Estimation of Power Dissipation in CMOS Combinational Circuits

Srinivas Devadas
Dept. of EECS
MIT, Cambridge

Kurt Keutzer
AT&T Bell Laboratories
Murray Hill

Jacob White
Dept. of EECS
MIT, Cambridge

Abstract

The high transistor density now possible with CMOS integrated circuits has made power dissipation an important design consideration. However, power dissipation in a logic circuit is a function of the input vector or vector sequence applied. This makes accurate estimation of worst-case power dissipation extremely difficult, since the number of input sequences that have to be simulated in order to find the sequence that produces the maximum power dissipation is *exponential* in the number of inputs to the circuit. In this paper we show that a simplified model of power dissipation relates maximizing dissipation to maximizing gate output activity, appropriately weighted to account for differing load capacitances. To find the input or input sequence that maximizes the weighted activity, we give algorithms for transforming the problem to a *weighted max-satisfiability* problem, and then present exact and approximate algorithms for solving weighted max-satisfiability. That is, transformations are presented that convert a logic description into a multiple-output Boolean function of the input vector or vector sequence, where each output of the Boolean function is associated with a logic gate output transition. It then follows that an assignment to the input vector or vector sequence which results in a maximum weighted number of these function's outputs becoming 1 corresponds to the input vector or vector sequence causing maximum weighted activity. Algorithms for constructing the Boolean function for dynamic CMOS as well as for static CMOS, which take into account dissipation due to glitching, are presented. Finally, we present efficient exact and approximate methods for solving the so generated weighted max-satisfiability problem.

1 Introduction

The high transistor density now possible with CMOS integrated circuits has made power dissipation an important design consideration. However, power dissipation in a logic circuit is a complex function of the propagation delays, device parameters, specific topology, and, most importantly, the input vector or vector sequence applied. The last aspect makes accurate estimation of worst-case power dissipation extremely difficult, since the number of input sequences that have to be simulated in order to find the sequence that produces the maximum power dissipation is *exponential* in the number of inputs to the circuit.

Estimating worst-case power dissipation in logic circuits is becoming an important problem and a similar problem, that of current estimation, has received recent attention [4]. For some applications, it is essential that a *tight upper bound* on the power dissipation be given. In this paper, we attempt to derive algorithms for *exact and approximate estimation of worst-case power dissipation in dynamic and static CMOS combinational circuits.* Our approach is to use a simple, but reasonable, model for power dissipation that allows for the use of boolean and graph manipulation techniques to determine the input vector set that maximizes dissipation. Then, either the simplified model or detailed circuit simulation can be used to accurately determine the dissipation for that input vector set.

The simplified model of power dissipation relates maximizing dissipation to maximizing gate output activity, appropriately weighted to account for differing load capacitances. To find the input or input sequence that maximizes the weighted activity, we first give algorithms for transforming the problem to a *weighted max-satisfiability* problem. That is, algorithms are presented that convert a logic description into a multiple-output Boolean function of the input vector or vector sequence, where each output of the Boolean function is associated with a logic gate output transition. It then follows that an assignment to the input vector or vector sequence which results in a maximum weighted number of these function's outputs becoming 1 corresponds to the input vector or vector sequence causing maximum weighted activity. Exact and approximate methods are given for solving the so generated weighted max-satisfiability problem.

Algorithms for constructing the multiple-output Boolean function for transitions in dynamic CMOS gates are simplest because dynamic gates reset at the beginning of every clock cycle. Determining the input pattern that maximizes power dissipation in static CMOS circuits is considerably more difficult, stemming from the fact that activity caused by an input vector is dependent on the circuit node values just before the input is applied, and therefore a function of the previous input vector. Thus, we have to search for *a two-vector input sequence* that produces maximum power dissipation. Further, unlike in dynamic circuits, a logic-gate can glitch due to hazards caused by multiple input changes, and therefore can make multiple transitions. The number of transitions a gate can make is a function of the arrival times of the transitions at its inputs, which in turn is dependent on the propagation delays of the gates on the different paths. We give transformational algorithms, *under various delay models*, for static CMOS circuits that produce a multiple-output Boolean function with the property mentioned above.

