IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 2, 2022, accepted April 20, 2022, date of publication May 3, 2022, date of current version May 10, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3172287

Power Prediction in Register Files Using Machine Learning

MOHAMMED ELNAWAWY, ASSIM SAGAHYROON, (Senior Member, IEEE), AND MICHEL PASQUIER, (Senior Member, IEEE)

Department of Computer Science and Engineering, American University of Sharjah, Sharjah, United Arab Emirates

Corresponding author: Mohammed Elnawawy (melnawawy@aus.edu)

This work was supported in part by the Department of Computer Science and Engineering and the Open Access Program from the American University of Sharjah.

ABSTRACT The advent of computer architecture and processor design in recent years has brought about the need to design larger register files that can hold more instructions and operands to support faster processors. This encouraged designers to design wider and deeper register files with multiple read and write ports to increase their throughput. Nevertheless, larger register files consume higher energy/access, leak more power, and occupy larger areas on the chip. This portrays a significant issue in the field of chip design due to the limited energy resources in mobile devices that dominate today's market. Therefore, it becomes crucial for chip designers to devise new mechanisms that help them study the effect of increasing the register file capabilities on those characteristics at an early stage during the design process. Artificial Neural Network (ANN) techniques, and with a reasonable degree of success have been used to predict the energy characteristics of a register file based on three parameters: the number of words in the file (D), the number of bits in one word (W) and the total number of Read and Write Ports (P). In this work, and using the same attributes, we attempt to predict the values of energy/access, leakage power, and occupied silicon area in register files using several machine learning algorithms to assess design alternatives and their energy and area tradeoffs. We compare our best algorithm to the ANN-based model reported in the literature using the same dataset. Support Vector Machine (SVM) models were able to achieve a correlation coefficient of 0.991, 0.991, and 0.989 when predicting energy/access, leakage power, and silicon area, respectively. On the other hand, the designed artificial neural network (ANN) achieves correlation coefficients of 0.974, 0.982, and 0.987, while the closest algorithms in performance to SVM achieve 0.917, 0.980, and 0.987, respectively. The results of the conducted experiments prove that SVM produce superior results when compared to ANN and other algorithms while maintaining a reasonable model training time and consuming lesser computational resources in most cases.

INDEX TERMS Register files, power consumption, leakage power, support vector machine.

I. INTRODUCTION

During chip design, to explore design spaces effectively, designers need options that rely on fast and accurate pre-silicon performance and power models. Simulation is commonly used for understanding architectural tradeoffs, however detailed simulators tend to be prohibitively slow.

In recent years, machine learning techniques have been experimented with in a variety of applications. Examples of reported work include [1] where authors discuss the use of an energy-aware technique based on the polynomial regression model to reduce the inefficiencies caused by the excessive power consumption in a data center. The technique attempts to predict the number of active physical machines required to run to reduce the inefficiency. A machine-learning based method to predict routing congestion for FPGA high-level synthesis was presented in [2]. After back tracing congestion metrics to IR operations, informative features were extracted, and three machine learning models were trained and compared. Experiments show that model can achieve a high prediction accuracy. Based on the model, routing congestion can be reduced easily, and performance is improved significantly. On estimation of power consumption during design, neural networks use has been explored in speeding up the process

The associate editor coordinating the review of this manuscript and approving it for publication was Jon Atli Benediktsson.

of finding a reliable estimate of power. In [3], the authors present a framework to select a standard cell library that can result in near-optimal power consumption while satisfying targeted frequency. The framework relies on a neural network model to quickly predict the total power of a block design associated with a given standard cell library to speed up the synthesis process. Experimental result based on sample synthesized benchmark circuits demonstrated the effectiveness of proposed approach.

Optimal register file design is critical to the performance of modern CPUs. Studies [4]–[6] have shown that register files, if not optimized can be a serious source of consumption. To counter this, researchers have continued to seek design alternatives that optimize their power consumption [6]–[8]. On the prediction side, similar efforts continue to try and develop techniques that would provide designers with a reasonable estimate of the power consumed by a perceived design and allow them to fine tune parameters to optimize the design. For example, in [9], authors present a Radial Base Function (RBF) Artificial Neural Network model for the prediction of energy/access and leakage power in standard cell register files designed using optimized Synopsys Design Ware components and an UMC130 nm library. In [10], authors presented a fast and light-weight cost estimation models for power and area that can be fully integrated in ASIP (Application Specific Instruction Set Processor) toolflows at system level. The cost estimation model gives a detailed overview of the power and area consumption of all basic ASIP components including register files. The objective is to allow designers to speed up the slow iterative design process without performing time consuming synthesis and power simulation. Virtual prototyping techniques to estimate register file power were discussed in [11]; using high level prototyping tool, researchers explored the methodologies of multi-port register file design to speed up power estimation while exploring design alternatives.

The addition of machine learning capabilities to electronic design automation solutions, and possible opportunities and challenges for the semiconductor industry is discussed in [12].

This paper is organized as follow, in section 2 we provide background information and discuss the source of data upon which the work presented here is based. In section 3, we illustrate the methodological approach and machine learning techniques experimented with. In sections 4 and 5 we discuss and assess the results; the paper is concluded in section 5.

II. BACKGROUND

Recent articles in the literature have continued to explore the power optimization problem in register files. In [13], authors discussed a gating scheme to reduce the dynamic power consumption in register files without impacting performance.

The proposed power saving scheme achieved a 19% reduction in power by taking advantage of the presence of frequent zero data produced in general purpose applications. Another technique that uses a power gating method to shut down unused register sub arrays is introduced in [14]. On average a 9% reduction in energy consumption is achieved. The use of latches in place of flip-flops to reduce area and power in register file design was explored in [15]. Note that flip-flops use two latches in their design, a master, and a slave. Hence, using only one latch will lead to reduction in both area and power. When synthetized using CMOS 45nm process libraries, the proposed approach produced a 23% reduction in leakage power.

Another method that reduced the average register file energy consumption by about 15% after overhead reduction is presented in [16]. The technique identifies duplicate data and eliminates it leading to the un-allocation of some of the register file banks and hence a saving in power.

In [17], researchers carried out a detailed simulation at the transistor level of various architectures of register files using optimized Synopsys Design Ware components from the UMC130 nm library. Layouts were generated for register files with a varying number of ports (P) ranging from 3 to 12, a depth that varies from 4 to 64 words (D), and a width (W) that varies from 8 to 128 bits. All these combinations of register files were designed; patristic capacitances in the routing wires and gate capacitances of each transistor were extracted from the layouts. The extracted netlist was then simulated using ModelSim with different switching activity factors to obtain power estimates. After completing over 100 register file design for the 130 nm technology node, the dynamic and leakage energy of each design was tabulated. Curve fitting was performed on each variable using register parameters D, W, and P as well as the activity factor as independent input variables. Furthermore, silicon area needs per design is also calculated. For all designs, it is assumed that each of the ports of the register file is driving a load of F04.

Equations (1), (2) and (3) below are the derived model equations, where Energy/Access and Leakage power are the subjects of the first two equations and silicon area is that of the third respectively:

$$\begin{aligned}\text{E/Access}\,(in:J) = {} & [2.23*10^{-11} - 8.06*10^{-13} \\ & *D - 5.89*10^{-13}*W - 3.35 \\ & *10^{-12}*P + 2.06*10^{-14}*D \\ & *W + 7.57*10^{-14}D*P + 6.34 \\ & *10^{-14}*W*P + 2.48*10^{-15} \\ & *D^2 + 9.93*10^{-16}*W^2 \\ & + 8.72*10^{-14}*P^2]*(HD/P)\end{aligned} \tag{1}$$

$$\begin{aligned}\text{Leakage Power}\,(in:\mu W) = {} & 5.43*10^{1} - 1.76*D - 1.62 \\ & *W - 8.42*P + 8.55*10^{-2} \\ & *D*W + 2.15*10^{-1}*D*P \\ & + 1.61*10^{-1}*W*P + 1.73 \\ & *10^{-3}*D^2 + 4.23*10^{-3} \\ & *W^2 + 2.10*10^{-1}*P^2\end{aligned} \tag{2}$$

$$\begin{aligned}\text{Area}\left(in:\mu m^2\right) = {} & 7.36*10^4 - 2.37*10^3*D \\ & - 2.12*10^3*W - 1.21*10^4 \\ & *P + 1.24*10^2*D*W \\ & + 3.33*10^2*D*P + 2.58 \\ & *10^2*W*P - 4.98*10^{-1} \\ & *D^2 + 1.56*W^2 + 2.71 \\ & *10^2*P^2 \end{aligned} \quad (3)$$

In the above equations HD is the total number of bits that switch (either from 1 to 0 or from 0 to 1) on the data and address lines from one read/write cycle to another. Upon examining the performance of these models, it is observed that they exhibited on average about 10% error when compared to the values obtained using detailed simulation.

In the work presented here, and to estimate the energy, leakage power and area, we use the 100 designs data sets obtained from detailed simulation discussed in [17]. We use this data set to explore the efficiency of various data mining techniques in estimating, energy, leakage power, and area in register files design. We compare our results with those obtained using the detailed simulation, as well as prediction estimates based on equations 1, 2, and 3 above [17].

III. METHODOLOGY

The problem at hand has many variables (D, P, W, HD), and the analytical models derived and given by equations 1, 2, and 3 above are not quite simple and the detailed simulation approach is expensive. Furthermore, the analytical models are only an approximation. Hence, in the work presented here, is an attempt in using machine learning techniques to leverage on the existing work.

Machine learning algorithms can be divided into four main categories: classification, regression, association, and clustering. Classification is the process of associating data instances with their discrete classes. Regression is also known as numeric prediction since it predicts a continuous output value unlike classification. Association learning focuses on uncovering hidden information in the dataset that was not previously known to the data scientist. Finally, in clustering problems the output class is not obvious and hence clustering algorithms tend to group data instances together based on some similarity features among instances of the same group.

In this work, we train and test machine learning models to predict three main parameters or characteristics that are critical to register file design, namely, the energy/access, leakage power, and area occupied on the chip. All these target characteristics are continuous in nature and hence this means that the problem explored here lends itself more to regression-based techniques.

One of the most prevalent machine learning tools is the Waikato Environment for Knowledge Analysis (Weka) [18]–[20]. Weka is an open-source suite written in Java that includes several visualization tools and machine learning algorithms used mainly for data mining and analysis.

In the work presented here, we use Weka platform to assess the performance of the following algorithms: Gaussian process, linear regression, support vector machine (SVM), k-nearest neighbor (k-NN), KStar, and random forest in predicting the register file parameters. These algorithms are used since they support regression problems that can predict continuous valued output classes like energy/access, leakage power, and area on the chip. Other algorithms like naïve Bayes which only support symbolic output classes were therefore discarded. We use the stratified 10-fold cross-validation technique that splits the entire dataset into 10 non-overlapping folds. After that, 9 folds are used for training and the remaining fold is used for testing the generated model. The process is repeated until all 10 folds have been used once for testing.

A brief introduction to the different machine learning algorithms used is provided below. A detailed discussion of these algorithms is beyond the scope of this paper but can be found in [21], [22].

A. LINEAR REGRESSION (LR)

Linear regression [23] is a linear technique used to model the relationship between independent attributes and the dependent outcome. In this paper, the independent attributes are words (D), width (W), and ports (P), while the outcomes are the energy/access, leakage power, and area. In fitting the linear regression model, Weka tends to minimize the squared error in what is known as least squares approach. The resulting linear regression model is described by Equation (4).

$$y = w_0 + w_1x_1 + w_2x_2 + \cdots + +w_nx_n \quad (4)$$

where w_i is the weight given to the i^{th} independent variable x_i, n is the number of independent variables considered in building the regression model, and y is the dependent response variable. w_0 is sometimes referred to as the bias. Linear regression models s result in a straight line in 2-dimensions, a plane in 3-dimentions, and a hyperplane in higher dimensions.

