Architectural Power Analysis: The Dual Bit Type Method

Paul E. Landman and Jan M. Rabaey

Abstract—**This paper describes a novel strategy for generating accurate black-box models of datapath power consumption at the architecture level. This is achieved by recognizing that power consumption in digital circuits is affected by activity, as well as physical capacitance. Since existing strategies characterize modules for purely random inputs, they fail to account for the effect of signal statistics on switching activity. The Dual Bit Type (DBT) model, however, accounts not only for the random activity of the least significant bits (LSB's), but also for the correlated activity of the most significant bits (MSB's), which contain two's-complement sign information. The resulting model is parameterizable in terms of complexity factors such as word length and can be applied to a wide variety of modules ranging from adders, shifters, and multipliers to register files and memories. Since the model operates at the register transfer level (RTL), it is orders of magnitude faster than gate- or circuit-level tools, but while other architecture-level techniques often err by 50–100% or more, the DBT method offers error rates on the order of 10–15%.**

Index Terms—**High-level design tools, library characterization, low power, power estimation, statistical modeling.**

I. Introduction

IN RECENT YEARS, power consumption has become a critical design concern for many VLSI systems. Nowhere is this more true than for portable applications, where power consumption has perhaps superseded speed and area as the overriding implementation constraint. Designers of high performance, nonportable systems are also beginning to run into power constraints. The recent introduction of a 50 W 300 MHz implementation of the DEC Alpha architecture [1], and the reliability and cost issues associated with packaging and cooling such a device is a clear indication of this fact.

Hence, for a variety of reasons, designers are increasingly led to consider power as a major system design criterion. This adds another degree of freedom—*and complexity*—to the design process and underscores the need for CAD tools that facilitate an efficient search of the full area-time-*power* (ATP) design space. Yet, while estimation and analysis tools exist for area and delay, similar tools for power are relatively scarce.

Furthermore, although the largest power reductions often stem from algorithmic and architectural modifications [2], [3], most contemporary power analysis tools operate at the gate or circuit levels. So, while offering good accuracy, they provide feedback too late in the design process to be particularly useful in either the optimization of a specific design or in the development of generic low-power heuristics. Moreover, the large number of gate- and circuit-level components makes rapid analysis of large systems difficult, if not impossible, at these levels.

In contrast, currently available architecture-level tools, while providing rapid results, sacrifice a good deal of accuracy - a weakness stemming from flaws in the underlying capacitance models. The Dual Bit Type (DBT) model overcomes these difficulties and is capable of producing simple and accurate black-box models for digital CMOS circuits through an automated module characterization process. As testimony to this, a DBT-based power/area analysis tool, dubbed SPA, has been developed and was used to generate many of the results presented in the following sections. Note that this paper focuses on the power, rather than the area, estimation capabilities of SPA.

The remainder of this paper consists of six sections. Section II will describe previous attempts at power analysis along with their limitations. Section III will then describe the DBT model and how it improves upon the state of the art. Section IV will describe how hardware libraries can be characterized to produce the data required during DBT power analysis. Section V will show how to use this data to analyze the power of digital systems given an RTL description of the architecture. Section VI will then present several case studies verifying the accuracy and flexibility of the DBT technique. Finally, Section VII will present conclusions along with directions for future work.

II. Previous Work

To date, nearly all research into power analysis and estimation has dealt with the circuit and gate levels of abstraction. At the lowest level, power analysis is typically achieved through circuit simulation, using transistor-, or perhaps, switch-level tools [4]–[7] such as SPICE [8], [9] or irsim [10]. More recently, a number of probabilistic gate-level tools [11]–[14] have emerged, including those of Ghosh [15], Najm [16], and Tsui [17]. For these tools, the boolean gate, rather than the transistor, is the primitive unit, and all capacitances are typically lumped at its output. The power of each gate is then calculated as $P(0 \rightarrow 1)C_L V^2 f$, where $P(0 \rightarrow 1)$ is the zero-to one transition probability of the output signal. Unfortunately, since these tools require a gate-level mapping as input, they

Manuscript received June 15, 1994; revised February 10, 1995. This work was supported by a fellowship from the National Science Foundation and by ARPA under Grant J-FBI 93-153.

P. E. Landman is with the Integrated Systems Laboratory, Texas Instruments, Inc., Dallas, TX 75243 USA.

J. M. Rabaey is with the EECS Department, University of California, Berkeley, Berkeley, CA 94720 USA.

IEEE Log Number 9411828.

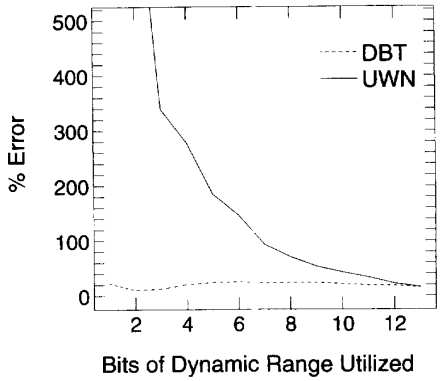

Fig. 1. Error in modeling multiplier power.

are useful more as a back-end verification step, rather than as an aid in architectural exploration.

Architectural power models do exist [18]–[20]; however, they offer significantly less accuracy than their circuit- and gate-level counterparts. For instance, a power model for an array multiplier might be represented as $P = C_{\mathrm{U}} N^2 V^2 f$, where N is the word length of the two inputs. Here, the quadratic dependence on N accounts for the N^2 full adder cells of which a typical array multiplier is composed. Given this model, the characterization process consists of simulating or measuring the power dissipation of a multiplier, and fitting the capacitive coefficient, C_{U}, to these results. This capacitive coefficient can be interpreted as the effective switching capacitance per one bit cell of the multiplier. As will be seen shortly, however, a module's power consumption depends on the inputs applied. It being impossible to characterize the module for all possible input statistics, purely random inputs—that is to say, independent uniform white-noise (UWN) inputs—are typically applied when deriving C_{U} .

This leads to the chief source of error in architectural power modeling as illustrated by Fig. 1, which displays the estimation error (relative to switch-level simulations) for a 16 × 16 multiplier. Clearly, when the dynamic range of the inputs doesn't fully occupy the word length of the multiplier, the UWN model becomes extremely inaccurate. Indeed, for a dynamic range of one utilized bit, the UWN error reaches 6543% (not shown in the figure for lack of space). Granted, with good design, word length utilization this poor would typically be avoided; however, errors in the range of 50–100% are not uncommon [21]. Furthermore, regardless of the exact magnitude of the error, the figure clearly indicates an inadequacy in the UWN model. The DBT model addresses this deficiency and, as a preview of later results, its modeling error for the multiplier has also been included in Fig. 1.

III. The DBT Power Analysis Model

This section details the DBT model, which combines the reduced complexity of the architecture level with the accuracy of the gate and circuit levels. This is done by producing a black-box model of the capacitance switched in each module for various types of inputs. If desired, these capacitance estimates can be converted to an equivalent energy, $E = CV^2$, or power, $P = CV^2 f$.

The DBT capacitance models are easily parameterized. For example, the user can specify precisely how the physical capacitance of each module should scale with its "size" or complexity. This allows the model to reflect the fact that a 16-b adder will contain roughly twice the physical capacitance of an 8-b adder.

Unlike previous attempts at architectural power analysis, the model accurately accounts for activity, as well as physical capacitance. Instead of having a single capacitive coefficient based on a uniform white noise assumption, the model employs several coefficients, each corresponding to a different input type. As a result, the effect of the input statistics on the module power consumption is reflected in the estimates. Since the technique accounts for two input bit types rather than one, we refer to it as the DBT model.

The following sections describe the DBT model in more detail. Section III-A tells how the models can be parameterized to scale with module complexity. Section III-B then describes how the model accurately accounts for variations in signal activity.

A. Modeling Complexity

Intuitively, the total power consumed by a module should be a function of its complexity (i.e., "size"). This reflects the fact that larger modules contain more circuitry and, therefore, more physical capacitance. For instance, one would expect a 16 × 16 multiplication to consume more power than an 16 × 16 multiplication. Under the DBT model, the user can specify arbitrary capacitance models for each module in the datapath library.

For example, consider modeling the capacitance of a ripple-carry subtracter. The physical capacitance an instance of this subtracter will contain is determined by its word length, N. In particular, an N-b subtracter will be realized by N one-bit full-subtracter cells. The total module capacitance should, therefore, be proportional to the word length as shown here:

$$C_T = C_{eff} N. \tag{1}$$

In this equation, C_{eff} is a capacitive coefficient, which describes the effective capacitance switched for each bit of the subtracter. A detailed discussion of the capacitive coefficients will be postponed until the section on modeling activity.

Many modules besides the subtracter also follow a simple linear model. For example, ripple-carry adders, comparators, barrel shifters, multiplexers, and registers all obey (1). The DBT method is not, however, restricted to linear capacitance models.

The flexibility of the DBT modeling strategy allows the user to specify even more complex capacitance models when necessary. Take the case of the logarithmic shifter depicted in Fig. 2. This module consists of several stages, each capable of performing successive shifts by powers of two, conditional on the binary value of the SHIFT input. If the unit is capable of performing a maximum shift by M bits, then the shifter will require $L = \lceil \log_2 (M + 1) \rceil$ stages. The capacitance model

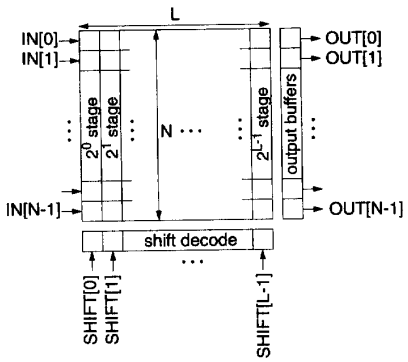

Fig. 2. Decomposition of a logarithmic shifter.

