Meta-heuristics for Circuit Partitioning in Parallel Test Generation

C. Gil[1], J. Ortega[2], A.F. Díaz[2], M.G. Montoya[1]
[1]Dept. de Arquitectura de Computadores y Electrónica. Univ. Almería
Almería (SPAIN). E-mail: cgil@peke.ualm.es
[2] Dept. de Arquitectura y Tecnología de Computadores. Univ. Granada
Granada (SPAIN)

Abstract

In this communication Simulated Annealing and Genetic Algorithms, are applied to the graph partitioning problem. These techniques mimic processes in statistical mechanics and biology, respectively, and are the most popular meta-heuristics or general-purpose optimization strategies. A hybrid algorithm for circuit partitioning, which uses tabu search to improve the simulated annealing meta-heuristics, is also proposed and compared with pure tabu search and simulated annealing algorithms, and also with a genetic algorithm. The solutions obtained are compared and evaluated by including the hybrid partitioning algorithm in a parallel test generator which is used to determine the test patterns for the circuits of the frequently used ISCAS benchmark set.

1 Introduction

As general-purpose parallel computers are increasingly being used to speed up different VLSI applications, the development of parallel algorithms for circuit testing, logic minimization and simulation, HDL-based synthesis, etc. is currently a field of increasing research activity. The circuit partitioning problem arises in many VLSI applications [1,2]. Due to the increasing complexity of VLSI circuits, the NP-complete [3] character of many VLSI CAD problems makes a "divide and conquer" approach more attractive to solve these problems in reasonable periods of time, and so circuit partitioning has become an important previous step in this kind of applications.

Several approaches for circuit partitioning have been reported [5-11]. They can be classified as move-based approaches [5], and cluster-based approaches [6]. The move-based procedures build the solution iteratively by applying a move or transformation to the current solution. The set of possible transformations that can be applied to a given solution defines the neighbourhood structure of the solution space, which is explored repeatedly by moving from the current solution to a neighbouring one. The move-based procedures are simple to describe and implement, and thus, are the most frequently used. These procedures include iterative improvement methods [5,7-8], which move from the current solution to the best solution in its neighbour, and stochastic meta-heuristics, such as Simulated Annealing (SA) [10,12], Tabu Search (TS) [9,13], and Genetic Algorithms (GA) [11,14], which allow movements towards solutions worse than the current one in order to escape from local minima. Hybrid algorithms that mix the previous meta-heuristics are also possible. In [11], a heuristic based on Tabu Search and Genetic

Algorithms is applied to the circuit partitioning problem, and a classification of hybrid algorithms is also provided. In this paper, we also present a hybrid algorithm that inserts elements of a Tabu Search into a Simulated Annealing algorithm, the Mixed Simulated Annealing and Tabu Search (MSATS) algorithm. The results provided by MSATS are compared with those obtained with a Tabu Search, a Simulated Annealing, and a Genetic Algorithm, respectively.

In the following, Section 2 gives a more precise definition of the circuit partitioning problem and presents the cost function associated with the circuit partitioning problem. The description of MSATS is provided in Section 3. In Section 4, a brief description of the Genetic Algorithm implemented to solve the circuit partitioning problem is provided. Finally, the experimental results provided by the different meta-heuristics are compared in Section 5, and Section 6 gives the conclusions of the paper.

2 The Circuit Partitioning Problem

The circuit partitioning problem consists of finding a decomposition of the target circuit into non-overlapping subcircuits with at least one logical gate in each subcircuit. Among the different objectives that may be satisfied by the desired partitioning are (i) the minimization of the number of cuts, (ii) the minimization of the number of subcircuits, and (iii) the minimization of the deviation in the number of elements (inputs, logical gates, outputs and fanout points) assigned to each partition. As our goal is to use all the available processors in the multicomputer to process the circuit in parallel while trying to keep all the processors working during the whole run time, the number of subcircuits is fixed to be equal to the number of available processors in the machine, and the objectives correspond to criteria (i) and (iii). This means obtaining subcircuits with similar sizes to balance the workload of the processor (considered as proportional to the number of nodes), and minimizing the number of cuts. In the following, a formulation of the problem is provided.

Let $G = (X, A)$ be the directed acyclic graph associated with a combinational circuit C, where X denotes the set of components (inputs, logical gates, and outputs) and A the set of lines used for signal propagation. The nodes of X can be classified as inputs, logical gates and outputs of circuit C. Thus X is the union of three disjoint sets, the set of inputs E, the set of logical gates P (nodes), and the set of outputs S.

