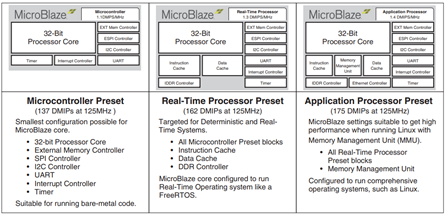
Ryan Melendez, Burhan Alestwani, Anas Salah Eddin, Mohamed El-Hadedy (Aly), Members, IEEE

[[1]](#footnote-1)

Low-Power High-Performance Reconfigurable Computing using FPGA (LPHP-RC)

*Abstract*—As the demand for computational power increases, the need to develop cost efficient methods for computing is constantly rising. To help solve this issue, our research is dedicated towards developing low power computational technology while being a cost-efficient solution compared to other resources. The project is aimed towards using Field Programmable Gate Array (FPGA) platforms that can be reconfigured to run applications that can be used in image processing, cyber security, encryption, and high-performance applications for Defense. The objective is to run an application on the FPGA board using a softcore processor (Microblaze) that will be able to efficiently run the C code. Once this objective is satisfied, our next goal is to implement the maximum units of soft-core processors working with each other efficiently on a single FPGA chip while using the minimum amount of resources. The configuration will be performed using master-slave format, which the load will be distributed across multiple cores utilized by the board. Our initial testing was performed using Discrete Fourier Transform (DFT). This was used to make sure that the Microblaze is fully functioning and exporting results. The second test we have performed was aimed towards benchmarking using various encryptions. We have used multiple encryptions on the soft core processor to generate the encryption keys. The next objective is to design communication interfaces to connect multiple low-power FPGA platforms for rapid scalability. This allows the FPGA boards to be used at large scale projects and support the high computational demand for the power-hungry applications.

*Index Terms*—FPGA, Microblaze, RSA, AES, ASCON128

# INTRODUCTION

T

HE MicroBlaze is a soft core processor created and developed by Xilinx for Field Programmable Gate Array boards. It is a reconfigurable microprocessor that has a 32-bit RISC architecture. The software development kit allows you to design your own specifications for the Microblaze with no prior experience in Verilog coding. Although FPGA boards have not been used widely for reliability reasons, real time optimization for applications allow FPGA boards to have a reduced latency by 20% on average [1]. According to Xilinx, the softcore processor will meet all the applications requirements for many fields such as the medical or defense industries. Moreover, FPGA boards in the past few years have been rapidly developed to run multiple processing cores that can run in parallel to distribute the load over all platforms [2].

One of the key features of the Microblaze is its five-stage pipelining, executing multiple processes consecutively and efficiently. On slower memories the instruction fetch may take multiple cycles and thus add latency which directly affects the pipeline. In order to counteract this, the Microblaze uses an instruction prefetch buffer which helps reduce the impact of such multi-cycle instruction memory latency. While the pipeline is stalled by a multi-cycle instruction in the execution stage, the prefetch buffer continues to load sequential instructions. When the pipeline resumes execution, the fetch stage can load new instructions directly from the prefetch buffer instead of waiting for the instruction memory access to complete [12].

Figure 1: Multiple base configurations of the MicroBlaze platform

Since the MicroBlaze’s architecture allows it to be reconfigurable, various settings and resources can be altered depending on the desired application of the FPGA board. Certain algorithms can be developed specifically to target the board so it can run efficiently on Microblaze [3]. Some presets already exist in the MicroBlaze configuration; however, it is recommended to adjust these settings according to the project outcomes. For small applications that do not require high computational power, a microcontroller preset can be used with the minimum amount of resources to perform the task. As for high demand utilization, an application processor preset is used to include all the required block with a memory management unit. This allows the MicroBlaze to host operating systems that require hardware-based paging and protection.

The only factor that limits FPGA boards is the total number of available resources and the clock rate of each specific board that it can generate. This allows any FPGA board running a MicroBlaze processor a substantial, application based, platform ready for the demands of such programs. Communication between the FPGA and the Microblaze takes place via the system’s shared data memory [7]. Reconfigurable processors are viable for redundant systems like small scale satellites or computational systems that need to be frequently updated or maintained. This can be entirely accomplished remotely by uploading a code file to the board instead of the necessity to be physically present for the appropriate adjustments. These boards can be designed to be more efficient by fully exploiting the combination of the software and hardware as one unit [6]. For designs that require heavy computational power, a multi-core MicroBlaze platform will provide the solution.

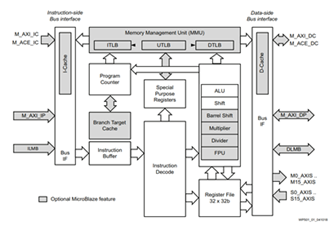


Figure 2: Microblaze Block Diagram

When implementing the MicroBlaze processor on an FPGA, the total amount of data and instruction memory can be adjusted to fit a specific program’s needs, as well as an option to design the processor with an emphasis on either area, frequency or performance. One of the most common uses of a MicroBlaze is as a co-processor to an ARM A9. The limit to the total amount of co-processors is based on the total amount of resources available on the FPGA chip itself.

To discuss the various encryptions, we have used for benchmarking, we need to introduce the main encryptions utilized in this paper and the way each encryption work. The two main encryptions are mainly RSA (Rivest–Shamir–Adleman) encryption and AES (Advanced Encryption Standard) encryption. Both algorithms have been widely used in the past in various applications. RSA encryption is generally used in different security services for secure data transmission and in digital signatures. However, since RSA is generally slow when encrypting a lot of data, it is mainly used alongside other encryptions such as AES to boost the speed of the encryption. Moreover, AES encryption itself became so successful that the US government after being accepted as the federal government standard for encryption. As a result, we decided to use both widely used encryptions on the softcore processors as a method to benchmark our data.

# Related Work

Another significant feature of the MicroBlaze is its ability to interface with an ARM, advanced RISC machine, that may be provided on a FPGA board. ARM processors have become very popular within the last few years because of their low power consumption and high performance when compared to a more conventional x86 chipset. For most benchmarks the ARM platforms are head to head with the Sandy Bridge i5 processor, except for the x264 and canneal benchmarks [13]. By interfacing multiple soft-core processors with the single ARM processor, we can increase our performance metrics through dividing the computational workload across various platforms. For application purposes, a Linux system was split across multiple cores, separating the Linux kernel from the API into two different units to work simultaneously together [6].  In addition, when comparing with the ARM Cortex processors, their single-threaded energy consumption is very close to that of the i5, but the extra cost of enabling the NEON units (15–20%) puts them at a disadvantage to the i5 for the Blackscholes benchmark, showing again that using SSE and AVX seems to have no impact on power dissipation for the Intel processors. Moreover, due to the limits on bandwidth and latency of the memory subsystem on the Cortex boards, the Streamcluster benchmark suffocates when NEON units are active, increasing the overall benchmark runtime [15]. When comparing the power consumed by the system and overall performance, the relationship between both of these parameters will always be inverse. There is a trade-off between the partly conflicting goals of high performance and low energy consumption. Comparing systems based on energy consumption alone can motivate the use of simple cores with low frequency. The *Energy-Delay Product* (EDP) puts greater emphasis on performance and corresponds to the reciprocal of performance energy unit [14].

In comparison to the ASIC chip, FPGA boards have the advantage of reconfigurability and fast adaptation to any changes. It was shown that FPGA accelerators are able to provide better energy utilization than CPUs and GPUS [4]. In some cases, two hardware coprocessors can be even used simultaneously to increase performance. These implementations are used in a wide variety of applications such as cryptography to accelerate the algorithms [9]. There have been specific scenarios in which an FPGA board can tremendously outperform a GPU. Network processing approaches using private and shared ques on FPGA boards can provide a throughput that is 7 times greater than the throughput of a GPU if the right algorithms are being implemented [10]. Multicore FPGA processors have been used before in several different applications. One the more documented projects was an implementation of a emotional-based agent architecture by the Universitat Politécnica de Valencia, Spain. Regarding the implementation on a Multicore, the robotic agent architecture including the belief, behavior, attention and emotional sub-systems is implemented as a partitioned system on a six-Core processor; the Intel Core i7-980X at 3.33 GHz per core. The i7 based computer has 8 MB cache of memory, 12GB DDR3 RAM. One of the cores is dedicated to the attention system, a second core is used for the behavior-belief systems and the remaining 4 cores implement the emotional system. In the Single-Core mode, all the emotional processes are assigned to one processor. In the Dual- and Quad-Core modes, the processes are distributed evenly among the different cores [11]. The multi based emotional system’s design could be a great model for dividing our computational workloads across multiple platforms.

