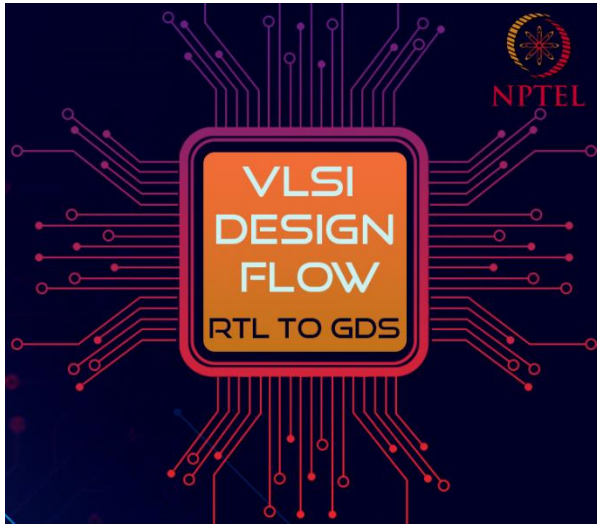
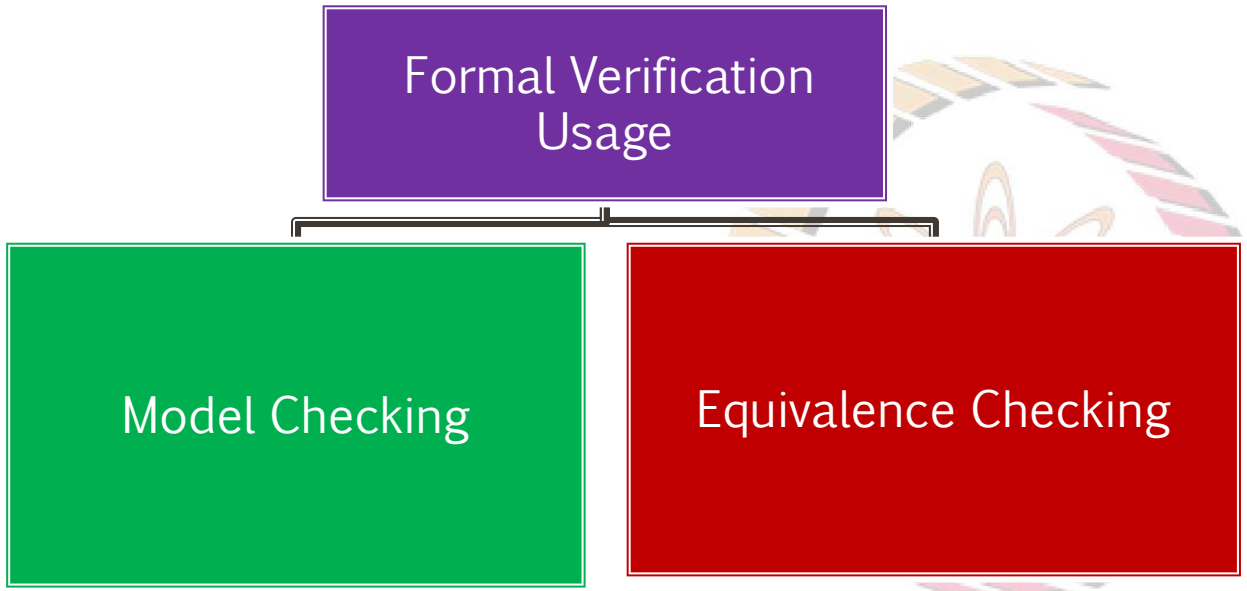
VLSI DESIGN FLOW: RTL TO GDS

Lecture 19
Formal Verification- III



Sneh Saurabh
Electronics and Communications Engineering
IIIT Delhi

Lecture Plan



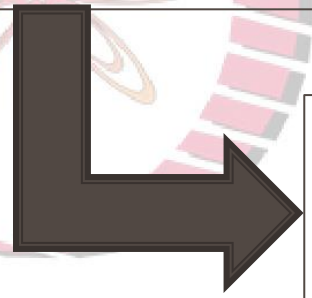
Model Checking

- Verifies that for a given model (design), whether the specifications or given set of properties are satisfied

