Modeling Power Efficiency of S-boxes Using Machine Learning

Rajat Sadhukhan, Nilanjan Datta, and Debdeep Mukhopadhyay

***Abstract*—In the era of lightweight cryptography, designing cryptographically good and power efficient 4x4 S-boxes is a challenging problem. While the optimal cryptographic properties are easy to determine, verifying the power efficiency of an S-box is non-trivial. The conventional approach of determining the power consumption using commercially available CAD-tools is highly time consuming, which becomes formidable while dealing with a large pool of S-boxes. This mandates development of an automation that should quickly characterize the power efficiency from the Boolean function representation of an S-box. In this paper, we present a supervised machine learning assisted automated framework to resolve the problem for 4x4 S-boxes, which turns out to be 14 times faster than traditional approach. The key idea is to extrapolate the knowledge of literal counts, AND-OR-NOT gate counts in SOP form of the underlying Boolean functions to predict the dynamic power efficiency. The experimental results and performance of our novel technique depicts its superiority with high efficiency and low time overhead. We demonstrate effectiveness of our framework by reporting a set of power efficient optimal S-boxes from a large set of S-boxes. We also develop a deterministic model using results obtained from supervised learning to predict the dynamic power of an S-box that can be used in an evolutionary algorithm to generate cryptographically strong and low power S-boxes.**

***Index Terms*—Power Efficiency, Optimal S-box, Dynamic power, Machine Learning**

I. Introduction

The advent of Internet-of-Things(IoT) era have resulted in wide shift of spectrum of devices from desktops and servers to embedded-systems, RFIDs and sensor-networks having huge resource constraints. The heavy resource constraints on these end devices make it impossible to run conventional cryptographic algorithms, which lead to the development of lightweight cryptographic algorithms and primitives. Recently, this development of lightweight cryptographic primitives has gained its momentum with the announcement of lightweight cryptographic project by NIST [1]. As S-boxes are the basic building blocks (used to provide the non-linearity) for designing block ciphers, designing cryptographically good and power efficient S-boxes is a widely discussed problem.

A series of research work is going on to make S-boxes power efficient using suitable architecture. In [2], Satoh et al. proposed a low power multi-stage PPRM-based S-box architecture. In [3], Bertoni *et al.* presented DSE-based (Decoder-Switch-Encoder) architecture for low power S-box design. This was followed by the work [4], where the authors showed use of ANF representation of Boolean function to achieve an optimized pipelining arrangement and short critical-path, low power S-box architecture. In [5], Trichina et al. introduced a side-channel resistant AES co-processor optimized for low power that uses an S-box architecture introduced in [6]. In all of the above mentioned papers, the main focus is given on the implementation architecture.

Another direction of research is to fix an architecture and then report the S-boxes which are power efficient. This approach was used in [7] where the authors aimed to report cryptographically good, power efficient S-boxes (LUT based architecture) from a large search space. They used a heuristic based approach to find optimal 4×4 S-boxes and then for each of the S-boxes they verified the power efficiency. As the algorithm runs over a large space, the time required to determine whether a S-box is power efficient or not, plays a crucial role.

A. Determining Power Efficiency of S-boxes: The Traditional Approach

In this subsection we discuss two generic methods that are used for determining the power efficiency of S-boxes. The first one is a simulation based approached and the second one being a probabilistic one. A brief description of both the approaches are given below. The design flow is shown in Fig.1

Simulation Based Technique. In this technique the average power dissipation of the circuit is obtained by recording the signal events over time, followed by tabulation and averaging of event data. The flow diagram of this methodology has been shown in Fig.1. This approach involves huge computation resources and is time consuming. This approach was used in [7] to report power efficient S-boxes from a large search space. A look-up-table (LUT) based S-box design is first synthesized using a technology library through a commercial synthesis tool to form gate-level-netlist and delay file. The gate-level-netlist along with delay model and test-bench file is used to generate a switching activity file containing details of toggle count of every signal. This can be done via commercial simulation tools. The activity file also contains information about the time attributes of every node which specify time durations for every nodes and signals at various levels. The test bench file contains all possible combinations of signal transitions to be given at input. The gate-level-netlist along with this generated switching activity file is used to determine the exact power using any commercial power estimation tool.

Probabilistic Approach. [8] In probability-based power estimation method a signal is viewed to be a random vari-

Rajat Sadhukhan (e-mail:rajat.sadhukhan@iitkgp.ac.in), Nilanjan Datta (e-mail:nilanjan_isi_jrf@yahoo.com) and Debdeep Mukhopadhyay (e-mail:debdeep.mukhopadhyay@gmail.com) are with the Dept. Computer Science and Engineering, Indian Institute of Technology, Kharagpur, India

