Symbolic Simulation

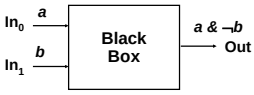
and its Connection to Formal Verification

Randal E. Bryant

Carnegie Mellon University

`http://www.cs.cmu.edu/~bryant`

Symbolic Simulation



Idea

- Encode set of values symbolically
- Evaluate system operation over these values

Effect

- In single run, compute information that would otherwise require multiple simulation runs
- If do it right, can even be used for formal verification

Advantages of Symbolic Simulation

- Relative to better known formal verification techniques
  - symbolic model checking

Modeling Capabilities

- Can use wide variety of circuit models
  - Including ones requiring event scheduling

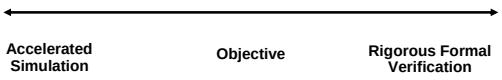
Efficiency

- Hybrid between symbolic and conventional simulation
  - Reduce coverage to make tractable
- Exploit abstraction capabilities of X
  - Form of abstract interpretation

Categorization #1

Verification Objective

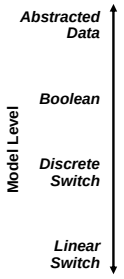
- Accelerated Simulation
  - Get more simulation done in less time
- Rigorous, formal verification
  - Don't trust anything that hasn't been proven



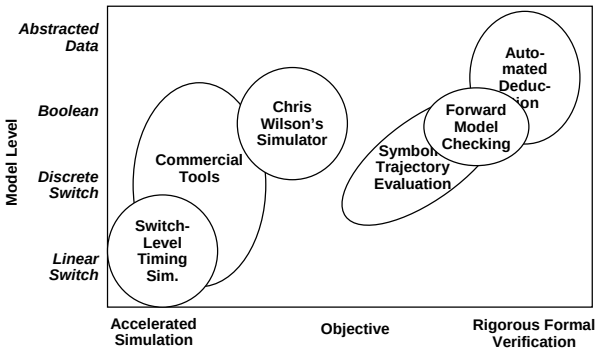
Categorization #2

Modeling Level

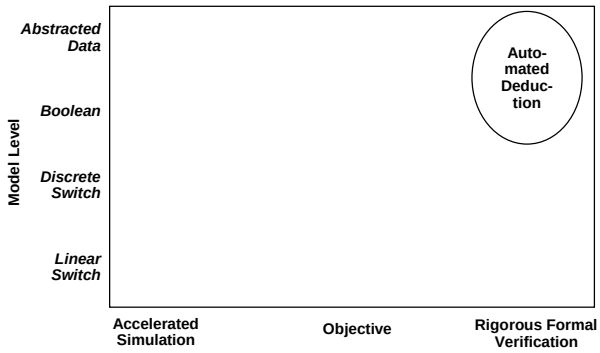
- Abstract away as much as possible
  - Especially data values & operations
- Boolean gate / RTL
  - Focus of 99% of verification research
- Transistor
  - Challenge to have tractable but accurate model



Symbolic Simulation Landscape



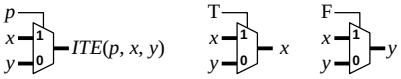
Automated Deduction



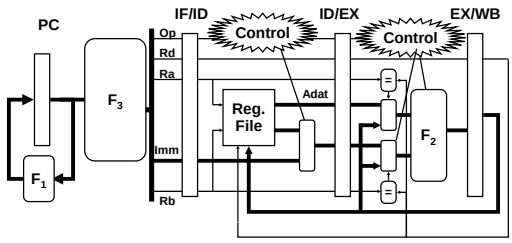
Abstracting Data

View Data as Symbolic "Terms"

- No particular properties or operations
  - Except for *equations*: $x = y$
- Can store in memories & registers
- Can select with multiplexors
  - ITE: If-Then-Else operation



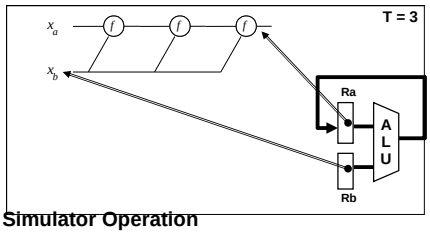
Abstraction Via Uninterpreted Functions



For any Block that Transforms or Evaluates Data:

- Replace with generic, unspecified function
- Also view instruction memory as function

Term-Level Symbolic Simulation



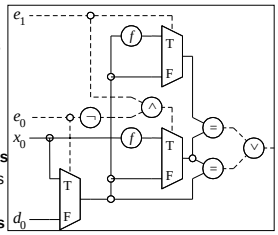
Simulator Operation

- Register states are term-level expressions
  - Denoted by pointers to nodes in Directed Acyclic Graph (DAG)
- Simulate each cycle of circuit by adding new nodes to DAG
  - Based on circuit operations
- Construct DAG denoting correctness condition

Resulting Decision Problem

Logical Formula

- Integer Values
  - *Solid lines*
  - Uninterpreted functions
    - » Integer variables
  - If-Then-Else operation
- Boolean Values
  - *Dashed Lines*
  - Uninterpreted predicates
    - » Propositional variables
  - Logical connectives
  - Equations & inequalities



Task

- Determine whether formula is universally valid
  - True for all interpretations of variables and function symbols

Deduction-Based Verification

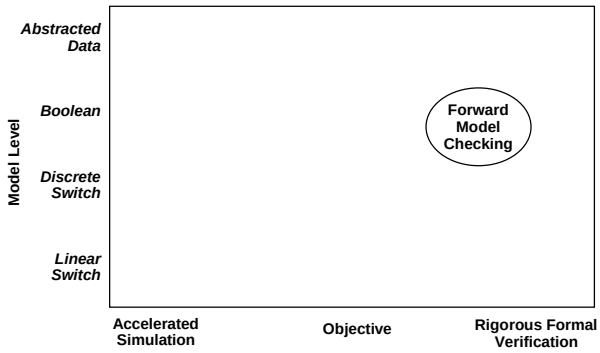
Automatic Theorem Provers

- Some of the earliest work in formal hardware verification
  - Gordon '83, Hunt '85, …
- Heavy focus on rigor
- Strong abstraction capabilities
  - Can selectively apply different levels of abstraction

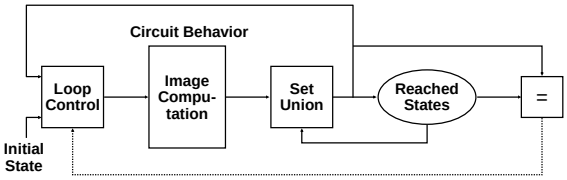
Increasing Degree of Automation

- Burch & Dill, CAV '94
  - Implement & tune decision procedure to match modeling needs
  - Automate generation of simulation relation
    - » For pipelined microprocessors
- Active research area
  - But, not focus of this talk

Forward Model Checking



Forward Reachability

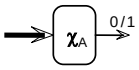


- Determine set of all reachable states of circuit
- Key step in model checking
  - Many (but not all) properties can be checked by some form of reachability computation

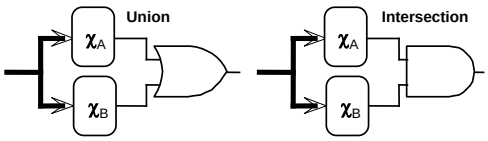
Characteristic Function Representation of Set

Concept

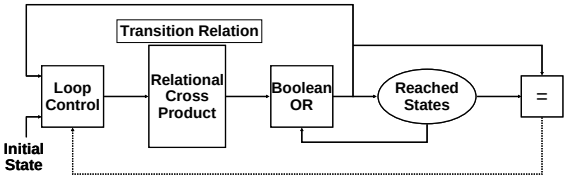
- $A \subseteq \{0,1\}^n$
  - Set of bit vectors of length n
- Represent set A as Boolean function χ_A of n variables
  - $X \in A$ if and only if $\chi_A(X) = 1$



Set Operations



Forward Reachability via Characteristic Functions

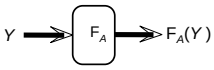


- Model system behavior as transition relation
- $\delta(s,s')$ = 1 when possible to change from state s to state s′ in one step
- Powerful, but expensive approach

