Advanced Power Estimation Techniques

MASSOUD PEDRAM
University of Southern California
Department of Electrical Engineering – Systems
Los Angeles, CA 90089
massoud@zugros.usc.edu

This chapter describes a methodology and techniques for power estimation and analysis at behavioral, register-transfer and gate levels of design abstraction. The major components of the proposed methodology are survey sampling techniques, probabilistic compaction techniques, RTL co-simulation engine, power macro-modeling, and high-level power estimation.

1 Statistical Sampling

Existing power estimation techniques at the gate level and the circuit level can be classified into two classes: *static* and *dynamic*. Static techniques [21, 5, 15, 13] rely on statistical information (such as the mean activity of the input signals and their correlations) about the input stream to estimate the internal switching activity of the circuit. While these are very efficient, their main limitation is that they cannot accurately capture factors such as slew rates, glitch generation and propagation, DC fighting, etc. Dynamic techniques [7, 3] explicitly simulate the circuit under a "typical" input vector stream. They can be applied at both the circuit and gate levels. Their main shortcoming is however that they are very slow. Moreover, their results are highly dependent on the simulated sequence. To alleviate this dependence and thereby produce a trustworthy power estimate, the required number of simulated vectors is usually high, which further exacerbates the run time problem.

To address this problem, a Monte Carlo simulation technique was pro-

posed in [1]. This technique uses an input model based on a Markov process to generate the input stream for simulation. The simulation is performed in an iterative fashion. In each iteration, a vector sequence of fixed length (called sample) is simulated. The simulation results are monitored to calculate the mean value and variance of the samples. The iteration terminates when some stopping criterion is met. This approach suffers from four major shortcomings. First, since the simulation vectors are generated internally based on statistics of the input stream, a large number of vectors needs to be examined to extract reliable statistics. Second, when the vectors are regenerated for simulation, the spatial correlations among various inputs cannot be adequately captured, which may lead to inaccuracy in the power estimates. Third, the required number of samples, which directly impacts the simulation run time, is approximately proportional to the ratio between the sample variance and square of sample mean value. For certain input sequences, this ratio becomes large, thus significantly increasing the simulation run time. Finally there is a general concern about the normality assumption on the sample distribution. Since the stopping criterion is derived based on the normality assumption, if the sample distribution significantly deviate from normal distribution, the simulation may terminate prematurely. Difficult distributions that cause premature termination include bi-modal, multi-modal, distributions with long or asymmetric tails.

The power estimation problem is addressed from a survey sampling perspective in [4]. The authors assume a sequence of vectors are provided to estimate the power consumption of a given circuit with certain statistical constraints, such as error and confidence levels. The power estimation problem is then transformed into a survey sampling problem by dividing the vector sequence into small units, e. g. consecutive vectors, to constitute the population for the survey. Power consumption is the characteristic under study. The average power consumption is estimated by simulating the circuit by a number of samples drawn from the population, a procedure referred to as sampling, using an accurate circuit- or gate-level simulator such as Power-Mill or Verilog-XL. The objective is to design a sampling procedure that will significantly reduce the number of simulated vectors while satisfying the given error and confidence levels.

The efficiency of simple random sampling (similar to what is used in [1]) is not very high. This can be explained in the context of power estimation as follows. Consider the case where the distribution of the population

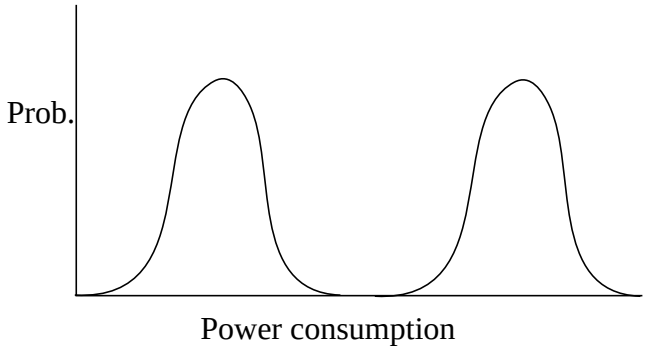


Figure 1: An example where simple random sampling performs poorly.

characteristic (i. e. , power consumption) is as shown in Figure 1 – that is, assuming the characteristic of half of the population is distributed around the right peak, and the characteristic of the other half is distributed around the left peak (the population is bi-modal). Since the selected unit could either come from the right peak or the left peak, the sampling variance is very high.

If, on the other hand, one divides the population into two halves, those units whose characteristic values are around the right peak are put into one subpopulation while the remaining units are put into the other subpopulation, and select the samples in such a way that half of the units in a sample are selected from each of the subpopulations, then the sampling variance will be significantly reduced. In order to divide the population into subpopulations, a predictor is often used. This predictor need not have a linear relationship with the characteristic under study, as it is only used to divide the population into subpopulations and is not directly used to calculate the power estimates. The authors of [4] propose a more efficient sampling procedure based on this scheme called *stratified sampling*.

Stratified sampling techniques have been widely used for surveys because of their efficiency. The purpose of stratification is to partition the population into disjoint subpopulations so that the power consumption characteristic within each subpopulation is more homogeneous compared to the original population. The partitioning is based on a low-cost predictor that needs to be efficiently calculated for each member in the population.

