Optimizing CMOS Circuits for Low Power using Transistor Reordering *

E. Musoll and J. Cortadella
Dept. of Computer Architecture
Universitat Politècnica de Catalunya
08071 Barcelona, Spain

Abstract

This paper addresses the optimization of a circuit for low power using transistor reordering. The optimization algorithm relies on a stochastic model of a static CMOS gate that includes the power of internal nodes of the gate. This power-consumption model depends on the switching activity and the equilibrium probabilities of the inputs of the gate. The model allows an exploration of the different configurations of a gate that are obtained by reordering its transistors. Thus, the best configuration of each gate is selected and the overall power consumption of the circuit is reduced.

1 Introduction

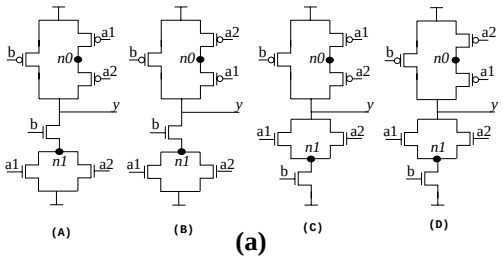
The continuous increasing packing density and clock frequency of static CMOS circuits has pushed low power as one of the principal design parameters, specially in battery-powered portable systems, such as note-pad computers, personal digital assistants, multi-media terminals and mobile telephones.

This paper addresses the optimization of a circuit for low power using transistor reordering from a gate-level description. The optimization algorithm uses a power-consumption model of a static CMOS gate that takes into account the power of the internal nodes of the gate. This model allows a fast exploration of the different configurations of a gate that are obtained by reordering its transistors. Thus, the best configuration of each gate is selected and the overall power consumption of the circuit is decreased.

We focus on combinational multilevel circuits, where it has been shown that the power consumption of useless signal transitions (i.e. those transitions that do not contribute to the final result of the circuit) accounts for a large fraction of the overall dynamic power consumption of the circuit. Thus, it is necessary to incorporate the switching activity of the input signals into the power-consumption of the gate.

1.1 Motivation examples

To illustrate why it is important to incorporate the switching activity information to the power estimation of a gate, consider the four possible configurations of the gate in Figure 1(a) that implement function $y = \overline{(a1 + a2)\, b}$. Different switching activity of the inputs (D_{a1}, D_{a2} and D_b) results in a different optimal transistor reordering of the gate as it is shown in Table 1(b). The equilibrium probability (i.e. the probability for a signal to be '1') of all input signals has been set to 0.5. Table 1(b) shows the power consumption for two different input switching activity scenarios (cases **(1)** and **(2)**) of the different configurations relative to configuration **(D)** in case **(1)**. Time intervals between two consecutive transitions of input signal k to the gate follow an exponential distribution with average $1/D_k$. In case **(1)** the best transistor ordering is given by configuration **(A)** of Figure 1(a); the power consumption is decreased by 19% with respect to configuration **(D)**. In case **(2)**, the power is decreased by 17% if configuration **(D)** is taken instead of **(A)**.



(a)

	Activity (trans./sec)	(A)	(B)	(C)	(D)	Red.
(1)	D_{a1} = 10K D_{a2} = 100K D_b = 1M	0.81	0.84	0.98	1.0	19%
(2)	D_{a1} = 1M D_{a2} = 100K D_b = 10K	0.58	0.53	0.53	0.48	17%

(b)

Figure 1: **(a)** Four implementations of function $y = \overline{(a1 + a2)\, b}$ and **(b)** relative power consumption for two different input activity scenarios.

The ripple-carry adder is another example in which the equilibrium probability does not give enough information to optimize gates for low power. Consider an adder implemented as a chain of full-adders that has to calculate the addition of two n-bit operands with equal equilibrium probability for all bits. The equilibrium probabilities of all inputs of the full-adders is 0.5, but it is clear that the switching activity of the inputs of the full-adders corresponding to the operands is low (0.5 transitions per operation) whereas the switching activity of the input corresponding to the propagated carry is higher (specially in those full-adders that compute the most-significant bits) because of the generation and propagation of useless signal transitions.

