[17] A. Sinha, S. K. Gupta, and M. A. Breuer, "Validation and test generation for oscillatory noise in VLSI interconnects," in *Proc. Int. Conf. Computer-Aided Design*, 1999, pp. 289–296.
[18] A. B. Kahng and S. Muddu, "An analytical delay model for RLC interconnects," *IEEE Trans. Computer-Aided Design*, vol. 16, pp. 1507–1513, 1997.
[19] Y. I. Ismail, E. G. Friedman, and J. L. Neves, "Fig.s of merit to characterize the importance of on-chip inductance," *IEEE Trans. VLSI Syst.*, vol. 7, pp. 442–449, Dec. 1999.
[20] A. Deutsch *et al.*, "When are transmission-line effects important for on-chip interconnections?," *IEEE Trans. Microwave Theory Tech.*, vol. 45, pp. 1836–1846, Oct. 1997.
[21] *IC Station User's Manual, Version v8.7_3, Mentor Graphics*, 1991–1999.
[22] B. Choi and D. M. H. Walker, "Timing analysis of combinational circuit including capacitive coupling and statistical process variation," in *Proc. 18th IEEE VLSI Test Symp.*, 2000, pp. 49–54.
[23] L. Chen, X. Bai, and S. Dey, "Testing for interconnect crosstalk defects using on-chip embedded processor cores," in *Proc. Design Automation Conf.*, 2001, pp. 317–320.
[24] K. Sekar and S. Dey, "LI-BIST: A low-cost self-test scheme for SoC logic cores and interconnects," *J. Electron. Testing: Theory Applicat.*, vol. 19, no. 2, pp. 113–123, 2003.
[25] *IC Station User's Manual*, 1991–1999.

IDAP: A Tool for High-Level Power Estimation of Custom Array Structures

Mahesh Mamidipaka, Kamal Khouri, Nikil Dutt, and Magdy Abadir

***Abstract*—While array structures are a significant source of power dissipation, there is a lack of accurate high-level power estimators that account for varying array circuit implementation styles. We present a methodology and a tool, the implementation-dependent array power (IDAP) estimator, that model power dissipation in SRAM-based arrays accurately based on a high-level description of the array. The models are parameterized by the array operations and various technology dependent parameters. The methodology is generic and the IDAP tool has been validated on industrial designs across a wide variety of array implementations in the e500[1] processor core. For these industrial designs, IDAP generates high-level estimates for dynamic power dissipation that are accurate with an error margin of less than 22.2% of detailed (layout extracted) SPICE simulations. We apply the tool in three different scenarios: 1) identifying the subblocks that contribute to power significantly; 2) evaluating the effect of bitline-voltage swing on array power; and 3) evaluating the effect of memory bit-cell dimensions on array power.**

***Index Terms*—Estimation, high-level power estimation, implementation-dependent array power (IDAP), implementation styles.**

Manuscript received July 24, 2003; revised December 15, 2003. This work was done in collaboration with Motorola Inc. and was supported in part by the National Science Foundation under Grant CCR 0203813. This paper was recommended by Associate Editor M. Pedram.

M. Mamidipaka and N. Dutt are with the Center for Embedded Computer Systems, University of California, Irvine, CA 92697 USA (e-mail: maheshmn@cecs.uci.edu; dutt@cecs.uci.edu).

K. Khouri and M. Abadir are with the High-Performance PowerPC Platforms Group, Motorola Inc., Austin, TX 78729 USA (e-mail: kamal.khouri@motorola.com; m.abadir@motorola.com).

Digital Object Identifier 10.1109/TCAD.2004.833609

[1]e500 is the Motorola processor core that is compliant with the PowerPC Book E architecture.

I. Introduction

Many factors have contributed to the increased demand for lowering power consumption in today's semiconductor designs. Market demand for portable electronics has driven the need for low-power devices, which rely on a battery for operation and, hence, the aim is to increase the lifetime of the battery between recharges. While performance has traditionally been the main driver for high-end desktop and network processors, the need for reducing power consumption has become a serious issue as these devices operate at maximum tolerance levels. For desktop computing, lower yields and higher cooling-system costs increase the overall cost of the system. Similarly, in the network-processor domain, heat-removal systems in a switch farm have a fixed capacity and, hence, a limit is imposed on the number of processors that can be placed on a single board.

Array structures, such as register files, branch-target buffers, tag arrays, and caches consume up to 70% of the overall power in a SoC [4]. It has also been shown that caches alone consume up to 40% of total power [9]. In this paper, we focus on the dynamic power estimation in CMOS-based array structures. Fig. 1 shows the typical design flow for custom memory structures. To meet the stringent power constraints, it is important to obtain power estimates at each level of design hierarchy. As we go down in the design hierarchy, the level of detail in the design increases, leading to more accurate estimates. Although there has been a sizable body of work on power estimation in array structures (Section II summarizes this research), the focus has either been toward modeling at the microarchitectural level or modeling through characterization after the availability of transistor level design. Models at the microarchitectural level lack accuracy because of the nonavailability of design-specific information, such as sense-amplifier type (differential or inverter based), decoder-style type (static CMOS or dynamic CMOS based) etc. On the other hand, models at the transistor level, while accurate, are available only in the latter stages of the design cycle. Hence, there is a need for accurate power models which bridge the gap between the microarchitecture level models and characterization-based models. To the best of our knowledge, this is the first attempt which tries to bridge this gap.

In this paper, we propose a methodology for accurate estimation of power dissipation in CMOS-based arrays using a high-level description of the design, which contains microarchitecture-level parameters and subblock circuit-implementation styles of the array structures. The main contributions of this work are: 1) the ability to represent various organization and implementations of arrays; 2) the ability to abstract parameters which define the power consumption in arrays for a given implementation style; and 3) a methodology to generate accurate power models based on these parameters at a higher level in the design flow. This technology provides designers with the ability to perform a wide variety of tasks, as follows.

- Conduct "what–if" studies on the effects that implementation changes may have on power, without the need to redesign at the transistor-level, and hence, avoid time-consuming SPICE simulations.
- Conduct accurate power-dissipation studies for new process technologies to better understand the impact a new transistor may have on a design.
- Generate highly accurate power models for register transfer level (RTL) design-space exploration.

