Guiding Architectural SRAM Models

Banit Agrawal Timothy Sherwood
Department of Computer Science, University of California, Santa Barbara
Email: {banit,sherwood}@cs.ucsb.edu

Abstract— **Caches, block memories, predictors, state tables, and other forms of on-chip memory are continuing to consume a greater portion of processor designs with each passing year. Making good architectural decisions early in the design process requires a reasonably accurate model for these important structures. Dealing with continuously changing SRAM design practices and VLSI technologies make this a very difficult problem. Most hand-built memory models capture only a single parameterized design and fail to account for changes in design practice for different size memories or problems with wire scaling. Instead, in this paper we present a high level model that can be used to make simple analytical estimates. Our model is built using the characterization of almost 60 real memory designs from the past 15 years. Our model and the presented methodology can be used to calibrate even more detailed memory models for better accuracy. Despite all of the things that could have gone wrong over the past 15 years, we show that the memory density and delay can be estimated with simple and intuitive functions and we present a technique to automatically extract important scaling trends that can be used to make accurate estimates across a variety of technology and architectural parameters.**

I. Introduction

The amount of the chip real estate devoted to memory has continued to grow at an astounding rate and consequently there is a continuous effort to develop faster, smaller, and lower power memory tiles. Because of the importance of on-chip memory in both modern processor and ASIC designs, making an accurate characterization of those memory technologies available to the designer at an early stage is critical in making good design decisions. While this is already a problem in cache design, many new architectures are built around the idea of many small tiles, and may be especially susceptible to these issues.

New designs are continuously being developed to trade off performance, power consumption, complexity, area, and a host of other design parameters. The best trade-off points shift and change over the years due, in part, to the fact that the underlying VLSI technology itself is a moving target. New problems such as poor wire scaling and leaky transistors make traditional designs sub-optimal and so new circuits are designed to compensate for these limitations. The breadth of these interacting issues makes a very complex environment for design exploration. Even understanding the different trends independently requires specialized knowledge across a wide range of disciplines. For example, if a detailed memory model is built for a particular design and technology, it cannot adapt well with changing technology and optimized designs. To attack this problem efficiently, the first thing that is needed is a careful and thorough evaluation of a large number of past designs that cover a wide range of design methods and technologies. Therefore, in this paper we present the methods necessary to perform such a study, and describe the scaling trends that we extracted from almost 60 reference designs taken over a period of 15 years. To the best of our knowledge, we are the first to put together a memory data set comprehensive enough to allow automatic extraction of analytical models that capture aspects of shifting design-methods and technology. We show how our model accurately captures the most important design scaling factors and how important ”rules-of-thumb” can be extracted from our methods.

One of the issues in dealing with real data is that there are outliers that hide the important trends. In Section IV we describe how important ”rules-of-thumb” can be extracted from our data in a way that is robust in the face of these outliers, and in Section V we show the actual scaling parameters that describe the last 15 years of SRAM designs. While this approach is by no means a replacement for more detailed hand built models that capture the tradeoffs at a level that requires a deep circuit-level understanding, our model can be used to verify and recalibrate these more detailed models as various design practices and scaling factors evolve over time. For example, we find that our model exhibits similar scaling trends for area with size, when compared against a detailed model (*Cacti* [1]). Although *Cacti* provides better relative results, it overestimates the area by 20% in past CMOS technolgy, and 40% in current CMOS technology. We recalibrate Cacti to get more accurate results by incorporating the trends in technology scaling and evolving design practices.

II. Related Work

We are clearly not the first to consider the problem of modeling and forecasting SRAM characteristics. Over the last decade there has been many proposed models aimed primarily at addressing the impact of the memory wall. Most of this research has been focused on low-level circuit characterizations to model the on-chip memory either through reference designs [2], [3], [4], [1], [5], [6] or regression techniques [7], [8], but thus far we have seen no large scale studies/models based on a wide range of published designs and their scaling trends. While there is a wide variety of papers in the area, due to space constraints we concentrate on those that consider delay and area, especially those that consider the effects of scaling.

Mulder et al. [2] proposed a simple area model for on-chip memory such as register files and caches. Their model is based on the area of a single register bit cell known as register bit equivalent (*rbe*). The area of the SRAM cell and

1-4244-9707-X/06/$20.00 ©2006 IEEE

DRAM cell is expressed in terms of *rbe* and a simple analytical formula is provided to calculate the area of direct-mapped cache and set associative caches again in terms of *rbe*. While this model is analytic, it does not incorporate the current design practices such as partitioning and sub-banking. Wada et al. [3] provide an analytical access time model for on-chip cache memories which is dependent on various cache parameters. This model is derived from a reference design almost 13 years ago, and provides no easy method of scaling and optimizing as technology changes.

