Analysis Of The Complexity Of Exact Power Estimation Problems

Abstract

Although many algorithms for power estimation have been proposed to date, no comprehensive results have been presented on the actual complexity of power estimation problems.

In this paper we select a number of relevant problems and show that they belong to classes of high complexity. In particular we show that: a) for a given combinational circuit, the decision version of the peak power estimation problem is NP-complete; b) average power estimation for combinational circuits is #P-complete under the uniform input distribution; c) the decision version of the sequential power estimation problem is PSPACE-complete.

We also address a number of related problems and conclude that even approximation algorithms for some of these problems are NP-hard.

1 Introduction

Extensive research has been done on algorithms for power estimation of CMOS circuits. A number of problems like peak power estimation and average power dissipation in combinational and sequential circuits have been addressed using different methods. Although there exists considerable agreement that efficient (i.e., guaranteed polynomial time) algorithms do not exist for the majority of these problems, no specific analysis of their actual complexity has been published.

This works addresses this lack of specific results, and shows, using a relatively straightforward analysis, that several power estimation problems belong to classes of very high complexity.

The first problem of interest we consider is the problem of *peak power estimation.* We show in section 3 that the decision version of this problem is NP-complete.

In sections 4 and 5, we consider the more general problem of power estimation of combinational and sequential circuits.

Existing algorithms for power estimation can be divided into two categories: simulation based methods and probabilistic methods.

Simulation based methods work by simulating a given input trace. The complexity of simulation based power estimation methods is easily seen to be polynomial on the size of the trace and on the size of the circuit. However, an accurate characterization of the operating conditions of the circuit require the existence of a trace that can be exponentially large on the dimension of the circuit, thereby making the complexity of these methods effectively exponential on the size of the circuit. To avoid this exponential blowup without incurring in a significant loss of precision, approximate methods like sequence compaction [8] and sequence synthesis [7] have been proposed.

The alternative to this approach are probabilistic power estimation methods. They do not explicitly require the existence of a trace and, in principle, can be used to compute the power dissipation of circuits without the need for an exponentially long description of the input stimuli. These methods usually assume simplified models of temporal and spatial correlations. One common simplification is the assumption that the primary inputs are uncorrelated in time and space. Both exact [5] and approximate [2] algorithms, for this problem have been proposed. Methods that use a more accurate modeling of spatial and temporal

correlations have also been presented. One such method [9] models the pairwise spatial correlations of the primary inputs and propagates them through the circuit. A different approach that is able to take into account the full set of temporal and spatial correlations at the inputs has also been proposed [3].

Experimentally, it has been observed that all statistical power estimation methods become computationally very expensive as the size of the circuit grows. In sections 4 and 5 we show that the problem of power estimation in combinational and sequential circuits is indeed inherently complex, and that this complexity cannot be avoided by the use of statistical power estimation methods. In fact, in section 6, we show, using a simple argument, that no efficient approximation algorithms are likely to exist.

2 Definitions

2.1 Combinational and sequential circuits

We will use the following abstract model for combinational and sequential circuits. A combinational circuit will be represented by a directed acyclic graph (DAG). Associated with each node of the graph is a variable, y_i and a representation of a logic function, f_i such that $y_i = f_i$. There is a directed edge e_{ij} from y_i to y_j if f_j depends explicitly on y_i or $\overline{y_i}$. For the purposes of power estimations, we associate with each node a capacity value, c_i, and we assume, in all cases, the zero delay model.

A (synchronous) sequential circuit is composed by a combinational circuit and a set of latches, l_i. A subset of the primary inputs of the combinational circuit is connected to the outputs of the latches, and a subset of the primary outputs is connected to the inputs of the latches.

2.2 Power dissipation in CMOS circuits

Most present day circuit implementations use CMOS logic. For CMOS circuits, the power dissipated by a circuit can be approximated, in first analysis, by

$$P = V_{DD}^2 f \sum_i C_i T_i \tag{1}$$

where V_{DD} is the voltage of the power supply of the circuit, f is the clock frequency at which it operates, C_i the capacitance off node i and T_i the number of transitions in node i.

For the purposes of simplifying the discussion, we will *normalize* expression 1 and assume that $V_{DD}^2 f$ equals 1. The computation of dissipated power therefore reduces to the summation in this expression, of which the significant effort is the computation of T_i, the actual number of transitions in node i for the given operating conditions of the circuit.

2.3 Input distributions

In general, one is interested in the power dissipated by a circuit under specific input sequences, since the actual values present at the inputs of the circuit strongly affects the power dissipation.

