Horizon 2020 Program (2014-2020)

FET-Open Novel ideas for radically new technologies

FETOPEN-01-2018-2019-2020

WiPLASH

Architecting More than Moore – Wireless Plasticity for Massive Heterogeneous Computer Architectures [1†]

D5.3: Package-level exploration

WP5 - Multi-scale Simulation

Contractual Date of Delivery	31/03/2022
Actual Date of Delivery	31/03/2022
Deliverable Security Class	Public
Editor	Giovanni Ansaloni (EPFL)
Contributors	EPFL (Leader), IBM
Quality Assurance	Peter Haring Bolívar (UoS) Renato Negra (RWTH)

[1†] This project is supported by the European Commission under the Horizon 2020 Program with Grant agreement no: 863337

Document Revisions & Quality Assurance

Deliverable Number	D5.3
Deliverable Responsible	EPFL
Work Package	WP5
Main Editor	Giovanni Ansaloni

Internal Reviewers

1. Peter Haring Bolívar (UoS)
2. Renato Negra (RWTH)

Revisions

Version	Date	By	Overview
0.1	21/02/2022	Giovanni Ansaloni	Document created
0.2	23/02/2022	Giovanni Ansaloni, Joshua Klein	Section 1 and 2 added
0.3	04/03/2022	Giovanni Ansaloni, Rafael Medina	Section 3.1 added
0.4	07/03/2022	Halima Najibi	Section 3.3 added
0.5	11/03/2022	Rafael Medina	Section 3.2 added
0.6	14/03/2022	Giovanni Ansaloni	Proof reading, section 4 added
1.0	31/03/2022	Renato Negra	Internal review

Executive Summary

This deliverable reports the progress of WP5 regarding the development of the WiPLASH system-level simulator, and its use to investigate the applicability and performance of the WiPLASH vision from a whole-system standpoint.

Building upon the initial findings described in D5.2, we here illustrate an extensive design space exploration of system embedding tightly-coupled Analog In-Memory computing (AIMC) memristive arrays, spanning different applications and architectural arrangements.

Moreover, we detail the characteristics of the updated gem5 wireless communication component. Our novel model implements two new and important characteristics. First, it can be instantiated anywhere in the system hierarchy (including in-between cache levels). Second, it can route coherency traffic as well as data. We showcase these features by modelling a system having a package-level wireless channel with different bandwidth constraints. We also provide an analysis showcasing the benefit from a thermal management standpoint of the use of wireless interconnects.

Abbreviations and Acronyms

AIMC	Analog In-Memory Computing
BW	Bandwidth
CNN	Convolutional Neural Networks
CPU	Central Processing Unit
FCFS	First Come First Served
ISA	Instruction Set Architecture
LSTM	Long Short-Term Memory
MAC	Medium Access Protocol
MLP	Multi-Layer Perceptron
MVM	Matrix-Vector Multiplication
SoC	System on Chip
SPM	Scratchpad Memory
WP	Work Package

The *WiPLASH* consortium is composed by:

UPC	Coordinator	Spain
IBM	Beneficiary	Switzerland
UNIBO	Beneficiary	Italy
EPFL	Beneficiary	Switzerland
AMO	Beneficiary	Germany
UoS	Beneficiary	Germany
RWTH	Beneficiary	Germany

Table of Contents

List of Figures

List of Tables

1 Introduction

Focus of Work Package 5 (WP5) is the development of a novel system simulation framework, instrumental for assessing of the impact of the innovations pursued by the WiPLASH project from a system- and application-wide perspective. In particular, the framework allows to quantify the benefits deriving from the use of nanoantennae for on-chip and in-package communication, and that of Analog In-Memory Compute (AIMC) tiles for speeding up computation.

System-level simulations allows to target realistic large-scale workloads, such as Artificial Intelligence (AI) applications, executed as part of complete software stacks, including operating systems. It also enables performance measurements on complex architectures, which may feature many processing cores, multiple layers of caches and specialized processing units.

In the context of the WiPLASH project, the project partners EPFL and IBM are employing to this end the widely adopted gem5 environment[2,3] including the gem5-X extensions developed by EPFL[4], which allows, among other advanced features, the use of fine-grained checkpointing and profiling, and the instantiation of scratchpad memories in simulated architectures. gem5-X supports Ubuntu Linux 16.04.

gem5-X has been extended with two novel components, which are key contributions of WP5:

- The first developed component allows to model AIMC accelerators of varying sizes. AIMCs are tightly coupled to standard gem5 processors as dedicated functional units. We provide Instruction Set Extensions (ISEs) to govern AIMC operations: programming weights, transferring inputs, enabling analogue computation, and retrieving outputs.
- The second component models on-chip and in-package wireless interconnects. Its functionality is based on the standard interconnect block provided by gem5, enhanced by adding wireless-specific characteristics such as support for collision detection and avoidance protocols.

This deliverable builds upon the activities described in D5.2, submitted on M24. It strengthens the findings described therein along two axes:

- We present a detailed design space exploration showcasing the performance of AIMC-accelerated systems. We target three different AI workloads: Multi-Layer Perceptrons (MLPs), Long Short-Term Memory (LSTM) recurrent neural networks and Convolutional Neural Networks (CNNs). For each, we propose different data mapping strategies to AIMC cores, achieving speedups in excess of 10X with respect to equivalent systems not featuring such accelerators.

[2] Binkert, N., Beckmann, B., Black, G., Reinhardt, S.K., Saidi, A., Basu, A., Hestness, J., Hower, D.R., Krishna, T., Sardashti, S. and Sen, R., 2011. The gem5 simulator. *ACM SIGARCH computer architecture news*, *39*(2), pp.1-7.

[3] Lowe-Power, Jason, Abdul Mutaal Ahmad, Ayaz Akram, Mohammad Alian, Rico Amslinger, Matteo Andreozzi, Adrià Armejach et al. "The gem5 simulator: Version 20.0+." *arXiv preprint arXiv:2007.03152* (2020).

[4] Qureshi, Yasir Mahmood, William Andrew Simon, Marina Zapater, Katzalin Olcoz, and David Atienza. "Gem5-x: A many-core heterogeneous simulation platform for architectural exploration and optimization." *ACM Transactions on Architecture and Code Optimization (TACO)* 18, no. 4 (2021): 1-27.

- We introduce a novel design of the wireless crossbar module first introduced in D5.2. As opposed to our previous implementation, our improved design can act as a bridge between any computing and memory components, including different cache hierarchy levels. It can therefore model in-package as well as on-die wireless links. We herein detail its implementation. In addition, we present a full-system study of the performance achieved by wireless communication in different architectural arrangements, including thermal considerations.

This deliverable is the endpoint of the development activities in WP5 focusing on the system simulation infrastructure. In conjunction with the wireless protocol and the low-level hardware design explorations performed in WP3 and detailed in D3.2, it allows the consortium to flexibly evaluate the potential of nanoantennae as an enabling technology for short-distance, high-bandwidth communication.

Beyond the contributions to the WiPLASH consortium, the multiscale simulations at the core of WP5 tangibly advance the state-of-the-art in the system simulation field. Our findings on AIMC-enabled systems have recently been submitted for publication[5]. Moreover, the initial implementation of the gem5-X on-chip wireless module (described in D5.2) has been presented in the IEEE LASCAS conference[6]. We are currently extending this work in a further manuscript that showcases the improved design introduced in this deliverable.

[5] Klein, Josua, Irem Boybat, Yasir Qureshi, Martino Dazzi, Alexandre Levisse, Giovanni Ansaloni, Marina Zapater, Abu Sebastian, David Atienza. “ALPINE: Analog In-Memory Acceleration with Tight Processor Integration for Deep Learning”, *submitted for peer review*.

[6] Medina, Rafael, Joshua Kein, Yasir Qureshi, Marina Zapater, Giovanni Ansaloni, David Atienza, “Full System Exploration of On-Chip Wireless Communication on Many-Core Architectures”, IEEE Latin American Symposium on Circuits and Systems (LASCAS), 2021.

2 Design space exploration of Analogue In-Memory acceleration

2.1 Tightly coupled AIMC systems

Following from D5.2 initial implementation, modeling, and interfacing of AIMC tiles, we formalize and integrate the necessary components into a novel extension of gem5-X. We call this extension, "ALPINE: Analogue In-Memory Acceleration with Tight Processor Integration for Deep Learning". ALPINE includes highly parameterizable software models of crossbars integrated into gem5-X that are interfaceable via custom ISA extension. Furthermore, aimclib, the software library that allows ease-of-use of the AIMC tile interfaces, is included with ALPINE to easily enable AIMC tile usage in digital reference applications.

