VLSI / SOC Testing

Lecture 22

1. Memory testing
   - defect may be in address decoder or cell array
   - since each cell of memory can have a different state, the # of vectors to test a memory is potentially very huge

2. Address decoder faults
   - no cell accessed for a given/specific address
   - a given cell can't be accessed by any address
   - a given address accesses multiple cells
   - a given cell can be accessed by multiple addresses

3. Cell-array faults
   - stuck-at: a bit cell stuck at a value
   - transition fault: a cell fails to make transition from 0 to 1 or vice versa
   - coupling fault: write in one cell affects a neighboring cell
     - idempotent coupling: neighboring cell affected the same way
     - inversion coupling: neighboring cell affected the opposite way
   - pattern-sensitive fault: can't write a specific value when neighboring cells have a particular pattern

   Example 1:

4. March Test Terminology

   - : stepping through memory array and tests all cells by ensuring that both the addressing and cells can retrieve/store 0 and 1
   - $\Uparrow_0^{n-1}$ $(r0, w1)$: read, compare if 0, then write 1 to all cells in ascending order. Complexity of $2n$. Can take care of forward coupling faults
   - $\Downarrow_0^{n-1}$ $(r0, w1)$: descending order, takes care of backward coupling faults

5. Basic March Test Algorithm

   $\Uparrow_0^{n-1}$ $(w0)$: initialize memory to all 0s
   $\Uparrow_0^{n-1}$ $(r0, w1)$: for $i = 0$ *to* $n - 1$
   read(i)
   compare with expected value 0
   write (i,1)

6. Sample march tests:

   - MATS: $\Uparrow (w0)$; $\Uparrow (r0, w1)$; $\Uparrow (r1)$: covers forward coupling and stuck-at faults
   - MATS+: $\Uparrow (w0)$; $\Uparrow (r0, w1)$; $\Downarrow (r1, w0)$: covers also backward coupling

7. Testing for inversely-coupling faults

8. ATPG for march tests

   - For a given fault, generate the corresponding march test
   - Problem: march test can be complex and requires long application time
   - In general, three steps are needed:
     - initialization

  - excitation
  - effect propagation via verifying/reading necessary cell values

Example 2: testing coupling fault

9. Viewing coupling faults by FSMs
   - apply FSM testing approaches to test memories

10. Testing Word-Oriented Memories
    - Intra-word coupling faults
    - Many possibilities exist
    - m-out-of-n codes (Hamming distance between any two code words is m) $\longmapsto$ m = n/2

Example 3:

11. Testing pattern sensitive faults

    - basic idea: initialize the base cell, then change the patterns around the base cell to see if content of base cell changes
    - Issues: how to minimize the number of patterns applied

12. Graph traversal

    - let each node in graph denote a specific pattern
    - an edge exists between two nodes if the patterns differ in only 1 bit
    - Hamiltonian path/sequence traverses the graph such that each node is visited exactly once
      $\longmapsto$ each pattern applied exactly once
    - Eulerian path/sequence traverses the graph such that each edge is traversed exactly once
      $\longmapsto$ each pattern may be applied multiple times, but all single bit transitions in the patterns exercised

    Example 4:

13. Dynamic faults: faults that require multiple, successive operations to sensitize. (eg. write followed by an immediate read flips the bit value)

14. Testing System-On-A-Chip

    - System consists of interconnecting blocks/cores
      $\longmapsto$ benefits: design reuse, test reuse, etc.
    - Test access mechanisms:
      - global test bus to deliver test patterns to each core
      - wrapper around each embedded core for test access purposes
      - self-test using embedded programmable cores as TPGs (software-based testing)
    - Issues
      - area overhead
      - test application time
      - power consumption during test

    Example 5:

    Example 6:

15. Area Overhead
    - Does TAM need to be available for every core? In other words, can a given core be testable without TAM?
    - Need: testability analysis method to evaluate embedded core's testability
    - Alternative: transparency mode in the predecessor core to allow transport of test vectors
      $\longmapsto$ must make sure costs due to transparency is less than conventional TAM

    Example 7:

16. Test Resource Partitioning and Test Scheduling
    - TAM bus width, wrapper, power, BIST, etc. are limited resources
    - even if power is not exceeded, it may not be possible to test two cores in parallel due to bus-width limitation
    - derive a schedule by which cores are to be tested
    - test cores in parallel to reduce test application time
    - must monitor power consumption
    - Need: constrained scheduler/optimizer
17. Test data compression
    - test data for all cores on an SOC can potentially be very large
    - unlike compaction, compression is to reduce the overall test data volume by compressing the data

- such as gzip, compress, etc.
- problems: gzip and compress generally don't work well on vectors
- need: other compression methods
- need: low-cost on-chip decompressor

18. Compression by Colomb code

    - based on (1) difference vector set and (2) run-lengths

    Example 8: