# Experiments in Attacking FPGA-Based Embedded Systems using Differential Power Analysis

Song Sun    Zijun Yan    Joseph Zambreno
Dept. of Electrical and Computer Engineering
Iowa State University
Ames, IA 50011
Email: {*sunsong, zijunyan, zambreno*}@*iastate.edu*

***Abstract*—In the decade since the concept was publicly introduced, power analysis attacks on cryptographic systems have become an increasingly studied topic in the computer security community. Research into countermeasures for these cryptographic systems has intensified as well. Experiments have been conducted showing the potential effectiveness of power analysis attacks and preventative techniques on both software (e.g. smartcard, DSP) and hardware (e.g. ASIC, FPGA) processing elements. One key observation that motivates our work is that the majority of the research into power analysis on FPGA-based cryptographic systems has been a) theoretical in nature, b) evaluated through simulation, or c) experimented using custom hardware that does not closely mirror real-world systems. In this paper, we look to bridge this gap between theory and practice by detailing our experience in performing a Differential Power Analysis (DPA) attack on a commercial FPGA development board. We present an automated data acquisition and analysis design for an FPGA-based implementation of the Data Encryption Standard (DES), and discuss some of the challenges and obstacles that we encountered when performing the DPA attack on our chosen commercial platform.**

## I. Introduction

Power analysis attacks are regarded as a very powerful approach to cracking cryptographic systems [1]. This class of attacks make use of the power consumption information from processing elements built using CMOS circuits. Introduced first by Kocher et al. [2] in 1998, there are two main flavors of power analysis attack: the Simple Power Analysis (SPA) attack and the Differential Power Analysis (DPA) attack. In the SPA attack, by tracing the whole system power consumption information, the adversary can deduce the types of instructions running in the processing element, which in a cryptographic system will often be directly influenced by the choice of secret key. By comparison, the DPA attack combines this power analysis with statistical and error correction techniques, leading to a more powerful approach. As will be described in Section III, due to the statistical nature of the DPA attack, an attacker using DPA is not required to know any details of the internal algorithmic implementation.

Reconfigurable computing systems, such as those based on Field Programmable Gate Array (FPGA) technology, are a very promising platform for designing high-performance cryptographic systems, due to their high throughput rates and inherent design flexibility [3]. The growing popularity of FPGAs as a cryptographic processing element has necessitated research into their susceptibility to power analysis attacks. Indeed, much previous effort has gone into applying DPA attacks on FPGA platforms [4], [5], [6], [7], and on developing corresponding anti-attack methods [8], [9], [10], [11], [12]. Unfortunately, in all of the previous research on this topic, the experimental results and analysis were based on either simulated power consumption models or synthetic hand-made FPGA boards. As a result, little is known as to the practical impact that DPA attacks can have on commercial FPGA boards. In our opinion this may lead to a disconnect between the theory and practice of protecting FPGA-based embedded systems.

In this paper, we describe an automated data acquisition and analysis system for applying a DPA attack on an FPGA executing a cryptographic algorithm. Using this system, we mounted an attack on a Xilinx Virtex-II Pro FPGA running a Data Encryption Standard (DES) core. Our choice of both FPGA board and cryptographic algorithm are driven by their respective popularity; the Virtex-II Pro was the first Xilinx FPGA that was capable of running hardware/software designs in an integrated reconfigurable fabric, and can commonly be found in both academic and industrial environments. DES was the first standardized cryptographic implementation broken by the power analysis community, and as such remains a popular target, even after the introduction of newer, more robust private-key algorithms.

Our goal in running these experiments is to provide an in-depth case study to the security community describing the challenges inherent in performing real-world DPA attacks on FPGA-based systems. One surprising result of our work is that the practical impact of DPA (and other power analysis) attacks on commercial FPGA boards is severely limited by several factors not considered in previous research, and as such we strongly suggest that many of the current efforts into DPA countermeasures may be misguided. At the very least our work implies that given the physical access required by the DPA methodology, an attacker's time would be better spent performing other non-invasive techniques. We hope to stir debate in the security community on the actual reach and importance of DPA attacks on real-world hardware systems.

