Equivalence Checking of Sequential Circuits

Sanjit Seshia

EECS

UC Berkeley

With thanks to K. Keutzer, R. Rutenbar

Today's Lecture

- 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

Recap: Cofactors

A Boolean function F of n variables $x_1, x_2, \ldots, x_n$

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

Cofactors of F:

$F_{x_1}(x_2, \ldots, x_n) = ?$

$F_{x_1'}(x_2, \ldots, x_n) = ?$

S. Seshia 3

Two Operations on Cofactors

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

Define

1. $C(x_2, \ldots, x_n) = F_{x_1} \cdot F_{x_1'}$ ← "Consensus"
2. $S(x_2, \ldots, x_n) = F_{x_1} + F_{x_1'}$ ← "Smoothing"

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

S. Seshia 4

Example

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

$F_a = b + c$

$F_{a'} = bc$

$C(b,c) = ?$

$S(b,c) = ?$

S. Seshia

5

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 …")

S. Seshia

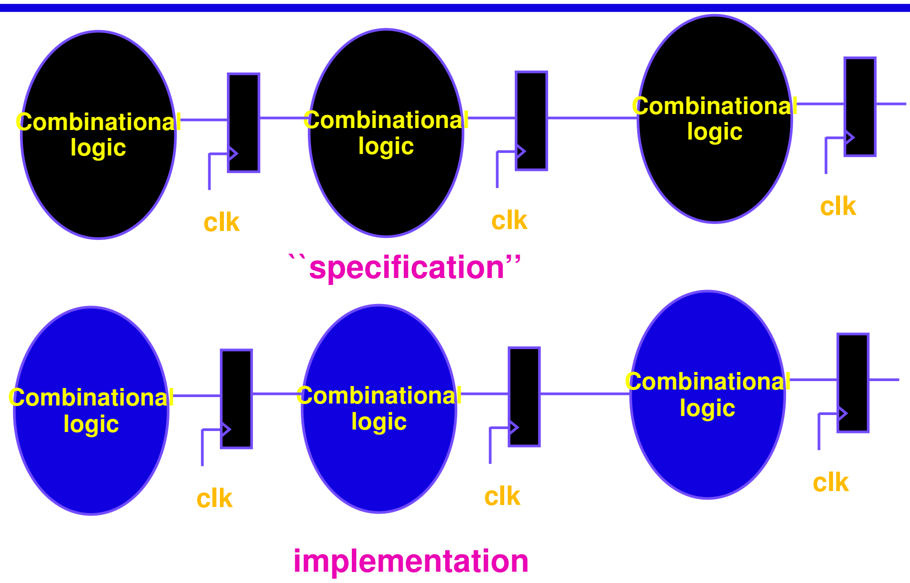
6

Back to Equivalence Checking . . .

S. Seshia

7

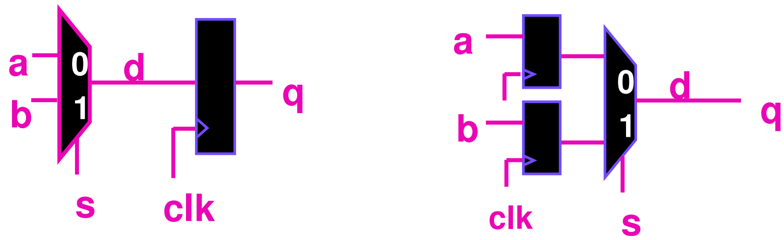
Equivalence Checking: Simple Case



S. Seshia

8

Retimed circuits



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

Encoding Problems

- Some logic 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

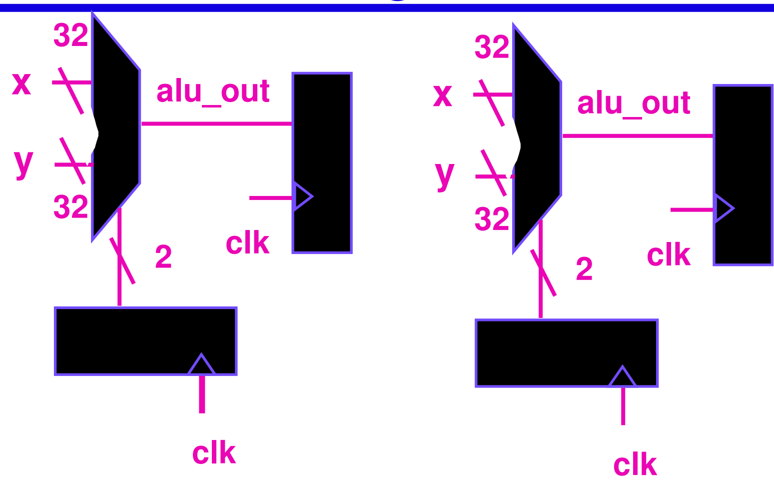
Symbol	Operation
ADD	00
SUB	01
XOR	10
INC	11

