Parameter Tuning in SVM-Based Power Macro-Modeling

António Gusmão L. Miguel Silveira José Monteiro
TU Lisbon, IST / INESC-ID

Rua Alves Redol, 9
1000-029 Lisboa, Portugal
{antoniog,lms,jcm}@algos.inesc-id.pt

Abstract

We investigate the use of support vector machines (SVMs) to determine simpler and better fit power macromodels of functional units for high-level power estimation. The basic approach is first to obtain the power consumption of the module for a large number of points in the input signal space. Least-Squares SVMs are then used to compute the best model to fit this set of points. We have performed extensive experiments in order to determine the best parameters for the kernels. Based on this analysis, we propose an iterative method of improving the model by selectively adding new support vectors and increasing the sharpness of the model. The macromodels obtained confirm the excellent modelling capabilities of the proposed kernel-based method, providing both excellent accuracy on maximum error (close to 17%) and average (2% error), which represents an improvement over the state-of-the-art. Furthermore, we present an analysis of the dynamic range of power consumption for the benchmarks circuits, which serves to confirm that the model is able to accommodate circuits exhibiting a more skewed power distribution.

Keywords

Power Estimation, Support Vector Machines, Macro-Model

1 Introduction

Electronic systems have become pervasive in our daily lives, work environments and even our social habits. Such systems are composed of a complex mix of digital and mixed-signal circuit blocks. Their design and verification prior to fabrication are challenging tasks due to inherent size and complexity. A common approach towards easing the design process and enabling verification is to replace large and complex design blocks by smaller macromodels that accurately represent relevant characteristics of the system. The resulting macromodelled system can then be verified and design exploration can be accomplished, leading to more efficient designs, with lower costs and added features.

Power consumption has become one of the most important parameters in the design of VLSI circuits and accurate power estimation a requisite of any design exploration framework and verification environment. Many high-level power estimation tools have been proposed before to enable the evaluation of different architectures in early stages of design [3]. The general approach is to use power macromodels for each functional unit. These macromodels are obtained in a pre-characterization phase, stored in a library for later use and represent the power dissipation of the unit as a function of the input statistics.

Kernel methods provide a powerful and unified framework for pattern discovery, motivating algorithms that can act on general types of data and look for general types of relations (*e.g.* rankings, classifications, regressions, clusters) [9]. The application areas range from neural networks and pattern recognition to machine learning and data mining. In this paper, we investigate the use of learning algorithms, in particular support vector machines (SVMs), to determine simpler and better fit power macromodels. The basic approach is first to obtain the power consumption of the module for a large number of points in the input signal space. Least-Squares SVMs (LS-SVM) are then used to compute the best model to fit these set of points. The statistics for each of the module's output signals can be computed in a similar manner, thus providing a means of propagating the switching probabilities through the circuit. We have performed extensive experiments in order to analyze the possible kernels which are the basis of the SVM formulation and to determine the best kernel parameters. An iterative method to improve the accuracy of the model by selectively adding new data points to the training set is proposed.

We present results that confirm the excellent modelling capabilities of the kernel-based methods and perform an analysis of the dynamic range of the circuits chosen as benchmarks. Experiments point out a wide variety of scenarios among benchmark circuits in terms of the distribution of the power consumption over different input patterns. Naturally, circuits with smaller dynamic range are easier to model, as observed in terms of the errors obtained. In fact, this analysis indicates that very simple macromodels may be used for some circuits. On the other hand, although more difficult to model, circuits with more skewed power consumption distributions are still accurately modeled the LS-SVM-based macromodels.

The paper is organized as follows. In Section 2, we review previous work on power analysis techniques at the RT level and provide some background on kernel methods. Section 3 discusses the tuning of the kernel parameters and Section 4 presents an iterative optimization method based on those observations. We present our results in Section 5 and draw conclusions in Section 6.

2 Related Work

2.1 Power Macro-Modeling

There has been a fair amount of work on generating models for power dissipation at higher levels of abstraction. Top-down approaches have been proposed in [5] and [4]. They are both based on the concept of entropy and their focus is to derive implementation-independent measures of the signal activity in the circuit. A number of assumptions are made in both [5] and [4] on how to propagate the entropy of the inputs through the circuits. These methods can be very efficient, though given all the required approximations and the fact that they ignore issues such as glitching implies that these techniques are not very accurate.