Parametric Representation of Set

Concept

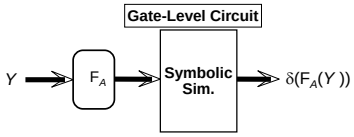
- $A \subseteq \{0,1\}^n$
  - Set of bit vectors of length n
  - Must be nonempty
- Represent set A as set of n Boolean functions F_A
  - Set indicated by function images
  - $X \in A$ if and only if for some Y, $F_A(Y) = X$
- Not unique
- Various algorithms to generate



Set Operations

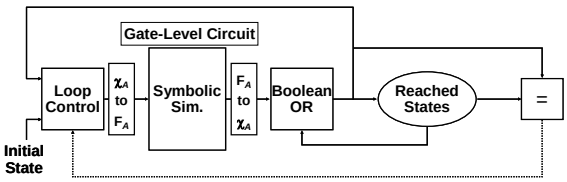
- Not clear how to do these!

Parametric Representation of Next State Set



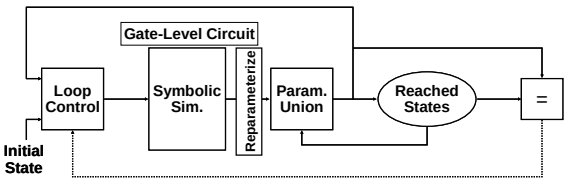
- One step of symbolic simulation generates parametric form of image computation
  - Set of states X′ such that X′ = δ(X) for some state X ∈ A

Forward Reachability via Parametric Representation #1



- Coudert & Madre '89
  - Among earliest work on symbolic reachability
- Converted to characteristic function to perform Boolean operations
  - Loses advantage of symbolic simulation

Forward Reachability via Parametric Representation #2



- Amit Goel, CMU '02
- Generate canonical parametric form from any other parametric form
  - Algorithm due to Coudert, Robert Jones
- New algorithm to compute set union in parametric form
  - Does not generate characteristic function explicitly or implicitly

Some Results

Circuit Information			VIS - IWLS		BFV	
Name	# FF	depth	time (s)	Peak (K)	time (s)	Peak (K)
s1269	37	9	8002	7623	565	2210
s1512	57	1023	14431	1348	21851	784
s3271	116	16	-	Memout	1493	2196
s4863	183	4	-	Memout	197	868

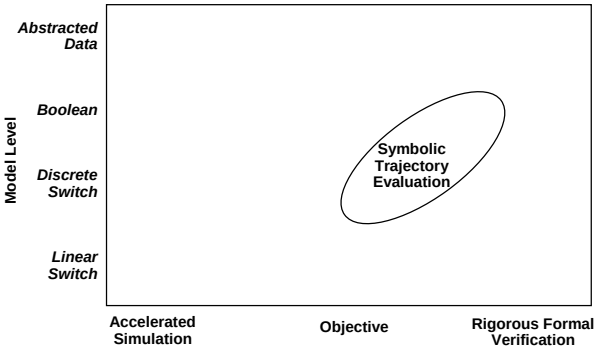
Comparison

- VIS with IWLS partitioning & ordering of transition relation
  - Based on characteristic functions
- Boolean Functional Vectors
  - Based on parametric representation

Performance

- Big improvement for some benchmarks

Symbolic Trajectory Evaluation



Symbolic Trajectory Evaluation

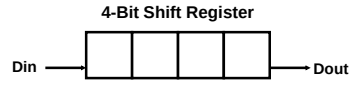
Formulation

- Bryant & Seger (1990)
- View symbolic simulator as form of model checker
  - For limited class of LTL formulas
  - Abstract states with ternary { 0, 1, X } logic

Extensions

- Enlarge class of safety properties
  - Seger (1995), Jain (1997), Chou (1999)
- Add fairness
  - "Generalized Symbolic Trajectory Evaluation"
  - Yang & Seger (2000)
  - All ω-regular properties

