CPU-GPU Processing

Zulfiqar A. Memon[1, *], Fahad Samad[1,*], Zafar Rehman Awan[2,+], Abdul Aziz[1,*], ~Shafaq Siraj Siddiqi[2]
[1]Department of Computer Science, FAST-National University of Computer and Emerging Sciences Karachi, Pakistan
[2]Department of Computer Science, Sukkur IBA University, Airport Road, Sukkur, Sindh, Pakistan

Summary
For achieving high performance from CPU-GPU Processors many steps were taken in the light of architectural advancement in design, computational techniques and optimization. We will first look at the state of art of GPU-CPU architectures and try to understand CPU-GPU design. Next, we are focusing on major enhancements in all areas of CPU-GPU Processors according to research studies. We will then gaze upon the synergy between CPU-GPU and their execution behavior. Ultimately, we will examine future for CPU-GPU systems.

Keywords:
CPU-GPU, CPU architecture design, computational technique, CPU-GPU execution behavior.

1. Introduction

CPU (central processing unit) is the primary component of the computer systems for performing arithmetic logic and control (I/O) specified instructions [1]. On the other hand, GPU (graphics processing unit) is specialized component intended to quickly control and modify memory to accelerate rendering of images for displaying. CPU and GPU have unique feature and strengths by combining intellectually higher performance can be gained. Architecturally, the CPU consist of few cores that interacts with caches that can deal with few instructions at a time. Conversely, a GPU is comprised of many cores that can deal with a large number of instructions at the same time. In today's computers, the GPU can now take on numerous multimedia tasks, such as accelerating transcoding (translating) video between many formats, image processing, pattern matching and others. To an ever increasing extent, the really difficult issues to deal with are those that have a parallel nature such as signal processing, video processing and image analysis etc.

2. GPU and CPU Background

In mid-2003, the highly advanced GPU and CPU from NVIDIA and Intel, individually, offered almost the same single-precision performance according to theoretical analysis. After nine years, in 2011, the performance of NVIDIA's highly advanced GPU was eight times greater than of Intel's highly advanced CPU. GPUs have the capability to provide much faster performance than relative to CPUs Because GPU have workloads that are more sensitive to aggregate throughput against single-threaded executions. GPU designers intelligently use multiple simple processing elements (PEs) rather than using large, highly expensive cores, and by providing much smaller die area to caches and control logic. SIMD (Single Instruction Multiple Data) control consist of multiple simple PEs by amortizing the area to instructions store and control logic. Rather than using large caches, GPUs is installed with memory latency with a combination of heavy multithreading and much higher memory bandwidth. Even if some threads are in waiting state for data from memory, there will also be many threads that can be executed in their place. This promises high utilization and high performance, but at the amount of increased latency for any individual thread.

These increases in performance have been combined with increases in the programmability and extensibility of GPUs.In an effort to permit GPU programs to create realistic images, GPU designs have been evolved from a fixed set of functions in pipeline stages into a complete programmable pipeline. Support for more complex control instructions and less fixed restrictions on memory access patterns are included in changes. In more Recently, GPUs added more features regarding of non-graphics such as support for higher-precision floating-point arithmetic, general-purpose computation and Error-correcting code (ECC) memory, are both essential for many applications, and global synchronization with low overhead, which is effective for iterative computations [12].

Table 1: Specifications of the top Intel CPU and AMD GPU as of March 2013

	Intel CPU	**AMD GPU**
Die Size (mm2)	513	352
Cores	10	32
Thread per core	2	2560
Total thread	20	81920
Total active threads	20	2048
Core Frequency (GHz)	2.0	0.4
Highest throughput (GFLOPS/s)	192	4096
Total cache (MB)	33.1	1.4
Memory channels	4	12
Highest memory bandwidth (GB/s)	43	288
Highest power consumption (W)	**130**	**250**

