AFRL-RI-RS-TR-2012-210

SECURE HARDWARE DESIGN FOR TRUST

AUGUST 2012

INTERIM TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ AIR FORCE MATERIEL COMMAND ■ UNITED STATES AIR FORCE ■ ROME, NY 13441

NOTICE AND SIGNATURE PAGE

AFRL-RI-RS-TR-2012-210 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/ S /
STEVEN T. JOHNS, Chief
Trusted Systems Branch
Computing & Communications Division

/ S /
PAUL ANTONIK
Technical Advisor, Computing &
Communications Division
Information Directorate

REPORT DOCUMENTATION PAGE	*Form Approved* **OMB No. 0704-0188**

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
AUGUST 2012	INTERIM TECHNICAL REPORT	AUG 2011 – JUN 2012

4. TITLE AND SUBTITLE
SECURE HARDWARE DESIGN FOR TRUST

5a. CONTRACT NUMBER IN HOUSE
5b. GRANT NUMBER N/A
5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Youngok Pino and Garrett S. Rose

5d. PROJECT NUMBER SHDT
5e. TASK NUMBER IN
5f. WORK UNIT NUMBER HO

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/Information Directorate
Rome Research Site/RITA
525 Brooks Road
Rome NY 13441-4505

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/Information Directorate
Rome Research Site/RITA
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
N/A

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RI-RS-TR-2012-210

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. PA# 88ABW-2012-4496
Date Cleared: 17 Aug 2012

13. SUPPLEMENTARY NOTES

14. ABSTRACT
A logic encryption (LE) algorithm has been developed to protect integrated circuits (ICs) from malicious attacks in the current supply chain model. Typical hardware attacks are performed by understanding functionality of designs. This technique is focused on obscuring the functionality of designs by inserting additional gates (called key gates) into the original design. The end user has to provide valid key bits to the key gates in order to enable correct functionality. In this approach, we leverage IC testing concepts to determine the locations of key gates. For XOR and multiplexer gate insertion techniques, Hamming distance (HD) as a security measure was analyzed using 10 benchmark circuits and the proof-of-concept was demonstrated with an AES core on an FPGA board. The goal is to achieve 50% of Hamming distance between the outputs on applying a correct key and an incorrect key. Most of the benchmark circuits encrypted with XOR or MUX gates achieved 50% of HD which proves the strength of the developed technique.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
					YOUNGOK PINO
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPONE NUMBER** (*Include area code*)
U	U	U	UU	17	**N/A**

Abstract

Over 50 percent of the integrated circuits (ICs) designed for modern US weapon systems are commercial off the shelf (COTS) parts, primarily field programmable gate arrays (FPGAs) which are mostly manufactured outside of the United States. These parts are vulnerable to malicious alterations that could be inserted during the design process and remain undetectable. Such malicious alterations could contain hidden back doors enabling an undetectable attacker to gain control of the system, disable networks, leak confidential information, or degrade signal integrity. Most malicious attacks are performed by understanding the functionality of ICs. If a designer is able to conceal the functionality of an IC, then most of these attacks can be prevented. In this work, we have developed a technique called Logic Encryption to harden computer circuitry against these malicious threats by hiding the functionality of a design. The functionality of a design will be concealed until it is configured by a designer after fabrication and all critical computing components are under Air Force custody and control. Logic encryption can be achieved by inserting additional gates like XORs and multiplexers such that correct outputs are produced only when specific inputs or keys are applied to these gates after manufacturing. We relate logic encryption to fault analysis in IC testing to develop an effective and efficient way to guide the insertion of key gates. This method is process independent and can yield valuable security benefits because one need not trust the fabrication, test, and other third party participants in the outsourced fabrication process model. Most importantly, this work will be used as an added layer of security to complement existing software and network security, especially within mission-critical systems.

1. Introduction

Chip design is becoming increasingly vulnerable to malicious activities and alterations as fabrication foundries continue to move offshore. This trend leads to IC designs that are susceptible to cloning, tampering, or reverse engineering. These issues have raised serious concerns for the assured security of integrated circuits, especially those employed within mission-critical hardware for surveillance, communications, and weapons operations. Military electronic systems require a high level of confidence that the microprocessors being used are authentic and secured. For example, a custom microprocessor design can be easily altered to operate identically to the genuine device [1]. Alterations can be designed to disable the system intentionally or to compromise the system in some manner to leak sensitive information at any point in the future.

In February 2008, the US Department of Justice announced that $78 million US dollars worth of counterfeit Cisco systems were imported from China the previous year. For instance, on January 4th, 2008, two individuals faced federal charges for trafficking counterfeit Cisco products. The traffickers imported counterfeit computer network hardware, from an individual in China, and sold it to retailers throughout the United States. They also shipped counterfeit products directly to the Marine Corps, Air Force, Federal Aviation Administration, defense contractors, Federal Bureau of Investigation, universities and financial institutions. In a *Business*

Week article on October 2nd, 2009, Melissa E. Hathaway, head of cyber security for the Office of the Director of National Intelligence, said that counterfeit products have been linked to the crash of mission-critical networks, and may also contain hidden ‘back doors’ enabling network security to be bypassed and sensitive data to be accessed. The US Department of Defense has recognized the potential national defense vulnerabilities caused by counterfeit devices within military systems [2].

