Reachability Analysis for Sequential Circuits

Supratik Chakraborty

IIT Bombay

State of a System

- System state
  - Information about system sufficient to determine future behaviour
  - Values of registers, controller flip-flops, memories, …
- Modern digital designs
  - Finite but extremely large state spaces
    - n two-state memory elements: 2^n states
    - k interacting components, with n_1, … n_k states
      - $n_1 \times n_2 \times \ldots \times n_k$ states
  - # states grows exponentially with # components
  - 10000 flip-flips: approx 10^{3000} states !!!
    - A mind-boggling number from not-so-large design

Why Do We Care About Reachability?

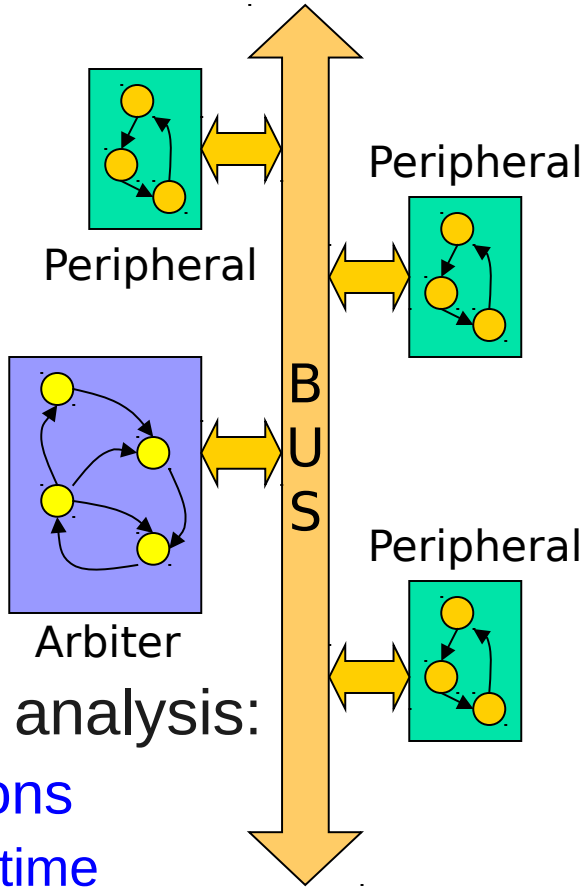
- State space exploration
  - Starting from given state, find all reachable states
    - Also known as **Reachability Analysis**
  - Crucial for several activities:
    - Synthesis: Optimize from knowledge of unreachable states
      - Unreachable states are external don't cares
    - Verification: Are “bad” states reachable?
    - Analysis: Improve accuracy from knowledge of reachable states

State space explosion: single largest hurdle

Techniques to battle explosion crucial

Illustration of Applicability

- PCI Bus
  - 3 peripherals, 1 arbiter
  - Peripheral: Application controller + Bus interface controller
  - Arbiter: Fixed arbitration scheme
  - Bus: Negligible delays
- Global behaviour
  - Composition of component FSMs