Even though the dataset under investigation is non-linear by nature, we decided to train a linear regression model as a starting point to our study to raise the baseline classifier performance compared to a random classifier. Therefore, we anticipate in advance that the linear regression model might not yield good results since it is meant for linear data points. Nevertheless, linear regression is a very easy-to-use and easy-to-interpret model which further encouraged us to try it on the register file dataset.

B. GAUSSIAN PROCESS (GP)

A Gaussian process [24] is a group of random variables such that every finite group of those random variables has a multivariate normal distribution. This means that every possible linear combination of those random variables follows a normal distribution. The Gaussian process distribution is the joint distribution of all these random variables. Gaussian processes use the kernel function that computes the similarity

between instances to predict the value for a new data instance. A vector-valued random variable $x \in R^n$ is said to have a multivariate Gaussian distribution with mean $\mu \in R^n$ and covariance matrix $\Sigma \in S^n++$ if it follows Equation (5).

$$P(x, \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)} \quad (5)$$

One of the main advantages of GP is the fact that it enables the data scientist to incorporate expert knowledge through the choice of the kernel function. The power prediction problem is a well-known non-linear problem; hence we opt for a polynomial kernel of 3rd order when fitting the three predictive models mentioned earlier. Nevertheless, the disadvantage of GP is its computationally expensive nature.

C. SUPPORT VECTOR MACHINE (SVM)

In classification problems, SVM [25], [26] is an algorithm that focuses on determining the optimal hyperplane that separates the classes under investigation through maximizing the distance between the separating hyperplane and the critical points at the decision boundary which are known as support vectors. Therefore, SVM finds the most optimal hyperplane that segregates the regions belonging to the different classes. Hence, when a test instance falls in a specific region it is associated with the class belonging to that region. On the other hand, in regression problems this hyperplane is not just a separating entity, but rather the decisive factor that helps us predict the continuous value of the dependent response variable. This means that the hyperplane is more of an equation that determines the continuous-valued output class. In this paper, we build an SVM regression model using a 3rd order polynomial kernel to account for the non-linear nature of the register file dataset. The 3rd order polynomial kernel is characterized by Equation (6).

$$K(x, y) = \left(x^T y + c\right)^3 \quad (6)$$

where x and y are vectors in the input space that resemble feature vectors computed from training or testing data points and c is a free parameter trading off the effect of higher-order versus lower-order terms in the polynomial.

D. K-NEAREST NEIGHBOR (K-NN)

k-NN [27] is one of the simplest yet most popular machine learning techniques used to build predictive models due to their power in generalizing well to most datasets. It is known to be a non-parametric machine learning method that does not rely on the parametric method of probability distributions. k-NN is also called as instance-based or lazy learning since the learnt model simply stores the data points themselves in memory and tries to find the k neighboring points to the incoming test instance and hence assigns the majority class of the neighboring instances to the test instance. In regression problems, since we do not have discrete classes, the model calculates the average response variable of the k-nearest neighbors and assigns it to the new instance. At first, we used the rule-of-thumb to pick a suitable value of k which says that k is approximately equal to the square root of the number of samples in the dataset. The number of samples in our possession is 80 samples and hence k must be 9. Nevertheless, after tuning the value of k we found out that $k = 3$ yields the best regression performance. As a result, we use $k = 3$ to consider only the three nearest neighbors in finding the predicted values of the data instances. In finding the nearest neighbors, k-NN usually uses some distance measures like the Euclidean and the Manhattan distances to find the closest training points to the incoming test point. Nevertheless, the Euclidean distance remains the most popular of all distance measures as it takes reduces the effect of noisy data points through squaring the difference between two data points.

The Euclidean distance is given by Equation (7):

$$Euclidean\ distance = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (7)$$

where x and y are the independent variables belonging to the two data instances under investigation (test and train instances).

E. KStar

KStar [28] is very similar to k-NN since it also finds the average response variable of the k-nearest neighbors. Nonetheless, the difference between KStar and k-NN lies in the usage of an entropy-based distance function instead of the normal distance functions discussed earlier like Euclidean and Manhattan distances. The intuition of entropy-based distance measures is that the distance between data points can be thought of as the complexity of transforming one data point to another data point [27].

F. WEIGHTED LEARNING (LWL)

LWL [29], [30] is another instance-based learning that operates in a very similar fashion to k-Nearest neighbor except that it weighs the contributions of each training point (nearest neighbors) according to their distances from the test point.

G. RANDOM FOREST (RF)

The seventh algorithm considered is random forest [31]. It is an ensemble training method that works through constructing multiple decorrelated decision trees to improve the regression performance. In regression problems, the test instance is routed through all trees simultaneously resulting in a predicted value at the leaf node of each tree. The final predicted response variable is then computed as the average of all leaf node values resulting from all trees in the forest. Decision trees are usually very good predictors of non-linearly separable datasets. However, decision trees suffer from the problem of overfitting if allowed to grow deeply since they tend to learn uncommon characteristics about data instances that might simply be noise in the dataset. Therefore, random forest is a very good solution that minimizes the effect of overfitting since it builds many uncorrelated trees and hence

significantly reduces the possibility of overfitting to the training dataset.

Initially, the data scientist specifies the number of required trees in the forest (N) and given the number of training instances (m), the algorithm builds N separate decision trees by sampling with replacement m training instances from the entire dataset. This results in trees that are decorrelated since the training sets used to train each tree are different. To find out the optimal number of trees in the forest we vary the number of trees from 10 all the way to 200 and observe the out-of-bag error. We observe that upon increasing the number of trees beyond 100 we do not significantly reduce the out-of-bag error; hence we use 100 trees in our random forest model. At every node of each decision tree, the algorithm picks randomly several independent variables (usually one third of the total number of variables for regression problems) and chooses one variable to split on based on a specific purity measure. The most famous purity measure is the entropy-based purity measure which is therefore used in this work.

