https://doi.org/10.17563/rbav.v39i3.1179

Design of DES encryption algorithm as bundled-data asynchronous pipeline using FPGA

Projeto do algoritmo de encriptação DES como uma pipeline assíncrona bundled-data em FPGA

Diego Augusto Silva[1,*], Duarte Lopes Oliveira[1], Gracieth Cavalcanti Batista[1]

ABSTRACT
Currently, digital systems that are able to meet major security restrictions are increasingly being demanded, both in the military and in commercial areas. Data security can be achieved by cryptographic algorithms. An important encryption algorithm known as data encryption standard (DES) was implemented in field programmable gate array (FPGA) in different synchronous architectures. In this paper, we have proposed the implementation of the DES algorithm in FPGA, in the asynchronous pipeline style. Compared to the implementation in FPGA using two different project styles, the proposed asynchronous obtained the average increase of 14.9% in throughput and the average reduction of 66.3% in latency.

Keywords: Asynchronous logic, Control, Side channel attacks, Pipeline, Data-path, System-on-chip, Differential power analysis.

RESUMO
Atualmente, sistemas digitais que são capazes de atender a maiores restrições de segurança estão em crescente demanda, ambos nas áreas comercial e militar. Segurança de dados pode ser atingida através de algoritmos criptográficos. Um importante algoritmo de encriptação, conhecido como Data Encryption Standard (DES), foi implementado em uma Field Programmable Gate Array (FPGA) em diferentes arquiteturas síncronas. Nesse artigo, foi proposta a implementação do algoritmo DES em FPGA no estilo de pipeline assíncrono. Em comparação às implementações em FPGA usando dois diferentes estilos de projeto, a arquitetura assíncrona proposta apresentou um aumento médio de 14,9% na taxa de transferência de dados e uma redução média de 66,3% na latência.

Palavras-chave: Lógica assíncrona, Controle, Ataques por canais laterais, Pipeline, Data-path, System-on-chip, Differential power analysis.

1. Departamento de Ciência e Tecnologia Aeroespacial - Instituto Tecnológico de Aeronáutica - Divisão de Engenharia Eletrônica – São José dos Campos (SP), Brazil.

Correspondence author: duarte@ita.br

Received: Oct. 29, 2020 **Approved:** Nov. 24, 2020

INTRODUCTION

In recent decades, there is a strong demand for digital systems that ensure the confidentiality of information, whether in processing or data storage. As an example, we have the purchasing activities on internet, banking, etc., which require transmission security and sensitive data storage. The digital system design, meeting these security restrictions, demands communication protocols and uses encryption methods. These methods are based on the arithmetic and focus on hiding data. Currently, there is also concern with the inclusion of traps in digital system-on-chip (SoC) systems design, especially for military purposes[1]. For example, cryptographic algorithms are intensively applied in software-defined radio, a strategic area in the military sector[2]. We can also mention aerial drones, which are mobile cyber-physical systems, with applications in military operations, package delivery, reconnaissance, etc. There are applications in which aerial drones must be highly targeted, therefore insurances, like in military ones, in which attacks on these drones should be frequent, so that some important information can be extracted[3].

Although the encryption algorithms implemented in SoC seek to be robust to resist the attempts of breaching confidential data, there is a number of techniques that demonstrate, through physical properties, that it is possible to reveal the secret processed data[4,5]. This class of techniques is known as side channel attacks (SCA), which extracts sensitive information based on physical features, such as power consumption, electromagnetic radiation, processing time, etc., allowing the discovery of the information protected by encryption. These attacks seek to establish a relation between the analyzed physical features and the processed data.

A cryptographic system typically uses a secret cryptographic key, which affects its efficiency. In modern cryptographic systems, knowing the key is equivalent to being able to perform operations on the encrypted system. Different encryption algorithms have been proposed to raise the reliability of data security, such as Rivest-Shamir-Adleman (RSA)[6], tiny encryption algorithm (TEA)[7], advanced encryption standard (AES)[8] and data encryption standard (DES)[9]. The DES algorithm became one of the most popular algorithms in the late twentieth century. It was developed by the International Business Machines Corporation (IBM), with some help from the National Security Agency (NSA) in the 1970s. In 1977, it was adopted as an information processing standard in USA agencies[10,11]. The security of the DES algorithm lies in the size of the key and in the difficulty in decrypting without knowing the key. The operations of DES encrypt and decrypt are publicly owned. The DES algorithm is relatively slow if implemented by software, due to the size of the key and the permutation involving a 64-bit input block.

Different proposals have been made for the implementation of cryptographic systems, aiming at a greater reliability concerning attacks by hardware. We can mention the implementation of DES algorithm in the synchronous style in field programmable gate array (FPGA)[12-20] or in very large-scale integration (VLSI)[21,22]. In deep-sub-micron (DSM) MOS technology, used today, the implementation of synchronous circuits causes difficulties related to the global clock signal, for example, clock skew, clock distribution network, high electromagnetic emission, low modularity and high noise. Asynchronous style is a promising alternative for solving problems related to the global clock signal. In the asynchronous style, there is the implementation of DES algorithm by Zhang et al.[23], which work in the quasi-delay-insensitive (QDI) class, and in other authors' works[24-26], which implement the globally asynchronous locally synchronous (GALS) style.

Circuits implemented in devices based on vacuum microelectronics have interesting properties, such as robustness to temperature variations, allowing high electrical current, as well as radiation tolerance[27,28]. These circuits are desirable in space applications and adapt very well to the asynchronous paradigm, even if they are of an optical or a quantum style.

