ANSim: A Fast and Versatile Asynchronous Network-On-Chip Simulator

Tom Glint
IIT Gandhinagar
India
tom.issac@iitgn.ac.in

Jitesh Sah
IIT Gandhinagar
India
jitesh.sah@iitgn.ac.in

Manu Awasthi
Ashoka University
India
manu.awasthi@ashoka.edu.in

Joycee Mekie
IIT Gandhinagar
India
joycee@iitgn.ac.in

***Abstract*—In a large chip, an asynchronous Network-on-Chip (NoC) is a suitable candidate for establishing an interconnection network between varied components. Architectural level simulation is an accepted methodology for evaluating such systems. In this paper, we propose a fast and versatile Asynchronous Network-on-Chip (NoC) Simulator - ANSim, which brings down the simulation time by $25\times$, compared to the state of the art simulators. It can model and analyze all synchronous, asynchronous, and mixed synchronous-asynchronous system of cores connected through NoC. ANSim can model routers with different delays, routers with asynchronous arbitration, connected in a wide range of topologies. ANSim supports individual routers modeled to have varying timing constraints. Further, it supports synthetic and real-workloads, and produces system-level latency, throughput, power, power-gating, and arbitration reports. ANSim has been verified against RTL models of NoCs, and other RTL verified simulators. An open-source synchronous NoC router, TNoC, and its asynchronous derivative are used to demonstrate ANSim's usefulness and features.**

***Index Terms*—Asynchronous NoC, Simulator, metastability.**

I. Introduction

Modern Systems-on-Chip (SoCs) are an integration of many heterogeneous modules [1]. Networks-on-Chip (NoCs) facilitate communication between these modules [2]. Designing synchronous NoC for a large number of modules is difficult due to clock synchronization issues and large clock power consumption [3], [4]. Asynchronous NoCs are a suitable candidate in such a scenario, as they don't rely on clocks and have lower latency under bursty traffic [4]. Architectural level simulation is an accepted methodology for evaluating such systems for different use cases, but simulator support for asynchronous NoC simulation is partial.

HNOCS [5] is the first heterogeneous and asynchronous simulator. Ved et al. have proposed PANE [4], a pluggable asynchronous NoC simulator. They have shown that PANE has the highest feature set among the same class of simulators while providing similar runtimes. However, these simulators still lack critical features (Table I) for correctly modeling and analyzing asynchronous NoCs for real-world workloads.

To deploy asynchronous NoCs for applications like control systems, machine learning, media encoding, etc., designers need to compare NoCs that are fully synchronous, fully asynchronous, mixed synchronous and asynchronous, and heterogeneous. *Heterogeneous* NoCs are NoCs with routers with varying timing constraints and can be synchronous or asynchronous. ANSim models all these cases and reports result so that design-related decisions can be taken early on in the design process. ANSim is $25\times$ faster, compared to state of the art, PANE [4]. It supports both synthetic and real benchmarks. The tool provides a power estimate of modern NoCs (with power gating, if possible) for a given configuration. Asynchronous NoCs suffer from arbiter metastability (explained in Section II), which is also modeled in the tool. Non-blocking power gating [6] for asynchronous NoCs is modeled in a simulator for the first time, to the best of our knowledge. The tool supports routers' easy modeling with different numbers of virtual channels (VC), buffer sizes, buffer allocation policies, routing policies, and arbiters [2] with a single configuration file.

TABLE I
Comparison of closest related works

Feature	BookSim	HNOCS	PANE	**Proposed**
Heterogeneous	No	Yes	Yes	**Yes**
Variability	No	No	Yes	**Yes**
Metastability	No	No	Partial	**Yes**
Design Space	+++	+	++	**++++**
Real-Benchmarks	Yes	No	Limited	**Yes**
Simulation Time	Very Fast	Slow	Slow	**Very Fast**
Power Gating	No	No	No	**Yes**

A. Related Works

A large number of NoC simulators have been proposed in earlier works. In the synchronous domain, Gem5 [7], Garnet [8], Multi2Sim [9], BookSim [10] are frequently used for synchronous NoC simulations, but are not capable of modeling asynchronous or heterogeneous routers within the same NoC.

HNOCS [5] is the first heterogeneous and asynchronous simulator and is based on Omnet++ [11]. HNOCS provides the ability to define the NoC with NED configuration files. Ved et al. have proposed PANE [4] based on Omnet++ and BookSim. Table I shows the comparison between existing asynchronous NoC Simulators and ANSim.

E. Beigne et al. [12] proposed an asynchronous router architecture and created a software framework to simulate NoCs with this router.

This work is supported through a startup grant at Ashoka University and grants received from SMDP-C2SD and YFRF fellowship under Visvesvaraya PhD scheme from the Ministry of Electronics and IT, and through SERB grants MTR/2019/001605 and CRG/2018/005013.

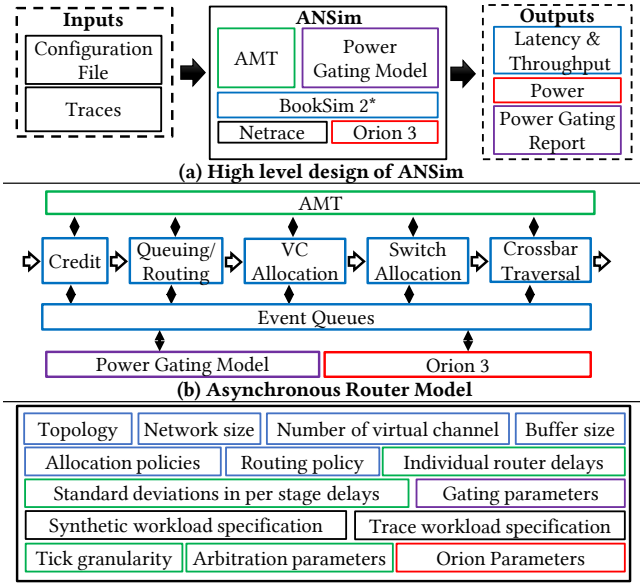

Fig. 1. High-Level Design and Asynchronous Router Model of ANSim.

II. Simulator Design

In synchronous NoCs, each action is initiated at a clock edge, and delays are measured in clock cycles. In asynchronous NoCs, the wall clock needs to be used and is captured in multiples of the smallest *tick*, where a tick is the smallest granularity of time (ps or ns). All timing parameters in the configuration file are specified in ticks. Each router and its components in the network are updated for every tick. Hence, when decreasing the *tick* from, say, 0.1ns to 0.01ns the simulation time also goes up from $1\times$ to $10\times$.

A. Design

Fig. 1 (a) shows all the modules involved in the design of ANSim. When compiled, it is an executable of size $\sim$ 1MB, which takes in a configuration file of the NoC that needs to be simulated and a set of override parameters as command-line arguments.

The *configuration file* contains the parameters shown in Fig. 1 (c). The parameters are color coded based on the module it configures in ANSim (Fig. 1 (a)), and are described in detail in [2]. Gating parameters includes gating-threshold [13] and break-even time [14]. The configuration specified in the base file can be over-ridden by supplying additional arguments. This allows for rapid scripting and parallel sweeping of the design space under consideration, without having to write a configuration for each NoC skew.

In Fig. 1 (a), *ANSim* reads the configuration file, creates the network topology based on BookSim's original model while the router models are created by the Asynchronous Model & Timing (AMT) module, which initializes the network. Traffic is either generated synthetically, based on a configuration file, or provided by traces pointed by the configuration file. Netrace [3] module is responsible for injecting real-world workload based packets into the network from the trace at a rate independent of the router specification. Orion 3 [15] module reads in the technology file and keeps a watch on the configuration of the router and the activity in the router while the simulation is taking place to compute and report the static and dynamic energy components of the NoC. The power gating module has visibility into each router and can set the state of the router to active, idle or gated, depending on the parameters specified in the configuration file during runtime. In this paper, static, non-blocking gating [6] policy is used, where the router is gated if it has been idle for a fixed duration. The AMT is also in charge of monitoring the arrival of data at the switch arbiter to model metastability. The details of implementation are provided in Section II.

