

CBL (Collaborative Benchmarking Laboratory)
Department of Computer Science
Campus Box 7550
North Carolina State University
Raleigh, NC 27695

Circuit Equivalence Classes and Design of Experiments for Performance Evaluation of Algorithms in Physical Design

Debabrata Ghosh Franc Brglez

Technical Report 1998-TR@CBL-09-Ghosh
September 1998
ft 1998 CBL
All Rights Reserved

“Permission to make digital/hard copy of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication and its date appear, and notice is given that copying is by permission of CBL. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.”

If you choose to cite this report, please add the following entry to your bibliography database:

```
@techreport{
  1998-TR@CBL-09-Ghosh,
  author = "D. Ghosh and F. Brglez",
  title = "{Circuit Equivalence Classes and Design of Experiments for
Performance Evaluation of Algorithms in Physical Design}",
  institution = "{CBL, CS Dept., NCSU, Box 7550, Raleigh, NC 27695}",
  number = "1998-TR@CBL-09-Ghosh",
  month = "Sep",
  year = "1998",
  note = "{Also available at http://www.cbl.ncsu.edu/publications}"
}
```

To contact the Collaborative Benchmarking Laboratory via Internet, you may consider:

WWW :	http://www.cbl.ncsu.edu/
Anonymous FTP :	ftp://ftp.cbl.ncsu.edu
For an auto-reply:	benchmarks@cbl.ncsu.edu
To deposit a file at CBL:	ftp cbl.ncsu.edu
	cd /pub/Incoming
	put new_benchmark.tar.Z

Circuit Equivalence Classes and Design of Experiments for Performance Evaluation of Algorithms in Physical Design

²CBL (Collaborative Benchmarking Lab), Dept. of Comp. Science, Box 7550, NC State U., Raleigh, NC 27695

<http://www.cbl.ncsu.edu/>

Debabrata Ghosh

Franc Brglez

Abstract – *The Internet-based desktop environment as defined in this paper consists of a cross-platform browser, a number of server icons (host nodes), a number of application icons (program nodes) and a number of data icons (file nodes). In contrast to typical desktops of today, where data icons may be dragged and dropped onto application icons for execution, this environment allows (1) user-defined and reconfigurable execution sequences by creating dependency edges between program nodes (application icons) and file nodes (data icons); (2) data-dependent execution sequences by dynamic scheduling of path as well as loop executions; (3) host-transparency as to the location of applications and data (both can reside on any host with a unique IP address).*

We demonstrate that the Internet-based workflow paradigm is suitable for creation of dynamically reconfigurable desktop environments. In related research, we show that the proposed desktop is particularly suitable for making such an environment collaborative and recordable.

I. INTRODUCTION

Design of Experiments and *Experimental Design* are well-established disciplines in sciences and manufacturing processes; keyword entries to popular search engines on the Web return up to 11,259 and 32,086 hits, respectively. However, the application of the Design of Experiments (DoE) methodology to performance evaluation of algorithms in CAD does not appear to be included in these hits.

Upon analysis of results in current publications on VLSI CAD, a statistician could well conclude that the incremental improvements in say wire length after placement, layout area after placement and routing, etc., *as typically reported*, have little if any statistical significance. The major problem is that experiments are performed on a *single instance* of few *unrelated* circuits – unlike the experiments in, say biomedicine, that are expected to be conducted on a *class of subjects* with well-controlled properties (same species, same sex, same age, same weight, same cholesterol level, etc.). As we demonstrate in the paper, the performance of a heuristic can vary widely just by changing the order in which the data for the same netlist is presented.

A number of approaches to synthesize circuit descriptions, expected as useful for ‘benchmarking’, have been published

This research has been 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).

“Permission to make digital/hard copy of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication and its date appear, and notice is given that copying is by permission of CBL. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.”

© 1998 CBL

recently. Unlike random graphs, these take into account constraints of digital circuits [1],[2] [3]. The approach in [4] relies on logic-invariant transformations and can generate larger and larger graphs without changing the function. However, as shown in this paper, reporting results of ‘benchmarking’ is not equivalent to reporting results of experimental design with equivalence circuit classes. We expand on the notion of ‘equivalence class’, as introduced in [3], to not only formalize the design of experiments but also to introduce an important new circuit equivalence class, the class of *sequential mutants*.

The paper is organized into sections that bring together the following: (1) formal introduction of equivalence classes, using the directed bipartite graph as the primitive class; (2) design of simple experiments to illustrate the performance evaluation of representative algorithms in physical layout with classes in (1); (3) characterization of invariants in *netlists with cycles* such that we can synthesize *sequential circuit equivalence classes* called *sequential mutants*.

II. BACKGROUND AND MOTIVATION

In this section we formalize the definitions that lead to graph equivalence classes and conclude with representative illustrations for each member of the class.

Crossing number in bidags. Consider a bipartite directed acyclic graph (bidag) $G_2 = (V_0, V_1, E)$ and its embedding in a plane so that the nodes in V_i occupy distinct positions on the line $x = i$ and the edges, directed from V_0 to V_1 , are straight lines. We consider each embedding as a *presentation* $\langle G_2, \pi_0, \pi_1 \rangle$, where π_i is a permutation of V_i . For any embedding, $C(\langle G_2, \pi_0, \pi_1 \rangle)$ denotes a *crossing number* of G_2 , the number of line intersections induced by the embedding. This number depends only on the permutation of V_i along $y = i$ and not on specific y-coordinates. The objective of the bigraph crossing problem is to compute (or approximate) $C(G_2) = \min_{\pi_0, \pi_1} C(\langle G_2, \pi_0, \pi_1 \rangle)$, a problem proven to be NP-hard [5].

Characteristic signatures of bidags. A *characteristic signature* of a *reference graph (bidag)* $G_{2,r}$ bidag can be any number of mappings based on parameters that relate to the size and distribution of vertices and edges in the graph. Examples of signatures that we find useful in this work include:

$$\begin{aligned} \sigma_{\text{iso}}(G_{2,r}) &= I \text{ (Identity)} \\ \sigma_{\text{mut}}(G_{2,r}) &= \{|V_0|, (\text{dist}|V_1|), (\text{dist}(|E|))\} \\ \sigma_{\text{rnd}}(G_{2,r}) &= \{|V_0|, |V_1|, |E|\} \end{aligned} \quad (1)$$

where $(\text{dist}|V_1|)$ denotes the distribution of vertices in V_1 , classified in terms of edges incident at each vertex, and $(\text{dist}(|E|))$ denotes the distribution of edges in E classified in terms of connected components that constitute $G_{2,r}$.

