

# A Comparative Study of Heterogeneous Processor Simulators

Muhammad aleem


*International Journal of Computer Applications*

## Cite this paper

Downloaded from [Academia.edu](#) 

[Get the citation in MLA, APA, or Chicago styles](#)

## Related papers

[Download a PDF Pack](#) of the best related papers 



[Multi2Sim: a simulation framework for CPU-GPU computing](#)

Perhaad Mistry

[GPU Acceleration for simulating massively parallel many-core platforms](#)

David Atienza

[Full system simulation of many-core heterogeneous SoCs using GPU and QEMU semihosting](#)

David Atienza, Luca Benini

# A Comparative Study of Heterogeneous Processor Simulators

Shagufta

Department of Computer Science,  
Capital University of  
Science and Technology (CUST),  
Islamabad, Pakistan

Muhammad Arshad Islam

Department of Computer Science,  
Capital University of  
Science and Technology (CUST),  
Islamabad, Pakistan

Muhammad Aleem

Department of Computer Science,  
Capital University of  
Science and Technology (CUST),  
Islamabad, Pakistan

Muhammad Azhar Iqbal

Department of Computer Science,  
Capital University of  
Science and Technology (CUST),  
Islamabad, Pakistan

## ABSTRACT

In 1970's, Gordon Moore perceived that the number of transistors in a processor would double after every 18 months. With the addition of more transistors on a single-chip, a processor's energy consumption increases exponentially. The solution to this problem is heterogeneous processors and machines. Heterogeneous machine is the combination of CPU and GPU platforms. Computer architecture is shifting from multi-core to heterogeneous era. Generally, computer architects practice of software simulation to model and analyze their ideas. Today, computer architects are using cycle-level simulators to discover and analyze new processor designs. To search the heterogeneous system design-space, we review and practically analyze heterogeneous simulators and their performance. In this study, we present a detailed comparative analysis of gem5-gpu, gem5, and multi2sim simulators.

## Keywords

Heterogeneous simulators, gem5-gpu, gem5, multi2sim.

## 1. INTRODUCTION

About fifty years ago, Gordon Moor has predicted that the number of transistors increasing after every eighteen months. After ten years this prediction was named as Moore's law. The additions of more transistors on single chip make processor faster and complex. The number of transistors and cores are increasing on chip through technology scaling, however the energy consumption also increases exponentially. The solution to this problem is heterogeneous integration to use multiple processors to gain more energy efficiency [1]. Today, computer architecture is shifting from multi-core to heterogeneous era [6]. Generally, computer architects use software simulators to model and analyze the processor design. The computing community is growing day by day and with passage of time variety of new processor designs are also evolving [4]. Due to the technology advancements, now it is possible to combine both CPUs and GPUs on single chip. This processor design will help to decrease latency between CPU—GPU and will enhance the application performances. With emerging heterogeneous processor design, the power shared CPU and GPU must use and the power budget must affect the performance. Therefore, technology trend must minimize the power consumption, delay of CMOS devices, allowing the hardware designers to add more devices on a chip. The

availability of more processing-units on a chip provides a great level of parallelism with low latency overhead.

Computer architects generally use CPU cycle-level simulators to analyze new processors design. To search the heterogeneous system design-space, we review the gem5-gpu [6], gem5 [2], and multi2sim [3] simulators. We discuss some important terms here before going in detail of these simulators.

## 2. BACKGROUND

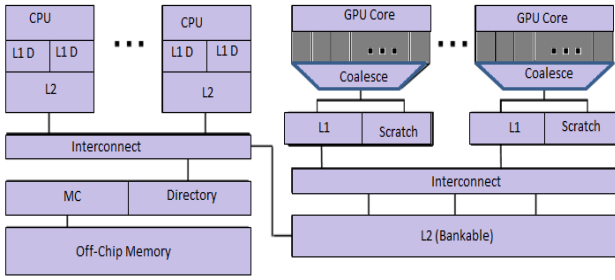
### MULTI-CORE CPUS

Single-core processors were based on arithmetic logic units, perform simple operations on data that was stored in registers. According to Moor law, a large number of elements were added to the processing unit and adding more procedures to spread the instruction set architecture to perform good performance. Multi-core processors generate an array of processors that as works single unit. Today, multi-cores are being employed in clusters, workstations, and in mobile devices [2].