Cacti [1], a widely used and powerful cache modeling tool, helps architects evaluate various on-chip cache designs. It can be easily extended to model on-chip SRAM as well. The initial version of this tool [4] only supported the access time of set-associative caches. Subsequent versions have extended this model to support access time, dynamic power, and area of set-associative and fully-associative caches. Cacti has been proven to be a very useful modeling tool for architects wishing to study various architectural trade-offs, optimize designs, evaluate the relative merits, and estimate the hardware overhead. Although Cacti provides high accuracy in comparing different designs of on-chip memory, the accuracy is not good in absolute terms compared to the state of the art SRAM circuit design methods. Cacti was designed for 0.80 μm CMOS technology and it used "fudge factor" to approximate the effect of changing technology. This is an incredibly important feature for architects and has most certainly added to Cacti's longevity. However, if the technology scaling factor does not reflect reality, this can lead to error in the estimates. In the later sections we will describe how to perform these comparisons and we will use our analytic model to help recalibrate the Cacti model with current design practices.

Cacti tool was further extended by Mamidipaka et al. [5] to support the modeling of write operation power, static power, and transistor width variation. It also uses fudge factor approximation to model delay and area for newer technology. Mamidipaka et al. also provide a high level power estimation tool (IDAP) [6] for SRAM data array that accounts for different circuit styles by feeding all low-level circuit parameters (cell area, sense amplifier design, etc) as input.

Amrutur et al. [9] provide an analytical model for calculating delay, power, and area of on-chip SRAM at a very low level using circuit level parameters. Then, they simplify the formula and also show the scaling trends with size and technology. While our work is parallel, it has several significant differences, the most important one being that instead of bottom-up it is top-down. By capturing the parameters from almost 60 different designs, we can identify trends not only in technology, but also in the way that circuit designers adjust and react to changing technology parameters [10]. Our model can be used to validate and calibrate more detailed models that tradeoff internal parameters, and can adapt to changes in any number of changing trends automatically.

Some optimization and regression techniques [7], [8] have been proposed to model on-chip memory. Coumeri et al. [7] uses stepwise linear regression based techniques to model on-chip memory. This approach mainly focuses on power and uses SPICE results, which demonstrates a slight bias towards the particular on-chip memory design. According to our finding, we also show that linear regression will not be able to truly capture the best fit to the model for a set of current best reference designs. In an another approach by Schmidt et al. [8], a black box modeling approach has been used to model only the power of on-chip memory. Although their approach uses non-linear regression, they only provide low-level circuit model for only one design constraint and the model is biased towards a particular SRAM design and a particular CMOS technology.

Unlike the past memory models, our model is built from almost 60 reference designs which represent the best design practices in academia and industries. Our model can accurately predict the scaling trends with architectural parameters and CMOS technology and provides a simple analytical equation for mathematical optimization. Moreover, our model and modeling methodology can be further used to recalibrate Cacti to achieve better accuracy. As shown later in the paper, recalibrated Cacti can reduce the overestimation error in area from 20-60% to less than 10%.

III. SRAM Scaling

Before we discuss our generic model, we briefly discuss the internal structure of an SRAM array, and the various parameters and constraints as it relates to our modeling problem.

A. A Simplistic Model of SRAM

A typical SRAM cell usually consists of six transistors, where four transistors are used as pair of inverters to store the bit. Reading/writing to this bit is controlled by two more transistors with *wordline* and *bitline*. For a two ported SRAM cell, the number of *bitlines* and *wordlines* doubles compared to a single ported SRAM cell. The length of the *wordline* and *bitline* wires and the height and width of a cell have the largest impact on the overall delay, power, and area. A SRAM array is usually divided into sub-arrays to reduce the length of the *wordline* and the *bitline*, which in effect reduces the delay and power. More description on various organization of SRAMs can be found in [4], [3], [9].

To understand our results, we need to compare with how we might *expect* technology scaling to impact area and delay. With each new technology the feature size is reduced, which effectively decreases the length and width of each component. In fact, with a decrease in feature size of a factor of 2 we should expect a factor of 4 decrease in area (n^2). Delay is a little more challenging. Because delay is strongly dependent on the capacitance of the wordlines and bitlines, and because the capacitance is approximately a linear function of the length of those lines, we might expect the delay to decrease as a function of the length of the wordlines and bitlines. If the feature size drops by a factor of 2, the area is now $1/4$ of what it was, and the bit lines are now approximately $\sqrt{1/4} = 1/2$ the length making the design 2 times faster.

TABLE I

AREA COMPARISON FOR ON-CHIP SRAM

SRAM Configuration	Estimated area by Cacti in mm^2	Published area in mm^2
0.18μm 64Kb 1-port	0.68	0.38 [12]
0.15μm 64Kb 2-ports	1.15	0.70 [12]
0.09μm 144Kb 1-port	0.675	0.41 [13]
0.5μm 1 Mb 1-port	95.47	78.8 [14]

We can also consider a simplistic model of how the area and delay scale with the size of SRAM. If we double the number of bits in the SRAM, we would expect the area to grow by a factor of two as well. Again assuming that delay is a direct function of line size, this should increase the delay by a factor of $\sqrt{2}$.

Clearly these models of memory area and delay are overly simplistic (there are many internal knobs designers turn), but the main idea behind these models drives the development of more complex ones. Our goal is to capture these set of best design practices in an adaptive way to evaluate scaling trends, and provide a high-level architectural model.

B. Recalibrating Models

Many of the prior approaches were designed with the goal of performing relative studies, for example answering such questions as, what are the best number of sub-banks to use in a design. While there are many studies in which the accuracy between different SRAM design options is important, many designers and researchers attempt to use these tools to answer absolute questions of design such as, is this space better used for a cache or some custom piece of logic. By far, the favorite tool for architects to use for answering such questions is Cacti [1]. The SRAM model internal to Cacti is an important part of overall tool as it is used to estimate the area and delay of the tag and the data. Even though Cacti was built for modeling caches, it has been used to estimate a wide range of memory aspects [11].

In table I, the area of several SRAM configurations are given. As we see from the table that Cacti overestimates the area by 20-60%. Hence, we need to recalibrate Cacti models based on the best known design practices of today. Plotting similar data as a function of the technology can further illustrate this point. Figure 1 shows the Cacti results and published results for 64kb SRAM configuration as the technology decreases from 0.18 μm to 0.13 μm. While these few points may not be representative of the entire spectrum of designs, it does show that the technology may not scale squarely in case of published results and the absolute results need to be recalibrated. While we will later show that Cacti does a good job of estimating the scaling trends and is excellent for relative studies, it can overestimate the absolute value of area by some fraction.

IV. MODELING APPROACH

Extracting a model from real designs that vary in technology, design method, organization, and size is a challenging problem and we must make some assumptions. First, we need to find the trends that govern the *majority of*, but not necessarily *all*, past published results. If a highly experimental memory is developed and published, and we extract that data, our model should be robust enough to handle that gracefully. If that design point later turns out to have serious concerns, such as problems with reliability or manufacturability, it will surely be an outlier from the rest of the designs and should be given little weight by our fitting methods. If on the other hand, the technique is useful and implementable, many related papers and design points will soon follow. That cluster of points should be recognized as important to the model. The second assumption that we need to make is that these devices are capable of being modeled in a way that makes intuitive sense to a designer and that the parameters we have chosen, CMOS feature size, size of the memory in bits, and the number of ports, actually determine the area and delay of a modern SRAM to a high degree. If we are to make sense of the important trends discovered by our model, we need to understand them in comparison to the simple models from Section III. For example, while we could fit our data to some polynomial of high degree, yet it is not likely that this will yield any fruitful understanding of the underlying trends.

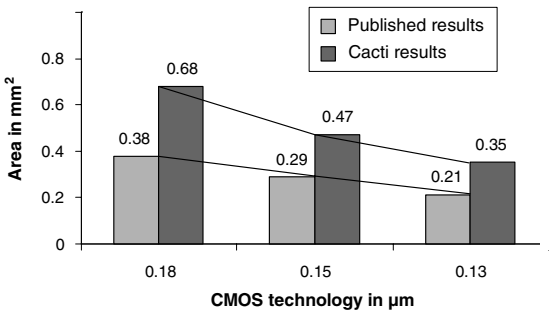

Fig. 1. Effect of technology variation on area for 64kb 1-port SRAM.

Our data points are taken from the published results for on-chip SRAM over the last 15 years by extensively scanning through all well known conferences proceedings and journals in solid state circuits, VLSI, memory design and architecture fields. Table II shows the references[1], memory configurations (*technology, size in bits, port*), and published results (*area in mm^2, cell area in μm^2, and access time in ns*). In addition to the academic papers, we included several points from Virtual silicon's memory compiler for UMC foundry. There are also some very recent results included, such as the 70 Mb SRAM memory in 0.065 μm technology by Intel Corporation.

Starting with design points extracted from memory compiler datasheets and published results we begin with a quick GRG search [15] to narrow in on the most likely range of parameters. Using this, along with the intuitive models for scaling discussed in Section III, we build a set of models which is generic enough to capture the scaling trends. These models

[1]Most of the references are in the form of *publication-startpage-year* to save space in the references list. For example, *jssc-p1047-90* reference means that the sram data is from Journal of solid state circuits (JSSC) in the year 1990 with 1047 as the starting page number in the proceeding. Similarly, isscc stands for International symposium on solid state circuits and virtual-silicon datas are from the datasheets [12].

TABLE II

ON-CHIP SRAM DATASET FROM PAST PUBLISHED RESULTS AND DATASHEETS

References	tech in um	size in $bits$	ports	cell-area in um^2	area in mm^2	delay in ns
virtual-silicon	0.13	65536	1	-	0.21	1.8
virtual-silicon	0.13	65536	2	-	0.43	2
virtual-silicon	0.13	18432	2	-	0.16	1.43
virtual-silicon	0.15	65536	1	-	0.29	1.5
virtual-silicon	0.15	6536	2	-	0.7	1.67
virtual-silicon	0.15	18432	2	-	0.25	1.46
virtual-silicon	0.18	65536	1	-	0.38	1.81
virtual-silicon	0.18	65536	2	-	0.91	2.2
virtual-silicon	0.18	18432	2	-	0.33	1.78
jssc-p564-00	0.25	1048576	1	12	72.03	0.55
jssc-p1631-00	0.18	16777216	1	1.93	54.08	2.5
jssc-p684-04	0.09	262144	1	1.25	0.52	2.8
isscc-p354-98	0.25	32768	1	21.6	2	-
isscc-p352-98	0.4	262144	1	12.5	-	60
isscc-p266-00	0.18	18874368	1	4.23	114.4	2.73
isscc-p190-99	0.18	294912	1	4.8	2.19	1.4
isscc-p460-03	0.09	147456	1	1.16	0.41	0.33
jssc-p1047-90	1	262144	1	82.08	47.557	8
jssc-p1057-90	0.8	1048576	1	45.05	112.56	5
jssc-p1063-90	0.55	4194304	1	19.04	143.22	15
jssc-p439-91	0.8	262144	1	-	42.5	6
jssc-p167-92	0.5	65536	1	17.5	77.88	-
jssc-p649-92	0.8	589824	1	95	97.7	3.5
jssc-p1490-92	0.4	16777216	1	8	228.63	12
jssc-p1504-92	0.55	4194304	1	18.56	165.48	6
jssc-p1511-92	0.3	1048576	1	6.6	29.304	7
jssc-p478-93	0.5	1048576	1	27.36	78.804	6
jssc-p484-93	0.8	73728	2	-	45.5	-
jssc-p1119-93	0.35	16777216	1	8.307	212.67	9
jssc-p1125-93	0.25	16777216	1	2.3	110.24	15
jssc-p1362-93	0.8	262144	1	67.562	47.294	5.8
jssc-p411-94	0.4	6291456	1	7.1552	215.27	12.5
jssc-p1317-94	0.4	16777216	1	8.52	258.57	4.5
jssc-p1344-94	0.5	262144	1	58	121	1.5
jssc-p480-95	0.25	4194304	1	3.84	46.56	6
jssc-p487-95	0.25	16384	1	-	13.53	2.6
jssc-p491-95	0.3	73728	1	30.24	5.25	0.8
jssc-p1189-95	0.25	4194304	1	8.19	112.8	3.3
jssc-p1196-95	0.4	32768	1	31.05	4.06	1
jssc-p1286-95	0.35	1048576	1	34.32	63.456	3.9
jssc-p1443-96	0.3	1205862	1	30.24	210.25	0.9
jssc-p1610-96	0.3	4194304	1	9.2	84.75	6
jssc-p870-05	0.13	16777216	1	0.78	-	-
jssc-p895-05	0.065	73400320	1	0.5704	-	-
jssc-p793-98	0.35	32768	1	31.46	2.184	-
jssc-p1650-98	0.25	4718592	1	9.84	128.15	1.8
jssc-p1659-98	0.25	32768	1	21.6	2	-
jssc-p1571-99	0.25	2359296	1	6.156	30.77	2
isscc-p50-91	0.8	524288	1	85.5	111.36	4
isscc-p54-91	0.6	4194304	1	20.72	156.98	7
isscc-p128-90	0.5	4194304	1	20.3	135.87	23
isscc-p136-90	0.8	1024	1	41	90.48	6.5
isscc-p146-96	0.3	4718592	1	7.82	124.2	1.8
isscc-p148-96	0.5	1048576	1	34.56	88.453	5.4
isscc-p156-96	0.25	294912	1	9.9	5.4481	2
isscc-p206-92	0.4	4096	1	10.2	-	-
isscc-p210-92	0.5	4194304	1	19.95	163.56	9

map the input variables, through a set of equations and scaling constants, to the output parameters. We then apply model fitting techniques to find the optimal setting of the scaling constants. Then, using robust estimation, we filter suspicious data points (*outliers*) and perform error analysis. The block diagram for our modeling approach is shown in Figure 2.

