GEM5-X: A GEM5-BASED SYSTEM LEVEL SIMULATION FRAMEWORK TO OPTIMIZE MANY-CORE PLATFORMS

Yasir Mahmood Qureshi
William Andrew Simon
Marina Zapater
David Atienza

Embedded Systems Lab (ESL)
Swiss Federal Institute of Technology Lausanne (EPFL)
1015 Lausanne, Switzerland
{yasir.qureshi,william.simon,marina.zapater,david.atienza}@epfl.ch

Katzalin Olcoz

Department of Computer Architecture
Complutense University of Madrid
28040 Madrid, Spain
katzalin@ucm.es

ABSTRACT

The rapid expansion of online-based services requires novel energy and performance efficient architectures to meet power and latency constraints. Fast architectural exploration has become a key enabler in the proposal of architectural innovation. In this paper, we present gem5-X, a gem5-based system level simulation framework, and a methodology to optimize many-core systems for performance and power. As real-life case studies of many-core server workloads, we use real-time video transcoding and image classification using convolutional neural networks (CNNs). Gem5-X allows us to identify bottlenecks and evaluate the potential benefits of architectural extensions such as in-cache computing and 3D stacked High Bandwidth Memory. For real-time video transcoding, we achieve 15% speed-up using in-order cores with in-cache computing when compared to a baseline in-order system and 76% energy savings when compared to an Out-of-Order system. When using HBM, we further accelerate real-time transcoding and CNNs by up to 7% and 8% respectively.

Keywords: many-core, architectural exploration, gem5, in-cache, HBM.

1 INTRODUCTION

On-line based services are experiencing an unprecedented growth. Among them, streaming services and machine-learning video analytics represent a wide range of applications like traffic control, surveillance and security, self driving cars, personal digital assistants, augmented reality, etc. This results in an escalating demand to meet the power and performance requirements of the server platforms hosting these services. These platforms must be able to serve a diverse range of multi-threaded applications to multiple users simultaneously. To meet the performance and Quality-of-Service (QoS) constraints, servers are comprised of many-core processors as described by Hwu et al. (2008). The processor cores might be asymmetric with different micro-architectures, such as in-order and out-of-order (OoO) cores used by ARM (2013), with thread level allocation policies appropriately allocating the workload to respective cores, meeting QoS constraints under limited power budget. To optimize energy efficiency and performance, a system level simulator is required, capable of simultaneously executing multi-threaded applications in parallel on a many-core system. Using such a simulation framework, we can identify application bottlenecks and develop fast architectural exploration methodologies to assess novel techniques at the system level.

In this paper we present gem5-X (*"a gem5-based full-system simulator with architectural eXtensions*"), a simulation framework that enables fast profiling, architectural exploration and performance-power characterization for system level architectural innovations. Gem5-X enables seamless simulation of applications running on top of a modern Linux operating system. Without loss of generality, to demonstrate the gem5-X framework and methodology, we use two applications as case studies in this paper: (i) real-time video transcoding and (ii) image classification using convolutional neural networks (CNNs). We chose video transcoding, as video streaming represents 58% of the overall downstream traffic in 2018 as presented by SANDVINE (2018). Similarly, the choice for CNNs, specifically Alexnet by Krizhevsky et al. (2012), is driven by the emergence of real-time video analytics, suggested by Ananthanarayanan et al. (2017) to be the "killer app" for artificial intelligence, with already a great impact in various domains of our daily life.

We assess the bottlenecks of real-time transcoding when running on ARM-64 in-order and OoO architectures equipped with the NEON SIMD accelerator. To demonstrate the capabilities of gem5-X for exploiting architectural extensions, we implement a novel in-cache computing as proposed by Simon et al. (2019) architecture as a use case in gem5-X, which executes a large number of operations simultaneously in-cache. We assess the benefits of in-cache computing in terms of performance and energy consumption from the system-level perspective across varying frequencies and core counts, within a fixed area limit. We further evaluate the new bottlenecks created when running real-time transcoding utilizing in-cache computing along with Alexnet. As a second case study to demonstrate gem5-X architectural extension capabilities, we implement a High Bandwidth Memory (HBM) based on 3D die stacking technology. We implement the HBM2 memory model as proposed by Sohn et al. (2017), which has bandwidth of 307 GB/s in comparison to 25.2 GB/s for DDR4 architectures described by Hsueh et al. (2014) to alleviate the memory bottlenecks generated due to the concurrent run of multiple memory bounded applications.

In particular, the contributions of our work are as follows:

- We present the gem5-X framework, which enhances gem5 developed by Binkert et al. (2011) with a full-system simulation of ARM-64 in-order and OoO architectures running on a modern Linux OS. We tune and validate performance of in-order and OoO against a real ARM JUNO platform developed by ARM (2015) with a mean absolute error (MAE) below 4%. Gem5-X is open-sourced to the community, enabling innovative extensions of ARM 64-bit architectures.
- We propose the gem5-X methodology to profile applications within gem5, identify bottlenecks, and validate architectural modifications and extensions for application acceleration. We use real-time video transcoding and Alexnet as application case studies, and in-cache computing and HBM2 as architectural extension case studies to validate the gem5-X methodology and simulation framework.
- We show how in-cache computing provides a speed-up of 62% to 88% for the video transcoding critical path function, resulting in an overall application speed-up of 15% and an energy efficiency improvement of up to 80%, for an in-order system. Our results also reveal that an 8-core in-order system with in-cache computing outperforms the energy consumption of the 4-core OoO system by up to 76% (both with SIMD acceleration), while meeting latency constraints (i.e., transcoding at a 24 frame-per-second –FPS– rate). HBM2 alleviates the memory bottlenecks generated when co-allocating video transcoding and Alexnet on the same platform, resulting in performance improvements of up to 7% for video transcoding and 8% for Alexnet.

