# A Microarchitectural-Level Step-Power Analysis Tool*

Wael El-Essawy, David H. Albonesi
Electrical and Computer Engineering
University of Rochester
{essawy, albonesi}@ece.rochester.edu

Balaram Sinharoy
Advanced Microprocessor Design
IBM Corporation
balaram@us.ibm.com

## ABSTRACT

Clock gating is an effective means for reducing average power consumption. However, clock gating can exacerbate maximum cycle-to-cycle current swings, or the step-power (*Ldi/dt*) problem. We present a microarchitecture-level step-power simulator and demonstrate its use in exploring how design alternatives impact relative step-power levels. We show how the tool can be used to identify major sources of high microprocessor step-power events. Our experiments indicate that branch mispredictions are a major cause of high step-power occurrences. We also show that high step-power events are infrequent which suggest that architectural techniques may limit step-power at potentially low performance cost.

## Categories and Subject Descriptors

C.5.3 [**Computer Systems Organization**]: COMPUTER SYSTEM IMPLEMENTATIONMicrocomputers; I.6.5 [**Computing Methodology**]: SIMULATION AND MODELINGModel Development

## General Terms

Reliability Design

## Keywords

step-power, Ldi/dt, inductive noise, microprocessors, clock-gating, architectural simulation

## 1. INTRODUCTION

Higher transistor densities, and higher frequencies have led to ever-increasing levels of microprocessors power consumption. Despite attempts to limit power consumption via lowering the supply voltage, microprocessor power consumption has continued to increase. This leads to rapidly increasing levels of chip-level *current* consumption.

High frequencies demand a complex power delivery network and resulting high inductance interconnect. These two characteristics of high performance microprocessors, high levels of current consumption, and high inductance in the power delivery network and I/O pads, leads to reliability concerns due to *inductive noise*, also referred to as *ground bounce*, the *Ldi/dt*, or the *step-power* problem. This issue increases the complexity of the power delivery network, and requires more on chip decoupling capacitance [5].

In recent years, significant effort has been expended to augment microarchitecture-level performance simulators with power estimation capability. Among the recently published power-performance simulators are Wattch [2], SimplePower [9], the Cai-Lim model [4], and PowerTimer [1]. However, these power simulators do not address the inductive noise problem in detail. With inductive noise increasing to prohibitive levels, it is essential to have a tool that accounts for chip-level inductive noise.

Very little research has addressed the inductive noise problem at the architectural level in microprocessor design. Pant et al. propose gradual activation/deactivation of functional units by the introduction of waking up, and deactivation time parameters to the clock gating scheme [8]. The smoother current transition of this scheme comes at the cost of both lower performance and higher average power.

Zhanyu et al. [7] modified the technique provided in [8] to reduce the performance loss introduced by gradual clock gating deactivation. The idea is to predict when an instruction is to be issued to the functional unit, and start gradual wake-up prior to the issue, thereby eliminating the wake-up delay. Although the performance degradation is lower, the power overhead was increased when compared to Pant et al.

The above two techniques work at the unit-level, not at the chip-level. In a more recent paper [6], Grochowski et al. propose to reduce ground bounce via a global feedback control system. Grochowski creates an RLC model for the power delivery network, and models the system as an LTI (Linear Time Invarient) system, whose input is the current consumption on the processor, and output is the power distribution voltage. However, the authors' methodology does not provide insight as to the sources of ground bounce at the microarchitecture level. Such insight is crucial to effectively solving the problem.

In this paper, we present a microarchitecture-level framework for studying the step-power problem within the commonly used SimpleScalar and Wattch simulators. Our model, although not as detailed as [6] in modeling the power distribution network, provides insight as to the causes of high step-power within a dynamic superscalar processor. We then use the model to identify such events using several of the SPEC2000 benchmarks.

