Synthesis of Wiring Signature-Invariant Equivalence Class Circuit Mutants and Applications to Benchmarking

Debabrata Ghosh[1] Nevin Kapur[1] Justin Harlow III[2] Franc Brglez[1]

[1]CBL (Collaborative Benchmarking Lab), Dept. of Comp. Science, Box 7550, NC State U., Raleigh, NC 27695, USA

[2]National Semiconductor Corporation, Santa Clara, CA 95052
http://www.cbl.ncsu.edu/

Abstract – *This paper formalizes the synthesis process of wiring signature-invariant (WSI) combinational circuit mutants. The signature σ_0 is defined by a* reference circuit η_0, *which itself is modeled as a canonical form of a directed bipartite graph. A wiring perturbation γ induces a* perturbed reference circuit η_γ. *A number of mutant circuits η_{γ_i} can be resynthesized from the perturbed circuit η_γ. The mutants of interest are the ones that belong to the wiring-signature-invariant equivalence class $\mathcal{N}_{\sigma 0}$, i.e. the mutants $\eta_{\gamma_i} \in \mathcal{N}_{\sigma 0}$.*

Circuit mutants $\eta_{\gamma_i} \in \mathcal{N}_{\sigma 0}$ have a number of useful properties. For any wiring perturbation γ, the size of the wiring-signature-invariant equivalence class is huge. Notably, circuits in this class are not random, *although for* unbiased *testing and benchmarking purposes, mutant selections from this class are typically random.*

For each reference circuit, we synthesized eight equivalence subclasses of circuit mutants, based on 0 to 100% perturbation. Each subclass contains 100 randomly chosen mutant circuits, each listed in a different random order. The 14,400 benchmarking experiments with 3200 mutants in 4 equivalence classes, covering 13 typical EDA algorithms, demonstrate that an unbiased random selection of such circuits can lead to statistically meaningful differentiation and improvements of existing and new algorithms.

Keywords: signature-invariance, equivalence class, circuit mutants, benchmarking.

I. Introduction

Today, we conduct experiments and report on 'performance' of EDA algorithms on the basis of unrelated instances of circuit benchmarks. In this paper, we argue that benchmarking EDA tools and algorithms for designing VLSI circuits in submicron-technology requires a new approach. Case-by-case evaluations, however detailed, of a few unrelated benchmark circuits are not likely to reveal a set of statistically consistent results that can drive and support important improvements for the new generation of algorithms and tools. However, by making a case for

(1) a near-infinite supply of equivalence classes of circuits, with properties of each class such as
- the *same* number of I/Os,
- the *same* number of cell nodes and cell types,
- the *same* number of cell node pins distributed across the *same* number of cell node levels,

and

(2) unbiased random selection of sufficiently large subsets of circuits from each of the equivalence classes,

we argue that selection of such circuits can and shall lead to statistically meaningful differentiation and improvements of existing and new algorithms.

Authors from NC State U. were 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).

Random graphs, a potentially unlimited source of circuit benchmarks, have not been accepted as realistic – until recently. New approaches to random circuit generation that take into consideration constraints of digital circuits have been reported [1, 2]. Both approaches have been motivated by the need to generate a large number of circuits to test FPGA architectures. The approach in [3] introduces logic-invariant transformations to generate a number of logically equivalent circuits. However, sizes of circuits in the same class can vary widely, *e.g.*, from 21 to 1894 nodes.

In this paper, we introduce the following concepts:

(1) A *reference circuit* η_0 and its wiring signature σ_0. The reference circuit may represent a case of a real design.

(2) A wiring signature and the equivalence class $\mathcal{N}_{\sigma 0}$ induced by it.

(3) A wiring perturbation γ resulting in a *perturbed reference circuit* η_γ.

(4) A synthesis procedure for *any* number of circuits in the same equivalence class, i.e. the mutants $\eta_{\gamma_i} \in \mathcal{N}_{\sigma 0}$.

The paper is organized into the following sections: (2) reference circuit canonical form and wiring perturbations; (3) reference circuit wiring signature; (4) synthesis of wiring signature-invariant (WSI) circuits; (5) proposal for design of experiments; (6) summary of 14,400 benchmarking experiments with 3200 circuits; (7) conclusions.

II. Canonical Form

A graph-based model of a netlist is not effective for the problems we consider. On the other hand, a model of a netlist as a *directed hypergraph* is not unique. A 'star model' or 'Steiner point model', representing multi-terminal nets, such as illustrated in Figure 1(a), is typical only when rendering a netlist schematic. The example shown is from [4]. To improve legibility of the schematic, a single net with high cardinality may be rendered with a variable number of 'Steiner points'. As to netlist labeling, the *lower case* labels in Figure 1 refer to nets; since all cells have a single output pin, we can infer the label of the cell from its net label. For example, the three-pin net r is driven by the cell node R, without labeling the cell explicitly.

The nature of our problem is such that it can be readily overloaded with formal notation and detract from the message. When discussing a netlist as a directed hypergraph, we refer to cells (or cell nodes) and nets in topological order. We use the notion of cell level, levels of net pins, and *netspan*[1]. The *canonical form of a bipartite directed graph*, a multi-level graph structure of alternating sets of net nodes and cell nodes, is a simple transformation of the underlying netlist: levels of some of its pins are redefined, and a new type of cell node, a *feedthrough cell* is introduced. The salient property of this form is that the netspan of all edges in this graph is well-defined: edges connecting net nodes and cell

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

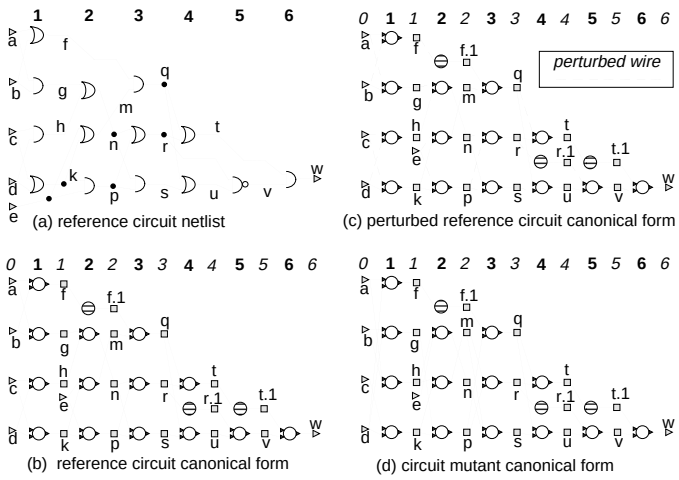


Fig. 1. Representations of a reference circuit and its mutant.

nodes have netspan = 1, edges connecting cell nodes and net nodes have netspan = 0. Interchangeably, we may refer to these edges as *wires* or *two-pin nets*. Also, we will refer to the set of all edges with netspan = 1 at level i and nodes incident at these edges as the *mutation channel* at level i. Similarly, we will refer to the set of all edges with netspan = 0 at level i and nodes incident at these edges as the *permutation channel* at level i. The motivation for these definitions will become clear later in this section.

Acyclic Netlist Model. In this paper, we only consider netlists that are acyclic, where each cell drives only one net, and is driven by one or more nets. The unique characteristics of a netlist that we capture before its transformation into a canonical bipartite directed graph model are the following:

(1) all nets not driven by a cell are assigned a *level = 0* and designated as *primary inputs* (PIs);

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

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

(4) for each net, netspan > 0, always;

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

An extension of the characterization for netlists with cycles and its canonical form will be presented elsewhere.

C-BIDAG Model. The canonical bipartite directed graph model (C-BIDAG) is a transformation of an acyclic netlist. The order of this transformation is important.

- **T1:** For each PI 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 1(a) is shown in Figure 1(b). For example, the PI net e has been moved from level 0 to level 1, and the 3-pin net r with netspan of 2 has been transformed into a net node r driving a feedthrough node *R.1* which in turn drives a net node *r.1*. With respect to Figure 1(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.

Wiring Perturbation Model of C-BIDAG. A wiring perturbation is defined for both the mutation channel and the permutation channel at level i as defined earlier. Basically, it amounts to the *removal* of p wires or q% wires from either or both of the channels at level i; starting at level 1 and completing at the highest level.

An example of 60% wiring perturbation is shown in Figure 1(c). For example, at level 1, there are a total of 8 wires in the mutation channel. A 60% perturbation implies a removal of 5 wires. A level-wise distribution of all 60% wiring perturbations in the mutation channels for this circuit is thus $\{5, 4, 4, 3, 1, 1\}$. The upper bound on the number of mutants for this perturbation is 288x18x6x9x1x1 = 279,936. Results using 100 randomly selected mutants show a surprising range of variations of optimizing objectives for the algorithms considered – even for such a small reference circuit.

Reference Circuit Mutation. A reference circuit mutation is induced by the p-wire or q%-wire perturbation of the reference circuit in C-BIDAG form. The wiring perturbation leaves some 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. An example of a circuit mutant induced by the 60% wiring perturbations in Figure 1(c) is shown in Figure 1(d). The circuit mutant does not look isomorphic to its reference circuit in Figure 1(b). However, it has a number of properties that make it belong to the mutant equivalence class as defined next.

III. Wiring Signature Invariance

As demonstrated, introducing edge perturbations and reconnecting the nodes can change the structure of the graph dramatically . The question arises what properties about the graph can be maintained as invariant during this process while still creating a diversity of useful graph mutations. The wiring signature, illustrated in Figure 2, is subject to *conditional* incidence relationships and *bounds* on the net node fanout and has the required properties.

Wiring Signature. The reference circuit representation in the canonical form as defined in this paper suggests a circuit signature as a *distribution of characteristic cuts* in a level-order.

Each *characteristic cut* is defined for 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;

$\Phi^x_{i,X}$: upper ($x = max$) and lower ($x = min$) bounds on the *total fanout* for net nodes of type X, where $X = I$ is the

Level i :	0	1	2	3	4	5	6
$L_{i,I}$:	4	1	0	0	0	0	0
$L_{i,O}$:	0	0	0	0	0	0	1
$L_{i,A}$:	0	0	1	0	1	1	0
$L_{i,1}$:	0	0	0	0	0	0	0
$L_{i,2}$:	0	4	3	3	2	1	1

BASIC SIGNATURE

Level i :	0	1	2	3	4	5	6
$\Phi^{min}_{i,C}$:	0	3	3	2	2	1	1
$\Phi^{min}_{i,I}$:	8	1	0	0	0	0	0
$\Phi^{min}_{i,A}$:	0	0	1	0	1	1	0
$\varphi^{min}_{i,C}$:	0	1	1	1	1	1	1
$\varphi^{min}_{i,I}$:	1	1	0	0	0	0	0
$\varphi^{min}_{i,A}$:	0	0	1	0	1	1	0
q_{i+1} :	0	8	7	6	5	3	2

Level i :	0	1	2	3	4	5	6
$\Phi^{max}_{i,C}$:	0	6	5	5	2	1	1
$\Phi^{max}_{i,I}$:	8	3	0	0	0	0	0
$\Phi^{max}_{i,A}$:	0	0	3	0	1	1	0
$\varphi^{max}_{i,C}$:	0	3	3	3	1	1	1
$\varphi^{max}_{i,I}$:	4	3	0	0	0	0	0
$\varphi^{max}_{i,A}$:	0	0	3	0	1	1	0

SIGNATURE EXTENSION

Fig. 2. An example of a circuit signature and its extension.

case of a PI net node, $X = A$ is the case of a net node driven by a feedthrough cell, $X = C$ is the case of a net node driven by a combinational cell;

$\varphi^{x}_{i,X}$: upper ($x = max$) and lower ($x = min$) bounds on *any single node fanout* for net nodes of type X, where $X = I$ is the case of a PI net node, $X = A$ is the case of a net node driven by a feedthrough cell, $X = C$ is the case of a net node driven by a combinational cell;

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

To keep the size of the signature manageable, we have restricted it to 1- and 2-input combinational cells only. The signature for a general sequential circuit is more complex and will be presented elsewhere. Specifically, the wiring signature for the reference circuit in Figure 1(b) is shown in Figure 2. We refer to the first 5 rows on the leftmost column as the 'basic signature', and the remainder as the 'extended signature'. For example, note that at level 2, the basic signature records the presence of 3 net nodes driven by 2-input cell nodes, and 1 net node driven by a feedthrough cell. While the statistics contained in the 'basic signature' may appear trivial at first glance, they give rise to *unique bounds on the fanout of the net nodes* shown as a part of the 'extended signature'. An example of these bounds will be discussed later in the section.

Conditional Incidence Relationships. Restoring the connections after *any perturbation* of the reference circuit canonical form, including the removal of *all wires*, may result in a circuit whose wiring signature will be different from the one generated for the reference circuit. The canonical form graph imposes a set of unique conditions on how net nodes may drive the cell nodes such that the wiring signature is maintained when reconnecting the edges that have been removed between the net nodes and cell nodes. Table I summarizes all of the *conditional* incidence relationships of net nodes at level $i-1$ to cell nodes at level i that must be maintained to restore a mutant circuit whose signature will match the signature of the reference circuit under any perturbation.

Net Node Fanout Bounds. Given the basic signature, the bounds on the fanouts of each net node are invariant. These are included as a signature extension in Figure 2. For example, at level 2, the minimum fanout on the net nodes of type C is 3 and the maximum is 5. The corresponding values for fanouts on individual net nodes of type C are 1 and 3 respectively.

During mutation, the signature is maintained for any choice of $\Phi_{i,X}(\varphi_{i,X})$ provided that $\Phi_{i,X} \in [\Phi^{min}_{i,X}, \Phi^{max}_{i,X}]$ and $\varphi_{i,X} \in [\varphi^{min}_{i,X}, \varphi^{max}_{i,X}]$, $X \in \{C, A, I\}$.

