Adaptive Least Mean Square Behavioral Power Modeling [*]

Alessandro Bogliolo Luca Benini[†] Giovanni De Micheli[†]

DEIS - University of Bologna
Bologna - I 40136

[†]CSL - Stanford University
Stanford - CA 94305-9030

Abstract

In this work we propose an effective solution to the main challenges of behavioral power modeling: the generation of models for the power dissipation of technology-independent soft macros and the strong dependence of power from input pattern statistics. Our methodology is based on a fast characterization performed by simulating the gate-level implementation of instances of soft macros within the behavioral description of the complete design. Once characterization has been completed, the backannotated behavioral model replaces the gate-level representation, thus allowing fast but accurate power estimates in a fully behavioral context.

Our power characterization procedure is a very efficient process that can be easily embedded in synthesis-based design flows. No additional effort is required from the designer, since power characterization merges seamlessly with a natural top-down design methodology with iterative improvement. After characterization, the behavioral power simulation produces accurate average and instantaneous power estimates (with errors around 7% and 25%, respectively, from accurate gate-level power simulation).

1 Introduction

As power dissipation becomes a critical cost metric for VLSI systems, reliable power estimates are needed as early as possible in the design flow. Accurate power estimation techniques have been proposed [1], but they operate at the circuit [2] or gate [3, 4] level and may require excessive computational resources to provide power statistics on large digital systems.

An even more severe challenge is given by the lack of detailed information on the circuit structure in the early phases of the design process. During behavioral specification, the designer does not have information on the actual implementation that will satisfy the constraints. On the other hand, CAD tools are increasingly effective. Automatic synthesis of designs specified at the behavioral level is becoming common practice [5, 6, 7]. Behavioral synthesis offers many advantages: it allows shorter design time and enables the exploration of design tradeoffs.

As far as power dissipation is concerned, several efforts have been made to provide power estimation at the behavioral level [8, 9, 10, 11]. All methods proposed in the literature rely on the assumption that a library of *primitive functions* is available during behavioral synthesis. The primitives (also known as *hard macros*) are circuit-level (or gate-level) blocks implementing behavioral operators such as addition or multiplication. The knowledge of both the inner structure and the functionality of a primitive can be exploited to construct models of its power consumption.

The main limitation of such characterization methodologies is that they assume the knowledge of the gate-level or circuit level structure of the primitives before the design is implemented. Unfortunately, this assumption is not always valid. While libraries of primitives are always available in behavioral synthesis, they are often specified as *soft macros*. A soft macro is a functional block for which a *synthesizable* HDL description is provided, but its gate-level implementation depends on the technology library of elementary gates used by the designer. When synthesis is performed, the soft macro is synthesized using the technology library available, and its gate-level description is generated on the fly. Intuitively, soft macros resemble software libraries written in a high-level programming language, while traditional hard macros are similar to binary libraries.

In this work we propose a *power estimation methodology* that is particularly suited for soft macros used in a behavioral synthesis environment. We refer to a top-down design flow with iterative improvements.

[*]Partially supported by NSF under contract MIP-9421129 and by AEI under grant De Castro.

The designer starts with a behavioral specification (in a HDL such as Verilog or VHDL), a library of soft macros, a technology library of elementary gates and a set of patterns to be used for the validation of the behavioral specification.

Fast synthesis and simulation steps are iteratively performed to explore the design space. Our approach consists of constructing behavioral power models for the instances of the soft macros as soon as they are synthesized. This is done by an automatic characterization procedure that runs within each simulation step. The design is simulated at the behavioral level, with the only exception of the last synthesized blocks, for which the gate-level descriptions are simulated to extract information about their power consumption. Power consumption data are concurrently collected for all macro instances under characterization. As soon as enough data are collected to characterize the power model for an instance of a macro, its gate-level description is no longer simulated. We then *switch back to its backannotated behavioral model*, therefore progressively speeding-up the simulation as more instances are characterized. In subsequent iterations, behavioral power models do not need to be recharacterized (unless the corresponding blocks have been directly affected by optimization). They are directly used to backannotate the corresponding behavioral specifications, thus exploiting the efficiency of behavioral power simulation for design exploration.

We model the power consumption of a complex combinational block by means of a linear combination of its input-output switching activities. Model characterization consists of finding an assignment for the coefficients that minimizes the mean square error made by the linear power estimate. The exact solution to this optimization problem can be found by means of *least squares fitting* techniques, that may become expensive in terms of CPU and memory requirements (all the data involved in the characterization process are to be available at the same time and a large system of linear equations is to be solved). To relax computational requirements we use a characterization algorithm based on a well known technique of adaptive signal processing, known as *least mean squares* (LMS) algorithm [12]. LMS has very mild computational costs and its key advantage is that it converges to the exact mean square model but it does not require collection of large amounts of data and can be performed on-line, while the simulation is running.

Notice that input spatio-temporal correlations are automatically taken into account by running the instances under characterization within the behavioral simulation of the entire design (*i.e.*, *in situ*). This is a key point: in general the LMS algorithm converges to different models for different instances of the same macro, even if the implementation is exactly the same. Each model is the best least squares linear fit to the power dissipation of the instance given the input patterns that are actually provided by the environment.

Nevertheless, *caching mechanisms* across multiple instances can be used either to provide first-cut estimates or to speed-up the characterization process. The behavioral power model of a previously characterized instance can be used as the starting point to construct the power model of new (or newly modified) instances of the same macro.

Our behavioral modeling strategy provides average and instantaneous power estimates with accuracy around 7% and 25% (respectively) from accurate gate-level power simulation (performed by PPP [3], the same simulator used for characterization). Although the accuracy loss is sizable, we believe that the generality and ease of use of our methodology can be very useful for steering the behavioral synthesis process and enables the power simulation of large systems.

2 Power estimation with LMS

We restrict our analysis to soft macros implemented by complex combinational logic blocks (hereafter called *units* for brevity). Consider a unit with n inputs and m outputs. Consider two time instants t_1 and t_2, $t_1 < t_2$. Assume that the unit is stable (*i.e.*, not affected by transient phenomena) at t_1 and t_2, and that an input transition occurs in the time interval $[t_1, t_2]$. We use $p(t_1, t_2)$ to denote the power consumption of the unit in the time interval $[t_1, t_2]$. Our goal is to find a *black-box model* of the power dissipation using only boundary information (*i.e.*, the knowledge of input-output transitions).

Least squares fitting

We approximate the power dissipation in the unit by means of a linear regression model based on its input-output activity. The input (output) activity is represented by a vector of Boolean variables $\mathbf{i} = (i_1, i_2, ..., i_n)$ ($\mathbf{o} = (o_1, o_2, ..., o_m)$) taking value 1 when there is a transition on the corresponding input (output) signal. In symbols, our power estimate is

$$p = c_0 + c_1 i_1 + ... + c_n i_n + c_{n+1} o_1 + ... + c_{n+m} o_m \quad (1)$$

where $\mathbf{c} = (c_0, c_1, ..., c_{n+m})$ are the fitting coefficients to be determined during characterization.

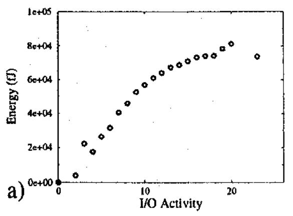
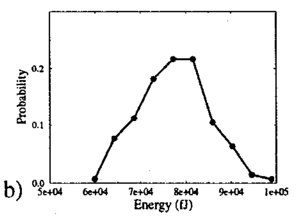
a)  b) 

Figure 1: a) Correlation between the I/O activity of an 8-bit carry-lookahead adder and its energy-per-cycle consumption. b) Bell-shaped distribution of the energy consumption of the same circuit due to input transitions corresponding to the same activity vectors (namely, $\mathbf{i} = (0011001100011011), \mathbf{o} = (00101001)$).

The rationale behind this model is simple: when there is little input-output activity the power dissipation is likely to be small, while high input-output switching activity usually implies high power dissipation in the internals of the block.

Obviously, the linear model is only a rough approximation: power dissipation is *not* exactly a linear function of input-output activity. A typical behavior is shown in Figure 1.a where the power dissipation is plotted as a function of the total input-output activity (*i.e.*, the number of inputs and outputs switching) for an eight-bit carry-lookahead adder. It is evident that in this case there is good correlation between power consumption and input-output activity. Moreover, Figure 1.b shows the distribution of power dissipation obtained for several input transitions corresponding to the same **i** and **o** (remember that a given value of the input switching activity can be produced by 2^n different input transitions). The bell-shaped curve closely resembles a Gaussian noise distribution on the estimated value p. This is an important observation, because least squares models maximize the probability that the predicted value of p is equal to the real value when the error in the estimation can be modeled as Gaussian noise.

