Design of an AES Device as Device Under Test in a DPA Attack

Septafiansyah Dwi Putra, Ma'muri, Sarwono Sutikno, Yusuf Kurniawan, Adang Suwandi Ahmad
(Corresponding author: Septafiansyah Dwi Putra)

School of Electrical Engineering and Informatics, Bandung Institute of Technology
Jl. Ganesha No.10, Lb. Siliwangi, Coblong, Kota Bandung, Jawa Barat 40132, Indonesia
(Email: sept4.182@s.itb.ac.id)

(Received May 20, 2017; revised and accepted Aug. 27 & Oct. 7, 2017)

Abstract

This paper presents a design for the implementation of the AES encryption algorithm in the hardware system. The proposed device is intended to be a device under test in a differential power analysis (DPA) attack. This device uses AES encryption with 128bit key length and electronic codebook (ECB) mode. The platform used in this device is FPGA- Cyclone IV EP4CE115F29C7. AESAVS is used to test the functionality of the device. This study proposes a design for an AES-128 encryption device synthesized in the Quartus IDE. It will feature support conducting side-channel attacks on real condition.

Keywords: AES128; DPA Attack; Side-Channel Analysis

1 Introduction

Information is a strategic resource that needs to be protected to ensure its safety and security [12]. One method of providing information security is encryption, which is done using a cryptographic algorithm [10, 27]. The decryption of the encrypted information is then done using the key, which is available only to the intended audience of the information. Without the key, decrypting well-encrypted information would be impossible. Any cryptographic algorithm relies on the secrecy of its key [20]. If the secret key is known to parties other than the intended recipient, then the security provided by the cryptographic algorithm would be compromised [6]. Without the key, decrypting well-encrypted information would be impossible.

An attack on an encryption algorithm is an attempt to decrypt encrypted information without the key. Attacks also search for and exploit the weaknesses of the encryption algorithm. Over the past few decades, there have been numerous studies on cryptography and encryption algorithm attacks. These studies resulted in the development of algorithms that provide high data security and authenticity such as RSA, ECC, AES, TDES and DSA [22]. Although the security of cryptographic algorithms on the mathematical level has already been achieved, maintaining the security of cryptographic algorithms in their physical implementation remains a major challenge.

Attacks on cryptographic algorithms initially exploited only the mathematical design weaknesses of algorithms [1]. It was assumed that if a cryptographic algorithm was mathematically secure, then its implementation was also secure. This paradigm changed when Kocher published papers on timing attack [18], and power analysis attack [19]. These papers revealed that an electronic device that implements a mathematically secure cryptographic algorithm can still divulge certain information. This leakage of information, referred to as a side-channel, can be used to find the encryption key. Modern cryptographic algorithms cannot be significantly attacked by using brute force methods and finding mathematical weaknesses in the algorithm [21]. As such, side-channel information such as timing information, power consumption, electromagnetic leaks or even sound/acoustic leaks are additional sources of information that can be used to decrypt the encrypted information. Some cryptanalysis attack techniques require technical knowledge of the internal operation of the cryptographic system that is implemented [2, 13, 16, 29, 30]. But DPA and statistical methods are amongst the most powerful techniques in solving complex cryptographic algorithms [11]. In addition to reviewing the aspects of the attack, researchers have conducted various software and hardware countermeasures against side-channel analysis (SCA). On the software level, these measures include transformation and masking of data [7]. On the hardware level, measures include desychronization and noise generation [25]. These countermeasures generally require high performance and have high costs; as such, they are currently still deemed unsuitable for embedded systems [17, 26].

Currently, there is a pressing need for an electronic communications system that is both fast and secure. One of the fundamentals s in establishing fast and secure com-

munication is the use of cryptographic algorithms in the form of a device or devices. This paper presents the design of an AES encryption device as a device under test (DUT) for differential power analysis attacks to be carried out at a later stage. Prototype hardware was developed to perform AES encryption on an FPGA platform with a 128-bit data width that is compatible with the processor embedded systems. The implementation is optimized in terms of the use of resources at an acceptable rate, so that the target data security can be achieved with limited resources. This study was developed during previous studies [23, 24]. and uses datasets provided in other studies. The use of datasets presents some limitations, especially in the development of countermeasures against attacks. So, there is a need for future studies of this device as a DUT to develop adaptive countermeasures.

2 Related Research

2.1 Comparison of AES Implementation

Implementation of an encryption algorithm in hardware tends to be faster than in software [2]. Reconfigurable hardware is a practical solution for implementing cryptographic algorithms in embedded systems and high-speed applications [3]. In using FPGAs as a computing platform, there are two main goals which are often conflicting: maximizing the efficiency of resources and maximizing processing speed. As such, the precise goal of the particular implementation needs to be identified before implementing the algorithm [4].

AES can be implemented with other encryption algorithms. Its standardized algorithms enable its adoption to other applications in general. AES is a symmetric cipher block algorithm, a type of encryption algorithms with the ability to encrypt data at high speeds without sacrificing the level of security [5]. In addition, AES is more efficient than public key algorithms for encrypting large amounts of data [8, 9]. In the implementation of AES, the absolute confidentiality of the encryption key needs to be maintained to ensure the security of the encrypted information. Implementations of AES must also have excellent performance on different applications and platforms, maintaining such qualities as: high speed, low latency values, low area usage, and low power usage. As such, software and hardware designs for AES are continually being developed.