2 Previous work and overview

Carlson [2] hinted the possibility to use the transistor-reordering technique to decrease power consumption and he presented an algorithm for delay/power/area optimization where high speed was synonym of high power consumption. This approach of measuring power consumption is not sufficiently accurate since it does not consider the probability and switching activity of signals. No power consumption reductions are reported in [2].

Input reordering conform a subset of transistor reordering techniques. For example, by reordering the inputs in a 3-input NAND gate, 6 different configurations of the gate are

*This work has been supported by CICYT TIC95-0419 and Dept. d'Ensenyament de la Generalitat de Catalunya.

ED&TC '96
0-89791-821/96 $5.00 © 1996 IEEE

obtained; the same occurs if we apply transistor reordering. But in the gate represented in Figure 1(a), only two different configurations are obtained by applying input reordering and four are obtained with transistor reordering. Input reordering has been used in [10, 1] along with transistor sizing to reduce power consumption. It is not clear in those works which is the contribution of the input reordering technique by itself. This is true specially in [1], where the largest power consumption reduction (15%) is achieved in the benchmark with the largest number of high-fanout gates. Input (or transistor) reordering techniques obtain better results when applied to low-fanout gates because these techniques focus on reducing the power consumption of the internal nodes of the gate. If the output capacitance is increased, the overall gate power reduction obtained is decreased [4]. Our work accounts for the transistor-reordering technique by itself. We maintain constant the size of the transistors when they are reordered in each gate.

In [4], input reordering techniques are applied to a large variety of CMOS NAND gates ranging from 2 to 8 inputs. The major contribution in [4] is the description of a new model for the power consumption of a MOSFET chain. This model accounts for the consumption of internal nodes and it depends only on the equilibrium probabilities of the inputs of the gate.

The closest works to ours are [8, 9]. In [8], a clear example of the effect of transistor reordering on the power consumption of a circuit is shown. An estimated average of 6% in power reduction is obtained for some MCNC benchmarks. Finally, in [9] the authors propose a set of simple transistor reordering rules for both basic and complex CMOS gates to minimize the switching activity at the internal nodes. Average reductions of 9% are reported for several circuits.

2.1 Overview of our approach

10K trans./sec 100K trans./sec 1M trans./sec	**a** **b** **c**	**b** **a** **c**	**a** **c** **b**	**c** **a** **b**	**b** **c** **a**	**c** **b** **a**
Power	0.91	0.91	0.95	0.96	0.99	1.0

Table 1: Relative power consumption in a 3-input NAND gate for different input activities.

We present a power-consumption model of a static CMOS gate that depends on both the static probabilities and the switching activity of its inputs. This model is based on the *transition density* measure of activity in digital circuits proposed by Najm [6]. Our power-consumption model differs from the one presented in [4] in that we take into account the switching information of the input signals. As an example, consider a 3-input NAND gate. Assume equilibrium probabilities of 0.4, 0.5 and 0.6 for its inputs **a**, **b** and **c** respectively. Table 1 shows the relative power consumption of the gate for different input switching activity scenarios. Our model discerns, among the six possible input reorderings, those that give the maximum and minimum power consumption. The model in [4], on the contrary, can not discern any of the reorderings and it provides the same power estimation for all of them.

An algorithm that traverses the gate-level description of the circuit and uses the gate model mentioned above is presented and results are reported for a wide range of MCNC benchmarks. A cell library with different instances of a single gate to obtain all its possible transistor reorderings has been implemented in a Sea-of-Gates design style. The results are based on switch-level simulations and show that power-consumption reductions of up to 20% may be obtained when applying the transistor reordering technique.

The paper is organized as follows: in Section 3, a power-consumption model of a static CMOS gate that takes into account the power consumption of the internal nodes is presented. In Section 4, an algorithm that traverses the gate-level description of the circuit and generates an optimized circuit for low power is described. Results are presented for several MCNC benchmarks in Section 5. In Section 6, some conclusions are drawn.

3 Power consumption of CMOS gates

In CMOS circuits there are three sources of power consumption: switching activity, direct-path short-circuit current and leakage current. In static CMOS, the switching activity source dominates the total power consumption because of the charging of capacitors. The average power consumption of a CMOS gate can be modeled with the equation $P_{avg} = \frac{1}{2}\frac{C_{load}\ V_{dd}^2\ D}{T_{cyc}}$, where C_{load} is the load capacitance, V_{dd} is the power supply, T_{cyc} is the global clock period and D is the number of signal transitions at the output of the gate per clock cycle.

3.1 Definitions and overview

Definition 3.1 (Stochastic process) *Let $x(t)$ be the value of a system characteristic being observed at time t. In most situations, $x(t)$ is not known before time t and may be viewed as a random variable. A stochastic process is a description of the relation between the random variables $x(t)$.*

Definition 3.2 (Stationary Markov process) *A stationary Markov process is a stochastic process where the probability law relating the next period's state to the current state does not change (or remains stationary) over time.*

Definition 3.3 (Equilibrium probability) *Let $x(t), t \in (-\infty, +\infty)$, be a 0-1 stationary Markov process with random transition times. The probability that it takes the value 1 at any given time t is the expected value $E[x(t)]$ at that time and it is independent of time. This value is called the equilibrium probability of $x(t)$ and is denoted as $P(x)$.*

Definition 3.4 (Switching activity) *The switching activity of a 0-1 stochastic process $x(t)$ is the number of 0-to-1 transitions and 1-to-0 transitions of $x(t)$ in a time unit.*

Henceforth, we will model logic signals of a circuit as 0-1 stationary Markov processes as in [6] to derive a power-consumption model of a static CMOS gate that includes the switching activity at the output and internal nodes of the gate. The model depends on both the equilibrium probabilities and the switching activity of the inputs of the gate.

The switching activity in both input and output nodes of a gate is measured with the *transition density* technique described in [6]. This technique is briefly reviewed in the next section.

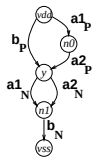
3.2 Transition density

The *transition density* is a compact measure of the switching activity in digital circuits. The transition density of a node is the average number of signal transitions per time unit of that node and it is defined as $D(y_j) = \sum_{i=0}^{n-1} P(\frac{\partial y_j}{\partial x_i}) D(x_i)$, where $P(z)$ is the equilibrium probability, $D(z)$ is the transition density, x (y) are the n (m) gate inputs (outputs). $\frac{\partial y_j}{\partial x_i}$ is named the *boolean difference* and it is a boolean function that may depend on all $x_p,\ p = 1 \ldots n,\ p \neq i$. The boolean difference $\frac{\partial y}{\partial x}$ is defined as $y\,|_{x=1} \oplus y\,|_{x=0} = y(x) \oplus y(\overline{x})$. If $\frac{\partial y_j}{\partial x_i} = 1$, then all the transitions at input x_i are propagated to output y_j.

Thus, the transition density provides a fast way of propagating switching activity from the primary inputs to the outputs of the gates that compose a circuit.

3.3 Extended power-consumption model

3.3.1 Notation




```
CALCULATE_H_FUNCTION (n_k, function)
    dfs_list_k = DEPTH_FIRST_SEARCH (n_k, vdd)
    foreach element (dfs_list_k, e_{k_i})
        ADD_INPUT_TO_MINTERM (e_{k_i}, function)
        if source(e_{k_i}) = vdd then
            CREATE_NEW_MINTERM (function)
```

(a) **(b)**

Figure 2: **(a)** Static CMOS gate representation and **(b)** algorithm to obtain function H_{n_k}.

We represent a static CMOS gate as a directed acyclic graph (V, E). $V = \{n_0 \ldots n_{p-1}, y, vdd, vss\}$ is the set of nodes representing the p internal nodes of the gate $(n_0 \ldots n_{p-1})$, the output node (y) and the power and ground nodes (vdd, vss). The set of edges representing the $2q$ transistors (q of type P and q of type N) that connect nodes in V is $E = \{e_{0_P} \ldots e_{q-1_P}, e_{0_N} \ldots e_{q-1_N}\}$. Figure 2(a) shows the graph representation of gate **(C)** in Figure 1(a). Note that this representation retains the transistor order information of the gate.