In the *Technology Horizons* report by US Air Force Chief Scientist, Dr. Dahm, it is stated that one of the technology-derived challenges to Air Force capabilities is having cyber operations in un-trusted environments in both software and hardware [3]. This is because most providers of critical computing equipment and infrastructure for cyber systems originate from multinational foreign based manufacturing operations. Thus, the ability to regulate, supervise, and secure any manufacturing processes are out of US control and jurisdiction to ensure trust and integrity of the computing hardware components. In addition, one of the greatest risks to national security is the loss of the critical hardware containing secured and trusted computing architecture designs in the field. If critical deployed hardware is lost, captured and no longer under US control, it can be subject to reverse engineering. Therefore, it is critical to develop secured hardware technologies that inherently prevent tampering and reverse engineering. This research and development effort describes techniques applied to IC processor designs that will provide the mechanisms to maintain secure data integrity and information assurance for Air Force information processing and computing systems. In addition, this computing architecture can provide the foundation for future trusted designs irrespective of fabrication origin. Thus, it will eliminate national defense, information and cyber security risks and vulnerabilities due to exposure to untrusted semiconductor manufacturing foundries.

2. Technical Background

The issue of hardware trojans and solutions to this problem have been studied extensively in recent years. In a technique called *state-based obfuscation* proposed by Chakraborty and Bhunia [4], finite state machines (FSMs) are inserted in the design which initially locks all processor functionality. Then, by using a secured key, the design can be unlocked causing the processor to be functional once again. The output of this FSM is connected to XOR gates with a few selected nodes of the circuit.

An approach proposed by Jarrod Roy [5] is based on using *XOR/XNOR gate-based obfuscation*. In this technique, a node within the circuit or net is selected and replaced by an XOR/XNOR gate. One of the inputs to the inserted XOR/XNOR gate is a key bit while the other is the original signal for the selected net. The XOR/XNOR gates act as buffers with the correct key while they become inverters with an incorrect key. In addition, a chip activation protocol is implemented based on public-key cryptography for strong security within the IP holder and foundry.

Finally, memory elements such as LUTs can also be inserted into a design. For example, four input LUTs can be inserted within various circuit paths. For each LUT, one input is selected as the true path and the other inputs are chosen to process random paths. Based on the LUT inputs, the output is selected from the contents of the respective LUT. The circuit will function correctly only when these elements are configured or programmed correctly. However, this approach incurs significant performance overhead.

3. Technical Approaches

The objective of this research effort is to develop techniques that prevent tampering during the design and manufacturing process of computing architectures. The resulting methodology will protect Air Force information computing systems hardware for air, space, and cyberspace computing applications.

3.1 Our approach

In XOR/XNOR based logic encryption [5], gates are inserted at random locations in a design. However, this approach does not guarantee that a wrong key will impact the output as its effects are not necessarily propagated to the output. This is similar to an IC testing scenario where the effect of a fault is not guaranteed to propagate to the output. In our approach, we relate these two scenarios and ensure that the effect of using incorrect keys always propagates to the outputs.

3.1.1. Principles of IC testing

Manufactured ICs may contain defects for a variety of reasons such as shrinking technology dimensions. Thus, all manufactured ICs have to be tested to prevent defective chips from entering the supply chain. During IC testing, input patterns are applied to an IC and the response is observed at the outputs [6]. If the observed response is different from the correct response, that IC would be classified as faulty. IC testing is enabled by modeling the defects as faults and designing algorithms to determine the input patterns that excite those faults and propagate their impact on the outputs. Most of the defects can be modeled as either stuck-at-0 (s-a-0) or stuck-at-1 (s-a-1) where a net in the design can be forced to either logic 0 or logic 1, respectively.

As traditional IC testing algorithms analyze the effect of faults in a circuit and provide ways to propagate their effects to the circuit output, we leverage them to perform logic encryption to guide key gate insertion.

3.2. Fault Analysis based Logic Encryption

We call the gates inserted during encryption key gates and use two types: 1) XOR/XNOR gates and 2) multiplexers.

In XOR/XNOR based encryption, XOR/XNOR gates are introduced into the circuit such that one input of the inserted gate is a key and the other input is a net in the original design, referred to here as the true net. Based on the key, the XOR/XNOR gate can either retain the true signal or invert it.

In multiplexer based logic encryption, multiplexers are inserted such that one input of the multiplexer will be the true (original) net in the design. The other input of the multiplexer, referred as the false input, is another net in the design and the select line of the multiplexer is connected to the respective key bit. On applying the correct key value, the true net is selected, retaining the correct functionality of the design, otherwise, the functionality is modified by selecting the incorrect or false net.

We will describe how to encrypt a design in such a way that any incorrect key causes an incorrect output. This is similar to the situation where an incorrect output is produced when the circuit has a fault which has been excited and propagated to the outputs.

3.2.1. Fault Excitation

Application of an incorrect key should change the logic value at the place where the corresponding key gate is inserted. Hence, applying an incorrect key can be associated with the activation of a fault. An example of a circuit with an excited fault can be seen in Figure 1.