The problem is to find a partition of X into a fixed number of K subsets X_k, k=1,...K, such that each induced subgraph G_k (X_k, A_k) satisfies the following conditions:

1. $X = \bigcup_{k=1}^{K} X_k$ and $X_k \cap X_h = \varnothing, \forall k \neq h, (k,h) \in \{1,...,K\}^2$
2. $p_k = \left| X_k \cap P \right| \neq \varnothing \quad \forall k = 1,\ldots,K$
3. $L_i \leq \leq p_k \leq L_u$, with $L_i = \lfloor n/K \rfloor - \lfloor n/K \rfloor * \theta$ and $L_u = \lfloor n/K \rfloor + \lfloor n/K \rfloor * \theta$, $\forall k$=1,...,K; where $n = |P|$; $\lfloor n/K \rfloor$ represents the number of gates that should be included in each subcircuit to obtain a partition of similar sized subcircuits;

and θ is the parameter representing the proportion of gates that is tolerated as a deviation with respect to $\lfloor n/K \rfloor$. In this work, θ has been set to values between 0.2 and 0.3.

4. $G_k(X_k, A_k)$ is a connected graph, $\forall k = 1,...,K$.

In this way, given a circuit graph $G=(X,A)$, the problem is formulated as a combinatorial optimization problem in which the cost function $c(s)$ to minimize is defined as:

$$c(s) = \alpha \cdot n\_cuts(s) + \beta \cdot \sum_{k=1}^{K} 2^{deviation(k)} \quad (1)$$

where:

$$deviation(k) = \text{maximum } \{ 0,\ |X_k \cap P| - L_u ,\ L_i - |X_k \cap P| \},$$

n_cuts(s) is the number of cuts of the solution, and *s* is any solution to the circuit partitioning problem, feasible or not, i.e. verifying the above condition 3 or not. Thus, whenever for all *k=1,..,K*, in a given partition *s*, the deviation in the number of gates of G_k with respect to $\lfloor n/K \rfloor$ is less than $\theta.\lfloor n/K \rfloor$, the solution is feasible and the cost is $c(s) = \alpha \cdot n\_cuts(s) + \beta K$, since $deviation(k) = 0$ ($k=1,..,K$). The second term in (1) penalizes the deviation from the feasible solution space and its magnitude is determined by the constant β. Nevertheless, according to the relative magnitude of α and β, a transition to a solution *s* determining a deviation greater than $\theta \cdot \lfloor n/K \rfloor$ in the number of gates of any subcircuit still decreases the cost function if the reduction in the number of cuts is sufficiently large.

3 The MSATS Algorithm

The MSATS (Mixed Simulated Annealing Tabu Search) procedure is a hybrid method that uses the best features of the two meta-heuristics, Simulated Annealing and Tabu Search [12-14], to outperform the results provided by each. At each iteration of MSATS, admissible moves are applied to the current solution, allowing transitions that increase the cost function as in Simulated Annealing. When a move increasing the cost function is accepted, the reverse move is forbidden during some iterations in order to avoid cycling, as in Tabu Search. The restrictions in the admissible moves are implemented by using a short term memory function which determines how long a tabu restriction will be enforced and the admissible moves at each iteration.

In MSATS, the temperature *t* is used as a parameter to control the probability of accepting a new solution, as in Simulated Annealing. At a given temperature, only the solutions which are selected by the SA cooling schedule are considered as candidates to produce a transition. Thus, a certain randomness is introduced into a pure TS, in order to explore zones of the solution space that do not appear very promising at first. As the algorithm also has the characteristics of a Tabu Search, it avoids the cycles around local minima, allowing a more efficient exploration of the solution space without revisiting solutions, as may occur in a pure SA.

Two different initial solutions, s1 and s2, have been used in our experiments. They are obtained by fast algorithms that assign *n/K* nodes to each partition. The initial solution s1 is obtained by an algorithm named Input Partitioning, in which the circuit graph is traversed in a depth-first way, starting from the inputs. Solution s2 is provided by an algorithm called Output Partitioning which processes the graph in a depth-first manner from the output nodes. These fast partitioning algorithms take $O(n)$ time to obtain partitions in which strongly connected components of the graph are assigned to the same partition.

