Efficient FPGA Implementation of Dual-Rail Countermeasures using Stochastic Models

Shivam Bhasin, Sylvain Guilley, Youssef Souissi, and Jean-Luc DANGER.

Institut TELECOM / TELECOM ParisTech, CNRS LTCI (UMR 5141)
Département COMELEC, 46 rue Barrault, 75 634 PARIS Cedex 13, FRANCE.Email:
`firstname.lastname@TELECOM-ParisTech.fr`

Abstract. Dual-rail precharge logic (DPL) is a data hiding countermeasure against side channel attacks (SCA). Many variants of DPL have been introduced in the literature which target ASICs, FPGAs and microcontroller. A common problem which leads to failure of DPL on FPGA is imbalanced routing. FPGA designers have limited control over the FPGA placement and routing tools and therefore symmetrically routing a DPL design in FPGA is very difficult. Some FPGA tools like Xilinx ISE give the option to manually route the wires but for complex cryptographic circuits the number of wires are quite high which makes manual routing of every wire impractical. In this article, we briefly discuss methods which could reduce routing imbalance in dual-rail circuits when implemented in FPGAs. Nevertheless some imbalance is always present. Next we show how side channel tools can come handy to a designer in precisely estimating different aspects of leakage in the side channel. We compare template attacks, stochastic models and mutual information analysis in the given context. Results show that stochastic models are the most appropriate evaluation tool in this context and provide information on the leakage sources. Once this information is known, the leakage sources can be manually balanced.

Keywords: Dual-rail precharge logic (DPL), Stochastic Models, Template Attacks, Mutual information analysis (MIA).

1 Introduction

Complex system often use cryptographic cores to encrypt communication on the system bus. This communication is as secure as the cryptographic core. Modern cryptographic algorithms are mathematically secure. However their physical implementation can be compromised which shifts the responsibility to provide a secure cryptographic core from cryptographers to designers. Therefore it is essential that a designer should have accurate information about the flaws in the implementation. Often physical countermeasure are used to protect the cryptographic core. It is equally important for a designer to analyze the effectiveness of these countermeasures. For a cryptographic core to be secure it should resist, to an extent, against both passive and active attacks. Passive attack are mounted by observing the physical leakages from the system, like power (Differential Power Analysis, or DPA [1]) or E/H field (Electromagnetic Analysis, or EMA [2]). Active attacks involve perturbing the system using faults during the execution of a cryptographic algorithm. Malicious techniques based upon the variations of supply voltage, clock frequency, temperature variation, or irradiation by a laser beam will most probably lead to a wrong computation result that can be exploited.

State of the art countermeasures can be widely classified into two categories *i.e.* information masking and information hiding. Masking [3] countermeasures rely on confusing the attacker. A random generated mask is used during execution of the algorithm which has an impact on the intermediate states without affecting the end result. Owing to this technique, the attacker observes leakage corresponding to mask and not the actual key bits. Although a nicely masked circuit can resist first order passive attacks but higher order attacks can still compromise the security of the design. Also masking does not provide any extra resistance against faults compared to an unprotected implementation.

Information hiding as the name suggests hides the information from attacker. The algorithm is implemented in such a way that leakage remains constant irrespective of the computations performed. Dual-rail precharge logic (DPL) [4] is a countermeasure based on information hiding. The principle of this countermeasure is to generate a design equivalent and with opposite behaviour of the target design such that every part of the circuit is perfectly balanced. This way the activity of the design remains constant and completely independent of the data processed. In DPL, every variable a involved in the algorithm is actually mapped into a couple of variables, named $(a_\mathrm{F}, a_\mathrm{T})$, and called the 'false' and 'true' halves of the dual-rail variable a. Similarly every function f is replaced by a couple f_T, f_F. The couple $(a_\mathrm{F}, a_\mathrm{T})$ alternates between two phases:

1. Precharge: Variable takes values $(0, 0)$ or $(1, 1)$, called NULL0 or NULL1, and designated as a NULL token, playing the role of spacer, and
2. Evaluation: Variable takes values $(1, 0)$ or $(0, 1)$, called VALID0 or VALID1, and designated as a VALID token, carrying the value of a.

One DPL computation alternates NULL and VALID tokens, with the remarkable property that exactly one bit toggle occurs in each transition. A pair of gates $(f_\mathrm{F}, f_\mathrm{T})$ respects the DPL convention if:

– It propagates the NULL values, *i.e.*, if all the inputs are NULL, then $(f_\mathrm{F}, f_\mathrm{T})$ is also NULL.
– It propagates the VALID values, *i.e.*, if all the inputs are VALID, then $(f_\mathrm{F}, f_\mathrm{T})$ is also VALID.