To determine the coefficients of Equation (1) we need a *sample* of input-output activities and corresponding power consumption. The sample of data collected during the characterization phase can be represented by a pair $(\mathbf{X}, \mathbf{p})$. If s is the sample size, $\mathbf{X}$ is an $s \times (n + m + 1)$ Boolean matrix containing the values taken by the independent variables during characterization (its k-th row being $\mathbf{x}^k = (1, i_1^k, i_2^k, ..., i_n^k, o_1^k, o_2^k, ..., o_m^k)$), while $\mathbf{p}$ is a vector of size s containing the corresponding values of the dependent variable (the k-th element being p^k) obtained from accurate gate-level power simulation.

Given a sample $(\mathbf{X}, \mathbf{p})$, coefficients $\mathbf{c}$ are the unknown of the following system of linear equations:

$$\mathbf{p} = \mathbf{X}\mathbf{c}. \qquad (2)$$

Due to the statistic nature of the characterization process, the sample size must be significantly larger than the number of parameters to be characterized. Hence, matrix $\mathbf{X}$ has much more rows than columns and the linear system is overdefined. The vector $\mathbf{c}$ giving the minimum mean square error among all possible linear estimators of $\mathbf{p}$ can be obtained from (2) using well-known techniques of least squares fitting [12]. An important property of the least squares linear model is that it always produces an estimate of p with the same average value as the average value of p in the sample used for fitting. Therefore it is guaranteed to perform at least as well as an average value approximation.

Finding a correct sample is the key issue in the application of regression models. If the input sample is not representative of the actual input values fed to the unit during its operation, the model will produce incorrect estimates: Landman *et al.* [8] showed that using uniform white noise input samples can lead to large errors even on average power estimates if the input distribution is not uniform and white. For this reason, the unit should be characterized within its environment. If the unit is characterized while running a global behavioral simulation the input patterns provided to it are the *actual patterns* that the unit will observe during operation. Notice that in this case multiple instances of the same unit will generally have different power models, because the input samples will generally be different.

Unfortunately, the determination of coefficients $\mathbf{c}$ requires the solution of a linear system of equations with $n + m + 1$ unknowns and as many equations as the sample size s (that may be very large). Moreover, determining the sample size is not a straightforward task. In the general case of stationary signals, it may require to solve the least squares system multiple times and test for convergence.

The LMS algorithm

If several units are in the characterization phase, the computational burden of solving multiple systems of equations multiple times will slow down the simulation and impose severe memory requirements (all the matrices and vectors have to be stored). A solution to these problems is offered by the LMS algorithm [12]. In synthesis, LMS is a gradient search technique that iteratively modifies the coefficients of Equation (1) and adaptively tries to minimize the least squares error produced by the model, until it reaches convergence in a neighborhood of the theoretical minimum error solution (*i.e.*, the least squares solution). An accurate description of LMS can be found in [12], here we simply outline the algorithm and the reasons of its usefulness in our case.

The iterative formula used for updating the coefficient vector is the following:

$$\mathbf{c}^{k+1} = \mathbf{c}^k + 2\mu\epsilon^k\mathbf{x}^k \tag{3}$$

where k is the iteration step, $\mathbf{c}^k$ is the current assignment of the fitting coefficients, μ is a fixed constant (to be discussed later), $\mathbf{x}^k$ is the input-output transition vector (*i.e.*, the k-th row of $\mathbf{X}$) and $\epsilon^k = p^k - p^k_{est}$ is the difference between the power actually dissipated corresponding to transition $\mathbf{x}^k$ and the power estimated by the model for the same transition. Intuitively, at each iteration the algorithm tries to modify the coefficients in order to reduce the error made by the model.

The initial value ($\mathbf{c}^0$) of the coefficient vector does not change the asymptotic properties of convergence. However, convergence will take less iterations if $\mathbf{c}^0$ is close to the optimum value. This property can be exploited to speed-up the re-characterization process when some instance in the system is marginally modified by optimization.

