FPGA Core Watermarking Based on Power Signature Analysis

Daniel Ziener [#1§], Jürgen Teich [*2]

[#]*Fraunhofer Institut for Integrated Circuits IIS*
Am Wolfsmantel 33, 91058 Erlangen, Germany
[1]znr@iis.fraunhofer.de

[*]*Department of Computer Science 12, University of Erlangen-Nuremberg*
Am Weichselgarten 3, 91058 Erlangen, Germany
[2]teich@cs.fau.de

***Abstract*— In this paper we introduce a new method to watermark FPGA cores where the signature (watermark) is detected at the power supply pins of the FPGA. This is the first watermarking method, where the signature is extracted in this way. We are able to sign cores at the netlist as well as the bitfile level, so a wide spectrum of cores can be protected. The power watermarking method works with all types of FPGAs, but with Xilinx FPGAs, we can integrate the watermarking algorithms and the signature into the functionality of the watermarked core. So it is very hard to remove the watermark without destroying the core. We introduce a detection algorithm which can decode the signature from a voltage trace with high probability. Additionally, a second algorithm is introduced which improves the detection probability in case of considerable noise sources. Using this algorithm, it is possible to decode the signature even if other cores operate on the same device at the same time.**

I. Introduction

In the 1970s, only basic functions like discrete logical gates were implemented on integrated circuits. With improvements in the chip manufacturing, the size of the transistors was drastically reduced and the maximum size of a die was increased as well. Now it is possible to integrate one billion transistors [1] on one chip. On the other hand, the market requires shorter product cycles. The only solution is to reuse cores, which have been written for other projects or were purchased from other companies. The number of companies that produce just cores constantly increases. The advantages of reuse of IP cores (Intellectual Property cores) are enormous. E.g., they offer a modular concept and fast development-cycles.

IP cores are licensed and distributed like software. One problem of the distribution of IP cores is the lack of protection against unlicensed usage. As the cores are provided, e.g., as netlist data, they can be easily copied like software. So there is only a small effort to get the illegal core to function. 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 modified.

[§]Part of this work was funded under the sixth Framework Programme (FP6) of the EU within the project "WorldScreen - Layered Compression Technologies for Digital Cinematography and Cross Media Conversion" Project No. 511333, http://www.worldscreen.org

Another approach is to hide a signature into the core, a so called watermark, which can be used as a proof of the original ownership. There exist many concepts and approaches on the issue of implementing a watermark into a core. But most of these concepts are not applicable due to the lack of verification capabilities. A good verification strategy is that the signature (watermark) can be read out only using the bought product. So no extra files or information must be obtained from the accused company.

Here, we present a strategy to verify the watermark by measuring the core supply voltage of an FPGA. The voltage can be sampled with a usual scope, analyzed and decoded with an algorithm developed to run on a PC. The decoded signature can be compared with the original signature, and so, the watermark can be verified. This method is not destructive and can be done using the given product only (see Fig. 1).

This work is organized as follows. In Chapter I, a short introduction and motivation of watermarking is given. In Chapter II, a short overview of related work for IP-Watermarking is given. Chapter III describes the analysis of the core voltage. Chapter IV presents the implementation, and Chapter V the embedding of the watermark. In Chapter VI and VII, the detection algorithms are described. Chapter VIII presents experimental results and Chapter IX concludes the paper.

II. Related Work

Hiding a unique signature into user data, such as pictures, video, audio, text, program code, or IP cores is called watermarking. Embedding a watermark into multimedia data is achieved by altering the data slightly at points, where human sense organs have lower perception sensitivity. For example, one can remove frequencies which cannot be perceived by the human ear by coding an audio sequence into an MP3 file. Now, it is possible to hide a signature into these frequencies, without decreasing quality of the coded audio sequence [2].

The watermarking of IP cores is different from multimedia watermarking, because the user data, which represents the circuit, must not be altered, since functional correctness must be preserved. Watermarking procedures can be categorized into two groups of methods: additive methods and constraint-based methods.

Additive methods have in common that the signature is added to the functional core, for example, by using unused

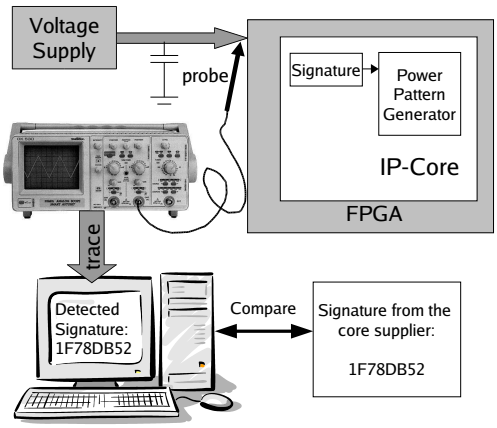

Fig. 1. Watermark verification using power signature analysis: From a signature (watermark) a power pattern inside the core will be generated that can be probed at the voltage supply pins of the FPGA. From the trace, a detection algorithm verifies the existence of the watermark.

lookup-tables in an FPGA [3]. The constraint-based methods were originally introduced by [4] 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 wire wrap of a scan-chain [5], preserve nets during logic synthesis [6], place constraints for CLBs in odd/even rows [7], or route constraints with unusual routing resources [7].