There are some countermeasures which combine hiding and masking techniques in order to achieve higher level of security.

Minor imbalances in DPL can leak exploitable information. Few works [5] [6] [7] show practical attacks on various variants of DPL. This problem is bigger in FPGA as designers have partial control over the FPGA tools. In this article, we first discuss the methods generally applied to implement DPL efficiently in FPGA. Then we show how side channel evaluation tools can be deployed to further improve the implementations on modern FPGAs.

Profiled side-channel attacks are the most powerful type of side-channel attacks. They are divided in two different stages. First stage known as a profiling phase uses a training device identical to the target which allows precise characterization of its physical leakage. Second stage which is an online exploitation phase is mounted on the target device in order to perform a key recovery. Standard profiled side-channel attacks include template attacks and stochastic models, respectively introduced in [8] and [9] respectively. Template attacks can precisely estimate the maximum information present in the side channel leakage. Stochastic models which may underestimate the information leakage are capable of localizing the source of leakage. There are also non-profiled attacks like mutual information analysis (MIA) which can be equally deployed. We carry out a comparative study of these three evaluation tools on simulated and real traces. We also show that in case of DPL, stochastic model can reveal the potential imbalance which the designer needs to fix.

The rest of the paper is organised as follows. In section 2, we discuss various methods used to counter the imbalance in DPL implementations on FPGA. Then in section 3 we propose a methodology to improve DPL implementations on FPGA. Section 4, briefly describe the evaluation tools which are used to evaluate DPL designs. This is followed by section 5 where the experimental behavior of the evaluation on simulated and real traces are shown with a comparative discussion on pros and cons of each evaluation tool. Finally, conclusions are drawn in section 6.

2 Methods to Reduce DPL Imbalance

DPL, in general, suffers from two major flaws i.e. early propagation effect (EPE) and imbalance. EPE is caused by difference of switching time between the true and false net. This difference causes data-dependant leakage on the side-channel. EPE has already been dealt in various publications ([10], [11]) and is out of the scope of this paper. Imbalance in DPL comes from two major sources. The primary source of imbalance is asymmetry between the routing of true (a_{T}) and false (a_{F}) signal. In FPGA, this imbalance is generally caused by automatic place and routing. Ideally, the true and false part of a DPL gate should be placed and routed identically. However, in order to optimize the design, FPGA tools may place and route them differently which can leak information in the side channel. The secondary source of imbalance is the difference in functions f_{F} and f_{T}. If f_{F} has a different power consumption from f_{T}, the imbalance can be exploited. This imbalance is of main concern in ASIC. In FPGA because each function is composed of standard configurable logic blocks (CLB) which have more or less uniform power consumption.

One of the first solution proposed to counter routing imbalance was based on randomization or path switching in MDPL [12] by using a majority function.

The idea is to randomly swap between true and false routes in order to remove the bias from one such path. In other words, some logic is added on top of DPL gate to swap randomly the logic interconnect pairs, in a view to statistically balance the routing mismatches. The problem with masking of DPL is that the size of a basic cell capable of path switching can be complex and huge. The implementation of such complex gates in FPGA can pose practical problems. Also since a single bit of mask is used, the masked can be attacked [13].

Two other solutions were proposed for separable DPL styles. The first is applied to Divided WDDL [4] which is based on copy and paste placement. It comprises of placing a single rail design efficiently which is then duplicated and inverted to derive the false part. The true and false networks are placed adjacent to each other. With the latest advances in side channel acquisition technique, it is possible to isolate the true network from the false. This scenario is similar to attacking a single rail design. The second solution called interleaved placement was recently introduced in [14]. This comprises of placing the true network of DPL design in alternate rows/columns of a mesh FPGA. The false network is then placed in empty rows/columns left between two rows/columns of the true network. Such placement does not allows isolation of true and false network in a DPL design which provides adequate security.

In [10], authors propose a non-separable DPL style with in-built synchronization scheme. Synchronization was able to counter early evaluation effect (another drawback in DPL [15]), still the attack was possible due to routing imbalance in certain sections of the circuit. Their analysis revealed that large fanout of registers and high gate count in timing path cause major leakage in the side channel. Few methods, mainly for ASICs, have been proposed to remove the bias due to place and route [16, 17]. Since designer does not have complete control over the FPGA tools, applying these techniques is not simple on FPGAs.

