Power Estimation Techniques for FPGAs

Jason H. Anderson, *Student Member, IEEE,* and Farid N. Najm, *Fellow, IEEE*

***Abstract*—The dynamic power consumed by a digital CMOS circuit is directly proportional to both switching activity and interconnect capacitance. In this paper, we consider early prediction of net activity and interconnect capacitance in field-programmable gate array (FPGA) designs. We develop empirical prediction models for these parameters, suitable for use in power-aware layout synthesis, early power estimation/planning, and other applications. We examine how switching activity on a net changes when delays are zero (zero delay activity) versus when logic delays are considered (logic delay activity) versus when both logic and routing delays are considered (routed delay activity). We then describe a novel approach for prelayout activity prediction that estimates a net's routed delay activity using only zero or logic delay activity values, along with structural and functional circuit properties. For capacitance prediction, we show that prediction accuracy is improved by considering aspects of the FPGA interconnect architecture in addition to generic parameters, such as net fanout and bounding box perimeter length. We also demonstrate that there is an inherent variability (noise) in the switching activity and capacitance of nets that limits the accuracy attainable in prediction. Experimental results show the proposed prediction models work well given the noise limitations.**

***Index Terms*—Capacitance, estimation, field-programmable gate arrays (FPGAs), modeling, power, switching activity.**

I. Introduction

FAST time-to-market, steadily decreasing cost, and improving performance continue to make field-programmable gate arrays (FPGAs) an attractive technology for digital circuit implementation. The programmability of FPGAs implies that more transistors are needed to implement a given logic circuit in comparison with custom ASIC technologies. This leads to higher power consumption per logic gate [1] and consequently, FPGA power dissipation is fast becoming a "first class" design consideration, along with the traditional objectives of circuit performance and area efficiency. In fact, power has been cited as a limiting factor in the ability of FPGAs to continue to replace ASICs [2]. Today's largest FPGAs implement complex systems with millions of gates that can consume several watts of power [3]. Efficient power-aware design for such systems requires estimation tools that gauge power dissipation early in the design flow. Such tools allow design tradeoffs to be considered at a high level of abstraction, reducing design effort and cost.

Several analyses of FPGA power consumption have appeared in the literature [1], [3], [4]. These works have shown that power dissipation in FPGA devices is predominantly in the programmable interconnection network. In the Xilinx Virtex-II family, for example, it was reported that between 50%–70% of total power is dissipated in the interconnection network, with the remainder being dissipated in the clocking, logic, and I/O blocks [3]. The reason for the dominance of interconnect in FPGA power consumption lies in the composition of the interconnect structures, which consist of prefabricated wire segments of various lengths, with used and unused routing switches attached to each wire segment. Wire lengths in FPGAs are generally longer than in ASICs due to the silicon area consumed by SRAM configuration cells and other configuration circuitry.

The majority of power dissipation in today's FPGAs is dynamic power dissipation [3], due to the charging and discharging of parasitic capacitance, as characterized by

$$P_{\text{avg}} = \frac{1}{2} \sum_{n \,\epsilon\, \text{nets}} C_n \cdot f_n \cdot V^2 \tag{1}$$

where P_{avg} represents average power consumption, C_n is the capacitance of a net n, V is the voltage supply, and f_n is the average toggle rate (switching activity) of net n. Estimating power through (1) requires two parameters for each net: the net's switching activity and its capacitance.

We can conceive of several different views of switching activity, depending on how circuit delays are accounted for. First, activity values can be computed assuming logic and routing delays are zero (*zero delay activity*). Second, activity values can be computed considering logic delays, but not routing delays (*logic delay activity*). Third, activity values can be computed considering complete logic and routing delays (*routed delay activity*). Various approaches to computing switching activity have been proposed in the literature, and they can generally be classified as either simulation-based approaches or as probabilistic approaches [5], [6].

When delays are considered, switching activity normally increases due to the introduction of *glitches*, which are spurious logic transitions on a net caused by unequal path delays to the net's driving gate. As transitions on gate inputs occur at different times, the net experiences multiple transitions before settling to its final value. The extra activity due to glitching consumes dynamic power, and previous work has suggested that 20%–70% of total power dissipation in ASICs can be due to glitches [7].

An understanding of how switching activity changes when delays are considered is important for several reasons. First, since FPGA power dissipation is dominated by interconnect, the consequences of glitching on total power consumption may be more severe in FPGAs versus ASICs. In addition, due to the

Manuscript received September 8, 2003; revised December 31, 2003. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada.

The authors are with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4, Canada (e-mail: janders@eecg.toronto.edu; f.najm@utoronto.ca).

Digital Object Identifier 10.1109/TVLSI.2004.831478

presence of programmable switches in the interconnection network, path delays in FPGAs are generally dominated by interconnect rather than logic delays, suggesting that the severity of glitching could conceivably be greater in FPGAs than in ASICs. Another reason to study switching activity is that low-power synthesis techniques may perform optimizations on the basis of zero delay switching activity data [8], [9], with the assumption that such data correlates well with routed delay activity data. It is unknown whether this assumption is valid for FPGA technology.

In addition to switching activity, (1) requires the capacitance of each net. Early capacitance prediction for FPGAs is not well studied and the prefabricated, programmable nature of FPGA interconnect makes the capacitance prediction problem for FPGAs significantly different from the associated problem in ASICs. The dominant role of interconnect in total FPGA power consumption implies that characterization and management of net capacitance is a crucial part of a power-aware FPGA CAD flow.

In this paper, we focus on estimating the power consumed by FPGA interconnect and specifically, we study two separate problems in FPGA power estimation: 1) switching activity prediction and 2) interconnect capacitance prediction. We propose models for predicting these parameters prior to routing completion, using the Xilinx Virtex-II PRO commercial FPGA [10] as our investigation vehicle. We envision that the proposed models could be applied in variety of scenarios, such as low-power synthesis systems, power-aware layout synthesis, and early power prediction, when accurate routing data is incomplete or unavailable. The paper is organized as follows. In Section II, we provide a background on Virtex-II PRO and give an overview of our prediction methodology. Section III considers *prelayout* switching activity prediction. To motivate our work, we study switching activity and examine whether zero delay activity values can be used reliably as estimates of routed delay activity. To our knowledge, this work represents the first published study of activity in FPGA technology. We then present our activity prediction model which estimates the routed delay activity of a net using the net's zero or logic delay activity, as well as functional and structural properties of a circuit. Section IV deals with interconnect capacitance prediction at the *placement* stage. One of the main results here is that capacitance is not well approximated by generic parameters, such as a net's bounding box half-perimeter. Prediction accuracy is improved significantly when architectural aspects of the FPGA interconnect are considered. Another important contribution of this work is the observation of significant "noise" in both the capacitance and activity of nets that imposes limits on the accuracy achievable by any predictor. Conclusions are offered in Section V. A preliminary version of a portion of this work has appeared in [11] and [12].

II. Background

A. Virtex-II PRO FPGA

The Virtex-II PRO FPGA consists of a two-dimensional (2-D) array of programmable logic and interconnect resources. The primary tile in Virtex-II PRO is called a configurable logic block (CLB). A simplied view of a CLB is shown in Fig. 1. A

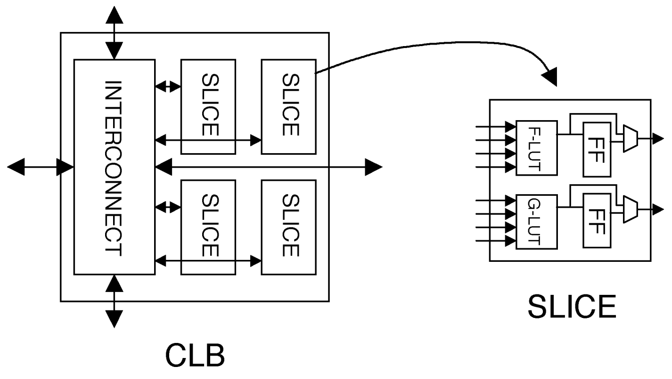

Fig. 1. Virtex-II PRO CLB and SLICE.

TABLE I
Characteristics of Benchmark Circuits

Circuit	LUTs	SLICEs	NETs
misex3	257	131	271
C3450	638	327	687
pair	464	240	637
ex1010	1112	567	1122
spla	229	116	245
pdc	609	308	625
apex2	400	204	436
alu4	500	252	514
seq	1193	605	1234
apex4	1078	548	1088
ex5p	557	286	565
cps	524	271	548
dalu	323	165	398
C2670	233	123	378

CLB's logic resources are arranged as four logic sub-blocks, called SLICEs. The main combinational logic element in Virtex-II PRO is a 4-input look-up-table (4-LUT), which is a small memory capable of implementing any logic function that requires ≤ 4 inputs. Each SLICE contains two LUTs (called the F-LUT and G-LUT), two flip-flops (FFs) as well as arithmetic and other circuitry. Nets in a Virtex-II PRO design connect SLICEs to one another and also connect SLICEs to other types of design objects, for example, I/Os.

The interconnection fabric in Virtex-II PRO is comprised of variable-length wire segments that connect to one another through programmable buffered switches. *Local*, *direct*, *double*, *hex*, and *long* interconnect resources are available. Local interconnect is internal to a CLB. Direct interconnect connects a CLB to its eight neighbors (includes diagonal neighbors). Double and hex resources are either horizontal or vertical and span two and six CLB tiles, respectively. Long resources span the entire width or height of the device.