2 RELATED WORK

The increasing complexity of emerging architectures as discussed by Shim et al. (2006) require methods for fast design-space exploration of new architectural innovations. Full-system architectural simulators tackle this challenge today by providing estimations of energy, performance and area trade-offs. Sniper, developed by Carlson et al. (2011), is a multi-core parallel simulator with fast simulation time, whose main drawback is that it only supports traditional x86 architectures. Simics, developed by Magnusson et al. (2002), provides a virtualized environment to run applications on different hardware platforms and can be combined with Simflex, developed by Hardavellas et al. (2004), to obtain timing information, but only for SPARC architectures. Gem5, as discussed in Binkert et al. (2011), supports multiple ISAs, as well as different CPU models like the in-order and OoO cores, capturing both the timing and functional behavior of a wide range of systems, making it the best candidate for architectural exploration. However, its higher accuracy and sequential nature result in slower simulation times. One of the major drawbacks of gem5 is that not all components work out-of-the-box i.e., in all simulation modes. In particular, the hybrid memory cube (HMC) is a memory type supported in gem5, but unavailable in full system (FS) mode. Similarly, system stability is not guaranteed for all component combinations. The Linux distributions and kernels provided are quite old with minimal package installation, and they can demonstrate FS mode, but are incapable of exploiting all the features in FS mode. Even so, we choose gem5 as our base simulation platform as its flexibility allows for straightforward architectural extensions. Our gem5-X simulation framework enhances gem5 with out-of-the-box system level support for many-core ARM-64 architectures and architectural innovations. In this paper, we showcase the architectural extension capabilities of gem5-X using in-cache computing and HBM2. In-cache computing allows massive Single Instruction Multiple Data (SIMD)-like operations to be performed in the cache hierarchy as proposed by Jeloka et al. (2016). In our work, we use an in-cache computing architecture similar to BLADE proposed by Simon et al. (2019), targeted for the L1 cache of ARM-based many-core systems, as opposed to the Last Level Cache (LLC), as in NeuralCache proposed by Eckert et al. (2018). Regarding HBM proposed by Lee et al. (2014), emerging memory architectures have been explored, but mainly for GPUs, as discussed in Chatterjee et al. (2017). To the best of our knowledge, this is the first work that simulates in-cache acceleration along with HBM at system level in Linux-based systems.

3 THE GEM5-X SIMULATION FRAMEWORK

The *"gem5-based full-system simulator with architectural eXtensions"*, namely gem5-X, extends the latest gem5 version (i.e., github commit a891159548) and supports out-of-the-box full-system (FS) many-core ARM simulation using a modern Ubuntu Linux distribution, as well as support for application profiling and architectural extensions. Gem5-X will be open-sourced to the community and a dockerized version will be released to enable using it out-of-the-box, enabling researchers to assess and add architectural extensions.

Figure 1 shows the contributions brought by the gem5-X simulation framework, which can be split into architectural extensions and support enhancements to the gem5 simulator.

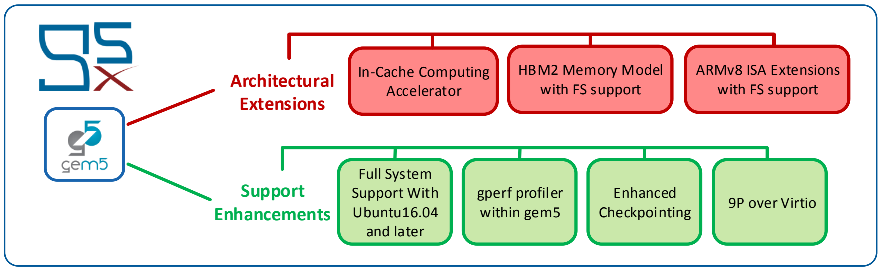

gem5
Architectural Extensions
In-Cache Computing Accelerator
HBM2 Memory Model with FS support
ARMv8 ISA Extensions with FS support
Support Enhancements
Full System Support With Ubuntu16.04 and later
gperf profiler within gem5
Enhanced Checkpointing
9P over Virtio

Figure 1: Gem5-X simulation framework

3.1 Architectural Extensions

Gem5 can be modified at any level of the architecture, from the multi-core pipeline through the interconnects and cache down to the DRAM. Gem5-X modifies the gem5 core to support extensions such as scratchpad memories, HBM2, modified CPU pipelines, in-cache computing architectures and ARMv8 ISA extensions.

All these extensions can be implemented and profiled to deduce their effectiveness for the target application. In this work, we use in-cache computing along with HBM2 as a case study to demonstrate how gem5-X can be used to simulate and explore new architectures and accelerators at the system level.

