2012 IEEE/ACM 45th International Symposium on Microarchitecture Workshops

From Cryptography to Hardware: Analyzing Embedded Xilinx BRAM for Cryptographic Applications

Shivam Bhasin Sylvain Guilley Jean-Luc Danger
Institut MINES-TELECOM, TELECOM ParisTech,
CNRS LTCI, "Laboratoire Traitement et Communication de l'Information" (UMR 5141),
Department COMELEC, 46 rue Barrault,
75 634 PARIS Cedex 13, FRANCE.
{bhasin,guilley,danger}@telecom-paristech.fr

Abstract

Design of cryptographic applications need special care. For instance, physical attacks like Side-Channel Analysis (SCA) are able to recover the secret key, just by observing the activity of the computation, even for mathematically robust algorithms like AES. SCA considers the "leakage" of a well chosen intermediate variable correlated with the secret. Field programmable gate-arrays (FPGA) are often used for hardware implementations for low to medium volume productions or when flexibility is needed. They offer many possibilities for the computation, like small Look-Up Tables (LUT) and embedded block memories (BRAM). Certain countermeasures can be deployed, like dual-rail logic or masking, to resist SCA on FPGA. However to design an effective countermeasure, it is of prime importance for a designer to know the main leakage sources of the device. In this paper, we analyze the leakage source of a Xilinx Virtex V FPGA by studying 3 different AES architectures. The analysis is based on real measurements by using specific leakage models of the sensitive variable, adapted to each architecture. Our results demonstrate that, BRAM which were considered to leak less traditionally, are found to be equally vulnerable if we change the attack target from address register to output latch. Hence by providing important clues about the leakage, this study allows the designers to enhance the robustness of their implementation in FPGA.

Keywords: Side-Channel Analysis, Embedded BRAM, Leakage Models, Accessed Data Leakage, Security, FPGA, Ghost Peaks.

1. Introduction

The FPGA have evolved a lot since their introduction in the late 90s. FPGA are no more just an array of few thousand programmable look-up tables (LUT). Modern devices contain up to several millions of LUT, high-density block memories, DSP cores, etc. These features along with reconfigurable nature and low-time to market have made FPGA attractive for industrial applications. Modern FPGA can be used in a wide variety of applications from running simple communication devices, image or digital processing systems to complex systems-on-chip (SoC).

Complex SoC which need to secure data on system bus or memory generally rely on embedded cryptography (ciphers) such as the advanced encryption standard (AES [1]). These ciphers are considered mathematically secure. However the physical implementation of these algorithms have been found vulnerable to attacks such as Side-Channel Attacks (SCA) [2] or Fault Attacks (FA) [3]. In such a scenario, the responsibility of securing a system is shifted from cryptographers to designers. As an FPGA designer, one should be aware of the behaviour of the target platform and what architecture would show higher resistance to given attacks.

Recently, decryption of protected bitstream on several Xilinx Virtex devices was found vulnerable to SCA [4]. This article also focuses on the issue of side-channel attacks, but on the user logic. Such attacks unfold in two stages: side-channel information collection and side-channel analysis. Side-channel collection is a "metrology" step, whereas SCA requires sophisticated theoretical tools to be efficient. Both aspects are advancing rapidly, as attested for instance by the "DPA contest" competitions [5]. An unprotected cryptographic-core (crypto-core) shows varied SCA resistance from one implementation to another. SCA resistance can also vary from one platform to another as the internal architecture of the device changes.

On one hand it is often recommended to use embedded block memories (BRAM) to implement non-linear parts of the ciphers. On the other hand, ciphers like DES [6] and PRESENT [7] have very small Sboxes which are better suited for FPGA logic. This trade-off between area and security is often a hard decision for a designer. Apart from block ciphers, other cryptographic primitives need careful implementation. For example Grøstl [8], a SHA-3 finalist based on the AES round function also suffers from similar implementation issues as AES. In the following, without loss of generality, we focus on the 128-bit version of AES as the cryptographic algorithm.

1.1. Different Implementations of AES on FPGA

AES is a substitution permutation network (SPN) product block cipher. It has an iterative structure, consisting of an initial key addition followed by ten identical rounds which are applied to the 16 bytes of message block to be encrypted. The 16 bytes are laid out as a matrix of four columns of four bytes $s_{i,j}$, where $0 \leq i \leq 3$ and $0 \leq j \leq 3$. A round consists of a fixed sequence of transformations. Apart from the last round, the other nine rounds are alike and consist of four transformations each. The last round is incomplete to ease the decryption. The four round transformations are called SubBytes, ShiftRows, MixColumns and AddRoundKey.

A simple AES crypto-core can be implemented on a FPGA in different ways. ShiftRows is a simple swapping of wires. MixColumns can be implemented with shift operations & XOR gates. AddRoundKey consists of XOR operation only. These linear operations of AES are generally implemented in FPGA logic. Sometimes DSP blocks can be used for linear operations which result in higher performance [9].

It is the non-linear layer i.e. SubBytes which can be implemented in logic blocks or memory blocks. The implementation of SubBytes in logic blocks (Fig. 1(a)) has advantages like asynchronous operation and platform independence. On the other hand, implementation of SubBytes in BRAM (Fig. 1(b)) provide merits like higher performance and fewer computation glitches. BRAM also provide an

978-0-7695-4920-0/12 $26.00 © 2012 IEEE
DOI 10.1109/MICROW.2012.11

advantage in terms of cost (*i.e.* area) as the register inside the BRAM can always be used as the state register of the cipher.