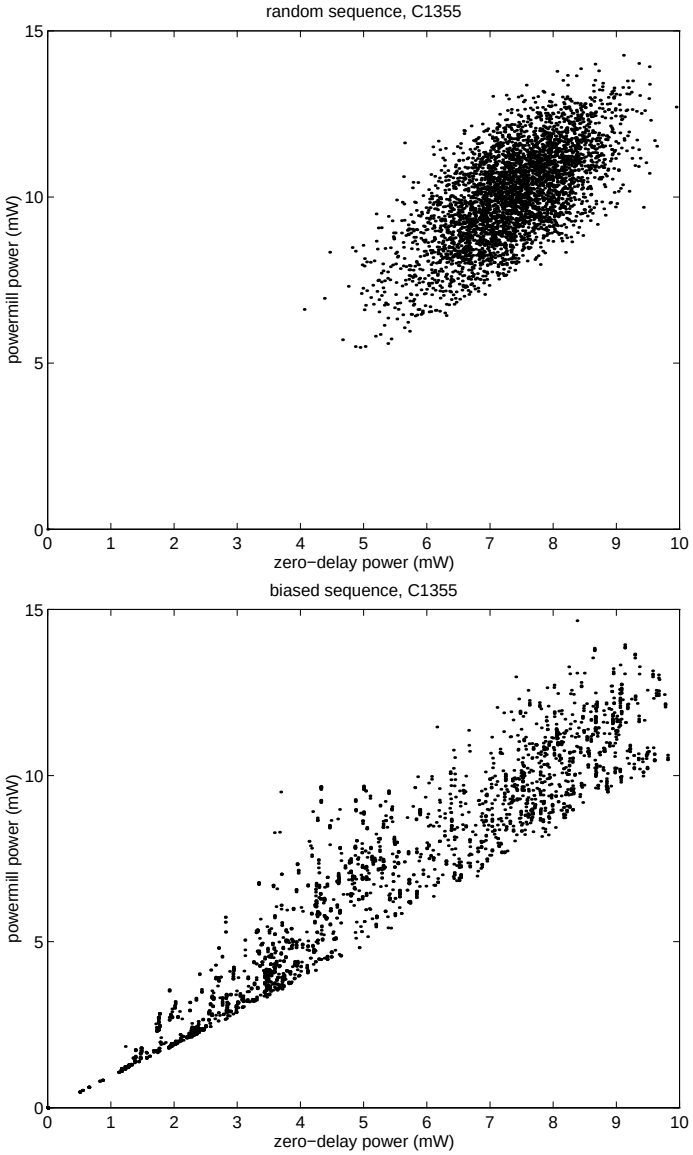


Figure 2: Examples of Power-Mill power values vs. zero-delay power estimates for both random and biased input streams.

In the stratified random sampling, the initial population is partitioned into k disjoint subpopulations, called *strata*. The main objective of stratification is to give a better cross-section of the population so as to gain a higher degree of relative efficiency. The stratification should be done in such a way that strata are homogeneous within themselves w. r. t. the characteristic under study.

There are a number of ways to construct strata. For power estimation, the zero delay power estimate (cycle-based simulator value) is used to construct the strata. Let the predictor value of unit u_i be x_i. Population U is first sorted according to the x_i value of each unit. Let the new order be $u_1, u_2, \ldots, u_n$. Then $K-1$ separators, $p_1, p_2, \ldots, p_{K-1}$, are selected such that

$$x_1 < p_1 < p_2 < \ldots < p_{K-1} < x_N$$

All the units whose x_i values are between two consecutive separators are put into the same stratum, and the strata disjointly cover the whole population. Let the size of each stratum be N_i, then

$$N_1 + N_2 + \ldots + N_K = N$$

Units in a sample are drawn from each stratum independently so that the sample size within the ith stratum become n_i $(i = 1, 2, \ldots, K)$ and

$$n_1 + n_2 + \ldots + n_K = n$$

If the sample is taken randomly from each stratum, the procedure is known as *stratified random sampling*. One way to tell how well the x_i behaves in relation to y_i is from the y_i vs. x_i plot of the population. In Figure 2, the y_i vs x_i plot is depicted for each consecutive vector pair in a biased sequence (i. e. non-random) and a random sequence for $C1355$ circuits (ISCAS85), where x_i and y_i are the zero delay and the PowerMill estimates of each consecutive vector, respectively. As can be seen, the zero delay power estimation produces poor accuracy on a vector-by-vector basis. However, in the stratified sampling, x_i is only used for forming the strata so that units of close y_i values can be put in the same stratum, the inaccuracy of the zero delay power estimates is acceptable.

Compared to [1], the technique of [4] offers the following advantages: 1) It performs sampling directly on the population and thus the estimation results are unbiased, 2) It is more efficient, 3) The sample distributions

are more likely to be a normal distribution, and 4) Difficult population distributions can be handled more effectively. When the population size is large, one can use a two-stage stratified sampling procedure to reduce the overhead of predictor calculation and stratification. The proposed two-stage stratified sampling technique can be easily extended to multi-stage sampling. Note however that the predictors used at different stages should be as uncorrelated as possible.

The proposed techniques have been implemented in C and tested on ISCAS85 benchmarks. Two experiments have been performed. In the first experiment, the authors compare the efficiency of the stratified and simple random sampling on a random and a biased sequences. In the second experiment, they compare the efficiency of two-stage stratified sampling to the technique proposed in [1] for infinite-size population (i. e. , no initial population was given). The results can be summarized as follows.