In this paper, FPGA is chosen as the platform for implementing AES. This choice is based on the information in Table 1 which shows a comparison between three AES implementations: ASIC, FPGA, and software. FPGA has advantages over software implementations based on its capability in parallel processing, pipelining, velocity, and resistance to tampering. Meanwhile, the design process of AES in FPGA, while not as fast as software implementations, is significantly faster than the ASIC implementations.

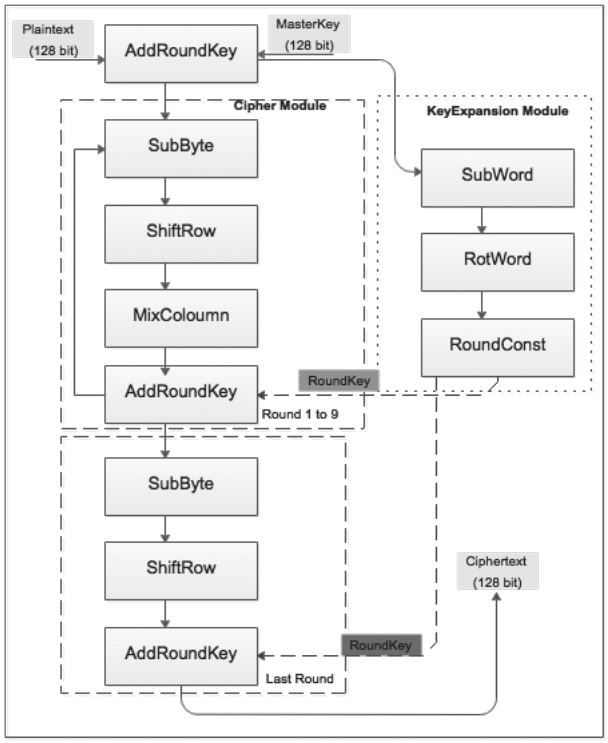

Figure 1: AES encryption algorithm structure

In general, the implementation of an AES encryption algorithm is divided into two major components: the cipher module and the key generation module.

2.2 AES Algorithm

The cipher module performs data encryption or decryption. In an AES algorithm with a 128-bit key, the cipher module does ten rounds of substitutions and permutations to encrypt the data input (plaintext). In the first nine rounds of the encryption process, the cipher module uses SubByte, ShiftRow, MixColumn, and AddRoundKey operations. In the final (tenth) round, Mixcolumn is used to complete the block encryption process. Figure 1 shows the standard structure of the AES algorithm. The Figure also shows the iterative process in combining different functions of each module with the cipher key expansion module. The input data in AES is represented as a 4 x 4 byte array and is called the state.

AddRoundKey is the initial and final function and is used to combine key information with the data under operation. The input of this function is 16 bytes of state, while 16 byte keys are obtained from the key expansion algorithm. The output value of this operation is the XOR bit between round sub key derived from master key. This function is essentially same for encryption and decryption processes. The transformation is denoted by $AddRoundKey(State, RoundKey)$.

The SubByte transformation is a non-linear byte substitution, operating on every byte of the state inde-

Table 1: Comparison of AES implementation

	ASIC	FPGA	Software Based
Parallel processing	Yes	Yes	Limited
Pipelining	Yes	Yes	Limited
Velocity	Very fast	Fast	Moderate
Resistance to tamper	Strong	Strong	Weak
Design process	Long time	Moderate	Fast
Field reprogramability	Moderate	Yes	Moderate
Lower Unit Cost	Yes	Moderate	Yes

pendently. The substitution table (or S-box) is invertible Each input of byte is independently replaced by another byte from lookup table called substitution box (S-box). There are 16 parallel S-boxes, each consisting of 8 input and 8 output values. The S-box operation is the only non-linear transformation in the AES algorithm. The transformation is denoted by $SubByte(State)$. This operation is reversible and is also used in the decryption process. The S-box design is achieved by combining two transformations. The first transformation is done by taking the inverse multiplication in the finite field $GF(2^8)$ where all the zero bit inputs are mapped to themselves. In the second part, the affine transformation is done over $GF(2)$.

$$M = \begin{Bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{Bmatrix}, C = \begin{Bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{Bmatrix}$$

ShiftRow function, the number of rows in the State are cyclically shifted over different offsets. In this function, the first line is not altered, but the second, third, and fourth lines are rotated by one, two, three bytes, respectively. The shifting the rows of the State over the specified offsets is denoted by:$ShiftRow(State)$. The MixColumn perform transformation each coloumn of the state matrix multiplied by a constant fixed matrix. The application of MixColoumn operation on all columns of the State is denoted by $MixColumn(State)$. This function can be written into a matrix multiplication as:

$$\begin{bmatrix} MC'_{0,c} \\ MC'_{1,c} \\ MC'_{2,c} \\ MC'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} MC_{0,c} \\ MC_{1,c} \\ MC_{2,c} \\ MC_{3,c} \end{bmatrix}$$