Our method follows a bottom-up approach [3], where the model is obtained from an actual circuit implementation. This offers the best level of accuracy. These methods build their models from data points, which consist on a power value for the circuit under some input conditions. From this set of data points, different strategies exist for generating a model that not only fits these data points, but offers the best possible generalization ability.

Lookup tables have been successfully proposed [2]. N-dimensional tables have been used, where each dimension represents an input parameter. Several strategies exist for reducing the number of dimensions and for interpolating among table points. Alternatively, regression can be used to compute the coefficients of an expression [1, 6]. A combination of both of these methods has also been proposed [2].

We believe the model we propose, based on LS-SVMs, is more robust than previously proposed approaches: it is generic, fully-automated, and uses an underlying methodology with properties that have been proven both theoretically and in practice in many different fields. Our results demonstrate just that.

2.2 LS-SVMs

Consider a general problem where we are given N input/output data points, $\{\mathbf{x}_k, z_k\}_{k=1}^{N} \in \mathbb{R}^p \times \mathbb{R}$. These data points follow an unknown function $z(\mathbf{x}) = m(\mathbf{x}) + e(\mathbf{x})$, where $m(\mathbf{x})$ is the target function we wish to estimate and $e(\mathbf{x})$ is a sampling error. Support Vector Machines (SVMs) are a method of obtaining $y(\mathbf{x})$, an estimate of $m(\mathbf{x})$, from the given data set, referred to as training set. SVMs achieve regression by non-linearly mapping the input space into a higher dimensional feature space where a linear approximant hyperplane can be found. This is implicitly made by the use of a kernel function.

A version of a SVM for regression was proposed by Vapnik el al [9]. This method is called support vector regression (SVR). The model produced by SVR only depends on a subset of the training data, because the cost function for building the model ignores any training data that are close (within a threshold ε) to the model prediction. The relevant data points form a set of support vectors and they immediately lead to a sparse representation. On the other hand, computing the model is a Quadratic Programming (QP) problem. To simplify this QP problem, Least Squares Support Vector Machines were introduced [8]. In both methods the model is:

Table 1. Common kernels.

Linear:	$\Psi(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
Polynomial:	$\Psi(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j + \theta)^n$
Exponential (RBF):	$\Psi(\mathbf{x}_i, \mathbf{x}_j) = \exp(\frac{-\lVert\mathbf{x}_i - \mathbf{x}_j\rVert^2}{\sigma^2})$
Hyperbolic Tangent	$\Psi(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\phi \mathbf{x}_i^T \mathbf{x}_j + \theta)$

$$y(\mathbf{x}) = \mathbf{w}^T \varphi(\mathbf{x}) + b \tag{1}$$

The $\varphi(\mathbf{x})$ mapping is usually a non-linear function that transforms the data into a higher dimensional feature space, and is weighted by $\mathbf{w}$. Constant b is the bias term.

LS-SVM correspond to solving the following constrained optimization problem:

$$\min_{\mathbf{w}} J = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\frac{1}{2}\sum_{k=1}^{N} e_k^2 \quad \text{s.t.} \quad z_k = \mathbf{w}^T\varphi(\mathbf{x}_k) + b + e_k \tag{2}$$

The $\mathbf{w}^T\mathbf{w}$ term stands for minimizing the length of the weight vector, while the C constant is the trade-off parameter between the smoothness of the representation and the minimization of training data errors. The number of training samples, known as support vectors, is given by N.

Using Lagrange multipliers, in order to transform the problem into an unconstrained optimization problem, gives:

$$L = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\frac{1}{2}\sum_{k=1}^{N} e_k^2 - \sum_{k=1}^{N} \alpha_k[\mathbf{w}^T\varphi(\mathbf{x}_k) + b + e_k - z_k] \tag{3}$$

