CECS

CENTER FOR EMBEDDED & CYBER-PHYSICAL SYSTEMS

Design Space Exploration framework for memory exploration in heterogeneous architectures

Kasra Moazzemi

Chen-Ying Hsieh

Prof. Nikil Dutt

Center for Embedded and Cyber-Physical Systems

University of California, Irvine

Irvine, CA 92697-2620, USA

{Moazzemi, chenyinh, dutt}@uci.edu

CECS Technical Report <16-03>

June, 2016

Design Space Exploration framework for memory exploration in heterogeneous architectures

Kasra Moazzemi
Department of Electrical Engineering and Computer Science
University of California Irvine
Irvine,
Email:moazzemi@uci.edu

Chen-Ying Hsieh
Donald Bren School of Information and Computer Science
University of California Irvine
Irvine,
Email:chenyinh@uci.edu

Nikil Dutt
Donald Bren School of Information and Computer Science
University of California Irvine
Irvine,
Email:dutt@ics.uci.edu

***Abstract*—The increasing amount of computation in heterogeneous architectures (including CPU and GPU cores) puts a big burden on memory subsystem. With the gap between compute units and the memory performance getting wider, designing a platform with a responsive memory system becomes more challenging. This issue is exacerbated when memory systems have to satisfy a high volume of traffic generated from heterogeneous compute units. Furthermore, as emerging memory technologies are being introduced to address these issues, a rapid and flexible mechanism is needed to evaluate these technologies in the context of heterogeneous architectures. This paper proposes HAMEX, a framework that enables early design space exploration of heterogeneous systems with a focus on resolving memory access bottlenecks. This framework first allows system designers to easily model heterogeneous architectures that can run both CPU and GPU workloads. Next, given a set of workloads partitioned on various compute units, traffic generated by these units are captured in order to explore different memory systems. We show the feasibility of design space exploration using HAMEX by simulating a contemporary commercial heterogeneous platform and explore the opportunities for power and performance improvements by adopting different memory technologies.**

I. Introduction

With the diverse application demand in embedded and even high end IOT domains, heterogeneous systems are becoming popular more than ever. Heterogeneity in these systems is manifested in many different parts. It can show itself in computational units; for example heterogeneous processors that accommodate different types of CPU cores or in architectures with various accelerators such as GPU that can perform high throughput computation. Heterogeneity also appears in different interconnect and memory modules, particularly with the availability of many new memory technologies (e.g., NVM, HMC, etc.). Figure 1 depicts a general picture of a heterogeneous computer architecture composed of heterogeneous CPUs, GPUs, accelerators and memories.

Although many existing tools are available to explore and design each component of Figure 1, there is lack of a framework that enables designers to explore rapidly the combined system ensemble of heterogeneous CPUs, GPUs, memories and interconnects. This work presents HAMEX, a Heterogeneous Architecture and Memory EXploration framework that addresses this need and empowers designers to perform rapid Design Space Exploration (DSE) of alternative heterogeneous system architectural configurations which covers the CPU, GPU, memory and interconnect dimensions.

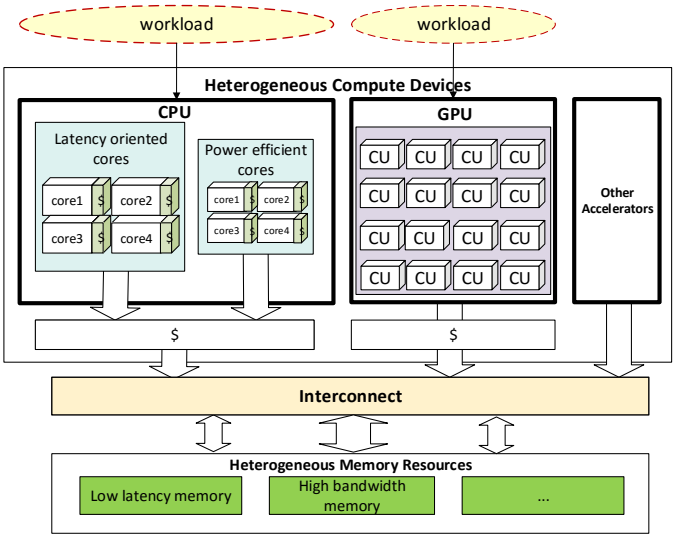


Fig. 1: General overview of a heterogeneous architecture

The contributions of this work are the following:

- We present a framework to model advanced heterogeneous systems that can include many-core processors, multiple GPUs and heterogeneous memory families.
- We capture the memory footprint of actual benchmarks executed on heterogeneous compute units to analyze memory access phases and bandwidth bottlenecks.
- We present a comprehensive power and performance exploration method for the memory subsystem to compare common or even emerging memory technologies.
- We model and analyze a contemporary heterogeneous SoC (ODROID-XU3) in this framework and demonstrate opportunities for improvements in memory organization.

The rest of the paper is organized as follows: Section 2 surveys the related tools and frameworks for architecture exploration. Section 3 describes the HAMEX design space exploration framework and its individual components. Section

4 shows trhe configurations of the HAMEX framework and experimental results and Section 5 finally concludes the paper.

