most relevant to power consumption:

- the number of input units to each router node
- the number of virtual channels per physical channel
- the number of buffer entries per channel
- flit size (in bits)

Prior work also provides a database of router architectures that vary these parameters [11]. We translated these architectures into Constellation configurations to create a dataset of NoC router architectures to train our learning model on.

To generate more realistic training data, we simulated the NoC routers of interest as part of a larger network instead of a standalone simulation. Router nodes with the same parameters as the test router are synthesized and connected to all but one of the NoC router's input ports; the remaining port is synthesized as an ingress port and connected directly to a network terminal. Output ports are constructed identically to input ports, with all but one of the NoC router's output ports connected to identically synthesized router nodes.

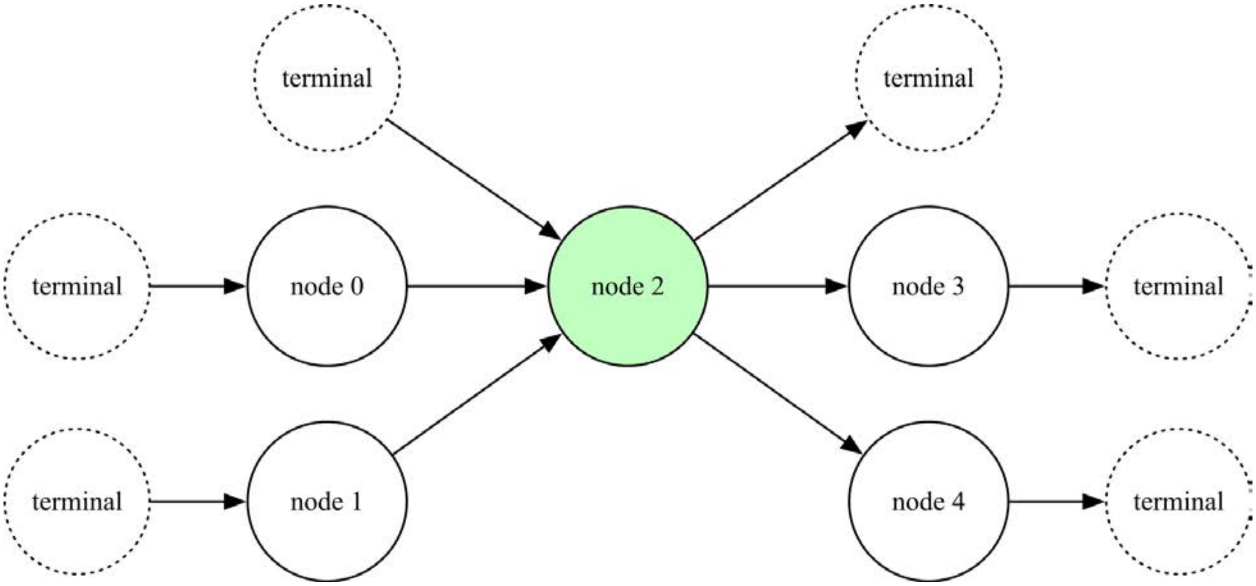

Figure 2.1: Example topology for power evaluation.

Figure 2.1 depicts an example network we use for generating training data, with Node 2 being the router node of interest. Two of Node 2's input ports are connected to other nodes in the network and the remaining ingress port is connected directly to a network terminal.

2.1.2 Testing Architectures

To evaluate model quality on real-world NoCs, we construct 10 NoCs of varying topologies, architectural parameters, and sizes. Tested topologies include 2D mesh, torus, and butterfly networks. Although the NoC is simulated as a complete system, relevant architectural parameters are extracted on a per-router basis and each router's power consumption is individually estimated by the learning model.

2.2 Generating Traces on NoCs

Dynamic NoC power consumption varies significantly based on the traffic carried by the NoC; an idle NoC would consume significantly less power than a fully saturated network.

2.2.1 Prior Approaches to Dynamic Activity

Prior work simulates the dynamic effects of packet traffic into a NoC by using power analysis tools to set switching activity for NoC router input signals, varying both how often input signals switch and the percentage of time each signal is set to 1 [11].

While faster than running gate-level simulations to capture realistic toggle rates, this approach can be inaccurate. First, the power analysis tools used in prior work use a zero-delay simulator to propagate user-annotated switching activity to the entire system; this method cannot capture glitching power, which can account for a nontrivial portion of a system's power consumption [17]. Further, power analysis tools rely on statistical assumptions when propagating user-annotated switching activity through the system, affecting accuracy due to issues related to signal correlation or re-convergence [19]. Finally, while annotation allows for comprehensive coverage of possible toggle rates, it can also generate toggle scenarios not found in realistic workloads.

2.2.2 Trace-Generation with Traffic Simulation

To avoid the inaccuracies present when using user-annotated switching activities, we used Constellation's traffic injection framework to simulate real packet traffic in our training dataset of NoCs. To capture representative samples of the traffic generated by real NoC workloads, we generate 2 classes of traffic matrices:

1. **Structured Traffic:** Each flow in the network, or entry in the traffic matrix Λ, is individually activated, with the rest of the flows in the network set to route 0 packets per cycle. The active flow is run at increasing flow rates until Constellation reports a throughput less than 1, indicating the network cannot service the requested traffic and the flow's saturating point has been found. After all individual flows are simulated, the methodology is repeated with all flows active.

2. **Random Traffic:** A global target flow rate is randomly pre-selected and each valid flow in the network is set to a random flow rate, collectively summing up to the selected global target. If Constellation reports a throughput less than 1, the datapoint is discarded because Constellation was unable to accept the entirety of the requested traffic and therefore did not achieve the requested utilization rate.

Simulations are run for 25,000 cycles, giving the network sufficient time to achieve steady-state performance. The granularity at which flow rates are incremented for structured traffic and the number of points gathered for random traffic are progressively increased until model performance no longer significantly improves.

We instrumented Constellation with performance counters to capture each channel's utilization rate, which we define to be the number of flits per cycle, on average, traversing the channel. We use utilization rate as an architectural-level abstraction for prior work's switching activity [14].

2.2.3 Trace-Generation with Formal Tools

We also explored the use of formal methods for traffic scenario generation. Random injection of traffic as presented in section 2.2.2 requires lengthy simulation of traffic matrices that may prove to request more traffic than the NoC can handle. Perhaps more importantly, edge-case utilization rates may be achievable with carefully constructed inputs that random traffic simulation may not generate.

To more rigorously examine utilization rates random traffic injection was not able to achieve, we embedded formal cover statements on specific utilization rates within the Constellation RTL and used JasperGold, an industrial formal verification tool, to generate minimal satisfying traces – input sequences that satisfy the cover property [9].

Cover Statement Construction

Constellation contains instrumentation that tracks the utilization, or number of flits entering, each input or ingress port within each NoC router. We add additional registers to separately aggregate the utilization across all input and ingress ports within each NoC routers.