- **In-cache Computing:** To add an in-cache computing architecture to gem5, we modify gem5's L1 cache model, as discussed by Binkert et al. (2011), to simulate in-cache addition, subtraction, multiplication, shifting, greater/less than and absolute operations in a timing and architecture-accurate manner. When an in-cache instruction is decoded by the CPU, the required operations are scheduled and its operands are loaded into cache from main memory. To guarantee architectural accuracy, we enforce data locality constraints upon the operands, as discussed by Aga et al. (2017) and Simon et al. (2019). In order for an operation to be performed between two operands, they must share the same bitlines. We ensure this by reserving 1GB of cacheable memory that can be *mmapped* by an application, allowing us to have fine grained control of where operands are stored in the cache. We also modify the target application to guarantee data alignment during operations to be performed in-cache.
  To guarantee timing accuracy and estimate power consumption, we develop and simulate our in-cache architecture in 28nm bulk CMOS technology using Cadence Virtuoso as discussed by Simon et al. (2019). We extract power and timing values and convert them to cycle counts that are integrated into gem5's event scheduler.
- **High Bandwidth Memory:** HBM, as described by Lee et al. (2014), is based on 3D stacked DRAM banks made possible due to Through Silicon Vias (TSVs) achieving a high bandwidth of up to 307.2 GB/s. To implement the functional behavior of the HBM2 memory model in gem5-X, we extend the DRAM controller model of gem5 according to the architectural details of HBM2, as summarized in Table 1. To have 8-channels with memory interleaving, we initialized 8 DRAM controllers, each 128 bits wide. We connect all 8 DRAM controllers to a 1024-bit wide system bus, that connects to the cache hierarchy. For accurate timing estimates we utilize the timing values as presented by Chatterjee et al. (2017).
  Gem5-X allows having hybrid memories i.e. both DDR and HBM in the same system, but for the case studies in this paper, we will be using either DDR4 or HBM. No separate software support is required, and hence we are able to boot the Ubuntu Linux distribution using HBM.

Table 1: Implemented HBM2 architecture

Parameter	**Value**	**Parameter**	**Value**
Clock period (tCK), Bandwidth	0.833ns, 2.4Gbps/pin	Channel width, #Channels	128 bits, 8
#I/Os	1024 pins	Ranks per channel	1
Banks per rank	16	Burst length	4
Bank groups per ranks	4	-	-

- **ARMv8 ISA Extension:** To support the in-cache computing architecture in FS mode in gem5-X, we extend the ARMv8 ISA, by ARM (2017), using reserved op-codes. The added instruction, when decoded, issues an in-cache computing request to the cache controller. To support the new instruction, the decoder in the gem5 ISA domain-specific language (DSL) was modified. We also added a new *cachecompute* flag with the instruction, so that the cache controller can recognize it as a cache compute request and handle it accordingly. This instruction can be issued through in-line assembly from any C or C++ program.
  By integrating the in-cache computing architecture into gem5, its performance can be measured on top of a full software and Linux kernel stack, allowing events such as context switching, cache line eviction, and performance loss due to complex data accesses to be evaluated. This is important when assessing the real-world applicability of any architectural innovation, and its ability to generalize to other applications.

3.2 Support Enhancements

To enable gem5-X support of all architectural extensions from the software perspective, various support enhancements are added to gem5.

- **ARM-64 Full System Support:** Current gem5 disk images utilize an Ubuntu 14.04 distribution with minimal installed packages and HDD space limited to 3GB, therefore providing limited support for ARMv8 (aarch64). In this work, we create an Ubuntu 16.04 LTS disk image with kernel v4.3, and 30GB storage to allow complete full system (FS) support, including the *pthread* library to allow thread-level parallelism on a multi-core system, which is required by most applications. This extension to gem5 allows the installation and execution of any application that runs on a regular Linux system, using both in-order and OoO ARMv8 cores.
- **Gperf Profiler:** We include profiling capabilities within FS, by installing the gperf profiler on the disk image. The *gperf* statistical profiler developed by Google (2011) provides profiling capabilities on gem5 itself with minimal overhead, enabling the identification of application bottlenecks and exploration of the effectiveness of architectural modifications and extensions.
- **Enhanced Checkpointing:** Checkpointing in gem5 drastically reduces simulation time in FS mode. Gem5-X enhances the existing checkpointing in gem5 by marking the Region-Of-Interest (ROI) of applications. Gem5-X simulations are launched using a simple functional CPU, run until the ROI and checkpointed. Then, simulations are switched to either in-order or OoO detailed models. Moreover, checkpointing reduces the burden of the debugging process, by checkpointing just before the point-of-failure and then resuming with the debug mode. Using this for the video transcoding, we reduced the simulation from 10 hours to 2 hours in the worst case-scenario for the debug process.
- **9P over Virtio:** We utilize the 9P protocol developed by Bell Labs (2018) over a virtio device driver developed by OSDev (2017), to allow fast modification of files without modifying the root file system in gem5-X. While this feature is available in vanilla gem5, it is not enabled by default and has no kernel support. Both of these features are provided in gem5-X. Once Linux is booted, a folder on the host machine can be mounted within gem5 to access files on the host system. Without 9P mounting, every time a program is modified, we need to reload the disk image required for FS simulation and reboot Linux. In gem5, this process can take up to 20-30 minutes, a bottleneck that gem5-X eliminates.

