RC 20520 (8/6/96)
Computer Science/Mathematics

# IBM Research Report

## Inaccuracies in Gate-Level Power Estimation

D. Brand and C. Visweswariah

IBM Research Division
T.J. Watson Research Center
Yorktown Heights, New York



IBM Research Division
Almaden · T.J. Watson · Tokyo · Zurich

# Inaccuracies in Gate-Level Power Estimation

Daniel Brand and Chandu Visweswariah
IBM T. J. Watson Research Center, Yorktown Heights, NY, USA

**Abstract**

This paper studies the confidence with which power can be estimated at various levels of design abstraction. We report the results of experiments designed to identify and evaluate the sources of inaccuracies in gate-level power estimation. In particular, we are interested in power estimation during logic synthesis. Factors that may invalidate or diminish the accuracy of power estimates include optimization, technology mapping, transistor sizing, placement and wiring, and choice of input stimuli.

## 1 Introduction

Power dissipation of integrated circuits has become a topic of great concern recently for two reasons. One reason is to protect circuitry from peak current surges in order to ensure reliability. The other is to minimize energy consumption, especially in battery-operated devices. This paper is concerned only with estimating the average power of a circuit, as opposed to peak power.

We consider the problem of power estimation for two purposes. First, designers need to estimate the power of various design alternatives. Second, power estimation is needed in the inner loop of design tools which attempt to minimize power consumption. The difference between the two is the size of the circuit for which we need the estimate. In the first context we need to estimate the power of circuits typically consisting of a large number of gates. In the latter context, tools need to evaluate alternatives involving many fewer gates. For example, a power optimization tool may consider as a "move" the substitution of one type of gate for another in its quest to save power.

There have been numerous reports in the literature on power estimation and power optimization methods [1, 2, 3, 4, 5, 6, 7]. Power estimation is carried out at different stages of the design process, e.g., at the design language level, during logic synthesis, before and after placement and wiring. Transformations or design decisions made at later implementation stages often invalidate power estimates made at earlier stages of the design; hence there is a need to understand the degree of inaccuracy of power estimates at various implementation stages. The degree of inaccuracy at each stage will allow us to infer a confidence level with which estimates can be made before lower-level details are available.

The most accurate power estimation is on the actual hardware. It is still only an estimate of the power the chip might dissipate with different input stimuli, different operating conditions or different manufacturing parameters. Understanding the impact of manufacturing variations on power dissipation is a difficult and important problem. However, it is not addressed in this paper.

Power estimates less accurate than hardware measurements can be obtained by a circuit simulator, e.g., [1, 7, 8, 9]. The authors of [7] report on two experiments, where PowerMill was compared with hardware measurements. They report relative errors of 1.7% and 7.6%, respectively. Based on our experience with circuit simulation for delay, we do not expect power estimates of circuit simulators to be better than $\pm 5\%$, particularly when simplified device models are used in order to speed up the simulation ([7, 9]).

This paper is not concerned with the accuracy of power estimation by measuring hardware or by circuit simulators, but with the accuracy of gate-level estimation. Gate-level estimators are needed because they are faster than circuit simulators, they can handle larger circuits and are applicable before all the circuit details are known. In order to calibrate our gate-level simulator, we will be comparing its results against those of the circuit simulator described in [9].

Our gate-level power estimator was written specifically to study its accuracy compared to that of a circuit simulator as well as the inaccuracies encountered at higher levels of the design process. We do not suggest that it is a practical power estimator and will describe it only to explain how it relates to circuit simulation.

We assume a design process that can be divided into the following stages.

1) Source description (e.g., VHDL) generated by a designer.

2) Logic synthesis, where we will consider the following steps:

2.1) Technology-independent optimization.

2.2) Technology mapping.

2.3) Repowering and fanout adjustments.

3) Placement and wiring.

4) Manufacturing.

After each stage, we would like to know how much energy would be consumed if all the subsequent stages were completed and the resulting hardware were run on a given sequence of inputs from a given starting state. We assume that the methods 2), 3), 4) used in the design stages are fixed, but their exact impact on the final hardware differs from design to design. Similarly, we assume that the sequence of inputs and the initial state are fixed, but we do not know them *a priori*. We can only collect statistics about the impact of the design stages as well as of different input stimuli, and then use these statistics to predict energy consumed by other designs, or due to other input sequences.

We consider the following overall causes of uncertainty in power estimation:

- Technology-independent optimization
- Technology mapping
- Repowering and fanout adjustments
- Placement and wiring
- Changes in input stimuli
- Gate-level vs circuit-level inaccuracies.

"Gate-level vs circuit-level" is on the above list together with changes to design or input stimuli, because using a gate-level estimator introduces an error by itself (Section 3). After Section 3 quantifies the error incurred in using a gate-level estimator, the subsequent sections evaluate other sources of inaccuracies based on the a gate-level simulator only. Section 4 reports on inaccuracies in power estimates due to input stimuli variations. Section 5 compares estimates made during logic synthesis with estimates made after placement and wiring. While Section 5 is concerned with power estimation for a whole partition consisting of many gates, Section 6 evaluates the accuracy and confidence of predicting whether one gate consumes more power than another.

# 2 Overview of experimentation

We have evaluated the accuracy of gate-level power estimation on the benchmarks shown in Table 1. Some of them are public benchmarks [10] (e.g., C432), and the rest are our internal designs, including microprocessor partitions as well as ASICs.

The benchmarks contain combinational gates and latches; none of them contains any off-chip drivers, busses or memories, which are actually more significant consumers of power. While our benchmarks do contain clocks, all our power estimates ignore power dissipated by clock trees for two reasons. First, the designs did not contain the actual clock trees, and second, power dissipated due to clocks is much more predictable than power dissipated by random logic. However, any power due to gating of clocks is included in our statistics.

For each benchmark we consider gate networks obtained at the following stages of design:

0) Before any optimization

1) After technology-independent optimization, but before technology mapping

2) After technology mapping, but before repowering

3) After repowering and fanout adjustment, but before physical design

4) After placement and wiring.

The column "gate count" gives the number of gates in stages 3 and 4.

| Name | Gate count | Primary inputs | Primary outputs | Register bits |
|---|---|---|---|---|
| B1 | 85 | 22 | 6 | 18 |
| B2 | 87 | 12 | 33 | 0 |
| B3 | 109 | 134 | 1 | 0 |
| C432 | 119 | 36 | 7 | 0 |
| B5 | 150 | 60 | 14 | 6 |
| C880 | 179 | 60 | 26 | 0 |
| B7 | 184 | 42 | 16 | 16 |
| C499 | 204 | 41 | 32 | 0 |
| C1908 | 212 | 33 | 25 | 0 |
| C1355 | 242 | 41 | 32 | 0 |
| B11 | 265 | 38 | 223 | 0 |
| C2670 | 280 | 233 | 140 | 0 |
| B13 | 330 | 124 | 42 | 83 |
| B14 | 424 | 84 | 18 | 17 |
| B15 | 425 | 38 | 30 | 86 |
| C3540 | 523 | 50 | 22 | 0 |
| B17 | 678 | 100 | 35 | 35 |
| B18 | 690 | 70 | 83 | 32 |
| B19 | 757 | 134 | 69 | 134 |
| B20 | 864 | 86 | 66 | 159 |
| B21 | 1373 | 176 | 263 | 235 |
| B22 | 1446 | 104 | 49 | 241 |
| B23 | 3057 | 30 | 83 | 1095 |

Table 1: Benchmarks used

The gate networks at stages 0, 1, 2, and 3 were obtained using the logic synthesis system described in [11]. Wire capacitances were estimates based on an average capacitance per fanout. This average capacitance was provided to us as part of technology support, and we did not change it. In order to evaluate the inaccuracies in power calculation due to estimated capacitances, we consider the network at stage 4. It is topologically identical to the one at stage 3, but nets have capacitances resulting from placement and wiring. However, we did not actually perform placement and wiring; instead we simulated their effect by setting net capacitances as described below.

Following [12], we assume that after placement and wiring, wire lengths follow a Weibull distribution with density function $f(x) = \alpha\beta x^{\beta-1}e^{-\alpha x^{\beta}}$. Each wire capacitance was randomly changed so that all the wire capacitances follow the Weibull distribution with $\beta = 0.5$ and with $\alpha$ calculated differently for each benchmark. We chose $\alpha$ so that the resulting average capacitance was as close as possible to the average capacitance used in the network in stage 3.