This paper has proposed a high-performance DES cryptographic processor, which is synthesized on asynchronous pipeline architecture and prototyped in FPGA. This proposed architecture consists of eight stages, operates on the two-phase handshake protocol and is bundled-data, so the data-path in each stage is synthesized in the conventional way, that is, single-rail[29]. Comparing two design styles– synchronous pipeline and multi-point GALS of [25]–, the proposed asynchronous pipeline achieved the average reduction of 66.3% in latency and the average increase in throughput of 14.9%.

PREVIOUS WORKS

In the end of the 1990s, some works concerning attacks by means of physical characteristics of devices running cryptographic algorithms were presented. Kocher[30] has reported weakness of some algorithms related to their timing characteristics, like differences in the encryption time of different keys and plaintexts that would be exploited for attacks in some processor architectures. He has suggested some countermeasures, as random delay insertion in the operations and time uniformization of all needed operations, what would turn the timing characteristics of algorithms unfeasible to analysis. The attacks based on timing characteristic were named timing attacks[30]. Another work has showed the possibility to analyze power consumption and electromagnetic emission from chips based on complimentary metal-oxide-semiconductor (CMOS) technology, due the switching characteristics of these devices. This work focused on a class of attacks called power analysis and brought special attention to the differential power analysis (DPA), which is simple to be performed. The authors have proposed some countermeasures against DPAs, like device shielding and noise insertion.

In the beginning of the 2010s, DPA countermeasures were divided in strategies of uniformization, randomization and masking. Soares et al.[24] developed a strategy against SCAs based on randomization. They implemented the DES algorithm in GALS pipeline architecture, using a two-phase handshaking protocol as an interface protocol between the synchronous islands and a random clock frequency system that at each round fed the islands with random clock frequencies. The handshaking protocol was also used for the communication between islands and its own clock systems. The goals of this architecture were to hide the leakage of information by randomization of execution time, provided by the random clocks, and to overlap the current measurements caused by the pipeline. This proposed architecture achieved robustness against SCAs attacks when compared to versions of the same algorithm implemented in full synchronous and asynchronous styles[24]. Other works[25,26] implemented the DES algorithm in the GALS style, providing increase in robustness to attack based on the analysis of the clock signal.

In 2017, a work[31] showed an energy-based attack flow against the architecture proposed for Soares et al.[24]. This attack was based on current traces time-alignment and subsampling and achieved success to find sub-keys of two-stage pipelined GALS architectures. The authors also proved that this attack flow had relatively low computational cost compared to other previous trials[31].

ASYNCHRONOUS CIRCUITS: OVERVIEW

Asynchronous circuits operate by events not needing a global signal to synchronize the operations. The synchronization in asynchronous circuits is performed by handshaking protocols. Four different design styles are used[32,33]: decomposition (controller with data-path); asynchronous pipeline; macromodules composition; and desynchronization. Each one of these styles can be designed in a different class. The class of an asynchronous design defines the delay model in which the circuit operates properly and its interaction with the environment[32,33].

The main delay models are: gates and wires with bounded delays (GWBD) ($t_{min} \leq t_d \leq t_{max}$); and gates and wires with unbounded delay (GWUD), in which the delay is not defined, but finite. Two important asynchronous circuits that follow the GWBD model are: burst-mode circuits and extensions and timed circuits[32].

Delay insensitive circuits (DI) follow the GWUD delay model. This model is more robust and free of timing analysis, but the application of these circuits is very limited[34].

Two classes of circuits that follow a less restrictive variant of the GWUD delay model are: speed independent circuit (SI) and QDI circuit. In the SI circuits, the gate delay is undefined, but finite, and the wire delay is zero[32]. QDI circuits follow the GWUD model with the isochronic fork restriction. This restriction defines that wires with fanout > 1 (with fork) have the same delay[34]. The circuit interaction with the environment is either performed in the generalized fundamental mode (GFM) (for example, burst mode) or in the input/output mode (M_I/O) (for example, DI, SI, QDI)[33]. In GFM, the change of a new burst input is allowed if the circuit has already reached a stable condition. In M_I/O, the reaching of the corresponding output signal enables the change of the input signals.

Asynchronous pipeline style

The design style denominated micropipeline was proposed by Sutherland[35] (see Fig. 1), and its architecture can be linear or nonlinear, with non-conventional registers and control composed of a C-element and an inverter gate. This paper deals with the linear architecture, which is used in signal processing. The main feature of the micropipeline is the simplification of the pipeline control, an important characteristic for asynchronous systems implemented in FPGA. The control is either distributed between stages or centralized, and it is responsible for the communication between the pipeline stages. In a FPGA platform, the control must be distributed in order to avoid hazard problems. The pipeline communication employs the handshake protocol with request and acknowledge signals[32,33]. The communication between the stages can be performed in two different protocols: 4-phase or 2-phase. Fig. 2 shows the behavior of both.

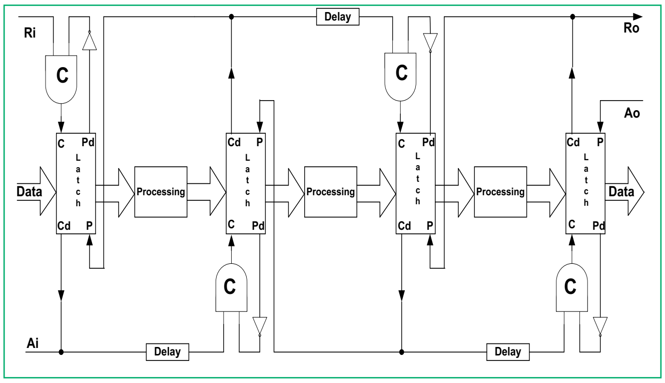

Figure 1: Bundled-data linear micropipeline[35].

The linear asynchronous pipeline design has two architectures. The first one is called bundled-data pipeline and uses only components of the synchronous paradigm. The second architecture is the QDI linear pipeline, which employs dual-rail components.