The max-satisfiability problem has been shown to be NP-complete. We present an exact branch-and-bound algorithm with associated pruning strategies to solve max-satisfiability. We show that the problem can be transformed into one of generating primes for the multiple-output logic function, enabling the use of efficient prime generation algorithms proposed in the past. For large circuits, not amenable to these techniques, we give approximate transformation algorithms, that produce simpler Boolean functions (for which it is relatively easy to find a max-satisfying assignment), *but do not underestimate the power dissipation in the circuit.*

We begin in the next section by briefly describing the physical model used for power dissipation. Transformations used for dynamic CMOS circuits are presented in Section 3. Algorithms for two-level and multi-level static CMOS circuits under a unit-delay and more general delay model are described in Section 4. Solving max-satisfiability is the subject of Section 5 – we also show how the transformations can be approximated in this section. Preliminary experimental results are given in Section 6.

2 A Power Dissipation Model

As mentioned above, CMOS logic circuits only dissipate energy when their node voltages are changing, and this suggests

IEEE 1990 CUSTOM INTEGRATED CIRCUITS CONFERENCE

that computing power dissipation involves finding the circuits transient response. The straight forward approach to determining the maximum power dissipation of a CMOS combinational network is to simulate the network for all possible sets of input vectors, and then to multiply the highest energy dissipation so derived by the maximum rate at which the inputs vectors to the network can change.

As exhaustive simulation of all input patterns is computationally infeasible for a network with more than a few inputs, instead we derive a simplified model of the energy dissipation that can be used, along with with boolean manipulation techniques we discuss in later sections, to efficiently determine the dissipation maximizing input pattern. Then, detailed simulation can be used to more accurately determine the actual dissipation.

A very simple relation between the logical behavior of a CMOS combinational network and the energy the circuits dissipate can be derived based on three simplifying assumptions: that the only capacitance in a CMOS logic gate is at the output node of the gate; that either current is flowing through some path from V_{DD} to the output capacitor, or current is flowing from the output capacitor to ground; and that any change in a logic gate output voltage is a change from V_{DD} to ground or vice-versa. All of these are reasonably accurate assumptions for well-designed CMOS gates [3], and when combined imply that the energy dissipated by a CMOS gate is given by

$$0.5CV_{DD}^2N \qquad (1)$$

where C is the output capacitance for the gate and N is the total number of gate output transitions, from either low to high or high to low.

As can be seen from Eqn. (1), with our simplifying assumptions, maximizing the energy dissipated by a CMOS combinational network over the possible input vector sets involves maximizing the weighted sum of logic gate output transitions, where the weights are given by the gate output capacitances. We will use this result in later sections to show that finding the input pattern that maximizes dissipation can be derived by examining an associated system of boolean equations.

3 Dynamic CMOS Circuits

Finding the inputs to a combinational logic network that maximizes the dissipation in one clock cycle is easiest for networks of dynamic logic gates. This is because at the beginning of a clock cycle all dynamic gate outputs are forced to either a logical one or zero, depending only on the type of gate and *not* on the inputs. In addition, the gate outputs can change at most once during the remainder of the clock cycle, based on the logical function being performed and the inputs *only during that clock cycle.*

Interconnections of N-type dynamic gates may suffer from race conditions that cause erroneous behavior, but correctly-designed networks will have the property that in one clock cycle all the gate outputs will be precharged high, and depending on the input vector, a subset of the gates in the network will undergo a *single* falling transition. This property implies that to find the input patterns which maximize dissipation we need only find a single input vector to the logical network that maximizes the weighted sum of falling transitions, where, as mentioned in Section 2, the weights are determined by the load capacitances. An analogous statement holds for P-type dynamic gates.

If all the gates are N-type and all have the same load capacitances, the problem simplifies to finding the input vector which maximizes the number of falling transitions. To determine this vector, we simply find the OFF-set [1] for each gate output node in the network, and find the input vector which is in the maximum number of gate output OFF-sets. Problems of this form are usually referred to as *max-satisfiability* problems. If weights are introduced, to model the varying load capacitance, we will refer to the resulting problem as a *weighted max-satisfiability* problem. Exact and approximate methods of finding such an input vector are the subject of Section 5.

[1] By OFF-set we mean the subset of all possible input vectors to the logical network that produce a logical zero at the node of interest.

4 Static CMOS Circuits

The problem of determining the maximum dissipation in static CMOS circuits is more complicated then for dynamic circuits, mostly because static gate output nodes are not "reset." This means that the transitions a static gate undergoes depend not only on the present input vector, but also on the previous input vector. Thus, we must search for a *two-vector sequence* that maximizes dissipation, or equivalently, the weighted sum of transitions. Note that this is a simplification that overestimates the dissipation. Ignored is the fact that in order to repeat the same two-vector sequence, the network's inputs must be returned from the second vector in the sequence to the first vector, and this reversed sequence may not provide as much dissipation.

Allowing for this overestimation, we wish to construct a multiple-output Boolean function which depends on the two-vector input sequence such that a maximum weighted satisfying assignment for the function causes maximum weighted activity in the circuit. Constructing this boolean function is complicated for general static CMOS logical networks because gates in these networks can glitch. That is, changing the input vector once may cause multiple transitions to occur on a particular gate's output node, and this can contribute substantially to dissipation. As this glitching phenomenon must be related to differing signal arrival times at a gate's inputs, glitching only occurs in two-level networks with varied gate delays or in multiple level networks, both of which are examined below.

4.1 Two-Level Networks

Two-level, or sum-of-product, networks are logical networks in which the primary inputs feed and gates whose outputs are inputs to or gates. The commonly used PLA's are logical networks of this form, and therefore two-level networks are an important special case. We denote the N primary inputs of the two-level network as i_1, i_2, .. i_N. In order to determine the two-vector sequence that maximizes dissipation in the network, we consider that the first vector $v0$ is applied to the primary inputs of the circuit and the circuit is allowed to stabilize. Then, at time t, we change the inputs to a second vector vt. In addition, we assume that the changes in the i_j's and their complements from $v0$ to vt happen simultaneously – this would correspond to the often-encountered case of combinational logic being embedded between sets of latches. Below, we derive transformational algorithms, under several gate delay models, for computing boolean equations for the transitions made by each gate output in the network.

4.1.1 Zero-Delay Model

Under the zero-delay model, transitions are assumed to happen instantaneously. Therefore, glitches cannot occur at the outputs of any of the gates and each gate can make at most one transition, $0 \rightarrow 1$ or $1 \rightarrow 0$. For an and gate g_i in a two-level network to make a $0 \rightarrow 1$ transition,

$$f_i^r = (c_i \cap v0 = \phi) \bigwedge (c_i \cap vt \neq \phi) \qquad (2)$$

must to be satisfied (evaluate to a logical one). Here, the superscript r denotes that this is the boolean equation for a rising transition, and c_i denotes the cube corresponding to

the and gate g_i. In general, c_i will contain literals corresponding to the variables i_j or their complements. If, for instance, $c_i = \{i_1, \overline{i_2}\}$, the Boolean equation corresponding to the above condition would be

$$f_i{}^r = (1 \oplus v0_1 \vee 0 \oplus v0_2) \bigwedge (1 \otimes vt_1 \wedge 0 \otimes vt_2).$$

This corresponds to the fact that in order for c_i to evaluate to a 1, $i_1 = 1 \wedge i_2 = 0$. The equation simplifies to:

$$f_i{}^r = (\overline{v0_1} \vee v0_2) \bigwedge (vt_1 \wedge \overline{vt_2})$$

Similarly, in order for an and gate g_i to make a $1 \rightarrow 0$ transition,

$$f_i{}^f = (c_i \cap v0 \neq \phi) \bigwedge (c_i \cap vt = \phi) \qquad (3)$$

must be satisfied, where here the superscript f stands for falling transition. For an or gate h_i to make a $0 \rightarrow 1$ transition, we require that the previous values of inputs all be 0, and at least one input make a $0 \rightarrow 1$ transition. Assuming that the and gates $g_1, g_2, .. g_M$ feed the or gate h_i, the boolean equation for the rising transition is:

$$e_i{}^r = ((c_1 \cap v0 = \phi) \wedge (c_2 \cap v0 = \phi) .. \wedge (c_M \cap v0 = \phi)) \bigwedge$$
$$(f_1{}^r \vee f_2{}^r .. \vee f_M{}^r) \qquad (4)$$

