DPA Resistant AES on FPGA using Partial DDL

Jens-Peter Kaps and Rajesh Velegalati
ECE Department, George Mason University
4400 University Drive, Fairfax, VA 22030, USA
Email: {jkaps, rvelegal}@gmu.edu

***Abstract*—Current techniques to implement Dynamic Differential Logic (DDL), a countermeasure against Differential Power Analysis (DPA) on Field Programmable Gate Arrays (FPGAs) lead to an increase in area consumption of up to factor 11. In this paper we introduce Partial DDL, a technique in which DDL is applied only to a part of the cryptographic hardware implementation. We propose principle rules for Partial DDL to guide the designer in how to split up a circuit into DDL protected and unprotected paths. In order to validate our approach we implemented a lightweight architecture of AES in the Partial Separated Dynamic Differential Logic (Partial SDDL) for FPGAs. The results show that our implementation with Partial SDDL is as resistant to DPA as a full SDDL implementation while it consumes only 76% of the total area occupied by the full SDDL design. This is an area increase of 2.3 times over an unprotected single ended design.**

I. Introduction

With ever increasing miniaturization and ubiquity of computing devices such as smart cards, wireless sensor network (WSN) nodes, radio frequency identification (RFID) tags etc., security threats against them have become a growing concern [1], [2]. Even though these devices protect confidential information using cryptographic algorithms that withstand rigorous cryptanalytic attacks, an adversary can obtain the secret information by observing the so-called side channel leakage from the cryptographic device. These side channels can be power consumption, execution time, or electromagnetic emanations of the device. Amongst these passive non-invasive side channel attacks (SCA), the power analysis attack [3], [4] has received the most amount of attention by the research community because it is very powerful, can easily be conducted, and has been used successfully many times. It can be applied to dedicated cryptographic processors as well as to general purpose processors running a cryptographic software. The fact that ultra-low power implementations of cryptographic algorithms perform most operations in a serial fashion, in order to conserve power, makes them especially susceptible to DPA.

When it turned out that the cryptographic devices are vulnerable to power analysis, there has been great effort in the development of countermeasures against DPA. Dynamic Differential Logic (DDL) styles have been very successful in thwarting DPA attacks on ASICs [5]. However, current implementations of DDL styles on FPGAs have a very large area overhead which is not suitable for low area implementations.

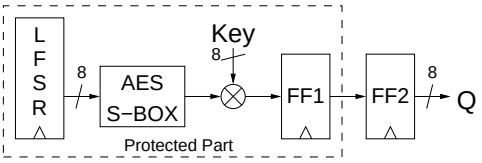

Fig. 1. Block Diagram of Test Circuit

DDL aims to break the connection between the instantaneous power consumption of a circuit and the data being processed. This would make a DPA attack infeasible. DDL accomplishes this goal through duplicating the circuit into a direct and a complementary logic. It pre-charges the inputs of the circuit and the outputs of all memory elements during one half of a clock cycle (pre-charge phase) and performs the computation in the other half (evaluation phase). This guarantees constant switching activity, i.e. either a bit in the direct or the corresponding bit in the complementary logic switch.

For example consider the circuit shown in Fig.1. An 8-bit LFSR is used to supply inputs to the SBOX. The output of the SBOX is XORed with key and stored in register FF1. The register FF2 drives the outputs of the chip and is implemented in I/O blocks (IOB). Table I shows the comparison between different DDL styles applied to the test circuit shown in Fig.1.

The Single Ended (SE) implementation of the circuit shown in Fig.1 makes heavy use of Wide Dedicated Multiplexers (WDM)s, a special intrinsic features in Xilinx FPGAs, when synthesizing the S-Box. The Separated Dynamic Differential Logic for FPGAs (SDDL for FPGAs) [6] design does not use the WDMs. Hence, the area consumption of the SDDL for FPGA design is nearly 4 times that of Single Ended (SE) design. The WDDL implementation has an area overhead of nearly 5 times and DWDDL nearly 11 times to that of a SE design. DWDDL is the most secure of the three DDL styles although its area consumption makes it impracticable for many FPGA applications. The area consumption and security estimates of SDDL for FPGAs style is taken from [6] and that of Wave Dynamic Differential Logic (WDDL) and Double Wave Dynamic Differential Logic (DWDDL) is taken from [7], [8].

Although DDL implementations on FPGAs are explored in [5], [9]–[11], to our knowledge not much work has been done to reduce the area overhead incurred due to DDL styles. In [9] Guilley et al. presented optimization techniques which reduce the size of a WDDL implementations on FPGAs. They were able to reduce the size of their WDDL implementation

TABLE I
COMPARISON BETWEEN DIFFERENT DDL STYLES APPLIED TO THE TEST CIRCUIT

Design Methodology	SE	SDDL for FPGAs	WDDL	DWDDL
DPA Resistance	No	Medium	Medium	High
Area Consumed (in slices)	70	283	409	818
Increase in Area Over SE	1.00	4.04	5.84	11.68

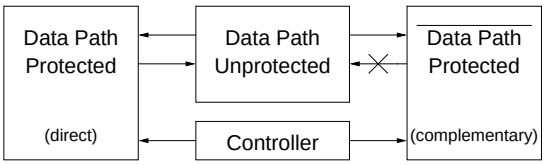

Fig. 2. Block Diagram of Partial DDL Implementation

of Triple DES by 23% [8] through a new synthesis flow. However, this design is still much larger than a single ended design due to the use of only positive logic as required by WDDL.

