Early Propagation and Imbalanced Routing, How to Diminish in FPGAs

Amir Moradi[1] and Vincent Immler[2⋆]

[1] Horst Görtz Institute for IT Security, Ruhr University Bochum, Germany
[2] Fraunhofer Research Institution for Applied and Integrated Security (AISEC), Munich, Germany
amir.moradi@rub.de, vincent.immler@aisec.fraunhofer.de

Abstract. This work deals with DPA-resistant logic styles, i.e., cell-level countermeasures against power analysis attacks that are known as a serious threat to cryptographic devices. Early propagation and imbalanced routings are amongst the well-known issues of such countermeasures, that – if not considered during the design process – can cause the underlying cryptographic device to be vulnerable to certain attacks. Although most of the DPA-resistant logic styles target an ASIC design process, there are a few attempts to apply them in an FPGA platform. This is due to the missing freedom in FPGA design tools required to deal with the aforementioned problems. Our contribution in this work is to provide solutions for both early propagation and imbalanced routings considering a modern Xilinx FPGA as the target platform. Foremost, based on the WDDL concept we design a new FPGA-based logic style without early propagation in both precharge and evaluation phases. Additionally, with respect to the limited routing resources within an FPGA we develop a customized router to find the best appropriate dual-rail routes for a given dual-rail circuit. Based on practical experiments on a Virtex-5 FPGA our evaluations verify the efficiency of each of our proposed approaches. They significantly improve the resistance of the design compared to cases not benefiting from our schemes.

1 Introduction

Counteracting state-of-the-art power analysis attacks (so called DPA [13]) is a must for cryptographic devices which may fall into the hands of malicious users, who can control over the device in a hostile environment. Up to now several DPA countermeasures at different levels of abstraction have been proposed. Many try to provide resistance by manipulating the underlying cryptographic algorithm in order to randomize its intermediate values, i.e., masking at the algorithmic level, e.g., [24, 25]. Some introduce noise, e.g., [7, 17] or randomize either the program flow or the order of the operations, i.e., shuffling, e.g., [10, 17]. A couple

⋆ The majority of the work was performed while Vincent Immler was with Ruhr University Bochum.

of other schemes try to solve the problem from scratch, i.e., avoiding the dependency of the power consumption of the circuit to the processed data. These countermeasures at the cell level, called DPA-resistant logic styles, aim at equalizing the power consumption of a cryptographic device regardless of any input, intermediate, or output value.

After a proper evaluation [30] it was discovered that most of the proposed logic styles, such as WDDL [32] and MDPL [27], suffer from the early propagation effect. This phenomena, also called data-dependent time-of-evaluation, refers to the cases where a gate fires (evaluates) its output at different time instances depending on the value of its input. It becomes more problematic when several of such gates are cascaded to realize a combinatorial circuit. So, it causes the power consumption pattern of the circuit to have a clear dependency to its input value.

Moreover, most of the known logic styles face a common difficulty, i.e., routing imbalances. Equal power consumption, which is expected to be achieved by Dual-Rail Precharge (DRP) logic, needs a proper balance between the capacitances of each dual-rail signal. Otherwise, transitions of `TRUE` and `FALSE` lines of a dual-rail signal require different amounts of energy, which can be explored by a DPA attack. Therefore, some place-and-route methods such as [6, 33] have been proposed to diminish the load imbalances of complementary signals in an ASIC design process. Although iMDPL [26], which solves the early propagation effect of MDPL, was designed to relax the necessity of balanced routings, still has exploitable leakages due to imbalanced routing of the dual-rail mask signal [20].

State-of-the-Art Even though most of the proposed logic styles target an ASIC platform, at the early stage of their development some have been evaluated using FPGAs. Since the FPGA design tools miss the flexibility required for balanced routing, most of the efforts in this direction led to *duplication* schemes. They follow the idea of dual-rail concept without precharging the signals. This indeed leads to making a dual copy of a fully placed-and-routed circuit which – in theory – should consume the complement amount of energy that the original counterpart does. However, the problem arising by this scheme is due to non-dual glitches happening in original and dual part of the circuit, that causes the design to be vulnerable to the state-of-the-art attacks.

In this direction we can refer to [11], where – in addition to the dual of the circuit – precharge registers are inserted to the design. As the authors also showed, their design can be broken because of glitches. In another work [8] a specific configuration for FPGA Look-Up-Tables (LUT) is used to make the delay of the gates constant. Two global signals connected to all LUTs are also used to handle the precharge and evaluation of the gates. After developing the circuit by this configuration, the dual part of the circuit is inserted to the design. Unfortunately their design still has glitches when the combinatorial circuit has two or more logic depth.

As another example we should refer to DWDDL [36] which applies the duplication on a circuit realized by a kind of WDDL. There exists other works which make use of the duplication on the circuits built by FPGA Block RAMs (BRAM). For example, the authors of [34] introduced a precharge signal for each BRAM address in order to provide precharge and evaluation phases in this context. By certain inappropriate assumptions, e.g., ignoring the leakage associated to glitches occurring at the LUTs' output as long as they do not leave the slice, they developed a design methodology.

In the work of [28] the authors tried to realize WDDL on an Altera Stratix-II FPGA. They used DES as the target algorithm and have examined two different place-and-route (PAR) strategies as 1) the `TRUE` and `FALSE` signals of each gate are placed and routed as close as possible, and 2) all `TRUE` signals are placed close together (the same for all `FALSE` signals). The drawbacks of their work are 1) no attempt to avoid early propagation, and 2) no control over the delay between the rails of dual-rail signals.

In another work [15] a triple-rail logic has been designed for a Xilinx Spartan-3 FPGA. In order to avoid early propagation in both precharge and evaluation phases they utilized a generalized form of Muller C-element (the main element of asynchronous logic designs [22]). Although the goal of preventing early propagation is fulfilled, the number of toggles happening at internal signals is not balanced, i.e., they are different depending on the gate inputs' value. It is because 6 LUTs are used to build a triple-rail gate, and the toggles of output of these 6 LUTs are not balanced for all input cases. Therefore, it most likely leads to different power consumption patterns detectable by a DPA attack.

Other works e.g., BCDL [23] employed a global precharge signal, which must be connected to all gates. The gates do not evaluate their output till the global precharge goes e.g., LOW thereby preventing early propagation in both phases. But at the start of the precharge phase all gates simultaneously precharge their output leading to a much higher power consumption peak compared to the evaluation phase. Based on an Altera Stratix-II FPGA each BCDL gate is realized by two 5-input LUTs, but they have not taken care about the routing of dual-rail signals.

In a follow-up work [3] the global precharge of BCDL is removed and following the WDDL concept each gate is actualized by two 4-input LUTs. The style, which is called DPL_noEE, prevents the early propagation in the evaluation phase, but nothing is considered to deal with the start of the precharge phase. Similar to the case of BCDL, a Stratix-II FPGA has been used for practical evaluation of an AES encryption module under DPL_noEE scheme. According to the claims of the authors, the leakage could be reduced to half while no restrictions have been put into the placement and routing processes.

Our Contribution In this work we first re-iterate the definition of *early propagation* and address the cases in the literature where this concept in the *precharge phase* has been mixed with the concept of *data-dependent time-of-precharge*. As an example we focus on DPL_noEE [3] and demonstrate that preventing early propagation in the evaluation phase might be not enough to reduce the associated leakage.

In the second part of this article we aim at designing a variant of WDDL for FPGA platforms without early propagation in both phases. With the help of the asynchronous design concept we achieve an architecture which follows the WDDL definitions, but its time-of-evaluation as well as its time-of-precharge is independent of the processed values. More importantly it propagates the evaluation wave (resp. the precharge wave) only when all inputs are evaluated (resp. precharged).

However, the imbalanced routings caused by uncontrolled FPGA design tools (placer and router) makes the power consumption of the circuit to be still related to its input value. Therefore, the next contribution of this article deals with balanced dual-rail routing in FPGAs. By means of a sophisticated routing algorithm as well as information we extract about the route delays, we are able to route the dual-rail signals with minimum imbalances. As a target design and platform we selected an AES S-box to be realized on a Xilinx Virtex-5 FPGA. Our experimental results show that both (more) balanced routings and avoiding early propagation significantly reduce the amount of leakage extractable by a power analysis attack.

2 WDDL and Early Propagation

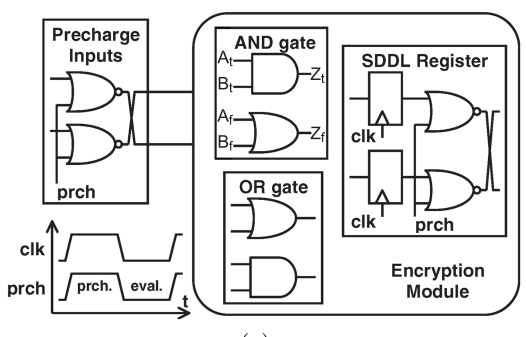
Wave Dynamic Differential Logic (WDDL) was developed to avoid the necessity of a full-custom design tool. Similar to other precharge logic styles every gate operates in two phases, i.e., precharge and evaluation, which are controlled by the clock signal. However, as shown by Fig. 1(a) the signals are converted to dual-rail precharge form prior to the WDDL circuit, and the clock signal is not routed to the WDDL logic cells. Figure 1(a) shows the concept and the design of a WDDL AND/NAND gate. These gates can straightforwardly be actualized by FPGA LUTs, but the concept which is followed by DPL_noEE [3] is shown in Fig. 1(b). The idea is to prevent the evaluation as long as the inputs are not complete[3].

In order to deal with early propagation issue we first assume that though the delay of different dual rails are not the same, the delay of `TRUE` and `FALSE` signals of each dual rail are the same. In other words,

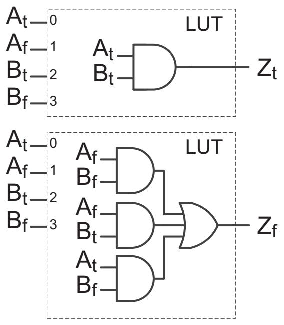
$$\mathrm{Delay}(\mathsf{A_t}) = \mathrm{Delay}(\mathsf{A_f}) \;>\; \mathrm{Delay}(\mathsf{B_t}) = \mathrm{Delay}(\mathsf{B_f}).$$

Figure 2 depicts the timing diagram of three different cases of the inputs for both WDDL and DPL_noEE AND/NAND gates. By comparing cases 1 and 3 (Δte_1 vs. Δte_3) we can conclude that the start of the evaluation phase of the WDDL gate depends on its input values. This issue, which is addressed in [30], is known as data-dependent time-of-evaluation. Examining cases 1 and 2 (Δtp_1 vs. Δtp_2) also shows the dependency of the start of the precharge phase to the gate input, which is referred as data-dependent time-of-precharge.

[3] Note that here we showed a simplified view of DPL_noEE, the authors of [3] considered both '0' and '1' as the precharge value of the gates in their design.



(a)



(b)

Fig. 1. (a) WDDL concept (taken from [32]), (b) DPL_noEE AND/NAND gate

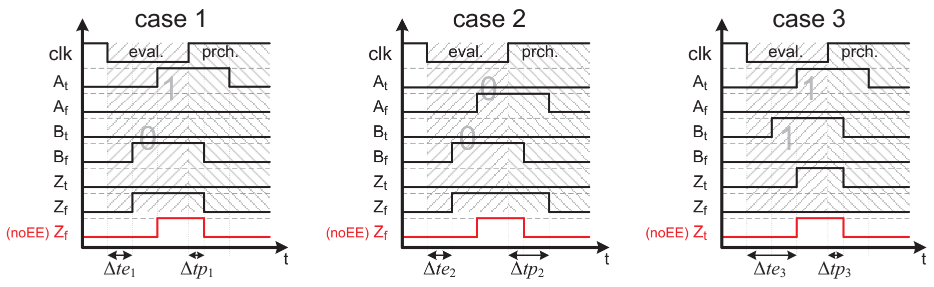


Fig. 2. Timing diagram of three input cases, WDDL and DPL_noEE AND/NAND gates

The situation for DPL_noEE is different as its time-of-evaluation is data independent. It also instantly goes to the precharge phase once one of its inputs goes to precharge. Although its time-of-precharge does not depend on its input value, it fits to the definition of *early propagation in precharge phase*. We should refer to [26], where it is stated as “According to our analysis, DRSL does not completely avoid an early propagation effect in the precharge phase”. From this perspective DPL_noEE and DRSL [5] have the same specification. Both have data-independent time-of-evaluation and time-of-precharge, and both do not avoid early propagation in the precharge phase. On the contrary, the precharge phase of each iMDPL [26] gate starts when all its inputs are in precharge. Here the question is how critical it is to not avoid early propagation in precharge phase while it is data independent?

