International Journal of Recent Engineering Research and Development (IJRERD)
ISSN: 2455-8761
www.ijrerd.com || Volume 03 – Issue 08 || August 2018 || PP. 53-62

Multi-Processors vs Multi-FPGAs vs GPUs – Which is Most Appropriate for Embedded Systems Development

Marcus Lloyde George[1], Shaun Bhimlal[2], Simeon Ramjit[3], Mustafa Nakhid[4]

University of the West Indies, St. Augustine, Trinidad and Tobago

Abstract: Technology is continuously growing and ever-present especially in industrial and commercial uses. Three technologies that are mainly considered in industrial and commercial applications are FPGA, Processor and GPU technology. This paper seeks to provide a comprehensive study of the three technologies and compare them in terms of key performance metrics. This is done through extensive desk research, analysis of case studies and grounded theory. Results were presented in the forms of graphs and tables. A detailed discussion is provided on the three technologies. It was concluded that the optimum technology for an application depends heavily on the application itself as well as the resources and demands of the user.

Keywords: Multi-processors, Multi-FPGA, GPU, Processor Cores, IP Threads, Parallel Processing

1. Introduction

After the Information Age, there has been a multitude of technologies that are omnipresent today. Among these technologies are specialized electronic circuits such as processors, graphics processing units (GPUs) and Field Programmable Gate Arrays (FPGA). Multi-FPGA systems are a growing area of research. They offer the potential to deliver high performance solutions to general computing tasks, especially for the prototyping of digital logic [1]. This research has shed light into 3-dimensional stacking of resources to create multi-processors and multi-FPGA. For many applications, ranging from industrial to commercial to even personal applications, these technologies are contested over, which brings forth the question: Which is the superior technology?

The aim of this paper is to analyze and compare multi-processors, multi-FPGAs and GPUs to come to an educated conclusion on which device is best and why. The paper has the following research objectives: To identify the operational and performance metrics of each device; To research the development of processors into multi-processors, the advantages and disadvantages that came with this upgrade and examples of applications that utilize this device; To research the development of FPGAs into multi-FPGAs, the advantages and disadvantages that came with this upgrade and examples of applications that utilize this device; To research GPUs and the advantages and disadvantages of using this device, and examples of applications that utilize this device; To compare the operational and performance metrics of each device. A conclusion of the findings is also included at the end of the paper.

2. Overview of FPGA Technology

Field Programmable Gate Arrays (FPGAs) are semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects [2]. These devices can be reprogrammed to adjust the functionality required for a particular application. This is the main difference between FPGAs and Application Specific Integrated Circuits (ASICs), which are custom made for a specific task.

The first commercial FPGA was designed and released by a company known as XILINX in the 1980s. This primary design was small, expensive and was sluggish in terms of performance. This device was equipped with a measly 1200 logic gates. Today, FPGAs contain millions of logic gates and can operate at speeds up to 300MHz. The prices of these boards can be as low as $10 US and as a result, manufacturers use them for a wide variety of products. Even factories which seek to automate their processes may benefit from the beneficial performance to cost ratio of modern-day FPGAs.

Previously, FPGAs were outperformed by the ASIC technology [3]. Studies have shown that designs to be implemented on FPGAs would take, on average, 40 times as much area, 12 times as much power and run at approximately one third the speed of the implementation of the same design on an ASIC. Recently, the FPGA have come to rival that of ASIC by minimizing power usage, materials cost, implementation area and maximizing speed. Designs that would require multiple ASICs can now be achieved on one FPGA.

Figure 1: Virtex-7 Chip developed by Xilinx
Source: [5]

FPGAs have many advantages when being considered for a design process. Most providers of FPGA technology also support the development and design process by providing design tools which are powerful and easy to use. These vendors also provide excellent documentation and personal support for the products. One of the key advantages of FPGA technology is the ability to be modified and adjusted at any stage of the design process. FPGAs also allow for dynamic reconfiguration of the hardware. These circuits also have shorter time to market and lower, non-recurring, engineering costs. [4]

Comparisons can also be made between FPGAs and Complex Programmable Logic Devices (CPLDs). The main difference between the two technologies are architectural. CPLDs are less flexible than FPGAs, the advantages being more predictable time delays and higher logic to interconnect ratio. On the other hand, FPGA architectures are more flexible in terms of different designs that can be implemented on one, the tradeoff being that it is more difficult to design for. FPGAs have much more resources than CPLDs, however CPLDs contain embedded flash memory to store the configuration whereas FPGAs require external memory.