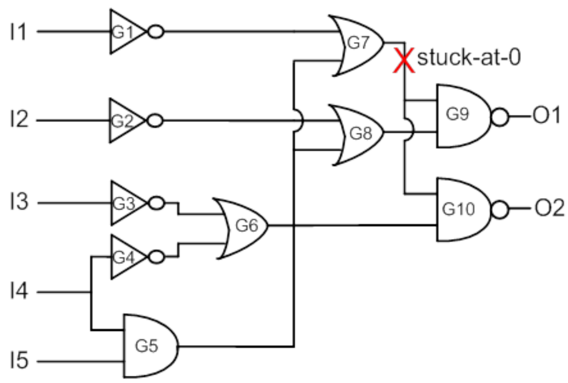


Figure 1: A circuit with a stuck-at-0 fault.

3.2.1.2. Fault Excitation: XOR/XNOR gates

In the case of XOR/XNOR gates, fault excitation is always guaranteed on applying an incorrect key as the true signal gets inverted, i.e., either a stuck-at-0 (s-a-0) or stuck-at-1 (s-a-1) fault will be excited. Figure 2(a) is a circuit encrypted with one XOR gate (E1). If an incorrect key (K1=1) is applied to the circuit, the value of net B is the negated value of net A. This is the same as exciting a s-a-0 when A=1 or s-a-1 when A=0 at the output of G7 as shown in Figure 1.

3.2.1.3. Fault Excitation: Multiplexers

In the case of multiplexer based key gates, the application of an incorrect key causes a false net to be selected instead of the true net. For example, Figure 2(b) shows a circuit encrypted with one multiplexer (E1). If an incorrect key (K1=1) is applied to the circuit, the value of net Y gets the false value of the net F instead of the true value of the net T. For the input pattern 1X111, the values on T and F are 1 and 0, respectively. On applying an incorrect key, a s-a-0 fault is excited at the output of G7 as shown in Figure 1.

3.2.2. Fault Propagation

Not all incorrect keys can corrupt the output as the effects of an incorrect key may be blocked for some of the input patterns. This is similar to the scenario where not all input patterns can propagate the effect of an excited fault to the output.

3.2.2.1. Fault Propagation: XOR/XNOR gates

Considering the circuit in Figure 2(a), an input pattern of 01110 and an incorrect key (K1=1) leads to an output of 11 which is the same as the correct functional output. Thus, the output is correct even though the s–a–0 fault gets excited at the output of E1.

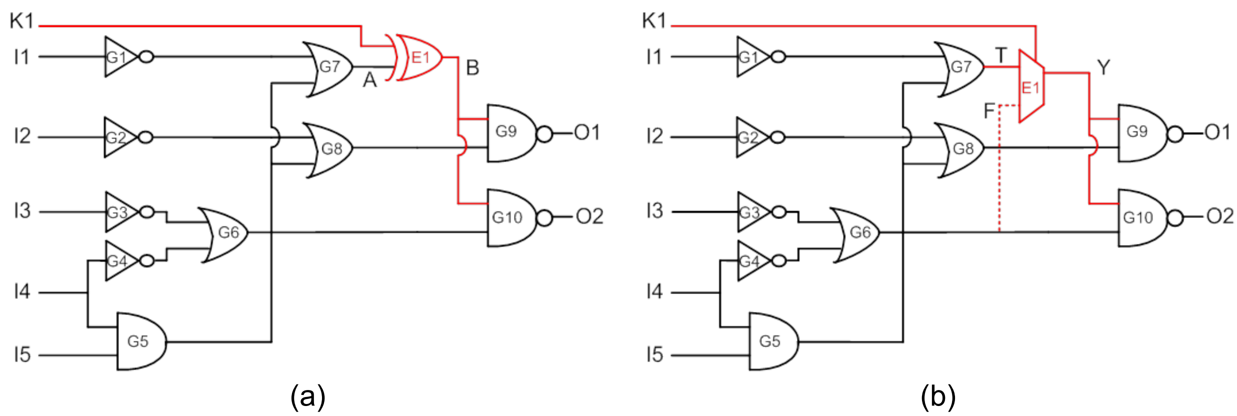


Figure 2: (a) A circuit encrypted with one XOR gate (E1). (b) A circuit encrypted with one multiplexer (E1).

3.2.2.2. Fault Propagation: Multiplexers

Consider the circuit with multiplexer based logic encryption in Figure 2(b). The input pattern 01110 excites a s-a-0 fault at the output of G7. However, the impact of the fault is blocked at G10, failing to corrupt the output O2.

To propagate the effect of an excited fault, non-controlling values should be applied to the other inputs of the gates that are on the propagation path of the fault. Since not all input patterns guarantee the non-controlling values to these gates, an incorrect key will not always corrupt the output as described in the sections 3.2.2.1. and 3.2.2.2.

3.2.3. Fault Masking

Inserting a single key gate and applying an incorrect key is equivalent to exciting a single stuck-at fault. Inserting multiple key gates and applying incorrect keys is equivalent to exciting multiple stuck-at faults. However, when multiple faults are excited, some excited faults might mask the effect of others. Analogously, in logic encryption, when multiple key gates are inserted, the effect of one key gate might mask the effect of another.

