# Gate-Level Masking Under a Path-Based Leakage Metric

Andrew J. Leiserson, Mark E. Marson, and Megan A. Wachs

Cryptography Research, Inc.
425 Market Street, 11th Floor, San Francisco, CA 94105, USA
{andy,mark,megan}@cryptography.com

**Abstract.** Masking is a popular countermeasure against differential power analysis (DPA) and other side-channel attacks. When designing integrated circuits to resist DPA, masking at the logic gate level has the benefit that it can be implemented without consideration of the high-level function of the circuit. However, the phenomena of *glitches* and *early propagation* reduce the effectiveness of many gate-level masking schemes. In this paper we present a new technique for gate-level masking that is free of glitches and early propagation, yet requires only cell-level "don't touch" constraints. Our technique, which we call *LUT-Masked Dual-rail with Precharge Logic* (LMDPL), can therefore be implemented in a typical FPGA or standard cell ASIC design flow. LMDPL does not require routing constraints, nor sequencing of the evaluation of individual gates with enables, registers, or latches. We verify our techniques with an AES implementation on an FPGA. Our implementation shows no significant leaks in evaluations using up to 200 million traces.

**Keywords:** DPA, Side-Channel Analysis, Masked Logic, Dual-Rail Precharge Logic, Glitches, Early Propagation, AES, S-box

## 1 Introduction

Many devices leak information through side channels such as power consumption or radiated electromagnetic energy. Side channel analysis techniques such as differential power analysis (DPA) [11] can recover information about secrets manipulated in a cryptographic device. Given enough measurements, these techniques may enable an attacker to recover a portion, or the entirety, of a secret key intended to be kept secure within a cryptographic device.

Masking countermeasures [4] seek to prevent DPA attacks by making the electrical activity in a device independent of secret values being operated upon. This is done by dividing the secret into multiple shares. The shares can be combined to recover the original secret, but each share is random when considered individually. Thus, operations may be performed on the shares without leaking information about the secret. For example, given a secret value $k$ in some group $G$, a first-order additive masking uses a mask $m$ chosen randomly from $G$, and divides $k$ into the shares $m$ and $m + k$. Each of these shares is, when considered individually, independent of $k$.

Gate-level masking strategies attempt to construct masked versions of the elemental Boolean functions (AND, OR, etc.). For example, two common masked versions of a two-input Boolean function $f : a, b \to q$ are:

$$g(a \oplus m_a, b \oplus m_b, m_a, m_b, m_q) = f(a, b) \oplus m_q \tag{1}$$
$$h(a \oplus m, b \oplus m, m) = f(a, b) \oplus m \tag{2}$$

The masked function $g$ uses two independent mask bits for the inputs, and produces an output masked with a third mask that is independent of the input masks. The function $h$ uses a common mask bit that is reused for both of the inputs and the output.

Suppose we can construct a masked gate that can compute Boolean functions without leaking the unmasked values of the secret data $a$, $b$, and $q$. Then, more complex functions can be constructed from those masked gates, ideally in the same manner that any circuit can be constructed out of standard logic gates. Alternatively, given an implementation of some cryptographic circuit constructed using standard Boolean gates, the masked gates could perhaps be swapped for the standard gates to yield a masked implementation of the circuit. Our goal is to create such a masked gate.

### 1.1 Previous Work

Masking countermeasures have been studied extensively. We focus here on techniques that are most relevant to hardware implementations. Trichina et al. made an early proposal for a masked AND gate using four ANDs and four XORs [28]. Subsequent study found that direct implementation of masked operations in hardware may leak information through extraneous signal transitions known as *glitches*, due to the multitude of paths through the circuit [7, 13].

This led to the proposal of masked dual-rail with precharge logic (MDPL) [22]. MDPL avoids glitches through the use of precharged, monotonic, dual-rail logic, with each signal $\mathbf{x}$ encoded as a complementary pair $(x, \overline{x})$. The authors observe that the $h$ version of a masked AND gate can be implemented as:

$$\begin{aligned} q_m &= \mathrm{MAJ}(a_m, b_m, m) \\ \overline{q_m} &= \mathrm{MAJ}(\overline{a_m}, \overline{b_m}, \overline{m}) \end{aligned} \tag{3}$$

However, it was later shown that MDPL circuits exhibited significant first-order leakage due to *early propagation* [12, 21, 26]. Improved MDPL (iMDPL) addresses early propagation, but requires use of latches to control the moment at which gates evaluate [21].

In addition to the above issues, MDPL and other maskings of the form $h$ described in Eq. 2 may not provide adequate resistance against attacks that examine leakage distributions [6, 8, 24, 29]. Another technique that can be used to attack protected implementations is the *collision-correlation attack* [17]. This is a powerful technique for exploiting complex leakages such as those arising from incompletely masked combinational logic [14, 15].

The maskings shown in Eqs. 1 and 2 divide a secret into two shares. It is also possible to utilize more than two shares. Techniques from the field of multiparty computation may be used to perform computations without ever operating on all the shares simultaneously, thus ensuring immunity from glitch-related leakage [23]. However, a *memory effect* was identified, in which leakages from a computation can persist in a circuit for a period of time after the computation occurs. This phenomenon can impact the security of schemes thought to be immune to univariate attacks [16].

The technique of Prouff and Roche [23] performs shared multiplications in GF($2^8$). In contrast, *threshold implementations* instead use bitwise shares. The product of the values $x = x_1 + x_2 + x_3$ and $y = y_1 + y_2 + y_3$ is a collection of $x_i y_j$ terms, which can be allocated to output shares such that no single output share contains sufficent information to leak the secret. Thus, threshold implementations also address the problem of glitches [2, 18, 19, 20].

Of the foregoing techniques, threshold implementations offer the greatest promise for strong masking of arbitrary circuits, but doing so still requires insertion of additional registers in some cases. In this work, we offer a strategy for general gate-level masking that does not require additional registers.