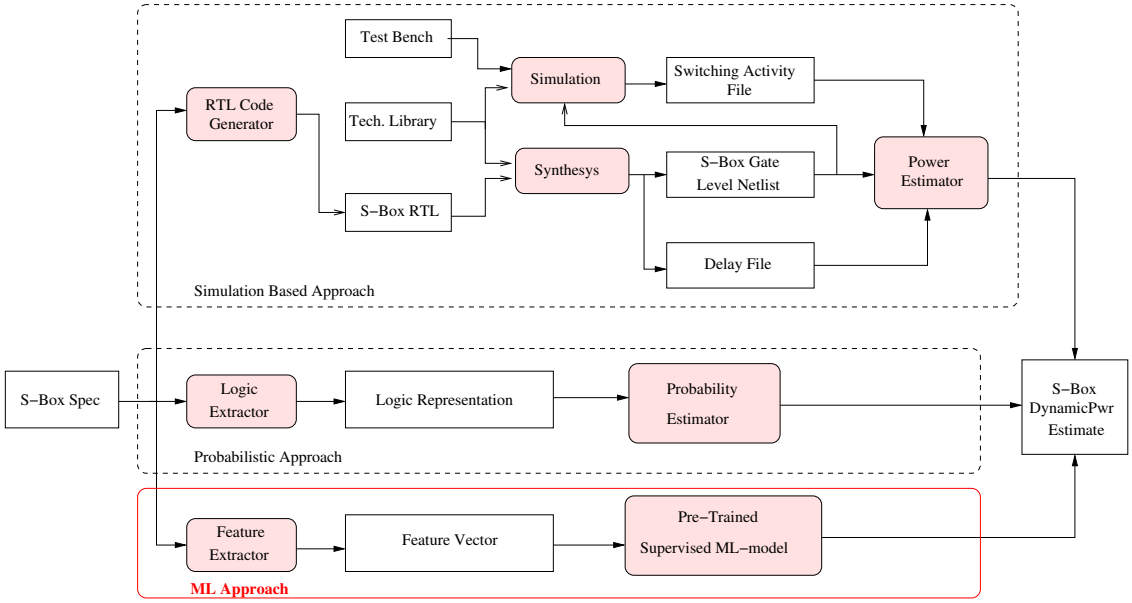


Fig. 1. Design Flow

able asserted with some statistical features (static probability, transition density, etc). This technique uses a simplified delay model (also called zero-delay model where same delay is assumed for all the gates) for power estimation unlike simulation-based approach where the exact delay is measured. Moreover, a static probability (usually 0.5) is assigned to each input signals, which may not be accurate. In this approach, the S-box specification is converted into a logic form (using signal probability or BDD model) using a logic extractor. Major issue with signal probability model is that it cannot handle toggle power as well as spatial and temporal co-relations (due to presence of *reconvergent fan-out* at various circuit nodes). BDD takes care of these limitations, but at the cost of speed. In BDD, switching activity is calculated for each ordering of the input variables. With the increase in size of inputs the problem becomes intractable.

B. Our Contribution

Central to this work is power efficiency of S-boxes. We have already seen in the previous subsection that existing approaches to determine power efficiency have issues with accuracy or speed. So, we primarily aim to devise a completely new methodology that determines the power efficiency of a set of S-boxes that reduces the time overhead and maintains good accuracy. Our contributions are two folded:

- In this paper, we present a supervised machine learning assisted automated framework to classify a set of $n \times n$ S-boxes into two classes (good and bad) based on their power efficiency. We mainly focus on the LUT based implementation of S-boxes with AND, OR, INVERTER (in short A-O-I) gates. We have observed that the switching activity of the component functions of an S-box is mainly dependent upon the literal counts (in SOP, factor and kernel extracted forms) and AND-OR-NOT gate counts of the underlying Boolean functions (also known as the component functions) corresponding to the S-box. However, it is hard to mathematically formulate such a relation. This motivates us to choose machine learning based approach to predict the dynamic power efficiency of a set of S-boxes as it is dependent on the switching activity of it's component functions.

- We demonstrate the effectiveness of our framework by reporting a set of power efficient S-boxes from a large set of 4×4 optimal. The experimental results shows that our tool is approximately 84% accurate for both the classifiers. In terms of speed, our algorithm turns out to be at least 14 times faster than the simulation based approach in determining power efficiency by commercially available CAD-tools.

- We have also developed a deterministic model where we try to mathematically formulate the correlation between the actual power values and the feature values reported from SIS. The effectiveness of this mathematical formulation to predict the actual power of an S-box is verified by another fresh pool of cryptographically good S-boxes. The result depicts that this model predicts the power with a very low mean relative error of 9.65%.

C. Significance of the Work

To the best of our knowledge, this is the first machine learning based approach that predicts whether an S-box is power efficient or not. Our algorithm requires roughly 0.874 seconds to determine power efficiency of an S-box, in an Intel Xeon machine, operating at 2 Ghz processor speed. We have also used the CAD-tool Synopsys *Design Compiler* to compute the power of an S-box, which takes roughly 11.843 seconds in the same machine. This depicts that our algorithm reduces the time overhead by a factor of around 14 times. This overhead

becomes formidable when we consider a very large set of S-boxes. For example, suppose we have a large set of 2^{20} optimal 4×4 S-boxes for which we want to report the power efficient S-boxes. In this case, our algorithm will take around 10.6 days where as the CAD tool will require 148.5 days i.e. close to almost 5 months time.

D. Organization

The remainder of the paper is organized as follows. In Sect. II, we provide a basic overview and cryptographic properties of S-boxes, machine learning models with performance metrics. In Sect. III we describe our machine learning based framework to classify set of S-boxes depending on the power efficiency. We describe all the features that we used in feature set for modeling. In Sect. IV we provide experimental results showing the efficiency and high performance of our tool for a set of 4×4 optimal S-boxes. Next we propose a deterministic model using the correlation between the actual power values and the feature values reported from SIS in Sect. V. Finally, we conclude in Sect. VI with interesting open problems.

II. Preliminaries

A. S-box Representation and Properties

In the standard cryptographic nomenclature, an $n \times n$ substitution box (abbreviated as S-box) is a nonlinear function from n bit to n bit: $S : F_2^n \rightarrow F_2^n$. It can be represented as a set of n vectorial Boolean functions $(s_1, \ldots, s_n)$, where $\forall i \leq n,\ s_i : F_2^n \rightarrow F_2$ are the component functions. Here, we briefly describe some important cryptographic properties of S-boxes.

Balancedness. We call an S-box S to be balanced if it takes every values of F_2^n same number of times.

Nonlinearity. The nonlinearity of a Boolean function $s : F_2^n \rightarrow F_2$ is defined as the minimum distance of the function from the set of all affine functions. Extending the idea, we define the nonlinearity of $n \times n$ S-box as the minimum of all the distances between the set of linear combinations of component functions of S to the set of all affine functions. More formally, nonlinearity of an $n \times n$-function S equals the minimum nonlinearity of all its component functions $v \cdot S$, where $v \in {F^*}_2^n$ [9]:

$$NL_S = 2^{n-1} - \frac{1}{2} \max_{\substack{a \in F_2^n \\ v \in {F_2^n}^*}} |W_S(a, v)|,$$

where

$$W_S(a, v) = \sum_{x \in F_2^n} (-1)^{v \cdot S(x) + a \cdot x},\ a, v \in F_2^n,$$

is the Walsh-Hadamard transform of the function S and $a \cdot b$ is the usual inner product of $a, b \in F_2^n$ that equals $a \cdot b = \bigoplus_{i=1}^{n} a_i b_i$.