It is worth mentioning, that although our prediction is based on Virtex-II PRO, we believe the techniques proposed are generic and can easily be adapted to other popular FPGA families. For example, the Altera Stratix FPGA [13] has logic and routing structures similar to Virtex-II. A basic tile in Stratix is called a logic array block (LAB) and it contains 10 4-LUT/FF pairs (versus eight 4-LUT/FF pairs in a Virtex-II CLB). Stratix interconnect consists of variable-length wire segments and buffered routing switches. The LAB local and direct interconnect in Stratix correspond to the CLB local

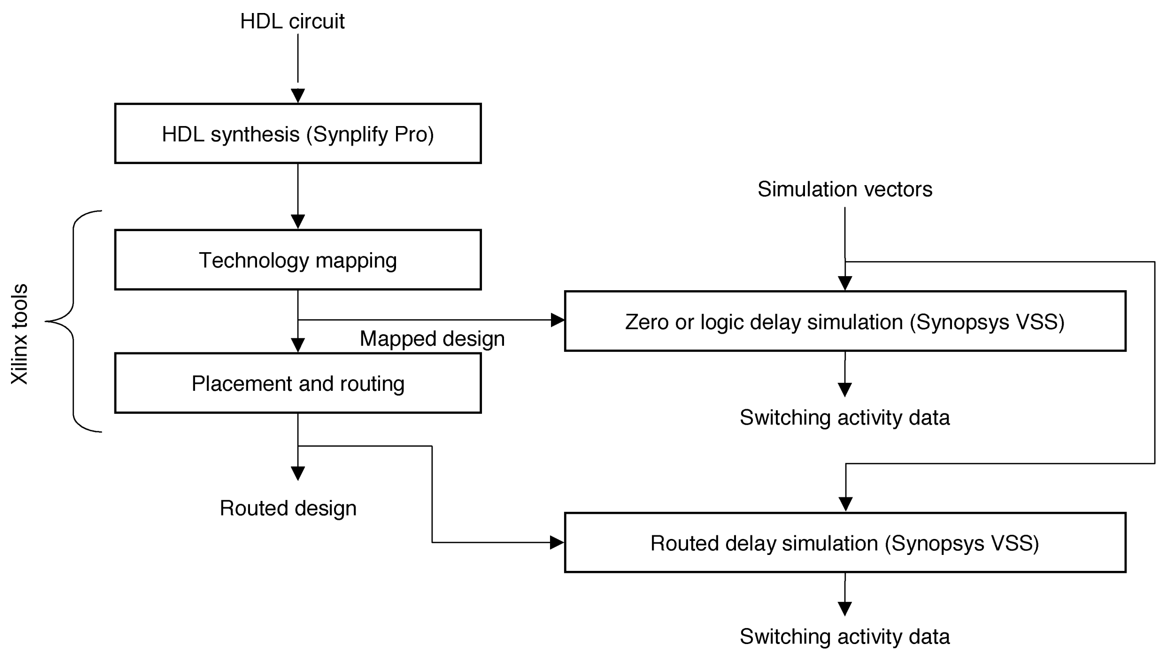

Fig. 2. CAD flow for activity analysis.

and direct interconnect in Virtex-II. Furthermore, Stratix has routing resources that span lengths of 4, 8, and 24/16 LAB tiles, which roughly resemble the double, hex, and long resources in Virtex-II. Due to such architectural similarities, we expect that the proposed techniques are widely applicable and not limited to use with Virtex-II.

B. Prediction Methodology Overview

One objective of this work is the construction of models for early prediction of a net's routed delay activity and interconnect capacitance, which we refer to as the *target parameters*. The prediction models we use are mathematical functions of a second set of parameters called *prediction parameters*, whose values are known prior to routing completion. The prediction parameters for activity and capacitance prediction are described in subsequent sections. The following set of steps convey the general methodology taken to build the prediction models.

1) A set of benchmark circuits were selected and mapped into Virtex-II PRO. Circuits were synthesized from VHDL using Synplicity's Synplify Pro tool (version 7.0) and then technology mapped, placed, and routed using Xilinx tools (version 5.2i).[1] Each circuit was mapped into the smallest FPGA device able to accommodate it. Table I provides detail on the benchmark circuits.
2) The circuits were arbitrarily divided into two sets, a *characterization* set and a *test* set. Shading in Table I differentiates the characterization circuits. We use the characterization circuits to derive models for predicting switching activity and interconnect capacitance.
3) The characterization circuits were analyzed, and prediction and target parameter values were extracted.
4) The prediction and target parameter values were fed into the GNU R statistical analysis framework [14]. Multi-variable regression analysis is employed to establish an empirical relationship between the target and prediction parameter values. Through this approach, a prediction model is tuned to a particular FPGA device and CAD flow. In practice, such model characterization would be done by an FPGA vendor to produce a prediction model incorporated into CAD tools used by engineers in the field.
5) Following the characterization step (number 4), we apply the prediction models to predict capacitance and routed delay activity values for nets in the *test* circuit set. Predicted values are verified by comparing with actual values (routed delay activity and routed interconnect capacitance).

[1]The placement and routing tools were run at the highest effort level, without performance constraints.

III. Switching Activity Prediction

We use a simulation-based approach to gather switching activity data. The CAD flow employed is shown in Fig. 2. The Synopsys VHDL System Simulator (VSS) is used for simulation. VSS has built-in capabilities for capturing the number of logic transitions on each net during a simulation, as well as the proportion of time each net spends in the high- and low-logic states. Simulation with zero or logic delays can be done after the technology mapping step. Simulation with routed delays must be done after placement and routing. In all cases, the VHDL simulation netlist was generated using the Xilinx tools, *ngdanno* and *ngd2vhdl*. The netlist is comprised of interconnected physical primitives which correspond to hardware resources in the FPGA, such as 4-LUTs, FFs, and multiplexers. For the delay-based simulations, a standard delay format (SDF) file, generated by the Xilinx annotation tool, is provided to VSS.

Circuits are simulated using 10 000 randomly chosen input vectors. Two different vector sets were generated for each circuit: one representing high input activity and a second repre-

TABLE II
EFFECT OF GLITCHING ON SWITCHING ACTIVITY

	High activity vector set		**Low activity vector set**		**High activity vector set**	
Circuit	% increase in transitions for logic dly sim vs. zero dly sim	% increase in transitions for routed dly sim vs. zero dly sim	% increase in transitions for logic dly sim vs. zero dly sim	% increase in transitions for routed dly sim vs. zero dly sim	Zero delay net activity mean error (dev)	Logic delay net activity mean error (dev)
misex3	24.3	52.3	12.5	29	38.7 (22.6)	21.6 (15.8)
C3540	20.6	125.2	15.2	93.8	46.3 (24.3)	18.7 (13.7)
pair	15.8	49	9.5	10.9	30.7 (22.8)	14.3 (15.8)
ex1010	60.3	114.9	33.9	60.9	59.7 (19.4)	41.7 (17.3)
spla	21.5	27.6	11.3	15.1	26.2 (19)	19.2 (14.2)
pdc	38.7	75.2	19.8	43.2	45.4 (19.5)	30.1 (15.9)
apex2	17.8	39.1	9.3	21.8	32.2 (19.5)	17.7 (14.1)
alu4	33.4	54.8	17.1	28.8	36.5 (19.3)	25.1 (14.5)
seq	23.5	44.7	11.6	23	35.1 (18.0)	22.4 (14.5)
apex4	40.6	96.3	22.6	51.9	40.0 (19.5)	27.8 (15.5)
ex5p	52.7	131.5	35.3	97.5	45.0 (14.5)	29.5 (11.7)
cps	32.9	50.2	17.4	27.6	34.9 (17.1)	26.6 (14.4)
dalu	24.8	48.3	15.3	31	44.1 (19.4)	24.9 (15.0)
C2670	27.3	51.8	17.1	36.1	43.1 (18.7)	25.3 (15.6)

senting low input activity. In the high (low) activity vector set, the probability of an individual input toggling between successive vectors is 50% (25%).

A. Activity Analysis

Using the flow of Fig. 2, we investigated how switching activity changes when delays are considered. Columns 2–5 of Table II compare the total number transitions in the logic and routed delay simulations of each circuit with the number of transitions in the zero delay simulation. Columns 2 and 3 (4 and 5) of the table present data for the high (low) activity vector set simulations. Each table entry represents, for a given circuit, the percentage increase in the number of transitions in the circuit's logic or routed delay simulation versus the circuit's zero delay simulation. Note that partial glitches were filtered out of this analysis and do not register as transitions.[2]

From Table II, we see that there is a significant increase in activity when delays are considered. For the high activity vector set, when logic delays are used, the percentage increase in transition count versus the zero delay simulation ranges from 16% to 60%. When routing delays are used, overall circuit delay increases and becomes more variable, leading to more glitching and higher activity. In the routed delay simulations, the increase in transition count versus the zero delay simulations ranges from 28% to 131%. Comparing the data for the two vector sets, we observe that the increases in activity are somewhat less drastic when the low activity vector set is used. Specifically, the activity increases are about 1/2 to 2/3 of that seen with the high activity vector set. In the low activity vector set, fewer inputs switch simultaneously between successive vectors, which reduces the potential for logic transitions on multiple (unequal delay) paths to a net, leading to reduced glitching.

To investigate whether the increase in activity due to glitching is distributed uniformly amongst the nets of a circuit, we view the zero and logic delay transition count for a net as estimates of the net's routed delay transition count. We then measure the absolute percentage error in the estimates. For example, the error in a net n's zero delay activity estimate is

$$\mathrm{EZ}_n = 100 \cdot \frac{|\mathrm{TR}_{z,n} - \mathrm{TR}_{r,n}|}{\mathrm{TR}_{r,n}} \tag{2}$$

where $\mathrm{TR}_{r,n}$ and $\mathrm{TR}_{z,n}$ represent the number of transitions on net n in the routed and zero delay simulations, respectively.