It is the goal of this paper to introduce Partial DDL, propose principle design rules of Partial DDL, and show its effectiveness in terms of area saving and DPA resistance through a proof-of-concept implementation of a lightweight AES design using SDDL for FPGAs and comparing it with one using Partial SDDL for FPGAs.

The rest of the paper is structured as follows. Section II explains the concept of Partial DDL and the principle rules for implementing Partial DDL. The secure design flow for implementing Partial DDL on FPGAs is described in Section III. Next, in Section IV the lightweight implementation of AES, the attack methodology, and the design considerations for AES Partial SDDL are discussed. In Section V we present the implementation results secured AES using Partial SDDL. Section VI concludes the paper and discusses future research perspectives related to Partial DDL.

II. PARTIAL DDL

Partial DDL describes the notion that DDL is applied to only a part of the cryptographic implementation. Consider the block diagram shown in Fig 2. The data path of the cryptographic implementation is divided into two parts, *Data Path Protected* and *Data Path Unprotected*. DDL is implemented only on the protected data path splitting it into a direct and a complementary part (indicated as $\overline{DataPath}$). DDL is not applied to the unprotected path and the control block of the implementation. It is already a common practice [9], [12] to not protect the control logic as it manipulates only public information of the algorithm. The protected and unprotected data paths are interconnected but there is no connection between the complementary data path and the unprotected data path, shown as crossed out line in Fig 2. The advantage of Partial DDL is that the area overhead is reduced (2 x protected data path) compared to the area overhead incurred due to applying DDL over the entire design. The drawback of Partial DDL is that the power consumption will not be as constant as when DDL is applied to the entire data path.

A. Partitioning the Data path

The first step in a DPA attack is to choose an intermediate result of the cryptographic algorithm being executed, which should be a function of input data and the secret key. Then we calculate the hypothetical power model depending upon these intermediate values. Some intermediate results are easier to model while others are harder. We partition the data path depending upon the complexity of these power models. The protected data path contains the parts of the cryptographic algorithm which are easier to model and thus need to be protected. The unprotected data path contains the parts of the cryptographic algorithm which are harder to model, requiring complex statistical test (template attacks etc.) and much larger number of measurements to disclose the key.

B. Principle Rules for Partial DDL

We propose the following principle rules for implementing Partial DDL on cryptographic implementations:

- The cryptographic implementations should be thoroughly analyzed with respect to DPA and *only then the data path should be partitioned*.
- The controller and the unprotected data path block should be placed inside the FPGA fabric in such a way that the distance of the signals from the control block and the output signals from the unprotected data path to the direct and complementary part should be as similar as possible, so that there would not be any delay of operation between the direct and complementary part. As the routing of these signals is performed by the FPGA tools in our design flow, we can not control the routing precisely.
- Special care must be taken when connecting the unprotected data path to the complementary circuit. The output signals from unprotected data path should either be inverted (which will result in additional area consumption) or appropriate changes in the LUT logic equation must be made. Control signals to the complementary part should not be inverted.
- Output signals from the complementary circuit either to the unprotected data path or to the IOBs will be terminated in the slice itself.

We demonstrate the partial DDL approach on a lightweight implementation of AES on FPGA in the next sections.

III. SDDL FOR FPGA

Our SDDL model [6] is shown in Fig. 3. Instead of applying De-Morgan's law to obtain the duplicate part we simply invert the inputs and outputs of the original logic gate. This technique is not suitable for ASICs because placing a inverter on both inputs and outputs will increase the area consumption.

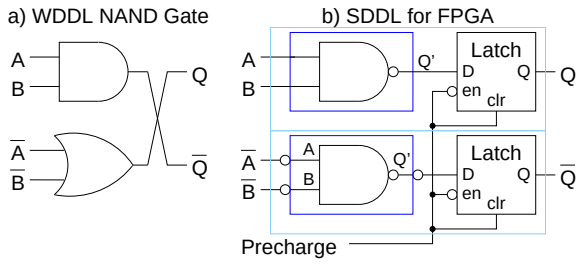

Fig. 3. Proposed SDDL Model

Whereas in FPGAs, we need to modify only the logic equation in the LUTs, which will not cause any area overhead. FPGA CAD tools are given the maximum flexibility to optimize a given design for the target FPGA thus, allowing logic packing in LUTs and also make use of all the intrinsic features present in the FPGA with the exception of WDMs.

