ON THE USE OF GENETIC ALGORITHMS IN GENERATING INPUT PAIRS THAT CAUSE THE MAXIMUM POWER CONSUMPTION IN CMOS COMBINATIONAL CIRCUITS

Alberto Palacios Pawlovsky

Toin University of Yokohama, Faculty of Engineering
Department of Electronics and Information Engineering
pawlovsky@cc.toin.ac.jp

ABSTRACT

We have been investigating on the generation of input pairs for combinational digital circuits that cause the maximum power consumption on them. This work shows the results we obtained using genetic algorithms (GA). We show a GA that runs almost in the same time that speed up simulated annealing algorithms and give better results than them for half of the ISCAS85 benchmark circuits.

1. INTRODUCTION

Developments in circuit integration bring us now chips with billions of transistor and the possibility of building complete systems on a chip. But, this increase in integration has also fostered the increase in power consumption and the heat a circuit has to endure. Usually we build these huge designs from many building blocks and we need to know their power requirements to determine hot spots and layout the system to accomplish certain requirements. Failing to do this could endanger the reliability of the system or short its life span. A lot of different approaches have been proposed to measure the power consumption of digital circuits. We have methods for combinational circuits [1], sequential circuits [2], and some that apply to both types [3]. We can divide these methods into simulation-based and non-simulation methods. There are methods to measure the dynamic power, others that focus on leakage power and methods to measure the average total power in a circuit [4]. Power consumption in a CMOS circuit consists of two main components: dynamic power and leakage power. We have been studying a non-simulation method that aims to find the input pair that causes the maximum dynamic power consumption in a combinational circuit. Since dynamic power is proportional to the number of switching gates in a circuit, we have been looking for methods that maximize this number. We have already proposed a simulated annealing (SA) based method that proved to be good but run time consuming for some circuits [5, 6]. We also relaxed some constraints in it, and published some results on a fast version of it [7]. We have been developing a GA-SA hybrid method to research the benefits and drawbacks of this kind of method when applied to the problem described above. This work gives some preliminary results about the GA implementation we have been able to develop and the results we have obtained with it. In the following section we briefly explain our method to generate new input vector pairs using genetic algorithms. In section 3 we show the results of the experiments we performed with that scheme. In section 4 we show some conclusions and topics for future work.

2. GENETIC ALGORITHM

Our GA is basically the one described by Goldberg [8]. We map a pair of inputs into a chromosome. In our scheme encoding is not binary. We represent each pair of binary values in a four-value system. In other words, a **00**, **01**, **10**, and **11** are represented by a **0**, **2**, **3**, and **1**, respectively (see Fig. 1).

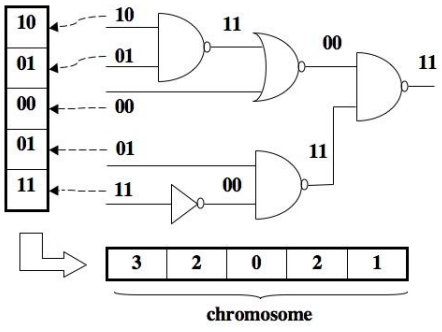

Figure 1: Input pairs and chromosomes.

2.1. Chromosomes and Switching Gates

Each input pair (represented by a chromosome) causes that certain gates switch in a circuit. As shown in Fig. 2(a) and Fig. 2(b) there are ineffective and effective chromosomes. We evaluate them in terms of their number of switching gates. In our GA 100

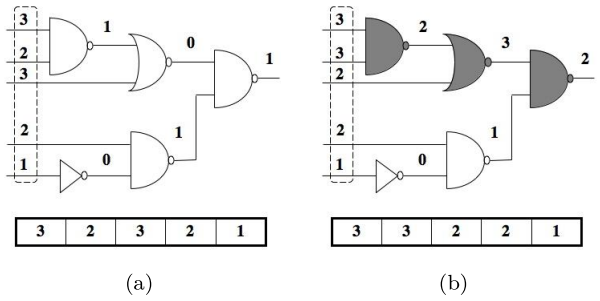

Figure 2: Inputs and switching gates (a) ineffective chromosome (b) effective chromosome.

chromosomes (or parents) form a generation (see Fig. 3). We have limited the number of chromosomes in a generation to be able to compare the running time of our GA to previous schemes.

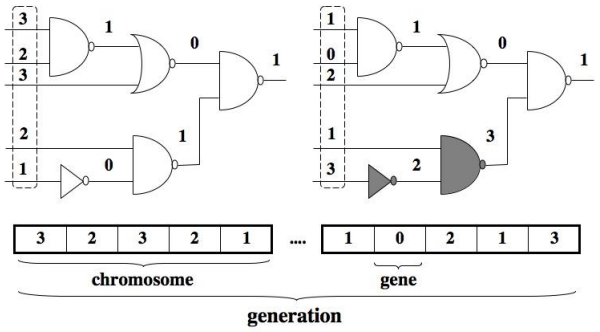

Figure 3: Chromosomes in a generation.