The remainder of the paper is organized as follows. Section II summarizes the related work in the area of array-power modeling and estimation. Section III provides a brief background in array structures

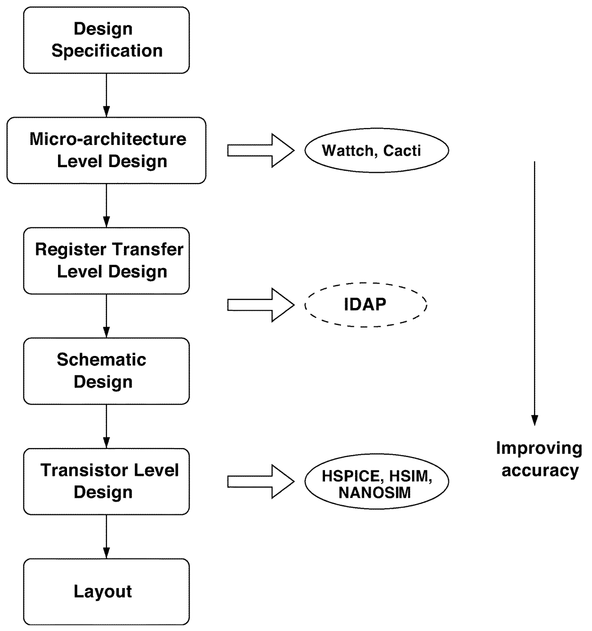

Fig. 1. Typical design flow for custom memory structures.

and design styles. Section IV describes our estimation methodology and presents an illustrative example that demonstrates generation of a power model using a high-level configuration file. Section V presents a validation of our models against transistor-level SPICE to measure the estimation accuracy. Section VI describes some applications of the tool, implementation-dependent array power (IDAP). Section VII contains concluding remarks and directions for future work.

II. Related Work

While there are tools for estimation of RTL-power dissipation [11], [12], they are meant for designs based on power-characterized modules and combinatorial logic and cannot be used for custom SRAM-based array designs. A study of different approaches used for modeling energy dissipation in SRAMs is illustrated by Evans *et al.* in [5]. Related work on power estimation in arrays can be categorized into characterization-based power modeling and microarchitecture-level power modeling. In this section, we will first detail the research work related to characterization-based modeling, followed by research in microarchitecture level power modeling.

Traditionally, power estimation for array structures has been performed at the transistor-level using SPICE. Although highly accurate, these SPICE-based simulations cannot be performed when application-level (simulation vectors in the order of thousands/millions) power estimates are required because of their impractical runtimes. To address this issue, an ASIC on-chip memory-characterization tool (PASTEL) [10] was developed by Ogawa *et al.*. More recently, Mamidipaka *et al.* proposed a more generic methodology to generate analytical models for power dissipation in arrays for wide variety of array-implementation styles through transistor-level simulations [8].

For estimation of power in arrays at higher levels in the design cycle, microarchitecture-level models have been proposed. However, researchers have focussed on modeling specific array-implementation styles. For instance, Zyuban and Kogge [21] propose analytical power-dissipation models for register file implementations. Simplified energy models for caches as a function of hits and misses are proposed by Su and Despain [18] and also used in [15] and [17] with minor enhancements for design-space exploration. Kamble and Ghose [6] proposed analytical models for estimating energy dissipation in conventional caches and low-power caches. In these models the power consumed due to control logic, decode logic, and sense amplifiers is considered negligible. While this assumption may hold for large conventional caches, it does not hold for high-performance custom arrays found in microprocessors. Energy models specific to caches have also been proposed by Li and Henkel [7]. These models are similar to those proposed by Kamble and Ghose except that the energy dissipation due to decoder, output drivers, and memory write is accounted for based on statistical assumptions not described in the paper. The Cacti [19] tool was enhanced for more accurate power estimation, but is applicable only for specific implementations of caches at the microarchitectural level. Brooks *et al.* proposed parameterizable analytical models to estimate power for different sizes of generic array structures [4]. These models, referred to as Wattch models, are based on the capacitance values estimated using the Cacti [19] tool. These microarchitecture-level models usually have limited absolute accuracy and are only used for relative comparisons.

Analysis on some industrial array designs show that the Wattch models can have an error of as much as 94%. This is because in reality, the power dissipation depends greatly on the subblock implementation styles which are not captured in the models. The Wattch models assume typical subblock circuit implementation styles for power estimation because of lack of such information at the microarchitectural level. For example, the Wattch models assume inverter sense-amplifier-based read logic, static CMOS-based decode logic, and precharge-based write logic for all array structures. This affects the accuracy of the models for varied implementation styles of the subblocks. In fact, experiments done on a 64×128 size array using SPICE simulations, show a variation of 29% in power dissipation for just two different implementations of sense-amplifiers in array read logic (differential sense-amplifier and static inverter sense-amplifier), whereas Wattch reports same power for both implementations. Moreover the existing high level models for arrays typically estimate the power based on wordline capacitance and bitline capacitances and do not capture any other nodes which could contribute to power significantly.

As can be noted, prior research focus has either been on power modeling based on simulations at the transistor level or at a level higher in the design hierarchy, where there are limited details about the array design. In practice, there are a variety of implementations for each subblock in an array, and the implementation choices significantly affect the power dissipation. In this paper, we present an accurate power-estimation methodology and an estimation tool (IDAP) applicable to a wide variety of array implementations. We evaluate IDAP by comparing its power estimates with detailed SPICE simulations.

III. Array Structures

Array structures contribute to a significant portion of the total system power dissipation. Caches, tag arrays, register files, branch table predictors, instruction windows, translation lookaside buffers are common examples of array structures in micro-processors. As shown in Fig. 2, array structures are primarily composed of address decoders, wordline drivers, memory core, read column logic, write column logic, read control, and write control logic.

For read and write operations, the row decoder selects the appropriate wordline corresponding to the input address, thereby activating a row in the memory array. For a read operation, the precharged bitlines either retain charge or discharge depending on the data stored in the cells selected by the wordline. The sense-amplifier detects the changes in the voltage on the bitlines and the appropriate data is multiplexed to the data output. For a write operation, the sense-amplifiers are isolated and the write buffers drive the bitlines in accordance to the data being