However, even though there are many advantages to FPGA technology, there are also limitations or disadvantages when using FPGAs. The first limitation is known as design partitioning. Complicated designs for complex systems will not fit in a single FPGA, in fact, these designs would have to be partitioned across several devices. This task is tedious, error-prone and difficult to accomplish. Another limitation is difficulty in troubleshooting or debugging. Despite having internal logic analyzers, the FPGAs are difficult to troubleshoot. This is because the logic analyzers have limitations which include: support for only single FPGA debug, limited memory size using FPGA internal memory and long place and route times to change probes. The third limitation of FPGAs is the performance. Issues related to printed circuit boards (PCBs) such as signal routing, capacitive loading and impedance matching all limit how fast the FPGA can run. Finally, the reusability of boards. As projects and designs grow, previously used boards may no longer be used for projects as the design will no longer fit on a single FPGA, and upgrades may be necessary.

Due to these limitations of FPGAs, research has been undertaken into multi-FPGAs. To minimize the size and power consumption of FPGAs even further, companies such as Tabula and Xilinx have begun to stack FPGAs. Xilinx's approach is to stack several active FPGA dies contiguously on a single piece of silicon that carries passive interconnects. This multi-die construction allows different parts of the FPGA to be made with several process technologies. This is because the process requirements would be different between the FPGA fabric and the high-speed serial transceivers. Such an FPGA is coined a heterogeneous FPGA.

Multi-FPGA technology is becoming increasingly desirable, especially for scalable systems, due to their power, size reduction and performance benefits. However, the cost of development of these technologies, interconnect density and thermal dissipation inhibit widespread adoption of Multi-FPGA technology. These disadvantages of the multi-FPGA technology are still being researched and worked upon, in the hopes that multi-FPGA technology become more widely used[5].

FPGAs may be used to solve almost any problem which is computable. FPGAs were competitors to CPLD technology to implement what is known as glue logic for PCBs. However, as the capabilities of the FPGA increased, it surpassed the CPLD technology and are now marketed as full systems on chips. Another typical use of FPGA technology is hardware acceleration, where an FPGA is used to speed up aspects of an algorithm by sharing the computations with a processor. Some common applications of FPGA technology are:

- Aerospace and Defense
- ASIC Prototyping

- Audio, Communications and Multimedia
- Automotive Systems
- Consumer Electronics
- Wireless Communications

3. Overview of Multi-Processor Technology

Processors or Central Processing Units (CPUs) (Figure 2) refer to a specific type of logic circuitry execute a computer program. They fetch, decode, execute instructions and writeback the result to the appropriate memory addresses[i]. CPUs contain the Arithmetic and Logic Unit (ALU), Control Unit (CU) (Figure 2) and in some cases a Floating-Point Unit (FPU). The memory addresses referred to previously are in the L1, L2 and in some cases L3 caches on the CPU chip itself. These different cache levels have different bandwidths and capacities in order to ensure that the processor has a steady flow of instructions to execute. Since the processor can either be faster or slower than some hardware peripherals the caches act as buffers and it is usually faster than fetching instructions from RAM[2]. CPU types are mainly based off the Von Neumann architecture; recently however the word architecture more commonly refers to 32-bit or 64-bit. The main difference between the two is the amount of RAM (Random Access Memory) that it can handle. The 32-bit CPU can handle up to 4GB whereas the 64-bit can handle up to 8TB. This is possible since the increased bit count means that there can be more memory addresses supported [3].

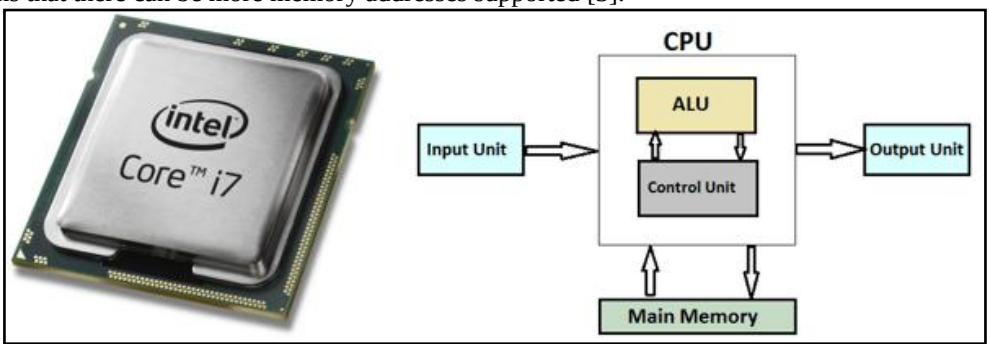

Figure 2: CPU (left) (Intel 2016) CPU Block Diagram (right)
Source: [12]

