FPGAs in Client Compute Hardware

Michael Mattioli
Goldman Sachs & Co.

Fig. 1. Xilinx Spartan-3A FPGA [1].

Abstract—Field-programmable gate arrays (FPGAs) are remarkably versatile. FPGAs are used in a wide variety of applications and industries where use of application-specific integrated circuits (ASICs) is less economically feasible. Despite the area, cost, and power challenges designers face when integrating FPGAs into devices, they provide significant security and performance benefits. Many of these benefits can be realized in client compute hardware such as laptops, tablets, and smartphones.

I. INTRODUCTION

A field-programmable gate array (FPGA) is an integrated circuit (IC) composed of configurable logic blocks (CLBs) connected via programmable interconnects; it can be configured to desired application or functionality requirements after having been manufactured (hence “field-programmable”). In contrast, an application-specific integrated cir-

cuit (ASIC) (e.g. CPU, GPU, or SoC) cannot be modified or changed after manufacturing.

Hardware designers use hardware description languages (HDLs), such as VHDL or Verilog, to describe the structure and/or behavior of the logic elements within the FPGA. Using electronic design automation (EDA) tools, the design is then synthesized and the FPGA configuration, often referred to as a bitstream, is generated; finally, the bitstream is applied to the FPGA. The aforementioned explanation is a drastic oversimplification and should serve as a rudimentary explanation of FPGAs.

The world’s largest FPGA manufacturers are Xilinx, which is to be acquired by Advanced Micro Devices (AMD), and Intel (formerly Altera, which Intel had acquired in 2015). As of 2019, the FPGA market size was valued at USD 9.0 billion and is expected to reach USD 14.20 billion by 2025 [2].

Despite their versatility and heavy use in a variety of applications, FPGAs are notoriously absent from modern client compute hardware (e.g. laptops, smartphones, desktops, tablets, etc.). The purpose of this paper is to examine the challenges and benefits of using FPGAs in client compute hardware.

II. CURRENT APPLICATIONS

FPGAs are commonly used in the early stages of hardware design for rapid prototyping, testing, and development because they can be reconfigured at-will. Otherwise, designers would have to send their designs to the foundry to be fabricated every time they were updated or modified; this can be very time consuming and costly. Other common applications include those in aerospace and defense (e.g. avionics and missile defense systems), audio and video (e.g. digital signal processing, encoding, decoding, etc.), medical (e.g. ultrasound, x-ray, etc.), and various other markets and industries.

FPGAs are also commonly used in cloud and data center applications. Microsoft’s Azure SmartNIC is a network card which uses an FPGA to

accelerate network performance (lower latency and higher throughput) of the virtual machines offered through their Azure cloud service [3]. Amazon Web Services (AWS) offers virtual machines with FPGAs which developers can use to accelerate their applications [4]. For those who wish to run applications in their own data centers, FPGAs are available as Peripheral Component Interconnect Express (PCIe) add-in cards which can be integrated into new or existing servers; developers can then use them to accelerate their applications [5].

Generally speaking, FPGAs are found wherever the volume of units or devices produced is relatively small such that it is more economical to use an FPGA for the application than it is to design and fabricate an ASIC.

III. CHALLENGES

There are a variety of non-trivial challenges that arise when hardware designers look to include FPGAs in client compute devices. Area (the amount of space on the PCB that the FPGA will occupy), power consumption, and cost are just some of the hurdles that must be overcome.

A. Area

In 2008 at Macworld, Steve Jobs, the CEO of Apple at the time, unveiled the MacBook Air - a laptop computer so thin and light it could fit inside an envelope. At the time, the consumer electronics industry had already been moving toward thinner and lighter devices; it was a natural progression. However, the MacBook Air was so radically thin and light in comparison to its competition at the time that OEMs had no choice but to press the fast-forward button and prioritize size and weight in their devices in order to remain competitive. To this day, size and weight remain a priority in the eyes of both the hardware designers and the consumers of client compute devices.

