CANSim: When to Utilize Synchronous and Asynchronous Routers in Large and Complex NoCs

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

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

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

***Abstract*—Asynchronous routers offer benefits in latency, energy, and area over synchronous routers, but their large system design adoption is limited due to inadequate tool support for performance quantification. We introduce CANSim, a fast and accurate simulator for complex asynchronous and synchronous NoCs. Verified against synthesis models, CANSim models any graph-representable topology, asymmetric links, data-dependent delays, meta-stability, and supports synthetic and real-world traffic. It also accommodates power gating, hierarchical networks, and GALS-like networks. Our comparisons reveal that asynchronous NoCs provide lower packet latency at no-load conditions, but synchronous NoCs may outperform near saturation. On average, asynchronous NoCs offer up to 36% latency and 52% power benefits, with power-gating potentially reducing static power by 90%.**

***Index Terms*—Synchronous NoC, Asynchronous NoC, Simulator, Complex NoCs**

I. Introduction

Networks-on-chip (NoC) are a standard way of facilitating on-chip communication between numerous computational elements inside a chip. These NoCs are implemented with either synchronous routers or asynchronous routers [1]. There are four main advantages of using asynchronous routers over synchronous routers in an NoC. First, individual stages can operate at their speed in the communication pipeline leading to system-level performance benefits [2]. In contrast, in synchronous designs, all stages operate at the speed of the slowest stage. Second, they eliminate the need for global clock management across a large chip [1], [3]. Third, unlike worst-case clocked synchronous designs, flit types in an asynchronous NoC can have separate timing paths within the router [2]. Fourth, asynchronous routers have lower latency and area than a comparable synchronous router [2]. However, the adoption of asynchronous routers in system design has been lagging due to a lack of tool support to quantify system-level performance. This work aims to fill this void and provides a tool for comparing complex synchronous and asynchronous NoCs. Further, we identify various network conditions, based on topology and traffic pattern where asynchronous NoCs have lower latency than synchronous NoCs.

Existing comparisons between NoCs with state-of-the-art synchronous and asynchronous routers are limited to 4×4 2D mesh topologies and are obtained with the help of synthesis tools [2]–[4]. From the system design viewpoint, using synthesis tools to compare candidate designs is not practical due to two major limitations. First, the scalability of synthesis level models for simulation is limited [5]. Second, a system-level comparison of multiple NoCs needs to be obtained earlier in the design phase before dedicating resources to implementing candidate designs at the synthesis level. A fast and accurate approach is necessary to compare multiple complex synchronous and asynchronous designs.

This work is supported through grants received from Science and Engineering Research Board (SERB), Government of India, under SERB-SUPRA grant SPR/2020/000450, and Semiconductor Research Corporation (SRC) through contracts 2020-IR-3005 and 2020-IR-2980

Main Contributions: We propose a Complex Asynchronous NoC Simulator (CANSim) with the following main feature set and use it to quantify the performance of large and diverse NoCs with state-of-the-art synchronous and asynchronous routers. (i) CANSim supports event-level simulation of each router stage for accurate modeling of router operations. (ii) Each router in the NoC can have separate timing constraints. This feature allows the modeling of heterogeneous NoCs. (iii) Support for variability in the timing path allows for modeling data-dependent delays. (iv) Complex NoCs, including heterarchical NoCs, can be specified in the form of a graph. (v) CANSim can generate various synthetic traffic patterns and supports dependency-driven real-world benchmarks. (vi) Activity tracking to create power and power-gating reports. (vii) CANSim is verified against synthesis models at both no-load conditions and saturating conditions.

Key findings from the NoCs comparison include: (i) asynchronous NoCs have about 10% lower average packet latency at no-load conditions; (ii) they saturate 20% earlier under specific traffic patterns; (iii) they display consistently lower average packet latency across all injection rates for bitrev, shuffle, and transpose patterns; (iv) asynchronous NoCs mitigate the latency increase due to complex routing by about 23%; (v) under uniform traffic, both synchronous and asynchronous hierarchical NoCs show similar latencies; (vi) in GALS NoCs, asynchronous versions are 36% faster for uniform traffic; (vii) under PARSEC traffic, asynchronous NoCs are 8% faster and consume 52% less power; (viii) despite process variation, they maintain a 15% speed advantage; (ix) both NoC types reduce latency by 8% (synchronous) and 14% (asynchronous) respectively with PARSEC traffic; (x) asynchronous GALS NoCs are 20% faster than their synchronous counterparts under PARSEC traffic.

On average, asynchronous NoCs use less power and deliver packets faster than their synchronous counterpart, and the performance can be quantified with CANSim.

II. Background

NoCs are used to establish communication between different IPs in a system. Routers are linked together in various topologies using wires to construct an NoC. The following are essential aspects of synchronous and asynchronous that affect NoC behavior.

A router receives, buffers, and sends packetized data between input and output ports. The output port selection is based on destination and routing algorithm. Fig. 1 shows submodules: buffer stores data temporarily, Input Queuing (IQ) receives packets, Route Compute (RC) computes output port, Virtual Channels (VC) handle multiple packets, Virtual Channel Allocator (VCA) selects VC for transmission, Switch

Allocator (SA) allocates crossbar to selected VCs. Balanced pipeline stages are formed by fusing adjacent modules, triggered by a clock signal.

A. Synchronous vs. Asynchronous Router

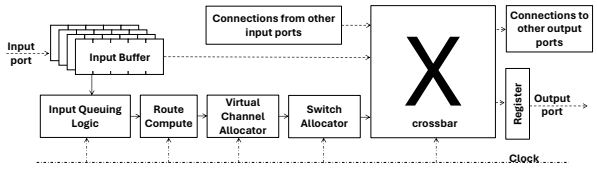


Fig. 1. Submodules of a synchronous router controlled by clock signal.

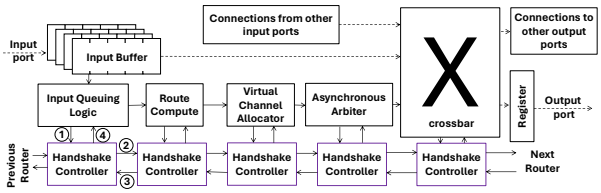


Fig. 2. Asynchronous router controlled by handshake controllers.

Fig. 2 depicts an asynchronous router with its submodules and control logic, resembling synchronous designs. The central component, the Asynchronous Arbiter (ASA), facilitates crossbar access on a First Come First Serve (FCFS) basis. The ASA's main function is to arbitrate requests arriving simultaneously to the same output port. To enable operation in the submodules, handshake controllers are employed in asynchronous routers. When a submodule completes processing and needs to transmit signals to the next one, it requests the next stage's availability through its Handshake Controller (HC). In Fig. 2, after input queuing is done, the IQ sends a request ① to its HC to proceed. The HC of IQ forwards this request ② to the VC's HC. If VC is currently idle, its HC acknowledges the request ③, which is then passed back to IQ's HC ④, indicating that IQ is available for processing the subsequent request. This implementation allows for delay-insensitive, clock-free operation in asynchronous routers. Additionally, when implementing an asynchronous router, adjacent submodules may be combined based on distinct timing paths, reducing the number of handshake controllers and forming the stages of the asynchronous pipeline.

B. Asynchronous communication and complex networks

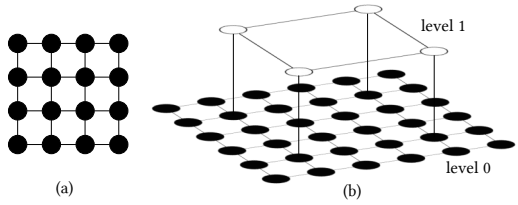


Fig. 3. (a) Simple Mesh Topology (b) A complex topology with two levels

Asynchronous communication between components requires handshaking protocols and data-encoding schemes due to the absence of a clock signal. In state-of-the-art asynchronous routers, transition signaling is used for handshaking, ensuring a single toggle of each control signal, resulting in only one round trip latency per transaction. For data encoding, a single-rail bundled-data transfer scheme is employed, where binary-encoded data is transmitted using additional request and acknowledgment wires alongside data wires [4].

Regarding complex networks, Fig. 3a illustrates a simple mesh network topology with routers arranged in a 2D array connected by links. However, for communication between numerous heterogeneous components, this topology may not scale efficiently. To address this, additional routers and links can be introduced to create alternate paths for data transmission. Fig. 3b demonstrates an example of a hierarchical network with two levels of routers. Comparisons between synchronous and asynchronous routers in such NoCs are yet to be explored due to the lack of tool support.

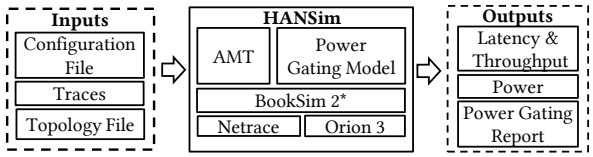
III. Simulation Framework

In this work, the goal of the simulation framework is to model the synchronous and asynchronous routers at the event level. This event-level simulation helps to model the functional and timing aspects of the NoC closely. Further, the topology of the NoC can be specified as a directed graph which allows for flexibility in the NoC definition.

