Countermeasures Against Physical Attacks in FPGAs

Benoit Badrignans, Florian Devic, Lionel Torres, Gilles Sassatelli, Pascal Benoit

Chapter 4
Countermeasures Against Physical Attacks in FPGAs

J.-L. Danger, S. Guilley, L. Barthe, and P. Benoit

Abstract This chapter presents a set of countermeasures against physical attacks specifically dedicated to FPGA. Countermeasures as masking, hiding, are first discussed. Then we give a set of information and an overview on different logic style designed to be robust against SCA. The main objective herein is to compare these techniques and show that they can be suitable and implementable for FPGA components. A comparison of different logic style will conclude this chapter.

4.1 Introduction

Off the shelf FPGAs are often used for high-end applications that require embedded cryptography. State of the art commercial FPGA technologies do not have any specific feature to withstand side channel attacks that target the user application. For this reason, methods to protect them have to be designed at the logic and back-end levels of the design stages. Many countermeasures have been developed for ASICs and this can be used as a basis to develop specific protections for FPGAs. However the regular FPGA tiling structure and the huge space of interconnect and programmable switches may limit or reduce the robustness of countermeasures originally designed for ASICs. In this chapter we provide some answers to attacks and countermeasures embeddable in off the shelf FPGAs, with a special focus on side-channel attacks in symmetrical cryptography.

The protection of cryptographic IPs against side channel attacks at logical level is currently not so advanced in FPGAS as it is in ASICs, even though at first sight, FPGAs appear to have an intrinsic structure which makes them much more vulnerable:

- ASIC protections at back-end level (such as the fat wires [59] or back-end duplication [20] methods) are hardly feasible in FPGAs because of constrained and *a priori* unknown or limited routing resources.
- The interconnection makes up the largest part of the FPGA [26]. It includes lines, pass transistors, transmission gates, bidirectional buffers, switch matrices

J.-L. Danger (✉)
Telecom Paris-Tech, Paris, France
e-mail: danger@enst.fr

and connection boxes. These components increase the capacitive load of the interconnection and thus overall power consumption. This particularity facilitates passive attacks.
- D Flip-Flop (DFF) are numerous and fast. There are two reasons for their rapidity: they speed up processing and fight metastability. However they greatly increase power consumption increase. Moreover attacks on register nodes can lead to knowledge of the activity of combinatorial nodes that contain the secret information.
- The use of switches like pass transistors makes the power consumption model quite specific to FPGAs in which the current can vary according to a higher order equation in V_{dd} [13, 14].

With regards to ASICs, a study by Kuon and Rose [26] shows that FPGAs are on average 35 times bigger and consume 12 times more than ASICs. To withstand attacks on programmable devices that are not specifically designed for security, it could be worthwhile designing dedicated logic styles and implementation methods. In this chapter we provide an overview of current techniques and focus on different countermeasures (masking, hiding), and logic styles (WDDL, STTL, BCDL) specifically designed to fight Side Channel Attacks (SCA). Our aim is to show the potentials and limitations of countermeasures in FPGAs that could obstruct or prevent attacks.

4.2 Countermeasures Against Side Channel Attacks in FPGAs

The types of countermeasures that are used to protect hardware devices against SCAs can be considered at levels Protocol, Architecture and Netlist.

Protection of protocol and of architecture is independent of the technology (either software or hardware). However the implementation in FPGAs can take advantage of hardware properties like concurrent computation, reprogrammability and high levels of algorithmic noise.

The Protocol level can merely consist in regular key changes as described in [25, 32] in order to prevent the adversary from accumulating enough traces to be able to attack. One advantage of this level of protection is that it is provable. As an illustration of this technique, Fig. 4.1 shows that at every 100th ciphering operations the key is changed, thus limiting the scope of the attack if more than 100 traces are needed.

However this type of protection can be time consuming and costly as it requires a specific key exchange or synchronization mechanism to ensure Alice and Bob always use the same key. Rather than changing the key, the FPGA offers the possibility to reprogram the implementation. As mentioned by F.-X. Standaert [51], this feature certainly merits further research as only a few studies have been conducted so far. Among them Chaudhuri et al. [9] details a dedicated FPGA architecture using agility to thwart SCAs, whereas Mentens [33] explains how security can be enhanced with dynamic reconfiguration.

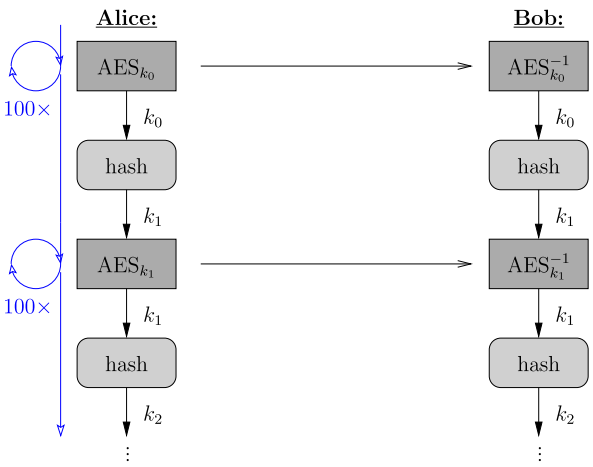


Fig. 4.1 Dynamic key change by hashing every 100th encryptions

Strengthening security at architectural level is certainly the least constraining method as it can be applied in all technologies that have logical model. For hardware implementations, this corresponds to the Register Transfer Level (RTL). One of the major countermeasures in this category is "masking" which consists in processing masked data rather than the data itself. The goal of the protection by masking is to randomize the power consumption and thus decorrelate as far as possible the computation activity from the secret information, which is generally the key for cryptographic algorithms. For this reason, masking provides a constant power consumption mean. As explained in detail in the next section, the mask has to be saved or processed.

Another means of providing protection consists in obfuscating the computation by generating noise which decreases the Signal to Noise Ratio "SNR" or more precisely the "leakage" to noise ratio. This makes the secret signal indiscernible. For instance, extra glitches in combinational gates can be added or extra jitter noise inserted at the clock stage. In software this can lead to the insertion of dummy instructions. Theoretically, this type of countermeasure is not very robust as the adversary can increase SNR by using more traces. For instance, the extra noise generated by the increase in pipelining reported in [53] or by unrolling the implementation as reported in [5] do not provide efficient protection against SCA.

At both architecture and netlist levels, one of the most efficient protection technique relies on the use of a differential logic. The efficiency of this type of countermeasure, often called "hiding", is based on attempting to make the power consumption constant by using dual-rail logic split in `True` and `False` networks. The rationale is to have one network consume power, while the other does not. One advantage of the hiding technique is to provide natural protection against fault attacks as explained in Sect. 4.4.

In this chapter we focus on two major countermeasures, masking and hiding. In particular, we discuss various differential logic styles that are well suited for FPGAs.

4.3 Countermeasures Based on Masking

4.3.1 Masking Principle

The masking countermeasure is certainly less complex to implement in FPGAs than elsewhere as it is applied at the architectural level only. Masking is performed on internal variables that are transformed into shares of masked variables and the mask itself. Software and hardware implementations both take advantage of this countermeasure, which has been the subject of many studies [2, 8, 19, 34]. The masking technique relies on concealing internal sensitive variables x by a mask m which takes random values. The internal variable x does not exist as a net in the cryptosystem but can be reconstructed by a pair of signals $(m, x_m = x\ \theta\ m)$, where x_m is the masked variable and θ is an operation which can be Boolean or arithmetic. Boolean masking uses the bit wise exclusive-or (xor) operation:

$$x_m = x \oplus m,$$

whereas arithmetic masking typically uses a modulus operation on a finite field:

$$x_m = x + m \pmod{n} \quad \text{or}$$
$$x_m = x * m \pmod{n},$$

where $n = 2^{|x|} = 2^{|m|}$ is equal to the number of values of the sensitive value or of the mask.

Indeed, for a correct masking scheme, the mask (and therefore the masked data) must be uniformly distributed throughout the secure data flow.

Another way to use the mask is the "random pre-charging" method [7]. This consists in temporal stages alternating between the mask m and the internal variable x. As a result, power consumption, which is mainly caused by the Hamming distance of two consecutive values, is not directly correlated with x. The drawback in hardware implementation is a decrease in throughput which is approximately a factor of 2 compared to "spatial" masking.

The implementation of masking is simple when the function f has the following linearity property:

$$f(x\theta m) = f(x)\theta f(m),$$

where θ is still a group operation.

The value of $f(x)$ can be reconstructed from the application of $f(x\theta m)$ and $f(m)^{-1}$, hence the computation of $f(x)$ can be extracted at the very end of the algorithm. This avoids direct leakage of information as $x\theta m$ and m are independent from x (as in Shannon's notion of "one-time pad" [50]).

If f is non-linear, the masking structure becomes more complex as $f(x)$ cannot be reconstructed mathematically from $f(x\theta m)$ and $f(m)$.[1] In symmetrical ciphering algorithms, the non-linear part corresponds to the S-boxes S. A common technique applied in software is to use a specific memory acting as a LUT S_m such that

[1] At least, not in a straightforward manner, *i.e.* without inverting f if it is invertible.

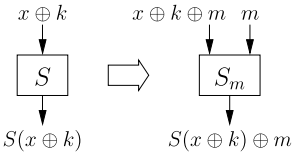


Fig. 4.2 S-box for masking

$S_m(x \oplus m) = S(x) \oplus m$. Consequently the size of this memory to implement the new table increases from 2^n to 2^{2n}, n being the number of bits of the mask.

