Full System Exploration of On-Chip Wireless Communication on Many-Core Architectures

Rafael Medina[1], Joshua Kein[1], Yasir Qureshi[1], Marina Zapater[2], Giovanni Ansaloni[1] and David Atienza[1]
[1]*Embedded Systems Laboratory (ESL), EPFL, Switzerland,* [2]*HES-SO, Switzerland.*

***Abstract*—In order to develop sustainable and more powerful information technology (IT) infrastructures, the challenges posed by the "memory wall" are critical for the design of high-performance and high-efficiency many-core computing systems. In this context, recent advances in the integration of nano-antennas, enabling novel short-distance communication paradigms, promise disruptive gains. To gauge their potential benefit for the next-generation of many-core server designs, it is crucial to explore the impact of wireless communication links from a whole-system viewpoint, considering complex architectures and applications characteristics. To this end, in this work we introduce an extension to the popular gem5 full system-level simulator, enabling the simulation of many-core platforms featuring on-chip wireless channels. This new extension allows the flexible investigation of different combinations of wireless and wired interconnects, as well as diverse connection protocols. We showcase its capabilities by performing an architectural exploration, targeting a multi-core system executing image inference using an AlexNet Neural Network benchmark. A 2.3x speedup is obtained when implementing wireless communication between cores instead of traditional on-chip wired interconnects.**

***Index Terms*—On-chip wireless communication, Full system simulation, Architectural exploration, Many-core architectures.**

I. Introduction

Cloud computing has emerged as a pillar of the digital economy. It allows to centralize global Information technologies (IT) services with ubiquitous access at a minimal cost. As such, cloud computing is increasing its capacity with the proposal to have adaptive heterogeneous many-core computing architectures. However, the increasing adoption of many-core heterogeneous systems and complex memory hierarchies places an increasing burden on data links. This fact is aggravated by the growing gap between the performance of processors and of memory. Furthermore, data-heavy applications such as Artificial Neural Networks (ANNs) demand huge bandwidth between computational elements and the memory hierarchy. These problems constitute the "memory wall", which has driven architectural research in the last several decades [1].

Numerous solutions have arisen to confront this problem. Most recently, in-memory and near-memory processing target reducing the amount of data being sent through the memory hierarchy by enabling computation within the caches, memory, and/or storage [2]. Alternatively, there have been attempts at implementing faster and more efficient data connections leveraging technological breakthroughs, such as on-chip photonic links [3]. In this context, a promising avenue is that of nano-integrated wireless communication, which displays efficient data sharing with low-latency transfers [4][5].

Wireless capabilities are provided to computing components (such as processors and accelerators) by interfacing them to a transceiver and a nano-antenna. The resulting wireless communication module is then employed to transmit and receive Radio Frequency (RF) signals throughout the chip. Nano-antennas enable highly efficient and high-performance short-distance communication links, which support both point-to-point and broadcast communications. While this technology is still in its infancy, wireless communication up to 22 Gbps has been recently demonstrated on-chip [6][7]. Moreover, scaling projections predict that such implementations could provide bandwidths of the order of 100 Gbps [8]. The performance of wireless communication allows the integration of large on-chip networks with massive broadcast capabilities and architectural flexibility. These networks could efficiently sustain massively heterogeneous systems without losing purpose generality, avoiding the constraints of the wired interconnect. Fig. 1 shows a heterogeneous System-in-Package in which on-chip antennas allow for flexible intra- and inter-chip communication.

Wireless on-chip solutions open up an extensive design space. Multiple parameters can be considered, such as the used protocols, the system elements to be wirelessly linked, and/or the bandwidth of the transfers. To analyze the different options at the system level, simulators are essential for enabling the fast assessment of configurations. To date, only application-level simulators have been employed for evaluating wireless-enabled systems, such as Multi2Sim [4][5]. These simulators only emulate the instructions related to the guest program, abstracting from the OS and virtualizing the system call interface. However, nowadays application-level simulators do not accurately account for the overall system architecture,

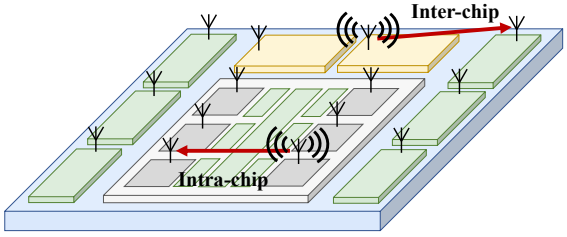

Fig. 1: Architecture of an heterogeneous 2.5-D System-in-Package integrating on-chip antennas into the chiplets.