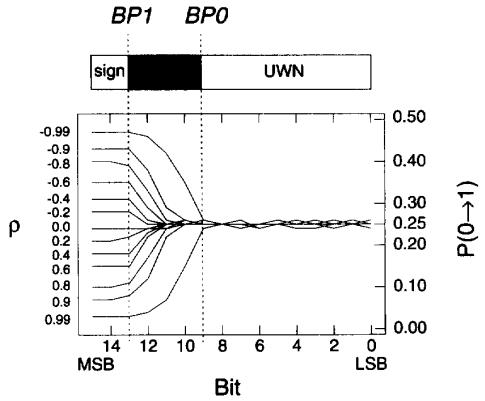

Fig. 3. Transition activity versus bit for Gaussian data streams with varying temporal correlation, ρ.

for the log shifter is a function of three parameters: the word length, N, the maximum shift value, M, and the number of shift stages, L. The exact capacitance model is given by the following equation:

$$C_T = C_0N + C_1L + C_2NL + C_3N^2L + C_4MNL. \quad (2)$$

Referring to the figure, the C_0N term reflects the capacitance of the N output buffers. Likewise, the C_1L term represents the overhead capacitance of the decoders and drivers for the L shift control signals. Since there are NL cells in the shift array, the capacitance model also contains a C_2NL term. The last two terms correspond to the amount of routing in the array. Each of these final terms is proportional to the number of cells in the array, NL. Since the amount of routing crossing each cell can range between N and M depending on the position of the cell, the expression includes a term for each case: C_3N^2L and C_4MNL.

Since this capacitance model has several terms, it also requires several capacitive coefficients. This does not pose a problem for the DBT model. The solution is to use vector rather than scalar arithmetic. In other words, instead of a scalar capacitive coefficient, C_{eff}, we use a capacitive coefficient vector

$$\mathbf{C}_{eff} = [C_0 \quad C_1 \quad C_2 \quad C_3 \quad C_4]^T. \quad (3)$$

Similarly, rather than a single scalar complexity parameter, N, we use a parameter vector:

$$\mathbf{N} = [N \quad L \quad NL \quad N^2L \quad MNL]^T. \quad (4)$$

The result is the following vector capacitance model:

$$C_T = \mathbf{C}_{eff} \cdot \mathbf{N}. \quad (5)$$

In summary, the capacitance models used by the DBT method are extremely flexible. The user or library developer can specify precisely how various complexity parameters affect the total capacitance of each module. Typical models range from simple linear expressions involving a single parameter and capacitive coefficient to complex functions of multiterm parameter vectors.

B. Modeling Activity

The previous section described how to model the effect of module "size" on capacitance through a complexity parameter, **N**. In that discussion, **N** was weighted by an effective capacitance coefficient, $\mathbf{C}_{eff}$. This section describes how $\mathbf{C}_{eff}$ can be used to model activity, as well as physical capacitance. First, Section III-B1) describes the motivation behind the DBT activity model. Then Section III-B2) and B3) discuss the capacitive coefficients required to estimate the capacitance switched during data and control input transitions, respectively.

1) The DBT Data Model: Traditional architectural modeling techniques use a single coefficient for each module that is derived for uniform white noise inputs. Therefore, the activity due to anything but random data transitions is poorly modeled. The solution is to use several effective capacitance coefficients–one for each type of data being modeled. This section explores what additional data types (aside from uniform white noise) should be modeled.

Fig. 3 shows the bit transition activity for several different data streams. In particular, each curve corresponds to a Gaussian process, X_t, with a different temporal correlation,

$$\rho = \frac{cov(X_{t-1}, X_t)}{\sigma^2}. \quad (6)$$

The uniform white noise model assumes that all bits have a zero to one transition probability of 0.25. This value can be derived by recognizing that for uniform white noise, zero and one bits are equally likely, and all bits are spatially, as well as temporally independent. Mathematically,

$$\text{UWN bit activity: } P(0 \rightarrow 1) = P(0)P(1) = 0.5^2 = 0.25. \quad (7)$$

Comparing this to the figure, the UWN model appears to work well for the least significant bits (LSB's) up to a low-end breakpoint $BP0$. Intuitively, this is not surprising. The LSB's tend to cycle rapidly with small changes in word value. This cycling has a randomizing effect on the bits which, as a result, begin to look like UWN bits. The uniformity of least

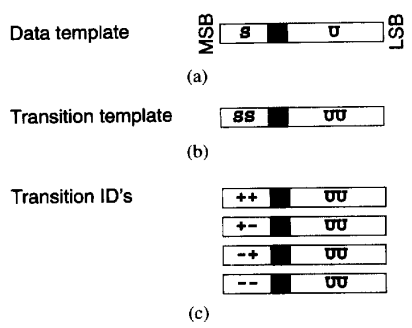

Fig. 4. Templates for identifying data bit types.

significant digits is a well-known statistical phenomenon, and a detailed discussion of the topic is given by Preece in [22].

In contrast, the high-order bits from the MSB down to breakpoint $BP1$ are sign bits and, in general, their behavior differs markedly from the data bits. Among the sign bits, $P(0 \rightarrow 1)$ represents the probability that the data will transition from a positive to a negative value. The figure shows that positively correlated signals ($\rho > 0$) experience lower activity in the sign region, while negative correlation ($\rho < 0$) leads to increased sign activity.

As a minor point, the UWN and sign bit regions do not quite cover the entire range of bits. A small intermediate region separates the two constant activity regions. It is not necessary to create a new bit type to model this region. Instead, linear interpolation of the sign and UWN activities models the intermediate region quite well.

The position of the model breakpoints depends on how many bits are required to represent the numbers in the data stream; the unused bits will be devoted to sign. A detailed discussion of how to derive the breakpoint positions is reserved for Appendix A. For now, suffice it to say that analytical expressions can be found that express the breakpoints as a function of word-level statistics such as mean (μ), variance (σ^2), and correlation (ρ):

$$BP1 = \log_2 (|\mu| + 3\sigma) \tag{8}$$

$$BP0 = \log_2 \sigma + \Delta BP0 \tag{9}$$

$$\Delta BP0 = \log_2 [\sqrt{1 - \rho^2} + |\rho|/8]. \tag{10}$$

Since the sign bit activity is so different from the LSB, or UWN, bit activity, the capacitance model should account for both types of bits. In other words, rather than having a single capacitive coefficient for UWN bits, there should also be capacitive coefficients for different sign bit transitions. The first step in determining these coefficients is to specify what types of bit transitions need to be characterized.

Fig. 3 demonstrated that the activity of any two's-complement data stream can be characterized by two types of bits: UWN and sign. This implies that a *data template* such as that shown in Fig. 4(a) can be associated with any data stream. The data template classifies bits in the data stream as either **U** or **S** for UWN and sign bits, respectively. The qualifier "template" refers to the fact that no specific value for the sign bits (i.e., + or −) is indicated.

Since the ultimate intent is to classify data *transitions*—which cause capacitive switching—rather than static values, the concept of a *transition template* illustrated in Fig. 4(b) will also prove useful. The **SS** indicates a transition from one sign value to another (possibly equal) value. Similarly, **UU** suggests that the LSB's all transition from their current random values to some new random values. As before, a transition "template" does not imply any particular sign values; however, a *transition ID* can be used to denote a specific sign transition. For instance, the transition template of Fig. 4(b) encompasses the four possible transition ID's of Fig. 4(c).

To summarize, the activity of datapath signals cannot be adequately described by using a pure UWN model. This section has proposed a DBT model that reflects the distinct behaviors of both the LSB's and the MSB's. The LSB's are still modeled as UWN bits, however, the MSB's are recognized as sign bits. Terminology was presented for identifying these bit types, as well as the possible transitions they can make. The next section will describe how the capacitance switched within a module for these various data transitions can be captured by using several effective capacitance coefficients, rather than just one.

2) Capacitive Data Coefficients: As mentioned above, the intent of the capacitive coefficients is to describe the average amount of capacitance switched within a module during an input transition. UWN techniques model only one type of data statistics and, therefore, require a single coefficient to characterize the effective capacitance of each module. In contrast, the DBT method accurately accounts for different types of input statistics by having a separate coefficient for each possible transition of both the UWN and the sign bits.

The terminology of the previous section, which was used to distinguish possible input transitions can also be used to categorize the required capacitive coefficients. For example, consider a bit-sliced module with a single data input as shown in Fig. 5. The module can be split into two regions based on the input bit types. The LSB region experiences random (UWN) input transitions and the effective capacitance of this region can, therefore, be characterized by a single capacitive coefficient, $C_{\mathbf{UU}}$. The MSB region, in contrast, experiences sign transitions and will require capacitive sign coefficients, $C_{\mathbf{SS}}$, to reflect its effective capacitance. Since there are four sign transitions that fit this template, there must also be four distinct coefficients. In summary, the effective capacitance of a one-input module is completely characterized by a table of five coefficients (see Table I).

