A Dynamic and Differential CMOS Logic Style to Resist Power and Timing Attacks on Security IC's.

Kris Tiri and Ingrid Verbauwhede

Contact Address:

Kris Tiri

UCLA Electrical Engineering Department

7440B Boelter Hall

P.O. Box 951594

Los Angeles, CA 90095-1594

Tel: 1-310-267-4940, Fax: 1-310-794-1592

E-Mail: tiri@ee.ucla.edu

Abstract

We present a dynamic and differential CMOS logic style, which has a signal independent switching behavior. It is shown that during each clock cycle, power consumption and all circuit characteristics, such as leakage current, instantaneous current and input-output delay are identical and independent of the logic value and the sequence of the input data. Implementing the encryption module in this logic will protect it against any Side Channel Attack that takes advantage of power, timing and leakage information. We have built a set of logic gates and a flip-flop needed for cryptographic functions and implemented a larger module, for which area, total power consumption and variation on the power consumption have been compared with implementations in Static Complementary CMOS logic, genuine Dynamic and Differential Logic and Current Mode Logic.

I. INTRODUCTION

Electronic banking, e-commerce, virtual private networks and so on cannot operate without encryption technology and a secure implementation of the encryption technology. To obtain security, many strong encryption algorithms have been developed. While usually strong against mathematical attacks, Side Channel Attacks (SCA's) can reveal the secret key through information leaked by the actual implementation of the encryption module [1]. Various attacks based on power consumption and timing information have been successfully mounted on hard- and software implementations of a wide range of encryption algorithms.

Differential Power Analysis (DPA) is the most powerful SCA. The attack is based on the fact that CMOS logic and application specific details cause logic operations to have power characteristics that depend on the input data. It relies further on statistical analysis and error correction to extract the information from the power consumption that is correlated to the secret key [2]. Numerous techniques have been proposed to prevent DPA. On the algorithmic level, random process interrupts interleave dummy instructions to avoid sequential execution of the algorithm [2],[3]. Integration techniques however, are able to resynchronize the power traces [4]. Masking is a technique that prevents intermediate variables to depend on an easily accessible subset of the secret key [5]. But DPA has been modified to handle masking [6],[7]. On the architectural level, techniques include adding random power consuming operations [2],[3] and duplicating logic with complementary operations. These procedures merely lower the side-channel information [2],[5] and might easily be disabled through tampering. Active power signal filtering with power consumption compensation, passive filtering, battery on chip and detachable power supply influence the power transfer itself [8]. The first method is likely to lag behind the fast power fluctuations [8],[9] and physical dimensions limit the latter three [8].

The foregoing techniques attempt to conceal the supply current variations at the architectural or the algorithmic level, while they originate at the logic level. A simple means to halt DPA is to have the encryption module or at least the sensitive parts of it implemented in a logic, whose power consumption is independent of the signal transitions. This can be achieved by two means. A first approach is to draw a fixed current from the supply as in Current Mode Logic (CML) [10]. Static power dissipation is the major drawback of CML. A better alternative is to charge in every cycle a capacitance with a constant value as we first proposed with Sense Amplifier Based Logic (SABL) [11]. Differential Logic, often put forward as the solution when first introduced to DPA, does not have a signal independent power consumption, nor do Dynamic Logic and regular Dynamic and Differential Logic (DDL). In this work, we will elaborate SABL and judge it against CML and DDL.

Another research effort also situated at the logic level consists of implementing asynchronous logic [12]. Asynchronous logic achieves its resistance against DPA through (1) a 1-of-n coding scheme, which produces a DDL-behavior and thus guarantees one wire transition for every symbol; and (2) a randomized execution process, which eliminates a time reference and thus obstructs the recovery of the remaining power differences immanent in genuine DDL. Its major disadvantage is a complex design flow.

Section II develops the criteria for a voltage mode logic style to have input signal independent power consumption. Based on the observations, we elaborate our SABL in section III. A special Differential Pull Down Network (DPDN), logic gates, a flip-flop and design rules for combinatorial and sequential logic are presented. Next in section IV, an experiment has been set up to evaluate SABL and 3 other logic styles, namely CML, regular DDL and Static Complementary CMOS Logic. Based on simulation results, we will compare the area, the total power consumption and the variation on the power consumption. Section V will show that SABL can be used to bar a wide class of other Side Channel Attacks, such as Timing Attacks, attacks based on subthreshold currents, Differential Fault Analysis and Electromagnetic Analysis. Finally, a conclusion will be formulated.

II. REQUIREMENTS FOR TRANSISITON INDEPENDENT POWER CONSUMPTION

Static Complementary CMOS Logic (SC-CMOS), which is the default logic style in standard cell libraries used for security IC's, primarily consumes its energy from the power supply when the output sees a ‘0’ to ‘1’ transition. During the ‘1’ to ‘0’ transition, the energy previously stored in the output capacitance is dissipated, but aside from a direct-path current no energy is consumed from the power supply. The ‘0’ to ‘0’ and the ‘1’ to ‘1’ transitions are degenerated events, in which a charge transfer does not occur. This asymmetric power demand provides the information used in DPA to find the secret key.

A logic style with transition-independent power consumption does not reveal this information. When logic values are measured by charging and discharging capacitances, we need to use a fixed amount of energy for every transition. In other words, even though different capacitances are switched, we need a logic style with the unique property of charging in every cycle a *“total”* capacitance with a constant value. This can be assured by fulfilling two requirements: (1) the logic style has one and only one switching event per cycle and this independently of the input signals; and (2) the logic style charges a constant capacitance during that switching event.

Implementing a Dynamic and Differential Logic style meets the first requirement: DDL has a switching factor of 100% and does not suffer from glitching. A Dynamic Logic style alternates precharge and evaluation phases, in which the output is precharged to ‘1’ and conditionally evaluated to ‘0’ respectively. A Differential Logic style holds two output signals with opposite polarity. As a result, the combination of Dynamic and Differential Logic will evaluate exactly one of both precharged output nodes to ‘0’ in order to generate a complementary output and this independently of the input value. During the subsequent precharge phase, the discharged node is charged and this independently of the input sequence.

Note that Differential Logic by itself does not have a switching factor of 100%. This is the main reason that addressing the DPA solely by balancing the Hamming weights cannot succeed [2],[5]. Whether it is done on algorithmic level (e.g. exclusively handling bytes with Hamming weight 4), architectural level (e.g. duplicating the module with a complementary module) or logic level (e.g. Differential Logic), the degenerated ‘0’ to ‘0’/’1’ to ‘1’ output-transitions do not consume power.