Perturbations in bidags. Given $G_{2,r}$, we induce a *perturbation graph (bidag)* $G'_{2,r}$ as follows: (1) we randomly select

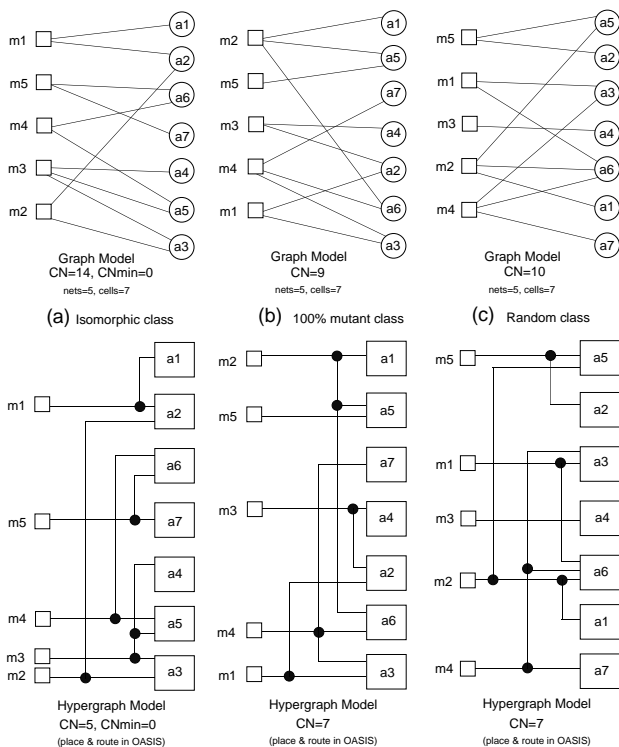


Fig. 1. Illustrating graph and hypergraph models of a directed bipartite graphs (bidags) for three equivalence classes.

and remove a fraction q of edges from E and put them into an empty urn, (2) we pick edges from the urn and randomly connect vertices V_0 and V_1 , preserving the direction from V_0 to V_1 , until the urn is empty.

Equivalence classes of bidags. We define *equivalence classes (of bidags)* in terms of presentations $\langle G_{2,r}, \pi_0, \pi_1 \rangle$, signatures $\sigma(G_{2,r})$, and perturbations $G'_{2,r}$. In this paper, we define three equivalence classes: isomorphic, mutation, and random:

$$\begin{aligned}
 (G_2)_{\text{iso}} &= \{\text{presentation}\langle G_{2,r}, \pi_0, \pi_1 \rangle | \sigma_{\text{iso}}(G_{2,r})\} \\
 (G_2)_{\text{mut}} &= \{\text{presentation}\langle G'_{2,r}, \pi_0, \pi_1 \rangle | \sigma_{\text{mut}}(G_{2,r})\} \\
 (G_2)_{\text{rnd}} &= \{\text{presentation}\langle G'_{2,r}, \pi_0, \pi_1 \rangle | \sigma_{\text{rnd}}(G_{2,r})\}
 \end{aligned} \quad (2)$$

Each class must satisfy the characteristic signature associated with the class. There can be several subclasses of the mutant class, depending on the size of the perturbation in $G'_{2,r}$. For uniformity with earlier classifications and illustrations in this paper, we refer to $(G_2)_{\text{iso}}$ as the **WD class**, $(G_2)_{\text{mut}}$ as the **WBB class**, $(G_2)_{\text{rnd}}$ as the **WRD class**.

Illustrations (graph model). Representative instances of reference bidags for each of the three classes are shown in the top part of Figure 1 (we defer the discussion about the bottom part to the next paragraph). In the context of the work that follows, we will refer to vertices in V_0 as *net nodes* and vertices in V_1 as *cell nodes*. The graphs in Figure 1 differ in connectivity only, all have 5 net nodes, 7 cell nodes, and 11 edges. Each instance of the graph satisfies the signature $\sigma_{\text{rnd}} = \{5, 7, 11\}$, but only instances of the isomorphism and mutant class satisfy the signature $\sigma_{\text{mut}} = \{5, (3, 4), (11)\}$. The latter signature shows that we have 3 cell nodes with 1 input and 4 cell nodes with 2 inputs; there is a single connected component with 11 edges. In contrast, note that the instance of the random graph consists of two connected com-

ponents and a distribution in $|V_1|$ that is different and may change with any new instance of the random graph.

The crossing numbers in this graph are shown to be 14, 8, and 10. While not obvious, the reference graph in the isomorphism class does have a bi-planar embedding, i.e. its crossing number is 0.

Illustrations (hypergraph model). When submitted to place and route tool such as OASIS [6], each graph in the top part of Figure 1 may be routed as the hypergraph model shown in the bottom part of Figure 1 – assuming that the node orders shown for the respective graphs above are preserved. Notably, with the hypergraph model, there can be a significant reduction in the wire crossing wrt the graph model: from 14 to 5, 8 to 7, 10 to 7, respectively. A number of questions arises when considering algorithm performance in terms of the graph and hypergraph models as shown in Figure 1:

- (1) “How effective are the algorithms that minimize the crossing number in graphs?”
- (2) “Can minimization of crossing number in graphs indirectly minimize the crossing number in the hypergraph model, leading to placements with shorter wire length and fewer tracks in the routing channel?”

In the next section, we provide a number of illustrative experimental designs that may provide, when fully implemented, answers of statistical significance to question posed above.

III. DESIGN OF EXPERIMENTS

Two of the fundamental principles of experimental design are *randomization* and *replication*. We adopt these principles for the experimental evaluation of heuristics by (1) creating a *presentation equivalence class* and (2) repeating the experiments for each member in the class. The basic abstractions for such experiments include

1. an equivalence class of experimental subjects, eligible for a treatment;
2. application of a specific treatment;
3. statistical evaluation of treatment effectiveness.

Here, a *treatment* is synonymous with a *heuristic* and an equivalence class of experimental subjects is synonymous with a graph presentation class. Figure 2 illustrates these abstractions in a generic flow. In this paper, treatments we apply include crossing number minimization, placement and routing, partitioning, and simulation. The indexes under evaluation include mincut, crossing number, wire length, layout area, entropy of circuit outputs and states.

Performance index specifics. For any given placement of nodes, the evaluation of the crossing number in a bidag is unambiguous and unique. This is not the case for its hypergraph model – the crossing number can vary with the placement of the Steiner points (as can be readily observed in Figure 1). However, by restricting each pair of nets to cross at most once, we bypass the Steiner point placement problem and calculate a well-defined lower bound on the crossing number in the hypergraph model of the bidag [10]. This bound is the crossing number for the hypergraph model reported in our experiments. The related crossing distribution problem, as formulated in [11], considers the graph model.

Assuming a point model for each node and unit grid placement, we developed simple formulas to evaluate wire length in the graph and hypergraph model [10]. In OASIS [6], wire length and area are reported directly. To evaluate the crossing number for each OASIS placement, we extract the placement and report its crossing number for both the graph and hypergraph model.

Experimental subjects. In this section, we present results of experiments based on isomorphism, mutant, and random

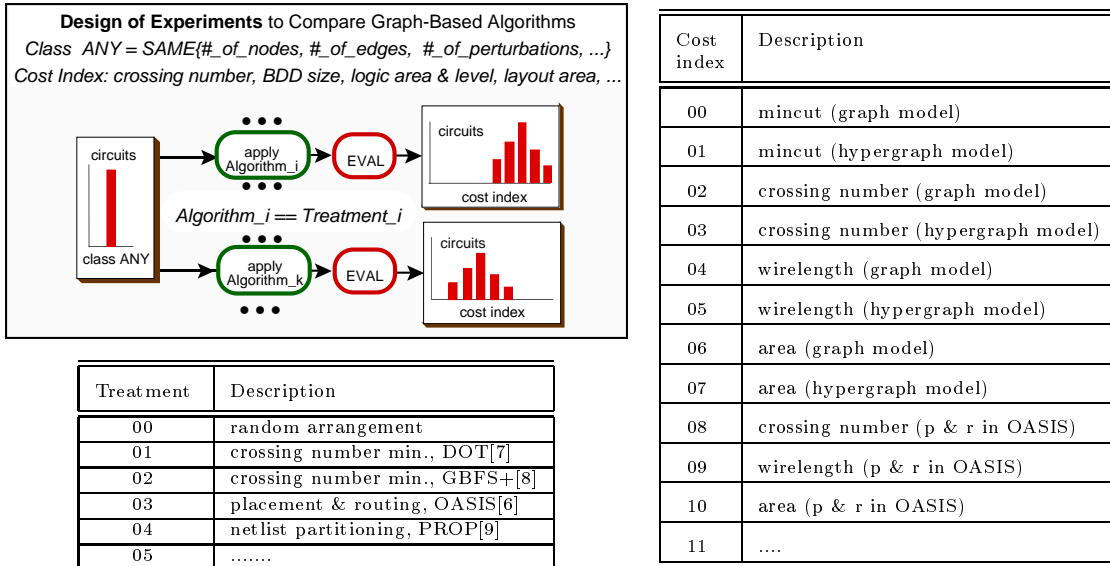


Fig. 2. Design of experiments to characterize properties of equivalence class test cases wrt a number of treatments.

classes (hundred graph instances in each class), each derived from the reference circuit `ref_multi1_016` (66 net nodes, 67 cell nodes, 134 edges). This is one of several graphs designed to evaluate asymptotic performance of crossing number minimization algorithms in [8].

Results and surprises. A partial summary of experimental results is shown in in Figure 3, with data arranged in 4 rows and 3 columns. Each of the columns relates to the isomorphism, mutant, and random class, respectively. We briefly highlight the following observations:

Crossing number in bidags (all treatments). There is no surprise for Treatment 00 (random placement). Regardless of the class, the crossing number mean is about the same and within range predicted by a formula [12]. The major surprise is the high variability of the crossing number for the *isomorphism class* – a perfect algorithm would have reported a crossing number of 32 for each and every presentation of $\langle G_2, \pi_0, \pi_1 \rangle$. The best average is achieved with Treatment 02 [8]. The average reported by Treatment 03 (OASIS, [6]) is surprisingly high, notwithstanding that the index minimized in OASIS is based on a model of a wire length. The performance of all treatments is relatively consistent across all equivalence classes.

Crossing number in hypergraphs (Treatment 03). There is significant reduction in the crossing number vis-a-vis the graph model. However, it is also clear that a placement with much less wirecrossing in hypergraph model are most likely available with other treatments.

Crossing number correlations (Treatment 03). The correlation experiments are designed to address the question “Will a placement with significantly reduced crossing number in the hypergraph model also reduce the wire length in the same model?” Results of limited testing nad with relatively crude models (and noisy data from the OASIS placement) are indicating that reduced wire length can be expected by reducing the crossing number in the hypergraph model. Significantly, this trend has best correlation coefficients for the isomorphism and mutant classes, but the correlation is insignificant for the

random class!!

Observations not illustrated. The most important observation is that we find insignificant correlation between the crossing number and wire length in the graph model across all equivalence classes – in stark contrast to the notable correlation between the respective parameters in the currently still crude hypergraph model.

Additional experiments with isomorphism and mutant classes for multi-level sequential circuits are reported in Section V.

IV. SEQUENTIAL MUTANT CLASS

Similar to mutation class $(G_2)_{\text{mut}}$ in (2) for bidags, we can introduce a procedure to synthesize a mutation class for multi-level electrical circuits. An electrical circuit, rendered acyclic by deletion of feedback edges (if any) and topologically sorted, can be modeled as a multi-partite bidag. As an extension of the model of a bidag presented in Section II, we model the circuit as a k -partite dag $G_{k,r}$ (the reference dag), with $k=2 \times \text{number of levels in the circuit}$.

$$G_{k,r} = ((V_0, V_1, E_0), (V_1, V_2, E_1), \dots, (V_{i-1}, V_i, E_{i-1}), \dots, (V_{k-1}, V_k, E_{k-1})) \quad (3)$$

We can think of $G_{k,r}$ as k bidags juxtaposed. Like the two-layer bidag $G_{2,r}$, the k -partite bidag has its characteristic signature $\sigma(G_{k,r})$, which is more complicated than $\sigma(G_{2,r})$ for obvious reasons and will be illustrated later in this section. Similarly we can define the mutant class $(G_k)_{\text{mut}}^s$ of the reference k -partite dag (the reference circuit) as