This work has been partially supported by the EC H2020 WiPLASH project (GA No. 863337), the EC H2020 FVLLMONTI project (GA No. 101016776), and the ERC Consolidator grant COMPUSAPIEN (GA No. 725657).

including hardware and software stacks, or the OS interaction. Thus, they offer an incomplete view of the impact of integrating on-chip antennas. Indeed, we can overcome this limitation by developing a new generation of full system-level simulators, which emulate the complete processor ISA, OS, system architecture and I/O hardware. Consequently, full system simulators offer powerful frameworks to holistically explore heterogeneous many-core systems, providing estimations of energy, performances and area trade-offs.

For this reason, we herein present a gem5-X [9] wireless extension. This extension can connect different processing elements within a system without any modification to the rest of the standard gem5 components [10]. It can accommodate different connection bandwidths and protocols for collision avoidance. The use of this extension can guide the first designs of real systems integrating on-chip wireless communication.

In summary, our main contributions are:

- We propose a new self-contained gem5 extension that encapsulates the model of wireless transmission channels and protocols.
- We introduce the integration methodology of the wireless component in a gem5-X simulated system.
- We present an initial exploratory study about the benefits of a wireless on-chip core-to-core communication channel when implementing an ANN application in a multi-core system.

II. Modelling Core-to-Core Links

The design of our gem5-X wireless extension allows the hiding of the implementation of the channel and protocol from the system perspective, and exposes clear and simple interfaces to the system hierarchies. In fact, all standard gem5 components (such as CPUs, caches, and memories) can be used in conjunction with the wireless module [10].

Our focus is to reproduce two defining characteristics of CPU-to-CPU wireless links, namely (a) limitations in bandwidth and (b) collisions when multiple CPUs try to transmit data through the same channel at the same time.

A. System Integration

One goal of our model is to decouple the content of the transmitted data, the connectivity among CPUs, and the characteristics of the wireless channel. For this reason, the module employs three components: (a) a component modelling the wireless channel, (b) buffers employed to store the data being received by each processor, and (c) crossbars connecting buffers, CPUs, and the wireless channel model, as illustrated in Fig. 2. This approach allows us to describe the characteristic of the wireless channel only within the wireless block. In this manner, we minimize the effort required to extend our implementation and support different protocol designs [4][5]. Furthermore, our integration method allows the definition of multiple wireless channels, by instantiating and connecting different wireless components to the system hierarchy.

To model the wireless transmission, we assign a buffer to each of the connected CPUs, which can be accessed

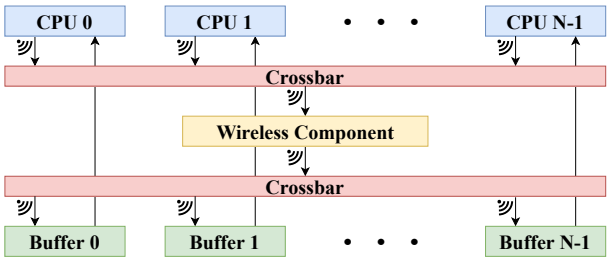

Fig. 2: Structure of the developed gem5 component modelling wireless communication among multiple cores. The component encapsulates the characteristics of the wireless medium and protocol (middle), managing the transmission and reception of data (bottom).

through a request to a specific range of addresses. A write operation from a CPU to the buffer of a different CPU is interpreted as a wireless transmission, and thus the limitations of bandwidth and packet collisions are considered for the communication. On the other hand, for modelling the reception of the transmitted data, we allow instantaneous read from a CPU to its own buffer. In case of collision, data loss is avoided thanks to the implemented collision protocols.

