GENERAL ITERATIVE HEURISTICS FOR VLSI MULTIOBJECTIVE PARTITIONING

Sadiq M. Sait, Aiman H. El-Maleh, Raslan H. Al-Abaji

King Fahd University of Petroleum & Minerals, Computer Engineering,
Dhahran - 31261, Saudi Arabia
{sadiq,aimane,raslan}@ccse.kfupm.edu.sa

ABSTRACT

The problem of partitioning appears in several areas ranging from VLSI, parallel programming, to molecular biology. The interest in finding an optimal partition especially in VLSI has been a hot issue in recent years. In VLSI circuit partitioning, the problem of obtaining a minimum cut is of prime importance. With current trends, partitioning with multiple objectives which includes power, delay and area, in addition to minimum cut is in vogue. In this paper, we engineer two iterative heuristics for the optimization of VLSI netlist bi-Partitioning. These heuristics are based on Genetic Algorithms (GAs) and Tabu Search (TS) and incorporate fuzzy rules in order to handle the multiobjective cost function. Both heuristics are applied to ISCAS-85/89 benchmark circuits and experimental results are reported and compared.

1. INTRODUCTION

Until the beginning of this decade, two main objectives of VLSI circuit design were the minimization of cut-set and the improvement of timing performance. A large number of efforts targeting either one (especially cut-set) or both of the above objectives are reported in the literature [1, 2]. The power consumption of the circuit was not of main concern while trying to optimize the above two objectives. Nevertheless quite a reasonable number of techniques aiming at low power objective are proposed for all phases in physical design including partitioning of circuit, floorplanning, placement and routing [1]. As different techniques are applicable and have been reported at different steps of the VLSI design process [3], the need for a system which incorporates all the three aspects of the design process (delay, cut, power) is increasing.

In this work, we address the above problem in the partitioning step at the physical level. Two iterative approaches based on Genetic Algorithm (GA) and Tabu Search (TS) are presented to solve the multiobjective optimization problem of partitioning. This paper is organized as follows: In the next section, the problem and the cost functions are formulated. Section 3 presents the employed approaches. Experimental results are reported and discussed in section 4.

2. PROBLEM FORMULATION AND COST FUNCTIONS

This work addresses the problem of VLSI netlist partitioning with the objective of optimizing power consumption, timing performance (delay), and cut-set while considering the Balance constraint (same as area constraint as unit area is assumed for every gate). Formally, the problem can be stated as follows: Given a set of modules $V = \{v_1, v_2, ..., v_n\}$, the purpose of partitioning is to assign the modules to a specified number of clusters k (two in our case) satisfying prescribed properties. In general, a circuit can have multi-pin connections (nets) apart from two-pin and therefore it is better to represent it by a hypergraph. A hypergraph $H(V, E)$ is defined where V is a set of nodes and E is a set of hyperedges. Node $v_i \in V$ corresponds to an element (e.g., a gate) in the circuit, and hyperedge $e_i \in E$ corresponds to a net in the circuit. Hyperedge e_i consists of the signal source node $S(e_i)$ and a set of destination nodes $D(e_i)$ and $e_i = (S(e_i), \{D(e_i)\})$. The signal source node $S(e_i)$ of the net e_i corresponds to the output of a gate and the set of destination nodes $D(e_i)$ corresponds to the inputs of the gates. Given a hypergraph $H(V, E)$ with $E = \{e_1, e_2, \cdots, e_m\}$ being the set of signal nets, each net is a subset of V containing the modules connecting the net. It is assumed that for each hyperedge $e \in E$, $|e| \geq 2$ (it connects at least two nodes). Our task is to divide V into 2 subsets (blocks) V_0 and V_1 in such a way that the objectives are optimized, subject to some constraints.

Cutsize The set of hyperedges cut by a cluster C is given by $E(C) = \{e \in E : 0 < |e \cap C| < |e|\}$ i.e., $e \in E(C)$ if at least one, but not all, of the pins of e are in C. The set of nets cut by a partitioning solution p^K can be expressed as $E(p^k) = \bigcup_{i=1}^{k} E(c_i)$ or equivalently $E(p^k) = \{e \in E | \exists u, v \in e, h \neq l$ with $u \in C_h$ and $v \in C_l\}$. We say that $|E(p^k)|$ is the cutsize of p^k. The cost function can be written as follows :

$$Minimize \quad f = \sum_{e \in \psi} w(e) \tag{1}$$

where $\psi \subset E$ denotes the set of off-chip edges. The weight $w(e)$ on the edge e represents the cost of wiring the corresponding connection as an external wire. If all weights equal one, the cost function becomes simpler:

$$Minimize \quad f = |\psi| \tag{2}$$

where $|\psi|$ denotes the cardinality of the set ψ.