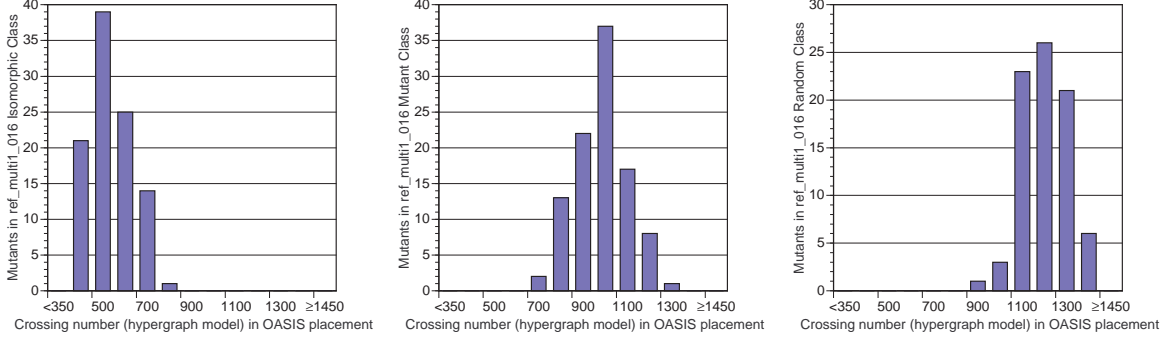
$$(G_{k,r})_{\text{mut}}^s = \{\text{presentation } \langle G'_{k,r}, \pi_0, \dots, \pi_k \rangle | \sigma_{\text{mut}}(G_{k,r})\} \quad (4)$$

In $G_{k,r}$, only bidags $G_{i-1,r} = (V_{i-1}, V_i, E_{i-1})$ where i is an odd number are perturbed to create mutations. In contrast to mutations in stand-alone bigraphs described in Section II, mutations in bidags that are embedded as per $G_{k,r}$ must satisfy a number of additional constraints that we did not require earlier. Furthermore, we must devise a process to break cycles and *extract* $G_{k,r}$ from a sequential circuit netlist (a hypergraph) in the first place, and when mutations are completed for each section $G_{i-1,r} = (V_{i-1}, V_i, E_{i-1})$, we must have the

(a) Crossing number (bidags) reports for all Treatments

Class ref_multi1_16-WD				Class ref_multi1_16-WBB				Class ref_multi1_16-WRD			
Treat-ment	min	mean	max	Treat-ment	min	mean	max	Treat-ment	min	mean	max
00	3941	4393	4970	00	3692	4373	4879	00	480	1683	3327
01	44	114	257	01	145	262	457	01	48	223	944
02	55	65	76	02	122	200	376	02	35	147	301
03	530	1139	2798	03	1616	2400	3249	03	2022	2946	3874

(b) Crossing number (hypergraph) histograms for Treatment 03



(c) Crossing number and wire length correlations for Treatment 03

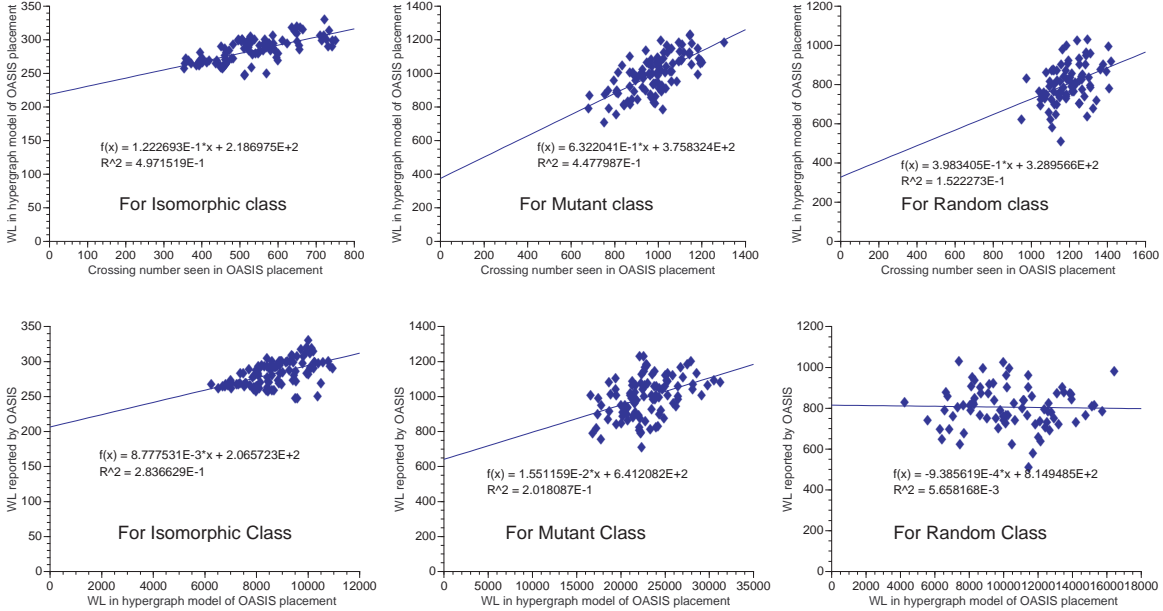


Fig. 3. Partial summary of crossing number and wire length experiments for a number of treatments applied to three equivalence classes (the isomorphic class, WD, the signature-invariant, component-invariant mutant class, WBB, and the random class, WRD) derived from a reference bigraph `ref_multi1_016`.

freedom to reconnect all of the cycles that have been broken. Clearly, the process is complex and can only be outlined in brief steps as follows:

- Form the k -partite form, $G_{k,r}$, from a netlist.
- Determine the characteristic signature $\sigma(G_{k,r})$ of the reference graph $G_{k,r}$.
- Induce perturbations in the reference graph $G_{k,r}$.
- Define mutant equivalence classes $(G_{k,r})_{mut}^s$ of $G_{k,r}$.
- Invoke the mutation process to generate $(G_{k,r})_{mut}^s$ for a sequential reference circuit $G_{k,r}$, with a description of

issues arising due to the presence of cycles in it.

Each of these steps is described in more detail next.

Generating the k -partite Graph – the Canonical Form. The electrical netlist is a hypergraph possibly with cycles. First of all, it is transformed into an acyclic graph and leveled through a topological sort as described below.

