A Tool for Activity Estimation in FPGAs

E. Todorovich[1], M. Gilabert[1], G. Sutter[1], S. Lopez-Buedo[2], and E. Boemo[2]

[1] INCA, Universidad Nacional del Centro, Tandil, Argentina
{etodorov, gsutter}@exa.unicen.edu.ar

[2] Computer Engineering School, Universidad Autonoma de Madrid, España
{eduardo.boemo, sergio.lopez-buedo}@uam.es

Abstract. In this paper, an activity estimation tool for FPGA-based combinational circuits is presented. The current version is able to estimate average activity for individual nodes. The tool is statistical-based, allowing the user to specify the tolerated error at a given confidence level. The tunable properties of the implemented technique have been carefully tested, demonstrating how the designer can control the accuracy-speed trade-off. The importance of a realistic input pattern characterization has also been verified.

1 Introduction

The main problem in power estimation for CMOS circuits is the activity measurement. Node activity is hard to estimate because it depends on the values at the primary inputs, the logical function of the circuit and finally, the temporal and spatial correlations among the inputs. Additionally, the so-called *pattern-dependence* problem is present: In actual circuits, is practically impossible to evaluate all possible input vector combinations, as well as to consider the effect of glitches, the other source of activity [1]. Thus, power estimation algorithms lead to an important computational effort.

Several techniques have been developed to estimate the power consumption of digital circuits. Each technique proposes a different approache to solve the pattern-dependence problem. Comprehensive surveys about power estimation can be found in [2]-[4]. In the FPGA arena, a coarse approximation to power estimation in Xilinx FPGAs is developed in [5]-[8]. Other estimation technique applicable to FPGAs is implemented in [9]. It is based on the propagation of probabilistic parameters from primary inputs to all the internal circuit nodes. Finally, in the latest *Integrated Software Environment (ISE),* a power estimation tool, called *XPower* [10], [11] is presented. However, the user still must to provide an arbitrary input vector sets. So, the tool cannot guarantee that simulated activity really converges to the average values. Also, the current version of *XPower* does not support all Xilinx FPGA families.

This work tries to contribute to the previous research lines by the development of a new FPGA-oriented activity estimator. Its main features are: integration with commercial design tools, automatic generation of input vectors according to user

specifications, and finally, applicability to any FPGA device. The current version of the tool can be integrated in the Xilinx *Foundation* suite.

2 Statistical Activity Estimation

The statistical approach for power estimation is based on the Monte Carlo simulation technique. It minimizes the pattern dependence problem: randomly generated input patterns are applied at the circuit inputs whilst the activity per time interval T is monitored by a simulator. The process continues until a *stopping criterion* is reached.

The first work applying a Monte Carlo technique for total average power estimation is [12]. In [13], and later [14], the technique is extended, providing both the total and individual-gate power values. Other works made use of the statistical approach on sequential circuits. For example, a warm-up period and Markov chain theory are utilized in [15]. A statistical technique for large sequential circuits like microprocessors is presented in [16].

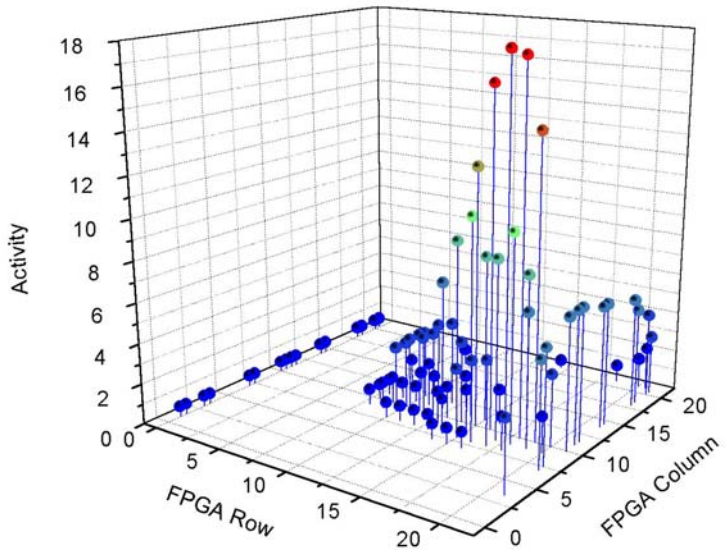

Fig. 1. 8-bit multiplier activity over a 20 × 20 CLB XC4010E FPGA. Node activities belonging to a single CLB are added

If the power consumed by a circuit over a period T has a distribution very close to normal, and if the successive input patterns are independently generated, it can be demonstrated that the required number, N, of random samples is:

$$N \geq \left(\frac{t_{\alpha/2} s}{p\varepsilon} \right)^2 . \quad \textbf{(1)}$$

This equation defines a stopping criterion where p is the average of the random power (activity) samples over a period T, s is the standard deviation of the random sample, $(1 - \alpha) \times 100\%$ is the confidence level that error ε in the measurement is less than a specified value. Finally, $t_{\alpha/2}$ is obtained from a t-distribution with $(N - 1)$ degrees of freedom.

Eq.1 leads to the so-called *slow convergence* problem. That means that this stopping criterion cannot be used to estimate individual gate activity: the lower is p, the larger is N. But individual gate estimations are useful to diagnose high consumption problems, and also to find the circuit nodes that consume more energy. For example, Fig.1 shows how the average activity increases at internal nodes in a FPGA from the inputs to the outputs.

In order to solve the slow convergence problem, a partition of the circuit nodes in two sets is proposed in [13] and [14]. If n is the measured average activity over a period T, and s is its standard deviation, the user defines an activity threshold n_{min} that classifies the nodes into regular and low density ones.

$$N \geq \left(\frac{z_{\alpha/2} s}{\bar{n} \varepsilon_1} \right)^2 \text{ . (a)} \qquad N \geq \left(\frac{z_{\alpha/2} s}{n_{\min} \varepsilon} \right)^2 \text{ . (b)} \tag{2}$$

Eq. 2a and 2b are used as stopping criterion for the regular nodes ($n > n_{min}$) and low-density nodes ($n < n_{min}$) respectively. It bounds the maximum number of samples tolerated by the algorithm. In both cases, the stopping criterion is tested after $N > 30$. Considering that low-density nodes have a negligible effect on the power figure of the circuit, this strategy reduces the execution time with a little penalty. ε_1 is an upper bound of the percentage error, ε is the user specified error tolerance ($\varepsilon = \varepsilon_1 / (1-\varepsilon_1)$). Thus, the product $n_{min}\varepsilon$ represents an absolute error limit that characterizes the accuracy for low-density nodes.

The benefits of the statistical approach are: a) any standard simulator can be used in the inner loop of the Monte-Carlo program making the technique easy to implement; b) if the tolerated error is not selected too small, the execution time can compete with the ones of probabilistic techniques; c) a simple input specification can be defined; d) temporal and spatial correlations are considered; and finally, e) glitches are taked into account.

3 Implementation Details

The described technique can be implemented using any simulator that reports the circuit activity, provided that it is able to interact with an external program that controls the simulation. Active-HDL [17] fulfills these conditions. Basically, a wrapper over the simulator was developed to implement the estimation technique.

Before using the tool, some steps must be done: The *ncd* file, generated after the place and route stages, must be used to produce the associated VHDL model and the *sdf* files (Standard Delay Format) [18]. For this purpose, Xilinx Foundation provides two commands: *ngdanno* and *ngd2vhdl*. The obtained VHDL model takes into

account the actual layout for the selected device, and the *sdf* file gives the simulator an accurate delay model.

The implementation of the software that governs the simulator is based on *Tcl-Tk* scripts. These scripts can execute external programs and Active-HDL macros that run simulator commands. The main procedure performs the following actions: 1) Call the program (Fig. 2) that shows the user interface, 2) Call the *Tk* interface with feedback about the estimation progress, 3) Initialize the simulator, 4) Execute the core script that actually runs the estimation algorithm, and 5) Build a report based on the resultant switching activity.