Of course, there is no guarantee that the result of actual placement and wiring will yield such an average capacitance. In general, placement and wiring has the effect of changing the average capacitance, as well as randomizing net-to-net capacitances. By merely randomizing the capacitances instead of actually performing placement and wiring we can study the effect of the randomization independently from changing the average capacitance. The impact of changing the average capacitance on power estimation is much more predictable.

# 3 Circuit-level vs gate-level

The purpose of this section is to evaluate the inaccuracies incurred by using a gate-level simulator as opposed to a circuit simulator. After understanding the gate-level inaccuracies, subsequent sections will evaluate other causes of inaccuracies using gate-level simulation exclusively. The comparisons in this section will be against the circuit simulator [9] on the identical circuit with identical wire capacitances and identical input patterns. For that purpose we will briefly describe the gate-level simulator and point out the causes of its inaccuracies vis-a-vis a circuit simulator.

Our gate-level estimator is an event-driven timing simulator written specifically for the purpose of performing this evaluation. For simplicity it assumes two-phase master-slave latches. Clocks may be gated, and multi-phase clocks are allowed, but their periods must be multiples of the period at which primary inputs arrive. Each simulation cycle starts with primary inputs receiving new values (all simultaneously), and slave latches receiving their values from their master. These values are then propagated through the combinational logic using a time wheel. Delay and slew on individual gates are obtained from a static delay calculator. At the end of the cycle, new values are latched into masters (if the corresponding clock becomes active).

The inaccuracies of gate-level estimation for static CMOS can be attributed to the following simplifying assumptions:

a) Transitions are digital rather than analog.

b) Power dissipation is assumed to be due to charging and discharging of capacitors only.

c) Internal gate structure is unknown to the gate-level simulator.

d) Gate delays are pre-computed in a characterization phase.

Assumption (a) states that each capacitor either completely charges or completely discharges, while in reality incomplete transitions occur. Voltage on a node may exhibit a small glitch, smaller than a full swing between $V_{dd}$ and Ground. The energy consumed during such a glitch is non-zero, but smaller than the energy consumed during two complete transitions. A gate-level simulator must represent such a small glitch either by a full one, in which case it will overestimate the energy, or by none, in which case it will underestimate the energy. Furthermore, a small glitch attenuates as it passes through subsequent stages of logic [13]. A gate-level simulator must either propagate a glitch as two full transitions, or not propagate it at all.

Due to assumption (b), the gate-level simulator ignores short-circuit current, leakage current and other effects that are typically considered to be insignificant power contributors [14]. Thus the gate-level simulator assumes that energy is consumed only when a capacitance $C$ switches from voltage 0 to $V_{dd}$ (or vice versa), and in that case consumes energy equal to $C{V_{dd}}^2/2$. (In reality, the MOSFETs are the dissipative elements that consume power, whereas the capacitors just store and release energy. However, in CMOS circuits, this energy can be shown to be equal to the energy required to charge or discharge the capacitors while switching the gate.)

Assumption (c) above stems from the fact that internal gate capacitances are unknown to the gate-level estimator, while they are known to the circuit simulator.

Assumption (d) concerns gate delay and slew. The circuit simulator computes this information dynamically based on actual analog waveforms. The gate-level simulator obtains the information from pre-characterized delay equations, which are independent of the actual input waveforms. This timing model, created for static timing analysis, is part of the technology library, and it is based on the significant assumption that only one input switches at a time. Even if the assumption holds, the delay and slew values are only approximations. But in the case of simultaneous gate input switching, the delay information is quite unreliable.

We try to compensate for simplifications (a) and (c), but make no compensation for assumptions (b) and (d). To compensate for simplification (a), we attempt to predict whether glitches on gate outputs will occur and whether they will have a full swing between $V_{dd}$ and Ground.

We distinguish between three types of glitches. A "healthy" glitch is one in which a signal has the time to complete a full transition before it is forced back to its starting value. "Healthy" glitches are represented in the gate-level simulator as two normal transitions, and they propagate through the rest of logic in the same manner as all other transitions.

The second kind of glitch arises when the inputs of a gate undergo staggered transitions. One input changes, causing a transition on the output. After the input completes its transition, *but before the output has a chance to complete its transition*, another input changes, forcing the output back to its initial value. In the gate-level simulator these kinds of glitches are detected by using slew information on the gate's output to calculate whether the signal had a chance to complete its transition before being forced back. If the answer is "yes" then this is a "healthy" glitch as described above. But if the answer is "no" then the glitch is completely ignored – it causes no power dissipation in the gate-level simulator and is not propagated.

The third kind of glitch is due to simultaneous switching. One gate input undergoes a transition, causing a transition on the output. But in contrast to the second kind of glitch, the other input starts switching *before the first input completes its transition.* These kinds of glitches due to simultaneous switching are detected by the gate-level simulator. One half of them, picked at random, are represented by two full transitions for power accounting purposes. The other half are ignored. In both cases the glitch is not propagated. Thus an estimate of power due to partial glitches, albeit crude, is accounted for in the total power.

Neither the second nor the third category of glitching is "healthy", but these types of glitches are treated differently for several reasons. The second kind of glitch is often caused by overloading a gate output and is less common than the third kind of glitch (due to simultaneous switching of gate inputs). Some glitches due to simultaneous switching turn out in reality to be completely "healthy", some will not even cause a ripple on the gate output, and some will be manifested as a small glitch reaching an intermediate voltage level. The decision to let just half of them contribute to power dissipation was reached by trying to match the power reported by the gate-level simulator to that reported by the circuit simulator. In the case of simultaneous switching, we could not use slew information to calculate whether the glitch is "healthy" or not because delay calculators do not give meaningful delay and slew data in the case of simultaneous switching.

To compensate for assumption (c), we associate two constants with each gate. One constant, the "output capacitance" is the amount of *internal* gate capacitance that is charged or discharged whenever a gate *output* has a transition. The other, "input capacitance", is the amount of *internal* capacitance that is charged or discharged whenever a gate *input* has a transition, independently of what happens at the gate output. If our goal were an accurate power estimator, the input and output capacitance numbers would be different for each library cell, and even that would be a crude approximation. But we make an even cruder approximation by using just one input capacitance value and one output capacitance value for all cells in a library. The reason for such a simplification is that our goal is to evaluate inaccuracies of power estimation at higher design stages, in particular before technology mapping. Thus our crude estimate of input and output capacitance allows us to evaluate the impact of not knowing the structure of the library cells when estimating power before technology mapping.

Latches are treated differently from combinational gates for two reasons. First they tend to have much more internal capacitance than combinational gates. (While about 50% of power consumed by combinational gates is due to internal capacitances, for latches it is about 90%.) Second, in contrast to combinational logic, latches are more predictable, even before logic synthesis is carried out. Although before logic synthesis we may not know the actual latch library cells, or clock gating, we know the number of latches and their functionality. (This is in contrast to combinational logic, whose implementation is much less predictable at a high-level of abstraction.) Since the state of a latch is completely predictable at all stages of logic synthesis, we can have a different power estimate of a latch depending on state changes. As a result, in addition to the input and output capacitances, we have four more constants for latches (common for all latch cells in a library) – the amount of capacitance charged or discharged whenever:

(i) A master clock toggles, and the master changes state,

(ii) A master clock toggles, but the master does not change state,

(iii) A slave clock toggles, and the slave changes state, and

(iv) A slave clock toggles, but the slave does not change state.

In addition to these four constants we also use the same input capacitance and output capacitance values as for combinational gates.

As can be seen, we try to compensate for assumptions (a) and (c) by adjusting the gate-level simulation with seven factors (input capacitance, output capacitance, the four constants for latch power estimation, and the factor of one half for glitches due to simultaneous switching). All these factors were determined empirically by analyzing circuit simulation results for various kinds of gates. These factors depend not only on our particular technology, but also on the frequency with which our technology mapping uses various library cells. These seven factors were set with the goal of making the gate-level simulator report power numbers close to the the circuit simulator's report. That implies, for example, that our gate-level simulator tends to underestimate the power of complex gates, which have more internal capacitance than the library average, and it tends to overestimate the power of simple gates, which have less than average internal capacitance. This point will be discussed in the context of Table 4.