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 without additional information from the producer. The bitfile of an FPGA can be extracted by wire tamping the communication between the PROM and the FPGA. But only the approach presented in [3] has the possibility to detect the watermark from these bitfiles. 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. In this case, only the verification over a scan chain is possible [5].

Also the introduced approaches at the HDL- and netlist-levels turn out not to be applicable due to the lack of verification possibilities. The only exception is the scan chain approach, but a scan chain is very unusual in FPGA designs. However, many cores are delivered in HDL or at the netlist level, so a watermarking strategy for these cores would be very useful.

The problem of watermarking FPGAs is not the coding and insertion of a watermark, rather than the verification with an FPGA embedded in a system. So our following methods will concentrate on the verification of watermarks, too. There are four sources to get information of a design from: Bitfile, Ports, Power [8], and Electromagnetic (EM) radiation [9].

The basic idea behind our approach is to extract the core signature from the FPGAs power consumption pattern. This idea is new and differs from [8] and [9] where the goal of using power analysis techniques is not watermarking and intellectual property protection, but the detection of cryptographic keys and their security issues.

There is no way to measure the power consumption of an FPGA directly, only through the voltage or the current. We decide to measure the voltage of the core close to the voltage supply pins, so the smoothing from the plane and block capacities are minimal and so no shunt is required. Most FPGA have ball grid array (BGA) packages and the majority of them have vias to the back of the printed circuit board (PCB) for the supply voltage pins. So it is easy to measure the voltage with a scope on the rear side of the PCB.

III. Concept of Power Watermarking

The consumed power of an FPGA can be divided into two areas, the static and the dynamic power. The static power consumption is founded in the leakage current from CMOS transistors and does not change over time if the temperature is not changed. The dynamic power consists of the power of the short circuit current and the power of reloading the capacities the transistors and the wires. The short circuit current occurs, if both transistors, the PMOS and the NMOS, are conducting for a short time during the switching activity. Both parts of the dynamic power consumption depend on the switching frequency [10]. As shown in [11], the main part of the FPGA's dynamic power results from capacity reloading.

So, what happens to the core voltage, if many switching activities occur at the same time, like at a rising edge of a clock signal? First, we can discover a breakdown of the core voltage, and then an overshoot (see Fig. 2). The real behavior of the core voltage depends on the individual FPGA, the individual printed circuit board and the individual voltage supply circuits.

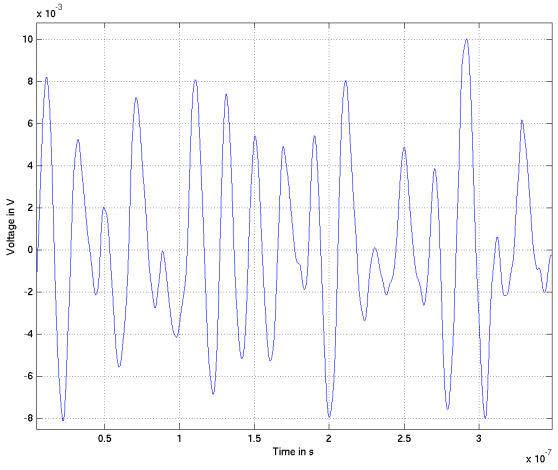

Fig. 2. A measured voltage signal from the voltage supply pin of an FPGA. Voltage breakdowns and overshoots can be seen.

In the spectrum of the voltage, we detect the clock frequency and even integer divisions of the clock frequency (see Fig. 3).

Now, we present two methods to encode a watermark into the core voltage characteristics. First, we vary the frequency and second, we change the amplitude.

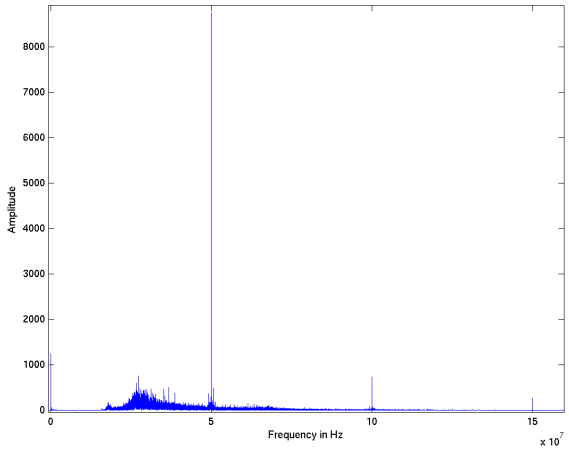

Fig. 3. The spectrum of the measured signal in Fig. 2. The clock frequency of 50 MHz and harmonics can be seen. Also, a peak at the half of the clock frequency is visible which is caused by switching activities from the logic.