Figure 4.2 illustrates the complexity change when this masking scheme is used.

It should be noted that it is not secure in hardware, because the register transfers unmask the data. The leakage, in the Hamming distance [53] model (which is implicit in FPGAs), is expressed as:

$$\underbrace{x \oplus m}_{\text{initial value}} \oplus \underbrace{S(x) \oplus m}_{\text{final value}} = x \oplus S(m).$$

For AES, masking can take advantage of the fact that the S-box is a combination of the inverse function in GF(2^8) and an affine function as proposed in [2] and [61]. However this implementation is very sensitive to zero-value attack [18]. This attack can be prevented by using the implementation proposed by Oswald in [39], which takes advantage of the multiplicative masking in GF(4) with only a slight increase in complexity.

4.3.2 Masking Implementations and Vulnerabilities

The robustness of masking countermeasures based on Boolean operators are provable against first order attacks [6], a first order attack being an attack where only the variable x is considered. However masking logic is sensitive to Higher-Order Attacks (HO-DPA) [35], where the attacker uses multiple observations of the same sensitive variable or multiple correlated variables. HO-DPA efficiency is directly related to the knowledge of the leakage, the way to observe the correlated variables [55] and the complexity of the masking implementation [2, 39, 40]. A pitfall introduced by the masked logics is the leakage of sensitive information through glitches. Unless special care is taken, the glitches can indeed depend on unmasked sensitive information, since some gates are likely to combine the masked data with the mask, thereby generating temporarily unprotected transitions [30]. Some special gates [12, 17] or synthesis techniques [37, 38] have been proposed to counter this effect.

In this chapter we will explain the HO-DPA on the classical hardware masking implementation referred to as "zero-offset" [62] and is illustrated in Fig. 4.3.

In the "zero-offset" circuit the second-order DPA can be performed by using observations of both the masked data $x_m = x \oplus m$ and the mask m that are computed concurrently. In order to understand the second-order DPA principle, let us consider the PMF (Probability Mass Function) of the activity corresponding to those of

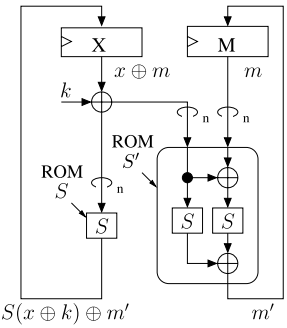


Fig. 4.3 "Zero-offset" masked DES, implemented with ROMs

the combined X and M registers in Fig. 4.3. The activity of these two registers is expressed by:

$$A = HW[\Delta(x, k) \oplus \Delta(m)] + HW[\Delta(m)]. \tag{4.1}$$

Where Δ expresses the distance of a register output, *i.e.*

$$\Delta(x, k) \doteq x \oplus S(x \oplus k),$$
$$\Delta(m) \doteq m \oplus m'.$$

If the registers have four bits (as for DES implementation), there are five possible PMFs depending on the $HW(\Delta(x, k))$ values, when the key is correct, as shown at the top of Fig. 4.4.

When the key is incorrect, the leakage corresponds to that of function A described in Equation 4.1 where:

- x is uniformly distributed in [0x0, 0xf], because the guessed key is wrong,
- m is uniformly distributed in [0x0, 0xf], because the mask is random and unknown to the attacker.

An HO-DPA attack of special interest is the mutual information analysis (MIA) introduced by Gierlichs [15].

It uses the Mutual Information $MI(O; \Delta(x, k) \oplus \Delta(m) + \Delta(m))$ as a distinguisher to build the attack, where optimized attacks close to MIA for example, variance based power attacks (VPA) [29] or entropy based power attacks (EPA) [28] can be devised.

4.3.3 Example of Protection for DES

Let us consider the "zero-offset" implementation of DES studied at UCL [54]. Its iterative architecture is illustrated in Fig. 4.5. This algorithmic masking associates a mask (ML, MR) with the plaintext (L, R).

At each round $i \in [0, 16[$, one intermediate mask (ML_i, MR_i) is calculated in parallel with the intermediate cipher word (L_i, R_i). If we leave aside the expansion

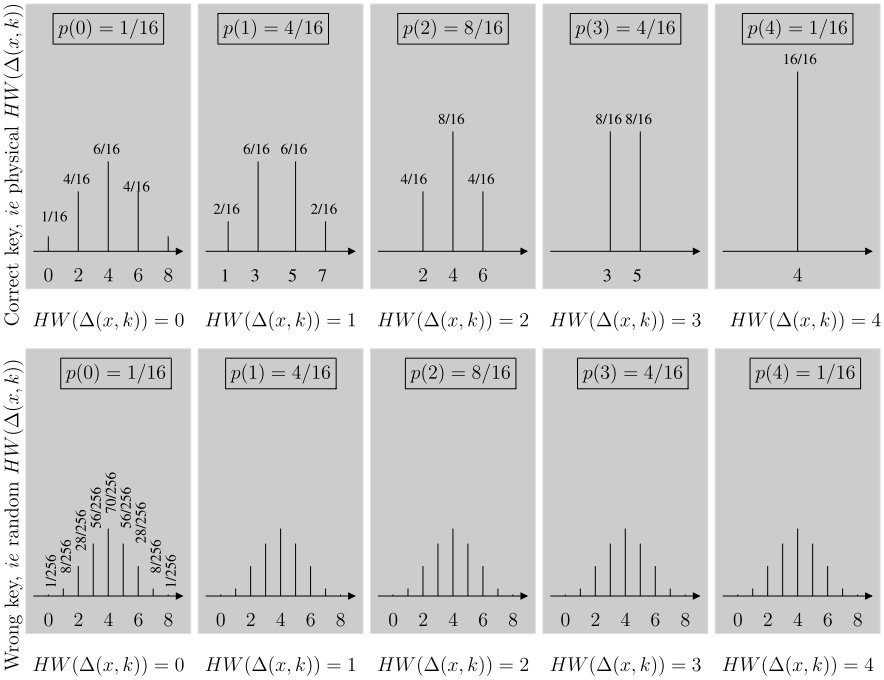


Fig. 4.4 PMFs corresponding to the five possible values of $HW(\Delta(x,k))$

E and the permutation P, the DES [1] round function f is implemented in a masked way by using a set of functions S and a set of functions S':

$$\begin{aligned} S(x_m \oplus k) &= S(x \oplus m \oplus k) = S(x \oplus k) \oplus m', \\ m' &= S'(x_m \oplus k, m) = S'(x \oplus m \oplus k, m). \end{aligned} \tag{4.2}$$

The variable m' is a new mask that can be used again in the next round.

The set of functions S contains the traditional S-boxes applied on masked intermediate words. The size of each S is 64 words of 4 bits when implemented with a ROM. The function S' is a new table which has a much greater ROM size of 4 K words of 4 bits, as there are two input words of 6 bits.

To mitigate the attacks on "zero-offset" implementations one possible solution is to balance the distribution. Authors in [27] propose to "squeeze" the leakage by inserting specific bijections before and after storing the mask. These bijections can be part of the "masked" ROM, see Fig. 4.6. The intermediate data (*e.g.* Sboxes output) have been protected by the same strategy, so as to ensure seamless "squeezing" throughout the combinational logic.

For AES, the implementation of the masked SubBytes structure S' would lead to ROM memory blocks with size 2^{16} words of 8 bits, which represents a huge increase in complexity. It is thus preferable to use substitution boxes with operators computing in smaller fields than GF(2^8) as in [39].

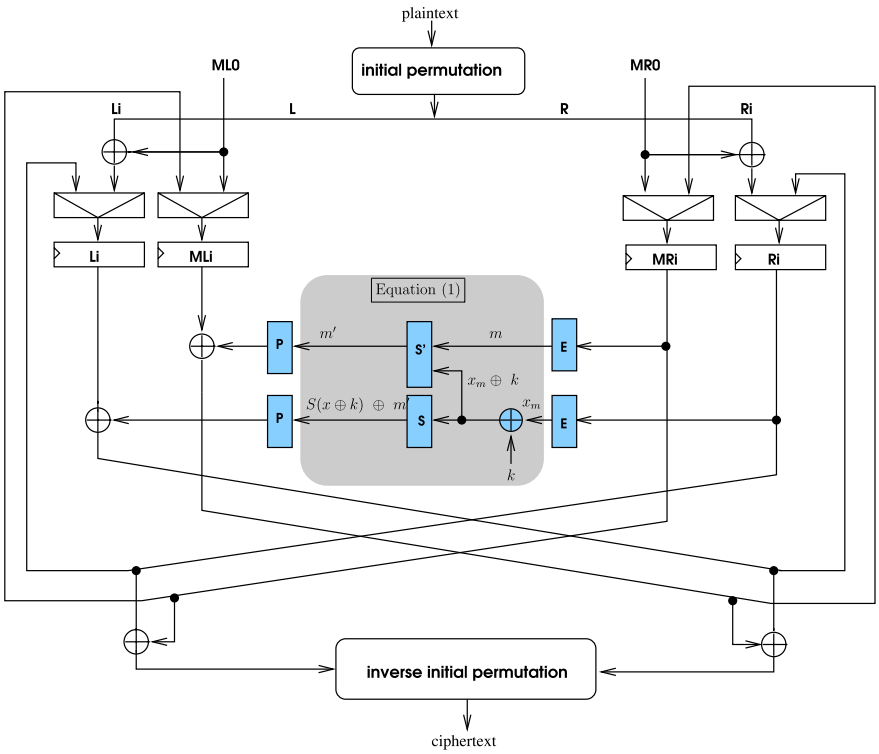


Fig. 4.5 Masked DES datapath

Another possible solution is reduce the implementation cost by regenerating m' from Eq. (4.2):

$$m' = S(x \oplus k \oplus m) \oplus S(x \oplus k). \tag{4.3}$$

This implementation called "universal S-Box masking" (USM) is illustrated in Fig. 4.7. It is such that some non-masked values (namely the S-box input and output) appear clearly in the implementation. This undoubtedly represents a potential threat. The size is merely increased by a factor of 2 due to the fact the S-box are duplicated. This is much less than the increase proposed in [54] where look up tables are 4 K $\times$ 4 memories.