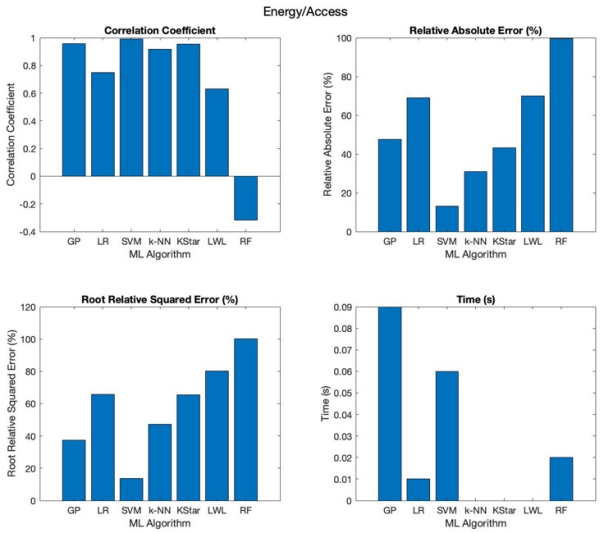

FIGURE 1. Energy per access comparison.

IV. RESULTS AND DISCUSSIONS

This section presents and discuss results obtained after running the algorithms on the dataset. The following performance metrics: Correlation coefficient (r), Mean absolute error (MAE), Root mean squared error (RMSE), Relative absolute error (%) (RAE), and Root relative squared error (%) (RRSE) are used to compare the performances of the different algorithms [32], [33]. Furthermore, we also compare the results to those obtained using different approaches, namely, an ANN-based approach in [9], and the empire prediction methodology discussed in [17].

Table 1 and Fig. 1 show results in terms of the performance metrics r, RAE, RRSE, and the time taken to build the seven models to predict energy/access. Note that, MAE and RMSE are not reported for Energy/Access since they were found out to be zeros for all the used algorithms. Fig. 1 shows that SVM surpasses all other algorithms in terms of r, RAE, RRSE, it has the highest r (0.991) and the least RAE (13.2%) and RRSE (13.6%). In addition, the correlation coefficient results prove that instance-based learning could be a potential candidate for accurately predicting the energy/access in a register file since they also result in a very high correlation between the predicted values and the actual values (0.956 using KStar and 0.917 using k-NN). Nevertheless, when we inspect the RAE and RRSE results, we can clearly see a huge increase in the errors of instance-based learning when compared to SVM. If we were to compare the RAE and RRSE of SVM and k-NN, for example, we would find that the use of k-NN increases the RAE and RRSE by approximately 18% and 33.5%, respectively. Moreover, random forest tends to yield the worst results in terms of all performance metrics including the correlation coefficient which shows a very poor relationship between the predicted and the actual results (−0.318), as well as RAE (99.7%) and RRSE (100%). Another interesting dimension to look at is the time taken to build the seven models as it highlights the difficulty with which the algorithm can build a representative model that could generalize to the dataset in hand. It is observed that instance-based learning takes negligible amount of time in building their models since they simply store the instances in memory, whereas Gaussian process takes a considerable amount of time to build the model. On the other hand, SVM, the best performer, takes a slightly longer time to generate its model compared to other algorithms.

TABLE 1. Energy per access comparison results.

Algorithm	Correlation Coefficient (r)	Relative Absolute Error (RAE) (%)	Root Relative Squared Error (RRSE) (%)	Time (s)
Gaussian Process	0.527	107.9	91.0	0.090
Linear Regression	0.748	68.9	65.7	0.010
SVM (3rd order)	0.991	13.2	13.6	0.060
k-NN (k = 3)	0.917	40.0	47.1	0.000
KStar	0.956	43.1	65.4	0.000
LWL	0.629	69.9	80.1	0.000
Random Forest	-0.318	99.7	100.1	0.020

Table 2 and Fig. 2 show the obtained results in terms of the performance metrics r, MAE, RMSE, RAE, RRSE, and the time taken to build the seven models mentioned earlier to predict leakage power. The leakage power models tend to behave like the models generated for energy/access. Yet again, SVM outperforms the other six algorithms in all performance metrics especially the correlation coefficient (0.991). Similarly, the instance-based algorithms tend to show very high positive correlation between the predicted and the actual values of the response variable (0.980 using KStar and 0.949 using k-NN).

TABLE 2. Leakage power estimate performance comparison.

Algorithm	Corr. Coeff. (r)	Mean Abs. Err.	Root Mean Squared Err.	Rel. Abs. Err. (%)	Root Rel. Squared Err. (%)	Time (s)
Gaussian Process	0.604	56.0	74.4	96.9	85.8	0.010
Linear Regression	0.813	33.5	50.0	57.9	57.6	0.000
SVM (3rd order)	0.991	8.50	11.7	14.7	13.4	0.020
k-NN (k = 3)	0.949	17.3	34.6	30.0	39.8	0.000
KStar	0.980	24.9	53.2	43.1	61.3	0.000
LWL	0.710	38.7	63.0	67.0	72.6	0.000
Random Forest	0.954	13.1	28.6	22.7	32.9	0.050

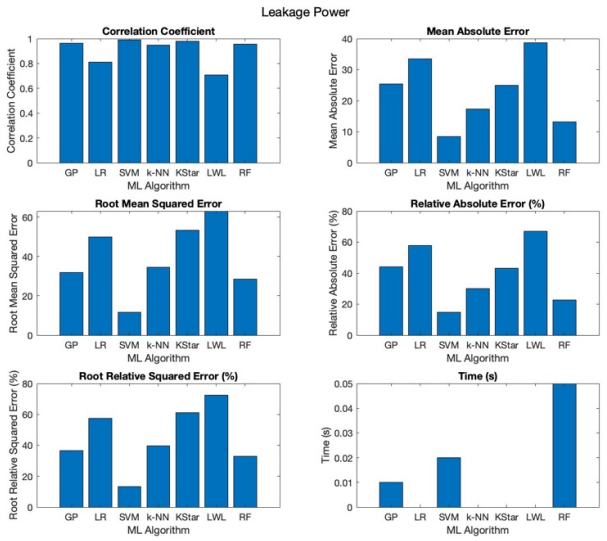

FIGURE 2. Leakage power performance.

