Evaluating the Duplication of Dual-Rail Precharge Logics on FPGAs

Alexander Wild(✉), Amir Moradi, and Tim Güneysu

Horst Gortz Institute for IT-Security, Ruhr-Universitat Bochum, Bochum, Germany
{alexander.wild,amir.moradi,tim.gueneysu}@rub.de

Abstract. Power-equalization schemes for digital circuits aim to harden cryptographic designs against power analysis attacks. With respect to dual-rail logics most of these schemes have originally been designed for ASIC platforms, but much efforts have been spent to map them to FPGAs as well. A particular challenge is here to apply those schemes to the predefined logic structures of FPGAs (i.e., slices, LUTs, FFs, and routing switch boxes) for which special tools are required. Due to the absence of such routing tools Yu and Schaumont presented the idea of duplicating (i.e., dualizing) a fully-placed-and-routed dual-rail precharge circuit with equivalent routing structures on an FPGA. They adopted such architecture from WDDL providing the Double WDDL (DWDDL) scheme.

In this work we show that this general technique – regardless of the underlying dual-rail logic – is incapable to properly prevent side-channel leakages. Besides theoretical investigations on this issue we present practical evaluations on a Spartan-6 FPGA to demonstrate the flaws in such an approach. In detail, we consider an AES-128 encryption module realized by three dual-rail precharge logic styles as a case study and show that none of those schemes can provide the desired level of protection.

1 Introduction

Side-Channel Analysis (SCA) is of major challenges for secure-hardware designers. The most popular techniques to harden a design are hiding, masking and leakage resilient architectures. The goal of power equalization schemes, which are a part of the hiding category, is to equalize the power consumption of a cryptographic circuit independent of the processed data. In hardware devices such schemes often follow the Dual-rail Precharge Logic (DPL) concept like SABL [21], WDDL [22], DRSL [6], MDPL [18] and iMDPL [17]. Most of the dual-rail schemes have been developed to be implemented in ASICs. The fixed architecture as well as limited wire routings on FPGAs does not allow a straightforward porting of those schemes.

1.1 Related Work

During the last years, much effort was put in the direction of bringing the DPL concept to FPGAs. Nassar et al. [16] introduced Balanced Cell-based Dual-rail

S. Mangard and A.Y. Poschmann (Eds.): COSADE 2015, LNCS 9064, pp. 81–94, 2015.
DOI: 10.1007/978-3-319-21476-4_6

Logic (BCDL) that connects a global precharge signal with all gate inputs to a rendezvous box which is placed in front of each gate. The rendezvous box triggers an evaluation signal for the corresponding gate in case of stable input signals. This work did not consider the routing of the *true* and *false* networks. So the wire capacities of their dual-rail networks will be different. He et al. followed an approximately similar approach [8]. In their work the evaluation of a gate is triggered by two global signals that are directly connected to the gates. Lomné et al. [11] followed a triple-rail approach where an artificially delayed asynchronous control signal is used. The additional delay guarantees the latest arriving of the control signal which triggers the gate evaluation and therefore prevents the gate from early propagation (EP).

In a more recent work [9] He et al. applied duplication while minimizing the area overhead. They duplicated a fully placed and routed circuit to realize the dualization. The original and duplicated circuit can be interleaved, which may leads to routing conflicts. Hence, a method has been developed to detect and correct these routing conflicts. As a result, the circuits are differently routed at some points which lead to side-channel information leakage.

In [19], Sauvage et al. evaluated different placement strategies to support the router in finding routes for the dual networks with minimal delay differences. The strategies they followed were first, placing the components of the original circuit as close as possible together as well as the components of the dual circuit and second, placing the related components of the original and dual circuit as close as possible together. It turned out that the placement did not have the desired impact on the routing.

Bhasin et al. introduced in their work [3] an improved version of WDDL called DPL-noEE. Since it has been shown that WDDL suffers from the early propagation effect [20], DPL-noEE connects the signals of the true and false network to the same gate and evaluates the gate output with the arrival of the last incoming signal. According to the authors' report, compared to WDDL the leakage is approximately halved by means of DPL-noEE. It has been pointed out in [15] that DPL-noEE just solves the early propagation effect in the evaluation phase of the circuit but not in the precharge phase. They introduced a logic style called AWDDL that solved this problem. An AWDDL gate switches to precharge state when the last input signal turns to precharge. Additionally, a customized router was developed that tries to find routes with minimal delay differences for the true and false network by moving the routing process into a Satisfiability (SAT) solvable problem. It is also noted by the authors that the router could minimize the leakage of the circuit but did not completely remove it due to the nonexistence of perfectly-identical dual-rail routes.

To deal with routing imbalances, Yu and Schaumont had formerly proposed to duplicate a fully-placed-and-routed WDDL circuit [24] (known as DWDDL). As a result, two equivalently routed WDDL circuits with swapped true and false networks (dualized) were placed on the same device. With investment of doubled resources, the leakage of a WDDL circuit was drastically reduced.

1.2 Motivation and Contribution

Power-equalization schemes place a circuit C and the dual of the circuit $\overline{C}$ on the same device. Ideally, the total power consumption of both circuits cancel out the data dependency in the power traces. This idea implies that the logic gates of C and $\overline{C}$ have to be synchronized and switch at the same point in time. This synchronization can be achieved by a global or asynchronous control signal connected to the gates like in [8,11,16] or with the arrival of the input signals at the gate like in [3,15,22]. To use the input signals as a trigger for the evaluation, the signals of the dualized circuit $\overline{C}$ must show exactly the same delays as their pendant signals in C. Beside the synchronization aspect, the signal delays are strongly correlated with the capacity of the used wires so that even in a control signal synchronized circuit the signal delays of both circuits shall be equivalent to minimize the data-dependent side-channel leakage. The task of placing two circuits with the same signal delays is hard to achieve in FPGAs due to the static routing structure. Some logic styles, e.g., the most popular one WDDL, require the interconnection of both C and $\overline{C}$ circuits. In such cases the task to place and route the logic gates in such a way that the routing delays of coupled signals are equivalent is challenging. In [15] this task was addressed with a custom router based on a SAT solver.

As stated, in DWDDL [24] the fully-routed-and-interconnected original circuit $(C, \overline{C})$ is additionally placed with inverted logic on the same device $(\overline{C'}, C')$. The cloning process was performed in a way that the routing information is transferred to the cloned circuit. According to the report of the Xilinx design tools, the signal delays of the original circuits $(C, \overline{C})$ and the cloned circuits $(\overline{C'}, C')$ are equivalent. So this method turned out to be the best way to implement two circuits of equivalent routing delays without any routing restrictions or any custom routers. The drawback of the technique is clearly the high resource overhead.

In this work we thoroughly investigate the duplication scheme of [24]. Since the early propagation issue of WDDL makes it still vulnerable to the state-of-the-art attacks, we consider its successors DPL-noEE and AWDDL as well to examine the benefit of the duplication. We show that even in case of AWDDL, where early propagation at both phases is avoided, applying the duplication does not prevent data-dependent time of evaluation. By means of a Spartan-6 evaluation platform (SAKURA-G [1]) we provide practical evidences to our findings that a DWDDL circuit and the equivalent ones realized by DPL-noEE as well as AWDDL still have leakage.

2 Logic Styles

Each of WDDL, DPL-noEE and AWDDL consists of gates with two outputs, O_t and O_f. In the precharge phase both outputs have the same value ($O_t = 0, O_f = 0$), while in the evaluation phase only one output changes its state so that exactly one transition per evaluation phase is guaranteed. O_t presents the true value while O_f the false pendant. The true outputs form a network we address as

true network (respectively *false network* made by the false outputs). In case of a negative gate, e.g., NAND or NOR, the corresponding non-negative gate (resp. AND or OR) is instantiated with switched output signals. Clearly, an inverter gate is realized by a connection switch swapping the dual rails. Hence such logic styles form two logical circuits that are interconnected. We further refer to this dual-rail circuit as the original circuit $(C, \overline{C})$. Following the duplication scheme of [24], we clone the original circuit, invert the logic and place it at a different location on the FPGA. The circuit made by this process is denoted as the duplicated circuit $(\overline{C'}, C')$. Below we shortly recall the specification of each of our considered logic styles.

WDDL is one of the most common DPL styles and mainly designed for ASICs. In WDDL only AND/NAND and OR/NOR gates are allowed. An XOR/XNOR gate is constructed by two AND/NAND and one OR/NOR WDDL gates. As stated, our evaluation platform is a SAKRURA-G where a Xilinx Spartan-6 FPGA is plugged that is equipped with 6-to-2 LUTs. This gives the advantage to realize each WDDL gate by one LUT. The building blocks of WDDL – with respect to 6-to-2 LUTs – can be seen in Fig. 1(a).

DPL-noEE is unofficially the successor of WDDL. As stated in [20] and in [3], a WDDL gate evaluates its output at different points in time depending on the input data. For example, O_t of a WDDL OR gate is derived from two signals of the true network. Regardless of the other input, O_t goes high once one of its inputs of the true network goes high. This phenomena is known as early propagation effect and causes data-dependent power consumption and hence side-channel leakage.

This issue is addressed by the DPL-noEE logic style. As shown in Fig. 1(b), O_t of a DPL-noEE OR gate depends on all four input signals, so that the gate evaluates when all signals are available. Due to the consideration of the true and false signals in each gate – contrary to WDDL – a DPL-noEE XOR/XNOR gate can be constructed by a single LUT. This results in less LUT utilization particularly in designs with high number of XOR gates.

The authors of [15] noted in their work that DPL-noEE prevents the early propagation at the beginning of the evaluation phase. It is shown that the output of DPL-noEE gates fall back into precharge as soon as one of the input signals changes; hence the propagation in the precharge phase is faster. Although such an issue is not data dependent, it has been shown in [15] that the amount of power consumption (resp. amount of side-channel leakage) at the precharge phase is higher than that of the evaluation phase.

AWDDL emulates an SR-Latch inside each gate by looping the gate output to its input. Hence an AWDDL gate does not change its state until all input signals are in the evaluation or all in the precharge phase. Figure 1(c) shows the internal structure of AWDDL gates. In comparison to WDDL and DPL-noEE,

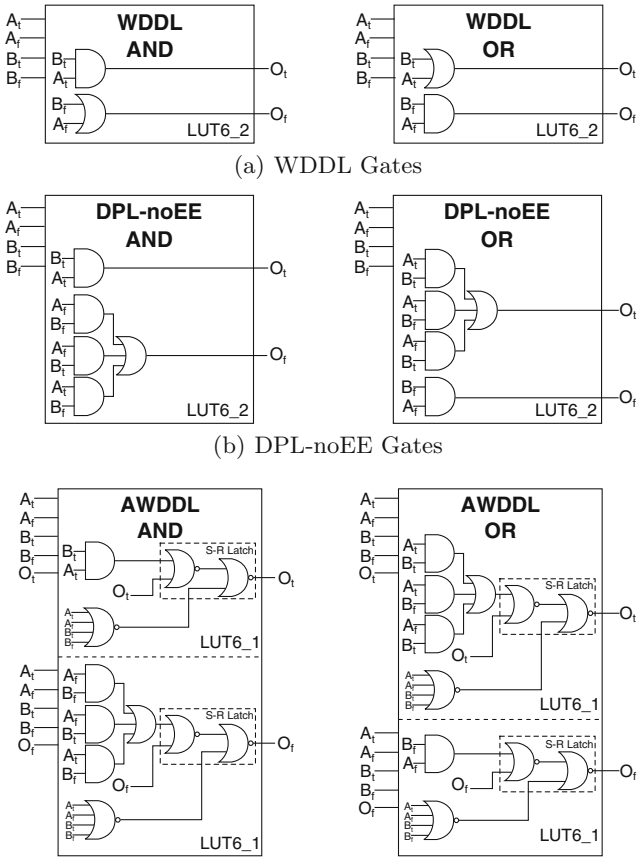


Fig. 1. Logic style gates

one AWDDL gate should be realized by two 6-to-1 LUTs. In fact, a true 6-to-2 LUT would suffice to make both outputs of a gate. However, as shown by Fig. 2 the 6-to-2 LUT available in Xilinx FPGAs are made of two multiplexed 5-to-1 LUTs [23].

The necessity of employing two 6-to-1 LUTs per AWDDL gate as well as the gate loop paths increase the resource utilization and routing complexity compared to DPL-noEE. The authors of [15] have proposed to place both LUTs of each gate at the same slice to minimize the delay difference between the true and false outputs. Further, a customized router have been developed which tries to find equivalent routes for the true and false signals. The authors reported that the router improves the result but does not fully avoid the leakage associated to the signal delay differences. Table 1 summarizes the properties of the aforementioned logic styles.

Table 1. Properties of the in this work underlying logic style.

Logic style	Native gates	LUT/Gate	Address routing	EP
WDDL	AND, OR	1		in Eval. &Prech
DPL-noEE	AND, OR, XOR	1		in Prech
AWDDL	AND, OR, XOR	2	(✓)	-

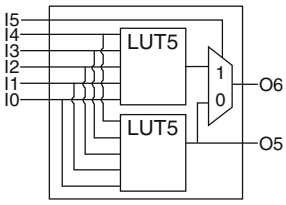


Fig. 2. 6-to-2 LUT

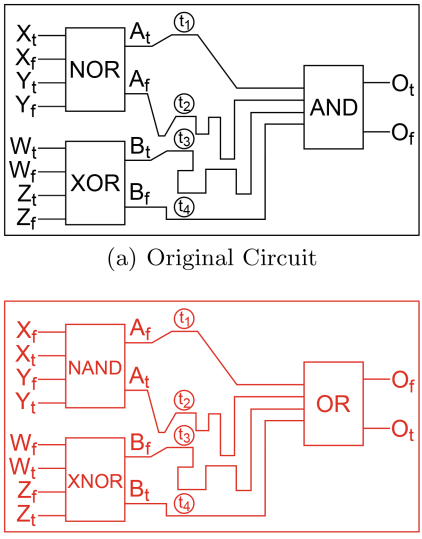


Fig. 3. Example of a fully routed circuit and its duplication.