STE Example

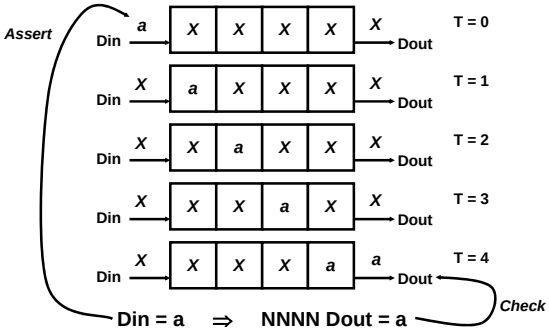


Specification

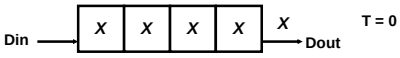
$$\text{Din} = a \quad \Rightarrow \quad \text{NNNN Dout} = a$$

- If apply input "a"
- Then four cycles later, will get output "a"
  - N is "next-time" operator
  - Similar to "X" in other temporal logics

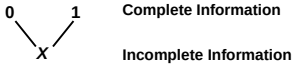
Verification by STE



Mathematical Basis for STE



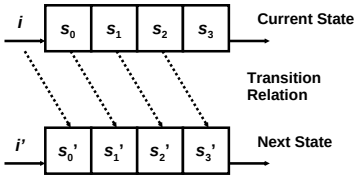
Partially Ordered State Model



Monotonic Circuit Behavior

- Any 0/1 behavior observed with all-X initial state will occur for arbitrary initial state
- Subtle details in simulator implementation

Compare: Model Checking with Characteristic Functions



Encode Entire System State Symbolically

- Two Boolean variables per state bit
- Impractical to model systems with very large memories
- Typically verify models with reduced data widths and memory capacities

Performance of STE

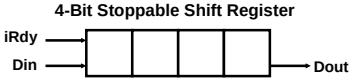
Key Property

- Use symbolic variables only to encode input and (part of) initial state
- Verification complexity depends on complexity of specification, not of system
- Can verify systems containing large memories

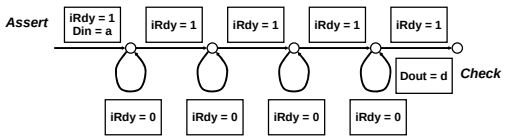
Industrial Applications of STE

- Motorola: Verify variety of memory subsystems
- Intel: Block-level verification

Increasing STE Expressive Power

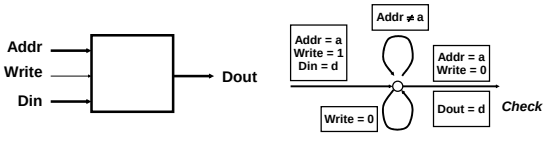


Specification



- Graphical notation more expressive and intuitive than textual
- Allows arbitrary number of idle cycles between inputs
- Implemented with simple fixed-point operation

RAM Verification by STE



Specification

- Perform write with address *a*
- Perform arbitrary number of reads, or operations with a different address
- Perform read with address *a*
  - Should get value *d* on Dout

Verification requirements for 2^m-bit memory

- Constant number of iterations
- O(m) Boolean variables

Generalized STE

- Yang & Seger (2000)

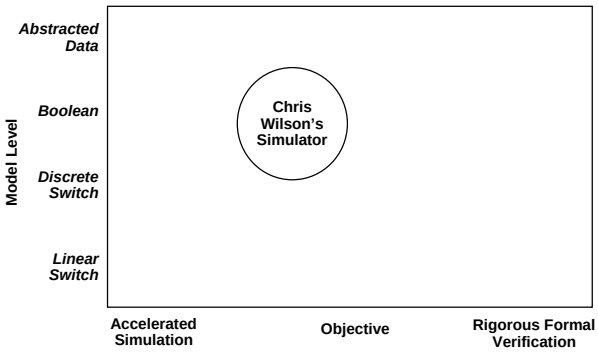
Extends Class of Trajectory Graphs
- Arbitrary graph structure

Adds Fairness Constraints
- Require that specified arcs be traversed infinitely often

Very Expressive
- ω-regular languages

Not Directly Comparable to CTL Model Checking
- Cannot express existential properties in GSTE
- Cannot describe path properties in CTL