## 3. GRAPHICS PROCESSORS (GPUS)

Today, the GPUs are being widely used as programmable devices. Graphic accelerators are the primary ancestors of GPUs. In mid-90s, there was no real presence of PC graphics. In 2000s, graphic controller came into being which deserve a new term: GPU *Graphic Processing Unit* [1] [17]. GPUs work as accelerators for providing computer graphics. Two most used applications frameworks for programming GPUs are Kronos's *Open Computing Language* (OpenCL) and NVIDIA'S *Compute Unified Device Architecture* (CUDA). Both application-programming frameworks are based on *Single Instruction Multiple Data* (SIMD) programming model employing large number of threads. The execution of a program on GPU for providing graphics is called *shader*. GPUs allow three level of concurrencies: 1) first level employs thousands of threads on GPUs, called thread-level concurrency where a single executing thread is called *work-item*; 2) second type is based on multiple kernels on the device, called kernel-level concurrency; and 3) third type let multiple applications on GPUs. GPU architecture is suitable for matrix-based mathematical operations and SIMD style of computing [2]. On a GPU, the OpenCL driver interacts with hardware via system-calls, while on CPUs OpenCL driver executes x86 instructions and is well matched with underlying

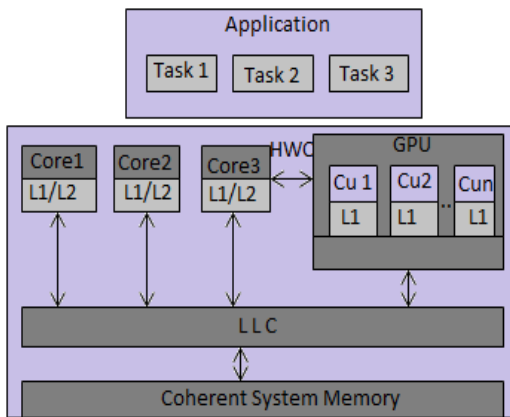
hardware [18]. Fig 1 depicts the heterogeneous CPU-GPU systems on a single chip. Cores are on the top private L1 and L2 cache memories.



**Fig1. Heterogeneous CPU-GPU architecture.**

#### 4. HETEROGENEOUS SYSTEMS

The combined CPU and GPGPU platforms are called *Heterogeneous Systems*. GPUs have great potential speedups and consume a little amount of energy. Heterogeneous system architecture reduces the latencies between the connected devices. Multiple programming models achieve the benefits from the GPGPU organization. These two platforms have their own memory hierarchy. Both the CPU and GPU can access the memory of each other i.e., GPUs accessing the CPU addresses for reading and writing and CPUs sharing page tables. GPUs have higher memory bandwidth whereas CPUs have higher capacity caches [19]. The heterogeneous systems can be programmed using NVIDIA's CUDA and OpenCL programming frameworks. CPU executes the serial part (*host program*) and the GPU execute the parallel part (*compute kernel*) of the application. The heterogeneous system includes a discrete CPU and GPU elements and requires explicit chip-to-chip data transfer. The overhead associated with the CPU-GPU communication reduces the performance and power efficiency of heterogeneous system. To overcome this, new architectures e.g., Intel's Sandy Bridge and AMD's Llano processors integrate both compute devices in the same chip.



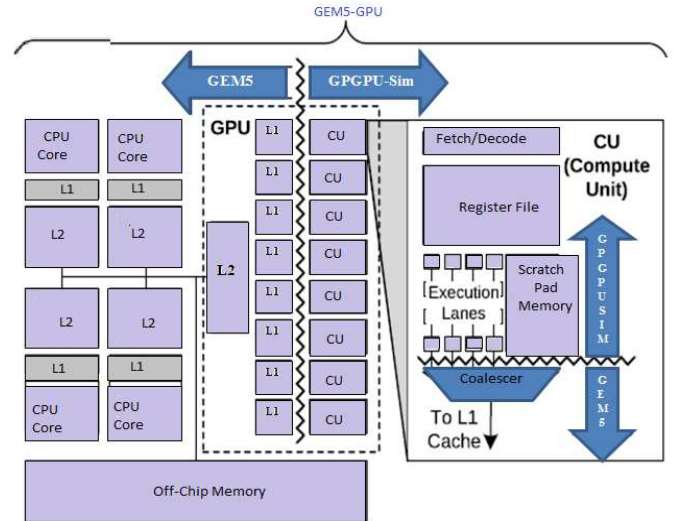
**Fig2. Heterogeneous system architecture**

#### 5. METHODOLOGY

For the analysis and evaluation of heterogeneous simulators, we have thoroughly studied many related articles. The study was done to figure out the performance parameters and attributes, which can be used in the comparative analysis. The three heterogeneous architecture simulators, *GEM5-GPU6*, *GEM54*, and *Multi2Sim5* have several different aspects, which are being analyzed in this study [17].

