Design Verification

Lecture 11 - Sequential Logic Verification I

1. FSM Design Flow: (1)*Specification* → (2) *State Minimization* → (3) *State Encoding* → (4) *Logic Optimization*

   We may need to verify circuits between any two steps of synthesis

   - State Minimization: minimize # states & transitions in state transition graph (STG)
   - State Encoding: will affect # literals in logic optimization.
     (a) minimize # product terms in 2-level representation for combinational logic, **OR**
     (b) minimize dependence among state variables → in turn, will affect # literals
     (c) estimate cost in multi-level representation

2. Views of FSM

3. Formal FSM Model & Notation: 6-tuple $< I, S, \delta, S_0, O, \lambda >$

   - I: Inputs
   - S: Finite set of states
   - δ: $S \times I \rightarrow S$ (next state function)
   - S_0: set of initial states
   - O: Outputs
   - λ: output function
     (1) $S \times I \rightarrow O$: Mealy machine
     (2) $S \rightarrow O$: Moore machine

4. Define: 2 states s_1 and s_2 are *distinguishable* if $\exists$ a sequence T such that $\lambda_1(s_1, T) \neq \lambda_2(s_2, T)$. In other words, output sequences of applying T to the 2 states differ. The sequence T is called a *distinguishing sequence* for (s_1, s_2).
$\mapsto$ If distinguishing sequence exists, worst-case/longest distinguishing sequence length $|T| = |S|$.

5. If no distinguishing sequence exists for (s_1, s_2), then the state pair (s_1, s_2) is said to be an *equivalent state pair.* Stated differently, s_1 & s_2 are equivalent iff each input sequence starting from s_1 yields an output sequence identical to that attained by starting from s_2, for all possible & legal input sequences.

6. Define: *Reset equivalence*: 2 sequential circuits, C_1 and C_2, with external reset signals are *reset equivalent* iff (s_1, s_2) are equivalent, where s_1 is the reset state of C_1, and s_2 is the reset state of C_2.

7. Identify state equivalence:

   - Naive Algorithm

     ```
     for each pair $(s_1, s_2)$ in FSM
        simulate $(s_1, T_i)$ and $(s_2, T_i)$ for all possible & different input sequence $T_i$
        if (output sequence differ)
           mark $(s_1, s_2)$ non-equivalent
     ```

   - Unmarked pairs are equivalent! Simply FSM accordingly.

     Example 1

8. More on state equivalence:

   - 2 states are equivalent iff they are n-equivalent, $n = |S|$.
   - 2 states $s_1,\ s_2$ are k-equivalent ($s_1 \equiv_k s_2$) iff $\nexists$ a distinguishing sequence of length k or less for s_1 & s_2.
   - 1-equivalence: examining the outputs of state transitions of states s_1 & s_2
   - k-equivalence: iteratively determine $(k-1)$-equivalences.

   Example 2

Example 3: Moore Machine (very similar to previous Mealy Machine)

9. Alternative for completely specified FSM: verify by reducing FSM Complexity $= O(N \; lg \; N), \; N = \#$ states (instead of checking for reset equivalence)

   - Given 2 STG's G_1 and G_2 (both completely specified)
     if starting/reset states $s_1 \in G_1, \; s_2 \in G_2$ given
     $\rightarrow$ Reduce G_1 to G_1', G_2 to G_2'
     if $|G_1'| \neq |G_2'|$, then they are not equivalent
     else check isomorphism/equivalence with (s_1, s_2) as the starting pair

10. The above techniques may be expensive for circuits with large number of states.
    $\longmapsto$ Need to enumerate states until the reset states are distinguished. Worst case: need complete reachable state space
    $\longmapsto$ Alternative is to use ATPG to detect the miter output fault, with constraint added for the stopping criteria (don't need to backtrace to all-unknown state). ATPG for the target fault may be exponential in complexity as well.