Equivalence Checking

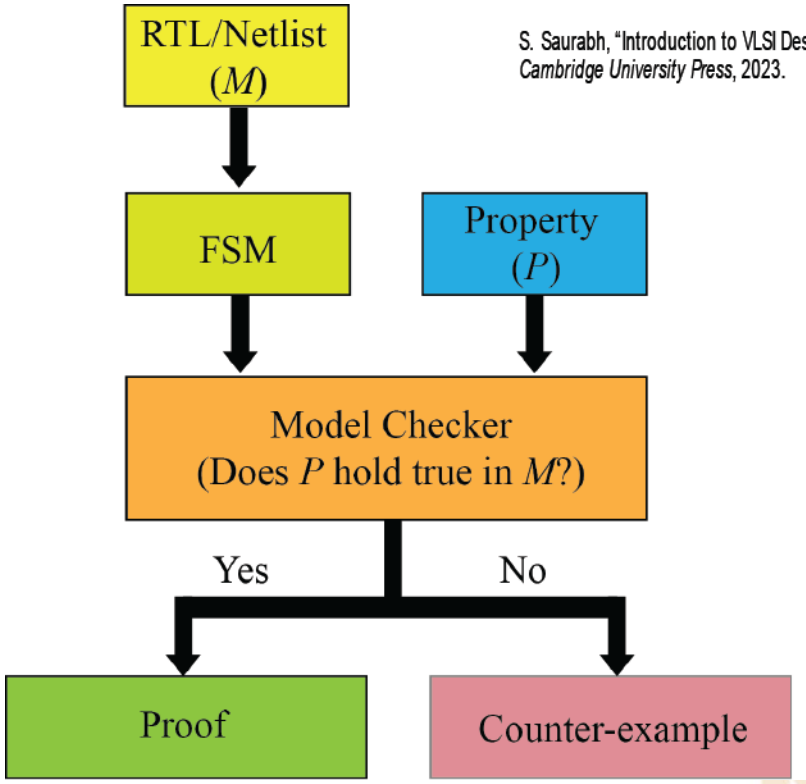
- Verifies that the two representations of the same design will exhibit exactly the same behavior

Formal Verification



Model Checking

Model Checking: Framework



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

Inputs

- Given design in RTL or Netlist
  - Typically modelled as an FSM
- Set of properties that a model checker needs to verify

Challenges

- Model checker needs to verify that the given property is valid as the FSM evolves through its states
- For N state elements: number of states can be 2^N
- This problem is known as the **state explosion problem**

Model Checking: Property Specification

- Properties are specified in Temporal Logic

Temporal Logic

- System properties need time-related specification, in addition to logic related expressions

Examples of temporal properties

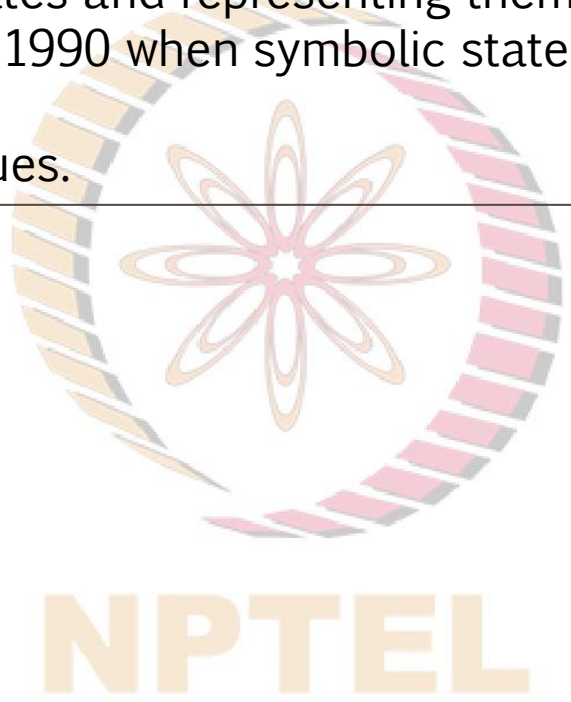
- *Whenever* a correct password is entered, the door *eventually* opens
- For traffic lights at a given post, one of the red, yellow or green light is *always* ON
- *Whenever* a request is made by multiple requesters, it is *never* granted to more than one requesters simultaneously

- Ordering of events is implicit
  - eventually, never, always, whenever, etc.
- Modelled using System Verilog Assertion (SVA)
  - ➢ Embed SVA in RTL
  - ➢ Checked by both model checker and simulators
  - ➢ Can specify assumptions and constraints (to make property checking easier for the tool)

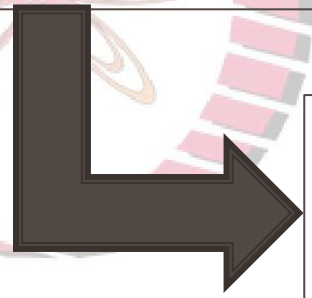
Model Checking: Techniques

Primary Difficulty:

- Exhaustive search becomes difficult due to the state explosion problem
  - Explicitly enumerating states and representing them as graphs could not scale
- A breakthrough came around 1990 when symbolic state-space exploration was proposed :
  - First it employed BDDs
  - Later SAT-based techniques.