The number of coefficients that are required will depend on how many inputs the module has. The one-input module discussed above requires five coefficients since only five types of input bit transitions are possible. A two-input module, on the other hand, must be characterized for transitions on more than one data stream. This situation is illustrated by Fig. 6. For this case, the LSB region sees random (UWN) transitions on both inputs. The transition ID for this region of the module is written **UU/UU**, where the "/" separates the transition ID for IN_1 from IN_2. The effective capacitance of the module in this region will be described by the coefficient $C_{\mathbf{UU/UU}}$.

In the sign region, the module transition template has three components (**SS/SS/SS**) rather than the two that might be

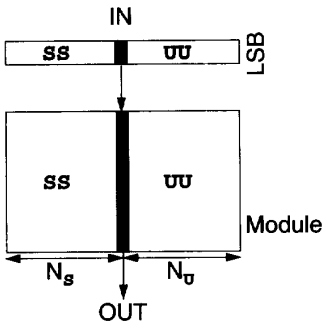

Fig. 5. Transition templates for one-input modules.

TABLE I
CAPACITIVE COEFFICIENTS FOR ONE-INPUT MODULES

Transition Templates	Capacitive Coefficients			
UU	C_{UU}			
SS	C_{++}	C_{+-}	C_{-+}	C_{--}

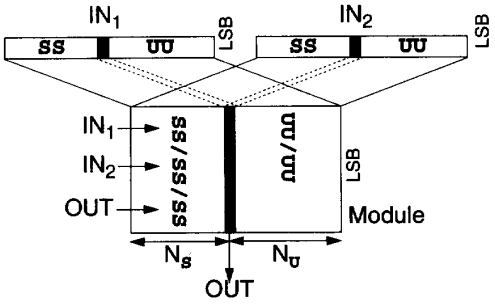

Fig. 6. Transition templates for two-input modules (with aligned breakpoints).

expected. This is because the output sign can affect the capacitance switched in the module, and for some modules the sign of the inputs does not completely determine the output sign. For example, consider characterizing the effective capacitance of a subtracter. Subtracting two positive numbers could produce either a positive or a negative result. If the transition template for the module only included inputs, then a particular transition might be classified as $++/++$. This makes it appear as though the module undergoes no activity even though it is possible that the output does make a sign transition (e.g., positive to negative). Specifying the output sign (e.g., $++/++/+-$) avoids this ambiguity. For this reason, the transition template for the sign region of a two-input module should contain an entry for the output, as well as the inputs. Expanding the module transition templates yields 65 different transition ID's—each of which must have a characteristic capacitive coefficient (see the **UU/UU** and **SS/SS/SS** entries of Table II).

In Fig. 6 the situation was simplified by assuming that the breakpoints of the two inputs aligned perfectly. In reality, the inputs might have different statistics that cause the number

TABLE II
CAPACITIVE COEFFICIENTS FOR TWO-INPUT MODULES

	Transition Templates	Capacitive Coefficients			
	UU/UU	$C_{UU/UU}$			
	SS/SS/SS	$C_{++/++/++}$	$C_{++/++/+-}$	$C_{++/++/-+}$	$C_{++/++/--}$
		$C_{++/+-/++}$	$C_{++/+-/+-}$	$C_{++/+-/-+}$	$C_{++/+-/--}$
		...	...	...	...
		$C_{--/-+/++}$	$C_{--/-+/+-}$	$C_{--/-+/-+}$	$C_{--/-+/--}$
		$C_{--/--/++}$	$C_{--/--/+-}$	$C_{--/--/-+}$	$C_{--/--/--}$
Misaligned breakpoints only	UU/SS	$C_{UU/++}$	$C_{UU/+-}$	$C_{UU/-+}$	$C_{UU/--}$
	SS/UU	$C_{++/UU}$	$C_{+-/UU}$	$C_{-+/UU}$	$C_{--/UU}$

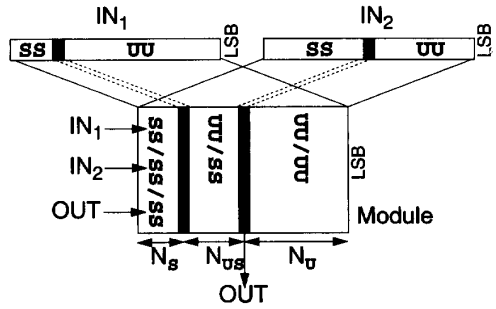

Fig. 7. Transition templates for two-input modules (with misaligned breakpoints).

of UWN bits and sign bits required by each of the inputs to be different. The general case of misaligned breakpoints is shown in Fig. 7. Relaxing breakpoint alignment creates the possibility of a third module region (**UU/SS**) with one UWN input and one sign input. This increases the number of capacitive coefficients and transition templates required to fully characterize the module to 73 as shown in the bottom portion of Table II.

Recall from the previous section that capacitive coefficients can be either scalars or vectors depending on the number of complexity terms in the capacitance model. For models with more than one term, the capacitive coefficients described here would also be vectors rather than scalars. For instance, the logarithmic shifter described above would require five coefficient vectors (corresponding to the entries in Table I) to fully characterize the effective capacitance. Each of these vectors in turn would consist of five scalar elements. For example, $\mathbf{C}_{UU} = [C_{0,UU}\ C_{1,UU}\ C_{2,UU}\ C_{3,UU}\ C_{4,UU}]^T$ for the uniform white noise entry of Table I.

The number of capacitive coefficients required to characterize a module grows rapidly with the number of inputs. Consequently, approximations must be used for datapath blocks with more than two inputs (e.g., multiplexers). Fortunately, most common datapath elements have two inputs or less (e.g., multipliers, shifters, adders, subtracters, comparators, etc.).

In summary, there are two types of bits (sign and UWN) that both have quite different activities. As a result, the effective capacitance of a module will be different depending on the exact input transitions that the module experiences. To accurately account for this effect, the DBT model uses distinct

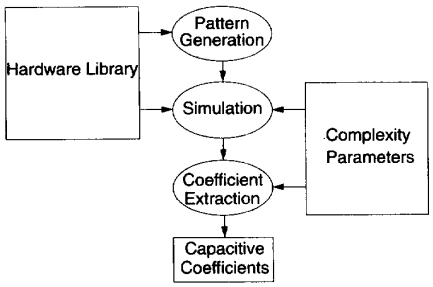

Fig. 8. Process flow for library characterization.

capacitive coefficients for each type of data transition that can occur.

3) Multifunction Units and Capacitive Control Coefficients: If a unit performs only a single function (e.g., an adder), then its power can be described by a single set of capacitive coefficients. Some units, however, are able to perform multiple functions, which may each consume different amounts of power—e.g., an ALU which can perform add, subtract, and shift operations. In this case, a single set of coefficients is insufficient. Instead, we must have a unique set of capacitive coefficients for each distinct function that the unit may be called on to perform. This minor extension allows the DBT model to handle input data transitions for multifunction modules in much the same manner as has been described above.

A multifunction module, however, will have control as well as data inputs. The control inputs together form an *instruction* to the module which tell it what function it is to perform. For example, an ALU may have several encoded control bits whose values determine whether the module will implement an add, subtract, or shift operation. As long as these control lines remain fixed, the module will execute the same function and data transitions can be handled in the manner already described using the set of capacitive coefficients corresponding to the current instruction. When a control input makes a transition, however, the state of the module can be affected just as if a data transition had occurred. This change in state is most often accompanied by some energy consumption as the internal nodes of the module transition to their new values. Therefore, in order to analyze the total power consumption of a module we must consider capacitance switching initiated by control transitions, as well as data transitions.

This capacitance can be modeled by adding capacitive *control* coefficients that describe the average amount of capacitance switched in the module when the control inputs transition to a new instruction. Since a *transition* involves an initial and final value, it may seem that a module with N functions would require N^2 coefficients to characterize all possible instruction transitions. Technically, this is true, however, from a practical standpoint, it is primarily the final instruction that determines the power consumption since the module outputs are being recomputed for this new instruction, not the previous one. So, a complete set of capacitive coefficients for a multifunction module would consist of a capacitive control coefficient for each instruction, as well as a table of capacitive data coefficients for each instruction.

To summarize, the capacitance switched under various types of input activity can be more accurately modeled by using a table of effective capacitance coefficients rather than the traditional single (UWN) coefficient. Separate coefficients can be used to characterize transitions on data inputs, as well as control inputs (for multifunction modules). These coefficients can be plugged into the capacitance models described in Section III-A in order to accurately model the effect of *complexity* and *activity* on module power consumption.

IV. Library Characterization Method

The preceding discussion assumes the existence of capacitive coefficient tables, but does not describe how to produce them. *Library characterization* refers to the process of generating effective capacitance coefficients for each module in the datapath library. This is a one-time process, not performed during power analysis, but instead performed whenever a new cell is added to the library.

The procedure consists of several steps (see Fig. 8) but can be automated so as to require very little intervention on the part of the user. *Pattern generation* is the first step in the three-stage process. During this phase, input patterns for various UWN, sign, and control transitions are generated for the module being characterized. Next, *simulation* is used to measure the capacitance switched for these input activity patterns. In order to characterize the influence of complexity, as well as activity, the module may be characterized for several complexity parameter values (e.g., word length, number of shift stages, etc.). Finally, during *coefficient extraction,* the capacitance models are fit to the simulated capacitance data to produce a set of "best fit" capacitive coefficients.

A. Pattern Generation

The input patterns applied during characterization must embody all important types of input transitions. This includes both data (UWN/sign) and control (instruction) input transitions. The pattern generation problem can be simplified by handling data and control separately.