A noteworthy observation in Fig. 2 is how the random forest algorithm was able to get close to the performance of SVM unlike the performance of random forest on the energy/access variable. In fact, random forest is the second-best algorithm according to all the performance metrics. Nevertheless, the time taken to build the random forest algorithm (0.050 s) is the highest among all other algorithms with SVM taking less than half the time (0.020 s). Moreover, the worst performer in case of leakage power is the LWL algorithm based on all performance metrics.

As mentioned earlier, the root mean squared error tends to be more useful than the mean absolute error in case the dataset had lots of outliers as squaring the errors would further highlight the discrepancies between the predicted and the actual values. However, we also notice from Fig. 2 that the mean absolute error results go hand in hand with the root mean squared error results, which indicates that the dataset used does not include a noticeable number of outliers. The same comment could be made using the results of relative absolute error and the root relative squared error.

Fig. 3 shows results related to the prediction of the required silicon area to design the register file. Here as well, SVM

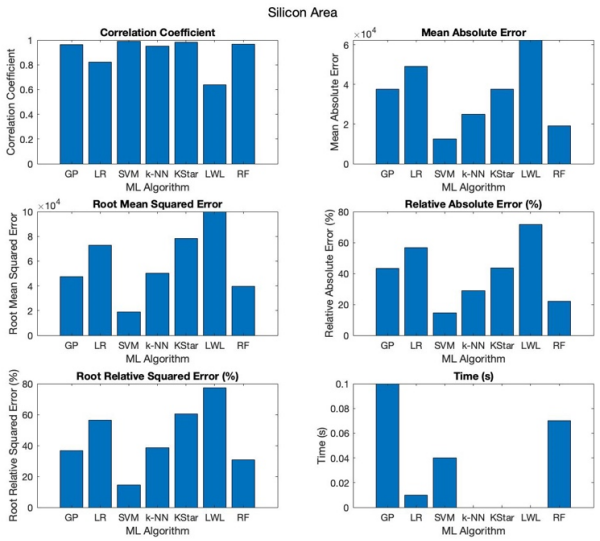

FIGURE 3. Area comparison.

demonstrates its superiority in terms of performance compared to the other six algorithms proving that it is the best algorithm one more time according to the five different performance metrics used. Similarly, random forest comes second in accurately predicting the area of the register file on the chip. Perhaps the worst algorithm in predicting the area on the chip is the LWL algorithm according to all metrics. Finally, SVM ensures its strength in one more aspect which is the time taken to build the model since it takes a reasonable amount of time to build compared to the random forest and Gaussian process algorithms.

To further study the results, we compare the correlation coefficients obtained in the work described here with those that were acquired using ANN techniques using the same register file dataset [9]. In their work, the authors used a multilayer RBF ANN that consists of two hidden layers where the first layer consists of 6 nodes, while the second layer consists of 4 nodes. Using the same dataset, the authors in [9] were able to achieve correlation coefficients of 0.97445 and 0.982399 for energy/access and leakage power, respectively. On the other hand, our SVM algorithm was able to achieve correlation coefficients of 0.9905 and 0.991 for energy/access and leakage power, respectively. In addition, and using the same dataset, the authors in [34] were able to achieve a correlation coefficient of 0.98722 for the area variable, whereas in our study, the SVM-based approach achieved a higher correlation of 0.9893. Results show a consistent improvement in correlation coefficient obtained with SVM when compared to ANN which signifies the power of SVM in predicting energy/access, leakage power, and area. Therefore, we can safely conclude that SVM has outperformed ANN in this domain.

Fig. 4 shows the prediction and accuracy of the SVM model based on the prediction of the energy/access of each data instance in the dataset when compared to the detailed simulation values.

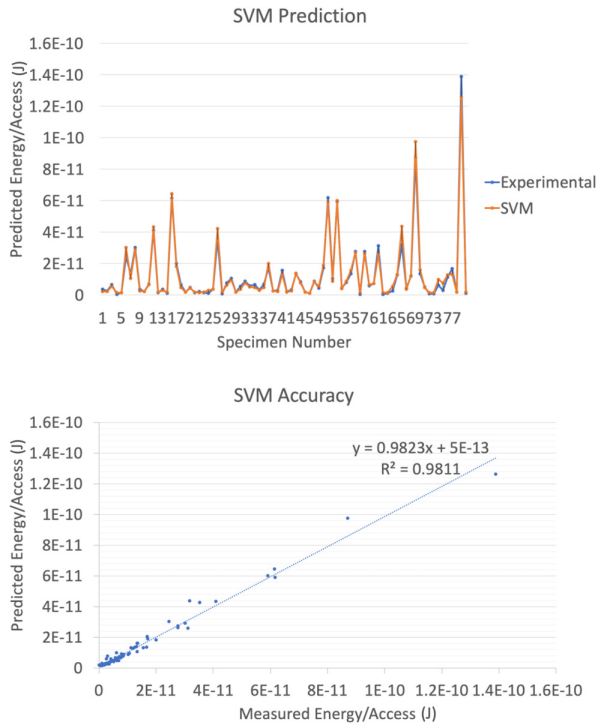

FIGURE 4. Energy per access.

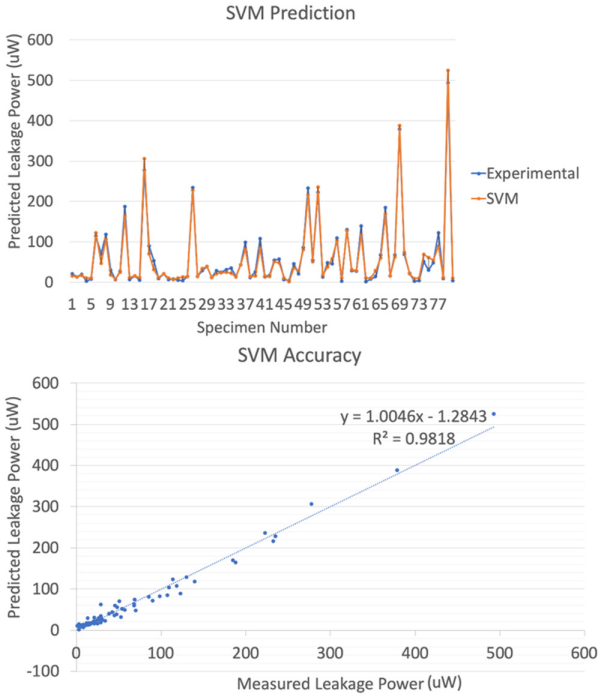

FIGURE 5. Leakage power.