3.2.3.1. Fault Propagation: XOR/XNOR gates

When the key bits (K1 and K2) are 00 for the circuit shown in Figure 3(a), the correct functional output is 00 for the input pattern '00000'. However, if the key bits are 11 (incorrect key), the effect introduced by the XOR gate E1 is masked by the XOR gate E2, and the circuit still produces a correct output at O2, i.e., a logic 0. Thus, similar to fault masking, the effect of one XOR gate can be masked by the effect of another XOR gate.

3.2.3.2. Fault Propagation: Multiplexers

When the key bits (K1 and K2) 00 are applied to the circuit shown in Figure 3(b), the correct functional output at O2 is 0 for the input pattern '0X110'. However, if the key bits are 11 (incorrect key), the effect introduced by the multiplexer E1 is masked by the multiplexer gate E2, and produces a correct output at O2, i.e., a logic 0. Thus, similar to fault masking, the effect of one multiplexer can be masked by the effect of another multiplexer.

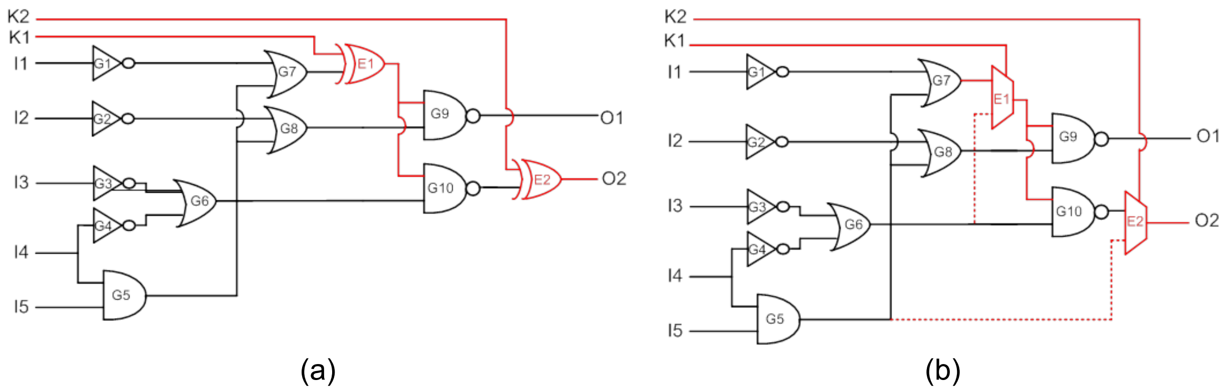


Figure 3: (a) A circuit encrypted with two XOR gate (E1 and E2). (b) A circuit encrypted with two multiplexers (E1 and E2).

3.3. Logic Encryption

3.3.1. Hamming Distance

If only one or more of the output bits are wrong and the other output bits do not change for an invalid key, then the attacker might figure out the functionality using the unaffected outputs. If all outputs are affected, then the output will be the exact complement of the correct output. Thus, an attacker may relate the outputs to the inputs. Ideally, if 50% of the output bits are affected, and if the set of affected outputs changes from one pattern to the other, then it is difficult for an attacker to relate the outputs and determine the functionality of the design. Hence, 50% of the output bits should be affected on applying an invalid key. In terms of fault simulation, this problem can be stated as finding a set of faults which together will affect 50% of the outputs.

3.3.2. Fault Impact

A greedy iterative approach is used to insert key gates. In each iteration, the fault that has the potential of propagating to a maximum number of outputs dictates the location of the key gate to be inserted. For every iteration (except for the first), the key gates inserted at previous iterations are provided with random incorrect keys thereby emulating a multiple stuck-at fault scenario, and accounting for the previous key gate insertions. To insert a key gate, we need to determine the location in the circuit where, if a fault occurs, it can affect most of the outputs for most of the input patterns. We use fault impact as defined by the following equation to identify such locations.

$$\text{Fault Impact} = (\#\text{ of Test Patterns}_{s\text{-}a\text{-}0} \times \#\text{ of Outputs}_{s\text{-}a\text{-}0}) + (\#\text{ of Test Patterns}_{s\text{-}a\text{-}1} \times \#\text{ of Outputs}_{s\text{-}a\text{-}1}) \qquad (1)$$

For any input pattern, depending upon the value at the control input of the gate either a s-a-0 or s-a-1 fault will be excited at the output of a gate. However, the effect of the fault will be observed at one or more outputs for only some of the input patterns. From a set of test patterns, we computed the number of patterns that detect the s-a-0 fault (# Test $\text{Patterns}_{s\text{-}a\text{-}0}$) on every net and the cumulative number of output bits that get affected by that s-a-0 fault (# $\text{Outputs}_{s\text{-}a\text{-}0}$). Similarly, for s-a-1 faults we computed # Test $\text{Patterns}_{s\text{-}a\text{-}1}$ and # $\text{Outputs}_{s\text{-}a\text{-}1}$. The net with the highest fault impact points to the location where the key gate is to be inserted.

