A Hybrid SA-GA Method for Finding the Maximum Number of Switching Gates in a Combinational Circuit

Ichiro R. Ruiz Obregon[1] and Alberto Palacios Pawlovsky[2]
[1]Graduate School of Biomedical Engineering, Toin University of Yokohama
225-8502 Yokohama, Aoba-ku, Kurogane-cho 1614, Japan
[2]Department of Electronics and Information Engineering, Toin University of Yokohama,
225-8502 Yokohama, Aoba-ku, Kurogane-cho 1614, Japan
E-mail : [1]ichiro@edu.toin.ac.jp, [2]pawlovsky@cc.toin.ac.jp

Abstract: This work shows the results obtained with the combined application of simulating annealing (SA) and genetic algorithms (GA) in the generation of inputs pairs that cause the maximum number of switching gates in combinational circuits. We found that a combination of SA and GA produces better results than those obtained using only one of them.

1. Introduction

Modern VLSI designs use integration technologies that make them very complex and with billions of transistors. This complexity now allows us to put a complete system on a chip and fostered the development of very complex portable devices that to handle their multiple functions require high performance designs. High performance usually implies the use of high frequency clocks and brings the problems of increasing heat and power consumption. This also comes together with the fact that each generation of mobile devices reduces the space allowed to batteries. Moreover, battery technology has not been able to keep pace with this downsizing trend making power reduction a pervasive problem in any new development. We can estimate the power consumption of a design at different levels. At the system level, high-level descriptions of the circuit and abstractions of capacitance and switching are used to estimate power [1]. At the functional level, abstraction of functional blocks (muxes, ALUs, registers) provide the models to get power estimates [2]. At the gate level, we can use BDD (Binary Decision Diagrams) to get some estimates [3] to address power related problems. At the transistor level, we can use model correlations to predict worst case power consumption [4]. We can break these techniques in non-simulative, simulative, and hybrid techniques. At the gate level we can divide these techniques in probabilistic and sampling-based (deterministic) techniques [5], [6], [7]. This paper is about an heuristic of the last type. There are several methods to find an input pair that causes the maximum number of switching gates in a circuit. The simulated annealing method of [9] improved the results of the iterative method of [8], but has the bottleneck of its running time for circuits with many inputs. Genetic algorithms (GA) have been used to determine the lower bounds on the maximum power [10]. Recently, a method using GA showed that the running time problem could be reduced, but at the expense of results with a reduced number of switching gates [11]. In the following section, we show our SA algorithm and the modifications we made to use it with the crossover and mutation operations of a GA. In section III, we show the experimental results we obtained with our approach and their comparison with other published results. In the last section, we give some conclusions and outline some topics that need further consideration.

2. Hybrid SA-GA Method

A brief scheme of the SA algorithm we use in our work is shown in Fig. 1. In it we can see that a new pair (p_i) is generated and compared to a current one p_{i-1}. If the difference in the number of switching gates ΔG ($G_i - G_{i-1}$) is greater than zero (i.e., there is an increase in the number of switching gates) we use the new pair to generate the next one p_{i+1}. This process is repeated a number of times (N_{in}) dictated by the inner loop of SA. In the case that ΔG does not show an increase in the number of gates, we do not discard immediately the new pair. Instead, we use the rejection algorithm (known also as the Metropolis algorithm[12]), and determine stochastically if we use the "bad" pair or not. If we reject the pair, the previous one, p_{i-1} will be used to generate a new one p_{i+1}. Otherwise we use p_i to generate a new one. The rejection or acceptance of a "bad" pair is subject to the evaluation of equation (1).

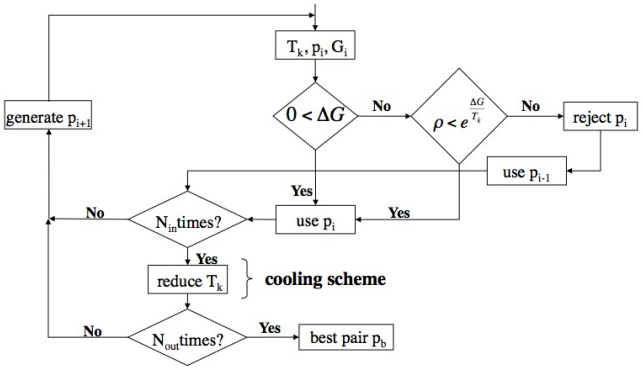

Figure 1. Flow diagram of the SA algorithm used in this work.

$$\rho < e^{\frac{\Delta G}{T_k}} \tag{1}$$

Where T_k is the parameter know as temperature in SA. This parameter is usually set to a big value, so we accept a lot of bad pairs at the beginning. This parameter is decreased every time we left the inner loop of SA so the chances of accepting "bad" pairs decrease with time. The decreasing of parameter T_k is done following a scheme known as the cooling scheme of SA. To accomplish this, we have a lot of different approaches. In this work, we use the simple geometric

cooling scheme given by equation (2).