The remainder of this paper is organized as follows. In the following section, we review current efforts in performing and preventing power analysis attacks on FPGA-based embedded systems, both of the SPA and DPA variety. In Section 3, we describe some of the basic theory and practice of the DPA attack, and describe how it can be applied to an FPGA board. In Section 4, we detail our automated data acquisition and analysis system, and describe the experimental setup and steps taken. We provide an analysis of the collected data in Section 5, and discuss some of the practical implications of our results. Finally, the paper is concluded in Section 6 with a summary of the direction for our future work, and an introductory discussion of suggestions for the DPA on FPGA research community.

## II. Related Work

Research into power analysis attacks on cryptographic systems has flourished ever since the topic was first announced by Kocher et al. [2]. Originally intended as a technique to use on smartcards and other portable software systems, in recent years power analysis has been applied to FPGA-based computing systems. The authors in [4] investigated DPA attacks on FPGA platforms through the use of a simulator that counted the transitions of CLB output signals to estimate power. The authors also evaluate the usefulness of various gate-level countermeasures to DPA through the use of this simulation infrastructure. Although the conclusions presented were of potential use (the authors concluded that only a few nodes in the circuit had a high relation to the bits of the secret key), the ultimate value of this and other similar approaches is lessened by the choice of a simulator that considers the power consumption of the FPGA in isolation from its supporting environment.

An investigation into performing DPA on actual FPGA hardware is presented in [5]. One limitation of this approach is that the FPGA is placed on a custom board in order to facilitate the power analysis. Conclusions made in this synthetic environment may not have a direct corollary when adapted to a commercial FPGA board.

The power consumption characteristics of an FPGA is not fundamentally different from that of an ASIC using CMOS technology [13]. The authors in [6] first generalize this power consumption model using DES transition counts, and then perform a DPA attack by correlating the real measurement data to their model.

As previously mentioned, many of the reported power analysis attacks are based on the DES encryption standard [14]. More recently some have began the initial work required to perfom a DPA attack on implementations of AES [15], [13] and elliptic curve cryptosystems [8].

After the initial reports of successful power analysis attacks, many countermeasures have been proposed in response. These SPA- and DPA-resistant techniques try to solve the problem from different angles, by looking at hardware design at the logic level, as well as the interaction between instruction set architecture and software. One of the first logic level countermeasures was the transformed masking method, which was introduced in [9]. In [10] the authors proposed a family of DPA-resistant compound standard cells, referred to as Wave Dynamic Differential Logic (WDDL). In theory, each WDDL gate has a constant power consumption profile; a fixed charge is used for each signal transition, making the consumption independent of the transition frequency. The same authors have also presented a place-and-route methodology [11] and full VLSI design flow [12] in support of their technique.

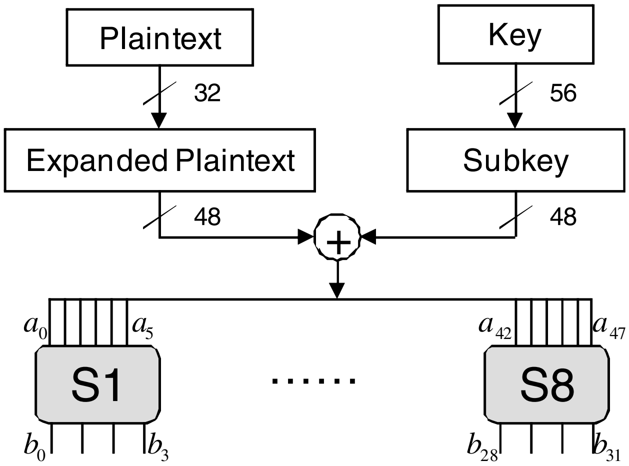


Fig. 1. Structure of the S-box in a round of DES

Many of these proposed anti-DPA measures are themselves nullified by improved attack technology. It is also important to note that any imperfect masking will only serve to obfuscate the power consumption characteristics of a circuit, which can be circumvented at the expense of additional computational or data acquisition time. Even if the DPA computational workload for an attacker is increased beyond reasonable limits [7], higher-order differential power analysis attacks may still be possible [16].