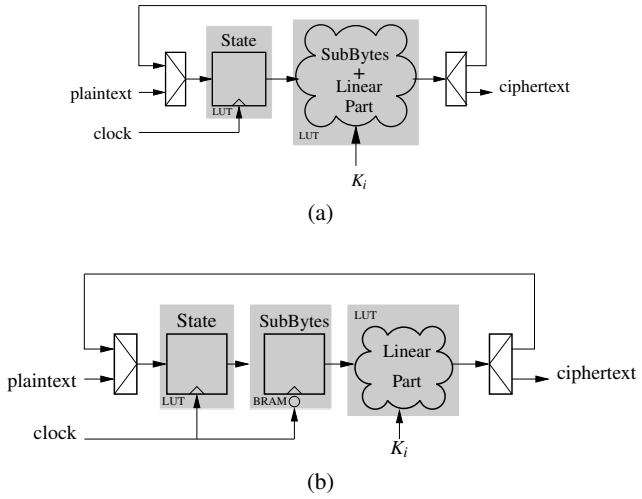

Figure 1: Illustration of AES with SubBytes in (a) FPGA logic, (b) FPGA BRAM.

Such architectures have been discussed in many articles. An AES architecture similar to Fig. 1(b) has been analyzed in [10] showing that the use of BRAM can improve side-channel resistance. The reasons are that the inter-Configurable Logic Block (CLB) routing is removed, and also the fanout is reduced [11]. In [9], authors have used a modified version of Fig. 1(b) more precisely, used a Tbox based implementation of AES, which merges MixColumns with SubBytes into a precomputed table. Even in the Tbox architecture, the last round is the same due to the absence of MixColumns. Therefore from the security/attacker point of view both architectures are more or less similar. They had implemented the linear part in DSP blocks and the state register is merged into the BRAM to achieve a high operating frequency at almost zero logic consumption. Both the architectures of Fig. 1 were also discussed in [12], with analysis against fault attacks stating that LUT based architecture perform better against faults.

In this article, we analyze various leakage sources of an FPGA to propose the best in terms of area, performance and security. We first lay the focus on detailed structure of an FPGA BRAM. Various leakage models are deduced from the knowledge of its architecture and tested on real implementations using side-channel attacks. We use Virtex V devices from Xilinx for analysis and experimental validation but the results can be simply extended to other devices.

The article is organized as follows: Section 2 gives a brief overview of FPGA and BRAM architectures. The experimental platform, leakage models and attack metrics used for the analysis are detailed in section 3. In section 4, we analyze various architectures for source of leakage and compare their attackability. This is followed by a focused attack on the BRAM address register (section 5). Finally section 6 gives general conclusions and possible extensions of this work.

2. FPGA Architecture

An FPGA is an integrated circuit composed of identical programmable logic cells or LUT as a basic building block. The LUT in a Virtex-5 FPGA can be configured as either 6-input LUT (64-bit ROM) with one output, or as two 5-input LUT (32-bit ROM) with separate outputs but few common inputs. The next level is a slice, which consists of four LUTs, four flip-flops and other logic elements like multiplexers, arithmetic carry logic etc. Two slices form a CLB, which are interconnected by a matrix of wires and programmable switches.

Modern FPGA also possess huge blocks of memories which are synchronous in nature. Every Virtex-5 FPGA has various blocks of 36Kbits true dual-port memories. The internal details of these BRAM are confidential and not published. However, the general architecture of the memory blocks can be found in the official documentation. The architecture of the BRAM is described in [13], which is same for Spartan and Virtex families. The architecture of a single port Xilinx BRAM is shown in Fig. 2.

In a Xilinx BRAM, the synchronization is done at the input. This means that the input address and the input data to a BRAM is stored in a register. This register can also be used as the state register of a cipher. It is only after the layer of registers that the address and data access the memory array.

A Sbox of a cipher can be implemented in the memory array, which is followed by a latch to hold the output content. An optional output register is also present after the latch, use of which would achieve better performance at the cost of one clock cycle. We use the BRAM as a ROM so the input data and write enable is always pulled down and the output register is disabled.

Xilinx and Altera are the two main players in the FPGA arena. So far, we have focused on Xilinx; here is now a short discussion about Altera. The equivalent of Xilinx BRAM is Altera AltSyncRam: actually, both have a similar architecture [14]. An input register stores the address and data before propagating them to the memory array. Even in Altera AltSyncRam, the output register is optional. Owing to similarity in the two architectures, we present our results for Xilinx BRAM but it should also hold for Altera families as well.

3. Experimental Platform and Attack Metrics

Our measurement setup consists of one Xilinx Virtex-V FPGA soldered on a SASEBO-GII platform, a 54855 Infiniium Agilent oscilloscope with a bandwidth of 6 GHz and a maximal sampling rate of 20 GSample/s, antennae of the HZ–15 kit from Rohde & Schwarz. The traces were acquired at a sampling rate of 2 GSample/s. We recorded the traces related to the activity of different AES crypto-processors running at 24 MHz clock frequency, hence we have 83.3 samples per clock cycle.

3.1. Leakage Models

Let us denote:

- L: a random variable representing the leakage (*e.g.* the power consumed or the magnetic field radiated by a cryptographic design);
- K: the n-bit secret cryptographic key, assumed to be uniformly distributed on $\mathbb{F}_2^n$;
- X: the known input or the output of the cryptographic design.
- $Z = f(X,K)$ for a given function f: a sensitive variable used internally, that depends only on X and K. We assume that this sensitive variable Z can be computed exhaustively from all possible K by the attacker (typically, $n = 8$ for AES, which is a feasible amount of 256 guesses). When the key guess is correct, Z and L are dependent.