$$T_{k+1} = T_k \times \alpha \tag{2}$$

Where α takes a positive value lower than 1 (usually it takes values between $0.7 \sim 0.9$). The number of times the temperature is reduced in SA is controlled by the outer loop of SA (N_{out}). In this paper, the number of solutions (pairs in our case) searched by SA is given by $N_{in} \times N_{out}$. However, in some cases it is tied to the elements that control the search space (in our case the number of inputs of a circuit). In the SA of [9], the number of inner loop repetitions is given by N_{in} multiplied by the number of circuit inputs. This approach increases the number of solutions searched, but also increases the running time needed. Once we complete the number of cycles of the outer loop, we leave our SA given as output of the best pair we were able to find (the one that causes the maximum number of switching gates among all pairs searched). In this work, we have adapted the crossover and mutation operations used in a GA to generate a new pair p_{i+1} in the inner loop of SA. They are explained in what follows.

2.1 Input Pair Representation

To handle input pairs with the crossover and mutation operations of a GA we need to represent them as chromosomes (an array of elements (genes in GA)). We use the representation of [9] and [11]. Where a pair of binary inputs, **00**, **01**, **10**, and **11** is replaced by 0, 2, 3, and 1, respectively.

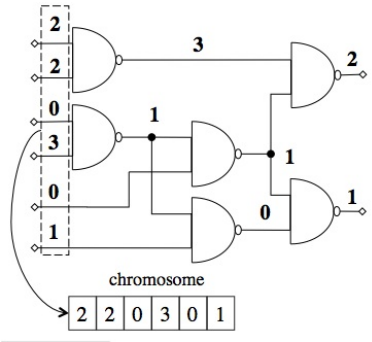

Figure 2. Four-valued signal change representation.

We use the values corresponding to the inputs of the circuit as a chromosome(Fig. 2).

2.2 Generation of Input Pairs in a SA

In a SA algorithm we work with one solution at a time, and a new solution (input pair) is usually generated from a current one. The SA of [9] uses two methods for generating input pairs. The best one determines randomly the number of pins to change and also those that will change. Fig. 3 shows this method when four inputs (the highlighted (underlined) ones of Fig. 3(a)) are selected to change. In Fig. 3(b) we can see that with the new pair we increase the number of switching gates from two to three (the ones with a gray circle inside the gate). In our work we replace this way of generating a new pair from a current one applying the operations of crossover and mutation used in a GA.

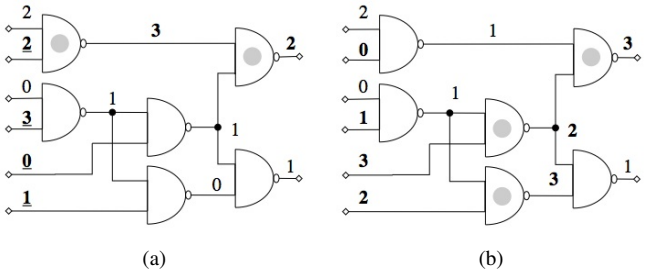

Figure 3. Input pair generation in [9] (a) previous pair (b) new pair.

2.3 Crossover and Mutation for SA

In a GA new chromosomes (in our case pairs) are generated using crossover and mutation. Mutation is applied to each individual obtained after crossover. However, crossover is usually applied to two parent chromosomes to produce two new children chromosomes for a new generation. Since in SA we work with one pair at each time, we had to adapt the crossover operation to handle only one chromosome. In the design of a GA, we have many ways to implement crossover. We chose one-point crossover. Fig. 4 shows it when applied to one chromosome. We apply crossover to a current pair with

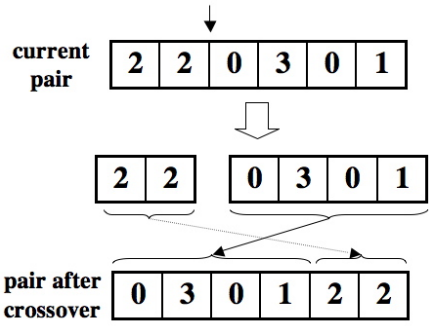

Figure 4. Crossover adapted to work with only one chromosome.

a probability p_c set to 0.9. When the pair crossovers, the point of crossover is determined randomly and the pair is cut in two and those parts transpose. When the pair does not crossover it is left unchanged. After crossover the pair is subject to mutation. Mutation is applied to each gene with a probability p_m given by equation (3).

$$p_m = \frac{1}{number\ of\ inputs\ in\ the\ circuit} \tag{3}$$

To see if a gene changes or not, we generate a random number between 0 and 1. If this number is smaller than p_m the gene takes a value different from the one it has. One example of mutation is shown in Fig. 5. In it, three genes change to new values generating a new pair p_{i+1} to be used in the next iteration of the SA algorithm.