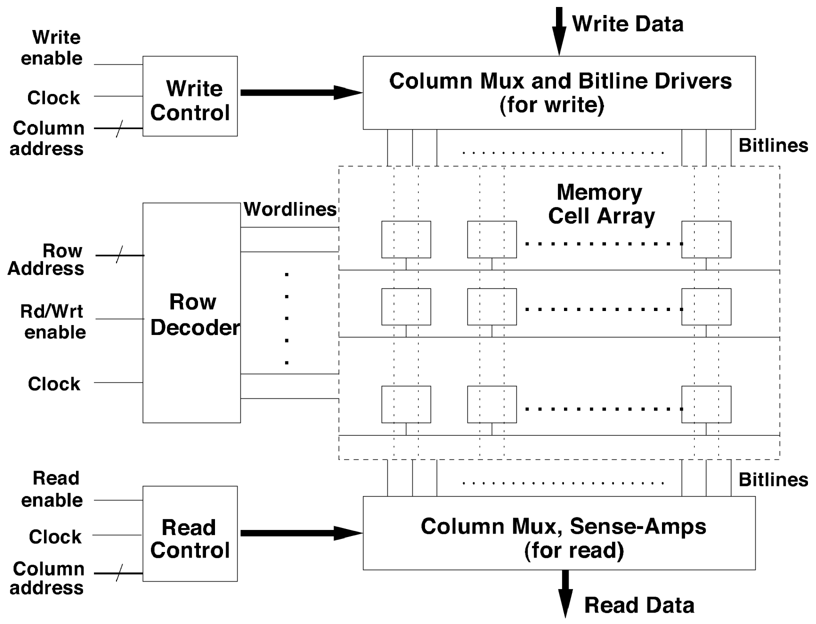

Fig. 2. Typical architecture of array structures.

TABLE I
ARRAY-SUBBLOCK CIRCUIT-IMPLEMENTATION STYLES

Module	Circuit implementation styles
Decoder	static or dynamic CMOS logic based
Bitline drivers	precharge transistor or buffer based
Read sense-amplifier	differential or inverter based
Bitline style	single-ended or double-ended
Memory cell	common or separate read/write ports
Self timed logic	dual clock based or delayed clock based or replica bitline based or none

written into the memory location corresponding to the write address. In the case of CAMs, there is an additional operation, *match*, that compares an input data with the internally stored data. To enable the match operation in CAMs, each CAM memory cell consists of the compare logic in addition to static RAM cell [13].

Arrays typically differ from each other in size, row and column organization, and subblock implementation style at the circuit-level. A specific circuit implementation style is chosen for optimal power or performance. Table I lists some examples of the commonly used subblock implementations. Note that the functionality of the read control and write control logic depends on the implementation of the read and write column logic, respectively. For example, the read control logic employing an inverter-based sense amplifier is significantly different from an implementation using a differential sense amplifier.

IV. ESTIMATION METHODOLOGY

In this section, we begin with an overview of the proposed methodology, followed by more detailed description of various phases of the methodology. Currently, the focus of this tool is on dynamic power dissipation since it is the major contributor to the total power. While the tool currently generates accurate models for capacitive switching power, we assume short-circuit power to be 10% of the switching capacitance power. However, we want to model short-circuit power more accurately in latter versions of the tool.

A. Overview

Fig. 3 is the flow diagram of our methodology. The inputs to the tool are the configuration file, technology file, and some optional parameters are shown in light blue/gray colored boxes. The configuration file is the user-provided information, which constitutes the specification of the array organization (for example, the number of rows and columns) and the circuit implementation style for each array subblock. The technology file for the intended CMOS is another input to the tool through which technology specific parameters such as unit gate capacitance, unit drain capacitance, and metal capacitance, are derived by the tool. Finally, the optional parameters like the input-drive and output-node capacitances help the tool generate more accurate power models. The output of the tool is the power model as a function of array primary inputs and outputs and operation on the array.

Since power dissipation in an array is the additive sum of the power in each subblock, the IDAP tool generates a model for each subblock based on information specified in a user-provided configuration file. The tool starts with analyzing the array configuration file to determine the organization of the array and its subblock-implementation styles. Then, the analyzer together with the knowledge base, determines the essential nodes (nodes within the subblock which contribute to power) and influential nodes (nodes in the subblock which contribute to power in other subblocks) based on parameters specified in the configuration file. More details on the knowledge base, configuration file, influential, and essential nodes are given in the following sections. The next phase of the methodology works on these priority nodes along with the process-technology parameters to estimate the switching capacitances in the subblock for each array operation. The analytical models required for capacitance calculation are obtained from the knowledge base as well. The capacitance estimator also has an optional input file specifying the input drive on the address and data-in bus and capacitive loads on the data-out bus. The estimator uses these parameters for more accurate estimation of internal node capacitances. The power models for the subblocks are then generated as a function of node capacitances and switching activity in the subblock. The switching activity is either dependent on the array inputs or independent. For dependent-switching activity, the model generated is parameterized based on those inputs (as shown by $F(T_i, C_i)$ in Fig. 3. The independent-switching activity is determined by the analyzer. For example, in a double-ended bitline-based read, half the bitlines discharge, regardless of the data. The loop containing the analyzer, capacitance estimator, and subblock-power model generator, as shown in the Fig. 3, is repeated for all the array subblocks. The final phase of the methodology stitches the power models

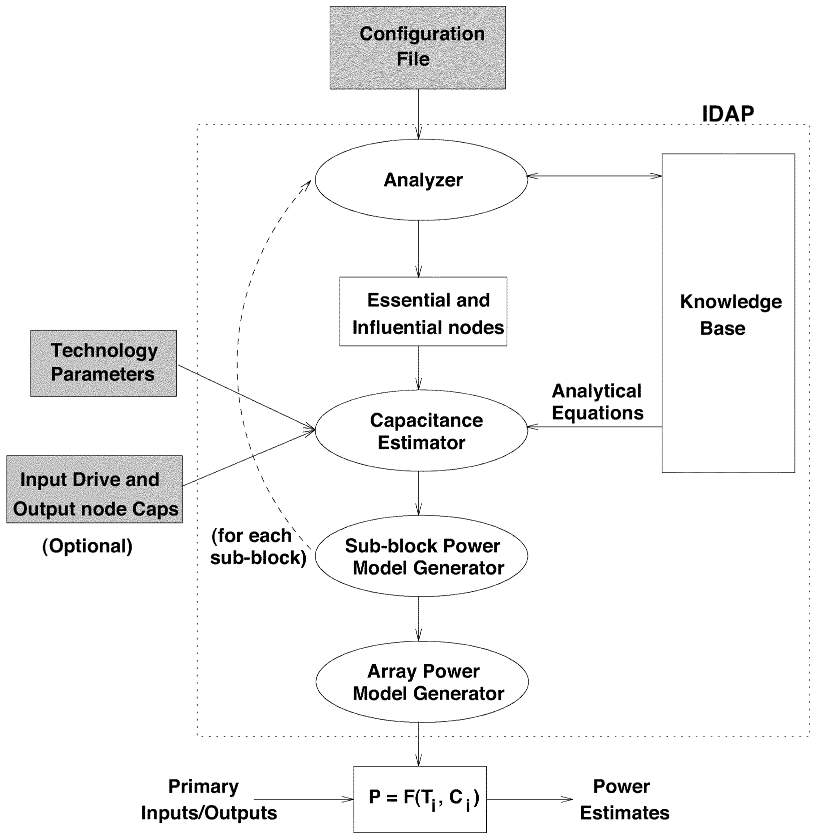

Fig. 3. Proposed methodology for power estimation in array structures.

of all the subblocks to generate the model for the whole array. The details of the configuration file, subblock-power model-generation flow, and the knowledge base are illustrated in the following sections.