To answer this question we should highlight two points:

– In this setting – considering a combinatorial circuit, e.g., an AES S-box, made by several gates – the propagation of the precharge wave is faster than that of the evaluation wave. This results in a difference between the power consumption patterns of the evaluation and precharge phases as the dynamic

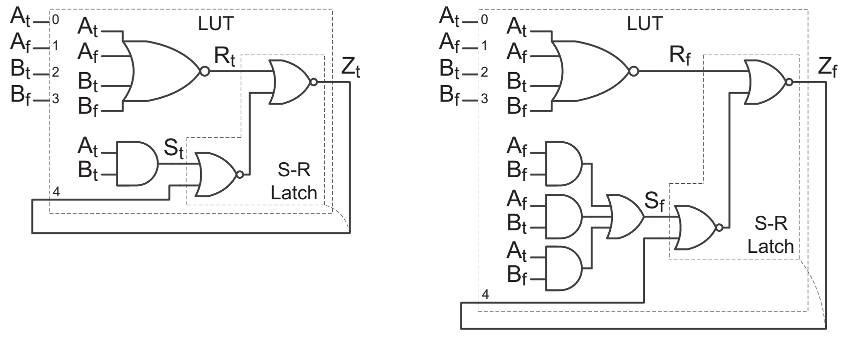


Fig. 3. AWDDL AND/NAND gate[4]

power consumption peak in the precharge phase is expected to be higher. The same is observed when a global precharge signal is used as in SABL [31].

– All the statements given above about the data independence of time-of-evaluation as well as of time-of-precharge are valid as long as the first assumption Delay($\mathsf{A_t}$) = Delay($\mathsf{A_f}$) (resp. Delay($\mathsf{B_t}$) = Delay($\mathsf{B_f}$)) holds. If the routings are imbalanced and this assumption is not valid, both time-of-evaluation and time-of-precharge would be data dependent.

Therefore, in this setting we expect that the leakage in the precharge phase is more easily detectable compared to that in the evaluation phase since slight routing imbalances still would exist in practice. We deal with this issue in Section 4 when we demonstrate the corresponding practical results.

2.1 Avoiding Early Propagation in both Phases

Our goal here is to develop a design similar to DPL_noEE but without early propagation in the precharge phase. So, we consider an asynchronous design for each WDDL gate. Following standard asynchronous design schemes one can make the flow table for the desired gate behavior, and realize it using e.g., S-R approach, where S-R latches are utilized. Figure 3 shows an exemplary design of our desired WDDL AND gate, which we call Asynchronous WDDL (AWDDL). Note that this design is only suitable for the FPGA platforms to realize the gate outputs by LUTs. It cannot be considered as an ASIC solution due to the unbalanced number of toggles happening on the internal signals. Since we use a modern Xilinx FPGA as the target platform, every gate is realized by two hard-coded LUT6 instances, i.e., to provide $\mathsf{Z_t}$ and $\mathsf{Z_f}$. Furthermore, every LUT output should be routed to its input, i.e., an external loop around every LUT

[4] The mapping of the input and output signals to the LUT6 instances is of highly importance. Having the *Xilinx Libraries Guide for HDL Designs* [35] in mind, input I5 of the LUT6_2 should be connected to '1' thereby utilizing only O6 as the gate output. The LUT must be configured in a way that O5 always provides '0'.

is essential. Since the 6-input LUTs are the currently biggest LUT available in FPGA architectures, unfortunately our proposed scheme cannot be extended to consider one more dual-rail mask input signal to realize a kind of iMDPL cell [26].

With respect to the asynchronous design concept and the S-R latches our design of AWDDL guarantees no early propagation in both precharge and evaluation phases. Due to the lack of space we omit presenting the figures for other AWDDL gates. However, the formulas for the set and reset signals (S,R) of the conceptual S-R latch of 2-input gates are listed below. Similar to WDDL, the inverted gate is realized by swapping the dual-rail output signals. Below, + stands for the logical OR operation.

$$\begin{aligned}
\text{all gates}: &\quad \mathsf{R_t} = \mathsf{R_f} = \overline{\mathsf{A_t} + \mathsf{A_f} + \mathsf{B_t} + \mathsf{B_f}} \\
\text{AND}: &\begin{cases} \mathsf{S_t} = \mathsf{A_t}\,\mathsf{B_t} \\ \mathsf{S_f} = \mathsf{A_f}\,\mathsf{B_f} + \mathsf{A_f}\,\mathsf{B_t} + \mathsf{A_t}\,\mathsf{B_f} \end{cases} \\
\text{OR}: &\begin{cases} \mathsf{S_t} = \mathsf{A_t}\,\mathsf{B_t} + \mathsf{A_t}\,\mathsf{B_f} + \mathsf{A_f}\,\mathsf{B_t} \\ \mathsf{S_f} = \mathsf{A_f}\,\mathsf{B_f} \end{cases} \\
\text{XOR}: &\begin{cases} \mathsf{S_t} = \mathsf{A_t}\,\mathsf{B_f} + \mathsf{A_f}\,\mathsf{B_t} \\ \mathsf{S_f} = \mathsf{A_t}\,\mathsf{B_t} + \mathsf{A_f}\,\mathsf{B_f} \end{cases}
\end{aligned}$$

We should emphasize that we are aware of the problem of single-rail WDDL register cells mentioned in [19]. Our contribution focuses on the combinatorial part of the logic style, and as stated in [19], the master-slave register cells must be used to prevent the leakage of the registers.

3 Dual-Rail Routing

The problem of dual-rail routing is a challenge to perfectly balance the capacitance of both rails of a signal. Otherwise, transitions on these signals need a different amount of energy, which may make it possible to distinguish on which rail the transition happens. On the other hand, the capacitive imbalance causes the delay of the rails to be different. Then, the arrival time of a dual-rail input signal of a gate depends on its value. Since a gate without early propagation in both phases, e.g., our AWDDL, fires (resp. precharge) the output when all input signals arrived (resp. precharged), the time of evaluation (resp. precharge) of the gate will still depend on its inputs' value. This propagates through the whole combinatorial circuit, and makes the power consumption patterns different depending on the circuit input value.

