IDAP: A TOOL FOR HIGH LEVEL POWER ESTIMATION OF CUSTOM ARRAY STRUCTURES

Mahesh Mamidipaka † *Kamal Khouri* ‡ *Nikil Dutt* † *Magdy Abadir* ‡

†Center for Embedded Computer Systems
School of Information and Computer Science
University of California, Irvine, CA 92697, USA
{maheshmn,dutt}@cecs.uci.edu

‡High Performance PowerPC Platforms
Semiconductor Products Sector
Motorola Inc., Austin, TX 78729 USA
{kamal.khouri,m.abadir}@motorola.com

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, parameterized by the array operations, the implementation styles, 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 highly accurate with an error margin of less than 22.2% of detailed (layout extracted) SPICE simulations.*

1. INTRODUCTION

Many factors have contributed to the increased demand for lowering power consumption in todays 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.

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

ICCAD'03, November 11-13, 2003, San Jose, California, USA.

Two observations motivate our research. The first, is the need for reducing power consumption drives a demand for *early* and *accurate* estimates of power for a given SoC on a given application domain. Such early estimates:

- aid micro-architects in rapidly exploring the design space and evaluating various power-performance trade-offs before committing to an architecture.
- allow SOC designers to track the power consumption of their blocks and ensure they are within the required budget.
- enable end users and system integrators to optimize their software [11], system-cooling requirements, and overall design strategy using early power estimates.

The second observation is that array structures such as register files, branch target buffers, tag arrays, and caches consume up to 70% of the overall power in a SoC [3]. It has also been shown that caches alone consume up to 40% of total power [8]. Furthermore, the power dissipation of array structures is greatly dependent on the specific circuit implementation style employed. This motivates the need to develop an accurate high-level power estimation capability that accounts for a variety of array implementation styles. Although there has been a sizable body of work on power estimation in array structures (Section 2 summarizes this research) the focus has either been towards modeling at the micro-architectural level or modeling through characterization after the availability of transistor level design. Models at the micro-architectural level lack accuracy because of the non-availability of design specific information such as sense-amplifier type (differential or inverter based), decoder style type (static CMOS or dynamic CMOS based) etc. Also it is not even possible to represent certain configurations commonly found in contemporary processors using the existing micro-architecture level models. For example, an SRAM of size 64x128 (64 rows and 128 columns) with a single read/write port, allowing a 32-bit write operation

and 64-bit read operation cannot be represented using existing micro-architectural models. On the other hand, models at transistor level, while accurate, are available only in latter stages of the design cycle. Hence, there is a need for accurate power models which bridge the gap between the micro-architecture 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 the accurate estimation of power dissipation in SRAM based arrays using a high-level description of the design which contains micro-architecture level parameters and sub-block 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.

The remainder of the paper is organized as follows. Section 2 summarizes the related work in the area of array power modeling and estimation. Section 3 provides a brief background in array structures and design styles. Section 4 describes our estimation methodology and presents an illustrative example that demonstrates generation of a power model using a high-level configuration file. Section 5 presents validation of our models against transistor-level SPICE to measure the estimation accuracy. Section 6 contains concluding remarks and directions for future work.

2. RELATED WORK

Related work on power estimation in arrays can be categorized into characterization based power modeling and micro-architecture level power modeling. A study of different approaches used for modeling energy dissipation in SRAMs is illustrated by Evans et al. in [4]. In this section we will first detail the research work related to characterization based modeling followed by research in micro-architecture 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 run-times. To address this issue, an ASIC on-chip memory characterization tool (PASTEL) [9] 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 [7].