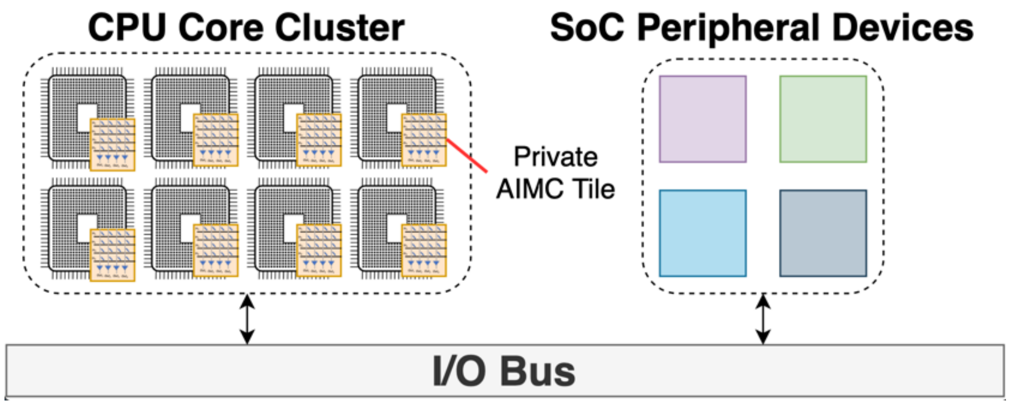

Figure 1: MPSoC architecture featuring tightly-coupled AIMC accelerators.

With such capabilities, exploring numerous possible computer system configurations and neural network architectures is possible. We adopt a tightly coupled integration strategy, as shown in Figure 1. That is, for each CPU core on the MPSoC, we assign one AIMC tile, and each AIMC tile is only accessible by its designated CPU core, *i.e.*, a CM_PROCESS instruction call by CPU X will only initiate the MVM computation in AIMC tile X. The organization of this architecture is similar to that of the ARM NEON SIMD coprocessor configuration[7].

In comparison to loosely coupled architectures, which usually involve off-chip accelerators with hard-coded digital logic, tightly coupled architectures allow for a high degree of configurability and on-the-fly changes to neural network structure because they rely on the CPU for digital logic capabilities. Therefore, performing activation functions such as ReLU and Sigmoid require no additional hardware in more basic neural networks, and intermediary functions that occur in between neural network layers (such as in long short-term memory cells) do not require expensive data transmission operations through peripheral buses.

[7] ARM Neon: https://developer.arm.com/architectures/instruction-sets/simd-isas/neon.

To evaluate the performance and energy benefits of utilizing tightly coupled AIMC tiles, as well as identify bottlenecks, we select three target applications for exploration studies: multilayer perceptrons (MLPs), recurrent neural networks (LSTMs), and convolutional neural networks (CNNs). These target applications are then subjected to multiple cases, where we examine different neural network layouts for the same neural network across single-CPU core and multi-CPU core configurations, comparing directly conventional CPU-only and AIMC-enabled applications.

We consider two test architectures in our experiments:

- A **high-power** platform operating at 2.1 GHz, having 64-kB L1 data and instruction caches private to each core, and a shared 1-MB L2 cache.
- A **low-power** platform operating at 0.8 GHz, with 32-kB L1 per-core data and instruction caches, and 512 kB shared L2 cache.

We use energy models from gem5-X for processors and memories to calculate the energy of the system. Where the AIMC tiles are used, we employed the AIMC energy characterization provided by the partner IBM[8,9].

2.2 Exploration Study 1: Multilayer Perceptrons

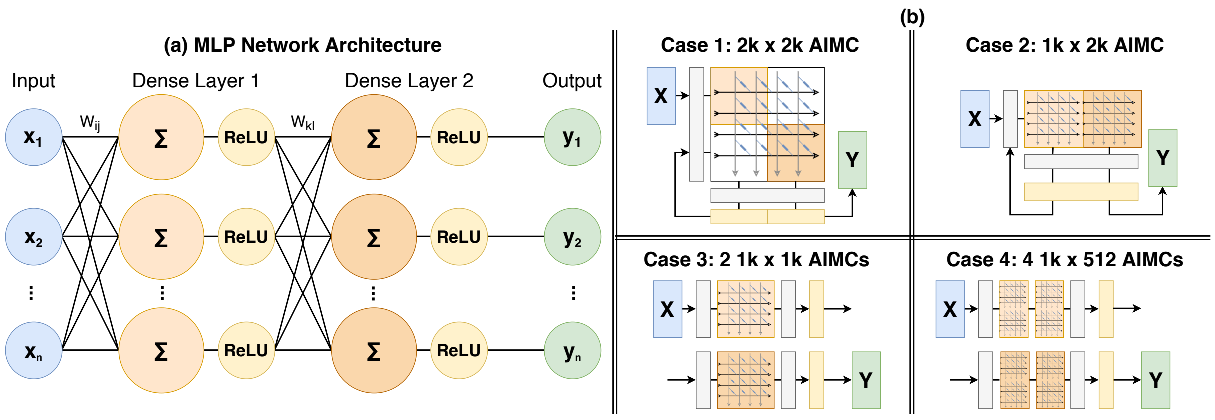

*Figure 2: (a) Target MLP Neural Network architecture, consisting of two fully-connected (dense) layers using ReLU activations.
(b) Test cases of various neural network mappings to different architectures. Cases 1 and 2 are single-CPU core cases that each pack both MLP layers into one AIMC tile. Cases 3 and 4 are dual- and quad-CPU core cases, respectively, that distribute weights across multiple AIMC tiles.*

For the first exploration study, we consider a simple MLP shown in Figure 2 (a). Each inference is dependent on the MVM of two dense layers with ReLU activation functions. In other words, the bulk of the inference workload is dependent upon the MVM. The

[8]R. Khaddam-Aljamehet al., “Hermes core – a 14nm CMOS and PCM-based in-memory compute core using an array of 300ps/lsb linearized CCO-based ADCs and local digital processing,” in 2021 Symposium on VLSI Circuits.

[9] S. Nandakumaret al., “Mixed-precision deep learning based on computational memory,” Frontiers in Neuroscience, 2020.

MLP is mapped to AIMC tiles in four different cases shown in Figure 2 (b), where Cases 1 and 2 are single-CPU core cases that use larger AIMC tiles while Cases 3 and 4 are multi-CPU core cases that use smaller AIMC tiles and have to synchronize data every inference. This exploration study provides insight into the maximum benefit of integrating tightly coupled AIMC tiles while allowing us to easily quantify the impact of individual subregion-of-interests (sub-ROIs) and deduce bottlenecks.

We run this MLP on our target low-power and high-power MPSoCs for 10 inferences. We are able to run this MLP for a small number of inferences because the per-inference deviation of performance statistics is observed to be very small.

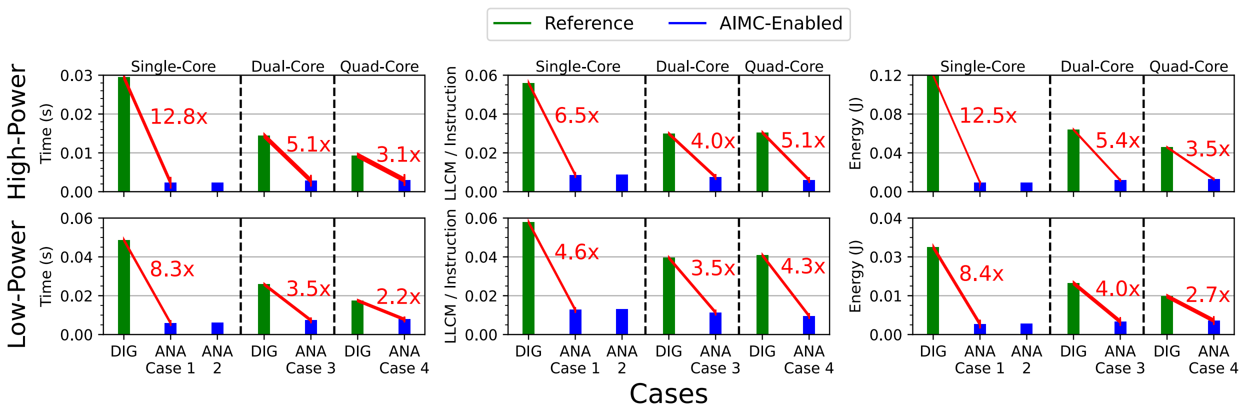

Figure 3: Aggregate results for running both digital reference MLP and its AIMC-enabled counterpart. The graphs from left to right show the total runtime, memory intensity, and total energy of the application. The top row shows the results for the high-power system while the bottom row shows the results for the low-power system.
"DIG" refers to the digital reference application while "ANA" refers to the analogue AIMC-tile enabled application, with case numbers corresponding to those in Figure 2. Results are grouped according to the number of CPU cores utilized.

