Sequential Circuits: Verification

Virendra Singh
Professor
Computer Architecture and Dependable Systems Lab
Department of Electrical Engineering
Indian Institute of Technology Bombay
http://www.ee.iitb.ac.in/~viren/
E-mail: viren@ee.iitb.ac.in

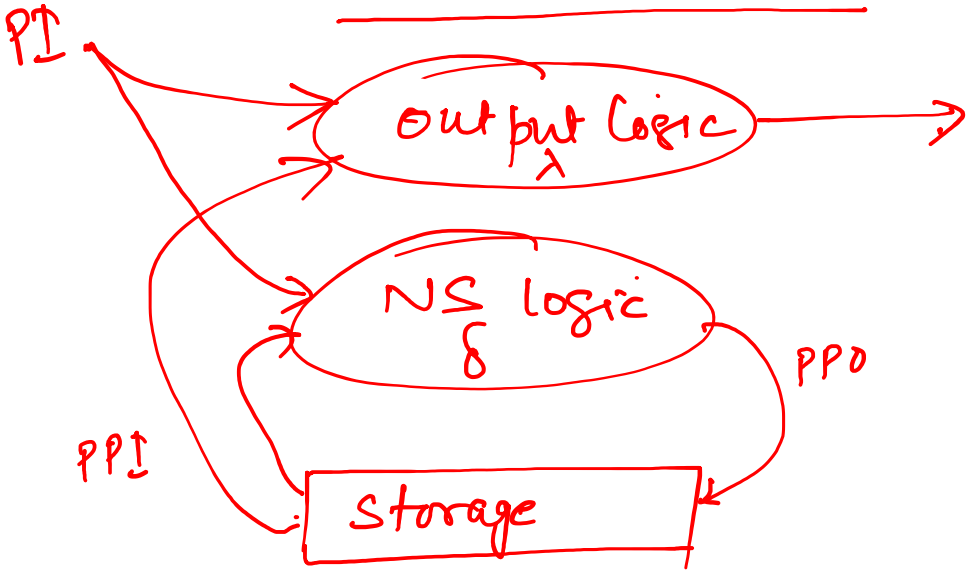
CS-226: Digital Logic Design

Lecture 25: 05 April 2021

CADSL

Finite State Machine

$M\ (I, O, S, S_0, \delta, \lambda)$ $\quad n = \log_2 |S|$



Storage

Master Slave configuration

D-FF.

How we can say that 'implemented' FSM is correct.

SPEC ≡ Implementation ?

VERIFICATION.

Finite State Machine

Combinational Circuits

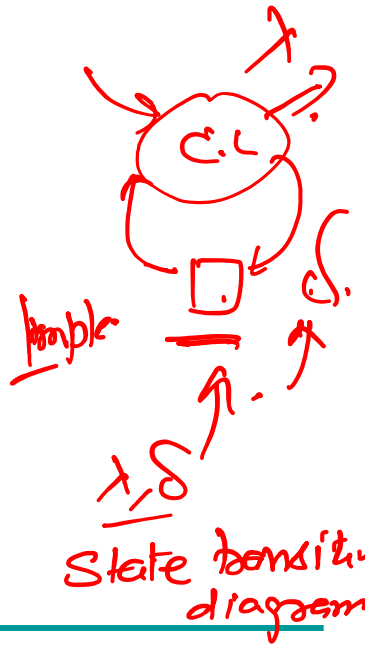
(1) EC → ROBDD
→ SAT

⇒ (2) How to check the equivalence of two machines (M_1 & M_2)

$M_1 \equiv M_2$?

