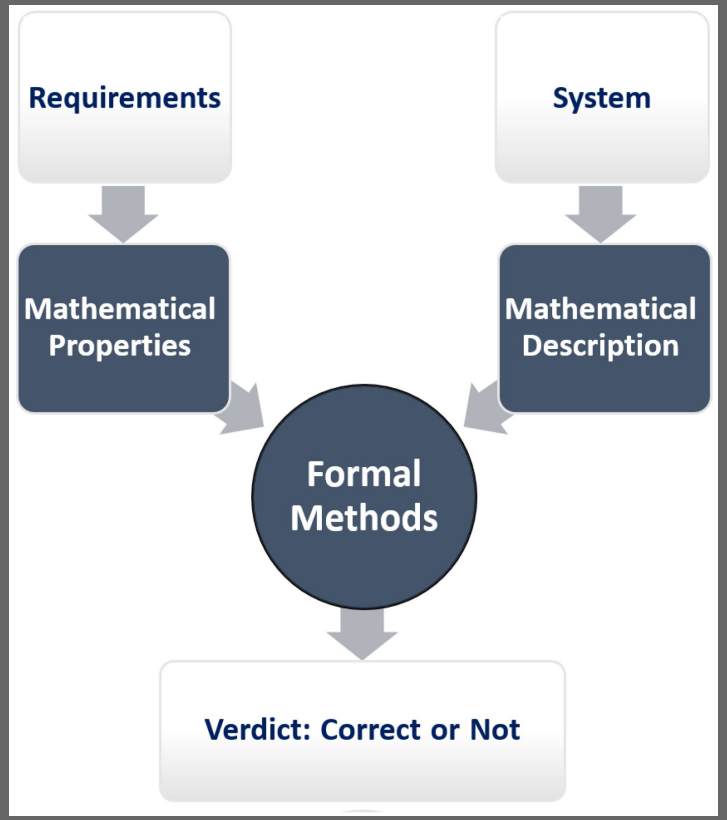
CPE 470 - Formal Verification



Exhaustive Testing

Glossary

State Space: Set of all possible states

- For small designs, it is possible to test all possible inputs
  - Cover the whole **state space**
  - Example: 8 bit Counter
    - Exhaustively tested in several hundred cases

- As designs scale up, get exponentially harder to test
  - State Space Explosion
- Example: 64 bit counter
  - 2^64 = 18446744073709551616 states
  - 1 test every nanosecond → Would take hundreds of years

- Cannot test all possible states
  - Could test a random subset, but then could miss bugs
  - How can we verify functionality on sufficiently complex systems?

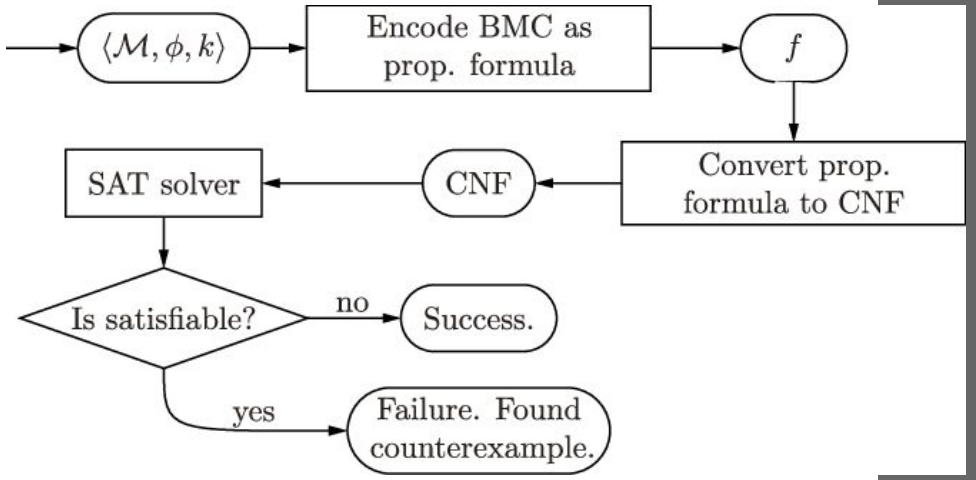
Formal Verification

Glossary
SAT: Boolean Satisfiability Problem

- How can we prove that our design works in all possible cases?
  - Formal Verification: use mathematical proofs to prove correctness in every case

