University of Illinois at Urbana-Champaign

CS598PS Machine Learning for Signal Processing - Project Report

Detection of Wireless Traffic Bandwidth for Automatic Communication Protocol Selection in Wireless Network-on-Chip Systems

Vimuth Fernando, Antonio Franques, Keyur Joshi
{wvf2, franque2, kpjoshi2}@illinois.edu

January 11, 2020

Contents

1 Introduction

Recent computer architecture trends herald the arrival of massive multiprocessors with more than a thousand cores within a single chip. In this context, as parallel programs continue to increase the amount of data sharing and signaling between cores, on-chip communication becomes a critical issue. Unfortunately, traditional on-chip networks have been proven to not scale well in terms of latency or energy consumption, slowing down the computation in multicore processors. The Wireless Network-on-Chip (WNoC) paradigm holds considerable promise for the implementation of on-chip networks that will enable such massive multicore chips.

One proposed method of implementing a WNoC is to augment each core on a chip with a transceiver and an antenna that uses a wireless channel to send and receive messages across the chip. A major challenge for this technique resides in the design of methods that provide fast and efficient access to the wireless channel while adapting to the constant traffic changes within and across applications. Existing approaches are either cumbersome or do not provide the required adaptivity.

To fully realize the tremendous potential of WNoC it is necessary to build Medium Access Control (MAC) mechanisms able to cope with the heterogeneity and stringent performance requirements of the multiprocessor scenario [1]. The MAC protocol needs to adapt to wild changes in the generated traffic, which can follow different patterns, while incurring little to no delay on the transmission.

A key insight towards that goal is that applications tend to have certain "behaviors" that can be exploited to boost the efficiency of the system. These behaviors can be used to classify the runtime of the program into *communication*, or *computation* phases. The *communication* phases consist of intensive usage of the network, with many nodes attempting to transmit at the same time. On the other hand, *computation* phases consist predominantly of local computations, and therefore show sparse traffic in the network. Different MAC protocols can be used to ensure optimal network usage in the two phases.

In this project we present a new algorithm, capable of dynamically adapting the MAC protocol to the characteristics and demands of the WNoC for every application, minimizing the transmission latency of all nodes, and increasing the overall throughput of the chip. We achieve that goal by combining the benefits of collision-free (Token-Ring) and contention-based (BRS [2]) protocols, and automatically detecting the phases in which it is best to use one over the other.

The rest of the report is organized as follows: Section 2 provides a brief background on the topic in which our classification algorithm will be applied. In Section 3 we perform a study on a set of multi-threaded representative applications. Section 4 explains the algorithm capable of detecting and predicting the different traffic phases, and Section 5 presents the experimental results. Finally, Section 6 concludes the report with a summary and some final remarks.

2 Background

2.1 Wireless Network-on-Chip

Wireless technology can enable us to reduce chip-wide latency to a few processor clock cycles, regardless of the size of the chip or the number of cores. To that end, one antenna and transceiver is co-integrated with each core or small cluster of cores, as shown in Fig. 1. Information coming from the cores is modulated and the resulting signals propagate through the chip package, bouncing off the metallic heat sink and reaching the receivers. The relatively high losses of the bulk silicon prevent the enclosed package from acting as a reverberation chamber [3].

Antennas are either variants of planar integrated dipoles [4, 5] or vertical monopoles implemented with vias that are drilled through the silicon die [6, 7]. In both cases, the operating frequency is chosen within the mm-wave and sub-THz spectrum to minimize the antenna size and avoid undesired near-field coupling. Further, simple modulations such as On-Off Keying (OOK) are often considered to avoid power-hungry components such as Phase-Locked Loops (PLLs) in the pathway to minimizing power consumption (towards 1 pJ/bit/cm) and chip area (towards 0.1 mm^2 per transceiver) while maintaining relatively high speeds (10+ Gb/s) [8, 9, 10, 11, 12]. The link budget is performed generally assuming a target Bit Error Rate (BER) below 10^{-12}, seeking to be as reliable as regular on-chip wires.