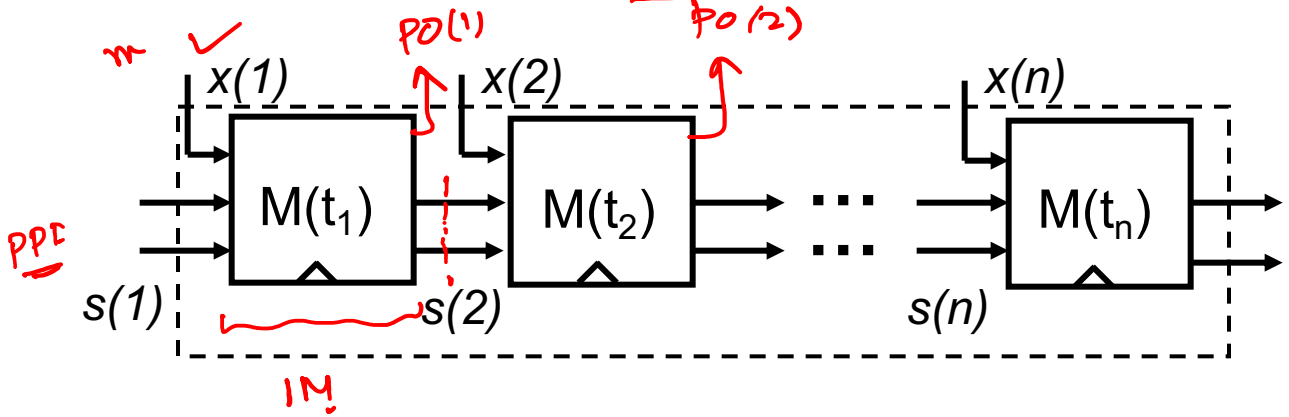
SPEC

Impl



Sequential Equivalence Checking

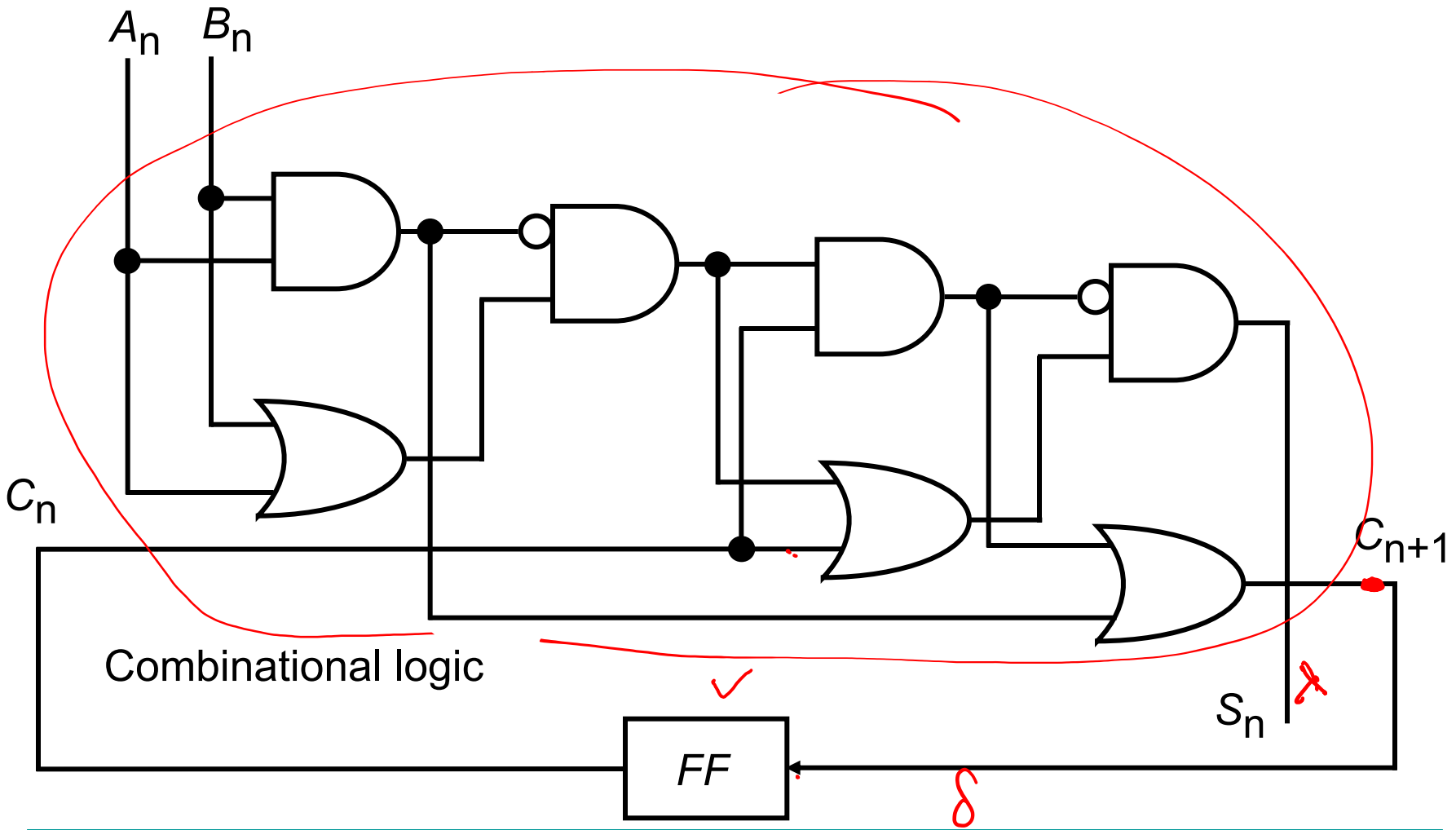
- Represent each sequential circuit as an FSM
  - verify if two FSMs are equivalent
- Approach 1: Reduction to *combinational* circuit
  - unroll FSM over *n* time frames (flatten the design)