A symmetric cryptographic algorithm in general consist of non-linear substitution box (S-box) and diffusion functions. If S-box is implemented in logic, many nets have high fanout. Fanout is also high for diffusion functions. Diffusion in hardware is done by swapping wires and combinatorial logic. Since the diffusion is quite complex for cryptography often a single gate drives multiple gates. Also these functions need many gates for computation and the timing path is long. Thus FPGA placement and routing tool, will place all the gates driven by the same inputs nearby for resource optimization. Since DPL has a complementary path with same fanout, it is difficult to provide identical routing to the two complementary net. Timing imbalance is also increased with high fanout. It can be roughly expressed as $\Delta T = K \times F$ where K is the constant capacitance and F is the fanout. In other words, high fanout means more chance that there is an imbalance, and exacerbated by the increased local congestion. S-box and diffusion function also have long timing path which cause further imbalance.

A simple way to reduce fanout and timing path in a cryptographic circuits is to use memories. Typically, ROMs can make up for complex unstructured or structured high algebraic degree combinational blocks. In FPGA, the density of user logic is about 30 times less than that of macros [18]. Therefore, using

ROMs can drastically reduce the overall design area and power consumption. For example, AES can be s-box and mixcolumns are the 2 operations which have high fanout. T-boxes were introduced in original Rijndael proposal are sets of 8x32 look-up tables which combine SubBytes, ShiftRows and MixColumns and can be implemented in memory. Its FPGA implementation is discussed in [19]. Therefore t-boxes can be used to reduce fanout and length of timing path.

3 Our Proposition

Analyzing the three proposed solution, masked DPL increases the size of the basic DPL gate and pose optimization problems. Interleaved routing seems a good solution but till now it has been applied only to separable DPL style like SDDL. It is not clear how can interleaved routing be done with non-separable DPL. Therefore we stick with the third solution. Our proposition to implement robust DPL on FPGA are:

1. Design DPL circuits with low fanout and automatic placement and routing.
2. Use advanced evaluation tools to find the imbalance bit.
3. Route them symmetrically using manual routing tools.
4. Iterate from step 2 until desired security achieved.

Step 1: We propose to design a DPL circuit with low fanout and limited number of levels in a timing path. This can be done by using memories. As mentioned earlier modern cryptographic circuits like AES can be implemented in memory using T-boxes. DPL implementation using memory is tricky and needs special care during the implementation step. DPL styles can use memories in two different ways. The first way which involves duplication of the RAM is costly. By duplication we mean duplication of the I/O. This duplication of I/O causes an exponential increase in cost of memory. An AES S-box which fits in $2^8 \times 8$ bits (2Kb) needs $2^{16} \times 16$ bits (1Mb) after duplication. Another approach is to use a true and a false RAM of size $2^8 \times 8$ each with special circuitry at the output to propagate the precharge as shown in figure 1. The net cost of RAM is increased by a little over two. However, this low-cost implementation (in figure 1) is vulnerable to glitches and the input of AND gate can leak information if not implemented properly. In figure 1, the input of the AND gate can be attacked. Also special routing is required for the precharge signal to the RAM output which is generally limited at the global DPL inputs.

Thus in other DPL styles, using memories without glitches may have an exponential area overhead. Recently, a new DPL style BCDL [11] was introduced which has a global synchronization signal as a special feature. Owing to presence of this global synchronization signal an AES s-box will need $2^9 \times 8$ bits (4Kb) of memory for each $Sbox_{\mathrm{T}}$ and $Sbox_{\mathrm{F}}$. Thanks to the global synchronization signal, the memory utilization is increased by only 2^{n+2} and not 2^{2n}. Similarly the cost of T-box will be increased by 4 times in BCDL.

The rest of the circuit can be synthesized, placed and routed automatically by the FPGA.

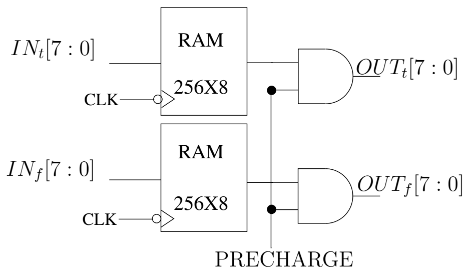

Fig. 1. A low cost DPL S-box in RAM.

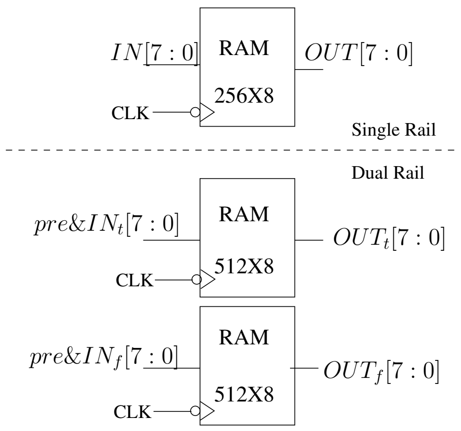

Fig. 2. BCDL S-box in RAM.