B. Configuration File

The configuration file is a high-level design specification of the array which abstracts out the details of the array that determine power dissipation. The parameters enable the specification of a wide variety of arrays used in micro-processor designs. To represent more optimized and complicated arrays the configuration file may be further enhanced and the knowledge base updated for the corresponding implementation styles. The specification is divided into three parts: 1) organization of the array; (2) circuit-level implementation style of each subblock; and 3) design and technology dependent parameters.

Fig. 4 is an example configuration file for a 64 $\times$ 80 tag array. In this example, lines 1–8 specify the organization of the array. This array has 64 rows and 80 columns (lines 1 and 2), memory cell of width 10 μ and height 20 μ (line 3), a single port for both read and write (line 4), a 2:1 read multiplexer, and a 4:1 write multiplexer (lines 7 and 8). The implementation styles for the various subblocks are described in lines 9–11. The read logic is a double-ended bitline, differential sense amplifier (DE:DSA)-based implementation (line 9), the write logic is also a double-ended bitline and precharge transistor (DE:precharge)-based implementation (line 10). The decoder implementation, as indicated in line 11, is as local static CMOS (predecoded address signals act as array inputs). Lines 12–14 are design-specific parameters. Since this array is self-timed, we specify the $perRdBlDis$ parameter (described in Section IV-C) in line 12. Line 13 specifies that the wordline architecture is unified[2] and the position of the read sense amplifiers is indicated in line 14 as after read multiplexer. In high-performance designs, the sense amplifiers are sometimes positioned before the read multiplexer. Lines 13–15 are technology dependent parameters specifying the voltage and frequency of operation for the array (lines 13 and 14) and the path to the process technology file (line 15). The values for the technology dependent parameters, such as the C_{gate}, C_{metal}, and C_{drain}, are extracted from the process file.

The configuration file can also represent more complex array organizations typically found in high-performance microprocessors. Consider an array with a total of 22 columns, a 4:1 write column multiplexer on the first 16 columns of the array, and no write multiplexer for the remaining six columns (the tag array for a LRU-based associative caches can have multiplexed write inputs for tags and nonmultiplexed inputs for LRU bits). Fig. 5 shows the configuration file of such an array.

C. Power-Model Generation: An Illustrative Example

We illustrate the subblock-power model-generation process for a differential sense amplifier-based read logic, with a 2:1 column multiplexer. Fig. 6 shows the implementation of the subblock under investigation. While other implementation styles are possible, this is a commonly used style because of its optimal speed and lower power dissipation.

During a write operation or an idle state there is no transition activity in the subblock and, hence, the dynamic power for this operation/state is zero. However, in the case of a read operation, the transition activity on the bitlines (BL0, BL0_b, BL1, BL1_b), the sense amplifier bitlines (SenseBL, SenseBL_b), and the data-out nodes (DOUT, DOUT b) contribute to the subblock power dissipation. These nodes, which contribute significantly to the subblock power dissipation, are defined as *essential* nodes. The isolation nodes $(\overline{ISO0}, \overline{ISO1})$, bitline precharge

[2]For detailed explanation of unified and divided wordline architectures, refer to [2].

```
########### Organizational Parameters ###########
 1 -rows 64                           # number of rows
 2 -cols 80                           # number of columns
 3 -cellSize 10:20                    # size of the memory cell(width:height)
 4 -rwPorts 1                         # number of read/write ports
 5 -rdPorts 0                         # number of read ports
 6 -wrtPorts 0                        # number of write ports
 7 -rdColMuxSize 2:1                  # size of read column multiplexer
 8 -wrtColMuxSize 4:1                 # size of write column multiplexer
########### Implementation Style Parameters ###########
 9 -rdType DE:DSA                     # Double Ended Bitline,
                                        Differential SenseAmp based read
10 -wrtType DE:precharge              # Double Ended Bitline,
                                        precharge transistor based write
11 -decoder local:static              # local decoder with
                                        static CMOS implementation
########### Design Specific Parameters ###########
12 -perRdBlDis 0.6                    # percentage discharge on
                                        the read bitlines during read
13 -wordline unified                  # unified wordline architecture
14 -rdMuxAfterSenseAmp FALSE          # position of read column multiplexer
########### Technology Dependent Parameters ###########
13 -freq 600MHz                       # frequency of operation
14 -voltage 3v                        # operating voltage
15 -tech tech.filename                # technology file
16 -verbose                           # show details during computations
```

Fig. 4. Input configuration file for a 64 × 80 tag array.

```
############ Configuration Parameters ###########
 1 -rows 16                           # number of rows
 2 -cols 22                           # total number of cols
 3 -cellsize 14:25                    # bit-cell size
 4 -rdports 1                         # 1 read port
 5 -wrtports 1                        # 1 write port
 6 -decoder full:static               # static CMOS based full decoder

 # Two different column types in array (ColType1 and ColType2)
 7 -nColTypes 2 ColType1 ColType2

 8 -cols ColType1 16                  # number of cols in ColType1
 9 -rdtype ColType1 DE:inverter       # ColType1 read logic
10 -wrttype ColType1 DE:precharge     # ColType2 write logic
11 -wrtColMuxSize ColType1 4:1        # write column mux size on ColType1

12 -cols ColType2 6                   # number of cols in ColType2
13 -rdtype ColType2 DE:inverter       # ColType2 read logic
14 -wrttype ColType2 DE:precharge     # ColType2 write logic
```

Fig. 5. Configuration file for a complicated array organization.

node ($\overline{PCH}$), sense bitline-precharge node ($\overline{SensePch}$), and sense-amplifier enable node ($SenseEn$) do not contribute to the power in this subblock. However, they affect the power dissipation in another subblock (read control logic) and will be accounted for in the power-model generation for the read control logic. We define the nodes which contribute to the power dissipation in other subblocks as *influential* nodes. Although all the nodes in this particular subcircuit example are categorized under influential/essential nodes, in subcircuits such as decoders and read/write control logic, many nodes have capacitances negligible compared to others. For instance, in address decoders, the capacitance on the first level of decoding logic is insignificant compared to capacitive load seen by the wordline drivers. Such nodes are termed as *nonessential* nodes and are not considered in the analysis. Essential and influential nodes are determined by the knowledge of the various implementation styles. Traditionally designers can gather this information from experience. Since the circuit-implementation styles across different technologies usually remains the same, prior array designs could be used to extract this information. Our goal is to build a knowledge base that contains the implementation specific information. Typically, there are limited numbers of subblock implementation styles used for a specific microprocessor family. Therefore, corresponding to each implementation style, the influential and essential nodes are determined and stored in the knowledge base.