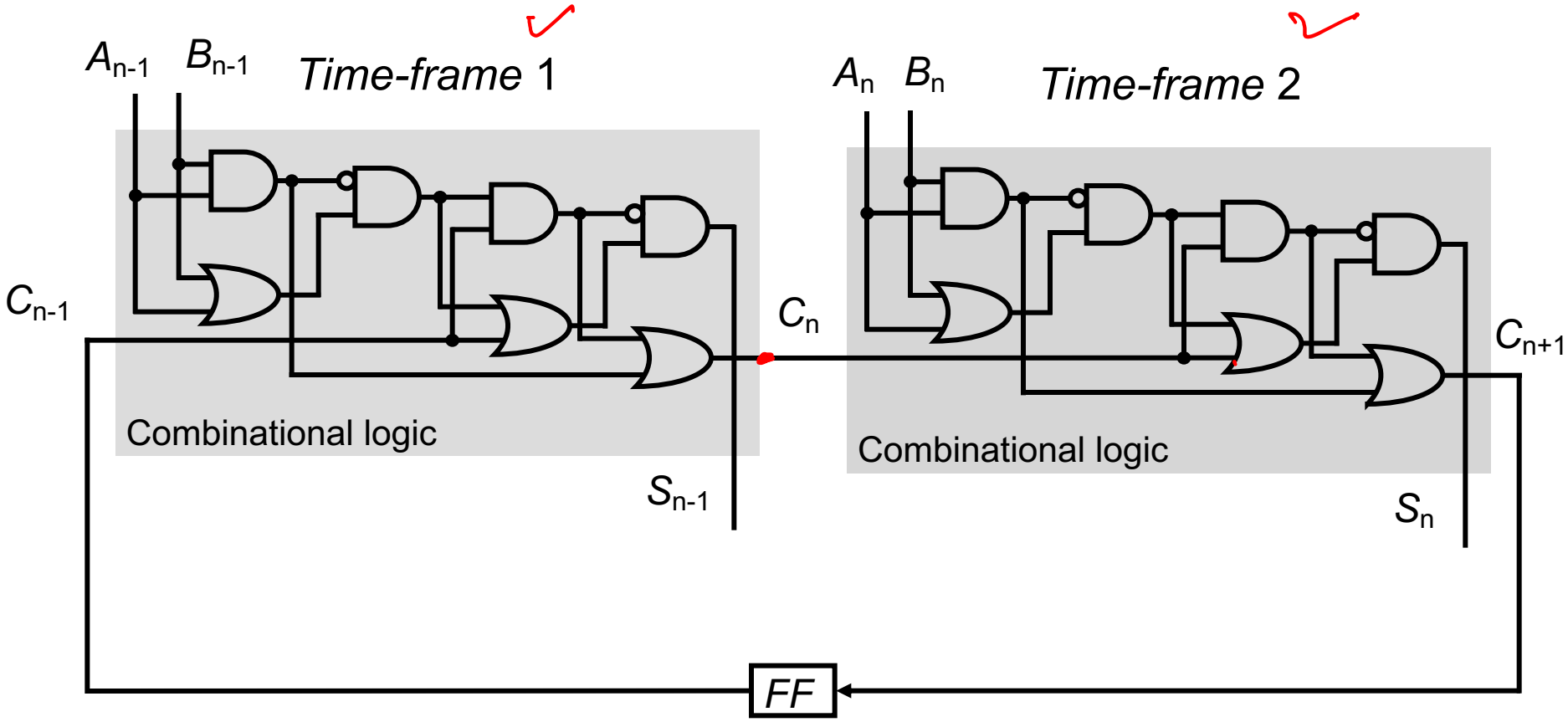
Combinational logic: *F(x(1,2, …n), s(1,2, … n))*

- check equivalence of the resulting combinational circuits
  - problem: the resulting circuit can be too large too handle