In order to protect the USM implementation, the above mentioned squeezing leakage principle can be applied. The protected USM is shown in Fig. 4.8. It is composed of layers of the encoded table including input and output bijections. The bijections have to be chosen to ensure the maximum possible balance between the distribution of activity for the right key. For instance a robustness evaluation facilitates this choice. The bijections for the expansion and the permutation of DES are linear ("XOR with constant" operation) as these functions split the 32-bit words. However it is important to use non-linear bijections (at least 3 bits) for the S-Box ta-

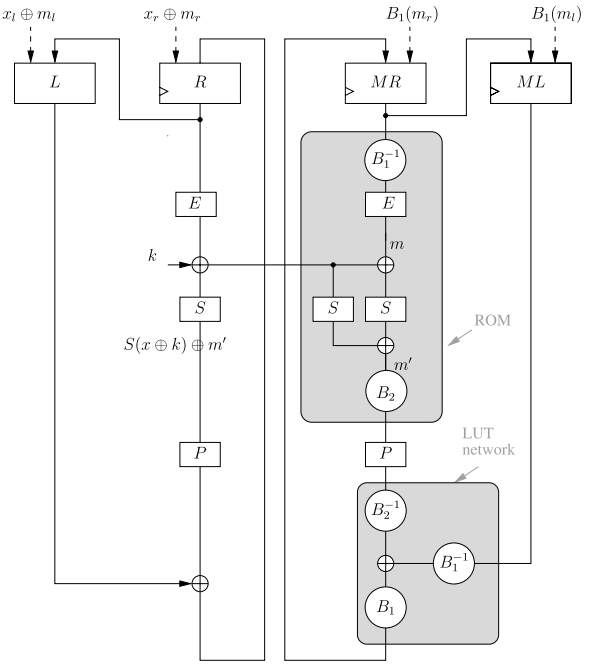


Fig. 4.6 Leakage squeezing of DES with a masked ROM implementation

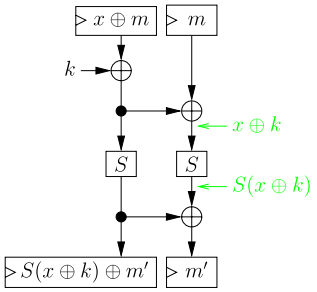


Fig. 4.7 Universal masked f function (of DES), with transiently unmasked values highlighted in *green* (color online)

ble as the Hamming distance reveals the unmasked value if linear bijection are used. The tables are implemented either in LUT networks or FPGA embedded RAMs.

These implementations were tested in a STRATIX II FPGA which is based on an adaptive LUT Module (ALM) cell. They were compared with unprotected DES, masked ROM and masked USM implementations without any leakage squeezing.

Table 4.1 summarizes the memories needed for each implementation and the estimated throughput.

These results show that in hardware implementations, the leakage squeezing method has little impact on complexity and speed. Moreover the USM implementation is particularly efficient as it avoids the use of large ROMs while maintaining high throughput.

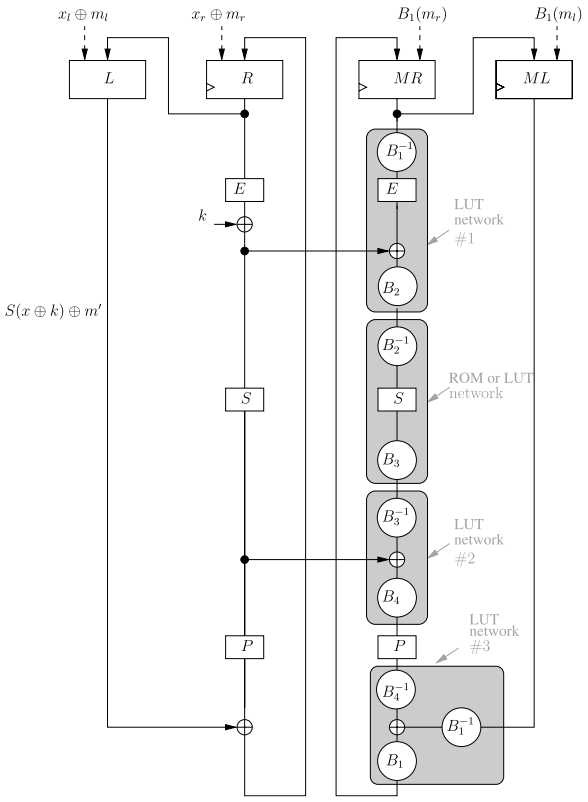


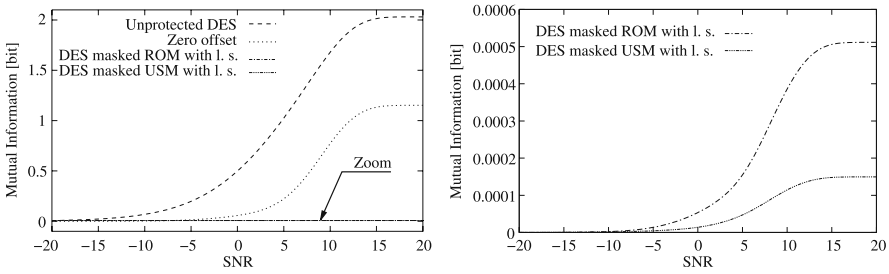
Fig. 4.8 Leakage squeezing of DES with a masked USM implementation

Robustness can be evaluated by using the theoretical framework introduced by Standaert et al. in [52]. The authors suggest analyzing side channel attacks with a combination of information theoretic and security metrics. These metrics aim at evaluating the amount of information provided by a leaking implementation and the possibility to turn this information into a successful key recovery.

Figure 4.9(a) shows the mutual information values obtained for each kind of implementation on simulated traces with respect to an increasing noise standard deviation over $[0.1, 10]$ (*i.e.* an increasing SNR over $[-20, 20]$).

Table 4.1 Complexity and speed results. "l. s." denotes the "leakage squeezing" countermeasure

Implementation	ALMs	Block memory [bit]	M4Ks	Throughput [Mbit/s]
Unprotected DES (*reference*)	276	0	0	929.4
DES masked USM	447	0	0	689.1
DES masked ROM	366	131072	32	398.4
DES masked ROM with l. s.	408	131072	32	320.8
DES masked USM with l. s.	488	0	0	582.8



(a) Comparison (b) Zoom

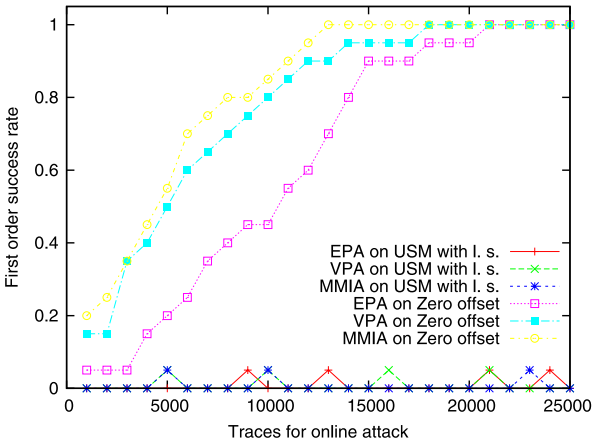
Fig. 4.9 Mutual information metric computed on several DES implementations

These results demonstrate the reduction in information leakage implied by the use of the leakage squeezing technique. As expected, the two implementations based on leakage squeezing leak less information than the zero offset implementation and the unprotected DES for all SNRs. Figure 4.9(b) is a zoom on the evolution of the mutual information in the case of the implementations based on the leakage squeezing technique in order to compare them.

Next real attacks were carried out on real power consumption traces in both the "zero offset" and "USM with squeezed leakage" implementations. For each scenario, a set of 25,000 power consumption traces was acquired using random masks and plaintexts. The first order success rate as described in [52] was calculated for different attack distinguishers; VPA [29], EPA [28] and MMIA [16]. The result is given in Fig. 4.10.

We can see that the attacks based on various distinguishers perform well in the "zero offset" implementation but not in the leakage squeezing implementation.

Fig. 4.10 First order success rate of 3 distinguishers, FPGA implementation



4.3.4 Example of Masked Processor for a Software Implementation

As explained in Chap. 2, experimental results suggest that pipelined processors increase the risk of SCAs, and have to be considered with care. A typical masking implementation for embedded processors in FPGAs requires some considerations.

One solution consists in implementing a dual pipelined datapath. Basically, the idea is to introduce a special datapath for the mask itself, which can be coupled to the classic RISC pipeline. Hence, instead of directly handling raw data, the processor operates on a dual datapath with masked data. The main role of the new datapath is to keep the corresponding mask for each masked data along the pipeline.

