Gem5-X: A Many-core Heterogeneous Simulation Platform for Architectural Exploration and Optimization

YASIR MAHMOOD QURESHI and WILLIAM ANDREW SIMON, Embedded Systems Laboratory (ESL), Swiss Federal Institute of Technology Lausanne (EPFL), Switzerland
MARINA ZAPATER, Embedded Systems Laboratory (ESL), Swiss Federal Institute of Technology Lausanne (EPFL), Switzerland and School of Engineering and Management Vaud (HEIG-VD), University of Applied Sciences Western Switzerland (HES-SO), Switzerland
KATZALIN OLCOZ, Complutense University of Madrid (UCM), Spain
DAVID ATIENZA, Embedded Systems Laboratory (ESL), Swiss Federal Institute of Technology Lausanne (EPFL), Switzerland

The increasing adoption of smart systems in our daily life has led to the development of new applications with varying performance and energy constraints, and suitable computing architectures need to be developed for these new applications. In this article, we present gem5-X, a system-level simulation framework, based on gem-5, for architectural exploration of heterogeneous many-core systems. To demonstrate the capabilities of gem5-X, real-time video analytics is used as a case-study. It is composed of two kernels, namely, video encoding and image classification using convolutional neural networks (CNNs). First, we explore through gem5-X the benefits of latest 3D high bandwidth memory (HBM2) in different architectural configurations. Then, using a two-step exploration methodology, we develop a new optimized clustered-heterogeneous architecture with HBM2 in gem5-X for video analytics application. In this proposed clustered-heterogeneous architecture, ARMv8 in-order cluster with in-cache computing engine executes the video encoding kernel, giving 20% performance and 54% energy benefits compared to baseline ARM in-order and Out-of-Order systems, respectively. Furthermore, thanks to gem5-X, we conclude that ARM Out-of-Order clusters with HBM2

This work has been partially supported by the ERC Consolidator Grant COMPUSAPIEN (GA No. 725657), the EC H2020 WiPLASH (GA No. 863337), the EC H2020 RECIPE (GA No. 801137), the Spanish CM (S2018/TCS-4423), the EU FEDER, and the Spanish MINECO (GA No. RTI2018-093684-B-I00).
Extension of conference paper: Y. M. Qureshi, W. A. Simon, M. Zapater, K. Olcoz and D. Atienza, "GEM5-X: A GEM5-Based System Level Simulation Framework to Optimize Many-core Platforms," *Proceedings of the High Performance Computing Symposium (HPC'19)*. This article extends the conference paper by (1) adding the heterogeneous compute core capability along with core clustering in gem5-X; (2) analyzing the HBM2 memory model in gem5-X with STREAM benchmark, giving insights into memory bandwidth scaling trends; (3) introducing a two-step architectural exploration and optimization methodology, capitalizing on our new gem5-X platform; (4) exploring and optimizing the architecture for video analytics application with three case-study scenarios, resulting in a fully functional heterogeneous architecture with core clusters for different kernels of the application along with HBM2 as the main memory.
Authors' addresses: Y. M. Qureshi, W. A. Simon, and D. Atienza, Embedded Systems Laboratory (ESL), Swiss Federal Institute of Technology, 1015 Lausanne, Switzerland; emails: {yasir.qureshi, william.simon, david.atienza}@epfl.ch; M. Zapater, Embedded Systems Laboratory (ESL), Swiss Federal Institute of Technology, 1015 Lausanne, Switzerland, School of Engineering and Management Vaud (HEIG-VD), University of Applied Sciences Western Switzerland (HES-SO), 1401 Yverdon-les-Bains, Switzerland; email: marina.zapater@heig-vd.ch; K. Olcoz, Complutense University of Madrid (UCM), 28040 Madrid, Spain; email: katzalin@ucm.es.

1544-3566/2021/07-ART44 $15.00
https://doi.org/10.1145/3461662

are the best choice to run visual recognition using CNNs, as they outperform DDR4-based system by up to 30% both in terms of performance and energy savings.

CCS Concepts: • **Hardware → Emerging architectures**; **Emerging simulation**; • **Computer systems organization → Heterogeneous (hybrid) systems;**

Additional Key Words and Phrases: Many-core, architectural exploration, gem5, in-cache, HBM, heterogeneous architectures, cluster

ACM Reference format:
Yasir Mahmood Qureshi, William Andrew Simon, Marina Zapater, Katzalin Olcoz, and David Atienza. 2021. Gem5-X: A Many-core Heterogeneous Simulation Platform for Architectural Exploration and Optimization. *ACM Trans. Arch. Code Optim.* 18, 4, Article 44 (July 2021), 27 pages.
https://doi.org/10.1145/3461662

1 INTRODUCTION

The rapid growth of online services and increasing adoption of digital products have influenced all spheres of our lives. This increased digitization of the society has led to the emergence of new applications, which are deployed all the way from **High Performance Computing (HPC)** systems and cloud servers to mobile devices. Consequently, this situation has led to data centers consuming 0.8% (200 TWh) of the global energy in 2019 [24]. These new emerging applications, such as video analytics [2], autonomous driving, natural language processing, content recommendation [37], bioinformatics and genome sequencing [49], have different performance/energy requirements and are deployed on a variety of different platforms. To optimize these performance/energy constraints, a system-level simulator is required that is capable of simulating multi-threaded applications in a full-system environment with a complete **operating system (OS)** on a heterogeneous multi-core system. Furthermore, the simulator should also be capable of a detailed system-level profiling to identify the bottlenecks, therefore, helping in designing strategies and architectures to alleviate these bottlenecks.

In this article, we present gem5-X ("a gem5-based full-system simulator with architectural eXtensions") and demonstrate its capabilities to seamlessly analyze multi-threaded applications, enable various architectural extensions both for the compute and memory sub-systems, and explore various architecture parameters to have an overall performance-energy optimized architecture. The video analytics application is used as a case study to demonstrate the architectural and system-level characterization capabilities of the gem5-X framework. Moreover, gem5-X is generic enough to be used for detailed architectural analysis in any other application. Then, we present an optimization methodology based on local exploration and optimization for each individual kernel and then a global optimization for the whole application to have complete optimized system. Each kernel can have its own optimized architecture, which might be different from another kernel, but collectively all those locally optimized kernels make up the complete optimized application. Hence, we develop a heterogeneous system-level architecture and ensure that all the kernels and system components are working coherently with each other.

As previously mentioned, real-time video analytics, dubbed as the next-generation "killer-app" [2], is used in a wide range of domains, from video surveillance for security and monitoring, safety on construction sites, traffic monitoring and thermal monitoring to identify patients with fever, as well as for fire hazards [35, 57]. Real-time video analytics is also used for autonomous drone navigation and rescue drone missions, e.g., during earthquakes, floods, or rescuing people at sea [25, 36]. Autonomous drones are also being deployed for deliveries of parcels by many companies around the globe. The emerging market of smart autonomous cars, along with their **Advance**

Driver-Assistance Systems (ADAS), use video analytics as one of the core components of the system to detect obstacles, traffic signs, and signals to navigate the car accordingly [45, 46]. Hence, this "killer-app" is being deployed all the way from low-power edge devices to high-performance cloud servers.

Video analytics is a combination of two kernels, namely, video processing (or video encoding) along with image classification and detection. The video encoding kernel has to meet the performance and **Quality-of-Service (QoS)** constraints of processing at 24 **frames-per-second (FPS)** for a seamless user viewing experience. Image classification has real-time constraints in real scenarios such as in surveillance to identify and alert of a potential security breach or safety alert on a construction site, collision avoidance in drone navigation, and safety features and decision processing in autonomous cars and ADAS systems. These different use-cases of video analytics on different architectures, such as surveillance in the cloud and drone navigation on the edge, require to have a performance and energy-optimized architecture while meeting the real-time constraints. The optimized architecture will enable longer battery life for edge devices and lower energy costs as well availability of resources to serve more users on the cloud. Hence, a system-level simulator is required to optimize the performance and energy of the complete system for multi-threaded applications on different architectures, either on the edge or in the cloud.

The main contributions of the article are as follows:

- We present the new gem5-X simulation platform, capable of simulating heterogeneous compute cores clusters and accelerators along with heterogeneous memory types such as the traditional DDR4 and 3D stacked memories.
- The 3D stacked **High Bandwidth Memory v2 (HBM2)** memory model in gem5-X is updated with improved interleaving to uniformly distribute the memory traffic. It is then analyzed with the STREAM benchmark, giving insight into memory **bandwidth (BW)** scaling and the potential BW bottlenecks in the system.
- A two-step architectural exploration and optimization methodology, capitalizing on our new gem5-X platform, is presented and showcased for a complex application like video analytics, composed of two kernels, namely, video encoding and image recognition using CNNs.
- We explore and optimize the architecture with video analytics application as a case-study, using the two-step methodology. We demonstrate a fully functional heterogeneous architecture with ARMv8 in-order cores with an in-cache computing engine called BLADE [55] for video encoding, along with ARMv8 **Out-of-Order (OoO)** cores on a separate cluster in the system for visual recognition tasks. For video encoding kernel, we achieve 20% and 54% performance improvement when using ARM in-order cores with BLADE compared to baseline ARM in-order and OoO core system, respectively. Additionally, replacing the DDR4 by HBM2 further improves the performance and energy efficiency by 10%. In the heterogeneous system along with HBM2, we achieve both performance and energy benefits of up to 30% in comparison to an equivalent system with DDR4, which in fact cannot meet the performance requirements for visual recognition tasks on OoO cores.

2 RELATED WORK

State-of-the-art complex applications deployed across the range of compute devices from edge devices to servers in the cloud require new innovative and heterogeneous architectures to run optimally in terms of performance and energy efficiency [54]. Full system architectural simulators are required for fast architectural exploration with accurate estimations of performance and energy to give an insight during the initial design phase by enabling early deployment of the application on the simulation platform and reduce the time-to-market of new products [14, 59].

Quite a lot of research has gone into developing architectural simulators, but each has its own shortcomings. Sniper [12] is a multi-core simulator with fast turnaround time, but only supports traditional x86 architectures. Simics [33] is another architectural simulator enabling applications to run on different hardware platforms. It is combined with Simflex [18] for timing information, but is limited to SPARC architectures only. Gem5 [9] is a **full-system (FS)** architectural simulator being widely used both in academia and industry, as it supports multiple ISAs, such as x86, ARMv7 ARMv8, MIPS, and ALPHA. In addition to a variety of ISAs, it also supports different CPU models for these ISAs, such as atomic, in-order, and OoO CPU models, as well as multiple caching protocols and coherences. On the memory side, it supports many traditional and emerging memories. Further, gem5 supports FS simulations via different Linux-based operating systems such as Ubuntu and Android, enabling applications to run as they would on a real platform. However, gem5 is quite slow with long simulation turnaround times. Also, not all configurations in gem5 are tested, and many models do not work in conjunction with others straight **out-of-the-box (OOB)**.

Our initial version of gem5-X [50] was a simulation platform that enhanced gem5 with OOB architectural extensions and new memory models. It enabled seamless FS simulation with profiling support to analyze the speedup and gains of the functional blocks within the application as a result of new architectural extensions and optimizations. It also provided file-sharing support between the host machine and the simulated system using **workload automation (WA)**, which is built into the Linux kernel provided for gem5-X. Therefore, it reduced the simulation turnaround time when debugging the application on a novel architecture within gem5-X.