We then embed cover statements requiring a specific utilization value for both ingress and input ports within a specific number of cycles; the latter constraint allows us to achieve a specific utilization rate, or specific number of flits per cycle on average. For example, to achieve a utilization rate of 0.1, we require the router receive 1 packet over 10 cycles. We also add constraints related to the Hamming distance between adjacent flits to ensure that flit content isn't set to 0 by the formal tool, as that would unrealistically decrease power consumption.

Formally-Generated Traces Evaluation

We find that formal trace generation is capable of achieving higher utilization rates than random traffic injection. We also observe that formally generated traces have similar power responses to traffic generated by Constellation's traffic simulator at utilization rates that Constellation's simulator is able to achieve, indicating that the formally-generated traces provide data that is representative of real traffic.

2.3 Power Estimation

In addition to traffic statistics, Constellation's injection framework provides a waveform from simulation, and we use Joules to estimate the NoC's average power consumption on these traces. Joules is the only power estimation tool with a built-in synthesis flow; all standard synthesis optimizations are performed on the provided RTL design and all power analysis is done on the post-synthesis gate-level design. While this approach takes longer than other power tools, it yields a significantly more accurate estimation of circuit power consumption [6, 10, 15].

Through Joules, we synthesize each NoC on the Intel 16 fabrication process, using typical parameters. We configure Joules to use `medium` effort to optimize leakage power and allow it to insert clock gating logic if doing so would reduce power consumption. We set a target clock period of 5ns, and configure synthesis to use `medium` effort to optimize the design and meet timing constraints. Average power across the simulation is computed.

2.4 Dataset Generation

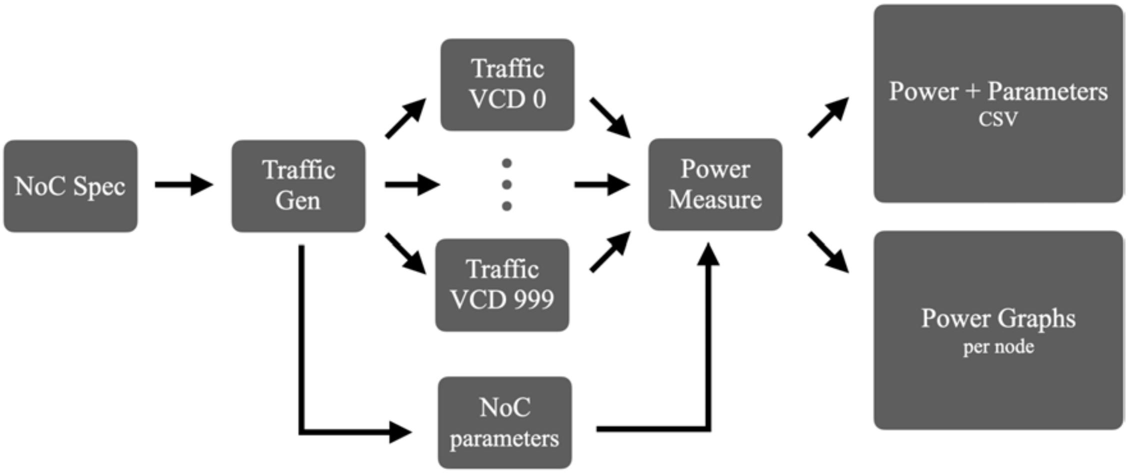

Figure 2.2: The dataset generation workflow.

We combined the approaches presented above to build a workflow to automatically generate model training datasets. Generated Constellation NoC configs are simulated using the evaluation harness to generate a series of simulation traces, which are passed into the Joules power estimation tool. Finally, the power data is combined with the NoC router's architectural parameters into a spreadsheet to create the dataset of training parameters and expected power responses. To sanity-check the data and visually understand power trends in the data, our workflow also graphs the power consumption of every router node in each specification as a function of utilization rate.

2.5 Learning Model Construction

Prior work uses a range of learning modeling techniques, often with differing modeling architectures [11, 14]. To recreate prior work and benchmark our improvements, we built a framework to post-process training data and flexibly train multiple models ranging from linear regression to gradient-boosted decision trees.

2.5.1 An Initial Model

Prior work in architectural learning models for NoC power estimation presents a model that uses

- **P:** the number of inputs into in the NoC router
- **V:** the number of virtual channels per physical channel
- **B:** the number of buffer entries per virtual channel
- **F:** flit size (in bits)

as input parameters and outputs a static prediction of average power consumption for the NoC router architecture [11]. However, as mentioned in section 2.2, we consistently find that NoC power consumption varies dramatically based on the traffic routed through the network. For a router in a Constellation-generated 2D mesh network, for example, power consumption when no traffic is present is less than half its peak power consumption when routing the maximum serviceable amount of traffic.

We decide this discrepancy in power responses necessitates building a workload-aware model. Based on the findings of prior work, we add utilization rate – the number of packets entering the NoC router per cycle, on average – as an additional parameter to the model [14]. For application-specific NoCs, the inclusion of dynamic parameters makes the model's predictions more relevant to the design scenario of interest. Conversely, if the NoC is not designed for a specific application, power predictions at a range of utilization rates can be averaged to recreate a workload-agnostic prediction.

In doing so, we construct and present a port of prior work in NoC learning architectural power models for the Constellation NoC generator platform. The inclusion of this model, when combined with Constellation's performance evaluation harness, allows designers to consider both performance and power consumption when designing a NoC in Constellation.

Chapter 3

A Power Model for Constellation NoCs

3.1 Evaluating The Initial Model

Using the methods described in Chapter 2, we recreate prior work on NoC router learning architectural models and modify the resulting model to be workload aware.

3.1.1 Evaluation with Train-Test Split

Prior work evaluates learning model performance by partitioning the dataset into training and test subsets [11]. When replicating this approach, our model performs similarly to prior work, with a normalized average error (NMAE) of 11.7%. When using a more advanced class of learning models, such as gradient-boosted decision trees, error further decreases to 8.8%.

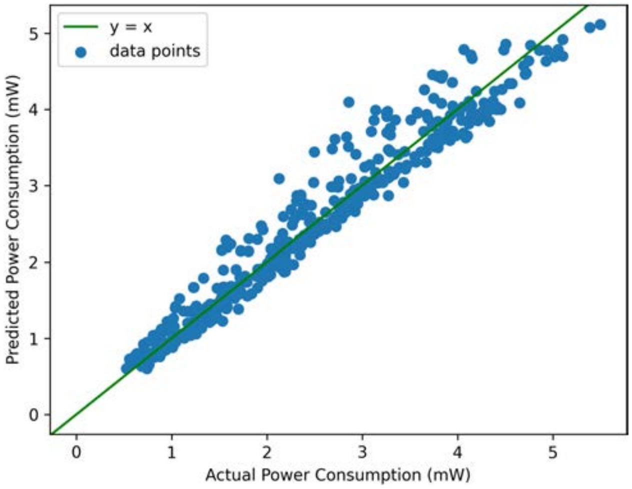

Figure 3.1: Actual power consumption versus predicted power consumption for a gradient-boosted regressor evaluated on train-test split.