Table 2 shows energy reported by the gate-level simulator after running some of our smaller benchmarks for 100 cycles. All latches were initialized to zero and primary inputs were generated randomly with a probability of 0.5 for each logic value. The results in Table 2 are normalized to a circuit simulator energy estimate of 100; the column "gate-level estimate" is therefore an indication of the relative error. Bold face indicates the largest and smallest entry. Some of the benchmarks were used in calibrating the gate-level simulator against circuit simulator; in other words, the above described seven constants were derived so as to match the results for those benchmarks. Those benchmarks are identified under the column "Used in calibration?". Power for the other benchmarks was estimated only after the calibration was complete.

| Name | Gate-level estimate | Used in calibration? |
|---|---|---|
| B1 | 96 | yes |
| B2 | 114 | yes |
| B3 | 98 | yes |
| C432 | 109 | yes |
| B5 | 117 | yes |
| C880 | 111 | yes |
| B7 | 106 | no |
| C499 | 97 | yes |
| C1908 | 98 | yes |
| C1355 | 97 | yes |
| B11 | **120** | no |
| C2670 | 105 | yes |
| B13 | 96 | no |
| B14 | 104 | no |
| B15 | **95** | no |
| C3540 | 101 | yes |
| B17 | 106 | no |
| B18 | 101 | no |

Table 2: Gate-level simulator vs a circuit simulator

From Table 2 we see that gate-level power estimates are off from -5% to +20%. For individual gates, however, the power estimates are much worse, as might be expected. In order to evaluate these inaccuracies, ideal ammeters were introduced during circuit simulation to monitor the power consumption of each individual gate. Those results were then compared with the power estimated for the same gates by the gate-level simulator.

Table 3 reports the results of that comparison for all the gates in all the benchmarks combined. For each gate, power reported by the circuit simulator is normalized to be 100, so that relative errors are shown in the table. For gates consuming very little power, even a small absolute error becomes a huge relative error (first line). Therefore the second line takes the statistics only over those gates whose power (as measured by the circuit simulator) is at least 1% of average gate power. Similarly the third line is restricted to gates whose power is at least 10% of average, and so on. For that reason the number of gates considered (second column) is decreasing. Ideally the columns "Minimum", "Maximum", and "Average" should all be 100, and "Standard deviation" should be 0. Limiting ourselves to gates consuming more power brings the average closer to 100, but the inaccuracy is still significant. Even if we restrict ourselves to gates consuming more than half of the energy ("Power limit" =

50%) there is a gate estimated to consume no energy ("Minimum" = 0) and there is a gate estimated to consume more than twice the actual energy ("Maximum" = 220).

| Power limit | Number | Minimum | Maximum | Average | Standard deviation |
|---|---|---|---|---|---|
| 0% | 5003 | 0 | 236429 | 364 | 3906 |
| 1% | 4736 | 0 | 1725 | 114 | 56 |
| 10% | 4286 | 0 | 614 | 110 | 38 |
| 25% | 3764 | 0 | 614 | 109 | 33 |
| 50% | 2919 | 0 | 220 | 104 | 27 |

Table 3: Errors in individual gate power averaged over the whole library

Much of the inaccuracy is due to the gate-level estimator treating all library cells the same, rather than using different output capacitances, etc., for different cells. (We would expect more accuracy with more cell-specific information.) We can observe that effect in Table 4, where each line gives the same statistics as Table 3, but only for gates that are *the same library cell.* Table 4 excludes gates that consume less than 1% of average energy (like the second line of Table 3), and also excludes library cells with less than 10 instances. Each line in Table 4 is identified by the cell's function and number of inputs; there are several lines with identical cells because of different power levels.

The variations among different lines of Table 4 are due to treating all cells the same. For example, power for inverting buffers (NOT) tends to be overestimated while power for non-inverting buffers (IDENT) tends to be underestimated. The reason is that IDENTs tend to have more internal capacitance because of being implemented as two inverters, while NOTs tends to have below-average internal capacitance. The variations within lines of Table 4 are due to the other inaccuracies associated with gate-level estimation.

The purpose of this section was to measure the inaccuracy due to using a gate-level estimation (as opposed to a circuit simulator), as gate-level estimation will be the basis of all our subsequent evaluations. For power estimation of a whole partition, Table 2 shows the error ranging from -5% to +20%. If the error were made more symmetrical it would be approximately $\pm$15%. For power estimation of individual gate we do not have any reasonable bound on the size of error; standard deviation of the error is about 50%.

| Cell | Number | Minimum | Maximum | Average | Standard deviation |
|---|---|---|---|---|---|
| AO21 | 23 | 55 | 142 | 108 | 23 |
| AO21 | 22 | 52 | 150 | 90 | 23 |
| AO22 | 63 | 35 | 212 | 100 | 30 |
| AO221 | **10** | 18 | 106 | 85 | 24 |
| AO2222 | 67 | 19 | 193 | 90 | 28 |
| AO33 | 12 | 32 | 125 | 97 | 24 |
| OA21 | **10** | 109 | 146 | 125 | 10 |
| OR2 | 43 | 2 | 220 | 127 | 37 |
| OR2 | 65 | 47 | 1136 | 131 | 127 |
| OR3 | 29 | 17 | 478 | 144 | 75 |
| OR4 | 55 | 2 | 400 | 172 | 89 |
| AND2 | 76 | 101 | 167 | 123 | 13 |
| AND2 | 112 | **0** | 237 | 104 | 36 |
| AND3 | 28 | 47 | 148 | 107 | 23 |
| AND4 | 33 | 23 | 265 | 102 | 45 |
| AND4 | 20 | 85 | 313 | 133 | 46 |
| AOI21 | 87 | 1 | 400 | 115 | 45 |
| AOI22 | 46 | 49 | 122 | 102 | 12 |
| MUX2 | 53 | 36 | 146 | 114 | 19 |
| NOR2 | 55 | **131** | 339 | **175** | 35 |
| NOR2 | 118 | 23 | 643 | 135 | 53 |
| NOR2 | 91 | 55 | 201 | 110 | 22 |
| NOR3 | 91 | 53 | **1725** | 147 | **171** |
| NOR4 | 131 | 12 | 269 | **52** | 33 |
| NOT | 97 | **0** | 237 | 166 | 29 |
| NOT | 287 | **0** | 576 | 137 | 40 |
| NOT | **436** | 16 | 458 | 117 | 27 |
| OAI21 | 50 | 23 | 158 | 115 | 21 |
| OAI22 | 75 | 34 | 194 | 109 | 26 |
| OAI211 | 22 | 19 | **99** | 71 | 22 |
| OAI221 | 33 | 44 | 464 | 103 | 72 |
| OAI31 | 30 | 67 | 520 | 108 | 85 |
| REG | 108 | 59 | 104 | 88 | **8** |
| XOR2 | 24 | 77 | 159 | 106 | 18 |
| XOR3 | 34 | 69 | 108 | 91 | 10 |
| XOR3 | 25 | 67 | 108 | 84 | 11 |
| NAND2 | 86 | 104 | 231 | 171 | 29 |
| NAND2 | 325 | **0** | 840 | 134 | 58 |
| NAND2 | 158 | 2 | 233 | 115 | 29 |
| NAND2 | 24 | 77 | 113 | 93 | 11 |
| NAND3 | 28 | 124 | 254 | 170 | 35 |
| NAND3 | 201 | **0** | 258 | 114 | 28 |
| NAND4 | 154 | **0** | 406 | 160 | 60 |
| NAND4 | 299 | **0** | 185 | 57 | 26 |
| XNOR2 | 21 | 82 | 182 | 125 | 29 |
| XNOR2 | 35 | 67 | 169 | 124 | 28 |
| XNOR2 | 52 | 52 | 116 | 84 | 15 |
| XNOR2 | 73 | 49 | 129 | 57 | 12 |
| IDENT | 38 | **0** | 250 | 98 | 34 |
| IDENT | 48 | 39 | 118 | 87 | 12 |

Table 4: Individual gate power for some library cells

# 4 Inaccuracy due to unknown inputs

Section 3 discussed the inaccuracies of a gate-level power estimator in comparison with a circuit simulator. From now on we will not be considering the circuit simulator and will only carry out comparisons between results reported by the gate-level simulator under various assumptions. This section compares power estimates produced with different input stimuli, while keeping all the other parameters constant.

One goal of power estimation is to calculate the amount of energy dissipated by a given chip in "actual usage." We will assume that the actual usage is represented by a given sequence of inputs and that the circuit starts from a given initial state. The sequence of inputs can be billions of cycles long and therefore it needs to be approximated. There are basically two approaches to the approximation – forming a sub-sequence [4], or characterizing the whole sequence in the form of probability distributions [2]. A closely related problem exists in processor performance estimation [15]. In this paper we make no effort to solve the problem of finding the correct input distribution. We merely show what we saw for our given input distributions.