2.2. Reproduction

In our scheme parents are not selected randomly, we choose adjacent parents in pairs. We do not shuffle the mating pool and generate a new generation combining characteristics of two parents to form two new children. Our GA is a generational one, where an entire generation is replaced with each iteration.

2.3. Recombination

Children are formed by modifying their parents by crossover and mutation. For each consecutive pair

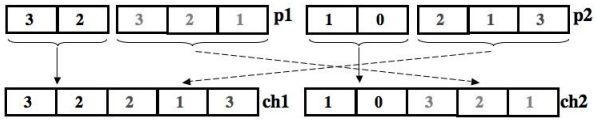

Figure 4: Crossover and children generation.

of parents we apply crossover (see Fig. 4) with probability p_c. Otherwise, we copy the parents.

For each offspring (child) we apply mutation with probability p_m independently for each gene. These allows us to generate new alleles (other forms of the same gene) as shown in Fig. 5.

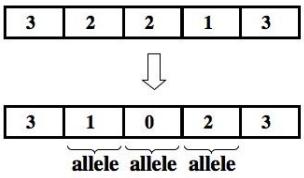

Figure 5: Mutation and generation of alleles.

The size of the population is recommended to be set by Eq.(1) , as suggested in [2].

$$population\ size = \mathbf{128} \times \sqrt{\mathbf{circuit\ \ inputs}} \quad (1)$$

This way of setting the number of chromosomes would let us maintain diversity, but on the other hand will increases the number of trials and the time needed to run them. As indicated above we limited the number of chromosomes in a generation to 100 individuals, and test 100 generations to be able to compare run time and results with previous methods for the same number of trials. Those results are shown in the next section.

3. EXPERIMENTAL RESULTS

As detailed above the processes of selecting parents, crossover, and mutation produces a new generation from an existing one. The first generation is randomly generated and the evolution process is repeated a 100 times. We run these simulations for a variety of values for p_c and p_m. We used values of p_c in the range $0.6 \sim 0.9$, and values of p_m in the range $0.4 \sim 0.9$. The value of p_c determines if two parents crossover to generate new children or not, and p_m determines if the value of a gene must change. If a gene must change it will always take another value, never the one it has. The fitness function is a simple counting function that determines the number of switching gates of a chromosome. The

best one in a generation is compared with the best of all the generations so far evaluated and if it exceeds becomes the best one. Otherwise it is just discarded. The best values we obtained for the ISCAS85 benchmark circuits [9] are shown in Table 1 (GA are our values). The values used for p_c are

Table 1: Switching gates and settings of our work

ISCAS85		Switching gates, settings		
circuit	gates	GA	p_c	p_m
c432	161	103	0.6	0.6
c499	202	160	0.7	0.6
c880	383	235	0.9	0.4
c1355	546	328	0.6	0.6
c1908	880	526	0.9	0.4
c2670	1193	927	0.7	0.8
c3540	1669	834	0.8	0.8
c5315	2307	1321	0.9	0.4
c6288	2416	1324	0.8	0.5
c7552	3512	2235	0.9	0.6

those recommended in the literature [10]. Many authors recommend very low values of p_m (0.01 in [2]) or values between $1/population \sim 1/circuit\ inputs$, but none of these settings improved the numbers shown in Table 1.

A comparison with other similar works and our results with simulated annealing are shown in Table 2. If we set the fast simulated annealing method as a reference, we can see that our GA implementation match those values and improves in more than a half of them in 10% or 20%. However, it still gives lower values than the iterative method of [1] or the SA-based method of [5].

In Table 3 we show the running times in seconds of three of these methods on the same machine (2 GHz Intel Core Duo, 2GB 667 MHz DDR2 SDRAM). Clearly the number of switching gates obtained with the SA of [5] are the best, but as this table shows its running time is the longest. The fast SA of [7] and the GA of this work have similar running times. If we again take as reference the fast SA of [7] the best figures of merit ($switching\ gates/running\ time$) are obtained with the GA described in this work.

4. CONCLUSIONS

We have shown in this work one way of implementing the GA method for generating input pairs for CMOS combinational circuits. The results are obtained almost in the same time of those of the fast SA method of [7]. We have been able to improve those results in 10% or 20% for half of the ISCAS85 benchmark circuits. Further work is needed to determine the best size of the generations and the number of generations needed to get better results. One possible choice is the use of Eq.(1). In this case we would possibly need a fewer number of generations (see [2]). We have not implemented survivor selection. This is present only in the case where crossing criteria is not met, parents are copied and no alleles appear in the new children. However, in this case the fitness of the parents is not taken into account, so there is no guarantee that good chromosomes goes into a new generation. This and other criteria as partially mapped crossover [11] and multiparent recombination [12] are topics of on going research. We need to study also the way in which we are going to use the characteristics of GA when merging it with SA to form a hybrid approach. We are considering now two schemes. The first one uses GA for the generation of new pairs in a pure SA method. The other one uses SA in the generation of the children in a pure GA method.

