VLSI Design, 2002 Vol. 15 (1), pp. 407–416

Genetic Spot Optimization for Peak Power Estimation in Large VLSI Circuits

MICHAEL S. HSIAO*

The Bradley Department of Electrical and Computer Engineering, 340 Whittemore, Virginia Tech. (VPI), Blacksburg, VA 24061, USA

(Received 12 February 01; Revised 10 July 01)

Estimating peak power involves optimization of the circuit's switching function. The switching of a given gate is not only dependent on the output capacitance of the node, but also heavily dependent on the gate delays in the circuit, since multiple switching events can result from uneven circuit delay paths in the circuit. Genetic spot expansion and optimization are proposed in this paper to estimate tight peak power bounds for large sequential circuits. The optimization spot shifts and expands dynamically based on the maximum power potential (MPP) of the nodes under optimization. Four genetic spot optimization heuristics are studied for sequential circuits. Experimental results showed an average of 70.7% tighter peak power bounds for large sequential benchmark circuits was achieved in short execution times.

Keywords: Power estimation; Peak power; Non zero delay; Genetic optimization; Low-power design

INTRODUCTION

The continuing decrease in feature size, increase in chip density, and increase in clock frequency in recent years have given rise to concerns about excessive power dissipation in VLSI chips. Since the failure rate for components roughly doubles for every 10°C increase in operating temperature [1], circuits become less reliable as large instantaneous power dissipation can cause overheating (local hot spots). Furthermore, the growing market of portable computing products such as cellular phones and portable computers demands low-power consumption for long operational lifetime.

The power dissipated in CMOS circuits is a complex function of the gate delays, clock frequency, process parameters, circuit topology and structure, and the input vectors applied. Once the processing and structural parameters have been fixed, the measure of power dissipation is dominated by the switching activity (toggle counts) of the circuit. It has been shown in Refs. [2–4] that power estimation can be extremely sensitive to different gate delays, since multiple toggles at internal nodes can result due to uneven circuit delay paths. Both Refs. [2,3] computed the *upper bound* of maximum transition (or switching) density of individual internal nodes of a combinational circuit via propagation of uncertainty waveforms, while Ref. [4] computed the sensitivity of internal nodes due to a switch on the primary input. While these measures are useful to compute the bounds for individual signals, they cannot be used to compute a tight bound of peak power dissipation on the entire circuit.

Unlike average power estimations [15–22] in which signal switching probabilities are sufficient to compute the average power, *peak* power is associated with a specific starting circuit state and a specific sequence of vectors that produce the power. Furthermore, as there are significantly more flip-flops than primary inputs in larger circuits, most activity is caused by switching between the initial and the intermediate states. Thus, both the initial and intermediate states traversed play significant roles in determining maximum power.

Several approaches to measuring maximum power in CMOS VLSI circuits have been addressed in the recent years. The problem of worst-case power computation was transformed to a weighted max-satisfiability problem on a set of multi-output boolean functions, obtained from the logic description of a *combinational* circuit [5]. Its limitations include the difficulty of incorporating delay information to this already very complex problem and the inability to handle large circuits efficiently. Peak current estimation for combinational circuits was addressed in Refs. [6,7]. The authors' approach was to find the time

*E-mail: mhsiao@visc.vt.edu

ISSN 1065-514X print/ISSN 1563-5171 online © 2002 Taylor & Francis Ltd
DOI: 10.1080/1065514021000012020

window during which a gate in the circuit could switch. Partial input enumerations were performed to resolve correlations and to make the time window smaller. A third approach using symbolic transition counts to compute maximum power cycles was introduced in Ref. [8], in which the state transition diagram (STG) is used to find the maximal average length cycle in the graph. The dual graph necessary to compute the power dissipation in a small circuit, s208, already exceeded 71 million edges [8]. An automatic-test-generation (ATG)-based estimation technique for sequential circuits was proposed in Ref. [9], in which the aim is to create toggles in the circuit for gates with the greatest numbers of fan-outs. However, only zero-delay power estimates were used, since ATG techniques are not easily adaptable to handle delay parameters. Ref. [10] extended the ATG approach to handle gate delays by circuit expansion in which multiple copies of internal gates are added at various propagation times for each gate. For small combinational circuits, the expanded circuit exceeds more than 500% increase in gate count. The expansion is even more significant as circuits become larger because the number of possible propagation times of its internal nodes increases with larger circuits. No results were reported for sequential circuits. A continuous optimization approach was proposed in Ref. [11], in which maximum power for combinational circuits is computed by transforming the circuit into a continuous function over a unit hypercube in the Euclidean space. Finally, a genetic-algorithm (GA) based approach was proposed in Refs. [12–14] in which the GA is used to maximize switching activity in both combinational and sequential circuits over various sequence lengths.

The GA-based technique [12,14] gave very tight lower-bounds on peak power. Comparisons with 100 million random state-vector tuples (more than 23 CPU hours of computation on the largest of the eight circuits) were made, and the best peak power estimates from 100 million tuples still lagged 4.4% behind the GA-based approach, which took less than 1 min for the all of the eight circuits. This suggests that if random-based methods were to achieve similar tightness on peak power bounds as the GA-based approach, several orders of magnitude higher in execution times would be needed. Although GA-based techniques significantly out-perform the random approach, it is unclear whether the lower-bounds can still be tighter for larger circuits.

Two major obstacles are faced when trying to achieve tight peak power bounds for large circuits. The first is in avoiding local maxima in the huge search space, and the second is in methods to get out of local maxima once the search is stuck at these points. Since peak power estimation attempts to maximize total switching activity in the circuit, there may exist many local maxima. Search algorithms may be stuck on these local maxima because the algorithms generally explore the search space in a fixed fashion. For instance, fewer toggles on more gates may dissipate more power than having more toggles on fewer high-output-capacitance gates. Moreover, switching activity of a given node in the circuit is not only dependent on the output capacitance of the node, but also heavily dependent on the gate delays in the circuit, since multiple switching events can result due to uneven circuit delay paths in the circuit. These added factors make peak power even more difficult to estimate.

This paper proposes various *genetic spot optimization* techniques in which we try to optimize activity on a spot (group of nodes) at a time. The optimization spot can shift, shrink, and expand based on the maximum power potential (MPP) of the spot. Unlike maximum switching density, which is based on signal probabilities in the circuit, MPP takes both the number of possible transitions and output capacitance into consideration.

Four genetic spot-optimization techniques are proposed and studied: (1) node-based, (2) path-based, (3) cone-based, and (4) distance-based. The first three techniques aim to maximize switching on nodes, paths, or cones with the highest MPP, respectively. The fourth technique follows from the observation that the initial and intermediate states play important roles and aims to exploit weighted distances of these two states. All four genetic spots can change and grow dynamically during the optimization process. Experiments show that genetic spot-optimization techniques are very effective for obtaining tighter bounds for larger circuits.

The remainder of the paper is organized as follows. The second section explains the delay model, peak power model, and genetic algorithm used in this work; third section describes the MPP and the various heuristics used for estimating peak power; experimental results are reported and discussed in the fourth section; finally, fifth section concludes the paper.

PRELIMINARIES

Delay Model

Since glitches and hazards are not taken into account in a zero-delay framework, the power dissipation measures may be off greatly from the actual powers [13]. Peak power measures can be very sensitive to the delay model used when peak powers are high [13]. Thus, it is important to use an accurate delay model during power estimation. However, it should be noted that any delay model can be handled by our framework.

Variable delay model is chosen for this work. Traditionally, a simple variable delay model based on the number of fan-outs has been used [10]. Though more accurate than the unit-delay model (e.g. every gate is assigned identical delay of one unit), fan-outs that feed larger gates are not taken into account, resulting in inaccuracies. A different variable delay model based on fan-outs of a given node as well as fan-ins of its successor nodes is used [13]. The gate delay data for various types and sizes of gates are obtained from a VLSI library.

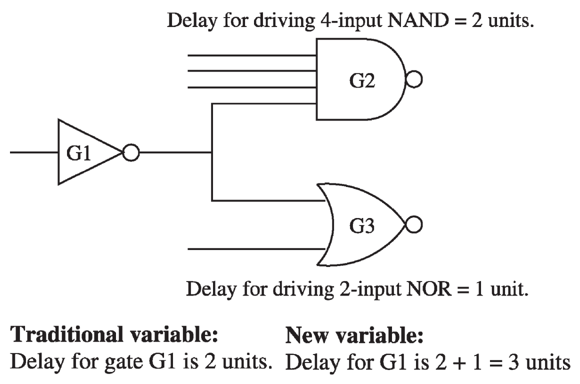
Delay for driving 4-input NAND = 2 units.
G2
G1
G3
Delay for driving 2-input NOR = 1 unit.
Traditional variable:
Delay for gate G1 is 2 units.
New variable:
Delay for G1 is 2 + 1 = 3 units

FIGURE 1 Variable delay models.

The difference between the two types of variable delay models for a typical gate is shown in Fig. 1. From this figure, the output capacitance of gate *G1* is estimated to be 2 units (the number of fan-outs) in the traditional variable delay model, while the delay calculated using the new variable delay model is proportional to the delay associated with driving the successor gates *G2* and *G3*, or simply $2 + 1 = 3$ units. Since traditional variable delay of a given gate does not consider the size of its succeeding gates, the delay calculations may be less accurate.

Peak Power Model

The unit of power used throughout the paper is energy per clock cycle and will simply be referred to as power. In estimating peak power in combinational circuits, the task is simply to search for a pair of two vectors (V_1, V_2) that generates the most gate transitions. In a typical sequential circuit, on the other hand, the switching activity is largely controlled by the state vectors and less influenced by input vectors, because the number of flip-flops far outweighs the number of primary inputs. As shown in Fig. 2, the power is controlled by both initial state S_1 and input vectors V_1 and V_2. The state S_1 and input vector V_1 initialize all gate outputs and determine the next state S_2. Then vector V_2 and state S_2 switch some of the gates, which accounts for the power dissipation. We will obtain a three-tuple (S_1, V_1, V_2) that maximizes this power under the new variable delay model. In fully-scanned circuits, the state S_1 can be initialized to any arbitrary value, and therefore, this bound is attainable in practice. However, in cases where the initial state is not fully controllable, we can only speculate that during the operation of the circuit, the machine may reach state S_1, and only then we can be assured that the bound is attainable.

The power dissipated in the combinational portion of the sequential circuit can be computed as

$$P = \frac{V_{dd}^2}{2 \times \text{cycle period}} \times \sum_{\text{for all gates } g} [\text{toggles}(g) \times C(g)], \quad (1)$$

where the summation is performed over all gates g, and toggles(g) is the number of times gate g has switched from 0 to 1 or vice versa within a given clock cycle; $C(g)$ is the output capacitance of gate g. Switching frequency (SF) per node is reported instead of total power in this paper, and it is computed simply as the second portion of Eq. (1) divided by the total number of capacitive nodes in the circuit (computed as the total number of gate inputs in the circuit),

$$\text{SF} = \frac{\sum_{\text{for all gates } g} [\text{toggles}(g) \times C(g)]}{\text{total number of capcacitive nodes}}. \quad (2)$$

In this work, we made the assumption that the output capacitance for each gate is equal to the number of fan-outs; however, assigned gate output capacitances can be handled by our optimization technique as well.