Step 2: Once the design has been routed a thorough evaluation should be conducted. The evaluation should not be superficial but evaluate every bit. The evaluation tool which can be deployed are covered in the next section. This step is practical in our solution because a circuit with low fanout and very few gates in the logic will have lesser wires which can potentially be imbalanced. The evaluation can precisely point out which bit pose a security problem.

Step 3: Once the source of leakage is known, it can be balanced manually. When memories are not used in a DPL design the fanout is increased and the number of logic cell used are significantly increased. This makes it impractical to balance manually. For example, two dual rail signal are shown in figure 3 and 4 with a fanout of 2 and 8 respectively. It is evident that the imbalance is more in a signal with higher fanout and it is harder to balance it manually. Therefore not a lot of effort is needed to balance the signal in fig 4.

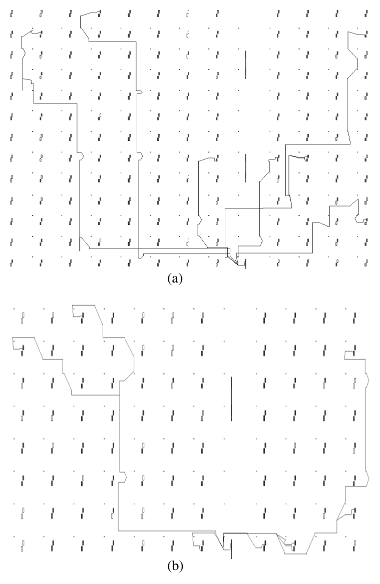

Fig. 3. Difference in routing of (a) true and (b) false signal with fanout 8 (not to scale).

Step 4: After the leaking bits are balanced the evaluation step is repeated to ensure if there are other bits which still leak and fix them.

4 Evaluation Tools

To evaluate a DPL implementation, mono-bit DPA is one of the most used analysis technique unlike single-rail implementations where multi-bit DPA performs better when based on hamming weight or hamming distance model. In a DPL implementation, the power consumption difference between different bit-flips ($0 \rightarrow 1$ & $1 \rightarrow 0$) can be exploited best on a single bit [5]. The leakage

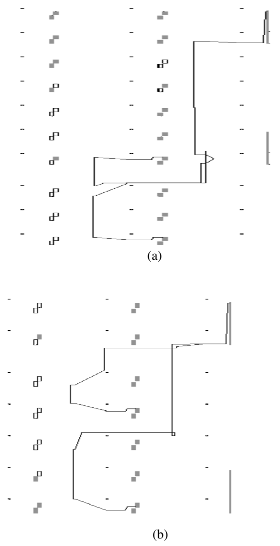

Fig. 4. Difference in routing of (a) true and (b) false signal with fanout 2 (not to scale).

in DPL is caused by the imbalance between the true and the false network, and this imbalance could be opposite for targeted bits, and therefore counterbalance themselves when combined. Therefore mono-bit attacks perform better. Another technique used against DPL implementations is Mutual Information Analysis (MIA [7] [10]). Profiled attacks are also used commonly for evaluations, however to our knowledge, it has not been applied on DPL implementations before. In the following, we briefly describe MIA and profiled attacks. Section 5 covers the practical implementation of these three attacks on DPL implementations.

4.1 Mutual Information Analysis

MIA was introduced as a side channel analysis tool in [7]. It calculates the mutual information between a sensitive variable X and a side channel leakage L, measured in bits is:

$$I(X;C) = \sum_{x}\sum_{c} P(x,c) \log \frac{P(x,c)}{P(x)P(c)} \tag{1}$$

This can be further simplified as:

$$I(X;L) = H(X) - H(X|L) = H(L) - H(L|X) \tag{2}$$

Here $H(X)$ gives the entropy of X and $H(X|L)$ gives the conditional entropy of X knowing L. Many methods have been proposed to estimate entropy like histograms, kernel density functions, Gaussian parametric estimators etc [20]. In our experiments we use the Gaussian parametric estimation where the distribution of X, L and the joint distribution of X, L is assumed to be Gaussian. In this case entropy can be calculated as a function of standard deviation σ_x of X as:

$$H(X) = \sum_i p(x_i)\log_2(p(x_i)) = \log_2(\sigma_x\sqrt{(2\pi e)})$$

4.2 Template Attacks

Template attacks were introduced by Suresh Chari *et al.* in [8]. The salient feature of template attacks is that it characterizes the noise in measurements, unlike other approaches. The main idea is to capture an amount n of traces $C_{M,k}(t)$ (typically $n = 1000$) on the programmable device for each subkey k and to describe the behaviour of the noise depending on k. Each set of $C_{M,k}(t)$ is averaged, to obtain a new set $\mathbf{A} = \{\mathbf{A}_k,\ \forall k \in K\}$. In order to reduce the profiling time, a set of point of interest has to be selected.