Only in the case of benchmark B23, a designer gave us an initial state and an input trace of about 15,000 cycles. This made the benchmark of particular interest. For all the other benchmarks we do not have such information and therefore we assumed the initial state to be all '0's and the input sequence to be random with 50% probability of a '1' on any input.

Figure 1 studies the error incurred by using only a sub-sequence instead of the whole sequence. The figure contains four curves of energy consumption in the benchmark B13. These average energy curves were obtained by running 500 cycles of random inputs. Assume that these are the input patterns in actual usage; we wish to estimate energy consumed after running 500 cycles of the given sequence. Equivalently, we want the average energy per cycle consumed during those 500 cycles. The curve (a) in Figure 1 represents the running average, that is, the value of the curve at the N-th cycle is equal to the energy consumed during those N cycles divided by N. Our goal is then to find the final value for N = 500. The two horizontal lines represent +5% and -5% of that final value. Recall that 5% is the expected accuracy of the circuit simulator and hence there is no reason for us to be more accurate than that.

The curve (b) represents the average energy over the previous 100 cycles; that is, the value of the curve at the N-th cycle is equal to the energy consumed during the previous 100 cycles divided by 100. The curve (c) represents the average energy over the previous 10 cycles, and the fourth curve plots the energy consumed in each cycle. In all four charts we include the same +5% and -5% lines for reference.

These curves are typical of what we saw in all our benchmarks, and therefore we do not show all of them. The curve (a) shows not only fast convergence to $\pm 5\%$ of its final value, but it actually happens to be always within the $\pm 5\%$ range. This was typical for most, but not all of the benchmarks (see later discussion of B23). In all our benchmarks the average of any 100 cycles (curve (b)) was practically always within a 5% error margin, which is the accuracy limit of the circuit simulator. This represents a situation of estimating power using 100 input patterns other than those of eventual hardware usage, but assuming the same probability distribution. The third curve represents a situation of estimating power by simulating only 10 cycles, and we see that it results in about a 10% error.

When using a gate-level simulator there is no reason to attempt better than 10% accuracy, and we found it surprising that such a small number of patterns is sufficient to achieve that accuracy level. While this result was consistent for all our benchmarks, we cannot generalize it to all designs, and all workloads. In particular, we are assuming here a given description of input activity, and we do not address the problem of justifying that it is in any sense "typical". We will mention this problem later when discussing the issue of initial state.

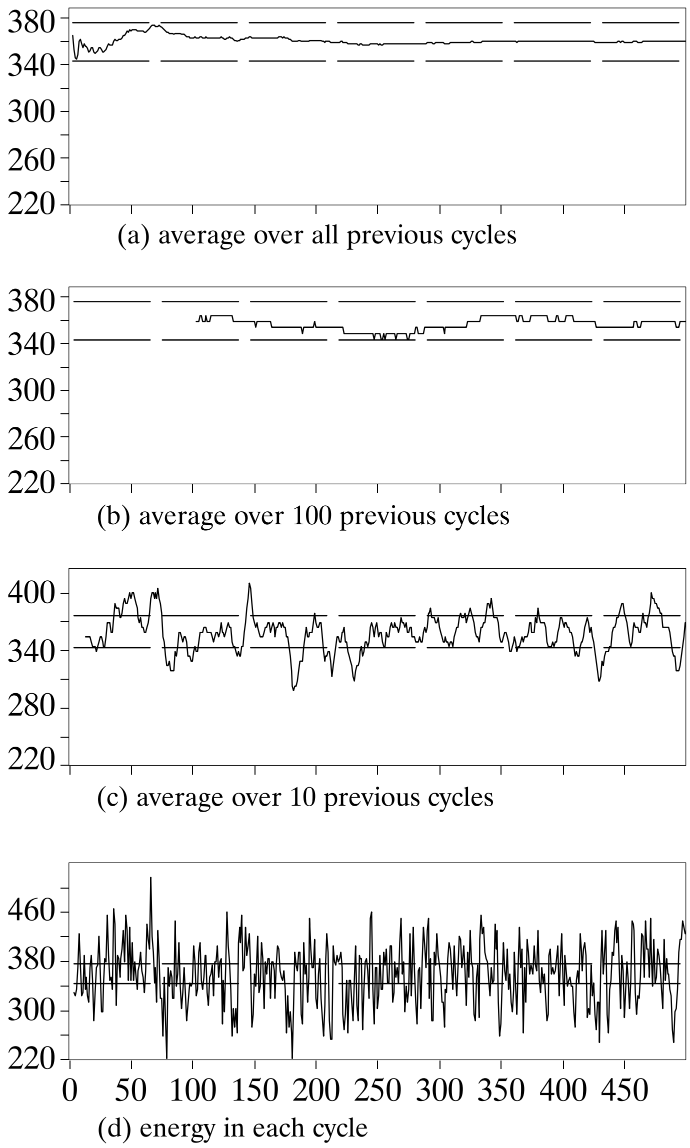



Figure 1: Average energy (in pJ) in B13 over 500 random inputs

As stated above, we were given a start state as well as a trace of 15,000 input patterns for the benchmark B23. The result of that simulation is in Figure 2. The figure shows three curves: (a) average energy over all previous cycles, (b) average energy over the 100 most recent cycles, and (c) average energy over 10 previous cycles. As above, we see that taking the average of any 100 consecutive cycles gives us $\pm 5\%$ accuracy, and averages taken over only 10 cycles are sufficient for $\pm 10\%$ accuracy.

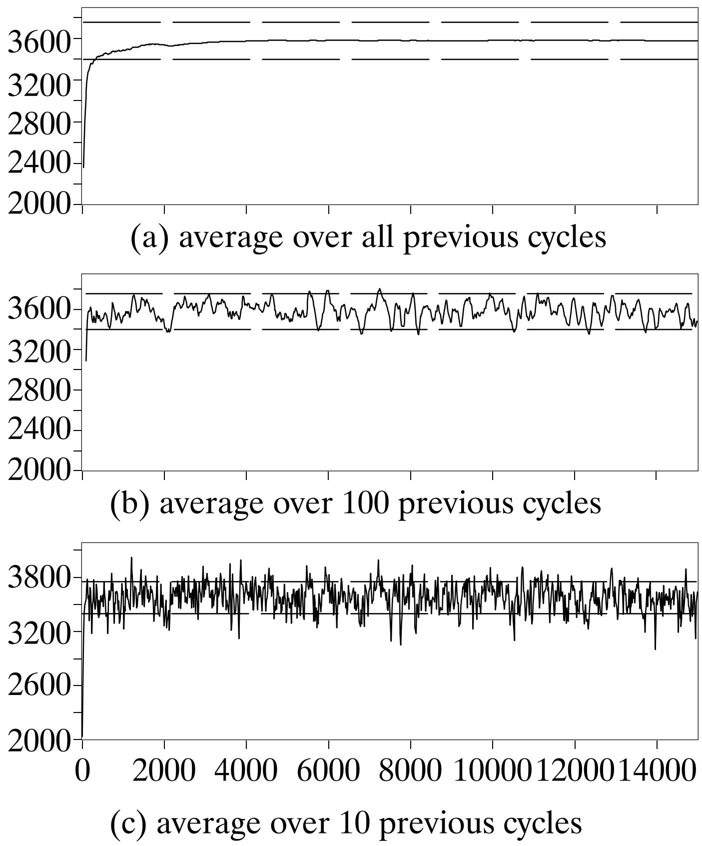



Figure 2: Average energy (in pJ) in B23 over 15,000 given inputs

A natural question arises as to the error incurred if we carried out the power estimation using random patterns rather than the patterns given by designers. Figure 3 shows a comparison of 500 cycles of simulation of B23, using the given input vectors, versus using random inputs. Again we show three charts as above, but each chart contains two curves – one for the designer-supplied inputs (solid) and the other for the random inputs (dotted). By considering the curves in (c) we see that energy consumption behaves differently for the two input sets, but the averages of energy are about the same. This similarity is in spite of the fact that the supplied inputs appear highly correlated in both time and space. However, in the supplied test patterns, '0's and '1's appear approximately with the same frequency, thus the probability density of the random patterns is very similar to that of the given patterns.

