Sequential Circuits

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

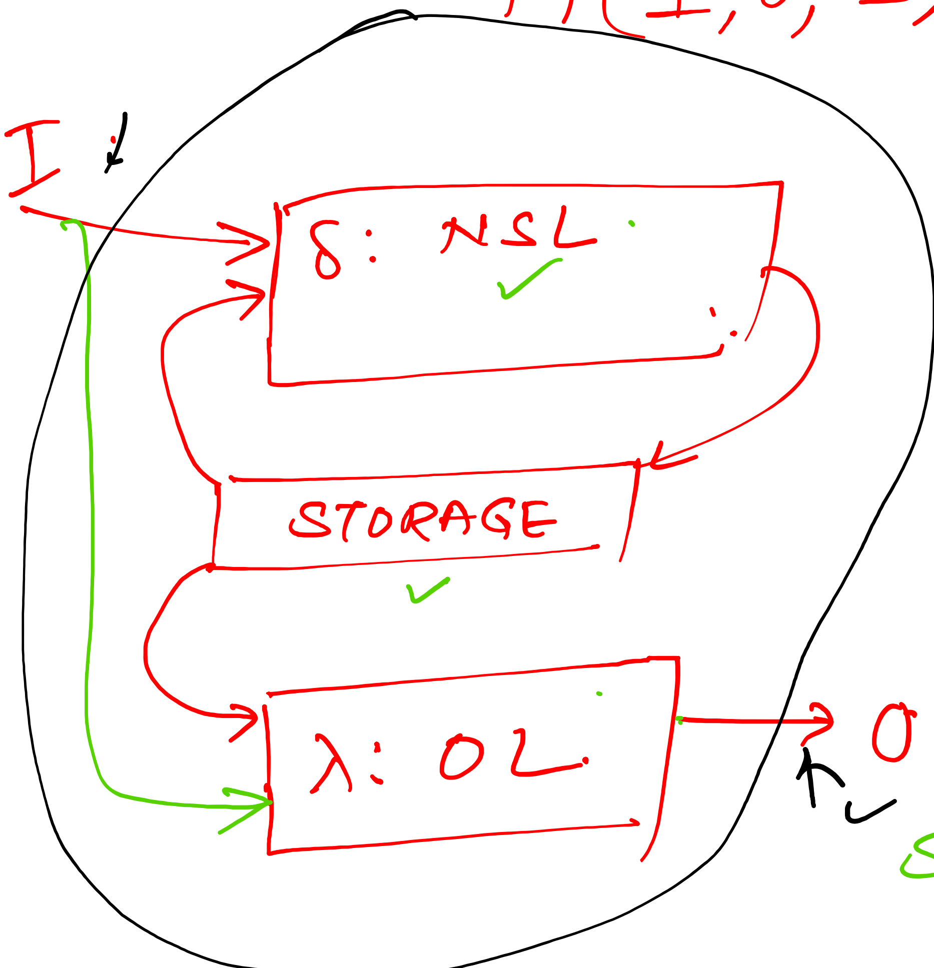
Lecture 18 (15 February 2022)

CADSL

Sequential Circuits → MEALY / MOORE

↳ temporal behaviour.

$M(I, O, S, S_0, \delta, \lambda)$



\# storage element

$= \log_2 n$.