By far, the main difficulty is encountered during the EX stage, where all mathematical operations are implemented. As a first hypothesis, the ALU operations are not customized for any mask in order to compute the correct value. Also even if SCAs are still effective on combinational logic, one may consider that the leakage at the register stages is predominant. Indeed, to tackle clock skew issues, buffers are used to drive long lines, involving thus increased power consumption at the register level. This is especially the case for FPGAs. Hence, the ALU operations are performed with unmasked data, whereas the EX pipeline registers are masked.

Moreover, RISC-based architectures are structured around load-store instructions. All potential critical data coming from the data memory use load instructions and could take advantage of a masking scheme when going to the register banks. This approach not only offers the advantage to handle any instruction using a masking scheme but also provides a full compatibility with the processor's instruction set.

A MicroBlaze instruction set compliant processor, the SecretBlaze [3], has been developed for Xilinx FPGA. It implements the ideas of the RISC-based masked datapath. Among different types of masking, the boolean masking was chosen because of its low overhead cost and its good integration into the pipeline. Hence, the masked data result from XOR operations between the raw data and the mask values.

The SecretBlaze provides a dual datapath, two register files, a MAsked Memory Unit (MAMU), and a pseudo random generator (PRNG). The PRNG generates a 32-bit mask value at each clock cycle. The goal of the MAMU is to manage memory accesses with a static mask. Figure 4.11 illustrates the core block diagram of the SecretBlaze. The main differences from the original MicroBlaze are highlighted in gray.

The masking strategy is performed whenever a new value is loaded from the data memory. The loaded data is immediately XORed with the mask generated from the PRNG. The effect of the static mask is afterward removed. Both, the masked data and the mask itself, are stored respectively into the register file and the mask register file during the next clock cycle. As a consequence, no unprotected data is stored into register files of the processor. By analogy, store instructions follow the same scheme in reverse.

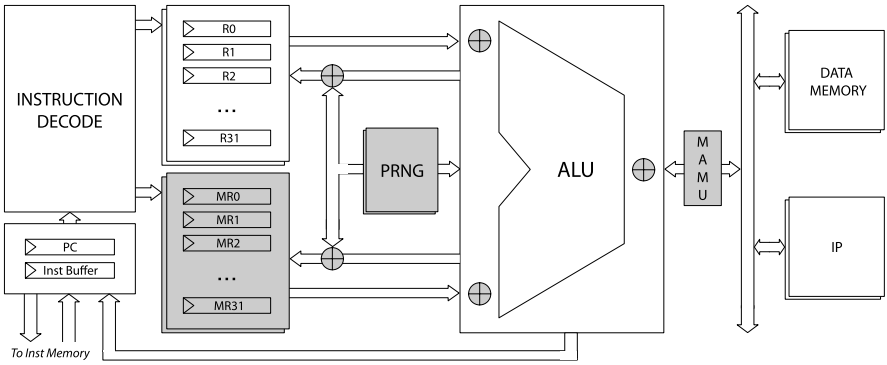


Fig. 4.11 SecretBlaze Block Diagram

For all instructions, the data have to be unmasked for the inputs of the ALU. Although another PRNG would have been the most secured solution, the mask of the first operand is XORed with the value of the PRNG in order to reduce the gate cost of the datapath (the XOR operation between two random numbers is a random number). The same remark can be made for the MA stage. Finally, instructions involving address computations such as loads, stores, as well as branches, have to be unmasked for the address assignments. This poses no significant threat because the addresses do not contain any sensitive information.

The performance and the resource impact of the proposed countermeasure were evaluated with an overhead of 80% of extra flip-flops, owing to the introduction of the PRNG and mask pipeline registers. Then, we observe a slight increase in the usage of slices and LUTs (+30%), related to extra-logic for the datapath of the mask. In terms of performance, the operating frequency is only reduced by 11.2%.

In order to evaluate the robustness of the masking technique, attacks were conducted on the processor running a software DES program, with and without the countermeasure at the hardware level. Results obtained show that the masked SecretBlaze offers a better resistance against DEMA of approximately a factor 2 during the execution of the critical instructions.

Figure 4.12 illustrates the DEMA obtained with 50,000 electromagnetic traces. Unlike the results observed in Chap. 2, this picture shows the efficiency of the protection, since the correlation at the different stages of the pipeline is no more relevant. However, the EX stage of the XOR instruction is still a weak point of the architecture. The conclusion to be drawn from these considerations is that the ALU, more generally combinational logic, is still a critical security issue in FPGAs, even registers are more numerous and more energy-consuming. Further investigations should be conducted to identify alternatives for securing the ALU of the processor, like hiding techniques detailed in the next section.

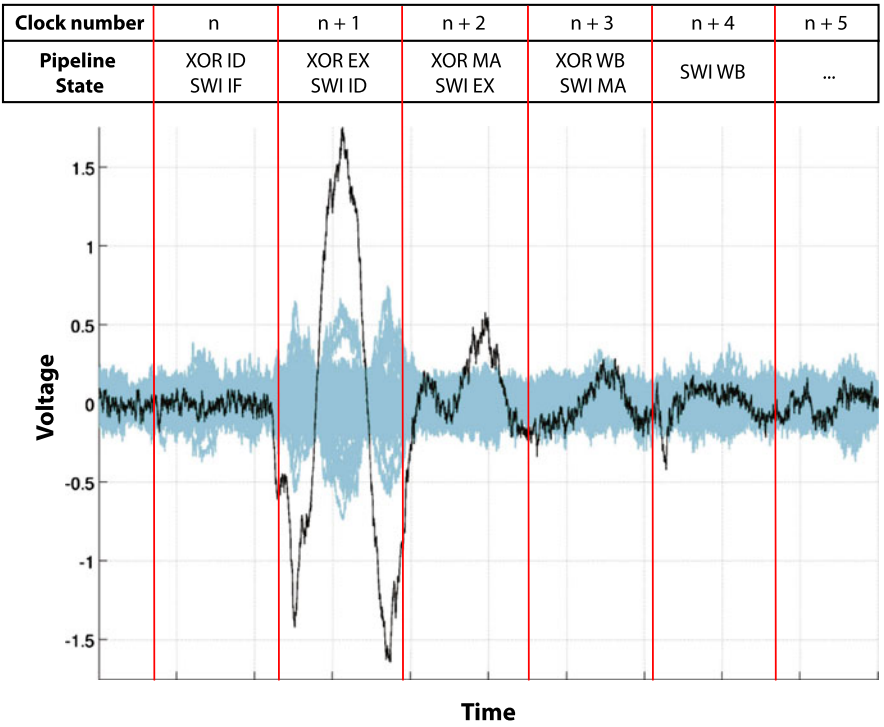


Fig. 4.12 DEMA traces obtained for the first sub-key of the DES software implementation with the SecretBlaze (*blue* = wrong key hypothesis, *black* = good key hypothesis) (color online)

4.4 Countermeasures Based on Hiding

4.4.1 Hiding Technique

The Hiding technique consists in achieving constant power consumption whereas Masking aims at averaging it. One way to obtain constant activity is to use differential logic characterized by the fact each variable is made up of two complementary signals. This logic is such that when one signal switches, the other does not and vice versa. This allows the design to be balanced in terms of activity since in CMOS technology the main power consumer is the switching rate. To make sure the number of transitions ($0 \rightarrow 1$ or $1 \rightarrow 0$) remains constant, the computation has two distinct stages:

1. A **Precharge** stage to reset all the signals in a known state, and
2. An **Evaluation** stage where the computation is performed with a fixed number of transitions.

The differential logic is also called "Dual Rail with Precharge Logic (DPL)" because the two signals from the same variable need twice as many routing resources. Therefore the complexity is at least twice that of an unprotected implementation.

The DPL signalization of the variable a is conveyed by two wires (a_t, a_f) for each Boolean variable, a_t is the TRUE signal and a_f is the FALSE. The state of the variable is either:

- NULL $= (0, 0)$ or $(1, 1)$ while in **Precharge**.
- VALID $\in \{(0, 1), (1, 0)\}$ while in **Evaluation**.

Therefore, every evaluation consists in the transition of exactly one wire ($(0, 0) \rightarrow (0, 1)$ or $(0, 0) \rightarrow (1, 0)$). If the design is properly balanced, which transition actually occurred is indiscernible by an attacker. The computation with DPL logics is a structure of a TRUE and FALSE networks with possible crossing wires interpreted as inversions. Although perfectly sound at logical level, in practice, DPL ends up being implemented in physical devices where the timing parameters impact the balance between the TRUE and FALSE networks. This unbalanced behaviors can damage the level of protection provided by DPL logics. Between the precharge to evaluation, and vice-versa, there may be:

1. Spurious transitions, referred to as glitches, that negate the hypothesis of activity invariability.
2. Early Evaluation (EE) effects. This takes place if the gate switching depends on the difference between the arrival time of the inputs.
3. Technological Bias (TB). This flaw results from the imbalance between the dual signals. It can be caused by manufacturing dispersion, by the place-and-route stage or merely by the types of gate driving the true and false networks. This could be exploited by an attacker who measures the signal from one wire of a pair.

In this section, we focus on the different styles that map very well in FPGAs: WDDL [58], STTL [43], BCDL [11, 36].

4.4.2 WDDL and Its Variants

The Wave Dynamic Differential Logic (WDDL) proposed by Tiri [58] is one of the simplest DPL logic styles.