Differential Uniformity. Let S be a function from F_2^n into F_2^m with $a \in F_2^n$ and $b \in F_2^m$. We define the *difference distribution table* of S with respect to a and b as:

$$\Delta_S(a, b) = \{x \in F_2^n : S(x) \oplus S(x \oplus a) = b\}.$$

The entry at position (a, b) corresponds to the cardinality of the difference distribution table $\Delta_S(a, b)$ and is denoted as $\delta_S(a, b)$. The *differential uniformity* δ_F is then defined as [10]:

$$\delta_S = \max_{\substack{a \in {F_2^n}^* \\ b \in F_2^m}} \delta_S(a, b).$$

Definition 1: A 4×4 S-box is said to be cryptographically optimal if it is balanced, has nonlinearity equal to 4, and differential uniformity equal to 4 [11].

We are mainly interested in optimal S-boxes as these S-boxes are good in terms of resisting linear and differential attacks. While designing block ciphers, another possible choices of S-boxes are the involutive ones.

B. Sources of Power Dissipation in CMOS

The sources of power dissipation in digital CMOS can be broadly classified based on their dependence on circuit topology. While *static power*, *leakage power* and *short-circuit power* are completely dependent on CMOS-technology and independent of circuit topology, *dynamic power* has a dependence on the structure of the circuit. In this paper we mainly concentrate on the dynamic power of CMOS circuits. The transition rate of a circuit node is not equal to the transition rate of clock. Statistically, the average *dynamic power* of a node in a circuit is given by the following equation [12]:

$$P_{\mathsf{dyn}} = \alpha_{0 \rightarrow 1} \cdot f_{\mathsf{clk}} \cdot C_L \cdot V_{\mathsf{dd}}^2,$$

where, V_{dd} is the supply voltage, C_L is the node capacitance, f_{clk} is the clock frequency and $\alpha_{0->1}$ is the *node transition activity factor* of the node. Combining gate transitions of every internal nodes, the total *dynamic power* of the circuit is given by the following expression:

$$P_{\mathsf{dyn}}^{\mathsf{total}} = \sum_{i=1}^{N} (\alpha_i \cdot C_i) \cdot f_{\mathsf{clk}} \cdot V_{\mathsf{dd}}^2,$$

where N is the total number of nodes in the circuit.

C. Machine Learning and It's Performance Metrics

There are two types of supervised machine learning frameworks, namely *classification* and *regression*. In case of regression, a continuous value is predicted while in case of classification a class label is predicted. The quality of prediction by an ML is evaluated by some widely known metrics. In this subsection, we briefly revisit the popular metrics used to evaluate ML based models. First we consider the confusion matrix, one of the most intuitive and easiest metrics used for finding the correctness and accuracy of classification based models. It is a 2×2 matrix (with dimension "actual" and "predicted") having the following entries:

- True Positives (TP). When the actual data and predicted data both belongs to 1 (True).
- False Positives (FP). In this case the actual class of the data point was 0 (False), however the prediction is 1 (True).
- False Negatives (FN). Here the actual class of the data point was 1 (True) while the prediction is 0 (False).

- True Negatives (TN). In this case both the actual class and predicted class of the data point is 0 (False).

Now we describe the performance metrics that we will consider to evaluate our algorithm.

Accuracy. Accuracy in classification problems is the number of correct predictions made by the model over all kinds predictions made. Formally, it is defined as

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}.$$

Note that, accuracy is a good measure only when the target variable classes in the data are nearly balanced.

F1 Score. Two important measures for classification based models are precision and recall. Precision is a measure that tells us what proportion of the S-boxes that we predicted as energy efficient, actually power efficient. On the other hand recall is a measure that tells us what proportion of S-boxes that actually are power efficient is detected by the algorithm as being power efficient. These two measures are clubbed together to have a single metric called F1 score, which basically is the harmonic mean of precision and recall. Formally,

$$\text{precision} = \frac{TP}{TP + FP}, \qquad \text{recall} = \frac{TP}{TP + FN}.$$

$$\text{F1 score} = \frac{2 \times \text{ precision} \times \text{ recall}}{(\text{precision} + \text{recall})}.$$

Note that the use of harmonic mean ensures that if one number is really small between precision and recall, the F1 score becomes more closer to the smaller number than the bigger one, giving the model an appropriate score rather than just an arithmetic mean.

ROC Plot. In statistics, a receiver operating characteristic curve, i.e. ROC curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The area under curve of an ROC curve represents the quality of a classifier, which ranges between 0 to 1 with higher values representing a better classifier.

III. Proposed Methodology

In this section we provide complete details of our machine learning based framework to evaluate and classify S-boxes based on their dynamic power consumption. As mentioned already, by dynamic power we basically mean dynamic power consumption for the AOI implementation (LUT based) of an S-box.

A. Machine Learning Model

We have followed supervised learning approach to construct a binary classifier for our problem. Given a predefined threshold power P_{th}, the classification is done as follows:

$$S \in \text{Class 0 (bad)}, \quad \text{if } P_{\mathsf{dyn}}(S) > P_{\mathsf{th}}$$

$$S \in \text{Class 1 (good)}, \quad \text{if } P_{\mathsf{dyn}}(S) \leq P_{\mathsf{th}}.$$

We have mapped our problem to classification rather than regression as we are interested to report whether a set of S-boxes are power efficient or not, rather than reporting the exact dynamic power of an S-box. By aligning our problem to classification problem we are apparently exploiting a large search space.

