A Generative AI for Heterogeneous Network-on-Chip Design Space Pruning

Maxime Mirka, Maxime France-Pillois, Gilles Sassatelli, Abdoulaye Gamatié
LIRMM, Université de Montpellier & CNRS
Montpellier, France
name.surname@lirmm.fr

***Abstract*—Often suffering from under-optimization, Networks-on-Chip (NoCs) heavily impact the efficiency of domain-specific Systems-on-Chip. To cope with this issue, heterogeneous NoCs are promising alternatives. Nevertheless, the design of optimized NoCs satisfying multiple performance objectives is extremely challenging and requires significant expertise. Prior works failed to combine many objectives or required an extended design space exploration time. In this paper, we propose an approach based on generative artificial intelligence to help pruning complex design spaces for heterogeneous NoCs, according to configurable performance objectives. This is made possible by the ability of Generative Adversarial Networks to learn and generate relevant design candidates for the target NoCs. The speed and flexibility of our solution enable a fast generation of optimized NoCs that fit users' expectations. Through some experiments, we show how to obtain competitive NoC designs reducing the power consumption with no communication performance or area penalty compared to a given conventional NoC design.**

***Index Terms*—Generative Adversarial Network, CAD, Network-on-Chip, DSSoC, Heterogeneous, Machine learning**

I. Introduction

System specialization offers a promising solution to design power, area and performance-efficient Systems-on-Chip (SoCs). When designed to meet the final application requirements, heterogenous SoCs usually produce competitive designs. However, new challenges arise with the design of such tailored domain-specific SoCs. The key issue is to find a design that optimally satisfies the application requirements. While High-Level Synthesis (HLS) tools allow non-expert users to generate hardware designs for their applications, these tools do not provide an optimal SoC architecture. A naive solution to identify the optimal architecture should be to evaluate each possible design and pick the best one. However, the set of candidate designs (design space) is often very broad, and the evaluation time would be excessively long. Hence, an exhaustive exploration of the design space is usually not tractable because of prohibitive exploration time. A methodology to reduce the design space is therefore desirable.

SoC interconnect is a key component that boldly influences performance and power consumption. An undersized interconnect fabric may lead to a communication bottleneck with a potentially drastic impact on global system performance, especially in Von Neumann-like architectures. On the other side, an oversized interconnect uselessly consumes many power and silicon area [1]. The parallelism and flexibility offered by Networks-on-Chip (NoCs) have made it the *de-facto* standard for multicore SoC. Although heterogeneous NoCs ensure the best trade-off between performance and power (see Section II), its architecture defining is challenging.

In this paper, we propose an AI-based method to prune the design space of heterogeneous NoCs. Benefiting from the recent progress in Generative Adversarial Networks (GANs), we devise a tool able to generate a reduced set of optimized NoC configurations. These generated NoCs are optimized according to multiple objectives defined by the user to satisfy the expectations. The addition of several Reward modules to the "classical" GAN architecture enables multi-objective optimization and results in the generation of a subset of NoC configurations close to the optimal Pareto frontier. Hence, for a given traffic pattern, our tool generates a subset of optimized designs. The size of this subset is specified by a user. The generated heterogeneous NoC design can improve the power consumption while preserving the throughput, compared to a similar size homogeneous NoC design. In our experiments, we observe up to 15% power saving.

II. Problem definition

NoC sizing is a tedious and complex process. Under-sizing an NoC directly deteriorates latency and throughput, while over-sizing substantially increases power consumption. NoC power consumption account for a significant fraction of the chip's power budget (up to 28%) [2]. When studying realistic NoC traffic workloads, non-uniform patterns are observed [3]. This results in greater pressure put on specific areas of the NoC. Consequently, heterogeneous NoCs offer a promising solution to handle these workload imbalances.

However, the design space size of a heterogeneous NoC is prohibitively large, even when merely considering the heterogeneity of routers, the number of NoC configurations being $Number\_Routers\_Types^{Number\_Routers}$. For a 64-router NoC with three classes of routers, we obtain 3^{64} NoC design options. Although SoC experts may have use intuitions to smartly reduce the design space, this approach remains non-practical and does not ensure optimal performance. As a result, a fast systematic method is required to prune the design space and generate optimal NoC designs. The present work addresses this issue by proposing a generative AI framework.

III. Related work

The exploration of the best NoC designs is related to multi-objective optimization. In this field, previous works proposed

978-3-9819263-6-1/DATE22/©2022 EDAA

active algorithms to approximate a set of Pareto-optimal designs from a larger design space [4], [5]. These algorithms combine the simulation of some design samples with heuristics to establish the set of probable Pareto-optimal designs. While they provide a relatively accurate set, they need to simulate a non-negligible number of design samples (up to 15% of the design space for NoC in [5]). Hence, these strategies require a consequent delay to provide a new optimum NoC design.

Regarding proposals targeting exclusively the design of NoC design, they struggle to handle flexible multi-objective optimization. The recent work of Alhubail *et al.* aims to aid in the design of on-chip interconnects for heterogeneous SoCs [6]. It combines two algorithms to handle two objectives, i.e. a Genetic Algorithm to reduce NoC latency, and a Strength Pareto Evolutionary Algorithm to target the power consumption. Although this contribution provides a tool to design heterogeneous NoCs, the adopted backend prevents using multiple concurrent objectives. In our work, we propose a methodology tackling multiple optimization objectives at once.

In [7], the *GANNoC* framework exploits Convolutional Neural Network (CNN) and GANs to generate irregular NoC topologies minimizing the number of inter-router connections. In this work, we leverage GANs to reduce the design space exploration for heterogeneous NoC composed of a diversity of routers. Our framework differs from *GANNoC* in three major points: 1) it proposes a novel GAN framework to handle multi-objective optimization, 2) it deals with router heterogeneity, and 3) it exploits graph neural networks [8] to model NoCs more accurately.

IV. Design Space pruning tool

The proposed tool benefits from a framework made of a GAN and Reward modules. The GAN allows the automatic generation of valid NoC configurations, while the Rewards help steering the generation process towards optimized solutions.

A. GAN and GNN neural networks

a) GAN: A Generative Adversarial Network (GAN) is a neural network architecture proposed by Goodfellow *et al.* [9].

A GAN consists of two neural networks: a Generator model and a Discriminator model. The Discriminator is a neural network trained to classify its inputs into fake or real data. It is alternatively fed by real data coming from a dataset and fake data originating from the Generator module. Thus, its job consists in labeling real data as real and generated data as fake. As for the Generator, its goal is to mislead the Discriminator by generating data as real as possible. Both neural networks are trained concurrently in an alternating fashion. Therefore, GANs are based on a game theory scenario in which the Generator tries and learns to fool the Discriminator. It eventually leads the Generator toward the production of original realistic data.

GANs convergence is a well-known issue. They are prone to vanishing gradient and mode collapse failures. Thus, Arjovsky introduced the Wasserstein loss (W-loss) and built upon this concept a WGAN mitigating those training problems [10].

Lately, the concept of "Reward WGAN" (RWGAN) was introduced [7], [11]. It adds a third network, called Reward, to provide further guidance to the Generator's learning. Hence, this new architecture specializes the Generator's learning to not only generate data within the "real domain", but within a chosen subset of the "real domain."

b) Graph Neural Networks: Due to the significant similarity between NoCs and graph models, we consider the NoC generation as a graph generation problem. Both are constituted from a set of links (i.e. connections/edges) and a set of nodes (i.e. routers/vertices). Recent advances in Graph Neural Networks (GNN) [8] therefore open attractive opportunities for NoC modeling and generation. In this work, we leverage graph convolutional networks (GCNs) [12] to devise Neural Network architectures providing accurate graph predictions.

B. Multi-Reward WGAN

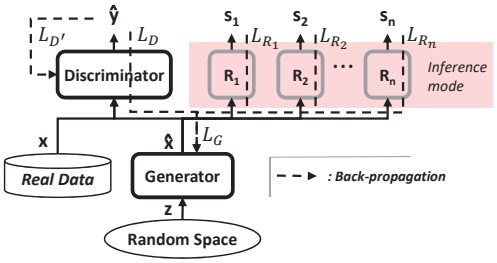


Fig. 1: Multi-Objectives RWGAN diagram, with the back-propagation path in dashed arrows.