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

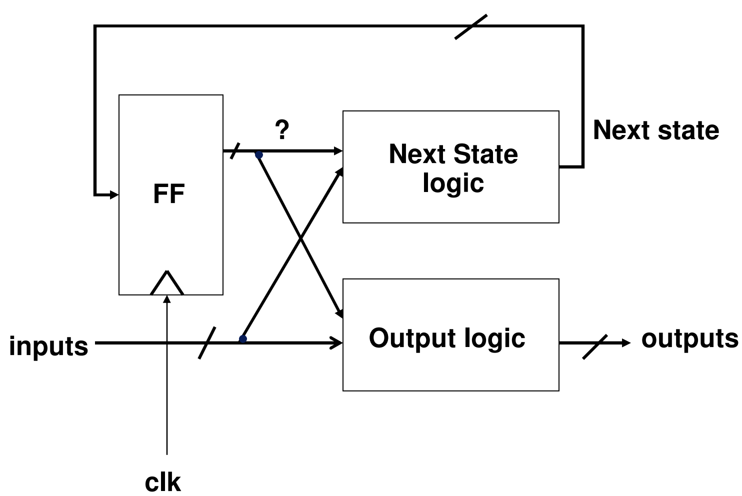
Let's view this problem mathematically ("formally"):

A combinational circuit is a Boolean function.

A sequential circuit is a ____________

S. Seshia 13

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



S. Seshia 14

Finite-state machine (FSM) Equivalence

Equivalence checking problem:

Given: 2 FSMs, with same inputs/outputs

To check:

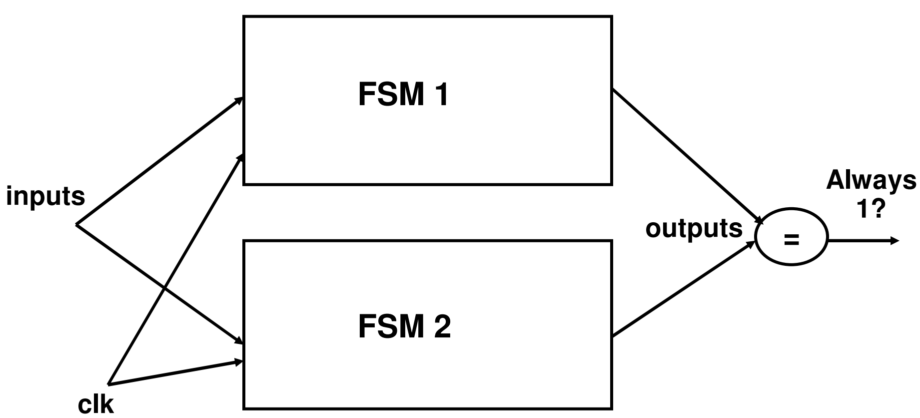
The output behavior of both machines is identical

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

S. Seshia

15

Visualizing the Problem



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

S. Seshia

16

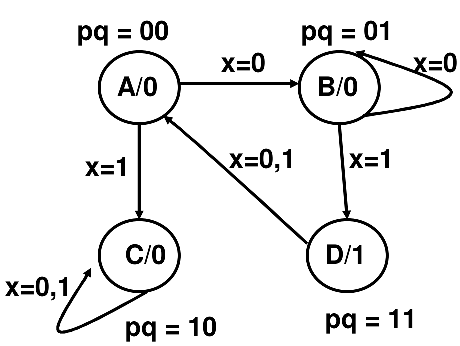
What goes in the boxes

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

1. Next state as a function of present state & inputs
2. Output as a function of present state & inputs

- Most 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

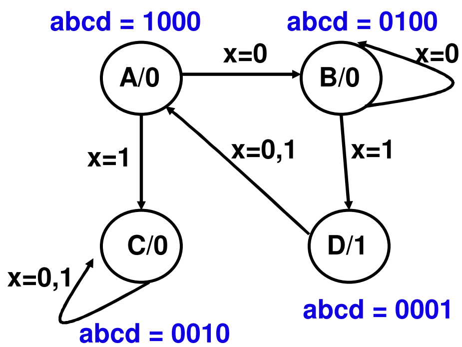
$q^+(x, p, q)$ = ?

