ALPS: A Peak Power Estimation Tool for Sequential Circuits

F. Corno, M. Rebaudengo, M. Sonza Reorda, M. Violante

Politecnico di Torino
Dipartimento di Automatica e Informatica
Torino, Italy
http://www.cad.polito.it/

Abstract

Tools for evaluating the worst-case peak power consumption of sequential circuits are highly useful to designers of low-power circuits. Previously proposed methods search for the initial state and the couple of vectors with maximum consumption, without fully considering the reachability of the initial state. This paper shows that this approach can lead to a significant underestimation of the maximum peak power consumption, and proposes a new algorithm that overcomes this drawback. Experimental results show that for many circuits the algorithm is able to provide better results than those known up to now, while an approximate version is able to deal even with the largest benchmark circuits.

1. Introduction

In the last years, design for low-power consumption has became a widespread design paradigm, due to the continuing increase in the chip density. Excessive power dissipation can cause performance degradation, run-time errors, or device destruction due to overheating. Large instantaneous power dissipation may cause local hot spots, and the failure rate for components roughly doubles for every 10º C increase in operating temperature. With the increasing demand for high reliability in modern VLSI designs, accurate estimation of the maximum power dissipation and of the maximum instantaneous current during the design process is becoming essential.

The maximum power is averaged over the clock cycle, resulting in the cycle-based maximum power value. Throughout this paper the unit of power is the energy per cycle and its maximum value will simply be referred to as *peak power*. Our paper deals with sequential circuits, where power consumption depends on the initial state S and on the sequence of input values applied starting from S. The focus of our paper is to find the initial state S and the couple of vectors (V_1, V_2) leading to the state transition with the peak power consumption.

A number of techniques have been developed to solve this problem. In [1] maximum power cycles were computed using symbolic transition counts. The largest circuit reported has less than 500 gates, and handling larger circuits is impractical. Automatic test pattern generation (ATPG)-based techniques [2] attempt to generate an input vector pair that produces the largest switched capacitance in the circuit for gates with the greatest numbers of fan outs.

The authors of [3] and [4] proposed a technique based on Genetic Algorithms: in their method the switching activity is used as an indicator of the quality of each individual in the genetic population. This technique can deal with large circuits, but is strongly approximated since the genetic algorithm ignores the reachability of the initial state S. If the algorithm finds an optimal 3-tuple (S, V_1, V_2) with an unreachable initial state S, a set of states S_{sim} is formed by selecting reachable states that are *similar* (as far as the Hamming distance is considered) to S. The couple of vectors (V_1, V_2) is simulated from every state in S_{sim} and the maximum power obtained is taken as the peak power. This algorithm is based on the assumption that, starting from two states with a low Hamming distance and applying a given sequence to the inputs, the power consumption is similar. In the following, we will analyze the behavior of the algorithm, and experimentally demonstrate that this assumption can underestimate the peak power when the number of reachable states is a minimal fraction of all the possible states. We will then propose a new approach, based on a Genetic Algorithm, which aims at overcoming this drawback, and exploits the knowledge about the reachable state set. Two versions of the method are proposed: one,

0-7695-0104-4/99 $10.00

based on symbolic pre-computation of the reachable state set, is suitable for small- and medium-sized circuits; the other, based on approximate dynamic exploration of the same set, is shown to be able to deal with large circuits with a good accuracy.

The remainder of the paper is organized as follows. Section 2 presents a brief definition of the problem, while Section 3 describes the optimization algorithm. Experimental results on benchmark circuits are presented and discussed in Section 4. Section 5 draws some conclusions.

2. Problem description

Heat dissipation in CMOS circuits can be effectively estimated by means of the switching activity. According to [5], dynamic power consumption coming from the application of the *i*-th vector is proportional to the circuit *weighted switching activity* (wsa^i), defined as:

$$wsa^i = \sum_g C_g NT_g^i \qquad (1)$$

where C_g is the physical capacitance associated to gate g and NT_g^i is the number of times the output of gate g changes due to the application of the i-th vector.

Different methods have been proposed to evaluate wsa^i [6]. In this paper we adopt the zero-delay model, where no delay is associated with any gate, and use a logic simulator to evaluate wsa^i.

The problem addressed in this paper is to find the 3-tuple (S, V_1, V_2) which maximizes the weighted switching activity.

3. The ALPS Algorithm

A Genetic Algorithm has been adopted to solve the above introduced search problem.

The *individual* is encoded as a unique binary string containing the state S and the input vector couple. The *cross-over* operator is a uniform cross-over: bits from the 2 parents are swapped with probability 0.5 at each string position when generating the two offspring. A *mutation* operator randomly changes a limited number of bits in the binary string. The *fitness function* evaluates each individual in terms of attained wsa^i.

The crossover and mutation operators may generate a state S^* not reachable by the circuit starting from the reset state. In this case we find the most similar reachable state S, i.e., the state S such that:

- S is reachable
- the Hamming distance between S^* and S is minimum.

