This is an early accepted version of the paper; the final version of this work is published in IEEE Design & Test special issue associated to the 16th IEEE/ACM International Symposium on Networks-on-Chip (NOCS 2022), DOI: 10.1109/MDAT.2022.3202996. Copyright belongs to IEEE.

SynFull-RTL: Evaluation Methodology for RTL NoC Designs

Neiel Leyva, *Barcelona Supercomputing Center, Universitat Politècnica de Catalunya,*
Alireza Monemi, *Barcelona Supercomputing Center,* and Enrique Vallejo, *Universidad de Cantabria, ICS FORTH*

***Abstract*—SynFull is a widely employed tool that generates realistic traffic patterns for the performance evaluation of a NoC. In this work, we identify the main limitations of SynFull: high variability and long simulation time and also that these limitations increase when SynFull is integrated with RTL designs. SynFull-RTL employs a statistical approach, simulating each application macro-phase only once and averaging according to its probability of occurrence and the measured traffic load. SynFull-RTL obtains higher accuracy than the original version and reduced variability, with observed $40\times$ reduction in simulation time and resources. A use-case with ProSMART validates the results.**

***Index Terms*—SynFull, Verilator.**

I. Introduction

SIMULATION is very important step to evaluate the performance of new designs. The fidelity of the results depends mainly on the characteristics of the network and the traffic model used. The more detailed traffic in the model is used, the higher the accuracy of the results, but also the higher the load and the simulation time.

SynFull [1] introduces a Markov-chain-based synthetic model that mimics real application traffic. It is very attractive, because it allows to feed the simulation with a realistic traffic pattern in a simple and fast way. SynFull divides the simulation into different long periods of time, denoted macro-phases, derived from traces of a real execution of the application.

However, in this paper we identify that SynFull does not execute all the phases in the proportion found in the real application. For this reason, the variability of the generated traffic may be very large and the results of the simulations become unreliable, since different executions using the same SynFull model may receive traffic that significantly differs. This may be remedied by running very long simulations, but it increases computational requirements.

In this work, we modify SynFull to feed an RTL simulator. Because of its increased simulation requirements, further enlengthening simulations to mitigate the innate variability becomes unfeasible. To avoid these problems, we introduce a novel methodology based on SynFull, denoted SynFull-RTL, which is both faster and more precise than the original implementation. Although the proposed methodology is not exclusive for RTL simulation, it is particularly relevant for these models because of their increased simulation cost. SynFull-RTL evaluates each macro-phase in isolation, and determines the average latency result according to the theoretical probability of occurrence of each macro-phase in steady-state and its measured traffic level.

This article was presented at the 2022 IEEE/ACM International Symposium on Networks-on-Chip and appears as part of the Design & Test special issue. Digital Object Identifier 10.1109/MDAT.2022.3202996

The methodology introduced in this work provides measurements faster and with high accuracy, leveraging the SynFull models based on real applications. The main contributions are:

- An integrated model of SynFull with RTL NoC router models, which allows to quickly evaluate the performance of the model using realistic traffic.
- An analysis of the main limitations of SynFull in this environment, mainly the use of a single seed, its high variability and the long simulation times.
- SynFull-RTL, an evaluation methodology that samples SynFull macro-phases and averages them according to their probability of occurrence and their traffic level.
- An evaluation of SynFull-RTL, which shows that it can provide more accurate results than the original SynFull approach while requiring $8\times$ to $40\times$ less simulated cycles and reducing time-to-solution by up to $40\times$.

II. Background

A. Traffic Modelling

Traffic modelling is very relevant for accurate performance evaluation of NoCs. There are multiple methodologies to generate traffic, including synthetic models (such as random uniform or permutations), use of large traces such as Netrace [2] or simulating a whole application, such as gem5 modelling. Traces or whole application simulation provide the highest accuracy, but they require huge files, large amounts of memory and long simulation time. [2] presents an interesting discussion and overview of the limitations of different models.

For the designers, a relevant feature is execution time; that is, the designers often need to test a new experimental feature in the NoC and have a hint about its estimated performance. In these cases, for waiting times is undesirable, because it delays decisions about new features and possible changes.

B. SynFull

SynFull is a tool designed to facilitate fast and realistic NoC evaluations. SynFull generates random traffic based on real applications, without relying on traces or full-system simulation. SynFull has been used in many relevant research works, such as [3]–[7].

SynFull divides each application into blocks called macro-phases, each of them lasting for an interval of 500,000 cycles.

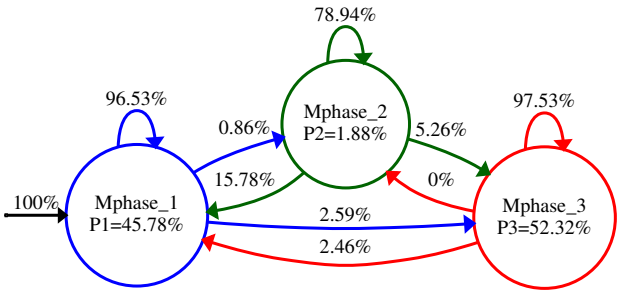


Fig. 1. State diagram of the macro-phases in the Barnes model. The number next to each arrow indicates the transition probability.

Each macro-phase models the behavior of an interval of the application. Each macro-phase has different characteristics, such as the packet injection rate and the pattern of destinations.