To fulfill the second requirement, the load at the two differential output nodes should be balanced. Both capacitances are identical if all its components, which are the intrinsic output capacitances of the gate, the interconnect capacitances and the input capacitances of the following gates, can be balanced through a careful layout.

(Dis)charging parasitic capacitances, which are internal to the gate, have an influence on the power consumption as well. This is the reason that genuine DDL [13],[14] cannot be used. Simulations indicate that e.g. for the AND-NAND gate in Cascode Voltage Switch Logic (CVSL) [15], which is depicted in Figure 1(left), the variation on the power consumption varies between 10 and 50% depending on the implementation. The variation originates from asymmetry in the gate. Depending on the input, different parasitic capacitances discharge during the evaluation phase, as is shown in Figure 1(right). In none of the 4 subsequent precharge events, the same combination of capacitances is charged. It is virtually impossible to match the discharge events by manipulating transistor sizes and/or layouts. The contribution of the internal nodes however, should and can be controlled exactly by making sure that all internal nodes eventually discharge and this independently of the inputs, as will be explained in the next section.

III. SENSE AMPLIFIER BASED LOGIC

Conforming to the observations made in section II, we design a gate that (1) is dynamic and differential; (2) is symmetric at both the input and the output nodes; and (3) discharges all the internal nodes of the Differential Pull Down Network.

A. *SABL gate*

The SABL gate is based on the StrongArm110 flip-flop (SAFF) [16]. To realize a gate, we select the sense-amplifier part (SA) of the SAFF, drop the static NAND gates of the Set-Reset latch, and incorporate a logic function by exchanging the input differential pair for a Differential Pull Down Network. Figure 2(left) depicts the SABL gate. The DPDN, of which the design and the special characteristics are discussed in subsection 1, will connect for a complementary input one of the external nodes X and Y to the common node Z.

The behavior of a SABL gate matches the behavior of the SA, which has been detailed in literature [17]. During the precharge phase (clk-signal low), node Z is disconnected from GND and the output nodes and nodes X and Y are precharged to VDD and VDD–V_t respectively, where V_t is the threshold voltage. During the evaluation phase (clk-signal high), the cross-coupled inverter will toggle and provide a stable output as soon as a branch of the DPDN provides a path to GND. In case node X is connected to node Z, node out will become '0', while node $\overline{out}$ remains at '1', and vice versa in case node Y is connected to node Z, node $\overline{out}$ will become '0'.

Transistor M_1, which is always on, has been integrated into the SA to guarantee static operation during the evaluation phase. Even if the input to the differential pair would change after the SA has toggled, the '0' output node is always connected to GND. The inputs to the SABL gate however, do not change after their initial transition, as they come from stable outputs of previous SABL gates. In our design, M_1 serves to discharge both external nodes X and Y of the DPDN. Whichever branch of the DPDN is on, both nodes are connected through M_1 and will eventually be discharged together with one of the output nodes. During the precharge phase, all the discharged nodes and capacitances are charged.

Note that once the SABL gate has toggled, it benefits from static operation. All nodes are connected either to VDD or to GND and there are no dynamic nodes that are noise sensitive and suffer from charge leakage.

1. Special Differential Pull Down Network (DPDN)

We create a special DPDN in order to control the contribution of the internal parasitic capacitances into the power consumption. In contrast with genuine DDL, where the design constraints are to minimize the device count and the number of stacked levels [18], the DPDN is designed such that for a differential input all internal nodes are connected to one of the external nodes. As a result, since both external nodes X and Y eventually discharge, all the internal nodes are discharged and will be charged together with one of the balanced output loads during precharge. That means that in every cycle SABL charges the same capacitance value: one of the balanced load capacitances and the sum of all the internal node capacitances. Hence SABL has signal independent power consumption.

As an example, Figure 3 shows the transformation of the DPDN that implements the AND-NAND function of Figure 1 to a special DPDN used in SABL. The original network has one internal node, node W, which possibly becomes floating depending on the input combination. Repositioning transistor M_2, which is located between nodes Y and Z and is driven by input $\overline{A}$, between nodes Y and W does not alter the functionality of the DPDN, $\overline{A}+\overline{B}=\overline{A}.B+\overline{B}$, but guarantees that for any differential input node W is connected to one of the output nodes.

Figure 4(left) depicts the resulting SABL AND-NAND gate. Figure 4(right) shows the transient HSPICE simulation of (dis)charging events in this gate for 2 different inputs. The figure demonstrates that for both events all the internal node capacitances and one of the balanced output nodes are (dis)charged. In each event, the same amount of charge, and hence the same amount of energy, is used. Furthermore, the same amount of charge is going through very similar discharge paths, which results in a constant delay and instantaneous current.

Design procedure

The design goal is to guarantee that all internal nodes are connected to one of the external nodes for a differential input. The design procedure is a transformation that repositions transistors in the DPDN. As a result, the total number of devices remains the same. The total evaluation depth may increase.

The design procedure of creating a special DPDN for a logical function f consists of 5 steps.

1. <u>Identify</u> 2 expressions x and y that combine to the logical function f. This results in an AND-operation, $x.y$, or an OR-operation, $x + y$.
2. <u>Complement</u> the expression in x and y to get the dual expression of $\overline{f}$. This results in an OR-operation, $\overline{x} + \overline{y}$, or an AND-operation, $\overline{x}.\overline{y}$ respectively.
3. <u>Transform</u> the OR-operation.

   The results of step 1. and step 2. are two dual expressions:

   either case A) $\begin{cases} f = x.y \\ \overline{f} = \overline{x} + \overline{y} \end{cases}$ or case B) $\begin{cases} f = x + y \\ \overline{f} = \overline{x}.\overline{y} \end{cases}$

   One expression is an AND-operation, the other an OR-operation. In the DPDN, the AND-operation is implemented as a series combination, the OR-operation as a parallel combination. At this abstraction level, only the series combination has an internal node.

   In case A), we transform the parallel connection into $\overline{x}.y + \overline{y}$, put network y at the bottom of the $x.y$ connection and share network y between the two branches $x.y$ and $\overline{x}.y + \overline{y}$.

   In case B) we transform the parallel connection into $x.\overline{y} + y$, put network $\overline{y}$ at the bottom of the $\overline{x}.\overline{y}$ connection and share network $\overline{y}$ between the two branches $\overline{x}.\overline{y}$ and $x.\overline{y} + y$.