Let, $\mathsf{Perm}(2^n)$ be the set of all permutations of $\{0, \cdots, 2^n - 1\}$ and $\Sigma_n \subset \mathsf{Perm}(2^n)$ be a set of S-boxes (permutations) having some properties P. To ensure good cryptographic properties, a typical choice for P could be optimal meaning that the S-boxes must be balance, highly nonlinear and have low differential uniformity. We begin with a training vector T_{train}, extract all the desirable features (listed in the next subsection) for each S-boxes corresponding to the training vector using the module Extract_Feature. Then we compute the power class of each S-box using the module Find_Class and train the ML classifier by feeding these features and power class. Now to predict the power classes corresponding to a vector of S-boxes, termed as test vector T_{test}, we extract the features of each S-boxes and use the already trained classifier. The formal algorithm corresponding our framework is presented in Algorithm 1.

Algorithm 1: ML-Framework for Classifying S-boxes

Input : $\Sigma_n \subset \mathsf{Perm}(2^n)$, a set of S-boxes having property P.
Output: Classification vector E_{test} of S-boxes corresponding to the test vector T_{test}

Construct $T_{\mathsf{train}} \subset \Sigma_n$ and $T_{\mathsf{test}} = \Sigma_n \backslash T_{\mathsf{train}}$;
Let $T_{\mathsf{train}} = (S_1, \ldots, S_t)$ and $T_{\mathsf{test}} = (S_{t+1}, \ldots, S_{|\Sigma_n|})$
for $i = 1;\ i \leq t;\ i = i + 1$ **do**
    $F_{S_i} = \mathsf{Extract\_Feature}(S_i)$;
    $E_{S_i} = \mathsf{Find\_Class}(S_i)$;
end
/* Train Model */
C_{train} =TrainModel($(F_{S_1}, \ldots, F_{S_t}),\ (E_{S_1}, \ldots, E_{S_t})$)
/* Classify*/
for $i = t + 1;\ i \leq |\Sigma_n|;\ i = i + 1$ **do**
    $F_{S_i} = \mathsf{Extract\_Feature}(S_i)$;
    $E_{S_i} = \mathsf{Predict\_Class}(F_{S_i}, C_{\mathsf{train}})$;
end
$E_{\mathsf{test}} = (E_{t+1}, \ldots, E_{|\Sigma_n|})$;
return E_{test};
}

B. Feature Set Considerations

A feature set is a set of measurable attributes or characteristics that is used by the model for classification. The feature extractor function Extract_Feature returns values corresponding each of features for a given S-box. In our model, we have considered the following features representing a S-box:

Feature 1: lc(SOP). This feature denotes the literal count in simplified Sum-of-Products(SOP) representation of the Boolean functions corresponding to the S-box. Recall that a

literal is a Boolean variable in its complement or normal form. The switching capacitance of a circuit has a direct correlation with its literal count [13] and hence its dynamic power.

FEATURE 2: lc(factor). This represents the literal count of the SOP when expressed as parenthesized algebraic expression. The power consumption of a circuit in terms of its literal count is given by [13] $\Sigma n_i \alpha_i$, where i^{th} literal is occurring n_i times with switching activity α_i. So, an algebraic factorization of the expression helps in reducing the number of literals, and hence the overall switching activity.

FEATURE 3: gc(kernel). This feature represents the gate count of kernel-extracted representation. A *cube* in a Boolean function is a product of one or more number literals. We call a Boolean expression *cube-free* if no cube can divide the expression evenly. A Boolean expression s can be expressed as $P \cdot Q + R$, where P, Q and R are Boolean expressions, Q being the quotient and R is the remainder. A *cube-free* expression is called *kernel* of F. Factoring out *kernel* from the expression reduces the total number of nodes in the overall circuit, which in actual reduces the total switching activity of the circuit [14]. The main advantage of extracting out *kernel* from a set of Boolean expression lies in the fact that one single copy of the *kernel* circuit is implemented, which results in significant reduction in the number of nodes in the circuit. Let $Q = f(l_1, l_2, \ldots, l_m)$ be an extracted *kernel* of a set of functions $s_1, \ldots, s_i$ having $I_1, I_2, \ldots, I_n$ as internal nodes. So, the output of driver gates of $l_1, l_2, \ldots, l_m$ drives $i-1$ lesser gates, which reduces the switching capacitance and hence the overall dynamic power with a power saving given by [15]: $(i-1).(\sum_{j=1}^{m} \alpha_{lj}.n_{lj} + \sum_{j=1}^{n} \alpha_{Ij}.n_{Ij})$, where n_x and α_x are the number of gates driven by node x and switching activity of node x.

FEATURE 4: gc(AND). This represents the number of AND gate count in the kernel extracted representation of an S-box.

FEATURE 5: gc(OR). This feature denotes the number of OR gate count in the kernel-extracted representation.

FEATURE 6: gc(NOT). This represents the number of NOT gate used in the kernel-extracted representation.

Now we will take an example of a 4×4 S-box to demonstrate the above mentioned features.

