Vulnerability Analysis of a Soft Core Processor through Fine-grain Power Profiling

William Diehl[1], Abubakr Abdulgadir[2] and Jens-Peter Kaps[2]

[1] Virginia Tech, Blacksburg VA 24061, USA
[2] George Mason University, Fairfax VA 22030, USA
wdiehl@vt.edu, {aabdulga, jkaps}@gmu.edu

Abstract. Embedded microprocessors are an important component of reconfigurable architectures. Fine-grain (e.g., cycle-accurate) power analysis of such processors has been used to improve power and energy efficiency, and detect implementation vulnerabilities, in embedded applications. However, such analysis is difficult to conduct; it requires either specialized and often expensive equipment, or construction of test architectures using disparate acquisition and analysis tools. In this research, we expand the Flexible Open-source workBench fOr Side-channel analysis (FOBOS) to facilitate exact time-domain correlation of clock cycle and device state to power measurements, and to perform power analysis on a soft core processor. We first validate the fine-grain power analysis capabilities of FOBOS through cycle-accurate analysis of power consumption of AES encryption running on a soft core processor in the Spartan-6 FPGA. We then analyze the results in the context of Simple Power Analysis side-channel attacks, and confirm power correlation of certain instructions with Hamming Weight or Hamming Distance of secret key bytes. Finally, we show that an assumption of a pure Hamming Distance power model for load-to-register instructions is not sufficient for this embedded processor architecture, and that power models using both Hamming Distance and Hamming Weight should be considered for Differential Power Analysis.

Keywords: Cryptography, FPGA, microprocessor, side channel, DPA, SPA

1 Introduction

Embedded processors facilitate powerful combinations of hardware and software co-design in lightweight applications, while preserving the ability for rapid prototyping, incremental development, and flexible reconfiguration. They are especially relevant in architectures consisting of sensors and actuators in the Internet of Things (IoT), including automotive, home smart-appliance, energy smart-grid, logistics, and medical applications.

The focus of this research is on one particular configuration of embedded processors – the so-called "soft core" processor – which is often specified in a hardware description language (HDL), and is downloaded to an FPGA in a bitstream file. Soft core pro-

cessors allow inclusions of features that require flexibility in architecture, such as instruction set extensions (ISE) and hardware accelerators. Popular examples include Xilinx MicroBlaze, Intel Nios II, and RISC-V.

Since embedded processors typically operate under severe resource constraints with tight power and energy budgets, designers must investigate novel methods to achieve low power and energy consumption. Additionally, embedded systems with soft core processors often handle sensitive data which require cryptographic protections, such as those specified in Federal Information Processing Standards (FIPS). However, embedded processors often operate in physically insecure locations and are vulnerable to side channel attacks, where an attacker is able to observe physical phenomena associated with device operation (such as power fluctuations or radio frequency emissions), and deduce the contents of sensitive variables.

In the above cases, it is important to be able to analyze processor power consumption during operation. In particular, we seek fine-grain power analysis at short time intervals, e.g., cycle-accurate analysis. Cycle-accurate power analysis facilitates optimizing algorithms, programs, and architectures for power and energy consumption, and helps to identify security vulnerabilities in certain instructions or conditions, which could result in leakage of information through power analysis side channel attacks. However, cycle-accurate analysis is not trivial; it is either performed on expensive commercial equipment outside the reach of most academic institutions, is performed using complex customized installations that are not easily replicated, or is not performed to the level of fidelity necessary to comprehend the data.

In this research, we apply an extension of the Flexible Open-source workBench fOr Side-channel analysis (FOBOS), called the FOBOS Profiler, to analyze power analysis vulnerabilities of a cryptographic application running on a soft core processor. FOBOS Profiler enables fine-grain power analysis by correlating time-domain power acquisition (i.e., samples) to exact corresponding clock cycle and device state. The Profiler is part of the single "acquisition to analysis" FOBOS model, with flexible open-source software and firmware available to entry-level power analysis investigators [1].

We first validate the ability of FOBOS Profiler to analyze a large group of native soft core instructions in an iterative routine, and then analyze vulnerabilities to power analysis side channel attacks due to Hamming Weight and Hamming Distance dependencies of certain instructions. We target an AES (Advanced Encryption Standard) encryption application running on a custom-designed very lightweight 8-bit soft core processor in the Spartan-6 FPGA.

Our contributions in this work are twofold: 1) We validate a methodology based on an open-source test bench for analyzing vulnerabilities of microprocessor instructions to power analysis side channel attacks, and 2) We confirm correlation of certain side channel attack vectors on Hamming Weight or Hamming Distance, and show that some power models should incorporate both Hamming Weight and Hamming Distance in order to better reflect dependencies on sensitive data.

2 Background

For many decades, researchers have sought fine-grain (e.g., instruction- or cycle-level) granularity for measurement of power consumption, to optimize for power and energy consumption, to design better hardware and instruction set architectures (ISA), and to analyze potential security vulnerabilities. Given the wide-spread use of soft core processors in reconfigurable applications, there have been many attempts to determine their cycle-accurate power consumption in order to improve efficiency. For example, in [2], the authors studied the effects of including instruction set extensions (ISE) in an FPGA implementation of a MicroBlaze soft core processor, and in [3], the authors investigated power consumption in soft core multiprocessor applications through the use of a simulation tool using power models described in SystemC. In these cases, processor power measurements were performed in simulation, but not in actual hardware.

Cycle-accurate power analysis can also be used to determine security vulnerabilities in physical cryptographic implementations, which can leak information through power analysis side channel attacks. Examples of these attacks include Simple Power Analysis (SPA) and Differential Power Analysis (DPA). In SPA, the attacker observes the amplitude of power spikes associated with operations or sensitive variables, and draws conclusions on the type of operation or value of sensitive operands. For example, in [4], the authors attack smart card implementations of microprocessors by noting two types of correlations: 1) Hamming Weight (HW), based on the number of bits set to 1 in a given operation, and 2) Hamming Distance (HD) (or "transition counts"), based on the number of bits which change during an operation. The authors note that knowing all HW of key bytes reduces key search space, e.g., from 2^{56} to 2^{40} for DES.

In the above attacks, identification of sensitive values requires directly associating HW or HD to the amplitude of a sensitive operand. In fact, the requirements for such an attack are 1) access to the power consumption of the device in operation; 2) the ability to identify individual clock cycles; and 3) a direct relation between power and HW or HD [5]. While theoretically this requires only a few power traces, in practice SPA is difficult, because the differences in dynamic power corresponding to different HW or HD are small, and getting smaller with reduction of technology feature size and V_{CC}. Therefore, the trend for power analysis attacks over the last decade has been toward Differential Power Analysis (DPA).

In DPA, described in [6], the attacker measures small power changes that take place at a targeted attack point based on changes in known and unknown variables. A power model is used to hypothesize the contents of the sensitive variable, and multiple power observations (e.g., traces) are collected during cipher operation to statistically correlate secret key bytes. DPA is more powerful than SPA, because an attacker 1) does not need to know as much about architecture, and 2) using statistical properties, can overcome reduced signal-to-noise ratio (SNR) [4]. However, developing a correct power model is not trivial, and often requires extended trial-and-error to develop the correct model.

In fact, fine-grain power analysis can assist in conducting more efficient DPA attacks, by 1) extracting feature points on which to focus DPA attacks, and 2) determining or verifying the dependence of a particular attack point on a particular power model,

e.g., HW or HD [7]. One example of this type of analysis is [8], where the authors attempt to automatically insert DPA countermeasures into microcontroller software implementations of cryptographic algorithms. Their procedure begins with reconnaissance of assembly instructions most vulnerable to DPA by analyzing cycle-level power and computing measures of information leakage. However, their vulnerability metrics are based on mutual information analysis, and are not designed to demonstrate HW or HD dependencies.

Additionally, in [9], the authors measure instruction-level power differences in Intel x86 architecture processors with the goal of discovering the instructions with the greatest side channel vulnerabilities. However, their methods are designed to detect vulnerabilities due to non-constant time execution; they are not suitable for identifying HW or HD power analysis side-channel vulnerabilities.

In summary, given the large variety of embedded processors, the ability to conduct fine-grain power analysis during design and prototyping is paramount to the achievement of secure and efficient reconfigurable implementations.

3 Methodology

3.1 FOBOS

In this research, we employ the Flexible Open-source Board fOr Side-channel analysis (FOBOS) to conduct fine-grain power measurements. FOBOS is a side-channel analysis (SCA) platform which provides a comprehensive SCA toolset from trace acquisition to analysis [1]. It uses commercial off-the-shelf FPGA boards (e.g., Digilent Nexys-3) to lower cost which is crucial for educational uses. The software is released as open-source and written in Python to facilitate portability to different operating systems.

3.2 FOBOS Hardware

As shown in Fig. 1, FOBOS is composed of two FPGA boards. The control board receives test vectors from the control PC, stores them temporarily in FIFOs, and forwards data to the DUT (device under test) board, on which the victim implementation is instantiated. It also triggers an oscilloscope to start capturing the power consumption of the DUT.

The DUT board includes a wrapper to handle communication with the control board. This wrapper stores received data into different FIFOs depending on the data type. The DUT wrapper in this research is customized to interface with the soft core processor (described below).

Randomly-generated test vectors are stored in `dinFile.txt` (in the host PC), and fed through the `dinFIFO` to soft core data RAM through the soft core `extdin` interface. The user program, pre-stored in the FOBOS wrapper, is loaded on cue into program RAM through the `extprogin` interface. Upon completion of processor operations, data RAM contents are dumped through the `extdout` interface, through `doutFIFO` and are collected in `doutFile.txt` in the host PC.

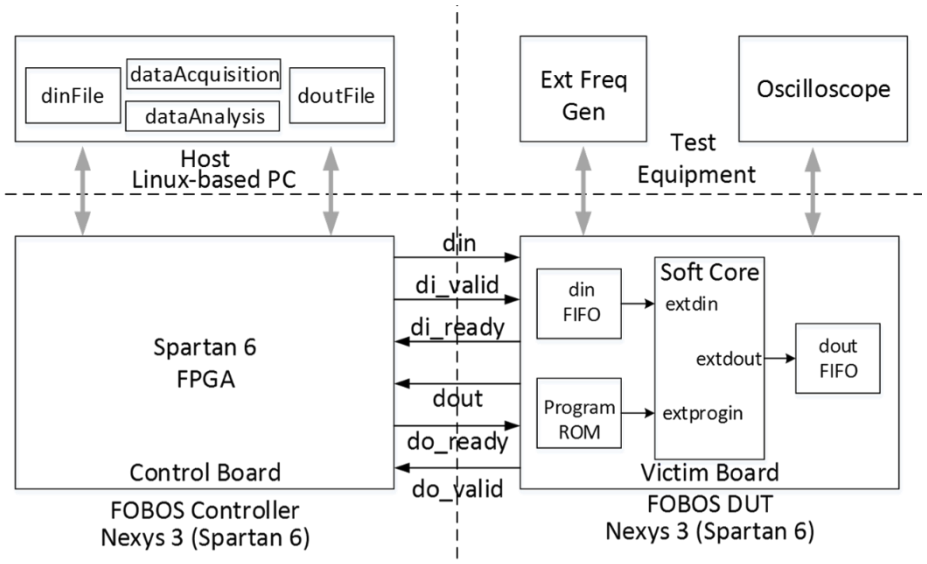

Fig. 1. FOBOS Hardware Architecture

3.3 FOBOS Software

FOBOS software is composed of two major modules – the acquisition module and the analysis module. The PC-based acquisition module reads test vectors from a file and sends one vector at a time to the control board, which forwards it to the DUT. The DUT runs the cryptographic operation and returns the result to the control board which returns it the PC-based acquisition module. The power traces are collected from the oscilloscope and saved in a file. Software scripts in the analysis module can be used to mount SCA on the traces collected by the acquisition module (e.g., Correlation Power Analysis), or to conduct fine-grain power analysis using FOBOS Profiler.

3.4 FOBOS Power Measurement

FOBOS measures device power consumed by the Spartan-6 1.2V bus, e.g., V_{CCINT}, by measuring voltage across a 1Ω shunt resistor. Voltage is amplified by a TI INA225 amplifier, recorded in an oscilloscope, and offloaded to the attached PC for post-run power computation. Power measurements are recorded at discrete time intervals corresponding to sample rate. Our current FOBOS installation uses the Agilent (Keysight) Technologies DSO6054A Oscilloscope, with a 4 Gsa/s sampling rate, and computes about 20,000 samples per trace. 100 test vectors, consisting of random input data generated in software, are used to generate power traces

3.5 FOBOS Profiler

We require the ability to map power trace samples to the exact clock cycles, in order to know state of the DUT at any specific sample. "FOBOS Profiler" is a set of scripts that map specific time domain events (e.g., clock cycle) to the power samples in traces collected from the oscilloscope. This enables the ability to: 1) Determine the exact clock cycle for information leakage (e.g., t-test); 2) Relate power consumption spikes to DUT state, executed instruction, or data being processed; and 3) Cycle-accurate power profiling.

Once the DUT starts a cryptographic operation, it deasserts a specific signal `di_ready` which the control board detects. The control board triggers the oscilloscope immediately when receiving this signal, or after this event by any number of clock cycles. The trigger signal is kept high for a user-specified number of clock cycles, or until the DUT finishes its operation. These trigger features are necessary to provide the reference point for mapping the clock number to time (sample number) in the trace. The acquisition module truncates the trace and aligns it in the time domain using the trigger signal as a reference.

Users generate a file that maps the internal state of the DUT to clock cycles. This "state file" can be generated by modifying an HDL test bench to write a value corresponding to each state to a file, or by an external simulation utility. The profiler script uses the state file and the power traces, and reports (in graphic or textual format) clock transitions and device states at any clock cycle.

FOBOS Profiler has been used to pin-point sources of remaining information leakage in an FPGA implementation of an authenticated cipher protected with DPA countermeasures [10], and publicly demonstrated in [11]. In this research, we adapt this tool to perform cycle-accurate power analysis of a soft core processor.

3.6 Custom soft core microprocessor

We demonstrate cycle-accurate power analysis on a custom-designed 8-bit reconfigurable microprocessor. This lightweight Reduced Instruction Set Computer (RISC) has only 30 native instructions, has separate 8-bit instruction and data buses in a Harvard Architecture, and is a pure load-store architecture. Most instructions execute in one clock cycle, however, some ALU instructions execute in two cycles. The ALU is designed for the easy insertion of user-defined ALU instructions that require no modification of the finite state machine logic. This feature is used to implement the instruction set extensions necessary for single-cycle Galois Field (GF) multipliers `gf2` and `gf3` instructions) used by the AES block cipher in this research.