Genetic Algorithms

The GA framework used in this work is similar to the simple GA described by Goldberg [23]. The GA contains a population of *strings*, also called *chromosomes* or *individuals*, in which each individual represents a state-vector tuple. Peak power estimation requires a search for the three-tuple (S_1, V_1, V_2) that maximizes power dissipation. This three-tuple is encoded as a single binary string, as shown in Fig. 3. The population size depends on the string length, which depends on the number of primary inputs and flip-flops. Larger populations are desired to accommodate longer individuals in order to maintain diversity [23]. However, larger populations also imply more simulation would be needed. In order to keep the population small enough for short execution times, but large enough to maintain diversity, the population size is set equal to $128 \times \sqrt{\text{string length}}$, as suggested by the empirical results obtained in Refs. [12–14,27]. Finding

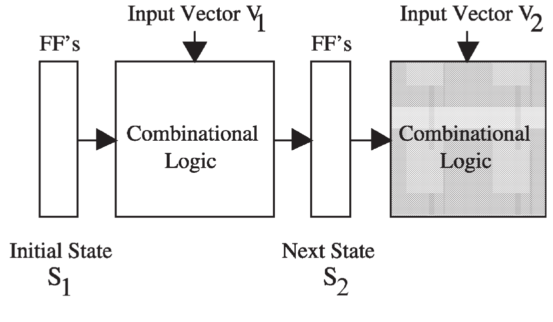

FIGURE 2 Power model for sequential circuits.

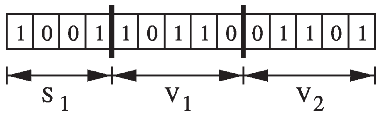

FIGURE 3 Encoding of an individual.

the optimal GA population size for each specific circuit is not the goal of this research, and as the results will illustrate that very tight bounds for peak power are obtained with this population size.

Each individual has an associated *fitness*, which measures the quality of the vector sequence in terms of switching activity, indicated by the total number of capacitive nodes toggled by the individual. The delay model and the amount of capacitive-node switching are taken into account during the evolutionary processes of the GA via the fitness function. The population is first initialized with random strings. A variable-delay logic simulator is then used to compute the fitness of each individual. The evolutionary processes of *selection*, *crossover*, and *mutation* are used to generate an entirely new population from the existing population. Evolution from one generation to the next is continued until a maximum number of generations is reached. In this work, a maximum of 32 generations is allowed. To generate a new population from the existing one, two individuals are selected, with selection biased toward more highly fit individuals. The two individuals are crossed to create two entirely new individuals, and each character in a new string is mutated with some small mutation probability. A mutation probability of 0.01 is used in this work, and since a binary coding is used, mutation is done by simply flipping the bit. The two new individuals are then placed in the new population, and this process continues until the new generation is entirely filled. At this point, the previous generation can be discarded. In our work, we use tournament selection without replacement and uniform crossover. In *tournament selection without replacement*, two individuals are randomly chosen and removed from the population, and the best is selected; the two individuals are not replaced into the original population until all other individuals have also been removed. Thus, it takes two passes through the parent population to completely fill the new population. In *uniform crossover*, bits from the two parents are swapped with probability of 0.5 at each string position in generating the two offspring. Because selection is biased toward more highly fit individuals, the average fitness is expected to increase from one generation to the next. However, the best individual may appear in any generation, so we save the best individual found.

The fitness function is a simple counting function that measures the amount of capacitive-node switching in a given time period. Since the majority of time spent by the GA is in fitness evaluation, parallelism among the individuals can be exploited. Parallel-pattern simulation [24] is used where 32 candidate sequences from the population are simulated simultaneously by bit-packing the sequences into 32-bit words.

GENETIC SPOT OPTIMIZATION

The main obstacles that peak power estimation have to overcome are (1) avoiding local maxima in the huge search space and (2) getting out of local maxima once the search is stuck at those points. Since peak power estimation attempts to maximize total switching activity in the circuit, many local maxima exist inside the search space. As circuit sizes become larger, finding a tight bound on peak power becomes more difficult. Thus, instead of maximizing the total power on the entire circuit all at once, genetic spot optimizations are proposed. A spot is a group of nodes on which the optimization is to be performed. Genetic techniques allow the spot to shift, shrink, and grow dynamically during the optimization process.

In addition, peak power estimation must consider both the number of toggles and the number of nodes with large output capacitances simultaneously during the maximization process. Maximization on either aspect alone is insufficient to ensure a tight bound on peak power measures; this is even more so with large circuits. Optimizations which merely try to generate one transition on as many gates with large output capacitances as possible may miss the opportunity to produce larger power consumption by generating more transitions on fewer gates. Conversely, maximizing merely the number of transitions may overlook the cases where greater power dissipations may result from fewer transitions on more nodes with larger output capacitances. Optimization of either aspect alone is already a very difficult task, making peak power estimation an even harder task, since both aspects need to be considered at the same time. Finally, in sequential circuits, getting to a *right* intermediate state is crucial, since the power consumption in the second time frame depends heavily on it.

Four genetic spot-optimization heuristics are used in large circuits. All four heuristics try to maximize the circuit's switching activity by dynamically expanding localized spots. These four heuristics are node-based, path-based, cone-based, and distance-based, respectively. In Ref. [9], the ATG-based approach tries to generate a vector pair that produces a toggle on the gate(s) with most number of fan-outs in combinational circuits. The node-based heuristic is similar to that of Ref. [9] except that we take delay information into account. The second and third heuristics localize maximal switching on paths and cones. The final heuristic aims to exploit state distance. All four heuristics will be described in detail. Before the explanations for each technique, the MPP with which genetic expansion is based on will be discussed first.

