Using the Power Side Channel of FPGAs for Communication

Daniel Ziener, Florian Baueregger, and Jürgen Teich
Hardware/Software Co-Design, Department of Computer Science
University of Erlangen-Nuremberg, Germany
email: {*daniel.ziener, teich*}*@cs.fau.de*

***Abstract*—In this paper, we present a novel technique for transmitting data over the power supply pins of an FPGA. Using this power side channel communication, a core inside the FPGA is able to send data to a receiver outside of the FPGA. Possible applications include monitoring, debugging, and watermarking. For the communication, we do not need any further resources, like IO pins or modifications of the board. We characterize the communication channel over the power pins and build a channel model. Furthermore, we present an encoding/decoding method which is independent of the board type and FPGA combination. With this approach, we achieve data rates up to 500 kbit/s. Finally, we provide a case study, which extends existing power watermarking techniques to the new encoding/decoding method and show experimental decoding results.**

I. Introduction

Communication over power supply facilities today features many applications. It is widely used as *power line communication* over conductors that are at the same time used as electric power supply. For example, applications like *home surveillance* (e.g., baby monitor) or *home control* (e.g., remote control of roller blinds, heating, etc.) often use power line communication. Furthermore, internet access over power line, so called *Broadband over Power Lines* (BPL), and small home networks using *PowerLAN* have emerged in the last years. As main advantage of such systems, the available infrastructure, like cables and wires, can be additionally used for communication without the need for new communication media.

However, communication over power lines on *printed circuit boards* (PCBs) or inside integrated circuits is very uncommon. Information on cryptographic operations inside embedded systems can be gathered by *power side-channel attacks*. Usually, the goal is to get the secret key or information about the implementation of the cryptographic algorithm.

Power analysis attacks are based on the observation that different instructions cause variations in the activities on the signal lines, which result in differences in a device's power consumption. With *simple power analysis* (SPA) [1], the measured power consumption is directly mapped to the different operations in a cryptographic algorithm. With this technique, program parts in a microprocessor, for example DES rounds or RSA operations, can be identified. Since the execution of these program parts depends on a key bit, the key bits can be restored. *Differential power analysis* (DPA) [1] is an enhanced method which uses statistical analysis, error correction and correlation techniques to extract exact information about the secret key.

Our goal is to bring these two aspects together and, based on their synergy, show that it is possible to transmit data from a core which is implemented on an FPGA over the power pins to the outside where the data can be decoded. No additional ports or pins are required. This idea to use the power pins for off-chip communication might be beneficial for a multitude of scenarios, for example if a communication possibility must be added late in the development phase and no further dedicated pins of the FPGA are available. Furthermore, the data can be sent from cores which are located deep inside the design hierarchy. Using the standard approach, the communication signals must be connected on each hierarchy level up to the top-module and then to the pins. This can affect many components which must be adapted. Using power communication, this effort is not necessary.

Further applications for using this *unidirectional communication channel* are a) *monitoring*, b) *debugging facilities*, and c) *power watermarking* [2], [3]. For monitoring purposes, the core can send measurements or status information over the power channel. For debugging, a dedicated debug circuit inside the core can record signal values after a special trigger and is able to send these values to an external device that measures the power or voltage of the FPGA, decodes, and displays these values. Watermarking means that a unique signature which clearly identifies the author is embedded into an IP core. A product developer might obtain an unlicensed version of the IP core and integrate this core into his product. Proof of the authorship can be established by extracting and comparing the signature from the product. The basic idea behind the power watermarking approach is to extract the core signature from the FPGAs power consumption pattern.

The remaining work is organized as follows: Section II describes the basic concepts for analysis of the core voltage. Section III discusses the power communication channel of FPGAs and Section IV describes the encoding/decoding of the transmitted data. Section V shows the experimental results for the communication channel measurements, and in Section VI, a case study using a power watermarking application is given. Finally, Section VII concludes the paper.

II. Concept

There is no way to measure the relative power consumption of an FPGA directly, only through measuring the relative supply voltage or current, or indirectly by measuring the temperature. We have decided to measure the core voltage as close as possible to the voltage supply pins, such that

the smoothing from the plane and block capacitances are minimal and no shunt is required. Most FPGAs have *ball grid array* (BGA) packages and the majority of them have vias to the back of the PCB for the supply voltage pins. So, the voltage can be measured on the rear side of the PCB with an oscilloscope. The voltage can be sampled using a standard oscilloscope, and analyzed and decoded using a program developed to run on a PC (see Figure 1).

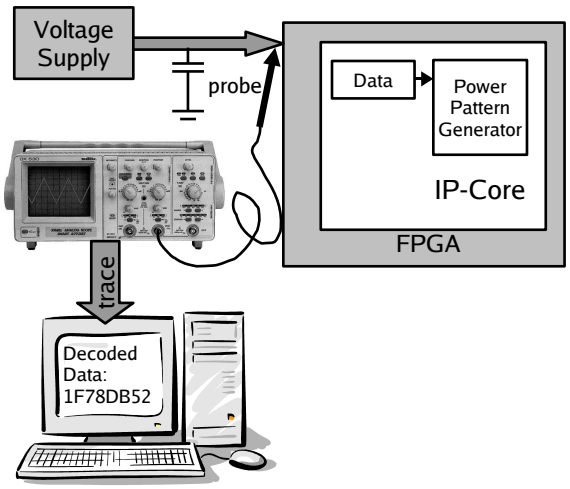

Fig. 1. Communication over the power channel: From a data source inside the core, a power pattern is generated that can be scanned at the voltage supply pins of the FPGA. An algorithm decodes the transmitted data from the trace.

