Real Time FPGA Implementation of a High Speed for Video Encryption and Decryption System with High Level Synthesis Tools

Ahmed Alhomoud
Department of Computer Sciences, Faculty of Computing and Information Technology,
Northern Border University, Rafha 91911, Saudi Arabia

***Abstract*—The development of communication networks has made information security more important than ever for both transmission and storage. Since the majority of networks involve images, image security is becoming a difficult challenge. In order to provide real-time image encryption and decryption, this study suggests an FPGA implementation of a video cryptosystem that has been well-optimized based on high level synthesis. The MATLAB HDL coder and Vivado Tools from Xilinx are used in the design, implementation, and validation of the algorithm on the Xilinx Zynq FPGA platform. Low resource consumption and pipeline processing are well-suited to the hardware architecture. For real-time applications involving secret picture encryption and decryption, the suggested hardware approach is widely utilized. This study suggests an implementation of the encryption-decryption system that is both very efficient and area-optimized. A unique high-level synthesis (HLS) design technique based on application-specific bit widths for intermediate data nodes was used to realize the proposed implementation. For HLS, MATLAB HDL coder was used to generate register transfer level RTL design. Using Vivado software, the RTL design was implemented on the Xilinx ZedBoard, and its functioning was tested in real time using an input video stream. The results produced are faster and more area- efficient (target FPGA has fewer gates than before) than those of earlier solutions for the same target board.**

Keywords—Security; encryption; decryption; AES; HDL coder; high levelsynthesis; FPGA; Zynq7000

I. Introduction

The development of strong, computationally light, and efficient encryption algorithms is therefore necessary to support the ongoing increase in data volume and throughput in Internet of Things applications, as well as video streaming, real-time video processing, mobile transmissions, and other related activities. This is because network security has become. A constant topic for business activities due to the advancements of information technology (IT) applications involving sensitive data [1,2]. For governments, banks, and high-security systems worldwide, the Advanced Encryption Standard (AES) continues to be the recommended encryption standard [3,4]. The last encryption technique is the most often used; it is utilized in 5G systems, WiMAX, Gigabit Ethernet, and Worldwide Interoperability for Microwave Access [5,6,7]. Furthermore, this algorithm can be effectively implemented on both software and hardware platforms; AES software versions give poorer physical security but require fewer resources [8, 9].

However, the increasing need for secure data transmissions at high speeds and volumes while maintaining physical security makes hardware implementation of the AES algorithm imperative [10,11]. The primary challenge with applications utilized in these domains is to ensure real-time system operation [2]. Implementing real-time functionality on a general-purpose computer is often unfeasible due to the inherent limitations of memory, CPU, and peripheral devices. Typically, numerous actions are executed on every pixel in the majority of image processing programs. The sequential execution of these operations by general-purpose processors has detrimental effects on both resource use and performance [2,3]. Nevertheless, FPGAs (Field Programmable Gate Arrays) possess the potential to function in a parallel manner in relation to hardware, setting them apart from conventional CPUs. Operations in FPGAs are partitioned into segments, allowing for concurrent execution of many operations [12,13]. Fig. 1 presents a complete system for real-time image and video processing using an embedded system.

The landscape of smart video encryption is evolving rapidly, demanding sophisticated analysis of live video streams to accurately identify objects, scenes, and critical events. This has led to the integration of advanced analysis mechanisms into the traditional image processing pipelines of modern video cameras [14]. However, the stringent constraints of real-time processing and low power consumption inherent in camera modules limit the complexity and number of operations that can be feasibly implemented [15]. To address this challenge, researchers have focused on a select few pre-processing tasks like motion estimation, image segmentation, and robust video encryption [15].

Recognizing the growing complexity of computer vision systems, designers are increasingly turning to higher-level programming and synthesis tools to expedite the development process and overcome hardware limitations. Two prominent tools in this arena are Simulink Hardware Description Language (HDL) Coder and Xilinx High-Level Synthesis (HLS) [16, 17]. Xilinx HLS stands out for its exceptional suitability for designing large-scale computer vision systems. It boasts the ability to seamlessly integrate pre-existing standard algorithms and offers comprehensive functional verification, ensuring accuracy and efficiency [17]. Additionally, HDL Coder empowers rapid synthesis and verification of diverse image processing methods, ranging from picture statistics gathering and custom filtering to color space conversion [16].

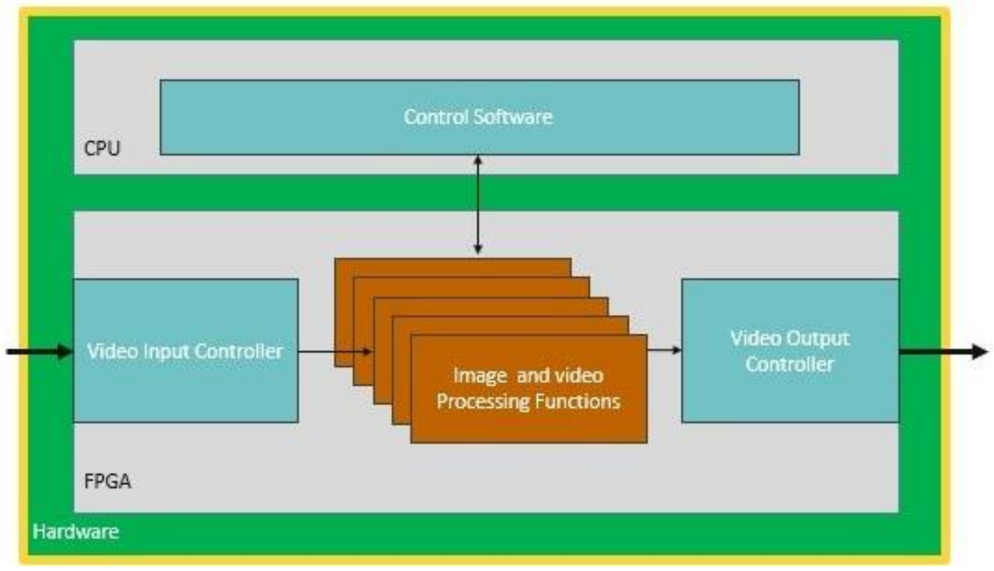

Fig. 1. Embedded real time image and video processing system.

