Evolutionary Algorithms for VLSI Multiobjective Netlist Partitioning

Sadiq M. Sait, Aiman H. El-Maleh, Raslan H. Al-Abaji
({sadiq, aimane, raslan}@ccse.kfupm.edu.sa)
King Fahd University of Petroleum & Minerals, Computer Engineering, Dhahran - 31261, Saudi Arabia

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 three iterative heuristics for the optimization of VLSI netlist bi-Partitioning. These heuristics are based on Genetic Algorithms (GAs), Tabu Search (TS) and Simulated Evolution (SimE). Fuzzy rules were incorporated in order to handle the multiobjective cost function. For SimE, fuzzy goodness functions are designed for delay and power, and proved efficient. A series of experiments are performed to evaluate the efficiency of the algorithms. ISCAS-85/89 benchmark circuits are used and experimental results are reported and analyzed to compare the performance of GA, TS and SimE.

Keywords: Genetic Algorithms, Tabu Search, Simulated Evolution, multiobjective, Fuzzy Logic, Netlist partitioning.

1. Introduction

VLSI circuit design has various objectives. Until the beginning of this decade, two main objectives of VLSI circuit design were the minimization of cutset and the improvement of timing performance. A large number of efforts targeting either one (especially cutset) or both of the above objectives are reported in the literature (Sait, 1995; Toulouse, 2002). 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 (Sait, 1995).

As different techniques are applicable and have been reported at different steps of the VLSI design process (Pedram, 1995), few performance-driven partitioning techniques at physical-level design exist in the literature. Therefore, the need for a system which incorporates all the three aspects of the design process (delay, cut, power) is increasing. In standard CMOS VLSI circuits, switching activity of circuit nodes is

responsible for most of the power dissipation. It is reported in (Kuroda, 2001) that this switching activity contributes 90% to the total power dissipation in the circuit. Therefore, most of the reported techniques focus on this aspect (Devadas, 1995).

For the partitioning phase, two low-power oriented techniques based on Simulated Annealing (SA) algorithm have recently been presented in (Choi, 1999). An enumerative optimal delay partitioning algorithm targeting low power is proposed by Vaishnav et al. in (Vaishnav, 1999). A circuit partitioning algorithm under path delay constraints is proposed by Tetsushi et al. in (Tetsushi, 1998). In this work, we address the problem of optimizing delay, power and cutset in the partitioning step at the physical level. Three iterative approaches based on Genetic Algorithm (GA), Tabu Search (TS) and Simulated Evolution are presented to solve the multiobjective optimization problem of partitioning. The following section gives a brief overview of methods for solving multi–objective problems, the details for which can be found in (Sait, 1999)

1.1. Methods for Solving Multi–Objective Problems

Ad-Hoc Weights
Historically, multiple objectives have been combined into a scalar objective function, usually through a linear combination (weighted sum) of the multiple attributes, or by turning objectives into constraints. One way is to assign a constant weight to each of the multiple objective functions. The weight assigned will depend on the importance of the objective. Assuming that all objectives are to be maximized, the fitness of an individual (solution) can be expressed as

$$f(x) = w_1 \cdot f_1(x) + w_2 \cdot f_2(x) + \cdots + w_n \cdot f_n(x) \qquad (1)$$

Where x is a string, n is the number of objective functions, $f(x)$ is a combined fitness function, $f_i(x)$ is the ith objective, and w_i is the weight of the ith objective. The problem with multi-objective functions is the difficulty in determining suitable weights. This is because, in most practical problems, no two objectives are related.

Pareto Optimality
A notion of optimality that respects the integrity of each of the separate criteria is the concept of Pareto optimality. Here, suppose we wish to *minimize* two objectives, expressed as f_1 and f_2. Let A, B, C, D, E, and F, be six possible solutions to our optimization problem, with the following fitnesses:

A : (10, 90) B : (20, 70) C : (08, 75)
D : (15, 60) E : (09, 65) F : (14, 63)

That is, solution A has a value of f_1=10 and f_2=90. If we plot the 6 points f_1 versus f_2, obviously those that are lower and on the left are regarded as the best. Points C and D are good choices since there are no points better than these in both the criteria. C is best with respect to f_1 and D with respect to f_2. On the other hand, A and B are poor choices. Solution A(10,90) is dominated by solution C(08,75), since $10 > 8$ and $90 > 75$. (If any solution p is to the right and top of another solution q, then we say p is dominated by q.) A is also dominated by E. Similarly, B(20,70) is dominated by D(15,60), E(09,65) and F(15,60). The set of solutions that are not dominated by any other solution is {C, D, E, F}. In this problem, as in any other multi-objective optimization problem, such a set of solutions comprises the Pareto-optimal (P-optimal) set. It is from this set that the decision maker has to make a choice. The Pareto optimality concept does not assist in making a *single* choice.

VEGA

The concept of Pareto optimality has been applied to solve multi-objective optimization problems using genetic algorithms. In VEGA, the population is divided into equally sized, disjoint sub-populations, each governed by a different objective function. Selection is performed independently of each criteria; however crossover is performed across sub-population boundaries. The problem with this scheme is, independent selection of best solution in each criterion results in potential bias against middle solutions (such as E and F in our case). That is, those which are good but not the best with respect to any single criterion.

