# SafeDRP: Yet Another Way Toward Power-Equalized Designs in FPGA

Maik Ender, Alexander Wild, and Amir Moradi

Horst Görtz Institute for IT-Security, Ruhr-Universität Bochum, Bochum, Germany
{firtsname.lastname}@rub.de

**Abstract.** Side-channel analysis attacks, particularly power analysis attacks, have become one of the major threats, that hardware designers have to deal with. To defeat them, the majority of the known concepts are based on either masking, hiding, or rekeying (or a combination of them). This work deals with a hiding scheme, more precisely a power-equalization technique which is ideally supposed to make the amount of power consumption of the device independent of its processed data. We propose and practically evaluate a novel construction dedicated to Xilinx FPGAs, which rules out the state of the art with respect to the achieved security level and the resource overhead.

## 1 Introduction

Unintended communication channels of a cryptographic device may leak information about the processed data. These channels – also known as side channels – can be used to recover a secret key from the device. Targeting the power consumption as a side channel, in the literature the corresponding attack is known as Power Analysis (PA) attacks. As a powerful and low-cost attack vector, it makes use of statistical dependencies between the processed data and the power consumption of the cryptographic device. Amongst the known countermeasures, those which are known as *hiding* techniques mainly try to reduce the exploitability of the leakages [12]. To this end, one solution for hardware platforms is to equalize the power consumption, and hence decorrelate the leakage from the processed data. Such countermeasures usually follow the Dual-Rail Precharge (DRP) concept, where the source of the data-dependent power consumption, i.e., transition in transistors, is addressed. Among several such schemes, we can refer to SABL [27], WDDL [28], DRSL [7], MDPL [21], and iMDPL [20], which have particularly been designed for Application-Specific Integrated Circuits (ASICs). However, due to the predefined structure and limited routing resources, they cannot be easily employed on Field Programmable Gate Arrays (FPGAs). It is noteworthy that several other works, e.g., [4, 8–11, 13, 14, 18, 22, 32], have already tried to adopt the concept of DRP schemes to FPGAs. Such schemes reduce the vulnerability against PA attacks, but almost all of them suffer from at least one of the known major pitfalls: the early propagation effect, glitches, and imbalanced routings. To the best of our knowledge, GliFreD [17] is of the rare FPGA-based

solutions addressing all pitfalls by a massive utilization of Flip Flops (FFs). Note that a more optimized variant of GliFreD has been introduced in [30].

In this work, as an FPGA-based power-equalizing solution, we introduce SafeDRP which also avoids the three aforementioned major problems. The selling point of SafeDRP is its low resource overhead. More precisely, it significantly reduces the number of utilized FFs with the price of a more complicated control logic. We investigate its effectiveness by a case study, i.e., an Advanced Encryption Standard (AES) encryption engine on a Kintex-7 FPGA. Our investigations include resource overhead as well as practical side-channel analysis evaluations. We further introduce a process on how to convert an unprotected circuit into its SafeDRP-variant.

It is noteworthy that power-equalization schemes in general cannot fully avoid the exploitability of leakages due to e.g., imperfection of balanced routings. Instead, in case of effective solutions, they can reduce the leakage interpreted by e.g., lower Signal-to-Noise ratio (SNR) leading to harder attacks with respect to the number of required side-channel measurements. Therefore, a combination of such a solution with a sound masking scheme, e.g., Threshold Implementation (TI) [19], is known to provide a high level of practical security (e.g. [17]). Hence, a suitable construction would be to implement a masked design under the concept of SafeDRP. However, in order to solely examine the effectiveness of SafeDRP, we considered an ordinary (not masked) design as the case study. Since the concept of SafeDRP and GliFreD are relatively similar, the same security achievements as in [17] are expected if TI and SafeDRP are merged.

## 2 Background

As the name says, the Dual-Rail Precharge (DRP) logic is a combination of Dual-Rail (DR) logic and precharge logic. DR logic makes use of a differential encoding of the signals, where every signal $a$ is encoded to $(a, \bar{a})$. Hence, a logic Hi is encoded to $(1, 0)$ and a logic Lo to $(0, 1)$. In general, a DR gate expects differentially-encoded input signals, and also produces a differentially-encoded output. Therefore, a DR gate has a double number of input and output signals compared to a functionally-equivalent single-rail gate. Commonly, the differential encoding of DR logic forms two networks which are often noted as positive and negative network.

A circuit built upon precharge logic alternates between a precharge and an evaluation phase. During the precharge phase, the gates of the circuit propagate a predefined constant value. During the evaluation phase, the gates evaluate the data intended to be processed by the circuit. Hence, the input and output signals of a circuit built in precharge logic are set to a constant value before the given data set is processed.

The combination of both (DR and precharge) techniques forms the concept of DRP logic. Hence, DRP uses a differential encoding of their signals which alternate between a precharge and evaluation phase. During the precharge phase,

the encoded signals are set to a constant value, i.e., $(0,0)$ or $(1,1)$. Hence, at a phase transition from precharge to evaluation exactly one of the wires changes their state, i.e., $(0,0) \rightarrow (1,0)$ or $(0,0) \rightarrow (0,1)$ (when precharge is $(0,0)$). The same holds for the transition from the evaluation phase back to precharge. This behavior can be scaled from a single gate to a full circuit, which results in a circuit with a constant number of wire transitions, independent of the data it processes. Hence, following the DRP concept, a circuit forms a promising basis to equalize the dynamic power consumption independent of the data it processes.

Beyond constant number of wire transitions, DRP schemes have to deal with three major pitfalls. The first pitfall is the *early evaluation* effect, which occurs if the scheme shows a data-dependent time of evaluation. The second is known as *glitches*, which are temporary, faulty transitions of a gate output signal. Thus, the output should change only once per phase transition. Therefore, a DRP scheme has to ensure that all input signals are stable at the gate's inputs before it evaluates. The third problem is *imbalanced routing*, which occurs especially in FPGAs due to the limited routing resources. The routing of the positive and negative networks need to be identical with respect to their length, i.e., capacitance. Otherwise, the power consumption to load coupled wires will be different and hence impede the aimed power equalization.

FPGAs are integrated circuits with reprogrammable logic cells. The cells are organized on a grid. Every cell can be individually addressed by its X/Y-coordinates in the grid. The first level of the hierarchy is the clock region. Next the tiles group different classes of distinct elements in the grid, such as the slices and specialized elements like Block RAM (BRAM) and Digital Signal Processors (DSPs). A slice holds several Look-Up-Tables (LUTs) and FFs which are the basic elements in an FPGA[1]. A LUT is the reprogrammable logic element of the FPGA, it evaluates every Boolean function based on the configuration. The LUT is realized by SRAM cells and a multiplexer tree. After each LUT a FF or rather a latch is placed which can store the LUT's output.