4 METHODOLOGY FOR ARCHITECTURAL EXPLORATION AND OPTIMIZATION

Using gem5-X, we present an architectural exploration and a flexible optimization methodology for any given application, as shown in Figure 2. The methodology has 3 phases: application characterization, architecture optimization and milestones. Each phase is then further divided into different stages. The different phases of the methodology are discussed in this section.

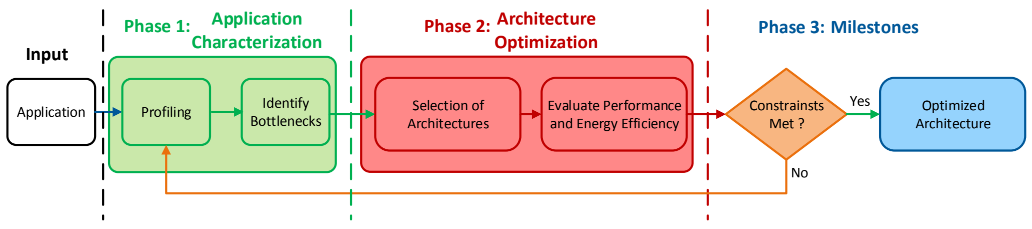

Phase 1: Application Characterization
Phase 2: Architecture Optimization
Phase 3: Milestones
Input
Application
Profiling
Identify Bottlenecks
Selection of Architectures
Evaluate Performance and Energy Efficiency
Constrainsts Met ?
Yes
No
Optimized Architecture

Figure 2: Architectural exploration methodology

4.1 Phase 1: Application Characterization

To optimize an architecture for any given application, we first profile it to identify bottlenecks.

- **Profiling:** Profiling provides two important insights. Firstly, it identifies the application kernels that are compute and memory intensive, and secondly, it gives information about the system resources being stressed by these kernels. In the gem5-X methodology, we use valgrind, developed by Nethercote and Seward (2007), when profiling on a hardware platform to collect performance counter statistics (e.g., instruction counts, cache misses, etc.). Valgrind adds some code instrumentation to the application, hence it is very slow. Therefore, we also use *gperf* statistical profiler developed by Google (2011) on both hardware and in the gem5 simulator to profile applications with minimal profiling overhead, and generate call graph trees showing what percentage of the total time each kernel takes.
- **Identify Bottlenecks:** Once we have the profiling information, we can identify the kernels that consume most of the execution time. Then using the profiling data, we can further analyze resource utilization by these kernels, giving us insight about the resources that are bottle-necked.

4.2 Phase 2: Architecture Optimization

Once the application is characterized and bottlenecks are identified, we optimize the architecture by alleviating the bottlenecks and improving performance and energy efficiency using a two-stage process.

4.2.1 Selection of Architectures

Gem5-X enables different strategies and allows to select from a range of architectural extensions, which comprise of but are not limited to, those discussed in Section 3.1.

- **Architecture Parameter Sweeping:** To accelerate the target application one can sweep through a number of parameters such as clock frequencies, cache sizes (at all levels like at L1, LLC), depth of the cache hierarchy, cache and DRAM latency and bandwidth, and number of integer or floating point functional units. As a case study in this paper, we sweep through different cache sizes, as discussed in Section 5.4.1.
- **Type and Number of Cores:** Selection of the type of cores between simple in-order cores and complex OoO cores and also the core count for each core type can be varied to meet required performance and power constraints. Simple in-order cores have low performance but consume low power as well in comparison to OoO cores.
- **Accelerators:** Accelerators can be used to overcome compute bottlenecks in an application. The choice and selection of accelerator is driven by profiling data. One of the accelerators enabled by gem5-X is the in-cache computing architecture discussed in Section 3.1. If a kernel involves many operations upon the same data chunk, resulting in low memory access times but high computation cost and processor-cache traffic, it is a good candidate for in-cache acceleration. We will discuss this in more detail in Section 5.4.2.
- **Higher Bandwidth Memories:** If the system is bottlenecked at the main memory, due to low available memory bandwidth, the HBM presented in Section 3.1 provides an alternative to speed-up the system by easing the memory bottleneck.

4.2.2 Evaluate Performance and Energy Efficiency

Once we have optimized the architecture using the strategies discussed in the previous section, we evaluate the performance and energy efficiency improvements achieved at the system level as well as application level. We also look at the cost in terms of area of the optimized system.

4.3 Phase 3: Milestones

In the third and final phase of gem5-X methodology, we check if the optimized system meets the power, performance and area constraints. We also co-locate different applications, in our case study real-time video transcoding and Alexnet, and check if the performance remains the same or is degraded, due to interference between applications. If any one of the constraints is not achieved in any of the cases, we go back to phase-1 and iterate over the whole methodology again. We iterate until all the constraints are met and milestones achieved, obtaining an optimized architecture. If multiple architectures meet the constraints, we use Pareto optimal architecture points, w.r.t., power, performance and area.