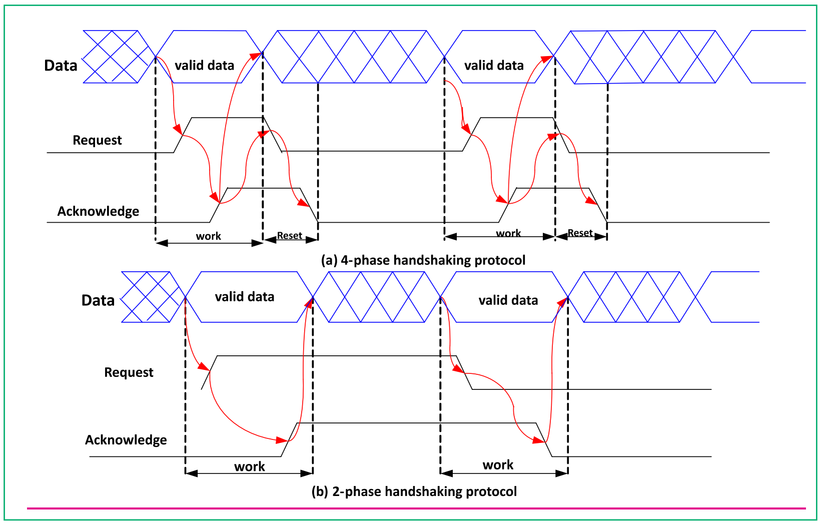

Figure 2: Handshake protocol: (a) 4-phase; (b) 2-phase.

Bundled-data asynchronous pipeline architecture

The bundled-data pipeline architecture uses single-rail functional units and delay elements between stages. The delay is defined considering the critical path of each stage. Different architectures have been proposed for the linear pipeline design style[36-39]. However, these pipeline architectures are generally focused on VLSI implementations, employing full-custom control[37]. FPGA-oriented pipelines have been proposed, but either complex delay mechanisms are used[38] or the operation in the fundamental-mode is not observed[39]. Mousetrap is an asynchronous pipeline architecture based on logic gates which can be implemented in FPGA and has good performance (see Fig. 3)[36].

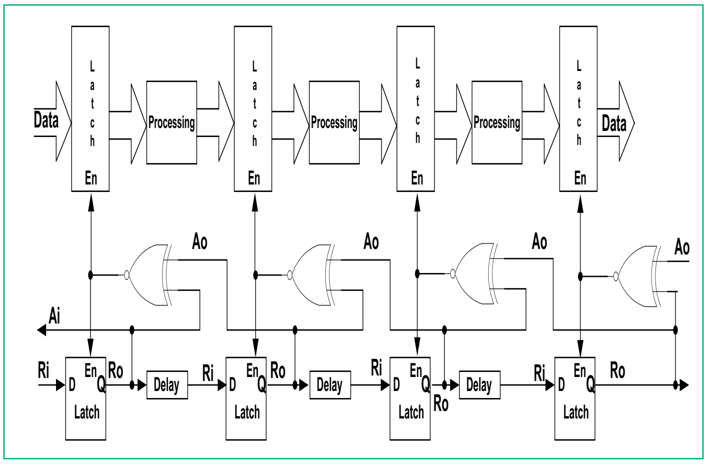

Figure 3: Mousetrap linear pipeline[36].

CRYPTOGRAPHIC ALGORITHM: DATA ENCRYPTION STANDARD

Figure 4 shows the basic flow of DES algorithm's encryption process. At the beginning of the process, the bits of the plaintext are permuted by the initial permutation IP and then divided into two symmetric parts. After this, 16 iterations of a round function is performed, consisting in nonlinear transformations of one side followed by an XOR operation. Each round uses a sub-key of 48 bits generated from an original key of 64 bits in a process called key scheduling. In the end, the right and left sides of the word are concatenated, and another permutation is applied. The permutations are simply input bits mapping to predetermined positions, which are defined in the official documentation of the DES algorithm by the Federal Information Processing Standard (FIPS). In hardware level, permutations can be implemented by means of wiring. The same structure of Fig. 4 can be used to perform the decryption process, differing from the encryption, which uses sub-keys in a reverse order of application[40].

The block diagram of the round function can be viewed in Fig. 5. This round function has four operations: expansion bits, XOR operation, SBOXes and permutation P. Like in the case of initial and final permutations, the operations of expansion bits, SBOXes and permutation are all defined in the official standard. In the expansion bits operation, a 32-bit word is expanded by a mapping process, generating a word of 48 bits. An operation of XOR between this expanded word and the round sub-key is performed in sequence. Then, the 48-bit word is applied to a set of eight SBOXes. Each SBOX has four rows and 16 columns, in which the addresses are defined by a 6-bit input, so the input word is divided in eight sets of addresses. The output of a SBOX is a 4-bit word defined by the address specified by the input. In this work, the SBOXes were implemented by Boolean functions in a XOR-SOP form, that presented better performance compared to implementations based on look-up table[41].

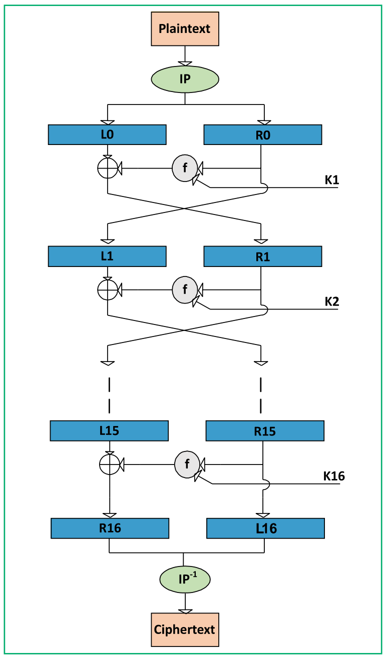

Figure 4: Sequence of digital encryption standard algorithm operations.