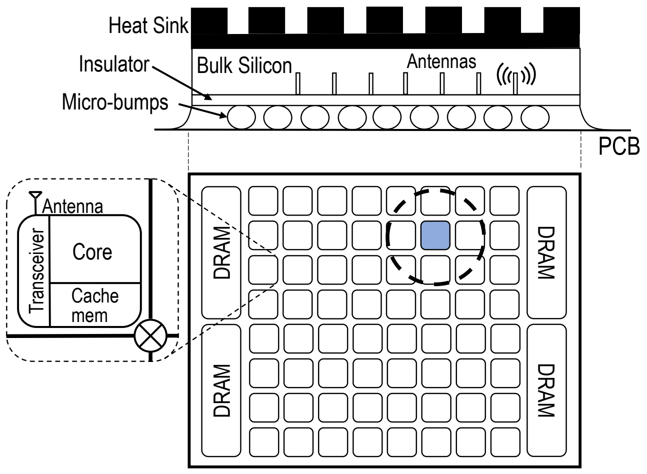


Figure 1: Schematic diagram of a Wireless Network-on-Chip (WNoC) within a conventional flip-chip package with one vertical monopole and transceiver per core.

2.2 MAC Background

Our prediction model switches between two previously existing MAC protocols, Token-Ring and BRS-MAC [2]. We provide a brief background on each of them.

Token-Ring: takes its name from the fact that implements a virtual ring, as shown in Figure 2. In this ring, only the node possessing the token is able to transmit.The MAC module of each node is connected to the ring through a register controlled by a *seize* bit, which is enabled when the node wants to transmit. The token is passed in either clock-wise or counter-clock-wise up to the first node that has its seize bit set. In wireless, the token is implicitly passed at the end

of a successful packet transmission, or whenever a slot goes unused.
BRS-MAC: a contention-based non-persistent MAC protocol based on collision avoidance. Similarly as in CSMA/CA, when a node is ready to send data, it listens to the medium and only transmits if the medium is idle. Otherwise, it waits during a backoff period and checks again whether the medium is idle. However, instead of transmitting the whole packet at once and then waiting for either a timeout (in case of a collision or a wrong checksum) or an acknowledgment (ACK), as in CSMA/CA, here only the preamble of the packet is sent at first, and is then followed by a short listening time, such that in case of a collision, the detecting nodes send a negative acknowledgment (NACK) that will prevent the transmitter from transferring the rest of the packet, and forcing it to enter into a backoff period. If no NACK is received during the listening time, the transmitter proceeds with the rest of the packet, assuming that all other nodes are aware of the transmission, and therefore they will remain silent. Notice that by using this intermission mechanism, BRS-MAC does not require the use of ACKs, which makes it very suitable for broadcast-oriented scenarios.

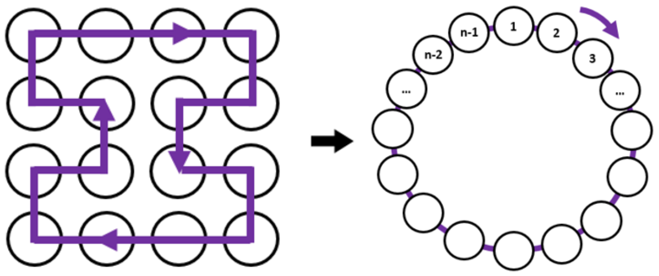


Figure 2: Physical distribution of the nodes (left) together with a possible virtual token ring configuration (right)

3 Input Data

In order to obtain our data we used Multi2Sim [13], which is a cycle-accurate architecture and microarchitecture simulator that is able to model a full processor and its cache hierarchy. The system configuration used for our measurements is presented in Table 1.

Table 1: Architecture modeled. RT means round trip.

