Design Verification

Lecture 10 - More Multi-Level Logic Verification

1. Probabilistic verification

   - To overcome size and complexity of boolean comparison such as OBDD
   - Achieve near 100% confidence on equivalence
   - Instead of evaluating on boolean vectors, integer vectors are used
     $\longmapsto$ Need to map boolean function to an arithmetic function for both spec and impl circuits $\longmapsto$ Evaluate integer vectors on these arithmetic functions to form *hash codes (H)* and check for equivalence $\longmapsto$ Exponential time in boolean verification reduceable to polynomial time in integer codes

2. Boolean to arithmetic transformation: **A**-transform

   - $\bar{x} = (1 - x)$
   - $x \wedge y = x \times y$
   - $x \vee y = x + y - x \times y$
   - note: all arithmetic operators are conducted modulo p, where p is a prime integer
   - compute hash functions H_1 and H_2 for functions f_1 and f_2
     $\longmapsto$ if $H_1 \neq H_2$, then we know *for sure* that the two functions are inequivalent
     $\longmapsto$ else we can say that the two functions are equivalent with a *very small* probability of error

   Example 1

3. Shannon's expansion applies to $\mathbf{A}()$

Example 2

4. Error Bounds

   - error can occur on aliasing effects
   - eg. when both the resulting integer $= 0$
   - Thus, a randomly chosen vector distinguishes the 2 functions with prob of at least $\frac{(p-1)^n}{p^n} \approx (1 - \frac{n}{p})$, where n is the number of inputs
     $\longmapsto$ if in a 64-input circuit, and p is a large 32-bit prime, then error $\epsilon = 1 - (1 - \frac{n}{p}) = \frac{n}{p} \approx 1.5 \times 10^{-8}$ (15 in a billion chance)
   - can reduce this error prob by applying k multiple runs of applying integer vectors. error prob now becomes ϵ^k
   - one may also avoid apply integer 0 or 1 as vectors

5. Mixed-mode

   - for n inputs, we can transform v to integers and $(n - v)$ remain as Boolean
   - boolean evaluation faster than arithmetic multiply
   - disadvantage: error prob increases
   - key: how to partition v variables

6. Implementation issues

   - one can build BDD for boolean function and convert that to hash function
     $\longmapsto$ need to build BDD - expensive
   - build BDD incrementally, as **A**-transform also takes place
   - convert to an equation and compile/execute. Size of equation may be large and involves modulo operations

7. Results

   - can handle large circuits that OBDD can't in fraction of time

8. Timing verification

   $\longmapsto$ Critical path = maximum delay path in combinational portion of circuit
   $\longmapsto$ Need to analyze and verify critical path to meet clock period
   $\longmapsto$ But there are too many paths!!

9. Define: data-ready or arrival time: time at which the signal would settle.

$$t_i = d_i + Max_{(j:(v_j,v_i)\in E)}\ t_j$$

   Example 3

10. Given a critical path requirement, we can obtain *required data-ready* time:

$$\bar{t_i} = Min_{(j:(v_i,v_j)\in E)}\ \bar{t_j} - d_j$$

   slack: (quantity of) difference between required arrival time and actual arrival time:

$$s_i = \bar{t_i} - t_i$$

Example 4

11. So far, we only talked about *topological paths* (based on graph of the circuit). It is possible that a topological path is a false path!

    Define: false path: a path when no event (signal transition) can propagate along it.

    Without eliminating false paths, longest topological path(s) may be pessimistic.

    Example 5

12. Define: a path is *sensitizable* if an event can propagate from its tail to its head.

    A critical path is a sensitizable path of maximum length.

    Example 6

13. Fixed delay vs. **bounded delay**

- Fixed delay is unrealistic, since we're dealing with abstractions of a fabricated circuit. In addition we're analyzing a family of such chips/circuits, not just a single chip.
- Need: *best* and *worst case* bounds on delays:
  $\longmapsto$ Bounded delay: (min, max)
  $\longmapsto$ Very difficult to simulate
- If min_delay $\neq 0$, we may not satisfy the *monotone speed-up* property, i.e., speeding up one gate may slow down the entire circuit.
- If min_delay $= 0$, then monotone speedup property is preserved.

Example 7

14. Define: Controlling value for an AND gate is 0.

    A gate is *controlled* if its output is a controlling value.

15. Define: a path is statically sensitized by vector V, if along each gate on the path, the gate output is a controlling value, and side-inputs to the path are all non-controlling.

16. How about a gate on path with 2 controlling input values?

    $\longmapsto$ Not statically sensitizable, but may be *co-sensitizable.*

17. In order to identify *true* false paths, at least one of the following 3 conditions must hold for **all possible** input vectors.

    - A gate along the path is controlled, not by the path input, but by a side-input
    - A gate along the path is controlled by both path and a side-input, but the side-input controlling value arrives first
    - A gate along the path is NOT controlled, but a side-input presents the non-controlling value last

    Example 8

18. Verification for power consumption: both **average** and **peak** power important
    - Guarantee battery life
    - Design will not result in hot spots
    - Ensure circuit reliability
    - Power Supply Integrity
    - Re-wiring of old buildings
19. Sources of power dissipation:

    $\longmapsto$ Static: leakage currents

    $\longmapsto$ Dynamc: short-circuit and switching current
20. $P = \frac{1}{2}CV^2\ f\ N$, where
    - C = output capacitance
    - $V = V_{dd}$
    - f = clock frequency
    - N = # times gate switch in one clock cycle
    - Need a vector pair to account for power. The first vector initializes all the gates in the circuit, the second vector toggles some gates
    - If assuming zero-delay, $N = 1$ at most (i.e., a gate can switch at most one time)
21. $P = \frac{1}{2}V^2\ f\ \sum_{i=1}^{n} C_i\ N_i$ for all n gates