11. For circuits without reset (still completely specified FSM)

    - Given 2 STG's G_1 and G_2 (both completely specified)
      $\rightarrow$ Reduce G_1 to G_1', G_2 to G_2'
      if $|G_1'| \neq |G_2'|$, then they are not equivalent
      else
      $\rightarrow$ concatenate G_1', G_2' to get G_3
      $\rightarrow$ reduce G_3 to G_3'
      $\rightarrow$ equivalent(G_1, G_2) iff $|G_3'| \equiv |G_1'|$

12. For incompletely specified circuits without reset
    $\rightarrow$ Reduced STG's are NOT canonical!!

    - There may exist 2 or more irredundant STG's of different size for the same STG
    - This is due to compatibility issues

    Example 4

13. For circuits without reset (FSM may *not* be completely specified)

    - different notions of equivalences possible
    - 0. classical FSM equivalence: for every state in one circuit, there is an equivalent state in the other, and vice versa. Computationally expensive.
    - 1. sequential hardware equivalence (SHE): two designs have equivalent behavior *after* synchronization of the circuits via an aligning sequence
      $\longmapsto$ an aligning sequence T_a is a sequence that takes M_1 to s_1 and M_2 to s_2 such that (s_1, s_2) is an equivalent state pair
    - 2. safe replacement equivalence: stronger than SHE, as one circuit can replace another, and vice versa:
      $\longmapsto$ if for any state $s_j \in M_2$, there exists a state $s_i \in M_1$ that can produce the same output sequence for any input sequence T applied. Then, M_2 is a safe replacement for M_1.
    - 3. three-valued equivalence: relaxed version of safe replaceability but stronger than SHE
      $\longmapsto$ $\lambda_1(uuu, T) \equiv \lambda_2(uuu, T)$ for any vector sequence T. In other words, outputs must match (they must belong to (0,0), (1,1), or (x, x), ie., even for don't-cares. But if the output is don't-care (from an initial unknown state), we only need to make sure the other circuit also produces don't-care.
    - 4. delay replacement equivalence: relaxed version of safe replaceability but strong than SHE.
      $\longmapsto$ If for any state $s_j \in M_2$, there exists a state $s_i \in M_1$ that can produce the same output sequence for any input sequence T, after n clock cycles in M_2, then M_2 is a delay replacement for M_1. Essentially, it is like safe replacement allowing for an n cycle delay version of the implementation.

14. Relationship among different notions of sequential equivalence

15. Define: a *synchronizing sequence* is an input vector sequence that can bring the machine to a unique state from any state.

16. Define: an *initialization sequence* is a synchronizing sequence that can be verified by 3-value simulation (i.e., $\delta(uuu, T) = s_1$ when T is an initialization sequence under 3-value simulation)
$\longmapsto$ an initialization sequence is also a synchronizing sequence, but not vice versa

17. Define: two states s_0 and s_1 are *alignable* if $\exists$ a sequence T such that $\delta(s_0, T) \equiv \delta(s_1, T)$. (states reached after application of T are equivalent.
$\longmapsto$ a universal alignment sequence T_a: $\delta(s_0, T_a) \equiv \delta(s_1, T_a) \forall (s_0, s_1)$.

18. Define: Terminal Strongly Connected Component (TSCC): a SCC that does not have any outgoing edges

19. Post-synchronized reachable state space is a TSCC

20. Reset Equivalence

    - Possible Algorithm:
    - Let T_1 and T_2 be the synchronizing/initialization sequences for C_1 and C_2 respectively. Note that the concatenation $T_1 \cdot T_2$ is also a synchronizing/initialization sequence. Let the states reached by $T_1 T_2$ be s_1 and s_2 in C_1 and C_2, respectively.
    - C_1 and C_2 are sequential hardware equivalent iff (s_1, s_2) is an equivalent state pair (i.e., $T_1 \cdot T_2$ is a universal alignment sequence)
    - What if sync sequences do not exist? Need to find the aligning sequence.
    $\longmapsto$ need to find an equivalent state pair $(s_1,\ s_2)$ such that both states are reachable in their respective circuits

21. Theorem: Two circuits are SHE iff all state pairs are alignable [Pixley 92]
22. Corollary: Two circuits are SHE iff there exists a universal alignment sequence
    - Possible Algorithm:
    - first find a set of equivalent state pairs $S_{\equiv}$.
    - compute the preimage (all states that can reach) of $S_{\equiv}$.
    - if the preimage is the universe of all states in the product machine, then the two circuits are SHE.
    - In fact, if the preimage contains any reachable state in the miter, then the two circuits are SHE!
23. Safe Replacement
24. Delay Replacement
    - With initial states of the 2 circuits in the miter unconstrained, check if it is possible to produce a 1 at the XOR miter output after k clock cycles
25. For all notions of sequential equivalence, the problem boils down to determining if a state pair is an equivalent state pair. Future lectures!!