Data pattern generation is performed assuming fixed control inputs. In order to fully characterize a module, the data stream must include patterns for each transition ID. So for a one-input module, the **UU** transition ID requires a random data stream. Similarly, the **SS** transition template requires inputs corresponding to the following sign transitions: positive to positive (++), positive to negative (+−), negative to positive (−+), and negative to negative (−−). Fig. 9 shows what a data stream used to characterize a module (with a word length of 16 b) might look like. Notice that input patterns containing a UWN component must be simulated for several cycles to allow the average capacitance switched to converge during simulation. Also recognize that the data stream must be repeated for all possible instructions of multifunction units.

Similar input patterns can be generated for two-input modules. This can be done in a manner that minimizes the overall simulation length by, whenever possible, using the previous

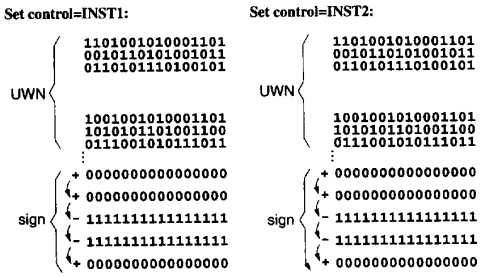

Fig. 9. Sample data input patterns for a 16-b one-input module.

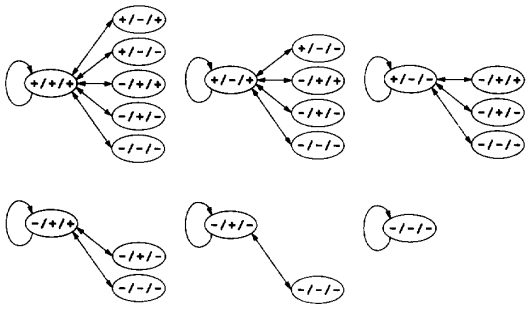

Fig. 10. Sign characterization sequence for an adder.

data values as the initial data types for the next transition ID. For example, if the current data values have signs given by $+/+/+$, then the next transition ID characterized should be of the form $+\square/+\square/+\square$ to avoid having to insert an initialization step during simulation. The full characterization sequence for the **SS/SS/SS** transition template of an adder is given in Fig. 10.

Notice that as mentioned previously, the transition ID's for two-input modules include an output sign specification. In order to control the output sign, the "sign" inputs cannot always be limited to all **0**'s (for positive) or all **1**'s (for negative) as was done for the one-input module. For example, a multiplier must be characterized for $(+)\times(-)=(-)$, but using all **0** or all **1** inputs would produce:

$$(\mathbf{000}\cdots\mathbf{00})\times(\mathbf{111}\cdots\mathbf{11})=(\mathbf{000}\cdots\mathbf{00}). \tag{11}$$

In other words, $(0)\times(-1)=(0)$. The solution is not to require sign inputs to have all **0**'s or all **1**'s, but instead to allow a small number of nonsign bits that can be used to manipulate the output sign. For instance, the correct multiplier signs can be generated by:

$$(\mathbf{000}\cdots\mathbf{01})\times(\mathbf{111}\cdots\mathbf{11})=(\mathbf{111}\cdots\mathbf{11}). \tag{12}$$

or $(1)\times(-1)=(-1)$.

For multifunction units, the input patterns must include control, as well as data transitions. In particular, input patterns exercising the various possible instruction transitions must be generated. Recall, however, that characterization of instruction transition energies depends primarily on the final instruction value. This simplifies the pattern generation process since it must only generate a transition *to* each instruction regardless of the initial instruction. This means that all control coefficients

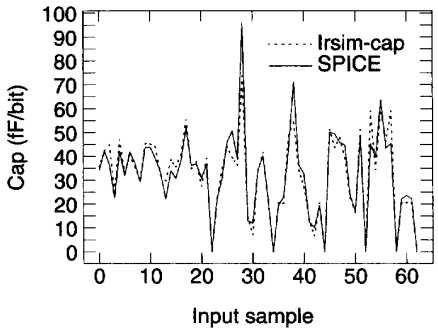

Fig. 11. Irsim-cap versus SPICE.

can be characterized by a single pass through the instruction set.

To summarize, the input patterns should exercise all important combinations of sign, UWN, and instruction transitions. This will allow all required effective capacitance coefficients to be represented during simulation. The algorithm for generating minimal length input pattern sequences is relatively straightforward and an automatic pattern generation tool (APG) that performs this function has been implemented.

B. Simulation

Once generated, the input patterns are fed to a simulator from which module switching capacitances are extracted. Any simulator may be chosen, depending on the accuracy desired and the time allotted to characterization. Switch-level simulators offer, perhaps, the best combination of speed and accuracy. All results presented here were derived using irsim-cap, which is a version of irsim-9.0 [10] with improved capacitance measurement capabilities. Irsim-cap is three to four orders of magnitude faster than SPICE with its average power estimates usually differing by only 10–15% (after proper calibration). Fig. 11 provides a graphical comparison of SPICE and irsim-cap for a subtracter operating on a stream of 64 inputs.

Ideally, the simulation would be based on a net list extracted from an actual implementation of the function being characterized. In this way, design parameters such as device sizes, as well as technology parameters including gate capacitances and device transconductances are automatically incorporated into the resulting model.

Unfortunately, transistor-level net lists of the hardware module are not always readily available. In these cases, logic synthesis tools can be used to produce a gate-level mapping of the function. The UWN and sign-bit input patterns, discussed above, can then be applied to this boolean network using one of the many existing gate-level simulators with capacitance measurement capabilities [11]–[17].

Characterization does not have to be based on simulation. If physical implementations of library modules are available, characterization can be performed by applying the input patterns to the actual chip while *measuring* incremental power consumption. This would, of course, be the most accurate technique, however, since library cells are not commonly

fabricated individually on chips, simulation must in most cases suffice.

In any case, it is interesting to note that since each module is characterized as a multibit unit (e.g., a 16-b or 32-b adder) glitching within the module can actually be accounted for during characterization. This is an improvement over many gate-level power estimation tools that employ a zero-delay model and, therefore, ignore glitching. Of course, the DBT model will only account for glitching if the simulator used during characterization models this phenomenon.

C. Coefficient Extraction

The simulation step produces an output file containing effective switching capacitances for the entire series of applied input transitions. It only remains to extract the capacitive coefficients from this mass of raw data. This can be achieved using a variety of model fitting techniques such as linear least-squares regression, which minimizes the mean-squared error of the model. This amounts to solving the following matrix equation for the capacitive coefficient vector, $\mathbf{C}_{eff}$, that minimizes the error, $\mathbf{e}$:

$$\mathbf{C}_{sim} = \mathbf{P}\mathbf{C}_{eff} + \mathbf{e} \tag{13}$$

where $\mathbf{C}_{sim}$ is a vector of simulated capacitance observations and $\mathbf{P}$ is a matrix of complexity parameter values corresponding to each observation. Since for each module, there are several capacitive coefficients (e.g., $\mathbf{C}_{UU}$, $\mathbf{C}_{++}$, $\mathbf{C}_{+-}$, etc.) the equation must be solved several times—once for each capacitive coefficient vector.

The first step in solving these equations is to form the parameter matrix, $\mathbf{P}$. Each row of the matrix corresponds to a different value of the complexity vector, $\mathbf{N}$, described in Section III-A. For example, assume the logarithmic shifter described in that section was characterized for word lengths $N \in \{8, 16, 32\}$ and number of stages $L \in \{1, 3, 5\}$, implying $M = \{1, 7, 31\}$. After plugging these values into the complexity vector, $\mathbf{N} = [N \quad L \quad NL \quad N^2L \quad MNL]$, the parameter matrix would be given by:

$$\mathbf{P} = \begin{array}{c} \begin{matrix} N & L & NL & N^2L & MNL \end{matrix} \\ \begin{bmatrix} 8 & 1 & 8 & 64 & 8 \\ 8 & 3 & 24 & 192 & 168 \\ 8 & 5 & 40 & 320 & 1240 \\ 16 & 1 & 16 & 256 & 16 \\ 16 & 3 & 48 & 768 & 336 \\ 16 & 5 & 80 & 1280 & 2480 \\ 32 & 1 & 32 & 1024 & 32 \\ 32 & 3 & 96 & 3072 & 672 \\ 32 & 5 & 160 & 5120 & 4960 \end{bmatrix} \end{array}. \tag{14}$$

This matrix and the vector $\mathbf{C}_{sim}$ of corresponding capacitance observations would then feed directly into a least-squares regression algorithm to produce the best-fit coefficient vector, $\mathbf{C}_{eff}$.

In summary, characterization is the process used to generate tables of capacitive coefficients for library modules, which can be used to evaluate effective capacitances during power analysis. Characterization consists of three phases: pattern generation, simulation, and coefficient extraction. As with pattern generation and simulation, the coefficient extraction process has been automated and is performed by a tool called automatic capacitance extractor (ACE).

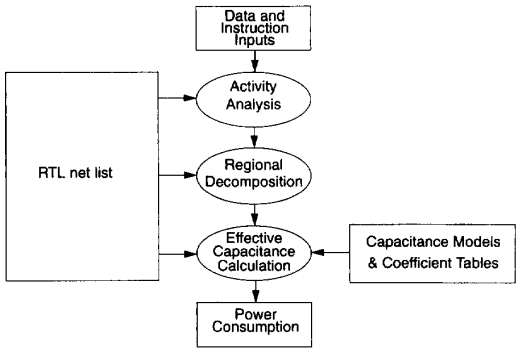

Fig. 12. Process flow for DBT power analysis.