Manuscript received September 5, 2017
Manuscript revised September 20, 2017

As a solid example of the differences between GPUs and CPUs, Table 1 compares the best accessible Intel CPU with most advanced AMD GPU using different number of matrices [2]. The GPU's massive parallelism is pretty much clear in a large portion of these measurements. While CPU consist of 10 independent cores, the GPU has 32 independent core. A single CPU core has only two hardware threads while a single GPU core has 2,560 threads. CPU have 64 threads are active in a single clock cycle while GPU have 2,048 threads active at the same time, that is 100 plus times as many as the CPU. The difference in high performance of both CPU and GPU is not quite as high. Be that as it may, because the CPU execute instructions at a much higher clock rate and can perform more computational operations in a thread in single cycle. Generally, the GPU's highest computational output is 21 times higher than that of the CPU. The GPU has a smaller advantage against CPU when it comes to memory bandwidth. Because the GPU has memory channels that are 3 times as many that is in CPU also GPU have wider memory interfaces plus higher memory clock speeds as it reaches 6.7 times more in memory bandwidth.

3. CPU and GPU Architecture

In recent past few years there has been the fusion between CPU and GPU in a single die. Both Intel and AMD launched such processors in year 2011, known as Intel Sandy Bridge and AMD Fusion1 [4]. The CPU and GPU are fused together in such a way that their designs share a single memory system, unlike the separate memories employed in traditional discrete designs. Subsequently, one potential benefit is a much lower or even irrelevant overhead in communication between CPU and GPU. However, the main limitation is that of GPU is much weaker in both computational capability and memory bandwidth. The reason behind is primarily based on financial rather than used of technology. AMD will likely to discharge a idea of processor chip incorporating with a high-performance GPU, in light of the fact that it would risk cannibalizing sales of its most selling discrete GPUs. Intel may possibly be free from such financial requirements, yet it doesn't deliver high-performance GPUs. For some reason AMD called these processors as Accelerated Processing Units (APUs) rather than Fusion processors. Their applications the reduced data transfer rate plus longer computation time and a Fusion GPU can speedup more than powerful dedicated GPU. However, discrete GPUs will likely remain the optimal choice for most applications requiring high performance.

For utilizing GPUs for non-graphics computational requirements which use corner cases of the graphics APIs. To use APIs for general purpose, programmers have to mapped data to the accessible shader buffer memory and operate data via the graphics pipeline. GPUs vendors have to add additional hardware and software support to non-graphics workload as demanded. NVIDIA's CUDA and AMD's CTM explicitly added hardware to support general purpose computations with providing multi-threaded hardware by using high level language interface[3][4]. The programmer can operate on data on separate GPU memory address space.

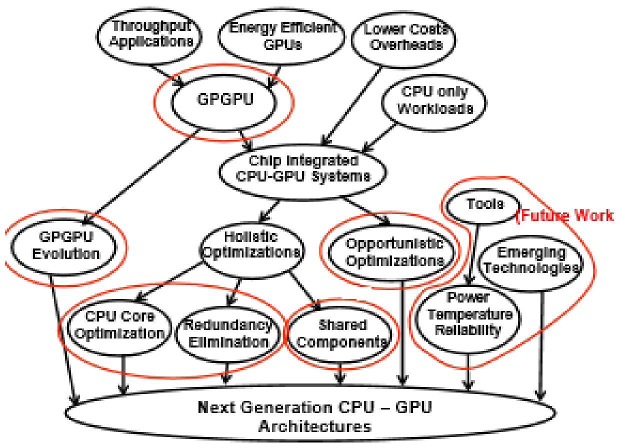


Fig. 1. **Evolution of CPU-GPU architectures.**

As we had discussed AMD's Fusion APUs, Intel's Sandy Bridge and also ARM's MALI provide solutions that integrate general purpose programmable GPUs together with CPUs on the single die[4][6][15]. In this model, the CPU and GPU do share memory and a common space. These solutions can be programmed either by using OpenCL or DirectX. Combining a CPU and GPU on the same chip has various advantages [8].