For estimation of power in arrays at higher levels in the design cycle, micro-architecture level models have been proposed. However, researchers have focussed on modeling specific array implementation styles. For instance, Zyuban and Kogge [17] 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 [15] and also used in [12, 14] with minor enhancements for design space exploration. Kamble and Ghose [5] 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 micro-processors. Energy models specific to caches have also been proposed by Li and Henkel [6]. 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 [16] tool was enhanced for more accurate power estimation, but is applicable only for specific implementations of caches at the micro-architectural level. Brooks et al. proposed parameterizable analytical models to estimate power for different sizes of generic array structures [3]. These models, referred to as Wattch models, are based on the capacitance values estimated using the Cacti [16] tool. These micro-architecture 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 sub-block implementation styles which are not captured in the models. The Wattch models assume typical sub-block circuit implementation styles for power estimation because of lack of such information at the micro-architectural 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 sub-blocks. In fact, experiments done on a 64x128 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 sub-block 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.

3. 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 Figure 1, array structures are primarily composed of address decoders, wordline drivers, memory core, read column logic, write column logic, read control, and write control logic.

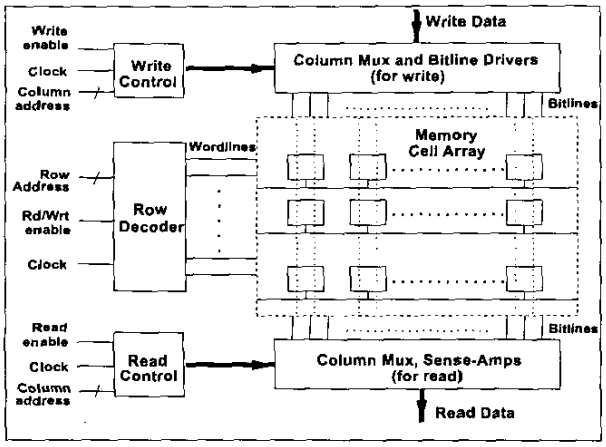

Figure 1: Typical Architecture of Array Structures

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 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 cells consists of the compare logic in addition to static RAM cell [10].

Arrays typically differ from each other in size, row and column organization, and sub-block implementation style at the circuit-level. A specific circuit implementation style is chosen for optimal power or performance. Table 1 lists some examples of the commonly used sub-block 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.

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

Table 1: Array Sub-block Circuit Implementation Styles

4. ESTIMATION METHODOLOGY

In this section, we begin with a brief overview of the proposed methodology, followed by an illustrative example describing the flow. We then formalize our methodology by describing the various stages in detail.

4.1. Overview

Since power dissipation in an array is the additive sum of the power in each sub-block, the IDAP tool generates a model for each sub-block based on information specified in a user provided configuration file. This configuration file contains the specification of the array organization (for example, number of rows and columns), the circuit implementation style for each array sub-block, and technology based parameters for the CMOS process being used. The generated power models are a function of the operation on the array and switching activity in each sub-block. 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.

4.2. Power Model Generation: An Illustrative Example

We illustrate the sub-block power model generation process for a differential sense amplifier based read logic, with a 2:1 column multiplexer. Figure 2 shows the implementation of the sub-block 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 sub-block 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 sub-block power dissipation. These nodes, which contribute significantly to the sub-block power dissipation, are defined as *essential* nodes. The isolation nodes ($\overline{ISO0}, \overline{ISO1}$), bitline precharge node ($\overline{PCH}$), sense bitline precharge node ($\overline{SensePch}$), and sense amplifier enable node ($SenseEn$) do not contribute to the power in this sub-block. However, they affect the power dissipation in another sub-block (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 sub-blocks as *influential* nodes. Although all the nodes in this particular sub-circuit example are categorized under influential/essential nodes, in sub-circuits 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 *non-essential* 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 number of sub-block implementation styles used for a specific micro-processor family. Therefore, corresponding to each implementation style, the influential and essential nodes are determined and stored in the knowledge base.