General Parameters	
Chip	64 cores at 22nm technology
Core	Out of order, 2-issue wide, 1GHz, x86 ISA
ROB; ld/st queue	64 entries; 20 entries
L1 I+D caches	Private 32KB WB, 2-way, 2-cycle RT, 64B lines
L2 cache	Shared with per-core 512KB WB banks
L2 bank	8-way, 6-cycle RT (local), 64B lines
Cache coherence	MOESI directory based
On-chip network	2D-mesh, 4 (default), 2 or 1 cycles/hop, 128-bit links
Off-chip memory	Connected to 4 mem controllers, 110-cycle RT
Wireless Network Parameters	
Per-core wireless memory	512KB, 6-cycle RT, 64-bit wide line
Wireless channel	20Gb/s; 1 cycle for collision detection
Baseline MAC protocols	BRS (exponential backoff), token passing in ring

The workload used to generate the features was composed of 7 representative benchmarks from the SPLASH-2 and Crono suites. These benchmarks, listed in Table 2, were handpicked to include a wide range of different scenarios, in order to make our predictor more robust against unseen applications.

Table 2: Summary of the applications.

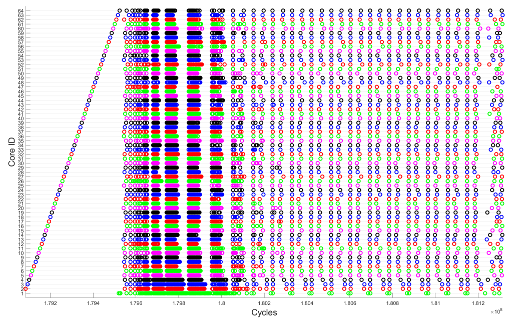
Name	**Description**	**Input**
Water [14]	Water molecules simulation (nsquared)	1000 molecules, 10 steps
BFS [15]	Breadth-first search	p2p-gnutella31 (from [16])
Pagerank [15]	Compute pagerank for nodes in a graph	p2p-gnutella31 (from [16])
CC [15]	Find connected components of a graph	p2p-gnutella31 (from [16])
Community [15]	Compute modularity of a graph	p2p-gnutella31 (from [16])
Ocean [14]	Large-scale ocean movements simulation	258×258 grid
FFT [14]	Fast Fourier Transform	2^{16} complex data points

In order to capture fine-grained behavior of the system we modified Multi2sim and added a function that automatically samples and saves into a file the occurrences for the following 29 events at every 10 microsecond (10,000 cycle) interval, referred to as a "window". **The concatenation of these events for the current and previous windows are the actual features we use for our model:**

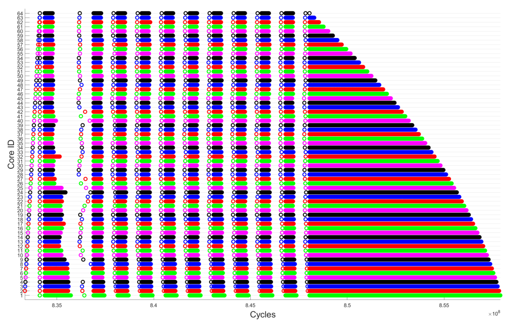
- L1 instruction cache: total number of accesses, read hits, read misses
- L1 data cache: total number of accesses, read hits, read misses, write hits, write misses

- L2 cache: total number of accesses, read hits, read misses, write hits, write misses
- Main memory: total number of accesses, reads, writes
- Wireless memory: total number of accesses, reads, writes
- Wired network-on-chip: number of transfers
- Wireless network-on-chip: number of transfers, collisions
- Functional units: number of accesses to the integer adder, integer multiplier, integer divider, effective address calculator, etc.