3.3.3. Contradiction Metric

As described in 3.2.1.3., fault excitation in multiplexer-based encryption will happen only if the value on the true net is different from the value on the false net. The true net has already been selected based on the fault-impact metric described in 3.3.2. The selection of the false net is done based on another metric, the *contradiction metric*, which aims at maximizing the probabilities of having complementary values on the true and the false nets to select the best false net:

$$\text{Contradiction Metric} = (P_{0,\ true} \times P_{1,\ false}) + (P_{1,\ true} \times P_{0,\ false}), \qquad (2)$$

```
input: Netlist
output: Netlist with key gates
for i ← 1 to KeySize do
 |    foreach gate j ∈ Netlist do
 |     |    Compute Fault Impact (Test Patterns, Random Key);
 |    End
 |    Select the gate with the highest Fault Impact;
 |    Insert a Multiplexer and update the Netlist;
 |    Increment KeySize;
 |    Calculate the Hamming Distance (Test Pattern, Random Key)
 |    between the obtained output and the correct functional
 |    output;
 |    if Hamming Distance == 50% then
 |     |    Terminate
 |    end
 |    if KeySize == Max KeySize then
 |     |    Terminate
 |    end
end
```

Algorithm 1: A fault impact based algorithm to insert key gates in circuit path.

where $P_{0,\ true}$ and $P_{1,\ true}$ are the probabilities of getting a 0 and 1 on the true net, respectively, while $P_{0,\ false}$ and $P_{1,\ false}$ are the probabilities of getting a 0 and 1 on the false net, respectively.

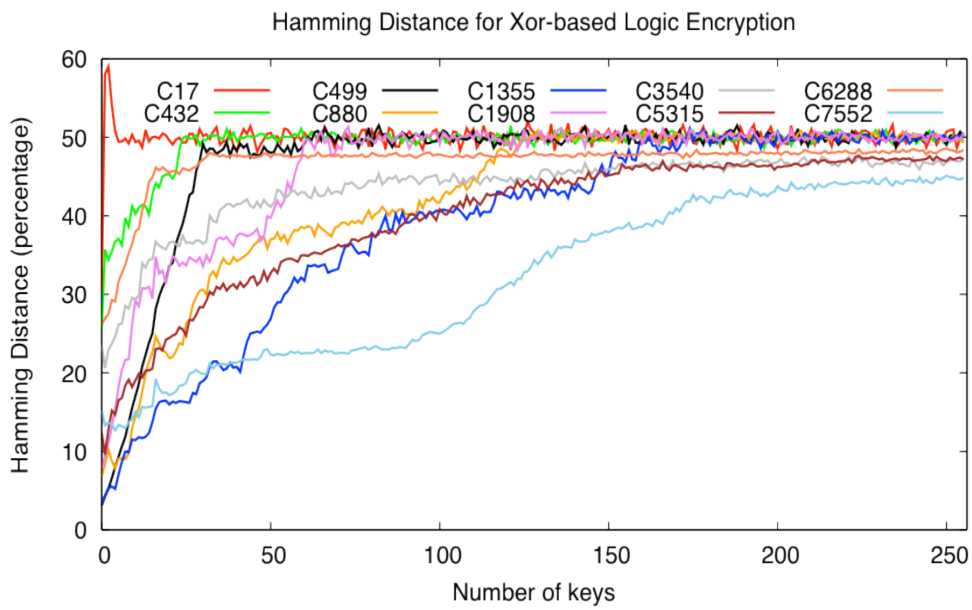
It is also worth noting that false net selection should avoid combinational loops. For the multiplexer-based encryption, a false net is selected based on the conditional probability with respect to the values on a true net.

4. Results

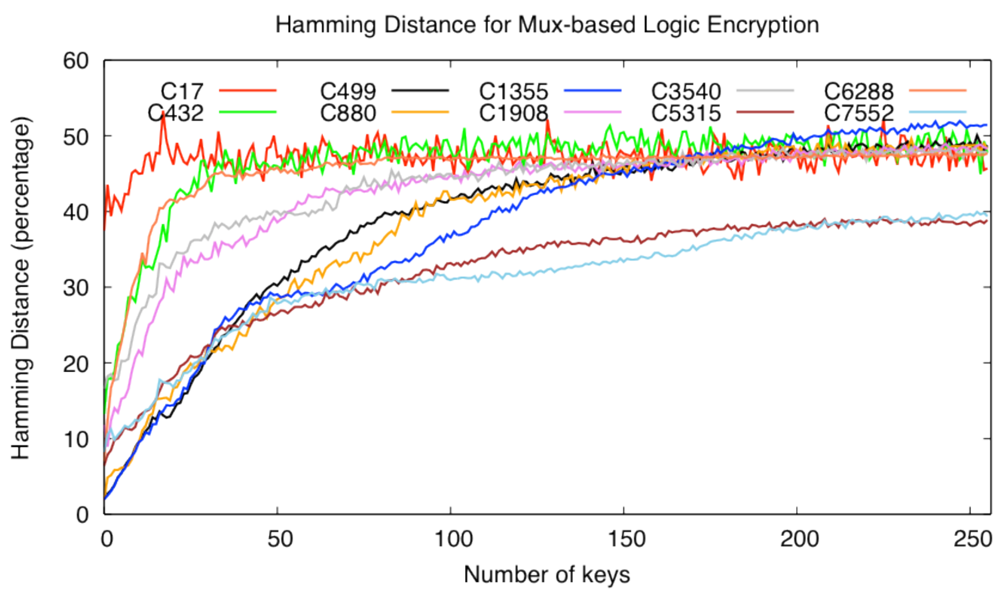
4.1. Experimental Setup

The proposed technique is evaluated using the ISCAS-85 combinational logic benchmark circuits [7]. The HOPE fault simulation tool [8] was used to calculate the fault impact of each net in a given circuit where 1000 random input patterns were applied to a netlist to observe the outputs of the unencrypted circuit. We calculated the fault impact for all possible stuck-at faults in each circuit. In multiplexer-based encryption, after selecting a true net, the contradiction metric was calculated for all possible nets and the net with the highest contradiction metric was selected as the false path. We applied valid and random incorrect keys to an encrypted netlist and determined the Hamming distance between the corresponding outputs.

