The BDD size of a typical Boolean circuit is highly sensitive to the order of variables in the BDD. Finding an optimal order which yields the smallest BDD for a given circuit is an NP-hard problem. Exact algorithms that find an optimum order are only practical for small instances, solutions for realistic problems must rely on heuristic methods. Such methods can be roughly divided into two groups: (1) static ordering and (2) dynamic ordering. The primary goal of static ordering is to find a “good ordering” before constructing the BDD. The dynamic ordering takes place during BDD construction itself, with the objective to reorder the variables from the given initial order in the hope of finding a smaller BDD.

In typical problems, one first needs to build a very large graph before finding a variable order that yields a BDD graph of an acceptable size. It may also happen that no BDD graph is constructed, due to limited memory and computational resources. Therefore, there is a strong motivation to find a good initial order, before building the BDD itself. In this proposal, we argue for such investigation despite some negative finding about the static variable ordering of a few years ago:
-- static variable ordering in VIS1.1 is no better than random variable ordering ([1], Figure 2)
-- static variable ordering in VIS1.3 may be worse than the static variable ordering in VIS1.1 ([1], Table 5)

Most of the static ordering algorithms are based on a variant of the topological structure of the circuit.

However, when examined with a rigorous experimental procedure [1], such orderings turned out no better than a random ordering. In this proposal, we outline a BDD variable ordering procedure (BBVO) that relies on the ‘black box’ sampling analysis of the Boolean function only, as defined by the underlying Boolean circuit, and not on its structural implementation. The main elements of BBVO include:

(1) simulation-based characterization of the Boolean circuit ‘black box’ functional behavior, which captures the effects of perturbations applied to circuit primary inputs at circuit primary outputs. The perturbation model itself is based on the familiar single and multiple ‘stuck-at’ fault models. Efficient tools for such simulation already exist for the single ‘stuck-at’ fault model.

(2) an analysis of the proposed simulation-based characterization in order to generate a “good initial variable ordering”. Such an ordering is expected to satisfy the two postulates below:
(a) the order reduces the average NodeSize, the size of BDD, when built by one or more methods available in BDD-constructor;
(b) the RunTime cost of finding this order does not exceed a given multiple of the RunTime cost of building the BDD itself.

A few small-scale experiments demonstrate the promise of the approach. However, a comprehensive study and sound experimental strategy is needed, on instance isomorphs of increasing size and diversity, to reliably demonstrate the merits of the BBVO procedure as outlined in this proposal.

About the BDD variable ordering problem

Outline of “black-box variable ordering” (BBVO) proposal

Small-example comparisons (VIS vs BBVO)

What (could be) next?

About the Problem

The BDD size of a typical Boolean circuit is highly sensitive to the order of variables in the BDD. Finding an optimal order which yields the smallest BDD for a given circuit is an NP-hard problem.

Exact algorithms that find an optimum order are only practical for small instances, solutions for realistic problems must rely on heuristic methods: (1) static ordering and (2) dynamic ordering.

The primary goal of static ordering is to find a "good ordering" before constructing the BDD. The dynamic ordering takes place during BDD construction itself, with the objective to reorder the variables from the given initial order in the hope of finding a smaller BDD.
Importance of initial ordering

• In typical problems, one first needs to build a very large graph before finding a variable order that yields a BDD graph of an acceptable size.

• It may also happen that no BDD graph is constructed, due to limited memory and computational resources.

• Finding a good initial order, before building the BDD itself will mitigate problems such listed above.

Some negative finding about the static variable ordering:
-- static variable ordering in VIS1.1 is no better than random variable ordering ([1], Figure 2)
-- static variable ordering in VIS1.3 may be worse than the static variable ordering in VIS1.1 ([1], Table 5)

Example with c432 (1--7 outputs)

For output 6:
Initial NodeSize may reach 10,000,000 nodes (Treatment 6)
before settling at 2000 nodes (Treatment 8)

Hence, “good initial order” is desirable ....
Alternative to Static Variable Ordering

• Most of the static ordering algorithms are based on a variant of the topological structure of the circuit. As shown, such orderings may not be any better than a random ordering.