## III. Differential Power Analysis Principles

DPA is a passive attack that is performed by externally observing the power consumption of a circuit performing cryptographic computations. The theory behind DPA is that the power consumed by the computational logic is statistically correlated with the internal bit transitions.

We selected DES as the attack target since DES was the first algorithm used to demonstrate the practicality of DPA in [2]. In the following explanation, we assume that the attacker knows the input plaintext values (the *known plaintext* variety of DPA attack). The DPA process can be described as follows:

First, the attacker records $N$ plaintexts and their corresponding power traces $P[N]$. In reality, the plaintext value are likely to be randomly distributed. The power trace for each input plaintext can be represented as $P[N][M]$. The voltage value on power trace $P[i]$ at time $j$ is $P[i][j]$. The value of $N$ and $M$ can be determined by the attacker. The larger the value of $N$, the more accurate is the guess of the final extracted key. A larger $M$ implies a higher power sampling rate.

Next, the attacker chooses an output bit of a S-box in the first round. The structure of the S-box in DES is shown in Fig. 1. Typically, the first output bit $b_0$ is chosen. Bit $b_0$

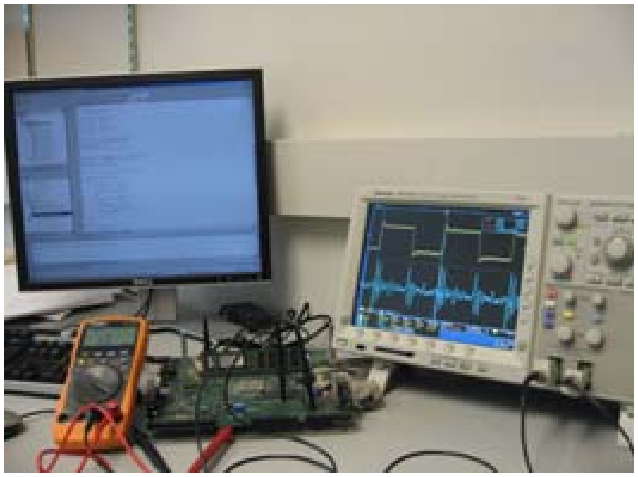


Fig. 2. Data acquisition experimental setup

depends on the six bits of the secret key and plaintext. The attacker makes an initial guess of those key bits (out of 64 possible values). Based on the guessed value of the six bits of the key and the known-plaintext, one can compute the guessed value of $b_0$. Since bit $b_0$ can only have two values (0 or 1), the attacker can divide the whole $N$ power traces into two groups according to the value of $b_0$. For each power trace of the $N$ iterations using varied plaintext values, it is assigned into the first group $A$ if the theoretical value of $b_0$ is 0; otherwise, it is assigned into the second group $B$. Once all of the power traces values have been acquired, the average power is calculated. The average power is calculated at each time point $j$ using the equations:

$$\bar{P}_A^j = \frac{1}{|A|}\sum_{i=1}^{|A|} P[i][j] \tag{1}$$

$$\bar{P}_B^j = \frac{1}{|B|}\sum_{i=1}^{|B|} P[i][j] \tag{2}$$

If the guessed six-bit key is not correct, the computed value of bit $b_0$ will be different from the real value with a probability of 0.5. This has the actual effect of placing a power trace vector randomly into two groups $A$ or $B$. The average power traces of the two groups will be the same if the number of different plaintext approaches to infinity. Hence the difference of average power traces between the two groups will approximately be zero as $N$ approaches infinity.

$$\lim_{N\to\infty}(\bar{P}_A^j - \bar{P}_B^j) = 0, 1 \leq j \leq M \tag{3}$$

However, if the guessed six-bit key is correct, the computed value of bit $b_0$ will be the same as the actual value with a probability of 1. As stated before, the power consumption of the electrical device is correlated to the internal bit transition. The power traces with the value of bit $b_0$ equal to 0 must be different from those with the value of bit $b_0$ equal to 1. In this sense, the average power of group $A$ will diverge from the average power of group $B$, as the number of input plaintext values approaches infinity. The other factors which affect the power data values (such as measurement errors, electrical noise, etc.) that are not correlated to the value of bit $b_0$ will approach zero as the value of $N$ approaches infinity. If the value of $M$ is large enough to encapsulate the power data for each of the sixteen rounds, a spike will be observed for the graph of the average power difference. Everywhere else in the graph the value will converge to zero.

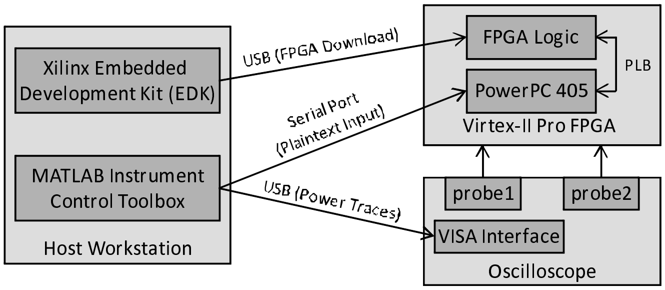


Fig. 3. Components in the experimental setup

$$\lim_{N\to\infty}(\bar{P}_A^j - \bar{P}_B^j)\begin{cases} \neq 0 & \text{for some j} \\ = 0 & \text{otherwise} \end{cases} \tag{4}$$

There are in total eight S boxes in the DES $F$ function. Finding all 48 bits of the subkey can be accomplished by combining all eight 6-bit keys. There are another eight bits in the 56-bit input key that can be easily found in the second round analysis. Once the subkey for the first round is known, the input to the second round can be considered as a known plaintext value. The search process can then continue in a similar fashion as for the first round.

## IV. Experimental Setup

The goal of our experiment is to automate the data collection and analysis stages of the DPA attack targeting a commercial FPGA board. As is shown in Fig. 2, our setup consists of essentially three parts: the digital oscilloscope, the FPGA-based development board and the host PC. Two kinds of software are run on the host PC which we will explain in detail in the following section.

The oscilloscope is responsible for collecting differential power traces from the FPGA board. It is connected with the Xilinx Virtex-II Pro FPGA board through two probe needles. The host PC is the control center of the whole system. It coordinates the activity sequence of the oscilloscope and the Virtex-II Pro board. The host PC and the oscilloscope are connected using USB. The bridge between the host PC and FPGA board is composed of two connections. One is a USB connection which is used for FPGA configuration download. The other one is a serial connection which is used as data communication between the host PC software and the software running on the FPGA (developed using Xilinx EDK) that controls the DES module. More details of the components in the experimental setup are shown in Fig. 3.

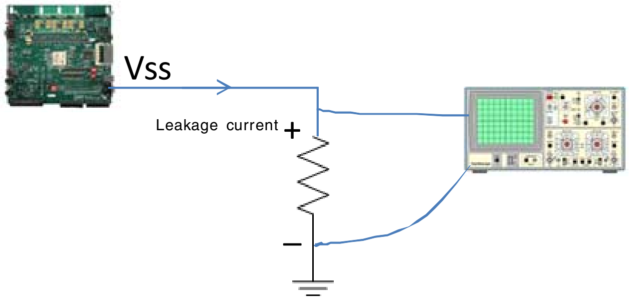


Fig. 4. Probe needle connection to the oscilloscope

### A. Oscilloscope

The oscilloscope is a Tektronix DPO4032 digital phosphor oscilloscope whose sampling rate is 2.5G/s on all channels. A USB 2.0 device port is used for direct PC control of the oscilloscope using the USBTMC protocol. One probe needle of the oscilloscope is connected to the trigger output of the DES module which is running on the FPGA board. The other probe needle is connected to the ground output of the board. The ground output of the oscilloscope is also connected to a resistor whose other end is connected to the true ground of the oscilloscope. One probe needle connection is shown in Fig. 4. The $V_{ss}$ pin is the ground of the board. Leakage current will flow through the resistor load whose two ends are connected to the probe needles of oscilloscope. The voltage measured varies with the activity inside the board, which includes the FPGA chip operation.

### B. Host PC

There are two kinds of software running on the host PC. One is the MATLAB Instrument Control Toolbox [17]. The other is the Xilinx Embedded Development Kit (EDK). The MATLAB instrument control toolbox interacts directly with Tektronix oscilloscopes, enabling users to acquire and analyze data, graphically visualize data, make custom measurements, generate reports, and develop automated applications. This can all be done through the USB port between the host PC and the oscilloscope. MATLAB itself also can communicate data with other devices connected to host PC by the serial port. In our setup, MATLAB sends DES plaintext to the software controller on the FPGA via the serial port.

### C. FPGA Board

The FPGA board contains a Virtex-II Pro XC2VP30 FPGA, which includes two IBM PowerPC 405 processor cores [18]. The architecture of the PowerPC-based embedded system for use in running these experiments is shown in Fig. 5. In the EDK project, we divide the whole system into a hardware component and a software component. The hardware part runs in the FPGA logic while the software part runs on the PPC. The PPC swaps data with the FPGA hardware logic through the CoreConnect Processor Local Bus (PLB). The data swapped between the PPC and FPGA logic consists of three parts: the DES encryption key, the input plaintext for DES, and the output for each plaintext and key combination.

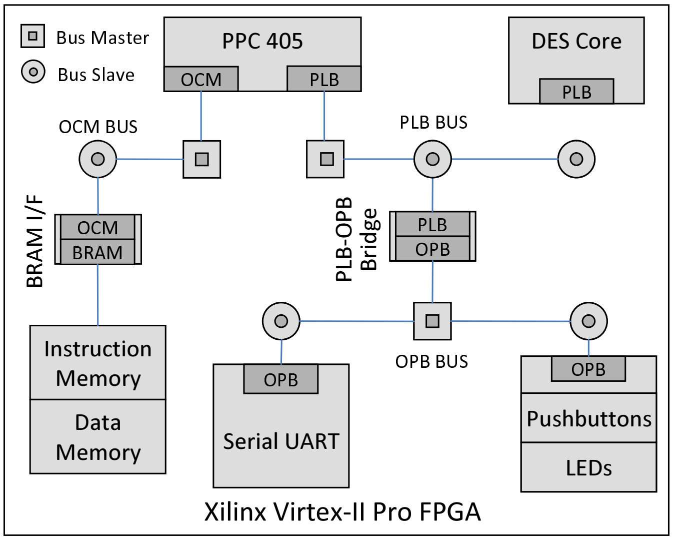


Fig. 5. Architecture of our target FPGA-based cryptographic system

### D. Comprehensive Operation

Upon system startup, the EDK project code is downloaded into the FPGA board, including both the hardware and software components. The software running on the PPC sits and waits to receive data from the MATLAB interface, while the hardware design running in the FPGA user logic is waiting to receive data from the PLB. The DES key information is embedded in the EDK code. More precisely, the 56-bit key is stored in the PPC software. Then it is extended and aligned to 64 bits and sent to the user logic via PLB.

After initializing all kinds of parameters, the MATLAB interface sends a plaintext value to the FPGA board via the serial port. Each plaintext is generated randomly. Although there exist alternative ways to obtain the plaintext for the known plaintext DPA attack, in our platform the randomly produced plaintext is used to simulate the real-world data encryption source.

The PPC receives the plaintext and transfers it to the user logic over the PLB. The DES core running in the user logic has both the key and plaintext. After one DES computation iteration, the ciphertext is made available at a shared register location. If necessary, the ciphertext can be read by the MATLAB interface through the serial port connecting them. What is more important is the power traces measured by the oscilloscope. In the MATLAB instrument control toolbox we wait for enough time (usually more than one DES cycle) to gather the data from the oscilloscope. We must synchronize the starting point of the power traces to the starting point of one DES iteration. The trick is that the rising edge of the trigger pulse can be aligned with the start of one DES round as shown in Fig. 6. Accordingly, the starting point of the DES transition power trace can be located at the rising edge of the trigger output.