As an example, $\Phi^{min}_{i,C}$ is given by

$$\Phi^{min}_{i,C} = \max\left(L_{i,1} + L_{i,2}, \quad L_{i+1,2} + L_{i+1,1}\right) \tag{1}$$

The formulation of $\Phi^{min}_{i,C}$ is dictated by the fact that the level of each combinational gate at level $i+1$ has to be asserted by an edge from a net node of type C at level i (condition **B1** in Table I), and, each net node must drive at least one edge. Maximum number of edges driven by C-type net nodes is given by:

$$\Phi^{max}_{i,C} = \min\left(\Phi^{1,max}_{i,C}, \quad \Phi^{2,max}_{i,C}\right) \tag{2}$$

where $\Phi^{1,max}_{i,C} = q_{i+1} - L_{i,I} - L_{i,A}$ and $\Phi^{2,max}_{i,C} = 2 \times L_{i+1,2} - L_{i,I} + \min(L_{i,2} + L_{i,1}, L_{i+1,1}) + \min(L_{i,2} + L_{i,1}, L_{i+1,A})$. The complete set of bounds (a total of 12), required to preserve the signature, is omitted. Note that, within the bounds, the reference circuit and the mutant may have a different distribution across the different types of net nodes. However, all mutants share the same signature extension, *i.e.* the same fanout bounds, with the reference circuit.

TABLE I
Conditional incidence relationships for the canonical form.

Net nodes at level $i-1$	Cell nodes at level i: ft[a]	lg[b]	Conditions
lg-driven	0	1	**A1**: cannot drive the same node twice
lg-driven	1	0	**A2**: cannot drive more than 1 feedthrough
lg-driven	1	1	**A3**: must satisfy **A1** and **A2**
ft-driven	0	1	**B1**: level of cell at level i must be asserted by a lg-driven net node at level $i-1$
ft-driven	1	0	**B2**: cannot drive more than 1 feedthrough
ft-driven	1	1	**B3**: must satisfy **B1** and **B2**
PO(term.)	0	0	**C0**: must be lg–driven at level $i-1$
PO(driving)	0	1	**C1**: must satisfy **C0**
PO(driving)	1	0	**C2**: must satisfy **C0** and **A2**
PO(driving)	1	1	**C3**: must satisfy **C0** and **A3**
PI[c]	0	1	**D1**: must satisfy **B1**
PI	1	0	**D2**: must not be allowed
PI	1	1	**D3**: must satisfy **B1** and **B2**

[a] feedthrough cell nodes
[b] logic cell node
[c] At least one PI node is at level 0

Wiring Signature-Invariant Class. Given a topologically sorted acyclic netlist model as defined in this paper, the wiring signature of its C-BIDAG model is unique. Consider a reference circuit η_0 and its wiring signature σ_0. A wiring-signature-invariant equivalence class $\mathcal{N}_{\sigma 0}$ is the set of all circuits whose wiring signature is σ_0. We say that these circuits are mutants of the reference circuit η_0. Specifically, the mutants are induced by a wiring perturbation γ and are denoted as $\eta_{\gamma_i} \in \mathcal{N}_{\sigma 0}$. In Section 5, we introduce 8 specific subclasses (A–H) of the wiring signature invariant mutants. The subclasses are differentiated by different degrees of perturbation (γ).

IV. Synthesis of Mutants

This section presents a synthesis procedure for *any* number of circuit mutants. The goal of the synthesis process is to reconnect the removed wires in a perturbed circuit such that the signature of the reference circuit is restored.

The synthesis algorithm `MUTATE`, outlined in this section, relies on the following theorem (proof omitted for brevity).

Theorem: Given the *basic signature* of the reference circuit, any circuit mutant satisfying the fanout bounds of the extended signature will remain in the same equivalence class.

Connection Assignments. We consider three connection assignments at level $i, i > 0$:

(A1): bounded net node edge connection (mutation channel). 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): restricted cell node edge connection (mutation channel). Here, q_{i+1} edges driven by net nodes at level i are connected to q_{i+1} input pins of cell node at level $i+1$, subject to forbidden permutation positions;

(A3): unrestricted permutation connection (permutation channel). Here, p_{i+1} edges driven by cell nodes at level $i+1$ drive p_{i+1} net nodes at level $i+1$. This step is well understood.

Illustrative Example. We make use of the single mutation channel i in Figure 3 to illustrates the process.

(a) The complete signature for the reference channel is $\{(1,0,1,1,1),([3,5],[1,3],[1,2]),([1,4],[1,3],[1,2]),7\}$. For readability, the signature is shown in four parts:

- basic signature;

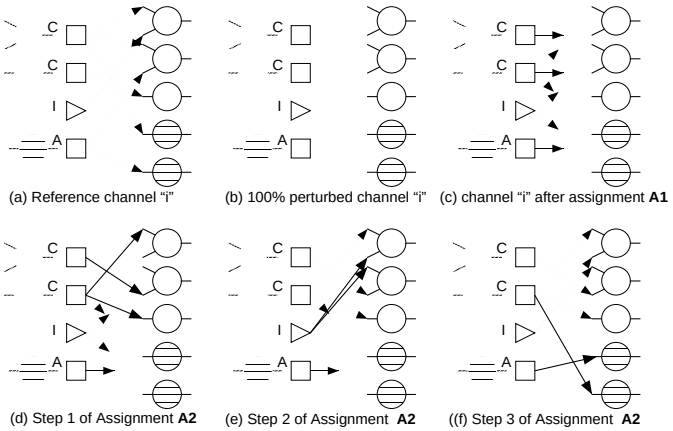


Fig. 3. Single channel mutation sequences.

- extended part of the signature relating to bounds on total fanout of net nodes of type C, I, A;
- extended part of the signature relating to bounds on individual fanout of net nodes of type C, I, A;
- number of cell input pins to be driven by net nodes at level i.

(b) Here we show the case of 100% wiring perturbation. The case of partial perturbation can be presented similarly.

(c) This is an illustration of the bounded net node edge connection (Assignment *(A1)*). According to the signature, the bound on C-type net nodes is $[3, 5]$. Here, the random choice returns 4 edges. Similarly, the bound on I-type net nodes is $[1, 3]$, and the bound on A-type net nodes is $[1, 2]$. The random choice returns 2 edges and 1 edge, respectively. Since there are 2 C-type nodes, we need to distribute the 4 C-type edges to individual net nodes such that the fanout on either net node does not exceed the individual fanout bound $[1, 4]$. Formally, the assignment is always done in the following order : $\Phi_{i,C}, \Phi_{i,I}, \Phi_{i,A}, \varphi_{i,C}, \varphi_{i,I}, \varphi_{i,A}$.

(d, e, f) This is an illustration of the restricted cell node edge connection (Assignment *(A2)*). The following three steps, in the strict order shown, complete *all* of the perturbed circuit connections.

- connect any edge driven by C-type net node to an input pin of a logic node such that condition **B1** in Table I is satisfied;
- connect any edge driven by I-type net node to an input pin of a logic node whose level has already been asserted by a C-type net node (this step satisfies conditions **D1, D2, D3** in Table I);
- connect all remaining edges driven by any net node type to an input pin of a logic or feedthrough cell node under the constraints of Table I.

Pseudo–code of MUTATE. In Figure 4, we show a section of pseudo-code of the `MUTATE` algorithm, notably the connection assignments *A1–A2*. Essentially, the mutation process consists of a sequence of connections from net nodes to cell nodes and *vice versa*. To generate each mutant, the process progresses from level 0 to the highest level. Selection of a specific pair of {net nodes, cell node} is done by a random choice out of a feasible set that conforms to the constraints in Table I and the fanout bounds such as Eqs. (1–2).

Time and Memory Complexity of the Algorithm. The algorithm always works on two adjacent levels and the computations at level i does not depend on any of the results of levels $\leq i-2$. Hence the process may be described as **memoryless**. The storage complexity is $O(m)$ where m is the maximum size of all the levels. The time complexity of the algorithm can be stated as $O(\mid V \mid + \mid E \mid)^k$, where $\mid V \mid$ is the number of cell nodes in the circuit and $\mid E \mid$ is the number of nets and $1 \leq k \leq 2$.

for # of mutants to be generated {
for each level i from 0 to max_level {
`do_assgn_A1; do_assgn_A2; do_assgn_A3; } }`
function do_assgn_A1() {
$\Phi_{i,C}$ =`rand_assgn(`$\Phi_{i,C}^{min}, \Phi_{i,C}^{max}$`)`; $remainder = q_{i+1} - \Phi_{i,C}$;
$\Phi_{i,I}$ =`rand_assgn(`$\Phi_{i,I}^{min}$, $\min\left(\Phi_{i,I}^{max},\ remainder\right)$ `)`;
$remainder = remainder - \Phi_{i,C}$; $\Phi_{i,A} = remainder$;
for each net node of type C {
remainder = fanouts left in $\Phi_{i,C}$;
$\varphi_{i,C}$ =`rand_assgn(`$\varphi_{i,C}^{min}$, $\min\left(\varphi_{i,C}^{max},\ remainder\right)$ `); }`
for each net node of type I {
remainder = fanouts left in $\Phi_{i,I}$;
$\varphi_{i,I}$ =`rand_assgn(`$\varphi_{i,I}^{min}$, $\min\left(\varphi_{i,I}^{max},\ remainder\right)$ `); }`
for each net node of type A {
remainder = fanouts left in $\Phi_{i,A}$;
$\varphi_{i,A}$ =`rand_assgn(`$\varphi_{i,A}^{min}$, $\min\left(\varphi_{i,A}^{max},\ remainder\right)$ `); }}`
function do_assgn_A2() {
`for each cell node y at i+1 { pick x at random from` $\Phi_{i,C}$ `and connect it to y }`
`for each net node y of type I at level` i `{`
`pick x at random from` $L_{i+1,2}$ `and connect y to x }`
`for each` φ_i `which is not yet connected {`
`calculate feasible_set of cell nodes;`
`pick x at random from the feasible_set and connect to the net node }}`

Fig. 4. A section of `MUTATE` pseudo–code

V. Design of Experiments

The capability to synthesize a large number of WSI circuit mutants, based on wire perturbation classes, motivates us to examine the *sampling methods* that arise in the *design of experiments*. Such methods, first formalized in [5], have been adopted widely in many fields of science. In this paper, we adapt them to analyze the performance of important graph-based algorithms in the context of EDA. For each reference circuit, we propose to synthesize equivalence subclasses of circuit mutants, based on 0 to 100% perturbation. Each subclass contains 100 randomly chosen mutant circuits, each listed in a different random order. This sample size is large enough for the sampling distributions to be considered normal or nearly normal; the population parameters may be estimated closely by their corresponding sample statistics. The eight equivalence subclasses, labeled from A to H, are defined in terms of the perturbations we use to generate each class. In order to encourage unbiased experiments with these classes, we have *permuted the perturbations* relative to the label assignments:

$$\{A, B, C, D, E, F, G, H\} = \text{permutation}\{0w, 1w, 2w, 5\%, 10\%, 20\%, 40\%, 100\%\} \quad (3)$$

In (3), WSI classes A-H are defined either in terms of q-% wire perturbations or 0-wire, 1-wire, 2-wire $(0w, 1w, 2w)$ perturbations. We plan to identify the labels A-H in terms of the respective perturbation classes in (3) later, once there are additional experiments reported by others and participants have the opportunity to meet and present their results at a joint session of a conference. More details about such plans can be found under http://www.cbl.ncsu.edu/experiments/.

A case study tutorial of average-case performance of two algorithms and their differences has demonstrated that up to six distinct equivalence classes of data are useful to render an unbiased comparison of two well-known sorting algorithms

[6]. The long-term goal of this series of experiments, presently starting with eight equivalence classes, is to facilitate generation of similar comparisons for the more complex and diverse algorithms in EDA. Whatever may be decided about the most suitable number of equivalence classes through wider participation later on, the class of 0-wire perturbations will remain important. As demonstrated in this paper as well as earlier [7], the objective functions used in a number of graph-based algorithms can be very sensitive to the *order of nodes* in the graph, even when graphs are isomorphic. In our experiments, the 100 netlists in the 0-wire perturbation class are simply isomorphic instances of the reference netlist in a randomized order.

Paraphrasing the context of the traditional *treatments and blocks* [8], we propose to archive data in the context of *algorithms and equivalence class mutants* as shown in Figure 5(a). For each of the 'a' algorithms we consider 'b' mutants in one of the equivalence classes in (3). For each algorithm `Alg_j` and mutant `M-X_k`, $X \in \{A, \ldots, H\}$, we record two observations: the *initial value* of the objective function tuple `X_jk-I`, and the *final value* of the objective function tuple `X_jk-F`. The initial value corresponds to a *placebo treatment* of the mutant `M-X_k`: it is the value of the objective function *before* engaging the algorithm to optimize it. The final value corresponds to the optimized value of the objective function *after* engaging the algorithm to optimize it.

A number of analyses can be performed once data is archived as shown in Figure 5(a) and only a few are discussed in this paper. For the most part, we shall concentrate on analyzing data as presented in Figure 5(b). In particular, for samples associated with each algorithm `Alg_j` and mutant class `M-X`, $X \in \{A, \ldots, H\}$, we evaluate the 95% confidence interval of the sample mean, the sample mean, and the sample variances as tuples `{M-X_j-I}` and `{M-X_j-F}` respectively. We summarize such evaluations in the form shown in Figure 5(c). The next section provides representative summaries of data samples we generated and archived under http://www.cbl.ncsu.edu/experiments/.

VI. Experiments

This section summarizes experiments based on four classes of WSI circuit mutants, based on reference circuits V65E3-64, V65E1-66, C499, and C1355. The first two reference circuits belong to a family of 2-layer graphs introduced in this paper, the other reference circuits belong to the ISCAS85 set [9] and are also documented in [10]. With eight subclasses A-H and 100 mutant circuits in each class, we summarize a total of 14,400 experiments covering representative cases of 13 algorithms.

Complete tables of *all* data samples summarized in this paper have been archived on our web site (http://www.cbl.ncsu.edu/experiments/). The archives are being updated periodically with *additional experiments* and cases of more detailed statistical analyses [11]. The web site provides an open forum to interested researchers for further sampling, *tests of significance and hypotheses*, and *statistical inference* of existing data and benchmarks, as well as for contributing new cases of benchmarks, new data of experiments, and new cases of statistical analysis. The site will maintain contributions of participants either as hyperlinks to data and documents on participant's web site, or new archives will be created under http://www.cbl.ncsu.edu/experiments/.

In this section we describe briefly the context of the experiments and provide a 'textbook' statistical summary of uncorrelated *95% confidence interval of the sample mean*, the *sample mean* and the *sample standard deviation*. We only briefly allude to some fundamental issues, such as

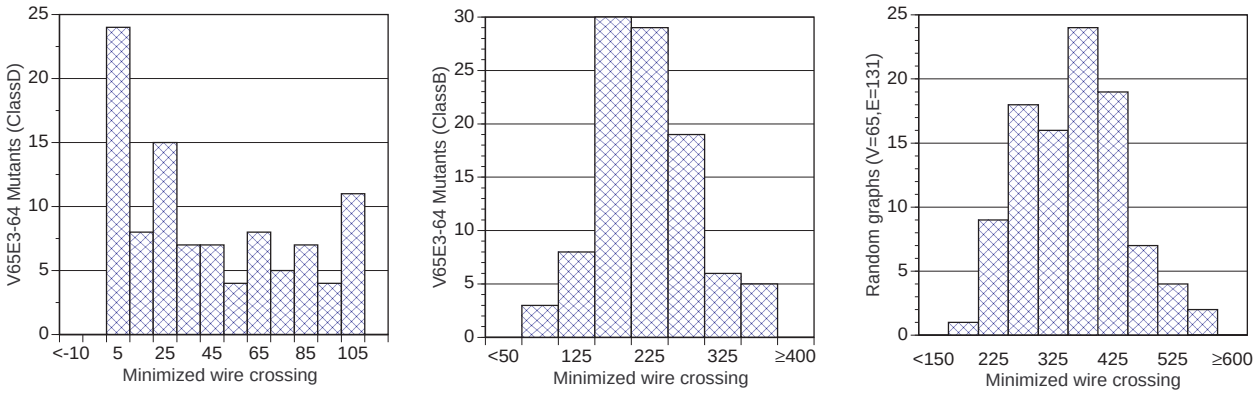
```
=========================================================
(a) algorithms-vs-mutants-vs-classes
=========================================================
                      M-H_1  ....  M-H_k  ....  M-H_b  |
                      ---------------------------------|
                   ................................    |
              M-B_1  ....  M-B_k  ....  M-B_b    |      |
              -----------------------------------|      |
           M-A_1  ....  M-A_k  ....  M-A_b    |  |      |
           -----------------------------------|  |      |
Alg_1-I      A_11-I ....  A_1k-I ....  A_1b-I |  |      |
Alg_1-F      A_11-F ....  A_1k-F ....  A_1b-F |  |      |
.....                                         |  |      |
Alg_j-I      A_j1-I ....  A_jk-I ....  A_jb-I |  |
Alg_j-F      A_j1-I ....  A_jk-F ....  A_jb-F |  |
.....                                         |  |
Alg_a-I      A_a1-I ....  A_ak-I ....  A_ab-I |
Alg_a-F      A_a1-F ....  A_ak-F ....  A_ab-F |

=========================================================
(b) classes-vs-mutants (for a given algorithm Alg_j)
=========================================================
Alg_j-F    M-._1  ....  M-._k  ....  M-._b
          ------------------------------------
M-A          A_j1-F ....  A_jk-F ....  A_jb-F  {M-A_j-F}
M-B          B_j1-F ....  B_jk-F ....  B_jb-F  {M-A_j-F}
....
M-H          H_j1-F ....  H_jk-F ....  H_jb-F  {M-H_j-F}
          ------------------------------------
          {M-._j1-F} ..{M-._jk-F} ..{M-._jb-F}

=========================================================
(c) statistics of algorithms-vs-mutant classes
=========================================================
           M-A        M-B    .....  M-H
          ------------------------------------
Alg_1-I   {M-A_1-I}  {M-B_1-I} ... {M-H_1-I}
Alg_1-F   {M-A_1-F}  {M-B_1-F} ... {M-H_1-F}
.....
Alg_j-I   {M-A_j-I}  {M-B_j-I} ... {M-H_j-I}
Alg_j-F   {M-A_j-F}  {M-B_j-F} ... {M-H_j-F}
.....
Alg_a-I   {M-A_a-I}  {M-B_a-I} ... {M-H_a-I}
Alg_a-F   {M-A_a-F}  {M-B_a-F} ... {M-H_a-F}
```

Fig. 5. Data structures and classes for the proposed experiments.

- Consider, for a given mutant class, (1) sample mean and standard deviation of the *unoptimized* objective function, and (2) sample mean and standard deviation of the objective function *optimized* via algorithm `Alg_j`. We have to decide: (H0) is the difference in the means due to chance, or (H1) is it due to the effect of algorithm `Alg_j`?
- Consider, for a given algorithm `Alg_j`, (1) sample mean and standard deviation of the *optimized* objective function in terms of mutant class A, and (2) sample mean and standard deviation of the *optimized* objective function in terms of mutant class B. We have to decide: (H0) is the difference in the means due to chance, or (H1) is it due to differences of the two mutant classes?
- Consider, for a given mutant class (1) sample mean and standard deviation of the objective function *optimized* by algorithm `Alg_j1`, and (2) sample mean and standard deviation of the objective function *optimized* by algorithm `Alg_j2`. We have to decide: (H0) is the difference in the means due to chance, or (H1) is it due to different performances of the algorithms `Alg_j1` and `Alg_j2`?

For example:

- Upon inspection of initial (DOT_I) and final (DOT_F) wire-crossing results in Figure 6, we do not need a t-test to declare that the wire crossing minimization algorithm implemented in `DOT` [12] is highly effective.



	W_ClassA	W_ClassB	W_ClassC	W_ClassD	W_ClassE	W_ClassF	W_ClassG	W_ClassH	ClassRND
DOT_I(V65E3-64)	[4177, 4270]	[4151, 4245]	[4129, 4230]	[4159, 4254]	[4149, 4242]	[4126, 4224]	[4098, 4197]	[4167, 4252]	[2019, 2107]
	4223, 234	4198, 237	4179, 253	4207, 239	4196, 234	4175, 247	4147, 250	4209, 215	2063, 222
DOT_F(V65E3-64)	[51.3, 63.9]	[205, 231]	[200, 223]	[24.8, 32.5]	[175, 200]	[136, 158]	[42.5, 53.6]	[112, 128]	[343, 377]
	57.6, 31.9	218, 65.6	212, 59.6	28.7, 19.3	188, 63.2	147, 56.2	48.0, 27.9	120, 39.5	360, 84.3
DOT_I(V65E1-66)	[4261, 4360]	[4263, 4367]	[4230, 4335]	[4270, 4372]	[4263, 4359]	[4205, 4319]	[4276, 4377]	[4248, 4348]	[1996, 2087]
	4310, 247	4315, 261	4282, 264	4321, 255	4311, 240	4262, 286	4327, 255	4298, 251	2041, 230
DOT_F(V65E1-66)	[156, 174]	[295, 324]	[287, 315]	[160, 174]	[235, 265]	[209, 238]	[159, 174]	[180, 205]	[379, 420]
	165, 45.7	309, 74.2	301, 72.0	167, 35.4	250, 74.2	224, 72.0	167, 38.8	192.6, 63.6	399, 101

Fig. 6. Statistical summary for 16 mutant classes and 2 random classes of 2-level graphs V65E1-66, V65E3-64.