The consumed power of an FPGA can be divided into two parts, namely the static and the dynamic power. The static power consumption is caused by the leakage current from CMOS transistors and does not change over time if the temperature stays constant. The dynamic power consists of the power related to short circuit currents and the power required of reloading the capacitances of transistors and wires. The short circuit current occurs when the *PMOS* and the *NMOS* transistors are both in conducting state for a short time during the switching activity. As shown in [4], the main part of an FPGA's dynamic power results from capacitance reloading. Both parts of the dynamic power consumption depend on the switching frequency [5].

What happens to the core voltage, if many switching activities occur at the same time, at the rising edge of a clock signal? The core supply voltage drops and rises (see Figure 2). In the frequency domain, the clock frequency with harmonics and even integer divisions are present. The real behavior of the core voltage depends on the individual FPGA, the individual printed circuit board and the individual voltage supply circuits.

In our approach, we alter the amplitude of the interferences in the core voltage. The basic idea is to add a power pattern generator (e.g., a set of shift registers), and clock it either with the operational clock or an integer division of thereof. Further, we control these power pattern generators according to the characteristics of the data sequence which should be sent. A logical '1' lets the power consumer operate one cycle (e.g., perform a shift), a '0' causes no operation. We detect higher amplitudes in the voltage profile over time corresponding

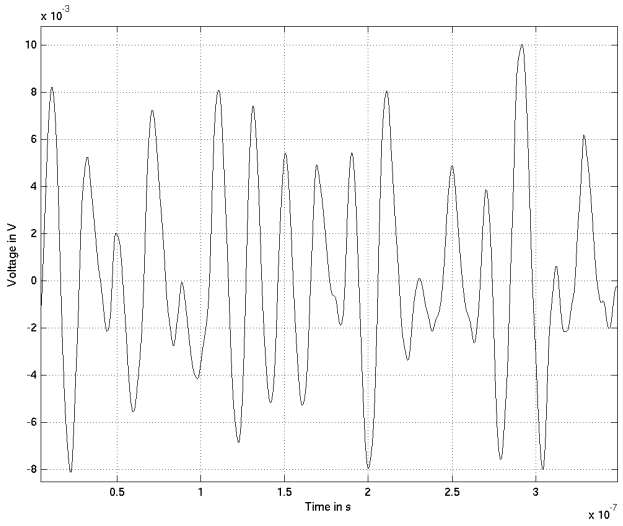

Fig. 2. A measured voltage signal from the voltage supply pin of an FPGA. The core supply voltage drops and rises. Note that the DC component is filtered out.

to the ones and smaller amplitudes according to the zeros. Note that the amplitude for the no-operation state is not zero, because the operational logic and the clock tree are still active.

III. Communication Channel

The transmission of a signal, generated from a source of data, over the supply voltage to a testing point outside of the FPGA, which can be accessed by an oscilloscope, presents an *unidirectional communication system*. The source is the power pattern generator inside the core, and the sink is a device that decodes the signal after it has been measured, digitized and recorded with an oscilloscope. This communication channel transforms the signal, and adds noise. If we know how to characterize the channel, we are able to build better encoding/decoding systems. Therefore, the behavior of the communication channel must be approximated in a channel model.

In current digital communication systems as well as in our technique, the source encoding is usually a binary encoding of the data to be transmitted. From the source encoded data, the transmission sequence is generated by channel encoding. The digital modulation then translates the encoded data sequence into single, successively transmitted *symbols*.

A symbol σ is the smallest unit which carries information. The sequence of different symbols, generated by modulation, is called *signal*. The signal is transmitted over and altered by the communication channel. After reception, the signal is demodulated and decoded to restore the source encoded data.

On a real communication channel, the signal is disturbed by interferences, noises, line losses, delays, etc., which complicates the decoding of the signal. With a certain probability $p_E > 0$, the symbol cannot be correctly decoded. For example, when using a binary modulation where the bit '1' corresponds to the symbol σ_1 and a '0' corresponds to the symbol σ_0, the decoder must first calculate the probability that the currently received symbol was sent as σ_0 or σ_1. After that, the decoder can decide in favour of one symbol. To lower p_E, the channel

code can add redundancy which increases the probability of correct decoding. The decision which channel code and modulation can be used depends on the communication channel as well as the sending and receiving characteristics. Thus, it is necessary to develop a channel model which adapts the actual communication and disturbance as close as possible.

For using the power channel for communication, the mapping of a bit sequence $s = \{0,1\}^n$ into a sequence of symbols $\{\sigma_0, \sigma_1\}^n$ is called encoding: $\{0,1\}^n \rightarrow \mathcal{Z}^n, n \geq 0$ with the alphabet $\mathcal{Z} = \{\sigma_0, \sigma_1\}$. Here each bit $\{0,1\}$ is assigned to a symbol. Each symbol σ_i is a triple $(e_i, \delta_i, \omega_i)$, with the *event* $e_i \in \{\gamma, \bar{\gamma}\}$, the *period length* $\delta_i > 0$, and the *number of repetitions* $\omega_i > 0$. The event γ is *power consumption through a shift operation* and the inverse event $\bar{\gamma}$ is *no power consumption*. The period length is given in terms of a number of clock cycles. For example, the encoding through 32 shifts with the period length 1 (one shift operation per cycle) if the data bit '1' should be sent, and 32 cycles without a shift operation for the data bit '0' is defined by an alphabet $\mathcal{Z} = \{(\gamma, 1, 32), (\bar{\gamma}, 1, 32)\}$

A. Impulse Response

Every linear, time invariant system or channel can be characterized by its *impulse response*. The impulse response $h(t)$ is the reaction of the system to an infinitely brief signal with an infinitely high amplitude – an impulse. Let $x(t)$ the *input signal* and $y(t)$ the *output signal* of the system, then the output signal can be calculated by the *convolution* of $x(t)$ with the impulse response $h(t)$:

$$y(t) = \int_{-\infty}^{\infty} h(\tau) \cdot x(t-\tau)\, d\tau. \tag{1}$$

With known impulse response, the original signal $x(t)$ can be reconstructed from the received signal, if the signal/noise ratio is not too low. To determine $h(t)$, an impulse must be generated on the input and the output must be measured. A *Dirac-pulse* is defined as an infinitely short signal with an infinitely high amplitude [6]. To measure the impulse response in a real system, an approximation of the Dirac-pulse must be used. A single short and high *rectangular signal* is suitable in most cases.