A. Hard-Macro for Register Pre-Charging

During our work on exploring the DPA resistance of WDMs when applied to SDDL [6] we made the following observations:

- The connection between the LUT and the flip-flop/latch in the same slice does not leak exploitable information.
- The connection between two slices in the same Configurable Logic Block (CLB) does not leak any exploitable information.
- Any connection between CLBs leaks exploitable information. This includes even the fast dedicated interconnects used by WDMs.

Hence the pre-charge circuit for registers should be present in the same CLB so that sensitive information is not leaked.

Pre-charge circuits are inserted into an SE design using the technique described in [6], [7]. In Xilinx Spartan3 FPGAs a CLB is comprised of four slices, each containing two Look-up Tables (LUT) which are always followed by two storage elements that can be used as either flip-flop or latch. The pre-charge circuit is implemented by using an asynchronously cleared latch to ensure the propagation of the '0' wave even if the output of an LUT is '1'. If a flip-flop is already used in the design then the pre-charge circuit should be inserted in a slice from the same CLB so that the routing between the two slices is at minimum. For designs that use low area resources, it is possible to control the Place and Route (PAR) tool to leave the slices present near the registers in the same CLB empty. However for larger designs, controlling the PAR tool becomes too cumbersome. Hence we use a hard macro to pre-charge the output of the register.

Hard macros are block level designs of logic functions that specify how the logic elements are interconnected and the connections between the components routed. There are 8 flip-flops in a CLB hence the maximum register length with a corresponding pre-charge circuit is 4 bits. We created a 4-bit register/pre-charge hard macro (see Fig. 4) as it fully utilizes a CLB and does not restrict the Xilinx router, which larger hard macros would. All registers in the AES module

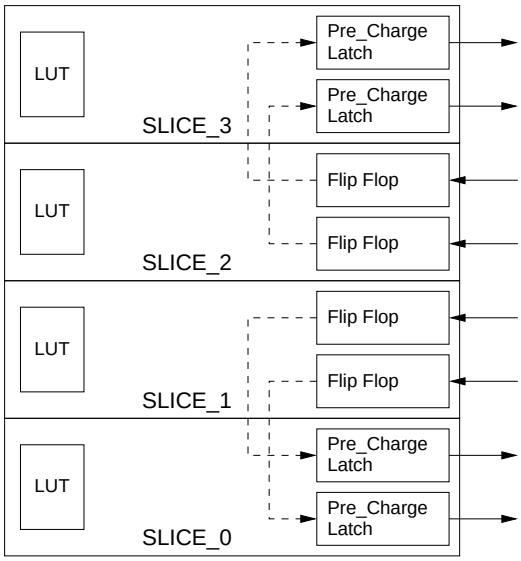

Fig. 4. Hard-Macro of Four Flip-Flops with Pre-Charge

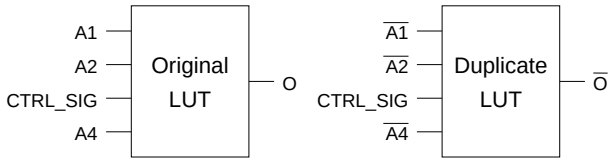

Fig. 5. Controlled Complementing of Logic

are a multiple of 4 in length. The dotted lines are the interconnects between the flip-flops and their corresponding pre-charge circuits. These routes are inter-slice connections and pass through the switch box.

B. Duplicating and Complementing

We duplicate the protected data path part only. The duplication and relocation processes are described in [6].

If $f(x)$ is the equation which defines a LUT in the direct path, then its complementary equation $g(\bar{x})$ is given by

$$g(\bar{x}) = \overline{f(\bar{\bar{x}})} = \overline{f(x)} \tag{1}$$

Since we do not apply SDDL to the control block of the implementation, the control signals that are to be connected to the complementary block are not inverted. Creating inverted control signals causes area overhead. Therefore, care must be taken, so that the control signals are not inverted in the logic equation, as shown in Fig. 5. When connecting the unprotected data path to the complementary path we follow the same principle.

C. Secure Design Flow for Partial SDDL

Our design flow for implementing Partial SDDL on FPGAs uses Xilinx ISE Design suite 10.1 and Perl scripts. It consists of three phases, as shown in Fig 6.

In the first phase, the single ended design is synthesized and implemented. We apply area constraints to perform the following tasks:

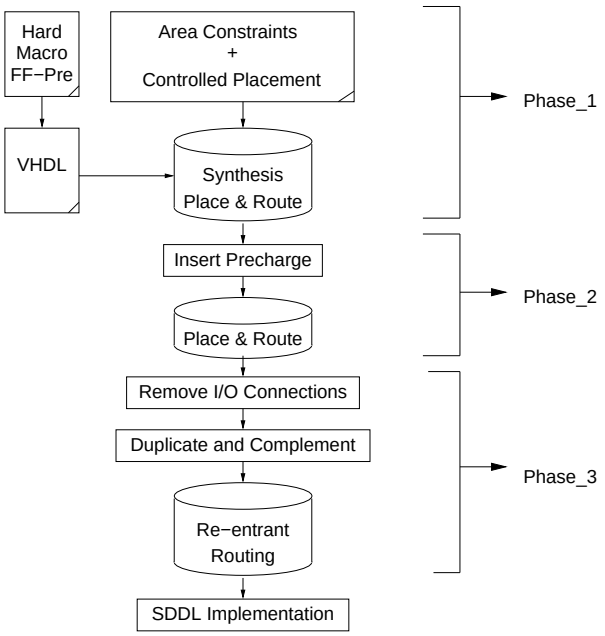

Fig. 6. SDDL Design Flow

- Limit the design to one section of the FPGA fabric, keeping other sections empty so that we can use them to place the complementary path.
- The controller block and unprotected data path part have to be constrained in such a way that the distance form them to the direct and complementary part are as similar as possible. This will minimize differences in the arrival time of unprotected data and control signals and hence ensure that the principles of SDDL on FPGAs are not violated.

In the second phase, the circuit description file from the first phase is converted into ASCII representation with help of the XDL (Xilinx Design Language) tool. Perl scripts interpret the XDL file and insert pre-charge. Subsequently only Place and Route is executed.

In the third phase, the I/O connections are removed and the design is again converted into XDL format. Our Perl scripts duplicate and complement the protected data path part. This duplication preserves the routing of the original design. Hence, each gate output in the original design drives an equivalent load as the corresponding output in the complementary part. The scripts are also used to link the unprotected data path and all control signals to the complementary circuit. However, the signals from the I/O pins and all signals connecting the complementary part to the controller block and unprotected data path are still not routed. In order to preserve the routing of the original and complementary parts we use Place and Route in re-entrant routing mode.

IV. AES Implementation and Attack Methodology

A. AES Implementation

The Advanced Encryption Standard (AES) [13] is one of the most widely used block ciphers. It was designed to be resistant towards linear and differential cryptanalysis. AES is a block cipher of fixed input size of 128 bits and key length of either 128 or 192 or 256 bits. For our AES implementation [14] we chose to use a key length of 128 bits. AES applies the same round function ten times to its inputs during encryption. The round function consists of four different transformations *SubBytes*, *ShiftRows*, *MixColumns* and *AddRoundKey* each changing the input by applying linear, non linear and key dependent transformations. Our AES implementation also assumes that the data input and the secret key are stored in memory. The AES transformations are grouped into four stages

1) Initial AddRoundKey-SubBytes-ShiftRows
2) MixColumns
3) AddRoundKey-SubBytes-ShiftRows
4) FinalAddRoundKey

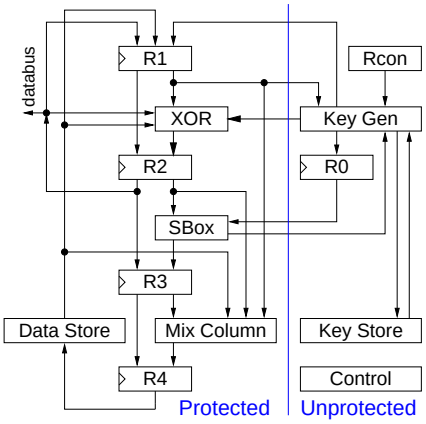

Fig. 7. Block Diagram of AES Module

The data path of our AES implementation is shown in Fig. 7. It is characterized by a pipelined architecture for stages 1 and 3. This enables us to re-use registers and minimize the number of internal memory accesses which in turn reduces the number of clock cycles. Five registers R_0, R_1, R_2, R_3, R_4 are used of which R_0 is used exclusively for *RotWord* operation. R_1 is used for key computation and state computation in *MixColumns* operation, R_2, R_3, R_4 are used for state computation. The boxes labeled as *Key store* and *Data store* are 128 bit registers used for Round keys and State Memory respectively.

In order to provide different plain text and key as input to the AES module we built a wrapper circuit, as shown in Fig. 8. A 128-bit Linear Feedback Shift Register (LFSR) provides input data to the AES module. An 8-bit wide 32:1 multiplexer is used to select data or key depending upon the address from the AES module.

B. Design consideration for AES- Partial SDDL

The implementation of the single ended AES design (AES SE) consumes 393 slices excluding the wrapper circuit. Table II breaks down the area consumption of AES into its components. The pre-charge design is the result of the second step of our SDDL design flow.

