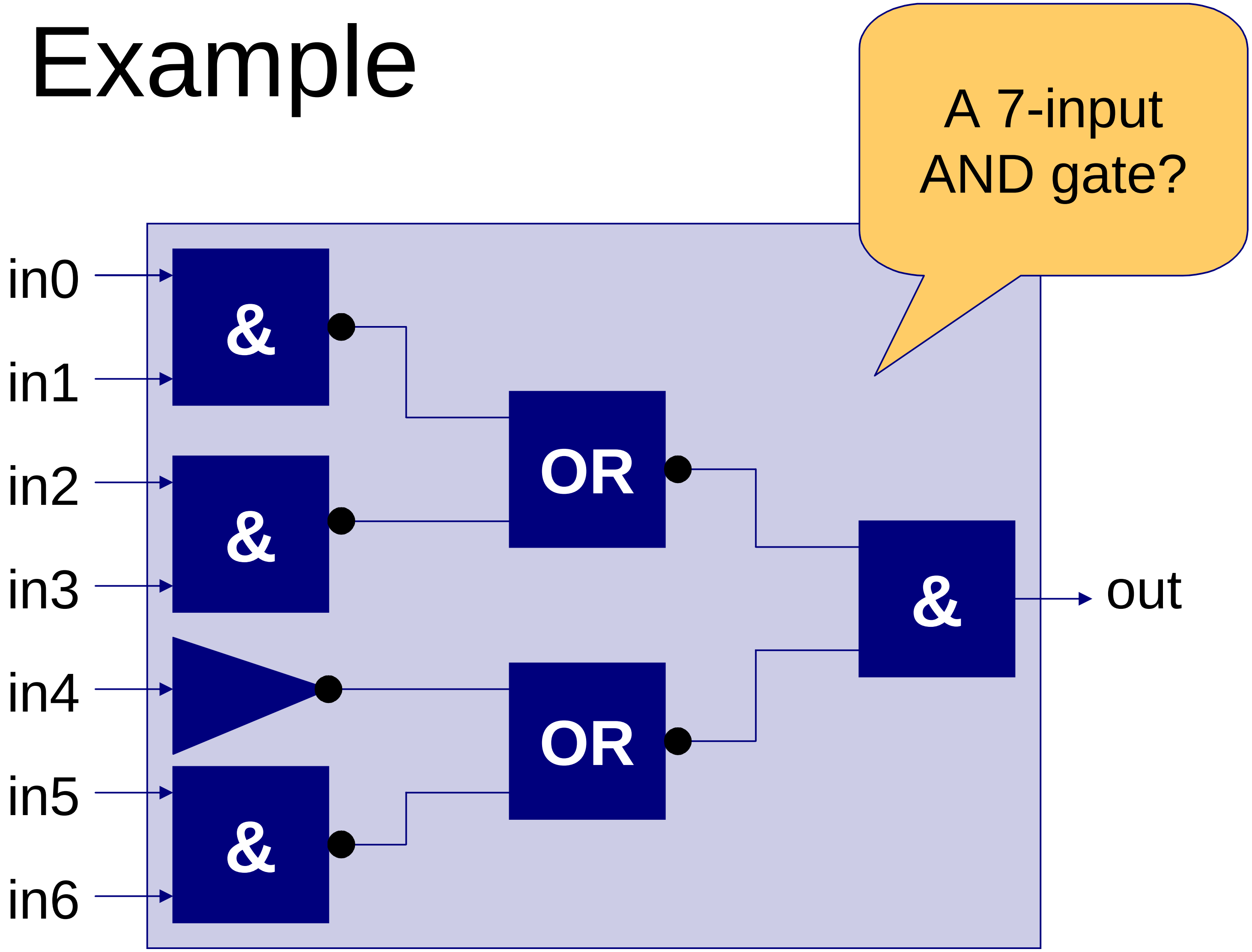
An Introduction to Symbolic Trajectory Evaluation

Koen Lindström Claessen
Chalmers University / Jasper AB
Gothenburg, Sweden

An Example



Verification by Simulation

(in0 **is** 0) **and**
(in1 **is** 0) **and**
(in2 **is** 1) **and**
(in3 **is** 1) **and**
(in4 **is** 0) **and**
(in5 **is** 1) **and**
(in6 **is** 0) ➔
(out **is** 0)