Delay In order to deal with a signal path, a hypergraph is decomposed into directed edges $e_k = (S(e_k)), w)$ for $e_k \in E$ and $w \in D(e_k)$. Let the graph which consists of a set of nodes V and a set of decomposed directed edges E be the directed graph $G' = (V, E)$. A signal path is represented by an alternating sequence of nodes and directed edges $v_1, e_1, v_2, e_2, ..., v_{k-1}, e_{k-1}, v_k$, where $e_l = (v_l, v_{l+1})(1 \leq l \leq k-1)$ and $v_i \neq v_j$, $i \geq 1$, $j \leq k$, $i \neq j$. The path from node v_i to node v_j is denoted by p_{ij}. Nodes which are included in the path p_{ij} are defined as $V(p_{ij})$. A path-cut number of path p_{ij}, denoted $ncut(p_{ij})$, is the number of nets cut which are included in the path p_{ij}. In the general delay model where gate delay $d(v)$ and constant inter-chip wire delay are considered, $d_c \gg d(v)$ where d_c is due to the off-chip capacitance denoted as C_{off}. Let the delay of node $v_i \in V$ be $d(v_i)$ and the delay of net $e_k \in E$ which is cut be d_c. Given a partition

0-7803-7761-3/03/$17.00 ©2003 IEEE

$\Phi : (V_A, V_B)$, the path delay $d(p_{ij})$ between nodes v_i and v_j is the sum of the node delays $d(v_i) \in V(p_{ij})$ and the delay of nets which are cut, that is :

$$Minimize \quad d(p_{ij}) = \left(\sum_{v_i \in V(p_{ij})} d(v_i) \right) + d_c \times ncut(p_{ij}) \qquad (3)$$

Power The average dynamic power consumed by a CMOS logic gate in a synchronous circuit is given by:

$$P_i^{average} = 0.5 \frac{V_{dd}^2}{T_{cycle}} C_i^{load} N_i \qquad (4)$$

where C_i^{load} is the load capacitance, V_{dd} is the supply voltage, T_{cycle} is the global clock period, and N_i is the number of gate output transitions per clock cycle. In our work N_i is calculated using the symbolic simulation technique of [4] under a zero delay model. C_i^{load} in Eqn. 4 consists of two components: C_i^{basic} which accounts for the load capacitances driven by a gate before circuit partitioning, and the extra load C_i^{extra} which accounts for the additional load capacitance due to the external connections of the net after circuit partitioning. Then, the total power dissipation of circuit ζ is:

$$P_\zeta = \beta \frac{v_{dd}^2}{T_{cycle}} \sum_{i \in \zeta} (C_i^{basic} + C_i^{extra}) N_i \qquad (5)$$

where β is a constant that depends on technology. When a circuit partitioning corresponds to a physical partitioning, C_i^{extra} of a gate that is driving an external net is much larger than C_i^{basic}.

Area or Balance constraint If we assume that the area of all cells is identical, then the problem reduces to balancing the two partitions in terms of the number of cells. The balance constraint is given bellow:

$$\frac{|\beta_1 - \beta_2|}{\phi} \leq \alpha \qquad (6)$$

where β_i is the number of cells in partition i, ϕ is the total number of cells in the circuit, α is the tolerance which is equal zero in case of a perfect balance.

2.1. Overall Fuzzy Cost Function:

In order to solve the multiobjective partitioning problem, linguistic variables are defined as: cut-set, power dissipation, delay and balance. The following fuzzy rule is used to combine the conflicting objectives:

IF a solution has
Small cut-set **AND**
Low power consumption **AND**
Short delay **AND**
Good Balance
THEN it is a *GOOD* solution.