- (1) the circuit is first decomposed into a number of strongly connected components (SCCs);
- (2) it is then rendered acyclic by deleting as few feedback edges as possible in each SCC;

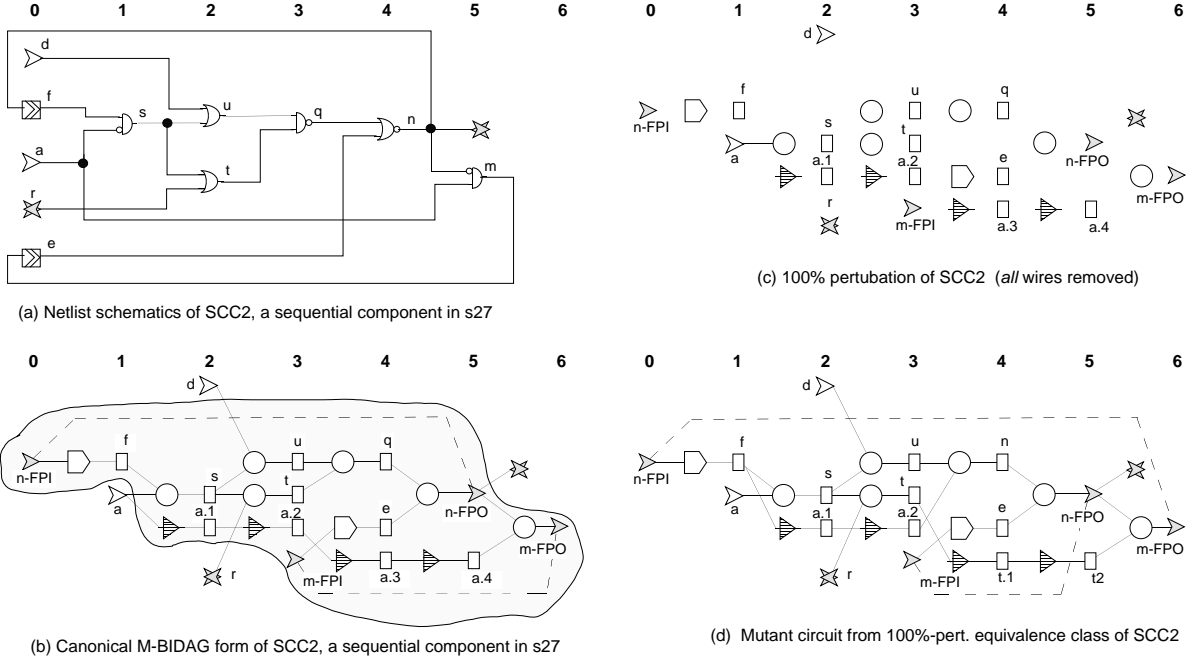


Fig. 4. Representations of one SCC (SCC_1) in the reference circuit and its mutant.

(3) each (deleted) feedback edge is represented by a feedback PO (FPO) at its source and a feedback PI (FPI) at its sink position.

(4) all nets not driven by a cell are assigned a $level = 0$ and designated as *primary inputs* (PIs), FPIs also are assigned a $level = 0$;

(5) all cells are assigned a level > 0 , determined uniquely by the topological sort of the netlist;

(6) each pin of the net inherits the level of the incident cell;

(7) for each net, $netspan^1 > 0$, always;

(8) a pin of the net not driving a cell is designated as a *primary output* (PO).

The graph obtained after these steps is still not k -partite because of the presence of nets spanning more than one level. The k -partite form is obtained through the following transformations. The order of these transformations is important.

T1: For each PI or FPI net, initially at level = 0, and driving cells at level = k_i , re-assign level = $\min\{k_i\} - 1$ since such a PI net does not determine the level of any of the cells it drives. This is due to the cell level topological order assignments.

T2: For each net, driven by pin i (at level j) and $netspan = k + 1$, $k > 1$, (a) connect net i to input pins of all incident cells at level $j + 1$, (b) place a single-input, single-output *feedthrough cell* at level $j + 1$, connect its input pin to pin i , and connect its output pin to input pins of all incident cells at level $j + 2$, (c) repeat step (b) x -times until $j + x = k + 1$;

T3: Replace each net with a net node (a net node mirrors the cell node: its fanin is 1 and fanout is variable).

A canonical form of the netlist in Figure 4(a) is shown in Figure 4(b). For example, the PI net a first has been moved from level 0 to level 1, and then the 2-pin net a with netspan of 5 has been transformed into a net node a , driving a feedthrough node $a.1$, which in turn drives net nodes $a.2$, $a.3$ and $a.4$. Notice one PI-like (r) and one PO-like structure in the canonical form of the SCC. These are component input and component

output respectively, which will be discussed later. With respect to Figure 4(b), it can be readily verified that edges connecting net nodes to cell nodes span levels $i - 1$ and i , while edges connecting cell nodes to net nodes remain at level i .

It may also be seen from Figure 4(b) that the k -partite graph in the canonical form consists of a number of bidags juxtaposed.

Characteristic Signature. The characteristic signature $\sigma(G_{k,r})$ of the k -partite graph is an extension of the signature $\sigma(G_{2,r})$ of the bidag that typifies the different cell and net nodes distribution.

Each distribution is defined for the bidag at level i as follows:

$L_{i,I}$: total number of PI net nodes;

$L_{i,O}$: total number of PO net nodes;

$L_{i,A}$: total number of net nodes driven by feedthrough cells;

$L_{i,1}$: total number of net nodes driven by 1-input combinational cells;

$L_{i,2}$: total number of net nodes driven by 2-input combinational cells;

$L_{i,k,C}$: total number of net nodes driven by k -input combinational cells and driving combinational cells only;

$L_{i,k,B}$: total number of net nodes driven by k -input combinational cells and driving a feedback vertex only;