The aggregate results for our MLP experiments are shown in Figure 3. The highlight result is that we see a maximum speedup of 12.8x and maximum energy improvement of 12.5x in the high-power system coming from the single-CPU core Case 1 with the largest AIMC tile. In general, including AIMC tiles always yields significant performance and energy benefits over the digital reference. However, it is interesting to note that Case 1 is actually the best performing MLP overall, even out-performing dual- and quad-CPU core cases. The reason for this can inferred by Figure 4 and is discussed in the following.

When examining the distribution of time in sub-ROIs in Figure 4, we see that the primary bottleneck in the single-core case is the analogue queueing operation (39.2%). The inclusion of AIMC tiles makes the MVM operation trivial (0.7% of the runtime) when compared to the average digital reference (96% of the runtime). Note that the analogue queueing time refers to the queueing for both layers of the MLP. In the multi-CPU core cases, analogue queueing time also includes the time taken to send the outputs of the first layer to the input memory of the AIMC tile of the second layer, which means there is extra overhead to interlayer communication when compared to the single-CPU core case.

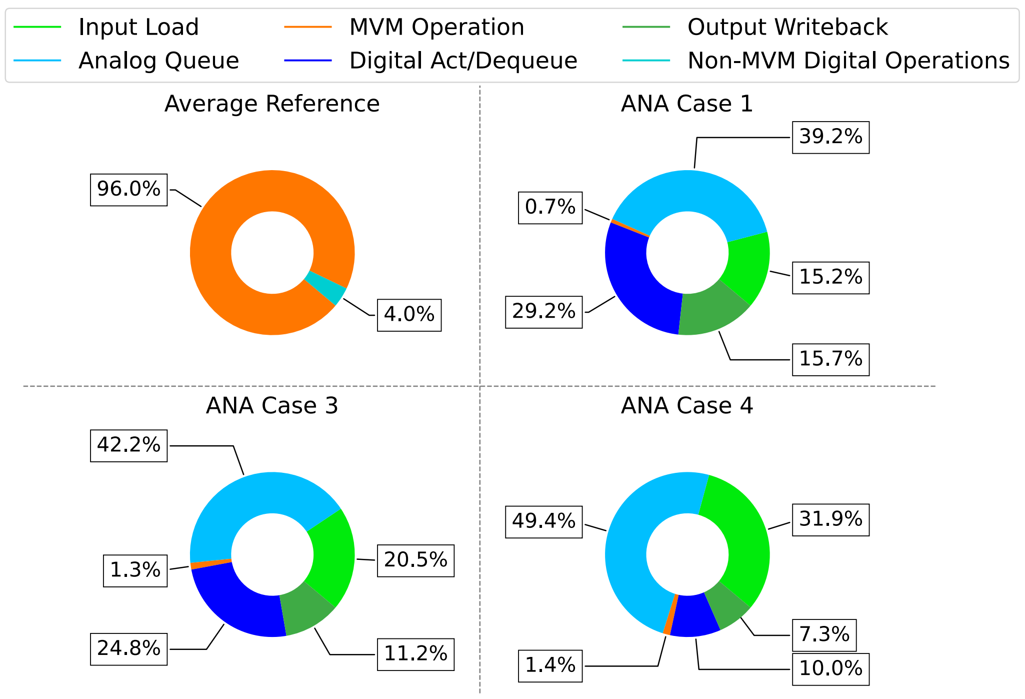

Figure 4: Average distribution of runtime in sub-ROIs of the MLP applications for selected cases. Reference refers to the digital reference MLP application. "Analog Queue" and "Digital Act/Dequeue" refer to the process of loading inputs into the AIMC tile input memory and fetching results from the AIMC tile output memory, respectively.
Activation functions are applied on dequeue to minimize memory accesses. The standard deviation is less than 2.3% for all cases.

Table 1: Algorithmic complexity (per inference) listing for each sub-ROI of the 2-layer MLP. Note that the ReLU activation function is performed at the same time as the analogue dequeue, so its complexity is included there.

Operation	**Digital Reference**	**AIMC-Enabled**
Input Load	$O(n)$	$O(n)$
Analog Queue	N/A	$2{\cdot}O(n)$
MVM	$2{\cdot}O(n^2)$	$2{\cdot}O(1)$
Analog Dequeue	N/A	$2{\cdot}O(n)$
ReLU	$2{\cdot}O(n)$	$O(1)^*$
Writeback	$O(n)$	$O(n)$
Total	$2{\cdot}O(n^2)+4{\cdot}O(n)\approx O(n^2)$	$6{\cdot}O(n)+2{\cdot}O(1)\approx O(n)$

* Note: the ReLU activation function is performed at the same time as the analogue dequeue, so its complexity is included there.

We quantify the benefits of the AIMC tile integration by doing a complexity analysis. Table 1 shows the algorithmic complexity of each sub-ROI for each inference of the MLP. The complexity confirms that the linear operations become the bottlenecks when the AIMC tile is integrated, because the MVM, the previous bottleneck, occurs in constant time. However, we notice in Figure 3 as well that the memory intensity of the

MLP decreases significantly. The reason for this can be seen in the space complexity listed in Table 2.

Table 2: Space complexity (per inference) listing for each memory component of the 2-layer MLP. 'n' indicates the MLP size.

Working Set Component	Digital Reference	AIMC-Enabled
Input	$O(n)$	$O(n)$
Dense Layers' Weights	$2{\cdot}O(n^2)$	N/A
First-layer Output	$O(n)$	$O(n)$
Final Output	$O(n)$	$O(n)$
Total	$2{\cdot}O(n^2) + 3{\cdot}O(n) \approx O(n^2)$	$3{\cdot}O(n) \approx O(n)$

During inference in the digital reference MLP, the weights of the dense layer must be loaded in during inference, thus making it apart of the working set, *i.e.*, the data memory is required to perform a single inference. If the working set is not contained within the L1 data cache of the CPU, run-time performance is hampered by L1 cache evictions and subsequent misses (cache thrashing), which result in a higher number of accesses to the larger, but slower, L2 caches or even to the main memory. Indeed, this is the case for the nonaccelerated implementations in both low- and high-power system configurations: weights are often evicted to lower levels of the memory hierarchy until they are used again in the next inference, leading to performance loss.

When the AIMC tile is introduced however, it is programmed once before inference starts and then the weights are never updated after initialization. Hence, at inference time memory is only required to load inputs, hold temporary variables, and store outputs. This data fits in the L1 cache, an important factor for increasing performance.

2.3 Exploration Study 2: Recurrent Neural Networks (LSTMs)

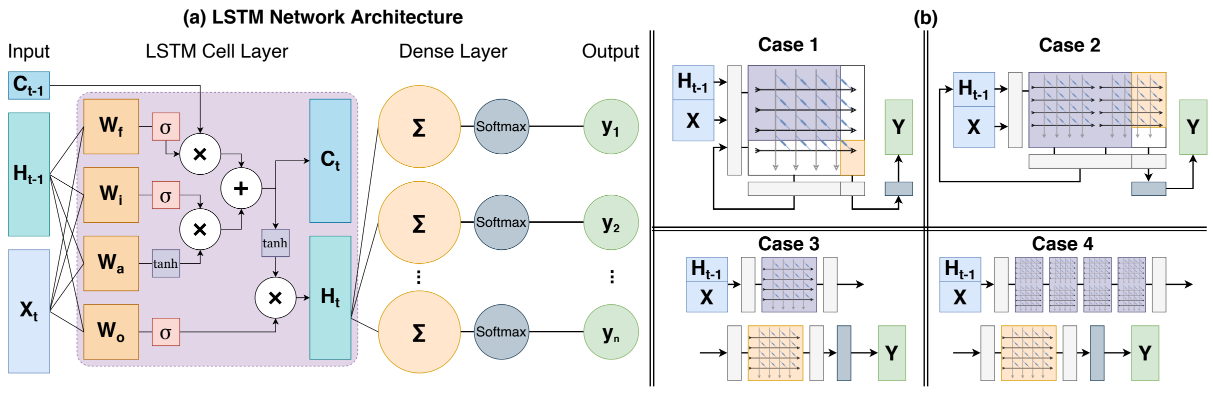

Figure 5: (a) Shows the target LSTM Network architecture, consisting of one LSTM cell (hidden) layer and one fully connected (dense) layer using Softmax activations. (b) Shows test cases of various neural network mappings to different architectures.
Cases 1 and 2 are single-CPU core cases that each pack both layers into one AIMC tile. Cases 3 and 4 are dual- and quin-CPU core cases, respectively, that distribute weights and layers across multiple AIMC tiles.

For the second exploration study, we consider a simple RNN, in this case a Long Short-Term Memory (LSTM) network, shown in Figure 5 (a). Each LSTM inference is dependent upon four MVM operations in the LSTM cell (hidden) layer, and one additional MVM in the dense layer. Additionally, each inference includes several digital gate operations in the LSTM cell that must be performed outside of the AIMC tiles using their results, as well as a time dependency from previous inference. Even for this application we consider four cases where we map the weights of each layer to the AIMC tile(s). Cases 1 and 2 are again single-CPU core cases while Cases 3 and 4 are multi-CPU cases. Note that Case 4 is now a quin-CPU core (5 CPU cores) case as opposed to Case 4 in the previous exploration study. Case 4 also has a much higher intralayer communication requirement, given that it needs to synchronize the output of four CPUs in the first layer before sending data to the second layer. Like in the MLP exploration study, we run the LSTM for 10 inferences.

In this exploration study, we also measure the effect of varying the dimension of the LSTM cell layer. We consider cell layer dimensions of 256, 512, and 750. We calculate the memory requirements of the weights to be 377.7 kB, 1.28 MB, and 2.6 MB, respectively, for the LSTM, where the two larger LSTMs are known to be larger than L1 and L2 caches for both of our test systems. We therefore expect the effects of cache thrashing to be pronounced in this exploration study.

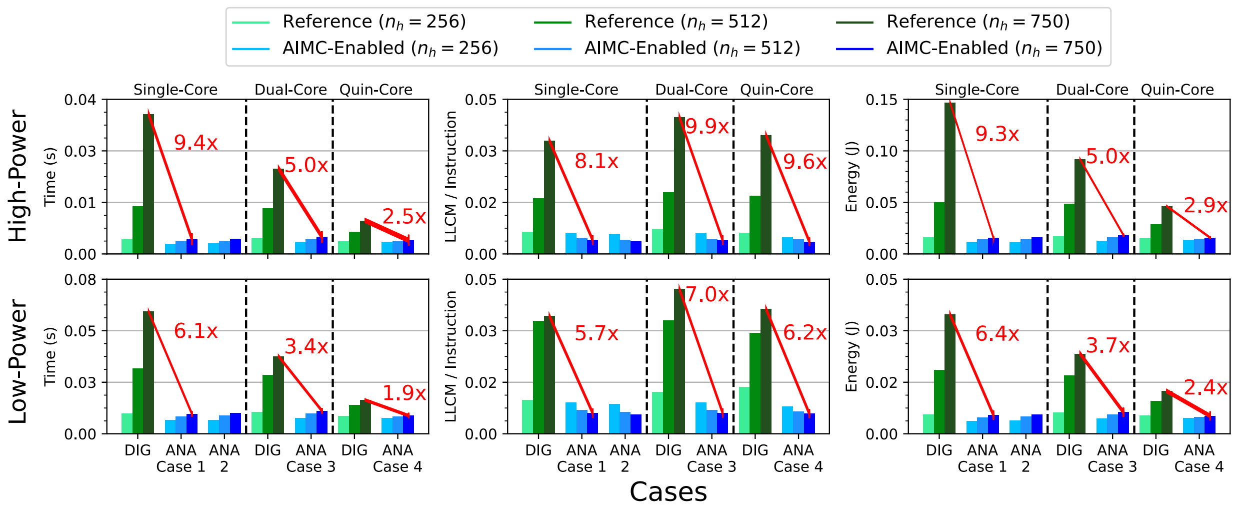

Figure 6: Aggregate results for running both digital reference LSTM and its AIMC-enabled counterpart. The graphs from left to right show the total runtime, memory intensity, and total energy of the application. The top row shows the results for the high-power system while the bottom row shows the results for the low-power system.
"DIG" refers to the digital reference application while "ANA" refers to the analogue AIMC-tile enabled application, with case numbers corresponding to those in Figure 5. The results are grouped according to the number of employed CPU cores.

The aggregate results for our LSTM experiments are shown in Figure 2.2.2. We observed a maximum speedup of 9.4x and maximum energy improvement of 9.3x in the high-power system, achieved by the single-CPU core case 1 with the largest AIMC tile. In general, including AIMC tiles always yields significant performance and energy benefits over the digital reference, however, unlike with the MLP, Case 4's quin-CPU core LSTM configuration actually performs the best with the largest hidden layer size, though with only ~20% benefit over Case 1's single-CPU core configuration.

Here we can see the effects of algorithmic complexity at play; when the dimension of the hidden layer increases, we see exponential increase in runtime in the digital reference LSTM across all cases. However, the change in runtime in the AIMC tile-enabled LSTM appears to be very small when the hidden layer size increases, as predicted. For completeness, the algorithmic complexity as a sum of sub-ROIs is calculated for the LSTM in Table 3.

Table 3: Algorithmic complexity (per inference) listing for each sub-ROI of the tested LSTM network. Note that the Softmax activation function is performed at the same time as the analogue dequeue, so its complexity is included there.

Operation	Digital Reference	AIMC-Enabled
Input Load	$O(n)$	$O(n)$
Analogue Queue	N/A	$2{\cdot}O(n)$
MVM (Cell + Dense)	$5{\cdot}O(n^2)$	$2{\cdot}O(1)$
LSTM Cell Ops	$9{\cdot}O(n)$	$9O{\cdot}(n)$
Sigmoid	$2{\cdot}O(n)$	$2{\cdot}O(n)^*$
Writeback	$O(n)$	$O(n)$
Total	$5{\cdot}O(n^2) + 13{\cdot}O(n) \approx O(n^2)$	$15{\cdot}O(n) + 2{\cdot}O(1) \approx O(n)$

Table 4: Space complexity (per inference) listing for each memory component of the LSTM network. 'n' indicates the dimension of a hidden layer being tested.

Working Set Component	Digital Reference	AIMC-Enabled
Input	$O(n)$	$O(n)$
Cell Layer's Weights	$4{\cdot}O(n^2)$	N/A
Cell Layer Output	$O(n)$	$O(n)$
Dense Layer's Weights	$O(n^2)$	N/A
Final Output	$O(n)$	$O(n)$
Total	$5{\cdot}O(n^2) + 3{\cdot}O(n) \approx O(n^2)$	$3{\cdot}O(n) \approx O(n)$

The space complexity of the LSTM for both the digital reference and AIMC tile-enabled application is also listed in Table 4. Here the differences in space complexity are more pronounced and correlate with the speedup and energy gains shown in the aggregate results. We calculate the space complexity of the reference application (including weights) to be 378 kB, 1.28 MB, and 2.6 MB while the space complexity of the AIMC tile-enabled application is estimated to be 0.66 kB, 1.17 kB, and 1.65 kB for hidden cell dimensions of 256, 512, and 750, respectively. Hence, while the working set of the smallest digital reference LSTM can fit in L1 and L2 caches for both test systems, the other digital reference LSTMs must go out to main memory every inference. Meanwhile, the working set of all AIMC tile-enabled LSTMs can be contained within the L1 cache of both test systems, contributing to the performance benefits.