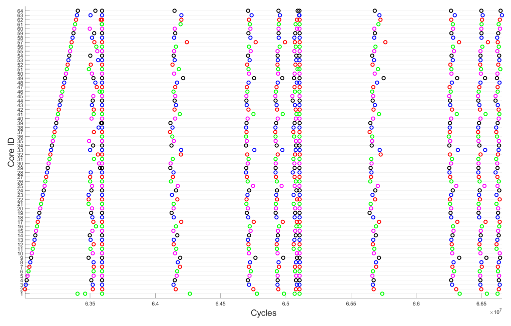
Figure 3 presents a subset of the features gathered from the processor and the memory system. Specifically, it shows the write occurrences to the wireless memory. The X-Axes show the cycle at which the write occurred, while the Y-Axes show the core that performed the write. Each of these writes end up producing a packet that will be sent through the wireless network-on-chip. As we can see in Figure 3, on-chip traffic is highly heterogeneous, meaning that the load (packet injection rate), burstiness (amount of packets sent in a row) and hotspotness (spatial distribution of the packets) of the network vary across applications, and even across the different phases of a single application. Detecting these patterns and organizing the accesses to the medium as efficiently as possible is crucial for the overall performance of the application, as inefficient wireless medium use can lead to excessive application latency.



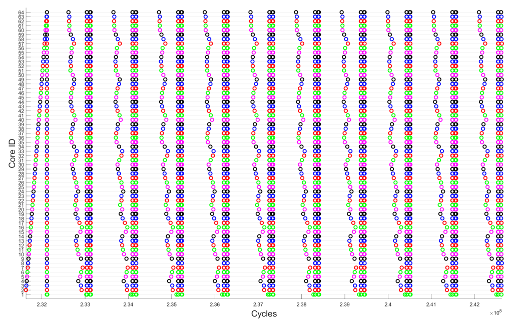
(a) BFS



(b) Community



(c) FFT



(d) Pagerank

Figure 3: Packet injection traces for several SPLASH-2 benchmarks with 64 cores, showing intra- and inter-application traffic differences across the execution.

4 Methodology

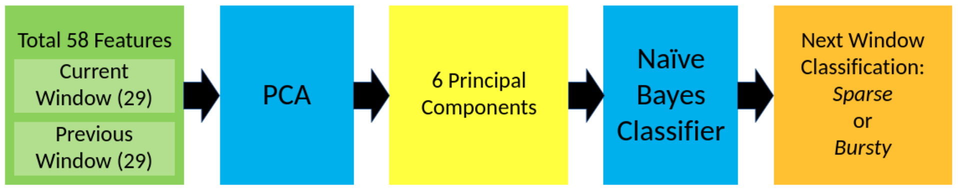


Figure 4: Block Diagram of our Classifier

We now present the methodology used to model our Medium Access Control (MAC) protocol predictor. This includes the separation of all the gathered data into the training and testing sets, labeling of the training data, dimensionality reduction of our features, and classification. Figure 4 shows a block diagram summarizing our classification approach.

4.1 Nature of Data

Our combined training / testing dataset consists of a set of feature vectors with the corresponding labels. Each feature vector is a concatenation of the 29 features of the *current* and the *previous* window, producing a total of 58 features. We found that adding additional previous windows did not cause any significant increase in accuracy. The corresponding label is the ideal MAC protocol for the *next* window.

4.2 Labeling the Data

To label our training / testing feature vectors, we counted the number of collisions that occurred in each window. We set a threshold of 250 collisions for using BRS. If the number of collisions is below this threshold, then there is not too much contention for the wireless bandwidth, and the ideal MAC protocol for this window is BRS. If the number of collisions is above this threshold, then then the Token-Ring protocol should be used instead to avoid excessive latency. We experimented with other thresholds, such as 100 collisions, and found this threshold to be a good trade-off point between the two protocols.

4.3 Splitting Data into Training / Testing Sets

From our set of feature vectors and their corresponding labels, we randomly picked 20% to be the testing dataset, and used the remaining 80% as a training dataset. We also experimented with different random seeds to create different training / testing splits for validation of our results across multiple testing sets.