n = # states in STG

⇓

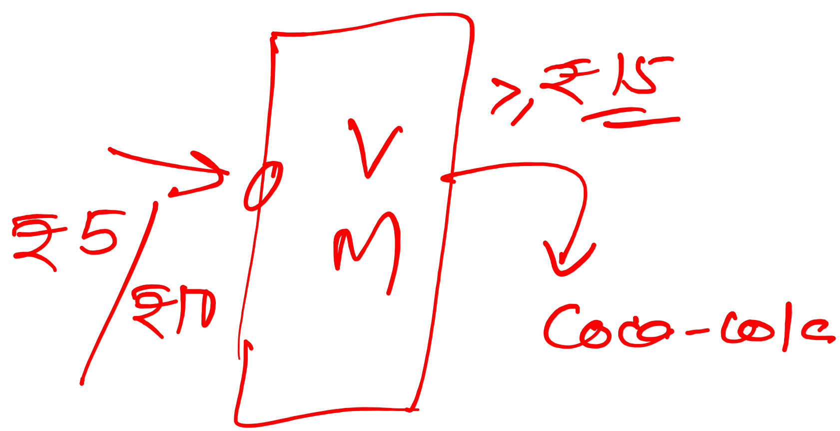
Minimize # states

δ & λ are function of current state & input

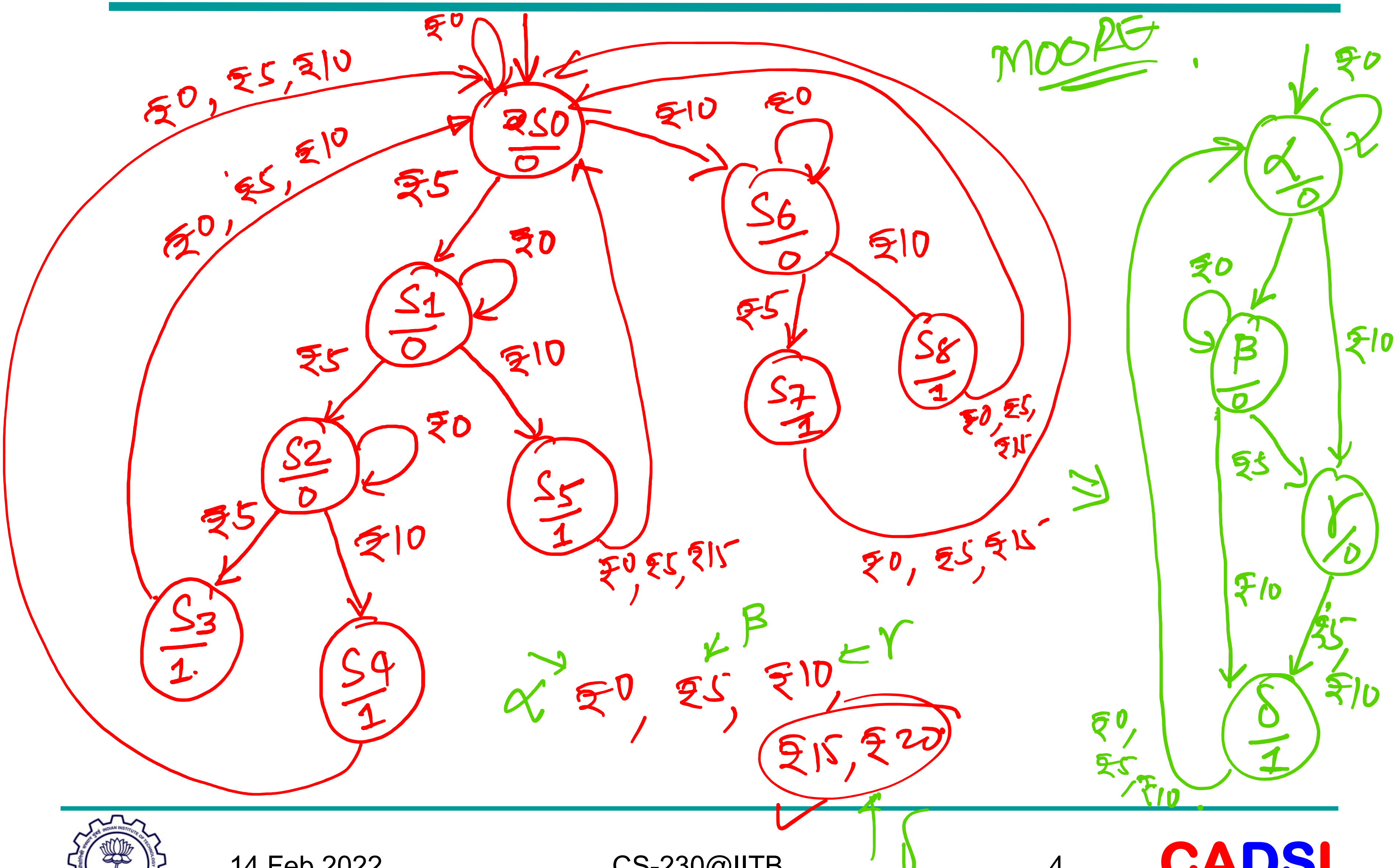
State encoding

State Machine

Minimize #States



Finite State Machine



State Minimization

X-Successor – If an input sequence X takes a machine from state S_i to state S_j, then S_j is said to be the X-successor of S_i

Strongly connected:- If for every pair of states (S_i, S_j) of a machine M there exists an input sequence which takes M from state S_i to S_j, then M is said to be strongly connected