Since many interesting input sequences can only be specified by exponentially long traces, one alternative possibility is to specify the values at the inputs through a specification of their statistics, or through some other compact description.

Two interesting examples are the *uniform distribution* and *binary code*. In the first case, we specify that the input bits are uncorrelated in time and space, and, therefore, that all transitions between the input words are equally likely. In the second case, we specify that the input words are presented in a sequence that follows a standard binary code. Note that each of these two conditions would require an exponentially long description if input traces were to be used.

Many other exponentially long input sequences can be specified using appropriate descriptions languages. One such language was proposed in [3], but, in general, we will not be concerned with the particular description language used to specify the input sequence. All the results in the following sections are independent of the particular method used to describe the input sequences, although we believe the demonstrations could be easily adapted to handle most description languages that are powerful enough to describe exponentially long input sequences.

2.4 Complexity classes

In the discussion that follows, we assume the reader is familiar with elementary complexity theory and related basic notions such as polynomial reducibility and completeness. In particular, we assume the reader to be familiar with the classes of decision problems P and NP, which stand for the classes of decision problems that can be solved in polynomial time with deterministic and non-deterministic Turing machines, respectively.

For some of the problems, we will have to refer to higher complexity classes. More specifically we will use the commonly accepted convention of using #P (number-P) to represent an important class of enumeration problems. An enumeration problem Π belongs to #P if there is a nondeterministic algorithm that for every instance of the problem $I \in D_{\Pi}$, the number of distinct guesses that lead to the acceptance of I is exactly the number of solutions of I and each of the solutions can be checked in polynomial time on the length of the instance description [4].

Finally, we will show that some power estimation problems belong to PSPACE, the class of all decision problems that can be solved using polynomial space.

3 Peak power estimation

In this section, we prove that the decision problem associated with problem of peak power estimation of combinational circuits is NP-complete. Given the normalized assumption from the previous section ($V_{DD}^2 f = 1$), the peak power P_{peak} is the maximum amount of capacitance that can switch in any given input transition. In practical terms, this corresponds to the maximum energy dissipated in any transition, divided by the clock period.

Consider the following power estimation problem, PEAK-POWER. This problem, stated as a decision problem is:

INSTANCE : A combinational circuit N, a set of node capacitances $S = c_1, c_2, .., c_m$ and a power value P.

QUESTION : Is there any transition at the inputs, such that the dissipated power of circuit N is equal or greater than P?

Theorem 1 *PEAK-POWER is NP-complete.*

Proof: It is easy to verify that PEAK-POWER is in NP. A nondeterministic algorithm for it need only guess a particular input transition and show, by simulating the circuit in polynomial time, that this particular input transition leads to power dissipation equal or greater than P. Thus the first of the two requirements for NP-completeness is met.

To prove that it is NP-hard, we will transform a known NP-complete problem [1], SAT, to PEAK-POWER.

Consider an instance of SAT, and build the equivalent circuit, where an OR gate corresponds to each clause in the SAT instance, and an AND gate computes the conjunction of all clauses. Let the circuit node

at the output of the AND gate be node O, as shown in figure 1. The capacitance C at the output node O, has the same value as the power P and the capacitance of all OR gates is set to 0.

Now, by running the algorithm for PEAK-POWER in this instance of the problem, it is possible to answer the original SAT problem in the following way:

- If the answer is YES, then there exists at least one transition in node O. This shows that there exists one input combination that sets O to 1, thereby answering positively the original SAT question.
- If the answer is NO, either node O is always at 1 or always at 0. The answer to the SAT problem can be obtained by simulating an arbitrary input combination. If it drives O to 1, the answer to SAT is YES. If it drives O to 0, the answer to SAT is NO.

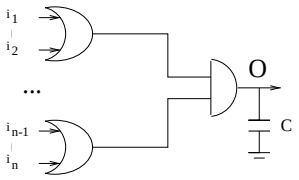

Figure 1: Transformation from SAT to PEAK-POWER.

Therefore, SAT $\propto$ PEAK-POWER, and the theorem is proved.

□

4 Average power estimation

We consider now the general problem of average power estimation of combinational circuits under specific input conditions. Clearly, when a polynomial size input trace that matches the desired conditions is available, any simulation based method represents a polynomial time algorithm for the problem. The difficulty lies in the fact that, for many operating conditions of interest, the trace description is, in itself, exponential on the size of the circuit description.