3 Duplicating Circuits

This section deals with the idea proposed in [24] to duplicate a dual-rail circuit. Figure 3 shows an overview of an exemplary circuit and its duplicated counterpart. Due to the copied structure the signal routings and corresponding delays $t_1 \dots t_4$ are transferred from the original to the duplicated circuit and just the logic gates are replaced. Attended by the gate replacement, the true and false networks are swapped in the duplicated circuit. Thus, the true network of the original circuit and the false network of the duplicated circuit (respectively the false network of the original circuit and the true network of the duplicated circuit) are equivalently routed that hence results in an overall design with very balanced true and false networks to minimize the leakage caused by different wire capacitances.

As stated before, DPL-noEE and AWDDL avoid the early propagation issue in contrast to WDDL. Along the same lines, the term *data-dependent time of*

evaluation is defined as the cases when the gate evaluates its output at different time instances depending on its input values [12]. The duplication concept proposed in [24] (DWDDL) aims at mainly avoiding such a data-dependent time of evaluation caused by difference in routing of dual-rail signals. In the following we show that such a scheme cannot avoid data-dependent time of evaluation even if the underlying logic style prevents the early propagation.

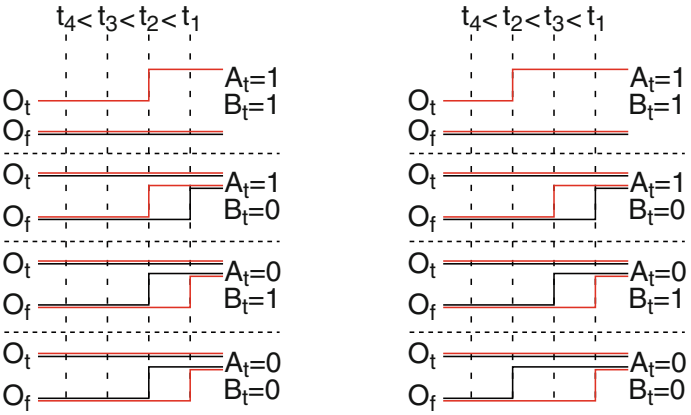
3.1 Data-Dependent Time of Evaluation

In order to explain the concept we focus on the example given in Fig. 3. We just consider the start of the evaluation phase of an AND gate in both original and duplicated circuit (O_t and O_f in Fig. 3). Further, due to the early propagation issue of WDDL, we suppose that the gates in these circuits are realized by DPL-noEE or AWDDL (there is no difference in this example since both avoid early evaluation).

As marked in Fig. 3, we denote the delay of the input signals of the considered AND gate as t_1 to t_4, which stay the same for the corresponding OR gate in the duplicated circuit. We should highlight that no customized router is employed to route the signals in the original circuit. Indeed, the idea of duplication [24] is to avoid such a necessity. Therefore, the signal delays t_1 to t_4 can have any arbitrary value, and there is no guarantee to keep them the same or even approximately the same.

We first suppose that $t_1 > t_2 > t_3 > t_4$, and draw the timing diagram of the output signals (O_t and O_f) of both original and duplicated circuits for all four possible input values, e.g., 11, 10, 01, and 00. The corresponding timing diagram is shown by Fig. 4(a), where the black waveform belongs to the AND gate of original circuit (Fig. 3(a)) and the red waveform to its complementary OR gate of the duplicated circuit (Fig. 3(b)). Under such a condition the AND gate evaluates when the last input signal arrives. In case of 11 and 10 the output is evaluated at t_1 (when A_t arrives). For other input cases 01 and 00 t_2 defines the evaluation time (when A_f arrives). The duplicated OR gate shows a complementary behavior for the time of evaluation. That is, independent of the input value one gate always evaluates at t_1 and the other one at t_2. Hence, the overall power consumption is then ideally independent of the given input value. It should be noted that exchanging the values of t_1 by t_2 and/or t_3 by t_4 as well as (t_1, t_2) by (t_3, t_4) (i.e., providing another condition e.g., $t_3 > t_4 > t_2 > t_1$) has no effect on the shown balanced behavior.

Figure 4(b) gives the waveforms under a different condition $t_1 > t_3 > t_2 > t_4$. For the given condition the gates evaluate the outputs at t_1, t_2 or t_3. In case of 11 and 00, t_1 and t_2 defines the evaluation time, while for 01 and 10 the evaluation time depends on t_1 and t_3. In other words, one gate either in the original circuit or in the duplicated one evaluates at t_1, but the other gate evaluates at t_2 or at t_3 depending on the input value. This clearly shows a data-dependent time of evaluation and should lead to a leakage exploitable by an attack. Again we should note that exchanging t_1 by t_3 and/or t_2 by t_4 as well as (t_1, t_3) by (t_2, t_4) does not show any difference on the presented data-dependent time of evaluation.



(a) under the condition $t_1 > t_2 > t_3 > t_4$

(b) under the condition $t_1 > t_3 > t_2 > t_4$

Fig. 4. Timing diagrams of the circuit of Fig. 3.

