Genetic Algorithm Based Fine-Grain Sleep Transistor Insertion Technique for Leakage Optimization

Yu Wang, Yongpan Liu, Rong Luo, and Huazhong Yang

Department of Electronics Engineering, Tsinghua University,
Beijing, 100084, P.R. China
wangyuu99@mails.tsinghua.edu.cn

Abstract. Fine-grain Sleep Transistor Insertion (FGSTI) is an effective leakage reduction method in VLSI design optimization. In this paper, a novel Genetic Algorithm (GA) based FGSTI technique is presented to decide where to put the sleep transistors (ST) when the circuit slowdown is not enough to assign sleep transistors everywhere in the combinational circuits. Penalty based fitness function with a built-in circuit delay calculator is used to meet the performance constraint. Although optimal FGSTI problem is proved to be NP-hard, our method can steadily give a flexible trade-off between runtime and accuracy. Furthermore a Successive Chromosome Initialization method is proposed to reduce the computation complexity when the circuit slowdown is 3% and 5%. Our experimental results show that the GA based FGSTI technique can achieve about 75%, 94% and 97% leakage current saving when the circuit slowdown is 0%, 3% and 5% respectively.

1 Introduction

With the development of the fabrication technology, leakage power dissipation has become comparable to switching power dissipation [1]. It is known that leakage power may make up 42% of total power at the 90nm technology node [2]. Thus various techniques are proposed to reduce the leakage power from system level down to physical level. Among these, Multi-Threshold CMOS (MTCMOS) technique is the most effective one, in which sleep transistors (ST) are placed between the gates and the power/ground (P/G) net in order to put the circuit into sleep mode when it is standby.

MTCMOS technique can be mainly categorized into two approaches: block based sleep transistor insertion (BBSTI) technique [3-6] and fine-grain sleep transistor insertion (FGSTI) technique [7-9]. In BBSTI, all the gates in the circuits are clustered into sizable blocks and then these blocks are gated using large ST; all the gates are assumed to have a fixed slowdown. On the other hand, FGSTI technique assigns ST with appropriate size to individual gates in the circuit while the circuit performance constraints are still satisfied as shown in Fig. 1. It is easier to guarantee circuit functionality in a FGSTI technique [8], since ST sizes are not determined by the worst case current of large circuit blocks which is quite difficult to determine without comprehensive simulation [3]. In addition, FGSTI technique leads to a smaller simultaneous switching current when the circuit changes between standby mode and active mode comparing to BBSTI technique. Furthermore, better circuit slack utilization can be achieved as the slowdown of each gate is not fixed, and then leads to a further reduction of leakage and area [7] [9].

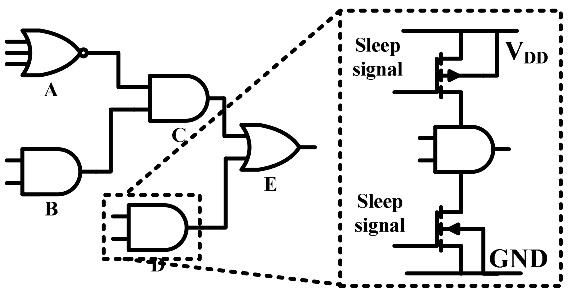

Fig. 1. Fine-grain sleep transistor insertion technique

The most different thing is that FGSTI technique can be performed when the circuit speed is not influenced, while BBSTI technique will definitely induce certain circuit slowdown, about 5% or more for most combinational circuits [9]. Thus the FGSTI technique is changed into a slack distribution problem to determine which gate can be assigned with ST. Recently, [9] use a one-shot heuristic algorithm to determine where to put ST in a FGSTI design, but how to perform FGSTI technique isn't addressed when the circuit slowdown is 0% and the one-shot heuristic algorithm may easily fall into a local optimal result. Our previous work [7] presents a mixed integer programming (MLP) model for FGSTI technique. Since the MLP problem is proved to be NP-hard [10], the MLP problem for large size circuit may take unbearable time to converge.