Formal Verification



BDD-based Model Checking

BDD-based Model Checking: Characteristics Function

Characteristic Function for a set:

- Consider an FSM with a finite set of states Q.
- Further, consider a subset of states $A \subset Q$
  - A can be represented by a Boolean function f such that for any state $x \in Q$, $f(x) = 1$ if and only if $x \in A$

Example:

- Consider an FSM with five states $Q = \{s_0, s_1, s_2, s_3, s_4\}$.
- Let the states be represented by 3 bits $\{x_2 x_1 x_0\}$. We refer to these bits as **state bits**.
- We can encode these states $\{s_0, s_1, s_2, s_3, s_4\}$ as $\{000,001,010,011,100\}$.
- In this representation we can represent the subset of states: $A = \{s_0, s_2, s_4\}$ as a Boolean function: $f(x_2, x_1, x_0) = x_2'x_1'x_0' + x_2'x_1x_0' + x_2x_1'x_0'$

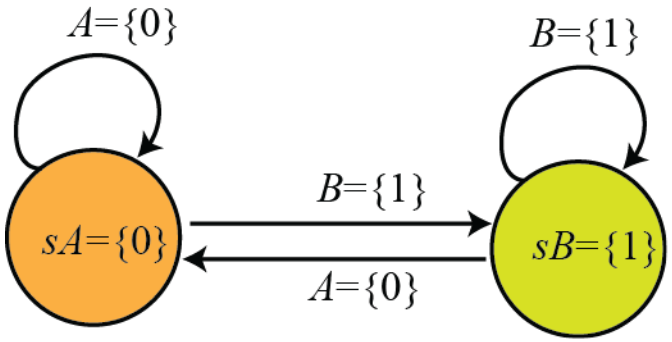
- We can represent a large set using its characteristics function with the help of compact BDDs
- We can also compute the transition from a set of states to another set of states very efficiently using BDDs

BDD-based Model Checking: Transition Relation

- Consider an FSM with set of states Q.
- Let us denote the set of input values as I.
- Let us denote the next-state function as $x' = \delta(x, i)$ for $x \in Q$ and $i \in I$.

Transition relation for an FSM:

- The transition can be defined using a transition relation $T(x, i, x')$ such that $T(x, i, x') = 1$ if and only if $\delta(x, i) = x'$.



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

x	i	x'	$T(x,i,x')$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

- We can represent a transition relation $T(x, i, x')$ compactly using BDD
- Subsequently, we use the transition relation in BDD-based model checking

BDD-based Model Checking: Image/Preimage

Image of a set of states:

- Image for a given set of states S is the set of states S' that we can reach in **one step** from S.
- We denote the image computation as $S' = Image(S, T)$ [T is the transition relation].

Preimage of a set of states:

- Preimage of a set of states S' is a set of states S from which we can reach S' in **one step**.
- We denote preimage computation as $S = Preimage(S', T)$.

- For a given set of states, we can compute image and preimage very efficiently using BDDs.
- BDD-based model checking relies on this computation

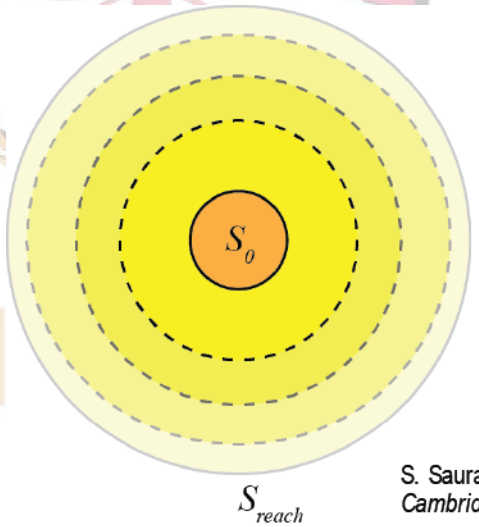
BDD-based Model Checking: Computing Reachable States

Input:
- Given starting set of states S_0
- Transition relation $T(x, i, x')$

Output:
- Returns reachable set of states S_{reach}

- The set of reachable states S_{reach} until no more new states are discovered
  - ➢ Algorithm is said to have attained a fixed point.