SynFull models define macro-phases as nodes of a Markov chain. Each model defines a low number (2 to 10 in the available models) of macro-phase types, obtained from traces of the actual application. Inside each macro-phase, the model that defines the traffic is denoted micro-phase, and it also follows an internal Markov model. The model also defines the transition probability between macro-phases i and j, p_{ij}, and the probability of occurrence of each macro-phase i in steady-state, P_i. In a Markov model, they are related by $P_j = \sum_i (P_i \times p_{ij})$.

Figure 1 depicts the macro-phase structure of the Barnes model. It comprises 3 macro-phase types, each one with a different probability of occurrence (P_1, P_2, and P_3). Note that these probabilities differ largely, with P_2 being very low. Transition probabilities (arrows) indicate how likely is to change from one macro-phase type to another, after each interval. The starting macro-phase is always type 1.

A simulation with SynFull injects traffic according to the current macro-phase type for the complete interval of 500,000 cycles. Then, it randomly changes to another macro-phase, according to the transition probabilities in the model. Although SynFull implements a hardcoded random seed, the sequence of macro-phases simulated is not always the same for each traffic model. This issue, which is further analyzed in Section III-B, occurs because the random sequence varies with packet reception, which in turn depends on the simulated NoC model.

C. ProSMART

ProSMART[1] [8] is a fully parametrizable NoC router design written in SystemVerilog, integrated with a module that supports multihop bypass. The NoC is configurable with many state-of-the-art features such as virtual channels, virtual networks, hard-built-in QoS, multicast, multihop bypass, different routing algorithms, and network typologies.

ProSMART allows injected flits to skip multiple routers in a dimension within a single cycle, resulting in a drastic latency reduction. The system employs the HPC_{Max} parameter to define the maximum multihop length; in a 4×4 mesh, HPC_{Max} varies from 1 to 3. This allows to model both a traditional mesh and one with multihop bypass. The effectiveness of ProSMART is only evaluated in [8] using synthetic traffic patterns, and it lacks empirical evaluations using realistic traffic patterns. This work employs ProSMART as a use-case to validate the proposed evaluation methodology.

[1]https://github.com/amonemi/ProNoC

III. SynFull-RTL integration and analysis of limitations

This section introduces the integration of SynFull with an RTL NoC model to support realistic RTL simulations. It follows with an analysis of the main limitations of SynFull.

A. Integration of ProSMART with SynFull

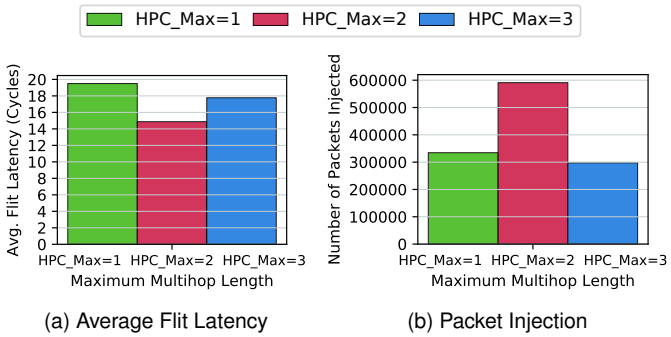
The ProSMART GUI provides a set of automation toolset that generates a simulation model for any custom-defined NoC model and also integrates several tools to facilitate performance evaluation. This section discusses two tools used for integration with SynFull: Questasim, and Verilator.

1) Questasim: It works directly on RTL code and uses the DPI functions to work with foreign programming languages like C++. The C++ module is compiled separately as a shared binary and the location is passed to Questasim with a flag, so it can find the binary at simulation time. This tool was used in the first approach to SynFull and it was decided to keep the socket communication. The DPI function has two parts, one is software and another one is RTL code. In software, receiving packets from the socket is handled and the packets are mapped on a direct connection to the RTL part of the DPI function, where we have ports for all the input queues where each packet is divided into flits.

2) Verilator: It converts the NoC RTL code to an equivalent cycle-accurate C++ model using Verilator simulator. Then it allocates three different software-based queues for each NoC's endpoint namely as injection, traversal, and ejection queues. These queues keep the desired traffic packets' information such as injection timestamp, the destination node address, and packet size in flits. The simulation model monitors the injection queues and feeds a corresponding packet to the NoC according to injection timestamp and NoC credit availability. Injected packets are moved to traversal queues and remain there until their corresponding packets reach their destination. At ejection time, packets are transferred to ejection queues. The simulation model collects several statistical information during the simulation and reports the performance results at the end of the simulation.

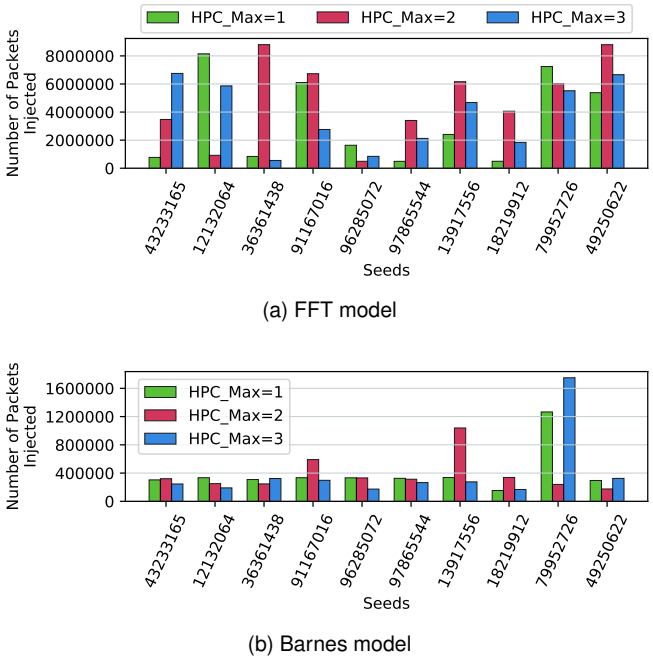
The ProSMART baseline simulation model can be effortlessly adopted with any existing traffic generator model such as Netrace or SynFull libraries. The integration is straightforward, and it only requires the mapping of injection/ejection functions from the traffic generator library to the ProSMART software-based injection/ejection queues.