Another main feature of the CPU is the core count. Modern CPUs have multiple cores on a single chip meaning that the CPU can perform more than one task (thread) simultaneously. As opposed to switching between multiple threads at a high frequency which increases the overall throughput[ii]. In some cases, one core can handle two threads; this is known as Intel's equivalent of "Hyperthreading" [3]. Since the cores are on the same chip the connection between them is faster and the power consumption is lower compared to a multi-processor system where there are two or more physical CPUs on one board. On a single CPU system there is one operating system and the CPU handles when and which tasks are executed. This method, along with a single CPU, has its limitations. The demandfor computing power and parallel processing is still on the rise despite the transistor count on CPUs levelling out and Moore's Law plateauing. The alternative to increasing clock speeds and shrinking transistors would be to develop a multi-processor system.

Multi-processor systems are used where there is a demand for parallel processing. In most cases regular home use or even gaming will not benefit from a multi-processor system (not to be confused with multiple Graphics Processing Unit system) since games and everyday programs are not designed to be used in a multi-CPU system. Multi-processor or multi-CPU systems, are commonly described in terms of symmetry, data streams and coupling types. Immediately we can see that there is a difference on how single CPU and multiple CPU systems are described. Single CPU systems focus on the computing power of the chip i.e. it's clock speed, core count and multi-threading capabilities. Multiple CPU systems focus on how tasks and task data are managed as well how each CPU communicates with each other which is referred to as the architecture. The architecture also dictates the topology as shown in Figure 3.

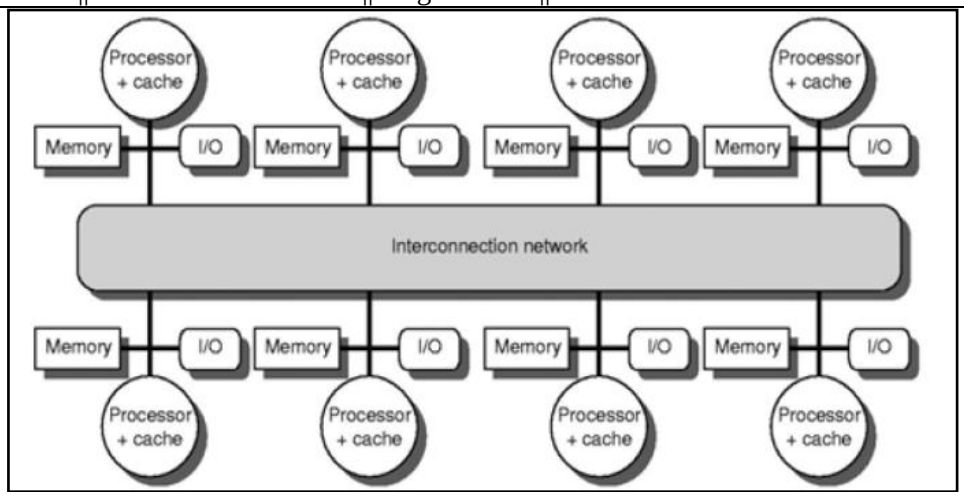

Figure 3: UMA Machine (Left) NUMA Machine (Right)
Source: [4]

The type of symmetry, data stream, coupling and architecture depends on the application of the system. Systems built for redundancy may lean towards Multiple Instruction Single Data (MISD) streams whereas vector processing will require SIMD. The interconnection network aims to keep all processes busy in order to improve throughput. Common switching methods for a UMA symmetry are uni-bus, crossbar and multistage switching. NUMA symmetry does not have a switching method due to the nature of it is operation which is essentially using a database to keep track of task memory addresses. Another key feature of the multi-CPU system is the operating system [3]. There are three models that can be implemented:

- Each CPU Has Its Own Operating System – Each OS has private memory and shared I/O
- Master-Slave Multiprocessors – OS is run by one CPU and that CPU commands the remaining CPUs
- Symmetric Multiprocessors – The OS is one memory and each CPU runs a copy of it as well as shared users

The remaining two features are synchronization and scheduling. Synchronization refers to how a parallel system handles one task. Multiple CPUs accessing the same memory location at the same time is never desired; as such mutexes and semaphores are used in order to protect memory in use. The basic concept is that mutexes and semaphores are keys to a resource or memory location. Once the memory is being used the OS 'locks' the location and only the task or CPU that was first to access gets the key, which is 'returned' when the CPU is finished with the memory location. If any task or CPU needs to use that memory location the CPU will have to either wait or it might get switched to a new thread depending on the configuration of the system. Scheduling refers to how tasks are run on each CPU with regards to time. Some methods of scheduling are as follows[iii]:

- Timesharing
- Gang Scheduling
- Smart Scheduling
- Affinity Scheduling
- Space Sharing

As we can see the emphasis of a single-CPU and a multi-CPU system is significantly different and therefore their applications as well. Table 2 gives a summary of where each system is used and Table 3 gives advantages and disadvantages between the two [4].

Table 1: Applications of single and multi-processor systems

Single- CPU	Multi-CPU
• Everyday use (Word processing, Web browsing) • Gaming	• CGI Render Farms • Protein Folding • Realtime robotic controllers[iv]

Table 2: Single and Multi-Processor Comparison

Single- CPU	Multi-CPU
• One task can be run per application • Only one user at a time • Low latency between cores for communication • Less space since multiple cores are onboard one CPU • Lower costs since only one CPU is needed together with a smaller motherboard • Consume less energy • All OS versions recognize single CPU systems, cheaper to use any OS	• More than one task can be run per application • Multiple users supported • High latency between CPUs for communication • Uses more space • Higher costs since more hardware is required • Special OS required to implement multi-CPU system, higher software cost.

4. Overview of GPU Technology

The graphics processing unit (GPU) converts video data into electronic signals that can then be displayed as images on your monitor. It is a microprocessor that is specialized in video calculations, which are mathematically intensive tasks. Millions of calculations must be done for a single frame in a three-dimensional image, just to give an example. It is for this reason that the GPU handles the production of accurate graphical images instead of the CPU which allows the CPU to run at full capacity without being slowed down by the strenuous calculations required to produce an image. [6]

5. Comparison of FPGAs, Multiprocessors and GPUs Technology

5.1 Comparison of FPGAs and GPUs

This leads to the question; which technology is superior to the other? The short answer is that it depends on the application, but here we will discuss the differences between the different technologies and compare the performance of these technologies to one another. First, we shall consider FPGAs versus GPUs. FPGAs perform simultaneous fixed-point operations by using an almost entirely hardware based programming approach. On the other hand, GPUs perform parallel processing of floating-point operations by utilizing thousands of small cores. Much of the differences between both technologies, and their applicability to certain projects, are derived from these architectural differences.

Comparing the two technologies is not straightforward as the performance of FPGAs are measured in GMACS whereas GPUs are measured in GFLOPS. GPUs are better than the FPGA technology in terms of development effort, cost, floating-point processing power and flexibility. However, FPGA technology also provide vast processing capabilities whilst minimizing spatial requirements and thermal management and maximizing power efficiency. Due to this, FPGAs can be integrated in several different environments easily.

Interfacing is another advantage of the FPGA. GPUs must use the PCI-e standard to interface with devices. If another standard is to be used, then it would require additional electronics. FPGAs however, have great interface flexibility which saw an advancement due to the integration of programmable logic with CPUs and other peripherals in SoCs. Another indicator of performance to consider is latency. GPUs improve the performance of a CPU, but FPGAs provide deterministic timing in the order of nanoseconds. This aspect of FPGAs is key in applications such as, audio coding, encryption, control or network synchronization that require small latencies.

