Efficient and Side-Channel-Secure Block Cipher Implementation with Custom Instructions on FPGA

Suvarna Mane
Department
Virginia Tech
Blacksburg, USA
suvarnam@vt.edu

Mostafa Taha
ECE Department
Virginia Tech
Blacksburg, USA
mtaha@vt.edu

Patrick Schaumont
ECE Department
Virginia Tech
Blacksburg, USA
schaum@vt.edu

***Abstract*—The security threat of side-channel analysis (SCA) attacks has created a need for SCA countermeasures. While many countermeasures have been proposed, a key challenge remains to design a countermeasure that is effective, that is easy to integrate in existing cryptographic implementations, and that has low overhead in area and performance. We present our solution in the context of an embedded design flow for FPGA. We integrate an SCA-resistant custom instruction set on a soft-core CPU. The SCA resistance is based on dual-rail precharge logic. A balanced-interleaved data format, combined with a novel memory organization, ensures that we can support both logic operations as well as lookup tables. The resulting countermeasure applies to a broad class of block ciphers. We demonstrate our results on an Altera Cyclone-II FPGA with Nios-II/s processor for a 128-bit Advanced Encryption Standard (AES) T-box implementation. We show SCA improvement of more than 400X for a system-wide electro-magnetic attack that covers both the FPGA and off-chip memory (SSRAM). This comes at an overhead of 2.7x in performance and 1.15X in area. Using comparisons with related work, we demonstrate that this represents an excellent trade-off between SCA resistance, (software and hardware) design complexity, performance, and circuit area cost.**

***Index Terms*—Side Channel Analysis; Custom Instructions; Softcore CPU; FPGA; Cryptography.**

I. Introduction

Since their introduction over a decade a ago, side-channel analysis (SCA) techniques have been successfully used to extract secret keys from cryptographic algorithms by exploiting side-channel information such as execution time, power consumption, or electromagnetic emissions. The field of SCA has been intensively researched for attacks and countermeasures. Several recent results highlight the risk of SCA to commercial, deployed systems with a trustworthiness requirement. This includes the use of SCA to extract keys from Virtex-II FPGA [15] and Virtex-4/5 FPGA [16] bitstream encryption, from the Mifare DESFire contactless card [13], from the Keeloq keyless entry system [8], and from the Atmel Cryptomemory non-volatile memory [2]. It's reasonable to assume that most of these devices were not designed with SCA in mind. Indeed, if SCA attacks are considered part of the threat model of a design, one can introduce suitable SCA countermeasures to hamper these attacks. Modern smart-cards, for example, have built-in countermeasures against SCA attacks, as well as against active (fault-based) attacks.

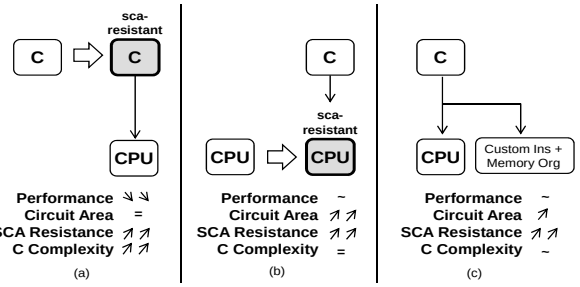

Figure 1. SCA resistant design by (a) C source code transformation, (b) Dedicated circuit styles and (c) Customized CPU

Nevertheless, the design and implementation of a side-channel countermeasure is a complex and error-prone process: literature shows a long string of attacks against countermeasures [17]. Our work is motivated by the need for an easy-to-use countermeasure, applicable to a wide range of designs and usable within a standard FPGA design flow. We consider protection of a general class of block ciphers that use logic operations and lookup tables. This includes AES, DES, and many others. We propose our methodology in the context of embedded designs with a CPU, and we develop side-channel resistance for cryptographic software executing on the processor.