Encryption algorithms must undergo a daunting series of tests and analysis before they can be deemed viable. Each year NIST hosts the lightweight cryptography competition where they challenge scholars to improve on the existing standards and/or showcase new architectures. As these new encryption submissions are released, there are various platforms and applications that they can be tested with. Noticing the lack of encryption as a form of benchmarking we look to be one of the first groups to implement and test these various AES permutations on a soft-core reconfigurable platform.

# Architecture/Math

Since we have used multiple encryptions to benchmark our platform, the main architecture used will be the math behind encrypting and decrypting the data processed through our softcore processor. The first encryption that we used for testing was the RSA encryption. To begin with, the encryption mainly consists of an asymmetric cryptographic algorithm to decode messages. In other words, the encryption will have two different keys utilized in the algorithm. The first key will be a public key that can be shared with anyone. This key is mainly used for encrypting the message, so anyone has the ability to encrypt the data that they desire. However, the second key will be a private key that only that key holder has access to. This key will be used for decrypting the messages, which only permitted users are allowed to read the encrypted messages. The encryption generally rotates around prime factorization of very large numbers. To be able to generate keys, the user will pick two large prime numbers to represent p and q, which are then multiplied to provide a modulus of n. The algorithm follows a trapdoor function method, which allows it to be simple in one direction but almost irreversible when trying to go backwards. After this, carmichael’s totient function is used to represent the lowest prime number that both “p” and “q” can be divided into. A prime number “e” is then picked to generate the public key and be used by the public. There are many more details regarding “e” that we will not discuss in this paper since this is not the focus of the project. As for the private key, it will be generated by “e” in conjunction with “d”, which “d” is a private number that only the key holder should know. This number should not be shared or else it will compromise the encryption.

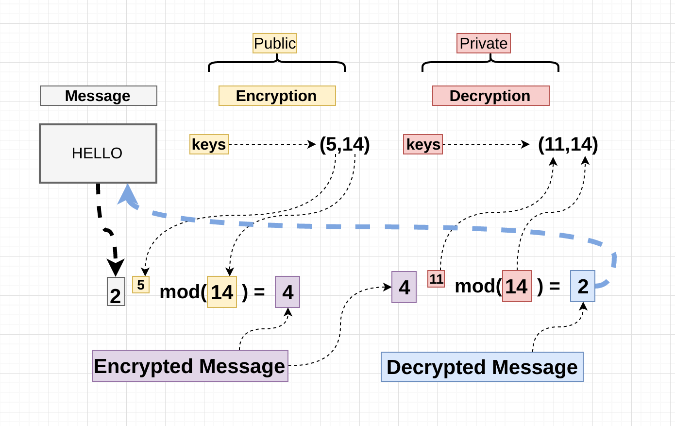


Figure 3: Logic with RSA encryption

The second encryption that we used on our platform was Advanced Encryption Standard, or AES for short. When compared to RSA, this encryption is actually a symmetric cryptographic algorithm that is used when decoding messages. In this case, the same generated key will be used for encrypting and decrypting the data at the same time. As a result, this key should be kept private since the inverse method will use the same key when decrypting the message. The entire AES encryption mainly uses a principle known as a substitution permutation network. This principle is very efficient in hardware and software, which will be sufficient for security while also being efficient at the same time. The encryption was adopted by the US government with a fixed 128 bits block size, however, they key size can range either 128, 192, or 256 bits. The level of key will correspond to the level of encryption security. As a result, the 128 bit will consist of 10 rounds, the 192 bit consists of 12 rounds, and the 256 bit will have 14 rounds for maximum security. To begin with, the message is divided into blocks of 16 bytes, making it a 128 bit block. After that, the message block is added to the specified key used by the user. The next stage contains byte substitution, which consists of a predetermined table that will replace the 16 byte in the block. This is to prevent the encryption from being linear, preventing others from decrypting the message using any relationship. The third step is mainly shifting rows in which the second row is shifted one to the left, the third row is shifted two to the left, and so on. This is only done to add an element of confusion to the algorithm. The last stage consists of mixing column, which the column of each block will undergo specific mathematical equations to diffuse the encryption even further. All of these stages were used to determine round one, the rest of the rounds will consist of the same stages using a different rounded key that originates from the first key used. After the encryption goes through all the rounds, the message is fully encrypted. The decryption method undergoes the same procedures, but it will be the inverse stages for each round.



Figure 4: Logic behind AES encryption

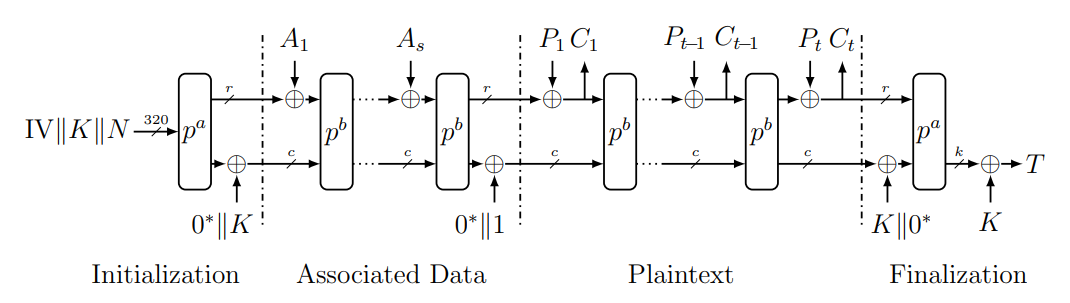
The third algorithm that was tested was ASCON128. ASCON 128 is another type of encryption created in the Lightweight Cryptography (LWC) that will use a lightweight authenticated algorithm. This was in response to the annual CAESAR competition. The inputs for the encryption procedure consists of a plaintext that is described later in the algorithm as “P”, with specific data described as “A”. It will also consist of a key known as “K” that will be kept secret from the public which uses k bits. Finally, it will have a public message number used as “N” in the algorithm with k bits. The final output of the encryption after going through all the procedures is an authenticated ciphertext that we will see in the algorithm as “C”, which is exactly the same length as P. The algorithm will also have an authentication tag known as T of size k bits, which authenticates both A and P:

Figure 5: Logic behind ASCON 128 encryption

The decryption is another process that can be complicated to explain, so we will provide a short description of the terms used. To begin with, the verification procedure will use a bunch of variables named as Da, b, k, and r. It will also use them with the input key K, N, the data A, ciphertext C and tag T, and will provide the output with the plaintext P if the verification was performed correctly using the tag T [21]. In order to ensure that our testing methods were unique, we decided to implement this design on the ARM Cortex A9 processor found on the PYNQ-Z1 FPGA board and compare our results to several higher end desktop and mobile processors. We have included all the benchmarks later in this paper which compared the algorithm running on an AMD and Intel hard core processor. Although the results had a huge gap when comparing the ARM processor to the computer processors, we noticed that the gap is not very noticeable when running low load applications. This algorithm can be implemented for various applications on different platforms, but having it running on an FPGA board will provide the extra mobility and low power cost when compared to other devices.

# Evaluation

When it comes to benchmarking a multicore processor its important to determine whether or not you are looking the traditional single-core performance or multicore performance. The historic measure of microprocessor capability is the single-thread performance of a traditional core. Many researchers have observed that single-thread performance has already leveled off, with only modest increases expected in the coming decades. Multiple cores and customization will be the major drivers for future microprocessor performance (total chip performance) [19].  Although having multiple cores can increase computational throughput by a factor of 4, if the operating system is not created with parallelism in a quad-core system, the system has no benefits when compared to a single processing chip. In general computer’s parallel design, the algorithm is divided into some parts which can be parallel executed in different threads in a single core. The different threads will be sharing a main thread, which is serial execution in essence. Although a computer can have multi-core processors, the number of cores is limited [16]. In addition to parallelism in the software aspect of the design, it is important to optimize the ISA, instruction set architecture of the system. To optimize processor performance for a particular application, a common approach is to extend the instruction set by application specific instructions. Adapting an instruction set to a particular problem is a difficult task, as many unknown issues have to be explored [17].

One key drawback for using an FPGA chip as a multicore processing chip is dissipating the heat produced by each core. As we saw, the speed of microprocessors was increased by both architectural innovations and increasing the clock speed. Whereas the number of transistors increased exponentially leading to architectural innovations, the increase of clock speed was gradual and has remained constant at around 3 GHz since 2006. This has been mainly due to heat dissipation in processors when the clock speed increased. There are three components of heat dissipation in processors when transistor switches toggle. They are: 1. The capacitance associated with a transistor when it charges and discharges when it switches state. 2. The leakage current increases as the transistor becomes smaller. 3. The short circuit current which leads to a power dissipation [18].