The clock frequency of the Xilinx board is set at 100 MHz. The time to run one DES round is 10 nanoseconds. The sampling rate of the oscilloscope is 2.5G/s. Consequently there are 25 samples we can get from the oscilloscope for each round. Even though we only use such a small part of the sampled power trace data, it is on the order of megabytes of data for each sampling operation. Transferring MBs of data through the USB port can take several seconds. To make the attack less time-consuming, the PPC sends $m$ plaintext values to the DES module running in the FPGA logic. For example, if the plaintext value sent by the MATLAB interface is $p$, the plaintext values used by DES module in one iteration is $p$, $p$+1, $p$+2, $p$+3 for $m$ equal to 2. That is, the MATLAB instrument control toolbox can obtain 16*4 rounds of power trace data each iteration. For the large number of plaintext values required by differential power analysis, this can save quite a bit of attack time. After collecting all of the power trace data needed, they are saved by the MATLAB interface as a single large matrix variable. Each row in this matrix contains the plaintext and its corresponding power trace data. There are in total $N$ rows and $M$+1 columns of power trace data.

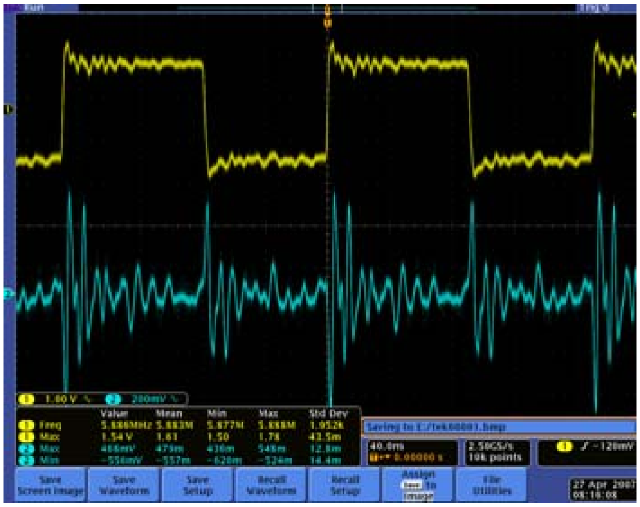
For each S-box, there are 64 possible input values. For each input value, we divide the power traces data into two groups according to the computed value of the first output bit. If the output bit value is 0, the corresponding power data is put in group $A$; otherwise, it is put in group $B$. Then we calculate the average difference value between the two the groups. In theory, as described in Section III, the average power difference graph with a spike corresponds to the correct subkey. However, in reality the average power difference cannot be absolutely zero because $N$ is not infinite. In practice a larger value of $N$ will give less error and noise in the output. In our experimental setup, $N$ is equal to ten thousand and $M$ is equal to 25 for each DES round. Figure 7 shows an average power difference for the first round of an S-box.

From this graph, we can see that the average power difference is very small and up to the micro-voltage scale. Thus, it is extremely difficult to find a spike in such a graph in reality. To avoid this problem, we adapted the original method to use the average value of the average difference instead to find the spike. The average value of the average difference is calculated and recorded with respect to the related plaintext. We call such an average value as the score of the input value. We select the input value with the maximum score value as the guessed part of the subkey. That is, the part of the subkey as the input of an S-box is computed as:

$$\max_{k=1,2,\ldots,64} \frac{1}{M}\sum_{j=1}^{M}(|\bar{P}_A^j - \bar{P}_B^j|) \qquad (5)$$

## V. Result Analysis

Figure 8 shows the score for the 64 input values. From this figure, one can recognize the maximum score. Unfortunately, however, the extracted key is not correct. In order to enlarge the chance of finding out the right data, we observe the spike figures for all 64 possible inputs, but still, no right key was found in any one of them.



|  | Value | Mean | Min | Max | Std Dev |
|---|---|---|---|---|---|
| 1 Freq | 5.886MHz | 5.883M | 5.877M | 5.888M | 1.952k |
| 1 Max | 1.54 V | 1.61 | 1.50 | 1.78 | 43.5m |
| 2 Max | 466mV | 479m | 436m | 548m | 12.8m |
| 2 Min | −556mV | −557m | −620m | −524m | 14.4m |

