Automatic Characterization and Modeling of Power Consumption in Static RAMs

Mauro Chinosi
STMicroelectronics CR&D
I-20041, Agrate B. (MI), Italy
Tel: +39 39 6036374
mauro.chinosi@st.com

Roberto Zafalon
STMicroelectronics CR&D
I-20041, Agrate B. (MI), Italy
Tel: +39 39 6035698
roberto.zafalon@st.com

Carlo Guardiani
STMicroelectronics CR&D
I-20041, Agrate B. (MI), Italy
Tel: +39 39 6035688
carlo.guardiani@st.com

1. ABSTRACT

An automatic modeling technique is presented in this paper that allows to build an accurate model of power consumption in embedded memory blocks. A software neural-network is used to create a regression tree by automatically splitting those variables that have a discontinuous effect on the power consumption. An application of the methodology to the modeling of a 0.35μm CMOS embedded SRAM is presented.

1.1 Keywords

Power estimation, Memory modeling, Static RAMs

2. INTRODUCTION

The importance of static and dynamic simulation of power consumption in VLSI circuits is continuously increasing. A number of tools at different levels of abstraction have been introduced [1], to facilitate power analysis throughout the design flow. An important class of VLSI circuits, like DSP cores, embedded processors, etc., make an extensive use of data-storage units like cache, frame buffers, register files etc. Therefore the power consumed by the memory units is a large fraction of the total power consumption for such circuits. Nevertheless, not quite the same degree of attention has been paid to the accurate modeling of power in memories. The models described in [2], [3] and [4] achieve a relatively good accuracy in modeling switching activity dependent power but are limited to relatively small size memories or to specific architectures. The problem of deriving a general model of the power consumption in memories is far from being trivial because, depending on the state of the control ports, it is possible to observe orders of magnitude differences in the power dissipation with similar I/O switching activity pattern. This is due to the wide range of unique operations affecting power consumption that different memories may perform. For example, direct R/W vs. page addressing, etc. Other control inputs or special architectures (e.g. self timing) may apply to specific types of memories only (e.g. DRAMs, multiple port RAMs). Furthermore, depending on the complexity, the internal architecture can be completely different. In order to derive a general model, the I/O behavior shall be considered only, thus ignoring the internal structure and building blocks of the memory. These constraints are even more demanding if the automatic characterization of the model is considered. In this paper we propose a black box approach for modeling of power consumption in embedded memories which is based only on I/O functional and timing information. We will demonstrate that the algorithm is capable of automatically detecting the particular power patterns associated with a given functionality without any knowledge of the hardware implementation. The organization of the paper is the following: the automatic modeling algorithm is described in Section 3. An application example is shown in Section 4. Section 5 will present conclusions and directions for future work.

ISLPED98, Monterey, CA, USA

3. AUTOMATIC MODELING ALGORITHM

In this section the flow for the automatic modeling of memory's power consumption is presented. The starting point is a spice netlist of the memory, including parasitics, and an appropriate set of stimuli. No assumptions will be made neither on the architecture nor on the operations of the memory. All the needed information shall be inferred from the simulated response of the memory to the applied stimuli. In general the model variables will be both the states and the transitions at the I/O ports. The target model is a linear regression tree as that described in [5]. However an adaptive power-driven algorithm based on a software neural network (SNN) is used in order to build the tree instead of using a statistical approach. The memory is considered as a state machine with number of states exponential to the number of primary inputs. Let's assume that it is characterized by u control inputs $(c_1, \ldots, c_u)$ and v non-control inputs $(d_1, \ldots, d_v)$. In general the power consumed by the memory will be a non linear and, likely, discontinuous function of the value of both control and non-control inputs, i.e.:

$$(1) \qquad PWR = P_{\text{n-lin}}(c_1, \ldots, c_u, d_1, \ldots, d_v)$$

where the name of an input signal in Eq. (1) is used to represent either the binary state or a transition event on the corresponding input. More often the power consumption will

be a strongly non-linear function of the state for some of the control inputs and a reasonably linear function of the transition events at every non-control input. Let assume that the unknown function $P_{\text{n-lin}}$ is sampled M times, represented by the couple $(\boldsymbol{X}, P)$ where $\boldsymbol{X} = (x_1, \ldots, x_{u+v})$ is a $(u+v \times 1)$ vector with $M \gg u+v$. The range of variation of P can be arbitrarily quantized in k intervals $P_i, i = 1, \ldots, k$. Initially a number of 'codebook vectors' m_i are selected to identify the different domains of the input vector X associated with the quantized values of P. Usually different codebook vectors are associated to each class of P values, and $(\boldsymbol{X}, P)$ is then assigned to the same class to which the nearest m_i belongs. Let m_c define the nearest m_i to $\boldsymbol{X}$. The value of m_i that minimize the misclassification errors in the above nearest-neighbor classification can be found as asymptotic values in the basic LVQ learning process [6]. The recognition accuracy relative to the codebook can be tested and, if a reasonable value is reached, a partitioning of the M couples $(\boldsymbol{X}, P)$ in k clusters is performed. The graph in figure 1 shows the power clustering for a 2 variable case.