- **SymbiYosys**: open source formal verification framework
  - Part of YosysHQ, greater yosys CAD suite

- Model Design as Boolean Expression
- Use a **SAT** solver to find all possible failures
  - Same mechanism used in math proofs
- If no failures are found, design is correct



Bounded Model Checking

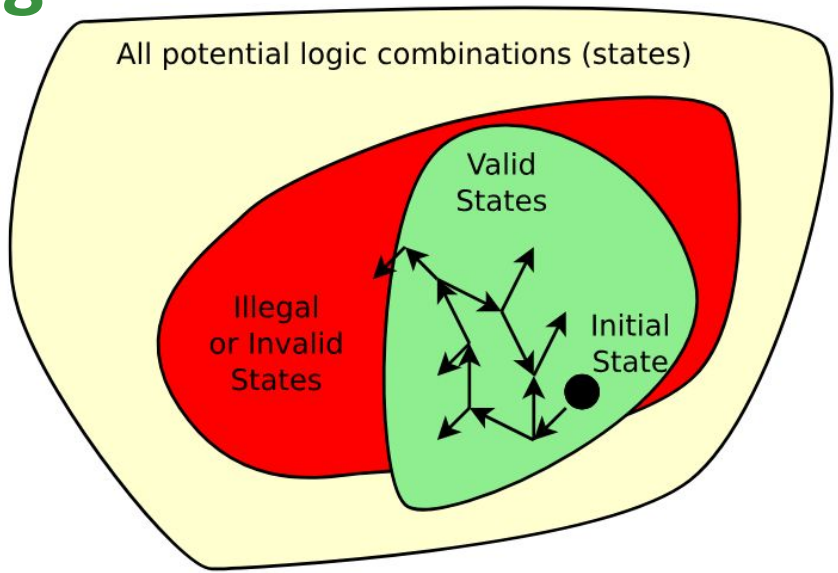
Glossary

BMC: Bounded Model Checking

BMC provides a way to ensure design correctness for N cycles

- N is the Bound: duration to cover
  - Higher Bound → More Expensive to Run
  - Lower Bound → Faster, Less Guarantees

Guarantees correctness for all possible inputs, but only for N consecutive cycles



Operator	Formal Verification
restrict ()	Restricts search space
assume()	
assert()	Illegal state

For bounded model checking,

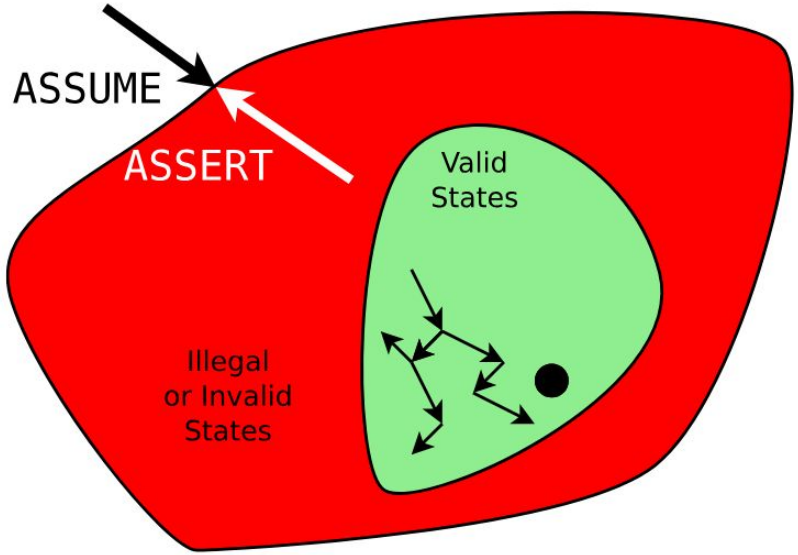
1. Start at the initial state
2. Examine *all* possible states for N clocks
3. Try to find a way to make an **assert** (); fail
4. If it's not possible in N clocks, then *pass*