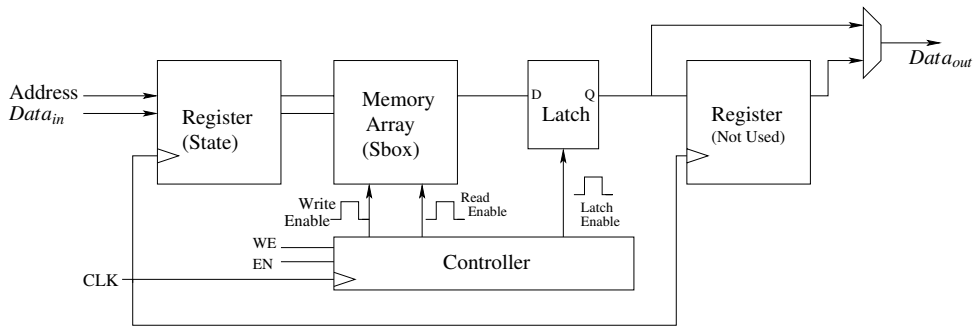

Figure 2: Internal Architecture of Xilinx BRAM.

SCA consists in the estimation of dependency between Z and L, for every key guess (*i.e.* for every value k of K). A correct analysis gives the greatest dependence for the correct value of the key, noted as $k^\star$. In practice, L is noisy and thus the link between Z and L is imperfect (it might involve other variables, making up the algorithmic noise). Therefore, many couples (X,Z) are required for the 2^n estimations (for each value of K) to find the correct key.

Now let us devise the leakage model for the previously discussed AES architectures targeting the last round of AES cipher. It has been shown in various publications that the relevant target to attack AES is the last round, when state register makes transition to the final output ciphertext. Indeed, this model involves only one key byte, whereas an attack on the first round would require a guess on a complete column (4 bytes) of the key. The model is the so-called *Hamming distance (HD) model*. It expresses at first-order the power consumption of CMOS gates in electronic devices as it correspond to signal transitions. The leakage can be expressed as follows:

$$L = HD(Z,R) + N = HW(Z \oplus R) + N\,,$$

where N is the noise, and where R is the reference state like the round 9 of AES-128. A particular case is when the reference state is initialized to zero. This yields the *the Hamming weight (HW) model* which is expressed as:

$$L = HW(Z) + N\,.$$

3.2. Attack Metrics

We use two SCA tools in our analysis. The first tool is Correlation Power Analysis (CPA [2]) as proposed by Clavier et al. CPA is a computation of the *Pearson Correlation Coefficient* ρ between the side-channel leakage L and the leakage model Z, which can be estimated as:

$$CPA : \rho(L,Z) = \frac{\Sigma_{i=0}^{n}(l_i - \mu_L)\cdot(z_i - \mu_Z)}{\sigma_L \cdot \sigma_Z}$$

where σ and μ denote the standard deviation and the mean respectively, and n is the traces count.

The CPA is efficient when L and Z are linearly related. Otherwise, the Mutual Information Analysis (MIA [15]) is a more appropriate tool as it is agnostic in the joint distribution (L,Z). Mutual information between a sensitive variable Z and a side-channel leakage L, measured in bits is:

$$MIA : I(L;Z) = H(L) - H(L|Z)\,.$$

Here $H(L)$ gives the entropy in bits of L and $H(L|Z)$ gives the conditional entropy of L knowing Z. Many methods have been proposed to estimate entropy like histograms, kernel density functions, Gaussian parametric estimators, etc [16]. In our experiments we use the Gaussian parametric estimation where the distribution of L, Z and the joint distribution (L,Z) are assumed to be Gaussian. This method might not be ideal for estimating entropy but works well in practice [16] due to the presence of environmental noise. Nevertheless other methods of estimating entropy can be applied. For instance, using Gaussian parametric estimation, the entropy of a random variable X taking values x in the support $\mathscr{X}$ can be calculated as a function of σ_X as:

$$H(X) = -\sum_{x\in\mathscr{X}} p(x)\log_2 p(x) = \log_2\left(\sigma_X\sqrt{2\pi e}\right)\,.$$

Architecture 1.

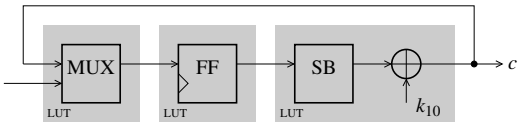

Figure 3: Leakage Model for AES architecture of Fig. 1(a).

4. Leakage Analysis

In this section, we analyze three different architectures of AES to determine the various leakage sources in an FPGA. The architectures differ in two parameters: implementation of Sboxes and state register. Now we deal with these architectures one by one.

4.1. Architecture 1: Both Sboxes and State in FPGA Logic

The most commonly used AES architecture are implemented in glue logic (Fig. 1(a)). The advantage of this architecture is its portability to both ASIC and FPGA with no or little modification. This architecture has a clear leakage from the state register which is implemented in glue logic. The state register which stores the value of round 9, leaks when the ciphertext is written into it.