VEGA mostly finds extreme solutions on the Pareto front. Schaffer suggested two approaches to improve VEGA. One is to provide a heuristic selection preference for non-dominated individuals in each generation. The other is a cross-breeding among the "species" by adding some mate selection.

MOGA

Recently, Murata and Ishibuchi proposed a Multi-Objective GA (MOGA) (Murata, 1995) which uses a weighted sum of multiple objective functions to combine them into a scalar fitness function. The key feature of MOGA is that the weights attached to the multiple objective functions are *not* constant but randomly specified for each selection. Therefore, the direction of search in MOGA is not fixed. Weights are chosen as follows:

$$w_i = \frac{random_i(\cdot)}{\sum_{j=1}^{n} random_j(\cdot)} \tag{2}$$

where $random_j(\cdot)$ is a non-negative random number. Note that w_i is a real number in the closed interval [0,1].

During the execution of MOGA, a tentative set of Pareto optimal solutions is stored and updated at every generation. A certain number (say N_{elite}) of individuals are randomly selected from the set at each generation. These solutions are used as elite individuals in MOGA. This elite preserve strategy has the effect in keeping the variety of each population.

The sequence of steps used in MOGA are as follows. Following the generation of initial population containing M strings, the values of the objective functions for the generated strings are calculated, and a tentative set of Pareto optimal solutions is updated. The fitness (Equation 1) of each string is then calculated using the random weights given in Equation 2. Next, pairs of strings are selected with a certain selection probability. The selection probability of string x in a population Ψ, denoted by $P(x)$, is specified as

$$P(x) = \frac{f(x) - f_{\min}(\Psi)}{\sum_{x \in \Psi}\{f(x) - f_{\min}(\Psi)\}} \tag{3}$$

where $f_{\min}(\Psi) = \min\{f(x) | x \in \Psi\}$. This step is repeated until $\frac{\mathrm{M}}{2}$ pairs of strings are selected from the current population. Then, following crossover and mutation, N_{elite} strings from the set of M strings generated by the previous operations are removed and replaced with N_{elite} strings randomly selected from a tentative set of P-optimal solutions. This process continues until a pre-specified set of stopping conditions is satisfied. MOGA returns a set of Pareto-optimal solutions to the decision maker. The best solution is then selected according to the decision makers preference (Murata, 1995).

Justification for Adopting Fuzzy Approach

For many problems, two distinct forms of problem knowledge exist:

1. Objective knowledge which is used a lot in engineering problem formulations (e.g., mathematical models, etc.),

2. Subjective knowledge which represents linguistic information that is usually impossible to quantify (e.g., rules, expert information, etc.).

Subjective knowledge is always ignored at the front end of engineering designs; but it is frequently used to evaluate such designs. The two forms of knowledge can be coordinated in a logical way using fuzzy logic. The motivation for using fuzzy logic can be readily summarized by the principle of incompatibility, as stated by Zadeh (Zadeh, 1965):

The closer one looks at a real world problem, the fuzzier becomes its solution. Stated informally, the essence of this principle is that as the complexity of a system increases, our ability to make precise and yet significant statement about its behavior diminishes until a threshold is reached beyond which precision and significance become almost mutually exclusive characteristics

The key elements in human thinking are not numbers, but labels of fuzzy sets, i.e., classes of objects in which the transition from membership to non-membership is gradual rather than abrupt. Indeed the persuasiveness of fuzziness in human thought processes suggests that much of logic behind human reasoning is not the traditional two-valued or even multi–valued logic, but a logic with fuzzy truth, fuzzy connectives, and fuzzy rules of inference. Actually, fuzzy logic has come to age. Its foundations have become firmer, its applications have grown in number and variety, and its influence within the basic sciences has become more visible and more substantive. Fuzzy Logic plays a pivotal role in computing with words. The computation with words finds its motivations when the available information is too imprecise to justify the use of numbers and also when there is a tolerance for imprecision which can be exploited to achieve tractability, robustness, low solution cost, and better rapport with reality.

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 cutset 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

(clusters) 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:

$$f = \sum_{e \in \psi} w(e) \tag{4}$$

where $\psi \subset E$ denotes the set of off-chip edges, i.e., nets cut. 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:

$$f = |\psi| \tag{5}$$

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 $\Phi : (V_A, V_B)$, 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. To optimize delay, we need to minimize the following function:

$$d(p_{ij}) = \left(\sum_{v_i \in V(p_{ij})} d(v_i) \right) + d_c \times ncut(p_{ij}) \tag{6}$$

Power: The average dynamic power consumed by a CMOS logic gate

in a synchronous circuit is given by:

$$P_i^{average} = 0.5 \frac{V_{dd}^2}{T_{cycle}} C_i^{load} N_i \tag{7}$$

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 (Ghosh et. al, 1992) under a zero delay model. C_i^{load} in Equation 7 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:

$$P_\zeta = \beta \frac{v_{dd}^2}{T_{cycle}} \sum_{i \in \zeta} (C_i^{basic} + C_i^{extra}) N_i \tag{8}$$

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}.