#### 5.1 GEM5-GPU

It is a modern simulator that incorporates the CPU-GPU systems. It posses BSD license (code available to all researchers without restrictions) and is accessible freely at [www.gem5-gpu.cs.wisc.edu](http://www.gem5-gpu.cs.wisc.edu). Gem5-gpu [20] executes unrestricted CUDA 3.2 source code. Build on top of Gem5 CPU simulator. Gem5-gpu simulator provides several benefits such as *detailed cache coherent model*, *full-system simulation*, and *check pointing*, etc. The Gem5-GPU simulator tightly integrates with the latest gem5 CPU simulator and provides increased extensibility of GPGPU programming model and system architecture [6]. *MOESI* is a directory based (*Modified Owned Exclusive Shared Invalid*) heterogeneous cache coherence protocol. Gem5-gpu employs the split version of the *MOESI-hsc* protocol [23] that simulates architectures with a CPU and GPU physical address spaces [6] [15] [16].



**Fig3. Gem5-gpu device architecture**

In Fig 3, architecture of GEM5-GPU compute device is presented. This architecture denotes four CPU cores and eight compute units for GPUs. Each compute unit contains a fetch and decode register file, execution lanes, and scratch-pad memory, and coalesce [2]. This architecture also shows on-chip and off-chip memories. The on-chip memory is joined with the processor on the same chip i.e., *instruction cache*, *data cache* and *on-chip SRAM*.

The data-cache and instruction act an interface between processor and off-chip memory. The on-chip SRAM also known is as **Scratch Pad memory**, existing on-chip separated from the off-chip memory but having same address and buses as on-chip. Scratch pad memory and cache are faster required less time to access their data. The key difference between data cache and scratch pad memory is that, scratch-pad memory assures a single cycle access time whereas data-cache compulsory, capacity and conflict misses. But off-chip memory (*DRAM*, *ROM*, or *PROM*) needs long time to access their data. **Coalescer** merges two adjacent blocks of memory i.e., when global memory is accessed, lane sends its address to the *coalescer*, it merges the free memory accesses to the similar block. GPU also have hierarchy of cache, which stores the data receive from global memory of device.

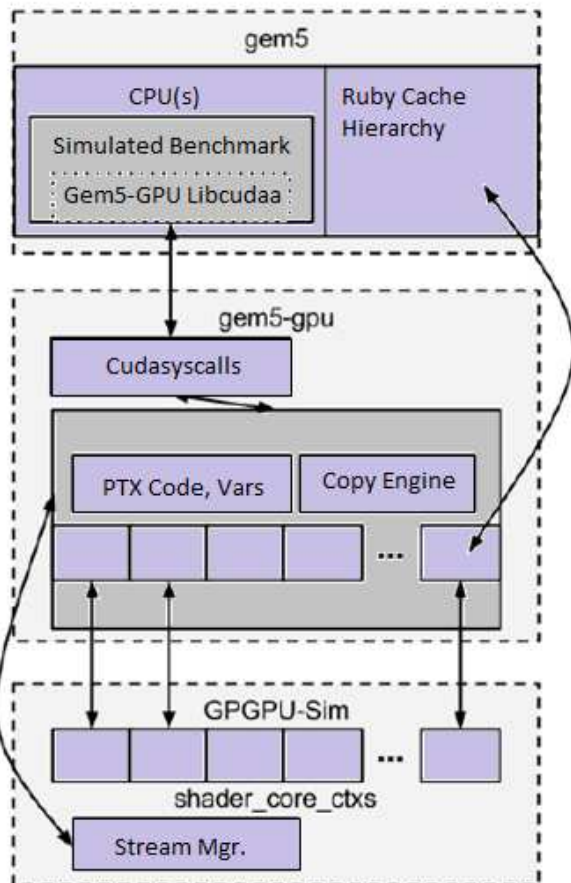
#### 5.2 Gem5 Simulator

**Gem5** simulator is the combination of *M5* and *GEMS* simulators. It merges the best features from both simulators. Computer architects generally use software simulation to

model and analyze their ideas. Mainly, the **Gem5** is used for architecture modeling and provides three features: *flexibility*, *widespread accessibility*, and *developer collaborators*. The simulator provides:

**CPU Model:** four types of CPU model based on *speed* and *accuracy* [4].

- 1) **Atomic Simple model:** IPC CPU, one cycle per instruction;
- 2) **Timing Simple model:** non-pipeline model uses the reference for modeling and memory access latencies;
- 3) **In-Order is a pipelined model** [10]: is a seven stages pipelined model 1) *fetch*, 2) *decode*, 3) *rename*, 4) *issue*, 5) *execute*, 6) *write back*, 7) *commit* created on physical register file architecture.
- 4) **O3 models** inter instruction dependencies for in-order/ superscalar, branch prediction, and load/store. Both models are simultaneous multithreading CPUs. O3 provides a complete system model [10] [13] [14].