The design of side-channel countermeasures is complex because side-channel leakage is a byproduct of the implementation of a cryptographic algorithm. Predicting the amount of side-channel leakage from, say, cryptographic software in C is difficult. Our objective is to systematically remove side-channel leakage while keeping a reasonable cost in circuit area and performance degradation. Figure 1 illustrates three approaches applicable to the context of embedded processors. The first, Figure 1a, transforms the crypto-software into an implementation without exploitable leakage, for example by using masking [4]. This countermeasure is usually algorithm-specific, and requires in-depth understanding of cryptographic operations. Moreover, masking becomes very complex under advanced SCA techniques [7]. The second approach, Figure 1b, is to implement the CPU in a SCA-resistant circuit style. Past research has shown that these techniques are very

expensive in hardware - costing 3 to 15 times the original circuit area [17] - and thus not applicable to a complete CPU. Our approach, Figure 1c, is to use a customized CPU, with a custom instruction-set and an optimized memory organization. This design configuration is supported by soft-core CPU in mainstream FPGA families.

Our work is not the first to suggest a customized CPU for side-channel resistant implementations; previous proposals have included masking-based [3], [21] and hiding-based [5], [18] designs. However, we will demonstrate that our solution offers several advantages over these proposals. The remainder of this paper is organized as follows. In the next section we discuss a few preliminaries, including a brief review of Dual-rail Precharge (used by our countermeasure), and a review of the building blocks of modern block ciphers. After that, we present the components of our solution, and show how we can efficiently map SCA-resistant block ciphers into FPGA. In Section IV, we implement a full block cipher (AES) with our technique, and we analyze circuit cost, performance, and SCA resistance. Section V elaborates on these results, discusses the limitations of our approach as well as related work. Finally we conclude the paper.

II. Preliminaries

In this section, we briefly review two important preliminaries of our proposed solution: Dual-rail Precharge Logic, and the overall structure of modern block ciphers.

A. Principle of Dual-rail Precharge Logic (DPL)

The cause of side-channel leakage is data-dependent processing. In CMOS logic, such processing gives data-dependent signal transitions, which in turn results in data-dependent power consumption or radiation. The idea of Dual-rail Precharge Logic (DPL) is to eliminate side-channel leakage at the level of the implementation.

DPL can be achieved as follows. First, every data bit in the circuit is stored and processed in complementary form. For example, for every logic operation `a and b`, there is a matching complementary operation `not(a) or not(b)` which is simultaneously executed. Second, every complementary data pair `(a, not(a))` is pre-charged to `(0,0)` before every evaluation. When combined, these two properties result in constant power consumption: every evaluation has an active $0 \rightarrow 1$ transition, either on the true net, or else on the complementary net.

DPL has been applied in many different forms since it was first proposed, including ASIC, FPGA, and software [5], [9], [11], [22]. Authors have also identified sources of residual leakage, including early evaluation and imbalance between complementary pairs [10]. However, DPL has demonstrated substantial reduction of side-channel leakage in prototypes. For this reason, we have selected it in our countermeasure.

B. Modern Block Ciphers

In this paper, we focus our efforts on protecting a broad class of symmetric-key algorithms known as block ciphers.

Table I
Block Cipher Operations (Encryption w/o Key Schedule)

Cipher	Structure	SBOX (*in* x *out*)	Operations
AES	SPN	8x32	XOR
Blowfish	Feistel	8x32	XOR, ADD32
Camellia	Feistel	8x8	XOR
CAST-256	SPN	8x32	XOR
Clefia	Feistel	8x8	XOR
DES	Feistel	6x4	XOR
GOST	Feistel	4x4	XOR, ADD32, ROT
KASUMI	Feistel	7x7+9x9	XOR, ROT
PRESENT	SPN	4x4	XOR
Serpent	SPN	4x4	XOR, ROT