Thus, leakage in this architecture can be written more precisely as:

$$M1 :\ L = HW(C \oplus SB^{-1}(SR^{-1}(C' \oplus K_{10}))) + N\,,$$

where SB^{-1} and SR^{-1} represent the inverse of SubBytes and ShiftRows operations respectively. K_{10} denotes the key guess on a byte of last round key and C, C' stand for two bytes of ciphertext, which are connected by the ShiftRows operation. The leakage can be modelled to the register FF (let's call it model $M1$) as shown in Fig. 3. In this figure, the blocks MixColumns & ShiftRows are not shown, because MixColumns is not executed in the last round and because ShiftRows does not complicate the leakage model. In the

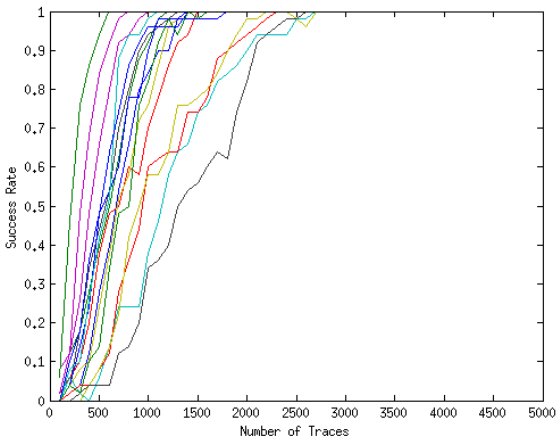

Figure 4: Success rate of CPA on Arch. 1, targeting FF using model $M1$ for all 16 Sboxes of AES.

Architecture 2.

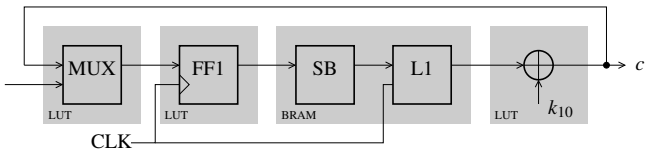

Figure 5: Leakage Model for AES architecture of Fig. 1(b) with state register in FPGA logic.

absence of ShiftRows (for instance in first row of AES state), C and C' are equal which simplifies the leakage model to:

$$M1: \; L = HW(C \oplus SB^{-1}(C \oplus K_{10})) + N,$$

This architecture shows very limited resistance against CPA. To recover the whole 128 bits of the key with a **success rate of 80% we need about 1,800 traces** (see Fig. 4).

4.2. Architecture 2: Sboxes in BRAM and State in FPGA Logic

The second architecture moves the Sboxes into BRAM while keeping the state register in glue logic. This architecture will also have a similar leakage as in the previous case, shown as $FF1$ (model $M1$) in Fig. 5. The address register of BRAM is not shown for simplicity which is clocked at the falling edge of the clock. The leakage model for both the registers should be the same so we focus only on $FF1$. Attacks targeting $FF1$ have already been dealt by various researchers [10]. However, no attention has been given to the internal latch of the BRAM which is shown as $L1$ in Fig. 5.

We repeated the attack on traces measuring the activity of this architecture. With $FF1$ as the leakage model, the CPA recovers the whole 128 bits of the key **in 5,000 traces with a success rate of 80%.** Our results on attack of FF (Arch. 1) and $FF1$ (Arch. 2) are in accordance with the findings of [10], thus the SCA resistance definitely increases when Sboxes are implemented in BRAM in terms of CPA.

Next we tried the attack on the latch $L1$ (model $M2$), to check whether it is a leakage source. The leakage model for the latch $L1$ as in Fig. 5, can be written as:

$$M2: \; L = HW(SB(C) \oplus (C \oplus K_{10})) + N,$$

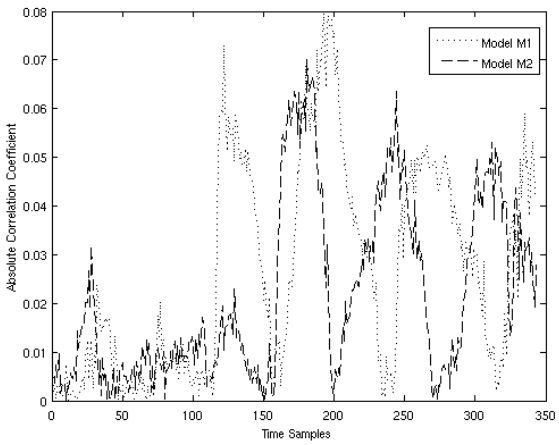

Figure 6: Leakage from Arch. 2.

where SB is the SubBytes operation.

A similar leakage should be observed when the optional output register is put in place, thus we lay focus on the latch. In some cases, the BRAM can be disabled after the encryption which would change the leakage model to Hamming weight of the round 9 output. In other implementations the next plaintext could be loaded instead of storing the ciphertext, thus depending on the implementation model M2 can be adjusted. We aim at just analyzing the leakage of the latch. This attack on leakage model $M2$ is more complicated, because the CPA always guesses two keys. The two guessed keys, namely $K^\star$ (the correct key) and $\neg K^\star$ (its one's complement), are *ex aequo*. Indeed,

$HD(C \oplus \neg K_{10}, SB(C)) = 8 - HD(C \oplus K_{10}, SB(C))$.

Moreover

$\forall (\alpha, \beta) \in \mathbb{R}^2, \alpha \neq 0, \quad |\rho(L, \alpha Z + \beta)| = |\rho(L, Z)|$.

Thus using $(\alpha, \beta) = (-1, 8)$, $|\rho(L, HD(C \oplus \neg K_{10}, SB(C)))| = |\rho(L, 8 - HD(C \oplus K_{10}, SB(C)))| = |\rho(L, HD(C \oplus K_{10}, SB(C)))|$. □

The attack on $L1$ works as good as an attack on $FF1$, which also recovered a pair for each of 16 bytes of the key **in 4,800 traces with a success rate of 80%.** The differential traces for attack on Sbox 0 of Arch. 2 using model $M1$ and $M2$ are shown in Fig. 6. The leakage $M1$ starts at a clock rising edge when the cipher is written into $FF1$. The leakage $M2$ is slightly delayed from $M1$ due to the propagation delay within the BRAM. $M1$ and $M2$ are separated by approximately 42 points, i.e. half a clock cycle, as we have 83.33 points per clock cycle. Since the BRAM is clocked at the falling edge of the clock, it is confirmed that the leakage $M2$ is coming from the internal latch of the BRAM (and all the downstream logic it drives).