TABLE II
Validation Data

(a) Flit latency reported by RTL tools and ANSim

Router	Tool	1 Hop	2 Hop	3 Hop
Asynchronous NoC [4]	Questa [19]	1.48ns	2.92ns	4.36ns
	ANSim	1.48ns	2.92ns	4.36ns
TNoC [17]	VCS [18]	3 cycle	6 cycle	9 cycle
	ANSim	3 cycle	6 cycle	9 cycle

(b) Flit latency reported by BookSim, PANE and ANSim

	Synchronous		Asynchronous	
	BookSim	ANSim	PANE	ANSim
Avg Flit Latency (ns)	16.56	16.56	17.99	17.68
Speedup of ANSim	1.1 (w.r.t. BookSim)		**26** (w.r.t. PANE)	

In NoCs with asynchronous routers, the crossbar forwards the input to the output on a first come first serve basis. The Switch Allocator (SA) allocates the crossbar when two requests are present. If the two requests arrive simultaneously at SA, it can lead to metastability, where the process of deciding which request to service takes arbitrarily long time. If two flits arrive with δ time difference, then, the time to come out of metastability is given by the formula, $\tau \times ln(\frac{Tw}{\delta})$, where τ and Tw are dependent on the circuit implementation [16].

Fig. 1 (b) shows a single asynchronous router within ANSim and the points of interaction with the rest of the models. The hollow arrows in Fig 1 (b) represent the data flow of the flit. In ANSim, each router in the NoC is individually addressable and can be set to either synchronous or asynchronous operation modes. The timing and behavior model of each router needs to be accordingly specified for individual routers.

At the end of the simulation, latency and throughput of the packet and flits are reported, power and area reports are generated, and power gating reports, including duration and times for which each router was gated, considering the break-even time are generated.

III. Verification of ANSim

ANSim was verified for both synchronous and asynchronous operations. For the asynchronous case, the end-to-end flit latency reported by ANSim when modeling a 2x2 NoC made of the asynchronous router specified in [4] was taken and was compared against its implementation in UMC 65nm technology library. For the synchronous case, TNoC [17] with 2 VCs was modeled in ANSim and VCS [18]. The latency reported by the tools are given in Table II (a). The *tick*, as defined in Section II, in ANSim was chosen as 10ps.

To verify the operation at system level, ANSim was compared with BookSim for the synchronous case and PANE

for the asynchronous case with hypothetical NoCs that act as targeted test vectors for all five stages modeled by the simulators. For the synchronous router, the cycle time was set to 0.5ns. For the asynchronous router, the per stage latencies of Input Queue (IQ), Route Compute (RC), VC, SA, and Switch Traversal (ST) were 1ns, 0.5ns, 0.5ns, 1ns, and 0.5ns respectively. The results for uniform traffic are presented in Table II (b).

Speedup: Run time for the system-level verification experiment was recorded to find the speedup of ANSim (Table II (b)). PANE requires Inter-Process Communication (IPC) using multiple sockets for synchronization as it decouples data-path (based on BookSim) and timing-path (based on Omnet++) into two programs. This mechanism allows for asynchronicity in PANE, but IPC sockets require many system calls and are slow [20]. ANSim does not have this limitation and is designed as a program with minimal dependencies. This allows ANSim to be compiled into a monolithic executable of size $\sim$ 1 MB, which can fit in most last level caches. Further, the asynchronicity is handled within the monolithic program using event queues. This allows ANSim to have 25 $\times$ speedup compared to PANE for the verification experiment. For longer simulation times, speedup increases further because of better memory management.

IV. Architectural Experimental Setup

To demonstrate the features and use cases of ANSim in Section V, we use the following experimental setup.

Routers - For constructing NoCs, a three-stage open-source router, TNoC [17], with two VCs and its asynchronous derivative were used. Here, each VC has a flit depth of 8. The first stage of the TNoC router is tasked with IQ. The second stage consists of RC, VC allocation, and SA. The third stage does ST. TNoC was synthesized in the UMC 65nm technology library, and the cycle latency was found to be $5.26ns$. For deriving an asynchronous TNoC router, click controllers were used. The click controller has a latency of $0.26ns$ when synthesized on the same technology node. Further, the round-robin SA was replaced with a First Come First Serve asynchronous SA with a latency of $1.71ns$. To find the per stage latencies of the asynchronous TNoC, each stage was separately synthesized. For this asynchronous TNoC router, the per stage latencies were $2.58ns$, $4.95ns$, and $2.38ns$ respectively. For flow control, wormhole switching scheme was used, while the routing policy was XY.

NoC Variants - The following NoC variants with a size of 8x8 were constructed using the two TNoC routers, and are the baseline NoCs.

SyncMesh and *AsyncMesh* are NoCs in mesh topology with synchronous TNoC and asynchronous TNoC routers respectively. In *HetMesh* half of the routers are synchronous TNoC routers, while the rest are asynchronous TNoC routers. *SyncTorus*, *AsyncTorus*, and *HetTorus* are the variations of SyncMesh, AsyncMesh, and HetMesh with Torus topology, respectively. To show the effect of asynchronous arbitration, we derive ASyncMet20, AsyncMet40, and AsyncMet60 from

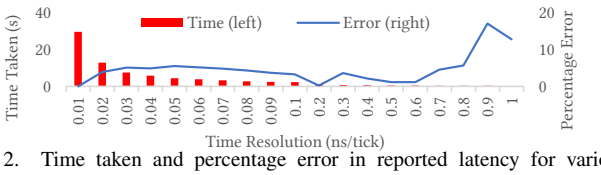

Fig. 2. Time taken and percentage error in reported latency for various resolution while simulating AsyncMesh with uniform traffic for 10000ns.

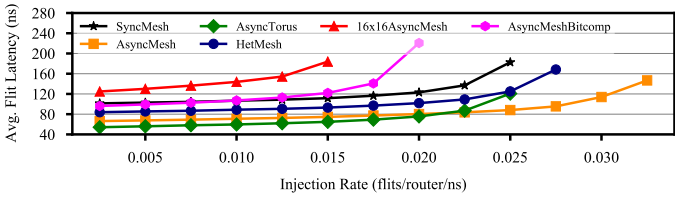

Fig. 3. Latency reported by ANSim when simulating various NoCs against different injection rates. Size of 16x16AsyncMesh = 16x16. AsyncMeshBitcomp has Bitcomp traffic

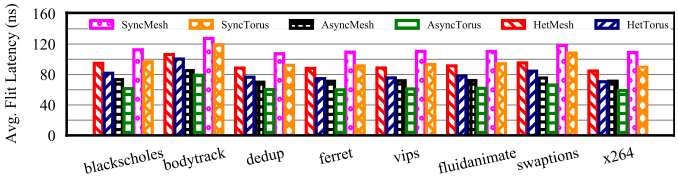

Fig. 4. End-to-end flit latency reported by ANSim for real-world application programs from the PARSEC benchmark.

AsyncMesh with a synchronizer latency of 20ns, 40ns, and 60ns respectively. The rest of the NoCs are derived from the three baseline cases, and the changes are mentioned in the required sections.

Simulation parameters - The time resolution of $0.2ns$ was chosen for simulations based on percentage error and time taken for simulation, as seen in Section V-1. For synthetic benchmarking, uniform traffic with packets of 1 flit size was injected at a rate of 0.005 flits/router/ns for $5 \times 10^5 ns$. To represent real-world application, Netrace [3] at 3Ghz was used. One billion cycles from each program were considered for evaluation.

V. Features and Use Cases of ANSim

1) Identification of simulation resolution: Larger tick granularity decreases the simulation time at the cost of simulation precision. The error will not be high for a tick granularity that completely divides the timing values of the simulated router. Identifying the sweet spot that gives a lower error and shorter simulation times is ideal for speeding up the design process. Fig. 2 shows that at a tick granularity of 0.2 ns, the error is near zero compared to a resolution of 0.001 ns while having significantly short simulation time.

2) NoC saturation point identification: After designing a router, it would be beneficial to know the no-load end-to-end flit latency and saturation point of NoCs made using this router. This would inform the designer regarding possible operating points of these NoCs for various traffic conditions. For example, in Fig. 3, it can be seen that the *AsyncTorus* has lower end-to-end flit latency compared to *AsyncMesh* between 0.0025 flits/router/ns and 0.02 flits/router/ns injection rates. The designer can conclude the following from this. 1) If the application's expected injection rate is higher than the saturation point of the NoC, then the NoC has to be

replaced - either with faster routers or a different topology. 2) If the expected injection rate is between 0.02 flits/router/ns and saturation point, it is better to use AsyncMesh. 3) If the traffic on the NoC is sparse and below 0.02 flits/router/ns injection rate, then it is better to use AsyncTorus among the three baseline NoCs. Further, it allows the designer to choose between two or more routers when implementing an NoC with the same topology. We believe that the ability to report system level, end-to-end flit latencies of a large variety of NoCs under various traffic patterns is the real strength of ANSim.

3) Real-world application benchmark: Even though synthetic benchmarks can roughly identify the operating region of an NoC, real-world traffic behaves very differently from the synthetic benchmarks [21]. Hence, it is necessary to narrow down the expected end-to-end flit latency for the desired application of the NoC. Fig. 4 shows that PARSEC applications prefer asynchronous NoCs over synchronous NoCs.

4) Modeling asynchronous arbitration: Fig. 5 and Fig. 6 reveal that the metastability of arbiters can cause a significant increase in flit latency at high injection rate, but with realistic applications like PARSEC [22] the effect of metastability is negligible for these NoCs. It might be possible that for a particular combination of asynchronous or heterogeneous NoCs, and desired NoC application, the effect of arbiter metastability is significant on the flit latency, and therefore needs to be found.

5) Modeling power and power gating: Modeling power is crucial to avoid over-provisioning of resources to improve timing performance. Fig. 7 shows the breakup of power used by the components in the NoC. From Fig. 7, it is observed that for Asynchronous TNoC, while running PARSEC programs, more than 90% of power is static power. This skew in power is due to the large buffer size, small frequency along with very sparse traffic. This presents the possibility of applying power-saving techniques like power gating. When employing an asynchronous variation of non-blocking power gating policy [6] on AsyncMesh with PARSEC traffic, the static power was reduced by 60% on average (Min 20%, Max 90%).

VI. Conclusion

In this paper, we have presented ANSim, an asynchronous NoC simulator, which can be used to model and analyze synchronous, asynchronous, and heterogeneous NoCs. ANSim has a large variety of NoC architectural configurations readily available for simulation. ANSim is significantly faster ($\sim 25\times$) than existing asynchronous NoC simulators. We demonstrate various ways in which the tool can help us characterize the workloads when run on an Asynchronous NoC, and how this can be used to improve energy efficiency using power gating. We show that for real benchmarks, where network traffic happens to be sparse, overall system performance does not degrade due to metastability, by modeling it in ANSim. The simulator is available at github.com/TomGlint/ANSim

Acknowledgement

The authors of this paper thank the authors of PANE [4] for providing the RTL source codes used in their work.

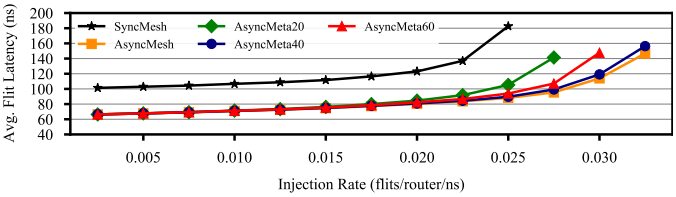

Fig. 5. End-to-end flit latency reported by ANSim for NoC's with arbitration metastability against baseline NoCs.

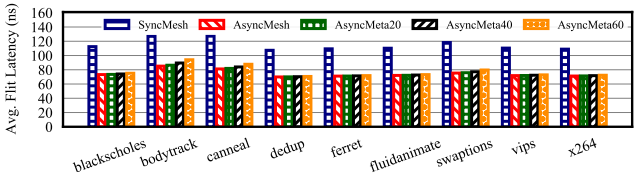

Fig. 6. End-to-end flit latency reported by ANSim for PARSEC program under NoC's with arbitration metastability against baseline NoCs.

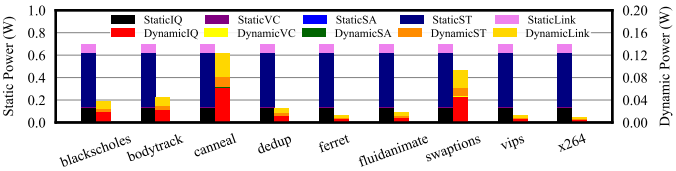

Fig. 7. Break up of static power (left) and dynamic power (right) reported by Orion 3 [15] integrated into ANSim for AsyncMesh for PARSEC programs.

References

[1] "Snapdragon 855 mobile platform," Nov 2019. [Online]. Available: https://www.qualcomm.com/products/snapdragon-855-mobile-platform

[2] W. J. Dally *et al.*, *Principles and practices of interconnection networks*. Elsevier, 2004.

[3] J. Hestness *et al.*, "Netrace: dependency-driven trace-based network-on-chip simulation," in *NoCArc '10*. ACM, 2010, pp. 31–36.

[4] S. N. Ved *et al.*, "Pane: Pluggable asynchronous network-on-chip simulator," *JETC*, vol. 15, no. 1, pp. 1–27, 2019.

[5] Y. Ben-Itzhak *et al.*, "Hnocs: modular open-source simulator for heterogeneous nocs," in *SAMOS*. IEEE, 2012, pp. 51–57.

[6] L. Chen *et al.*, "Power punch: Towards non-blocking power-gating of noc routers," in *HPCA*. IEEE, 2015, pp. 378–389.

[7] N. Binkert *et al.*, "The gem5 simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.

[8] N. Agarwal *et al.*, "Garnet: A detailed on-chip network model inside a full-system simulator," in *ISPASS*. IEEE, 2009, pp. 33–42.

[9] R. Ubal *et al.*, "Multi2sim: A simulation framework to evaluate multicore-multithreaded processors," in *SBAC-PAD'07*. IEEE, 2007, pp. 62–68.

[10] N. Jiang *et al.*, "A detailed and flexible cycle-accurate network-on-chip simulator," in *ISPASS*. IEEE, 2013, pp. 86–96.

[11] A. Varga, "Omnet++," in *Modeling and tools for network simulation*. Springer, 2010, pp. 35–59.

[12] E. Beigné *et al.*, "An asynchronous noc architecture providing low latency service and its multi-level design framework," in *ASYNC*. IEEE, 2005, pp. 54–63.

[13] N. Nasirian *et al.*, "Traffic-aware power-gating scheme for network-on-chip routers," in *DCAS*. IEEE, 2016, pp. 1–4.

[14] H. Matsutani *et al.*, "Ultra fine-grained run-time power gating of on-chip routers for cmps," in *NOCS*. IEEE, 2010, pp. 61–68.

[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] L. R. Marino, "General theory of metastable operation," *IEEE Transactions on Computers*, vol. 100, no. 2, pp. 107–115, 1981.

[17] T. Ishitani. [Online]. Available: github.com/taichi-ishitani/tnoc

[18] [Online]. Available: www.synopsys.com/verification/simulation/vcs.html

[19] [Online]. Available: www.mentor.com/products/fv/questa/

[20] A. Venkataraman *et al.*, "Evaluation of inter-process communication mechanisms," *Architecture*, vol. 86, p. 64, 2015.

[21] P. Gratz *et al.*, "Realistic workload characterization and analysis for networks-on-chip design," in *CMP-MSI*, 2010, pp. 1–10.

[22] C. Bienia *et al.*, "The parsec benchmark suite: Characterization and architectural implications," in *PACT*, 2008, pp. 72–81.