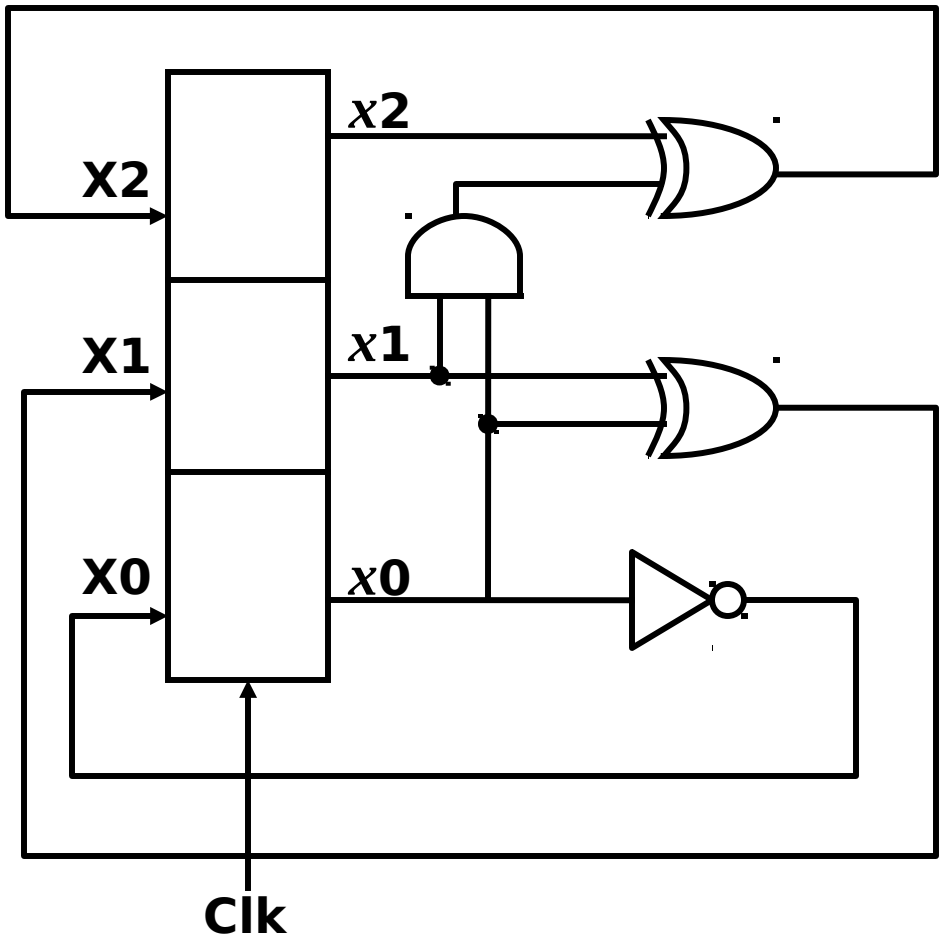
- Properties verifiable using reachability analysis:

  For arbitrary sequences of bus transactions
  - Only one peripheral can be master at any time
  - Peripheral requesting to be master becomes one in < 3 cycles **...**

Outline

- Basics of reachability analysis
- Explicit enumeration & symbolic approaches

Example: Sequential Circuit



Model

State transition graph defined by

X0 = NOT($x0$)

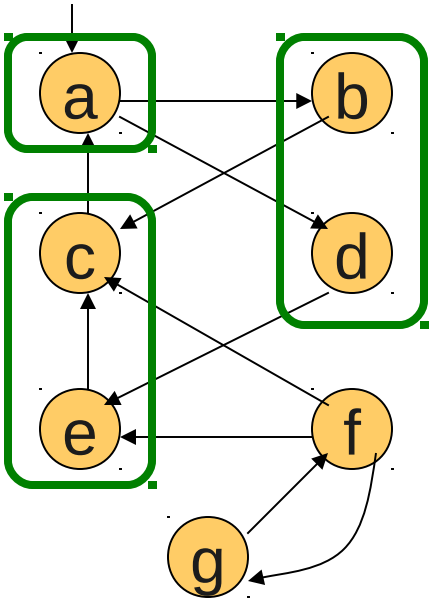
X1 = XOR($x1$, $x0$)

X2 = XOR($x2$, $x0$. $x1$)

Property to check

State $x0$, $x1$, $x2$ = 111 is reached starting from state 000

Basic Reachability Analysis



Reachable = {a} : Initial

Reachable = {a, b, d} : upto 1 step

Reachable = {a, b, d, c, e} : upto 2 steps

Reachable = {a, b, d, c, e} : upto 3 steps

⋮

Reachable = {a, b, d, c, e} : upto n steps

States unreachable from a: {f, g}

Forward Reachability Algorithm

Given: State transition system T, Initial states S

Find: All states reachable from initial states

- Reachable := InitialStates;
- LastReachable := EmptySet;
- While (Reachable ≠ LastReachable)
  - Img := {s | In T, s reached from some s' ∈ Reachable in 1 step};
    /* Also called Image of Reachable set under T */
  - LastReachable := Reachable;
  - Reachable := Reachable U Img;