Let $\mathbf{T} = \{t_i,\ 1 \leq i \leq p\}$ be a set of p points of interest. For a given key k, we can now compute a noise vector for each traces $C_{M,k}(t)$ as follows:

$$\mathbf{N}_k(M) = [C_{M,k}(t_1) - \mathbf{A}_k(t_1), \ldots, C_{M,k}(t_p) - \mathbf{A}_k(t_p)] \tag{3}$$

Let $\mathbf{N}_{k,t}$ be the vector of all elements of $\mathbf{N}_k$ at the instant t. Now, we can compute the covariance matrix which has its elements defined as:

$$\Theta_k[t_i, t_j] = cov(\mathbf{N}_{k,t_i}, \mathbf{N}_{k,t_j}) \tag{4}$$

The couple $(\mathbf{A}_k, \Theta_k)$ is the template for the key k. Profiling phase is finished when a template is computed for each key $k \in K$.

The key extracting phase uses the maximum likelihood principle. For each key k and for each measured traces, we compute a noise vector $\mathbf{n}$ on the points of interest (using $\mathbf{A}_k$). Thereafter we compute $f_k(\mathbf{n})$, where f_k is the multivariate Gaussian distribution, as follows:

$$f_k : \mathbb{R}^p \to \mathbb{R} \qquad f_k(\mathbf{n}) = \frac{1}{\sqrt{(2\pi)^p.|\Theta_k|}} e^{-\frac{1}{2}\mathbf{n}^{\mathrm{T}}\Theta_k^{-1}\mathbf{n}} \tag{5}$$

where $|\Theta_k|$ is the determinant of Θ_k. $f_k(\mathbf{n})$ will give the highest value if k is the good guess. It gives the probability of each key candidate. Once the probability of each key candidate is known, we can compute the entropy and eventually mutual information using (2).

4.3 Stochastic Model Attack

Stochastic Models [9] are also a type of profiled attacks slightly. The profiling phase needs only one test key i.e. the power consumption is modeled, at a time t as follows:

$$W_t(x, k) = h_t(x, k) + \mathcal{B}_t \tag{6}$$

where x is the plain text and k the key. The first summand h_t is the deterministic part of the power consumption (which depends on x and k) and $\mathcal{B}_t$ a random noise with zero expectation ($\forall t, \mathbb{E}(\mathcal{B}_t) = 0$). The first profiling step consists in approximating h_t, followed by estimation of $\mathcal{B}_t$ using h_t. h_t is assumed to have the *EIS* property (Equal Image under different Subkeys), which implies that only one test key is needed for the profiling phase. Let $\tilde{h}_t$ be the best estimation of h_t computed as:

$$h_t(x,k) = \beta_0 + \sum_{i=1}^{u} \beta_{it} g_i(x,k) \tag{7}$$

where the g_i are chosen base functions, which depend on x and k, and β_{it} are coefficients, which estimates the system. It is the choice of base functions which define the degree of stochastic models. A linear model takes just a function of individual bits where as a higher degree model considers multiple bits for each co-efficient. We assume that β_0 is always equal to 1. As we attack the output of an AES s-box, in the linear model we take the 8-bits at the output of the s-box as base vectors giving base function of length 9 (8 bits and 1 constant). For the second degree base vectors we take the product of all the individual bits which gives us 37 (1+8+28) base vectors. Similarly base vectors of degree 3 and 4 are of length 93 and 163 respectively. The second step of the profiling phase consists of characterization of the noise. First, some relevant instants have to be selected (*e.g.* by using the T-Test or Euclidian norm of the coefficients β_{it}). The noise is characterized by constructing the probability density function of the multivariate normal distribution, using a covariance matrix (computed with a noise random variable associated on each point of interest). When the first device is profiled, attack can be performed using the maximum likelihood principle.

5 Experimental Results

We conducted two sets of experiments. The first experiment is based on simulated traces. The aim of this experiment is to observe the behavior of each of the three evaluation tools when the environmental noise is varied. Thereafter, we test these tools on real traces acquired from a AES implementation protected with DPL running on a Altera Stratix FPGA.