For measuring the power communication channel, such a single short pulse is used. We use several power consumers which are all active only for a short time. A set of *shift registers* with common clock input is suitable for the generation of the impulse. To obtain a high amplitude, the toggle rate in each shift register must be maximized. This can be done by initializing the shift registers with the sequence "010101 . . . " and feeding back the output to the input to perform a cyclic shift. If all shift registers are enabled for exactly one clock cycle, this causes maximum power consumption in minimal time.

The resulting signal is recorded and digitized outside the FPGA with a *digital oscilloscope*. To reduce the noise and disturbances in the result, several responses of different impulses are recorded and then averaged. The resulting signal is the impulse response for this board and FPGA. Note, that the impulse response is different for each combination of FPGA and board, because the power switching characteristic depends mainly on different capacitances inside the FPGA as well as on the board. Furthermore, the power supply circuit has also a high influence on the impulse response. Figure 3 shows an example impulse response for the Digilent Spartan-3 starter board [7].

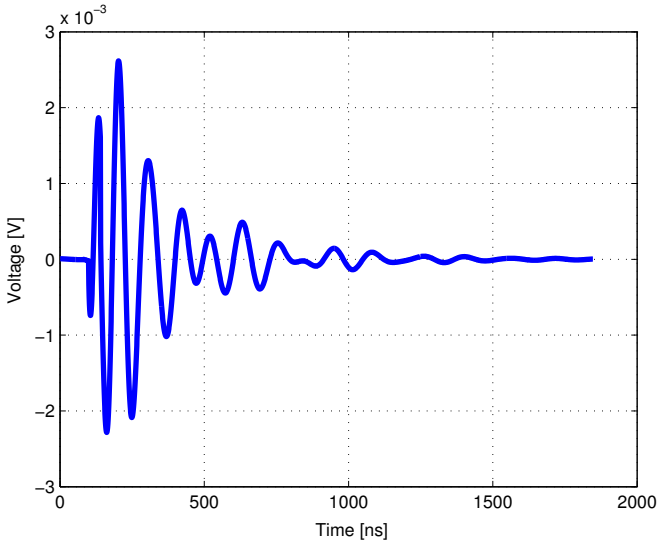

Fig. 3. The impulse response obtained by a shift of a huge shift register, implemented with 128 SRL16 primitive cells in the Spartan-3 FPGA on the Digilent Spartan-3 starter board [7] with the experimental setup from Section V.

Experimental results (see Section V) show that the impulse responses of different boards look similar, but are not completely the same. Using *mathematical approximation*, the impulse response can be generalized to find a function which can be parameterized over as few parameters as possible and which hopefully covers all possible FPGA and board combinations. After transformation of the measured impulse responses into the frequency domain, only few independent frequency components can be identified in the spectrum. Each of those components is approximated through a *basic frequency component* and an *envelope function*. A good starting point for approximating the envelope is the *probability density* function of the χ^2_n*-distribution* The χ^2_n-distribution is a common *continuous probability distribution* of a sum of squares of n independent random variables. The density function of the χ^2_n-distribution $P(n,t)$ is especially suitable for approximation, because it has only one extra parameter n besides the time t:

$$P(n,t) = \frac{t^{\frac{n}{2}-1} \cdot e^{\frac{-t}{n}}}{2^{\frac{n}{2}} \cdot \Gamma(\frac{n}{2})} \quad \forall n, t > 0, \tag{2}$$

where the gamma function $\Gamma(x)$ is defined as:

$$\Gamma(x) = \int_0^{\infty} t^{x-1} e^{-t}\, dt. \tag{3}$$

A frequency component $h_i(t)$ can therefore be approximated by:

$$h_i(t) = \alpha_i \cdot P(n_i, t) \cdot sin(2\pi f_i t + \phi_i), \tag{4}$$

with four parameters that need to be adapted: n_i, α_i, f_i, and ϕ_i. The approximated impulse response consists of the

combination of l different frequency components:

$$h(t) = \sum_{i=1}^{l} h_i(t) \tag{5}$$

As our experimental results in Section V show, only two or three different frequency components are necessary to get a good approximation. The different parameters can be adapted using a *genetic algorithm* [8] on the measured impulse response from a given FPGA and board combination. We used the minimization of the average square error as optimization goal. Note that the number of used shift registers influences the impulse response as well. However, we show later that the usage of more or less shift registers only influences the amplitude α of the signal which can be adapted by a constant factor.

B. Synchronization

For measuring the impulse response as well as for the decoding the transmitted data later, several repeated equal data sequences can be accumulated to reduce the noise and disturbances which are not related to the sequence. For this approach, it is important that the individual sequences to be accumulated are exactly aligned over the time axis. This can be done by synchronizing the *oscilloscope sample clock* with the *shift clock* inside the FPGA.

Another way to get exact, noise-reduced data is by multiple oversampling inside the oscilloscope for recording the measurements. Here, a subsequent resynchronization on the measured data might be necessary.

To check this, the different sequences are plotted on top of each other. If the sequences match exactly, no resynchronization is necessary. If there is a shift over time, a resynchronization must be done. It is only necessary to look for the positions of the maxima if a sequence consists of only one event. A resynchronization is necessary, if the positions of the maxima drift away over time and can be done by inserting or removing samples for compensating the deviation.