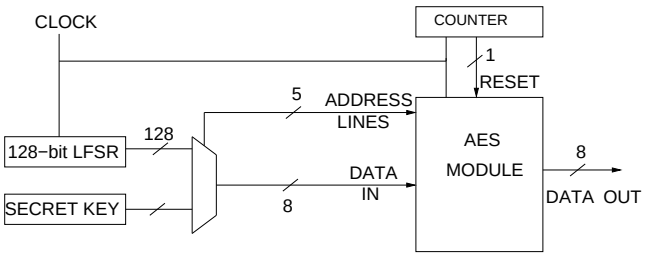

Fig. 8. AES Module With a Wrapper Circuit

TABLE II
ESTIMATED SLICE CONSUMPTION OF INDIVIDUAL COMPONENTS IN AES

AES-Component	Slice Count			
	SE Design	Pre-charge	Partial SDDL	Full SDDL
Data Store + Mix Columns	106	178	356	356
Key Store + Key Expansion	98	98	98	196
AES Computations	51	71	142	142
SBOX	64	129	258	258
Controller and Address	74	74	74	148
Total	393	550	928	1100

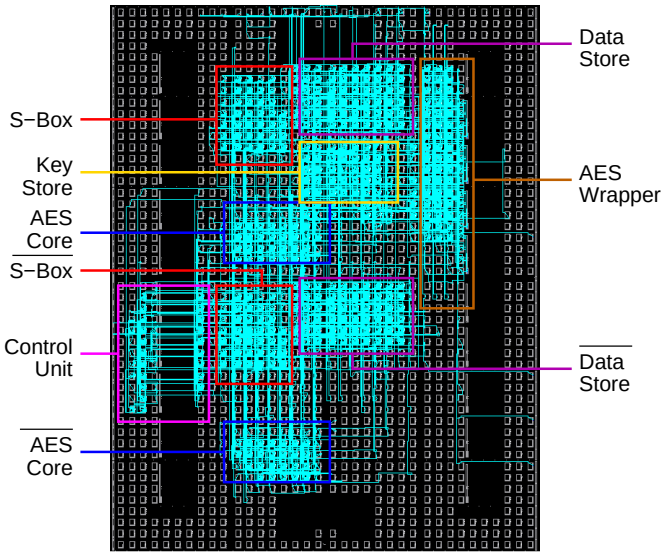

Fig. 9. Placement of AES Modules on FPGA Fabric

Pre-charging increases the slices consumed by a register by factor 2. In our design we substituted the registers with the hard macro. For example the data store register which is 128-bits in length can be implemented in 64 slices in the SE design. After pre-charging the area consumed by the data store component becomes 128 slices. We restricted our pre-charged AES design to use only 4:1 multiplexers as our earlier experiments [6] showed that the use of WDMs leaks information. Due to this restriction the slice consumption of the SBOX component increases to twice that of the original value.

The third and fourth column in Table II estimate the slice count for the Partial SDDL and full SDDL implementations respectively. For the full SDDL implementation the entire pre-charge implementation is duplicated with the exception of the wrapper circuit. The Partial SDDL implementation does also not duplicate the Key Store, Key Expansion, Controller, and Address generation units. In summary, SDDL implementations of simple logic functions incur an area increase of factor 2 due to duplication of logic. This penalty is increased when the original SE design uses WDMs. The overhead for registers is a factor 4 due to pre-charge and duplication. This greatly effects lightweight implementations which are severely area constraint.

C. Attack Methodology

Consider the data flow from R_2 to R_3 and R_4 in Fig. 7. Resetting the AES module changes the data value in these registers to 0x00. In the first clock cycle, the first byte of the key is loaded into R_1. Registers R_2, R_3, and R_4 are not enabled and hence remain at 0x00. In the second clock cycle the output of the register R_3 changes from 0x00 to 0x63 i.e. SBOX(R_2) and the data value in R_2 changes to the input data XORed with key from R_1. In the subsequent clock cycle the data value in the register R_3 changes from 0x63 to SBOX(R_2). This sequence of change in the data values of the register R_3 i.e $0x00 \rightarrow 0x63 \rightarrow SBOX(Key \oplus Inputdata)$ occurs every time the AES module is reseted. Thus we know two consecutive data values of a register and can apply the Hamming Distance (HD) model to simulate the power consumption of the register. Therefore, we use a counter to reset the AES module after every 10 clock cycles. The power model for the single ended design is given by Equation (2). It calculates the HD between the SBOX value of key guess XOR data and the hex value of 0x63.

The key register R_0 is a probable attack point, although the attack itself will be difficult, i.e. it will probably require more Measurements To Disclosure (MTD) compared to attacking the registers R_3 or R_4. The reason is that we cannot compute an HD model as the key store register output is always an unknown value (Round Keys). In this case the Hamming weight (HW) model as shown in Equation (4) can be used. In order to attack the register R_0 the adversary has to identify the power consumption levels corresponding to the HW of the data being processed. This requires complex statistical testing [15]. Due to this reason and the fact that the SDDL version of the key store register will cause a large area over head, we decided not to duplicate the key store register. This is a risk we are taking to limit the area overhead although the security of the final SDDL design may decrease. We also decided not to duplicate the controller as it leaks only the public information of the algorithm. The Controller does not manipulate any sensitive information. Fig. 7 shows the blocks which are pre-charged and duplicated on the left and the unprotected blocks on the right.