Fig. 5 shows the prediction and accuracy of the SVM model based on the prediction of the leakage power of each data instance in the dataset when compared to the detailed simulation values.

Fig. 6 shows the prediction and accuracy of the SVM model based on the prediction of the silicon area of each data instance in the dataset when compared to the detailed simulation values.

TABLE 3. SVM complexity parameter (c) optimization.

Target	Complexity Parameter (c)	Correlation Coefficient (r)	Relative Absolute Error (RAE) (%)	Root Relative Squared Error (RRSE) (%)
Energy/Access	8	0.991	12.9	13.6
Leakage Power	6	0.990	15.2	14.0
Silicon Area	16	0.990	14.4	14.4

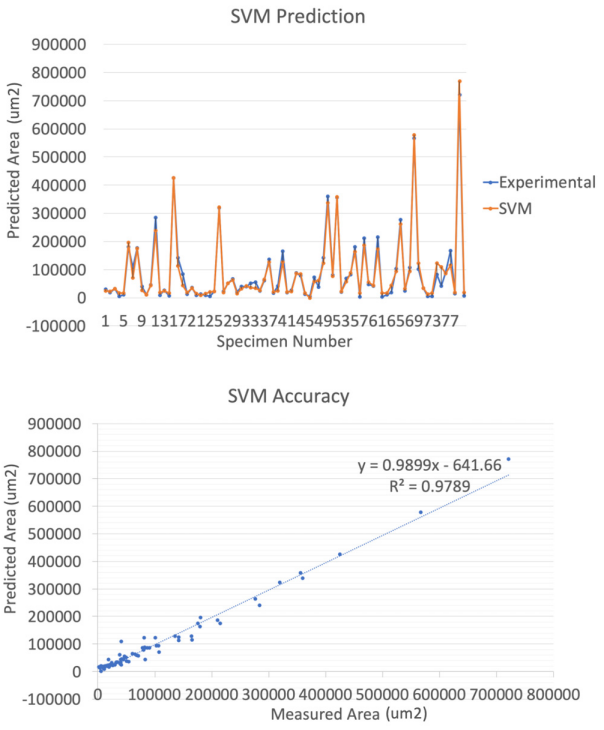

FIGURE 6. Area.

Now that we have concluded that SVM outperforms the other algorithms in predicting the three target variables, we attempt to optimize the complexity parameter c of the polynomial kernel given by Equation (6). To do so, we used Weka's CVParameter Selection meta-classifier [20] to vary c from 1 to 20 in steps of 1. Table 3 shows the obtained complexity parameter for the energy/access, leakage power, and silicon area models, in addition to the resulting performance metrics.

V. PARAMETRIC STUDY

To further compare the performance of the different prediction models, we varied the input parameters (width, ports, and depth) and computed the resulting outputs. For brevity, comparative plots are only shown in Fig. 7 for Leakage Power.

For example, in 7(a) and 7(b), D and P are fixed at 16,9 and 32,12 respectively and W was varied. From Fig. 7(a),

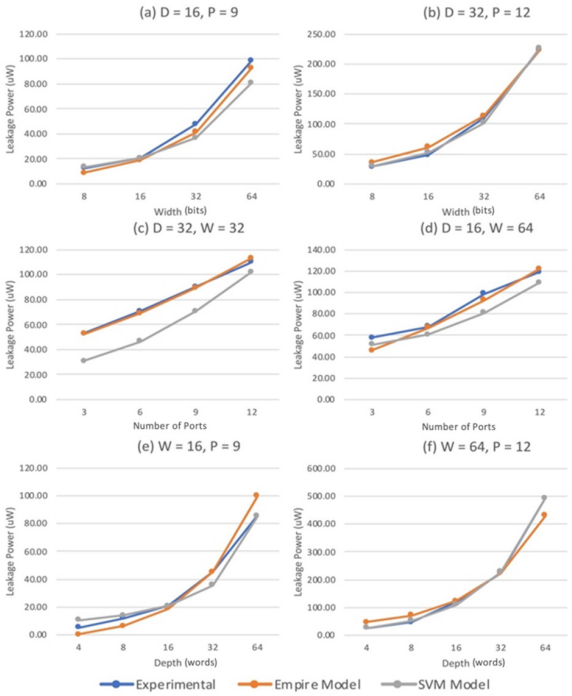

FIGURE 7. Comparison of leakage power for register files with different parameters.

Empire [17] seems to underestimate the Leakage Power prediction for wider designs with relatively fewer ports since from Fig. 7(b), as the Depth and number of Ports have increased, the estimates of the SVM model have improved. Similar observations can be made from the rest of the figures. Evidently, when one takes into consideration the amount of time and resources to complete a detailed transistor-level based simulation of these models, SVM prediction succeeds in providing designers with acceptable estimates in a short amount of time.

VI. CONCLUSION

In recent years, register files have become crucial components to the manufacturing of microprocessors due to their direct impact on the performance and throughput of commercial processors. The conclusions of the work presented in this paper can be summarized as follows:

- When designing register files, the number of words in the file (D), the number of bits in one word (W) and the total number of Read and Write Ports (P) have a direct impact on the power consumption. Increasing one or more of these parameters impacts the power consumption characteristics of mobile devices that are powered by limited energy resources. Hence, we investigated alternative register file designs by varying these three main parameters of the register files to have an insight before committing to silicon.
- We compared the performance of several machine learning algorithms in forecasting crucial design parameters such as energy/access, leakage power, and the area on the silicon chip of the register file early during the design process based on the previously mentioned attributes.
- It was evident that the Support vector machine (SVM) algorithm tends to outperform a variety of high-end regression models including random forest, and k-NN in terms of correlation coefficient, relative absolute error, root relative squared error and others. SVM achieved the highest correlation coefficient of 0.991, 0.991, and 0.989 when predicting energy/access, leakage power, and silicon area, respectively.
- On the other hand, the next top performing machine learning algorithms achieved correlation coefficients of 0.917, 0.980, and 0.987, respectively. This leads to a percentage increase of 8.07%, 1.12%, 0.20% in correlation coefficients, respectively.
- In addition, we also compared the SVM approach to a previously reported artificial neural network-based model in the literature. The reported ANN was able to achieve a correlation coefficient of 0.974, 0.982, and 0.987 when predicting energy/access, leakage power, and silicon area, respectively, using the same dataset. This leads to a percentage increase of 1.75%, 0.92%, 0.20% in correlation coefficients, respectively, while requiring lesser computational resources on average. This helps us conclude that SVM is still superior to ANN models.

