Uma ferramenta baseada em inteligência artificial para exploração de espaço de projeto para redes em chip

An artificial intelligence-based tool for exploiting design space for chip networks

DOI:10.34117/bjdv7n1-134

Recebimento dos originais: 05/12/2020
Aceitação para publicação: 08/01/2021

Jefferson Igor Duarte Silva
Mestrado
Instituição: IFRN
Endereço: RN 118, S/N, Povoado Base Física, Ipanguaçu/RN
E-mail: Jefferson.igorbr@gmail.com

Márcio Eduardo Kreutz
Doutorado
Instituição: UFRN
Endereço: Campus Universitário Lagoa Nova, Natal/RN
E-mail: kreutz@dimap.ufrn.br

Monica Magalhães Pereira
Doutorado
Instituição: UFRN
Endereço: Campus Universitário Lagoa Nova, Natal/RN
E-mail: monicapereira@dimap.ufrn.br

RESUMO
Com o incremento em números de núcleos em sistemas em chip, arquiteturas de barramento tem sofrido com algumas limitações. Os requisitos das aplicações demandam mais largura de banda e baixas latências. Em face a esse cenário, redes em chip emergiram como uma opção para superar essas limitações. Redes em chip são compostas por um conjunto de roteadores e enlaces de comunicação. Nesse trabalho, nós propomos o uso de técnicas de inteligência artificial para otimizar a arquitetura das redes em chip. A ferramenta explora o espaço de projeto em termos de predição de área, latência e potência para diferentes configurações. Os resultados têm demonstrado a validade dessa proposta e a adequação as restrições impostas pelo projetista.

Palavras-chave: Redes em chip, inteligência artificial, sistemas em chip.

ABSTRACT
With the increasing number of cores in Systems on Chip (SoCs), bus architectures have suffered some limitations. The application requirements demand more bandwidth and lower latencies. Facing this scenario, Networks-on-Chip (NoCs) emerged as a way to overcome those limitations. NoCs are composed of a set of routers and communication links. In this work, we propose the usage of Artificial Intelligence techniques to optimize NoC architectures. The tool explores the design space in terms of area, latency, and power

prediction for different NoC configurations. The results have proven to be valid and adequate to the constraints of the target applications.

Keywords: Networks on chip, artificial intelligence, systems on chip.

1 INTRODUCTION

Networks on Chip are a natural choice for communication architectures on MPSoC projects due to issues such as scalability and parallelism. This paradigm uses a set of routers and links to become able to get the performance level required by applications. However, routers have many parameters that can impact on performance, area, or power consumption. Therefore, choose the correct values for each parameter is a vital designer task, but these choices demand time for simulation every NoC configuration. In this point, the designer needs to choose: high accuracy or lower simulation time. If he selects high accuracy, every simulation will require a long time to simulate every clock cycle. However, high abstraction level demands lesser simulation time, using languages such as SystemC, but it will lose accuracy.

Independent of accuracy or simulation time, every NoC configuration needs to be analyzed to comply with design constraints. In order to find a suitable setting, designers can perform thousands of simulations leading to long times and huge computational efforts. In this way, simulate every option is not adequate because it is a combinatorial problem of the NP-Hard problem class. Starting from this scenario, one have two options for improving the design: to speed up the simulation time without losing accuracy or speed up the process to find adequate NoCs configurations. In this paper, we approach both, using Machine Learning techniques to predict the latency value and Genetic Algorithm to explore the design space.

Many studies applied these techniques to improve NoC characteristics for irregular topologies. However, for regular topologies there are few papers targeting the same goal. While irregular topologies have as the main advantage, best options for area reduction, they do not fit well when the same architecture may be reused for more than one project. This paper focus on 2D mesh NoC. Next Section approaches the paper contributions and its proposal.

1.1 PAPER CONTRIBUTIONS

Using Machine Learning methods to avoid simulations improves our required time to evaluate a specific NoC. Instead of using NoC simulator, a classifier was trained

to predict the latency value correctly for an NoC. Usually in literature is seem the use of analytical models to model an NoC and predict some metric such as required area or latency value, but in practice, analytical models are tight and need few parameters (Marculescu et al., 2005). On this, the use of Artificial Intelligence can lead to a flexible solution to predict any desired metric. Specifically for latency prediction, performance metric, few papers approached this metric using AI techniques.

Concerning the use of a Genetic Algorithm (GA) to speed up the process, some method is required to avoid running all design space. The goal is to analyze the minimum number of NoCs models as possible and find optimized configurations. Heading this presents as the main contribution to the combined usage of ML techniques with the GA based optimization method. By using this approach, we expect to accelerate both sides: latency prediction value and speeding up the design space exploration for NoCs architectural configurations.

The third contribution of this relies on the use of ML techniques to predict also area and power consumption. Therefore, this solution can target the optimization of three constraints. However, since the approach is mono-objective, values of power and area are only reported hence not used to drive the space exploration. This is left as future improvements.

2 RELATED WORKS

The main goal of this paper is to develop a machine learning-based tool for NoC design space exploration. That said, this section aims to show some papers that have the same main goal. Every cited paper only uses regular topologies. Irregular topologies are out of scope.