- Firstly, it reduced cost due to the use of shared structures.
- Secondly, this increases to performance because no data transfers are required between the CPU and GPU as they shared same memory.
- Thirdly, programming is much simpler because GPU memory management is not required. It also enables more directions in system development.

It reduces data transfer costs plus increased bandwidth and opens new optimizations paths that were not available. Now, there are more problems to consider. According to a theoretical survey, the research and development for CPU-GPU systems in figure 1. The figure 1 shows the facts that have cause the development of current GPGPU systems.

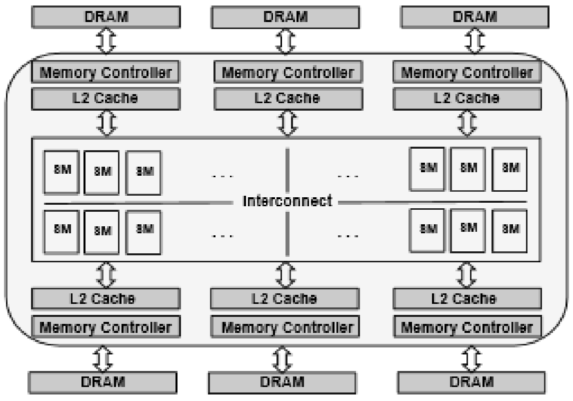


Fig. 2. Contemporary GPU architecture.

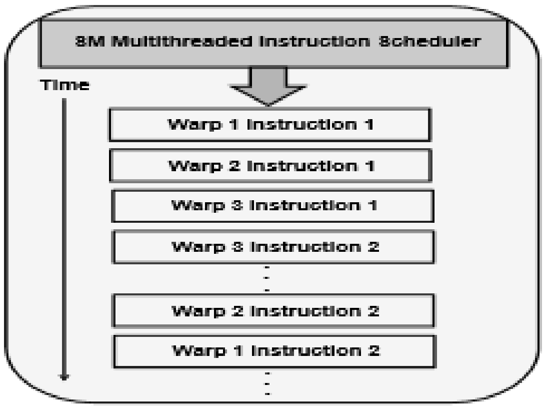


Fig. 4. Example of warp scheduling.

4. General Purpose GPU Architectures

General-purpose computing which is done in GPU as GPU can compute only for graphical purposes but it can handle application computation as well which is traditionally handled by CPU. As GPU have many cores that can operate on graphics much effectively and much faster than a CPU. The use of data had to transform into a graphical data in order to compute in GPU.
Latest GPU consist of fixed function graphics pipeline which contains pixel fragment processors executing pixel shader programs and vertex processors that are executing vertex shader programs.
Pixel fragment processing consist of operation on rasterizer output to filling with interpolated values in interior of triangle primitives. Vertex processors operate on point, line and triangle vertex primitives[11].

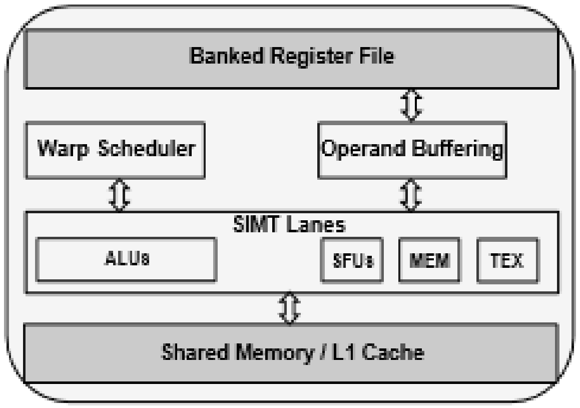
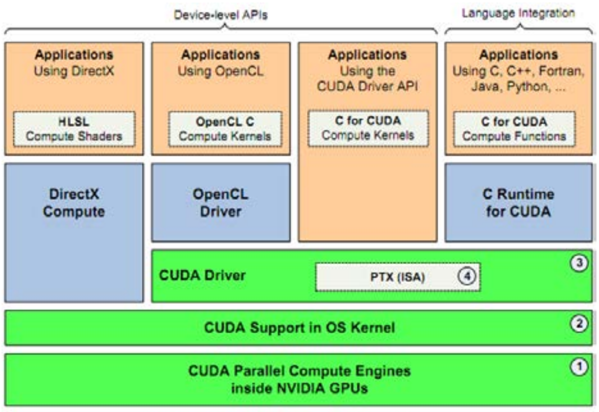


Fig. 3. Streaming Multiprocessor (SM) architecture.