2.3 Key Generation Module

The key generation module are derived from the MasterKey by means of the key schedule. This consists of two main components: the MasterKey Expansion and the SubKey Selection. The key generation algorithm is as follows:

```
1: KeyExpansion(byte key[4*Nk], word w[Nb*(Nr+1)], Nk)
2: Begin
3: word temp
4: i=0
5: while i < Nk do
6:    w[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
7:    i = i+1
8: end while
9: i=Nk
10: while (i < Nb * (Nr+1)) do
11:    temp = w[i-1]
12:    if (i mod Nk = 0) then
13:       temp = SubWord(RotWord(temp))  Rcon[i/Nk]
14:    else if (Nk > 6 and i mod Nk = 4) then
15:       temp = SubWord(temp)
16:    end ifw[i] = w[i-Nk]  temp
17:    i = i + 1
18: end while
19: End
```

The KeyGeneration is a linear array of 4-byte words and is denoted by $W[Nb*(Nr+1)]$. The first Nk words contain the MasterKey. All another words are defined recursively in terms of words with smaller indices. The key expansion function depends on the value of Nk. This function consists of three sub-modules: SubWord, RotWord, and RCon. SubWord is is a function that returns a 4-byte word in which each byte is the result of applying the AES S-box to the byte at the corresponding position in the input word. RotWord takes a cyclic permutation of those in its input such that the input $word(a,b,c,d)$ produces the output $word(b,c,d,a)$. The ratio between key length and block size in terms of words for different types of AES is shown in Table 2.

2.4 Device Optimization

Requirements of throughput, power and design need to be met by AES algorithm designers. As such, AES devices need to be optimized in order to meet these requirements. Most approaches to AES design optimization can

Table 2: Comparison of key length, block size, and AES round

AES Type	Key Length	Block Size	Round
AES 128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

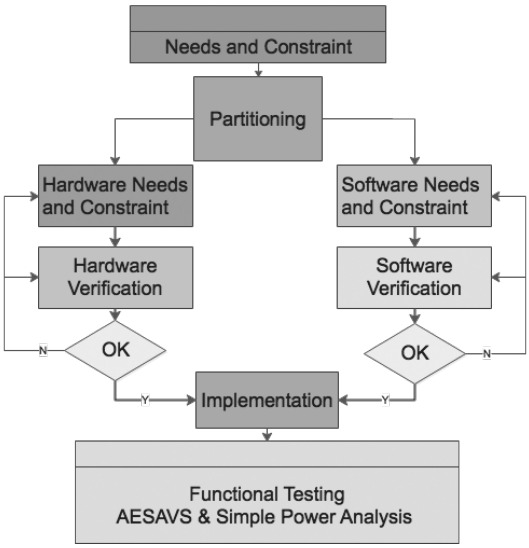

Figure 2: Design of AES device methodology

be generally divided into two: algorithm optimizations and architecture optimizations.

Algorithm optimizations use the fact that AES is based on finite fields operations. Thus, the choice that represents the finite field and composite fields is the use of isomorphisms in developing efficient and compact design [17]. Furthermore, an AES round has properties which allow the encryption and decryption processes to be done using the same method. Platforms such as the FPGA provide enables efficient implementation of some of the AES round transformations.

Meanwhile, standard architectural optimization techniques such as pipelining can also be applied to improve the hardware design throughput of AES [11]. The separation of data and key generation scheduling is helpful in improving the efficiency of an AES design [13]. This optimization technique can support all three versions of AES, namely 128.192, and 256 bits [14].

3 Methodology

This paper presents the stages involved in designing an AES encryption device. The proposed method is shown in Figure 2.

The initial stage is reviewing the aims and limitations for designing the AES cryptographic device. The aim of this research is to create an AES device implemented on an FPGA platform which will be a simulated DUT device in a DPA (differential power analysis) test. The restriction on this device is AES with 128-bit key length and an input value message with 128-bit length.

The next design stage is categorizing the identified aims and limitations into hardware and software. The initial approach is to generate a Verilog pseudocode and RTL schematic of the desired circuit in simulation software such as ModelSim. The resulting scheme can be verified using simulation software that shows the input and output waveforms of the circuit. The output is the Verilog code and a RTL schematic design for an AES cryptographic algorithm to be verified on FPGA hardware. After the aims and limitations have been verified on both hardware and software, the next step is implementation and functionality testing.

The implementation and testing process begins with designing and conducting an AES cryptographic algorithm testbench using FPGA software in Verilog. The testbench process checks whether the device gives the correct output, given a standard input. After passing the testbench, the design is ready to be implemented on an FPGA platform. The end point of the FPGA prototype is that when a Verilog model is decoded into "gates and cables" are mapped onto a programmable logic device such as an FPGA. Tests were conducted to determine the success rate of the system that was developed. Three stages of testing were conducted:

- Functional testing of each system block.
- System-wide functional testing.
- Hardware performance measurement.

