Power estimation in logic circuits using activity simulation

Hijdra, M.B.

Award date:
1995

Eindhoven University of Technology
Department of Electrical Engineering
Design Automation Section (ES)

Power Estimation in Logic Circuits using Activity Simulation

By M.B. Hijdra

Master Thesis

performed from February 1995 to October 1995
by order of prof.Dr.-Ing. J.A.G. Jess
supervised by dr. ir. M.R.C.M. Berkelaar

Abstract

Several methods exist to estimate the power consumption of combinational circuits. The conventional methods are based on logic simulation. New methods use a probabilistic approach. The model used for probabilistic power estimation uses uses a zero or general delay model. A zero delay model is based on single-input switching and a general delay model on multiple-input switching. Using a general delay model it is possible to detect glitches which contribute to the total power consumption. This glitching effect can be decreased by inertial delay filtering. These different methods are implemented in two programs. Testing these programs for 30 different circuits ranging in size from one to 178 gates gives results which are compared on accuracy and run-time. Finally the statistical behavior is found by estimating the total circuit power 20 times. In every run different circuit capacities are used. This represents variations in interconnect wire-length and technology tolerances.

Contents

Chapter 1

Introduction

With the emergence of battery-operated applications that demand intensive computation in a portable environment, power analysis and optimization has become one of the most important problems to be solved by computer-aided design tools. It has become a very serious requirement that integrated circuits have limited power dissipation, therefore an accurate simulation of power dissipation in an early stage of design has become highly desirable.

Power dissipation in integrated circuits is closely related to the choice of technology, circuit design, logic design, and the choice of architecture. One of the major reasons for the use of CMOS technology in the current VLSI designs is that power dissipation in CMOS circuits is significantly lower than in other technology circuits at comparable speeds. However, with the tremendous increase of the number of devices in VLSI and the ever-increasing number of portable applications, such as hand-held telephones and lap-top computers, requiring low power and high throughput, it has become increasingly important to reduce power dissipation. Hence, low power circuit design has become essential.

The first step in low power circuit design is the estimation of dissipated power. Using the result of this estimation it is possible to change the circuit design and to compare the dissipated power of the changed design against the original. This strategy can be repeated several times to reach an acceptable circuit design solution with respect to power dissipation.

To reduce the power dissipation a new element in the cost function for circuit design must be introduced. This element is a measure of the estimated power dissipation in this new circuit. Optimizing this circuit for low power can be achieved by observing the power part in the cost function. It is extremely important to have an accurate and fast method for estimating the circuit power. These aspects, accurate and fast power estimation, will be handled in this thesis report.

Chapter 2

Problem Description

This chapter describes the problem of power estimation. First the power dissipation in CMOS logic will be analyzed. The next step describes how to use the single cell power consumption for estimation of the overall dissipated power in a circuit.

2.1 CMOS Power Dissipation

The power dissipated in circuits, designed in CMOS technology, is formed by three main sources:

- switching currents
- short-circuit currents
- leakage currents

The leakage currents are very small in CMOS circuits and can be neglected for power estimation. The short-circuit currents depend on the capacitive load of the the gate and can become as high as 10%. However, this is generally much lower and is therefore not included in power estimation.

The remaining problem is formed by the switching currents. This switching component, resulting from the charging and discharging of the load capacitors, dominates in the total power consumption. It is referred to as dynamic power dissipation. This dynamic power consumption for a CMOS gate, with a loading capacitor C_{load}, is given by the equation:

$$P_{dyn} = p_t * C_{load} * V_{dd}^2 * f \tag{2.1}$$

where C_{load} is the loading capacitance,
V_{dd} is the supply voltage,
f is the clock frequency,
p_t is the probability of a power consuming transition.

In this report a signal change from *zero* to *one* or from *one* to *zero* will be called a *transition*. A *power consuming transition* is defined as a *zero* to *one* transition, because only by this transition power from the supply is consumed. A *one* to *zero* transition is nothing else but a discharge of a nodal capacitance, so no power from the supply is dissipated. Because *zero* to *one* and *one* to *zero* transitions always alternate, the number of transitions in a circuit will be approximately twice as large as the number of power consuming transitions, therefore p_t has a value of 0.5.

With aid of equation 2.1 the power dissipation of one circuit node can be calculated. For total circuit power this must be calculated for all nodes of the circuit. The calculation uses the assumption that every node changes from *zero* to *one* and from *one* to *zero*. This is of course not true, many nodes don't switch that often or some of them switch even more often than two times in one single clock cycle. To take this effect in account a transition density is introduced which represents the number of transitions in one clock cycle.

For gate number n this results into equation:

$$P_{dyn,n} = 0.5 * C_{load,n} * V_{dd}^2 * f * D_n \tag{2.2}$$

where $C_{load,n}$ is the loading capacitance at node n,
V_{dd} is the supply voltage,
f is the clock frequency,
D_n is the transition density at node n.

The total consumed circuit power is now given by:

$$P_{circuit} = \sum_{gates} P_{dyn,n} \tag{2.3}$$

2.2 Analysis of CMOS Power Consumption

Formula 2.2 describes the consumed power in a gate as a function of four variables. These variables must be known to evaluate the power in a circuit. For abstraction it is allowed to leave out the clock frequency because this is a scalar which is equal for all gates in the circuit.

The supply voltage property is also a constant and can also be left out for calculation. Thus the evaluation of equation 2.2 consists now of determining $C_{load,n}$ and D_n. The estimated power in Watts is now expressed in Joules.

The $C_{load,n}$ represents the total capacity at a node which consists of gate capacity and wire capacity. The gate capacity is technology dependent and is caused by the parasitic capacities. In a typical CMOS two-input NAND gate (figure 2.1) the major parasitic capacities are: C_{gate_gnd} C_{source_gnd} and C_{drain_gnd}.

The wire capacity is formed by the interconnect wires between the gates. These capacities can become quite large if many gates are connected to one single gate output (fan-out > 1). In large circuits the interconnect wires can be very long and have a large capacity accordingly. Also the gate capacities of the input transistors in the connected gates are a considerable capacitive load addition.

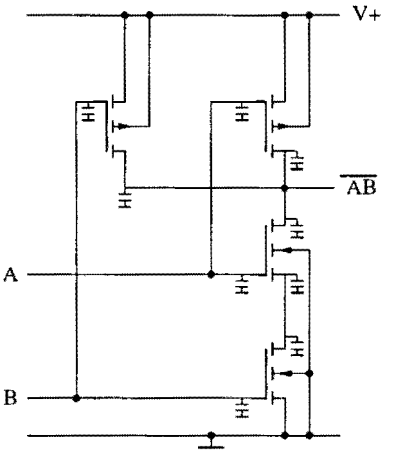

Figure 2.1: Two input CMOS NAND gate

The transition density D_n is the other unknown value in the estimation process. Calculation of this value is not very easy. The transition density at a node, also called switching activity, is dependent on the activity of its predecessors. This relation must be followed until the primary circuit inputs are found.

The calculation is not as straightforward as described above, because also a delay model must be taken into account. Also other effects like glitching and spatial correlations may have a significant influence.

This thesis report is mainly focused on the influence of these effects on the power calculation with respect to the accuracy. Special attention is paid on glitching and on filtering of these glitches by inertial delays.

Chapter 3

Probability and Density

In this chapter basic statistic theory is discussed concerning stochastic 0-1 processes and transition density.

3.1 Stochastic 0-1 processes

Let $x(t)$, $t \in (-\infty, +\infty)$, be a stochastic process that takes the values 0 or 1, changing state between them at random times. Such a process is called a 0-1 process. A stochastic process is said to be Strict-Sense Stationary (SSS) if its statistical properties are invariant to a shift of the time origin.

When $x(t)$ is SSS, then the probability that it takes the value 1 at any given time t, denoted by $P\{x(t) = 1\}$, which is equal to its *mean* or *expected* value $E[x(t)]$ at that time, is a constant, independent of time. This value is referred to as the *equilibrium probability* of $x(t)$, denoted by $P(x) \triangleq P\{x(t) = 1\}$. Since $x(t)$ is also mean-ergodic, then:

$$\lim_{T\to\infty} \frac{1}{T} \int_{\frac{-T}{2}}^{\frac{+T}{2}} x(t) = E[x(t)] = P(x) \tag{3.1}$$

Let $n_x(T)$ denote the random number of transitions of $x(t)$ in $(\frac{-T}{2}, \frac{+T}{2}]$. When $x(t)$ is SSS, then $E[n_x(T)]$ depends only on T, and is independent of the location of the time origin.

The ratio $E[n_x(T)]/T$ is the expected number of transitions per unit time. We will refer to it as the *transition density* of $x(t)$, to be denoted by $D(x) \triangleq E[n_x(T)]/T$.

Let $z(t) \triangleq \frac{d}{dt} n_x(t)$ be the output of a differentiator whose input is $n_x(t)$. The process $z(t)$ is an impulse train that can be written as:

$$z(t) = \sum_{i=-\infty}^{+\infty} \delta(t - t_i) \tag{3.2}$$

where the t_i's are the transition time points of $x(t)$. Since a differentiator is a linear system, we have:

$$E[z(t)] = \frac{d}{dt}E[n_x(t)] = D(x) \tag{3.3}$$

Therefore, $z(t)$ has a constant mean.

$$\lim_{T\to\infty} \frac{n_x(T)}{T} = \lim_{T\to\infty} \frac{1}{T} \int_{\frac{-T}{2}}^{\frac{+T}{2}} z(t)dt = E[z(t)] = D(x) \tag{3.4}$$

3.2 Random Logic Signals

Let $x(t)$ be a logic signal that switches with zero rise and fall times between its 0 and 1 value. It is clear that $x(t)$ can be thought of as a sample of a stochastic 0-1 process $x(t)$. If $x(t)$ is SSS and mean-ergodic, then in view of equation 3.1

$$\lim_{T\to\infty} \frac{1}{T} \int_{\frac{-T}{2}}^{\frac{+T}{2}} x(t) = P(x) \tag{3.5}$$

which means: "$P(x)$ is, asymptotically, the fraction of time that $x(t)$ spends in the 1 state."

Similarly, in view of equation 3.4, we have:

$$\lim_{T\to\infty} \frac{n_x(T)}{T} = D(x) \tag{3.6}$$

Thus "$D(x)$ is, asymptotically, the average number of transitions of $x(t)$ per unit time." $D(x)$ is called the transition density. The two results 3.5 and 3.6 provide the essential link between physical reality (the quantities on the left hand side) and the stochastic measures $P(x)$ and $D(x)$.

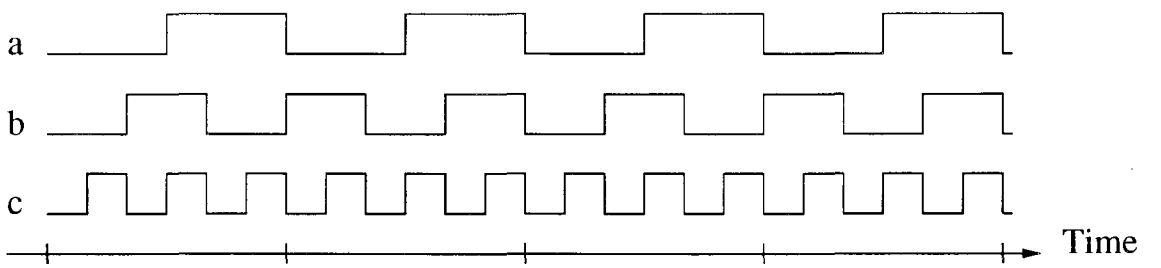

Figure 3.1: Transition Density

For example figure 3.1 shows three logic signals. Signal a,b and c all have the same probability of 0.5, but the signal densities are *not* the same. Signal a has a density of 2, meaning two transitions per unit time. Signal b makes 3 transitions per unit time and signal c 6 transitions. From figure 3.1 can be observed that probability and density are two independent properties, except in two special cases: if the probability of a signal $x(t)$ is 0 or 1, no signal transitions can be made resulting in zero transition density.

3.3 Logic Problem Definition

Consider a multi-input multi-output logic module M, whose outputs are boolean functions of its inputs, as shown in figure 3.2.

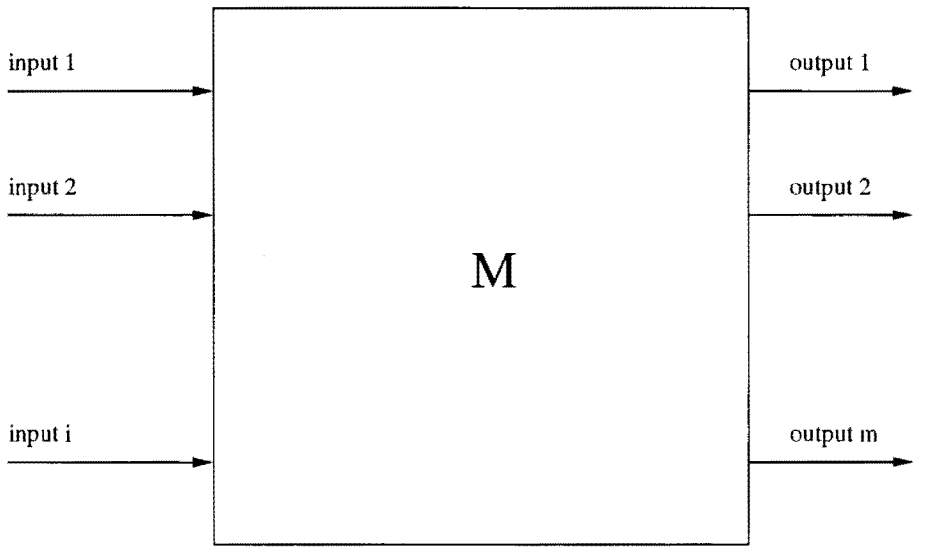

Figure 3.2: Logic module M

We now consider the problem of propagating equilibrium probabilities and densities from the module inputs to its outputs. Since the internal structure of M is not known, the problem is actually even more generic than that, and can be expressed as "given a boolean function $f(x_1, ..., x_n)$ and that each x_i can be high with a probability $P(x_i)$, what is the probability that f is high?" An easy way for representing these functions is based on Binary Decision Diagrams (BDDs).

3.4 Transition Density Calculation

The next step is the density propagation problem. Using a *boolean difference* it is possible to calculate the propagation function.

If y is a boolean function that depends on x, then the boolean difference of y with respect to x is defined as:

$$\frac{\delta y}{\delta x} \triangleq y\,|_{x=1} \oplus y\,|_{x=0} = y(x) \oplus y(\overline{x}) \tag{3.7}$$

where $\oplus$ denotes the exclusive-or operation. Note that, if x is an input and y is an output of M, then $\delta y/\delta x$ is a boolean function that does *not* depend on x, but may depend on all other inputs of M. Therefore, $\delta y/\delta x$ and x are independent. A crucial observation is that if $\delta y/\delta x$ is 1, every transition at x will cause a transition at y. Output y does never change when input x changes if $\delta y/\delta x$ is 0.

The transition densities $D(y_i)$ of a logic module at its outputs $y_j(t), j = 1, ..., m$ can now be written as a function of its input densities $D(x_i)$:

$$D(y_j) = \sum_{i-1}^{n} P\left(\frac{\delta y_j}{\delta x_i}\right) D(x_i) \tag{3.8}$$

If the function resulting from the boolean difference operation is available, its probability can be evaluated. This results in evaluating the probability of a boolean function knowing the probability of its inputs.

3.4.1 Transition Density Calculation Example

For example the output transition density of a two-input AND gate, shown in figure 3.3, can be calculated using equations 3.7 and 3.8:

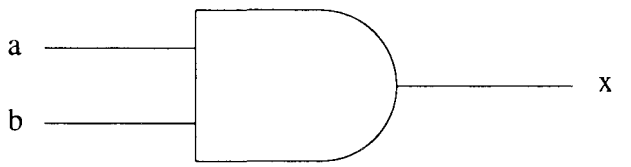

Figure 3.3: Two input AND gate

Given the logic function $x(a,b) = a \cdot b$ the boolean difference of the output can be calculated in relation to an input using equation 3.7:

$\frac{\delta x(a,b)}{\delta a} = x(a,b)\mid_{a=1} \oplus x(a,b)\mid_{a=0}$

$\Rightarrow x(1,b) \oplus f(0,b) = 1 \cdot b \oplus 0 \cdot b = b \oplus 0 = b$

$\frac{\delta x(a,b)}{\delta a} = b$

similar for input b: $\frac{\delta x(a,b)}{\delta b} = a$

substitute these results in equation 3.8: $D(x) = P\left(\frac{\delta x(a,b)}{\delta a}\right) D(a) + P\left(\frac{\delta x(a,b)}{\delta b}\right) D(b)$

result is: $D(x) = P(b) \cdot D(a) + P(a) \cdot D(b)$

This example shows that the transition density of an output node x can be easily calculated using equations 3.7 and 3.8 combined with the given input probability and density.

Chapter 4

Power Estimating Models

Power estimation can be done in different ways. Every model used for estimation has its own advantages and disadvantages. In this chapter different model aspects are described.

4.1 Circuits

Two sorts of circuits can be analyzed for power consumption:

- combinational circuits
- sequential circuits

Estimation of power consumption for sequential circuits is different from combinational circuits. In a sequential circuit signal transitions can only occur at the rising or falling edge of a clock transition, whereas, a block of combinational logic can have very different input signals which have no time correlation at all.

The dissipated power in a sequential circuit consists of two parts. The first part is the power consumed by the clock signal. This can be rather large in big circuits because the clock signal must be connected to all clocked devices spread out all over the chip. This results in long wires resulting in large capacities. The second part consists of the internal signal transitions in the clocked devices.

In combinational circuits, where the inputs are only stimulated at a clock edge, power is dissipated until the latest signal has reached a stable state. Every transition, useful or not, contributes to the total power consumption.

Most sequential circuits have a block of combinational logic between latches or other clocked devices. Because the inputs of the combinational blocks are formed by the outputs of a sequential logic block, all the input transitions occur at exactly the same time. A decoupling can be made to estimate the power in the combinational and sequential part of the circuit independent of each other.

An improvement of this method is possible by taking signal correlations into account. These are called spatial correlations between inputs. It is even possible to determine the behavior of the sequential blocks in time. This behavior is also known as temporal correlations.

4.2 Input Modeling

The choice of the input model has a big influence on the power consumption. Of importance are:

- single input switching
- multiple input switching

There are two input simulation models which can be used. The fist model, called single input switching, is based on the situation that only one input changes at a time. In the other model, called multiple input switching, several or all inputs can change state simultaneously.

The calculation method based on boolean difference implies a single input switching model. The reason is that by using the difference the influence of one input on the output density is calculated. This is repeated for all inputs of the circuit which results in the total transition density for this output.

By examination the contribution of each input separately it is assumed that each input is independent of the others, in other words only one input switches at a time. By using this independency of the inputs it is obvious that spatial input correlations are *not* taken into account.

The boolean difference calculation cannot be applied to the multiple input switching model, instead a symbolic simulation method must be used (see section 4.3.3).

4.3 Delay Modeling

For the power calculation one of the following delay models must be chosen:

- zero delay model
- unit delay model
- general delay model

Delay modeling can be split up in two separate approaches: zero delay modeling and unit or general delay modeling.

Zero Delay Model

A zero delay model implies no delays at all in the circuit. This results in a circuit representation of two levels, input and output level. Using this model the outputs of a circuit are represented as a boolean function of the primary inputs.

Power estimation of this model consists of taking the boolean difference of all inputs in the expression for this output. The next step is evaluating the probability of the boolean difference result and finally this is multiplied with the input density. This calculation procedure is repeated for all outputs. The results are summed, representing the total circuit transition activity.

The main advantage is, that this is a fast method for obtaining information about the density of primary circuit outputs.

The main disadvantage is that timing problems, which can result in glitches or hazards, are not taken into account by this delay model.

Unit and General Delay Model

A general delay model is a circuit model which takes the effect of signal delay into account resulting from charging and discharging the circuit capacities at every gate. Unit delay modeling is a special case of general delay. A unit delay model represents every circuit delay in a number of fixed unit delays. A general delay can have any arbitrary value, therefore, a unit delay model can be represented by a general delay model.

Calculation of the transition densities of a general delay model is different from the zero delay model. This is because an output can only change its state when its inputs change. These inputs are mostly *not* primary circuit inputs but outputs of predecessing nodes. Therefore, the transitions of all internal predecessing nodes must be known in order to calculate the correct output transition density.

4.3.1 Unit Delay Model

Even under the idealization of a unit-delay model, the gate output nodes of a multilevel network can have multiple transitions in response to a two-vector input sequence. In fact, it is possible for a gate output to have as many transitions as levels in the network.

The boolean functions are constructed describing the gate outputs at the discrete points in time implied by the unit-delay model. In this manner only discrete times, t, $t+1$, ..., $t+l$ are considered, where t is the time when the inputs change from $V0$ to Vt, and l is the number of levels in the network.

For each gate output i, symbolic simulation is used to construct the $l+1$ boolean functions $f_i(t+j), j \in 0, ..., l$ which evaluate to 1 if the gate's output is 1 at time $t+j$. Note that it is assumed that no gate has zero delay and that the network has settled before the inputs are changed from $V0$ to $V1, where f_i(t)$ is the logic function performed for $V0$ at the i^{th} gate output.

Finally, it can be determined whether a transition occurs at a boundary between discrete time intervals $t+j$ and $t+j+1$ by exclusive-OR'ing $f_i(t+j)$ with $f_i(t+j+1)$.

4.3.2 General Delay Model

To determine an output transition in a general delay model a similar strategy as in the unit delay model is followed. The only difference is that a transition can occur at any time, not just on a multiple of a unit time-step. The method for transition calculation using input-vectors is called symbolic simulation.

4.3.3 Symbolic Simulation

A gate output can only change when its inputs change, and then only if the boolean function describing the gate evaluates differently. For example, a 2-input AND gate's output will change between clock cycle t and $t+1$ if

$$(i_1(t) * i_2(t)) \oplus (i_1(t+1) * i_2(t+1)) \tag{4.1}$$

evaluates to 1, where $i_1(t)$, $i_2(t)$ and $i_1(t+1)$,$i_2(t+1)$ are the inputs to the gate at clock cycle t and $t+1$ respectively.

To compute the average power dissipated in a combinational network a similar approach is used. A two-vector input sequence is needed to to stimulate the activity in static gates. A logic output will switch with a change in the primary input vector from $V0$ to Vt if:

$$f_i = ((h_i(V0) = 0) \wedge (h_i(Vt) = 1)) \vee ((h_i(V0) = 1) \wedge (h_i(Vt) = 0)) \tag{4.2}$$

$$\text{or denoted as} \quad f_i = h_i(V0) \oplus h_i(Vt) \tag{4.3}$$

is satisfied, where h_i is the logic function corresponding to gate g_i's output. The average power dissipated in the network is the computed by using equation 4.2. The probability of the transition function f_i is calculated, resulting in a transition probability D_i. This value is inserted in equation 2.2 resulting in the power consumption for the transition of this gate.

4.3.4 Symbolic Simulation Example

Consider the multilevel combinational circuit shown in figure 4.1. It has four primary inputs and one output and consists of four CMOS gates. Figure 4.2 shows the signal transitions at the primary input nodes of the network. The time points are in normalized units. The first three inputs, $i1, i2$ and $i3$ switch simultaneously at time point 0. The fourth input, $i4$, is a late arriving signal that switches between the time points 5 and 6. In this example, the delays of both gate 1 and gate 2 are one time unit. Gate 2 has a delay of two units while gate 4 has a delay of four units. In figure 4.2 the waveforms e_i's, representing the signals at the outputs

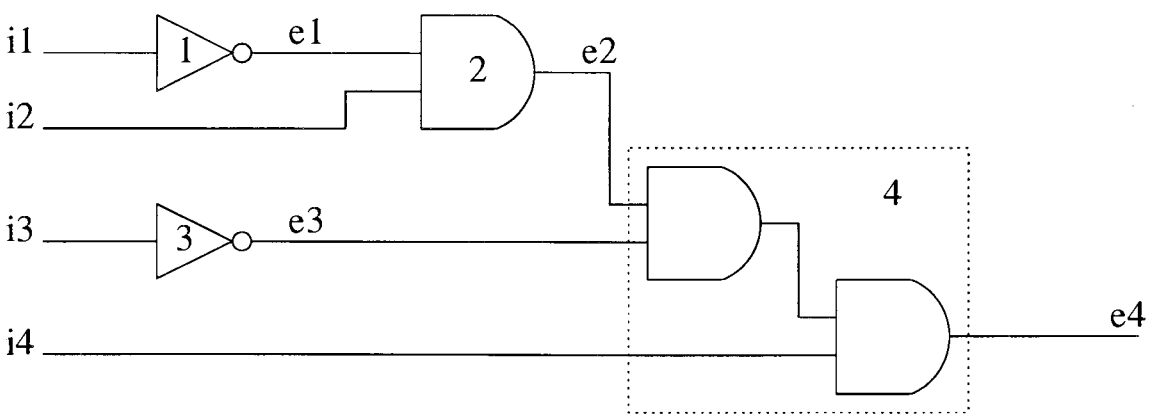

Figure 4.1: A multilevel combinational circuit

of the logic gates, are shown. Each of the possible transitions, $e_{i,j}$ represents either a low-to-high or high-to-low signal transition. The number of all possible transitions at a gate output

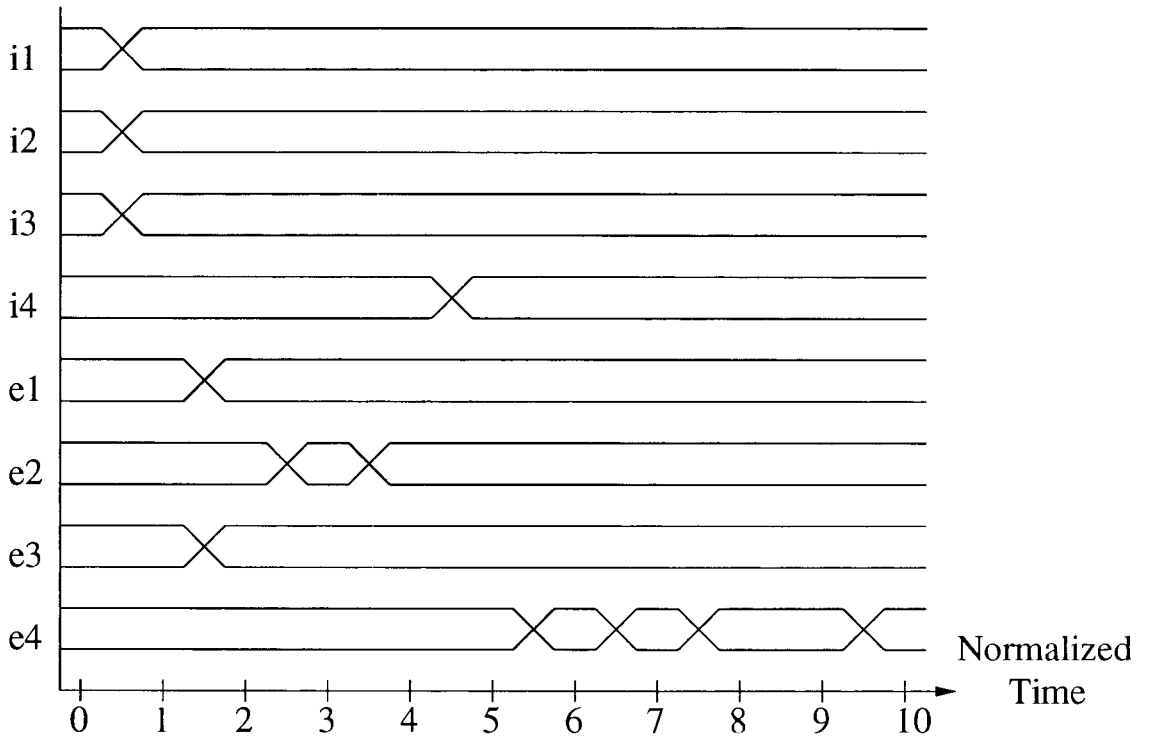

Figure 4.2: Signal waveforms

may equal the sum of all possible transitions at the gate inputs. These transitions are delayed by the gate's propagation delay.

4.4 Accuracy Aspects

Some effects must be taken into account for a correct and accurate power calculation. The most important effect is glitching. This effect can contribute significantly to the total power dissipation of a logic circuit. The most important effects are explained in this section, i.e.:

- glitching
- inertial delay
- temporal correlations
- spatial correlations

4.4.1 Glitching

A unit or general delay model must be used to detect glitches because there is no glitching in a zero delay model. To detect these glitches a description in time is needed beside the logical function. This time description indicates when signals switch state.

As can be seen from figure 4.1 and 4.2 all logic functions of the inputs of a gate must be known at any time in order to calculate the output. Using a symbolic simulation it is possible to calculate these states in time.

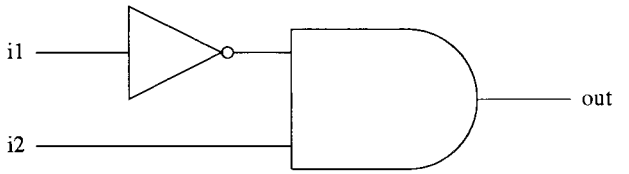

Figure 4.3: Glitching in a CMOS circuit

For example take the circuit of figure 4.3 which consists of an AND gate and an inverter, both having the same delay (1 time-unit). If the input vector is $< 0, 0 >$ and the circuit is in a stable state, then the output of the AND is zero. A transition in the next state to $< 1, 1 >$, results in an output state change one time unit later. Because of the delay of the inverter the output *out* becomes *zero* another time unit later. Thus changing the input vector from $< 0, 0 >$ to $< 1, 1 >$ results in two output transitions, a *zero* to *one* and a *one* to *zero* transition. Because the initial and final state of the output *out* are equal, two useless transitions have taken place in between: a glitch.

4.4.2 Inertial Delays

As seen in section 4.4.1 glitching can easily occur even in small circuits. In larger circuits where the delays of the gates differ a small fraction from each other, the number of glitches can increase significantly.

Beside the transport delay, there is also an inertial delay in every gate. This inertial delay, being the time the gate needs to change state, has a filtering effect on glitches. If the input pulse of a gate is shorter than the inertial delay, the gate cannot follow this pulse. In that case the output state remains unchanged. This filtering effect is important for estimating power and must be incorporated in the simulation model for obtaining more accurate values.

4.4.3 Temporal Correlations

Temporal correlation is the correlation of states of the input sequence of an input as a function of time. As example: after a *zero* to *one* transition the input stays one during the following 3 clock cycles. Temporal correlations require a fundamental knowledge of the input signal behavior in time, what is not always possible during the design phase of a circuit.

All models described in this thesis report don't take temporal correlations into account. Considering no temporal correlations makes the inputs independent. Therefore, the transition probability can be derived from the signal probability according to equation 5.4.

4.4.4 Spatial Correlations

Spatial correlations describe the relation between the internal signals of the circuit or between the inputs of the circuit. An example of a spatial correlation between inputs is that two in-

puts always have the same value. Internal spatial correlations, e.g. re-convergent outputs, are correlations which arise if an output signal of a gate is distributed to several logic blocks.

In a symbolic simulation the effect of internal spatial correlation is incorporated. However, none of the methods mentioned in this report make use of the spatial correlations on the inputs.

For example: the transition density of a two input AND gate is (see section 3.4.1):
$D(x) = P(b) \cdot D(a) + P(a) \cdot D(b)$.
The case where signal b has the opposite level of signal a resulting in a continuous output-level of *one* for this gate. This means that the output transition density must be zero. This example shows an overestimation if the spatial input correlation is not taken into account.

Consider the above described circuit only now with input a and b tied together. The output density must be equal to the input density because the AND gate is used as a buffer. But using equation $D(x) = P(b) \cdot D(a) + P(a) \cdot D(b)$ in this case changes to $D(x) = 2 * P(in) \cdot D(in)$. The expected relation would be $D(x) = D(in)$ thus the factor $2 * P(in)$ is responsible for the deviation of reality. When $2 * P(in)$ is larger than one this circuit gives an overestimation, when $2 * P(in)$ is smaller than one it gives an underestimation of the switching activity.

Chapter 5

Circuit Power Estimation

This chapter describes four different programs for evaluating switching activity. SLS and SIS are widely known programs. Log_dens and log_pattern are programs developed during this thesis project. Also the properties for setting up a simulation are described.

5.1 Programs for Power Estimation

5.1.1 SLS

SLS is a switch-level simulator, which can simulate the logical and timing behavior of digital CMOS circuits. The circuit to be simulated can be described at transistor level, gate level and functional level. SLS generates an output file with logic levels of all inputs and outputs of a circuit. For correct circuit activity simulation these levels must be converted to transitions. This is done by counting the *one* to *zero* and *zero* to *one* transitions for every signal. Using a general delay model requires a file with circuit capacities. To simulate the circuit, input vectors must be supplied. Using these vectors together with the given circuit capacities, the logic levels in the circuit are calculated. If a signal changes more than once its state between two succeeding input vectors, it is obviously a glitch. After all nodes have reached their stable state, the next input vector can be applied. This process must continue until the total number of transitions in a fixed time interval reaches a stable value. This method is unfortunately not possible with SLS because the total set of input vectors must be given to start the simulation. The simulation can't give intermediate results but the whole set of input vectors must be used to complete the simulation. The only way to use this simulation method is to apply a surplus of input vectors. A practical value giving good results even on circuits up-to 200 gates are about 2000 input vectors.

Because all states are explicitly calculated, internal spatial correlations (re-convergent output fanout) are automatically incorporated. Input vectors must be carefully chosen, they represent simultaneously input density, input correlations and input probability.

This switch-level-simulation gives realistic results about the circuit activity at every node. Therefore, this method provides reference values for comparing other methods in relation to accuracy, run time and glitching.

