

Equivalence Classes of Circuit Mutants for Experimental Design

Debabrata Ghosh Franc Brglez
 CBL (Collaborative Benchmarking Lab), Dept. of Computer Science, Box 7550
 NC State University, Raleigh, NC 27695, USA
<http://www.cbl.ncsu.edu/>

Abstract – *Experimental Design, as defined in science and manufacturing, relies on data sets that belong to well-defined equivalence classes. This paper introduces a method to generate tightly controlled equivalence classes of circuit mutants, given the graph-based characterization of a reference netlist, which in general may contain cycles. A set of experiments that measure the performance for a number of core algorithms in CAD demonstrates merits of the method.*

Keywords: experimental design, circuit equivalence classes, benchmarking.

1 INTRODUCTION

Experimental Design, as defined in science and manufacturing, and the context in which it may be applied to the performance evaluation of core algorithms in CAD, has been outlined in [1]. This paper generalizes the notion of the isomorphism equivalence class of a reference netlist [2], and introduces a method to synthesize equivalence classes of circuit mutants that are distinct, but still closely related to the reference netlist isomorphism class.

Section 2 characterizes this relationship using a k-partite graph canonical form and its signature. The invariance of this signature defines the equivalence class. We outline the mutant equivalence class synthesis method in Section 3, and conclude with a representative sample of experimental results in Section 4. Earlier work, related to this topic can be found in [3], [4], [5], and [6].

2 CANONICAL FORM

An electrical netlist, with or without cycles, is usually represented as a directed hypergraph, *e.g.*, Figure 1(a). We introduce an ordered directed k-partite graph representation of the netlist, *e.g.*, Figure 1(b), as follows:

(1) *Decomposition:* These steps generate an acyclic graph.

- We first decompose the circuit into its strongly connected components (SCCs). In Figure 1(a), the two non-trivial SCCs are shaded.

- The graph is then rendered acyclic by deleting as few feedback edges as possible *in each SCC*. For example, the feedback edges in Figure 1(a) are $l \rightarrow f3$ (SCC_1), $m \rightarrow f1$ (SCC_2), $n \rightarrow f2$ (SCC_2). Note that, a feedback edge can exist only between a combinational cell and a flip-flop since we are dealing with synchronous circuits only.

- Each deleted feedback edge is represented by a feedback PO (FPO) at its source and a feedback PI (FPI) at its sink position (diamond-shaped in Figure 1(b)).

(2) *First Levelization:* This procedure is well known.

- All PIs and FPIs are assigned a $level = 0$.
- All cells are assigned a $level > 0$ using topological sort.
- Each net inherits the name and level of the driving cell.

(3) *Second levelization:* The initial levelized form is further modified as follows:

- Each PI or FPI net is assigned a $level = l - 1$ where l is the minimum level of cells that it is driving. For example, PI d in Figure 1(b) drives cells at a minimum level of 2, hence, d is assigned a $level = 1$.

- Each net is assigned a netspan $\lambda = p_{max} - p_{min}$ where p_{max} and p_{min} are the maximum and minimum level of cells it is adjacent to.

- Each net of netspan $\lambda > 1$, is broken into segments by introducing $\lambda - 1$ single-input, single-output feedthroughs. In Figure 1(b), we see one feedthrough on the net $f1$. Feedthrough at level i is named as $\langle net_name \rangle . i$.

- Finally, each net is replaced by a net node with fan-in of 1, but variable fanout.

- A net node at SCC i is called a *component IO* or *CIO* if it drives a cell at another SCC $j (\neq i)$. Net $g5$ is a CIO in Figure 1(b).

(4) *Find the local clustering at each level:* We capture the local clustering by finding the *connected components (CCs)* of the underlying undirected graph at each level. This is done by a simple DFS search. CCs at level i consists of cell nodes at level i and net nodes at level $i - 1$. In Figure 1(b), each net and cell node is marked with the CC number that it belongs to.