The functional testing of the encryption device was done using the Advanced Encryption Standard Algorithm Validation Suite (AESAVS). AESAVS is designed to test compliance with the NIST FIPS197 standard. This testing standard provides a security measure for an AES product. Test validation was performed to assist in detecting any errors of unintended implementation [14]. So this validation technique should not be interpreted as an evaluation or overall product security support. The AESAVS testing method has the following philosophy design:

1) AESAVS is designed to allow testing of implementation under test (IUT). AESAVS and IUT exchange data via request and response file.

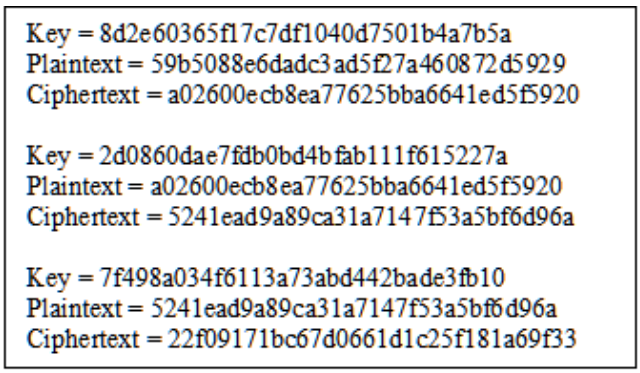

Figure 3: Example of output requirements and monte carlo test input

2) Testing is done in AESAVS using statistical sampling.

Devices are required to pass AESAVS to claim compliance to the FIPS 197 standard on AES and FIPS 140-2 standard on the security requirements for cryptographic modules.

AESAVS uses two approaches; namely, the known answer test (KAT) and Monte Carlo test (MCT). KAT tests KeySbox, Variable Key and Variable Text. MCT tests the cipher with as many as 100 pseudorandom texts. These texts are generated using an algorithm related to the mode of operation under test. An example of the three initial outputs of MCT from AES128 is shown in Figure 3.

4 AES Implementation

This study uses Verilog language as the hardware description language because of its flexibility; Verilog codes can be easily implemented in other devices without design changes. The hardware used for this research is Altera DE2 board. This board tool is used for writing, debugging, optimizing, performing simulations, and checking performance results using the simulation tools available on the Quartus IDE design software.

The results of this study are based on simulations of Altera DE2 and Quartus tools, using analysis and synthesis, RTL viewer, and state machine viewer. A design iteration method is implemented to minimize hardware utilization and adaptation performed on Altera Quartus.

Figure 4 shows a block diagram of the hardware implementation. In the block diagram is also mentioned the encryption and decryption module and key generation module.

The system clock represented by *clk* sets the crypto-processor frequency. The clock frequency dictates the operating speed of the system. In this design, the clock has two main purposes: system controller and bus or line controller. The reset function *rst* enables a return of the system to the initial condition. This function is active in low or 0 condition. At the key generation module initialization, the *start* signal states the time at which the clock starts. The *decrypt* function, one of the main functions of this system, transforms ciphertext into plaintext. This function can be changed into encryption mode (on which the function transforms plaintext into ciphertext) by changing the input value of En (encryption mode when En = 1, decryption mode when En =0). Additional input values are the 128-bit key and din which are the key and text input, respectively. These input values produce 128-bit *dout*, the same size as the input.

The key generation process is shown in Figure 6. This process generates the round keys from the master key; these round keys are only in the encryption rounds. The clk function is the input in key generation; the generated keys are stored in the internal ROM and read by encryption and decryption block for each round. The encryption/decryption module accepts a 128-bit plaintext or ciphertext input when the decrypt is off ($En = 0$). The key generation process accepts 128-bit key input, divided into 4 sections (ki_0, ki_1, ki_2, ki_3) of 32bit. In this design, more than one register is used in each round, producing ten clocks efficient for key generation.

The system design architecture is shown in Figure 7. There are at least three connected components: AES crypto-processor, personal computer, and digital sampling oscilloscope (DSO). The crypto-processor is the DUT from which side channel information would be harvested by the DSO, creating a traces curve. The PC collects the traces and performs statistical analyses to find the key by modeling the traces curve using key guesses. The DUT and the oscilloscope communicate using USB and RS232.

The results shown in Table 1 illustrate the condition of the implementation of the AES encryption algorithm on the Cyclone IV E FPGA device. Generally, the required element of this device is 8% of the total device. This shows that this system and platform can be applied to other solution problems. The small memory usage of $< 1\%$ and the required registers indicates that the AES device design meets excellent performance standards on various application forms and platforms, as well as high speed and low latency values.

The use of a small amount of area and a compact design is shown in Figure 8. This shows a slice of the usage of the FPGA EP4CE115F29C7 processor. For a small to medium sized processor, the slice usage is extremely small at 8%. This shows a potential for implementing KGS (knowledge growing system) as countermeasures for DPA/SCA.

