Proceedings of the 1999 IEEE Canadian Conference on Electrical and Computer Engineering
Shaw Conference Center, Edmonton, Alberta, Canada May 9-12 1999

GRASP: An Effective Constructive Technique For VLSI Circuit Partitioning

S.M. Areibi
Electrical Engineering Departmen t
Ryerson Polytechnic University
Toronto, Ontario, Canada
sareibi@ee.ryerson.ca

Abstract

Iterative methods are greedy or local in nature and get easily trapped in local optima. Usually interchange methods fail to converge to optimal solutions unless they initially begin from good starting points. The choice of starting point is a very crucial factor in the performance of the iterative improvement algorithms [1]. GRASP is a random adaptive simple heuristic that intelligently constructs good initial solutions in an efficient manner. Good initial partitions obtained by GRASP allow the iterative improvement method to refine that initial partition quality in a reasonable amount of time, thus reducing the computational time and enhancing the solution quality. Results obtained indicate that on average the cut-size is reduced by 20% and speedups of up to 90% were achieved using the GRASP technique.

1 Introduction

Circuit partitioning deals with the task of dividing a given circuit into two or more parts such that the total weight of the signal nets interconnecting these parts is minimi zed while the size of the different parts meet a certain criteria. Traditionally, this problem was important for breaking up a complex system in to several custom ASICs. Now, with the increasing use of FPGA-based emulators and prototyping systems, partitioning is becoming even more critical. While it is possible to solve the case of unbounded partition sizes exactly [2], the case of balanced partition sizes is NP complete [3]. As a result, numerous heuristic algorithms have been proposed [4]. Kernighan and Lin (KL) [5] proposed a two-way graph partitioning algorithm which became the basis for most of the subsequent partitioning algorithms. The KL algorithm operates only on balanced partitions[5] and performs a number of passes over the cells of the circuit where each pass comprises a repeated operation of pairwise cell swapping for all pairs of cells. Fiduccia and Mattheyses (FM) [6] obtained a faster implementation of KL with the help of a new data structure, called the bucket data structure. FM can operate on unbalanced partitions and employs a single cell move instead of a swap of a cell pair at each step in a pass. Krishnamurthy [4] added to FM a look-ahead ability, which helps to break ties better in selecting a cell to move. Sanchis [4] generalized Krishnamurthy's algorithm to a m ulti-way circuit partitioning algorithm. Recently some new approaches that enhanced the performance of the original KL algorithm appeared [7] [8] and the reader is referred to the excellent survey in [4].

1.1 Heuristic Techniques

Iterative improvement techniques based on module interchange are the most robust, simple and successful heuristics in solving the partitioning and placement problems. The main disadv antage of these heuristics is that they mainly focus on the immediate area around the current initial solution, thus no attempt is made to explore all regions of the parameter space. More importantly, it has been shown that interchange methods fail to converge to "optimal" or "near optimal" solutions unless they initially begin from "good" initial starting points [2]. Sechen [9] showed that over 100 trials or different runs were required to guarantee that the best solution would be within twenty percent of the optimum solution.

In this paper we introduce a greedy randomized approach that is capable of obtaining good initial solutions. These solutions allow the iterative improvement technique to further refine them thus reducing the computational time and enhancing the solution quality. Section 2 introduces the main concept behind the GRASP technique and details of implemen tation.

0-7803-5579-2/99/$10.00 © 1999 IEEE

Section 3 gives a detail explanation of the different parameters that affect the GRASP technique and means of tuning them. Finally in Section 4 we summarize the results obtained using GRASP and show that it is applicable for flat Partitioners and also for those based on multilevel criteria [8, 10].

2 GRASP

GRASP is a greedy randomized adaptive search procedure that has been successful in solving many combinatorial optimization problems efficiently [11]. The GRASP methodology was developed in the late 1980s, and the acronym was coined by Feo [12]. Each iteration consists of a construction phase and a local optimization phase. The key to success for local search algorithms consists of the suitable choice of a neighborhood structure, efficient neighborhood search technique, and the starting solution. The GRASP construction phase plays an important role with respect to this last point, since it produces good starting solutions for local search. The construction phase intelligently constructs an initial solution via an adaptive randomized greedy function. Further improvement in the solution produced by the construction phase may be possible by using either a simple local improvement phase or a more sophisticated procedure in the form of Tabu Search or Simulated Annealing.

Next, the various components comprising a GRASP are defined, and a demonstration of how to adapt such a heuristic for the circuit partitioning problem is presented.

2.1 Impleme ntation

Figure 1 shows a generic pseudo-code of the GRASP heuristic. The main body of the GRASP algorithm starts by reading the circuit netlist. The algorithm starts with a construction phase followed by a local improvement phase. The GRASP implementation terminates after a certain number of phases or runs have passed. The construction phase as shown in Figure 1(B) is iterative, greedy and adaptive. It is *iterative* because the initial solution is built by considering one element at a time. The choice of the next element to be added is determined by ordering all elements in a list. The list of the best candidates is called the restricted candidate list (RCL). It is *greedy* because the addition of each element is guided by a greedy function. The construction phase is *randomized* by allowing the selection of the next element added to the solution to be any element in the RCL. Finally, it is *adaptive* because the element chosen at any iteration in a construction is a function of those previously chosen. The improvement phase typically consists of a local search procedure as shown in Figure 1(C). A more sophisticated local search based on Tabu Search can be implemented instead of the simple local search procedure.

```
GREEDY RANDOMIZED SEARCH
(A) MAIN_GRASP()
    1. Read_Circuit_NetList();
    2. do
        3. Construction_Phase(Greedy,Rand,Adapt)
        4. Local_Improvement_Phase(Initial Solution)
        5. Store_Best_Solution(Previous Solutions)
    6. While not Done
    7. Report_Best_Solution()

(B) Construction_Phase()
    1. While (construction not done)
        2. Greedy: Create Candidate List (RCL)
        3. Random: Module = Select from RCL
        4. Adaptive: Add new element to solution
        5. Feasibility: Check Feasibility of Solution
    6. EndWhile

(C) Local_Improvement_Phase()
    1. Read Initial Solution
    2. While (local optimum not reached)
        3. NewSolution = Local_Changes(Solution)
    4. EndWhile
    5. Return_Best_Solution()
```

Figure 1: GRASP (Greedy Adaptive Search)

2.1.1 Construction Phase

Initially, all modules are placed into the same block and the gains associated with modules are calculated in an efficient manner. The discussion here will be based on four-way partitioning and this can be generalized for the multi-way partitioning case. Assume there are n modules and four blocks A, B, C, and D. The heuristic could either place all modules initially in block A and sequentially fill the other blocks by moving the n modules to blocks B, C, and D, or can create a dummy block (say X) and perform the same operation until block X is empty. At each iteration of the construction phase, the gains for moving modules to the current block being filled are examined, and an RCL list is created using the modules with the highest gains. As a module is moved it is locked to its new position (block) and its associated gain is removed from the bucket list. The gains of the other modules affected are updated accordingly. The construction phase terminates when a feasible solution (partition) is generated; i.e, all blocks contain a certain number

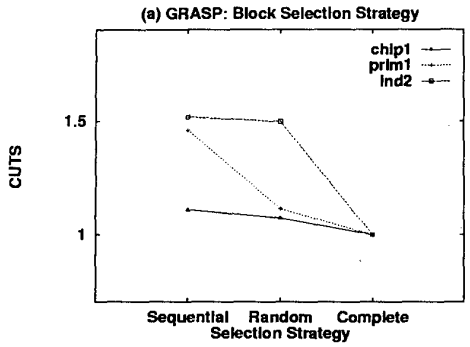

Figure 2: Block Selection Strategy

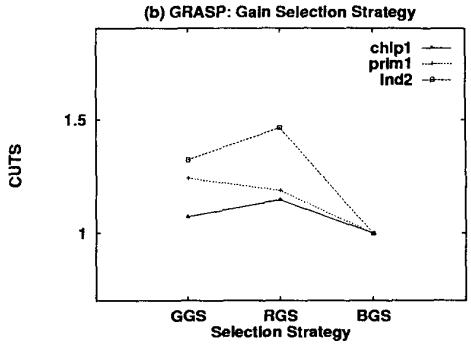

Figure 3: Gain Selection Strategy

of modules (and every module belongs to exactly one partition). The randomness in the GRASP heuristic is due to the selection strategy that is used to determine the next module to be appended to a certain block. The probabilistic component of the GRASP randomly chooses one of k best candidates in the restricted candidate list (RCL), but not necessarily the top candidate. This randomized selection strategy introduces diversification of initial solutions to the method.

