Constant Power Reconfigurable Computing

Adrien Le Masle [1], Gary C. T. Chow [2], Wayne Luk [3]

Department of Computing, Imperial College London
180 Queen's Gate, London SW7 2BZ, UK
{[1]al1108,[2]cchow,[3]wl}@doc.ic.ac.uk

***Abstract*—We present Constant Power Reconfigurable Computing, a general and device-independent framework based on a closed-loop control system used to keep the power consumption constant for any reconfigurable computing design targeting FPGA implementation. We develop an on-chip power consumer, an on-chip power monitor and a proportional-integral-derivative controller with circuit primitives available in most commercial FPGAs. We demonstrate the effectiveness of the proposed methodology on a square-and-multiply exponentiation circuit implemented on a Spartan-6 LX45 FPGA board. By reducing the peak autocorrelation values by a factor of 2.7 on average, the proposed Constant Power Reconfigurable Computing approach decreases the information leaked by the power consumption of this system with only 26% area overhead and 28% power overhead.**

I. Introduction

Encryption algorithms are designed to make brute-force attacks or exhaustive key search computationally infeasible and to resist cryptanalysis based on theoretical weaknesses of the algorithm. However, the physical implementation of an encryption algorithms can leak information and create security flaws. Attacks exploiting these physical flaws are called side channel attacks.

Since their initial publication [1], a relevant type of side channel attacks called power attacks have been extensively studied. Power attacks are successfully demonstrated on many common encryption methods, including private key encryption methods such as DES [2] and AES [3], finite field based public key methods such as RSA [4] and Diffie-Hellman, and elliptic curve based public key encryption [5]. Theoretically, power attacks can be used to attack any cryptosystem with a key-dependent power consumption. They are of two types. Simple power analysis (SPA) gains direct information about the encryption key by looking at a single power trace. Differential power analysis (DPA) extracts information from multiple power traces using statistical methods. The keys are usually broken when enough power traces have been collected.

Field Programmable Gate Arrays (FPGAs) are suitable platforms for implementing cryptographic algorithms in particular, and computationally demanding applications in general – this field is known as reconfigurable computing. First, the structure of FPGAs makes them particularly fit for pipelined applications, which is the case for most of the basic cryptographic operations. Second, FPGAs can be used to embed security into low power environments keeping very good performance. Finally, a pure hardware implementation of a cryptographic algorithm is inherently less vulnerable than its software counterparts which are usually run in a multi-tasking operating system. However, without adopting suitable countermeasures, an FPGA implementation is as vulnerable to power attacks as its software counterparts running on a processor. As a matter of fact, the transistors switching inside the device can leak information about the operations performed.

Two different solutions exist in order to make an FPGA implementation resistant to power attacks. The first solution involves modifying the hardware implementation of the algorithm so that it becomes harder for an attacker to extract useful information out of its power trace. This solution is application-dependent and is often implemented at the expense of slowing down the system. The second solution involves making the FPGA itself resistant to attacks. This is often achieved through adapting placement and routing methods which can result in consuming larger area.

In this paper, we present a novel framework called Constant Power Reconfigurable Computing. Our main contributions are:

- A general application-independent and device-independent framework based on a closed-loop control system used to keep the power consumption of any FPGA implementation constant
- An on-chip power monitor based on a network of uniformly distributed ring oscillators
- An on-chip power consumer using a network of long interconnects driven by a switching signal
- A control circuit based on a proportional-integral-derivative controller with auto-tuning capabilities
- An evaluation of our framework on the power regulation of a square-and-multiply exponentiation circuit implemented on a modified Spartan-6 FPGA board

Our results on the exponentiation circuit show that using our framework decreases the information leaked by the power consumption of the system. The peak autocorrelation values of the system's power consumption are reduced by a factor of 2.7 on average while only using 4500 extra lookup tables, that is 26% of the area taken by the exponentiation circuit. For this application, the average power overhead due to our framework is 28%.

The rest of the paper is organised as follows. Section II explains the background relevant to our work. In section III, we present our framework and the realisation of each of its components for FPGAs. Section IV evaluates our framework using an exponentiation circuit as a case study. Finally, section V concludes the paper.

II. Background

This section presents the background relevant to our work. We first discuss simple and differential power analyses and different countermeasures. Then we present the standard model for measuring the power consumption of an FPGA.

A. Simple and differential power analyses

Power analysis is based on the fact that the energy consumed by a hardware module depends on the switching activity of its transistors. Hence, by measuring the power consumed by a chip performing a given cryptographic operation, an attacker can recover information about the data being processed and the secret keys used.

Simple power analysis (SPA) proceeds by direct observation of a power trace. An implementation whose power consumption is different depending on which bit of the secret key is being processed is vulnerable to SPA. This is for instance the case of some implementations of the square-and-multiply modular exponentiation algorithm used in RSA or in the Diffie-Hellman key exchange protocol. The power trace of an unsecured implementation of this algorithm, in which squaring and multiplication operations have significantly different power traces, is presented in [6]. In this example, the private key can be recovered easily with only a single power measurement. Scalar multiplication in elliptic curve cryptography (ECC) is also vulnerable to SPA. As a matter of fact, point doubling or point addition operations are performed, depending on the value of the key.

Differential power analysis (DPA) uses statistical properties of multiple power traces. This method is introduced in [1] where a DPA on a smartcard implementation of the DES algorithm is successfully performed. DPA relies on the correlation between the power consumption of a module and the intermediate data it is computing at a given time. A few bits of the key are determined by considering intermediates that only depend on these bits. For every value of the bits examined, all the possible computational intermediates are enumerated. The sub-key bits are then recovered by examining the correlation between the computational intermediates and the power trace. Then the same method is used with different intermediates to recover the other key bits.

B. Countermeasures

In designing a secured hardware-based cryptosystem one needs to incorporate protections against SPA and DPA. These countermeasures are often application-dependent. For example, power attacks against the modular exponentiation algorithm can be made harder by using the Montgomery powering ladder [7] instead of the square-multiply algorithm. Masking is another application-dependent countermeasure which consists in obscuring intermediate values of the algorithm with random numbers. This technique has been successfully applied to several algorithms [8], [9] but usually leads to area and performance overheads.

Several general countermeasures also exist. When the attacker does not have physical access to the device, filtering the power supply or introducing noise into the measurements are two possible solutions. Another solution is to introduce randomness into the system by using random pre-charges [3]. However, this solution comes at the expense of a reduction in throughput. The mapping, placement and routing algorithms can also be made security-aware. In [10], [11], wave dynamic differential logic (WDDL) and symmetrical routing are used to reduce the power consumption fluctuations. These techniques require specific placement and routing algorithms and can lead to up to 3 times area overhead. Finally, random dynamic voltage and frequency switching have also been proposed as power attack contermeasures [12], [13]. However, these techniques introduce a performance overhead and for current commercial FPGAs, voltage switching would need to be implemented off-chip, compromising the security of the system. No actual hardware implementation of these two last solutions have been presented.

C. FPGA power measurement model

Figure 1 shows a simplified model of the common setting used to measure the power consumption of an FPGA chip as presented in [5] and [14]. A shunt resistor R_{EXT} is placed on the core logic power supply rail V_{CCINT} in series with the FPGA. R_{NET} represents the internal resistance of the power distribution network inside the FPGA. I is the current drain due to circuit switching. The power consumed by the FPGA is given by the following equations:

$$P = V_{INT} I = (V_{CCINT} - V_{TOT}) I \tag{1}$$

$$V_{TOT} = V_{EXT} + V_{NET} \tag{2}$$

$$R_{TOT} = R_{EXT} + R_{NET} \tag{3}$$

Since the voltage drop due to the shunt resistors V_{TOT} is usually small compared with V_{CCINT}, the power consumption can be approximated by the following equations:

$$P \approx V_{CCINT} I \tag{4}$$

$$I = V_{EXT}/R_{EXT} \tag{5}$$

$$\begin{aligned} I &= V_{TOT}/R_{TOT} \\ &= (V_{CCINT} - V_{INT})/R_{TOT} \end{aligned} \tag{6}$$

As shown in the equations, the power consumption of the FPGA is proportional to the voltage drop across the resistors. Using equation 5, an attacker with physical access to the voltage supply pin can obtain a power trace by measuring the voltage drop V_{EXT}. In our constant power framework, we monitor the FPGA's power consumption by measuring the internal voltage V_{INT} as shown in equation 6. The detailed working principles of the power monitor are explained in section III-A.

III. Reconfigurable Computing with Constant Power

Our key idea for constant power reconfigurable computing is summarised in Fig. 2. Our goal is to keep the power constant at a certain value (the setpoint) higher than the maximum power consumed by the user logic. This is illustrated in Fig. 3. Our