4.4 Dimensionality Reduction

As an initial step, we used PCA to reduce the number of feature dimensions from 58 to 6. We experimented with different numbers of principal components and settled at 6 as that was the point where the accuracy increase as a result of adding more dimensions became insignificant. We used our own implementation of PCA – the implementation we used for homework 2&3.

4.5 Classification

Next, we trained a Naïve Bayes classifier on the dimension-reduced training dataset. We used a full covariance matrix for the classifier. Finally, we used the trained classifier to predict the label for each feature vector in the test set. We found that adding a small probability bias towards the BRS class slightly improved accuracy. This is in line with our observation that the number of windows that should use BRS is greater in our dataset than the number of windows that should use Token.

4.6 Using Predictions in the Simulator

To evaluate the performance impact of our prediction algorithm within the scope of a full manycore, we build a wireless architecture simulator that can be plugged to Multi2sim. This allows us to reuse the same real application traces that we described in Section 3, to later feed to the network emulation part of the simulator. These traces are used to obtain the latency statistics of the two static MAC protocols (BRS and Token) compared to ours, and in the case of our protocol, they are used in conjunction with the window-by-window prediction coming out of our classifier (Figure 4). This window-by-window prediction is also fed to our network emulator, so that at the beginning of each interval we can switch to the predicted MAC protocol, and eventually gather the average packet latency resulting from using such a switching protocol approach as opposed to a static protocol approach.

5 Experimental Results

5.1 Evaluation of the Prediction Algorithm

The result of the classification on a execution of the Pagerank benchmark is presented in Figure 5. The top plot compares the evolution of the predicted protocol versus the one we determined to be ideal (as labeled in Section 4.2). The X-Axis is the window number and the Y-Axis is the selected protocol. Here we can observe the high accuracy with which our prediction algorithm matched the one we were expecting to get, for each window. In the bottom plot of Figure 5, we can visually inspect how our classifier decided between the two protocols. The X-Axis is again the window number and the Y-Axis is the log probability of selecting a protocol. The plot shows that even though there are some windows for which it is significantly easier to choose BRS (windows with very sparse traffic), this is not so much the case for Token, since Token is always a very fair protocol anyway, and even in windows with sparse traffic would not be too detrimental to use. This would no longer be true in bigger chips, where the waiting time for the token to arrive at each node would also be bigger.

Table 3: Confusion matrix of our prediction algorithm

		Predicted Protocol	
		BRS	Token
Ideal	BRS	0.9690	0.0310
Protocol	Token	0.2606	0.7394

These results lead to the confusion matrix presented in Table 3, which shows that we can predict BRS windows correctly 96.9% of the time, and Token windows correctly 73.9% of the time.

5.2 Impact of the Prediction Outcome to Real Application Traces

Networks are evaluated on the basis of the average latency of packets, which is defined as the time between the moment in which they are added to the input buffer, until the moment they are correctly received. Since the processor is governed by a global processor clock, the latency is expressed in clock cycles.

Figure 6 shows the speedup in terms of packet latency of our prediction algorithm with respect to the other static MAC protocols. Notice that since the vertical axis depicts the relative average packet latency with respect to our prediction algorithm, the higher the value, the worse it is with respect to our design. This plot exemplifies the strengths of our algorithm, which consistently provides a latency similar to the best of the two protocols. On average, the latency provided by our prediction algorithm is 1.82× lower than BRS and 1.22× lower than Token.