**Fig4. Software architecture of gem5-gpu**

To model energy consumption, the simulator depends on *Multicore Power, Area, and Timing* (McPAT) [12]. McPAT is a modeling framework for multi-threaded and multi-/many-core processors. It models *power*, *area*, and *timing* simultaneously. The power model depends on the activity of the elements such as *frequency* and *voltage* of different domains. The power model is specified only for those, which have a concept of clock and voltage related with simulated

object modeled. Thus, for every clock object (within the Gem5 simulator) power consumption model can be generated in the system.

Gem5 simulator leverages three types of interfaces: 1) *port*, 2) *buffer*, and 3) *EXTRAS interfaces*. *Port interface* is used for message sending and receiving. It connects all the memory objects such as CPUs to caches, caches to busses, busses to devices, and memories in the classic memory system. Port interface support three modes. *Timing mode* provides the detail about the memory accesses. *Atomic mode* provides data about timing. *Functional mode* updates the simulator without any change of timing information. Ruby memory model uses port interface to connect CPUs to devices. *Buffer interface* provides high-communicated interface among the components in the system. In buffer interface the message-typing and storage operation also employed. EXTRAS interface identify external code compiled in the gem5 binary. Complex components in the gem5 can be added and removed via this interface [11].

Gem5 simulators's system mode consists two types of modes: *System Call* and *Full system* emulations. In the *system call emulation mode* the operating system and devices modeling is not included, imitating system level services, e.g., *read ( )* etc. No dynamic thread scheduler is employed in system emulation mode whereas the threads are mapped statically to processor cores. The *full system emulation mode* implements both the kernel and user level instructions to produce a complete system model. This mode supports interrupt-exceptions, fault-tolerance, privilege instructions, I/O devices, and fault handler.

Gem5's memory system provides two memory models: *classic* and *ruby*. Both the two models implement cache coherence protocols. The main difference between classic and ruby is *Specification Language for Implementing Cache Coherence* (SLICC). SLICC allows Gem5 to models more variety of cache coherence protocols. Gem5 employs the directory and broadcast based cache coherence protocols. Moreover, it allows several different protocols with negligible programming effort. The *classic* memory model is a fast and easily configurable. The *ruby* memory model provides the substructure capability of accurate simulations of cache coherent memory system.

The ruby memory model supports two types of network models: *simple* and *Garnet* network models. The *simple* network model enables acquisition of linkage bandwidth and latency. However, the simple network model is not suited to model control flow and resource contention. The garnet network model provides the details of router of micro architecture, control flow, resource contention, and timing information. The garnet memory model is well suited for on-chip model studies.

Gem5 simulator supports different interconnection devices such as *Network Interface Cards* (NICs) and *Integrated Device Electronics* (IDEs). NICs translate parallel signals produced by the computer into the serial format sent over the network. IDE controller controls the flow of data among I/O devices such as CD/DVD, readers/writers external devices, DMA engine, frame-buffers, and AURTs etc. Multi-system communicating is based on TCP/IP.

Gem5 supports modeling of different ISAs, including [4]:

**ALPHA:** this ISA is mostly used on gem5 based on DEC Tsunami system. It can be boot with unchanged Linux kernel and supports up to 64 cores;

**ARM:** ISA models Cortex-A9 (ARMv7 ISA) and supports different instruction set extensions such as *Thumb*, *Thumb-2*, *NEON*, and *VFPv3*;

**MIPS:** a 32-bit MIPS processor based ISA is supported;

**X86:** this is a generic 64-bit CPU and can boot unchanged Linux kernels with symmetric multi-core processors support;

**POWERPC:** based on the power processor ISA of 32-bit processor;

**SPARK:** this architecture models ULTRASPARC T1 processors.

### 5.3 Multi2Sim Simulator

It is an open source and accurate-cycle simulator that supports ISAs-level simulations of the AMD Evergreen GPU and the x86 CPU devices. The simulator can be downloaded from [www.multi2sim.org](http://www.multi2sim.org). Multi2Sim provides a comprehensive simulation framework for CPU-GPU computing written in C language. Multi2Sim covers models for super-scalar, single threaded, multithreaded, and multi-core CPU-GPU architectures. It is a Linux-based command-line toolset, without source modification runs OpenCL applications [24]. It offers accurate-cycle for multiple architectures. The goal of this simulation is to increase research into the architecture designed and applications by providing detailed models of the fundamental architectures [5][18]. It exploits system call emulation and *timing-first* approach. The timing-first approach provides the detail of efficiency, robustness, and performance simulation on various levels. Timing-first approach does not simulate the whole operating system, however it has ability to run parallel workload with dynamic thread creation [7].

Multi2Sim simulator provides three simulation models: 1) *functional simulation*, 2) *detailed simulation*, and 3) *event-driven simulation*. Functional simulation engine is made as an independent library and runs as an interface for the other simulators and support parallel workloads execution. In this model, information about the hardware threads is not maintained. It has functions to generate and end the entities of software and provides check & control machine instructions. Functional supports is multi-cores by replicating multi-core pipeline. Its highest conflict-point is the interconnection network. Performance of multiple sequential workloads does not include interconnection overhead. This is because each software entity has own memory, in this way the physical address spaces are not connected with each other. Therefore, when there is no connection exists between contexts of cache blocks coherence activity is not employed. This makes the sense to run the parallel workloads to assess multi-core processors and maintain high interconnection network activity produced by coherence protocol transactions [9]. *Detailed simulation* support multithreading, contained pipelining, branch predictors, and supports memory latencies of varying sizes. The pipeline stages can be allocated to several executing threads. In multithreading pipelining the sharing strategy is different because of the execute stage and functional units can operate and write results back to register-file. Multi2Sim supports *fine-grained*, *coarse-grained*, simultaneous multithreaded parallelism. By giving fetch priority to threads, global throughput is augmented when running long latency operations. It uses functional engine to achieve timing-first simulations [7][9]. The execution driven simulation is performed by the activity of updating the existing contexts state in each cycle. *Event-driven simulation* permits cycle-by-cycle simulation [9]. Multi2Sim provides memory hierarchy configuration in the INI file with the option `-mem-`

`config<file>`, interconnection networks (the locations and the no. of interconnect depends on sharing strategy of the instruction and data caches) and coherence protocol configuration (which provides consistency amongst cache blocks). In multi2Sim, threads can employ in-order or out-of-order pipelining mechanism. There is no power consumption model supported in multi2Sim [7][9]. Multi2Sim does not require guest operating systems and can directly employ threads scheduling. The operating system is removed from the guest S/W stack in application simulation mode only. The application and operating system can communicate with each other at runtime through system calls. In Multi2Sim, both the *mesh* and *ring* topologies are employed. In mesh topology, the user is responsible to enter routing table entries related to routes. Ring topology holds network links cycle [8][18]. MOESI cache coherence is a directory-based protocol employed by Multi2Sim. Multi2Sim enables L1 and L2 cache as either shared or private among cores. Multi2Sim is particularly best for exploring multiple device executions by utilizing a new *NMOESI* cache coherence protocol [8][15][18]. Table 1 summarizes the common attributed of heterogeneous simulators discussed in this study:

## 6. DETAIL OF TABLE 1

### EMULATION MODES:

(a) **Fully System Emulation mode:** executes both the kernel-level and user-level instructions and models a complete system including OS, devices, exceptions, interrupts, I/O devices etc. [4].

(b) **System Call Emulation:** In this mode the operating system and device modeling is not included imitating system level services. No thread scheduler is employed in system calls emulation [7].

(c) **The timing first:** gives the detail of efficiency, robustness, and performing simulation on various levels. It does not simulate the whole operating system and has the ability to run parallel workload with dynamic thread creation [1].

**BSD BASED LICENSE:** (code available to all researchers without restrictions [5].

### MEMORY MODELS:

(a) **Memory model “Ruby”:** provides the capability of accurate simulations of cache coherent memory system [2].

(b) **Memory model “Classic”:** Fast and easily configurable memory system model [1].

(c) **Memory model “SLCC”:** found in ruby memory system. SLICC allows Gem5 to models more variety of cache coherence protocols. In Gem5, the cache coherence protocols are directory based broadcast protocols. Moreover, lets different protocols with negligible programming attempt [4].

### INSTRUCTION SET ARCHITECTURE:

(a) **ALPHA:** This ISA is mostly used on gem5 based on DEC Tsunami system. It can be booting unchanged Linux kernel and supports up to 64 cores.

(b) **ARM:** This ISA models Cortex-A9 (ARMv7 ISA) supports different instruction set extensions such as *Thumb*, *Thumb-2*, *NEON*, and *VFPv3*.