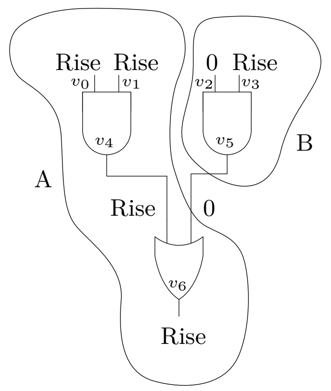
### 1.2 Roadmap

The paper is organized as follows. First, we briefly describe the idea of path-based leakage assessment. Next, we introduce LUT-Masked Dual-rail with Precharge Logic (LMDPL), a masking technique that is leak-free under a path-based leakage metric. Finally, we present some experimental results obtained from FPGA implementations of an LMDPL AES core.

## 2 Path-Based Leakage Assessment

Many previous countermeasures have been justified with arguments that the settled final values of each circuit node in each clock cycle are independent of secret data. However, such analyses cannot identify ways in which the transient electrical behavior may correlate with secret data. In practice, designs constructed without consideration of transient electrical behavior have remained vulnerable to side-channel attacks.

Most contemporary semiconductor devices are implemented using complementary metal-oxide-semiconductor (CMOS) or closely related technology. In CMOS technology, when a logic gate changes state, the parasitic capacitance at the inputs of downstream gates must be either charged or discharged. Ignoring quasi-static operating conditions such as supply voltage and temperature, the time it takes to (dis)charge the inputs of downstream gates still depends on many factors. The factors can include the number of inputs of a gate that are switching, the transition time (slew rate) at the switching inputs, and the logic state (voltage) present at non-switching inputs. When considering whether the electrical activity is independent of a secret, these effects should be considered cumulatively for the entire propagation path. For example, in a two-share scheme, if the output transition of an early gate exhibits a slight delay depending on the value of one share, and this output propagates to a gate at which the activity depends on the other share, the combination of these two effects may make the electrical activity at the downstream gate correlated with the unmasked secret.



| | $v_0$ | $v_1$ | $v_2$ | $v_3$ | $v_4$ | $v_5$ | $v_6$ |
|---|---|---|---|---|---|---|---|
| $A$ | r | r | | | r | 0 | r |
| $B$ | | | 0 | r | | 0 | |

**Fig. 1.** Two activity images, A and B, for a simple circuit

To investigate whether masked logic styles leak due to this type of electrical effect, we have developed a technique that we call *activity image analysis*. Due to space constraints we include only a brief description of the technique here. Activity image analysis determines whether electrical activity at upstream and downstream gates can combine to leak a secret by considering the switching events at adjacent gates jointly, rather than separately. The idea is illustrated in Fig. 1. A circuit is leak-free under an activity image metric if, for each activity image, observation of that image does not correlate with any secret value. This is a significantly stronger condition than balancing the distribution of final gate output values.

Activity image analysis is more comprehensive than toggle simulation analysis, which analyzes a single extracted model of propagation time through gates and wires, and applies to a single combination of operating conditions. Similar to structural clock domain crossing checks, activity image analysis examines the logical structure of the circuit and provides an assurance that is robust to timing variation. We also believe activity images can be helpful in detecting early propagation, but have no formal proof.

Appendix A shows an activity image analysis of iMDPL. Residual leakage in an iMDPL implementation due to circuit effects was also examined in [14]. Based on the results we have obtained from activity image analysis, we question whether mapping a single-rail circuit to a dual-rail circuit (as done e.g. in [5]) is an effective technique for producing first-order masked implementations.

## 3 LUT-Masked Dual Rail Logic

In this section, we introduce LMDPL, explain its usage, and then describe how we implemented AES using LMDPL.

### 3.1 The LMDPL Non-linear Gate

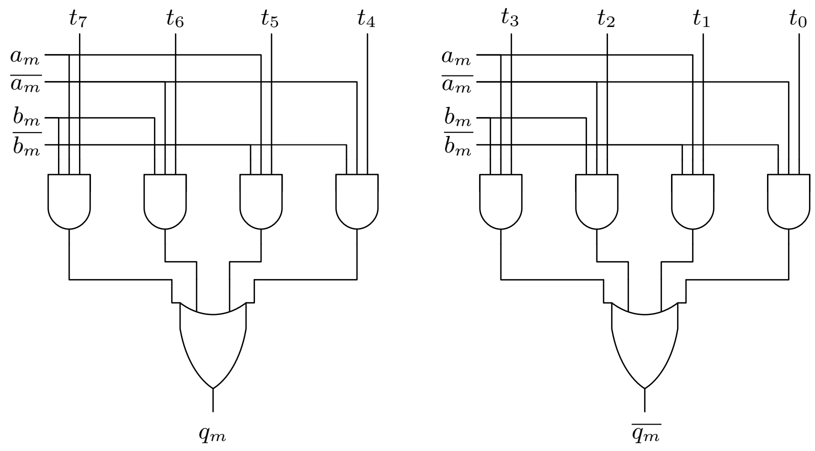


**Fig. 2.** LMDPL Non-linear Gate

It is well-known that linear functions are amenable to being computed on a shared representation of their argument, while non-linear functions pose substantial difficulty. Consequently, our efforts focused on identifying a way of computing non-linear functions in masked logic while satisfying the activity image leakage metric. We arrived at the dual-rail table lookup structure shown in Fig. 2. In our schematics, wires shown crossing at a right angle are not electrically joined, whereas wires shown meeting at a tee are electrically joined.

The LMDPL non-linear gate is intended to be used with a masking in the form of Eq. 1. Secret inputs $a$ and $b$ are converted to masked representation by obtaining two random mask values $m_a$ and $m_b$, and computing

$$\begin{aligned} a_m &= a \oplus m_a \\ b_m &= b \oplus m_b \end{aligned} \tag{4}$$

The values $m_a$ and $m_b$ constitute one share (the "mask share"), and $a_m$ and $b_m$ constitute the other share (the "masked data share"). In dual-rail logic, each logical value is represented by a complementary pair of signals, only one of which may be active at any time. The masked data values $a_m$ and $b_m$ are input in dual-rail encoding at the left of Fig. 2. The eight $t_i$ inputs at the top of the figure provide the values of a lookup table. By supplying the appropriate lookup table corresponding to the desired function $f$ and the mask values $m_a$, $m_b$, and $m_q$, the LMDPL gate produces a pair of complementary outputs that are a dual-rail encoding of $q_m$. We will return to the computation of the lookup table values in Section 3.2. Although the LMDPL non-linear gate may be used to implement an arbitrary two-input function, more compact alternatives are available for linear functions.