A. Exponent-product model

Building on the intuition from Section III, we use an exponent-product model to study the scaling of delay and area. This model assumes that area and access time are proportional to product of *tech*, *bits*, and *port*, each taken to some fixed exponent. The analytical equation for this model is given in

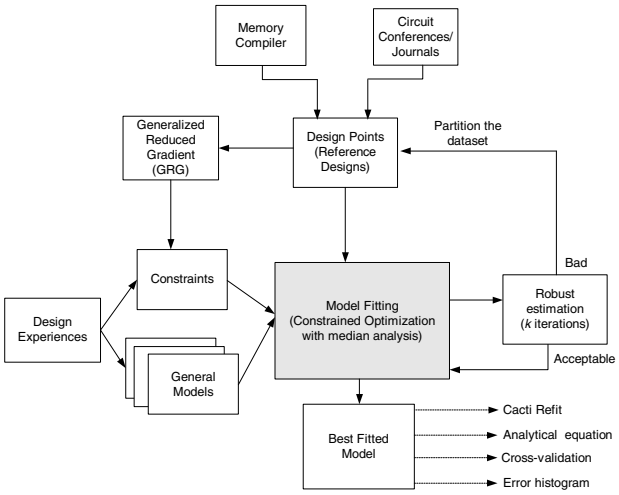

Fig. 2. Block diagram of our modeling approach. Model fitting using constrained optimization along with robust estimation method ensures best fitted results. The output of the best fitted model can also be used to tune Cacti to get accurate absolute results.

equations 1 and 2.

$$area = p_{area} * (tech)^{a_{area}} * (bits)^{b_{area}} * (port)^{c_{area}} + k_{area} \quad (1)$$

$$delay = p_{delay} * (tech)^{a_{delay}} * (bits)^{b_{delay}} * (port)^{c_{delay}} + k_{delay} \quad (2)$$

If the intuitive idea of scaling laid out in Section III was found to be true, we would expect a_{area} to be exactly 2 and b_{area} would converge exactly to 1. The coefficient a_{delay} would be 1, and b_{delay} should be sqrt, or exactly 0.5.

By fitting these models to real data we can find out whether after all the fabrication variations, all the circuit design improvements, and all the memory layout optimizations, if SRAM is really scaling at the rate at which we would expect. The coefficients we find for (p, a, b, c and k) will decide the fit to our dataset of the past published results. Furthermore we can fit this model to Cacti, and see how Cacti estimates these same scaling factors. Then, this information can be used to calibrate Cacti to get better accuracy.

B. Model fitting

We use constrained minimization techniques to find the values of the coefficients that fits best to our dataset of best reference designs. The minimization function for our constrained optimization problem is the sum of absolute percentage error, which is shown in equation 3 for delay. We use an equivalent minimization function for area.

$$error\_sum_{delay} = \sum_{all\ points} error_{delay}$$
$$error_{delay} = \frac{abs(delay_{dataset} - delay_{calc})}{max(delay_{dataset}, delay_{calc})} \quad (3)$$

Given that our errors are highly non-gaussian, we found that the sum of absolute error balanced our need to fit many points without weighing outliers too heavily. We did not use the GRG method [15] for our full optimization runs as it can get caught in local minima depending on the starting points. Instead, we use an iterative method where we vary each coefficient with the required precision and we improve our minimization whenever possible and update the best coefficient found so far. At the end of all iterations, we have the best possible values from the minimization function and the final values of all the coefficients that fits best to our selected model. To ensure that we do converge on a model for at least a majority of the data, we also use a median analysis which enforces a certain percentage of total design points to be fitted within a designated acceptable percentage error. The resulting hyperparameters [16] from this analysis can also be tuned to get better estimates of the coefficients. We also run some cross-validation experiments to provide guaranties for generalization ability of our model to previously unseen data points [17]. While we use the exponent-product model for SRAM in our modeling framework, this framework can also be used to build even more complex models for different types of memory.

V. Results

Using the modeling approach discussed in Section IV, and the dataset off of which we base our results, we find the best fitted coefficients for the area and delay model. To show the robustness of our modeling coefficients, we tune hyperparameters with cross-validation, which we describe in Section V-B. We also feed the Cacti results to our model and find the fitted coefficients. The difference between these two will tell us how the models differ in their treatment of scaling, and will help us recalibrate Cacti if needed.

A. Model coefficients for Area and Delay

Area - After fitting the equations using the methods described in Section IV, we find analytical equation 4 (in Figure 3) as our on-chip SRAM area model. Using our robust estimation and optimization framework we find the values of coefficients p_{area}, a_{area}, b_{area}, c_{area}, and k_{area} to be 0.001, 2.07, 0.9, 0.7, and 0.0048 respectively. By substituting the coefficients into equation 1, it becomes equation 4. As we will describe later, this equation is quite accurate across the entire range of designs.

We can see that the relationship between CMOS technology and area is nearly quadratic in nature as expected, but that the relationship with number of bits is sub-linear. This actually makes sense, as a larger SRAM is more capable of amortizing the overheads from the decoders and sense amps. We find similar results when extracting data from Cacti. For Cacti we find the coefficients p_{area}, a_{area}, b_{area}, c_{area}, and k_{area} to be 0.001179, 1.97, 0.92, 1.21, and 0.0012 respectively. As we can see that there is a similar scaling trend in area with size and CMOS technology. In terms of ports, we do not have enough data points for our results to be statistically significant, but a value of less than one seems to be intuitive.

While we see a similar scaling trend in both Cacti and our modeled results, there is a noticeable difference in the proportionality constant p_{area} and a_{area}, which captures the over-estimation of Cacti results. We can apply a new multiplication factor that can be multiplied with Cacti results

$$area \quad (in \quad mm^2) = 0.001 * (tech)^{2.07} * (bits)^{0.9} * (port)^{0.7} + 0.0048 \tag{4}$$

$$delay \ (in \quad ns \ ) = 0.27 * (tech)^{1.38} * (bits)^{0.25} * (port)^{1.30} + 1.05 \tag{5}$$

Fig. 3. Area and Delay models with scaling factors extracted from published circuits data

to get accurate absolute results. We call this factor as *area-recalibration-factor* and its value is given in equation 6.

$$\begin{aligned} area\text{-}recalibration\text{-}factor &= (0.001/0.001179) * \\ tech^{2.07-1.97} &= 0.85 * tech^{0.1} \end{aligned} \tag{6}$$

Delay - We use analytical equation 2 to model the *delay* of on-chip SRAM, and again fit our coefficients for published data. Equation 2 captures the best design practices well and is found to be reasonably accurate. We use the published results to find the best fitted coefficients for this analytical equation. For the delay model, we find the values of coefficients p_{delay}, a_{delay}, b_{delay}, c_{delay}, and k_{delay} to be 0.27, 1.38, 0.25, 1.30, and 1.05 respectively. Hence, equation 2 becomes equation 5 (shown in Figure 3).

We find that the delay varies approximately as $\sqrt[4]{SRAMsize}$. Furthermore the delay of the SRAM is getting better with technology with a coefficient of 1.38 instead of 1. (recall that with each year tech gets smaller, and hence a large exponent will mean a lower delay). We can infer that SRAM designers are creating better delay-optimized SRAM as technology progresses. Using Cacti results, we find that delay varies with approximately the cube root of SRAM size, which is closer to our modeled results. The reason that these models are better than the $\sqrt{n}$ model (described in Section III) is that designers internally partition and sub-bank their designs to keep bitlines and wordlines small.

B. Scaling trends with technology

To analyze how our model adapts and scales with newer technology, we formulate a cross validation experiment with hyperparameter tuning. In this experiment, we remove all the points for two latest CMOS technologies from the dataset and fit the area and delay model with the remaining design points. Then, we predict the area and delay for these design points of newer CMOS technologies with the fitted model and validate the predictability in our model using cross-validation technique.

Area - To cross-validate the area scaling with technology for our model, we cull two design points of $90nm$ and three points of $130nm$ CMOS technology from the dataset. We predict the area from the model fitted for the remaining dataset and find that our model predicts the future design points with an average error of less than 10%. We also show the Cacti results and recalibrated Cacti results for these design points in Figure 4. It is clear from the figure that while our model and re-calibrated Cacti model predicts the area of $90nm$ and $130nm$ design points with high accuracy (less than 10% error), Cacti overestimates the area of almost all design points by a large amount (20-60%). Hence, our models and the modeling approach helps Cacti to get closer to reality.

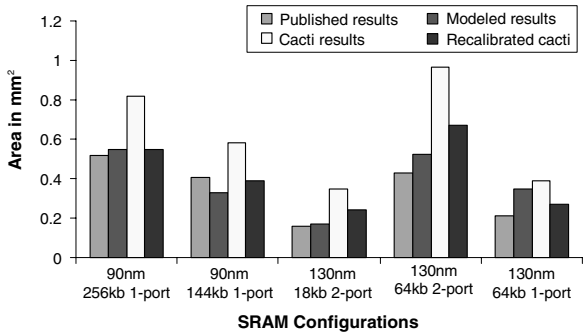

Fig. 4. Predicting the area for newer technology using our model.

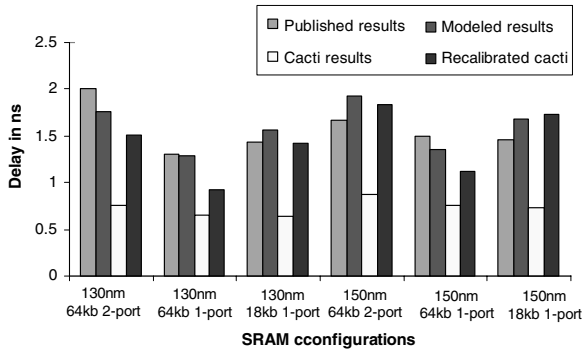

Fig. 5. Predicting the delay for newer technology using our model.

Delay - Delay scaling with technology is analyzed by picking three points of $130nm$ and three points of $150nm$ CMOS technology and predicting the delay of these points from the fitted model for the remaining dataset. We show these predicted results along with Cacti results in Figure 5. We find that our model can predict the delay for next two-generation CMOS technology from older technologies with an average error of just 9.4%. But Cacti provides much lower delay compared to the published results and this may be due to the optimum number of subbanks, or optimum number of *wordline* and *bitline* divisions used inside Cacti. We find that recalibrated Cacti's results are very close to the published results with an average error of less than 10%.

C. Model error analysis

In this subsection, we provide error analysis of our model in the form of error histogram. This histogram also shows the enforcement of our median analysis described in Section IV. We also take the cross-validation error histogram into account to compare the model fitting errors when some points are removed from the dataset. We measure the difference between these two histograms for both delay and area using χ^2 distance metric [18].

Area - The error histogram for area is shown in Figure 6 for both the predictive and non-predictive cases. As we see that

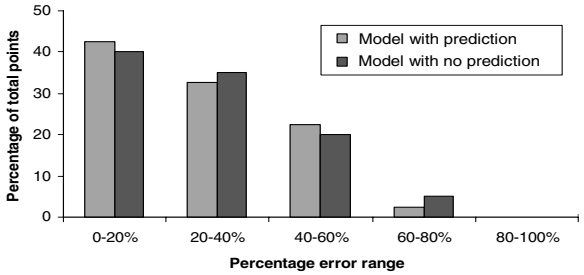

Fig. 6. Error distribution for our area model

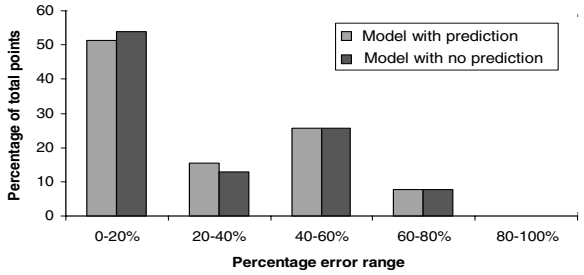

Fig. 7. Error distribution for our delay model

most of the design points (about 75%) are limited to 0-40% percentage error. There are only few design points in 40-80% error range. The error histogram is found to be very similar in both the cases with slight differences and the value of χ^2 distance between these two histograms is found to be 1.9.

Delay - We show the delay error histogram for both predictive and non-predictive model in Figure 7. Although both the histograms show similar characteristics, it is much different than the error histogram for area. Here, we find that more than 50% of total points are found to be with in 0-20% percentage error. The value of χ^2 in this case is found to be 0.635, which is very low.

This error analysis shows that our model is robust as it fits the model gracefully for both area and delay even when some of the points are taken away in the predictive analysis.

VI. Conclusion and Future Work

Unlike past SRAM models, accurately accounting for the impact of technology scaling across a wide range of design sizes and styles requires that we start with a model grounded with physical designs. We provide a set of methods, including an useful SRAM delay and area model and the means to fit it, that capture the design trends over a period of 15 years. Using a constrained optimization formulation over a data set of circuits papers, our model can automatically capture the most important scaling trends with underlying technology and size. These data can also be used to guide the design and recalibration of more detailed implementation-specific models. While our modeling framework is not restricted to a particular model, further complex models can also be built.

