BIST grouping optimization for large-scale embedded memory design

Ben Kao Karl Chang, Mathew Tsai - August 30, 2013

Design-for-test (DFT) engineers often struggle to develop a memory built-in self-test (BIST) grouping plan, deciding which memories belong to which BIST group, to improve test time, routing effort, and minimize area overhead. Memory BIST grouping is so complicated that the grouping process requires a significant amount of design time and can prove unreliable if the amount of memory is large. In order to facilitate the process, an automation tool and convenient interface is required to speed the process and guarantee grouping quality.

It currently takes DFT engineers several hours to group memories for BIST on a design that features hundreds of memories. Figure 1 shows that, as the number of memories increases, the time spent on BIST grouping increases exponentially. For a design with thousands of memories, when all tasks are taken into account--grouping constraint, test time, routing effort, and area overhead—it is exceedingly difficult to achieve reliable memory BIST grouping results in a compressed amount of time.

Figure 1 As the number of memories increases, the time spent on BIST grouping increases exponentially.

Figure 2 shows the flow for BIST group adjustment that reduces the amount of routing effort. BIST grouping depends upon the floor plan of the net list, taking into consideration memory locations. The flow is executed iteratively each time the net list or floor plan is updated. When the number of memories or the floor plan is changed, the BIST memory group must also be adjusted to shorten test time and minimize routing resources.
Without the tool’s help, designers will be iteratively defining BIST groupings significantly extending design time.

**_constraints on BIST memory grouping**

There are always both "hard" and "soft" constraints that come into play when grouping memories for BIST.

Hard constraints preclude organizing memories of different constraint types into the same BIST group. Hard constraints include: power domains, frequency domains, synchronous/asynchronous memory types, and user-specified constraint. Memories in different power and frequency domains cannot be grouped in the same BIST, and synchronous memories cannot be grouped with asynchronous memories in the same BIST either.

Soft constraints limit the final grouping result to a specified target but do not necessarily restrict BIST members when taking into consideration parameters such as area overhead, test time, power, and routing efforts. In some cases, DFT engineers attempt to reduce area overhead by restricting the number of BIST to a maximum value. Alternatively, restriction on test time prevents long test run times. To prevent an IR drop problem, power is a consideration during memory BIST testing. Critically, routing must be reduced.

All constraints should be considered when assembling BIST grouping. But for soft constraint in particular, there is the inevitable trade-off between test time, power, and area overhead. Assuming that all BISTs are tested in serial and memories are tested in parallel, to reduce area overhead on the BIST circuit the number of memories in a BIST must be increased. That, in turn, leads to an increase in power consumption. If BISTs are tested in parallel and memories are tested in serial, the decreased BIST number will result in increased test time. Conversely, reducing test time will impact area overhead or power.
Smart memory BIST grouping

Smart memory BIST grouping allows DFT engineers to use an algorithm to automatically group memories or to manually group memories through the GUI interface. The best results have been achieved so far by using automatic grouping and then adjusting BIST groups based on a customized design spec for what-if analysis (Figure 3).

To run memory BIST grouping on SMBG, prepare the memory floor plan (DEF), memory information (LEF), power domain information, and the request form. The request form is a file to record the BIST grouping result, ungrouped memories, and other information. The project environment can be recorded into a SMBG script and read from SMBG (Figure 4). Engineers can also specify their own constraints for the project in the SMBG script.
SMBG integrates an automatic grouping algorithm and GUI interfaces to allow manual tuning. An automatic grouping algorithm provides an initial grouping result based on the floor plan. Engineers can then adjust the grouping result for what-if analysis. **Automatic grouping**

The automatic grouping algorithm provides the initial BIST grouping result. Users can adjust the BIST grouping based on their own specific requirements. The automatic grouping algorithm distributes memories with different hard constraints into different groups and clusters memories with low routing efforts into the same group.

Engineers determine soft constraints for customized design spec, except for routing. SMBG guarantees routing resource during BIST grouping based on the floor plan to make sure the design is routable. Routing effort is the prime consideration when designing both the grouping and optimization algorithms.

The algorithm is implemented in three stages: classification, clustering, and refinement (**Figure 5**).

In the classification stage, memories with different hard constraint type are separated into different classes. Grouping memories through classification prevents results that would violate hard constraints that could easily go unnoticed during manual operations.

In the clustering stage, memories are organized within each class to form BIST groups. SMBG collects memories into groups based on the floor plan. The distance between memories is an essential factor. When completed, the number of BISTs can be adjusted to accommodate area overhead and test time concerns.