Remarks: In case of DWDDL the situation is much worse. That is, for every condition (except $t_1 = t_3$, $t_2 = t_4$) there is a data-dependent time of evaluation due to the early propagation of WDDL. The same issue holds true at the start of the precharge phase for all three considered logic styles. As a result, since in the certain conditions the duplication can be beneficial, we expect the duplication scheme to reduce but not fully avoid the leakage.

3.2 Duplication Tool

As our target platform is a Spartan-6 FPGA, we developed a tool to realize the duplication procedure. The tool clones the structure of the original circuit (in Xilinx Spartan-6 netlist format) and changes the LUTs content. It is indeed based on the same concept as the one introduced in [24]. Below a detailed description of the developed tool is given.

Xilinx netlists are stored in a proprietary file format (NCD). For low-level access Xilinx provides a tool to convert the proprietary netlist file format into (and back from) a human-readable format called Xilinx Design Language (XDL). This tool offers the ability to access and manipulate a fully-routed circuit on netlist level. Note that the XDL structure may change between FPGA families caused by technological progress in FPGA structure. The duplication tool was written with the help of RapidSmith [10] which is a Java-based Application Programming Interface (API) to read, write and edit XDL files. The reader interested in detailed information of the XDL file format is referred to [2].

The XDL file format is organized in instances and nets. An instance is an instantiated component on the device, e.g., a slice. The configuration of an instance is given by its attributes. These attributes also contain the LUT content. By changing the LUT content of an instance, the logic gates of a circuit can easily be changed. In case of the circuit duplication, an AND gate of the

original circuit needs to be converted to an OR gate in the duplicated circuit, a NAND to a NOR and an XOR to an XNOR gate and vice versa. The physical location of an instance is defined in the instance header. Components of the Xilinx FPGAs are organized in a grid and can be identified by their X and Y coordinates. Hence, cloning a slice instance of the original circuit with manipulated LUT and placement coordinates adds a complementary-behaving instance to the design.

The routing of the signals is organized in a quite similar manner. The nets are routed via switch boxes that are able to interconnect different wires and are identifiable via their X and Y coordinates. A switch matrix configuration is called Programmable Interconnect Point (PIP). Hence, all PIPs used to route a net are written to the configuration part of that net. Copying the net with the edited PIP coordinates adds a net which is equivalently routed to a design. So, this process is performed for all nets of the original circuit. As an exception, the circuit I/O signals have to be handled differently. These nets cannot and are not needed to be duplicated. It is assumed, that the I/O signals (related to e.g., plaintext and ciphertext) do not leak any information. To route the I/O signals of the duplicated circuit, the standard Xilinx ISE routing tool cannot be used. In some cases it changes the already-routed nets, which may destroy the symmetry between the original and duplicated circuits. To route the I/O nets, the FPGA Editor which is also a part of the Xilinx ISE design suite offers the possibility to only route the specified signals. It is an adequate scenario for the low number of I/O signals.

To guarantee that the physical area, at which the cloned circuit should be placed, is unused by the remaining design, the Xilinx ISE synthesizer can be parametrized with the *prohibit* constraint to avoid placing any instance in that specified area. Another constraint we used is *area_group* to keep the original circuit in a specified area as well. We also made sure that no routing resources of the prohibited area is used by other logics.

4 Analysis

We implemented three AES-128 encryption cores using WDDL, DPL-noEE, and AWDDL gates. Following the scenario explained in Sect. 3.2, each of these cores is duplicated to realize the DWDDL, DDPL-noEE and DAWDDL AES cores. Hence, in total we analyze six full AES encryption cores. In general, for a fair comparison the best would be to keep the placement and routing of all these cores the same. However, WDDL has no defined native XOR gate; so a DPL-noEE circuit cannot be converted to a corresponding WDDL one. Also, AWDDL requires feedback loops as well as two LUTs per gate, hence converting an AWDDL circuit to a corresponding DPL-noEE one would be not fair with respect to resource utilization. Therefore, an identical placement and routing for these three logic styles is not possible.

For the AES core we implemented a round-based architecture with a composite field Sbox of [5] (see Fig. 5). Due to the known issue of register cells in dual-rail logics [14], we followed the master-slave fashion for the registers,

i.e., two register stages in each rail. Therefore, every cipher round is operated in two clock cycles: one for the precharge and the other one for the evaluation. Indeed, in an interleaved fashion one of the round register stages holds the AES state and the other one the precharge 0 value.