Before we focus on our dual-rail routing approach, we should emphasize that, as stated in [32], a WDDL circuit – without an inversion – can be implemented using a divided approach, where first a network of `TRUE` signals, i.e., $\mathsf{Z_t}$ output of all gates, are placed and routed. Then, the dual of the same network with the same routing is copied to make the `FALSE` signals. However, this approach is not applicable in case of our proposed AWDDL since each `TRUE` and `FALSE` part of

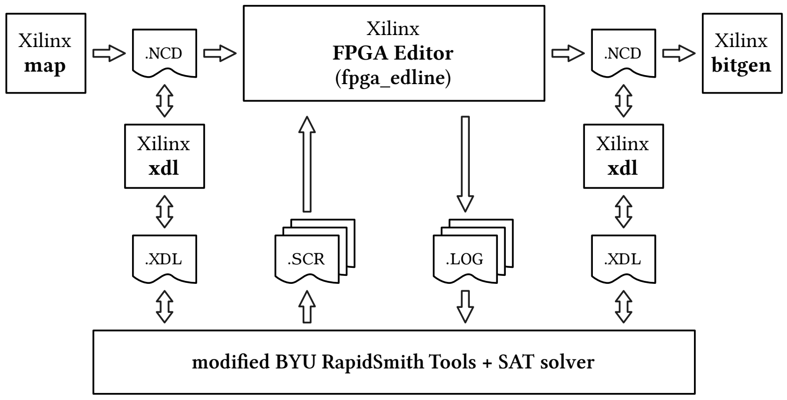


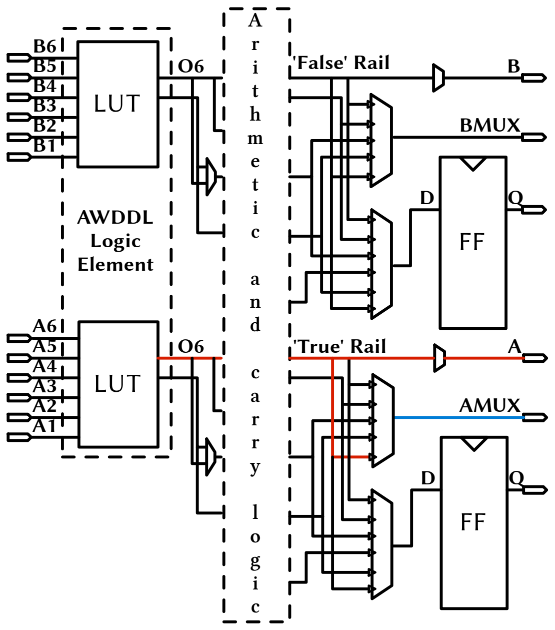
Fig. 4. Illustration of the custom workflow to process designs containing dual-rails requiring balanced routing

every gate requires to have both `TRUE` and `FALSE` rails of all input signals. For the same reason we cannot make use of the approach presented in [9] because their scheme is also based on complete separation of `TRUE` and `FALSE` networks. Therefore, we need to develop a mechanism capable of routing dual-rail signals as balanced as possible. While it is unlikely to achieve a perfectly balanced routing due to the given structure of the FPGA, it is likely to reduce the leakage compared to the default routing of the FPGA standard tools. We tried to achieve this by developing a customized router which is presented hereafter.

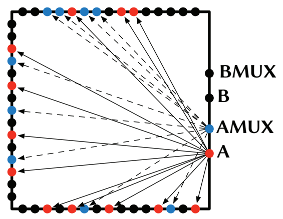
3.1 Customized Router

As stated before, our target is a Xilinx Virtex-5 FPGA; therefore, we could make use of the RapidSmith library [14]. In order to route balanced dual-rails we implemented a customized local router. It utilizes the Xilinx Design Language (XDL) and is based on a custom workflow as depicted in Fig. 4. Up to the end of mapping, the design is processed by the default Xilinx ISE tools. Note that for the case of Virtex-5 `map` performs the placement as well. The only modifications we made up to this step are i) to put all elements requiring balanced routing into a closed group which is area constrained, ii) to keep the input PIN positions of the LUTs realizing the AWDDL gates locked, and iii) to put LUTs of each AWDDL gate into the same slice using the LUTs (A, B) or (C, D) for $(\mathsf{Z_t}, \mathsf{Z_f})$ respectively.

After mapping, the intermediate file is processed by our customized router. The first step is to extract all dual-rail connections (source, destination) from the given data structure. This is done by using a simple naming scheme to detect corresponding nets within the XDL file. The next step is to find a set of possible routings for each of the dual-rail connections. To do so, for each possible output of a LUT (e.g., A and AMUX), all possible exit nodes of the adjacent switch box are used once to find a possible route. This idea is illustrated in Fig. 5. The



(a) Each pair of LUTs hold a single AWDDL logic element. Hence, a maximum of two AWDDL gates per slice is possible.



(b) A simplified representation of a switch box. Blue and red dots represent PIPs that may be used as exit node.

Fig. 5. Basic idea how to create multiple routing candidates

routing itself for each of these candidates uses a maze router with a priority queuing, favoring those nodes with the least Manhattan distance. This process is executed for both connections of a dual-rail to finally make a set of possible routings for each of the dual-rail connections.

Since extracting the capacitance of each route is not feasible, we had to consider other metrics to give priorities to the different routings. They include:

- the signal delays, extracted using the command line tool `fpga_edline` (due to the phenomena expressed above),
- the number of switch boxes the signal passes (due to their significant role in amount of dynamic energy consumption),
- the number of Programmable Interconnect Points (PIP), i.e., internal connections of the switch boxes, and
- the type of the wire, e.g., `long`, `pent`, and `double`.

For each of the routing possibilities, the above mentioned properties are extracted. Properties not dependent on the delay are easily extracted by their XDL representation. The delay of each single route must be extracted using a script file (`.scr`) that controls `fpga_edline`. An example for a script is given below:

```
open design ncdFileName.ncf pcfFileName.pcf
setattr main edit-mode Read-Only
setattr main auto_load_delays true
select netName
delay
exit
exit⁵
```

The result of running this script is a log file (`.log`) that contains the delay information of every route within that net. It is therefore required to parse the log files and extract the only valid delay for the route made (see Fig. 4). Note that for each single route the corresponding NCD file must be generated. Then, the above script and process should be run to extract the associated delay.

Here we make the restrictions. Based on the extracted information (as explained above) we restrict the dual-rail routes based on a threshold for delay of each route, for the difference between delay of the rails, for the number of switch boxes each route passes, for the number of PIPs, and etc. As the last step, as given below, the output is converted into a Boolean satisfiability (SAT) problem to select a conflict-free routing. If the problem is satisfiable, the conflict-free setting is put together and written to a new XDL file. All the previously routed nets are then locked, while the remaining unrouted nets are auto-routed using `fpga_edline`.