For the following analysis and results, the version with Verilator is used, because it allows the execution of multiple simulations in parallel without license restrictions.



(a) Average Flit Latency (b) Packet Injection

Fig. 2. Variability observed in the simulation results of 10 million cycles (20 macro-phase iterations) using the original SynFull running the Barnes model.



(a) FFT model

(b) Barnes model

Fig. 3. Number of packets injected in 10 million cycles using the original SynFull and two different application models, for different seed values.

B. Analysis of SynFull Limitations

This subsection presents some limitations and challenges that were encountered during the integration of SynFull with ProSMART. First, it highlights that, even with a single hard-coded seed, minor differences in the NoC model cause large differences in observed performance. Next, it analyzes the variability, confirms that it comes from differences in the macro-phases executed, and observes that generated traffic may not be completely representative of the original application. Finally, it discusses simulation time.

1) Single seed and large variability: SynFull makes use of a pseudorandom number generator, which relies on a hard-coded seed value. Having a fixed seed always generates the same sequence of values for consecutive random generations. However, we observe that in SynFull this does not guarantee the same sequence of macro-phases and, thus, an equal or very similar traffic pattern for each model.

An analysis of the implementation of SynFull reveals that random calls are employed for macro- and micro-phase transitions as well as packet generation. Packet generation occurs both at the start of each macro-phase and in response to packet reception. The use of different NoC models modifies the delay of each packet, and in turn the evolution of the pseudorandom sequence and the transition between macro-phases.

Figure 2 illustrates the impact of this issue with three executions of Barnes with different values of HPC_{Max} in ProSMART. It shows packet injection and average flit latency. Larger HPC_{Max} values allow for higher multihop length and are expected to reduce latency, which may increase traffic. However, results in Figure 2b show that load is much higher with $HPC_{Max} = 2$ than $HPC_{Max} = 1$ or $HPC_{Max} = 3$. Similarly, Figure 2a shows that apparently $HPC_{Max} = 2$ obtains the best flit latency. The comparison is erroneous, because the traffic injected in each case differs significantly.

In conclusion, the traffic presents a large variability in both the traffic load and average latency between different simulations, and the use of a fixed seed prevents from observing different outcomes, resulting in erroneous conclusions.

2) Variability and application representativeness: We explore a modified version of SynFull that accepts a seed value as a parameter. Figure 3 shows the amount of traffic in several simulations of FFT and Barnes using 10 different seeds. Each simulation runs for 10 million cycles (20 macro-phases of 500,000 cycles). The large variability observed within a single seed also occurs for different seed values.

SynFull models comprise different macro-phases, each one with different micro-phase patterns of traffic injection and load. The number of macro-phases depends on the application model; FFT and Barnes contain 5 and 3 different macro-phases respectively. Figure 4 dissects the amount of traffic injected in each macro-phase in both applications. Within a macro-phase the amount of traffic is almost constant, and barely depends on the multihop distance used in the router model. Error bars in these figures, which represent one standard deviation, are barely visible. Similarly, the average latency per macro-phase presented in Figure 5 is also quite constant.

These analyses confirm that the variability in the application traffic load observed in Figures 2 and 3 does not come from micro-phase variability, but it depends on the specific macro-phases simulated in each case. An analysis of the simulations confirms that the executions in Figure 3 with a higher injected load also simulate a higher number of macro-phases with high load (phase 5 in FFT and phase 2 in Barnes), and vice versa.

Figure 6 shows average flit latency results for three applications running for 10 million cycles, averaging results from 10 different seeds. The standard deviation is significant, so the result is not reliable. The high standard deviation is caused by the high variability of SynFull's packet injection process.

To guarantee that a simulation corresponds to the steady state situation, this is, it properly represents the application from which the model was captured, the observed percentage of occurrence of each macro-phase should be similar to the steady-state probability of occurrence of such macro-phase. However, this clearly does not occur in the different executions shown in Figure 3. Indeed, in some cases we observed that a given macro-phase (with low probability of occurrence) is

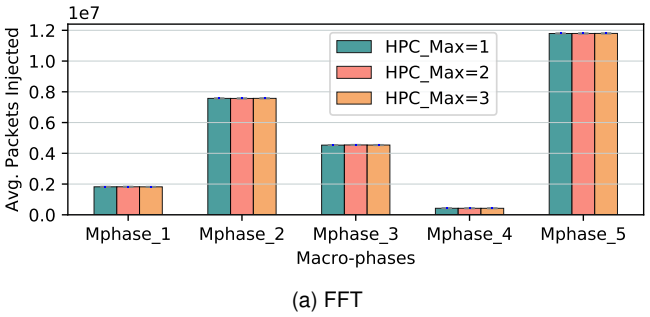


Fig. 4. Per macro-phase number of injected packets, for 10 million cycles (20 macro-phase iterations), averaging results from 10 different seeds.