The accuracy obtained by characterization depends on the suitability of the chosen capacitance model, as well as the number of observations used in fitting the model; however, as an approximate figure, the logarithmic shifter described here can be modeled with an rms error of only 9.9%. Of course, other sources of error, aside from model characterization, such as inaccuracies in the capacitance simulations, contribute to the overall error of the technique, which is more on the order of 10–15% (relative to switch-level simulations).

V. Power Analysis Method

The capacitive coefficient tables produced by library characterization can be used to analyze the power consumed by datapath modules. This section describes the specifics of the DBT power analysis method. Fig. 12 shows the sequence of steps in the power analysis process. First, the user must provide a register-transfer level description of the chip architecture to be analyzed. This gives the power analysis tool a list of all the different modules that must be analyzed and their complexities. With this information, the tool can access the appropriate capacitance models and coefficient tables in the hardware database. The user must also supply a set of data and instruction input vectors for which the power consumption will be evaluated. These will be used along with the RTL description to derive the necessary activity statistics.

The following sections will describe the three main phases of power analysis in more detail. First, Section V-A will describe how the activity parameters are derived based on the RTL net list and the supplied input vectors. Then, Section V-B will go on to show how the modules are decomposed into UWN and sign bit regions based on these word-level statistics. Finally, Section V-C will describe how the effective capacitance switched within each module is calculated using a region-by-region analysis.

A. Activity Analysis

The activity parameters for the DBT model are the word-level statistics that help to determine the relative activities of

the UWN and sign regions of datapath modules and buses. In particular, these statistics include: mean (μ), variance (σ^2), correlation (ρ), and the sign transition probabilities. The first three affect the placement of breakpoints during regional decomposition, while the sign (and instruction) transition probabilities are required to weight the appropriate capacitive data and control coefficients during the effective capacitance computation.

We can consider several possible techniques for deriving these statistics including statistics propagation, transfer function evaluation, and functional simulation. In most cases, functional simulation provides the most general and useful strategy for deriving DBT activity parameters. Given a set of typical input vectors (data and instruction) for the chip, the design is simulated at the functional, or RT, level. During this simulation, the required statistics, including sign transition probabilities, are accumulated directly. The simulation must be carried out over a long enough sequence of data samples to allow the statistics to converge to their true values. This allows reconvergent fan-out and feedback to exert their influence on the activity of the design. The exact number of simulation cycles required depends on the particular design being analyzed; however, it is possible to monitor the activity statistics for convergence and then to terminate the simulation once the desired confidence interval is reached. Most designs require much less than 100 cycles to adequately converge [23]. Since RTL simulation is very fast, simulation time is not an important concern and is typically on the order of seconds or minutes.

The output of the simulation process is a set of activity parameters that can be used in the DBT power analysis calculations. These activities, of course, correspond to the data and instruction input vectors that were supplied by the user. In this way, the designer can characterize how much power an architecture consumes for various instruction streams. Such a characterization might prove useful at the algorithm level in designing compilers that generate "low-power" programs.

B. Regional Decomposition

Following activity analysis, the modules in the chip must be decomposed into sign and UWN regions. This allows the power analysis tool to account for differences in the behavior and activity levels of these regions. The decomposition strategy utilizes the breakpoint formulas, i.e., (8)–(10), presented in Section III-B1 and derived in Appendix A. The activity statistics derived for each module during functional simulation are used in these formulas to compute the appropriate breakpoints.

Consider the case of a one-input module as depicted in Fig. 5. The task of regional decomposition reduces to figuring out the number of sign bits, $N_{\mathbf{S}}$, and the number of UWN bits, $N_{\mathbf{U}}$. Linear interpolation of power consumption for the small intermediate region can be achieved by attributing half of the $N_{\mathbf{I}}$ intermediate bits to the sign region and half to the UWN region:

$$N_{\mathbf{I}} = BP1 - BP0 - 1 \tag{15}$$

$$N_{\mathbf{S}} = (N - BP1) + \frac{N_{\mathbf{I}}}{2} \tag{16}$$

$$N_{\mathbf{U}} = (BP0 + 1) + \frac{N_{\mathbf{I}}}{2}. \tag{17}$$

Similar decomposition formulas can be derived for the two-input case of Figs. 6 and 7.

C. Effective Capacitance Calculation

The output of the decomposition process is a list of regions contained in each module along with their relative sizes. Once the modules have been decomposed, the effective capacitance switched in each region of each module can be analyzed independently by applying the appropriate capacitance model and coefficients. The general formula for calculating the capacitance of a region is given by:

$$C_r = \frac{N_r}{N} \sum_{i \in \begin{pmatrix} \text{transition IDs} \\ \text{for region } r \end{pmatrix}} P(i)\mathbf{C}_i \cdot \mathbf{N} \tag{18}$$

where i steps through the transition ID's of region r, $\mathbf{C}_i$ is the capacitive coefficient vector for transition ID i, $\mathbf{N}$ is the complexity vector for the module, $P(i)$ is the probability that transition i occurs, and N_r is the number of bits in region r. In this expression, $\mathbf{C}_i \cdot \mathbf{N}$ taken alone would be the effective capacitance if the entire module were experiencing transition i. This must be weighted by the fraction of bits that actually are in region r, N_r/N, as well as the probability that transition i occurs, $P(i)$.

This general formula can be applied to the cases of one- and two-input modules. For example, when applied to the case of one-input bit-sliced modules, the formula leads to the more specific expression:

$$C_T = \frac{N_{\mathbf{U}}}{N}[\mathbf{C_{UU}} \cdot \mathbf{N}] + \frac{N_{\mathbf{S}}}{N}\left[\sum_{\mathbf{ss} \in \begin{pmatrix} ++,+-, \\ -+,-- \end{pmatrix}} P(\mathbf{SS})\mathbf{C_{SS}} \cdot \mathbf{N}\right]. \tag{19}$$

A similar expansion can be performed for the case of a two-input module.

To summarize, analysis of chip power consumption using the DBT model occurs in three stages. In the first step, functional simulation of an RTL net list is used to derive activity statistics for a set of user-supplied data and instruction input vectors. Using these statistics, the modules in the design are decomposed into UWN and sign regions of various sizes. Finally, the effective capacitance of each region in each module is computed using the appropriate capacitance models and capacitive coefficients, which are stored in a hardware database.

As with all models, the applicability of DBT-based power analysis is subject to certain limitations. For instance, the model is currently based on two's-complement data representations. It could, however, be extended fairly easily to account for other representations such as sign-magnitude or even unsigned. Furthermore, the DBT model applies primarily

TABLE III
CAPACITIVE COEFFICIENTS FOR A SUBTRACTER

Transition Templates	Capacitive Coefficients (fF/bit)															
uu/uu	264															
uu/ss	203	351	342	115												
ss/uu	118	316	315	199												
ss/ss	0	190	273	21	302	0	124	0	363	171	0	0	4	0	0	0
	0	236	0	347	107	254	242	107	0	375	0	0	348	184	0	0
	0	0	461	405	0	0	305	0	51	218	318	71	307	0	129	0
	0	0	0	16	0	0	366	486	0	230	0	319	5	169	152	19

to datapath elements. More specifically, datapath elements are those which operate on collections of bits having an interpretation as a data word. This includes cells such as comparators, shifters, adders, subtracters, counters, multipliers, buffers, registers, memories, etc. The blocks that do not fall under this category are those for which bit vectors do not have a collective interpretation. An example of this is control logic for which bits, typically, represent boolean flags and states rather than arithmetic quantities. A separate model called the activity-based control (ABC) model has been developed to handle RT-level power analysis of these elements [24]. Other techniques for controller power analysis involve building statistical models based on data from a set of benchmarks. For example, Mehra described a model for predicting controller power based on high-level parameters such as the number of states, the number of control bit outputs, and the number of status inputs [25].

VI. RESULTS

The DBT power analysis method has been implemented in a tool called SPA. This section presents results gathered using this power analysis tool. The first several case studies demonstrate the accuracy of the DBT model as applied to four common RTL building blocks: subtracters, shifters, multipliers, and memories. The final example shows how SPA was used to design a low-power digital filter. This real-world example illustrates SPA's ability to handle complex designs and provides a concrete demonstration of the power analysis/design methodology proposed in this paper.

A. Subtracter

The ripple-carry subtracter (and adder) are staples of any hardware library. Therefore, it is critical that any proposed modeling techniques perform well on these units. Before the power consumed by the subtracter can be analyzed, we must specify a capacitance model and derive a table of capacitive coefficients. For a linear unit, such as this, the appropriate capacitance model was derived in Section III-A:

$$C_T = C_{eff} N. \tag{20}$$

The table of coefficients resulting from the characterization of a subtracter in a 1.2 μm technology is given in Table III.

To test the overall accuracy of the DBT model, the time-multiplexed input stream for a 16-b subtracter in a sub-band speech coding filter was captured, applied to a switch-level simulator (irsim-cap), and also analyzed using the DBT method (see Fig. 13). The results are quite good, with the

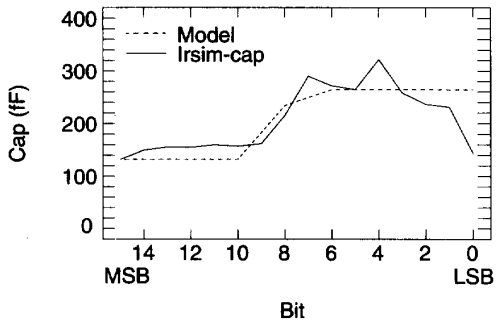

Fig. 13. Subtracter: irsim-cap versus DBT model.