3 The GRASP Parameters

The GRASP has two characteristics which make it appealing to researchers. First it is easy to implement, as seen from the previous section. Furthermore, only a few parameters need to be set and tuned. Therefore, development can focus on implementing efficient data structures to assure quick GRASP iterations. Some of the parameters that need to be tuned for the circuit partitioning problem are: *Block Selection Strategy*, *Gain Strategy*, and the *RCL Length*. The Block Selection Strategy determines the order by which blocks are filled to obtain an initial solution. In Random Selection Strategy (*RSS*), all modules are placed into the same block and the initial gains associated with moving modules to every other block are calculated. The other blocks are filled randomly according to the best gain associated with the move involved. In Sequential Selection Strategy (*SSS*), all modules are placed into one block in a similar fashion to (*RSS*), but the other blocks are filled sequentially by removing excess modules from the initially oversized block and placing them into the current block under consideration. In Complete Selection Strategy (*CSS*), all modules are placed in a temporary block say X, and then every other block is filled until completion (all blocks meet the size constraint). As seen in Figure 2[1] CSS gives the best performance with respect to RSS and SSS techniques.

The Gain Strategy determines the highest gain module to be assigned to a certain block. In Greedy Gain Strategy (*GGS*), the module with the highest gain is always selected and assigned to the appropriate block. In Random Gain Strategy (*RGS*) all modules are randomly selected from the RCL and assigned to the blocks according to the Block Selection Strategy used. Finally, the Biased Gain Strategy (*BGS*) is a combination of the above two methods. Figure 3 illustrates that BGS strategy works well for most circuits followed by GGS and RGS respectively. The length of the RCL or restriction imposed on its values is a key success for the implementation of the algorithm. Each GRASP iteration produces a sample solution from an unknown distribution of all obtainable results. The mean and variance of the distribution are functions of the restrictive nature of the candidate list. For example, if the cardinality of the restricted candidate list is limited to one, then only one solution will be produced and the variance of the distribution will be zero. On the other hand if a less restrictive cardinality limit is imposed, many different solutions will be produced implying a larger variance. In [11] two different restrictions are imposed on the RCL, Value Restriction (*RCL-VR*), and Cardinality Restriction (*RCL-CR*). In RCL-VR a module is allowed to be in the restricted candidate list if its gain is within some percentage (α) of the maximum gain. In RCL-CR, the candidate list size is limited by including only the (β) best ele-

[1] The gain has been normalized to one for the three circuits.

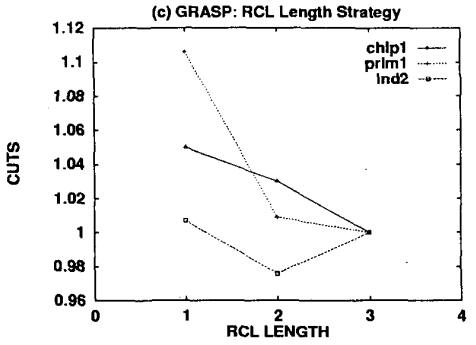

Figure 4: RCL Length Strategy

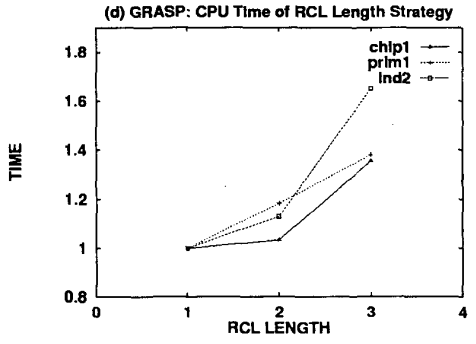

Figure 5: CPU Time of RCL Length Strategy

ments. In this implementation, a third type of restriction based on a combination of RCL-CR and RCL-VR is used (RCL-CVR). Figures 4, 5 show the performance of the GRASP in terms of computation time and quality of solution with different RCL lengths.

4 Computational Results

We experimentally evaluated the quality of partitions produced by GRASP on some hypergraphs that are part of widely used ACM/SIGDA circuit partitioning benchmarks suite [13]. The characteristics of these hypergraphs are shown in Table 1. The number of nodes, nets and pins vary from one circuit to another. The last column indicate the average number of nets connected to a cell. The proposed method is implemented in C language on a Sun Ultra1/140 workstation. Table 2 compares the initial solutions obtained by GRASP and those obtained by a Genetic Algorithm technique[14] and Barnes's eigenvector algorithm[15]. The comparison is based on good initial solutions obtained by each heuristic and the computation time involved to obtain these solutions. The CPU time for the partitions obtained by the Barnes's algorithm include the time for forming the graph adjacency matrix, finding the eigenvalues and eigenvectors. The computation time used by the GRASP is the least compared to the two other methods. The results in Table 3, 4, 5 assess the performance of Sanchis Interchange heuristic, to that of the GRASP. It is clear from the tables that the quality of solutions obtained by the GRASP using only 5 different runs are superior to those obtained using the Sanchis heuristic from 50 different random starting points. The running times of the GRASP are also much faster since less initial solutions are used. Another reason for the fast computation time is that the local search heuristic has to perform less number of passes, since it is starting from a good initial solution, thus the fast convergence is obtained.