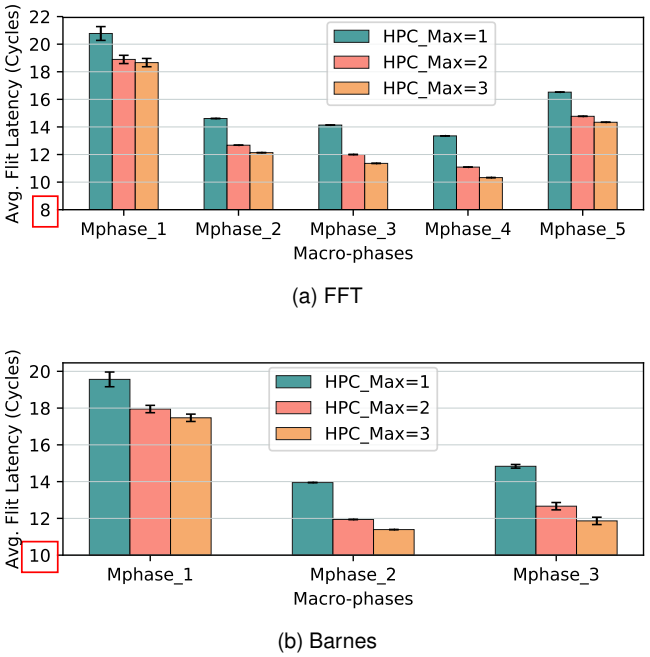


Fig. 5. Per macro-phase average flit latency, running for 10 million cycles (20 macro-phase iterations), averaging results from 10 different seeds.

never executed in certain simulations.

Longer simulations with a higher number of averaged simulations may be used to increase the traffic fidelity. Figure 7 shows the average number of injected packets for FFT and Barnes as the number of cycles increases, averaging the result of 10 different seeds. As expected, the average number of packets grows proportionally to the simulation length. However, the observed standard deviation remains very large, even averaging simulations of 400 million cycles.

Several conclusions can be obtained from the previous anal-

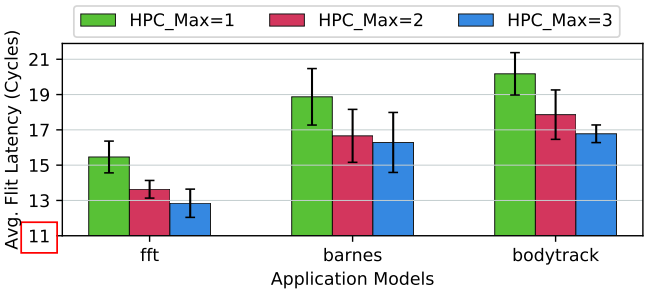


Fig. 6. Latency Results for 10 million of cycles in ProSMART+SynFull, averaging 10 different seeds.

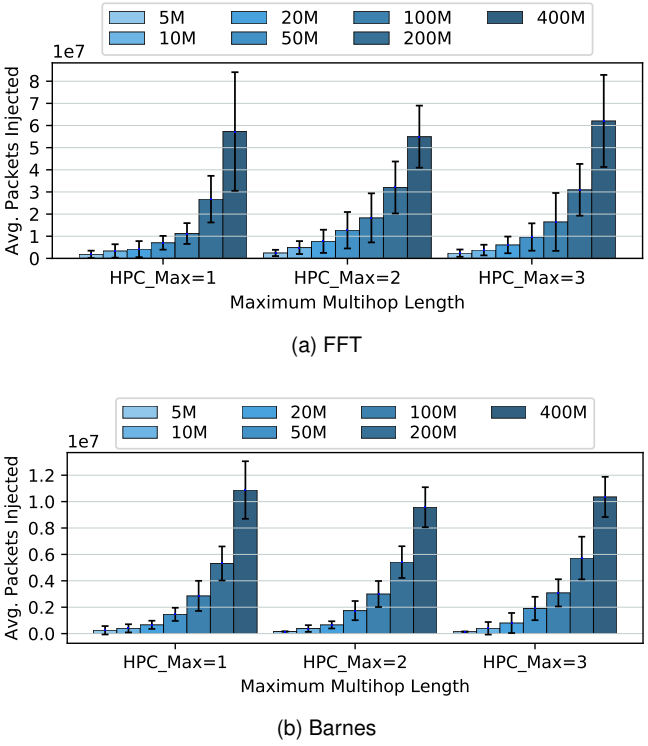


Fig. 7. Number of injected packets for different lengths of a SynFull simulation, from 5 to 400 million cycles, and resulting variability.

ysis. First, a single simulation may not represent accurately the behaviour of the application from which the SynFull model was obtained, because the macro-phases are not executed proportionally to their expected probability of occurrence. Indeed, some macro-phases may not be executed at all. Second, in order to accurately model the application, SynFull requires averaging multiple simulations with long simulation times.

3) Simulation time: RTL designs are more complex and detailed than the functional models in software simulators such as BookSim. For this reason, RTL simulations consume more resources and are slower than high-level software simulations.

The previous analysis suggests that a large number of simulated cycles are required for convergence, such as 400 million. Running in a typical HPC cluster, we measured an average 5.4 hours of simulation using SynFull with Booksim. However, using SynFull integrated with an RTL design like ProSMART, we measured average simulation times of 39

hours, 7.2× slower. Besides the increased amount of computation resources, due to their slow execution speed, RTL simulations may exceed the maximum allowed job run time in some or all the cluster queues, forcing the use of longer queues with lower priority and typically more congestion. All these effects reduce the benefit of SynFull as a mechanism for fast evaluation of RTL NoC designs with realistic traffic.

IV. SynFull-RTL methodology

This section introduces SynFull-RTL, which solves the limitations of SynFull in RTL models identified in Section III. SynFull-RTL employs the original SynFull models, but executes each macro-phase in isolation and averages the results according to the steady-state probabilities in the model.

Our proposal simulates each macro-phase individually, avoiding a large variation in packet injection in each simulation. Figures 4 and 5 show that the standard deviation of average packet injection and average flit latency per macro-phase can be reduced to negligible values. SynFull-RTL employs two parameters: First, N represents the number of different simulations (different seeds) that are averaged per macro-phase. $N = 10$ is used in Figures 4 to 7. Second, L represents the length of each simulated macro-phase in each simulation, in iterations. Since each iteration corresponds to a fixed value of 500,000 cycles, L can be also indicated in cycles. Finally, the number of distinct macro-phases M depends on the application model. The $M \times N$ simulations can run in parallel, reducing the time-to-solution.

For each simulation i $(1 \leq i \leq N)$ corresponding to macro-phase m $(1 \leq m \leq M)$ we obtain the average flit latency $FlitLat_{im}$, the average packet latency $PktLat_{im}$, and the number of sent packets $NumPkt_{im}$ and sent flits $NumFlits_{im}$. The application model provides the probability of each macro-phase, P_m. From these values, we derive the average and the standard deviation of the packet and flit latency in steady state. The following discussion explains how to calculate packet latency, and the calculation for flit latency is analogous.

For each macro-phase m, the average and the standard deviation of the packet latency ($AvgPktLat_m$ and $SdevPktLat_m$) and average number of packets ($AvgNumPkt_m$) are calculated from the results of the N simulations. To obtain the overall results, we cannot simply average the results from all the M macro-phases, because they differ in their probability P_m and their load intensity. Instead, the value must be calculated using a weighted average, in which each weight w_m represents the overall percentage of packets that are injected in macro-phases of type m.

A normalized overall number of packets is defined by $\sum_{m=1}^{M}(AvgNumPkt_m \times P(m))$, which considers the contribution of all the macro-phase types. Therefore, the weight to be used for each macro-phase is defined in Equation 1:

$$w_m = \frac{AvgNumPkt_m \times P(m)}{\sum_{m=1}^{M}(AvgNumPkt_m \times P(m))} \tag{1}$$

With these weights that represent the contribution of each macro-phase type, the average packet latency can be simply calculated as:

$$AvgPktLat = \sum_{m=1}^{M}(w_m \times AvgPktLat_m) \tag{2}$$

To estimate the accuracy of the mechanism, we also calculate the standard deviation of the latency estimation. The standard deviation is the root of the variance. The variance of a linear combination of random variables is given by:

$$Var(\sum_{i=1}^{M} a_i \cdot X_i) = \sum_{i=1}^{N} a_i^2 \cdot Var(X_i) + 2 \sum_{1 \leq i < j \leq N} a_i \cdot a_j \cdot Cov(X_i, X_j) \tag{3}$$

For independent executions the covariance $Cov(X_i, X_j)$ is null, so we can estimate:

$$VarPktLat = \sum_{m=1}^{M} w_m^2 \times SdevPktLat_m^2 \tag{4}$$

And finally we obtain the standard deviation as $SdevPktLat = \sqrt{VarPktLat}$.

V. Evaluation

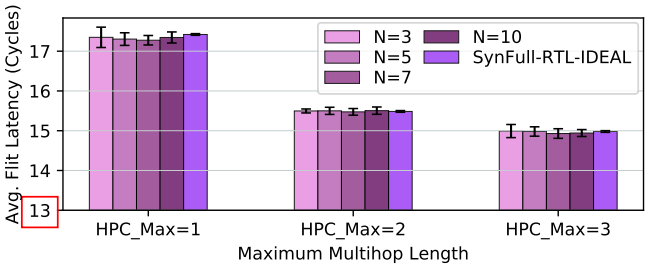
This section presents an evaluation of the accuracy and validity of the SynFull-RTL model, followed by a use-case evaluating the latency improvements of the ProSMART router.

Methodology: The modelled network is a 4×4 mesh using DOR and 1 virtual channel, with the ProSMART router [8] with $HPC_{Max} = 1, 2, 3$ in all cases. Unless otherwise noted, the evaluations in this section employ *SynFull-RTL* with $L = 20$ iterations per macro-phase and $N = 5$ seeds. In order to quantify the inaccuracy of our statistical methodology[2], we also consider a *SynFull-RTL-Ideal* model, which considers $L = 400$ iterations and $N = 20$ seeds. Whit these large parameters, the cycle count is so large that a variation in the number of seeds does not modify the results in a noticeable way. Evaluations using the original SynFull average $N = 10$ different seeds (to obtain similar confidence interval), running for the specified time.

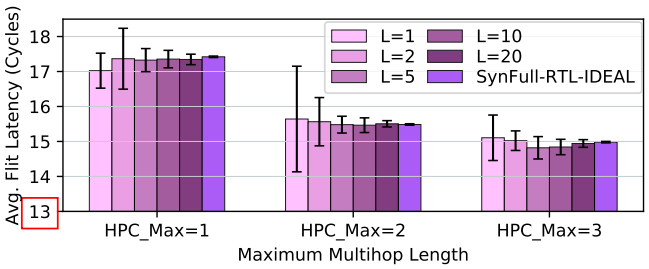
Plots include the confidence interval for a 95% confidence level. This interval is proportional to the standard deviation depicted in previous sections. In particular, it is calculated as $CI = \pm Z \cdot \frac{s}{\sqrt{N}}$, with $Z = 1.960$ for a 95% confidence level, s the standard deviation and N the number of samples.