The processor has optimizations for cryptographic block ciphers, such as single-cycle increment (`inc`), decrement (`dec`), exclusive or (`xor`), and table substitutions (`trf`, e.g., for an S-Box substitution). The user can instantiate up to 4K of program RAM, 64K of data RAM, and 1K of table ROM (i.e., four 256-byte look up tables). Memory is implemented as distributed RAM, and is instantiated at synthesis time. The amount of memory instantiated, based on the required number of bytes (req_bytes), is computed as $2^{\lceil log_2 req\_bytes \rceil}$.

This processor is evaluated in [12] and documented at [13].

3.7 AES implementation

AES (Advanced Encryption Standard) is the U.S. federal and defacto worldwide standard for symmetric block ciphers as defined in [14]. AES-128 uses a 128-bit key, encrypts (or decrypts) 128-bit blocks of plaintext (or ciphertext), and consists of 10 rounds. There are four transformations which occur on a state defined as a 4×4 matrix of bytes.

The SubBytes transformation introduces non-linearity through a one-to-one byte substitution. SubBytes is often rendered as 16 8-bit S-Boxes which can be implemented in look-up tables. The ShiftRows transformation performs a permutation on the state word, where the ith row is rotated left by i bytes for rows 0 through 3. The MixColumns transformation is equivalent to multiplication of each column of the state by a 4×4 matrix of constants in GF(2^8). The MixColumns transformation is skipped in the final round. In the AddRoundKey transformation, a 128-bit round key is added to the state by a bitwise `xor` operation.

We use the AES encryption application as described in [12] and available at [13], which is written in native assembly code for this custom processor, and computes round keys on-the-fly. It uses only 396 program bytes, 56 data bytes, and 256 table bytes, and therefore fits into a soft core instantiated with 512 bytes of Program RAM, 64 bytes of Data RAM, and 256 bytes of Table ROM. The soft core processor using the above resources, when implemented in the Spartan-6 (xc6slx16csg324-3) FPGA, requires 100 combinational logic block (CLB) slices, 337 look up tables (LUTs), 75 flip flops, and has a maximum frequency of 103.6 MHz.

The iterative design of this AES implementation is shown in Fig. 2. Cycle-accurate power analysis (discussed in "Results") is performed on rounds one through nine, and Hamming Weight and Distance analysis is performed on the prewhitening stage.

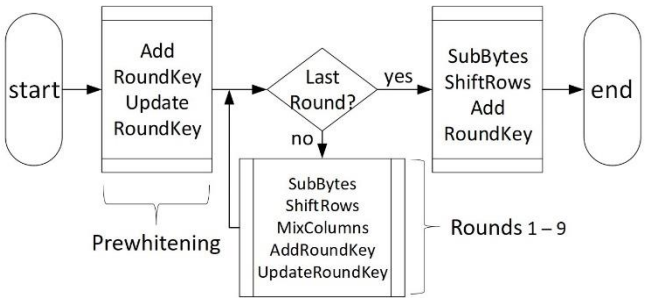

Fig. 2. AES Encryption with Iterative Round Structure

4 Results

4.1 Cycle-accurate analysis of AES rounds

This implementation of AES requires 14,329 clock cycles. Power analysis is conducted on nine rounds consisting of 1452 clock cycles each. The first nine rounds of this implementation are identical; the 10th round lacks a MixColumns transformation, and is omitted. Power samples are gathered at 20 MHz externally generated clock frequency at an ambient temperature of 22.5 °C. Analysis is conducted on 100 traces using randomly-generated key and plaintext, with 18,151 samples per trace, for an average of 12.5 samples per clock cycle.

Results for all instructions used in rounds 1 - 9 are shown in Table 1, where "(2)" denotes the second cycle of a two-cycle instruction. The average intra-round and inter-round variations are 1.4% and 0.13% of average mean power, respectively. This shows that power fluctuations due to changing conditions are very small, but that conditions are remarkably stable from round-to-round, confirming minimal error and excellent trace-to-trace sample alignment.

Table 1. Mean power and standard deviation of AES instructions.

Mean Power				Standard Deviation			
Instr	mW	Instr	mW	Instr	mW	Instr	mW
gf2	10.015	jmp(2)	9.874	jsr(2)	0.230	jmp	0.126
gf2(2)	10.008	lds	9.873	jsr	0.212	jmp(2)	0.120
gf3	9.949	mvi(2)	9.864	mvi(2)	0.190	bzi	0.114
gf3(2)	9.932	inc	9.861	mvi	0.187	ret	0.110
dec	9.894	mvi	9.861	ret(2)	0.186	bzi(2)	0.110
bzi	9.885	ret(2)	9.848	lds	0.176	trf	0.102
jmp	9.882	ret	9.835	inc	0.167	gf3(2)	0.090
xor	9.880	trf	9.795	sts	0.157	gf3	0.072
sts	9.880	jsr	9.744	xor	0.141	gf2	0.071
bzi(2)	9.879	jsr(2)	9.736	dec	0.136	gf2(2)	0.061

Results show that the ALU instructions for Galois Field multiplication, `gf2` and `gf3`, use the highest power. This confirms previous results (e.g., [15]) that ALU instructions tend to use higher power, with multiplication in particular being a high power consumer. The `dec` instruction uses more power than `inc`, and loads (i.e., `lds`) use less power than stores (i.e., `sts`) which also mirrors results in [15]. The `bzi` (branch on zero immediate) instruction uses higher power than `jmp`, despite similar construction, due to additional decision logic (in this architecture, comparisons occur and condition flags are set during arithmetic operations).