3.2 Representing Routing as SAT

Let n denote the number of dual-rail connections that the router should make. We first make a collection $\mathbb{S} = \{S^1, S^2, \ldots, S^n\}$, where $S^{i \in \{1,\cdots,n\}}$ represents a set of possible routing candidates $\{s^i_1, s^i_2, \ldots, s^i_{n_i}\}$ for the dual-rail connection i.

Accordingly we define Boolean variables x^i_j indicating whether the dual-route s^i_j is selected. Clearly, one must select exactly one candidate s^i_j from each set S^i to achieve a complete routing. This requirement can be encoded using the following formula [12]:

$$\text{AtLeastOne}(S^i) = \bigvee_{j=1}^{n} x^i_j,$$

$$\text{AtMostOne}(S^i) = \bigwedge_{j=1}^{n-1} \bigwedge_{k=j+1}^{n} (\neg x^i_j \vee \neg x^i_k),$$

$$\text{ExactlyOne}(S^i) = \text{AtLeastOne}(S^i) \wedge \text{AtMostOne}(S^i).$$

Therefore, ExactlyOne(S^i) = `TRUE` for $\forall\, i \in \{1, \ldots, n\}$ are added as clauses to the SAT.

Another issue is related to the loop which must be made at every LUT. As stated in Section 2.1, the output of the LUT must be presented as one of

⁵ Using the exit command twice is required to properly exit the command-driven mode in addition to gracefully terminate the tool.

its inputs to realize the internal S-R latch. For simplicity and consistency we tried to make the same loop at every LUT which is used as AWDDL gate. To achieve this we define collection $\mathbb{S}^\star = \{S^{\star 1}, S^{\star 2}, \ldots, S^{\star l}\}$, where l denotes the number of possible dual-rail loop routings, and $S^{\star i \in \{1,\ldots,l\}}$ a set of the same dual-rail loop routing for all AWDDL gates of the design, i.e., $\{s_1^{\star i}, \ldots, s_m^{\star i}\}$, where m stands for the number of AWDDL gates in the design. Therefore, only one of these sets amongst collection $\mathbb{S}^\star$ must be selected. Accordingly we define Boolean variables $x_j^{\star i}$ due to the selection of the routing $s_j^{\star i}$. Moreover, a set of l commander-variables $C = \{c_1, \ldots, c_l\}$ are defined indicating the selection of $S^{\star 1}, \ldots, S^{\star l}$.

In order to consider the commander-variables into the SAT, one needs to include ExactlyOne$(C) =$ `TRUE` to make sure that only one loop set is selected. Moreover, the following formula must be also considered to prevent a selection of a mixture of different loop sets:

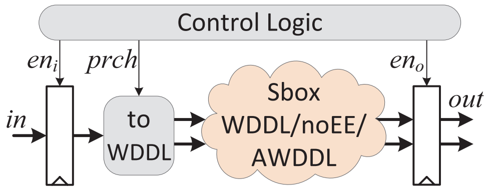
$$\text{AllFalse}(S^{\star i}) = c_i \vee \bigwedge_{j=1}^{m} \neg x_j^{\star i} \quad \text{AllTrue}(S^{\star i}) = \neg c_i \vee \bigwedge_{j=1}^{m} x_j^{\star i}.$$

Therefore, AllFalse$(S^{\star i}) =$ `TRUE` and AllTrue$(S^{\star i}) =$ `TRUE` for $\forall\, i \in \{1, \ldots, l\}$ are added to the SAT.

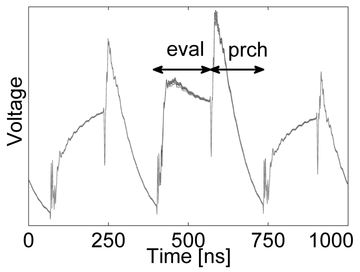
We should emphasize that two slices are connected to the same switch box, and we use either (A, B) or (C, D) LUTs to realize each AWDDL gate. Since the loop routing possibilities of these cases are different, we have to consider four different $\mathbb{S}^\star$ collections (and four different commander-variable set C respectively) to cover all these cases.

However, the above illustrated expressions do not reflect possible routing conflicts yet. The conflicts must also be encoded by doing a pairwise comparison of all possible routing candidates, and the corresponding clauses must be added to the SAT. Suppose that x_j^i and $x_{j'}^{i'}$ are corresponding Boolean variables of two conflicting routings s_j^i and $s_{j'}^{i' \neq i}$. So, $\neg x_j^i \vee \neg x_{j'}^{i'} =$ `TRUE` must be added to the SAT. This should be done for all possible pairwise conflicting routings (extracted by means of RapidSmith) including those which can be between collections $\mathbb{S}$ and $\mathbb{S}^\star$. This encoding can be realized in $\mathcal{O}(n^2)$. It should be noted that if the SAT solver fails to find a conflict-free solution, the restrictions – explained at the end of Section 3.1 – to make collections $\mathbb{S}$ and $\mathbb{S}^\star$ should be relaxed and the process should be repeated.

The runtime of the whole process is determined by the slow file conversion (Xilinx ISE tools) from NCD to XDL and vice-versa. This step needs to be executed for each delay extraction and though being massively parallelized using 16 cores (siblings), still takes around 6 hours for a design including $m = 122$ AWDDL gates and 8 additional LUTs for the single-to-dual rail conversion, where $n = 606$ dual-rail connections should be made. In contrast, SAT encoding typically requires about 20 minutes and solving less than a minute using `CryptoMiniSat 2.9.4` [1].



(a)



(b)

Fig. 6. (a) The exemplary design block diagram, (b) Superimposition of 256 mean traces of the WDDL design

4 Practical Investigations

In order to examine the effectiveness of our proposed schemes we made an exemplary design which in addition to surrounding and control logics consists in an AES S-box. We have taken the area-optimized S-box by Canright [4], and manually instantiated all the logic by 2-input AWDDL gates (in sum 122). A block diagram of the design is shown by Fig. 6(a). The Virtex-5 FPGA (XC5VLX50) of the side-channel evaluation platform SASEBO-GII [2], on which our target design is embedded, receives an input byte from the PC (via a controlling FPGA) and stores it into the input register by controlling en_i signal. At a certain clock cycle, the control logic disables *prch* signal, and the "to WDDL" conversion unit (the same scheme shown in the left part of Fig. 1(a)) propagates the dual-rail input to the AWDDL AES S-box thereby initializing the evaluation phase. In the next half of the clock cycle the control logic enables *prch* signal and the precharge phase is started. In a common WDDL circuit en_o should be active at the start of the precharge phase in order to store the output of the combinatorial circuit (here the AES S-box). However, since we aim at evaluating only the leakage associated to the combinatorial circuit, we must exclude the leakage of the output register (see [19]). Therefore, the control logic does not enable en_o signal and the register does not store the S-box output[6]. During these two (evaluation and precharge) phases the power consumption of the Virtex-5 FPGA is measured using a LeCroy WaveRunner HRO 66Zi oscilloscope at the sampling rate of 1GS/s while the design runs at a clock frequency of 3 MHz.

