VLSI / SOC Testing

Lecture 12

1. Combinational Test Set Compaction
   - Motivations: (1) test application time, (2) test data volume
   - Static compaction: compaction performed after ATPG is finished
   - Dynamic compaction: compaction performed along with ATPG
2. Dynamic Compaction:
   - When PODEM derives a vector v, v may still have many unspecified bits
   - fill the X's more intelligently to maximize detection of remaining faults $\longmapsto$ can use GAs here as well
3. Static Compaction: if test set not fully specified
   - can combine 2 or more vectors if they are *compatible*:
4. Static Compaction Overview
   - Idea: a fault may be detectable by a number of vectors in the test set, so we want to choose a smallest set of vectors from the original test set such that every originally detected fault is detected
   - Need: faultsim without fault-dropping $\longmapsto$ so a fault may be detected multiple times by different vectors
   - Construct: a dictionary mapping faults to vectors

5. Compaction Procedure

   - Identify essential vectors, which are the vectors by which some faults are detected exclusively
   - Since essential vectors must be included in the compacted test set, remove faults that are detected by these essential vectors first
   - For the remaining faults and vectors, find the best *cover* (subset) of vectors such that all faults in the detection dictionary are detected
   - GreedyCovering Procedure

   ```
   while compaction not finished
       sort vectors according to # remaining faults each detects
       pick $v$ that detects the most of remaining faults
   ```

6. Dictionary can potentially be very large

   - To reduce dictionary size, one thing we could do is quickly identify the essential vectors and remove all faults they detect, then build the dictionary for the rest of the faults
   - To quickly identify essential vectors, simply perform faultsim with 2-det (drop fault after it's detected 2 times)
     $\longmapsto$ any fault that is detected only once is an *essential fault* and corresponding vector is an essential vector
   - If the remaining faults still large, do not build full dictionary, instead, build dictionary for N-detects (a fault can have at most N detection vectors)
     $\longmapsto$ This will result in a slightly suboptimal compaction

7. Reverse Simulation: A simple and cheap static compaction

   - No need to build dictionary
   - Original test set $\{v_1, v_2, ..., v_n\}$
   - a vector v_k is in the test set because it detects at least one fault missed by $v_1 \ldots v_{k-1}$
     $\longmapsto$ In other words, v_k detects some hard faults
   - by simulating the vectors in reverse order, perhaps the faults detected by some vectors in the beginning are detected by later vectors!

Example 1:

8. Test Vector Order Problem
   - Order the compacted sets such that vectors that detect most faults are placed early in the test set
   - If the entire test set will not be used, the first 80% vectors can still detect majority of faults
9. Sequential ATPG
   - A test sequence (of vectors) needed: problem much more complex than combinational ATPG
   - Vector order within sequence important, they determine the specific order of states visited
   - Iterative Logic Array (ILA) model used:

10. Deterministic ATPG

    - Excitation and propagation may each require several time-frames
    - Step 1: time-frame 0 excitation
    - Step 2: propagate fault effect till a PO, possibly in a later time-frame $\rightarrow$ more values needed at the FFs of time-frame 0, demanded by propagation
    - Step 3: justify the state at time frame 0

11. Problems and Issues in Sequential ATPG

    - State at time-frame 0 may be illegal/unreachable
    - If state is unjustifiable, need to backtrack and possibly find a new multi-frame propagation solution!
    - Must justify state from an all unknown state - sequence could be long

12. Ways to reduce ATPG costs

    - Minimize # state variables needed for state justification
    - Minimize # time-frames needed to justify
    - Detect illegal/unreachable states early to avoid future backtracks

    Example 2:

    Example 3:

13. Because defect/modeled fault is present in every time-frame, we now need 9-value algebra instead of just 5 values in combinational ckts

14. Approximation using 5-value algebra

    - use 5 values only, but must check after sequence is generated to make sure target fault is indeed detected

15. Some faults prevent the faulty machine from being initialized
    $\longmapsto$ to detect such faults, we need multiple observation strategy by exhausting all combinations of initial states

16. We can make better controllability/observability measures to help making better decisions in sequential ATPG

    - in addition to C0, C1, and O, also add Drivability, D(), of a signal
    - D(g) tells how easy/hard it is to drive a D or $\overline{D}$ from g w.r.t. controllability values

- it further differentiates observing a D from a $\overline{D}$

Example 4:

17. S-Graph of a sequential Circuit: a graph where vertices are FFs and directed edges between 2 vertices indicate a combinational path exists between them.

    Example 5:

18. Properties of S-Graph

    - If s-graph is acyclic (no cycles), then the faulty state is always initializable
    - Define: d_{seq} = sequential depth of the circuit = # FFs on the longest path in s-graph

- A test sequence for a detectable, non-FF fault in a cycle-free circuit has at most $d_{seq} + 1$ vectors.
- If s-graph contains cycles, the test sequence length is unbounded, since the sequential depth would be ∞

19. Sequential ATPG for acyclic circuits

- If circuit is acyclic, then any fault can be tested within $d_{seq} + 1$ vectors
- Unroll circuit $d_{seq} + 1$ time frames, and use combinational ATPG?
  $\longmapsto$ not so fast, fault is present in every time frame
- In acyclic circuits, one can often lay out the FFs in a pipepline fashion, implicitly unrolling the circuit. In this case, perhaps each gate appears only once in the rolled-out circuit?
  $\longmapsto$ what if there exist multiple paths to a FF?

Example 6: