Mont-Blanc 2020: Towards Scalable and Power Efficient European HPC Processors

Adrià Armejach*† Bine Brank‡ Jordi Cortina§ François Dolique• Timothy Hayes◇ Nam Ho‡
Pierre-Axel Lagadec▷ Romain Lemaire• Guillem López-Paradís*† Laurent Marliac▷ Miquel Moretó*†
Pedro Marcuello§ Dirk Pleiter‡ Xubin Tan§ Said Derradji▷

*Barcelona Supercomputing Center †Universitat Politècnica de Catalunya
‡Forschungszentrum Juelich, Juelich Supercomputing Centre §Semidynamics Technology Services, SL
•Univ. Grenoble Alpes, CEA, List, F-38000 Grenoble, France ◇Arm ▷Atos

***Abstract*—The Mont-Blanc 2020 (MB2020) project has triggered the development of the next generation industrial processor for Big Data and High Performance Computing (HPC). MB2020 is paving the way to the future low-power European processor for exascale, defining the System-on-Chip (SoC) architecture and implementing new critical building blocks to be integrated in such an SoC. In this paper, we first present an overview of the MB2020 project, then we describe our experimental infrastructure, the requirements of relevant applications, and the IP blocks developed in the project. Finally, we present our emulation-based final demonstrator and explain how it integrates within our first generation of HPC processors.**

I. Introduction

There is a stringent need to process the ever increasing amounts of information required to tackle High Performance Computing (HPC) challenges. In many research domains, exascale machines are required to further advance the field. However, the challenges to reach exascale computing are huge and require careful co-design between application requirements and technological choices. The Mont-Blanc 2020 (MB2020) project aims to be the stepping stone towards the development of low-power European processors for exascale. MB2020 lays the foundation for a European consortium aiming at delivering a processor co-designed for energy efficiency targeting HPC and server workloads. A first generation product is scheduled in the 2022 time frame.

Europe owns a major fraction of the HPC applications used worldwide, leading with more than a fifty percent share in chemistry and material science domains. It is also one of the biggest producers of data in the world. Therefore, undertaking the design and production of a European processor is a step in the right direction to make a significant contribution to strengthen European industry and science in the HPC domain and in other sectors. Development is driven by both scientific and industrial needs, addressing critical aspects like usability and efficiency, while also being economically viable due to the existence of market volume. Finally, the end goal is to generate industrial capacity in processor design that will enable the creation of a complete European value chain in scientific domains and industrial sectors relying on high performance processing [1].

In October 2011, the first Mont-Blanc project started to pioneer the development of energy-efficient HPC systems using embedded and mobile devices by building a full energy-efficient HPC prototype with mobile SoCs [2]. The second iteration of the project enabled support for ARMv8 64-bit processors, provided significant advances towards a competitive HPC software stack, and the initial design of the Mont-Blanc exascale architecture. The third iteration of the project further improved the software ecosystem and simulation infrastructures, and delivered a prototype HPC system based on Marvell ThunderX2 processors that later became a commercial product.

MB2020 builds on this foundation to design, implement, and demonstrate key intellectual property (IP) blocks. In particular the main objectives of the MB2020 project are:

- Define a low-power System-on-Chip (SoC) implementation for exascale, with built-in security and reliability features.
- Introduce technology innovations such as the Arm Scalable Vector Extensions (SVE) [3] and High Bandwidth Memory (HBM) to improve efficiency for real-world applications.
- Develop, validate and demonstrate key IP modules such as an on-die Network-on-Chip (NoC) and a high-bandwidth low-latency memory hierarchy solution.
- Provide a working prototype exhibiting these key components with an FPGA-based demonstrator, following a co-design approach based on real-world applications.
- Explore the reuse of these building blocks to serve other markets than HPC.

The ambition of the consortium is to quickly industrialize the proposed research. Follow-up industrial projects under the European Processor Initiative (EPI) are underway to manufacture by 2022 a first generation of HPC processors to demonstrate the innovative components developed in MB2020.

II. MB2020 Project Overview

In this section we present a general overview of the proposed architecture, including a high-level view of the SoC. We later describe the steps MB2020 follows to go from application requirements that shape the specification of the target IP blocks, to the infrastructure needed to evaluate on the FPGA-based emulator.

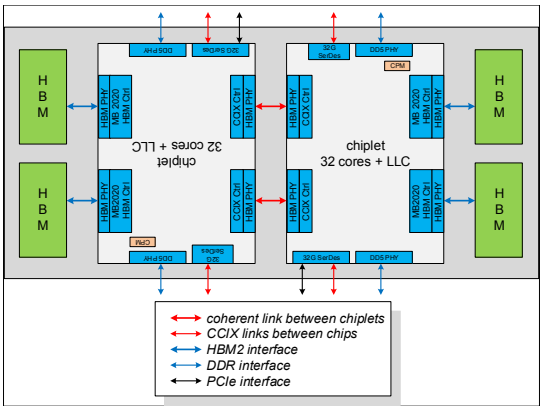

Fig. 1: Overview of the MB2020 SoC architecture.

A. MB2020 Proposed Architecture

In order to achieve fast market adoption, the processor needs to be designed using an instruction set architecture (ISA) backed with a strong software ecosystem. Capitalizing on the ecosystem developed in previous Mont-Blanc iterations, the project choice is to use the ARMv8 ISA. Given current trends, we believe the Arm ecosystem will deliver all the system software and applications required for a successful adoption in HPC as well as other markets.

There are three key challenges to achieve the desired performance-energy trade-offs: (i) a high-bandwidth low-power memory system with sufficient capacity and bandwidth; (ii) a scalable on-die NoC able to supply enough bandwidth to the processing units; and (iii) a processing unit able to deliver high floating-point operations (Flops) per second per Watt.

Figure 1 shows a high-level overview of the modular MB2020 architecture. The compute elements are low-power Arm cores with vector units implementing the SVE vector extension. They deliver high throughput of floating-point operations with a high Flops per second per Watt ratio. The project aims to design a low-power high-bandwidth NoC based on the AMBA 5 Coherent Hub Interface (CHI) architecture [4]. Finally, we propose an interposer-based HBM solution to reduce the energy-per-bit for off-die memory accesses. To achieve this, we designed an HBM controller.

B. Steps in the MB2020 Project

To achieve the objectives of the project, the consortium has four main research lines:

- We develop a comprehensive infrastructure to be able to deploy a working demonstrator of the whole MB2020 SoC architecture into an FPGA emulator.
- To ensure a suitable design, we employ a co-design approach where the hardware is designed based on real-world HPC application requirements. Therefore, we select a set of representative applications and derive multiple requirements for SVE, the NoC, the memory system, and reliability.
- Based on the requirements, we design and implement the two main hardware IP blocks: an AMBA CHI-compliant NoC and an HBM controller.
- The actual final demonstration platform that will emulate a full-sized NoC with multiple HBM controllers.

The next four sections describe these efforts in detail.

III. Project Infrastructure

In the MB2020 project, we require an infrastructure to evaluate workload performance on the developed IP blocks. In addition, these workloads make use of novel technologies, such as SVE, for which no commercially available hardware was available at the project's start. Therefore, we devise an infrastructure that enables executing real-world HPC applications with SVE instructions on an FPGA emulator that models a full-sized NoC with multiple HBM controllers.

This infrastructure consists of three main components: (i) an architectural full-system simulator with SVE support; (ii) a tracing methodology able to abstract away the processing core behaviour; and (iii) a synthesizable RTL traffic injector that consumes these traces and exchanges messages with the CHI interface of the NoC cross-points.

For the first component, we extended the gem5 simulator [5] to have SVE support. Additionally, we further extend the in-order and out-of-order processor models to support a new vector register file, vector execution units, changes to the load/store queue, and new concepts such as first faulting loads and predicated execution [3]. These changes, in part MB2020 contributions, are publicly available in the gem5 repository.

For the second component, we devise a tracing methodology that captures application behavior by recording memory misses at the L1 cache and by encoding the memory and compute stalls the core is experiencing. By replaying these traces with the RTL traffic injector, we can then inject representative traffic into the NoC based on real-world multi-threaded applications. The trace is generated using the SVE-enabled gem5, configured to mimic the target MB2020 SoC architecture of one chiplet (see Figure 1).

Finally, the RTL traffic injector models three aspects of a core: (i) models a private L2 cache with a full Finite State Machine (FSM); (ii) generates traffic towards the NoC; and (iii) responds to coherence traffic. Figure 2 shows the block diagram of the RTL traffic injector. The Trace Injector Manager reads the traces and send the requests to the emulated L2 cache. The L2 block is a simplified version of a realistic L2 cache with no data, since the main purpose of this logic block is to generate network traffic. However, it fully implements a MESI coherence protocol, as well as the support for exclusive operations (LDX/STX) through the implementation of Logical Processor (LP) monitors, as described in the AMBA 5 CHI specification [4]. The L2 block is fully parametrizable, including the size, associativity, number of banks, etc. Finally, the CHI Agent implements a fully coherent Request Node (RN) agent that generates requests to the NoC, as well as the responses to requests that arrive from the NoC.

This infrastructure enables emulation and evaluation of the two main IP blocks developed in the MB2020 project, the NoC and the HBM controller. By abstracting the core models with traces and using an RTL traffic injector, we can fit a fully sized NoC for 32 cores and 4 HBM controllers within the FPGA emulation platform.

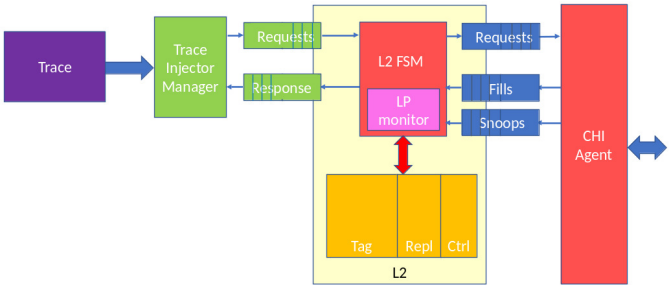

Fig. 2: RTL traffic injector architectural overview.

TABLE I: Selected applications in the MB2020 project.

Application	Computational pattern	Prog. language	Prog. model
Arbor	dense linear	C++	MPI + threading
HACCKernels	n-body methods	C++	OpenMP
HPCG	sparse linear	C++	OpenMP
KKRnano	dense linear	Fortran	MPI + OpenMP
Grid	structured grids	C++	MPI + OpenMP
MiniAMR [9], [10]	structured grids	C	MPI
RAJAPerf	set of HPC kernels	C++	OpenMP
SWFFT	spectral methods	C++/Fortran	MPI + OpenMP
XSBench	embarassingly parallel	C	OpenMP

IV. Application Requirements

We follow a co-design approach to ensure the final hardware design meets real-world HPC application requirements. To this end, we first select a set of applications representative of complex problems found in science and engineering. These applications drive the evaluation of the developed IP blocks (i.e. the NoC and HBM controller), stressing the target requirements of the MB2020 project (i.e. SVE, memory hierarchy, NoC, and reliability).

A. Application Selection

We gather applications currently running on a production machine such as the Curie supercomputer, and also those present in relevant benchmark suites, such as: (i) the Unified European Applications Benchmark Suite (UEABS) established by PRACE [6], (ii) the CORAL-2 benchmarks [7], or (iii) the Exascale Proxy Applications from the Exascale Computing Project (ECP) [8]. As a result, we obtain a broad list of 28 relevant HPC applications, including large applications and well-known representative mini-applications and kernels.

To make the final selection from this list, we define four sets of criteria to establish a project-wise selection methodology. With this methodology, we evaluate the previous list and select nine applications that fulfil the four criteria sets. Table I lists the selected applications and information of interest, including the associated main computational pattern, the programming language, and the programming model employed.

B. MB2020 SoC Architecture Requirements

SVE Requirements. We employ our gem5 infrastructure to derive SVE requirements. We evaluate systems with DDR and HBM technologies for different SVE vector lengths, from 128 to 2048 bits. We highlight the following requirements:

- The use of HBM technologies is mandatory to boost performance on common memory-bound benchmarks. DDR technologies, which do not scale with core count, fall short at delivering the necessary memory bandwidth.
- In terms of memory bandwidth per core, doubling the available bandwidth from 16 GB/s to 32 GB/s per core leads to small performance improvements with vector lengths of 256 and 512 bits. In addition, most modern systems can only sink about 20 GB/s per core of peak bandwidth. Therefore, we recommend 16 GB/s per core as it gives a good balance in terms of cost-performance.
- The 512 bits SVE configuration starts to be efficient with floating-point operations per byte (flops/byte) ratios greater than 1.5. However, it does not provide any performance improvement over 256 bits SVE for common flops/byte ratios in the 0.5 range. Therefore, we advocate for a vector length of 256 bits, since vector units are a major component in terms of area and power consumption.

Memory Hierarchy Requirements. We propose a methodology based on an analytical model that allows categorising applications according to their ability to exploit a multi-tier memory architecture. In particular, a tier optimized for large-capacity (DDR) and a tier optimized for high-bandwidth (HBM). We derive the following requirements for such a system:

- To maximize performance in terms of throughput of floating-point operations the use of high-bandwidth memory technologies is strongly recommended.
- Applications can utilise multiple memory tiers with data locality and data transfer managed in software. The letter should be accelerated by hardware, e.g. through DMA engines, to maximise data transfer rate and minimise involvement of processing cores such that data transfer and data processing can be optimally overlapped.
- The analysed applications indicate that the bandwidth to the large-capacity memory tier can be small compared to the high-bandwidth memory tier.

Network-on-Chip Requirements. We combine different tools, namely NocStudio and Platform Architect, to setup a strong methodology able to model a SoC that contains 36 mono-core clusters (including 4 spares), 4 HBM2 memory controllers, the Cache Coherence Controller, and the Last Level Cache, which constitute the equivalent of the Home Node agents that route the traffic. By using this platform we can derive requirements in a realistic setup in which NoC designers can compare with a reference design. We derive the following requirements:

- The NoC topology should allow for a homogeneous value of bandwidth and latency from all cores to the memory subsystem in order to avoid hidden NUMA effects.
- The coherence protocol should incur minimal traffic on the NoC to devote maximum bandwidth for the user payloads.
- Hardware performance counters must be provided to give users a good understanding of NoC resource usage.
- NoC links to/from the different memory controllers need to provide sufficient bandwidth to saturate the HBM stacks.

Reliability Requirements. RAS (Reliability, Availability and Serviceability) is an increasingly important topic in HPC to protect compute jobs that run for several days on many processors. We provide a bottom up description of the requirements starting at the component level, then the node, the rack and finally the whole machine:

- At the component level, all circuits (functional units and different data paths) must be protected. All data paths should be at least CRC protected to minimize Failure In Time (FIT).
- A monitoring unit should be present to be able to understand quickly where the error comes from and retire the faulty components to prevent fatal errors from occurring again.
- End to End retransmit protection schemes across components help a lot to improve the Mean Time To Failure (MTTF) and are therefore desired.
- Software can help mitigate errors via checkpoint and restart mechanisms when an uncorrectable error is detected.
- An exascale HPC system requires an aggressive focus on reliability. For our target processor with an exascale configuration our MTTF targets are 24 hours for Detectable Unrecoverable Error (DUE) and 720 hours for Silent Data Corruption (SDC).

V. MB2020 IP Blocks

A. Low-Power High-Bandwidth NoC

The NoC implementation is scalable and modular based on custom IPs (CrossPoint (XP), Snoop Filter, Home Node and Last Level Cache) which are connected and configured through an automated generation tool. It supports up to 128 high performance cores and is flexible enough to match the MB2020 target topologies with tailored number of instances of each IPs, link bandwidth, cache and Snoop Filter size and input buffers depth.

To speed up physical implementation, the diversity of IP versions should be limited. As an example, in the emulator reference configuration, the 36 XPs of the central area can be cloned as the same hard IP. The NoC implements the CHI protocol, which is natively used by Arm high performance cores. The coherency protocol between the cores is handled by the Home Node (HN) which is also the point of serialization where the ordering between requests from different agents is determined. It embeds a Snoop Filter to reduce coherence protocol overheads, and a Last Level Cache to limit data movement. Direct Memory and Direct Cache Transfers are used to reduce the read latency by permitting the memory and the cores to send data directly to the requesters.

The Snoop Filter avoids sending snoop messages to all RNs when checking whether they have a cached copy of a cache line. To reduce its memory size but keep a very low level of evictions initiated by the snoop filter, techniques such as twin lines, coarse mode and victim buffer can be implemented by the NoC generation tool. The twin lines implementation is used in the emulator to reduce the memory size by 22%.

A mesh topology is used due to its scalability. As there is a dependency between the different CHI channels, i.e. request

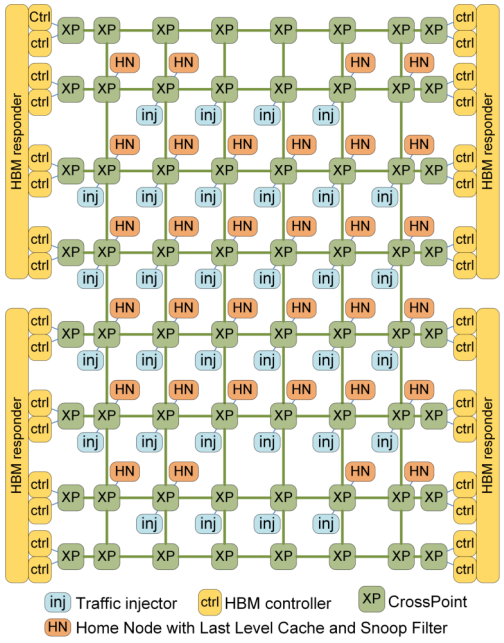

Fig. 3: The MB2020 NoC architecture.

(REQ), response (RSP), data (DAT) and snoop (SNP), it is necessary to make sure that they can flow independently of each other to prevent protocol deadlock. This independence is guaranteed using a dedicated mesh network for each channel. Multiple physical channel lanes of the same kind can be used between two XPs to increase the bandwidth. Custom algorithms are implemented for each channel using routing tables in each XP input port to guarantee the best load balancing and an efficient use of paths. The supported routing algorithms include Dimension ordered Routing or any kind of turn models. Multicasting of snoop messages is implemented not only at the Home Node output but throughout the network with a one hot encoding of the destination in order to optimize link usage.

In order for the NoC to manage cache coherency between clusters and memory access with throughput comparable to the aggregated cores bandwidth; the Home Node protocol engine, the associated Snoop Filter, and the Last Level Cache are split in 32 slices in the emulator configuration, as shown in Figure 3. In this configuration, each direction between XPs uses three REQ and RSP physical channel lanes, two DAT physical channel lanes, and one SNP physical channel; providing 128GB/s data bandwidth between two XPs and 768GB/s bisection bandwidth.

To meet the reliability requirements, all buses are protected with ECC throughout the mesh network and, in the Home Node, all the RAM modules are also protected. SECDED (Single Error Correction and Double Error Detection) protection is used for critical parts such as the Last Level Cache or input buffers whose data cannot be recovered. The Snoop Filter Presence Vector is protected with SED as it is possible to broadcast a snoop to recover the data. Techniques such as poison data are used to avoid raising an error when a cache line is corrupted but never used.

To allow traffic isolation and real time traffic for specific applications such as automotive, two Virtual Channels (VCs) can be implemented throughout the mesh network by the NoC generation tool. Any QoS level can be mapped on the priority VC if the associated messages need a very low/deterministic latency. This feature is not implemented in the emulator as it is not useful for HPC applications.

Power consumption must be a differentiator that allows to build sustainable exascale systems with a reasonable total cost of ownership. To achieve a very high efficiency, the NoC implements state-of-the-art power design techniques such as power islands and clock gating. The NoC mesh is clocked at 1 GHz and an ultra-low power mode with a clock running at 500 MHz and a lower voltage is implemented.

The biggest challenge was to provide a configurable NoC with a high message rate and low latency. This has been achieved thanks to the use of multiple slices of the Home Node, a 16MB Last Level cache to limit data movement outside the NoC, custom routing algorithms to decrease the congestion in the center of the mesh, snoop multicast throughout the network to optimize the links usage, and Direct Memory and Direct Cache Transfers to reduce the read latency.

B. HBM Controller

High Bandwidth Memory (HBM) is a high-performance interface for 3D-stacked DRAM that provides higher bandwidth and less energy consumption compared to conventional DRAM. High bandwidth is achieved across different independent interfaces called channels. Each channel provides access to a different set of DRAM banks, that could be independently clocked and managed in a way that commands to different channels do not interfere among them. Conventional implementations of HBM provide up to 8 128-bit channels, resulting in a total width of 1024 bits.

The HBM Controller in the MB2020 project is a digital circuit that manages the access to the data stored in the HBM DRAM. The main functionality of the implemented HBM controller is:

- Read requested data from the HBM and return it to the NoC.
- Write the data provided by the NoC into the HBM.
- Refresh HBM data to avoid losing the data stored in it.
- Detect and correct (if possible) data errors via ECC or parity-check.

Figure 3 shows 16 HBM controllers at the edges of the mesh. Each instance controls 2 channels of the HBM stack. On one side, the Memory Controllers are attached to the NoC and communicate with it through the AMBA CHI bus interface. On the other side, the HBM controller communicates with the HBM DRAM stacks through the PHY layer and the interposer.

The HBM controller specification meets the requirements for the MB2020 SoC and fulfils the following objectives:

- To provide data requests and responses to the NoC in a timely manner according to the selection policy specified by the user. The memory controller also provides fairness mechanisms to avoid starvation of requests.

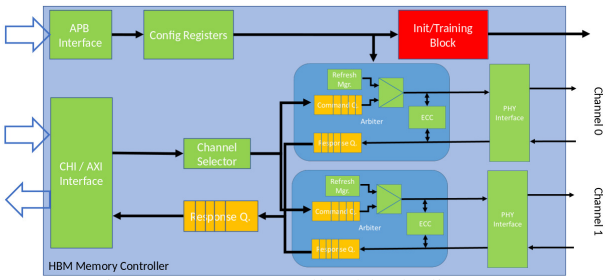

Fig. 4: HBM controller block diagram.

- To provide a high-sustained bandwidth to the NoC matching the one provided by the HBM data stack towards the memory controller.
- The memory controller is designed to be configurable and programmable. Configurable to be able to adapt its final design to different HBM standards (e.g. HBM vs HBM2), as well as support different sizes for its internal structures; and programmable to enable the user to choose between different scheduling policies for the incoming memory requests.

Figure 4 shows the high-level block diagram of the HBM controller. As channels work independently, the design has two separated blocks to manage the requests for each of the channels individually. The mapping of the addresses for each channel is configurable through the special registers of the HBM controller. The main block of the channel controller is the *Arbiter*, which selects among all the pending requests from the NoC for that channel and the refresh requirements. The selection policy is configurable, and it includes from a simple FIFO scheme, to a more complex scheme that is able to reorder the requests in order to minimize the total number of accesses and commands sent to the HBM DRAM. Memory accesses and refreshes are also converted to HBM commands and queued into the PHY interface. Additionally, all requests and responses, must compute and check the ECC/parity in case the HBM controller is configured to do it.

VI. Final Demonstrator Flow

MB2020 proposes an innovative platform to build HPC processors. The platform relies on two key IP blocks: a low-power high-bandwidth NoC and an HBM controller. The purpose of the final demonstrator flow is to verify these components in a real-world application environment and extract meaningful performance metrics.

A. Methodology based on hardware emulation

Before any silicon sample is produced, the only way to enable demonstration at hardware speed using real-world data from applications is to accelerate full SoC RTL simulations with an emulation platform. The project is relying on the Veloce Strato Emulator from Mentor Graphics [11]. The current design is targeting up to 8 emulation boards corresponding to a total of around 320 MGates. The emulation is cycle-accurate, typically $1000\times$ faster than simulation while preserving full visibility on the internal signals for debug.

The emulated design is built around the top SoC (*top_soc*) composed of the NoC (8×8 2D-mesh topology) and the 4

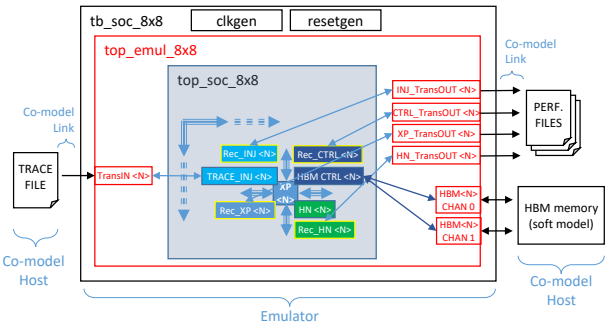

Fig. 5: Emulation platform overview

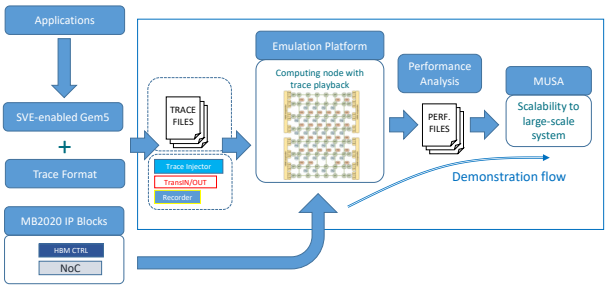

Fig. 6: Demonstration flow

HBM controllers (each composed of 4 blocks of 2 channels) as illustrated on Figure 5. The NoC itself gathers 64 XP routers and 32 Home Nodes (*HN*). Connected directly to the routers, 32 traffic injectors (*TRACE_INJ*) are used to mimic the Arm cores based on the trace files obtained according to the tracing methodology previously detailed. Additionally, recorder (*Rec_***) modules have been integrated to the top SoC in order to periodically record relevant performance counters from the different modules.

The top SoC is instantiated within a specific emulation top (*top_emul*). This top contains a set of transactors to enable communication between the emulated RTL and the external host system piloting the emulation infrastructure. To optimize emulator performance, the transactors for trace injection and performance recording are unidirectional with a flow control mechanism. On one side, they collect data from trace files to feed the injectors, on the other side, data from the recorders is transmitted to files stored for post-emulation analysis.

Finally a top test-bench (*tb_soc*) manages the clock and reset generations. In parallel of the emulated RTL, external hosts (co-models) deal with the files management. For the HBM, that are not part of the MB2020 design, an external soft model is also executed on the co-model host.

B. Flow description, objectives and RTL mapping results

The emulation platform is part of a full demonstration flow (Figure 6) to fulfil two objectives of the MB2020 project: *(i)* functional validation of key IP blocks (NoC and HBM controller) by achieving execution of full application traces, *(ii)* and extracting performance metrics collected during the emulation process. The recorder modules periodically receive performance counter values from the design and store them in performance files. Numerous metrics are recorded, such as: input or output bandwidth, latency for each transaction at injector level, transaction counters and queue load in the NoC, cache memory hit/miss at various levels, total execution time. Metrics are collected successively for each application.

The full design has been synthesized and successfully mapped on the emulator. The results (Table II) present the number of Look-Up Tables (*LUT*), Flip-Flops (*FF*) and memory blocks required at different level of hierarchy in the design. Most of hardware ressources are used by the NoC (75-80% of the design). The total used LUTs represent 7.84 boards, the FF usage is less constrained (equivalent to 5.78 boards).

TABLE II: Results for RTL mapping on the emulator.

Module	Instance	LUT	FF	Memory (kbit)
HN	32	661 801	349 059	1 552
XP	64	236 166	71 871	17
NoC	1	31 858 304	14 690 436	50 517
HBM_CTRL	4	1 077 136	482 568	1 424
RECORDER	128	647	329	0
TRACE_INJ	32	154 351	96 953	234
top_soc_8x8	1	41 189 019	19 765 348	63 705

VII. Conclusions

The outcome of the MB2020 project will be the basis for a new range of European HPC exascale systems and is expected to achieve market adoption and economic sustainability within the coming years. The first generation of HPC processors using this NoC is expected to be ready by 2022 in the context of the European Processor Initiative (EPI).

Acknowledgment

This work is supported by the European Community's Horizon 2020 Framework Programme under the Mont-Blanc 2020 project, grant agreement n. 779877.

References

[1] COM, "European Cloud Initiative - building a competitive data and knowledge economy in Europe," 2016.

[2] N. Rajovic, P. M. Carpenter, I. Gelado, N. Puzovic, A. Ramirez, and M. Valero, "Supercomputing with Commodity CPUs: Are Mobile SoCs Ready for HPC?" in *International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC '13, 2013.

[3] N. Stephens, S. Biles, M. Boettcher, J. Eapen, M. Eyole, G. Gabrielli, M. Horsnell, G. Magklis, A. Martinez, N. Premillieu, A. Reid, A. Rico, and P. Walker, "The Arm Scalable Vector Extension," *IEEE Micro*, 2017.

[4] Arm, "AMBA CHI Issue E Specification."

[5] J. Lowe-Power, A. M. Ahmad, A. Akram, M. Alian, R. Amslinger, M. Andreozzi, A. Armejach, N. Asmussen *et al.*, "The gem5 simulator: Version 20.0+," *arXiv preprint arXiv:2007.03152*, 2020.

[6] Unified European Applications Benchmark Suite. [Online]. Available: https://repository.prace-ri.eu/git/UEABS/ueabs/

[7] Coral 2 Benchmarks. [Online]. Available: https://asc.llnl.gov/coral-2-benchmarks

[8] ECP Proxy Applications Suite. [Online]. Available: https://proxyapps.exascaleproject.org/ecp-proxy-apps-suite/

[9] A. Armejach, H. Caminal, J. M. Cebrian, R. González-Alberquilla, C. Adeniyi-Jones, M. Valero, M. Casas, and M. Moretó, "Stencil codes on a vector length agnostic architecture," in *International Conference on Parallel Architectures and Compilation Techniques, PACT*, 2018.

[10] A. Armejach, H. Caminal, J. M. Cebrian, R. Langarita, R. González-Alberquilla, C. Adeniyi-Jones, M. Valero, M. Casas, and M. Moretó, "Using Arm's Scalable Vector Extension on Stencil Codes," *J. Supercomput.*, vol. 76, no. 3, pp. 2039–2062, 2020.

[11] Mentor Graphics. (2020) Veloce strato platform. [Online]. Available: https://www.mentor.com/products/fv/emulation-systems/veloce-strato