Maximum Power Potential

All four heuristics require the knowledge of which nodes have the maximum potential for switching. Instead of merely relying on output capacitance or signal probability, a node's MPP takes into consideration both the output capacitance and the number of possible transitions. The delays in various paths that lead to a node contribute to different possible times on which the node may toggle. For instance, Fig. 4 shows the lists of possible transition times

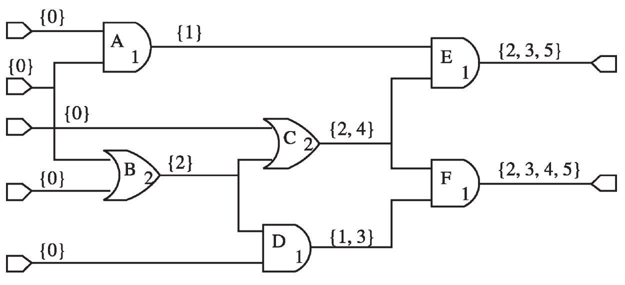

FIGURE 4 Computation of possible transition times.

for each gate in the circuit. Variable delay model is used and gate delays are indicated inside each gate.

Enumeration of possible transition times for each gate can be done in a levelized fashion in $O(n)$ time, where n is the number of gates in the circuit. All inputs to the combinational portion of the circuit (i.e. primary inputs and flip-flop outputs) have only one possible transition time, namely 0. The set of transition times for any internal node can simply be obtained from a two-step process. First, we form a set union of the sets of transition times from all the immediate input nodes. Next, we offset each element in the set union by the gate delay of the current node. For example, in order to compute the set of possible transition times for node E, we first union the transition time sets of its inputs A and C: $\cup\ \{\{1\},\{2,4\}\} = \{1,2,4\}$. Then, we add the gate delay of node E, 1, to each element in the union, resulting in the set $\{2,3,5\}$. It should be noted that the set of possible transition times for each node varies with the underlying delay model, but all can be computed in the same manner.

Once the transition times have been determined for all gates in the circuit, the MPP for a given node is computed as the *product* of the *cardinality of its set of transition* and its *output capacitance*:

$$\mathrm{MPP}(n) = |S_n| \times C_n. \tag{3}$$

Note that under the zero-delay assumption, $|S_n| = 1$ since a gate can transition at most once in a time frame, so MPP of a gate is simply the output capacitance of that gate.

Although transitions may not be possible to occur at all the possible times in the set S_n for a given node n, the MPP values computed in Eq. (3) can be done very quickly, and they serve as good guidances to selecting groups of nodes during spot optimization. A good guidance is one which selects the initial spot as well as expands the spot in a way that will least likely trap the search to a local maximum during the search process.

Node-based Heuristic

This is the most simple heuristic in which the optimization spot begins at the node with the greatest MPP. This approach is based on the assumption that peak power consumed in the circuit will likely include toggles on nodes with greatest MPP. This heuristic is similar to the idea used in Ref. [9], except that only zero-delay model was considered in Ref. [9].

Even though delay is considered in our case, by considering only one or a few nodes with greatest MPP there may still remain shortcomings during the optimization process, particularly if many switches on a high-capacitive node may be less favorable than having more low-capacitive nodes switch multiple times in the circuit. To remedy this problem, the spot on which the optimization focuses on expands steadily to include more nodes that can potentially increase the peak power. Genetic spot expansion is done by bringing additional nodes of greatest MPP that are not currently in the optimization spot. Note that additional nodes may not be neighboring nodes. In doing so, the GA no longer works on a static spot. Instead, the spot changes whenever needed, as in the *dynamic fitness objectives* developed in Ref. [25]. In the original dynamic fitness objective [25], dynamic objectives are formed initially at the justification and/or propagation frontier for automatic test pattern generation (ATPG). The GA fitness function tries to maximize the number of justification/propagation frontier values justified. The fitness function aims to dynamically advance the justification/propagation of fault-effects beyond the current frontier by placing emphasis on the nodes that are most critical to justify at this time.

With the fitness objectives gradually changing, the GA is adapted to monotonically increase the peak power obtained from a spot to an expanded spot. However, even when the power within the spot increases, the overall power dissipation may not increase since there are nodes outside the optimization area that the GA is not working on. The algorithm for the node-based heuristic is shown in Fig. 5.

Path-based Heuristic

Node-based heuristic focuses on maximizing the switching activity on a set of nodes present in the optimization spot. It is observed, however, that switching activity on a given node heavily relies on the switching activity of its predecessor nodes. If a gate n has two or more immediate predecessor gates, the predecessor p_i with higher MPP has a greater influence on n since p_i has potential for producing more transitions that can propagate to its

```
Compute MPP for each gate;
Pick the gate $g_1$ with the largest MPP;
Spot = gate $g_1$;
While there is improvement
    While there is improvement on switching frequency of Spot
        Call GA to maximize switching on gates included in Spot
    Pick gate $g_2$ with the next largest MPP;
    Expand Spot to include $g_2$;
```

FIGURE 5 Algorithm for node-based heuristic.