2.4 Exploration Study 3: Convolutional Neural Networks

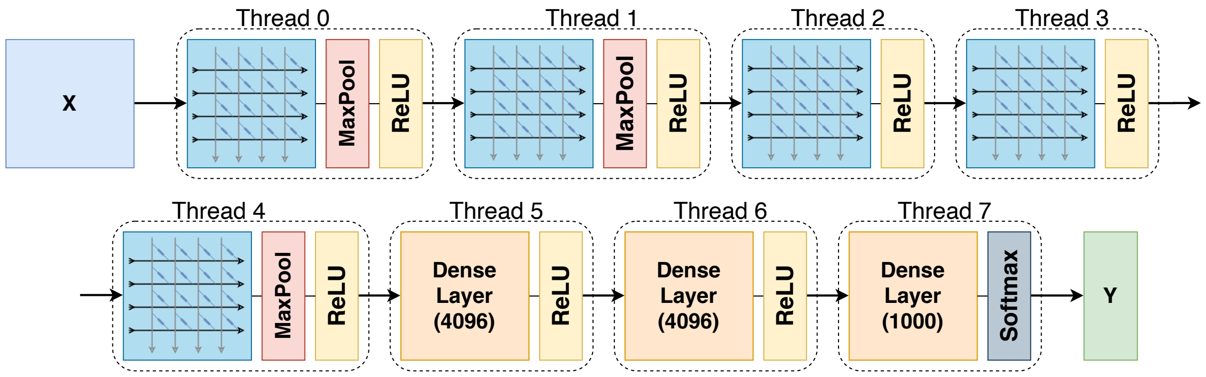

Figure 7: Target CNN architecture, consisting of 5 convolutional layers (in blue) and 3 fully connected (dense) layers in orange. 3 convolutional layers use the Max Pooling operation, and the last layer uses a Softmax activation function. All other layers use the ReLU activation function. In the AIMC tile-enabled version of the application, we map only the convolutional layers into tiles.

For the third and last exploration study, we consider CNNs modeled after the ones presented in Chatfield *et al.*[10] All of the CNNs have the same structure, as shown in Figure 7, but have different network parameters that lead to three categories: F(ast), M(edium), and S(low). Instead of multiple numbered cases, we implement each of these CNNs in a digital reference application and AIMC tile-enabled application, just as before. We map each layer of this CNN to a single CPU core, utilizing all 8 CPU cores in our test systems. Furthermore, we pipeline all layers of the system using pthreads and mutexes for synchronization.

The CNNs allow us to explore alternative uses for AIMC tiles during inference; while the AIMC tiles have so far seen only one MVM per inference in the two prior exploration studies, the convolution operation requires a special configuration of AIMC tiles in order for them to be utilized during inference. We utilize the strategy presented in Yakopcic *et al.*[11], which suggests that we flatten convolution kernels into the columns of the AIMC tile crossbar, and then flatten the shifting input into the AIMC tile's input memory. The result of the MVM operation is then the depth-wise output of the convolutional layer, meaning the AIMC tile must perform numerous MVM operations

[10] K. Chatfield et al., "Return of the devil in the details: Delving deep into convolutional nets,"arXiv preprint arXiv:1405.3531, 2014.

[11] C. Yakopcic, M. Z. Alom, and T. M. Taha, "Memristor crossbar deep network implementation based on a convolutional neural network," in 2016 International joint conference on neural networks (IJCNN). IEEE, 2016, pp. 963–970.

per inference. The total number of parameters in each CNN is on the order of 40 millions.

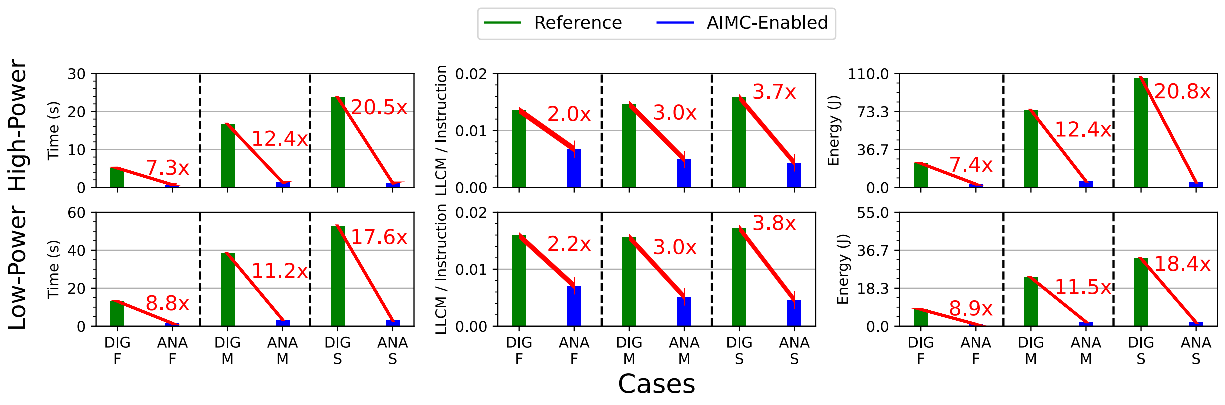

Figure 8: Aggregate results for running both digital reference CNNs and their AIMC-enabled counterparts. The graphs from left to right show the total runtime, memory intensity, and total energy of the application.
The top row shows the results for the high-power system while the bottom row shows the results for the low-power system.
"DIG" refers to the digital reference application while "ANA" refers to the analogue AIMC-tile enabled application, with cases referring to the F(ast), M(edium), and S(low) CNN configurations.

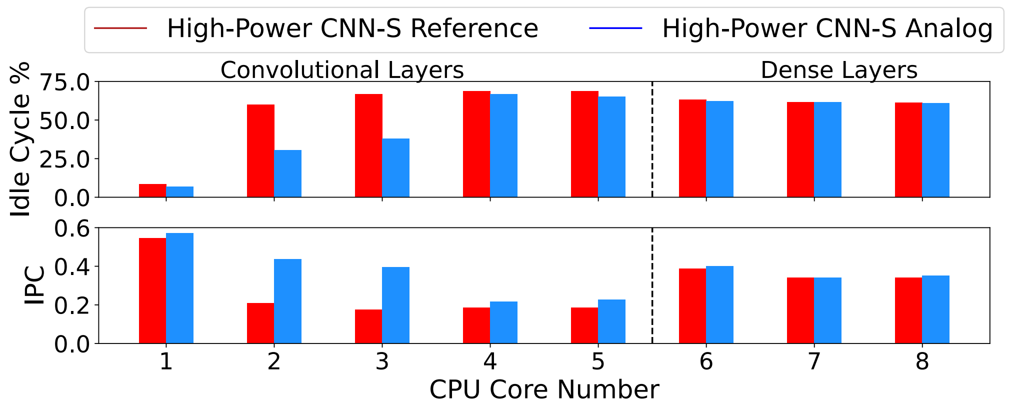

Figure 9: Core-wise utilization of the largest and slowest CNN-S in the high-power system with the metrics of IPC and percentage of idle cycles during inference.

The results of running the CNN inferences can be seen in Figure 8. We observe a maximum speedup of 20.5x and energy improvement of 20.8x in the high-power system running the slowest and largest CNN. The computational load of the convolutions is very high, and thus we measured a significant speedup through the use of the AIMC tiles.

The load-balancing of the CNN-S network on the high-power architecture is shown in Figure 9. We observed that the first layer acts as one bottleneck, because the input has to be loaded from storage. Beyond that however, all layers are relatively load-balanced, exhibiting similar idle time and IPC metrics. However, with the introduction of the AIMC tiles, we see the effects of uneven load balancing. The IPC of the AIMC tile-enabled convolutional layers starts out very high with a low idle rate, only to worsen as the layers propagate. This, as well as the hints from previous exploration studies suggesting that in AI applications the workload is communication-bound when considering AIMC acceleration, further justifying the WiPLASH focus on novel wireless-based in-package communication paradigms.

3 gem5-X generic wireless crossbar

3.1 Wireless crossbar design

This section details the second contribution of this deliverable, which focuses on the modelling of wireless-capable Systems-on-Chip. In this context, we build on the first iteration of the design of the gem5-X wireless component described in D5.2. That component provided an initial proof-of-concept for our system simulation approach, but also presented important limitations, which we address in this updated implementation.

First, the design in D5.2 could only model core-to-core wireless links. It therefore did not support processor-to-memory or memory-to-memory communications, which are crucial components of a system interconnect *e.g.*, for carrying the traffic between levels of the memory hierarchy. Such limitation leads to an underutilization of the wireless transmission channel in most cases, because processors can only issue loads or stores of limited width (8 bytes in our simulations), resulting in small transmission packets and high overheads. In contrast, messages among caches, which we support in this updated wireless transmission module, are typically much wider (up to 64 bytes in our target system), better amortizing the cost of accessing the wireless channel and generating less transmission conflicts.

Our novel design is developed by customizing the standard communication module in the gem5 framework ("gem5 crossbar" in Figure 10, left). Such an approach allows eliminating spurious delays due to arbitering and routing, accurately modelling the timing behaviour of on-chip and on-package wireless communication schemes.

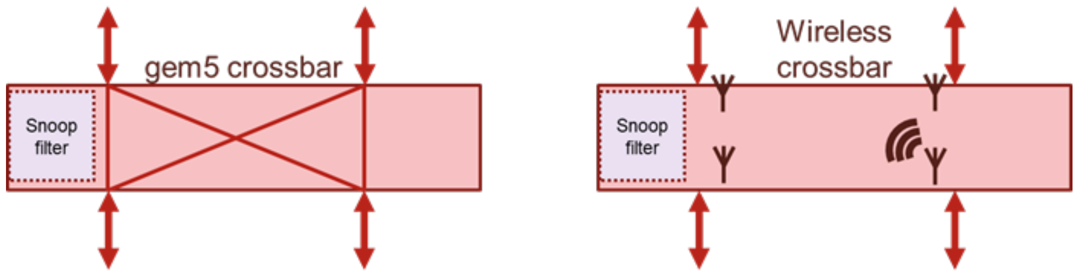

Figure 10: The developed wireless crossbar gem5 components derive from the standard gem5 interconnect module, adding support for wireless transmission protocols and transmission conflict detection / avoidance.

A high-level view of the developed component ("wireless crossbar") is provided on the right side of Fig. 10. As in the standard (nonwireless) gem5 component, the crossbar can handle many-to-many connections as well as (by employing a snoop filter) memory coherence traffic. In addition, it provides the necessary features to characterize wireless communication:

- The time required for data serialization and deserialization, modulation / demodulation and collision detection at the physical layer of the communication stack are taken into account as a tunable delay.
- A modular implementation is provided for the Medium Access Control (MAC) layer, which arbiters transmissions and implements a wireless

communication protocol. The wireless crossbar can support various protocols by implementing mechanisms for both satisfying a transmission request and dealing with collisions, which happen when two or more data transmitter try to access a wireless channel concurrently. The type of employed protocol, as well as its characteristics, are specified as parameters in the configuration file describing a target system using the standard gem5 syntax.

- The wireless channel itself is configured by providing the total bandwidth available for wireless communication between nanoantennae. The bandwidth of the channel is employed for computing the transmission time of a packet according to its size.

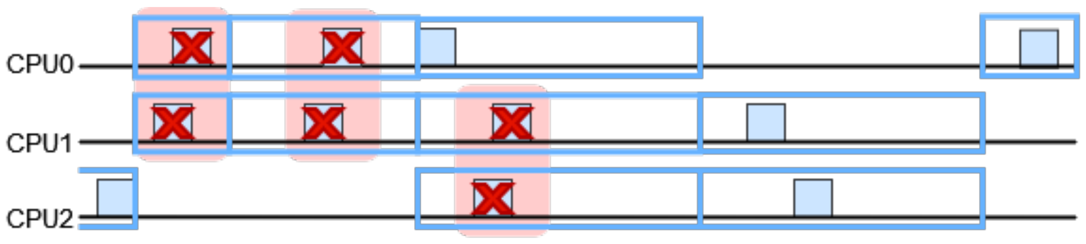

Figure 11: Timing diagramme of exponential backoff protocol. Red crosses mark colliding transmissions.

While the design of the wireless crossbar is not tied to a unique protocol, we currently support an exponential backoff retransmission strategy. As shown in Fig. 11, in this protocol all transmissions are randomly scheduled in windows of varying size. Whenever a collision occurs, the size of the windows is increased to reduce congestion. Conversely, when a transmission succeeds the window size is decreased to reduce the communication delay.

To implement such a protocol in the wireless crossbar, when a message is received at one of the ports, it is assigned a tentative end time in case the transmission succeeds. At the end time, collisions are checked retroactively. If no collision occurs, the packet is sent through the according port, the physical layer delays are applied, and the window size is decreased. Otherwise, the window size is increased, and the colliding packets are reassigned a tentative end time, when collisions will be checked again.

Unlike the first iteration of the wireless component design described in D5.2, every message that is send through the wireless crossbar is considered a wireless transmission. Thus, bandwidth limitations, MAC protocol, and physical layer delays are realistically applied to all data and cache traffic.

To simulate the integration of on-chip wireless communication in a system, the interfaced elements are connected to one of two sides of the wireless crossbar: processing elements and intermediate caches between processors and the crossbar are interfaced towards the CPU side, while memory and intermediate caches between main memory and the crossbar are connected to the memory side. The distinction is done to handle arbitering of the different types of messages: data requests, data responses, snoop requests, and snoop responses.

3.2 Using wireless crossbars to emulate in-chip and in-package wireless communication

Thanks to the ability of the new wireless crossbar to connect heterogeneous system elements, a broad range of wireless-enabled system-on-chip architectures can be explored, as depicted in Fig. 12.

The first possibility, indicated as (1) in Fig. 12, is wireless communication between caches, either at the same level (*e.g.*, L1-to-L1) and between caches at consecutive levels (L1-to-L2). With this approach, complex wired interconnects can be avoided for systems with many local and shared caches, and broadcast capabilities can be employed to streamline the snooping process.

Secondly, the wireless crossbar can be employed to connect processing cores and scratchpad memories (SPMs) for sharing information without needed to traverse the memory hierarchy (those links are marked as type (2) in Fig. 12). SPMs can be organized in two different ways. They can either be local to each core, so that one core can access to its SPM with a physical bus, and only remote accesses from different cores are done wirelessly. Alternatively, a single large SPM can be shared among all processors, only allowing for wireless accesses. Both these organization are useful for synchronization and data sharing.

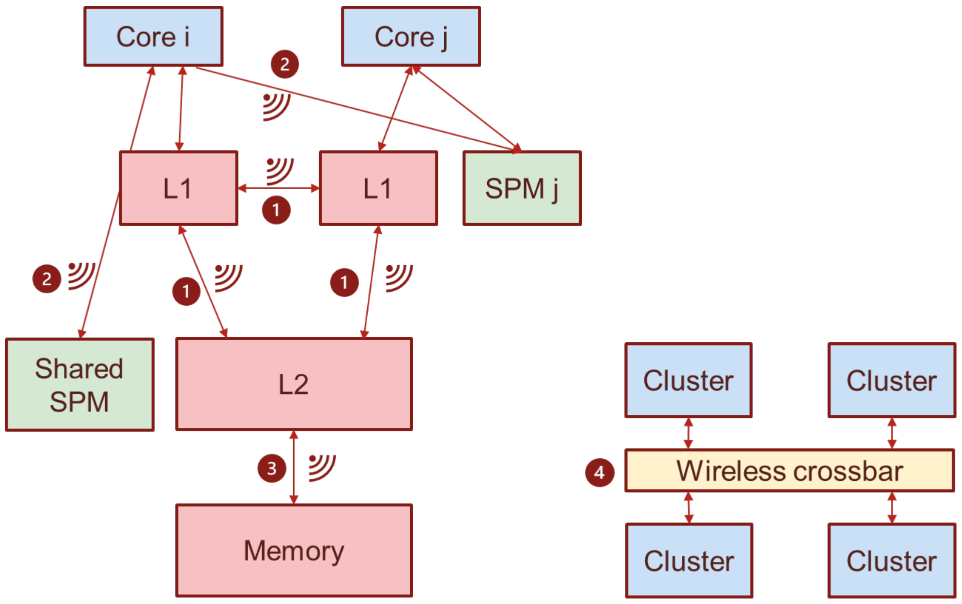

Figure 12: Connection links supported by the wireless crossbar: cache-to-cache (1), cores-to-SPMs (2), caches-to-main-Memory (3), cluster-to-cluster (4).

On-chip wireless communication could also substitute the memory bus (links of type (3) in Fig. 12), in a similar way as wireless connection between caches. With this approach, flexible communication between the memory controller and last level caches can be supported.

Finally, the wireless crossbar supports communication between clusters in a System-on-Package or Network-on-Chip, depicted as links of type (4) in Fig 12. Especifically, a shortcut path can be offered between physically distant cores, while short-distance messages can be routed via the traditional NoC topologies, such as rings or XY meshes.

3.3 Wireless System-on-Chip architectural exploration

3.2.1 Experimental setup

To assess the performance of a wireless-enabled many-core system, we considered a 64-cores System-on-Chip employing an 8x8 XY mesh topology, as displayed in Fig. 13. Adjacent cores are connected using physical wires able to transmit 128 bits per clock cycle. Control and data packages are routed through these links to send information throughout the network. Additionally, every cluster of 4 cores is interfaced to a transceiver and a nanoantenna, as shown in Fig. 13.b, enabling wireless communication among clusters, as in the configuration (4) in Fig. 12. A shared memory (modelling a L3 cache) is also interfaced through the wireless channel. The system characteristics are as following:

- CPUs: 64x ARMv8 Out-of-Order, operating at 2.0 GHz
- L1 Instruction cache: 64x private caches, 32 kB
- L1 Data cache: 64x private caches, 32 kB
- L2 cache: 64x private caches, 256 kB
- Main memory: 4 GB DDR4 DRAM

The system is simulated using gem5-X.

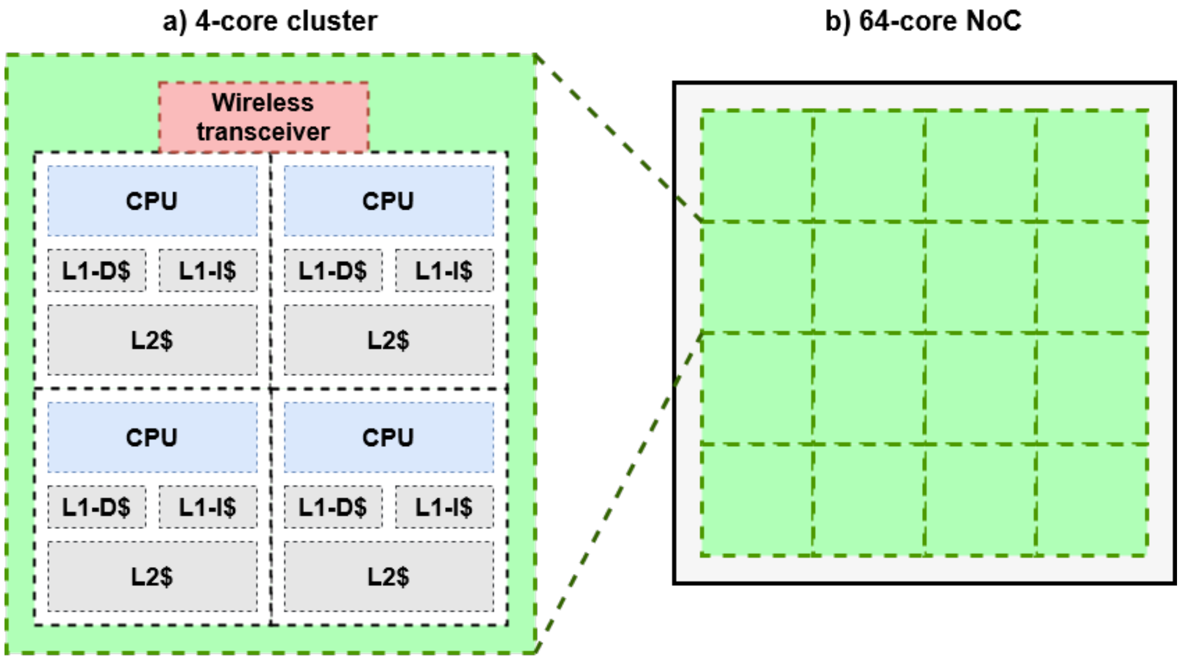

Figure 13: Block diagram of the simulated system:
(a) 4-core cluster integrating a wireless transceiver
(b) 64-core SoC built with 16 clusters, forming a XY mesh topology

The developed wireless crossbar component is employed to emulate a single communication channel accessed by each of the 4-core clusters and by the shared memory. The employed MAC protocol is exponential backoff. Simulations are run for different wireless bandwidths: 50, 100, 200, and 500 Gbps.

We executed on the system the STREAM benchmark[12], which measures peak memory bandwidth and, thus, is a suitable test for chip interconnects. The benchmark is run on 16 cores of the system, one in each 4-core cluster.

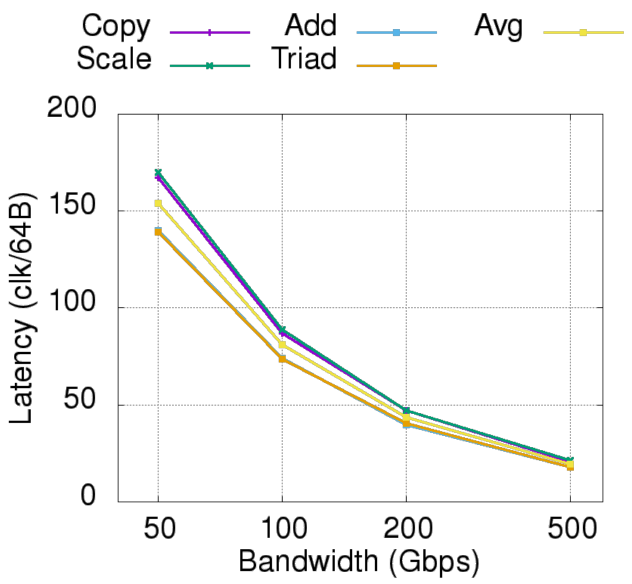

Figure 14: Effective latency of wireless communication obtained when running STREAM kernels, employing the exponential backoff protocol for wireless collision handling

3.2.2 Functional exploration outcomes

Figure 14 shows the average effective latency of the wireless transmission at different bandwidths when running the benchmark. For comparison, similar wired topologies in literature (such as DRAIN[13], Low-Cost / UberNoC[14], and Hermes[15]) report average latencies of between 15 and 35 clock cycles per 64 bytes.

They also showcase that latency is highly dependent on the adopted routing protocol. In this light, a co-design of protocol and network, such as the one enabled by our framework, is key. As an example, a strategy based on exponential backoff may not be well suited for a wide multicluster system, as it potentially incurs in a large number of collisions (and, therefore, retransmissions) when running memory intensive benchmarks such as STREAM.

[12] J. McCalpin, "STREAM: Sustainable Memory Bandwidth in High Performance Computers", available at http://www.cs.virginia.edu/stream/

[13] M. Parasar et al., "DRAIN: Deadlock Removal for Arbitrary Irregular Networks", HPCA 2020

[14] H. Farrokhbakht et al., "UBERNoC: Unified Buffer Power-Efficient Router for Network-on-Chip", NOCS 2019

[15] C. Iordanou et al., "Hermes: Architecting a Top-Performing Fault-Tolerant Routing Algorithm for Network-on-Chips", ICCD 2014

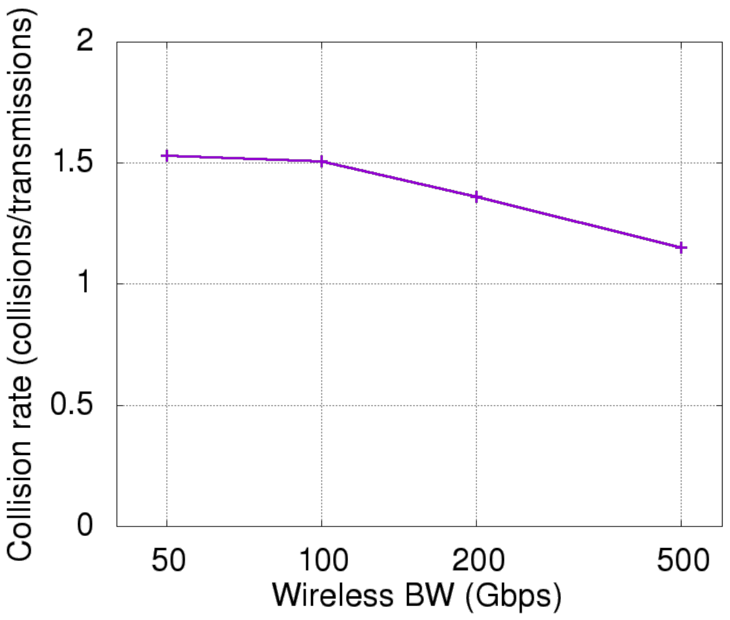

Figure 15: Wireless channel collision rate when running the STREAM benchmark and employing the exponential backoff protocol

We choose such architectural configuration as a test vehicle to showcase the scalability of our approach. Moreover, the considered benchmark is illustrative of a worst-case scenario, where the memory interconnect must support a very large number of data transfers between clusters and memory. Indeed, as displayed in Fig. 15, the collision rate when running STREAM is higher than one collision per transmission, and thus at the number of retransmissions has to be at least twice the number of transmissions.

Summarizing our exploration, Fig. 16 illustrates the exponential increase in STREAM runtime with the different wireless effective latencies obtained. Such a behaviour is a consequence of the exponential increase of transmission windows in presence of collisions of the backoff MAC algorithm.

To improve the effective latency of wireless communication, we are considering MAC protocols better suited for larger networks (such as token passing or Fuzzy-Token[16]). These medium access mechanisms would improve the utilization of the available wireless bandwidth, ultimately increasing performance.