5.1.2 SIS

SIS is an algorithmic sequential circuit optimization program. In version 1.3 a power dissipation estimation package has been added, based on the method described by Ghosh in [2]. The average power dissipated in a circuit is given by:

$$P_{tot} = 0.5 * Vdd^2 * \sum_i (p_i * C_i) * f \tag{5.1}$$

where Vdd = 5V
f = 20MHz
p_i= expected number of transitions of node i in one clock cycle
C_i = capacitive load of node i

The expected number of transitions per clock cycle at each node is calculated through symbolic simulation, based on the static probabilities of the primary inputs (by default $P(x = 1) = P(x = 0) = 0.5$). The capacitive load of a node is obtained by summing the gate capacitances of its fanout nodes and adding some internal drain capacitance to it. This capacity value can also be given in an external data file. SIS supports three delay models: zero, unit and general.

For a typical gate the power is calculated by:

$$P_{avg} = 0.5 * C_{load} * (V_{dd}^2/T_{cyc}) * E(transitions) \tag{5.2}$$

where P_{avg} denotes the average power, C_{load} is the load capacitance, V_{dd} is the supply voltage, T_{cyc} is the global clock period, and $E(transitions)$ is the expected value of the number of gate output transitions per global clock cycle, or equivalently the average number of gate output transitions per clock cycle. All parameters in equation 5.1 can be determined from technology or circuit layout information except $E(transitions)$, which depends on the logic function being performed and the statistical properties of the primary inputs.

To compute $E(transitions)$, the power estimation package uses the symbolic simulation method. Under a zero delay, unit delay or general delay model (where delays are obtained from library cells), the symbolic simulation method takes into account the correlation due to reconvergence of input signals and accurately measures the switching activity, according to some given input probabilities.

5.1.3 Log_dens

During this thesis project two power calculation tools were developed. Both are based on Binary Decision Diagrams (BDDs) [32].

The first tool, called *log_dens*, incorporates a zero delay and a general delay model. The zero delay model calculates the dissipated circuit power using the boolean difference technique, the general delay model uses symbolic simulation.

Zero Delay Model

To explain the internals of zero delay power estimation program it is described using the following pseudo code:

Program log_dens (zero delay model)
{
Parse the .slif file which describes the circuit in boolean logic.

Parse experiment file. This file contains the probability and transition density for every input. Also the capacities for every node are declared in this file.

Build the BDD for every output of the circuit.

For all outputs do
{
determine the support set for this output.
for x in support set do
{
trans = boolean difference of this output to input x
calculate the probability of trans
add to this output density: probability $*$ input density
}
}

Dissipated power of a output is: transition density $*$ node capacitance.

Output results: total transition density, total dissipated power.
}

General Delay Model

The general delay based method in log_dens shares the input parsing section with the zero delay program.

Because this method is based on symbolic simulation it is not possible to build the BDD for the circuit outputs directly. These output BDDs are build as last step in the simulation.

The calculation starts by determining the initial and next state value of all inputs. The transition times of the successors of each input are put in the Event list. These times are found by using the capacitive load of the gate as delay value.

When all inputs have made their transition from the initial state to the next state the first signal going to change must be found. This signal is the first event in time from the Event list.

The event is processed by exclusive-OR'ing the old state value and the next state value, which results in the transition function. The probability of this function is determined resulting in a transition density. This value is multiplied with the node capacity and the result is stored in the signal.

The next step is putting all the succeeding signals of this node at the correct time in the Event list. This time is found by adding the current delay value to the current time in the simulation.

The next signal to process is searched in the Event list and is processed analog to the above. This continues until the Event list is empty.

Finally some results including the total estimated power consumption and the total transition

density are printed.

5.1.4 Log_pattern

The log_pattern program is also based on symbolic simulation. The symbolic simulation uses a different approach as the log_dens program. It is also possible to use the inertial delay filtering effect for short glitch removal.

The symbolic simulation builds a signal pattern, describing the states for a single node in time. Thus log_pattern knows all states for a single node for any time, in contrast to log_dens which knows the states for all nodes at one time. Knowing this time behavior it is possible to move a window over the pattern, having a width equal to the inertial delay for filtering out short glitches.

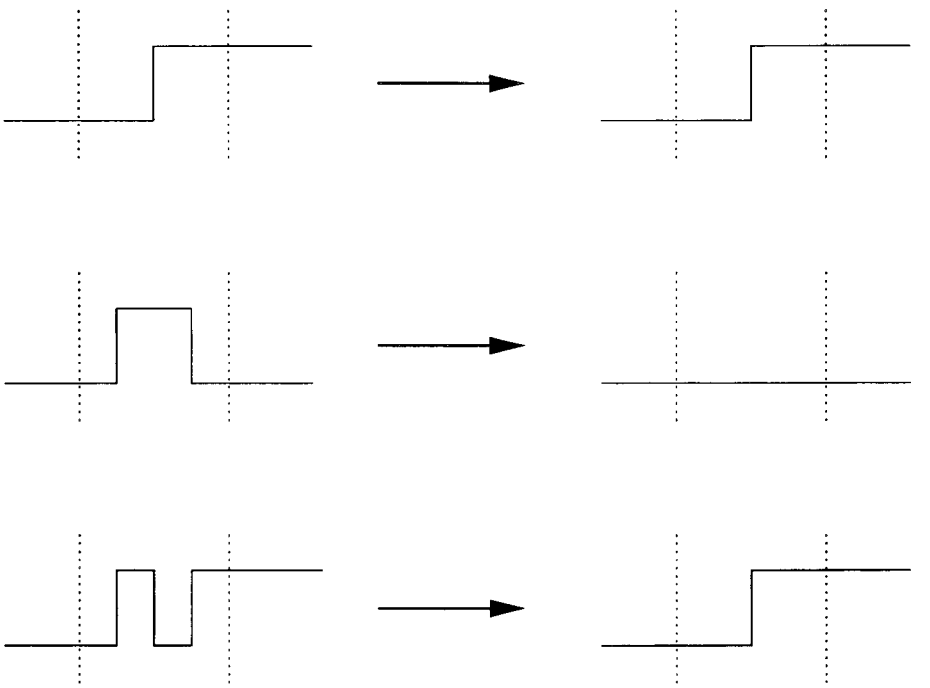

Figure 5.1: Glitch removal in log_pattern

As can be seen in figure 5.1 signals which change state and return to their initial state within the inertial delay are not propagated through the circuit. The disadvantage of this method is an increase in run time for circuits with a lot of glitching, because this is an expensive method in terms of run time.

The value used for the inertial delay is one half of the transport delay. This is a very coarse prediction of reality. By using more accurate values the accuracy of the glitch removal increases.

5.2 Input Vector Generation

Generation of input vectors for the SLS method can be divided into two categories. The first category is changing only one input in every input-vector. The other one is changing a number of inputs simultaneously. This is equivalent to respectively single input switching and multiple input switching.

Because the temporal correlations of the inputs are not taken into account, a relation between probability and density can be found. Evaluating a binary Markov source, figure 5.2, gives the requested relation.

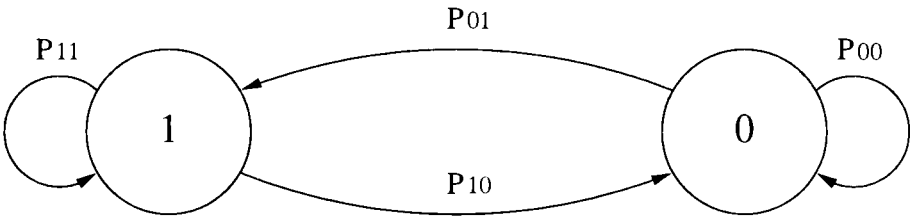

Figure 5.2: Binary Markov source

A transition occurs if the variable is in the *one* state (probability $P_{11} = P(x = 1)$) and the next state is the *zero* state (probability $P_{00} = P(x = 0)$). Thus the probability of changing from *one* to *zero* is:

$$P_{10} = P(x = 1) * P(x = 0). \tag{5.3}$$

If the initial state is *zero* a similar strategy can be followed resulting in a state change probability of: $P_{01} = P(x = 1) * P(x = 0)$. The probability for a signal to change its state is: $P_{01} + P_{10} = 2 * P_{01} = 2 * P(x = 1) * P(x = 0) = 2 * P(x = 1) * (1 - P(x = 1))$. This results into:

$$D(x) = 2 * P(x = 1) * (1 - P(x = 1)) \tag{5.4}$$

Generation of the input vectors for SLS, representing the single switching model is simple. The first step is to generate a random vector, being the initial state. For every next time step one input state in the vector may change. The probability of this change is 0.5. The following time step an other or even the same input may change its state. This is repeated for as many vectors as needed. The transition density for an input is not 0.5 as shown in equation 5.4 but lower. This is because an input can change its state on average once in every n vectors, where n is the number of inputs. Thus, a single input model based on input probability 0.5 gives an input transition density of $0.5/n$.

Input vectors representing multiple input switching don't have this problem, because every input can change its state in each input vector. The transition density for every input is found by using equation 5.4. For every input this results in a probability of 0.5 and a transition density of 0.5.

5.3 Capacities

To obtain estimated values for node capacities a critical path analysis using log_crit can be performed. The net-capacities and other properties for a chosen technology are estimated. The capacity values are given a random element (-50% .. +100%) representing tolerances in technology and length of interconnect wires. The capacities are very important for the total power calculation because the dissipated power is proportional to these values (see equation 5.2). The main problem is to use more realistic values but in this early design stage the wire lengths and gate capacities are not known. The best option is to use a value based on the gate's fanout and to use a margin of (-50% .. +100%) representing technology and net-wiring tolerances.

5.4 Statistics

To obtain the sensitivity for small changes in circuit parameters all experiments have been simulated 20 times. During these simulations the input probabilities and densities remain constant. Only the delay values for every gate, caused by the capacitances, are changed (see section 5.3).

The average, maximum and minimum values from the circuit experiments are stored and the standard deviation, a measure for the sensitivity, is then calculated.

5.5 Glitching

Power estimation method based on unit or general delay can detect glitches.

Switch-level-simulation can detect glitches very well, because in a glitch free circuit every circuit node may change its state maximally one time between two successive input vectors. If a signal would change two times, it returns to its original value and therefore, makes a useless transition (see section 4.4.1).

SIS has unfortunately no possibility to take the glitching effect into account. It is expected that this method overestimates the circuit switching activity because the useless transitions are not detected.

The log_dens program suffers from the same overestimation problem because it doesn't take the inertial delays into account.

The log_pattern method can detect glitches and uses the inertial delay filter effect to remove short glitches. It is expected this method has less overestimation than the log_dens program and therefore, a lower total estimated power.

Using a zero inertial delay for log_pattern, the power estimation results are exactly the same as the general delay log_dens method. Therefore, the amount of power consumption by glitches, which are filtered out by the inertial delay, causes the difference between the consumed power of log_dens and log_pattern with an non-zero inertial delay.

5.6 State Probability

To test the different power calculation methods on their correctness, a simulation for small circuits can be done. The result of these simulations can be compared to theoretically obtained values. This section describes how the transition probabilities are found for small circuits e.g.: inverter, two input NAND and a 3 input NOR gate.

Inverter:

Table 5.1: Inverter states

step	zero state	output	next state	output	single
0	0	1	0	1	*
1	0	1	1	0	*
2	1	0	0	1	*
3	1	0	1	0	*

Assuming an input probability of $P(x = 1) = 0.5$ results in a state probability of 0.25 (there are four states). The probability that the output makes no state change (step 0 and 3) is $2 * 0.25 = 0.5$. The probability that the output makes a state change (step 1 and 2) is $2*0.25 = 0.5$. Thus for an inverter with input probability 0.5, the chance of an output state transition is 0.5.

Two input NAND gate:

For a two input NAND gate these state calculations can be made, but here it must be taken into account whether a single input transition model or a multiple input transition model is used. The state vectors can be seen in table 5.2. The column 'zero state' denotes the initial input vector and the column 'next state' denotes the next state vector. A '01' in these columns means input 1 has a *zero* level and input 2 has a *one* level. The rows marked with an asterisk are state transitions where only one input changes its state (single input switching). This example uses ($P_{in}(x = 1) = 0.5$).

Evaluating the output transition as function of a multiple input transition gives:
$0 \rightarrow 0 = \frac{1}{16}$ $0 \rightarrow 1 = \frac{3}{16}$
$1 \rightarrow 1 = \frac{9}{16}$ $1 \rightarrow 0 = \frac{3}{16}$
This results in a probability of an output state transition of $\frac{3}{16} + \frac{3}{16} = 0.375$. The probability that the output state remains equal is $\frac{1}{16} + \frac{9}{16} = 0.625$.

For evaluating the output state changes when using *single* input transitions the rows marked with asterisks are used. Because all these states are equally likely to occur, the probability of every step in the table is $\frac{1}{12}$. Hence:
$0 \rightarrow 0 = \frac{1}{12}$ $0 \rightarrow 1 = \frac{2}{12}$
$1 \rightarrow 1 = \frac{7}{12}$ $1 \rightarrow 0 = \frac{2}{12}$
This results in a probability of an output state transition of $\frac{2}{12} + \frac{2}{12} = 0.25$. The probability that the output doesn't change state is $\frac{1}{12} + \frac{7}{12} = 0.75$.

Three input NOR gate:

For a three input NOR gate similar state transitions can be found. Considering a multiple input switching model there are 64 states, 8 initial states $*$ 8 next-states. The probability of every state is $\frac{1}{64}$. The probability for no change of output state is $\frac{49}{64} + \frac{1}{64} = 0.781$. The probability of an output state transition is $\frac{7}{64} + \frac{7}{64} = 0.219$.

Table 5.2: NAND states

step	zero state	output	next state	output	single
0	00	1	00	1	*
1	00	1	01	1	*
2	00	1	10	1	*
3	00	1	11	0	
4	01	1	00	1	*
5	01	1	01	1	*
6	01	1	10	1	
7	01	1	11	0	*
8	10	1	00	1	*
9	10	1	01	1	
10	10	1	10	1	*
11	10	1	11	0	*
12	11	0	00	1	
13	11	0	01	1	*
14	11	0	10	1	*
15	11	0	11	0	*

The number of transitions that can be made using a single input transition model is 32. The probability of a non output state change is $\frac{26}{32} + \frac{1}{32} = 0.844$. The probability of an output state transition is $\frac{2}{32} + \frac{3}{32} = 0.156$

It is expected that transition densities found with SLS simulation, using single and multiple input switching, are close to these theoretical values. Table 5.3 shows the theoretical values and the simulated values using SLS.

Table 5.3: State change probability

Switching method	Inverter	2-input NAND	3-input NOR
Theory			
single input	0.5	0.25	0.156
multiple input	0.5	0.375	0.219
Simulation			
SLS single	0.521	0.269	0.135
SLS multiple	0.510	0.364	0.220

5.7 Overview Methods and Programs

This section shows an overview for all described delay models and accuracy effects. In table 5.4 the relations are given between the simulation and estimation programs and the effects or methods. The accuracy row gives a measure for the different calculation methods compared to SLS multiple.

Table 5.4: Overview power estimation methods

	SLS single	SLS multiple	SIS	log_dens	log_pattern	zero delay
calculation						
simulation	yes	yes	no	no	no	no
probabilistic estimation	no	no	yes	yes	yes	yes
input model						
single input switching	yes	no	no	no	no	yes
multiple input switching	no	yes	yes	yes	yes	no
symbolic simulation	no	no	yes	yes	yes	no
effects						
glitching	yes	yes	yes	yes	yes	no
inertial delay	yes	yes	no	no	yes	no
correlations						
re-convergent outputs	yes	yes	yes	yes	yes	no
temporal correlations	no	no	no	no	no	no
run time	high	high	moderate	low	low	very low
accuracy	low	reference	moderate	moderate	high	low

Chapter 6

Results

6.1 Test Circuits

To compare the power estimation programs, small circuits from the LGSynth91 Combinational Multi Level benchmark set are used. A few special test circuits, not in the LGSynth91 test set, are added to test the glitching effect. All the circuits are technology mapped using SIS on a library consisting of an inverter, 2, 3 and 4 input NAND gates and 2, 3 and 4 input NOR gates. The circuit sizes of this test set can be seen in table 6.1.

Circuit 0, 1, 2 and 3 are single gate circuits. These are used to test the accuracy of the calculation programs. Glitching is not possible in a single gate, therefore, the single input switching calculation methods must have exactly the same dissipated power. The multiple input switching programs must also have exactly the same results for these circuits. For the inverter the single and multiple switching methods must have the same results because there is only one input.

Circuit 4, 5, 7 and 8 are not a part of the LGSynth91 benchmark set. These circuits are used to detect glitch activity. The schematic layout of these circuits is given in appendix A.1.

The xorglitch circuit, is a very interesting circuit (for a schematic layout see appendix A.1). It is a mapped XOR gate, with one input connected to the output of an inverter. The other input of the XOR gate and the input of the inverter are tied together and form the only circuit input. This circuit produces *only* glitches. It is useful for testing the inertial delay filtering.