5. ACKNOWLEDGMENTS

We want to thank here to Mr. T. Kikuchi, Mr. N. Maeda and Mr. A. Nikaidou for helping us in running many of the simulations and gathering data from them for this work.

6. REFERENCES

[1] 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.

[2] Michael S. Hsiao, "Genetic Spot Optimization for Peak Power Estimation in Large VLSI Circuits," VLSI Design, Vol. 15(1), pp.407-416, 2002.

[3] Yi-Ming Jiang, Kwang-Ting Cheng, and Angela Krstic, "Estimation of Maximum Power and Instantaneous Current Using a Genetic Algorithm," Proc. of the IEEE 1997 Custom Integrated Circuits Conference, pp.135-138, 1997.

[4] Yongjun Xu, Jinghua Chen, Zuying Luo, and Xiaowei Li, "Vector Extraction for Average Total Power Estimation," Proceedings of the 2005 Asia and South Pacific Design Automation Conference

Table 2: Comparison with other methods.

ISCAS85		switching gates				[7]=1			
circuit	gates	[1]	[5]	[7]	GA	[1]	[5]	[7]	GA
c432	161	145	102	90	103	1.6	1.1	1	1.1
c499	202	118	157	140	160	0.8	1.1	1	1.1
c880	383	318	280	195	235	1.6	1.4	1	1.2
c1355	546	299	327	305	328	1.0	1.1	1	1.1
c1908	880	601	710	505	526	1.2	1.4	1	1.0
c2670	1193	809	1130	772	927	1.0	1.5	1	1.2
c3540	1669	921	953	812	834	1.1	1.2	1	1.0
c5315	2307	1484	1912	1290	1321	1.2	1.5	1	1.0
c6288	2416	1564	1630	1242	1324	1.3	1.3	1	1.1
c7552	3512	2178	2933	2258	2235	1.0	1.3	1	1.0

Table 3: Time comparison and figure of merit.

ISCAS85		run time (secs)			[7]=1			figure of merit		
circuit	gates	[5]	[7]	GA	[5]	[7]	GA	[5]	[7]	GA
c432	161	45.7	1.3	1.3	35.7	1	1.0	0.032	1	1.1
c499	202	44.7	1.1	1.1	41.0	1	1.0	0.027	1	1.2
c880	383	241.0	4.0	4.0	59.8	1	1.0	0.024	1	1.2
c1355	546	241.1	5.9	6.0	40.6	1	1.0	0.026	1	1.1
c1908	880	366.6	11.2	11.2	32.9	1	1.0	0.043	1	1.0
c2670	1193	3178	13.7	13.7	231.8	1	1.0	0.006	1	1.2
c3540	1669	1217	24.5	24.5	49.6	1	1.0	0.024	1	1.0
c5315	2307	6735	38.9	39.1	173.4	1	1.0	0.009	1	1.0
c6288	2416	3082	98.0	99.9	31.5	1	1.0	0.042	1	1.0
c7552	3512	9908	50.8	51.3	195.0	1	1.0	0.007	1	1.0

(ASP-DAC 2005), Shanghai, China, pp.1086-1089, January 2005.

[5] 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 25, 2005.

[6] A. Palacios Pawlovsky and H. Ishikawa, "A Study of Cooling Schemes for the Generation of Input Pairs of CMOS Combinational Circuits Using the Simulated Annealing Algorithm," Proceedings of the 2005 International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC 2005), Jeju, Korea, pp.179-180, July, 2005.

[7] A. Palacios Pawlovsky and M. Haraguchi, "On the Speeding Up of the Simulated Annealing Algorithm in Generating Input Pairs for CMOS Combinational Circuits," Proceedings of the 2006 International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC 2006), Chiang Mai, Thailand, Vol. I, pp.I21-I24, July, 2006.

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

[9] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits," Proc. IEEE International Symposium on Circuits and Systems, pp. 695-698, 1985.

[10] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, Springer-Verlag, Berlin Heidelberg, 2003.

[11] D.E. Goldberg and J. R. Lingl, "Alleles, Loci and the Traveling Salesman Problem," Proceedings of the International Conference on Genetic Algorithms and their Applications, pp. 154-159, Hillsdale NJ, 1985.

[12] A. E. Eiben, "Multiparent Recombination in Evolutionary Computing," *Advances in Evolutionary Computing: Theory and Applications*, Springer-Verlag, New York, NY, 2003.