Wang et al. used an ACO to optimize the number of virtual channels for a given regular NoC. This optimization is made after an analysis of the application graph and trying to detect a waste of resources. Besides it, this work restricts the NoC configuration to XY routing algorithm and does not support varying in other NoC attributes. They adopted MPEG4 and VOPD applications to compare the performance and resource-saving. The gain was up to 33% and 24%, respectively, for each evaluated application (J. Wanget al., 2013).

Rout et al. developed another work that optimizes the virtual channel number. The solution analyzes the traffic and determines how many virtual channels are needed. This implementation was made in hardware, it does not use any machine learning resource,

but using two units to handle it: Power Management Controller (PMC) and Utilization Computation Unit (UCU) Rout et al., 2018). They utilized noxim simulator with six traffic patterns (Bit-reversal, Butterfly, Random, Shuffle, Transpose1, and Transpose2) to compare a baseline router with their technique. The achieved results presented a power reduction up to 83% with only 4.2% of throughput penalty.

Sangaiah, Hempstead, and Taskin implemented a regressive model to explore design space. The main goal is to find the best NoC configuration for a determined CPU (mono or multicore). This work has two steps. First, the solution explores the CPU architecture, and after, it finds the optimized NoC. The proposed solution uses nine attributes to predict performance. Area metric is accounted for by Cacti (memory cache) and Orion 2.0 (NoC) using the 65nm technology. The achieved accuracy is until 88.2%. Another obtained result was the reduction in simulation time, from 1370 years (using gem5) to 180 hours (using the proposed work). However, they did not cite which NoC attributes were optimized (Sangaiah et al., 2015).

Another work is the of Zhang et al. they created an approach to explore the design space of NoC using Congestion Matrix (Zhang et al., 2016). This technique is based on congestion of the routing path and the diameter of the network. Moreover, this framework provides a circuit area and power consumption (static and dynamic), both use Orion 2.0 (Kahng et al., 2009) and DSENT tool (Sun et al., 2012). Each configuration has seven parameters of NoC. Booksim 2.0 was adopted for gathering the results. Eight traffic patterns were used, and nine benchmarks to drive the simulator. The experiments were based on a 64-node (8x8) NoC with 65nm manufacture technology and 1GHz as working frequency. The achieved results showed until 92% of accuracy. According to the authors, when the injection increases from 0.10 to 0.55, the reliability fall to 63%, using ANN and SVM (Zhang et al., 2016).

Sinaei and Fatemi compared three multi-objective approaches to explore the NoC design space. The first two, Multi-Objective Simulated Annealing (MOSA) and Multi-Objective Particle Swarm Optimization (MOPSO), were developed by themselves, and the third is the NSGA-II algorithm. They considered as metric energy consumption, performance, and cost (Sinaei & Fatemi, 2016). Each metric was modelled as a Kahn Process Network (KPN) to estimate the individual metric cost. As a case study, they evaluated using an M-JPEG encoder application and the architecture model is defined as a general-purpose CPU, DSPs, ASICs, SRAMs, and DRAM. Authors verified that MOSA accuracy is better than MOPSO and NSGA-II in all generations.

Obaidullah and Khan used a Hybrid Multi-Swarm Optimization to improve the NoC configuration. They focused in to optimize the task mapping and NoC architectural. It had two steps: first, the solution enhances the task mapping, and then the NoC configuration is optimized. The proposed application explores the task mapping to provide the best setup, which results in optimal communication cost, power, and chip area (Obaidullah & Khan, 2017). They combined the two tasks (task mapping and NoC configuration) at the same swarm (each one is a sub-swarm), to find the more optimized solution. The area and power suffered a reduction in the range of 30-120% and 8-40%, respectively.

Table 1 presents a comparison among the papers.

2.1 PAPER ANALYSIS

Table 1: Comparison among cited papers and brief description of them.

Paper	**Litograph Process (nm)**	**Benchmark**		**Used Simulator**	**Goal**
		Synthetic	**Real Applications**		
J. Wang et al. (2013)	-	X		Not specified	Optimize virtual channel number
Sangaiah et al. (2015)	65	X		Cacti 6.5 and Orion 2.0	Optimize CPU architecture
Zhang et al. (2016)	65	X		BookSim 2.0	Optimize four NoC attributes
Sinaei & Fatemi (2016)	65		X	Not specified	Optimize power, performance and area
Obaidullah & Khan (2017)	90	X		Not specified	Optimize task mapping and NoC configuration
Rout et al. (2018)	32	X		Noxim	Optimize virtual channel number

Two kind of information are relevant for the analysis and comparison with the approach presented here. First, all papers use 32nm, 65nm, or 90nm as manufacturing technology and do not support predictions about them, requiring a library. Second, three of four papers adopted synthetic applications for their benchmarks. That makes it easy to characterize the applications with, for instance, a specific number of packets or number of tasks. Also, there is a lack in describing more precisely the simulation approach adopted.

Concerning the advantages and disadvantages of each approach, this work overcomes all quoted papers in the number of attributes. This characteristic is important because more attributes allow us to consider the router's internal components behavior,

heading to more precise results. On the other hand, other approaches only support area or power prediction when tight to a particular lithograph process supported by the library. In our work, this is overcome since we can deal with different forecasts for these metrics. Currently, the main drawback of our work regards on applying only mono-objective solutions. The next Section discusses the solution architecture in details.