The structure shown in Fig. 2 is important. If EDA tools are permitted to freely restructure the logic, the gate will no longer pass a path-based leakage assessment. Fortunately, it is not difficult to instruct common EDA tools to preserve certain cell instantiations with a mechanism known as a *don't touch* constraint. Limited restructuring of the gate is acceptable. For example, ASIC implementations may prefer the NAND/NAND structure obtained by applying De Morgan's Law. We suggest some strategies for implementing LMDPL with common tools in Appendix B

Between each evaluation, the circuit must be precharged by driving both signals in each masked data pair to zero. Zeros on the four masked data inputs will propagate to the outputs, hence a precharge applied at the masked data inputs of a collection of LMDPL gates will propagate to the final outputs. During the evaluation of the gate, a transition away from zero on an output requires a non-zero value to have arrived on one of the component signals of each dual-rail input pair. Thus, the LMDPL gate does not exhibit early propagation.

LMDPL avoids glitches through the use of monotonic gates, in the same manner as the original MDPL. In the course of any evaluation, each of the $q_m$ and $\overline{q_m}$ outputs will transition at most once.

On any evaluation, exactly one of the AND gates in the LMDPL non-linear gate will produce a rising transition at the output. Even after fixing any or all of the unmasked data values, each of the eight AND gates has an equal probability of being the active gate upon each evaluation, depending upon the mask values. This effect is similar to Baddam et al.'s path switching countermeasure [1].

## 3.2 Implementing LMDPL

A simple circuit constructed using LMDPL is illustrated in Fig. 3. The circuit has three inputs, $x$, $y$, and $z$, and one output, $w$. The top portion of the figure operates on the mask share, and the bottom portion of the figure operates on the masked data share. The lookup tables $t_i$ for the LMDPL gates are passed from the mask share to the masked data share through registers. There are two non-linear gates, so the mask share takes two fresh mask bits from the RNG. Each of the mask share logic and masked data share logic is constructed by making modifications to the original circuit. The mask share retains linear elements, ties the output of each non-linear element to an RNG bit, and instantiates a "Table Gen" component for each non-linear element. The masked data share replaces the linear elements with corresponding dual-rail versions, and replaces the non-linear elements with instances of the LMDPL non-linear gate.

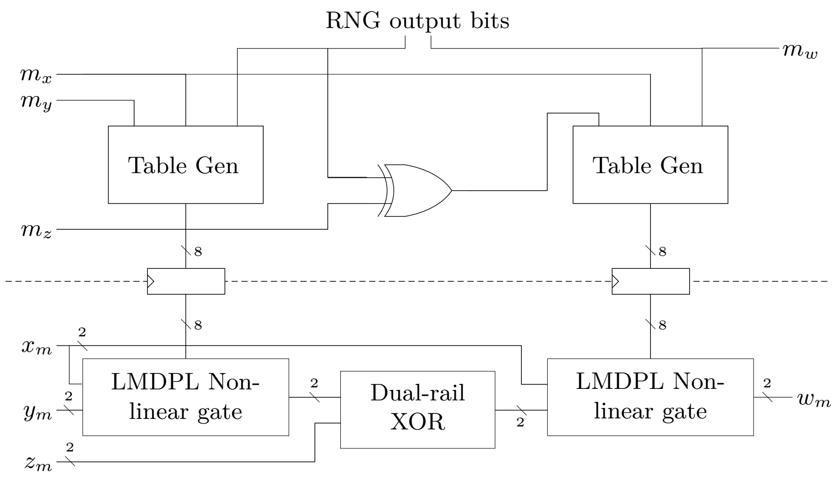


**Fig. 3.** A simple circuit using LMDPL

The “Table Gen” components compute the $t_i$ values for each non-linear gate in the manner typical of masked lookup tables. A function $f : \mathrm{GF}(2) \times \mathrm{GF}(2) \to \mathrm{GF}(2)$ is assigned to each table generator according to the original circuit. Each table generator accepts input masks $m_a$ and $m_b$ and an output mask $m_q$, which vary for each evaluation, and computes a varying table for each evaluation by the following formula.

Let $\mathbf{m} = (m_b, m_a)$ and $\mathbf{i} = (i_1, i_0)$ with $i_0, i_1 \in \{0, 1\}$. Then,

$$\begin{aligned} t_{4+2i_1+i_0} &= f(\mathbf{i} \oplus \mathbf{m}) \oplus m_q = f(i_1 \oplus m_b, i_0 \oplus m_a) \oplus m_q \\ t_{2i_1+i_0} &= t_{4+2i_1+i_0} \oplus 1 \end{aligned} \tag{5}$$

The non-linear function implemented by the LMDPL gate will commonly be a logical AND: $f(a, b) = a \cdot b$. The operation of the table generation logic for this case is shown in Table 1.

## 3.3 Implementing Linear Functions

Circuits typically also include gates that are linear (or affine) under boolean masking. When implementing linear gates, it is not necessary to consider the masking, so LMDPL is compatible with the linear gates from non-masked dual-rail logic styles such as WDDL [27]. We review briefly how to implement NOT and XOR gates.

A NOT gate can be implemented without any transistors, simply by swapping the complementary dual-rail signals. That is, $\mathbf{q} = \mathrm{NOT}(\mathbf{a})$ is implemented by:

$$\begin{aligned} q &= \overline{a} \\ \overline{q} &= a \end{aligned} \tag{6}$$

**Table 1.** Computation of the $t_i$ for $f(a,b) = a \cdot b$

| $m_q$ | $m_b$ | $m_a$ | $t_7$ | $t_6$ | $t_5$ | $t_4$ | $t_3$ | $t_2$ | $t_1$ | $t_0$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |

XOR gates should be implemented as monotonic logic (i.e., constructed out of AND and OR gates) to ensure the logic remains glitch-free and to correctly propagate the precharge state. An XOR gate $\mathbf{q} = \text{XOR}(\mathbf{a}, \mathbf{b})$ can be implemented as follows:

$$\begin{aligned} q &= \overline{a} \cdot b + a \cdot \overline{b} \\ \overline{q} &= \overline{a} \cdot \overline{b} + a \cdot b \end{aligned} \tag{7}$$