(c) **MIPS:** A 32-bit processor called MIPS.

(d) **X86:** This is a generic 64-bit CPU can boot unmodified Linux kernels in symmetric multi-core processors

(e) **POWERPC:** Based on the power ISA, also a 32-bit processor.

(f) **SPARK:** This architecture models ULTRASPARC T1 processor [2].

#### SIMULATION MODELS:

(a) **Functional Simulation:** Functional simulation engine is made as an independent library that runs an interface for the other simulators and support parallel workloads execution. No information about the hardware threads, generate and end the

entities of software and check and control machine instructions [9].

(b) **Detailed Simulation:** Detailed simulator evaluates the nature of recently implemented machine instructions and sums the latencies operations suffered by hardware structure.

(c) **Execution driven simulation** performed the activity of updating the existing contexts state in each cycle. That is sequence of call to the kernel [7].

**Table1. Common attributes of a heterogeneous simulator.**

Architectural Attributes		Gem5-gpu	Gem5	Muti2Sim
Execution Modes	Full System	✓	✓	×
	System Call Emulation	✓	✓	✓
	Timing Simulation	✓	✓	✓
License (BSD)		✓	✓	✓
Memory System Models	Ruby	✓	✓	✓
	Classic	×	✓	×
	SLICC	✓	✓	×
Applications (Runtime)	Runs un-modified CUDA source code	✓	×	×
	OpenCL	×	×	✓
	Python and C++ Classes	×	✓	×
Architectural Modelling		✓	✓	✓
Load/Store pipeline		×	✓	✓
Cycle-Accurate Simulators		✓	✓	✓
Heterogeneous System Simulators		✓	✓	✓
Supported ISAs Or Framework		X86, SPARC, Alpha, ARM CPU	X86, ARM CPU, MIPS, Power SPARC	X86, ARM AMD, GCN, GPUs
Shared virtual address space		✓	✓	×
Multi-core Supported	Multiple cores	✓	✓	✓
	Cache Hierarchy Configuration	✓	✓	✓
	Interconnection Network	✓	✓	✓
	Coherence Protocol	✓	✓	✓
Single Thread	In-Order Pipeline	×	×	✓
	Out-Of-Order Pipeline	×	✓	✓
	Power Consumption Modelling	×	✓	×
Multithread	Multithread	✓	✓	✓
	FGMT, CGMT, SMT	×	×	✓
Guest OS (s)		Android12, Linux	Android, FreeBSD, Linux, Solaris	×
Simulation Models	Functional simulation engine	×	✓	✓
	Detailed simulator	×	✓	✓
	Execution-driven simulation	×	✓	✓
CPU Models	Atomic-Simple	×	✓	×
	Timing-Simple	×	✓	×
	Out-Of-Order CPU	×	✓	×
	In-Order CPU	×	✓	×
Topologies		Mesh, Torus, Crossbar, Default cluster	Mesh, Torus, Crossbar	Ring Mesh
Non-Blocking Mechanism		✓	×	×
DMA engine		✓	✓	×
Multiple CPU		✓	✓	✓
Standard Interfaces	Port Interface	×	✓	×
	Message Buffer Interface	×	✓	✓
	EXTRAS Interface	×	✓	×
Simulation	Execution-driven simulation	✓	✓	✓
	Trace-driven simulation	✓	×	✓



#### STANDARD INTERFACE SUPPORT:

(a) **Port interface** is used for message sending and receiving. It connects all the memory objects such as CPUs to caches, caches to busses, busses to devices and memories in the classic memory system.

(b) **Message buffer interface** give high-communicated interface among components in the system. In buffer interface the message typing and storage operation also performed.

(c) **EXTRAS interface** identify external code compiled in the gem5 binary. Complex components in the gem5 can be add and remove via this interface [11].

**Cache Hierarchy:** There are two types of cache hierarchy CPU's and GPU's cache hierarchies. Each CPU core has its own private L1 and L2 cache. This cache hierarchy retains the evaluation usage of caches among cores. The GPU cache hierarchy is different from CPU hierarchy. The GPU hierarchy contains scratch-pad memory that is explicitly accessible by the programmer. GPU core has also L1 and L2 caches. L1 cache allows incoherent data and L2 cache participates in coherent protocol with CPU cores [21].

#### POWER CONSUMPTION MODELS:

For power models we depend on **McPAT**. . Power model depend on the activity of the elements also on frequency and voltage of different domains. The power model is specified only for those, which have a concept of clock and voltage related with simulated object modeled.