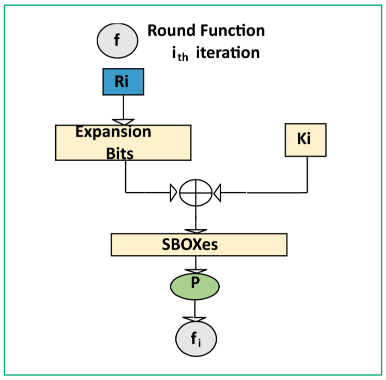

Figure 5: Block diagram of a round function of the digital encryption standard algorithm.

The key scheduling is the process in which a set of 16 sub-keys for each round are created from an original key, as shown in Fig. 6. The original key of 64 bits is permuted in the beginning of the process, as labeled in PC1, and then divided into two sub-sets with 28 bits each one, in which eight parity bits are discarded. At each round, the sub-sets are exposed to left shifts, from one or two positions according to the round number. After shifting, an operation of permutation is used to deliver the sub-key to the round function, as defined by PC2. The permuted choices PC1 and PC2 are also defined by standard and use only wiring operations. In a pipeline structure with hardware replication, the left-shift operations can be implemented also by means of wirings, and this is the case of this work.

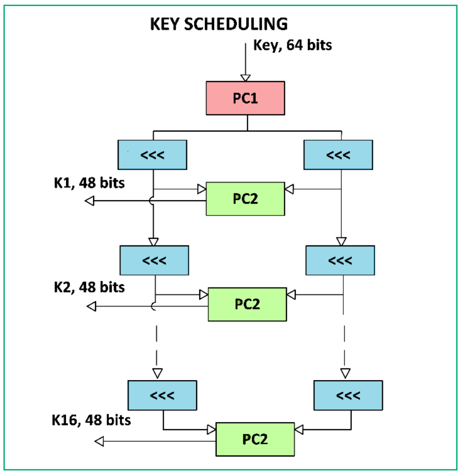

Figure 6: Sequence of operation digital encryption standard algorithm key scheduling.

PROPOSED ASYNCHRONOUS PIPELINE

The proposed bundled-data pipeline architecture is shown in Fig. 7, in which the bundled-data style is composed of N + two lines, with N lines related to the data and two lines to the request and acknowledge signals, which are used to carry out the communication. The proposal is made up of flip-flop D-based registers. At each stage, there is a data-path that is responsible for processing the data. For each register, there are an XNOR gate and a control that is an asynchronous finite state machine (AFSM). The XNOR gate allows the registers to be activated at both edges of the signal. Between the stages, there is a delay element that is defined by the critical path of the data-path, *i.e.*, the propagation time of the data-path. The AFSM is responsible for the communication between the stages, through the two-phase handshaking protocol, that is, using input and output request signals [*Ri, Ro*] and input and output acknowledgment signals [*Ai, Ao*]. The AFSM and delay elements ensure synchronization of pipeline operations.

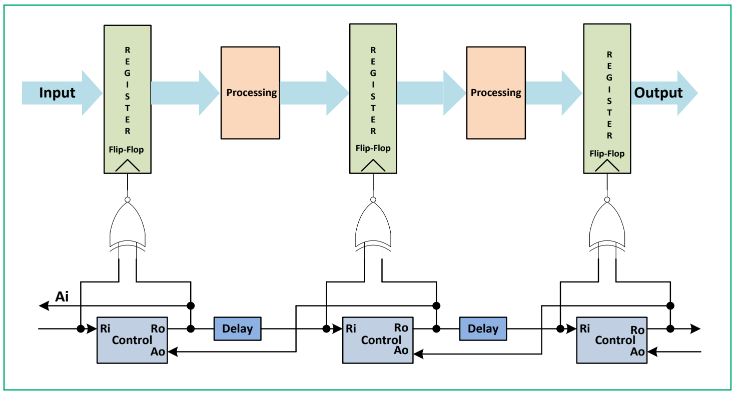

Figure 7: Proposed linear pipeline architecture.

Figure 8 shows the specification of the proposed control that was described in signal transition graph (STG). The STG specification is an interpreted Petri net and was proposed by Chu[42]. The STG of the proposed control for the input signals are [*Ri, Ao*], and the output signal is [Ro]. The control synthesis involves two steps. In the first step, the state graph (SG) is generated, and the property complete state coding (CSC) is checked to see if it is satisfied; if not, state signals must be inserted[43]. An SG satisfies the CSC property if every pair of different states which are assigned the same binary code enables exactly the same set of non-input signals.

Figure 9a indicates the SG of the control, which satisfies the CSC property, so it is in conditions to enable the implementation. The second step is to obtain the output equation *Ro*, which must be hazard free. Figure 9b shows the extraction of the Ro signal through the Karnaugh map. The equation of the signal Ro can be implemented with a C-element[43] and an inverter gate, as shown in Figs. 9c and 9d. Figure 10 presents the proposed asynchronous pipeline architecture.

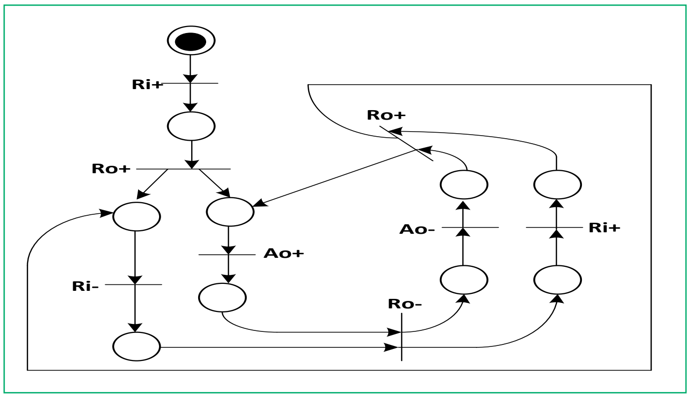

Figure 8: Signal transition graph specification of control.