• As an alternative, we outline a BDD variable ordering procedure (BBVO) that relies on the ‘black box' sampling analysis of the Boolean function only, as defined by the underlying Boolean circuit, and not on its structural implementation.

The main elements of BBVO are discussed next.

BBVO Concepts and Flow

• use simulation to characterize the functional behavior of the Boolean circuit 'black box', i.e. propagate the effects of perturbations at circuit primary inputs to circuit primary outputs.

• use perturbations based on the familiar single and pairwise 'stuck-at' fault models.

• analyze simulator responses to generate a "good initial variable ordering".

... from ICCAD'98
A “Good Initial Ordering”

Such an ordering should satisfy the two postulates:

(1) the order reduces the average NodeSize, the size of BDD, when built by one or more methods available in BDD solver;

(2) the RunTime cost of finding this order does not exceed a given multiple of the RunTime cost of building the BDD itself.

Boolean Observability matrix

Given the definition of observabilities ...

... we create an algebraic observability matrix whose syndromes are computed from Boolean observability functions shown here ... 

In practice, we approximate this matrix using the simulator...
Order-inducing algebraic O-matrix

Ranking the relative observability indexes induces a variable permutation ([\text{beg}] \Rightarrow [\text{end}]).

\[
\begin{bmatrix}
\text{beg} & S3 & S2 & A3 & A2 & A1 & A0 & B3 & B2 & B1 & B0 & \text{end}
\end{bmatrix}
\]

\[
\begin{array}{cccccccccccc}
A3 & 0.211 & 0.211 & 0.137 & 0.133 & 0.133 & 0.133 & 0.211 & 0.070 & 0.070 & 0.070 & S3 \\
A2 & 0.211 & 0.211 & 0.133 & 0.137 & 0.133 & 0.133 & 0.070 & 0.211 & 0.070 & 0.070 & S2 \\
A1 & 0.211 & 0.211 & 0.133 & 0.133 & 0.133 & 0.137 & 0.070 & 0.070 & 0.211 & 0.070 & A3 \\
A0 & 0.211 & 0.211 & 0.133 & 0.133 & 0.133 & 0.137 & 0.070 & 0.070 & 0.070 & 0.211 & A2 \\
B3 & 0.211 & 0.211 & 0.211 & 0.070 & 0.070 & 0.070 & 0.105 & 0.070 & 0.070 & 0.070 & A1 \\
B2 & 0.211 & 0.211 & 0.070 & 0.211 & 0.070 & 0.070 & 0.105 & 0.070 & 0.070 & 0.070 & A0 \\
B1 & 0.211 & 0.211 & 0.070 & 0.070 & 0.211 & 0.070 & 0.105 & 0.070 & 0.070 & 0.070 & B3 \\
B1 & 0.211 & 0.211 & 0.070 & 0.070 & 0.211 & 0.070 & 0.105 & 0.070 & 0.070 & 0.070 & B2 \\
S3 & 0.469 & 0.429 & 0.211 & 0.211 & 0.211 & 0.211 & 0.211 & 0.211 & 0.211 & 0.211 & B1 \\
S2 & 0.429 & 0.469 & 0.211 & 0.211 & 0.211 & 0.211 & 0.211 & 0.211 & 0.211 & 0.211 & B0 \\
\hline
0.258 & 0.258 & 0.138 & 0.138 & 0.138 & 0.138 & 0.116 & 0.116 & 0.116 & 0.116 &
\end{array}
\]

Extracted order: S3, S2, A3, A2, A1, A0, B3, B2, B1, B0

.... not a good order, 61 MDD nodes -- more work is needed!!