MSATS stops when one of the following conditions is verified: (*i*) the temperature is equal to a final value (in this paper a temperature equal to zero has been used as final temperature), (*ii*) the number of moves applied without improving the best solution found so far (*n_failures*) reaches a maximum bound of consecutive iterations (*max_failures*), and (*iii*) the number of iterations reaches the value *max_iteration.*

4 Genetic Algorithm

A Genetic Algorithm simultaneously examines and manipulates a set of possible solutions. Each candidate solution is represented by a string of symbols called a chromosome. The set of solutions, P_j, is referred to as the population of the j^{th} generation. The population evolves for a prespecified total number of generations under the application of evolutionary rules called Genetic Operators

```
GENETIC  Algorithm
Begin
    Select popu_size, max_gen, num_gen=0
    Initialize Population
    while (num_gen < max_gen) do
        Evaluate Fitness
        for (i=1 to popu_size)
                Select (mate1,mate2)
                if (rnd(0,1) leq cross_rate) then
                        child = Crossover(mate1, mate2)
                if (rnd(0,1) leq mutate_rate) then
                        child = Mutation( )
        end_for
        Add offsprings to new generation
        num_gen=num_gen+1
    end_while
Return best chromosomes
End_Genetic_Algorithm
```

Fig. 1. Genetic Algorithm for circuit partitioning.

The Genetic Algorithm implemented is shown in Fig. 1. It begins with an encoding and initialization phase during which each string in the population is

assigned a uniformly distributed random point in the solution space. Each iteration of the genetic algorithm begins by evaluating the fitness of the current generation of strings. A new generation of offspring is created by applying crossover and mutation to pairs of parents who have been selected according to their fitness. The specific characteristics of the GA implemented are described below.

Solution Representation: Let K be the number of subcircuits into which the circuit with graph G is divided, and let n ($n=|P|$) be the number of logic gates of the original circuit; then each solution is represented by an array S of n elements as

$$S=A_1, A_2,A_3,...,A_n \text{ with } A_i \in [1,...,K], \forall i \in \{1,...,K\}$$

where the A_i element in array S represents the subcircuit to which the logic gate i belongs. Then, for the circuit partitioning problem, the representation is based on an integer array of length M.

Once the coding has been fixed, the next step is to initialize the population. Let N be the population size. The algorithm has been run by using solutions s1 and s2, indicated in Section 3, and random solutions as initial ones.

Fitness function: The fitness function is obtained directly from the cost function described in (1).

Reproduction or selection: The fitness value c_i of the best string i of Generation T is compared with the fitness value c_j of the worst string j of generation T+1**.** If $c_i>c_j$, then string j is replaced by string i. This strategy ensures that the maximum fitness value of the population does not decrease as the process of evolution continues.

Crossover: Several crossover procedures have been tried in order to achieve an efficient search of the solution space. These crossover operators should generate feasible solutions when applied to a given population in order to implement an efficient search in the solution space. It begins by randomly choosing a cut point b where $1 \leq b \leq L$, where L is the string length; an interval is then generated with b and b+rank. The elements of the string that correspond to boundary gates in the two selected solutions will be exchanged.

Mutation: The operation of mutation involves the perturbation of a string position which has been randomly chosen. The perturbation involves changing the string position to one of its possible positions taking into account the pseudo-inputs and pseudo-outputs.

Stop condition: The algorithm terminates either (i) when convergence is reached, i.e., all the strings in the population have nearly equal values for their fitness function, or (ii) when a previously specified number of generations is reached.

5 Experimental Results

In this section we summarize the results obtained by using the algorithms MSATS, SA, TS and GA for circuit partitioning. The algorithms are programmed in C and executed on a Power Challenge XL (Silicon Graphics). The benchmark

circuits used to evaluate the performances are those included in the ISCAS´85 [23], which is the set of circuits usually considered to evaluate test-pattern generation procedures**.**

The values of the parameters are set to their best values according to previous experimental results obtained. These values for the MSATS, SA and TS algorithms are the following. *Max_failures* is set to 0.25*(*max_iterations*), and *tfactor* to 0.99. The value of *max_iterations* is usually taken as 1000, and the initial temperature, t_0 as 100. For these values, the cost function reaches a stable final value at the end of the 1000 iterations. The parameters that produced the best experimental results in the GA algorithm are the following: *Population size:* Among the population sizes checked, the value producing best experimental results was 100. *Probability of crossover*: Good performance is associated with a crossover probability from 0.8 to 0.9. *Probability of mutation:* The probability of mutation varies from 0.01 to 0.09.