Table 6 shows the effect of GRASP on clustering. Here the circuits are first clustered [16], the condensed network is partitioned using a simple dynamic hill climbing search technique. In the second stage, a local search heuristic is used on the flattened network to optimize local partitions of cells. The first part of Table 6 is based on a random partition of the clustered network. The second part is based on a GRASP technique partition. It is evident from the table that GRASP here achieves on average an improvement of up to 10% compared with the random technique. Also, the speedup increases as the size of the circuit increases. For clustered circuits the improvement is not as large as for flat circuits since the clustering approach reduces the complexity of the problem and fewer local minima in the k-interchange neighborhood structure are present. If a multilevel clustering approach were used then GRASP would have obtained better results than the single level criteria used here.

5 Summary

In this paper we presented a greedy randomized adaptive search procedure for circuit partitioning. GRASP is easy to implement and only a few parameters need to be set and tuned. Quality of solutions and running times of GRASP are far superior to those based on Sanchis interchange heuristic. Good initial partitions obtained by GRASP allow iterative

improvement methods to refine the initial partition quality in a reasonable amount of time, thus reducing the computational time and enhancing the solution quality. Currently we are working on a GRASP version for standard cell placement and comparing the results obtained with those based on a global solution using a quadratic measure. This technique will be embedded within a multilevel clustering technique for placement. We are also investigating the effect of a technique called CLIP [7] on the GRASP performance.

Circuit	Nodes	Nets	Pins	Degree
Chip1	300	294	845	6
Chip2	274	239	671	5
Prim1	832	901	2906	9
Ind1	2271	2192	7743	10
Prim2	3014	3029	11219	9
Bio	6417	5711	20912	6
Ind2	12142	12949	47193	12
Ind3	15057	21808	65416	12

Table 1: Benchmarks used as test cases

Ckt	Eigen		GRASP		GA	
	Cuts	Time	Cuts	Time	Cuts	Time
Chip1	80	7.8	83	0.2	65	4.9
Chip2	51	5.1	43	0.2	62	4.1
Prim1	298	13.5	185	2.2	177	14
Prim2	925	183	569	8.8	529	54
Ind1	359	45.1	216	7.1	183	39
Bio	872	163	426	26	1677	105
Ind2	4643	234	1345	78	2363	243
Ind3	6627	647	2456	70	2444	365

Table 2: Constructive Methods for 4-Way Partitioning

Ckt	INTER		GRASP		% IMP	
	Cuts	Time	Cuts	Time	Cuts	Time
Chip1	20	18	20	2.4	-	78%
Chip2	15	19	14	1.7	6%	91%
Prim1	60	91	56	14.0	6%	84%
Prim2	226	433	179	66	21%	84%
ind1	42	211	50	28.8	-16%	86%
Bio	102	1058	89	124	13%	88%
ind2	593	2661	325	155	45%	87%
ind3	514	2294	520	123	-1%	94%
AVG	196	848	156	64	20%	92%

Table 3: GRASP 2-Way partitioning

Ckt	INTER		GRASP		% IMP	
	Cuts	Time	Cuts	Time	Cuts	Time
Chip1	55	27.4	59	3.4	-6%	87%
Chip2	39	24.4	31	3.5	20%	85%
Prim1	155	96.4	127	12	18%	87%
Prim2	627	787	438	102	30%	87%
Ind1	259	526	160	42	38%	91%
Bio	680	2627	386	213	43%	91%
Ind2	2102	12729	1148	312	45%	97%
Ind3	2183	5874	1898	432	13%	92%
AVG	762	2836	530	140	30%	95%