Applications such as *Ocean-NC* or *FFT* have a relatively low load and, as such, token performs poorly. Other applications such as *BFS*, *CC*, *Community*, or *Pagerank* are communication-intensive and inherently bursty. As a result, BRS shows a latency up to 3.67 times higher than our model. In some cases such as *Water* or *FFT*, our prediction method even outperforms both BRS and token, since the traffic for those applications heavily alternates between different types of patterns, and therefore switching to the right protocol at each window

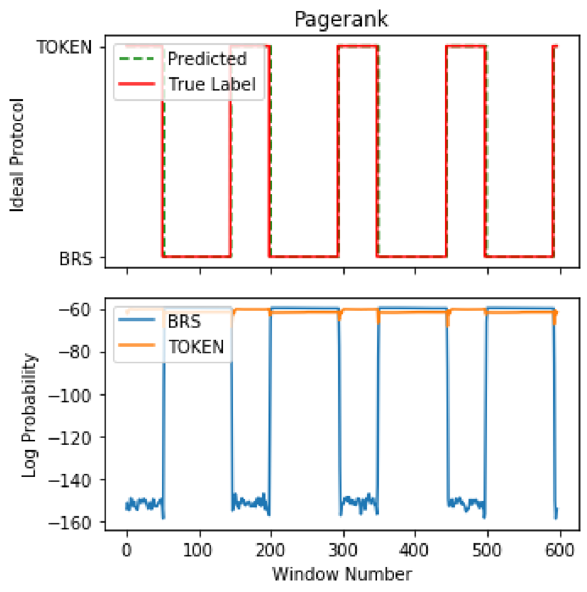


Figure 5: Evolution of the difference between the predicted and ideal protocol (top figure), and probability of choosing either BRS or Token (bottom figure), for each window of the Pagerank benchmark.

proves to be consistently better than using either of the two protocol statically throughout the whole application.

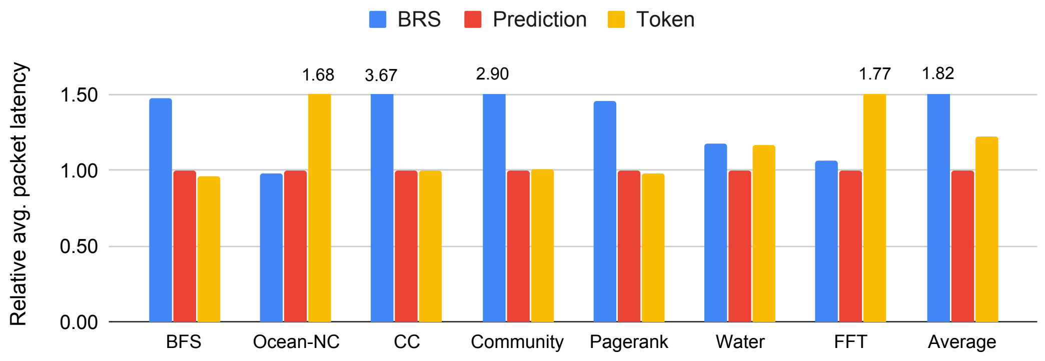


Figure 6: Average packet latency, normalized to our prediction algorithm, for different application traces.

6 Conclusions

In this project we have presented a prediction algorithm, capable of detecting patterns in the execution of several representative benchmarks, and dynamically switching to the most suitable MAC protocol for each of the fixed-size windows into which these applications are divided. We have shown that with a relatively simple and common prediction scheme, our prediction algorithm can achieve the low latency of contention-based protocols at low loads and the high throughput of fair collision-free protocols such as token passing. We have evaluated our protocol in a variety of real application traces, demonstrating an average speedup of up to 1.82 times with respect to other state-of-the-art static protocols. Our contribution, together with other advances in the field of on-chip networking, prove the feasibility of detecting communication/computation phases within and across applications, showing the benefits of taking advantage of such information, and paving the way for scalable and efficient manycore processors for general-purpose computing, machine learning acceleration, and data center and high-end server processors.

References

[1] S. Abadal, A. Mestres, J. Torrellas, E. Alarcón, and A. Cabellos-Aparicio, "Medium access control in wireless network-on-chip: A context analysis," *IEEE Communications Magazine*, vol. 56, no. 6, 2018.

