Transactions Briefs

Extracting Secret Keys From Integrated Circuits

Daihyun Lim, Jae W. Lee, Blaise Gassend, G. Edward Suh, Marten van Dijk, and Srinivas Devadas

***Abstract*—Modern cryptographic protocols are based on the premise that only authorized participants can obtain secret keys and access to information systems. However, various kinds of tampering methods have been devised to extract secret keys from conditional access systems such as smartcards and ATMs. Arbiter-based physical unclonable functions (PUFs) exploit the statistical delay variation of wires and transistors across integrated circuits (ICs) in manufacturing processes to build unclonable secret keys. We fabricated arbiter-based PUFs in custom silicon and investigated the identification capability, reliability, and security of this scheme. Experimental results and theoretical studies show that a sufficient amount of inter-chip variation exists to enable each IC to be identified securely and reliably over a practical range of environmental variations such as temperature and power supply voltage. We show that arbiter-based PUFs are realizable and well suited to build, for example, key-cards that need to be resistant to physical attacks.**

***Index Terms*—Identification, physical random function, process variation, tamper resistance, unclonability.**

I. Introduction

For many applications that need to identify and authenticate users, system security is based on the protection of secret keys. In these applications, malicious users can impersonate authorized users when the secret key is uncovered. Thus, secret key storage devices, such as smartcards, provide active logical controls to protect the secret keys against various kinds of logical and physical tampering attacks.

However, recently developed invasive and noninvasive physical tampering methods [1] such as micro-probing, laser cutting, glitch attacks, and power analysis have made it possible to extract digitalized secret information from integrated circuits (ICs) and compromise conditional access systems by using illegal copies of the secret information. For example, an adversary can remove a smartcard package and reconstruct the layout of the circuit using chemical and optical methods. Even the data in nonvolatile memories such as EEPROM and NVRAM can be revealed by sophisticated tampering methods.

To prevent these physical attacks, researchers have invented various protection mechanisms. For example, additional metallization layers that form a sensor mesh above an actual circuit can be introduced to provide a tamper-sensing environment [1]. This sensor mesh technique has been used in some commercial smartcard CPUs such as ST16SF48A and in some battery-buffered SRAM security processors such as DS5002FPM and DS1954 [2]. Using a sensor mesh, any interruption and short-circuit can be monitored while power is available, and a laser cutter or selective etching access to bus lines can be prevented. Though the tamper-sensing environment can cause difficulty for an adversary, the sensor mesh cannot prevent the extraction of hardwired information when the circuit power is off.

As an alternative, pseudorandom functions can be used to avoid storing actual bits of secret keys in digital circuits [3]. However, sophisticated reverse engineering still enables an adversary to build illegal copies of the circuits that implements the pseudorandom functions. Instead of using digital circuitry, we propose to build random functions based on the randomness in physical materials. This type of random function is resistant to the cloning attacks because an adversary is not in control of the heterogeneous materials that hold secret information, and, therefore, he cannot physically clone such a random function.

Pappu *et al.* introduce the concept of physical one-way functions (POWFs) in [4] and [5]. They use a transparent optical medium with a three-dimensional (3-D) micro-structure as a POWF. The input/challenge of the POWF is an incoming laser beam and the output/response of the POWF is a fixed-length bit vector derived from the resulting interference pattern. The interference pattern depends on the angle and frequency of the incoming beam and the speckle pattern in the optical medium. The total number of possible challenge-response pairs (CRPs) depends on the volume of the optical medium and the minimal possible wavelength of the laser beam. There are tens of thousands of possible CRPs. This means that it is feasible for an adversary to measure and create a complete list of possible CRPs which can be used as a counterfeit of an original POWF.

Gassend *et al.* [6], [7] introduce the concept of silicon physical random functions, also called silicon physical unclonable functions (PUFs). Silicon is a good base material to build physical random functions because we can use common CMOS manufacturing processes. Given a fixed challenge, a corresponding response varies across different ICs because the PUF responses are designed to be sensitive to circuit delays which vary across ICs due to process variation in transistors and wires. Since process variation is beyond the manufacturers' control, even an adversary who has the detailed information of the PUF circuit cannot physically clone the silicon PUF. Silicon PUFs have an exponential number of possible CRPs such that the "software clone" based on the exhaustive measurement of possible CRPs is impossible.

In silicon PUFs, noise is an important issue because circuit delays are sensitive to environmental variations such as temperature and power supply voltages. In this paper, we propose a scheme named *arbiter-based PUF* that improves the reliability of the PUFs against environmentally induced noise by using a differential structure. Instead of measuring absolute delay values for PUF responses, we compare two identical delay paths and generate digital information using an *arbiter*. Arbiter-based PUF test-chips were fabricated using a TSMC 0.18-μm process. From experimental results, we infer the identification capability, reliability, and security of the arbiter-based PUF scheme. Lastly, we propose an alternative architecture to improve the security of the arbiter-based PUFs against software model-building attacks by adding unpredictable nonlinearity.

