VLSI / SOC Testing

Lecture 13

1. Testing Acyclic Sequential Circuits
   - Transform acyclic sequential circuit to a balanced combinational circuit $\longmapsto$ targeting some single faults may become multiple-faults
   - Apply only combinational ATPG to get test vectors
   - Transform vector to test sequence
2. Balanced Model (BM) of a circuit:
   - In a balanced circuit, all signal paths between any 2 nodes have the same number of FFs
   - Internally balanced: a balanced circuit where all node-pairs except those involving PIs are balanced
   - Strongly balanced: more strict than balanced, in which all reachable paths from an internal node to all PIs have the same sequential depth
3. Balanced Model Generation:
   - Step 1: Compute the weight of each PO to be the maximum sequential depth from any PI to the target PO
   - Step 2: For each PO, compute the weights of all gates within its cone, duplicate and split whenever necessary

Example 1:

4. Issues in balanced model:
   - Most sequential circuits are not balanced
   - Duplication-and-split may become expensive
   - Some faults become multiple-stuck-at-faults
   - Positive: combinational ATPG generally less expensive than sequential ATPG
5. Simulation-Based Sequential ATPG
   - Avoids backtracing and backtracking
   - Simplest: (weighted) random TG
6. CONTEST: 3 phases
   - phase 1: initialize fault-free circuit (no X's in FFs) by guided random walk
     $\longmapsto$ generate a set of vectors, pick the best, where
     $\longmapsto$ cost function = # FF's initialized
     $\longmapsto$ repeat till all FFs specified
     $\longmapsto$ Then, drop all detected faults by the initialization sequence

- phase 2: concurrent fault detection
  $\longmapsto$ let current state be s_f
  $\longmapsto$ generate a set of vectors, pick the best, where
  $\longmapsto$ cost function = # additional faults detected from s_f
  $\longmapsto$ repeat till remaining faults < threshold
- phase 3: single fault target
  $\longmapsto$ pick one fault at a time from remaining faults
  $\longmapsto$ generate a set of vectors, pick the best, where
  $\longmapsto$ cost function = $K\times$ activation + propagation
  $\longmapsto$ repeat till remaining faults < another threshold

7. GA-Test:
   - Similar to CONTEST, except now use GA to optimize/find best vector
   - No single target fault phase (only phases 1 and 2)
   - Instead of adding 1 vector at a time, individuals consist of sequences
   - GA individual:
8. Define: *distinguishing sequence:* a sequence, when applied, can distinguish 2 or more states by producing different output sequences
9. DIGATE
   - Power of distinguishing sequences: if it can distinguish a large set of states from another, it is very powerful
   - Dist. seqs learned by the ATPG at run-time. For each FF, store up to 5 dist. seqs.
   - Application to ATPG: when a fault is activated to one or more FFs, propagation of the fault effect is similar to distinguishing the faulty machine from the fault-free machine

10. Distinguishing Sequence Learning

11. DIGATE Notes

    - Fault-free and faulty circuits not exactly identical (but very similar), thus distinguishing sequence may not always work
    - Instead of targeting a group of faults, target single faults
    - For each fault, first activate it to a FF, then seed dist. seq. learned in the GA population to help propagate the fault effect to a PO
    - Dist. seqs useless if target fault is not activated

12. Targeting fault activation

    - Divide fault activation into 2 phases: (1) single time frame, (2) state justification
    - Single time frame is similar to combinational GA-based ATPG, with a relaxation step to make some FFs don't-cares
    - State justification: justify the required FFs

Example 2: State Relaxation

13. State Justification of the relaxed state

    - Define: set/reset sequences: a sequence that sets/resets a particular FF from an all-unknown state
    - Use set/reset sequences to guide GA to justify the required relaxed state

14. Problems with set/reset sequences

    - set/reset sequences sets/resets the specified FF, and at the same time, may set/reset other FFs as well
      $\longmapsto$ conflicts in trying to justify a combination of FFs simultaneously

15. Pseudo-register justification

    - Instead of set/reset sequences for individual FFs, use sequences that sets a group of FFs to a specific value

16. STRATEGATE: state-traversal based

    - treat entire state instead of partitioning the state
    - find similar states visited before, and use those sequences to justify the desired target state
    - data structures:

17. Genetic engineering partial solutions to get the target state

18. Logic-Simulation Based Sequential ATPG

    - Observation: effective traversal of circuit state space helps to detect more faults
    - Thus, using only logic simulation, traverse as many states as possible
    - problem: fault coverage plateaus early. Addition of arbitrary new states may not be helpful.
      $\longmapsto$ need to differentiate new states

19. Differentiation of new states