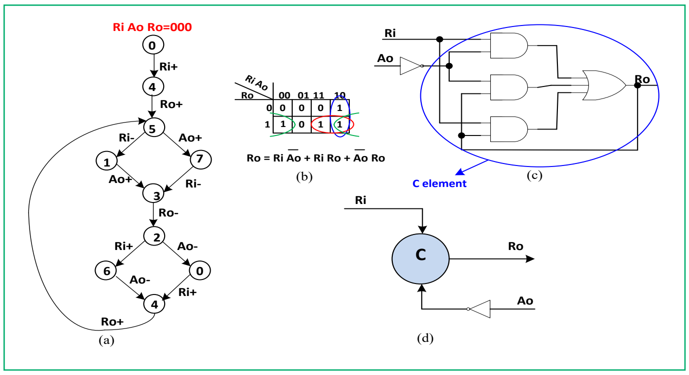

Figure 9: Synthesis: (a) state graph; (b) map of Karnaugh for Ro signal and covered; (c) logic circuit; (d) logic circuit based in C-element.

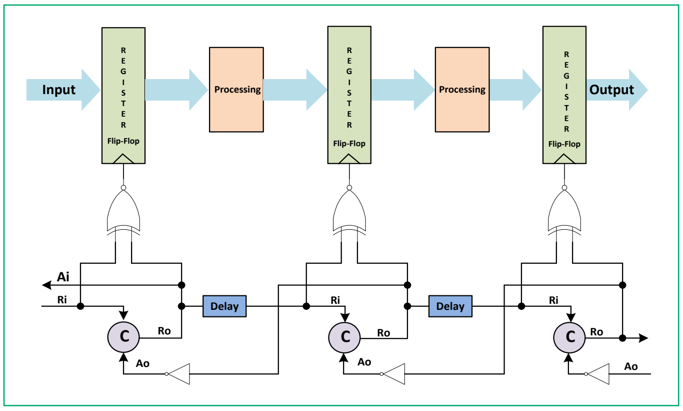

Figure 10: Proposed linear pipeline architecture.

DATA ENCRYPTION STANDARD ENCRYPTION: ASYNCHRONOUS HARDWARE DESIGN

The DES_Encryption algorithm was decomposed into eight stages, as shown in Fig. 11. The novelty is the Key Block proposal, as it appears in Fig. 12. In the case of encryption, the displacements are for the left, and in the decryption, they are done to the right. It ensures that the keys are applied in the opposite direction[13].

The keys of the previous round enter the Key Block, suffer displacements, which are basically repositioning the wires, and multiplexed according to the selection signal MODE (0 = Enc, 1 = Dec), and pass through a permutation box called PC2, which also is implemented by means of wire repositioning. From this permutation box, comes the key to be applied in the function of the round. Figure 13 shows the eight-stage asynchronous pipeline of the DES algorithm, which follows the proposed decomposition presented in Fig. 11.

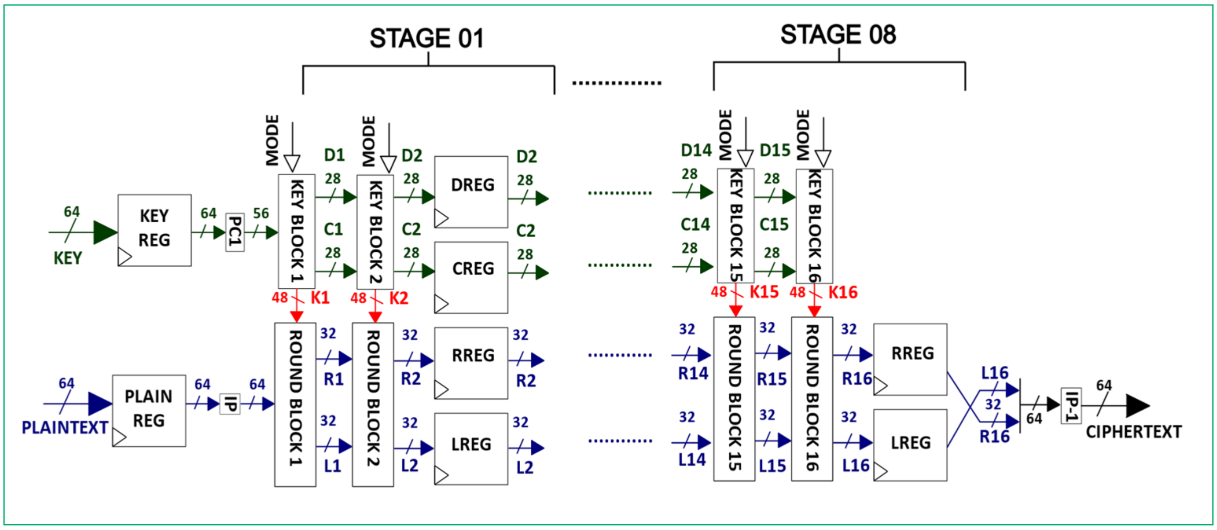

Figure 11: Decomposition proposal of digital encryption standard algorithm.