4.3. Architecture 3: Both Sboxes and State in BRAM

The third architecture tested was a modified implementation of Arch. 2. In the present architecture, we merged the state register of the cipher with the internal address register of BRAM. This modification brought two advantages: low logic utilization and higher operating speed.

The leakage model is shown in Fig. 7, which is similar to the previous case. There are still two leakage sources: the register $FF2$ and the latch $L2$. The difference between Fig. 5 and 7 is that unlike $FF1$, $FF2$ is internal to the BRAM which might change its leakage characteristics.

The differential traces for Arch. 3, using leakage models $M1$ and $M2$ are shown in Fig. 8. Leakage model $M2$ which targets the latch

Architecture 3.

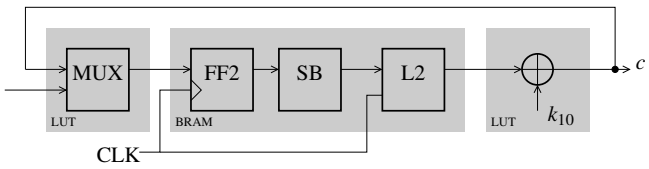

Figure 7: Leakage Model for AES architecture of Fig. 1(b) with state register in FPGA BRAM.

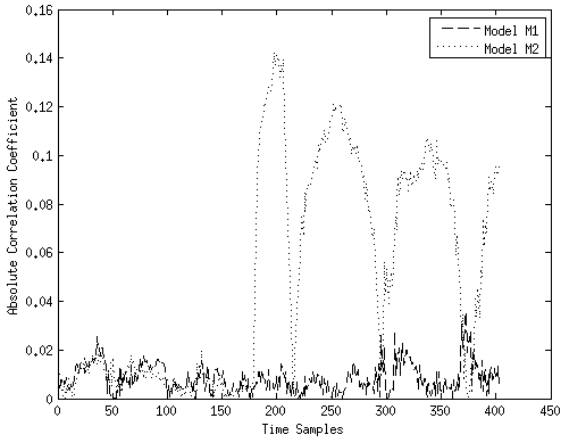

Figure 8: Leakage from Arch. 3.

$L2$ reveals the 16 pair for 16 bytes of key in 2,200 traces. To our surprise, the attack on $FF2$ totally fails (there is no "peak" for model $M1$ in the CPA curve given in Fig. 8). Even with 200,000 traces (i.e. $100\times$), the CPA is unable to reveal the secret key. If we focus on the first byte, at position $(0,0)$ of AES state, the correct key 163 is ranked 5^{th} at the end of the attack and the best ranked key is 244. We suspect that the signal-to-noise ratio (SNR) of the leakage from $FF1$ is so low that the CPA is unable to distinguish the correct key from the wrong ones. An in-depth explanation is provided in Sec. 5.

4.4. Discussion

Tab. 1 and 2 summarize the CPA results (for a chosen Sbox) and the cost in terms of area and speed for the three AES architectures. As expected, Arch. 3 is the smallest because the state register is merged with the input address register of BRAM. We can conclude that:

1. The maximum leakage comes from Target FF (Arch. 1) and $L2$ (Arch. 3) which are almost equal.
2. Then comes $FF1$ and $L1$ (both Arch. 2), which have same amount of leakage delayed by one half of clock cycle.
3. Finally $FF2$ does not seem to leak in the side-channel (denoted 'X' in Tab. 1).

Table 1: Minimum Number of Traces to retrieve the correct key for Sbox 0 using CPA.

Leakage Model / Architecture	$M1$ **(target)**	$M2$ **(target)**
Arch. 1	800 (FF)	N.A.
Arch. 2	2,910 (FF1)	2,600 (L1)
Arch. 3	X (FF2)	900 (L2)

Table 2: Cost comparison of the three architectures on Virtex V.

Architecture	Arch. 1	Arch. 2	Arch. 3
LUT	1,360	848	848
Registers	268	268	140
BRAM	0	8	8
Max. Frequency [MHz]	214.3	218.5	218.5

A careful look on Fig. 3, 5 and 7 explains the first two results. Once there is a transition on flip-flop or latch, the new signals are propagated on the routing nets. Routing nets have signal buffers at regular interval to boost the propagating signal which improves the SNR. Thus there is a leakage at the charging of sequential element which is then amplified by a series of buffers on the routing nets. For the same reason, the leakage is spread over time rather than localized just to the clock edges.

Note that in Fig. 5, the XOR gate, multiplexer and $FF1$ are normally implemented in a single LUT. This will yield a similar propagation path (routing) for $FF1$ (LUT $\rightarrow$ BRAM input) and $L1$ (BRAM output $\rightarrow$ LUT). Thus $FF1$ and $L1$ have similar leakage. $L2$ has a higher leakage as compared to $L1$ for two reasons. Firstly, $L2$ has a longer propagation path than $L1$ i.e. BRAM output $\rightarrow$ LUT $\rightarrow$ BRAM input. Secondly, the transition $FF1$ which is considered as noise w.r.t to model $M2$ is much higher than the noise introduced of by transition of $FF2$ (see Fig. 6 and 8). Therefore $L2$ has a higher SNR of leakage which makes it easier to attack. Same is the case with FF.