   Now the DPDN connects the internal node of the series connection to the output node.
4. <u>Repeat</u> the procedure for the logical expressions x and y till the network consists of only 1 transistor.

5. Substitute the results.

For a given schematic of a DPDN, the design procedure translates to (1) identify all networks in series; (2a) open the corresponding parallel connections at the bottom of the network and (2b) connect them to the internal nodes of the series connections; and (3) unroll the network.

Figure 5 illustrates the design procedure applied to a complex DPDN. In the final DPDN, both the true and the inverse of a signal control a transistor for every internal node: independent of the input, every internal node is connected to another node, which is either an external node or another internal node, for which both the true and the inverse of a signal control a transistor. As a result independent of the input, every internal node is connected to an external node.

The transistors in the DPDN can only charge the internal nodes as long as they are on. To assure that every cycle the same amount of charge is consumed, this charge time must be constant. That is the case if at every node both the signal and the inverse of the signal control a transistor that loads the node. Whether it is the transistor controlled by the signal or the one controlled by the inverse, the total charge time for the internal nodes is the time needed to precharge the outputs of the preceding gate. This requirement is fulfilled by the special DPDN.

Enhanced special DPDN

The idea behind the enhanced special DPDN is to make the evaluation depth of the DPDN equal for all discharge events. For this purpose, we use a pass-gate, which is a connection between two nodes that is always open for a differential input combination and which is built by a parallel combination of 2 transistors driven by both the signal and its complement. The pass-gates are inserted if different discharge paths do not have the same number of transistors, as e.g. is the case with the SABL AND-NAND gate. We insert a pass-gate controlled by signals that do not yet control a transistor in that particular discharge path. Figure 6 shows the SABL AND-NAND gate with enhanced special DPDN.

The introduction of the dummy transistors has two effects. First, there is a constant resistance in the discharge path. As a result, each gate has a constant delay as now both C and R are independent of the inputs. Second, no evaluation will start before all inputs are stable. As a result, the delay of an entire combinatorial logic tree will be constant, as each gate evaluates when the input-pair with the longest delay has switched.

2. p-type gate

All SABL gates presented above are n-type gates. The dual gate is a SABL p-type gate, which is shown in Figure 2(right). This gate predischarges to GND when the clk-signal is high and evaluates one node to VDD through the differential pull up network (DPUN) when the clk-signal is low.

B. Building combinatorial and sequential logic

1. Combinatorial logic: cascading of dynamic gates

For correct operation, all transistors of the conditional path should be off at the onset of the evaluation phase. SABL, like every Dynamic Logic style, is therefore connected using either Domino Logic, in which each gate is extended with a pair of static inverters, or np-Logic, in which n-type and p-type gates are alternated. The utilization of static gates in Domino Logic does not harm our objective of transition independent power consumption. The pair of inverters fulfills the requirements presented in section II as exactly one inverter will have a switching event; no internal nodes exist; and both inverters are balanced at the input nodes and at the output nodes.

2. Sequential logic: charge storage during precharge phase

The SAFF uses a static Set-Reset latch to hold the output value during the precharge phase of the SA. If the flip-flop stores the same value for 2 or more consecutive cycles, the latch will not switch, but keeps its state. Consequently, there is no power consumption in the static latch and the original SAFF is vulnerable to DPA. We present a master-slave flip-flop that operates without a static latch.

A combination of a p-SA and an n-SA, as shown in Figure 7, acts as a master-slave flip-flop. The p-SA evaluates at the falling edge of the clk-signal and keeps this value till the rising edge, while the n-SA evaluates at the rising edge of the clk-signal and keeps this value till the falling edge. As a result, the value is stored during one entire clock cycle. This is a negative edge triggered flip-flop that stores the input-signal at the instant that the surrounding combinatorial logic will precharge. It releases the output-signal at the instant that the surrounding combinatorial logic will evaluate. For correct operation, the actual precharge time has to be large enough to evaluate the correct differential input. If necessary, the hold time can be fixed by adding a delay Δt has at the input of both the p-SA and the n-SA. This can be done with an extra load capacitance or with static inverters.

For the logic gate, which directly precedes the flip-flop, the rules presented above in subsection IIII.B.1 (Combinatorial logic: cascading of dynamic gates) do not hold. Both an n-type and a p-type gate can be directly connected to the flip-flop: the logic gate has a differential output at the negative clock edge. For the logic gates, which directly follow the flip-flop, the rules hold. The flip-flop needs to be followed by p-type gates or by a pair of inverters followed by n-type gates: the second stage of the flip-flop and the gates will both evaluate at the rising clock edge.

IV. EXPERIMENTAL RESULTS

Encryption algorithms use arithmetic that differs from the two's complement arithmetic on integers and real numbers. Instead, they use Galois field arithmetic and operations such as substitutions and permutations [19]. These operations can be implemented with the following set of cells: inverter, NAND, XOR and flip-flop. We have built this set in 4 different logic styles, which are Sense Amplifier Based Logic, Static Complementary CMOS Logic, Current Mode Logic, and Dynamic and Differential Logic, and used it to implement a substitution box. The substitution box is the S9-box found in the Kasumi algorithm, which is the encryption algorithm in 3G cellular standard [20]. The module consists of 5 inverters, 86 NAND's and 92 XOR's and substitutes 9 input bits with 9 other bits.

Several types of CML, which can be implemented in CMOS technology, exist. We use Current Steering Logic (CSL) [21]. A CSL gate consists of a current source connected between VDD and a parallel combination of a PDN and a diode-connected n-mos transistor. Depending on the input, the current is guided to GND along either the PDN (output voltage low) or the diode-connected device (output voltage high). To restrict channel-length modulation, which is the influence of the drain to source voltage (V_{ds}) on the current of a transistor with constant gate to source voltage (V_{gs}) [10], we have implemented the current source using two cascoded p-mos devices. Other than the remaining channel length modulation, residual current spikes exist due to the displacement current charging the parasitic capacitances between VDD and the output [21].

For the Dynamic and Differential Logic, we use CVSL, which has been presented in section II.

Circuits are designed in a 0.18-µm, 1.8V CMOS technology. The implementations in SABL and CVSL are built according to the np-Logic design rule. The SABL implementation did not use the enhanced DPDN. Layout is created with LayoutPlus. Layout to netlist is done with Analog Artist. Simulations on the extracted circuits are done in HSPICE.