Chris Wilson's Simulator



Wilson's Symbolic Simulator

- Chris Wilson, PhD, Stanford (2001)

Less Pessimistic X Handling
- Can verify simple forms of data propagation

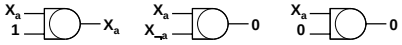
Automatic Variable Classification
- When to use X's, and when to use symbols
- Major headache for users of other symbolic simulators
  - Too many → get X's for check values
  - Too few → BDD blowup

Integrate BDDs with Explicit Case Simulation
- When BDDs get too big, start enumerating variable values rather than encoding them symbolically
- Guarantees useful partial results

Tagged X Values

Can Tag X with Literal
- X_a, $X_{\neg a}$, X_b, $X_{\neg b}$, etc.

Allow Limited Propagation of Tags



- When value depends on multiple tags, revert to regular X

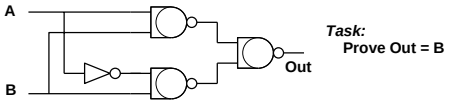
Handles Simple Data Propagation
- Data moved across busses, stored in registers, passed through multiplexors

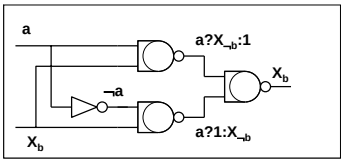
Automatic Variable Classification

Two Ways to Represent Symbolic Value
- BDD variable *a*
- Tagged X value X_a

Strategy
- Start with only tagged X's
- Simulate symbolic test
- If check is X, then select some symbol to strengthen
  - As BDD variable, rather than as tagged X
- Resimulate
- Continue process until check either proved or disproved

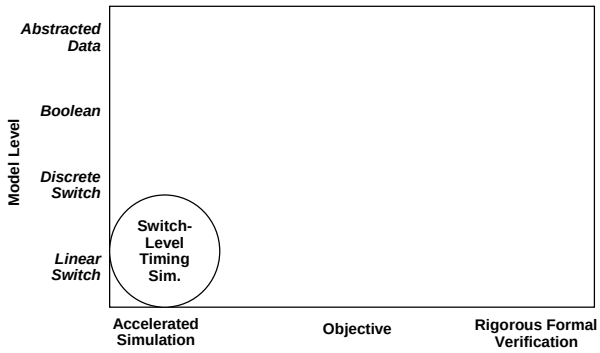
Reclassification Example



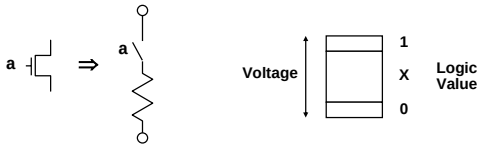


- Simple heuristics determine which variable to strengthen
- Must rerun entire simulation every time strengthen variable

Switch-Level Timing Simulation



Linear Switch-Level Simulation



Linear Switch-Level Simulation

- RSIM (Terman), nRSIM (Chu), IRSIM (Horowitz)
- Model transistor as switched, linear resistor
- Ternary (0, 1, X) node states
- Elmore (RC product) model of circuit delay

Symbolic Timing Simulation

Symbolic Implementation of Linear Switch-Level Simulation

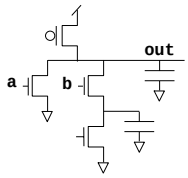
- SirSim: McDonald, ICCAD '99
- Symbolic Extensions
  - BDD node values
  - MTBDD delay calculations
- Exactly equivalent to running 2^n IRSIM simulations

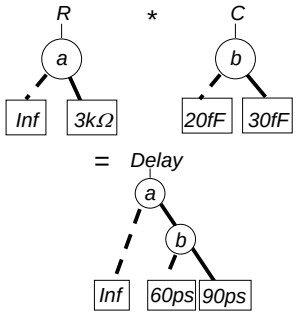
Is This Formal Verification?