For this purpose, the algorithm stores information about the reachable states. We consider two different approaches:

- an exact method suitable for small circuits with a limited number of flip-flops
- an approximate method applicable for larger circuits with a high number of flip-flops.

In the exact method the complete set of reachable states is stored. This set is obtained by means of a symbolic traversal procedure. The computation of the reachable state S from S^* is performed by finding the nearest satisfying assignment in the BDD of the reachable states.

In the approximated method, an initial set of reachable states is pre-computed by simulating a random sequence of vectors, and stored in a hash table. This initial set is then dynamically updated by adding to it possible new states reached through the vectors evaluated by the fitness function. The hash function has been defined so that it heuristically allows the retrieval of the reachable state having minimal Hamming distance from S^*.

It should be noted that our algorithm can be easily extended to make it able to solve similar problems: as an example, the cost function can be modified to consider the peak n-cycle power [3] to find the best (n+2)-tuple (S_1, V_1, ..., V_{n+1}) that maximizes the average power of a contiguous sequence of n clock cycles.

4. Experimental Results

A preliminary tool named ALPS (*Analyzer of Low Power Systems*) has been written, which implements the above introduced procedures. The tool amounts to about 500 lines of ANSI C code. In the following, results are reported to demonstrate the impact of the initial state on the peak power. Moreover, results assessing the effectiveness of ALPS are reported.

A subset of the ISCAS89 sequential circuits (the ones the symbolic traversal procedure is able to manage) has been used to evaluate the performance of ALPS: all the experiments have been performed on a Sun SparcStation 5/110 with 64 MB RAM. Moreover, four of the larger benchmarks (s1423, s5378, s5378 and s35932) have been used to evaluate the approximated method.

To compare ALPS with a state-of-the art tool, we re-implemented the algorithm proposed in [4], adopting a zero-delay model to evaluate the switching activity of the gates. When considering the circuits published in [4], the results we obtained are almost equivalent, thus proving the correctness of the re-

implementation. We can therefore compare ALPS with our re-implemented algorithm on a larger set of circuits than that published in [4].

Table 1 reports the results obtained with the re-implementation of the algorithm proposed in [4]. Power is expressed in terms of Peak Switching Frequency per node (*PSF*), which is the normalized *wsa'* (i.e., the weighted number of transitions on all nodes over the total number of capacitive nodes) as defined in [3]. The column PSF_U reports the maximum PSF without considering the reachability of the initial state; while the column PSF_H reports the PSF considering only reachable states. The column ΔPP represents the relative PSF loss between PSF_U and PSF_H. The column *FF* reports the number of flip-flops in the circuit. The column *H* reports the minimum Hamming distance from the ideal state S^* and the closest reachable state S; *Dim* reports the number of reachable states having Hamming distance from S^* equal to *H*. The column *R* reports the ratio between *H* and *FF*, i.e., the percentage of the initial state bits modified to obtain the reachable state. Finally, the CPU time requirements are reported.

By observing the two columns ΔPP and *R*, one can observe that several circuits exist where the peak power strongly depends on the reachability of the initial state. Where a significant loss in PSF is found, the corresponding value of R is high, i.e., there is a strong difference between the ideal initial state and the selected reachable one.

A comparison between the results provided by ALPS and those produced by our re-implementation of the algorithm in [4] is reported in Table 2.

In Table 2, the column Δ measures the improvement in Peak Switching Frequency obtained by ALPS over the algorithm of [4]. Table 2 shows that ALPS is able to effectively improve the peak power computation for circuits where the power consumption of a sequence strongly depends on the initial state. For such circuits (s298, s386, s499, s510, s526, s635, s641 and s713) high values of *R* (from Table 1) have been computed, showing that the initial state obtained by [4] is a heavy approximation of the ideal but unreachable state. Conversely, ALPS is able to find the optimal state within the reachable states set, thus improving the peak power consumption.

On the other hand, circuits exist where the impact of the initial state on the peak power consumption is limited. For such circuits (s344, s349, s382, s400, s444, s1196, s1238, and s1494) an improvement less than 1.2% has been observed, although high values of R have been computed. Finally, ALPS and [4] give the same results for the remaining circuits.

When larger circuits are considered, ALPS performs 35% better on the average with respect to the re-implementation of the algorithm in [4], while requiring a larger CPU time due to the hash table overhead.

5. Conclusions

We described a new algorithm to compute the peak power consumption of a sequential circuit. The algorithm exploits the knowledge about the reachable state set, and is thus able to produce better results than those provided by previously proposed methods, especially when dealing with circuits where the peak power strongly depends on the circuit initial state.

The obtained results are a lower bound of the peak power consumption, since the currently adopted delay model lacks accuracy. A more accurate delay model (e.g., the unit-delay one) will be adopted in the future without modifying the overall proposed algorithm.

6. Acknowledgments

The authors wish to thank Valere Speranza for his significant contribution and for performing most of the experiments this paper is based on.