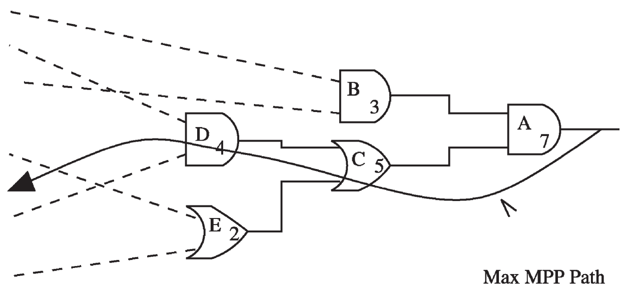

FIGURE 6 Construction of Max MPP path.

```
Compute MPP for each gate;
Compute the max MPP cone $c_1$;
Spot = path $c_1$;
While there is improvement
    While there is improvement on switching frequency of Spot
        Call GA to maximize switching on gates included in Spot
    Compute the next MPP cone $c_2$;
    Expand Spot to include $c_2$;
```

FIGURE 8 Algorithm for cone-based heuristic.

successor gates. Thus, by induction, one would prefer to form a path from node n to a primary input or flip-flop via a set of nodes with greater MPP. We call this path a *maximal MPP path*. A maximal MPP path from n can be constructed simply by depth-first search of a chain of nodes with greatest MPP, starting from node n. Figure 6 shows construction of a segment on the maximal MPP path. The number inside each gate indicates the MPP value for that gate. If we start constructing the MPP path from gate A, the first node to be added is gate C since it has a higher MPP than gate B. The construction continues in a depth-first manner from gate C by adding D to the path, and so on.

The initial genetic spot consists of the maximal MPP path containing the node with the greatest MPP. The spot expands by adding other MPP nodes on the path(s) into the spot. The construction of other MPP paths are done in a similar fashion except that we made a restriction that no two MPP paths should share any common segments. This restriction can be relaxed, however, to allow paths to share segments, with the disadvantage of slower spot expansion since paths share segments. In our implementation, we disallow segment sharing. Figure 7 shows the path-based heuristic.

Cone-based Heuristic

The maximal path construction in the path-based heuristic greedily forms a set of paths based on their MPP values. However, peak switching activity may occur when the nodes on one selected MPP path are not experiencing peak activity. In other words, a tighter bound on peak power may require peak activity along other paths from a given node.

To remedy this problem, the cone-based heuristic is proposed in which all paths which lead up to an MPP node n are considered, and the optimization is focused on producing maximum power inside the MPP cone. It should be noted, however, that when large cones comprised of many nodes are brought in for spot optimization, the optimization becomes more susceptible to the problems which small-spot optimizations did not have to face before; that is, the search can be stuck at a local maximum inside the cone if care were not taken.

Generally, the initial MPP cone includes less than 15% of the entire circuit for most circuits. And each addition of another MPP cone during expansion adds less than 15% additional nodes since cones usually share a portion of common nodes. Figure 8 shows the cone-based heuristic.

```
Compute MPP for each gate;
Compute the max MPP path $p_1$;
Spot = path $p_1$;
While there is improvement
    While there is improvement on switching frequency of Spot
        Call GA to maximize switching on gates included in Spot
    Compute the next MPP path $p_2$;
    Expand Spot to include $p_2$;
```

FIGURE 7 Algorithm for path-based heuristic.

Distance-based Heuristic

In order for a given node to have maximal number of transition times, many flip-flop values in the intermediate state should have transitions. Indeed, results from the first three heuristics suggest that greater Hamming distances between the initial and intermediate states are favored when estimating peak power, where Hamming distance is the number of different flip-flop values between two states. For instance, the states 101101 and 011001 have a Hamming distance of 3 (with the flip-flop values differ in the first, second, and the fourth bit positions). This is an intuitive finding since having a greater Hamming distance between two consecutive states suggests that more activity can occur from the flip-flops in the second time frame, resulting in higher dissipation in the circuit. By the same token, the reverse is true as well: peak power unlikely occurs when the initial and intermediate states are the same (i.e. Hamming distance is 0).

Accounting for only the Hamming distance may be misleading, however, since toggles on different flip-flops in the circuit contribute differently to power consumption. A toggle on a flip-flop may induce a controlling value to some paths that block hyperactive activity from further propagation. Moreover, a state transition from 000000 to 111111 has a maximum Hamming distance, but may not generate a power that is even close to the peak. So instead of merely counting the hamming distance, different weights are placed on each flip-flop based on the favorability of a transition on the flip-flop.

The optimization in this heuristic tries to maximize the following function that computes a weighted-distance

```
Compute MPP for each gate;
Compute the reverse max MPP paths;
Compute the weight on each flip-flop;
Call GA to maximize switching on circuit with fitness function
    set to maximize the weight-distance function;
```

FIGURE 9 Algorithm for distance-based heuristic.

function:

$$\sum_{\text{for all flip-flops}} b_i \times w_i, \qquad (4)$$

where b_i is a Boolean variable and is equal to 1 when the flip-flop values on bit position i differ between the initial and intermediate states, and w_i is the weight for flip-flop i. The weight of a flip-flop i indicates the potential influence on total power dissipation from a toggle on flip-flop i; it is computed as the sum of MPP values on the reverse maximal MPP path starting from flip-flop i. A reverse maximal MPP path is similar to a maximal MPP path except that construction of a reverse MPP path is done in a forward manner, i.e. starting from a flip-flop and moves forward to a primary output or flip-flop.

The advantage of this weighted distance-based heuristic is that a zero-delay simulator is sufficient to evaluate the maximization function, since we only need to know if a transition can occur at each flip-flop. When the optimized state-vector tuple is obtained, a variable-delay simulator is then used to calculate the power consumed by the tuple. Figure 9 shows the distance-based heuristic.

EXPERIMENTAL RESULTS

Peak powers under the new variable delay model were estimated for large ISCAS89 sequential benchmark circuits [26] and two synthesized circuits [27]. All computations were performed on a Sun Ultra-I with 64 MB RAM.