The connection between all elements is realized by a routing engine. The routes between the elements are reprogrammable as well. Reprogrammable switching matrices before a group of elements (e.g. two slices) connect these groups with distinct hardwired routes. Beside the routes between the elements, an FPGA is equipped with a clock network, since clock signals have usually a high fan-out and need a minimal skew to guarantee the proper instantiation of synchronous circuits. This clock network is also known as *clock tree*, which has a direct connection to every clocked element placed on the FPGA. Depending on the FPGA architecture, the clock tree is able to reach non-clocked elements as well.

Our proposed scheme is implemented and evaluated on a Xilinx 7-Series FPGA which makes use of the Vivado tool flow. The Vivado Design Suite is used to synthesis, translate, map, and placed and route a Hardware Description Language (HDL) code to generate a bitstream, which programs the FPGA. It also supports the Tool Command Language (Tcl) command line interface and

[1] The exact number of available resources in an FPGA highly depends on its device family and differs between the available architectures and manufacturer.

scripting language which are essential for our work. With Tcl we are able to manipulate objects within the design after each synthesis step.

## 3 Concept

The majority of cryptographic algorithms implemented on FPGAs are based on the standard components like LUTs, FFs and latches. Hence, the concept of SafeDRP focuses on these components. This does not automatically conclude that other dedicated hardware components cannot be instantiated in a SafeDRP-based design. For instance, the inclusion of BRAMs may be achieved by adapting a concept proposed in [3].

### 3.1 Controlling LUTs

In order to define a proper DRP scheme we need to address the problem of early evaluation and glitches. Therefore, we define control signals to trigger the phase transition of the LUTs. The number of used control signals, further referred as *active* signals, highly depends on the underlying hardware and the logic depth of the hardware design. Figure 1(b) shows the waveforms of four *active* signals, connected to the simple circuit given in Figure 1(a). As shown, the *active* signals are periodic and run at the same frequency. They are phase shifted to each other, and come up with different duty cycles.

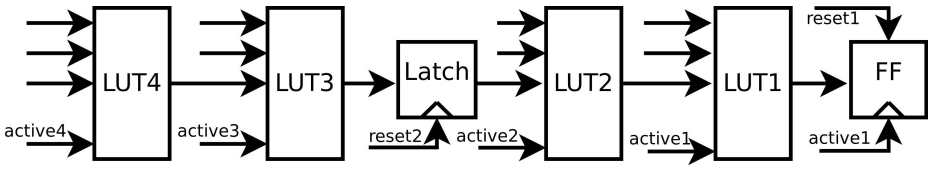
Each LUT is connected to one of the *active* signals which controls the LUT's precharge and evaluation phases. Note that this grants no restriction to the LUT's logical function but reduces the number of available inputs per LUT by one. It is because the LUT functionality must be of the form

$$f'(x) = active \wedge f(x), \tag{1}$$

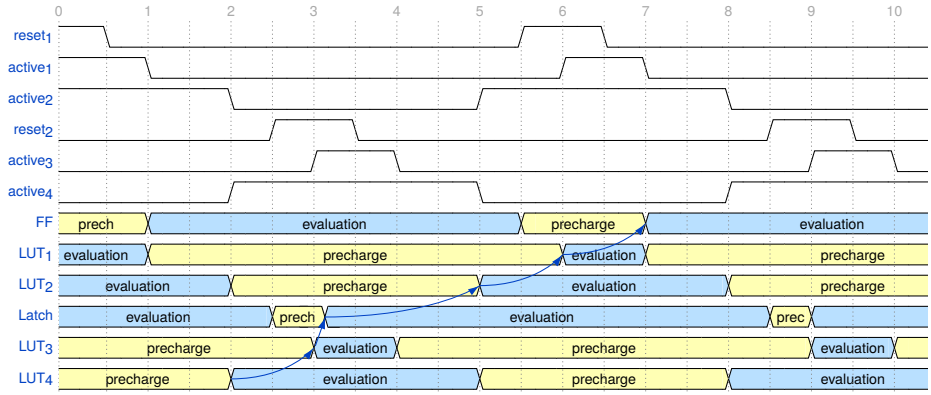
while the LUT evaluates at *active*=Hi and is set to precharge on *active*=Lo. In order to avoid glitches at the LUT output, the *active* signal is supposed to be the last arriving signal at the LUT to trigger the phase transition after all input signals are stable. Therefore, consecutive LUTs are not connected to the same *active* signal.

In detail, a combinatorial circuit is organized in LUT stages. The stage of a LUT is defined by the maximum number of LUTs the input signals have to pass to reach it. The LUTs with the same stage label are connected to the same *active* signal. The *active* signals of consecutive stages are slightly phase shifted so that they evaluate one after another. The reduced duty cycle ensures a glitch-free transition from evaluation to precharge of the stages in reverse order. A small exemplary circuit with corresponding waveforms is given in Figure 1

The *active* signals define the phase transitions of the LUTs, and are hence very critical signals which require a low skew. Therefore, the *active* signals are routed via the special routing network called *clock tree*. To reach the first LUT




(a) Exemplary Circuit




(b) Waveform for *active* signals and phases of LUTs, FF and latch of the exemplary circuit

**Fig. 1.** Visualization of the SafeDRP concept.

input (and the FF's reset pin), the clock tree is left in the switching matrix attached to the slice. It is noteworthy that a switch matrix is limited to move just two signals from the clock tree to the logic.

### 3.2 Controlling Memory Elements

FFs and latches play an important role in sequential circuits. In a DRP scheme, a FF is mostly built on a master-slave fashion, as one of the two FFs must be in precharge while the other FF holds the data value. Alternatively, it is not mandatory to consecutively place the FFs in a design. To minimize the critical path and hence increase the performance of a circuit, one of the FF stages can be moved into the middle of the combinatorial circuit. This splits the combinatorial circuit into two parts. The control signals of the circuit parts work inverse to each other to keep the FFs alternating between precharge and evaluation. Hence, either the first part (LUTs and FFs) of the circuit is in precharge and the second part evaluates or the other way around. This can be seen in Figure 1 as well. In general, a combinatorial circuit can be split into an even number of parts while every second part is connected to the same control signals. The number of parts used to form the circuit depends on the circuit constraints. By increasing the number of parts, the critical path will be reduced which results in a higher clock frequency but also increases the latency of the circuit. Further, the overhead

with respect to the number of FFs increases as well but does not necessarily increase the area consumption of the design since the FPGA structure provides a FF right after each LUT which left unused in most cases.