Proof by Induction

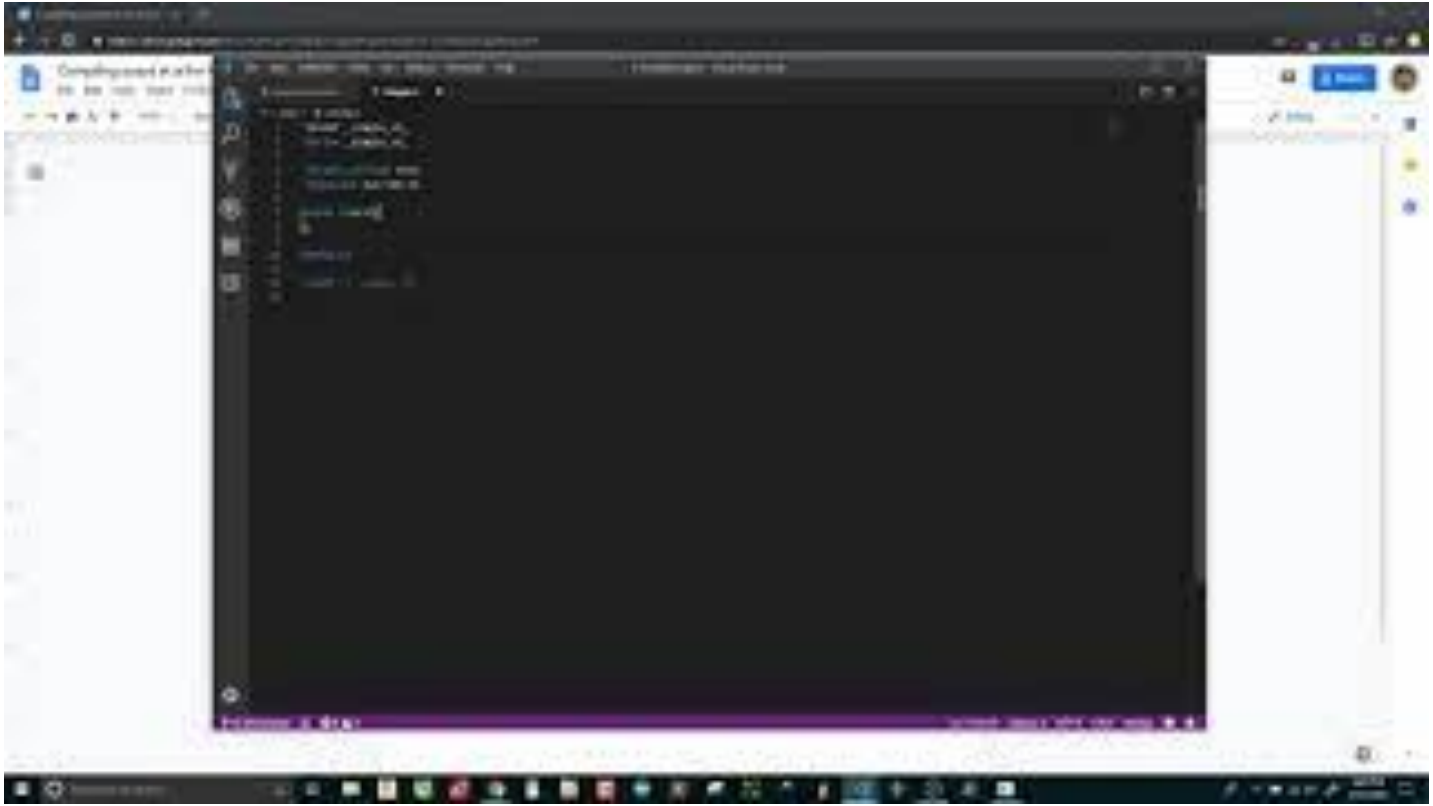
- BMC gave us guarantee correctness for set amount of cycles
- How can we prove correctness across indefinite amount of time?

→ Use Induction

- Start with a Base Case
  - Base Case is just BMC: prove it works within k bounds
- Prove that for every k^{th} state, k+1 is still a valid state



Formal Example



References

- https://zipcpu.com/tutorial/class-verilog.pdf