All power estimates are compared against the estimates obtained from 262,000 randomly generated state-vector tuples as well as those obtained using algorithms presented in Refs. [12,14]. All powers are expressed in peak switching frequency per node (PSF), which is the average frequency of peak switching activity of the nodes (ratio of the weighted 0-to-1 and 1-to-0 transitions on all nodes to the total number of capacitive nodes) in the circuit. No zero-width spikes are considered in our approach. In evaluating peak power, we made the assumption that the output capacitance for each gate is equal to the number of fan-outs; however, assigned gate output capacitances can be handled by our optimization technique as well.

As indicated in Refs. [12,14], GA-based technique obtains very tight lower-bounds on peak power for many of the circuits, especially for the smaller circuits. Results for 100 million random state-vector tuples were compared in Ref. [14], and the GA technique outperformed the near-exhaustive search for all eight circuits. Table I shows the peak power estimates for the smaller ISCAS89 sequential circuits. Because simple GA alone can obtain extremely tight lower-bounds for these circuits, improvement achieved by the genetic spot optimization techniques is small. In addition, the difference among the four spot optimization heuristics is very small; thus, only one column is shown. For each circuit, the number of flip-flops is given in parenthesis next to the circuit name. Then, the maximum SF obtainable, maximum power potential switching frequency (MPP SF), is reported; MPP SF reports the maximum SF when every gate achieves the MPP measure. Listed next in the table are the peak power obtained from best of random simulations is given, finally the results of Ref. [14] and our genetic spot optimization. The MPP SF measure is a very loose upper bound, where it

TABLE I Results for small ISCAS89 sequential circuits

				Genetic spot opt.	
Circuit (FF's)	MPP SF	Random PSF	PSF Ref. [14]	PSF	Time (s)
s298 (14)	2.447	1.045	1.095	**1.096**	1.24
s344 (15)	4.417	1.054	1.057	**1.081**	6.02
s382 (21)	2.652	1.045	**1.129**	**1.129**	3.65
s400 (21)	2.693	1.126	**1.158**	**1.158**	3.17
s444 (21)	3.325	1.187	**1.198**	**1.198**	3.70
s526 (21)	2.506	0.786	**0.905**	**0.905**	2.40
s641 (19)	13.431	1.110	1.206	**1.641**	25.70
s713 (19)	32.532	1.135	1.308	**1.833**	16.36
s820 (5)	2.142	1.001	**1.013**	**1.013**	7.59
s832 (5)	2.144	1.000	**1.016**	**1.016**	7.76
s1196 (18)	8.906	0.982	1.104	**1.119**	17.67
s1238 (18)	9.964	0.969	1.042	**1.074**	17.28
s1488 (6)	4.004	1.067	**1.074**	**1.074**	13.18
s1494 (6)	4.020	1.072	**1.078**	**1.078**	13.19
Impr			5.70%	**12.3**%	

All powers expressed in PSF.
Impr: Avg. improvements over the best random estimates.

TABLE II Large sequential circuits statistics

Circuit	FF's	Cap nodes	MPP SF
s1423	74	1243	41.615
s5378	179	4440	6.715
s9234	228	8221	15.024
s13207	669	12031	12.088
s15850	597	14343	51.307
s35932	1728	30317	7.435
s38417	1636	33988	17.967
am2910	87	1998	138.483
mult16	55	1323	27.353

assumes that all gates in the circuit can achieve the maximum switching simultaneously.

Note that there is little difference between our results and [14] for most circuits, indicating that simple genetic technique alone is adequate to compute tight lower bound for peak power in small circuits. Moreover, all of our peak power measures are below the MPP SF measures. Results for s641 and s713 (35 PI's and 19 FF's each) are more significant due to their larger search spaces. Genetic spot optimization obtained up to 61.5% tighter peak power bounds over random and 40% over Ref. [14] in these two circuits. On average, Ref. [14] achieved a 5.70% improvement over best of random, while 12.3% improvement was achieved by the proposed genetic spot optimization. The execution times are consistently small for all circuits.

Results are much more significant for larger circuits. Every large circuit studied has at least 55 flip-flops, with circuits s35932 and s38417 being the largest, each with more than 1600 flip-flops. Table II first shows the circuit characteristics, namely, for each circuit, the number of flip-flops, the total number of capacitive nodes, and the MPP SF. The peak power estimates for large sequential circuits are shown in Table III using various approaches. For each circuit, the MPP SF is first given in parenthesis next to the circuit name. Then, the results of the random approach (best of 262,000 random simulations) are reported. Next, the peak powers obtained from Ref. [14] are shown along with the improvements Ref. [14] has over the random approach. Finally, the peak powers and their corresponding improvements over the random approach using the four proposed heuristics are reported in the table. All powers are expressed in terms of PSF. The highest power estimates are highlighted in bold. The number of GA generations used for Ref. [14] was extended to be four times the original number in order to match the number of random simulations used in the random approach. Unlike the algorithm described in Ref. [14], no seeding of the best of random is used in our genetic spot optimization. Results from the ATG-based approach used in Ref. [10] (based on expanded combinational circuit) are not included, since bounds obtained from Ref. [14] are better and large sequential circuits were not reported in Ref. [10].

For all circuits, genetic spot-optimization heuristics surpassed both random and the original GA-based [14] approaches. Among the four dynamic heuristics, the cone-based technique gave the tightest bounds most consistently. For instance, in circuits s1423 and am2910, the cone-based technique obtained 309 and 53% improvements, respectively, over the random approach. However, occasionally path-based and distance-based heuristics outperformed the cone-based heuristic. On the average, the node-based heuristic performed slightly better than the algorithm proposed in Ref. [14]. Path-based heuristic achieved an average of 61.9% improvement over the random approach, an average 70.7% improvement for cone-based heuristic, and 34.4% for the distance-based heuristic.