Our proposed Multi-Reward WGAN relies on two fundamental considerations which are as follows:

a) NoC generation: The Multi-Reward WGAN (M-RWGAN) concept extends the RWGAN architecture by implementing several Reward modules, leading to a multi-objective Generator training (see Figure 1). In the proposed framework, the architecture of the Discriminator and Rewards are based on GNN (i.e. GCN [12]). The Discriminator training is similar to traditional WGAN. It takes both x and $\hat{x}$, respectively real data and generated data, and output a score regarding the "realness" of the input, i.e. $\hat{y}$. It is then trained to minimize its prediction error i.e. minimize its loss function $L_{D'}(x, \hat{x})$ implemented as a W-loss. The Generator outputs $\hat{x}$ from a random noise input z. $\hat{x}$ is processed by the Discriminator and the Rewards (R_i, with $i \in [1, n]$). As illustrated in Figure 1, the Generator parameters are trained according to the outputs of these modules, i.e. $\hat{y}$ from the Discriminator, and s_i from the Rewards R_i. Following the WGAN training, the Generator learns to minimize the Wasserstein loss coming from the Discriminator, while minimizing as well the losses from the Rewards. Those latter losses are obtained via a mean-squared error measure, w.r.t. to the corresponding score goal. Rewards are supervised models trained before the Generator and Discriminator training, with labeled datasets created by using a NoC simulator. Rewards modules are only used in inferring mode within the M-RWGAN framework.

b) Multi-Objective Loss Function: The Generator's training consists in minimizing its loss. From the RWGAN Gener-

ator loss function presented in a previous work [7], we derive the M-RWGAN Generator loss principle as follows:

$$L_G(z) = (1-\lambda)L_D(\hat{x}) + \lambda[\sum_{i=1}^{n} \beta_i L_{R_i}(\hat{x})] \quad (1)$$

, where L_G, L_D and L_{R_i} are respectively the loss function of the Generator, Discriminator and i^{th} Reward. λ represents the training ratio between the Discriminator and Reward feedback. $\sum_{i=1}^{n} \beta_i = \beta$, where β is a weight coefficient to balance the order discrepancies that may appear between Discriminator loss values and Rewards loss values. *Note:* β must be adjusted w.r.t to the number of Rewards n to avoid losses deterioration.

As a consequence, our proposed architecture provides a solution to train a generative neural network upon multi-objectives. It further enables to tune the consideration by the generator of each goal with weight coefficients (i.e. β_i), leading to finely customized objective functions.

V. Evaluation

A. Experimental setup

a) Use case definition: In this paper, we limit the router heterogeneity to three router classes {Big, Medium, Small}. There are three-stage pipeline router with one virtual channel. Their input buffers size are respectively {12, 4, 2} flits. We consider a 2D-mesh NoC composed of 64 routers organized in a 8x8 grid. We use a classical XY routing algorithm. We devise the proposed framework to optimize the generated NoCs according to three objectives: increasing NoC throughput, reducing power consumption, and decreasing silicon area. Each objective is assessed by a dedicated Reward. We evaluate NoC performances for a hotspot traffic pattern. The "hot" router was arbitrarily set at position $[6, 6]$. 30% of the messages are sent to the hotspot, other messages obey a uniform pattern.

b) NoC Simulator: We performed NoCs performances with a fast high-level discrete-event simulator for communication networks named *Omnet++* [13]. The HNOCS framework [14] enhances *Omnet++* with additional NoC specific-support. The power and area assessments are achieved with the *Orion3* library [15], an accurate high-level estimations tool. The used technological parameters comprise a 45nm manufacturing, a Vdd of 1.0V, and a 650MHz frequency.

c) M-RWGAN neural networks models: We distinguish the architecture of the Generator and both the Discriminator and the Rewards. The Generator is a basic multi-layer perceptron. It is composed of three dense layers (i.e. fully connected) of size {768, 1536, 192}, with leakyReLU as activation function. Its input is a random vector of size 100. Its output is reshaped to a 64x3 matrix, matching the target NoC size. Since the router's type is encoded as a one-hot vector, the three channels stand for the number of router classes. A *GumbelSoftmax* function allows enforcing this one-hot encoding [11].

The Discriminator and the Rewards implement the same architecture. They are composed of four GCN layers of size {128, 64, 64, 32}, with a dense output layer of size 1. Activation functions are LeakyReLU. An exception is made for the Reward devoted to the NoC area estimation. As we estimate the NoC area solely based on the routers, the NoC topology has no impact. Thus, the use of GCN is irrelevant, and we approximate the area with a neural network composed of two CNN layers of size {128, 32} with 3x3 filters, and 2 dense layers of size {128, 1}. Models are implemented in Python, using Keras [16] and Spektral [17] API.

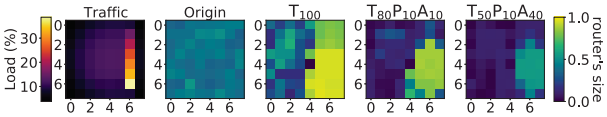
B. Training methodology

The proposed tool requires "offline" training: (1) the supervised training of the Rewards, (2) the GAN training.