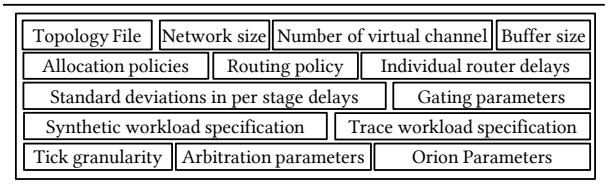
A. Simulation precision and speed

Synchronous NoCs operate based on clock edges, measuring delays in clock cycles. In contrast, asynchronous NoCs use the wall clock, measured in multiples of the smallest unit of time known as a *tick* (e.g., ps or ns). In CANSim, event queues are checked every *tick*, and matured events are executed while new events are added based on the current events. The simulation precision can be adjusted by using a smaller *tick* size, like 0.001 ns instead of 0.1 ns, at the cost of longer simulation time. Running a simulation for 1 million ticks takes approximately 8 seconds on a modern computer, and batch processing can leverage parallel processing within the same executable. All timing parameters in the configuration file are specified in ticks.

B. High level design of CANSim



(a) High level design of CANSim



(b) Configuration File

Fig. 4. Modules of CANSim along with configuration file.

Fig. 4a illustrates the modules involved in CANSim's operation. The simulation starts by ingesting a configuration file (shown in Fig. 4b) containing parameters described in detail in [6]. CANSim then creates the network topology based on the specified *Topology file* using BookSim's [7] network topology model for typical mesh NoCs. Traffic can be synthetic or traced, injected into the network either from the configuration file or via Netrace [8] module. The Asynchronous Model & Timing (AMT) module initializes the network and creates router models. Orion 3 [9] module reads the technology file, monitoring router configuration and activity during the simulation. For synchronous NoCs, Orion 3 directly reports static and dynamic energy components. For asynchronous NoCs, the same module generates activity reports used to determine

energy consumption. The power-gating module has visibility into each router, allowing it to set routers' states (active, idle, or gated) during runtime according to the configuration file's parameters. At the end of the simulation, CANSim reports packet and flit latency and throughput, generates power and area reports, and provides power-gating reports, including router gating duration and times based on the break-even time.

C. Asynchronous router in CANSim

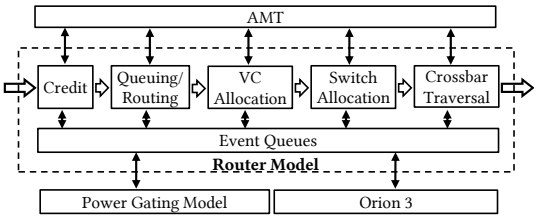


Fig. 5. Interaction of router model with other modules in CANSim

Fig. 5 depicts a single router within CANSim and its interactions with other models. The hollow arrows in Fig. 5b represent the flow of flit data. In CANSim, each router in the NoC can be individually addressed and set to either synchronous or asynchronous operation modes. Based on the configuration file, CANSim models events within the router as described in Section II. These submodules are organized into different pipeline stages according to the router's timing, enabling the modeling of routers with various pipeline stages while preserving all functional aspects.

D. Additonal features of asynchronous router model

1) Power-gating

CANSim implements a non-blocking power-gating model [10] for routers in NoCs. This model wakes up routers within two hops of the flit path if the router holding the flit is already gated, based on the routing function through the intermediate router. The wake-up process occurs before the flit reaches the gated router. The break-even time, calculated to justify the energy overhead of gating, is a parameter in the base configuration file. CANSim calculates the total gated duration per router, considering the break-even time, and derives the energy overhead due to gating based on the number of times the router was gated.

2) Asynchronous Arbiter

In NoCs with asynchronous routers, the crossbar forwards input to the output on a first-come-first-served basis. The Asynchronous Arbiter (ASA) allocates the crossbar when two requests are present. Simultaneous arrival of two requests at ASA can lead to metastability, where the decision process for servicing the requests takes an arbitrary amount of time. The time to resolve metastability is given by the formula $\tau \times \ln(\frac{T_w}{\delta})$, where τ and T_w depend on the circuit implementation [11]. CANSim includes a model to account for this metastability but focuses on the discussed model in this work.

IV. Verification

To ensure the trustworthiness of simulator models, verification of functionality is crucial. CANSim's accuracy is confirmed by matching its output with synthesis models at all injection ranges. The study compares a 4×4 2D mesh network using CANSim against the Transition-Signaling Bundled-Data Lightweight Asynchronous router (TaBuLA) and its synchronous counterpart [2]. The routers' details are given in Table I. To verify CANSim, synchronous and asynchronous 4×4 2D mesh NoCs are constructed and simulated. The results are compared against synthesis results for injection rates ranging from 100 to 700 Mfilt/Port/sec, with a packet size of 3 flits. Fig. 6 demonstrates that the deviation of Async and Sync NoCs modeled by CANSim, compared to AsyncRef and SyncRef from [2], varies by less than ∼4% across all ranges, from no-load conditions to saturation points. These results indicate that CANSim accurately models both synchronous and asynchronous routers, ensuring precision and reliability.