The power consumption of a node (output or internal) of a gate is potentially affected by all the inputs of the gate. In particular, the power consumption of node n_k produced by input x_i ($W_{n_k}|_{x_i}$) is $\frac{\frac{1}{2} V_{dd}^2 C_{n_k} T_{x_i \rightarrow n_k}}{T_{cyc}}$, where $T_{x_i \rightarrow n_k}$ is the number of transitions that node n_k undergoes because of input x_i. In other words, $T_{x_i \rightarrow n_k}$ expresses how many signal transitions of x_i (D_{x_i}) are propagated to node n_k. C_{n_k} is the capacitance of node n_k[1].

Henceforth, we assume that a node is charged (discharged) only when there is a direct path from power (ground) supply to the node, i.e. we do not consider charge sharing among the nodes of the gate.

3.3.2 Computation of the model

To compute $T_{x_i \rightarrow n_k}$, the boolean function that represents all possible paths from power supply to node n_k needs to be calculated. Let us call H_{n_k} this function; similarly, G_{n_k} is the boolean function that represents all possible paths from n_k to ground[2].

The algorithm to obtain the H_{n_k} function is depicted in Figure 2(b). Function H_{n_k} is obtained by generating a minterm for each possible path from node n_k to supply node vdd. A path from node n_k to vdd is a set of r edges e_{k_i} so that $dst(e_{k_0}) = n_k, dst(e_{k_1}) = src(e_{k_0}), \ldots, dst(e_{k_{r-1}}) = src(e_{k_{r-2}})$ and $src(e_{k_{r-1}}) = vdd$, where $src(e_k)$ ($dst(e_k)$) is the source (destination) node of edge e_k. Using a depth-first-search approach [3], a list of all edges to visit is created ($depth\_first\_search\_list_k$). Afterwards, the edges of this list are added to the current minterm of the H_{n_k} boolean function (ADD_INPUT_TO_MINTERM()) until an edge e_{k_j} is reached so that $src(e_{k_j}) = vdd$. In this case, a new minterm is created (CREATE_NEW_MINTERM()), sharing with the last created minterm all its edges but the last one visited.

[1] Because of the task of modeling the capacitances of the nodes of a gate is difficult, these capacitances should be extracted and stored for all gates of the library whenever it is possible. This is the approach followed in this paper.

[2] Note that G_{n_k} and H_{n_k} are complementary functions only when n_k is the output node (y) of the gate.

In the example of Figure 2(a), the four minterms generated when calculating H_{n_1} are $\{a1\,\overline{b}, a1\,\overline{a2}, \overline{a1}, a2\,\overline{b}, a2\,\overline{a2}\,a1\}$, leading to $H_{n_1} = \overline{b}\,(a1 + a2)$. Similarly, G_{n_k} can also be derived. In Figure2(a), $G_{n_1} = b$. The time complexity of these algorithms is linear in the number of transistors of the gate.

Afterwards, the boolean difference of function H_{n_k} with respect to input x_i (that is $\frac{\partial H_{n_k}}{\partial x_i}$) and the equilibrium probabilities of node n_k need to be calculated. The boolean function $\frac{\partial H_{n_k}}{\partial x_i}$ is calculated as explained in Section 3.2. The equilibrium probability of node n_k is obtained as follows [4]: the probability of node n_k of being '1' at a given instant of time ($P(n_k)|_c$) is the probability that n_k was '1' in the instant before ($P(n_k)|_b$) and it is not discharged ($\overline{P(\frac{\partial G_{n_k}}{\partial x_i})}$) or that it was '0' ($\overline{P(n_k)|_b}$) and it is charged ($P(\frac{\partial H_{n_k}}{\partial x_i})$), i.e.

$$P(n_k)|_c = P(n_k)|_b \; \overline{P(\frac{\partial G_{n_k}}{\partial x_i})} + \overline{P(n_k)|_b} \; P(\frac{\partial H_{n_k}}{\partial x_i}) \;,$$

where $\overline{P(f)} = 1 - P(f)$.

Since all signals are assumed to be 0-1 stationary Markov processes, the steady state value of $P(n_k)$ can be derived as

$$P(n_k) = \frac{P(\frac{\partial H_{n_k}}{\partial x_i})}{P(\frac{\partial H_{n_k}}{\partial x_i}) + P(\frac{\partial G_{n_k}}{\partial x_i})} \;.$$

We conclude that $W_{n_k}|_{x_i}$ is

$$\frac{\frac{1}{2} V_{dd}^2 C_{n_k} D(x_i) \left(P(\frac{\partial H_{n_k}}{\partial x_i}) \overline{P(n_k)} + P(\frac{\partial G_{n_k}}{\partial x_i}) P(n_k)\right)}{T_{cyc}} \;.$$

If the contributions of all nodes (output and internal) are taken into account, the power estimation of the gate is obtained as $P_{gate} = \sum_{k=0}^{p-1} (\sum_{i=0}^{n-1} W_{n_k}|_{x_i}) + \sum_{i=0}^{n-1} W_y|_{x_i}$, where p is the number of internal nodes and n is the number of inputs of the gate.

4 Power-optimization algorithm

```
OBTAIN_PROBABILITIES (circuit)
gate_list = DEPTH_FIRST_TRAVERSE (circuit)
for each gate gate in gate_list do
    info_inputs = OBTAIN_PROB_AND_DENS (gate, circuit)
    FIND_BEST_REORDERING (info_inputs, gate, circuit)
    info_output = CALCULATE_DENS (info_inputs, gate)
    UPDATE_CIRCUIT_INFORMATION (info_output, circuit)
```

Figure 3: Optimization algorithm.

In this section, an algorithm that traverses the gate description of the circuit is presented. For each gate, it finds the best transistor reordering using the power-consumption model explained in Section 3.

4.1 Algorithm overview

Finding the best transistor reordering implies an exhaustive exploration of each gate. Since most gates only have a small number of transistors in series, an exhaustive exploration is feasible. The algorithm to obtain all possible transistor reorderings of a gate will be addressed later.

A simplified algorithm of the optimization process for low power is shown in Figure 3. The probabilities for all

output nodes of the gates of the circuit are computed in OBTAIN_PROBABILITIES() following the algorithm proposed in [7]. DEPTH_FIRST_TRAVERSE() returns the list of gates (*gate_list*) of the circuit (*circuit*) ordered in a depth-first fashion [3] from the outputs, i.e. every gate appears somewhere after all of its transitive fan-in gates. For each gate (*gate*) of this list, the probability and transition density information for all of its inputs is obtained from the circuit (OBTAIN_PROB_AND_DENS()). Afterwards, the best reordering is derived for gate *gate* (FIND_BEST_REORDERING()). Finally, the transition density of the output node of the gate is calculated (CALCULATE_DENS()) and this information is transferred to the circuit (UPDATE_CIRCUIT_INFORMATION()).

4.2 Monotonic characteristic

The algorithm takes advantage of the following property of the model explained in Section 3: *the reduction of the power in an individual gate always decreases the power of the circuit*. The reason of this monotonic behavior is that all possible transistor reorderings of a gate lead to the same probability and transition density at its output node if the model explained in Section 3 is used to compute them. Since (a) the model precisely relies on the probability and transition density of the inputs of a gate to decrease its power consumption and (b) the power of the circuit is the sum of the power of its gates, it is clear that the reduction of the power in an individual gate always decreases the total power of the circuit. This monotonic behavior may not correspond to the actual behavior of a circuit, but the experiments have shown that this local (greedy) approach results in an overall power reduction for the whole circuit.

Thus, with only one traversal of the circuit, the optimal reordering (always with respect to the model) for all gates is obtained.

4.3 Exhaustive exploration

Gate	#C	Gate	#C	Gate	#C
inv	1	aoi21[A,B]	4	oai21[A,B]	4
nand2	2	aoi22	8	oai22	8
nand3	6	aoi31[A,B]	12	oai31[A,B]	12
nand4	18	aoi32[A,B]	24	oai32[A,B]	24
nor2	2	aoi33	72	oai33	72
nor3	6	aoi211[A,B,C]	12	oai211[A,B,C]	12
nor4	18	aoi221[A,B,C]	24	oai221[A,B,C]	24
		aoi222	48	oai222	48

Table 2: Number of different configurations for some standard gates obtained by reordering its transistors.