C. Channel Approximation

The measured signal $y(t)$ can be approximated with the help of the impulse response:

$$y(t) = c \cdot \left(\int_{-\infty}^{\infty} h(\tau) \cdot x(t-\tau)\, d\tau \right) + n(t). \tag{6}$$

$x(t)$ is the input signal, consisting of Dirac-pulses which encode the signature. For each dirac-pulse, a scaled impulse response is inserted. The *scaling factor* c depends on the amplitude of the pulse which therefore corresponds to the number of currently active shift registers. The assumption that the amplitude of the pulses depends linearly on the number of shift registers has been confirmed by experiment. Figure 4 shows the measured amplitude of the impulse response caused by different numbers of active shift registers.

Furthermore, the noise component $n(t)$ has influence on the decoding quality. The quality of the signal $y(t)$ can be measured by the SNR. If the SNR is low, it is difficult or impossible to correctly decode the corresponding symbol. The SNR is therefore an indicator of how reliable the decoding of a given signal is.

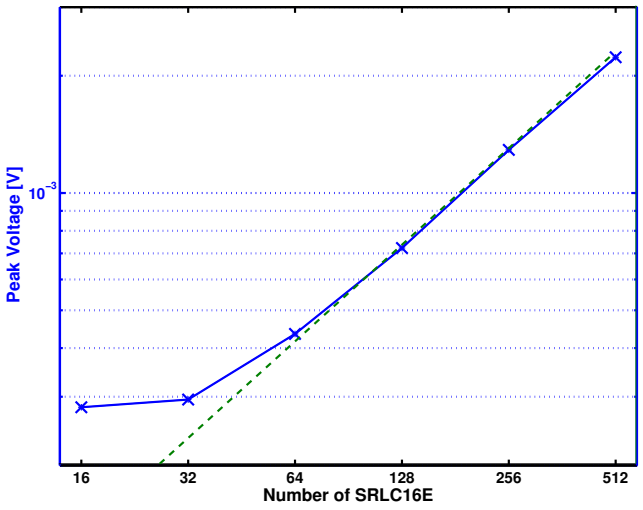

Fig. 4. The amplitudes of the impulse responses caused by different numbers of active shift registers in the Spartan-3 FPGA on the Digilent Spartan-3 starter board. With more than 32 SRLC16E in use, the peak voltage is directly proportional to the number of shift registers. If less than 32 SRLC16E are used, the peak amplitude is in the same range as the noise amplitude, which falsified the measurement.

D. Intersymbol Interference

If the symbols superpose, for example when choosing a too small symbol period length, then *intersymbol interference* (ISI) occurs. This has the same effect as noise and decreases the quality of the decoding and results in a higher bit error rate. If a certain symbol σ_0 has the length τ_0 and the time slots for transmitting the symbols τ' are shorter than the symbol length ($\tau' < \tau_0$), then the symbol σ_0 interferes with $\left\lfloor \frac{\tau_0}{\tau'} \right\rfloor$ subsequent symbols.

IV. Encoding and Decoding

In this paper, we present a method for decoding data through correlation. To achieve good correlation results, the encoding of the signature should be interference-free. To avoid ISI, the time slots for sending the symbols τ' must be larger then the symbol length τ_σ. So that the impulse response $h(t)$ can be used directly for correlation, all symbols should consist only of one pulse. Therefore, we use the encoding: $\mathcal{Z} = \{(\gamma, \delta, 1), (\bar{\gamma}, \delta, 1)\}$. The symbol length is equal to the length of $h(t)$, and a time slot is $\tau' = \delta \cdot \frac{1}{f_{clk}}$. A necessary condition for an interference free code is: $\delta \geq \tau_\sigma \cdot f_{clk}$.

The question is how to estimate the symbol length τ_σ which is also the length of the impulse response. The length of $h(t)$ depends on the combination of FPGA and board which can be measured and approximated (see Section III). However, the FPGA and board combination is sometimes not known at design time. This is possible if the sending core is published as netlist core, e.g., as a power watermarked core. In this case, it is up to the core customer to choose the FPGA and the board. Due to this reason, a safety margin should be considered. Experimental results in Section V show that between 80 and 100 ns after the start of the symbol, 90% of the energy is emitted and after 125 ns, over 95%. If we assume clock frequencies $f_{clk} < 200\ MHz$, then the number of clock cycles $\delta = 25$ used for sending one symbol should be sufficient.

The *cross-correlation* $Z_{x,y}(t)$ of two functions $x(t), y(t)$ is defined as:

$$Z_{x,y}(t) = \int_{-\infty}^{\infty} x^*(t) \cdot y(t+\tau)\, d\tau. \tag{7}$$

In equation 7, $x^*(t)$ denotes to the complex conjugate of $x(t)$. The cross-correlation is a measure of the similarity of functions $x(t)$ and $y(t)$, if $y(t)$ is shifted over the time axis t. The maximum of $Z_{x,y}(t)$ denotes the time with the highest similarity.

To decode the data, the signal is correlated with the approximated impulse response $h(t)$. If the signature bit '1' was sent, then the correlation result has a peak at this position, otherwise when a '0', was sent, no peak occurs. For better decoding, the signal and the impulse response are mixed down into the base band of one frequency component of $h(t)$, e.g., $h_1(t)$. Then, the mixed down signal and impulse response are correlated. Figure 5 shows such a correlation result. A possible decoding algorithm can look at the positions of the symbols in order to find peaks. If the first symbol position is known, the other positions can be calculated based on the encoding scheme and the clock frequency. The symbols on these positions can be decoded by making a threshold decision. If the signal value is higher than a certain threshold, the decoder decides on a '1', otherwise on a '0'. However, after transmission of several bits with the value '1', the absolute peak values are increasing slowly. Therefore, the threshold value must be adapted dynamically based on the precedented values.