Which is guaranteed to have a global minimum when:

$$\frac{\partial L}{\partial \mathbf{w}} = 0; \frac{\partial L}{\partial b} = 0; \frac{\partial L}{\partial e_k} = 0; \frac{\partial L}{\partial \alpha_k} = 0,$$

resulting in the following linear system of equations:

$$\begin{bmatrix} 0 & \mathbf{1}^T \\ \mathbf{1} & \mathbf{\Omega} + C^{-1}\boldsymbol{I} \end{bmatrix} \begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{z} \end{bmatrix} \tag{4}$$

where $\mathbf{\Omega}$ is the kernel matrix, $\Omega_{kl} = \Psi(x_k, x_l) = \varphi(x_k).\varphi(x_l)$, $k, l = 1, \ldots, N$, and Ψ is the kernel function. The resulting LS-SVM is given by

$$y(\mathbf{x}) = \sum_{k=1}^{N} \boldsymbol{\alpha}_k \Psi(\mathbf{x}, \mathbf{x}_k) + b \tag{5}$$

The simplicity of (5) is not without a cost. While SVMs have a built in way of selecting the most significant data points from their training set, LS-SVMs do not. Sparseness is lost due to the usage of all the data points as support vectors (a large N). This adds a new complexity to the problem. It becomes necessary to carefully choose the training points used to effectively cover the input space.

There are many kernels from which to choose from, some of which are shown in Table 1. In this work, we will focus exclusively on the RBF kernel since it has good empirical results and has a nice smooth behavior.

3 Macromodel Tuning

In this section, we describe the method for computing a power macromodel based on LS-SVM. We start by presenting how we generate the data points used as the training set and the error metrics we use. Then, experiments are performed to tune the kernel parameters.

3.1 Input Space Analysis

To generate the desired black-box macromodel it is necessary to obtain a set of data points to be plugged into the LS-SVM, $\{\mathbf{x}_k, z_k\}$. To analyze the performance of the LS-SVM method for power estimation, we need data points where z_k represents the power dissipation of a circuit under inputs $\mathbf{x}_k$. For this purpose, we can either use experimental values obtained from actual circuit measurements, or values computed by a simulator. Naturally, the accuracy of the model will be directly related to the quality of the data points.

Note that there is some flexibility in terms of what $\mathbf{x}_k$ represents. In this work, we are using the switching probability of each of the p inputs to the functional unit, hence a vector of size p with values between 0 and 1. Alternatively, we could aggregate all inputs and have their distribution probability (for example, in the case a set of bits represent some numerical value). Additionally, if specific information is available regarding joint probability distributions, it can be used to bias the choice of data points.

We should also observe that z_k can represent static, dynamic power, or total power. The results we present in the next section were obtained from a logic simulator which only accounts for dynamic power. Yet, the results should easily extrapolate for static, and hence, total power.

We use the following error functions to provide some insight into LS-SVM model's performance:

Relative error: $E_R = \{\frac{|z_k - y(\mathbf{x}_k)|\}}{z_k}\}_{k=1}^{N}$

Average relative error: $E_1 = \frac{1}{N}\sum_{k=1}^{N} E_{R_k}$

Maximum relative error: $E_2 = \max_k E_{R_k}$

Fraction below 10% : $E_3 = \frac{|\{a \in E_R : a < 10\%\}|}{N}$

Ultimately we aim to achieve E_3 above 90% and E_1 smaller than 5%. Though error values are not bounded, E_2 is a good representation of the worst-case scenario.

As data points used during training represent the total knowledge LS-SVMs have for model construction, their selection method is of crucial importance. For each circuit, the objective is to generate N vectors $\mathbf{x}$ of size p with values within $[0, 1]$. To effectively cover the input space three distributions were tested:

1. UNIFORM, 100% of the data points follow an uniform distribution between 0 and 1.
2. NORM, 100% follow a normal distribution with 0.5 mean and a chosen variance γ (every value not contained in the [0,1] interval is resampled).
3. UNMIX, a mix of 1 and 2, 50% follow an uniform distribution and 50% follow a normal distribution.

The Normal distribution was considered since, in practice, circuit input probabilities should be around 0.5 so it makes sense to have a good input space representation of that region.

In Table 2 the errors defined earlier are shown, averaged over a large set of benchmark circuits. We conclude that the difference in error performance between the tested methods is not significant. Since the proposed algorithm (see Section 4) automatically selects training points from the data, sets of 10,000 points were constructed using equal quantities of each of the 5 methods presented, for each of the benchmark circuits.

Table 2. Comparison of input distributions.