Level i :	0	1	2	3	4	5	6	Level i :	0	1	2	3	4	5	6	Level i :	0	1	2	3	4	5	6
$L_{i,I}$	0	1	1	0	0	0	0	$\Phi_{i,C}^{min}$	0	0	2	2	1	2	0	$\Phi_{i,C}^{max}$	0	0	2	2	1	2	0
$L_{i,O}$	0	0	0	0	0	0	1	$\Phi_{i,I}^{min}$	0	2	2	0	0	0	0	$\Phi_{i,I}^{max}$	0	2	2	0	0	0	0
$L_{i,A}$	0	0	1	1	1	1	0	$\varphi_{i,A}^{min}$	0	0	1	1	1	0	$\varphi_{i,A}^{max}$	0	0	1	1	1	0	0	0
$L_{i,1,C}$	0	0	0	0	0	0	0	$\varphi_{i,C}^{min}$	0	0	1	1	1	0	$\varphi_{i,C}^{max}$	0	0	2	1	1	2	0	0
$L_{i,2,C}$	0	0	1	2	1	0	0	$\varphi_{i,I}^{min}$	0	1	1	0	0	0	$\varphi_{i,I}^{max}$	0	2	2	0	0	0	0	0
$L_{i,2,S}$	0	0	0	0	0	1	0	$\varphi_{i,A}^{min}$	0	0	1	1	1	0	$\varphi_{i,A}^{max}$	0	0	2	1	1	1	0	0
$L_{i,FPI}$	2	0	0	0	0	0	0	$\Phi_{i,D}^{min}$	0	1	0	0	1	0	$\Phi_{i,D}^{max}$	0	2	0	0	1	0	0	0
$L_{i,D}$	0	1	0	0	1	0	0	$\varphi_{i,D}^{min}$	0	1	0	0	1	0	$\varphi_{i,D}^{max}$	0	2	0	0	1	0	0	0
$L_{i,F}$	0	2	0	0	1	0	0	$\varphi_{i,D}^{min}$	0	1	0	0	1	0	$\varphi_{i,D}^{max}$	0	2	0	0	1	0	0	0
								BASIC SIGNATURE							SIGNATURE EXTENSION								

Fig. 5. An example of a circuit signature and its extension.

¹For each net, $netspan = p_{max} - p_{min}$, where the two numbers denote the maximum and the minimum pin level of the net.

$L_{i,k,S}$: total number of net nodes driven by k -input combinational cells and driving both a feedback vertex only and combinational cells;

$L_{i,D}$: total number of net nodes driven by flip-flops or latches;

$L_{i,FPI}$: feedback PI, total number of feedback net nodes driving a feedback vertex (flip-flop/latch);

$L_{i,FPO}$: feedback PO, total number of feedback net nodes, $L_{i,FPO} = \sum_k \{L_{i,k,B} + L_{i,k,S}\}$ ($k=2$ or 1 normally);

$\Phi_{i,X}^x$: upper ($x = \max$) and lower ($x = \min$) bounds on the total fanout for net nodes of type X , where $X = I$ is a case of a PI net node, $X = A$ is a case of a net node driven by a feedthrough cell, $X = C$ is a case of a net node driven by a combinational cell, $X = D$ is a case of a net node driven by a sequential cells (flip-flop);

$\varphi_{i,X}^x$: upper ($x = \max$) and lower ($x = \min$) bounds on any single node fanout for net nodes of type X , where X is as defined earlier;

q_{i+1} : total number of edges at level $i + 1$ to be driven by net nodes at level i .

For simplicity, we have restricted it to 1- and 2-input combinational cells only. The signature gives rise to *unique bounds on the fanout of the net nodes* shown as a part of the ‘extended signature’. The sequential mutation process must maintain these bounds in addition to the *conditional incidence relationships* that we have developed for the sequential circuits following [3]. However, it is not sufficient for the sequential mutants to conform to the fanout bounds and conditional incidence relationships only. Due to the sequential nature of the problem, they need to also maintain the feedback vertex set invariance with the reference circuit which will be defined later in this section.

Perturbation in the k -partite Graph. A perturbation amounts to the *removal* of p wires or $q\%$ wires from either or both of the channels at level i . An example of 100% wiring perturbation is shown in Figure 4(c). Other degrees of perturbation are essentially the same. The wiring perturbation leaves some or all of the nodes in the circuit floating, and the mutation process consists of restoring, as a random process, *all connections* that have been removed by the wiring perturbations. The circuit mutant does not look isomorphic to its reference circuit in Figure 4(b). However, it shares the same characteristic signature $\sigma(G_{k,r})$ as the reference graph that makes it belong to the mutant equivalence class of $G_{k,r}$.

Mutant Equivalence Class. The mutant equivalence class of $G_{k,r}$ is induced by its signature $\sigma(G_{k,r})$ in the same way as in the bidag model, although the signature for a k -partite graph is more elaborate than that of a bidag. The equivalence class exists due to the uniqueness of $\sigma(G_{k,r})$ (unique under a given feedback vertex set). The *signature-invariant equivalence class* $(G_{k,r})_{\text{mut}}^s$ of a circuit $G_{k,r}$ is the set of all circuits whose signature is $\sigma(G_{k,r})$. We say that these circuits are *mutants* of the reference circuit $G_{k,r}$.

Mutation Process in k -partite $G_{k,r}$. Before we can discuss the mutation process, we need to discuss the issues arising due to the sequential nature of the circuit and the presence of cycles.

FVS (Feedback Vertex Set) Invariance. The sequential mutants must maintain the signature of the reference circuit. This requires maintaining two following two sequential constraints:

S1: Each node on a cycle must have a path to at least one PO of the circuit².

²This condition is required for having a good fault coverage, observability and reachability.

S2: The feedback vertex set (FVS) present in the reference circuit must be preserved³. We maintain this feedback vertex set for all the mutants.

For the sake of brevity, the process for maintaining the FVS invariance is not explained in full details here. In the following we mention the basic concepts towards FVS invariance.

SCC-Graph, G_s . The SCC-graph G_s of a cyclic graph G is defined as a graph (possibly a multi-graph) with vertices as the strongly connected components SCC_1, \dots, SCC_n . There exists k edges⁴ $SCC_i \rightarrow SCC_j$ if there are k nodes in SCC_i that drive some node in SCC_j . The net nodes which drive a node in another SCC are called a **component IO** or **CIO**. It is a component output for the driving SCC and a component input for the SCC driven by the net. The CIO carries loop information from one SCC to another.

SCC-Signature. After casting the whole circuit into the canonical form, we extract the signature of *each* SCC. (The mutation process works only on individual SCCs and not on the whole circuit).

SCC-level. The SCC-levels determine the levels of the SCC nodes in the SCC-graph. The SCC-levels are determined by a topological sort on the SCC-graph.