Table 2 shows the number of possible configurations (#C) obtained by reordering the transistors of some standard gates. These configurations are obtained by *pivoting* on the internal nodes of the gate. More formally, this process of obtaining a new configuration is described as follows: let BS be the bottom subgraph (of the graph representing the PMOS or NMOS blocks of a gate) composed of all edges whose source is node n_k and whose destination node is either node n_{bs} or another node n_{dummy}. The same definition is recursively applied to node n_{dummy} until all destination nodes are n_{bs}. Similarly, let TS be the top subgraph with its associated node n_{ts}. A new configuration of the gate is obtained by interchanging subgraphs BS and TS.

For example, in all configurations of Figure 1(a), new configurations are obtained by pivoting on nodes n_1, being n_{ts} and n_{bs}, in all cases, y and vss respectively.

Assuming the graph representation of the gate explained in Section 3.3, an algorithm that finds all possible transistor reorderings of a gate is presented in Figure 4. The strategy of pivoting on an internal node to obtain a new reordering of transistors leads to the generation of repeated reorderings. A dynamic programming approach with memoization [3] is used to avoid the generation of overlapping subproblems.

```
PIVOTE_AND_SEARCH (gate_graph, visited_reords, current_node)
    gate_graph = PIVOTING_ON_NODE (gate_graph, current_node)
    if not VISITED (gate_graph, visited_reorderings) then
        visited_reords = ADD_TO_VISITED_REORDS (gate_graph)
        for index = 1 to number_of_internal_nodes do
            if index ≠ current_node then
                PIVOTE_AND_SEARCH (gate_graph, visited_reords, index)

FIND_ALL_REORDERINGS (gate_graph)
    visited_reords ← ∅
    for index = 1 to number_of_internal_nodes do
        PIVOTE_AND_SEARCH (gate_graph, visited_reords, index)
```

Figure 4: Exhaustive exploration algorithm.

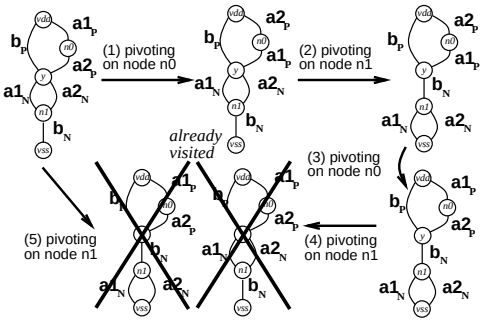


Figure 5: Execution example.

The algorithm recursively points to an internal node ($current\_node$) and *pivots* on it to obtain a new reordering (PIVOTING_ON_INTERNAL_NODE()). Further searching for new reorderings is pruned if the reordering obtained has already been visited (VISITED()). If it has not been visited, it is added to the set of transistor reorderings of the gate already visited (ADD_TO_VISITED_REORDERINGS()) and the algorithm is called again for all internal nodes of the gate except the current one (this is so to prevent the generation of a reordering that we know beforehand that we have already visited). In [5] it is demonstrated that all possible transistor reorderings of a gate are generated with the algorithm in Figure 4.

To illustrate how this algorithm works, it has been applied to the gate implementing the function $y = \overline{(a1 + a2)\, b}$. Figure 5 shows the execution. The starting graph representation of the gate is the one in Figure 2(a). We observe that all four possible reorderings (those already seen in Figure 1(a)) are generated.

The algorithm in Figure 4 works for gates that can be represented with a series-parallel graph. The gates of typical libraries can be all represented with this type of graphs.

5 Results

5.1 Scenarios for the experiments

A wide range of MCNC circuits have been used as benchmarks. They have been mapped into the gate library shown in Table 2. In some cases, to obtain all transistor reorderings of a gate, it is necessary to have more instances of that gate. For example, there are two instances of gate **oai21**: **oai21[A]**, which is able to implement configurations **(A)** and **(B)** of Figure 1(a) and **oai21[B]**, which is able to implement configurations **(C)** and **(D)**. All instances of the gates in Table 2 have been implemented in a Sea-of-Gates design style.