In this new version of gem5-X, we support heterogeneous architectures, such as in-order and OoO cores along with custom accelerators like an in-cache computing engine [55]. In addition to heterogeneity at the compute side, it also supports heterogeneous memory types like DDR4 alongside 3D stacked HBM2, thus enabling a highly heterogeneous system with full Linux stack. To the best of our knowledge, this is the first work that simulates a complete Linux-based system with clustered heterogeneous compute cores (in-order and OoO) with in-cache computing engine along with 3D stacked HBM2 memory. In this article, we also perform the **bandwidth (BW)** analysis on HBM2 memory model in the new gem5-X using the STREAM benchmark [34] to validate that it provides the required BW in comparison to DDR4, as will be discussed in Section 4.3.

With increasing growth and popularity of cloud-based services, many of the emerging workloads can be deployed either on the cloud or on HPC systems. Gem5-X supports **virtual machines (VMs)** that can be deployed in the cloud infrastructure, as discussed in Reference [44]. It can also be used to explore and optimize architectures for HPC workloads. In particular, the work in Reference [49] demonstrates the use of gem5-X in enabling architectural exploration for the next-generation of genome sequencing HPC workloads.

3 VIDEO ANALYTICS APPLICATION

The gem5-X simulation framework supports full Linux stack and is generic enough to run and optimize any multi-threaded application, but to demonstrate its capabilities, a real-time video analytics application is used as a case-study in the context of this article. The choice of this particular application is due to two reasons. First, video analytics is deployed and used in different spheres of our daily life, e.g., in surveillance for safety and security, drone navigation, and autonomous cars. These different scenarios also present a challenge to system architectures because of different compute capabilities and energy constraints of the systems, all the way from the edge device to the cloud servers. The second reason for using video analytics as a case-study application is because of the complexity of the application itself and the fact that it is composed of two distinct kernels—video encoding and image classification using CNNs. This gives us the opportunity to

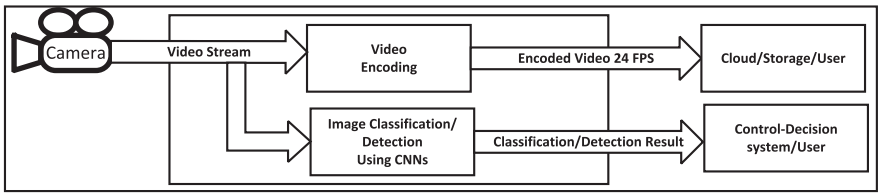


Fig. 1. Video analytics application composed of two kernels running concurrently, i.e., video encoding using Kvazaar [60] and image classification/detection using CNNs.

Table 1. Video Analytics Application Scenarios

Case	Video Resolution (pixels)	Image Classification (FPS)
Surveillance: Construction Site/Pedestrians [35, 57]	640 × 480	5
Drone Navigation [25, 36]	300 × 200	10
Autonomous Driving and ADAS [45, 46]	640 × 480	15

exploit and demonstrate the capabilities of gem5-X in optimizing various memory and compute sub-systems to have an overall optimized architecture.

3.1 Video Analytics Application Structure

As depicted in Figure 1, real-time video analytics consists of two kernels running in parallel alongside each other, namely, video encoding/processing and image classification and detection [32, 51]. Video encoding is used to encode the video stream and transmit it to the end-user or to store it in the cloud, e.g., in a security video surveillance system, the video is being streamed to a control room or stored in the cloud [35]; in case of a drone navigation system, it is being transmitted to the drone pilot [25]. In addition, to video encoding, the video frames from the video stream are being analyzed simultaneously by the image classification application. CNN-based image classification may be used for facial recognition, counting people, and potential hazard on a construction site in case of surveillance application [57]. It is also used for obstacle detection in drone navigation [25, 36] and autonomous cars [45, 46]. In a drone rescue mission, it can be used by drones to identify people at sea, in flood, or in the rubble during an earthquake. In essence, both the video encoding and image classification needs to run side-by-side for a fully functional video analytics. The video encoding needs to process 24 FPS for a seamless user experience. Simultaneously, the image classification needs to classify a number FPS depending on the application scenario. Table 1 summarizes the various application scenarios of video analytics application with different resolutions for video encoding and image classification/detection rate.

In the following sections, we will present the video encoding and image classification kernels, both of which together make up the video analytics.

3.2 Video Encoding

Video encoding is an essential part of video analytics. It also has significant importance in online video streaming services, which contributed 58% of downstream internet traffic in 2018 [53], hence, the need for a performance-energy-optimized video encoding application. For this purpose, Kvazaar [60], a state-of-the-art open source **High Efficiency Video Coding (HEVC)** application,

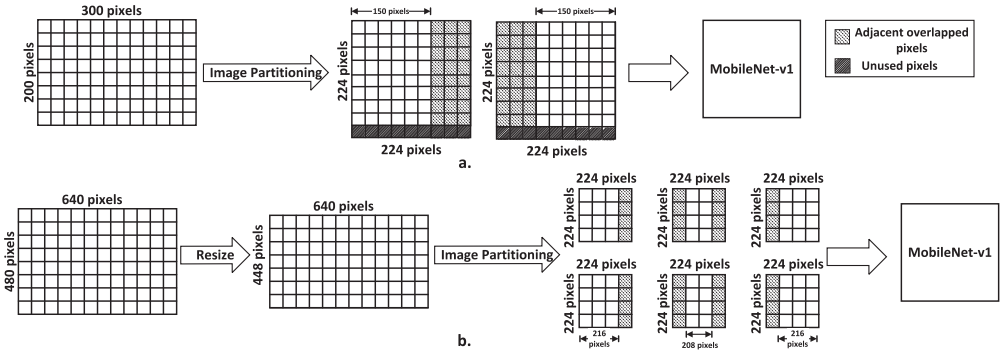


Fig. 2. Image partitioning and resize for input image resolution of (a) 300x200 pixels, (b) 640x480 pixels.

compliant with H.265 coding standard, is used as real-time video encoding kernel. HEVC offers twice the compression of its predecessors, but at the cost of significantly increased computational cost [10].

In HEVC encoder, the most complex block is the motion estimation of the video, which plays a critical role in compression (and therefore, bandwidth) and quality. The goal of encoding is to serve videos to users in real-time, i.e., achieving a sustained frame rate of 24 FPS, regardless of the video resolution. Real-time HEVC encoding is achieved by means of thread-level parallelization of different blocks. In this work, we choose Kvazaar, as it comes with a wider range of parallel processing capabilities. Kvazaar is very well optimized for software, leaving limited headroom for further software-based optimization.

As discussed in Reference [50], FIR filter and the **Sum of Absolute Transform Differences (SATD)** are the two dominating blocks in Kvazaar that represent 21% and 26% of overall instructions executed, respectively. The remaining 53% of the computation is spread in chunks of less than 10% (in average 7%) instructions in other blocks. Hence, we will focus on defining a suitable architecture with gem5-X to optimize the execution of the FIR filter and SATD block in Kvazaar.

3.3 Image classification using CNNs

The video analytics includes video encoding, which is either sent to the end-user or stored in the cloud, and CNN-based visual recognition, as discussed in References [13, 32, 51]. CNN-based image recognition is now becoming a standard in both the industry and academia for computer vision tasks because of the impressive results they have achieved [30]. All the case study scenarios of surveillance, drone navigation, and ADAS use CNNs for the visual recognition part of the application, as discussed in References [25, 57], and [46], respectively. Because of ample research in CNN architectures, there are a variety of CNNs to choose from [8]. The choice to deploy a particular CNN depends on various factors such as accuracy, inference time, memory requirements, computation complexity, and the size of the network. As the video analytics scenarios presented in Table 1 vary from being deployed on the edge device like in drone navigation all the way up to highend cloud servers like in a surveillance system. Therefore, the CNN we select should meet the memory, computational complexity, and performance requirements in all the scenarios.

MobileNet [20] is an efficient CNN architecture that can be deployed on the edge as well as in the cloud. It has a Top-1 accuracy of 70.4% on the Imagenet benchmark, which is better than other complex popular CNNs, such as GoogleNet, Alexnet, and Squeezenet. MobileNet-v1 is also quite small in size and complexity, and it is feasible to be deployed on the edge node. MobileNet

has previously been used in surveillance [57], drone navigation [26, 61], and autonomous cars and ADAS [29]. As most of the edge nodes are based on ARM architectures, we use the **ARM Compute Library (ACL)** [5] framework to deploy MobileNet. The limitation of MobileNet is that it can only process image sizes of up to 224x224 pixels, smaller than the case-study scenarios listed in Table 1. Hence, for these images to be processed by MobileNet, we perform image partitioning and resizing (if required) as shown in Figure 2. For the 300x200 pixel frame, we partition it into two 224x224 images. Since, the resolution of the resulting two 224x224 images is greater than the original image, we use the extra pixels to overlap the split images horizontally. This overlap allows us to detect objects that would have otherwise been split between images. The unused pixels at the bottom of the images are ignored and filled with black, as shown in Figure 2(a). For the frame resolution of 640x480, we first resize it to 640x448 pixels, so it can evenly be distributed vertically to 224 pixel size. After that, we partition it to six 224x224 pixel images, with the extra pixels being used to overlap the adjacent images to avoid splitting of objects between images, as shown in Figure 2(b). The partitioned images are then sent to MobileNet-v1 for image recognition.

MobileNet has been used in our case study, as it is a small, fast, and high accuracy CNN, which makes it feasible to be used both on the energy-constrained edge device, as well as on the servers in the cloud. It has also previously been used in our case-study scenarios in Table 1. However, any other CNN can be used instead of MobileNet, but the methodology we will use in the following sections will remain the same.

4 GEM5-X SIMULATION PLATFORM

The proposed gem5-X simulation platform [50] is the enabler for the architectural exploration and optimization of our case-study real-time video analytics application (cf. Section 3). The platform is composed of compute cores and memory devices. It simulates an ARMv8 64-bit ISA FS simulation with Linux kernel 4.3 and Ubuntu 16.04/18.04 as the OS. It also has application profiling support provided by the gperf profiler [17], with minimal overhead, to identify bottlenecks within the application executing in the simulator. In addition, gem5-X also supports 9P [28] over virtual IO [42], for fast modification and sharing of files between the simulated and host system.

The video analytics application has both high compute and memory requirements. Hence, we will now look into the compute and memory sub-systems of the gem5-X platform, as shown in Figure 3, with different exploration parameters highlighted in red.

4.1 Compute Sub-system

Gem5-X enables different strategies and aspects of the compute sub-system.

- **Type of Cores:** Gem5-X supports ARMv8 64-bit energy-efficient in-order cores as well as high-performance **out-of-order (OoO)** cores, both with a validation error of up to 4% as shown in our previous work [50]. We have added support for heterogeneous architecture simulation allowing both in-order and OoO cores to be simulated simultaneously, with different applications or application kernels being allocated efficiently to different core types to maximize performance, as well as energy efficiency, as described in Section 6.3.1.
- **Clustering:** Gem5-X enables clustering of the compute cores, both for heterogeneous as well as homogeneous core types. Each cluster has it own **last-level-cache (LLC)**. System architects can further expand the cluster implementation with each cluster having its own independent clock. Clustering enables independent applications or independent kernels of the application that do not share resources with other kernels of the application to execute on independent multi-core cluster with its own LLC.

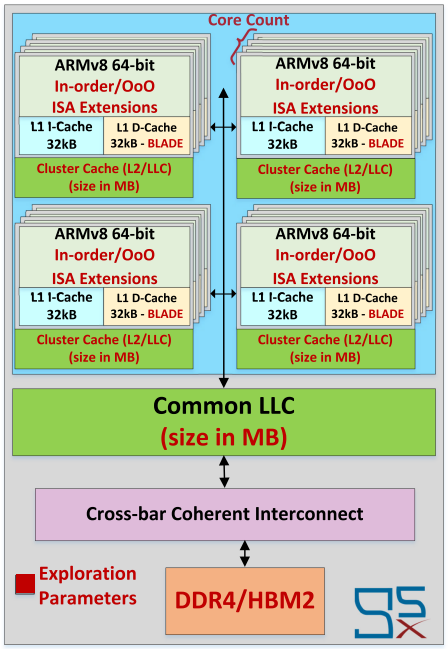


Fig. 3. Gem5-X platform with architectural exploration parameters.