In Figure 3.1, we see error is higher when the model overestimates power consumption, as opposed to when the model underestimates power consumption. While over-estimation of power consumption unnecessarily pushes designers towards more conservative architectures, resulting architectures still meet the set power ceilings. Under-estimation of power consumption, in contrast, risks producing designs that violate power consumption ceilings.

3.1.2 Evaluation with Realistic Topologies

While prior work indicates relatively low error when evaluating model performance via train-test split, the training data of individual NoC routers may not be representative of real-world NoC designs. To validate this hypothesis, we tested our recreation of models proposed by prior work on the testing NoC architectures presented in Section 2.1.2.

	Ransac Regressor	Gradient Boosting Regressor	SVM with RBF Kernel
MAPE	0.28	0.24	0.22
R^2	-0.24	0.49	0.10

Table 3.1: Performance of prior-work-based architectural learning models on realistic NoCs

On realistic NoCs, we find prior work performs noticeably worse than train-test split evaluation. The consistently low R^2 across models indicates the models' predictions are not significantly correlated with the evaluation dataset, either due to missing model parameters or due to the training dataset being unrepresentative of our evaluation dataset.

3.2 Identifying Additional Model Parameters

3.2.1 Utilization Rate

Prior work presents the number of packets entering a NoC router in each cycle, on average, as an architectural abstraction for circuit switching activity – we refer to this metric as utilization rate [14]. To validate utilization's rate effectiveness as an architectural abstraction for switching activity, we explored the relationship between utilization rate and power consumption for NoC routers of various configurations.

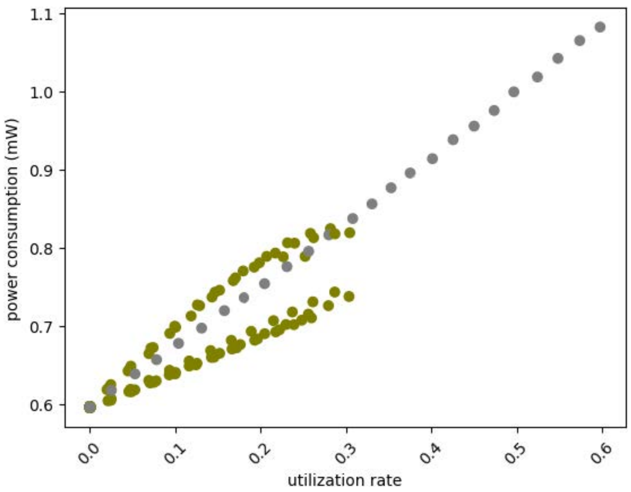

Figure 3.2: Power consumption as a function of utilization rate for a router in a large mesh-topology NoC.

Consistently, with one example shown in Figure 3.2, we find that there is no clear relationship between power consumption and utilization rate. Figure 3.2 uses data-points collected with the structured traffic approach presented in section 2.2.2; green data points represent traffic matrices with one flow active and gray data points represent traffic matrices with all flows active. While power consumption is linear with respect to utilization rate when all flows are active, traffic matrices with one flow active create two distinct non-linear plots. As a result, the same utilization rate can have three distinct power responses on the structured traffic generation approach.

We also witness a similar trend in data generated using random traffic, as shown in Figure 3.3. In random traffic scenarios, distinct curves are less pronounced because a random number of flows are active. Further, likely for the same reasons, the number of distinct power responses for similar utilization rates increases. In Figure 3.3, for example, up to 4 unique power responses for similar utilization rates are visible.

Utilization rate, or an equivalent, is the only dynamic factor considered by prior work. For data-points collected on the same NoC router architecture (as in Figure 3.2), then, utilization rate is the only differentiating factor for the learning model to predict power consumption with. To enable the model to differentiate between different power scenarios, additional model parameters are necessary.

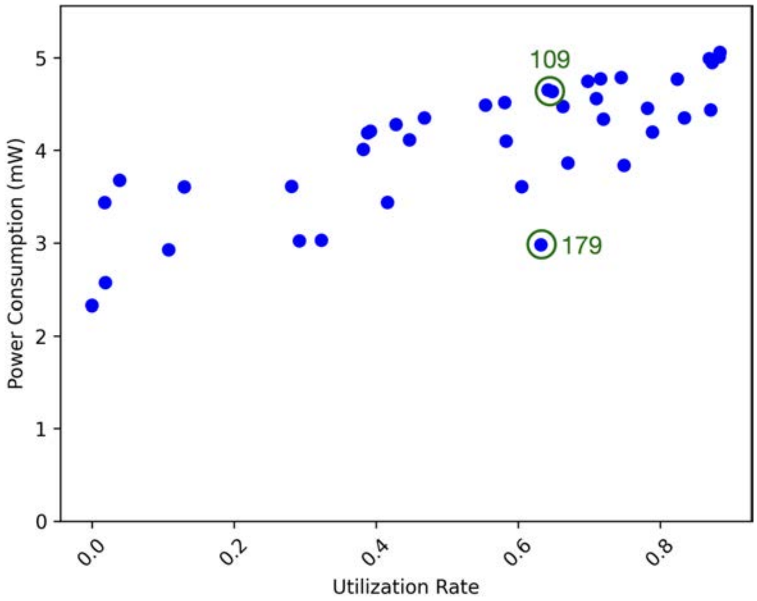

Figure 3.3: Power consumption as a function of utilization rate for a single router node from a power evaluation topology.

We used data from the router depicted in Figure 3.3 to investigate additional model parameters. Specifically, we investigated data points 109 and 179: two data points with similar utilization rates but significantly differing power consumption. Recognizing the differing power consumption must be caused by differences in router-internal switching activity, we examined the traces yielding both power responses.

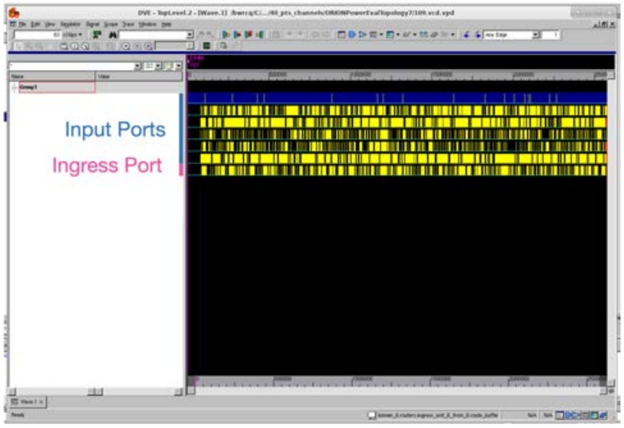
DVE - TopLevel.2 - [Wave.1] /bwrcq/C/.../40_pts_channels/ORIONPowerEvalTopology7/109.vcd.vpd
Group1
Input Ports
Ingress Port

Figure 3.4: Trace for data point 109 (high power consumption)