In any case, instability in current workloads influenced a pixel processor and unified vertex design. NVIDIA's Tesla first implemented Unified processing that increases in resource utilization and align it in single generalized design. In Figure 2 GPU design contains streaming multiprocessors (SMs), on-chip L2 cache and 6 high-bandwidth DRAM channels. In Figure 3 SM contains 32 single instruction multiple thread (SIMT) issued a number 32 instruction per cycle per thread. 32 threads are arranged in group known as wrap. All wraps then execute using a general program counter [8].
NVidia's CUDA allowed programmer that they do not need to use the graphical concepts in order gain high-performance from GPU. As it has hardware that support Microsoft's DirectCompute and Apple/Khronos Group's OpenCL. GPGPU pipelines can speed up GPU without any need of data conversion in graphical form.
OpenGL or DirectX are high-level shading languages as a result graphics accelerators can be used for non-graphical applications in later 90s. Many problem like protein folding, stock option pricing have been worked out on graphic accelerators exhibit notable speedup. Implementing graphic accelerators turned out to be difficult because graphic APIs missing programming feature so the implementation was quite difficult as they had to be expressed in term of graphic and shader concepts. Floating point computation was impossible in the beginning then GPU were re-developed as highly threaded streaming processors with a programming model extending c [9].
Compiler and runtime system are then permit to utilize as GPU as general-purpose processor to increase performance.

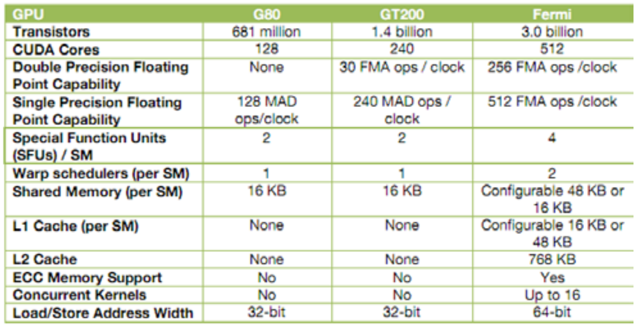

Device-level APIs
Language Integration
Applications Using DirectX
HLSL Compute Shaders
Applications Using OpenCL
OpenCL C Compute Kernels
Applications Using the CUDA Driver API
C for CUDA Compute Kernels
Applications Using C, C++, Fortran, Java, Python, ...
C for CUDA Compute Functions
DirectX Compute
OpenCL Driver
C Runtime for CUDA
CUDA Driver
PTX (ISA)
CUDA Support in OS Kernel
CUDA Parallel Compute Engines inside NVIDIA GPUs

CUDA Software Architecture

Latest NVIDIA's CUDA was purposely designed for the support the dynamic use of GPUs. Table 2 shows features of many GPUs considered in the sequel [14].

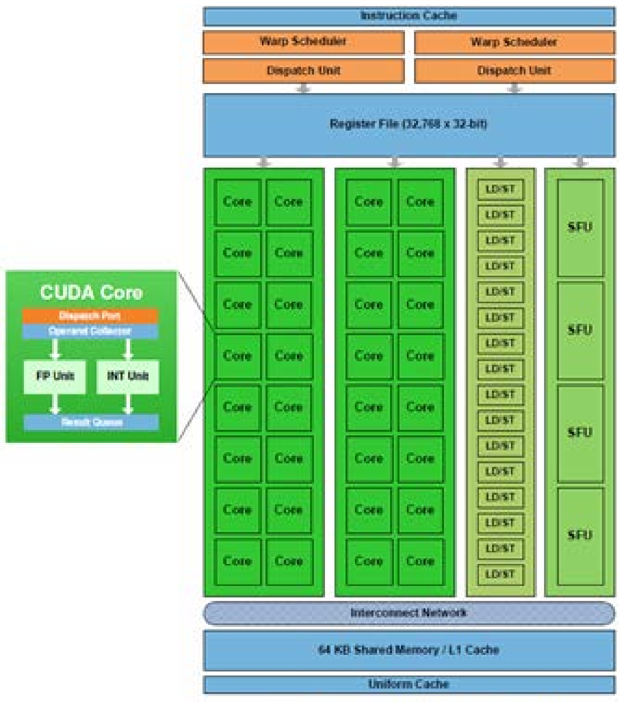
GPU	# cores	Clock (GHz)	Memory (GB)
GeForce 7800 GTX	24	0.58	0.512
GeForce 8600 GTX	32	0.54	0.256
GeForce 9600 GT	64	0.65	0.512
GeForce GTX 260	192	1.4	0.9
GeForce GTX 280	240	1.296	1
GeForce GTX 285	240	1.476	1
GeForce GTX 295	240	1.24	1
GeForce GTX 480	480	1.4	1.536
Tesla C1060 [6]	240	1.3	4
T10 (Tesla S1070)	240	1.44	4
C2050 [1]	448	1.15	3

Table 2
OVERVIEW OF NVIDIA GPUS QUOTED IN THE PAPER



GPU	G80	GT200	Fermi
Transistors	681 million	1.4 billion	3.0 billion
CUDA Cores	128	240	512
Double Precision Floating Point Capability	None	30 FMA ops / clock	256 FMA ops /clock
Single Precision Floating Point Capability	128 MAD ops/clock	240 MAD ops / clock	512 FMA ops /clock
Special Function Units (SFUs) / SM	2	2	4
Warp schedulers (per SM)	1	1	2
Shared Memory (per SM)	16 KB	16 KB	Configurable 48 KB or 16 KB
L1 Cache (per SM)	None	None	Configurable 16 KB or 48 KB
L2 Cache	None	None	768 KB
ECC Memory Support	No	No	Yes
Concurrent Kernels	No	No	Up to 16
Load/Store Address Width	32-bit	32-bit	64-bit

Improvements in CUDA GPU Architecture



Fermi Streaming Multiprocessor (SM)

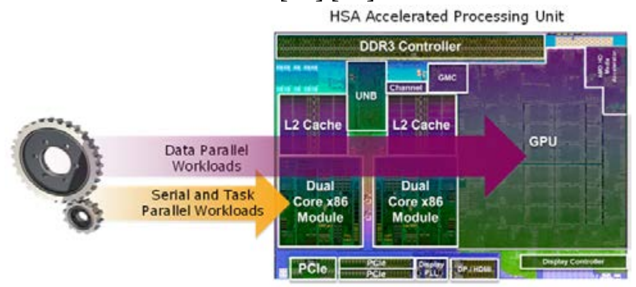
Recently, Fermi design is just like last generation NVIDIA's GPU Design. It is built around scalable multithreaded Streaming Multiprocessors which features more than 512 CUDA cores with 3 billion transistors. One core of CUDA can execute a floating point per clock cycle in a thread. Every core in CUDA has pipelined Arithmetic Logic Unit and Floating Point Unit. Fermi is built on the basis of IEEE 754-2008 floating-point standard CUDA has the library known as Basic Linear Algebra Subroutines (cuBLAS) that support programmers to solve large scale problems. The cuBLAS library is a standard BLAS library [7].

Traditionally, CPU was only way to communication GPU in order display on device. As time passes, GPU is permitted that it can first analyze data like image, 2D or 3D video as it understand these kind of data much quickly as it hard for CPU. As straightforward answer that GPU have to render values on some view like computer graphics now it has passed back to CPU in order to make some adjustments in overall view. A more complex example that GPU need to detect edges output values to represent something as it have fast and local hardware to access and operate more easily. GPGPU is much of logical concept not physical that is type of procedure not a piece of hardware. Nonetheless specific hardware design that can actually increase the effectiveness of its pipeline.