Order-inducing cor-coeff. O-matrix

Ranking the observability correlation coefficients induces a variable permutation ([\text{beg}] \Rightarrow [\text{end}]).

\[
\begin{bmatrix}
\text{beg} & S3 & S2 & A3 & A2 & A1 & A0 & B3 & B2 & B1 & B0 & \text{end}
\end{bmatrix}
\]

\[
\begin{array}{cccccccccccc}
S3 & 1 & 0.983 & 0.666 & 0.666 & 0.666 & 0.666 & 0.750 & 0.750 & 0.750 & 0.750 & S3 \\
S2 & 0.98 & 1 & 0.666 & 0.666 & 0.666 & 0.666 & 0.750 & 0.750 & 0.750 & 0.750 & S2 \\
A3 & 0.666 & 0.666 & 1 & 0.336 & 0.336 & 0.336 & 0.67 & 0.508 & 0.508 & 0.508 & B3 \\
A2 & 0.666 & 0.666 & 0.336 & 1 & 0.336 & 0.336 & 0.67 & 0.508 & 0.508 & 0.508 & A3 \\
A1 & 0.666 & 0.666 & 0.336 & 0.336 & 1 & 0.336 & 0.508 & 0.508 & 0.67 & 0.508 & B2 \\
A0 & 0.666 & 0.666 & 0.336 & 0.336 & 0.336 & 1 & 0.508 & 0.508 & 0.508 & 0.67 & A2 \\
B3 & 0.750 & 0.75 & 0.508 & 0.508 & 0.508 & 0.508 & 0.67 & 1 & 0.170 & 0.170 & B1 \\
B2 & 0.750 & 0.75 & 0.51 & 0.508 & 0.508 & 0.508 & 0.170 & 1 & 0.470 & 0.470 & A1 \\
B1 & 0.750 & 0.75 & 0.508 & 0.51 & 0.508 & 0.508 & 0.470 & 0.470 & 1 & 0.470 & B0 \\
B0 & 0.750 & 0.75 & 0.508 & 0.508 & 0.51 & 0.666 & 0.470 & 0.470 & 0.470 & 1 & A0 \\
\hline
\end{array}
\]

Extracted order: S3, S2, B3, A3, B2, A2, B1, A1, B0, A0,

.... much better, 31 MDD nodes, which is best result with VIS
O-matrix-based ordering

21 instances of Treatment 10: 21 random pattern perturbation simulations of 256 patterns, applied to a single circuit from class warp15.

21 instances of Treatment 10: 21 random pattern perturbation simulations of 256 patterns, applied to a single circuit from class ex_carry15.

... less node variability than best orders from VIS!!

BDD solver configurations (SolverIDs)

<table>
<thead>
<tr>
<th>Treatment</th>
<th>Initial Ordering</th>
<th>Dynamic Ordering</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Natural</td>
<td>None</td>
</tr>
<tr>
<td>1</td>
<td>VIS</td>
<td>None</td>
</tr>
<tr>
<td>2</td>
<td>VIS</td>
<td>Sift during construction</td>
</tr>
<tr>
<td>3</td>
<td>VIS</td>
<td>Sift during construction, Sift once after construction</td>
</tr>
<tr>
<td>4</td>
<td>VIS</td>
<td>Sift during construction, Sift twice after construction</td>
</tr>
<tr>
<td>5</td>
<td>VIS</td>
<td>Sift during construction, Sift-converge after construction</td>
</tr>
<tr>
<td>6</td>
<td>Natural</td>
<td>Sift during construction</td>
</tr>
<tr>
<td>7</td>
<td>Natural</td>
<td>Sift during construction, Sift once after construction</td>
</tr>
<tr>
<td>8</td>
<td>Natural</td>
<td>Sift during construction, Sift twice after construction</td>
</tr>
<tr>
<td>9</td>
<td>Natural</td>
<td>Sift during construction, Sift-converge after construction</td>
</tr>
<tr>
<td>10</td>
<td>BBVO</td>
<td>None</td>
</tr>
</tbody>
</table>
Ordering comparisons: VIS vs BBVO

What (could be) next …?

• pencil & paper (small) examples to verify, illustrate, and formalize the BBVO procedure in a tech report
• identify a fast combinational fault simulator engine and incorporate an efficient method to process single and pairwise perturbations, injected on n-inputs of the Boolean circuit and observed on its m-outputs.
• organize simulation results into efficient data structures (n * n matrices for each output)
• implement a matrix traversal that returns a “good initial order”
• design a series of experiments, on instance isomorphs of increasing size and diversity, to reliably demonstrate the merits of the BBVO procedure.