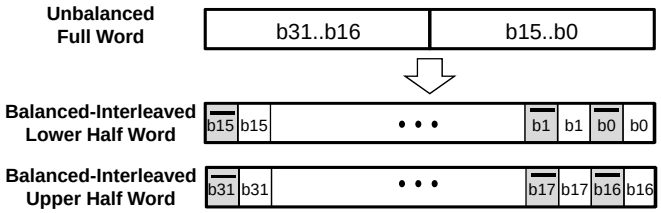

Figure 2. Balanced Interleaved data format

Block ciphers encrypt a block of plaintext into ciphertext through successive round transformations. As illustrated in Table I, and earlier observed by Kaps [12], the majority of modern block ciphers are constructed from a limited set of operations, including substitutions with lookup-tables (SBOXes), and operations such as Xor, modular addition, rotations, and shift. Furthermore, round transformations have a common structure, and use either a substitution-permutation network (SPN), or a Feistel network. Of course, within this framework, there are important differences among block ciphers as well, such as the number and size of lookup tables used, and the detailed configuration of the operations.

In the following sections, we implement DPL countermeasures for software implementations of block ciphers on soft-cores. We develop specific techniques to handle lookup tables, and logic operations. Our benchmarks and experiments have focused on AES-128 executing on a Nios/II core configured in a Cyclone-II FPGA; but our results apply to other block ciphers as well.

III. Our Solution

We implement a side-channel resistant block cipher by creating DPL versions of both the lookup tables as well as the logic operations in hardware. These modules are efficiently integrated into the soft-core processor using the custom-instruction set interface. This way, SCA-resistant block ciphers can be executed as a sequence of custom instructions. Non-crypto software, on the other hand, is written using the regular instruction set without performance hit. The custom-instruction hardware for lookup tables is built from on-chip RAM macro's. Research has demonstrated that such dedicated structures increase side-channel resistance [19], and we further improve this technique. We next discuss the three components

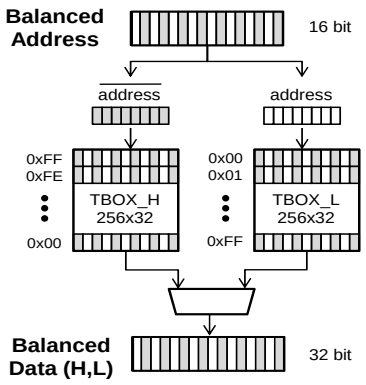

Figure 3. Balanced-Interleaved T-box Organization

of our solution: the organization of data, the memory organization for lookup tables, and the system integration of SCA-resistant block cipher hardware into software.

A. SCA-resistant Data organization

We need a data format that is compatible with the requirements of DPL and uses the word-level organization of an embedded system. Figure 2 shows our data arrangement. Each 32-bit word is split into two balanced half-words, and each bit from the original word is interleaved with an associated complementary bit. We call this representation a balanced-interleaved (BI) format. The logical and physical proximity of complementary bits improves symmetry between the bits (e.g. similar electrical loads), and in turn, this improves SCA resistance. Indeed, at the logical level, adjacent bits will share adjacent storage locations. In embedded architectures, storage organization may use a wordlength which is different from the processor wordlength; a 32-bit memory may be organized, for example, as two half-word banks. Keeping complementary bits adjacent ensures that they will share the same physical storage bank. Furthermore, at the physical level, adjacent bits will have closely related routing patterns on the FPGA and PCB, improving symmetry.

A consequence of using a balanced-interleaved format is that each 32-bit operation from the original, unprotected block cipher, requires expansion into two balanced operations, each processing a balanced half-word.

B. Memory Organization for Lookup Tables

Because lookup tables are so common in block ciphers, we use a dedicated approach to implement side-channel resistant lookup tables using the RAM macro's of the FPGA fabric. We use the AES T-box implementation as a case study. The T-box is a lookup table with 8 input bits and 32 output bits. The T-box is defined by grouping several steps of the AES round transformation; for the purpose of explaining our method, we treat the T-box simply as an 8x32 lookup table. The complete AES algorithm requires five different T-box tables.