II. Arbiter-Based PUF

A. General Description of Arbiter-Based PUFs

An arbiter-based PUF is composed of delay paths and an arbiter located at the end of the delay paths. Fig. 1 depicts an arbiter-based PUF circuit. In this scheme, we excite two delay paths simultaneously and make the transitions race against each other. The arbiter determines which rising edge arrives first and sets its output to 0 or 1 depending

Manuscript received October 7, 2004; revised July 3, 2005.

The authors are with the Massachusetts Institute of Technology, Cambridge, MA 02139-4307 USA (e-mail: daihyun@mit.edu).

Digital Object Identifier 10.1109/TVLSI.2005.859470

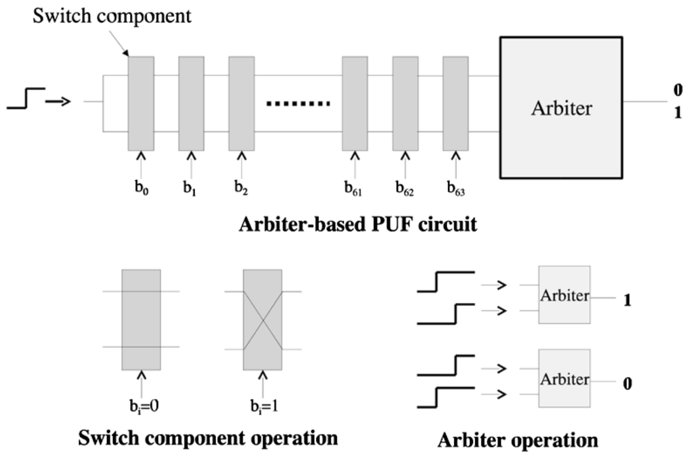
Switch component
Arbiter
0
1
b_0 b_1 b_2 b_{61} b_{62} b_{63}
Arbiter-based PUF circuit
b_i=0
b_i=1
Switch component operation
Arbiter
Arbiter
Arbiter operation

Fig. 1. Structure of an arbiter-based PUF (basic arbiter scheme).

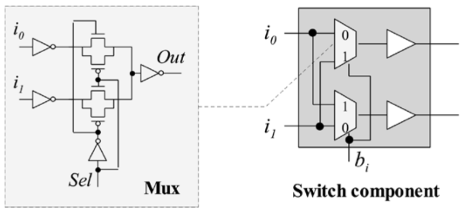

Fig. 2. Implementation of a switch component.

on the winner. The circuit takes 64 challenge bits (b_i) as an input to configure the delay paths and generate a 1-b response as an output.

There are m switches, and each of them can change the configuration of the delay paths. Thus, the number of possible different configurations of the delay paths is 2^m, i.e., exponential in m. The delay difference between the top and bottom paths is determined by the configuration of delay paths. The response of the arbiter can vary across ICs if the maximum delay variation in manufacturing is greater than the delay difference. For the challenges whose delay difference is greater than the maximum delay variation, responses are only biased to 0 or 1 and do not change across ICs. Thus, in order to maximize the inter-chip variation of PUF responses, the delay paths must be placed and routed as symmetrically as possible so as to minimize the nominal delay difference between two paths.

Fig. 2 details a switch component. A switch connects its two input ports (i_0 and i_1) to the output ports (o_0 and o_1) with different configurations depending on the control bit (b_i); for $b_i = 0$, the paths go straight through, while for b_i=1 they are crossed. It is implemented with a pair of 2-to-1 multiplexers and buffers. In our test-chips, the cells are placed and routed symmetrically and the wires in the delay circuit effectively cover the entire chip. This layout technique makes it extremely difficult for an adversary to probe internal nodes to read out a logic value without breaking the PUF, i.e., without changing the delays of wires or transistors.

For an arbiter, a simple transparent data latch has been used. Fig. 3 shows the operation of the latch as an arbiter. If the rising edge of a data input D comes earlier than a rising edge of a gate input G by more than the setup time of the latch, an output Q samples 1. Otherwise, Q becomes 0.

In implementation, the standard cell of a D-latch was used for the arbiter. We note that an output is preset to 0 and input signals must satisfy the setup time of a latch to switch the output to 1. Therefore, the arbiter favors the path to output 0. This property introduces a skew factor in the delay model of arbiter-based PUFs. As a result, only 10% of total responses from the test-chips are 1's on average. This imbalance

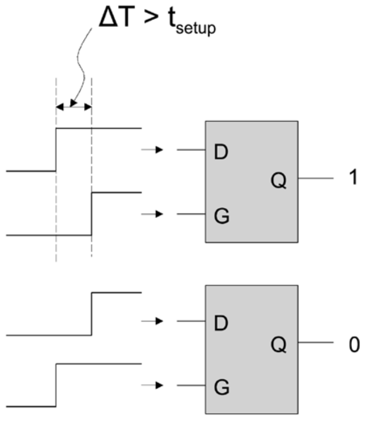

Fig. 3. Signal transitions of simple transparent data latch for an arbiter.

can be improved by using more symmetrically implementable latches (e.g., SR latches). In practice, to compensate for the skews, we fix some of the most significant challenge bits to effectively lengthen the delay path connected to a gate input. Experimental results show that we need to fix 12 out of 64 challenge bits $(b_{52}, \ldots, b_{63})$ to achieve balanced responses.

B. Characteristics of Arbiter-Based PUFs

To identify individual PUF chips, we generate challenge-response pairs (CRPs) of each PUF. We define the inter-chip variation τ between two different PUFs as the probability that responses from two PUFs are different from each other for a random challenge. If τ is sufficiently large compared to the noise probability, then, by generating a sufficient number of CRPs, we can distinguish different ICs with negligible probability of error.

Environmental variation, metastability, and aging cause noise in measurements of PUF responses. To quantify the effect of noise, we define the *noise* (μ) as the probability that a newly measured response of a challenge for a given IC is different from the corresponding *reference response* of the same challenge for the same IC. To estimate the identification capability of the arbiter-based PUF scheme, this noise probability should be precisely measured by experiments.

In the arbiter-based PUF scheme, we use a relative delay measurement which can reduce a significant amount of response noise induced by environmental variation. Even if environmental variation changes the absolute values of two delays, the difference between the two delays is likely to be preserved; relative delay measurement significantly reduces the effect of environmental noise. Additionally, this measurement scheme takes only one cycle to finish response generation as opposed to the ring oscillator scheme in [6] and [7]. We can increase the number of bits by replicating the same circuit up to the desired number of bits.

Alternatively, the arbiter-based PUF can be used as a hardware random number generator by using the metastable responses. The arbiter generates metastable responses when the delay differences between the top and bottom paths are negligibly small. Arbiter responses are determined by thermal noise in the circuit and a sequence of random numbers can be generated by repeating the evaluation of the arbiter response. This PUF-based random number generator can be used for low-cost security applications that need nonalgorithmic random number generation [8].

C. Security of Arbiter-Based PUFs

An arbiter-based PUF circuit can be represented as a linear model, and an adversary can use machine learning algorithms to build a software model of a PUF circuit [9], [10]. An adversary can use the soft-

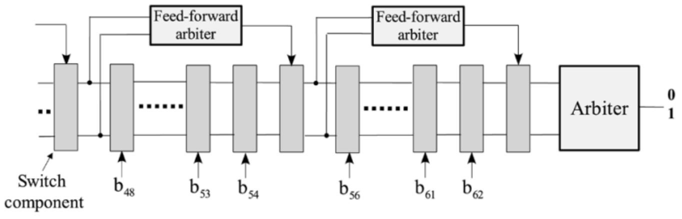

Fig. 4. Feed-forward arbiter scheme for improved security.

ware model as a virtual counterfeit to simulate the original PUF and predict the responses of random challenges. If the model correctly predicts the responses with high probability, the model can be used as a "software clone" of the PUF circuit. If the modeling attack is a concern, then the security of PUFs can be fortified by adding unpredictable nonlinearity to make the model building more difficult.[1] For example, in the feed-forward arbiter scheme depicted in Fig. 4, multiple challenge bits are determined by the racing result in intermediate stages instead of being provided by a user [10], [12]. Since the internal feed-forward bits are hidden to an adversary, he cannot build a precise model of a feed-forward arbiter PUF.

The other direction of improving the security of arbiter-based PUFs is reducing noise probability in PUF responses. For an adversary's software model to be successful, the response prediction must have less errors than the maximum tolerated noise in CRP measurements. Thus, by reducing the noise in PUF responses, we can increase the difficulty of the software model building attack. In [9], the reliability of PUF responses are improved by using "robust challenges" that generate consistent responses in the induced changes of environmental conditions.

III. Experiments

In this section, we present the experimental results of the primary characteristics of arbiter-based PUFs such as *inter-chip variation*, *environmental noise*, and *measurement noise*. Environmental noise has been measured over a practical range of environment variation. We also examine an *aging effect* that can potentially degrade identification capability after prolonged use. To evaluate the characteristics of arbiter-based PUFs, we generated 100 000 CRPs for each PUF in each environmental condition. In order to reduce measurement noise, the majority of 11 repeated measurements has been used as a response for each challenge. We performed the same experiments for feed-forward arbiter circuits, which were fabricated together with the regular arbiter-based PUFs.

A. Interchip Variation

Interchip variation of arbiter-based PUFs has been tested using 190 different PUF pairs. Fig. 5 shows the histogram of evaluated inter-chip variations. The average inter-chip variation is 23% and the minimum inter-chip variation is 17%. For the feedforward arbiter scheme, we obtained a histogram, as in Fig. 5, showing a shape similar to regular arbiter PUFs, where the average and minimum inter-chip variations increased to 38% and 28%, respectively. The increases are caused by the inter-chip variation of the racing result in the intermediate stages in the feed-forward arbiter scheme.