From the application viewpoint, wireless channels are defined as memory-mapped regions. At the start of the program, the pointers to the data to be wirelessly transferred are allocated to the addresses corresponding to the destination buffer. This method allows the transmissions to bypass the standard memory hierarchy. Then, within the program, a write to those regions is interpreted as a transmission from the executing core. The later read from the core owning the buffer corresponds to the reception of the data.

B. The gem5 Wireless Component

The wireless extension defines a component that integrates the different elements involved in wireless communication, namely: (a) the physical layer which performs data serialization and modulation for transmission, detects collisions and deserializes and demodulates the received data; (b) the Medium Access Control (MAC) which dictates when data can be transmitted; and (c) the wireless channel itself, i.e., the frequency band used for transmission between antennas.

To connect the wireless module to the system hierarchy, it includes two ports facing distinct crossbars: one facing the CPU cores and the other facing the buffers for wireless transmission. When receiving a request from the CPU-side port, the component decides whether the access is local or remote, depending on the CPU that initiated it and the access address. Local accesses are directly forwarded to the buffer-side port, while the transmitter services the remote ones.

This transmitter unit, embedded in the wireless component, serves the incoming write request packets and enables the modelling of bandwidth and collisions. The transmitter can be either busy or idle. It is busy when it is sending the data and cannot accept new incoming requests, for a time corresponding to the available transmission bandwidth. If there is no collision, the release of the transmitter signals the

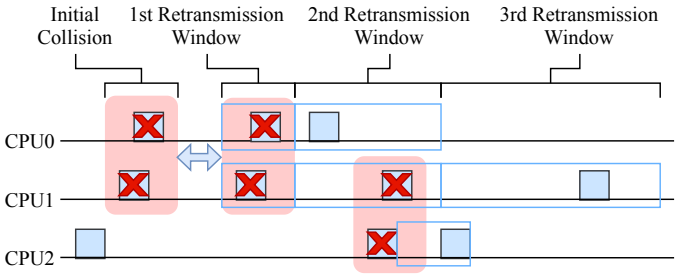

Fig. 3: Timing diagram illustrating the transmission of packets from different cores and their collisions.

successful completion of the wireless transmission. The time taken by the transmission of a packet (T) is set according to the size of the packet and the modelled bandwidth (B; bits per second) of the channel, according to the following formula:

$$T = \text{size(packet)}/B$$

The transmitter will reject packets if further transfers are initiated while in the busy state, which can be interpreted as:

1) Reaching the bandwidth limit of the channel, if the CPU originating the rejected packet is the same as the one transmitting. In this case, the rejected packet is served once the previous transmission finishes.
2) A collision of packets, if the source of the new packet is different from the one of the packet being transmitted. This is equivalent to two or more different CPUs trying to transmit at the same time wirelessly. The collision is handled according to the modelled protocol.

C. Adopted wireless communication protocol

Our approach allows extending the wireless model to support a variety of MAC protocols, such as token passing [5]. For the experiments shown in Section III, we consider a retransmission-based protocol. According to this protocol, whenever the transmitter rejects a packet, the CPU that sent the packet is stalled, and is signaled to retry the transmission once the current transmission has been served. The CPUs obtain the ownership of the transmitter in a First-Come First-Served (FCFS) fashion. In case the transmitter is idle and two CPUs try to transmit in the same cycle, the order of ownership is sorted following gem5-X's core ID.

As depicted in Fig. 3, to model the timing behavior of collisions we delay the write responses for a random time in a parameterizable window, which results in colliding transmissions to incur in random delays. Likewise, retransmission attempts can also collide. Hence, we repeat the above strategy to increase the window size by following an exponential backoff approach [4].

Collisions are detected by checking if (a) a CPU is trying to use the transmitter while it is servicing a request of another CPU, or (b) a CPU is trying to use the transmitter while previous collisions are still being resolved. In the first case, to account for the simultaneity of the transfers, we simulate the collisions by delaying the involved packets by a time equivalent to the sum of their transmission time:

$$\text{delay} = (n-1) \times W$$

where n is the number of collisions to be addressed and W is the minimum window size. This delay is equal to the necessary waiting time for all the colliding transmissions to have been served by the transmitter. After the delay, each response packet of the colliding transmissions is assigned a random time to be forwarded to the source CPU. In case a new transmission arrives during the resolution of the previous collisions, the response packet of this transmission is immediately assigned a random time, as the transmitter has already served all the previous requests.