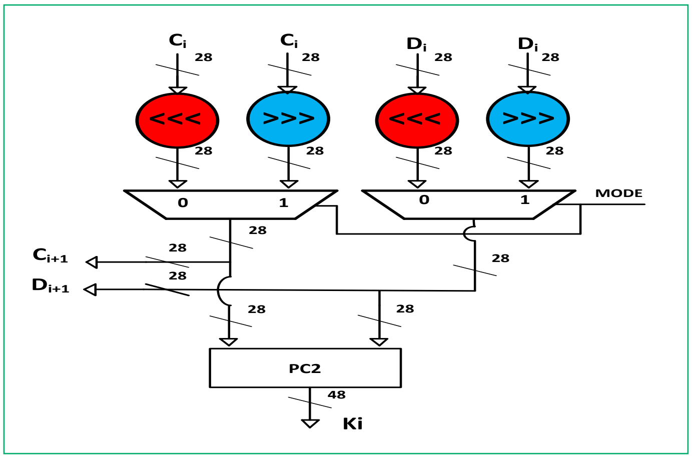

Figure 12: Proposed scheme for the Key Block.

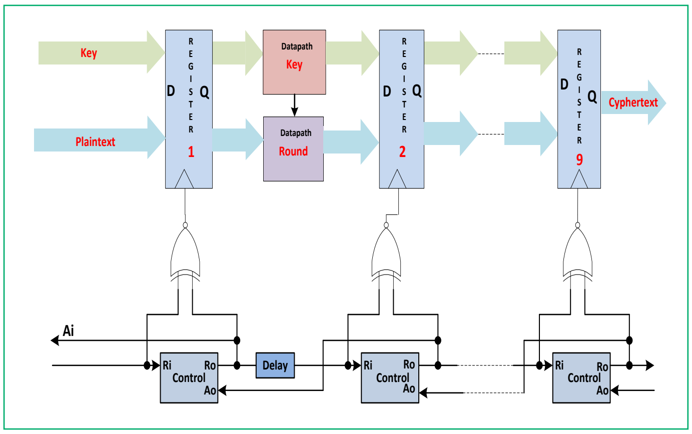

Figure 13: Proposed bundled-data asynchronous pipeline of eight-stage for digital encryption standard encryption algorithm.

SIMULATIONS AND RESULTS

To show the feasibility of the proposed project, the DES encryption algorithm was synthesized in three styles:

- synchronous of four stages;
- multi-point GALS[25], that consists of four modules;
- proposed asynchronous pipeline of eight stages.

The three projects were described in structural very high-speed integrated circuits hardware description language (VHDL), compiled and synthesized in post-layout in Intel Altera® tool, Quartus II software, version 9.0, Cyclone III family, in EP3C16F484C6[44] device.

Simulations

Figure 14 shows the simulation of the asynchronous pipeline of encryption DES algorithm (DES_E) with initial data and the simulation of DES_E algorithm after nine requests (REQ signal) operating on the two-phase protocol. The waveforms were exactly the expected for the DES_E.

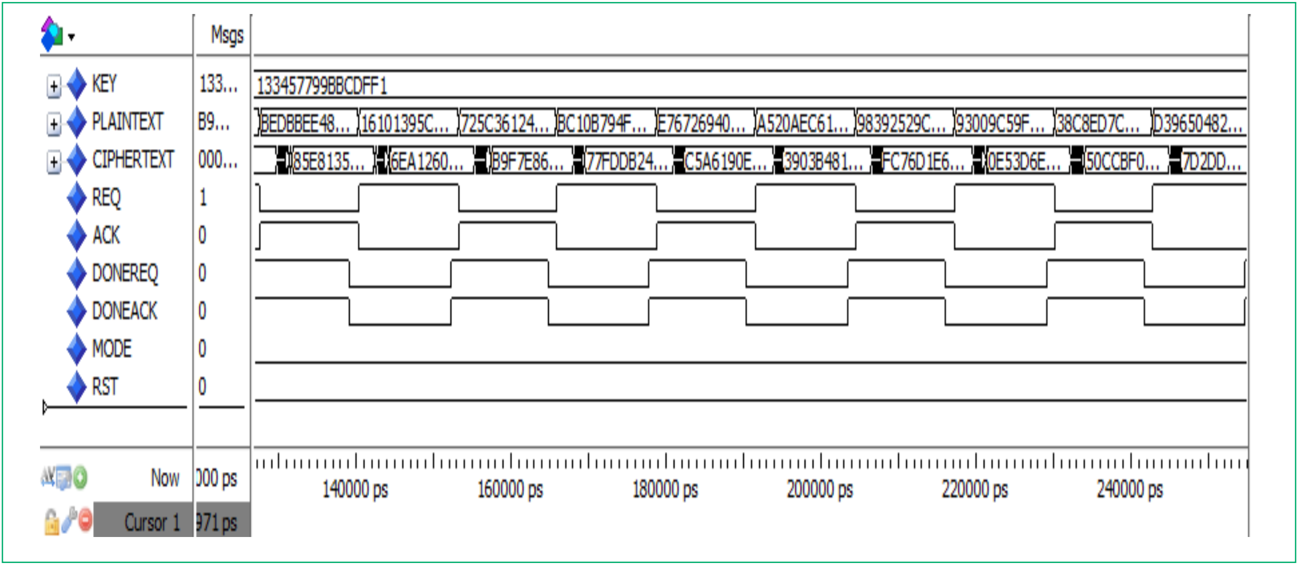

Figure 14: Simulation: asynchronous pipeline of digital encryption standard algorithm.

Results

Table 1 presents the results of the implementation of the DES algorithm in the versions of synchronous pipeline, multi-point GALS[25] and proposed asynchronous pipeline. Comparing the three styles, the proposed pipeline architecture had the average penalty of 166.9 and 7%, respectively, in dynamic power and area (LUTs – Look-Up-Table + FFs), when compared to the other two styles. The proposed architecture had the average reduction of 66.3% in latency time and the average increase in throughput (MOPS – 106 operations per second) of 14.9% when compared with two other styles. The high-dynamic power penalty of our proposal is related to the high number of LUTs used; this is due to the delay elements and the type of functional units.