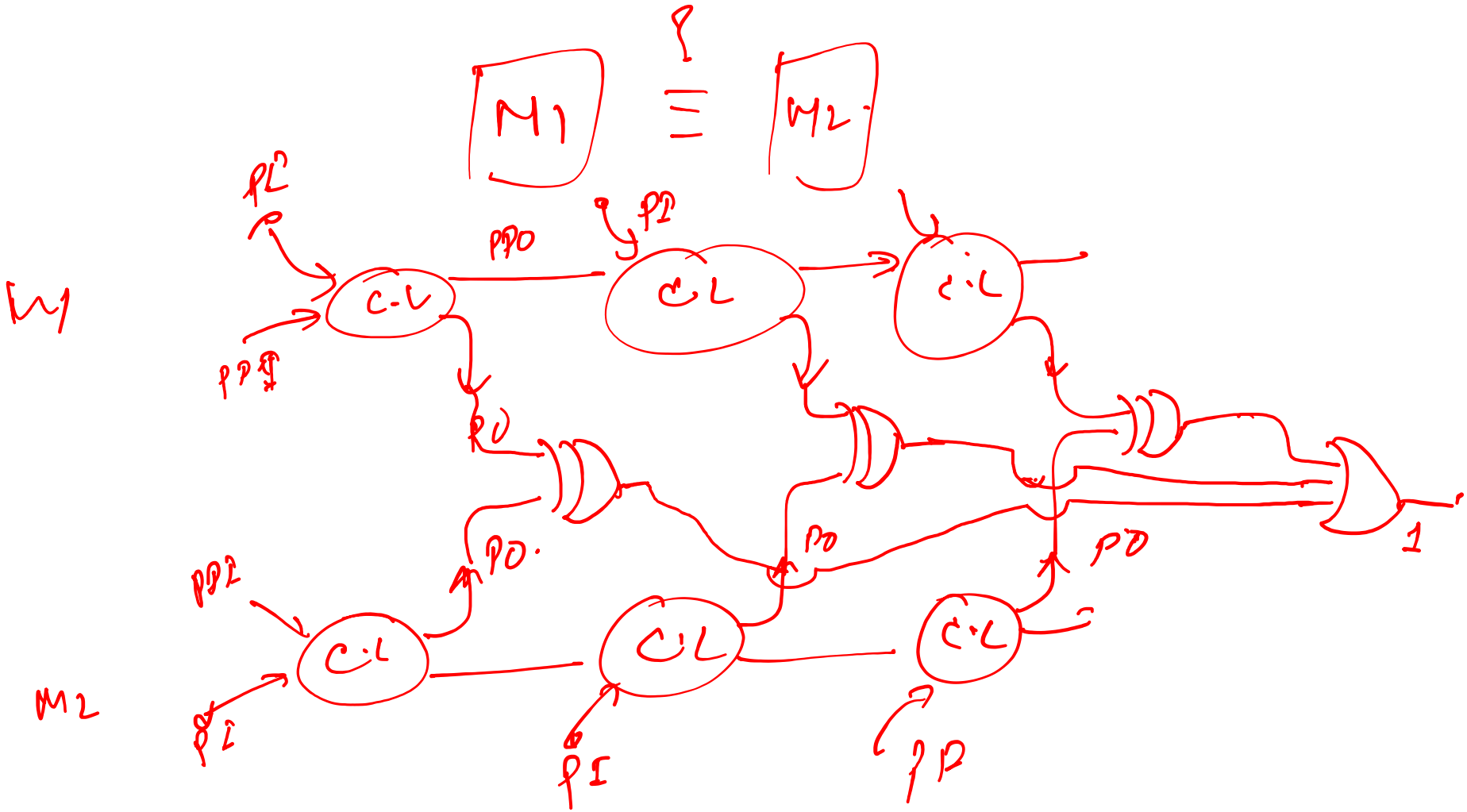
Example: A Serial Adder



Time-Frame Expansion



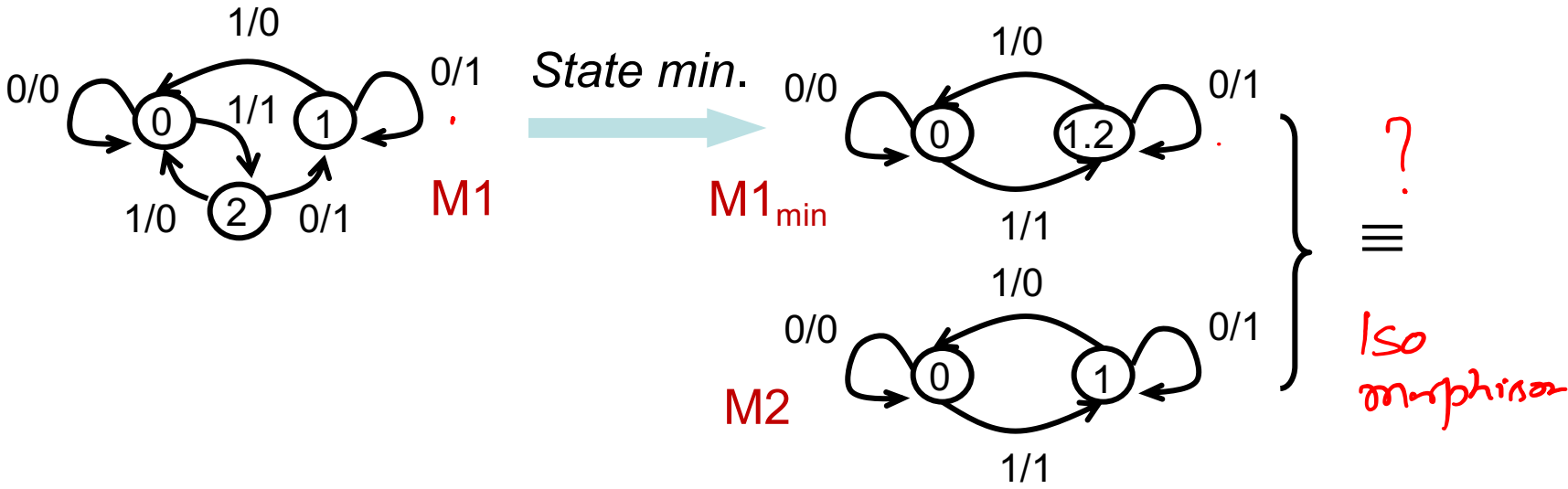
Equivalent Combinational Circuit



Sequential Verification

- Approach 2: Based on isomorphism of state transition graphs
  - two machines M1, M2 are *equivalent* if their state transition graphs (STGs) are *isomorphic*
  - perform state minimization of each machine