A CONCRETE EXAMPLE. Let S be a 4×4 optimal S-box given in . Corresponding component functions (s_1, s_2, s_3, s_4) of S are given as follows:

$s_1(x_1, x_2, x_3, x_4) = \overline{x_1} \cdot x_2 \cdot x_3 + x_1 \cdot x_2 \cdot \overline{x_3} + x_1 \cdot \overline{x_2} \cdot x_3 + x_1 \cdot x_4$
$s_2(x_1, x_2, x_3, x_4) = \overline{x_1} \cdot x_2 \cdot \overline{x_3} + x_1 \cdot \overline{x_2} \cdot \overline{x_3} + x_1 \cdot \overline{x_2} \cdot x_3 + x_1 \cdot \overline{x_4}$
$s_3(x_1, x_2, x_3, x_4) = \overline{x_1} \cdot \overline{x_3} \cdot \overline{x_4} + \overline{x_1} \cdot \overline{x_2} \cdot x_3 + \overline{x_1} \cdot x_2 \cdot x_4 + x_3 \cdot x_4$
$s_4(x_1, x_2, x_3, x_4) = x_1 \cdot x_2 \cdot \overline{x_4} + \overline{x_2} \cdot x_4 + x_1 \cdot \overline{x_3}.$

Now in factor form this can be expressed as:

$s_1(x_1, x_2, x_3, x_4) = \overline{x_1} \cdot x_2 \cdot x_3 + x_1 \cdot (x_2 \cdot \overline{x_3} + \overline{x_2} \cdot x_3 + x_4)$
$s_2(x_1, x_2, x_3, x_4) = \overline{x_1} \cdot x_2 \cdot \overline{x_3} + x_1 \cdot (\overline{x_2} \cdot \overline{x_3} + \overline{x_2} \cdot x_3 + \overline{x_4})$
$s_3(x_1, x_2, x_3, x_4) = \overline{x_1} \cdot \overline{x_3} \cdot \overline{x_4} + \overline{x_1} \cdot \overline{x_2} \cdot x_3 + x_4 \cdot (\overline{x_1} \cdot x_2 + x_3)$
$s_4(x_1, x_2, x_3, x_4) = x_1 \cdot (x_2 \cdot \overline{x_4} + \overline{x_3}) + \overline{x_2} \cdot x_4.$

From the above two representations, it is easy to see that the literal counts in SOP form and factor form are 40 and 34 respectively.

Now we consider the kernel-extracted representation of the Boolean functions. In our example, the maximum common *kernel* is $Q = x_1 \cdot \overline{x}_2 + \overline{x_1} \cdot x_2$ and the kernel extracted Boolean function is given by:

$s_1(x_1, x_2, x_3, x_4) = x_3 \cdot Q + x_1(x_2 \cdot \overline{x_3} + x_4)$
$s_2(x_1, x_2, x_3, x_4) = \overline{x_3} \cdot Q + x_1(x_2 \cdot x_3 + \overline{x_4})$
$s_3(x_1, x_2, x_3, x_4) = \overline{x_1} \cdot \overline{x_3} \cdot \overline{x_4} + \overline{x_1} \cdot \overline{x_2} \cdot x_3 + x_4 \cdot (\overline{x_1} \cdot x_2 + x_3)$
$s_4(x_1, x_2, x_3, x_4) = x_1 \cdot (x_2 \cdot \overline{x_4} + \overline{x_3}) + \overline{x_2} \cdot x_4$

It is easy to see that in this representation total number of two input AND, OR, NOT gates are 31, 17 and 10 respectively. The kernel representation has less number of gates and hence lesser switching activity as shown in Fig.2(b), where gate Q is used only once as compared with Fig.2(a).

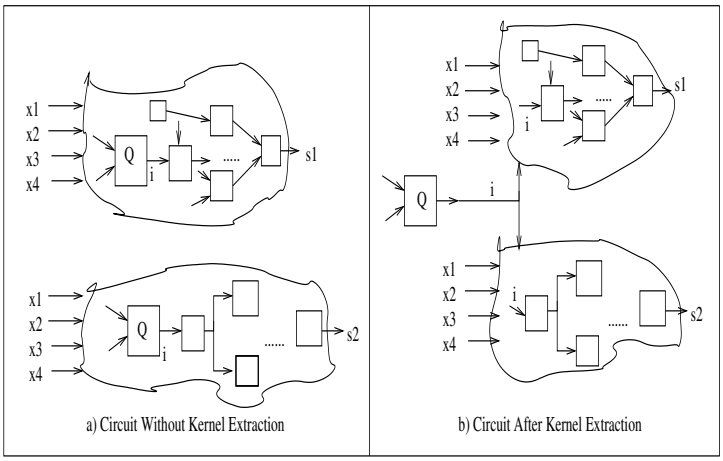


Fig. 2. Circuit of s1 and s2 with and without Kernel Extraction

C. Model Selection

In this subsection, we briefly describe the two models that we have chosen for the machine learning and justify the selection.

SUPPORT VECTOR MACHINE (SVM). SVM [16] is a traditional and very popular machine learning model. In this model, one or more hyperplanes are constructed in a multi-dimensional space based on which classification is done. Construction of the hyperplanes are done by maximizing it's distance from the nearest data point on either side of the plane. These distances are known as support vectors, which are maximized to obtain the best accuracy. This model provides very good accuracy if a linear separation is obtained. This model uses a subset of support vectors in decision making process, which makes it fast and memory efficient.

RANDOM FOREST (RF). Random Forest [17] is a type of *ensemble* method where multiple decision trees are combined to form an effective and very powerful model. A single decision tree has tendency to overfit on the training data, where the model gets perfectly trained by the training data, but unable to generalize on new or test dataset. In RF, a label is predicted by taking majority vote from all decision trees used to construct the model. As this model combines decisions from multiple decision tree, chances of over-fitting of data becomes negligible, increasing the accuracy of the model. This model runs very effectively on very large number of data points.

IV. Application to 4 X 4 Optimal S-box

In this section we apply our tool to report all the power efficient S-boxes from a set of 4×4 optimal S-boxes. We first describe the complete set-up, briefly discuss tuning parameters corresponding to both the classifiers used and finally show the high performance and efficiency of the tool.