- Upon inspection of data in Figure 7, we want to determine whether, for the algorithm that optimizes `mincut`, the difference in reported means for `Class_A` and `Class_G` is due to chance or due to differences in mutant classes. For the values shown in the table, we find $t = 10.79$. Hence, we accept the hypothesis that there is a significant difference between the mutant classes A and G for *this* algorithm.
- Upon inspection of data in Table II, we want to determine, for the mutant `Class_A`, whether the difference in reported means for the algorithm `MIS` that optimizes `alg-nodes` and the algorithm `SIS` that optimizes `alg-nodes` is due to chance or due to different performances of the algorithms. For the values shown in the table, we find $t = 5.17$. Hence, we accept the hypothesis that there is a significant difference between the two algorithms for the mutant `Class_A`.

A comprehensive multiple comparison analysis is beyond the scope of this paper and will be presented elsewhere. The subject of multiple comparisons, even when enough data has been collected, requires careful data interpretation and application of tools[2].

Significantly, reader should observe the sensitivity of several algorithms when evaluating the 0-wire perturbation class (`ClassD`). For `ClassD`, the distribution should ideally be a delta-function – however we do observe noticeable spreads of distributions! Given that the netlists in this class are isomorphic, the observations demonstrate the *fallacy of relying on a single measurement of any benchmark circuit* – variations for many of the 'improvements' published to date may well be attributed to chance rather than any intrinsic improvement of the algorithm.

Figure 6 and classes of V65E3-64, V65E1-66. Here, we report results of wire crossings, using `DOT` [12]. Circuits V65E3-64, V65E1-66 belong to two families of parameterized 2-layer directed *sparse* graphs: `Vn+1E3-n` (E3-graphs) and `Vn-1E1-n` (E1-graphs) [11]. In each case, the numbers correspond to the net-node signatures at level 0 and level 1, defined in Figure 2. Properties of E3-graphs are: number of nodes at level 0 = `n+1`, number of 1-input nodes at level 1 = `3`, number of 2-input nodes at level 1 = `n`, number of edges = `2*n + 3`, number of wire crossings = `0`. Properties of E1-graphs are: number of nodes at level 0 = `n-1`, number of 1-input nodes at level 1 = `1`, number of 2-input nodes at level 1 = `n`, number of edges = `2*n + 1`, number of wire crossings = `n - 2`.

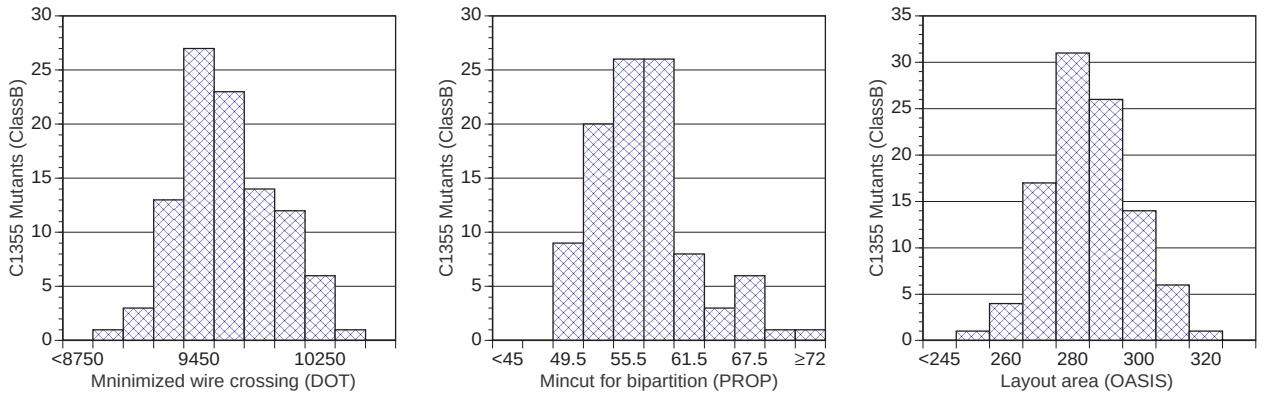
The expected number of wire crossings in 2-layer graph structures with *randomly placed nodes*, such that the likelihood of connecting node node pairs (i, j) or (j, i) is the same, is given by a formula in [14]. For graphs with parameters such as V65E3-64 the expected crossing number is 4258, and for V65E1-66 the expected crossing number is 4324. Remarkably, this number is approached *for all of the eight* randomly placed *mutant subclasses in both circuits* as tabulated in Figure 6. On the other hand, reported means of the wire crossings for the instances of the *randomly placed* random circuit classes (`class_RND`) are 2019 and 2041 respectively [11]. We attribute the significant reduction in the expected wire crossing to clustering of edges at some of the nodes, so the formula in [14] does not apply to the case of randomly generated graphs.

Using `DOT`, the wire crossing is reduced significantly for all mutant classes – less so for the random class. We observe a distinct value of the *minimized mean wire crossing* for each of the eight mutant classes. Note however, that there are *relatively few optimal solutions returned* for mutants in `ClassD` where *known minimum* crossing numbers are 0 and 64! Multiple comparison of the means will be performed later.

Crossing theory has been developed to introduce technique which enhance the readability of hierarchical structures [14]. The problem for placing the nodes for minimum wire crossing is NP-complete [15], even for the 2-layer graphs. The crossing number and wire area have been investigated for effective wire lower bounds and effective edge length for a variety of computational VLSI circuits in [16]. An extensive analysis of wire-crossing minimization algorithms is available in [17].

Figure 7 and classes of C1355. Here, we report result of wire crossings, balanced bi-partition mincut, and placed & routed layout: using `DOT` [12], `PROP` [18], and OASIS [19]. Again, we notice a dramatic reduction in wire crossing after

[2] According to [13], page 175: 'If multiple comparison methods rank second in frequency of *use*, they perhaps rank first in the frequency of *abuse*'.



	W_ClassA	W_ClassB	W_ClassC	W_ClassD	W_ClassE	W_ClassF	W_ClassG	W_ClassH
Wire_Xing-I	[24083, 24261]	[30950, 31209]	[29461, 29693]	[23583, 23706]	[27729, 27944]	[26280, 26470]	[23746, 23884]	[24938, 25111]
	24172, 446	31080, 652	29577, 583	23645, 309	27836, 539	26375, 477	23815, 346	25025, 434
Wire_Xing-F	[4736, 4882]	[9578, 9701]	[8898, 9028]	[4066, 4210]	[7391, 7522]	[6101, 6233]	[4381, 4525]	[5374, 5505]
	4809, 366.5	9640, 307.9	8963, 327.5	4138, 363.0	7457, 329.8	6167, 333.1	4453, 361.7	5440, 327.3
Mincut	[34.7, 36.2]	[55.3, 57.2]	[55.7, 57.6]	[26.4, 28.0]	[58.5, 60.5]	[53.4, 54.8]	[29.1, 30.6]	[45.3, 46.3]
	35.4, 3.6	56.2, 4.8	56.7, 4.8	27.2, 4.0	59.5, 5.0	54.1, 3.5	29.8, 3.7	45.8, 2.6
Lay. area	[2.12M, 2.14M]	[2.83M, 2.88M]	[2.68M, 2.72M]	[2.11M, 2.13M]	[2.42M, 2.46M]	[2.27M, 2.29M]	[2.12M, 2.14M]	[2.18M, 2.20M]
	2.13M, 53.6k	2.85M, 125K	2.70M, 110k	2.12M, 47.9k	2.44M, 88k	2.28M, 65.9k	2.13M, 48.6k	2.19M, 68.3k
Lay. wirelen	[304k, 308k]	[495k, 505k]	[462k, 470k]	[300k, 305k]	[395k, 403k]	[346k, 352k]	[303k, 308k]	[320k, 325k]
	306k, 11k	500k, 25k	466k, 21k	303k, 10k	399k, 19k	349k, 15k	306k, 12k	322k, 14k

Fig. 7. Wire crossing, mincut and layout results for mutant classes of circuit C1355.