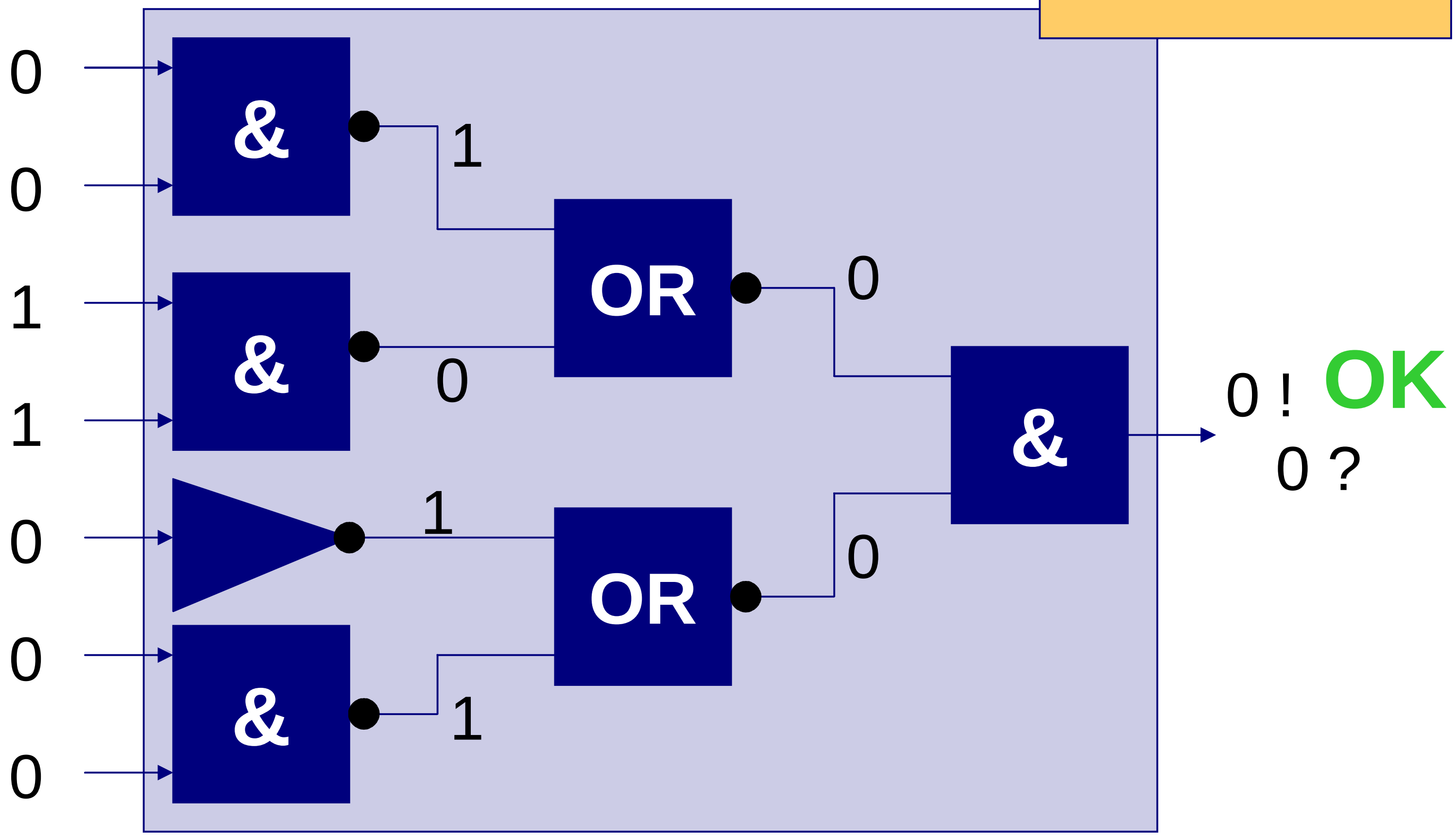
"Antecedent" *driving*

Simulation specification

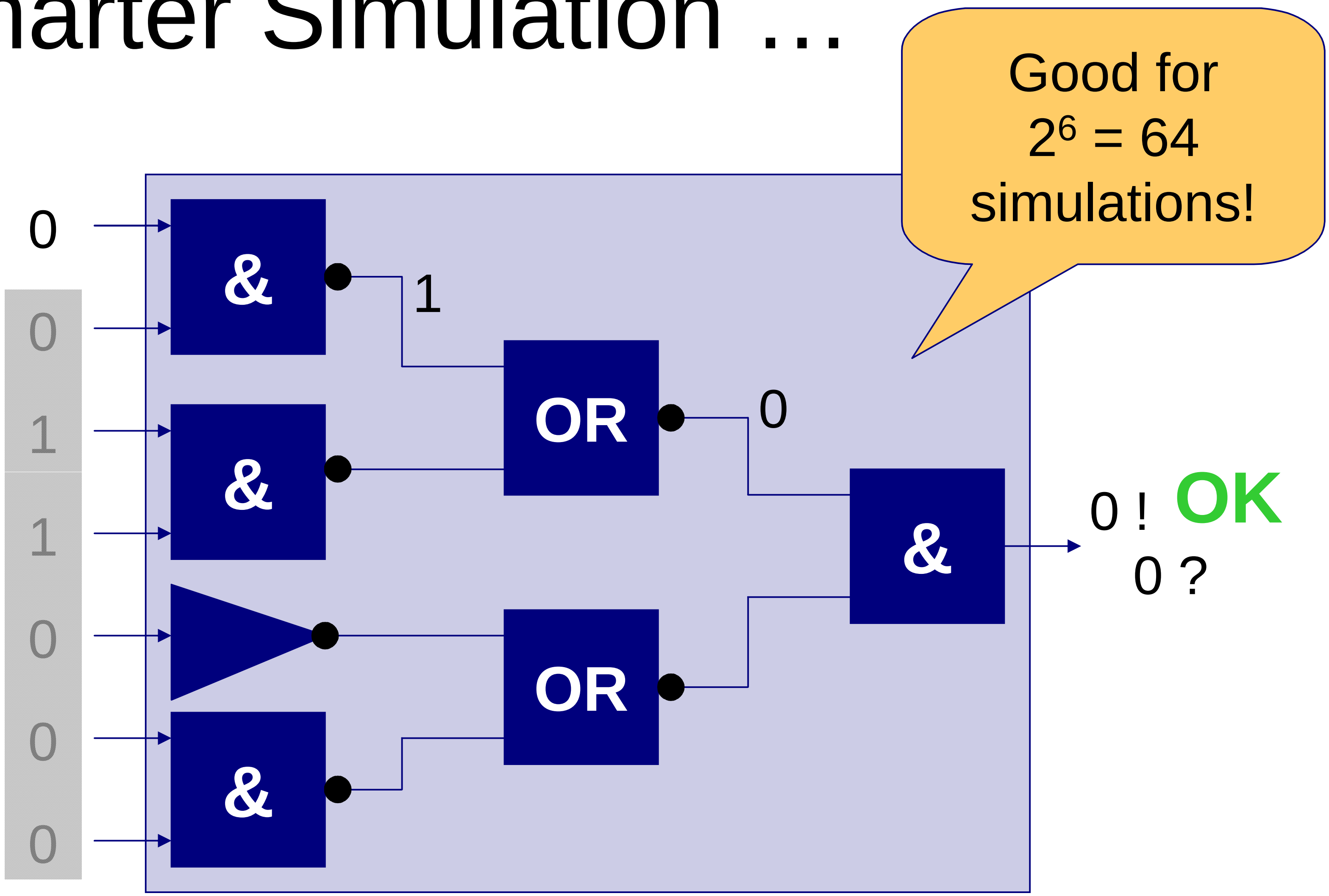
"Consequent" *checking*

Simulation ...