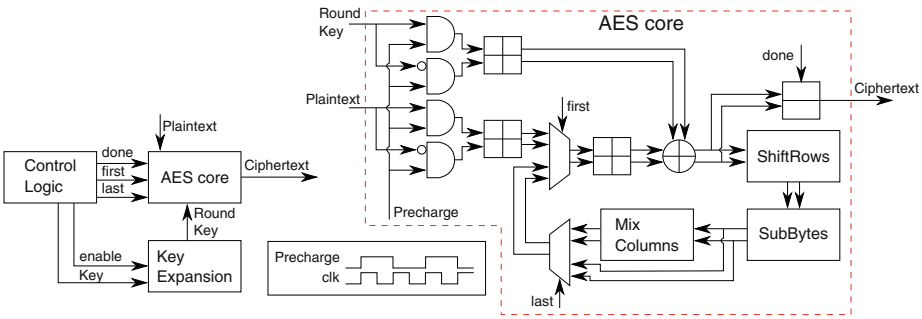


Fig. 5. Architecture of the full AES encryption with key expansion and control logic.

As we do not target any template attacks and ignore the leakage solely associated to the process on the key, the key expansion unit is implemented normally without using any secure logic style. As shown in Fig. 5, plaintext and round key (the output of the key expansion) are converted to a dual-rail precharge form and stored in dual-rail master-slave registers. For the sake of simplicity, we have not shown the key expansion and the control unit, which we kept equal regardless of the used logic style. The area which is marked (by a red dashed line) in Fig. 5 is the only part of the circuit that is implemented by WDDL, DPL-noEE, or AWDDL. Further, only this area is duplicated to realize DWDDL, DDPL-noEE, and DAWDDL circuits. The resource consumption of each core can be seen in Table 2.

Table 2. Overview of the implemented AES cores.

Logic style	Utilized resources		
	Slice	LUT	FF
WDDL	3,214	8,154	1,672
DWDDL	6,428	16,308	3,344
DPL-noEE	3,712	3,834	1,672
DDPL-noEE	7,424	7,668	3,344
AWDDL	3,724	7,146	1,672
DAWDDL	7,448	14,292	3,344
Control Logic	15	30	20
Key Expansion	83	307	132

4.1 Side-Channel Evaluation

For the practical evaluations we used SAKURA-G [1] as a side-channel evaluation board which is equipped with a Xilinx Spartan-6 FPGA. The power traces have been collected by monitoring the voltage drop over a 1 Ω resistor at Vdd path. The target FPGA operated at a frequency of 3 MHz; power traces have been obtained by means of a digital oscilloscope at a sampling rate of 1 GS/s. We also made use of the amplifier embedded on the SAKURA-G board and limited the acquisition bandwidth to 20 MHz to achieve high quality signals.

As the evaluation metric we used the leakage assessment methodology (t-test) presented in [7]. In a *non-specific* t-test (known also as *fix vs. random* test) a fix input (here plaintext) is selected, and during the measurements the fix or a random input is given to the target device[1]. The traces are categorized into two groups based on the associated fix or random input. Then, the Welch's (two-tailed) t-test is computed based on the mean and variance of each group of the traces (at each sample point independently). The outcome gives a level of confidence to reject a hypothesis as the traces of these two groups are drawn from the same population. If so (i.e., $|t| > 4.5$), it can be confidently concluded that a first-order leakage can be exploited from the device under evaluation.

Such a fix vs. random test is useful particularly to evaluate masked implementations. For instance, the same scheme has been used to examine the leakage of a higher-order attack resistant implementation of KATAN block cipher in [4]. However, in case of our designs where no masking is involved we cannot easily apply such a test. That is because the plaintext, which is not masked, is sent during the communication (between the FPGAs of the SAKURA-G) and processed by the target FPGA. Therefore, regardless of the protection that the AES core provides the leakage associated to the plaintext is observable at every sample point of the traces[2].

As a solution, a *semi fix* vs. random test can be performed. In such a scenario, a set of plaintexts are selected, all of which lead to the partially same intermediate value. We have precomputed 1024 plaintexts, in such a way that the first 64 bits of the cipher state at the start of the round 5 of the AES encryption are 0 (a similar scheme has been introduced in [7]). During the measurements a random plaintext or one of the precomputed plaintexts (again randomly) is taken. The rest of the evaluation stays the same as that of a fix vs. random test.

A sample trace of the WDDL design is shown at the top of Fig. 6. The first high peaks are related to the conversion and synchronization of the plaintext and the roundkey to the dual-rail precharge form (see Fig. 5). The trace of the other designs – particularly the duplicated ones – look like the same but with higher peak-to-peak value. For each of the six designs we collected 1 000 000 traces following the scenario explained above. It means that approximately 500 000 traces with random plaintext and the rest with a randomly-chosen plaintext amongst the 1024 precomputed ones. The result of the tests on all six designs are shown by Fig. 6.

[1] Note that such a selection should also be randomized.

[2] One reason is also related to the static leakage [13].

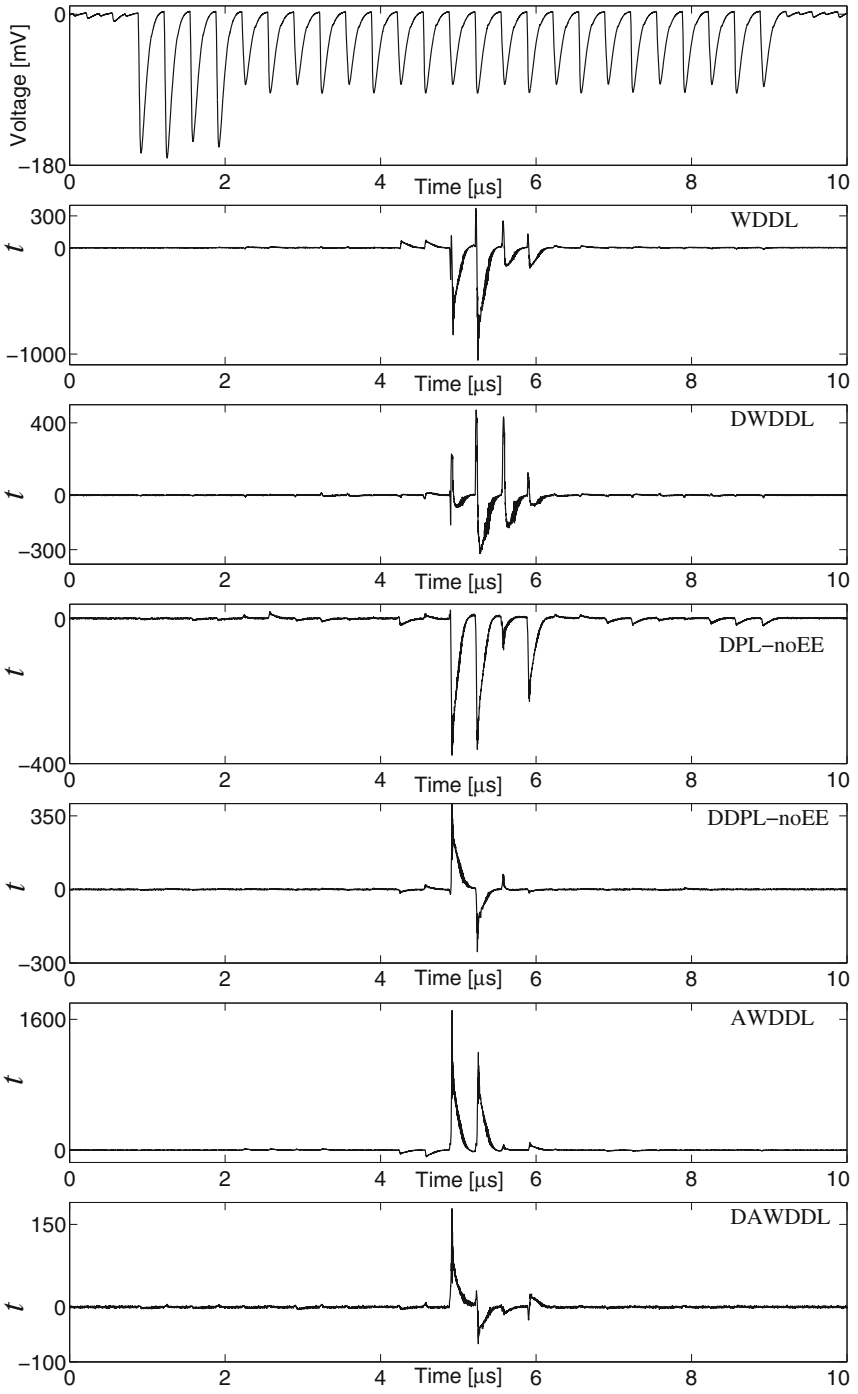


Fig. 6. A sample trace and the t-test results using 1 000 000 traces.

It can be seen that none of the designs is able to avoid the leakage, and the tests strongly confirm the existence of a leakage. However, as we expected – stated in Sect. 3.1 – the duplicated designs can reduce the leakage, but due to the flaw (i.e., data-dependent time of evaluation for certain conditions) it cannot be prevented. It is noteworthy that the evaluation scheme which we performed is a "leakage assessment methodology" on one of the middle rounds of the cipher. Although we have not performed any key-recovery attack, and have not provided any information about the simplicity/hardness of such an attack, the result of the presented tests indicate the failure of the underlying design methodologies to prevent the leakages.

5 Conclusion

Regardless of its significant area overhead, duplicating a dual-rail precharge circuit (DWDDL) was considered as the only scheme that can be used for power equalization on FPGAs. In this work we have shown that this scheme is not flawless. By an extensive evaluation we found situations where the time of evaluation (of the gates in a double dual-rail precharge circuit) still depends on the input values. Such a data-dependent time of evaluation is caused by the difference between the signal delay of the gate inputs that cannot be avoided. Our theory is supported by practical analysis that we conducted on a Xilinx Spartan-6 FPGA. Although DPL-noEE and AWDDL avoid the well-known early propagation issue of WDDL, we have shown that still none of them can be considered as a potential solution to prevent the side-channel leakage.