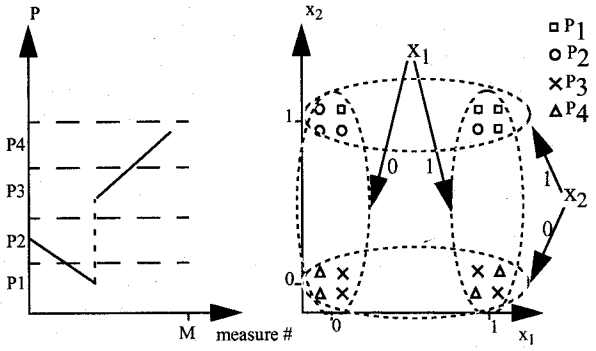

Fig. 1:Power to cluster mapping.

Given k clusters $C_1, \ldots, C_k$, it is possible to define a corresponding vector of *centers of gravity*: $G = (g_{j,1}, \ldots, g_{j,u+v})$ where each component $g_{j,i}$ is defined as:

$$g_{ji} = \sum_{l=1}^{Q_j} \frac{b_{li}}{Q_j} \qquad i = 1, \ldots, u+v \qquad j = 1, ..$$

Q_j is the number of codebook vectors b_l belonging to the j-th cluster ($\sum Q_j = \sum Codebooks$). Each g_{ji} represents the average value of input x_i in cluster j, that is when the power consumption is close to P_j. Let sort the k classes by increasing values of P and observe the behavior of the piece-wise function $g_i(P)$ obtained by joining all the $g_{j,i}$ corresponding to input $i-th$ as shown in figure 2. If, for some input i, $x_i = 0(1)$ corresponds to "low" P values and, at the same time $x_i = 1(0)$ corresponds to "high" P values, the curve $g_i(P)$ should show a sudden step. Otherwise, a smooth or oscillating behavior should be observed. For example, the curves $g_1(P)$ and $g_2(P)$ plotted in figure 2 are obtained by considering the data from the example in figure 1. Notice that $g_2(P) = 1$ for $P < P_2$ and 0 for $P > P_2$.

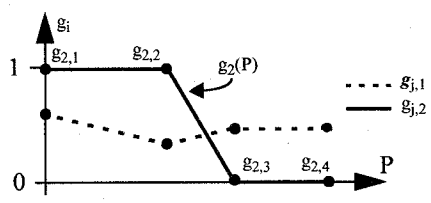

Fig. 2:Creation of $g_i(P)$.

This indicates that the value of input x_2 discriminates among different power regions. The function $g_1(P)$ instead is smooth and always close to 0.5. Therefore the input x_1 can't be associated with different power consumption patterns. This is also evident from the $x_1 \leftrightarrow x_2$ graph in figure 1, as x_1 assumes indifferently value 1 and 0 in all the clusters whereas x_2 is 1 in P_1, P_2 and 0 in P_3, P_4. It is thus evident that x_2 is a natural candidate for splitting. In order to extract the splitting variables automatically from the analysis of the functions $g_i(P), i = 1, \ldots, u+v$ it is necessary to define some selection criteria. First, monotonicity is required; then a fitness function can be defined by taking into account the stationary points, the steepness and position of steps, etc., For example, the simple variance $s(g_i) = \sum_j (g_{ji} - \langle g_i \rangle)^2$ can be used as the fitness function. The selection of variables for splitting will be based upon the value of the fitness function, i.e., variables associated with largest fitness values will be selected first. The use of a generic fitness function allows for a more flexible and general implementation of the algorithm. It might happen that no variables are available for splitting. In this case the proposed algorithm fails, and a statistical technique can be applied, or another splitting variable may be selected. The above described modeling algorithm is iterative, as it is re-applied successively to the tree obtained after each branching, until certain stopping criteria are met. Since after each iteration the number of sub-models is increased by one, a stopping criteria should be defined in order to control the trade-off between accuracy and complexity. Three different criteria have been defined: *i)* a predefined accuracy goal is met, *ii)* the maximum number of branches has been reached, *iii)* the accuracy improvement is less than a predefined tolerance. If *i)* or *ii)* are met then the algorithm is stopped. If *iii)* is met then the current branch is terminated, but other branches could be further expanded until *i)* or *ii)* are met. If the accuracy is still not satisfactory it is possible to start a new run by either

changing the training set, or by redefining the power regioning, the stopping criteria or both.

4. APPLICATION EXAMPLE

The proposed methodology was tested on a static, synchronous RAM with 32 lines, 16 bit words. This memory supports several functions: *i)* read; *ii)* write; *iii)* power off; *iv)* output three-state; *v)* read & write. The required accuracy threshold was set at ±20% max error with respect to PowerMill. In figure 3 the obtained regression tree is shown.

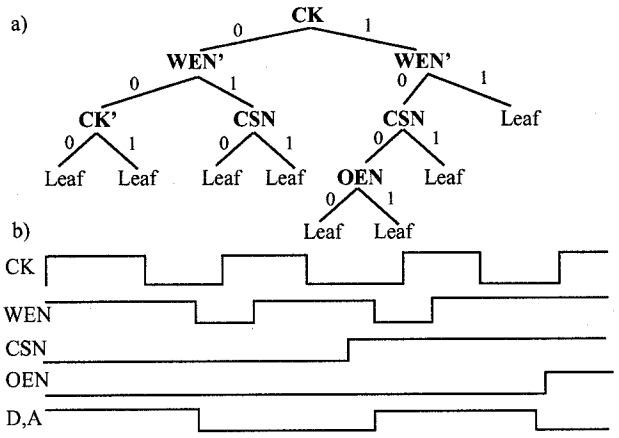

Fig. 3:a) Regression tree for a 32x16 static RAM. b) Timing

The algorithm first selected the clock status CK as the root of the tree, thus identifying a synchronous circuit. The next variable selected was a transition on write enable (WEN') thus discriminating between write and read mode (WEN'= 0 -> read). The other branches were generated similarly. Only control variables have been selected for splitting. A shmoo plot showing model prediction vs. simulation is presented in figure 4. The target accuracy with respect to PowerMill was achieved by using a linear model for the leafs. Better accuracy can be obtained by tightening the precision constraint (thus growing the tree). However, by considering that a 20% error magnitude at the level of each single transition, corresponds to a much lower RMS error, the achieved trade-off between model simplicity and accuracy is indeed extremely good.

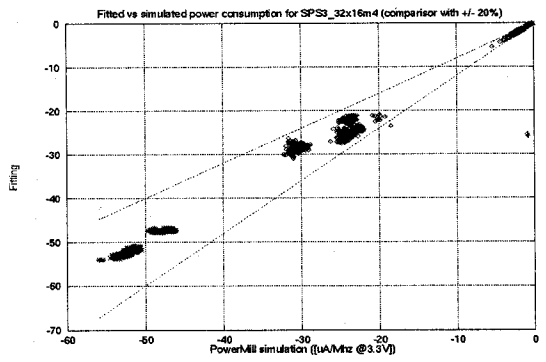

Fig. 4:Model vs. simulation. Comparison with 20% error.

The benchmarks results are summarized in Table 1, showing that a 100X computational time speedup is achieved with a limited loss of accuracy.

Benchmark	Powermill		Design Power		Error
# of vectors	CPU	P [mW]	CPU	P [mW]	%
1436	12h 45min	1.128	1min	1.214	+7.6
700	20h	2.202	1min	2.356	+7
700	2h	0.0345	1min	0.0371	+7.4
256	1h 50min	0.910	1min	0.965	+6%

TABLE 1: TOTAL AVERAGE POWER RESULTS

5. CONCLUSION

An automatic approach for modeling and characterizing the power consumption in semiconductor memories have been proposed. The methodology does not require any knowledge of the memory architecture or operations. Because of this, the algorithm is suitable for integration in an automatic design flow. The capability of automatically detecting the control signals and the modes of operations of the macroblock is general and can be applied to other kinds of memories. The automatic modeling capabilities have been demonstrated on an application example showing that a user controllable level of accuracy can be obtained while keeping the model simple. As future work we plan to improve the accuracy of the model by using non-linear leaf models and to include other parameters, like cut-size, etc., in order to automatically produce power models for module generators.

6. REFERENCES

[1] Design Power. "Synopsys Power Product Reference v1997.01", Synopsys Inc., Mountain View, CA.

[2] D.Liu and C. Svensson, "Power Consumption Estimation in CMOS VLSI Chips", IEEE Journal of Solid State Circuits, Vol. 29, No. 6, June 1994

[3]R. J. Evans, P. D. Franzon, "Energy Consumption Modeling and Optimization for SRAM's", IEEE Journal of Solid State Circuits, Vol. 30, No. 5, May 1995

[4]K. Ogawa, "PASTEL: A Parametrized Memory Characterization System", Proceedings of DATE98, March 1998

[5]L. Benini, A. Bogliolo, M. Favalli and G. De Micheli, "Regression models for behavioral power estimation", Proceedings of PATMOS 96, pp. 179-187, September 1996

[6]T. Kohonen, "Improved versions of learning vector quantization", Proc. of the Intern. Joint Conference on Neural Networks, pp. 545-550, June 1990