II. Background and Related Work

Heterogeneous architectures are now becoming ubiquitous in the computing landscape. For instance, GPGPUs are used as accelerators for data parallel tasks, executing along side latency-oriented and power efficient cores. Furthermore, since low power consumption is one of the major design constraints in most computer systems nowadays, the trend towards heterogeneous platforms comprising CPUs, GPUs and other accelerators will continue.

One of the main challenges faced in designing heterogeneous architectures is the memory bottleneck problem. The increasing gap between the bandwidth requirements of recent heterogeneous architectures and data rate delivered by the main memory worsens the classic Memory Wall [1] problem. This problem is exacerbated in systems including different types of compute units that can access memory simultaneously. This race over accessing memory modules becomes more critical in SoCs that have power limitations and restricted memory bandwidth.

In order to increase the memory efficiency in new systems and to keep up with high demand of memory traffic, many emerging memory technologies such as NVM, PCM, HBM and HMC have been introduced. Each memory family comes with its own unique characteristics. Some try to reduce the energy consumption of memory subsystem, while others focus on providing better performance by reducing memory access latency or increasing bandwidth. Thus there is a critical need to enable comparative evaluation of memory configuration and architectures early in the design stage. Although many works such as [13][6] tried to simulate these memory modules, there is lack of a holistic framework for simulating complete unmodified workloads on heterogeneous cores and comparing memory subsystems based on applications demand, and thus posing challenges for early design exploration of a heterogeneous system.

Literature about computer architecture simulators is very mature. However, frameworks that attempt to couple all components of a common heterogeneous system together are limited. To tackle the previously mentioned challenges with respect to application execution flow, performance, contention and memory bottlenecks in heterogeneous architectures with different DRAM memories, a complete exploration framework is needed. The design space exploration framework for such heterogeneous architectures should be able to reflect the effects of all components in these platforms including both CPU and GPU as well as interconnect and complex memory hierarchy. Below we briefly describe some of the related work.

CPU simulators/emulators: Many works have focused on simulation/emulation of CPU models such as PLTsim [8], Qemu [10] and MARSS [17]. These environments focus on studying architectural details of a family of CPU cores or to test and verify software written for a specific ISA. In many cases these simulators ignore the effects of memory accesses. PLTsim [8] models an X86 ISA in RTL-level for a cycle accurate simulations. Gem5 [2] is one of the well known cycle accurate frameworks that give designers a flexible environment for CPU simulation with a complete memory hierarchy. We benefit from accuracy and flexibility of Gem5 by integrating it into our HAMEX framework.

Memory systems: In terms of modeling DRAMs and other memory technologies CACTI [13] and DRAMsim2 [6] are some of the most cited tools. These tools enables estimation of performance and power of DRAM memories. USIMM [7] models the progression of application instructions through the reorder buffers of a multi-core processor, and manages the memory controller read/write queues. MSCsim [9] also provides a multilevel simulation tool used for modeling levels of memory hierarchy including cache, main and virtual memories.

GPU simulators: With the extensive use of GPUs in server and embedded systems, there have been some frameworks that provide simulators for further studies on these units. GPGPU-sim [12] provides a detailed simulator for running CUDA and OpenCL code. It provides a good insight about pipeline stages in GPU compute units. Multi2Sim [5] is a flexible framework for simulating GPUs to verify whether a proposed alternative design is correct. We evaluated both of these works to utilize as GPU simulator in our framework. Eventually we decided to use Multi2Sim because of its flexibility in modeling and up-to-date support.

Hybrid frameworks: A majority of the tools developed for computer architecture simulation concentrate on modeling part of a full system, but only few efforts aim to model a complete system. Due to the fact that simulating each of the components in a full system is a daunting task, hybrid simulators have been proposed in order to combine two or more tools together for a more realistic system simulation. Fusionsim [11] combines GPGPU-sim with PTLsim [8] to model a X86 system that can also run CUDA workloads using a NVidia GPU. This framework is limited to X86 families in its architectural design choices. Gem5-GPU [3] proposes a similar work to glue the Gem5 simulator with GPGPU-sim together in order to utilize a variety of CPU ISAs provided by Gem5 framework. The usage of this simulator is bounded by the memory access coming from the GPU side as they have to be trapped and transferred to memory hierarchy in CPU simulator which vastly reduces the simulation speed. An extension of Multi2Sim [4] provides a heterogeneous simulator close to our work focusing more on detailed and time consuming simulations rather than rapid exploration.

In contrast, our HAMEX exploration framework aims to support rapid design space exploration of all common components included in a modern CPU/GPU systems, with a variety of memory families. Therefore we devised an exploration mechanism for a full CPU/GPU system by utilizing Gem5 [2] as the CPU simulator and Multi2Sim [5] as the GPU simulator and analyzing their traffic using DRAMSpec [14] and DRAMPower [15]. Architectures modeled in our framework can have X86, Alpha or ARM cores as CPU and a subset of AMD

and NVidia GPUs. The modularity in our frameworks allows designers to analyze effects of other types of accelerators that they might want to have in their architecture as long as the memory accesses of these accelerators can be captured. Furthermore, Trace-based memory exploration enables us to rapidly test many combination of workloads for a variety of memory models in terms of total latency, contention, buffering delay and energy. This can speed up the test process and adaptation of new memory technologies with respect to realistic workloads.