SynFull-RTL parameters: Figure 8 analyzes the impact of the number of averaged seeds N and the number of iterations L of each macro-phase in SynFull-RTL. The application employed is Barnes, although other applications behave similarly. The average latency in Figure 8a hardly depends on the number of averaged simulations, N. Using $N \geq 5$ provides an error interval within a 1%. In Figure 8b, $N = 5$ simulations with different seeds are averaged, but the number of intervals simulated per macro-phase range from $L = 1$ to $L = 20$. The initialization impact, this is, the error observed when the number of iterations is low because the network is initially

[2]The inaccuracy introduced by the model generation is impossible to avoid.



(a) number of seeds, N ($L = 20$ in all cases)



(b) macro-phase iterations, L ($N = 5$ in all cases)

Fig. 8. Parameter analysis of SynFull-RTL using the Barnes application.

empty, can be appreciated clearly for small values; longer simulations progressively converge. In this case, a high value of L is more relevant for accuracy. Using $N = 5$ seeds and $L = 20$ provides average results which differ less than 0.67% of SynFull-RTL-Ideal.

Original SynFull vs SynFull-RTL: Figure 9 compares the estimation of SynFull-RTL (using $N = 5$ and $L = 20$) with the original SynFull, with different number of simulated cycles and $N = 10$ seeds. SynFull needs a higher number of seeds to obtain a comparable confidence interval. Four applications are selected: FFT, Barnes, Blackscholes and Bodytrack.

In the four cases we observe that short simulations provide incorrect results in SynFull, requiring many hundred of iterations to get close to the result provided by SynFull-RTL. The results with short simulations differ by up to 15.8% from the result provided by SynFull-RTL-Ideal. The bias of the error depends on the latency of the initial macro-phase 1. In FFT, Barnes and (to a lesser extent) Bodytrack the latency of macro-phase 1 is high, and short simulations overestimate the result because simulations always start there; in Blackscholes the bias is the opposite, for an analogous reason. Even with 400M simulated cycles, the confidence margin obtained with SynFull is clearly larger than with SynFull-RTL in all cases. SynFull-RTL average latency is within a 0.58% of the ideal value, whereas SynFull values with 400M cycles differ by up to 3.2%.

Longer SynFull simulations could not be run, because they exceeded the 2-day time limit in our systems. By contrast, the different simulations of $L = 20$ iterations (10M cycles) in SynFull-RTL can be run in parallel, reducing time-to-solution by up to $400M/10M = 40\times$. With these parameters, SynFull-RTL simulates $N \cdot L = 5 \cdot 20 = 100$ iterations per macro-phase (the number of macro-phases ranges from 2 to 10 in the available models). Compared to the $N \cdot 800 = 8000$ iterations (in the 400M case) in SynFull, computing resources are reduced by $8\times$ to $40\times$, while providing more accurate results.

Two aspects of Barnes are relevant. First, short simulations with $HPC_{Max} = 2$ and $HPC_{Max} = 3$ present excessive latency but reduced confidence interval. This occurs because the first macro-phase has both high probability and latency. In the "steady-state" execution mode (which ends simulation prematurely when results converge) this can produce incorrect results, because only one macro-phase is run. Second, the latency result with 400M cycles is overestimated, compared to the result from SynFull-RTL. We found that macro-phase 2 has a very low probability ($P_2 = 1.88\%$) but a significant weight ($w_2 = 22.6\%$) because it injects about 9 times more traffic than the two others. Additionally, it has the lowest latency, as seen in Figure 5b. An analysis of our SynFull simulations shows that it is underrepresented, with less than 1% of the overall iterations. We attribute this problem to its low transition probability and the bias towards the result from macro-phase 1 because SynFull always starts from it.

ProSMART evaluation using SynFull-RTL: Finally, Figure 10 presents SynFull-RTL results of all the available models, characterizing ProSMART. As expected, the confidence interval is very narrow. All the applications confirm that increasing the multihop length is profitable. Additionally, the use of $HPC_{Max} = 3$ does not yield as much benefit as increasing from $HPC_{Max} = 1$ to 2: In a 4×4 mesh, the occurrence of multihops of length 3 (i.e. from side to side of the mesh) is less frequent. Overall, average flit latency is reduced by 11.3% and 14.8% by increasing HPC_{Max} to 2 or 3 respectively.

VI. Related Work

In this work, we use SynFull as a tool to feed an RTL design. In the process, we detected several limitations related to simulation time and application representativeness. Similar problems were detected in [9], where SynFull is used to feed heterogeneous systems with CPU+GPU. In their case, the simulation time does not allow to explore all the GPU phases. Therefore, results do not reflect its higher traffic load. They modify the macro-phases to balance the relation between the workloads and the transitions between them, and they also modify the size of the macro-phases to include more micro-phases during the execution. To guarantee that all macro-phases occur, they record and replay the macro-phase original order. In contrast, our SynFull-RTL approach uses a sampling solution based on a weighted average that depends on both steady-state probability and traffic load of each macro-phase, with minor changes to the SynFull design using. This reduces the need of a large number of cycles to obtain reliable results.

The original SynFull mechanism [1] already considers steady-state convergence. Their proposal dynamically analyzes the convergence of the performance metrics and terminates the simulation prematurely. However, we have observed particular cases in which this alternatively instead increases the error of the result. Our approach relies on the steady-state probabilities of each macro-phase, derived from the transition probabilities, and sampled simulation.

