Design for Eliminating Operation Specific Power Signatures from Digital Logic

Md Badruddoja Majumder, Md Sakib Hasan, Aysha Shanta, Mesbah Uddin and Garrett Rose

Great Lake Symposium on VLSI 2019 (GLSVLSI'19), May 9-11, 2019, Tysons Corner, VA, USA.

Citation Information (BibTex):

```
@INPROCEEDINGS{Majumder:2019,
author={Md Badruddoja Majumder and Md Sakib Hasan and Aysha Shanta and and Mesbah Uddin and
Garrett Rose},
title={Design for Eliminating Operation Specific Power Signatures from Digital Logic},
booktitle={Great Lakes Symposium on VLSI 2019 (GLSVLSI '19)},
month={May},
year={2019},
address={Tysons Corner, VA, USA}
}
```

Design for Eliminating Operation Specific Power Signatures from Digital Logic

Md Badruddoja Majumder, Md Sakib Hasan, Aysha Shanta, Mesbah Uddin, Garrett Rose
University of Tennessee, Knoxville
{mmajumde,mhasan4,ashanta1,muddin6,garose}@utk.edu

ABSTRACT

Conventional digital logic operations have distinguishable power signatures. Side channel power analysis combined with classification algorithm can reveal unknown logic operations. Revealing the underlying operations is the main task in reverse engineering an application. In this paper, we propose an unconventional way of overcoming this vulnerability by using chaos based reconfigurable logic operations. The chaos gate used in this paper is built from a simple 3 transistor chaotic oscillator capable of generating aperiodic states starting from a suitably chosen initial condition. We propose a design methodology using chaos gate to implement different logic operations with very similar power profiles. Therefore, it becomes significantly harder to distinguish logical operations built with chaotic logic gates in contrast to the conventional static CMOS logic gates. In addition, we show a mixed implementation of bitwise logic operations using different proportions of chaos and conventional gates resulting in a significant reduction of total overhead.

1 INTRODUCTION

Side channel power analysis is one of the major vulnerabilities in computing. Power profiles contain non-trivial information about computation and thereby pose a major threat to security. Power traces can be leveraged to determine processor instructions leading to software reverse engineering. Every operation in a digital system has its own power signature and can be distinguished from each other. It has been shown that side channel power analysis can be performed to reverse engineer machine code executed by a processor [10, 11, 13] .

Different mitigation schemes have been proposed against side channel power attack in different levels of abstraction starting from very low level hardware based approaches to high level code obfuscation [1, 11]. Two of the most common techniques in eliminating power signature from a computation are hiding and masking. Most of these approaches focus on eliminating data dependency of an operation from its power trace while the operation still can be recognized.

Earlier research pointed to chaos based logic design for power profile obfuscation [2, 12]. However, none of them was very specific about the type of vulnerabilities that it may help to resolve. A chaos based arithmetic logic unit (ALU) design was presented in [9] that can prevent power analysis based template attack to reverse engineer its instructions.

In this paper, we present a security aware logic design approach using a chaotic oscillator where each logic operation exhibits almost identical power traces to prevent power analysis assisted operation reverse engineering. In this design approach, a logic operation leaks minimal information about the type of operation it is performing since each operation exhibits very similar power signature. We perform machine learning based classification among the power traces of a set of logic operations implemented using the proposed method and a conventional method to compare their immunity to reverse engineering. We also perform a scalability analysis of the proposed design method up to 64-bit bitwise logic operations along with a mixed implementation technique using different proportions of chaos and conventional CMOS logic instances.

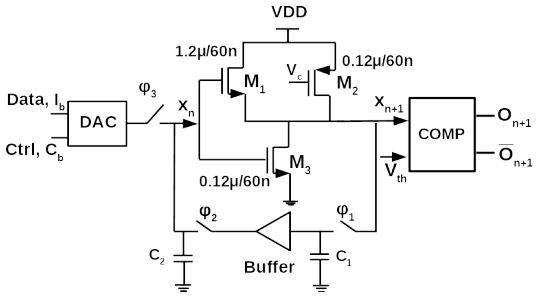

Figure 1: Logic gate built from a discrete time chaotic map circuit.