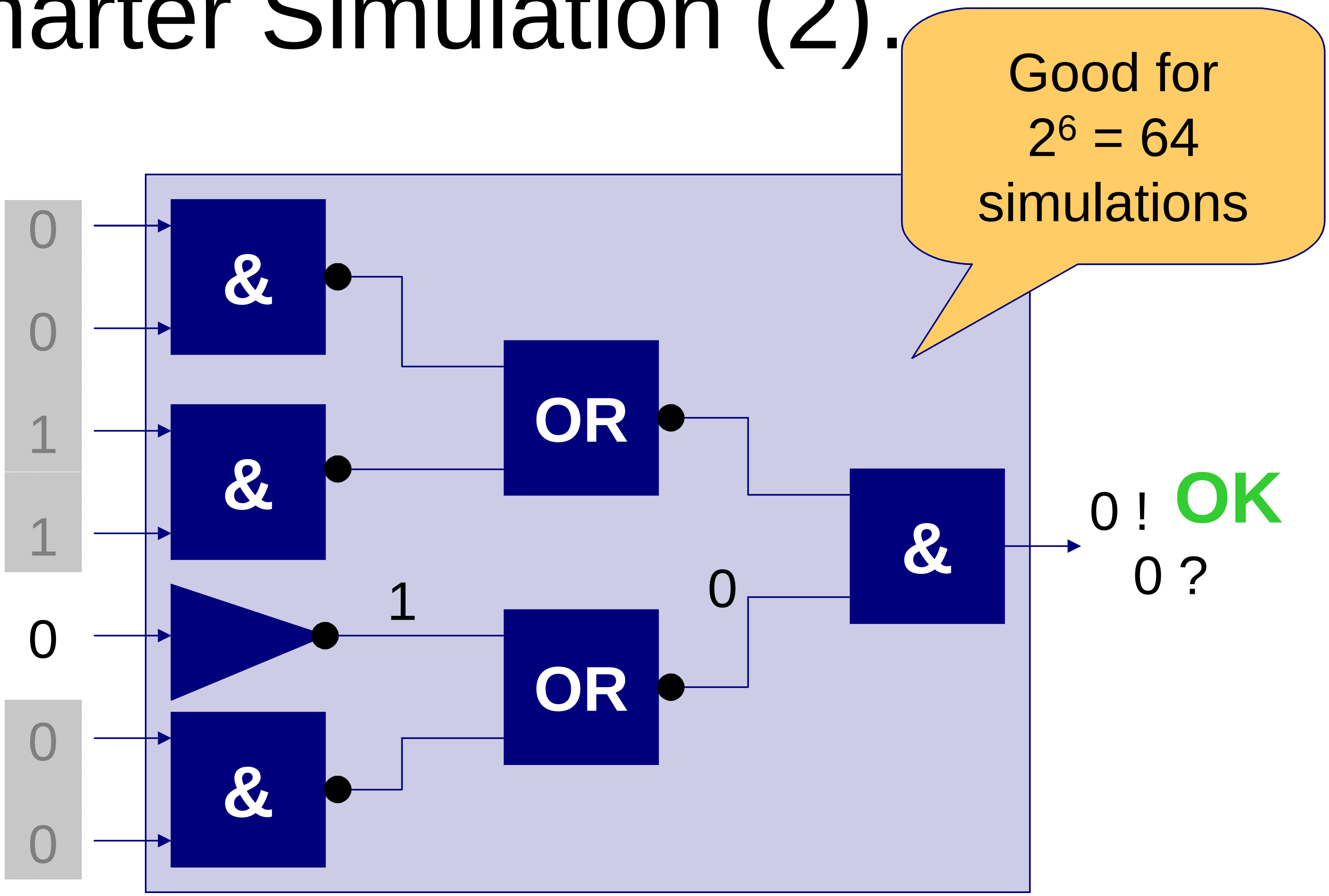
$2^7 = 128$
simulations



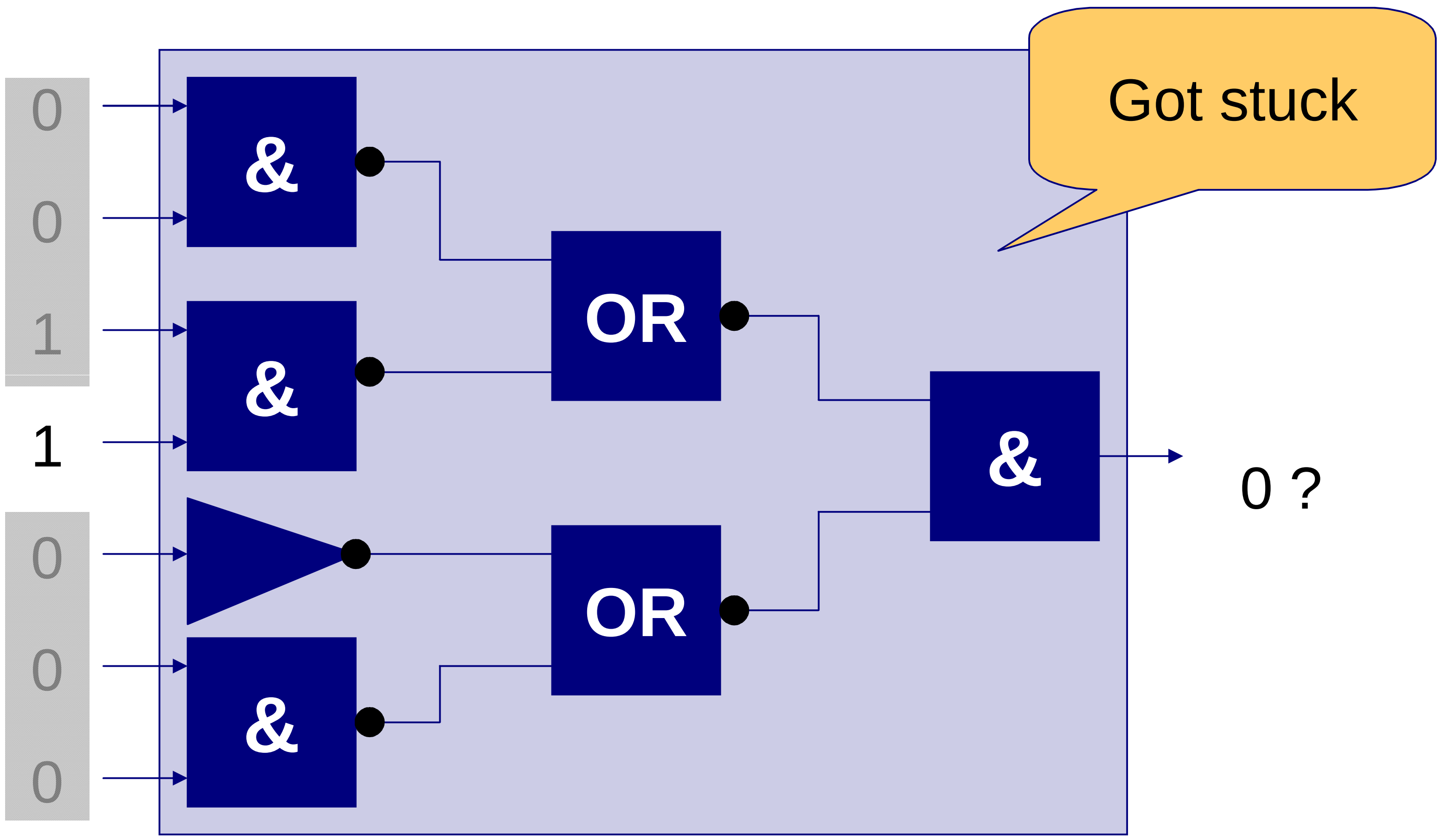
Smarter Simulation …



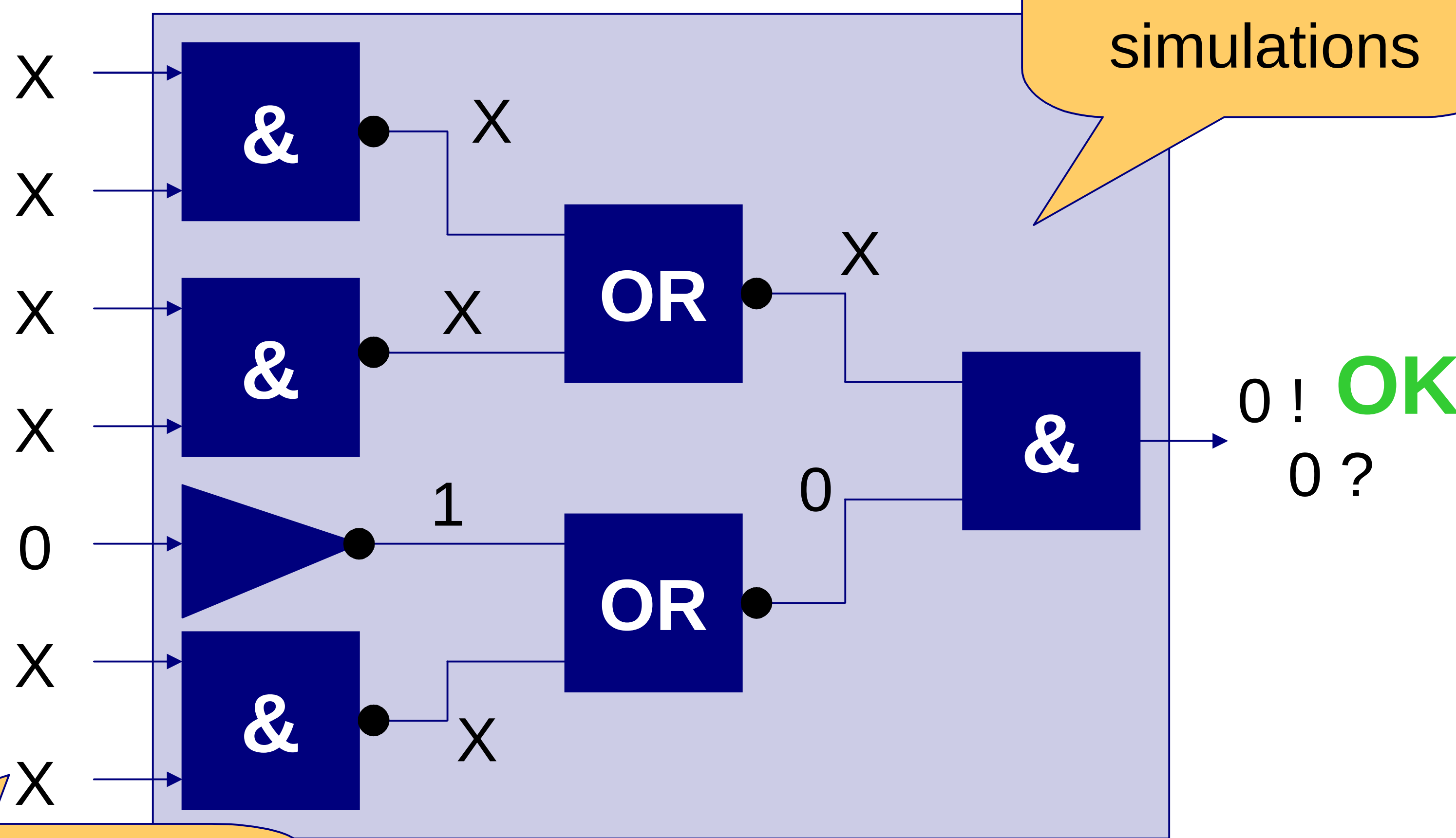
Smarter Simulation (2)...



Smarter Simulation?



Three-Valued Simulation: 0 1 X



Good for
$2^6 = 64$
simulations

X = “unknown”

Simulating with 0,1,X

abstraction:
X = {0,1}

x	•x
0	1
1	0
X	X

x y	x & y
0 0	0
0 1	0
1 0	0
1 1	1
X 0	0
0 X	0
X 1	X
1 X	X
X X	X

x y	x OR y
0 0	0
0 1	1
1 0	1
1 1	1
X 0	X
0 X	X
X 1	1
1 X	1
X X	X

enough information

not enough information

Three-Valued Specification

- (in0 **is** 0) ➔ (out **is** 0)
- (in1 **is** 0) ➔ (out **is** 0)
- (in2 **is** 0) ➔ (out **is** 0)
- (in3 **is** 0) ➔ (out **is** 0)
- (in4 **is** 0) ➔ (out **is** 0)
- (in5 **is** 0) ➔ (out **is** 0)
- (in6 **is** 0) ➔ (out **is** 0)
- (in0 **is** 1) **and** (in1 **is** 1) **and** … **and** (in5 **is** 1) **and** (in6 **is** 1) ➔ (out **is** 0)

not mentioned in antecedent means driven with “X”

8 simulations in total

Symbolic Simulation

- Boolean expression datatype
  - Variables; a, b, c
  - Logical operations; not, and, or
  - Compositional
  - Canonical representation

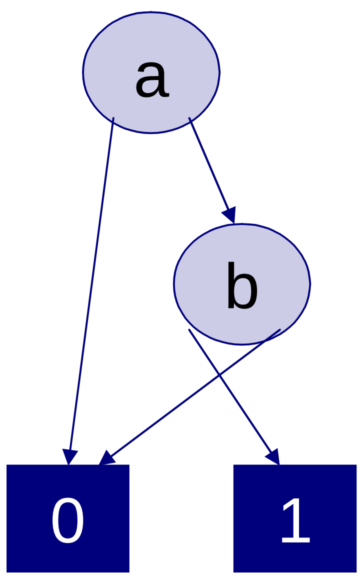
(Reduced Ordered) Binary Decision Diagrams (BDDs)

Compositional?

F & G

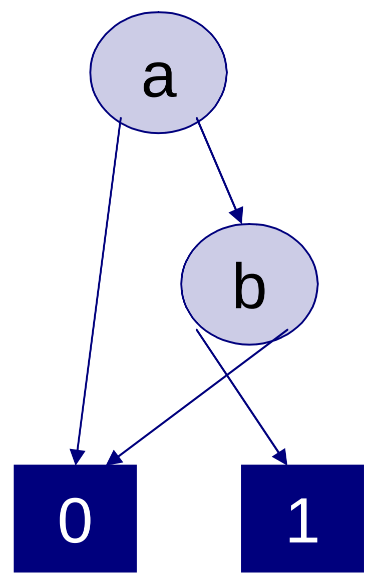
Canonical?

a & ~b



=

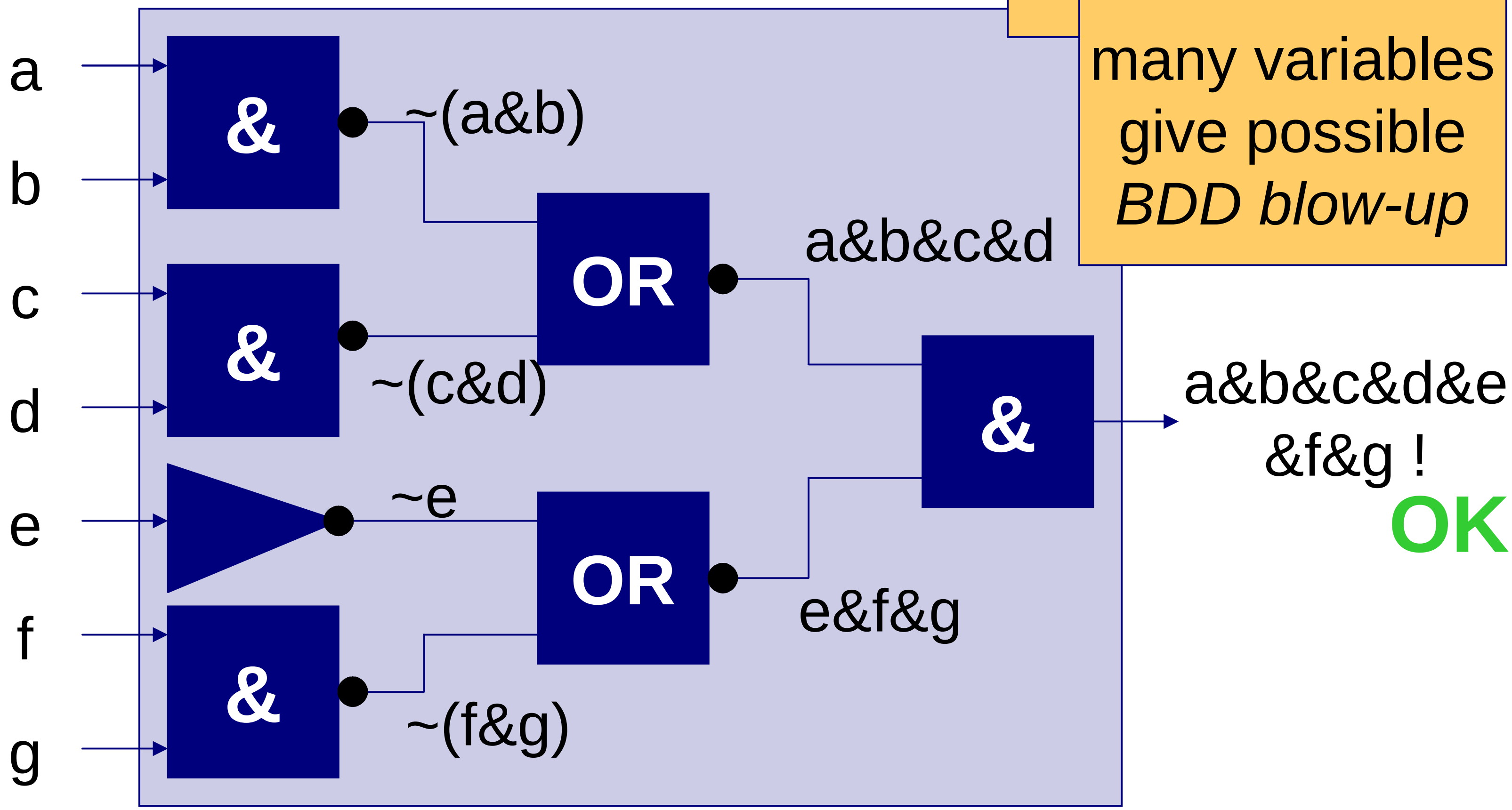
~(~a v b)