For random and biased input sequences, the stratification approach results in an efficiency improvement factor (reduction in simulation time) of 1.34 and 7.1 over the simple random sampling approach (for an error level of 5% and a confidence level of 99%). The efficiency improvement of stratified sampling in the random sequence is much smaller than that in biased sequence which is explained as follows. Since the transition probability of each circuit input was assumed to be 0.5, from the law of large numbers, the distribution of the number of input bit changes has a high peak around a value equal to half of the circuit input count. Therefore the power distribution is already very homogeneous. In addition, for a 100,000 random simulations of the biased sequence, no error values greater than 20% was detected for the stratified random sampling while for about 0.1% of the cases, the simple random sampling resulted in such high errors. This demonstrates that stratified sampling can handle difficult population distributions more effectively. As expected, there is no high error violation observed in the random sequence. Power-Mill was used for actual power measurements.

They also compare the efficiency of the proposed two-stage stratified sampling technique to that proposed in [1]. Since this experiment cannot be performed in a reasonable time on PowerMill, the authors use an in-house real-delay gate-level power estimator instead. 1,000 simulation runs with 0.99 confidence and 5% error levels were performed. There was no high error violation observed in either technique. The proposed two-stage stratified sampling technique is however more than twice as efficient as the

Markov-based technique.

2 Probabilistic Compaction

Another approach for reducing the power simulation time is to compact the given long stream of bit vectors using probabilistic automata [11]. The idea is to build a *stochastic state machine* which captures the relevant statistical properties of a given bit stream and then excite this machine by a small number of random inputs so that the output sequence of the machine is statistically equivalent to the initial (much larger) sequence. The relevant statistical properties can for example denote the signal and transition probabilities, and first-order spatio-temporal correlations among bits and across consecutive time frames. The procedure then consists of decomposing the SSM into a set of deterministic state machines, and realizing it through SSM synthesis with some auxiliary inputs. The compacted sequence is generated by random excitement of the auxiliary inputs. As an example, a long sequence of 1-bit vectors 10101010... is compressed to a much shorter sequence 101 by building a machine with states 0 and 1 and a transition probability of 1 to move out of the present state to the next state. Similarly, a long sequence of 1-bit vectors 110011001100... is compressed to a much shorter sequence 11001 by building a machine with states 0 and 1 and a transition probability of 0.5 to stay in the present state and 0.5 to move out of the present state.

The number of states in the probabilistic automata is proportional to the number of distinct patterns in the initial vector sequence. Since this number may be large (i.e., worst-case exponential in the number of bits in each vector), one has to manage the complexity by either 1) partitioning the n bits into b groups with a maximum size of k bits per group and then building a probabilistic automata for each group of bits independently. 2) partitioning the long vector sequence into consecutive blocks of vectors such that the number of distinct vectors in each block does not exceed some user defined parameter, say K. The shortcoming of the first approach is that one may generate a bit pattern (vector) that was not present in the initial sequence (since correlations across different groups of bits are ignored). This is in turn a problem in certain applications with forbidden input patterns (codes not used, illegal instructions, bad memory addresses, etc.). Thus, the

second approcah is often more desirable.

An improved algorithm for vector compaction is presented in [14]. The foundation of this approach is also probabilistic in nature: it relies on adaptive (dynamic) modeling of binary input streams as first-order Markov sources of information and is applicable to both combinational and sequential circuits. The adaptive modeling technique itself (best known as *Dynamic Markov Chain modeling*) was recently introduced in the literature on data compression [2] as a candidate to solve various data compression problems. This original formulation is extended in [14] to manage not only correlations among adjacent bits that belong to the same input vector, but also correlations between successive input patterns. The model captures thus completely spatial correlations and first-order temporal correlations. Conceptually, it has no inherent limitation to be further extended to capture temporal dependencies of higher orders.

Results of these approaches show 1-4 orders of magnitude compaction (depending on the initial length and charactersistics of the input sequence) with negligible error (i.e. $\leq 5\%$ in most cases) using PowerMill as the simulator (see [11] and [14] for details). As a distinctive feature, note that none of the two approaches need the actual circuit to compact the input sequences.

3 Co-simulation for Power Evaluation

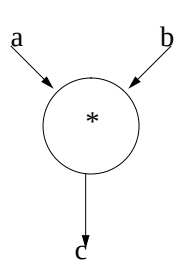
The register-transfer level (RTL) power estimation problem can be stated as follows: "Given an RTL circuit description consisting of m modules, and an input vector sequence of length N, estimate the average power consumption of the circuit over the N cycles". The standard simulation based RT-level power estimation process consists of two steps:

1) Perform behavioral simulation and collect the input statistics for all modules in RTL descriptions.

2) Evaluate the power macro-model equation for each module and sum over the modules.

Busses, clock trees, random logic, etc. are processed separately. There is a wide range of behavioral simulators available today. RTL power evaluation

can be implemented in the form of a *power co-simulator* for standard behavioral/RTL simulators. The co-simulator is responsible for collecting input statistics from the output of behavioral simulator and producing the power value at the end. If the co-simulator is invoked by the behavioral simulator every simulation cycle to collect activity information in the circuit, it is called *census macro-modeling* (cf. Figure 4(a)).




```
c=a*b;
r1 = a^a';
r2 = b^b';
for(i=0;i<16;i++){
    sw_a[i] += r1&1;
    sw_b[i] += r2&1;
    r1 = r1 >> 1;
    r2 = r2 >> 1;
}
a' = a;
b' = b;
```

Figure 3: Overhead of macro-modeling

There are two problems with the census macro-modeling:

- Input data statistics must be collected for every simulation cycle. The statistic gathering overhead is however very large and hence slows down the behavioral simulation. Consider a simple example. For a 16-bit multiplier, the behavioral simulation needs only one instruction, while the statistic gathering requires tens of cycles. (See Figure 3.)[1] Thus the macro-modeling can slow down the simulation significantly. This shows that census macro-modeling is very costly, especially when the vector sequence is very long (tens or hundreds of thousands of vectors).