Figures 4(a) and 4(b) show the Hamming distance for XOR/XNOR-based and multiplexer-based encryption schemes, respectively. The proposed XOR/XNOR-based and MUX-based insertions achieved 50% Hamming distance for most of the benchmarks as the algorithm takes the fault



(a)



(b)

Figure 4: Hamming distance between the outputs on applying a correct key and an incorrect key for different ISCAS-85 benchmark circuits. (a) XOR/XNOR-based encryption, (b) multiplexer-based encryption.

propagation and masking effects into account. In these types of encryption, a fault is always excited on applying an incorrect key.

Table I shows the number of key gates required to achieve 50% Hamming distance between the correct and the incorrect output in XOR/XNOR-based and MUX-based encryptions for some of the ISCAS-85 benchmark circuits. It can be inferred that the 50% Hamming distance objective can be achieved by inserting tens of XOR/XNOR gates in a design with a few thousand gates. This leads to low area overhead encryption. Such an effect is achieved because the technique identifies effective locations to insert the gates based on excitation, propagation, and masking principles from IC testing.

5. Demo

We encrypted an Advanced Encryption Standard (AES) circuit [9] and the encrypted design was implemented on an FPGA. The AES circuit will perform the encryption correctly only when the correct key is applied to the circuit.

5.1. Introduction of AES

The Advanced Encryption Standard (AES) algorithm is a block cipher that operates on data in blocks of 128-bits using a key size of 128, 192, or 256-bits [9]. Unlike its predecessor Data Encryption Standard (DES) which operates on a Feistel network, the AES algorithm works on a substitution permutation network using a 4x4 byte state matrix. There are four major operations: SubBytes, ShiftRows, MixColumns, and AddRoundKey each of which will modify the state matrix. Before the four major operations are performed, an initial RoundKey operation is added. These four operations will be looped a certain number of times depending on the key size used.

5.1.1. SubBytes

The SubBytes operation is a substitution transformation that performs a mapping of the state matrix with the S-Box. The resulting state matrix provides non-linearity. For example, if $State_{1,1} = [DF]$ then the new $State_{1,1} = [9E]$.

5.1.2. ShiftRows The ShiftRows operation moves the state matrix by a certain number of offset bytes. The first row of the state matrix is not shifted and remains constant. The second,

TABLE 1: Number of key gates required to achieve 50% Hamming distance between the correct and the incorrect output in (a) XOR/XNOR-based encryption and (b) MUX-based encryption for some of the ISCAS-85 benchmark circuits.

	# of key gates	
Benchmark	**(a) XOR**	**(b) MUX**
C17	2	18
C432	26	66
C499	38	210
C880	125	207
C1355	160	200
C1908	61	194
C3540	170	197
C5315	212	209*
C6288	33	82
C7552	251	225*

* For the multiplexer case, circuits C5315 and C7552 do not achieve 50% Hamming distance. The # of key gates listed for these two cases is for the maximum Hamming distance possible.

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Figure 5: S-Box [7].

third, and fourth rows are cyclically shifted by increments of one, i.e., the second row is shifted by 1, the third row is shifted by 2, and the fourth row is shifted by 3.

5.1.3. MixColumns

The MixColumns operation performs a column by column multiplication with a constant matrix. The addition and multiplication of the matrix is performed over a Galois Field (GF) of 2^8. This means that unlike traditional arithmetic, the addition in GF(2^8) will be an XOR operation. The multiplication operation is similar to normal arithmetic. Here, if the state value is greater than 0x80, the process is to left-shift by 1 and then XORed with 0x1b to prevent overflow in the Galois field. If the state value is less than 0x80, a left-shift by 1 is performed as this is the same as multiplying by 2.

5.1.4. AddRoundKey

The AddRoundKey operation is a simple XOR operation between the RoundKey and the state matrix.

5.2. Design

Logic Encryption hides the functionality of the AES design by adding additional gates into the

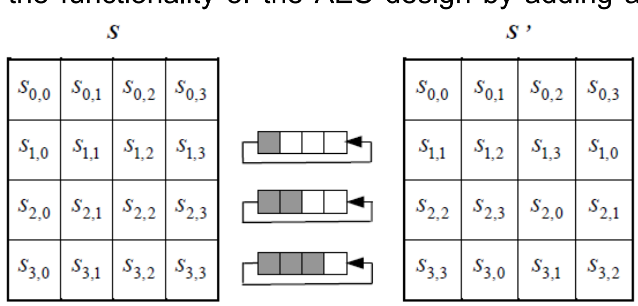


Figure 6: ShiftRows [7].