Nevertheless, there is no specific support for picture segmentation tasks in the present toolbox version. To accomplish this, a Simulink model is created that increases this toolbox's capacity and supports this essential feature. While academics have recently proposed a number of advanced methods for image encryption, implement" encryption decryption algorithm" implemented [14, 15] into our proposed hardware in order to minimize the use of logic resources. Moreover, it has been shown that switching from moving averaging to weighted averaging reduces the amount of logic resources needed without sacrificing the accuracy of the results. As a result, the following can be used to summarize the contributions of the completed work: Creation of a synthesizable Simulink model for the AES-based cryptosystem, which isn't yet accessible as an intrinsic block in the Simulink HDL Coder/Vision HDL Coder toolbox (MATLAB R2018b). By substituting the weighted average for the moving average, which necessitates an expensive division operation, logic resource conservation is achieved. In comparison to earlier methods, this work presents a real-time implementation of video encryption and decryption on a Xilinx ZedBoard and shows that it is faster and uses less space on the FPGA. A unique high-level design technique was used to create the design, which synthesizes the design with intermediate signal widths limited by the application (input stimulus). The rest of this essay is structured as follows: An introduction to high- level synthesis is given in Section I. Related work is given in Section II. FPGA High level synthesis is given in Section III. The AES Encryption and Decryption architecture is explained in Section IV. Image and video acceleration is given in Section V. RTL design implementation mentioned in Section VI. Finally Section VII concludes the paper.

II. Related Work

Recently, numerous researchers have undertaken investigations on cryptographic algorithms inside the realm of Internet of Things (IoT). Reference [18] introduced a dynamic pass- word access approach for uniformly storing the key matrix on all nodes. The sender did not need to transmit a basic key, but rather the storage coordinates of the key matrix. The receiver then extracted the key from the matrix based on these coordinates, therefore enhancing the security of key transmission. Reference [19] introduced a Very Large-Scale Integration (VLSI) design that incorporates a 64-bit data path for the lightweight cipher present. This architecture achieves excellent performance, low power consumption, and a compact footprint on FPGA, resulting in a high throughput rate. In order to fulfill the security demands of various application contexts, distinct techniques are necessary for the encryption and decryption system to handle data. Reference [20] developed a customizable encryption system that allowed users to choose their preferred encryption method from a range of options specified in the FPGA configuration file, hence enhancing the flexibility of the system. Reference [21] suggested a hybrid protocol architecture for Short Message Services (SMS) that incorporates AES and Rivest Cipher 4 (RC4) algorithms to enhance the security of smart houses. This solution offered secure communication in the IoT context, ensuring confidentiality and randomness. However, it incurred a certain level of cost in terms of both time and space. Hossain et al. In study [21] author developed a flexible encryption method on the FPGA. Users have the option to choose between the AES, Data Encryption Standard (DES), and 3DES algorithms for encryption, based on their specific needs. This design enhanced system adaptability, but incurred wastage of logic resources. The advent of dynamically reconfigurable technology offered a superior option to achieve a balance between system flexibility and hardware resource usage.

Many studies are devoted to creating specialized hardware accelerators that can be utilized to carry out specific tasks in applications related to image and video processing. To showcase the system's functionality, spatial filters were implemented on the embedded platform Zedboard [22]. The paper examines a recent study on a hardware accelerator for video processing, which was developed on an Altera Cyclone IV FPGA. The accelerator is engineered to possess reduced processing and memory bandwidth demands. The research findings are detailed in citation [23, 24]. Platforms Based Design (PBD) analyzes the Virtex-5 FPGA's performance in real time while processing images and videos by looking for commonalities and differences among different design criteria. To implement an effective architectural solution, the Xilinx ML-507 platform runs a PBD. This system is capable of real-time 60 fps video capturing in VGA resolution [5]. On the Xilinx Zynq7000 System on a Chip (SoC), different designs have been constructed using the Histogram of Oriented Gradients, or HOG, method. These designs can process images with a resolution of 1920 x 1080 pixels and achieve a frame rate of 39.6 frames per second: [6]. The authors demonstrated how to use the OpenCV function on an ARM processor for hardware implementation. The several classifications of hardware accelerated systems that employ Field-Programmable Gate Arrays (FPGAs) for image analysis are concisely listed in reference [2]. FPGAs have the capability to perform concurrent execution of several threads, allowing them to effectively handle a diverse array of applications, including those commonly encountered in the automotive industry. When it comes to driver assistance systems (DA), most researchers have highlighted the improvements in image and video processing [14]. Claus et al. [15] proposed the utilization of dynamic partial reconfiguration (DPR) in driver assistance (DA) systems, employing Multiple Processor System-on-Chip (MPSoC) architecture, to enhance security in various driving situations. The "Autovision" architecture was designed to demonstrate the benefits of DPR by utilizing hardware accelerated engines. Several observations focus on the development of hardware-based real-time lane identification for advanced driving assistance systems. Using the Hough Transform, real-time lane detection is implemented at a clock speed of 100 MHz on the Xilinx Zynq-7000 platform. Thanks to its 480 × 270-pixels resolution, it can run at 130 fps per second. The implementation is performed via the Vivado HLS tool [16]. FPGAs are well-suited for complex video and image processing workloads, such as K-means clustering.

The process of dividing an image into segments and compressing it without any loss of data is referred to as picture segmentation and compressing without loss [17]. Edge detection is an advanced image processing technique mostly used in surveillance systems. The article authored by Kowalczyk et al. [8] examines the practical execution of 4K streaming of videos on Xilinx devices. A high-definition video streaming system utilizing quick prototyping has been developed, employing an FPGA-based edge detection design [9]. The research presents a comparative analysis and investigation into edge recognition filters implemented on Field-Programmable Gate Arrays (FPGAs) for real-time processing of video and images techniques [10]. Babu et al. [11] present a succinct analysis of the various classifications of FPGA architecture and their corresponding applications.

III. FPGA High-Level Synthesis for Image Processing System with MATLAB HDL Coder

After the text edit has been completed, the paper is ready for the template. Duplicate the template file by using the Save As command and use the naming convention prescribed by your conference for the name of your paper. In this newly created file, highlight all of the contents and import your prepared text file. You are now ready to style your paper; use the scroll down window on the left of the MS Word Formatting toolbar.

Fig. 2 displays the simplified block diagram of the proposed system architecture. The design is suitable for both image and video encryption and decryption, as it allows for the sequential streaming of information for each pixel. However, when it comes to video processing, the method may require a single pixel or several pixels.

The system comprises a range of video processing processes that can be controlled by specific algorithms, resulting in the faster execution of certain filters on the integrated blocks of the Xilinx SoC platform. The output is directly transferred to a video processing component and display component. In addition, the processed output is produced and displayed outside on an HDMI monitor. Xilinx implements various video processing accelerators in the programmable logic (PL) based on the design.

Fig. 2 shows the design schematic for the proposed system. This method is applicable to video and image processing due to the sequential processing of the input pixels. When encrypting and decrypting embedded videos, mega pixels are required. Built on the embedded Xilinx SoC platform blocks, it can run complex algorithms faster thanks to a video block that can be configured with unique algorithms. The video processing block and display component directly accept the output from the video and picture systems [16]. The Xilinx Zynq 7000 platform systems are utilized for the implementation of several FPGA video and image processing accelerators [5,14]. The Advanced Extensible Interface (AXI) in the architecture shown in Fig. 1 connects the processor responsible for transferring the incoming video to the video processing pipeline system. The ARM CPU and the USB camera communicate over interfaces and exchange data. The memory controller contains the frame data. The HDMI display exhibits the method when the video IP core has finished executing it on the frame. The pipeline is being maintained for future versions [10].

The necessity to create intricate DSP systems that call for specialized arithmetic units, such the addition-compare- select unit for the Viterbi decoder, gave rise to the MBD approach for FPGAs. A more refined degree of FPGA-based circuit optimization is needed for these computing units [17]. This degree of optimization is typically linked to conventional digital design systems for processing images and videos. In essence, model-based design describes the real- time interactions that a system will have with the analog environment. The bulk of these apps make use of the widely used Unified Modeling Language (UML). The methods by which these tools establish and describe a system differ. These tools may employ various implementation tactics, some of which may prove to be less effective than others. However, they assure rapid system prototyping, ensuring punctuality. Some factors that affect the choice of a tool are its level of flexibility, its availability as pre-built libraries and blocks, and its general understanding.

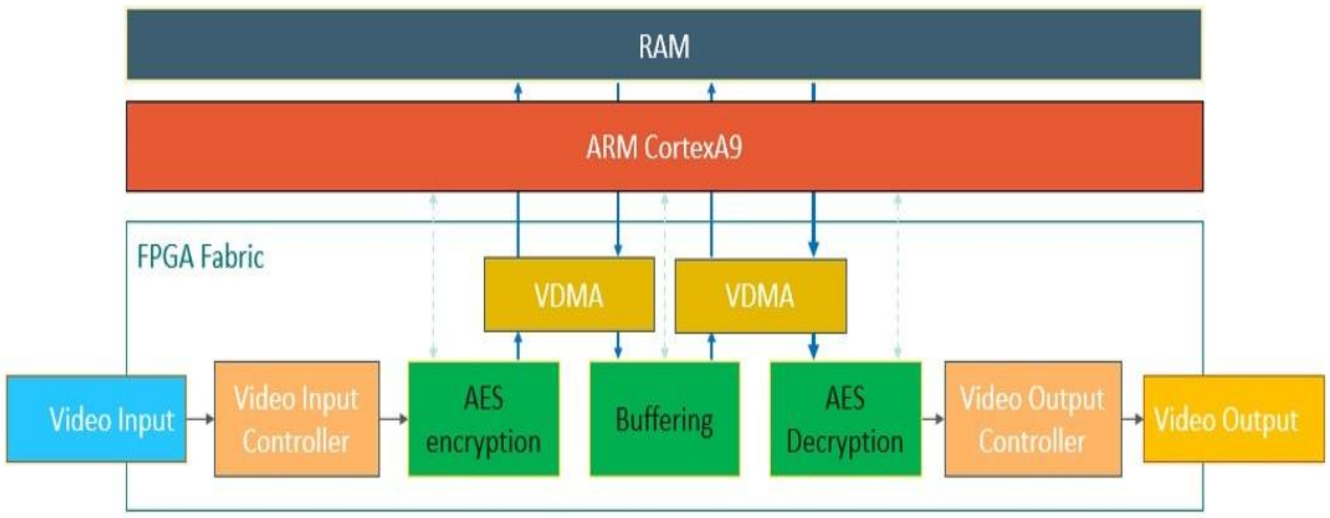

Fig. 2. Crypto-video system on FPGA based on Zynq7000 platform.

Matlab, Simulink Realtime Workshop, Arti-Real Time Studio, and Rhapsody from Ilogix are among the UML-based tools [15]. These instruments are used in the creation of embedded multiprocessor environments. There are two types of tools that facilitate the creation of HDL code for FPGA: block-based tools and C language-based tools that utilize blocks to produce HDL code from block diagrams. Following that, the HDL code is utilized by the hardware of the synthesis tool in order to put the system design into action on the FPGA computer. Synplify DSP, Xilinx System Generator, DSP Builder Altera, and Simulink HDL Coder are some of the tools that are predicated on the Simulink and MATLAB environments. The majority of these tools are based on these environments. These technologies ensure a sophisticated modeling environment for signal processing algorithms. Combining the IP cores from FPGA manufacturers with blocks from the Simulink library results in the creation of HDL code that is singular to the platform in question. The designer benefits from more freedom through the utilization of tools such as Simulink HDL Coder, which seamlessly incorporates MATLAB functions and m-block files. The designer generates a Simulink model and subsequently transforms it into the FPGA environment with these tools.

The second group of MBD tools uses C to construct a system design abstraction. Among these are the Handel-C from Celoxica and the Catapult C from Mentor Graphics. A popular tool for developing embedded systems on FPGA, the Simulink HDL coder is the main engine behind these products [9].

MathWorks introduced its hardware/software workflow for Zynq-7000, with a specific focus on Model-Based Design (MBD), in September 2013, as stated in references [10, 11]. According to the depicted process in Fig. 3, Simulink is utilized with HDL toolkit to create models that may effectively demonstrate a fully dynamic system. These encompass a Simulink model designed for algorithms specifically tailored for the Xilinx Zynq SoC platform. Additionally, it enables the rapid creation of software-hardware implementations for the Zynq platform directly from the algorithm and system architecture.

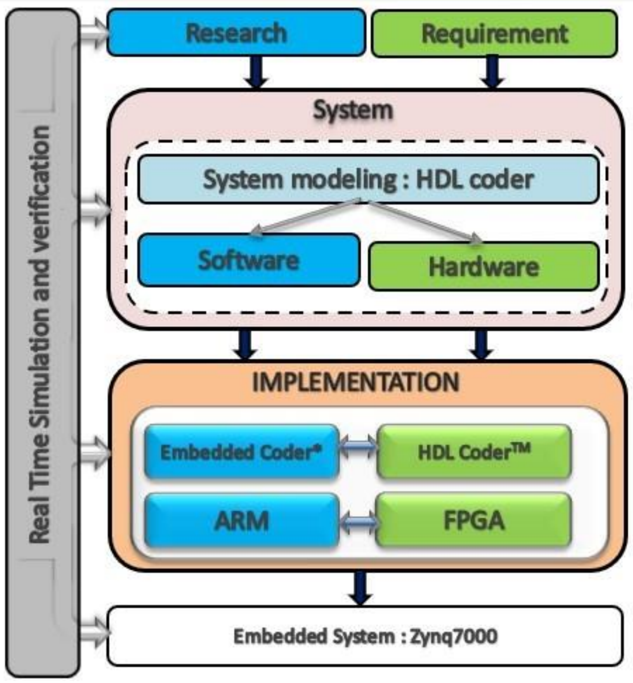
Real Time Simulation and verification
Research
Requirement
System
System modeling : HDL coder
Software
Hardware
IMPLEMENTATION
Embedded Coder®
HDL Coder™
ARM
FPGA
Embedded System : Zynq7000

Fig. 3. Model based design prototyping with MATLAB / HDL coder.

The development of the suggested real-time video processing system is bifurcated into two components: 1) Designing the architecture of a video processing system, and 2) Designing algorithms for video processing. The initial section examines the primary elements that contribute to the video processing system on the Zynq platform, specifically focusing on the AXI4 Interfaces utilized for efficient data transmissions. The subsequent section delves into the strategies and enhancements implemented for constructing video processing algorithms using Vivado HLS [25]. The proposed approach is founded upon the following fundamental principle:

- The utilization of Simulink simulation by system designers and algorithm developers serves two purposes.

The designer's task is building models for an entire system, encompassing communications, image, and video processing components. The second purpose is to enable the division of the model into hardware and software components and achieve a favorable balance for high-level synthesis.

- The Xilinx Zynq 7000 platform can be equipped with high-speed I/O cores and IP cores through the utilization of HDL code generation from HDL coder TM.
- The Zynq Cortex-A9 cores can be programmed using the embedded coder from Simulink, which facilitates quick iteration of embedded software development.
- The ARM processor system and programmable logic with support for Xilinx Zynq 7000 may generate automatic AXI4 interface cores.
- Integration with subsequent processes, such as software compilation, generating the executable for the ARM, and creating bit streams using Xilinx implementation tools like Vivado, allows for a fast prototype process. These jobs can be directly downloaded to Zynq 7000 platform boards.

IV. Fundamentals of the AES-128 Encryption / Decryption Algorithm

Similar to the Data Encryption Standard (DES), the AES algorithm presented in Fig. 4 operates as a block cipher at the bit level. Each block length is set at 128 bits, whereas the key length can be any value between 128 and 256 bits [20]. Every 128-bit data block is divided into 16 bytes, which are then mapped onto a 4 × 4 array called state. Every byte in the state represents a 2 x 8 cardinality Galois Field (GF) element. The algorithm consists of n rounds, or iterations, depending on the length of the key. For a key length of 128 bits, 192 bits, or 256 bits, the number of rounds is 10, 12, or 14. With the exception of the final round, each encryption round consists of the following four operations:

- Substitute Bytes
- Shift Rows
- Mix Columns

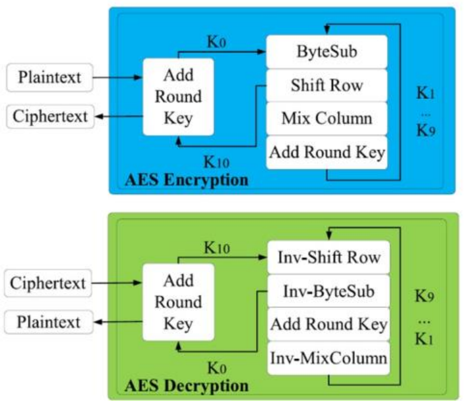

Fig. 4. AES encryption and decryption process.

- Add Round Key

Every operation is executed in turn throughout each round, with the exception of the first Add Round Key; the Mix Columns action is skipped in the last round (see Fig. 5).

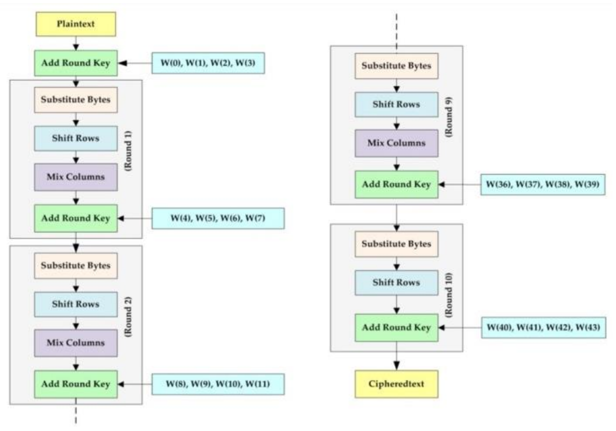

Fig. 5. AES 128 encryption algorithm.

The Substitute Bytes step is a non-linear transformation in which the relationship between the key and the cipher-text is hidden by replacing each byte in the state array with the entry of a fixed 8-bit Substitution Box (Sbox), which is implemented as a lookup table with 2 8 words of 8 bits each. To prevent assaults based on basic algebraic features, the Sbox utilized is constructed from the multiplicative inverse over GF(28) and paired with an invertible geometric transformation, yielding a 16 × 16 bytes table (see Fig. 5). Based on the most and least significant nibbles of the 8-bit input data, the permutation is obtained T The bytes in each row are circularly shifted by a specified offset during the Shift Rows step's operation on the state array's rows. Every byte in the second row is moved one position to the left; similarly, the third and fourth rows are shifted by two and three bytes to the left, respectively. The first row remains unmodified. In the Mix Columns phase, each column of the state array is mixed linearly by treating it as a polynomial over GF(28). Each column is then multiplied, modulo z4 + 1, by a fixed polynomial (c(z) = 03z3 + 01z2 + 01z + 02). The relationship between the plain text and the ciphertext must be concealed using both the Mix Columns and Shift Rows methods.

V. Image and Video Acceleration with High Level Synthesis

The functional diagram of the suggested system design is displayed in Fig. 6. Since the input pixels are processed sequentially, the approach can be used to both image and video processing. Mega pixels are needed in the embedded video processing process. It is constructed from a video block that can be programmed with customized algorithms, speeding up complicated and resource- and time-consuming algorithms on the embedded Xilinx SoC platform blocks. A video processing block and a display component receive the output directly from the video and image systems [16]. The Xilinx Zynq 7000 platform systems are used to implement the several FPGA video and image processing accelerators [5] One MBD tool that makes system modeling, analysis, and simulation possible is Simulink HDL coder. It provides the designer with an organized graphical environment that enables the creation of highly complicated system designs. Moreover, the user of this application can generate adaptable custom blocks using MATLAB functions. utilizing Simulink blocks, the designer can produce bit-precise and synthesizable HDL code from the model by utilizing the Simulink HDL coder. Altera Quartus II, Synplify, and Xilinx Vivado are some of the tools that can be used to synthesis and map the obtained HDL code to the target FPGA board. There are numerous built libraries in the Simulink HDL coder [11]. Adders, multipliers, accumulators, integrators, multi-port switches, lookup tables, etc. are a few examples of these preset libraries.

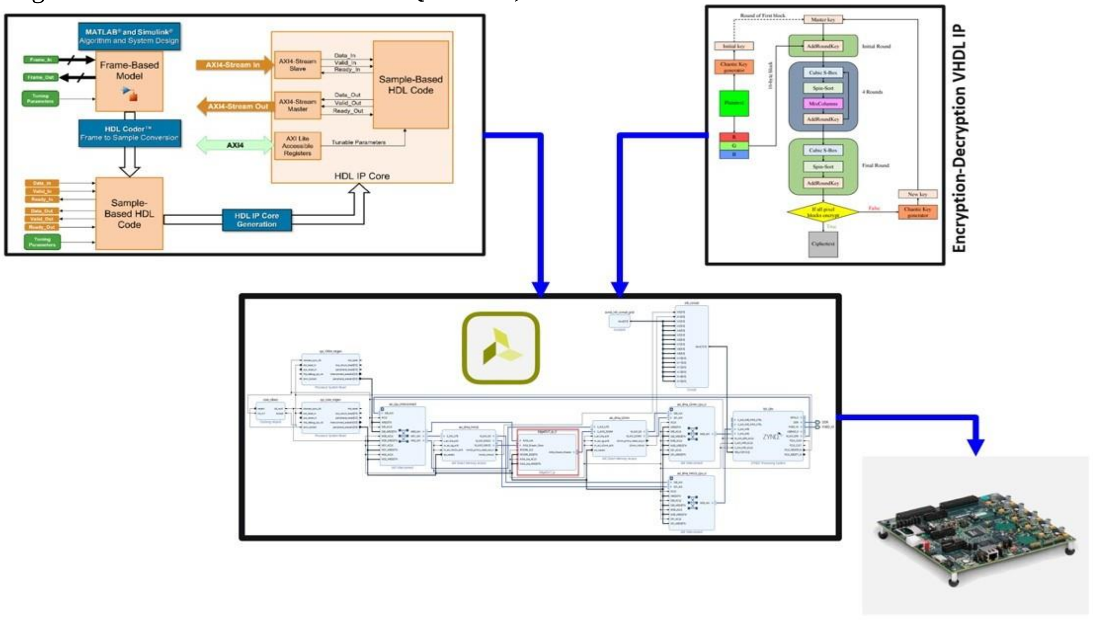

Fig. 6. FPGA crypto system design.

A typical MBD design flow for implementing FPGA-based video and image processing system is shown in Fig. 6. this system is divided on two part,First part is to generate a complete video processing system based on Matlab/HDL Coder . This generated vivado project system is without encryption and decryption IPs.The second part is designing with VHDL language the top level for encryption and decrytion block.After Encryption and decryption IPs simulation and verification. Theese IPs are added to the full complete vivado project for real time video process-ing.Finally the final combined system for video encryption and decryption is implemented and verified on zynq7000 paltform.

VI. RTL Design Implementation

A. Simulation

To validate the accuracy of the encryptions and decryption implementation, a testbench has been created to compare two distinct ciphertexts generated by this implementation with the expected true ciphertexts provided in reference. The implementation successfully passes the verification process, and a snapshot of the waveform produced from the simulation using the vivado 2017.4 simulator is shown in Fig. 7.

B. Synthesis and FPGA Implementation

In the third and final phase (see Fig. 8), the Simulink HDL Coder transforms the Software-Hardware model of the video processing system into an IP core that is compliant with the AXI4 streaming bus and is in the form of HDL (VHDL) source code. Encryption and decryption IP designed with VHDL language are integrated using Vivado to the full System to generate the RTL video encryption and decryption design as presented in Fig. 8.

In order to confirm the functionality of this IP core in a real-time, practical setting, a Hard- ware–Software co-design (HW–SW) has been directly implemented on a 170MHz Xilinx Zynq-7000 AP SoC XC7Z020- CLG484 FPGA. A Full Vivado project is generated for the HW-SW by the Simulink HDL coder. The Xilinx Vivado tool (version 2017.4), along with all the hardware and software add-ons, can be used to implement this project. Additionally, a single bus connects the Sobel core and the Color transform IP core. The video processing system Vivado project now includes the AES encryption-decryption based on VHDL as an IP. The RTL design for our crypto-video system is shown in Fig. 8.

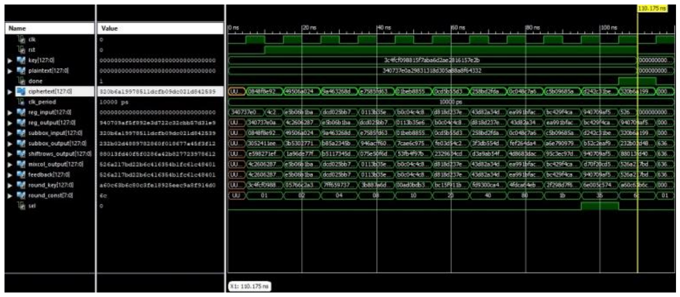

Fig. 7. RTL simulation for AES encryption.

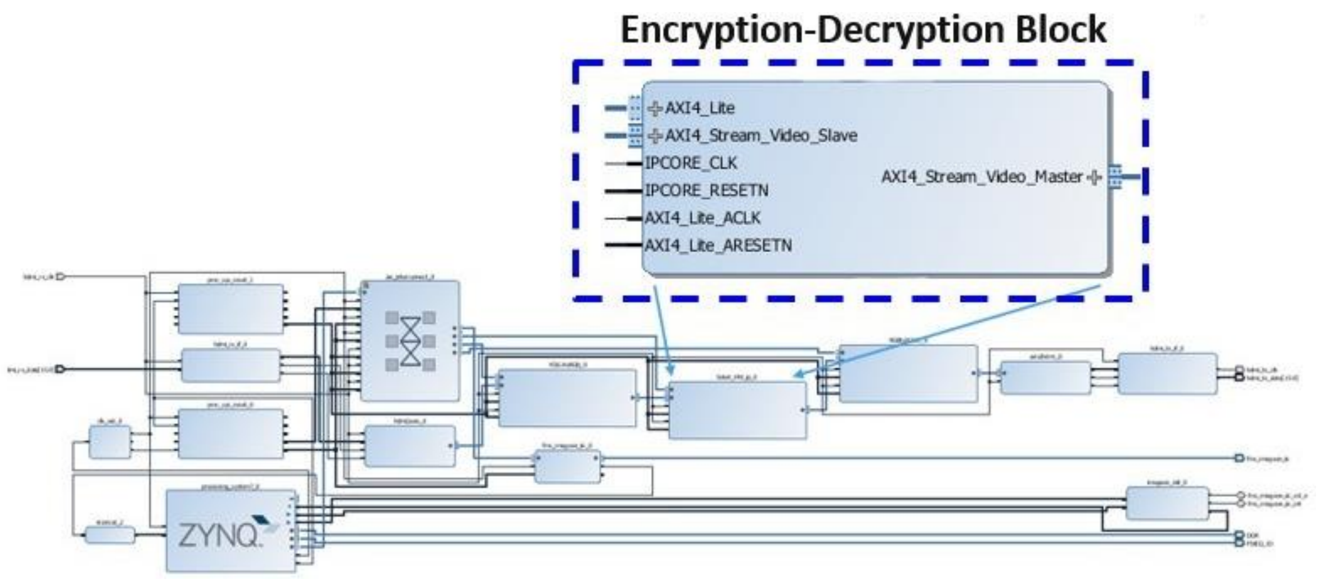

Fig. 8. RTL video encryption and decryption design.

C. Resource Utilization

The proposed system used the Zynq-7000 SOC to implement the cores of our video processing system. Table I lists the materials that were utilized. BRAM is used to store data values, firmware, and instruction memory.

The architecture of the processor, which includes control signals, internal registers, and mi- crocode, is defined by the consumed LUTs and DFFs. Our suggested system core operates best at about 296MHz with a throughput of 95FPS. The Zynq7000 SOC's resources are impacted by the intricacy of the applied design. The suggested architecture operates with a 1080-1920 frame input resolution. These findings support the Vivado-based reconfigurable SOC platform.Table I provides a breakdown of the resources used by the system, including Block RAM (BRAM) for data storage and Look-Up Tables (LUTs) and Flip-Flops (DFFs) for the processor architecture. This information is crucial for understanding the resource footprint of the design and its feasibility for different Zynq-7020 models.

D. Encryption / Decryption Time Test

Encryption and dcreyption speed is crucial. The timer is implemented to precisely track encryption and decryption cycles, determining the exact number of operations needed for each process. This granular data allows us to optimize performance and ensure efficient data protection.

$$T_{Proc} = N_C \times T_{cycle} = \frac{N_c}{FPGA_{Freq}}$$

where, Nc, the total encryption cycles for the image, is calculated by multiplying the clock cycle duration (Tcycle = 1/FPGAFreq) by the total execution time.

For a 512x512 image, the proposed encryption decryption algorithm required approximately 53.96 million cycles (75ms) and decryption required 54.85 million cycles (80ms). This compares favorably to other solutions in the literature (see Table II).

TABLE I. VIDEO ENCRYTION AND DECRYPTION RESOURCES UTILISATION

Xilinx Platform	Zynq 7000 XC7Z020-1CLG484C
Maximum Frequency	296.789MHz
LUT-FF Pairs	1104
LUTs as Logic	1104
LUTs as Memory	18
Slice Registers	264
RAM 36/18	1
Frame Rate	95FPS

TABLE II. PROPOSED SYSTEM TIME COMPARISON

IP	**Time (s)**
Encryption Time	0,075
Decryption time	0,08
Image preprocessing time	0.035
Total	**0.19**

VII. CONCLUSION

Accurate implementation PPA details can be difficult, if not impossible, to obtain during the algorithm development phase in a normal design flow because doing so means putting a lot of work on the implementation team to complete experimental implementations. Algorithm developers employ imprecise estimations for PPA prediction because of this difficult task. The PPA objectives are frequently not met as a result of algorithm developers' inaccurate estimations. Algorithm developers an quickly obtain precise PPA information using the MATLAB connection with Stratus HLS, which enables measurement- driven algorithm improvement. The solution automates a large portion of the manual process from MATLAB to implementation details with minimal disruption to the current design flow. Additionally, this connection automates the creation of more optimal RTL and micro-architectural exploration, resulting in shorter design deadlines and better PPA. According to preliminary findings, this process will essentially become the norm for creating and executing silicon-targeted algorithms. In this study, the application of typical AES encryption is investigated on a Xilinx ZedBoard equipped with a Zynq- 7000 SoPC. This effort concentrated on the encryption side of AES128; however, it would not be difficult to construct and test the decryption side as well. Xilinx Vivado High Level Synthesis was used to implement the AES after it was first coded in a high-level language. It will be easy to swiftly implement our design and perform changes that significantly raised the AES algorithm's throughput thanks to the Xilinx HLS tool. Additionally, HLS has the ability to enable thorough examination of a design's resource utilization in comparison to high-level code placement, as well as hardware testing during the early phases of design.future work will explore expanding the HLS-MATLAB integration to encompass the decryption side of AES as well as investigating its application to a wider range of algorithms across diverse domains. This continued exploration holds immense promise for revolutionizing the way the proposed system is developed and deployed efficiently, high-performance hardware solutions.

Acknowledgment

The authors extend their appreciation to the Deanship of Scientific Research at Northern Border University, Arar, KSA for funding this research work through the project number "NBU-FFR-2024-1092-01".

References

[1] Li, L.; Li, S. High throughput AES encryption/decryption with efficient reordering and merging techniques. In Proceedings of the 2017 27th International Conference on Field Programmable Logic and Applications (FPL), Gent, Belgium, 4–6 September 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1–4.

[2] Babu, P., and Parthasarathy, E. (2021). Hardware acceleration of image and video processing on Xilinx zynq platform. Intell. autom. soft comput, 30(3).

[3] Elsayed, G., Soleit, E., and Kayed, S. (2023). FPGA design and implementation for adaptive digital chaotic key generator. Bulletin of the National Research Centre, 47(1), 122.

[4] Li, K., Li, H., and Mund, G. (2023). A reconfigurable and compact subpipelined architecture for AES encryption and decryption. EURASIP Journal on Advances in Signal Processing, 2023(1), 1-21.

[5] Visconti, P.; Capoccia, S.; Venere, E.; Vela´zquez, R.; Fazio, R.d.10 Clock-Periods Pipelined Implementation of AES-128 Encryption- Decryption Algorithm up to 28 Gbit/s Real Throughput by Xilinx Zynq UltraScale+ MPSoC ZCU102 Platform. Electronics 2020, 9, 1665. https://doi.org/10.3390/electronics9101665.

[6] Abd El-Maksoud, A. J., Abd El-Kader, A. A., Hassan, B. G., Rihan.

[7] N. G., Tolba, M. F., Said, L. A., ... and Abu-Elyazeed, M. F. (2020). FPGA implementation of integer/fractional chaotic systems. Multimedia Security Using Chaotic Maps: Principles and Methodologies, 199-229.

[8] Wang, D., Lin, Y., Hu, J., Zhang, C., and Zhong, Q. (2023). FPGA Implementation for Elliptic Curve Cryptography Algorithm and Circuit with High Efficiency and Low Delay for IoT Applications. Micromachines, 14(5), 1037.

[9] Maazouz, M., Toubal, A., Bengherbia, B., Houhou, O., and Batel.

[10] N. (2022). FPGA implementation of a chaos-based image encryption algorithm. Journal of King Saud University-Computer and Information Sciences, 34(10), 9926-9941.

[11] Rajasekar, P.; Haridas, M. Efficient FPGA implementation of AES 128 bit for IEEE 802.16e mobile WiMax standards. Circuits Syst. 2016, 7, 371–380.

[12] Elsayed G, Kayed SI (2022) A comparative study between MATLAB HDL Coder and VHDL for FPGAs design and implementation. J Int Soc Sci Eng 4:92–98.

[13] Al-Musawi, W. A., Wali, W. A., and Al-Ibadi, M. A. (2021, July). Implementation of Chaotic System using FPGA. In 2021 6th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS) (pp. 1-6). IEEE.

[14] Del-Valle-Soto, C.; Vela´zquez, R.; Valdivia, L.J.; Giannoccaro, N.I.; Visconti, P. An Energy Model Using Sleeping Algorithms for Wireless Sensor Networks under Proactive and Reactive Protocols: A Performance Evaluation. Energies 2020, 13, 3024.

[15] Noorbasha, F.; Divya, Y.; Poojitha, M.; Navya, K.; Bhavishya, A.; Rao, K.; Kishore, K. FPGA design and implementation of modified AES based encryption and decryption algorithm. Int. J. Innov. Technol. Explor. Eng. 2019, 8, 132–136.

[16] Ghodhbani, R., Saidani, T., Alhomoud, A., Alshammari, A., and Ahmed, R. (2023). Real Time FPGA Implementation of an Efficient High Speed Harris Corner Detection Algorithm Based on High-Level Synthesis. Engineering, Technology and Applied Science Research, 13(6), 12169-12174.

[17] Sikka, P., Asati, A. R., and Shekhar, C. (2021). Real time FPGA implementation of a high speed and area optimized Harris corner detection algorithm. Microprocessors and Microsystems, 80, 103514.

[18] Tsai, Y. H., Yan, Y. J., Hsiao, M. H., Yu, T. Y., and Ou-Yang, M. (2023). Real-Time Information Fusion System Implementation Based on ARM-Based FPGA. Applied Sciences, 13(14), 8497.

[19] Park, J.; Park, Y. Symmetric-Key Cryptographic Routine Detection in Anti-Reverse Engineered Binaries Using Hardware Tracing. Electronics 2020, 9, 957.

[20] Ghodhbani, R., Horrigue, L., Saidani, T., and Atri, M. (2020). Fast FPGA prototyping based real-time image and video processing with high- level synthesis. International Journal of Advanced Computer Science and Applications, 11(2).

[21] Bellemou, A.M.; Garc´ıa, A.; Castillo, E.; Benblidia, N.; Anane, M.; A´ lvarez-Bermejo, J.A.; Parrilla, L. Efficient Implementation on Low Cost SoC-FPGAs of TLSv1.2 Protocol with ECCAES Support for Secure IoT Coordinators. Electronics 2019, 8, 1238.

[22] MathWorks, inc: HDL CoderTM User's GuideCOPYRIGHT 2012-2015 (2012) https://www.mathworks.com/help/hdlcoder/ Accessed 20 Feb 2023.

[23] Guerrieri, A., Upegui, A., and Gantel, L. (2023). Applications Enabled by FPGA-Based Technology. Electronics, 12(15), 3302.

[24] Sankar D, Lakshmi S, Babu C, Mathew K (2023) Rapid prototyping of predictive direct current control in a low-cost fpga using hdl coder. Int J Power Energy Syst 43(10):1–9. https://doi.org/10.2316/J.2023.203-0437.

[25] Yoon, I., Joung, H., and Lee, J. (2016). Zynq-based reconfigurable system for real-time edge detection of noisy video sequences. Journal of Sensors, 2016.