We have simulated a random input sequence of length 500 clock cycles at a clock frequency of 66MHz. and recorded the total energy that has been consumed in every cycle. From the resulting 500 samples, the Normalized Deviation (ND) and the Normalized Standard Deviation (NSD) have been calculated. ND, which ranges from 0 to 1, is defined as the difference between the maximum and the minimum normalized with the maximum. NSD is defined as the standard deviation normalized with the mean. The closer ND and NSD are to 0, the more measurements are necessary and the more accurate the measuring devices have to be in order to extract the side-channel information.

Figure 8 shows the histogram of the observed energy consumption per cycle. The SC-CMOS implementation experiences a large variation on the energy consumption, whereas the deviations on the energy consumption of the 3 other implementations remain in a narrow band. SABL has a noticeable smaller variation than CVSL, but it takes more energy to achieve this effect. CSL has the smallest variation, yet it has the highest energy consumption. Especially if a module has many levels of logic, CSL has a high power penalty. The current of a CSL gate is set to charge the output capacitance within a certain time interval. During the rest of the period, this current is wasted.

Table I summarizes the important characteristics. The numbers confirm the observations. SABL achieves an NSD a factor of 43 smaller than the NSD of SC-CMOS. The reduction comes with a tradeoff in an increase of a factor 2.2 in energy consumption and a factor 1.8 in area. Using SABL instead of CVSL offers an additional reduction of a factor 2.5 to 3 in the variation of the energy consumption. SABL however, uses more energy and area. CSL has the smallest variation, but the highest area and the highest energy consumption of all logic styles.

Figure 9 shows a superposition of the power supply current of the transient response. It shows that the instantaneous current of the SC-CMOS implementation is highly irregular. The SABL and CVSL implementations seem to have the same behavior. Though, Table I and Figure 8 have indicated that there is an important difference. CSL shows no visible variation.

V. SABL AND OTHER SCA'S

A. Timing Attacks

Timing Attacks (TA's) are a class of cryptanalysis that uses timing information leaked by the encryption module [22],[23]. A careful design, which at all times has a worst case running time, bars the regular TA. Power measurements however, still provide substantial timing information. They expose idle cycles, which have been inserted to hide conditional branches with unequal lengths, but also the actual delay in a clock cycle. As shown in Figure 9, an important variation exists on the time span that the SC-CMOS implementation draws current, or in other words on the time span that switching events take place.

For SABL however, this is not the case. When desired, idle cycles can be inserted. Every gate has a switching event in every cycle, whether or not useful data is processed. Furthermore, we have seen in Figure 4 and in Figure 9 that the delay and the instantaneous current of a gate and the complete module are constant and independent of the input vector.

B. Attacks based on subthreshold currents

Concerns exist about the information leaked by subthreshold currents [9]. The subthreshold current, which is also called leakage current, is the current that flows through a transistor with V_{ds} larger then 0 when V_{gs} is smaller then V_t [10]. The actual value depends on the terminal voltages, the width and the type of the transistor. This means that the total subthreshold current of a gate depends on which particular transistors are off, and thus on the state of the gate.

A SABL gate has basically two states: a precharged state and an evaluated state. During precharge, no data values are stored and therefore no information can be leaked. In the evaluated state, all nodes of the DPDN are at GND. As a result, these transistors have a V_{ds} of 0 and do not carry a subthreshold current. The only part, which suffers from subthreshold current, consists of the cross-coupled inverter pair and the two clocked pmos transistors. This circuit has only two states, which since the circuit is symmetric are equivalent. As a result the leakage current is independent of the state the circuit is in.

C. Differential Fault Analysis

In Differential Fault Analysis (DFA), the attacker tries to force an error in the internal state of the circuit, and subsequently exploit weaknesses of the algorithm under malfunctioning [1]. The main countermeasure against DFA is fault detection [1], where the idea is to monitor if one has tried to tamper the device, and in such an event to shut down the processor and delete any valuable information as soon as possible. While most DFA's seem purely theoretically, Glitch Attacks (GA's) are viewed as a practical DFA [1]. In a GA, the clock frequency is temporarily increased with the intention to force a state-bit or a selected conditional branch by not giving enough time for all calculations to complete.

SABL can detect GA's. The enhanced DPDN only evaluates if all incoming signals are stable. If at the falling edge, some input signals are not differential, which can be detected with a ready signal, the clock frequency has been increased and the circuit should be put in the alarm state. Since the delay of the encryption module is independent of the input vector, the critical path has always the longest delay: the detection only needs to be performed at the receiving flip-flop of the critical path. Good design practice is to create on purpose a critical path inside a DFA detection module.

D. Electromagnetic Analysis

The flow of electric charges produces an electromagnetic field. Electromagnetic Analysis (EMA) is the equivalent of a power attack but instead uses the electromagnetic fields generated by the (dis)charging gates as the side channel information [24]. There are differences. A power attack has only access to the global power consumption. An EMA however, can do measurements that are confined to a small area of the security IC. On the other hand, a power attack can be mounted rather quickly with off-the-shelf devices, whereas an EMA requires special probes and an advanced measurement setup.

SABL withstands EMA. The instantaneous current of a SABL gate is independent of the switching events. As a result, the electromagnetic fields, which are generated by the electric charge flowing through the VDD and the GND lines, are independent of the switching events too. In addition, the charge flows also through either of the 2 differential output wires. These interconnects are routed in the same environment and over the same distance. Therefore, the electromagnetic fields of these wires are also independent of the output transitions.

VI. CONCLUSION

We have presented a voltage mode logic style that has a power consumption that is practically independent of the input vector. As a result, SABL is a technique that removes the foundation of Differential Power Analysis. Compared to SC-CMOS logic, which is the default logic style for standard cells used in security IC's, the logic style has a reduction of more than a factor 40 in the variation of the energy consumption of a large module. The reduction comes with an increase in area and in power consumption of a factor 2. Compared to genuine DDL, SABL offers a reduction of a factor 3.

An important advantage of the logic style is that it is a distributed solution. Tampering the device in order to cancel the countermeasure is impossible as opposed to techniques proposed at e.g. the architectural level. Furthermore, we have evaluated the logic style and concluded that it will protect the encryption module against a wide range of Side Channel Attacks.