The NULL state $(0, 0)$ is propagated by a wave of $(0, 0)$ pairs through the netlist thanks to the use of positive gates. A Boolean function f is said to be positive if for two Boolean variables x and y:

$$x \cdot y = x \quad \Rightarrow \quad f(y) \geq f(x).$$

This type of function corresponds to an assembly of AND and OR gates. Figure 4.13 illustrates a two-input "AND2" gate, the logic network TRUE T receives the two TRUE inputs a_t and b_t. The dual "OR" function is implemented by the

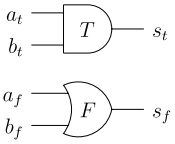


Fig. 4.13 "AND" WDDL gate

F network which receives the FALSE inputs a_f et b_f. Thus $T(x) \doteq H(x)$ and $F(x) \doteq \overline{H(\overline{x})}$, $F(x)$ can be obtained by using the De Morgan's law.

Non-positive logic like inverters or NAND gates are implemented by crossing the two networks. The number of Flip-Flops is necessarily multiplied by four. These are duplicated for the two TRUE and FALSE networks and also for the precharge and evaluation stage as shown in Fig. 4.14.

The timings in Fig. 4.15 show that the number of transitions is the same (two) during the Precharge $\Rightarrow$ Evaluation stage and vice-versa. As in CMOS technologies, power consumption is directly correlated with this number, the logic is reputed to be balanced.

The positivity of WDDL ensures the absence of glitches in the complete netlist. However, as shown in [56, 57], WDDL is prone to early evaluation (EE). The early evaluation effect is due to the difference in timing between two variables of one gate. This timing difference is transferred to the WDDL gate output during the transitions Precharge $\Leftrightarrow$ Evaluation. Figure 4.16(b) illustrates the principle of early evaluation for a 2-input AND gate and its dual 2-input OR gate, as represented in Fig. 4.13. Output switching Δt_1 is different from Δt_2 and therefore can reveal information about the state of the inputs.

In addition, WDDL still has technological bias (TB) *i.e.* imbalance between the dual TRUE and FALSE networks at both structure and routing levels. Constraining

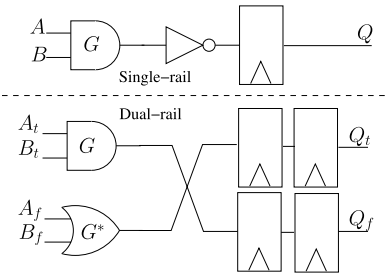


Fig. 4.14 A digital circuit and its WDDL equivalent

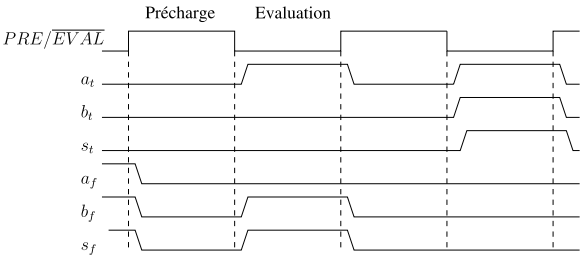


Fig. 4.15 Simulation showing the WDDL timings

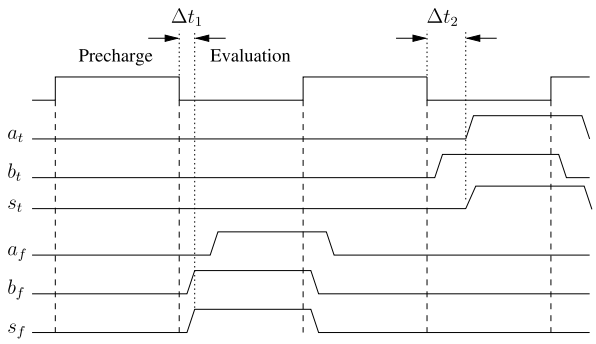


Fig. 4.16 Early evaluation

routing is not so easy in FPGAs because the interconnect structure is confidential. The two causes of imbalance plus the EE effect have enabled some attacks on WDDL circuits, as described by the authors of WDDL themselves in an ASIC [60] or independently in an FPGA [44].

The impact of the place and route (P/R) steps on the timings of dual rail designs is of major importance to obtain dual rail balance and thus reduce the correlation between the processed data and power consumption. Without any P/R constraints it was shown in [44] that WDDL is attackable. Novel P/R strategies that take advantage of FPGAs with cells composed of two-output LUTs have appeared. For instance in [45] the balancing strategy is to place and route the gate in the same STRATIX II ALM.

Some variants of WDDL have been devised to facilitate the balance of the WDDL networks. For instance Double WDDL (DWDDL) was introduced in [63] to counterbalance one unbalanced network with a dummy dual one. The main drawback of DWDDL is its complexity which is double that of WDDL.

Isolated WDDL (IWDDL) [31] is a different strategy to separate a WDDL netlist into two unconnected halves. Here, inverters are kept but a potential glitch is stopped by systematically inserting one register after it. This strategy is expensive in terms of gate complexity and requires a redesign of the controller. Additionally, the design becomes much more pipelined, which requires much higher clock frequencies to maintain an acceptable throughput. However, the advantage of this approach is stopping the propagation of the EE wave.

As DWDDL and IWDDL are complex but theoretically robust, one point is questionable: won't completely separating the netlist open the door to well located EMA attacks that can selectively record the activity of one specific part of the circuit, thus defeating the activity invariability property.

4.4.3 Synchronized Logics: STTL, BCDL

Another strategy to get rid of the early evaluation effect is to synchronize the variables before starting the gate evaluation and precharge stages. However, to make sure no glitches occur, the following conditions have to be met:

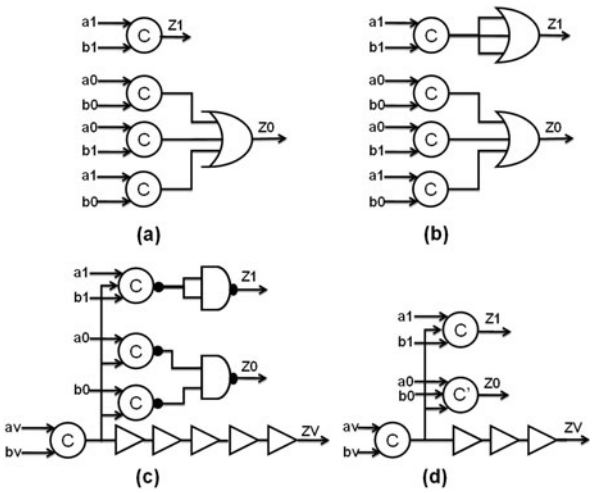


Fig. 4.17 Two-input AND gate implementations: (**a**) basic dual rail AND (**b**) SecLib dual rail AND, (**c**) STTL AND (**d**) compact STTL AND

1. Evaluation starts after all the input signals are valid.
2. Precharge starts by following one of these two rules:
   - Rule `sync-1` : Precharge starts after all the inputs becomes NULL.[2]
   - Rule `sync-2` : Precharge starts before the first input becomes NULL.

If the precharge is always late (rule `sync-1`), the gate outputs need to be memorized. While for rule `sync-2` no memorization is necessary but there is a specific precharge signal.

STTL In [43], the authors suggest using an additional third wire to synchronize the input arrivals by using C-elements to create the Secure Triple Track Logic (STTL), like in Asynchronous logic. In this case, rule `sync-1` applies. This third wire should indicate whether the output data is stable (and thus valid) or not. Figure 4.17 displays different implementations of a dual rail two-input AND gate.

Figure 4.17(a) represents the basic dual rail AND in asynchronous logic. Figure 4.17(b) is a more secure dual rail AND gate, also called SecLib [22] where the two dual outputs are balanced. Figures 4.17(c) and 4.17(d) represent the STTL AND gate, with a more compact triple rail AND in (d). Operator C stand for a C-element [49], ($Z = (a+b)\cdot c + Z\cdot(a+b+c)$), and C' for a generalized C-element. Implementations (b), (c) and (d) are power balanced. However, the third rail in (c) and (d) must fulfill a timing constraint to effectively obtain a quasi data independent timing behavior at block level.

The validity output pin ZV of triple rail gates is controlled by buffers, three in the case of Fig. 4.17(d). These buffers ensure that the delay Θv in propagation from the validity inputs (av, bv) to the output ZV remains greater than the delays Θd from $(a1, a0, b1, b0)$ inputs to the data outputs $(Z0, Z1)$. Note that the number of

[2]NULL is the value in precharge phase.

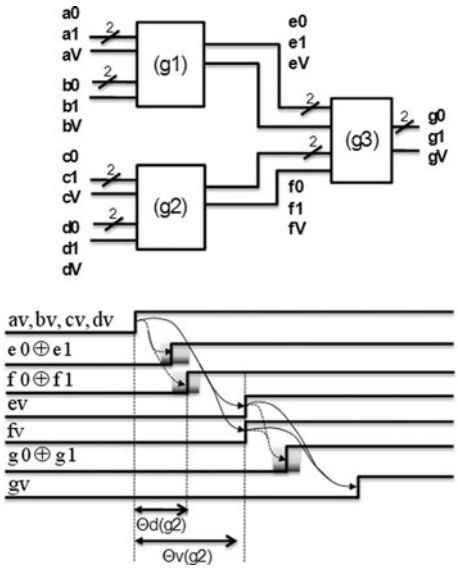


Fig. 4.18 The basic operation of secure triple track logic

buffers must be defined by the designers to guarantee that this timing characteristic is satisfied even in the presence of output load mismatches introduced by the place and route step as described in [43]. With such design guidelines of triple rail gates, we can confidently guarantee that the time at which a triple rail gate fires is independent of the specific data processed by its containing block.