Table 1 presents the best results obtained by SA, TS, GA, and MSATS compared with the initial solution s1 for partitions of 2, 4, 8, and 16 subcircuits. The row *cuts reduction* indicates the average of the reduction in the number of cuts obtained by MSATS with respect to the best solution of those provided by SA, TS, and GA, in each case. As can be seen, the results obtained by MSATS outperform those obtained with TS, SA, and GA in most cases. As the circuit size increases, the neighbourhood of a given solution also grows, and the effect of considering tabu transitions is more important in MSATS. Table 1 compares the computing times for the different algorithms. As can be seen the times for MSATS are similar to those of Simulated Annealing and lower than those of TS.

The MSATS algorithm has been used as a first step in a parallel test-pattern generator. It starts by applying MSATS to partitioning the circuit under test, and after this each processor receives one of the subcircuits obtained. Thus, it is possible for all the processors to concurrently apply the test generation algorithm to determine the test patterns for the stuck-at faults in the nodes of the corresponding subcircuit [16]. Thus, one way to demonstrate the performance of MSATS is to consider the increase in the speedup provided by the parallel test-generator when the number of processors grows. If the speedup grows proportionally to the number of processors, the performance of MSATS is adequate according to the conditions given in Section 2. Speedup results for the ISCAS´85 circuits are provided in Table 2. The parallel test generator has been run in a multicomputer Intel Paragon. The speedups obtained with *K* processors are given in the columns labeled *S_K* and the number of cuts produced when the circuit is partitioned into *K* subcircuits are given in the columns *C_K* in Table 2.

Table 1. A summary of results with the best obtained solutions and the times for the algorithms: initial solution (s1), Simulated Annealing (SA), Tabu Search (TS), Genetic Algorithm (GA), and MSATS.

Circuit	Algorithmm	*K*=2		*K*=4		*K*=8		*K*=16	
		cuts	sec.	cuts	sec.	cuts	sec.	cuts	sec.
	s1	66	0,2	106	0,3	158	0,4	186	0,5
	SA	23	2,0	45	2,2	68	2,8	93	3,9
c432	TS	22	6,1	41	9,2	62	11,2	87	13,6
	GA	26	10,6	42	15.8	73	23,8	96	29,7
	MSATS	**20**	2,1	**40**	2,3	**61**	2,8	**85**	3,7
	cut red%	10		3		2		3	
	s1	44	0,3	100	0,4	155	0,6	222	0,7
	SA	19	3,0	41	3,8	98	5,1	**129**	6,3
c880	TS	19	18,1	43	28,5	89	35,7	**129**	42,3
	AG	22	21,3	54	32,5	92	40,8	135	48,7
	MSATS	**17**	3,1	**37**	3,9	**80**	4,9	**129**	6,1
	cut red%	11		10		11		0	
	s1	123	0,6	210	0,7	298	0,9	450	1,2
	SA	55	8,0	74	9,5	125	11,5	168	12,9
c1908	TS	50	54,2	76	60,2	120	68,5	156	78,5
	GA	50	65,4	**70**	72,4	**108**	81,8	143	89,1
	MSATS	**35**	8,2	71	9,5	**108**	10,9	**125**	12,1
	cut red%	30		-2		0		13	
	s1	116	0,8	333	0,9	535	1,2	771	1,5
	SA	67	15,0	156	16,2	250	20,9	375	26,5
c3540	TS	96	49,2	179	83,2	259	76,4	377	90,7
	GA	60	69,2	149	91,6	225	99,8	295	106,3
	MSATS	**46**	15,1	**132**	16,9	**221**	20,1	**298**	25,4
	cut red%	24		12		2		2	
	s1	60	1,4	155	1,8	334	2,3	671	2,9
	SA	48	28,1	135	30,8	315	35,1	450	44,8
c6288	TS	**46**	115,3	157	145,7	320	168,5	550	211,8
	GA	**46**	126,7	112	187,4	**243**	234,3	470	254,6
	MSATS	**46**	28,4	**102**	30,6	301	34,2	**355**	42,6
	cut red%	0		9		-20		22	

Table 2. Speedups, number of cuts and fault coverages obtained with the parallel test-pattern-generator.

Circt.	faults	S_2	C_2	S_4	C_4	S_8	C_8	S_16	C_16	Cover.
c432	864	1.69	20	3.70	40	6.23	61	10.34	85	98%
c499	998	1.78	16	3.85	54	6.34	78	11.43	119	99%
c880	1760	1.82	17	3.87	37	7.05	80	12.56	129	100%
c1355	2710	1.90	17	2.93	45	6.25	70	11.65	98	98%
c1908	3816	1.97	35	2.50	71	6.86	108	10.26	125	98%
c3540	7080	1.78	46	3.67	132	6.40	221	10.20	298	98%
c6288	12570	1.67	46	2.75	102	6.53	301	10.45	355	97%