The remainder of the paper is organized as follows. Chaos based logic design technique is presented in Section 2. Section 3 describes the proposed security aware design methodology. Signal to Noise ratio (SNR) of the power traces is analyzed in Section 4 to quantitatively assess the similarity of power traces. Power analysis based classification results of both chaos and CMOS logic operations is presented in Section 5. Section 6 presents the mixed implementation method and its effectiveness against the attack. Section 7 discusses about the implementation overhead of the proposed chaos based logic. Finally, the paper is concluded in Section 8.

2 LOGIC USING CHAOS

Chaos computing refers to the idea of using nonlinear dynamics of chaotic systems for computation. Chaotic system produces aperiodic sequence over time dependent on the initial condition. Generation of a pattern from a chaotic system over time starting with an initial condition is called chaotic evolution. Evolution of a chaotic system may occur either in the continuous time domain or the discrete time domain. Chua's circuit is an example of continuous time domain chaotic system [3]. On the other hand, the logistic map, tent map and shift map are examples of the discrete time domain chaotic system [7]. In discrete time chaos generator, chaotic patterns are generated in an iterative process. In each iteration output of the previous iteration becomes the input of the present one and thus evolves in discrete time. Dudek *et al.* proposed a 3 transistor

Table 1: Evolution of chaotic output (V_C=0.68 V, $C_b = 1$).

X_0 (V)	X_{n+1} (V)				
	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$
0.171(00)	1.19	0.52	0.40	0.93	0.36
0.514(01)	0.41	0.88	0.34	1.15	0.49
0.857(10)	0.33	1.16	0.50	0.49	0.52
1.2(11)	0.52	0.40	0.94	0.37	1.06

non-linear circuit that works as a discrete time chaotic map [5]. This chaotic map circuit can be used to implement reconfigurable boolean functions [8].

In this work, we design a 65nm scaled version of the design presented in [8] for security aware chaos based logic design as shown in Fig.1. Similar to the basic functionality of any discrete time chaotic map, the output of the non-linear circuit is fed back to the input and iteratively generates a sequence of voltage. In addition to the non-linear map circuit, the chaos gate requires two sample-and-hold circuits controlled by the non-overlapping clocks, ϕ_1 and ϕ_2. When ϕ_1 is high, the output, X_{n+1} of the map circuit is sampled onto C_1. When ϕ_2 is high, X_{n+1} is sampled onto C_2 so that it becomes the input of the map circuit in the next iteration. A digital to analog converter is used to map the digital input, (C_b, I_b) to analog input, X_n of the chaos gate. Control bit, C_b is used for reconfigurability of the logic functions.

Initially, the circuit takes external input to start the operation which is controlled by the clock, ϕ_3. In subsequent iterations, the output is fed back to the input to continue chaotic operation. Bias voltage, V_c determines the voltage sequence of the chaotic evolution from this circuit. Output of the chaotic circuit is digitized using a comparator so that it works as a digital logic gate. Threshold voltage, V_{th} of the comparator also works as a control parameter for this logic gate. Thus, different boolean functionalities are generated at different iterations depending on 4 control parameters: bias voltage, V_c, control bit, C_b , iteration number, n and threshold voltage, V_{th}. The comparator used in this design is a dynamic comparator having both non-inverted and inverted output [6]. Any control parameter that produces a particular logic function also produces its inverted version. Therefore, AND/NAND, OR/NOR, XOR/XNOR, these functions are generated as pairs. Table 1 shows example analog output voltage sequences of the proposed design of chaotic gate for all possible input combinations with V_c=0.68 V and C_b=1. Chaotic circuit shown in Fig. 1 can work as 6 logic functions with appropriate threshold voltage, V_{th} and sampling iteration, n. which is demonstrated in Table 2. For instance, if we digitize the output analog voltage at 2^{nd} iteration from Table 1 with a threshold voltage of 0.66 V, we get the XOR/XNOR functionality. Similarly, other functionalities can also be verified using the information provided in Table 1 and 2.

3 DESIGN FOR SECURITY

The goal of this work is to construct logic operations exhibiting very similar power profiles such that minimal information is leaked through power side channel. This can be achieved by proper choice of parameters in the proposed chaos based logic design. Proposed logic design method provides a large design space to choose its

Table 2: Example chaos gate configuration for different operations.

Parameter	Instruction		
	AND/NAND	OR/NOR	XOR/XNOR
V_c (V)	0.68		
C_b	1		
V_{th}	0.46	0.41	0.66
n	2	5	2

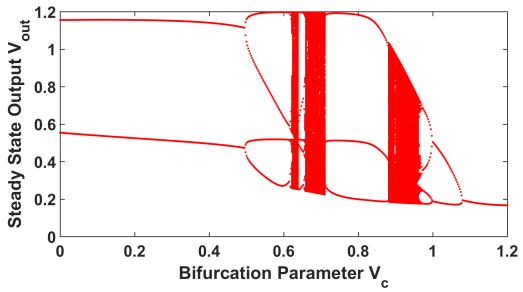

Figure 2: Bifurcation diagram for the 3 transistor chaotic map circuit.

design parameters due to chaos dynamics. Design space is reduced by imposing different design constraints led by security goals and performance criteria. In this section, we discuss these constraints and how to apply them to achieve our desired design for security.

3.1 Security Constraints

There are four parameters in the chaos circuit that we can vary to realize different logic functions. These parameters are: control bit, C_b, bias voltage, V_C, threshold voltage, V_{th}, and sampling iteration, n. In this chaos circuit, C_b and V_c, respectively controls the initial input and the sequence of chaotic output which effectively dictates the power trace. For the type of security desired in this paper, power profiles of all logic operations need to be alike. In order to have maximum similarity of the power profiles among different operations, we keep C_b and V_c same for all functions with varying V_{th} and n. Due to differences in V_{th}, the comparator has some operation specific signatures in the power profile. However, this variation is negligible to distinguish the operations based on their power profile which will be presented in later sections of the paper. We use a delay stage at the comparator output to make the final output available at the same time which helps in hiding the time induced power signature of all the operations.

3.2 Performance Constraints

Bifurcation diagram of a chaotic map circuit plots the possible outputs with respect to a bifurcation parameter. Fig. 2 is the bifurcation diagram of the chaotic oscillator shown in Fig. 1 where the bifurcation parameter is V_c. Chaotic operation occurs in a region in the bifurcation diagram where the outputs are densely populated within a certain range. The geometry of the circuit is chosen based on an empirical analysis as analytical approach is very complex and

Table 3: Performance metric for different geometry of chaotic oscillator.

Index	Geometry			Metric
	$W_1(\mu m)$	$W_2(\mu m)$	$W_3(\mu m)$	$\frac{mV}{fJ.\mu m^2}$
1	4.8	0.48	0.48	0.85
2	4	0.15	0.15	5.1
3	4	0.12	0.12	7.1
4	6	0.15	0.15	3.5
5	2	0.15	0.15	4.7
6	1.2	0.12	0.12	7.2
7	2.4	0.24	0.24	2.1

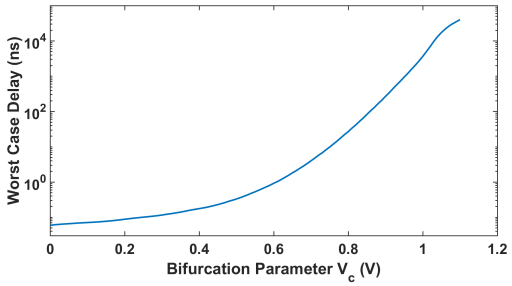

Figure 3: Illustration of worst case delay vs. bifurcation parameter.

out of scope of this paper. We choose 7 different test geometries for the circuit and rank each of them based on a performance metric. The performance metric (PM) is formulated as

$$PM = \frac{C^{w1}}{A^{w2}.P^{w3}.D^{w4}} \quad (1)$$

where C=width of chaotic region in bifurcation diagram and A, P, D are the average area, power and delay of the chaotic map circuit, respectively. $w1$, $w2$, $w3$ and $w4$ are corresponding weight for these factors which are considered 1 in this work, i.e. $w_1 = w_2 = w_3 = w_4 = 1$. However, depending on applications we may modulate these weights to prioritize particular factors in the performance metric. As width of chaotic region in the bifurcation diagram contributes to the large functionality space of the chaotic gate, it is kept as the numerator in the performance metric. On the other hand, quantities contributing to design overhead such as area, power and delay are kept in the denominator. Table 3 shows the performance metric for each of the 7 geometries. Geometry 6 which has the highest performance value is chosen in the final design used in this work. W_1 , W_2, W_3 represents the width of transistors M_1, M_2 and M_3 in Fig. 1. Length of the transistors are chosen as $60nm$ which is the minimum length of the $65nm$ process.

For the chosen geometry, there are two distinct regions of V_c that causes chaotic operation as can be seen in the bifurcation diagram. The first one is 0.62 V to 0.71 V and the second region is 0.88 V to 0.96 V. Primarily, V_c is chosen from one of these two regions for having the maximum functionality space of chaotic operations. Investigating the circuit, it is found that the delay is maximum

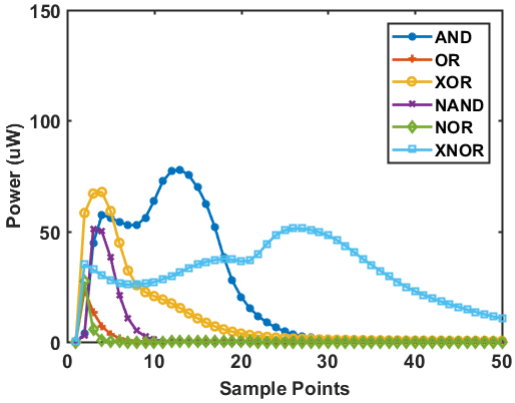

Figure 4: Power trace of CMOS logic operations.

when the input makes a transition from VDD to 0. This worst case delay for chaotic gate increases with V_c as shown in Fig. 3. The first region of V_c results in considerably less delay than the second one. Therefore, we constrain the V_c to the first region only.

Chaotic operations are sensitive to initial condition. Small perturbation can cause significant deviation in its evolution over time and amount of deviation increases with time. However, we can avoid this problem by restricting the iteration of the chaotic circuit within a certain limit that the deviation in the chaotic evolution is negligible and the circuit can function reliably. Furthermore, longer iteration time results in longer delay. In this design, we consider 5 iterations to choose the logic functionality from the chaotic circuit.

3.3 Design flow

In the first step of our design flow, we generate a set of possible configuration parameters for each of the logic functions individually. While generating this set of configuration parameters, we keep the iteration number, n within 5. After that, a subset of this parameter set is considered based on common V_C and C_b which is our security constraint described earlier. We further reduce the set based on the worst case delay criteria mentioned earlier where V_C is constrained between 0.62 V and 0.71 V. Such a combination of the parameters is shown in Table 2 for the proposed design of logic gates. It is to be noted that this particular combination is arbitrarily chosen and is one of many which satisfies all the constraints.

Fig. 4 and 5 shows the power profile of different operations for the same operand implemented using CMOS and chaos logic, respectively. It can be seen from visual inspection that power traces in CMOS logic are different from each other while they are almost indistinguishable in the proposed chaos logic. In order to compare the degree of difficulty in distinguishing the power trace of different logic operations, quantitative analysis such as signal to noise ratio (SNR) and machine learning based classification are performed in the following sections.

4 SNR ANALYSIS

In this work, we are interested in measuring the distinguishability of the power traces corresponding to different logic operations. The

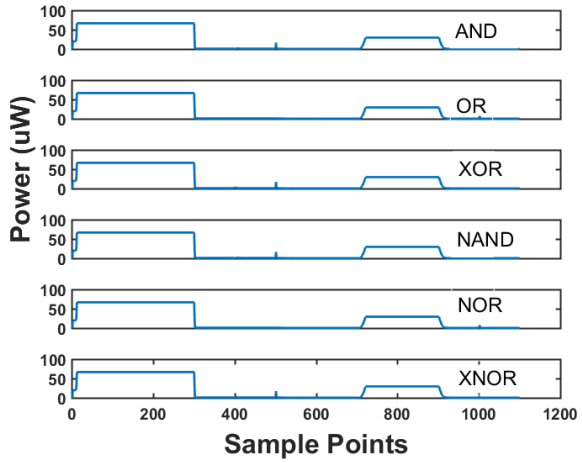

Figure 5: Power trace of chaos logic operations.

more the power traces are scattered among different operations, the more distinguishable they are. On the contrary, they are less distinguishable if the power traces are scattered more within operations due to different operands. In order to measure the SNR in this work, inter- and intra-operation scattering, S_{inter} and S_{intra} are considered as the signal and noise, respectively.

Since unnecessary dimensions affect distinguishability, we estimate the signal to noise ratio for the power traces with reduced dimensionality using linear discriminant analysis [4]. In this method, original data is projected to a lower dimensional space using a linear transformation. Let X be a set comprising C (number of classes) subset of D, N-dimensional power traces. X is transformed to Y where each power trace has a reduced dimensionality N_r. For Fisher's linear discrimant analysis, $N_r = C - 1$.

$$Y_i = X_i W, \tag{2}$$

where W is a N x N_r transformation matrix.

We can calculate inter-class and intra-class scatter matrix, S_{inter} and S_{intra}, respectively for the original N-dimensional power traces.

Intra-operation scatter matrix for each class is as follows.

$$S_{intra}(X_i) = \sum_{x \epsilon C_i} (x - m_i)^t (x - m_i), \tag{3}$$

where m_i is the N-dimensional sample mean for power traces of each class. Overall intra-operation scatter matrix is then

$$S_{intra}(X) = \sum_{i=1}^{C} S_{intra}(X_i). \tag{4}$$

Inter-scattering matrix can be defined as,

$$S_{inter}(X) = \sum_{i=1}^{C} D(m_i - m)^t (m_i - m). \tag{5}$$

where, m is the sample mean across all classes. Now, inter- and intra-scattering matrix can be found for the data with reduced dimension.

$$S_{intra}(Y) = W^T S_{intra}(X) W. \tag{6}$$

$$S_{inter}(Y) = W^T S_{inter}(X) W. \tag{7}$$

Criterion function for maximum distinguishability can be defined as,

$$J(W) = \frac{det(S_{inter}(Y))}{det(S_{intra}(Y))}. \tag{8}$$

According to our definition, the best case SNR of the power traces from different logic operations is

$$SNR = J(W_{max}) = \frac{det(W_{max}^T S_{inter}(X) W_{max})}{det(W_{max}^T S_{intra}(X) W_{max})}. \tag{9}$$

For the analysis of SNR, we simulate each logic operation both for the conventional CMOS design and chaos based design in Cadence Spectre circuit simulator. A power trace for a particular logic operation depends on its input data transition. There are 16 possible input data transitions for a 2 input logic operation covering all cases. We consider 6 logic operations for this work. Each operation has 16 different power traces. Dimension for power trace is 100 and 1100 for CMOS and chaos gates, respectively. More sampling points are required for chaos gate since it requires larger operation time. Using Eq. 9, SNR in the power traces for CMOS and chaos logic operation is found to be $1.265x10^{-5}$ and $3.384x10^{-11}$, respectively. Power traces of chaos logic is significantly harder to distinguish as its SNR is 6 orders smaller than that of CMOS logic.

5 CLASSIFICATION ATTACK AND RESULTS

We demonstrate power analysis based classification attack on a set of logical operations AND, OR, XOR, NAND, NOR, XNOR implemented using conventional CMOS logic gates and proposed chaos based logic gates. Based on the results of the attack, we can compare the resiliency of both implementation against power analysis.

5.1 Attack Method

The attack performed to distinguish each operation from a set of logic operations uses a classification algorithm. The classifier is trained using power traces from known operations. In the work flow of the attack method, we collect training power traces, reduce the dimension of data and finally apply the classification algorithm on the test data set. These steps are explained here.