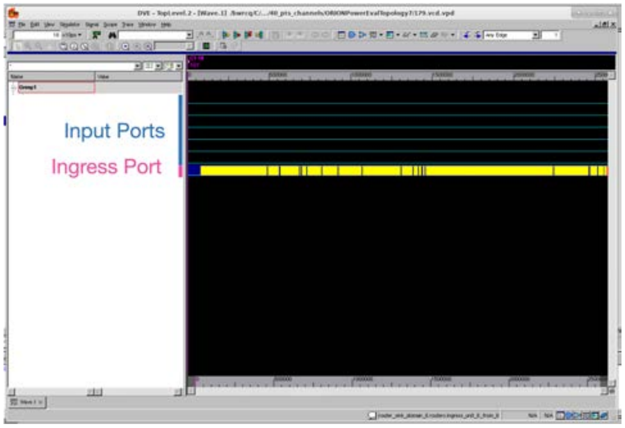
DVE - TopLevel.2 - [Wave.1] /bwrcq/C/.../40_pts_channels/ORIONPowerEvalTopology7/179.vcd.vpd
Group1
Input Ports
Ingress Port

Figure 3.5: Trace for data point 179 (low power consumption)

Figures 3.4 and 3.5 depict very similar utilization rates caused by differing traffic patterns. While the traffic for data point 109 (Figure 3.4) is distributed across all input and ingress ports, the traffic for data point 179 (Figure 3.5) is concentrated on the single ingress port into the router node. As characterized in section 1.1.2, the architecture of ingress ports and input ports varies significantly. The crediting logic for input ports in particular requires additional state and logic to send credits back to upstream router nodes, non-trivially differentiating the power characteristics of input and ingress ports.

We address this by reporting input unit utilization rate and ingress unit utilization rate as separate model parameters, with results reported in table 3.2.

	Ransac Regressor	Gradient Boosting Regressor	SVM with RBF Kernel
MAPE	0.27	0.17	0.20
R^2	0.48	0.67	0.72

Table 3.2: Performance of architectural learning models on realistic NoCs after input port and ingress port utilization rates are separated.

As compared to 3.1, average percent error decreases and R^2 increases for each model, indicating the separation of these parameters are significant.

3.2.2 Flit Hamming Distance

We also examine the relationship between the Hamming distance of adjacent flits and power consumption. Despite routing logic within NoCs being data-independent, data composition may affect the signal transitions of data-bus wires within NoC channels and buffers.

We use the formal trace generation method presented in section 2.2.3 to create traces with consistent Hamming distances between flits; this reduces variance in power response. We tested four Hamming distances, each inspired by specific classes of workloads:

- Hamming distance 0: All flits have the same content. This simulates, for example, an operating system using `memset` to clear a page of memory.
- Hamming distance 2: A sparse matrix in an uncompressed format consists of many entries that are set to 0, resulting in a low Hamming distance on average.
- Hamming distance 32: The expected number of differing bits between two 64-bit random values, such as 64-bit wide flits, is 32.
- Hamming distance 64: The contents of adjacent flits are bitwise negations of each other. This represents an adversarial power scenario in which the switching activity of data lines is maximized.

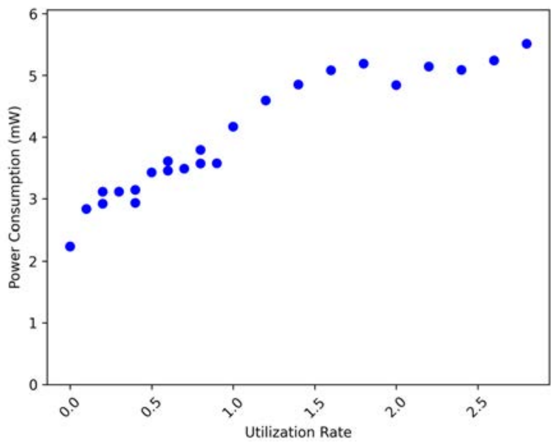

Figure 3.6: Router power consumption when adjacent flits have a Hamming distance of 0.

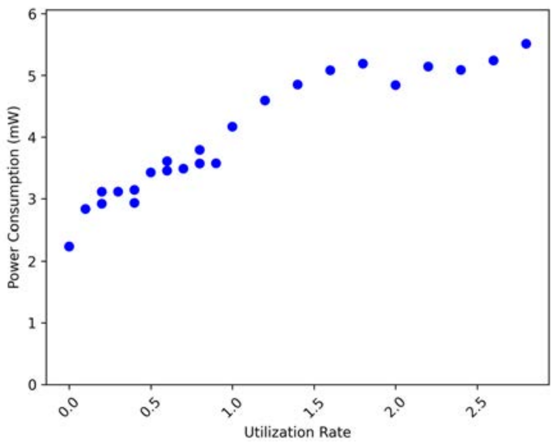

Figure 3.7: Router power consumption when adjacent flits have a Hamming distance of 2.

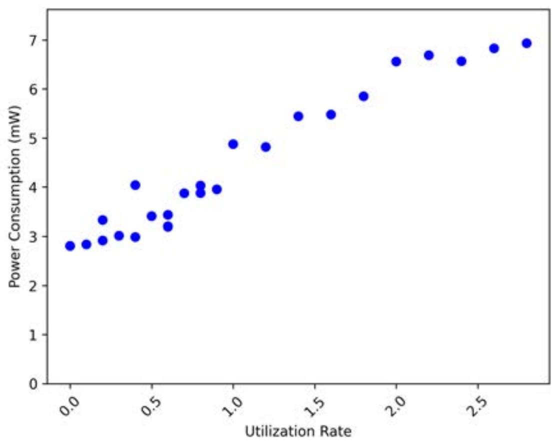

Figure 3.8: Router power consumption when adjacent flits have a Hamming distance of 32.

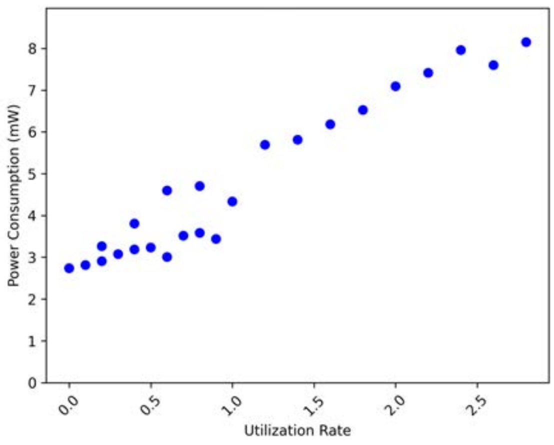

Figure 3.9: Router power consumption when adjacent flits have a Hamming distance of 64.

The data in Figures 3.6 - 3.9 was collected on a NoC router node from the training dataset with 5 input ports, 3 virtual channels, and 3 buffer entries per virtual channel. We see that power consumption at similar utilization rates increases as the Hamming distance between adjacent flits increases. Figures 3.8 and 3.9 also indicate Hamming distance's effects on power consumption is dependent on utilization rate. While power consumption for Hamming distances of 32 and 64 are relatively similar at lower utilization rates, the difference in power consumption becomes more pronounced as utilization rate increases.

Although it is evident Hamming distance has an effect on power consumption, we do not integrate it into our power model as a parameter. Generating model training data that covers both the space of utilization rate and Hamming distance exponentially increases the domain of training data necessary to build an accurate model. Further, since the average Hamming distance between packets in a particular workload can be easily characterized, it is relatively quick and easy to train a model with the specific Hamming distance characteristics found in workloads of interest.

3.3 A Power Model for Constellation NoCs

We construct an architectural learning-based power model for NoC routers using the findings presented above. Our model's architectural parameters are

- $\mathbf{P_1}$**:** the number of input units in the NoC router
- $\mathbf{P_2}$**:** the number of ingress units in the NoC router
- **V:** the number of virtual channels per physical channel
- **B:** the number of buffer entries per virtual channel
- **F:** flit size (in bits)
- $\mathbf{U_1}$**:** Aggregate utilization rate across router input ports
- $\mathbf{U_2}$**:** Aggregate utilization rate across router ingress ports

and the model's output is the router's estimated power consumption. Hamming distance is implicitly encoded; all training and evaluation data is constructed on traffic with the same average Hamming distance. We continue to use a variety of modeling techniques, as no one technique yields the best performance.

3.3.1 Model Evaluation

When trained and evaluated on a train-test split, our model reports an average percent error of 5.1% and an R^2 of 0.97, indicating strong correlation between the model's predictions and the training dataset.

When trained and evaluated on the datasets presented in sections 2.1.1 and 2.1.2, our model reports the statistics depicted in table 3.3.

	Gradient Boosting Regressor	SVM with RBF Kernel
MAPE	0.15	0.17
Max Error	0.79	0.78
R^2	0.72	0.66

Table 3.3: Performance of our architectural learning model on realistic NoCs.

On realistic NoC topologies injected with simulated traffic, our model displays an average percent error similar to models from prior work that were trained on annotated activity factors and evaluated using a train-test split.

Model Shortcomings

Prior work using an SVM regressor with an RBF Kernel presented a maximum prediction error of 20% [11]. In contrast, our model reports a significantly higher max prediction percent error. This could potentially indicate that, as compared to annotated activity factors, simulated traffic results in a greater diversity of power scenarios that cannot be fully characterized by the model parameters we present.

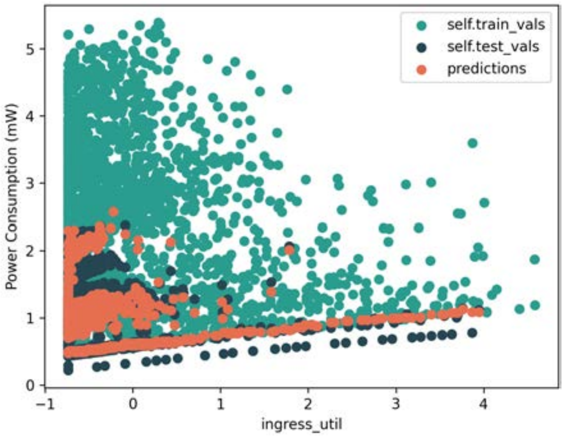

Figure 3.10: Training dataset, evaluation dataset, and model predictions versus ingress utilization rate.

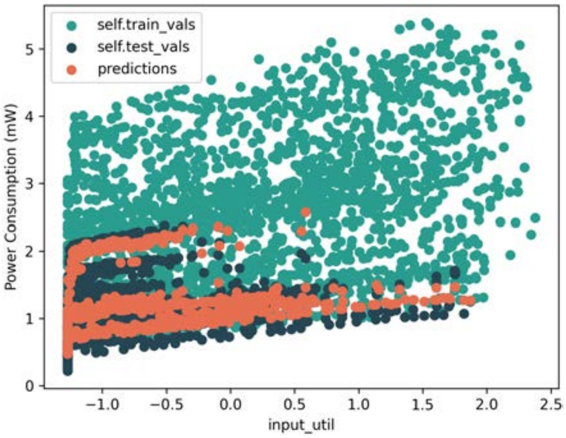

Figure 3.11: Training dataset, evaluation dataset, and model predictions versus input utilization rate.

Figures 3.10 and 3.11 compare model predictions to the training and evaluation datasets. To better reflect model inputs, the horizontal axes depict ingress and input utilization rates scaled to unit variance after the mean is removed; this standardization is expected by many machine learning estimators.

In Figure 3.10, we see a set of evaluation data points outside the range of the training dataset. This indicates that our training dataset of router architectures, derived from prior work, may lack coverage for smaller NoC designs. These smaller NoC designs are relevant because, as depicted in the graph, they can still service relatively large utilization rates since packets need only a few hops to reach their destination. In Figure 3.11, the model significantly mispredicts the power consumption of evaluation datapoints between 1 and 2 mW. When examining the depicted training dataset, we see a decreased density of training dataset points between 1 and 2 mW as compared to the 2 to 2.5 mW range, potentially again indicating a lack of coverage in the training dataset.

As an aside, we caveat this representation with the disclaimer that model predictions, when accurate, obscure the underlying evaluation data point. This format, then, visually emphasizes data-points with higher prediction error. We also acknowledge the evaluation dataset does not cover the entire domain of the training dataset, particularly excluding higher power consumption scenarios. This is intentional: although the model training dataset ought to cover adversarial and edge-case power scenarios, we focus on evaluating our model in more typical use-cases.

3.3.2 Model Usability

Despite the documented shortcomings, our power model for Constellation NoCs achieves performance similar to prior work in architectural ML-based power models for NoCs, except on realistic NoC designs.

	Gradient Boosting Regressor	SVM with RBF Kernel
Training Time (seconds)	0.27	0.76
Prediction Time (seconds)	0.11	0.39

Table 3.4: Model training and prediction time.

Table 3.4 depicts the time, averaged across 10 runs, needed to train and infer on the modeling techniques used for our power model. We train and infer on a 32-core machine with Intel Xeon Gold 6134 CPUs @ 3.2 GHz and 1.1 terabytes of RAM. In comparison, the reference power tool takes approximately 2 days to generate the training dataset on the same machine. The quick runtime of our power model makes it well-suited for high-level design space exploration and iteration in pre-RTL development.

Chapter 4

Heterogeneous NoC Router Power Modeling

Introduced in chapter 1.1.1, heterogeneity in NoCs can be found at both the router and channel level. While the architectural power model for NoC routers presented in chapter 3 can model the power consumption of NoC routers of varying architectural configurations, it assumes that all input and ingress channels into a NoC router have identical micro-architectural parameters. However, as presented in section 1.1.1, NoC routers with irregular channel configurations may be necessary when designing heterogeneous NoCs.