    Example 9 (assuming 0 delay)

    Example 10

22. Signal Probability: probability of a signal/gate = logic 1
    - P(PI) = 0.5
    - P(switch on PI) = P(01 or 10) = P(01) + P(10) = $(0.5 \times 0.5) + (0.5 \times 0.5)$ = 0.5
23. Average Switching Activity

    Example 11

24. Static Signal Probabilities
    - NOT gate: $P_Z = 1 - P_A$
    - AND gate: $P_Z = P_A \times P_B$
    - OR gate: $(A+B = \overline{\bar{A}\bar{B}})$ $P_Z = 1-((1-P_A)(1-P_B)) = P_A+P_B-(P_A \times P_B)$

    Example 12

25. To avoid signal correlation $\rightarrow$ write function as a **disjoint** sum of products

    Example 13

26. Given switching probability, compute switching activities $\Rightarrow$ Need also gate delay effects

    Let $e_g = g(0) \oplus g(t)$:

    - g(0) = initial value of gate g
    - g(t) = value of gate g at time t
    - $e_g = 0$, if g(0) = g(t) = 0;
    - $e_g = 0$, if g(0) = g(t) = 1;
    - $e_g = 1$, if g(0) = 0 and g(t) = 1;
    - $e_g = 1$, if g(0) = 1 and g(t) = 0;

27. So,

    $N_{avg} = \Sigma_{for\ all\ gates\ g} N_g$
    $N_g = Prob(e_g)$
    $N_g(g = PI) = 0.5$
    $N_g(othergates) = P(f(g = 0) \oplus f(g = 1))$

    Example 14

    $P(g == 1) = 0.8 \rightarrow P(g == 0) = 0.2$
    $N_g = 2(0.8 \times 0.2) = 2 \times 0.16 = 0.32$

    Example 15 (Unit Delay)

28. Peak Power: Need to find an input vector pair that maximizes circuit activity
    - Aspect 1: maximize switching on gates with many fanouts $\rightarrow$ can be achieved using test generation techniques
    - Aspect 2: maximize # toggles on every gate $\rightarrow$ need delay information
    - NEED TO CONSIDER BOTH ASPECTS!!
29. Exact Peak Power difficult to estimate:
    - Lower bound for peak: power is attainable
    - Upper bound: actual peak power may be lower than this bound
      - Compute peak switching frequency for each node
        $\rightarrow$ get all possible switching times (need delay information)
      - Sum up for all gates
        $\rightarrow$ this is a loose upper bound
30. Power in Sequential Circuits: Power vectors consist of PI's and FF state
    - Issue 1: probability of machine being in a particular state
    - Issue 2: intermediate state not fully controllable
31. To resolve Issue 1:
    - Need STG or extract it from netlist
    - calculate probability of circuit being in each state:
      $Prob(S_i) = \Sigma_m prob(S_m) \times prob(edge_{m \rightarrow i})$
      where $m = \#\ fanin\ edges\ to\ S_i$

$$\sum_{i=1}^{k} P(S_i) = 1$$

Example 16

32. Resolving Issue 2: correlation between starting & intermediate states
    - One can take account of all state combinations
    - Exact method *exponential* in computation (such as Chapman-K)
      $\rightarrow$ Need approximation methods!
33. Approximation Method
    - Assume all present state lines independent
    - Simply propagate line probability as in combinational circuits

Example 17

34. High-Level Power Estimation Goal: We want to know power dissipated as a igh-level model, without having to go through gate-level simulation

    $\longmapsto$ in order to satisfy the power constraints before any synthesis considerations, including different scheduling, resource sharing.

    $\longmapsto$ without high-level power estimator, the design has to be fully synthesized to gate-level, before any power estimation is performed. This is inefficient and expensive.

35. Different from low-level power estimation in that power is estimated not at the gate level, but at an equation/model level. However, low-level estimation is more accurate

36. Without low-level information, absolute accuracy is not as important; rather, relative accuracy is of interest.

    $\longmapsto$ allow quickly estimation of power without simulating 100K+ gates
    $\longmapsto$ compare power efficiency of different hardware configurations

37. Main technique: Macro-Model

    $\longmapsto$ A Macro-model is an equation/model which gives power in terms of some quantities, which are easily observable at high-level.

    Example 18

38. Different Macro-models

    - Analytical: independent of internal structure, but on some parameters that characterize the complexity of the circuit.
    - Pre-characterization: use information from low-level implementation. It is generally more accurate.

39. Analytical Methods: Entropy measure

40. Pre-characterization: given a set of vectors, calculate P_{in}, D_{in}, D_{out}, etc.

$\longmapsto$ Given parameters of an input set (P_{in}, etc.), we can compute the average power dissipated for the entire set.

41. Cycle-by-cycle power estimation: instead of having one average power value, can we use the macro-model to estimate pair-wise through a given vector set?

42. Architectural level power: estimating power consumed by a program on a target architecture.
    - estimate power of a particular instruction
    - granularity: macro-model for the entire processor, or partition the processor into sub-components
    - trade-off between speed and accuracy
43. One power measure per instruction
    - memory instruction
    - ALU instruction
    - branches
    - no-ops
44. Estimate power by architectural simulation
    - simulate each pipeline stage and estimate power for each stage.
    - may capture some circuit activity information by switches on buses, etc.
45. Low-power architectures?
    - voltage and frequency scaling
    - power mode selection for the memory (turn parts of memory to low-power mode)
    - how do we guarantee execution correctness in place of component power-downs?