Generic Fuzzy Logic CAD Development Tool

Eric Q. Kang

Dept. of Computer Sciences
University of Minnesota
Minneapolis, MN 55455
Tel: 612-623-4269
Fax: 612-831-0887
e-mail: qkang@cs.umn.edu

Eugene Shragowitz

Dept. of Computer Sciences
University of Minnesota
Minneapolis, MN 55455
Tel: 612-625-3368
Fax: 612-625-0572
shragowi@cs.umn.edu

Abstract— This paper describes a generic fuzzy logic CAD development tool and reports on application of it to some important CAD problems. This menu-based tool allows to introduce linguistic variables in a textual and graphic form by clicking a menu. It permits users to define membership functions in an analytical, table or graphical forms. It connects linguistic variables by fuzzy logic operators to create fuzzy logic and generates their graphical representations. [1]

I. Introduction

Fuzzy logic, first applied to control problems, is now used in many new domains. One of the new areas of application of fuzzy logic is CAD for electronic systems. Several applications of fuzzy logic in CAD were recently reported [8], [4]. In the process of development of CAD applications, we observed that creation of fuzzy logic decision-making structures can be greatly improved and accelerated by a specialized tool that allows easy manipulation of major blocks involved in construction of fuzzy logic decision-makers. Requirements for such tool are very straightforward:

1. Easy definition of fuzzy logic linguistic variables.
2. Simple graphical and analytical definitions of membership functions.
3. Simple ways to define fuzzy logic operations and set up fuzzy logic rules.
4. Easy-to-support hierarchical structures of decision-makers.
5. Verification of consistency.
6. Easy to port or incorporate the designed graphical decision-makers to host programs.

[1] This work is supported in part by the National Science Foundation under Grant MIP-9123945.

Guided by these goals, we developed a generic fuzzy logic development tool that enables users to develop decision-making mechanism for their applications and then incorporate it into any CAD tool under development. It can be stressed that applications of fuzzy logic decision-maker are not limited to specifically designed fuzzy logic applications. Fuzzy logic decision-makers can be included into existing tools to improve their flexibility in selecting and balancing of different goals, which receive numerical values from traditional algorithms. Moreover, in our own applications, fuzzy logic is mostly used to make decisions over numerical data.

II. Fuzzy rules and operations

A. Fuzzy logic operations

All logic operations in classical logic can be extended to fuzzy logic. But there are various ways of doing that. In this paper, fuzzy logic operations **and**, **or** and **not** will be defined as follows.

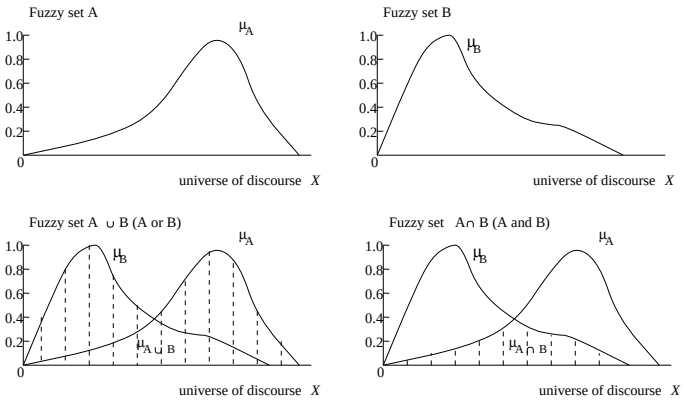

Fig. 1. Fuzzy sets and fuzzy logic operations

If A and B are two fuzzy sets, then A **and** B (also denoted as AB) is a fuzzy set with a membership function produced by the function min of membership functions

of A and B, i.e., $\mu_{AB}(x) = min(\mu_A(x), \mu_B(x))$. Similarly, A **or** B (also denoted as $A \cup B$) is defined as a fuzzy set with a membership function realized by the function max of membership functions of A and B, i.e., $\mu_{A\cup B}(x) = max(\mu_A(x), \mu_B(x))$. (see Figure 1). Negation **not** is extended directly, i.e., the membership function $\mu_{\ A}$ for **not**(A) is given as $\mu_{\ A} = 1 - \mu_A$.
By using linguistic values, linguistic variables and fuzzy logic operations, fuzzy logic rules can be constructed in a way similar to reasoning used by human beings. For example, if one wants to place cells by utilizing connectivity information and timing information, one can have a rule like "if a cell η has *critical timing* and *strong connectivity* to the partial placement, then it is a *good candidate* to be placed at a current position". Here, *strong connectivity* and *critical timing* are linguistic values of linguistic variables *connectivity* and *timing*, respectively. *Good candidate* is a linguistic value of the linguistic variable *candidate*. *Strong connectivity*, *critical timing* and *good candidate* are implemented as fuzzy sets, some specific examples of applications of these linguistic values can be found later in this paper. The word "and" in this rule is the fuzzy logic operation implemented according to the above definition.