Group adjustment for soft constraint is performed in the refinement stage. Area overhead has been determined by the number of BIST groups in the clustering stage, so the major task in this stage is to adjust test time. A test time optimization algorithm has been developed that takes routing into consideration to perform test time adjustment between groups. As a result, test time can improve approximately 30.04% on average after this stage (**Figure 6**).
Currently, there is relatively little research on BIST grouping. Some discuss restrictions on the BIST structure that cannot be applied to the current memory BIST implementation tool. Tzuo-Fan Chien provides an ILP-based solution to identify each BIST one-by-one.

SMBG finds all BIST groups on a single pass. Masahide Miyazaki provides an area-concern algorithm with a limitation on word depth and bit width on memories in a group, while SMBG uses a routing-concern algorithm. Routing effort is the major concern since the design with low routing effort also has lower area overhead on the routing wire.

There are also some third-party EDA tools with memory BIST grouping function. When comparing routing efforts between a third-party EDA tool and SMBG, the SMBG routing effort is lower than the third-party EDA tool, as shown in the groupings in the circled region of Figure 7. The routing resource saves, on average, nearly 50% compared to the third-party EDA tool (Figure 8).

**Figure 6** Test time can improve using the SMBG test time optimization algorithm.

<table>
<thead>
<tr>
<th>Case</th>
<th>Original Test Time (ns)</th>
<th>Optimized Test Time (ns)</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>73646100</td>
<td>45711400</td>
<td>37.93%</td>
</tr>
<tr>
<td>Case 2</td>
<td>35712000</td>
<td>17856000</td>
<td>50%</td>
</tr>
<tr>
<td>Case 3</td>
<td>186140000</td>
<td>169830000</td>
<td>8.76%</td>
</tr>
<tr>
<td>Case 4</td>
<td>55869400</td>
<td>40632300</td>
<td>27.27%</td>
</tr>
</tbody>
</table>

**Figure 7** The SMBG routing effort is lower than the third-party EDA tool in this grouping result comparison.
Figure 8 The routing resource saves nearly 50% on average compared to the third-party EDA tool.

IR drop is also a concern. By grouping memories by location with memories in a serial protocol, the SMBG BIST grouping demonstrated low IR drop risk (Figure 9).

GUI interface for ‘what-if’ analysis

The SMBG integrated interface shows floor plan, test time, and constraint information (Figure 10). In the floor plan window, memories appear based on the floor plan and are color-coded to represent the BIST group to which they belong. When engineers select a memory/BIST, an information box appears showing such information as power domain and frequency domain. The grouped result status is also listed in the memory/BIST list that contains all memories and BISTs with a corresponding colored square.
Figure 10 The SMBG GUI interface shows floor plan, test time, and constraint information.

Engineers can use the GUI to adjust BIST groups by performing what-if analysis using the memory/BIST information box, test time analysis window, and constraint class window.

The test time for current BIST groups can be observed in the test time analysis window (Figure 11). Based on the analysis result, engineers can adjust test time by referencing memory required test time in the information box and memory locations. However, most of test time adjustment is done by the automatic grouping algorithm. Manual adjustments are almost never needed.

Figure 11 The test time for current BIST groups can be observed in the test time analysis window.

A constraint class window allows engineers to observe memory classes to prevent memories from being grouped with others in different classes when adjusting for what-if analysis (Figure 12). It provides the flexibility to choose which constraints are to be considered. Engineers can even select a specified constraint type to see memories with the constraint in this window.
Figure 12 A constraint class window allows engineers to observe memory classes to prevent memories from being grouped with others in different classes when adjusting for what-if analysis.

SMBG has significantly reduced execution time. The execution time differential increases significantly for designs with substantial memories (Figure 13).

Figure 13 The execution time differential increases significantly for designs with substantial memories.

Summary

The SMBG tool provides a convenient interface and automatic grouping algorithm to help engineers group memory BIST. The tool generates BIST groups with low routing efforts and adjusts the groups for what-if analysis.

The quality of the BIST design is improved with test time optimization providing about 30% test time improvement on average. The routing efforts of the BIST grouping result are quite low. Routing resources have been improved nearly 50% compared to the results from a third-party EDA tool. IR drop analysis shows the grouping to have a low IR-drop risk.

SMBG provides a what-if analysis environment that saves time spent on BIST grouping, optimization,
and constraint checking.

Execution time is also significantly reduced. For designs with thousands of memories, operation time improvement can be up to 80%.

References


Also see:

- Handling X-bounding in LBIST designs
- Design planning for large SoC implementation at 40nm: Guaranteeing predictable schedule and first-pass silicon success
- Identification and reduction of safe-stating points in LBIST designs
- The practicing instrumentation engineer's guide to the DFT