## 2. THE STEP-POWER SIMULATOR

Our simulator models the amount of current switching on a cycle-by-cycle basis. In a constant supply voltage circuit, power consumption is proportional to the current consumption. To model *di/dt*, we can equivalently model the amount of power switching per cycle *dp/dt*, which we call *step-power*. Although our model does not yet accurately characterize the magnitude of the switching noise, it does serve to identify the causes of the highest step-power events. It therefore serves as a microarchitecture-level aid to controlling large on-chip current swings. We built our simulator on top of the Wattch [2] power simulation extensions to the SimpleScalar toolset [3]. We used Wattch's cc3 clock-gating power estimation

*This research was supported in part by DARPA/ITO under AFRL contract F29601-00-K-0182, NSF under grants CCR-9701915 and CCR-9811929, by an IBM Faculty Partnership Award and an IBM Research Fellowship Award.


*ISLPED'02*, August 12-14, 2002, Monterey, California, USA.

**Table 1: Benchmarks**

| Bench-mark | Dataset | F-Fwd Inst. | Simulation window | |
|---|---|---|---|---|
| | | | Inst. | Cycles |
| gcc | scilab | 1000 | 562 - 890 | 500 |
| mcf | ref | 4000 | 1625-1738 | 500 |
| parser | ref | 250 | 593 - 822 | 500 |
| gzip | program | 40 | 823 -1423 | 500 |
| art | ref | 2900 | 494 - 565 | 500 |
| swim | ref | 250 | 1267-1364 | 500 |
| applu | ref | 500 | 1003-1634 | 500 |
| apsi | ref | 30 | 378 - 395 | 200 |

model [2]. Cc3 assumes linearly scaled power depending on usage when accessed, and 10% of base power when the structure is not accessed.

Wattch does not provide a model with no clock gating. We created a model (named cc0) that assumes no clock gating in processor structures. The main difference between the cc0 and cc3 models is the way the clock tree is estimated. While cc3 scales the clock tree power with the amount of clock gating that takes place on the processor in any cycle, cc0 always assumes the full clock power is consumed every cycle, irrespective of how much activity is taking place on the processor chip. This is intuitive, since no attempt is made to shut off the clock in this model. Another difference is that we assumed an idle factor for representing the ratio of power consumed in the combinational logic when idle. We assumed the idle factor in our simulations to be 20%.

We also modified the clock power model by separating it into two different components: local and global clock distribution. The global clock distribution network is that part of the clock distribution tree that remains active when local structures are turned off. This includes the trunk of the (H) clock tree and global buffers (repeaters) within this tree, and the clock generator (not simulated in Wattch). The local component of the clock tree includes the load capacitance of processor structures in all different stages. Wattch scales the whole clock power with the power consumed in the processor structure. We modified it in order to scale the local clock power component with the current cycle activity of all the processor structures.

Our simulator is capable of dynamically tracking the power and step-power levels over time. The simulator tracks the maximum power, the average power, the minimum power, the average step-power, and the maximum step-power over a specific window of simulation cycles (*resolution period*). The dynamic power and step-power curves provide a visual means to understanding the time varying behavior of power and step-power. In order to determine the distribution of power and step-power levels, we create power and step-power histograms. These histograms report the frequency of the occurrence of particular power and step-power values. We set the ranges to be at one-Watt slots, but the resolution of the histograms can be easily varied.

# 3. SIMULATION METHODOLOGY

The simulation runs to show the use of this tool were performed using applications from the SPEC2000 benchmark suite using the reference data sets. Simulations were run for a duration of 500 million cycles for all the benchmarks except apsi which took about 200 million cycles to run the entire program. During the simulations, we skipped the initialization part of each benchmark. Table 1 specifies the benchmarks used along with input datasets, number of instructions skipped, and the simulation window in terms of both number of instructions and number of cycles simulated. The number of simulated instructions in the simulation window varied with different configurations for the same number of simulated cycles. Table 2 shows the baseline simulation parameters.

**Table 2: Simulated Processor Configuration**

| | |
|---|---|
| Branch Mispredict Penalty | 7 |
| Decode Width | 4 |
| Issue Width | 6 |
| Retire Width | 11 |
| Memory Ports | 2 |
| L1 Data Cache | 64-KB, 2-way set associative |
| L1 Instruction Cache | 64-KB, 2-way set associative |
| L2 Unified Cache | 1-MB direct mapped |
| L1 Cache Latency | 1 cycle |
| L2 Cache Latency | 12 cycles |
| Integer ALUs | 4 |
| Floating Point ALUs | 2 |
| Load/Store Queue Size | 64 entries |
| RUU Size | 80 entries |

# 4. RESULTS

Figure 1 depicts the dynamic values of power consumption and step-power over time for four of the benchmarks. For each application, four graphs are presented. The upper graphs present the dynamic power consumption, while the lower ones present dynamic step-power values. The graphs to the left show the results with no clock gating (cc0), while the graphs to the right show the clock gating results (cc3).