Table 1: Results of the design styles of DES_E algorithm.

Design styles	Latency	Throughput MOPS	Dynamic power	Macrocells	
				LUTS	Flip-Flops
Synchronous pipeline (fMAX = 125MHz)	268 ns	100.0	137.97 mW	**4,604**	1,866
Multi-poing GALS from Curtinhas et al.[25]	189 ns	83.3	**46.41 mW**	4,926	2,095
Asynchronous pipeline proposed	**75.8 ns**	**105.3**	246.06 mW	6,192	**1,024**

GALS: globally asynchronous and locally synchronous; LUTS: look-up-table.

CONCLUSION

This work proposed a data encryption standard FPGA-based implementation according to the device Altera Cyclone III family, in EP3C16F484C6, using an 8-stage asynchronous pipeline design style and a sub-key multiplexing system for the encryption/decryption modes. At each clock cycle, a new key and a new plaintext can be processed, and at each new cycle of clock, after the pipeline fulfillment, a new valid ciphertext can be obtained in the output. Furthermore, the system can be switched between the operation's mode of encryption and decryption by means of a single signal, which is used to select between two sets of sub-keys to be applied at each round, one for encryption and the other one for decryption. The implementation uses 6192 LUTs and 1024 D flip-flops, with latency time of 75.8 ns and throughput of 105.3 MOPS. This encryption/decryption rate makes this architecture suitable to applications like smart cards, satellite communication and aerial drones, while the flexible change of key can be used in a key-search machine for the DES algorithm.

REFERENCES

1. Adee S. The hunter for the kill switch. IEEE Spectrum. 2008;45(5):34-9. https://doi.org/10.1109/MSPEC.2008.4505310

2. Lussari E, Oliveira DL, Faria LA, Verducci O. Software-defined radio design based on GALS architecture for FPGAs. In: IEEE 29th Symposium on Integrated Circuits and Systems Design (SBCCI); 2016. p. 1-6. https://doi.org/10.1109/SBCCI.2016.7724076

3. Ozmen MO, Yavuz AA. Dronecrypt - an efficient cryptographic framework for small aerial drones. In: IEEE Military Communications Conference (MILCOM); 2018. p. 1-6. https://doi.org/10.1109/MILCOM.2018.8599784

4. Kocher P. Timing attacks on implementations of diffie-hellman, RSA, DSS, and others systems. In: 16th International Cryptology Conference on Advances in Cryptology (CRYPTO'96); 1996. p. 104-113. https://doi.org/10.1007/3-540-68697-5_9

5. Kocher P, Jaffe J, Jun B. Differential power analysis. In: 19th International Cryptology Conference on Advances in Cryptology; 1999. p. 388-97. https://doi.org/10.1007/3-540-48405-1_25

6. Rivest RL, Shamir A, Adleman L. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM. 1978;21(2):120-6. https://doi.org/10.1145/359340.359342

7. Wheeler DJ, Needham R. TEA, a Tiny Encryption Algorithm. In: Fast Software Encryption; 1994, v. 1008. Leuven: Springer-Verlag; 1994. p. 363-6.

8. Advanced Encryption Standard (AES). Federal Information. Processing Standards Publication 197 [Internet]. National Institute of Standards and Technology; 2001 [cited 28 Oct 2020]. Available from http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

9. Digital Encryption Standard (DES). Federal Information. Processing Standards Publication 46-2 [Internet]. National Institute of Standards and Technology; 1993 [cited 28 Oct 2020]. Available from: http://www.itl.nist.gov/fipspubs/fip46-2.htm

10. United States. National Bureau of Standard. Data encryption Standard (DES). Federal Information. Processing Standards Publication 46. Springfield: National Technical Information Service; 1977.

11. United States. National Bureau of Standard. DES modes of Operation. Federal Information. Processing Standards Publication 81. Springfield: National Technical Information Service; 1980.

12. Stinson DR. Cryptography. Theory and Practice. 2nd ed. Boca Raton: Chapman & Hall/CRC; 2002.

13. Schneier B. Applied cryptography: protocols, algorithms, and source code in C. 2nd ed. New York: Wiley & Sons; 1996.

14. Dichou K, Tourtchine V, Rahmoune F. Finding the best FPGA implementation of the DES algorithm to secure smart cards. In: 4th International Conference on Electrical Engineering; 2015. p. 1-4. https://doi.org/10.1109/INTEE.2015.7416749

15. Wong K, Wark M, Dawson ME. A single-chip FPGA implementation of the Data Encryption Standard (DES) algorithm. In: IEEE Globecom; 1998. https://doi.org/10.1109/GLOCOM.1998.776849

16. Kaps JP, Paar C. Fast DES Implementations for FPGAs and Its Application to a Universal Key-Search Machine. In: Tavares S, Meijer H, eds. Selected Areas in Cryptography. SAC 1998. Lecture Notes in Computer Science, v. 1556. Berlin / Heidelberg: Springer; 1999. p. 234-247.

17. Patterson C. A dynamic implementation of the serpent block cipher. In: Workshop Cryptographic Hardware and Embedded Systems; 2000. p. 142-55. https://doi.org/10.1007/3-540-44499-8_10

18. McLoone M, McCanny JV. A high performance FPGA implementation of DES. In: Signal Processing Systems; 2000. p. 374-83. https://doi.org/10.1109/SIPS.2000.886736

19. Abd KMA, Hamed HFA, Hasaneen EAM. FPGA implementation of the pipelined data encryption standard (DES) based on variable time data permutation. Online J Electr Electr Eng. 2011;2(3):298-302.