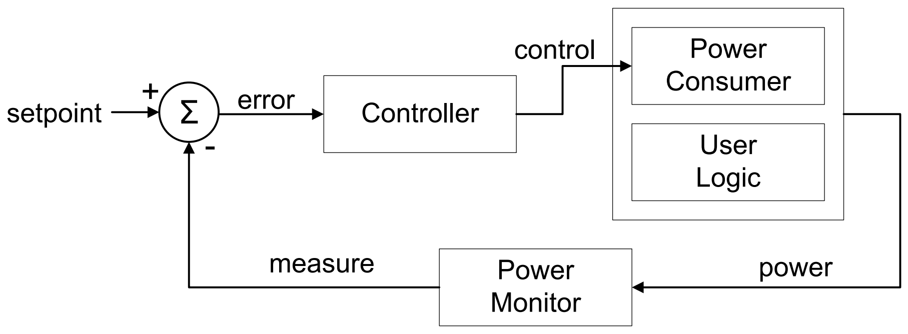


Fig. 2. Constant-power framework

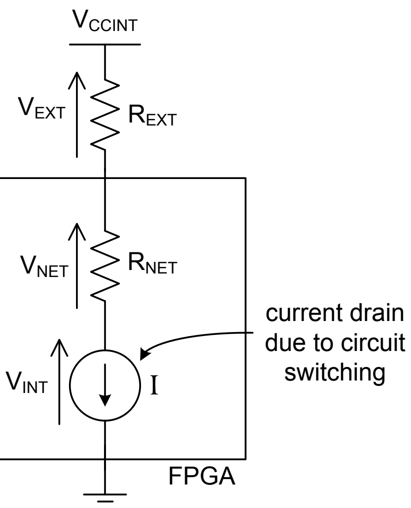


Fig. 1. FPGA power measurement simplified model

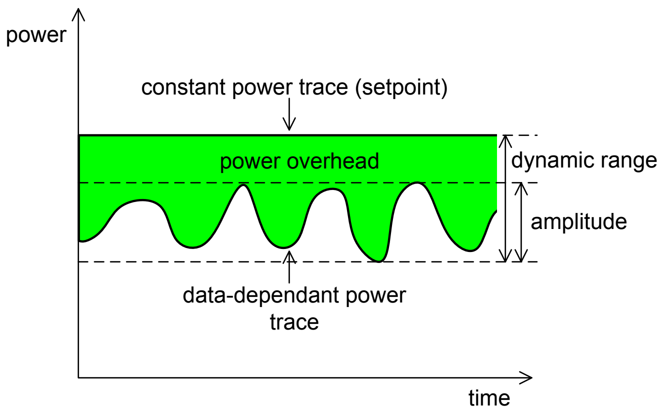


Fig. 3. Power control principle

framework needs to be device-independent and application-independent. It is based on the principles for a closed-loop control system.
The three main components of the control system together with their requirements are presented below:

Power monitor. The power monitor measures the on-chip power of the FPGA. Its input is a value correlated to the on-chip power consumption, such as the average voltage across the power network of the FPGA. Its output is a value proportional to the input that can be easily interpreted by the controller. The power monitor should provide precise and uniform power measurement across the chip. Its resolution should be high enough to detect any small variation of power that can be measured externally.

Power consumer. The power consumer is used to compensate the on-chip power consumption. The power amplitude of the consumer should be higher than the power dynamic range of the user logic (as defined in Fig. 3) so that the power of the system can be kept constant. Its resolution should be high enough to compensate for the smallest variation of power measurable by the power monitor.

Controller. The controller manages the power consumer. Its goal is to make the measurement given by the power monitor match the setpoint. In order to quickly compensate for any power variation of the user logic, the controller needs to be chosen and tuned so that it has good regulation properties. In particular, the controller's response to a sudden power change should be fast enough to hide the power trace of the operations performed by the user logic.

The following sections describe our implementations of the power monitor, the power consumer, and the controller. These three modules are self-contained into the FPGA fabric. This makes our framework resistant to attacks which would consist in removing or replacing some of the on-board power modules in order to bypass the power regulation process.

A. *Power Monitor*

We use ring oscillators (ROs) to monitor the power consumption of the FPGA. Since the circuit switching speed of an FPGA is correlated with its supply voltage V_{INT}, the oscillation frequency of a ring oscillator is affected by the supply voltage [15]. According to equation 6, we can therefore measure the FPGA's power consumption by tracking the oscillation frequencies of ring oscillators implemented inside the FPGA. If the voltage variation in the power supply rail V_{INT} is small, a linear approximation can be used to model the relationship between power and oscillation frequency f_R:

$$f_R \approx k_1 V_{INT} + f_0 \approx -k_2 P + f_0 \tag{7}$$

where k_1, k_2 and f_0 are positive constants, and P is the power consumption of the FPGA. One of the major challenges of using ring oscillator to measure FPGA's power consumption is the trade-off between resolution and response time. In order to obtain a sufficient resolution, we need to accumulate enough oscillations from the ring oscillators. This implies running the ring oscillator for a long period of time, which increases the measurement period. However, increasing the measurement period decreases the number of power measurements that can be taken per second and therefore reduces the response time of the controller. To solve this problem, we evenly distribute a network of ring oscillators among the FPGA. This is shown in Fig. 4. When a new measurement starts all the ring oscillators and the counters are reset. Then the signal from each ring oscillator is accumulated locally by a counter during a fixed amount of time. The outputs of all the counters are summed together and used as the power measurement. The uniformly distributed ring oscillators architecture allows much better resolution with a shorter measurement period at the expense of some area overhead. It also provides a more consistent measurement because the effect of voltage variations within the FPGA is averaged. In our Spartan 6 example, each ring oscillator is implemented in a single configurable logic block (CLB). By using hard macros, we make sure that all the ring oscillators have identical placements and routings within the CLB. This ensures that all the ring oscillators have similar oscillation frequencies and responses to voltage variations.

Using ring oscillators to monitor the power has three main advantages:

- Ring oscillators can be built using primitives that are available to all commercial FPGAs. Hence no FPGA architectural change is required.
- They are relatively small and can thus be easily uniformly distributed among the chip for measuring average power consumption.
- The ring oscillators' frequency scales with the advances of fabrication technology. When the clock frequency of the user logic is increased, a shorter controller's response time is required. Given that the oscillators' frequency is improved proportionally to the clock frequency, the same resolution can be obtained with the same number of ring oscillators.

