Design of Low-Power DPA-Resistant Cryptographic Functional Units

Zeynep Toprak[†]
zeynep.toprak@epfl.ch
Yusuf Leblebici[†]
yusuf.leblebici@epfl.ch
Ajay K. Verma[‡]
ajaykumar.verma@epfl.ch
Paolo Ienne[‡]
paolo.ienne@epfl.ch

Ecole Polytechnique Fédérale de Lausanne (EPFL)
[†]School of Engineering
[‡]School of Computer and Communication Sciences
CH–1015 Lausanne, Switzerland

Christof Paar
cpaar@crypto.rub.de

Ruhr-Universität Bochum
Communication Security Group (COSY)
D–44780 Bochum, Germany

ABSTRACT

A design methodology is proposed to implement instruction-set extensions for cryptographic processors and hardware accelerators with the aim to improve their potential robustness against *Differential Power Analysis (DPA)* attacks. The approach has a high-level component that is based on identifying the critical units and functions in a design with respect to DPA resistance, and a gate-level component that is based on implementing the critical units using an inherently DPA-resistant circuit design style. Our methodology can be used to automatically identify and to isolate the DPA-critical nodes, as well as to map them onto a robust cell library that is compatible with conventional deep-submicron CMOS technologies.

1. INTRODUCTION

For a long time, the main concern of the IT security community was to secure traditional computer networks, such as LANs, intranets and the Internet. The next generation of IT applications might look quite different, however: most future pervasive computing applications will exhibit security solutions which are different from, say, building firewalls for a corporate network. In particular, the role of cryptographic engineering will become more crucial.

By definition almost all devices in a pervasive network are embedded nodes, and providing security for such devices is heavily dependent on the device hardware and firmware. Most of the challenges will come from intrinsic or economic limitations of the embedded nodes: these will include a moderate available computing power, a very low manufacturing cost, and often very limited energy budgets. These limitations are independent of cryptographic applications—the latter will only add to the computational and energy burden. Yet, additional challenges to pervasive embedded nodes come from side-channel attacks which can use information such as instantaneous power consumption or electromagnetic radiation to break the code. Such attacks are very hard to resist but robustness to them is fundamental in many applications, such as Smart Cards. Generally, improving the resistance against power-based attacks requires both hardware and software countermeasures.

Successful design of secure pervasive nodes will critically depend on the availability of design methodologies encompassing all these problems and helping designers to explore solutions which are at once good for performance and security. We present in this paper a first step toward a largely automated methodology to implement instruction-set extensions which will speed up cryptographic applications and will make a processor robust to *Differential Power Analysis (DPA)* [7]. This is one of the most powerful techniques to perform side-channel attacks. DPA uses the fact that the instantaneous power consumption of the circuit implementing the cryptographic algorithm contains useful information—that is, is correlated to the value of the secret key bits and to the internal state. Such correlation is used in statistical analysis to compare different hypotheses on the secret key.

In this paper, we present a low power circuit design methodology to implement DPA-resistant cryptographic processing units for pervasive computing applications. Our main contribution is twofold: (1) an innovative use of a known circuit technique to develop secure units resistant to attacks based on power consumption variations and sufficiently robust for automatic EDA flows, and (2) a novel identification technique to put a minimum of hardware in a secure unit for maximal performance gain and comprehensive DPA-resistance.

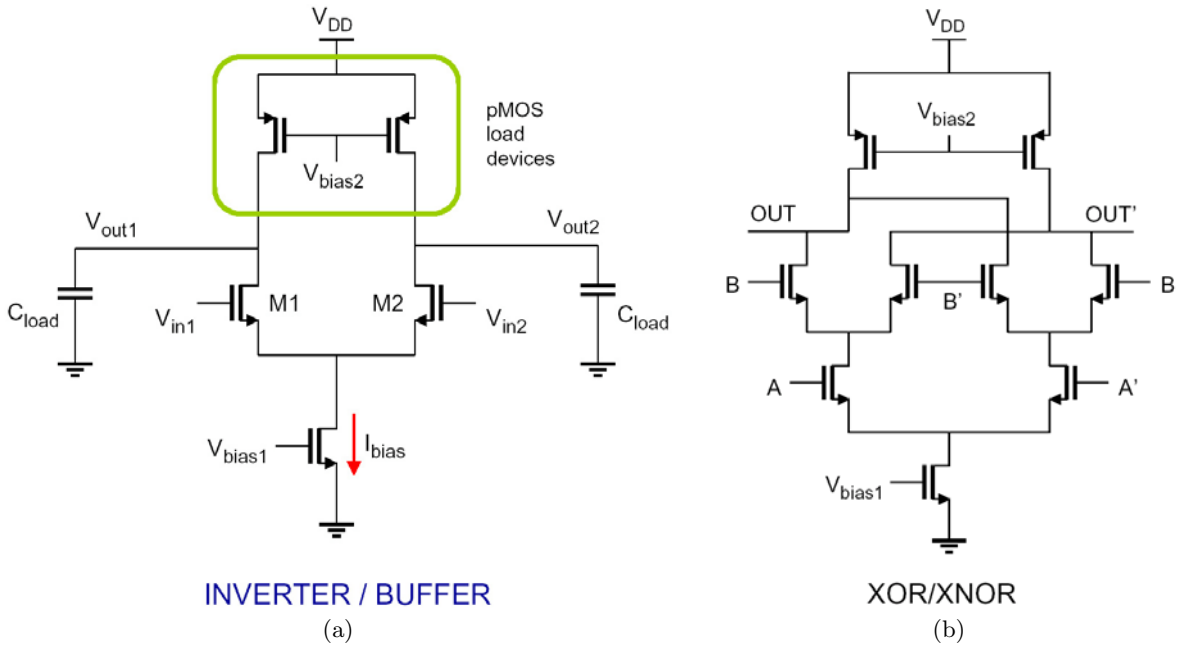

Figure 1: Schematics of (a) an MCML inverter and (b) an MCML XOR/XNOR gate.

2. RELATED WORK

DPA attacks are *known-plaintext* or *known-ciphertext* attacks of a cryptographic algorithm. At the heart of DPA is the possibility for an attacker of identifying one or more *DPA selection functions*. A selection function computes some internal state of the algorithm as a function of a hypothesis on the key and of known variables such as the plaintext or ciphertext. After measuring several power traces using varying input/output data over different encryptions, the attacker can class the traces in different groups depending on a key guess and on the corresponding value of the selection function: if such groups have some statistical difference (such as a nonnull difference of the average trace at some time), the attacker has found a correlation between a key guess, some internal state of the algorithm, and the power consumption—in other words, he/she has substantiated the specific hypothesis on the key.

The need for secure hardware platforms which counteract DPA attacks has been known since 1998/99 when SPA/DPA was developed by Kocher et al. [7]. Since then, many publications on software countermeasures have been presented but only a few approaches concerning hardware countermeasures have been published: among them are passive power supplies [11], self-timed circuits [8], dual-rail logic [9], and SABL logic design [14]. All of these approaches use special design styles which usually do not fit well in classic design flows: they require special design skills, have higher design cost and risk, and/or consume more power than standard CMOS. We will follow the lead of some of these authors, but we will try to minimise the use of special techniques to the very minimum part of the processor critical for security.

3. DESIGN OF DPA-RESISTANT FUNCTIONAL UNITS

The circuit-level implementation of DPA-resistant logic gates requires systematic use of circuit techniques that: (i) have significantly suppressed power supply current levels, (ii) do not produce prominent current spikes or fluctuations during the switching events, and (iii) do not exhibit a significant input pattern-dependence with respect to current drawn from the power supply [14]. It is worth noting that the classical CMOS logic gates do not fare particularly well in any of these categories, and therefore, are not considered to be a good choice for DPA-resistance, in general. The CMOS logic gate draws a large amount of current from the power supply during each low-to-high transition of its output node voltage, in order to charge up the output load capacitance through the pull-up (pMOS) network. Therefore, the magnitude of current as well as the temporal occurrence of these current pulses depend on the applied input pattern, and on the fanout load conditions of the gate.

Several different circuit design styles have already been explored as possible candidates for better DPA-resistance, including differential circuit techniques [14, 13]. In particular, a dynamic differential circuit technique (Sense Amplifier Based Logic, SABL) was considered as a viable alternative for implementing logic functions that are inherently immune against activity detection based on remote monitoring of the power supply current. However, due to the fact that the SABL circuit operation is charge-based (dynamic operation), the time-domain behaviour of the circuit depends critically on the parasitic output load capacitances. Thus, even minute imbalances in the layout parasitics may, in certain cases, offset the expected advantages of the circuit concerning its DPA-resistance [14]. Also, the temporal behaviour of each gate is strongly dependent on the fanout load of the gate, which eventually complicates the design of larger systems that contain a large number of logic gates.