5.1.1 Data Collection. We perform classification on bitwise logic operations of different bit sizes. In this work, bitwise logic operations of arbitrary operand size is implemented by using multiple 2-input single bit instances of the desired functionality. For example, an N-bit AND operation uses N copy of single bit AND gate. If instantaneous power profiles of the single bit gate is known for each possible input transitions, we can estimate the power profile of an arbitrary sized operation by adding up the profile of each logic instance for respective input transition. For this work, instantaneous power profiles of each of the six 2-input single bit logic gate are pre collected for all 16 possible input transitions using Cadence Spectre simulator for both CMOS and chaos based implementations.

5.1.2 Dimensionality Reduction. It is a common practice to reduce the dimensionality of data in order to make the classification more efficient in terms of computation and at the same time avoid the noise created by irrelevant points. From [10], it was found that

Table 4: Power analysis based classification accuracy of logic operations.

Method	Logic	Classification Accuracy(%)						
		AND	OR	XOR	NAND	NOR	XNOR	**Overall**
(K=1)-NN	CMOS	98	97	99	97	93	99	**97**
	Chaos	23	38	24	24	37	24	**28**
(K=3)-NN	CMOS	98	96	98	97	91	99	**96**
	Chaos	23	33	22	22	32	23	**26**
(K=)5-NN	CMOS	97	95	98	97	89	99	**96**
	Chaos	21	32	20	20	31	22	**24**

principle component analysis (PCA) gives the best result for classifying instructions in a microcontroller using power traces and that is why PCA has been used in this work. In PCA, original data is projected orthogonally into a subspace with lower dimension. The projection subspace is chosen mathematically in such a way that maximizes the variance among projected data. Here,the first 30 principal components have been used since they contain more than 95% of the total variance.

5.1.3 Classification Algorithm. There are various algorithms that can be used for pattern classification such as K-nearest neighbor (KNN), multivariate Gaussian, support vector machine, etc. From the previous work on instruction classification presented in [10], KNN was found to be the most efficient in classifying different operations performed by a microcontroller. We, therefore, choose KNN with K=1, 3 and 5 as the classification algorithm to be used in this work. In KNN algorithm, the classifier calculates the distance between the test sample and all training samples and choose the K training samples having the least distance. Among these K samples, the class that most of these samples belong to is treated as the class of the test sample. Distance between two samples has been calculated using Euclidean distance function in this work.

5.2 Attack Results

In order to test the recognition rate of the classifier, we used 60% of the total collected data for training and the remaining 40% data is used for testing the classifier. The set of training and testing samples are shuffled among the total collected sample space to cross validate the classifier. We choose a total data set of 4000, 8000, 16000 and 32000 samples for performing the classification attack on 8 bit, 16 bit, 32 bit and 64 bit operations, respectively. Classification accuracy for both conventional and chaos based implementation of 8 bit logic operations are tabulated in Table 4. For the conventional implementation of considered logical operations, overall accuracy of classification based on power profile is found to be 97%, 96%, 96% for nearest neighbor approach with K=1, 3 and 5, respectively while it is only 28%, 26% and 24% for the proposed chaos based design. Such reduction in classification accuracy reflects that proposed chaos logic operations are designed in a way that they carry minimal distinguishing features from each other in the power profiles which is consistent with the result of SNR analysis in section 4.

6 SCALABILITY AND MIXED IMPLEMENTATION

Logical operations in general purpose computing systems are performed bitwise on different operand size as machine instructions.

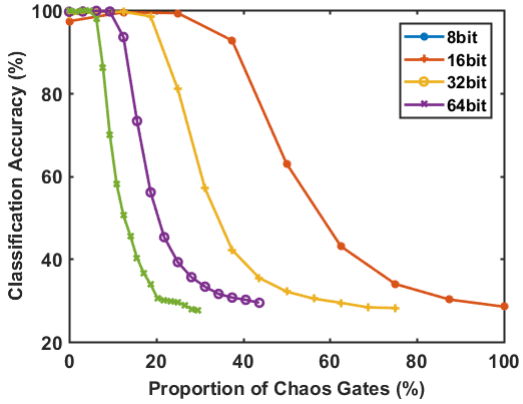

Figure 6: Classification accuracy of bitwise logic operations for different operand sizes and different proportions of chaos gates.