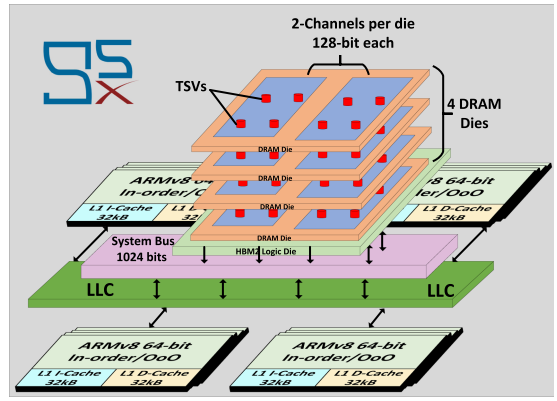


Fig. 4. 3D stacked HBM2 architecture.

- **Accelerator Support:** Gem5-X supports adding custom accelerators for specific tasks. To utilize the added accelerator, in FS mode simulation, one can either memory map or trigger it via a custom instruction. In case of memory mapping of the accelerator, it acts as an external device in the system. For an accelerator tightly integrated within the CPU pipeline, adding a custom instruction is the more feasible approach. For the video encoding part of our case-study application, an in-cache computational engine called BLADE [55] was added to the L1-D cache for increased performance and energy-efficient execution of Kvazaar, accelerating certain FIR filter and the SATD blocks within it, as presented in Reference [50]. The accelerator is supported in conjunction with heterogeneous compute cores as well as clustering.
- **ISA Extension:** To support an accelerator or to add a new functional unit within the CPU core, gem5-X supports ISA extension, using the reserved op-codes of the original ISA specifications. To support BLADE in FS mode, we added a custom instruction using one of the unused opcodes in ARMv8 ISA specification [4]. The accelerator can be used by issuing in-line assembly calls to the new instruction in C/C++.
- **Core Count:** Gem5-X enables many-core simulation with support for 256 cores simulation. In this work, the number of cores we need are up to 31 cores, as discussed in Section 4.3.2.

4.2 Memory Sub-system

The memory sub-system is as essential as the compute sub-system, especially in the case of memory intensive applications, like the video analytics application. During video encoding, the raw video frames are read from memory and the encoded frames written back to it. During MobileNet CNN image classification, the input image, the weights, and the parameters of the network are both

stored in memory and have to be accessed for each image inference. Hence, memories with high bandwidth help in alleviating the memory bottleneck in these memory-intensive applications.

The 3D **High Bandwidth Memory v2 (HBM2)** [58] can achieve a bandwidth of up to 307.2 GB/s, enabled by multiple channels and the **Through Silicon Vias (TSVs)** that connect the 3D stacked DRAM banks. Gem5-X implements the HBM2 memory model with timing estimates as in Reference [50]. Thus, we have updated the memory interleaving among different channels for the HBM2 model to uniformly distribute the memory accesses among different channels. No additional support is required from the software perspective, thus enabling any application to be executed either on a traditional DDR4- or HBM2-based memory system in FS mode without modifying the software. Figure 4 shows the 3D stacked HBM2 with multiple channels in gem5-X. Each channel is 128 bits wide; hence, for 8-channels HBM2, the system bus is configured to be 1,024 bits wide.

Gem5-X also supports heterogeneous memory types such as traditional DDR4 along with 3D HBM2 simultaneously in the same system. Both memories are *mmapped* separately in the gem5-X configuration file. During the Linux boot-up, one of the memories is defined as the base memory, with the full address space available to the kernel. The other memory can then be allocated using *mmap()*, from within the application, if required. By default, allocations are done to the base memory defined during the kernel boot.

4.3 HBM2 Bandwidth Analysis

The gem5-X ARMv8 compute cores were already tuned to have a validation error of up to 4% in our previous work. In this article, we now perform the **bandwidth (BW)** analysis on HBM2 memory model in gem5-X to validate that it provides the required BW in comparison to DDR4 and also to check that the memory traffic is being uniformly distributed among different memory channels, as mentioned previously in Section 2. STREAM [34], which is a well-known memory bandwidth benchmark, is used for the HBM2 bandwidth analysis.

4.3.1 BW Analysis Methodology. Figure 5 shows the methodology for the analysis of HBM2 BW and its comparison to DDR4 in gem5-X using the STREAM benchmark by changing various architectural parameters in the system.

- Input: The STREAM benchmark is used as a benchmark on the ARM-based platform. The input array size of the benchmark is set to 100M, which is equivalent to a memory requirement of 2.2 GB. The array size is chosen to have a tradeoff between memory utilization and simulation turnaround time, which increases linearly with the increase in array size.
- Memory Channels: We run STREAM for HBM2 with 1, 2, 4, and 8 memory channels to see the effect of channel count on BW.
- Memory Types: We use both HBM2 and DDR4 as memory types to compare their performance. DDR4 is used with 1, 2, and 4 memory channels, as the number of channels scale from 1 to 4 in commercially available DDR4-based systems when we move from low-power mobile devices to high-end server-class systems [21–23].
- Core Types: All the benchmark experiments are run for both ARMv8 in-order and OoO cores.
- Core Frequency: We repeat all the benchmark experiments at 1 GHz, 2 GHz, and 4 GHz core frequencies. The 4 GHz frequency is used just for scaling analysis, as we are not aware of any ARM cores operating at 4 GHz.
- Core Count: As we want to analyze the effect of core count on BW, we benchmark core counts of 8 and 16 cores.
- Cache Hierarchy: We also look into changes of the cache hierarchy, namely, the effects of no-LLC system in comparison to system with LLC.

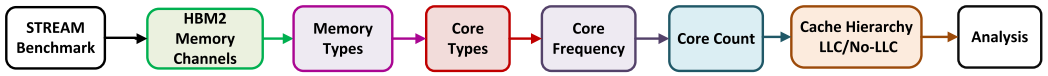
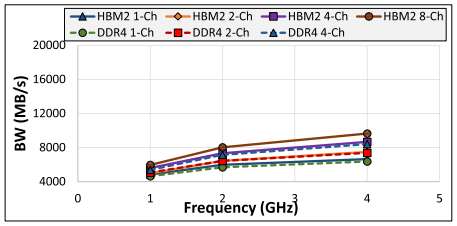
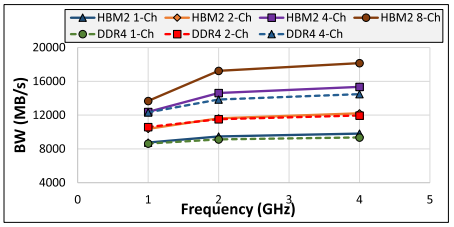


Fig. 5. Methodology for BW analysis using STREAM benchmark.



(a) 8 in-order cores.



(b) 8 OoO cores.

Fig. 6. BW scaling of HBM2 with respect to the number of channels and core frequency in comparison to DDR4 for both 8 in-order and OoO cores when running the STREAM benchmark.

- Analysis: In addition to analysing the results after changing each architectural parameter discussed in this methodology, we globally analyze the results and run the experiment with a change in any architectural parameters necessary.

4.3.2 Bandwidth Analysis Results. We now look into the BW analysis results of HBM2 in comparison to DDR4.

- Figure 6 shows the scaling of HBM2 bandwidth with the number of channels and core frequency, as compared to DDR4 with 1, 2, and 4 channels, for 8-core ARM in-order and OoO systems with a LLC (L2) of 1 MB. We see that the BW increases with the number of channels. HBM2 performs better than DDR4 by 5% for both in-order and OoO cores for same number of channels, whereas 8-channel HBM2 gives up to 34% and 48% more BW as compared to single channel DDR4, for in-order and OoO cores, respectively. Moreover, 8-channel HBM2 gives up to 12% and 19% more BW as compared to 4-channel DDR4. BW scales with the core frequency for 8 in-order cores as in Figure 6(a). We also see that OoO cores utilize much more of the available BW as compared to in-order cores at the same frequency. However, the BW scaling with frequency is low after 2 GHz. To investigate this further, we run STREAM with higher number of cores as discussed below.
- We run the STREAM benchmark for 16 ARM in-order and OoO cores, both with LLC. Figure 7 shows that BW saturates for OoO cores at higher frequency, and BW utilization of in-order cores converge to that of OoO cores. If we compare the BW of 8-OoO cores in Figure 6(b) to that of 16-OoO cores in Figure 7, the BW does not scale with the increase in the core count of OoO cores. However, it does scale with the number of in-order cores. The BW of 8-channel HBM2 is 44%–46% higher than 1-channel DDR4 and 17%–19% higher than 4-channel DDR4. Hence, we limit ourselves to 8-channel HBM2 BW scaling analysis for now.
- Through our analysis of the data path between the compute cores and memory, we observe that the LLC is the bottleneck on the available BW, which explains why OoO cores do not exhibit a linear scaling with the number of cores. However, the reduced bandwidth scaling is also an indication of better performance, as the memory is being accessed less frequently due to the caching effects of LLC. However, we remove the LLC from the system and run the STREAM benchmark again. As shown in Figure 8, the BW scales with number of cores

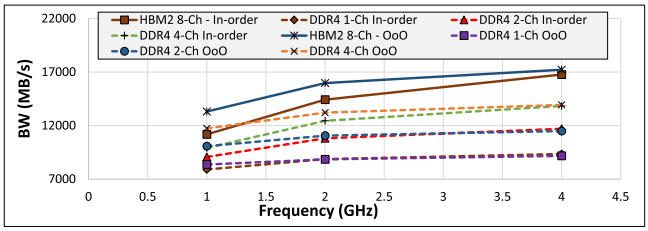
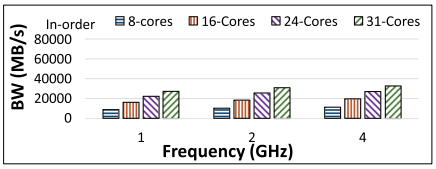
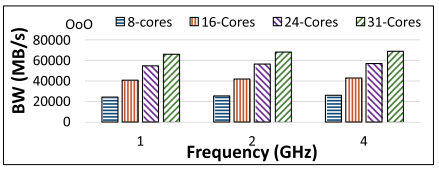


Fig. 7. BW scaling with frequency for 16 cores.



(a) BW scaling with in-order cores.



(b) BW scaling with OoO cores.

Fig. 8. BW scaling of HBM2 with core count and core frequency with no LLC.

both for in-order and OoO cores without the LLC. From Figure 8(a) and Figure 8(b), it is also evident that OoO cores can utilize almost 2× more BW as compared to in-order cores. Also, these figures show that the BW utilization depends on the core count and type of cores. Thus, the BW provided by HBM2 is available, the greater the number of cores, the more its utilization. However, we observe that the BW utilization does not scale linearly in relation to the core frequency and remains almost constant.

- We investigate the L1 cache to see if it is causing any bottlenecks in relation to BW scaling with frequency. We change the size of **miss-status-holding-registers (MSHR)**[1] from 4 to 10. However, there was no change in BW for in-order cores, and around 16% BW improvement for OoO cores when using 10 MSHRs instead of 4. However, the change with frequency scaling was still quite constant. Hence, the reason for almost constant BW being accessed across different frequencies is not due to L1, but results from the fact that the core frequency to issue memory request is high compared to HBM2 serving those requests. Thus, whenever the request goes to HBM2, the cores are stalled waiting for memory to respond to those read/write accesses. As STREAM is a memory intensive benchmark, changing the core frequency does not improve the stressed BW of the memory.