Each point in each graph represents the power value over a simulation window of a million cycles. In each of these simulation windows, the maximum, average, and minimum values are calculated. We do not plot the minimum value of step-power as it is zero for all intervals.

We draw several conclusions from these graphs. First, although clock gating achieves considerable average power savings across all applications, it has little effect on the maximum power consumption. Figure 2 depicts the percentage savings in average and maximum power for the eight simulated benchmarks. While the average power savings is in the range of 24% (applu) to 52% (mcf), with the exception of apsi, in which the savings is 29%, the maximum power savings is only in the range of 1% to 6%. The reason for this disparity is that while the average power savings reflects the common case of clock gating idle structures, the maximum power reflects the worst case scenario in which most of the processor structures are busy, with no opportunity to clock gate them. Apsi has a regular periodic pattern of power consumption across all processor structures due to its small basic block loops. The application fits into the instruction and data caches with almost no misses, and near perfect branch prediction. This regularity makes the worst case very similar to the average case in which considerable power savings can be achieved by clock gating.

Second, because clock gating has little effect on the maximum power, but a large effect on the average and minimum power, it increases the likelihood of high step-power events. Figure 3 presents the relative increase in average and maximum step-power due to clock gating. As expected, there is a considerable increase in the step-power due to clock gating. On average a 35% increase in average step-power and a 37% increase in maximum step-power is incurred with clock gating.

Step-power histograms are presented in Figure 4. In this figure, the x-axis represents the step-power value, and the y-axis the normalized cumulative frequency of occurrence. The curves clearly indicate that the highest step-power values rarely occur. For example, while gzip has maximum step-power values of 102W and 73W for cc3 and cc0, respectively, the step-power is larger than 42W (31W) for cc3 (cc0) for only 1% of the simulated time. In other words, the step-power does not exceed 42% of the maximum step-power for gzip for 99% of the execution cycles. The graphs also show that in 90% of the cases the step-power is less than 23W and 16W for cc3 and cc0, respectively. As shown in the figure, the rest

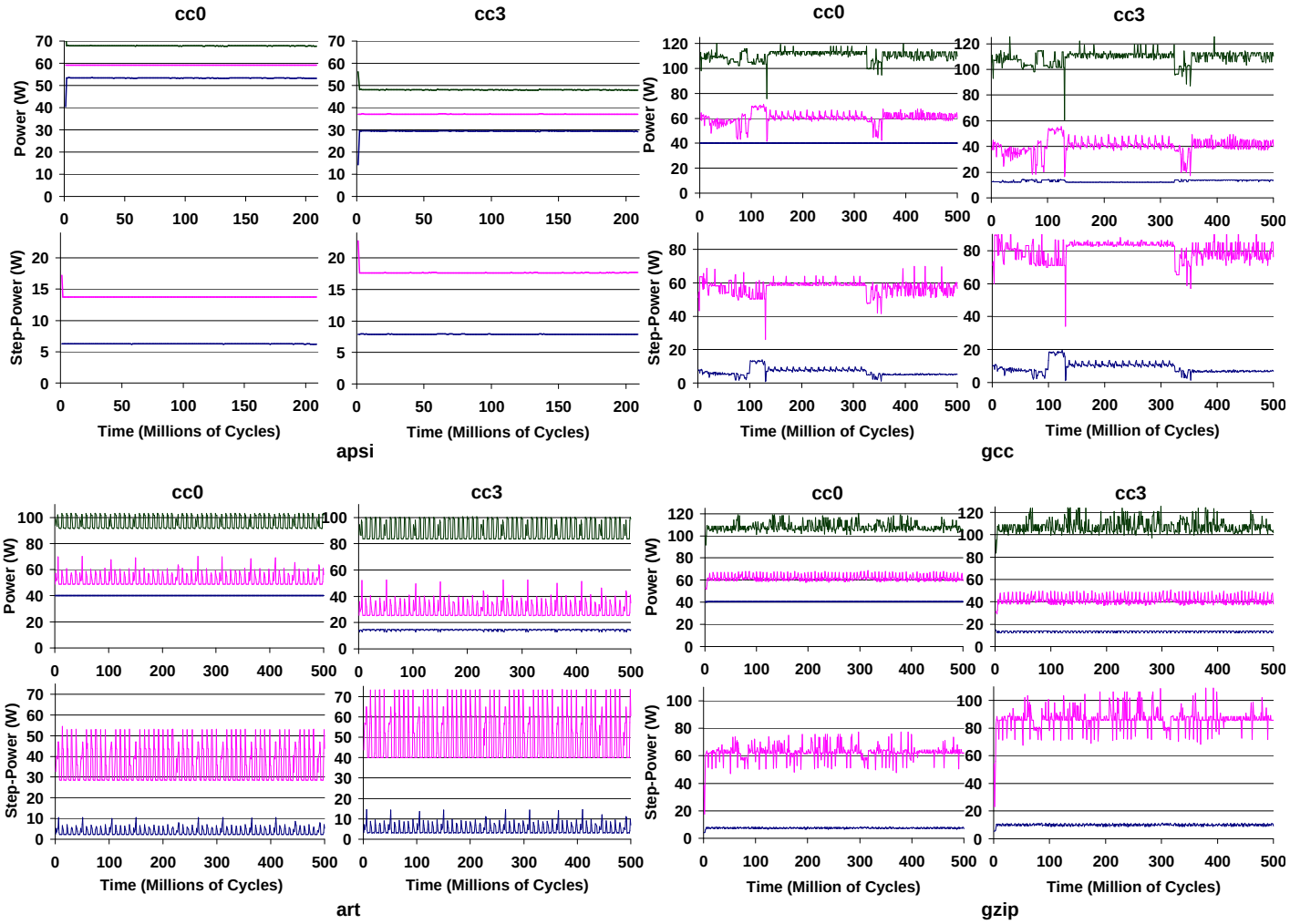


**Figure 1: Dynamic Power and Step-Power Profiles for Four of the Simulated Benchmarks**

of applications show similar trends. This indicates the potential for microarchitectural techniques that can eliminate high step-power events potentially with little performance penalty.

High step-power events occur when a large power consumption increase (or decrease) occurs across many processor structures in the same cycle. By analyzing the infrequent high step-power values through program dumps, we found two common events associated with high step-power: data cache access variation and branch mispredictions. Although branch mispredictions and cache access variation alone do not necesarily cause high step-power, these events were detected in most of the high step power cycles. Increasing the number of ports linearly increases the power consumption of the data cache, and also increases the step-power of the cache. For example, a high step-power event will occur when all of the cache ports are accessed in one cycle and none are used in the next (or previous) cycle. While the base architecture has two ports (as shown in Table 2), we also ran simulations with one, four, and six port. The results, shown in Figure 5 for gcc, demonstrate a clear correlation between the number of memory ports and the step-power. Table 3 lists the average power per cycle, average power per committed instruction, maximum step-power, and the IPC statistics for both cc0 and cc3 when running gcc on machine configurations with different numbers of memory ports. If we rely on the data provided by Wattch (cc3 average power numbers), then the six and four ports configurations are the best choice in terms of performance, and power efficiency (power per instruction), respectively. However, different conclusions can be reached if we consider a no clock gating design, or if we consider inductive noise. In the cc0 model, the base architecture with two ports is the most power efficient system. This is because in such a system the clock power in the data cache and load/store queues is not saved when the corresponding structures are idle. And considering the maximum step-power, we find that increasing the number of memory ports from two to four, and to six increases the step-power by 14% and 31%, respectively, while increasing the IPC by about 5.9% for both cases. Reducing the number of ports to only one lowers the maximum step-power by 16% with a 27% loss of IPC.

Our analysis of the step-power dumps indicates that the highest step-power events are associated with branch mispredictions. Branch mispredictions have a dual effect on the step-power. First,

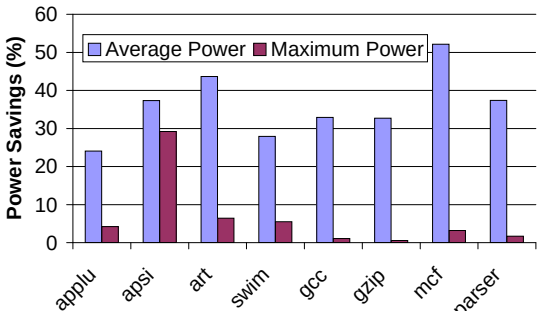


**Figure 2: Power Savings Through Clock Gating**

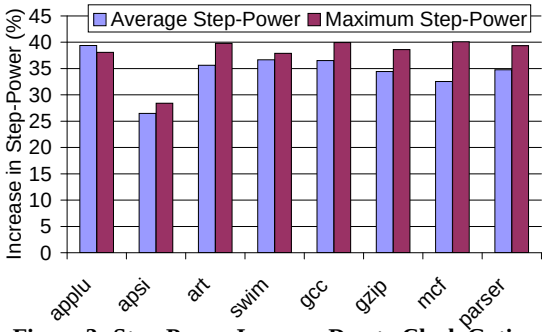


**Figure 3: Step-Power Increase Due to Clock Gating**

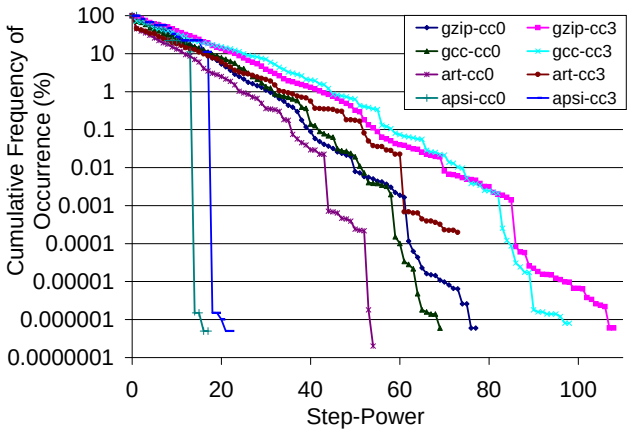


**Figure 4: Step-Power Histograms**

it takes a few cycles to resume fetching instructions from the instruction cache after a mispredict, so this causes a sudden drop in the instruction fetch unit power. Second, the flushing of instructions from the mispredicted path causes many processor units to suddenly become idle, resulting in a large drop in power consumption. In order to show the two effects separately, we simulated two configurations, *nolat* and *perfect*. In the nolat configuration, the processor starts fetching new instructions in the cycle following the detection of a branch misprediction. This limits the branch misprediction effect to draining instructions from processor queues, but does not halt the instruction fetch unit. On the other hand, the perfect configuration has a perfect branch predictor, which eliminates both instructions draining from processor structures, and instruction queue halts.

Figure 6 presents results for the two configurations running the gzip application. From this figure we find that the nolat configuration reduces maximum step-power by 7.8% and 8.3% for cc0 and cc3, respectively. On the other hand, perfect branch predic-