Performance metrics of some NoC designs sampled out of the design space are obtained in simulation, as described above. In this work, we empirically found that the simulation of 10 000 samples is sufficient to achieve well-trained Rewards. Indeed, the dataset is carefully spread over the design space. Hence, despite accounting for an infinitesimal part of the design space (3^{64}), these samples are enough representative of the NoC design behaviors to train a Neural Network. While the creation of a dataset and the training of the Rewards is time-consuming, i.e. approx. 3 days in our case, this step must be done only once for each considered traffic pattern.

The GAN is trained for specific configurations of the user objectives. This tool configuration is denoted $\mathrm{T}_x\mathrm{P}_y\mathrm{A}_z$, where x, y, z respectively stand for the throughput, power and area Rewards ratio. From the user point of view, these ratios represent the optimization effort performed by the tool to respectively: maximize the throughput, reduce the power consumption, and decrease the NoC area. From a theoretical perspective, these ratios define the share assigned to each objective during the Generator's loss computation (i.e. $\frac{\beta_i}{\beta}$, see Eq. 1).

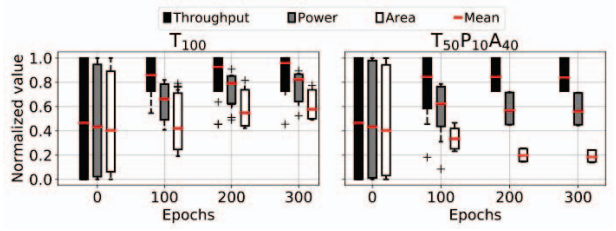
C. Proof of concept



(a) Hotspot traffic.

Fig. 2: Impact of Rewards trade-offs on the generated NoCs.

Figure 2 illustrates the learning capability of the proposed framework. We train our GAN according to several objectives trade-offs: T_{100} (i.e. $\mathrm{T}_{100}\mathrm{P}_0\mathrm{A}_0$), $\mathrm{T}_{80}\mathrm{P}_{10}\mathrm{A}_{10}$ and $\mathrm{T}_{50}\mathrm{P}_{10}\mathrm{A}_{40}$. After the training, we configure our tool to generate 100 NoC designs at once. The per-router NoC traffic intensity is depicted on the left. The other plots show the average router's size over the 100 generated NoC designs. The "origin" plot represents the produced designs before training. We first notice the tool tends to increase the size of routers in correlation with the traffic pattern when throughput is set as a sole objective. Afterward, we observe a reduction of the router's size for routers in the periphery of the "hot" routers, which results of Power and Area optimization. This shows the M-RWGAN indeed properly identified these locations are good candidates for Small routers given the relatively low traffic flowing in these areas and thereby the limited resulting performance degradation.



(a) Hotspot traffic.

Fig. 3: Impact of different rewards trade-off on generated sets. Boxes comprise 95% of data.

Figure 3 illustrates the convergence of the generator training toward subsets of optimized NoCs. It displays the measured NoC performances along the training i.e. for various epochs. We perform this analysis for two different tool configurations.

We notice that at the beginning of the training process the NoCs performances are rather widespread. After a few training epochs, the generated NoCs converge toward values matching the user objectives. The T_{100} configuration, optimized for high throughput, generates NoCs designs gathered around the optimal throughput at the end of the training. The second tool configuration, named $T_{50}P_{10}A_{40}$, achieves an interesting trade-off between the three user objectives. The generated NoCs offer a high throughput with a reduced area, while the power consumption remains at a medium level.

D. Generated NoCs evaluation

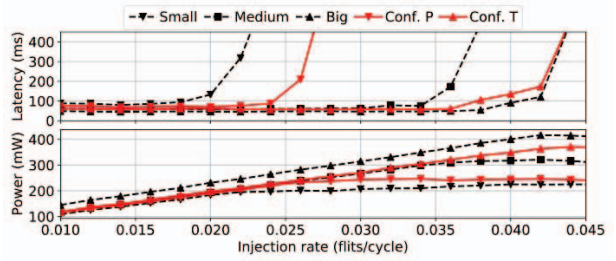


Fig. 4: Examples of generated NoCs throughput and power performance for hotspot traffic pattern.