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

Figure 7: MixColumns matrix multiplication [7].

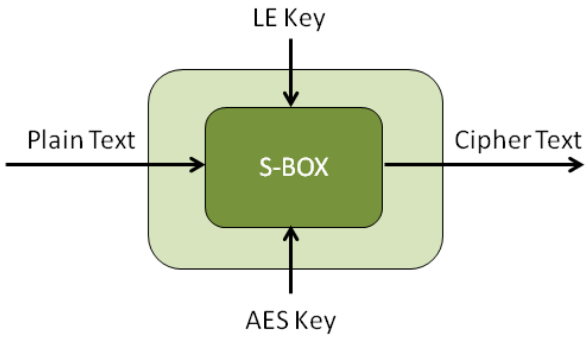


Figure 8: Encrypted S-Box with keys.

original design. In this project we implemented logic encryption on the S-Box portion of the AES code as shown in Figure 8. This was done mainly because the Verilog code for the S-Box consists of combinational circuits only. We encrypted the S-Box with the secret key set to all zero, for the sake of simplicity. The design then was synthesized with Xilinx ISE and downloaded to an Atlys FPGA board.

5.3. Simulation

Two different simulations of the AES code were performed using Xilinx iSim. At first, the AES code was simulated without logic encryption with the simulation result shown in Figure 9. Logic encryption with XOR gates was performed in the S-Box and the results with a correct key and incorrect key are shown in Figures 10 and 11, respectively. In Figure 11, we changed one bit of the LE key from the one in Figure 10 to produce an incorrect key. As a result, we see an avalanche effect on the incorrect output of the AES ciphertext. In other words, after changing only one key bit more than half of the output bits are impacted.

```
Plaintext:              0x00112233445566778899aabbccddeeff
AES key:                0x000102030405060708090a0b0c0d0e0f
Correct LE key:         0x00000000000000000000000000000000
Correct Ciphertext:     0x69c4e0d86a7b0430d8cdb78070b4c55a
Incorrect LE key:       0x00000000000000000000000000000001
Incorrect Ciphertext:   0xccbcf3b4ea75clbf2e0eeccaf0513259
```

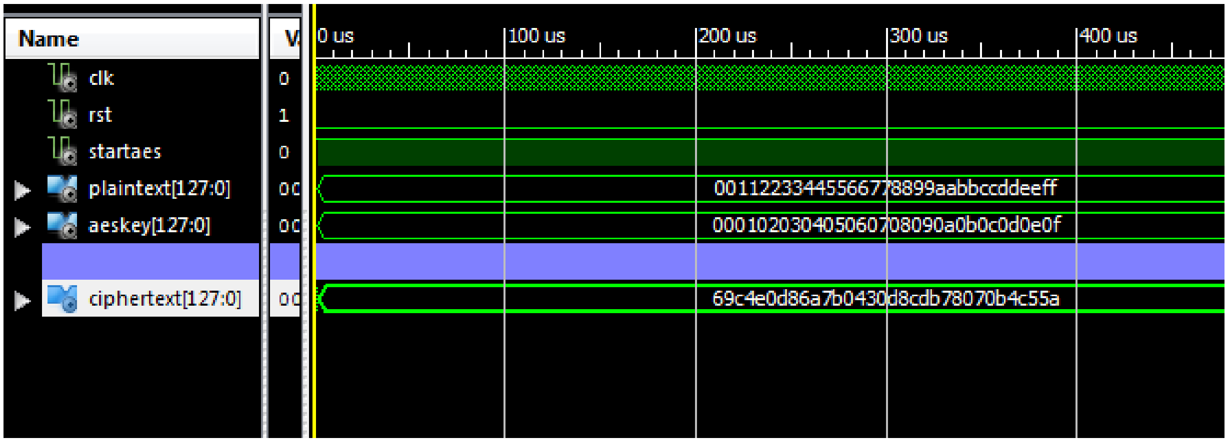


Figure 9: AES without encryption.

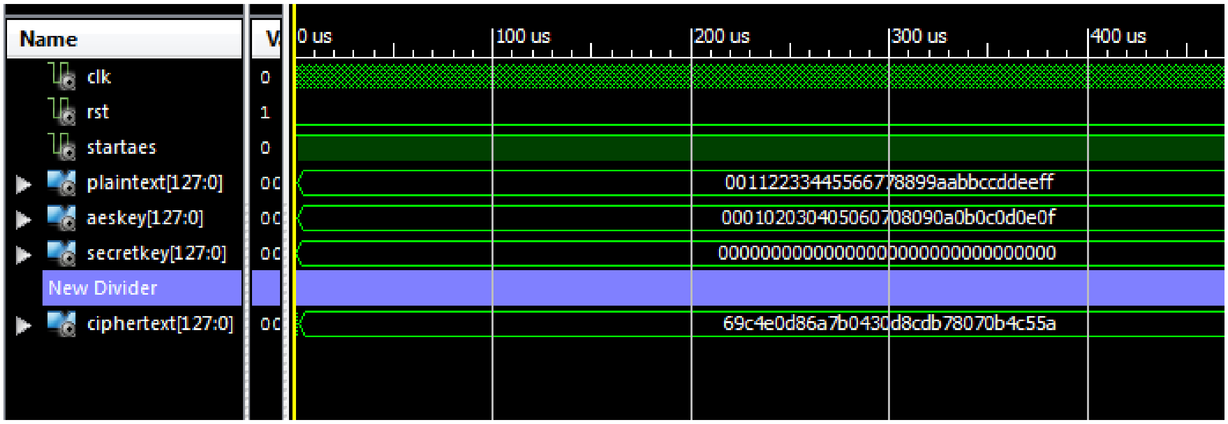


Figure 10: AES with encryption (Correct Key).