Inferring from the same logic we can explain one of the observations in [10]. The authors have shown that a synchronous Sbox implemented in distributed memory leaks more than the one in BRAM. Since the distributed memory uses the same routing resources, the leakage is higher than that of a BRAM. It should be noted that internal latch of the BRAM ($L1/L2$ in our case) is a significant source of leakage and it should also be taken into account when planning countermeasures.

5. Why Attack on $FF2$ Fails?

All the leakage models for the three studied architectures were found vulnerable to SCA except the address register $FF2$ of the BRAM. A higher resistance of $FF2$ can be explained by absence of FPGA routing buffers at the output of $FF2$. However, a complete failure of CPA was unexpected. We further investigated by computing the mutual information between the traces and the correct key, which determined linear and non-linear dependency between the two variables. The mutual information trace computed over 100K traces as shown in Fig. 9 confirms that there is a leakage of very small magnitude, related to $FF2$. There are four leakages which are related to model $M1$ around the samples: 120, 194, 260 and 340. Model $M2$ is leaking much more than $M1$ around the latter three samples.

5.1. Low SNR *versus* Ghost Peaks

A possible explanation for the failure of CPA using $M1$ could be due to "ghost peaks". Ghost peaks occur when a model finds a higher

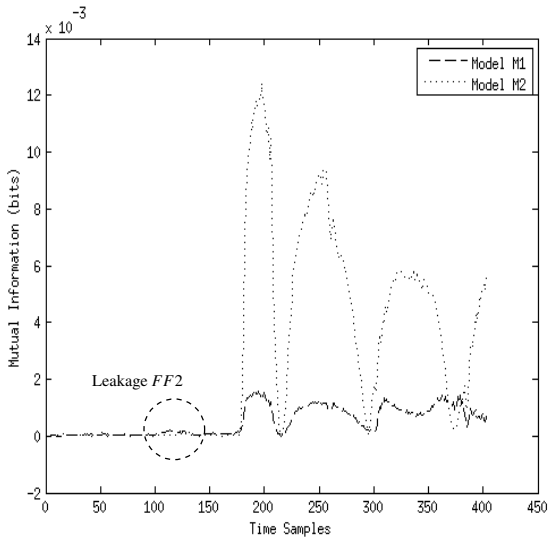

Figure 9: Mutual information between model $M1$ and Leakage from Arch. 3.

correlation with a wrong key candidate[1]. In our case, they happen because the Hamming distance at the input (model $M1$) and at the output (model $M2$) of an Sbox are actually weakly dependent, from a statistical point of view, even if the Sbox has good confusion properties. Thus, the correlation coefficient between a leakage following $M2$ and a model $M1$ is nonzero. As shown in dotted line in Fig. 10, ghost peaks appear at samples 194, 260 and 340. The actual leakage (correlation between a leakage $M1$ and the model $M1$) at sample 120 is very small, actually smaller than the ghost peaks, because of its low SNR.

Now, the ghost peaks correspond to a "cross-model" correlation on a signal with a strong SNR. Even if they exist for the correct key (163), they can be even more correlated with other values. To check that we should compute:

$$\text{argmax}_K \left|\rho\left(\underbrace{\text{HW}(SB^{-1}(C\oplus K)\oplus C)}_{\text{M1}},\underbrace{\text{HW}(C\oplus K_{10}\oplus SB(C))}_{\text{M2}}\right)\right| , \qquad (1)$$

that is not necessarily K_{10}. Notice that in Eqn. (1), the correlation ρ is computed over the values of random variable $C \in \mathbb{F}_2^8$.

For $K_{10} = 163$, we have the ghost peaks (or "cross-model") listed partially in Tab. 3. The best key for the ghost peak is 244. The correlation of model $M1$ with this (incorrect) key is plotted in dashed line in Fig. 10. It clearly appears that this curve is the greatest amongst the ghost peaks, but correlates badly on the actual leakage (at sample 120).

These results are in accordance with our previously tested CPA

[1]There are actually several kinds of "ghost peaks". In this footnote, we give some examples from the SCA literature in order to disambiguate the different origins of wrong key guesses phenomenon. One example is when a high spurious correlation is obtained at an irrelevant position of the trace, which happens when there is a lot of noise and few traces traces available to estimate the CPA. This is illustrated for instance in Fig. 3 of [17] (it is labelled "*noisy peak*"). Another kind of ghost peak can appear because unrelated activity occurs simultaneously with that being exploited. This has been exemplified in [18, §5.1], on the example of a hardware DES. If the attacker applies a mono-bit difference-of-means, then the attack can fail, because the three ignored bits (assumed erroneously to be independent from the one under analysis) leak information that overcomes the target bit, and fools the attacker into finding an incorrect key. In this section, we account for another type of ghost peaks, that happen later in time than the primary leakage, for a leakage model that involves the sensitive variable used "transformed".

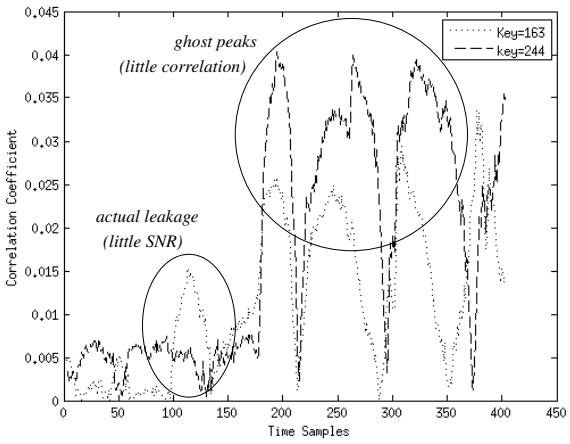

Figure 10: CPA on Arch. 3 using model $M1$, assuming the correct key (163) and best ghost peak (244).