The power model given in Equation 8 can be further simplified. It is assumed that the power dissipation contribution due to variations of C_i^{basic} under different partitioning solutions is negligible. Furthermore, considering that the fixed overhead capacitance for an external net is dominant within C_i^{extra}, it can be assumed that C_i^{extra} is identical for each net. This leads to the following objective function (Vaishnav, 1999).

$$O_\zeta = \sum_{i \in \zeta_v} N_i \tag{9}$$

where ζ_v corresponds to the set of visible gates, i.e., the set of gates that drive a load external to the partition.

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 below:

$$\frac{|\beta_1 - \beta_2|}{\phi} \leq \alpha \tag{10}$$

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. When balance is used as cost, it will be $|\beta_1 - \beta_2|$.

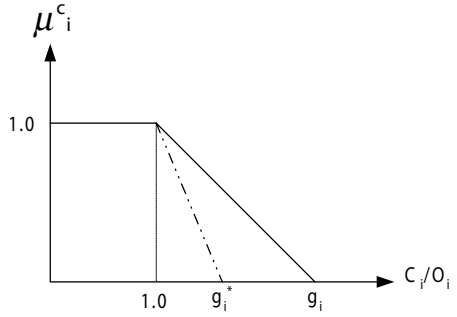

Figure 1. Membership functions

2.1. Overall Fuzzy Cost Function:

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

Rule 1: IF a solution has *Small cutset* **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 (Yager, 1998) and the membership $\mu(x)$ of a solution x in fuzzy set *good solution* is given as:

$$\mu^c_{pdcb}(x) = \beta^c \times min(\mu^c_p(x), \mu^c_d(x), \mu^c_c(x), \mu^c_b(x)) + (1-\beta^c) \times \frac{1}{4} \sum_{j=p,d,c,b} \mu^c_j(x) \qquad (11)$$

where $\mu^c(x)$ is the membership of solution x in fuzzy set of acceptable solutions, $\mu^c_{pdcb}(x)$ is the membership value in the fuzzy sets of " within acceptable *power*", "within acceptable *delay*", "within acceptable cutset" and "within acceptable *balance*" respectively. β^c is a 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, and Small cutset* are shown in Figure 1. 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 where $g_j \geq 1.0$.

The lower bounds O_j (shown in Fig. 1) for different objectives are computed as follows:

$$O_b = 1, \qquad \text{(to avoid divide by zero)} \tag{12}$$

$$O_p = \sum_{i \in \zeta} (C_i^{basic}) N_i \qquad \forall v_i \in \{v_1, v_2, ..., v_n\}, \tag{13}$$

$$O_d = \sum_{j=1}^{k} CD_j, \qquad \forall v_j \in \{v_1, v_2, ..., v_k\} \text{ in path } \pi_c, \tag{14}$$

$$O_c = 1. \tag{15}$$

where O_j for $j \in \{b, p, d, c\}$ are the lower bounds on the costs for balance, power dissipation, delay and cutset respectively, n is the number of nets in the circuit, CD_j is the switching delay of the cell j driving net v_j, N_i is the switching probability of net v_i, π_c is the most critical path with respect to optimal interconnect delays assuming that no net on this path is cut, k is the number of nets in π_c. The minimum power is obtained if no net is cut, which means substituting 0 for C_i^{extra} in Equation 8. The components of the goal vector G are calculated as follows:

$$g_{balance} = \frac{|\zeta|}{O_b} \tag{16}$$

$$g_{cut} = \frac{n}{O_c} \tag{17}$$

$$g_{delay} = \frac{Initial\ delay}{O_d} \tag{18}$$

$$g_{power} = \frac{Initial\ power}{O_p} \tag{19}$$

Where $|\zeta|$ is the number of cells in the circuit. Initial delay and initial power are the values computed from initial solutions.

3. Simulated Evolution (SimE) Approach

The Simulated Evolution algorithm (SimE) is a general search strategy for solving a variety of combinatorial optimization problems.

The pseudo-code of SimE is given in Figure 2 (Sait, 1999). SimE operates on a single solution, termed as *population*. Each population consists of elements. In case of the partitioning problem, these elements are cells to be moved. The algorithm has one main loop consisting of

three basic steps, Evaluation, Selection and Allocation. In the *Evaluation* step, *goodness* of each element is measured. Goodness of an element is a single number between '0' and '1', which is a measure of how near is the element from its optimal location. After that comes *Selection* which is the process of selecting those individuals which are unfit (badly placed) in the current solution. For that purpose, the goodness of each individual is compared with a random number (in the range [0,1]); if the goodness is less than the random number then it is selected. Allocation is the SimE operator that has the most impact on the quality of solution. Its main function is to mutate the population by altering the location of selected cells. The three steps are executed in sequence until no noticeable improvement to the population goodness is observed after a number of iterations or a prefixed number of iterations are completed. A higher value of goodness means that the element is near its optimal location. For single objective optimization, the goodness can be calculated as follows,

$$g_i = \frac{O_i}{C_i} \tag{20}$$

where O_i is the estimated optimal cost and C_i is the actual estimate of the cost.