Feedback tokens (F-type tokens). We associate *feedback tokens* with each feedback vertex in the cyclic graph. In the acyclic graph that we start with, each feedback PI (FPI) issues a token and the token is named after the FPI. When the FPI is connected to a logic node, the token is propagated to the node. Eventually, all the tokens should reach at least one PO of the circuit, and they must also reach the feedback POs (FPOs).

Component tokens (C-type tokens). Component tokens are issued by the component outputs (CO) under certain conditions. If a CO in SCC_j receives feedback tokens $\{f_1, \dots, f_n\}$, at least one of which has not yet propagated to any PO, the CO 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).

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’⁵ tokens (both C and F-type) issued up to level i . In other words, to pick up all the tokens, we need to look up all the nodes in the forward token net set. The goal of the mutation process is to *maximize* the forward token net set. The bounds on the forward token net set is omitted here for the sake of brevity.

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. The goal of the mutation process is to *minimize* this set. The bounds on the forward token net set is omitted here for the sake of brevity.

Without further explanation we make a note here that maintaining the two types of net sets are critical for maintaining the FVS invariance.

Synthesis of Mutants. The *reference* netlist is characterized to extract (1) the signature, (2) the set of feedback vertices, (3) boolean logic implemented by each logic node and (4) a hierarchical description of the netlist in terms of the strongly connected components (SCCs) in the circuit. A signature is extracted for each SCC. The mutation process works on individual SCCs, following a certain order. The order is determined by a topological sort on the SCC-graph. The mutant graph is obtained by flattening the hierarchy of mutant

³Note that the feedback vertex set is not unique. The problem of finding a minimum feedback vertex set is known to be NP-hard. We find a minimized feedback vertex set using a polynomial time algorithm.

⁴Implying that the graph may be a multi-graph.

⁵A token is said to be alive if it is yet to propagate to a PO.

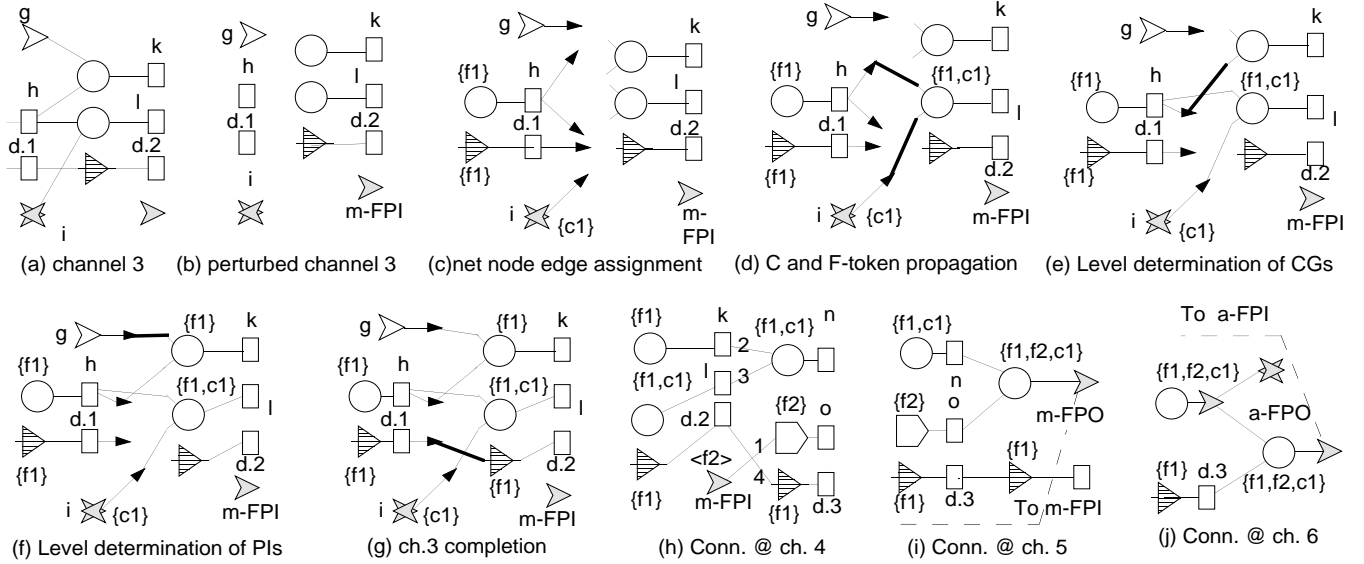


Fig. 6. Representations of a reference circuit and its mutant.

SCCs. The signature of the mutant circuit is compared with that of the reference to verify that the mutant belongs to the same equivalence class.

Connection Assignments. We consider *four* connection assignments at each level i , $i > 0$. As mentioned before, the mutation is carried out only for bidags (V_{i-1}, V_i, E_{i-1}) where i is odd.

(A1): *bounded net node edge connection.* 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 through restricted cell node edge connection.* Here, both F and C-type tokens are combined on a cell nodes such that the bounds on the size of the forward token net set is satisfied.

(A3): *feedback loop completion* Cells designated as FPOs are connected, to token-carrying net nodes at input side and FPIs at the output side, such that the FPI gets back the token that it issued.

(A4): *restricted cell node edge connection* Here, q_{i+1} edges driven by net nodes level i are connected to q_{i+1} input pins of cell node at level $i + 1$, subject to forbidden permutation positions;

Illustrative Example. We make use of the single mutation channel i (of SCC_1 of the reference circuit s27) in Figure 6 to illustrate the process. Figure 6(b) is a 100% perturbed version of the slice 3 in Figure 6(a). Note that only wires in mutation channel are deleted.

Figure 6(c) illustrates the fanout assignment to each net node under certain bounds. In Figure 6(d), the forward token net set is formed. Note that tokens f_1 (F-type) and c_1 (C-type) are ‘alive’ here, *viz.*, they need to be propagated towards some PO. 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 6(e) and (f) perform connections to ensure the levels of combinational gates (CGs) and PIs, as dictated by the connection constraints. All the remaining connections are completed in Figure 6(g). (In Figure 6(d)–(g), connections at each stage are shown by bold edges).

Connections in Figure 6(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 6(i), the feedback loop is completed. It may be noted that in the reference circuit, the cell was connected to FPI a-FPI. However, in the mutant it can drive either of the FPIs, a-FPI or m-FPI, since it receives tokens issued by both the FPIs. By random choice we connected the cell to m-FPI. Finally, in Figure 6(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 to this SCC have been propagated to another component (SCC_0 in this case), the mutation process is considered to be successful for SCC_1 .