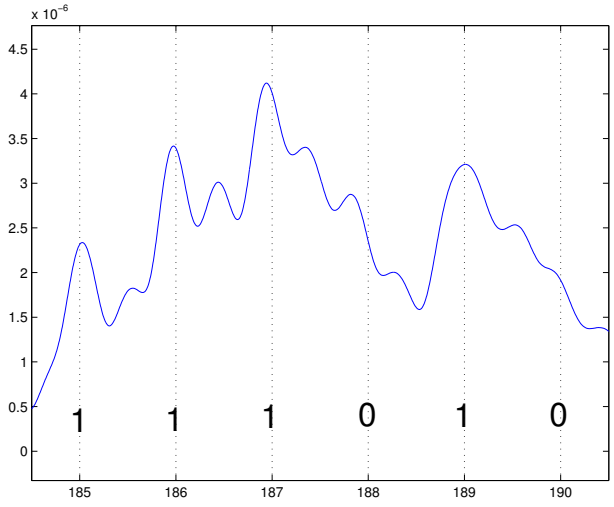

Fig. 5. Detection of the data bits through down mixing into the base frequency of the impulse response component with the most energy content (here $f_i = 43\ MHz$) and correlation with the down mixed $h_i(t)$. Detection is done by searching the maxima and applying a dynamic threshold decision. The sample times for the decisions are depicted with dotted lines.

Obviously, better results could be achieved if more frequency components are integrated into the decoding process. If this is necessary, the decoding is done for each component alone and the decoded data is compared. If bits differ, then these bits are less reliable.

In our methodology, the data is sent in multiple data packets. The first symbol position in a data packet can be detected (e.g., by *synchronization methods*). In order to increase detection ratio for the first symbol, we use a *preamble* to determine the correct start position of the payload. The preamble is a known bit sequence that has the same encoding as the user data and is transmitted directly before the payload data.

In [3] and [2], we have proposed power watermarking methods which have different encoding/decoding schemes and which can also be used for communication over the power channel. The *basic method* uses an encoding scheme of $\mathcal{Z} = \{(\gamma, 1, 1), (\bar{\gamma}, 1, 1)\}$. The experimental results in [2] show that the decoding is often not possible due to heavy intersymbol interference. The encoding scheme of the *enhanced robustness encoding method* is $\mathcal{Z} = \{(\gamma, 1, \omega), (\bar{\gamma}, 1, \omega)\}$. Encoding with $\omega = 32$ improves the decoding rate by far. However, this encoding scheme does not strictly avoid ISI. The impact of ISI is less because the symbol length is extended. For certain FPGA and board combinations, the different superimposed impulse responses might cancel out each other. This could happen if the phases of two subsequently following impulse responses are complementary. Therefore, the interference-free encoding of the correlation method presented in this paper is preferred even if the experimental results of our setup shows a better decoding with the non-interference-free enhanced robustness encoding method.

The *BPSK method*, introduced in [3], uses a two step encoding. First, the data is encoded using *binary phase shift keying* (BPSK) and then the result is encoded with an *on-off keying* (OOK) with $\mathcal{Z} = \{(\gamma, 1, \omega), (\bar{\gamma}, 1, \omega)\}$. Experimental results using $\omega = 10$ have shown that correct decoding is possible even if many interferences from other working cores are present. The disadvantage of this method is the decreased data transfer rate. However, on designs with heavy interferences from other cores, the BPSK method can also be used for transmitting data over the power channel. Furthermore, the encoding of the OOK modulation can be adapted to the ISI-free encoding scheme of the correlation method.

The decoding results can be further improved by sending each data packet repeatedly and averaging several signals before decoding to reduce the noise level and filter out interferences which are not correlated with the encoded data. This can in general be done for all different encoding schemes, but again with the disadvantage of a decreased transfer rate.

V. Experimental Results

In the following experiments, we used the same two FPGA-boards as in [3], the Digilent Spartan-3 Starter Board [7], and a board equipped with a Xilinx Virtex-II XC2V250 FPGA. On the second board, many other components such as an ARM micro-controller and interface chips are integrated to demonstrate that the algorithm is also working on multi-chip boards. The Spartan-3 board operates at a clock frequency of $50\ MHz$, the Virtex-II board at $74.25\ MHz$.

On both boards, the voltage is measured on the back of the printed circuit board directly on the via which connects the FPGA with the power plane of the printed circuit board. We used a $50\ \Omega$ wire with a $50\ \Omega$ terminating resistor soldered directly on the vias. We have used a *DC block* element and a $25\ MHz$ high pass filter to filter out the DC component and

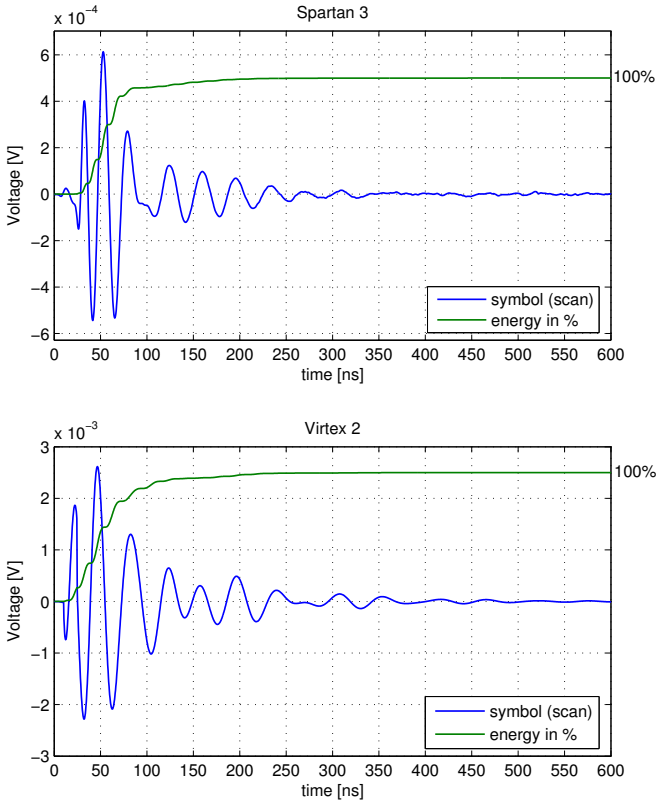

Fig. 6. The measured impulse responses of the Spartan-3 Board (above) and the Virtex-II Board (below). The impulse responses were averaged over many single impulse responses on each board. Furthermore, for each impulse response the percentage of the energy content is shown over the time.