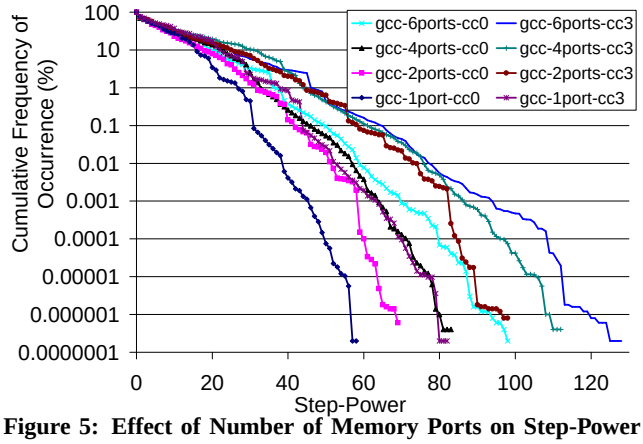


**Figure 5: Effect of Number of Memory Ports on Step-Power for gcc**

**Table 3: Power and Performance vs. Number of Memory Ports for gcc**

| Ports | cc0 Power Per: | | Max | IPC | cc3 Power Per: | | Max |
|---|---|---|---|---|---|---|---|
| | Cycle | Inst | Step | | Cycle | Inst | Step |
| 1 | 56.66 | 50.36 | 58 | 1.125 | 36.09 | 32.08 | 82 |
| 2 | 62.78 | 40.61 | 69 | 1.546 | 41.99 | 27.16 | 98 |
| 4 | 67.57 | 51.26 | 83 | 1.6377 | 42.93 | 26.21 | 112 |
| 6 | 72.14 | 44.04 | 98 | 1.6382 | 43.62 | 26.63 | 128 |

tion reduces the maximum step-power by 23.4% and 22.2% for cc0 and cc3, respectively. We conclude that a good technique for step-power reduction must handle branch mispredictions efficiently, so that no sudden drop in power occurs in a single cycle.

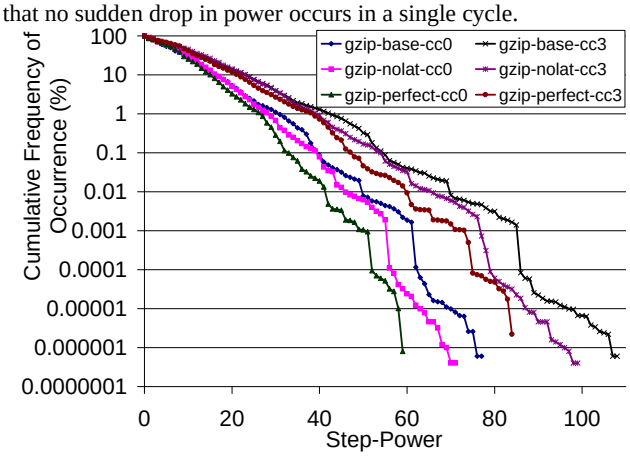


**Figure 6: Effect of Branch Mispredictions on Step-Power for gzip**

# 5. CONCLUSIONS AND FUTURE WORK

We present a microarchitecture-level step-power simulator that allows designers to take step-power into account when making microarchitecture-level tradeoffs. Experiments using our tool indicate that branch mispredictions and cache access variations are major sources of high step-power events. Our experiments also show that such events are highly infrequent which suggests that architectural techniques have the potential to reduce inductive noise at potentially low performance cost. The development of such techniques are part of our future work.

# 6. REFERENCES

[1] D. Brooks, et al. Power-Aware Microarchitecture: Design and Modeling Challenges for Next-Generation Microprocessors. IEEE Micro, 20(6):26–43, Nov/Dec 2000.

[2] D. Brooks, et al. Wattch: A Framework for Architectural-Level Power Analysis and Optimizations. In Proceedings of the 27th ISCA, June 2000.

[3] D. Burger and T. Austin. The Simplescalar Tool Set, Version 2.0. Technical Report CS-TR-97-1342, University of Wisconsin, Madison, Wisconsin, June 1997.

[4] G. Cai and C. Lim. Architectural Level Power/Performance Optimization and Dynamic Power Optimization. In Proc. Cool Chips Tutorial at 32nd ISCA , Nov. 1999.

[5] M. Gowan, et al. Power Consideration in the Design of the Alpha 21264 Microprocessor. In Proceedings of the 35th Design Automation Conference, June 1998.

[6] E. Grochowski, et al. Microarchitectural Simulation and Control of di/dt-induced Power Supply Voltage Variation. In Proceedings of the 8th HPCA , Feb. 2002.

[7] Z. Tang, et al. Ramp Up/Down Floating Point Unit to Reduce Inductive Noise. In Workshop on PACS, in conjunction with the 9th ASPLOS , Nov. 2000.

[8] V. Tiwari, et al. An Architectural Solution for the Inductive Noise Problem due to Clock-Gating. In ISLPED, Aug. 1999.

[9] W. Ye, et al. The Design and use of SimplePower: A Cycle-Accurate Energy Estimation Tool. In Proceedings of the 37th Design Automation Conference, June 2000.