The above rule is translated to *and-like* OWA fuzzy operator [6] and the membership $\mu(x)$ of a solution x in fuzzy set *good solution* is given as:

$$\mu_{pdcb}^c(x) = \beta^c \times min(\mu_p^c(x), \mu_d^c(x), \mu_c^c(x), \mu_b^c(x)) + (1 - \beta^c) \times \frac{1}{4} \sum_{j=p,d,c,b} \mu_j^c(x) \qquad (7)$$

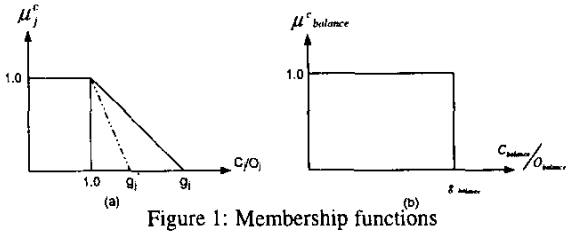

Figure 1: Membership functions

where $\mu^c(x)$ is the membership of solution x in fuzzy set of acceptable solutions, $\mu_{pdcb}^c(x)$ is the membership value in the fuzzy sets of " within acceptable *power*", "within acceptable *delay*", "within acceptable cut-set" and "within acceptable *balance*" respectively. β^c is the constant in the range $[0, 1]$, the superscript c represents the cost. In this paper, $\mu^c(x)$ is used as the aggregating function. The solution that results in maximum value of $\mu^c(x)$ is reported as the best solution found by the search heuristic.
The membership functions for fuzzy sets *Low power consumption, Short delay, Small cut-set*, are shown in Fig. 1(a) We can vary the preference of an objective j in the overall membership function by changing the value of g_j which represents the relative acceptable limits for each objective whrere $g_j \geq 1.0$. Fig. 1(b) represents the membership functions for fuzzy set *good Balance*. O_i is the estimate of lower bound on the cost of an individual i, and C_i is the actual cost of i. O_i's are independent of iteration, therefore, these are estimated only in the beginning. Whereas, C_i has to be calculated in every iteration for every element.

3. PROPOSED APPROACHES

In this section, implementation details of the proposed approaches are described. First, the details of the partitioning Genetic Algorithm for multiobjective optimization are discussed, followed by a brief description of the Tabu Search (TS) implementation.

3.1. Genetic Algorithm (GA) For Timing and Low Power Driven Partitioning

GA algorithm starts with a set of initial solutions called *population* that is generated randomly. In each iteration (*known as generation in GA terminology*), all the individual chromosomes in the population are evaluated using a *fitness function*. Then, in the *selection* step, two of the above chromosomes at a time are selected from the population. The individuals having higher fitness values are more likely to be selected. After the selection step, different operators namely *crossover, mutation* act on the selected individuals for evolving new individuals called *offsprings*. These genetic operators are described below.

In GA implementation we use an encoded representation of solution in the form of a simple string made up of symbols called *genes*. The string of genes is called *chromosome*.

One important genetic operator is *crossover* . It is applied on two individuals that were selected in the selection step earlier to generate an offspring. The generated offspring inherits some characteristics from both its parents in a way similar to natural evolution. There are different crossover operators namely *simple(single point), order, partially mapped, and cycle*. The simple crossover,

for instance, works by choosing a random cut point in both parent chromosomes (the cut point should be the same in both parents) and generating the offspring by combining the segment of one parent to the left of the cut point with the segment of the other parent to the right of the cut [7]. For description of other crossover operators see [2, 7, 8].

The mutation operator is used to introduce new random information in the population. It is usually applied after the crossover operator. It helps in producing some variations in the solutions so that the search does not get trapped in a local minima. An example of mutation operation is the swapping of two randomly selected genes of a chromosome. However, mutation is applied with a low rate so that GA does not turn into a memory-less search process [2]. Two mutation variation are used the first one is by random selection of a cell and swapping its partition. The second is by randomly selecting two cells one from each partition and swapping them.

For addressing a multi-objective optimization problem to minimize three mutually conflicting objectives, a measure is needed which can quantify the overall quality of a solution with respect to all three objectives collectively. Fuzzy membership functions and fuzzy rules are used for evaluating the fitness of a solution. A fitness value between 0 and 1 is assigned to each solution. The fitness value of a chromosome is its membership value $\mu(x)$ in the fuzzy set of acceptable solution. This membership is computed using Eqn. 7. Individuals are selected based on the *elitism-random selection (ernd)*where the best $\frac{N_P}{2}$ chromosomes are selected and the remaining $\frac{N_P}{2}$ are selected randomly. Based on experimental results, this scheme offers better choice than other schemes, because it provides balance between greediness and randomness.