As we described earlier, we can implement the operation of arbitrary bit size using multiple copies of the 1 bit instance. Using the simulated power data, we can perform the classification attack on any arbitrary bit size. In order to demonstrate the scalability of our design technique while preserving its immunity against the classification attack, we choose 8 bit, 16 bit, 32 bit and 64 bit logic operations.

An interesting aspect of the security with chaos based logic is that for bitwise operation in larger system, all single bit instances do not require to be implemented using chaos logic. Rather, a mixed implementation where only a fraction of total instances implemented using the proposed chaos logic is sufficient to defend against side channel attack. This significantly reduces the overhead of the entire system. In order to demonstrate this, we perform the classification attack on several cases of mixed implementations for 8 bit, 16 bit, 32 bit and 64 bit logical operations. For example, for 8 bit logic operations, we consider all cases such as CMOS only, chaos only and other possible proportions of CMOS and chaos logic instances. For this analysis, we only consider nearest neighbor with K=1 using PCA as it can classify with higher accuracy.

The results of mixed implementations are demonstrated in Fig. 6. Here, number of chaos gate=0 refers to the CMOS only implementation. It can be seen that in the CMOS only implementation, classification accuracy is close to 100%. With the increase of chaos gate proportion, signature of CMOS instances in the power profile diminishes and the classification accuracy drops. At a certain proportion of CMOS and chaos gates, the classification accuracy reaches around 30% which is a significant improvement over the traditional design. The effectiveness of this mixed implementation technique lies in the fact that the implementation cost is reduced significantly in the design of proposed side channel resistant logic operations in larger computing systems.

7 OVERHEAD ANALYSIS

In conventional design of digital logic, operation performed by a logic gate can be revealed through the power measurements. Proposed logic design using chaotic dynamics prevents information

Table 5: Overhead estimates for chaotic logic in gate level and system level. 32 bit and 64 bit systems are considered for the system level overhead estimates.

Method	Power-Delay product (PDP)				Area			
	value(fJ)	OV_{gate}	OV_{sys}		value(μm^2)	OV_{gate}	OV_{sys}	
			32b	64b			32b	64b
CMOS	54	-	-	-	0.29	-	-	-
Chaos	95	1.8x	1.2x	1.1x	1.22	4x	1.9x	1.6x

leak about the operation of a logic gate through the power trace and helps mitigate these vulnerabilities. Design for security comes with a cost of implementation overhead as compared to conventional design technique. Here, we compare the overheads of our proposed design with conventional static CMOS logic design. Comparisons are made based on the average of area and power-delay product (PDP) of different types of logic gates. Overhead estimates are given in Table 5. Proposed chaos based logic requires approximately 1.8x PDP and 4x area per gate as compared to standard CMOS logic gate. CMOS gates considered here for comparison are designed in 65 nm process where the basic gates (NAND, NOR) are equivalent to an inverter having an aspect ratio of 480nm/60nm and 240nm/60nm for the PMOS and NMOS, respectively.

Due to the mixed implementation capability discussed earlier in this paper, proposed chaos logic can be more efficient while implemented in larger computing system as bitwise logical instructions. In a mixed implementation, bitwise logic operations are built with a proportion of chaos and standard CMOS gates and hence, the overhead can be reduced. If k is the proportion of chaos gates for the bitwise implementation of a n bit system, implementation overhead can be calculated as

$$OV_{sys} = \frac{(1-k)G_D + kG_C}{G_D} \tag{10}$$

where G_D and G_C are the implementation cost associated with each standard CMOS gate and chaos gate, respectively. Eq. 10 can be represented in terms of per gate overhead of chaos gate compared to the digital gates.

$$OV_{sys} = 1 + k(OV_{gate} - 1) \tag{11}$$

where $OV_{gate} = \frac{G_C}{G_D}$.

Area and PDP overhead for bitwise chaos logic for 32 and 64 bit system is tabulated in Table 5. From Fig. 6, proportion factor, k of chaos gates in 32 bit and 64 bit logic operations are calculated as 0.3 and 0.2, respectively considering a maximum classification accuracy of 30%. Now, the mixed implementations of bitwise logic in 32 bit and 64 bit system exhibit a PDP overhead of 1.2x and 1.1x, respectively and area overhead of 1.9x and 1.6x, respectively.

8 CONCLUSION

Chaos based logic design shows great promises for application in security against side channel power analysis. A logical instruction set having very similar power profiles can be designed according to the proposed method which helps mitigate power analysis based instruction reverse engineering. Mixed implementation overheads associated with bitwise logic operations scale down significantly with the operand size and thus outweighs the per gate overhead. This paper mainly focused on application of the proposed design in hiding the logic operations from the power profile. Since this design methodology can be implemented using any discrete time chaotic map circuit, other circuit implementations can be explored in the future to reduce the overhead.

9 ACKNOWLEDGEMENT

This work was supported by the Air Force Office of Scientific Research under Award FA9550-16-1-0301.

REFERENCES

[1] Jude Angelo Ambrose, Roshan G Ragel, and Sri Parameswaran. 2007. RIJID: random code injection to mask power analysis based side channel attacks. In *Proceedings of the 44th annual Design Automation Conference*. ACM, 489–492.
[2] James Bohl, Lok-Kwong Yan, and Garrett S Rose. 2015. A two-dimensional chaotic logic gate for improved computer security. In *Circuits and Systems (MWSCAS), 2015 IEEE 58th International Midwest Symposium on*. IEEE, 1–4.
[3] Leon O Chua and G-N Lin. 1990. Canonical realization of Chua's circuit family. *IEEE transactions on Circuits and Systems* 37, 7 (1990), 885–902.
[4] Richard O Duda, Peter E Hart, and David G Stork. 2000. Pattern classification (pt. 1). *Danvers, MA: Wiley Interscience* (2000).
[5] P Dudek and VD Juncu. 2003. Compact discrete-time chaos generator circuit. *Electronics Letters* 39, 20 (2003), 1431–1432.
[6] Bernhard Goll and Horst Zimmermann. 2009. A comparator with reduced delay time in 65-nm CMOS for supply voltages down to 0.65 V. *IEEE Transactions on Circuits and Systems II: Express Briefs* 56, 11 (2009), 810–814.
[7] Behnam Kia, John Lindner, William L Ditto, et al. 2015. Nonlinear dynamics based digital logic and circuits. *Frontiers in computational neuroscience* 9 (2015), 49.
[8] Behnam Kia, John F Lindner, and William L Ditto. 2016. A simple nonlinear circuit contains an infinite number of functions. *IEEE Transactions on Circuits and Systems II: Express Briefs* 63, 10 (2016), 944–948.
[9] Md Badruddoja Majumder, Md Sakib Hasan, Mesbah Uddin, and Garrett S Rose. 2018. Chaos computing for mitigating side channel attack. In *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 143–146.
[10] Mehari Msgna, Konstantinos Markantonakis, and Keith Mayes. 2014. Precise instruction-level side channel profiling of embedded processors. In *International Conference on Information Security Practice and Experience*. Springer, 129–143.
[11] Thomas Popp, Stefan Mangard, and Elisabeth Oswald. 2007. Power analysis attacks and countermeasures. *IEEE Design & test of Computers* 24, 6 (2007).
[12] Garrett Steven Rose. 2014. A chaos-based arithmetic logic unit and implications for obfuscation. In *VLSI (ISVLSI), 2014 IEEE Computer Society Annual Symposium on*. IEEE, 54–58.
[13] Dennis Vermoen, Marc Witteman, and Georgi Gaydadjiev. 2007. Reverse engineering java card applets using power analysis. *Information Security Theory and Practices. Smart Cards, Mobile and Ubiquitous Computing Systems* (2007), 138–149.