(a) **Atomic Simple model** (IPC CPU, one cycle per instruction)

(b) **Timing Simple model** (non-pipeline model uses the reference for modeling memory access latencies)

(c) **In-Order** is a pipelined model.

(d) **O3** is a seven stages pipelined model: that is 1: fetch, 2: decode, 3: rename, 4: issue, 5: execute, 6: write back, 7: commit created on physical register file architecture. O3 models inter instruction dependencies for in-order/superscalar, branch prediction, and load/store. Both models are simultaneous multithreading CPUs. O3 provides a complete system model [4][10][13][14].

**CACHE COHERENCE PROTOCOL** to maintain cache coherence via messaging and share locks between worker node and virtual machine. The messages use a message protocol, this protocol includes three things: hash-able object key, message type, and region identifier [22].

**EVENT-DRIVEN** simulation permits cycle-by-cycle simulation [9].

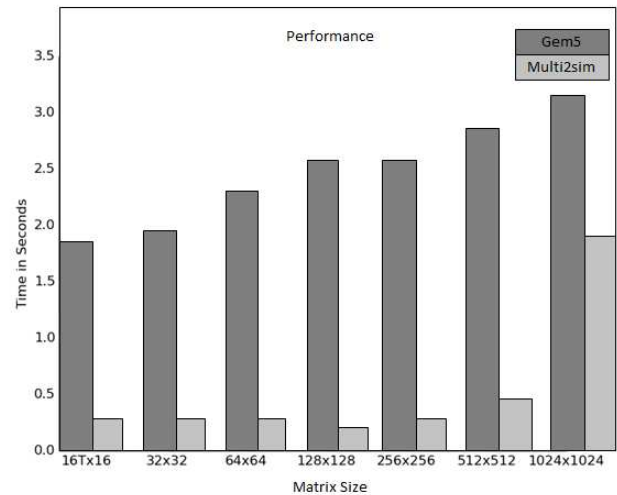
## 7. EXPERIMENTAL RESULTS

With analysis of the experimental work, it can be observed that all the three simulators provide architecture modeling and accurate cycle-level simulations. We performed experiments using matrix multiplication compute-intensive application with varying input sizes. Experiments show that the multi2sim provide faster performance compared to the gem5 simulator. Table 2 shows the comparative results of execution times of the experimented application. The table 2 shows that the multi2sim achieves 0.59 times to 9.4 times more speedups compared to the gem5 simulator. Figure 5 also shows the comparative execution time results of the experiment.

**Table 2. Matrix multiplication application performance.**

Matrix Size	Execution Time (in seconds)						
	16x16	32x32	64x64	128x128	256x256	512x512	1024x1024
Multi2sim	0.20	0.24	0.26	0.25	0.25	0.37	2.2
Gem5	1.90	1.56	1.77	2.5	2.6	2.9	3.5

The performance results show that the multi2sim simulator achieves better performance for the compute intensive applications compared to the gem5.



**Fig 5. Matrix multiplication performance on the multi2sim and gem5 simulators**

## 8. CONCLUSIONS AND FUTURE WORK

This paper presented a summary of three simulators; gem5, gem5-gpu, and multi2sim. All the three simulators are provide support for heterogeneous compute architecture. Gem5 is the combination of M5 and GEMS highly capable simulators. It is available with all the required documentation resources. Gem5-gpu provides the accessibility with both current and future heterogeneous architectures and the data. Multi2Sim adds more diversified attributes to the current simulation environment (SimpleScaler, simies GEMS). In this paper, we thoroughly investigated the attributes and features available in these heterogeneous simulators. Moreover, we performed experiments using a compute-intensive application (matrix multiplication) employing both the gem5 and multi2sim simulators. The experimental results show that the multi2sim performs better compared to the gem5 and achieves up to 9.4 times better performance results.

## 9. REFERENCES

- [1] J. Power, J. Hestness, M. S. Orr, M. D. Hill, D. A. Wood, "gem5-gpu: A heterogeneous cpu-gpu simulator," Computer Architecture Letters, vol. 14, no. 1, pp. 34--36, 2015.
- [2] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, D. A. Wood, "The gem5 simulator," ACM SIGARCH Computer Architecture News, vol. 39, no. 2, pp. 1--7, 2011.