The convergence of the LMS algorithm is controlled by parameter μ. It can be shown that for convergence the following condition must hold:

$$0 < \mu < \frac{1}{tr[\mathbf{R}]} \tag{4}$$

where $tr[\mathbf{R}]$ is the trace of the *correlation matrix* $\mathbf{R}$ [12]. Element r_{ij} of $\mathbf{R}$ represents the correlation between the i-th and the j-th independent variables (*i.e.*, $r_{ij} = E[x_i x_j]$). Since in our case the independent variables may only take value 0 or 1, the elements of $\mathbf{R}$ cannot be greater than 1. An upper bound for the trace of $\mathbf{R}$, is then $tr[\mathbf{R}] \leq n + m + 1$.

Even if convergence can be easily ensured, a more subtle tradeoff is involved in the choice of μ. If the convergence of the iteration is too fast, the accuracy of the final solution can be compromised. In [12] a measure of the accuracy of the solution is defined: the *misadjustment* M. LMS tries to find the optimum $\mathbf{c}$ by minimizing the *mean square error* (MSE) of the linear model. However, due to the non-linearity of the dependent variable, the minimum MSE ($minMSE$) is always greater than 0. The misadjustment is an adimensional measure of the distance between the current solution and the best one: $M = (MSE - minMSE)/minMSE$. It can be shown that the value of M at the end of the adapting process is estimated by $M = \mu \cdot tr[\mathbf{R}]$.

On the other hand, μ is directly proportional to the speed of convergence, the time constant being $T_{MSE} = \frac{1}{4\mu\lambda_n}$, where λ_n is the smallest eigenvalue of $\mathbf{R}$ [12]. It is apparent that the choice of μ is paramount for obtaining a satisfying regression model with LMS. In our implementation μ si chosen as one tenth of the (worst case) maximum allowed value. The user can however override this choice either by specifying μ or by specifying the maximum number of patterns for which the characterization must be run.

LMS has numerous advantages over the least squares solution, while retaining the same desirable properties of robustness and statistical significance. First, the computational requirements are mild: the iteration formula has complexity linear in the number of inputs and outputs. Second, no additional storage of past data is required. Third, the characterization process can be performed on the fly, while the simulation is running, and several units' instances can be characterized at the same time without sensible performance penalty.

3 On-line characterization

The adaptive power model described so far actually bridges the gap between fast behavioral simulation and accurate gate-level power estimate. To do this, a simulation engine supporting both levels of abstraction is required. Moreover, a straightforward interface to the synthesis environments is necessary to allow iterative design improvements. To meet these requirements, we implemented the procedures for behavioral power characterization and estimation as additional features of PPP [13], an unified synthesis and simulation tool based on Verilog-HDL.

The basic simulation engine of PPP is Verilog-XL, that parses the hierarchical description of the network and performs event-driven logic simulation. Routines for gate and behavioral level power characterization/evaluation have been implemented in C and integrated into the logic simulator using the programming language interface of Verilog-XL.

The gate-level power model used in PPP has been presented in [3]. It allows accurate and efficient power estimations, with accuracy within 5% from Spice even for local single-pattern estimates. The adaptive characterization procedure of the behavioral power model is described next.

The characterization procedure

We refer to the simple situation of Fig. 2, in which two instances of the same soft macro (namely, an 8-

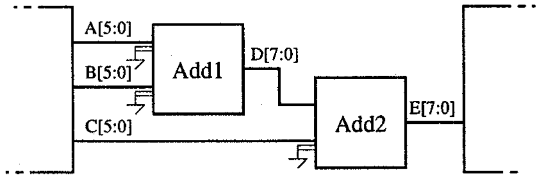


Figure 2: Example design.

bit adder) are to be synthesized in the context of a larger design. We choose a carry-lookahead trial implementation for the adders and we run the adaptive modeling procedure to build the power consumption model of the two units. In the Verilog description, each unit is represented by a module containing both the gate-level implementation of the carry-lookahead and a task calling the LMS routine.

During characterization, the gate-level implementation of the two units is accurately simulated in the context of a behavioral simulation of the entire design. In this way, typical input vectors are automatically generated for Add1 and Add2 as a typical test sequence is applied at the primary inputs of the design.