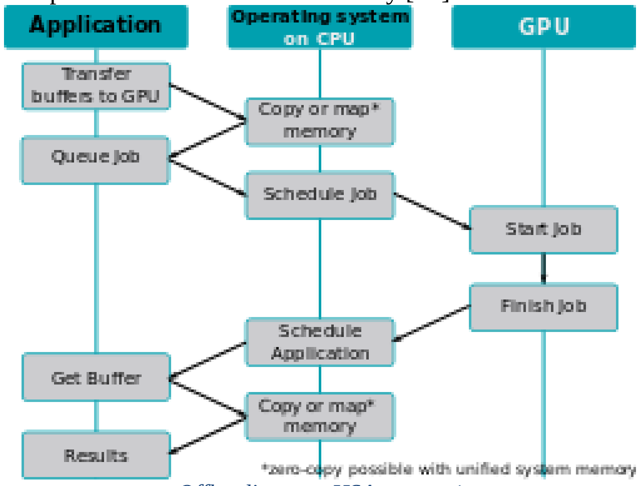
The following are some of the areas where GPUs have been used for general purpose computing General purpose computing same of major areas like computer clusters or any of its variant as in high-performance computing cluster or so on in Grid computing, Load-balancing clusters or in learning algorithms machine learning and data mining computations, Cryptography and cryptanalysis or Neural networks and much bigger field of bio information like DNA Analysis, Protein searching and in molecular dynamics.

5. Heterogenous Computing

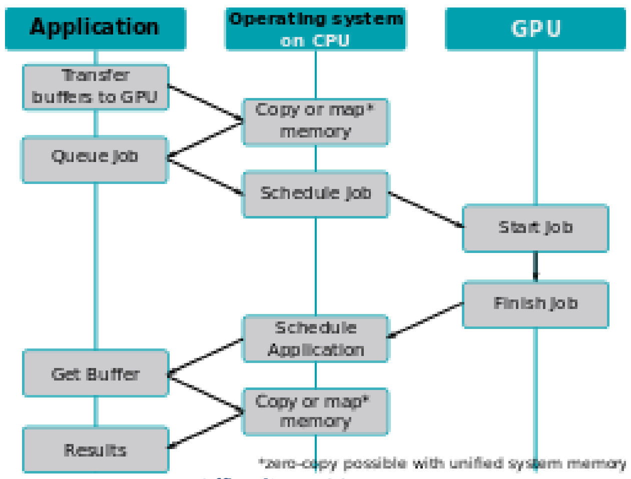
Use of two different multi-core processors that are specialist in their specific tasks. Heterogeneous System Architecture (HSA) design multiple processor types both CPU and GPU in one die that effectively handle their respective works at their best that is GPU can render graphics and can perform computation on large scale while CPU can perform scheduling, sequential tasks on OS. Demands of new user experience to access more natural interface like speech, gesture and devices to manage different kind of contents [10] [16].



HSA is introduced by HSA Foundation and many others (AMD and ARM). The aim is to improve latency between CPU and GPU. Devices that are using HSA are more compatible with programming like moving data between devices that is currently done with CUDA and OpenCL. It is easy for programmers that they can perform off load computation to the GPU much easily [13].