In order to estimate the power dissipation for various operations, the capacitance on the influential and essential nodes needs to be calculated. This is performed by a set of analytical models that are a function of organizational parameters and attributes specified in the configuration file. For example, the bitline capacitances (BL0, BL0_b, BL1, BL1_b) and bitline-precharge capacitance ($\overline{PCH}$) are determined by (1) and (2), respectively

$$\begin{aligned} C_{\mathrm{BL}} &= N_{\mathrm{rows}}.(C_{\mathrm{memCell}} + C_{\mathrm{metal}}.H_{\mathrm{memCell}}) \\ &\quad + 3.C_{\mathrm{drain}} \end{aligned} \tag{1}$$

$$C_{\mathrm{precharge}} = 3.W_{\mathrm{pmos}}.C_{\mathrm{gate}} \tag{2}$$

where, $W_{\mathrm{pmos}} = f(C_{\mathrm{BL}}, T_{\mathrm{precharge}})$;

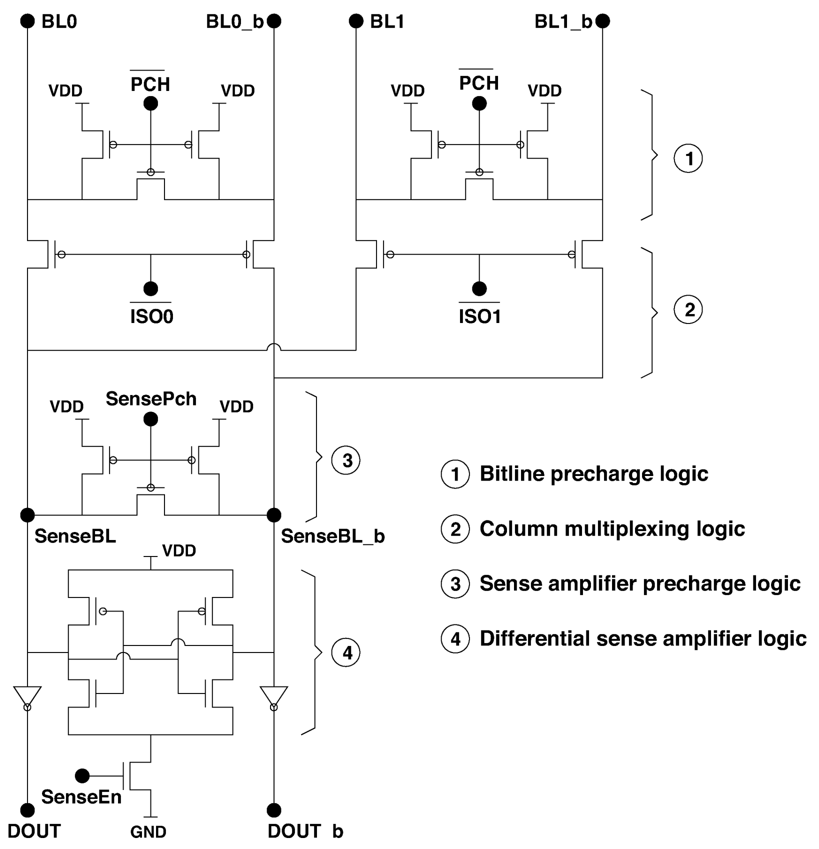
BL0
BL0_b
BL1
BL1_b
VDD
PCH
ISO0
ISO1
SensePch
SenseBL
SenseBL_b
SenseEn
DOUT
GND
DOUT_b
1 Bitline precharge logic
2 Column multiplexing logic
3 Sense amplifier precharge logic
4 Differential sense amplifier logic

Fig. 6. Typical structure of read logic based on differential sense amplifier.

In (1), C_{memCell} is the capacitive load seen by the bitlines in each memory cell of the array column. N_{rows} indicates the number of rows in the array, C_{metal} indicates the metal capacitance per unit micron, H_{memCell} indicates the height of the memory cell in microns, C_{drain} indicates the drain capacitance per unit micron. In (2), C_{gate} is the gate capacitance per unit micron, and W_{pmos} indicates the width of the precharge PMOS transistors. The W_{pmos} is calculated as a function of the bitline capacitance (C_{BL}) and precharge time $(T_{\mathrm{precharge}})$. The $T_{\mathrm{precharge}}$ is derived from the frequency of operation parameter defined in the configuration file. Also, C_{drain}, C_{gate}, and C_{metal} are process technology dependent parameters and H_{memCell} and N_{rows} are organizational parameters specified in the configuration file (more details on the configuration file are given in Section IV-B). The capacitances on all the essential and influential nodes can be calculated using similar analytical equations. In the case of self-timed logic-based [1] reads, the bitlines discharge only partially during read operations. Hence, there is a factor $perRdBlDis$, which indicates the percentage of bitline discharge. The dynamic power model for this subblock can thus be obtained using

$$P_{\mathrm{dyn}} = \begin{cases} 0 & \text{for write operation or idle state} \\ (2.C_{\mathrm{BL}}.perRdBlDis \\ \quad + C_{\mathrm{SenseBl}} + C_{\mathrm{Dout}}).V_{\mathrm{dd}}^2.f & \text{for read operation.} \end{cases} \tag{3}$$

Note that the parameters $perRdBlDis$, V_{dd}, and f in (3) are obtained in the configuration file. A similar analysis is performed on the remaining subblocks to generate a model for the entire array.