Suppose that the circuit is to be used at a frequency of 50Mhz. The characterization phase then consists of applying a sequence of input patterns with a 20ns time period. According to Equation (3), the k-th *training step* (*i.e.*, the k-th sample) for the model of a unit is based on its switching activity $\mathbf{x}^k$ at time $k \cdot 20$ns and on its power consumption p^k in the time period $[k \cdot 20\text{ns}, (k+1) \cdot 20\text{ns})$. Both $\mathbf{x}^k$ and p^k are available at the end of the time period, when all transient phenomena are extinguished: the value of p^k is provided by the gate-level simulation of the implementation, while $\mathbf{x}^k$ is obtained as the exclusive OR between the current and previous values of the I/O vectors.

During characterization, the adaptive procedure is automatically called at the end of each time period for each unit in the design. At the first call, a linear model is created having an independent variable for each input/output signal and a default vector of coefficients, $\mathbf{c}^0$. Since the initial assignment of the fitting coefficients does not impact the final model, we use 0 as default value for the elements of $\mathbf{c}^0$. Notice that in an iterative design flow, the coefficients of an already characterized model can be used as initial guess to speedup subsequent re-characterizations after marginal design improvements.

At subsequent calls, Equation (3) is used to update coefficients. The characterization process terminates when the model has reached convergence (*i.e.*, when the coefficients do not change significantly over several iterations). We use $20 \cdot (1+n+m)$ iterations as a conservative estimate of the convergence time, but different sample sizes can be specified by the user. The final assignment of $\mathbf{c}$ is stored in a file in order to be used for subsequent power analysis or re-characterizations.

In our example situation, both units are simultaneously characterized and their modeling processes take the same time. In general, however, different units require different characterization times. Whenever the model of a unit has reached convergence, its gate-level implementation is automatically disabled and replaced by the behavioral model to speedup simulation.

Furthermore, the overall size of the units under characterization may exceed the limiting size for gate-level simulation. This may happen either because the overall design has been synthesized at the same time, or because several synthesis tasks have been performed before running characterization. In both cases, the set of units can be partitioned and the characterization step repeated for each subset. In our implementation, a control mechanism is available that automatically selects and characterizes clusters of units without stopping simulation. This is done by switching on the fly between gate and behavioral representations.

In summary, adaptive characterization allows us to find the best linear estimator for the power consumption of a unit in the context in which it operates. This means that model coefficients are automatically set in order to realize the best fit of the dependent variable corresponding to those configurations of the independent ones that really occur in practice.

Example 1 *For the design of Fig. 2, we obtained completely different estimators for Add1 and Add2, leading to an overall root mean square error of 18% on pattern dependent power estimates. To check the consistence of this result we tried to use for both units the model obtained for Add1 (Add2). The overall mean-square error became 25% (35%).*

4 Results and conclusions

We tested our method running two sets of experiments. We first checked the advantage of using a linear model instead of a constant (pattern-independent) estimator. For this experiment we used combinational benchmarks extracted from the MCNC suite as well as gate-level implementations of logic and arithmetic functions. Circuits were mapped on a cell library with accurate power models and simulated with PPP [3].

For each circuit, a random sequence of $20(n+m+1)$ test patterns with 50% signal and transition probabilities was used to perform characterization. The average power consumption measured during the characterization phase was taken as a constant estimator, while the LMS algorithm was used to characterize the linear model of power consumption. Accuracy was tested running concurrent gate-level and behavioral-level simulations. Two different test sequences (of 200 vectors each) were used: the first one with the same input statistics of that used for characterization, the second with a lower average input activity (of 20% instead of 50%).

Experimental results are reported in Table 1. The first 2 columns contain the name of the circuit and the number of I/O signals. The last 4 columns compare the accuracy of the two models in terms of relative root mean square error ($rmse = \sqrt{mse}/p_{avg}$) and relative error on the average estimate ($avge = |p_{avg} - p_{avg}^{est}|/p_{avg}$). For each benchamrk, results on the first row refer to the first test sequence with 50% input activity. The two models have almost the same accuracy on average power estimates, but the linear one shows lower values of *rmse* since it follows the pattern-dependent instantaneous behavior of power consumption. The second row refers to the second test sequence with lower input activity. Though the accuracy of both models is impaired by the change of input statistics, it is evident that the linear estimator is much more flexible than the constant one.