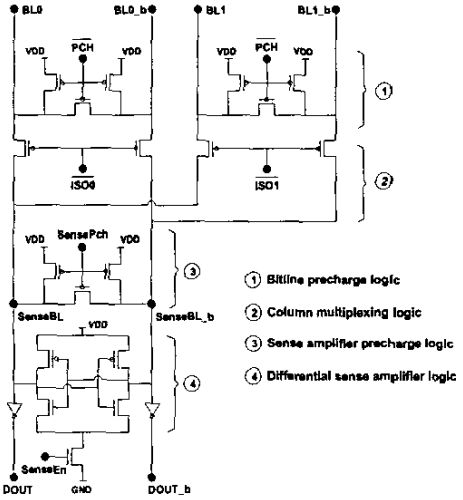

Figure 2: Typical Structure of Read logic based on Differential Sense Amplifier

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 Equations 1 and 2 respectively.

$$C_{BL} = N_{rows}.(C_{memCell} + C_{metal}.H_{memCell}) + 3.C_{drain} \quad (1)$$

$$C_{precharge} = 3.W_{pmos}.C_{gate}; \quad (2)$$
where, $W_{pmos} = f(C_{BL}, T_{precharge})$;

In Equation 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 Equation 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 ($T_{precharge}$). The $T_{precharge}$ 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 4.3). The capacitances on all the essential and influential nodes can be calculated using similar analytical equations. In the case of self-timed logic [1] based 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 sub-block can thus be obtained using:

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

Note that the parameters, $perRdBlDis, V_{dd}, and\ f$ in Equation 3 are obtained in the configuration file. A similar analysis is performed on the remaining sub-blocks to generate a model for the entire array. In Section 4.4 we automate and formalize the flow.

4.3. 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 3 parts: (1) organization of the array, (2) circuit-level implementation

```
########## 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
```

Figure 3: Input Configuration File for a 64x80 Tag Array

style of each sub-block, and (3) design and technology dependent parameters.

Figure 3 is an example configuration file for a 64x80 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 10μ and height 20μ (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 sub-blocks 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 4.2) 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 C_{gate}, C_{metal}, and C_{drain} are extracted from the process file.

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

4.4. Methodology

Figure 4 is the flow diagram of our methodology. The input is a configuration file containing the implementation and organizational details of the array as described in Section 4.3. The analyzer first analyzes the array configuration file to determine the organization of the array and its sub-block implementation styles. Then, the analyzer together with the knowledge base, determines the essential nodes (nodes within the sub-block which contribute to power) and influential nodes (nodes in the sub-block which contribute to power in other sub-blocks) based on parameters specified in the configuration file. More details on the knowledge base are given in the following sub-section. The next phase of the methodology works on these priority nodes along with the process technology parameters to estimate the switching capacitances in the sub-block 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 sub-blocks are then generated as a function of node capacitances and switching activity in the sub-block. 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 Figure 4). 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 sub-block power model generator, as shown in the Figure 4, is repeated for all the array sub-blocks. The final phase of the methodology stitches the power models of all the sub-blocks to generate the model for the whole array.

4.5. Knowledge Base

The knowledge base consists of two main components corresponding to each sub-block implementation style: (1) The influential and essential nodes in the sub-block and (2) parameterized analytical equations which enable calculating the capacitance on the essential and influential nodes (similar to Equations 1 and 2 in Section 4.2). The essential and influential nodes can be determined based on simulation and characterization of prior designs and designers experience. The parameterized equations for these priority nodes can then be derived for the given sub-block implementation and

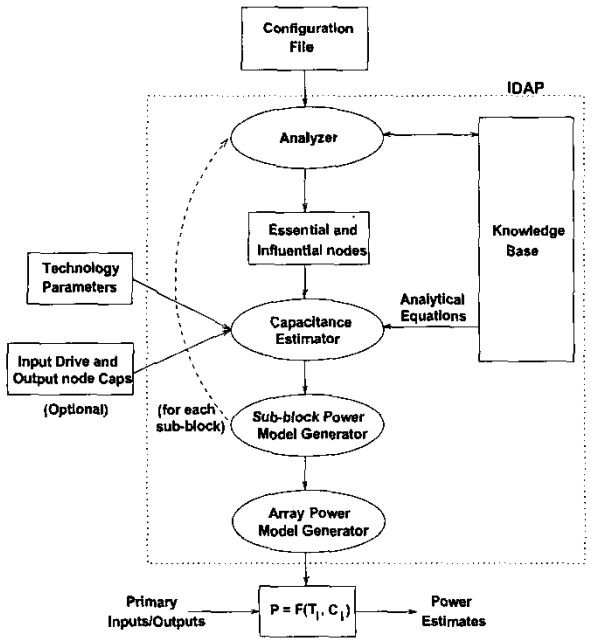

Figure 4: Proposed Methodology for Power Estimation in Array Structures

stored in the knowledge base. The parameterized analytical equations, however, need to be independent of technology. For a given sub-block, 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 [7] 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.