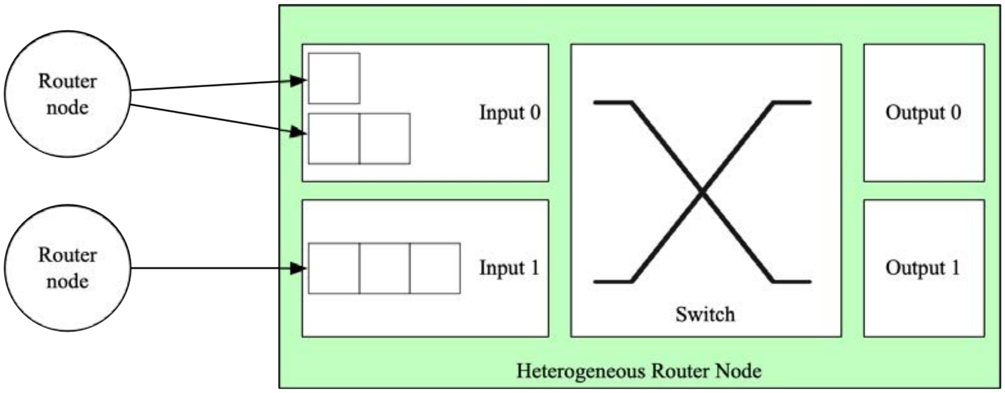

Figure 4.1: An example heterogeneous NoC router. Input 0 has two virtual channels with one and two buffer entries respectively, while Input 1 has one virtual channel with three buffer entries.

Building a learning power model for heterogeneous NoC routers with varying per-channel configuration requires providing the model with per-channel data. Unfortunately, there is no immediately intuitive way to construct a model that predicts the power consumption of a heterogeneous router in aggregate while allowing training data to be channel-index agnostic. For this reason, we decide to model the power consumption of router components individually.

4.1 Router Power Consumption Decomposition

We begin by breaking down the power consumption of NoC routers into individual microarchitectural components to identify router components that contribute significantly to a router's overall power consumption.

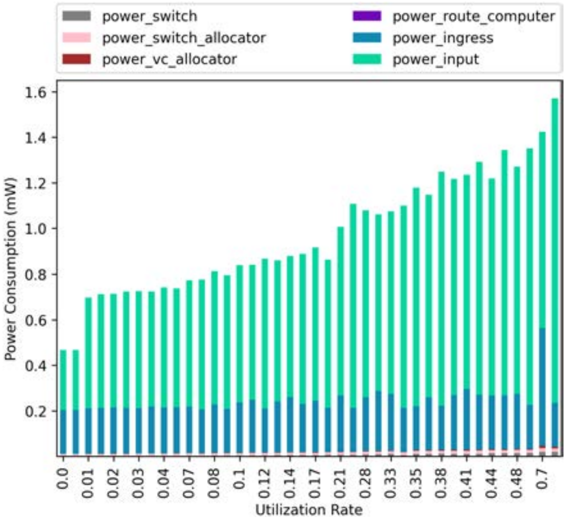

Figure 4.2: Power per-component for a router in a large torus network

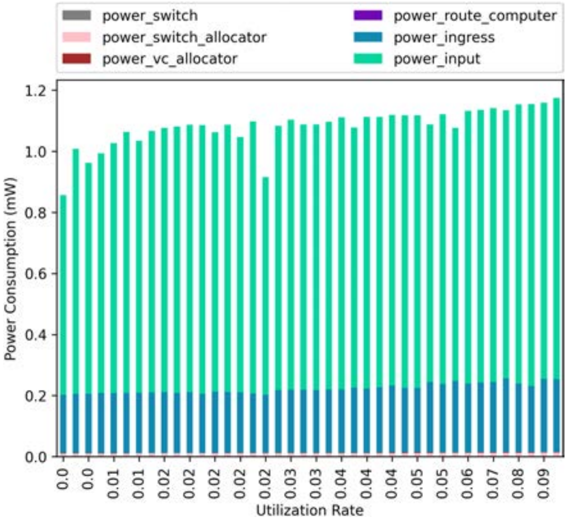

Figure 4.3: Power per-component for a router in a large 2d mesh network

Figures 4.2 through 4.5 depict the power consumption per router component for NoC routers of various configurations and in various topologies as a function of the router's utilization rate. Consistently, we see that power consumption is dominated by input channels and ingress channels, with other router components consuming a negligible amount of power given the amount of error typically found in architectural-level learning power models.

For ease of representation, our graphs revert to utilization rate aggregated across input and ingress channels despite their different power consumption characteristics. This oversimplification explains sudden decreases in power consumption when utilization increases seen on the presented graphs, as these drops represent cases where ingress unit traffic comprises a proportionately greater percentage of overall utilization.

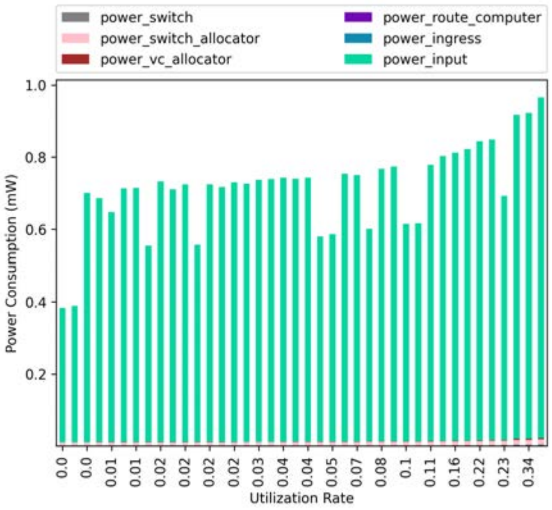

Figure 4.4: Power per-component for a router in a butterfly network

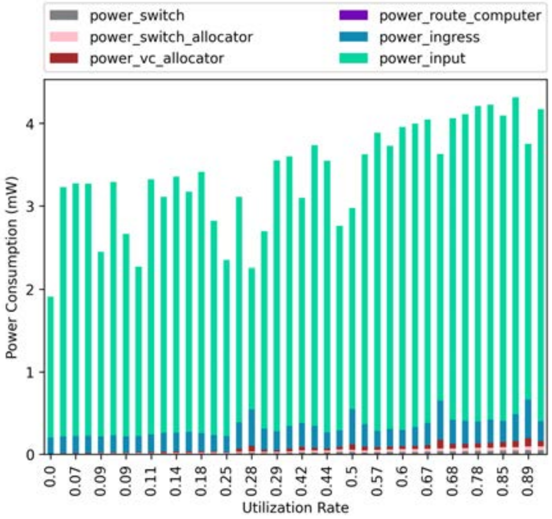

Figure 4.5: Power per-component for a router from the training dataset

Figure 4.5 graphs data collected from a NoC router with 7 input ports, each with 7 virtual channels. As expected, the power consumption for the router switch and virtual channel allocator – both absolutely and relative to other components – are greater than in Figures 4.2 through 4.4. However, power consumption remains dominated by the router's ingress and input units.

Thus, we believe that, to build an architectural-level learning model for Constellation-based heterogeneous NoC routers, it is sufficient to accurately model the power consumption of the router's input and ingress channels.

4.1.1 Input and Ingress Unit Power Characteristics

To evaluate the difficulty of modeling ingress and input unit power consumption, we also analyze the power consumption of ingress and input units with respect to their utilization rates. Here, we specify utilization rate to mean the number of packets entering each individual unit per cycle, on average.

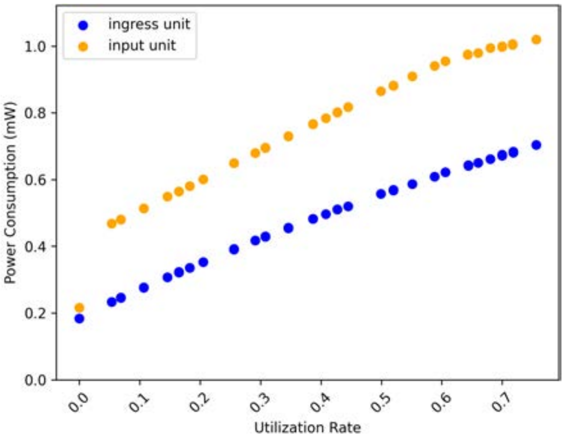

Figure 4.6: Input and ingress unit power consumption in a line topology.

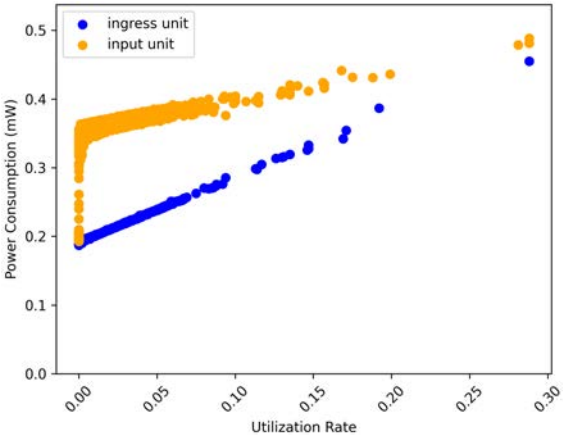

Figure 4.7: Input and ingress unit power consumption in a large mesh topology.

Figure 4.6 depicts ingress and input unit power consumption for all routers in a small line-topology NoC. As expected, power consumption is linear with respect to unit utilization rate. When utilization rate is 0, power consumption for ingress and input units are nearly identical because the credit system found in input units is inactive. However, for small but non-zero utilization rates, such as 0.05, we see a significant discrepancy between ingress and input unit power consumption, illustrating the significant power overhead of the credit system. This difference in power response also quantifies the need to separate ingress and input unit utilization rates in the homogeneous router power model presented in section 3.2.1.

Figure 4.7 depicts ingress and input unit power consumption for all routers in a large mesh topology. Similar to Figure 4.6, input unit and ingress unit power consumption are similar at a utilization rate of 0, but quickly diverge as utilization rate increases. Unlike Figure 4.6, however, Figure 4.7 depicts a range of power consumptions for input units with similar utilization rates. This is likely due to complexities within the credit system, but even when compounded, we believe it is unlikely this fuzziness in power response will result in significant error when predicting power for a NoC router in aggregate.

4.2 Constructing a Heterogeneous Router Power Model

Recognizing input and ingress unit power consumption dominates overall power consumption within a NoC router, our heterogeneous power model estimates NoC router power consumption by aggregating individual estimations of the power consumption of all ingress and input ports on the router. The heterogeneous model uses the same parameters as the homogeneous model characterized in 3.3.

When attempting to model router-internal components other than ingress and input units, we find model estimations are inaccurate and correlation between model predictions and both the training and evaluation datasets are low. This indicates that our model parameters sufficiently capture power-relevant architectural features for input and ingress ports but are insufficient for modeling other NoC router components. However, since ingress and input port power consumption dominate overall NoC router power consumption, we can still accurately model router power consumption.

4.3 Component-specific Model Performance

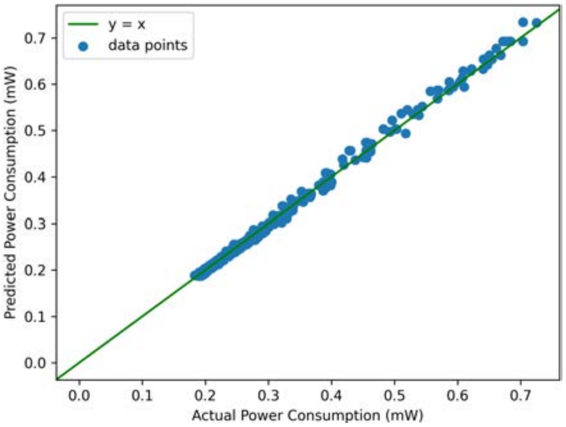

Figure 4.8: Actual vs predicted power consumption for NoC router ingress units.

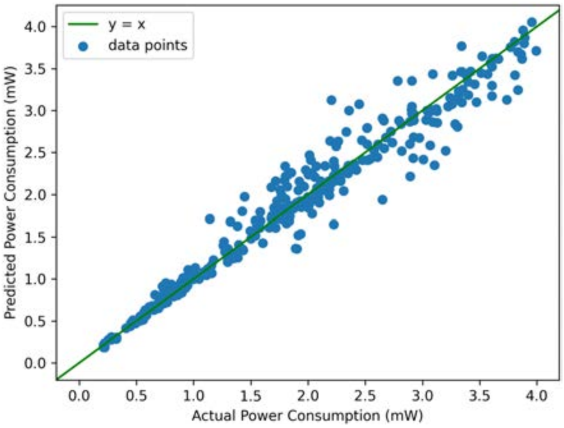

Figure 4.9: Actual vs predicted power consumption for NoC router input units.

When comparing model predictions against reference power consumption for NoC router ingress and input units, we see that input unit predictions are less accurate as power consumption increases – this is expected. The micro-architectural parameters considered by the model do not clearly capture the behavior of the credit system used by input units, and higher power consumption scenarios correlate with greater activation of crediting logic. Further, since individual crediting logic is synthesized for each channel between NoC routers, model parameters that capture the behavior of crediting logic would be micro-architectural specific and require knowledge of, for example, the routing algorithm used by the NoC. Since the architectural model is intended to be high-level, we choose not to include these parameters.

	Input Channel Predictor	Ingress Channel Predictor
MAPE	0.08	0.01
Max Error	0.34	0.03
R^2	0.82	0.99

Table 4.1: Performance of our architectural learning model built on a gradient-boosted regressor on NoC input and ingress units.