III. HAMEX DSE FRAMEWORK

The following section explains the overall structure and role of individual components of our HAMEX design space exploration (DSE) framework. HAMEX accepts an input specification file that determines the characteristics of architecture components. In addition to properties of memory subsystem, this file includes high level information about the type and number of CPU and GPU units and their caches. We integrated cycle accurate Gem5 [2] CPU simulator and Multi2Sim [5] GPU simulator for modeling these computational units. Figure 2 shows general view of the HAMEX framework. Components defined in the input specification file can have multiple sets of options to enable design space exploration. The output of HAMEX will be generated by the memory exploration component integrated with DRAMSpec [14] and DRAMPower [15] which includes metrics related to bus contention, performance and power of memory modules examined.

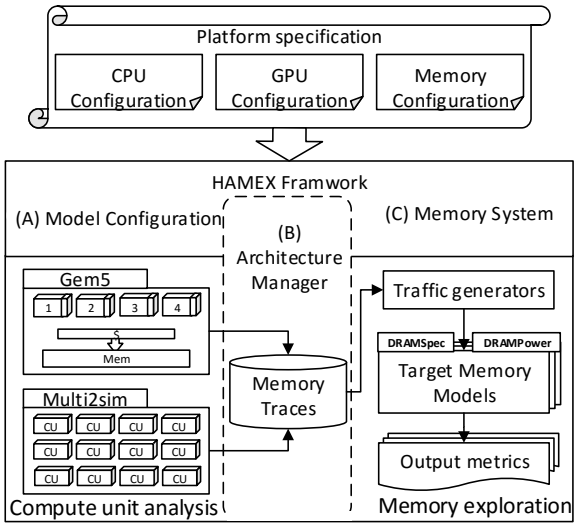


Fig. 2: Overview of HAMEX framework

A. Model configuration

1) CPU simulation: We use Gem5 [2] as our CPU simulator. Gem5 is a cycle-accurate micro-architectural simulator which provides simulation for many ISAs. This includes in-order and out-of-order cores from different ISAs that are highly configurable regarding the micro-architectural parameters such as pipeline stages, load/store queue entries and reorder buffer entries. This tool also provides flexible configurations for caches, memories and interconnects. Gem5 has two simulation modes: Full System (FS) and System-call Emulation (SE). FS mode simulates a complete system with devices and operating system. FS mode is particularly useful for designers that want to analyze effects of OS scheduling on their applications performance. In SE mode, system services are provided by the simulator hence it only needs a statically linked user-space program.

2) GPU simulation: For simulating general purpose parallel workloads, the Multi2Sim [5] GPU simulator is selected and integrated in our HAMEX framework. For this purpose, GPU and dependent modules architectural characteristics are extracted from the input specification file and fed into the simulator. This specification file determines the type and number of compute units in the GPU and their properties such as frequency, wavefronts and optional pipeline details. In addition, designers may specify the cache and main memory architecture used in GPU memory hierarchy. Multi2Sim requires a series of networks and ports to connect compute units, caches and memory modules, all of which are generated by the HAMEX architecture manager (block B in Figure 2). In order to fully model a GPU execution flow, an emulated CPU is added to simulation to start the command queue and transfer the input to GPU unit.

CPU ISA	GPU ISA	memory families
ARM	AMD Evergreen	DDR3
X86	NVIDIA Fermi	DDR4
ALPHA	NVIDA Southern Islands	LPDDR2
SPARC		GDDR5
MIPS		WideIO
POWER		HMC

TABLE I: design choices available in HAMEX provided by integrated simulation tools

B. Design choices and configurations

One of the important aspects in selecting simulation tools to integrate with HAMEX framework, was the number of available design choices in each simulation tool and their flexibility in configuration. Gem5 provides a wide variety of CPU and memory families integrated with DRAMSpec. Multi2sim also provide a set of GPU families both from NVIDIA and AMD. Flexibility of multi2sim in configuring GPU components in various metrics such as number of registers, leatencies, pipeline details, etc. makes it even more expandable for designers. Table I provides a list of available ISAs and memory families while using HAMEX framework.

C. Architecture manager

At this point in the flow of HAMEX, both CPU and GPU workloads have been simulated on their corresponding components and their memory traces have been captured. As shown in block (B) of Figure 2, the architecture manager in

addition to CPU and GPU traces, receives memory features such as type, size and bandwidth of memory from the input specification file. Using this information, the manager invokes memory exploration component comprised of DRAMSpec integrated with Gem5 memory and protocol buffer traffic generator from Google [18]. The designers have the option to specify a configuration to modify trace replay. For example the architecture manager can add delays to the start of one of the workloads or add priority to one of the compute units, then generate the improved trace for traffic generator units. Interconnect used in memory exploration also can be configured for different width, coherency, frontend and response latency, frequency, etc.