ACKNOWLEDGMENT

This article represents the opinions of the authors and does not mean to represent the position or opinions of the American University of Sharjah.

REFERENCES

[1] A. Rayan and Y. Nah, "Energy-aware resource prediction in virtualized data centers: A machine learning approach," in *Proc. IEEE Int. Conf. Consum. Electron.-Asia (ICCE-Asia)*, Jun. 2018, pp. 206–212, doi: 10.1109/ICCE-ASIA.2018.8552101.

[2] J. Zhao, T. Liang, S. Sinha, and W. Zhang, "Machine learning based routing congestion prediction in FPGA high-level synthesis," in *Proc. Des., Autom. Test Eur. Conf. Exhib.*, 2019, pp. 1130–1135, doi: 10.23919/DATE.2019.8714724.

[3] S. H. Lim, Y. W. Lim, S. Mashohor, N. A. Kamsani, R. M. Sidek, S. J. Hashim, and F. Z. Rokhani, "Generating power-optimal standard cell library specification using neural network technique," in *Proc. IEEE Asia Pacific Conf. Postgraduate Res. Microelectron. Electron.*, Oct. 2017, pp. 101–104, doi: 10.1109/PRIMEASIA.2017.8280374.

[4] X. Guan and Y. Fei, "Reducing power consumption of embedded processors through register file partitioning and compiler support," in *Proc. Int. Conf. Appl.-Specific Syst., Archit. Process.*, Jul. 2008, pp. 269–274, doi: 10.1109/ASAP.2008.4580190.

[5] Y. Zhou, H. Guo, and J. Gu, "Register file customization for low power embedded processors," in *Proc. 2nd IEEE Int. Conf. Comput. Sci. Inf. Technol.*, 2009, pp. 92–96, doi: 10.1109/ICCSIT.2009.5234988.

[6] M. Abdel-Majeed and M. Annavaram, "Warped register file: A power efficient register file for GPGPUs," in *Proc. IEEE 19th Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2013, pp. 412–423, doi: 10.1109/HPCA.2013.6522337.

[7] Z. Zou, J. Han, Z. Yu, and X. Zeng, "A 32×32B 65NM 4R/2W register file for low-power operation," in *Proc. 12th IEEE Int. Conf. Solid-State Integr. Circuit Technol. (ICSICT)*, Oct. 2014, pp. 1–3, doi: 10.1109/ICSICT.2014.7021598.