3 WORK PROPOSAL

This solution proposed in this work is based on Artificial Intelligence methods to speed up and predict metric values. Figure 1 presents the solution workflow.

Figure 1: workflow for entire solution, first it checks the values, if it is correct, the GA is called and starts its execution until meet the requirements.

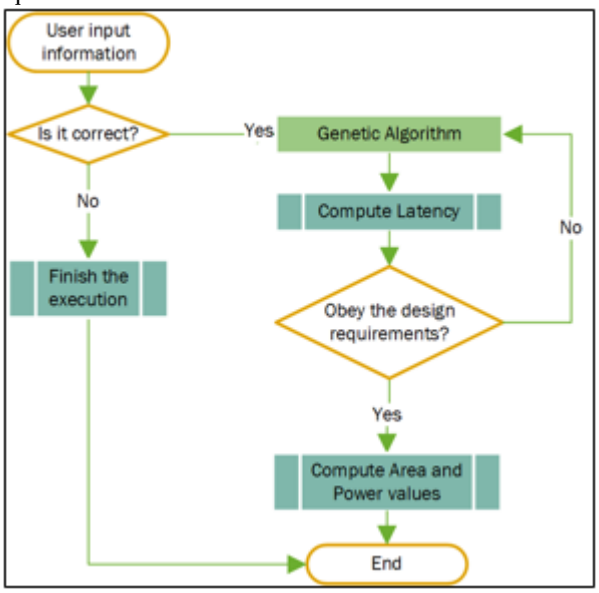


Figure 1 shows the workflow for the entire solution. Initially, the designer provides some necessary information such as the number of packets, required bandwidth (also called Packet Injection Rate), size of NoC, topology, and the expected average latency value. Having these pieces of information, the GA is initialized and evaluates every individual of the initial population. The algorithm will execute until someone individual attend the designer latency restriction or exhaust the number of evaluations. Before the best reached individual to be presented to the designer, their metrics are calculated (area and power requirements). Therefore, area and power metrics are not taken into account during the design space exploration phase.

Genetic Algorithms need a chromosome, it is the individual representation, and Figure 2 shows an example of an individual.

Figura 2: Each individual has ten parameters, being four supplied by designer, five are handled by GA and the latency is filled by GA using the latency predictor.

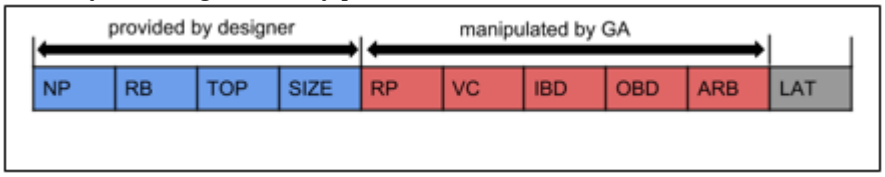


Figure 2 presents an example of chromosome. In total, each one has ten attributes, being nine handled or provided by the designer, already the last attribute is calculated using the trained classifier for latency prediction. Besides it, each classifier was fed by simulators as can be seen in Section 4.

4 METHODOLOGY

This work rely on two steps: AI methods (classifiers) are used to predict the area, latency, and power and a Genetic Algorithm performs tests with the attributes of NoCs seeking optimized configurations out of the search space. All the adequate classifiers were applied to this problem, aiming to find a better option. The tests were done using the Weka software, version 3.8, and using the datasets specified in Sections 4.1 and 4.2. Statistical test t-test was applied to validate the results. In next subsections are approached the datasets, starting with latency dataset.

4.1 LATENCY DATASET

Each instance has a network size between 2x2 to 16x16. These sizes were needed because the non-linear behavior in some circumstances, such as latency on 16x16 size using non-deterministic routing algorithm, does not obey to any linear formula. Still, there is the possibility to occur contention or deadlocks. As a result, the created dataset contains 686 instances.

We enhance the chromosome presented in Figure 2 with number of packets (NoP) and required bandwidth (RB). The NoP chosen values were 128, 1024, 2000, and 8192 per communication flow. Four values for RB were defined: 64, 512, 1024, and 2000Mbps. We experiment with the whole set of values for NoP and RB. This variety is necessary for improving the classifiers learning process (more different instances can lead to a better model).

Latency dataset has instances with five spatial distribution pattern: Bit-Reversal, Butterfly, Uniform, Perfect Shuffle, and Transpose Matrix. All instances concerns from

audio/video and signaling applications. Frequency and channel bandwidth are defined as 100MHz and 3200Mbps, respectively. The simulations execute until all packets are delivered since each configuration could result in better or worse classifier accuracy.

RedScarf simulator (da Silva et al., 2019) generates all instances with each simulation resulting in a different instance. RedScarf is a simulation environment targeting performance evaluation of NoCs models by simulating SystemC RTL descriptions. We evaluate ten classifiers, as presented in paper (Silva et al., 2019). Nonetheless, this simulator does not support area or power metric. Next Subsection explores these metrics.