Current Mode Logic (CML) circuits have also been considered as possible candidates for building DPA-resistant logic blocks, but they were not fully explored as a viable alterna-

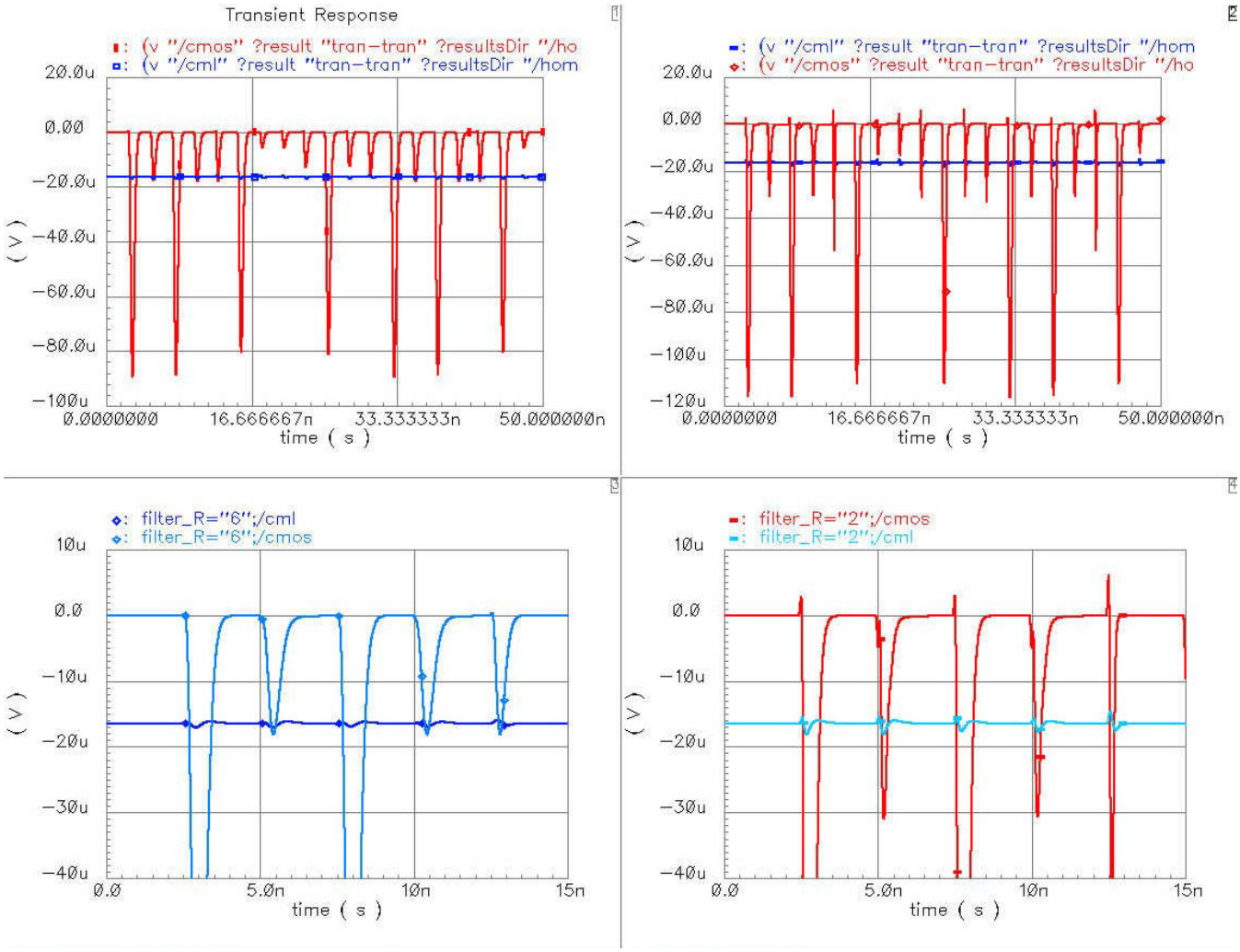

Figure 2: Simulated instantaneous power supply current during all possible input transitions for a CMOS XOR gate (spiky trace) and an MCML XOR gate (almost constant trace). Power supply variation monitored with a 1GHz current probe (left) and with a 3GHz current probe (right). Close-up views of the waveforms are shown at the bottom.

tive mainly due to the impression that any current mode circuit must necessarily consume a large amount of power. In the following, we present a very robust and low power implementation of the *MOS Current Mode Logic (MCML)* circuit style that exhibits about two orders of magnitude smaller power supply current variation in comparison to classical CMOS (for the same fanout load), while dissipating approximately the same amount of power.

