Switching Activity Analysis and Pre-Layout Activity Prediction for FPGAs

Jason H. Anderson
Dept. of Electrical and Computer Engineering
University of Toronto
Toronto, Ontario, Canada M5S 3G4
janders@eecg.toronto.edu

Farid N. Najm
Dept. of Electrical and Computer Engineering
University of Toronto
Toronto, Ontario, Canada M5S 3G4
f.najm@utoronto.ca

ABSTRACT

It is well-known that dynamic power dissipation in digital CMOS circuits depends linearly on switching activity. In this paper, we study switching activity in a commercial FPGA and propose a novel approach to pre-layout activity prediction. 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). Low-power synthesis and early power estimation are typically done on the basis of zero delay activity values, with the assumption that such values correlate well with routed delay activity values. We investigate whether this assumption is valid for FPGA technologies, where critical path delay is often dominated by interconnect delay. We then present an approach for early prediction of routed delay activity values. Our approach is novel in that it estimates each net's routed delay activity using only zero or logic delay activity values along with structural and functional properties of a circuit. Results show that in comparison with zero or logic delay activity values, the predicted activity values are substantially more representative of routed delay activity values.

Categories and Subject Descriptors

B.7 [**Integrated Circuits**]: Design Aids; C.4 [**Performance of Systems**]: Modeling techniques

General Terms

Design, Algorithms

Keywords

Field-programmable gate arrays, FPGAs, power, estimation, low-power design

1. INTRODUCTION

Since their introduction in the mid-1980s, the speed and capacity of field-programmable gate arrays (FPGAs) has been steadily increasing. Through architectural enhancements, supply voltage reductions and technology scaling, the power consumed per FPGA logic element has been decreasing with each successive FPGA generation [5]. However, these power reductions have been more than offset by the increasing logic density of FPGAs, with the result being that the power consumed by the largest devices is now measured in watts. State-of-the-art FPGAs are capable of implementing complex systems having millions of gates and operating at frequencies in the hundreds of megahertz [1]. To minimize design time for such systems, power estimation tools are needed to gauge power dissipation early in the design flow, especially prior to the time consuming layout phase. Such tools allow design trade-offs to be considered at a high level of abstraction, reducing design effort and cost.

Several analyses of FPGA power consumption have appeared recently in the literature [12, 5, 10]. These works have shown that power dissipation in FPGA devices is predominantly in the programmable interconnection network. In the Xilinx Virtex-II [1] 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 [12]. The reason for the dominance of interconnect in FPGA power consumption lies in the composition of the interconnect structures, which consist of pre-fabricated wire segments of various lengths, with used and unused routing switches attached to each wire segment.

Currently, the majority of power dissipation in FPGAs is dynamic power dissipation [12], resulting from the charging and discharging of parasitic capacitance and characterized by:

$$P_{avg} = \frac{1}{2} \sum_{n \in nets} C_n \cdot f_n \cdot V^2 \quad (1)$$

where P_{avg} represents average power consumption, C_n is the load capacitance of a net n, V is the voltage supply and f_n is the average toggle rate or *switching activity* of net n. 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 [9, 14].

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 in fact that 20-70% of total power dissipation in ASICs can be due to glitches [13].

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 presence of programmable switches in the interconnection network, path delays in FPGAs are generally dominated by interconnect rather than logic delays. FPGA interconnect delays are variable and often difficult to predict [6], 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 [7, 3], 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 this paper, we study switching activity in the Xilinx Virtex-II FPGA. We 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. High-level power estimation using (1) requires accurate prediction of the capacitance and activity of each net. In this paper, we address the latter problem and present a novel approach for activity prediction. Our approach 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 the circuit. The method operates at the *pre-layout* stage and thus, we envision that our predictor could be applied in a variety of scenarios, such as early power estimation/planning, low-power synthesis or whenever routed delays are unknown or delay-aware activity data is unavailable. We demonstrate our prediction method by applying it to predict activity values for Virtex-II. The paper is organized as follows: In Section 2 we present our activity analysis. Our approach to activity prediction is described in Section 3. Conclusions are offered in Section 4.

