Effective Capacitance Macro-Modelling for Architectural-Level Power Estimation

Muhammad M. Khellah and M. I. Elmasry
E&CE Department, University of Waterloo, Canada

Abstract

This paper presents a simple, yet efficient method to characterize the effective capacitance in data-path macros for architectural-level power estimation. Given a library of hard-macros, a capacitance model based on linear regression is derived for each macro. A transistor-level tool is employed for capacitance extraction. The capacitance models can be used during architectural-level power estimation. Unlike previous approaches, our characterization methodology assumes no specific word-level statistics of the input data, requires little knowledge about the structure of the modules, allows the user to trade-off accuracy and characterization time, and propagates effective capacitance directly from transistor-level (real) implementations. Simulation experiments on a set of data-path components with various sizes are performed. Compared to a previously published approach [1], our scheme significantly improves the accuracy of RTL power estimation and produces results within 15% from a transistor-level tool on the average.

1 Introduction

Low-power consumption has become a primary target in contemporary digital VLSI design. Two main factors are behind this trend. First, increased circuit speed accompanied with high integration densities have resulted in significantly large heat dissipations. Unless power consumption is brought down to acceptable levels, reliability is affected and expensive cooling and packaging systems are required. Second, battery-driven mobile electronics put a high demand on low-power dissipation in order to extend the battery life to the maximum possible level before recharging is needed.

The above two factors have added to the complexity of digital IC design since low power dissipation has become an important design goal as high-speed and small area. To minimize the power consumption of a digital system, the design that dissipates the least power has to be chosen provided that it meets some speed performance and/or area constraints. To aid designers to achieve that in a reasonable amount of time, power-sensible CAD tools at all levels of the design hierarchy have been developed over the past few years [2].

More recently, particular attention has been given to high-level (that is, RTL and above) power optimization techniques since design experience and research efforts have shown that decisions made at the highest levels could have the largest impacts on power savings. To assess the effects of these decisions on power consumption, transistor-level power simulators (for example, PowerMill [3]), can be used. However, this approach is extremely unpractical because: (1) the number of possible high-level decisions might be substantial, and (2) modern circuits may contain a large number of components with each composed of thousands of transistors. This motivates the need for developing a very fast, yet (relatively) accurate tool for estimating power at the high-level.

As an initial step towards such a tool, a new technique to ***characterize*** the power dissipation in data-path components (that is, adders, multipliers, etc.) is presented in this paper. Given a hard data-path module with N primary inputs, and M primary outputs, exactly $N \times (N + M + 1)$ effective capacitance coefficients are derived for the module. A transistor-level tool is used for capacitance extraction. Our methodology is a based on linear regression theory. The salient features of our ***characterization*** methodology are: 1) it makes no assumption about input data statistics, 2) it requires little knowledge about the structure of the modules, 3) it allows the user to trade-off accuracy and characterization time, and 4) it propagates effective capacitance directly from transistor-level (real) implementations.

The rest of this paper is organized as follows. In the next Section, the main sources of power dissipation in static MOS circuits are described. In Section 3, we briefly review previous related work. Section 4 presents our power characterization technique. In Section 5, we give simulation results to verify the applicability of the approach. Section 6 provides conclusions and directions for future work.

0-8186-8409-7/98 $10.00

2 Power Dissipation in Digital Circuits

In well-designed static MOS digital circuits, such as CMOS and CPL circuits (with swing restoration), power is mostly dissipated as a result of input signal transitions. This power is called *dynamic power* and it has two components: capacitive power and short-circuit power. *Capacitive power* is used for charging/discharging capacitive loads during logic transitions. For a circuit with S gates, this power is given by:

$$P = 0.5 \times \left(\sum_{i=1}^{S} \alpha_i C_i \right) \times V_{dd}^{\ 2} \times f_{clk} \qquad (1)$$

where C_i is the lumped capacitive load at the output node of gate i, V_{dd} is the supply voltage, f_{clk} is the clock frequency which controls the rate at which the circuit's primary inputs change values and, α_i is the *switching activity* defined as the probability the output node of gate i will make a transition during a clock cycle ($1/f_{clk}$), for i = $1,2,..,S$. The term between the parentheses in equation (1) represents the average switched capacitance per clock cycle (C_{cap}).