5 DUT Testing

The AES encryption tool is built in the ECB operation mode and uses Verilog as a programming language. Description and verification was done using a ModelSim simulator - Intel FPGA Starter 10.5b edition. Quartus software performs logic synthesis, mapping, placing, and

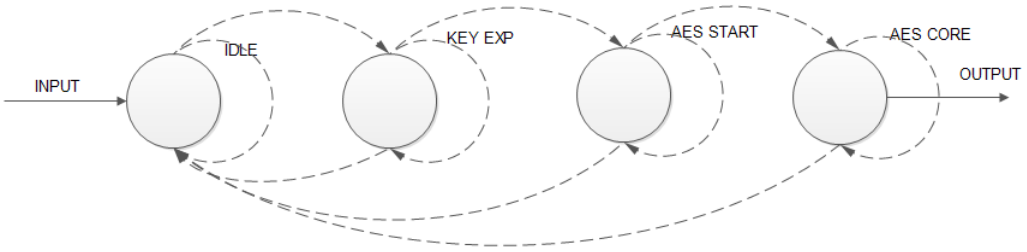

Figure 4: Finite-state of AES device

Table 3: Result comparison testing

Testing Component	Iteration Length	Encrypt/Decrypt Pass	Time
AESAVS VarKey test data for ECB AES128	128	Yes	92505 ps
AESAVS KeySbox test data for ECB	21	Yes	15465 ps
AESAVS VarTxt test data for ECB	128	Yes	92505 ps
AESAVS MCT test data for ECB	128	Yes	28044345 ps

Name	Flow	Width	
clk	in	1	System clock
rst_n	in	1	Reset, active LOW
start	in	1	Start signal, 1clock active HIGH
decrypt	in	1	1: Decrypt mode, 0: Encrypt mode
key	in	128	Key input
din	in	128	Data input
busy	out	1	Busy signal, active HIGH
dout	out	128	Data output

Figure 5: Block diagram of resulting AES device

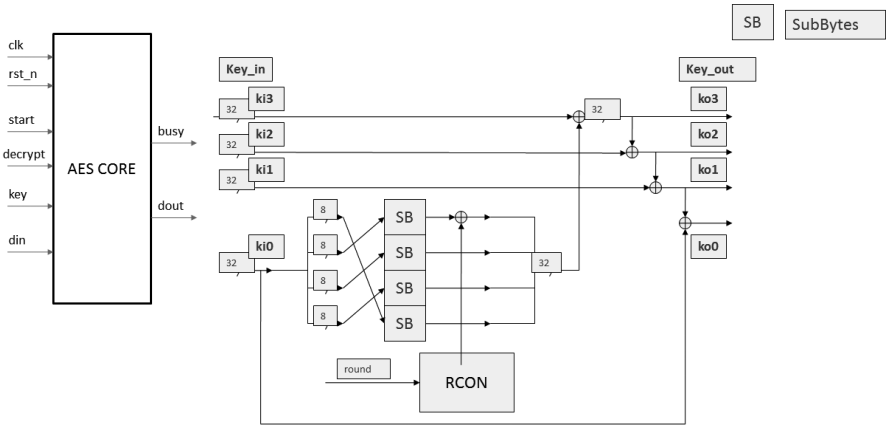

Figure 6: Key generation process

routing; Verilog pseudocodes are used for the input of this process. All built-in circuits are contained in a single FPGA device, with Quartus software determining the optimal location for a compact design. Table 4 shows the results of AESAVS testing compare with another AES design.

AESAVS test results through the KAT test (known answer test) and the MCT (Monte Carlo test) have shown results that meet the NIST FIPS197 standard and does not detect implementation errors. This validation tool has been evaluated and complies with NIST's AES FIPS197 standard.

Table 4 provides a comparison between several commonly used AES standards: Sasebo [14], SakuraX [15], and the design proposed in this paper. It shows that the proposed communication interface standard has a higher flexibility than other devices. This flexibility gives more opportunities for researchers to develop countermeasures to attacks. The proposed device has a size of 90nm in design technology, which is still larger than Sakura-X whose value is 28 nm. However, this value affects neither performance nor speed. It only affects the area/wafer of the device.

This study also performs synthetic analysis on three devices: AESFPGA, asic.ws, and our proposed design. Our proposed design has advantages over the other designs in total register, pins, and memory bits. Our proposed DUT has 277 registers, which is smaller than that of the other devices. Our proposed device uses 389 input and output pins, which is better than the AESFPGA design standard. The most important factor affecting the speed and performance of the proposed DUT is the usage of memory bits as cache. This results in a much faster speed than the other two designs.

Besides testing with KAT, we also do trace sampling analysis with SPA (simple power analysis). The purpose of this test is as a first step DPA attack techniques. Fig-

Table 4: Comparison our proposed with FPGAs as device under test (DUT)

Board	Control FPGA	Tech	Host Data Communication	Status
SASEBO [14, 15, 28]	Virtex-2 Pro	130 nm	RS232	Discontinued
SASEBO-G [14, 15, 28]	Virtex-2 Pro	130 nm	RS232,FT245RL(USB)	-
SASEBO-GII [14, 15, 28]	Spartan-3A	65 nm	FT245RL	Discontinued
SASEBO-B [14, 15, 28]	Stratix-2	90 nm	RS232,FT245RL(USB)	-
Saxura-X [14, 15]	Spartan-6	28 nm	USB	-
Our Proposed	Altera Cyclone II	90 nm	RS232,FT245RL(USB),JTAG Mode	-

Table 5: Synthesis result of AES device

Component	**Our Proposed**	AESFPGA1 [14]	ASIC.WS
Revision Name	AES_CORE	SASEBO	ASIC
Top-level Entity Name	AES_CORE	SASEBO	AES ASIC
Total logic elements	9,153 / 114,480 (8%)	9,023	4,940
Total registers	277	396	530
Total pins	389 / 529 (74%)	392 /529	388
Total virtual pins	0	0	0
Total memory bits	2,048 / 3,981,312 (1%)	0	0
Embedded Multiplier 9-bit elements	0 / 532 (0%)	0	0
Total PLLs	0 / 4 (0%)	0	0

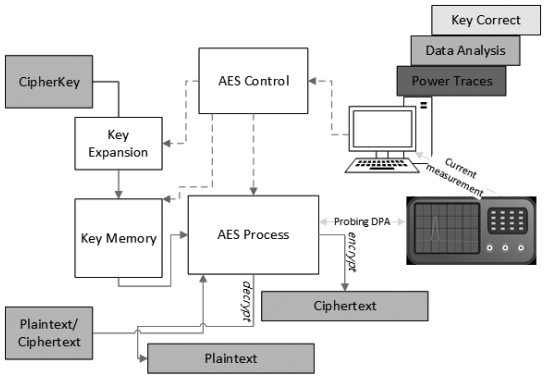

Figure 7: Architecture of proposed DUT

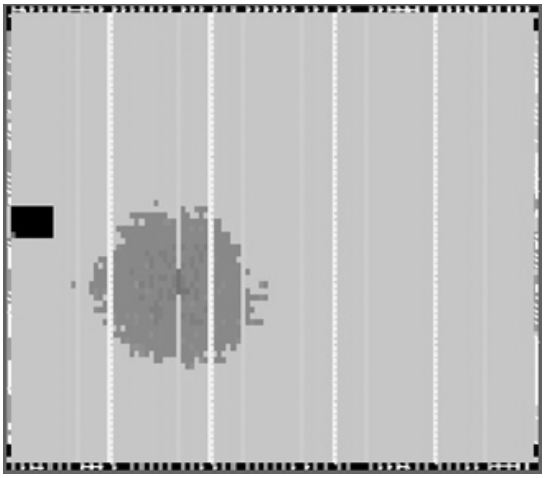

Figure 8: Use of AES slice area generated

ure 9 shows approximately 1 ms of a trace collected from a DUT performing an AES-128 encryption operation. The power consumption was sampled at 100 MHz. The trace were captured by placing a resistor in series with the devices ground line in accordance with architecture DUT Figure 7, then using an oscilloscope to measure the voltage at the ground input. The trace from a DUT shows the ten rounds clearly visible, and ready for DPA attacks.

6 Concluding Remarks

This paper presents a design for implementation of 128-bit AES FPGA. This encryption tool supports encryption and decryption processes that meet the NES AES FIPS 197 standard and the advanced encryption standard algorithm validation suite (AESAVS) testing. In future research, the AES encryption device will need to be tested against simple power analysis and differential power analysis. Currently, this DUT is still under development. It is hoped that this design would help hardware researchers in studying cryptographic attacks and countermeasures. The usage of open source and reconfigurable hardware (FPGA) allows for easy application to wider research areas, such as higher order DPA and correlation power analysis. Future development of this device is planned so that

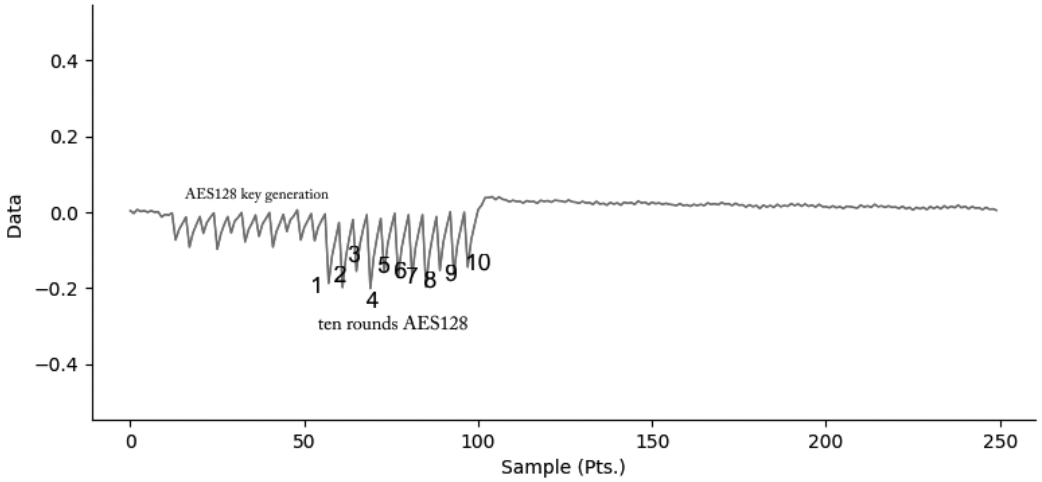

Figure 9: Power traces from a DUT performing an AES128

it could be configured to different encryption algorithms, such as RSA, ECC, DES, and BC3. Our proposed DUT also has applications in various platforms, such as smartcards, microcontrollers, and ASIC, with high throughput and latency.