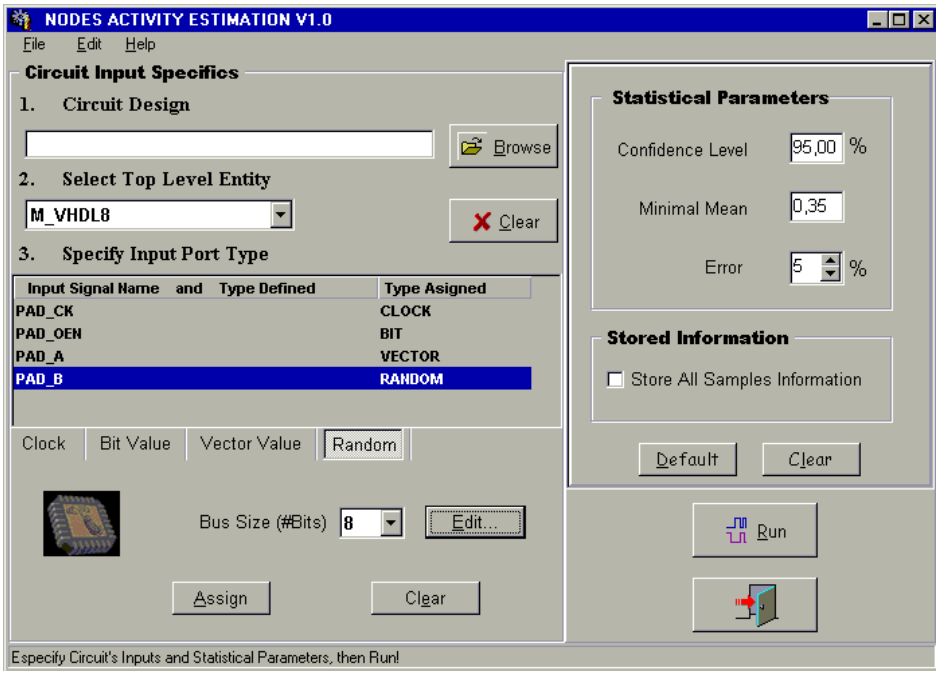

Fig. 2. Activity estimation tool: User interface

3.1 User Configuration - Input Specification

User configuration is divided in two parts: the statistical parameter settings, and the activity specification on the primary inputs.

The first ones are necessary to perform the individual nodes activity estimation, applying a Monte Carlo simulation according to Eq. 2. In the other hand, the file containing the VHDL model must be specified, and the top-level entity must be selected. For this entity, all the port descriptions are loaded. Then, the user must choose the activity type for each port: Clock, Bit Value, Vector Value, or Random

(Fig. 2). An alternative way to configure the tool is loading a previously defined configuration file.

By default, each line x_i of an input port specified as Random has a signal probability 0.5 and an average transition count per clock cycle 0.5. This is equivalent to assume temporal independence. But users can set each input signal to any signal probability $P_s(x)$ or transition density $D(x)$. Any individual input line of type Random can also be defined as connected to a counter or to a fixed logical signal. This allows the user to specify complex activity configurations at primary inputs.

3.2 The Monte Carlo Implementation

The main loop for the activity estimation was also implemented with a Tcl script. This script iteratively calls several programs until the stopping criterion is met for all the nodes. The iteration body is made of the following steps: 1) Call the program that generates a set of input vectors according to user specifications; 2) The simulator run the generated command file with these input vectors; 3) Other Active-HDL script of macros saves the resulting activity; 4) The executable core analyzes the activity reported by the simulator. This module keeps the necessary data in a simple database. Finally two executable programs are called to: 5) To update the mean activity and standard deviation for each node, and 6) To evaluate if the stopping criteria is reached for all nodes.

Table 1. Test circuits

Circuit	# Inputs	# Outputs	# Nodes	Device	# CLBs
C1: Behavioral VHDL Multiplier	16	16	769	XC4010E PC84 -4C	54
C2: Hatamian-Cash Multiplier	16	16	1447	XC4010E PC84 -4C	96

Table 2. Long simulation results. For these runs, 99% confidence and 1% error was specified. The minimum mean varies between 0.1 and 2 transitions per clock cycle

	VHDL Behavioral Multiplier		Hatamian-Cash Multiplier	
Min. Mean	# Samples	Total Av. Trans. per clock cycle	# Samples	Total Av. Trans. per clock cycle
0.10	364900	981	-	-
0.35	367750	979	509450	1226
0.50	347950	979	378950	1225
1.00	225050	980	244600	1224
1.50	180050	981	159550	1229
2.00	116400	978	116750	1229

4 Results

In order to evaluate the tool, two combinational multipliers whose main characteristics are listed in Table 1 were analyzed.

First, a set of long simulations was run to obtain values to compare with. The number of samples for each run, and the total average number of circuit transitions are shown in Table 2.

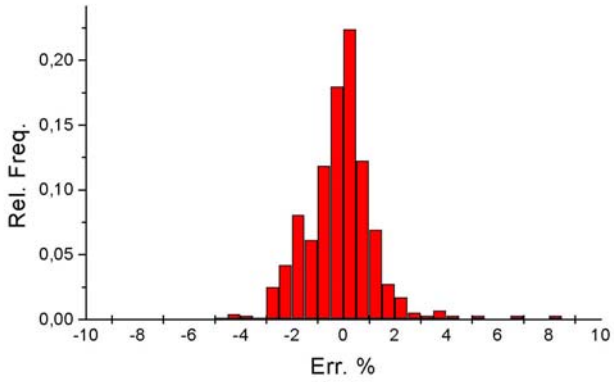

Fig. 3. Error in activity for individual nodes in circuit C1

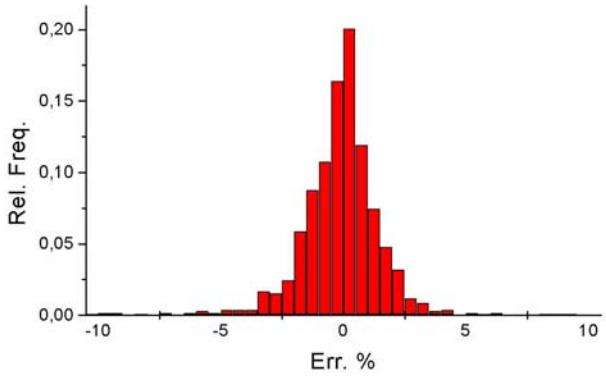

Fig. 4. Error in activity for individual nodes in circuit C2

In order to check if the error is within the specified values, several simulations were run with 95% confidence, 5% error and 0.35 minimum mean. Fig. 3 and 4 show the relative error on the average number of transitions per clock cycle for individual nodes, compared with the corresponding long simulation values. It is observed that

more than 95% of the nodes is within the 5% error, while the nodes with the highest relative errors are also low activity nodes, with negligible absolute error. For both test circuits, more than 98% of the nodes have an error less than 5%. In fact, more than 86% of the nodes have an error less than 2%. This is due to the highest activity nodes, which converge earlier in the estimation process, and are over-analyzed.

The tunable accuracy-execution time properties of the technique implemented in this work was studied in a second test. Fig. 5 shows the results of different runs with 95% confidence and 5% error, varying the minimum mean. For all the experiments mentioned in this paragraph, the input signals were specified as time independent, with probability 0.5 and 0.5 transitions per clock cycle. As expected, the required number of samples monotonically increases as the minimum mean decreases. Nevertheless, a low impact on the estimation error is observed, as stated in [14]. In fact, the correlation between the minimum mean and the relative error is very low as shown in Table 3.

Table 3. Correlation between minimum mean and error for individual nodes. The selected nodes are the most active (named C1H), the 10^{th} most active (C1M), and a low regular- but near the threshold- active node (C1L). Values extracted from simulations of C1

Node	Correlation
C1H	0.3405
C1M	0.4402
C1L	0.1262

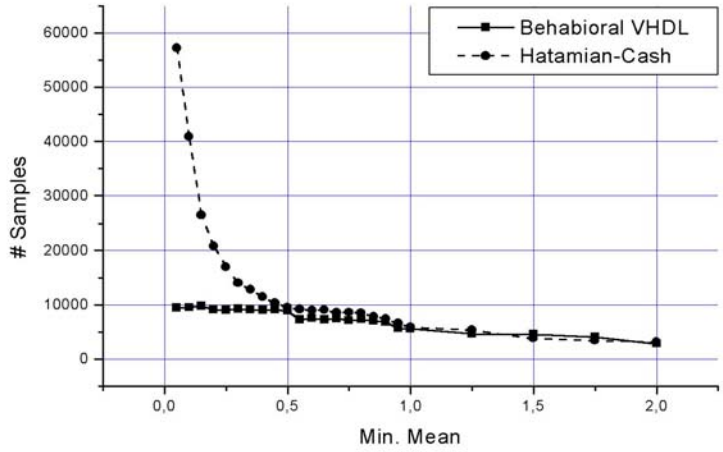

Fig. 5. Tunable property for the user defined threshold in nodes activity

The same tunable property has been checked with respect to the pair error-confidence: as error decreases and confidence level increases, the number of samples monotonically increases (Fig. 6).

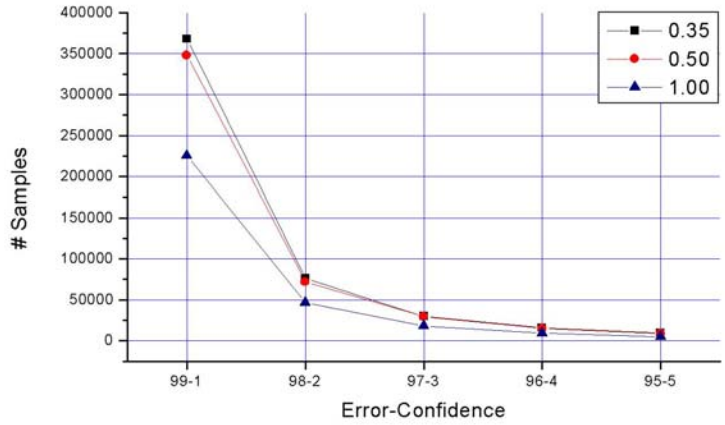

Fig. 6. Error-Confidence tunable property. Values extracted from simulations of C1. Inputs defined as independent

In order to show how the activity can fluctuate in both individual nodes and the whole circuit, the cases defined in Table 4 have been simulated. The different execution times are shown in Fig. 7. It has also been verified the variation in the activities for individual nodes as shown in Table 5.

Table 4. User defined input patterns.

	Input Pattern
1	All primary inputs are independent random patterns
2	The activity of the MSB is set to 0.05 and is increased linearly to 0.95 for the LSB
3	The activity of the MSB is set to 0.95 and is decreased linearly to 0.05 for the LSB
4	The activity is set to 0.75 for the 4 MSBs and 0.25 for the LSBs
5	The activity is set to 0.25 for the 4 MSBs and 0.75 for the LSBs
6	The signal probability is set to 0.75 for the 4 MSBs and 0.25 for the LSBs
7	The signal probability is set to 0.25 for the 4 MSBs and 0.75 for the LSBs
8	The 4 MSBs of the pattern are independent random patterns and the remaining bits are connected to a counter
9	The 4 MSBs of the pattern are connected to a counter and the remaining bits are independent random patterns

5 Conclusions

A statistical-based power estimation tool oriented to FPGA devices has been presented. The experiments confirm the robustness of the technique, allowing a

tunable accuracy. According to the precision required at each moment in the design process, appropriate values can be set for both the minimum mean activity and the error-confidence pair. The execution time of the tool grows with the required precision. For instance, running the tool on the C1 multiplier, and selecting 1 as the minimum mean, the execution times are 20 minutes for 95% confidence and 5% error, and 60 minutes for 97% confidence and 3% error, using a PC with a 1-GHz AMD processor with 256 MB RAM memory.

It has also been verified that the actual relative error for individual nodes is bounded by the one specified by the user. As predicted by Eq.2, nodes with higher activity have less error than the specified one: they converge earlier in the estimation process and are over-analyzed according to the specified tolerated error.

Finally, the importance of properly defined input pattern characteristics is pointed out. The use of this tool with a default or arbitrary input pattern, can result in an activity figure with unpredictable error.