6 Conclusions

In this communication, the circuit partitioning problem is considered in the framework of parallel VLSI CAD applications. Specifically, the use of circuit partitioning in a parallel test pattern generator has been presented. The circuit partitioning problem has been formulated as a combinatorial optimization problem by using a cost function comprising the contribution of the number of cuts and the deviation with respect to a balanced distribution of the gates among the different subcircuits. Some general-purpose optimization algorithms such us Simulated Annealing and Genetic Algorithms, which are physically and biologically-based heuristics, respectively, and Tabu Search have been applied to solve the optimization problem considered. We have also developed a new algorithm, called MSATS, which reduces the possibility of cycles in the search process by applying the Tabu Search characteristics to a Simulated Annealing algorithm.

MSATS outperforms the TS and SA algorithms when applied to the same cost function. In a shorter time, MSATS not only provides a lower number of cuts, but also a more balanced distribution of gates among subcircuits. Compared with a Genetic Algorithm, MSATS is able to provide solutions with similar qualities in most cases, and better solutions, but requires less time. Indeed, the obtention of sufficiently good solutions in a short time is very important in order to achieve good efficiencies in the parallel test pattern generator.

Acknowledgements: The authors thank Dr. Inmaculada García for her useful discussions. This paper has been suported by project TIC97-1149 (CYCIT, Spain) .

References

[1] Alpert, C.J., and Kahng, A., "Recent Developments in Netlist Partitioning: A survey". *Integration: the VLSI Journal*, 19/ 1-2 (1995) 1-81.

[2] Kumar, V., Grama, A., Gupta, A., and Karypis, G., *Introduction to Parallel*

Computing. Design and analysis of algorithms, The Benjamin/Cummings Publishing company, 1994.

[3] Garey, M.R., and Johnson, D.S, *Computers and Interactibility: A Guide to the Theory of NP-Completeness*, W.H. Freeman & Company. San Francisco, 1979.

[4] Kumar, V., Grama, A., and Rao, V.N., "Scalable load balancing techniques for parallel computers", *Journal of Distributed and Parallel Computing*, 22 (1994) 60-79.

[5] Kernighan, B.W., and Lin, S., "An Efficient Heuristic Procedure for Partitioning Graphics", *The Bell Sys. Tech. Journal*, (1970) 291-307.

[6] Shin, H., and Kim, C.,"A simple yet effective technique for partitioning", *IEEE Trans. VLSI Systems*, 1/3 (1993).

[7] Fiducia, C.M., and Mattheyses, R.M., "A linear time heuristic for improving network partitionings", *Proc ACM/IEEE Design Automation Conf.*, (1982) 175-181.

[8] Sanchis, L.A.,"Multiple-Way network partitioning with different cost functions", *IEEE Trans. on Comp.*, 42/12 (1993) 1500-1504.

[9] Andreatta, A. A., and Ribeiro, C. C., "A Graph Partitioning Heuristic for the Parallel Pseudo-Exhaustive Logical Test of VLSI Combinational Circuits", *Annals of Operations Research* 50 (1994) 1-36.

[10] Shperling, I., and McCluskey, E.J., "Circuit Segmentation for Pseudo-Exhhaustive Testing Via Simulated Annealing", *International Test Conference IEEE*, (1987).

[11] Areibi, S., and Vannelli, A., "Advanced Search Technique for Circuit Partitioning", *DIMACS Series in DIscrete Mathematics and Theoretical Computer Science*, 16 (1993) 77-98.

[12] Aarts, E., and Korst, J., *Simulated Annealing and Boltzmann Machines. A stochastic Approach to Combinatorial Optimization and Neural Computing*, John Wiley & Sons, 1990.

[13] Glover, F., and Laguna, M., "Tabu Search", in: C.R. Reeves (eds.), *Modern Heuristic Techniques for Combinatorial Problems*, Blackwell, London, 1993, 70-150.

[14] Reeves, C.R., "Genetic Algorithms", in: C.R. Reeves (eds.), *Modern Heuristic Techniques for Combinatorial Problems*, Blackwell, London, 1993, 151-196.

[15] Brglez, F., and Fujiwara, H., "Neural Netlist of Ten Combinational Benchmark Circuts and a Target Translator in FORTRAN", *Proc. IEEE Int. Symp. Circuits Syst., Special Session ATPG*, 1985.

[16] Gil, C. and Ortega, J.,"Parallel Test Generation using circuit partitioning and spectral techniques", *Euromicro Workshop on Parallel and Distributed Processing, IEEE* (January 1998).