When the assigned transmission time arrives, the wireless channel module checks for overlaps with the other scheduled retransmissions. If an overlap occurs, the involved packets are reassigned new random retransmission times; otherwise, the transmission of the packet successfully ends. The random transmission time is chosen within an interval whose size is decided via the exponential backoff algorithm. The minimum interval size (W) and the base of the exponent for backoff can be configured. In our first experiments, we fixed the retry window values to $(2c-1) \times W$, where c is the number of retries incurred by a packet. This approach makes colliding more unlikely the more retransmissions occur.

This process is illustrated in Fig. 3. First, transmissions from CPUs 0 and 1 collide, and thus are rescheduled within the first retransmission window. There, they collide and get rescheduled again, considering random times inside exponentially increasing transmission windows. In the second window, CPU 0 succeeds at sending the packet, while the retry from CPU 1 collides with a new transmission from CPU 2. Finally, both are successfully served in the next retransmission window.

III. Experiments

A. Setup

To perform a first evaluation of the performance of wireless-enabled multi-CPU SoC, we consider an 8-core target system executing the Alexnet convolutional neural network.

AlexNet [11] performs the inference of ten possible objects on three images from the CIFAR10 dataset [12]. Inputs are of dimensions 32x32x3 (32x32 images in RGB format). The outputs contain, for each image, one score for each of the ten classes. AlexNet is composed of nine layers: five convolutional layers, two max. pooling layers, and two linear layers.

The CNN execution is pipelined at the image level: the cores can process in parallel the activations of the different images, as soon as they receive them from previous cores. The activations are transmitted from core to core using a wireless link. Fig. 4 shows the mapping of the layers to the SoC cores.

The gem5-X-simulated system is a multi-core system with eight CPU cores. The system includes private L1 data and instruction caches and a shared L2 cache, as well as single channel wireless communication using the parameters shown in Table I. We compare it with a baseline system using a traditional wired cache hierarchy.

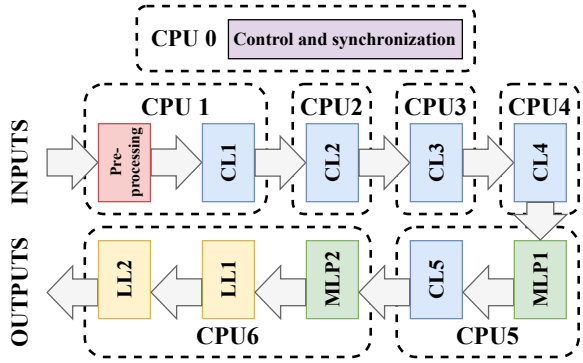

Fig. 4: Mapping of the AlexNet layers to the CPUs of the target system

TABLE I: System parameters

Processors	8x cores @2.0 GHz ARMv8 ISA, in-order
L1 Instruction Caches	8x private, 32 kB, 2-way, 2 cycle access
L1 Data Caches	8x private, 32 kB, 2-way, 2 cycle access
L2 Cache	Shared, 512 kB, 2-way, 20 cycle access
Memory	DDR4 2400 MHz, 4 GB
Operating System	Ubuntu LTS 16.04
Wireless Channel	100 Gbps bandwidth, single channel, 64 B or 5.12 ns min. window size, retransmission by exponential backoff

B. Results

Fig. 5 shows a comparison of the runtime and the communication latencies at different levels of the memory hierarchy, when executing the AlexNet application, between the wireless-enabled system and the baseline. The depicted L1 latency collects the sum of the latencies to all L1s caches in the system. As shown, the wireless channel produces a 2.3x execution speedup since it is a more efficient link and data can bypass the cache hierarchy.