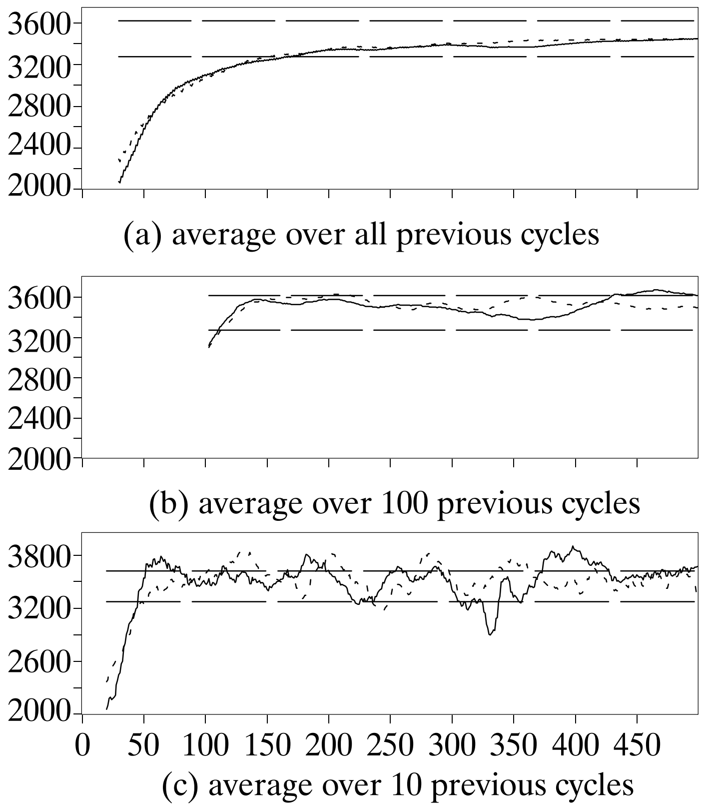



Figure 3: Energy (in pJ) consumed with supplied patterns vs random patterns

The close match between the random and the given input patterns raises the question of the degree of dependence of the power dissipation on input signal probability in benchmark B23. Therefore we performed several experiments where we varied signal probabilities of primary inputs, but not the starting state of latches; see Figure 4. The two lines of +5% and -5% are relative to the energy consumed with random inputs, where both '0' and '1' have a probability of 50% (solid). The dotted curves represent energy consumed with random inputs, where '0' has a probability of 75%. We see that inputs where '0' is more likely result in less power, but the reduction varies from cycle to cycle. Other experiments (not shown here) show that the reduction is even more pronounced when all inputs are constant '0' as well as when all inputs are constant '1'.

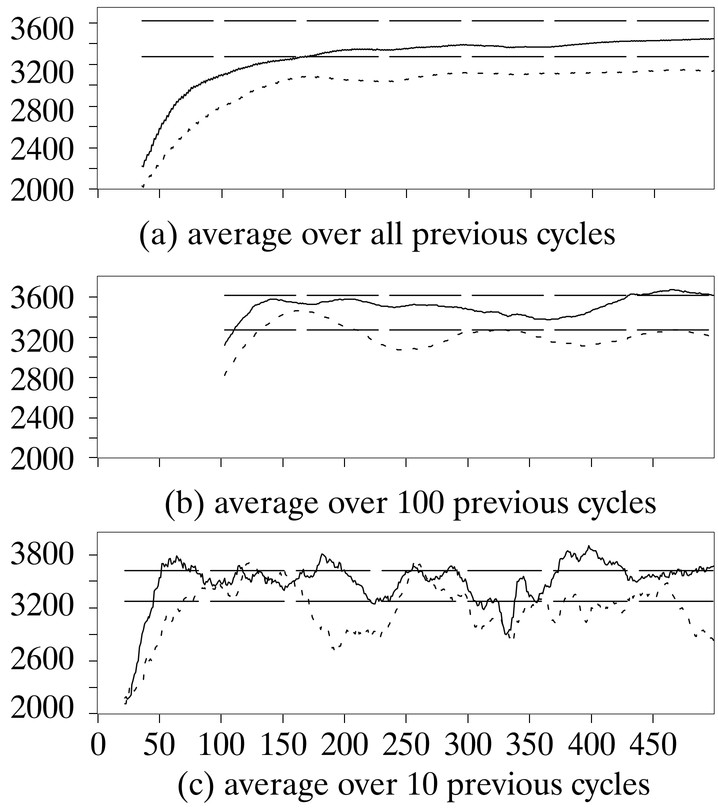
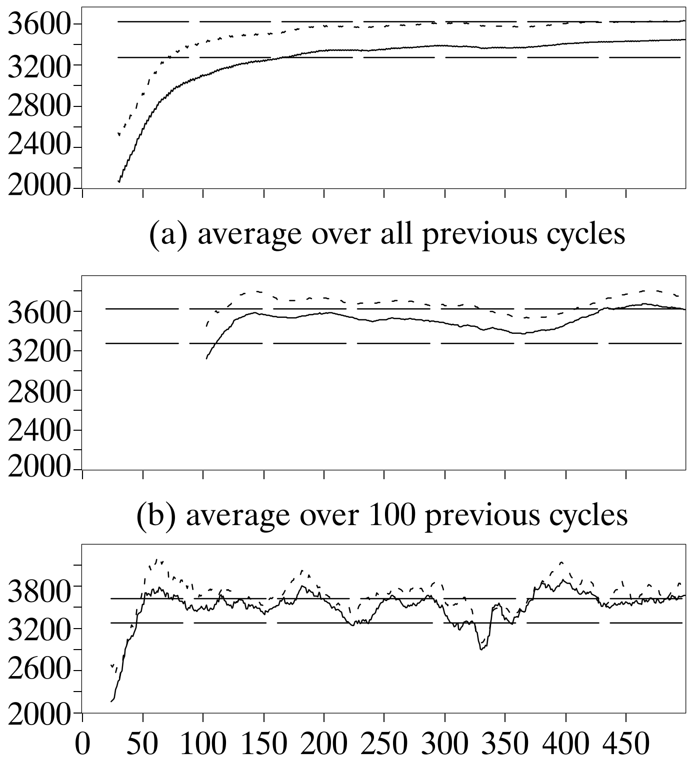



Figure 4: Energy (in pJ) consumed as a function of input probability

As mentioned above, in all our experiments the number of input patterns had surprisingly little impact on average energy. However, the benchmark B23 did show dependence on the initial state. Please note that in Figure 2 it takes about 2000 cycles for average energy shown in (a) to converge to its eventual level, and it takes about 500 cycles to reach the -5% error line. The curves (b) and (c) reach the $\pm 5\%$ region much faster because they quickly "forget" the initial cycles where the power estimate is low. The convergence of the benchmark B23 is much slower than what we saw in all the other benchmarks. We attribute this behavior to the process of the finite state machine settling to a steady state. The benchmark B23 is a digital filter, whose coefficients are designed to gradually adjust to the input patterns. The slow rise of average energy is due to this process of adjustment. This observation was confirmed by starting a new simulation with the state obtained at the end of a previous simulation run; this experiment resulted in average energy converging immediately to its final value.

We have run further experiments to study the impact of initial state on power estimates. Figure 5 shows a comparison of energy consumption for B23 with the supplied test patterns, but with two different start states. One is the start state supplied by the designer (solid) and the other is a starting state of all '0's (dotted). The two lines of +5% and -5% are relative to the energy consumed with the designer-supplied initial state. We see that the cycle to cycle variation in energy appears roughly the same, but for the case of an initial state all '0's, power is consistently greater by about 5%.

This dependence on initial state is an indication that some understanding of the design is needed before we can estimate power. For example, if the digital filter were adjusting its coefficients much slower, the initial rise of average energy might take millions of cycles, which would be difficult to detect.




(a) average over all previous cycles

(b) average over 100 previous cycles

(c) average over 10 previous cycles

Figure 5: Energy (in pJ) consumed as a function of initial state

All the investigations in this paper concern *average* energy, as opposed to *typical* energy. To illustrate the difference, we plot a histogram of energy consumed per cycle. Figure 6 shows the distribution for the benchmark B13, and Figure 7 shows the distribution for the benchmark B23. The vertical axis is energy consumed per cycle and the horizontal axis is the number of cycles in which that range of energy was consumed. The distribution of Figure 6 appears close to normal and is representative of the distribution we saw in all the other benchmarks, where average energy was also typical. In contrast, the distribution for benchmark B23 (Figure 7) is bimodal, because that logic has two clocks, one running at half the speed of the other. Thus on every even cycle all latches get updated, while only half the latches get updated on odd cycles. We see that in B23 typical and average energies are very different.

The shape of the distribution for B23 again points to the importance of understanding a design before estimating its power. If our goal were to calculate not only average energy, but the actual shape of its distribution, we would have to avoid simply summing power distributions of individual gates, as such a simplistic approach would result in an approximately normal distribution. Instead it would be essential to take into account the two-clock behavior of the design.