  - check if STG(M1) and STG(M2) are isomorphic

M1 | M2

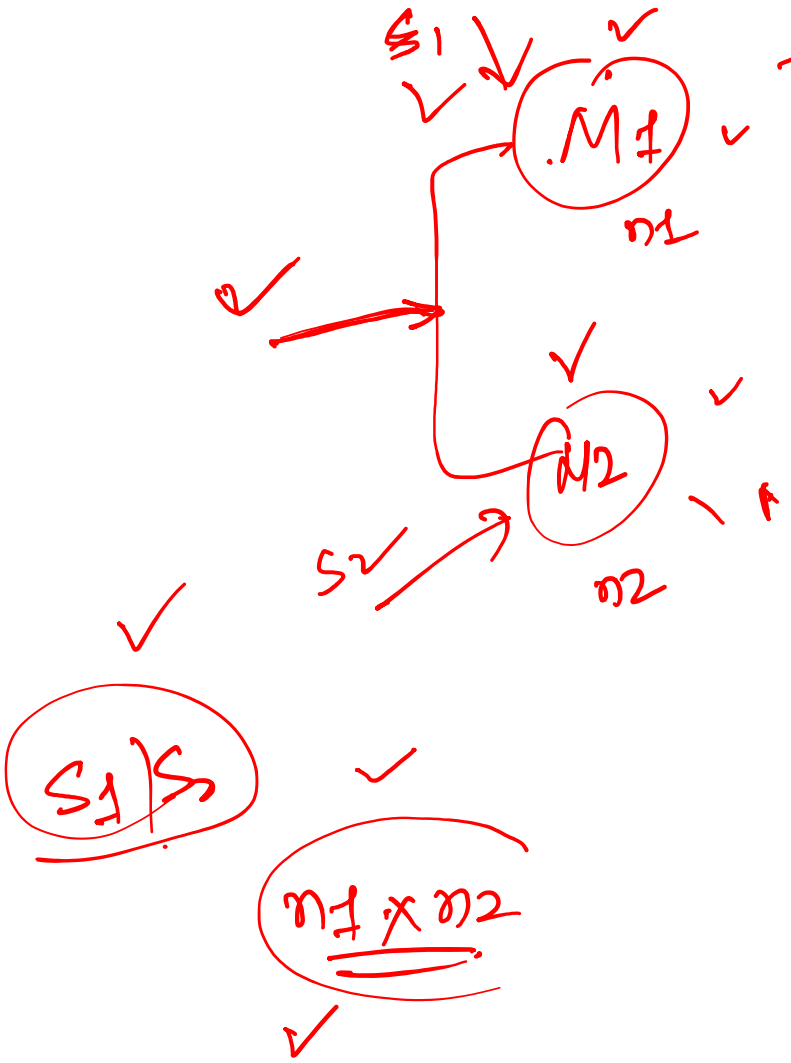


Machine Equivalence

- Two machines M1, M2 are said to be equivalent if and only if, for every state in M1, there is corresponding equivalent state in M2

- If one machine can be obtained from the other by relabeling its states they are said to be isomorphic to each other