Error analysis results (for the high activity vector set) are given in columns 6 and 7 of Table II, which shows the average and standard deviation of error for each circuit. Note that for this analysis, we ignored the error on nets that transitioned on fewer than 3% of the simulation vectors as we did not consider the error data for such low activity nets to be statistically significant. In Table II, we see that the mean error in the zero delay activity values falls in the 26%–60% range. The mean error in logic delay activity ranges from 14% to 42%. We also observe that coupled with these large mean errors are large error deviations, ranging from 14% to 24% for the zero delay case and 12%–17% for the logic delay case. This leads us to conclude that zero delay and logic delay activity values do not necessarily correlate strongly with routed delay activity values. Although not shown here, we also computed error data for the low activity vector set simulations. We observed smaller errors for this vector set, with the mean and deviation of error for each circuit being about 1/2 to 2/3 of that observed for the high activity vector set.

The results in Table II show that activity can change considerably when delays are brought into the picture and motivates the need for early prediction of a net's routed delay activity. Prior to presenting our prediction approach, we analyze the noise in the prediction problem.

B. Noise in Switching Activity

In an effort to understand the difficulty of generating accurate prelayout activity estimates, we study the noise in routed delay activity values by performing an activity analysis on the circuit shown in Fig. 3. The circuit is highly regular from the structural

[2]A partial glitch on a net is a glitch of short duration, shorter than the logic delay of the net's driving gate.

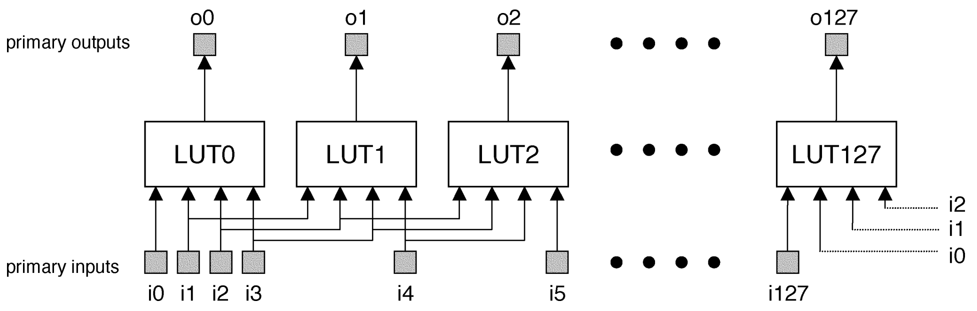

Fig. 3. Circuit with regularity.

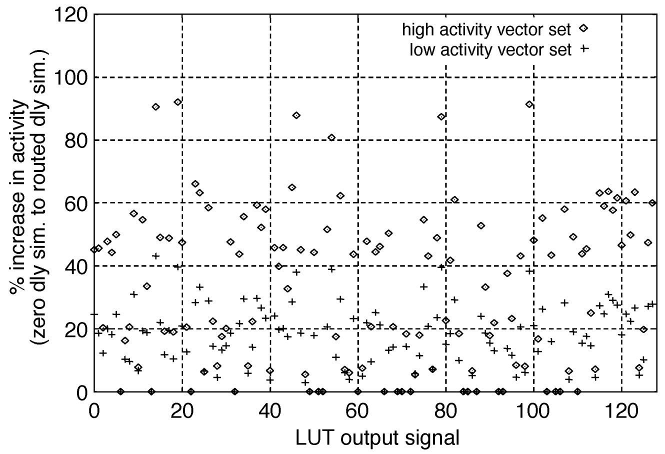

Fig. 4. Activity change in regular circuit.

and functionality viewpoint and consists of 128 inputs driving 128 4-input LUTs, which in turn drive 128 outputs. Each LUT in the circuit is programmed to implement a 4-input logical AND function.

We mapped the circuit in Fig. 3 into Virtex-II PRO and simulated it with both the high and low activity vector sets. We then examined the percentage increase in activity on the LUT output signals in the routed delay simulation versus the zero delay simulation. Note that the circuit's regularity implies that variability in the activity change on the LUT output signals is largely a result of variable path delays that are known only after layout is complete. The results of the analysis are shown in Fig. 4. The figure shows the percentage increase in activity on each LUT output signal for both simulation vector sets (128 data points are shown for each vector set—one for each LUT output signal). Observe that despite the circuit's regularity, the variability in the activity increase on the LUT output nets is considerable due to the wide variety of routing resources (and delay paths) in the FPGA routing fabric and the different delays associated with the four input-to-output paths through a LUT. [3] For the low activity vector set, the activity increase for most nets is in the range of 0% to 40%; for the high activity vector set, the increase ranges from 0% to 90%.

Real circuits are likely to be much less regular than the circuit of Fig. 3, and we therefore conclude that it will be difficult to achieve a high degree of accuracy in activity prediction at the prelayout stage. Nevertheless, in Section III-C, we offer a prediction approach which produces activity values that, in comparison with zero or logic delay activity values, are superior estimates of routed delay activity.

C. Prediction Model

Before describing our activity prediction approach, we review some terminology related to the graph representation of digital circuits. The combinational part of a logic circuit can be represented as a *Boolean network*, which is a directed acyclic graph (DAG) in which each node represents a single-output logic function and edges between nodes represent input–output dependencies between the corresponding logic functions. A primary input node is a node with an in-degree of 0; a primary output node has an out-degree of 0. Our prediction approach accepts a technology mapped (Virtex-II PRO) FPGA circuit as input. At this level of abstraction, internal DAG nodes correspond to the LUTs and other logic elements in the target FPGA device. For a node y in a circuit DAG, let $\mathrm{inputs}(y)$ represent the set of nodes that are fanins of y. A node w is said to be a predecessor of a node y if there exists a directed path in the circuit DAG from w to y. The *depth* of a node y, D_y, is the length of the longest path from any primary input to y. In this section, we refer to a node and the net driven by the node interchangeably; for example, a node y drives net y.

In FPGA technology, path depth (number of LUTs) is frequently used as a predictor of path delay at the prelayout stage [15], [16]. The reason for this is that, unlike in ASIC technologies such as standard cell, the logic blocks in FPGAs are uniform and have equal drive capability. Furthermore, the programmable routing switches in an FPGA's routing fabric are typically buffered, making connection delay relatively independent of fanout. Consequently, without access to more accurate delay information extracted from physical layout, depth is viewed as a reasonable estimate of delay. We leverage this FPGA-specific property in our activity prediction approach, which incorporates delay estimation into a simple model of net glitching severity.

Our approach to activity prediction is analogous to the generate and propagate notion that defines how carry signal values are assigned in arithmetic circuits. In such circuits, the carry value for a particular bit may either be *generated* by the bit, or it may be *propagated* from a lower-order bit. For activity prediction, consider a node y with logic function $y = f(x_1, x_2, \ldots, x_n)$. Similar to carry signal operation, glitches on y's output may come from two sources: they may

[3]LUT input pins are logically equivalent. The selection of a LUT input pin for a particular LUT fanin signal is made by the router.

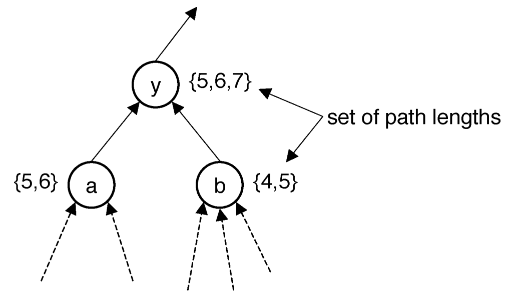

Fig. 5. Finding the set of path lengths for y.

be propagated from one of y's inputs, $x_1, x_2, \ldots, x_n$, or they may be generated by y itself. We define a prediction function that quantifies the severity of glitching on y's output as follows:

$$\mathrm{PR}_y = f(\mathrm{GEN}_y, \mathrm{PROP}_y, D_y) \tag{3}$$

where f is a function defined below and GEN_y and PROP_y represent the amount of glitching generated by y and the amount of glitching propagated from y's inputs, respectively. PR_y is the predicted percentage change in the activity of net y due to glitching. The depth term (D_y) of (3) is included to reflect the intuition that glitching severity typically increases with combinational depth. All things being equal, we expect that a node with shallow depth will experience less glitching than a deep node. Note that we compute prediction values for the nodes of a circuit in a specific order, from primary inputs to primary outputs.

Prior to defining the generate term of (3), we introduce a new parameter. Let PL_y represent the *set* of different path lengths from a primary input to node y. This parameter can be computed in linear time by an input-to-output DAG traversal that maintains a set of path lengths for each node. When a node is traversed, its path length set is populated by taking the union of incremented path lengths of each of its immediate fanin nodes. More formally

$$\mathrm{PL}_y = \bigcup_{x_i \,\epsilon\, \mathrm{inputs}(y)} \{p+1 | p \,\epsilon\, \mathrm{PL}_{x_i}\}. \tag{4}$$

Observe that a given node may have a larger set of path lengths than any of its immediate fanins. Consider the example shown in Fig. 5, in which a node y has two fanin nodes, a and b. The set of path lengths for each node is shown adjacent to the node. We see that node y has three path lengths, whereas its fanins have only two path lengths. Thus, we say that one path length is *introduced* at y. We define the generate term of (3) to be equal to the number of path lengths introduced at node y, defined as

$$\mathrm{GEN}_y = \min_{x_i \,\epsilon\, \mathrm{inputs}(y)} \{|\mathrm{PL}_y| - |\mathrm{PL}_{x_i}|\}. \tag{5}$$

The rationale for incorporating the number of path lengths to a node into our prediction function is that variable path lengths to a node generally correlate with variable (unequal) path delays to the node, leading to glitching at the node's output.