5 EXPERIMENTAL SETUP AND RESULTS

5.1 Validation of the Simulation Framework

To validate our framework against real hardware, we configure and tune gem5-X to match the architecture of the in-order cores as well as OoO cores of the ARM JUNO platform from ARM (2015) which is based on ARMv8 64-bit ISA. The starting points for the values of these parameters were ARM Cortex-A57 as presented in Bolaria (2012) for OoO cores and ARM Cortex-A53 as presented in Krewell (2012) for in-order cores. We tuned the gem5 in-order and OoO core parameters, specifically, the of number of fetch units, executions stages, and number of functional units and their latencies, until we obtained a 4% error compared to the JUNO platform in terms of execution time for a real-time video transcoding app Kvazaar, developed by Viitanen et al. (2016), using different video resolutions, as described in Section 5.2, using all the components (all CPU functional units, including NEON SIMD, caches, memory). We do not tune the network and disk models in gem5-X as the purpose of this paper is to explore novel compute and memory architecture. Therefore, our case study applications are compute and memory intensive, and we use application run-time to quantify both of them during validation. As a result, validation of the simulation framework ensures full confidence in the results provided by gem5-X.

5.2 Experimental Setup

- **Applications:** We use two applications; the real-time video transcoding app Kvazaar, and image classification with CNNs (Alexnet), due to their current relevance, as described in section 1. To demonstrate our methodology, we run experiments with three types of videos using Kvazaar, w.r.t. their resolution: high, medium and low, which correspond to 1920x1080, 416x240 and 176x144 pixels, respectively. We utilize an Alexnet model in the ARM Compute Library framework (ACL), developed by ARM (2018), for our CNN experiments.
- **Power Model:** To compute energy values, we use the power model for 28nm bulk CMOS A57 OoO cores proposed by Pahlevan et al. (2018). For in-order cores, we use the energy ratio between A57 and A53 cores at different frequencies as proposed by Frumusanu and Smith (2015a), Frumusanu and Smith (2015b). The power model accounts for core active, wait-for-memory (WFM) and static energy (in J/cycle), and the LLC read and write energy (in J/access). We did not use gem5 power model, as proposed by Reddy et al. (2017) or McPAT power model, as proposed by (Butko et al. 2016), as they both are for ARMv7 32-bit ISA, whereas we are using ARMv8 64-bit cores.
- **Hardware Architecture:** As a starting point for our exploration, we model the ARM JUNO platform in gem5-X, with 4 OoO cores instead of 2 to have a fair comparison between different architectures and core types, each with 4 cores as a starting point. The simulated architecture is summarized in Table 2.

Table 2: Initial architecture for simulation framework

Parameter	**Value**	**Parameter**	**Value**
Core ISA	ARMv8 64-bit	# In-order, # OoO cores	4, 4
Core Frequency	2GHz	DDR4 size	4GB
L1-I cache, L1-D cache	32kB, 32kB	LLC	1MB

5.3 Profiling and Bottlenecks

The first step in the gem5-X methodology is to profile the target application to identify memory and compute bottlenecks. We first use Valgrind on the JUNO platform to profile Kvazaar. Our results show that the FIR filter and the Sum of Absolute Transform Differences (SATD) are the two main blocks in the application 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%). L1 cache read miss counts for FIR filter and SATD blocks were as low as 4.8% and 5.2% respectively, demonstrating high data locality. We also profile Kvazaar on the ARMv8 64-bit JUNO platform in both in-order and OoO using gperf. The profiling results are consistent in both types of cores as well as across the profiler used, in terms of the bottlenecked functions. Moreover, percentages only differ between 1% to 4% from valgrind to gperf due to gperf's statistical profiling.

Profiling demonstrates that the FIR filter and SATD blocks are the primary bottlenecks in Kvazaar. Due to the high locality in the L1 cache and the relatively simple arithmetic operations they perform, they are potential candidates for our case study architectural extension of in-cache computing.

5.4 Strategies for Architecture Optimization

5.4.1 Sweeping the Cache Sizes

As discussed in the previous section, FIR filter and SATD, the primary bottlenecks in Kvazaar, exhibit high cache locality. To explore the effect of cache size, we use gperf in gem5-X and vary the size of the L1 and LLC. We observe that for an L1 of 32KB, varying the LLC from 512KB to 16MB provides 6% application speed-up. Similarly, for an LLC of 16MB, increasing the L1 from 8KB to 128KB provides a 3.2% speed-up. As the speed-up is not significant, indicating that data fits in all cache sizes, we select a 32KB L1 cache and a 1MB LLC for the remainder of the paper, as they represent an adequate trade-off between energy, performance, and area.

5.4.2 Acceleration with In-cache Computing

Table 3 shows the speed-up of the individual FIR filter and the overall application speed-up obtained via in-cache computing with 4 in-order cores, using gem5-X. Note that all comparisons in this section are against in-order (and OoO) architectures equipped with a Neon SIMD accelerator. The FIR filter block is accelerated by 62% to 88% depending on the video resolution. The SATD acceleration is lower, reaching ~5% maximum, as not all SATD blocks can be accelerated, and because of the alignment constraints on the operands for in-cache computing. However, accelerating SATD blocks is still very beneficial in terms of energy. Accelerating both the FIR filter and the SATD block gives us ~15% application speed-up across low, medium and high resolution videos compared to an in-order core without in-cache computing. As presented in Table 3, we also get an overall energy reduction of 11.5% to 16.47%, for different video resolutions.