4.2 AREA AND POWER DATASETS

We created one dataset for both metrics, differing only the last attribute (power or area attribute). Each dataset contains seven attributes that are listed below (along with its respective range of values evaluated):

- Injection rate: 0 to 0.99;
- Number of Buffer per Virtual Channel: 4 to 64;
- Number of Virtual Channels: 1 to 16;
- Number of bits per flit: 16 to 64;
- Frequency: 100Mhz to 10GHz;
- Lithography: 11, 22, and 45nm;
- Area/Power.

The first attribute, injection rate, is essential to consider due to the power consumption and injection rate are proportional. The second attribute is relevant because the size of the buffer reflects directly in the amount of area and power required, as seen in (Marculescu et al., 2005) and (Cilardo & Fusella, 2016). Still, according to the authors, only increasing the size from two to three in a 4x4 mesh NoC, the router area increases 30\%. Hence, this parameter may influence the area and power values. According to Mello et al. (Mello et al., 2005), virtual channels may be adopted to reduce the contention (increase the throughput) or help to comply with the deadlines. In both cases, it impacts on power and area required.

The width of the network channels impact directly on performance because the bandwidth is the product between channel frequency and channel width (Cilardo et al.,2015). In this way, it plays an essential role in NoC design. The next attribute,

frequency, impacts directly in power consumption. Lithography process affects area and power.

Lastly, area and power attribute is the result of the combination of the previous attributes. Power and area datasets have the same attributes aiming to standardize the datasets. Frequency, for instance, does not impact directly in the required area for the router, but, aiming at working with just one dataset, it was included in both.

During analysis phase we generate models instances using DSENT tool, which uses an analytical approach to calculate area and power required by routers (including, optical and electrical links) (Sun et al., 2012). We compared DSENT against ORION 2.0 tool (Kahng et al., 2009) and SPICE simulations (Rashid & Rashid, 2005) and discovered that, while they present an average error rate of 949\% and 9\%, respectively, DSENT relies on 93.2\% of accuracy. Faced with this result, we decided on using DSENT to generate the instances. In this work we test a dataset containing 108,180 instances.

Next section covers the attribute selection for the datasets.

4.3 ATTRIBUTE SELECTION

Designers model the datasets considering all significant attributes, but it could not be the best way to extract information from these characteristics. In this way, a standard option is the use of attribute selection methods, such as Best First or Evolutionary Search (Xue et al., 2016). According to the L. Wang et al. (2016), these methods can lead to reducing the required time to create the model, but when it is misused can lead to accuracy degradation. Therefore, it is necessary to evaluate if the attribute selection techniques do not lead to a loss of accuracy by the classifier. Ideally, these techniques would simplify the dataset, but without penalizing accuracy (they would only take noise and correlated) (Liu et al., 2017).

4.4 GENETIC ALGORITHM

We use an evolutionary approach, Genetic Algorithms (GAs), to perform the design space exploration. GAs have been successfully used as a solution for a broad set of minimization problems, including mapping problems, and they can deal with the hard constraints as performance or power (Strum, Chau et al., 2015). This technique uses each predictor presented earlier to avoid simulations. Thereby, it is expected that the execution to be even faster than the design space exploration using simulations. An essential issue

in GA algorithms is the objective function; for this work, we define average latency value as the objective function aiming at minimize it.

The tool is developed on Java language, version 8. As presented in Figure 1, GA executes until reaching one of these conditions:

- Maximum number of evaluations;
- Reached expected latency.

The conditions limit the execution to avoid non stopping executions and to not waste time and resources going beyond the designer requirements. Another essential characteristic is the size of the population. Each individual of the population is a potential solution for the problem, but in this case, there are many invalid NoC configurations. For instance: an NoC without buffer or routing algorithm. Therefore a repair strategy is required. It checks each vector field (NoC characteristic) and, when it detects something inadequate, it modifies the value using a random function, still according to the restrictions.

4.5 GENETIC OPERATORS

The mutation operator improves the diversity of a population avoiding many individuals with the same allele (Sastry et al., 2014). One possible improvement is self-adaptation. According to Karafotias, Hoogendoorn, and Eiben it allows the EA to use appropriate parameter values in different stages of the search process (Karafotias et al., 2015).

In the first moment, the mutation rate is 1\%, but if the GA seems that it is limited to local optimal (sub-optimal solution), it increments (5\% by time). This rate aiming search in whole design space, not only a point. This technique allows for reaching better results and major population diversity (Zamuda & Brest, 2015). We set crossover to two randomly selected points, as shown in Figure 3.

Figure 3: Example of two points crossover where three attributes were copied from mother and father.

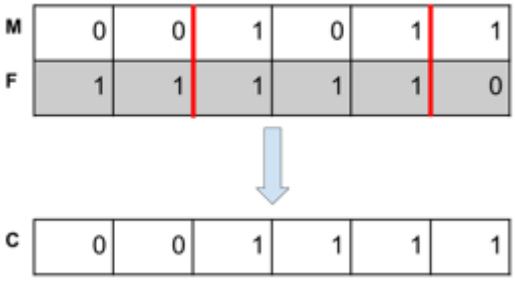


Figure 3 presents a crossover with two points to split. Three attributes are copied from mother individual (identified as "M" in image) and three from father ("F") to child (described as "C"). The red line delimits this segmentation, where inside the lines it is the region copied from the father. The Selection methods adopted in this work are detailed below.