Distribution	E_1	E_2	E_3
UNIFORM	4.57	50.3	90.7
UNMIX ($\gamma = 0.1$)	4.21	49.5	91.0
UNMIX ($\gamma = 0.3$)	4.38	47.7	91.0
NORM ($\gamma = 0.1$)	4.82	55.1	88.6
NORM ($\gamma = 0.3$)	4.43	49.1	90.6

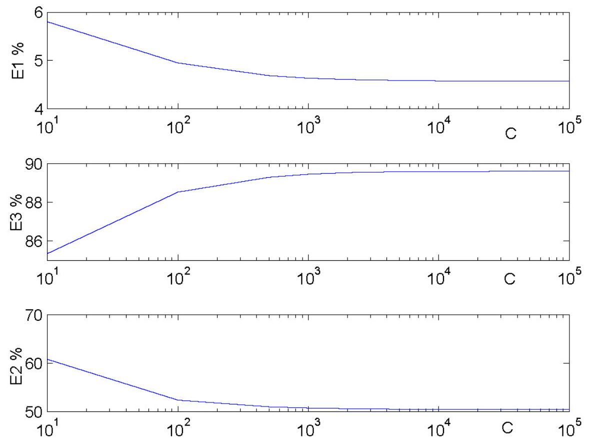

Figure 1. Error variation with parameter C.

3.2 Parameter Analysis

The regression process is not totally automated, since there is a number of parameters that need to be selected, namely C (see (4)), σ (see Table 1) and the number of support vectors. We also have an option as to which kernel to use, however the exponential RBF kernel (Table 1) has been reported to perform particularly well under LS-SVM [7]. The analysis and results presented in the remainder of this paper were all obtained with this kernel.

As referred in Section 2.2, C is a constant that permits a tradeoff between the training error and the smoothness of the model. The training error in sample k is given by $e_k = \frac{\alpha_k}{C}$, indicating that larger values of C force the estimator to reduce the training errors. We have computed the errors obtained using different values of C, maintaining all remaining parameters constant ($\sigma = 2$ and with 2,000 SVs). The graphs in Figure 1

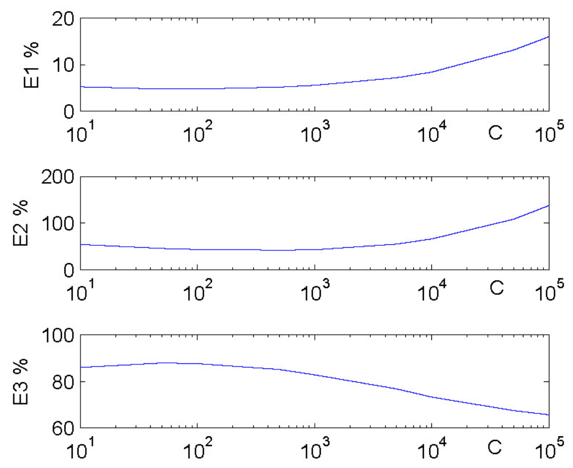

Figure 2. Error variation with parameter C for noisy data (10%).

show the test error variation with C averaged over all the benchmark circuits (given in Table 3). For all the circuits the behavior was similar so it is safe to derive conclusions from the average results. We can observe that the error decreases as C increases, but for values above $C = 10^4$ the error remains almost constant. Hence, both training and test errors benefit from larger C.

A new experiment was devised to show that the optimum C depends on the noise present in the data points. Consider the earlier data set $\{\mathbf{x}_k, z_k\}$ with added noise: $z'_k = (1+\epsilon)z_k$ where ϵ follows a uniform distribution between $[-e; e]$.

Figure 2 shows the test errors for noisy data with maximum noise $e = 10\%$. It becomes clear that the value of C depends on the quality of the data set. We have set $C = 10^4$ since it is approximately optimal for the noiseless case and it is still good when there are small errors present.

In the exponential RBF kernel (Table 1), σ represents a width factor. If it is large, then the influence of each support vector (SV) spreads, smoothing the solution. If σ is small, each SV has a small influence over the space around it, which reduces the information the model has over all the input space. At the extremes, if $\sigma \to \infty$ we obtain a constant function, and if $\sigma \to 0$ our model is only able to estimate input **x** equal to its SVs, having null generalization ability. Training error is then inversely proportional to σ values. Figure 3 presents the error obtained for different values of σ on test data ($C = 10^4$, obtained above, and the same 2,000 number of SVs were used). Tests show that the optimum value should be $\sigma = 1.2$ (the training error is negligibly small for this σ).