In the first case, a watermark can be identified if we produce another frequency line in the spectrum of the core voltage, which is not an integral multiple or a rational fraction of the clock frequency. For achieving this, we need a circuit that consumes a considerable amount of power and generates a signature-specific power pattern, and a clock which can be identified in the spectrum. The power consumer can be for example an additional shift register. If we derived the clock source from the operational clock, we cannot definitely distinguish the frequency line in the spectrum from operational logic. Another opportunity is to generate a clock using combinatorial logic. This can be identified as a watermark, but the jitter of a combinatorial clock source might be very high, and no clean frequency line can be seen in the spectrum. This means, that we need a higher additional power consumption to make the watermark readable. Another drawback is that we have only limited possibilities to encode a reliable signature in these frequency lines.

In our approach, we therefore alter the amplitude of the interferences in the core voltage. The basic idea is to add a power pattern generator (e.g., shift registers), and clock them with the operational clock or an integer division. Further, we control these power pattern generator according to the characteristic watermark. A logical '1' lets the power consumer operate one cycle (e.g., perform a shift), a zero '0' leads to no operation. In the voltage profile over time, we detect higher amplitudes corresponding 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 is still active.

The advantage of power watermarking methods is that the signature can easily be read out from a given device. Only the core voltage of the FPGA must be measured and recorded. No bitfile is required, which needs to be reverse-engineered. Also, these methods work with encrypted bitfiles whereas methods where the signature is extracted from the bitfile fail. Moreover, we are able to sign netlist cores, because our watermarking algorithm does not need any placement information. So, also cores at this level can be protected.

IV. The Method

Our power watermarking method uses two shift registers, a big shift register for causing a recognizable signature-depending power consumption pattern, and a shift register storing the signature itself (see Fig. 1). The signature shift register is clocked by the operational clock and the output bit enables the power pattern generator. If the output bit is a '1', the power pattern register will be shifted at the next rising edge of the operational clock. At a '0', no shift is done. To avoid interference from the operational logic in the measured voltage, the signature is only generated during the reset phase of the core.

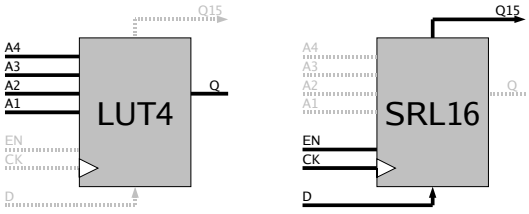

Fig. 4. In the Xilinx Virtex architecture, a lookup table (LUT4) can also be configured as a 16 Bit shift register (SRL16).

In some FPGA architectures (e.g., Xilinx Virtex), the lookup tables (LUTs) can also be used as a shift register [12]. A 4 Bit lookup table can also be used as a 16 Bit shift register (see Fig. 4). And, for example in the Xilinx Virtex architecture, the content of such a shift register can be adressed by the LUT input ports. So the shift register can be also used as a lookup table. 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, so 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.

Also, it is possible to initialize the content of the power consumption shift register shifted, which are also part of the functional logic. Only during the reset state, when the signature is generated, the functional logic can be initialized correctly. So, normal core operation cannot start before the signature was generated. The advantage is that the core is unable to operate at the beginning and only after "sending" the signature, the core works properly. Also, to avoid a too short reset time in which the watermark cannot be exactly detected, the right functionality will only be established if the reset state is longer than a predefined time. This prevents that the user can leave out or shorten the reset state with the result that the signature cannot be properly detected.

The signature itself can be implemented as part of the functional logic in the same way. Some lookup tables are connected together and the content, the function of the LUTs, represents

the signature. This principle makes it almost impossible for an attacker to change the content of the signature shift register.

The advantage of using the functional logic of the core also as a shift register is the reduced resource overhead for the watermarking and the robustness of this method, because these shift registers are embedded in the functional design, and it is hard if not impossible to remove shift registers without destroying the functional core. Also, our watermarking procedure is difficult to be detected in a netlist file, because the main part of the required logic for signature creation depends on the functional logic for the proper core. Another benefit is that our watermark cannot be removed by an optimization step during the mapping into CLBs (Configurable Logic Blocks).

V. Embedding of the watermark

In this chapter, we describe the procedure of watermarking a core. The watermarking procedure is easy to use and consists of only 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 for the generation of the power pattern generator and attaches the corresponding control signal from the control logic in the wrapper (see Fig. 5).