Figure 4.18 illustrates this key characteristic of secure triple rail logic. After the firings of av, bv, cv and dv (assumed to occur at the same time without loss of generality), $e0$, $e1$, $f0$ and $f1$ fire first. Then, the firing of ev and fv occur, which in turn triggers $g0$ or $g1$, followed by gv, since validity rails have a longer propagation delay. Thus the firing of triple rail gates is triggered by the validity rails characterized by a switching speed lower than that of data rails. In other words, the validity rail array (arrows in Fig. 4.18) operates as a backbone of the logical block, sequencing the events independently of data processing (dashed arrows in Fig. 4.18).

During the firing sequence, the time at which $e0$ ($f0$, $g0$), $e1$ ($f1$, $g1$) settle may diff, due to possible output load mismatches. This is represented by the gray rectangles on Fig. 4.18. However, these arrival time mismatches do not affect the firing of the following gates, which are triggered by the validity rails. This characteristic avoids the effect of load mismatches piling up on timing along data paths. This guarantees quasi data independent power consumption and computation time at the block level.

BCDL Balanced Cell-based Differential Logic (BCDL) [11, 36] is a synchronized logic which takes advantage of a global signal that allows the designer to both reduce the complexity of the design and speed up the calculation. This global precharge signal called *PRE* allows rule `sync-2` to be fulfilled, and as a result, no memorization function, such as C-elements, is needed.

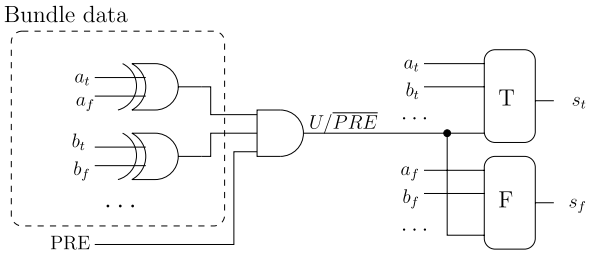


Fig. 4.19 The basic BCDL cell

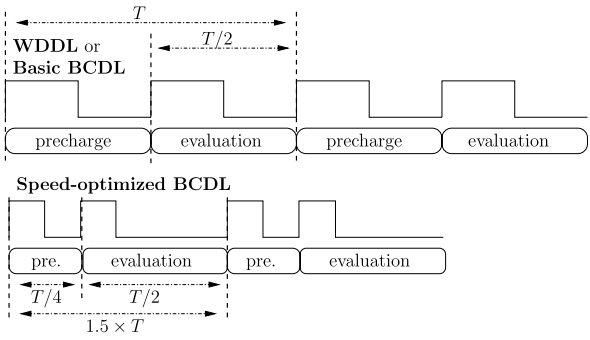


Fig. 4.20 Timing optimization in the DPL protocol when the precharge time is reduced

The basic BCDL gate is presented in Fig. 4.19. Synchronization is performed by a "unanimity to 1" operator on the left side of the figure and can operate on a bundle of data.

The global *PRE* signal is constrained to be faster than any inputs. Consequently when the $U/\overline{PRE}$ of Fig. 4.19 falls to $0 \Rightarrow$ the precharge is forced, and when $U/\overline{PRE}$ rises to $1 \Rightarrow$ the evaluation begins after "unanimity to 1".

As the calculation in Tables T and F can be completely separated, the complexity of the tables is reduced as they receive 2^{n+2} inputs rather than 2^{2n} in others DPL, n being the number of gate inputs. Therefore the implementation with embedded RAM in FPGAs is appropriate. Another complexity gain is the ease of implementation of 2-input gates as they are not limited by the positiveness constraint. For instance, a XOR BCDL gate including synchronization can be done in one ALM of STRATIX II or one LUT6 (dual LUT5) in VIRTEX 5 families.

As the BCDL design allows squeezing of the precharge step, it can compute about 75% faster than WDDL because the precharge is global. This possibility is depicted in Fig. 4.20.

Table 4.2 gives the complexity and speed of an AES implementation in a STRATIX FPGA for WDDL and BCDL.

Table 4.2 Complexity and speed of an AES implementation in WDDL and BCDL

	ALM	Reg	RAM	Max. freq.	Max. throughput
No protection	1078	256	40 Kb	71.88 MHz	287.52 Mbps
WDDL	4885	1024	–	37.07 MHz	74.14 Mbps
BCDL	1841	1024	160 Kb	50.64 MHz	151.92 Mbps

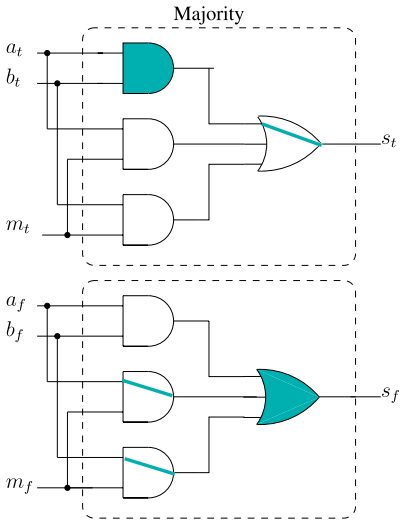


Fig. 4.21 MDPL AND gate

4.4.4 DPL with Masking: MDPL

Masked Dual-rail with Precharge Logic (MDPL [41]) is an attempt to fix the otherwise imbalance of WDDL. The assumption is that, in some conditions, it may be difficult to constrain a router to balance the differential interconnect. Indeed, the two solutions available in the literature, namely the fat wire [59] and the backend duplication [20] methods, apply primarily to ASICs. Transposition to FPGA is possible, although with less fine grained control over the result [21]. For this reason, MDPL swaps the true and the false routes with a random mask, so as to protect from fatal routing unbalance. By the same token, it makes up for the structural unbalance of the dual pair of gates. The only gates involved in the logic are majority functions, both for the true and the false networks. Figure 4.21 represents the MDPL AND gate. When the random mask m is 0 the `TRUE` network performs the AND, whereas the `FALSE` network performs the OR, which is the dual gate of AND, it is the other way round when m is 1.

Although MDPL fails to provide a solution to the early evaluation and precharge of WDDL as presented in [42], it could be efficient against the TB effect. It has been shown that the mask itself could be attacked [46]. Consequently MDPL was enhanced and became the improved MDPL (iMDPL) by adding a synchronization

stage [24]. The Dual Rail switching Logic (DRSL) [10] is very similar to iMDPL and more dedicated to ASIC. However it has been shown in [11] that without P/R care the DRSL can generate glitches when returning to the precharge phase. Synchronized logic like BCDL can take advantage of the random masking to mitigate the TB if few constraints apply during the P/R phase. For instance, BCDL with a mask (MBCDL) would need an extra mask input to the evaluation tables for random swapping of the `TRUE` and `FALSE` networks.

4.4.5 Intrinsic Fault Resilience of Dual-Rail With Precharge Logics

Single bit faults are inefficient against DPL because they turn a VALID data into a NULL token, that propagates and results in a non-exploitable error since it hides the faulted value. This is the typical scenario described in the seminal paper [48], introducing the intrinsic immunity of DPL against some classes of DFA.

Highly multiple faults ($(1,0) \leftrightarrow (0,1)$) randomly generate a large quantity of NULL values along with some more unlikely but devastating bit-flips. However, as NULL values are systematically propagated, they proliferate very quickly after some combinatorial logic layers traversal. And as they have the nice property of being able to contaminate VALID values, the risky coherent bit-flips (simultaneous $0 \xrightarrow{*} 1$ and $1 \xrightarrow{*} 0$ in one dual-rail couple) is very likely to be jammed through the propagation towards the algorithm output. This absorption property is all the more efficient as the number of NULL generated by the multiple faults is high. Therefore, the only way to inject a 'poisonous' fault is to stress the circuit sufficiently to generate multiple faults, without nonetheless creating too many faults so as to leave a chance for them not to be absorbed during their percolation towards the outputs [4, 23].

4.4.6 Comparison of DPL Families

Table 4.3 compares robustness, speed and complexity of a few DPLs. In fact, it is difficult to evaluate robustness fairly, as the DPL countermeasure depends on the target technology and the quality of the P/R stages. However most non-synchronized DPL, such as WDDL and MDPL have been attacked without any particular P/R effort [42, 44, 57]. The analysis of DPL robustness is still an ongoing research project. Some ideas come from the information theory with mutual information analysis [15] and the stochastic approach [47]. In Table 4.3, robustness against SCA is indicated by the logic capacity to be insensitive to early evaluation and technological bias. The fault column indicates if the logic able to cope with symmetric fault (faults being '1' or '0'), which is preferable, rather than asymmetric. Fault detection can be combinatorial or sequential. If it is sequential the cost is higher.

Table 4.3 Comparison of robustness, complexity and speed of a few DPLs

Logic	Compl.	Speed	Robust. SCA		Robust. FA		Design Constr.
			EE	T. B.	Fault	Det.	
WDDL	*	$< 1/2$			asym	comb	Positive gates
MDPL	*	$< 1/2$		✓	asym	comb	MAJ gate + RNG
STTL	*	$< 1/4$	✓		sym	seq	50% more wiring
DRSL	*	$< 1/2$	partly	✓	sym	comb	+ RNG
IWDDL		$< 1/2 \cdot n$	✓		asym	comb	superpipeline
BCDL	**	$> 1/2$	✓		sym	comb	
MBCDL	*	$> 1/2$	✓	✓	sym	comb	+ RNG

Table 4.4 Hardware countermeasures overhead

Countermeasure	None: reference	Masking	Hiding: WDDL	Hiding: BCDL
Period	1	$\approx 1\times$	$\approx 2\times$	$\in [1, 2]\times$
Area: gates	1	$\approx 2\times$	$\approx 2\times$	$\approx 2\times$
Area: memory	$2^n \times m$	$2^{2n} \times m$	$2^{2n} \times 2m$	$2^{n+1} \times 2m$

Two stars in the complexity column means that the DPL needs less gates/ memories to be implemented. The ratio with an unprotected implementation is given in the speed column. If the ratio is greater than $1/2$, this means the DPL has an accelerated precharge stage, like for BCDL. Finally the design constraints are listed, for instance the needs of an RNG.