TABLE II

FAULT COVERAGE, MAPPING AND LOGIC OPTIMIZATION RESULTS FOR MUTANT CLASSES OF CIRCUIT C1355.

	W_ClassA	W_ClassB	W_ClassC	W_ClassD	W_ClassE	W_ClassF	W_ClassG	W_ClassH
nodes-I	518	518	518	518	518	518	518	518
levels-I	25	25	25	25	25	25	25	25
Fault Coverage	[96.2, 97.2]	[90.5, 91.6]	[92.6, 93.6]	[99.3, 99.3]	[92.7, 93.6]	[93.2, 94.1]	[98.3, 98.7]	[94.6, 95.5]
(%)	96.7, 2.54	91.0, 2.69	93.1, 2.38	99.3, 0.00	93.2, 2.29	93.7, 2.32	98.5, 1.07	95.0, 2.21
Area (mapped-SIS)	[706k, 709k]	[651k, 655k]	[643k, 647k]	[714k, 714k]	[658k, 662k]	[683k, 687k]	[710k, 711k]	[700k, 703k]
	707k, 7.9k	653k, 10.8k	645k, 10.4k	714k, 1.3k	660k, 9.8k	685k, 8.3k	710k, 4.2k	701k, 7.9k
Delay (mapped-SIS)	[31.47, 31.81]	[33.91, 34.53]	[33.30, 33.88]	[30.42, 30.56]	[32.99, 33.47]	[32.31, 32.79]	[30.93, 31.19]	[31.93, 32.33]
	31.64, 0.87	34.22, 1.57	33.59, 1.47	30.49, 0.35	33.23, 1.20	32.55, 1.20	31.06, 0.67	32.13, 1.03
alg-nodes (MIS)	[233, 239]	[581, 595]	[568, 579]	[184, 184]	[496, 506]	[394, 403]	[201, 204]	[394, 403]
	236, 14	588, 35.7	573, 27.7	184, 0.0	501, 23.7	398, 20.8	203, 8.49	398, 20.8
alg-level (MIS)	[21.5, 22.4]	[29.7, 30.5]	[29.5, 30.2]	[17.1, 17.3]	[28.3, 29.1]	[25.9, 26.8]	[19.5, 20.2]	[25.9, 26.8]
	21.9, 2.07	30.1, 2.03	29.9, 1.92	17.2, 0.66	28.7, 1.97	26.3, 2.22	19.8, 1.62	26.3, 2.22
alg-nodes (SIS)	[244, 249]	[557, 569]	[542, 552]	[205, 210]	[485, 493]	[393, 400]	[219, 223]	[310, 316]
	246, 13.2	563, 30.7	547, 25.5	208, 11.6	489, 21.9	396, 18.6	221, 11.1	313, 15.4
alg-level (SIS)	[20.6, 21.3]	[28.2, 28.7]	[27.9, 28.4]	[16.9, 17.2]	[27.4, 27.9]	[25.4, 26.0]	[18.9, 19.5]	[23.1, 23.6]
	21.0, 1.64	28.5, 1.26	28.2, 1.37	17.0, 0.8	27.7, 1.42	25.7, 1.35	19.2, 1.46	23.3, 1.41

TABLE III

ROBDD-SIZE SUMMARY FOR 8 MUTANT CLASSES OF REFERENCE CIRCUIT C499.

	W_ClassA	W_ClassB	W_ClassC	W_ClassD	W_ClassE	W_ClassF	W_ClassG	W_ClassH
Initial order	[58064, 66164]	[175211, 302499]	[123919, 178438]	[25894, 25894]	[123730, 170007]	[110372, 156884]	[40234, 45874]	[85695, 101887]
	62114, 20251	238855, 318217	151178, 136297	25894, 0	146869, 115693	133628, 116279	43054, 14100	93791, 40479
CAL-good_sift	[32000, 35911]	[10989, 13660]	[13431, 16915]	[26118, 26118]	[19539, 23156]	[28347, 32790]	[30550, 32641]	[33770, 38035]
	33955, 9777	12325, 6675	15173, 8709	26118, 0	21348, 9041	30569, 11109	31595, 5225	35903, 10660
CU-good_sift	[36106, 40107]	[10590, 13075]	[12357, 15977]	[26294, 26294]	[19019, 22436]	[28331, 33211]	[33213, 36310]	[34999, 39488]
	38106, 10004	11832, 6213	14167, 9051	26294, 0	20727, 8541	30771, 12200	34762, 7745	37244, 11224
CMU-good_sift	[34922, 38825]	[11889, 14535]	[14604, 17799]	[26054, 26054]	[20575, 24641]	[30465, 35088]	[31905, 34299]	[35347, 39441]
	36873, 9756	13212, 6616	16202, 7988	26054, 0	22608, 10165	32776, 11559	33102, 5984	37394, 10234
CAL-free_sift	[33117, 37144]	[11217, 13858]	[13048, 16470]	[26958, 26958]	[18289, 22297]	[27512, 32224]	[32227, 34775]	[33960, 38165]
	35130, 10067	12537, 6603	14759, 8557	26958, 0	20293, 10021	29868, 11778	33501, 6370	36062, 10514
CU-free_sift	[29454, 44141]	[10535, 13327]	[13047, 16018]	[38346, 38346]	[18898, 22321]	[27356, 31766]	[39726, 42447]	[34287, 38682]
	36798, 29291	11931, 6981	14532, 7428	38346, 0	20609, 8559	29561, 11024	41086, 6803	36484, 10987
CMU-free_sift	[36978, 41388]	[12670, 15683]	[13799, 17976]	[31494, 31494]	[19595, 23077]	[29429, 34553]	[35379, 38060]	[36724, 41492]
	39183, 11025	14177, 7532	15888, 10443	31494, 0	21336, 8706	31991, 12810	36719, 6704	39108, 11920

optimization. Since we have no access to report initial values of mincut and layout, only optimized values are reported. Here, we can analyze the effectiveness of the mutant classes to distinguish the performance of the algorithms.

Table II and classes of C1355. Here, we report results of fault coverage, technology mapping, and logic optimization using the algorithms in MIS [20] and SIS [21]. Note that all mutant classes have the *same number of 2-input nodes and levels!* Consistent with experience with other WSI mutants, fault coverage remains high for all mutant classes, indicating that wiring mutations have introduced relatively few redundancies. Technology mapping is reported for the `lib2.genlib` [10] only, since the differences with a smaller library were negligible. As discussed earlier, logic optimization algorithms in MIS and SIS exhibit significantly different performance for `Class_A`, favoring MIS. Note however that this behavior is not consistent for all classes. In fact, both algorithms can return an 'optimized circuit' that has more 2-input nodes than the initial mutant circuit! A comprehensive multiple comparison of the means, for mutant classes of several reference circuits, will be performed later.