TABLE I
SYNTHESIS DATA FOR STATE-OF-THE-ART ASYNCHRONOUS AND SYNCHRONOUS ROUTER

	TaBuLA (Async)	xpipes (Sync)
Head Latency (ps)	1165	943
Payload Latency (ps)	486	943
Link Latency (ps) (for 1mm)	414	943
Area	12433	14266
Energy per flit (pJ)	3.88	4.69

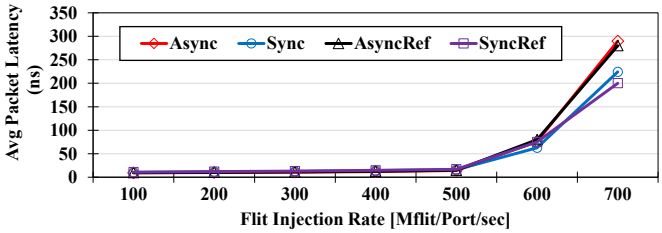


Fig. 6. Comparison of packet latency reported by CANSim (Async and Sync) and synthesis results (ASyncRef and SyncRef)

V. Experimental Setup

CANSim allows for a vast range of comparisons between synchronous and asynchronous NoCs. This paper limits the discussion to the following NoC configurations and traffic. We use the state-of-the-art asynchronous router TaBuLA and its synchronous counterpart xpipes to represent asynchronous and synchronous routers, respectively. For more details on the routers, refer Section IV and [2]. The default NoC size in this work is 8×8 with Mesh topology. The routing algorithm used is DOR XY. The packet size is three flits, while the buffer is four flits deep. The routers use a wormhole switching scheme, and the packets are injected at a rate of 100MFlits/port/sec to 1000MFlits/port/router.

VI. Results

The results in this section are organized based on topology. Section VI-A shows the comparison for planar topologies while Section VI-B and Section VI-C show results for large and complex NoCs - Hierarchical NoCs and GALS NoCs, respectively.

A. *Planar NoC*

1) Synthetic Traffic

In the development of a general-purpose 8x8 NoC, we have utilized synthetic benchmarks with seven diverse traffic patterns—bitcomp, neighbor, tornado, uniform, bitrev, shuffle, and transpose (Fig. 7 and Fig. 8) to test its capabilities. In patterns like bitcomp, neighbor, and tornado, asynchronous NoC outperforms synchronous NoC in terms of no-load latency, registering about 5-10% faster speeds. However, it saturates earlier due to increased router latency or switch contention. The uniform traffic pattern, which ensures a balanced load, shows asynchronous NoC with approximately 7%

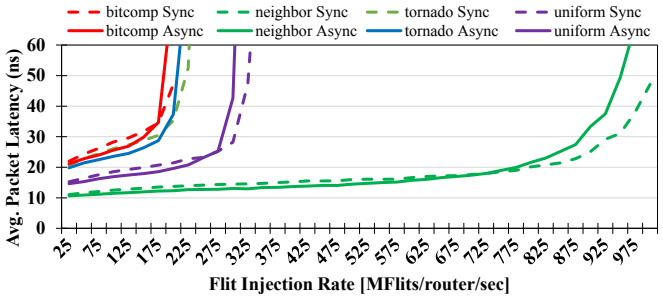


Fig. 7. Operational region for bitcomp, neighbor, tornado and uniform traffic.

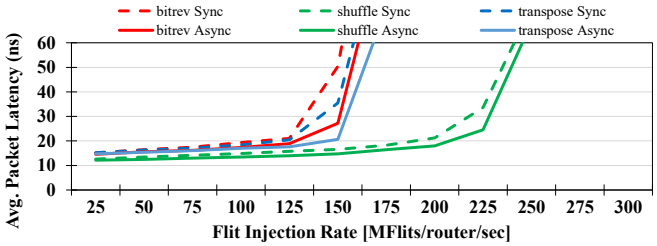


Fig. 8. Operational region for bitrev, shuffle and transpose traffic.

lower latency, but it also hits the saturation point 5% earlier than synchronous NoC. In bitrev traffic, asynchronous NoC excels, with a 5% faster no-load latency and 10% higher saturation injection rate due to reduced switch contention. Similarly, in the shuffle pattern, asynchronous NoC is around 7% faster, maintaining packet latency near saturation due to its directional data flow. Transpose traffic, which causes a network bisection bottleneck, surprisingly benefits asynchronous NoC. Despite an imbalanced workload, it achieves an impressive 35% higher saturation injection rate, courtesy of lesser switch contention. To sum up, asynchronous NoCs surpass synchronous NoCs in no-load latency across all patterns and exhibit higher saturation injection rates for bitrev and transpose patterns, even under unbalanced workloads.