4.5 Comparison of FPGA, ASIC and Software Countermeasures

As shown in 4.3 and 4.4, FPGAs and ASICs allow for the implementation of masking and hiding techniques that do not affect the processing speed too much. Those targets are indeed much easier to protect than software targets, since the designer has full control over implementation. Instead, on a processor, only some instructions are available, which makes the implementation of countermeasures very awkward.

The overhead of hardware implementations is given in Table 4.4. Regarding hardware countermeasures, it is remarkable that the throughput is almost unchanged. Indeed, the mask can be processed in parallel, the only interaction between the several shares occurring during recombination, merely consists in xor operations. In dual-rail logics, the precharge inevitably causes a dead cycle for each evaluation cycle. And since, in addition, some gates cannot be used for all logics (for instance, positive gates are required in WDDL), the complexity and speed results are worse than without protection. Thus, the throughput is about halved in WDDL. BCDL has the special feature that the precharge cycle can be shrunk, which results in an overhead in terms of throughput that is less than a factor of two. Regarding the number of

resources required to implement the countermeasures, they are about doubled, since two paths are created both for masking and hiding strategies. The biggest difference comes from the RAMs. In masking, unless special structures can be used (such as the factorizing of the S-Box in AES), the number of address bits for the masking is doubled. The same goes for the hiding style, where the number of output bits is also doubled. BCDL is an exception, as the precharge is global. Thus only one additional input bit is required (for the precharge global line) to zero the dual-rail output.

The difference between FPGAs and ASICs is due to the fact that RAMs are available in larger quantities in FPGAs. Thus makes all the countermeasures presented in this section especially attractive. In addition, the FPGAs can take advantage of their reconfigurability capability to mutate their implementation, thereby preventing some attacks that rely on the hypothesis of constant architecture.

Comparing the overhead of Hardware versus Software implementations is more difficult. First of all, no software "hiding" countermeasure has been proposed so far. Regarding masking, it is usually accepted that an overhead of 100 for the throughput is a reasonable approximation. The advantage of hardware is thus clear.

4.6 Conclusions

Cryptographic algorithms can be mapped without difficulty in FPGAs. Although these targets are *a priori* leaking more than ASICs, thanks to their genericity, they also welcome traditional countermeasures, typically those based on masking and hiding. Programming these countermeasures was shown to be feasible, and a number of case-studies have been cited. We conclude that the remarkable property of countermeasures in hardware is that they have almost no affect on the throughput of the algorithm. This contrasts greatly with software countermeasures, which are considerably slowed down when a rigorous masking is enforced. Also, the reconfigurability of FPGAs enables algorithmic countermeasures, such as period implementation update, which impedes attacks that rely on a stable side-channel measurement. This potentiality is rarely addressed in the literature; it is clear that such high-level countermeasures can be further enhanced for greater system-level security.

References

1. NIST/ITL/CSD. Data Encryption Standard. Fips Pub 46-3. http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf (1999)
2. Akkar, M.-L., Giraud, C.: An implementation of DES and AES secure against some attacks. In: Proceedings of CHES'01, Paris, France, May. LNCS, vol. 2162, pp. 309–318. Springer, Berlin (2001)
3. Barthe, L., Benoit, P., Torres, L.: Investigation of a masking countermeasure against side-channel attacks for RISC-based processor architectures. In: FPL, pp. 139–144 (2010)
4. Bhasin, S., Danger, J.-L., Flament, F., Graba, T., Guilley, S., Mathieu, Y., Nassar, M., Sauvage, L., Selmane, N.: Combined SCA and DFA countermeasures integrable in a FPGA design flow. In: ReConFig, Cancún, Quintana Roo, México, December 9–11,

pp. 213–218. IEEE Comput. Soc., Los Alamitos (2009). doi:10.1109/ReConFig.2009.50. http://hal.archives-ouvertes.fr/hal-00411843/en/
5. Bhasin, S., Guilley, S., Sauvage, L., Danger, J.-L.: Unrolling cryptographic circuits: a simple countermeasure against side-channel attacks. In: RSA Cryptographers' Track, CT-RSA, San Francisco, CA, USA, March 1–5. LNCS, vol. 5985, pp. 195–207. Springer, Berlin (2010). doi:10.1007/978-3-642-11925-5_14
6. Blomer, J., Guajardo, J., Krummel, V.: Provably secure masking of AES. In: Proceedings of SAC'04, Waterloo, Canada, August. LNCS, vol. 3357, pp. 69–83. Springer, Berlin (2004)
7. Bucci, M., Guglielmo, M., Luzzi, R., Trifiletti, A.: A power consumption randomization countermeasure for DPA-resistant cryptographic processors. In: PATMOS. LNCS, vol. 3254, pp. 480–490. Springer, Berlin (2004)
8. Chari, S., Jutla, C., Rao, J., Rohatgi, P.: Towards sound approaches to counteract power-analysis attacks. In: CRYPTO, August. LNCS, vol. 1666 (1999). ISBN 3-540-66347-9
9. Chaudhuri, S., Danger, J.-L., Guilley, S., Hoogvorst, P.: FASE: an open run-time reconfigurable FPGA architecture for tamper-resistant and secure embedded systems. In: IEEE 3rd International Conference on Reconfigurable Computing and FPGAs (ReConFig 2006), San Luis Potosi, Mexico, September, pp. 1–9 (2006)
10. Chen, Z., Zhou, Y.: Dual-rail random switching logic: a countermeasure to reduce side channel leakage. In: CHES, Yokohama, Japan. LNCS, vol. 4249, pp. 242–254. Springer, Berlin (2006)
11. Danger, J.-L., Guilley, S., Bhasin, S., Nassar, M.: Overview of dual rail with precharge logic styles to thwart implementation-level attacks on hardware cryptoprocessors.—New attacks and improved counter-measures—. In: SCS, IEEE, Jerba, Tunisia, November 6–8, pp. 1–8 (2009). doi:10.1109/ICSCS.2009.5412599. Complete version online: http://hal.archives-ouvertes.fr/hal-00431261/en/
12. Fischer, W., Gammel, B.M.: Masking at gate level in the presence of glitches. In: CHES, Edinburgh, UK, August 29–September 1. LNCS, vol. 3659, pp. 187–200. Springer, Berlin (2005)
13. Garcia, A.: Power consumption and optimization in field programmable gate arrays. PhD thesis (in French). Ecole Nationale Supérieure des Télécommunications (August 2000)
14. García, A.D., Burleson, W.P., Danger, J.-L.: Power modelling in field programmable gate arrays (FPGA). In: FPL, Glasgow, UK, August 30–September 1. LNCS, vol. 1673, pp. 396–404. Springer, Berlin (1999)
15. Gierlichs, B., Batina, L., Tuyls, P., Preneel, B.: Mutual information analysis. In: CHES, 10th International Workshop, Washington, D.C., USA, August 10–13. LNCS, vol. 5154, pp. 426–442. Springer, Berlin (2008)
16. Gierlichs, B., Batina, L., Preneel, B., Verbauwhede, I.: Revisiting higher-order DPA attacks: multivariate mutual information analysis. In: CT-RSA, San Francisco, CA, USA, March 1–5. LNCS, vol. 5985, pp. 221–234. Springer, Berlin (2010)
17. Golic, J.D., Menicocci, R.: Universal masking on logic gate level. Electron. Lett. **40**(9), 526–528 (2004). doi:10.1049/el:20040385
18. Golic, J., Tymen, C.: Multiplicative masking and power analysis of AES. In: CHES, San Francisco, USA, vol. 2523, pp. 198–212. Springer, Berlin (2003)
19. Goubin, L., Patarin, J.: DES and differential power analysis. In: CHES, August. LNCS, pp. 158–172. Springer, Berlin (1999)
20. Guilley, S., Hoogvorst, P., Mathieu, Y., Pacalet, R.P.: The "backend duplication" method. In: CHES, Edinburgh, Scotland, UK, August 29th–September 1st. LNCS, vol. 3659, pp. 383–397. Springer, Berlin (2005)
21. Guilley, S., Chaudhuri, S., Sauvage, L., Graba, T., Danger, J.-L., Hoogvorst, P., Vong, V.-N., Nassar, M.: Place-and-route impact on the security of DPL designs in FPGAs. In: HOST (Hardware Oriented Security and Trust), IEEE, Anaheim, CA, USA, June, pp. 29–35 (2008)
22. Guilley, S., Chaudhuri, S., Sauvage, L., Hoogvorst, P., Pacalet, R., Bertoni, G.M.: Security evaluation of WDDL and SecLib countermeasures against power attacks. IEEE Trans. Comput. **57**(11), 1482–1497 (2008)