B. Fuzzy logic rules

Linguistic variables and fuzzy logic operations can be used to construct fuzzy logic rules. In the following, two examples of fuzzy logic rules and their numerical evaluations are given.

> **Rule 1:**
> **If** a net's *driver* is a *strong driver*, **then** it can drive a *large load*.

Here *driver* and *load* are two linguistic variables. The possible linguistic values for a variable *driver* can be *weak driver, driver with driving capability of about 1.5*ns/pF, *strong driver* etc., and the possible linguistic values for *load* can be *small load, average load, large load*.
The **if** part of **Rule 1** is called a predicate, while the **then** part is called an action. In this application, the action part is simply implemented as an identification function unless it is explicitly specified. Therefore the result of evaluation of the predicate part is identified as a degree of membership for the linguistic value used in the action part.
Let $\mu_{strong\ driver}(\cdot)$ be the membership function for linguistic value *strong driver* (Figure 2). Suppose the driving capability of a driver is 1.4ns/pF, and let L be the load this driver can drive according to **Rule 1**. Since no fuzzy logic operation is involved in the predicate part of **Rule 1**, the degree of membership of the *driver* of being a member of fuzzy set *strong driver* is identified as the degree of membership of L in the fuzzy set *large load*. Thus

$$\mu_{large\ load}(L) = \mu_{strong\ driver}(1.4) = 0.75,$$

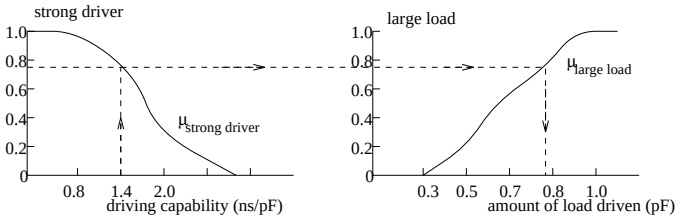

Fig. 2. Evaluation of **Rule 1**

and so $L = 0.78$.
Connectives such as *and* and *or* can be used in approximate reasonings to join two or more linguistic values.

III. Generic hierarchical fuzzy logic structures for decision-making

In this section, generic hierarchical fuzzy logic structures for decision-making are described. To take the advantage of Object-Oriented Programming (OOP) technology, the basic block is designed to be a basic OOP class, which can be extended to form more specific blocks (classes) with special properties. These blocks can be

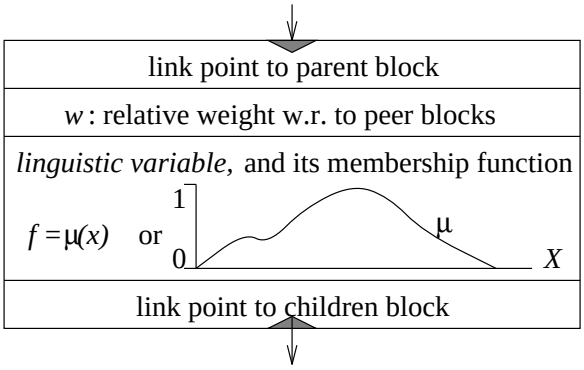

Fig. 3. Basic block of fuzzy logic decision-making structure

instantiated easily to form a hierarchical fuzzy logic decision-making structure. Figure 3 shows the graphical representation of the internal elements of a basic block. As one can see, a basic block is basically a linguistic variable, which represents either an objective or a criterion. The membership function for a linguistic variable can be given either by an analytical formula or in a table form which is then interpolated into a piecewise linear function. Users can customize the linguistic variable and its membership function for each basic block. If the link point to a parent for a basic block is not used, then the block is at the highest level, similarly if the link point to children is not used, then this block is at the lowest level of the hierarchy. Otherwise, both link points will be used.
Figure 4 illustrates a hierarchical fuzzy logic decision-maker generated from several basic blocks. For sim-

plicity, the internal structures of these basic blocks are not shown. In the following graphical representations of decision-makers, each block represents a linguistic value and is implemented as an OOP object. Depending on the

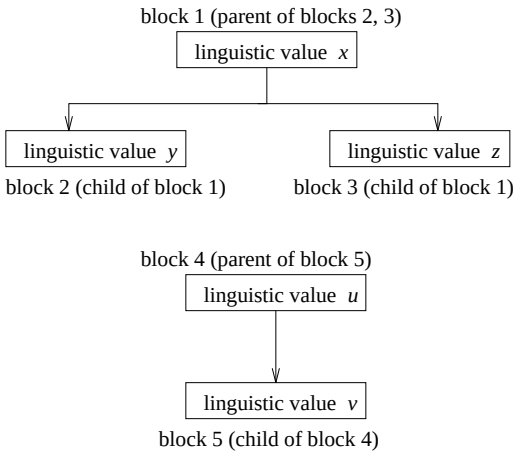

Fig. 4. Examples of hierarchical fuzzy logic decision-making structures

context, a linguistic value can be either an objective or a criterion used to define an objective. For example, a linguistic value x can be the objective *good placement*, a linguistic value y can be the criterion *strong connectivity to partial placement* and a linguistic value z can be the criterion *almost within a feasible interval*. Fuzzy logic rules are used to define relations between a linguistic value of outcome and linguistic values of conditions, or in other words, between a linguistic value at a parent block and linguistic values at children blocks. In the future, we will use a term "rule" instead of "fuzzy logic rule". For example, "**if** A has a *linguistic value* y **and** B has a *linguistic value* z, **then** C has a *linguistic value* x" is a rule that relates the linguistic value x to the linguistic values y and z. In this case, the membership function f_x for the linguistic value x is given by:

$$f_x = and(f_y, f_z)$$

where f_y, f_z are membership functions for the linguistic values y, z, respectively.

Fuzzy logic rules that define relations among peer linguistic values (linguistic values at the same level) are called preference rules. For example, "**if** A has a *linguistic value* w, **then** y has a *strong preference* **over** z" is a preference rule that gives higher weight to the linguistic value y than to the linguistic value z. If h is the membership function for *strong preference*, then it is assumed that the linguistic value y has a weight h, and the linguistic value z has a weight $1 - h$. In this work, *preference* is viewed as a linguistic variable, the membership functions for various linguistic values of *preference* are implicitly derived from the membership functions of fuzzy predicates in the preference rules. For example, if f_w is the membership function of a fuzzy predicate (for example, *linguistic value* w in the above preference rule), then the membership function for *strong preference* can be given as $\frac{1}{2}f_w + 0.5$, and the membership function for *mild preference* can be defined as $\frac{1}{4}f_w + 0.5$. If no predicates are used, then linguistic values of *preference* have crisp values, e.g., *strong preference* may have a value 0.85, and *mild preference* may have a value 0.65. If no preference rules are given among peer criteria (objectives), then they are treated as having equal preference.

Rules and preference rules can be combined in the same structure. If in the example above, both the preference rule and the rule are applied, then the membership function f_x for the linguistic value x will be given as follows:

$$f_x = and(hf_y, (1-h)f_z)$$

The hierarchical structure allows membership functions of linguistic values at a higher level to be defined by membership functions of linguistic values on a lower level. The membership functions of linguistic values at the lowest level have to be given explicitly either by analytical formula, or in table forms. If a membership function is given in a table form, then linear interpolation can be used to generate a continuous piecewise linear function. The parameters on which these lowest-level linguistic values depend are always explicitly specified. Some examples will be given in the following sections.

IV. The fuzzy logic development tool (FZDT)

The basic theory given in the previous section, is applied to build a Fuzzy Logic Development Tool (FZDT) for CAD designs. A graphical user interface for the FZDT is developed by utilizing the Tcl/Tk tool [7] and the C language in the X11 window environment. The FZDT allows users to graphically design and verify various fuzzy logic decision-making systems, and export the decision-maker in either a C code or in a binary code.

The top level interface is shown in Figure 5. Clicking

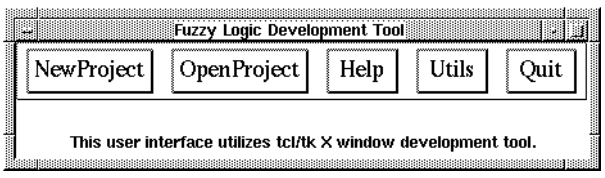

Fig. 5. Top level menu of FZDT

either "NewProject" or "OpenProject" will generate a "Fuzzy Logic Project Development Environment". The only difference between these two buttons is that the first button will generate an entirely new environment, and the second button will request from users an existing project

path name and then will bring up the project for modification. Both choices generate the same working environment and menus. Figure 6 presents a development environment with a partially developed project. Multiple clicks on these buttons will generate multiple working environments.