2) Routing Policy and Packet Size

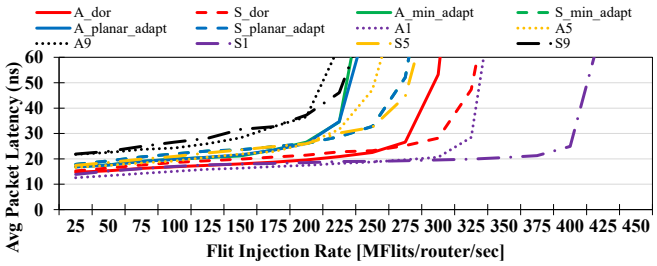


Fig. 9. Latency curve for synchronous(S) and asynchronous(A) NoCs with different routing algorithms and packet size for uniform traffic

Routing algorithms can range from simple static policies like DOR XY to complex adaptive ones like planar_adapt, and their hardware implementation creates different timing paths [6]. As shown in Fig. 9, with asynchronous routers, the added delay only affects the head flit, reducing the latency increase due to adaptive routing by about 23% compared to synchronous routers, where changes in routing complexity directly impact the cycle time.

Our simulator's capability to support various packet sizes provides further insights. Notably, the head flit latency and body flit latency in an asynchronous router differ, unlike in state-of-the-art synchronous routers. Consequently, the end-to-end latency varies with the number of flits per packet, as does the saturation injection rate. For instance, this rate decreases by 50% for nine flit packets compared to one flit packet, and the latency gap between synchronous and asynchronous routers narrows as the number of flits per packet increases.

3) PARSEC Traffic

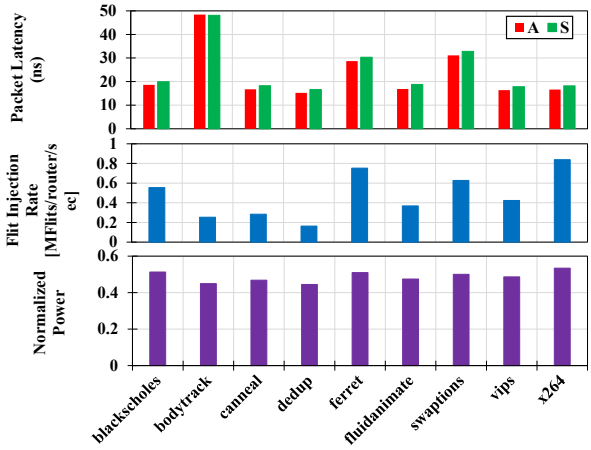


Fig. 10. Top: Average packet latency for PARSEC applications in asynchronous (A) and synchronous (S) NoCs. Middle: Average flit injection rate of PARSEC applications. Bottom: Relative power of asynchronous NoC compared to synchronous NoC for PARSEC applications.

Synthetic benchmarks suggest that asynchronous NoCs perform better than synchronous ones in specific scenarios based on traffic patterns and injection rates, prompting an investigation of their performance with real-world traffic. Using netrace to inject packets from PARSEC applications, we see in Fig. 10 that asynchronous NoC's latency is about 8% lower than synchronous NoC's. Notably, some applications, like *bodytrack*, show higher average packet latency than synthetic traffic patterns despite operating near no-load conditions, underscoring the utility of CANSim for such analysis. Also evident is that the asynchronous NoC uses around 52% less power, primarily due to clockless operation, positioning it as an advantageous choice for real-world traffic handling.

4) PARSEC Gating

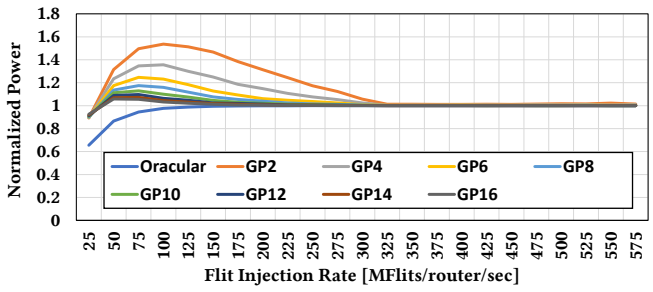


Fig. 11. Normalized power of NoCs with Oracular and basic power-gating policy(GP) against asynchronous NoC without power-gating