D. Memory System

One of the merits of our HAMEX framework is the ability to do extensive exploration on memory systems used in heterogeneous architectures. HAMEX supports two common early design space exploration use cases: 1) comparing common or new memory families, and 2) evaluating emerging new memory families.

1) Comparing memory components: The memory component and its interconnect used in HAMEX framework is modeled in Gem5 classic memory system integrated with DRAMSpec and DRAMPower for estimating performance and power metrics. There are many DRAM memory modules such as DDR3, LPDDR3, GDDR5, WideIO, etc. already modeled in this environment. This database can be easily expanded as described in next section. This wide variety of available of the shelf components can provide designers with more design choices to explore at early stages of design.

2) Evaluating new memories: One of the goals of HAMEX is to provide an environment to evaluate new memory technologies for heterogeneous architectures. There are two ways to add a new memory family to HAMEX for evaluation: a) Using datasheet information to model the new memory family. This information specifies the characteristics of the new memory such as burst size, address mapping, page policy, current values, etc. b) Utilizing a new memory modeled in SystemC. In sync with industry, the interconnect used in HAMEX has a SystemC TLM [19] connection that allows to connect and evaluate new memories described in SystemC.

The goal of HAMEX as an exploration framework is to provide support for DSE of systems which are comprised of many components in various architectural arrangements. Support for a vast variety of design choices in computational units comes from simulation options as described in Sections III-A1 and III-A2. In addition, flexibility in realization of interconnect and memory systems in Sections III-C and III-D would allow designers to execute their applications on standard or custom architectures. This would allow designers to rapidly and easily test their target platform in early stages of design. In terms of simulation time, using a cycle-accurate CPU simulator for analyzing effects of target benchmark suite on few memory choices can take up to hundreds of simulation hours. In contrast, our HAMEX framework simulates the benchmark suite only once and then uses the rapid memory exploration mechanism to compare the memory types and configurations. Therefore HAMEX can perform memory exploration in the order of minutes (as opposed to mentioned simulations) which saves considerable analysis time in early stages of design.

IV. Experiments

In this section, we first evaluate the robustness of the HAMEX framework and then show usefulness of HAMEX to perform early memory design space exploration.

To validate the fidelity and capability of our HAMEX framework, in Section IV-A we model a modern heterogeneous MPSoC (ODROID-XU3). In Section IV-B we analyze the effects of different classes of workloads on the memory system and then demonstrate the capability of HAMEX for exploring three alternative memory families (DDR3, LPDDR3 and HMC) in terms of latency, contention and power. Here we highlight the insights given by HAMEX for memory design space exploration.

A. *Flexibility in modeling and robustness of HAMEX*

There are two parts that we address in order to validate HAMEX as a design space exploration framework. First, we demonstrate HAMEX's ability to simulate a modern heterogeneous architecture by modeling the ODROD-XU3 MPSoC platform. Second, we evaluate the fidelity of our framework by comparing the performance of the ODROID-XU3 computational units against our modeled simulations.

*1) **Modeling a modern heterogeneous architecture**:* In order to show capabilities of HAMEX framework to model heterogeneous architectures, we selected the ODROID-XU3 board as our platform for experiments. Odroid-XU3 uses the Samsung Exynos5422 SoC which integrates four ARM Cortex-A15 cores, four ARM Cortex-A7 cores and a Mali-T628 GPU as shown in Figure 3. This platform is a contemporary representative MPSoC with a heterogeneous architecture as it integrates a GPU as well as CPUs with heterogeneous cores. The Mali GPU embedded in this platform can be used for graphics rendering as well as general purpose computation. ARM big.LITTLE architecture brings the advantage of both high performance and power efficiency for this platform. Although CPU and GPU units provide large computation power, the race over accessing data between these units can be a bottleneck for this system. This memory issue will be analyzed in Section IV-B. In this section we focus on modeling the computational units of the ODROID-XU3 platform.

There are two clusters of CPU cores in the ODROID-XU3 platform. In order to model these heterogeneous cores (ARM Cortex A15 are out-of-order and A7 are in-order cores), we used ARM detailed and MinorCPU models in Gem5. We obtained micro-architectural parameters of detailed model from [16] to model the Cortex-A15 cores. Endo et al. in [16] also simulate Cortex-A7 by using modified O3 model which provide a cycle-approximate in-order pipeline. Instead, we used the recently introduced MinorCPU model to simulate in-order cores which simulates in-order pipeline cores with more

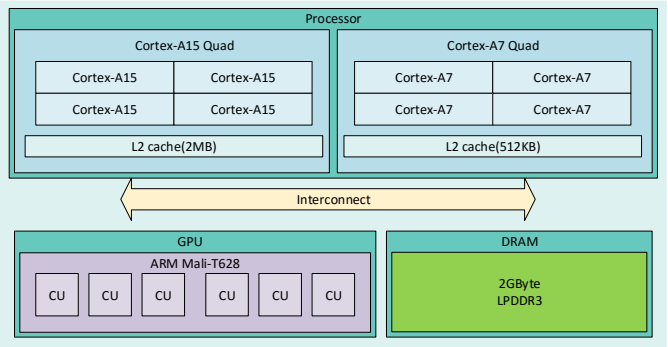


Fig. 3: ODROID-XU3 platform

precision. We model individual and shared cache hierarchy for CPU cores similar to the platform. We tested our modeled CPU in both System Emulation(SE) and Full System (FS) simulation mode. These two modes represent similar cases if target applications run on real platform in bare-metal or running with the operating system.