It is not surprising that the cone-based heuristic performed the best among the various heuristics in our study. Since the node-based heuristic focuses only on a few MPP nodes, path(s) that may have greater impact on the overall switching activity may be ignored. The path-based heuristic attempts to relieve this problem, but it still cannot guarantee that the MPP path(s) chosen will be the most influential path(s) on the overall power consumption. The cone-based heuristic solves this problem by considering all paths leading to a MPP node. The sizes of cones considered are, on average, less than 15% of the entire circuit size. If cone sizes were significantly larger, the heuristic would be similar to that of the original GA technique, where optimization was performed over

TABLE III Peak power estimates for large sequential circuits

		Ref. [14]		Node-based		Path-based		Cone-based		Distance-based	
Circuit (MPP SF)	Random PSF	PSF	Impr	PSF	Impr	PSF	Impr	PSF	Impr	PSF	Impr
s1423 (41.615)	1.081	1.432	32.5	1.404	29.9	4.064	275.9	**4.427**	**309.5**	1.406	30.1
s5378 (6.715)	0.856	1.213	41.7	1.249	45.9	1.263	47.5	**1.271**	**48.5**	1.261	47.3
s9234 (15.024)	0.693	1.286	85.6	1.229	77.3	1.268	83.0	**1.316**	**89.9**	1.264	82.4
s13207 (12.088)	0.919	1.261	37.2	1.256	36.7	1.262	37.3	1.208	31.4	**1.265**	**37.6**
s15850 (51.307)	0.592	0.693	17.1	0.722	22.0	0.765	29.2	**0.786**	**32.8**	0.734	24.0
s35932 (7.435)	1.396	1.396	0.0	1.410	1.0	**1.417**	**1.5**	1.408	0.9	1.415	1.4
s38417 (17.967)	0.636	0.637	0.2	0.737	15.9	0.690	8.5	**0.756**	**18.9**	0.738	16.0
am2910 (138.483)	4.798	5.688	18.5	5.586	16.4	5.720	19.2	**7.354**	**53.3**	5.691	18.6
mult16 (27.353)	1.861	2.410	29.5	2.601	39.8	**2.892**	**55.4**	2.815	51.3	2.828	52.0
Impr			29.2		31.6		61.9		**70.7**		34.4

All peak powers expressed in peak switching frequency per node (PSF).
Impr: % Improvement over the best random estimate.

TABLE IV Execution times

Ckt	Random	Ref. [14]	Node-based	Path-based	Cone-based	Dist-based
s1423	6.44	7.52	4.28	8.03	8.00	1.08
s5378	44.44	44.76	31.77	42.78	44.98	7.18
s9234	44.00	49.20	39.88	51.78	57.03	10.04
s13207	57.40	67.52	57.78	72.68	88.73	12.58
s15850	52.72	60.68	54.10	68.75	101.77	13.13
s35932	239.68	240.68	268.75	312.35	454.38	55.03
s38417	169.68	186.12	196.47	223.82	365.37	61.72
am2910	34.48	34.52	25.23	29.88	30.60	5.10
mult16	6.88	7.32	7.65	8.92	7.31	1.56

All times expressed in minutes.

the entire circuit all at once. Finally, the distance-based heuristic performed better than the node-based heuristic; however, due to focusing only on toggling of the flip-flops, the search may be blind to propagating hyperactivity across certain paths that can contribute peak power.

Even though the genetic spot-optimization heuristics performed better than the algorithm presented in Ref. [14] for most circuits, there are a few cases where Ref. [14] was able to achieve tighter peak power bounds than one or two genetic spot-optimization heuristics. For instance, in the circuit s1423, the PSF obtained by Ref. [14] was higher than both node-based and distance-based heuristics, etc. In cases such as these, node-based and distance-based heuristics narrowed the search too quickly, resulting in a local maximum, and spot expansion did little to help the search get out of the local maximum.

The execution times for various techniques are shown in Table IV for each circuit. The execution times required for the random simulations are very close to the original GA-based technique [14] since similar numbers of simulations were evaluated. However, Ref. [14] sometimes reaches a local maximum very quickly, requiring only one-third to one-half of the time. The execution times for the distance-based heuristic are consistently lower because zero-delay simulation is sufficient during the optimization process; significantly fewer event evaluations are needed since a gate can switch at most once in a time frame under this delay model. Execution times for the other three heuristics are, in contrast, much higher. The extra times needed are due to much more event evaluations, and execution times are directly proportional to the total number of events. As a result, the genetic cone-optimization heuristic generally required higher execution time, since tighter bounds were obtained.

CONCLUSIONS

Getting tight bounds on peak power requires efficient search algorithms in enormous search spaces. In this paper, four genetic spot-optimization heuristics are proposed to avoid local maximums during the search process by shifting and expanding the optimization spots dynamically. The proposed heuristics have been shown to be very effective in large sequential circuits. When compared to the results of 262,000 random simulations, the genetic spot-optimization heuristics achieved up to an average of 70.7% improvement in large sequential benchmark circuits. Significant improvements were also observed when compared to the results using the algorithm in Ref. [14].

References

[1] Small, C. (1994) “Shrinking devices put the squeeze on system packaging”, *EDN*, 41–46, February.

[2] Najm, F.N. and Zhang, M.Y. (1995) “Extreme delay sensitivity and the worst-case switching activity in VLSI circuits”, *Proc. Des. Aut. Conf.*, 623–627.

[3] Teng, C., Hill, A.M. and Kang, S. (1995) “Estimation of maximum transition counts at internal nodes in CMOS VLSI circuits”, *Proc. Int. Conf. CAD*, 366–370.