Table III and classes of C499. Here, we report results of BDD-node sizes for different variable ordering algorithms. The VIS 1.1 environment was used throughout the experiments [22]. Result for `initial` refer to the 'best' known static order ('best' for nominal circuit C499 only, courtesy of Fabio Somenzi). Three algorithms `CAL` [23], `CU` [24], `CMU` [25], were executed on all mutants in two modes: `good_sift` and `free_sift`. Here, `good_sift` starts with a known good order [24], followed by the sifting method of dynamic ordering [26]. In contrast, `free_sift` computes its own static order with no hints, then uses sifting dynamic reordering.

Data in this table needs to be analyzed globally due to apparent overlap of confidence intervals of reported sample means. Locally, looking at means alone for `W_Class_A` and `free_sift` cases, one could conclude that `CAL`<`CU`<`CMU`. However, t-test of difference reveals that the differences between `CAL` and `CU`, and `CU` and `CMU`, are due to chance only, while there is still significant difference between `CAL` and `CMU`. Overall, data shows that variable ordering algorithms for BDDs are intrinsically unstable. We could not build BDDs for all mutants of the C1355 class, yet none of the mutants in this class present any problems for redundancy identification algorithms in SIS!!

One may argue that WSI mutants are not the most appropriate test cases for BDD algorithms. Alternatives to be considered include the *entropy-signature invariant (ESI)* classes as reported in [27].

VII. Conclusions

We have demonstrated the first-generation capability to synthesize a large number of equivalence classes and a large number of circuit samples in each class. The availability of such circuits provides the opportunity to introduce *design of experiments techniques* and *sampling methods*, to benchmark the performance of a number of EDA algorithms.

Collaborative web-based experiments, initiated under http://www.cbl.ncsu.edu/experiments/, with mutants based on a range of reference circuits, may well change the paradigm of benchmarking EDA algorithms as we know it today.

Acknowledgments. We appreciate the access to tools used in this research: SIS and VIS from the team at UC Berkeley, PROP from Roman Kužnar from U. of Ljubljana (Slovenia), and DOT from AT&T Research. Fabio Somenzi deserves special mention for his collection of BDD orders.

References

[1] 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.

[2] M. Hutton, J.P. Grossman, J. Rose and D. Corneil. Characterization and Parametrized Random Generation of Digital Circuits. In *Design Automation Conference*, June 1996.

[3] K. Iwama and K. Hino. Random Generation of Instances for Logic Optimizers. In *Design Automation Conference*, pages 430–434, June 1994.

[4] J. P. Roth and R. M. Karp. Minimization over Boolean Graphs. *IBM Jl. of Res. and Dev.*, pages 227–238, April 1962.

[5] R. A. Fisher. *Statistical Methods, Experimental Design, and Scientific Inference*. Oxford University Press, 1993. Reprinted, with corrections, from earlier versions, 1925-1973.

[6] F. Brglez. A Tutorial on Design of Experiments to Benchmark Algorithms in EDA. Technical report, 1997. Also available at `http://www.cbl.ncsu.edu/publications/`.

[7] M. R. Hartoog. Analysis of Placement Procedures for VLSI Standard Cell Layout. In *Design Automation Conference*, pages 314–319, July 1986.

[8] K. A. Brownlee. *Statistical Theory and Methodology In Science and Engineering*. Krieger Publishing, 1984. Reprinted, with revisons, from second edition, 1965.

[9] F. Brglez and H. Fujiwara. Special Session on ATPG (Also introducing 'A Neutral Netlist of 10 Combinational Benchmark Circuits'). In *Int. Symp. On Circuits and Systems*, 1985. A basis for a benchmark directory ISCAS85, now archived at `http://www.cbl.ncsu.edu/benchmarks/`.

[10] S. Yang. Logic Synthesis and Optimization Benchmarks User Guide. Technical report, MCNC, RTP, NC, Jan. 1991. Report and benchmarks archived at `http://www.cbl.ncsu.edu/benchmarks/`.

[11] F. Brglez (Ed.). Collaborative Archives of On-Going Experiments with Algorithms in EDA: Taxonomy of Data Samples and Analysis of Multiple Comparisons. Technical report, 1997. Also available at `http://www.cbl.ncsu.edu/publications/`.

[12] E.R. Gasner *et el.* A Technique for Drawing Directed Graphs. *IEEE Trans. Software Engg.*, 19:214–230, 1993. Software available at `http://www.research.att.com/sw/tools/graphviz/`.

[13] Jason C. Hsu. *Multiple Comparisons: Theory and Methods*. Chapman & Hall, London, 1996.

[14] J. N. Warfield. Crossing Theory and Hierarchy Mapping. *IEEE Trans. Sys., Man, and Cybernetics*, pages 505–523, July 1977.

[15] M. R. Garey and D. S. Johnson. Crossing Number is NP-complete. *SIAM J. Algebraic Discrete Methods*, 4:312–316, 1983.

[16] F. T. Leighton. New Lower Bound Techniques for VLSI. *Math. Systems Theory*, 17:47–70, 1984.

[17] M. Jünger and P. Mutzel. 2–Layer Straightline Crossing Minimization: Performance of Exact and Heuristic Algorithms. *Journal of Graph Algorithms and Applications (JGAA)*, 1(1):1–25, 1997. Available at `http://www.mpi-sb.mpg.de/~mutzel/mpireports/`.

[18] R. Kužnar and F. Brglez. PROP: A Recursive Paradigm for Area-Efficient and Performance Oriented Partitioning of Large FPGA Netlists . In *IEEE Intl. Conf. Computer-Aided Design*, November 1995.

[19] K. Kozminski, (Ed.). OASIS2.0 User's Guide. MCNC, Research Triangle Park, N.C. 27709, 1992.

[20] R. Brayton *et al.* MIS: A Multiple-Level Logic Optimization System. *IEEE Trans. Computer-Aided Design*, pages 1062–1081, November 1987.

[21] SIS – Release 1.1. UC Berkeley Software Dist., Sept. 1992.

[22] The VIS Group. VIS: A system for verification and synthesis. In R. Alur and T. Henzinger, editors, *Proc. 8th Intl. Conf. on Computer Aided Verification*, number 1102 in Lecture Notes in Comp. Sc., pages 428–432, New Brunswick, NJ, July 1996. Springer.

[23] P. Ashar and M. Cheong. Efficient breadth–first manipulation of binary decision diagrams. In *Proceedings of the International Conference on Computer Aided Design*, pages 622–627, 1994.

[24] F. Somenzi *et al.* Colorado University Decision Diagram package (CUDD), release 2.1.2, 1997. Available from `ftp://vlsi.colorado.edu/pub/cudd-2.1.2.tar.gz`.

[25] D. E. Long. CMU BDD package, 1993. Available from `http://emc.cmu.edu/pub/bdd/bddlib.tar.Z`.

[26] R. Rudell. Dynamic variable ordering for ordered binary decision diagrams. In *ICCAD'93*, pages 42–47, 1993.

[27] J. E. Harlow III and F. Brglez. Synthesis of ESI Equivalence Class Combinational Circuit Mutants. Technical report, 1997. Also available at `http://www.cbl.ncsu.edu/publications/`.