To demonstrate simulation strategies for general purpose GPUs, we model the ARM Mali-T628 used in the ODROID-XU3 platform. In real hardware, Mali GPU has 6 computational units which are separated into two clusters of two and four cores respectively due to cache coherency issue. We ran the GPU benchmarks on the 4-core cluster only hence we model only four general purpose GPU cores in our simulation environment. The modeled GPU can have the same frequencies as the real platform during simulation but it cannot change during runtime due to the fact that there is no DVFS governor in the GPU simulation environment. Furthermore, to fully capture the execution flow of the GPU, we add an emulated CPU unit to the simulation platform to start the command queue and transfer the input to the GPU unit.

*2) **Fidelity of HAMEX for DSE**:*

Recall that the primary use case for HAMEX is early design space exploration of alternative architectural configurations. Towards that end, the fidelity of exploration is of paramount importance. That is, the trend shown by our HAMEX simulations should follow those of a real platform, so that designers can use the simulations results to make high-level trade-offs during comparative evaluation of different configurations. In order to achieve this goal, in the previous section we modeled the CPU and GPU units embedded in the Odroid-XU3 platform in our simulation environments. Here, we evaluate the fidelity of the modeled units to deliver meaningful results for designers at early stages of design.

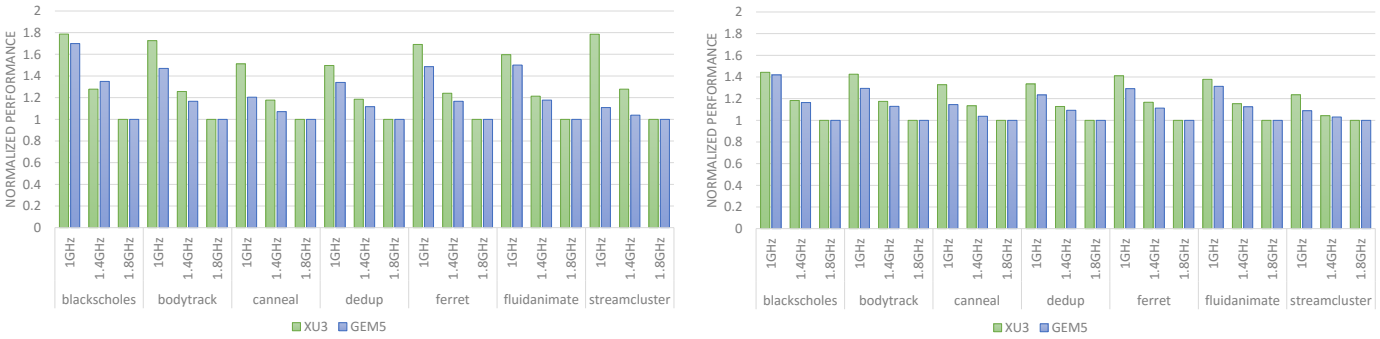
In order to evaluate the fidelity of our computational units modeled in HAMEX, we selected the PARSEC benchmark suite as our CPU workload and AMD SDK benchmark set for our GPU workload. Our main goal is to establish fidelity of the simulation framework to model the changes in behavior of design in response to changes in configurations. These configurations include the size of memory or cache, frequency, software managements, etc. We select frequency as our knob of configuration because it is configurable in both the real platform and our simulation framework. In order to test the fidelity of HAMEX for design space exploration, we run a series of experiments showing the trend of simulation time while changing frequency in compute units compared to changes in actual board's execution time in response to same changes.

Figure 4 shows the trend of simulation and execution time for three different frequencies in each CPU cluster. We use a single thread of execution to run on either Cortex-A15 or Cortex-A7 cores. Figure 4a shows the trend of simulation results of Cortex-A15 in comparison to total execution time on real hardware. Figure 4b shows similar trend to their corresponding hardware cores for modeled Cortex-A7 cores running with different frequencies. To better show the trend while changing frequencies we normalized values to the highest frequency execution time. We simulate our CPUs in full system mode with a Linux operating system running on top of them to capture all the behavior of the ODROID hardware which also runs a Linux OS. This would mimic all the effect of OS on CPU cores and memory units. From Figure 4, it should be clear that our HAMEX simulations follow the measured trends on the real board, thereby establishing fidelity of our processor simulations.