Characteristic Wiring Signature. The wiring signature captures many attributes of the physical structure of the netlist. We characterize each cell node (Figure 1(a)) in the canonical form as a 6-tuple of the form:

$\langle name \rangle \langle type \rangle \langle input, output \rangle \langle level \rangle \langle CC \rangle \langle SCC \rangle$

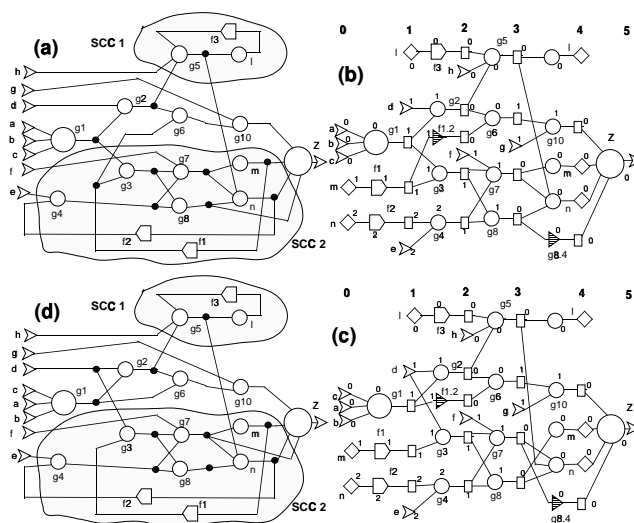


Fig. 1. (a) reference netlist, (b) reference canonical form, (c) instance of a mutant (canonical form), (d) mutant netlist.

This research was supported by contracts from the Semiconductor Research Corporation (94-DJ-553), SEMATECH (94-DJ-800), and DARPA/ARO (P-3316-EL/DAAH04-94-G-2080 and DAA655-97-1-0345).

$g_1, C, \{3,1\}, 1,0,0$	$a, I-N, 0,0,0$	Lev:	0	1	2	3	4	5
$f_1, D, \{1,1\}, 1,1,2$	$m, i-N, 0,1,2$	$L_{i,I}:$	3	2	2	1	0	0
.	.	$L_{i,i}:$	2	1	0	0	0	0
.	.	$L_{i,O}:$	0	0	0	0	0	1
$g_5, C, \{3,1\}, 3,0,1$	$g_1, C-N, 1,1,0$	$L_{i,o}:$	0	0	0	0	3	0
$g_6, C, \{2,1\}, 3,0,0$	$f_1, D-N, 1,1,2$	$L_{i,D}:$	0	2	1	0	0	0
.	.	$L_{i,F}:$	0	0	1	0	1	0
$n, C, \{3,1\}, 4,0,2$.	$L_{i,C_{4,1}}:$	0	0	0	0	0	1
$m, C, \{1,1\}, 4,0,2$	$g_5, C-Y, 3,0,1$	$L_{i,C_{3,1}}:$	0	1	0	2	1	0
$g_{8.4}, F, \{1,1\}, 4,0,0$.	$L_{i,C_{2,1}}:$	0	0	3	2	1	0
.	$m, o-Y, 4,0,2$	$L_{i,C_{1,1}}:$	0	0	0	0	2	0
.	.	$CC_C:$	0	3	3	2	2	1
$Z, C, \{4,1\}, 5,0,0$	$Z, 0-N, 5,0,0$	$CC_N:$	3	3	2	2	2	1
(a)	(b)	(c)						

Fig. 2. Characteristic signatures: (a) cell nodes, (b) net nodes, (c) compressed form.

Here, **name** is the name of the cell node; **type** denotes the type of the cell node and can be one of $\{C, D, F\}$ depending on whether it is combinational gate, a sequential gate (a flip-flop) or a feedthrough respectively; the pair (**input,output**) denotes the number of input and output pins of the cell; **level**, **CC** and **SCC** are the level, **CC** and **SCC** the cell belongs to.

Similarly, we characterize each net node ((Figure 1(b)) as a 5-tuple of the form:

$$\langle \text{name} \rangle \langle \text{type} \rangle \langle \text{level} \rangle \langle \text{CC} \rangle \langle \text{SCC} \rangle$$

Here, **type** is of the form $x-z$ where $x \in \{I, i, 0, o, C, D, F\}$ which correspond to net nodes of type PI, FPI, PO, FPO, net nodes driven by C-type cells, net nodes driven by D-type cells, net nodes driven by F-type cells respectively; $z \in \{Y, N\}$ corresponding to whether the net node is a CIO or not.

We compress, without any loss of information, the detailed signatures in Figure 1(a),(b) by sorting the cell/net nodes based on their type and level and report a levelwise distribution. The compressed form of the *characteristic wiring signature* is shown in Figure 2(c).

At each level i , $L_{i,X}$ denotes the total number of net nodes of type X where $X \in \{I, i, 0, o, D, F\}$ (definitions of these symbols are as before) or X may be of type C. Nets of type C are further sub-divided based on the number of input pins and output pins of the cell driving it. Hence, $L_{i,C_{m,n}}$ denotes the total number of net nodes at level i that are driven by cell nodes with m input and n output pins.

The entities CC_C and CC_N in Figure 2(c) denote the number of CCs formed with cell nodes and net nodes respectively at level i .

Using the characteristic signature, we can derive a number of fanout bounds which are essential for the proper termination of the mutation process. Discussions of the fanout bounds are omitted (see [7] for details).

Mutant Equivalence Class. The characteristic signature of a circuit is unique (unique under a given feedback vertex set)¹. It induces an equivalence class that is defined as the collection of all circuits that have the same signature as the current circuit (called the *reference circuit*). Each circuit in this equivalence class is called a *mutant* of the *reference circuit*. In the next section we describe the method for generating the *signature-invariant mutant circuits* in this equivalence class.

¹Note that the feedback vertex set (FVS) is not unique. The problem of finding a minimum FVS is known to be NP-hard. We find a **minimized FVS** using a polynomial time algorithm and maintain it for all the mutants.

The mutant synthesis procedure consists of two basic steps: *perturbation* and *mutation*. The process is initiated by a *perturbation* in the canonical form. It amounts to random removal of p wires or $q\%$ wires at each level i .

Mutation consists of restoring all the connections as a random process. However, any such connection must conform to the *conditional incidence relationships* [2], [7] which dictate the allowable connections between a net and a cell node based on their individual types. Additionally, the connections must maintain the fanout bounds derived from the signature[7]. These are essential for *maintaining the characteristic signature*.

Observing the fanout bounds and the conditional incidence relationships are necessary but *not* sufficient conditions for maintaining the signature in a sequential circuit. Due to the nature of the sequential circuits (presence of loops etc.), one needs to observe the following two constraints that are unique to sequential circuits:

- S1, Feedback Vertex Set Invariance (FVI):** The feedback vertex set (FVS) present in the reference circuit must be preserved. FVI may be equivalently stated as *feedback edge set invariance*, FEL, since there is a one-to-one correspondence between a feedback edge and a feedback vertex.
- S2:** Each node, including each feedback vertex, on a cycle must have a path to at least one PO of the circuit.

FVI is essential for maintaining the signature. Without the FVI, the levelization of the cells changes, hence, the signature cannot be maintained. Condition S2 is related to the quality of the mutants generated – without S2 there is a possibility of having isolated/floating parts in the mutant that cannot be observed at any of the POs of the circuit. For example, for the mutant in Figure 1(d), if the cycle in SCC_1 did not have a path $g_5 \rightarrow n \rightarrow Z$ to the PO Z , the entire circuitry in SCC_1 would be floating and would not serve any purpose.