- Model is too simplistic to justify this

Symbolic Delay Calculation

From "a" rising to "out" falling



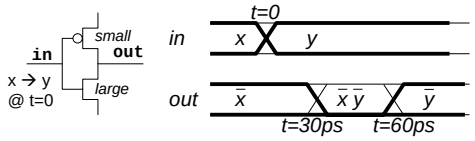


- Delays computed as (driver resistance) * (load capacitance)

Handling Data-Dependent Delays

- Schedule event for each possible time point
- Event includes mask indicating conditions under which update should occur

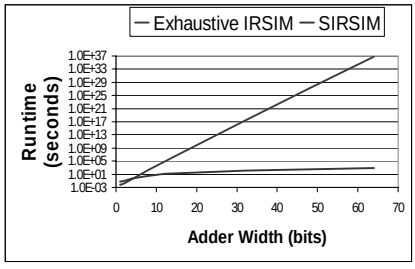
NodeVal = (Mask & NewVal) ∨ (¬Mask & OldVal)



@t=30ps : out = (y &¬y ∨ ¬y & ¬x) = ¬x & ¬y

@t=60ps : out = (¬y & ¬y ∨ y & ¬x & ¬y) = ¬y

Manchester Adders



- Speedup of 10^{33} over exhaustive IRSIM for 64 bit adder
- Sirsim < 15 min
- IRSIM > 10^{29} yrs
- Runtime=O(n^3)

R. E. Bryant, Symbolic Simulation, 2002

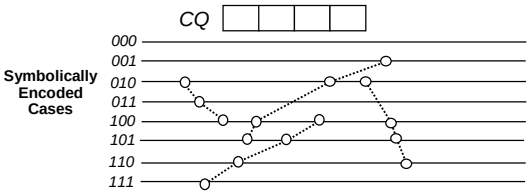
Alpha Microprocessor Circuits

Description	#FETs	#I/Os	Time
56-bit way select	1500	228	28 sec.
52-bit magnitude compare	1539	106	117 sec.
64-bit barrel shifter	8192	196	20 sec.

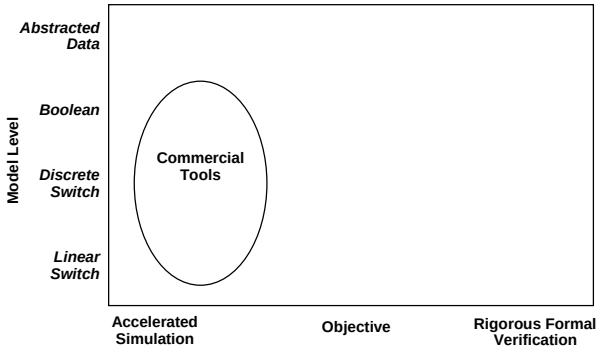
Cluster Scheduling

Group events into clusters with symbolic event times

- **"Cluster-Queue" structure maintains proper ordering**
- **Up to 8x speedup on previously published cases**
- **Exponential speedup demonstrated**



Commercial Symbolic Simulators



Commercial Symbolic Simulators

Innologic

- **Verilog-Based Symbolic Simulator**
  - **Handles all of Verilog**
  - **Not just synthesizable subset**
- **Extend input vector format to allow symbolic values**
- **Biggest successes to date are in memory verification**

Synopsys

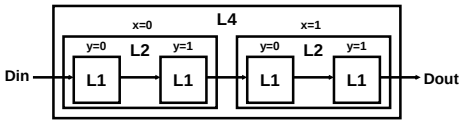
- **Part of formalVERA (a.k.a., Ketchum) assertion checker**
  - **Uses multiple strategies: automatic test generation, symbolic simulation, bounded model checking**

Exploiting Hierarchy

Hierarchical Modeling

- **Symbolically encode circuit structure**
  - **Based on hierarchy in circuit description**
- **Simulator operates directly on encoded circuit**
  - **Use symbolic variables to encode both data values & circuit structure**
- **Implemented by Innologic, variant by Synopsys (DAC '02)**