Table 3: Values of "cross-model" correlation (Eqn. (1)), for $k_{10} = 163$, sorted by decreasing value of $|\rho|$.

Rank	Key k	$\lvert\rho\rvert$
1	*244*	*0.203222*
2	25	0.199056
3	29	0.186691
4	126	0.171591
5	*163*	*0.170469*
⋮	⋮	⋮

which also guessed 244 as the best key. Thus the correct key which could be guessed at sample 120 ($FF2$) is shadowed by the ghost key at sample 200 ($L2$).

5.2. Attack on Localized Samples

To verify our hypothesis above, we thought about repeating a CPA on the same traces by eliminating the samples which lead to ghost peak. The CPA was repeated on just 100 samples in the range (50,150), which includes sample 120. As expected, this CPA is not mitigated by ghost peaks and it is able to reveal the best key in **88,000 traces** (Fig. 10) . The attack can be accelerated a bit further by reducing the number of samples attacked.

From our experiment, we can deduce that integrating the state register into the BRAM is a good choice, despite the leakage of $L2$ because:

- It is cost-efficient, both in terms of area and speed.
- The input register is quite resistant against CPA attacks due to lack of amplification from routing buffers.
- Model $M2$ makes it harder to distinguish the keys than model $M1$, because the key appears outside the Sbox (for a given SNR, more traces are required, since the distance to the nearest rival decreases [19]). This is illustrated in Fig. 12 and **??**.
- As a corollary, a brute-force of 2^{16} is required to get the complete 16 bytes of the key from the 16 pairs of sub-keys $(k^\star, \neg k^\star)$ guessed by the attack.

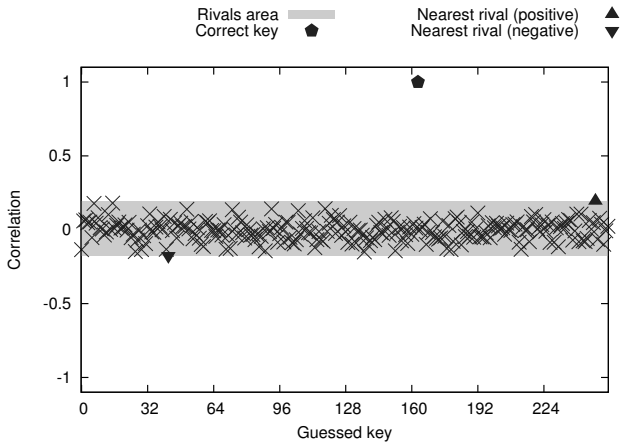

Figure 11: Simulated CPA values with model M1, and correct key $K_{10} = 163$ and for all the guessed keys $K \in [\![0, 255]\!]$.

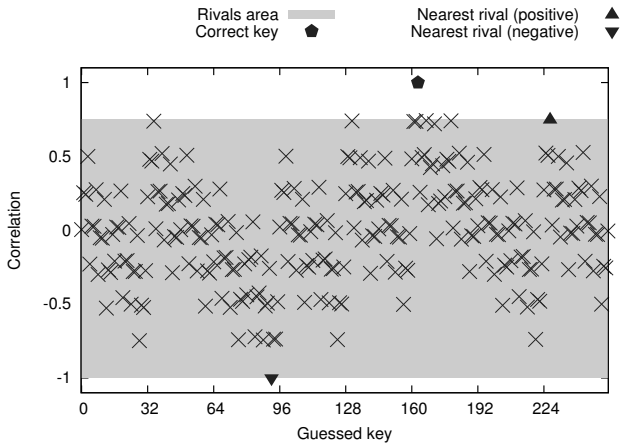

Figure 12: Simulated CPA values with model M2, and correct key $K_{10} = 163$ and for all the guessed keys $K \in [\![0, 255]\!]$.

6. Conclusions and Perspectives

In this paper, we investigated the leakage sources in a Xilinx Virtex V FPGA with the help of three different AES architectures. We found that implementations in logic and BRAM are equally vulnerable to SCA. BRAM which was considered more resistant to SCA as compared to FPGA logic, leaks significantly at the output latch instead of input register. The vulnerability of the BRAM output exists whether or not the optional output register is used. Indeed, when it is not the case, the output is latched, and thus all the accessed bits are leaking synchronously, as if they were sampled in a real DFF. Thus, countermeasures are essential to resist SCA on FPGA implementation of cryptographic applications. As mentioned previously, owing to the similarity of BRAM architecture in Altera devices as well, the results can be extended.

Our results can also be used to strengthen the implementation of countermeasures. For instance, a masking countermeasure [20] which may otherwise break-in a few million traces can be made stronger just by merging the masked state register with the address register of the BRAM. Even the dual-rail logic countermeasure can benefit from the presented results. A BRAM based dual-rail countermeasure [11] can use the input and output register of BRAM to implement the two stages of register which are closely placed.

As an extension to this work, we would like to assess the gain in term of security if the latch *L2* is protected by low-cost countermeasures like random precharge [21].

Acknowledgments

This research is partly supported by Strategic International Cooperative Program (Joint Research Type), Japan Science and Technology Agency (JST), and the French Agence Nationale pour la Recherche (ANR), via grant for project SPACES (Security evaluation of Physically Attacked Cryptoprocessors in Embedded Systems). The authors wish to thank Julien Francq and Antoine Wurker (EADS/Cassidian, Cyber Security Solutions Center) for insightful discussions about power attacks on AES and Grøstl [8].