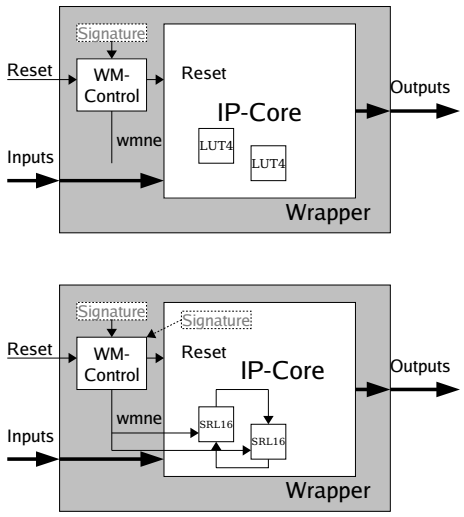

Fig. 5. The core and the wrapper before (above) and after (below) the netlist alternation step. The signal "wmne" is an enable signal for shifting the power pattern generator shift register.

The wrapper contains the control logic for emitting the watermark and the shift register, holding the signature. If functional lookup tables are used as signature shift register, we add or convert this shift register in the second step and so the wrapper contains only the control logic. Some control signals have no sink yet, because the sink will be added in the second step (e.g., the power pattern generator shift register). So we must use synthesis constraints to prevent the synthesis tool from optimizing these signals away. The ports of the wrapper are the same for the core, so we can easily integrate this wrapper into the hierarchy. The control logic shifts the signature shift register, while the core is in reset state. Also, the power pattern shift register is shifted corresponding 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 positions of the content in the shift register are not in the correct state. The control logic shifts the register content into the correct position and leaves the reset state to start the normal operation mode.

The translation of four input lookup tables (LUT4) of the functional logic into 16 Bit shift registers (SRL16) is done at the netlist level. The usage of a LUT4 as a SRL16 is only possible if the LUT4 is not part of a multiplexer logic. This is not possible, because the additional shift logic and the multiplexer share common resources in a slice. Also, if the lookup table is a part of an adder, the mapping tool splits the lookup table and the carry chain. In these two cases, additional slices would be required, so we do not convert these lookup tables into shift registers.

The above conversion is done by a program which reads an EDIF-netlist and also writes a modified EDIF-netlist. First, the program reads all LUT4 instances, checks if the following logic is not a "MUXF5" or a "MUXCY" or a "XORCY". Then, the instances are coverted to a shift register (SRL16), if required, initialized with the shifted value and connected to the clock and the watermark enable (wmne) signal to these shift registers. Always two shift registers are connected together to rotate their contents. Finally, the modified netlist is written.

VI. Detection Algorithm

The measured voltage will be probed, digitized and decoded by a signature detection algorithm (see Fig. 6). To decode the digitalized voltage signal, the sampling rate, the clock frequency of the shifted signature and the bit length of the signature is needed. The clock frequency can be extracted from the Fast Fourier Transformation of the measured signal. Our detection algorithm consists of five steps: downsampling, differential step, accumulation, phase detection and quantization (see Fig. 6). After the quantization step, the decoded signature can be simply compared with the signature from the core supplier bit by bit.

As mentioned before, the main characteristic caused by a switching event is the breakdown of the voltage followed by a subsequent overshoot. This results in extreme slopes. Our detection algorithm can find each rising edge as follows:

First, the measured signal will be downsampled from the recorded sample rate to the quadruple of the clock frequency, so each signature bit is represented by four samples. Then, the discrete derivative of the signal will be calculated. This transforms the rising edges of the switching events into peaks. The easiest way to calculate the discrete derivative is to take the difference over time (see Fig. 7).

$$D(k) = s(k) - s(k-1), \tag{1}$$

where s is the sampled probed voltage signal and D the discrete derivative and k denotes the sample index.

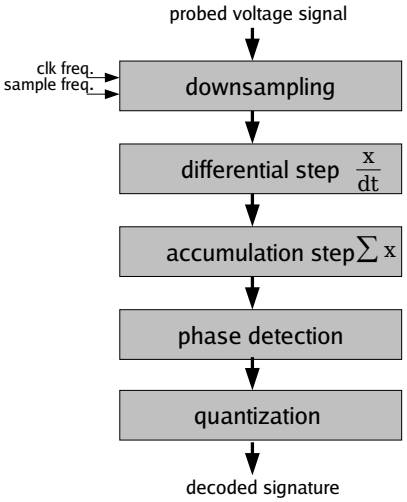

Fig. 6. The different steps of the detection algorithms.