Finite State Machine



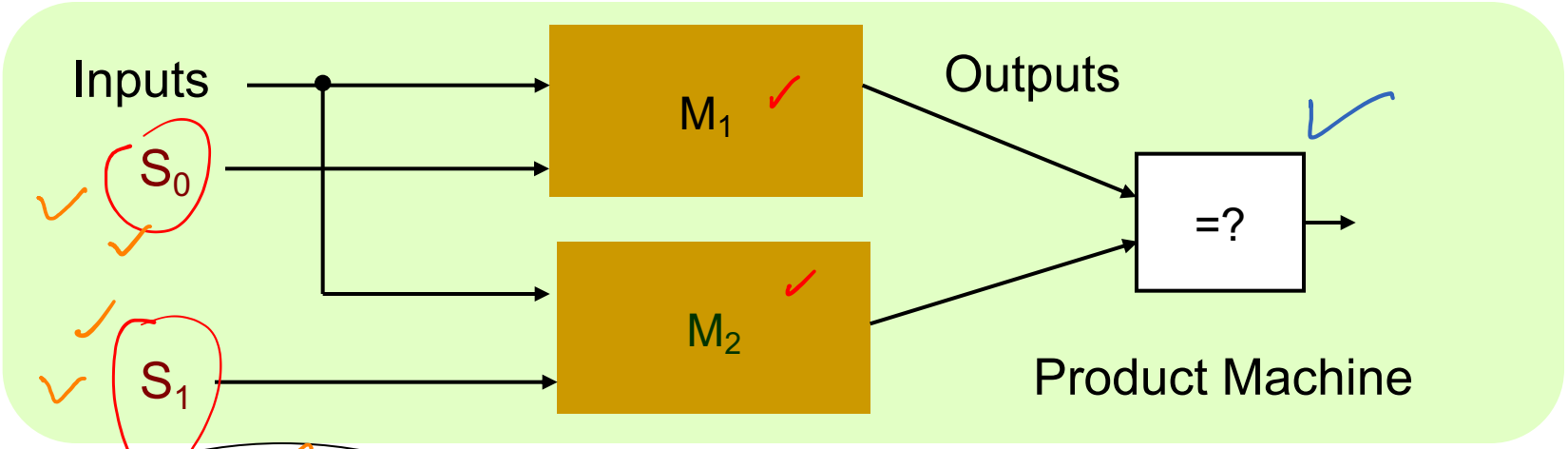
$M_1 \equiv M_2$

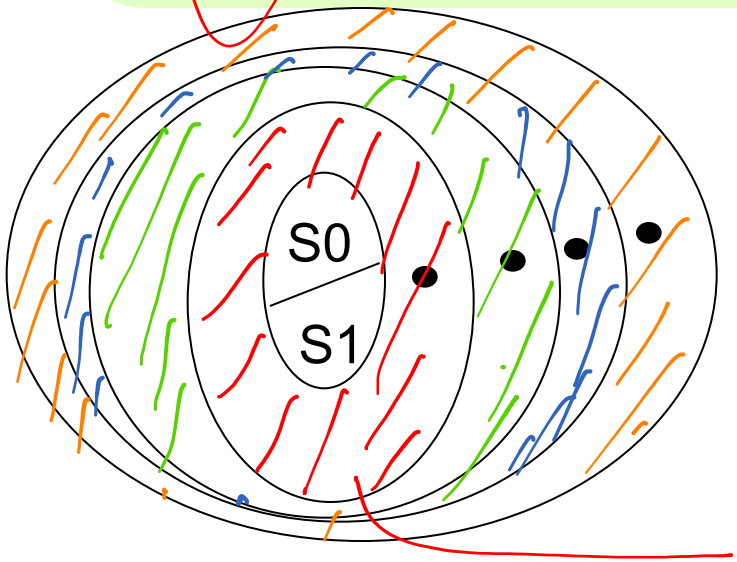
for any input sequence the machines M_1 & M_2 must produce the same output.