In comparison to ASICs, FPGAs use more area for the equivalent amount of logic and functionality implementation. The units of measure in question are square millimeters (mm^2) which may seem negligible at first glance; however, in modern device design every micron counts.

B. Cost

From a bill of materials (BOM) perspective, FPGAs are more costly in comparison to ASICs. Depending on the target audience and market, such as the consumer electronics market, there are often tight margins and very specific price targets that must be met for the product to be economically feasible and sell at volumes worth while for the OEM or system designer. Increasing the BOM by introducing an FPGA may cause the overall target price of the product or system to be outside of an acceptable range.

C. Power

In comparison to ASICs, FPGAs consume more power [6]. In large, complex systems where power consumption is either not as much of a constraint or the increase in power from one or multiple FPGAs is negligible then this might not be much of a concern. However, in client compute devices, power consumption is highly prioritized in overall system design. Lower total cost of ownership (TCO) and increased battery life (in the context of mobile devices) are highly desirable characteristics in the consumer electronics world.

IV. BENEFITS

There are a variety of benefits which can be realized by integrating FPGAs into client compute hardware [7]. They can be used to accelerate otherwise expensive operations which, in turn, can lead to increased performance and power efficiency. Their reconfigurable nature allows for hardware which has already been deployed to be updated throughout its lifecycle (e.g. fix a security issue, improve performance, etc.). If done carefully, choosing to implement a particular function in hardware, by means of an FPGA, has the *potential* to increase the overall security of a device.

A. Hardware acceleration

Hardware acceleration and heterogeneous compute architectures are becoming more and more prevalent. In other words, "use the right tool for the job"; not all workloads are well-suited to one particular type of hardware (e.g. CPUs or GPUs).

There are several examples of this in modern compute devices:

- Apple’s M1 system on a chip (SoC) features dedicated neural network hardware which they refer to as the Neural Engine; it’s used for FaceID, Animoji, and other machine learning workloads/tasks. While these tasks could be processed on the CPU or the GPU, there are significant power-efficiency and performance benefits of using dedicated hardware.
- ARM’s big.LITTLE architecture combines two different types of CPU cores: those designed for higher performance and those designed for power efficiency; certain tasks and workloads are more suited to different power and performance profiles.
- The Advanced Encryption Standard New Instructions (AES-NI) accelerate AES encryption and decryption operations. Given how frequently these operations are performed, accelerating them in hardware brings performance and power efficiency benefits.

FPGAs can be used to accelerate specific workloads in situations where such workloads are significant enough to realize performance and/or power benefits. OEMs and system designers may favor an FPGA over an ASIC for a variety of reasons:

- As previously mentioned, it is not economically feasible to design and manufacture an ASIC.
- No such hardware exists in the market to accelerate such workload.
- The desired integration or functionality of the accelerator does not exist using hardware available in the market.

B. Patching and updates

As previously discussed, a major benefit of using an FPGA is its ability to be reconfigured once deployed. In practice, this means that hardware can be modified or updated over time. This benefits both the designers (e.g. less manufacturing overhead) and the consumers (e.g. no confusion over which product to buy or product obsolescence). For example, if accelerating the encode or decode of a particular audio or video codec, the implementation may change or be updated over time. Another major benefit is the ability to fix or patch security

Fig. 2. High-level depiction of the Xbox One X SoC architecture [9].

vulnerabilities which are discovered over time [8]. ASICs suffer from the inability to be modified after being manufactured; Spectre and Meltdown, and variations of each, are glaring examples of how crippling a hardware vulnerability can be. The ability to patch hardware cannot be overstated.

C. Enhanced security

Before diving into this section I want to make two things crystal clear:

- 1) **There is no such thing as a perfectly secure system.** Philosophically, we, as human beings, are imperfect and the systems we design and use are, inherently imperfect.
- 2) **Hardware is not inherently more secure than software.** If something is implemented in hardware as opposed to software or firmware that does not magically make it more secure.