Table 6.1: Circuit sizes

no.	circuit	#inputs	#outputs	#gates
0	inverter[1]	1	1	1
1	nor-2[1]	2	1	1
2	nand-2[1]	2	1	1
3	nor-3[1]	3	1	1
4	nandtreesmall[2]	4	1	3
5	xorglitch[2]	1	1	6
6	C17	5	2	6
7	nandtree[2]	8	1	7
8	overestimate[2]	3	3	9
9	majority	5	1	10
10	nandtreebig[2]	16	1	15
11	cm138a	6	8	17
12	cm42a	4	10	21
13	cm152a	11	1	24
14	cm82a	5	3	28
15	decod	5	16	34
16	cmb	16	4	43
17	cm163a	16	5	50
18	cu	14	11	51
19	cm162a	14	5	53
20	z4ml	7	4	65
21	parity	16	1	75
22	cm150a	21	82	82
23	pcler8	27	17	101
24	mux	21	1	104
25	ldd	9	19	104
26	unreg	36	16	126
27	lal	26	18	128
28	count	35	16	167
29	c8	28	17	178

[1] single gates.

[2] these circuits are described in Appendix A.1.

6.2 Simulation Results

6.2.1 Power Dissipation

Figure 6.1 shows the average dissipated power obtained with different calculation methods. The x-axis shows the circuit numbers in increasing circuit size.

The first observation from this figure is that all circuits dissipate more power with increasing circuit size. An other observation is that the parity circuit (number 21) shows big differences for the estimated power using different methods. The schematic layout of this circuit can be found in appendix A.2.

The multiple input simulation method using SLS is seen as the ideal power calculation result. Figure 6.2 shows the estimated power for the other methods normalized on these SLS multiple values.

The first observation from this figure is the large power peak of the parity circuit (21). The second observation is that all methods are above the SLS multiple normalization line. This means that all methods overestimate the dissipated circuit power. The power of single gate circuits (1, 2, 3) is overestimated by both single input calculation methods. This is because every input contribution is seen independently of the other inputs as a result of using the boolean difference. Symbolic simulation does take this spatial correlation effect into account. Therefore, the results of log_dens and log_pattern are exactly equal to the SLS multiple power.

In figure 6.3 the comparison of the single switching methods is seen. For all examples the zero delay simulation is on average a few percent below the SLS single value except for the xorglitch circuit (5).

Simultaneous input switching is compared in figure 6.4. The exact simulation results for single gates is logical, because glitches cannot occur. Circuits with glitching show a deviation of the SLS multiple value. For example the xorglitch circuit (5) shows this because this circuit produces only glitches at its output.

The parity circuit (21), is an extreme example. Log_dens overestimates the power by 100%. Log_pattern filters the glitches but is still 60% above the real value.

Generally the log_pattern estimation methods is better than the log_dens estimation method. Log_pattern estimates the power within 25% overestimation for normal circuits. Circuits with much XOR functionality are not accurately estimated, for example circuits 5, 21 and 25.

Finally the total estimated power is shown in figures 6.6, 6.7, 6.8, 6.9 and 6.10 for all the different estimation methods. These figures show the average, maximal and minimal values for the estimated power of 20 simulation runs. Randomizing the circuit capacities in a range of (-50% .. +100%) results in minimal and maximal values of the estimated power which slightly differ from the average power.

The circuits are also simulated using the SIS program. Because the results are exactly the same as the log_dens results, SIS is not shown in these figures.