4.6 SELECTION METHODS

In this work, designers have the choice to choose between the tournament or roulette wheel as a selection method. When the tournament is selected, the designer needs to provide the size of the tournament. Both are traditional methods for selection, but there is no guarantee that good individuals will be selected (Yadav &Sohal, 2017). Despite this, the roulette wheel tends to have a better result because it assumes that the chromosomes are selected based on their probabilities that are proportional to their fitness value. Its selecting principle is similar to that of a roulette wheel, while tournament may lead to minor diversity and, hence, a sub-optimal solution (Shukla et al., 2015).

We are not aware of any other work using a similar approach as the one employed in this work since we compare two selection methods. This is important since it is not possible to know a priori which one produces the best results. The experiments with the two methods are analyzed in Section 5. The following subsection approaches the choice of parameters values to genetic algorithm execution.

4.7 GENETIC ALGORITHM CONFIGURATION

Another issue is the choice of GA parameter values. There are two options to guide this process: to test all possibilities to each parameter, to analyze all results, and to find out the best configuration or use some automatic algorithm configuration. The last option exploits the configuration options using statistical tests to verify which are the best configurations (solutions) without having to perform all analyses. In this work, irace tool was used to discover the best values for each parameter. It implements an iterated racing procedure which is an extension of Iterated F-race (I/F-race) (López-Ibáñez et al., 2016).

5 RESULTS AND DISCUSSION

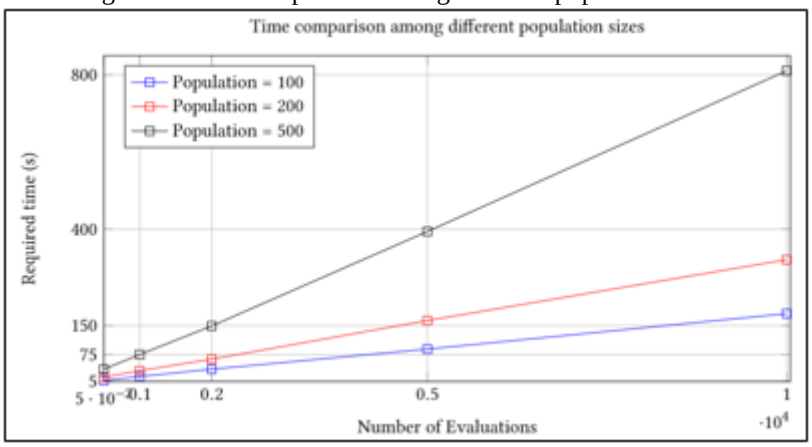
Below, we present the results of GA. First, we compare the required time to GA execution with different settings, then the parameters of GA and a naive execution against

an optimized one. Finally, we compare the GA self-adaptation against a default GA execution.

5.1 TIME COMPARISON

One characteristic of GA is a significant amount of time required for its execution when compared with the local search algorithm, for instance (Chenet al., 2015). We need to know how much time is necessary by solution execution. It is essential to know because a vast population demands more time to repair, evaluate, and to perform GA operators. In an ideal scenario, the population size would be the smallest as possible. Figure 4 shows a comparison among three sizes of populations, each one with four number of evaluations.

Figure 4: Time comparison among different population sizes.



As we can see, the Figure 4 confirms the idea that the required time increases as long as the population or number of evaluations increase. The number of evaluations increases almost proportionally to demanded execution time. The most significant difference is when the population equals to five hundred, and the evaluation number equals to ten thousand. In this case, the difference between the expected growth and the registered was 5.8%; all the others were less than 1.98%. Next section approaches the choice of values for all parameters involved in this solution.

5.2 PARAMETERIZATION OF GENETIC ALGORITHM

Irace was applied to find out the adequate values of GA parameters. The values found are presented below.

- Size of population: 92;
- Number of evaluations: 466,358;

- Crossover rate: 0.277;
- Elitist strategy: No;
- Selection Method: Roulette Wheel.

The roulette wheel method is described, such as a selection method that guarantees the best population diversity when compared with tournament method in related literature. Even though this feature does not guarantee better results when compared to other selection methods, the result of the irace notes that it has achieved a better result. All experiments were carried out considering these values.

5.3 COMPARISON BETWEEN OPTIMIZED AND NON-OPTIMIZED GA

Below we present a comparison between an optimized and a non-optimized GA execution, the goal is to verify if the optimization effectively overcomes other configurations.

- Size of population: 50;
- Number of evaluations: 466,358;
- Crossover rate: 0.9;
- Elitist strategy: Yes;
- Selection Method: Roulette Wheel.

This configuration is similar to the optimized provided by irace, and the main difference is a higher crossover rate and smaller size of the population. The number of evaluations was kept equals to both setups to guarantee the same evolution process for both cases. In all performed experiments, the metric was average latency. Figure 5 compares the result of executions.

Figure 5: Three variations of the application, keeping the same number of packets and size, but varying the required bandwidth.