Similarly, for an or gate h_i to make a $1 \rightarrow 0$ transition, we require that at least one previous value of input be a 1, and all inputs that were 1 should make a $1 \rightarrow 0$ transition. The condition is encapsulated by:

$$e_i{}^f = ((c_1 \cap v0 \neq \phi) \vee (c_2 \cap v0 \neq \phi) .. \vee (c_M \cap v0 \neq \phi)) \bigwedge$$
$$(((c_1 \cap v0 \neq \phi) \otimes f_1{}^f) .. \wedge ((c_1 \cap v0 \neq \phi) \otimes f_M{}^f)) \qquad (5)$$

The multiple-output Boolean function we are interested in is the function whose outputs correspond to Eqns. 2, Eqns. 3, Eqns. 4 and Eqns. 5. Each of these equations will have an associated weight related to the size of the load capacitance on the gate output involved, as described in section 2, and an assignment for $v0$ and vt that maximizes the sum of the weights associated with the satisfied equations will also maximize dissipation.

4.1.2 Unit-Delay Model

The unit-delay model results in exactly the same analysis as the zero-delay model. Each and gate can make a single transition at most. Since we assume unit delays for all the and gates, this means that the inputs to the or gates change simultaneously. This implies that the or gate makes a single transition, *i.e.* does not glitch. Eqns. 2-5 can be used directly to estimate maximum dissipation in a two-level circuit under the unit-delay model.

4.1.3 A General Delay Model

In the general case, there will be differences in the switching times of the different gates in a network. Large fan-in gates will tend to switch more slowly. Switching times are also affected by fanout loads.

Under a general delay model, delay attributes of different gates can be arbitrary positive integer values. This complicates dissipation estimation since glitches can occur. Assuming, as before, that the inputs change simultaneously, the and gates can make at most a single transition. However, an or gate may see two transitions occurring at different inputs at different times. If the transitions happen to be of the form of Figure 1, we will have a glitch the or gate output.

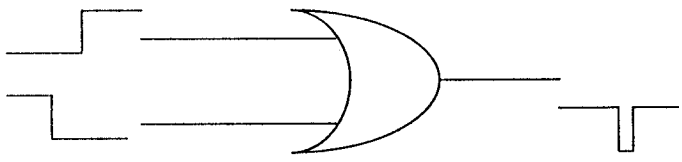

Figure 1: Glitch at or gate Output

Lemma 4.1 *An* or *gate in a two-level logic circuit can make no transition, a* $0 \rightarrow 1$ *transition, a* $1 \rightarrow 0$ *transition or glitch* $1 \rightarrow 0 \rightarrow 1$. *No other output changes can occur under an arbitrary delay model due to the application of a two-vector input sequence.*

Proof: Any and gate that feeds the or gate in question can make at most a single transition $0 \rightarrow 1$ or $1 \rightarrow 0$. If the former occurs, we have a *controlling* value at the input to the or gate, and whatever other input transitions follow, the output of the or gate will not change from logic 1. Thus, if the or gate makes a $0 \rightarrow 1$ transition, its output stays constant afterward. This implies that a $0 \rightarrow 1 \rightarrow 0$ glitch cannot occur, and the only transitions possible are those enumerated above. ■

If we wish to find the correct input pattern that maximizes the dissipation under a general delay model, we need to augment the set of equations derived earlier to include or gate glitching. The and gate equations, namely, Eqns. 2 and Eqns. 3 remain the same. For each or gate, the strategy we will use is to construct a 2-output function that corresponds to the number of transitions the gate will make. A value of 00 for the 2-output function corresponds to no transitions, 01 corresponds to a $0 \rightarrow 1$ transition, 10 corresponds to $1 \rightarrow 0$ transition, and finally 11 corresponds to a $1 \rightarrow 0 \rightarrow 1$ glitch (two transitions).

The conditions necessary for a $0 \rightarrow 1$ transition are simply those given by Eqns. 4. The conditions for a $1 \rightarrow 0$ transition and *no change thereafter* are given by Eqns. 5. The conditions for a $1 \rightarrow 0 \rightarrow 1$ glitch are summarized below:

- No and gate stays at 1.
- At least one and gate feeding the or gate should make a $1 \rightarrow 0$ transition.
- At least one and gate should make a $0 \rightarrow 1$ transition.
- All the and gates making $1 \rightarrow 0$ transitions should do so *before* the and gates making $0 \rightarrow 1$ transitions.