[8] Z. Liu, Z. Zhu, J. Shi, J. Liu, and S. Li, "A low power buffer-aided vector register file for LTE baseband signal processing," in *Proc. 33rd IEEE Int. Conf. Comput. Des. (ICCD)*, Oct. 2015, pp. 403–406, doi: 10.1109/ICCD.2015.7357134.
[9] A. A. Sagahyroon and J. A. Abdalla, "Dynamic and leakage power estimation in register files using neural networks," *Circuits Syst.*, vol. 3, no. 2, pp. 119–125, 2012, doi: 10.4236/cs.2012.32016.
[10] S. A. A. Shah, J. Wagner, T. Schuster, and M. Berekovic, "A lightweight-system-level power and area estimation methodology for application specific instruction set processors," in *Proc. 24th Int. Workshop Power Timing Model., Optim. Simul. (PATMOS)*, Sep. 2014, pp. 1–5, doi: 10.1109/PATMOS.2014.6951886.
[11] N. Liu and B. Calhoun, "Design optimization of register file throughput and energy using a virtual prototyping (ViPro) tool," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2016, pp. 535–540, doi: 10.1109/ISVLSI.2016.50.
[12] E. Fallon, "Machine learning in EDA: Opportunities and challenges," in *Proc. ACM/IEEE 2nd Workshop Mach. Learn. (CAD)*, Apr. 2020, p. 103, doi: 10.1145/3380446.3430687.
[13] A. M. Radaideh and P. V. Gratz, "Exploiting zero data to reduce register file and execution unit dynamic power consumption in GPGPUs," in *Proc. 57th ACM/IEEE Des. Autom. Conf. (DAC)*, Jul. 2020, pp. 1–6, doi: 10.1109/DAC18072.2020.9218547.
[14] X. Wang and W. Zhang, "Packing narrow-width operands to improve energy efficiency of general-purpose GPU computing," in *Proc. IEEE High Perform. Extreme Comput. Conf. (HPEC)*, Sep. 2020, pp. 22–24, doi: 10.1109/HPEC43674.2020.9286215.
[15] A. K. Patan, D. Stathis, P. Dhilleswararao, Y. Yang, S. Boppu, and A. Hemani, "Design and implementation of optimized register file for streaming applications," in *Proc. 25th Int. Symp. VLSI Des. Test (VDAT)*, Sep. 2021, pp. 16–18, doi: 10.1109/VDAT53777.2021.9600984.
[16] A. Yazdanpanah, S. Sajadimanesh, and S. Safari, "EREER: Energy-aware register file and execution unit using exploiting redundancy in GPGPUs," *Microprocess. Microsyst.*, vol. 77, Sep. 2020, Art. no. 103176, doi: 10.1016/j.micpro.2020.103176.
[17] P. Raghavan, A. Lambrechts, M. Jayapala, F. Catthoor, and D. Verkest, "EMPIRE: Empirical power/area/timing models for register files," *Microprocess. Microsyst.*, vol. 33, no. 4, pp. 295–300, Jun. 2009, doi: 10.1016/j.micpro.2009.02.009.
[18] S. R. Garner, "WEKA: The waikato environment for knowledge analysis," in *Proc. 2nd New Zealand Comput. Sci. Res. Students Conf.*, Hamilton, New Zealand, 1995, pp. 57–64.
[19] A. Naik and L. Samant, "Correlation review of classification algorithm using data mining tool: WEKA, Rapidminer, Tanagra, Orange and Knime," *Proc. Comput. Sci.*, vol. 85, pp. 662–668, Jan. 2016, doi: 10.1016/j.procs.2016.05.251.
[20] T. Smith and E. Frank, *Introducing Machine Learning Concepts With WEKA* (Methods in Molecular Biology), vol. 1418. New York, NY, USA: Humana Press, 2016, pp. 353–378, doi: 10.1007/978-1-4939-3578-9_17.
[21] S. S. Shwartz and S. B. David, *Understanding Machine Learning: From Theory to Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2014.
[22] J. D. Kelleher, B. M. Namee, and A. D'Arcy, *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies*. Cambridge, MA, USA: MIT Press, 2015.
[23] S. Swaminathan. (Jan. 15, 2021). *Linear Regression—Detailed View*. [Online]. Available: https://towardsdatascience.com/linear-regression-detailed-view-ea73175f6e86
[24] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, 2006.
[25] S. K. Shevade, S. S. Keerthi, C. Bhattacharyya, and K. R. K. Murthy, "Improvements to the SMO algorithm for SVM regression," *IEEE Trans. Neural Netw.*, vol. 11, no. 5, pp. 1188–1193, Sep. 2000.
[26] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statist. Comput.*, vol. 14, no. 3, pp. 199–222, Aug. 2004.
[27] D. Aha, D. Kibler, and M. Albert, "Instance-based learning algorithms," *Mach. Learn.*, vol. 6, pp. 37–66, Jan. 1991, doi: 10.1007/BF00153759.
[28] J. G. Cleary and L. E. Trigg, "K*: An instance-based learner using an entropic distance measure," in *Proc. 12th Int. Conf. Mach. Learn.*, 1995, pp. 108–114, doi: 10.1016/B978-1-55860-377-6.50022-0.
[29] E. Frank, M. Hall, and B. Pfahringer, "Locally weighted naive Bayes," in *Proc. 19th Conf. Uncertainty Artif. Intell.*, Acapulco, Mexico, 2003, pp. 249–256.
[30] C. Atkeson, A. Moore, and S. Schaal, "Locally weighted learning," *Artif. Intell. Rev.*, vol. 11, no. 1, pp. 11–73, Apr. 1997.
[31] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[32] S. Orozco-Arias, J. S. Piña, R. Tabares-Soto, L. F. Castillo-Ossa, R. Guyot, and G. Isaza, "Measuring performance metrics of machine learning algorithms for detecting and classifying transposable elements," *Processes*, vol. 8, no. 6, p. 638, May 2020, doi: 10.3390/pr8060638.
[33] M. Gharib and A. Bondavalli, "On the evaluation measures for machine learning algorithms for safety-critical systems," in *Proc. 15th Eur. Dependable Comput. Conf. (EDCC)*, Sep. 2019, pp. 141–144, doi: 10.1109/EDCC.2019.00035.
[34] A. Sagahyroon and J. Abdalla, "Area and timing estimation in register files using neural networks," *Circuits Syst.*, vol. 3, no. 3, pp. 269–277, 2012, doi: 10.4236/cs.2012.33037.

MOHAMMED ELNAWAWY was born in Giza, Egypt, in 1995. He received the B.Sc. *(summa cum laude)* and M.Sc. degrees in computer engineering from the American University of Sharjah, United Arab Emirates, in 2017 and 2019, respectively. He worked as a Graduate Teacher and a Research Assistant at the American University of Sharjah. In 2020, he joined the American University of Sharjah as a Laboratory Instructor at the Department of Computer Science and Engineering. His research interests include machine learning, field-programmable gate arrays, and embedded systems. He is a member of the Upsilon Pi Epsilon Honor Society. He won several awards, including the American University of Sharjah Graduate Student Research, Scholarly, and Creative Work Excellence Award, and the Sharjah Islamic Bank Research Award.

ASSIM SAGAHYROON (Senior Member, IEEE) received the M.Sc. degree in electrical engineering from Northwestern University, Evanston, IL, USA, and the Ph.D. degree from the University of Arizona, Tucson, AZ, USA. From 1993 to 1999, he has been a member of the Department of Computer Science and Engineering, Northern Arizona University. In 1999, he joined the Department of Math and Computer Science, California State University. In 2003, he joined the Department of Computer Science and Engineering, American University of Sharjah, where he served as the Department Head for seven years. He was an invited as a technical reviewer for some of the National Science Foundation programs and served on the technical program committees of many conferences. He has many publications in international conferences and journals. His research interests include innovative applications of emerging technology in the medical field, power consumption of portable electronics, and FPGAs-based design.

MICHEL PASQUIER (Senior Member, IEEE) received the Diploma degree in electrical engineering and the Ph.D. degree in computer science from the Grenoble Institute of Technology, France, in 1985 and 1989, respectively. In 1989, he left the LIFIA Laboratory (now INRIA) for the Science City, Tsukuba, Japan, to do research at the Electro-Technical Laboratory (AIST). In 1992, he was at the Sanyo's Intelligent Systems Laboratory. In 1994, he joined Nanyang Technological University, Singapore, as Faculty Member in AI and machine learning. In 2004, he became the Founding Director of the Centre for Computational Intelligence. In 2009, he joined the American University of Sharjah (AUS), United Arab Emirates. He has led projects, published, and served as a consultant in many AI application areas, including mobile robotics, intelligent transportation, medical diagnosis, healthcare, online learning, and financial engineering. His research interests include adaptation and learning, intelligent systems and robotics, decision support systems, knowledge discovery, and nature-inspired methods and algorithms.