5.1 Simulated traces

This experiment was conducted on simulated traces l_i given by:

$$l_i = HW(v_i) + n_i \tag{8}$$

Here HW is the Hamming Weight function while n_i is the added Gaussian noise. l_i is a linear leakage. We conducted various experiments, each time varying the variance of added noise. Let $v_i = Sbox^{-1}(x \oplus k)$ is the intermediate value i.e. one byte at the output of round 9 of AES based on which the profiling was performed. We built templates as well as stochastic model of degree 1,2,3 and 4 for the value v_i to compute the mutual information. We also compute the mutual information using MIA with Gaussian parametric estimation. Figure 5(a) shows that when the leakage is perfectly linear (l_i) stochastic model of linear degree are as accurate as template attacks.

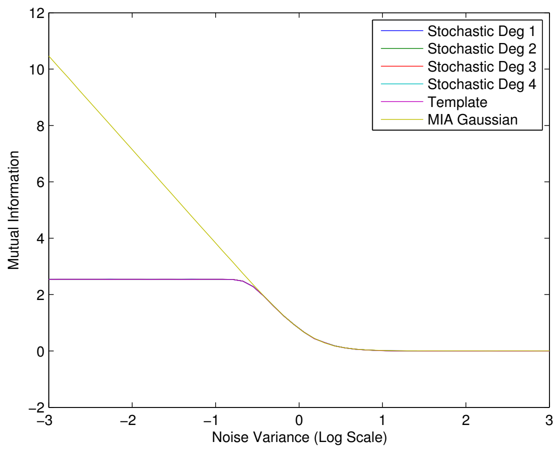

Fig. 5. Mutual information for simulated traces with linear leakages model as a function of added Gaussian noise.

An interesting observation is that when the added noise is very low, the MIA using Gaussian parametric estimation overestimates the information present in the traces. However, when we increase the level of noise then information estimated by MIA decays rapidly and becomes equivalent to templates and stochastic models. This means that for higher level of noise, MIA using Gaussian parametric estimation can be as reliable as templates or stochastic models. Next we apply these methods on real traces to check the validity of our assumption.

5.2 Real Traces

After studying the effectiveness of the evaluation tools on simulated traces, we decided to test them on real traces acquired from FPGA implementation of AES protected with DPL. The implementation details of DPL used is given in [10] however the findings can be extended to any DPL implementation. We took power consumption measurements (*traces*), using an *electromagnetic probe* capturing the field of a leaking capacitor on the back-side of the FPGA core with a *54855 Infiniium oscilloscope* from Agilent Technologies. In order to reduce acquisition noise, each trace was averaged 64 times. When dealing with simulated traces, the Gaussian noise assumption works fine because the introduced noise is perfectly Gaussian. However, this might not be case with real traces. We acquired a set of 90000 traces. 50000 traces were used to in the profiling stage i.e. building templates and approximating h_t and $\mathcal{B}_t$. For the attack a set of 10000 traces was taken. Figure 6 shows the mutual information calculated using templates and stochastic model of degree 1,2,3 & 4. We equally computed MIA with Gaussian parametric estimation on 90000 traces.

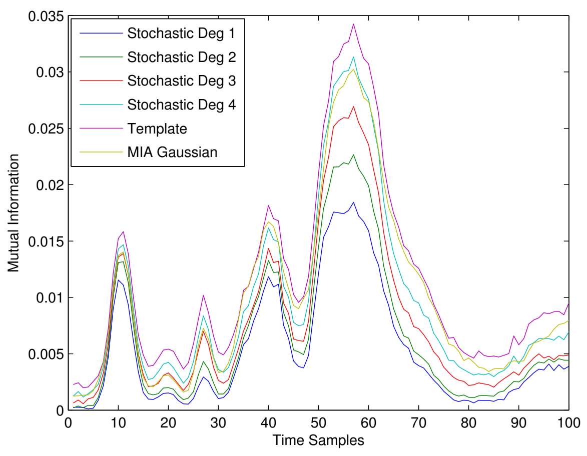

Fig. 6. Perceived information estimated using templates and stochastic model of degree 1,2,3,4.

Information estimated by templates is more than the stochastic model of degree 1. This means that the leakage is not linear. Stochastic model of degree 2,3 and 4 add further information. Information estimated by stochastic model of degree 4 is approaching the information estimated by templates. Therefore in this case computing a stochastic model of degree 4 which is computationally less resource consuming will be as good as template. An advantage of stochastic models over templates is that stochastic models are capable to localise the leakage source. In this case, if we see the linear coefficients it shows that 3^{rd} bit of the targeted s-box is leaking much more than other bits (figure 7). This information is critical for designer who wants to improve his design. This information was called β-characteristics in [21]. It is clear from figure 7(a) that it is only bit 2 that is leaking.