DBT model achieving an overall error of only 0.14%. We might suspect that some error cancellation is occurring here and, indeed, when we split the error into its UWN and sign region components we get +5.1% and −12.9%, respectively. This is still quite good, however, since this level of accuracy is achieved based purely on an architecture-level model—that is, without any low-level run-time simulation nor any direct knowledge of the internal construction of the unit.

B. Logarithmic Shifter

The subtracter obeys a very simple linear capacitance model. The case of a logarithmic shifter illustrates what can be achieved for more complex models. The behavioral parameters for this example were presented in Section III-A and are: the word length, N; the maximum shift value, M; and the number of shift stages, $L = \lceil \log_2 (M+1) \rceil$. The current value, S, of the SHIFT input also affects the power consumption. This influence could be handled in two ways. Since, technically, SHIFT is a control input, the technique suggested in the preceding text would be to consider each SHIFT value as a different instruction and have a separate coefficient table for each case. This is not practical, however, since there may be 16, 32, or more possible values of S. Another solution would be to add S as an additional capacitance model parameter along with N, M, and L. This is a preferable solution since only one coefficient table would be required. With this addition, the new capacitance model for the shifter is given by:

$$C_T = C_0 N + C_1 L + C_2 NL + C_3 N^2 L + C_4 MNL + C_5 SNL. \tag{21}$$

Fig. 14 shows the fit of the extracted DBT model to simulation results for a left shift over a wide range of parameter values. The rms error over all cases is 9.9%.

C. Array Multiplier

The complexity of an array multiplier (Fig. 15) grows quadratically with the input word length. Specifically, if the two inputs have word lengths N_1 and N_2, respectively, then the multiplier will require $N_1 N_2$ full-adder cells. We refer to modules in which the inputs are laid out in a grid with a cell at each intersection as *bit-meshed*, as opposed to *bit-sliced*,

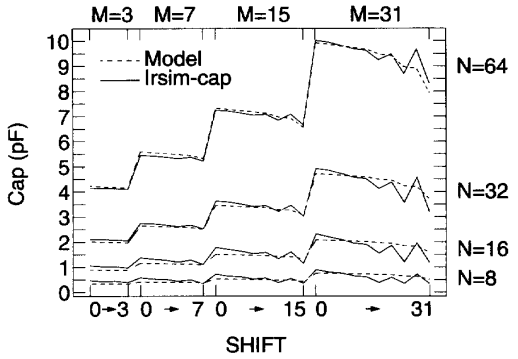

Fig. 14. Log shifter: irsim-cap versus DBT model.

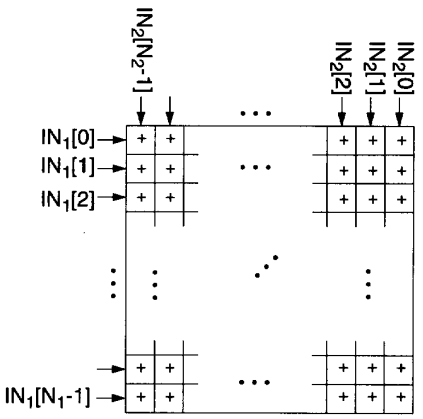

Fig. 15. Complexity of an array multiplier.

modules. The appropriate capacitance model for a bit-meshed module is given by

$$C_T = C_{eff} N_1 N_2. \tag{22}$$

The regional decomposition of a bit-meshed module differs slightly from the bit-sliced case. Each bit of a bit-sliced module is a function of the corresponding bits of the inputs. In other words, the transition template for bit i of a two-input module will be determined by the transition template for bit i of the two inputs: that is, $\mathrm{IN}_1[i]$ and $\mathrm{IN}_2[i]$. In contrast, the cells of a bit-meshed module bring together all combinations of input bits: $\mathrm{IN}_1[i]$ and $\mathrm{IN}_2[j]$. This results in the module decomposition shown in Fig. 16. Even though their interpretation differs, the capacitive coefficients of Table II still apply.

As a demonstration of model accuracy, the power of a multiplier within an LMS noise cancellation filter was analyzed using irsim-cap, as well as the DBT and UWN models. Irsim-cap placed the average capacitance switched per cycle at 40.2 pF, while the DBT model predicted 41.9 pF—an error of only 4.2%. In contrast, the UWN model put the multiplier capacitance at 64.8 pF—a substantial overestimate of 61.2%. Fig. 1 provides the reader with additional multiplier results.

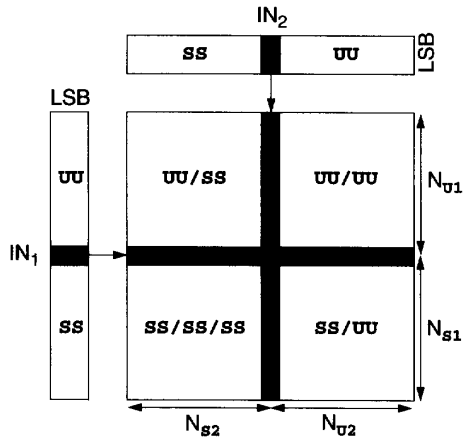

Fig. 16. Regional decomposition of an array multiplier.

D. Memory

VLSI chips, of course, consist of more than just computational elements—data must also be stored and retrieved in memory units. Memory can either be foreground or background. Foreground memory usually consists of register files, which are used to store small amounts of frequently accessed local data for rapid retrieval. Background memory typically refers to larger, denser, and slower storage units such as RAM's and ROM's, which may contain more global data or instructions. With relatively few modifications, the DBT model can be extended to handle memories.

The capacitance switched during an access to a memory module will be affected by the size, or complexity, of the module. Therefore, the capacitance formula must be parameterized in terms of the appropriate complexity measures. For example, a register file contains a total of W registers with N bits each. In determining the appropriate capacitance model for the register file we refer to Fig. 17. Accessing a register activates the input buffers, the output buffers, and a single register in the file. Each of these contains N bits. We, therefore, expect a capacitance term proportional to N. We also must broadcast the data across the entire register file traversing W words for each of N bits. This gives us a WN term in the overall capacitance expression. In addition, there might be some control overhead for each of the W registers in the file, giving rise to a W term in the formula and, perhaps, a constant overhead term for the entire file. Weighting each of these by a capacitive coefficient, gives us an aggregate capacitance model of

$$C_T = C_0 + C_1 W + C_2 N + C_3 WN. \tag{23}$$

Background memories (Fig. 18) such as SRAM's, DRAM's, and ROM's have a structure very similar to a register file and, therefore, can use the same capacitance model. This model assumes a memory partitioned into a single block of cells. Often, memories will be partitioned into several blocks, with only a subset of them powered-up for any one access. In this case, the form of the model remains the same, but the interpretation of the W and N parameters must be modified.

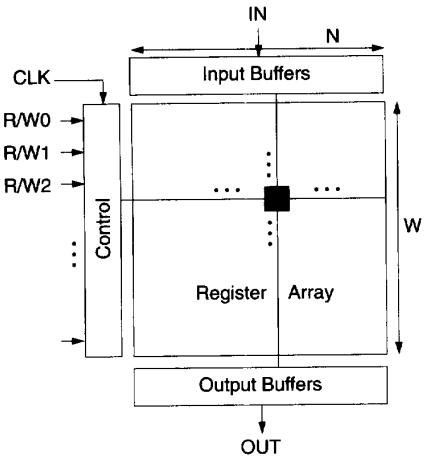

Fig. 17. Basic structure of a register file.

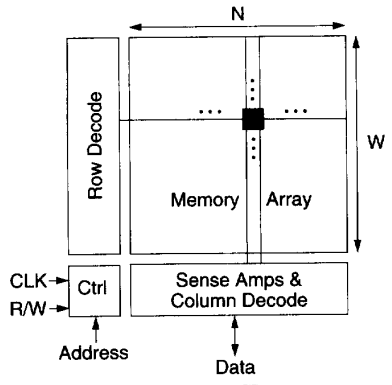

Fig. 18. Basic structure of a memory.

Specifically, W should be interpreted as the number of rows in the active memory blocks and N should be interpreted as the number of columns in the same blocks. Aside from this slight modification, the overall capacitance model remains unchanged.

Fig. 19 shows the results of a comparison between the capacitance model and switch-level simulations for register files of varying sizes. For the Read operation, the maximum error is only 5.7% and is even lower for the Write operation at about 4.2%. Similarly, for the SRAM Read and Write operations, the memory model provides estimated capacitances within 7.7% and 0.63%, respectively, of irsim-cap for memories ranging in size from 16 × 4 to 128 × 32.