GPUs are cost efficient in terms of both hardware installation and development whereas FPGAs need specialized engineers who are proficient in HDL, electronics, algorithms and communications to name a few. The costs between the two technologies isn't as strong an issue as the time and effort that the engineer must expend to facilitate the design to an application. However, this is being diminished for FPGA technology with the introduction of new development environments and auto-coding techniques. The figures below were taken from a study done by BERTEN in 2016, where they summarized the above points:

Figure 4 is a simple bar graph comparing GPUs and FPGAs in terms of processing power and cost. From the above chart is clear that FPGA technology has better power efficiency than GPUs, however, GPUs are more cost effective.

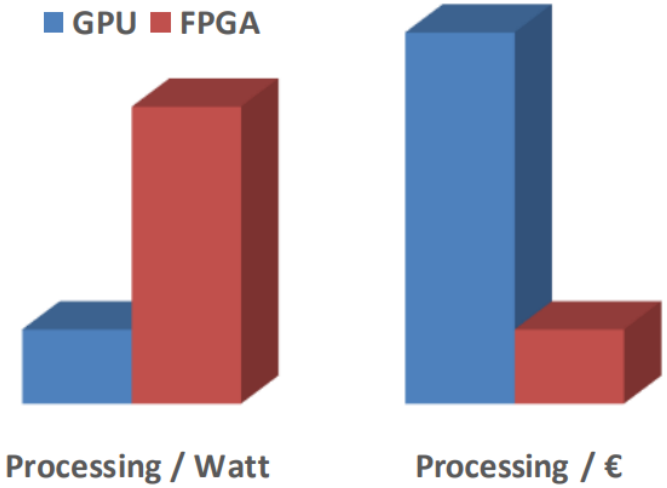

Figure 4: Processing Efficiency FPGA vs. GPU
Source:[13]

Figure 5 below shows the areas where each of the technology excels. The blue area shows that GPUs excel in floating-point processing, processing cost efficiency, flexibility and to a lesser extent, backward compatibility, development and timing latency. It pales in comparison to the FPGA technology in areas such as size, interfacing and processing power efficiency. The red area indicates that FPGA technology excels in timing latencies, interfacing and to a lesser extent, processing power efficiency, floating-point processing and size. However, in terms of backward compatibility and processing cost efficiency it is vastly inferior to GPUs.

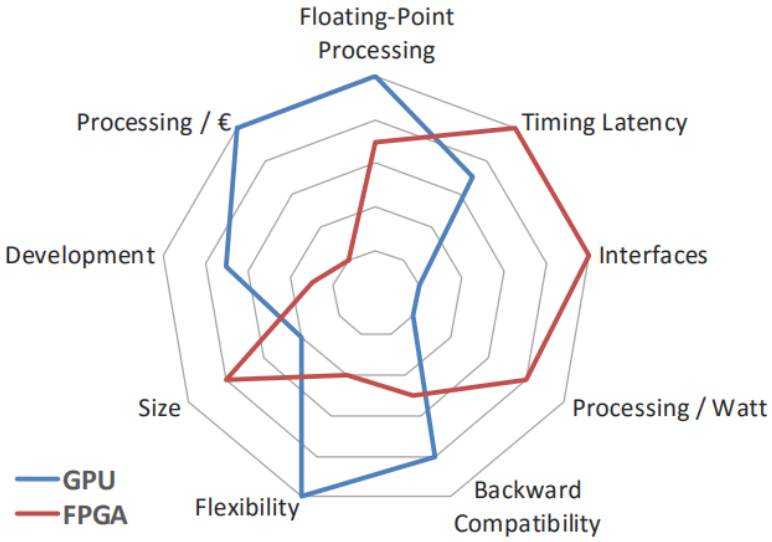

Figure 5: Qualitative Comparison of FPGA and GPUs
Source:[13]

The Table 4 below summarizes the information shown in Figure 5.

Table 4: Evaluation of FPGA and GPUs characteristics
Source: [13]

Feature	Analysis	Winner
Floating-point Processing	The total floating-point operations per second of the best GPUs are higher than the FPGAs' with the maximum DSP capabilities.	GPU
Timing Latency	Algorithms implemented into FPGA provide deterministic timing, with latencies one order of magnitude less than GPUs.	FPGA
Processing / Watt	Measuring GFLOPS per watt, FPGAs are 3-4 times better. Although still far away, latest GPU products are dramatically improving the power burning.	FPGA
Interfaces	GPUs interface via PCIe, while FPGA flexibility allows connection to any other device via -almost- any physical standard or custom interface.	FPGA
Backward Compatibility	Software developed for older GPUs will work in the new devices. FPGA HDL can be moved to newer platforms, but with some reworking.	GPU
Flexibility	FPGA lacks flexibility to modify the hardware implementation of the synthesized code, being a no-problem issue for GPUs developers.	GPU
Size	FPGA's lower power consumption requires less thermal dissipation countermeasures, implementing the solution in smaller dimensions.	FPGA
Development	Many algorithms are designed directly for GPUs, and FPGA developers are difficult and expensive to hire.	GPU
Processing / €	Mid-class devices can be compared within the same order of magnitude, but GPU wins when considering money per GFLOP.	GPU