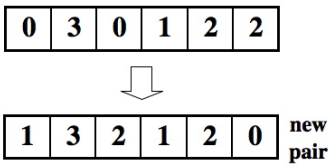

Figure 5. Mutation to generate a new input pair.

3. Experimental Results

We ran a set of simulations using the ISCAS85 combinational benchmark circuits. In the cooling scheme we set α to 0.5, 0.6, 0.7, and 0.9 to see the range where we obtain the best results. We determined that the best values where within 0.6 and 0.7. To complement these first results we ran a new set of simulations with α set to 0.62, 0.64, 0.66, and 0.68. The maximum number of switching gates is plotted for all these values of α in Fig. 6. We set T_0 (the initial temperature in

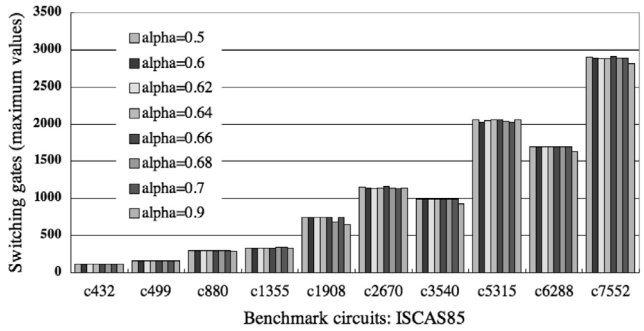

Figure 6. Maximum number of gates for different values of α.

SA) to 1000. The inner and outer loop parameters (N_{in} and N_{out}) were both set to 100. These values of T_0, N_{in} and N_{out} were chosen to compare our results to those of [9]. The best results were obtained with α set to 0.66, and are shown in Table 1 together with the results of [8], [9] and [11]. In Table

Table 1. Comparison with other methods.

ISCAS85		Max. Switching Gates				
circuit	gates	[8]	[8]GA	[9]	[11]	SA-GA
c432	161	145	NA[1]	102	103	119
c499	202	118	NA	157	160	161
c880	383	318	318	280	235	304
c1355	546	299	296	327	328	326
c1908	880	601	588	710	526	748
c2670	1193	809	791	1130	927	1161
c3540	1669	921	919	953	834	990
c5315	2307	1484	1449	1912	1321	2057
c6288	2416	1564	1539	1630	1324	1697
c7552	3512	2178	2100	2933	2235	2916

[1]NA: Not Available

2 we set the values of [9] as reference and could see that our approach excels at the number of switching gates for almost all the benchmark circuits of ISCAS85 (the only exception is c7552). These figures show that our approach leads to better input pairs than those of the SA of [9] and also better than those given in the iterative method of [8] or in [11] for a GA

Table 2. Comparison taking [9] as reference.

ISCAS85	[9]=1				
circuit	[8]	[8]GA	[9]	[11]	SA-GA
c432	1.42	NA	1.0	1.01	1.17
c499	0.75	NA	1.0	1.02	1.03
c880	1.14	1.14	1.0	0.84	1.09
c1355	0.91	0.91	1.0	1.00	1.00
c1908	0.85	0.83	1.0	0.74	1.05
c2670	0.72	0.70	1.0	0.82	1.03
c3540	0.97	0.96	1.0	0.88	1.04
c5315	0.78	0.76	1.0	0.69	1.08
c6288	0.96	0.94	1.0	0.81	1.04
c7552	0.74	0.72	1.0	0.76	0.99

approach. We have added also in these tables the values referenced in [8] for a GA (these are shown as [8]GA). With the hybrid SA-GA method, we see an increase in the number of switching gates that ranges from 3% to 17%. We ran all of our experiments in a Dual Core 2GHz PC the same machine machine used in [11]. The corresponding running times are given in Table 3 together with the values published in [11]. The running time reduction is mainly due to the fact that the number of iterations does not depend on the number of inputs as it is does in [9].

Table 3. Running Time Comparison.

ISCAS85	run time (secs)			[9]=1		
circuit	[9]	[11]	SA-GA	[9]	[11]	SA-GA
c432	45.7	1.3	1.2	1	0.03	0.03
c499	44.7	1.1	1.0	1	0.02	0.02
c880	241.0	4.0	3.9	1	0.02	0.02
c1355	241.1	6.0	5.8	1	0.02	0.02
c1908	366.6	11.2	11.0	1	0.03	0.03
c2670	3178	13.7	13.4	1	0.004	0.004
c3540	1217	24.5	24.1	1	0.02	0.02
c5315	6735	39.1	38.9	1	0.01	0.01
c6288	3082	99.9	103.5	1	0.03	0.03
c7552	9908	51.3	52.6	1	0.01	0.01

Therefore, the running times of our hybrid SA-GA are only a small fraction (0.4% to 3%) of the one needed by the SA of [9] and comparable to the GA of [11].