Acknowledgments

The first author acknowledges support from the Indonesia Endowment Fund for Education (LPDP -Lembaga Pengelola Dana Pendidikan) scholarship, Ministry of Finance, The Republic of Indonesia. The authors gratefully acknowledge the anonymous reviewers for their valuable comments.

References

[1] F. Bao, C. C. Lee, and M. S. Hwang, "Cryptanalysis and improvement on batch verifying multiple rsa digital signatures," *Applied Mathematics and Computation*, vol. 172, no. 2, pp. 1195–1200, 2006.

[2] S. S. Chawla and N. Goel, "FPGA implementation of an 8-bit AES architecture: A rolled and masked s-box approach," in *Annual IEEE India Conference (INDICON'15)*, pp. 1–6, Dec. 2015.

[3] C. Chiţu and M. Glesner, "An FPGA implementation of the AES-Rijndael in OCB/ECB modes of operation," *Microelectronics Journal*, vol. 36, no. 2, pp. 139–146, 2005.

[4] H. S. Deshpande, K. J. Karande, and A. O. Mulani, "Area optimized implementation of AES algorithm on FPGA," in *International Conference on Communications and Signal Processing (ICCSP'15)*, pp. 10–14, 2015.

[5] O. S. Dhede and S. K. Shah, "A review: Hardware implementation of AES using minimal resources on FPGA," in *International Conference on Pervasive Computing (ICPC'15)*, pp. 1–3, 2015.

[6] J. F. Dooley, *A Brief History of Cryptology and Cryptographic Algorithms*, Springer, 2013.

[7] J. D. Golić and C. Tymen, "Multiplicative masking and power analysis of AES," in *Cryptographic Hardware and Embedded Systems (CHES'02)*, pp. 198–212, 2003.

[8] J. M. Granado-Criado, M. A. Vega-Rodríguez, J. M. Sánchez-Pérez, and J. A. Gómez-Pulido, "A new methodology to implement the AES algorithm using partial and dynamic reconfiguration," *Integration, the VLSI Journal*, vol. 43, no. 1, pp. 72–80, 2010.

[9] T. Gulom, "The encryption algorithm AES-PES16-1 and AES-RFWKPES16-1 based on network PES16-1 and RFWKPES16-1," *International Journal of Electronics and Information Engineering*, vol. 3, no. 2, pp. 53–66, 2015.

[10] T. Gulom, "The encryption algorithm GOST28147-89-PES16-2 and GOST28147-89-RFWKPES16-2," *International Journal of Electronics and Information Engineering*, vol. 6, no. 1, pp. 1–11, 2017.

[11] M. Hutter, M. Kirschbaum, T. Plos, J. M. Schmidt, and S. Mangard, "Exploiting the difference of side-channel leakages," in *Constructive Side-Channel Analysis and Secure Design*, pp. 1–16, 2012.

[12] M. S. Hwang, C. T. Li, J. J. Shen, Y. P. Chu, "Challenges in e-government and security of information", *Information & Security: An International Journal*, vol. 15, no. 1, pp. 9–20, 2004.

[13] J. N. Jr, "Analysis of venkaiah *et al.*'s AES design," *International Journal of Network Security*, vol. 9, no. 3, pp. 285–289, 2009.

[14] T. Katashita, Y. Hori, H. Sakane, and A. Satoh, "Side-channel attack standard evaluation board sasebo-w for smartcard testing," *Power*, vol. 3, pp. 400, 2012.

[15] T. Katashita, A. Satoh, K. Kikuchi, H. Nakagawa, and M. Aoyagi, "Evaluation of dpa characteristics of sasebo for board level simulations," in *International Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE'10)*, vol. 36, pp. 39, 2010.

[16] G. Khedkar, D. Kudithipudi, and G. S. Rose, "Power profile obfuscation using nanoscale memristive devices to counter DPA attacks," *IEEE Transactions on Nanotechnology*, vol. 14, pp. 26–35, Jan. 2015.

[17] P. Kocher, R. Lee, G. McGraw, A. Raghunathan, and S. Moderator-Ravi, "Security as a new dimension in embedded system design," in *Proceedings of the 41st annual Design Automation Conference*, pp. 753–760, 2004.

[18] P. C. Kocher, "Timing attacks on implementations of diffie-hellman, RSA, DSS, and other systems," in *Advances in Cryptology (CRYPTO'96)*, pp. 104–113, 1996.

[19] P. C. Kocher, J. M. Jaffe, and B. C. Jun, *Differential Power Analysis*, US Patent 7, 599, 488, Oct. 6, 2009..

[20] I. C. Lin, M. S. Hwang, C. C. Chang, "A new key assignment scheme for enforcing complicated access control policies in hierarchy", *Future Generation Computer Systems*, vol. 19, no. 4, pp. 457–462, May 2003.