However, this is a qualitative analysis on the two technologies. Performance metrics should also be evaluated to give a quantitative comparison between the two technologies. Several top-tier GPU models were compared to top-tier FPGA technology in terms of key performance indicators such as, processing single, power efficiency, price and price efficiency. The findings were recorded in a table 5 as shown below:

Table 5: KPIs of GPUs and FPGAs
Source: [13]

	Model	Processing Single	Power Efficiency	Approx. Price	Price Efficiency
GPU	GeForce GT 730	0.69 TFLOPS	7 GFLOPS/W	80 €	0.10 €/GFLOPS
	Radeon R9 390X	5.91 TFLOPS	16 GFLOPS/W	420 €	0.07 €/GFLOPS
	Radeon R9 Fury X	7.17 TFLOPS	20 GFLOPS/W	600 €	0.08 €/GFLOPS
FPGA	Artix-7 200T	0.65 TFLOPS	72 GFLOPS/W	190 €	0.29 €/GFLOPS
	Kintex-7 480T	1.80 TFLOPS	72 GFLOPS/W	2,500 €	1.39 €/GFLOPS
	Virtex-7 690T	3.12 TFLOPS	78 GFLOPS/W	11,200 €	3.59 €/GFLOPS

The analysis concludes that GPUs are cost efficient with excellent floating-point processing power as well as power efficient FPGAs with dynamic interfaces and minimum latencies. Thus, the selection of the device ultimately depends on the application, budget and development capacity.

5.2 Comparison of Multi-processor and Multi-FPGA

Comparing these two technologies is difficult since neither have the same performance metrics, however they do have some similarities since both are used in parallel processing. FPGAs allow a user to manually configure the physical circuit configuration whereas a processor has a fixed configuration and the user can only instruct that one configuration. FPGAs therefore allows a user to get a more hardware tailored and possibly more efficient solution as opposed to a multi-processor approach. Even though FPGAs are clocked

within the megahertz range (up to 300MHz) and modern processors are in excess of 4GHz, more work is done per clock cycle for a single FPGA as opposed to a multi-processor system where data flow is limited by bus size (32 or 64 bit)[v]. Also, the programming language used in processors (Assembly, C) is sequential in nature, whereas VHDL or HDL is parallel and offers the ability to pipeline which allows an increased throughput. The true parallel nature of HDL and it ability to use state machines eliminate the issue of jitter associated with thread prioritization and software execution. A small point to note is that FPGAs are standalone devices in the sense that the development board is all you need to get a system up and running.

With a processor, a motherboard, operating system, additional RAM and other hardware components are required before programming it is possible. Due to the user describing how the hardware is configured on an FPGA, multi-FPGAs do not suffer from problems with scheduling tasks, sharing memory, or operating systems like multi-processor systems. However, a multi-FPGA system cannot handle multiple users the way a multi-processor can. This is largely due to the operating systems implemented on multi-processor platforms[vi]. FPGAs also scale with memory, meaning that the more complex a system is the more memory it will use to be implemented. Whereas processor scale with instruction cycles i.e. the more complicated the program the longer it would take to run, though it does indirectly take more memory to run (RAM) in some instances but since the memory type is different it is not comparable.