In a CMOS gate, *short-circuit* power is consumed by the transient current that flows between V_{dd} and ground when both the NMOS and PMOS devices are turned on during logic transitions. Similar to the capacitive-power, the short-circuit power can be approximated through an equivalent short-circuit capacitance (C_{sc}). C_{sc} is usually much smaller than C_{cap}. However, in some cases, it represents a significant portion and should not be neglected. Thus, the total effective capacitance is given by:

$$C_{eff} = C_{cap} + C_{sc} \qquad (2)$$

Because C_{eff} depends on the input activities, the problem of dynamic power estimation is known as highly input pattern dependent.

3 Previous Related Work

Given a module, it is infeasible to characterize its power dissipation for every possible input vector transition since the total number of these transitions is exponential. Thus, it becomes important to select a subset of these transitions and use them to model C_{eff}. The way this subset is chosen forms the main difference in existing power characterization techniques.

The Dual-Bit Type (DBT) model [4] was the first to address the input data dependence problem. For a given module, the DBT model derives a number of capacitive coefficients which reflect the different behaviors of bits in 2's complement data streams. Although accurate, the main limitation of this approach is that it is biased for a specific input behavior and data representation. In addition, the characterization process requires some knowledge about the basic structure of the modules.

In [5], block-level piece-wise linear modelling of power that captures the variation of output glitching activity with various word-level parameters at the block's inputs was proposed. Similar to the DBT model, this method derives power coefficients assuming a particular input data distribution.

Recently, a power characterization approach based on clustering was described in [6]. This technique does not have the disadvantages of the DBT model. The basis is to collapse closely related input vector transitions into effective capacitive coefficients (called clusters) and to store these coefficients in a library. For a good accuracy, the number of clusters can be very large especially for a big circuit. Thus, looking-up the right cluster during power estimation is a time consuming process.

L. Benini et. al. first proposed power characterization based on regression theory [1]. The main idea is to model the power as a linear function of single input and output bit activities. The coefficients of the regression model are derived by simulating the module at the gate-level using a very large sample size and then solve an over-defined system of equations using least-square fitting techniques. If a circuit has N primary **data** inputs ($I_1,I_2,...,I_N$) and M primary **data** outputs ($O_1,O_2,...,O_M$), then a single equation is obtained to model the power as given below:

$$P = \beta_0 + \beta_1 I_1 + \ldots + \beta_N I_N + \beta_{N+1} O_1 + \beta_{N+2} O_2 + \ldots + \beta_{N+M+1} O_M \qquad (3)$$

During architectural-power estimation, an I (O) in the above equation takes the value of "1" if there is a transition on the corresponding input (output) line, other wise it takes a value of "0". The above technique is attractive because of its simplicity. Only ($M+N+1$) β coefficients have to derived and stored for each module. These coefficients can be quickly found during power estimation. This can be done, for example, by XORing consecutive input (output) vectors as shown in Figure 1.

As mentioned before, to obtain the β coefficients, a sample of input and output bits activities is needed. Given this sample, the module is simulated using accurate gate-level power simulators and for every input vector transition, the I and O values along with the dissipated power are recorded. The I and O values are stored in a matrix A. The ith row of this matrix has the value

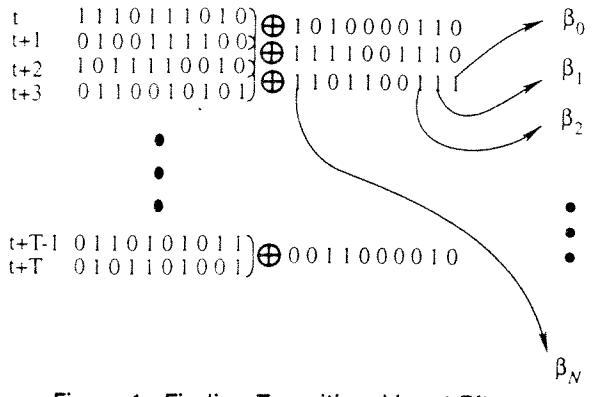

Figure 1: Finding Transitional Input Bits.