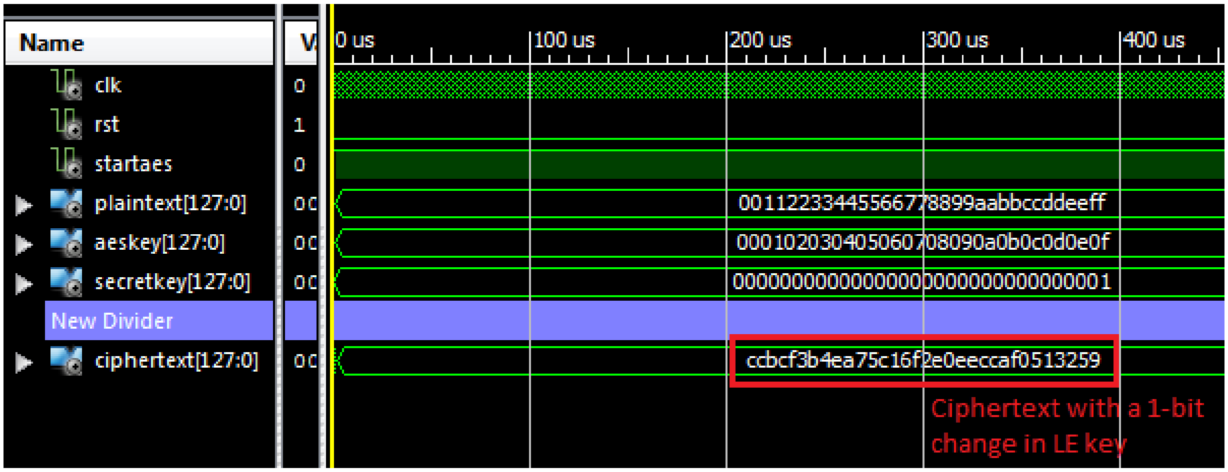


Figure 11: AES with encryption (Incorrect Key).

6. Conclusion

Fault analysis based logic encryption with XOR/XNOR gates achieves 50% Hamming distance between the correct and the corresponding wrong outputs when an invalid key is applied to the design. To improve the Hamming distance of multiplexer-based encryption, one can select the false net based on the conditional probability with respect to the values on the true net as opposed to the absolute probability. Alternatively, the fault analysis technique can be employed to guide the selection of false nets. Logic encryption can also be performed using a fault simulator that supports multiple stuck-at fault models to account for fault masking effects. Even though in this work we demonstrated our approach for encrypting combinational designs, one can also extend it to encrypt sequential designs. In addition, we implemented an AES algorithm using logic encryption as a proof-of-concept and placed the design on an FPGA board to verify the results of the outputs.

References

[1] T. Kean, D. McLaren, and C. Marsh, "Verifying the authenticity of chip designs with the DesignTag system," in *Proceedings of the IEEE International Workshop on Hardware-Oriented Security and Trust*, Anaheim, CA, pp 59–64, Jun. 2008.

[2] D. Collins, "DARPA Trust in IC's Effort," *DARPA Microsystems Technology Symposium*, San Jose, CA, Mar. 2007.

[3] W.J.A. Dahm, "Report on Technology Horizons, A Vision for Air Force Science & Technology During 2010-2030 Volume 1," U.S. Department of Air Force, Washington, D.C., Rep. AF/ST-TR-10-01-PR, May 15, 2010.

[4] R.S. Chakraborty and S. Bhunia, "Security against Hardware Trojan through a Novel Application of Design Obfuscation," in *Proceedings of the 2009 International Conference on Computer-Aided Design*, pp. 113–116.

[5] J.A. Roy, F. Koushanfa, and I.L. Markove, "EPIC: Ending Piracy of Integrated Circuits," in *Proceedings of the IEEE/ACM Design, Automation and Test in Europe*, pp. 1069–1074.

[6] M.L. Bushnell and V.D. Agrawal, *Essentials of Electronics Testing for Digital Memory, and Mixed-Signal VLSI Circuits*. Boston, MA: Kluwer Academic Publishers, 2000.

[7] M. Hansen, "Unveiling the ISCAS-85 benchmarks: a case study in reverse engineering," *IEEE Design & Test of Computers*, vol. 16, no. 3, pp. 72–80, 1999.

[8] H. Lee and D.S. Ha, "HOPE: An Efficient Parallel Fault Simulator for Synchronous Sequential Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 9, pp. 1048–1058, 1996.

[9] "Federal Information Processing Standards Publication 197: Announcing the Advanced Encryption Standard (AES)," *http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf*, Nov. 26, 2001.