Product Machine

Reachability-Based Equivalence Checking

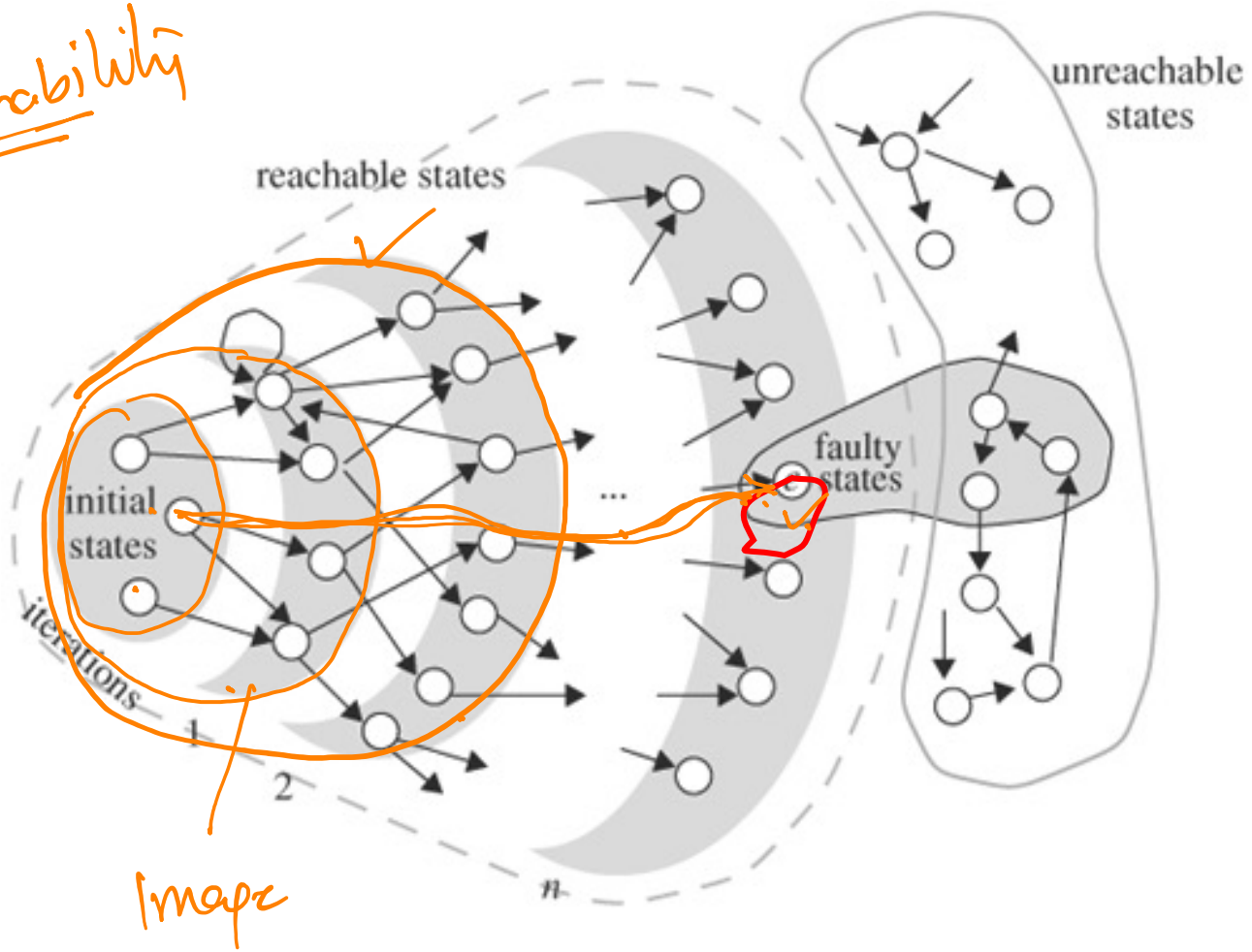
Approach 3: Symbolic Traversal Based Reachability Analysis





- Build product machine of M_1 and M_2
- Traverse state-space of product machine starting from reset states S_0, S_1
- Test equivalence of outputs in each state
- Can use any state-space traversal technique

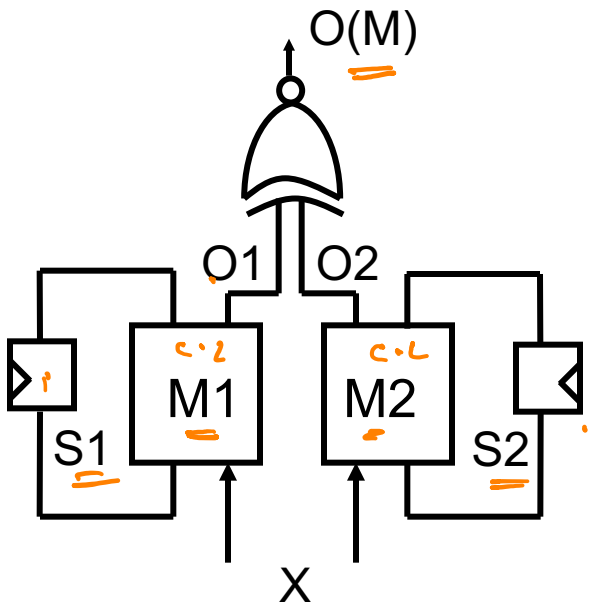
Forward Reachability



Sequential Verification

- Symbolic FSM traversal of the product machine

- Given two FSMs: $M_1(X,S_1, \delta_1, \lambda_1,O_1)$, $M_2(X,S_2, \delta_2, \lambda_2,O_2)$
- Create a product FSM: $M = M_1 \mathbf{x} M_2$
  - traverse the states of M and check its output for each transition
  - the output O(M) =1, if outputs $O_1 = O_2$
  - if all outputs of M are 1, M_1 and M_2 are *equivalent*
  - otherwise, an *error state* is reached
  - *error trace* is produced to show: $M_1 \neq M_2$