The secure T-box design shown in Figure 3 uses a balanced-interleaved data organization. An 8x32 T-box thus needs two 8x32 balanced-interleaved tables, each storing a half-word of the original T-box with its complementary bits. Each balanced-interleaved table is stored in a separate RAM macro. In order to achieve balancing in the address decoding logic, we follow the storage order suggested in [19], namely that complementary RAM macro's require complementary addresses. The difference with our design, however, is that the complementary RAMs do not store complementary data: the data within each RAM is already balanced. Summarizing, our proposed memory organization for lookup tables achieves side-channel resistance by combining three elements. First, the use of RAM cells reduces side-channel leakage because the increased logic density they offer. Second, the use of balanced-interleaved addressing for the overall lookup table. Third, the use of balanced-interleaved data storage for lookup table content.

Table II
SCA-RESISTANT INSTRUCTION SET FOR AES

Instruction	Return Value
`CONV_INV(a)`	$0, 0, .., a[2], a[0]$
`CONV_BIL(a)`	$\overline{a[15]}, a[15], .., \overline{a[0]}, a[0]$
`CONV_BIH(a)`	$\overline{a[31]}, a[31], .., \overline{a[16]}, a[16]$
`B_XOR(a,b)`	balanced-interleaved $xor(a, b)$
`B_TBx_L(a)`	balanced-interleaved lookup-table (lower)
`B_TBx_H(a)`	balanced-interleaved lookup-table (upper)

Each AES T-box has its own `B_TBx_H(a)` and `B_TBx_L(a)`; x=0,1,2,3,4

C. System Integration

An important, but often overlooked, aspect of side-channel countermeasures is the system integration. On an embedded processor, SCA-resistant encryption is just one of the many tasks handled by software. We have integrated our countermeasures as custom instructions into a soft-core processor. A custom-instruction interface offers the ability to introduce custom-hardware modules in the execution stage of a RISC pipeline.

Table II shows the side-channel resistant instruction set for AES. These instructions are implemented in custom hardware using DPL. `CONV_INV(a)` extracts the even bits from a word, and thus converts balanced-interleaved format into direct form. `CONV_BIL(a)` and `CONV_BIH(a)` generate balanced-interleaved form from the lower resp. higher half-word of the input argument `a`.

The round function for a T-box based AES only requires a balanced XOR, which can be supported through a single custom instruction `B_XOR(a,b)`. Move, shift and rotate operations are compatible with balanced-interleaved arguments, so that no custom instruction is needed for those.

The AES T-box has 5 different T-box tables. There is a `B_TBx_L(a)` and a `B_TBx_H(a)` to access the lower resp. higher half of each T-box table. These instructions are specific for the AES block cipher; a different block cipher would need to use different lookup tables. However, it is perfectly feasible to make the lookup tables fully reconfigurable, so that they can be programmed with the Sbox content required for a specific block cipher. The approach to implement lookup tables in the processor is an important difference with earlier work

by Chen [5], and we will show how this brings considerable performance gain.

The AES T-box algorithm can be written in C by making use of custom instructions embedded as inline assembly macro's. The pre-charge operation can be supported from C as well, as illustrated in the snippet below. Note the use of `volatile` to prevent the removal of precharge by an optimizing compiler.

```
volatile int t1, t2, t3;
t1 = 0;              // precharge
t1 = B_TB0_L(in);    // T-box0 lower word
t2 = 0;              // precharge
t2 = B_TB1_L(in);    // T-box1 lower word
t3 = 0;              // precharge
t3 = B_XOR(t1, t2); // XOR
```

A strong feature of this approach is that it is fully compatible with the existing memory hierarchy of an embedded system. Variables can be stored into RAM in balanced-interleaved form, and they will maintain their low side-channel leakage provided that pre-charge is properly implemented. Thus, our approach is independent of the number of processor registers; it will not run out of foreground storage (in contrast to e.g. [20]).

Our design makes sure that all sensitive data variables are always in balanced-interleaved format outside custom hardware boundary. Thus, transactions of these data variables to other storage elements such as, cache, external memories etc, do not result in side-channel leakage. Storing balanced interleaved format in background memory may still cause side-channel leakage due to asymmetry in the physical layout of background memory. We will analyze this in the next section of the paper.