Since manufacturing variation consists of die-to-die, wafer-to-wafer, and lot-to-lot variations, inter-chip variation can possibly be dependent on die and wafer locations. We compared the inter-chip variation within a single wafer and across wafers. Fig. 6 shows the average, minimum, and maximum inter-chip variation within wafers 5 or 6 and across the two wafers. We can verify that the inter-chip variation across the wafers

[1]For PUFs where the response is obfuscated, this attack cannot be carried out successfully [11].

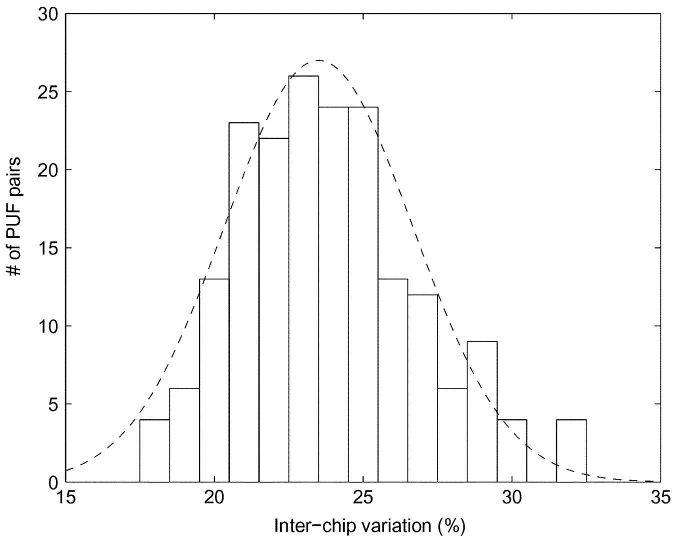

Fig. 5. Histogram of the inter-chip variation $y_{i,j}$ for 190 PUF pairs.

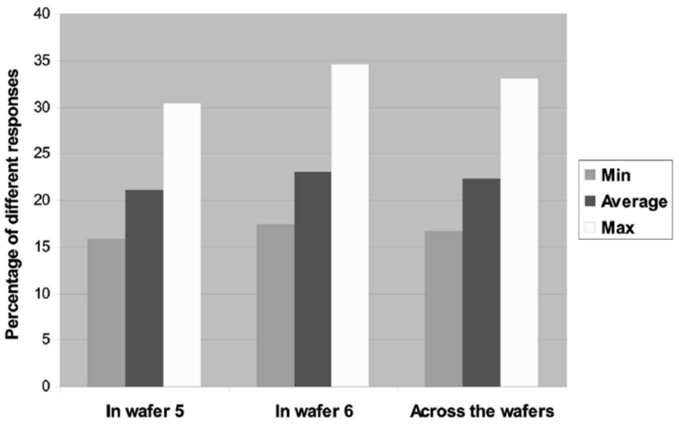

Fig. 6. Inter-chip variation of the PUFs from a single wafer and across wafers.

is similar to that within a single wafer. Thus, an amount of inter-chip variation is not strongly dependent on the location of dies on wafers.

In testing, more than 80% of total challenges generate a response variation in 20 PUF chips and bear identification information. Further analysis in [9] shows that these information-bearing challenges give less delay difference between top and bottom paths than the challenges that generate biased responses.

B. Noise in PUF Responses

Environmental variations such as temperature and power supply voltage variations are the primary causes of noise in PUF responses. Even without environmental variations, a setup time violation for an arbiter or the small fluctuation of junction temperatures and internal voltages can cause measurement noise.

Fig. 7 shows the amount of environmental noise introduced by temperature (μ_t) and voltage (μ_v) variations. The reference responses are measured at 27 °C and 1.8-V power supply voltage. Even if the temperature increases to 70 °C, $\mu_t \approx 4.82\%$. Also, with ±2% power supply voltage variation, $\mu_v \approx 3.74\%$. This shows that the differential structure of an arbiter-based PUF circuit reduces the environmental variations well below the average inter-chip variation (23%). The measurement noise (μ_m) is approximately 0.7%.

For the feed-forward arbiter scheme, we obtained the figure given in Fig. 7 that shows a similar shape with 9.84% maximum environmental noise and 4.5% measurement noise. Both of them are well below the

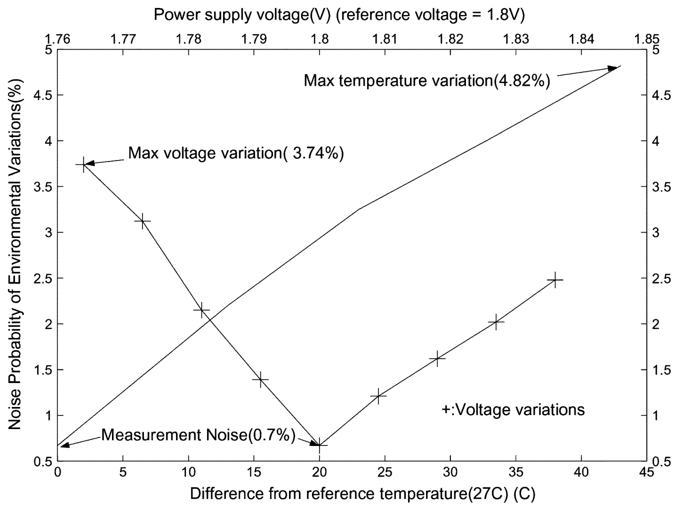

Fig. 7. Variation of PUF responses subjected to temperature and supply voltage changes.

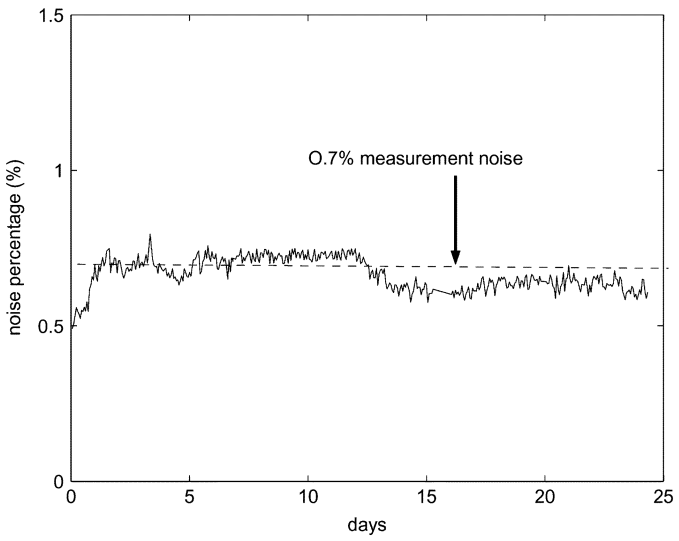

Fig. 8. Aging effect on an arbiter-based PUF for 25 days.

average inter-chip variation (38%) and, from the calculation of identification capability, by using 320 CRPs, 10^9 PUFs chips can be identified with less than 10^{-9} error probability (cf. Section IV-B). The noise probability has been increased in the feed-forward arbiter since there are seven internal arbiters, and the noise of each internal arbiter response can cause the noise in the final response.

Electromigration and hot-carrier effects cause the aging of wires and transistors in ICs. To consider the aging effect, Fig. 8 shows the result of a one-month-long aging test. We calculated the percentage of response bit differences from newly generated CRPs to the reference CRPs, which were generated at the start of the one-month period. The percentage varies slightly around $\mu_m \approx 0.7\%$. We conclude that any significant performance degradation has not been observed in this one-month aging test under normal operating conditions. However, a longer term aging test in more severe environmental fluctuation must be performed to guarantee the reliability of the PUF scheme in practice.

For commercial use of the PUF scheme, the test range of environmental conditions should be enlarged to prove the reliability of operation. For applications where a secret key is generated, we require substantially higher reliability. This can be achieved using error correction. A secure error-correction scheme for arbiter-based PUFs is described in [11].

C. Performance

For a given 64-b challenge, it takes 50 ns for an input rising edge to transmit across the 64-stage parameterized delay circuit and evaluate an output at an arbiter. It takes about 24.5 μs to generate 490 CRPs, which is sufficient to distinguish 10^9 arbiter-based PUFs (cf. Section IV). To extract a secret key of 160 b, we need 1800 CRPs [11], which takes 90 μs (we need more CRPs because the measurement noise needs to be corrected to extract a secret key; for identification, the measurement noise does not need to be corrected). This is sufficiently fast for most applications since a PUF is evaluated only infrequently to obtain a secret key. We can boost the performance by replicating multiple delay paths and arbiters to evaluate responses in parallel.

IV. Discussion

A. Modeling Arbiter-Based PUF Circuits

We number PUF ICs from 1 to n. For a fixed PUF i and fixed challenge c, the difference Δ between the delays of the top and bottom paths consists of three quantities: an average delay difference $\Delta(c)$, which is determined by the delay path configuration, the process variation $p_i(c)$, and the delay difference d, which is caused by measurement noise, the remaining uncompensated skew s_i, and modeling noise. When the challenge c is randomly chosen, the distribution of $\Delta(c)$ and $p_i(c)$ can be assumed to be Gaussian, $\Delta(c) \sim N(0, \sigma_\Delta)$, and $p_i(c) \sim N(0, \sigma_p)$, according to the Central Limit Theorem [13]. We model $d \sim N(s_i, \sigma_n)$. Hence, $\Delta = d + \Delta(c) + p_i(c) \sim N(s_i, \sigma)$, where $\sigma = \sqrt{\sigma_\Delta^2 + \sigma_p^2 + \sigma_n^2}$. The response corresponding to c measured by PUF i is determined by the sign of Δ.

In this delay model, the ratio $\sigma_P = \sigma_\Delta / \sigma_p$ is an important parameter to estimate the amount of process variation in a given process technology. In [9], σ_P is represented as a function of the probability p_i that a reference response of PUF i is equal to 1, the probability p_j that a reference response of PUF j is equal to 1, and the probability $p_{i,j}$ that both the reference responses of PUF i and PUF j corresponding to a same challenge are equal to 1. To simplify the calculation, we assume that $\Delta(c)$ and $p_i(c)$ are statistically independent. The maximum-likelihood estimation method [14] can be used to estimate σ_P from the experimental results of p_i, p_j, and $p_{i,j}$.[2] This method also leads to the confidence intervals with which the correctness of the model assumptions can be verified and the modeling noise can be computed.

To estimate σ_P, ten independent pairs of two different PUFs were used. For each pair, we generated 20 000 CRPs to estimate p_i, p_j, and $p_{i,j}$. Fig. 9 shows the estimates of σ_P in ten independent experiments and their confidence intervals. Notice that, on average, $\sigma_P \approx 1.75$. The delay variation by configuration changes is only 1.75 times larger than the delay variation caused by process variation. In this figure, for each confidence interval, the probability that σ_P is outside of the given confidence interval is $\epsilon = 0.35$. The figure shows that an overlap region exists between the confidence intervals, which means that σ_P has been estimated consistently in ten independent experiments.

We notice that σ_Δ can be calculated by simulating the circuit layout with extracted parasitic information of wires and transistors. Together with the estimated σ_P, we can measure the amount of process variation σ_p. Thus, we can implement the same arbiter circuit in different technologies to compare their process variation by estimating σ_P. Further statistical analysis can be performed to consider the correlation between $\Delta(c)$ and $p_i(c)$ in the calculation.

[2]The normalized skews s_i/σ, where $\sigma = \sqrt{\sigma_\Delta^2 + \sigma_p}$, have been analyzed in [9].

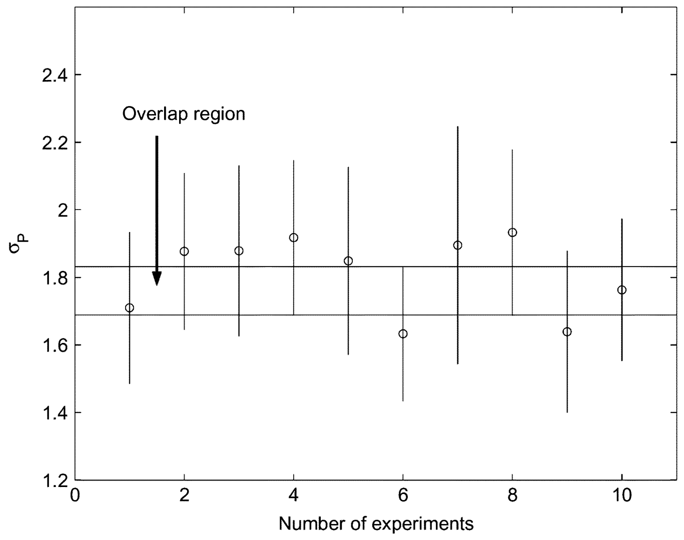

Fig. 9. Estimated σ_P and the confidence intervals in ten independent experiments.

B. Identification/Authentication Capability

Assume that a server has a database with CRP profiles of $N = 10^9$ PUFs. Suppose that a PUF wants to authenticate itself as Alice. Then the server asks the PUF to generate a list of CRPs corresponding to the CRP profile of Alice in the server's database. If the PUF is indeed Alice, then each generated response bit differs from the corresponding profile's response bit with probability μ. If the PUF is not who it claims to be, then this probability is equal to the inter-chip variation $\tau(>\mu)$.

Suppose that the server authenticates the PUF as Alice only if the number of response bit differences is less than or equal to some threshold t. This leads to two kinds of error probabilities: the false alarm rate (FAR) corresponds to Alice not being authenticated and recognized as Alice and the false detection rate (FDR) corresponds to a PUF different from Alice being authenticated as Alice.

The authentication capability is measured by FAR and FDR; the more CRPs used for authentication, the smaller these error probabilities will be. Let each CRP profile have k CRPs. Then

$$\mathrm{FAR} = \sum_{i=t+1}^{k} \binom{k}{i} \mu^i (1-\mu)^{k-i}$$

$$\mathrm{FDR} = \sum_{i=0}^{t} \binom{k}{i} \tau^i (1-\tau)^{k-i}.$$

We are interested in the minimal k (number of CRPs needed per authentication) such that there exists an appropriate t for which the average $(\mathrm{FAR}+\mathrm{FDR})/2 \leq 1/N$. For arbiter-based PUFs, we take $\tau = 0.230$ and $\mu = 0.0482$, which yield $k = 470$ and $t = 56$. For the feed-forward arbiter scheme, $\tau = 0.380$, $\mu = 0.098$, $k = 304$, and $t = 65$.

If a server wants to identify a PUF, then it asks the PUF to generate a list of CRPs which it compares to each of the CRP profiles in its database. The CRP profile with the least number of response bit differences identifies the PUF. The identification probability (IP) is the probability that the PUF is identified as the correct PUF with noise probability μ, which can be represented as follows:

$$\mathrm{IP} = \sum_{j=0}^{k} \binom{k}{j} \mu^j (1-\mu)^{k-j} (1 - \mathrm{FDR}\,|_{t=j})^{N-1}.$$

For arbiter-based PUFs, the minimal k for which $1 - \mathrm{IP} \leq 1/N$ is equal to 490. For the feed-forward arbiter scheme, $k = 320$.

