Design Verification

Lecture 15 - RTL to Gate-level Issues

1. RTL to gate-level verification
   - step 1: perform a fast-synthesis of the RTL circuit
   - step 2: perform a gate-level equivalence checker
   - issues: fast synthesis from undefined behavior at RTL introduces don't care conditions
     $\longmapsto$ need to incorporate don't care conditions
2. Verify block by block
   - verify a portion of RTL code with a gate-level block
     $\longmapsto$ need to first establish signal correspondences between RTL and gate-level
   - Easy *if* the signal names are preserved between RTL and gate-level
     can't count on name preservation
   - If names not preserved, how to identify correspondences? (mapping problem)
3. Other benefits of establishing signal correspondences / mapping relation
   - allow designer to identify a portion of the gate-level implementation to a section of its RTL spec
   - allow to identify portions of design unlikely to change after modifications made in RTL
     $\longmapsto$ saving re-synthesis work
     $\longmapsto$ re-verify on the portions that were changed
   - allow incremental verification of blocks
4. Basic idea in the mapping problem
   - Assume for a given RTL signal s, there exists a gate-level net g
   - Then, the responses for s should match the responses for g for each input vector over a given input vector set

- But, checking for matching responses for each (RTL signal, gate-level net) pair expensive
- Alternative: suppose RTL signal s is stuck-at-v, the resulting response of primary output signals with this fault may differ from the original fault-free RTL response
- Idea: can we take the faulty RTL response with s stuck-at-v and try to diagnose in the gate-level circuit to see which net would be the faulty site?

Example 1

5. Treat the mapping problem as a diagnosis problem
   - need: the faulty RTL response must differ from the fault-free RTL response when s is assumed stuck-at-v ($v = 0$ or 1)
   - Thus, the input vector set must be rich enough to capture many faults
   - We can generate input vector set, T, by gate-level ATPG programs

Example 2: Injection of fault at RTL

Example 3:

6. Diagnosis procedure

- Given gate-level circuit C, vector set T, fault-free response G to T, and faulty response U (obtained on RTL circuit), we'd like to identify in the gate-level circuit presence of which fault corresponds to U
- may need to try two separate stuck-faults at RTL in case T cannot distinguish one fault from the fault-free circuit
- if neither stuck-at-1 nor stuck-at-0 fault at RTL produces different responses at the primary output, we can't proceed further
- Simple procedure:
  - 1: for each fault f in gate-level circuit C
  - 2: inject f into circuit C
  - 3: simulate C with T and obtain output response U_f
  - 4: if $U_f \equiv U$ for every vector in T, f is the corresponding fault in gate-level netlist to the RTL fault
  - Note: fault f is some net n stuck at 0 or 1, n would be a possible correspondence to signal s in RTL
  - Note 2: there may exist multiple nets at gate-level that produce identical U_f; multiple matches possible
- Speed-up heuristic: for a given fault f, can stop simulation as soon as U_f differs from U in steps 3 and 4

7. Diagnostic issues

   - since more than 1 fault may produce *same* output responses on T
   - diagnostic quality of T important, in addition to fault coverage of T
     $\longmapsto$ Even if T can distinguish f from the fault-free circuit, we'd like T to produce a *unique* response for f from all other faults
   - two faults f_1 and f_2 are *indistinguishable* if the responses of the circuit with the faults are identical to test set T. They are *distinguishable* otherwise.
   - One solution to enhance diagnostic quality of T: *diagnostic test generation*, in which a vector set is produced such that no 2 unequivalent faults produce the same faulty output response.

8. Incremental mapping

   - Suppose there are a number of RTL signals we wish to find mapping gate-level nets
   - Once a mapping is found for one signal, convert the correspondence pair (s, n) into primary outputs (for simulation only)
     $\longmapsto$ This results in more outputs to compare, thus we may identify other correspondences more quickly

Example 4

9. Simple diagnostic test generation

Diagnostic_ATPG {
perform regular test generation to obtain initial vector set T
identify indistinguishable pairs (f_i, f_j) using T
for each indistinguishable pair (f_i, f_j) {
generate an additional test t for f_i such that f_j is not detected
or f_j is detected on different primary output
if previous failed, generate another test t for f_j such that f_i is not detected
or f_i is detected on different primary output
}
append additional vectors t to T
}

- NOTE: the constrained ATPG for f_i or f_j different, since the objectives differ. Generating a distinguishing test for one case may be easier than the other