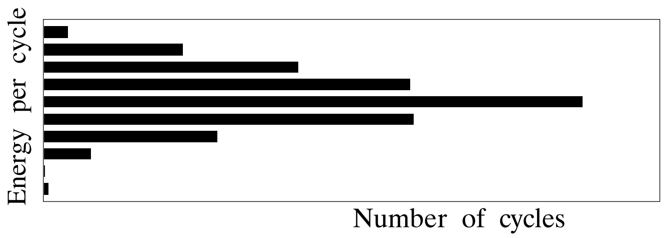



Figure 6: Histogram of energy consumption per cycle in B13

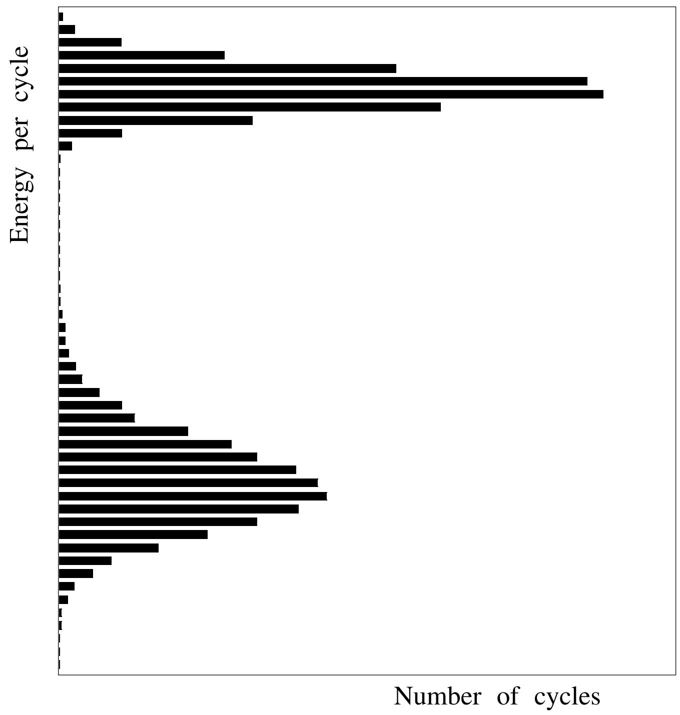



Figure 7: Histogram of energy consumption per cycle in B23

# 5 Inaccuracies due to synthesis and physical design

As described in Section 2, each of the benchmarks was processed through a fixed set of implementation stages. The stages were technology independent optimization, technology mapping, power level and fanout adjustment, and placement. (Recall that the impact of placement and wiring was approximated by randomly re-distributing wiring capacitances while retaining the same average capacitance.)

The results are summarized in Table 5. Each column represents a power estimate before or after one of the four stages. The estimate was obtained by running the gate-level simulator for 500 cycles. Each estimate, even those before technology mapping, was carried out as if that were the final implementation. (Approximations for delay were available to us even before technology mapping.) All the data are normalized so that the result "After placement" is 100. The largest and smallest values of each column are shown in bold face.

Each column "Before X" in Table 5 reports the extent to which power estimations are invalidated by stage X and subsequent stages. The last row shows the averages, which give us factors by which estimates obtained at each stage should be adjusted in order to make them on the average equal the power estimate after PD. But more important is the range of errors from the smallest to the largest in each column of Table 5. The range of errors gives us insight into the *confidence levels* with which power can be predicted at each of these stages. We see that the inaccuracy due to physical design (column 5) ranges between -18% and +13%, and the inaccuracy due to repowering, fanout correction, and physical design ranges between -24% and +9%. The range of inaccuracy for those two columns is about the same, but the average estimate "Before repowering" is smaller than "Before placement"; that is to be expected because repowering increases transistor sizes and introduces extra buffers. It should be noted that the inaccuracy of power estimation before technology mapping is much larger than after technology mapping.

All the power estimates in Table 5 include estimates of glitch power. Table 6 shows the percentage of glitch power for the various benchmarks and synthesis stages. As above, bold face indicates maximum or minimum in each column. We can see that in Table 6 the percentage of glitch power in the last column varies rather uniformly between 7% and 43%, which makes it difficult to estimate glitch power without mea-

| Name | Before optimization | Before tech. map | Before repowering | Before placement | After placement |
|---|---|---|---|---|---|
| B1 | 112 | **120** | 102 | 97 | 100 |
| B2 | 471 | 172 | 103 | **113** | 100 |
| B3 | 288 | 198 | 106 | 100 | 100 |
| C432 | 497 | 156 | 95 | 105 | 100 |
| B5 | **70** | **434** | 103 | 106 | 100 |
| C880 | 379 | 193 | 94 | 102 | 100 |
| B7 | 181 | 170 | 105 | **113** | 100 |
| C499 | 425 | 348 | 91 | 105 | 100 |
| C1908 | **1173** | 358 | 94 | 105 | 100 |
| C1355 | 640 | 230 | 89 | 97 | 100 |
| B11 | 150 | 175 | 86 | 91 | 100 |
| C2670 | 577 | 251 | 95 | 102 | 100 |
| B13 | 126 | 139 | 85 | 89 | 100 |
| B14 | 187 | 195 | 95 | 96 | 100 |
| B15 | 125 | 139 | **76** | **82** | 100 |
| C3540 | 505 | 210 | 96 | 103 | 100 |
| B17 | 157 | 137 | 80 | 88 | 100 |
| B18 | 120 | 122 | 95 | 100 | 100 |
| B19 | 147 | 167 | 88 | 93 | 100 |
| B20 | 165 | 153 | 88 | 96 | 100 |
| B21 | 163 | 204 | 98 | 99 | 100 |
| B22 | 78 | 169 | 82 | 86 | 100 |
| B23 | 137 | 257 | **109** | 93 | 100 |
| Mean | 299 | 204 | 94 | 98 | 100 |

Table 5: Impact of synthesis and placement on power estimates, including glitching

suring it. On the other hand, glitch power seems to be more a property of a design than of the implementation stage (at least in our design process). In other words, the estimate of glitch power before repowering is close to what it is found to be after placement. And even a crude estimate of glitching before optimization gives us a reasonable estimate of how much glitch power we can expect after placement. While this is true in most benchmarks, it is not true in general and later we will comment on the sharp decrease in glitch power for B1, B3, and B23 as a result of repowering.

It is also interesting to note that the public benchmarks tend to glitch more than our internal ones. We attribute that to the fact that many of our internal benchmarks are taken from high performance microprocessors, where we can expect less glitching than in designs where timing is not so critical.

| Name | Before optimization | Before tech. map | Before repowering | Before placement | After placement |
|---|---|---|---|---|---|
| B1 | 36% | 34% | 27% | 17% | 18% |
| B2 | 27% | 19% | 14% | 12% | 12% |
| B3 | 25% | 35% | 39% | 29% | 28% |
| C432 | 43% | 41% | 28% | 28% | 27% |
| B5 | 35% | 28% | 19% | 19% | 21% |
| C880 | 38% | 34% | 35% | 36% | 34% |
| B7 | 14% | 10% | 12% | 11% | 11% |
| C499 | 53% | 50% | 33% | 37% | 39% |
| C1908 | 57% | **56%** | 38% | 41% | 42% |
| C1355 | 53% | 45% | 33% | 36% | 36% |
| B11 | 21% | 22% | 17% | 16% | 16% |
| C2670 | 31% | 30% | 30% | 29% | 29% |
| B13 | 10% | 11% | **8%** | **7%** | **7%** |
| B14 | 8% | 6% | 14% | 10% | 12% |
| B15 | 13% | 12% | 13% | 11% | 11% |
| C3540 | **58%** | 54% | **47%** | **44%** | **43%** |
| B17 | 16% | 12% | 13% | 13% | 13% |
| B18 | **7%** | **8%** | 15% | 12% | 12% |
| B19 | 11% | 11% | 9% | 8% | 8% |
| B20 | 19% | 17% | 12% | 12% | 12% |
| B21 | 18% | 24% | 22% | 18% | 17% |
| B22 | 15% | 16% | 17% | 13% | 13% |
| B23 | 29% | 36% | 38% | 23% | 25% |

Table 6: Percentage of glitch power

If glitch power were to be ignored (an assumption commonly made in some tools), we would get the results shown in Table 7. In that table 100 is the estimate of power after placement *including glitch power.* As in Table 5, the two columns before technology mapping show large errors, as much as an order of magnitude. The three columns after technology mapping show smaller ranges of error, and a consistent under-estimate of up to 45%.

We would like to point out the impact of repowering on power consumption. Before repowering, all gates are at their lowest power level (size). The goal of repowering is not to improve timing, as no timing requirements were set, but merely to avoid overloading. Thus during repowering, the power levels of overloaded gates are increased to match the capacitance they have to drive, and fanout trees are built to cope with large fanouts. We would expect this process to increase power consumption (and the opposite process of reducing transistor sizes or "powering down" to save power). In Table 5 we can see that this expectation is usually met, except for benchmarks B1, B3 and B23. These are also the benchmarks that experience the sharpest drop in glitching as a result of repowering (Table 6). While increases in power level do not always lead to reductions in glitching, sometimes they result in such large reductions in glitching that the overall power consumption goes down. A limited analysis of this phenomenon suggest that it is caused by equalization of path length during repowering and fanout correction; two signals arriving at a gate at different times can cause a *healthy* glitch consisting of two full transitions, but if their arrival times are made approximately equal, then any glitch is likely to be only due to *simultaneous switching*, which consumes less power. This observation suggests that it is important to consider glitch power during the reverse process of powering down so as to avoid the possibility of increasing power consumption.

| Name | Before optimization | Before tech. map | Before repowering | Before placement | After placement |
|---|---:|---:|---:|---:|---:|
| B1 | 77 | **84** | 79 | 87 | 82 |
| B2 | 376 | 152 | 96 | **108** | 88 |
| B3 | 226 | 139 | 69 | 76 | 72 |
| C432 | 304 | 97 | 73 | 80 | 73 |
| B5 | **48** | **321** | 87 | 90 | 79 |
| C880 | 249 | 135 | 65 | 69 | 66 |
| B7 | 163 | 161 | **97** | 105 | 88 |
| C499 | 213 | 185 | 65 | 70 | 61 |
| C1908 | **546** | 168 | 63 | 67 | 58 |
| C1355 | 331 | 136 | 65 | 68 | 64 |
| B11 | 129 | 148 | 77 | 84 | 84 |
| C2670 | 427 | 187 | 71 | 76 | 71 |
| B13 | 121 | 132 | 84 | 89 | **93** |
| B14 | 185 | 199 | 86 | 94 | 88 |
| B15 | 120 | 132 | 73 | 79 | 89 |
| C3540 | 229 | 102 | **55** | **62** | **57** |
| B17 | 145 | 132 | 76 | 84 | 87 |
| B18 | 120 | 122 | 87 | 95 | 88 |
| B19 | 141 | 159 | 86 | 93 | 92 |
| B20 | 145 | 136 | 84 | 90 | 88 |
| B21 | 143 | 165 | 82 | 88 | 83 |
| B22 | 72 | 154 | 73 | 81 | 87 |
| B23 | 97 | 165 | 68 | 71 | 75 |
| Mean | 200 | 153 | 77 | 83 | 79 |

Table 7: Impact of synthesis and placement on power estimates excluding glitching

# 6 Power comparisons

During logic synthesis we need to know not only the power of a whole partition, but also the relative power of individual gates. The latter is needed to identify gates that are the main consumers of energy, and to evaluate the effect of power-saving transformations. However, we saw in Section 3 that the error in power estimates for individual gates, just due to using gate-level rather than circuit-level power estimation, can be orders of magnitude, and the standard deviation is around 50%. Thus the estimation error is larger than the typical power savings resulting from a single transformation! *Therefore, it would appear that we cannot base logic transformations on gate-level power estimates of individual gates.*

Fortunately, the effect of power-saving transformations can be judged without knowing the power consumed by individual gates; it is sufficient to be able to predict *relative power consumption* with reasonable confidence. This information is sufficient as long as such comparisons track throughout the implementation process; for example, if we estimate before technology mapping that one gate consumes 30% more power than another, a valid question is, "Will that relationship hold after the remaining stages of implementation are completed?" This section reports on how such comparisons track through synthesis and physical design.

The amount of tracking or correlation is different from design to design. For benchmark B3 it is shown in Figure 8. It is obtained by considering all pairs of gates. For each gate pair we predict the power difference during synthesis and plot it against the power difference after placement. A $45^o$ line would be the curve of perfect predictability. A curve randomly oscillating around the constant 0 would represent predictability of a random number generator.

Figure 8 contains four curves corresponding to four design stages. We see that curve 4 is close to a $45^o$ line; for example, if we wanted to make sure that one gate has 27% less power than another, before placement it is sufficient to insist on a predicted power difference of at least 30%. But if we were making the prediction before repowering (curve 3) then we would require a 50% predicted difference. Before technology mapping the difference would have to be predicted to be at least 80%. Before optimization no amount of predicted difference is sufficient to ensure 27% difference after placement.

The curves in Figure 8 were obtained by dividing all gate pairs into 20 groups. A pair of gates fell into the first group if their predicted power difference was between 0% and 5% of the "hotter" gate in the pair. The pair fell into the second group if the difference was between 5% and 10%. Pairs in the last category had a power difference between 95% and 100%, that is, one gate consumed at least 20 times the power of the other. Then after placement, for each group we considered those pairs of gates that survived the synthesis steps, and plotted the difference after placement against the predicted difference. If gates predicted "hotter" turned out to be "cooler" after placement, then the difference was counted to be negative.

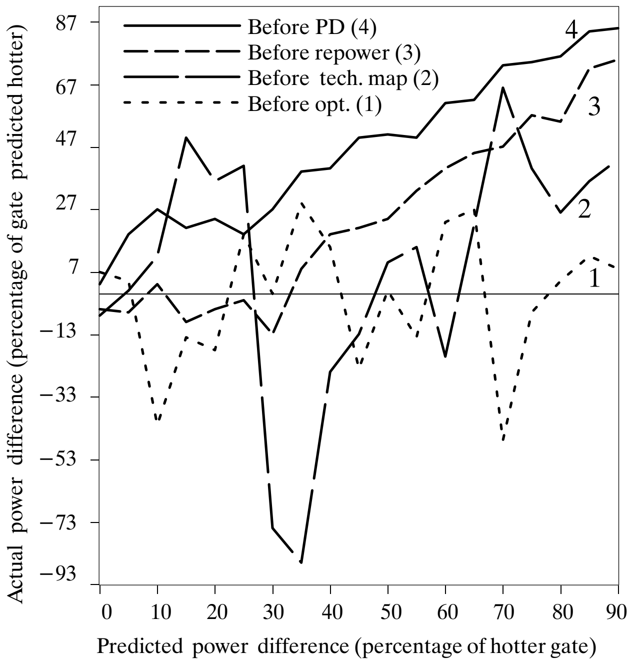



Figure 8: Predictability of power difference in B3

The data in Figure 8 seem to indicate that we have to be conservative when comparing the power of two gates. While this is true in general, the benchmark B3 had the worst predictability characteristics of all our benchmarks. The data for the benchmark B17 in Figure 9 shows better predictability, which was more typical. The main difference between the two benchmarks is their size. While each of the 20 groups for benchmark B3 had tens of gate pairs, there were hundreds or thousands of gate pairs per group in benchmark B17.

Even if power is the only optimization criterion, area cannot be increased arbitrarily in order to save power in the partition being synthesized. The reason is that an area increase in just one partition might lead to increased wire lengths throughout the chip and hence increased power. Before a transformation accepts a logic change it needs to check that the resulting power savings will be sufficiently high. Such a check is in general rather involved because a local change may have global implications, but simple comparisons between two gates (Figure 8 and Figure 9) form the basis for a more general check. We see that in order to have confidence in a power saving "move", it helps for a transformation

- to operate late in the synthesis process (after technology mapping),
- to predict relatively high power savings and
- to be applied many times.