Saving to E:/Tek00001.bmp

Fig. 6. Power traces for four DES iterations in one trigger pulse period

After thoroughly checking our code, the focus of our concern switched to the board-level circuitry. Eventually, we discovered that one main reason our DPA attack failed was because there are a group of decoupling capacitors around the input of the FPGA internal power supply. Decoupling capacitors can effectively prevent the internal power supply from bouncing, which turns out to effectively mask the needed power leakage information. As a result, the likelihood of successfully mounting DPA attacks on this specific FPGA board are largely reduced with the capacitors in place. It is a general rule that the decoupling capacitors are essential in maintaining a stable-working high-performance FPGA circuit with signal and power integrity. In this way, the decoupling capacitor itself can be a very good preventative method against a DPA attacker who does not want to physically break the board.

Another factor that may affect the final result is the noise introduced by the functionalities, other than the DES core, which are also running on the FPGA. Due to the time complexity of collecting and post-processing the power data, Xilinx EDK was used to automate these steps. However, the EDK tool itself generates a large number of interface VHDL codes which are eventually configured into the FPGA board. Due to the fact that all of the programs are sharing the same internal logic power supply, these logic modules may also affect the power leakage traces. As a result, these two inherent obstacles prevented us from obtaining satisfactory results.

## VI. Conclusion

We have presented a platform to automatically perform DPA on a real-world FPGA board. This platform gives us a systematic view on how to successfully perform the DPA attack in a practical sense. The efficiency of analysis is critical to DPA if the attacker wants to break the FPGA cryptographic system. This requirement comes from two scenarios. Firstly,

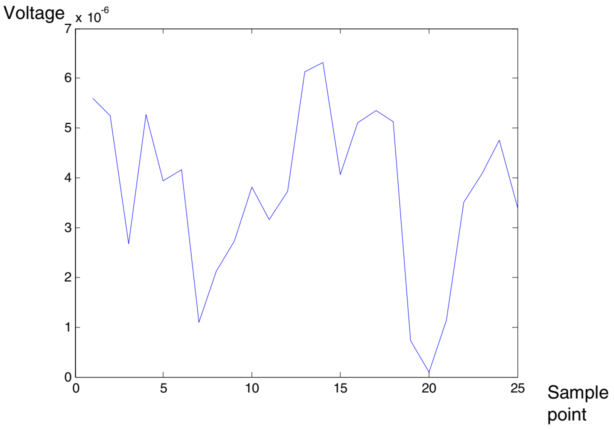


Fig. 7. Average power difference for the first round of an S-box

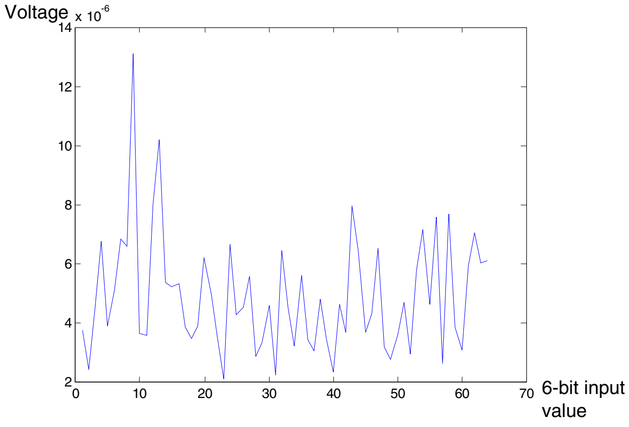


Fig. 8. Score for 64 input values of an S-box in DES

most modern cryptographic algorithms are based on the fact that they can be broken in theory, but not in practice. For example, it will take billions of years to break a 256-bit AES system in a brute-force search. Secondly, the key of cryptosystem like AES or DES is usually changed after a variable period of time. The new key is newly negotiated by the peers in a security protocol like IPSec. From this point of view, DPA must be able to destroy the cryptosystem in a limited time range.