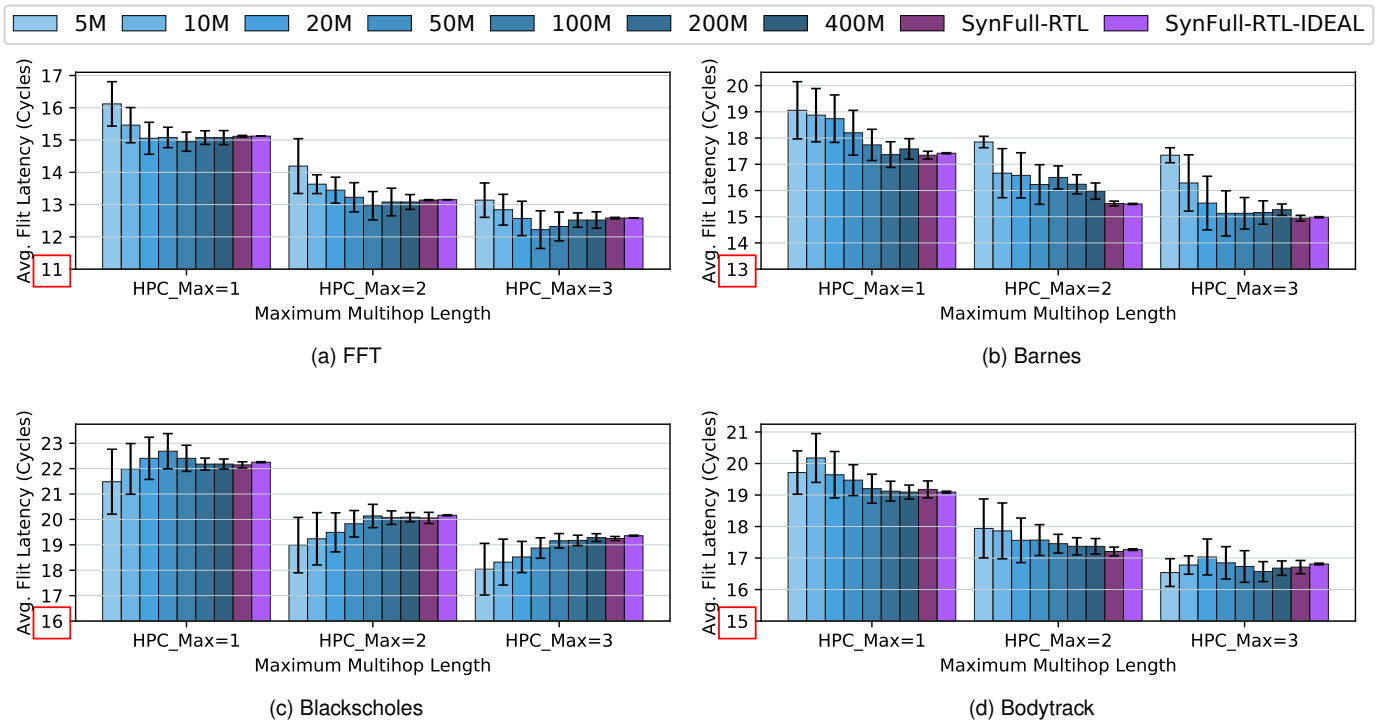


Fig. 9. SynFull (blue) and SynFull-RTL (purple) results for average latency with varying number of simulated cycles (in millions). SynFull-RTL runs each macro-phase for 10M cycles using N=5 seeds. Each SynFull bar averages results from N=10 seeds.

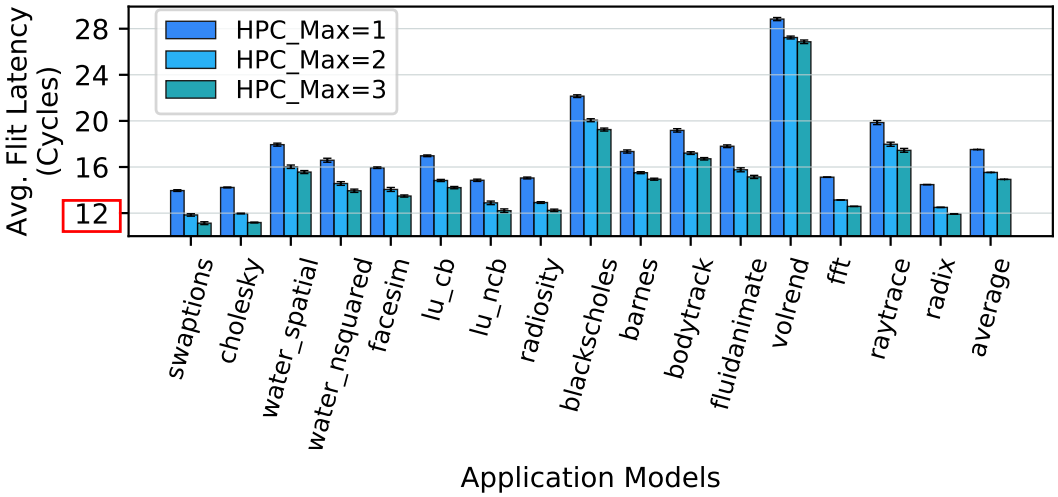


Fig. 10. ProSMART average flit latency running the complete set of available models using SynFull-RTL.

Other works focus on the SynFull methodology to capture the traffic burst in real applications. This is the case of Mocktails [10], which employs a similar approach to recreate the space-time memory access behavior in heterogeneous systems based on memory IP blocks. This is a very different approach to the SynFull-RTL methodology.