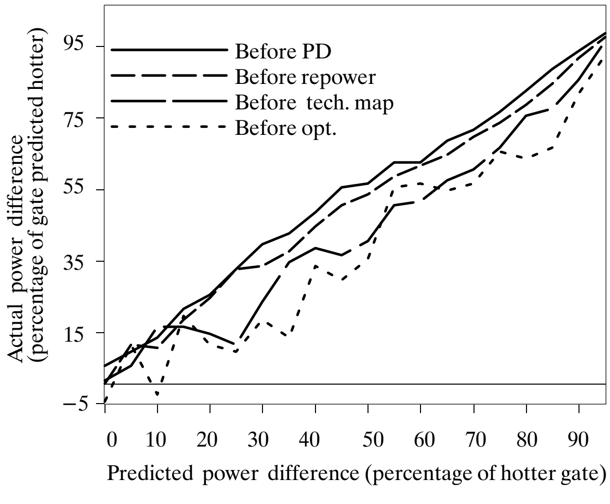



Figure 9: Predictability of power difference in B17

When a synthesis transformation does not increase area, it is not necessary to predict the power saved; sometimes we merely want to make sure that power will not get worse. To see how much confidence we can have in this type of prediction, consider Figure 10. The horizontal axis again represents the predicted power difference between gate pairs during various stages of synthesis. The vertical axis gives the percentage of pairs for which we predicted correctly which gate consumes more power. Ideally we would like all curves to be constant 100%. The line of 50% represents predictability of a random coin toss. The four curves in Figure 10 represent predictions performed at the four stages of synthesis as above. (The higher the curve the later the synthesis stage). For example, suppose we wanted an 80% confidence in our prediction, that means, we want to be wrong for only one pair in five. If we are making the prediction after technology mapping (the two higher curves), then we have to predict a power difference of at least 35%. If we are making a prediction before technology mapping, then we need a predicted power difference of more than 60%, and before optimization it must be at least 70%.

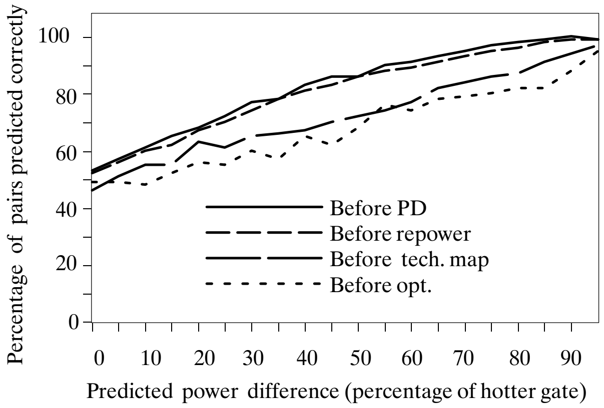



Figure 10: Predictability of which gate is "hotter" in B17

# 7 Conclusions

We have considered power estimations in two contexts – estimating the power of a whole partition, and estimating the relative power of two gates for the purpose of performing synthesis transformations. Our results are by necessity specific to our technology, simulators, and design processes. However, we have attempted to generalize our experiments wherever possible. The extent to which these results are applicable in general will be determined only after similar experiments are performed in different environments. The conclusions of such experiments are valuable in guiding a synthesis or power optimization system.

In this paper, we considered several sources of inaccuracies in power estimation. They are listed below in decreasing order of their impact of power estimation of whole partitions.

1. Optimization and technology mapping may cause power estimates to be off by an order of magnitude.

2. Internal gate capacitances, which are a function of the target library, accounted for about half the power.

3. Glitch power varied between 7% and 43%.

4. Repowering and physical design introduced inaccuracies below 20%.

5. Using a gate-level simulator as opposed to a circuit simulator caused an error of about $\pm 15\%$.

6. The number of input stimuli did not cause any error above the 10% mark if we considered at least 10 input patterns.

Obviously, these sources of inaccuracies have a much larger negative impact on power estimation of *individual gates*. We saw that just by using a gate-level simulator rather than a circuit simulator the standard deviation is about 50%. This uncertainty is too large for the needs of logic synthesis, which must predict the impact of transformations with greater accuracy. However, the study of correlations between anticipated and realized improvements between pair-wise combinations of gates is useful information. These correlations depend on the particular design and on the stage of synthesis at which the prediction is made. This type of prediction is not preserved for individual gate pairs, but is preserved on the average for a sufficiently large number of pairs.

Based on our experiments we draw the following conclusions about performing power estimation in our environment.

1. The use of gate-level estimation as opposed to circuit simulation is acceptable if we can tolerate an error of about 15% in the power estimate of a whole partition.

2. Simulation with as few as 10 patterns is often sufficient to reach confidence levels matching those of gate-level estimation.

3. Such a small number of input patterns is sufficient only if the patterns are drawn from a "typical" input distribution and only if the starting state is "typical". Determining what is "typical" cannot be done without understanding the behavior internal and external to the given design.

4. It is necessary to take into account the effect of optimization and technology mapping. Before technology mapping, the accuracy levels are unacceptable.

5. It is necessary to take into account internal gate capacitances, as counting net capacitances alone would result in a gross underestimate. However, average internal capacitances seem sufficient.

6. Glitch power can be ignored if we add 33% to a power estimate, and if we can tolerate the resulting error of about 20%.

7. Glitch power cannot be ignored for transistor resizing, which can change glitch power.

8. The impact of placement and wiring can be ignored as long as we know average capacitances. The error due to net-to-net capacitance variations induced by physical design is comparable to the error of using a gate-level estimator.

9. Gate-level power estimation cannot be used for individual gates.

10. Power improving transformations should be run in late stages of synthesis (taking into account optimization and technology mapping), they should be applied only if they can predict significant power improvement, and should be applied many times (hundreds) to maximize the confidence of positively impacting the design.

# 8 Acknowledgements

We are grateful to H. Chao, V. Iyengar, P. Kudva, D. Kung, R. Puri, L. Stok, and L. Trevillyan for valuable discussions and reading of the manuscript. H. Chao also provided us with the technology models.

# References

[1] S. M. Kang, "Accurate simulation of power dissipation in VLSI circuits," *IEEE Journal of Solid-State Circuits*, vol. SC-21, pp. 889–891, October 1986.

[2] M. A. Cirit, "Estimating dynamic power consumption of CMOS circuits," in *Proceedings of the IEEE International Conference on Computer-Aided Design*, (Santa Clara, CA), pp. 534–537, IEEE, November 1987.

[3] A. Ghosh, S. Devadas, K. Keutzer, and J. White, "Estimation of average switching activity in combinational and sequential circuits," in *Proceedings of the 29th ACM/IEEE Design Automation Conference*, pp. 253–259, ACM/IEEE, June 1992.

[4] R. Burch, F. N. Najm, P. Yang, and T. N. Trick, "A Monte Carlo approach for power estimation," *IEEE Transactions on VLSI Systems*, vol. 1, March 1993.

[5] F. N. Najm, "A survey of power estimation techniques in VLSI circuits," *IEEE Transactions on VLSI Systems*, vol. 2, December 1994.

[6] M. Favalli and L. Benini, "Analysis of glitch power dissipation in CMOS ICs," in *Proceedings of the 1994 International Workshop on Low Power Design*, (Napa, CA), pp. 27–32, IEEE, ACM, April 1994.

[7] C. X. Huang, B. Zhang, A. C. Deng, and B. Swirski, "The design and implementation of PowerMill," in *Proceedings of the International Symposium on Low Power Design*, (Dana Point, CA), ACM-SIGDA and IEEE-CAS, April 1995.

[8] C. Visweswariah and R. A. Rohrer, "Piecewise approximate circuit simulation," in *Proceedings of the IEEE International Conference on Computer-Aided Design*, (Santa Clara, CA), IEEE, November 1989.

[9] C. Visweswariah and R. A. Rohrer, "Piecewise approximate circuit simulation," *IEEE Transactions on CAD of ICs and Systems*, pp. 861–870, July 1991.

[10] F. Berglez, P. Pownall, and R. Humm, "Accelerated ATPG and fault grading via testability analysis," in *International Symposium on Circuits and Systems*, pp. 695–698, IEEE, June 1985.

[11] D. Brand, R. Damiano, L. van Ginneken, and A. Drumm, "In the driver's seat of BooleDozer," in *Proceedings of the IEEE International Conference on Computer Design*, (Cambridge, MA), pp. 518–521, IEEE, October 1994.

[12] S. Sastry and A. C. Parker, "Stochastic models for wireability analysis of gate arrays," *IEEE Transactions on Computer-Aided Design*, vol. CAD-5, pp. 52–65, January 1986.

[13] T. L. Chou, K. Roy, and S. Prasad, "Estimation of circuit activity considering signal correlations and simultaneous switching," in *Proceedings of the IEEE International Conference on Computer-Aided Design*, (San Jose, CA), pp. 300–303, IEEE, November 1994.

[14] H. J. M. Veendrick, "Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits," *IEEE Journal of Solid-State Circuits*, vol. SC-19, pp. 468–473, August 1984.

[15] V. S. Iyengar, L. H. Trevillyan, and P. Bose, "Representative traces for processor models with infinite cache," in *Proceedings of the 2nd International Symposium on High-Performance Computer Architecture*, (San Jose, CA), pp. 62–72, IEEE, February 1996.