4. Conclusions

To ensure reliability, signal integrity, and an adequate thermal design of a circuit we need to know its maximum power dissipation. If we have several implementations of a design and have to choose the lowest power consuming one, we need to know their maximum power consumption. This work aims to find the pair of inputs that cause the maximum number of switching gates (hence the maximum power consumption) in a combinational circuit. We have shown a hybrid SA-GA method that is very promising. We obtained results similar or better than those given by a pure SA method. Our results in some cases improved them even in 17%. The running time of our SA-GA is no more than 3% of the time required by the SA of [9].

We worked with only one value of the initial temperature of SA (T_0). Clearly, other values need to be evaluated. We also used a geometric cooling scheme in this work, but plan

to use the cooling scheme of [9] and others to make further comparisons and see the influence of the cooling scheme in the hybrid approach. As detailed in the crossover section, we used a fixed value of 0.9 for p_c. Usually, a high value of it in a GA leads to rapid convergence to a local solution. However, in our work it led to very good results. Further work is needed regarding the setting of this value and the one of mutation (p_m). This last one has been made dependent on the number of the inputs the circuit has, but we need to see the influence of it when set to a constant value.

References

[1] P. Agrawal,Srinivasa R. STG, A. N. Oke, and S. Vijay, "A Scalable Modeling Technique to Estimate Dynamic Thermal Design Power of Datapath Intensive Designs," Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI'07), pp.389-394, March 2007.

[2] L. Zhong, S. Ravi, A. Raghunathan, and N. K. Jha, "Power Estimation for Cycle-accurate Functional Descriptions of Hardware," Proceedings of the 2004 International Conference on Computer Aided Design (ICCAD'04), pp. 668-675. November 2004.

[3] R. L. Wright and M. A. Shanblatt, "Improved Power Estimation For Behavioral and Gate Level Designs," Proceedings of the IEEE Computer Society Workshop on VLSI 2001 (WVLSI'01), pp. 0102, April 2001.

[4] Savithri S., Venkatesan R., and Bhaskar, "An Assertion Based Technique for Transistor Level Dynamic Power Estimation," Proceedings of the 13th. International Conference on VLSI Design, pp. 34, January 2000.

[5] T. Kim, K.S. Chung, C. L. Liu, "A Static Estimation Technique of Power Sensitivity in Logic Circuits," Proceedings of the 38th Design Automation Conference (DAC'01), pp.215-219, June 2001.

[6] C.Y. Wang and K. Roy, "Maximum Power Estimation for CMOS Circuits Using Deterministic and Statistical Approaches," Proceedings of the 9th. International Conference on VLSI Design, pp. 364-369, January 1995.

[7] C.Y. Wang and K. Roy, "COSMOS: A Continuous Optimization Approach for Maximum Power Estimation of CMOS Circuits," Proceedings of the 1997 International Conference on Computer Aided Design (ICCAD'97), pp. 52-55. November 1997.

[8] K. Zhang, H. Takase, T. Hayashi, and H. Kita, "An Enhanced Iterative Improvement Method for Evaluating the Maximum Number of Simultaneous Switching Gates for Combinational Circuits," Proceedings of the 1997 Asia and South Pacific Design Automation Conference (ASP-DAC'97), Chiba, Japan, pp.107-112, January, 1997.

[9] Alberto Palacios Pawlovsky, "Using simulated annealing to generate input pairs to measure the maximum power dissipation in combinational CMOS circuits," IEICE ELEX, Vol.2, No.4, pp. 115-120, February 2005.

[10] Y.M. Jiang, K.T. Cheng and A. Krstic, "Estimation of Maximum Power and Instantaneous Current Using a Genetic Algorithm," Proceedings of the IEEE Custom Integrated Circuits conference (CICC'97), pp.135-138, May 1997.

[11] Alberto Palacios Pawlovsky, "On the use of Genetic Algorithms in Generating Input Pairs that Cause the Maximum Power Consumption in CMOS Combinational Circuits," Proceedings of the XIII Iberchip Workshop (IWS-2007), Lima, Peru, pp.183-186, March, 2007.

[12] S. Kirkpatrick, C. D. Gelatt, Jr., M. P. Vecchi, "Optimization by Simulated Annealing," Science, Vol. 220, No. 4598, pp.671-680, May 1983.