For the GPUs, we model the Mali GPU in the Multi2Sim environment using four computational units that can run in four different frequency steps(650, 480, 350 and 177 MHz) similar to the ones supported with the GPU on a real board. Figure 5 shows the behavior of the simulated GPU along side the Mali GPU. This figure shows the fidelity of simulation in regards to changes in configurations of design space as we see a similar trend to real hardware. We run benchmarks from AMD SDK on both the ODROID board and the Multi2Sim simulation environment in order to collect performance metrics and memory traffic. During our experiments we fix the frequency on the ODROID board as Multi2Sim doesn't support running a live OS or a real time DVFS governor. We also note that a majority of our benchmark simulations are within 20 percent of error in accuracy, with only a few benchmarks showing higher errors in accuracy due to the low number of instructions in their parallel kernels. Nevertheless, we note that for early DSE the fidelity of our HAMEX framework is most important, and has been established as illustrated in Figures 4 and 5 where our simulations closely track the actual measured trends on a real hardware platform.

B. Memory exploration

Recall that many new memory families (e.g., HMC, NVM, etc.) become available and provide opportunities for addressing the challenges in memory organizations. Thus there is a need for a comprehensive framework to evaluate these new memory organizations. Our HAMEX framework provide a flexible and comprehensive memory exploration mechanism to evaluate these alternative memory families. Designers can observe the effect of their target applications on a specific memory module or compare the performance and energy



(a) ARM Cortex A15

(b) ARM Cortex A7

Fig. 4: Fidelity Evaluation for CPU Models

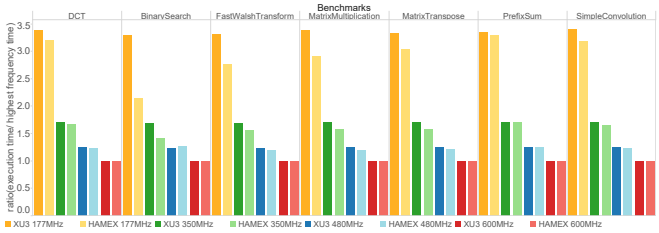


Fig. 5: Fidelity Evaluation for Mali GPU Model

efficiency of different memory modules in response to their target application. We performed extensive memory exploration analysis using HAMEX. In the interest of brevity, here we report the key experiments that represent the type of explorations that can be done in this framework.

1) ***Setup****:* we selected the PARSEC benchmark suite as the CPU workload and AMD SDK benchmarks for GPU evaluation. In our simulations there were only a few benchmarks (e.g., x264 and MatrixMultiplication) that needed modifications to enable execution. We excluded them from this report in order to have an untouched set of benchmarks in our experiments. To better represent design space exploration in our experiments we classify GPU benchmarks into different categories (large, medium, small) based on their throughput. This enables a clear representation based on the traffic imposed from the GPU compute units.

HAMEX's integration of Gem5 classic memory and DRAMSpec provides the ability to evaluate a wide variety of already implemented memory modules such as LPDDR3, WideIO, DDR3, GDDR5, etc. This set can be extended easily and the ability to connect the interconnect interfaced in HAMEX to SystemC components provides an opportunity to test external memory systems. In our experiments we show comparison between DDR3, LPDDR3 and HMC memories. We believe these three memory modules represent both a clear power-performance trade-off and as well the ability to evaluate newer memory families.

2) ***Latency and Energy results****:* In the first experiment we evaluate the effects of different classes of workloads on a specific memory module (DDR3). Figure 6 shows the overhead of GPU workload's memory traffic imposed on the memory system while simultaneously running CPU benchmarks. GPU workloads with higher throughput (larger memory access traffic over time) causes queuing delays and in some cases read and write retries while accessing the memory. This issue is investigated in details in Section IV-B3. Comparing to benchmarks with small amount of traffic this queuing delay and contention cause 3% and 13% higher memory access latency respectively for benchmarks with medium and large throughput.

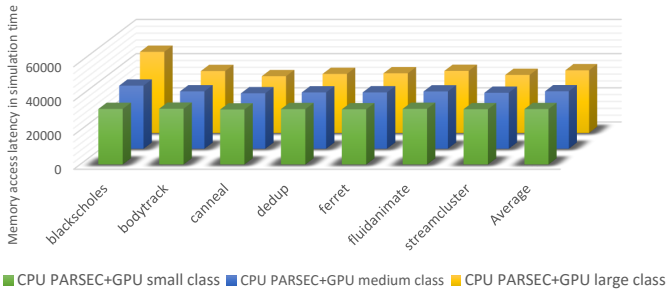


Fig. 6: DDR3 average memory access latency

In the second set of experiments, we compare three memory modules (DDR3, LPDDR3 and HMC) in terms of memory access latency and energy. The results are shown in Figures 7 and 8. In these experiments we simulate each GPU workload with a CPU workload and report the average to compare the three memories. Many different parameters can be evaluated for each memory module but here we focus on memory access latency and energy consumption. Designers can evaluate the average memory latency from the available design options in response to their target workloads. Figure 7 shows a 45% lower average memory latency for hybrid memory cube(HMC) over LPDDR3. Also normal DDR3 module is 21% faster than LPDDR3. This high latency in LPDDR3 might be due to its power efficient design compared to other low latency or high bandwidth memory units. Average latency represented in this figure is measured in simulation ticks and can be converted to

real time based on the frequency of memory module.

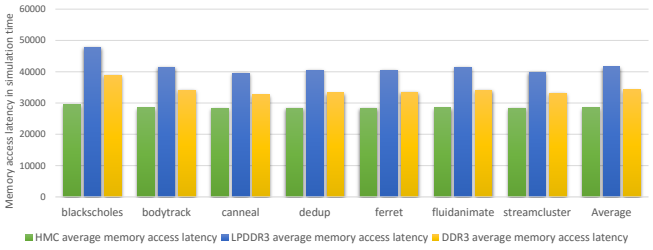


Fig. 7: Average memory access latency

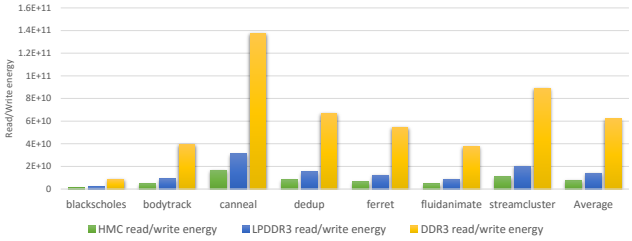


Fig. 8: Total read/write energy (pJ) in various memory modules

Figure 8 shows a comparison between the same three memory modules in terms of energy consumption due to read and write accesses. Although power estimation tools used in HAMEX framework provide energy metrics for all phases of memory accesses, we report the energy consumed by read and write accesses because they show the effects of heterogeneous workloads more clearly. In Figure 8 we can observe HMC 3D stacked memory can provide excellent energy efficiency. Meanwhile LPDDR3 delivers lower energy consumption comparing to DDR3 memory. The reason behind high energy consumption of the canneal and streamcluster benchmarks is the larger memory traffic generated by CPU causing more read/writes in memory.

3) ***Queuing and contention delay****:* In order to further investigate effects of heterogeneous workloads, we evaluate the average queuing delay and number of total retries caused by high number of memory accesses. Figure 9 shows trend of increase in number of memory access retries while increasing the GPU memory traffic. We can see in some benchmarks like Streamcluster and Canneal there is not much difference between retries for large or small GPU throughput. This is because these benchmarks have much higher CPU traffic than even large GPU set. We see almost four times difference in retries for blackscholes benchmark and it's due to the fact that throughput of this benchmark is close to medium set of GPU traffic, thus causing a larger contention when facing large memory traffic from GPU. Although we see simillar trend in all three memory families, low latency DDR3 can response to individual memory requests faster which reduces number of retries in memory.

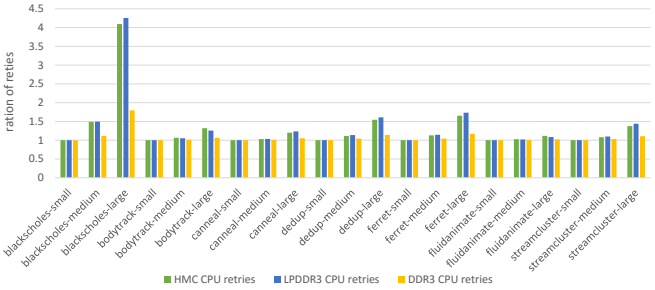


Fig. 9: Number of retries (normalized to small GPU set)

Figure 10 shows average queuing delay in DDR3, LPDDR3 and HMC memory families in regards to heterogeneous workloads. We can observe when the traffic simultaneously increases from different compute units (CPU and GPU) memory queues start to fill which causes higher average queueing delay. Figures 9 and 10 show some of the main reasons we have seen higher memory latency while dealing with larger GPU traffic in Section IV-B2.

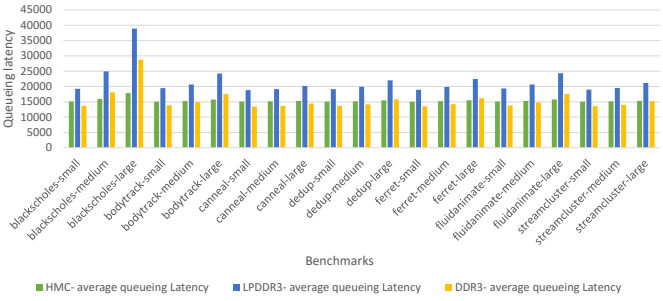


Fig. 10: Average queuing delay

All these results provide a trade-off between high cost HMC, energy efficient LPDDR3 and low latency DDR3. Designers can use this as a model of design space exploration to choose the suitable memory modules that best fit their platform and target application space.