the interferences of the switching voltage controller. We used a *LeCroy Wavepro 7300 oscilloscope* with 20 Giga Samples per second to measure the voltage. The voltage amplitude of the measured switch peak is very small, so we used a *digital enhanced resolution filter* to improve the dynamics, at the cost of a decreased bandwidth. The signal of the length of $200\ \mu s$ is recorded on the internal hard disc of the oscilloscope. This trace file is then transferred to a personal computer and analyzed there.

First, we measured the impulse responses of the two boards of the experimental setup. The impulse is generated using a shift of 64 *16 bit* shift registers (SRL16) inside the FPGA which are initialized with a "$010101\ldots$" pattern to achieve the highest possible toggle rate. Figure 6 shows the measured impulse responses of the two boards.

TABLE I
APPROXIMATION VALUES AND ERRORS FOR $h(t)$ FOR THE SPARTAN-3 AND VIRTEX-II BOARD.

	α	f	ϕ	n
Spartan-3 Board Approx. Error: 1.5%				
$h_3(t)$	$0.56\ mV$	$43.0\ MHz$	1.7π	11.3
$h_2(t)$	$0.17\ mV$	$26.9\ MHz$	1.6π	4.49
$h_1(t)$	$0.16\ mV$	$75.7\ MHz$	1.5π	19.2
Virtex-II Board Approx. Error: 4.7%				
$h_3(t)$	$1.96\ mV$	$36.5\ MHz$	0.06π	12.6
$h_2(t)$	$1.49\ mV$	$26.6\ MHz$	1.1π	12.3
$h_1(t)$	$1.02\ mV$	$66.5\ MHz$	1.7π	23.8

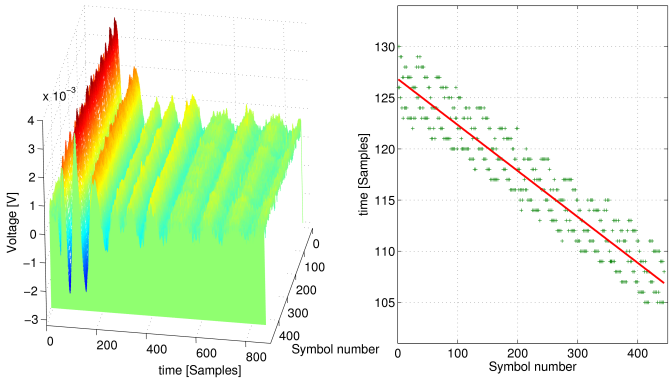

Fig. 7. Many subsequently measured impulse responses (symbols) are depicted on the left side. The measurement is done for the Virtex-II Board, where the measured drift is too high. By plotting the maximum of each symbol the drift can been seen (shown on the right side). Each cross corresponds to the maximum of a symbol and the straight line depicts the average drift.

The approximation of the impulse responses of the two boards was done using a genetic algorithm and for three frequency components ($l = 3$) each. The resulting parameters α, f, ϕ, and n are shown in Table I.

The measurements for the synchronization between the clock frequency of the board and the sample frequency of the oscilloscope depicted that the sample drift for the Spartan-3 board is good enough, whereas for the Virtex-II board, we need a subsequent resynchronization (see Figure 7). This was done by removing sample values for each symbol to lower the drift into a specified range.

The achieved data transfer rates depend on the encoding scheme as well as on the repetition rate and the size of the data packets. Using no data repetition, the encoding scheme $\mathcal{Z} = \{(\gamma, 25, 1), (\bar{\gamma}, 25, 1)\}$, and $f_{clk} = 200\ MHz$, we are able to achieve gross data rates of up to $8\ Mbits/s$. On a more realistic scenario with a packet repetition rate of ten and a 12 bit preamble and 32 bit data, the data achievable transfer rate is $500\ kbit/s$. The data rate for the enhanced robustness encoding method is similar to the correlation method, whereas the robust BPSK method achieves only data rates of approximately $450\ kbit/s$ without data packet repetition.

VI. CASE STUDY: POWER WATERMARKING

In this section, we present as case study a power watermarking technique, introduced in [2] and [3]. First, a short introduction and related work are shown. The technique will be extended by the correlation method introduced in Section IV and experimental decoding results will be presented.

A. Introduction and Related Work

IP cores are licensed and distributed like software. One problem of the distribution of IP cores, however, is the lack of protection against unlicensed usage. The cores can be easily copied. Some core suppliers encrypt their cores and deliver special development tools which can handle encrypted cores. The disadvantage is that common tools cannot handle encrypted cores and that the shipped tools can be cracked so that unlicensed cores can also be processed.

Another approach is to hide a signature in the core, a so called watermark, which can be used as a proof of the original

authorship. There exist many concepts and approaches on the issue of implementing a watermark into a core. Watermarking procedures can be categorized into two groups of methods: *additive methods* and *constraint-based methods*.

In additive methods, the signature is added to the functional core, for example, by using unused lookup-tables in an FPGA [9]. The constraint-based methods were originally introduced in [10] and restrict the solution space of an optimization algorithm by setting additional constraints which are used to encode the signature. Some methods for constraint-based watermarking in FPGAs exploit the scan-chain [11], preserve nets during logic synthesis [12], place constraints for CLBs in odd/even rows [13], or route constraints with unusual routing resources [13].

The major drawback of these approaches are the limitations of the verification possibilities of the watermarked core. With a good watermarking strategy, the verification can be done only with the given product and without additional information from the producer. The bitfile of an FPGA can be extracted by wire tapping the communication between the PROM and the FPGA. Only the approaches presented in [9] and [14] offer the possibility to detect the watermark in these bitfiles.

The approach in [14] is using the content of the lookup tables in an FPGA for the identification of a core. From the FPGA bitfile, all lookup table contents and positions are extracted and compared with the lookup table contents of the netlist IP core. Then, a covering is calculated to show if the core is included in the FPGA design or not. However, some FPGA suppliers provide an option to encrypt the bitstream. The bitfile is stored in the PROM in encrypted form and will be decrypted inside the FPGA. Monitoring the communication between PROM and FPGA in this case is useless, because only the encrypted file will be transmitted. Other possibilities are to verify the watermark by monitoring the temperature [15] or, in our case, the power. The temperature approach for bitfile cores is the only watermarking technique which is commercial available. An overview and evaluation of existing watermark techniques is given in [16].

B. Basic Concept

The power watermarking methods described in [2] and [3] use two shift registers, a large one for causing a recognizable signature-dependent power consumption pattern, and a shift register storing the signature itself (see Figure 1 in Section II). The signature shift register is clocked by the operational clock and controls the power pattern generator corresponding to the encoding scheme. To avoid interference from the operational logic in the measured voltage, the signature is only generated during the reset phase of the core.

In some FPGA architectures (e.g., Xilinx Virtex), the lookup tables (LUTs) can also be used as a shift register [17]. A 4 Bit lookup table can also be used as a 16 Bit shift register. And, also for the Xilinx Virtex architecture, the content of such a shift register can be addressed by the LUT input ports. This allows us to use functional logic for implementing the power pattern generator. The core operates in two modes, the *functional mode* and the *reset mode*. In the functional mode, the shift is disabled and the shift register operates as a normal lookup table. In the reset mode, the content is shifted according to the signature bits and consumes power which can be measured outside of the FPGA. To prevent the loss of the content of the lookup table, the output of the shift register is fed back to the input, such that the content is shifted circularly. When the core changes to the functional mode, the content must be shifted to the proper position to have a functional lookup table for the core.

The advantages of using the functional logic of the core as a shift register are the reduced resource overhead for watermarking and the robustness of this method, because these shift registers are embedded in the functional design. It is hard if not impossible to remove shift registers without destroying the functional core.

The watermarking embedding procedure is easy to use and consists only of two steps. First, the core must be embedded in a wrapper, which contains the control logic for emitting the signature. This step is done at the HDL-level and before synthesis. The second step is at the netlist level after synthesis. A program converts suitable four input lookup tables (LUT4) into shift registers (SRL16) for the generation of the power pattern generator and attaches the corresponding control signal from the control logic in the wrapper. The initial content of the lookup table is left unchanged.

The wrapper contains the control logic for emitting the watermark and the shift register, holding the signature. The ports of the wrapper are the same for the core, so we can easily integrate this wrapper into the hierarchy. The control logic enables the signature shift register, while the core is in reset state. Also, the power pattern shift registers are shifted in correspondence to the output of the signature shift register. If the reset input of the wrapper gets inactive, the function of the core cannot start at the same cycle, because the content in the shift registers is probably not in the correct position. After that the control logic deactivates the internal reset signal to start the normal function mode.

C. Correlation-Based Detection Method

The signature is encoded and decoded for the correlation detection method as described in Section IV. The measurements are done with the experimental setup, described in Section V. The functionality of the correlation detection method is evaluated for a *Des56* core from *opencores.org* [18]. After the synthesis step, only 40 out of 715 lookup tables from the *Des56* core have been transformed into SRL16 and a 32 Bit signature has been added. The decoded sequence was compared with the encoded signature from the core to evaluate the bit error rate.

We have evaluated three cases, one where only the watermarked core is implemented (case A), and one where the watermarked core and the original core are implemented to check the functionality of the watermarked core (case B). This is done by connecting both cores to the same pseudo random input data and comparing the output when the cores are not in reset state. In case C, the unwatermarked core has an inverted reset, so the core is working when the watermark is sending the signature. We embedded the signature S_1 ="153CA9F8" with a 12 bit preamble. Note that our correlation detection methods

use an ISI-free encoding. Therefore, the experimental results are not depending on the signature, unlike methods in [3].

TABLE II
DECODING RESULTS OF THE CORRELATION-BASED DETECTION METHOD OBTAINED FOR A *Des56* CORE.

Case	Board	Bit Error Rate in %	RMS in mV	SNR $h_3(t)$ in dB	SNR $h_2(t)$ in dB
Des56 Core					
Signature S_1, Decoding without packet repetition					
A	Spartan-3	0	-	3.6	-9.6
B	Spartan-3	18.5	0.698	1.9	-
C	Spartan-3	15.3	1.13	1.6	-
B	Virtex-II	6.2	2.44	3.4	2.5
C	Virtex-II	15.3	3.28	2.8	1.9
Signature S_1, Decoding using four repetitions					
B	Spartan-3	3.1	0.648	2.0	-
C	Spartan-3	6.2	0.987	1.6	-
B	Virtex-II	3.1	2.18	3.6	2.7
C	Virtex-II	3.1	2.93	2.9	2.2

The experimental results for the correlation detection method are shown in Table II. The decoding was done with the correlation of the highest frequency component of the corresponding impulse response. On both boards, this is the component $h_3(t)$ (see Table I). To enhance the results, the second strongest frequency component, on both boards $h_2(t)$, are additionally used.