In *Selection*, an individual having high goodness measure still has a non-zero probability of assignment to selected set. It is this element of non-determinism that gives **SimE** the capability of escaping local minima.

Cut Goodness: Let $V_i = \{v_1, v_2, ..., v_k\}$ be the set of nets connected to cell i, and U_i be a subset of V_i containing the connected nets to cell i that are cut. The goodness function for a cell is defined and computed as follows:

$$gc_i = \frac{d_i \ - \ w_i}{d_i} \tag{21}$$

Where d_i is the total number of nets connected to cell i (i.e., $|V_i|$), and w_i is the number of nets connected to cell i that are cut (i.e., $|U_i|$).

The cut goodness is simply the number of uncut nets over the total nets connected. Since gc_i is between 0 and 1, we can take the fuzzy membership μ_c as equal to the goodness $\mu_c = gc_i$. An example of goodness calculation is shown in Fig. 3; the goodness of cell 5 is calculated as follows: $gc_5 = \frac{3-2}{3} = 0.33$.

Power Goodness: The power goodness gp_i of a cell is defined as a measure of how well placed is the cell in its present block according to

ALGORITHM *Simulated_Evolution*(B, Φ_i, SC)
B = Bias Value. Φ = Complete Solution.
Φ_i = Initial Solution. SC = Stopping Criterion.
e_i = Individual link in Φ.
O_i = Lower bound on cost of i^{th} link.
C_i = Current cost of i^{th} link in Φ.
g_i = Goodness of i^{th} link in Φ.
S = Queue to store the selected links.
Allocate(e_i, Φ_i) allocates e_i in partial solution Φ_i.
Repeat

EVALUATION: **ForEach** $e_i \in \Phi$ **DO**
Begin
Evaluate g_i
End
SELECTION: **ForEach** $e_i \in \Phi$ **DO**
Begin
If random(1) $>$ min$(g_i + B, 1)$
Begin
$S = S \cup e_i$;
$\Phi = \Phi - e_i$;
End
End
sort(S)
ALLOCATION: **ForEach** $e_i \in S$ **DO**
Begin
Allocate(e_i, Φ_i)
End

Until *SC is satisfied*
Return (Best solution)
End *Simulated_Evolution*

Figure 2. Structure of the simulated evolution (SimE) algorithm.

power consumption and can be computed as follows:

$$gp_i = \frac{\sum_{j=1}^{k} S_j \ (j \ \in \ V_i) - \sum_{j=1}^{k} S_j \ (j \ \in \ U_i)}{\sum_{j=1}^{k} S_j \ (j \ \in \ V_i)} \tag{22}$$

S_j is the switching probability of the cell that drives the net. The goodness is equal to the sum of the switching probabilities of the cells that are driving the uncut nets over the sum of the switching probabilities of the cells that are driving all nets connected. In this way a cell is placed in the partition where the sum of the switching probability of the cut nets is optimized. Results show that this goodness function gives high quality solutions with less power dissipation. Since $0 \leq gp_i \leq 1$ we can take the fuzzy membership $\mu_p = gp_i$. An example of power goodness

Figure 3. Power and Cut Goodness Calculation.

calculation is shown in Fig. 3; the goodness of cell 5 is calculated as follows: $gc_5 = \frac{0.7-0.4}{0.7} = 0.428$.

The power and cutset objectives are possibly conflicting. Hence it is possible to find alternative solutions for a specific circuit. For example, there may exist a solution with high number of cuts and low power consumption (because the nets cut have less switching probability) and another with lower cuts and higher power consumption.

Delay Goodness: In our problem, we deal with multi-pin nets, which makes it hard to design a suitable and simple delay goodness function. We propose the following delay goodness:

$$gd_i = \frac{|K_i| - |L_i|}{|K_i|} \tag{23}$$

where gd_i is the delay goodness of cell i. We consider the set of the *critical paths* passing through i and define the set K_i as the set of all cells connected to these paths. We also define L_i as a subset of K_i, containing those cells which are connected to the critical paths passing through i and are not in the same block as i. This goodness function will tend to drive the cells that are connected by the critical path to the same block, thus minimizing the delay along the path. A cell is considered good in its block if the majority of cells connected to all critical paths passing through it are also placed in the block. An example for delay computation is given in Fig. 4. To calculate gd_4, we first compute $|K_4| = 5$ for the critical path {1,4,5,7,6} which is the only one connected to cell 4. $|L_4| = 3$ which are cells $\{1, 5, 7\}$. This

Figure 4. Delay Goodness Calculation.

gives $gd_4 = \frac{5-3}{5} = 0.4$. However, $gd_5 = 0.6$, and hence is better placed according to the delay consideration.

