Sequential Circuits

Equivalence of State Machines

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

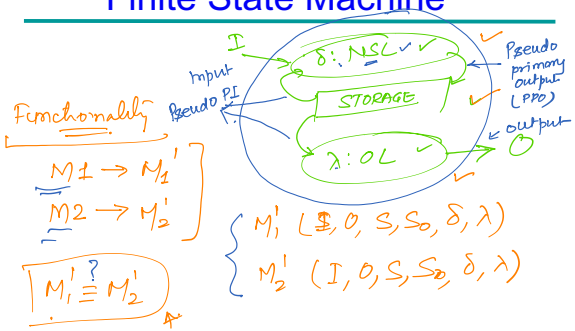
CS-230: Digital Logic Design & Computer Architecture

Lecture 19 (17 February 2022)

CADSL

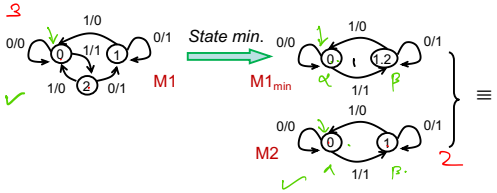
State machine
↳ temporal behaviour.
Finite State Machine (FSM)
State → STG (State Transition Graph)
Fully specified M(I, O, S, S_0, δ, λ)
Partially specified state machine: state compatibility
Minimization (automatic)
(Equivalence Checking)
Minimized State machine.

Finite State Machine



Sequential Verification

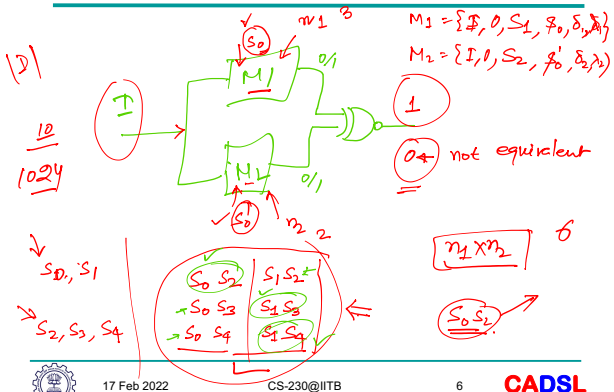
- Approach 1: Based on isomorphism of state transition graphs
  - two machines M1, M2 are *equivalent* if their state transition graphs (STGs) are *isomorphic*
  - perform state minimization of each machine
  - check if STG(M1) and STG(M2) are isomorphic



Machine Equivalence

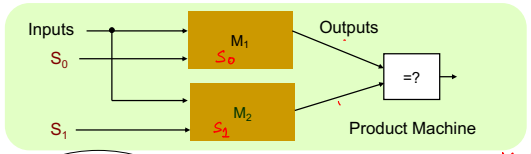
- Two machines M1, M2 are said to be equivalent if and only if, for every state in M1, there is corresponding equivalent state in M2
- If one machine can be obtained from the other by relabeling its states they are said to be isomorphic to each other

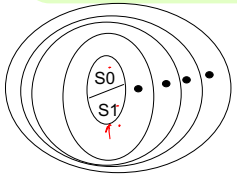
Finite State Machine



Reachability-Based Equivalence Checking

Approach 2: Symbolic Traversal Based Reachability Analysis

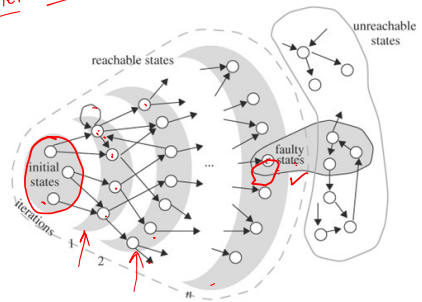




- Build product machine of M_1 and M_2
- Traverse state-space of product machine starting from reset states S_0, S_1
- Test equivalence of outputs in each state
- Can use any state-space traversal technique

Forward Reachability

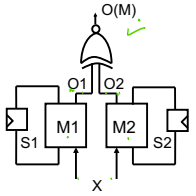
Counter example



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: $M = M_1 \times M_2$
  - 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: $M_1 \neq M_2$

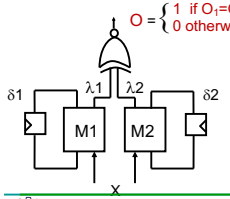


Product Machine - Construction

- Define the product machine M(X, S, S^0, δ, λ,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, $\lambda(s, x) : (S_1 \times S_2) \times X \rightarrow \{0,1\}$