B. Power Consumer

Figure 5 shows the architecture of our power consumer. It consists of two major components: the power consuming wires and the control circuit. A power consuming wire is a routing interconnect that spans edge to edge vertically or

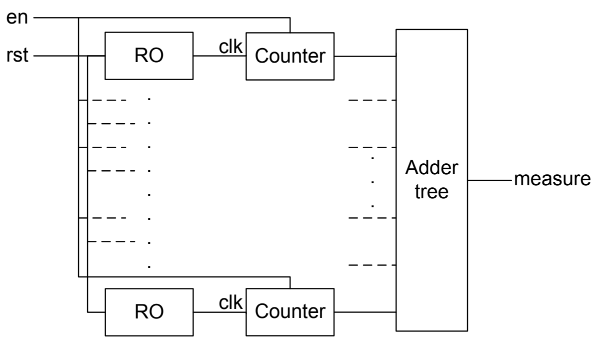


Fig. 4. Power monitor

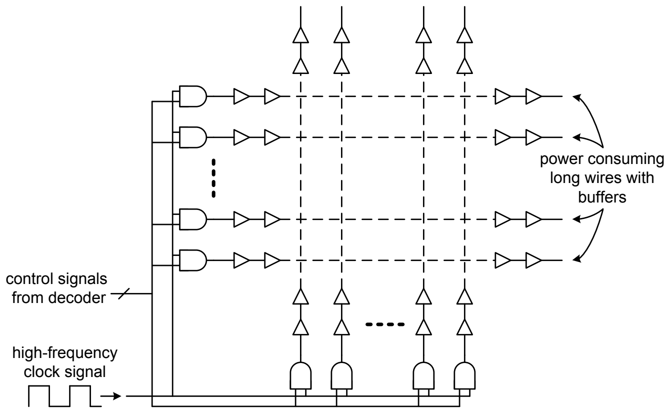

power consuming long wires with buffers
control signals from decoder
high-frequency clock signal

Fig. 5. Power consumer

horizontally across the FPGA. In modern commercial FPGAs all these long routing interconnects are buffered many times to reduce the logic delay. When a periodic switching signal such as a clock signal is fed into one of these long wires, current is drawn at each buffer in order to drive the parasitic capacitance along the wire. Thus significant power is drawn evenly along the wire being activated. We distribute the power consuming wires evenly across the FPGA. We control the number of activated wires using a decoder and an array of AND gates. The hardware descriptions and constraints to guide the uniform placement of the wires are generated automatically by a script. A multiplexer is used at the clock input of the power consumer array to choose between several clock signals with different frequencies. The power consumed by the power consumer can be calculated as follows:

$$P_{consumer} = NCV_{INT}^2 f \tag{8}$$

where N is the number of activated wires, C is the parasitic capacitance of each wire, V_{INT} is the supply voltage of the FPGA's core logic and f is the clock frequency of the power consumer.

C. Controller

We use a proportional-integral-derivative (PID) controller to regulate our system. The PID controller is a commonly used feedback controller and, if well-tuned, has very good regulation and response properties. The controller module has two different modes:

Configuration mode. The following parameters of the system are determined and set up: the power setpoint, the optimal clock frequency of the power consumer, the optimal proportional, integral and derivative constants of the PID.

Regulation mode. The PID controller is regulating the power consumption of the FPGA.

When the FPGA is powered on, the controller begins with the configuration mode. The controller configurations sequence is shown in Fig. 6. First, the user logic is operated during a certain amount of time during which the minimum and maximum power values are obtained. This determines the user logic's power amplitude as shown in Fig. 3. The configuration run is not protected against power attacks. Hence for cryptographic applications, a key different from the secret key should be used. Then the setpoint is set to the maximum power value plus a given margin. This margin takes into account a possible increase in maximum power when using the actual secret key. The user logic's power dynamic range is calculated as the difference between the setpoint and the minimum power value. It is shown in Fig. 3 and corresponds to the maximum power that needs to be generated by the power consumer. Then the power consumer clock frequency is sequentially tuned so that the power consumer's amplitude is greater than but as close as possible to the user logic's power dynamic range. If the power consumer's amplitude is smaller than the user logic's power dynamic range, the control system might not be able compensate the power consumption. However, if the power consumer's amplitude is much greater than the user logic's power dynamic range, the control system is likely to experience quantization effects which would reduce its effectiveness. Finally, the PID controller parameters are determined using the relay feedback auto-tuning method.

The relay feedback auto-tuning method is commonly used to find the optimal parameters of a PID controller. We make the output of the system oscillate by alternating the control command between its maximum value and its minimum value. This corresponds to replacing the PID controller with a relay. The amplitude and the period of the oscillations are determined and the PID coefficients are calculated using the empirical constants given by the Ziegler-Nichols Frequency Domain (ZNFD) method [16], [17].