The FF provides the data for the subsequent part, and can hence be precharged only when all stages of the subsequent part are in precharge. On the other hand, the data provided to be stored in the FF is just valid for a short period of time. Hence, the precharge phase of the FF has to be handled very carefully. SafeDRP handles this problem in two different ways which are depicted in Figure 1 as well. Both approaches make use of a *reset* signal which is phase shifted to the last *active* signal of a circuit part prior to the FF.

The first approach connects the *reset* signal to the reset pin of the FF and precharges it at the rising edge of the *reset* signal. The last *active* signal of the previous circuit part is also connected to the clock pin of the FF. The FF is negative edge-triggered and stores on falling edge of the *active* signal. At this time, the input signal of the FF is still valid and the subsequent circuit is in precharge. The drawback of this technique is the additional global *reset* signal which has to be moved from the clock tree to reset pin of the FF. As previously noted, the number of signals that can be moved from the clock tree to the logic is limited to two signals per Configurable Logic Block (CLB). This limitation can lead to problems during the place and route process. An additional problem of this method is the hold time of the LUT's data signal. At the falling edge of the *active* signal, the LUT starts going to the precharge phase, and at the same time the FF is triggered. If the wire between these two elements is too short, the data signal becomes invalid before the FF is able to store its input, hence a hold time violation[2].

The second approach replaces the edge-triggered FF with a level-triggered latch. The *reset* signal is connected to the clock pin of the latch which will be transparent at *reset*=Hi. Indeed, in this case the latch is not forced to reset, but the *reset* signal plays the role of the enable signal of the latch. The *reset* signal enables the latch slightly before the LUT (that is connected to the input pin of the latch) evaluates. Hence, the latch first passes the precharge value (which is the prechagred LUT output), and then holds the output after the LUT output is evaluated. Since all latches in a circuit should be controlled by the same *reset* signal, this method requires the entire latches to be connected to the last LUT stage of the underlying circuit part. In other words, the input of all latches should be supplied by the LUTs from the same stage. Otherwise, the latch(es) might not pass a precharged value.

### 3.3 Duplication

Facing the problem of imbalanced routing, the method proposed in [32] is utilized to create the positive and negative networks. Therefore, each cell from the positive network (e.g. LUTs and FFs) needs to be cloned and inverted. First,

[2] This failure appears on the Kintex-7 if the FF and the LUT controlled by the same active signal are placed at the same slice.

a single rail circuit (so-called positive network) which follows the principles explained above (Section 3.1 and Section 3.2) is fully placed and routed at a defined area on the FPGA. Second, the positive network is copied and placed at another equivalent reserved area on the FPGA while the relative routing and placement is retained. Third, by inverting the functionality of the copied circuit, the negative network of the design is formed. Indeed, the combination of both circuits follows the DRP definition and shows a balanced routing structure. More details about the duplication process is given in Section 4.3

### 3.4 Resources & Limits

Every LUT stage of the circuit needs one *active* signal, also an additional *reset* signal for the FF/latch is needed. Note that the invert of every *active* signal of one circuit part exists in the other part, except for *active* signals connected to the last LUT stage. Figure 2 shows that $active_2$ and $active_5$ or $active_3$ and $active_6$ are complementary. Therefore, the *active* signals of the first part can be reused in the second part, only the last LUT stage needs an individual *active* signal. Note that, the active signals are not inverted for the second part. Instead, the LUTs in the second part are configured to evaluate on *active*=Lo. It is useful to have the same number of stages in both parts to share the most number of *active* signals. In total, $d + 1 + 2$ *active* signals are required, where $d$ is the maximum logic depth of a circuit part, 1 individual *active* signal for the last LUT stage and 2 *reset* signals for the two FF/latch stages. Since the clock tree is used to distribute the *active* signals, the maximum logic depth is limited by the number of available clock trees. For example, on the 7-Series FPGAs 12 clock trees exist in every clock domain. Thus, the maximum logic depth for each circuit part is $d = 12 - 3 = 9$.

The duplication concept requires to invert the functionality of every logical element in the positive network. Some dedicated hardware components like the DSPs or multiplexer are not invertible and hence not usable in this construction. The *active* signal connected to the LUT reduces its functionality to an $(N-1)$-to-1 LUT, i.e., in the 7-Series only 5-to-1 LUTs can be used rather than the 6-to-1 LUTs. The duplication process doubles the consumed resources as the negative network is a copy of the positive network.

The maximum frequency of this construction strongly depends on the design, i.e., the logic depth of the circuit parts which defines the number of *active* signals and the critical path of every stage. Here the critical path is the maximal signal delay between two consecutive LUT stages. The duty cycle $duty_i$ at stage $i$ is defined by its critical path[3] $p_{i,max}$ and the duty cycle of the subsequent LUT stage. An exception is the last LUT stage, where the duty cycle depends only on the critical path from the last LUT stage to the FF/latch. This results in

[3] Since we reuse the *active* signals to control stages of all circuit parts, we should consider the highest stage delay at all circuit parts.

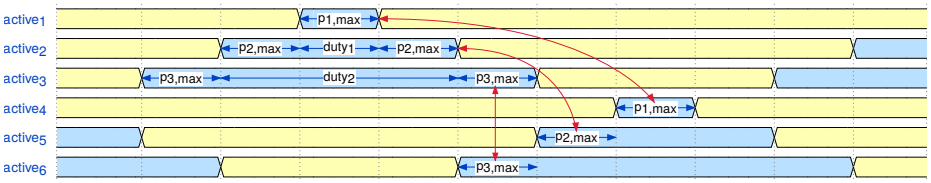



**Fig. 2.** The relation between the *active* signals, the composition of their duty cycles and phase shifts.

$$duty_i \geq \begin{cases} p_{1,max} & \text{if } i = 1 \\ 2 \cdot p_{i,max} + duty_{i-1} & \text{otherwise.} \end{cases}$$

The frequency is defined by the first LUT stage, as it has the highest duty cycle (e.g. $active_3$ or $active_6$), and a margin to reset the FF/latch. The *active* signals are phase shifted in a way that the delay between the rising edges of $active_{i-1}$ and $active_i$ is $p_{i,max}$. The duty cycle estimation and signal alignment is visualized in Figure 2.