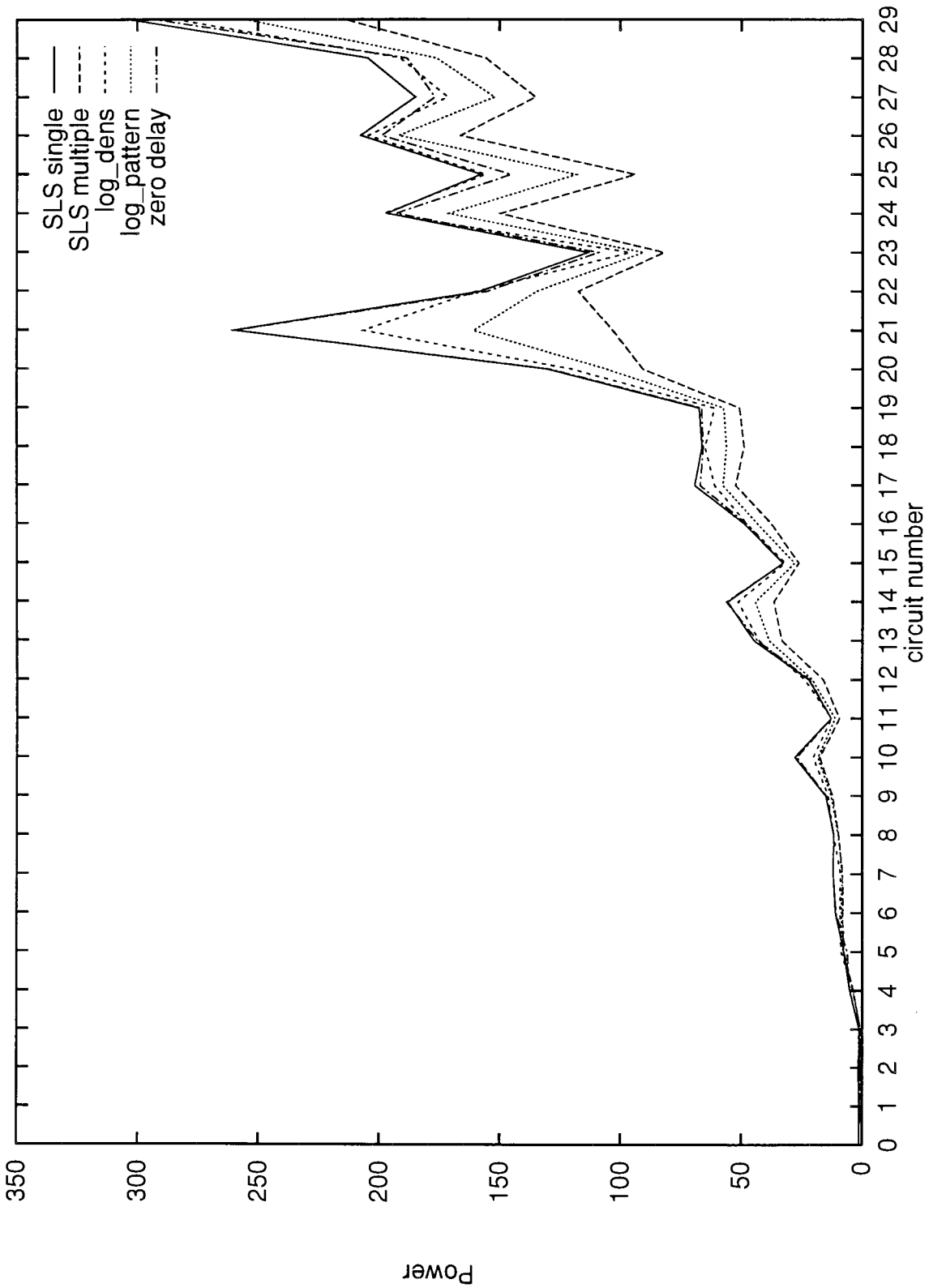

Figure 6.1: Average dissipated power

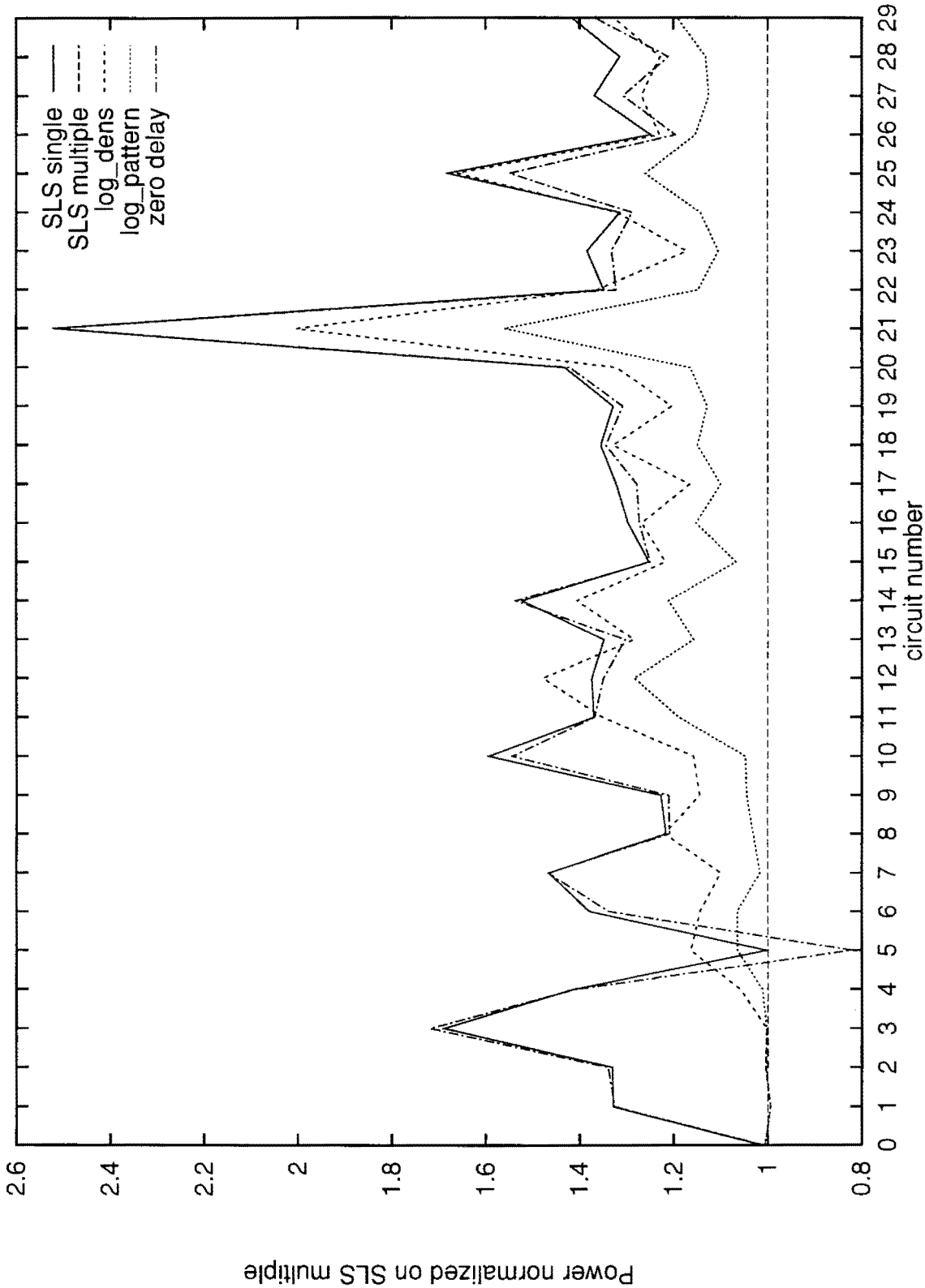

Figure 6.2: Power normalized on SLS multiple

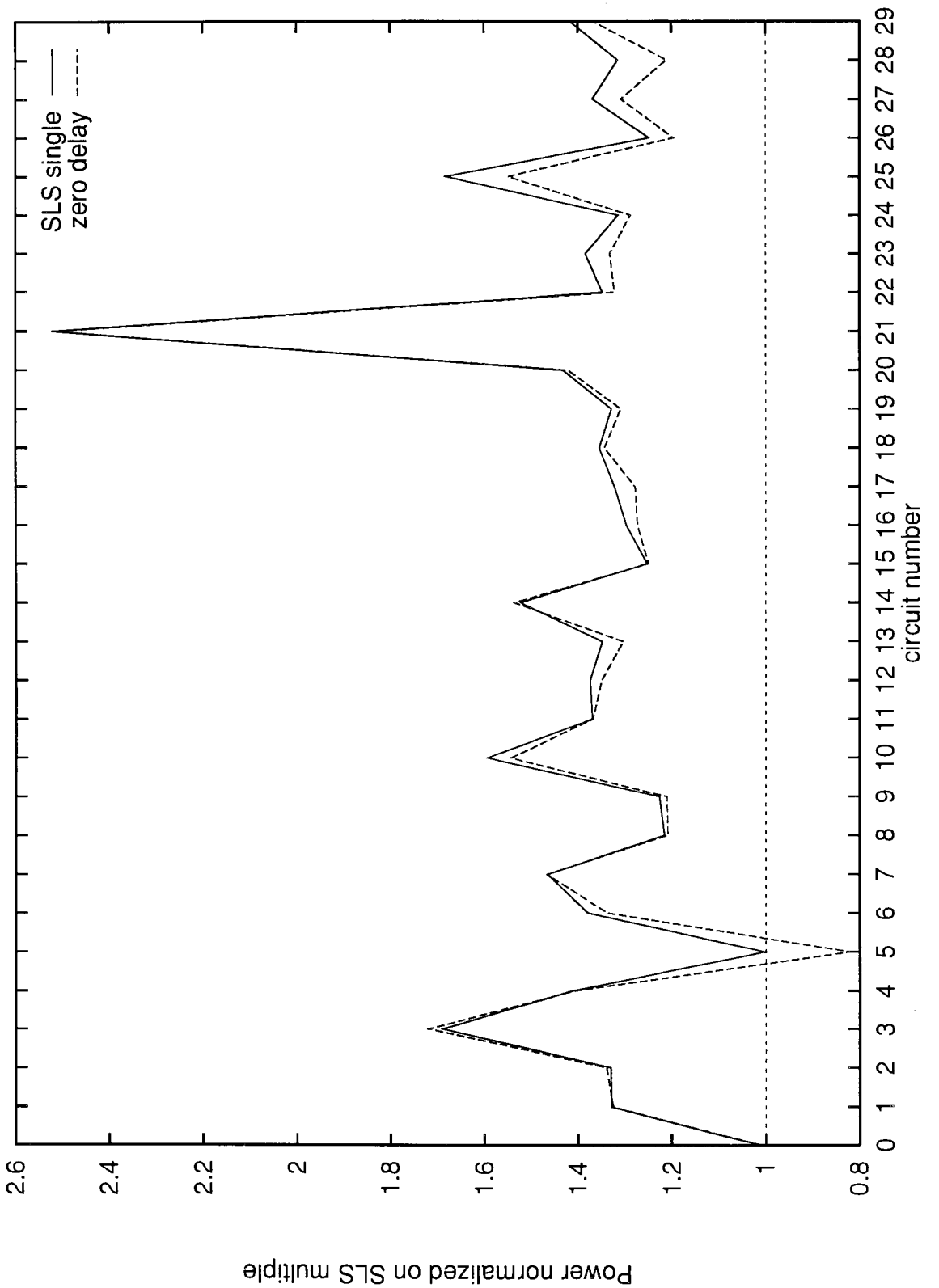

Figure 6.3: Normalized single switching power

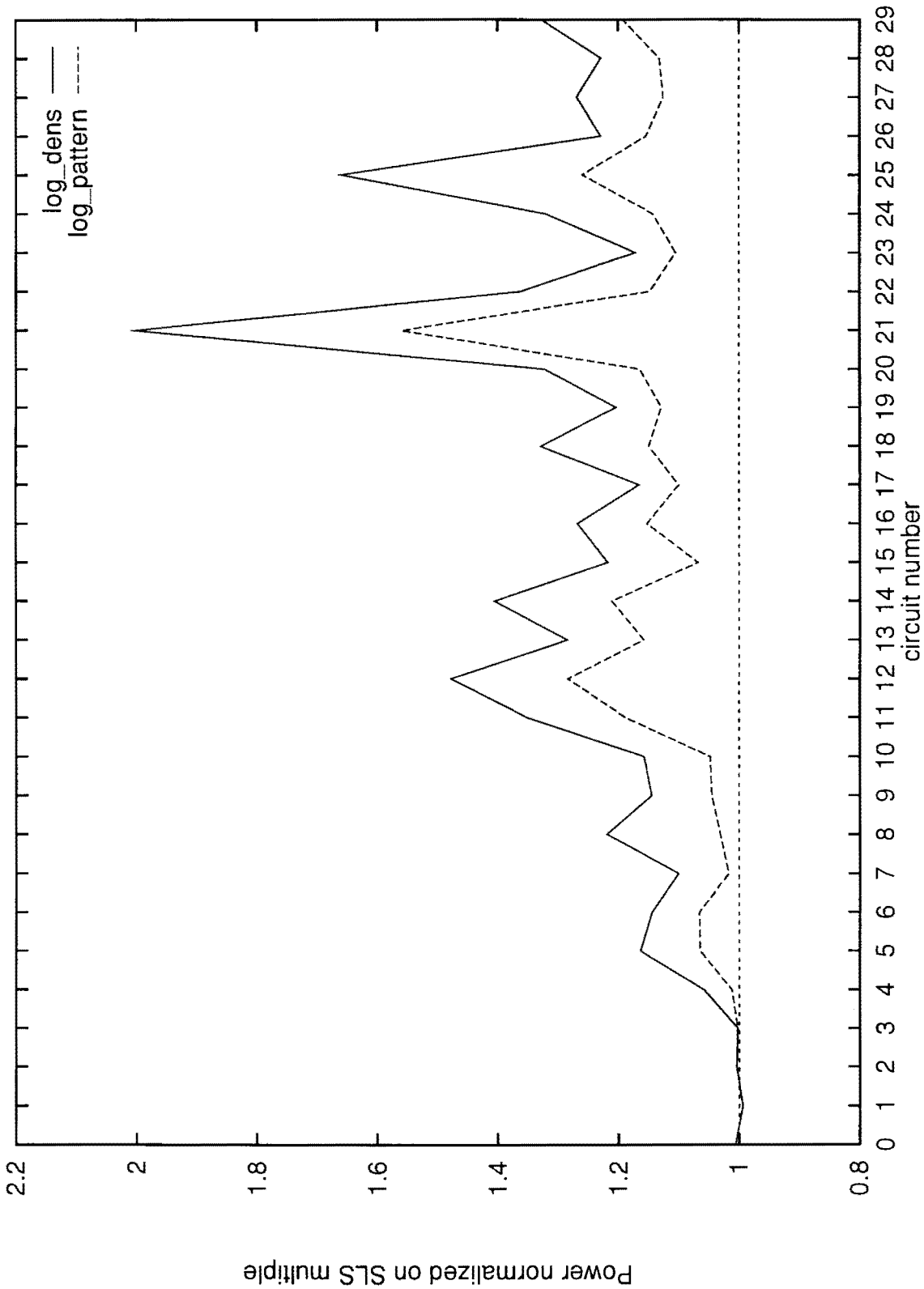

Figure 6.4: Normalized multiple switching power

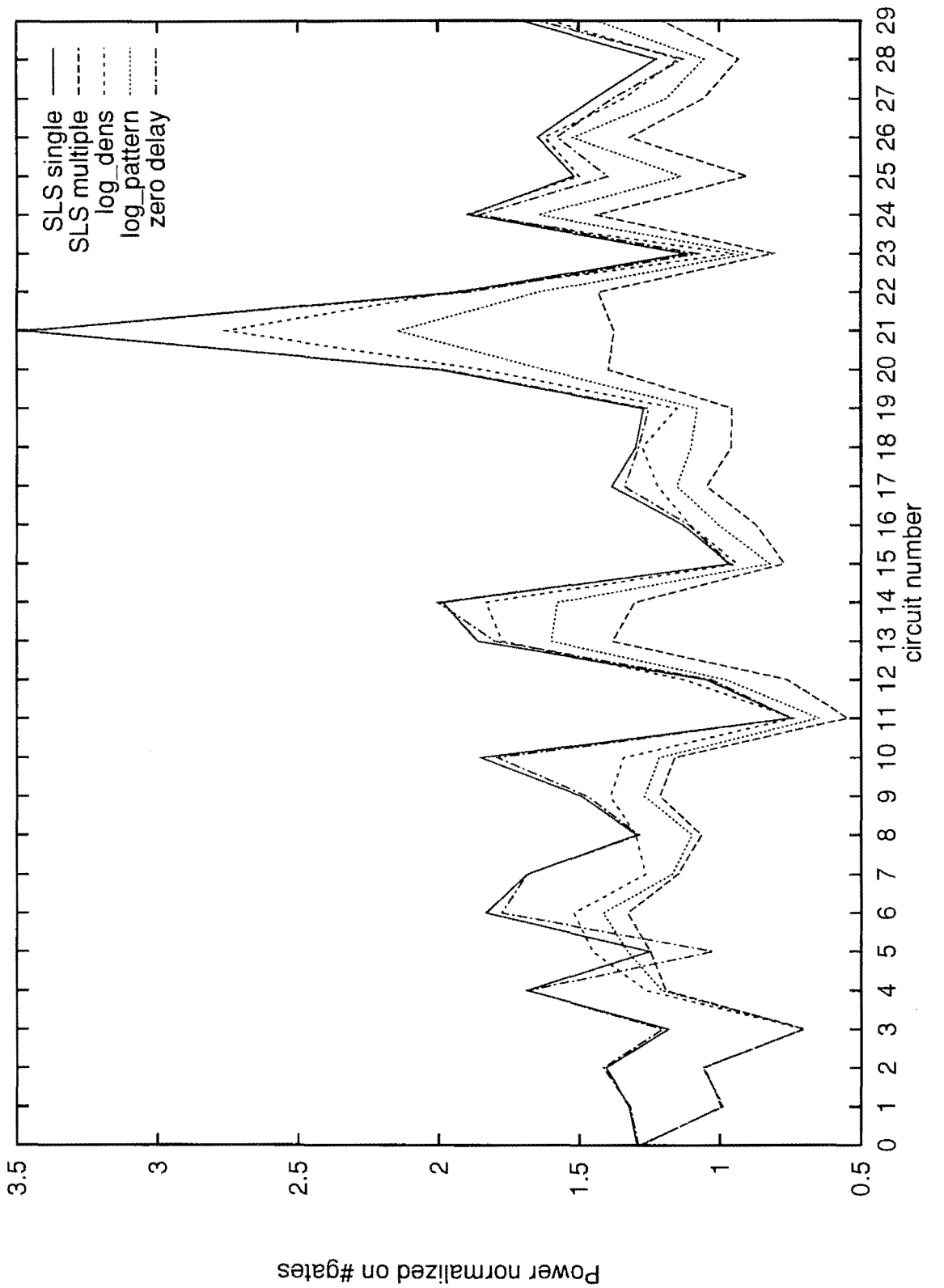

Figure 6.5: Power normalized on #gates

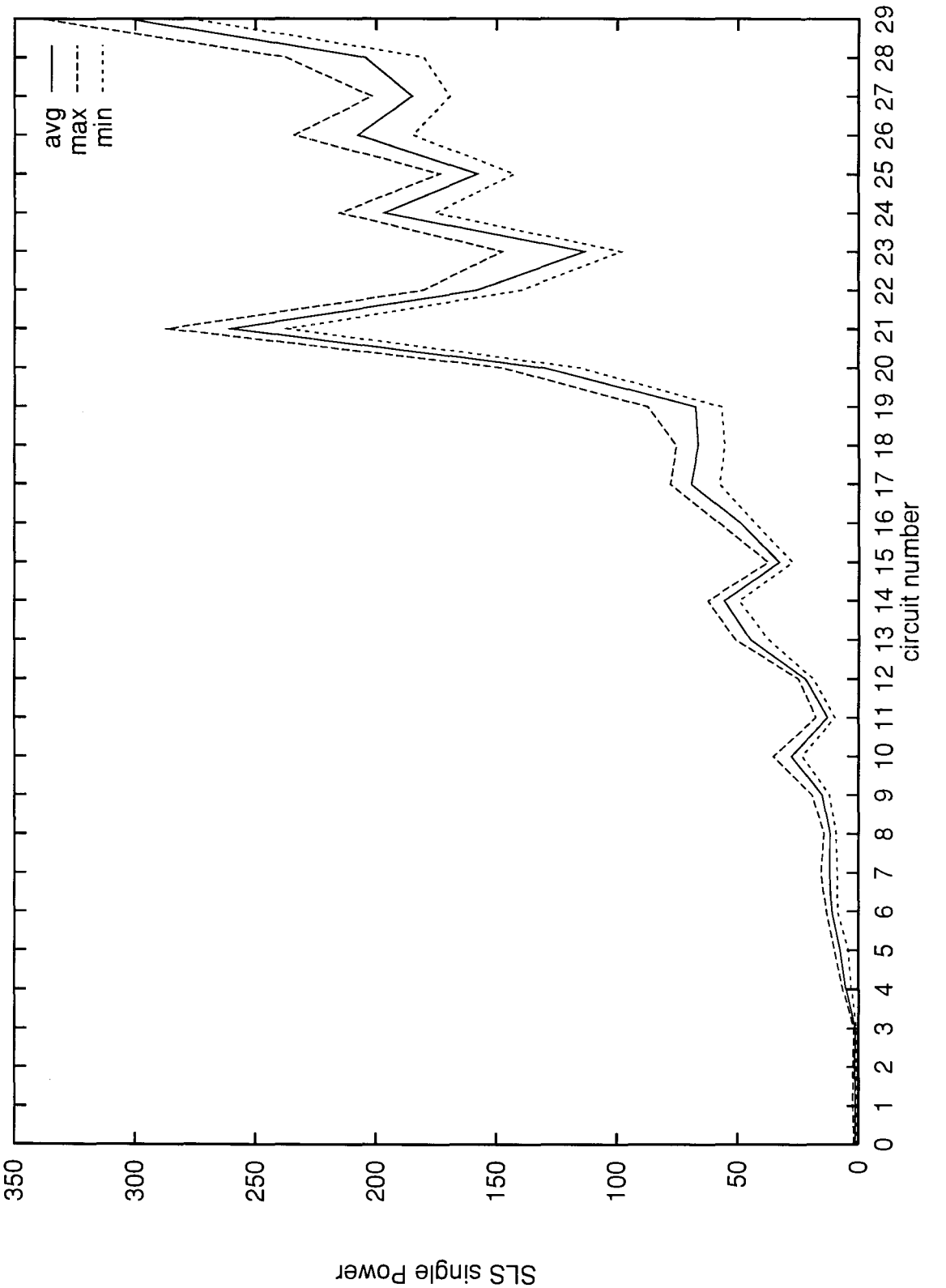

Figure 6.6: SLS single power

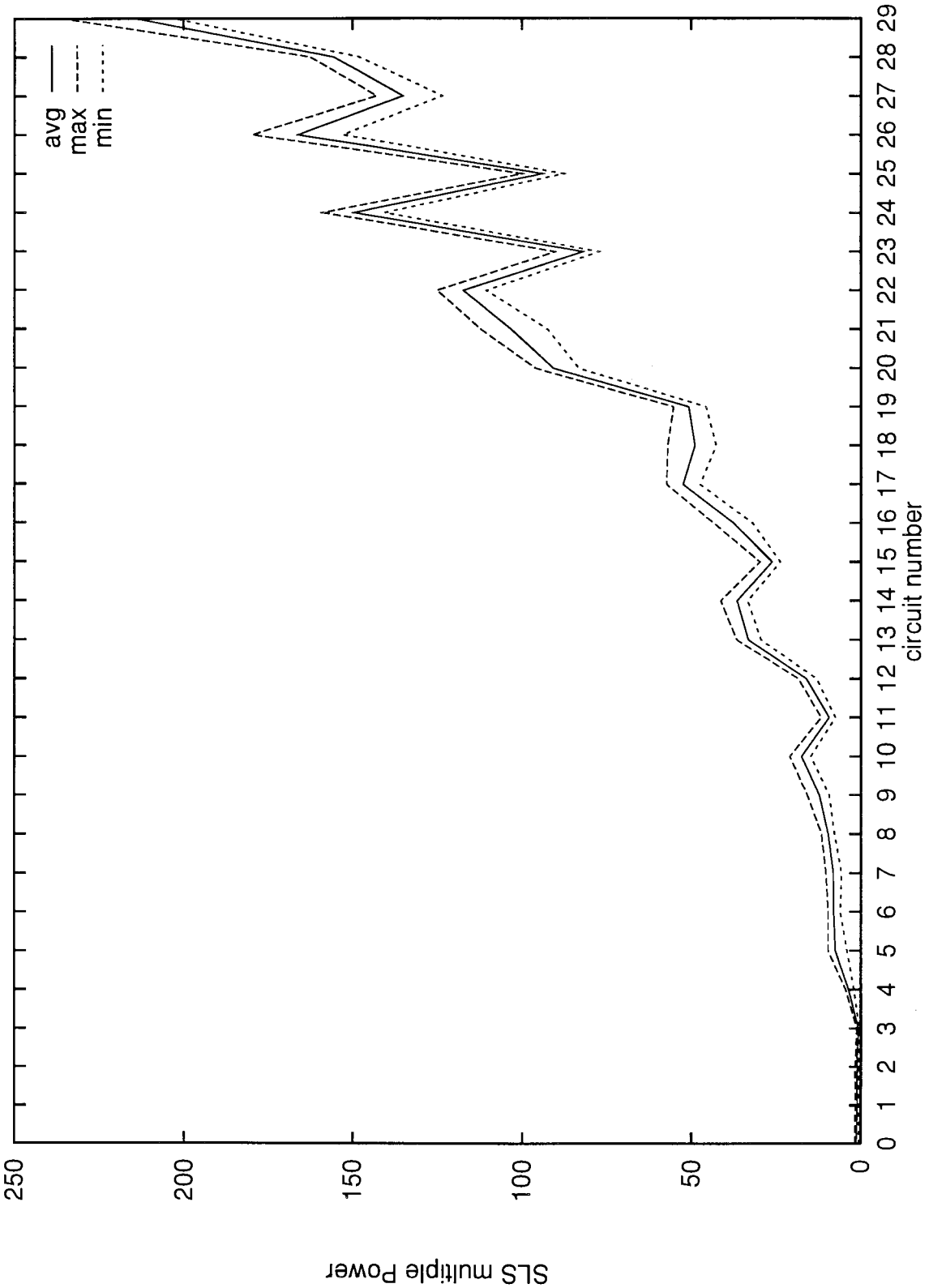

Figure 6.7: SLS multiple power

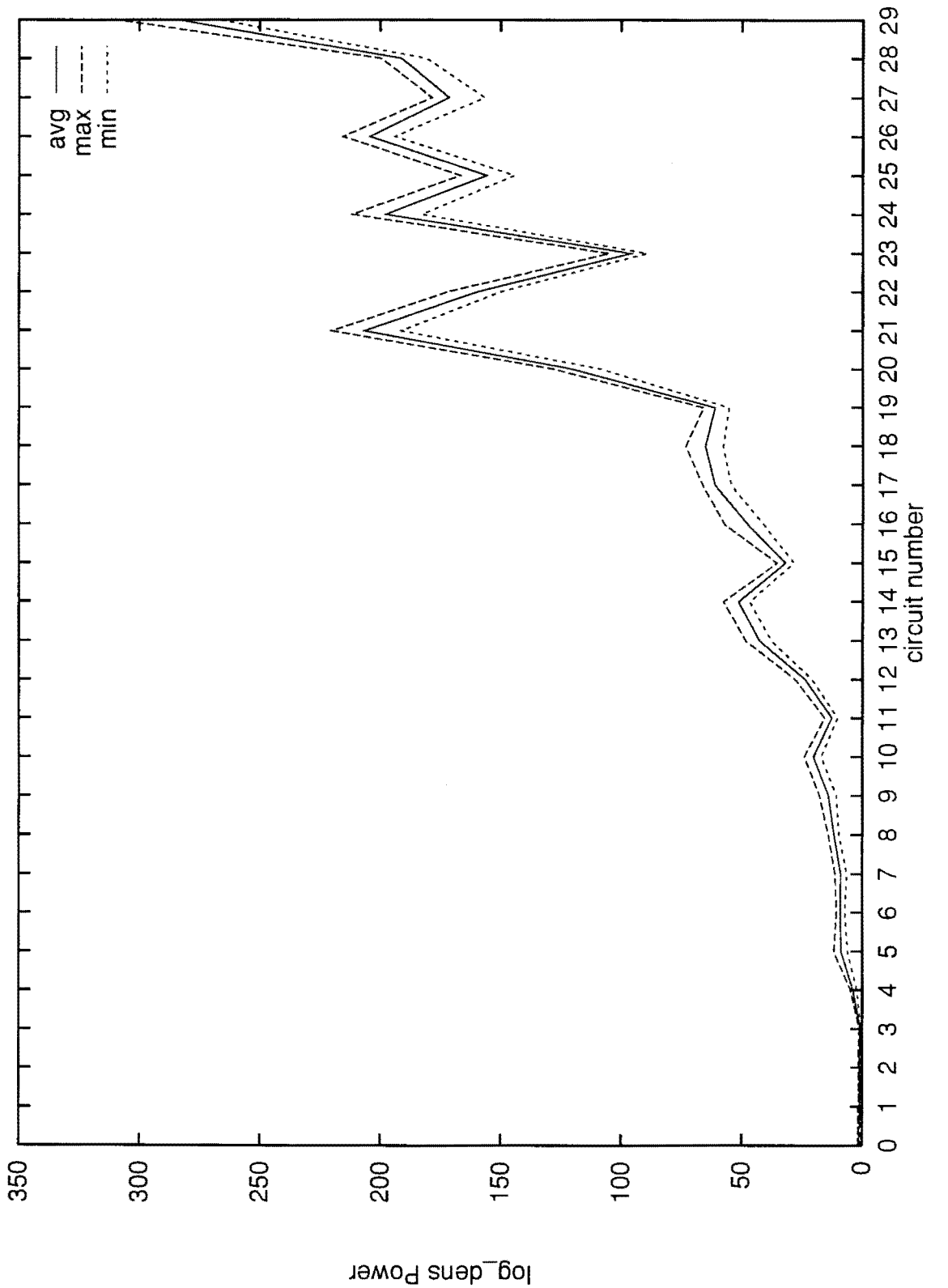

Figure 6.8: Log_dens power

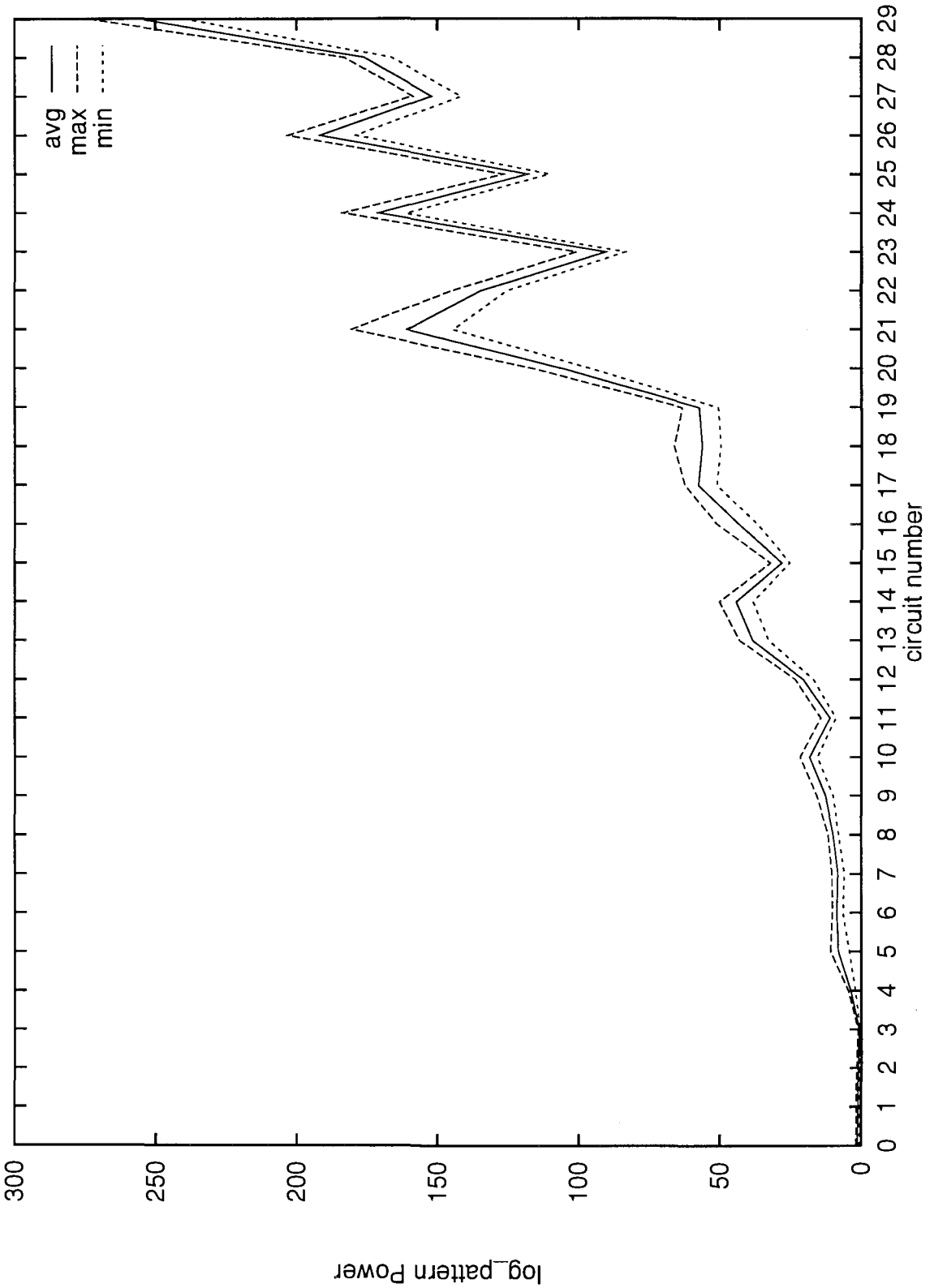

Figure 6.9: Log_pattern power

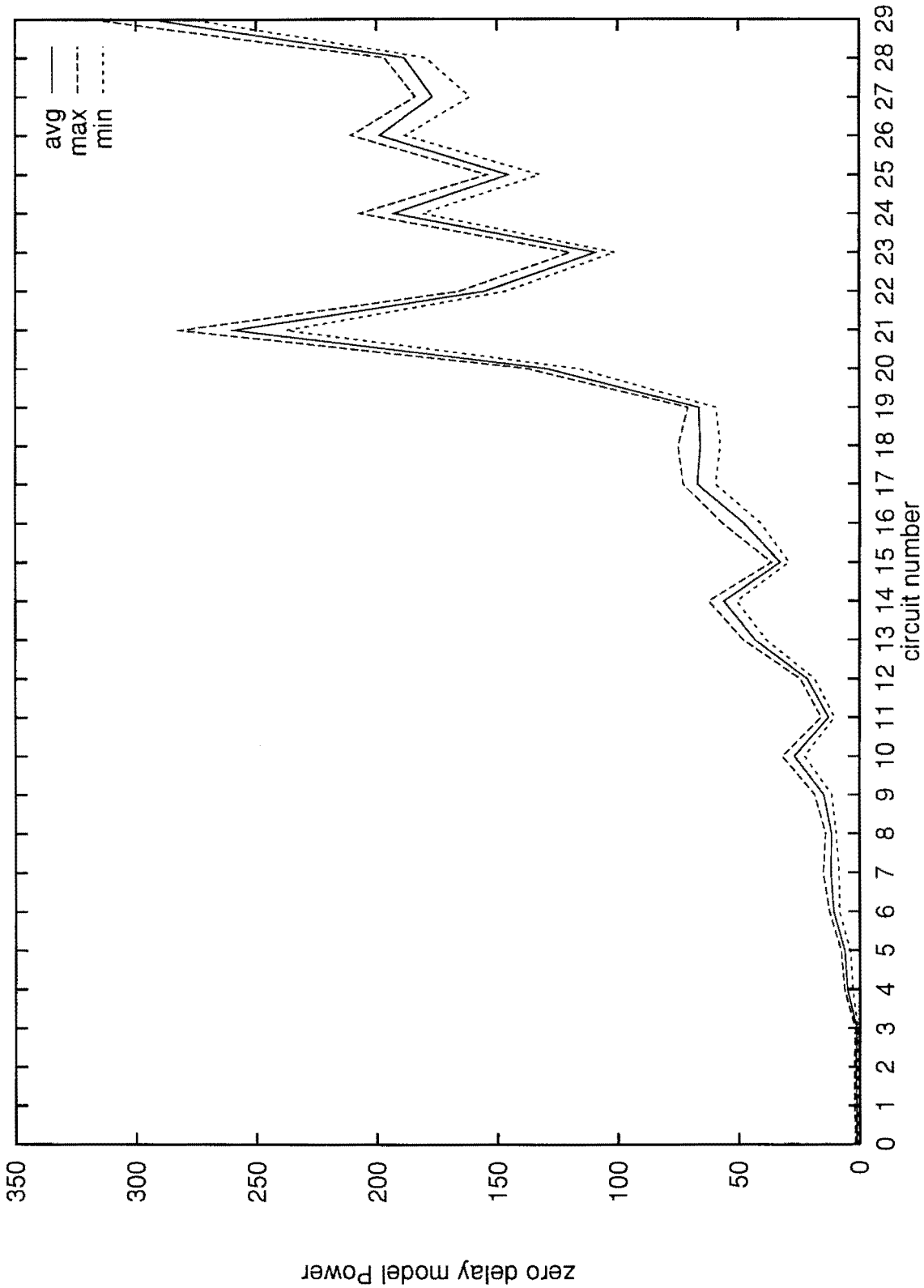

Figure 6.10: Zero delay power

6.2.2 Circuit Glitching

Figure 6.11 shows the glitching in percent of the total number of transitions. The SLS multiple method experiences more glitching than the SLS single method. This is because more inputs switch simultaneously in SLS multiple, therefore, the chance of a glitch anywhere in the circuit increases.

The xorglitch circuit (5) shows an equal result for both methods regarding the glitching percentage, this is logical because this circuit has only one input.