We can write Boolean equations for each of the conditions above:

$$((c_1 \cap v0 = \phi) \vee (c_1 \cap vt = \phi)) \bigwedge ..$$
$$\bigwedge ((c_M \cap v0 = \phi) \vee (c_M \cap vt = \phi))$$

$$f_1{}^r \vee f_2{}^r \vee .. \vee f_M{}^r$$

$$f_1{}^f \vee f_2{}^f \vee .. \vee f_M{}^f$$

$$\overline{(f_1{}^r\, f_2{}^f\, (d_1 \leq d_2)} \wedge .. \wedge \overline{f_1{}^r\, f_M{}^f\, (d_1 \leq d_M))} \bigwedge$$

$$\overline{(f_2{}^r\, f_3{}^f\, (d_1 \leq d_2)} \wedge .. \wedge \overline{f_2{}^r\, f_M{}^f\, (d_1 \leq d_M))} \bigwedge .. \bigwedge$$

$$\overline{(f_{M-1}{}^r\, f_M{}^f\, (d_{M-1} \leq d_M))} \qquad (6)$$

where $f_i{}^r$ and $f_i{}^f$ are as defined in Eqns. 2 and Eqns. 3, and d_i is the delay of and gate g_i.

One can now use the max-satisfiability algorithms of Section 5 on the constructed function to find the two-vector input sequence that maximizes dissipation under an arbitrary delay model. It should be noted that given a circuit, the d_i values are constants, and the inequalities in Eqns. 6 evaluate to 1 or 0.

4.2 Multi-Level Static CMOS Circuits

In this section, we will generalize the transformations presented in the previous section to apply to arbitrary multi-level implementations under the zero and unit delay assumptions. As before, the multi-level circuit is assumed to have N inputs i_1, i_2, .. i_N, and we will assume that the first vector $v0$ is applied to the those inputs and the circuit allowed to stabilize, then at time t, we apply the second vector vt. Again, we will assume that the changes in the i_j from $v0$ to vt happen simultaneously.

4.2.1 Zero-Delay Model

Under the zero-delay model, each gate in a multi-level circuit can make at most a single transition (since all transitions happen instantaneously). Therefore, we can use the logic function of each gate to determine if the gate will switch or not on the application of vt. This is simply represented by the condition:

$$f_i = ((h_i \cap v0 = \phi) \wedge (h_i \cap vt \neq \phi)) \bigvee$$

$$((h_i \cap v0 \neq \phi) \wedge (h_i \cap vt = \phi)) \qquad (7)$$

where h_i is the logic function corresponding to gate g_i in the multi-level network. Finding an input sequence $v0$, vt such that maximize $\sum_j C_j$ where C_j is the output capacitance of gate j and the sum is taken over the indices j for which f_j is satisfied.

4.2.2 Unit-Delay Model

Even under the idealization of a unit-delay model, the gate output nodes of a multi-level 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. For problems of this kind, it is easiest to construct the boolean equations that describe the dependence of gate output transitions on the input sequence $v0$, vt by a post-processing of symbolic simulation.

Specifically, we construct the boolean functions describing the gate outputs at the discrete points in time implied by the unit-delay model. That is, we consider only discrete times $t, t+1, ..., t+l$, where t is the time when the input changes from $v0$ to vt, and l is the number of levels in the network. For each gate output i, we use symbolic simulation to construct the l 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 as we assume no gate has zero delay and that the network has settled before the inputs are changed from $v0$ to vt, $f_i(t)$ is the logic function performed on $v0$ at the i^{th} gate output. Finally, we can determine whether a transition occurs at a boundary between discrete time intervals $t+i$ and $t+i+1$ by XOR'ing $f_i(t+i)$ with $f_i(t+i+1)$.