Ever since the genetic algorithm was introduced by Holland [11], lots of empirical evidences have indicated that GA can find good solutions to some complex problems. In this paper, a novel GA based FGSTI technique for leakage optimization is proposed. Our contributions include:

1. To our best knowledge, this is the first work to use GA based techniques to decide where to put ST in an FGSTI problem which is NP-hard. Penalty based fitness function is adopted to perform genetic search from both feasible and unfeasible solution space. Furthermore, our method can give a flexible trade-off between runtime and leakage saving which is becoming more important while the problem size is growing up. The computation complexity of our method turns out to be quite stable.
2. A Successive Chromosome Initialization (SCI) method is invented based on the successive attribute of the chromosome to further reduce the computation time. Our experiments show that this method could reduce the computation time significantly.

The rest of our paper is organized as follows. The preliminaries are given in Section 2. The detailed genetic algorithm optimization framework is illustrated in

Section 3. Section 4 is devoted to our SCI method. The implementation and experimental results are shown and analyzed in Section 5. Finally, we draw the conclusions.

2 Preliminaries

A combinational circuit is represented by a directed acyclic graph $G = (V, E)$ where a vertex $v \in V$ represents a CMOS gate from the given library, and an edge $(i, j) \in E$, i, $j \in V$ represents a connection from vertex i to vertex j.

2.1 Leakage Current Model

The original leakage current of gate v is denoted as $I_{w/o}(v)$, while the leakage current of gate v assigned with ST is denoted as $I_w(v)$. Obviously, the leakage current of gate v with ST depends on the ST's size. We choose the largest ST size $(W/L)v = 16$ for simplicity, which leads to the minimum delay overhead as shown below. Due to the stacking effect, $I_{w/o}(v)$ is about two orders of magnitude larger than $I_w(v)$. Thus if more gates in the circuit are assigned with ST, more leakage saving is achieved.

Extensive HSPICE simulations are used to create two leakage current look up tables for all the gates in the circuits to represent these two values: $I_{w/o}(v)$ and $I_w(v)$.

2.2 Delay Model

As shown in [12], the gate delay is influenced by the ST insertion. The load dependent delay $d_{w/o}(v)$ of gate v without ST is given by:

$$d_{w/o}(v) = \frac{KC_L V_{DD}}{(V_{DD} - V_{THlow})^{\alpha}} \tag{1}$$

where C_L, V_{THlow}, α, K are the load capacitance at the gate output, the low threshold voltage, the velocity saturation index and the proportionality constant respectively. The propagation delay $d_w(v)$ of gate v with ST can be expressed as:

$$d_w(v) = \frac{KC_L V_{DD}}{(V_{DD} - 2V_x - V_{THlow})^{\alpha}} \tag{2}$$

where V_x is the drain to source voltage of the ST. Suppose that $I_{ON}(v)$ is the current flowing through ST during the active mode, it can be expressed as given by [9]:

$$I_{ON}(v) = \mu_n C_{ox}(W/L)_v((V_{DD} - V_{THhigh})V_x - \frac{V_x^2}{2}) = \mu_n C_{ox}(W/L)_v(V_{DD} - V_{THhigh})V_x \tag{3}$$

where μ_n is the N-mobility, C_{ox} is the oxide capacitance, V_{THhigh} is the high threshold voltage, $(W/L)_v$ represents the size of the ST inserted to gate v. The voltage drop V_x in gate v due to ST insertion can be expressed as:

$$V_x = \frac{I_{ON}(v)}{\mu_n C_{ox}(V_{DD} - V_{THhigh})} \times \frac{1}{(W/L)_v} \tag{4}$$

Combining equation (1), (2) and (4), the propagation delay $d_w(v)$ of gate v with ST can be rewrite as:

$$d_w(v) = d_{w/o}(v) + \left(\left(1 - \frac{2\dfrac{I_{ON}(v)}{\mu_n C_{ox}(V_{DD} - V_{THhigh})} \times \dfrac{1}{(W/L)_v}}{V_{DD} - V_{THlow}} \right)^{-\alpha} - 1 \right) d_{w/o}(v) = d_{w/o}(v) + \varphi((W/L)_v) d_{w/o}(v) \tag{5}$$

where $d_{w/o}(v)$ is constant which can be extracted from the technology library. Referring to equation (5), a larger $(W/L)_v$ leads to a smaller delay overhead. Here the largest ST size $(W/L)_v = 16$ is still chosen which makes $\varphi((W/L)_v)$ a constant.