4.4 Power Models and Area

In addition to performance analysis using gem5-X, we are also interested in the power and energy consumption of the systems. However, we did not use gem5 or McPAT power model, as proposed by References [52] and [11], respectively, as they both are for ARMv7 32-bit ISA, whereas we are using ARMv8 64-bit cores. For the CPU energy analysis, we use the power model for 28 nm CMOS bulk technology node for ARM 64-bit in-order and OoO cores proposed in References [44] and [50]. This power model includes the core active, static, and **wait-for-memory (WFM)** energy, LLC read/write, and static cache energy. For the memory power models, power values as reported in References [31] and [43] are used for DDR4 and HBM2, respectively. Counters in gem5-X statistics

[1]MSHRs keep track of outstanding cache misses, thus enabling multiple accesses and outstanding misses in the cache.

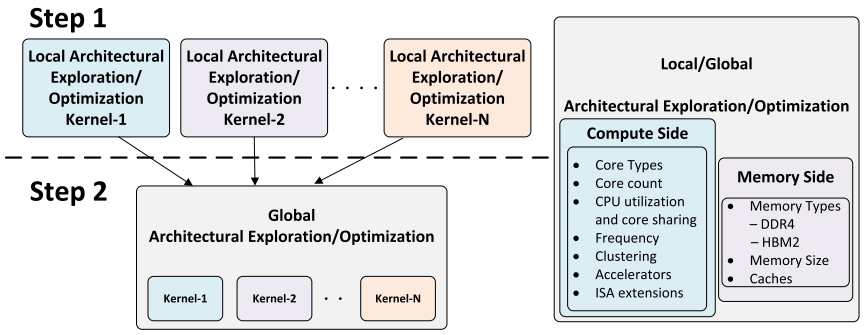


Fig. 9. Architectural exploration and optimization methodology.

such as active CPU cycles, WFM cycles, cache read and writes hits, and main memory accesses are used for power modeling.

For area estimates, we use the values reported in References [16] and [15] for ARM OoO and in-order cores, respectively.

5 ARCHITECTURAL EXPLORATION AND OPTIMIZATION METHODOLOGY

We propose a methodology to explore and optimize architectures for applications that are composed of multiple kernels such as the video analytics application or multiple independent applications allocated simultaneously on a platform. Figure 9 shows the two-step methodology for architectural exploration and optimization of this multiple kernel scenario. The first step is the local architectural optimization for each kernel based on its own bottlenecks and compute/memory requirements. The optimization is on both compute and memory fronts, exploiting their respective architectural parameters as presented in Figure 9, Step 1, and also discussed in Figure 3 in Section 4. Once all the kernels are independently architecturally optimized, we co-allocate them together and globally optimize the architecture for all kernels, if further necessary, as in Figure 9, Step 2. The optimization strategy in Step 2 is the same as for local optimization in Step 1, both on compute and memory side, hence leaving limited room for further optimization, unless there are conflicting architectural requirements between different applications. In such a scenario, the focus is on minimum performance requirements or QoS for each kernel and then further explore the architecture.

Our case-study video analytics application is composed of two kernels, video encoding and CNN-based image recognition. Hence, the complexity of the methodology in Figure 9 is linear. However, if there is an application with more than two independent kernels, i.e., N > 2, then the complexity for the exploration methodology will still be linear. In such scenario, we would co-allocate all independent architecturally optimized kernels as in Step 2, because they do not share any resources except for at shared cache or the main memory level. The shared caches can be made independent and private for each kernel with the clustering approach.

6 COMPUTE RESOURCE ANALYSIS AND OPTIMIZATION FOR REAL-TIME VIDEO ANALYTICS

To have an optimal architecture in terms of performance and energy efficiency for our case-study video analytics, we use the two-step exploration methodology discussed in Section 5. Therefore, as in Step 1 of the methodology in Figure 9, we do a separate architectural exploration and optimization for Kvazaar and MobileNet. Then, we co-allocate them together on a single platform as a video-analytics application and perform global optimization as in Step 2 of the methodology.

Table 2. Initial Experimental Setup

Parameter	Value	Parameter	Value	Parameter	Value
Number of Cores	4	Core Frequency	2 GHz	DDR4 size	4 GB
L1-I cache, L1-D cache	32 KB, 32 KB	L1-I, L1-D associativity	2	L1 MSHRS	4
LLC size	1 MB	LLC associativity	2	LLC MSHRS	20
Cache Coherence Protocol	MOESI				

6.1 Kvazaar Video Encoding

Kvazaar [60] is an open-source HEVC transcoding application capable of performing both online encoding and decoding. For video analytics, we are interested only in the video encoding part of Kvazaar. The performance requirement for encoding is 24 FPS to have a seamless user experience. Hence, for all video resolutions in Table 1, we need to meet 24 FPS requirement. As discussed in Section 3.2 and in Reference [50], FIR filter and SATD are the two dominating blocks in the Kvazaar encoder. Hence, we will focus on optimizing the architecture for these significantly contributing blocks to have an architecture optimized for the overall application.

6.1.1 Experimental Setup. We use ARMv8 64-bit cores booted in Ubuntu 16.04 OS. As a starting point, gem5-X is configured as the ARM JUNO platform [3], which is a validated configuration as in our previous work [50]. The initial configuration is summarized in Table 2.

Our experimental setup is based on general-purpose compute cores (ARMv8 cores) and not ASICs, as quite a lot of effort and time is required for the development of an ASIC. A software-based solution can ease this with just downloading and optimizing the code and running it on available general-purpose compute system. Furthermore, applications developed to run on ASIC cannot be updated/upgraded if there are new improvements to the algorithm. However, CPU-based system allows for updating the currently deployed application with updated or new algorithms on the same platform.

6.1.2 Profiling, Bottlenecks, and Sweeping Parameters. We profiled Kvazaar using Valgrind [38] on ARM JUNO and gperf [17] in gem5-X. On the computational dominancy end, FIR and SATD blocks were dominating with 21% and 26% instructions, respectively, and hence they are potential candidates for optimization. The remaining 53% was distributed among other blocks with contributions of less than 10% and hence not considered for optimization. This profiling data was consistent for both ARM in-order and OoO cores and also both in gem5-X as well as ARM JUNO.

On the memory side, the L1 cache miss rate was 4.8% and 5.2% for FIR and SATD, respectively, indicating high locality. Hence, we sweep the sizes of both L1 and LLC from 8 KB to 128 KB and 512 KB to 16 MB, respectively, to find the optimal size. The L1 size of 32 KB along with a LLC of 1 MB is found to be optimal in terms of both performance and area, as also discussed in Reference [50].

6.1.3 BLADE - In-cache Computing Engine. BLADE is a hardware extension for CPU caches that enables computing directly within the cache. Blade operations are performed by first precharging the bitlines of the cache subarray, as illustrated in Figure 10(a). Then, two wordlines are activated simultaneously, thus allowing the contents of two rows of bitcells to be connected to the bitlines, as illustrated in Figure 10(b). The bitlines are discharged according to the contents of the bitcells, resulting in an `and` operation on the bitline and a `nor` operation on the inverse bitline. These signals can be further combined via a `nor` gate to achieve a `xor` operation. Finally, further processing allows complex operations such as addition and multiplication to be performed [55, 56]. The operation results are then written back to the cache. Application runtime is improved in two respects:

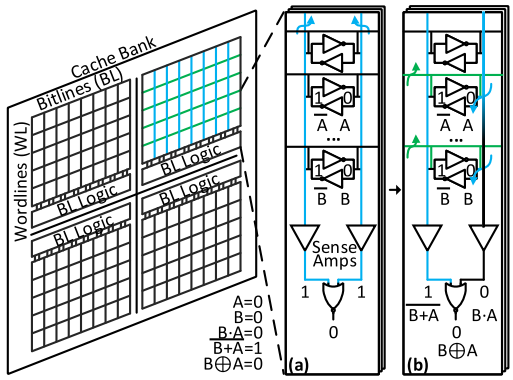


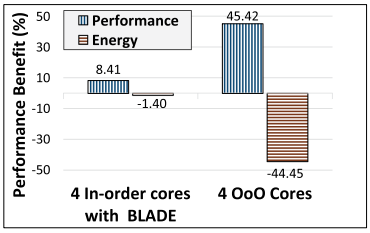
Fig. 10. Cache subarray with AND/NOR/XOR bitline computing on values A=0 and B=0. Bitwise operations are performed by first (a) precharging the bitlines, then (b) activating multiple wordlines, thus discharging the bitlines through the connected bitcells.

first, data movement is reduced, and second, **Single Operation Multiple Data (SIMD)** operations can be performed on multiple operands simultaneously, for example 128 1-byte operations in a cache with two subarrays and 64-byte wordlines. It also reduces energy consumption, as the computation is performed in the cache, hence saving energy both on the internal bus transactions as well as static energy for the idle core when the data is being fetched.

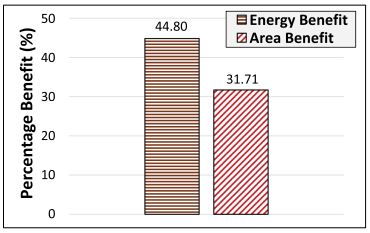
BLADE provides acceleration primarily for applications that exhibit high cache locality, as it performs SIMD operations on many physically successive operands. Since FIR and SATD perform simple operations with high cache locality, they are potential candidates for BLADE acceleration, as discussed in Reference [50], and have thus been modified at the source-code level to support BLADE operations. BLADE has been incorporated in the L1 cache for ARM in-order cores.

6.1.4 Architecture Exploration - Results and Analysis. There are two video resolutions, 300x200 pixels and 640x480 pixels, that cover all three-video analytic case-study scenarios, namely, surveillance, drone navigation, and autonomous cars, as presented in Table 1. Therefore, for the architectural exploration and optimization of Kvazaar, we will be using these two resolutions. Our baseline architecture will be as in Table 2, with ARMv8 64-bit in-order cores with the core count dependent on video resolution to meet the 24 FPS requirement. BLADE will be used as an accelerator along with ARMv8 in-order cores. We will also be comparing the performance to ARMv8 OoO cores.

- *300x200 Resolution:* Figure 11(a) shows the performance and energy benefits of ARM 4-core in-order system with BLADE and ARM 4 OoO cores in comparison to the baseline of ARM 4 in-order cores, all operating at 2 GHz and DDR4 as main memory, while processing 300x200 resolution video at 24 FPS or more. We observe that 4 in-order cores with BLADE are around 8% performance efficient, with a negligible energy overhead of 1% as compared to in-order system without BLADE. OoO cores, however, improve performance by 45% but at the cost of consuming 44.44% more energy than 4 in-order cores. Hence, we scale the OoO core count to 2-cores, to just meet the 24 FPS and not more.
  Also, we select 4 in-order cores with BLADE as our reference and compare their energy and area to 2-OoO cores. Now both systems have the same performance to encode at 24 FPS, but 2 OoO cores consume 45% more energy along with an area overhead of 31%, as depicted in Figure 11(b). Therefore, a 4-core in-order system with BLADE is the optimal architecture for 300x200 resolution video.

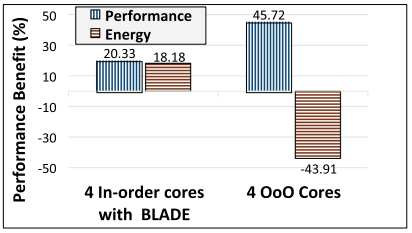


(a) Performance and energy benefit in comparison to 4 in-order cores.

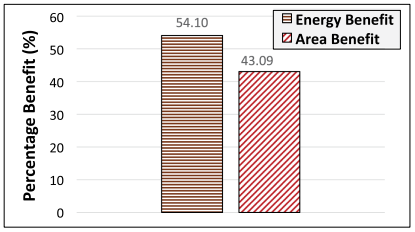


(b) Energy and area benefit of 4 in-order cores with BLADE compared to 2 OoO cores.