In this section, we will analyze two power estimation problems: average power dissipated when binary code is presented at the inputs of the circuit and average power dissipated under the uniform distribution of inputs.

4.1 Average Power For Binary Input Code

In this section we will consider a specific input sequence, although the result can be easily generalized for many other conditions. In particular, we will consider the input sequence that is defined by a binary code at the inputs. This problem, BIN-POWER, can be formalized in the following way:

INSTANCE : A combinational circuit N and a set of node capacitances $S = c_1, c_2, \ldots, c_m$.
QUESTION : What is the average power dissipated in the circuit when binary code is presented at the inputs?

Given the normalizing assumption from section 2, the average power can be computed by counting the number of transitions in the nodes in the circuit. We will then consider the following counting problem, #BIN-TRANSITIONS:

INSTANCE : A combinational circuit N and a node O.
QUESTION : What is the number of transitions in node O when binary code is presented at the inputs ?

Theorem 2 *#BIN-TRANSITIONS is #P-complete.*

Proof: Clearly, #BIN-TRANSITIONS is in #P, since proving that a particular input transition causes a transition in node O can be verified in polynomial time. To prove that #BIN-TRANSITIONS is #P-hard, we will transform #SAT, a known #P-hard [10] problem to #BIN-TRANSITIONS. For that, we build the circuit of figure 2. In this circuit, we added an extra input z, that forces the output of the AND gate to become 0.

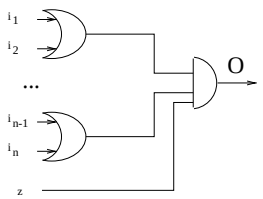

Figure 2: Transformation from #SAT to #BIN-TRANSITIONS.

This extra input helps in relating the number of transitions to the number of satisfying assignments.

From an analysis of this circuit, it is easy to see that the number of transitions in node O when binary code is presented at the inputs (with variable z seen as the least significant bit) is exactly twice the number of satisfying assignments to the original SAT instance. This proves that #SAT can be transformed to #BIN-TRANSITIONS and therefore that #BIN-TRANSITIONS is #P-complete.
□

It should be clear that #BIN-TRANSITIONS is just a restricted version of BIN-POWER, where all capacitances of the nodes are zero except for node O. Therefore, the following result follows immediately:

Corollary 1 *BIN-POWER is #P-complete.*

4.2 Average Power For Uniform Input Distribution

An assumption commonly used by statistical power estimation methods is that the input bits are uncorrelated in time and space. This is effectively equivalent to the assumption that all transitions between input words are equiprobable, and, from a practical point of view, is supposed to simplify the power estimation process.

In this section, we show that this problem is, in fact, no easier than the problem addressed in the previous section. Consider, then, the UNIF-POWER problem:

INSTANCE : A combinational circuit N and a set of node capacitances $S = c_1, c_2, \ldots, c_m$.
QUESTION : What is the average power dissipated in the circuit for a sequence of inputs where all possible transitions appear an equal number of times?

Before we address the complexity of this problem, we prove that a restricted version of it, formulated as a decision problem, is NP-complete. This problem, UNIF-POWER-GT0, stated as a decision problem is:

INSTANCE : A combinational circuit N and a set of node capacitances $S = c_1, c_2, \ldots, c_m$.
QUESTION : Does circuit N dissipate power for a sequence of inputs where all possible transitions appear an equal number of times?

Theorem 3 *UNIF-POWER-GT0 is NP-complete.*

Proof: It is easy to see that UNIF-POWER-GT0 is in NP. A nondeterministic algorithm needs only guess a particular input transition and show, by simulating the circuit in polynomial time, that this particular input transition leads to power dissipation in the circuit. To prove that it is NP-complete, we again transform SAT to UNIF-POWER-GT0. Given an instance of SAT, we build a circuit as in section 3. Now, by running the algorithm for UNIF-POWER-GT0 in this instance, we answer the SAT question following a procedure similar to the one used for PEAK-POWER:

- If the circuit dissipates power, the answer to the original SAT question is YES, since node O went to 0 and to 1 at least once.
- If the circuit does not dissipate power, either node O is always at 1 or always at 0. Again, the answer to the SAT problem can be obtained by simulating an arbitrary input combination. If it drives O to 1, the answer to SAT is YES. If it drives O to 0, the answer to SAT is NO.

Therefore, SAT $\propto$ UNIF-POWER-GT0, and the theorem is proved.
□

We will now prove that UNIF-POWER is also #P-complete. To achieve this, we first need to define a code that has a distribution of input transitions as desired. We will call this code *min-unif-code*. It is straightforward to show that, for a circuit with n inputs, there exists a code with $2^n(2^n-1)+1$ words that has one and exactly one transition between each possible input word[1]. After defining this code, the proof proceeds in a similar way, although the result is slightly more involved.

Again, UNIF-POWER can be solved by counting the transitions in the nodes. This leads us to define the following counting problem, #UNIF-TRANSITIONS:

INSTANCE : A combinational circuit N and a node O.
QUESTION : What is the number of transitions in node O when *min-unif-code* is presented at the inputs ?

Theorem 4 *#UNIF-TRANSITIONS is #P-complete.*

Proof: Again, it is clear that #UNIF-TRANSITIONS is in #P. To prove that #UNIF-TRANSITIONS is #P-hard, we will again transform #SAT using the circuit of figure 2. A relatively straightforward combinatorial analysis shows that, if *min-unif-code* is presented at the inputs, the solution to #SAT, m, is related to u, the solution to #UNIF-TRANSITIONS by:

$$m = \frac{2^n - \sqrt{2^{2n} - 2u}}{2} \tag{2}$$

Therefore, a solution to #SAT can be obtained in polynomial time from a solution to #UNIF-TRANSITIONS, showing that #UNIF-TRANSITIONS is #P-complete.
□

It is interesting to note that theorem 2 could also be easily proved by finding a *parsimonious transformation*[2] from the search version of SAT to the search version of PEAK-POWER. However, such a *parsimonious transformation* is not so easily obtainable for the problem of UNIF-TRANSITIONS, so a direct reduction from #SAT to #UNIF-TRANSITIONS represents a simpler demonstration of the main result.

[1] For space reasons, we omit the description on how such a code can be obtained, but, as an example, for $n = 2$, such a code could be 00,01,10,11,10,01,11,01,00,10,00,11,00.

[2] A transformation from a search problem Π to a search problem Π' is *parsimonious* if it preserves the number of solutions.

5 Power estimation for sequential circuits

In this section, we will study the problem of power estimation in sequential circuits. More specifically, we will study the complexity of a decision version of the problem, SEQ-POWER-GT0:

INSTANCE : A sequential circuit N and a set of node capacitances $S = c_1, c_2, \ldots, c_m$.
QUESTION : Is there a sequence of inputs that makes circuit N dissipate power ?

From a practical point of view, the added difficulty of this problem lies in the sequential elements present, that create temporal correlations that are expensive to compute. A number of algorithms for computing the exact correlations have been proposed, but all the exact algorithms have been observed to become rapidly very inefficient when large circuits are under consideration.

We will show that, indeed, the problem is intrinsically difficult. In particular, we have the following result:

Theorem 5 *SEQ-POWER-GT0 is PSPACE-complete.*

Proof: It is straightforward to show that SEQ-POWER-GT0 is in PSPACE, since all possible transitions of the state transition graph of a sequential circuit can be exercised using only a polynomial amount of space. To prove that it is PSPACE-hard, we will transform the *output of finite memory programs* (OUT-FIN-PROG) to SEQ-POWER. OUT-FIN-PROG[3], a known PSPACE-complete problem [6] [4] is defined as follows:

INSTANCE: Finite set X of variables, finite alphabet Σ, a program P, defined by a sequence $I_1, I_2, \ldots, I_m$ of instructions of the form *read* x_i, *write* v_j, *if* $v_j = v_k$ goto I_l, *accept* or *halt* , where each $x_i \in X$, each $v_j \in X \cup \Sigma \cup \{\$\}$ and I_m is either *halt* or *accept.*
QUESTION: Is there a string $w \in \Sigma^*$ such that program P generates any output ?

We will consider only finite programs where the input alphabet Σ is $\{0, 1\}$, since OUT-FIN-PROG remains PSPACE-complete even in that case [6]. To transform OUT-FIN-PROG into SEQ-POWER-GT0, we observe that the behavior of a finite program P can be simulated by a sequential circuit with $\log_2(|X|(|\Sigma| + 1)) + \log_2 m$ latches. It should also be obvious that the combinational logic can be easily constructed in polynomial time from the structure of the program.

Now, to generate the output, we will create a node O in the sequential circuit, that will have value 1 whenever a *write* instruction is executed an (and, therefore, output is being generated), and that will have value 0 when no output is being generated.

Given this sequential circuit, we now define C_i to be equal to 0 for all nodes except node O. Now, by running the algorithm for SEQ-POWER-GT0 in this sequential circuit, we can answer the question for OUT-FIN-PROG in the following way:

- If the answer is YES, then the answer to the original OUT-FIN-PROG is also YES.
- If the answer is NO, simulate the circuit with an arbitrary input sequence. If node O is always at 1, then the answer is YES. Otherwise, the answer is NO.

Therefore, OUT-FIN-PROG $\propto$ SEQ-POWER-GT0, and the theorem is proved.
□

[3]This problem is a special case of INEQUIVALENCE OF FINITE MEMORY PROGRAMS problem, where program P_2 is a fixed program with no write instructions and hence no output.

6 Approximation algorithms and related problems

From the transformations presented in sections 3 to 5, it should be clear that approximation algorithms for some of the problems addressed are unlikely to exist.

For space reasons, we will not address extensively this question. However, we note that the reductions presented for PEAK-POWER and UNIF-POWER-GT0 immediately prove that efficient approximation algorithms for the related estimation problems[4] do not exist unless $P = NP$.

In fact, for PEAK-POWER and UNIF-POWER, any approximation algorithm that generates a value P' that is in the interval $[P(1-\epsilon), P(1+\epsilon)]$, (where P is the exact value of dissipated power and $0 < \epsilon < 1$) of the desired answer would enable us to solve SAT.

It should now be clear that similar results could also be proved for other versions of power estimation problems.

We should also point out that the hardness of the power estimation problems we addressed is **not** caused by the artificial concentration of node capacitances in a single node, used in the proofs. It is easy to show that even if all the node capacitances have equal values, the complexity of the problems addressed remains the same.

7 Conclusions

We addressed the complexity of a number of power estimation problems and proved that they belong to classes of high complexity.

In particular, we proved, using a straightforward transformation, that the decision version of the peak power estimation problem for combinational circuits is NP-complete.

For power estimation problems, we have shown that exact computation of the average power dissipated by combinational circuits under the uniform input distribution is #P-complete. The same is true for power estimation of combinational circuits when the input sequence is binary code. Furthermore, we believe that this complexity analysis is easily extensible to a large number of operating conditions that involve exponentially large input sequences. This result is remarkably strong, since #P-complete problems are known to be no simpler than any problem in the polynomial hierarchy.

Finally, we proved that the decision problem associated with power estimation in sequential circuits is PSPACE-complete, a strong but not surprising result given known related results in sequential circuit analysis and synthesis.

Based on these results, we also argued briefly that approximation algorithms for the majority of interesting problems in power estimation do not exist unless $P = NP$.

To the best knowledge of the authors, these results represent the first comprehensive overview of the complexity of the analysis of dissipated power in combinational and sequential circuits.

References

[1] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[2] J. Costa, J. Monteiro, and S. Devadas. Switching Activity Estimation using Limited Depth Reconvergent Path Analysis. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 184–189, August 1997.

[4] The related estimation problems ask for the actual value of the peak or average power.

[3] A. T. Freitas, A. L. Oliveira, J. C. Monteiro, and H. C. Neto. Exact power estimation using word level transition probabilities. In *Proceedings of the Ninth International Workshop on Power and Timing Modelling, Optimization and Simulation*, pages 355–364, Kos Island, Greece, October 1999.

[4] Michael R. Garey and David S. Johnson. *Computers and Intractability*. Freeman, New York, 1979.

[5] A. Ghosh, S. Devadas, K. Keutzer, and J. White. Estimation of Average Switching Activity in Combinational and Sequential Circuits. In *Proceedings of the* 29^{th} *Design Automation Conference*, pages 253–259, June 1992.

[6] Neil D. Jones and Steven S. Muchnick. Even simple programs are hard to analyze. *Journal of the ACM*, 24(2):338–350, April 1977.

[7] D. Marculescu, R. Marculescu, and M. Pedram. Stochastic sequential machine synthesis targeting constrained sequence generation. In *33rd Design Automation Conference (DAC'96)*, pages 696–701, New York, June 1996. Association for Computing Machinery.

[8] D. Marculescu, R. Marculescu, and M. Pedram. Hierarchical sequence compaction for power estimation. In *33rd Design Automation Conference (DAC'97)*, pages 570–575, New York, June 1997. Association for Computing Machinery.

[9] R. Marculescu, D. Marculescu, and M. Pedram. Efficient Power Estimation for Highly Correlated Input Streams. In *Proceedings of the* 32^{nd} *Design Automation Conference*, pages 628–634, June 1995.

[10] C. Papadimitriou. *Computational Complexity*. Addison Wesley, 1994.