The overall glitching factor obtained with the SLS multiple method is less than 10% of the total number of transitions. Even the parity circuit (21) only glitches about 7%. This is surprisingly low considering the XOR circuit structure.

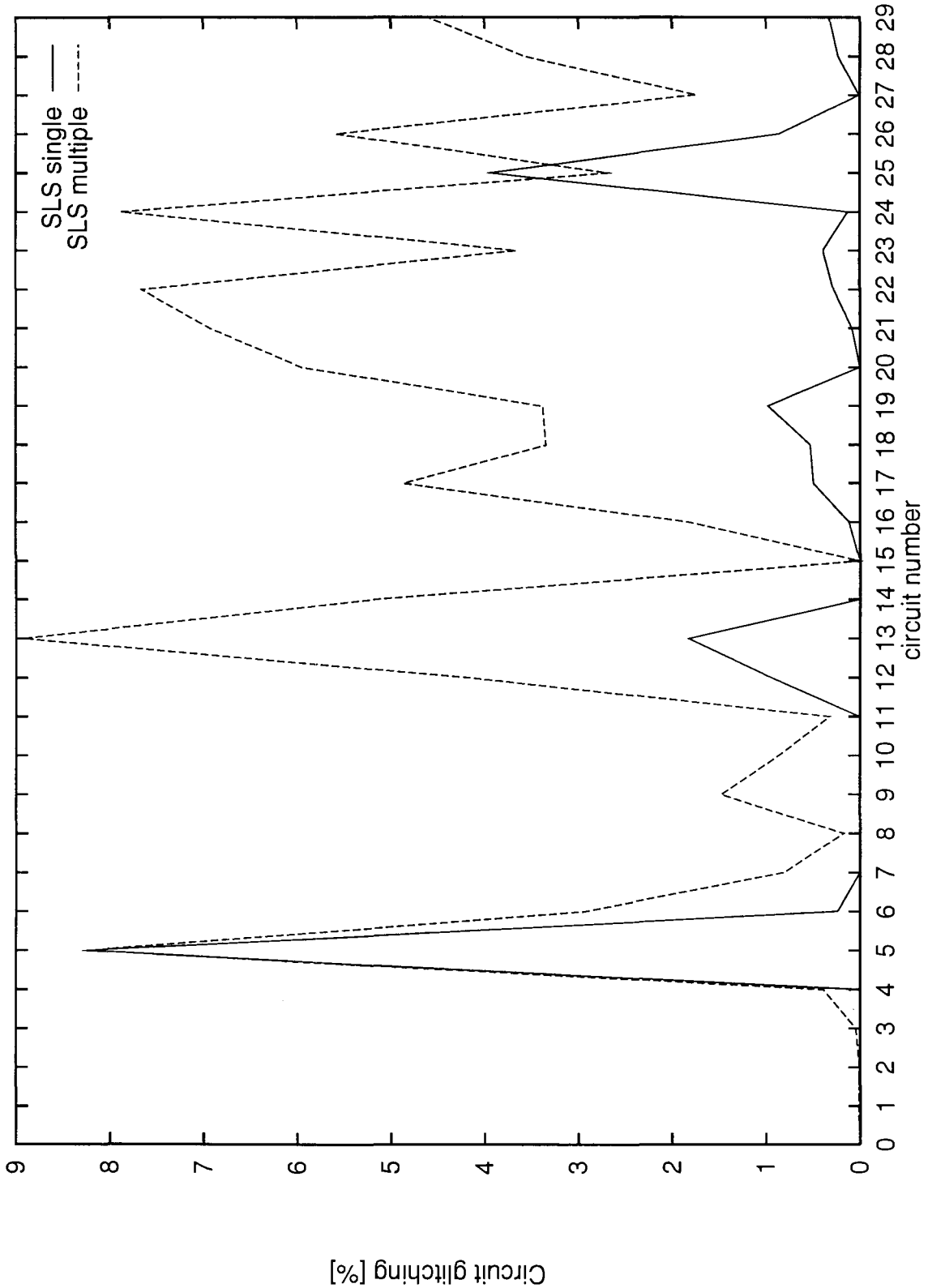

Figure 6.11: Average glitching

6.2.3 Statistical Effects

Figure 6.12 shows the statistical effect on the total estimated circuit power when the node capacities are randomized in an interval of (-50% to +100%). This figure shows the standard deviation of 20 simulation runs using different node capacities in every run.

The first observation is that the SLS single curve has a higher SD than the others meaning this method is more sensible for variations in node capacities. The SLS single curve gives an upper limit for the SD and the SLS single method gives a lower limit.

The BDD based methods are all about equally sensitive to node capacity variations. The parity circuit (21) is the only exception. For this specific circuit the zero delay method shows a relative larger sensitivity for variations than the other methods.

Figure 6.13 shows the SD but now in percent of the total circuit power. It can be seen that the relative SD is lower in larger circuits. The explanation is that large circuits, with many node capacities, have an averaging effect on the capacity variations. E.g. the influence of one node capacity is almost not noticeable in the estimated circuit power.