Fig. 11. Performance, energy, and area comparison for Kvazaar video encoding of 300x200 resolution video.



(a) Performance and energy benefit in comparison to 4 in-order cores.



(b) Energy and area benefit of 10 in-order cores with BLADE compared to 6 OoO cores.

Fig. 12. Performance, energy, and area comparison for video encoding of 640x480 resolution video.

- *640x480 Resolution:* For 640x480 resolution video, we start with a core count of 4 cores for ARM in-order, in-order with BLADE, and OoO cores at 2 GHz. As shown in Figure 12(a), 4 in-order cores with BLADE perform 20% better than in-order cores without BLADE in terms of FPS, along with 18% less energy consumption. The performance and energy benefit for in-order cores with BLADE is more in comparison to 300x200 resolution video, as a larger dataset exploits more cache capacity and provides more opportunities for using BLADE vector processing in the cache. The OoO cores improve performance by 45%, but consume 44% percent more energy in comparison to in-order without BLADE, similar to 300x200 resolution video. However, with 4 cores none of these architectures meet the 24 FPS requirement. Hence, we increase the core count until the 24 FPS requirement is met. For in-order cores without BLADE, we increase the number of cores to 12 to reach 24 FPS, whereas, with BLADE only 10 cores are necessary, indicating a performance benefit of 20%. The number of OoO cores at 2 GHz required to encode 24 FPS is 6. While processing 24 FPS for a 640x480 resolution video, we compare the energy and area required for OoO cores in comparison to in-order cores with BLADE. We find that 10 in-order cores with BLADE are 54% more energy-efficient compared to 6 OoO cores, and at the same time taking 43% less area as well, as shown in Figure 12(b).

The BLADE in-cache computing engine reduces the energy consumption, as it operates on larger dataset using the **single-instruction-multiple-data (SIMD)** approach as discussed in References [50, 55]. Furthermore, it reduces the data movement between cache and the functional units in the CPU, as the operations are performed in cache, which reduces the energy consumption. For the video encoding kernel, Table 3 shows the core and memory power for 300x200 and 640x480 resolution videos while meeting the 24 FPS requirement. Here, we see that the energy reductions

Table 3. Time, Core Power (at 2 GHz) and Memory Power for Video Encoding, while Meeting 24 FPS Requirement

Parameter	300x200 Video Resolution			640x480 Video Resolution		
	In-order 4 Cores	OoO 2 Cores	In-order 4 Cores with BLADE	In-order 12 Cores	OoO 6 Cores	In-order 10 Cores with BLADE
Time (s)	1.002	1.0942	0.918	1.006	1.092	0.962
Core Power (W)	0.872	1.460	0.8567	3.835	6.448	2.987
Memory Power (W)	0.0291	0.0259	0.141	0.175	0.137	0.444

Table 4. MobileNet Classification Rate for Various Application Scenarios

Case	**FPS - Original Resolution**	**FPS - Split Images**
Surveillance (640x480)	5	30
Drone Navigation (300x200)	10	20
Autonomous Driving and ADAS (640x480)	15	90

mainly come from the core power. In-order cores with BLADE in-cache computing has the least power, as compared to OoO cores, as well as the in-order cores without BLADE. In summary, this situation leads to low-power ARM in-order cores with BLADE to match the performance of high-performance ARM OoO cores with 54% less energy budget.

Therefore, for the video encoding portion of video analytics, we will be using ARM64 bit in-order cores along with BLADE in-cache computing engine, as it produces the best energy efficiency and area occupancy, while achieving the required 24 FPS. All the cores in the system are 100% utilized. Hence, there cannot be any resource sharing.

6.2 MobileNet

MobileNet [20] is a state-of-the-art CNN used for image classification, with low computational cost designed for deployment on mobile and edge devices. As image classification and detection are necessary for video analytics, we will be using MobileNet for this purpose. The ARM ACL [5] framework is used to deploy an ARM optimized version of MobileNet, which we use in our gem5-X-based experiments to look into further architectural exploration and optimization choices.

MobileNet allows input image size of 224x224 pixels, whereas our three case-study scenarios in Table 1 have 300x200 and 640x480 pixel resolution. As discussed in Section 3.3, each 300x200 and 640x480 frame is split into 2 and 6 images of 224x224 pixels as inputs to MobileNet, respectively. Therefore, the resulting FPS requirement for MobileNet is higher, as shown in Table 4.

6.2.1 Performance Comparison - CPU vs. GPU. GPUs are increasingly being used both in training and inference of the CNNs architectures due to their high-performance capability of processing vectored data, making them a perfect fit for CNNs [27, 62]. Then, as our case-study scenarios involve edge devices, we looked into energy-efficient GPUs for mobile and edge computing.

Nvidia Jetson Nano is low-power state-of-the-art GPU designed for **artificial intelligence (AI)** tasks on embedded and edge devices [41]. We compare the performance and energy consumption for MobileNet inference between ARM in-order cores, OoO cores, and Jetson Nano. The Nvidia Jetson Nano platform [40, 41] comes with 128 Nvidia Maxwell GPU cores integrated with 4 ARM Cortex A-57 cores. Furthermore, the Nvidia Jetson Nano has better software support, as compared to other platforms such as ARM Mali GPU [7] and Ethos NPU [6]. It can run the networks

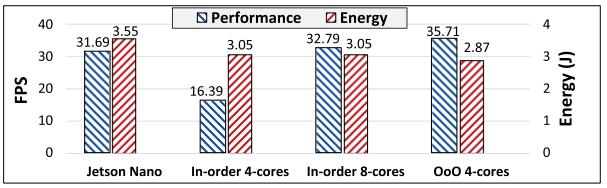


Fig. 13. MobileNet performance and energy comparison.

implemented in Tensorflow directly and has the optimized versions of the networks in TensorRT[39], an optimized framework to deploy CNNs on Nvidia GPUs. MobileNet is deployed on Jetson Nano using the Nvidia TensorRT framework [39]. The energy statistics on Jetson Nano are collected using the Nvidia *tegrastats* utility. Jetson Nano is set to its high-performance mode (10 W mode) with its 128 GPU cores operating at 921 MHz. The CPU and GPU in the Jetson Nano are in separate power domains [40], and in high-performance mode, the power rails VDD_CPU and VDD_GPU for CPU and GPU, respectively, are both set to 1.322 V. These power rails only account for the CPU and GPU compute cores of Jetson Nano and not any other peripherals or I/Os, as they are in separate power domains. Both ARM in-order and OoO cores are configured in gem5-X to operate at 2 GHz with 32 KB L1 instruction and data cache. The LLC is configured to 1 MB for 4 cores. As we will discuss later in Section 6.2.2, MobileNet performance scales linearly with multiple threads on multi-core system up to 4 cores. Hence, for an 8-core simulation, we launch two instances of Mobilenet, each using 4 cores independently, so the performance scales with core count. Thus, we use a clustering approach that we discuss in detail in Section 6.2.2. Then, HBM2 is used as memory instead of DDR4, which is an architectural optimization that we discuss in detail in Section 6.2.3.

Figure 13 shows the performance in terms of FPS achieved by MobileNet on different architectures. This figure shows that ARM 8 in-order and 4 OoO cores surpass the performance of Jetson nano by 3.3% and 11%, respectively. Hence, 4 OoO cores achieves the highest performance. The reason as to why the lower number of ARM CPU-based system matches or outperforms Jetson Nano comes from the fact that ARM CPU-based system is operating at 2 GHz, whereas, Jetson Nano GPU-based platform with 128 GPU cores is operating at 921 MHz. Regarding energy comparison, we computed the energy for processing 21 MobileNet images. Jetson Nano consumes the highest energy, as can be seen in Figure 13, whereas OoO cores are the most energy-efficient. OoO cores are 27% more energy-efficient as compared to Jetson Nano and 6% in comparison to 8 in-order cores. The energy of 8 in-order cores is the same as 4 in-order cores, as 8 in-order cores achieve double the FPS (half the execution time) in comparison to 4 in-order cores. Overall, OoO cores with HBM2 are the best both in terms of performance and energy efficiency for processing MobileNet. Jetson Nano consumes more energy in comparison to ARM CPU-based system, because the high number of GPU compute cores (128 cores) in Jetson contribute largely towards the high energy consumption [19]. Secondly, the energy for the Jetson Nano platform includes the GPU energy as well as the 4 ARM Cortex-A57 cores, which are mostly idle and not used for the actual computation of the MobileNet CNN, but are used as the host cores for the GPU. As these cores are mostly idle, their static energy adds to the total energy of the Jetson Nano platform.

6.2.2 MobileNet Scaling and Clustering. To achieve higher FPS, e.g., 90 FPS for an ADAS system, more cores are required to process the images, as well as the use of multiple threads with one thread per core. Therefore, we investigate how MobileNet scales with the number of cores.

We find that FPS scales linearly up to 4 cores, however, after that, the performance does not scale up with core counts and number of threads. This is due to the fact that, since MobileNet is a

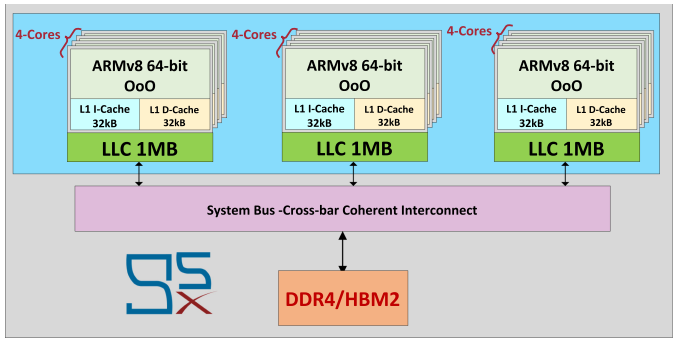


Fig. 14. Clustered architecture for multiple MobileNet instances.

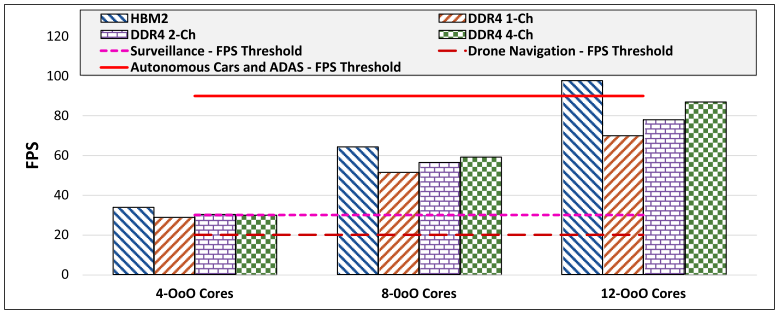


Fig. 15. MobileNet FPS scaling with core count (OoO) for HBM2 (8-Ch) and DDR4 with different number of channels. The horizontal lines are the threshold that should be met for scenarios in Table 1.

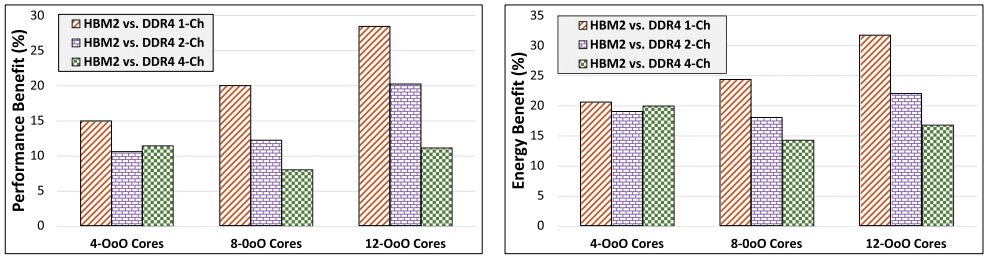
lightweight network, the cost of sharing data among higher number of cores affects the performance per core, thus suppressing the overall performance.