REFERENCES

[1] E. Hess, N. Janssen, B. Meyer, and T. Schütze, "Information Leakage Attacks Against Smart Card Implementations of Cryptographic Algorithms and Countermeasures - A Survey," in *Proc. of EUROSMART Security Conference*, 2000, pp. 55–64.

[2] P. Kocher, J. Jaffe and B. Jun, "Differential Power Analysis," in *Proc. of Advances in Cryptology (CRYPTO'99), Lecture Notes in Computer Science*, vol. 1666, 1999, pp. 388-397.

[3] J. Daemen and V. Rijmen, "Resistance Against Implementation Attacks: A Comparative Study of the AES Proposals," in *Proc. of the Second Advanced Encryption Standard (AES) Candidate Conf.*, http://csrc.nist.gov/encryption/aes/round1/conf2/aes2conf.htm, March 1999.

[4] C. Clavier, J.-S. Coron and N. Dabbous, "Differential Power Analysis in the Presence of Hardware Countermeasures," in *Proc. of Cryptographic Hardware and Embedded Systems (CHES 2000), Lecture Notes in Computer Science*, vol. 1965, 2000, pp. 252-263.

[5] S. Chari, C. S. Jutla, J. R. Rao and P. Rohatgi, "Towards Sound Approaches to Counteract Power-Analysis Attacks," in *Proc. of Advances in Cryptology (CRYPTO'99), Lecture Notes in Computer Science*, vol. 1666, 1999, pp. 398-412.

[6] J.-S. Coron and L. Goubin, "On Boolean and Arithmetic Masking against Differential Power Analysis," in *Proc. of Cryptographic Hardware and Embedded Systems (CHES 2000), Lecture Notes in Computer Science*, vol. 1965, 2000, pp. 231-237.

[7] T.S. Messerges, "Using Second-Order Power Analysis to Attack DPA Resistant Software," in *Proc. of Cryptographic Hardware and Embedded Systems (CHES 2000), Lecture Notes in Computer Science*, vol. 1965, 2000, pp. 238-251.

[8] A. Shamir, "Protecting Smart Cards from Passive Power Analysis with Detached Power Supplies," in *Proc. of Cryptographic Hardware and Embedded Systems (CHES 2000), Lecture Notes in Computer Science*, vol. 1965, 2000, pp. 71-77.

[9] J.-S. Coron, P. Kocher, D. Naccache, "Statistics and Secret Leakage," in *Proc. of Financial Cryptography'00, Lecture Notes in Computer Science*, 2000.

[10] J. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits: Second Edition*, Prentice Hall, 2003.

[11] K. Tiri, M. Akmal and I. Verbauwhede, "A Dynamic and Differential CMOS Logic with Signal Independent Power Consumption to Withstand Differential Power Analysis on Smart Cards", in *Proc. Of 28th European Solid-State Circuits Conference*, September 2002, pp. 403-406

[12] S. Moore, R. Anderson, P. Cunningham, R. Mullins and G. Taylor, "Improving Smart Card Security using Self-timed Circuits", *Eighth International Symposium on Advanced Research in Asynchronous Circuits and Systems*, 2002.

[13] P. Ng, P. T. Balsara, and D. Steisss, "Performance of Differential Circuits," *IEEE J. Solid-State Circuits*, vol. 31, No. 6, pp. 841-846, June 1996.

[14] K. Bernstein, K. M. Carrig, C. Durham, P. R. Hansen, D. Hogenmiller, E. J. Nowak and N. J. Rohrer, *High Speed CMOS Design Styles*. Kluwer Academic Publishers, pp. 111-123, 1998.

[15] L. G. Heller, W. R. Griffin, J.W. Davis and N. G. Thoma, "Cascode Voltage Switch Logic: A differential CMOS Logic Family," *Digest of Technical Papers, IEEE International Solid-State Circuits Con*ference, pp. 16-17, Feb. 1984.

[16] J. Montanaro, et al., "A 160-MHz, 32-b, 0.5-W CMOS RISC Microprocessor," *IEEE J. Solid-State Circuits*, vol. 31, No. 11, pp. 1703-1712, November 1996

[17] B. Nikolic, V. G. Oklobzija, V. Stojanovic, W. Jia, J.K. Chiu and M.M. Leung, "Improved Sense-Amplifier-Based Flipflop: Design and Measurements," *IEEE J. Solid-State Circuits*, vol. 35, pp. 876-883, June 2000.

[18] K. M. Chu and D. I. Pulfrey, “Design Procedures for Differential Cascode Voltage Switch Circuits,” IEEE *J. Solid-State Circuits*, vol. 21, pp. 1082-1087, December 1986.

[19] A. J. Menezes, P. C. Van Oorschot, S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.

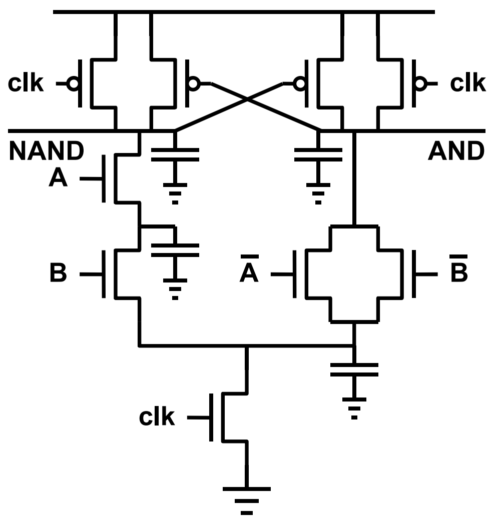
[20] “Universal Mobile Telecommunications System (UMTS); Specification of the 3GPP confidentiality and integrity algorithms; Document 2: Kasumi algorithm specification (3GPP TS 35.202 version 5.0.0 Release 5),” http://pda.etsi.org/pda/, June 2002.

[21] H. Ng, D. Allstot, “CMOS Current Steering Logic for Low-Voltage Mixed-Signal Integrated Circuits,” *IEEE Transactions on VLSI Systems*, vol. 5, no. 3, Sept. 1997.

[22] P. Kocher, “Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems,” in *Proc. of Advances in Cryptology (CRYPTO’96), Lecture Notes in Computer Science,* vol. 1109, 1996, pp. 104-113.

[23] J.F. Dhem, F. Koeune, P.A. Leroux, P. Mestre, J.J. Quisquater and J.L. Willems, “A Practical Implementation of the Timing Attack,” in Proc. of CARDIS, Sept. 1998.

[24] Jean-Jacques Quisquater and David Samyde. ElectroMagnetic Analysis (EMA): Measures and Couter-Measures for Smard Cards. In *Smart Card Programming and Security (E-smart 2001)*, Cannes, France, LNCS 2140, pp.200-210. Sept. 2001.



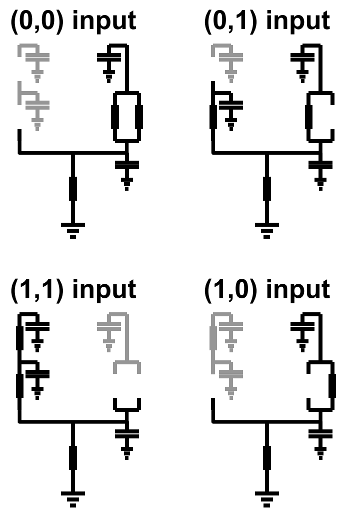
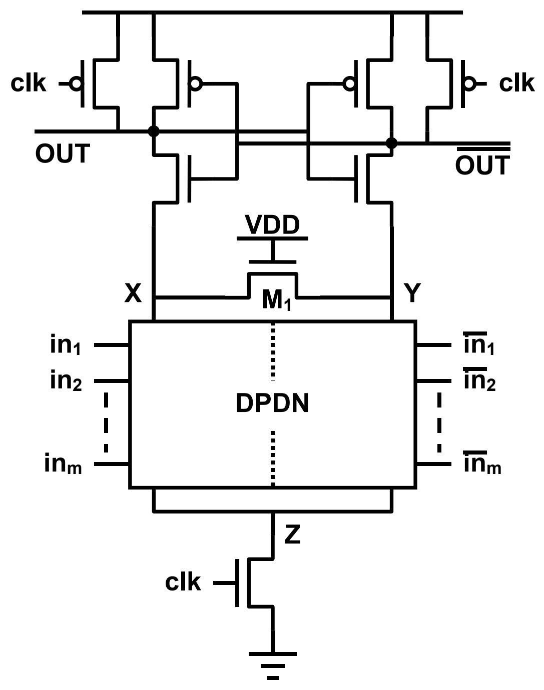


Figure 1: CVSL: AND-NAND gate with parasitic capacitances (left); and discharge events (right).



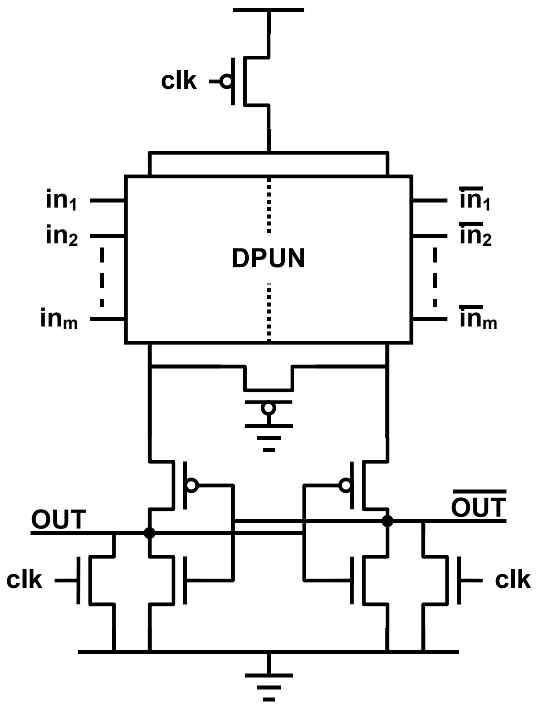


Figure 2: SABL: n-type gate (left); and p-type gate (right).

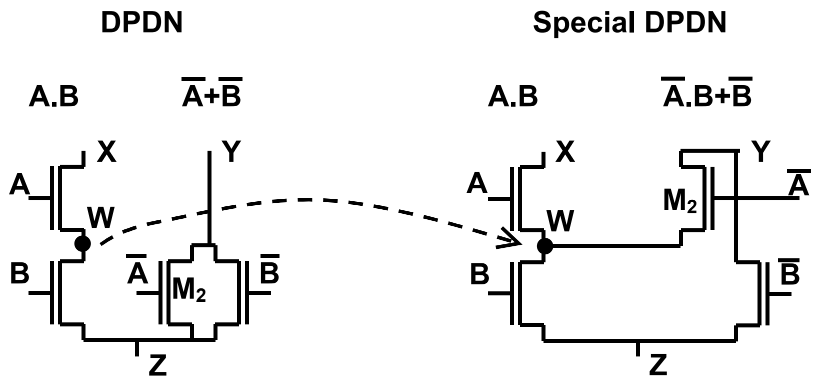
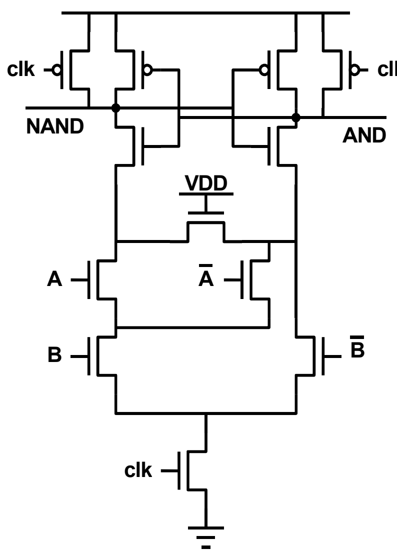


Figure 3: Transformation of DPDN to special DPDN used in SABL for AND-NAND gate.



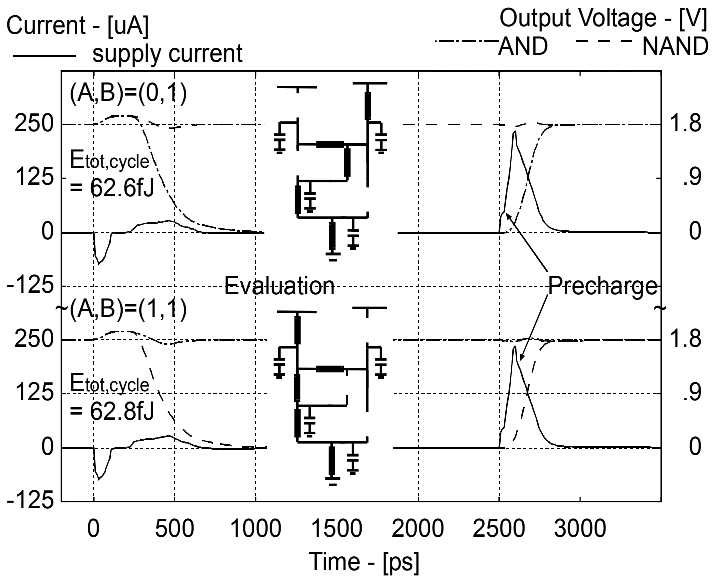


Figure 4: SABL AND-NAND gate (left); and transient simulation of (dis)charging events in AND-NAND gate (right): (0,1)-input (top); and (1,1)-input (bottom).

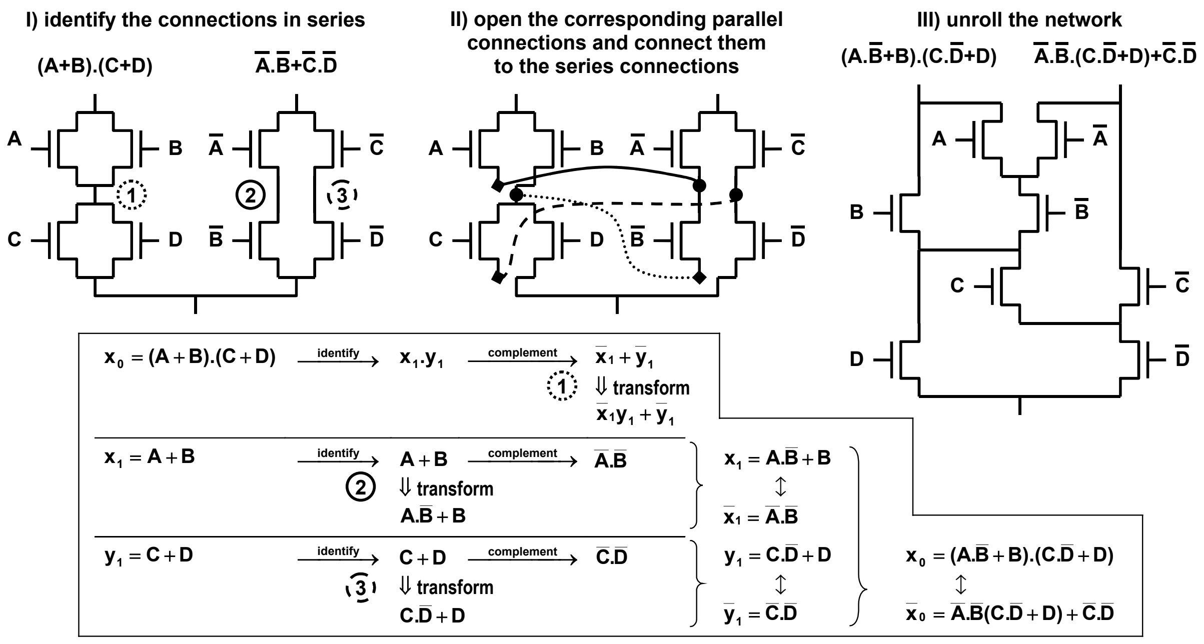

I) identify the connections in series
(A+B).(C+D)
$\overline{A}.\overline{B}+\overline{C}.\overline{D}$
II) open the corresponding parallel connections and connect them to the series connections
III) unroll the network
$(A.\overline{B}+B).(C.\overline{D}+D)$
$\overline{A}.\overline{B}.(C.\overline{D}+D)+\overline{C}.\overline{D}$
$x_0 = (A+B).(C+D) \xrightarrow{identify} x_1.y_1 \xrightarrow{complement} \overline{x}_1+\overline{y}_1$
⇓ transform
$x_1y_1+\overline{y}_1$
$x_1 = A+B \xrightarrow{identify} A+B \xrightarrow{complement} \overline{A}.\overline{B}$
⇓ transform
$A.\overline{B}+B$
$x_1 = A.\overline{B}+B$
$\overline{x}_1 = \overline{A}.\overline{B}$
$y_1 = C+D \xrightarrow{identify} C+D \xrightarrow{complement} \overline{C}.\overline{D}$
⇓ transform
$C.\overline{D}+D$
$y_1 = C.\overline{D}+D$
$\overline{y}_1 = \overline{C}.\overline{D}$
$x_0 = (A.\overline{B}+B).(C.\overline{D}+D)$
$\overline{x}_0 = \overline{A}.\overline{B}(C.\overline{D}+D)+\overline{C}.\overline{D}$

Figure 5: Transformation of DPDN to special DPDN: design example.

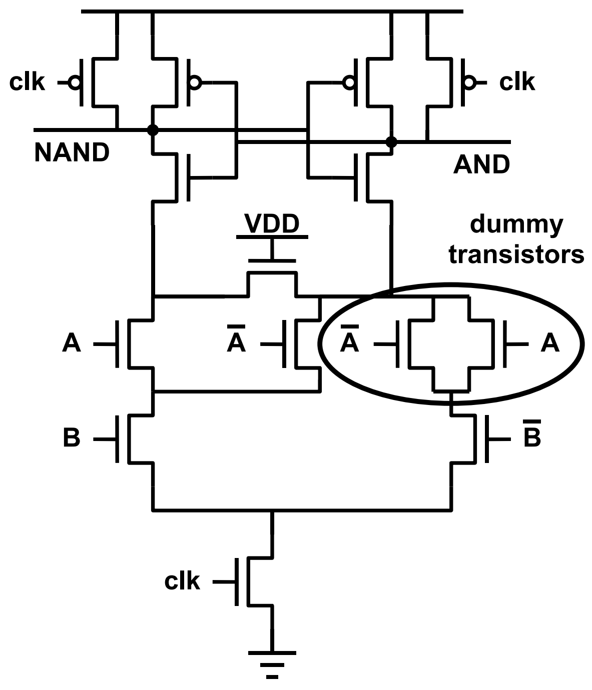
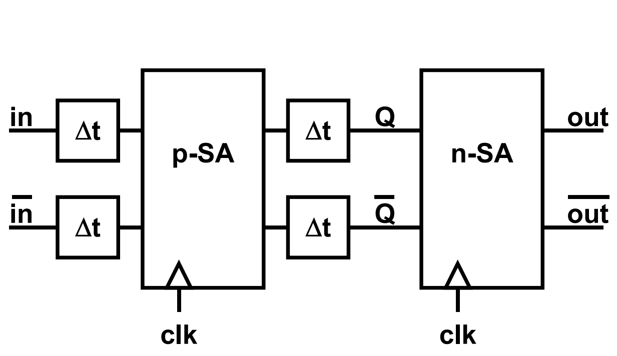


Figure 6: SABL AND-NAND gate with enhanced special DPDN.



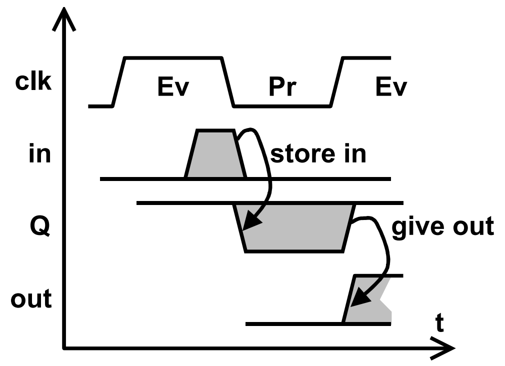


Figure 7: Negative edge triggered master-slave flip-flop: schematic (left); and operational behavior (right).

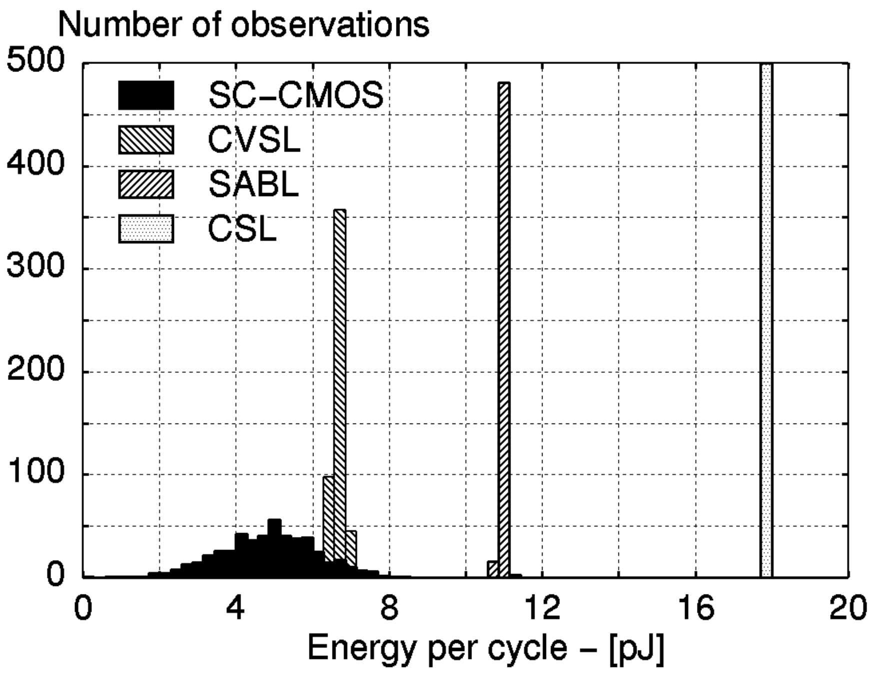


Figure 8: S9-box: histogram of the energy consumption per cycle for a simulated transient response of 500 cycles.

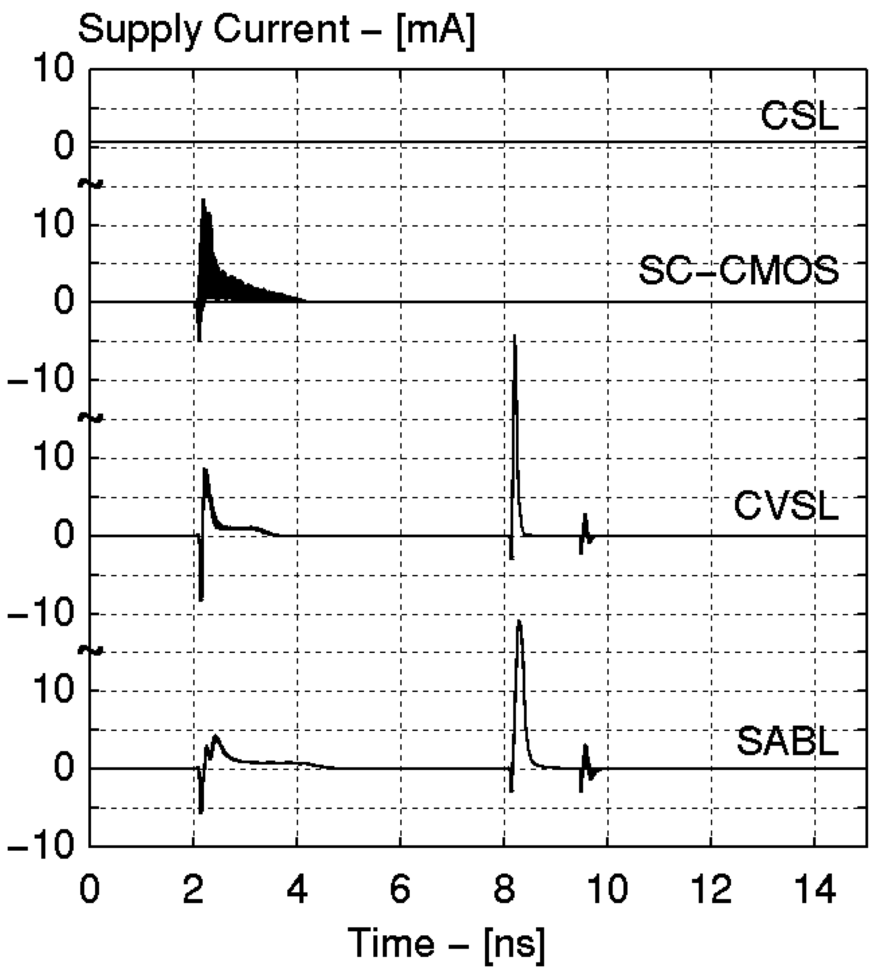


Figure 9: S9-box: superposition of the power supply current for 500 successive cycles of the simulated transient response.

TABLE I: S9-box: supply current description of simulated transient response and area

	NED	NSD	E_{mean}/cycle (pJ)	Area (μm^2)
SCMOS	1.000	0.260	4.97	22,500
CSL	0.002	0.000	18.10	45,500
CVSL	0.086	0.018	6.82	25,600
SABL	0.035	0.006	11.13	39,950