In a practical instantiation, the device capabilities limit the phase shift, duty cycle and hence the frequency. For example, we can make use of the Mixed-Mode Clock Manager (MMCM) to generate the control signals (more details in Section 4.2), but its capabilities on the 7-Series FPGAs directly influence the discrete duty cycle steps and the minimal phase shift angle (see [31] for more detailed information).

It is worth to note that all control signals should become active only when the corresponding stage of the circuit is in the precharge phase in order to introduce no glitches. For example, a multiplexer, which selects two different signals, should only switch while both signals hold the precharge value. One can use the last LUT stage's *active* signal as a control clock to switch the other part, e.g. in Figure 2 the circuit connected to $active_4$ to $active_6$ can be controlled while $active_1$ is Hi.

## 4 Tooling

Developing a design which follows the above-illustrated concept is an intensive task. To reduce the workload and support hardware designers, this section introduces a tool flow which helps to transform arbitrary combinatorial circuits written in HDL into circuits which follows the concept of SafeDRP. Since the hardware design process highly depends on the FPGA architecture, we focus on the Xilinx 7-Series and the associated Vivado tool flow.

### 4.1 Circuit Mapping

As stated before, the phase transition of a LUT is determined by an *active* signal. Following the concept of [17], the *active* signal should be connected to the first multiplexer stage, i.e., first input pin of the LUTs. The Vivado tool flow does not support this constraint to append an *active* signal to each LUT during synthesis. Hence, it is required to map the HDL design into 5-to-1 LUTs and append the control signals afterwards. Further, the maximal logic depth of a SafeDRP-based circuit highly depends on the number of available clock trees, which requires to add memory elements into large combinatorial circuits.

To synthesize, optimize, edit, and map a given HDL code we used ABC [2], an open-source tool from the Berkeley Logic Synthesis and Verification Group. The tool strongly supports the needs of our construction and is natively capable to map a given HDL design into 5-to-1 LUTs. To reduce the manual overhead of a hardware designer, we extended ABC by two functions. First, our modified version of ABC is now capable to add memory elements to a combinatorial circuit after a defined logic depth. Second, each 5-to-1 LUT of the mapped design is replaced by a 6-to-1 LUT. Based on their logic depth, the corresponding *active* signal is attached to the LUTs first input pin which drives the first multiplexer stage. Further, the LUT's *INIT* attribute is extended to fulfill Equation 1.

### 4.2 Placement Restrictions

Notwithstanding, for a large design – e.g. an AES round function – the place and route algorithms of Vivado with standard settings could mostly not generate a routable SafeDRP design or rather a design with a higher leakage than expected, as the routing of non-clocking signals is challenging.

As stated before, we make use of clock trees to route *active* signals with low skew. Hence, to generate the *active* signals we use an MMCM, which is hardwired to the clock tree. The clock tree is also hardwired to each switch matrix located in the CLBs. As given before, the limited resources of the switch matrix allows to extend only two clock trees to data pins of the slices located in the CLB.

The placer is not aware of the constraint to connect the *active* signal to the LUTs' first input, and may also place two different LUT stages in one slice, which results in an unroutable design. It may also happen that the *active* signal leaves the clock tree at a different switching matrix and is routed via the routing fabric, which results in a larger skew. Hence, the placement and the routing problem need to be addressed. We consider two methods to overcome this problem. First, the placer can be restricted to use just a single LUT per slice which obviously results in a heavy area increase. Second, the placer is not restricted, but the placement has to be corrected manually. Swapping the LUTs between nearby slices can group the LUTs to from the same stage in one slice.

### 4.3 Design Duplication

The output of ABC forms just the positive network of SafeDRP. In order to address the problem of imbalanced routing, SafeDRP adapts the duplication

concept proposed in [32]. Hence, the placed-and-routed positive network is duplicated and inverted to form the negative network. The full duplication process is split into the following sub-processes:

1. Place an additional instance of the positive network at a reserved area on the FPGA and keep its placement and routing structure.
2. Invert the second instance's LUT functionality to form the negative network.
3. Logically connect all I/O signals of the negative network to the control logic.
4. Route the I/O signals of the negative network to the control logic.

Vivado includes the Tcl shell and scripting language, which is used to manipulate objects within the design. We are using a Tcl script to perform the duplication process and no third party tool is needed to manipulate the objects within the design. First, all cells of the positive network are cloned. Our script can deal with all primitives of the FPGA, which are in general LUTs, FFs, latches, and clock buffers. Consequently, a new object is created and all properties are copied from the positive cell to the new (negative) cell. The set of negative cell hence form the negative network of our scheme. To place the cells of the negative network at a different location, a constant value is added to the X-/Y-coordinate. Note that the addition of a constant to the location coordinates does not change the relative placement structure between the cells of the negative network. In order to invert the logic of the negative network, just the LUTs' content are changed, since only the LUTs define the logical function of the design. The LUT content is adjusted to fulfill Equation (2). The behavior of the *active* signal is maintained, as mentioned before.

$$active \wedge f_{\text{neg}}(x) = active \wedge \overline{f_{\text{pos}}(\overline{x})} \tag{2}$$

The routes of the positive network are cloned in a similar way. First, we have to make a distinction between internal nets, i.e., nets which connect only cells inside the SafeDRP-based circuit, and external nets, i.e., nets which connect the SafeDRP-based circuit with the remaining design (control logic). Internal nets contribute to the leakage and are important for the DRP logic, while external nets do not contribute to the leakage. Similar to internal signals, the routes of the *active* signal inside a clock region are kept equal during the duplication process. The remaining part of the *active* signals used to reach the clock region is handled separately and routed to the clock source. In order to route a given internal net of the positive network, a new net is created and connected to the respective negative cells. The routing information (located in the "ROUTE" property) of the positive route is copied to its negative pendant. Since this information is relative to its location and the relative placement of the negative network is equivalent to the positive network, no further changes are needed. Next, the external signals are connected to the negative network. Each connection of each external signal is connected to its negative pendant. In the last step another run of the Vivado routing algorithm is needed to route the external signals, while the already existing routes are preserved.

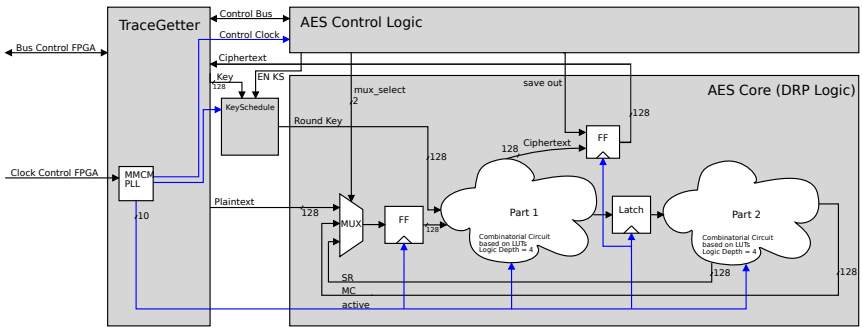



**Fig. 3.** The case study's design of an AES-128 on the Kintex-7.

## 5 Evaluation

In a case study we examine the effectiveness of our construction by an round-based AES-128 encryption on a Kintex-7 FPGA.

The used AES core is designed to compute one full round in one clock cycle. For the S-Boxes the area optimized Canright S-Box [6] is employed. Figure 3 shows a block diagram of the final design, while TraceGetter is a custom framework which provides a communication interface between the computer, oscilloscope and the targeted device. During the implementation, the ABC synthesizer was queried until the AES core had a logic depth of 8, then the core was split-up in the middle to get two equally-sized parts. FFs are used in front of Part 1 as the multiplexer before the FFs introduces enough delay that no hold time violation occurs. Also, the multiplexer has no *active* signal, thus the reset signal to the FF is the only *active* signal leaving the clock tree in the corresponding switch matrix. Part 1 and Part 2 are divided by means of latches, i.e., the second approach explained in Section 3.2. Here 382 latches are placed in front of Part 2. In 222 cases a LUT, which is connected to the last *active* signal in Part 1, exists right before the latch. The latch gets its precharge value from this LUT. If the LUT that supplies the latch's input is not connected to the last *active* signal, a pass-through LUT is inserted. This pass-through LUT is connected to the last *active* signal to provide the precharge value to the latch (see Figure 1(b)). This is done in the 160 remaining cases ($222 + 160 = 382$). We further have used an MMCM and a Phase-Locked Loop (PLL) to generate the *active* signals.

### 5.1 Resource Utilization

To measure the resource overhead of our construction, we performed a normal Place and Route (PAR) without any modifications. We, in fact, replaced the synthesized AES from ABC with the unmodified Verilog sources. In addition, we implemented the same AES design under the concept of GliFreD to enable a fair comparison. Table 1 shows the resources used for the AES core in SafeDRP

**Table 1.** Resource consumptions, comparison between a normally PAR (Plain), SafeDRP, GliFreD, and the improved GliFreD AES designs.

| | SafeDRP | | Improved GliFreD [30] | | GliFreD [17] | | Plain |
|---|---|---|---|---|---|---|---|
| | Doubled | Single | Doubled | Single | Doubled | Single | |
| LUTs | 3712 | 1856 | 3466 | 1733 | 3466 | 1733 | 1262 |
| Register | 1276 | 638 | 11360 | 5680 | 22080 | 11040 | 256 |
| Slices | 1296 | 648 | 11638 | 5819 | 15502 | 7751 | 392 |
| Latency[a] | 11 | | 154 | | 308 | | 11 |
| Pipeline | 0 | | 14 | | 14 | | 0 |
| Throughput[b] | 116 | | 116 | | 58 | | 116 |

[a] clock cycles
[b] MBit/s @ 10 MHz

both before and after duplication, the same for a corresponding GliFreD variant, and the AES in plain. The resource consumption is doubled after the duplication process, as all cells are duplicated, and no further cells are added. Comparing the normally PAR design to the single-rail (SafeDRP/GliFreD) core, we get the overhead by the reduced LUT input pins.

An overall comparison to plain design reveals a factor of 2.94 more LUTs and 3.30 more slices, while the number of registers has an overhead of a factor of 4.98. The number of added registers ($636 - 256 = 382$) results from cutting the circuit into two parts. Compared to the improved GliFreD, the number of registers is drastically reduced, i.e., by a factor of $\frac{5680}{638} \approx 8.9$. In order to examine the overhead of the 5-LUT design, we need to subtract the 160 pass-through LUTs from the 1856 LUTs used in the single-rail design. Since these delayed LUTs are added in order to precharge the latches, they add no overhead for the 5-LUT design. Therefore, the overhead of the 5-LUT design compared to plain is $\frac{1856-160}{1262} = 1.343$, while the 5-LUT design can slightly reduce the number of LUTs compared to GliFreD (4-LUT) $(1856 - 160) = 1696 < 1733$. Including the pass-through LUTs, the number of LUTs is slightly increased by SafeDRP compared to GliFreD.

Table 1 also shows the throughput for the three considered designs. Since SafeDRP does not form any pipeline, its throughput – at the same frequency – is the same as the plain design, but it outperforms the first GliFreD variant due to the high number of pipeline stages of GliFreD. However, the GliFreD design can operate at an extremely higher frequency compared to SafeDRP, whose frequency is limited by the logic depth and the performance of the employed MMCM. Therefore, at maximum frequency the GliFreD design has a much higher throughput than its SafeDRP variant. Our design reaches up to 81 MHz, as the critical path in each stage is $\approx 1.1$ ns. In order to increase the frequency, one could insert pipeline stages as usual or enhance the placement process, as stated before.

### 5.2 Measurement Setup

We used a SAKURA-X evaluation board [1], equipped with a Kintex-7 XC7K160T FPGA, which hosts the investigated AES core(s). The power consumption is measured by a PicoScope 6402B at 1.25 GS/s. Between the measurement points on the SAKURA-X and the PicoScope, we placed two Mini-Circuits ZFL-1000LN+ AC amplifiers to amplify the signal. A trigger generated by the targeted FPGA ensures well-aligned traces.

We developed three different profiles of our design to examine the effectiveness of our construction, while we activate and deactivate the different protection mechanisms between the profiles. All modifications are done on the placed-and-routed design, which keeps the placement and routing untouched and hence allows a fair comparison.

- Profile 1 is the reference profile, in which the SafeDRP design is untouched.
- In Profile 2 the duplicated negative circuit is removed to test the Dual-Rail concept.
- In Profile 3 the duplicated negative circuit is removed, the precharge of the LUTs and FFs/latches are turned off and the LUTs are always active.

It should be noted that Profile 3 is a fully unprotected AES core. The only difference to a normal AES circuit is the unused first input pin of the LUTs (by constantly keeping the *active*=Hi) and the delay LUTs in front of the latches.