To overcome this issue of scaling, we launch multiple independent MobileNet instances, with each instance using a maximum of 4 cores. Hence, we instantiate 4 core clusters in gem5-X, each with its own LLC, so there is no thrashing in the cache. The clustered architecture with three clusters is shown in Figure 14.

6.2.3 Architecture Exploration - Results and Analysis. For the compute side optimization of MobileNet, we already concluded in Sections 6.2.1 and 6.2.2 that ARM OoO cores are the best in terms of compute core performance, and that to use clustering for parallel processing and launching separate instance of MobileNet on each cluster.

Next, regarding the memory side exploration, we consider 8-channel **high bandwidth memory (HBM2)**, as CNNs are memory-intensive applications due to weight and bias accesses of the network. For our three-video analytics application case-study scenarios, we need to meet the FPS requirement for the visual recognition portion, as in Table 4. Hence, we run experiments with ARM OoO cores at 2 GHz, with the architecture shown in Figure 14, both with DDR4 and HBM2 memory types.

Figure 15 shows that 4 OoO cores with HBM2 meets the FPS requirement for drone navigation and surveillance scenarios, whereas, 4 OoO cores with DDR4 (1 to 4 channels) only meet the FPS requirement for drone navigation. However, using HBM2 instead of DDR4 results in energy benefit of up to 21%, as shown in Figure 16(b). For ADAS and autonomous smart cars,



(a) Performance benefit. (b) Energy benefit.

Fig. 16. MobileNet performance and energy benefit of using HBM2 (8-Ch) over DDR4 with different number of channels for OoO cores.

where the requirement is 90 FPS, we launch three instances of MobileNet, each on a 4-core cluster. Figure 15 shows that 12 OoO cores with HBM2 meet the 90 FPS requirement. Again, 12 OoO cores with DDR4 (1 to 4 channels) fail to reach the required FPS. We also launch the experiments with 8 OoO cores with two MobileNet instances and see that it scales well with our clustering approach, from 4-core to 8-core to 12-core systems.

Finally, we compare the percentage performance and energy benefit of using 8-channel HBM2 instead of DDR4, with number of channels varying from 1 to 4, for MobileNet. Figure 16(a) shows that the performance benefit varies from 8% to 28%, as it scales with the number of cores as well as with the number of DDR4 channels. Similarly, as shown in Figure 16(b), there are energy savings of up to 32% for 12-core system when using HBM2 instead of single channel DDR4. The energy savings when using HBM2 instead of 4-channel DDR4 is between 16% to 20%, depending on the number of compute cores.

Therefore, ARM OoO core system with 8-channel HBM2 is the best-performing system both in terms of performance and energy efficiency for MobileNet-based visual recognition tasks.

6.3 Video Analytics

Once the Kvazaar video encoding and MobileNet visual recognition kernels are independently optimized for performance and energy efficiency, we move to Step 2 of the methodology in Figure 9. Then, the global architectural optimization and exploration is performed with all the application kernels co-allocated on the same platform, running in parallel. This can result in heterogeneous architectures with different clusters, as each locally optimized architecture might be different from one kernel to another, which share the crossbar interconnect and memory.

6.3.1 Heterogeneous Architecture. Figure 17 shows the heterogeneous architecture for the complete video analytics application. There is a separate compute cluster for Kvazaar, with ARM in-order cores and ISA extensions to use the in-cache computing engine, BLADE, integrated into the L1-D. Similarly, there are multiple clusters for MobileNet with 4 ARM OoO cores per cluster. The number of clusters depends on the required FPS. All of the clusters have their own LLC. HBM2 is used as the main memory in the system, as it offers more BW compared to DDR4 and it is necessary for MobileNet to achieve the required FPS. All the cores and clusters are 100% utilized for both the Kvazaar and MobileNet kernels. Hence, there will be no resource sharing between different kernels in terms of compute cores and their components, as it will not benefit the application performance. We will look into the performance and energy benefits of using HBM2 for Kvazaar as well as the overall system in the next section.

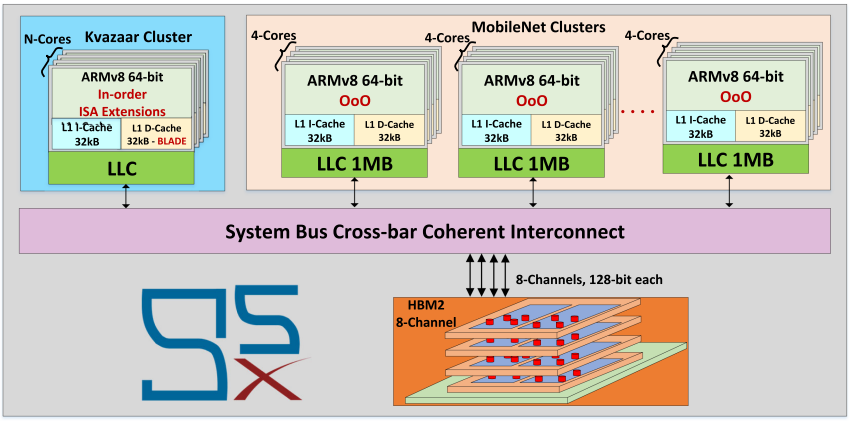


Fig. 17. Heterogeneous architecture for video analytics.

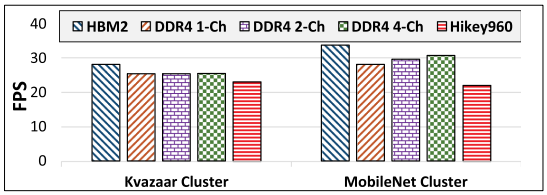
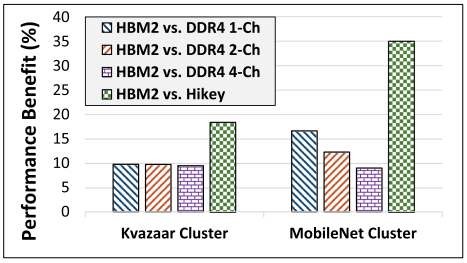


Fig. 18. Video Analytics FPS for Kvazaar and Mobilenet clusters for 300x200 resolution video for drone navigation when using HBM2, DDR4, and on Hikey960 platform.

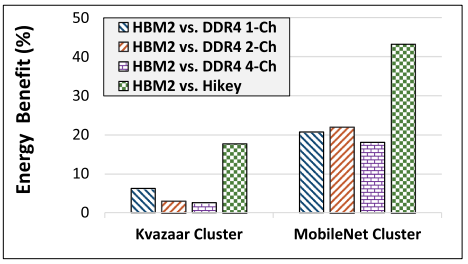
6.3.2 Architecture Exploration - Results and Analysis. For video analytics, our primary goal is to meet the QoS and performance requirements as set out in Tables 1 and 4 and improve the energy efficiency as much as possible. As there are three scenarios, one with a resolution of 300x200 pixels (drone navigation) and two with 640x480 pixels (surveillance and autonomous cars), we will explore and discuss the results according to the video resolutions.

- **300x200 resolution:** We concluded in Section 6.1.4 that 4 in-order cores at 2 GHz with BLADE in-cache computing is the optimal architecture to meet 24 FPS requirement for the Kvazaar video encoding part of video analytics. For visual recognition using MobileNet, 4 OoO cores at 2 GHz were optimal in terms of performance and energy to meet the FPS requirement of 20 FPS for drone navigation, with HBM2 as the main memory. Hence, the architecture in Figure 17 will have one Kvazaar cluster with 4 cores and LLC of 1 MB and one Mobilenet cluster of 4 OoO cores.

  Figure 19(a) shows that performance benefit of 10% and 16.6% for Kvazaar and MobileNet clusters, respectively, when HBM2 is utilized instead of single-channel DDR4. The FPS values for both clusters with HBM2 and DDR4 are shown in Figure 18. The performance benefit is approximately 9% for both clusters when using HBM2 instead of 4-channel DDR4. We also observe an energy benefit of 6% and 20.7% for Kvazaar and MobileNet, respectively, when HBM2 is compared to single-channel DDR4. When compared to 4-channel DDR4, the energy benefit is 2.5% and 18% for Kvazaar and MobileNet, respectively. The FPS requirement of Kvazaar is that of 24 FPS, whereas that of MobileNet is 20 FPS, and we are above these minimum requirements for both applications, as shown in Figure 18. Moreover, we also compare our proposed architecture with HBM2 to commercially available ARM big.LITTLE



(a) Performance Benefit.



(b) Energy Benefit.

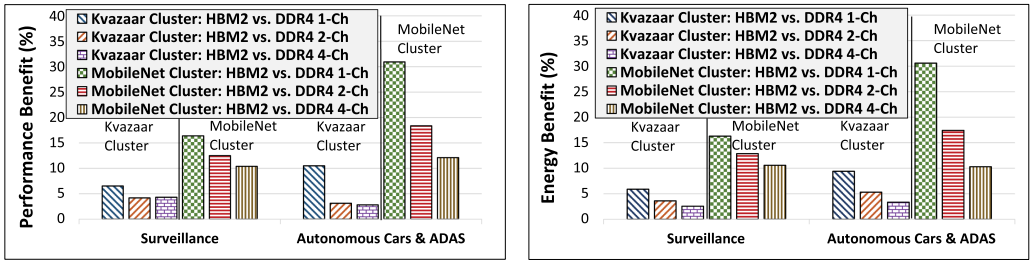
Fig. 19. Percentage performance and energy benefit when using HBM2 (8-Ch) in comparison to DDR4 with different number of channels and Hikey960 platform for video analytics of 300x200 resolution video for drone navigation.

Hikey960 platform [1], which has 4 in-order and 4 OoO ARM cores. As shown in Figure 18, the Hikey960 achieves the least FPS, leading to 18% and 35% performance benefit of our proposed architecture using HBM2 over Hikey960, for Kvazaar cluster and MobileNet cluster, respectively, as shown in Figure 19(a). With regards to energy, we achieve 18% and 43% energy savings over Hikey960, for Kvazaar and MobileNet clusters, respectively, as shown in Figure 19(b).

To further optimize the architecture, we use Step 2 of our optimization methodology to optimize for energy efficiency, as we have already met the performance requirements. Both kernels scale linearly with frequency. Hence, we reduce the frequencies of both clusters to meet minimum performance requirements, therefore, enabling more energy savings. Reducing the frequency of Kvazaar 4-core cluster from 2 GHz to 1.72 GHz results in an energy saving of 32% while achieving 24 FPS. Similarly, for the MobileNet cluster, we reduce the frequency from 2 GHZ to 1.2 GHz, resulting in an energy saving of 59%.

- **640x480 resolution:** For the 640x480 resolution, two case-study scenarios exist. First, one of video surveillance with MobileNet requirement of 30 FPS. Second, autonomous cars and ADAS with MobileNet are used for image recognition with a requirement of 90 FPS. The video encoding for both scenarios is required to execute at 24 FPS. Hence, as discussed in Section 6.1.4, a 10-core in-order cluster at 2 GHz with BLADE is instantiated for Kvazaar, with HBM2 as main memory instead of DDR4. For MobileNet clusters, a 4-core OoO cluster at 2 GHz meets the 30 FPS requirement and three 4-core clusters meet the 90 FPS requirement, as discussed in Section 6.2.3.