IV. Experimental Setup and Results

To demonstrate that our solution improves the resistance against SCA, this section presents the experimental results based on real attacks.

A. Experimental setup

The designs are implemented on an Altera DE2-70 evaluation board, that has a Cyclone-II EP2C70F896C6 FPGA device and NiosII softcore. Our system incorporates a 32-bit NiosII/s (50MHz, pipelined) processor, an offchip memory (SDRAM or SSRAM) and communication peripherals (UART, GPIOs). We use Quartus-II with SOPC-builder to integrate the desired peripherals into the system. Electromagnetic emissions are captured with an ETS-LINDGREN EM probe (Model 7405-903) and are sampled on an Agilent Oscilloscope DSO5032 (300MHz bandwidth, 2GSa/s sampling rate). A Correlation Power Attack (CPA) [14] using the Hamming weight model on the T-box output is used to analyze the acquired EM traces. An oscilloscope is configured to average out 32 consecutive traces so as to reduce the noise in the acquired traces.

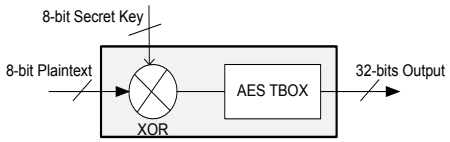

Figure 4. Single T-box Experiment

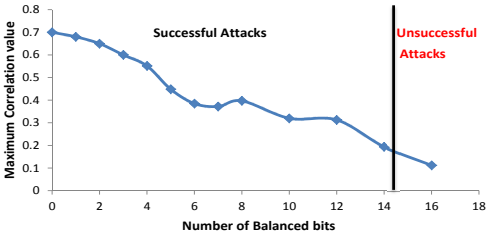

Figure 5. Security Improvement: Single Tbox test

Our experiment is divided into two parts, a proof-of-concept experiment (Single T-box attack) and a real world implementation (128-bit AES T-box attack). The following subsections give the details.

B. Single T-box Experiment

In this experiment, we target an attack on single T-box operation to evaluate security gain due to specialized memory organization and balanced-interleaved data format. As illustrated in Figure 4, this test design incorporates essential components of a block cipher (AES) i.e. logical XOR and T-box lookup. SCA-resistant XOR and lookup table operations are implemented in a custom hardware and are accessed through custom instructions. We vary the number of balanced bits in a BI dataword from 0 (unsecure) to 16 (fully secure) and perform an SCA attack on the output of a lookup table to evaluate its resistance against SCA attacks. For each of these experimental steps, we reconfigure XOR operation and change the format of T-box table contents to have required number of balancing bits.

Figure 5 shows the results, where the maximum correlation value for correct key guess is plotted against the number of balanced bits present in a dataword. The correlation is calculated for 2000 traces, at its best attack point. It can be seen that the correlation of the correct key guess reduces with increasing number of balanced bits. For completely secure case, the correlation value reduces to 0.11 at 2000 traces. We could not attack fully balanced design successfully with 170000 averaged traces. This shows that our countermeasure achieves a significant security improvement.

C. 128-bit AES-Tbox Prototype

In the second part of our experiment, we implement an SCA-resistant AES T-box (128-bit) prototype to evaluate its efficiency in terms of security, performance and cost. We use the same platform as that of single T-box experiment with two different configurations of offchip memory (SDRAM and SSRAM). The T-box lookup tables are implemented in

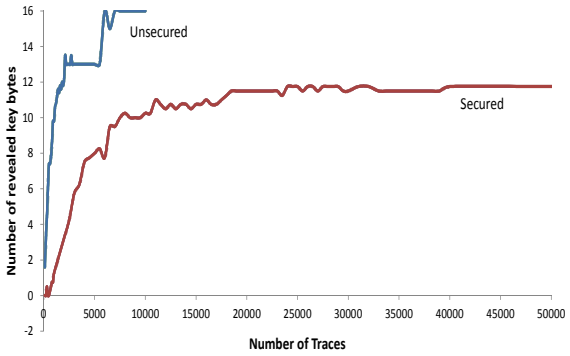

Figure 6. AES-TBOX implementation: NiosII/s + SDRAM

onchip RAM macros and offchip memory is used for program execution and stack. The software uses custom instructions for secure operations and includes hardcoded secret key in a balanced-interleaved format. All intermediate variables are precharged to 0 before they are used for next operation. We attack first round of AES and conduct CPA analysis to evaluate its SCA-resistance.