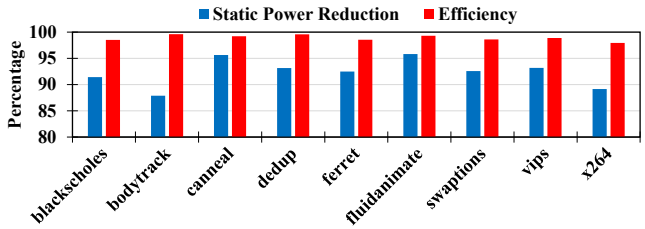


Fig. 12. Percentage power reduction in static power from power-gating asynchronous NoC for PARSEC applications and relative efficiency of the basic gating policy against Oracular policy

Exploring power-saving methods like power-gating in asynchronous NoCs is valuable, despite their clockless operation and dynamic power dominance when fully utilized [2], [12]. Under PARSEC applications, approximately 60% of power is static due to low injection rates, creating a scope for power conservation. Using the 'oracular' power-gating technique, CANSim helps pinpoint times for potential energy saving, when routers aren't routing packets for longer than the break-even threshold. We also evaluate a 'basic' power-gating policy (GP), power-gating the router after a period of inactivity, with the gating-threshold tailored to the application-NoC pair and a 16 ns break-even time [12]. As shown in Fig. 11, power-gating can result in around 60% power reduction for low-traffic asynchronous NoCs, suggesting a benefit in applying these techniques. However, this must be done cautiously as basic policy (GP) might increase power consumption at high injection rates. In Fig. 12, a 'basic' power-gating policy (GP4) with a 4ns gating-threshold demonstrates around 90% static power savings due to low injection rates while maintaining nearly 95% power-saving efficiency compared to the oracular policy.

5) Process Variation

CANSim can simulate delay changes caused by process variation, common in chip fabrication. Asynchronous NoCs better handle these variations, as they don't need to match the slowest speed like synchronous NoCs. Fig. 13 represents such NoCs, each quadrant with varying subcomponent latencies due to process variation. Quadrants of 8x8 asynchronous NoCs (A1, A2, A3, A4) illustrate how routers in fast quadrants operate 500ps faster, while those in slow quadrants are 500ps slower. A synchronous NoC (S) operates 500ps slower overall due to a shared clock. Studying these variations, as shown in Fig. 13 for PARSEC programs, reveals their impact on application performance. Asynchronous NoCs, despite being affected by process variation, are about 15% faster than their synchronous counterparts.

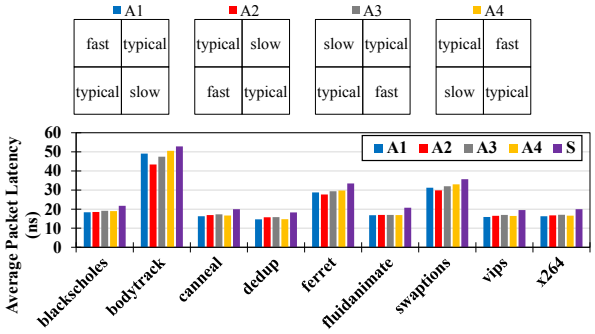


Fig. 13. End to end flit latency reported by HANSim for NoC Variations of AsyncMesh with each quadrant consisting of 4×4 routers. Each quadrant has different subcomponent latencies.

B. Hierarchical NoC

As NoC scales, the added links and routers can complicate the topology, increasing latency in asynchronous routers and reducing the latency advantage of clockless operation with longer link length [4], [13]–[16]. Fig. 14 shows a hierarchical NoC to illustrate this, with a 15x15 mesh at *level 0* and a 3x3 mesh at *level 1*, effectively reducing the network diameter from 28 to 14 for a minimal area overhead. For larger NoCs, traffic pattern typically falls between uniform and nonuniform [14]. Fig. 14 compares the latency of planar and

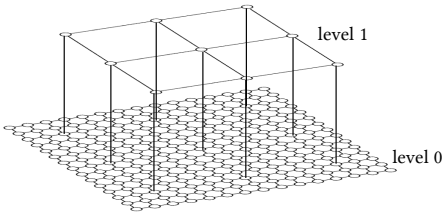


Fig. 14. Hierarchical NoC with two levels

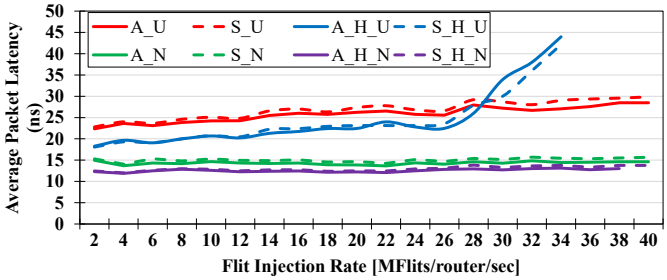


Fig. 15. Latency curve comparing hierarchical and planar NoCs with synchronous and asynchronous routers