[2] A. Mestres, S. Abadal, J. Torrellas, E. Alarcón, and A. Cabellos-Aparicio, "A mac protocol for reliable broadcast communications in wireless network-on-chip," in *Proceedings of the 9th International Workshop on Network on Chip Architectures*, 2016.

[3] X. Timoneda, S. Abadal, A. Franques, D. Manessis, J. Zhou, J. Torrellas, E. Alarcón, and A. Cabellos-Aparicio, "Engineer the Channel and Adapt to it: Enabling Wireless Intra-Chip Communication," *arXiv preprint arXiv:1901.04291*, 2018. [Online]. Available: https://arxiv.org/pdf/1901.04291.pdf

[4] O. Markish, B. Sheinman, O. Katz, D. Corcos, and D. Elad, "On-chip mmWave Antennas and Transceivers," in *NoCS*, 2015, p. Art. 11.

[5] R. S. Narde, J. Venkataraman, A. Ganguly, and I. Puchades, "Intra-and Inter-Chip Transmission of Millimeter-Wave Interconnects in NoC-based Multi-Chip Systems," *IEEE Access*, vol. PP, pp. 1–1, 2019.

[6] X. Timoneda, S. Abadal, A. Cabellos-Aparicio, D. Manessis, J. Zhou, A. Franques, J. Torrellas, and E. Alarcón, "Millimeter-Wave Propagation within a Computer Chip Package," in *Proceedings of the ISCAS '18*, 2018.

[7] V. Pano, I. Tekin, Y. Liu, K. R. Dandekar, and B. Taskin, "In-Package Wireless Communication with TSV-based Antenna," in *Proceedings of the ISCAS '19*, 2019, pp. 19–21.

[8] C. W. Byeon, C. H. Yoon, and C. S. Park, "A 67-mW 10.7-Gb/s 60-GHz OOK CMOS transceiver for short-range wireless communications," *IEEE Transactions on Microwave Theory and Techniques*, vol. 61, no. 9, pp. 3391–3401, 2013.

[9] Z. Wang, P. Y. Chiang, P. Nazari, C. C. Wang, Z. Chen, and P. Heydari, "A CMOS 210-GHz fundamental transceiver with OOK modulation," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 3, pp. 564–580, 2014.

[10] X. Yu, S. P. Sah, H. Rashtian, S. Mirabbasi, P. P. Pande, and D. Heo, "A 1.2-pJ/bit 16-Gb/s 60-GHz OOK Transmitter in 65-nm CMOS for Wireless Network-On-Chip," *IEEE Transactions on Microwave Theory and Techniques*, vol. 62, no. 10, 2014.

[11] X. Yu, H. Rashtian, and S. Mirabbasi, "An 18.7-Gb/s 60-GHz OOK Demodulator in 65-nm CMOS for Wireless Network-on-Chip," *IEEE Transactions on Circuits And Systems -I: Regular Papers*, vol. 62, no. 3, 2015.

[12] S. Subramaniam, T. Shinde, P. Deshmukh, S. Shamim, M. Indovina, and A. Ganguly, "A 0.36pJ/bit, 17Gbps OOK Receiver in 45-nm CMOS for Inter and Intra-Chip Wireless Interconnects," in *Proceedings of the SOCC '17*, 2017.

[13] R. Ubal, P. Mistry, D. Schaa, H. Ave, and D. Kaeli, "Multi2Sim: A Simulation Framework for CPU-GPU Computing," in *PACT*, 2012.

[14] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The splash-2 programs: Characterization and methodological considerations," in *ISCA*, 1995.

[15] M. Ahmad, F. Hijaz, Q. Shi, and O. Khan, “CRONO: A benchmark suite for multithreaded graph algorithms executing on futuristic multicores,” in *IISWC*, 2015.

[16] “Stanford Network Analysis Project `snap.stanford.edu`,” 2018.