We perform SCA attacks on unsecure and secure implementations for SDRAM and SSRAM configurations. We have used several different secret keys and Table III lists the average security gain for the set keys. An unsecure AES implementation on NiosII/s with SDRAM offchip memory reveals 12 key bytes at around 1600 traces whereas, the secure implementation needs 40000 traces to reveal 12 key bytes. This results in an overall security gain of 25x at 75% success rate. Figure 6 plots the number of key bytes revealed as a function of number of traces for SDRAM configuration. In case of SSRAM configuration, an unsecure implementation achieves 75% success rate at an average of 633 traces, whereas we could not attack secure implementation for 300000 traces. Figure 7 shows the correlation trace of correct key byte for 300000 samples. This results in security gain of at least 474, which significantly differs from that of an SDRAM configuration. We investigate the possible reasons for this difference in the next section.

Table III
AES IMPLEMENTATION: SECURITY

Configuration	MTD (# revealed keys)		Security gain
	Unsecure	Secure	
NiosII/S + SDRAM	1600 (12)	40000 (12)	25
NiosII/S + SSRAM	633 (12)	300000 (0)	>474

This security improvement comes at the cost of performance and area overhead. A secure implementation occupies extra logic for customized hardware and needs to split every 32-bit sensitive dataword into two 32-bit balanced words. Additionally, all variables need to be precharged before they can be reused. This overhead of the additional instructions causes a small performance degradation. Our secure implementation is 2.7 times slower than unsecure implementation and takes 15% more area. Table IV enlists these results.

Table IV
AES IMPLEMENTATION: AREA AND PERFORMANCE

Configuration	Area (LEs, M9K)		Cycle count	
	Unsecure	Secure	Unsecure	Secure
NiosII/S + SDRAM	3452, 143	3889, 161	13839	36977
NiosII/S + SSRAM	2814, 31	3252, 49	7375	19980

Area of a system with CPU, memory controller and custom hardware.

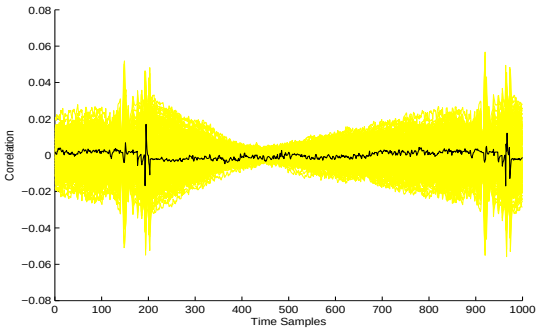

Figure 7. Attack results on secure implementation: NiosII/s + SRAM. *Trace of correct key guess (here, first key byte) is plotted in black, while all other key guesses are in yellow(gray). The buried trace means unsuccessful attack.*

V. ANALYSIS AND COMPARISON

In this section, we analyze the possible causes for residual side-channel leakage in case of SDRAM-system and compare our work with related published secure implementations.

A. Impact of PCB Layout

The location of peripheral chips on a PCB board has a significant impact on the security of overall system. Figure 8 depicts the layout of DE2-70 board. We can see that, SSRAM has more symmetric location with respect to the CycloneII FPGA than that of SDRAM. A 32-bit SDRAM is configured as two 16-bit memory banks, whereas SSRAM is a 32-bit memory chip. With this layout, SDRAM does not always offer adjacent data-pin locations for a complementary bit-pair. This creates an imbalance between direct and complimentary bitlines irrespective of their balanced format. On the other hand,the SSRAM has a more symmetric data-pin pattern, which routes complementary bit lines together and thus, reduces the residual side-channel leakage. Note that storing the secret data in offchip memories makes the system susceptible to a simple probing attack on its chip pins. In our work, we assume that such attacks do not happen.