- [3] R. Ubal, B. Jang, P. Mistry, D. Schaa, D. Kaeli, "Multi2Sim: a simulation framework for CPU-GPU computing," in ACM, New York, 2012.
- [4] J. V. Quiroga Esparza, "Heterogeneous CPU/GPU Memory Hierarchy Analysis and Optimization," Universitat Politècnica de Catalunya, 2015.
- [5] Y. Ukidave, "Architectural and Runtime Enhancements for Dynamically Controlled Multi-Level Concurrency on GPUs," Northeastern University Boston, 2015.
- [6] P. R. Panda, N. D. Dutt, A. Nicolau, "On-chip vs. off-chip memory: the data partitioning problem in embedded processor-based systems," ACM Transactions on Design Automation of Electronic Systems (TODAES), vol. 5, no. 3, pp. 682--704, 2000.
- [7] R. Ubal, J. Sahuquillo, S. Petit and P. Lopez, "Multi2sim: A simulation framework to evaluate multicore-multithread processors," in Citeseer, Rio Grande do Sul, 2007.
- [8] R. Ubal, B. Jang, P. Mistry, D. Sachaa, D. Kaeli, "The Multi2Sim Simulation Framework," International Conference on Parallel Architectures and Compilation Techniques, pp. 335--344, 19-23 September 2012.
- [9] V. Spiliopoulos, A. Bagdia, A. Hansson, P. Aldworth and S. Kaxiras, "Introducing DVFS-management in a full-system simulator," Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2013 IEEE 21st International Symposium on, pp. 535--545, 14-16 August 2013.
- [10] A. Gutierrez, J. Pusdesris, R. Dreslinski, T. Mudge, C. Sudanthi, C. Emmons, and N. Paver, "Sources of error in full-system simulation," Performance Analysis of Systems and Software (ISPASS), 2014 IEEE International Symposium on, pp. 13--22, 23-25 March 2014.
- [11] F. A. Endo, D. Couroussé and H. P. Charles, "Micro-architectural simulation of in-order and out-of-order ARM microprocessors with gem5," Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIV), 2014 International Conference on, pp. 266--273, 14-17 July 2014.
- [12] F. A. Endo, D. Couroussé, H.-P. Charles, "Micro-architectural simulation of embedded core heterogeneity with gem5 and McPAT." Proceedings of the 2015 Workshop on Rapid Simulation and Performance Evaluation, Proceedings of the 2015 Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools, p. 17, 22 January 2015.
- [13] J. Yin, O. Kayiran, M. Poremba, N. E. Jerger, "Efficient synthetic traffic models for large, complex SoCs," 2016 IEEE International Symposium on High Performance Computer Architecture (HPCA), pp. 297--308, 12-16 March 2016.
- [14] J.-J. Cheng, S.-H. Hung, C.-W. Yeh, "Rapid analysis of interprocessor communications on heterogeneous system architectures via parallel cache emulation," Proceedings of the 2015 Conference on research in adaptive and convergent systems, pp. 418--423, 9-12 October 2015.
- [15] H. Wang, V. Sathish, R. Singh, M. J. Schulte, N. S. Kim, "Workload and power budget partitioning for single-chip heterogeneous processors," Proceedings of the 21st international conference on Parallel architectures and compilation techniques, pp. 401--410, 19-23 September 2012.
- [16] S. Gurfinkel, "The Distribution of OpenCL Kernel Execution Across Multiple Devices," University of Toronto, Toronto, 2014.
- [17] A. Seyhanli, "Memory Controller Design for GPU Simulation in Multi2sim," Universitat Politècnica de Catalunya, Barcelona, 2015.
- [18] J. V. Quiroga Esparza, "Heterogeneous CPU/ (GP) GPU Memory Hierarchy Analysis and Optimization," Universitat Politècnica de Catalunya, BarcelonaTech, 2015.
- [19] "Multi2Sim," 29 September 2012. [Online]. Available: <http://www.multi2sim.org/>. [Accessed 21 June 2016].
- [20] "Main Page Gem5," 31 May 2011. [Online]. Available: <http://www.gem5.org/>. [Accessed 21 June 2016].
- [21] J. Hestness, S. W. Keckler, D. A. Wood, "A Comparative Analysis of Microarchitecture Effects on CPU and GPU Memory System Behavior," Workload Characterization (IISWC), 2014 IEEE International Symposium on, pp. 150--160, 26-28 October 2014.
- [22] P. Petev, "Cache coherence protocol". United State US Patent 11/118,902, 29 April 2005.
- [23] J. Power, A. Basu, J. Gu, S. Puthoor, B. M. Bechmann, M. D. Hill, S. K. Reinhardt, D. A. Wood, "Heterogeneous system coherence for integrated CPU-GPU systems", Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture, pp. 457--467, ACM New York, NY, USA.
- [24] A. Munshi, "The OpenCL specification." In 2009 IEEE Hot Chips 21 Symposium (HCS), pp. 1-314. IEEE, 2009.