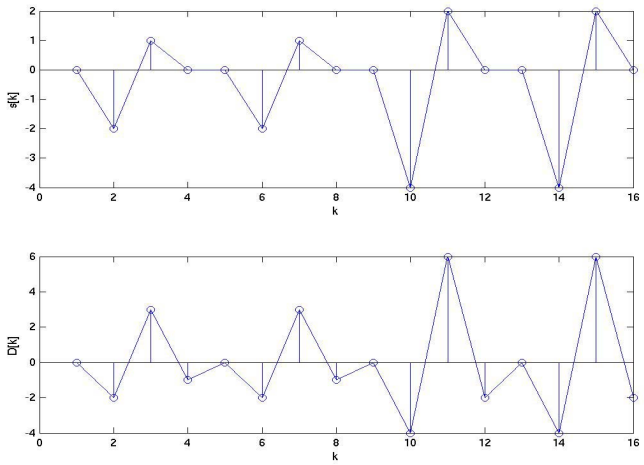

Fig. 7. An example voltage signal which represents the signature "0011" (above). The example voltage signal after the differential step (below).

Since the signature is repeated many times during the reset state, the signal can be accumulated and averaged to reduce the noise level. To accumulate the coherent pattern, we need to know the bit length of the signature. If we record a longer signal sequence, we can accumulate more patterns and reduce noise and also switching events which do not belong to the power consumption register of the watermarking algorithm, for example from other cores on the chip. The disadvantage is that we would need a longer time for the reset.

After this third step, we have a signal where each signature bit is represented by four samples. But only one sample has the information of the rising edge. Since the measurement is not synchronized with the FPGA clock, the phase (position) of the relevant sample of a bit is unknown. We divide the signal into four new signals, where one signature bit is represented in one sample. The four signals have a phase shift of 90^o to each other. Let

$$S[k], \qquad k = 0, 1, .., 4n - 1 \tag{2}$$

denote the sampled voltage signal after the accumulation step. Then, we obtain the four following phase shifted signals

$$S_0 = S[4k], \qquad k = 0, 1, .., n - 1 \tag{3}$$
$$S_{90} = S[4k + 1], \qquad " \tag{4}$$
$$S_{180} = S[4k + 2], \qquad " \tag{5}$$
$$S_{270} = S[4k + 3], \qquad " \tag{6}$$

where S is the accumulated signal and S_0, S_{90}, S_{180} and S_{270} are the phase signals and n denotes the length of the signature (see Fig. 8).

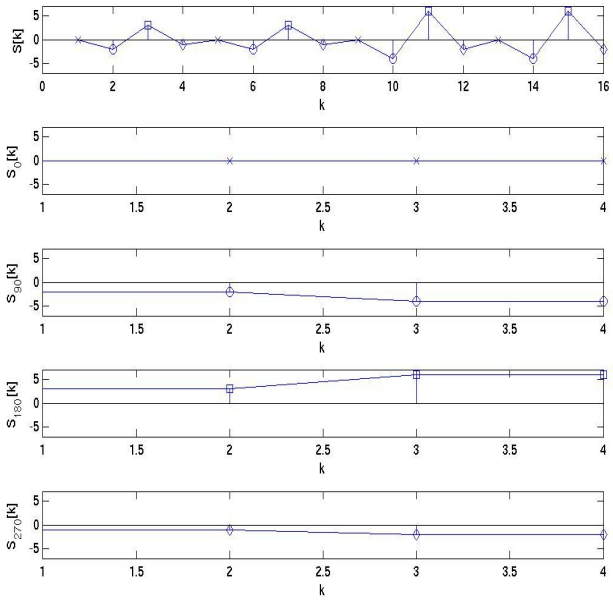

Fig. 8. The example voltage signal after the accumulation step (above) and the four phase shifted signal (below). Here, S_{180} corresponds to the right phasing.

We are able to win the phasing of the signal if we calculate the mean value of each phase-shifted signal. The maximal mean value corresponds to the correct phasing, because the switching event should cause the greatest rising edge in the signal.

Now, we have a signal in which each sample is represented by the accumulated switching activities on one bit of the signature. The decision if the sample corresponds to a signature bit '1' or '0' can be done by comparing the sample value with the mean value of the signal. If the sample value is higher than the mean value, the algorithm decides a '1', in the other case a '0'.

VII. Enhanced Robustness Encoding Method

Experimental results (see Chapter VIII) have shown that the decoding of the signature works well, but on some targets problems occur in the decoding of signatures with long runs of '1' followed by many zeros, like "1111111100000000". For

the first 8 bits, we see a huge amplitude and then a phase in where the amplitude is faded out (see Fig. 9). The phase can be many clock cycles long and could lead to a wrong detection of the following bits.

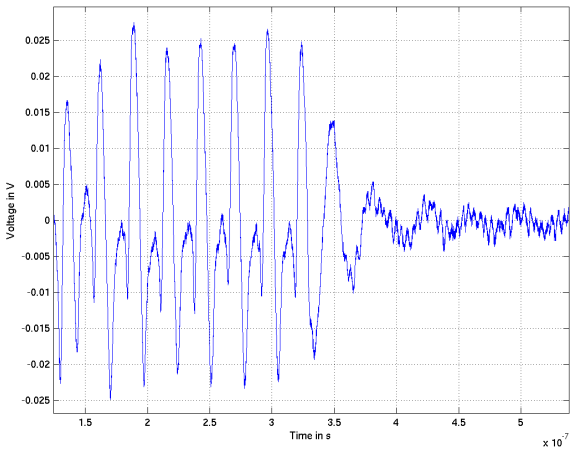

Fig. 9. Measured voltage supply signal when sending "FFFF0000" with a large power pattern generator shift register.