5.3 Comparison of GPU and Multi-processors

When comparing CPUs and GPUs it should be noted that the GPU is a specialized chip and cannot replace CPU, which is the main component of any computer. If the CPU is the brain, the GPU can be likened to the brawn of a computer. The CPU is actually capable of carrying out the functions of the GPU, but it will do so at a considerably slower rate. Both CPUs and GPUs can work in tandem.Some benefits of having a GPU are:

- Alternate processing unit dedicated to graphic processing, thus relieving work from the CPU resulting in better performance overall
- Faster desktop response time
- Most GPUs in the market today can support multiple monitors

GPUs are responsible for handling anything that needs to be displayed on your monitor. This can be anything from rendering videos, games, screensavers and your monitors background picture. GPUs are most notably used to enhance multimedia performance, for example, in games. GPUs can be sold separately from the main unit and the price varies according to its specifications. Laptops usually come with onboard or integrated graphics cards therefore, switching them out is usually not possible[6]. The Table 6 below shows a comparison between microprocessors and GPUs. Most of the information was sourced assuming the GPUs were part of SoC (system on chip) and ICs (integrated circuits)[8].

Table 6: Microprocesors vs GPUs
Source: [8]

	Microprocessors	GPUs
Chip Count	Requires supporting chips	Single
Cost	High	Price increases according to specifications
Data/Computing Width	16-bit, 32-bit & 64-bit	16-bit, 32-bit & 64-bit
Clock Speed	GHz	MHz – GHz (capable of Turbo)
Memory (RAM) in ranges	512 MB to several GB	Can range from Megabytes to Gigabytes
Time Critical/deterministic applications	No	Yes
Power Consumption	High	Varies upon usage, can be very high depending on the application
Applications	Intensive computing	Video processing, frame rendering
Image Processing	No	Yes
Physical Size	Large	Small
Product Examples according to [8]	Raspberry Pi board, Beaglebone BlackBoard	Nvidia GTX series cards, AMD cards, Intel integrated graphics cards

6. Conclusions

Based on the research conducted in this paper, FPGA, processors and GPU technology are different to one another. Although they can be used for similar applications, the way in which each of these technologies perform depend on several parameters. Some of these parameters include, cost, processing efficiency, durability, speed and much more. Depending on the application, and the resources available, one technology is preferred over the other.

Each of these technologies are also continuously growing and expanding. More research is being conducted on FPGA technology into building 3D FPGAs as well as stacking dyes to create a better performing device. As previously mentioned, companies such as XILINX and Tabula have begun experimenting with this technology.

Currently, companies that are leading the market in terms of GPU development such as Nvidia and AMD, are researching the possibilities of multi-chip GPU designs. The plan is to fit multiple GPU modules into a single chip. Nvidia has stated that using a single chip design will eventually reach a performance ceiling. According to Nvidia's GPU simulations, multi GPUs can maintain performance very close to a single chip with an enormous number of transistors (a monolithic design 256 streaming multiprocessor which isn't buildable) by using a high-speed interconnect. The simulations showed that the multi-chip module GPU (which is buildable) performed within 10% of the speed of the monolithic GPU [9].

The future of silicon processors is nearing its end. Other semi-conductors or carbon nano-tubes will have to be investigated since the layers of semi-conductors in a transistor can only become so small (currently at 11nm) before we have to consider the quantum effects of electrons (tunneling) which make a transistor erratic. Even if semi-conductors hold no hope, quantum computing is slowly gaining traction and has a bright future since the core technology has no semi-conductors involved. As it currently stands however, quantum computing is expensive and requires enormous liquid Nitrogen cooling solutions to implement [10].

References

[1] S. A. Hauck, "Multi-FPGA Systems," ACM Digital Library, Seattle, 1995.

[2] XILINX, "Highest Performance and Integration at 28nm," 2018. [Online]. [Accessed 27 March 2018].

[3] A. Amara and F. Amiel, "FPGA vs. ASIC for low power applications," Microelectronics Journal, 2006.

[4] R. Joost and R. Salomon, "Advantages of FPGA-based multiprocessor systems in industrial applications," IEEE Xplore, January 2011.

[5] D. F. Bacon, R. Rabbah and S. Shukla, "FPGA programming for the masses," ACM Digital Library, 2013.

[6] J. Papiewski, 2018. [Online]. Available: http://smallbusiness.chron.com/purpose-graphics-card-55327.html.

[7] R. Edmonds, "Everything you need to know about GPUs," 2017. [Online]. Available: https://www.windowscentral.com/everything-you-need-know-about-gpu.

[8] S. Thornton, 2017. [Online]. Available: https://www.microcontrollertips.com/microcontrollers-vs-microprocessors-whats-difference/.

[9] R. Coleman, 5 July 2017. [Online]. Available: https://www.pcgamesn.com/nvidia/nvidia-multi-chip-GPU.