Acknowledgment. This work was partially funded by the European Horizon 2020 project SAFEcrypto (grant no. 644729), German Research Foundation (DFG), and DFG Research Training Group GRK 1817/1.

References

1. Side-channel AttacK User Reference Architecture. http://satoh.cs.uec.ac.jp/SAKURA/index.html
2. Beckhoff, C., Koch, D., Tørresen, J.: The Xilinx Design Language (XDL): tutorial and use cases. In: ReCoSoC 2011, pp. 1–8. IEEE (2011)
3. Bhasin, S., Guilley, S., Flament, F., Selmane, N., Danger, J.: Countering early evaluation: an approach towards robust dual-rail precharge logic. In: WESS 2010, pp. 6. ACM (2010)
4. Bilgin, B., Gierlichs, B., Nikova, S., Nikov, V., Rijmen, V.: Higher-order threshold implementations. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part II. LNCS, vol. 8874, pp. 326–343. Springer, Heidelberg (2014)
5. Canright, D.: A very compact s-box for AES. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 441–455. Springer, Heidelberg (2005)
6. Chen, Z., Zhou, Y.: Dual-rail random switching logic: a countermeasure to reduce side channel leakage. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 242–254. Springer, Heidelberg (2006)

7. Goodwill, G., Jun, B., Jaffe, J., Rohatgi, P.: A testing methodology for side-channel resistance validation. In: NIST Non-Invasive Attack Testing Workshop (2011)
8. He, W., de la Torre, E., Riesgo, T.: A precharge-absorbed DPL logic for reducing early propagation effects on FPGA implementations. In: ReConFig 2011, pp. 217–222. IEEE Computer Society (2011)
9. He, W., Otero, A., de la Torre, E., Riesgo, T.: Automatic generation of identical routing pairs for FPGA implemented DPL logic. In: ReConFig 2012, pp. 1–6. IEEE Computer Society (2012)
10. Lavin, C., Padilla, M., Lamprecht, J., Lundrigan, P., Nelson, B., Hutchings, B., Wirthlin, M.: RapidSmith - A Library for Low-level Manipulation of Partially Placed-and-Routed FPGA Designs. Technical report, Brigham Young University, September 2012
11. Lomné, V., Maurine, P., Torres, L., Robert, M., Soares, R., Calazans, N.: Evaluation on FPGA of triple rail logic robustness against DPA and DEMA. In: DATE 2009, pp. 634–639. IEEE Computer Society (2009)
12. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards. Springer, Heidelberg (2007)
13. Moradi, A.: Side-channel leakage through static power. In: Batina, L., Robshaw, M. (eds.) CHES 2014. LNCS, vol. 8731, pp. 562–579. Springer, Heidelberg (2014)
14. Moradi, A., Eisenbarth, T., Poschmann, A., Paar, C.: Power analysis of single-rail storage elements as used in MDPL. In: Lee, D., Hong, S. (eds.) ICISC 2009. LNCS, vol. 5984, pp. 146–160. Springer, Heidelberg (2010)
15. Moradi, A., Immler, V.: Early propagation and imbalanced routing, how to diminish in FPGAs. In: Batina, L., Robshaw, M. (eds.) CHES 2014. LNCS, vol. 8731, pp. 598–615. Springer, Heidelberg (2014)
16. Nassar, M., Bhasin, S., Danger, J., Duc, G., Guilley, S.: BCDL: a high speed balanced DPL for FPGA with global precharge and no early evaluation. In: DATE 2010, pp. 849–854. IEEE Computer Society (2010)
17. Popp, T., Kirschbaum, M., Zefferer, T., Mangard, S.: Evaluation of the masked logic style MDPL on a prototype chip. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 81–94. Springer, Heidelberg (2007)
18. Popp, T., Mangard, S.: Masked dual-rail pre-charge logic: DPA-resistance without routing constraints. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 172–186. Springer, Heidelberg (2005)
19. Sauvage, L., Nassar, M., Guilley, S., Flament, F., Danger, J., Mathieu, Y.: DPL on stratix II FPGA: what to expect?. In: ReConFig 2009, pp. 243–248. IEEE Computer Society (2009)
20. Suzuki, D., Saeki, M.: Security evaluation of DPA countermeasures using dual-rail pre-charge logic style. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 255–269. Springer, Heidelberg (2006)
21. Tiri, K., Akmal, M., Verbauwhede, I.: A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards. In: ESSCIRC 2002, pp. 403–406 (2002)
22. Tiri, K., Verbauwhede, I.: A Logic level design methodology for a secure DPA resistant ASIC or FPGA implementation. In: DATE 2004, pp. 246–251. IEEE Computer Society (2004)
23. Xilinx: Spartan-6 Libraries Guide for HDL Designs, October 2013
24. Yu, P., Schaumont, P.: Secure FPGA circuits using controlled placement and routing. In: CODES+ISSS 2007, pp. 45–50 (2007)