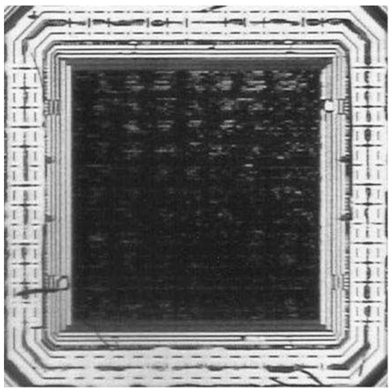

Fig. 10. Die photograph of the fabricated chip.

V. Conclusion

We proposed a candidate PUF implementation, fabricated it, and investigated its security and reliability. The test chip was built in TSMC's 0.18-μm, single-poly, six-level metal process with standard cells, whose die photograph is shown in Fig. 10. It contains eight sets of the arbiter circuit generating an 8-b response for a challenge and a JTAG-like serial interface and measures a total area of 1212 μm $\times$ 1212 μm. The maximum allowable frequency is 100 MHz, and each arbiter-based PUF circuit consumes 137 μW.

Experimental results indicate that there exists significant delay variation of wires and transistors across ICs implementing this circuit and that the idea of leveraging this variation to reliably identify and authenticate an IC is promising. This variation will be more significant in the technology node of 130 nm and beyond, and more inter-chip variation will give better identification capability of the PUF scheme. The PUF delay variation model can be utilized as a test structure to measure the amount of delay variation for a given technology.

References

[1] O. Kommerling and M. G. Kuhn, "Design principles for tamper-resistant smartcard processors," in *Proc. USENIX Workshop Smartcard Technology*, 1999, pp. 9–20.

[2] R. Anderson and M. Kuhn, "Tamper resistance—A cautionary note," in *Proc. 2nd USENIX Workshop Electronic Commerce*, Nov. 1996, pp. 1–11.

[3] O. Goldreich, S. Goldwasser, and S. Micali, "On the cryptographic applications of random functions," *Proc. Crypto Advanceds in Cryptology*, pp. 276–288, 1985.

[4] P. S. Ravikanth, "Physical one-way functions," Ph.D. dissertation, Dept. Media and Art Sci., Massachusetts Inst. Technol., Cambridge, 2001.

[5] R. Pappu, B. Recht, J. Taylor, and N. Gershen-Feld, "Physical one-way functions," *Science*, vol. 297, pp. 2026–2030, 2002.

[6] B. Gassend, "Physical random functions," M.S. thesis, Dept. Elect. Eng. Comput. Sci., Massachusetts Inst. Technol., Cambridge, Jan. 2003.

[7] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," in *Proc. Computer Communication Security Conf.*, Nov. 2002, pp. 148–160.

[8] D. Ranasinghe, D. Lim, S. Devadas, D. Janali, Z. Zhu, and P. H. Cole, "An integrable low cost hardware random number generator," *Proc. SPIE*, vol. 5649, pp. 627–639, 2004.

[9] D. Lim, "Extracting secret keys from integrated circuits," M.S. thesis, Dept. Elect. Eng. Comput. Sci., Massachusetts Inst. Technol., Cambridge, May 2004.

[10] B. Gassend, D. Lim, D. Clarke, M. van Dijk, and S. Devadas, "Identification and authentication of integrated circuits," *Concurrency Computat.: Practice Experience*, vol. 16, pp. 1077–1098, 2003.

[11] M. van Dijk, D. Lim, and S. Devadas. (2004, May) Reliable Secret Sharing with Physical Random Functions. [Online]. Available: http://csg.lcs.mit.edu/pubs/

[12] J. W. Lee, D. Lim, B. Gassend, E. G. Suh, M. van Dijk, and S. Devadas, "Building a secret key in integrated circuits for identification and authentication applications," in *Proc. Symp. VLSI Circuits*, 2004, pp. 176–179.

[13] J. A. Power, B. Donnellan, A. Mathewson, and W. A. Lane, "Relating statistical MOSFET model parameter variabilities to IC manufactuing process fluctuations enabling realistic worst case design," *IEEE Trans. Semicond. Manuf.*, vol. 7, no. 3, pp. 306–318, 1994.

[14] A. J. Dobson, *An Introduction to Generalized Linear Models*. London, U.K.: Chapman and Hall/CRC, 2002.