At the first step, we defined an area in the target FPGA for the placement of AWDDL gates, and as stated before we constrained the placer to assign two LUTs of the same slice to each AWDDL gate. At this stage we did not apply our customized router and used the default ISE routing tools. For the sake of similarity and fair comparison, we made also a WDDL and DPL_noEE com-

[6] In order to check the correct functionality of the circuit, en_o signal becomes active by the control logic in another clock cycle which is not covered by the measured power traces.

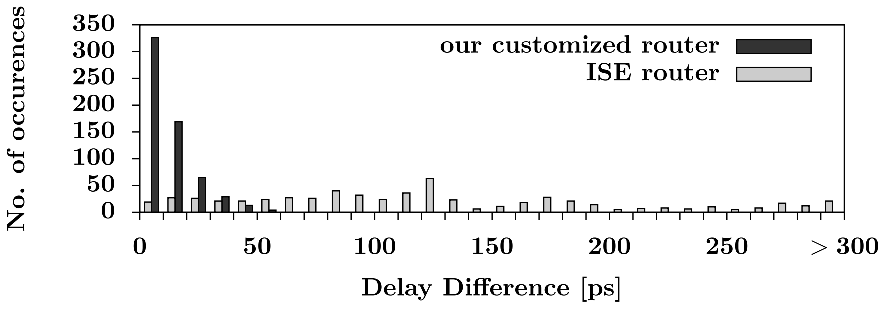


Fig. 7. Histogram of the delay difference of all dual-rail routes of AWDDL designs

patible version of the fully placed-and-routed AWDDL design. This has been done by editing the XDL file (of the AWDDL design) and only modifying the content of the LUTs[7]. So, the placement and routing of all these three designs are the same allowing for a fair comparison. As the fourth design we used our customized router in order to route the AWDDL design while its placement has not been altered.

In sum we evaluated four design profiles as:

1. WDDL AES S-box routed by ISE,
2. DPL_noEE AES S-box, the same placement and routing as profile 1,
3. AWDDL AES S-box, the same placement and routing as profile 1,
4. AWDDL AES S-box, the same placement as profile 1, but routed by our customized router.

As listed in Section 3.1, we have considered many different criteria to make routing collections $\mathbb{S}$ and $\mathbb{S}^\star$. The best result is achieved by considering a delay difference below 60 ps, the realization of identical feedback loops, and by considering minimum number of switch boxes and PIPs in each route. It should also be mentioned that we used the same wire type for both rails of a signal and prohibited to use `long` wires.

In order to compare the result of our routing with that of ISE, we provided two histograms illustrating the difference between the delay of each dual-rail signal in the AWDDL S-box circuit. Although it is not possible to find dual-rail routes with zero delay difference for all 606 dual-rail connections, the histograms shown in Fig. 7 indicate the effectiveness of our approach. Nevertheless, it is worth to mention that the average and the worst delay difference in case of our router are 11.7 ps and 58 ps respectively. These numbers are incomparable to those obtained by the ISE router as 125 ps and 520 ps respectively.

To evaluate and fairly compare the side-channel leakage of our target designs we applied the information theoretic (IT) analysis of [29] as it has been used for

[7] Although it is possible to realize each WDDL as well as DPL_noEE gate by a 5-to-2 LUT, we made the two 6-to-1 LUT version to keep it as similar as the AWDDL one and to follow the same design architecture given in [3].

the same purpose e.g., in [16]. So, we collected $512,000$ traces for each target design, i.e., 2000 traces for each S-box input value. By estimating the mean and variance of the traces for each S-box input we obtained 256 Gaussian distributions at each sample point (256 mean traces of the WDDL design are shown by Fig. 6(b) where evaluation and precharge phases are marked). It allows us to estimate the probability distribution of the leakages essential in the IT analysis. Here we should emphasize two points:

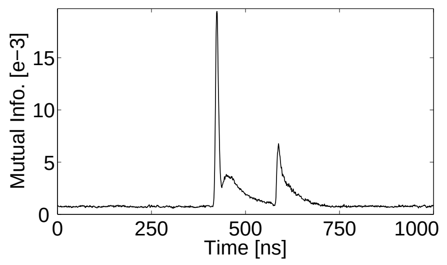
- There is no source of randomness in our exemplary designs, and electrical noise – well modeled by Gaussian [17] – is the only noise source in our measured traces. Therefore, ignoring the higher statistical moments in probability estimations – as in Gaussian – does not cause any information loss.
- The AES S-box circuit operates in precharge-evaluation mode, and in contrast to a CMOS combinatorial circuit its leakage does not depend on two consecutive inputs (input transitions). Therefore, our selection of estimating the probabilities based on the S-box input is a valid choice (the same is given in [16]).

Performing the IT analysis using the mean and variance traces of each of our target designs led to the mutual information curves presented by Fig. 8. As expected, the WDDL design – due to its data-dependent time-of-evaluation and time-of-precharge – has the highest leakage. Interestingly the DPL_noEE and the AWDDL designs have relatively the same amount of leakage in evaluation phase as they operate the same in this phase. However, DPL_noEE has a higher leakage in the precharge phase. It indeed confirms our claim in Section 2 that due to the early propagation of DPL_noEE in the precharge phase, in presence of imbalanced routings its leakage should be more easily detectable compared to that in the evaluation phase.

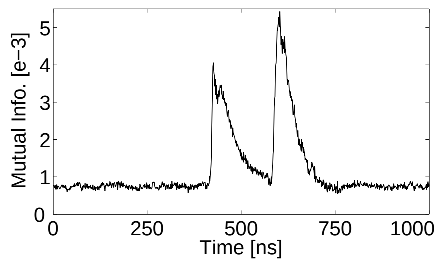
A comparison between the result of the AWDDL designs (routed by ISE vs. routed by our customized router) clearly shows the effectiveness of our developed router to reduce the information leakage. We should stress again that except the LUTs' configuration all details and specification of the first three designs are the same, that allowed us to fairly compare these logic styles. The same holds for the two AWDDL designs which only differ in the routing of the AES S-box circuit.