E. Quadrature Mirror Filter

SPA allows the designer to efficiently explore the design space, searching for low-power solutions. This example will demonstrate a design flow that employs SPA to minimize the power consumed by a quadrature mirror filter (QMF) such as might be used in a sub-band coding algorithm [26]. A quadrature mirror filter takes an input signal and splits it into two bands: a low-pass band, $H_{LP}(\omega)$, and a high-pass band, $H_{HP}(\omega)$. The sample rate chosen for the filter would,

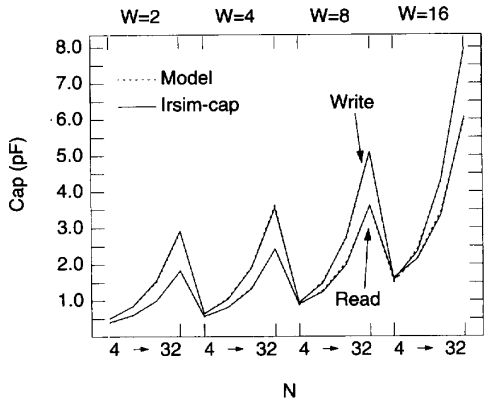

Fig. 19. Register file: irsim-cap versus model.

TABLE IV
SPA POWER/AREA PREDICTIONS FOR THE QMF EXAMPLE

	Initial	Shift-Add	Retimed	Pipelined
V_{dd} (V)	5	5	1.5	1.25
Power (mW)	348.8	149.5	21.0	7.0
Area (mm^2)	95.9	17.2	24.4	115.4

in general, depend on the application. For the purposes of this example, we select a sample period of 0.3 μs or about 3.33 MHz. Four candidate architectures were explored using SPA.

The first version is a direct, naive implementation of the algorithm. The power and area predictions provided by SPA are shown in the "Initial" column of Table IV. Four costly array multipliers are required to meet the throughput requirements of the algorithm at 5 V and this leads to a large die size of 95.9 mm^2.

In the second version of the design, the expensive array multipliers are replaced by shift-add operations, reducing the chip area to a more reasonable 17.2 mm^2. SPA reveals that this version of the chip consumes 57% less power than the initial version, while at the same time occupying 82% less area.

A third implementation of the QMF example was generated by applying retiming, which reduces the critical path to 60 ns at 5 V. This allows us to lower the supply voltage of the implementation to about 1.5 V while still meeting the sampling rate constraint. Analysis using SPA shows a 7.1× reduction in power. Additional hardware requirements, however, increase the implementation area from 17.2 mm^2 to 24.4 mm^2.

A fourth version of the filter can be generated by pipelining the algorithm enabling a fully parallel implementation. This further reduces the critical path and allows the voltage supply to be reduced to 1.25 V. SPA confirms an additional power reduction of 3× for an overall reduction (from version one to version four) of 50×. Interestingly enough, voltage reduction accounts for only 46% of the power saved by going from the retimed to the pipelined design. Fully 54% of the power saved by pipelining can be attributed to a distributed architecture which preserves signal correlations and, thus, minimizes switching activity. SPA is able to model these effects, but traditional estimators based on white-noise activity models are not.

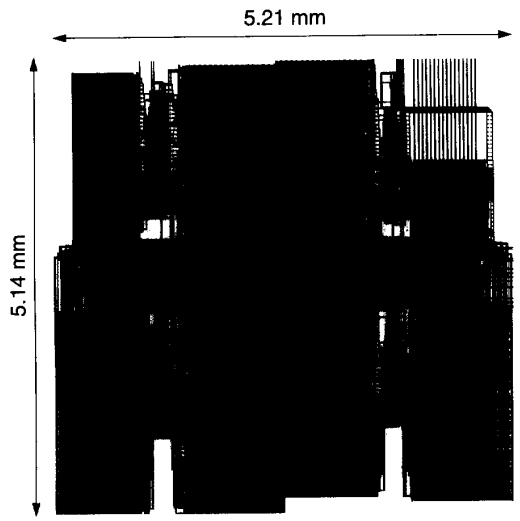

Fig. 20. Layout of the retimed QMF design.

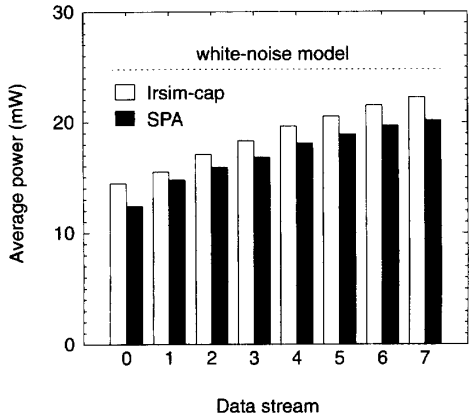

Fig. 21. Comparison of SPA to irsim-cap for the retimed QMF design.

While the pipelined example at 7 mW consumes less power than the 21 mW retimed design, it is at the cost of a 4.7× area increase. As a result, the retimed example, which is still 17× lower power than the initial solution and requires only 24.4 mm^2, is probably a more desirable solution.

To verify that SPA provided accurate area and power estimates, this version of the filter has been synthesized (down to layout), extracted, and simulated. The chip plot is shown in Fig. 20. The predicted area of 24.4 mm^2 is within 9% of the actual 26.8 mm^2 area. A comparison of the SPA power predictions to switch-level simulation using irsim-cap is given in Fig. 21. The figure shows the average power consumed by the chip for data streams corresponding to increasing input signal powers. SPA's estimates are within 5% to 14% of irsim-cap for all data streams. Estimates based on the white-noise model are also included. The white-noise estimates do not track signal statistics and, therefore, err by as much as 71% for some of the data streams.

By using SPA to compare four candidate architectures we were able to significantly reduce design time. The four filter versions described here were synthesized using the Hyper high-level synthesis system [27] and analyzed with SPA in 5 minutes on a Sun SPARCstation 10. In contrast, laying out and simulating the retimed version took 3.2 hours. Laying out and analyzing all four designs using low-level power analysis tools would have required 13 hours or more.

VII. Conclusions

In summary, previous attempts at architectural datapath power models have left much to be desired in terms of accuracy. These inaccuracies can be attributed to the fact that standard UWN-based models ignore the influence of sign bits and signal correlations on power consumption. The Dual Bit Type (DBT) model presented in this paper addresses this concern by accounting for both LSB and MSB (sign) behavior. Under this model, datapath library cells can be characterized for both types of bits. This process results in a table of effective capacitance coefficients, which allow the power analysis process to be reduced to little more than a series of table-lookups. In addition, the DBT power analysis method uses parameterizable capacitance models that accurately reflect the effect of complexity, as well as activity on module power consumption. Aside from the coefficient table, the inputs to the analysis process are a small number of word level statistics describing the inputs to the module—in particular, mean, variance, correlation, and sign transition probabilities.

A power/area analysis tool called SPA has been developed, which employs the DBT model. A version of SPA resides within Hyper—a suite of tools targeting the high level synthesis of DSP systems [27]. A stand-alone version of SPA has also been implemented using VHDL as an RTL input language and simulation platform [28].

Appendix A
Derivation of Breakpoint Formulas

The position of the model breakpoints depends on how many bits are required to represent the numbers in the data stream; the unused bits will be devoted to sign. First, consider the high-end breakpoint, $BP1$. The bits between $BP1$ and the MSB are all sign bits. Therefore, the high-end breakpoint corresponds to the maximum number of nonsign bits required to represent all likely values of the data word. The range of likely values is illustrated by Fig. 22, which shows the probability density function (PDF) for a Gaussian variable X_t. The curve denotes the probability that a particular value of X_t will occur. This probability is highest at the mean (μ) and decays to almost zero within three standard deviations (3σ). So the most common values of X_t fall within the range $R_x = [\mu - 3\sigma, \mu + 3\sigma]$. Since the largest likely magnitude of the signal, then, is $|\mu| + 3\sigma$, the maximum number of nonsign (data) bits will be:

$$BP1 = \log_2 (|\mu| + 3\sigma). \tag{24}$$

Now consider the lower breakpoint, $BP0$. Bits below $BP0$ behave randomly—like uniform white noise bits. This behavior suggests that the bits from the LSB to $BP0$ are almost completely uncorrelated with the other bits of X_t.

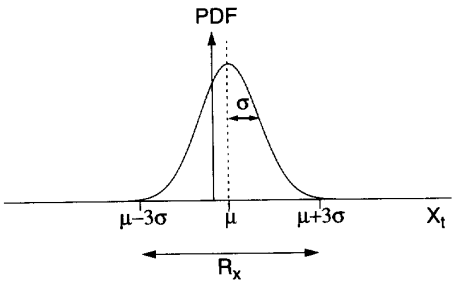

Fig. 22. Range of likely values for X_t based on its PDF.