## 3.4 AES Implementation

To test the effectiveness of LMDPL, we developed an implementation of AES. The overall architecture of the AES implementation is shown in Fig. 4. The design computes a complete round transformation in parallel, and thus has 16 S-boxes. We favor simplicity and use a clock-based approach for the precharge, driving inputs to the LMDPL logic to zero in alternate cycles. Recall that sophisticated masking techniques are required only for non-linear operations, and the only non-linear operation in AES is the GF($2^8$) inversion within SubBytes. We implement only the inversion in LMDPL, and implement the remainder of the round transformation (including the linear portions of SubBytes) in ordinary logic. The sequence of operation is:

0. Initially, the LMDPL inversion logic is precharged.
1. In cycle 1 of a cipher operation, ordinary logic performs AddRoundKey and converts bytewise to the subfield basis used for inversion. The LMDPL logic is still precharged.
2. In cycle 2, the LMDPL logic computes bytewise inversion in GF($2^8$).
3. In cycle 3, ordinary logic converts bytewise to the standard AES basis, applies the SubBytes affine transformation, performs ShiftRows, MixColumns, and AddRoundKey, and then converts bytewise back to the subfield basis. Also in cycle 3, the LMDPL logic is precharged.
4. In subsequent even cycles, the LMDPL logic is active.
5. In subsequent odd cycles, the ordinary logic is active, and the LMDPL logic is precharged.

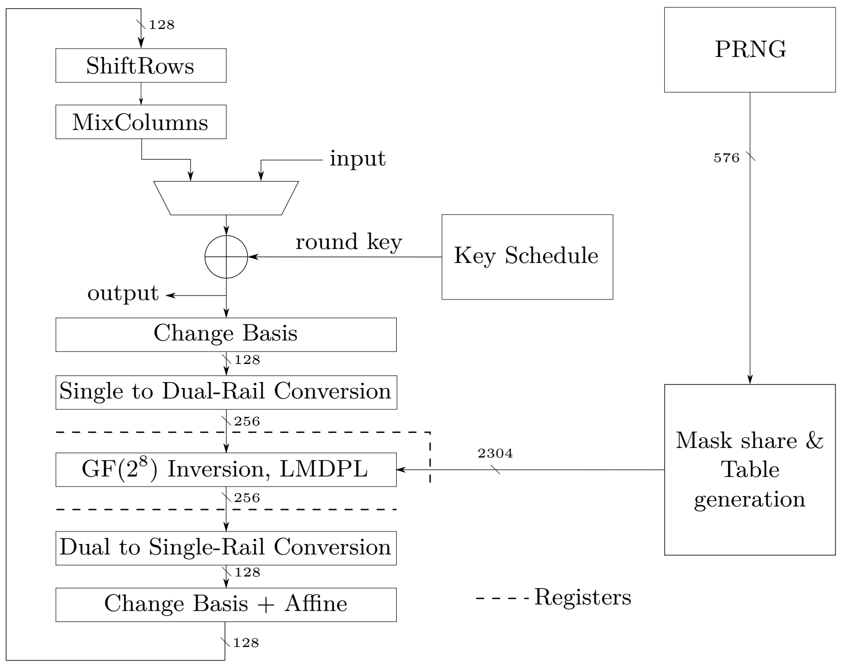


**Fig. 4.** Architecture of the AES implementation

Fig. 4 emphasizes the masked data share logic. The mask share logic (not shown in detail) mirrors the masked data share logic, with the table generation implemented according to Fig. 3, and without the need for registers surrounding the GF($2^8$) inversion.

The GF($2^8$) inversion uses the GF($((2^2)^2)^2$) normal basis identified in [3]. This implementation requires 36 bit-multiplications in GF(2). Some additional detail on our implementation of the inversion is provided in Appendix B.

The mask share (the $t_i$) would ideally be kept in the Hamming-weight-balanced 8-bit encoding to minimize leakages usable by second-order attacks. However, this is quite expensive. At some cost in resistance to second-order attacks, we generate and register only half of the table. The complementary half is obtained by inversion. In some cases, registers with complementary outputs could be used.

For purposes of comparison, we synthesized an ASIC version of our LMDPL AES core using Synopsys Design Compiler 2013.03-SP2. Table 2 compares our implementation with several others reported in the literature. Note that the threshold implementations [2, 18] have the advantage that the S-box can be pipelined, meaning the overall throughput is one S-box evaluation per clock rather than 1/latency. However, this benefit disappears in fully parallel implementations, as it is necessary to obtain the previous round's SubBytes output and apply the remaining transformations of an AES round before the next Sub-

**Table 2.** Comparison of implementations. Area reported both as count of Virtex 5 LUTs and as NAND2-normalized ASIC area ("Gate Equivalents"). Area does not include PRNG. LMDPL S-box area includes pre- and post-inversion data registers, single/dual rail conversion, table generation, table registers, and basis converters implemented in single-rail logic.

| | Random bits per S-box | S-box latency | Parallel AES | | Per S-box | |
|---|---|---|---|---|---|---|
| | | | LUTs | GE | LUTs | GE |
| [16] | 8 | 132 | 21,328 | | 1,387 | |
| [18] | 48 | 5 | | | | 4,244 |
| [2] | 44 | 3 | | | | 3,003 |
| This work | 36 | 2 | 8,538 | 59,311 | 447 | 2,825 |

Bytes input is ready. Also, note that although it requires fewer random bits per S-box, the parallel AES presented in this work requires more random bits in per-clock terms (576/2) compared to the threshold implementations with 8-bit datapaths (44/3 and 48/5). As was the case for the threshold implementations, we have provided ASIC area figures for comparison, while presenting evaluation results from an FPGA.

## 4 Experimental Results

This section presents assessments of DPA resistance on two designs incorporating LMDPL. Each design is described in Verilog, and implemented for Xilinx Virtex-5 FPGA using Synplify Pro 2009.03 and Xilinx ISE 13.2.

### 4.1 Evaluation Methodology

To evaluate the information leakage in different designs, we used the test vector leakage assessment (TVLA) methodology proposed by Goodwill et al. [9]. The TVLA methodology is designed to measure information leakage and provide an objective score. It specifies test vectors and uses Welch's t-test to measure the significance in the difference of means of two distributions. One of the tests in the methodology is known as the "fixed versus random" (FVR) test. In this test, the measurements are collected as the device operates repeatedly using fixed input data and randomly varying input data. (The fixed and random input vectors are randomly interleaved.) Welch's t-test is then used to score the differences between the two sets of measurements. We follow [9] and use $|t| < 4.5$ as the criteria for a passing result.