5.5 Architectural Exploration

Gem5-X allows us to assess different architectures with and without in-cache computing. Our goal is to minimize the energy consumption while meeting the 24FPS requirement for all video resolutions.

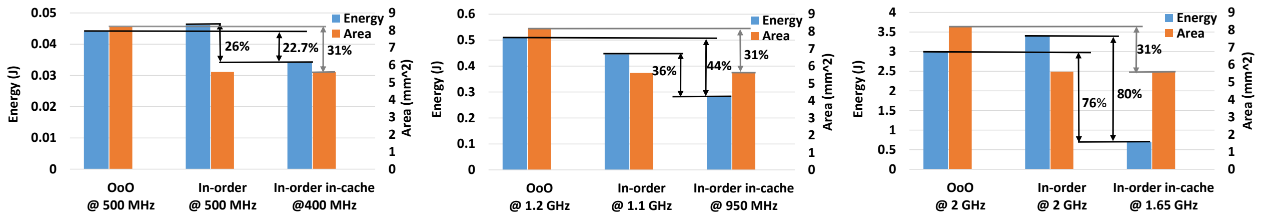
Table 3: Filter and application acceleration and energy reduction for different video resolutions

Video Resolution	Filter Speed-up	Application Speed-up	Overall Energy Reduction
Low Resolution	88%	14.4%	16.47%
Medium Resolution	62.34%	14.7%	14.1%
High Resolution	64.64%	15.36%	11.5%

We start by assessing all configurations capable of satisfying 24FPS for low resolution videos within a fixed area budget of 4 OoO cores, i.e 8.2, as described by *mm*^{2} A. Frumusanu and R. Smith (2015b). Figure 3a shows the energy consumption and area for various systems when running 24FPS of a low resolution video. We see that 8 in-order cores with in-cache computing at 400MHz is 22.7% and 26% more energy efficient when compared to 4 OoO cores at 500MHz and 8 in-order cores at 500MHz, respectively. Since in-order cores are 3 times smaller than OoO cores, 8 in-order cores take 31% less area w.r.t. to 4 OoO cores. The area overhead of the L1 in-cache computing unit is 0.5% of the core area using the estimates by Eckert et al. (2018) and Wikichip (2016), which is not significant. Hence, it will not be considered for the rest of the paper. The number of cores and frequency values are optimal, in terms of area and energy efficiency, when doing a sweep across different core counts at different frequencies.

Figure 3b shows the energy consumption and area for optimal in-order and in-order with in-cache computing systems in comparison to OoO system when satisfying 24FPS of a medium resolution video. We see that 8 in-order cores with in-cache computing at 950MHz gives 44% energy benefit w.r.t. 4 OoO cores at 1.2GHz and area benefit of 31%. In comparison to 8 in-order cores without in-cache computing at 1.1GHz, the in-order system with in-cache computing is 36% more energy efficient.

Finally, we assess high resolution videos, obtaining results consistent with lower resolutions, as shown in Figure 3c. In-order with in-cache (8 cores @1.65GHz) provides the best results in terms of energy efficiency, with energy savings of 76% in comparison to OoO cores (4 cores @2GHz) and 80% in comparison to in-order cores (8 cores @2GHz). The area benefit remains the same, being 31% over OoO.



(a) Low resolution (b) Medium resolution (c) High resolution

Figure 3: Energy and area comparison for different video resolutions

Figure 3 also reveals that the energy savings of in-order cores with in-cache computing increases for higher video resolution in comparison to both in-order and OoO cores without in-cache computing. This situation implies that, as the computational requirements increase, the in-cache computing performs better as it has larger data chunks to process and achieves more energy savings. The energy benefits shown in Figure 3 are attributed to the lower operating frequency used, the in-order cores and the in-cache computing unit.

5.6 Many-core Multi-Application System

To demonstrate the many-core and multi-application simulation capabilities of gem5-X, we simulate a realistic server scenario by concurrently executing Alexnet image classification inference, along with real-time transcoding. The system uses in-cache computing to accelerate the dominant blocks in Kvazaar. The in-simulator gperf profiler is useful in this case to look for new bottlenecks in the kvazaar application on the architecture with in-cache computing when co-located with Alexnet, which is memory intensive application. Analysis of the Kvazaar application for medium resolution and high resolution video reveals that memory functions, like *memcpy* and *memset* together contribute 16.5% and 10% respectively towards execution time. Profiling Alexnet reveals a 23.5% *memset* function contribution.

To accelerate the memory functions described above, HBM is used instead of the conventional DDR4 due to its high bandwidth. The experimental setup includes HBM with 8 in-order cores. Kvazaar is allocated to the 4 cores equipped with in-cache computing, while Alexnet is allocated to the remaining 4 cores, all operating at 2GHz. Table 4 shows the percentage speed-up achieved when using HBM instead of DDR. We see that Alexnet is always accelerated by more than 7%. Moreover, Kvazaar achieves a higher acceleration in case of medium resolution as compared to high resolution, because the contribution of memory functions is higher in medium resolution videos as compared to high resolution ones.