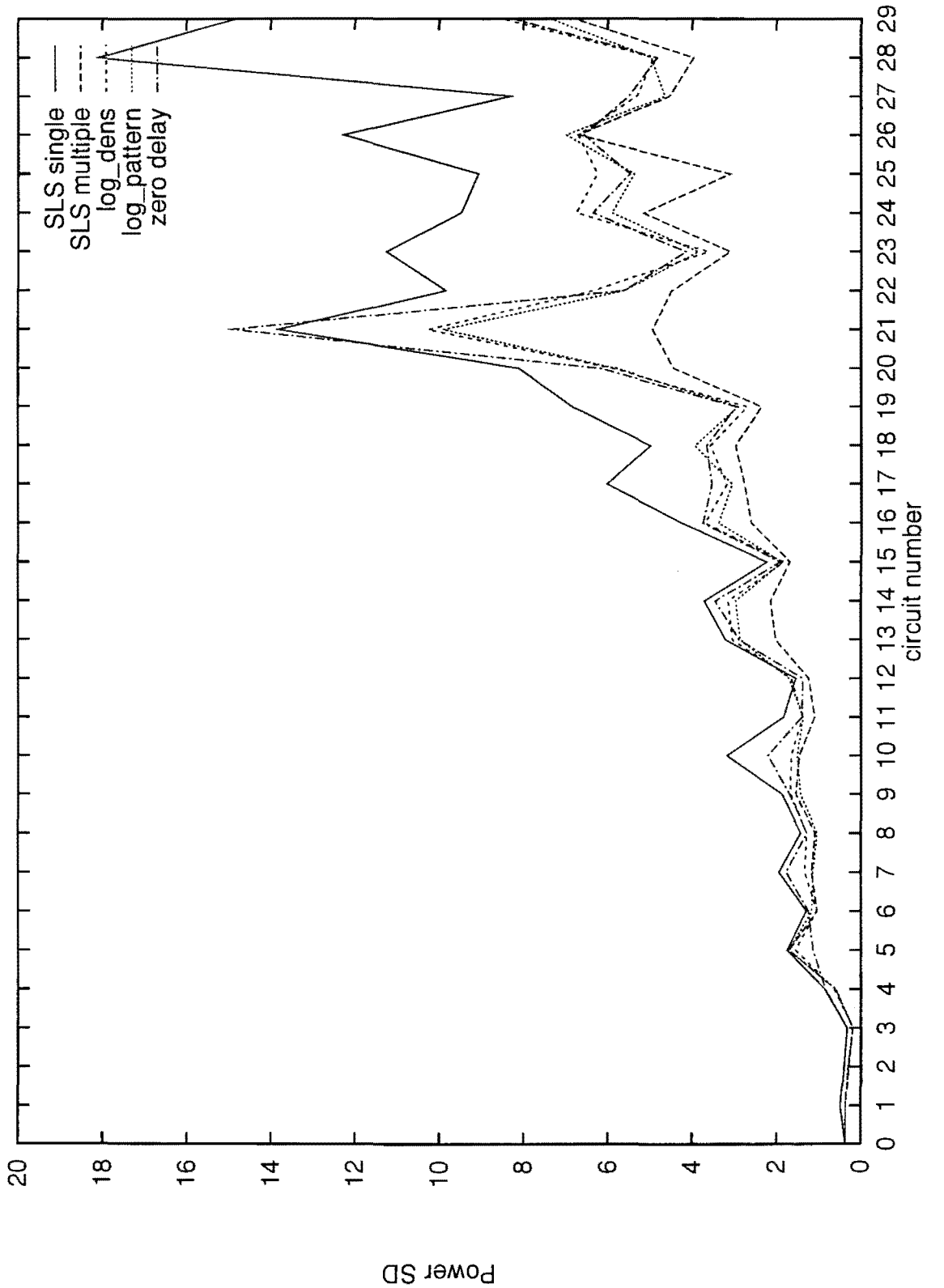

Figure 6.12: Power standard deviation

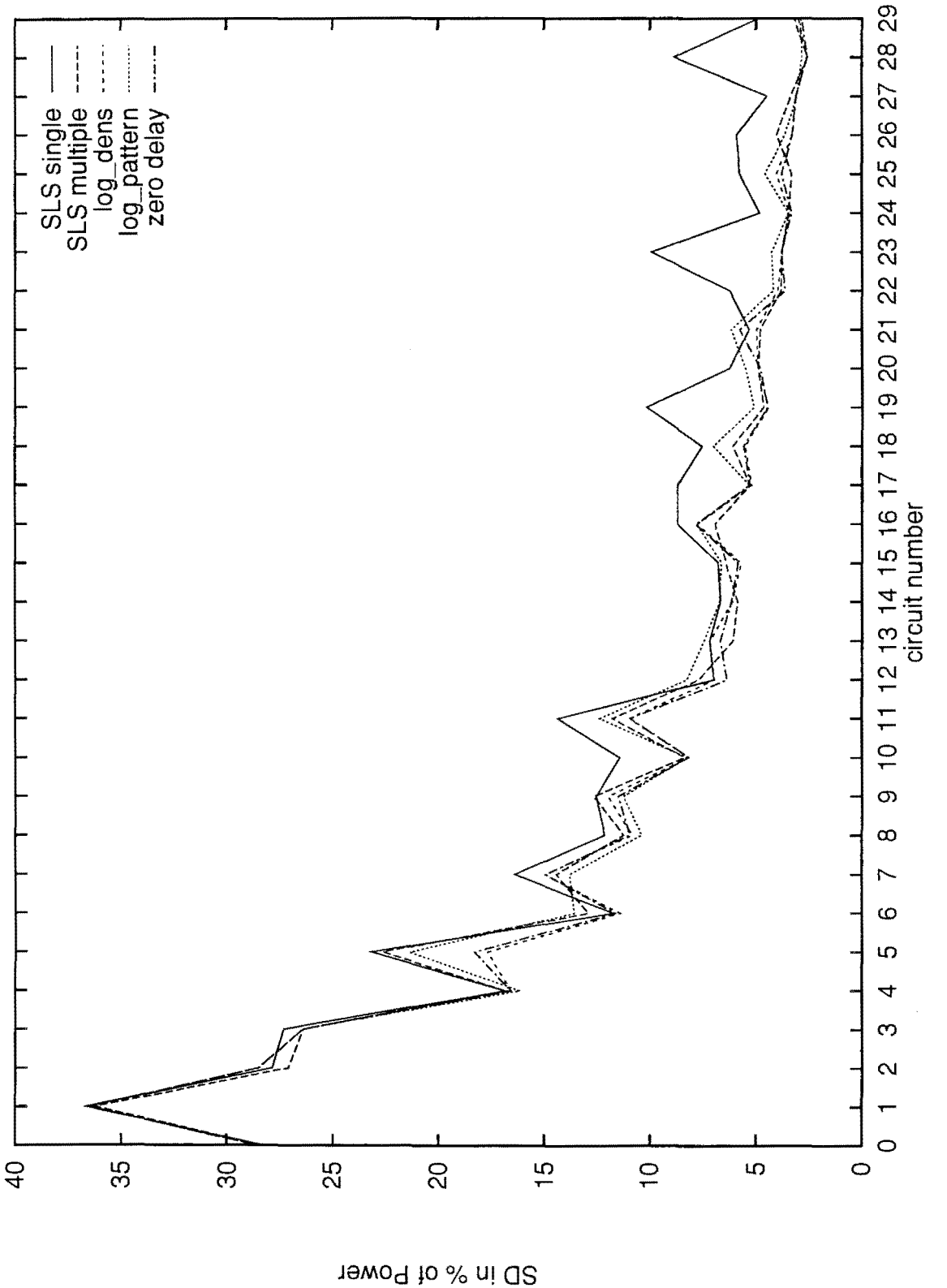

Figure 6.13: Normalized power standard deviation

6.2.4 Simulation Run-time

Figure 6.14 shows the simulation run times for all circuits performed on a HP9000-735 99 MHz workstation.

The first observation is that larger circuits use more calculation time. The second observation is that both SLS methods are much slower than the methods based on BDDs.

However, one exception exists: the xorglitch circuit (21). Log_pattern is very slow for this circuit because the outputs glitch very often. Therefore, the patterns for the output signals become very large. The inertial delay filter processes these large patterns consuming a considerable amount of time.

Another observation is that the SLS multiple method is slower than the SLS single simulation method. This can be explained by the fact that more output signal transitions occur for the same set size of input vectors in the multiple input simulation.

Figure 6.15 shows the simulation times for the BDD based estimation programs. The zero delay curve is almost on the x-axis, meaning it uses hardly any simulation time. The largest circuit (c8), consisting of 178 gates, is estimated in only 0.3 seconds. This is that short that this method can be used very well in an phase of circuit design.

The SIS program, performing exact the same function as the log_dens program, is considerably slower. This can be explained by the inefficient implementation of the symbolic simulation in SIS.

Log_dens is slightly faster in the power estimation than log_pattern as long as amount of glitches is not noticeable. The log_pattern program must be optimized for circuits experiencing many glitches such as the parity (21) and the mux (24) circuit.

Based on the simulation times log_pattern is the best choice for power estimation in combinational logic circuits. If the circuit has XOR functionality the log_dens method is preferred. For a quick power indication the zero delay program is very good, but information about glitching or internal spatial correlations is lost.

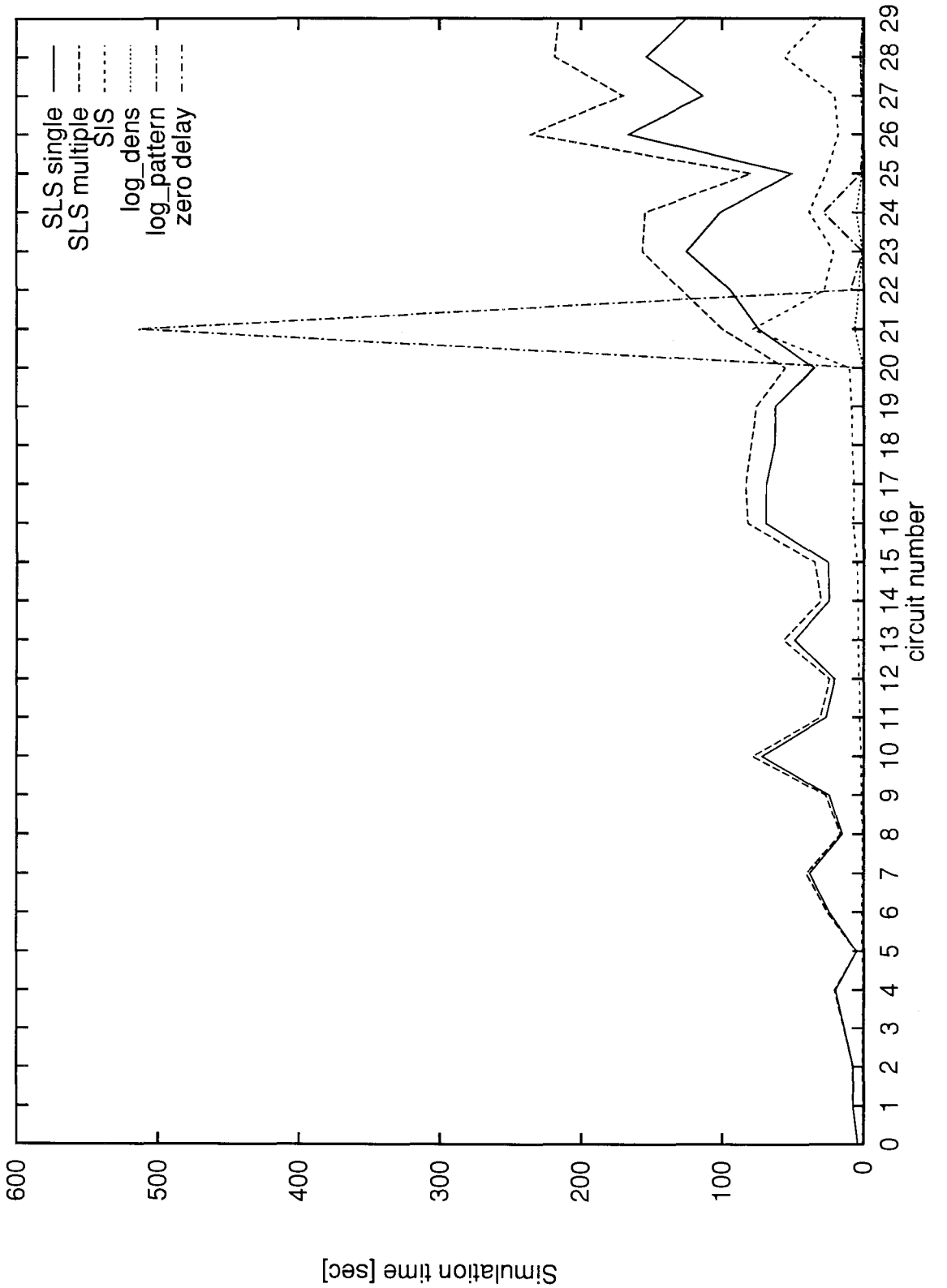

Figure 6.14: Simulation run time

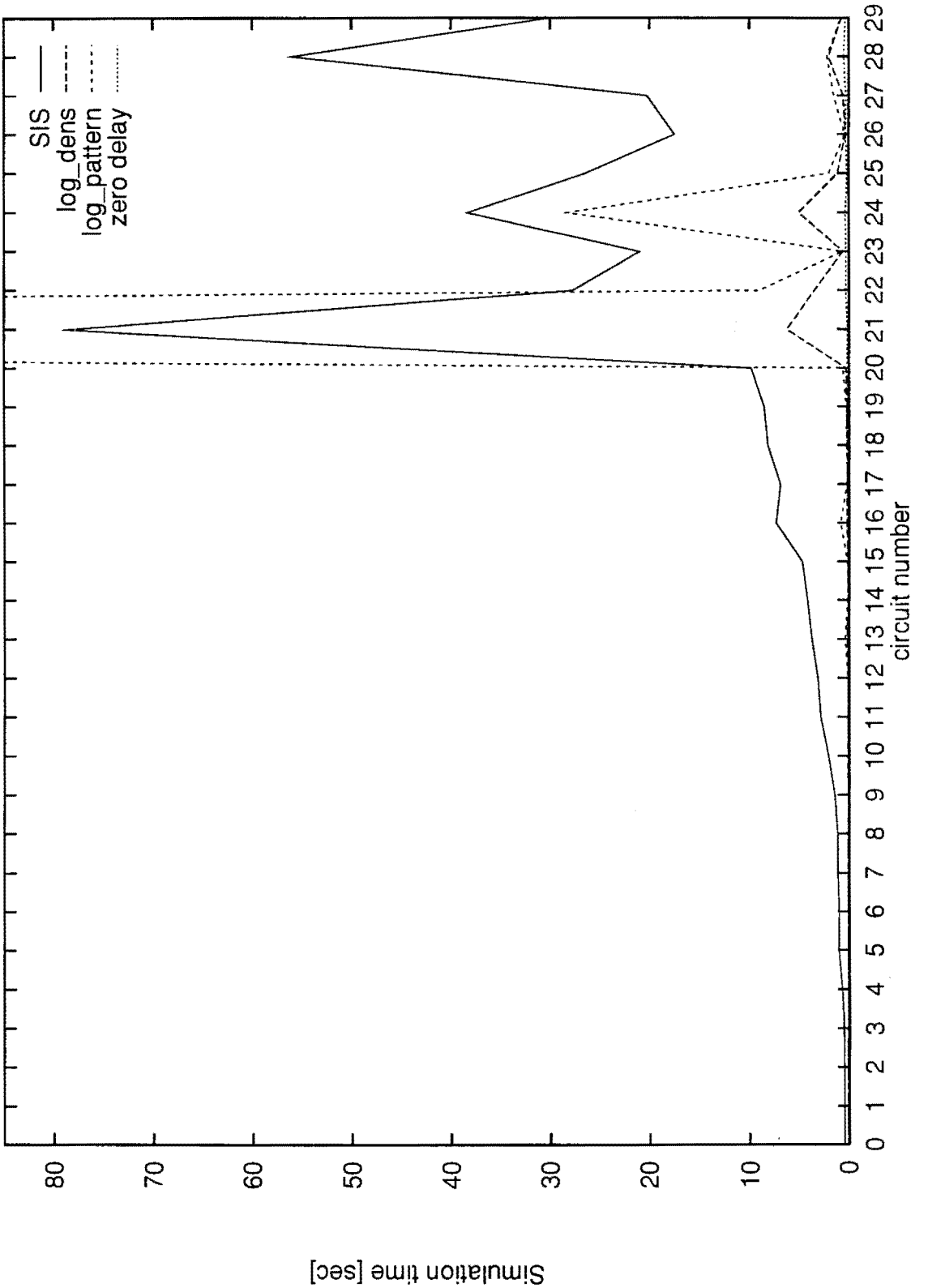

Figure 6.15: Simulation run time

Chapter 7

Conclusions

A switch-level-simulation is the most accurate method for determining the dissipated power in a circuit because it counts the transitions and calculates the corresponding power dissipation. Power dissipation can also be estimated using a probabilistic method based on a zero or general delay method.

The results obtained from the SLS method using multiple input switching are used for reference and can be seen as the optimal results which any estimation method should give.

The zero delay method, based on calculating the boolean difference, compares excellent to the SLS single input switching method. The results differ maximal a few percent. Power estimation using a zero delay model gives a consequent overestimation of power compared to the multiple input switching SLS method. However, this value is a good upper limit for power consumption. Because this is also the fastest estimation method it can be used for power estimation based on a worst-case approach.

Symbolic simulation, used in non zero delay models, is an accurate method for power estimation. It takes the re-convergent outputs into account and can even detect the power consumed by glitches.

All power estimation methods based on BDDs are very fast. The use of inertial delays, which give more accurate results, still results in a very fast estimation method. Log_pattern has a bad performance in run time compared to non-inertial delay methods for circuits which have very much glitching. But both the SLS methods are much slower. For large circuits the run time is about one second per gate using 2000 input vectors.

Randomizing circuit parameters has little effect on the total power consumption. Statistical influences on the estimated power are very small, even when the circuit capacities have variations over a large range.

Chapter 8

Future Work

For accurate power estimation, the exact net-capacitances must be known. The random factor now used has no relation to an actual circuit layout. A large improvement would be if the net-capacities are derived from a router program.

The SLS program, which delivers very accurate results, has one major disadvantage. It cannot detect the convergence of the total power while simulating the circuit. If this is possible the simulations will be more accurate because it stops when convergence is reached. In most cases this is faster than supplying a fixed number of input vectors.

Internal spatial correlations are automatically incorporated into the symbolic simulation, but spatial correlations on inputs are not taken into account. When these correlations are used, the overestimation will be hopefully lower.

The effect of temporal correlations of input signals is not investigated. Especially in sequential circuits this can have a rather large effect.

The log_dens and log_pattern methods are very fast, but the implementation is not optimal. An improvement in run time is expected after program optimization.

References

1. Najm, F.N.
TRANSISITION DENSITY, A STOCHASTIC MEASURE OF ACTIVITY IN DIGITAL CIRCUITS.
In: 28th ACM/IEEE Design Automation Conference, San Francisco, CA, USA, 17-21 June 1991.
New York: ACM, 1991. Vol. 25, p. 644-649.

2. Ghosh, A. and S. Devadas, K. Keutzer, J. White
ESTIMATION OF AVERAGE SWITCHING ACTIVITY IN COMBINATIONAL AND SEQUENTIAL CIRCUITS.
In: 29th ACM/IEEE Design Automation Conference, Ananheim, CA, USA, 8-12 June 1992.
Los Alamitos: IEEE Comput. Soc. Press, 1992. Vol. 26, p. 253-259.

3. Tan-li Chou and K. Roy, S. Prasad
ESTIMATION OF CIRCUIT ACTIVITY CONSIDERING SIGNAL CORRELATIONS AND SIMULTANEOUS SWITCHING.
In: 1994 IEEE/ACM International Conference on Cumputer-Aided Design, San Jose, CA, USA, 6-10 Nov. 1994.
New York: ACM, 1994. Vol. 30, p. 300-303.

4. Marculescu, R. and D. Marculescu, M. Pedram
SWITCHING ACTIVITY ANALYSIS CONSIDERING SPATIOTEMPORAL CORRELATIONS.
In: 1994 IEEE/ACM International Conference on Cumputer-Aided Design, San Jose, CA, USA, 6-10 Nov. 1994.
New York: ACM, 1994. Vol. 30, p. 294-299.

5. Berkelaar, M.R.C.M. and P.H.W. Buurman, J.A.G. Jess
COMPUTING THE ENTIRE ACTIVE AREA/ POWER CONSUMPTION VERSUS DELAY TRADE-OFF CURVE FOR GATE SIZING WHITH A PIECEWISE LINEAR SIMULATOR.
In: IEEE/ACM International Conference on Computer-Aided Design, San Jose, CA, USA, 6-10 Nov. 1994.
New York: ACM, 1994. Vol. 30, p. 474-480.

6. Ercolani, S. and M. Favalli, M. Damiani, P. Olivo, B. Ricco.
ESTIMATE OF SIGNAL PROBABILITY IN COMBINATIONAL LOGIC NETWORKS.
In: Proceedings of the 1st European Test Conference, Paris, 12-14 April 1989.
Vol. 14 (1989), p. 132-138.

7. Najm, F.N.
TRANSITION DENSITY: A NEW MEASURE OF ACTIVITY IN DIGITAL CIRCUITS.
IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,
Vol. 12 (1993), no. 2, p. 310-323.

8. Kapoor, B.
IMPROVING THE ACCURACY OF CIRCUIT ACTIVITY MEASUREMENT.
In: 31st Design Automation Conference, San Diego, CA, USA, 6-10 June 1994.
New York: ACM, 1994. Vol. 28, p. 734-739.