Therefore, $BP0$ occurs at the least significant bit that begins to exhibit some correlation to the word value of X_t. This correlation first appears when the significance (or place value) of the bit first approaches the "size" of the range of values covered by X_t. The standard deviation (σ) is a good measure of the smallest significant range of values covered by X_t. Since the place value of bit i is 2^i, $BP0$ occurs at:

$$BP0 = \log_2 \sigma. \tag{25}$$

This analysis was performed using a univariate approach (considering only X_t). Technically, the breakpoints of $P(0 \rightarrow 1)$ depend on both X_{t-1} and X_t. Still, (25) remains valid when X_{t-1} and X_t are uncorrelated ($\rho = 0$); however, when the signals are correlated, ρ can have an effect on $BP0$ as Fig. 3 shows. The joint PDF for X_{t-1} and X_t with a significant correlation ($\rho = 0.8$) is shown in Fig. 23. Notice that the 1-D measure of distribution "spread," σ, has been replaced by the 2-D notion of a unit variance ellipse, or σ-contour. When calculating $BP0$, there are now two measures of "spread" in the X_t direction to consider. The first relates to the width of the ellipse, and can be quantified as the standard deviation of X_t conditional on X_{t-1}: $\sigma(X_t \mid X_{t-1}) = \sigma\sqrt{1-\rho^2}$. The second is related to the skew of the ellipse from the $X_t = 0$ axis. This term becomes important as $|\rho| \rightarrow 1$ and the ellipse collapses to its horizontal centroid: $X_t = \rho X_{t-1}$. It is important because even though the width of the ellipse goes to zero, the skew of its centroid still allows it to cover a significant range of X_t values. Since the centroid follows $X_t = \rho X_{t-1}$ and extends from the origin to $X_{t-1} = \pm\sigma$, its length along the positive X_t axis is $|\rho|\sigma$. At least a fraction, f, of this should contribute to the "spread" of the distribution. Combining the ellipse width and skew measures yields $\sigma' = \sigma\left[\sqrt{1-\rho^2} + f|\rho|\right]$ where the first term dominates for small correlations (wide ellipses) and the second term dominates for large correlations. The result of replacing σ with σ' in (25) is an offset to the uncorrelated value of $BP0$:

$$BP0 = \log_2 \sigma + \Delta BP0 \tag{26}$$

$$\Delta BP0 = \log_2 [\sqrt{1-\rho^2} + |\rho|/8] \tag{27}$$

where f has been replaced by the empirically derived value of $\frac{1}{8}$.

Equations (24), (26), and (27) fully describe how to determine the breakpoints of the DBT data model. The equations express the breakpoints as a function of word-level statistics such as mean (μ), variance (σ^2), and correlation (ρ). Although the derivation of these equations utilized Gaussian random variables for the purposes of illustration, the resulting breakpoint formulas are fairly independent of the underlying distribution. Altering the distribution merely introduces a small multiplicative constant to the σ terms in the above equations, but the effect of this is minimized by the logarithmic form of the breakpoint equations.

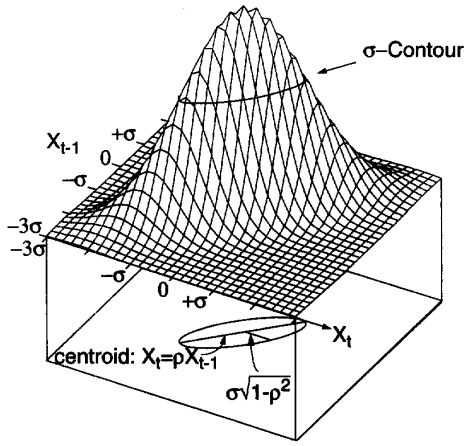

Fig. 23. Joint PDF of X_{t-1} and $X_t(\rho = 0.8)$.

ACKNOWLEDGMENT

The authors wish to recognize the support provided by the developers of the Hyper system, as well as, the useful editorial suggestions offered by A. Burstein, R. Mehra, and K. Landman.

REFERENCES

[1] W. Bowhill *et al.*, "A 300 MHz 64b quad-issue CMOS RISC microprocessor," in *ISSCC'95 Digest of Tech. Papers,* Feb. 1995, pp. 182–183.
[2] A. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS design," *IEEE J. Solid-State Circuits,* pp. 472–484, Apr. 1992.
[3] P. Duncan, S. Swamy, and R. Jain, "Low-power DSP circuit design using retimed maximally parallel architectures," in *Proc. 1st Symp. Integrated Syst.,* Mar. 1993, pp. 266–275.
[4] F. Najm, I. Hajj, and P. Yang, "An extension of probabilistic simulation for reliability analysis of CMOS VLSI circuits," *IEEE Trans. Computer-Aided Design,* pp. 1372–1381, Nov. 1991.
[5] ——, "Probabilistic simulation for reliability analysis of CMOS VLSI circuits," *IEEE Trans. Computer-Aided Design,* pp. 439–450, Apr. 1990.
[6] R. Tjarnstrom, "Power dissipation estimate by switch level simulation," in *Proc. IEEE Int. Symp. Circuits and Syst.,* May 1989, pp. 881–884.
[7] M. A. Cirit, "Estimating dynamic power consumption of CMOS circuits," in *Proc. IEEE Int. Conf. Computer Aided Design,* Nov. 1987, pp. 534–537.
[8] L. W. Nagel, "SPICE2: A computer program to simulate semiconductor circuits," Univ. California, Berkeley, Tech. Rep. ERL-M520, 1975.
[9] S. M. Kang, "Accurate simulation of power dissipation in VLSI circuits," *IEEE J. Solid-State Circuits,* pp. 889–891, Oct. 1986.
[10] A. Salz and M. Horowitz, "IRSIM: An incremental MOS switch-level simulator," in *Proc. 26th Design Automation Conf.,* 1989, pp. 173–178.
[11] J. White, S. Devadas, and K. Keutzer, "Estimation of power dissipation in CMOS combinational circuits using boolean function manipulation," *IEEE Trans. Computer-Aided Design,* pp. 377–383, Mar. 1992.
[12] B. Hoppe, "Optimization of high speed CMOS logic circuits, with analytical model for signal delay, chip area, and dynamic power dissipation," *IEEE Trans. Computer Design,* pp. 236–247, Mar. 1990.

[13] A. Shen, A. Ghosh, S. Devadas, and K. Keutzer, "On average power dissipation and random pattern testability of CMOS combinational logic networks," in *Proc. Int. Conf. Computer-Aided Design,* Nov. 1992, pp. 402–407.
[14] R. Burch, F. Najm, P. Yang, and T. Trick, "McPower: A Monte Carlo approach to power estimation," in *Proc. Int. Conf. Computer-Aided Design,* Nov. 1992, pp. 90–97.
[15] A. Ghosh, S. Devadas, K. Keutzer, and J. White, "Estimation of average switching activity in combinational and sequential circuits," in *Proc. 29th Design Automation Conf.,* June 1992, pp. 253–259.
[16] F. Najm, "Transition density, a stochastic measure of activity in digital circuits," in *Proc. 28th Design Automation Conf.,* June 1991, pp. 644–649.
[17] C.-Y. Tsui, M. Pedram, and A. Despain, "Efficient estimation of dynamic power consumption under a real delay model," in *Proc. Int. Conf. Computer-Aided Design'93,* 1993, pp. 224–228.
[18] S. R. Powell and P. M. Chau, "Estimating power dissipation of VLSI signal processing chips: The PFA technique," *VLSI Signal Processing IV,* pp. 250–259, 1990.
[19] S. Powell and P. Chau, "A model for estimating power dissipation in a class of DSP VLSI chips," *IEEE Trans. Circuits and Syst.,* pp. 646–650, June 1991.
[20] J. Ward *et al.,* "Figures of merit for VLSI implementations of digital signal processing algorithms," *Proc. IEEE,* vol. 131, part F, pp. 64–70, Feb. 1984.
[21] P. Landman and J. Rabaey, "Power estimation for high level synthesis," in *Proc. EDAC-EUROASIC '93,* Paris, France, Feb. 1993, pp. 361–366.
[22] D. A. Preece, "Distributions of the final digits in data," *The Statistician,* vol. 30, no. 1, pp. 31–60, Mar. 1981.
[23] P. Van Oostende, P. Six, J. Vandewalle, and H. De Man, "Estimation of typical power of synchronous CMOS circuits using a hierarchy of simulators," *IEEE J. Solid-State Circuits,* vol. 28, no. 1, Jan. 1993.
[24] P. Landman and J. Rabaey, "Activity-sensitive architectural power analysis for the control path," in *Int. Symp. Low-Power Design'95,* Apr. 1995.
[25] R. Mehra and J. Rabaey, "High-level power estimation and exploration," in *1994 Int. Workshop on Low Power Design,* Apr. 1994.
[26] N. Jayant and P. Noll, *Digital Coding of Waveforms.* Englewood Cliffs, NJ: Prentice-Hall Signal Processing Series, 1984.
[27] J. M. Rabaey, C. Chu, P. Hoang, and M. Potkonjak, "Fast prototyping of datapath-intensive architectures," *IEEE Design & Test of Comput.,* pp. 40–51, June 1991.
[28] P. Landman, "Low-power architectural design methodologies," Ph.D. dissertation, Univ. of California. Berkeley, Aug. 1994.

Paul E. Landman received the B.S., M.S., and Ph.D. degrees in electrical engineering and computer science from the University of California, Berkeley, in 1989, 1991, 1994, respectively.

His research at Berkeley focused on low-power digital design techniques and tools with an emphasis on DSP applications. After completing his dissertation, he joined the Integrated Systems Laboratory at Texas Instruments, Dallas. He is currently developing low-power algorithms, architectures, and circuits for portable video applications.

Dr. Landman is a National Science Foundation Fellowship recipient and a member of Tau Beta Pi and Eta Kappa Nu.

Jan M. Rabaey received the EE and Ph.D. degrees in applied science from the Katholieke Universiteit Leuven, Belgium, in 1978 and 1983, respectively.

From 1983 to 1985, he was connected to the University of California, Berkeley as a Visiting Research Engineer. From 1985 to 1987, he was a Research Manager at IMEC, Belgium, where he pioneered the development of the CATHEDRALII synthesis system for digital signal processing. In 1987 he joined the faculty of the Electrical Engineering and Computer Science Department at the University of California at Berkeley, where he is now a Professor. He has authored or coauthored more than 100 papers in the area of signal processing and design automation. His research interests include the exploration of architectures and algorithms for digital signal processing systems and their interaction. He is also active in various aspects of portable, distributed communication and computation systems, including low-power design, networking and design applications.