The breakdown of latencies reveals that access to L2 and memory amounts to 90% of the total application runtime in the baseline (non-wireless) architecture, without accounting for the parallel execution of cores and access to L1 cache. This latency overhead is due to the communication of the layer activations from core to core. Since their large size makes L1 evictions frequent, it is necessary to traverse through the cache hierarchy instead of direct L1-to-L1 data sharing. Consequently, as Fig. 5b) depicts, enabling a wireless link between CPUs minimizes the need for L2 and DRAM access, as activations are being directly transmitted from core to core. This configuration thus greatly lowers the cumulative L2 and DRAM latency to near-zero. Hence, it reduces the total time needed for data communication and alleviates the memory-bound limitations of the application.

Looking at the latency of L1 caches, we observe a smaller reduction when implementing the wireless channel. Because the instructions, the convolution weights, and the temporary data are local to each core and stored in the L1 caches, access to these cannot benefit from direct core-to-core communication. At the same time, Fig. 5b) shows L1 caches and wireless communication to present a comparable latency. Nonetheless, since wireless transmissions and L1 access can occur in parallel, performance is not significantly affected by the introduced wireless latency.

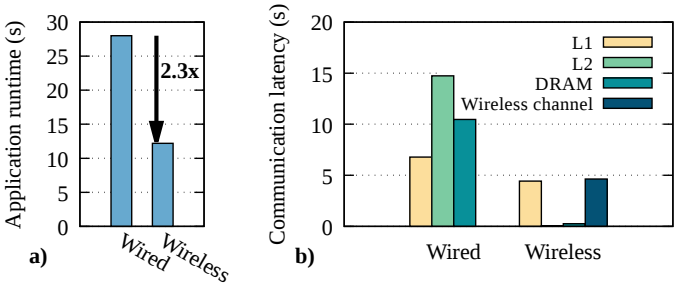

Fig. 5: (a) Runtime of the AlexNet benchmark on the simulated system with and without wireless communication between CPUs, and (b) breakdown of the communication delays during execution.

IV. Conclusion

In this paper, we have introduced a new extension to the popular gem5 full system-level simulator to enable the simulation of many-core platforms featuring on-chip wireless channels. This new extension enables the exploration of wireless interconnects and different connection protocols. We demonstrated its capabilities on an architectural exploration of a many-core system executing an image inference benchmark. Our results have shown a 2.3x speedup by using a wireless communication scheme. Indeed, such insights are crucial in the design and sizing of upcoming on-chip wireless interconnects for future many-core systems targeting specific application domains. Thus, we have shown that the impact of different architectural decisions can be accurately gathered with our novel gem5-X extension for full system simulation.

References

[1] W. A. Wulf and S. A. McKee, "Hitting the memory wall: Implications of the obvious," *SIGARCH Comput. Archit. News*, 1995.
[2] S. Bavikadi *et al.*, "A review of in-memory computing architectures for machine learning applications," in *Proc. ACM GLSVLSI*, 2020.
[3] C. Batten *et al.*, "Designing chip-level nanophotonic interconnection networks," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2012.
[4] S. Abadal *et al.*, "Wisync: An architecture for fast synchronization through on-chip wireless communication," *ACM SIGPLAN Notices*, 2016.
[5] V. Fernando *et al.*, "Replica: A wireless manycore for communication-intensive and approximate data," in *Proc. ASPLOS*, 2019.
[6] X. Yu *et al.*, "Architecture and design of multichannel millimeter-wave wireless NoC," *IEEE Design Test*, 2014.
[7] N. Weissman and E. Socher, "9mW 6Gbps bi-directional 85–90GHz transceiver in 65nm CMOS," in *9th European Microwave Integrated Circuit Conference*, 2014.
[8] M. F. Chang *et al.*, "CMP network-on-chip overlaid with multi-band RF-interconnect," in *IEEE 14th International Symposium on High Performance Computer Architecture*, 2008.
[9] Y. M. Qureshi *et al.*, "Gem5-X: A many-core heterogeneous simulation platform for architectural exploration and optimization," *ACM Trans. Archit. Code Optim.*, 2021.
[10] N. Binkert *et al.*, "The gem5 simulator," *SIGARCH Comput. Archit. News*, 2011.
[11] A. Krizhevsky *et al.*, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, 2017.
[12] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.