We run the AES core at a frequency of 10 MHz, thus one encryption requires 1.1 $\mu$s. An exemplary trace per profile are displayed in Figure 4, which cover the full 2,000 measured sample points. The encryption starts around point 230 and terminates around point 1670. The plaintext bytes for each encryption are randomly selected from a uniform distribution and the key is kept constant.

### 5.3 Side-Channel Analysis

We used different side-channel analysis methods in order to quantify leakage reduction caused by SafeDRP. We applied

- Signal-to-Noise ratio (SNR) [12] which examines how large the exploitable signal compared to the available noise is,
- Information-Theoretic (IT) metric [24] that gives an overview about the information available in the side-channel leakages with respect to the concept of information theory,
- Correlation Power Analysis (CPA) [5] with the Hamming Weight (HW) and Bit models to get an impression about the required number of traces of common key-recovery attacks, and
- Moments-Correlating DPA (MC-DPA) [16] attack to relax the necessity of a suitable power model in case of CPA, and examine the exploitable leakage through first-order leakages.
- Semi-fix vs. random Welch's t-test [23] which gives an overview about the existing detectable leakage.

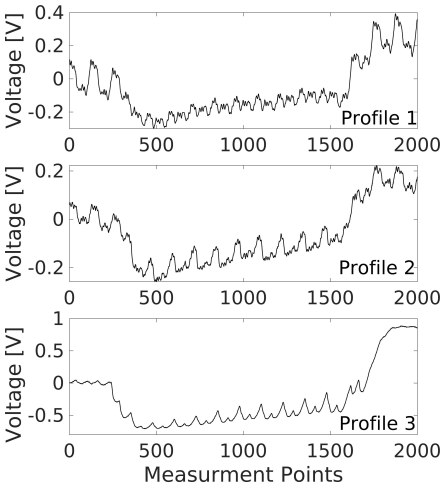



**Fig. 4.** Exemplary power traces.

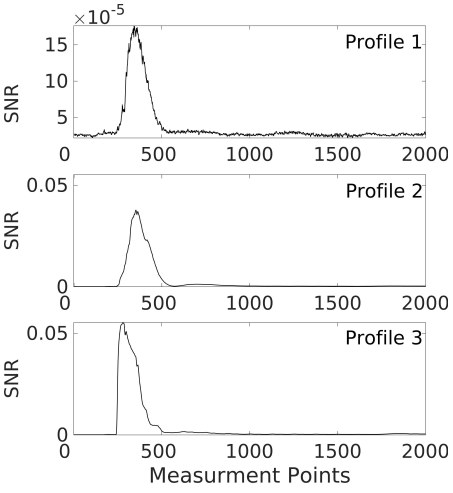



**Fig. 5.** SNR curves.

Power-equalization schemes are designed to reduce the SNR and make the attacks harder. Therefore, we limit our investigations to univariate first-order analyses. Thus, every method is conducted on each sample point separately. For each design, we measured $n = 10,000,000$ traces, and all attacks and analysis methods – except the t-test – make use of the entire measured traces while focusing on the first key byte.

We start with $SNR = \frac{var(signal)}{var(noise)}$. Following the procedure given in [12], we first categorize the traces by the value of the targeted plaintext byte and estimate the mean and variance for each group. Then the variance of the means states the $var(signal)$ since the average over each group represents mostly the noise-free signal depending on the underlying plaintext byte value. Further, the mean over the variance traces determines the $var(noise)$. Figure 5 shows the SNR curves based on the first plaintext byte of all profiles. No high non-sensitive peak is visible as the plaintext is applied to the circuit before the measurement and encryption start. Comparing Profile 2 and Profile 3, the precharge and evaluation of the LUTs and FFs/latches could only slightly reduce the SNR. However, when it is combined with DR, i.e., Profile 1, the SNR is reduced by a factor of $0.055/0.00018 \approx 313$.

By Information-Theoretic (IT) analysis [24] we can measures the amount of exploitable information by estimating the mutual information. Since our construction is a realization of hiding schemes and we limit our analyses to first order, in order to estimate the Mutual Information (MI) we can estimate the conditional entropy by means of Probability Density Functions (PDFs) based on Gaussian distributions. To this end, we can re-use the mean and variance traces estimated for the SNR. The resulting curves are given in Figure 6. The results are similar to the SNR; comparing Profile 1 and Profile 3, mutual information reduced by a factor of $\approx 265$.

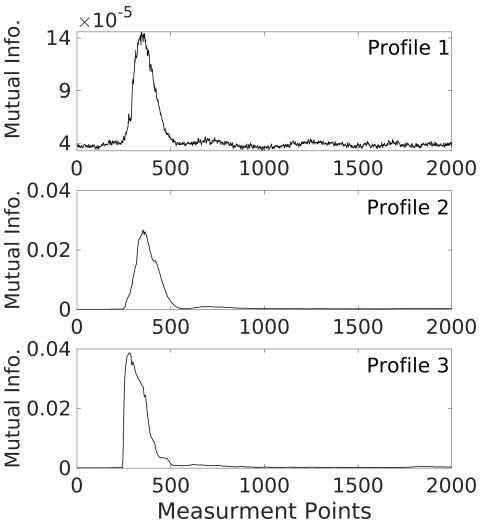



**Fig. 6.** Mutual information curves.

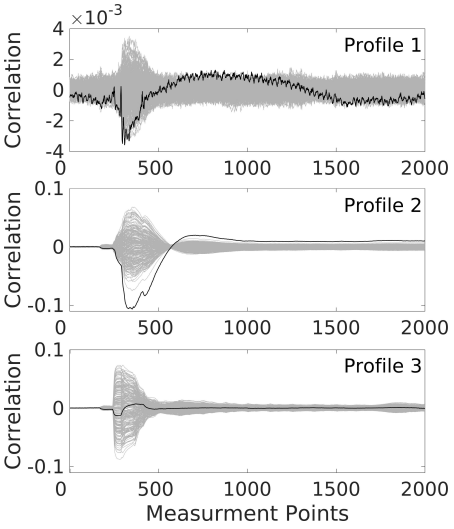



**Fig. 7.** CPA attack results, HW model.

We also conducted the commonly-used CPA attacks with different power models. We have first examined the attacks with HW of the S-Box output, which all led to unsuccessful key recovery. The reason is the underlying architecture of our case study, where the S-Box outputs are not stored in registers. Instead, the design has two FF stages (see Figure 3). The first FF stage contains the plaintext in the first round. After the first round, the output of the MixColumns is saved in these FFs. In order to predict the value of these FFs, at least 4 key bytes need to be guessed. Hence, we focus on the latches after Part 1, which are easier to predict.