D. Knowledge Base

The knowledge base consists of two main components corresponding to each subblock implementation style: 1) the influential and essential nodes in the subblock and 2) parameterized analytical equations which enable calculating the capacitance on the essential and influential nodes (similar to (1) and (2) in Section IV-C). The essential and influential nodes can be determined based on simulation and characterization of prior designs and the designer's experience. The parameterized equations for these priority nodes can then be derived for the given subblock implementation and stored in the knowledge base. The parameterized analytical equations, however, need to be independent of technology. For a given subblock, the knowledge base may initially contain the obvious nodes which contribute to power. Then, based on simulations of prior designs and its power analysis, the knowledge base may be enhanced by capturing more priority nodes which contribute to power. This process can be iterated until the required accuracy is achieved. A tool such as the one described in [8] can be used for the purpose of characterizing arrays and extracting the necessary information to build the knowledge base. Note that once the knowledge base is developed, it may be used earlier in the design cycle for the next generation of processor design and process technology.

V. Model Evaluation

The IDAP tool was developed using C++, and the knowledge base was populated with an analysis of the arrays and SPICE-level simulations. The time taken for developing the knowledge base for almost all subblock implementation styles in arrays from the Motorola family of micro-processors that are PowerPC Book E compliant took 4–5 weeks. Because of the highly repetitive structure of the arrays, the number of distinct priority (influential and essential) nodes for any subblock implementation was observed to be less than 10. In this section, we show the results of the evaluation of the IDAP-based power estimates with those based on fully-extracted SPICE simulations. Since the IDAP tool consists of mostly analytical equations, the time taken to run the tool

TABLE II
COMPARISON OF THE POWER MODELS WITH SPICE

	Size of array	Error	
	(# of bit cells)	READ	WRITE
Array1	352	+22.2%	-20.6%
Array2	704	-6.1%	-13.6%
Array3	1024	+14.4%	-16.1%
Array4	1536	+20.5%	-2.8%
Array5	5120	+0.4%	-3.1%
Array6	5888	+4.6%	+1.2%
Array7	9504	-10.5%	-6.7%

for a given array configuration and stimuli was on the order of 1–2 s, in contrast with SPICE simulations that took on the order of tens of hours (depending on the size of the array).

Table II shows the comparison across different arrays used in an industrial e500 processor core design. The actual power numbers and the names of the array are not shown because they are Motorola proprietary data and cannot be published. Instead, we show the percentage error between the model estimates and SPICE. Column 2 indicates the size of the array in terms of the number of bit cells, Columns 3 and 4 indicate the percentage error in the model estimates for read and write operations respectively. The percentage error is calculated as $(model\_value - actual\_value)/actual\_value$, where, the $actual\_value$ is the value obtained from SPICE. The SPICE simulations are done on a transistor-level netlist with *RC* back annotation obtained from layout. The power values are calculated as the average power for a large number of input stimulus. This stimulus was obtained from the benchmarks: dhrystone, goke_fft, and six Motorola internal benchmarks. The designs are based on 0.13-μm bulk CMOS technology operating at a frequency of 850 MHz. These arrays differ from each other in size, row/column organization, number of memory bit-cell ports (single read/write, multiple read/write, and dedicated read/write), memory bit-cell dimensions, read logic styles, write logic styles, and self-timed read-logic styles.

Some of the main features of the arrays used in the experiments are illustrated below. Arrays 1 and 2 have separate read and write ports for simultaneous read and write accesses. While the write operation was implemented using single ended bitline and static inverter-based write logic, the read operation was implemented using double ended bitline and inverter-based sense-amplifier. Array 3 has multiple read/write ports (5 read and 2 write), each implemented using single ended bitline-based logic. Array 4 has configuration similar to the example illustrated in Fig. 5 with write multiplexer only on a section of columns. Arrays 5–7 have similar subblock implementation styles with single read/write port (double ended bitlines, differential sense-amplifier-based read and precharge-based write), but differ mainly in the size and row/column organization. From Table II the error margin varies from -20.6% to $+22.2\%$. The reasons for variation include:

- differences in the node capacitances estimated versus the actual node capacitances from layout. For example, in differential sense-amplifier-based read logic, illustrated in Section IV-C, the calculated precharge node capacitance ($C_{\text{precharge}}$, (2)) and the actual node capacitance can differ for a number of reasons: margin of error in determination of the width of the PMOS transistor, margin of error induced due to lack of accountability of capacitances associated with vias and coupling capacitances in analytical models.
- various custom design optimizations for speed which are not accounted for in the model. For example, gate skewing [16] in designs leads to reduced node capacitances.
- a margin of error, due to short circuit currents. This is because the short circuit power is assumed to be a constant percentage of capacitive power and accounted in the model using a scaling factor. However, in the actual estimates, the contributions of short circuit currents may vary depending on the internal signal transition time.

It can be noted that because of the reasons illustrated above, the models yield to an overestimate of power in some array designs and an under-estimate in some arrays depending on its implementation. Hence, a variation between -20.6% to $+22.2\%$ in error is seen between the model estimates and the actual power based on SPICE simulations. Although the experiments were conducted on arrays from a specific technology, we think a similar power estimation accuracy will hold for arrays from a different technology because of the technology independent methodology used in the tool.

VI. APPLICATIONS

More general applications of the tool were illustrated in the introduction section. Some of the specific usages of the IDAP tool are listed below: 1) for estimating power values for system designs at the RTL (assuming that the organizational and implementation details of array subblocks are known at RT level); 2) for power-performance tradeoff analysis of various RT and system-level optimizations; 3) to study the effect of different organizational parameters on array power dissipation during early phases of the design; and 4) since the proposed methodology yields power models for each subblock the array designers may analyze subblock power hungry subblocks for optimization for overall array power dissipation. While there are a multitude of places in which IDAP can be deployed, we show some of the possible applications of the tool in the following subsections.

A. Subblock Contributions to Total Array Dynamic Power Dissipation

In this section, using IDAP, we show the percentage contributions of array subblocks to the total dynamic-power dissipation for read and write operations. We considered typical implementation styles for the array subblocks and the contributions are analyzed for varying array sizes, in terms of rows and columns. We think this analysis will help determine the subblocks on which low-power techniques can be focussed.