  The performance benefit of 10-core Kvazaar cluster varies from 6.5% to 10.5% when using HBM2 instead of single-channel DDR4, while meeting the 24 FPS requirement, as shown in Figure 20(a), and between 2.8% to 4.3% when comparing HBM2 to 4-channel DDR4. We can see that the percentage benefit of the Kvazaar cluster for 640x480 resolution video, when HBM2 is being used instead of DDR4, is less as compared to 300x200 video resolution as in Figure 20(a). The reason being that, the higher the resolution, the more efficiently BLADE in-cache computing and caching is used as compared to lower resolution video. Hence, lower BW utilization of main memory and less performance benefit. The corresponding energy savings of using HBM2 for the Kvazaar cluster is between 6% to 10% and 2.6% to 3.3%, in comparison to single-channel and quad-channel DDR4, respectively, as depicted in Figure 20(b). For the MobileNet clusters, the performance as well as energy efficiency scale, as we increase the 4-core clusters from 1 to 3 for surveillance and ADAS, respectively. The



(a) Percentage performance benefit.

(b) Percentage energy savings.

Fig. 20. Video analytics for 640x480 resolution video for video surveillance and autonomous cars and ADAS systems. 10-core Kvazaar cluster is used for video encoding in both scenarios. A 4-core OoO cluster is used for MobileNet in case of surveillance, and three 4-core OoO clusters are used for autonomous cars and ADAS.

performance improvement of using HBM2 is up to 30% with a corresponding energy benefit of 30% in comparison to single-channel DDR4. When compared to 4-channel DDR4, the performance and energy benefits of using HBM2 are up to 12% and 10%, respectively, for the MobileNet clusters. Comparing the results of MobileNet cluster in Figure 20 to that of in Figure 19, it can be observed that the performance and energy savings are higher for three 4-core MobileNet clusters when it is co-allocated with Kvazaar than when it is standalone. This is because, when more applications are being allocated on a single platform sharing the same memory, the memory BW requirements increase, which HBM2 serves better than DDR4.

The overall energy reduction when using HBM2 instead of DDR4 is due to the fact that HBM2 not only has lower energy per access in comparison to DDR4 [31, 43], but also enables faster memory access via eight memory channels. The faster access in turn implies less stalling of the processor pipelines when waiting for data, resulting in overall energy reduction. Table 5 summarizes the core power, memory power, and execution time per frame for both kernels of video analytics in drone navigation, surveillance, and ADAS scenarios. We see that the runtime to process each frame is less when using HBM2 instead of DDR4. Second, the memory power when using HBM2 is less as compared to when using DDR4. Therefore, lower memory power and fast processing time for each frame (both for video encoding and MobileNet kernels) contributes to the energy benefits we see when using HBM2 instead of DDR4.

In these case-study scenarios for surveillance and autonomous cars, we do not use Step 2 of the methodology, as the performance requirements are just met with the proposed architectures. Thus, we cannot further reduce the frequency to optimize for energy, as we did in the case of drone navigation, because this will degrade the performance below the minimum requirement.

In summary, capitalizing on our new gem5-X simulation platform, we propose an optimized heterogeneous architecture for a video analytics application, composed of ARM in-order cores along with BLADE in-cache computing engine being used for the video encoding portion and ARM OoO core clusters for CNN-based image recognition with HBM2 as the main memory. This heterogeneous architecture meets all the performance requirements and at the same time is energy-efficient in all the three case-study scenarios. In this case study, Step 2 of the methodology was used to optimize for energy by reducing the frequency for drone navigation scenario. Other than this, as the two kernels of the application do not give rise to any conflicting requirements, hence, no tradeoffs

Table 5. Time, Core Cluster Power (at 2 GHz) and Memory Power for Video Analytics Case-study Scenarios

Parameter	Clustered Architecture with HBM2			Clustered Architecture with DDR4		
	Drone Navigation	Surveillance	ADAS	Drone Navigation	Surveillance	ADAS
Time Kvazaar Frame (ms)	35.45	38.11	39.47	39.29	40.76	42.83
Core Power Kvazaar Cluster (W)	0.865	2.943	2.873	0.832	2.924	2.940
Time MobileNet Frame (ms)	29.55	29.41	10.465	35.44	35.17	15.015
Core Power MobileNet Cluster (W)	4.490	4.738	14.215	4.723	4.732	15.645
Memory Power (W)	0.198	0.511	1.222	0.631	1.672	3.302

have to be made. Moreover, the methodology is generic and if an application requires further architectural exploration during Step 2, it can be performed. For instance, in case of different kernels of the application running on different cores, when allocated together on a single system, they might all share the same LLC. In that case, the strategy to follow in Step 2 would be to optimize LLC size to fit the working dataset of all kernels. In another instance, the memory requirement for individual kernels might be sufficient, but when co-allocated, the application might run out of memory. Thus, in Step 2, we will select appropriate memory size, forcing us to even change the memory technology being used, as some memories like the 3D stacked HBM2 are limited in terms of the memory size, as compared to traditional DDR4 memories. Therefore, Step 2 of the methodology can be used to optimize the architecture for the compute and memory sub-systems in cases when conflicting requirements arise after co-allocating all the kernels of the application together, in addition to the tuning of core frequencies for optimal performance and energy. Furthermore, the two-step methodology is generic and not specific to any application or application domain. It can be used to explore and optimize architecture for any application domain. The advantage of using the methodology for a general-purpose CPU-based system is that it gives an optimized architecture for the application domain and is not specific to an application. This enables updating the application or replacing the application or one of its kernels with a better application or kernel, which is not possible in ASICs, as they are application-specific. This has been demonstrated by using MobileNet for image classification kernel in this work, instead of AlexNet, which was used in our previous work in Reference [50]. From Figure 19, we can see that using an 8-core system with 4 cores allocated to Kvazaar and 4 cores for MobilNet, the performance benefit of using HBM2 instead of DDR4 is up to 10% and 16.6% for Kvazaar and MobileNet, respectively, in comparison to 7.72% and 8.35% performance benefit for Kvazaar and AlexNet, respectively, in the previous work in Reference [50], in a similar 8-core system.

7 CONCLUSION

In this article, we have presented the new gem5-X simulation platform to enable exploration of many-core heterogeneous architectures to optimize performance and energy consumption in new emerging dynamic applications, such as CNNs for image classification and multi-view video encoding. Compared to gem5, the latest version presented in this article of gem5-X supports validated ARMv8 in-order and OoO cores in FS mode with a validation error of up to 4%. Also, gem5-X supports multiple heterogeneous ARMv8 cores along with heterogeneous memories including DDR4

and the new 3D stacked HBM2 in FS mode, completely integrated and tested, which gem5 does not support straight out of the box. In addition to heterogeneous compute cores, gem5-X enables exploration of clustering configurations of compute cores. Moreover, gem5-X supports accelerators to be integrated within the system like the BLADE in-cache computing engine by capitalizing on an ISA extension support. Furthermore, in comparison to gem5, gem5-X enables profiling support using the gperf profiler, supports advanced check-pointing to help reduce the simulation turnaround time, and comes with **workload-automation (WA)** to enable file sharing between host machine and the simulated system. All in all, in this work, we have proposed a new two-step architectural exploration and optimization methodology of new many-core architectures for new dynamic applications and benchmarks.

Using the system configuration capabilities of our new gem5-X framework and exploration methodology for many-core systems, we have analyzed the benefits of HBM2 vs. DDR4 in the STREAM benchmark. Then, real-time video analytics was used as a complete case-study application to explore and optimize architectures in different execution scenarios. In particular, we have used three scenarios of video analytics, namely, surveillance, drone navigation, and autonomous cars and ADAS, for architecture exploration and optimization, as each has different performance requirement. In the end, thanks to gem5-X and our exploration methodology, an optimal clustered heterogeneous architecture with ARM in-order cores cluster along with a BLADE in-cache computing engine embedded within L1-D cache was defined as optimal option to process the video encoding kernel, as well as an OoO core cluster was included to process the CNN-based image classification kernel. Overall, gem5-X allows to select the right number of clusters and their operating frequencies for different case-study scenarios according to the performance requirement in each case. Furthermore, on the memory side, gem5-X proved that the use of HBM2 for the video analytics application led to both performance and energy benefits of up to 30%, compared to similar DDR4-based system. Hence, we demonstrated that using gem5-X enables a truly complete and fast design space exploration for many-core architectures.

The gem5-X simulation framework is open-sourced to the community, enabling out-of-the-box, fast simulation of many-core ARM 64-bit heterogeneous architectures with innovative architectural extensions. It is readily available for download online in this link: https://esl.epfl.ch/gem5-x. [48]. A technical reference manual for gem5-X has also been published online [47], which includes a quick-start guide as well as instruction on how to use different architectural extensions and support enhancements in gem5-X.

REFERENCES

[1] 96Boards. 2018. HiKey960. Retrieved from https://www.96boards.org/product/hikey960/.

[2] G. Ananthanarayanan, P. Bahl, P. BodÃČÂŋk, K. Chintalapudi, M. Philipose, L. Ravindranath, and S. Sinha. 2017. Real-time video analytics: The killer app for edge computing. *Computer* 50, 10 (2017), 58–67.

[3] ARM. 2015. ARM Versatile Express Juno r2 Development Platform. https://developer.arm.com/-/media/Arm%20Developer%20Community/PDF/Juno%20r2%20datasheet.pdf.

[4] ARM. 2017. ARM Architecture Reference Manual ARMv8. https://developer.arm.com/documentation/ddi0487/ga.

[5] ARM. 2018. ARM Compute Library Framework. Retrieved from https://developer.arm.com/technologies/compute-library.

[6] ARM. 2021. Arm ethos-n series processors. Retrieved from https://developer.arm.com/ip-products/processors/machine-learning/arm-ethos-n.

[7] ARM. 2021. Mali GPUs for graphics processing. Retrieved from https://www.arm.com/products/silicon-ip-multimedia.

[8] S. Bianco, R. Cadene, L. Celona, and P. Napoletano. 2018. Benchmark analysis of representative deep neural network architectures. *IEEE Access* 6 (2018), 64270–64277.

[9] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish,

Mark D. Hill, and David A. Wood. 2011. The Gem5 simulator. *SIGARCH Comput. Archit. News* 39, 2 (Aug. 2011), 1–7. DOI: https://doi.org/10.1145/2024716.2024718

[10] Benjamin Bross. 2012. High efficiency video coding (HEVC) text specification draft 9 (SoDIS). In *11th JCT-VC meeting*.

[11] A. Butko, F. Bruguier, A. Gamatié, G. Sassatelli, D. Novo, L. Torres, and M. Robert. 2016. Full-system simulation of big.LITTLE multicore architecture for performance and energy exploration. In *MCSOC*. 201–208. DOI: https://doi.org/10.1109/MCSoC.2016.20

[12] Trevor E. Carlson, Wim Heirman, and Lieven Eeckhout. 2011. Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation. In *SC*. 1–12.

[13] Tarek Elgamal, Shu Shi, Varun Gupta, Rittwik Jana, and Klara Nahrstedt. 2020. SiEVE: Semantically Encoded Video Analytics on Edge and Cloud. arXiv:cs.DC/2006.01318

[14] Cagkan Erbas, Andy D. Pimentel, Mark Thompson, and Simon Polstra. 2007. A framework for system-level modeling and simulation of embedded systems architectures. *EURASIP Journal on Embedded Systems* 2007 (2007), 1–11. DOI: https://doi.org/10.1155/2007/82123

[15] A. Frumusanu and R. Smith. 2015. Cortex A53 - Performance and Power. Retrieved from https://www.anandtech.com/show/8718/the-samsung-galaxy-note-4-exynos-review/4.

[16] A. Frumusanu and R. Smith. 2015. Cortex A57 - Performance and Power. Retrieved from https://www.anandtech.com/show/8718/the-samsung-galaxy-note-4-exynos-review/6.

[17] Google. 2011. gperftools. Retrieved from https://github.com/gperftools/gperftools.