The fixed versus random evaluation technique does not target specific leaks. Rather, it measures aggregate information leakage at each point in time during the cryptographic operation. It is extremely powerful and can often find potential vulnerabilities with fewer traces than needed to identify specific leakages. In particular, for designs where the parallelism exceeds the portion of the key that can be guessed by a DPA attack, a leak identified by the FVR test is stronger than that which would actually be available to an attacker who cannot guess the entire key at once. Nevertheless, a failure of the FVR test does represent some correlation with secret intermediates, and the goal of masking is to eliminate such correlations.

Another characteristic of the FVR test is that false positives may arise due to the plaintext and ciphertext being fixed. The dilemma is similar to the need in conventional DPA attacks to select an intermediate separated from the plaintext or ciphertext by a non-linear function. We avoided the problem of input and output leaks by splitting the input into separate mask and masked data shares prior to transfer to the device under test (DUT), and likewise retrieving mask and masked data shares from the DUT before combining. We refer to this scheme as *externally applied masking* and the more conventional scheme where the DUT divides the data into shares as *internally applied masking.*

We also perform a variant of a collision correlation attack [17]. Our simulated collision correlation (SCC) attack operates by dividing the pool of traces into two equal-size groups and computing for each group the 256 means corresponding to the possible values of the S-box input. Then, for each of 256 possible "guesses" of a linear key byte distance, the means in one group are permuted according to the guess, and the correlation computed between the two sets of means. The unpermuted case represents the "correct" guess. To select points for this attack, we used one-way analysis of variance (ANOVA) to identify points with the strongest dependency on the S-box input value.

Our evaluation setup uses a Sasebo-GII board and a Signatec PX14400A PCI-E card for data acquisition. The signal is taken from the $1\,\Omega$ supply-side sense resistor on the Sasebo-GII and connected through a Mini-Circuits BLK-89-S+ DC blocker, a Mini-Circuits BLP-150+ LPF, and a Mini-Circuits ZFL-1000+ amplifier before driving the input of the Signatec card, which has a sample rate of 400 MS/s and 14 bits of resolution.

The design operates at 24 MHz. Our evaluation harness performs 2,000 consecutive AES operations with data obtained from and stored to buffers on the FPGA. The design provides a trigger signal concurrent with the start of the first AES operation. This signal is used as an external trigger for the Signatec card. To ensure that the 400 MS/s sample rate does not impact the alignment quality when analyzing our traces, we use a technique similar to that of [10] to achieve sub-sample alignment resolution.

### 4.2 Results from a Single S-box Design

Prior to presenting results from the full AES implementation, we present results from a simplified design. The simplified design maintains the 128-bit parallel datapath of the full AES implementation, but replaces 15 of the 16 S-boxes with passthroughs. We chose this approach, rather than a true 8-bit datapath AES, to focus on leakage from the LMDPL S-box as opposed to leakage from registers.

We first disabled the mask generator and collected waveforms from 10,000 encryptions. For each encryption, we chose with even probability between the

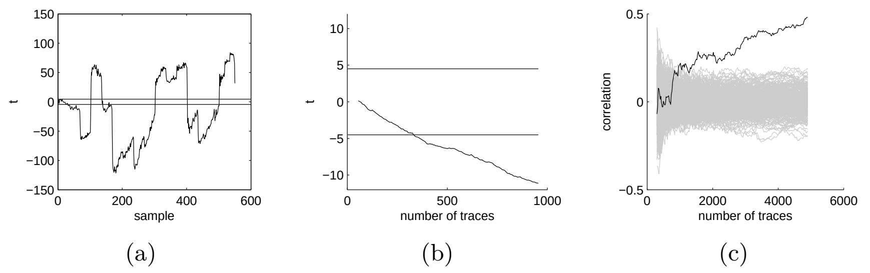


**Fig. 5.** Single S-box design, masking disabled. (a) sample-wise $t$ statistic on 10,000 traces, (b) sample 71 $t$ statistic vs. number of traces, (c) SCC attack using sample 71