- Power macro-models are developed by using a training set of input vectors. The training set satisfies certain assumptions such as being pseudo-random data, speech data, etc. Hence the macro-model is biased, meaning that it produces very good results for the class of data

[1] The overhead changes from one macro-model equation to another and from one simulator implementation to another.

which behaves similarly to the training set; otherwise, it produces poor results.

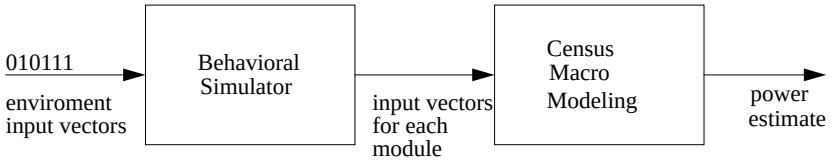
Two macro-modeling schemes are proposed in [6]:

1) *Sampler macro-modeling* collects and analyzes the input vectors for modules only for a relative small number of cycles (cf. Figure 4(b)). In this manner, the overhead of collecting input statistics at every cycle which is required by census macro-modeling is substantially reduced.

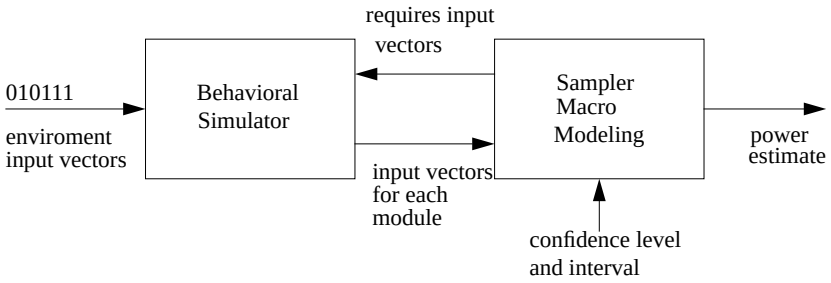
2) *Adaptive macro-modeling* not only interacts with behavioral simulator, but also involves a gate-level simulator on a small number of cycles to improve the estimation accuracy (cf. Figure 4(c)). In this manner, the "bias" of the static macro-models (which is due to the choice of the training set) is reduced or even eliminated.

To solve the efficiency problem of census macro-modeling, the authors of [6] have developed a sampler macro-modeling using statistical random sampling methods. To improve the accuracy or make the power evaluation more sensitive to the population variation, they have developed an adaptive power macro-modeling using a double sampling regression estimator. Both techniques use statistical estimation methods. The sampler macro-modeling uses a simple random sampling technique to reduce the number of cycles during which data statistics is collected without loss of much accuracy while adaptive macro-modeling relies on regression analysis combined with gate-level simulation on a minimum number of cycles to "correct" the static macro-model estimate and hence can be thought of as a self-adjusting macro-model. The designer can select either of these techniques to make accuracy versus speed trade-off.

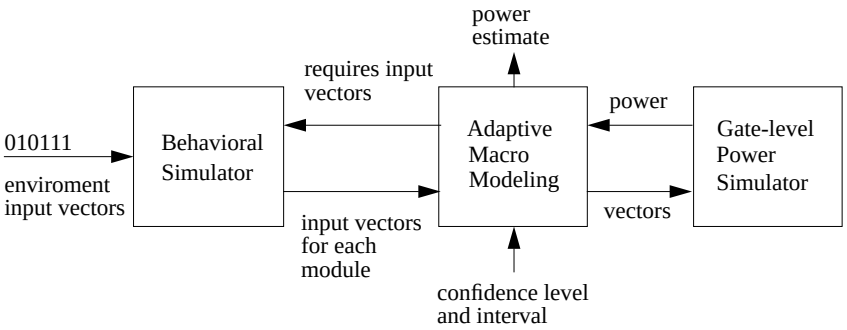
Evaluating the macro-model equation at each cycle during the simulation is actually a census survey. To reduce the run time overhead, one can use simple random sampling to select a sample and only calculate the macro-model equation for the vector pairs in the sample. The sample size is determined before simulation. The sampler macro-modeling randomly selects n cycles and marks those cycles. When the behavioral simulator reaches the marked cycle, the macro-modeling invokes the behavioral simulator for the current input vectors and previous input vectors for each module. The input statistics is only collected in these marked cycles.



(a) Census Macro Modeling



(b) Sampler Macro Modeling



(c) Adaptive Macro Modeling

Figure 4: Power macro-modeling.

The confidence interval of the sample mean can be also derived. However, to calculate the confidence interval, more than one sample is needed, and the sample mean should be close to the normal distribution. From the experiments, the sample mean will approach normal distribution when the sample size is greater than 30. Instead of selecting only one sample of large size, several samples of size 30 (or larger) are selected before the simulation. Then the average value of sample means is the estimate of population mean.