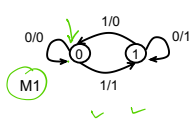
$O = \begin{cases} 1 & \text{if } O_1 = O_2 \\ 0 & \text{otherwise} \end{cases}$

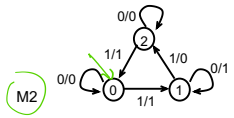
$\lambda(s,x) = \lambda_1(s_1,x) \oplus \lambda_2(s_2,x)$



- 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

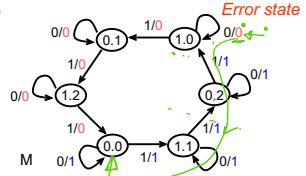




Initiall states: s_1=0, s_2=0, s=(0.0)

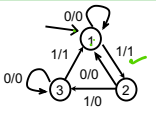
State reached	Out(M) x=0	x=1
• *New* ^{0} = (0.0)	1	1
• *New* ^{1} = (1.1)	1	1
• *New* ^{2} = (0.2)	1	1
• *New* ^{3} = (1.0)	0	0

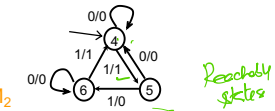
Error state



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

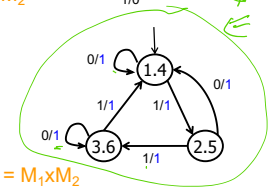
FSM Traversal in Action





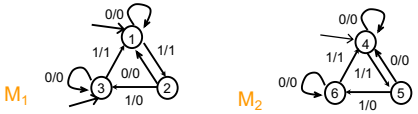
Initiall states: s_1=1, s_2=4, s=(1.4)

State reached	Out(M) x=0	x=1
• *New* ^{0} = (1.4)	1	1
• *New* ^{1} = (2.5)	1	1
• *New* ^{2} = (3.6)	1	1

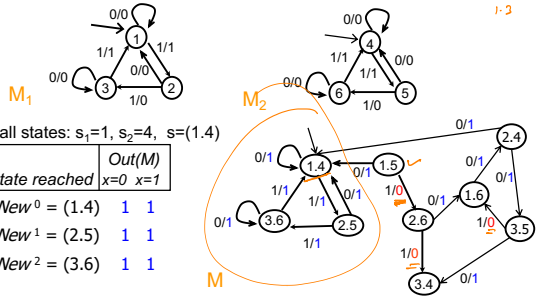


- STOP - No new reachable state

FSM Traversal in Action



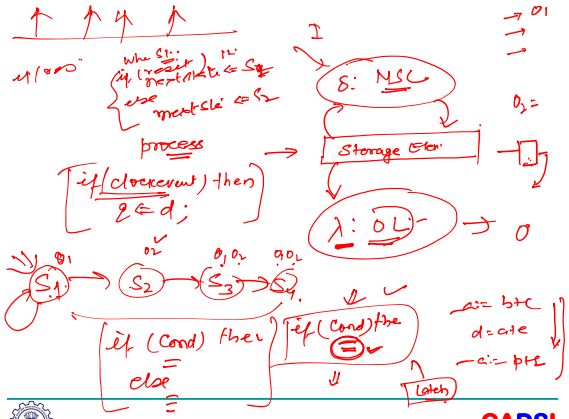
FSM Traversal in Action



Initiall states: s_1=1, s_2=4, s=(1.4)

State reached	Out(M) x=0	x=1
• *New* ^{0} = (1.4)	1	1
• *New* ^{1} = (2.5)	1	1
• *New* ^{2} = (3.6)	1	1

- Erroneous states are not reachable



FSM Traversal - Algorithm

- Traverse the product machine M(X,S,δ, λ,O)
  - start at an initial state S_0
  - iteratively compute symbolic image $Img(S_0,R)$ (set of *next states*):

$$\mathrm{Im}\, g(S_0, R) = \exists_x \exists_s S_0(s).R(x,s,t)$$

$$R = \prod_i R_i = \prod_i (t_i \equiv \delta_i(s,x))$$

until an *error state* is reached

  - transition relation R_i for each next state variable t_i can be computed as $t_i = (t \otimes \delta(s,x))$

(this is an alternative way to compute transition relation, when design is specified at gate level)

Thank You