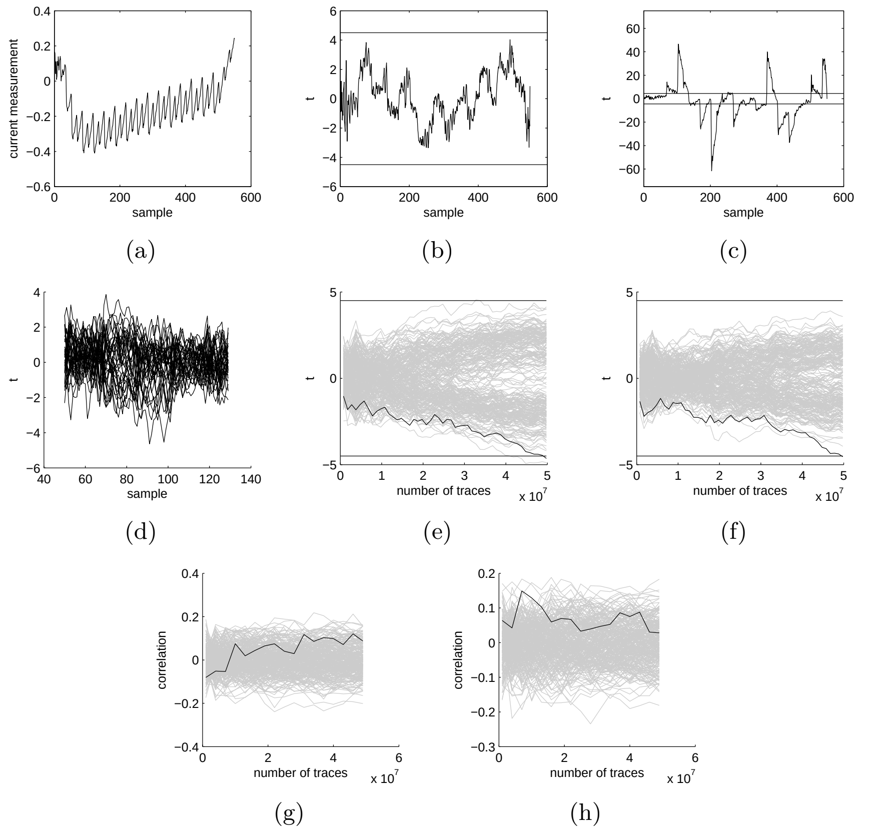


**Fig. 6.** Single S-box design, masking enabled. (a) average of 100M traces, (b) sample-wise FVR $t$ statistic, (c) sample-wise 2nd-order FVR $t$ statistic, (d) overlay of 2nd-order $t$ for each of the 36 S-box non-linear gate outputs, (e) $t$ for each of 256 possible key guesses, bit 25 sample 92, (f) $t$ for each of 256 possible key guesses, bit 25 sample 99, (g) first-order SCC using sample 86, and (h) second-order SCC using sample 86.

fixed plaintext and a random plaintext. Fig. 5 shows several analyses of these traces. A sample-wise plot of the $t$ statistic (a) immediately indicates that the design is leaking. We selected sample 71, the point in the first round with the greatest $|t|$ value, for further analysis. For this design, slightly over 300 traces are needed before the $|t| > 4.5$ threshold is reached for sample 71 (b). We then performed a SCC attack at sample 71. For this evaluation, between 1,000 and 1,500 traces are needed before the correct guess becomes dominant (c).

We next enabled the mask generator and collected 100,000,000 traces, again choosing evenly between a fixed plaintext or a random plaintext for each encryption. Fig. 6 presents analysis of these traces. With the masking enabled, the $t$ statistic does not exceed the $|t| > 4.5$ threshold with 100M traces (b), demonstrating that the first-order masking is effective. However, the design exhibits second-order leakage, as can be seen by using the $t$ statistic to compare the squared residuals between the two groups (c).

We used the 50,000,000 random traces out of the same data set to develop an attack exploiting the second-order leakage. We sorted the traces based on the output from each of the 36 non-linear gates in the S-box. The difference in variance due to the value of a single bit is smaller than the difference that arises when the entire plaintext is fixed, but it it still detectable. We examined all 36 candidates (d) and selected for the attack the bit and time sample combinations with the largest $|t|$. The first candidate, bit 25 at sample 92, does not result in selection of the correct key guess with 50 million traces (e). The second candidate, bit 25 at sample 99, does result in the selection of the correct key guess with 50 million traces (f). First- and second-order versions of our SCC attack on this design were not successful (g,h).

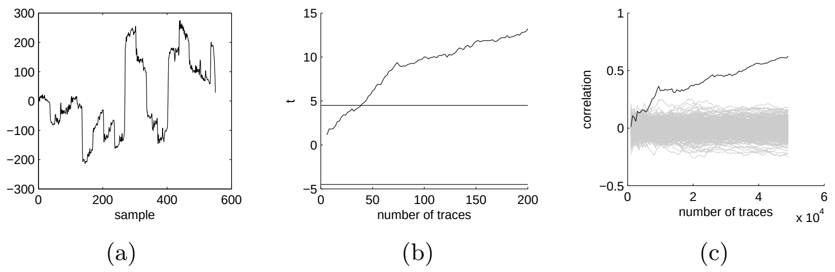


**Fig. 7.** Parallel design, masking disabled. (a) sample-wise FVR $t$ statistic on 100,000 traces, (b) sample 441 FVR $t$ statistic vs. number of traces, (c) SCC attack using sample 71

## 4.3 Results from a Parallel Design

One possible strategy to improve upon the resistance of the single S-box implementation would be to incorporate higher-order masking. However, in low-noise environments, the security benefit of higher-order masking is limited [25]. With this in mind, we explored the resistance of an AES-256 implementation performing SubBytes on the entire round state in parallel.

We again measured the design with masking disabled as a baseline. For the parallel design we collected 100,000 traces. For each trace, we chose randomly between the fixed plaintext or a random plaintext. Fig. 7 shows our analysis of these traces. Fig. 7(a) is a plot of the FVR $t$ statistic versus the sample index, and as with the corresponding plot for the serial implementation, provides immediate evidence that the design is leaking. Fig. 7(b) plots the $t$ statistic between the fixed and random traces at sample 441 (the sample with the largest absolute $t$ value), and shows that less than 50 traces are needed before the $|t| > 4.5$ threshold is reached. Finally, Fig. 7(c) shows the results of our SCC attack at sample 71. The correct key guess becomes dominant after about 10,000 traces.

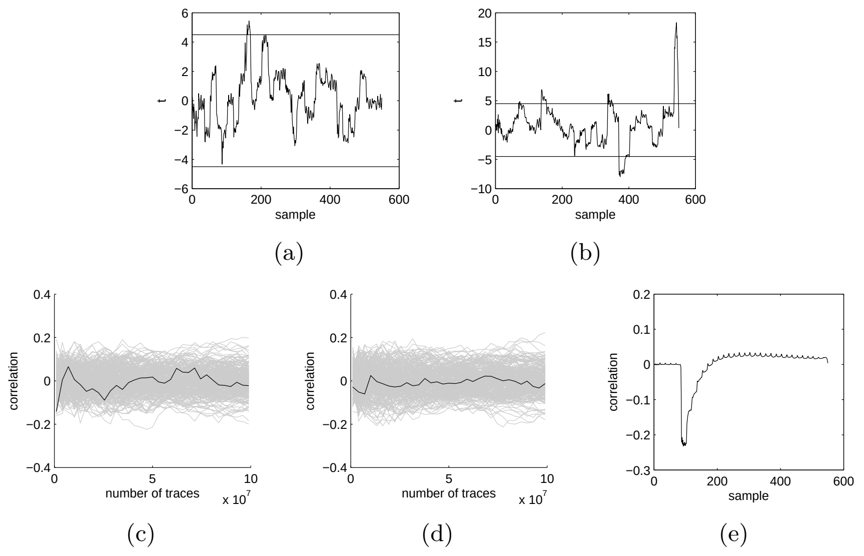


**Fig. 8.** Parallel design, masking enabled. (a) sample-wise FVR $t$ statistic on 200,000,000 traces, (b) sample-wise 2nd-order FVR $t$ statistic on 200,000,000 traces, (c) first-order SCC attack at sample 87, (d) second-order SCC attack at sample 87, (e) CPA vs. round1-round2 mask Hamming distance

Finally, we enabled the mask generator in our parallel design and collected 200,000,000 traces. Fig. 8 shows our results. The first-order FVR $t$ has only marginally exceeded the $|t| > 4.5$ threshold with 200,000,000 traces. In contrast with the serial implementation, where the second-order $t$ statistic reached significantly larger values than the first-order $t$, the second-order $t$ for the parallel implementation reaches only slightly larger values than the first-order $t$. The spike at the end of the second-order analysis in Fig. 8(b) is due to the final masked output, and mask, being manipulated at the end of the calculation, and does not represent a leak of sensitive information. We performed the SCC attack on this design, and it was not successful (c,d).