Despite these two sobering points, implementing a particular function in hardware *can* improve the overall security of a system.

- The SoC used in the Microsoft Xbox One video game console, illustrated in Figure 2, has a dedicated hardware pin used to access secret keys for decryption of sensitive data. This ensures that the keys are never exposed to software or any other part of the system at any point in time.

Fig. 3. High-level depiction of the storage architecture on Apple devices which use the T2 SoC [10].

- Apple's storage architecture in systems which use the T2 SoC has both security and performance benefits in comparison to devices which use more conventional storage architectures. As illustrated in Figure 3, the SoC sits in the direct memory access (DMA) path between the CPU and the non-volatile storage. This keeps secret keys used to encrypt and decrypt data out of the hands of the rest of the system (e.g. the CPU and any software running on the CPU). An additional benefit of this architecture is enhanced performance; the T2 SoC acts as the storage controller which performs all reads and writes to the NAND flash modules.
- Biometric authentication, means by which one's body measurements such as fingerprint or face are used to authenticate, is becoming more and more prevalent because it is stronger in comparison to password authentication. When processing one's biometric information, there are significant privacy and security concerns that arise such as how that information is being stored, processed, and handled. Early biometric authentication mechanisms processed an individual's biometric information directly on the host device (i.e. in software on the CPU); this means that one's biometric information is now exposed to software and various components within the system. Modern biometric authentication mechanisms, such as the Synaptics FS7600 whose architecture is illustrated in Figure 4, store, process, and handle all biometric information on a single SoC; this means that one's biometric information is never exposed to the host operating system, CPU, or any other hardware or software component of the

Fig. 4. High-level depiction of the Synaptics FS7600 biometric authentication SoC [11].

Fig. 5. High-level depiction of an example storage device architecture used in a client compute device.

system.

As highlighted, choosing to implement certain functions of a system's architecture can yield security benefits. FPGAs offer the same potential benefits as an ASIC would with the added benefit of being reconfigured throughout the system's lifecycle. If security is a priority in system design (which it should be), then being prepared for when security vulnerabilities are found and being able to remediate them (perhaps by means of an update to the FPGA) should also be a priority.

V. PUTTING IT ALL TOGETHER

This section focuses on a scenario in which an FPGA is used instead of an ASIC to demonstrate the practicality of integrating programmable logic into client compute hardware designs. In this scenario, a solid-state storage device (SSD) inside a client compute device uses an FPGA to implement the functions of the storage controller; a high level depiction of the SSD's architecture is illustrated in Figure 5. The FPGA will be used to implement:

- Crypto Engine - Performs all encryption/decryption operations on data that is read/written.
- DRAM Controller - Interfaces with the DRAM (typically used as a cache).
- NAND Controller - Interfaces with the NAND flash modules within the storage device.

Fig. 6. Shape/layout of the late 2020 13-inch MacBook Pro PCB which indicates the placement of the NAND flash modules and the Apple M1 SoC.

- Processor - Core logic of the storage device.
- Host Interface Controller - Interfaces with the host (e.g. laptop, desktop, smartphone, tablet, etc.) via NVMe (or SATA, SAS, etc.).

It's important to note that many of the points being made in this section would apply if an FPGA was used in other ways (e.g. audio/video encode/decode, facial recognition, etc.).

A. Area

A standard M.2 2280 SSD measures 22mm x 80mm; a total of 1,760 mm². This form-factor is a single, rigid piece of hardware. While an FPGA required to implement the logic described in this hypothetical SSD would be larger than most storage controller ASICs found on commercial SSDs, it would be advantageous to the designer to break apart the components. Rather than cram everything onto a single M.2 2280 form-factor, why not take advantage of the space of the PCB and spread the components out?

As illustrated in Figure 6, the late 2020 13-inch MacBook Pro uses an irregular shaped PCB to take advantage of every square millimeter available in the chassis. Components are placed throughout rather than using larger, rigid components such as M.2 SSDs. As illustrated in Figure 7, an M.2 SSD would take up a considerable amount of space on the PCB and make it more challenging to fit other components. By breaking the components apart and spreading them across the PCB there is better use of space. This is a fair compromise to ensure that size/space requirements are met.

Fig. 7. Example layout of the components of the SSD distributed throughout the PCB rather than lumped together in a M.2 2280 form-factor.

B. Power and Performance

An Intel Optane 905p SSD consumes 9.35W active and 2.52W idle; it has a sequential read and write bandwidth of 2,600 MB/s and 2,200 MB/s, respectively [12]. Ruan et al. have demonstrated a SSD design similar to the one proposed in this section (an FPGA-accelerated SSD) consumes around 10W active and realizes average 12x performance increase in comparison to an SSD which uses a quad-core ARM CPU as the storage controller [13]. While there is an increase in power consumed, there is a significant increase in performance.

C. IP theft

By reverse engineering an IC, one can learn its structure, design, and functionality; a common protection against this is IC camouflaging, however, this brings with it significant area, power, and delay overhead [14]. TechInsights offers professional services to reverse engineer ICs [15] and Degate even offers software products which can be used to perform reverse engineering of ICs [16]. Reverse engineering is commonly performed for a variety of reasons; however, a malicious entity could reverse engineer an IC for the purpose of stealing/pirating the design. When using an FPGA, a malicious entity attempting to reverse engineer the FPGA IC would only learn of the FPGA itself and not the logic or IP implemented.

While FPGAs may offer protection from IC reverse engineering, it is worth noting that they are potentially exposed to other malicious attacks and/or piracy through manipulation and/or reverse engineering of the bitstream. The bitstream can be manipulated such that when it is loaded into the FPGA it causes malicious or unwanted behavior;

it can also potentially be extracted and reverse engineered. While this may seem alarming, there are both commercial [17] [18] and academic [19] methods for protecting the bitstream available today.

D. Cost

For the end-users of the compute device, their overall TCO is reduced. Throughout the lifecycle of the device, as implementations of the IP blocks in the SSD improve (e.g. efficiency, security, etc.) and the FPGA configuration is updated, the need for a "new" device decreases over time (why buy a new device when the one you have can be "good as new"?). Capital expenses are reduced because fewer devices need to be purchased; this also lowers the overall carbon footprint for the end-users (less e-waste). Operational expenses are reduced because, in the event of an inevitable security-related issue, the remedy is a simply update to the SSD which, presumably, bears no performance penalty. The only issue, realistically, is wear leveling in the NAND flash modules inside the SSD; however, realistically, this would only be an issue if there was very extensive use of the storage device such as exceeding the SSD's drive writes per day (DWPD).

For the designers of the compute device, as with any engineering project, the overall cost is more than just the BOM; non-recurring engineering (NRE) costs are often very significant in major projects. There is also the continued development and support costs that come with revising hardware designs and providing iterative product updates. While there will likely be an increase in BOM cost (as opposed to using an ASIC), using an FPGA to implement the SSD controller may offer reduced ongoing development costs because the functionality of the controller may be updated over time thereby reducing the need to design a new ASIC and have the product go through another round regulatory certifications. System designers need only "push" an update to the SSD (e.g. via the internet or some other means) to update or change its functionality. It would be worthwhile for OEMs and system designers to consider balancing the reduced ongoing development costs when using FPGAs and the increase in BOM cost. In other words, it may be more feasible to "absorb" some

of the increased BOM cost that otherwise would've been passed onto the end-users because the designers' overall costs likely would have been reduced.

VI. CONCLUSION

We've discussed the challenges and benefits of using FPGAs in client compute hardware. While the challenges cannot be discounted, the benefits strongly outweigh the work and effort required to successfully integrate FPGAs into client compute hardware. There are very few yet notable examples of FPGAs being used in client compute hardware already:

- The Samsung Galaxy S5, a smartphone released in 2014, featured a Lattice Semiconductor LPIK9D FPGA [20].
- The Apple iPhone 7, a smartphone released in 2016, featured a Lattice Semiconductor ICE5LP4K FPGA [21].
- The current-generation Apple Mac Pro, a desktop computer released in 2019, can be configured with an Apple Afterburner accelerator card which uses an FPGA to accelerate the decode and playback of ProRes and ProRes RAW video files [22].

These are products which are not only sold by well-known and reputable OEMs but products which have been received very well by their target audiences and markets.

Interestingly, AMD recently applied for a patent which integrates programmable logic into a CPU [23]. The integration of programmable logic into other types of hardware (i.e. not just as another component in the system but as a part of the component itself) opens the door for more types of hardware designs. For example, if an independent software vendor (ISV) application uses a particular operation that is computationally expensive it would make sense to accelerate it in hardware for both power efficiency and performance benefits. However, as a hardware designer, it is impractical to account for all ISV applications. By integrating programmable logic, or "programmable" execution units, into the CPU of a client compute device, the ISV can bundle the information required to configure those programmable execution units (i.e. the bitstream) such that the CPU uses them to accelerate those expensive operations when that

Fig. 8. High-level depiction of programmable execution units integrated into a CPU architecture [23].

application is in use. A high level depiction of this is illustrated in Figure 8.

In the end, how do things like hardware acceleration, ease of product updates, and enhanced security translate to the consumers of client compute hardware? Very simply: better overall experience when using the product and lower overall TCO. These are characteristics that all client compute devices are designed for (no one wants something that's difficult to use and expensive) so it should be no surprise that the aforementioned products have been well-received and it should serve as an indicator that, if done properly, products which integrate FPGAs can be successful.