V. Conclusion

When designing a heterogeneous server or embedded systems, considering memory effects is essential in determining the system's power and performance. Knowledge about the volume and phases of memory traffic generated due to executing comprehensive benchmarks on various compute units can give a clear insight to system designers about the bottlenecks in their system. In addition, being able to compare different memory technologies can guide designers to select the right memory module for their system. In this work, we presented HAMEX, a comprehensive framework that enables design space exploration for heterogeneous architectures using the novel memory systems. The integration of Gem5 cycle-accurate full system simulator with Multi2sim as GPU simulator, allows system designers to easily explore and

study challenges in a heterogeneous system for their target applications. The experiments done in this work shows the ease and flexibility of prototyping a heterogeneous platform at early stages of design and aims to show merits of different memory systems in response to captured traffic generated by CPU and GPU simulators.

REFERENCES

[1] A. Nowatzyk, Fong Pong and A. Saulsbury, "Missing the Memory Wall: The Case for Processor/Memory Integration," Computer Architecture, 1996 23rd Annual International Symposium on, 1996, pp. 90-90.

[2] 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 (August 2011), 1-7.

[3] J. Power, J. Hestness, M. S. Orr, M. D. Hill and D. A. Wood, "gem5-gpu: A Heterogeneous CPU-GPU Simulator," in IEEE Computer Architecture Letters, vol. 14, no. 1, pp. 34-36, Jan.-June 1 2015.

[4] Amir Kavyan Kavyan Ziabari, Jose L. AbellÃąn, Rafael Ubal, Chao Chen, Ajay Joshi, and David Kaeli. 2015. Leveraging Silicon-Photonic NoC for Designing Scalable GPUs. In Proceedings of the 29th ACM on International Conference on Supercomputing (ICS '15). ACM, New York, NY, USA

[5] Rafael Ubal, Byunghyun Jang, Perhaad Mistry, Dana Schaa, and David Kaeli. 2012. Multi2Sim: a simulation framework for CPU-GPU computing. In Proceedings of the 21st international conference on Parallel architectures and compilation techniques (PACT '12). ACM, New York, NY, USA,

[6] Paul Rosenfeld, Elliott Cooper-Balis, and Bruce Jacob. 2011. DRAMSim2: A Cycle Accurate Memory System Simulator. IEEE Comput. Archit. Lett. 10, 1 (January 2011), 16-19.

[7] Awasthi, M., Balasubramonian, R., Chatterjee, N., Chishti, Z., Pugsley, S.H., Shevgoor, M., Shafiee, A., Sudan, K., Udipi, A.N.. (2012). Usimm: the Utah Simulated Memory Module a Simulation Infrastructure for the Jwac Memory Scheduling Championship.

[8] M. T. Yourst, "PTLsim: A Cycle Accurate Full System x86-64 Microarchitectural Simulator," 2007 IEEE International Symposium on Performance Analysis of Systems Software, San Jose, CA, 2007, pp. 23-34.

[9] L. M. N. Coutinho, J. L. D. Mendes and C. A. P. S. Martins, "MSCSim -Multilevel and Split Cache Simulator," Proceedings. Frontiers in Education. 36th Annual Conference, San Diego, CA, 2006, pp. 7-12.

[10] Fabrice Bellard. 2005. QEMU, a fast and portable dynamic translator. In Proceedings of the annual conference on USENIX Annual Technical Conference (ATEC '05). USENIX Association, Berkeley, CA, USA, 41-41.

[11] Vitaly Zakharenko, Tor Aamodt, and Andreas Moshovos. 2013. Characterizing the performance benefits of fused CPU/GPU systems using FusionSim. In Proceedings of the Conference on Design, Automation and Test in Europe (DATE '13). EDA Consortium, San Jose, CA, USA, 685-688

[12] Aamodt, Tor M; Fung, Wilson WL; Singh, I; El-Shafiey, A; Kwa, J; Hetherington, T; Gubran, A, Boktor, A Rogers, T Bakhoda, A ",GPGPUSim 3. x manual,2012-08-08)[2013-08-08].

[13] K. Chen et. al, CACTI-3DD: Architecture-level Modeling for 3D Diestacked DRAM Main Memory, in DATE, 2012

[14] O. Naji, C. Weis, M. Jung, N. Wehn and A. Hansson, "A high-level DRAM timing, power and area exploration tool," Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS), 2015 International Conference on, Samos, 2015, pp. 149-156.

[15] DRAMPower: Open-source DRAM Power and Energy Estimation Tool Karthik Chandrasekar, Christian Weis, Yonghui Li, Sven Goossens, Matthias Jung, Omar Naji, Benny Akesson, Norbert Wehn, and Kees Goossens, www.drampower.info, 2012

[16] Fernando A. Endo, Damien CouroussÃl', Henri-Pierre Charles, "Micro-architectural simulation of embedded core heterogeneity with gem5 and McPAT" RAPIDO '15 Proceedings of the 2015 Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools. January 2015.

[17] A. Patel, F. Afram, S. Chen and K. Ghose, "MARSS: A full system simulator for multicore x86 CPUs," Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE, New York, NY, 2011, pp. 1050-1055.

[18] Google, "Protocol buffers", https://developers.google.com/protocol-buffers/

[19] Frank Ghenassia, "Transaction-Level Modeling with Systemc: Tlm Concepts and Applications for Embedded Systems" Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.