The principles of the PID auto-tuning method applied to our system are shown in Fig. 7. First the power consumer is set to half its maximum control value $C_{max}/2$ in order to determine the nominal bias value for the power monitor around which the system would oscillate. Then the control command is set to 0. In order to maintain the oscillations, the power consumer control value is set to C_{max} when the power monitor measurement value becomes greater than the nominal bias value. It is set back to 0 when the power monitor measurement value becomes less than the nominal bias value. We wait for the oscillations to stabilise and obtain the amplitude A and the period T of the oscillations. We stop the PID auto-tuning procedure after a few hundred clock cycles.

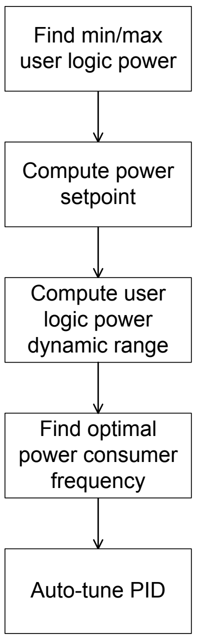

Find min/max user logic power
Compute power setpoint
Compute user logic power dynamic range
Find optimal power consumer frequency
Auto-tune PID

Fig. 6. Controller configuration sequence

The ultimate gain can be computed as follows:

$$K_u = \frac{4h}{\pi A} \tag{9}$$

where h is the control action (amplitude of the command) and A is the amplitude of the response. The ultimate period T_u is equal to the oscillation period T. Using the notations of Fig. 7, we deduce:

$$K_u = \frac{2C_{max}}{\pi A} \tag{10}$$

$$T_u = T \tag{11}$$

The proportional, integral and derivative constants of the PID controller are:

$$K_p = K \qquad K_i = \frac{K}{T_i} \qquad K_d = KT_d \tag{12}$$

where according to the ZNFD method for PID controllers [16]:

$$K = 0.6K_u \qquad T_i = 0.5T_u \qquad T_d = 0.125T_u \tag{13}$$

Finally, the three PID constants used in the discrete control algorithm are defined by the following equations:

$$K_0 = K_p + K_i + K_d \tag{14}$$

$$K_1 = -K_p - 2K_d \tag{15}$$

$$K_2 = K_d \tag{16}$$

These equations are obtained by discretization of the PID controller's equation [17].

After the configuration of the system is finished, the controller switches to regulation mode. At that time the user logic operations can be performed securely. The PID control

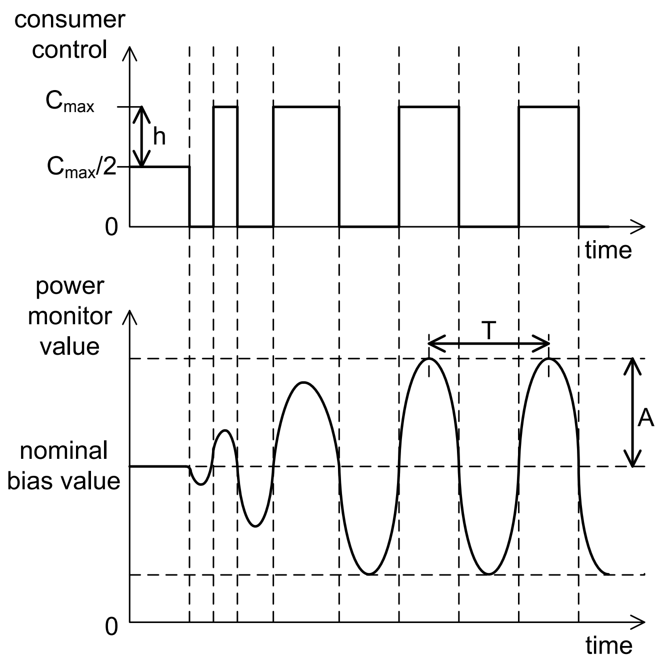


Fig. 7. PID auto-tuning method

algorithm is shown in Alg. 1. Each time a new value from the power monitor is received, the PID control command is computed and the power consumer control value is updated accordingly.

Algorithm 1: PID control algorithm

```
Input: $C_{max}$: maximum power consumer control command
       $setpoint$: setpoint
       $K_0$, $K_1$, $K_2$: PID controller constants
$e = 0$, $e_1 = 0$, $e_2 = 0$, $C = 0$
while $PIDControllerRunning()$ do
    $val = GetPowerMonitorValue()$
    $e_2 = e_1$, $e_1 = e$
    $e = setpoint - val$
    $delta = K_0.e + K_1.e_1 + K_2.e_2$
    $C = C + delta$
    if $C > C_{max}$ then $C = C_{max}$
    if $C < 0$ then $C = 0$
    $SetPowerConsumerControl(C)$
end
```

IV. Results

A. Experimental Setting

Our experimental setting is based on a modified Pico E-101 FPGA board. The Pico E-101 is a small 5x7 cm board embedding a Spartan-6 LX45 FPGA. The modified Pico E-101 board is shown in Fig. 8. It has two main power rails: a 1.2V rail for the FPGA chip and a 3.3V rail for IOs. It is powered in 5V through a USB port. Originally, two switching regulators are used to convert the 5V supply voltage to 1.2V and 3.3V respectively. However, the switching of these regulators creates noise in the power trace.