One way to reduce the gap between the power macro-model equation and the gate-level power estimation is to use a regression estimator. Figure 5 shows plots of the the gate level power values (that is, the y values) versus the macro-model equation estimates (that is, the x values) for a 16-bit adder and a 16-bit multiplier (for a macro-model equation with $2 \times n$ parameters where n is the number of inputs to the module). It can be seen that the macro-model equation is a good predictor for gate-level power value (using a linear regression line).

In general, the regression estimator applies to situations where the scatter-plot of y versus x reveals an approximately linear relation of the form:

$$y = \alpha + \beta x.$$

It can be shown that an estimator of the population mean is:

$$\overline{Y}_{lr} = \overline{y} + b(\overline{X} - \overline{x})$$

where $\overline{X}$ is the population mean of characteristics x, $\overline{x}$ and $\overline{y}$ are the subpopulation means for characteristics x and y of the sampled data, respectively while b is obtained by using the method of least-square-error on the sampled data. It is straight-forward to derive the estimator variance and thus the confidence interval and the stopping criterion for sampling. (See [6] for details.)

The regression estimator can achieve very high efficiency if the correlation between the gate-level and the macro-modeling power estimates is high. The lower the correlation between the gate-level and the static macro-model power estimates, the slower the convergence rate of the regression sampler. This also points out that the static macro-model needs to be designed and trained with great caution. In addition, note that a non-linear regression model can provide better results for certain types of circuits (for example, multipliers).

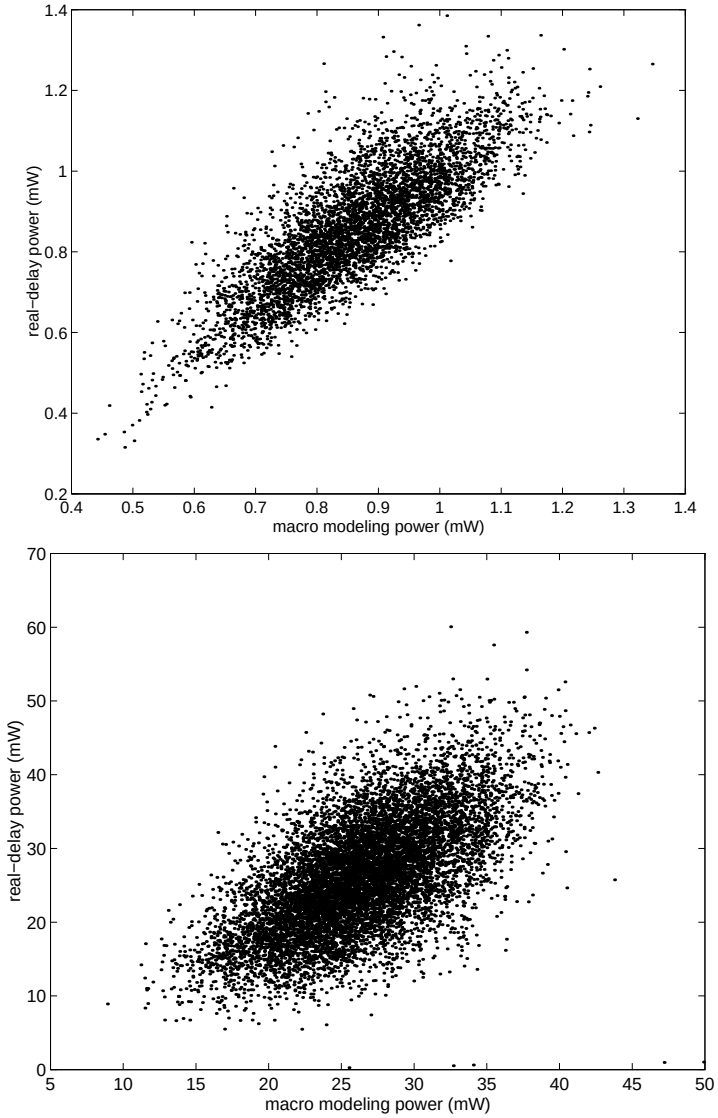


Figure 5: The x-y plots of macro-model equation power vs real delay power of a 16-bit adder (top) and a 16-bit multiplier (bottom) on 10,000 vectors.

In contrast to static macro-models, adaptive macro-models can provide the designers with information (i.e. the confidence level) about the accuracy of predicted value for a given input sequence which is desirable. Adaptive macro-models also guarantee to eliminate the estimator bias given enough gate-level simulations, hence, they tend to be more accurate.

A double-sampling regression estimator can be built on top of sampler macro-modeling (see [6] for details). In the double-sampling regression estimation scheme, n' cycles are selected and marked as 'x' and n cycles randomly selected from the n' cycles and marked as 'y'. Every time the co-simulator encounters cycles marked 'x', the macro-modeling power is evaluated. On the other hand, every time the co-simulation enters cycles marked 'y' then the vector pairs are recorded for later gate-level power simulation. After the behavioral simulation is completed, the gate-level simulation is performed on the set of recorded vector pairs. Then, the regression estimation is performed. Similar to sampler macro-modeling, the confidence level and interval can be derived. For the single-stage regression estimator, the macro-modeling power estimates of all N cycles need to be evaluated, which is not efficient.

The proposed techniques have been implemented in C and tested on standard high-level benchmarks consisting of both data-path and control intensive circuits (e.g., differential equation solver, DSP filters, discrete cosine transform, robot arm controller). Experimental results show that sampler macro-modeling results in an average efficiency improvement of 50X over the census macro-modeling with and average error of 1%. Furthermore, whereas the census macro-modeling incurs large error (an average of 30% for the benchmark circuits) compared to gate level simulation, the adaptive macro-modeling exhibits an average error of only 4.2%.