Fig. 8(e) is shown to demonstrate a technique that we use to investigate the behavior of our designs and to verify that our data collection is correct. The masked implementation used for evaluation allows re-seeding of the PRNG with an externally-supplied per-encryption seed. This allows us to compute the values of circuit intermediates that are a function of the mask, which would normally not be predictable by an attacker. The figure shows the correlation between the current measurement and the Hamming distance between the round one and round two masks. Because the mask values for successive rounds overwrite each other in the mask share logic, a strong correlation is expected and is indeed present. We additionally note that the memory effect [16] is clearly visible here. The register update occurs at the time of the initial downward spike around sample 83. A strong correlation exists for around 50 samples (3 clock cycles) after the register update, and a weak correlation persists throughout the encryption.

## 5 Conclusion

In this work, we propose the use of a path-based model for the leakage from combinational circuits. Unlike traditional methods that focus on the settled values of circuit nodes, activity image analysis considers ways that data-dependent behavior can accumulate as transitions propagate through combinational logic.

We also present LMDPL, a new technique for gate-level masking. LMDPL compares competitively or favorably with previous techniques on multiple metrics. Furthermore, LMDPL does not require routing constraints, and does not require that sequential elements or enable signals be used to delay the propagation of signals through the circuit.

## References

1. Baddam, K., Zwolinski, M.: Path switching: a technique to tolerate dual rail routing imbalances. Des Autom Embed Syst 12(3), 207–220 (2008).
2. Bilgin, B., Gierlichs, B., Nikova, S., Nikov, V., Rijmen, V. *A More Efficient AES Threshold Implementation.* Cryptology ePrint Archive, Report 2013/697. `http://eprint.iacr.org/2013/697`.
3. Canright, D.: A Very Compact S-Box for AES. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 441–455. Springer, Heidelberg (2005).
4. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards Sound Approaches to Counteract Power-Analysis Attacks. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 398–412. Springer, Heidelberg (1999).
5. Chen, Z., Haider, S., Schaumont, P.: Side-Channel Leakage in Masked Circuits Caused by Higher-Order Circuit Effects. In: Park, J.H., Chen, H.-H., Atiquzzaman, M., Lee, C., Kim, T.-h., Yeo, S.-S. (eds.) Advances in Information Security and Assurance. LNCS, vol. 5576, pp. 327–336. Springer, Heidelberg (2009).

6. De Mulder, E., Gierlichs, B., Preneel, B., Verbauwhede, I.: Practical DPA attacks on MDPL. In: First IEEE International Workshop on Information Forensics and Security, 2009. WIFS 2009, pp. 191–195 (2009).
7. Fischer, W., Gammel, B.M.: Masking at Gate Level in the Presence of Glitches. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 187–200. Springer, Heidelberg (2005).
8. Gierlichs, B.: DPA-Resistance Without Routing Constraints? In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 107–120. Springer, Heidelberg (2007).
9. Goodwill, G., Jun, B., Jaffe, J., Rohatgi, P.: A testing methodology for side-channel resistance validation. In: Non-Invasive Attack Testing Workshop, Nara (2011). `http://csrc.nist.gov/news_events/non-invasive-attack-testing-workshop/papers/08_Goodwill.pdf`.
10. Homma, N., Nagashima, S., Imai, Y., Aoki, T., Satoh, A.: High-Resolution Side-Channel Attack Using Phase-Based Waveform Matching. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 187–200. Springer, Heidelberg (2006).
11. Kocher, P., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999).
12. Kulikowski, K., Karpovsky, M., Taubin, A.: Power attacks on secure hardware based on early propagation of data. In: 12th IEEE International On-Line Testing Symposium, pp. 131–138. IEEE Computer Society Press, Los Alamitos (2006).
13. Mangard, S., Pramstaller, N., Oswald, E.: Successfully Attacking Masked AES Hardware Implementations. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 157–171. Springer, Heidelberg (2005).
14. Moradi, A., Kirschbaum, M., Eisenbarth, T., Paar, C.: Masked Dual-Rail Precharge Logic Encounters State-of-the-Art Power Analysis Methods. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 20(9), 1578–1589 (2012).
15. Moradi, A., Mischke, O.: How Far Should Theory Be from Practice? In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 92–106. Springer, Heidelberg (2012).
16. Moradi, A., Mischke, O.: On the Simplicity of Converting Leakages from Multivariate to Univariate. In: Bertoni, G., Coron, J.-S. (eds.) CHES 2013. LNCS, vol. 8086, pp. 1–20. Springer, Heidelberg (2013).
17. Moradi, A., Mischke, O., Eisenbarth, T.: Correlation-Enhanced Power Analysis Collision Attack. In: Mangard, S., Standaert, F.-X. (eds.) Cryptographic Hardware and Embedded Systems, CHES 2010. LNCS, vol. 6225, pp. 125–139. Springer, Heidelberg (2010).
18. Moradi, A., Poschmann, A., Ling, S., Paar, C., Wang, H.: Pushing the Limits: A Very Compact and a Threshold Implementation of AES. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 69–88. Springer, Heidelberg (2011).
19. Nikova, S., Rechberger, C., Rijmen, V.: Threshold Implementations Against Side-Channel Attacks and Glitches. In: Ning, P., Qing, S., Li, N. (eds.) Information and Communications Security. LNCS, vol. 4307, pp. 529–545. Springer, Heidelberg (2006).
20. Nikova, S., Rijmen, V., Schläffer, M.: Secure Hardware Implementation of Nonlinear Functions in the Presence of Glitches. Journal of Cryptology 24(2), 292–321 (2010).
21. Popp, T., Kirschbaum, M., Zefferer, T., Mangard, S.: Evaluation of the Masked Logic Style MDPL on a Prototype Chip. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 81–94. Springer, Heidelberg (2007).