3.1. Proposed Fuzzy Evaluation Scheme and Selection

With the classical goodness of cut only, it is possible that a cell having a high goodness with respect to cut may not be selected even though it may have low goodness with respect to circuit delay and power. In order to overcome this problem, it is necessary to include power and delay in the goodness measure along with cut goodness. Also, it is not desirable to select all the cells even if they all have a low goodness value. In this case, it is desirable to select those cells which are far from their lower bounds as compared to other cells in the design. For this purpose, the following fuzzy rule is proposed.

Rule R2: (*as compared to other cells*)
IF cell i is
near its optimal Cutset goodness
AND
near its optimal power goodness
AND
near its optimal net delay goodness
OR
$T_{max}(i)$ *is much smaller than* T_{max}
THEN it has a high goodness.

T_{max} is the delay of the most critical path in the current iteration and

$T_{max}(i)$ is the delay of the longest path traversing cell i in the current iteration.

The membership function is illustrated in Fig. 5.

The above-mentioned fuzzy rule is interpreted as follows:

$$g_i = \mu_i(x) = \beta \times min(\mu_{ic}(x), \mu_{ip}(x), \mu_{id}(x)) + (1-\beta) \times \frac{1}{3} \sum_{j=c,p,d} \mu_{ij}(x) \quad (24)$$

where

$$\mu_{id}(x) = \beta_d \times max(\mu_{dg}(x), \mu_{ipath}(x)) + (1-\beta_d) \times \frac{1}{2}(\mu_{dg}(x) + \mu_{ipath}(x)) \quad (25)$$

The superscript e is used here to represent evaluation so that other fuzzy notations in other steps of SimE can be distinguished. The term x represents the block of cell i, $\mu_i(x)$ is the membership in the fuzzy set of high goodness and g_i is the goodness of cell i. β and β_d are constants between 0 and 1 to control OWA operators. $\mu_{ic}(x)$ and $\mu_{ip}(x)$ represent the membership in fuzzy sets of near optimum cutset and near optimum power as compared to other cells. Moreover, $\mu_{id}(x)$ is the overall delay goodness, and represents the membership in fuzzy set of near optimum timing performance. It is obtained after applying "OR-like" OWA to $\mu_{dg}(x)$ and $\mu_{ipath}(x)$, which are the memberships in fuzzy sets of near optimum cell delay goodness as compared to other cells and $T_{max}(i)$ (most critical path passing trough cell i) is much smaller than T_{max} (current most critical path of the circuit). $\mu_{ipath}(x)$ is included in the computation of $\mu_{id}(x)$ because if a cell is not in the critical path then it must have high goodness with respect to delay objective. After considerable number of iterations, it is possible that a cell is in the critical path but is also very near to its optimal delay goodness. In that case, it is not possible to optimize it further. At this stage, $\mu_{dg}(x)$ overrides $\mu_{ipath}(x)$.

The base values for cutset and power are not needed since the membership is directly computed as described in Section **??**. As for cell delay goodness it is composed of net delay $\mu_{dg}(x)$ which is computed directly by using Equation 23. For computing $\mu_{ipath}(x)$ we define base value $X_{ipath}(x)$ for fuzzy set $\{$ $T_{max}(i)$ much smaller than $T_{max}\}$, and is computed in Equation 26:

$$X_{ipath}(x) = \frac{T_{max}}{T_{max}(i)} \quad (26)$$

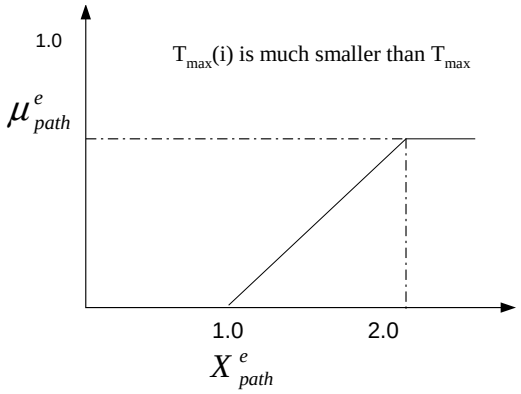

Figure 5. Membership function for $T_{max}(i) \ll T_{max}$.

Experimentally we found that a base value of 2 is suitable to quantify that $T_{max}(i)$ is much smaller than T_{max}. In our implementation, the Biasless Selection scheme proposed by Khan et al in (Khan et. al, 2002) is used. The selection bias B is totally eliminated and a cell is selected if $Random > goodness_i$.

4. Genetic Algorithm (GA) and Tabu Search (TS)

In this section, implementation details of the genetic algorithm and tabu search for solving the multiobjective partitioning problem 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.

4.1. 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 a 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 (Sait, 1999). For description of other crossover operators see (Sait, 1999; Sipakoulis99, 1999; Abaji, 2002). In this implementation, the simple crossover is used.

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 (Sipakoulis99, 1999). In our work, two mutation variations 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 multiobjective optimization problem to minimize three mutually conflicting objectives, fuzzy membership functions and fuzzy rules are used for evaluating the fitness of a 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 Equation 11. 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 a balance between greediness and randomness.

4.2. Tabu Search Approach

Tabu Search (TS) is one of the most popular iterative heuristics and there have been many efforts involving application of TS to the partitioning problem (Hammami, 2003). 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 (Sait, 1999). 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.