To determine the number of SVs, we performed similar experiments with increasing size of the training data set (Figure 4). As it was expected, errors decrease when the size of the training set increases. Since having more SVs translate linearly to the size of the model and its computation time, it is necessary to achieve a tradeoff between model complexity, the number of SVs, and model accuracy. One should be aware that the number of SVs will depend on the shape of power surface (steep functions need more SVs) and the σ value used (smaller sigmas

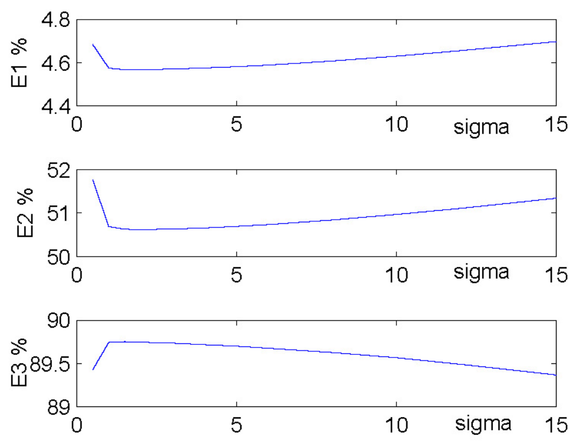

Figure 3. Errors for different values of σ.

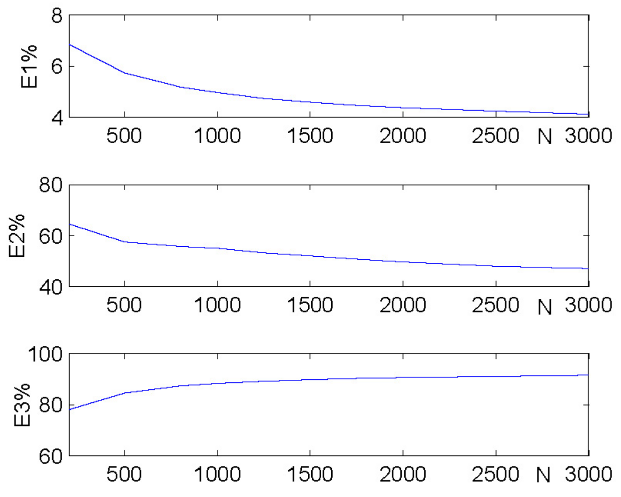

Figure 4. Error dependence on the number of support vectors.

imply more SVs). This motivates the use of an iterative algorithm to add SVs to the training set when there is a need to do so.

3.3 Model Storage and Evaluation

From (5), we know that evaluating the LS-SVM model to compute the power requires the sum of N elements, and each element contains a norm which is a sum of p elements. Hence, the model requires $O(Np)$ operations to compute $y(\mathbf{x})$. This is not a problem for the considered values of N.

In terms of memory, we need to store: N SVs, each of size p; N α values; and the constant b. Hence, the model has $O(N(p+1)+1) = O(Np)$ memory complexity.

If we use the float data type to store these values (usually 4 bytes long), a model of a circuit with 200 inputs ($p = 200$) and $2,000$ SVs ($N = 2000$) will need $(2000 \times (200+1)+1) \times 4 = 1.53MB$ of storage space. It is affordable, but still expensive.

4 Model Optimization

The analysis in the previous section indicates that the best number of support vectors to use and the value of the σ parameter are interdependent. We propose an iterative method to automatically determine these parameters so that we maximize the generalization of the LS-SVM-based power macromodel.

The standard method for computing kernel parameters is simply to solve the linear system given in (4), using a set of data points $\{\mathbf{x}_k, z_k\}_{k=1}^N$, the training set. The model obtained can then be evaluated against a different and disjoint set of data points, the test set. From the errors obtained in each of these sets, we can conclude:

1. large errors on the test set might be due to two reasons: the region of the input space where samples have large test errors is badly covered by the training set; the kernel function is excessively local, which means that the influence of each SV is limited to a very small region around it.
2. large errors on the training set indicate that the kernel function is smoother than it should be, not having enough flexibility to approximate steep surfaces.