Two scenarios have been considered to evaluate the power-consumption savings obtained with the transistor reordering technique (see Figure 6(a)). In **Scenario A**, the circuit is considered to be embedded in a larger digital system. Thus, the equilibrium probability and the transition density of the inputs of the circuit may take very different values. In this scenario, the probabilities and transition density of the primary inputs of each circuit are randomly set

with a uniform distribution. Probabilities range from 0 to 1 and transition densities range from 0 to 1 million transitions per second. In **Scenario B**, the circuit is considered to be the whole digital system, with latches at its inputs and working at a fixed frequency. In this scenario, the probability and the transition density of the primary inputs of the circuit are set to, respectively, 0.5 and 0.5 transitions per cycle. In both scenarios, the optimization algorithm has been applied to the original gate-level description of the circuits to obtain, for each gate, the best instance and, for each instance, the best input reordering. Because of all instances of the same gate have the same area, the total area of the optimized circuit remains the same.

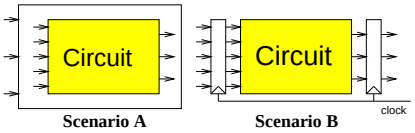


Figure 6: The two scenarios considered.

Circuit name	G	Scenario A			Scenario B		
		M	S	D	M	S	D
alu2	224	9.5	14.6	1.2	5.7	6.4	-0.2
c432	148	6.2	14.1	-0.7	4.3	8.2	-3.7
c499	316	2.8	4.6	-1.6	1.7	3.9	-3.0
c8	99	11.5	13.7	-15.5	4.1	5.1	-2.9
cht	117	9.9	9.3	4.4	3.7	6.9	2.5
cm150a	43	7.8	11.9	1.1	2.7	5.5	10.8
cm85a	24	15.4	20.0	0.0	8.7	15.3	-1.2
comp	94	5.8	10.2	17.5	2.2	3.7	7.4
cordic	64	6.5	11.9	-2.7	1.8	1.6	-0.7
i5	244	10.2	12.5	9.4	3.8	7.4	14.4
mux	55	9.7	12.5	13.3	4.1	5.5	-2.6
my_adder	128	4.3	12.3	6.2	1.7	0.7	5.4
parity	45	2.8	5.9	1.1	0.6	0.1	0.0
too_large	459	10.5	11.0	-8.0	4.6	3.2	-2.6
x1	192	11.2	12.4	11.7	5.3	4.9	10.2
x4	313	11.0	13.3	-2.4	5.1	6.4	-1.9
pcle	47	8.4	13.6	11.8	6.4	10.0	15.3
pcler8	64	7.7	16.8	13.8	6.4	11.5	14.1
frg1	67	12.2	15.2	8.4	4.9	8.3	4.4
sct	62	13.0	13.8	1.5	5.5	3.6	8.1
unreg	49	8.4	3.8	0.0	0.8	0.1	-14.0
z4ml	41	10.7	15.4	-5.3	3.0	1.7	-4.9
f51ml	73	13.8	15.0	7.5	4.5	5.4	-4.3
symml	84	12.7	12.6	3.7	3.0	3.3	-4.9
apex7	155	7.9	11.5	11.7	4.7	8.3	5.9
count	80	11.8	18.2	4.0	6.1	9.6	5.3
c1355	540	2.1	2.9	2.3	1.9	4.6	2.2
c1908	401	5.4	9.0	1.0	3.5	5.0	-2.0
c880	235	8.6	13.6	-0.4	4.5	10.2	-6.4
alu4	424	7.9	12.0	3.6	5.7	7.7	4.3
apex6	442	7.5	12.3	0.2	4.2	7.1	-3.4
example2	222	6.2	8.4	17.2	2.1	2.5	16.4
i6	284	7.7	7.6	8.1	1.6	4.6	-6.1
i7	411	8.4	7.9	9.6	1.2	2.2	10.9
i9	516	5.6	3.3	1.3	4.5	5.5	0.3
rot	408	8.8	12.3	13.5	4.6	7.0	15.2
term1	206	12.9	13.7	6.0	6.5	5.7	-4.7
ttt2	132	12.5	13.4	-11.2	7.1	5.7	-9.2
x3	485	11.0	13.3	-2.4	7.0	7.1	17.2
Average		8.9	11.7	3.6	4.1	5.7	2.3

Table 3: Results obtained for several MCNC benchmarks for both scenarios considered. The number of gates is given in column **G**.

