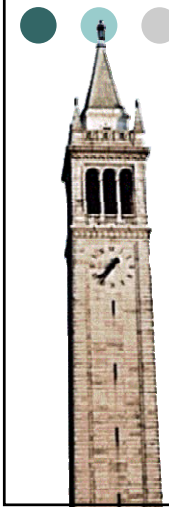


# Fundamental Algorithms for System Modeling, Analysis, and Optimization

Edward A. Lee, Jaijeet Roychowdhury, Sanjit A. Seshia
UC Berkeley
EECS 144/244
Fall 2011



**Sequential Equivalence Checking**

Thanks to Kurt Keutzer and Rob Rutenbar for several slides

## Lecture Outline

**What we know:**

- How to check two combinational circuits for equivalence

**What we need:**

- Checking equivalence of sequential circuits
- E.g., a circuit and its retimed version

**Today's lecture is about using Boolean function manipulation & BDDs for doing this**

- Basics
- Sequential equivalence checking: the problem
- Algorithms

## Two Operations on Cofactors

Given: $F(x_1, \ldots, x_n)$

Define

$C(x_2, \ldots, x_n) = F_{x_1} \cdot F_{x_1'}$ ← "Consensus"

$S(x_2, \ldots, x_n) = F_{x_1} + F_{x_1'}$ ← "Smoothing"

What are the ON-sets of C and S in terms of the ON-sets of $F_{x_1}$ and $F_{x_1'}$ ?



## Example

$F(a,b,c) = ab + bc + ac$

$F_a = b + c$

$F_{a'} = bc$

$C(b,c) = ?$

$S(b,c) = ?$

# Quantification

**Consensus also called "universal quantification"**

- $C(x_2, \ldots, x_n) = F_{x_1} \cdot F_{x_1'}$
  $= \forall x_1 F(x_1, x_2, \ldots, x_n)$ ("for all $x_1$ …")

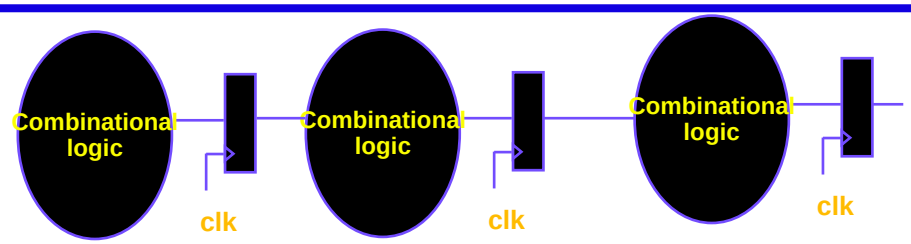
**Smoothing also called "existential quantification"**

- $S(x_2, \ldots, x_n) = F_{x_1} + F_{x_1'}$
  $= \exists x_1 F(x_1, x_2, \ldots, x_n)$ ("there exists $x_1$ …")

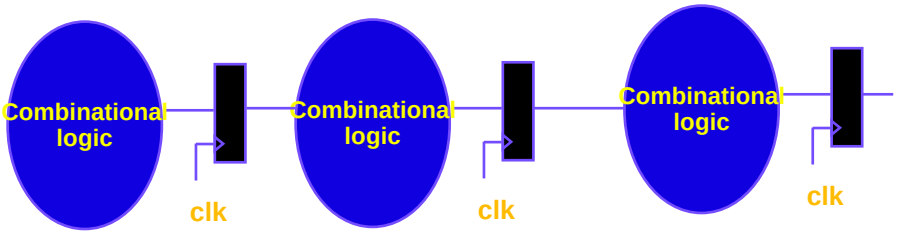


**Back to Equivalence Checking . . .**

# Equivalence Checking: Simple Case



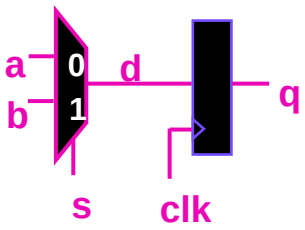

``specification''

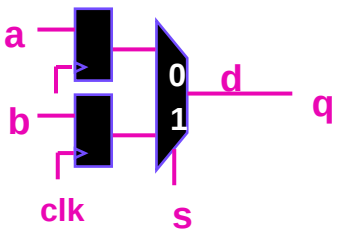



implementation



# Retimed circuits




**Circuits are equivalent but it is not possible to show that they are equivalent using Boolean equivalence**

## Encoding Problems

Some circuit specifications are "symbolic" rather than binary-valued

e.g. specification for an ALU

| Symbol | Operation |
|---|---|
| ADD | + |
| SUB | - |
| XOR | Exclusive-OR |
| INC | Increment |

Can assign any binary op code to the symbolic values, so long as they are different



## Different State Encodings

Circuit 1

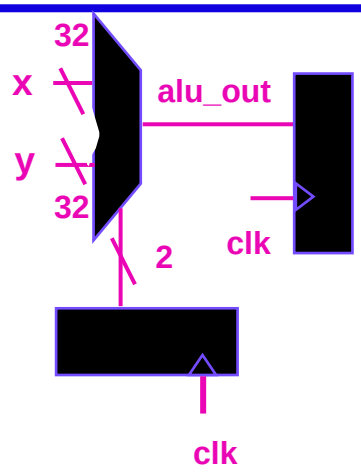
| Symbol | Operation |
|---|---|
| ADD | 00 |
| SUB | 01 |
| XOR | 10 |
| INC | 11 |

Circuit 2

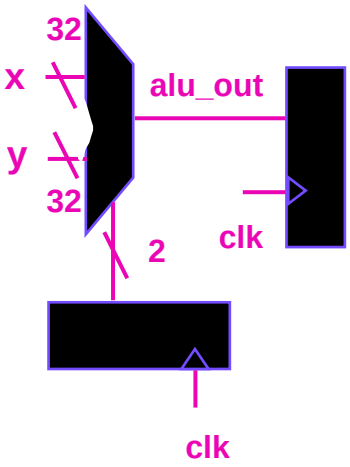
| Symbol | Operation |
|---|---|
| ADD | 11 |
| SUB | 10 |
| XOR | 00 |
| INC | 01 |

Different state encodings make circuits no longer amenable to combinational logic equivalence checking

# Different Encodings




ALU ``ADD''s on 00




ALU ``ADD''s on 11



# A Fresh Look at Equivalence Checking

Given: Two sequential circuits, with same inputs and outputs

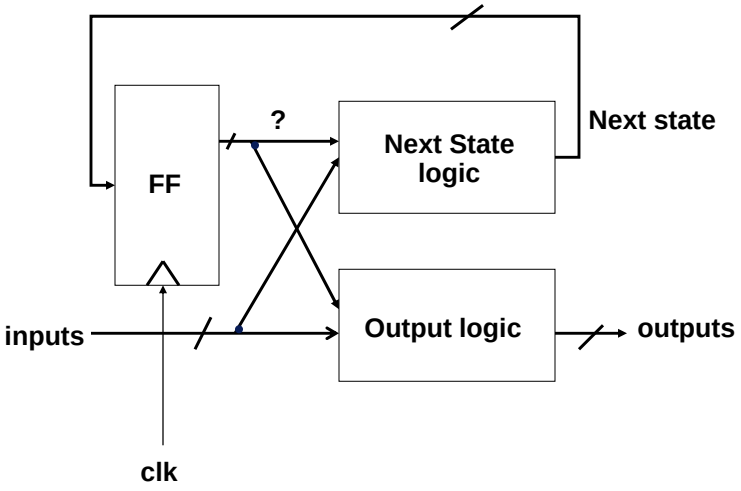
- But state bits might differ

Let's view this problem mathematically ("formally"):

A combinational circuit is a Boolean function.

A sequential circuit is a _____ ?

# What's in a Finite-State Machine (FSM) ?






# Finite-state machine (FSM) Equivalence

**Equivalence checking problem:**

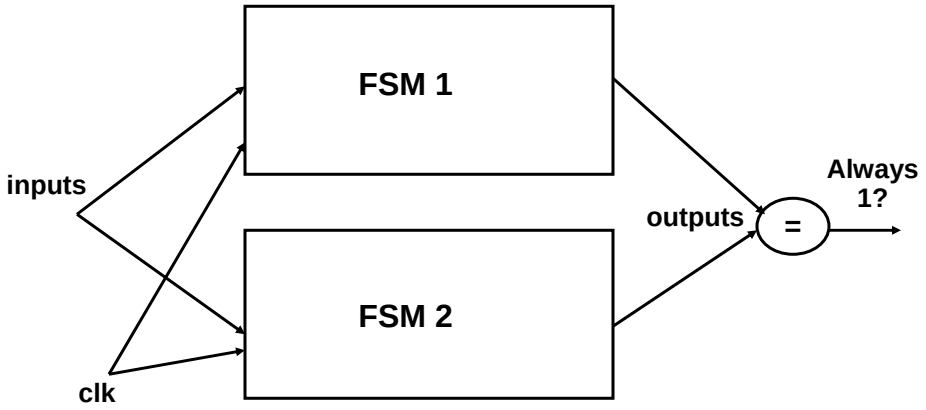
**Given: 2 FSMs, with same inputs/outputs**

**To check:**

**The output behavior of both machines is identical**

- **over all time points, starting from a common "initial" / "reset" state**
- **for every sequence of inputs**

# Visualizing the Problem: Compose FSMs




Q1. What goes inside the boxes?
Q2. How can we decide if the output is always 1?



# What goes in the boxes

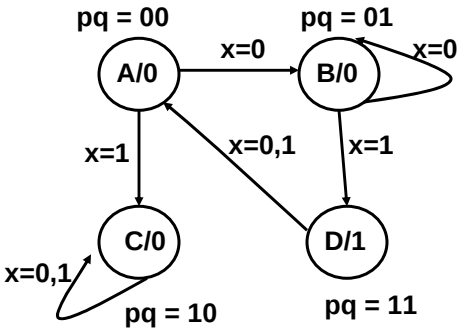
From the finite-state machine description, we write Boolean equations that describe

Next state as a function of present state & inputs

Output as a function of present state & inputs

Often this is how the system is most easily described

# Example: FSM1




Denote next state encoding as $p^+q^+$ and output as z

$p^+(x, p, q) = ?$

$pq' + p'x$

$q^+(x, p, q) = ?$

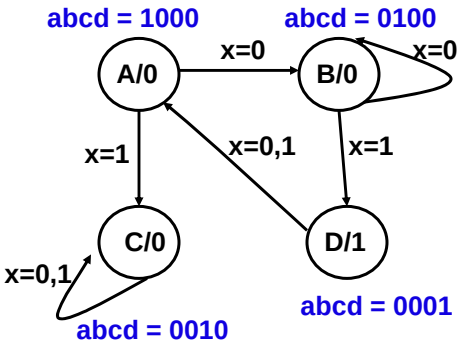
$p'x' + p'q$

$z(x, p, q) = ?$

$pq$



# Example: FSM 2 (different state encoding)




Denote next state encoding as $a^+b^+c^+d^+$ and output as z

$a^+(x, a, b, c, d) = ?$

$b^+(x, a, b, c, d) = ?$

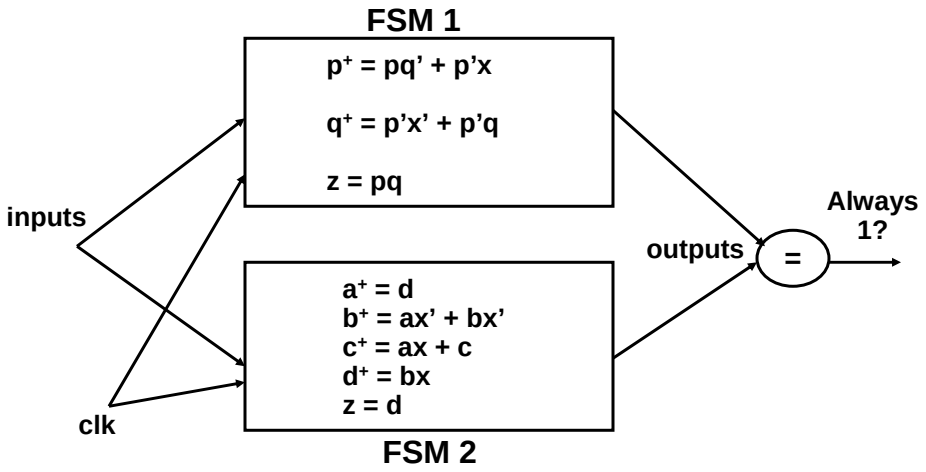
$c^+(x, a, b, c, d) = ?$

$d^+(x, a, b, c, d) = ?$

NOTE: We never start with a state graph like the one above – WHY?

$z(x, a, b, c, d) = d$

# Back to the Problem




Q1. What goes inside the boxes? ✓

Q2. How can we decide if the output is always 1?



# Rephrasing the Problem

Is the output always 1?

Can the output ever be 0?

Solved using "reachability analysis"

- Is there a state that the combined FSM can reach such that the output is 0?

# Performing Reachability Analysis

3 Main ideas:

Represent sets as Boolean functions

- Use BDDs to represent Boolean functions

Represent FSMs "symbolically"

- FSM = set of states and set of transitions
- FSM can be encoded using BDDs

Perform Symbolic Reachability Analysis

- Start in initial state
- Compute set of states reachable from initial state in 1, 2, 3, … clock ticks
- This computation must terminate – WHY?

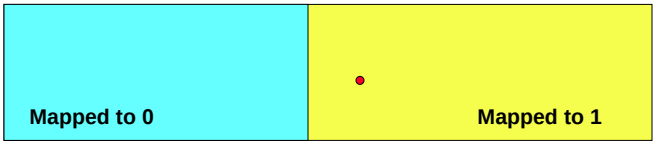


# 1. Sets as Boolean functions

A Boolean function F of n variables x1, x2, …, xn

$$F : \{0,1\}^n \rightarrow \{0,1\}$$

can be represented as set (its ON-set)




Similarly, for a set of size <= $2^n$, you can encode each element as a string of <= n bits

Each string can be viewed as a minterm

View the set as the ON-SET of a Boolean function

## Set Operations as Boolean Operations

$A \cup B = ?$

$A \cap B = ?$

$A \subset B$ ?

Is A empty?



## 2. Symbolic Encoding of FSM

FSM is

Set of states

- State encoding is a minterm
- This is what we want to compute!

Set of transitions

- To compute set of reachable states, we first need a way of encoding transitions

# Encoding Transitions

Define a new function, $\delta$, called the "transition function/relation"

$\delta$ (current state s, input x, next state $s^+$)

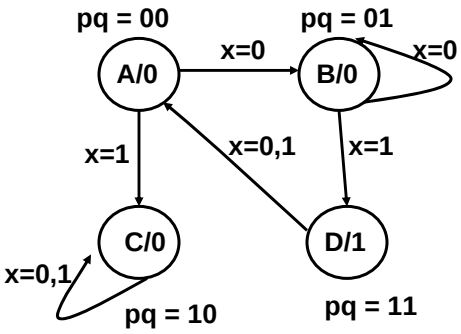
= 1 if we can go to $s^+$ from s on x

= 0 otherwise

i.e. $\delta$ encodes all legal transitions ("edges" in the state graph)



# Example of $\delta$




$p^+ = pq' + p'x$

$q^+ = p'x' + p'q$

$z = pq$

Denote next state encoding as $p^+q^+$ and output as z

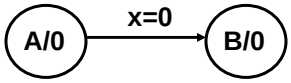
$\delta(p, q, x, p^+, q^+)$

$\delta(0, 0, 0, 0, 1) = ?$

$\delta(1, 1, 1, 1, 1) = ?$

How to construct $\delta$?

- Pick an edge & encode it




- Add a term into the SOP for $\delta$ for that edge

  $\delta = p'q'x'p^{+'}q^+ + \ldots$
- There's an easier way…

Just turn each Boolean eqn into Boolean identity (iff), and AND together the identities for each state bit

# 3. Reachability Analysis

**Given:**

**A minterm corresponding to initial state $R_0$**

$\delta$

**To find:**

**All states reachable from $R_0$ in 1, 2, 3, … clock ticks**

**Idea:**

**Enumerate all states by repeatedly evaluating FSM equations, starting from an initial state?**



# 3. Reachability Analysis

**Given:**

**A minterm corresponding to initial state $R_0$**

$\delta$

**To find:**

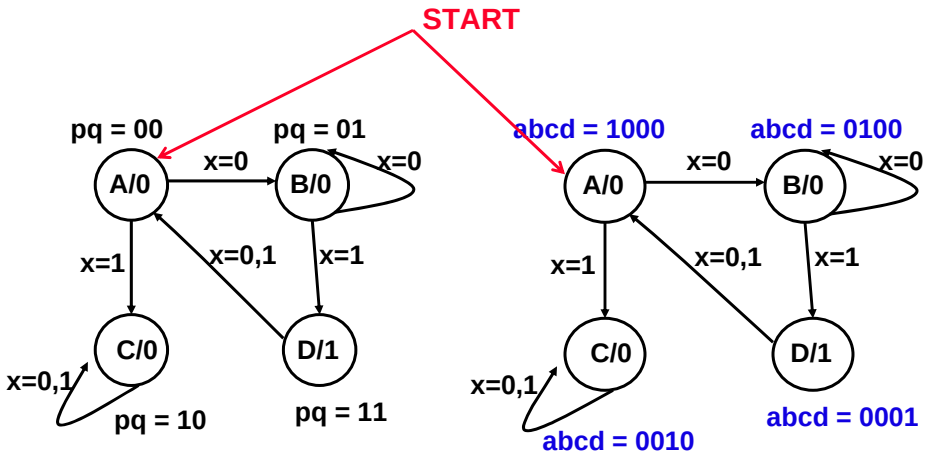
**All states reachable from $R_0$ in 1, 2, 3, … clock ticks**

**Strategy: Denote set of states reachable from $R_0$ in k (or less) clock ticks as $R_k$**

**Express $R_k$ as a function of $R_{k-1}$ and $\delta$ and solve recurrence relation**

- **Remember: Every set is represented as a Boolean function (BDD)**

# What's the initial state?




Start state for combined FSM is p'q'ab'c'd' = $R_0$

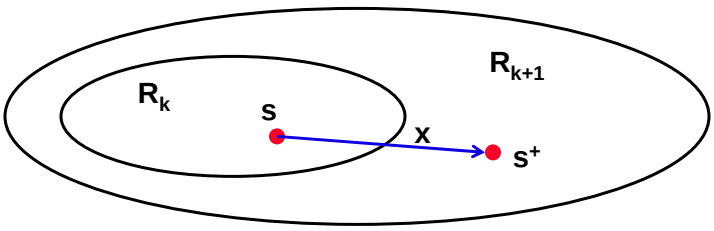


# Computing $R_k$, k >= 1

How do we define $R_{k+1}$ in terms of $R_k$ and $\delta$?

(think in terms of sets)

– Is $R_k$ inside $R_{k+1}$ ?



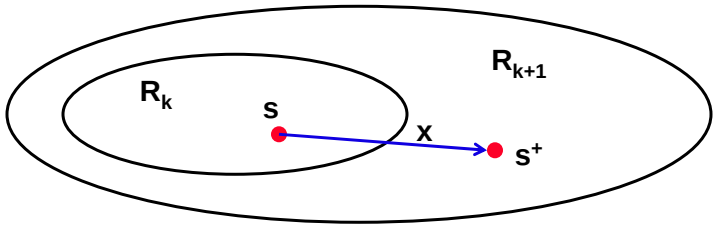
# Computing $R_k$, k >= 1

Desribing the edge in math notation:

$s \in R_k$

$s^+ \in R_{k+1}$

$\delta(s, x, s^+) = 1$






# Looking at it another way…

Suppose I gave you a $s^+$ and asked you whether it was in $R_{k+1}$, i.e.: When is $R_{k+1}(s^+) = 1$? What are the cases?

Answer this question in terms of $R_k$ and $\delta$? (in English)

Either
1. $s^+$ is in $R_k$, i.e., $R_k(s^+) = 1$
Or
2.
There exist current state s and input x such that:
- $R_k(s) = 1$
- $\delta(s, x, s^+) = 1$

## Writing out an equation for $R_{k+1}$

$$R_{k+1}(s^+) = R_k(s^+) + \exists\, s, x \{ R_k(s) . \delta(s, x, s^+) \}$$

**Either**
**1. $s^+$ is in $R_k$, i.e., $R_k(s^+) = 1$**
**Or**
**2.**
**There exist current state s and input x such that:**

- **$R_k(s) = 1$**
- **$\delta(s, x, s^+) = 1$**



## Computing $R_k$

**Start with $R_0$**

**Repeatedly compute $R_{k+1}$ as:**

$$R_{k+1}(s^+) = R_k(s^+) + \exists\, s, x \{ R_k(s) . \delta(s, x, s^+) \}$$

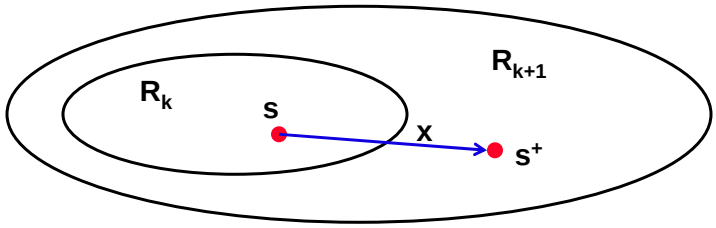
**Note: everything is represented as a Boolean function**

**When do we stop?**

# Termination

When $R_k$ and $R_{k+1}$ are the same

Why is this guaranteed to happen?






# Recap of Reachability Analysis

- Compute start state $R_0$
- Compute expression for $\delta$
- Repeatedly compute $R_k$ until termination criterion is true
- Resulting $R_k$ for largest k is the set of all states reachable from $R_0$

## Sequential Equivalence Checking

- Connect the two FSMs to form combined FSM
- Compute combined start state $R_0$
- Compute expression for $\delta$
- Repeatedly compute $R_k$ for increasing k until termination criterion is true
- The $R_k$ for largest k is the set of all states reachable from $R_0$
- Check if any of these states can generate output 0 (showing that the two FSM outputs are different) → HOW?



## Sequential Equivalence Checking

- Connect the two FSMs to form combined FSM
- Compute combined start state $R_0$
- Compute expression for $\delta$
- Repeatedly compute $R_k$ for increasing k until termination criterion is true
- The $R_k$ for largest k is the set of all states reachable from $R_0$
- Check if any of these states can generate output 0 (showing that the two FSM outputs are different) → Check if the Boolean function $R_k(s) \cdot \overline{z(s)}$ is satisfiable, satisfying assignment is the state in which outputs differ

# Summary

- Sequential equivalence checking can be done using FSM reachability analysis
- In practice, very computationally intensive
  - Memory intensive: BDDs can grow quite big
- Currently limited to a few hundred to few thousand state bits
- Scaling this up is an active area of research
  - New techniques based on SAT solving are available: e.g., the ABC system developed in Prof. Brayton's group at Berkeley