Reachability as Fix-point Computation

Given

- S_0 : set of states
- T : state transition system

Let

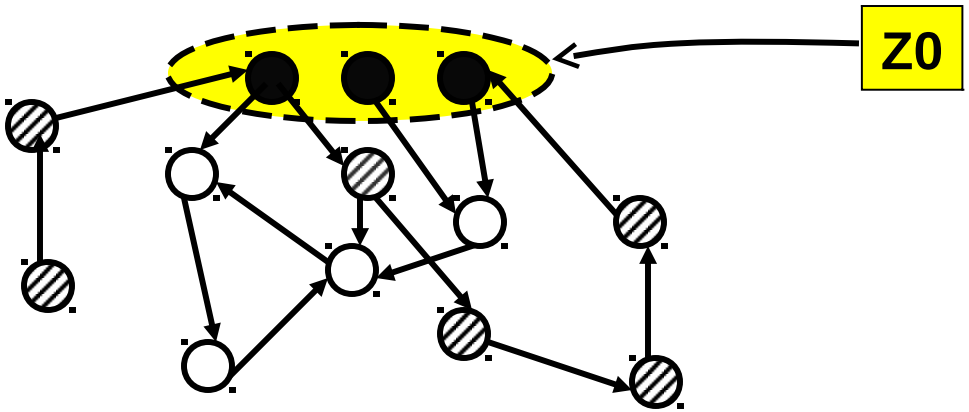
- $S_{i+1} = F(S_i) = S_0 \cup Image(S_i, T)$
- Reachable states from $S_0 = F^r(\phi)$,

  where $F^r(\phi) = F^{r+1}(\phi)$

Least fix-point of F

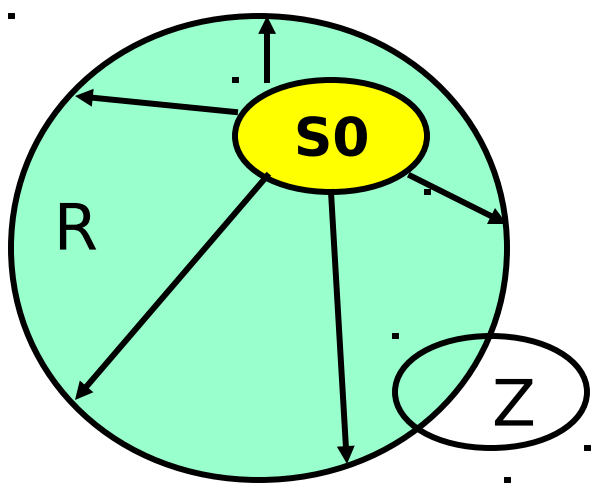
Backward Reachability

- Give a set Z0 of states
  - Compute set of states from which some state in Z0 can be reached.
  - Analogous to forward reachability with minor modifications

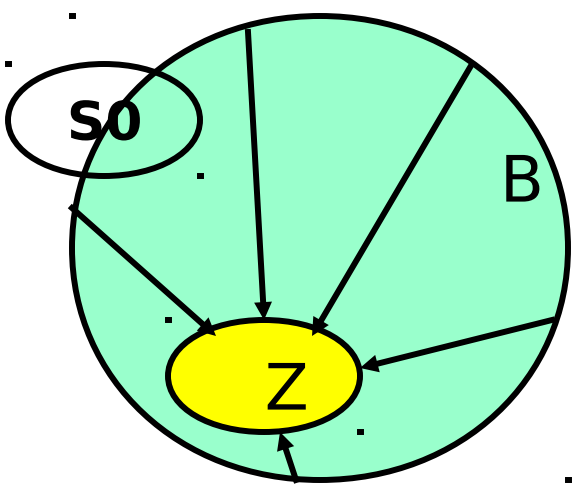


Checking Reachability

- Z = set of “bad” states, S0 = set of “initial” states
- 2 ways of checking if a state in Z is reachable from S0