1: $S_{reach} \leftarrow S_0$
2: $S_{new} \leftarrow S_0$
3: $k = 0$
4: while ($S_{new} \neq \{\}$) do
5: $k \leftarrow k + 1$
6: $S_k \leftarrow Image(S_{new}, T)$
7: $S_{new} \leftarrow S_k - S_{reach}$
8: $S_{reach} \leftarrow S_{reach} \cup S_{new}$
9: end while
10: return S_{reach}



- States represented compactly as characteristic functions using BDDs
- Canonicity of BDDs eases manipulation
- Pre-image computation: set of all states from which S_0 can be reached.

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

BDD-based Model Checking: Technique

Example: Suppose it is required to check whether a state satisfying a Boolean function P is reachable from a given initial state s_0.

- Let both the states and the Boolean function P be represented in terms of state bits.

- A model checker considers a set of states S_P for which P holds.
- Let us represent the set S_P using the characteristic function CF_{SP}.
- But, $CF_{SP} = P$ (i.e., the characteristic function of the set of states for which P holds is nothing but P)
  - Consider a state x for which P holds. Therefore, $P(x) = 1$ and x should belong to S_P. Hence, $CF_{SP}(x) = 1$
  - Consider a state y for which $CF_{SP}(y) = 1$. Therefore, y belongs to S_P and P should hold for it. Hence, $P(y) = 1$.
  - Thus, $CF_{SP} = P$

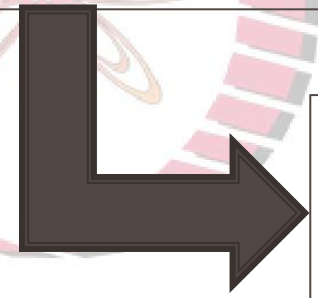
- Using preimage computation, we can determine the set of all states $S_{reach}{}'$ from which S_P can be reached.
- If $S_{reach}{}'$ includes the initial state s_0, given property holds.
- If $S_{reach}{}'$ does not include the initial state s_0, the given property does not hold

BDD-based Model Checking: Limitations

- In the worst case, the size of BDD can be exponential in the number of inputs.
  - A BDD-based representation of transition relation can blow up with an increase in the number of state bits.
  - Different variable orders can be tried

- Manual interventions, such as adding constraints, can help.
  - Simplify the problem for the model checker and make it solvable.

Formal Verification



SAT-based Model Checking

SAT-based Model Checking: Technique

Approach:

- Obtain a counterexample of a finite length n (n is the number of clock cycles from the initial state)
- We derive a Boolean function ϕ_n using the given circuit and the given property such that: the function ϕ_n is satisfiable if and only if a counterexample of length exists.
- This type of model checking is known as **Bounded Model Checking (BMC)**
  - Typically, we carry out BMC iteratively by incrementing n.
  - It continues until we have found a counterexample or the problem becomes too complicated to be handled by the SAT solver.

Mechanism:

- To derive ϕ_n, we unfold the behavior of the system one cycle at a time using the next-state function until it reaches nth clock cycle.
- The Boolean function ϕ_n is the logical conjunction (ANDs) of clauses obtained from:
  - Given initial state.
  - The system behavior obtained from the next-state function.
  - A Boolean expression that evaluates to 1 for a counterexample (derived from the given property).

SAT-based Model Checking: Merits/Demerits

Merits:

- Avoids the problem of memory blow-up in representing transition relations in the BDD-based model checking
  - ➢ Next-state function grows linearly as the BMC traverses the next state in each cycle.
  - ➢ But can take longer time by SAT solver because introduction of new variables
- Exploits power of SAT solver

Demerits:

- Lacks completeness.
  - ➢ In practice, can quickly find bugs (if it exists)

References

- J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L.-J. Hwang. "Symbolic model checking: 10^{20} states and beyond." *Information and Computation* 98 (1992), no. 2, pp. 142–170.
- J. Herve, S. Hamid, L. Bill, K. B. Robert, and S.-V. Alberto. "Implicit state enumeration of finite state machines using BDD's." *Computer-aided Design*, 1990ICCAD-90. Digest of Technical Papers. *1990 IEEE International Conference on* (1990), pp. 130–133.
- S. Saurabh, "Introduction to VLSI Design Flow". Cambridge: *Cambridge University Press*, 2023.