A second set of experiments was run to verify the effectiveness of the adaptive characterization procedure performed *in situ*. To this purpose, a more complex design was used: a fully functional high-performance IEEE standard floating point adder/subtractor in double precision described in Verilog HDL.

The design was composed of four units: the mantissa datapath (53 bit wide), the exponent datapath (11 bit wide), the rounding logic and the control logic. The adder was designed to perform an addition/subtraction per clock cycle, thus it was not pipelined. The design was built starting from a library of behavioral primitives such as adders, subtractors, multiplexers and comparators. To perform our test, we fully synthesized the exponent datapath using Synopsys and DesignWare cells, with a small gate library characterized for gate-level power simulation in PPP [3]. The design has several internal signal reconvergences and control inputs. Not only the uniform white input distribution hypothesis was not valid within the block, but we could not even assume any simple distribution (such as that proposed in [8]) for the numerous control inputs.

Using PPP, the full floating point adder was simulated at the behavioral level, with the exception of the exponent logic under characterization, for which the gate-level implementation was simulated. The input patterns provided to the adder were taken from those used for testing the functional correctness of the design. The on-line characterization of all instances in the exponent datapath was performed concurrently in less than 30 minutes on a DEC5000/260 workstation.

At the end of the characterization process, the entire floating point adder was simulated at the behavioral level, with a different sequence of input patterns, using the backannotated power models for all units in the exponent logic.

Table 2 reports the accuracy of the power model of each instance, with respect to accurate gate-level estimates provided by the same simulator used for characterization. Results are reported in terms of relative root mean square error (*rmse*) and relative error on average estimate (*avge*). The accuracy of behavioral power models characterized off-line (with uniform white inputs) is also reported.

The quality of the adaptive models characterized *in situ* is substantially higher than that of the off-line models: the average improvement is of 35% on *rmse* and of 75% on *avge*. Notice also that error compensation is not improving the overall accuracy of the estimates. For the entire exponent logic, we obtained $rmse = 42.3$ and $avge = 34.6$ using behavioral models characterized off-line, and $rmse = 22.9$ and $avge = 6.1$ using on-line characterization.

The gate-level simulation of the entire design with the backannotated power models took less than 100 seconds on the same machine mentioned above (for 1000 test patterns). More than 2 hours were spent to run the equivalent gate-level simulation.

These results show the accuracy and efficiency of our approach on circuits of practical interest (the specification of the floating point adder is available upon request for benchmarking).

Conclusions

In this work we have described a novel approach to behavioral power characterization and simulation. Our methodology integrates well with design flows based on behavioral synthesis tools, because it provides a viable solution to the problem of modeling technology-independent soft macros, for which no power characterization procedures have been proposed in the past.

Power is a strongly pattern dependent function. Input statistics greatly influence instantaneous and average power. We solve the pattern dependence problem by characterizing *in situ* the mapped instances of the soft macros. In this way we generate linear

Table 1

Circuit		Const. model		Lin. model	
Name	$n+m$	*rmse*	*avge*	*rmse*	*avge*
cmb	20	35.5	2.2	20.7	2.1
		81.4	78.3	40.6	30.4
decod	21	68.6	7.5	38.9	6.0
		206.0	179.7	54.2	8.1
alu2	16	70.8	1.8	24.4	1.9
		56.4	53.9	24.4	2.5
c17	7	62.4	7.5	34.3	6.6
		195.1	178.8	55.3	29.8
c432	43	37.4	5.1	21.7	4.2
		66.1	45.2	29.6	16.4
c1908	58	18.5	13.2	17.4	12.9
		42.5	40.8	39.6	37.2
cmp	23	23.2	3.3	14.5	3.5
		98.6	93.8	26.2	16.9
sqrt	13	42.0	3.6	25.8	1.1
		83.8	66.9	30.9	10.4
fastdiv	26	46.4	4.9	28.8	5.1
		80.3	20.1	59.1	16.7