Estimation of FMAX and ISB in Microprocessors

Yossi Abulafia and Avner Kornfeld

***Abstract*—Inherent process device variations and fluctuations during manufacturing have a large impact on the microprocessor maximum clock frequency and total leakage power. These fluctuations have a statistical distribution that calls for usage of statistical methods for frequency and leakage analysis. This paper presents a simple technique for accurate estimation of product high-level (Full Chip) parameters such as the maximum frequency (FMAX) distribution and the total leakage (ISB). Moreover, this technique can grade critical paths by their failure probability and perform what-if analysis to estimate FMAX after fixing specific speed paths. Using our FMAX/ISB prediction, we show good correlation with silicon measurements from a production microprocessor.**

***Index Terms*—Critical path delay, die-to-die process variation, FMAX distribution, path-to-path correlation, spatial timing model (STM), static timing analysis (STA), statistical static timing analysis (SSTA), subthreshold leakage (ISB), systematic process variation, timing model (TM), within-die process fluctuation.**

I. Introduction

The microprocessor timing model (TM) represents the nominal timing characteristics of the path population on the die. Usually, the TM is built using static timing analysis (STA) or dynamic circuit simulation. It is clear that, without process parameter variation, FMAX is defined by the slowest path in the TM. However, this population is susceptible to process parameter variations that affect the physical parameters of paths, such as device channel length, threshold voltage, and interconnect resistance and capacitance, which change path nominal speed. Process variation classifications relate to die-to-die variation (σ_{d2d}), systematic variation ($\sigma_{\rm SYS}$), and random variation ($\sigma_{\rm RND}$). Since die-to-die variation affects every device on a chip equally, it will bias the whole TM, whereas local fluctuations, induced by systematic and random process variation, will affect TM path ordering. Traditionally, STA uses case analysis to model process variation. This enables deterministic STA, where best case, nominal, and worst case scenarious constructed the TM. This approach assumes that die-to-die variations are the main cause of FMAX distribution. However, following Moore's Law, poly-silicon gate lengths have decreased to the point where the ability to control critical device parameters on a single die has become increasingly difficult. This poses a major obstacle for deterministic STA and its ability to predict FMAX distribution, and gives rise to the need for statistical static timing analysis (SSTA) [6] approaches for FMAX estimation.

Previous studies on statistical circuit simulation [1], [2] concentrated primarily on path-level analysis, where deterministic timing analysis was performed on a set of worst/best cases that were scaled to system level to estimate FMAX distribution. A significant drawback of these methods is their pessimistic assumption of a zero path-to-path correlation. However, in practice, systematic process variation induces path-to-path correlation, which depends on the systematic process variation-length scale and the spatial distribution of paths. This path-to-path correlation has a significant impact on TM path ordering and FMAX distribution. Recent studies address this issue using *path-based* [3], [4], [7], [9], [10] or *block-based* [8] approaches, where *path-based* is a kind of statistical postprocessing, while *block-based* propagates statistical properties through the timing graph. However, neither method is as accurate as Monte Carlo simulation.

In this paper, we will describe a Monte Carlo-based method that will lay the groundwork for an SSTA estimation methodology for both frequency and leakage. Using this method, one can represent the spatial distribution of process fluctuation and combine it with a specific TM in order to calculate path-to-path correlations. Using this method, we will show how to estimate product parameters such as FMAX distribution and total leakage (ISB). Furthermore, we will present path-level analysis for determining critical path failure probability and for critical path what-if-analysis.

II. Process Variation Model

To model process variation, we assume that different correlation length ρ enables independent classification of process variations. These classifications relate to die-to-die variations (σ_{d2d}), systematic variations ($\sigma_{\rm SYS}$), and random variations ($\sigma_{\rm RND}$), whose probability distribution function (pdf) can be modeled by a normal distribution function $N(\mu, \sigma^2)$.

Die-to-die variation has a correlation length that is bigger than the die size and affects every device on a chip similarly. Hence, the die-to-die correlation length between any *gate i* and *gate j* is described by $\rho_{ij\_d2d} \equiv 1$. In that sense, die-to-die variation should be considered a bias variation.

Systematic variation exhibits a correlation length that is smaller than the die size, thereby introducing a high probability that devices that are close to each other will have similar device features. Usually, $\rho_{ij\_{\rm systematic}}$ is described by

$$\rho_{ij\_{\rm systematic}} = \exp\left(-\left(\frac{R_{ij}}{R_L}\right)^2\right) \tag{1}$$

where R_{ij} represents the distance between the elements and R_L is the process correlation distance. This induces spatial correlation, which is better described by the following relation between correlation length and the covariance between two elements:

$$\rho_{e1,e2} = \frac{{\rm COV}(e1, e2)}{\sigma_{e1}\sigma_{e2}}. \tag{2}$$

In the next sections, we will show that this spatial correlation has a significant impact on TM path ordering and FMAX and ISB distribution.

Random variations have a device scale correlation length where each device is affected differently and can be described by $\rho_{ij} \approx 1$ for $i = j$ and $\rho_{ij} \approx 0$ for $i =\neq j$.

III. Impact of Process Variation on the TM

To estimate the impact of process variation on the TM, one has to compute the path delay variation [5] and its correlation to other paths.

Manuscript received November 24, 2004; revised July 21, 2005.

The authors are with the Intel Corporation, Haifa 31015, Israel (e-mail: Yossi.abulafia@intel.com; avner.kornfeld@intel.com).

Digital Object Identifier 10.1109/TVLSI.2005.859469