20. Oukili S, Bri S. FPGA implementation of Data Encryption Standard using time variable permutations. In: 27th International Conference on Microelectronics; 2015. https://doi.org/10.1109/ICM.2015.7438004

21. Liang D, Hongyi C. An efficient and scalable VLSI implementation of DES. In: 4th International Conference on ASIC; 2001. p. 341-3.

22. O'Melia S, Elbirt AJ. Enhancing the performance of symmetric-key cryptography via instruction set extensions. IEEE Transactions on Very Large Scale Integration (VLSI) Systems. 2010;18(11):1505-18. https://doi.org/10.1109/TVLSI.2009.2025171

23. Zhang Q, Cao J, Cao X, Zhang X, Ye Y, Zhao Y, et al. Optimization design of a low power asynchronous DES for security applications based on Balsa and synchronous tools. In: International Conference on Electronics, Communications and Computers; 2015. p. 124-9. https://doi.org/10.1109/CONIELECOMP.2015.7086938

24. Soares R, Calazans N, Moraes F, Maurine P, Torres L. A robust architectural approach for cryptographic algorithms using GALS pipelines. IEEE Design Test Computers. 2011;28(5):62-71. https://doi.org/10.1109/MDT.2011.69

25. Curtinhas T, Oliveira DL, Saotome O, Brandolin JB. FPGA implementation of low-latency robust asynchronous interfaces for GALS systems. In: IEEE XXV International Conference on Electronics, Electrical Engineering and Computing; 2018. p. 1-4. https://doi.org/10.1109/INTERCON.2018.8526456

26. Silva DA, Verducci O, Oliveira DL. Implementation of DES algorithm in new non-synchronous architecture aiming DPA robustness. In: IFIP/IEEE 27th International Conference on Very Large Scale Integration (VLSI-SoC); 2019. p. 1-2. https://doi.org/10.1109/VLSI-SoC.2019.8920302

27. Brodie I, Schwoebel PR. Vacuum Microelectronic Devices. Proc IEEE. 1994;82(7):1006-34. https://doi.org/10.1109/5.293159

28. Ismailoglu N, Vargün B, Sever R, Okcan B, Karakus K, Kapucu K, et al. GEZGIN-2: image processing subsystem of RASAT. In: 5th International Conference on Recent Advances in Space Technologies; 2011. p. 944-9. https://doi.org/10.1109/RAST.2011.5966981

29. Wu H, Chen W, Su Z, Wei S, He A, Chen H. A method to transform synchronous pipeline circuits to bundled-data asynchronous circuits using commercial EDA tools. In: IEEE International Conference on Electron Devices and Solid-State Circuits; 2019. p. 1-2. https://doi.org/10.1109/EDSSC.2019.8754234

30. Kocher PC. Timing attacks on implementations of diffie-hellman, RSA, DSS, and other systems. In: Koblitz N, ed. Advances in Cryptology. Lecture Notes in Computer Science, v. 1109. Berlin / Heidelberg: Springer; 1996. p. 104-113.

31. Lellis R, Soares R, Souza AA. An energy-based attack flow for temporal misalignment countermeasures on cryptosystems. In: IEEE International Symposium on Circuits and Systems; 2017. p. 1-4. https://doi.org/10.1109/ISCAS.2017.8050844

32. Myers CJ. Asynchronous circuit design. 2nd ed. New York: Wiley & Sons; 2004.

33. Sparsø J, Furber S. Principles of asynchronous circuits design. Dordrecht: Kluwer Academic Publishers; 2001.

34. Martin J. The limitations to delay insensitive in asynchronous circuits. In: 6th MIT Conference on Advanced Research in VLSI Processes; 1990. p. 263-77. https://doi.org/10.1007/978-1-4612-4476-9_35

35. Sutherland IE. Micropipelines. Communication of the ACM. 1989;32(6):720-38. https://doi.org/10.1145/63526.63532

36. Singh M, Nowick SM. MOUSETRAP: high-speed transition-signaling asynchronous pipelines. IEEE Trans VLSI Systems. 2007;15(6):684-98. https://doi.org/10.1109/TVLSI.2007.898732

37. Singh M, Nowick SM. The design of high-performance dynamic asynchronous pipelines: lookahead style. IEEE Trans VLSI Systems. 2007;15(11):1256-69. https://doi.org/10.1109/TVLSI.2007.902205

38. Prabakar TN, Lakshminarayanan G, Anilkumar KK. FPGA Based Asynchronous Pipelined Multiplier with Intelligent Delay Controller. In: International SOC Design Conference; 2008. p. 304-9. https://doi.org/10.1109/SOCDC.2008.4815633

39. Oliveira DL, Garcia K, Santana L, Faria LA. FPGA implementation of high-performance asynchronous pipelines with robust control. In: 9th IEEE Latin American Symposium on Circuits and Systems; 2018. p. 1-4. https://doi.org/10.1109/LASCAS.2018.8399923

40. Stinson DR. Cryptography. Theory and Practice. 2nd ed. Boca Raton: Chapman & Hall/CRC; 2002.

41. Wong K, Wark M, Dawson ME. Single-chip FPGA implementation of the Data Encryption Standard (DES) algorithm. In: IEEE Globecom; 1998. p. 827-32. https://doi.org/10.1109/GLOCOM.1998.776849

42. Chu TA. Synthesis of self-timed VLSI circuits from graph-theory specifications [PhD thesis]. Cambridge: Massachusetts Institute of Technology; 1987.

43. Beerel P, Ozdag R, Ferretti M. A designer's guide to asynchronous VLSI. Cambridge: Cambridge University Press; 2010. 337 p.

44. Altera Corporation. Portal [Internet]. Altera; 2020 [cited 28 Oct 2020]. Available at: www.altera.com