In table 4.1, we see that ingress unit power consumption can be predicted more accurately than input unit power consumption, again likely because of the crediting logic found in input units. Since input and ingress unit power consumption dominate NoC router power consumption in all traffic scenarios, accurate modeling of input and ingress unit power consumption likely leads to accurate NoC router power prediction. Further, the decomposition of utilization rate as a per-channel metric, as opposed to an aggregate NoC router metric, leads to more accurate power predictions.

Chapter 5

Conclusion

In this thesis, we present our progress towards building an architectural-level workload-aware power model for networks on chip. We specifically target pre-RTL exploration and build a model that offers an estimation of NoC power consumption solely using the NoC's architectural features and coarse metrics for the workloads expected to run on the SoC. We evaluate our model on realistic NoCs of varying topologies and architectures to validate model performance. Additionally, we present progress towards a NoC power model for heterogeneous router architectures. We document the power breakdown of NoCs built on Constellation and leverage our findings to build component-specific power models that can be used to predict the power consumption of a channel-level heterogeneous NoC router.

Compared to prior work, we generate training datapoints for our ML-based power model by simulating traffic on the training dataset of NoC architectures. Instead of annotating switching activity within the power estimation tool, power training data is gathered by replaying traces of traffic on NoCs synthesized using the Intel 16 manufacturing process. Further, while previous architectural power models offer static predictions for an architecture's power consumption, our model is capable of offering workload-specific power estimations. For application-specific NoCs, this makes the model's predictions more relevant to the design scenario of interest. If the NoC is not designed for a specific application, power predictions with varying dynamic factors can be averaged to recreate a workload-agnostic prediction. Finally, we generalize our power model to support per-channel heterogeneity. While prior work assumes NoC router channels have uniform architectures, our heterogeneous power model is capable of providing power estimations for routers with irregular channel constructions, such as those that bridge NoC sub-networks of differing performance levels.

While our heterogeneous model appears to accurately model NoC component power consumption, its applicability for design space exploration appears reduced. Although per-channel utilization rates can be extracted by running workloads on Constellation-generate RTL, these metrics are difficult to calculate pre-RTL. As manufacturing processes improve, data movement has increasingly begun to dominate power consumption. The challenges of characterizing this movement, in addition to identifying other data-relevant power modeling parameters, indicates there remains an abundance of research opportunities for NoC power modeling.

Bibliography

[1] Alon Amid et al. “Chipyard: Integrated Design, Simulation, and Implementation Framework for Custom SoCs”. In: *IEEE Micro* 40.4 (2020), pp. 10–21. ISSN: 1937-4143. DOI: 10.1109/MM.2020.2996616.

[2] Jonathan Bachrach et al. “Chisel: Constructing hardware in a Scala embedded language”. In: *DAC Design Automation Conference 2012*. 2012, pp. 1212–1221. DOI: 10.1145/2228360.2228584.

[3] Nicholas P Carter et al. “Runnemede: An architecture for Ubiquitous High-Performance Computing”. In: *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*. Feb. 2013, pp. 198–209.

[4] William James Dally and Brian Patrick Towles. *Principles and Practices of Interconnection Networks*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2004. ISBN: 9780080497808.

[5] Saurabh Dighe et al. “Within-Die Variation-Aware Dynamic-Voltage-Frequency-Scaling With Optimal Core Allocation and Thread Hopping for the 80-Core TeraFLOPS Processor”. In: *IEEE J. Solid-State Circuits* 46.1 (Jan. 2011), pp. 184–193.

[6] *Genus Synthesis Solutions*. https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/synthesis/genus-synthesis-solution.html. Accessed: 2023-04-25.

[7] Boris Grot et al. “Kilo-NOC: A Heterogeneous Network-on-Chip Architecture for Scalability and Service Guarantees”. In: *Proceedings of the 38th Annual International Symposium on Computer Architecture*. ISCA ’11. San Jose, California, USA: Association for Computing Machinery, 2011, pp. 401–412. ISBN: 9781450304726. DOI: 10.1145/2000064.2000112. URL: https://doi.org/10.1145/2000064.2000112.

[8] Yong Hu et al. “Machine Learning Approaches for Efficient Design Space Exploration of Application-Specific NoCs”. In: *ACM Trans. Des. Automat. Electron. Syst.* 25.5 (Aug. 2020), pp. 1–27.

[9] *Jasper RTL Apps*. https://www.cadence.com/en_US/home/tools/system-design-and-verification/formal-and-static-verification/jasper-gold-verification-platform.html. Accessed: 2023-05-01.

[10] *Joules RTL Power Solution.* https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/power-analysis/joules-rtl-power-solution.html. Accessed: 2023-04-23.

[11] Andrew B Kahng, Bill Lin, and Siddhartha Nath. “ORION3.0: A Comprehensive NoC Router Estimation Tool”. In: *IEEE Embedded Sys. Lett.* 7.2 (June 2015), pp. 41–45.

[12] Andrew B Kahng et al. “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.* Apr. 2009, pp. 423–428.

[13] Donggyu Kim et al. “Simmani: Runtime Power Modeling for Arbitrary RTL with Automatic Signal Selection”. In: *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture.* MICRO ’52. Columbus, OH, USA: Association for Computing Machinery, Oct. 2019, pp. 1050–1062.

[14] Seung Eun Lee and Nader Bagherzadeh. “A high level power model for Network-on-Chip (NoC) router”. In: *Computers & Electrical Engineering* 35.6 (2009). High Performance Computing Architectures, pp. 837–845. ISSN: 0045-7906. DOI: https://doi.org/10.1016/j.compeleceng.2008.11.023. URL: https://www.sciencedirect.com/science/article/pii/S0045790608001365.

[15] *New RTL Synthesis Tool Saves Hours of Your Time.* https://youtu.be/SvF-aMKRec8. Accessed: 2023-04-25.

[16] Anh T Tran and Bevan Baas. “NoCTweak: a highly parameterizable simulator for early exploration of performance and energy of networks on-chip”. In: *VLSI Computation Lab, ECE Department, University of California, Davis, Tech. Rep. ECE-VCL-2012-2* (2012).

[17] B V P Vasantha Kumar et al. “A technique to eliminate glitch power consumption at physical design stage in CMOS circuits”. In: *2011 World Congress on Information and Communication Technologies.* Dec. 2011, pp. 639–644.

[18] Zhiyao Xie et al. “APOLLO: An Automated Power Modeling Framework for Runtime Power Introspection in High-Volume Commercial Microprocessors”. In: *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture.* MICRO ’21. Virtual Event, Greece: Association for Computing Machinery, Oct. 2021, pp. 1–14.

[19] Yanqing Zhang, Haoxing Ren, and Brucek Khailany. “GRANNITE: Graph Neural Network Inference for Transferable Power Estimation”. In: *2020 57th ACM/IEEE Design Automation Conference (DAC).* July 2020, pp. 1–6.

[20] Jerry Zhao et al. “Constellation: An Open-Source SoC-Capable NoC Generator”. In: *2022 15th IEEE/ACM International Workshop on Network on Chip Architectures (NoCArc).* Oct. 2022, pp. 1–7.