Design Verification

Lecture 16 - Model Checking I

1. Motivation for Model Checking

   - equivalence checking may not be feasible for large circuits
   - equivalence checking cannot check for correct functionality under certain transactions, or properties such as safety, liveness, fairness
     - safety - denotes that a certain condition crucial for a proper functioning must not be violated to any time instance. *Something bad will never happen.*
     - liveness - comprises properties where a desired or necessary system condition will be reached. *Something good will eventually happen.* i.e., when told to perform task X, it will eventually do it.
     - fairness - used when certain properties must hold again and again. *Something good will happen infinitely often.*
   - Model checking components:
     - Modeling: a design view/representation that is acceptable to model checking, such as FSM (or variation of it), netlist, etc. that fits the implementation behavior
     - Specification: properties that the design model must satisfy.
       $\longmapsto$ issue 1: how should these properties be expressed? Need a suitable specification language that can model time
       $\longmapsto$ issue 2: is the set of properties complete?
     - Verification: verify properties in the design model. Need an efficient proof/verification algorithm - all sequences generated by implementation must satisfy the specification
   - Example properties:
     (a) At time step 3, the output of each z_i is 0
     (b) Does there exist a sequence of inputs such that z_1, z_2, z_3 are all 1's at the same time?
     (c) Does there exist a sequence of inputs such that z_1, z_2, z_3 becomes 0 consecutively?

(d) Does there exist a sequence of inputs such that z_1 becomes 1, and then sometime later, z_2 becomes 0?

(e) Does there exist a sequence of inputs such that z_1 becomes 1 and stays 1 forever?

(f) For every sequence of inputs α, there exists a sequence β such that on applying α followed by β, the output z_1 becomes 0

First 4 properties are SAFETY properties, the 5^{th} is liveness.

2. Model Checking Flow

3. Design Model: Kripke structure, a 4-tuple $(S,\ S_0,\ R \subseteq S \times S,\ L : S \to 2^{AP})$, where

- S is the set of states
- S_0 is the set of initial states (can be empty)
- R is the transition relations
- L is the labeling for each state $\in S$, and
- AP stands for atomic proposition.

Example 1

4. For combinational circuits, property checking can be done simply by existential quantifiers via ROBDD's, or by ATPG (verifying that the property is (or cannot be) satisfiable.

   $\rightarrow$ Example: combinational (first-order) property expressed as atomic propositions: outputs y_1, y_2, y_3 must hold the following: $(y_1 + \overline{y_2})y_3$

5. Temporal structure

   $\rightarrow$ A temporal structure $M = (S, R, L)$ consists of:

   - a finite set of states $S = \{s_0, s_1, ..., s_n\}$
   - a transition relation $R \subseteq S \times S$ with $\forall s \in S, \exists s' \in R$, such that $(s, s') \in R$
   - a labeling function $L : S \rightarrow V$. eg., $L(s_9) = 110100$

6. Branching Time

   $\rightarrow$ Given M and a starting state s_0, the temporal structure is traversed according to the successor states, resulting in an *infinitely* branching tree.

Example 2

7. Propositional Temporal Logic CTL* (Computational Tree Logic)

Syntax of CTL*

(a) Atomic formulas: $(q_i = 0)$ or $(q_i = 1)$ are CTL formulas.

(b) Boolean connectives: if f and g are CTL formulas, so are $(f \wedge g), \neg f, (f \vee g)$

(c) Temporal formulas:

- $G\phi$: (always) formula ϕ holds for all successor states on this path
- $F\phi$: (sometimes) formula ϕ must be true for at least one successor state on this path
- $X\phi$: (next) formula ϕ must be true for the immediate successor state on this path
- $\phi_1 U \phi_2$: (until) formula ϕ_1 must be true until formula ϕ_2 becomes true on this path
- $\phi_1 wU \phi_2$: (weak until) same as until, except that it is not required that formula ϕ_2 ever holds true on this path
- $\phi_1 B \phi_2$: (before) formula ϕ_1 must be true before formula ϕ_2 becomes true on this path
- A: for all paths
- E: there exists a path

Semantics of CTL*

- state formula: $s \models \phi$: reads "formula ϕ holds in state s"; if $\phi = (q_2 = 0)$, then $s_5 \models (q_2 = 0)$ holds if $q_2 = 0$ in state s_5. Similarly if ϕ was a temporal formula.
- path formula: $p \models \psi$: ψ holds in path p
- given a path formula f, then $E\ f$ and $A\ f$ are state formulas, but $\neg f$, $X\ f$, $F\ f$, $G\ f$, $f_1 \wedge f_2$, $f_1 \vee f_2$, $f_1\ U\ f_2$ are still path formulas
- $F\psi$: there exists a $k \geq 0$, such that $s^k, ... \models \psi$
- $G\psi$: for all $k \geq 0$, it holds that $s^k, ... \models \psi$

Some equivalences of formulas (ϕ is a state property, while ψ is a path property):

- $A\phi \leftrightarrow \neg E(\neg\phi)$
- $F\psi \leftrightarrow true\ U\psi$
- $G\psi \leftrightarrow \neg F \neg\psi$
- $AX\phi \leftrightarrow \neg EX(\neg\phi)$
- $AG\phi \leftrightarrow \neg EF(\neg\phi)$
- $AF\phi \leftrightarrow \neg EG(\neg\phi)$
- $EF\phi \leftrightarrow E(true\ U\psi)$

$\rightarrow$ Given a state s and machine M, a formula f is satisfiable if $s \models f$ holds.

8. Subclasses of CTL*

   - CTL: ($\subset CTL*$) temporal operators X, F, G, U, B must immediately be preceeded by path quantifiers A or E
     $\longmapsto$ EG, AF valid CTL formulas
     $\longmapsto$ FG, XF not valid CTL formulas
   - Linear Temporal Logic (LTL): ($\subset CTL*$) all formulas have the form $A\ \psi$ or $E\ \psi$ where ψ is a path formula
     $\longmapsto$ $A(EF\ p)$ not valid LTL formula

9. Example CTL properties

   - $EF(Start \wedge \neg Ready)$: is there a state where $Start$ holds but $Ready$ doesn't hold?
   - $AG(Req \rightarrow AF(Ack))$: If a request occurs, it will always be acknowledged

Example 3: illustration of 8 base operators of CTL

Example 4: CTL formulas example

10. Difference between LTL and CTL

- in LTL, each instance has exactly one next state, while in CTL, there may exist multiple next state possibilities
- Example property: if a is active and b is not active in the next state, then eventually a will become inactive
- In CTL, this property is ambiguous (essentially, do all next states must have b inactive?)
  (a) $AG(a \wedge AX\neg b) \rightarrow AX\ AF\neg a)$
  (b) $AG(a \wedge EX\neg b) \rightarrow AX\ AF\neg a)$
  (c) $AG(a \rightarrow AX(\neg b \rightarrow AF\neg a))$
- In LTL, this property is clear:
  - $AG(a \rightarrow X(\neg b \rightarrow F\neg a))$