3.2. Tabu Search Approach

In what follows TS implementation is described briefly. Tabu search starts from an initial feasible solution and carries out its search by making a sequence of random moves or perturbations. A Tabu list is maintained which stores the attributes of a number of previous moves. This list prevents taking the search process back to recently visited states. In each iteration, a subset of neighbor solutions is generated by making a certain number of moves and the best move (the move that resulted in the best solution) is accepted, provided it is not in the Tabu list. Otherwise, if the said move is in the Tabu list, it is accepted only if it leads to a solution better than the best solution found so far (aspiration criterion). Thus, the aspiration criterion can override the Tabu list restrictions. The solution encoding and initialization steps are similar to those described above for GA. In each iteration, we generate a number of neighbor solutions by making perturbations as follows: two cells are selected randomly, then their locations are interchanged. The number of neighbor solutions generated in each iteration is dependent on circuit size. The characteristic of the move that we keep in Tabu list is the indices of the cells involved in interchange. The size of Tabu list is taken also depending on the circuit size i.e., 10% of the total number of cells. In this work, short term memory element was used for TS implementation. The aspiration criterion used is as follows, if the current best solution is the best seen so far i.e., better than the global best, then it is accepted and Tabu restriction is overridden.

3.3. Experimental results and Discussion of GA versus TS

The results obtained from GA and TS are compared in terms of overall quality of best solution and run time in Table 1. $P(sp)$ represents the cost due to power, that is the sum of the switching probabilities of all the cut nets; it has no unit since switching probability has no unit. $D(ps)$ is the delay of the most critical path in $picoseconds$ (ps), $\mu(x)$ is the membership value, $T(s)$ is the total run time, and $Best(s)$ is the execution time in seconds for reaching the best solution. In both TS and GA each run consists of 10, 000 iterations or generations.

The results shown are the best case results obtained after the tuning of various algorithmic parameters of GA and TS (only one time for all circuits). In the case of GA the population size is 10, the crossover used is simple with a probability equal to 0.99, while for mutation it is 0.01. In case of TS, the size of neighborhood is also 10, while Tabu list size is chosen to be 0.1 the size of the circuit. From the results, it is clear that TS performed better than GA for most of the circuits in terms of the quality of the best solution as well as run time. In terms of quality of solution, TS consistently performs better, and the advantage of TS over GA gets emphasized when the size of the circuit gets bigger. Also execution time of GA increases significantly with the increase in circuit complexity. The higher execution time of GA can be attributed to its parallel nature i.e., a population of solutions is to be processed in each generation. Fig. 2 shows the performance of TS and GA against execution time in seconds. It is clearly noticed that TS is by far faster and of better final quality. Fig. 3 and Fig. 4 show the trend of solution's (a) cutset, (b) delay, (c) power, (d) balance, (e) average fitness, (f) best fitness for GA and TS respectively, in case of circuit S12307. It is clear from the shown plots that TS achieves a membership that is better than that reached by GA.

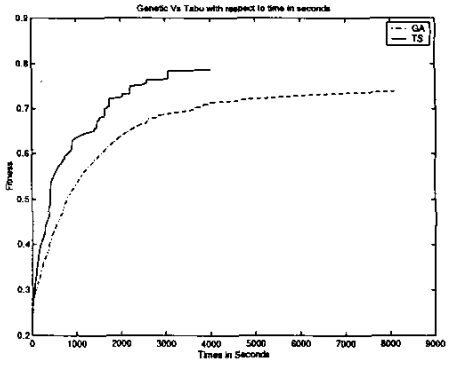

Figure 2: Comparison between GA and TS for the circuit S13207 with respect to execution time in seconds.

4. CONCLUSIONS

In this paper, two multiobjective optimization iterative algorithms namely GA and TS for VLSI partitioning were proposed. Fuzzy logic is used to integrate the objectives namely power, delay, cutset and balance into a scalar cost value. It is clear from the results that TS outperforms GA in terms of final solution costs and execution time, and the difference gets higher with the increase in circuit complexity. The higher execution time of GA can be contributed to its parallel nature i.e., a population of solutions is to be processed in each generation. The superiority of TS can be attributed