The iterative method developed tries to address these two issues in the following manner:

1. to increase the generalization of the model, a given number of data points with largest error is moved from the validation set into the training set.
2. as discussed in the previous section, the σ parameter defines how local or global the impact of each support vector is. Hence, by reducing this value we reduce the error on each training point. This may potentially lead to worst generalization, which is compensated by the data points that are moved from the test to the training set. Nevertheless, σ should not be greatly reduced between iterations.

The method starts by splitting the set of available data points into three sets:

- the final test set, with data points which will only be used to evaluate the final model (hence, after the iterative process has finished)
- an initial training set of size M
- a validation set, which could be much larger than M

A first model is computed solving Equation 4 on the initial training set and the errors on the training and validation set are obtained. If these errors exceed a user-specified value, namely average or maximum error above a certain threshold, the model is recomputed with the following modifications:

- if the validation error exceeded specifications, we add to the training set the k data points in the validation set with largest error
- while the error on the training set exceeds specifications, parameter σ is reduced by s

When all specifications are met the process terminates. Naturally, if these specifications are too stringent, convergence may be difficult. Since the number of data points used in the training set is directly related to the size of the model, a maximum size is defined that, when reached, terminates the iteration process.

Parameters k and s define the granularity of the process. Larger values will reduce the number of iterations, but may lead to a oversized model. The size of the initial training set should be relatively small to give room for the addition of the more problematic data points in the test set, which will contribute to a better generalization.

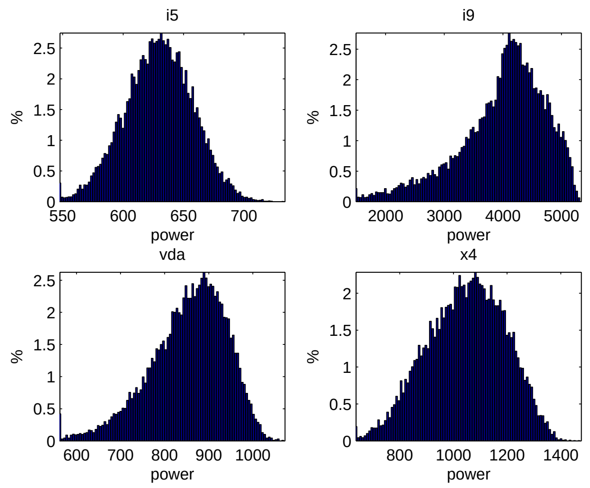

Figure 5. Dynamic Power distribution for uniformly distributed input

5 Results

The specifications for the optimization process were the following:

1. the initial number of data points was $M = 500$, and is increased by $k = 20$ in each iteration;
2. the starting value was $\sigma = 10$, and is reduced when the training error $E1$ is bigger than 1% by $s = 0.75$;
3. iteration stops when $3,000$ SVs are reached or when $E1 \leq 2\%$ and $E2 \leq 30\%$ and $E3 \geq 98\%$

In Table 3 error values are shown for the 4D lookup table method (from [2]), the optimized LS-SVM and the regular LS-SVM with $N = 3,000$, where N is the number of support vectors. The optimized version achieved similar average results as the regular LS-SVM even though most models are smaller ($< N$). For the circuits which could not meet the desired expectations($N = 3000$) there were considerable $E2$ and $E3$ improvements. Compared with the lookup tables, the proposed method obtains, on average, similar $E2$ while reducing $E1$. These values are more circuit dependent. The proposed methodology could be applied only to a subset of circuits to achieve excellent models, such as the ones for $C499$ and $C1355$.

The total CPU time was in the order of minutes for the construction of the models, therefore irrelevant for a characterization step.

To give some insight into the problem we are facing, the dynamic power distribution is plotted in Figure 5 for four different circuits under uniformly distributed input. We observe

Table 3. Comparison with the results of the lookup-table method of [2].