B. Related Implementations

In this section, we compare our solution with other secure implementations. As shown in Table V, these implementations target different technologies, different countermeasures and different cryptographic algorithms. As the attack methods are not standardized, it is not a straight-forward process to compare them on the same scale. Therefore, the table should be consulted while at the same time referring to the original publications by Barte *et. al.* [3], Chen *et.al.* [5], Ambrose *et. al.* [1], and Regazzoni *et. al.* [18].

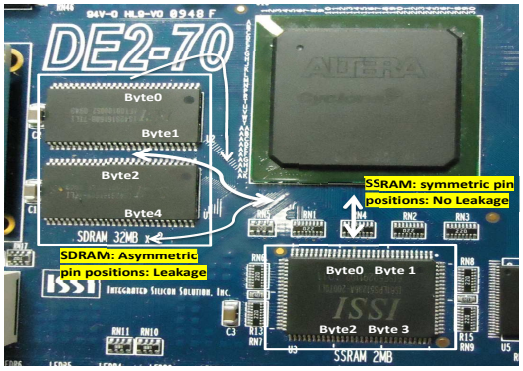

Figure 8. Impact of PCB Layout on Residual Leakage

Table V
RELATED WORK: COMPARISON

Work	Technology, Base processor	Implementation	Security gain/Area overhead/Performance degradation
[3]	Spartan-3, MicroBlaze	Masking, DES	2X / 1.34X[1] / –
[20]	Virtex-4, Leon3	Masking, AES	3.5X / – / 2X
[5]	Spartan-3E, Leon3	Hiding (VSC), AES	20X / 3.3X / 6.5X
[18]	ASIC 180nm, OpenRISC1000	Hiding (MCML), PRESENT	– / 2.65X / –
Our Design	CycloneII, NiosII/s	Hiding, AES-Tbox	>474X / 1.15X[2] / 2.7X

1 This number represents area overhead in terms of slice-count.
2 Area of a system with only processor and SRAM memory controller.

Compared to this earlier work, our design is very systematic, making the design phase simpler than above mentioned implementations. We believe that, for FPGA implementations, it exceeds above-mentioned solutions in terms of the trade-off between security, performance, area and design flexibility.

VI. CONCLUSION

Security against side-channel attacks are an important concern with increased use of embedded systems in security applications. This paper reports an efficient and secure embedded system design on FPGA by using industrial design flow. We use a novel memory organization technique and interleaved data format in combination with a hiding countermeasure. Though, we have demonstrated our results on an Altera FPGA for AES-Tbox implementation, the methodology is portable to other FPGA platforms for majority of the block ciphers. We discuss how location of peripheral offchip components on PCB board plays an important role in the overall security evaluation. Our experimental results establish the feasibility of proposed methodology to implement an embedded system to achieve desired security at reasonable cost.

VII. ACKNOWLEDGMENTS

This research was supported in part by National Science Foundation Grant no. 1115839.

REFERENCES

[1] J.A. Ambrose, S. Parameswaran, A. Ignjatovic. MUTE_AES: A Multiprocessor Architecture to prevent Power Analysis based Side Channel Attack of the AES Algorithm *ICCAD 2008, pp. 678 -684.*

[2] J. Balasch, B. Gierlichs, R. Verdult, L. Batina, and I. Verbauwhede. Power Analysis of Atmel CryptoMemory - Recovering Keys from Secure EEPROMs. In Topics in Cryptology - *CT-RSA 2012*, The Cryptographers' Track at the RSA Conference, Lecture Notes in Computer Science 7178, O. Dunkelman (ed.), Springer-Verlag, *pp. 19-34,* 2012.