The methodology in FNoC [11] employs FPGAs to evaluate router models. While the simulation speed may be higher than our approach, it requires the development of complex traffic generation units and is limited by the FPGA resources.

A stochastic methodology for NoC traffic generation is introduced in [12], including a feedback loop that propagates delays through the memory hierarchy, absent in SynFull. The application of our sampling methodology to this approach could be considered in the future.

VII. Conclusions

The original methodology in SynFull presents significant limitations that are analyzed throughout this paper. The generated traffic has a large variability and it can be not completely representative of the original application being modelled, because some macro-phases may be underrepresented or not executed at all in a given simulation, despite reaching the status of steady state. To mitigate variability, long execution times

are required, and this is exacerbated when interfacing with complex RTL models.

The methodology introduced with SynFull-RTL simulates each macro-phase in isolation and averages the results according to the steady-state probability of occurrence and the measured traffic. While the methodology can be applied to traditional functional-level software simulations, it has shown to be particularly relevant for our RTL modelling approach, in which a straightforward connection could exceed the typical simulation times of many compute clusters and delay the results for many days. Indeed, our methodology not only reduces the required computing resources by up to $40\times$, but also reduces time-to-solution by up to $40\times$. Furthermore, this speedup is obtained with increased accuracy: we maintain the error interval within a 1%, lower than the figure measured with the original mechanism.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their helpful insights. We would also thank Miquel Moretó for helpful suggestions on this work. This work has been supported by the Spanish Science and Technology Commission under contract PID2019-105660RB-C22 and the European HiPEAC Network of Excellence. Enrique Vallejo has been partially supported by the Ministry of Universities, Subprograma Estatal de Movilidad, grant number PRX21/00757. This work also received funding from the European Union Horizon 2020 research and innovation programme under grant agreements number 826647 (EPI) and 946002 (MEEP).

REFERENCES

[1] M. Badr and N. E. Jerger, "SynFull: Synthetic traffic models capturing cache coherent behaviour," in *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, 2014, pp. 109–120.

[2] J. Hestness, B. Grot, and S. Keckler, "Netrace: dependency-driven trace-based network-on-chip simulation," 01 2010.

[3] F. Alazemi, A. AziziMazreah, B. Bose, and L. Chen, "Routerless network-on-chip," in *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2018, pp. 492–503.

[4] Z. Li, J. S. Miguel, and N. E. Jerger, "The runahead network-on-chip," in *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2016, pp. 333–344.

[5] S. Saxena, D. S. Manur, M. S. Shamim, and A. Ganguly, "A folded wireless network-on-chip using graphene based THz-band antennas," in *Proceedings of the 4th ACM International Conference on Nanoscale Computing and Communication*, ser. NanoCom '17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3109453.3109455

[6] N. Jindal, S. Gupta, D. P. Ravipati, P. R. Panda, and S. R. Sarangi, "Enhancing network-on-chip performance by reusing trace buffers," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 4, pp. 922–935, 2020.

[7] A. Ejaz, V. Papaefstathiou, and I. Sourdis, "DDRNoC: Dual data-rate network-on-chip," *ACM Trans. Archit. Code Optim.*, vol. 15, no. 2, jun 2018. [Online]. Available: https://doi.org/10.1145/3200201

[8] A. Monemi, I. Pérez, N. Leyva, E. Vallejo, R. Beivide, and M. Moretó, "PlugSMART: a pluggable open-source module to implement multihop bypass in Networks-on-Chip," in *2021 15th IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, 2021, pp. 41–48.

[9] J. Yin, O. Kayiran, M. Poremba, N. E. Jerger, and G. H. Loh, "Efficient synthetic traffic models for large, complex SoCs," in *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2016, pp. 297–308.

[10] M. Badr, C. Delconte, I. Edo, R. Jagtap, M. Andreozzi, and N. E. Jerger, "Mocktails: Capturing the memory behaviour of proprietary mobile architectures," in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, 2020, pp. 460–472.

[11] T. V. Chu, S. Sato, and K. Kise, "Fast and cycle-accurate emulation of large-scale networks-on-chip using a single FPGA," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 10, no. 4, dec 2017. [Online]. Available: https://doi.org/10.1145/3151758

[12] Y. Wang, G. Balakrishnan, and Y. Solihin, "MeToo: Stochastic modeling of memory traffic timing behavior," in *2015 International Conference on Parallel Architecture and Compilation (PACT)*, 2015, pp. 457–467.

Neiel Leyva is a PhD. Student at Universitat Politècnica de Catalunya and Research engineer at Barcelona Supercomputing Center (BSC). His current research interest are Networks-on-Chip (NoC) design and memory hierarchies for many-core systems. He received the M.S degree in Computer Engineering from Computing Research Center of the National Polytechnic Institute of Mexico. neiel.leyva@bsc.es.

Alireza Monemi received the M.S. and Ph.D. degrees in electrical engineering from Universiti Teknologi Malaysia, Malaysia, in 2011 and 2017, respectively. He is currently a Postdoctoral research associate at Barcelona Supercomputing Center (BSC). His current research interest is cache-coherent Networks-on-Chip (NoC) design. alireza.monemi@bsc.es.

Enrique Vallejo is an associate professor in the University of Cantabria and a visiting researcher at FORTH. His research interests are in computer architecture, mainly interconnection networks and parallel architectures. He holds a PhD in computer architecture from the U. Cantabria. enrique.vallejo@unican.es.