Design Verification

Lecture 23 - Diagnosis I

1. Objective: identify locations where the error may reside
   - Given an implementation that incorrectly models the spec, we'd like to know what causes the difference
   - Good for debugging, time-to-market, etc.
2. Some definitions:
   - primary output pair (O_i^S, O_i^I): the i^{th} output pair for the specification and implementation
   - erroneous (input) vector: a binary input vector that can differentiate at least one primary output pair
   - erroneous output: if an erroneous vector v differentiates the i^{th} primary output pair, then O_i^I is an erroneous output; otherwise, O_i^I is the correct output with respect to v.
   - correctable vector: an erroneous vector v is correctable by a signal a in the implementation circuit if there exists a new function for signal a such that v no longer is an erroneous for the resulting new circuit
     $\longmapsto$ Note: signal a can correct erroneous vector v iff (1) there exists a sensitizable path from a to the erroneous output, and (2) v cannot sensitize a discrepancy from a to any correct output
   - single-fix: signal a is a single-fix for the implementation circuit iff every erroneous vector can be correctable by a

Example 1

3. If the erroneous implementation is single-signal correctable, then there exists a single fix signal a. What should we substitute a with?

   - Define *unconstrained erroneous vector set*:

     $$E(X,\ a) = \Sigma_{i=1}^{n}(f(O_i^S(X)) \oplus f(O_i^I(X,a)))$$

     $E(X,\ a)$ represents the set of all input vectors (plus signal a) that causes at least one primary output inconsistent with the specification
     $\longmapsto$ $E(X,\ a)$ can be represented as a BDD
   - Want: $E(X,\ a^{new}) \equiv 0$, where signal a is replaced by a new function a^{new}

   $\longmapsto$What if implementation is not single-signal fixable?

   - one possibility is to partition the erroneous outputs, and find a single fix for each erroneous output cone.

4. Simulation-based diagnosis approaches

   - main idea: gradually filter the error locations based on the following heuristics: cone intersection, sensitization, back-propagation

5. Diagnosis by cone intersection

   - step 1: divide the outputs into correct outputs and erroneous outputs
   - step 2: compute fanin cones starting from the erroneous outputs

- step 3: compute intersection of the erroneous output cones
- Key: any signal not included in the intersection cannot be solely responsible for the error

Example 2

6. Diagnosis by sensitization

- idea: if v is correctable by signal a, then there must exist a sensitizable path from a to some erroneous output(s)
- note: there may result in multiple candidate error locations, and there might have existed multiple errors in design
- Algorithm:

for each erroneous vector v
  logic simulate with v
  for each signal a
    complement the value at a
    simulate the fanout cone of a due to this complementation
    if one or more erroneous output value is flipped, then v can sensitize a discr

Example 3

7. Diagnosis by back propagation

   - similar to critical path tracing, it finds candidate signals by backtracing from erroneous outputs
   - backtracing is done by traversing through sensitizable paths

   Example 4

   Example 5

8. Vector dependence of errors

   - vector-independent:
   - vector-dependent:

9. Diagnosis by fault simulation

   - idea: if signal a is a candidate error location, then for an erroneous vector, the stuck at fault at a must be detectable, and in fact, correct some or all erroneous outputs
   - note: due to vector dependence, different stuck values on a may apply to different erroneous vectors
   - thus, signal a can correct erroneous vector v if either stuck-at-0 or stuck-at-1 fault on a can be detected
   - Algorithm:

     for every erroneous vector v
       logic simulate v
       for every candidate signal a
         fault simulate a-stuck-at-$\overline{v}$
         if none of erroneous outputs eliminated
           remove a from candidate signal list

10. Extension to multiple errors

    - we can extend the fault simulation concept to 2 or more errors
    - Algorithm:

      for every erroneous vector v
        logic simulate v
        for every candidate signal pair (a, b)
          fault simulate multiple fault (a-stuck-at-$\overline{a}$, b-stuck-at-$\overline{b}$
          if none of erroneous outputs eliminated
            remove pair (a, b) from candidate signal list

    - problem: $O(n^2)$ possible pairs, and $O(n^k)$ for k design errors
    - problem: if more than 2 design errors, simulation becomes expensive

11. Diagnosis by X-list

    - idea: if signal a is a candidate error site, then there must exist a sensitizable path. Thus, if a don't-care (X) is placed on a, it should propagate to at least some erroneous outputs
    - Algorithm:

score for every signal = 0
for every erroneous vector v
  logic simulate v
  for every candidate signal a
    place an X on a and simulate
    score(a) += computeScore()
      remove a from candidate signal list

- score of simulating X from a is computed as follows
  - if X propagates only to correct outputs, score -= K1
  - if X propagates to erroneous outputs, score += K2
  - if X propagates to combination of erroneous and correct outputs, score += K3, where K3 is smaller than K2
- We can extend the X-list concept to regions of Xs
  - a region is defined by a center and a radius
  - key concept: if by placing X's at the outputs of the region, and the X's do not propagate to any erroneous outputs, then we can safely conclude that the error is not fully contained in the given region

Example 6

12. Important issues in diagnosis

- diagnostic accuracy - was the actual error in the final candidate list?
- diagnostic resolution - among the final candidate list, how was the actual error ranked?
- ranking of the candidate error locations can be done by scoring