[18] Nikolaos Hardavellas, Stephen Somogyi, Thomas F. Wenisch, Roland E. Wunderlich, Shelley Chen, Jangwoo Kim, Babak Falsafi, James C. Hoe, and Andreas G. Nowatzyk. 2004. SimFlex: A fast, accurate, flexible full-system simulation framework for performance evaluation of server architecture. *SIGMETRICS Perform. Eval. Rev.* 31, 4 (2004), 31–34.

[19] Sunpyo Hong and Hyesoon Kim. 2010. An integrated GPU power and performance model. *SIGARCH Comput. Archit. News* 38, 3 (June 2010), 280–289.

[20] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient convolutional neural networks for mobile vision applications. Retrieved from http://arxiv.org/abs/1704.04861.

[21] Intel. 2015. Intel xeon processor E5-1620. Retrieved from https://ark.intel.com/content/www/us/en/ark/products/64621/intel-xeon-processor-e5-1620-10m-cache-3-60-ghz-0-0-gt-s-intel-qpi.html.

[22] Intel. 2016. Intel atom x5-z8350 processor. Retrieved from https://ark.intel.com/content/www/us/en/ark/products/93361/intel-atom-x5-z8350-processor-2m-cache-up-to-1-92-ghz.html.

[23] Intel. 2017. Intel Core i7-4790 Processor. Retrieved from https://ark.intel.com/content/www/us/en/ark/products/80806/intel-core-i7-4790-processor-8m-cache-up-to-4-00-ghz.html.

[24] George Kamiya. 2020. Data centres and data transmission networks - Analysis - IEA. Retrieved from https://www.iea.org/reports/data-centres-and-data-transmission-networks.

[25] Elia Kaufmann, Antonio Loquercio, René Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. 2018. Deep drone racing: Learning agile flight in dynamic environments. Retrieved from http://arxiv.org/abs/1806.08548.

[26] Dong-Hyun Kim, Yong-Guk Go, and Soo-Mi Choi. 2018. First-person-view drone flying in mixed reality. In *SIGGRAPH Asia Posters (SA'18)*. Association for Computing Machinery, New York, NY. DOI: https://doi.org/10.1145/3283289.3283346

[27] H. Kim, H. Nam, W. Jung, and J. Lee. 2017. Performance analysis of CNN frameworks for GPUs. In *the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS'17)*. 55–64.

[28] Bell Labs. 2018. Plan 9 from Bell Labs. Retrieved from https://9p.io/plan9/about.html.

[29] Y. Lai, C. Ho, Y. Huang, C. Huang, Y. Kuo, and Y. Chung. 2018. Intelligent vehicle collision-avoidance system with deep learning. In *the IEEE Asia Pacific Conference on Circuits and Systems (APCCAS'18)*. 123–126.

[30] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (May 2015), 436–444. DOI: https://doi.org/10.1038/nature14539

[31] S. Lee, H. Cho, Y. H. Son, Y. Ro, N. S. Kim, and J. H. Ahn. 2018. Leveraging power-performance relationship of energy-efficient modern DRAM devices. *IEEE Access* (2018), 31387–31398.

[32] Yuyang Liu, Ce Zhu, Min Mao, Fangliang Song, Frederic Dufaux, and Xiang Zhang. 2018. Video analytical coding: When video coding meets video analysis. *Sig. Proc.: Image Commun.* 67 (2018), 48 – 57. DOI: https://doi.org/10.1016/j.image.2018.05.012

[33] P. S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hallberg, J. Hogberg, F. Larsson, A. Moestedt, and B. Werner. 2002. Simics: A full system simulation platform. *Computer* 35, 2 (2002), 50–58.

[34] John D. McCalpin. 1995. Memory bandwidth and machine balance in current high performance computers. *IEEE Comput. Soc. Tech. Committ. Comput. Archit. Newslett.* (Dec. 1995), 19–25.

[35] Anup Mohan, Ahmed S. Kaseb, Kent W. Gauen, Yung-Hsiang Lu, Amy R. Reibman, and Thomas J. Hacker. 2018. Determining the necessary frame rate of video data for object tracking under accuracy constraints. In *the IEEE Conference on Multimedia Information Processing and Retrieval (MIPR'18)*. IEEE. DOI: https://doi.org/10.1109/mipr.2018.00081

[36] Viacheslav Moskalenko, Alona Moskalenko, Artem Korobov, and Viktor Semashko. 2018. The model and training algorithm of compact drone autonomous visual navigation system. *Data* 4, 1 (Dec. 2018), 4. DOI: https://doi.org/10.3390/data4010004

[37] Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G. Azzolini, Dmytro Dzhulgakov, Andrey Mallevich, Ilia Cherniavskii, Yinghai Lu, Raghuraman Krishnamoorthi, Ansha Yu, Volodymyr Kondratenko, Stephanie Pereira, Xianjie Chen, Wenlin Chen, Vijay Rao, Bill Jia, Liang Xiong, and Misha Smelyanskiy. 2019. Deep Learning Recommendation Model for Personalization and Recommendation Systems. Retrieved from arXiv:cs.IR/1906.00091.

[38] Nicholas Nethercote and Julian Seward. 2007. Valgrind: A framework for heavyweight dynamic binary instrumentation. In *the SIGPLAN International Conference on Programming Language Design and Implementation*. 89–100.

[39] Nvidia. 2019. Nvidia TensorRT. Retrieved from https://github.com/NVIDIA/TensorRT. ([n. d.]).

[40] Nvidia. 2019. NVIDIA Jetson Nano System-on-module data sheet. Retrieved from https://developer.nvidia.com/embedded/dlc/jetson-nano-system-module-datasheet.

[41] Nvidia. 2019. Nvidia Jetson Nano. Retrieved from https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/.

[42] OSDev. 2017. Virtio. Retrieved from https://wiki.osdev.org/Virtio.

[43] Mike O'Connor, Niladrish Chatterjee, Donghyuk Lee, John Wilson, Aditya Agrawal, Stephen W. Keckler, and William J. Dally. 2017. Fine-grained DRAM: Energy-efficient DRAM for extreme bandwidth systems. In *the International Symposium on Microarchitecture (MICRO'17)*. 41–54.

[44] A. Pahlevan, Y. M. Qureshi, M. Zapater, A. Bartolini, D. Rossi, L. Benini, and D. Atienza. 2018. Energy proportionality in near-threshold computing servers and cloud data centers: Consolidating or Not? In *the Design, Automation and Test in Europe Conference*. 147–152.

[45] Jeng-Shyang Pan, S. Ma, S.-H. Chen, and C.-S. Yang. 2015. Vision-based vehicle forward collision warning system using optical flow algorithm. *J. Inf. Hiding Multim. Sig. Proc.* 6 (07 2015), 1029–1042.

[46] G. Prabhakar, B. Kailath, S. Natarajan, and R. Kumar. 2017. Obstacle detection and classification using deep learning for tracking in high-speed autonomous driving. In *the IEEE Region 10 Symposium (TENSYMP'17)*. 1–6.

[47] Yasir Qureshi, William Simon, Marina Zapater, Katzalin Olcoz, and David Atienza. 2020. Gem5-X Full System Manual. Retrieved from https://eslweb.epfl.ch/masters/img/20200814gem5%_X_TechnicalManual_v1.pdf.

[48] Yasir Mahmood Qureshi. 2020. Gem5-X: A gem5-based simulator with architectural eXtensions. Retrieved from https://esl.epfl.ch/gem5-x.

[49] Y. M. Qureshi, J. M. Herruzo, M. Zapater, K. Olcoz, S. Gonzalez Navarro, O. Plata, and D. Atienza. 2020. Genome sequence alignment—Design space exploration for optimal performance and energy architectures. *IEEE Trans. Comput.* (2020), 1–1. DOI: https://doi.org/10.1109/TC.2020.3041402

[50] Yasir Mahmood Qureshi, William Andrew Simon, Marina Zapater, David Atienza, and Katzalin Olcoz. 2019. GEM5-X: A GEM5-based system level simulation framework to optimize many-core platforms. In *the High Performance Computing Symposium (HPC'19)*. Society for Computer Simulation International.

[51] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen. 2018. DeepDecision: A mobile deep learning framework for edge video analytics. In *the IEEE Conference on Computer Communications*. 1421–1429.

[52] B. K. Reddy, M. J. Walker, D. Balsamo, S. Diestelhorst, B. M. Al-Hashimi, and G. V. Merrett. 2017. Empirical CPU power modelling and estimation in the gem5 simulator. In *the International Conference on Power and Timing Optimization and Simulation*. 1–8. DOI: https://doi.org/10.1109/PATMOS.2017.8106988

[53] SANDVINE. 2018. Global internet phenomena report. 2018. Retrieved from https://www.sandvine.com/hubfs/downloads/phenomena/2018-phenomena-report.pdf.

[54] H. Shim, S. Lee, Y. Woo, M. Chung, J. Lee, and C. Kyung. 2006. Cycle-accurate Verification of AHB-based RTL IP with transaction-level system environment. In *the International Symposium on VLSI Design, Automation and Test (VLSI-DAT'06)*. 1–4. DOI: https://doi.org/10.1109/VDAT.2006.258143

[55] William Andrew Simon, Yasir Mahmood Qureshi, Alexandre Levisse, Marina Zapater, and David Atienza. 2019. BLADE: A bitline accelerator for devices on the edge. In *the Great Lakes Symposium on VLSI (GLSVLSI'19)*. Association for Computing Machinery, New York, NY, 207–212.

[56] W. A. Simon, Y. M. Qureshi, M. Rios, A. Levisse, M. Zapater, and D. Atienza. 2020. BLADE: An in-cache computing architecture for edge devices. *IEEE Trans. Comput.* 69, 9 (2020), 1349–1363. DOI: https://doi.org/10.1109/TC.2020.2972528

[57] A. Sobti, C. Arora, and M. Balakrishnan. 2018. Object detection in real-time systems: Going beyond precision. In *the IEEE Winter Conference on Applications of Computer Vision (WACV'18)*. 1020–1028.

[58] K. Sohn, W. Yun, R. Oh, C. Oh, S. Seo, M. Park, D. Shin, W. Jung, S. Shin, J. Ryu, H. Yu, J. Jung, H. Lee, S. Kang, Y. Sohn, J. Choi, Y. Bae, S. Jang, and G. Jin. 2017. A 1.2 V 20 nm 307 GB/s HBM DRAM With at-speed wafer-level I/O test scheme and adaptive refresh considering temperature distribution. *IEEE J. Solid-State Circ.* (Jan. 2017), 250–260.

[59] R. Varona-Gómez and E. Villar. 2009. AADL simulation and performance analysis in SystemC. In *the 14th IEEE International Conference on Engineering of Complex Computer Systems*. 323–328.
[60] Marko Viitanen, Ari Koivula, Ari Lemmetti, Arttu Ylä-Outinen, Jarno Vanne, and Timo D. Hämäläinen. 2016. Kvazaar: Open-source HEVC/H. 265 encoder. In *the Multimedia Conference*. 1179–1182.
[61] J. Wang, Z. Feng, Z. Chen, S. George, M. Bala, P. Pillai, S. Yang, and M. Satyanarayanan. 2018. Bandwidth-efficient live video analytics for drones via edge computing. In *the IEEE/ACM Symposium on Edge Computing (SEC'18)*. 159–173.
[62] Leyuan Wang, Zhi Chen, Yizhi Liu, Yao Wang, Lianmin Zheng, Mu Li, and Yida Wang. 2019. A unified optimization approach for CNN model inference on integrated GPUs. In *the 48th International Conference on Parallel Processing (ICPP'19)*. Association for Computing Machinery, New York, NY.

Received August 2020; revised March 2021; accepted April 2021