According to our finding, although Cacti provides better relative results, it overestimates the area by 20% in past CMOS technolgy to 40% in current CMOS technology. Using a recalibration method for Cacti and our model as a guide, the internal parameters of Cacti could be adjusted to bring them more in line with reality. Interestingly, we found that the delay of SRAM increases proportional to $\sqrt[4]{bits}$ with increasing number of SRAM $bits$, whereas in Cacti, delay increases proportional to $\sqrt[3]{bits}$ with increasing SRAM bits. We also demonstrated the forecasting ability of our model through a cross-validation technique to predict the SRAM area and delay for newer technology. While our methods are statistical and computationally intensive, the result is a simple analytical equation, which will serve as a useful tool for designers and architects for early-stage design exploration.

Acknowledgments

We would like to thank the anonymous reviewers for providing useful comments. We also thank Sanjeev Kumar from UCSD, John Brevik from UCSB and labmates for providing useful insights and comments on the initial draft. This work was funded in part by NSF Career Grant CCF-0448654.

References

[1] P. Shivakumar and N. P. Jouppi, "Cacti 3.0: An Integrated Cache Timing, Power and Area Model, Tech. Rep. Western Research Lab (WRL) Research Report, 2001/2.

[2] J. Mulder, N. Quach, and M. Flynn, "An area model for on-chip memories and its applications," *IEEE Journal of Solid-State Circuits*, vol. 26, no. 2, pp. 98–106, February 1991.

[3] T. Wada, S. Rajan, and S. A. Przybylski, "An analytical access time model for on-chip cache memories," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 8, pp. 1147–1156, August 1992.

[4] S. Wilton and N. Jouppi, "An enhanced access and cycle time model for on-chip caches, Tech. Rep. 93/5, DEC Western Research Lab, 1994.

[5] M. Mamidipaka and N. Dutt, "eCACTI: An Enhanced Power Model for On-chip Caches, Tech. Rep. CECS TR-04-28, September 2004.

[6] M. Mamidipaka, K. S. Khouri, N. D. Dutt, and M. S. Abadir, "IDAP: A Tool for High Level Power Estimation of Custom Array Structures," in *ICCAD*. IEEE Computer Society / ACM, 2003, pp. 113–119.

[7] S. L. Coumeri and D. E. Thomas, "Memory modeling for system synthesis," in *ISLPED '98: Proceedings of the 1998 international symposium on Low power electronics and design*. ACM Press, 1998, pp. 179–184.

[8] E. Schmidt, G. von Colln, L. Kruse, F. Theeuwen, and W. Nebel, "Memory power models for multilevel power estimation and optimization," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 10, no. 2, pp. 106–108, 2002.

[9] B. Amrutur and M. Horowitz, "Speed and power scaling of sram's," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 2, pp. 175–185, February 2000.

[10] S.-L. Lu and S. Hsu, "All you want to know about circuits as an architect but were afraid to ask," in *HPCA-11 Tutorial*, 2005.

[11] B. Agrawal and T. Sherwood, "Modeling tcam power for next generation network devices," in *Proc. of IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, Mar. 2006.

[12] http://www.virtual silicon.com/memory.cfm, "Memory compiler and instances," 2004.

[13] H. Akiyoshi et.al., "A 320ps access, 3ghz cycle, 144kb sram macro in 90nm cmos technology using an all-stage reset control signal generator," in *Digest of Technical Papers. ISSCC*, 2003, pp. 460–469.

[14] T. Seki et al., "A 6-ns 1-mb cmos sram with latched sense amplifier," *IEEE Journal of Solid-State Circuits*, vol. 28, no. 4, pp. 478–483, April 1993.

[15] L. S. Lasdon, A. D. Waren, A. Jain, and M. Ratner, "Design and testing of a generalized reduced gradient code for nonlinear programming," *ACM Trans. Math. Software*, vol. 4, no. 1, pp. 34–50, 1978.

[16] D. J. C. MacKay, "Comparison of approximate methods for handling hyperparameters," *Neural Computation*, vol. 11, no. 5, pp. 1035–1068, 1999.

[17] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *IJCAI*, 1995, pp. 1137–1145.

[18] H. Chernoff and E. L. Lehmann, "The use of maximum likelihood estimates in χ^2 tests for goodness-of-fit," *The Annals of Mathematical Statistics*, vol. 25, pp. 579–586, 1954.