3 Genetic Algorithm Based FGSTI Technique

The FGSTI problem is first formulated as a mathematical model:

$$\begin{cases} \min \quad I_{leak} = \sum\limits_{v \in V} \left(I_{w/o}(v) \times (1 - ST(v)) + I_w(v) \times ST(v) \right) \\ \text{Subject to:} \\ \qquad D_{circuit} \le T_{req} \\ \qquad ST(v) \in \{0,1\}, \ \forall v \in V \end{cases} \tag{6}$$

where I_{leak} is the total leakage current in the circuit; $ST(v)$ is used to represent sleep transistor state of gate v: $ST(v) = 1$ means that gate v is assigned with sleep transistors, $ST(v) = 0$ means that gate v is not modified; $D_{circuit}$ is the longest path delay of the modified circuit; T_{req} represents the circuit performance constraint. $D_{circuit}$ is derived by a built-in longest path calculator using the delay models in Section 2.

Genetic Algorithm is widely applicable for complex problems and makes few assumptions from the problem domain; and it is not biased towards local minimums. A genetic algorithm based FGSTI technique is developed to solve above problem. The representation structure, chromosome initialization fitness function and genetic operator of our genetic algorithm are shown as follows.

3.1 Encoding and Chromosome Initialization

A binary vector $B = (ST(v_1), ST(v_2), \ldots, ST(v_N))$ is used as a chromosome to represent where to assign ST in a combinational circuit, and N refers to the total gate number. Apparently, if every gate in the circuit is not modified, the performance constraints are satisfied. Hence a chromosome (0, 0, …, 0) is a surely feasible chromosome. Suppose the population size is M, the other M-1 chromosomes are randomly chosen in order to gain a better capability to search the whole state space.

3.2 Fitness Function

Penalty terms are used in our fitness functions in order to perform genetic search from both feasible and infeasible parts in the search space towards the global optimal results. Therefore, two terms are included in our fitness function: the total leakage current for feasible solution and the penalty term for infeasible solution. Assuming B^k is the kth chromosome in the current population, N is the length of the chromosome, and

$ST^k(v_i)$ is the binary variable for each gate. The total leakage current of the circuit for the kth chromosome is directly derived by equation (6):

$$f_k(B^k) = \sum_{i=1}^{N} \left(I_{w/o}(v_i) \times (1 - ST^k(v_i)) + I_w(v_i) \times ST^k(v_i) \right) \tag{7}$$

For some chromosomes, the modification to the original circuit leads to a large total circuit delay variance which may violate the circuit performance constraints in equation (6). Those chromosomes project to infeasible solutions. The penalty coefficient P_k is proportional to the difference between the modified circuit delay and the performance required.

$$P_k = \begin{cases} 0, & \text{if constraint in (6) is satisfied} \\ \alpha_0 (D_{circuit} - T_{req}), & \text{else} \end{cases} \tag{8}$$

where α_0 is a large user-specified positive penalty value.

Hence our fitness function can be given as:

$$eval(B^k) = \frac{1}{f_k(B^k) + P_k}, \ k = 1, 2, \ldots, M \tag{9}$$

where M is the population size.

Referring to our previous work on Static Timing Analysis (STA) [13], an extended Breadth-first search is used to calculate the circuit delay of modified circuits. The computation complexity of BFS in a DAG $G = (V, E)$ is $O(V+E)$; thus our algorithm runs in time $O(N)$, where N is the total gate number, and also the length of the chromosome in our genetic algorithm.

3.3 Elitist Selection Strategy

The chromosomes are selected by ranking of the chromosomes according to their fitness from initial to final stage of genetic search. This mechanism can maintain the diversity of the species in the beginning of genetic search, as the fitnesses are scaled down so that the influence of high fitness is diminished; while at the later stage of the genetic search, when most of the chromosomes have similar high fitnesses, fitness ranking can address the effect of higher fitness and thus facilitate selection of the best chromosome for faster convergence [14].

The tournament selection is adopted to preserve the best chromosome from the current generation to the next generation.

3.4 Chromosome Crossover and Mutation

A “Scattered” crossover function is used; it first creates a random binary vector with the same length as the chromosome and then selects the genes from the first parent where the vector is a 1, and the genes from the second parent where the vector is a 0. With further research in the future, genes in the chromosome may be grouped to several highly dependent parts due to the circuit attributes. Hence the scattered crossover function can be easily adapted to crossover the fixed groups of genes.

An adaptive mutation method is used to avoid disrupting a good chromosome from based on non-uniform mutation method [15]. The key idea is that mutations are probabilistically performed more towards the weak chromosomes in order to explore different regions, meanwhile the best few chromosomes are disrupted with much less probability than those with weak fitnesses in order to find the optimum solutions. This strategy is especially important at the later stage of generations.

As shown above, the computation complexity of each generation is $O(N^*N_C)$, where N is the chromosome length and N_C is the population size. Suppose the genetic algorithm takes M generation to converge to an acceptable results, the total complexity of our GA based FGSTI is $O(M^*N^*N_C)$.

4 Successive Chromosome Initialization

Generally speaking, the solution space of a GA problem consists of two parts: feasible solution region and infeasible solution region. Referring to our specialized GA problem, T_{req} which corresponds to the performance constraint, can be chosen from the original circuit delay to 1.05 times the original circuit delay. That is, the modified circuit delay may vary in the range of 5% from the original circuit delay. Suppose when the circuit performance is not influenced by adding sleep transistors to various gates in the circuits, the feasible solution region is Region A in Fig. 2.

Obviously, every solution in Region A must be a feasible solution when the performance constraint T_{req} changes to 1.03 and 1.05 times the original circuit delay. Furthermore every feasible solution of 3% circuit slowdown must be a feasible solution when the circuit slowdown is 5%. Thus the feasible solution regions of 3% and 5% are represented using Region B and C respectively in Fig. 2.

It should be clear that the solution spaces need not to be convex or continuous as shown in Fig. 2. Suppose solution a and b are the best individual in Region A and Region B respectively, which means a is the best individual when the circuit slowdown is 0% and b is the best individual when the circuit slowdown is 3%. When we solve the problem under the 3% circuit slowdown constraint, a is definitely a feasible solution. As the solution space is 2^N, where N is the chromosome length, the possibility of which almost all the initial chromosomes are in the infeasible solution region is very large. It may take a long time for genetic search to find the feasible solution region. Naturally, it is reasonable that the best solution of 0% circuit slowdown is used as one of the initial chromosome when the circuit slowdown is 0%, and this may reduce the computation complexity because a may be very near to b the best solution of Region B. Furthermore, we can use the last generation of 0% circuit slowdown as the initial generation of 3% circuit slowdown. It is the same case that the best solution b or the last generation of 3% circuit slowdown can be used for initial point of 5% circuit slowdown. This chromosome initialization mechanism is called “Successive Chromosome Initialization” in our specialized GA problem.

From Fig. 2 we can further confirm why using a penalty based fitness function. Suppose the best solution of 5% circuit slowdown is c_2, as shown in Fig. 2 the distance

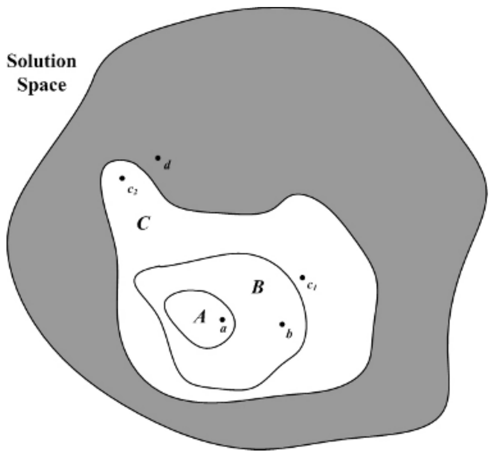

Fig. 2. Solution space: feasible solution region for different constraints and infeasible solution region in grey

between c_2 and infeasible solution d is much less than the distance between c_2 and feasible solutions a, b, c_1. Since we do not know where the best solution is in the solution space, the genetic search should be performed from both directions: feasible solution region and the infeasible solution region.

5 Implementation and Experimental Results

ISCAS85 benchmark circuit is used to verify our GA based FGSTI technique, all the netlists are synthesized using Synopsys Design Compiler and a TSMC 0.18μm standard cell library. The two leakage current look up tables for all the standard cells with and without sleep transistors are generated using HSPICE. The values of various transistor parameters have been taken from the TSMC 0.18μm process library, i.e. V_{DD}=1.8V, V_{THhigh}=500mV, V_{THlow}= 300mV, and I_{ON}= 200μA for all the gates in the circuit. The genetic algorithm are implemented using MATLAB.