Operation of MCML circuits are based on the principle of re-directing (or switching) the current of a constant current source through a fully differential network of input transistors, and utilising the reduced-swing voltage drop on a pair of complementary load devices as the output [15, 12]. The circuit diagrams of an MCML inverter and of an XOR/XNOR gate are shown in Figure 1. True differential operation of the circuit with small output voltage swing ensures fast switching times. Note that the propagation delay is proportional to the output swing, and independent of the power supply voltage. Other advantages include better noise immunity compared to classical CMOS logic circuits, and much smaller switching noise generation.

The operation principle already suggests that the power consumption is static—i.e., the circuit must dissipate the same amount of current continuously—regardless of the switching activity. This fact, together with the reduced output voltage swing, can be utilised effectively to achieve a very high switching speed, especially in comparison to classical CMOS logic gates [6]. Typically, it is possible to achieve a factor of 2–3× improvement in terms of switching speed over classical CMOS logic, and this has long been considered as the sole advantage of MCML circuit style over classical CMOS. It is also interesting to note that the power dissipation of the MCML gate remains essentially independent of the switching frequency as well as the fanout load conditions. Hence, for very high-speed operation, the power dissipation of an MCML gate will become *smaller* than that of a classical CMOS gate, which continues to increase as a linear function of the switching frequency [12, 10].

From the DPA-resistance point-of-view, it can be seen that the supply-current variation of the MCML gate will

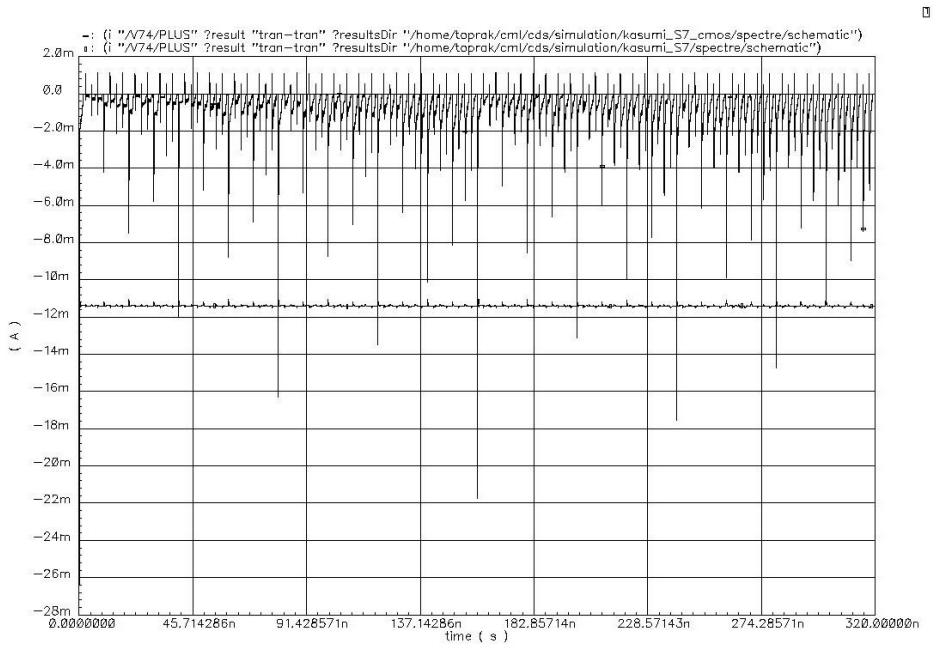

Figure 3: Simulated instantaneous power-supply current for all possible input combinations for a CMOS and an MCML 7-input Kasumi S-box.

remain significantly smaller during switching events, compared to that of a classical CMOS gate. At the same time, the magnitude of the supply-current variation is largely independent of the applied input vector, as well as of the fanout load capacitance. If the transistor sizing is done to satisfy modest speed constraints (e.g., a typical switching speed of 400MHz), the amount of static current dissipation can be reduced dramatically while preserving all of the advantages concerning the DPA-resistance, at a lower speed. To demonstrate this point, we have designed elementary Boolean function gates, both in classical CMOS logic style and in the current-limited MCML design style, using 0.18μm CMOS technology parameters. Both circuit styles were designed to operate at a nominal switching frequency of 400MHz, under the same output load conditions. The ratio between the power dissipation of the MCML XOR gate and the classical CMOS XOR gate was found to be less than a factor of two at this nominal frequency, which compares quite favourably with other DPA-resistant circuit styles.