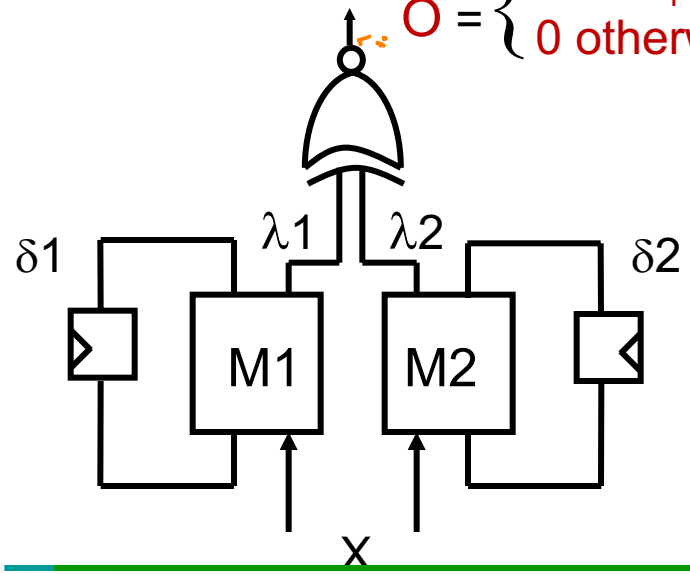


Product Machine - Construction

- Define the product machine $M(X, S, S^0, \delta, \lambda, O)$
  - states, $S = S_1 \times S_2$
  - next state function, $\delta(s, x) : (S_1 \times S_2) \times X \rightarrow (S_1 \times S_2)$
  - output function, $\lambda(s, x) : (S_1 \times S_2) \times X \rightarrow \{0,1\}$

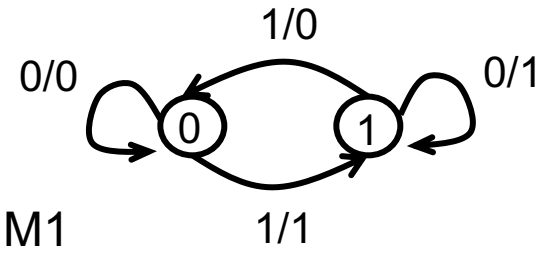
$$O = \begin{cases} 1 & \text{if } O_1 = O_2 \\ 0 & \text{otherwise} \end{cases}$$

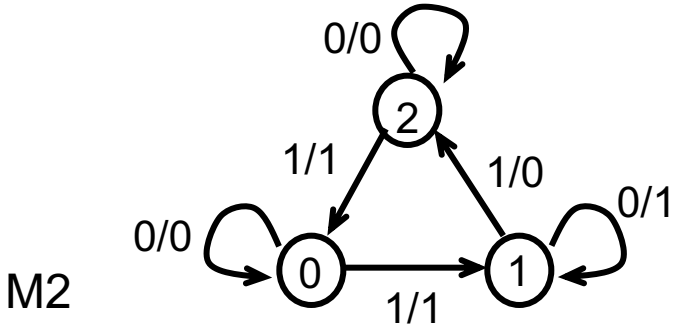
$$\lambda(s,x) = \lambda_1(s_1,x) \oplus \lambda_2(s_2,x)$$



- Error trace (*distinguishing sequence*) that leads to an error state
  - sequence of inputs which produces 1 at the output of M
  - produces a state in M for which M1 and M2 give different outputs

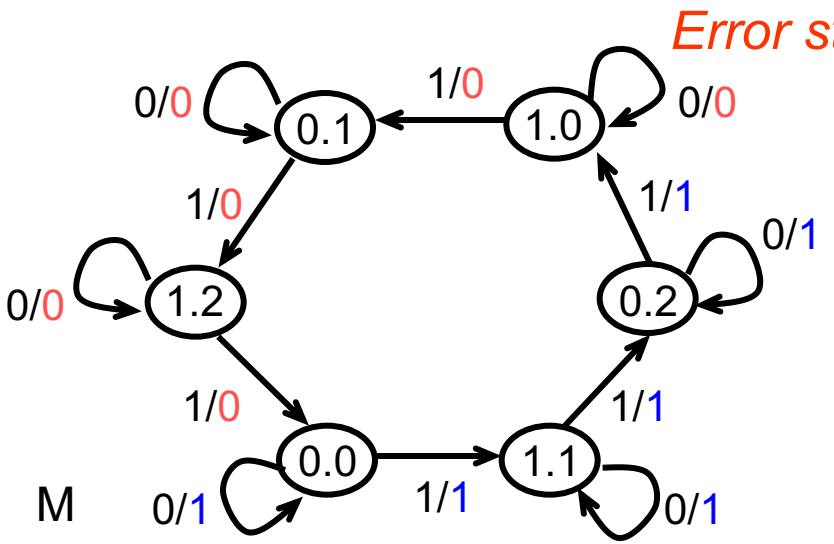
FSM Traversal in Action





Initiall states: s_1=0, s_2=0, s=(0.0)

State reached	*Out(M)* *x=0*	*x=1*
• *New* ^{0} = (0.0)	1	1
• *New* ^{1} = (1.1)	1	1
• *New* ^{2} = (0.2)	1	1
• *New* ^{3} = (1.0)	0	0



- STOP - backtrack to initial state to get *error trace: x*={1,1,1,0}

Thank You