This fading out amplitude might be produced by a resonance circuit which consists of the capacities and resistance from the power supply plane and its blocking capacities. This behavior is dependent on the printed circuit board and the power supply circuit.

To avoid such a false detection, we spread up the signature bits, so the time for sending one bit is higher than the swing out from the printed circuit board. We spread up the bits by repeating one bit m times, where m is the number of steps one cycle of the power consumption shift register needs. If we connect two SRL16 together, one cycle for this shift register needs 32 steps. If the reset phase ends and we have finished sending one bit, the content in the shift register which also represents a part of the logic of the core, is in the correct position.

The detection algorithm differs for this method. First, the signal will be downsampled and the approximate derivation will be calculated as in the original method (see Chapter VI).

Now, we average the signal to suppress the noise. But here, the length of one signature word is the length of the signature (n) multiplied by the number of times each bit is sent (m). As defined before, this is the square of the length of the signature.

$$D[k], \qquad k = 0, 1, .., K-1 \tag{7}$$

$$N = \left\lfloor \frac{K}{4m \cdot n} \right\rfloor, \tag{8}$$

$$S = \frac{1}{N} \sum_{r=0}^{N-1} D[4m \cdot n \cdot r, .., 4m \cdot n \cdot r + 4m \cdot n - 1], \tag{9}$$

where D is the voltage signal after the differential step with index k and N is the number of repetitions of the pattern in D.

The phase detection of the shift clock is the same as in the original method (see Chapter VI), but we also need the position p where a new signature bit starts. This is done in a loop to detect this position. In the beginning, we assume that the starting position is the beginning of our trace ($p = 0$). First, we accumulate m successive values, where m is the repetition of one bit:

$$A_p[q] = \sum_{u=0}^{m-1} S_\phi[u + p + mq], \qquad q = 0, 1, .., n-1 \tag{10}$$

where S_ϕ is the signal after the phase detection step. Now, we subtract the mean value and generate the absolute value and calculate the sum of it.

$$F_p = \sum_{q=0}^{n-1} \left| A_p[q] - \frac{1}{n} \sum_{a=0}^{n-1} A_p[a] \right| \tag{11}$$

F_p identifies how good our signature bit starting position p fits to the real position. Now, we shift our trace one value ($p = 1$) and calculate the fitting value again, and so on. This is done m times. The starting position with the best fitting value will be used.

The decoding of the watermark signature is done like in the first method (see Chapter VI) by comparing the sample values with the mean value of the samples.

VIII. Experimental Results

For experimental results, we used two FPGA-Boards, the Digilent Spartan-3 Starter Board [13], and a board with a Xilinx Virtex II XC2V250 FPGA. On the second board, many other components, like an ARM micro controller or interface chips are integrated to demonstrate that the algorithm is also working on multi-chip boards. The Spartan 3 board operates with a clock frequency of 50 MHz, the Virtex II board with 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 Ohm wire with a 50 Ohm terminating resistor. This wire is directly soldered on the vias. We have used a DC block element and a 25 MHz high pass filter to filter the DC component and the interferences of the switching voltage controller. We used a LeCroy Wavepro 7300 scope 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 decrease in bandwidth. The signal of the length of 200 μs is recorded on the internal hard disc of the scope. This signal file is transferred to a personal computer and analyzed there.

The functionality of this method is evaluated with a DES Core with 56 Bit from opencores.org [14] and an arithmetic coder core. Arithmetic coders are used within JPEG2000 [15]. Research within the European Union project "WorldScreen" focuses on hardware implementation of JPEG2000 encoders.

After the synthesis step, only 16 out of 715 lookup tables from the DES56 core have been transformed into SRL16 and a 32 Bit signature has been added. Also, for the arithmetic coder

core, 92 out of 1332 lookup tables have been transformed into SRL16. Both core inputs were stimulated with a pseudo random sequence, generated by a linear feedback shift register to simulate input data.

The decoded sequence was compared with the encoded signature from the core to evaluate the bit error rate. Further from the signal, where the bit decision is done, two quality indicators were calculated. One is the signal to noise ratio (SNR) of these signals. Because we make a threshold decision, SNR values under 4 dB are difficult to decode. We also calculate the SNR from the decoded sequence, so bit errors falsified our SNR. In these cases, the real SNR is lower than the calculated SNR. The second indicator called bit gain is the difference from the mean level of the bits and the threshold level. This indicator shows how big the difference of the voltage swing between ones and zeros of the signature is. Also, the root mean square (RMS) from the recorded signal without the DC part is measured. Figure 10 shows a good signal before the bit decisions with an SNR value of 37 dB. The signal shown in Figure 11 is of lower quality and has a SNR of 9 dB.

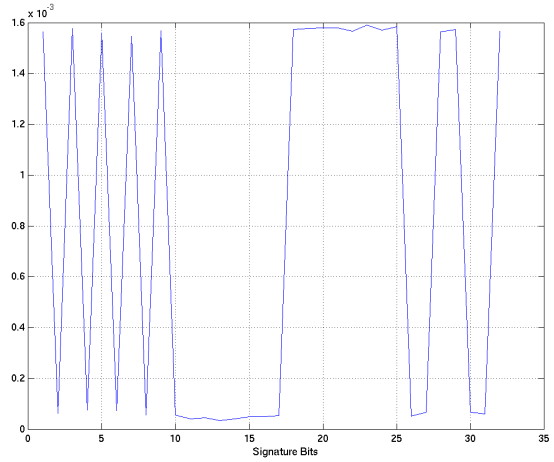

Fig. 10. A good signal for the bit decisions with an SNR value of 37 dB.

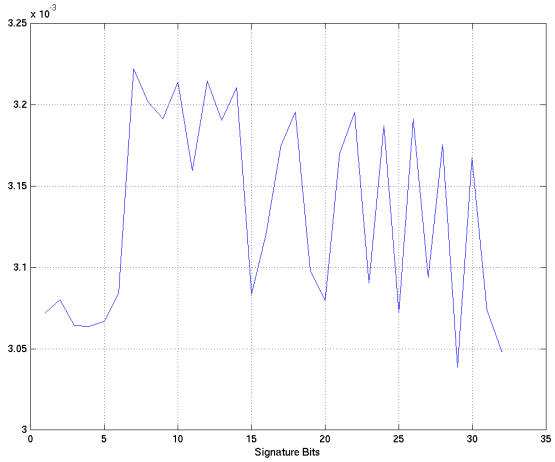

Fig. 11. A signal of lower quality with a SNR of 9 dB, but without bit errors.

TABLE I
RESULTS OF THE ORIGINAL METHOD

Case	Board	Bit Error Rate in %	Signal RMS in mV	SNR in dB	Bit Gain
Signature S_1					
A	Spartan3	0	0.376	8.5	0.126
B	Spartan3	9.4	0.511	4	0.112
A	VirtexII	21.9	0.821	4	0.277
B	VirtexII	31.2	1.047	4	0.263
Signature S_2					
A	Spartan3	0	0.374	8.5	0.147
B	Spartan3	3.1	0.513	4.5	0.137
A	VirtexII	6.2	0.859	4	0.561
B	VirtexII	0	1.063	8.5	0.632
Signature S_3					
A	Spartan3	6.2	0.380	4	0.111
B	Spartan3	12.5	0.516	3	0.122
A	VirtexII	na	0.841	3.5	0.368
B	VirtexII	9.4	1.073	3.5	0.381

First, the original method described in Chapter VI is evaluated (see Table I). We decoded the signature with both boards and the DES56 core where only 16 lookup tables are transformed into SRL16. We have evaluated two cases, one where only the watermarked core is implemented (case A) and one where the watermarked core and the original core is 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 compare the output when the cores are not in the reset state. We embedded three signatures (S_1, S_2 , S_3) in the core. The Signature S_1 is "5C918CBA" and represents a realistic random signature. Signature S_2 is "333333333" and signature S_3 is "FF335500". With these signatures, we can evaluate the decoding method with different bit toggle rates.

Table I shows that decoding is not always working without bit errors, but here, we have transformed only 16 lookup tables into SRL16.

In case A, the detection works better than in case B. In case B, more logic is used, but this logic is in the reset state. Nevertheless, the clock tree is still active which can be seen in the higher signal RMS value. The signature S_3 is difficult to decode, because there are many equal bits lumped together and so the printed circuit board works as a resonator (see Chapter VII).

To evaluate the enhanced robustness method described in Chapter VII, we use the same test cases and implement only the signature S_3, which is harder to decode (see Table II). Also, we define two additional test cases. In C, the unwatermarked core has an inverted reset, so the core is working when the watermark is sending the signature. In D, two cores are working, while the signature is emitted. Not all combinations in D are possible because the FPGA is too small for three cores. Additionally, we have evaluated this method with the arithmetic coder core.

Table II shows that the detection of the watermarked sig-

TABLE II

RESULTS OF THE ENHANCED METHOD

Case	Board	Bit Error Rate in %	Signal RMS in mV	SNR in dB	Bit Gain
DES 56 Core					
A	Spartan3	0	0.384	22	0.087
B	Spartan3	0	0.508	23	0.110
C	Spartan3	0	1.21	22	0.109
D	Spartan3	0	2.15	10.5	0.0539
A	VirtexII	0	0.794	18	0.067
B	VirtexII	0	1.022	22.5	0.191
C	VirtexII	0	2.698	12	0.067
Arithmetic Coder Core					
A	Spartan3	0	0.618	37	0.758
B	Spartan3	0	0.617	38	0.720
C	Spartan3	na	4.488	3	0.216
A	VirtexII	0	1.347	37.5	1.248
B	VirtexII	0	1.343	37	1.191

TABLE III

RESULTS WITH DECREASED RECORD TIME

	200 μs		100 μs		50 μs	
Case	SNR in dB	Bit Gain	SNR in dB	Bit Gain	SNR in dB	Bit Gain
A	22	0.087	21.5	0.091	16	0.090
B	23	0.110	19.5	0.110	16.5	0.111
C	22	0.109	18	0.107	18.5	0.107
D	10.5	0.054	10	0.057	9.5	0.061

nature works much better than with the original method. The decoding for the DES56 core works fine even if one or two of the same DES56 cores operate at the same time the signature is emitted. Also here, we use only 16 SRL16 in the DES56 core. For the arithmetic coder core, we use more lookup tables and if no other core operates, the decoding is better than for the DES56 core, but if another arithmetic coder core is active, the decoding is impossible. The signal RMS indicates that the arithmetic coder core has a very high toggle rate.

In Table III, we decreased the recording length to see the impact of the quality of our results. This is done with the DES56 core in all four cases. The quality degenerates but with the recording length of 50 μs, it is still possible to detect the watermark without bit errors in case D even if two other cores are simultaneously active.

IX. CONCLUSIONS

We have presented a watermark technique for FPGA cores where the signature is easily extracted over the power pins of the FPGA. So, it is possible to read out the watermark only with the given device without further information from the vendor of the product. We have shown how the watermark is easily integrated into the core. With Xilinx FPGAs, it is possible to integrate the watermark algorithm and the signature into the functionality of the core, so it is hard to remove the watermark, and only very few additional resources were required for control. In Chapter VI, we described the detection algorithms, and experimental results showed that the functionality of the core is not altered and that it is possible to detect the signature over the power trace of the FPGA. Also, we introduced an enhanced robustness technique, which improves the decoding of the signature highly. With this enhanced decoding method, we are able to decode the signature even if other cores are simultaneously active on the same hardware device. We also introduced quality indicators to evaluate the result of the decoded signature. With these indicators, we can show how reliable the technique is.

The experimental results have shown that decoding is not possible in all cases, but we can improve the quality of the results if we transform more lookup tables into shift registers or extend the recording time. In addition, the signature width can be increased and so, a higher number of signatures, error codes and cyclic redundancy checks (CRC) are possible.

REFERENCES

[1] Xilinx, Inc. Next-Generation Virtex Family From Xilinx to top one Billion Transistor Mark. 03131_nextgen.htm. [Online]. Available: www.xilinx.com/prs_rls/silicon_vir/

[2] L. Boney, A. H. Tewfik, and K. N. Hamdy, "Digital watermarks for audio signals," in *International Conference on Multimedia Computing and Systems*, 1996, pp. 473–480. [Online]. Available: citeseer.ist.psu.edu/boney96digital.html

[3] 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. [Online]. Available: citeseer.ist.psu.edu/lach98signature.html

[4] 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. [Online]. Available: citeseer.ist.psu.edu/kahng01constraintbased.html

[5] D. Kirovski and M. Potkonjak, "Intellectual property protection using watermarking partial scan chains for sequential logic test generation," in *ICCAD*, 1998. [Online]. Available: citeseer.ist.psu.edu/218548.html

[6] 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. [Online]. Available: citeseer.ist.psu.edu/article/kirovski98intellectual.html

[7] 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. [Online]. Available: citeseer.ist.psu.edu/kahng98robust.html

[8] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," *Lecture Notes in Computer Science*, vol. 1666, pp. 388–397, 1999. [Online]. Available: citeseer.ist.psu.edu/kocher99differential.html

[9] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, "The em side-channel(s)," in *CHES '02: Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems*. London, UK: Springer-Verlag, 2003, pp. 29–45.

[10] A. Chandrakasan, S. Sheng, and R. Brodersen, "Low-Power CMOS Digital Design," 1992. [Online]. Available: citeseer.ist.psu.edu/chandrakasan95low.html

[11] 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.

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

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

[14] Opencores.org. Basic des crypto core: Overview. overview. [Online]. Available: www.opencores.org/projects.cgi/web/basicdes

[15] D. S. Taubman and M. W. Marcellin, *JPEG 2000: Image Compression Fundamentals, Standards and Practice*. Norwell, MA, USA: Kluwer Academic Publishers, 2001.