Hierarchical Circuit Representation



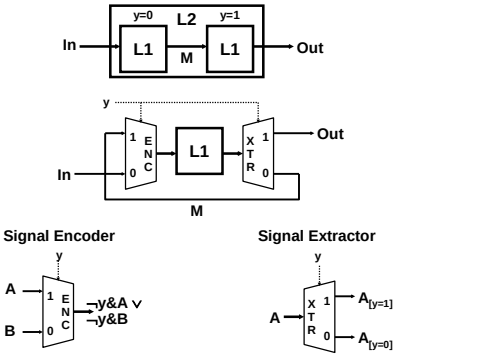
Hierarchy

- **Follows that in circuit representation**

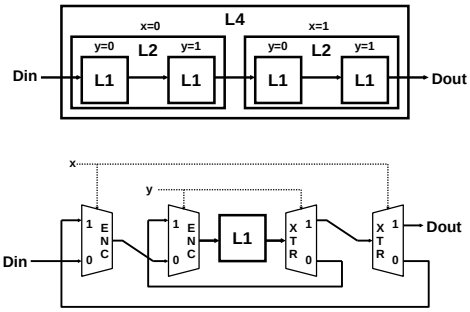
Encoding

- **Introduce Boolean variables to encode module instances**

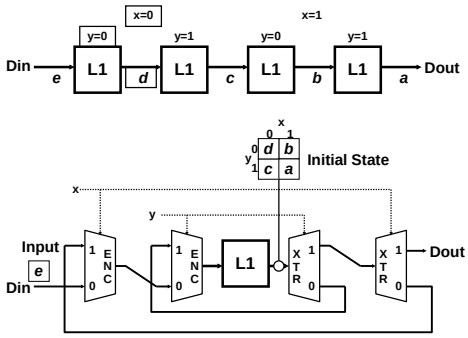
Symbolically Encoding Circuit Operation



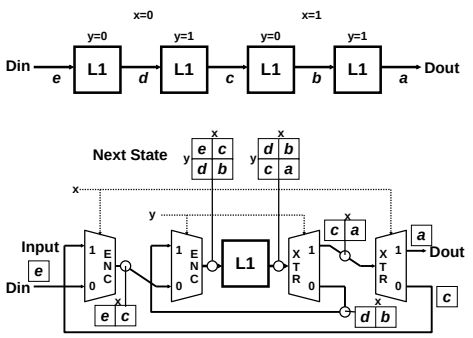
Symbolically Encoding Circuit Operation



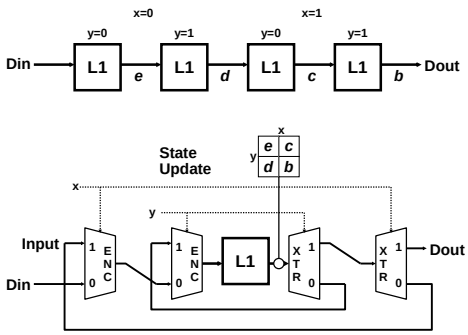
Simulating with Encoded Circuit



Simulating with Encoded Circuit



Simulating with Encoded Circuit



State Encoding Advantage

Possibilities

- Exponential reduction in circuit representation
- Exponential reduction in state representation

Example Verification (from Innologic)

- 256-Mbit memory
- Fully verified

Useful with Conventional Simulation

- Conventional wisdom
  - Cannot simulate circuit with less than 1 bit / node
  - To store state of each node
- Can beat this with encodings!

Conclusions

Symbolic Simulation Occupies Important Niche

- Accelerated simulation
- Specific forms of formal verification
  - Especially good at circuits with large memories
  - Regular model checking perhaps better for control-intensive circuits

Niche is Expanding

- Greater generalizations as formal verifier
- Improved efficiency
  - Better use of X's
  - Hierarchical encoding
- More sophisticated circuit models

Some Research Challenges

Merging Model Checking with STE

- Enlarge class of properties handled by STE
  - Include existential properties
- Make use of X's to perform data abstraction in model checking

Debugging with Symbolic Simulation

- How to communicate failure information to users
- Wealth of information, but need useful distillation

Coverage Metrics

- Is there any useful way to compare coverage by symbolic simluation to that by conventional simulation?
- Conventional simulation covers miniscule fraction of cases, but seems to find most of the bugs