The propagate term of (3) borrows ideas from the concept of transition density [17] and uses the notions of *Boolean difference* and *static probability*, which we briefly review here. The Boolean difference of a function, $y = f(x_1, x_2, \ldots, x_n)$, with respect to one of its inputs, x_i, is defined as

$$\frac{\partial y}{\partial x_i} = f_{x_i} \oplus f_{\overline{x}_i} \tag{6}$$

where f_{x_i} $(f_{\overline{x}_i})$ is the Boolean function obtained by setting $x_i = 1$ $(x_i = 0)$ in $f(x_1, x_2, \ldots, x_n)$, and $\oplus$ denotes the exclusive-OR operation. When the Boolean difference function, $\partial y/\partial x_i$, is 1, a transition on x_i will cause a transition on y.

The static probability of a signal is defined to be the fraction of time that the signal is in the logic-1 state. Thus, the static probability of a Boolean difference function, $P(\partial y/\partial x_i)$, represents the probability that a transition on x_i will cause a transition on y. Clearly, the ability of glitches on an input signal, x_i, to propagate to y depends on $P(\partial y/\partial x_i)$. Furthermore, we expect that the influence of a node input on the node's output will depend partly on the input's switching activity. The propagate function is therefore defined as

$$\mathrm{PROP}_y = \frac{\sum_{x_i \,\epsilon\, \mathrm{inputs}(y)} P\left(\frac{\partial y}{\partial x_i}\right) \cdot \mathrm{TR}_{z,x_i} \cdot \mathrm{PG}_{x_i}}{\sum_{x_i \,\epsilon\, \mathrm{inputs}(y)} P\left(\frac{\partial y}{\partial x_i}\right) \cdot \mathrm{TR}_{z,x_i}}. \tag{7}$$

The $P(\partial y/\partial x_i) \cdot \mathrm{TR}_{z,x_i}$ in the numerator can be viewed as a weight quantifying the influence of glitching on x_i to glitching on y. PG_{x_i} is defined as $\mathrm{PROP}_{x_i} + \mathrm{GEN}_{x_i}$ and represents the amount of glitching on input x_i. The denominator of (7) normalizes the values computed by the propagate function so they are relatively independent of the transition counts and probabilities involved. Note that TR_z in (7) can be replaced with TR_l (the logic delay transition count) if logic delay activity data is available. In our experiments, the probability and transition data needed to compute (7) is extracted from zero or logic delay circuit simulation (see Section III-D below). However, such data need not be derived from simulation; it can be computed efficiently using probabilistic approaches, such as those described in [18]. Thus, simulation is not a requirement for the use of our prediction model.

D. Results and Discussion

Following the methodology outlined in Section II-B, we use the characterization circuits to derive a model relating the activity change on a net due to glitching to the *prediction function* (3). In this case, the prediction parameters are the PROP, GEN and D terms in (3). We began by using a linear function for f and gradually increased its complexity by adding higher order terms. We eventually settled on a quadratic function of the form

TABLE III
PREDICTION MODEL AND REGRESSION ANALYSIS DETAILS (ZERO DELAY ACTIVITY AND LOGIC DELAY ACTIVITY-BASED PREDICTION MODELS)

Parameter	Coeff	P-value	Lower 95% / Upper 95%
int	35.909	< 2E-16	31.330 / 40.488
$PROP$	37.413	< 2E-16	33.184 / 41.642
GEN	-3.653	0.129	-8.369 / 1.064
D	3.214	< 2E-16	2.866 / 3.563
$PROP^2$	-0.728	0.135	-1.683 / 0.227
GEN^2	-0.950	0.109	-2.113 / 0.212
$PROP \cdot D$	-0.484	< 2E-16	-0.593 / -0.374
$GEN \cdot D$	0.313	0.0033	0.104 / 0.523
int	48.875	< 2E-16	46.008 / 51.742
$PROP$	21.361	< 2E-16	18.014 / 24.709
GEN	-4.165	8.613E-05	-6.241 / -2.088
D	-2.276	< 2E-16	-2.630 / -1.921
$PROP^2$	0.974	6.905E-04	0.412 / 1.536
D^2	0.042	< 2E-16	0.034 / 0.050
$PROP \cdot D$	-0.452	3.1E-13	-0.573 / -0.331
$PROP \cdot GEN$	1.193	0.08697	-0.173 / 2.559

$$\begin{aligned}\mathrm{PR}_y = &\alpha \cdot \mathrm{GEN}_y + \beta \cdot \mathrm{GEN}_y^2 + \gamma \cdot \mathrm{PROP}_y \\ &+ \upsilon \cdot \mathrm{PROP}_y^2 + \nu \cdot D_y + \xi \cdot D_y^2 \\ &+ \eta \cdot \mathrm{PROP}_y \cdot \mathrm{GEN}_y + \iota \cdot \mathrm{GEN}_y \cdot D_y \\ &+ \rho \cdot \mathrm{PROP}_y \cdot D_y + \phi \end{aligned} \tag{8}$$

where α, β, γ, etc., are scalar coefficients whose values are determined through regression analysis. We considered using higher order models but found they did not offer substantially improved accuracy. Two prediction models were constructed: one that predicts routed delay activity from *zero* delay activity and one that predicts routed delay activity from *logic* delay activity.

For completeness, Table III shows the prediction model and regression details. Shading is used to differentiate the logic delay activity-based prediction model from the (separate) zero delay activity-based prediction model. Column 1 of the table lists the parameters used in each model. Observe that all of the terms of (8) are not present for each model—we used the *step* function in the *R* package to prune (8) to contain only the required, significant terms [14]. Column 2 presents the coefficient values for each parameter. Column 3 of the table gives the *P-value* for each parameter, which is a frequently-used significance metric in regression analysis. It represents, for a given parameter, the probability that the parameter's actual coefficient is zero rather than that specified in the model. Thus, low *P-values* are generally associated with high parameter significance. Column 4 of the table gives the 95% confidence bounds for each parameter's coefficient.

Following characterization, to apply our prediction model (derived from the characterization circuits), we first compute the model's value for a net in one of the test circuits. This yields a predicted percentage change in activity for the net versus the net's zero (or logic) delay activity. We use the predicted percentage change to compute an estimate of the net's routed delay activity.

We evaluate our approach numerically by computing the percentage error in predicted activity values (relative to routed delay activity values) using (2). The error data for the test circuits is shown in Table IV. The table gives the average absolute percentage error across all nets for each circuit. Column 2 of the table shows the error results for the model that predicts routed delay activity from zero delay activity; column 3 gives results for the model that predicts routed delay activity from logic delay activity. In Section III-A, we saw that logic delay activities are "closer" to routed delay activities than are zero delay activities. Table IV confirms that using logic delay activities as the basis of a prediction model yields smaller error values. The error data for each circuit in the table can be compared with the error data in columns 6 and 7 of Table II. Observe that the average error of the predicted activities is significantly less than the error of the zero or logic delay activities; the error is reduced by a factor of 2 for many of the circuits.

TABLE IV
ERROR IN PREDICTED ACTIVITY VALUES

Circuit	Predicted net activity mean error (from zero dly activity)	Predicted net activity mean error (from logic dly activity)
alu4	17.8	16
seq	23.8	19.8
apex4	21.4	20.2
ex5p	17.1	13.3
cps	19	16.7
dalu	23.9	16.4
C2670	22.9	17.8

Fig. 6(a) and (b) shows zero delay and predicted activity values versus routed delay activity values. Each point in these plots corresponds to a net in one of the test circuits. The vertical axis of Fig. 6(a) represents zero delay activity (transition count); the horizontal axis represents routed delay activity. The vertical axis of Fig. 6(b) represents predicted activity. Observe in Fig. 6(a), that the absence of glitching creates a "ceiling" in the zero delay activity values at about 5000 transitions. This ceiling is eliminated in the predicted activity values [Fig. 6(b)], which more closely resemble the routed delay activity values. Fig. 7 gives analogous plots for logic delay and (logic delay-based) predicted activity values. Both sets of predicted values exhibit a considerable "spread" about the $y = x$ line shown. This is expected and is in line with our noise analysis in Section III-B.

From Figs. 6 and 7, we see that as expected, the errors in the zero and logic delay activity values are largely one sided (under estimation), whereas the errors in the predicted activity values are balanced about the $y = x$ line. The use of zero or logic delay activity values in power estimation will lead to significant underestimates of circuit power. Conversely, since our approach under-predicts activity for some nets and over-predicts for others, we expect that the use of the predicted activity values will produce average power estimates much closer to actual average circuit power. Eliminating the one-sided bias in error is one of the key advantages of our prediction method, making it attractive for use in applications such as early power estimation.

To investigate the dependence of the prediction results on the division of benchmarks into the characterization and test sets, we created an *alternate* benchmark division by swapping three of the seven circuits in each set. We reconstructed the prediction models using the new characterization set and then applied

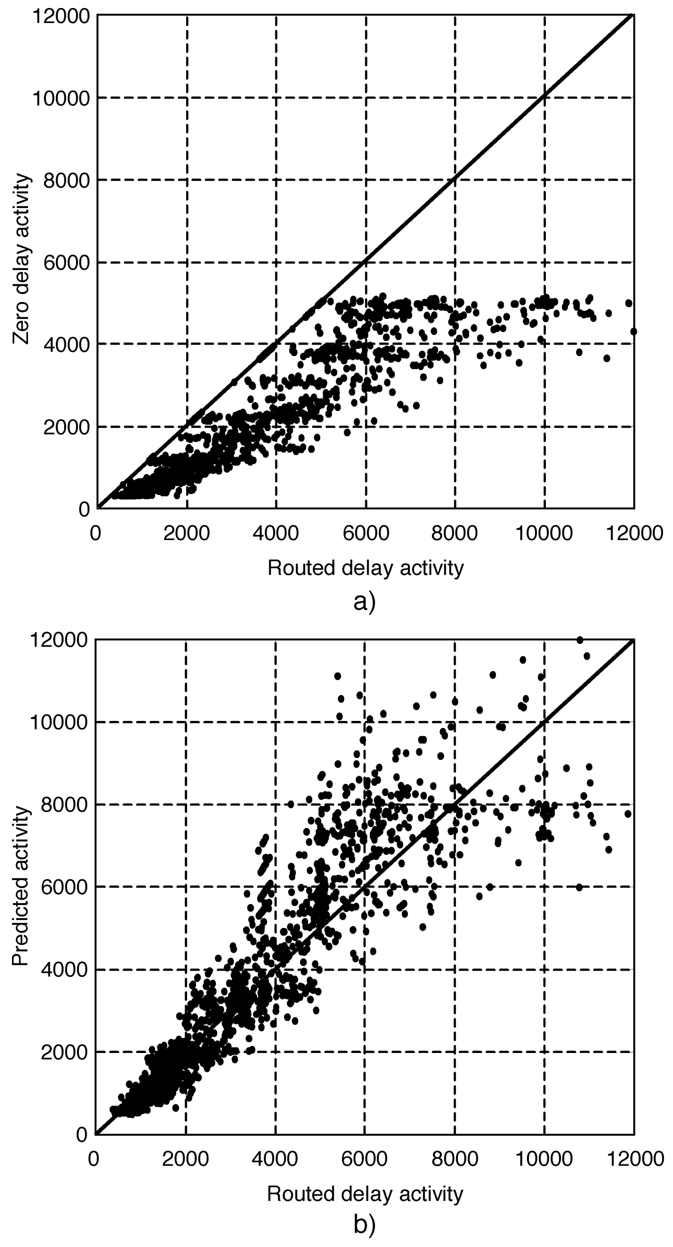

Fig. 6. Zero delay activity and predicted activity versus routed delay activity. (a) Zero versus routed delay activity. (b) Predicted versus routed delay activity.

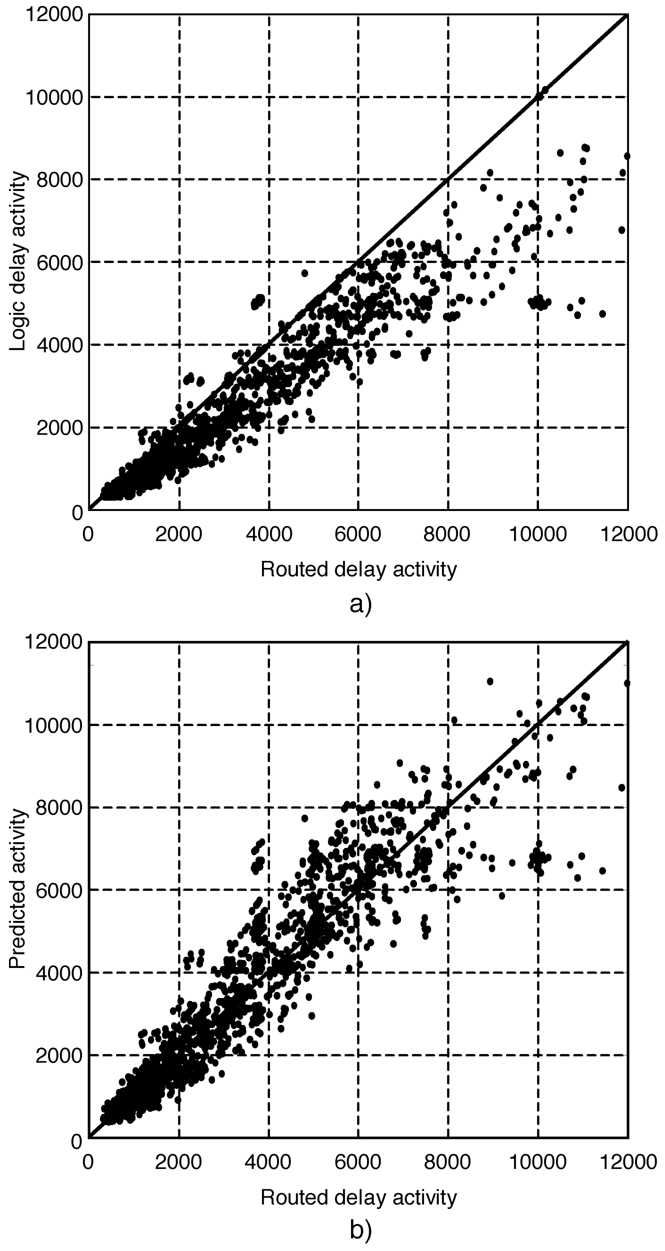

Fig. 7. Logic delay activity and predicted activity versus routed delay activity. (a) Logic versus routed delay activity. (b) Predicted versus routed delay activity.

the models to predict activity values for nets in the new test set. The average absolute percentage error in the predicted activity values is shown in Table V. The columns of the table are analogous to those of Table IV. As with the original benchmark division, the average errors of the predicted activity values are considerably less than the errors of zero or logic delay activities (compare with columns 6 and 7 of Table II). Shading in Table V is used to show the circuits that are common between the original and alternate test circuit sets. The error results for these circuits can be compared with the results in Table IV (for the original benchmark division). Observe that, despite the use of different characterization sets, the prediction errors for these circuits are fairly similar. This implies that our prediction model is not tied to a specific choice of characterization and test set. We expect that the dependence of the prediction model on the benchmark division can be reduced further through the use of a larger characterization set, as would be available to commercial FPGA vendors.

TABLE V
ERROR IN PREDICTED ACTIVITY VALUES (ALTERNATE CHARACTERIZATION/TEST BENCHMARK DIVISION)

Circuit	Predicted net activity mean error (from zero dly activity)	Predicted net activity mean error (from logic dly activity)
pdc	19.5	14.3
apex2	26.0	15.9
ex1010	22.7	28.5
ex5p	16.1	15.6
cps	18.1	15.1
dalu	23.7	17.0
C2670	22.0	17.8

The prediction results presented above were based on the simulation data for the high activity vector set. We also generated prediction models for the low activity vector set data and found similar results, though there is less severe glitching associated with lower input activity. Thus, we expect that our prediction method can be applied effectively for a range of input switching activities. A direction for future work is to augment the prediction approach to automatically account for various amounts of primary input switching activity.

IV. Interconnect Capacitance Prediction

We now move on to interconnect capacitance prediction at the placement stage. We begin with a brief review of related work.

A. Related Work

Several works have considered capacitance estimation in the context of power-aware FPGA CAD tools. At the technology mapping level (prelayout), net capacitance has been estimated using a linear function of fanout [8], [9]. Previous placement-based capacitance estimates have appeared in [19] and [20]. At this level, the approach taken has been to use a combination of a net's bounding box half-perimeter and its fanout to estimate its routed capacitance. These prior works use generic, non-architecture-specific parameters to predict capacitance.

A problem related to capacitance estimation is that of FPGA delay estimation, which is well studied in the literature. The problems differ from each other in that delay estimates are needed for individual driver/load connections whereas capacitance estimates are needed for entire (multi-fanout) nets. In [21], a delay estimation model is constructed during placement by executing a pre-routing step in which "dummy" routes having known x and y distances are made. Following this, a table that relates delay to distance is constructed; table values are used as delay estimates during placement. The delay estimation model development approach used in [22] and [23] is similar to our own. Routed designs are analyzed and connection delays are correlated with placement parameters, producing an empirically-derived estimation model. In [24], characteristics of a target FPGA's interconnect architecture are used to predict delay within a partitioning-based placement system. In this case, the FPGA interconnect is hierarchical and the placer's partitioning levels are chosen to match the underlying FPGA interconnect hierarchy. As such, the placer has knowledge of the interconnect resources likely to be used in the routing of nets that are cut (and uncut) at a given partitioning level. Like [24], our capacitance estimator also considers architecture-specific criteria to improve estimation accuracy.