One unexpected result is that `jsr` (jump to subroutine) uses, on average, less power than `jmp` (branch unconditional). Intuitively, we expect `jsr` to use more power, since this instruction requires storing a return address on the stack, decrementing the stack pointer `sp`), and updating the program counter (`pc`), whereas `jmp` only requires updat-

ing pc. However, jsr has the highest intra-round standard deviation among instructions, and there is a 700 μW difference (7% of the average power of jsr – 9.744 mW) among the 12 occurrences of jsr in each round. Seven of the jsr instances have higher power than the mean power of jmp, so it remains to explain why some jsr instances have much lower power. An analysis of HW of the target pc shows that the higher the HW, the higher the power. Within groups of the same HW, there are also differences of sp, where sp=0xFF appears to draw more power than sp=0xFD. Additionally, within groups of the same HW of target pc, a preceding instruction of ret(2) (return from subroutine – 2nd cycle) appears to draw more power than a preceding sts instruction. An analysis of control signals in the finite state machine (FSM) controller shows that 12 control bits toggle from ret(2) to jsr, while only ten control bits flip from sts to jsr. In summary, multiple influences combine to determine instruction power, and one must examine the exact architecture of control logic in order to determine the influence of each component; not just register and memory writes in the datapath.

In terms of intra-round standard deviation, average powers of data loads and stores (i.e., lds and sts) vary more widely than average powers of arithmetic instructions. In particular, gf2 and gf3 have the lowest standard deviations. This supports the conclusions of [8], that loads and stores have higher vulnerabilities to side channel attack than ALU instructions. However, where as many sources in literature target substitution boxes (S-Boxes) of block ciphers as attack points, we note that the trf (transformation) instruction, which computes a one-to-one eight-bit mapping (such as an AES S-Box), has one of the lowest average powers and lowest intra-round standard deviations. This infers a higher difficulty of using the S-Box as DPA attack point in this FPGA architecture and implementation.

4.2 Analysis of SPA based on Hamming Weight (HW)

We next determine the feasibility of using cycle-accurate power analysis to conduct a side channel attack using SPA. In this research, we analyze the xor instruction, which occurs during the AddRoundKey transformation in the first round. This is often called a "prewhitening" transformation, and uses the original secret key bytes k_i prior to generation of round keys. By selecting a chosen plaintext of 0^{128}, we can analyze the instruction xor r2, r3, which translates to $r_3 = r_2 \oplus r_3$, where $r_2 = 0$ and $r_3 = k_{i\in\{0,1,\ldots,15\}}$. Our motivation for conducting an SPA attack based on HW of secret key bytes is discussed in [4,5], where recovery of HW can reduce key search space from 2^{128} to 2^{90}.

Results in Table 1 show that xor has a relatively high average power and intra-round variation. Additionally, analysis in [16] showed that xor power consumption has a near linear variation with HW. In order to conduct a 1st round SPA attack on AES using the relation $HW(0 \oplus k_i) = HW(k_i)$, we first determine if an xor dependence on HW holds for this implementation on this FPGA. We construct a loop to cycle through all $2^8 = 256$ possible values of a key byte k_i accessed by xor, and measure resulting power consumption over 100 averaged traces with identical operands. Results

of HW analysis of the last 64 bytes 0xC0 to 0xFF, shown in Fig. 3, indicate a 94% correlation of measured average power with theoretical HW.

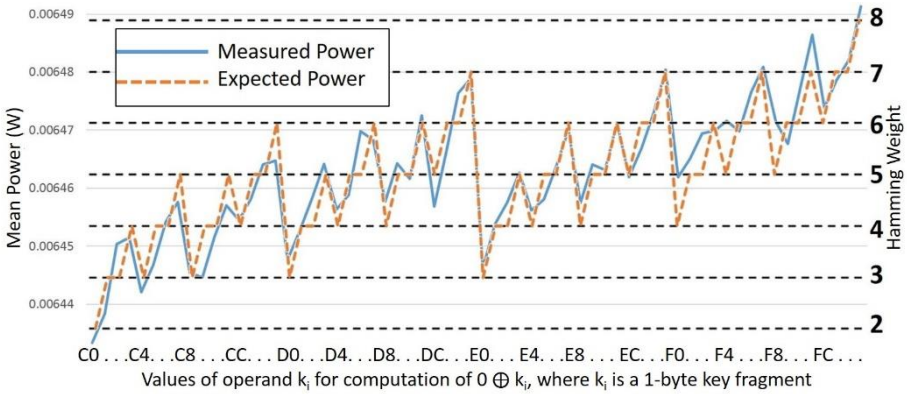

Fig. 3. Correlation of measured power at 1 MHz clock frequency (solid) to Hamming Weight (HW) of operand (dashed) for `xor` instruction. Mean power depicted on left (in Watts); HW of operand is on right.