Figure 4 illustrates the ability of our method to obtain competitive NoC configurations according to the user objectives. We compare *Conf. P* and *Conf. T*, two configurations produced by a Generator, respectively trained with the Rewards trade-offs $T_{10}P_{90}$ and $T_{50}P_{10}A_{40}$. Considering the hotspot traffic pattern, the first configuration favors power reduction whereas the second targets throughput optimization. The resulting NoC configurations exhibit competitive performances over naive homogeneous configurations. For instance, the *Conf. P* design reduces the power consumption and the silicon area by respectively 24% and 25%, while incurring 22% of throughput loss compared to the *Medium* NoC. Considering the *Conf. T* NoC, it provides the best possible throughput (equal to the *Big* NoC), with 15.2% power savings and 65.9% area reduction.

VI. Conclusion

The design of Heterogeneous NoCs in the scope of multi-objective optimization is particularly challenging.

In this paper, we propose a tool based on generative AI to address this issue. The proposed tool prunes a design space according to multiple objectives and generates a subset of near performance-optimal candidates designs. The tool is driven at its core by M-RWGAN, a specific GAN-based network architecture which is a contribution of this paper. This M-RWGAN makes it possible to steer the generative process towards solutions having high fitness with respect to the chosen optimization criteria and their respective significance. A significant strength of this proposal compared to the literature, therefore, lies in its ability to perform fine-grain parametric design space exploration under multiple optimization criteria.

The performance evaluation of the generated NoCs is shown to be competitive compared to naive NoC configurations. Future works aim at evaluating this framework in the case of multi-application traffic patterns. The goal being to produce heterogeneous NoC configurations satisfying multiple application requirements while optimizing non-functional parameters such as area and power consumption.

References

[1] B. K. Daya *et al.*, "Scorpio: A 36-core research chip demonstrating snoopy coherence on a scalable mesh noc with in-network ordering," in *ACM/IEEE 41st Int. Symp. on Comput. Architecture*, 2014, pp. 25–36.
[2] Y. Hoskote *et al.*, "A 5-ghz mesh interconnect for a teraflops processor," *IEEE Micro*, vol. 27, no. 5, pp. 51–61, 2007.
[3] P. V. Gratz and S. W. Keckler, "Realistic workload characterization and analysis for networks-on-chip design," in *The 4th Workshop on Chip Multiprocessor Memory Systems and Interconnects (CMP-MSI)*, 2010.
[4] G. Palermo, C. Silvano, and V. Zaccaria, "Respir: A response surface-based pareto iterative refinement for application-specific design space exploration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 12, pp. 1816–1829, 2009.
[5] M. Zuluaga, G. Sergent, A. Krause, and M. Püschel, "Active learning for multi-objective optimization," in *Proc. of the 30th Int. Conf. on Mach. Learn.*, vol. 28, no. 1. PMLR, Jun 2013, pp. 462–470.
[6] L. Alhubail *et al.*, "Noc design methodologies for heterogeneous architecture," in *28th Euromicro International Conference on Parallel, Distributed and Network-Based Process. (PDP)*, 2020, pp. 299–306.
[7] M. Mirka *et al.*, "Gannoc: A framework for automatic generation of noc topologies using generative adversarial networks," in *Proc. of the 2021 Rapid Simulation and Perf. Eval.: Methods and Tools Proc.* New York, NY, USA: Association for Computing Machinery, 2021, p. 51–58.
[8] Z. Wu *et al.*, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, p. 4–24, Jan 2021.
[9] I. J. Goodfellow *et al.*, "Generative adversarial networks," in *Advances in Neural Information Processing Systems 27 (NIPS)*, 2014.
[10] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," 2017, *arXiv preprint arXiv:1701.07875*.
[11] N. D. Cao and T. Kipf, "Molgan: An implicit generative model for small molecular graphs," *CoRR*, vol. abs/1805.11973, 2018.
[12] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2017, *arXiv preprint arXiv:1609.02907*.
[13] A. Varga, "Using the omnet++ discrete event simulation system in education," *IEEE Trans. on Educ.*, vol. 42, no. 4, pp. 11 pp.–, 1999.
[14] Y. Ben-Itzhak, E. Zahavi, I. Cidon, and A. Kolodny, "Hnocs: Modular open-source simulator for heterogeneous nocs," in *International Conference on Embedded Computer Systems (SAMOS)*, 2012, pp. 51–57.
[15] A. B. Kahng *et al.*, "Orion3.0: A comprehensive noc router estimation tool," *IEEE Embedded Systems Letters*, vol. 7, no. 2, pp. 41–45, 2015.
[16] F. Chollet *et al.*, "Keras," https://keras.io, 2015.
[17] D. Grattarola and C. Alippi, "Graph neural networks in tensorflow and keras with spektral," 2020, *arXiv preprint arXiv:2006.12138*.