[16] A. Franques et al., “Fuzzy-Token: an Adapive MAC Protocol for Wireless-Enabled Manycores”, DATE 2021

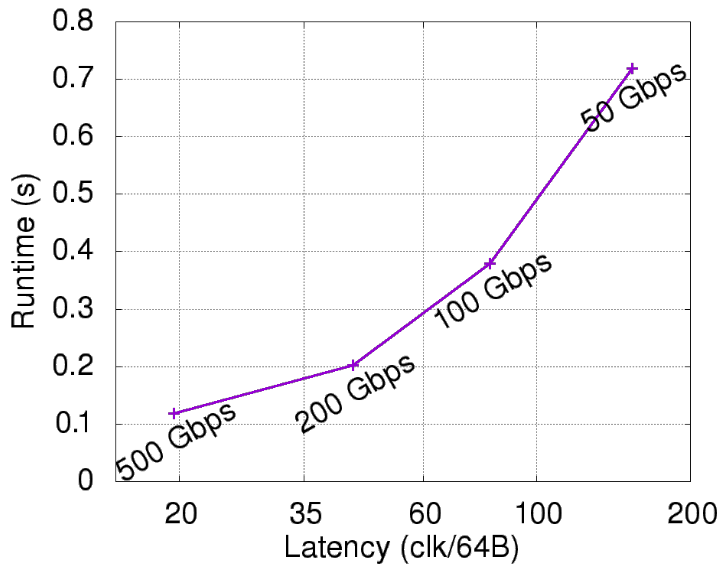

Figure 16: Runtime of STREAM benchmark using the enabled wireless channel and the exponential backoff protocol

3.4 Thermal exploration

Besides run-time performance, the use of wireless communication may also have a deep impact on how task mapping is performed. In particular, as high-performance communication between cores (or clusters) does not require physical proximity when employing wireless transmission, thermal considerations can be at the forefront when allocating computing tasks to resources.

To illustrate this effect, we consider in this section the thermal behavior of a multi-core system in two different scenarios:

1. In the first one, depicted on the left side of Figure 17, the communication between cores is done via physical wires. The workload is here mapped into adjacent cores in order to reduce communication latency.

2. In the second case (right side of Figure 17) the communication between cores is done via wireless links. Hence, workload mapping is scattered into cores that are physically located in different areas of the chip. This results in smaller regions with high power density (hotspots) and better thermal dissipation.

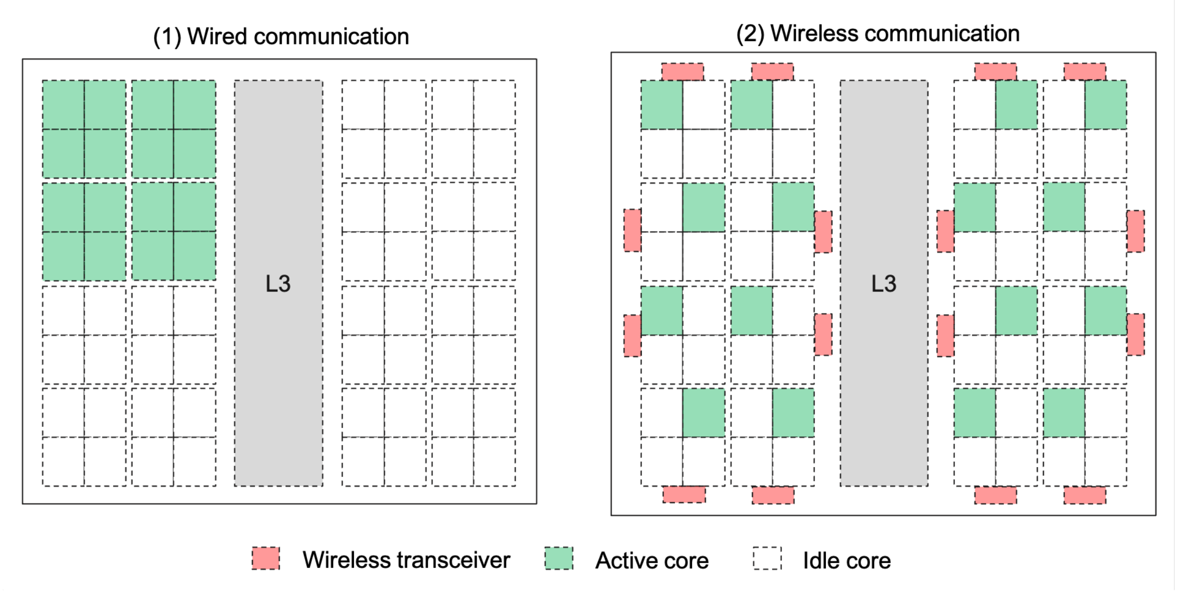

Figure 17: Multicore processor floorplans with wired and wireless communication

The power map of the multicore systems are modeled as follows:

- The cores are based on ARM Cortex-A75, a high-performance micro-architecture for the mobile market. The cores consume up to 2W and have a size of 1.75 mm^2. The power distribution of the cores is uniform across their area, including the power consumption of the L1 and L2 caches.
- The cores share a 10 MB L3 cache. The L3 occupies 39 mm^2 and has a power density of 50 mW/mm^2 in full usage.
- Each cluster of 4 cores share a wireless transceiver to communicate with other core clusters. The wireless transceivers are modeled based on their area/power values in full usage as reported in D3.2, assuming a uniform power distribution due to their low impact on the thermal dissipation across the chip. The total size of each wireless transceiver is 0.4 mm^2 and its power consumption is 22 mW.

The multicore processor is simulated using the compact thermal simulator 3D-ICE. The simulator constructs a mesh of 50 µm x 50 µm thermal cells and performs the thermal analysis of the system.

The temperature maps of the multicore processor under both scenarios (1) and (2) are shown is Figure 14. The results indicate that mapping workloads into cores that are scattered across the chip enables significant decrease in peak temperature compared to the case where active cores are concentrated in one high-power density area, with peak temperatures of 61°C in the wireless task mapping configuration, where tasks are scattered on cores in different SoC, with respect to 75°C when tasks are instead mapped close together.

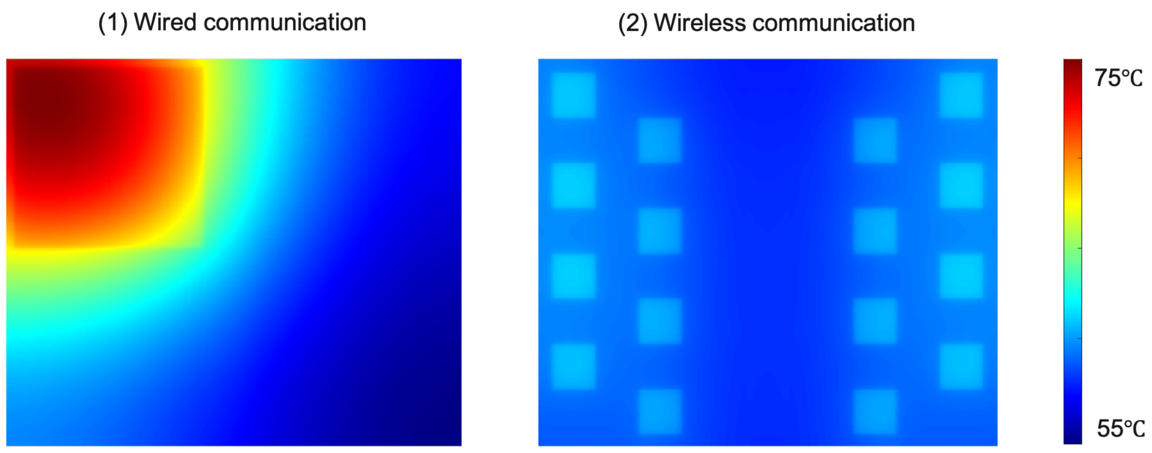

Figure 18: Temperature maps of multicore processor with wired (right) and wireless (left) task mapping strategies.

4 Conclusions and Perspectives

The deliverable details the activities in WP5, which have enabled the system simulation of AIMC-accelerated and wirelessly communicating systems, integrating tightly coupled resistive arrays and graphene-based nanoantennae.

To this end, the WP5 partners, have developed two new component models (dedicated to AIMC and wireless transmission emulation, respectively) as part of the gem5-X full system simulation framework. Both components can be easily integrated in complete hardware/software systems, targeting realistic software stacks (including applications and operating systems) as well as complex multicore architectures, which are key for the performed system design space explorations.

We plan to release the code base of these gem5-X extensions for Deliverable 5.4.