5. Experimental results

The results obtained from GA and TS are compared in terms of the overall quality of the best solution and run time in Table I. $P(sp)$ 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, $Best(s)$ 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). The details of these algorithmic parameters and their fine tuning are discussed in (Sait, 2003). 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, and the advantage of TS over GA gets emphasized when the size of the circuit gets bigger. Also the 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. 6 shows the performance of TS and GA against execution time in seconds for the circuit S13207. It is clearly noticed that TS is by far faster and of better final quality. Fig. 7 and Fig. 8 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 S13207. It is clear from the shown plots that TS achieves a membership that is better than that reached by GA.

Comparing SimE to GA and TS, as can be seen from Table 1, SimE achieves significantly better results for most of the circuits. It achieves a higher fitness value in 12 of the 15 circuits than GA or TS. For the circuits S3330, S5378, S9234, and S15850, the values achieved for delay, cutset, and power are significantly better. Figure 6 shows the performance of SimE versus TS and GA with respect to time for the circuit S13207. Clearly, SimE achieves a higher quality solution in much less time. Figure 9 shows the trend of solution (a) cutset, (b) delay, (c) power, (d) cells selected, (e) average cells goodness, and (f) best fitness for the circuit S13207. As can be observed, the cutset, delay, and power reduce at a much faster rate than TS and GA. Also the number of cells selected becomes smaller which indicates that the cells are getting better assigned as the algorithm progresses.

A synthesis algorithm is proposed for the design of low-power combinational circuits under area constraints in (Choi, 1999). Partitioning is performed through an adaptive simulated annealing algorithm, employing a cost function modeled for low-power consumption under given area constraints. Experiments have been performed for the MCNC benchmark circuits using the power analysis package provided in the the Synopsys Design Analyzer. Results show that the proposed algorithm generates circuits which consume less power than those by the area-optimization package in Synopsys Design Analyzer and precomputation algorithm.

An adaptive Genetic algorithm for VLSI circuit partitioning and another for VLSI circuit placement are presented in (Sipakoulis99, 1999). These Genetic algorithms are able to modify some of their own parameters during the search, based on their performance. The algorithms are applied to partitioning and placement of a circuit, respectively, and their performance is compared with the performance of a non-adaptive Genetic algorithm. The proposed Genetic algorithms lead to significantly superior solutions in less computation time.

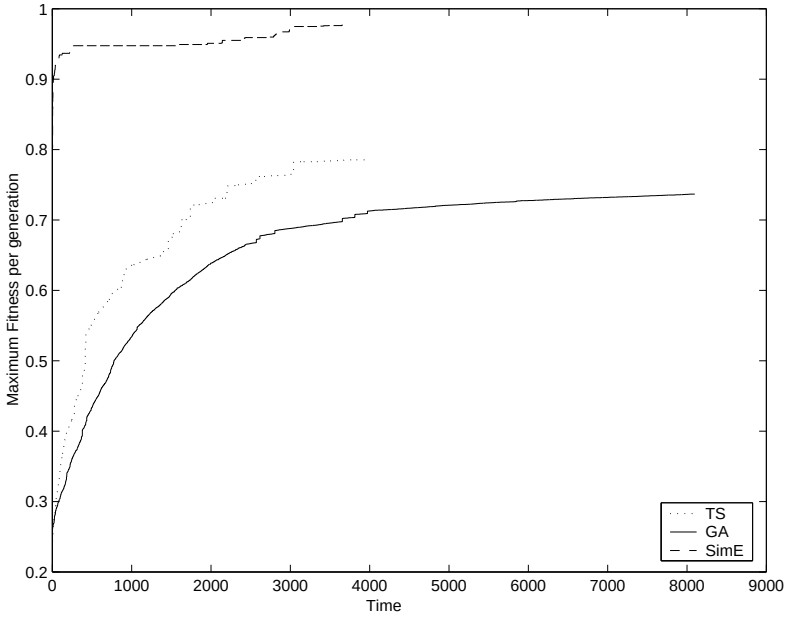

Figure 6. Multiobjective SimE, GA and TS performance for the circuit S13207 against time.

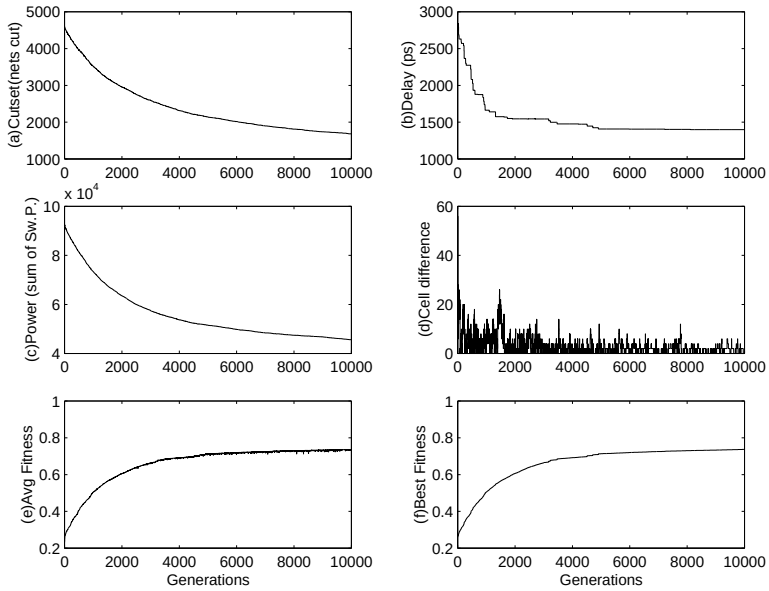

Figure 7. Performance of GA for the circuit s13207.

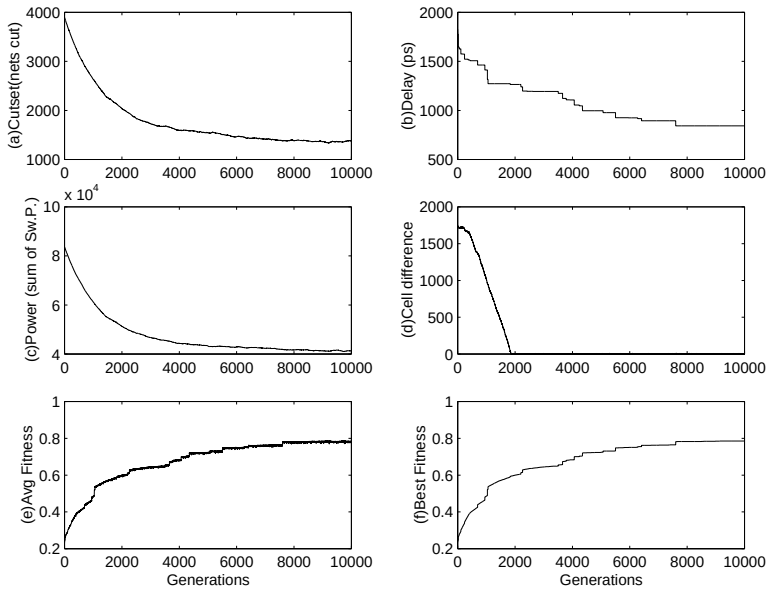

Figure 8. Performance of TS for the circuit s13207.

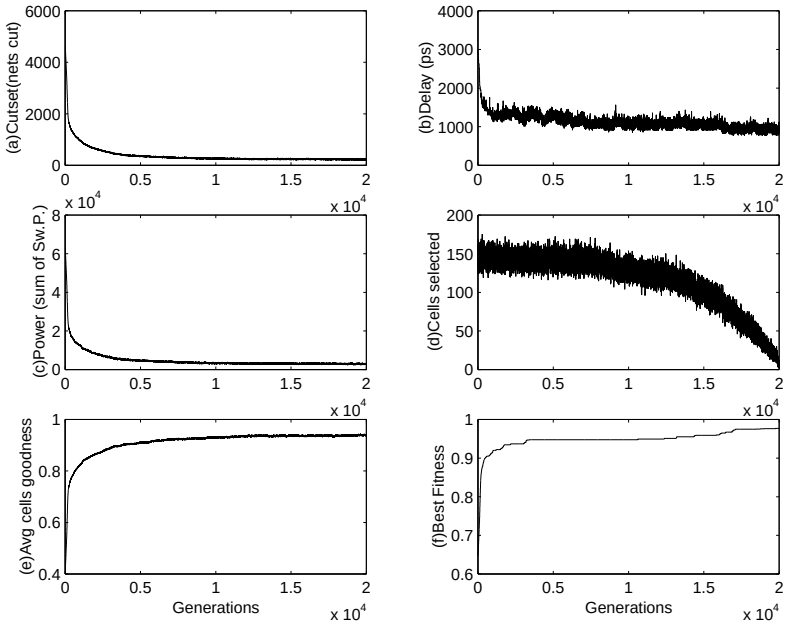

Figure 9. Multiobjective SimE performance for the circuit S13207.

6. Conclusions

In this paper, iterative algorithms for multiobjective optimization namely GA, TS and SimE for VLSI partitioning were proposed. Fuzzy logic is used to integrate the objectives namely power, delay, cutset and balance into a scalar cost value. Fuzzy goodness functions where developed for SimE. 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 superiority of TS can be attributed to its directed search approach and its higher greediness tendency as compared with GA to obtain a good solution. For most of the circuits, SimE achieved significantly better results than TS and GA. For the large circuits, the superiority of SimE in achieving higher quality solutions is highlighted. This is attributed to the smart strategy of the algorithm in selecting badly assigned cells and attempting to assign them in better partitions.

Acknowledgment

The authors thank King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia, for support, under Project code COE/ITERATE/221.

References

Al-Abaji, R.H. Evolutionary Techniques for Multi-Objective VLSI Netlist Partitioning. M.S. Thesis, King Fahd University of Petroleum and Minerals, Dhahran, 2002.

Ackley, D. H.. A Connectionist Machine For Genetic Hillclimbing. *Kluwer*, 1987.

Al-Abaji, R. H.. Evolutionary Techniques for Multi-objective VLSI Netlist Partitioning. Master's thesis, King Fahd University of Petroleum and Minerals, Dhahran, Kingdom of Saudi Arabia, May 2002.

Choi, I.S. and S.Y. Hwang. Circuit Partitioning algorithm for Low-Power Design Under Area Constraints Using Simulated Annealing. *IEE Proc. Circuits Devices Systems*, 146(1):8–15, February 1999.

Devadas, S. and Sharad Malik. A Survey of Optimization Techniques Targeting Low Power VLSI Circuits. *32nd ACM/IEEE Design Automation Conference*, 1995.

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

Hammami, M. Ghedira, K. Tabu search for the k-graph partitioning problem Computer Systems and Applications, 2003. Book of Abstracts. ACS/IEEE International Conference on, Vol., Iss., 14-18 July 2003 Pages: 85

Khan, Junaid A., Sadiq M. Sait, and Mahmood R. Minhas. Fuzzy Biasless Simulated Evolution for Multiobjective VLSI Placement. *IEEE CEC 2002, Hawaii USA*, 12-17 May 2002.

Benchmark		Simulated Evolution SimE					Genetic Algorithm					Tabu Search				
Circuit	Size	D (ps)	Cut	P(sp)	$\mu(x)$	$T_{Best(s)}$	D (ps)	Cut	P(sp)	$\mu(x)$	$T_{Best(s)}$	D (ps)	Cut	P(sp)	$\mu(x)$	$T_{Best(s)}$
S298	136	197	11	837	0.95	62	233	19	1013	0.79	43	197	24	926	0.81	21
S386	172	393	28	1696	0.74	152	356	36	1529	0.75	151	386	30	1426	0.76	77
S641	433	886	16	1738	0.98	966	1043	45	2355	0.83	1540	889	59	2281	0.85	818
S832	310	400	39	3132	0.691	257	444	45	3034	0.68	276	446	50	2731	0.682	80
S953	440	476	48	2473	0.93	249	526	96	2916	0.69	182	466	99	2518	0.734	225
S1196	561	415	78	5488	0.82	398	396	123	5443	0.76	373	301	106	4920	0.801	134
S1238	540	350	77	5960	0.73	205	475	127	5713	0.72	365	408	79	4597	0.75	160
S1488	667	612	83	5892	0.7	716	571	104	5648	0.71	1183	528	98	5529	0.72	405
S1494	661	502	71	6250	0.81	802	614	102	5474	0.70	1040	585	101	5339	0.71	427
S2081	122	325	13	706	0.94	89	302	26	787	0.73	32	225	17	770	0.79	16
S3330	1962	394	46	8431	0.98	812	571	299	10358	0.75	2074	533	295	10298	0.79	994
S5378	2994	554	161	14094	0.95	465	587	573	18437	0.74	2686	590	430	16527	0.79	1100
S9234	5845	831	196	25672	0.98	3853	1313	1090	38149	0.72	5949	1052	918	34055	0.81	2821
S13207	8652	1014	313	35014	0.98	3129	1399	1683	45611	0.74	8097	843	1332	41114	0.79	3690
S15850	10384	1189	416	40716	0.96	1850	1820	2183	51747	0.74	10206	1411	1671	47480	0.831	5130
		Average $\mu(x)$ 0.876					Average $\mu(x)$ 0.736					Average $\mu(x)$ 0.774				

Table I.Comparison between SimE, GA and TS.
Size: indicates the size of the particular benchmark circuit in terms of the number of cells.

Kuroda, T. CMOS design challenges to power wall. Microprocesses and Nanotechnology Conference, 2001 International, Vol., Iss., 2001 Pages:6-7

Murata, T. and Ishibuchi, H. MOGA multi–objective genetic algorithms. *Proceedings of International Conference on Evolutionary Computation*, pages 289–294, 1995.

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

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

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

Sait, S.M. and El-Maleh, A.H. and Al-Abaji, R.H. General iterative heuristics for VLSI multiobjective partitioning. Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on, Vol.5, Iss., 25-28 May 2003 Pages: V-497- V-500 vol.5

Sipakoulis, G.C. Karafyllidis, I. Thanailakis, A. Genetic partitioning and placement for VLSI circuits. Electronics, Circuits and Systems, 1999. Proceedings of ICECS '99. The 6th IEEE International Conference on, Vol.3, Iss., 1999 Pages:1647-1650 vol.3

Tetsushi, J. M. and Koide S. W.. A Circuit Partitioning Algorithm Under Path Delay Constraints. *IEEE*, pages WT32–1.1 WT32–1.4, 1998.

Ouyang, M. and Toulouse, M. and Thulasiraman, K. and Glover, F. and Deogun, J.S. Multilevel cooperative search for the circuit/hypergraph partitioning problem. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, Vol.21, Iss.6, Jun 2002 Pages:685-693

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

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

Zadeh, L. A. Fuzzy sets. *Information Contr.*, 8:338-353, 1965.