4. EXPERIMENTAL RESULTS

Figure 2 shows the simulated instantaneous power-supply current drawn by a classical two-input CMOS XOR gate during all of the 12 possible input transitions, compared to that of an MCML XOR gate (monitored using two current probe models, with 1GHz bandwidth and 3GHz bandwith, respectively). It can be seen that the magnitude of current peaks generated by the CMOS XOR can be as high as 120μA, while the power-supply current fluctuation generated by the MCML XOR gate is limited to 1μA. In addition, the supply-current variation of the CMOS XOR exhibits a very strong input pattern dependence, while the minute current peaks generated by the MCML XOR gate remain largely independent of the particular input patterns applied to the gate. Note that the average power dissipation of the CMOS XOR gate and the MCML XOR gate are approximately the same in this example, yet the detectable indicators of switching activity are reduced by two orders of magnitude in the MCML circuit.

In order to demonstrate the utility of the current-limited MCML gates in DPA-resistant design, we have constructed the 7-input Kasumi S-box function (cf. 3GPP TS 35.202 Technical Specification version 3.1.1) that consists of 105 two-input AND gates and 77 two-input XOR gates. The power-supply currents of the classical CMOS version and the MCML version are shown in Figure 3, for all 512 possible input combinations. The peak current fluctuation of the classical CMOS realization is seen to be in the order of 28mA, while the current fluctuation of the MCML version remains confined to a narrow band of about 0.5mA, around the constant value of 11.5mA. A close-up view of the supply-current variation of the S7 Kasumi S-box function block clearly indicates significant input-pattern dependence of the classical CMOS version. In contrast, the power supply current of the MCML version does not exhibit any noticeable variation that depends on the applied input patterns.

Thus, we demonstrate that the proposed reduced-current MCML circuit technique could be used as a viable method for constructing DPA-resistant logic function blocks.

5. DPA-RESISTANT PROCESSORS

Many vendors of embedded processors now propose customisable architectures. Tensilica Xtensa [5], ARC ARC-

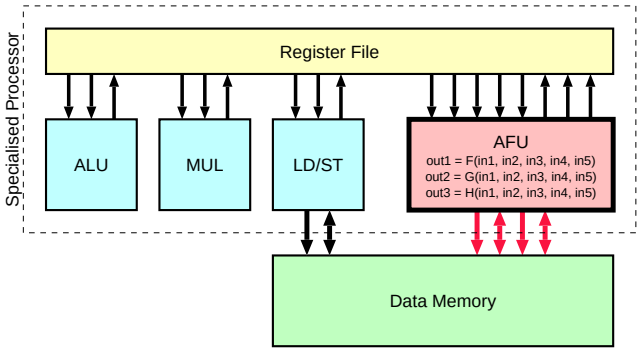

Figure 4: Typical extensible processor with a five-input three-output *Application-specific Functional Unit (AFU).*

tangent [3], STMicroelectronics ST200 (a.k.a. Lx) [2], and MIPS CorExtend [4] are only a few of the many application-specific customisable architectures which have emerged in the recent years. These extensible CPUs offer to the customers the possibility to tailor the processor to a specific application. Roughly, such processor cores accept extensions in the form of application-specific functional units as shown in Figure 4. Such functional units should be used to implement in MCML logic style DPA-critical operations and thus achieve the desired robustness.

We discuss in this section the problem of identifying automatically the most effective DPA-resistant instruction-set extensions for a given architecture and for an application specified in a high-level language.

5.1 DPA Criticality

We call $G(V, E)$ the *Direct Acyclic Graphs (DAGs)* representing the dataflow of each basic block in the algorithm; the nodes V represent primitive operations and the edges E represent data dependencies. This representation is classic in compilers and can be easily obtained by well known analyses of the cryptographic application high-level code.

DPA attacks depend on the possibility of substantiating guesses on some bits of the key by partitioning power traces into sets whose difference is correlated to some internal state x of the algorithm (e.g., in DES a possible x, in one of the possible attacks, is bit b of the intermediate L before round 16th [7]).