($1, I'_1, I'_2, \ldots, I'_N, O'_1, O'_2, \ldots, O'_M$). The power values are stored in a vector P. At the end of the simulation, we have the linear system of equations:

$$P = A\beta \qquad (4)$$

where P is a vector of size [*sample_size x 1*], A is a matrix of size [*sample_size x M+N+1*], and β is a vector of size [*N+M+1 x 1*]. To overdefine the system, the *sample_size* must be much larger than *(M+N+1)*. By using least-square fitting techniques to solve the above system, equation (3) is derived.

It is noted in [1] that the least square fitting always produces an estimate of P with the same average value as the average value of the sample employed for fitting. If the input patterns used for fitting are uniformly distributed and independent (which must be the case in order not to be biased to a particular input data behavior), then this guarantees that the Linear Regression (**LR**) model will perform at least as good as using UWN (average) value approximation. In this case, β_0 in the above equation approaches the average value of the sample with the rest of the coefficients being (additive/subtractive) correction terms. If the module is used in a design with highly correlated input data patterns, then adding β_0 to the total power for every pair of input data can significantly degrade the estimation accuracy.

In this paper, we propose two important modifications to the LR approach. First, we significantly improves the accuracy of the model by introducing a piece-wise linear regression techniques. Second, we propose a Monte-Carlo based simulation approach to limit the size of the sample needed to extract the power coefficients. Limiting the sample size is critical in order to reduce the characterization time especially if accurate (and more versatile) **transistor-level** simulators are to be used to measure the power.

4 Modelling Effective Capacitance

In static MOS circuit, it is well-known that there is a good correlation between input and output bits transitions and C_{eff}[1]. Several simulations were performed to verify this observation. One result is shown in Figure 2 for a 5x5 array multiplier ($N = M = 10$). The figure shows two curves: the upper plots C_{eff} as a function of the total number of input bit transitions, *IN*, which in this case can range from 1 to 10, and the lower curve plots C_{eff} as a function of the total number of output bit transitions, *OUT*, (which again can be from 1 to 10). Each point on each curve represents the average switched capacitance over a very large number of randomly generated runs, where each run satisfies an *IN (OUT)* value (see the onset in Figure 2). A **transistor-level** power simulation tool (**PowerMill**), was used for capacitance extraction.

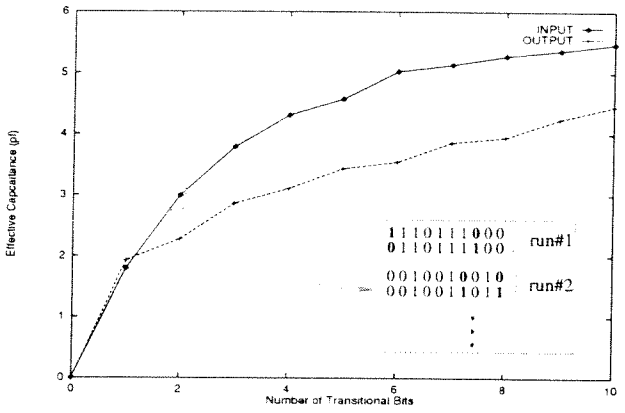

Figure 2: Dependence of C_{eff} on the Number of Input and Output Bit Transitions for a 5x5 Array Multiplier.

Figure 2 clearly illustrates the strong dependency between C_{eff} and *IN* (*OUT*). In addition, it shows that a large difference in C_{eff} (up to 3 times in this case) can result depending on the number of transitional bits. Similar behaviors were also noticed for other data-path modules with various sizes. Although this correlation is not necessarily linear, it gives a good *validation* to the simple LR approach as pointed in [1]. Moreover, the LR approach gives more details than Figure 2 since it accounts for individual input and output bit activities.

As pointed out in the previous section, the main limitation of the LR technique is that it can be extremely inaccurate if employed to estimate the power when highly correlated input data sequence is used. The reason for this inaccuracy is that the least-square fitting solution will pro-

1 Extension of our approach for multi-functional modules is discussed at the end of Section 4.

duce a β_0 term that converges to the average value of the random sample used for fitting. Adding this term for every input vector transition during architectural-level power estimation greatly over-estimate the actual power. 3

In order to overcome this problem, we propose a modification to the LR approach. The basic idea is to generate an *IN* number of LR equations. That is, one equation is derived for *each* possible value of *IN*, where *IN* = *1,...,N*. For each value of *IN*, a random sample is generated such as between every consecutive input vectors exactly *IN* input bits are allowed to change (see the onset in Figure 2) and least-square fitting techniques are used as before to produce a *set* of LR models as given below:

$$C_{eff}^{IN=1} = \beta_0^{IN=1} + \sum_{i=1}^{N} \beta_i^{IN=1} \cdot I_i + \sum_{j=N+1}^{N+M+1} \beta_j^{IN=1} \cdot O_j$$

$$C_{eff}^{IN=2} = \beta_0^{IN=2} + \sum_{i=1}^{N} \beta_i^{IN=2} \cdot I_i + \sum_{j=N+1}^{N+M+1} \beta_j^{IN=2} \cdot O_j$$

$$\vdots \qquad (5)$$

$$C_{eff}^{IN=N} = \beta_0^{IN=N} + \sum_{i=1}^{N} \beta_i^{IN=N} \cdot I_i + \sum_{j=N+1}^{N+M+1} \beta_j^{IN=N} \cdot O_j$$

The rationale behind this technique is that C_{eff} has a direct relationship with *IN* (see Figure 2) and therefore it is intuitively more accurate to use this dependance to construct piece-wise linear models. In this case, the minimum mean square error obtained from fitting is not global (over all possible values of *IN*) but rather local (over a single value of *IN*). As a result, the constant ($\beta^{IN}{}_0$) will be confined to the average capacitance over a single value of *IN*.

The main limitation of the above approach, henceforth called Modified LR (MLR), is that it requires long characterization time. Instead of performing one simulation using a very large sample, we need to execute ***N simulations one for each value of (IN)***. This can take very long time, especially if transistor-level tools are used for extracting the switched capacitance. To handle this problem, we use a Monte Carlo based scheme to execute each simulation.

Power modelling using Monte Carlo simulations is commonly used to find the average power of a module using random/user-specified input data [7]. Instead, we employ it here to quickly find the average power when the module is stimulated using input vectors with a particular *IN* value. For each simulation (value of *IN*), the sample size is determined by the number of input patterns needed to make ($C^{IN}{}_{eff}$) reaches convergence. As a result, we do not need to use a very large sample for each value of *IN*. Each simulation is executed as follows. For a given number of input bit transitions *IN*, the module is simulated *T* times (**using a transistor-level simulator**) where in each time (run), a *randomly generated* pair of consecutive input vectors that satisfy *IN* is used. This ensures that the characterization procedure is not biased for a specific input data. The *T* switched capacitance values obtained from simulation are then added up and divided by *T* to get an estimate of the effective capacitance. The above process is repeated *R* times until ($C^{IN}{}_{eff}$) converges. From Monte Carlo simulation techniques, a lower bound on *R* required to have (1-ξ)x100% confidence that $\left|C^{IN}{}_{R} - C^{IN}{}_{eff}\right| < \frac{s \times t_{\xi/2}}{\sqrt{R}}$, is:

$$R \geq \left(\frac{s \times t_{\xi/2}}{\varepsilon \times C^{IN}{}_{R}} \right)^2 \qquad (6)$$

where ($C^{IN}{}_{eff}$) is the *actual* effective capacitance which is unknown, ($C^{IN}{}_{R}$) is the *measured* capacitance calculated as the sum of the *R* effective capacitances divided by *R*, (*s*) is the standard deviation among the *R* values, $t_{\xi/2}$ is obtained from the t-distribution with (*R*-1) degrees of freedom, and ε is the maximum desired error in ($C^{IN}{}_{eff}$). Both ε and ξ can be set by the user to trade-off between characterization time and accuracy. The *I* and *O* values of each vector pair are stored along with thier switched capacitance and at the end of the simulation are used to derive the corresponding LR equation.

Notice that during the characterization process exactly *2xTxR* input vectors are simulated for a particular *IN* value. Another possible way which cuts the characterization time in half is to use the destination vector of each pair as the starting vector of the next pair. Thus, the total number of vectors simulated is *(T+1)xR*. The above technique was found faster, however, because of the simulator used for capacitance extraction.

The value *T* is fixed for all values of *IN*. However, the value *R* changes and depends on (other than the user-specified ε and ξ): (1) the standard deviation among the *R* values (*s*) and (2) the magnitude of ($C^{IN}{}_{eff}$.) For small values of *IN* (highly correlated input patterns), (*s*) is high since the effective capacitance is greatly affected not only by *IN* but also by the state of the non-switching input signals. In addition, the effective capacitance ($C^{IN}{}_{eff}$) is the very small. Both these factors forces *R* to be large (See Eq. 6). As *IN* gets larger, (*s*) gets smaller and ($C^{IN}{}_{eff}$) gets larger, decreasing *R* as a result. Thus, for each value of *IN*, this technique automatically adjusts the sample size to enable convergence and cuts the characterization time. As an

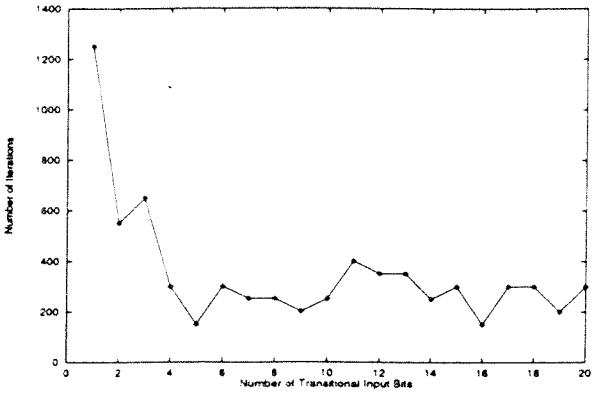

Figure 3: Number of Cycles Required To Characterize 10x10 Array Multiplier (IN = 1,2,...,20).

example Figure 3 shows the number of cycles required to characterize a 10x10 array multiplier. Notice how the number of cycles needed decreases as *IN* gets large.

The use of a Monte Carlo based technique in the characterization process has the implicit assumption that (C^{IN}_{eff}) has a normal (Gaussian) distribution. We checked the normality hypothesis by plotting the distribution of (C^{IN}_{eff}) obtained from several modules. A typical example is shown in Figure 4 for a 5x5 array multiplier.: the bell-like curve for each (C^{IN}_{eff}) value is similar to that of a normal distribution. This observation is not assumed to be general but we found it to valid across several data-path modules. Notice that for smaller values of (*IN*), the curves are more distorted and wider than for larger values (which is why the Monte Carlo technique requires more iterations for these values).

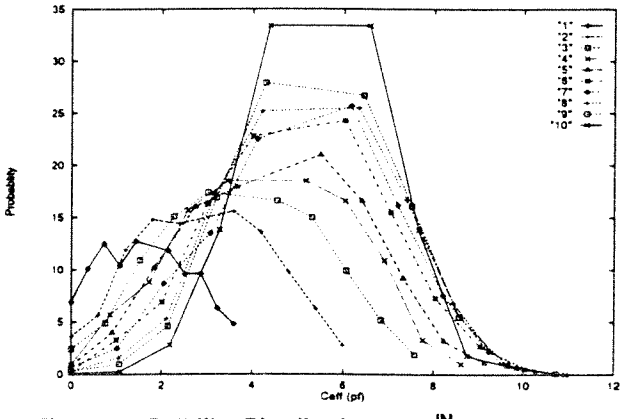

Figure 4: Bell-like Distribution of (C^{IN}_{eff}) for a 5x5 Array Multiplier (IN = 1,2,...,10).

A summary of our characterization approach is shown in Figure 5. This procedure is only valid for uni-functional modules. For multi-functional units (an ALU for example), the procedure has to be repeated for every possible setting at the control inputs. Thus, a unique set of linear regression equations must be computed for each distinct function that the module may be called on to execute. During power estimation, the control inputs are used to select among the different regression functions.

When we apply our technique during power estimation, typical/user-specified input data can be used. For each pair of consecutive input vectors, the *I* and *O* values are found as well as the value of *IN*. This value is used to select the LR equation that was derived using input patterns having the same *IN*. At the end of the simulation, all cycle capacitances are added up and the average switched capacitance is obtained. To find the actual power dissipation, Eq. 3 can be used.

5 Results

We tested our methodology on a set of data-path circuits: a 16-bit Ripple Carry Adder (RCA), 15x15 Array Multiplier (AM), a 16-bit Conditional Sum Adder (CSA), a 6x6 modified Booth Multiplier (BM), a 16-bit Magnitude Comparator (COMP), and an 8-by-4 Divider (DIV). Each circuit was laid out in 0.8μm CMOS technology. The layout was extracted, and the netlist was translated into PowerMill format.

The capacitance characterization procedure in Figure 5 was applied for each circuit. The least-square fitter in MatLab was used for model generation. The following parameters were used for the Monte Carlo simulations: ε = 0.10 (that is, maximum 10% error), ξ = 0.01 (that is, 99% confidence level), and T = 50 cycles. We also simulated each circuit using a very large sample and derived a *single* LR equation (as in [1]).

To test the performance of the proposed approach, two long data streams were used: (1) **Random** where the

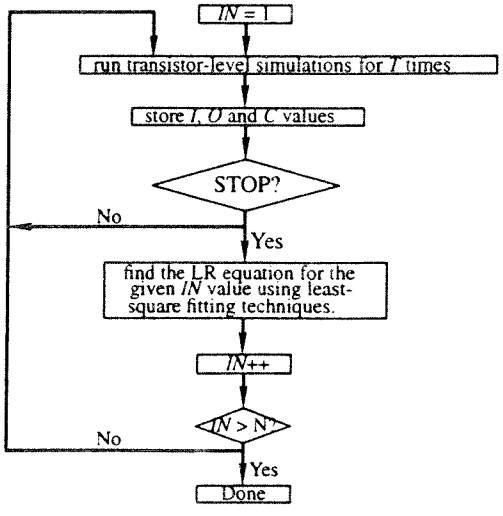

Figure 5: Summary of the Capacitance Characterization Technique.

transition activity of each single input line was set to about 0.5, and (2) **Correlated** where the average switching activity per bit was about 0.15 and random correlations between different bits were generated using the state lines of an up counter (similar biased data sequence was used in [1]). Notice that the way the correlated stream was created is different from the method used for characterization.

Table 1 summarizes the characteristics of each circuit and also gives the accurate effective capacitance values obtained from PowerMill for the two testing streams. For each module, note the large difference in the C_{eff} magnitude obtained using the random sequence and the correlated stream.

Table 1 - Modules Characteristics and Thier C_{eff} Values from PowerMill.

Module Name	# of input bits	# of output bits	Style	Random (pf)	Correlated (pf)
RCA_{16}	33	17	CMOS	12.8	3.3
AM_{15x15}	30	30	CPL	119.4	33.5
CSA_{16}	33	17	CPL	3.9	0.96
BM_{6x6}	12	12	CPL	26.1	18.3
$COMP_{16}$	32	2	CMOS	1.2	0.30
$DIV_{8\text{-}by\text{-}4}$	12	5	CPL	15.8	12.2

We compare the accuracy of the MLR approach to the LR technique [1] in Table 2. The error measure used is the relative error on the average defined as:

$$error = \left| \frac{PowerMill - estimate}{PowerMill} \right| \times 100 \qquad (7)$$

Table 2 - Summary of Power Estimation Results.

Module Name	LR (% error)		MLR (% error)	
	Random	Correlated	Random	Correlated
RCA_{16}	1.2%	80%	1.6%	11.2%
AM_{15x15}	0.0%	146%	0.9%	7%
CSA_{16}	1.7%	61%	1.2%	2.3%
BM_{6x6}	0.0%	35%	1.6%	14%
$COMP_{16}$	0.8%	130%	2.6%	34%
$DIV_{8\text{-}by\text{-}4}$	0.1%	13.2%	0.4%	13%
Average	0.63%	77.3%	1.4%	13.6%

Both LR and MLR shows good performance when the input data is random (LR is slightly better). For correlated data, MLR is superior to LR and has an average error within 15% from PowerMill. Both our approach and the one in [1] produced large errors per cycle and so neither is suitable for cycle-based RTL power estimation.

Both techniques performed *power estimation* very quickly (less than 3.0 CPU seconds for the largest circuit on a Sparc 20 station with 240M memory). LR is slightly faster since the coefficients of only one equation has to be looked up during power estimation (as compared to selecting one out of N equations and then looking up its coefficients for MLR). In [1], the amount of time needed to *characterize* a module is not given. Our approach requires about 8000 PowerMill cycles for the largest circuit. Of course, much longer time would have been required if the Monte-Carlo simulation technique was not used (See Figure 3). If accurate gate-level power simulation tools are used for characterization, this time can be even made much smaller. However, not all circuits are gate-level representable, and so in such cases transistor-level simulation is the only alternative.

6 Conclusions

In this paper, a new procedure to characterize the effective capacitance in data-path components was described. The new technique is based on piece-wise linear regression modelling using the number of transitional input bits as the split criteria. We also employ a Monte-Carlo simulation scheme to cut-down the *characterization time*. Our approach significantly improves the estimation accuracy as compared to [1] with similar *estimation time*. Future work includes improving accuracy and further reducing the characterization time. The latter can be done, for example, by creating equivalent sets of patterns with "nearby" *IN* values to reduce the number of simulations and linear regressions that have to be performed.

References

[1] L. Benini, et. al., "Regression Models for Behavioral Power Estimation," *Sixth International Workshop on Power and Timing Modelling, Optimization, and Simulation*, Sept. 1996.

[2] D. Singh, et.al., "Power Conscious CAD Tools and Methodologies: A Perspective," *Proc. of the IEEE*, pp. 570-594, April 1995.

[3] A. Deng, "Power Analysis for CMOS/BiCMOS Circuits," 1994 *Int'l Workshop on Low Power Design*, pp. 3-8, 1994.

[4] P. Landman and J. Rabaey, "Black-Box Capacitance Models for Architectural Power Analysis," *International Workshop on Low Power Design*, pp. 165-170, April 1994.

[5] A. Raghunathan, et. al., "Register-Transfer Level Estimation Techniques for Switching Activity and Power Consumption," ICCAD, Nov. 1996

[6] H. Mehta, et. al., "Energy Characterization Based on Clustering," *33rd Design Automation Conference*, pp. 702-707, 1996.

[7] R. Burch, et. al., "A Monte Carlo Approach for Power Estimation," *IEEE Transactions on VLSI Systems*, Vol. 1, No. 1, pp. 63-71, March. 1993.