To address this issue, we remove the 3.3V switching regulator and replace it with a low-noise 3.3V circuit. Our low-noise circuit consists of a low dropout (LDO) regulator together with two resistors used to adjust the output of the regulator. The modification of the 1.2V power supply rail is more complicated as we want to add some power measurement features. The 1.2V switching regulator and the output filtering capacitors are removed and replaced by the same type of low-noise regulation circuit used on the 3.3V rail. For power measurement, a 1-ohm shunt resistor is inserted between the output of the LDO regulator and the 1.2V power rail. The voltage drop in the shunt resistor is measured through an SMA socket connected in parallel with the shunt resistor. The use of such a socket greatly reduces the measurement noise. To ensure that the voltage across the resistor can be measured without differential probes, the FPGA board is powered by a floating 5V power supply. In order to compensate for the average voltage drop in the shunt resistor which depends on the application, the 1.2V regulator output can be adjusted with a variable resistor.

Three main characteristics of our modification ensure low measurement noise by reducing the parasitic capacitance and resistance of the setting: the small size of the board used, the SMA socket used to measure the voltage drop across the shunt resistor, and careful and direct on-board soldering of all the components without using long wires.

The peak-to-peak noise obtained at the shunt resistor of our modified board for a modular exponentiation circuit in idle state is 4.2 mV. This is as low as the measurement noise of the side-channel attack SASEBO-GII board [14].

For each experiment, the design is loaded into the FPGA through JTAG. Then the JTAG cable is disconnected from the board so that the ground is left floating, and the measurements are started. We use a Tektronix MSO 2024 oscilloscope for all our measurements. This oscilloscope has a 200 MHz bandwidth and a 1 GHz sampling rate.

B. Study Case: Modular Exponentiation

As a proof of concept, we consider a hardware implementation of 512-bit modular exponentiation using the square-and-multiply algorithm. The modular exponentiation module is based on the Montgomery multiplier presented in [18]. The square and multiply operations are both performed by the Montgomery multiplier, which makes them hardly distinguishable. To get a better reading of the power consumption, we implement two 512-bit modular exponentiation cores on our Spartan-6 LX45 FPGA and set the clock frequency to 5 MHz. This low frequency also enables a reasonable routing time of our very congested design due to the small size of the FPGA. Both cores are given the same set of inputs in parallel. The three components of our framework are implemented alongside the two exponentiation cores. For this experiment,

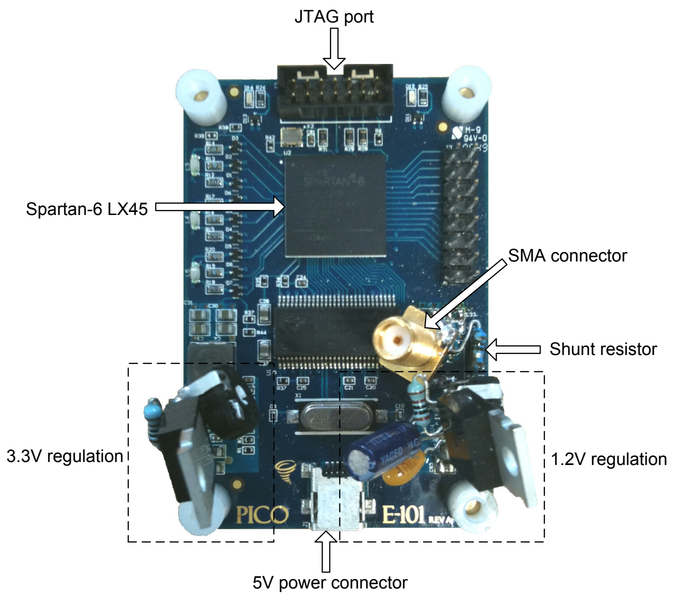


Fig. 8. Modified Pico E-101 FPGA board

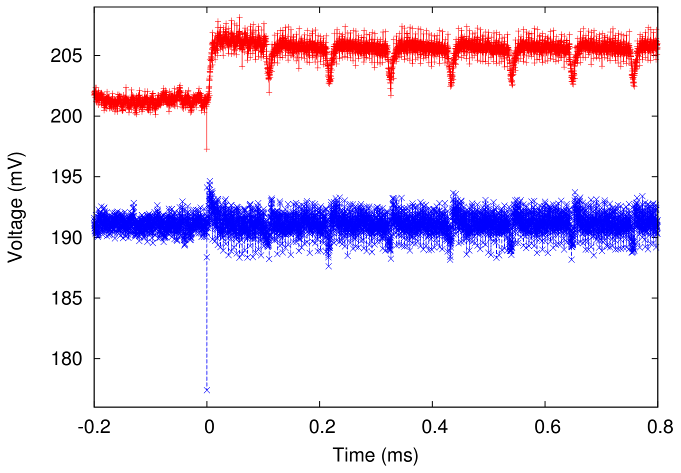


Fig. 9. Modular exponentiation power traces (top trace: without power control, bottom trace: with power control)

the power monitor uses a grid of 144 ring oscillators. The ring oscillators are oscillating at around 350 MHz and the power monitor reading is updated every 2 clock cycles. The power consumer consists of 231 vertical and horizontal interconnects. Our implementation of the framework only takes 4500 extra lookup tables, that is 26% of the area taken by the exponentiation circuit. To reduce the effects of noise, we perform each measurement 10 times and report the average power trace.

Fig. 9 shows the average modular exponentiation power trace with and without power control. The bottom trace is offset by -60 mV. For both traces the exponentiation is triggered at time t=0. Without power control (top trace), as soon as the exponentiation starts the power raises quickly. We can clearly see a repeating pattern every 100 μs. This approximately corresponds to 512 clock cycles, which is the time to perform one Montgomery multiplication. As planned, we cannot easily differentiate a multiplication from a squaring operation. However, an attacker can easily see that the exponentiation has been started and differentiate between each modular multiplication operation. This makes an attack such as a chosen-message power attack [19] possible. With power control (bottom trace), the average power consumption before and after starting the exponentiation are almost the same. The power is higher than the maximum power consumed by the exponentiator without power control. The average power overhead due to power control is 28%. Even if still possible, it is harder to see that an exponentiation has been started and to distinguish between the different multiplications in the power trace. Note that the large power spike seen at t=0 is created by the IO switching of the FPGA pin used to trigger our oscilloscope. This phenomenon would not happen in a real system.

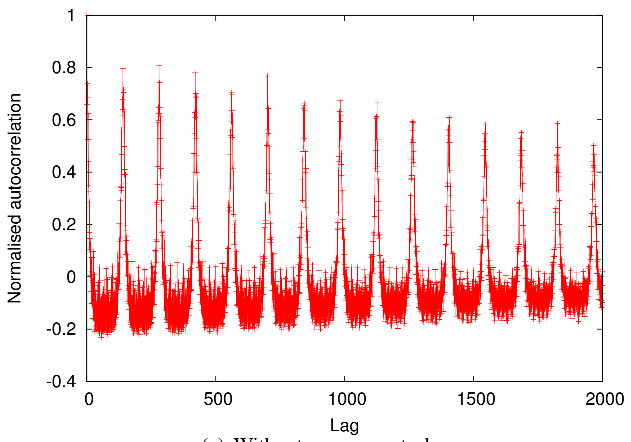
To quantify the security improvement due to the use of our constant-power framework, we compute the autocorrelation of a longer power trace with and without power control. The results are shown in Fig. 10. In both cases, we can distinguish correlation spikes corresponding to the repeating pattern of modular multiplications. However, by switching on the power control the correlation spikes are reduced by a factor of 2.7 on average. Hence less information about the exponentiation operation is leaked, and therefore it would be more difficult to extract the secret key.

The autocorrelation could be reduced further by adding a source of random noise to our system. This could also prevent the power fluctuation information to be identified in shorter time frame. One solution would be to randomly choose the setpoint in a given for each measurement period instead of keeping it constant. The PID controller could also be improved by integrating filtering, setpoint weighting and anti-windup algorithms in its hardware implementation [16].

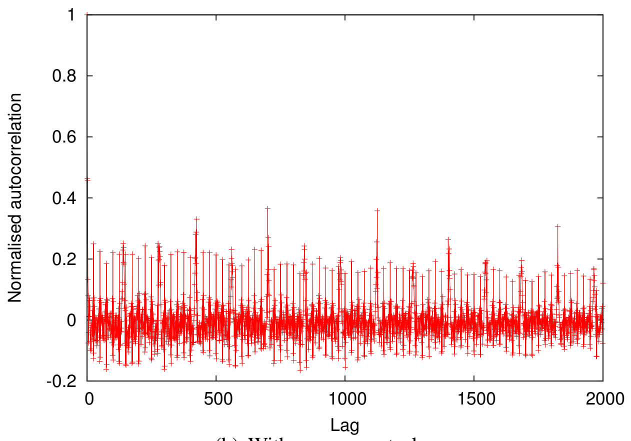
Our framework would still work at a frequency higher than 5 MHz. For the power monitor to keep a high enough resolution, we just need to increase the number of ring oscillators. This would not pose a problem on FPGAs with much more area available than the small Spartan-6 used. In order to run at 50 MHz with the same autocorrelation properties, we estimate that our framework would need 1440 ring oscillators and would take around 35 000 LUTs. The area overhead is less than linear in that case because each counter counts less oscillations and their bit-widths can therefore be reduced. On the Spartan 6 XC6SLX150 FPGA this would represent a 60% area overhead, assuming that the remaining area is taken by exponentiation cores.