[4] Chen, Z., Roy, K. and Chou, T.-L. (1997) “Power sensitivity—a new method to estimate power dissipation considering uncertain specifications of primary inputs”, *Proc. Int. Conf. CAD*, 40–44.

[5] Devadas, S., Keutzer, K. and White, J. (1992) “Estimation of power dissipation in CMOS combinational circuits using boolean function manipulation”, *IEEE Trans. CAD*, 373–383, March.

[6] Kriplani, H. (1992) “Worst case voltage drops in power and ground busses of CMOS VLSI circuits”, Ph.D. Thesis, University of Illinois.

[7] Kriplani, H., Najm, F., Yang, P. and Hajj, I. (1993) “Resolving signal correlations for estimating maximum currents in CMOS combinational circuits”, *Proc. Des. Aut. Conf.*, 384–388.

[8] Manne, S., Pardo, A., Bahar, R.I., Hachtel, G.D., Somenzi, F., Macii, E. and Poncino, M. (1995) “Computing the maximum power cycles of a sequential circuit”, *Proc. Des. Aut. Conf.*, 23–28.

[9] Wang, C., Roy, K. and Chou, T. (1996) “Maximum power estimation for sequential circuits using a test generation based technique”, *Proc. Custom Integrated Circuits Conf.*

[10] Manich, S. and Figueras, J. (1997) “Maximizing the weighted switching activity in combinational CMOS circuits under the variable delay model”, *Proc. Eur. Design Test Conf.*, 597–602.

[11] Wang, C.-Y. and Roy, K. (1997) “COSMOS: a continuous optimization approach for maximum power estimation of CMOS circuits”, *Proc. Int. Conf. CAD*, 52–55.

[12] Hsiao, M.S., Rudnick, E.M. and Patel, J.H. (1997) “K2: an estimator for peak sustainable power of VLSI circuits”, *Proc. Int. Symp. Low Power Electr. Des.*, 178–183.

[13] Hsiao, M.S., Rudnick, E.M. and Patel, J.H. (1997) “Effects of delay model in peak power estimation of VLSI sequential circuits”, *Proc. Int. Conf. CAD*, 45–51.

[14] Hsiao, M.S., Rudnick, E.M. and Patel, J.H. (2000) “Peak power estimation of VLSI circuits: new peak power measures”, *IEEE Trans. VLSI Syst.* **8**(4), 435–439, August.

[15] Ghosh, A., Devadas, S., Keutzer, K. and White, J. (1992) “Estimation of average switching activity in combinational and sequential circuits”, *Proc. Des. Aut. Conf.*, 253–259.

[16] Najm, F.N. (1994) “A survey of power estimation techniques in VLSI circuits”, *IEEE Trans. VLSI Syst.* **2**(4), 446–455.

[17] Chou, T. and Roy, K. (1996) "Accurate power estimation of CMOS sequential circuits", *IEEE Trans. VLSI Syst.* **4**(3), 369–380, September.

[18] Uchino, T., Minami, F., Mitsuhashi, T. and Goto, N. (1995) "Switching activity analysis using boolean approximation method", *Proc. Int. Conf. CAD*, 20–25.

[19] Tsui, C., Monteiro, J., Pedram, M., Despain, A. and Lin, B. (1995) "Power estimation methods for sequential logic circuits", *IEEE Trans. VLSI Syst.* **3**(3), 404–416, September.

[20] Najm, F.N., Goel, S. and Hajj, I.N. (1995) "Power estimation in sequential circuits", *Proc. Des. Aut. Conf.*, 635–640.

[21] Kozhaya, J.N. and Najm, F.N. (1997) "Accurate power estimation for large sequential circuits", *Proc. Int. Conf. CAD*, 488–493.

[22] Benini, L., De Micheli, G., Macii, E., Poncino, M. and Scarsi, R. (1997) "Fast power estimation for deterministic input streams", *Proc. Int. Conf. CAD*, 494–501.

[23] Goldberg, D.E. (1989) Genetic Algorithms in Search, Optimization, and Machine Learning (Addison-Wesley, Reading, MA).

[24] Abramovici, M., Breuer, M.A. and Friedman, A.D. (1990) Digital Systems Testing and Testable Design (Computer Science Press, New York, NY).

[25] Hsiao, M.S., Rudnick, E.M. and Patel, J.H. (1997) "Sequential circuit test generation using dynamic state traversal", Proc. European Design and Test Conf., pp 188–195.

[26] Brglez, F., Bryan, D. and Kozminski, K. (1989) "Combinational profiles of sequential benchmark circuits", *Int. Symp. Circuits Syst.*, 1929–1934.

[27] Hsiao, M.S., Rudnick, E.M. and Patel, J.H. (1996) "Automatic test generation using genetically-engineered distinguishing sequences", *Proc. VLSI Test Symp.*, 216–223.

Michael S. Hsiao received the B.S. in computer engineering with highest honors from the University of Illinois at Urbana-Champaign in 1992, and M.S. and Ph.D in electrical engineering in 1993 and 1997, respectively, from the same university. During his studies, he was recipient of the Digital Equipment Corp. Fellowship and McDonnell Douglas Scholarship. Between 1997 and 2001, Dr Hsiao was an Assistant Professor in the Department of Electrical and Computer Engineering at Rutgers, the State University of New Jersey. Since 2001, he has been an Assosciate Professor in the Bradley Department of Electrical and Computer Engineering at Virginia Tech. Dr Hsiao is a recepient of the National Science Foundation CAREER Award. His current research focuses on low-power design, VLSI testing, design verification, and diagnosis.