We assume $(W/L)_v$= 16, corresponding to a delay variance of 6% if we assign sleep transistors to all the gates in the circuit [7]. Thus when the circuit slowdown varies in the range of 6% circuit original delay, we can not assign sleep transistors to every gate in the circuit.

The gate number N of the circuit is also the chromosome length as shown in Table 1, the search space of the problem is very large. When N is smaller than 1000, we set the population size to 200 and the max generation number to 1500; when N is larger than 1000, we set the population size to 500 and the max generation number to 4000. Table 1 shows the leakage saving using our GA based FGSTI technique; these results are the best ones of five runs.

Comparing to the leakage savings with MLP method [7]: 79.8%, 94%, 95% for 0%, 3%, 5% circuit slowdown respectively, the GA based FGSTI method is comparable. The MLP model for a certain circuit consists of about 7N variables and

Table 1. Leakage reduction using GA based FGSTI technique

ISCAS85 Benchmark Circuits	Gate Number *N*	Original leakage current (pA)	0% circuit slowdown (pA)	3% circuit slowdown (pA)	5% circuit slowdown (pA)
C432	169	4609	1764	479	211
C499	204	21375	14530	1080	109
C880	383	9261	684	318	179
C1355	548	11874	6495	1666	806
C1908	911	23418	3065	1027	316
C2670	1279	35191	2081	564	372
C3540	1699	40370	3470	1154	284
C5315	2329	56292	2938	1008	634
Leakage saving	N/A	N/A	74.8%	94.6%	97.7%

the corresponding constraints for each variable, hence the problem size is becoming extremely large when the gate number increases. It is unstable since it does not converge well for some circuits in our experiments. However, our GA based FGSTI technique can give a better solution for the circuits that MLP model can not converge well; this leads to a larger leakage saving when the ciruit slowdown is 3% and 5%.

In our GA based FGSTI technique, the population size and the max generation number are controllable; meanwhile the genetic algorithm can perform a linear scale down of the object function. Therefore, it is more flexible to solve this problem using GA compare to MLP method which is unstable and with less controllable.

Table 2 shows the computation complexity using our Successive Chromosome Initialization method. There are two different Successive Chromosome strategies: 1. using the best individual as one of the initial point; 2. using the last generation including the best individual as the initial pool. They are represented as Successive Chromosome Initialization with best individual (SCI_BI), Successive Chromosome Initialization with last generation (SCI_LG). The population size is set to 200, the generation number when the best results are achieved are compared shown in Table 2. These results are average of five runs. As we can see, the SCI_BI strategy and the SCI_LG strategy is efficient compared to the original one.

Table 2. Computation complexity comparison using generation number (Original Chromosome Initialization, SCI_BI, SCI_LG)

	C432 0% Slow down	C432 3% slow down	C432 5% slow down	C499 0% slow down	C499 3% slow down	C499 5% slow down
Original	107	132	167	110	156	176
SCI_BI	N/A	37	24	N/A	117	59
SCI_LG	N/A	30	22	N/A	102	45

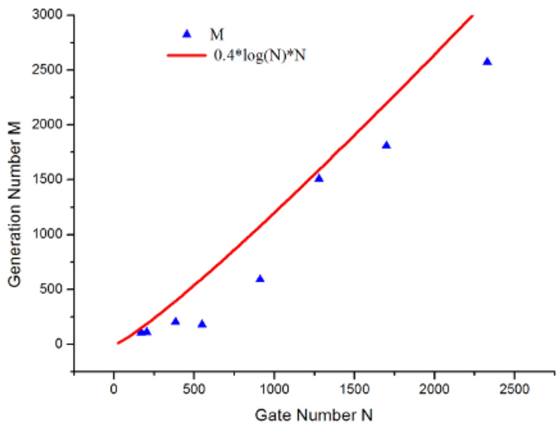

Fig. 3. Complexity analysis (Gate Number *N* vs Generation Number *M*)

Furthermore, we look into the complexity of our GA based FGSTI technique. In this case, all the population size is set to 200; the search is stopped when the leakage saving is within 5% compared to the results shown in Table 1. Fig. 3 shows the relationship between the gate number *N* (chromosome length) and the final generation number *M*.