[3] L. Barthe, P. Benoit, L. Torres. Investigation of a Masking Countermeasure against Side-Channel Attacks for RISC-based Processor Architectures. *FPL 2010: 139-144.*

[4] S. Chari, Charanjit S. Jutla, J. R. Rao, P. Rohatgi. Towards Sound Approaches to Counteract Power-Analysis Attacks. *CRYPTO 1999: 398-412.*

[5] Z. Chen, A. Sinha, P. Schaumont. Implementing virtual secure circuit using a custom-instruction approach. *CASES 2010: 57-66.*

[6] Z. Chen, P. Schaumont. Virtual Secure Circuit: Porting Dual-Rail Precharge Techniques into Software on Multicore IACR ePrint Archive 2010/270 (2010)

[7] J.S. Coron, E. Prouff, M. Rivain. Side Channel Cryptanalysis of a Higher Order Masking Scheme. *CHES 2007: 28-44.*

[8] T. Eisenbarth, T. Kasper, A. Moradi, C. Paar, M. Salmasizadeh, M. T. Manzuri-Shalmani. On the Power of Power Analysis in the Real World: A Complete Break of the KeeLoqCode Hopping Scheme. *CRYPTO 2008: 203-220.*

[9] S. Guilley, L. Sauvage, P. Hoogvorst, R. Pacalet, G. Bertoni, and S. Chaudhuri. Security Evaluation of WDDL and SecLib Countermeasures against Power Attacks *IEEE Transactions on Computers (2008) 57 (11): 1482-1497.*

[10] S. Guilley, L. Sauvage, F. Flament, V. Vong, P. Hoogvorst, R. Pacalet. Evaluation of Power Constant Dual-Rail Logics Countermeasures against DPA with Design Time Security Metrics. *IEEE Transactions on Computers (2010 Jan 1) 59 (9): 1250-1263.*

[11] P. Hoogvorst, G. Duc, J.L. Danger. Software Implementation of Dual-Rail Representation. *COSADE 2011.*

[12] J. Kaps, G. Gaubatz, B. Sunar. Cryptography on a Speck of Dust *IEEE Computer Magazine, 40(2):38-44, 2007.*

[13] T. Kasper, D. Oswald, C. Paar. Side-Channel Analysis of Cryptographic RFIDs with Analog Demodulation. *RFIDSec 2011: 61-77*

[14] S. Mangard, E. Oswald, T. Popp, Differential Power Analysis in *Power Analysis Attacks: Revealing the Secrets of Smart Cards.*, Springer, 2007.

[15] A. Moradi, A. Barenghi, T. Kasper, C. Paar. On the vulnerability of FPGA bitstream encryption against power analysis attacks: extracting keys from xilinx Virtex-II FPGAs. *ACM Conference on Computer and Communications Security 2011: 111-124*

[16] A. Moradi, M. Kasper, C. Paar. Black-Box Side-Channel Attacks Highlight the Importance of Countermeasures - An Analysis of the Xilinx Virtex-4 and Virtex-5 Bitstream Encryption Mechanism. *CT-RSA 2012: 1-18*

[17] A. Moradi, A. Poschmann. Lightweight Cryptography and DPA Countermeasures: A Survey. *Financial Cryptography Workshops 2010: 68-79.*

[18] F. Regazzoni, A. Cevrero, F.X. Standaert, S. Badel, T. Kluter, P. Brisk, Y. Leblebici, P. Ienne. A Design Flow and Evaluation Framework for DPA-Resistant Instruction Set Extensions. *CHES 2009: 205-219.*

[19] S. Shah, R. Velegalati, J.P. Kaps, D. Hwang. Investigation of DPA Resistance of Block RAMs in Cryptographic Implementation on FPGAs. *Reconfig 2010: 274 - 279.*

[20] S. Tillich, M. Kirschbaum, A. Szekely. SCA-Resistant Embedded Processors- The Next Generation. *ACSAC 2010: .*

[21] S. Tillich, M. Kirschbaum, A. Szekely. Implementation and Evaluation of an SCA-Resistant Embedded Processor. *CARDIS 2011: 151-165.*

[22] K. Tiri, I. Verbauwhede. A digital design flow for secure integrated circuits *IEEE TCAD (2006 Jan 1) 25 (7): 1197-1208.*