B. Noise in Interconnect Capacitance

To gauge the inherent noise in capacitance estimation, we take an approach similar to that used in [25]. Specifically, we place each benchmark circuit (using Xilinx tools) and generate a placed netlist. We then create a copy of the placed netlist and modify the copy, reversing the *order of the nets* but leaving all other aspects of the design intact (including the placement).[4] The order of the nets in the netlist is arbitrary and generally

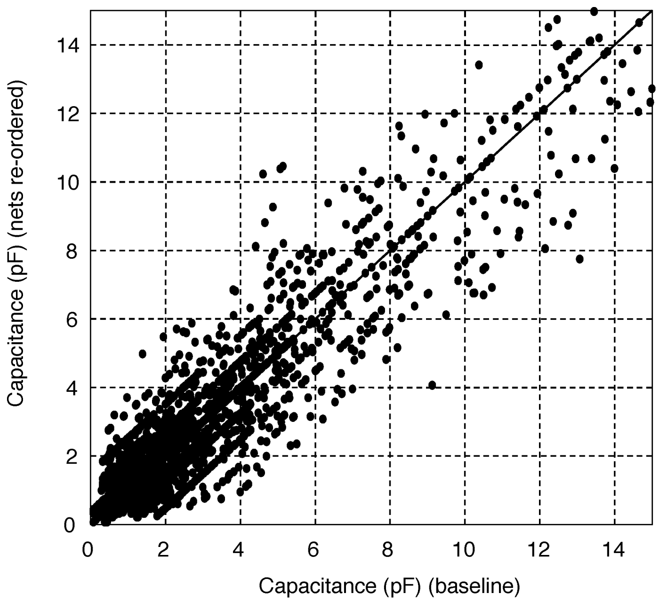

Fig. 8. Noise in interconnect capacitance.

not under user control. The original placed netlist and the modified netlist for each design are then routed to produce *baseline* and *alternate* routing solutions, respectively. Interconnect capacitance values for nets are ascertained by running XPower [26], the Xilinx power estimation tool, on the routed circuits.[5] XPower produces a log file containing the capacitance of each net in femtofarads (fF). The capacitance values for each net in the two routing solutions can be compared to assess routing variability (since both routing solutions have the same placement). Differences in the net capacitance between the baseline and alternate routing represent noise that one cannot correct or account for in estimation. Note, however, that differences in the two-routing solutions for a design generally do not represent problems with the routing tool. The tool aims to minimize total routing resource usage, which involves tradeoffs between the FPGA resources allocated to each net; such tradeoffs may be resolved arbitrarily in some cases.

Fig. 8 shows the results of the noise analysis. Each point in the figure represents a net in one of the benchmark circuits. The horizontal axis represents net capacitance in the baseline routing solution; the vertical axis represents net capacitance in the alternate routing solution. Ideally, in the absence of variability, all points would lie on the line shown $(y = x)$. However, Fig. 8 shows there to be substantial noise in net capacitance. Notice that the results in Fig. 8 illustrate that one routing solution is unlikely superior to the other: the symmetry in spread about the $y = x$ line suggests that the number of nets for which net capacitance increased in the alternate routing solution is approximately equal to the number of nets for which capacitance decreased. This assertion is also true at the individual circuit level as evidenced by the data in column 3 of Table VI. Column 3 shows, for each circuit, the number of nets for which capacitance increased and decreased (in parentheses) in the alternate

[4]Netlist modifications were made by reversing the order of instances in each placed design's Xilinx Design Language (XDL) ASCII representation.

[5]FPGA vendors do not provide users with a transistor-level or layout-level FPGA representation. Commercial capacitance extraction tools, commonly used in custom ASIC design, cannot be applied by FPGA users to determine capacitance values.

TABLE VI
NOISE IN INDIVIDUAL CIRCUITS

Circuit	Avg. absolute % change in cap.	# of nets with cap increase (decrease)
misex3	14.9	27 (38)
C3540	18.2	115 (117)
pair	19.1	112 (103)
ex1010	23.7	234 (255)
spla	22.8	33 (28)
pdc	23.4	146 (112)
apex2	23.7	77 (95)
alu4	20.0	97 (86)
seq	23.1	246 (259)
apex4	23.7	226 (235)
ex5p	24.0	124 (128)
cps	23.1	88 (78)
dalu	23.0	72 (65)
C2670	22.8	49 (65)
Average:	21.8	

routing solution versus baseline. The number of increases and decreases are roughly equal for each circuit.

We computed the absolute value of the percentage change in capacitance for each net in the alternate routing versus the baseline. Column 2 of Table VI shows the average absolute change for each circuit. The average change across all circuits is 22%. This represents a statistical lower bound on the error in capacitance estimation; estimation accuracy cannot be improved beyond this noise floor error limit.

C. Prediction Model

Having analyzed the noise in capacitance, we now present our prediction model. In comparison with prediction at the prelayout stage, a larger number of prediction parameters are available for prediction at the placement stage (e.g., physical design data). Consequently, we take a slightly different approach to capacitance prediction as compared with that taken for activity prediction. We first define the prediction parameters used in our model and following this, we explain the rationale of why the chosen parameters may correlate with net interconnect capacitance. Section IV-D presents experimental results showing which of the parameters are best for use in a capacitance prediction model.

CAD applications such as power-aware placement and early power planning require that capacitance estimates be produced quickly as they are typically needed within the inner loop of design optimization. Consequently, we focus on parameters with low computational requirements. Considering a net n, the following are known at the placement stage:

FO_n	fanout of net n;
BB_n	half-perimeter of net n's bounding box, as measured in CLB tiles;
$\mathrm{XS}_n, \mathrm{YS}_n$	span of net n in the x- and y-dimensions, respectively;
NT_n	number of CLB (or I/O) tiles in which net n has at least 1 pin;
$\mathrm{X6}_n, \mathrm{Y6}_n$	defined as $\mathrm{XS}_n \bmod 6$ and $\mathrm{YS}_n \bmod 6$, respectively;
$\mathrm{FP}_n, \mathrm{GP}_n$	number of load pins on net n that are F-LUT and G-LUT inputs, respectively;
CG_n	average estimated routing congestion in net n's bounding box.

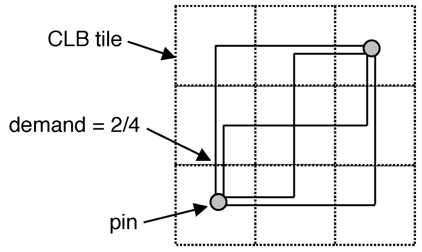

Fig. 9. Routing congestion estimation.

The fanout and bounding box of net n are generic parameters, frequently used to predict capacitance in the ASIC domain. Breaking the bounding box into its x and y spans through XS_n and YS_n allows us to evaluate whether there is a capacitance bias associated with the use of horizontal versus vertical routing resources.

In contrast to the fanout and distance terms, parameters NT_n, $\mathrm{X6}_n$, $\mathrm{Y6}_n$, FP_n, and GP_n are specific to the Virtex-II PRO architecture. As mentioned in Section II-A, the FPGA contains an array of CLB tiles. Most of the interconnect resources connect CLB tiles to one another, with the exception of the local interconnect that is internal to a CLB. A CLB contains four SLICEs (eight LUTs/FFs) and, therefore, a net n may have multiple pins placed in a single CLB. In such cases, some of the net's routing *between* CLBs may be shared by the net's pins *within* in a single CLB. The sharing of routing resources amongst pins may influence net capacitance and the NT_n term aims to account for this possibility.

An important routing resource in the FPGA interconnect is the hex-length wires spanning six CLB tiles. We expect that long nets may be routed using a sequence of hex lines, with the "left over" distance being composed of the shorter double-length, direct or local resources. Hex-length resources likely have more capacitance than shorter resources. $\mathrm{X6}_n$ and $\mathrm{Y6}_n$ represent the left over distance in the x and y dimensions, respectively, and roughly correspond to the number of short resources needed for a net. Similarly, we expect that the different types of pins on logic and I/O blocks may have different capacitance values associated with them. The FP_n and GP_n parameters allow the F and G-LUT input pins (see Section II-A) to be differentiated from other types of pins.

Routing congestion may lead to nets with long circuitous paths and excess capacitance. We estimate the congestion for a net n, CG_n, using a probabilistic method similar to that described in [27], chosen for its simplicity and computational efficiency. We summarize the approach here; the interested reader is referred to [27] for details. Nets are first converted into a set of two-pin connections by finding their minimum spanning tree using Prim's algorithm. The routing demand of a two-pin connection is computed probabilistically, considering its potential routing topologies. An example for a two-pin connection with a 3-by-3 CLB tile bounding box is shown in Fig. 9. As illustrated, only routing topologies that have *at most* two jogs are included. There are four possible route options for the connection. Generally, the number of route options for a connection having a bounding box with c columns and r rows is

$$N_{\mathrm{ropt}} = \begin{cases} c + r - 2, & \text{for } c > 1, r > 1 \\ 1, & \text{otherwise.} \end{cases} \tag{9}$$

Similarly, the number of a connection's route options that cross a *specific* CLB tile edge can also be expressed analytically (details are omitted here due to limited space). Dividing the number of a connection's route options that cross a specific CLB tile edge by the total number of route options for the connection yields the probability that the connection's route will traverse the CLB tile edge. This probability can be viewed as the *demand* exerted by the connection on a tile edge (see Fig. 9). The routing demands contributed by each two-pin connection in each net are tallied to produce a total routing demand on each CLB tile. The CG_n term represents the *average* routing demand across all CLB tile edges within net n's bounding box.

The capacitance of nets in the characterization circuit set were fit to a mathematical function of the parameters as described. The result is a mathematical model that may be applied to predict capacitance values of nets in the test circuit set. We developed separate estimation models for high-fanout nets (>10 loads) and low-fanout nets (≤ 10 loads) and apply each model accordingly in our experimental study (Section III-D). We evaluate a range of models and use the labels lin, quad, and cubic to represent linear, quadratic, and cubic functions, respectively. Models are specified using a function type, followed by a parameter list in parentheses. Using this terminology, a model specified as $\mathrm{lin}(\mathrm{FO}, \mathrm{BB})$ would predict the capacitance of a net n, C_n, using a linear function of the net's fanout and its bounding box half-perimeter

$$\mathrm{lin}(\mathrm{FO}, \mathrm{BB}) : C_n = \alpha \cdot \mathrm{FO}_n + \beta \cdot BB_n + \gamma \qquad (10)$$

where α, β, and γ are scalar coefficients with values determined through regression analysis. Note that cross-variable terms (e.g., $FF_n \cdot BB_n$) are omitted, unless explicitly included in the parameter list.