Table 5. Maximum variation on activity for individual nodes and the different user specified input patterns. The shown values are extracted from simulations of C1

C1H		C1M		C1L	
#Pattern	Activity	#Pattern	Activity	#Pattern	Activity
3	6.7032	7	0.7916	8	0.5178
6	4.7383	6	4.5503	6	0.0958

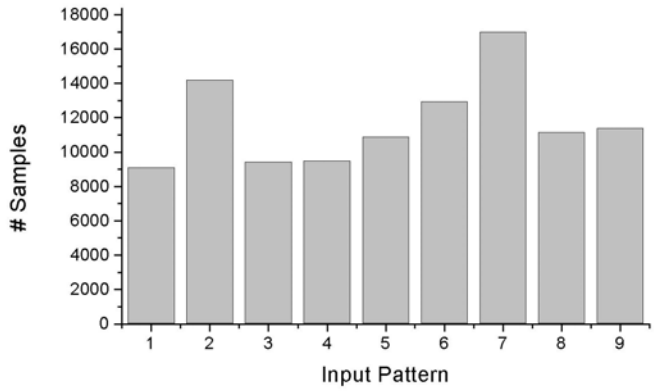

Fig. 7. Execution time for several user defined input patterns. Values extracted from simulations of C1

Acknowledgments

Spanish Ministry of Science and Technology has supported this work, under Contract TIC2001-2688-C03-03. Additional funds have been obtained from Project 658001 of

the *Fundación General de la Universidad Autónoma de Madrid*. G. Sutter and E. Todorovich are granted by CONICET of Argentine.

References

1. Boemo, E., Gonzalez de Rivera, G., Lopez-Buedo, S., Meneses, J.: Some Notes on Power Management on FPGAs. Lecture Notes in Computer Science, No. *975,* Springer-Verlag, Berlin (1995) 149-157
2. Najm, F.: Estimating Power Dissipation in VLSI Circuits. IEEE Circuits and Devices Magazine, Vol 10, No 4 (1994) 11-19
3. Pedram, M.: Design technologies for Low Power VLSI. In Encyclopedia of Computer Science and Technology, Vol. 36, Marcel Dekker, Inc. (1997) 73-96
4. Macii, E., Pedram, M., Somenzi, F.: High-Level Power Modeling, Estimation, and Optimization. Computer-Aided Design of Integrated Circuits and Systems (1998)
5. Fawcett, B.: FPGAs, Power and Packages. *XCELL* (1997)
6. Xilinx Press: A Simple Method of Estimating Power in XC4000XL/EX/E FPGAs. Application Brief, XBRF 014 (1997)
7. Tan, J: Virtex Power Estimator User Guide. XAPP 152 (1999)
8. Xilinx Inc.: XC4000XL Power Calculation. XCELL, Nº27 (2000) pp 29
9. Osmulski, T., Muehring, J.T., Veale, B., West, J. M., Li, H., Vanichayobon, S., Ko, S-H, Antonio, J.K, Dhall, S.K: A Probabilistic Power Prediction Tool for the Xilinx 4000-Series FPGA. Proc. of the 5th International Workshop on Embedded/Distributed HPC Systems and Applications (EHPC 2000), Cancun, Mexico (2000) 776-783
10. Xilinx Inc.: ISE 4 User Guide. http://www.xilinx.com
11. Xilinx Inc.: XPower Tutorial: FPGA Design, XPower (v1.1). (2001) http://www.xilinx.com.
12. Burch, R., Najm, F. N., Yang, P., Trick, T.: A Monte Carlo approach for power estimation. IEEE Transactions on VLSI Systems, 1(1) (1993) 63–71
13. Xakellis, M., Najm, F.: Statistical Estimation of the Switching Activity in Digital Circuits. 31st ACM/IEEE Design Automation Conference, San Diego, CA (1994) 728-733
14. Najm, F. N., Xakellis, M. G.: Statistical estimation of the switching activity in VLSI circuits. VLSI Design, vol. 7, no. 3 (1998) 243-254
15. Chou, T., Roy, K.: Accurate Power Estimation of CMOS Sequential Circuits. IEEE Trans. on VLSI, Vol.4, nº3 (1996) 369-380
16. Kozhaya, J., Najm, F. N.: Accurate power estimation for large sequential circuits. IEEE/ACM International Conference on Computer-Aided Design (1997) 488-493
17. See On-line documentation at http://www.aldec.com
18. P1497 DRAFT Standard for Standard Delay Format (SDF) for the Electronic Design Process. IEEE SDF P1497 Draft 0.10 Specification, June 7, 2000, http://www.eda.org/sdf/