5. 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 sub-block implementation styles in arrays from the Motorola family of microprocessors 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 sub-block 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 for a given array configuration and stimuli was on the order of 1-2 seconds, in contrast with SPICE simulations that took on the order of tens of hours (depending on the size of the array).

Table 2 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 6 Motorola internal benchmarks. The designs are based on 0.13μ bulk CMOS technology operating at a frequency of 850MHz. 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 Figure ?? with write multiplexer only on a section of columns. Arrays 5, 6, and 7 have similar sub-block 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 2 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 4.2, the calculated precharge node capacitance ($C_{precharge}$, equation 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 [13] 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 over-estimate 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.

	Size of array (# of bit cells)	Error	
		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%

Table 2: Comparison of the Power Models with SPICE

6. CONCLUSIONS 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 sub-block 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%.

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 multi-bank configurations.

7. ACKNOWLEDGEMENTS

This work was partially supported by Motorola Inc., and NSF grants CCR-0203813 and CCR-0205712.

8. REFERENCES

[1] B. S. Amrutur, et al. A replica technique for wordline and sense control in low power SRAMs. In *IEEE JSSC*, 1998.

[2] B. S. Amrutur, et al. Fast low-power decoders for RAMs. In *IEEE JSSC*, 2001.

[3] D. Brooks, et al. Wattch: a framework for architectural-level power analysis and optimizations. In *ISCA*, pages 83–94, 2000.

[4] R. Evans, et al. Energy consumption modeling and optimization for SRAMs. In *IEEE JSSC*, pages 571–579, 1995.

[5] M. Kamble, et al. Analytical energy dissipation models for low power caches. In *ISLPED*, 1997.

[6] Y. Li and J. Henkel. A framework for estimating and minimizing energy dissipation of embedded HW/SW systems. In *DAC*, 1998.

[7] M. Mamidipaka, K. Khouri, and N. Dutt. A methodology for accurate modeling of energy dissipation in array structures. In *VLSI Design*, pages 320–325, 2003.

[8] J. Montanaro. et al. A 160-MHz, 32b, 0.5-W CMOS RISC microprocessor. In *IEEE JSSC*, pages 1703–1712, Nov' 96.

[9] K. Ogawa, M. Kohno, and F. Kitamura. PASTEL: A parameterized memory characterized system In *DATE*, pages 15–20, 1998.

[10] N. Weste and K. Eshraghian. *Principles of CMOS VLSI Design*. Addison-Wesley, 1985.

[11] K. Roy, et al. Software design for low power. In *NATO ASI Series on Low Power Design in DSE*, Aug. 1996.

[12] W-T. Shiue and C. Chakraborthy. Memory exploration for low power, embedded system In *DAC*, pages 140–145, 1999.

[13] T. Thorp, G. Yee, and C. Sechen. Design and synthesis of monotonic circuits. In International Conference on Computer Design, 1999.

[14] P. Hicks, M. Walnock, and R. Owens. Analysis of power consumption in memory hierarchies In *ISLPED*, pages 239–244, 1997.

[15] C. Su et al. Cache design tradeoffs for power and performance optimization: a case study In *ISLPED*, pages 63–68, 1995.

[16] S. Wilton and N. Jouppi. *An enhanced access and cycle time model for on-chip caches*. WRL Research Report 93/5, June 1994.

[17] V. Zyuban, et al. The energy complexity of register files. In *ISLPED*, pages 305–310, 1998.