Independent of the attack strategy, IT analysis captures the amount of information available to the worst-case adversary. In order to quantify the data complexity (the number of required traces) of attacks on our target designs, we performed first-order profiling moments-correlating DPA [21]. Indeed, for each design profile we used a set of $100,000$ profiling traces to estimate first-order moments, and made use of them as power models to perform a CPA attack on another set of $100,000$ traces. The corresponding results are shown by Fig. 9. Thanks to the metric feature of moments-correlating DPA, we can directly conclude the following ratios between the data complexity of the attack on different profiles:

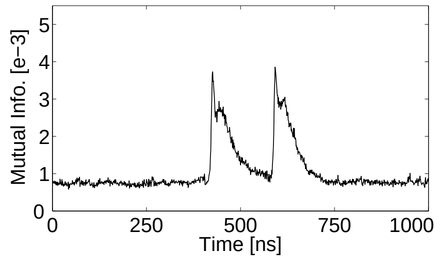
- DPL_noEE versus WDDL: $\left(\frac{0.111}{0.067}\right)^2 = 2.7$,



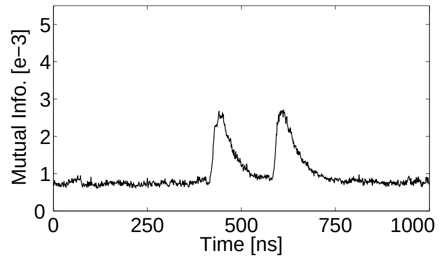
(a) WDDL



(b) DPL_noEE



(c) AWDDL



(d) AWDDL (custom routing)

Fig. 8. Mutual information curves for all profiles

- AWDDL versus DPL_noEE: $\left(\frac{0.067}{0.054}\right)^2 = 1.5$,
- AWDDL (custom routing) versus AWDDL: $\left(\frac{0.054}{0.038}\right)^2 = 2.0$.

As a side note, though the leakage extractable from our AWDDL design is mitigated, it is not a perfect solution to prevent a key-recovery attack. Therefore – as it is well known – DPA-resistant logic styles, e.g., AWDDL, should be combined with other countermeasure such as algorithmic masking which usually cannot prevent DPA attacks when implemented in hardware [18].

5 Conclusions

In this work we have shown how to design WDDL gates for FPGA platforms with independent time-of-evaluation and time-of-precharge. This, achieved by realizing a latch inside every LUT by means of a feedback loop, could guarantee the disappearance of early propagation in both evaluation and precharge phases. Our practical investigations confirm that by using our designed AWDDL style the level of security improves when compared to classical WDDL or to its main competitor DPL_noEE of [3]. However, routing imbalances still impose a threat to the security of dual-rail precharge logic. Therefore, as the second contribution of this work we developed a customized tool to reduce this imbalance by selecting the most similar routes for the signals of a dual-rail connection. This approach, whose effectiveness has been demonstrated using our proposed logic style, could

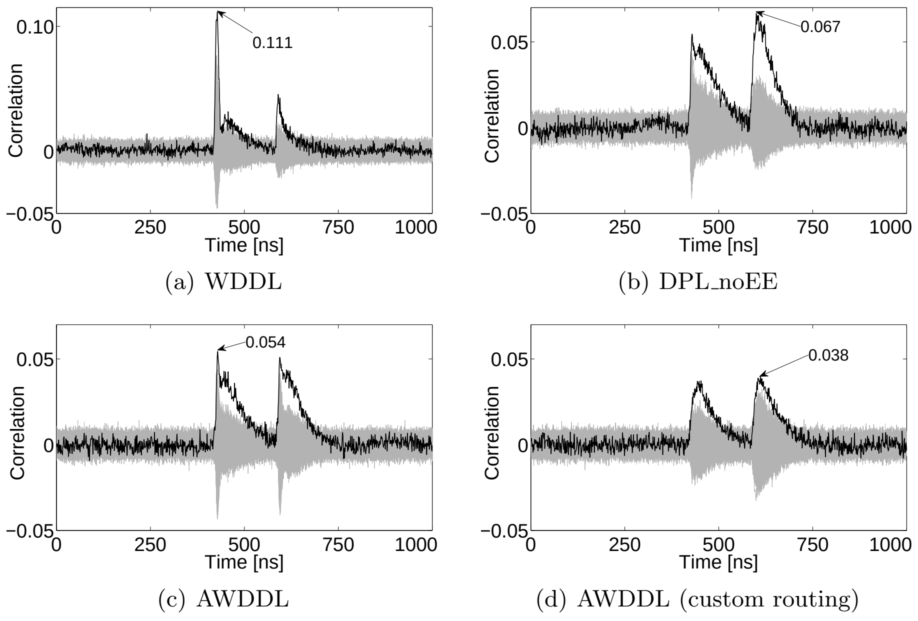


Fig. 9. Result of first-order profiling moments-correlating DPA on all profiles

also be applied to similar logic styles or other applications requiring balanced routes, e.g., TRNGs and PUFs. It is noteworthy to mention that applying our customized router does not cause any area overhead. In fact, it only changes the way the routing resources (PIPs) are configured.

The only available source for delay of the signal routes is the ISE tool. Therefore, the effectiveness of a customized router relies on the conformity of ISE reports and the underlying FPGA chip. Due to the process variation as well as publicly unknown architecture of the FPGAs these numbers might be different from chip to chip or (even slightly) different to reality. Hence, as a future work, we plan to develop a mechanism to practically examine the differential delay as well as the power consumption of the dual-rail routings based on the target FPGA chip, where the design is supposed to be realized.

Acknowledgment

The authors would like to thank Ali Ahari, from Sharif University of Technology (Tehran, Iran), for useful discussions on routing features of RapidSmith.