22. Popp, T., Mangard, S.: Masked Dual-Rail Pre-charge Logic: DPA-Resistance Without Routing Constraints. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 172–186. Springer, Heidelberg (2005).
23. Prouff, E., Roche, T.: Higher-Order Glitches Free Implementation of the AES Using Secure Multi-party Computation Protocols. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 63–78. Springer, Heidelberg (2011).
24. Schaumont, P., Tiri, K.: Masking and Dual-Rail Logic Don't Add Up. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 95–106. Springer, Heidelberg (2007).
25. Standaert, F.-X., Veyrat-Charvillon, N., Oswald, E., Gierlichs, B., Medwed, M., Kasper, M., Mangard, S.: The World Is Not Enough: Another Look on Second-Order DPA. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 112–129. Springer, Heidelberg (2010).
26. Suzuki, D., Saeki, M.: Security Evaluation of DPA Countermeasures Using Dual-Rail Pre-charge Logic Style. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 255–269. Springer, Heidelberg (2006).
27. Tiri, K., Verbauwhede, I.: A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation. In: DATE 2004, vol. 1, pp. 246–251. IEEE Computer Society Press, Los Alamitos (2004).
28. Trichina, E., Korkishko, T., Lee, K.H.: Small Size, Low Power, Side Channel-Immune AES Coprocessor: Design and Synthesis Results. In: Dobbertin, H., Rijmen, V., Sowa, A. (eds.) Advanced Encryption Standard – AES. LNCS, vol. 3373, pp. 113–127. Springer, Heidelberg (2005).
29. Ye, X., Eisenbarth, T.: On the Vulnerability of Low Entropy Masking Schemes. In: CARDIS 2013, Berlin (2013).

## A Activity Image Analysis Example

Table 3 presents an activity image analysis of iMDPL in tabular form. Each row describes one activity image. Columns to the left of the double line represent states observed at the output of each gate in the circuit. The columns to the right of the double line are labeled with a value $x^*$ of the secret $x = x_m \oplus m$, and entries in those columns report the count of observations of that row's activity image when $x$ takes the value $x^*$. In this example there are eight possible inputs to the circuit, corresponding to the two possible values for each of $x_m$, $y_m$, and $m$. Evaluation of a circuit for a given input may exhibit multiple activity images.

We define a circuit to be balanced under the activity image metric if, for any value $x^*$ that the secret $x$ may take, the conditional probability that $x = x^*$, given that some particular activity image was observed, is the same as the unconditional probability that $x = x^*$. In other words, the observation counts in each row of the table must have the same proportion as the global probabilities of the associated $x^*$ values. In the case of iMDPL, $\Pr\{x = 0\} = \Pr\{x = 1\} = 0.5$, so the requirement is that the counts in each row be equal. The iMDPL circuit is not balanced, as the observation counts are different in 6 of the 10 rows. The same analysis for the LMDPL circuit (omitted for space reasons) shows that it is balanced under this metric.

**Table 3.** iMDPL AND, assuming AND/OR decomposition

$$a0 = x_m \cdot y_m \qquad a1 = x_m \cdot m \qquad a2 = y_m \cdot m \qquad q_m = a0 + a1 + a2 \tag{8}$$

| $x_m$ | $y_m$ | $m$ | a0 | a1 | a2 | $q_m$ | $x^* = 0$ | $x^* = 1$ |
|---|---|---|---|---|---|---|---|---|
|  | 0 | r |  |  | 0 |  | 1 | 1 |
|  | r | 0 |  |  | 0 |  | 1 | 1 |
|  | r | r | 0 | 0 | r | r | 0 | 1 |
| 0 |  | r |  | 0 |  |  | 0 | 2 |
| 0 | r |  | 0 |  |  |  | 1 | 1 |
| r |  | 0 |  | 0 |  |  | 0 | 2 |
| r |  | r | 0 | r | 0 | r | 1 | 0 |
| r | 0 |  | 0 |  |  |  | 1 | 1 |
| r | r |  | r | 0 | 0 | r | 0 | 1 |
| r | r | r | r | r | r | r | 1 | 0 |

## B Details on implementing AES with LMDPL

As discussed in Section 3.1, LMDPL requires the structure of the non-linear gate be preserved with don't touch constraints (sometimes called "keep" constraints). For an ASIC design, library cells implementing the elemental functions may be instantiated in the HDL description, and a don't touch attribute applied to the instantiations. A common and simple way to do this is to use a distinguishing prefix in the instance names, and use a wildcard pattern to identify for the tool the cells not to touch. For either an ASIC or an FPGA, the elemental functions (AND/OR/NAND) used in the gate may be placed in a dedicated module, and a hierarchy-preserving attribute or directive applied to that module.

For the Virtex 5 FPGA, hierarchy preservation attributes limited the amount of packing the place and route tools would perform. We obtained better results by applying net preservation directives to the interface of the modules implementing the elemental functions, or to the interface of the module implementing the LMDPL gate. For example, preserving the interface of the dual-rail XOR (a 4-input, 2-output function) allows it to be packed in a single dual-output LUT. Similarly, appropriate constraints enable the eight AND gates of the non-linear gate to be packed pairwise into four dual-output LUTs.

Our AES implementation incorporates an optimized inversion circuit, which uses functions other than AND for some of the 36 non-linear gates. We created a Liberty-format library description containing cells of unit area implementing the XOR and XNOR functions, and cells of ten units area implementing each of the non-linear two-input boolean functions. We then used Synopsys Design Compiler to map the normal-basis GF($2^8$) inversion onto this library. The netlist from Design Compiler contained 37 non-linear gates rather than the expected 36, however, inspection revealed that two of the non-linear gates could be combined with minor rearrangement of neighboring XORs to achieve a 36-gate implementation. This optimized circuit was used as the basis for translation to LMDPL mask and masked data share implementations.