Table 2

Unit			Off-line model		On-line model	
Macro	$n+m$	Instance	*rmse*	*avge*	*rmse*	*avge*
CMPXX	23	AZero	23.8	11.3	8.1	0.2
		AMax	24.8	8.9	14.6	2.4
		BZero	23.1	10.2	8.2	3.1
		BMax	25.0	1.1	14.7	2.3
		EQCmp	21.5	11.6	15.9	5.3
		ResZ1	61.1	60.7	38.5	9.2
		ResZ2	60.8	58.4	41.0	6.1
		ResM1	59.4	57.6	39.9	7.4
		ResM2	63.9	55.5	37.3	7.0
CMPGT	26	GTCmp	18.1	2.5	15.4	1.7
EXPSBS	38	SubA-B	23.2	7.1	17.9	3.6
		SubB-A	24.8	6.9	17.7	1.6
		ExpAdj1	59.6	22.3	39.1	10.1
INC	24	ExpIncl	61.9	50.8	38.7	14.3
		ExpInc2	66.2	65.8	41.3	7.5
MUX21	35	ExpFSel	72.3	69.4	48.4	15.6
		ExpDSel	35.2	14.9	33.3	14.3
		ExpTSel	44.2	43.4	22.6	10.2
MUX51	71	ExpFSel	79.6	77.1	39.6	7.6
Entire exponent logic			42.3	34.6	22.9	6.1

Table 1: Comparison between constant and linear power estimators. For each benchmark, estimates reported on the first row have been obtained with the same input statistics used for characterization, estimates on the second row have been obtained with completely different input statistics. Table 2: Experimental results on the 19 units in the FP exponent data path.

regression models that approximatively take into account not only the transition activity at the inputs and outputs, but also the input arrival times and the spatio-temporal correlation of the data.

We leverage the computational efficiency of the LMS algorithm to enable concurrent fast characterization of multiple macro instances, and we exploit the iterative nature of the algorithm to minimize the memory overhead for model building. Moreover, we obtain safe convergence conditions that guarantee small deviation of our models compared to ideal least square fitting.

Although our results are positive, more work is required to improve the accuracy of the linear model on the instantaneous power estimates. We are investigating non-linear fitting techniques that may produce more accurate pattern-dependent power models with improved instantaneous accuracy.

References

[1] F. Najm, "A survey of power estimation techniques in VLSI circuits," *IEEE Transaction on VLSI Systems*, vol. 2, no. 4, pp. 446–455, 1994.

[2] C. Huang, B. Zhang, *et al.*, "The design and implementation of Powermill," in *Proc. of IEEE Symp. on Low Power Electronics*, pp. 105–110, 1995.

[3] A. Bogliolo, L. Benini, and B. Riccò, "Power Estimation of Cell-Based CMOS Circuits," in *Proc. of Design Automation Conf.*, pp. 433 – 438, 1996.

[4] B. J. George *et al.*, "Power analysis and characterization for semi-custom desing," in *Proc. of Int.l Workshop on Low Power Design*, pp. 215–218, 1994.

[5] R. Camposano and W. Wolf, *Trends in High-Level Synthesis.* Kluwer Academic Publishers, 1991.

[6] G. De Micheli, *Synthesis and optimization of digital circuits.* McGraw-Hill, 1994.

[7] D. Knapp, T. Ly, D. MacMillen, and R. Miller, "Behavioral Synthesis Methodology for HTL-based Specification and Validation," in *Proc. of Design Automation Conf.*, pp. 286–292, 1995.

[8] P. Landman and J. Rabaey, "Architectural power analysis, the Dual Bit Type method," *IEEE Transaction on VLSI Systems*, vol. 3, no. 2, pp. 173–187, 1995.

[9] R. S. Martin and J. Knight, "Power-Profiler: optimizing ASICs power consumption at the behavioral level," in *Proc. of Design Automation Conf.*, pp. 42–47, 1995.

[10] D. Liu and C. Svensson, "Power consumption estimation in CMOS VLSI chips," *IEEE J. of Solid State Circuit*, vol. 29, no. 6, pp. 663–670, 1994.

[11] L. Benini, A. Bogliolo, M. Favalli, and G. De Micheli, "Regression models for behavioral power estimation," *to appear in Proceedings of PATMOS*, 1996.

[12] B. Widrow and S. D. Stearns, *Adaptive Signal Processing.* Prentice-Hall, 1985.

[13] L. Benini, A. Bogliolo, and G. De Micheli, "Distributed EDA tool integration: the PPP paradigm," *in Proceedings of ICCD*, 1996.