References

[1] NIST/ITL/CSD, "Advanced Encryption Standard (AES). FIPS PUB 197," Nov 2001, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[2] E. Brier, C. Clavier, and F. Olivier, "Correlation Power Analysis with a Leakage Model," in *CHES*, ser. LNCS, vol. 3156. Springer, August 11–13 2004, pp. 16–29, Cambridge, MA, USA.

[3] E. Biham and A. Shamir, "Differential Fault Analysis of secret key cryptosystems," *CRYPTO 97, Springer 1997, LNCS*, vol. 1294, pp. 1513–1521, 1997.

[4] A. Moradi, M. Kasper, and C. Paar, "Black-Box Side-Channel Attacks Highlight the Importance of Countermeasures - An Analysis of the Xilinx Virtex-4 and Virtex-5 Bitstream Encryption Mechanism," in *CT-RSA*, ser. Lecture Notes in Computer Science, O. Dunkelman, Ed., vol. 7178. Springer, 2012, pp. 1–18.

[5] TELECOM ParisTech SEN research group, "DPA Contest (2nd edition)," 2009–2010, http://www.DPAcontest.org/v2/.

[6] NIST/ITL/CSD, "Data Encryption Standard. FIPS PUB 46-3," Oct 1999, http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf.

[7] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe, "PRESENT: An Ultra-Lightweight Block Cipher," in *CHES*, ser. Lecture Notes in Computer Science, vol. 4727. Springer, September 10-13 2007, pp. 450–466, Vienna, Austria.

[8] "Grøstl – a SHA-3 candidate," Accessed on Friday 2012/08/17, http://www.groestl.info/Groestl.pdf.

[9] S. Drimer, T. Güneysu, and C. Paar, "DSPs, BRAMs and a Pinch of Logic: New Recipes for the AES on FPGAs," in *IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 14-15 Apr 2008, pp. 99–108, stanford, Palo Alto, CA.

[10] S. Shah, R. Velegalati, J.-P. Kaps, and D. Hwang, "Investigation of DPA Resistance of Block RAMs in Cryptographic Implementations on FPGAs," in *ReConFig*, V. K. Prasanna, J. Becker, and R. Cumplido, Eds. IEEE Computer Society, 2010, pp. 274–279.

[11] S. Bhasin, S. Guilley, Y. Souissi, T. Graba, and J.-L. Danger, "Efficient Dual-Rail Implementations in FPGA using Block RAMs," in *ReConFig*. IEEE Computer Society, November 30 – December 2 2011, pp. 261–267, Cancún, Quintana Roo, México. DOI: 10.1109/ReConFig.2011.32.

[12] S. Bhasin, N. Selmane, S. Guilley, and J.-L. Danger, "Security Evaluation of Different AES Implementations Against Practical Setup Time Violation Attacks in FPGAs," in *HOST (Hardware Oriented Security and Trust)*. IEEE Computer Society, July 27th 2009, pp. 15–21, DOI: 10.1109/HST.2009.5225057; In conjunction with DAC-2009, Moscone Center, San Francisco, CA, USA.

[13] Xilinx, "Spartan-6 FPGA Block RAM Resources User Guide — UG383 (v1.5)," http://www.xilinx.com/support/documentation/user_guides/ug383.pdf.

[14] Altera, "Stratix-II Device Handbook — Volume 1," http://www.altera.com/literature/hb/stx2/stratix2_handbook.pdf.

[15] L. Batina, B. Gierlichs, E. Prouff, M. Rivain, F.-X. Standaert, and N. Veyrat-Charvillon, "Mutual Information Analysis: a Comprehensive Study," *J. Cryptology*, vol. 24, no. 2, pp. 269–291, 2011.

[16] E. Prouff and M. Rivain, "Theoretical and Practical Aspects of Mutual Information Based Side Channel Analysis," in *ACNS*, ser. LNCS, Springer, Ed., vol. 5536, June 2-5 2009, pp. 499–518, Paris-Rocquencourt, France.

[17] S. Guilley, L. Sauvage, J.-L. Danger, N. Selmane, and R. Pacalet, "Silicon-level Solutions to Counteract Passive and Active Attacks," in *FDTC*, L. Breveglieri, S. Gueron, I. Koren, D. Naccache, and J.-P. Seifert, Eds. IEEE Computer Society, 2008, pp. 3–17.

[18] C. Canovas and J. Clediere, "What do S-boxes Say in Differential Side Channel Attacks?" Cryptology ePrint Archive, Report 2005/311, 2005, http://eprint.iacr.org/.

[19] O. Benoît and T. Peyrin, "Side-Channel Analysis of Six SHA-3 Candidates," in *CHES*, ser. Lecture Notes in Computer Science, vol. 6225. Springer, August 17-20 2010, pp. 140–157, Santa Barbara, CA, USA.

[20] M. Nassar, Y. Souissi, S. Guilley, and J.-L. Danger, "RSM: a Small and Fast Countermeasure for AES, Secure against First- and Second-order Zero-Offset SCAs," in *DATE*, March 12-16 2012, pp. 1173–1178, Dresden, Germany. (TRACK A: "Application Design", TOPIC A5: "Secure Systems"). On-line version: http://hal.archives-ouvertes.fr/hal-00666337/en.

[21] D. Réal, V. Dubois, A.-M. Guilloux, F. Valette, and M. Drissi, "SCARE of an Unknown Hardware Feistel Implementation," in *CARDIS*, ser. LNCS, vol. 5189. Springer, 2008, pp. 218–227, London, UK.