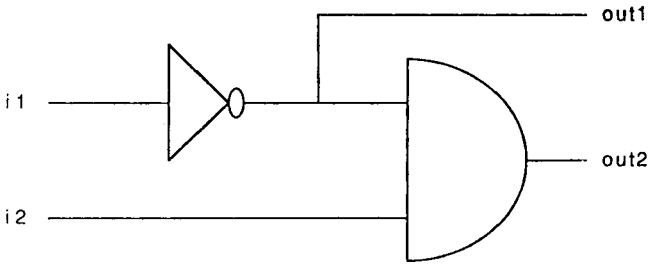

Figure 2: Two-Level Inverter-And Network

For example, consider the two-level inverter-AND network in figure (2). For this network,

$$f_1(t) = \overline{v0_1}$$

$$f_2(t) = \overline{v0_1} \wedge v0_2.$$

Assuming both gates have unit delay,

$$f_1(t+1) = \overline{i_1(t)} = \overline{vt_1}$$

$$f_2(t+1) = f_1(t) \wedge i_2(t) = \overline{v0_1} \wedge vt_2.$$

Finally,

$$f_2(t+2) = f_1(t+1) \wedge i_2(t+1) = \overline{vt_1} \wedge vt_2$$

For this example there are three possible transitions: that the inverter changes state from t to $t+1$, that the and gate changes state from t to $t+1$, and that the AND gate changes state from $t+1$ to $t+2$. The boolean equations for these transitions are respectively:

$$e_1 = f_1(t) \oplus f_1(t+1)$$

$$e_2 = f_2(t) \oplus f_2(t+1)$$

$$e_3 = f_2(t+1) \oplus f_2(t+2).$$

Note, if it is possible to find a two-vector sequence $v0$, vt that simultaneously satisfies e_1, e_2 and e_3, $v0$, vt is the input sequence that will maximize dissipation.

It is not usually necessary to generate an $f_i(t+j)$ for all values of j between 0 and l, many of these terms can be discarded by considering two easily computed quantities.

Definition 4.1 *The minrank of a logic gate output is one plus the minimum of the minranks of the logic gate's inputs. Primary inputs have a minrank of zero.*

Definition 4.2 *The maxrank of a logic gate output is one plus the maximum of the maxranks of the logic gate's inputs. Primary inputs have a maxrank of zero.*

The *minrank* and *maxrank* quantities are useful because of the following lemma.

Lemma 4.2 *The boolean equations for all possible transitions of the i^{th} logic gate can be determined from* XOR*'ing neighbors in the ordered set of boolean functions*

$$\{f_i(t),\ f_i(t+minrank_i),\ f_i(t+minrank_i+1), ...,$$

$$f_i(t+maxrank_i-1),\ f_i(t+maxrank_i)\} \qquad (8)$$

The above lemma follows directly from the unit delay model. A gate output cannot change until the change in the nearest input propagates through, and will stop changing when the input furthest away finally arrives.

5 Solving Max-Satisfiability

5.1 Introduction

The classical definition of max-satisfiability involves finding a satisfying assignment for *clauses* in a Boolean function, where the clauses represent disjunctions of literals [2]. Algorithms have been proposed (*e.g.* [5]) for exact and approximate solutions to this problem. Here, we are concerned with finding satisfying assignments for arbitrary Boolean functions that can themselves involve conjunctive and disjunctive terms. In the sequel, we will present two strategies we have developed for general max-satisfiability.

5.2 A Branch-and-Bound Algorithm

Max-satisfiability can be solved using a branch-and-bound strategy. The efficiency of any such algorithm depends greatly on the pruning/bounding methods that are used while branching over various solutions. We use a pruning method based on a maximal disjoint set heuristic.

The algorithm is described below. We assume that we are given the ON-sets of the N outputs in the multiple-output Boolean function, in sum-of-product form. U corresponds to the set of all N outputs. L corresponds to the current set of selected outputs. R corresponds to the currently remaining outputs. Initially, $R = U$, $L = \phi$.

1. Remove all outputs from R whose ON-sets have a null intersection with the intersected ON-sets of the current element set L.

2. Find a maximal set of maximal disjoint groups of outputs in R, namely D_1, D_2, .. D_M. A disjoint group of outputs satisfies the property that the the pairwise intersection of their ON-sets is null. No output can belong to more than one D_i. If ($||L|| + ||R|| - \sum_{i=1}^{M}(||D_i|| - 1)$) is less than or equal to the best solution found thus far, return from this level of recursion. Else, if $R = \phi$, declare this solution as the best recorded thus far.

3. Heuristically select an output f from R (Its ON-set will have a non-null intersection with the intersected ON-sets of the selected output set, L). Recur with $L = L \cup f$ and $R = R - f$. Recur also for L unchanged, $R = R - f$.