We perform Correlation Power Analysis [16] using Pearson's Correlation [16], [17] in conjunction with either Hamming Distance model or Hamming Weight model (depending upon the situation).

$$P_{guess} = \mathrm{HD}\left(0x63, (\mathrm{SBOX}((Key_{guess} \oplus Input))_i\right) \quad (2)$$

$$P_{guess} = \mathrm{HD}\left(0x00, (\mathrm{SBOX}((Key_{guess} \oplus Input))_i\right) \quad (3)$$

$$P_{guess} = \mathrm{HD}\left(0x00, (Key_{guess})_i\right) = \mathrm{HW}(Key_{guess}) \quad (4)$$

We estimate the output of the SBOX for all possible key guesses and mount a DPA attack on SE design using Equation (2). We use a different power model to mount a DPA attack on SDDL designs, given by Equation (3). The pre-charge phase sets all logic outputs to '0' therefore, the Hamming distance is computed between '0' and the estimated outputs of the SBOX for all possible key guesses.

D. Placement of AES Modules

The placement of the AES modules of our partial SDDL design is shown in Fig. 9. The SBOX, Data Store, and AES Computation (AES Core) are duplicated and complemented such that they use the same CLB and routing resources. We placed these closely connected parts near each other. The location of the controller and the key store register inside the FPGA fabric has an impact on the DPA resistance. We placed the control unit close to the AES Core which it controls. The key store is placed between the data stores as they all have to communicate with the AES wrapper. It can clearly be seen that it is impossible to have symmetrical connections from the protected modules SBOX, Data Store, and AES computation to the unprotected path.

V. Results and Analysis

A. Experimental Setup

The target platform for our designs is a Xilinx Spartan 3e starter board containing a XC3S500eFG320-4 FPGA. We removed the capacitances of the core voltage net and connected it to an external regulated power supply in order to obtain a clear power measurement. Power consumption is measured using a Tektronics CT-1 current probe and an Agilent DSO6054A oscilloscope, which has a bandwidth of 500MHz and samples at 4GSa/sec.

B. Analysis of AES test circuit

We implemented three different designs of our AES circuit. AES-SE is a single ended design which use 32:1 WDMs. AES-Partial SDDL is the symmetrically routed design of the said single ended and AES SDDL is a full SDDL implementation. Figure 10 shows the power consumption traces for all three designs. The single ended wave form has its peaks near the rising edge of the clock. The SDDL designs show lower peaks during pre-charge phase and higher peaks during the evaluation phase. It can also be clearly seen that the peaks of the SDDL designs are more uniform compared to the ones of the single ended. The largest peaks in the waveform of both designs occur when the 128-bit LFSR is clocked.

The key byte which we are attacking is a fixed value of 10 for all designs. The correlation plot between power guess and power measured for the AES SE implementation taken

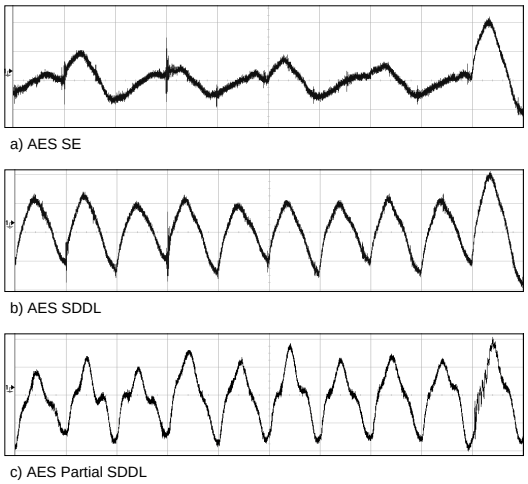

Fig. 10. Power Traces (5 mV/div, 1 μs/div)

TABLE III
Results of AES-Partial SDDL Implementation

Design	Slices	Increase in Area over SE	MTD
AES SE	393	1	500
AES Partial SDDL	928	2.3	$> 12,000$
AES SDDL	1222	3.1	$> 12,000$

over 500 measurements shows a sharp peak at the key guess 10, as shown in Fig. 11. Corresponding to Fig. 11 is Fig 14 in which the blue lines indicate the maximum and minimum correlations of all key guesses depending on the number of measurements. The red line indicates the correlation of the correct key over number of measurements. The point in which the red line crosses the blue line and the red line maintains a clear maximum constitutes the number of measurements to disclosure of the key (MTD). The correlation plots for the SDDL designs did not show a definite peak after 500 measurements. Therefore, we had to take multiple sets of measurements. We obtained a clear peak for the correct key after more than 12,000 measurements for Partial SDDL as well as for full SDDL as shown in Fig. 15 and Fig. 16. The corresponding correlation plots are in Fig. 12 and Fig. 13.

Table III shows the comparison of the AES designs and their respective MTD. The final AES Partial SDDL implementation is larger than the AES SE by a factor of 2.3. The area consumption of AES Full SDDL is 3.1 times larger then the single ended design. The partial SDDL is 76% of the total area consumed by AES Full SDDL implementation. The security provided by the AES Partial SDDL is 30 times to that of the AES SE design and similar to the full SDDL implementation. The area consumption of the SDDL designs are very close to the ones we expected from our estimation in Table II.

VI. Conclusion and Future Work

Securing cryptographic implementations using DDL logic leads to a large area penalty. We have shown through our example implementation of AES using SDDL for FPGAs, that partial DDL can significantly reduce the area consumption of a DDL implementation by 24%, yet maintain the same

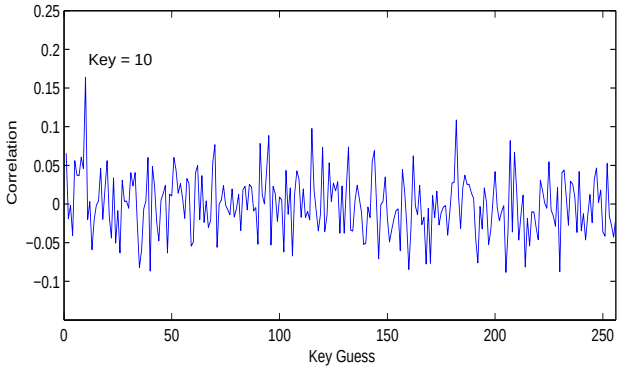

Fig. 11. AES SE — Key Correlation after 500 Measurements

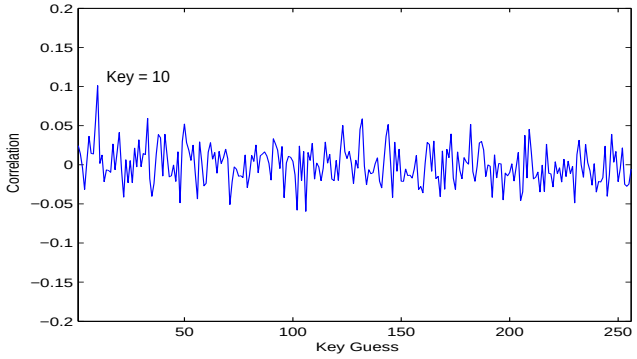

Fig. 12. AES SDDL — Key Correlation after 30,000 Measurements

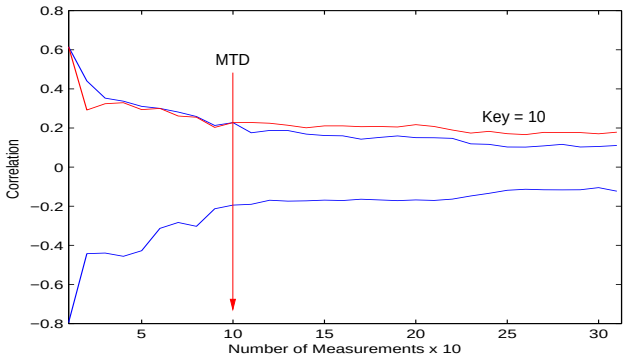

Fig. 14. AES SE — Measurements to Disclosure

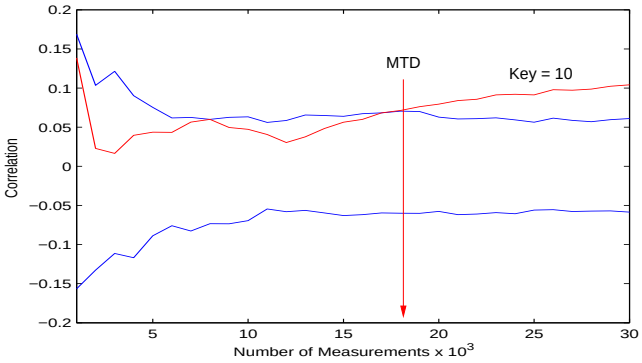

Fig. 15. AES SDDL — Measurements to Disclosure

level of resistance against DPA. Our Partial SDDL can still be broken due to glitches and imbalance of power consumption between the direct and complementary circuits. The next step of our research is to verify that partial DDL offers similar advantages for other ciphers and implementation techniques (e.g. using Block RAMs) and also for other DDL logic styles such as WDDL. Our future work also includes investigating more closely the effect of the placement of protected and unprotected parts with regards to information leakage and to reduce the glitches.

References

[1] M. Hutter, S. Mangard, and M. Feldhofer, "Power and EM attacks on passive 13.56 MHz RFID devices," in *Cryptographic Hardware and Embedded Systems - CHES 2007*, ser. Lecture Notes in Computer Science (LNCS), vol. 4727. Springer, 2007, pp. 320–333.

[2] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Investigations of power analysis attacks on smartcards," in *USENIX Workshop on Smartcard Technology*, 1999, pp. 151–161.

[3] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology - CRYPTO'99*, ser. Lecture Notes in Computer Science (LNCS), vol. 1666. Berlin: Springer Verlag, Aug 1999, pp. 388–397.

[4] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks, Revealing the Secrets of Smart Cards*. Springer, 2007.

[5] K. Tiri and I. Verbauwhede, "A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation," in *Proc. Design, Automation and Test in Europe (DATE'04)*. IEEE Computer Society, Feb 2004, pp. 246–251.

[6] R. Velegalati and J.-P. Kaps, "DPA resistance for light-weight implementations of cryptographic algorithms on FPGAs," in *Field Programmable Logic and Applications, FPL 2009*, M. Daněk, J. Kadlec, and B. Nelson, Eds. IEEE, Aug 2009, pp. 385–390.

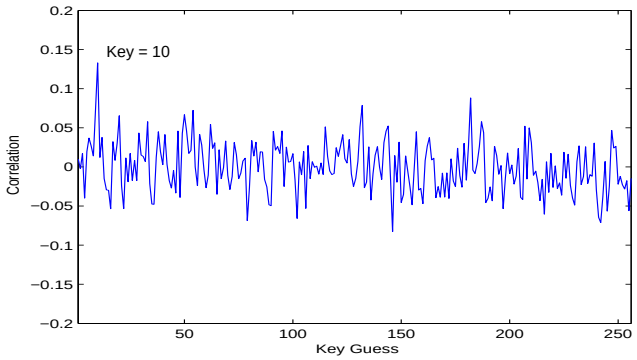

Fig. 13. AES Partial SDDL — Key Correlation after 30,000 Measurements

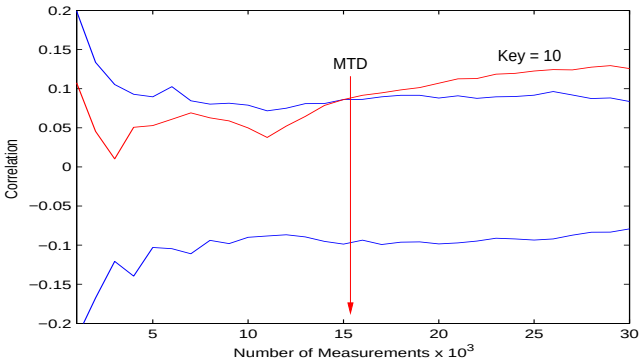

Fig. 16. AES Partial SDDL — Measurements to Disclosure

[7] P. Yu and P. Schaumont, "Secure FPGA circuits using controlled placement and routing," in *CODES+ISSS '07: Proceedings of the 5th IEEE/ACM international conference on Hardware/software codesign and system synthesis*. New York, NY, USA: ACM, 2007, pp. 45–50.

[8] S. Guilley, L. Sauvage, J.-L. Danger, and P. Hoogvorst, "Area optimization of cryptographic co-processors implemented in dual-rail with precharge positive logic," in *Field Programmable Logic and Application – FPL 2008*, U. Kebschull, M. Platzner, and J. Teich, Eds. IEEE, Sep 2008, pp. 161–166.

[9] S. Guilley, L. Sauvage, J. Danger, T. Graba, and Y. Mathieu, "Evaluation of power-constant dual-rail logic as a protection of cryptographic applications in FPGAs," in *Secure System Integration and Reliability Improvement (SSIRI '08)*. IEEE, Jul 2008, pp. 16–23.

[10] T. Popp and S. Mangard, "Masked dual-rail pre-charge logic: DPA-resistance without routing constraints," in *Cryptographic Hardware and Embedded Systems – CHES 2005*, ser. Lecture Notes in Computer Science (LNCS), J. R. Rao and B. Sunar, Eds., vol. 3659. Heidelberg: Springer, 2005, pp. 172–186.

[11] R. P. McEvoy, C. C. Murphy, W. P. Marnane, and M. Tunstall, "Isolated WDDL: A hiding countermeasure for differential power analysis on FPGAs," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 2, no. 1, pp. 1–23, Mar 2009.

[12] S. Guilley, S. Chaudhuri, L. Sauvage, T. Graba, J.-L. Danger, P. Hoogvorst, Vinh-Nga, and M. Nassar, "Place-and-route impact on the security of DPL designs in FPGAs," in *Hardware-Oriented Security and Trust, HOST 2008*. IEEE, 2008, pp. 26–32.

[13] *Advanced Encryption Standard (AES)*, National Institute of Standards and Technology (NIST), FIPS Publication 197, Nov 2001, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[14] J.-P. Kaps and B. Sunar, "Energy comparison of AES and SHA-1 for ubiquitous computing," in *Embedded and Ubiquitous Computing (EUC-06) Workshop Proceedings*, ser. Lecture Notes in Computer Science (LNCS), X. Z. et al., Ed., vol. 4097. Springer, Aug 2006, pp. 372–381.

[15] S. Mangard, "A Simple Power-Analysis (SPA) Attack on Implementations of the AES Key Expansion," in *Information Security and Cryptology ICISC 2002*, ser. Lecture Notes in Computer Science, P. Lee and C. Lim, Eds., vol. 2587. Berlin: Springer, Nov 2002, p. 343358.

[16] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Cryptographic Hardware and Embedded Systems – CHES 2004*, ser. Lecture Notes in Computer Science, vol. 31. Berlin / Heidelberg: Springer, Aug 2004, pp. 135–152.

[17] S. Aumônier, "Generalized correlation power analysis," in *Ecrypt Workshop Tools For Cryptanalysis 2007*, 2007.