A. Setup

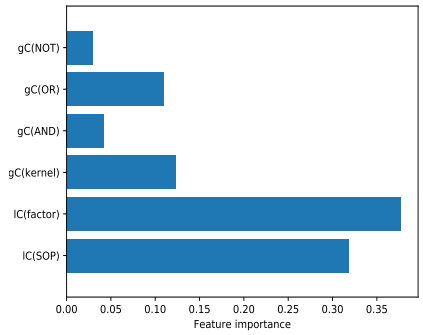
In order to evaluate our proposed methodology we started with a list of 10000 cryptographically strong S-boxes having differential uniformity and non-linearity of 4 obtained using genetic algorithm. We used tournament method of size 3 as selection method for our genetic algorithm. To extract features of every S-box we used *Sequential Interactive System(SIS)* [18] version 1.3 and *Espresso* [19] version 2.3. We have synthesized every S-box using Synopsys Design Compiler version J-2014.09-SP1 to report their dynamic power. In the synthesis process we have forced the synthesis tool to use standard cell library ($180nm$) consisting of only 2-input AND, OR, and NOT gate and used TSL18FS120 cell library from Tower Semiconductor Ltd. The standard cell library is characterized using Silicon Smart Software (Version: 2008.02-SP1p1) characterized under Fast-Fast process(P), 1.98V voltage(V) and -40 degree C temperature(T). Finally, to build our ML model and data analysis we used *Scikit-Learn* [20] ML tool version v0.19. Before feeding our data into the ML-tool we *profiled* the data such that it contains equal number of good and bad S-boxes based on some pre-defined threshold dynamic power. In our case, we have chosen $130\mu W$ power value as threshold. So our dataset contains 5000 S-boxes having dynamic power value less than $130\mu W$ (called as good S-boxes) and 5000 S-boxes having power more than $130\mu W$ (called as bad S-boxes).

B. Performance

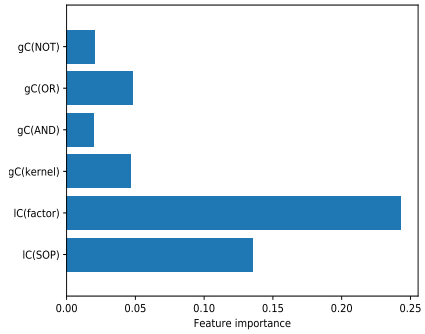
In this subsection we show the effectiveness of our tool in classifying the power efficiency of the S-boxes. For that we measure the classification accuracy, F1 score and area of ROC curves corresponding to both RF and SVM classifier. The training and validation sets are chosen randomly from a set of 10000 labeled samples, with their sizes in ratio $7:3$. The sample set consists of 5000 power efficient and 5000 power in-efficient S-boxes in order to achieve an unbiased training.

Feature Importance Assessment. We have shown the results of the feature importance assessment in Fig. 3. The Y-axis represents the features (as mentioned in) and the X-axis represents importance of each feature scaled to an interval of [0,1]. It is interesting to observe that all the features are required (importance value for each of them is greater than 0) during the classification.

Learning Curves. The learning curve conventionally depicts improvement in performance as we increase the number of training examples. Learning curves corresponding to our experiment using RF and SVM are depicted in Fig. 4. The red line signifies the training score while the green line signifies cross-validation score.



(a) Feature Importance Graph for RF Model



(b) Feature Importance Graph for SVM Model

Fig. 3. Feature Importance Graph for 4×4 Optimal S-boxes

Accuracy and F1 Score. The accuracy and F1 score obtained in our experiment is shown in Table I. The results depict that our classifier models perform reasonably well to predict the power efficiency of a set of S-boxes.

Classifier	TN	FP	FN	TP	Accuracy	F1 Score
RF	1296	197	274	1233	84.3%	0.846
SVM	1310	203	296	1191	83.4%	0.840

TABLE I
Performance Result for 4×4 Optimal S-boxes

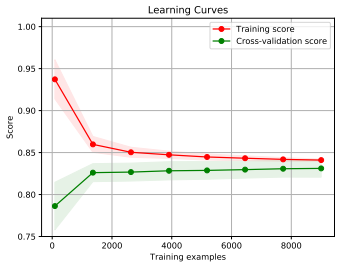
ROC Area under the Curve. We have also provided the ROC curves for both the classifier. From Fig 5, we see that the area under curve (AUC) for RF and SVM are 0.92 and 0.89 respectively, showing goodness of the classifier.

In order to verify the robustness of the learning, we ran each of our experiments several times. The average accuracy, F1 score and area of ROC curves for RF is 84.2%, 0.841 and 0.92 respectively. For SVM the values are 83.1%, 0.836 and 0.89 respectively.

C. Efficiency

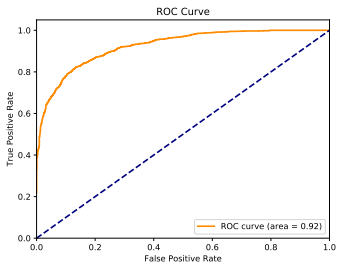
Our experiment takes 43.7 minutes of running time to predict the power classes of 3000 S-boxes. So, on an average our algorithm requires roughly 0.874 seconds to predict the power class for each S-boxes. On the other hand, we have run the CAD-tool Synopsys *Design Compiler* to compute

(a) Learning curves for RF Model

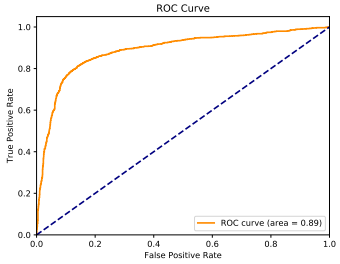


(b) Learning curves for SVM Model

Fig. 4. Learning Curves for 4×4 Optimal S-boxes



(a) ROC Curve for RF Model



(b) ROC Curve for SVM Model

Fig. 5. ROC plot for 4×4 Optimal S-boxes

the power of an S-box. To evaluate each S-box, the Design Compiler takes approximately 11.843 seconds of running time in a machine with Intel Xeon at @ 2 Ghz processor speed. This shows that our algorithm reduces the time overhead by a factor of around 14 times.

V. Towards A Deterministic Model to Predict Power

In this section first we develop a relation between the actual power values and the feature values reported from SIS, and then using correlation we will verify the nature of relationship and develop a deterministic model connecting the the two.