2. ACTIVITY ANALYSIS

We use a simulation-based approach to study switching activity in 14 of the largest MCNC combinational circuits mapped into Virtex-II FPGAs. The CAD flow we employ to gather activity data is shown in Figure 1. HDL benchmark circuits are first synthesized using Synplicity's Synplify Pro (ver. 7.0), and then technology mapped, placed and routed using Xilinx tools (ver. 4.1i). Each circuit is mapped into the smallest Virtex-II device able to accommodate it[1].

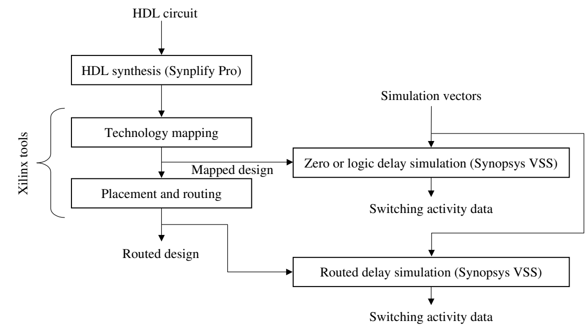


Figure 1: CAD flow for activity analysis

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-input look-up-tables (4-LUTs), flip-flops, and multiplexers. For the delay-based simulations, an SDF (standard delay format) file, generated by the Xilinx annotation tool, is provided to VSS.

Circuits are simulated using 5000 randomly chosen input vectors. Two different vector sets were generated for each circuit: one representing high input activity and a second representing low input activity. In the high (low) activity vector set, the probability of an individual input toggling between successive vectors is 50% (25%).

2.1 Analysis Results and Discussion

Using the flow of Figure 1, we examined how switching activity changes when delays are considered. Columns 2-5 of Table 1 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.

From the data in Table 1, 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 5% to 84%, with all but two of the circuits having an increase larger than 15%. 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

[1]We target Virtex-II devices with the -6 speed grade. The placement and routing tools were run at the highest effort level.

Table 1: Effect of glitching on switching activity

Circuit	High activity vector set: % increase in transitions for **logic** dly sim vs. zero dly sim	High activity vector set: % increase in transitions for **routed** dly sim vs. zero dly sim	Low activity vector set: % increase in transitions for **logic** dly sim vs. zero dly sim	Low activity vector set: % increase in transitions for **routed** dly sim vs. zero dly sim	High activity vector set: **Logic** delay net activity mean error (std. dev.)	High activity vector set: **Zero** delay net activity mean error (std. dev.)
C3540	83.4	137.9	66.5	98.4	25.9 (9.9)	50.8 (17.5)
apex2	17.8	49.6	11.3	26.4	25 (12.5)	37.4 (14.8)
ex5p	49.3	137.1	44.1	92.6	36.8 (13.3)	49 (16.9)
ex1010	33.5	114.4	20.2	60.3	37.5 (16.4)	50.9 (20.6)
spla	5.5	24.5	3.6	12.4	20.6 (14)	25.4 (16.2)
C2670	22.2	58.3	18.5	40.3	31.8 (15)	48.1 (18.8)
pdc	22.3	62.6	13.6	35.6	29.8 (13.1)	41.3 (16.6)
alu4	16.3	64.7	10.2	32.4	30.6 (13)	39.9 (17.1)
seq	17.3	48.7	10.3	24.9	24 (14.3)	34.5 (15.8)
apex4	39.2	98.2	23.6	50.5	30.7 (14.3)	46.5 (19.2)
pair	27.6	42.6	17.3	25.2	18.7 (14.1)	40.4 (15.9)
cps	13.9	54.2	8.3	26.2	28.6 (13.5)	38.2 (16)
dalu	18.8	58.6	13.7	34.6	33.1 (13.2)	48.1 (15.6)
misex3	22.3	63.8	14.4	34.1	27.8 (11.8)	40.1 (17.1)

count versus the zero delay simulations ranges from 24% to 138% and is greater than 40% for 13 of the 14 circuits. 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:

$$error\_zero(n) = 100 \cdot \frac{|trans_{zero}(n) - trans_{routed}(n)|}{trans_{routed}(n)} \quad (2)$$

where $trans_{routed}(n)$ and $trans_{zero}(n)$ 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 1, which shows the average and standard deviation of net error for each circuit. Note that for this analysis, we ignored the error on nets that transitioned on fewer than 5% of the simulation vectors (experienced fewer than 250 transitions) as we did not consider the error data for such low activity nets to be statistically significant. In Table 1, we see that the mean error in the logic delay activity values falls in the 18-38% range. The mean error in zero delay activity ranges from 35-50% for all but one of the circuits. We also observe that coupled with these large mean errors are large error deviations, ranging from 10-16% for the logic delay case and 15-20% for the zero 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.

Table 2: Effect of glitching on activity in timing-constrained designs

Circuit	% increase in transitions for **routed** dly sim vs. zero dly sim
C3540	118.4
alu4	50.6
spla	22.6

To reduce path delays in circuits, designers typically specify timing constraints to the synthesis and layout tools. To investigate the effect of constraints on activity, we supplied pad-to-pad delay constraints to the placement and routing tools for three circuits and re-simulated the circuits to determine new transition counts. The results of this experiment (for the high activity vector set) are given in Table 2. Comparing the data in Table 2 with that in Table 1, we observe that the transition count increase in the constrained circuits (from the zero to routed delay simulation) is smaller than the transition count increase in the unconstrained circuits. Reducing delays through constraints does indeed reduce glitching; however, in these circuits, the reduction is not that substantial.

A potential source of inaccuracy in any CAD research relates to the choice of benchmark circuits. It is well-known that glitching can be reduced by balancing delay paths or reducing combinational depth [11]. We believe that the depths of the MCNC circuits are generally larger than the depths of modern industrial circuits, which are typically subjected to delay optimizations such as pipelining and re-timing. Such optimizations also reduce glitching severity, and for this reason, it is conceivable that modern circuits may experience less glitching than the MCNC circuits. On the other hand,

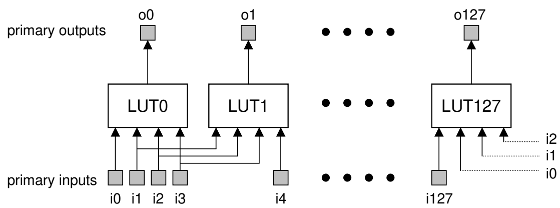


Figure 2: Circuit with regularity

the size of modern circuits is up to two orders of magnitude larger than the MCNC circuits. Larger circuits may possess longer interconnect delays, which increase opportunities for glitching.

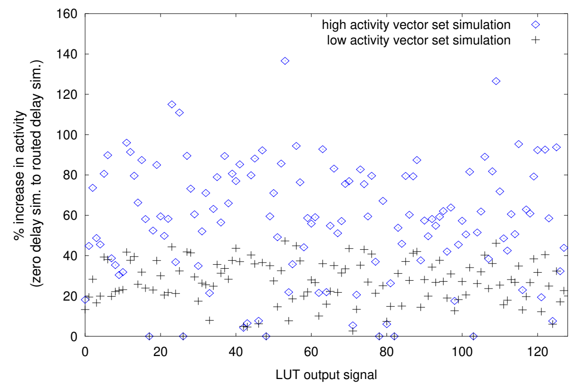


Figure 3: Activity change in regular circuit

3. ACTIVITY PREDICTION

The results in Section 2 showed that switching activity increases substantially when delays are brought into the picture. In this section, we present our approach to pre-layout routed delay activity prediction. Prior to this however, we gauge the difficulty of the prediction problem.

3.1 Problem Difficulty

In an effort to understand the difficulty of generating accurate pre-layout activity estimates, we performed an activity analysis on the circuit shown in Figure 2. The circuit is highly regular from the structural and functionality viewpoint and consists of 128 inputs driving 128 4-input look-up-tables (LUTs), which in turn drive 128 outputs. 4-input LUTs are small memories capable of implementing any logic function having ≤ 4 inputs; most commercial FPGAs use 4-input LUTs in their logic blocks [1]. Each LUT in the circuit is programmed to implement a 4-input logical AND function.

We mapped the circuit in Figure 2 into the Virtex-II FPGA 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 Figure 3. 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[2]. 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 100%.

Real circuits are likely to be much less regular than the circuit of Figure 2, and we therefore conclude that it will be difficult to achieve a high degree of accuracy in activity prediction at the pre-layout stage. Nevertheless, in the next section, 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.

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

3.2 Prediction Approach

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) 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 $inputs(y)$ 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 is the length of the longest path from any primary input to the node. 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 (# of LUTs) is frequently used as a predictor of path delay at the pre-layout stage [2, 4]. 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 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 $y = f(x_1, x_2, ..., x_n)$. Similar to carry signal operation, glitches on y's output may come from two sources: they may be propagated from one of y's inputs, $x_1, x_2, ..., x_n$, 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:

$$predict(y) = \alpha \cdot gen(y) + \beta \cdot prop(y) + \phi \quad (3)$$

where α, β and ϕ are scalar coefficients, 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. In the next section, we will establish an empirical relationship between the value of (3) for a net and the change in the net's activity due to glitching when delays are accounted for. 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 few parameters. Let $PL(y)$ represent the *set* of different path lengths from a primary input to node y. This parameter can be computed easily during a input-to-output DAG traversal by maintaining 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:

$$PL(y) = \bigcup_{x_i \,\epsilon\, inputs(y)} \{p + 1 \mid p \,\epsilon\, PL(x_i)\} \quad (4)$$

Observe that a given node may have a larger set of path lengths than any of its immediate fanins. Consider the example shown in Figure 4, 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. The number of path lengths introduced at node y is defined as:

$$IPL(y) = \min_{x_i \,\epsilon\, inputs(y)} \{|PL(y)| - |PL(x_i)|\} \quad (5)$$

We are now ready to define the generate term of (3):

$$gen(y) = IPL(y) + \gamma \cdot depth(y) \quad (6)$$

where γ is a scalar coefficient. 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 second term of (6) reflects the fact that glitching severity typically increases with combinational depth. Thus, a node with shallow depth is likely to experience less glitching than a deep node, even if the number of path lengths to the two nodes is similar.

The propagate term of (3) borrows ideas from the concept of transition density [8] and uses the notions of *Boolean difference* and *static probability*, which we briefly review here. The Boolean difference of a function, $y = f(x_1, x_2, ..., x_n)$, with respect to one of its inputs, x_i, is defined as:

$$\frac{\partial y}{\partial x_i} = f_{x_i} \oplus f_{\overline{x}_i} \quad (7)$$

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

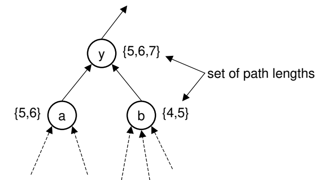


Figure 4: Finding the set of path lengths for y

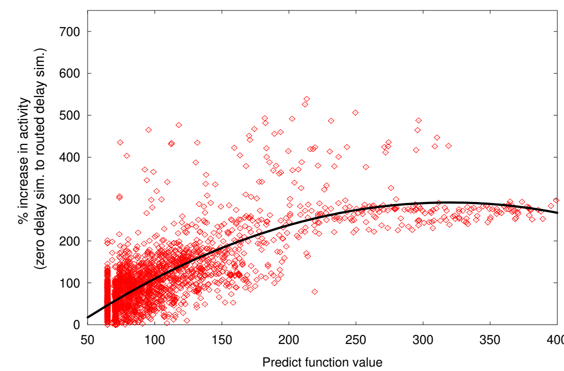


Figure 5: Prediction function value versus activity increase

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, $P(\frac{\partial y}{\partial x_i})$, 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 $P(\frac{\partial y}{\partial x_i})$. 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:

$$prop(y) = \frac{\sum_{x_i \,\epsilon\, inputs(y)} P(\frac{\partial y}{\partial x_i}) \cdot trans_{zero}(x_i) \cdot predict(x_i)}{\sum_{x_i \,\epsilon\, inputs(y)} P(\frac{\partial y}{\partial x_i}) \cdot trans_{zero}(x_i)} \quad (8)$$

The $P(\frac{\partial y}{\partial x_i}) \cdot trans_{zero}(x_i)$ in the numerator can be viewed as a weight quantifying the influence of glitching on x_i to glitching on y. The denominator of (8) normalizes the values computed by the propagate function so they are relatively independent of the transition counts and probabilities involved. Note that $trans_{zero}$ in (8) can be replaced with $trans_{logic}$ if logic delay activity data is available.