Forward Reachability



Backward Reachability

Issues in Reachability Analysis

- Representing sets of states and transitions
  - Can get very large !
- Computing image, union, set operations
  - Can be tricky for large sets of states
- Checking whether two sets of states are equal or non-intersecting
  - Decision procedures needed

Explicit enumeration techniques:

- Represent and manipulate sets of states explicitly

Symbolic reachability analysis

- Symbolic representation and manipulation of state sets

Outline

- Basics of Reachability Analysis
- Explicit enumeration & symbolic approaches

Explicit Enumeration Approaches

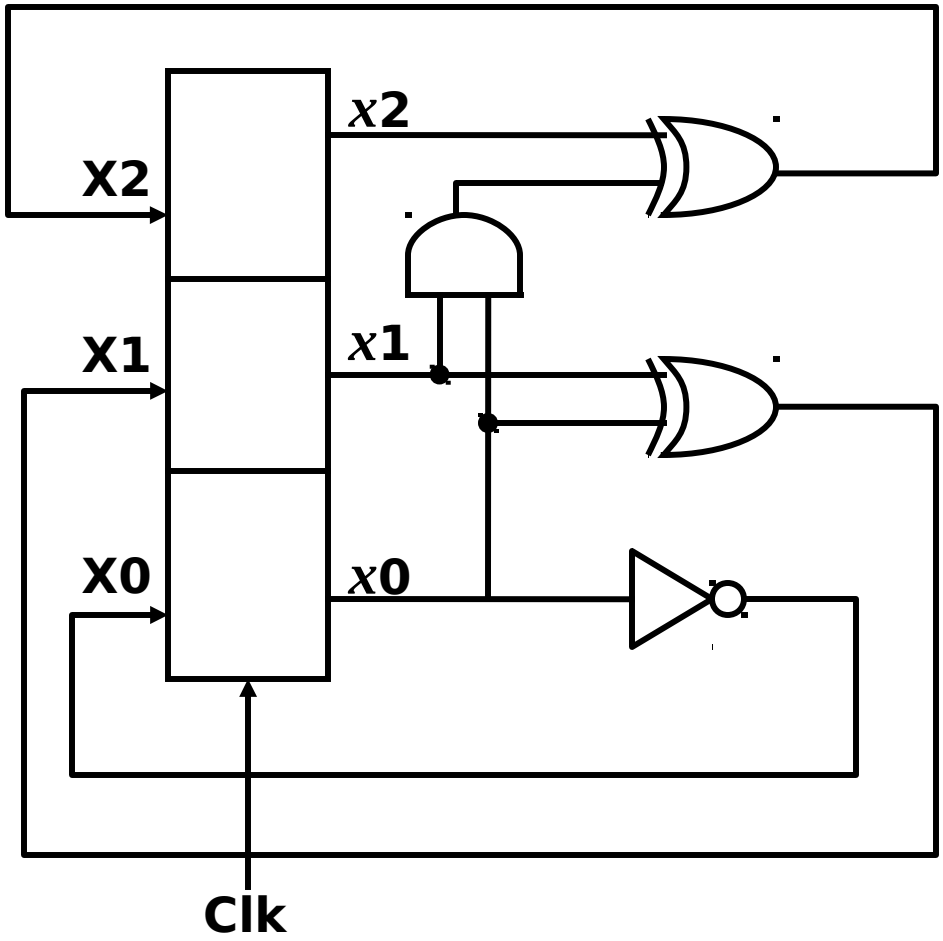
- Early reachability analyzers
- Some modern analyzers also use this approach
  - SPIN, Murφ, SMC, EMC …