Symbolic Simulation …

only 1 simulation!

many variables give possible *BDD blow-up*



Symbolic Specification

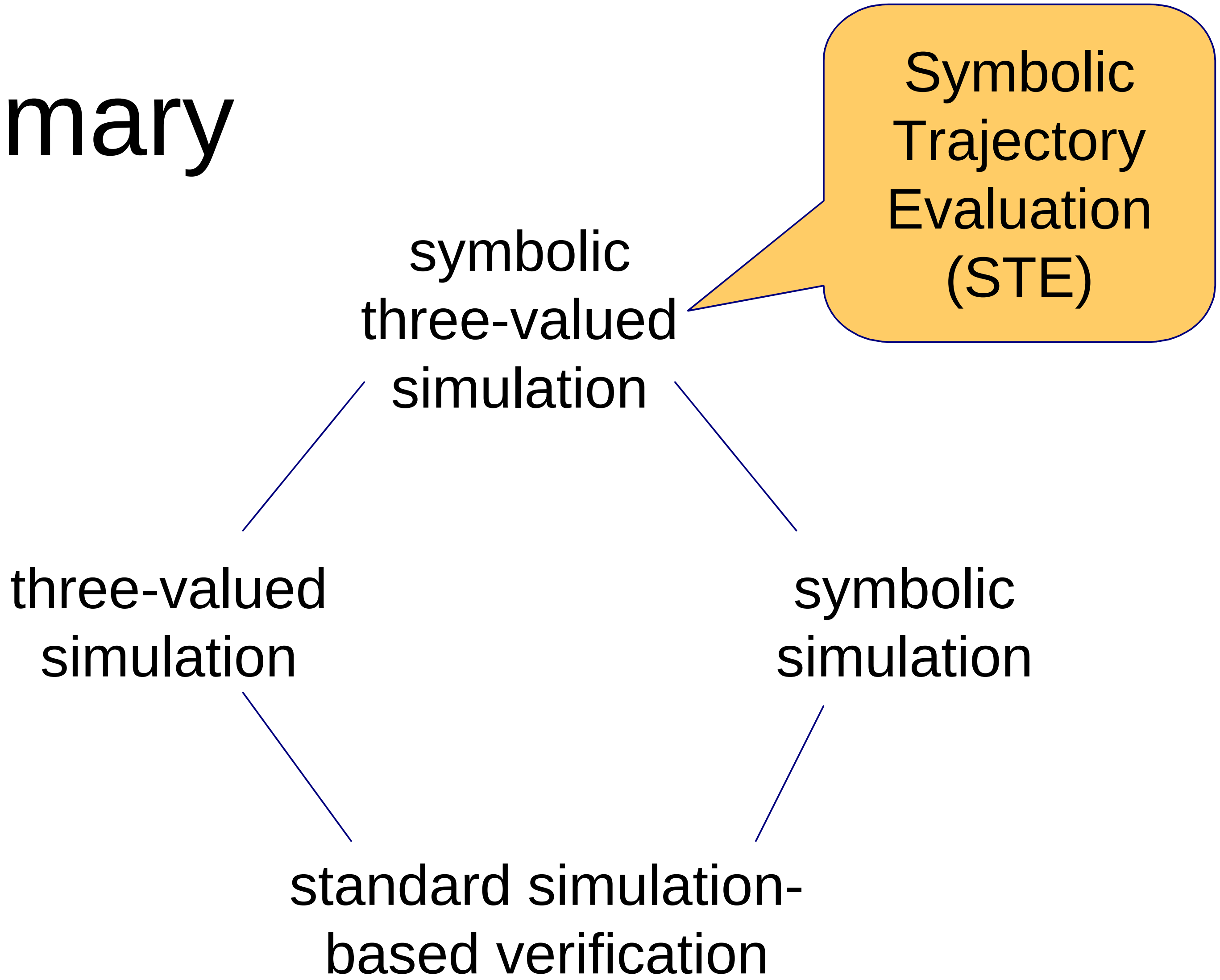
(in0 **is** a) **and**
(in1 **is** b) **and**
(in2 **is** c) **and**
(in3 **is** d) **and**
(in4 **is** e) **and**
(in5 **is** f) **and**
(in6 **is** g) ➔
(out **is** (a&b&c&d&e&f&g))

symbolic variable

expected symbolic value

circuit node

Summary


Symbolic Trajectory Evaluation (STE)
symbolic three-valued simulation
three-valued simulation
symbolic simulation
standard simulation-based verification

Idea

- 128 ordinary simulations
  - require 7 symbolic variables
- 8 three-valued simulations
  - require only 3 symbolic variables!
  - call these p,q,r
- When p=q=r=1, all inputs are 1
- Otherwise, <pqr> indicates which input is 0
- Expected value of out?

"symbolic indexing"

out **is** (p&q&r)

STE Spe n

→ is a new operator

((~p&~q&~r) → (in0 **is** 0)) **and**
((~p&~q& r) → (in1 **is** 0)) **and**
((~p& q&~r) → (in2 **is** 0)) **and**
((~p& q& r) → (in3 **is** 0)) **and**
((p&~q&~r) → (in4 **is** 0)) **and**
((p&~q& r) → (in5 **is** 0)) **and**
((p& q&~r) → (in6 **is** 0)) **and**
((p& q& r) → ((in0 **is** 1) **and** (in1 **is** 1) **and** …
and (in5 **is** 1) **and** (in6 **is** 1)))
➔ (out **is** (p&q&r))

Only 3 symbolic variables; less risk of blow-up!

Conditional Driving

P → A

Only use A to drive simulation when P is true

Otherwise, nodes in A are unknown: X

Logically: Implication

Three-Valued Symbolic Expressions

- Simulator needs to deal with
  - boolean values 0,1
  - unknown value X
  - symbolic variables a, b, c
  - expressions with **&**, **OR**, ●, over the above
- Solutions
  - new datastructure
  - dual-rail encoding

Dual-Rail Encoding

Each three-valued entity is represented by a pair of two-valued entities

x0 says when x is 0

x1 says when x is 1

x	(x0,x1)
0	(1,0)
1	(0,1)
X	(0,0)

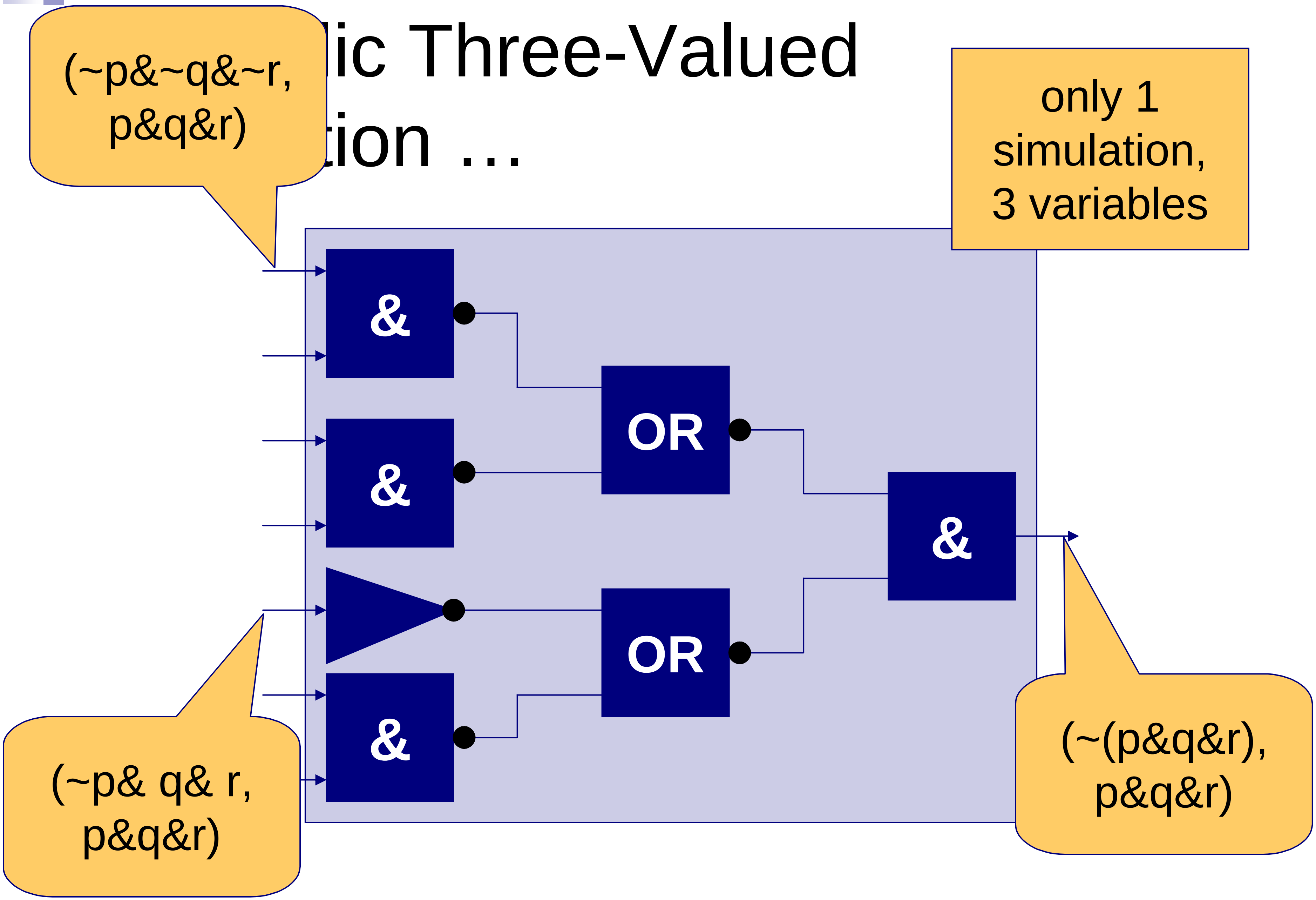
X means neither 0 nor 1

(x0,x1) **&** (y0,y1)
=
(x0 **OR** y0, x1 **&** y1)