Table 4: Speed-up using HBM instead of DDR

Application	**Medium Resolution + Alexnet Speed-up**	**High Resolution + Alexnet Speed-up**
Kvazaar Speed-up	7.72%	2.8%
Alexnet Speed-up	8.35%	7.48%

6 CONCLUSION

In this paper, we have presented the gem5-X simulation framework and methodology for optimizing performance and power of a many-core system. Gem5-X extends the gem5 simulator with innovative architectural extensions. Gem5-X is generic, allowing architecture optimization for any given application. As a case study, we used gem5-X to analyze and accelerate a real-time video transcoding application. By properly selecting the optimal number, type and operating frequency of the cores, we have showed that in-order cores with in-cache computing achieve 15% speed-up w.r.t. in-order cores with SIMD, and reduce energy consumption by up to 76% compared to OoO systems, while still meeting latency constraints. To further demonstrate the system level capability of gem5-X, we co-simulated real-time transcoding along with a deep learning CNN, thus achieving performance benefits of up to 7% and 8%, respectively by alleviating the memory bottlenecks using HBM.

The gem5-X simulation framework is open-sourced to the community, together with a technical whitepaper, enabling out-of-the-box, fast simulation of many-core ARM 64-bit architectures with innovative architectural extensions. It is readily available at esl.epfl.ch/gem5-x

ACKNOWLEDGMENTS

This work has been partially supported by the EC H2020 RECIPE (GA No. 801137) project, the ERC Consolidator Grant COMPUSAPIEN (GA No. 725657), the EU FEDER and the Spanish MINECO (GA No. TIN2015-65277-R).

REFERENCES

Aga, S., S. Jeloka, A. Subramaniyan, S. Narayanasamy, D. Blaauw, and R. Das. 2017. “Compute Caches”. In *HPCA*, pp. 481–492.

Ananthanarayanan, G., P. Bahl, P. Bodík, K. Chintalapudi, M. Philipose, L. Ravindranath, and S. Sinha. 2017. “Real-Time Video Analytics: The Killer App for Edge Computing”. *Computer*, pp. 58–67.

ARM 2013. “big.LITTLE Technology Moves Towards Fully Heterogeneous Global Task Scheduling”.

ARM 2015. “ARM Versatile Express Juno r2 Development Platform”.

ARM 2017. "ARM Architecture Reference Manual ARMv8".

ARM 2018. "ARM Compute Library Framework". https://developer.arm.com/technologies/compute-library.

Binkert, N., B. Beckmann, G. Black, S. K. Reinhardt et al. 2011, August. "The Gem5 Simulator". *SIGARCH Comput. Archit. News* vol. 39 (2), pp. 1–7.

Bolaria, J. 2012. "Cortex-A57 extends ARM's Reach High-End 64-bit CPU Strives for Servers". *Microprocessor Report*.

Butko, A., F. Bruguier, A. Gamatié, G. Sassatelli et al. 2016. "Full-System Simulation of big.LITTLE Multicore Architecture for Performance and Energy Exploration". In *MCSOC*, pp. 201–208.

Carlson, T. E., W. Heirman, and L. Eeckhout. 2011. "Sniper: Exploring the Level of Abstraction for Scalable and Accurate Parallel Multi-core Simulation". In *SC*, pp. 1–12.

Chatterjee, N., M. O'Connor, D. Lee, D. R. Johnson, S. W. Keckler, M. Rhu, and W. J. Dally. 2017, Feb. "Architecting an Energy-Efficient DRAM System for GPUs". In *2017 HPCA*, pp. 73–84.

Eckert, C., X. Wang, J. Wang, A. Subramaniyan, R. Iyer, D. Sylvester, D. Blaaauw, and R. Das. 2018. "Neural Cache: Bit-Serial In-Cache Acceleration of Deep Neural Networks". In *ISCA*, pp. 383–396.

A. Frumusanu and R. Smith 2015a. "Cortex A53 - Performance and Power". https://www.anandtech.com/show/8718/the-samsung-galaxy-note-4-exynos-review/4. Accessed Sep. 11, 2018.

A. Frumusanu and R. Smith 2015b. "Cortex A57 - Performance and Power". https://www.anandtech.com/show/8718/the-samsung-galaxy-note-4-exynos-review/6. Accessed Sep. 11, 2018.

Google 2011. "gperftools". https://github.com/gperftools/gperftools.

Hardavellas, N., S. Somogyi, T. F. Wenisch, R. E. Wunderlich et al. 2004. "SimFlex: A Fast, Accurate, Flexible Full-system Simulation Framework for Performance Evaluation of Server Architecture". *SIGMETRICS Perform. Eval. Rev.*, pp. 31–34.

Hsueh, T., G. Balamurugan, J. Jaussi, S. Hyvonen et al. 2014, Feb. "26.4 A 25.6Gb/s differential and DDR4/GDDR5 dual-mode transmitter with digital clock calibration in 22nm CMOS". In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pp. 444–445.

Hwu, W.-m., K. Keutzer, and T. G. Mattson. 2008. "The Concurrency Challenge". *IEEE Des. Test*, pp. 312–320.

Jeloka, S., N. B. Akesh, D. Sylvester, and D. Blaauw. 2016. "A 28 nm Configurable Memory (TCAM/BCAM/SRAM) Using Push-Rule 6T Bit Cell Enabling Logic-in-Memory". *JSSC*, pp. 1009–1021.