For each scenario and circuit, two new gate-level descriptions have been created. One of them contains the best transistor reordering for low power for all gates found with the optimization algorithm whereas the other one contains the worst one. A switch-level simulator [11] extracts the power consumption of each description. Thus, the maximum power reduction for each scenario is obtained. The input signals to the circuits used by the switch-level simulator have been generated with an exponential distribution, i.e. time intervals between two consecutive transitions of input signal k to the gate follow an exponential distribution with average $1/D_k$, being D_k the transition density of input signal k.

Table 3 shows the results obtained. Columns **M** and **S** show the power-consumption reduction (best case with regard to worst case for low power) obtained with the model and with switch-level simulations respectively. Column **D** shows the increase in delay (best case for low power with regard to a mapping into the original cell library). The delay increases in most of the benchmarks because not always the best transistor reorderings of a gate for low power and low delay coincide. In fact, the rule of thumb that states that the critical transistor should always be placed near the output terminal to obtain a fast gate contradicts the low power rule of placing it close to the ground node as can be observed in the motivation example (case **(2)**) in Section 1.1 and in [9].

It is shown that the average improvement in power consumption in scenario **A** is 12% with an average increase in delay of 4%. The estimated average improvement is 9%. The reason of this lower value in the estimated improvement is that the model, in general, overestimates the power consumption by an offset, thus leading to a lower estimated improvement.

The power reduction in scenario **B** is roughly half the one in scenario **A**. The power and delay of latches and the clock line in scenario **B** has not been included in the results. In both scenarios there is a small average increase in delay.

Thus, significant power consumption reduction can be obtained in both scenarios with little average increase in delay and it is possible to achieve power reductions without increasing the delay of the circuit.

6 Conclusions

This paper shows that average power reductions of 12% with a 4% increase in delay can be achieved by applying the transistor reordering technique. An optimization algorithm that uses a power-consumption model of a static CMOS gate has been presented. This novel power-consumption gate model takes into account both the probabilities and the transition densities of the inputs of the gates that compose the circuit.

The results suggest that (a) current libraries may be upgraded with more instances of the gates with different transistor reorderings, so that an optimization algorithm can choose the best instance for power reduction and (b) it is possible to obtain power reductions without increasing the delay of the circuit. Our future work in the transistor reordering field is devoted to this second direction.

References

[1] M. Borah, M. Irwin, and R. Owens. Minimizing power consumption of static CMOS circuits by transistor sizing and input reordering. In *Proc. of the Int. Conf. on VLSI Design*, pages 294–298, Jan. 1995.

[2] B. Carlson and C. Chen. Performance enhancement of CMOS VLSI circuits by transistor reordering. In *Proc. DAC*, pages 361–366, 1993.

[3] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms.* McGraw-Hill, 1990.

[4] R. Hossain, M. Zheng, and A. Albicki. Reducing power dissipation in serially connected MOSFET circuits via transistor reordering. In *Proc. of the Int. Conf. on Computer Design*, pages 614–617, Oct. 1994.

[5] E. Musoll and J. Cortadella. Optimizing CMOS circuits for low power using transistor-reordering. Technical report, UPC/DAC, Aug. 1995.

[6] F. Najm. Transition density, a stochastic measure of activity in digital circuits. In *Proc. DAC*, pages 644–649, 1991.

[7] K. Parker and E. McCluskey. Probabilistic treatment of general combinational networks. *IEEE Trans. on Comp.*, C-24:668–670, June 1975.

[8] S. Prasad and K. Roy. Circuit optimization for minimization of power consumption under delay constraint. In *Proc. Int. Workshop on Low Power Design*, pages 15–20, Oct. 1994.

[9] W. Shen, J. Lin, and F. Wang. Transistor reordering rules for power reduction in CMOS gates. In *ASP-DAC*, July 1995.

[10] C. Tan and J. Allen. Minimization of power in VLSI circuits using transistor sizing, input ordering, and statistical power estimation. In *Proc. Int. Workshop on Low Power Design*, pages 75–80, Apr. 1994.

[11] A. van Gerenden. SLS: An efficient switch-level timing simulator using min-max voltage waveforms. In *Proc. VLSI 89 Conf.*, pages 79–88, Aug. 1989.