V. EXPERIMENTAL RESULTS

We report results with two equivalence mutant classes of the benchmark s1423 – the 0% mutation class or the isomorphic class, and the 100% mutation class. In this experiment we have chosen 100 mutants for each of the 0% and 100% classes.

The tools we used in this experiments are: (1) DOT [7] to generate circuit schematics and analyze the number of wire crossings; (2) PROP [9] to generate a mincut of a balanced bipartition; (3) OASIS [6] to layout the circuits and report layout area and wire length; (4) Verilog to simulate the circuits for estimating their entropy.

The results of these experiments are summarized with histograms in Figure 7.

Figure 7(a): Here we report distributions of the number of wire crossings, as optimized by DOT, in the mutant classes. The histogram indicates the number of mutants having wire crossing in a particular range.

Figure 7(b): Here we report the size of the mincut for a balanced bipartition. As can be seen in the figure, the two classes do induce a distribution in mincuts. The two classes offer a varied degree of difficulty to the bipartitioning tool under investigation. It remains to be seen how another tool might behave with the two sets of mutants.

Figure 7(c), (d): Here we report the area and wire length for the layout tool in OASIS. Again, the two parameters show a near-normal distribution for both the classes. It will be interesting to see the performance of another tool on these mutants.

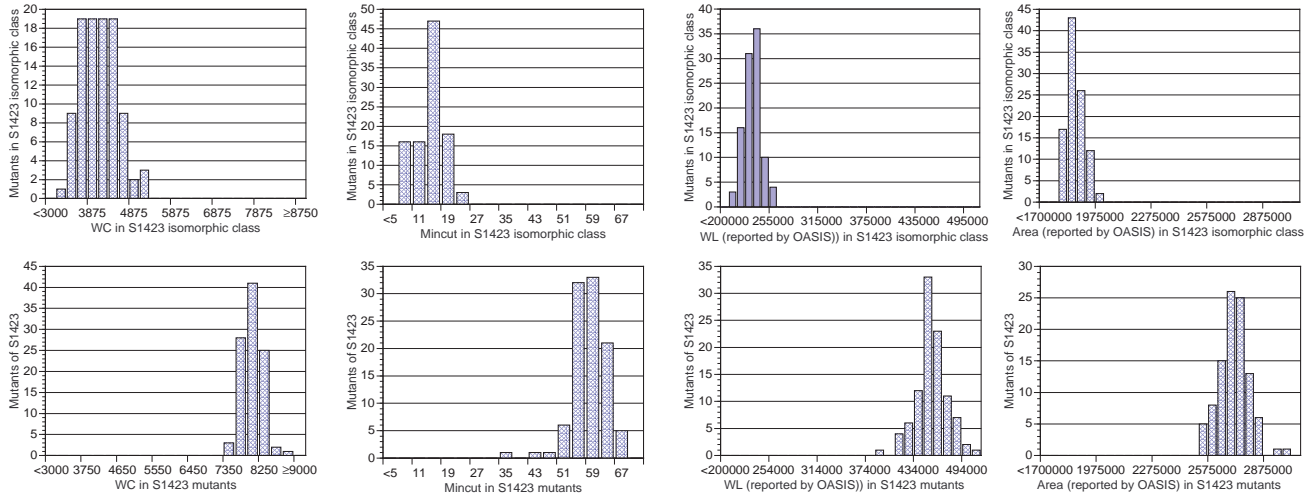


Fig. 7. Results of experiments with two equivalence classes of a sequential circuit (s1423)

Entropy Simulation: Entropy calculation, based on Verilog simulation of each mutant, showed that every circuit is 'alive', *viz.*, not stuck at a particular state. Results are omitted here for the sake of brevity.

CONCLUSION AND FUTURE WORK

We have demonstrated that

- (1) benchmarking CAD algorithms cannot be done based on a few experiments/test cases
- (2) it is feasible to synthesize a large number of equivalence class mutants of sequential circuits that can serve as the 'classes' needed in such benchmarking process,
- (3) the mutants show excellent correlation with physical design parameters, and hence, they are not random circuits,
- (4) the equivalence classes induce significant distributions in terms of algorithm performance metrics.

The mutants find applications mostly in benchmarking physical design algorithms.

Future works include a proper design of experiments with these mutants and subsequent statistical analysis of results.

ACKNOWLEDGMENTS. We appreciate the access to remote servers and tools used in this research: user accounts on two remote servers, facilitated by Dr. Richard Newton at UC Berkeley and Dr. Gershon Kedem at Duke U., SIS and WELD from teams at UC Berkeley, PROP from Roman Kužnar from U. of Ljubljana (Slovenia), and APR from Xilinx Inc.

REFERENCES

- [1] Michael Hutton, J.P. Grossman, J. Rose and D. Corneil. Characterization and Parametrized Random Generation of Digital Circuits. In *ACM/IEEE 33rd Design Automation Conference*, June 1996.
- [2] 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.
- [3] D. Ghosh, N. Kapur, J. E. Harlow, and F. 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-DATE-Ghosh>.
- [4] K. Iwama and K. Hino. Random Generation of Instances for Logic Optimizers. In *ACM/IEEE 31st Design Automation Conference*, pages 430–434, June 1994.
- [5] M. R. Garey and D. S. Johnson. Crossing Number is NP-complete. *SIAM J. Algebraic Discrete Methods*, 4:312–316, 1983.

- [6] 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).
- [7] E.R. Gasner, E. Koutsifios, S.C. North and K.P. Vo. A Technique for Drawing Directed Graphs. *IEEE Trans. Software Engg.*, 19:214–230, 1993.
- [8] Heuristics and Experimental Design for Bigraph Crossing Number Minimization. Technical report. Information about authors withheld to preserve anonymity.
- [9] R. Kužnar and F. Brglez. PROP: A Recursive Paradigm for Area-Efficient and Performance Oriented Partitioning of Large FPGA Netlists. In *ICCAD '95*, pages 644–649, 1995.
- [10] Correlating wire crossing, wirelength, and channel density for a large number of placements. Technical report. Information about authors withheld to preserve anonymity.
- [11] M. Marek-Sadowska and M. Sarrafzadeh. The Crossing Distribution Problem. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 14(4):423–433, April 1995.
- [12] J. N. Warfield. Crossing Theory and Hierarchy Mapping. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-7(7):505–523, July 1977.