A. Relation between Predicted Dynamic Power and Feature Values

As pointed out earlier that power consumption of a circuit in terms of its literal count is given by [13] $\Sigma n_i \alpha_i$, where i^{th} literal is occurring n_i times with switching activity α_i and kernel extraction reduces the switching capacitance and hence the overall dynamic power with a power saving given by [15]: $(i-1).(\sum_{j=1}^{m} \alpha_{lj}.n_{lj} + \sum_{j=1}^{n} \alpha_{Ij}.n_{Ij})$, where n_x and α_x are the number of gates driven by node x and switching activity of node x. Taking this relation into account, we develop the following relation between predicted power value X and the feature values as:

$$X = \alpha_N * \mathsf{gc(NOT)} + \alpha_A * \mathsf{gc(AND)} + \alpha_O * \mathsf{gc(OR)} + \alpha_F * \mathsf{lc(FACTOR)},$$

where α_N, α_A, α_O are the switching activity corresponding to the NOT, AND, OR gates respectively. By definition, we have $\alpha_N = 0.5$, $\alpha_A = 0.25$ and $\alpha_O = 0.75$. We use α_F to denote the switching activity of a literal and we assume it to be equally likely with 0.5 probability.

B. Correlation between Predicted and Reported Dynamic Power

In this section we will study the correlation between the reported dynamic power and predicted power values obtained using the equation introduced in sectionV-A. We begin with $4\,000$ S-boxes and continue to increase the number until the correlation becomes static. As shown in Fig.6, the correlation become static around $16\,000$ to $20\,000$. So, we consider a set of $20\,000$ cryptographically strong S-boxes and the result of the correlation graph is shown in Fig.7. We observed a constant

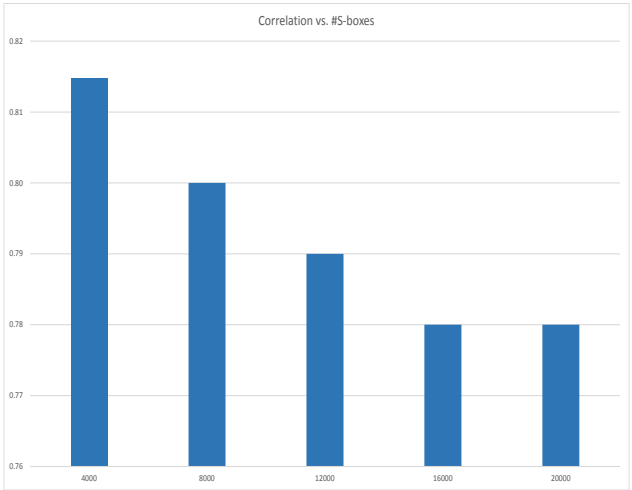


Fig. 6. Correlation vs. Number of S-boxes

correlation of 0.78 from $16\,000$ S-boxes. So, with the data of $20\,000$ S-boxes we plotted the reported dynamic power from $Design\ Compiler$ along the Y-axis versus the predicted dynamic power along the X-axis as shown in Fig.6. We tried to fit a linear model as a first approximation to describe this correlation as: $Y = b * X + a$, where X is predicted dynamic power, b and a are the slope and Y-intercept of the line respectively. The values of a and b are given by:

$$\begin{aligned} b &= \mathsf{Cov}(X,Y)/\ \mathsf{Var}(X) \\ a &= \mathsf{mean}(Y) - b * \mathsf{mean}(X). \end{aligned}$$

From Fig.6 the value of a and b is calculated as 23.3 and 2.7 respectively.

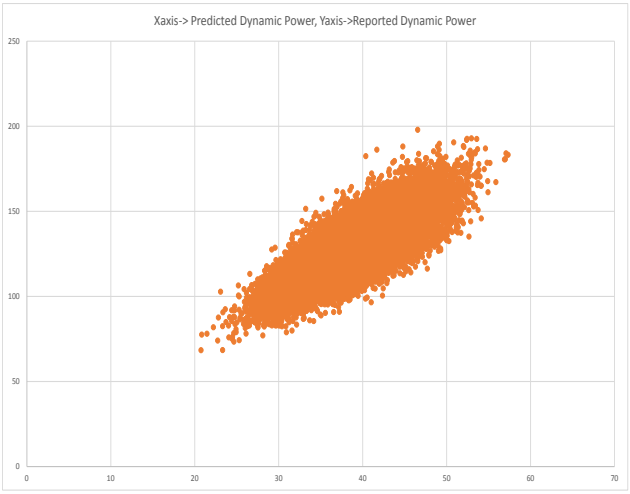


Fig. 7. Reported Dynamic Power(Design Compiler) vs. Predicted Dynamic Power

C. Model Verification

To verify the relation obtained in section V-B, we considered a $10\,000$ different set of S-boxes comprising of both Optimal as well as Involutive Optimal S-boxes, and we observed a mean error of 9.65% as shown in Fig.8. The red line in Fig.8 is the predicted values of $10\,000$ S-boxes according to the equation equation $Y = 2.7 * X + 23.3$, while the blue cluster are their actual values obtained from CAD-tool. In Fig.8 the orange dots and blue dots shows predicted dynamic power and actual dynamic power respectively for each S-boxes. It is also clear from the figure that orange and blue band are in sync with each other showing a linear correlation between them.

VI. Conclusion and Future Works

In this paper, we have presented a supervised machine learning assisted automated framework to report the power efficiency of cryptographically strong S-boxes. We have validated the effectiveness of our framework by reporting a set of power efficient S-boxes from a large set of 4×4 optimal S-boxes. The experimental results shows that for 4×4 S-boxes, our methodology is around 84% accurate and approximately 14 times faster (using AND-OR-NOT gates) than the traditional approach. We have also extended this result to mathematically formulate the correlation between the actual power values and the feature values reported from SIS. Our experiment shows that the above formulation can predict the power with high accuracy and can be very useful in an evolutionary algorithm to generate cryptographically good S-boxes with low power. Extension of this machine learning based methodology for predicting the dynamic power efficiency of larger S-boxes such as 8×8 S-boxes seems to be an intriguing future direction.