REFERENCES

- [1] W. Commons, "File:xerox_colorqube_8570_main_controller - xilinx_spartan_xc3s400a-0205.jpg," https://commons.wikimedia.org/wiki/File:Xerox_ColorQube_8570_-_Main_controller_-_Xilinx_Spartan_XC3S400A-0205.jpg, Aug 2019, accessed: 2021-01-07.
- [2] G. Research, "Field programmable gate array market size report, 2020-2027," <https://www.grandviewresearch.com/industry-analysis/fpga-market>, Apr 2020, accessed: 2021-01-07.
- [3] D. Firestone, A. Putnam, S. Mundkur, D. Chiou, A. Dabagh, M. Andrewartha, H. Angepat, V. Bhanu, A. Caulfield, E. Chung, H. K. Chandrappa, S. Chaturmohta, M. Humphrey, J. Lavier, N. Lam, F. Liu, K. Ovtcharov, J. Padhye, G. Popuri, S. Raindel, T. Sapre, M. Shaw, G. Silva, M. Sivakumar, N. Srivastava, A. Verma, Q. Zuhair, D. Bansal, D. Burger, K. Vaid, D. A. Maltz, and A. Greenberg, "Azure accelerated networking: Smartnics in the public cloud," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. Renton, WA: USENIX Association, Apr. 2018, pp. 51–66. [Online]. Available: <https://www.usenix.org/conference/nsdi18/presentation/firestone>
- [4] X. Wang, Y. Niu, F. Liu, and Z. Xu, "When fpga meets cloud: A first look at performance," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2020.
- [5] R. Venkatakrishnan, A. Misra, and V. Kindratenko, "High-level synthesis-based approach for accelerating scientific codes on fpgas," *Computing in Science Engineering*, vol. 22, no. 4, pp. 104–109, 2020.
- [6] I. Kuon and J. Rose, "Measuring the gap between fpgas and asics," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 2, pp. 203–215, 2007.
- [7] R. Tessier, K. Pocek, and A. DeHon, "Reconfigurable computing architectures," *Proceedings of the IEEE*, vol. 103, no. 3, pp. 332–354, 2015.
- [8] G. Dessouky, T. Frassetto, P. Jauernig, A. R. Sadeghi, and E. Stappf, "With great complexity comes great vulnerability: From stand-alone fixes to reconfigurable security," *IEEE Security Privacy*, vol. 18, no. 5, pp. 57–66, 2020.
- [9] M. Mattioli and A. Lahtiranta, "The hidden potential within video game consoles," *IEEE Micro*, pp. 1–1, 2021.
- [10] A. Support, "Dedicated aes engine," <https://support.apple.com/guide/security/dedicated-aes-engine-sec4ea70a303/1/web/1>, Mar 2020, accessed: 2021-01-17.
- [11] A. Shilov, "Architecture and analysis - synaptics' next-gen fingerprint sensor security: The fs7600 match-in-sensor," <https://www.anandtech.com/show/13077/synaptics-discusses-fs7600-match-in-sensor-fingerprint-sensor/2>, Aug 2018, accessed: 2021-01-18.
- [12] Intel Optane SSD 905P Series Product Specifications, <https://ark.intel.com/content/www/us/en/ark/products/148607/intel-optane-ssd-905p-series-380gb-m-2-110mm-pcie-x4-20nm-3d-xpoint.html>, Intel ARK, June 2018.
- [13] Z. Ruan, T. He, and J. Cong, "Insider: Designing in-storage computing system for emerging high-performance drive," in *Proceedings of the 2019 USENIX Conference on Usenix Annual Technical Conference*, ser. USENIX ATC '19. USA: USENIX Association, 2019, p. 379–394.
- [14] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri, "Security analysis of integrated circuit camouflaging," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, ser. CCS '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 709–720. [Online]. Available: <https://doi.org/10.1145/2508859.2516656>
- [15] TechInsights, "Scope of analysis — techinsights," <https://www.techinsights.com/technical-capabilities/overview/scope-of-analysis>, accessed: 2021-01-30.
- [16] Degate, "Reverse engineering integrated circuits with degate - documentation," <https://degate.org/documentation/>, accessed: 2021-01-30.
- [17] Xilinx, "Ar# 40360: Fpga - what are the methods to protect the fpga bitstream against unauthorized duplication?" <https://www.xilinx.com/support/answers/40360.html>, accessed: 2021-01-31.

- [18] *FPGA IFF Copy Protection Using Dallas Semiconductor/Maxim DS2432 Secure EEPROMs*, https://www.xilinx.com/support/documentation/application_notes/xapp780.pdf, Xilinx, May 2010.
- [19] R. Karam, T. Hoque, S. Ray, M. Tehranipoor, and S. Bhunia, "Robust bitstream protection in fpga-based systems through low-overhead obfuscation," in *2016 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, 2016, pp. 1–8.
- [20] M. Alarcon, R. Fontaine, D. James, R. Krishnamurthy, J. Morrison, D. Yang, and C. Young, "Samsung galaxy s5 teardown," <https://www.techinsights.com/blog/samsung-galaxy-s5-teardown>, Apr 2014, accessed: 2021-01-18.
- [21] S. Wegner, A. Cowsky, C. Davis, D. James, D. Yang, R. Fontaine, and J. Morrison, "Apple iphone 7 teardown," <https://www.techinsights.com/blog/apple-iphone-7-teardown>, Sep 2016, accessed: 2021-01-18.
- [22] A. Support, "About the afterburner accelerator card for mac pro (2019)," <https://support.apple.com/en-us/HT210748>, Nov 2020, accessed: 2021-01-17.
- [23] A. G. Kegel, "Method and apparatus for efficient programmable instructions in computer systems," U.S. Patent Application US20 200 409 707A1, Jun 25, 2019.

Michael Mattioli leads the Hardware Engineering team within Goldman Sachs. He is responsible for the design and engineering of the firm's digital experiences and technologies. He is also responsible for the overall strategy and execution of hardware innovation both within the firm and within the broader technology industry. Contact him at michael.mattioli@gs.com.