The SNR values in Table II are small compared to the methods reported in [3]. One reason is that only the used frequency component counts to the signal and all other frequency components of the impulse response are calculated to the noise. Furthermore, the values reported in Table II are acquired by decoding only one signature pattern, whereas the decoding for the other methods uses the average of many repeated signatures to lower the noise level. The values in the lower half of Table II correspond to a decoding which uses the average of four repeated signature patterns.

Nevertheless, the BPSK and the enhanced robustness encoding method exceeds this method in forms of the bit error rate for these boards. However, this method is signature and board independent and can be further enhanced, e.g., for multiplex methods where multiple cores can concurrently send signatures.

VII. CONCLUSION

In this paper, we introduced a method for using the power pins as a communication channel for sending data from the cores out of the FPGA. The data may be received by decoding the measured power supply voltage. Possible applications include a) monitoring, b) portless debugging, and c) power watermarking. The advantage of this technique is that no further pins are required, which makes this method suitable if an additional communication channel is needed without alteration of the board or the pin assignment.

With our experimental setup consisting of a digital oscilloscope and a decoding algorithm running on a standard PC, no real time decoding is possible. However, by designing an integrated decoding device which is able to measure and concurrently decode the signal, real time communication should be possible.

We characterized the power communication channel for real FPGAs on two example boards and presented a method for a general characterization. A new encoding/decoding method based on correlation of the impulse response has been introduced. Furthermore, a *power watermarking* case study has been presented. Here, the power watermarking techniques, introduced in [2] and [3] are extended by an advanced encoding/decoding approach. Experimental results show that decoding is possible even if interferences from other cores are present. However, the decoding can be further improved if more shift registers or a higher repetition rate of per data packet are used. We expect this encoding/decoding method to scale to other (newer) devices as well. Furthermore, the protection of the transferred data using error correction codes is envisaged which should further improve the communication quality.

REFERENCES

[1] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," *Lecture Notes in Computer Science*, vol. 1666, pp. 388–397, 1999.

[2] D. Ziener and J. Teich, "FPGA Core Watermarking Based on Power Signature Analysis," in *Proceedings of IEEE International Conference on Field-Programmable Technology (FPT 2006)*, Bangkok, Thailand, Dec. 2006, pp. 205–212.

[3] D. Ziener and J. Teich, "Power Signature Watermarking of IP Cores for FPGAs," *Journal of Signal Processing Systems*, vol. 51, no. 1, pp. 123–136, April 2008.

[4] L. Shang, A. S. Kaviani, and K. Bathala, "Dynamic power consumption in Virtex-II FPGA family," in *FPGA '02: Proceedings of the 2002 ACM/SIGDA tenth international symposium on Field-programmable gate arrays*. New York, NY, USA: ACM Press, 2002, pp. 157–164.

[5] A. Chandrakasan, S. Sheng, and R. Brodersen, "Low-power CMOS digital design," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 4, pp. 473–484, 1992.

[6] I. Gel'fand, G. Shilov, and E. Saletan, *Generalized functions*. Academic Press New York, 1964.

[7] Digilent, Inc. Spartan-3 board. S3BOARD.cfm. [Online]. Available: www.digilentinc.com/info

[8] M. B. Gordy, "GA.M: A matlab routine for function maximization using a genetic algorithm," 1996. [Online]. Available: ftp://all.repec.org/RePEc/cod/html/Matlab/gordy.m

[9] J. Lach, W. H. Mangione-Smith, and M. Potkonjak, "Signature Hiding Techniques for FPGA Intellectual Property Protection," in *proceedings of ICCAD*, 1998, pp. 186–189.

[10] Kahng, Lach, Mangione-Smith, Mantik, Markov, Potkonjak, Tucker, Wang, and Wolfe, "Constraint-Based Watermarking Techniques for Design IP Protection," *IEEETCAD: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, 2001.

[11] D. Kirovski and M. Potkonjak, "Intellectual Property Protection Using Watermarking Partial Scan Chains For Sequential Logic Test Generation," in *ICCAD*, 1998.

[12] D. Kirovski, Y.-Y. Hwang, M. Potkonjak, and J. Cong, "Intellectual property protection by watermarking combinational logic synthesis solutions," in *proceedings of ICCAD*, 1998, pp. 194–198.

[13] A. B. Kahng, S. Mantik, I. L. Markov, M. Potkonjak, P. Tucker, H. Wang, and G. Wolfe, "Robust IP Watermarking Methodologies for Physical Design," in *Design Automation Conference*, 1998, pp. 782–787.

[14] D. Ziener, S. Aßmus, and J. Teich, "Identifying FPGA IP-Cores based on Lookup Table Content Analysis," in *Proceedings of 16th International Conference on Field Programmable Logic and Applications*, Madrid, Spain, Aug. 2006, pp. 481–486.

[15] T. Kean, D. McLaren, and C. Marsh, "Verifying the authenticity of chip designs with the DesignTag system," in *IEEE International Workshop on Hardware-Oriented Security and Trust, 2008. HOST 2008*, 2008, pp. 59–64.

[16] D. Ziener and J. Teich, "Evaluation of Watermarking methods for FPGA-based IP-cores," University of Erlangen-Nuremberg, Department of CS 12, Hardware-Software-Co-Design, Am Weichselgarten 3, D-91058 Erlangen, Germany, Tech. Rep. 01-2006, Mar. 2006.

[17] Xilinx, Inc. Virtex-ii platform fpgas: Complete data sheet. ds031.pdf. [Online]. Available: direct.xilinx.com/bvdocs/publications

[18] Opencores.org, "Opencores," *URL: www.opencores.org*.