Figs. 7 and 8 show the percentage-power contributions of the array subblocks for read and write operations, respectively. During a read operation, the read column logic consumes as much as 72% of the total dynamic power. Further analysis showed that the bitlines contribute to the majority of power dissipation in read column logic. The row-decoder contributes to less than 2% of the total dynamic power. The read control logic, which drives control signals to precharge, sense amplifier, and isolation logic, contributes to as much as 30% of the total dynamic power. The contributions of write control and write column logic are not shown because these subblocks do not contribute to any dynamic power during a read operation. Interestingly, it can be noted that these percentage contributions remain the same for arrays which have the same number of rows (independent of the number of columns). This is because the power dissipation in all subblocks increases in proportion to the number of columns. However, when the number of rows increases, because of the increasing bitline lengths, the fraction increase read column logic power is more than that of the other subblocks. So, the percentage contribution of read column logic increases with the increasing number of rows. Similar observations can be made in Fig. 8, showing the percentage subblock power contributions for a write operation. The write column logic dominates the subblock power contributions. It can be observed that the percentage-power contributions of the array subblocks are independent of the array size.

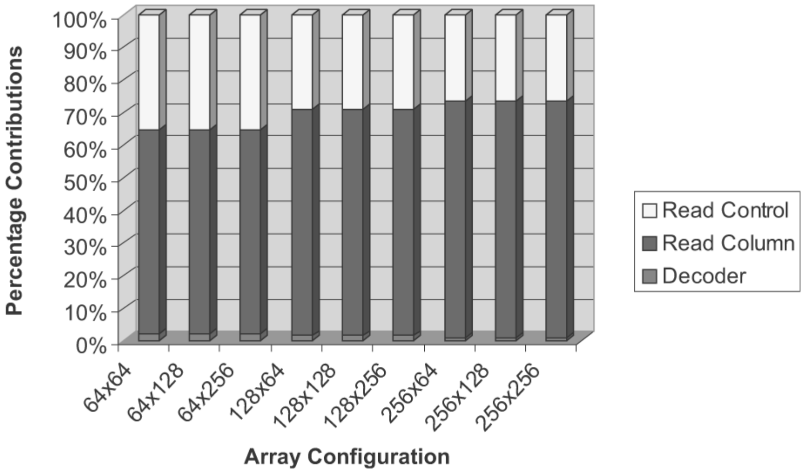

Fig. 7. Subblock-power contributions for read operation.

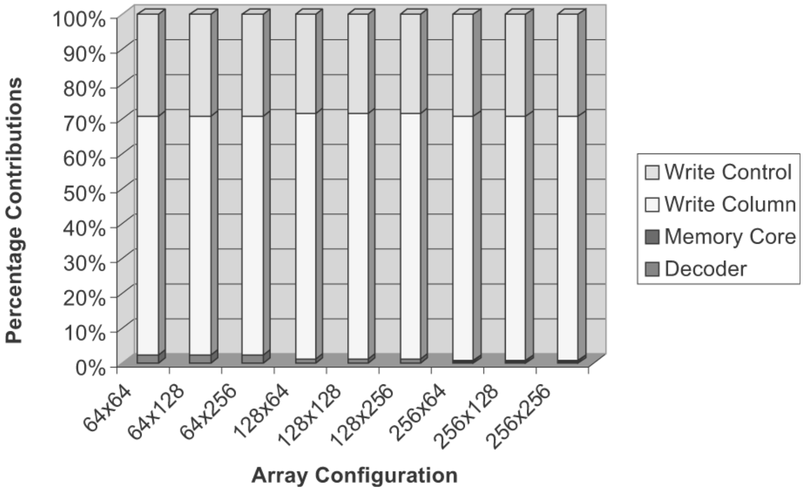

Fig. 8. Subblock-power contributions for write operation.

B. *Effect of Bitline-Voltage Swing on Read Power*

As described in Section VI-A, bitlines consume a significant portion of the total array power. To reduce the power dissipation during a read operation, optimizations based on self-timed read control logic (such as phased clock, delayed clock [1]) are typically used in contemporary array designs. These techniques achieve reduction in power dissipation by reducing the voltage swing on the bitlines during a read operation. The plot in Fig. 9 shows the effect of this percentage bitline-voltage swing on the read array power dissipation. The read power is estimated for arrays of sizes 64 × 128, 128 × 128, and 256 × 128 with typical subblock implementation styles.

In contemporary array designs, the bitlines discharge by just 15% of the supply voltage [3], [20] during a read operation. This means that the optimization results in as much as 63% reduction in array read power dissipation for an 256 × 128 sized array. Also, it can be noted that with an increasing number of array columns, the percentage-power reduction increases because of increasing contribution of bitlines to the total array read power.

C. *Effect of Memory-Cell Dimensions on Power*

The dimensions of the memory cell have a significant impact on the power dissipated in the array. While a memory cell requires a specific amount of area, there is flexibility in choosing its width and height. In this experiment, we vary these dimensions (width and height) within

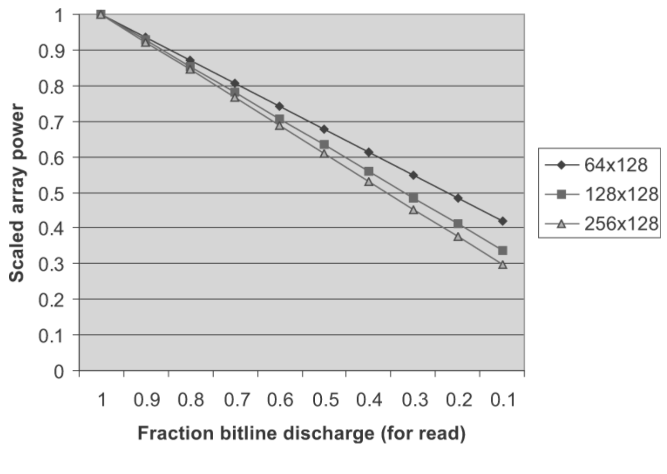

Fig. 9. Effect of bitline-voltage swing on array read power.

an acceptable margin in which the memory cell can be laid out, while keeping the area constant. For example, if a memory cell requires an area of 50 μm^2, the dimensions of the memory cell are varied as $(5 \times 10\ \mu\text{m})$, $(6 \times 8.33\ \mu\text{m})$, and $(8 \times 6.25\ \mu\text{m})$.