7. References

[1] S. Manne, A. Pardo, R. I. Bahar, G. D. Hachtel, F. Somenzi, E. Macii, M. Poncino, "Computing the maximum power cycles of a sequential circuit", Proc. of IEEE/ACM Design Automation Conference, pp. 23-28, 1995

[2] C.-Y. Wang, K. Roy, "Maximum Current Estimation in CMOS Circuits Using Deterministic and Statistical Techniques", IEEE Transactions on VLSI Systems, pp. 134-140, March 1998

[3] M.S. Hsiao, E.M. Rudnick, J. Patel, "K2: An Estimator for Peak Sustainable Power of VLSI Circuits", Proc. of International Symposium on Low Power Electronics and Design, pp. 178-183, 1997

[4] M.S. Hsiao, E.M. Rudnick, J. Patel, "Effects of Delay Models on Peak Power Estimation of VLSI Sequential Circuits", Proc. of IEEE/ACM International Conference on Computer-Aided Design, pp. 45-51, 1997

[5] A. Shen, A. Ghosh, S. Devadas, K. Keutzer, "On Average Power Dissipation and Random Pattern Testability of CMOS Combinational Logic Networks", Proc. of IEEE/ACM International Conference on Computer-Aided Design, pp. 402-407, 1992

[6] A. Ghosh, S. Devadas, K. Kuetzer, J. White, "Estimation of average switching activity in combinational and sequential circuits", Proc. of IEEE/ACM Design Automation Conference, pp. 253-259, 1992

	FF	PSF_U	PSF_H	*ΔPP[%]*	*Dim*	*H*	*R [%]*	*CPU [s]*
s208	8	0.726	0.726	0.0	1	0	0	12.2
s298	14	0.848	0.686	19.1	1	5	35.7	14.6
s344	15	0.797	0.732	8.2	3	2	13.3	18.9
s349	15	0.789	0.709	10.1	1	2	13.3	18.1
s382	21	0.799	0.652	18.4	3	6	28.6	18.7
s386	6	0.772	0.547	29.1	1	3	50.0	17.2
s400	21	0.790	0.640	19.0	3	6	28.6	18.8
s420	16	0.728	0.728	0.0	1	0	0	23.5
s444	21	0.752	0.662	12.0	3	8	38.1	20.2
s499	22	0.595	0.201	66.2	21	20	90.9	19.0
s510	6	0.584	0.529	9.4	2	1	16.7	23.3
s526	21	0.790	0.614	22.3	3	8	38.1	23.5
s635	32	0.745	0.086	88.5	1	29	90.6	28.5
s641	19	0.826	0.704	14.8	1	6	31.6	59.4
s713	19	0.806	0.690	14.4	1	6	31.6	50.4
s820	5	0.827	0.827	0.0	1	0	0	34.4
s832	5	0.816	0.816	0.0	1	0	0	33.9
s1196	18	0.613	0.605	1.3	4	6	33.3	51.3
s1238	18	0.613	0.604	1.5	4	6	33.3	56.0
s1488	6	0.679	0.679	0.0	1	0	0	65.4
s1494	6	0.672	0.633	5.8	3	1	16.7	65.3

Table 1: Experimental results reimplementing the algorithm of [4].

			[4]		ALPS		
Method	*Circ.*	*FF*	PSF_H	*CPU [s]*	*PSF*	*CPU [s]*	*Δ [%]*
Exact	s208	8	0.726	12.2	0.726	12.3	0.0
	s298	14	0.686	14.6	0.716	17.6	4.4
	s344	15	0.732	18.9	0.739	32.3	1.0
	s349	15	0.709	18.1	0.709	22.9	0.0
	s382	21	0.652	18.7	0.652	33.4	0.0
	s386	6	0.547	17.2	0.683	16.7	24.9
	s400	21	0.640	18.8	0.640	33.7	0.0
	s420	16	0.728	23.5	0.728	24.2	0.0
	s444	21	0.662	20.2	0.662	36.6	0.0
	s499	22	0.201	19.0	0.214	21.2	6.5
	s510	6	0.529	23.3	0.554	22.8	4.7
	s526	21	0.614	23.5	0.648	37.5	5.5
	s635	32	0.086	28.5	0.116	25.4	34.9
	s641	19	0.704	59.4	0.732	50.1	4.0
	s713	19	0.690	50.4	0.709	50.9	2.8
	s820	5	0.827	34.4	0.827	34.0	0.0
	s832	5	0.816	33.9	0.823	33.7	1.2
	s1196	18	0.605	51.3	0.607	99.6	0.3
	s1238	18	0.604	56.0	0.608	105.3	0.7
	s1488	6	0.679	65.4	0.679	66.7	0.0
	s1494	6	0.633	65.3	0.650	65.9	1.0
Approximated	s1423	74	0.529	233.3	0.590	693.0	11.5
	s5378	179	0.356	395.3	0.520	2,303.6	46.1
	s9234	228	0.092	2,405.9	0.133	2,470.1	44.6
	s35932	1,728	0.471	3,087.2	0.638	8,407.5	35.5

Table 2: Comparing the two algorithms.