However, attempts to recover actual HW of unknown key bytes are complicated by non-constant variables and noise sources. For example, during each execution of `xor`, `pc` is updated as $pc = pc + 1$. The differences of power due to addition and register storage create conditions which are difficult to analyze. Additional sources of noise include fluctuations of temperature and air flow over the FPGA, radio frequency noise, amplifier distortion in the FOBOS architecture (at low frequencies), and attenuation of high frequency signals by the RCL network in the victim FPGA itself (at high frequencies).

Attempts to recover two key strings, Test Vector (TV) TV1: {0xEB, 0x97, 0xC4, 0xA0, 0x92, 0xB5, 0xA7, 0xF1}, and TV2: {0x01, 0x03, 0x07, 0x0F, 0x1F, 0x3F, 0x7F, 0xFF}, are shown in Fig. 4 left and right, respectively. TV1 is arbitrarily chosen, and TV2 is in order of increasing HW. To compensate for the variance in power due to `pc` update, we first measure reference test vectors with minimum HW (0^{128}) (i.e., the bottom lines), and maximum HW (1^{128}) (i.e., the top lines), and then compare to the test vector (TV) consisting of secret key bytes (i.e., the middle lines). Results show close correlation of HW to corrected average power levels for each instance of `xor`, however, differences in HW amplitude levels are only about 10 μW (measured at 1 MHz with 200 V/V amplification). In these two test vectors, results are often off by one or more HW levels. As such, reliably identifying HW of secret key bytes on this architecture at this frequency remains challenging.

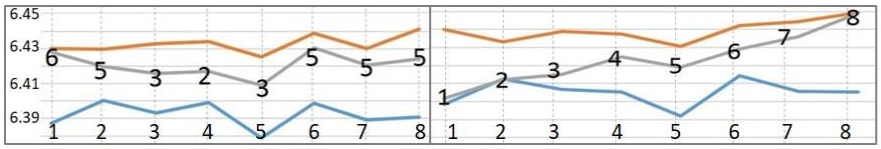

Fig. 4. Measured power of minimum HW reference (bottom), maximum HW reference (top), and test vector (TV) with actual HW of each byte (middle); TV1 shown at left, and TV2 shown at right. Power in mW is shown on y-axis.

4.3 Analysis of SPA based on Hamming Distance (HD)

There are other instructions on which an SPA attack could be conducted. One example is `lds r1, r3` (load to register) which occurs immediately before `xor r2, r3`. Here, `r1` is the index to consecutive memory addresses in which key bytes are stored, and `r3` is the register into which successive key bytes are loaded. As discussed in [4,7,17], power consumption of data loads from memory to registers typically varies with HD. If we could combine a power correlation based on $HD(r_{3'}, r_3)$, where $r_{3'}$ is the previous register contents, and $r_3 = k_{i \in \{0,1,\ldots,15\}}$, with a correlation based on $HW(k_{i\in\{0,1,\ldots,15\}})$, we could further reduce key search space for a brute-force attack. In fact, we perform Monte Carlo simulations to show that key search space is reduced from 2^{90} to 2^{70}. While a search space of 2^{70} is still computationally intensive, it is far less than the minimum attack resistance strength of 2^{112} recommended in [18]. As before, our first task is to demonstrate a HD correlation for register loads.

To show an `lds` dependence on HD, we loop through all 256 possible operands and measure resulting power. While we note a distinct HD correlation, power measurements are biased in a linearly increasing trajectory; the amplitudes of differences between maximum HD and minimum HD are clearly distinguishable, but increase linearly with each instruction occurrence. We hypothesize that the bias is proportional to a HW of at least one operand, and generate a corrected set of power results by subtracting the differential bias equal to $(P_{maxHW} - P_{minHW})/8 * HW(index)$, where "index" is the memory address index in `r1`. The results for 64 indices 0xC0 to 0xFF shown in Fig. 5 confirm a 94% correlation between measured average power and $HD(r_{3'}, r_3)$.