On the other hand, MIA using Gaussian parametric estimation does not overestimates the information. It seems to perform almost as good as stochastic models and templates. We also performed a mono-bit MIA on the same traces. The results are shown in figure 7(b) where its clear that only bit 2 is leaking. Analysis of the FPGA floorplan confirmed that this particular bit was asymmet-

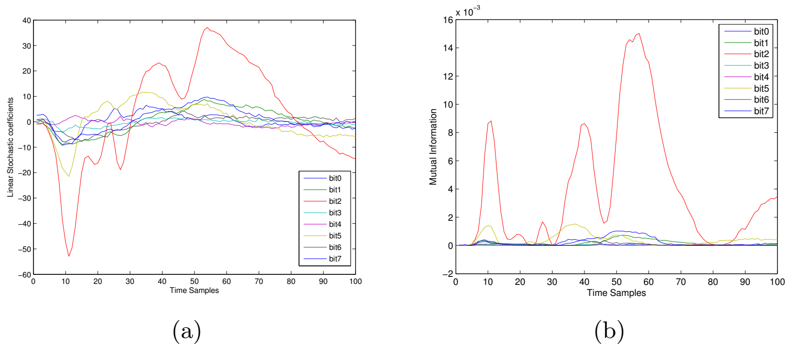

Fig. 7. Localizing the leakage using (a) Linear Stochastic coefficients and (b) Mono-bit MIA.

rically routed. Thus mono-bit MIA has an advantage of pinpointing the leakage like stochastic coefficients.

5.3 Discussion

From the experiments presented above we can infer that template attacks provide the best estimation but are not capable of pinpointing the vulnerabilities. On the other hand stochastic models of first degree is as good as template when the leakage is perfectly linear. Templates are more efficient when the leakage is not perfectly linear which is common in real traces. Higher degree stochastic models can be explored to approach the estimation of templates in such cases. The main advantage of stochastic models is that they are capable of pinpointing the vulnerabilities. In case of DPL, it can tell exactly which bits are leaking. Stochastic models can also outperform templates when the number of traces are insufficient as stochastic models performs an approximation of lesser degrees of freedom. Finally MIA using Gaussian parametric estimation can sometimes overestimate the information. In our case, the traces acquired were quite noisy and it seemed to work fine nevertheless the risk of overestimation is present. MIA is also able to pinpoint the leakage as in Stochastic models when used in mono-bit mode. An advantage of MIA using Gaussian parametric estimation is that they are easy and faster to mount and no profiling is required. Comparing the three we can say that stochastic models are best suited for evaluating DPL designs because they are capable of correctly estimating information and localizing the leakage.

6 Conclusion

In this article, we present a methodology to efficiently implement DPL in FPGAs. The problem of imbalanced routing of DPL in FPGA is widely known. Our solution is based on an iterative approach. It begins with designing a DPL with cells having low fanout and using memories. Then the design is automatically placed and routed by the FPGA tools followed by a detailed evaluation. A careful evaluation will reveal the leakage sources and which can be balanced manually. Designing DPL using low fanout is important because it can significantly reduces the number of cells to be balanced. We equally compared common evaluation tools which are template, stochastic models and MIA using Gaussian parametric estimation. On comparing them we found that stochastic models are most appropriate for evaluating DPL implementations. Templates cannot localize the leakage while MIA can overestimate the information.

Further research can focus on two main direction. As observed during the analysis using stochastic model, higher degree stochastic model does contain some information. It would be interesting to work on the physical significance of higher order leakages and other methods which can exploit it. The other research direction has more of an experimental flavor to deal with issues like shifting of leakage point (decoupling capacitor).

Acknowledgments

This work has been supported by the french national research agency (ANR), through the `ANR-07-ARFU-010` grant "SeFPGA" (Secured Embedded FPGAs). Authors also acknowledge interesting discussions with **Dr. F-X Standaert** from Universite Catholique de Louvain, Belgium on evaluation tools for DPL.