4 Power Macro-model Design

Most RT-level power estimation techniques use capacitance models for circuit modules and activity profiles for data or control signals [19, 9, 10]. Such techniques, which are commonly known as *power macro-modeling*, consist of generating circuit capacitance models for some assumed data statistics or properties. The statistics of input data is gathered during behavioral simulation of the circuit. Power macro-modeling problem is thus defined as

follows: "Given an input vector sequence of size N, an RT-level circuit with m modules, and assuming N is large enough to capture the typical operation of the circuit, derive a simple function such that the function value of the N vector inputs is as close as possible to the power consumption of the N-vector sequence".

The simplest form of the macro-model equation is given by:

$$Pwr = 0.5V^2 \ f \ C \ E$$

where C is the effective capacitance, E is the mean of the input switching activity, and f is the clock frequency. The Power Factor Approximation (PFA) technique [19] uses an experimentally determined weighting factor, called the power factor, to model the average power consumed by a given module over a range of designs.

More sophisticated macro-model equations can be used to improve the accuracy. The *dual bit type model*, proposed in [9], exploits the fact that the switching activities of high order bits depend on the temporal correlation of data while lower order bits behave similarly to white noise data in the data path or memory modules. Thus a module is completely characterized by its capacitance models in the MSB and LSB regions. The break-point between the regions is determined based on the applied signal statistics collected from simulation runs. The macro-model equation form is:

$$Pwr = 0.5V^2 \ f \ (C_{LSB} \ E_{LSB} + C_{MSB} \ E_{MSB})$$

where C_{LSB} and C_{MSB} represent the capacitance coefficients for the mean activities of the LSB and MSB regions of the input sequence, respectively. Note that E_{LSB} and E_{MSB} are calculated by behavioral/RTL simulation of the system (environment) in which the module is placed. Going further in this direction, one can use a *bitwise data model* as follows:

$$Pwr = 0.5V^2 f \sum_{i}^{n} C_i E_i \tag{1}$$

where n is the number of inputs for the module in question, C_i is the effective capacitance for input pin i, and E_i is the switching activity for the ith pin of the module. The above equation can be made more accurate by including, for example, the spatio-temporal correlation coefficients among circuit inputs (this will however significantly increase the number of variables in the macro-model equation and thus the evaluation overhead).

One can also improve the accuracy by power macro-modeling with respect to both the average input and output activities (the *input-output data model*), that is,

$$Pwr = 0.5V^2 \ f \ (C_{in} \ E_{in} + C_{out} \ E_{out})$$

where C_{in} and C_{out} represent the capacitance coefficients for the mean activities of the input and output bits, respectively. Again, E_{in} and E_{out} are calculated by behavioral/RTL simulation of the system (environment). The dual bit type model or the bitwise data model may be combined with the input-output data model to create a more accurate, yet more costly, power macro-model form.

The Activity-Based Control (ABC) model [20] is proposed to estimate the power consumption of random-logic controllers. All of the above macro-models assume some statistics or properties about the input sequence.

A systematic method for generating power macro-models based on input activity is described in [23]. The following is a summary of that paper.

Without loss of generality, consider the power macro-modeling eqn. 1. Let Pwr_k denote the power consumption of the module at cycle k. One can also write the macro-model equation in a cycle-by-cycle form as follows:

$$Pwr_k = 0.5V^2 f \sum_{i}^{n} C_i E_{i,k} \tag{2}$$

where $E_{i,k}$ is the switching activity (assumes a value of either 0 or 1) for the ith input of the module at cycle k and is obtained from functional simulation of the system in which the module is placed.

The above equation illustrates that macro-modeling can be used to estimate the power consumption at each cycle; this ability is critical to the statistical approach described in [6]. The authors thus distinguish between **(multi-cycle) cumulative** macro-models (such as eqn. (1)) and **cycle-based** macro-models (such as eqn. (2)).

In general, the three basic criteria for effective macro-model design are:

1. The space and time complexity for collection of parameter values for $\mathcal{F}$ and for each evaluation of this function (should be as small as possible).

2. The accuracy of the macro-model (should be as high as possible).

3. The error sensitivity of the macro-model to variations in population behavior (should not be too sensitive).

During the macro-model design phase, some typical input vector sequence, called the *training set*, is used. Next, circuit level power simulation of the typical input vector sequence is performed and curve fitting is applied to derive the coefficients of the macro-model. The macro-model can be very accurate on the original training set. However, the error may become very large when the input sequence statistically deviates from the original training set. For example, a macro-model trained by a pseudo-random input sequence will not produce accurate results for a real application data that exhibits high correlations on the higher order bits. One can imagine alleviating this problem by creating application specific (data-specific) macro-models for each module in the RTL library. This leads to a "proliferation" of macro-models associated with a module and the need to automatically determine what macro-model to use.

The authors of [23] present two approaches for macro-model generation. In one approach, detailed information about module (core) structure and functionality is used to derive a **specialized** closed form capacitance equation with a relatively small number of variables. This approach leads to macro-model equations with high accuracy and low evaluation cost. However, it requires detailed knowledge of the module structure and functionality and cannot be fully automated. In the second approach, one starts with a **general-purpose** macro-model equation with a large number of variables (for example, all pairwise spatio-temporal correlation coefficients among the module inputs) and then use a carefully selected training sequence and a statistical test to eliminate as many variables as possible without incurring large errors. This technique leads to less accurate macro-models with higher evaluation costs, but the advantage is that it can be fully automated. The authors of [23] describe an automatic procedure for macro-model generation based on statistical sampling for the training set design and regression analysis combined with appropriate statistical tests for macro-model variable selection and coefficient calculation. The statistical framework enables them to predict both the power value and the confidence level for the predicted power value.