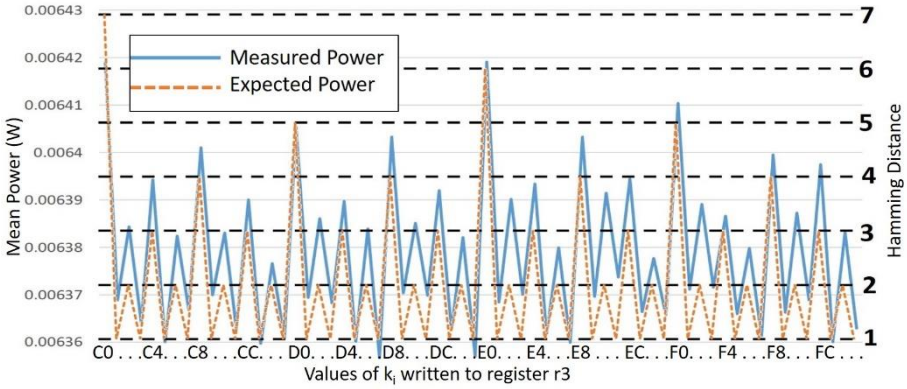

Fig. 5. Correlation of corrected power measurements of `lds` instruction (solid) to Hamming Distances $HD(r_{3'}, r_3)$ (dashed) at 1 MHz clock frequency. Values of k_i written to r_3 by `lds r1, r3` are shown on the x-axis, where previous contents of r_3 ($r_{3'}$) are computed as $r_{3'} = r_3 - 1$.

It remains to confirm the source of the HW bias. One might assume that the HW bias is proportional to the value being loaded from memory to the register `r3`. To test this hypothesis, we measure power on test vectors where memory contains all zeroes. The results, shown in Fig. 6, refute this hypothesis, since the HW bias clearly remains. In fact, the bias is proportional to the value in the index register `r1`, which is incremented at each iteration. This supports the observations of [4], where relatively large power fluctuations are observed depending on values asserted on memory address buses. Our attempts to remove the HW bias are likewise depicted in Fig. 6, and show that, even after applying a linear bias correction, a significant source of noise remains in corrected power measurements that hampers our ability to form HD correlations based on secret key bytes. The removal of HW bias could be improved through exact modeling of this bias (including high-order polynomial or non-linear fittings), filtering using signal processing techniques, and experimentation with different frequency and amplification settings.

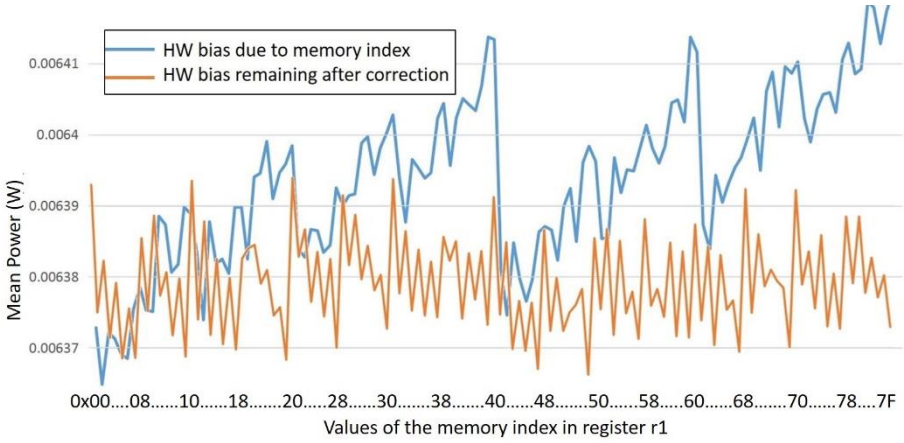

Fig. 6. Power measurements for `lds r1, r3` for r3 operands with zero Hamming Distance. Hamming Weight (HW) bias due to memory index in r_1 shown in top line, and attempted removal of HW bias shown in bottom line.

Thus the power consumed by the load-to-register instruction *depends on both HD and HW*. This has important implications for development of power models used in DPA attacks. Many documented DPA attacks (a recent example is [19]) assume a purely HD power model for attacks targeting registers. However, our analysis hypothesizes that a model employing both HW and HD, e.g., $f(HW(r_1), HD(r_{3'}, r_3))$, could recover a secret key using fewer traces in a DPA attack; this hypothesis will be evaluated in future work.

5 Conclusion

In this research we demonstrated the fine-grain power analysis capabilities of the Flexible Open-source workBench fOr Side-channel analysis (FOBOS) Profiler through analysis of AES encryption software running on a custom soft core processor in a Spartan-6 FPGA. Through cycle-accurate power analysis of AES rounds, we demonstrated excellent round-to-round power measurement stability and trace-to-trace sample alignment. We also confirmed the relatively high intra-round power variations of loads and stores, likely due to data dependence, which suggests the construction of power analysis attacks targeted at these instructions. We showed that arithmetic instructions (particularly multiplications) use the most power, and that loads from memory to register use less power than stores from register to memory. In terms of intra-round variation, we confirmed that loads and stores have greater power variation than arithmetic operations.

Unexpectedly, we observed that unconditional branches used more power than jump-to-subroutines, and presented hypotheses to explain this behavior.