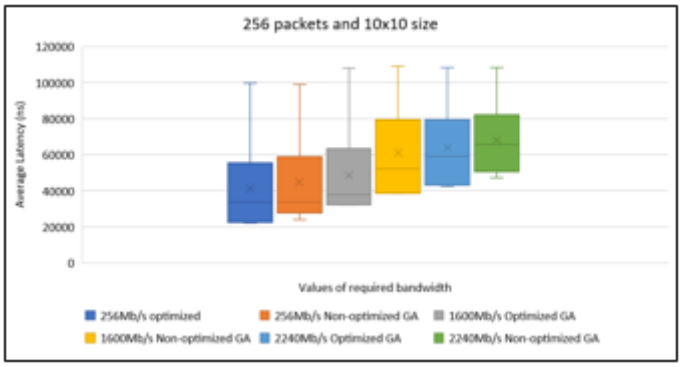
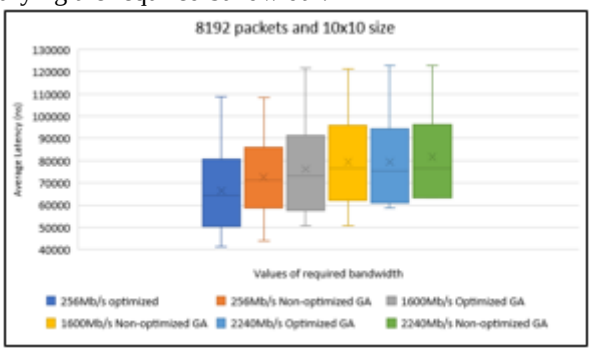


Figure 5 presents three scenarios (with two configurations each one). Firstly, it is the scheme which requires less network bandwidth, 256Mb/s, thus tending to have less contention. Due to the lower network demanding, both solutions (optimized and non-optimized) reached near result, but the optimized performed better. The second scenario presented, also two hundred and fifty-six packets, but with 1600Mb/s as required bandwidth, a similar behavior, optimized GA execution performing better than the default configuration, but with whole first quartile better than non-optimized GA. Besides that, the worst case in both situations have similar latency values. The configuration provided by irace overcame the default configuration up to 30% for the mean solution in the second case. The last two configurations, 256 packets with 2240Mb/s as bandwidth, also kept the behavior seen in previous configurations: optimized execution obtained better latency values.

Figure 6 shows other NoC configuration, now with 8192 packets, keeping the size equals to 10.

Figura 6: Now, with 8192 packets, the same three variations of the application, keeping the same number of packets and size, but varying the required bandwidth.



Comparing Figure 5 and Figure 6, it is possible to notice that the second has the same behavior as the first. All three situations presented the optimized GA execution overcoming the non-optimized in all cases. Figure 6 reflects the same performance. It means the configuration of GA provided by \textit{irace tool}, is adequate for all evaluated experiments and, consequently, we must adopt for all cases. Also, it is possible to notice an increment on latency values for the worst case, when compared with Figure 5. The literature shows the same pattern, since the more communication dependent the application, the greater the use of network resources tends to be, which in turn tends to increase the average latency.

Trying to observe if the mentioned result is consistent in environments with a higher number of packets, Figure 7 presents another result.

Figure 7: Now, with 24000 packets per communication flow, the same three variations of the application, keeping the same number of packets and size, but varying the required bandwidth.

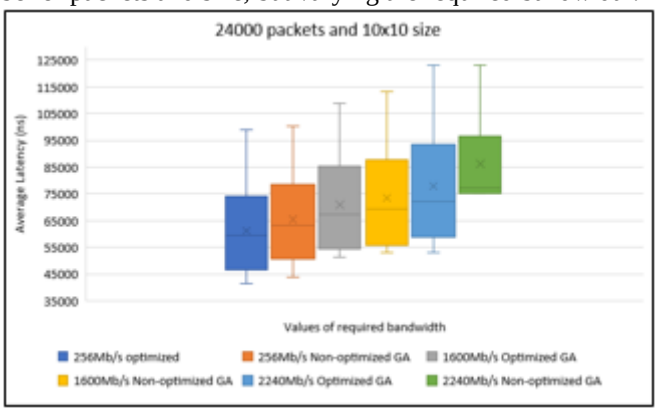


Figure 7 confirms the hypothesis which the GA optimized by irace parametrization overcome in all cases a simple GA configuration. Specifically, about the last two experiments, both with 2240Mb/s, a more significant difference among the values than in the previous experiments is perceptible. This difference does not occur in any other experiment and, hence, we can not affirm that is a general demeanor.

Figure 8 presents the same experiment that was presented by Figure 5, but with size equal to 14x14.

Figure 8: Now, with 256 packets per communication flow, but size equals to 14x14 and the same three variations of the application, keeping the same number of packets and size, but varying the required bandwidth.

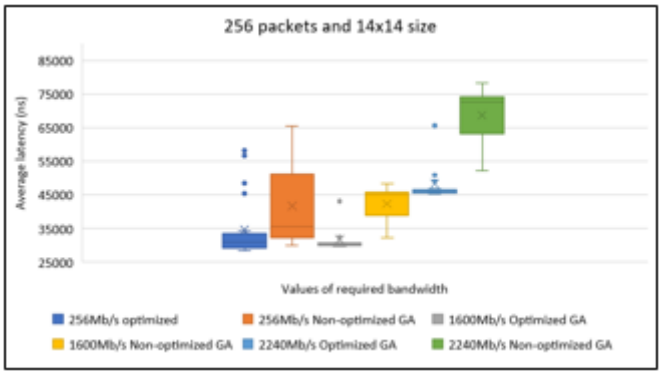


Figure 8 shows a similar behaviour to Figure 5, the optimized execution overcame a naive execution. Besides that, the data distribution is different, Figure 8 shows the values of optimized execution more concentrated, although the outliers. It means that the GA reached the best or near the best value several times. Specifically, the results for optimized 1600 and 2240Mb/s, it is not clear to notice the quartiles or median, and it corroborates this statement. All results of naive execution show sparse values when

compared with the other three results (optimized), and the upper limit is until 30% greater than the median value.

Figure 9 and 10 complements the experiment showing more two cases.

Figure 9: Now, with 8192 packets per communication flow, the same three variations of the application, keeping the same number of packets and size, but varying the required bandwidth.

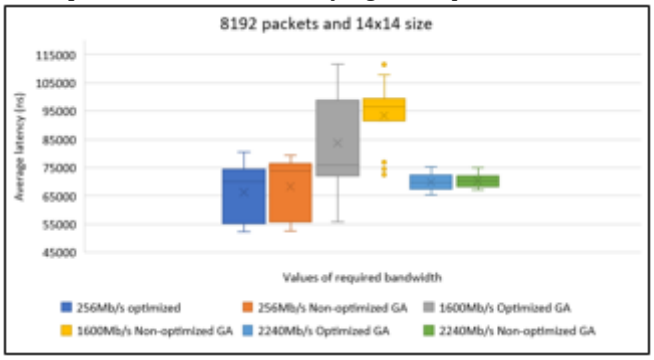
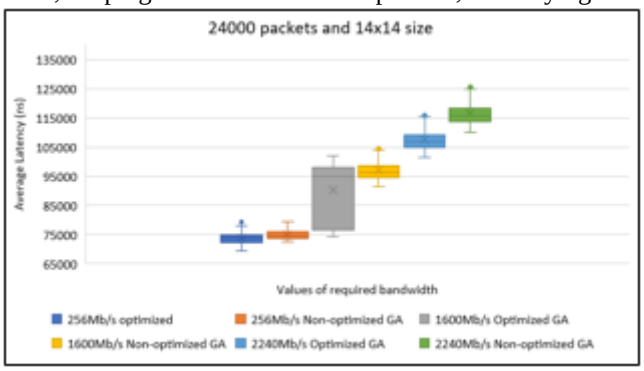


Figure 9 shows that, for 256Mb/s, optimized and non-optimized execution had a similar result, with a difference less than 6% for value of median and 3% for mean. Already for second configuration, 1600Mb/s, the optimization had its best gain, where only the outliers from non-optimized execution overcame the mean of optimized execution. However, the worst case for optimized GA was surpassed by non-optimized (lowest value).

Figure 10: Now, with 24000 packets per communication flow with size equals to 14, the same three variations of the application, keeping the same number of packets, but varying the required bandwidth.



The first configuration, 256Mb/s of required bandwidth, in Figure 10 shows the first quartile for optimized GA was better than all first quartile for non-optimized execution. The naive execution cannot explore all design space with the same quality of the optimized. Higher quality means lower NoC latency. For the last two cases, we have an unexpected result: the best result for non-optimized execution could not reach the mean value of optimized GA. Particularly in the last case, 2240Mb/s, the best value of non-

optimized cannot reach the median value of optimized GA. Faced with this result, we can conclude that for size equals to 14x14, and the Genetic Algorithm depends clearly on the optimization to reach the best possible result.

For all last six Figures, it is possible to identify the impact of needed bandwidth in average latency, what is expected based on literature. The attribute selection phase also reached the same conclusion for this NoC attribute.

6 FINAL REMARKS

This paper has as the primary goal to optimize the NoC design space exploration using Artificial Intelligence to speed up the process. The contributions were the use of ML classifiers to avoid unnecessarily simulations and a Genetic Algorithm to perform the design space exploration. In this way, we combined both methods to find suitable NoCs configurations complying with application constraints. After running the experiments, we could conclude that the proposed solution was able to head on satisfactory results. The predictors based on ML techniques have achieved accuracy up to 99.9% for the area and power prediction and up to 89% for latency inferences. The second step, GA, was efficient to explore the design space adequately and to provide an adequate NoC configuration for the designs. The optimized version of GA, it provided by irace tool, could surpass the non-optimized one by up to 30%. This type of result demonstrates the importance of the correct algorithm parametrization, which was one primary concern in this work. As the second optimization phase, we compared the default GA and a GA with the self-adaptation method. Analyzing the results, we noticed the behavior foreseen in the literature: the self-adaptation was able to improve the overall system performance.

As future works, we intend to expand our router characteristics to include operation frequency and supports dynamic voltage frequency scaling (DVFS), as well as targeting power and area optimizations.

ACKNOWLEDGEMENT

This research was supported by NPAD/UFRN.

REFERENCES

Chen, B., Zeng, W., Lin, Y. & Zhang, D. (2015). A new local search-based multiobjective optimization algorithm. IEEE Transactions on Evolutionary Computation, 19 (1), 50–73.