The partitioning in sets and the observation of the difference between sets is essential in separating minimal data-dependent power consumption from huge amounts of uncorrelated noise. Such possibility depends on the availability of the DPA selection function $D(\cdot)$, which is used to asses the internal state x as a function of the key guess and of known variables—for instance, $D(C, b, K_s)$ for DES, wherein C is the ciphertext, b is the examined bit of L, and K_s is the 6-bit key hypothesis entering one substitution box. We call $X = \{x_1, x_2, ...\}$ the set of all possible internal states for which a DPA selection function can be found. Also, we call Y the union set of all known variables for the above DPA selection functions (namely, C for the classic selection function of DES).

DPA-critical nodes have therefore the following necessary (but not sufficient) property:

PROPERTY 1. *DPA-critical nodes are* critical successors *of the nodes of G representing the key K and* critical successors *of the nodes of G representing the values X. We say that a node v is a* critical successor *of u if there exists at least a path between u and v in G which does not contain any node or edge representing a variable in Y.*

Nodes which are neither successors of K nor of X are independent from their values and therefore their contribution to the power trace is noise. Nodes which are successors of either K or X, but not of both, bring also a data-dependent contribution to power consumption but this is irrelevant for DPA because it cannot be separated from other power-consumption noise: for instance, in the DES case, nodes dependent only on K would bring equally distributed contributions to the two groups of power traces corresponding to bit b of L equal to 0 and 1 respectively. Finally, we can stop looking for successor nodes of K and X when we run into edges or output nodes representing known values.

The identification of all possible DPA selection functions (and thus of the corresponding X variables) is an algorithmic task carried out by cryptographic experts for the relevant cryptographic algorithm. But once the set X is known, we can use Property 1 to mark automatically all nodes potentially DPA-critical.

5.2 DPA-Resistant Instruction-Set Extensions

To choose the most efficient instruction-set extensions which make the processor DPA-resistant, the dataflow graph G is associated with an extended dataflow graph $G^+\left(V \cup V^+, E \cup E^+\right)$ which contains additional nodes V^+ and edges E^+. The additional nodes V^+ represent input and output variables of the basic block. The additional edges E^+ connect nodes V^+ to V, and nodes V to V^+.

A potential additional instruction S is an induced *subgraph* of G: $S \subseteq G$. Since each node can be either part of a subgraph or not, there are $2^{|V|}$ possible special instructions, where $|V|$ is the number of nodes in G. A subgraph S is *convex* if there exists no path from a node $u \in S$ to another node $v \in S$ which involves a node $w \notin S$. We call $\mathrm{IN}(S)$ the number of predecessor nodes of those edges which enter the subgraph S from the rest of the graph G^+ and $\mathrm{OUT}(S)$ is the number of predecessor nodes in S of edges exiting the subgraph. Finally, a function $\mathrm{M}(S)$ measures the merit of a possible solution S; it represents an estimation of the speedup achievable by implementing S as a special instruction.

The formulation of the identification problem is:

PROBLEM 1. *Given the dataflow graphs G and G^+, find a subgraph S of G which maximises $\mathrm{M}(S)$ under the following constraints:*

1. $\mathrm{IN}(S) \leq N_{\mathrm{in}}$,
2. $\mathrm{OUT}(S) \leq N_{\mathrm{out}}$,
3. *all DPA-critical nodes belong to S, and*
4. *S is convex.*

The user-defined values N_{in} and N_{out} indicate the register-file read and write ports, respectively, which can be used by the special instruction. The convexity constraint is needed to ensure that a schedule for the program can be found by the compiler once all nodes of the subgraph S are collapsed into a single node representing the new instruction.

```
identification() {
  for (i = 0; i < NODES; i++) cut[i] = 0;
  topological_sort();
  search(1, 0);
  /* new test follows */
  if (cutdpa[0] == 1) return;
  search(0, 0);
}
search(current_choice, current_index) {
  cut[current_index] = current_choice;
  if (current_choice == 1) {
    if (!output_port_check()) return;
    if (!convexity_check()) return;
    /* additional condition follows */
    if (input_port_check() &&
        all_cutdpa_in_cut()) {
      calculate_speedup();
      update_best_solution();
    }
  }
  if ((current_index + 1) == NODES) return;
  current_index = current_index + 1;
  search(1, current_index);
  /* new test follows */
  if (cutdpa[current_index] == 1) return;
  search(0, current_index);
}
```

Figure 5: The identification algorithm of [1] modified to force inclusion of DPA-critical nodes (represented by a one in the relevant position of array `cutdpa[]`). Changes are marked in the pseudocode.