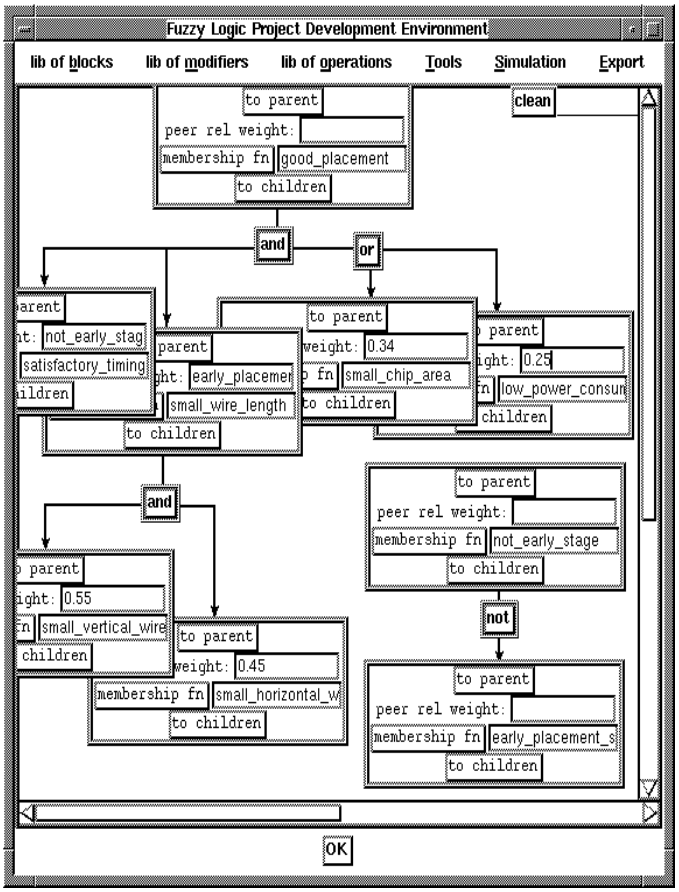

Fig. 6. Environment with a partially developed decision-maker

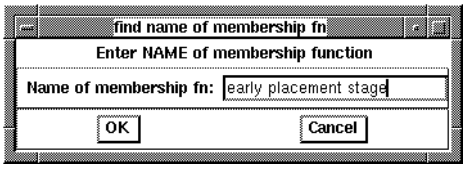

Fig. 7. Window requesting for name of membership function

the button "Setup membership fn" under the menu "Tools" will pop up a window that asks users to enter the corresponding block name (Figure 7). Suppose "early placement stage" is typed in, then <Return> or clicking

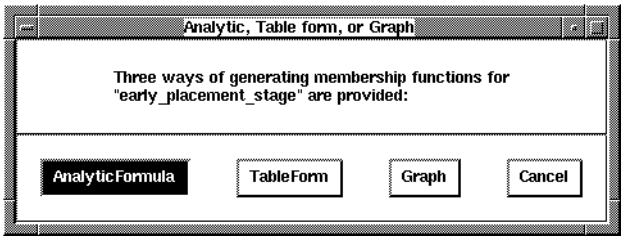

Fig. 8. Window for selecting way to customize membership function

the button "OK" will bring up another window, that will ask users to select a way for setting up membership function (Figure 8). If either the button "AnalyticFormula" or

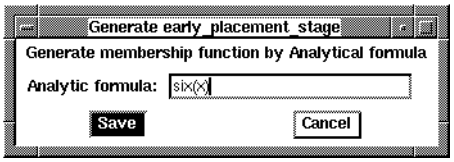

Fig. 9. Window for entering membership function by analytical formula

the button "TableForm" is clicked, then a window shown in Figure 9 or Figure 10 will be provided for a user to enter an analytical formula in C language, or to enter the data in table form. A user can also define a membership function by drawing a graph. If users have clicked the

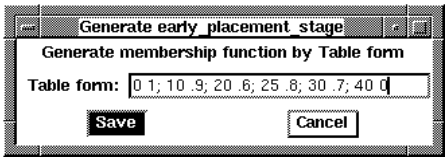

Fig. 10. Window for entering membership function in table form

button "Graph" in the window shown in Figure 8, then the window shown in Figure 11 is generated. Each circle in the graph represents a point on the curve and it is movable. More circles can be generated by clicking button "more circles" in this window. Such features allow users to construct a graph of almost any shape. Circles positioned above $y = 1$ or below $y = 0$ will be internally adjusted to have y coordinates 1 or 0.

A. Customization of membership functions

The membership function for each basic block (a linguistic variable) can be customized or modified. Clicking

B. Verifications and export

Hierarchical fuzzy logic decision-makers designed by the FZDT tool provide verifications for possible errors and can convert decision-makers designed into either a C code or a binary codes.

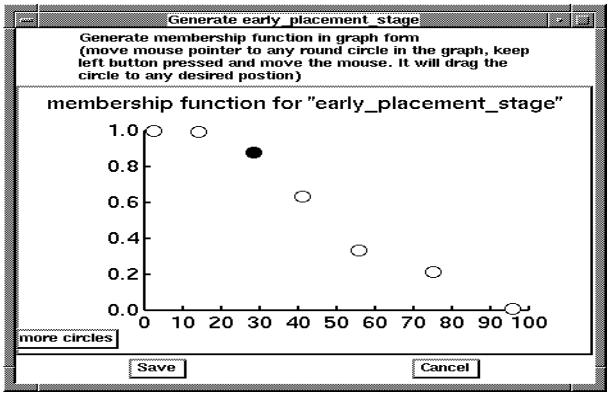

Fig. 11. Setting up membership function by constructing the graph

design name	# of cells	# of nets	# of IOs
Fract	125	163	24
Struct	1888	1920	64
Biomed	6417	7052	97

TABLE I
CHARACTERISTICS OF BENCHMARK TEST CASES

V. Applications of FZDT

A. Application to standard cell placement tool development

The FZDT has been successfully applied to develop a standard cell placement tool GOPHER. The GOPHER uses a hierarchical placement strategy and it extensively uses various fuzzy logic decision-makers generated by FZDT at all levels. Effectiveness of GOPHER has been tested on 3 MCNC standard cell benchmark test cases (The only MCNC benchmarks with timing information). The basic characteristics of these benchmarks are given in the Table I. Testing the performance of a placer is difficult, because final layout depends also on quality of global and channel routers used by the layout system. After Physical Design Workshops [5], it became a standard procedure to provide the same combination of global and channel routers for competing placement algorithms. Such strategy allows to eliminate differences in quality of routing from consideration when placers are compared. We compared our placer with two well-known and available placers: OASIS (VPNR) and TimberWolf6.1. As for some previously conducted contests, global and channel routing for all placers is performed by OASIS global and channel routers. Table II presents best area results achieved by OASIS and TW6.1 placers in conjunction with OASIS or TW6.1 global routers, and OASIS channel router [1], [3], [5], [6]. Timing delay in this table comes from our own experiments with OASIS and TW6.1 packages. Table 3 lists results for the GOPHER placer with three different combinations of goals. When the set of rules emphasizing timing was selected, propagation delay (Table IIIA) with respect to OASIS and TW6.1 was reduced 36% in average, with accompanying area increasing by 4.5 %.

design name	Fract		Struct		Biomed	
	OASIS	TW6.1	OASIS	TW6.1	OASIS	TW6.1
propagation delay(ns)	0.98	1.12	6.47	6.15	14.3	13.32
chip area(mm^2)	.53	.53	7.05	6.89	52.01	51.34

TABLE II
LAYOUT RESULTS BY OASIS AND TW6.1 AFTER DETAILED ROUTING

	A. Timing emphasized			B. Area emphasized			C. Balanced solution		
	Fract	Struct	Biomed	Fract	Struct	Biomed	Fract	Struct	Biomed
propagation delay(ns)	0.41	4.84	9.84	0.81	6.33	12.6	0.65	5.32	10.9
chip area(mm^2)	.56	6.9	55.3	.51	6.7	49.7	.53	6.8	51.4
number of rows	6	17	37	6	17	37	6	17	37
total wire length(μ)	51767	610257	6897561	49010	562213	6351751	49235	583011	6550218
total # of tracks	61	221	1024	57	198	921	58	211	958
the longest row(μ)	766	2922	6585	721	3012	6392	743	2954	6423

TABLE III
LAYOUT RESULTS BY GOPHER

When the set of rules emphasizing area was invoked, area produced was 3% smaller (Table IIIB) for all benchmarks compared to the best area achieved by OASIS and TW6.1, while timing delay was also reduced by 8% in average.