Cilardo, A. & Fusella, E. (2016). Design automation for application-specific on-chip interconnects: A survey. Integration, the VLSI Journal, 52, 102–121.

Cilardo, A., Fusella, E., Gallo, L. & Mazzeo, A. (2015). Exploiting concurrency for the automated synthesis of mpsoc interconnects. ACM Transactions on Embedded Computing Systems (TECS), 14 (3), 57.

da Silva, E. A., Kreutz, M. E. & Zeferino, C. A. (2019). Redscarf: An open-source multi-platform simulation environment for performance evaluation of networks-on-chip. Journal of Systems Architecture, 101633.

https://doi.org/https://doi.org/10.1016/j.sysarc.2019.101633

Hsu, P. (1938). Contribution to the theory of" student's" t-test as applied to the problem of two samples. Statistical Research Memoirs.

Kahng, A. B., Li, B., Peh, L.-S. & Samadi, K. (2009). Orion 2.0: A fast and accurate noc power and area model for early-stage design space exploration, In Design, automation & test in europe conference & exhibition, 2009. date'09. IEEE.

Karafotias, G., Hoogendoorn, M. & Eiben, Á. E. (2015). Parameter control in evolutionary algorithms: Trends and challenges. IEEE Transactions on Evolutionary Computation, 19 (2), 167–187.

Liu, J., Lin, Y., Lin, M., Wu, S. & Zhang, J. (2017). Feature selection based on quality of information. Neurocomputing, 225, 11–22.

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M. & Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives, 3, 43–58.

Marculescu, R., Hu, J. & Ogras, U. Y. (2005). Key research problems in noc design: A holistic perspective, In Hardware/software codesign and system synthesis, 2005. codes+ isss'05. third ieee/acm/ifip international conference on. IEEE.

Mello, A., Tedesco, L., Calazans, N. & Moraes, F. (2005). Virtual channels in networks on chip: Implementation and evaluation on hermes noc, In Proceedings of the 18th annual symposium on integrated circuits and system design. ACM.

Obaidullah, M. & Khan, G. N. (2017). Hybrid multi-swarm optimization based noc synthesis, In System-on-chip conference (socc), 2017 30th ieee international. IEEE.

Rashid, M. H. & Rashid, H. M. (2005). Spice for power electronics and electric power.

Crc Press.

Rout, S. S., Mondai, H. K., Juneja, R., Gade, S. H. & Deb, S. (2018). Dynamic noc platform for varied application needs, In Quality electronic design (isqed), 2018 19th international symposium on. IEEE.

Sangaiah, K., Hempstead, M. & Taskin, B. (2015). Uncore rpd: Rapid design space exploration of the uncore via regression modeling, In Proceedings of the ieee/acm international conference on computer-aided design. IEEE Press.

Sastry, K., Goldberg, D. E. & Kendall, G. (2014). Genetic algorithms, In Search methodologies. Springer.

Shukla, A., Pandey, H. M. & Mehrotra, D. (2015). Comparative review of selection techniques in genetic algorithm, In Futuristic trends on computational analysis and knowledge management (ablaze), 2015 international conference on. IEEE.

Silva, J., Kreutz, M., Pereira, M. & Costa-Abreu, M. D. (2019). An investigation of latency prediction for noc-based communication architectures using machine learning techniques. The Journal of Supercomputing. https://doi.org/10.1007/s11227-019-02971-x

Sinaei, S. & Fatemi, O. (2016). Novel heuristic mapping algorithms for design space exploration of multiprocessor embedded architectures, In Parallel, distributed, and network-based processing (pdp), 2016 24th euromicro international conference on. IEEE.

Strum, M., Chau, W. J. Et al. (2015). Using genetic algorithms for hardware core placement and mapping in noc-based reconfigurable systems. International Journal of Reconfigurable Computing, 2015, 1.

Sun, C., Chen, C.-H. O., Kurian, G., Wei, L., Miller, J., Agarwal, A., Peh, L.-S. &

Stojanovic, V. (2012). Dsent-a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling, In Networks on chip (nocs), 2012 sixth ieee/acm international symposium on. IEEE.

Wang, J., Li, Y., Chai, S. & Peng, Q. (2013). Minimizing virtual channel buffer for network-on-chip, In Fifth international conference on machine vision (icmv 2012): Algorithms, pattern recognition, and basic technologies. International Society for Optics and Photonics.

Wang, L., Wang, Y. & Chang, Q. (2016). Feature selection methods for big data bioinformatics: A survey from the search perspective. Methods, 111, 21–31.

Xue, B., Zhang, M., Browne, W. N. & Yao, X. (2016). A survey on evolutionary computation approaches to feature selection. IEEE Transactions on Evolutionary Computation, 20 (4), 606–626.

Yadav, S. L. & Sohal, A. (2017). Comparative study of different selection techniques in

genetic algorithm. Journal Homepage: http://www. ijesm. co. in, 6 (3).

Zamuda, A. & Brest, J. (2015). Self-adaptive control parameters randomization frequency and propagations in differential evolution. Swarm and Evolutionary Computation, 25, 72–99.

Zhang, M., Shi, Y., Zhang, F. & Liu, Z. (2016). Comrance: A rapid method for network-on-chip design space exploration, In Green and sustainable computing conference 2016 seventh international. IEEE.