9. Xakellis, M.G. and F.N. Najm
STATISTICAL ESTIMATION OF THE SWITCHING ACTIVITY IN DIGITAL CIRCUITS.
In: 31st Design Automation Conference, San Diego, CA, USA, 6-10 June 1994.
New York: ACM, 1994. Vol. 28, p. 728-733.

10. Najm, F.N.
LOW-PASS FILTER FOR COMPUTING THE TRANSITION DENSITY IN DIGITAL CIRCUITS.
IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,
Vol. 13 (1994), no. 9, p. 1123-1131.

11. Roy, K. and S.C. Prasad.
CIRCUIT ACTIVITY BASED LOGIC SYNTHESIS FOR LOW POWER RELIABLE OPERATIONS.
IEEE Transactions on Very Large Scale Integration [VLSI] Systems,
Vol. 1 (1993), no. 4, p. 503-513.

12. Najm, F.N.
ESTIMATING POWER DISSIPATION IN VLSI CIRCUITS.
IEEE Circuits and Devices Magazine,
Vol. 10 (1994), no. 4, p. 11-19.

13. Najm, F.N.
FEEDBACK, CORRELATION, AND DELAY CONCERNS IN THE POWER ESTIMATION OF VLSI CIRCUITS.
In: 32nd Design Automation Conference, San Francisco, CA, USA, 12-16 June 1995.
New York: ACM, 1995. Vol. 29, p. 612-617.

14. Mehta, H. and M. Borah, R.M. Owens, M.J. Irwin
ACCURATE ESTIMATION OF COMBINATIONAL CIRCUIT ACTIVITY.
In: 32nd Design Automation Conference, San Francisco, CA, USA, 12-16 June 1995.
New York: ACM, 1995. Vol. 29, p. 618-622.

15. Najm, F.N. and M. Y. Zhang.
EXTREME DELAY SENSITIVITY AND THE WORST-CASE SWITCHING ACTIVITY IN VLSI CIRCUITS
In: 32nd Design Automation Conference, San Francisco, CA, USA, 12-16 June 1995.
New York: ACM, 1995. Vol. 29, p. 623-627.

16. Najm, F.N. and S. Goel, I.N. Hajj
POWER ESTIMATION IN SEQUENTIAL CIRCUITS.
In: 32nd Design Automation Conference, San Francisco, CA, USA, 12-16 June 1995.
New York: ACM, 1995. Vol. 29, p. 635-640.

17. Schneider, P. H. and U. Schlichtmann, K.J. Antreich
NEW POWER ESTIMATION TECHNIQUE WITH APPLICATION TO DECOMPOSITION OF BOOLEAN FUNCTIONS FOR LOW POWER.
In: Proceedings of the 1994 European Design Automation Conference,
Vol. 19 (1994), p. 388-393.

18. Monteiro, J. and S. Devadas, B. Lin
A METHODOLOGY FOR EFFICIENT ESTIMATION OF SWITCHING ACTIVITY IN SEQUENTIAL LOGIC CIRCUITS.
In: 31st Design Automation Conference, San Diego, CA, USA, 6-10 June 1994.
New York: ACM, 1994. Vol. 28, p. 12-17.

19. Najm, F. N.
IMPROVED ESTIMATION OF THE SWITCHING ACTIVITY FOR RELIABILITY PREDICTION IN VLSI CIRCUITS.
In: Proceedings of the Custom Integrated Circuits Conference 1994, Vol. 16 (1994), p. 429-432.

20. Tjarnstrom, R.
POWER DISSIPATION ESTIMATE BY SWITCH LEVEL SIMULATION.
In: IEEE International Symposium on Circuits and Systems, Portland, OR, USA, 8-11 May 1989. New York: IEEE, 1989. Vol. 40, p. 881-884.

21. Najm, F.N.
A SURVEY OF POWER ESTIMATION TECHNIQUES IN VLSI CIRCUITS.
IEEE Transactions on Very Large Scale Integration [VLSI], Vol. 2 (1994), no. 4, p. 446-455.

22. Chandrakasan, A.P. and S. Sheng, Brodersen, R.W.
LOW-POWER CMOS DIGITAL DESIGN.
IEEE Journal of Solid-State Circuits, Vol. 27 (1992), no. 4, p. 473-484.

23. Kang, S.M.
ACCURATE SIMULATION OF POWER DISSIPATION IN VLSI CIRCUITS.
In: Nineteenth Asilomar Conference on Circuits, Systems & Computers, Pacific Grove, CA, USA, 6-8 Nov. 1985.
New York: IEEE, 1986, p. 548-551.

24. Devadas, S. and K. Keutzer, J. White
ESTIMATION OF POWER DISSIPATION IN CMOS COMBINATIONAL CIRCUITS USING BOOLEAN FUNCTION MANIPULATION.
IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 11 (1992), no. 3, p. 373-383.

25. Cirit, M. A.
ESTIMATING DYNAMIC POWER CONSUMPTION OF CMOS CIRCUITS.
In: IEEE International Conference on Computer-Aided Design: ICCAD-87, Santa Clara, CA, USA, 9-12 Nov. 1987.
New York: IEEE, 1987, p. 534-537.

26. Hachtel, G.D. and E. Macii, A. Pardo, F. Somenzi
PROBABILISTIC ANALYSIS OF LARGE FINITE STATE MACHINES.
In: 31st Design Automation Conference, San Diego, CA, USA, 6-10 June 1994.
New York: ACM, 1994. Vol. 28, p. 270-275.

27. Burch, R. and F. Najm, P. Yang, D. Hocevar
PATTERN-INDEPENDENT CURRENT ESTIMATION FOR RELIABILITY ANALYSIS OF CMOS CIRCUITS.
In: Proceedings of the 25th Design Automation Conference, June 1988, Vol. 22 (1988), p. 294-299.

28. Najm, F.N. and R. Burch, P. Yang, I.N. Hajj
PROBABILISTIC SIMULATION FOR RELIABILITY ANALYSIS OF CMOS VLSI CIRCUITS.
IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 9 (1990), no. 4, p. 439-450.

29. Markowsky, G.
BOUNDING SIGNAL PROBABILITIES IN COMBINATIONAL CIRCUITS.
IEEE Transactions on Computers, Vol. C-36 (1987), no. 10, p. 1247-1251.

30. Seth, S.C. and L. Pan, V.D. Agrawal
PREDICT - PROBALISTIC ESTIMATION OF DIGITAL CIRCUIT TESTABILITY.
In: IEEE 15th Annual International Symposium on Fault-Tolerant Computing, Ann Arbor, June 1985.
New York: IEEE, 1989. Vol. 15, p. 220-225.

31. Parker, K.P. and E.J. McCluskey
PROBALISTIC TREATMENT OF GENERAL COMBINATIONAL NETWORKS.
IEEE Transactions on Computers,
Vol. C-34 (1975), no. 6, p. 668-670.

32. Brace, K.S. and R.L. Rudell, R.E. Bryant
EFFICIENT IMPLEMENTATION OF A BDD PACKAGE.
In: Proceedings of the 27th ACM/IEEE Design Automation Conference(DAC), Orlando, Florida, USA, 24-28 June 1990.
Los Alamitos, California, USA: IEEE Computer Society Press, 1990, p. 40-45.

Appendix A

Circuit Schematics

A.1 Special Test Circuits

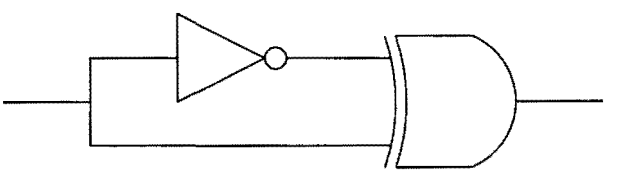

Figure A.1: xorglitch circuit

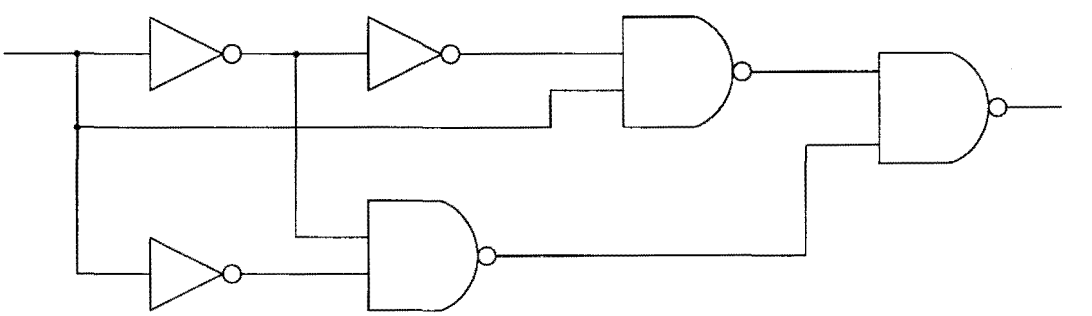

Figure A.2: Mapped xorglitch circuit

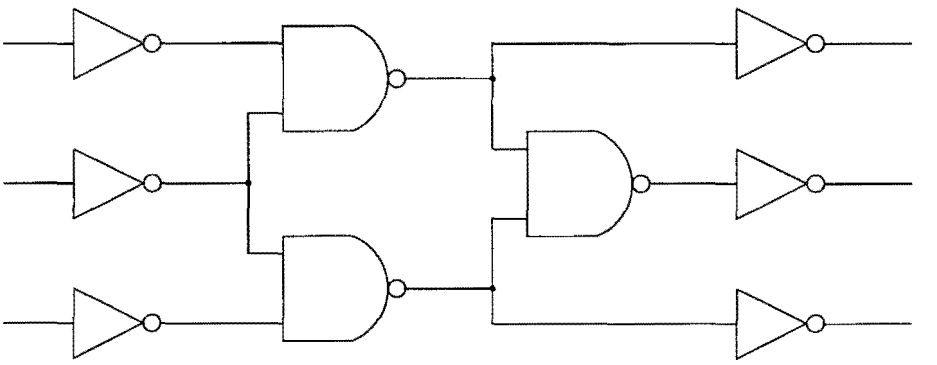

Figure A.3: overestimate

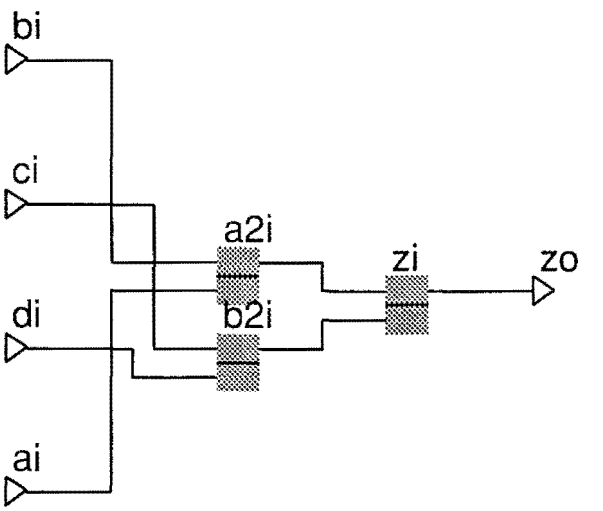

Figure A.4: nandtreesmall

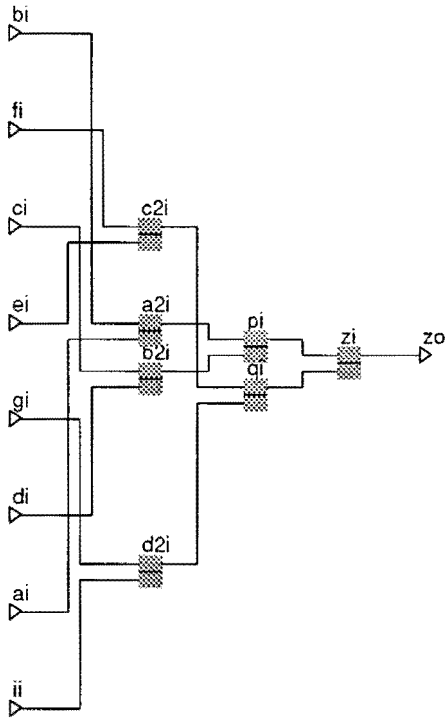

Figure A.5: nandtree

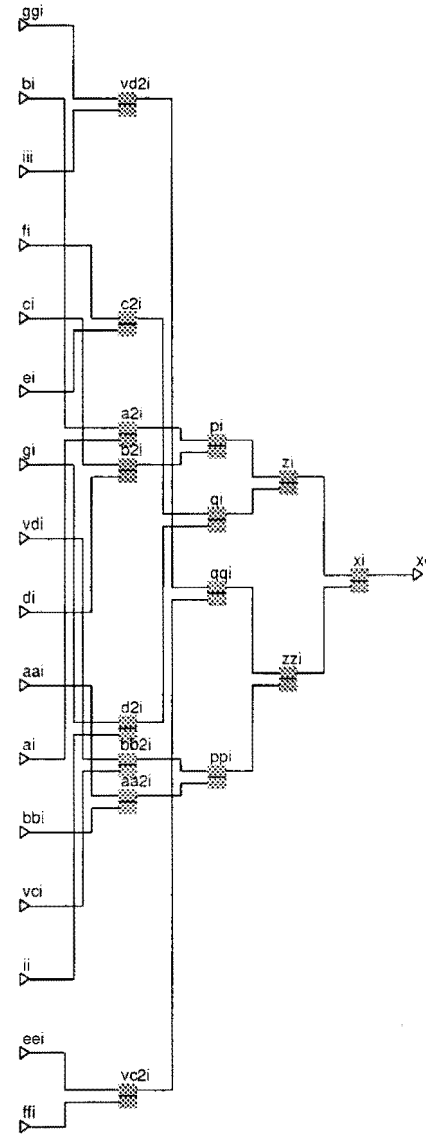

Figure A.6: nandtreebig

A.2 LGSynth91 Circuits

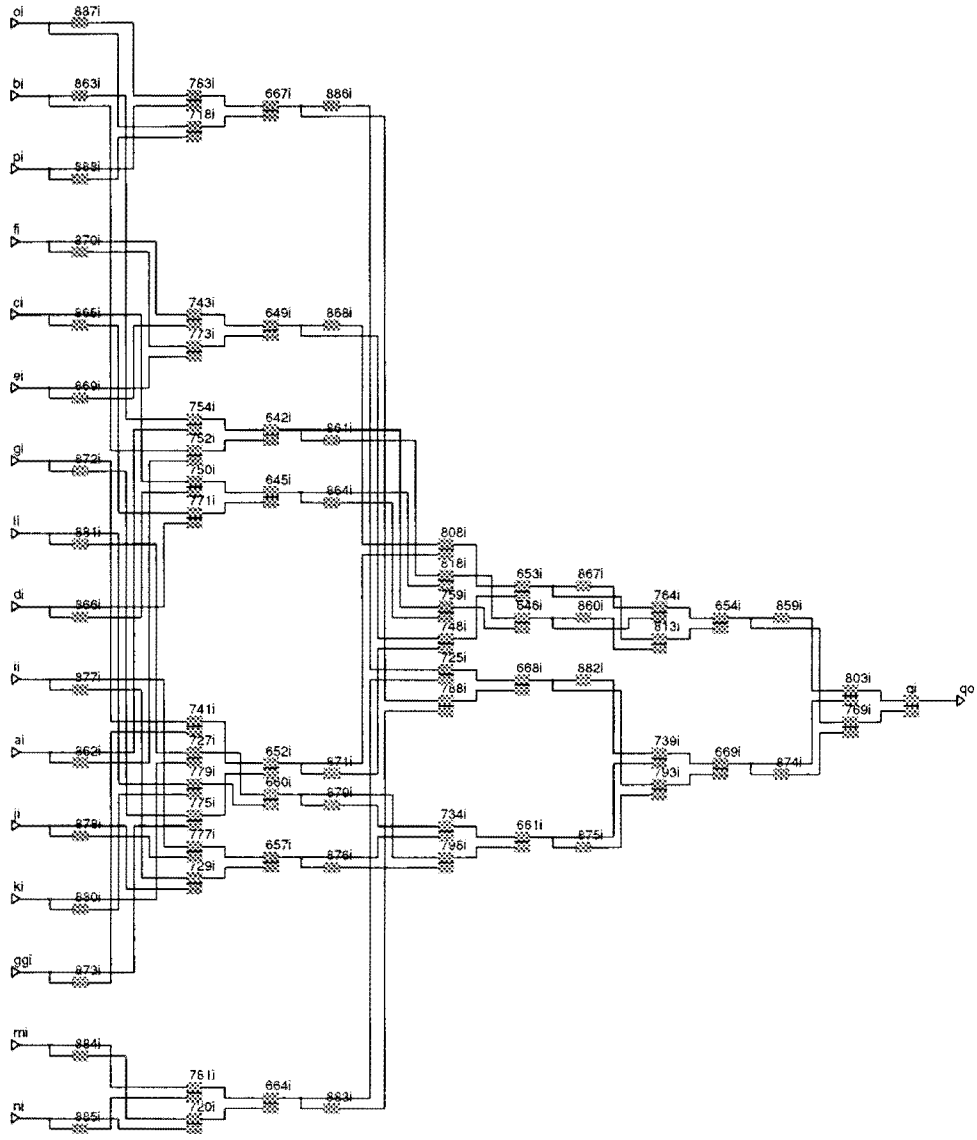

Figure A.7: parity