We should emphasize that our design somehow merged the S-Boxes with their subsequent MixColumns. However, the latches are placed before the end of the S-Box calculations. It means that every latch bit depends on only one plaintext byte and one key byte. Further, for each state byte (plaintext XOR key) between 18 and 30 latches have been instantiated, which is mainly caused by the application of ABC's synthesis algorithms and the fact that ABC synthesizes the full AES round. Figure 7 depicts the result of the CPA attacks with the HW of the intermediate values, i.e., value of the latches. In the graphics, the curve for the correct key candidate is plotted in black while that of other candidates in gray. The attack is unsuccessful for Profile 3, since the latches in this Profile never become precharged. We did not change the power model to HD in Profile 3 since the distance is hard to predict in the first round. As well a comparison with an attack in the last round to the first round would not be fair. However, we can already observe the advantage of Profile 1 over Profile 2. More precisely, due to the quadratic inverse relation between the correlation and the required number of traces [12], i.e., $\rho^2 \propto \frac{1}{n}$, we can conclude that the attack on Profile 1 needs around $\left(\frac{0.1}{0.0035}\right)^2 \approx 816$ times more traces comapred to that on Profile 2.

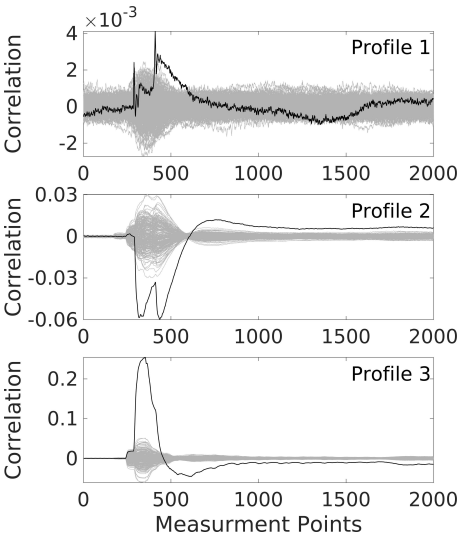



**Fig. 8.** CPA attack results, bit model.

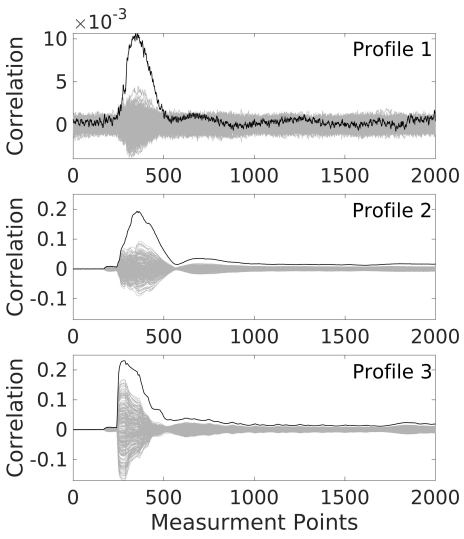



**Fig. 9.** MC-DPA results.

To conduct a successful attack on Profile 3, we considered the bit model as well, i.e., correlating the traces to a predicted certain latch bit. We have examined all latches independently; the result of the best attack (on Profile 3) is shown by Figure 8. In this case, the advantage of each profile compared to the others can be observed. For example, the effect of precharge concept, i.e., Profile 2 versus Profile 3, can be expressed by $\left(\frac{0.2544}{0.0599}\right)^2 \approx 18$ times more traces, and the effect of duplication (i.e., Profile 1 versus Profile 2) can be seen by $\left(\frac{0.0599}{0.0041}\right)^2 \approx 213$ more traces to exploit the leakage.

The feasibility of such CPA attacks strongly depends on the soundness of the underlying hypothetical power model, i.e., its linear relation to the actual leakage of the device. Therefore, we applied MC-DPA attack [16] as a sophisticated scheme that relaxes such a necessity and does not require any predefined power model. We used the profiled version of MC-DPA, where the first $n/2$ traces are used to generate the profiles, and the second $n/2$ traces for the attack. Figure 9 shows the correlation curves as the result of the attack on all profiles. The attack successfully recovers the correct key for all profiles. This is indeed expected because SafeDRP– as am power-equalization scheme – can only reduce the leakage, which results in a higher number of required traces for a successful attack. With respect to the concept of MC-DPA that can exploit any first-order leakage independent of the actual leakage function of the device, we can conclude that our construction Profile 1 succeeds in reducing the exploitable leakage with respect to the number of required traces. Comparing the results, the attack on Profile 1 needs $\left(\frac{0.195}{0.0106}\right)^2 \approx 338$ and $\left(\frac{0.230}{0.0106}\right)^2 \approx 470$ more traces compared to Profile 2 and Profile 3 respectively.

We also used the Welch's t-test [23] to quantify the leakage of our proposed scheme using $1,000,000$ traces. Following the same procedure in [29], we ap-

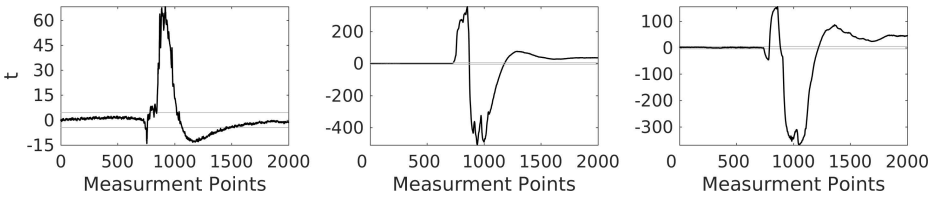



**Fig. 10.** Welch's semi-fix vs. random t-test using $1,000,000$ traces.

plied the semi-fix vs. random test to discard the leakage associated to plaintext/ciphertext. Prior to each measurement, a coin is flipped thereby selecting the plaintext from either the semi-fix or the random poll. The random plaintexts are selected from a uniform distribution, while the semi-fix plaintextes have been pre-computed in such a way that half of the cipher state (i.e., 64 bits) at the fifth round is filled by zero. Figure 10 shows the results of the t-test, where the reduction in leakage is visible again between the profiles. It is noteworthy that due to the memory effect of the employed amplifier [15], the detected leakage still appears after the fifth cipher round.