In the context of planning for SPA side-channel attacks on AES, we showed that an `xor` instruction in the prewhitening AddRoundKey subroutine has a 94% correlation with the Hamming Weight of the corresponding secret key byte, but that low signal-to-noise ratio makes a direct key recovery difficult. We additionally showed that the load-

to-register instruction has a 94% correlation to Hamming Distance, but that it is also dependent on the Hamming Weight of the addressed memory location. This impacts planning for DPA attacks on similar architectures, in that attackers should consider both Hamming Weight and Hamming Distance in power models targeting register writes, in order to reduce the number of traces required to recover a secret key.

References

1. CERG, "Flexible Open-source workBench fOr Side-channel analysis (FOBOS)", Oct. 2016, Internet: https://cryptography.gmu.edu/fobos/ [Accessed on Feb. 9, 2019].
2. P. Biswas, S. Banerjee, N. Dutt, P. Ienne and L. Pozzi, "Performance and Energy Benefits of Instruction Set Extensions in an FPGA Soft Core", VLSID'06, Jan. 2006.
3. Z. El Hariti, A. Alali and M. Sadik, "Virtual Platform for Modeling the Power Consumption of a Soft-core Processor by the SystemC/TLM", 2018 4th International Conference on Optimization and Applications (ICOA), Apr. 2018, pp 1 – 4.
4. T. Messerges, E. Dabbish and R. Sloan, "Examining Smart-card Security Under the Threat of Power Analysis Attacks", *IEEE TCOM*, vol. 51, May, 2002, pp. 541 – 552.
5. L. Xiao and H. Heys, "A Simple Power Analysis Attack against the Key Schedule of the Camellia Block Cipher", *Information Processing Letters*, vol. 95, 2005, pp. 409 – 412.
6. P. Kocher, J. Jaffe, B. Jun and P. Rohatgi, "Introduction to Differential Power Analysis", *Journal of Cryptographic Engineering*, vol. 1, Apr. 2011, pp. 5 –27.
7. D. Roy and D. Mukhopadhyay, "Security of Crypto IP Core: Issues and Countermeasures", *Fundamentals of IP and SoC Security: Design, Verification, and Debug*, 2017, pp. 67 –114.
8. A. Bayrak, F. Regazzoni, P. Brisk, F. X. Standaert and P. Ienne, "A First Step Towards Automatic Application of Power Analysis Countermeasures", *IEEE DAC*, Jun. 2011, pp. 230 – 235.
9. R. Callan, A. Zajic and M. Prvulovic, "A Practical Methodology for Measuring the Side-Channel Signal Available to the Attacker for Instruction-Level Events", 47th Annual IEEE/ACM International Symposium on Microarchitecture, Dec. 2014, pp. 242 – 254.
10. W. Diehl, F. Farahmand, A. Abdulgadir, J.P. Kaps and K. gaj, "Fixing the CLOC with Fine-grain Leakage Analysis", ASHES 2018, Oct. 2018, pp. 75 – 80.
11. A. Abdulgadir, W. Diehl, R. Velegalati and J.P. Kaps, "Flexible, Opensource workBench fOr Side-channel analysis (FOBOS)," IEEE HOST 2018 Hardware Demo, May 2018, Internet: http://www.hostsymposium.org/host2018/hwdemo/HOST_2017_hwdemo_5.pdf [Accessed on Feb. 9, 2019].
12. W. Diehl, F. Farahmand, P. Yalla, J. P. Kaps and K. Gaj, "Comparison of Hardware and Software Implementations of Selected Lightweight Block Ciphers", 27th International Conference on Field Programmable Logic and Applications (FPL), Sep. 2017, pp. 1 – 4.
13. W. Diehl, "SoftCore", Dec. 2016, Internet: https://github.com/willja001/SoftCore [Accessed Feb. 9, 2019].
14. Federal Information Processing Standards Publication 197, "Advanced Encryption Standard (AES)", 2001.
15. C. Cernazanu-Glavan, M. Marcu, A. Amaricai, S. Fedeac, M. Ghenea, Z. Wang, A. Chattopadhyay, J. Weinstock and R. Leupers, "Direct FPGA-based Power Profiling for a RISC Processor", IEEE I2MTC, May. 2015, pp. 1578 – 1583.
16. J. Park, H. Lee, J. Ha, Y. Choi, H. Kim and S. Moon, "A Differential Power Analysis Attack of Block Cipher based on the Hamming Weight of Internal Operation Unit", International Conference on Computational Intelligence and Security, vol. 2, Nov. 2006, pp. 1375 – 1380.

17. A. Heuser, S. Picek, S. Guilley and N. Mentens, "Lightweight Ciphers and their Side-channel Resilience", *IEEE Transactions on Computers*, 2017.
18. National Institute of Standards and Technology, "Submission Requirements and Evaluation Criteria for the Lightweight Cryptography Standardization Process", Aug. 2018.
19. L. Mazur and M. Novotny, "Differential power analysis on FPGA board: Boundaries of Success", 6th Mediterranean Conference on Embedded Computing (MECO), Jun. 2017, pp. 1 – 4.