(x0,x1) **OR** (y0,y1)
=
(x0 **&** y0, x1 **OR** y1)

●(x0,x1)
=
(x1,x0)

…lic Three-Valued …tion …



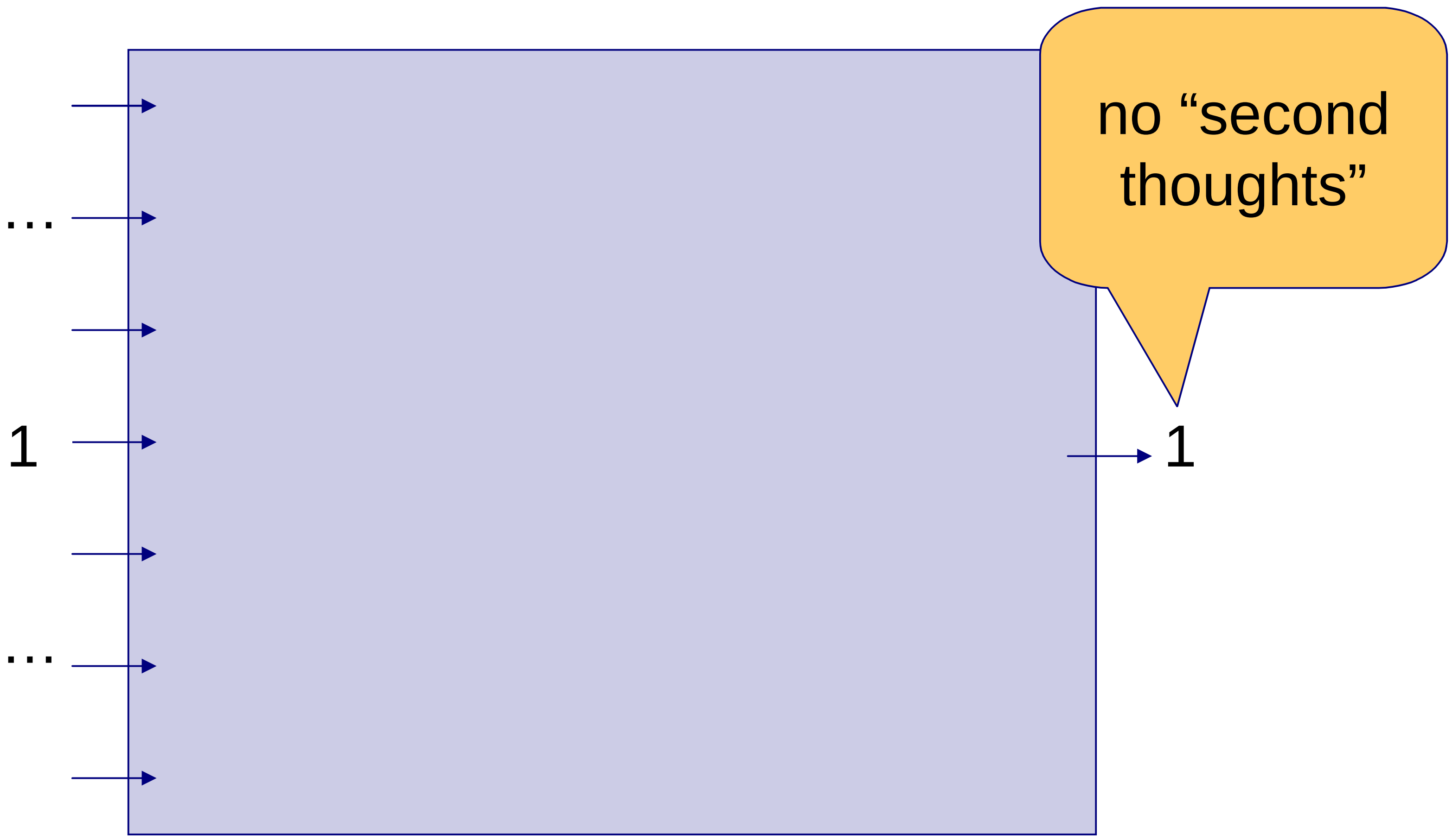
Symbolic Trajectory Evaluation

- Invented in 1995 by Seger and Bryant
- Used industrially
  - Mainly Intel; heavy use
    - Forte
    - ReFLect/IDV
  - Memory-intensive circuits
    - Hard for other verification methods

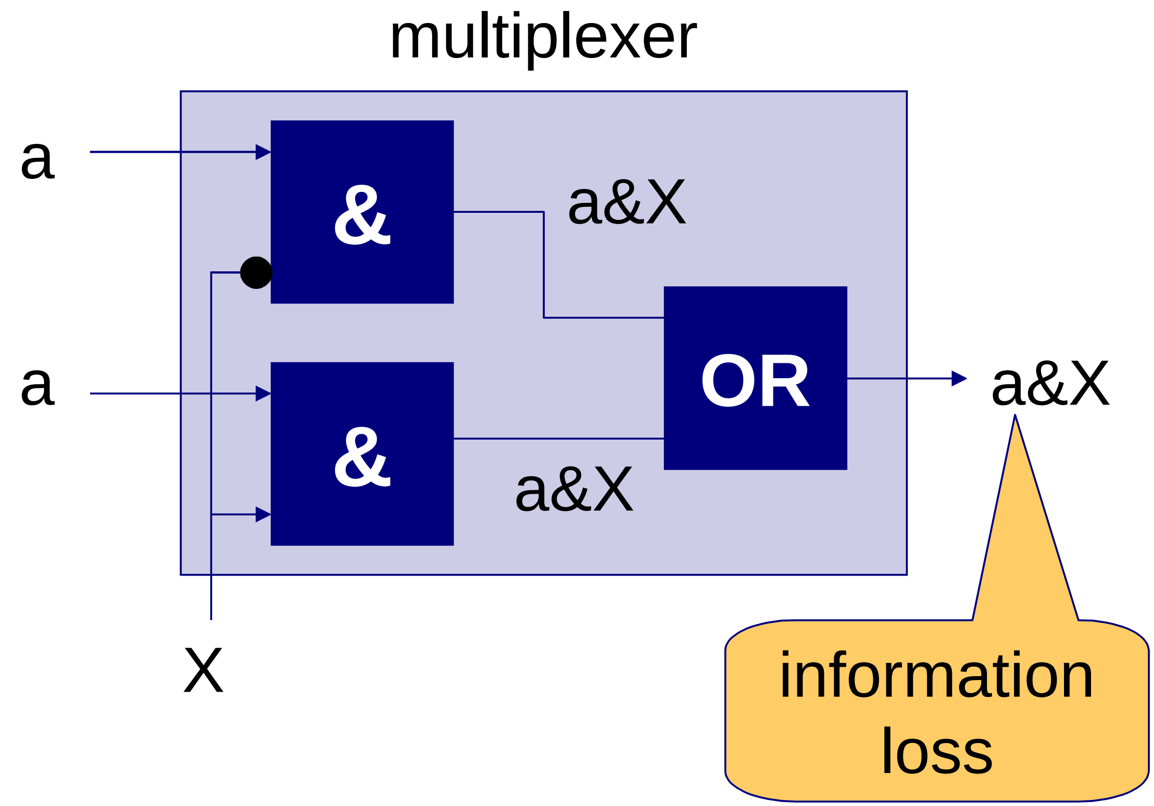
The Rest of this Lecture

- Some pitfalls
- More interesting example: Memory
- Semantics
- Current directions

What Does X Mean?



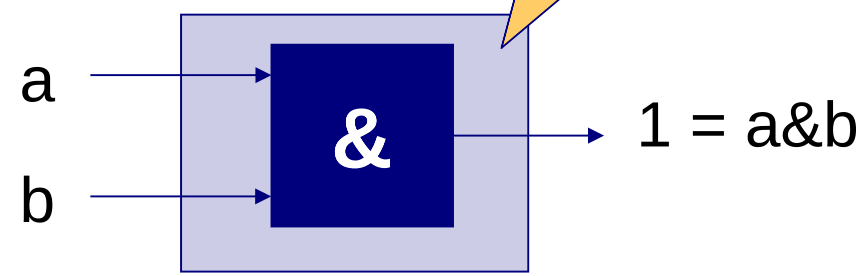
Pitfall 1



manual abstraction

(sel **is** b) **and**
(in0 **is** a) **and** (in1 **is** a) ➔ (out **is** a)