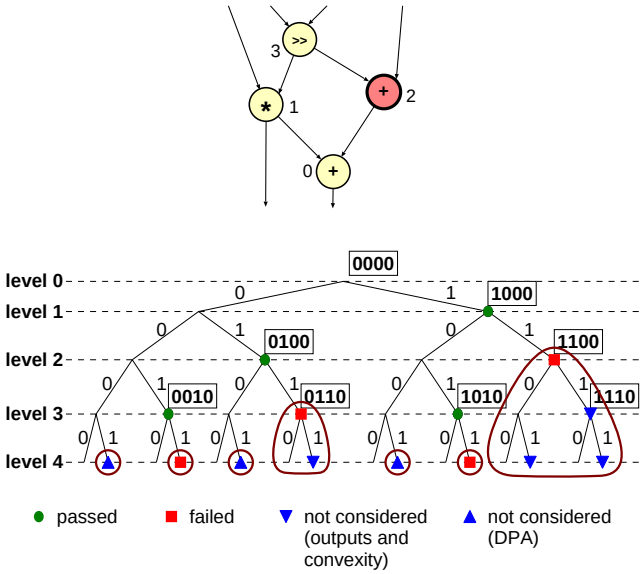

Figure 6: The execution trace of the algorithm (bottom) for the small sample graph shown at the top. Note that the node whose topological order is 2 is marked DPA-critical.

It can be easily solved with the algorithm of Figure 5, modified from the original algorithm of [1]. The basic idea of the original algorithm is to visit all possible solutions in a special order based on the graph topology, such that if constraints 2 or 4 of Problem 1 are violated, a set of potential solutions can be completely ignored without loss. The algorithm of Figure 5 contains two differences compared with the original: (1) in the exploration of the design space, whenever a node is DPA-critical, the algorithm is made ignore all further solutions not including that node, and (2) each potential solution is not only tested for a compatible number of inputs but also for the inclusion of all DPA-critical nodes. The former condition is not sufficient to ensure the latter because of the peculiar order used to visit the tree nodes.

Figure 6 shows the application of the algorithm to a sample graph where, for instance, one looks for $N_{\text{out}} = 1$ and considers node 2 as DPA-critical. The tree depicted represents the search space of the algorithm—i.e., all possible subgraphs (S's) of the initial sample graph. It is built from a root representing the empty subgraph, and each couple of 1- and 0-branches at level i represents the addition or not of the node having topological order i, to the subgraph represented by the parent node. Nodes of the search tree immediately following a 0-branch represent the same subgraph as their parent node, and can be ignored in the search. The search proceeds as a preorder traversal of the search tree, and at every level of the tree, a decision is made on whether or not to include in S a node of the graph—specifically, the node labelled with the level of the graph where the decision is taken.

Because constraints 2 and 4, when violated cannot be recovered by descending further in the tree, in some cases there is no need to branch towards lower levels; therefore the search space is pruned. Only 4 subgraphs pass both the output-port and convexity check, and include the DPA-critical node. Four subgraphs violate the either the output-port or the convexity check and therefore prevent the algorithm from descending further in the tree—four subgraphs are not even considered. Finally, of the 15 possible subgraphs (escluding the trivial empty cut), 3 are tested but rejected before checking their speedup because they do not include the node marked DPA-critical—an additional pruning compared to the original algorithm. It should be noted that the execution time of the algorithm cannot but improve compared to the original algorithm, since larger parts of the design space now will not be analysed. Also, one should observe that the algorithm still solves the problem optimally.

Finally, one should notice that the basic identification problem—that is, Problem 1 without constraint 3, always admits a solution, albeit possibly trivial (e.g., a single node). Problem 1, instead, might not have a solution at all for some given constraints N_{in} and N_{out}, and a required S_{DPA}. Several possible solutions could be imagined to address the problem: On one side one could try to cover S_{DPA} with several independent instruction-set extensions, each satisfying the constraints; this can be accommodated in a more complex version of the selection algorithm equally presented in [1]. Alternatively, one could repeatedly apply the algorithm of Figure 5 with relaxed input/output constraints, and accept the solution with minimal N_{in} and N_{out}; small microarchitectural changes could accommodate the relaxed constraints at the price of some performance loss. We consider such extensions beyond the present work and reserve it for future optimisations.