- Explore image of each state starting from initial state
- As new states encountered, store in table
  - Interesting aside: “stateless search” (Verisoft)
- If a state already in table, it is not explored again
- Storage for each state: a few bytes in practice
- Can store approx 10^9 states on modern machines
  - Use sophisticated techniques to store selected states

Storing States in Explicit Approaches

- State hashing:
  - Table stores hash values of states
  - On encountering a state, hash and check if in table
  - Incompleteness: Two states may hash to same value
    Only one of them explored
  - Subset of reachable states explored
  - Every state explored is reachable, but not every reachable state may be explored
  - Very sophisticated state hashing schemes exist
- Other state table compaction schemes:
  - Partition states into equivalence classes
  - Store at most 1 representative from each class

Symbolic Reachability Analysis



Recall 3-bit counter

X0 = NOT($x0$)

X1 = XOR($x1$, $x0$)

X2 = XOR($x2$, $x0$. $x1$)

Symbolic States and Transitions

- Encode states using Boolean variables
  - 3-bit counter: x0, x1, x2: 000, 001, … 111
- Encode sets of states using Boolean predicates
  - {000, 010, 011, 001} represented by

    $S(x0, x1, x2) = S(\mathbf{x}) = x0.$
- Encode state transitions using Boolean predicates

  $N(x0, x1, x2, X0', X1', X2') = N(\mathbf{x}, \mathbf{X'}) =$

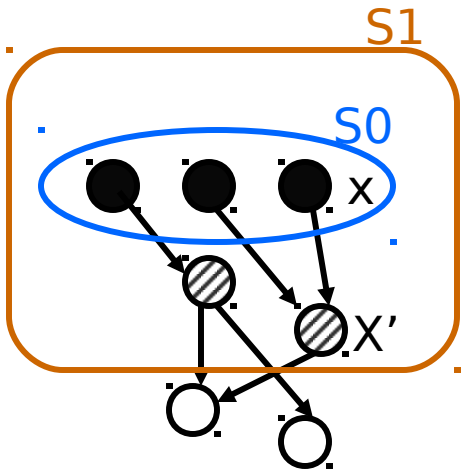
  $(X0' \Leftrightarrow \neg\, x0) \wedge (X1' \Leftrightarrow x1 \oplus x0)) \wedge (X2' \Leftrightarrow x2 \oplus (x1 \wedge x0))$
- Check reachability by manipulation of Boolean expressions
  - **States NEVER explicitly represented**

Symbolic Image Computation

Given set S_0 of states, can we reach a state in set Z_0?

- N(**x**, **X'**) : Transition relation predicate
- States reachable in at most 1 step:

  $S_1 = S_0 \cup \{ \mathbf{X'} \mid \exists\, \mathbf{x} \text{ in } S_0 \text{ and } N(\mathbf{x}, \mathbf{X'}) = \text{true}\}$



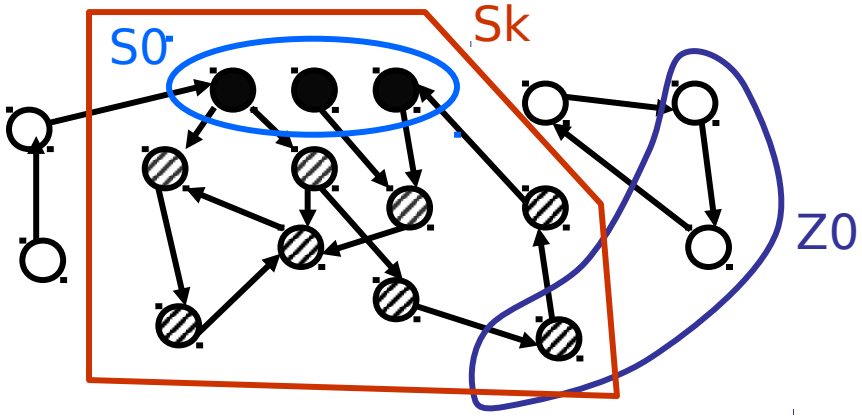
  - Expressed as Boolean predicates:

$$S_1(X0', X1', X2') = S_0(X0', X1', X2') \vee \exists\, x0\, \exists\, x1\, \exists\, x2\ (S_0(x0, x1, x2) \wedge N(x0,x1,x2,X0',X1,'X2'))$$

  - ***Given predicates S_0 and N, S_1 can be obtained***

Symbolic Forward Reachability

- Compute S_1 from S_0, S_2 from S_1, …
  - $S_{i+1} = F(S_i)$
- Continue until $S_{k+1} = F(S_k) = S_k$
  - Least fix-point of F
  - S_k = Set of all states reachable from S_0
    - **Computed as a Boolean predicate**



  - **Check if $S_k \wedge Z_0$ is a satisfiable predicate**

Symbolic Backward Reachability

- Give a set Z_0 of states
  - Compute states from which some state in Z0 is reachable
  - $Z_{i+1}(\mathbf{x}) = F(Z_i(\mathbf{x})) = Z_0(\mathbf{x}) \vee \exists\, \mathbf{X'}\, (N(\mathbf{x}, \mathbf{X'}) \wedge Z_i(\mathbf{X'}))$
    - Desired set: least fixed point

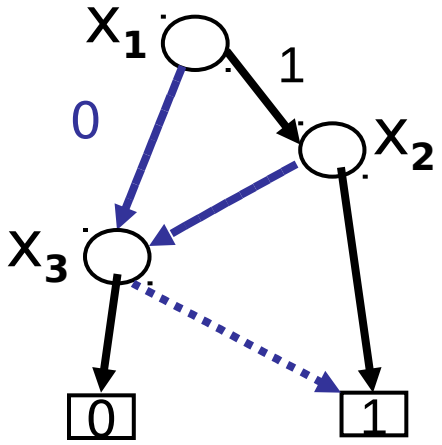
Fairly mature symbolic reachability analyzers exist:
NuSMV, VIS, FormalCheck, Bingo, …

Symbolic Reachability: Issues

- Need good representation of Boolean functions
  - Canonicity
  - Compactness
  - Efficient application of $\wedge$, $\vee$, $\neg$, $\forall$, $\exists$
- Efficient decision procedures for propositional logic
- Compact representations of Boolean functions can allow compact representations of large sets of states
- Two mainstream approaches
  - Reduced Ordered Binary Decision Diagrams (ROBDD)
  - Bounded reachability analysis using SAT solvers

Binary Decision Diagrams

- DAG representation of Boolean functions
- Example: $f = (x_1 \wedge x_2) \vee \neg x_3$
  - Evaluating f:
    - Start from root
    - For each vertex xi
      - blue branch if xi = 0
      - else black branch
- Ordering of variables
  - In all paths, node labels in specified order
- Reduced graphs
  - No two vertices represent same function



REDUCED ORDERED BDD (ROBDD)

Operations on BDDs

- Given ROBDDs for f1 and f2, algorithms exist for computing ROBDD for f1 *op* f2 …

$$op \in \{\wedge, \vee, \neg, \Leftrightarrow\}$$

- Complexity polynomial in BDD sizes
  - If size can be kept under control, we are in business!
  - Works well for circuits with upto a few 100 flip-flops
  - BDD size limiting factor in larger applications
- Quantification:
  - $\exists$ x1. f(x1, x2, x3) = f(0, x2, x3) $\wedge$ f(1, x2, x3)
  - $\forall$x1. f(x1, x2, x3) = f(0, x2, x3) $\vee$ f(1, x2, x3)