## 6 Conclusion

In this paper we introduced a novel equalization scheme SafeDRP and presented a general method to transform unprotected circuits into SafeDRP logic. By an AES implementation in SafeDRP as a case study we practically evaluated the effectiveness of SafeDRP to reduce the exploitable leakage resulting in hardening the key-recovery attacks.

In almost all duplication schemes, the power consumption of the positive and negative networks are still slightly different caused by process variations, different temperatures, aging of cells, and the chip internal supply voltage differences. Further, different times of evaluation are caused by e.g., the skew of the control signals[4]. Therefore, similar to the other power-equalization schemes, SafeDRP cannot completely avoid the leakage, but the practical results showed its success to extremely reduce such leakages. As a side note, for a complete practical protection such power-equalization techniques, e.g., SafeDRP, should be combined with proper masking schemes, e.g., the case shown in [17].

Apart from its high level of security, the advantage of SafeDRP over the known and similar schemes is its low overhead. Compared to the GliFreD scheme, SafeDRP reaches a slightly higher factor of reduced leakage. GliFreD reduces the SNR and MI by a factor of $\approx 100$ (SafeDRP $\approx 300$). The number of required traces and the factor for the CPA and MC-DPA attacks is mostly the same for both schemes. This leads to the following assumption: GliFreD's analysis is done on a 45 nm FPGA, while this case study is done on a 28 nm FPGA. Thus the smaller manufacturing process most likely causes the reduced leakage.

[4] Such features are already used for identification purposes [25] as well as randomness generation [26].

Nevertheless, the tested SafeDRP-based AES design uses fewer resources than a GliFreD-based design. With the cost of a more complicated control logic and 7% more LUTs SafeDRP requires only 11% of the FFs which are essential in corresponding GliFreD design.

Future research might be the adaptation of a different duplication strategy, as the separate true and false DRP cores could exploit some leakage in a localized EM attack.

## References


1. Side-channel AttacK User Reference Architecture. `http://satoh.cs.uec.ac.jp/SAKURA/index.html`.
2. Berkeley Logic Synthesis and Verification Group. ABC: A System for Sequential Synthesis and Verification, Release ae0be2deffef. `http://www.eecs.berkeley.edu/~alanmi/abc/`.
3. S. Bhasin, J. Danger, S. Guilley, and W. He. Exploiting FPGA Block Memories for Protected Cryptographic Implementations. *TRETS*, 8(3):16, 2015.
4. S. Bhasin, S. Guilley, F. Flament, N. Selmane, and J. Danger. Countering early evaluation: an approach towards robust dual-rail precharge logic. In *WESS 2010*, page 6. ACM, 2010.
5. E. Brier, C. Clavier, and F. Olivier. Correlation Power Analysis With a Leakage Model. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 16–29. Springer, 2004.
6. D. Canright. A Very Compact S-Box for AES. In *CHES 2005*, volume 3659 of *LNCS*, pages 441–455. Springer, 2005.
7. 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.
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 Computer Society, 2012.
10. J. Kaps and R. Velegalati. DPA Resistant AES on FPGA Using Partial DDL. In *FCCM 2010*, pages 273–280. IEEE Computer Society, 2010.
11. 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 2009*, pages 634–639. IEEE Computer Society, 2009.
12. S. Mangard, E. Oswald, and T. Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, 2007.
13. R. P. McEvoy, C. C. Murphy, W. P. Marnane, and M. Tunstall. Isolated WDDL: A Hiding Countermeasure for Differential Power Analysis on FPGAs. *TRETS*, 2(1):3:1–3:23, 2009.
14. A. Moradi and V. Immler. Early Propagation and Imbalanced Routing, How to Diminish in FPGAs. In *CHES 2014*, volume 5984 of *LNCS*, pages 598–615. Springer, 2014.

15. A. Moradi and O. Mischke. On the Simplicity of Converting Leakages from Multivariate to Univariate - (Case Study of a Glitch-Resistant Masking Scheme). In *CHES 2013*, volume 8086 of *LNCS*, pages 1–20. Springer, 2013.
16. A. Moradi and F.-X. Standaert. Moments-Correlating DPA. In *Workshop on Theory of Implementation Security*, TIS '16, pages 5–15. ACM, 2016.
17. A. Moradi and A. Wild. Assessment of Hiding the Higher-Order Leakages in Hardware - what are the achievements versus overheads? In *CHES 2015*, LNCS, pages 453–474. Springer, 2015.
18. M. Nassar, S. Bhasin, J. 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 Computer Society, 2010.
19. S. Nikova, V. Rijmen, and M. Schläffer. Secure Hardware Implementation of Nonlinear Functions in the Presence of Glitches. *J. Cryptology*, 24(2):292–321, 2011.
20. 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.
21. 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.
22. L. Sauvage, M. Nassar, S. Guilley, F. Flament, J. Danger, and Y. Mathieu. DPL on Stratix II FPGA: What to Expect? In *ReConFig 2009*, pages 243–248. IEEE Computer Society, 2009.
23. T. Schneider and A. Moradi. Leakage Assessment Methodology - A Clear Roadmap for Side-Channel Evaluations. In *CHES 2015*, volume 9293 of *LNCS*, pages 495–513. Springer, 2015.
24. F. 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.
25. G. E. Suh and S. Devadas. Physical Unclonable Functions for Device Authentication and Secret Key Generation. In *Proceedings of the 44th Design Automation Conference - DAC 2007, San Diego, CA, USA, June 4-8, 2007*, pages 9–14. IEEE Computer Society, 2007.
26. B. Sunar, W. J. Martin, and D. R. Stinson. A Provably Secure True Random Number Generator with Built-In Tolerance to Active Attacks. *IEEE Trans. Computers*, 56(1):109–119, 2007.
27. 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 *ESSCIRC 2002*, pages 403–406, 2002.
28. 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.
29. A. Wild, A. Moradi, and T. Güneysu. Evaluating the Duplication of Dual-Rail Precharge Logics on FPGAs. In *COSADE 2015*, volume 9064 of *LNCS*, pages 81–94. Springer, 2015.
30. A. Wild, A. Moradi, and T. Guneysu. GliFreD: Glitch-Free Duplication – Towards Power-Equalized Circuits on FPGAs. *IEEE Transactions on Computers*, 2017.
31. Xilinx. *UG472 7 Series FPGAs Clocking Resources User Guide*, June 2015.
32. P. Yu and P. Schaumont. Secure FPGA circuits using controlled placement and routing. In *CODES+ISSS 2007*, pages 45–50, 2007.