Circuit	Circuit Information			Testing- Optimized				Testing- Non Optimized			4D Lookup Tables	
	Ins	Outs	Nodes	N	E_1	E_2	E_3	E_1	E_2	E_3	E_1	E_2
add_rpl_16	32	16	214	500	1.66	16.9	99.7	1.20	13.0	100.0	-	-
apex6	135	99	411	3000	4.16	25.7	93.5	4.91	28.5	92.0	-	-
i5	133	66	161	500	1.62	13.2	99.9	1.48	8.97	100.0	-	-
i9	88	63	353	3000	7.39	99.8	80.1	8.33	140.6	76.8	-	-
pair	173	137	877	880	2.24	12.1	100.0	2.37	16.3	99.8	-	-
rot	135	107	390	1300	2.50	14.1	99.6	2.58	18.0	99.3	-	-
vda	17	39	304	680	2.14	20.5	99.3	1.70	43.9	98.6	-	-
x3	135	99	332	2240	3.34	19.2	97.7	3.66	25.7	96.7	-	-
x4	94	71	211	2940	3.86	36.8	95.0	5.16	42.3	89.8	-	-
Average:					3.21	28.7	96.1	3.49	37.5	94.8	-	-
C432	36	7	160	1320	2.67	23.1	98.1	2.62	39.3	97.4	4.409	17.35
C499	41	32	202	500	1.28	9.3	100.0	0.53	3.5	100.0	3.95	16.34
C880	60	26	383	1300	2.43	17.8	99.2	2.74	21.9	98.4	3.62	13.97
C1355	41	32	546	500	1.11	7.7	100.0	0.51	4.1	100.0	4.03	15.04
C1908	33	25	880	500	1.92	14.8	99.7	1.10	11.1	100.0	3.73	15.69
C2670	233	140	1193	2340	3.32	24.4	98.1	3.31	23.5	98.0	2.18	10.16
C3540	50	22	1669	860	2.36	16.2	99.5	2.07	21.0	99.2	3.22	15.55
C5315	178	123	2307	900	2.21	13.4	99.7	2.17	12.9	99.6	2.08	12.20
C6288	32	32	2406	500	1.00	12.5	100.0	0.49	9.8	100.0	2.218	17.37
C7552	207	108	3512	3000	3.94	20.0	95.5	4.07	27.8	94.3	2.65	14.32
Average:					2.22	15.9	98.9	1.96	17.5	98.7	3.27	14.9

that circuit i5 only requires a very simple model since the power dissipated ranges over a small interval. For circuits vda and x4 we have slightly larger power ranges and an approximately constant variance. The power dissipated in circuit i9 spans over a large interval and does not resemble a Gaussian distribution. The error values obtained in Table 3 show that i5 is indeed simple to model, vda and x4 are more challenging but still accurately modeled and i9 is very hard to model. It is obvious that the smaller the power range is, the better the models are. On the other hand, constant variance seems to make modelling easier.

6 Conclusions

Least Squares Support Vector Machines were successfully applied to power macromodeling of digital circuits, achieving an average error of about 2%. A comprehensive study of the training parameters was presented and optimal values were calculated for the power estimation problem. From this study an iterative optimization algorithm was derived which allows for automatic selection of the size of the model required to meet desired error specifications. Model performance was validated on a large set of benchmark circuits. Future work involves tweaking the optimization parameters and integrating feature selection methods into the iterative algorithm.

References

[1] A. Bogliolo, L. Benini, and G. D. Micheli. Regression-Based RTL Power Modeling. *ACM Transactions on Design Automation of Electronic Systems*, 5(3):337–372, 2000.

[2] S. Gupta and F. Najm. Power Modeling for High-level Power Estimation. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 8:18–29, 2000.

[3] E. Macii, M. Pedram, and F. Somenzi. High-level Power Modeling, Estimation, and Optimization. *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, pages 1061–1079, 1998.

[4] D. Marculescu, R. Marculescu, and M. Pedram. Information Theoretic Measures of Energy Consumption at Register Transfer Level. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 81–86, Apr. 1995.

[5] F. Najm. Towards a High-Level Power Estimation Capability. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 87–92, Apr. 1995.

[6] A. Raghunathan, S. Dey, and N. Jha. High-level Macro-modeling and Estimation Techniques for Switching Activity and Power Consumption. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 11(4):538–557, Aug. 2003.

[7] W. Shang, S. Zhao, and Y. Shen. Application of LSSVM with AGA Optimizing Parameters to Nonlinear Modeling of SRM. *Industrial Electronics and Applications, 3rd IEEE Conference on*, pages 775–780, June 2008.

[8] J. Suykens, T. V. Gestel, J. D. Brabanter, B. D. Moor, and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific Pub., 2002.

[9] V. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.