From Fig. 3, it is shown that our genetic algorithm is stable, since all the generation number *M* is below the red line which corresponds to function 0.4*log(*N*)**N*. From our previous complexity analysis, the total complexity of our GA based FGSTI is $O(M*N*N_C)$, where *N* is the chromosome length and N_C is the population size, *M* is the generation number. As N_C is assumed to be a constant, the computation complexity of our GA based FGSTI technique is $O(N*\log(N)*N)$ based on our experimental results.

6 Conclusions

In this paper, a novel genetic algorithm based FGSTI technique is proposed, which can assign sleep transistors to appropriate gates in order to achieve the max leakage saving when the circuit is standby. Penalty based fitness function with a built-in circuit delay calculator is used to meet the performance constraint. The GA based FGSTI technique can achieve about 75%, 94% and 97% leakage current saving when the circuit slowdown is 0%, 3% and 5% respectively. Furthermore, the experimental results show that our SCI mechanism leads to further reduction of the computation time when the circuit slowdown is 3% and 5%. Our genetic algorithm is stable based on our complexity analysis, which is superior to the MLP approaches.

Acknowledgement

The authors would like to thank the grants from the National 863 project of China (No. 2005AA1Z1230), National Natural Science Foundation of China (No. 60506010).

Thanks to Qian Ding for helpful suggestions during coding and using MATLAB. We would like to thank all the anonymous reviewers.

References

1. G. Moore, “No exponential is forever: But forever can be delayed,” in IEEE ISSCC Dig. Tech. Papers, 2003, pp. 20 - 23
2. J. Kao, S. Narendra, A. Chandrakasan, “Subthreshold Leakage modeling and reduction techniques”, in Procs. of ICCAD, 2002, pp 141 – 149
3. J. Kao, S. Narendra, A. Chandrakasan, “MTCMOS hierarchical sizing based on mutual exclusive discharge patterns,” in Procs. of DAC, 1998, pp. 495–500
4. M. Anis, S. Areibi, and M. Elmasry, “Dynamic and leakage power reduction in MTCMOS circuits using an automated efficient gate clustering technique,” in Procs of DAC, 2002, pp. 480–485
5. W. Wang, M. Anis, S. Areibi, “Fast techniques for standby leakage reduction in MTCMOS circuits” in Procs. of IEEE SOC, 12-15 Sept. 2004 pp 21 – 24
6. C. Long, L. He; “Distributed sleep transistors network for power reduction” in Procs. of DAC, 2-6 June 2003 pp. 181 – 186
7. Y. Wang, H. Lin, HZ. Yang, R. Luo, H. Wang, “Simultaneous Fine-grain Sleep Transistor Placement and Sizing for Leakage Optimization”, in Procs. of International Symposium on Quality Electronic Design (ISQED)’06, March 2006, pp. 723-728
8. B. H. Calhoun, F. A. Honoré, and A. P. Chandrakasan, “A Leakage Reduction Methodology for Distributed MTCMOS,” IEEE JSSC Vol. 39, No. 5, May 2004, pp. 818 - 826
9. V. Khandelwal, A. Srivastava; “Leakage Control Through Fine-Grained Placement and Sizing of Sleep Transistors ,” in Procs. of ICCAD 2004, pp 533 - 536
10. Gerard Sierksma, Linear and Integer Programming: theory and practice, Marcel Decccckker, 2002
11. J. H. Holland, Adaptation in Natural and Artificial Systems (Univ. of Michigan Press, Ann Arbor, MI, 1975; reprinted by MIT Press, Cambridge, MA, 1992).
12. S. Mutoh et al. “1-V Power Supply High Speed Digital Circuit Technology with Multithreshold Voltage CMOS,” in IEEE JSSC, Vol. 30, No. 8 August 1995
13. Y. Wang, HZ. Yang, H. Wang, “Signal-path Level Dual-Vt Assignment for Leakage Power Reduction”, in Journal of Circuits, System and Computers Vol. 15, No. 2 (2006)
14. David E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, 1989
15. T. C. Fogarty, "Varying the probability of mutation in genetic algorithms," in Proceedings of Third International Conference on Genetic Algorithms, pp. 104-109, 1987