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

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

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

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

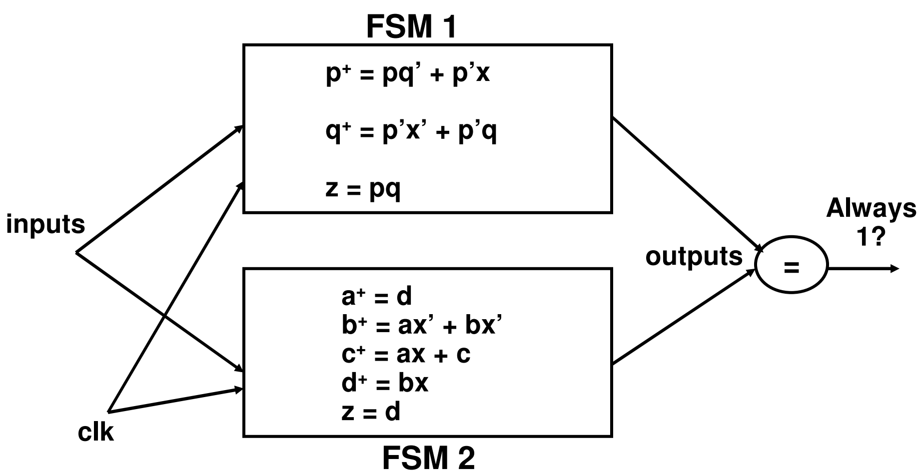
$d^+(x, a, b, c, d) =$?

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

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

S. Seshia

Back to the Problem



Q1. What goes inside the boxes? ✓

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

S. Seshia

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?

S. Seshia

21

Performing Reachability Analysis

3 Main ideas:

1. Represent sets as Boolean functions
   - Use BDDs
2. Represent FSMs "symbolically"
   - FSM = set of states and set of transitions
   - FSM can be encoded using BDDs
3. 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?

S. Seshia

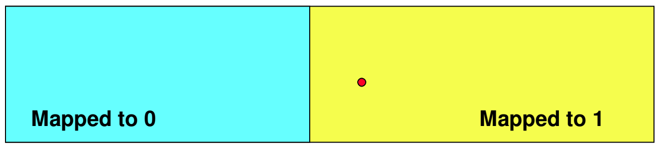
22

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



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
  - Each state 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
  - WHY NOT just enumerate all the states by repeatedly evaluating equations, starting from an initial state?

Encoding Transitions

Define a new function, δ, called the "transition relation"

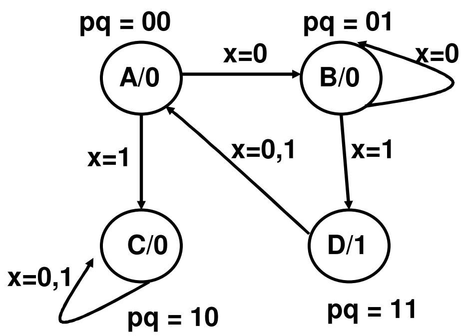
δ (current state s, input x, next state s^+)

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

= 0 otherwise

i.e. δ encodes all legal transitions ("edges" in the state graph)

Example of δ



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

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

$z = pq$

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

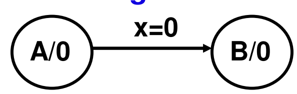
$\delta(p, q, x, p^+, q^+)$

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

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

How to construct δ?