Fig. 10 shows the effect of varying cell-size dimensions on the array power for read and write operations. The array used for our experiment is a 64 row and 96 column array, with differential sense-amplifier-based

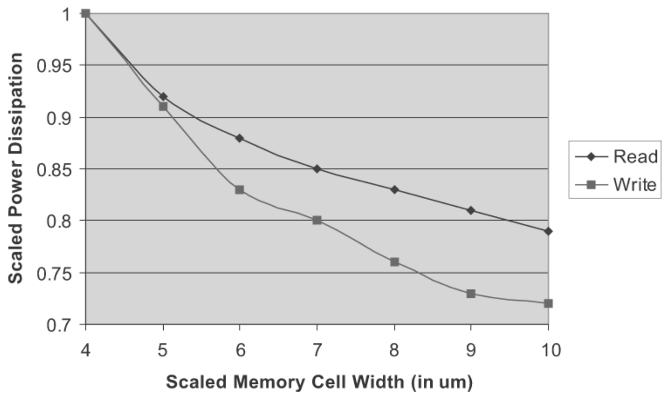

Fig. 10. Plot showing effect of memory-cell dimensions on array power.

read and precharge-based write. As can be seen the dimensions of the cell has a significant effect on the total power dissipation in the array. As much as 21% variation in reads and 28% in writes can be seen in this example. IDAP was used to conduct similar experiments on three other arrays and a similar trend was observed The power dissipation in the array decreases with increasing the width or decreasing the height of the memory cell (since area is constant, height $\propto$ 1/width). This is because the bitline power is linearly proportional to the height of the bit cells and has significant contribution to the overall power. So, although the wordline power increases, because of the increasing bit cell width, the overall power dissipation decreases with decreasing bit cell height. It can be observed that the reduction in power for a write operation is more than that of a read operation. This is because the read operation in the array is based on self-timed logic [1] and hence bitlines discharge by only a fraction of the total precharge voltage. However, in the case of write operations, the columns corresponding to the write discharge completely and, hence the effect of reduced bitline lengths on power is more prominent for a write operation than for a read operation.

Note that changing the dimensions of the memory cell can affect the performance of the array. Since the delay on wordlines increases with increasing width and delay on bitlines decreases with decreasing height, the overall delay for a read and write access may not be affected significantly. However, the aspect ratio of the array layout also changes, which might affect the floorplan.

VII. Conclusion and Future Work

In this paper, we presented a methodology and a tool IDAP, that can be used for accurate power estimation of arrays early in the design cycle. The tool takes the array subblock-circuit implementation styles and its configuration parameters as input and generates a power model for the various operations supported by the array. The generated models were evaluated by comparing against detailed SPICE simulations on leading industrial designs. The error margin is seen to be less than 22.2%. We used the tool to analyze the contributions of the array subblocks to the total power dissipation and also studied the effects of memory cell dimensions and bitline-voltage swing on the array power.

This work can be expanded in a number of ways. The contribution of leakage currents to the total power is becoming increasingly significant in newer technologies. We are currently enhancing the tool to estimate leakage power as a function of the operation on an array. Also, we want to analyze and model the parameters related to short-circuit current for more accurate estimation of dynamic power dissipation. The tool can currently handle only a single bank of memory. However, in larger arrays there are multiple banks and the power dissipation depends on the memory-bank organization. We also plan to enhance the tool to capture the power for multibank configurations.

References

[1] B. S. Amrutur and M. Horowitz, "A replica technique for wordline and sense control in low power SRAMs," *IEEE J. Solid-State Circuits*, vol. 33, pp. 1208–1219, Aug. 1998.
[2] ——, "Fast low-power decoders for RAMs," *IEEE J. Solid-State Circuits*, vol. 36, pp. 1506–1515, Oct. 2001.
[3] ——, "Techniques to reduce power in fast wide memories," in *Proc. Int. Symp. Low-Power Electron. Design*, Oct. 1994, pp. 92–93.
[4] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for architectural level power analysis and optimizations," in *Proc. Int. Symp. Comput. Archit.*, 2000, pp. 83–94.
[5] R. Evans and P. Franzon, "Energy consumption modeling and optimization for SRAMs," *IEEE J. Solid-State Circuits*, vol. 30, pp. 571–579, May 1995.
[6] M. Kamble and K. Ghose, "Analytical energy dissipation models for low power caches," in *Proc. Int. Symp. Low-Power Electron. Design*, 1997, pp. 143–148.
[7] Y. Li and J. Henkel, "A framework for estimating and minimizing energy dissipation of embedded HW/SW systems," in *Proc. Design Automation Conf.*, 1998, pp. 188–193.
[8] M. Mamidipaka, K. Khouri, and N. Dutt, "A methodology for accurate modeling of energy dissipation in array structures," in *Proc. Int. Conf. VLSI Design*, 2003, pp. 320–325.
[9] J. Montanaro *et al.*, "A 160-MHz, 32 b, 0.5-W CMOS RISC microprocessor," *IEEE J. Solid-State Circuits*, pp. 1703–1712, Nov. 1996.
[10] K. Ogawa, M. Kohno, and F. Kitamura, "PASTEL: A parameterized memory characterized system," in *Proc. Design Automation Test Eur.*, 1998, pp. 15–20.
[11] N. Potlapally, A. Raghunathan, G. Lakshminarayana, M. Hsiao, and S. Chakradhar, "Accurate power macro-modeling techniques for complex RTL circuits," in *Proc. Int. Conf. VLSI Design*, 2001, pp. 235–241.
[12] Available: http://www.sequencedesign.com/2_solutions/2b_power_theater.html [Online]
[13] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design*. Reading, MA: Addison-Wesley, 1985.
[14] K. Roy and M. Johnson, "Software design for low power," *NATO ASI Series Low-Power Design in DSE*, pp. 433–460, Aug. 1997.
[15] W.-T. Shiue and C. Chakraborthy, "Memory exploration for low power, embedded system," in *Proc. Design Automation Conf.*, 1999, pp. 140–145.
[16] T. Thorp, G. Yee, and C. Sechen, "Design and synthesis of monotonic circuits," in *Proc. Int. Conf. Computer Design*, 1999, pp. 569–572.
[17] P. Hicks, M. Walnock, and R. Owens, "Analysis of power consumption in memory hierarchies," in *Proc. Int. Symp. Low-Power Electron. Design*, 1997, pp. 239–244.
[18] C. Su and A. M. Despain, "Cache design tradeoffs for power and performance optimization: A case study," in *Proc. Int. Symp. Low-Power Electron. Design*, 1995, pp. 63–68.
[19] S. Wilton and N. Jouppi, "An enhanced access and cycle time model for on-chip caches,", WRL Res. Rep. 93/5, June 1994.
[20] M. Zhang and K. Asanovic, "Highly associative caches for low power processors," presented at the Kool Chips Workshop, 33rd Int. Symp. Microarchit., Monterey, CA, 2000.
[21] V. Zyuban and P. Kogge, "The energy complexity of register files," in *Proc. Int. Symp. Low-Power Electron. Design*, 1998, pp. 305–310.