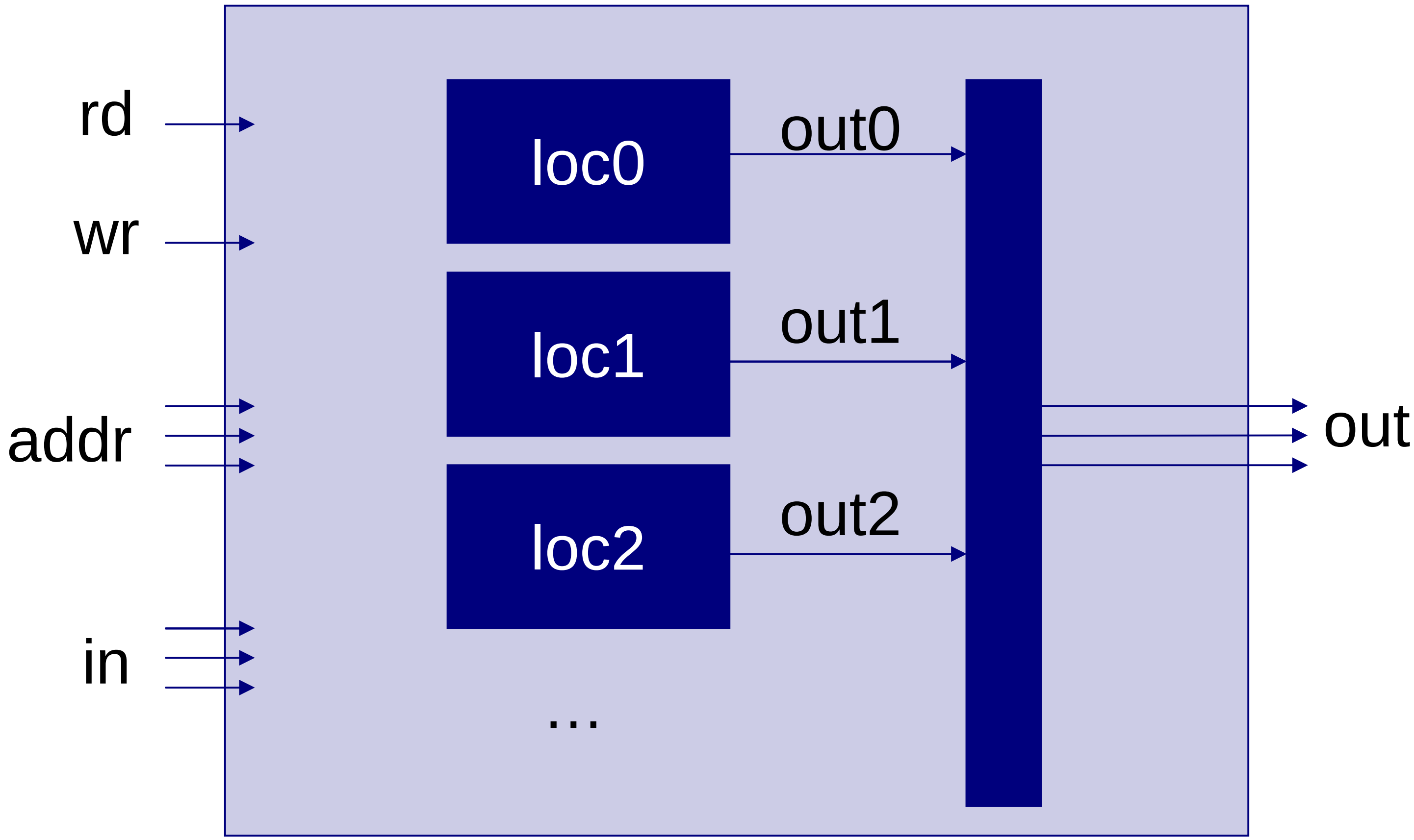
Pitfall 2



(in0 **is** a) **and** (in1 **is** b) **and**
(out **is** 1) ➔ (in0 **is** 1) **and** (in1 **is** 1)

we need a semantics! *predictability*

Example: Memory



Memory

- Address width k
  - 2^k locations
- Data width n
  - n*(2^k) state-holding elements

state-based model checkers?

k=16, n=16: 1048576 elements

A Specification (k=2,n=1)

first we write d to address a0a1

(wr is 1) **and** (in **is** d) **and**

(addr0 **is** a0) **and** (addr1 **is** a1) **and**

then we read from address a0a1

N ((rd is 1) and

(addr0 is a0) and (addr1 is a1)) ➔

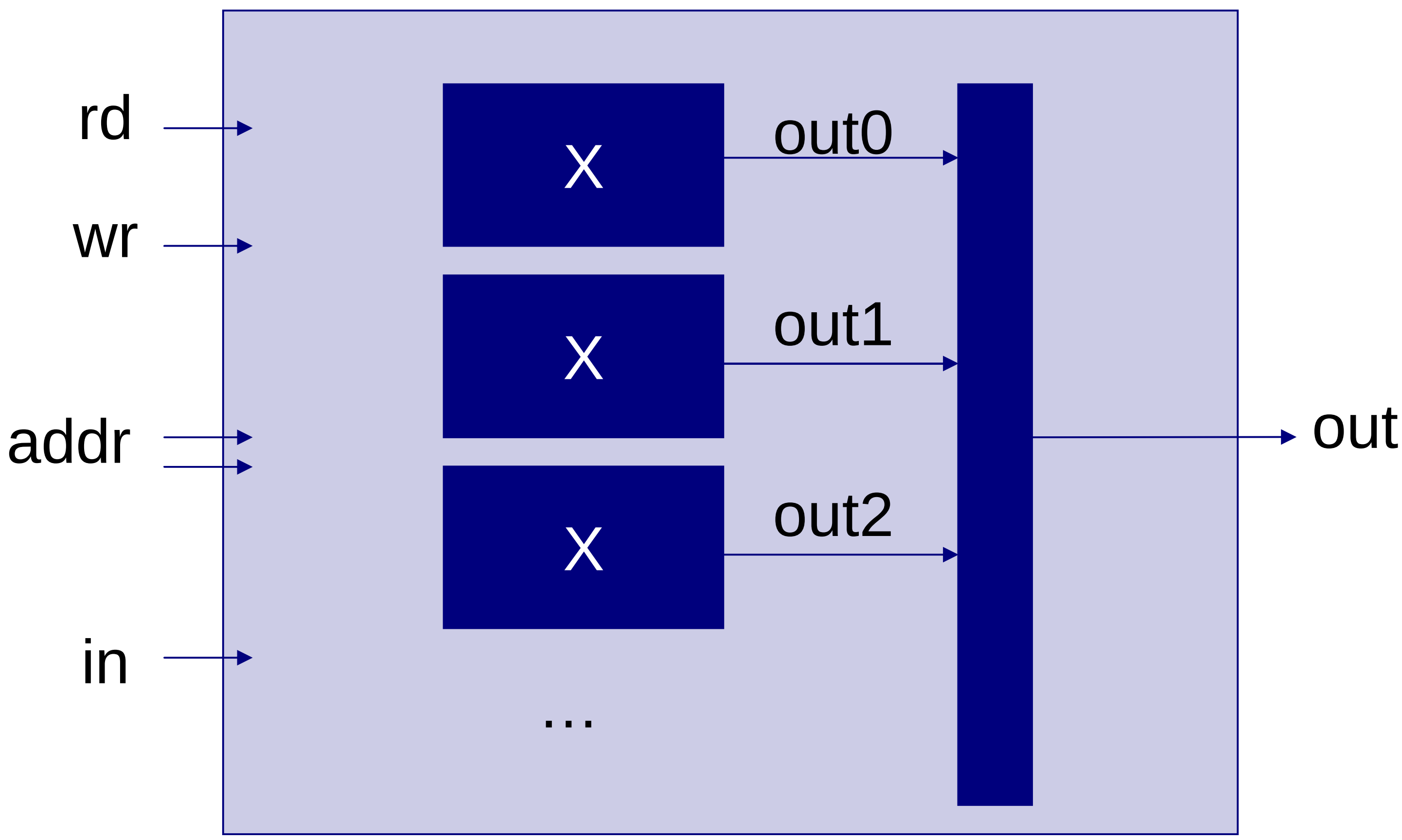
next point in time

N (out **is** d)