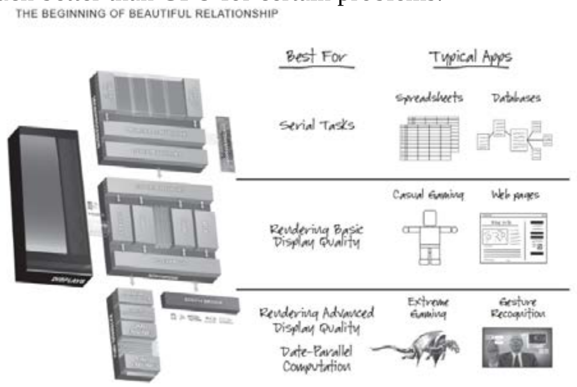


Offloading non-HSA system 1



Offloading HSA system 1

Effort to increase GPU for general purpose computation takes a coincided for consumer culture. Recently consumer are hungry for rich visual experiences but at the same time new mainstream OS had to perform better.
To increase performance, power and scalability of multicore CPU led designers use GPU's vector processing capabilities. Advanced vector processors consist of thousands of cores that can operate in same time to compute. This lead GPU to be much effectively to deal with intensive computation in a large scale.
As GPU vector processing is not solution of everything like overhead associated with small data array can be resolved much faster by CPUs scaler approach. As CPU still do much better than GPU for certain problems.


THE BEGINNING OF BEAUTIFUL RELATIONSHIP
Best For
Typical Apps
Serial Tasks
Spreadsheets
Databases
Rendering Basic Display Quality
Casual Gaming
Web pages
Rendering Advanced Display Quality
Data-Parallel Computation
Extreme Gaming
Gesture Recognition

Heterogeneous computing is really amazing that it can process video in HD to translation and interpretation in real-time. It can improve parallelism and power efficiency.

6. Future of CPU-GPU Systems

In modern era, GPUs and CPUs are increasingly being presented as imperative co processors, as they cannot interchange each other. Heterogeneous computing has been effectively explored to utilize in several applications ranging from natural sciences to engineering and still many challenges remain.

As it also appears the industry is moving towards heterogeneous computing. AMD, NVidia and Intel are all pointing toward same directions that journey would be full of challenges. Heterogeneous computing is listed as major advancement in the field of computer architecture that it will increase power efficiency. *By* 2020 it might be possible a 4W processor can rival 100W processor. Heterogeneous computing are now an emerging trend in every domain of computing, you can find in all high-performance computers, servers, tablets, mobile phones and embedded devices.

7. Conclusion

In this work we investigate the architecture CPU-GPU systems. We looked at CPU-GPU background. We then discuss current architecture CPU-GPU. Then we talked about general purpose GPU. We examine hybrid computing in modern architecture. Lastly, we what will be the future of CPU-GPU systems.

References

[1] CPU and GPU
[2] http://www.wikipedia.org
[3] NVIDIA GPU and Intel CPU family comparison articles. http://www.wikipedia.org
[4] NVIDIA’s next generation cuda compute architecture: Kepler GK110. Technical report, 2012.
[5] AMD http://www.amd.com/.
[6] AMD OpenCL Programming Guide. http://developer.amd.com.
[7] ARM Mali-400 MP. http://www.arm.com
[8] NVIDIA Corporation. CUDA Toolkit 4.0. http://developer.nvidia.com/ category/zone/cuda-zone
[9] The Architecture and Evolution of CPU-GPU Systems for General Purpose Computing-Manish Arora
[10] Improving Resource Utilization in Heterogeneous CPU-GPU Systems–Michael Boyer.
[11] A Survey of CPU-GPU Heterogeneous Computing Techniques
[12] Sparsh Mittal, Oak Ridge National Laboratory Jeffrey S. Vetter, Oak Ridge National Laboratory and Georgia Tech.
[13] Computing Performance Benchmarks among CPU, GPU, and FPGA Christopher Cullinan, Christopher Wyant and Timothy Frattesi
[14] Recent Advances on GPU Computing in Operations Research
[15] Vincent Boyer and Didier El Baz
[16] Heterogeneous system architecture helps AMD and ARM deal with demands http://www.theinquirer.net/inquirer/feature/2435169/heterogeneous-system-architecture-helps-amd-and-arm-deal-with-mammoth-compute-demands
[17] CUDA GPUs https://developer.nvidia.com/cuda-gpus
[18] Intel www.intel.com
[19] What is Heterogeneous Computing? http://developer.amd.com/resources/heterogeneous-computing/what-is-heterogeneous-computing/