6. CONCLUSIONS

In this paper, we present a design methodology to implement instruction-set extensions for cryptographic processors and hardware accelerators with the aim to improve their robustness against DPA attacks. The approach has a high-level component that is based on identifying the critical units and functions in a design with respect to DPA resistance, and a gate-level component that is based on implementing the critical units using an inherently DPA-resistant circuit design style. We discuss the gate-level implementation of DPA-resistant function blocks using the MCML circuit design style. It was demonstrated that the proposed design style offers about two orders of magnitude improvement in terms of power-supply current fluctuations during switching events, while keeping the overall power dissipation at levels comparable to classical CMOS logic, and significantly reducing the input-pattern dependence of power traces. Additionally, a systematic identification algorithm is sketched to determine the critical nodes within a complex design, and the key properties of this identification approach have been discussed.

Our methodology can be used to automatically identify and to isolate the DPA-critical nodes, as well as to map them onto a robust cell library that is compatible with conventional deep-submicron CMOS technologies.

7. REFERENCES

[1] Kubilay Atasu, Laura Pozzi, and Paolo Ienne. Automatic application-specific instruction-set extensions under microarchitectural constraints. In *Proceedings of the 40th Design Automation Conference*, pages 256–61, Anaheim, Calif., June 2003.

[2] Paolo Faraboschi, Geoffrey Brown, Joseph A. Fisher, Giuseppe Desoli, and Fred Homewood. Lx: A technology platform for customizable VLIW embedded processing. In *Proceedings of the 27th Annual International Symposium on Computer Architecture*, pages 203–13, Vancouver, June 2000.

[3] Tom R. Halfhill. ARC Cores encourages "plug-ins". *Microprocessor Report*, 19 June 2000.

[4] Tom R. Halfhill. MIPS embraces configurable technology. *Microprocessor Report*, 3 March 2003.

[5] Tom R. Halfhill. Tensilica's software makes hardware. *Microprocessor Report*, 23 June 2003.

[6] Payam Heydari. Design and analysis of low-voltage current-mode logic buffers. In *Proceedings of 4th IEEE International Symposium on Quality Electronic Design*, pages 293–98, San Jose, Calif., March 2003.

[7] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael Wiener, editor, *Advances in Cryptology—CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, Berlin, August 1999.

[8] Simon Moore, Ross Anderson, Paul Cunningham, Robert Mullins, and George Taylor. Improving Smart Card security using self-timed circuits. In *Proceedings of the 8th International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 211–18, Manchester, April 2002.

[9] Simon Moore, Ross Anderson, Robert Mullins, George Taylor, and Jacques Fournier. Balanced self-checking asynchronous logic for Smart Card applications. *Microprocessors and Microsystems*, 27(9):421–30, October 2003.

[10] Jason Musicer and Jan Rabaey. MOS current mode logic for low-power, low-noise CORDIC computation in mixed-signal environments. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 102–7, Rapallo, Italy, July 2000.

[11] Adi Shamir. Protecting smart cards from passive power analysis with detached power supplies. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems—CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 71–77, Berlin, August 2000. Springer.

[12] Akira Tanabe, Masato Umetani, Ikuo Fujiwara, Takayuki Ogura, Kotaro Kataoka, Masao Okihara, Hiroshi Sakuraba, Tetsuo Endoh, and Fujio Masuoka. 0.18um CMOS 10-Gb/s multiplexer/demultiplexer ics using current mode logic with tolerance to threshold fluctuation. *IEEE Journal of Solid-State Circuits*, 36(6):988–96, June 2001.

[13] Kris Tiri, Moonmoon Akmal, and Ingrid Verbauwhede. A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on Smart Cards. In *Proceedings of the 28th European Solid-State Circuits Conference*, pages 403–6, Florence, September 2002.

[14] Kris Tiri and Ingrid Verbauwhede. Securing encryption algorithms against DPA at the logic level: Next generation smart card technology. In Colin D. Walter, Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems—CHES 2003*, volume 2779 of *Lecture Notes in Computer Science*, pages 125–136. Springer, Berlin, September 2003.

[15] Masakazu Yamashina and Hachiro Yamada. An MOS current mode logic (MCML) circuit for low-power sub-GHz processors. *IEICE Transactions on Electronics*, E75-C(10):1181–87, October 1992.