The authors of [22] report a specialized macro-model for various archi-

tectures (parallel, Wallace tree, etc.) of a 16-bit multiplier which uses 7 variables. The estimation error of this macro-model equation (compared to circuit level simulation result) is 11% on a cycle-by-cycle basis and 4% on an average power basis over a wide range of input streams (not restricted to the training set itself). This result should be compared to those obtained for three automatically generated macro-models of the same multipliers which use 3, 14 and 20 parameters, respectively. The simplest model results in 50-100% error on a cycle-by-cycle basis and 15-20% error on an average power basis. The second model results in 10-20% error on a cycle-by-cycle basis and 2-5% error on an average power basis. The last macro-model results in the same error values as the specialized macro-model. Obviously, complete functional, structural, or layout information about the (IP) cores may not be available (and even when they are known, one may not be able to automatically come up with a concise and accurate macro-model similar to the one generated for the multiplier), and therefore, one must use partitioning of the core (using behavioral hints, or algorithmic partitioning techniques that identify the combinational kernel from the other parts of the design) followed by invocation of specialized or general-purpose macro-model equations on each part.

5 High Level Power Estimation

Most of the high level power prediction tools use profiling and simulation to capture data activity in the circuit. Important statistics include number of operations of a given type, number of bus, register and memory accesses, and number of I/O operations executed within a given period. Instruction level simulation or behavioral simulators are easily adapted to produce this information. Statistical techniques described previously are used to improve the efficiency of these simulators.

In addition to data activity, power dissipation is dependent on the physical capacitances that are being switched from one voltage level to another. Estimating this capacitance at high level of abstraction is difficult and imprecise because it requires estimation of load values in circuits whose RTL structure are not determined yet.

Two approaches are possible at this high level of design abstraction:

1) Develop analytic models for estimating the switched capacitance as a function of the circuit complexity and technology/library parameters.

2) Synthesize the circuit and then estimate the power dissipation of the circuit using either RTL or gate-level estimation techniques described previously.

In the first approach, the key issue is estimation of the circuit complexity. Some circuit parameters that relate to this complexity include, but are not limited to, number and type of arithmetic and/or Boolean operations in the behavioral description, number of states and/or state transitions in a controller description, number of cubes (literals) in a minimum sum-of-products (factored-form) expression of a Boolean function, and amount of decrease in entropy from circuit inputs to circuit outputs. Works reported in [20] and [12] are steps in this direction. More accurate estimates based on probabilistic and/or information theoretic models are possible. One can subsequently use regression analysis and/or regression sampling techniques to correct the analytical model estimates.

The second approach tends to be more accurate. However, it requires the development of a **quick synthesis** capability that mimics a full synthesis program, yet it is much more efficient in terms of its runtime. For a behaviorally described circuit, say in Verilog, the "basic" synthesis tasks include generation of a control and data flow graph (CDFG), operation scheduling, module and register type selection, allocation and binding, and interconnect synthesis. Quick versions of the four basic optimization steps can be developed in a straight-forward manner. For example, one can use HDL parsing and compiler-type code optimization to generate the CDFG, a simple list scheduling to create the operation schedule, a greedy clique covering algorithm to solve the module/register type selection, allocation and binding problems, and a simple multiplexor or bus-based interconnect scheme to connect the RTL components. Chances are good that the quick synthesis option will produces results that are reasonably close to the full synthesis program since - in the absence of tight resource, timing, or power constraints - most of the heuristic techniques used during high level synthesis produce similar results. Unfortunately, this observation does not hold when the user-specified constraints are tight, that is, a fast synthesis option in this case may produce results that are substantially different from those generated by the full synthesis program. Thus, the quality of estimates generated based on the quick synthesis result may become poor. Detailed consideration of

these constraints however, can considerably reduce the supposed efficiency of the quick synthesis option. This is a difficult situation that requires further study and experimentation.

Other important decisions at the behavioral level are choice of hardware partitioning, type of I/O (DMA, port-mapped), and pipeline design (none, functional, structural), synchronization scheme (global clock, local handshaking), and possible use of gated-clocks and multiple-supply voltage levels for low power designs. These decisions tend to have a very large impact on the actual power estimates and must therefore be accurately captured by the quick synthesis option. Fortunately, user (or environment) often provides hints as to which of these alternatives are selected.

Now assume that the basic architecture of the design is determined and the mapping to RTL description has been completed. At this time, the following classes of components are used: memory elements, data path modules, controllers, and (random or glue) logic blocks.

Pre-characterizing (i.e., macro-modeling) the operational modules (such as adders, multipliers, and IP cores) and memory elements (memory core and register file) is possible. Techniques described previously can be used here to model the power dissipation (that is, switched capacitance) of these components.