[21] M. Masoumi, P. Habibi, and M. Jadidi, "Efficient implementation of masked AES on side-channel attack standard evaluation board," in *International Conference on Information Society (i-Society'15)*, pp. 151–156, 2015.

[22] A. Mersaid, T. Gulom, "The encryption algorithm AES-RFWKIDEA32-1 based on network RFWKIDEA32-1," *International Journal of Electronics and Information Engineering*, vol. 4, no. 1, pp. 1–11, 2016.

[23] S. D. Putra, A. S. Ahmad, and S. Sutikno, "Dpa-countermeasure with knowledge growing system," in *International Symposium on Electronics and Smart Devices (ISESD'16)*, pp. 16–20, 2016.

[24] S. D. Putra, A. S. Ahmad, and S. Sutikno, "Power analysis attack on implementation of DES," in *International Conference on Information Technology Systems and Innovation (ICITSI'16)*, pp. 1–6, 2016.

[25] S. Ravi, A. Raghunathan, and S. Chakradhar, "Tamper resistance mechanisms for secure embedded systems," in *17th International Conference on VLSI Design*, pp. 605–611, 2004.

[26] Y. Souissi, S. Guilley, S. Bhasin, and J. L. Danger, "Common framework to evaluate modern embedded systems against side-channel attacks," in *IEEE International Conference on Technologies for Homeland Security (HST'11)*, pp. 86–91, 2011.

[27] G. Tuychiev, "The encryption algorithms GOST28147-89-IDEA8-4 and GOST28147-89-RFWKIDEA8-4," *International Journal of Electronics and Information Engineering*, vol. 6, no. 2, pp. 59-71, 2017.

[28] R. Velegalati and J. P. Kaps, "Introducing FOBOS: Flexible open-source bOard for side-channel analysis," in *Third International Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE'12)*, Work in Progress Session, 2012.

[29] J. Wu, Y. Shi, and M. Choi, "Measurement and evaluation of power analysis attacks on asynchronous S-box," *IEEE Transactions on Instrumentation and Measurement*, vol. 61, pp. 2765–2775, Oct. 2012.

[30] W. Yu and S. Köse, "A voltage regulator-assisted lightweight AES implementation against dpa attacks," *IEEE Transactions on Circuits and Systems*, vol. 63, pp. 1152–1163, Aug. 2016.

Biography

Septafiansyah Dwi Putra received the B.S. and M.S. degrees in electrical engineering and informatics from Lampung University and Institut Teknologi Bandung, in 2011 and 2014, respectively. Currently, he is working toward the Ph.D. degree in CAIRG-Research Group, School of Electrical Engginering and Informatics, Institut Teknologi Bandung, Indonesia. His current research interests are computer security, cognitive artificial intelligence and hardware security. He is currently associated with Management of Informatics - Politeknik Negeri Lampung as lecturer.

Ma'muri received the B.S. degree in electrical engineering from Institut Teknologi Bandung (ITB), Bandung, Indonesia, in 2004, respectively. He is currently the Master Student of School of Electrical Engineering and Informatics ITB. His research interests focus on the design and analysing hardware security.

Sarwono Sutikno received B.S in Electronics degree from Institute Technoloy of Bandung, Bandung, Indonesia, in 1984, and received the Master of Engineering degree and Doctor of Engineering degree in Integrated System from Tokyo Institute of Technology, Tokyo, Japan in 1990 and 1994, respectively. His research interests focus on implementation of cryptographics algorithms in Integrated Circuits including Embedded System Security. His Security Engineering focus includes Information Security Management System. He holds several professional certifications including Certified Information System Auditor and ISMS Provisional Auditor, he is also appointed ISACA Academic Advocate. Currently, He is advisor to the Corruption Eradication Commission Republic of Indonesia.

Yusuf Kurniawan received the B.S. degree,master degree, and doctoral degree in electrical engineering from Institut Teknologi Bandung (ITB), Bandung, Indonesia, in 1994, 1997, 2007, respectively. He is currently as lecture of School of Electrical Engineering and Informatics ITB. His research interests focus on the design of block cipher and cryptology.

Adang Suwandi Ahmad (ASA) received his engineer-

ing degree (Ir.) in Electrical Engineering from ITB in 1976, Diplome Etude Approfondi Signaux et Bruits (DEA) option Electronique, and Docteur Ingenieur Signaux et Bruits option Electronique (Dr.-ing) from Universite des Sciences du Languedoc Montpellier, France in 1978 and July 1980 respectively. He became Institut Teknologi Bandungs Professor in Intelligent Electronics Instrumentation System in 2000. ASA past researches were in Electronics Instrumentation Systems (Devices and Systems) and Intelligent Electronics Systems/Artificial Intelligence. Cooperation with Navy Research Service (1992) had yielded a War Game Simulator. He founded Intelligent System Research Group(ISRG)ITB in1993. Now he focuses in information sciences, intelligent computations, and intelligent-based systems. He is the former Dean of the School of Electrical Engineering and Informatics ITB, Bandung, Indonesia.