Table 4: GRASP 4-Way partitioning

Ckt	INTER		GRASP		% IMP	
	Cuts	Time	Cuts	Time	Cuts	Time
Chip1	77	39.4	67	4.8	13%	87%
Chip2	63	40.2	59	4.6	6%	88%
Prim1	181	133	153	18.8	15%	85%
Prim2	773	1382	644	101	17%	92%
Ind1	364	769	303	85	17%	89%
Bio	821	4657	451	359	45%	92%
Ind2	2430	25132	1464	1066	40%	96%
Ind3	2640	13131	2843	371	-7%	97%
AVG	918	5660	748	251	18%	95%

Table 5: GRASP 6-Way partitioning

Ckt	Random		GRASP		% IMP	
	Cuts	Time	Cuts	Time	Cuts	Time
Chip1	51	0.3	54	0.4	-5%	-5%
Chip2	36	0.3	34	0.4	5.5%	-5%
Prim1	104	1.4	102	1.4	2%	0%
Prim2	316	6.8	301	6.6	4.7%	3%
Ind1	94	3.3	91	3.7	3.1%	-12%
Bio	267	19	244	17	8.6%	10%
Ind2	650	54	582	48	10.4%	11%
Ind3	1051	79	921	64	12.3%	18%
AVG	321	21	291	17	10%	19%

Table 6: Clustering Based 4-Way partitioning

REFERENCES

[1] L.Y. Song and A. Vannelli. An Efficient Linear Systems Approach for Finding Netlist Partitions. In *Proceedings of The 1992 Custom Integrated Circuits Conference*, pages 5.2.1 – 5.2.4, 1992.

[2] S. Areibi. Ph.D Thesis: Towards Optimal Circuit Layout Using Advanced Search Techniques, 1995.

[3] M.R. Garey and D.S. Johnson. *Computers and Intractability.* Freeman, San Francisco CA, 1979.

[4] C.J. Alpert and A.B. Kahng. Netlist Partitioning: A Survey. *Integration, the VLSI Journal*, pages 64–80, 1995.

[5] B.W. Kernighan and S. Lin. An Efficien t Heuristic Procedure for Partitioning Graphs. *The Bell System Technical Journal*, 49(2):291–307, February 1970.

[6] C.M. Fiduccia and R.M. Mattheyses. A Linear-Time Heuristic for Impro ving Network Partitions. In *Proceedings of 19th DAC*, pages 175–181, Las Vegas, Nevada, June 1982. ACM/IEEE.

[7] S. Dutt and W. Deng. VLSI Circuit P artitioning by Cluster-Removal Using Iterative Improvement Techniques. In *IEEE International Conference on CAD*, pages 194–200. ACM/IEEE, 1996.

[8] C.J. Alpert, J. Huang, and A. Kahng. Multilevel Circuit Partitioning. In *Proceedings of 34th DAC*, pages 530–533, Anaheim, CA, June 1997. ACM/IEEE.

[9] C. Sechen and D. Chen. An improved Objective Function for Min-Cut Circuit P artitioning. *IEEE Transaction on CAD*, pages 502–505, 1988.

[10] G. Karypis and V. Kumar. Multilev el Graph Partitioning Schemes. In *Proceedings of the 1995 Intl conf on Parallel Processing*, pages 113–122. ACM/IEEE, 1995.

[11] T. Feo, M. Resende, and S. Smith. A Greedy Randomized Adaptive Search Procedure for The Maxim um Independent Set. *Operations Research*, 1994. To Appear In: Operations Research.

[12] T. Feo and M. Resende. A Probabilistic Heuristic for a Computationally Difficult Set Covering Problem. *Operation Research Letters*, 8:67,71, 1989.

[13] K. Roberts and B. Preas. Physical Design Workshop 1987. Technical report, MCNC, Marriott's Hilton Head Resort,South Carolina, April 1987.

[14] S. Areibi and A. Vannelli. Advanced Search Techniques for Circuit Partitioning. In *DIMA CS Series in Discrete Mathematics and Theoretical Computer Science*, pages 77–98, Rutgers State University, New Jersey, 1994.

[15] S. Areibi and A. Vannelli. A Combined Eigenvector Tabu Search Approach for Circuit Partitioning. In *Proceedings of The 1993 Custom Integrated Circuits Conference*, pages 9.7.1 – 9.7.4, San Diego, 1993.

[16] S. Areibi and A. Vannelli. An Efficient Clustering Technique for Circuit Partitioning. In *IEEE International Symposium on Circuits and Systems*, pages 671–674, San Diego, California, 1996.

0-7803-5579-2/99/$10.00©1999 IEEE