Obviously, one cannot use macro-modeling for controllers or general logic (such as random logic, glue logic, address or data encoder/decoder) circuits. It is however not too difficult to develop a quick logic synthesis option since the overall circuit architecture, clock frequency, supply voltage level, combinational logic boundaries, etc. are fixed at the logic level. One such approach will start with a general description of a circuit (Boolean network for combinational logic, State Transition Graph for finite state machines, and a combination of combinational and FSM kernels in the form of interacting FSMs for more general circuits) and perform a fast minimal-length encoding followed by fast algebraic extraction and mapping to a gate library. In particular, developing a quick synthesis option for structured design styles appears to be easy and is expected to show higher correlation with the full synthesis program.

Interconnect however complicates the problem as it plays an increasingly important role in determining the capacitive loading of gates while its estimation is a very difficult task even after technology mapping due to lack

of detailed place and route information. Approximate estimates can be obtained by using information derived from a companion placement solution [17] or by using stochastic/procedural interconnect models [18].

6 Acknowledgment

This article could have not been written without the help of my students at USC (in particular, C-S. Ding, C-T. Hsieh, D. Marculescu, R. Marculescu, and Q. Wu) whose research works are reflected here. The research was supported in part by DARPA under contract no. F33615-95-C1627 and an NSF NYI award.

References

[1] R. Burch, F. N. Najm, P. Yang, and T. Trick. "A Monte Carlo approach for power estimation," IEEE *Transactions on VLSI Systems*, 1(1):63–71, March 1993.

[2] G. V. Cormack and R. N. Horspool. "Data compression using dynamic Markov modeling," *Computer Journal*, 30(6):541–550, June 1987.

[3] C. Deng. "Power analysis for CMOS/BiCMOS circuits," *Proceedings of 1994 International Workshop on Low Power Design*, April 1994, pages 3–8.

[4] C-S. Ding, C-T. Hsieh, Q. Wu and M. Pedram. "Stratified random sampling for power estimation," *Proc. Int'l Conf. on Computer Aided Design,* November 1996, pages 577–582.

[5] A. A. Ghosh, S. Devadas, K. Keutzer, and J. White. "Estimation of average switching activity in combinational and sequential circuits," *Proceedings of the 29th Design Automation Conference*, June 1992, pages 253–259.

[6] C-T. Hsieh, C-S. Ding, Q. Wu and M. Pedram. "Statistical sampling and regression estimation in power macro-modeling," *Proc. Int'l Conf. on Computer Aided Design,* November 1996, pages 583–588.

[7] C. M. Huizer. "Power dissipation analysis of CMOS VLSI circuits by means of switch-level simulation," *IEEE European Solid State Circuits Conf.*, 1990, pages 61–64.

[8] S. Iman and M. Pedram. "POSE: Power optimization and synthesis environment," *Proc. 33rd Design Automation Conf.,* June 1996, pages 21–26.

[9] P. Landman and J. Rabaey. "Power estimation for high-level synthesis," *Proceedings of IEEE European Design Automation Conference*, February 1993, pages 361–366.

[10] D. Liu and C. Svensoon. "Power consumption estimation in CMOS VLSI chips," *IEEE Journal of Solid State Circuits*, 29(6):663–670, June 1994.

[11] D. Marculescu, R. Marculescu and M. Pedram, "Stochastic sequential machine synthesis targeting constrained sequence generation," *Proc. 33rd Design Automation Conf.,* June 1996, pages 696–701.

[12] D. Marculescu, R. Marculescu and M. Pedram. "Information theoretic measures for power analysis," *IEEE Trans. on Computer-Aided Design,* 15(6):599–610, June 1996.

[13] R. Marculescu, D. Marculescu and M. Pedram. "Efficient power estimation for highly correlated input streams," *Proc. 32nd Design Automation Conf.,* June 1995, pages 628–634.

[14] R. Marculescu, D. Marculescu and M. Pedram. "Adaptive models for input data compaction for power simulators," To appear in *Proceedings of the 2nd Asia-Pacific Design Automation Conference*, January 1997.

[15] F. N. Najm. "Transition density: A new measure of activity in digital circuits," IEEE *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 12(2):310–323, February 1993.

[16] M. Pedram. "Power minimization in IC design: principles and applications," *ACM Trans. on Design Automation of Electronic Systems*, 1(1): 3-56, January 1996.

[17] M. Pedram and N. Bhat. "Layout driven technology mapping," *Proc. 28th Design Automation Conf.,* June 1991, pages 99–105.

[18] M. Pedram, B. T. Preas. "Accurate prediction of physical design characteristics of random logic," *Proc. Int'l Conf. Computer Design: VLSI in Computers and Processors,* October 1989, pages 100–108.

[19] S. Powell and P. Chau. "Estimating power dissipation of VLSI signal processing chips: The PFA techniques," *Proceedings of IEEE Workshop on VLSI Signal Processing IV*, Vol. IV, 1990, pages 250–259.

[20] J. Rabaey P. Landman. "Activity-sensitive architectural power analysis for the control path," *Proceedings of International Symposium on Low Power Desing*, April 1995, pages 93–98.

[21] C-Y. Tsui, M. Pedram, and A. M. Despain. "Efficient estimation of dynamic power dissipation under a real delay model," *Proceedings of the* IEEE *International Conference on Computer Aided Design*, November 1993, pages 224–228.

[22] Q. Wu and M. Pedram. "Statistical design of macro-models for RT-level power evaluation," Technical Report CENG 96-24, University of Southern California, October 1996.

[23] Q. Wu, C-S. Ding, C-T. Hsieh, and M. Pedram. "Statistical design of macro-models for RT-level power evaluation," To appear in *Proceedings of the 2nd Asia-Pacific Design Automation Conference*, January 1997.