The essence of maintaining the FVI is to create a path, in the acyclic canonical form, from an FPI n_i to the FPO m_o that is designated to be connected to n_i . It ensures that in the cyclic graph the edge between m_o and n_i will remain a feedback edge. Further, since each FPI drives a single flip-flop, it is ensured that no additional feedback loop will be introduced in the cyclic graph. Details of the FVI maintenance procedure is involved and are omitted (see [7] for details). In the following, we describe a few concepts that are useful for this purpose.

Feedback tokens (F-type tokens). We associate a *feedback token* with each feedback vertex (or, equivalently, with each feedback PI (FPI) since the only fanout from an FPI is to a feedback vertex). Each FPI, n_x , issues a *distinct* feedback token α_{n_x} . For any connection from a cell node C_i (net node N_j) to a net node N_j (cell node C_i), N_j (C_i) inherits all the tokens of C_i (N_j). The mutation process must ensure that the FPO, m_x , that is going to be connected to n_x (in the cyclic graph) receive the token α_{n_x} (to satisfy condition S1). Additionally, α_{n_x} must reach at least one PO (to satisfy condition S2).

Component tokens (C-type tokens). Component tokens are issued by the component IOs (CIOs) under certain conditions. If a CIO in SCC_j receives feedback tokens $\{f_1, \dots, f_n\}$, at least one of which has not yet propagated to any PO, the CIO issues a C-token c_i . c_i ceases to exist when all of $\{f_1, \dots, f_n\}$ reach some PO (possibly through some other path). Component tokens play a pivotal role in carrying the feedback tokens to a PO from an SCC that

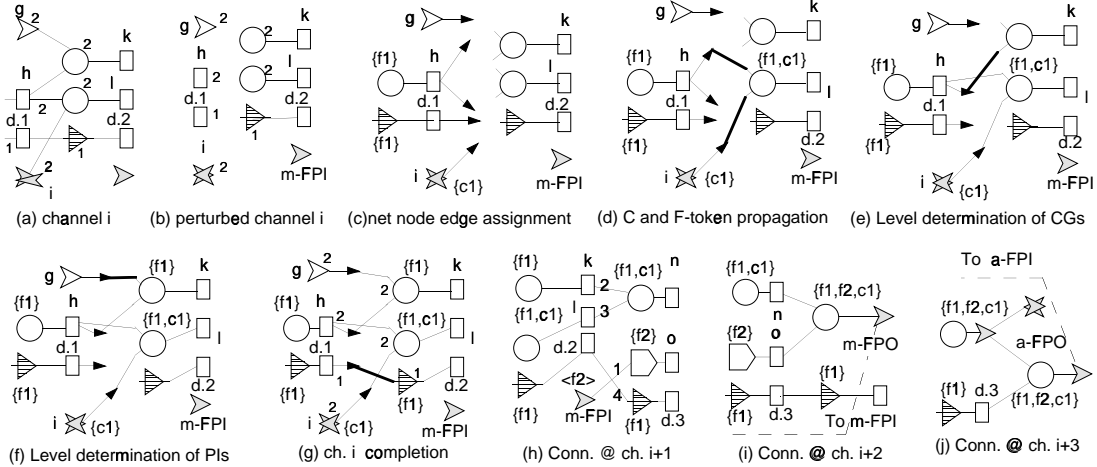


Fig. 3. Mutation process in channels $i, i + 1, i + 2, i + 3$.

does not have any PO in it.

Forward token net set. A forward token net set at level i is defined as the *minimal* set of net nodes (at level i) that contains all the ‘alive’ tokens (tokens that are yet to propagate to a PO), both C and F-type, issued up to level i . In other words, to pick up all the tokens, it suffices to look up the nodes in the forward token net set.

Feedback token net set. A feedback token net set at level i is defined as the *maximal* number of net nodes that must carry all the tokens.

Connection Assignments. The connection assignments at each level $i, i > 0$, can be broadly classified into *five* steps.

(A1): *bounded net node edge assignment.* Here, p_i net nodes at level i are assigned a bounded distribution of q_{i+1} edges that are to be connected to q_{i+1} input pins of cell nodes at level $i + 1$;

(A2): *forward token net set formation.* Both F and C-type tokens are combined on cell nodes, subject to the bounds on the forward token net set.

(A3): *feedback loop completion.* Cells driving FPOs are connected to token-carrying net nodes, and, FPOs are connected to FPIs such that each FPI gets back the token it issued.

(A4): *local CC formation.* Ensures that each cell/net in CC_k has an undirected path to every other cell/net in CC_k .

(A5): *restricted cell node edge connection.* Remaining of the q_{i+1} edges sourced at the net nodes are sunk at input pins of cell nodes, subject to the conditional incidence relationships.

Note that no connection is allowed between cell/net nodes belonging to different CC and/or SCC. The only exception are the CIOs which drive cells in different SCCs.

Illustrative Example. We illustrate the mutant synthesis process through a simple example in Figure 3. Figure 3(b) is a 100% perturbed version of the channel at level i shown in Figure 3(a). Note that at this level we have two local CCs, CC_1 consisting of the feedthrough $d.2$ and the net node $d.1$ and CC_2 consists of all other cells and net nodes.

Figure 3(c) illustrates the fanout assignment to each net node under certain bounds. In Figure 3(d), the forward token net set is formed. Note that tokens f_1 (F-type) and c_1 (C-type) are ‘alive’ here. The upper bound on the forward token net set at this level is 2. By random choice, we assign both f_1 and c_1 to the single node l , thereby forming a forward token net set of size 1. Figure 3(e) and (f) per-

form connections to ensure the levels of combinational gates (CGs) and PIs, as dictated by the conditional incidence relationships. Note that, after the step in Figure 3(f), CC_2 has been reconstructed; if not, we would have to do that in the next step. All the remaining connections are completed in Figure 3(g) (in the process, CC_1 gets formed). (In Figure 3(d)–(g), connections at each stage are shown by bold edges).

Connections in Figure 3(h) are done in the order as marked on the edges. Note that, the FPI named m-FPI issues a new feedback token f_2 at this stage.

In Figure 3(i), the feedback loop is completed. The FPO m-FPO receives both the tokens f_1 and f_2 , corresponding to FPIs a-FPI and m-FPI. Hence it can be connected to either of a-FPI or m-FPI. By random choice, we connected it to m-FPI.

Finally, in Figure 3(j), the feedback loop corresponding to token f_1 (issued by a-FPI) is completed and all the tokens $\{f_1, f_2, c_1\}$ are propagated to a component output. Since all the tokens issued in this SCC have been propagated out of it, the mutation process is considered to be successful.

4 EXPERIMENTAL RESULTS

In this section we produce experimental results to show the usefulness of the mutant classes as benchmarks for testing CAD algorithms. We report results with two equivalence classes of the benchmark **s1423** - the isomorphism class and the 100% mutation class (more extensive results are reported in [7]). There are 128 circuits in each class. These mutants, in addition to preserving the signature, also preserve the number of strongly connected components (SCCs). Preserving the numbers of SCCs is not guaranteed by the mutation process, although it attempts to do so. In these experiments, we filtered the generated mutants to have mutants with the same number of SCCs as the reference circuit. The success rate was about 70%.

We judge the quality of the mutation classes by comparing the performance of various physical design algorithms on (1) the isomorphism class and (2) the mutation class. The tools we used in this experiments are: (1) DOT [8] to generate circuit schematics and analyze the number of wire crossings; (2) PROP [9] to report the mincut of a balanced bipartition; (3) OASIS [10] to layout the circuits using standard cell place and route methodology and report layout area and wire length; (4) VPR [11] to place and route in an

TABLE I
STATISTICAL SUMMARY OF THE DISTRIBUTIONS.