References

1. CryptoMiniSat. Available as download here https://gforge.inria.fr/frs/?group_id=1992.
2. Side-channel Attack Standard Evaluation Board (SASEBO). Further information are available via http://www.morita-tech.co.jp/SASEBO/en/index.html.
3. S. Bhasin, S. Guilley, F. Flament, N. Selmane, and J.-L. Danger. Countering early evaluation: an approach towards robust dual-rail precharge logic. In *WESS 2010*, page 6. ACM, 2010.
4. D. Canright. A Very Compact S-Box for AES. In *CHES 2005*, volume 3659 of *LNCS*, pages 441–455. Springer, 2005.
5. Z. Chen and Y. Zhou. Dual-Rail Random Switching Logic: A Countermeasure to Reduce Side Channel Leakage. In *CHES 2006*, volume 4249 of *LNCS*, pages 242–254. Springer, 2006.
6. S. Guilley, P. Hoogvorst, Y. Mathieu, and R. Pacalet. The "Backend" Duplication Method. In *CHES 2005*, volume 3659 of *LNCS*, pages 383–397. Springer, 2005.
7. T. Güneysu and A. Moradi. Generic Side-Channel Countermeasures for Reconfigurable Devices. In *CHES 2011*, volume 6917 of *LNCS*, pages 33–48. Springer, 2011.
8. W. He, E. de la Torre, and T. Riesgo. A Precharge-Absorbed DPL Logic for Reducing Early Propagation Effects on FPGA Implementations. In *ReConFig 2011*, pages 217–222. IEEE Computer Society, 2011.
9. W. He, A. Otero, E. de la Torre, and T. Riesgo. Automatic generation of identical routing pairs for FPGA implemented DPL logic. In *ReConFig 2012*, pages 1–6. IEEE, 2012.
10. C. Herbst, E. Oswald, and S. Mangard. An AES Smart Card Implementation Resistant to Power Analysis Attacks. In *ACNS 2006*, volume 3989 of *LNCS*, pages 239–252. Springer, 2006.
11. J.-P. Kaps and R. Velegalati. DPA Resistant AES on FPGA Using Partial DDL. In *FCCM 2010*, pages 273–280. IEEE Computer Society, 2010.
12. W. Klieber and G. Kwon. Efficient CNF Encoding for Selecting 1 from N Objects. In *Workshop on Constraints in Formal Verification - CFV*, 2007.
13. P. C. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *CRYPTO 1999*, volume 1666 of *LNCS*, pages 388–397. Springer, 1999.
14. C. Lavin, M. Padilla, J. Lamprecht, P. Lundrigan, B. Nelson, B. Hutchings, and M. Wirthlin. RapidSmith – A Library for Low-level Manipulation of Partially Placed-and-Routed FPGA Designs. Technical report, Brigham Young University, September 2012.
15. V. Lomné, P. Maurine, L. Torres, M. Robert, R. Soares, and N. Calazans. Evaluation on FPGA of triple rail logic robustness against DPA and DEMA. In *DATE 009*, pages 634–639. IEEE, 2009.
16. F. Macé, F.-X. Standaert, and J.-J. Quisquater. Information Theoretic Evaluation of Side-Channel Resistant Logic Styles. In *CHES 2007*, volume 4727 of *LNCS*, pages 427–442. Springer, 2007.
17. S. Mangard, E. Oswald, and T. Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, 2007.
18. S. Mangard, N. Pramstaller, and E. Oswald. Successfully Attacking Masked AES Hardware Implementations. In *CHES 2005*, volume 3659 of *LNCS*, pages 157–171. Springer, 2005.

19. A. Moradi, T. Eisenbarth, A. Poschmann, and C. Paar. Power Analysis of Single-Rail Storage Elements as Used in MDPL. In *ICISC 2009*, volume 5984 of *LNCS*, pages 146–160. Springer, 2009.
20. A. Moradi, M. Kirschbaum, T. Eisenbarth, and C. Paar. Masked Dual-Rail Precharge Logic Encounters State-of-the-Art Power Analysis Methods. *IEEE Trans. VLSI Syst.*, 20(9):1578–1589, 2012.
21. A. Moradi and F.-X. Standaert. Moments-Correlating DPA. Cryptology ePrint Archive, Report 2014/409, 2014. `http://eprint.iacr.org/`.
22. D. E. Muller and W. S. Bartky. A Theory of Asynchronous Circuits. Report no. 78 at University of Illinois at Urbana-Champaign. Dept. of Computer Science, 1959.
23. M. Nassar, S. Bhasin, J.-L. Danger, G. Duc, and S. Guilley. BCDL: A high speed balanced DPL for FPGA with global precharge and no early evaluation. In *DATE 2010*, pages 849–854. IEEE, 2010.
24. S. Nikova, V. Rijmen, and M. Schläffer. Secure Hardware Implementation of Non-linear Functions in the Presence of Glitches. *J. Cryptology*, 24(2):292–321, 2011.
25. E. Oswald, S. Mangard, N. Pramstaller, and V. Rijmen. A Side-Channel Analysis Resistant Description of the AES S-Box. In *FSE 2005*, volume 3557 of *LNCS*, pages 413–423. Springer, 2005.
26. T. Popp, M. Kirschbaum, T. Zefferer, and S. Mangard. Evaluation of the Masked Logic Style MDPL on a Prototype Chip. In *CHES 2007*, volume 4727 of *LNCS*, pages 81–94. Springer, 2007.
27. T. Popp and S. Mangard. Masked Dual-Rail Pre-charge Logic: DPA-Resistance Without Routing Constraints. In *CHES 2005*, volume 3659 of *LNCS*, pages 172–186. Springer, 2005.
28. L. Sauvage, M. Nassar, S. Guilley, F. Flament, J.-L. Danger, and Y. Mathieu. DPL on Stratix II FPGA: What to Expect? In *ReConFig 2009*, pages 243–248. IEEE Computer Society, 2009.
29. F.-X. Standaert, T. Malkin, and M. Yung. A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 443–461. Springer, 2009.
30. D. Suzuki and M. Saeki. Security Evaluation of DPA Countermeasures Using Dual-Rail Pre-charge Logic Style. In *CHES 2006*, volume 4249 of *LNCS*, pages 255–269. Springer, 2006.
31. K. Tiri, M. Akmal, and I. Verbauwhede. A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards. In *Solid-State Circuits Conference - ESSCIRC 2002*, pages 403–406, 2002.
32. K. Tiri and I. Verbauwhede. A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation. In *DATE 2004*, pages 246–251. IEEE Computer Society, 2004.
33. K. Tiri and I. Verbauwhede. Place and Route for Secure Standard Cell Design. In *CARDIS 2004*, pages 143–158. Kluwer, 2004.
34. R. Velegalati and J.-P. Kaps. Techniques to enable the use of Block RAMs on FPGAS with Dynamic and Differential Logic. In *ICECS 2010*, pages 1244–1247. IEEE, 2010.
35. Xilinx. Virtex-5 Libraries Guide for HDL Designs. Available via `http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_1/virtex5_hdl.pdf`, April 2012.
36. P. Yu and P. Schaumont. Secure FPGA circuits using controlled placement and routing. In *CODES+ISSS 2007*, pages 45–50. ACM, 2007.