References

[1] K. A. McKay, L. Bassham, M. S. Turan, and N. Mouha, "Report on Lightweight Cryptography," 2017, http://nvlpubs.nist.gov/nistpubs/ir/2017/NIST.IR.8114.pdf.

[2] S. Morioka and A. Satoh, "An optimized s-box circuit architecture for low power aes design," in *Cryptographic Hardware and Embedded Systems - CHES 2002*.

[3] G. Bertoni, M. Macchetti, L. Negri, and P. Fragneto, "Power-efficient asic synthesis of cryptographic sboxes," in *Proceedings of the 14th ACM Great Lakes Symposium on VLSI*.

[4] M. M. Wong and M. L. D. Wong, "A high throughput low power compact aes s-box implementation using composite field arithmetic and algebraic normal form representation," in *2nd Asia Symposium on Quality Electronic Design (ASQED)*, Aug.

[5] H. Dobbertin, V. Rijmen, and A. Sowa, Eds., *Advanced Encryption Standard - AES, 4th International Conference, AES 2004, Bonn, Germany, May 10-12, 2004, Revised Selected and Invited Papers*, ser. Lecture Notes in Computer Science, vol. 3373. Springer, 2005. [Online]. Available: https://doi.org/10.1007/b137765

[6] J. Wolkerstorfer, E. Oswald, and M. Lamberger, "An asic implementation of the aes sboxes," in *Proceedings of the The Cryptographer's Track at the RSA Conference on Topics in Cryptology*.

[7] S. Picek, B. Yang, V. Rozic, and N. Mentens, "On the construction of hardware-friendly 4\times 4 and 5\times 5 s-boxes," in *Selected Areas in Cryptography - SAC 2016 - 23rd International Conference, St. John's, NL, Canada, August 10-12, 2016, Revised Selected Papers*.

[8] F. N. Najm, "A survey of power estimation techniques in vlsi circuits," *IEEE Trans. Very Large Scale Integr. Syst.*

[9] K. Nyberg, "On the construction of highly nonlinear permutations," in *Advances in Cryptology - EUROCRYPT' 92*, ser. Lecture Notes in Computer Science, R. Rueppel, Ed. Springer Berlin Heidelberg, 1993, vol. 658, pp. 92–98.

[10] ——, "S-boxes and round functions with controllable linearity and differential uniformity," in *Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994, Proceedings*, 1994, pp. 111–130.

[11] G. Leander and A. Poschmann, "On the Classification of 4 Bit S-Boxes," in *Arithmetic of Finite Fields*, ser. Lecture Notes in Computer Science, C. Carlet and B. Sunar, Eds. Springer Berlin Heidelberg, 2007, vol. 4547, pp. 159–176.

[12] A. P. Chandrakasan and R. W. Brodersen, Eds., *Low-Power CMOS Design*, 1st ed. Wiley-IEEE Press, 1997.

[13] L. Benini and G. De Micheli, *Logic Synthesis for Low Power*. Boston, MA: Springer US, 2002, pp. 197–223.

[14] S. Iman and M. Pedram, *Logic Restructuring for Low Power*. Boston, MA: Springer US, 1998, pp. 87–107. [Online]. Available: https://doi.org/10.1007/978-1-4615-5453-0_5

[15] ——, *Logic Restructuring for Low Power*. Boston, MA: Springer US, 1998, pp. 87–107. [Online]. Available: https://doi.org/10.1007/978-1-4615-5453-0_5

[16] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*.

[17] T. K. Ho, "Random decision forests," in *Proceedings of 3rd International Conference on Document Analysis and Recognition*, Aug 1995.

[18] E. Sentovich, K. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. Stephan, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Sis: A system for sequential circuit synthesis," EECS Department, University of California, Berkeley, Tech. Rep., 1992.

[19] R. L. Rudell, "Multiple-valued logic minimization for pla synthesis," EECS Department, University of California, Berkeley, Tech. Rep., 1986.

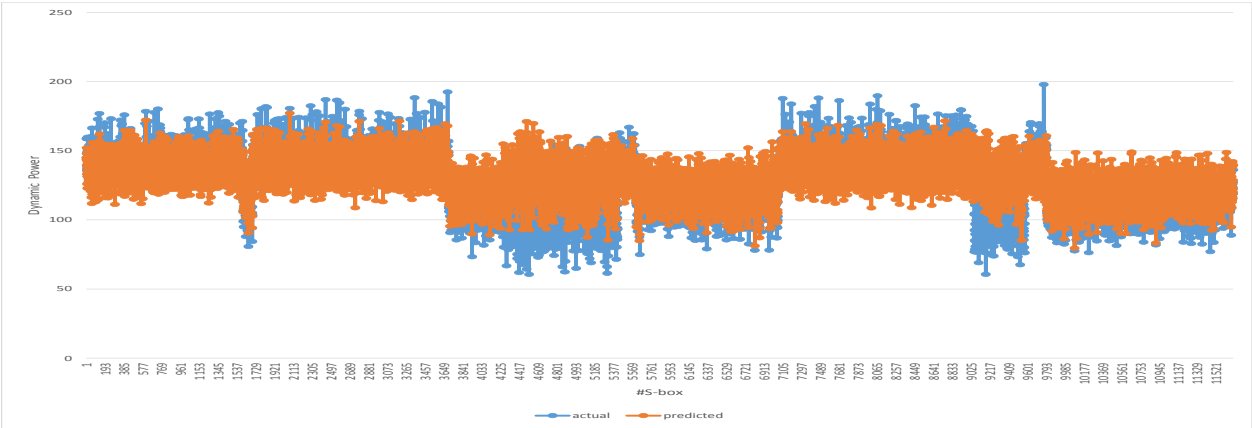


Fig. 8. Actual and Predicted Dynamic Power for 10,000 S-boxes

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.