The bounding strategy used is as follows: If at any given point in the algorithm, we have a group of outputs in R whose pairwise ON-set intersections are all null, it implies that we can select at most one output from the set. This corresponds to the $||D_i|| - 1$ term in Step 2 above. Finding a *maximum* disjoint group is itself NP-complete, however, we need only a large disjoint group, and therefore we use a fast, greedy algorithm for this purpose. By finding large, disjoint groups efficiently we can prune the search space considerably and many searches can be terminated high up in the recursion.

5.3 Max-Satisfiability Via Prime Implicant Generation

In this section, we will show how the set of primes for a multiple-output Boolean function can be searched to find a maximum satisfying assignment over a set of weighted outputs. This is a useful observation since efficient prime generation algorithms (*e.g.* [6]) have been developed in the past.

Theorem 5.1 : *Given a multiple-output Boolean function with associated positive weights for individual outputs, a maximum satisfying assignment can be found by inspecting all the primes of the multiple-output function.*

Proof: Assume that a max-satisfying assignment corresponds to setting the set of outputs $S \subseteq U$ to 1. In order to prove the theorem, we have to show that a prime exists with an output part that contains the outputs in S. Assume not. Since we have a simultaneous satisfying assignment for the outputs in S, it implies that we have a cube in the Boolean n-space corresponding to the multiple-output function whose output part contains the outputs in S. By the definition of primality, this cube is either a prime or is covered by a prime. In the former case, we have a direct contradiction. In the latter case, the prime that covers this cube has to contain all the outputs in S in its output part, leading to a contradiction. ■

To find a maximum satisfying assignment under arbitrary weights for outputs, we simply walk down the list of primes computing the value of the weight function for each prime, and pick the prime with the highest value.

5.4 Approximation Strategies

For very large circuits, the algorithms described in the previous two sections may require too much memory or CPU time to complete. In this case, it is of interest to find a tight upper bound on the maximum power dissipation. Since we are dealing with worst-case power, the approximation or bounding algorithm should *not* underestimate the power dissipation.

The number of primes in a multiple-output Boolean function strongly depends on the number of outputs in the function. One way of reducing the number of primes, thereby easing the problem of max-satisfiability, it to merge two (or more) outputs together into a single output that is the Boolean or of the two outputs, with an associated weight that is the sum of the individual weights. This implies that we have *expanded* the ON-sets of the two outputs. Expansion of the ON-sets provides a correct bound, since the weight function corresponding to any assignment only increases (or stays constant). Arbitrarily large expansions will result in undesirably loose bounds – one wishes therefore to select outputs with large ON-set intersections and merge these outputs together, rather than outputs with small/null ON-set intersections.

We can compute the worst-case error due to an expansion of outputs as follows: Let the original ON-sets be ON_i, $1 \leq i \leq N$. Let the expanded ON-sets be ON_i'. Compute $E_i = ON_i' - ON_i$. Find a set of maximal disjoint groups of outputs using the E_i, namely D_1, D_2, .. D_M, such that $D_1 \cup D_2 \cup .. D_M = Q$, where Q is the set of outputs such that $E_i \neq \phi$, and no output belongs to more than one D_k. In the worst-case we may erroneously assume satisfiability for an output from *each* of the D_i. Thus, the worst-case error is given by M.

Merging outputs can also reduce the complexity of the branch-and-bound algorithm. In the worst-case, one searches for all possible 2^N combinations of intersecting outputs, given N outputs. Reducing N can speed up the algorithm considerably. This approximation allows power estimation for arbitrarily large circuits, but an intelligent merging of outputs has to be performed in order not to significantly overestimate worst-case power dissipation.

EX	#inp	#out	#gate	#level
tckt1	5	3	31	2
tckt2	7	3	147	2
tckt3	9	1	140	2
tckt4	16	65	231	2
mckt1	8	38	45	3
mckt2	7	36	53	4
mckt3	8	27	104	6
mckt4	16	27	195	8

Table 1: Statistics of Examples

Another means of approximation is simply to assume that certain gates make the maximum number of possible transitions (bounded by the number of levels of logic that precede the gate), and not generate the transition equations for these gates. This is especially useful when a large number of gates occur in the first 2-3 levels of the logic circuit and much fewer gates occur in later levels. We ignore the transition equations corresponding to the "deeper" gates and add their maximum number of possible transitions to the worst-case estimate obtained on the circuit after deleting these gates.