23. Guilley, S., Sauvage, L., Danger, J.-L., Selmane, N.: Fault injection resilience. In: FDTC, Santa Barbara, CA, USA, August 21. IEEE Comput. Soc., Los Alamitos (2010). Complete version: http://hal.archives-ouvertes.fr/hal-00482194/en/
24. Kirschbaum, M., Popp, T.: Evaluation of a DPA-resistant prototype chip. In: ACSAC, Honolulu, Hawaii, July, pp. 43–50. IEEE Comput. Soc., Los Alamitos (2009)
25. Kocher, P.C.: Leak-resistant cryptographic indexed key update, March 25, 2003. United States Patent 6,539,092 filed on July 2nd, 1999 at San Francisco, CA, USA
26. Kuon, I., Rose, J.: Measuring the gap between FPGAs and ASICs. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **26**(2), 203–215 (2007)
27. Maghrebi, H., Guilley, S., Danger, J.-L.: Leakage squeezing countermeasure against high-order attacks. In: WISTP, Heraklion, June (2011)
28. Maghrebi, H., Guilley, S., Danger, J.-L., Flament, F.: Entropy-based power attack. In: HOST, Anaheim Convention Center, Anaheim, CA, USA, pp. 1–6. IEEE Comput. Soc., Los Alamitos (2010). doi:10.1109/HST.2010.5513124
29. Maghrebi, H., Danger, J.-L., Flament, F., Guilley, S.: Evaluation of countermeasures implementation based on boolean masking to thwart first and second order side-channel attacks. In: SCS, IEEE, Jerba, Tunisia, pp. 1–6 (2009). doi:10.1109/ICSCS.2009.5412597. Complete version online: http://hal.archives-ouvertes.fr/hal-00425523/en/
30. Mangard, S., Popp, T., Gammel, B.M.: Side-channel leakage of masked CMOS gates. In: CT-RSA, San Francisco, CA, USA. LNCS, vol. 3376, pp. 351–365. Springer, Berlin (2005)
31. McEvoy, R.P., Murphy, C.C., Marnane, W.P., Tunstall, M.: Isolated WDDL: a hiding countermeasure for differential power analysis on FPGAs. ACM Trans. Reconfigurable Technol. Syst. **2**(1), 1–23 (2009). doi:10.1145/1502781.1502784
32. Medwed, M., Standaert, F.-X., Groéschédl, J., Regazzoni, F.: Fresh re-keying: security against side-channel and fault attacks for low-cost devices. In: AFRICACRYPT, Stellenbosch, South Africa, May 03–06. LNCS, vol. 6055, pp. 279–296. Springer, Berlin (2010). doi:10.1007/978-3-642-12678-9_17
33. Mentens, N., Gierlichs, B., Verbauwhede, I.: Power and fault analysis resistance in hardware through dynamic reconfiguration. In: CHES, Washington, D.C., USA, August 10–13. LNCS, vol. 5154, pp. 346–362. Springer, Berlin (2008)
34. Messerges, T.S.: Securing the AES finalists against power analysis attacks. In: Fast Software Encryption'00, New York, April 2000 (2000)
35. Messerges, T.S.: Using second-order power analysis to attack DPA resistant software. In: CHES, Worcester, MA, USA, August 17–18. LNCS, vol. 1965, pp. 71–77. Springer, Berlin (2000)
36. 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, Dresden, Germany, March 8–12, pp. 849–854. IEEE Comput. Soc., Los Alamitos (2010)
37. Nikova, S., Rechberger, C., Rijmen, V.: Threshold implementations against side-channel attacks and glitches. In: ICICS, Raleigh, NC, USA, December 4–7. LNCS, vol. 4307, pp. 529–545. Springer, Berlin (2006)
38. Nikova, S., Rijmen, V., Schläffer, M.: Secure hardware implementation of non-linear functions in the presence of glitches. In: ICISC, Seoul, Korea. LNCS, vol. 5461, pp. 218–234. Springer, Berlin (2008)
39. Oswald, E., Mangard, S., Pramstaller, N., Rijmen, V.: A side-channel analysis resistant description of the AES S-box. In: Proceedings of FSE'05, Paris, France, February. LNCS, vol. 3557, pp. 413–423. Springer, Berlin (2005)
40. Peeters, E., Standaert, F.X., Donckers, N., Quisquater, J.J.: Improved higher-order side-channel attacks with FPGA experiments. In: CHES. LNCS, vol. 3659, pp. 309–323. Springer, Berlin (2005)
41. Popp, T., Mangard, S.: Masked dual-rail pre-charge logic: DPA-resistance without routing constraints. In: Proceedings of CHES'05, Edinburgh, Scotland, UK, September. LNCS, vol. 3659, pp. 172–186. Springer, Berlin (2005)

42. Popp, T., Kirschbaum, M., Zefferer, T., Mangard, S.: Evaluation of the masked logic style MDPL on a prototype chip. In: CHES, Vienna, Austria, September. LNCS, vol. 4727, pp. 81–94. Springer, Berlin (2007)
43. Razafindraibe, A., Robert, M., Maurine, P.: Analysis and improvement of dual rail logic as a countermeasure against DPA. In: PATMOS, Göteborg, Sweden, pp. 340–351 (2007)
44. 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, April, pp. 640–645. IEEE Comput. Soc., Los Alamitos (2009)
45. Sauvage, L., Nassar, M., Guilley, S., Flament, F., Danger, J.-L., Mathieu, Y.: Exploiting dual-output programmable blocks to balance secure dual-rail logics. Int. J. Reconfigurable Comput. **2010**, 375245 (2010). 12 pages. doi:10.1155/2010/375245
46. Schaumont, P., Tiri, K.: Masking and dual rail logic don't add up. In: CHES, Vienna, Austria, September 10–13. LNCS, vol. 4727, pp. 95–106. Springer, Berlin (2007)
47. Schindler, W., Lemke, K., Paar, C.: A stochastic model for differential side channel cryptanalysis. In: CHES, Edinburgh, Scotland, UK, September. LNCS, vol. 3659, pp. 30–46. Springer, Berlin (2005)
48. Selmane, N., Bhasin, S., Guilley, S., Graba, T., Danger, J.-L.: WDDL is protected against setup time violation attacks. In: FDTC, Lausanne, Switzerland, September 6th, pp. 73–83. IEEE Comput. Soc., Los Alamitos (2009). doi:10.1109/FDTC.2009.40. In conjunction with CHES'09. Online version: http://hal.archives-ouvertes.fr/hal-00410135/en/
49. Shams, M., Ebergen, J.C., Elmasry, M.I.: Modeling and comparing CMOS implementations of the C-element. IEEE Trans. VLSI Syst. **6**(4), 563–567 (1998)
50. Shannon, C.E.: Communication theory of secrecy system. Bell Syst. Tech. J. **28**, 656–715 (1949)
51. Standaert, F.-X.: Secure and efficient implementation of symmetric encryption schemes using FPGAs, pp. 295–320 (2009)
52. Standaert, F.-X., Malkin, T., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In: EUROCRYPT, Cologne, Germany, April 26–30. LNCS, vol. 5479, pp. 443–461. Springer, Berlin (2009)
53. Standaert, F.-X., Örs, S.B., Preneel, B.: Power analysis of an FPGA: implementation of Rijndael: is pipelining a DPA countermeasure? In: CHES, Cambridge (Boston), MA, USA, August 11–13. LNCS, vol. 3156, pp. 30–44. Springer, Berlin (2004)
54. Standaert, F.-X., Rouvroy, G., Quisquater, J.-J.: FPGA implementations of the DES and triple-DES masked against power analysis attacks. In: Proceedings of FPL 2006, Madrid, Spain, August (2006)
55. Standaert, F.-X., Veyrat-Charvillon, N., Oswald, E., Gierlichs, B., Medwed, M., Kasper, M., Mangard, S.: The world is not enough: another look on second-order DPA. Cryptology ePrint Archive, Report 2010/180. http://eprint.iacr.org/ (2010)
56. Suzuki, D., Saeki, M.: Security evaluation of DPA countermeasures using dual-rail pre-charge logic style. In: CHES. LNCS, vol. 4249, pp. 255–269. Springer, Berlin (2006)
57. Suzuki, D., Saeki, M.: An analysis of leakage factors for dual-rail pre-charge logic style. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **E91-A**(1), 184–192 (2008). doi:10.1093/ietfec/e91-a.1.184
58. Tiri, K., Verbauwhede, I.: A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation. In: DATE'04, Paris, France, February, pp. 246–251 (2004)
59. Tiri, K., Verbauwhede, I.: Place and route for secure standard cell design. In: Proceedings of WCC/CARDIS, Toulouse, France, August, pp. 143–158 (2004)
60. Tiri, K., Hwang, D., Hodjat, A., Lai, B.-C., Yang, S., Schaumont, P., Verbauwhede, I.: Prototype IC with WDDL and differential routing—DPA resistance assessment. In: Proceedings of CHES'05, Edinburgh, Scotland, UK, August 29–September 1. LNCS, vol. 3659, pp. 354–365. Springer, Berlin (2005)
61. Trichina, E., Seta, D.D., Germani, L.: Simplified adaptive multiplicative masking for AES. In: CHES, pp. 187–197 (2002)

62. Waddle, J., Wagner, D.: Towards efficient second-order power analysis. In: CHES. LNCS, vol. 3156, pp. 1–15. Springer, Berlin (2004)
63. Yu, P., Schaumont, P.: Secure FPGA circuits using controlled placement and routing. In: CODES+ISSS'07: Proceedings of the 5th IEEE/ACM International Conference on Hardware/Software Codesign and System Synthesis, pp. 45–50. ACM, New York (2007). doi:10.1145/1289816.1289831