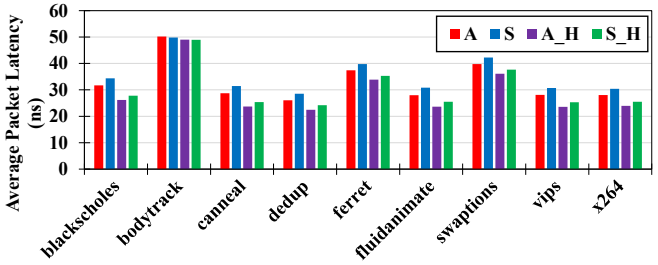


Fig. 16. Comparison of latency for PARSEC applications on planar and hierarchical NoCs

hierarchical NoCs, synchronous (S/S_H) and asynchronous (A/A_H), under uniform (_U) and nonuniform (_N) traffic. Hierarchical topology reduces packet latency by around 14% for nonuniform traffic, and 27% for uniform traffic at no-load injection rate. However, latency increases beyond planar NoCs at around 30 MFlits/port/sec due to link saturation. Both synchronous and asynchronous hierarchical NoCs exhibit similar average flit latencies, suggesting the best topology and router choice depends on traffic patterns and injection rates (Fig. 15). Real-world traffic comparison (Fig. 16) shows asynchronous hierarchical NoCs having the least latency, though the gap with synchronous hierarchical NoC is just about 4%. Hierarchical NoCs generally reduce latency by around 15% compared to planar NoCs.

C. GALS NoC

Globally Asynchronous, Locally Synchronous (GALS) architectures have garnered attention for integrating numerous computational units while decoupling their operations [2]. GALS systems feature islands of synchronicity, but the islands operate asynchronously [1], [2], [17]. Fig. 17 depicts a GALS-based hierarchical NoC, with 5x5 islands operating at half the speed of asynchronous (A_GALS) and synchronous (S_GALS) routers. In A_GALS, clockless operation obviates the need for equalizing performance across pipelines, unlike S_GALS that requires inter-domain synchronization, adding latency [1], [18]–[21].

Fig 11 compares A_GALs and S_GALS under uniform and nonuniform traffic, showing S_GALS performance is around

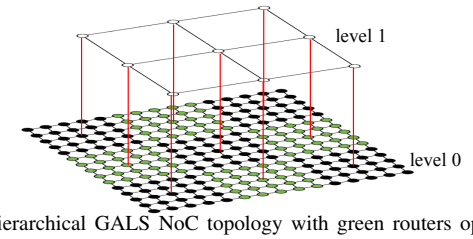


Fig. 17. Hierarchical GALS NoC topology with green routers operating at half the rated speed

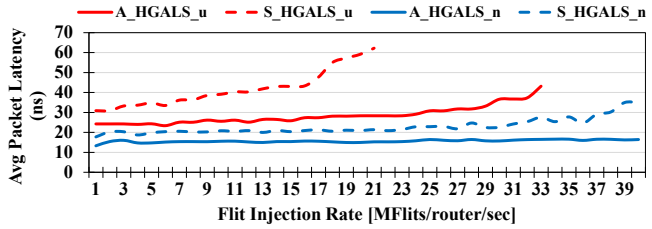


Fig. 18. Latency curve for synchronous and asynchronous hierarchical GALS NoC

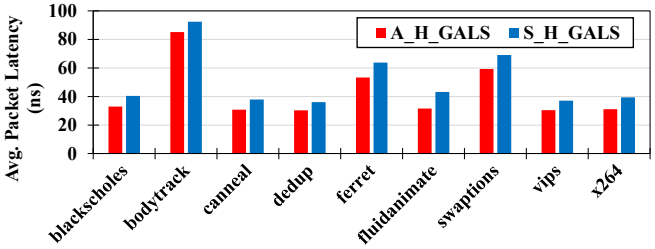


Fig. 19. Packet latency for PARSEC application in synchronous and asynchronous hierarchical GALS NoC

30-36% lower due to synchronization. Similarly, under PARSEC traffic, A_GALS is roughly 20% faster than S_GALS on average, though the benefits vary by application (Fig. 19), highlighting the need for a tool like CANSim.

VII. Related Works

In heterogeneous NoCs simulation, HNOCS [22], the first asynchronous simulator, has significant limitations, including inability to simulate real traffic and model data-dependent router delays. PANE [5] addressed these issues but is impractically slow due to its use of Inter-Process Communication (IPC) [23]. ANSim [24] improves upon PANE but can't simulate complex networks. In the hierarchical NoC domain, research quantified the benefits of synchronous hierarchical NoC for chip multiprocessors [14], [15]. For GALS systems using asynchronous routers, automated synthesis tool flow was presented [2], and a non-blocking power-gating mechanism was proposed for synchronous routers [10]. Until now, evaluating mixed synchronous-asynchronous networks was hampered by a lack of tool support. Our work enables such evaluation and informs essential design decisions.