- Pick an edge & encode it



- Add a term into the SOP for δ for that edge
  $\delta = p'q'x'p'q + \ldots$
- There's an easier way…

3. Reachability Analysis

Given:

1. A minterm corresponding to initial state R_0
2. δ

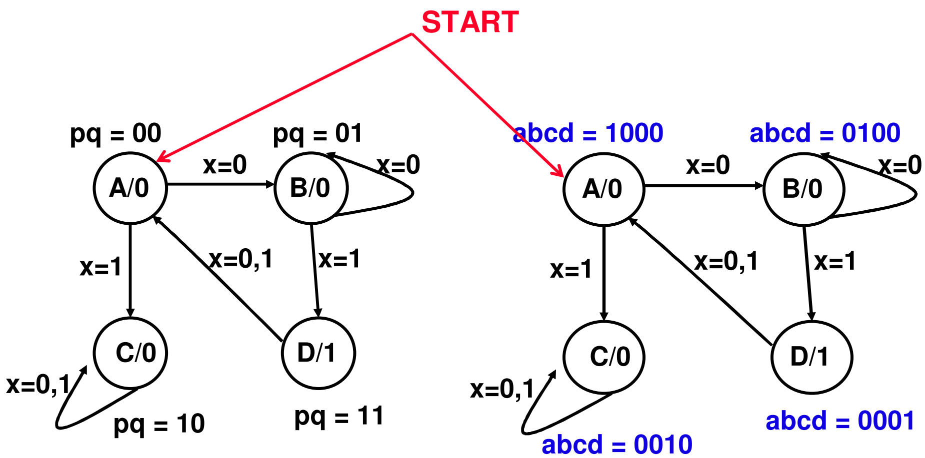
To find:

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 δ 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 pqabcd = 001000 = R_0

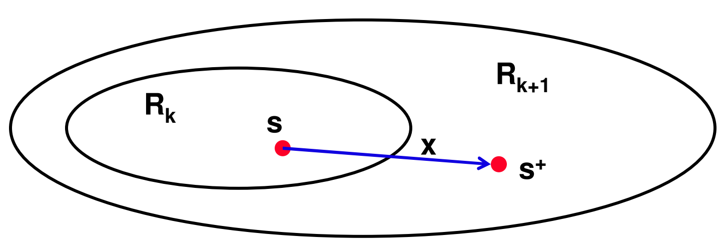
S. Seshia

29

Computing R_k, k >= 1

What’s the relation between R_k and R_{k+1}?

(think in terms of sets)



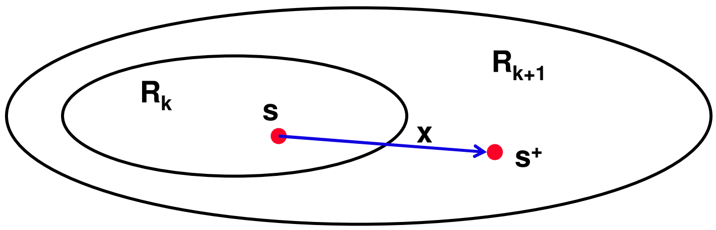
S. Seshia

30

Computing R_k, k >= 1

To get from R_k to R_{k+1} there must be some triple (s, x, s^+) such that:

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



Looking at it another way…

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

Can you phrase the answer to this question in terms of R_k and δ? (say 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:
- $R_k(s) = 1$
- $\delta(s, x, s^+) = 1$

S. Seshia

33

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?

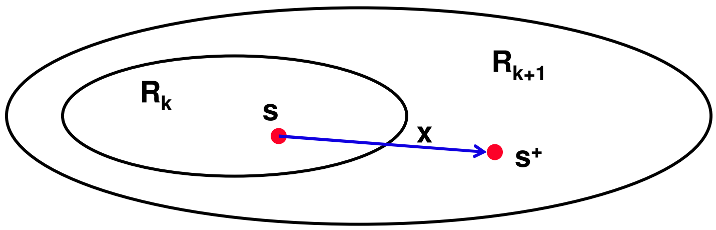
S. Seshia

34

Termination

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

Why is this guaranteed to happen?



S. Seshia

35

Recap of Reachability Analysis

1. Compute start state R_0
2. Compute expression for δ
3. Repeatedly compute R_k until termination criterion is true
4. Resulting R_k for largest k is the set of all states reachable from R_0

S. Seshia

36

Sequential Equivalence Checking

1. Connect the two FSMs to form combined FSM
2. Compute combined start state R_0
3. Compute expression for δ
4. Repeatedly compute R_k until termination criterion is true
5. Resulting R_k for largest k is the set of all states reachable from R_0
6. Check if any of these states can generate output 0 (showing that the two FSM outputs are different)

S. Seshia

37

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 state bits
- Scaling this up is an active area of research
  - New techniques based on SAT solving are available

S. Seshia

38