6 Experimental Results

In this section, we present preliminary experimental results using the power estimation techniques described earlier. We chose several PLA and multilevel circuit examples whose statistics are summarized in Table 1. The PLA examples are **tckt1** through **tckt4** and the multilevel circuits are **mckt1** through **mckt4**. The number of inputs, outputs, gates and logic levels in each of the circuits are given in Table 1.

We used a general delay model for the two-level circuits, since the difference in the sizes of the gates was large. A unit-delay model was used for the multilevel circuits.

In Table 2, we present the results obtained using the transformation and max-satisfiability algorithms. The number of inputs and outputs in the transformed logic function is indicated. The total CPU time required on a $\mu VAX - III$ for transformation, collapsing to a two-level representation using the program MIS [1], and max-satisfiability checking by prime generation using the program ESPRESSO [6] are given in Table 2. The maximum number of outputs that could be simultaneously satisfied is also indicated. In certain cases, prime generation on the transformed circuit was not possible. For all the examples (even those for which exact max-satisfiability was possible), we used the approximation strategy (which reduces the complexity of the satisfiability check) described in Section 5. The difference between the approximation and exact strategies is small, however, the approximation algorithm takes significantly less CPU time and is viable for larger functions. The worst-case error (overestimation error) using the approximation strategy is also given for each of the examples – for examples **tckt1** through **tckt3**, and examples **mckt1** through **mckt3**, the error in approximation is significantly smaller than the worst-case error.

We are currently implementing the branch-and-bound strategy for max-satisfiability described in Section 5. The pruning strategies in the algorithm will afford exact max-satisfiability for significantly larger functions.

7 Conclusions and Acknowledgements

The difficulty in estimating worst-case power dissipation in combinational circuits mainly stems from the fact that power dissipation is input pattern dependent and the number of possible input patterns grows exponentially with the number of circuit inputs. By focusing on a simplified model of power dissipation, namely weighted gate output activity, we were able to transform the problem to one of weighted max-satisfiability. We developed exact and approximate algorithms to solve weighted max-satisfiability. The approximate algorithms are robust in the sense that they do not underestimate the worst-case power dissipation in the circuit.

EX	EXACT		APPROXIMATE		
	CPU time	#trans	CPU time	#trans	w.c. error
tckt1	20m	11	10m	11	5
tckt2	3.9h	53	1.8h	57	15
tckt3	4.8h	59	1.7h	64	12
tckt4	> 10h	–	2.5h	72	15
mckt1	1.7h	22	0.7h	23	5
mckt2	1.9h	22	1.0h	24	10
mckt3	3.5h	33	1.6h	37	10
mckt4	> 10h	–	2.1h	80	15

Table 2: Worst-Case Power Dissipation Results

In future work we will examine the effectiveness of our approximate algorithm for max-satisfiability applied to the equations generated by more general multi-level networks. In addition, similar techniques can be applied to finding the peak supply current densities. This is also an important problem because peak current densities are linked to the rate of metal migration failures.

The author's would like to thank Dr. Y. T. Yen of Digital Equipment Corp. for bringing this problem to our attention, and Mark Coiley for many valuable discussions on the subject. This work was supported in part by the Defense Advanced Research Projects Agency under contract N00014-87-K-0825, and grants from Digital Equipment Corporation and Analog Devices.

References

[1] R. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, and A. Wang. MIS: A Multiple-Level Logic Optimization System. In *IEEE Transactions on Computer-Aided Design*, pages 1062–1081, November 1987.

[2] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness.* W. H. Freeman and Company, 1979.

[3] L. Glasser and D. Dobberpuhl. *The Design and Analysis of VLSI Circuits.* Addison-Wesley, 1985.

[4] R. Burch F. Najm P. Yang D. Hocevar. Pattern-independent current estimation for reliability analysis of cmos circuits. In *ACM/IEEE 25th Design Automation Conference*, pages 294–299, 1988.

[5] D. S. Johnson. The np-completeness column: an ongoing guide. *Journal of Algorithms*, 6:291–305, 1985.

[6] R. Rudell and A. Sangiovanni-Vincentelli. Multiple-Valued Minimization for PLA Optimization. In *IEEE Transactions on Computer-Aided Design*, pages 727–751, September 1987.