Algorithm	μ for ISO	μ for MUT	% diff.	σ for ISO	σ for MUT	% diff.	t-test score
WC (DOT[8])	3172	3249	2.4	515	530	3.0	1.3
Area (OASIS[10])	1.84	1.88	2.2	0.47	0.50	6.8	6.6
WL (OASIS[10])	2.30	2.39	4.1	0.10	0.11	5.6	6.8
WL (VPR[11])	2665	2696	1.2	109	105	4.1	2.3
mincut (PROP[9])	13.6	15.3	12	4.2	4.4	5.7	3.1

FPGA architecture and report the routing wire length. Results are shown in Figure 4 which shows the frequency distribution of the cost function for the respective algorithm. Table I shows the relevant statistics of these distributions, *e.g.*, the mean (μ), the standard deviation (σ), the percentage variation of these parameters in the mutation class with respect to the isomorphism class, and results of statistical t -test [12] to determine how different the two distributions are.

Figure 4(a),(e): Here we report distributions of the number of wire crossings, as optimized by DOT [8]. It is evident that the two distributions are very similar. It may be noted that unlike the isomorphic circuits, the circuits in the mutation class are all different, but the class offers the same level of difficulty to the tool (DOT, in this case) as the class of isomorphic circuits would. Table I shows that the mean for the two classes vary by an insignificant 2.4%. A t -test also reveals, with a 95% confidence, that the two distributions are indistinguishable [12].

Figure 4(b),(f): Here we report the area for the layout tool in OASIS. Again the distributions for isomorphism and mutation class are very similar and there is insignificant percentage variation in their mean values as shown in Table I. The t -test score says that the two distributions are distinguishable, which is not surprising since the two classes are different. However, the mutation class offers approximately the same degree of difficulty to the CAD tool as the isomorphism class.

Figure 4(c),(g): Here we report the routed wire length for the FPGA place and route tool VPR [11]. The distributions are similar with insignificant variations in mean. The t -test score of 2.3 implies that although the distributions are different, the difference is small.

Figure 4(d),(h): Here we report the size of the mincut for a balanced bipartition. Visually, the two distributions appear similar. There is some variation in the mean but the variation in standard deviation is smaller. A t -score of 3.1 reveals that the two distributions are different but still quite close.

These results show that the mutation class behaves very similarly to the isomorphism class under four different physical design algorithms optimizing four different cost functions. This has been possible through an accurate representation of the netlist in the canonical form, and subsequently, in the signature.

As a further experiment, we generated an equivalence class of 128 circuits by randomly choosing 8 mutants and generating 16 isomorphic instances of each. For wire crossing optimization by DOT, this new class produces a distribution that is indistinguishable from the original mutant class as well as the reference isomorphism class. This shows the closeness of the mutants.

5 CONCLUSIONS AND FUTURE WORK

In this paper we have presented a useful method of generating circuit equivalence classes that preserve many netlist properties of the reference circuit. Four different physical

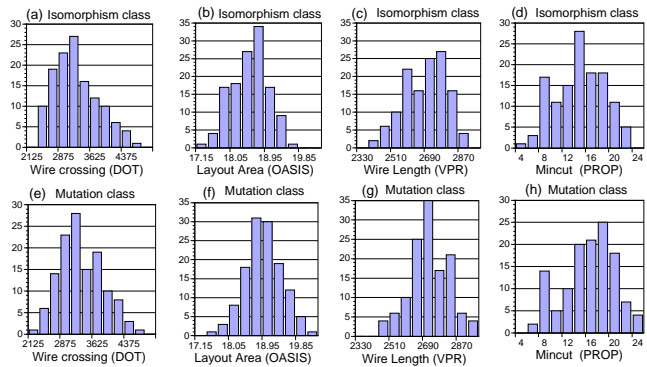


Fig. 4. Comparison of the performances of various physical design algorithms on the isomorphism and mutation classes.

design tools produce similar results for the mutants as they would for the isomorphism class, in spite of the mutants being all different circuits. Hence, they can be used for design of experiments as described in [1] for benchmarking CAD algorithms for physical design.