Krewell, K. 2012, 11. "Cortex-A53 is ARM's Next Little Thing New CPU Core Brings 64 Bits to Big.Little, Mobile". *Microprocessor Report*.

Krizhevsky, A., I. Sutskever, and G. E. Hinton. 2012. "ImageNet Classification with Deep Convolutional Neural Networks". In *NIPS*, pp. 1097–1105.

Labs, B. 2018. "Plan 9 from Bell Labs". *URL: https://9p.io/plan9/about.html*.

Lee, D. U., K. W. Kim, K. W. Kim, H. Kim et al. 2014. "25.2 A 1.2V 8Gb 8-channel 128GB/s high-bandwidth memory (HBM) stacked DRAM with effective microbump I/O test methods using 29nm process and TSV". In *ISSCC)*, pp. 432–433.

Magnusson, P. S., 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*, pp. 50–58.

Nethercote, N., and J. Seward. 2007. "Valgrind: A Framework for Heavyweight Dynamic Binary Instrumentation". In *SIGPLAN PLDI*, pp. 89–100.

OSDev 2017. "Virtio". *https://wiki.osdev.org/Virtio*.

Pahlevan, A., Y. M. Qureshi, M. Zapater, A. Bartolini et al. 2018. "Energy proportionality in near-threshold computing servers and cloud data centers: Consolidating or Not?". In *DATE*, pp. 147–152.

Reddy, B. K., 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 *PATMOS*, pp. 1–8.

SANDVINE 2018. "Global Internet Phenomena Report. 2018". *URL: https://www.sandvine.com/hubfs/downloads/phenomena/2018-phenomena-report.pdf*.

Shim, H., S. Lee, Y. Woo, M. Chung, J. Lee, and C. Kyung. 2006, April. "Cycle-accurate Verification of AHB-based RTL IP with Transaction-level System Environment". In *VLSI-DAT*, pp. 1–4.

Simon, A., J.-M. Galicia, A. Levisse, M. Zapater, and D. Atienza. 2019. "A Fast, Reliable and Wide-Voltage-Range In-Memory Computing Architecture". In *DAC*, pp. 1–6.

Simon, W., Y. M. Qureshi, A. Levisse, M. Zapater, and D. Atienza. 2019. "BLADE: A BitLine Accelerator for Devices on the Edge". In *GLSVLSI*, pp. 1–6.

Sohn, K., W. Yun, R. Oh, C. Oh et al. 2017, Jan. "A 1.2 V 20 nm 307 GB/s HBM DRAM With At-Speed Wafer-Level IO Test Scheme and Adaptive Refresh Considering Temperature Distribution". *JSSC*, pp. 250–260.

Viitanen, M., A. Koivula, A. Lemmetti, A. Ylä-Outinen, J. Vanne, and T. D. Hämäläinen. 2016. "Kvazaar: Open-Source HEVC/H. 265 Encoder". In *Multimedia Conference*, pp. 1179–1182.

Wikichip 2016. "Cortex-A53 - Microarchitectures - ARM". https://en.wikichip.org/wiki/arm_holdings/microarchitectures/cortex-a53. Accessed Jan. 7, 2019.

AUTHOR BIOGRAPHIES

YASIR MAHMOOD QURESHI received his Master degree in Embedded Computing Systems from NTNU, Trondheim in 2013. He is currently a Ph.D. student at the Electrical Engineering Doctoral program, in Embedded Systems Laboratory, EPFL. His research interests are energy efficient servers, heterogeneous compute and hybrid memory architectures. His email address is yasir.qureshi@epfl.ch.

WILLIAM ANDREW SIMON received his Master degree in Electrical Engineering, specialization in micro and nanoelectronics, from EPFL in 2017. He is currently a Ph.D. student in Electrical Engineering in the Embedded Systems Laboratory at EPFL. His research interests are in-memory computing, neural networks, and emerging memory architectures. His email address is william.simon@epfl.ch.

MARINA ZAPATER is a Post-Doctoral researcher in ESL-EPFL since 2016. She received her Ph.D. degree in Electronic Engineering from Universidad Politécnica de Madrid, Spain, in 2015. Her research interests include thermal and power optimization of complex heterogeneous systems, and energy efficiency in novel architectures, servers and data centers. Her email address is marina.zapater@epfl.ch.

KATZALIN OLCOZ received a Ph.D. degree in Physics in 1997 from the Complutense University of Madrid. She is currently Associate Professor within the Department of Computer Architecture and System Engineering of the UCM. Her current research addresses emerging issues related to asymmetric processors, heterogeneous systems and energy-aware computing, with a special emphasis on the interaction between the system software and the underlying architecture. Her email address is katzalin@ucm.es.

DAVID ATIENZA received his Ph.D. degree in computer science and engineering from UCM, Spain, and IMEC, Belgium, in 2005. He is currently an Associate Professor in electrical and computer engineering, and the Director of the Embedded Systems Laboratory at EPFL, Switzerland. His research interests include system-level design methodologies for multiprocessor system-on-chip (MPSoC) and low-power embedded systems, many-core servers and ultra-low-power system architectures for IoT systems. His email address is david.atienza@epfl.ch.