D. Results and Discussion

Having analyzed the noise in capacitance estimation, we now evaluate the accuracy of our estimation approach. Fig. 10 gives results for some of the estimation models we evaluated. The vertical axis gives the average error in capacitance estimate for a given estimation model, shown on the horizontal axis. The error for a model was computed by averaging the absolute values of percentage estimation errors of all nets in the test circuit set. Models are labeled from M1 to M10, in order of increasing complexity.

Model M1 estimates capacitance using a linear function of fanout, yielding an error of about 84%. This represents the error one could expect in capacitance estimation at the *prelayout* stage. M2 incorporates physical data, namely, bounding box half-perimeter, and reduces error to 66%. In M3, the bounding box parameter is partitioned into separate x and y domains. Estimation accuracy is not improved and, therefore, we conclude there is very little directional bias in Virtex-II: the capacitance "cost" of using horizontal routes is approximately equal to that of vertical routes. Previous work on FPGA capacitance estimation, such as [19], used models equivalent to M2 or M3.

Beginning with M4, we insert architecture-specific parameters into the model. M4 includes NT_n, which is the number of CLB tiles in which a net n has pins. Incorporating this parameter reduces error from 66% to 54%. In model M5, the $X6_n$ and $Y6_n$ parameters are brought in (related to the hex-length

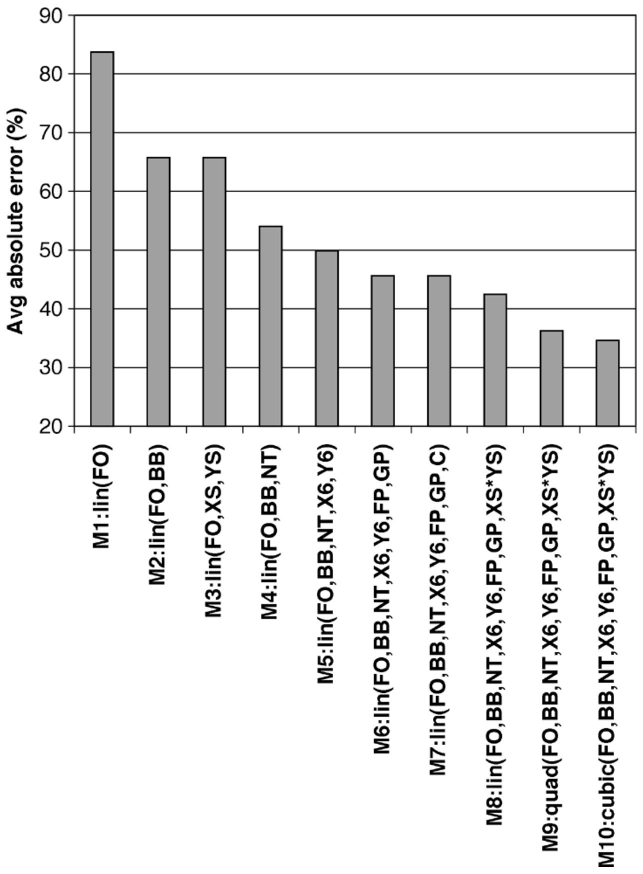

Fig. 10. Average error for a variety of prediction models.

resources in the interconnect) and error is further reduced, to about 50%. M6 considers the pin types on a net (through FP_n and GP_n) and yields an average error of 46%. Comparing the results for M6 to those for M3, we see the considerable benefits of tying model parameters to the underlying FPGA interconnect structure.

In model M7, congestion is introduced and surprisingly, very little benefit to error reduction is observed. There are a number of potential explanations for this. First, it is possible that there are sufficient routing resources in Virtex-II PRO such that routing congestion is not a problem and circuitous routes are not needed to achieve routability. We consider this to be likely and believe the routing stress imposed by the MCNC circuits to be relatively low in comparison with modern industrial designs. A second possibility is that the congestion metric employed does not accurately reflect routing congestion in Virtex-II PRO. The impact of congestion on routing in commercial FPGAs is not well studied and is likely to be highly architecture dependent.

Model M8 includes a cross term, the x span of a net multiplied by its y span $(\mathrm{XS}_n \cdot \mathrm{YS}_n)$. The intuition behind this is to differentiate nets that span both dimensions from those that span only a single dimension. Error is reduced somewhat, from 46% to 42%. Models M9 and M10 have the same parameter set as M8, but estimate capacitance using quadratic and cubic functions, respectively. Observe that using a quadratic function (M9) reduces error to about 36%. The benefits of moving to a cubic function (M10) are minimal. We also investigated higher order models but found they did not significantly improve estimation accuracy.

TABLE VII
PREDICTION MODEL AND REGRESSION ANALYSIS DETAILS (LOW-FANOUT AND HIGH-FANOUT PREDICTION MODELS)

Parameter	Coeff	P-Value	Lower 95% / Upper 95%
int	-544.425	< 2E-16	-662.320 / -426.530
FO	359.435	1.48E-13	264.419 / 454.452
BB	53.297	6.72E-10	36.412 / 70.182
NT	521.498	3.39E-15	392.219 / 650.777
GP	-177.252	1.23E-9	-234.301 / -120.202
$X6$	74.155	2.5E-4	34.537 / 113.774
$Y6$	52.402	0.00903	13.073 / 91.731
$XS \cdot YS$	-10.159	6.66E-6	-14.574 / -5.743
FO^2	22.694	0.09255	-3.751 / 49.139
BB^2	-1.222	0.01139	-2.168 / -0.276
NT^2	-69.444	2.25E-5	-101.536 / -37.352
FP^2	-44.168	0.00323	-73.551 / -14.785
GP^2	-53.275	0.00272	-88.097 / -18.453
$X6^2$	-9.777	0.02367	-18.246 / -1.308
$Y6^2$	-9.391	0.02421	-17.558 / -1.225
$(XS \cdot YS)^2$	0.053	0.00088	0.022 / 0.085
FO^3	-1.703	0.06502	-3.513 / 0.106
NT^3	4.638	7.92E-5	2.337 / 6.940
FP^3	3.862	0.01468	0.761 / 6.964
GP^3	5.335	0.00387	1.716 / 8.954
int	2012.409	9.96E-5	1011.598 / 3013.220
FO	67.576	0.04597	1.218 / 133.934
BB	-42.029	0.18612	-104.479 / 20.422
NT	182.588	0.00087	75.991 / 289.184
FP	216.387	5.01E-6	125.225 / 307.548

The parameter coefficients for M10 and other regression analysis data are provided in Table VII. The unshaded portion of the table shows the low-fanout net prediction model ($\leq$10 loads); the shaded portion of the table corresponds to the prediction model used for high-fanout nets. About 5% of the nets in the circuits are high-fanout. The columns of the table are analogous to those of Table III. Observe that a very simple model suffices for predicting the capacitance of high-fanout nets. Only the net's fanout, bounding box, the number of CLB tiles, and pin types are included. Further, the *P-value* for the bounding box term implies a fairly weak significance. We experimented with dropping this term and observed only a small decrease in prediction accuracy (<1%). We found that the use of more complex models for the high-fanout nets resulted in "overfitting" to the characterization circuits that led to lower prediction accuracy in the test circuits. This is likely due to a greater noise presence in high-fanout versus low-fanout nets. The overfitting phenomenon was not observed for the low-fanout net modeling: better fitting during characterization for such nets resulted in better prediction accuracy.

Model M10 yields average error values of about 35%. Error results for the individual test circuits are shown in column 2 of Table VIII. From Table VI, we see that the noise floor errors for these circuits fall in the 20%–24% range. The difference between the prediction and noise floor errors limits the potential for improvement in prediction accuracy. Given the range of routing resource types available in the FPGA, we consider the prediction accuracy to be quite good.

Fig. 11 plots the predicted (vertical axis) and actual (horizontal axis) capacitance values for all nets in the test circuit set, as predicted using model M10. Observe that capacitance is under-predicted for some nets and over-predicted for others, leading to under and overestimates of a net's power. The under

TABLE VIII
ERRORS FOR INDIVIDUAL CIRCUITS; RESULTS FOR ALTERNATE CHARACTERIZATION/TEST BENCHMARK DIVISION

Circuit	Mean absolute prediction error (%)	Alternate test circuit	Mean absolute prediction error (%)
alu4	32.9	pdc	37.4
seq	34.4	apex2	34.6
apex4	33.1	ex1010	34.9
ex5p	34.1	ex5p	33.9
cps	39.4	cps	39.3
dalu	32.2	dalu	33.6
C2670	38.8	C2670	38.9
Average:	35.0		36.1

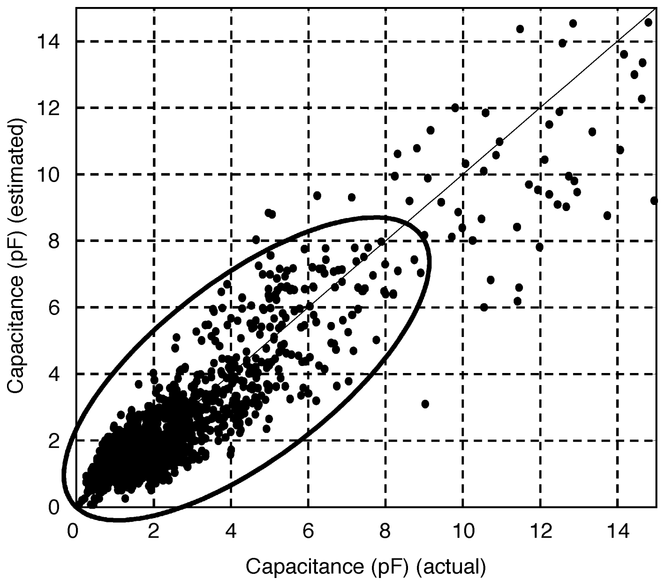

Fig. 11. Estimated versus actual values (approximately 4000 points in ellipse).

and over-predictions are roughly equally distributed and consequently, we anticipate that average power estimates made using the proposed model will be close to actual power values. Note also, the similarity between the estimation results and noise results (Figs. 8 and 11), which is quite interesting as the noise error cannot be resolved in estimation.