References

1. Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: Proceedings of CRYPTO'99. Volume 1666 of LNCS., Springer-Verlag (1999) 388–397
2. Agrawal, D., Archambeault, B., Rao, J.R., Rohatgi, P.: The EM Side-Channel(s). In: CHES. Volume 2523 of LNCS., Springer (2002) 29–45
3. Akkar, M.L., Giraud, C.: An Implementation of DES and AES Secure against Some Attacks. In LNCS, ed.: Proceedings of CHES'01. Volume 2162 of LNCS., Springer (2001) 309–318 Paris, France.
4. Tiri, K., Verbauwhede, I.: A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation. In: DATE'04, IEEE Computer Society (2004) 246–251 Paris, France. DOI: 10.1109/DATE.2004.1268856.
5. Sauvage, L., Guilley, S., Danger, J.L., Mathieu, Y., Nassar, M.: Successful Attack on an FPGA-based WDDL DES Cryptoprocessor Without Place and Route Constraints. In: DATE, Nice, France, IEEE Computer Society (2009) 640–645
6. Moradi, A., Salmasizadeh, M., Shalmani, M.T.M.: Power Analysis Attacks on MDPL and DRSL Implementations. In: ICISC. Volume 4817 of Lecture Notes in Computer Science., Springer (2007) 259–272 Seoul, Korea.
7. Gierlichs, B., Batina, L., Tuyls, P., Preneel, B.: Mutual information analysis. In: CHES, 10th International Workshop. Volume 5154 of Lecture Notes in Computer Science., Springer (2008) 426–442 Washington, D.C., USA.
8. Chari, S., Rao, J.R., Rohatgi, P.: Template Attacks. In: CHES. Volume 2523 of LNCS., Springer (2002) 13–28 San Francisco Bay (Redwood City), USA.

9. Schindler, W., Lemke, K., Paar, C.: A Stochastic Model for Differential Side Channel Cryptanalysis. In LNCS, ed.: CHES. Volume 3659 of LNCS., Springer (2005) 30–46 Edinburgh, Scotland, UK.
10. Bhasin, S., Guilley, S., Flament, F., Selmane, N., Danger, J.L.: Countering Early Evaluation: An Approach Towards Robust Dual-Rail Precharge Logic. In: WESS, ACM (2010) 6:1–6:8 Scottsdale, Arizona, USA. DOI: 10.1145/1873548.1873554.
11. Nassar, M., Bhasin, S., Danger, J.L., Duc, G., Guilley, S.: BCDL: A high performance balanced DPL with global precharge and without early-evaluation. In: DATE'10, IEEE Computer Society (2010) 849–854 Dresden, Germany.
12. Popp, T., Mangard, S.: Masked Dual-Rail Pre-charge Logic: DPA-Resistance Without Routing Constraints. In: Proceedings of CHES'05. Volume 3659 of LNCS., Springer (2005) 172–186 Edinburgh, Scotland, UK.
13. Schaumont, P., Tiri, K.: Masking and Dual Rail Logic Don't Add Up. In: CHES. Volume 4727 of LNCS., Springer (2007) 95–106 Vienna, Austria.
14. Velegalati, R., Kaps, J.P.: Improving security of SDDL designs through interleaved placement on Xilinx FPGAs. In: Field Programmable Logic and Applications, FPL 2011, IEEE (2011) accepted, to be published.
15. Suzuki, D., Saeki, M.: Security Evaluation of DPA Countermeasures Using Dual-Rail Pre-charge Logic Style. In: CHES. Volume 4249 of LNCS., Springer (2006) 255–269 Yokohama, Japan. `http://dx.doi.org/10.1007/11894063_21`.
16. Baddam, K., Zwolinski, M.: Divided Backend Duplication Methodology for Balanced Dual Rail Routing. In: CHES. Volume 5154 of LNCS., Washington, DC, USA, Springer (2008) 396–410 DOI: 10.1007/978-3-540-85053-3_25.
17. Tiri, K., Verbauwhede, I.: Place and Route for Secure Standard Cell Design. In Kluwer, ed.: Proceedings of WCC / CARDIS. (2004) 143–158 Toulouse, France.
18. Kuon, I., Rose, J.: Measuring the Gap Between FPGAs and ASICs. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **26** (2007) 203–215
19. Fischer, V., Drutarovský, M.: Two Methods of Rijndael Implementation in Reconfigurable Hardware. In Çetin Kaya Koç, Naccache, D., Paar, C., eds.: CHES. Volume 2162 of Lecture Notes in Computer Science., Springer (2001) 77–92
20. Prouff, E., Rivain, M.: Theoretical and Practical Aspects of Mutual Information Based Side Channel Analysis. In Springer, ed.: ACNS. Volume 5536 of LNCS. (2009) 499–518 Paris-Rocquencourt, France.
21. Kasper, M., Schindler, W., Stöttinger, M.: A stochastic method for security evaluation of cryptographic FPGA implementations. In Bian, J., Zhou, Q., Athanas, P., Ha, Y., Zhao, K., eds.: FPT, IEEE (2010) 146–153