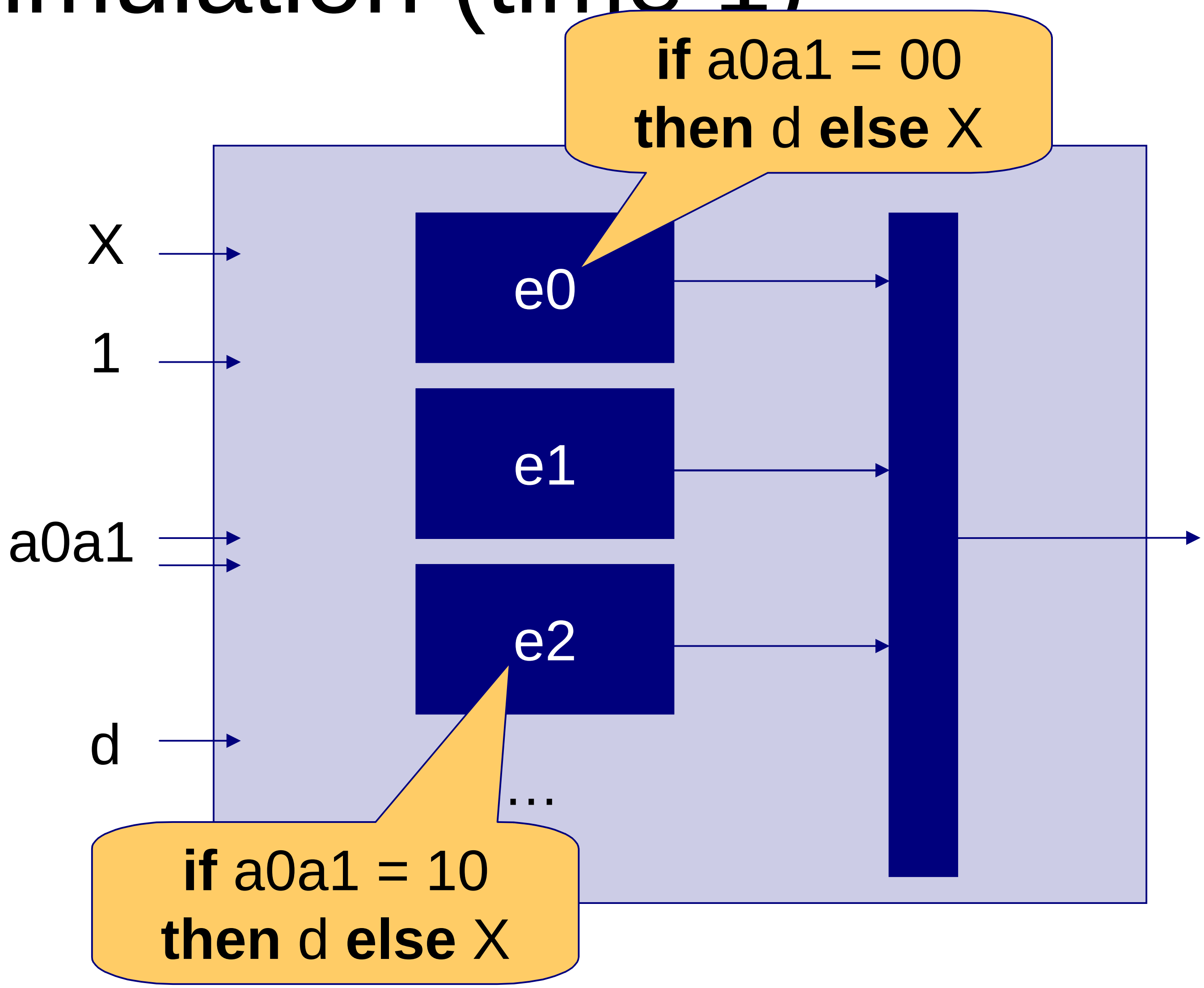
we expect d to come out

symbolic variables: a0,a1: address, d: data

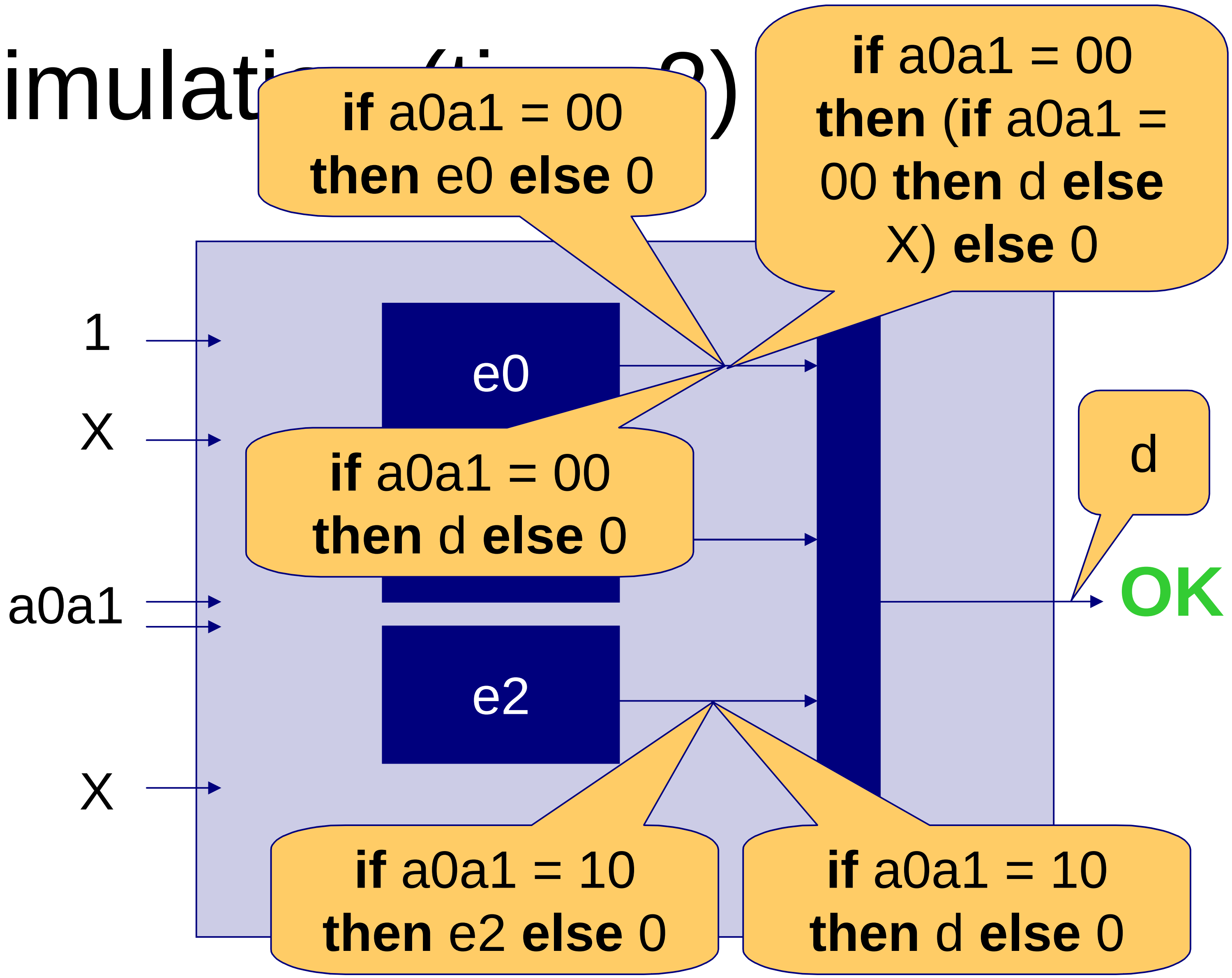
Simulation (initially)



Simulation (time 1)

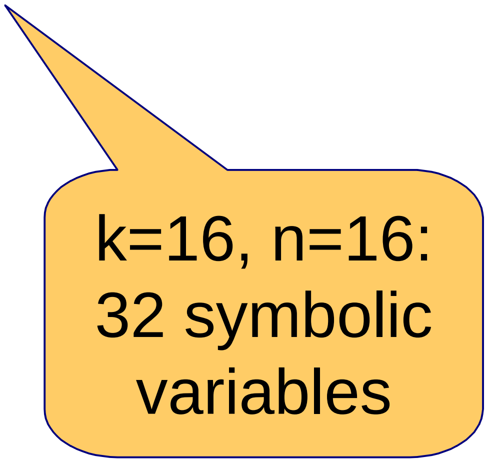


Simulati… (ti… 2)

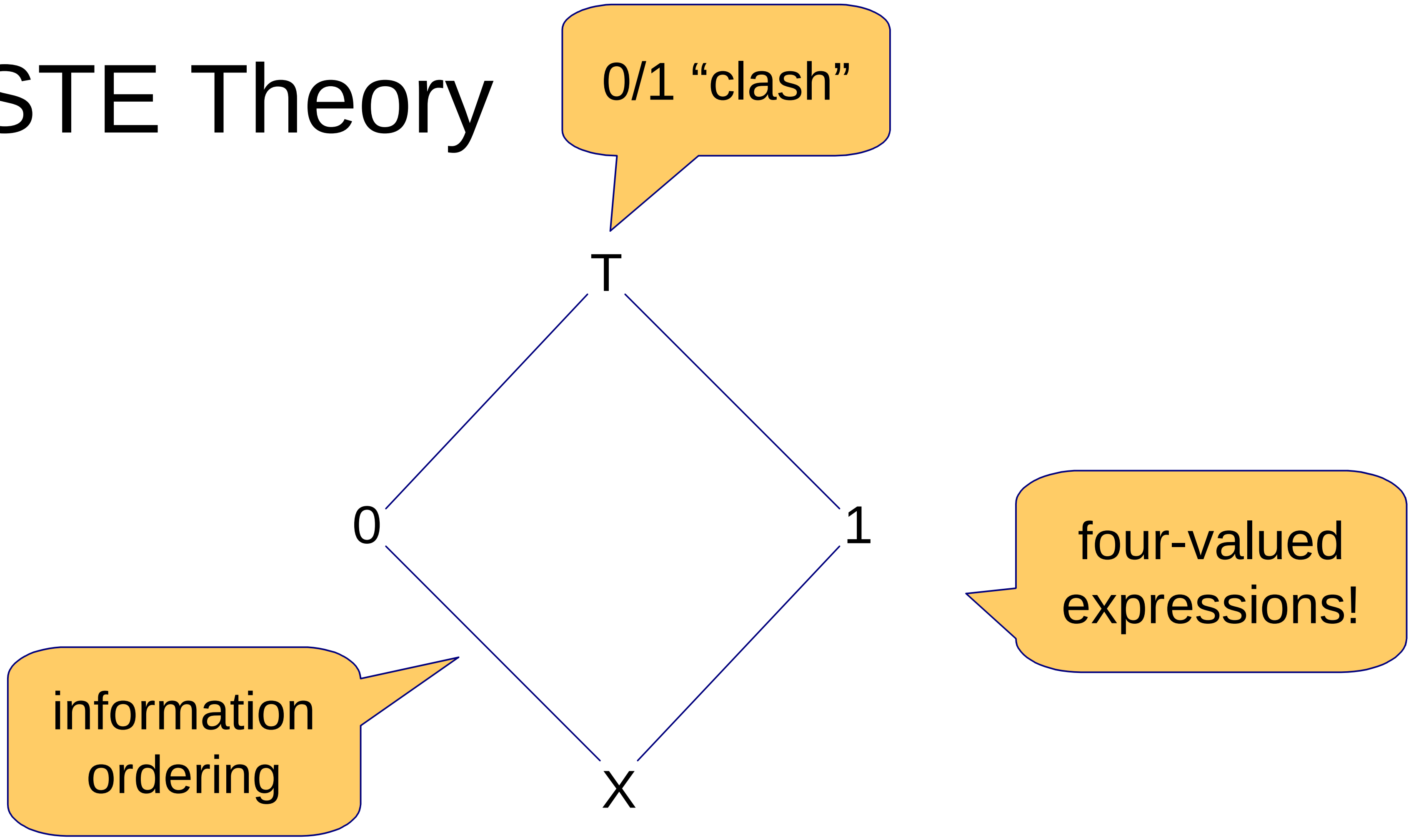


Memory with STE

- Address width k, data width n
  - 2^k locations
  - n*(2^k) state-holding elements
  - k+n symbolic variables



STE Theory



information lattice

4-Valued Gates

- T **&** y = T   y **&** T = T
- T **OR** y = T   y **OR** T = T
- •T = T

- Gates are *monotonic* w.r.t. information ordering

no "second thoughts"

Circuit Model

example:
{in0,in1,out}

- Set of *nodes* N
  - state-holding: n vs n'
- Set of *states* s : S = N → {X,0,1,T}
- Circuits are modelled as *closure functions* F : S → S

propagates given values to other nodes

can be easily constructed from the netlist

Closure Function F : S → S

- Monotonic
  - s1 <= s2 implies F(s1) <= F(s2)
- Idempotent
  - F(F(s)) = F(s)
- Extensive
  - s <= F(s)

no second thoughts

completely simulated

do not invent own things

Sequences of States

- Sequences seq : Seq = Time → S
- Closure function over time F* : Seq → Seq
  - Connecting all state-holding registers
  - Monotonic
  - Idempotent
  - Extensive

Trajectory Evaluation Logic (TEL)

A,B,C ::= n **is** 0
| n **is** 1
| P → A
| A1 **and** A2
| **N** A

n **is** P
shorthand for
(P → n **is** 1) **and**
(~P → n **is** 0)