As with activity prediction, we investigated the dependence of our capacitance prediction model on the division of benchmarks into the characterization and test circuit sets. We reconstructed our model using the alternate benchmark division (described in Section III-D). Column 3 of Table VIII lists the benchmarks in alternate test set. Shading is used to show the benchmarks that are common to the two test sets. Column 4 gives error results for the alternate test circuit set. The error data for the common circuits can be compared with that in column 2 of the table. Observe that the error results for these circuits are similar, even though the characterization sets used for model construction differ. This provides evidence that the prediction model is robust and not strongly dependent on the benchmark division.

V. CONCLUSIONS

The dominance of interconnect in overall FPGA power consumption implies that understanding and managing switching activity and interconnect capacitance is a mandatory component of power-aware FPGA computer-aided design. In this paper, we studied activity and capacitance prediction for FPGAs and proposed models for the early prediction of these parameters. Our activity prediction approach estimates routed delay activity values for a net using zero or logic delay activity values as well as circuit functional and structural properties. The proposed capacitance prediction model uses generic parameters such as fanout and bounding box length as well as parameters that are specific to the underlying FPGA routing fabric. We conducted a noise analysis of activity and capacitance and established limits on the potential accuracy achievable in prediction. The prediction models work well given the noise limitations and we expect that they will be useful in applications such as low-power synthesis, early-power estimation, and power-aware layout.

REFERENCES

[1] V. George and J. Rabaey, *Low-Energy FPGAs: Architecture and Design*. Norwell, MA: Kluwer, 2001.

[2] L. Stok and J. Cohn, "There is life left in ASICs," in *Proc. IEEE Int. Symp. Physical Design*, 2003, pp. 48–50.

[3] L. Shang, A. Kaviani, and K. Bathala, "Dynamic power consumption in the Virtex-II FPGA family," in *Proc. ACM Int. Symp. Field-Programmable Gate-Arrays*, 2002, pp. 157–164.

[4] K. Poon, A. Yan, and S. Wilton, "A flexible power model for FPGAs," in *Proc. Int. Conf. Field-Programmable Logic and Applications*, 2002, pp. 312–321.

[5] F. Najm, "A survey of power estimation techniques in VLSI circuits," *IEEE Trans. VLSI Syst.*, vol. 2, pp. 446–455, Dec. 1994.

[6] H. Soeleman, K. Roy, and T.-L. Chou, "Estimating circuit activity in combinational CMOS digital circuits," *IEEE Des. Test Comput.*, pp. 112–119, Apr.–June 2000.

[7] A. Shen, A. Ghosh, S. Devadas, and K. Keutzer, "On average power dissipation and random pattern testability of CMOS combinational logic networks," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1992, pp. 402–407.

[8] H. Li, W.-K. Mak, and S. Katkoori, "LUT-based FPGA technology mapping for power minimization with optimal depth," in *Proc. IEEE Computer Society Workshop VLSI*, 2001, pp. 123–128.

[9] A. Farrahi and M. Sarrafzadeh, "FPGA technology mapping for power minimization," in *Proc. Int. Workshop Field-Programmable Logic and Applications*, 1994, pp. 167–174.

[10] *Virtex II PRO FPGA Data Sheet*, Xilinx Inc., San Jose, CA, 2003.

[11] J. Anderson and F. Najm, "Switching activity analysis and pre-layout activity prediction for FPGAs," in *Proc. IEEE Int. Workshop System-Level Interconnect Prediction*, 2003, pp. 15–21.

[12] ——, "Interconnect capacitance estimation for FPGAs," in *Proc. IEEE Asia South Pacific Design Automation Conf.*, Yokohama, Japan, 2004, pp. 713–718.

[13] *Stratix FPGA Device Handbook*, Altera Corp., San Jose, CA, 2003.

[14] (2003) The R Project for Statistical Computing. [Online] Available: http://www.r-project.org

[15] J. Cong and Y. Ding, "Flowmap: an optimal technology mapping algorithm for delay optimization in look-up-table based FPGA designs," *IEEE Trans. Computer-Aided Design*, vol. 13, pp. 1–12, Jan. 1994.

[16] R. Francis, J. Rose, and Z. Vranesic, "Technology mapping for lookup table-based FPGAs for performance," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1991, pp. 568–571.

[17] F. Najm, "Transition density: a new measure of activity in digital circuits," *IEEE Trans. Computer-Aided Design*, vol. 12, pp. 310–323, Feb. 1993.

[18] G. Yeap, *Practical Low Power Digital VLSI Design*. Norwell, MA: Kluwer, 1998.

[19] K. Roy, "Power-dissipation driven FPGA place and route under timing constraints," *IEEE Trans. Circuits Syst.*, vol. 46, pp. 634–637, May 1999.

[20] B. Kumthekar and F. Somenzi, "Power and delay reduction via simultaneous logic and placement optimization in FPGAs," in *Proc. IEEE Design, Automation Test Eur. Conf.*, 2000, pp. 202–207.

[21] A. Marquardt, V. Betz, and J. Rose, "Timing-driven placement for FPGAs," in *Proc. ACM Int. Symp. Field-Programmable Gate Arrays*, 2000, pp. 203–213.

[22] T. Karnik and S.-M. Kang, "An empirical model for accurate estimation of routing delay in FPGAs," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1995, pp. 328–331.

[23] P. Maidee, C. Ababei, and K. Bazargan, "Fast timing-driven partitioning-based placement for island style FPGAs," in *Proc. IEEE Design Automation Conf.*, 2003, pp. 598–603.

[24] M. Hutton, K. Adibsamii, and A. Leaver, "Adaptive delay estimation for partitioning-driven PLD placement," *IEEE Trans. VLSI Syst.*, vol. 11, pp. 60–63, Feb. 2003.

[25] S. Bodapati and F. Najm, "Pre-layout estimation of individual wire lengths," in *Proc. ACM Int. Workshop System-Level Interconnect Prediction*, 2000, pp. 91–96.

[26] (2003) Xilinx Power Tools. [Online]. Available: http://www.xilinx.com/ise/power_tools

[27] J. Lou, S. Thakur, S. Krishnamoorthy, and H. S. Sheng, "Estimating routing congestion using probabilistic analysis," *IEEE Trans. Computer-Aided Design*, vol. 21, pp. 32–41, Jan. 2002.

Jason H. Anderson (S'97) received the B.Sc. degree in computer engineering from the University of Manitoba, Winnipeg, MB, Canada, in 1995, and the M.A.Sc. degree in electrical and computer engineering from the University of Toronto, Toronto, ON, Canada, in 1997, where, since 2001, he has been working toward the Ph.D. degree in computer engineering.

In 1997, he joined Xilinx, Inc., San Jose, CA, as a member of Implementation Tools Group, where he developed placement and routing tools for Xilinx field-programmable gate arrays (FPGAs). Presently, he is a Senior Staff Engineer at the Xilinx Toronto Development Center, Toronto, ON, Canada. He is an inventor on more than a dozen issued and pending U.S. patents. His research interests include all aspects of computer-aided design (CAD) and architecture for FPGAs.

Mr. Anderson received the Ross Freeman Award for Technical Innovation, the highest innovation award given by Xilinx, for his contributions to the Xilinx placer technology in 2000. He was awarded the Natural Sciences and Engineering Research Council (NSERC) of Canada Postgraduate Scholarship in 2001, and the Ontario Graduate Scholarship in 2003.

Farid N. Najm (S'85–M'89–SM'96–F'03) received the B.E. degree in electrical engineering from the American University of Beirut (AUB), Beirut, Lebanon, in 1983, and the M.S. and Ph.D. degrees in electrical and computer engineering (ECE) from the University of Illinois at Urbana-Champaign (UIUC) in 1986 and 1989, respectively.

He worked as an electronics engineer with AUB from 1983 to 1984. In 1989, he joined Texas Instruments in Dallas, TX, as a Member of the Technical Staff with the Semiconductor Process and Design Center. In 1992, he joined the ECE Department at UIUC as an Assistant Professor, where in 1997, he was promoted to a Tenured Associate Professor. In 1999, he joined the ECE Department at the University of Toronto, Toronto, ON, Canada, where he is currently Professor and Vice-Chair of ECE. He is coauthor of *Failure Mechanisms in Semiconductor Devices*, (New York: Wiley, 2nd Ed., 1997). His research interests include the general area of CAD tool development for low-power and reliable VLSI circuits, including power estimation and modeling, low-power design, power-grid analysis and verification, and reliability analysis and prediction.

Dr. Najm is an Associate Editor for the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN (CAD). He received the IEEE TRANSACTIONS ON CAD Best Paper Award in 1992, the National Science Foundation Research Initiation Award in 1993, and the National Science Foundation CAREER Award in 1996. From 1997 to 2002, he was an Associate Editor for the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS. He has also served on the technical committees of various IEEE conferences.