[10] G. Marshall, "Beyond silicon: We discover the processors of your future tech," Techradar, 26 August 2016.

[11] BERTEN Digital Signal Processing, "GPU vs FPGA Performance Comparison," BERTERN DSP, 2016.

[12] IEEE, "Chip Hall of Fame: Xilinx XC2064 FPGA," IEEE Spectrum: Technology, Engineering, and Science News, 30 June 2017.

[13] J. Gallagher, "VERIFICATION TECHNIQUES: GOING BEYOND SIMULATION," 13 February 2006. [Online]. [Accessed 28 March 2018].

[14] XILINX, "Field Programmable Gate Array (FPGA)," 2018. [Online]. Available: https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html. [Accessed 21 March 2018].

[15] Wisconsin Parts Database Instructions and Discussion Team/Group, "Xilinx virtex-4 LX200 multi-fpga soc plateform," 2017. [Online]. [Accessed 24 March 2018].

[16] L. H. Ceze, "Shared-Memory Multiprocessors," Encyclopedia of Parallel Computing, 01 January 1970.

[17] J. Dorsch, "CPU, GPU, or FPGA?," Semiconductor Engineering, 16 June 2016.

[18] T. Fisher, "Central Processing Unit (CPU)," Lifewire, 16 January 2018.

[19] J. Martin, "PCs, Phones, Apps, Processors: We Explain the Difference between 32-bit and 64-bit," Tech Advisor, 11 April 2017.

[20] National Instruments, "Understanding Parallel Hardware: Multiprocessors, Hyperthreading, Dual-Core, Multicore and FPGAs," 06 December 2011. [Online].

Author Profile

Marcus Lloyde George received the Bsc degree in Electrical and Computer Engineering from the University of the West Indies, St. Augustine in 2007 and his MPhil degree in Electrical and Computer Engineering from the University of the West Indies, St. Augustine in 2011. Marcus is currently in the examination stage of his PhD degree in Electrical and Computer Engineering from the University of the West Indies, St. Augustine.

Marcus is Chairman, Chief Executive Officer and Founder of the Ultimate Virtual Market Limited Group of Companies which include a range of online-based service-providing companies in the areas of Agriculture, Automotive and Education. He is also the author of several books in the area of Life Foundations and is the author of upcoming books "Expert Mathematics: Strategies and Solutions" and "Digital Electronic Systems – Principles and Practices".

His research engineering interest include the business administration, strategic planning and management, engineering education, formal specification, modelling and verification, field programmable architectures, embedded systems design, intelligent electronic instrumentation, CADs for field programmable architectures, biomedical engineering, network on chip architectures, reconfigurable computing, and information and communication technology (ICT).

Shaun Bhimlal, Simeon Ramjit and **Mustafa Nakhid** received the Bsc degree in Electrical and Computer Engineering from the University of the West Indies, St. Augustine in 2018 and were former research students of Marcus Lloyde George.

[i] D, Subhash, et al. "What Is a CPU? Definition and Working [with Block Diagram]." IT4nextgen, 2 July 2017, www.it4nextgen.com/what-is-a-cpu-central-processing-unit/.

[ii] "What Is Multi-Core Processor? - Definition from WhatIs.com." SearchDataCenter, searchdatacenter.techtarget.com/definition/multi-core-processor.

[iii] Tanenbaum, Andrew S., and Herbert Bos. Modern Operating Systems. Harlow: Pearson Education, 2015. Chapter 8 Section 8.1.4

[iv] "Multiprocessor System for Realtime Robotics Applications." Microprocessors and Microsystems. February 20, 2003. Accessed April 02, 2018. https://www.sciencedirect.com/science/article/pii/014193319090120K.

[v] Habib, Aamir. "What Are the Advantages and Disadvantages of FPGAs Compared to Micro-controllers?" Research Gate. 2015. Accessed April 2, 2018. https://www.researchgate.net/post/What_are_the_advantages_and_disadvantages_of_FPGAs_compared_to_micro-controllers.

[vi] Kalnoskas, Aimee, and Ibrahim. "MCU vs FPGA - What's the Diff?" Microcontroller Tips. December 16, 2016. Accessed April 02, 2018. https://www.microcontrollertips.com/faq-mcu-vs-fpga-whats-the-diff/.