V. Conclusion and Future Work

This paper presents Constant Power Reconfigurable Computing, a general and device-independent framework based on a closed-loop control system that can be used to keep the power consumption constant for reconfigurable computing implementations. We demonstrate a realisation of each component of this framework in current commercial FPGA technology. We describe a modification of the Pico E-101



(a) Without power control



(b) With power control

Fig. 10. Autocorrelation of the modular exponentiation power traces

FPGA board that makes it suitable for side-channel attacks with measurement noise as low as what specially designed boards can reach. Our framework is evaluated on an implementation of the square-and-multiply exponentiation algorithm on our board. Our constant power framework decreases the information leaked by the power consumption of the system. The peak autocorrelation values of the system's power consumption are reduced by a factor of 2.7 on average with only 26% area overhead and 28% power overhead.

These results are our first experimental results on Constant Power Reconfigurable Computing. Many aspects of this framework still need to be explored. Current and future work includes adding a random component to the setpoint of the control system to improve the security properties of the framework, integrating filtering, setpoint weighting and anti-windup algorithms to the PID controller, investigating other possible feedback techniques, evaluating our framework (security/area/power/speed trade-offs) based on a wide range of applications and at higher frequencies, assessing more precisely its effectiveness of protecting a system against single and differential power attacks, and evaluating if the proposed countermeasure is tolerant to electromagnetic analysis.

Acknowledgments

The support of BlueRISC, Inc, the FP7 EPiCS (Engineering Proprioception in Computing Systems) Project, the UK EPSRC (Engineering and Physical Sciences Research Council), HiPEAC, and Xilinx is gratefully acknowledged.

References

[1] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology CRYPTO 99*, 1999, pp. 789–789.

[2] F.-X. Standaert, S. B. Ors, J.-J. Quisquater, and B. Preneel, "Power analysis attacks against FPGA implementations of the DES," in *FPL '04*, 2004, pp. 84–94.

[3] F. Standaert, F. Mace, E. Peeters, and J. Quisquater, "Updates on the security of FPGAs against power analysis attacks," in *Reconfigurable Computing: Architectures and Applications*, 2006, pp. 335–346.

[4] T. Messerges, E. Dabbish, and R. Sloan, "Power analysis attacks of modular exponentiation in smartcards," in *CHES 1999*, 1999, pp. 724–724.

[5] S. Ors, E. Oswald, and B. Preneel, "Power-analysis attacks on an FPGA: First experimental results," in *CHES 2003*, 2003, pp. 35–50.

[6] P. Rohatgi, "Protecting FPGAs from power analysis," 2010, http://www.eetimes.com/design/programmable-logic/4199399/Protecting-FPGAs-from-power-analysis.

[7] M. Joye and S.-M. Yen, "The Montgomery powering ladder," in *CHES 2002*, 2003, pp. 1–11.

[8] F. Regazzoni, Y. Wang, and F.-X. Standaert, "FPGA implementations of the AES masked against power analysis attacks," in *Proceedings of COSADE 2011*, 2011, pp. 56–66.

[9] C. Rebeiro and D. Mukhpodhyay, "Power attack resistant efficient FPGA architecture for Karatsuba multiplier," in *Proceedings of the 21st Int. Conf. on VLSI Design*, ser. VLSID '08, 2008, pp. 706–711.

[10] K. Tiri and I. Verbauwhede, "A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation," in *DATE '04*, 2004.

[11] P. Yu and P. Schaumont, "Secure FPGA circuits using controlled placement and routing," in *CODES+ISSS '07*, 2007, pp. 45–50.

[12] S. Yang, W. Wolf, N. Vijaykrishnan, D. Serpanos, and Y. Xie, "Power attack resistant cryptosystem design: a dynamic voltage and frequency switching approach," in *DATE '04*, 2005, pp. 64–69.

[13] K. Baddam and M. Zwolinski, "Evaluation of dynamic voltage and frequency scaling as a differential power analysis countermeasure," in *20th Int. Conf. on VLSI Design*, 2007, pp. 854 –862.

[14] T. Katashita, A. Satoh, T. Sugawara, N. Homma, and T. Aoki, "Development of side-channel attack standard evaluation environment," in *Euro. Conf. on Circuit Theory and Design*, 2009, pp. 403 –408.

[15] J. Franco, E. Boemo, E. Castillo, and L. Parrilla, "Ring oscillators as thermal sensors in FPGAs: Experiments in low voltage," in *Programmable Logic Conference (SPL), VI Southern*, 2010, pp. 133 –137.

[16] K. J. Astrom and R. M. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*, Princeton University Press, pp. 293-313 (2008).

[17] V. Toochinda, "Digital PID controllers," 2009, http://www.dewinz.com/docs/ecs/pid.pdf.

[18] A. Le Masle, W. Luk, J. Eldredge, and K. Carver, "Parametric encryption hardware design," in *6th Inter. Symp. on Reconf. Computing*, 2010, pp. 68–79.

[19] N. Homma, A. Miyamoto, T. Aoki, A. Satoh, and A. Samir, "Comparative power analysis of modular exponentiation algorithms," *IEEE Trans. on Computers*, vol. 59, no. 6, pp. 795–807, june 2010.