Most previous work focuses on using a custom FPGA board as the target of DPA attack. While it is a great jump from theory to practice, commercial secure embedded systems should be the target for all researchers in this area. As the analysis demonstrated above, the experimental result shows that the DPA attacks are not as powerful as expected when facing the commercial FPGA platform due to the decoupling capacitors. The FPGA board must be physically broken before successfully applying DPA, making it no longer a passive attack. The decoupling capacitor is a natural countermeasure.

The last obstacle limiting DPA attacks in from practice is that there is more than one electrical device on an FPGA board. In our example, the on-chip PPC processor also participated in the power consumption. Other on-board components may overwhelm the power consumed by the reconfigurable logic. It is also likely that there would be more than one module concurrently running on the FPGA. All of this leads to a DPA attack being difficult and expensive to perform on this kind of system. How to solve these challenges in a demonstrable way is still a topic which deserves further research.

## References

[1] H. Bar-El, “Introduction to side channel attacks,” available at http://www.discretix.com, 2007.

[2] P. Kocher, J. Jaffe, and B. Jun, “Differential power analysis,” in *Proceedings of Advances in Cryptology (Crypto)*, Aug. 1999, pp. 388–397.

[3] J. Zambreno, D. Nguyen, and A. Choudhary, “Exploring area/delay tradeoffs in an AES FPGA implementation,” *Proceedings of the International Conference on Field-Programmable Logic and its Applications (FPL)*, pp. 575–585, Aug. 2004.

[4] Larry T. McDaniel III, “An investigation of differential power analysis attacks on FPGA-based encryption systems,” Master’s thesis, Virginia Polytechnic Institute and State University, 2003.

[5] S. B. Ors, E. Oswald, and B. Preneel, “Power-analysis attacks on an FPGA – first experimental results,” in *Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, Sep. 2007, pp. 35–50.

[6] F.-X. Standaert, S. B. Ors, J.-J. Quisquater, and B. Preneel, “Power analysis attacks against FPGA implementations of the DES,” in *Proceedings of the International Conference on Field-Programmable Logic and its Applications (FPL)*, Aug. 2004, pp. 84–94.

[7] C. Clavier, J.-S. Coron, and N. Dabbous, “Differential power analysis in the presence of hardware countermeasures,” in *Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2000, pp. 252–263.

[8] J.-S. Coron, “Resistance against differential power analysis for elliptic curve cryptosystems,” in *Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 1999, pp. 292–302.

[9] M.-L. Akkar and C. Giraud, “An implementation of DES and AES secure against some attacks,” in *Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2001, pp. 309–318.

[10] 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 *Proceedings of the European Solid-State Circuits Conference (ESSCIRC)*, Sep. 2002, pp. 403–406.

[11] K. Tiri and I. Verbauwhede, “Place and route for secure standard cell design,” in *Proceedings of the Smart Card Research and Advanced Application IFIP Conference (CARDIS)*, 2004, p. 143.

[12] ——, “A digital design flow for secure integrated circuits,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 7, pp. 1197–1208, Jul. 2006.

[13] S. B. Ors, F. Gurkaynak, E. Oswald, and B. Preneel1, “Power-analysis attack on an ASIC AES implementation,” in *Proceedings of the International Conference on Information Technology (ITCC)*, 2004.

[14] National Institute of Standards and Technology, “FIPS PUB 46-3, Data Encryption Standard (DES),” available at http://www.nist.gov, Oct. 1999.

[15] A. Schuster and E. Oswald, “Differential power analysis of an AES implementation,” Institute for Applied Information Processing and Communications, Graz University of Technology, Tech. Rep. IAIK-TR 2004/06/25, Jun. 2004.

[16] M. Joye1, P. Paillier, and B. Schoenmakers, “On second-order differential power analysis,” in *Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2005, pp. 293–308.

[17] The MathWorks, “Instrument control toolbox 2.4.2,” available at http://www.mathworks.com, 2007.

[18] Xilinx, “Virtex-II Pro family complete data sheet,” available at http://www.xilinx.com, 2007.