VIII. Conclusion

Adopting asynchronous routers in large system design has been lagging due to the inability to quantify the performance benefits compared to a synchronous router at the early design stage. We present CANSim, a fast and accurate tool for comparing complex synchronous and asynchronous NoCs with numerous routers to fill this void. CANSim is verified against synthesis models at both the network's no-load and saturating conditions. CANSim supports both synthetic and real-world benchmarks. With CANSim, we compare various large and complex NoCs with state-of-the-art synchronous and asynchronous routers and identify regions of operation where synchronous NoCs and asynchronous NoCs provide lower latency. We observe that using asynchronous routers in NoCs can have latency benefits up to 36% and power benefits up to 52% compared to similar synchronous NoC designs. The CANSim simulator used in this work is available at https://github.com/TomGlint/CANSim

References

[1] A. Ghiribaldi *et al.*, "A transition-signaling bundled data noc switch architecture for cost-effective gals multicore systems," in *DATE*. EDA Consortium, 2013, pp. 332–337.

[2] D. Bertozzi *et al.*, "Cost-effective and flexible asynchronous interconnect technology for gals systems," *IEEE Micro*, 2020.

[3] M. Imai *et al.*, "The synchronous vs. asynchronous noc routers: an apple-to-apple comparison between synchronous and transition signaling asynchronous designs," in *NOCS*. IEEE, 2016, pp. 1–8.

[4] W. Jiang *et al.*, "An asynchronous noc router in a 14nm finfet library: comparison to an industrial synchronous counterpart," in *DATE*. IEEE, 2017, pp. 732–733.

[5] S. N. Ved *et al.*, "Pane: Pluggable asynchronous network-on-chip simulator," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 15, no. 1, pp. 1–27, 2019.

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

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

[8] J. Hestness *et al.*, "Netrace: dependency-driven trace-based network-on-chip simulation," in *Proceedings of the Third International Workshop on Network on Chip Architectures*. ACM, 2010, pp. 31–36.

[9] 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.

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

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

[12] N. Nasirian *et al.*, "Traffic-aware power-gating scheme for network-on-chip routers," in *2016 IEEE Dallas Circuits and Systems Conference (DCAS)*. IEEE, 2016, pp. 1–4.

[13] D. Matos *et al.*, "Floorplan-aware hierarchical noc topology with gals interfaces," in *2012 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2012, pp. 652–655.

[14] R. Das *et al.*, "Design and evaluation of a hierarchical on-chip interconnect for next-generation cmps," in *2009 IEEE 15th International Symposium on High Performance Computer Architecture*. Ieee, 2009, pp. 175–186.

[15] J. Yin *et al.*, "Modular routing design for chiplet-based systems," in *ISCA*. IEEE, 2018, pp. 726–738.

[16] C. J. Alpert *et al.*, "Buffer insertion with accurate gate and interconnect delay computation," in *DAC*, 1999, pp. 479–484.

[17] M. Krstic *et al.*, "Globally asynchronous, locally synchronous circuits: Overview and outlook," *IEEE Design & Test of computers*, vol. 24, no. 5, pp. 430–441, 2007.

[18] R. Dobkin *et al.*, "Data synchronization issues in gals socs," in *10th International Symposium on Asynchronous Circuits and Systems, 2004. Proceedings.*, 2004, pp. 170–179.

[19] F. Burns *et al.*, "Gals synthesis and verification for xmas models," in *DATE*, 2015, pp. 1419–1424.

[20] I. M. Panades *et al.*, "Bi-synchronous fifo for synchronous circuit communication well suited for network-on-chip in gals architectures," in *NOCS'07*. IEEE, 2007, pp. 83–94.

[21] D. Ludovici *et al.*, "Mesochronous noc technology for power-efficient gals mpsocs," in *Proceedings of the Fifth International Workshop on Interconnection Network Architecture: On-Chip, Multi-Chip*, 2011, pp. 27–30.

[22] Y. Ben-Itzhak *et al.*, "Hnocs: modular open-source simulator for heterogeneous nocs," in *2012 international conference on embedded computer systems (SAMOS)*. IEEE, 2012, pp. 51–57.

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

[24] T. Glint *et al.*, "Ansim: A fast and versatile asynchronous network-on-chip simulator," in *2020 IEEE 38th International Conference on Computer Design (ICCD)*, 2020, pp. 619–622.