VLSI DESIGN FLOW: RTL TO GDS

Lecture 18
Formal Verification - II

Sneh Saurabh
Electronics and Communications Engineering
IIIT Delhi

Lecture Plan

Formal Verification Techniques

Boolean Functions using Binary Decision Diagrams (BDDs)

Boolean Satisfiability (SAT) Problem Solvers

- Boolean Satisfiability Problem Solvers

Formal Verification

Satisfiability Problem Solver

Satisfiability : Problem Definition (1)

- Given $y = f(x_1, x_2, x_3, \dots, x_n)$ where the variables $\{x_1, x_2, x_3, \dots, x_n\}$ are Boolean variables
- Can y be evaluated to 1 by any assignment of variables $\{x_1, x_2, x_3, \dots, x_n\}$?
- If yes, then f is a satisfiable (SAT) instance, else it is an unsatisfiable (UNSAT) instance.

Given: $f(x_1, x_2, x_3) = x_1 x_2 + x_1 x_3 + x_2' x_3$. Is it a SAT instance?

- The function f can be evaluated to 1 by following assignments:
  - $x_1 = 0, x_2 = 0, x_3 = 1$
  - $x_1 = 1, x_2 = 0, x_3 = 1$
  - $x_1 = 1, x_2 = 1, x_3 = 0$
  - $x_1 = 1, x_2 = 1, x_3 = 1$
- Yes, it is a SAT instance

Given: $g(x_1, x_2, x_3) = (x_1 + x_2)(x_1' + x_2')(x_1' + x_2)(x_1 + x_3)(x_1 + x_3')$. Is it a SAT instance?

- Cannot be evaluated to 1 for any combination of values of $\{x_1, x_2, x_3\}$
- It is an UNSAT instance

Satisfiability: Problem Formulation

Inputs to SAT solvers are typically given in Conjunctive Normal Form (CNF)
- CNF is AND of clauses
- Clauses are OR of literals
- Literals are variable or its complement

$f(x_1, x_2, x_3) = (x_1 + x_2)\,(x_1' + x_2)(x_1 + x_3')$

- Variables: x_1, x_2, x_3
- Literals: x_1, x_2, x_1', x_3'
- Clauses: $(x_1 + x_2)$, $(x_1' + x_2)$ and $(x_1 + x_3')$

Why to use CNF in SAT solver?
- It reduces to 0 if any of the clauses is 0.
  - To make a function satisfiable, all its clauses must be made 1
- SAT Solvers can exploit this observation
  - Easily detect conflicts
  - Apply reasoning and reduce search space
- A given combinational logic circuit can be transformed into a CNF representation in linear time and space

Satisfiability : k-SAT problem

- We encounter various forms of SAT problems
- **k-SAT problem**: each clause in the CNF representation of a Boolean function is of maximum k literals

Examples

- 2-SAT Problem: $f(x_1,x_2,x_3) = (x_1+x_2)\,(x_1'+x_2)(x_1 + x_3\,')$
- 3-SAT Problem: $f(x_1,x_2,x_3) = (x_1+x_2+x_3)\,(x_1'+x_2)(x_1 + x_3\,')$
- 4-SAT Problem: $f(x_1,x_2,x_3,x_4) = (x_1+x_2+x_3)\,(x_1'+x_2)(x_1 + x_2 + x_3' + x_4\,')$

Complexity:

- 2-SAT Problem: can be solved in polynomial time
- 3-SAT Problem: NP-complete problem (No known algorithm exist that can solve in polynomial time for the worst case)
- k-SAT where k >3: NP-complete

Satisfiability Solver: Technique (1)

- For a function of n variables, there are 2^n possible variable assignments.
  - In the worst case, we need to try all of them (not feasible for practical cases)
- Perform a systematic search and pruning the search space

- **Davis–Putnam–Logemann–Loveland (DPLL) algorithm:**
  - Heuristically assigning a value 0/1 to an unassigned variable.
  - Deduces the consequences of the assignments or determines forced assignments

- **Unit Clause and Implications:** clause in which all but one literal takes a value 0 and the corresponding forced assignment of variable is called implication
- Assignment of variables leads to implications
  - Implication can further generate unit clauses
- **Example:** $f(x_1,x_2,x_3) = (x_1+x_2)\,(x_1'+x_3)(x_2'+x_3')$. Let us assign $x_1 = 1$.
  - $(x_1'+x_3)$ becomes unit clause. $x_3 = 1$ is an implication
  - $(x_2'+x_3')$ becomes unit clause. $x_2 = 0$ is an implication
- **Boolean Constraint Propagation (BCP):** deduce implications iteratively until possible

Satisfiability Solver: Technique (2)

Conflict and Backtracking:

- Variable assignments (and associated implications), can make all literals in a clause evaluate to 0.
  - This scenario is known as a conflict
  - Requires to backtrack some earlier decisions by flipping the variable assignment

- **Example:** $f(x_1,x_2,x_3) = (x_1+x_2)\,(x_1'+x_3)(x_1'+x_3')$. Let us assign $x_1 = 1$.
  - $(x_1'+x_3)$ becomes unit clause. $x_3 = 1$ is an implication
  - All literals in $(x_1'+x_3')$ becomes 0.
  - Backtrack $x_1 = 0$ and proceed.

- **When is a function satisfiable:** If no conflict is encountered, the solver goes on assigning variables until all variables get assigned.

- **When is a function unsatisfiable:** If we obtain a conflict and no more backtracking is possible, the function is unsatisfiable.

Satisfiability Solver: Algorithm (1)

Input: Given function f in CNF
Output: return SAT if satisfiable and UNSAT if not satisfiable

```
decision_level ← 0
while (DECIDE( f, decision_level) != ALL_ASSIGNED) do
    if (DEDUCE( f, decision_level) = CONFLICT) then
        backtrack_level ← DIAGNOSE( f, decision_level)
        if (backtrack_level = NOT_POSSIBLE) then
            return UNSAT
        else
            BACKTRACK( f, decision_level, backtrack_level)
            decision_level ← backtrack_level
        end if
    else
        decision_level ← decision_level + 1
    end if
end while
return SAT
```


- S. Saurabh, "Introduction to VLSI Design Flow". Cambridge: *Cambridge University Press*, 2023.

Satisfiability Solver: Algorithm (2)

Improvements:

- Preprocessing to simplify the SAT problem,
- Employing efficient data structure for BCP
- Intelligent pruning of search spaces and random restarts
- Multicore processing

References

- S. Saurabh, "Introduction to VLSI Design Flow". Cambridge: *Cambridge University Press*, 2023.
- S. Malik and L. Zhang. "Boolean satisfiability from theoretical hardness to practical success.", *Communications of the ACM 52* (Aug. 2009), pp. 76–82.
- S. A. Cook. "The complexity of theorem-proving procedures." *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, (New York, NY, USA) (1971), pp. 151–158, ACM.