When the set of rules emphasizing both area and timing was selected (Balanced solution), no increase in area with respect to the best OASIS or TW6.1 solutions was accompanied by propagation time reduction by 28% in average (Table IIIC)

Excellent results on area reported in [1] for GORDIAN/DOMINO placers were not included in comparison, because they were obtained with different set of routers than those used in our experiments [3]. No timing information for solutions corresponding to minimal area had been published for these solutions, and authors did not have an opportunity to generate this information.

Experiments with OASIS and the GOPHER placer were conducted on the HP400t machines running DOMAIN_OS. Comparison of execution time for OASIS and the GOPHER placers is given in Table IV.

design name	OASIS	Fuzzy Logic
Fract	6(sec)	2(sec)
Struct	7(min)	15(min)
Biomed	3.2(hr)	2.2(hr)

TABLE IV
CPU RUN TIME FOR PLACERS

Comparison of running time with TW6.1 is not provided, because IRIS R4000 Indigo was used to run TimberWolf6.1.

B. Application to circuit partitioning problem

The FZDT tool has also been applied to generate a fuzzy logic decision-maker for a circuit partitioning algorithm. This partitioning algorithm was tested on two types of graphs. These graphs were generated according to the standard procedures reported in [2]. The first type of graphs was called Random Graphs, and the second type Geometric Graphs. 10 test cases of each type of graphs have been generated. As a result, 20 test cases were used in the experiments. The last column of Table V shows results of application of fuzzy logic circuit partitioning algorithm to these test cases. Results for three other partitioning algorithms [9] to the same test cases are also included for comparison. The results of comparison are favorable for fuzzy logic algorithm.

Random Graph	SA	MSH	LD	FL
Avg. initial cut-size	2530	2530	2530	2530
Avg. final cut-size	1410	1415	1368	1356
Avg. run time(s)	229	174	251	90
Geometric Graph				
Avg. initial cut-size	2380	2380	2380	2380
Avg. final cut-size	176	131	137	113
Avg. run time(s)	172	183	191	43

TABLE V
CIRCUIT PARTITIONING RESULTS
SA - SIMULATED ANNEALING, MSH - MODIFIED SEQUENCE HURISTICS, LD - LINEAR DIFFERENCE, FL - FUZZY LOGIC

VI. CONCLUSION

This work demonstrated that fuzzy logic can be successfully applied to decision-making in CAD tools. Efforts on development of fuzzy logic decision-making structure can be substantially eased by application of the tool described in this paper.

REFERENCES

[1] K. Doll, F. M. Johannes, G. Sigl, *Accurate Net Models for Placement Improvement by Network Flow Methods*, Proc. of IEEE ICCAD'92, pp.594-597

[2] D. Johnson, C. Aragon, L. McGeoch, C. Schevon, *Optimization by Simulated Annealing: An Experimental Evaluation; Part I, Graph Partitioning* Operation Research, Vol.37, No.6, 1989, pp.865-892

[3] Frank M. Johannnes, Private Communications.

[4] Rung-Bin Lin, Eugene Shragowitz, *Fuzzy Logic Approach to Placement Problem*, Proc. of 29th DAC, 1992, pp.153-158

[5] K. Kozminski, *Benchmarks for Layout Synthesis - Evolution and Current Status*, Proc. of 28th DAC 1991, pp.265-270

[6] Krzysztof Kozminski, Private Communications.

[7] J. K. Ousterhout, *Tcl and the Tk Toolkit* Addison-Wesley Publishing, ISBN 0-201-63337-X.

[8] M. Razaz and J. Gan, *Fuzzy Set Based Initial Placement for IC Layout,* Proc. of the European DAC 1990, pp. 655-659

[9] E. Shragowitz, R. B. Lin, *Combinational Optimization by Stochastic Automata* Annals of Operation Research 22(1990), pp.293-324

[10] L. A. Zadeh, *The Concept of a Linguistic Variable and its Application to Approximate Reasoning-I,* Information Science 8, pp. 199-249, 1975.

[11] H. J. Zimmermann, *Fuzzy Sets , Decision Making, and Expert Systems,* Kluwer Academic Publishers, Boston, 1987.