Future work includes setting up elaborate experiments for the performance evaluation of the physical design tools. For updates on experiments, see http://www.cbl.ncsu.edu/experiments/DoE_Archives/

REFERENCES

- [1] F. Brglez and R. Drechsler. Design of Experiments in CAD: Context and New Data Sets for ISCAS'99. In *IEEE 1999 International Symposium on Circuits and Systems - ISCAS'99*, May 1999. A reprint also accessible from <http://www.cbl.ncsu.edu/publications/#1999-ISCAS-Brglez>.
- [2] Debabrata Ghosh, Nevin Kapur, Justin E. Harlow III, and Franc Brglez. Synthesis of Wiring Signature-Invariant Equivalence Class Circuit Mutants and Applications to Benchmarking. In *Proceedings, Design Automation and Test in Europe*, pages 656–663, Feb 1998. Also available at <http://www.cbl.ncsu.edu/publications/#1998-DAT-E-Ghosh>.
- [3] Michael Hutton, J.P. Grossman, J. Rose and D. Corneil. Characterization and Parameterized Random Generation of Digital Circuits. In *Proceedings of the 33rd ACM/IEEE Design Automation Conference*, June 1996.
- [4] Michael Hutton, J.P. Grossman, J. Rose and D. Corneil. Generation of Synthetic Sequential Benchmark Circuits. In *ACM Symposium on FPGAs*, pages 149–155, February 1997. Available from <http://www.eecg.toronto.edu/~jayar/pubs/pubs.html>.
- [5] Michael Hutton, J.P. Grossman, J. Rose and D. Corneil. Characterization and Parameterized Random Generation of Combinational Benchmark Circuits. *IEEE Trans. Computer-Aided Design*, 1999. To appear. Postscript available from <http://www.eecg.toronto.edu/~jayar/pubs/pubs.html>.
- [6] J. Darnauer and W. Dai. A Method for Generating Random Circuits and its Application to Routability Measurement. In *4th ACM/SIGDA Int'l Symp. on FPGAs, FPGA96*, pages 66–72, February 1996.
- [7] D. Ghosh and F. Brglez. Equivalence Classes of Sequential Circuit Mutants for Experimental Design. Technical Report 1999-TR@CBL-02-Ghosh, CBL, CS Dept., NCSU, Box 7550, Raleigh, NC 27695, March 1999. Also available at <http://www.cbl.ncsu.edu/publications/#1999-TR@CBL-02-Ghosh>.
- [8] E.R. Gansner, E. Koutsifios, S.C. North and K.P. Vo. A Technique for Drawing Directed Graphs. *IEEE Trans. Software Engng.*, 19:214–230, 1993.
- [9] R. Kužnar and F. Brglez. PROP: A Recursive Paradigm for Area-Efficient and Performance Oriented Partitioning of Large FPGA Netlists. In *IEEE International Conference on Computer-Aided Design*, pages 644–649, November 1995.
- [10] K. Kozminski, (Ed.). OASIS2.0 User's Guide. MCNC, Research Triangle Park, N.C. 27709, 1992. (Over 600 pages, distributed to over 60 teaching and research universities worldwide).
- [11] V. Betz and J. Rose. VPR: A New Packing, Placement and Routing Tool for FPGA Research. In *Proceedings of the 7th International Workshop on Field-Programmable Logic*, pages 213–222, August 1997. Software and postscript of paper can be downloaded from <http://www.eecg.toronto.edu/~vaughn/vpr/vpr.html>.
- [12] G. E. P. Box, W. G. Hunter, and J. S. Hunter. *Statistics for experimenters: An Introduction to Design, Data Analysis, and Model Building*. John Wiley & Sons, 1978.