3.3 Prediction Results and Discussion

To evaluate our prediction approach, we divided our benchmark circuits into two sets, a *characterization* set and a *test* set, each containing 7 circuits. We use the characterization circuits to derive a model relating the activity increase on a net due to glitching to the *predict function* (3). We employed multiple linear regression analysis to select values for parameters α, β, γ and ϕ in (3) and (6). This tunes the parameters of our model to a particular FPGA device and CAD flow. In practice, such model characterization would

Table 3: Error in predicted activity values

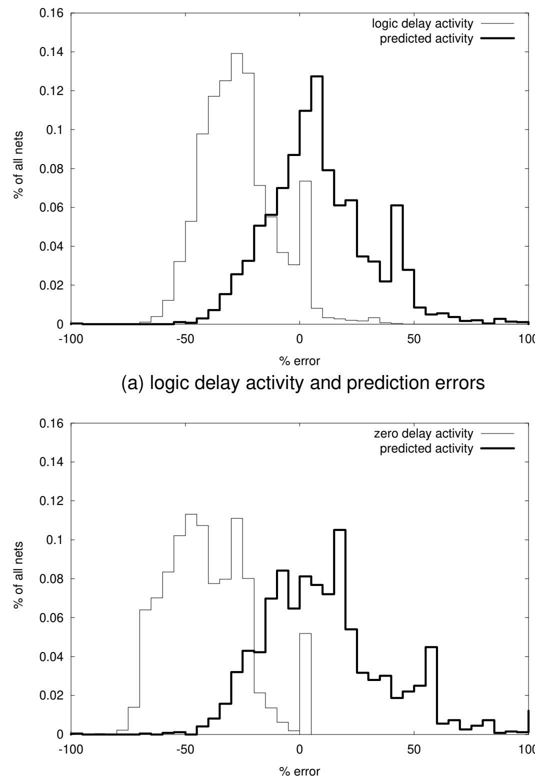
Circuit	**Predicted** net activity mean error (std. dev.) (from logic dly activity)	**Predicted** net activity mean error (std. dev.) (from zero dly activity)
alu4	14.4 (11.8)	17.8 (13.4)
seq	21.5 (17)	29.6 (26.1)
apex4	17.5 (18.1)	23 (22.1)
pair	23.5 (16.6)	21.5 (18.8)
cps	15.8 (13.5)	20.0 (15.7)
dalu	15.5 (13)	18.7 (13.1)
misex3	14 (12.2)	17.9 (13.5)

be done by an FPGA vendor to produce an activity prediction model for use by designers in the field. Following characterization, we apply the model to predict the activity increase on nets in the test circuit set. In these experiments, to apply our model we use static probability and toggle data extracted from a zero or logic delay simulation (see below). However, such data need not be derived from simulation; it can be computed efficiently using probabilistic approaches, such as those described in [15]. Thus, simulation is not a requirement for the use of our prediction model.

We begin by plotting the increase in activity on a net in the routed delay case (relative to its activity in the zero delay case) versus the value produced by our predict function for that net. The results for the characterization circuits, using the high activity vector set simulation data, are shown in Figure 5. Each point in the figure represents a single net in one of our benchmarks. Figure 5 shows that generally, increasing amounts of glitching correlate with increasing prediction function values. The figure also shows there to be a range of activity increases for each prediction function value. This is expected and in line with our analysis of the difficulty of the prediction problem in Section 3.1. Figure 5 also shows a quadratic line of best fit, which has the equation: $y = -0.004x^2 + 2.4x - 93.7$.

To apply our prediction method, we first compute the value of (3) for a net in one of the test circuits. We then use this value as the argument in the equation of the best fit line in Figure 5 (derived from the characterization circuits), yielding a predicted percentage increase in activity for the net versus the net's zero delay activity. We use the predicted percentage increase to predict the routed delay activity for the net. A similar process is used to develop and apply a model that predicts routed delay activity values from *logic* rather than zero delay activity values.

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 3. The table gives the average absolute percentage error across all nets for each circuit; error deviations are shown in parentheses. Column 2 of the table shows the error results for a model that predicts routed delay activity from *logic* delay activity; column 3 gives results for a model that predicts routed delay activity from *zero* delay activity. In Section 2.1, we saw that logic delay activities are "closer" to routed delay activities than are zero delay activities. Table 3 confirms that using logic delay activities as the basis of a prediction model yields



(a) logic delay activity and prediction errors

(b) zero delay activity and prediction errors

Figure 6: Error histograms

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 1. 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. The only exception to this is the logic delay activity-based predicted values for the circuit *pair*, which exhibit slightly worse error.

Figure 6 shows histograms of true rather than absolute error data for all of the test circuits. The figure shows the percentage of nets (in all test circuits) having an error in a specific range. Part (a) of the figure shows the error in logic delay activity values and (logic delay activity-based) predicted values; part (b) gives the analogous data for zero delay activity values. 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 centered around zero. The use of zero or logic delay activity values in power estimation will lead to significant underestimates of circuit power. Conversely, since our approach underpredicts activity for some nets and overpredicts for others, we expect that the use of the predicted activity values will produce average power estimates much closer to actual 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.

The prediction results presented above were based on the simulation data for the high activity vector set. We regenerated Figure 5 for the low activity vector set data and found the shape to be similar, though shifted lower on the y-axis due to the 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 our prediction approach to automatically account for various amounts of primary input switching activity.

4. CONCLUSIONS

In this paper, we studied switching activity in FPGA technology and showed that activity increases considerably when delays are accounted for versus when delays are zero. We demonstrated the difficulty of predicting routed delay activity values at the pre-layout stage and proposed a novel approach for pre-layout activity prediction. Our method estimates routed delay activity values using zero or logic delay activities as well as a circuit's structural properties and functionality. The key advantages of our prediction approach are its simplicity and its ability to eliminate the one-sided bias in error associated with the use of zero or logic delay activity values. We believe that the proposed approach will be useful in low-power synthesis systems or early power estimators, when routed delay activity data is unavailable.

5. ACKNOWLEDGEMENTS

The authors gratefully acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada.

6. REFERENCES

[1] *Virtex II Platform FPGA Data Sheet*. Xilinx, Inc., San Jose, CA, 2002.

[2] J. Cong and Y. Ding. Flowmap: An optimal technology mapping algorithm for delay optimization in lookup-table based FPGA designs. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, pages 1–12, January 1994.

[3] A. H. Farrahi and M. Sarrafzadeh. FPGA technology mapping for power minimization. In *Proc. Int. Workshop on Field-Programmable Logic and Applications*, pages 167–174, 1994.

[4] R.J Francis, J. Rose, and Z. Vranesic. Technology mapping for lookup table-based FPGAs for performance. In *IEEE Int. Conf. on Computer-Aided Design*, pages 568–571, 1991.

[5] V. George and J. M. Rabaey. *Low-Energy FPGAs: Architecture and Design*. Kluwer Academic Publishers, Boston, MA, 2001.

[6] M. Hutton, A. Leaver, and K. Adibsamii. Timing-driven placement for hierarchical programmable logic devices. In *ACM Int. Symp. on FPGAs*, pages 3–11, 2001.

[7] H. Li, W-K. Mak, and Srinivas Katkoori. LUT-based FPGA technology mapping for power minimization with optimal depth. In *IEEE Computer Society Workshop on VLSI*, pages 123–128, 2001.

[8] F. Najm. Transition density: A new measure of activity in digital circuits. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 12:310–323, February 1993.

[9] F. Najm. A survey of power estimation techniques in VLSI circuits. *IEEE Trans. on VLSI Systems*, 2(4):446–455, December 1994.

[10] K. Poon, A. Yan, and S. J. E. Wilton. A flexible power model for FPGAs. In *Int. Conf. on Field-Programmable Logic and Applications*, pages 312–321, La Grande Motte, France, 2002.

[11] J. Rabaey and M. Pedram. *Low Power Design Methodologies*. Kluwer Academic Publishers, Boston, MA, 1996.

[12] L. Shang, A. Kaviani, and K. Bathala. Dynamic power consumption of the Virtex-II FPGA family. In *ACM Int. Symposium on FPGAs*, pages 157–164, 2002.

[13] A. Shen and et. al. On average power dissipation and random pattern testability of CMOS combinational logic networks. In *IEEE Int. Conf. on Computer-Aided Design*, pages 402–407, 1992.

[14] H. Soeleman, K. Roy, and T.-L. Chou. Estimating circuit activity in combinational CMOS digital circuits. *IEEE Design and Test of Computers*, pages 112–119, April-June 2000.

[15] G. Yeap. *Practical Low Power Digital VLSI Design*. Kluwer Academic Publishers, Boston, MA, 1998.