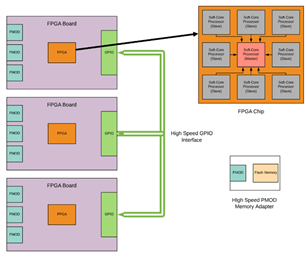


Figure 6: MicroBlaze Multiplatform configuration

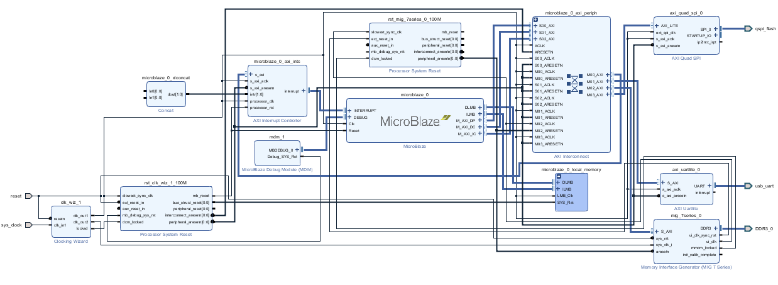
The following diagram illustrates how multiple FPGA boards are connected in parallel to distribute the computational load across them. Each FPGA board will be connected to the other boards using GPIO ports. This can be done either by implementing ethernet connections across all boards, or by using the GPIO pin connections while executing our own transfer protocol. Using these cluster-based FPGA boards will provide various applications with better compilation time and performance [5]. For the individual FPGA board, multiple softcore processors (MicroBlaze) will be optimized using the same resources.  As seen in Figure 1, the softcore processors will have a master/slave configuration to handle the communication protocol. Each board will have a master MicroBlaze, similar to a memory management unit, in control of multiple slave Micro-Blazes to share the load more efficiently. When choosing the data width of the network, careful consideration should be made to match the data width to the width of hardware arithmetic units on the target FPGA. Wider data widths than the width of hardware arithmetic units necessitate additional sets of those units to implement the design, thereby significantly increasing the final hardware requirements of the implementation [20].

Figure 7: Block diagram for RSA and AES encryptions

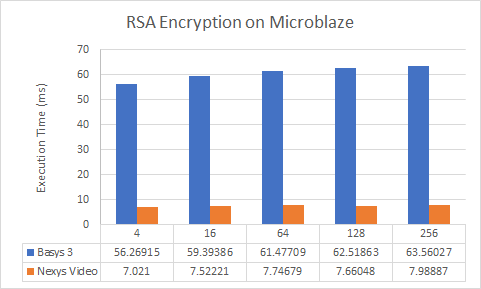
The RSA encryption algorithm that was used in this project is included in our references and was implemented on the Microblaze platform. In order to demonstrate that each Microblaze instance was properly measuring the peak performance available on a given board, each instance was implemented with the highest settings available. The figure below represents our results:

Figure 8: RSA Encryption results

The execution time of each implementation was calculated in milliseconds for ease of illustration. The Nexys Video implementation of RSA encryption significantly outperformed the Basys 3 implementation because of the increase in local memory speeds and sizes. Since RSA encryption implementation depends on a processors ability to receive and interpret a given cipher text and decrypting it.

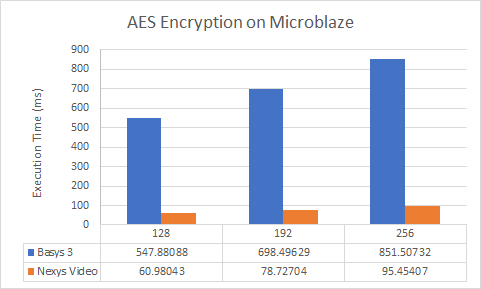
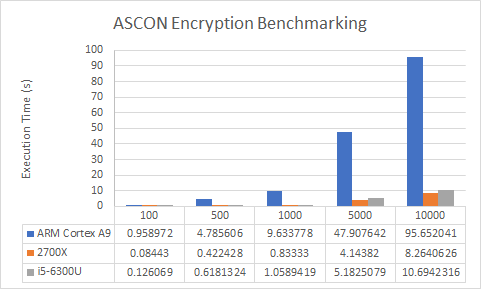
The AES encryption algorithm that was used for benchmarking is modified version of the tiny-AES implementation []. This algorithm was capable of testing AES-128. 192, 256-bit formats while maintaining a relatively small program size. Each microblaze instance was created in a manner similar to that of the RSA implementation. The figure below represents our results:

Figure 9: AES 128, 192, 256bit implementation results

As expected, AES encryption on the Nexys Video board resulted in a smaller amount of time as compared to the Basys 3 due to the increase in memory size and speed. Both of these implementations demonstrate the Microblaze’s capability of running various versions of common encryption techniques that can be used for various applications.

The ASCON128 encryption algorithm was implemented on the ARM Cortex A9 processor on the PYNQ Z1 FPGA board. The ARM A9 is a dual core 32bit processor with 512kB of L2 cache and 32kB of L1 cache for instructions and processor respectively. In order to accurately compare the performance of this processor we used both a desktop and mobile processor from both Intel (i5-6300U Mobile) and AMD (Ryzen 7 2700X Desktop) at base speeds. The figure below represents the data taken at several intervals of data. Each interval of data represented a single message string encrypted and decrypted successfully.

Figure 10: ASCON128 Encryption/Decryption results

To maintain a consistent testing platform, each implementation was compiled using the same python system version and executed from terminal. The results above do show a drastic decrease in execution time across each platform as expected.

# Conclusion

The goal of this research was aimed towards developing a reconfigurable computing platform to provide cost efficient methods. Although the computational processing speed is still far from using a conventional hardcore processor, the ability to run applications using a very low power consumption platform can be extremely useful in specific applications. We were able to successfully utilize a softcore processor on an FPGA board to run our benchmarks using C code. Although we did not reach the projected target of running multiple Microblaze at the same time, we were able to record a high ratio of processing a code when compared to the amount of power the FPGA board consumed. We have used a Basys3 board and a Nexus Video board for providing real time processing for recently developed encryption methods. Both RSA and AES encryption were implemented on both boards for comparison. The Nexus Video did give us improved and faster processing time due to the increase in memory size on the board.

We also used an ARM cortex A9 chip for running the ASCON encryption, which was compared to two of the top tier hardcore processors. The time it took to process the encryption was ten times more than the hardcore processor, nonetheless, the power consumed is way less when it comes to running an ARM processor. Our original product is still underdevelopment due to licensing issues encountered when trying to modify the Zynq 7020 SoC FPGA chip; with the amount of documentation and data we collected we can conclude that FPGA processing is a viable option for application specific tasks while maintaining a low power consumption.

# References

[1] Sanaullah, A., Yang, C., Alexeev, Y., Yoshii, K., & Herbordt, M. C. (2018). Real-time data analysis for medical diagnosis using FPGA-accelerated neural networks. BMC Bioinformatics, 19(18), 19–31.<https://doi-org.proxy.library.cpp.edu/10.1186/s12859-018-2505-7>

 [2] Dewey, M. (2019). Best Practices for User-Programmable Fpga Instrumentation. EE: Evaluation Engineering, 58(1), 18–21. Retrieved from<http://proxy.library.cpp.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,uid&db=aph&AN=134140652&site=ehost-live&scope=site>

[3] LONE, M. R., & HAKIM, N.-D. (2018). FPGA implementation of a low-power and area-efficient state-table-based compression algorithm for DSLR cameras. Turkish Journal of Electrical Engineering & Computer Sciences, 26(6), 2927–2942.<https://doi-org.proxy.library.cpp.edu/10.3906/elk-1804-208>

[4] AYAT, S. O., KHALIL-HANI, M., & AB RAHMAN, A. A.-H. (2018). Optimizing FPGA-based CNN accelerator for energy efficiency with an extended Roofline model. Turkish Journal of Electrical Engineering & Computer Sciences, 26(2), 919–935.<https://doi-org.proxy.library.cpp.edu/10.3906/elk-1706-222>

[5] CHTOUROU, S., MARRAKCHI, Z., AMOURI, E., PANGRACIOUS, V., ABID, M., & MEHREZ, H. (2017). Performance analysis and optimization of cluster-based mesh FPGA architectures: design methodology and CAD tool support. Turkish Journal of Electrical Engineering & Computer Sciences, 25(3), 2044–2054.<https://doi-org.proxy.library.cpp.edu/10.3906/elk-1506-51>

[6] Kelefouras, V., Kritikakou, A., & Goutis, C. (2014). A methodology for speeding up edge and line detection algorithms focusing on memory architecture utilization. Journal of Supercomputing, 68(1), 459–487.<https://doi-org.proxy.library.cpp.edu/10.1007/s11227-013-1049-x>

[7] ARM processor heads for FPGAs. (2005). Electronics Systems & Software, 3(6), 47. Retrieved from<http://proxy.library.cpp.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,uid&db=aph&AN=19497752&site=ehost-live&scope=site>

[8] Galanis, M., Dimitroulakos, G., & Goutis, C. (2006). Performance Improvements from Partitioning Applications to FPGA Hardware in Embedded SoCs. Journal of Supercomputing, 35(2), 185–199.<https://doi-org.proxy.library.cpp.edu/10.1007/s11227-006-2953-0>

[9] Granado-Criado, J., & Vega-Rodríguez, M. (2017). Hardware coprocessors for high-performance symmetric cryptography. Journal of Supercomputing, 73(6), 2456–2482.<https://doi-org.proxy.library.cpp.edu/10.1007/s11227-016-1929-y>

[10] Falamarzi, R., Bahrambeigy, B., Ahmadi, M., & Rajabzadeh, A. (2018). High-performance multi/many-core architectures with shared and private queues: Network processing approaches. Journal of High Speed Networks, 24(2), 89–106.<https://doi-org.proxy.library.cpp.edu/10.3233/JHS-180583>

[11] Domínguez, C., Hassan, H., Crespo, A., & Albaladejo, J. (2015). Multicore and FPGA implementations of emotional-based agent architectures. Journal of Supercomputing, 71(2), 479–507.<https://doi-org.proxy.library.cpp.edu/10.1007/s11227-014-1307-6>

[12] Mishra, A., Kedar, K. P., Shah, H. M., & Raju, K. S. (2015). Implementing Quadratic Function with Resource Constraints Using MicroBlaze Soft-Core in Xilinx EDK. IUP Journal of Telecommunications, 7(1), 7–14. Retrieved from<http://proxy.library.cpp.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,uid&db=buh&AN=101878016&site=ehost-live&scope=site>

[13] Cebrián, J., Natvig, L., & Meyer, J. (2014). Performance and energy impact of parallelization and vectorization techniques in modern microprocessors. Computing, 96(12), 1179–1193.<https://doi-org.proxy.library.cpp.edu/10.1007/s00607-013-0366-5>

[14] Liu YL, Zhang Y, Zhao DM, Tao F, Zhang L (2012) Concept and framework of reconfigurable intelligent optimization algorithm. Adv Mater Res 479-481(5):1875–1879. [https://doi.org/10.4028/ www.scientific.net/AMR.479-481.1875](https://doi.org/10.4028/)

[15] Bayro-Corrochano, E. (2016). Editorial Intelligent Computing for Pattern Recognition, Image Processing and Computer Vision. Intelligent Data Analysis, 20, S1–S3.<https://doi-org.proxy.library.cpp.edu/10.3233/IDA-160841>

[16] Xiaofu Zou, Lina Wang, Yue Tang, Yilong Liu, Shicheng Zhan, & Fei Tao. (2018). Parallel design of intelligent optimization algorithm based on FPGA. International Journal of Advanced Manufacturing Technology, 94(9–12), 3399–3412.<https://doi-org.proxy.library.cpp.edu/10.1007/s00170-017-1447-y>

[17] Abderazek, B., Yoshinaga, T., & Sowa, M. (2006). High-Level Modeling and FPGA Prototyping of Produced Order Parallel Queue Processor Core. Journal of Supercomputing, 38(1), 3–15.<https://doi-org.proxy.library.cpp.edu/10.1007/s11227-006-6719-5>

[18] Rajaraman, V. (2017). Multi-core microprocessors. Resonance: Journal of Science Education, 22(12), 1175–1192.<https://doi-org.proxy.library.cpp.edu/10.1007/s12045-017-0580-0>

 [19] BORKAR, S., & CHIEN, A. A. (2011). The Future of Microprocessors. Communications of the ACM, 54(5), 67–77.<https://doi-org.proxy.library.cpp.edu/10.1145/1941487.1941507>

[20] Rico-Gallego, J. A., Díaz-Martín, J. C., Manumachu, R. R., & Lastovetsky, A. L. (2019). A Survey of Communication Performance Models for High-Performance Computing. ACM Computing Surveys, 51(6), 1–36.<https://doi-org.proxy.library.cpp.edu/10.1145/3284358>

[21] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, Martin Schl¨affer, (2016). Ascon v1.2 Submission to the CAESAR Competition. Graz University of Technology. <https://ascon.iaik.tugraz.at/>

1. [↑](#footnote-ref-1)