# of TEL

given boolean evaluation phi for symbolic variables

given a sequence of states seq

phi, seq |= n **is** 0	*iff.* seq(n)(0) >= 0
phi, seq |= n **is** 1	*iff*. seq(n)(0) >= 1
phi, seq |= P $\rightarrow$ A	*iff*. phi |= P *implies* phi,seq |= A
phi, seq |= A1 **and** A2	*iff*. phi,seq|=A1 *and* phi,seq|=A2
phi, seq |= **N** A	*iff*. phi, seq^1 |= A

time shift

Trajectories

sequence following from simulation

- A sequence seq is a *trajectory:*
  - F*(seq) = seq

- Alternatively:
  - Exists seq' . F*(seq') = seq

Final Semantics

F |= A ➔ C

iff.

restriction to three-valuedness

for all phi, and for all trajectories traj of F:

phi,traj |= A *implies* phi,traj |= C

Fundamental Theorem of STE

all trajectories traj of F
for which phi,traj |= A
are characterized by
the *weakest* trajectory traj
for which phi,traj |= A

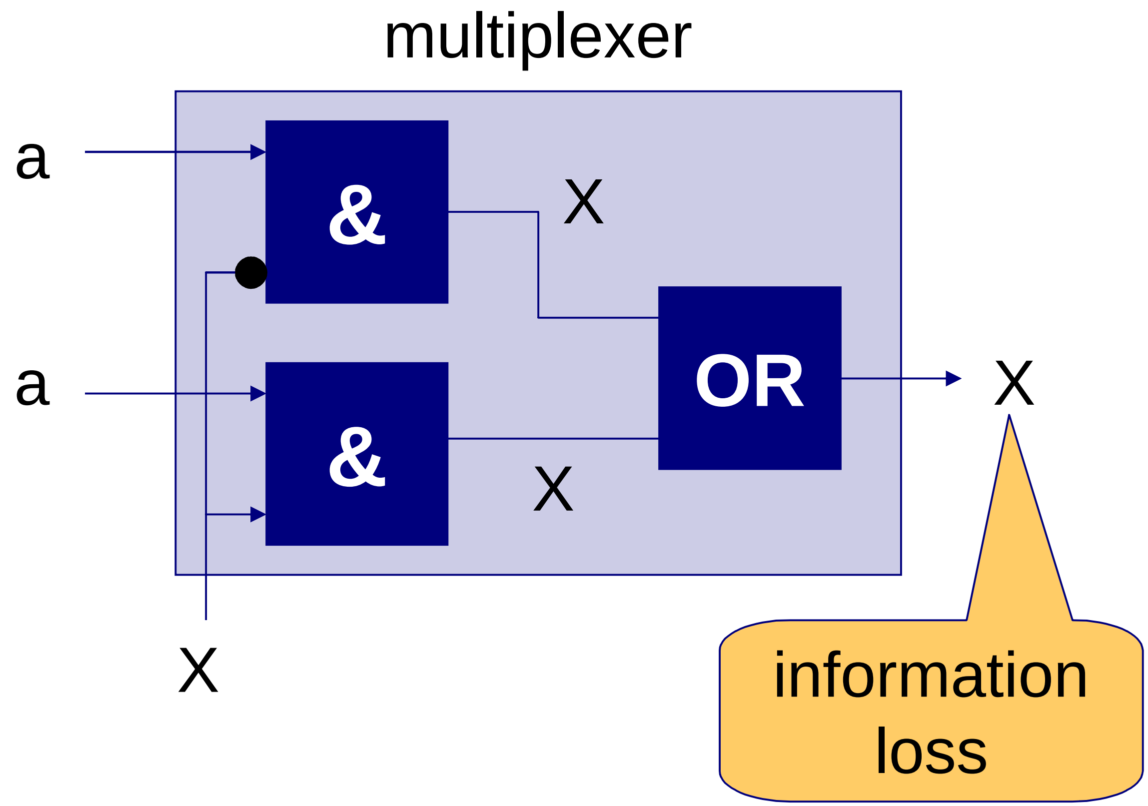
enough to just calculate the weakest trajectory

Abstraction Refinement

- Failed STE assertion
  - "real" counter example
    - something is really wrong
  - "spurious" counter example
    - too many X's in the simulation
- After spurious counter example
  - Specification needs to be *refined*

hard to know what kind

Pitfall 1



(in0 **is** a) **and** (in1 **is** a) ➔ (out **is** a)

"Weakest Strengthenings"

(in0 **is** a) **and** (in1 **is** a) → (out **is** a)

↓

a=1

in0=1

in1=1

sel=1

out=1

(sel **is** 1) **and** (in0 **is** 1) **and** (in1 **is** 1) → (out **is** 1)

weakest *satisfying* strengthening

“Weakest Strengthenings”

(in0 **is** a) ➔ (out **is** a)

⇩

a=1

in0=1

in1=0

sel=1

out=0

weakest *contradicting* strengthening

Weakest Strengthenings

- Implemented in a tool “STAR”
- SAT-based
- Available from Chalmers
- CAV´06

Content-Addressable Memory (CAM)

- “Lookup table”
- 2 memories: tagmem, datamem
- Each tag is coupled with a data
- Store
- Retrieve

CAM Specification (1)

symbolic variables: t,t0,..,t15,d0,..,d15

too many variables: blow-up!

(rd **is** 1) **and** (tag **is** t) **and**
(tagmem0 **is** t0) **and … and**
(tagmem15 **is** t15) **and**
(datamem0 **is** d0) **and … and**
(datamem15 **is** d15)
➔
((t = t0) → (out **is** d0)) **and … and**
((t = t15) → (out **is** d15))

CAM Specification (2)

symbolic indexing: t,i,d

(rd **is** 1) **and** (tag **is** t) **and**

(i = 0 → (tagmem0 **is** t) **and**

(datamem0 **is** d)) **and**

…

(i = 15 → (tagmem15 **is** t) **and**

(datamem15 **is** d))

➔

(out **is** d)

STAR output

- Weakest *contradicting* strengthening
  - i=3
  - t=0010
  - d=11111100

  - rd=1
  - tag=0010
  - tagmem1=**0010**
  - tagmem3=0010
  - datmem1=XXXXXX**1**X
  - datmem3=111111100
  - out=111111X

the rest is X

Conclusions

- STE
  - Powerful
  - Find the right abstraction
  - This can be hard (help)

STE Limitations

- Expressivity
  - Like LTL with finitely many times
  - No initial states
  - No concept of reachable states

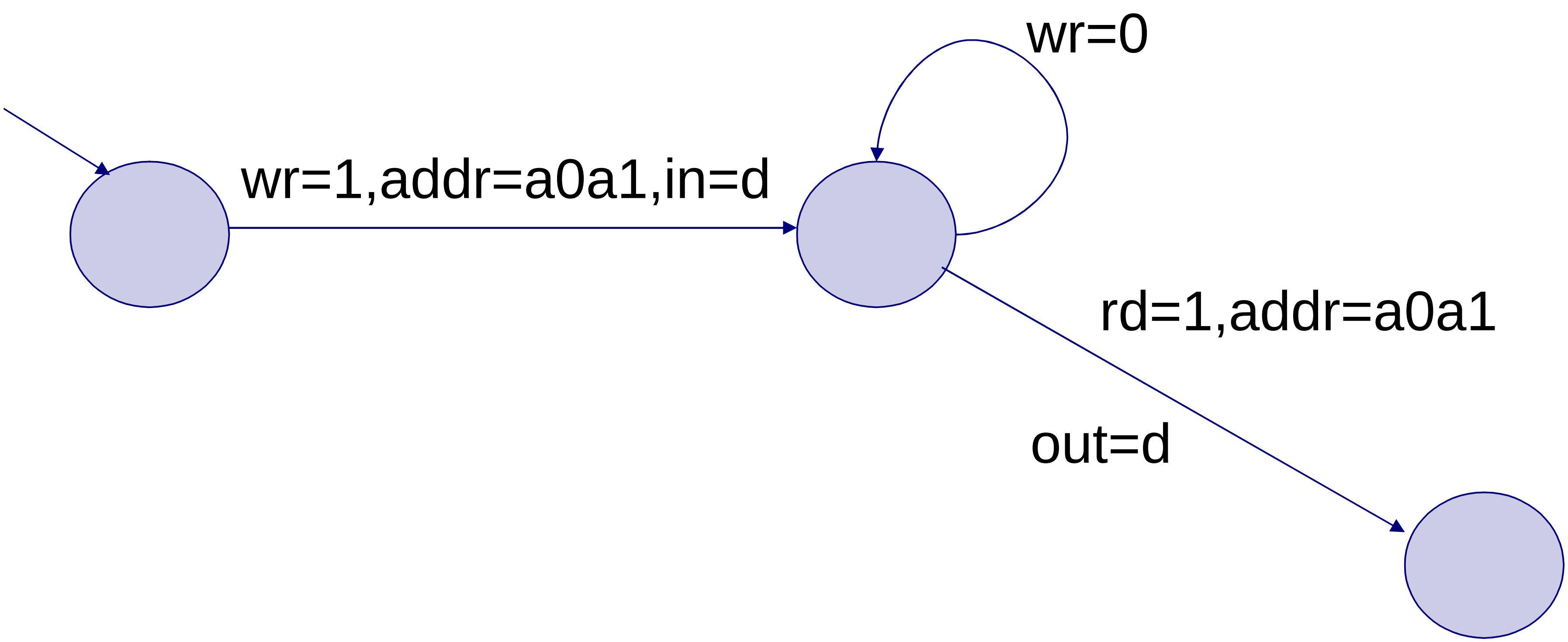
Solution 1: Induction

- B should hold for all *reachable* states
- Prove in STE:
  - I ➔ B (I characterizes the initial states)
  - B ➔ **N** B
- Conclude that B always holds
- Need theorem prover for *meta-reasoning*

vital!

Solution 2: GSTE

- *Generalized* STE
- Specification is a graph:



Active Research

- What are the right algorithms for (G)STE?
  - BDD-based
  - SAT-based
- What is the right semantics for GSTE?
- A logic for GSTE specifications
  - Melham (Oxford)
- (G)STE refinement?
  - Automatic
  - Semi-automatic