Table 1: Comparison between costs of the best solutions generated by GA and TS

	GA						TS					
Circuit	D (ps)	Cut	P(sp)	$\mu(x)$	T(s)	Best(s)	D (ps)	Cut	P(sp)	$\mu(x)$	T(s)	Best(s)
S298	233	19	1013	0.79	123	43	197	24	926	0.81	62	21
S386	356	36	1529	0.75	163	151	366	30	1426	0.76	82	77
S641	1043	45	2355	0.83	1868	1540	889	59	2281	0.85	939	818
S832	444	45	3034	0.68	289	276	446	50	2731	0.682	148	80
S953	526	96	2916	0.69	618	182	466	99	2518	0.734	313	225
S1196	396	123	5443	0.76	375	373	301	106	4920	0.801	184	134
S1238	475	127	5713	0.72	397	365	408	79	4597	0.75	187	160
S1488	571	104	5648	0.71	1238	1183	528	98	5529	0.72	616	405
S1494	614	102	5474	0.70	1228	1040	585	101	5339	0.71	616	427
S2081	302	26	787	0.73	94	32	225	17	770	0.79	47	16
S3330	571	299	10358	0.75	2096	2074	533	295	10298	0.79	1078	994
S5378	587	573	18437	0.74	2687	2686	590	430	16527	0.79	1338	1100
S9234	1313	1090	38149	0.72	5963	5949	1052	918	34055	0.81	2992	2821
S13207	1399	1683	45611	0.74	8098	8097	843	1332	41114	0.79	4001	3690
S15850	1820	2183	51747	0.74	10214	10206	1411	1671	47480	0.831	5131	5130

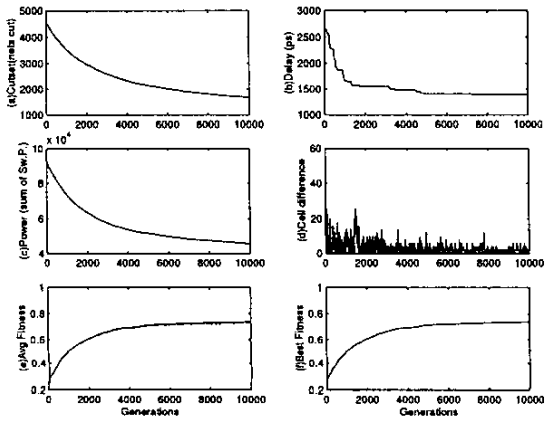

Figure 3: Performance of Ga for the circuit s13207.

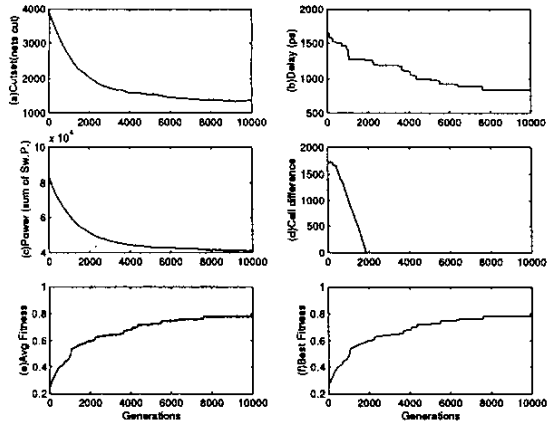

Figure 4: Performance of TS for the circuit s13207.

to its directed search approach and its higher greediness tendency as compared with GA to obtain a good solution.

Acknowledgment: The authors thank King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia, for support, under project #: COE/ITERATE/221

5. REFERENCES

[1] Sadiq M. Sait and Habib Youssef. VLSI Physical Design Automation: Theory and Practice. *McGraw-Hill Book Company, Europe*, 1995.

[2] K. Shahookar and P. Mazumder. VLSI Cell Placement Techniques. *ACM Computing Surveys*, 2(23):143–220, June 1991.

[3] M. Pedram. CAD for Low Power: Status and Promising Directions. *IEEE International Symposium on VLSI Technology, Systems and Applications*, pages 331–336, 1995.

[4] A. Ghosh, S. Devadas, K. Keutzer, and J. White. Estimation of Average Switching Activity in Combinational and Sequential Circuits. *Design Automation Conference*, pages 253–259, 1992.

[5] H. Vaishnav and M. Pedram. Delay optimal partitioning targeting low power VLSI circuits. *IEEE Trans. on Computer Aided Design*, 18(6):298–301, june 1999.

[6] R. R. Yager. On Ordered Weighted Averaging Aggregation Operators in Multicriteria Decisionmaking. *IEEE Transaction on Systems, MAN, and Cybernetics*, 18(1), January 1988.

[7] Sadiq M. Sait and Habib Youssef. *Iterrative Computer Algorithms with Applications in Engineering: Solving Combinatorial Optimization Problems*. IEEE Computer Society Press, California, December 1999.

[8] J. P. Cohoon and W. D. Paris. Genetic placement. *IEEE Trans. on CAD*, pages 956–964, 1987.