$S_i \xrightarrow{0} S_k \xrightarrow{1} S_j$

S_j 01 S_1

$S_i \xrightarrow{X} S_j$

State Equivalence

- Two states S_i and S_j of machine M are distinguishable if and only if there exists at least one finite input sequence which, when applied to M, causes different output sequences, depending on whether S_i or S_j is the initial state

$\underbrace{0\ 1}$

$S_i \rightarrow 1\ 1$

$S_j \rightarrow 1\ 0$

$S_i \xrightarrow{0} S_p \xrightarrow{1} S_q$

$S_j \xrightarrow{0} S_k \xrightarrow{1} S_t$

$S_i \rightarrow 1\ 1$

$S_j \rightarrow 1\ 1$

S_i

S_j

State Equivalence

- Two states S_i and S_j of machine M are distinguishable if and only if there exists at least one finite input sequence which, when applied to M, causes different output sequences, depending on whether S_i or S_j is the initial state
- The sequence which distinguishes these states is called a distinguishing sequence of the pair (S_i, S_j)

distinguishing seq

01

S_i

S_j 01

output

1 1

1 0

State Equivalence

- Two states S_i and S_j of machine M are distinguishable if and only if there exists at least one finite input sequence which, when applied to M, causes different output sequences, depending on whether S_i or S_j is the initial state
- The sequence which distinguishes these states is called a distinguishing sequence of the pair $(S_{i,} S_j)$
- If there exists for pair $(S_{i,} S_j)$ a distinguishing sequence of length k, the states in $(S_{i,} S_j)$ are said to be k-distinguishable

State Equivalence

Machine M1

PS	NS, z	
	X = 0	X = 1
A	E, 0	D, 1
B	F, 0	D, 0
C	E, 0	B, 1
D	F, 0	B, 0
E	C, 0	F, 1
F	B, 0	C, 0

(A, B) – 1 Distinguishable

$A \xrightarrow{0} E, 0$

$A \xrightarrow{1} D, 1$

$B \xrightarrow{0} F, 0$

$B \xrightarrow{1} D, 0$

State Equivalence

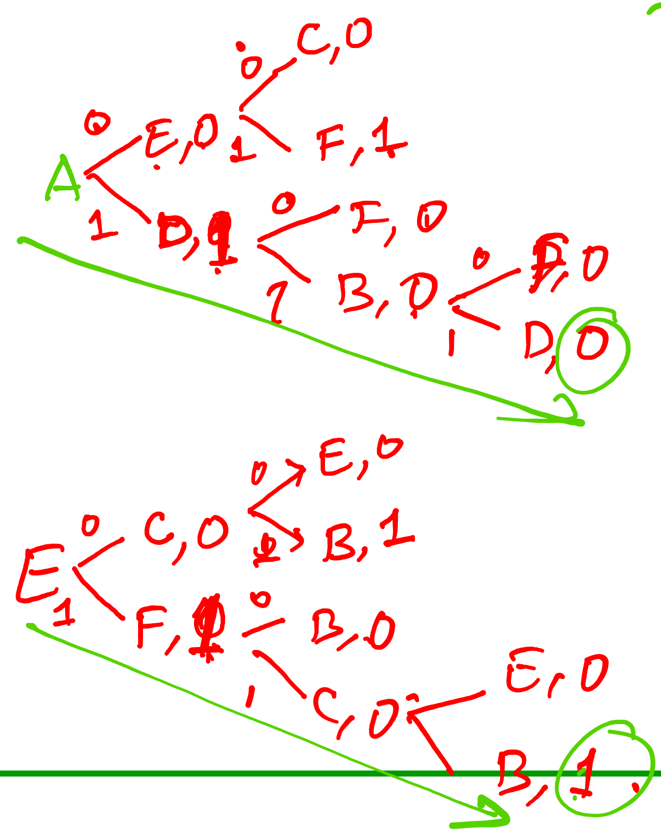
2- indistinguishable

Machine M1

PS	NS, z	
	X = 0	X = 1
A	E, 0	D, 1
B	F, 0	D, 0
C	E, 0	B, 1
D	F, 0	B, 0
E	C, 0	F, 1
F	B, 0	C, 0

(A, E) – 3 Distinguishable

Seq - 111



1, 0, 0

1, 0, 1

State Equivalence

Machine M1

PS	NS, z	
	X = 0	X = 1
A	E, 0	D, 1
B	F, 0	D, 0
C	E, 0	B, 1
D	F, 0	B, 0
E	C, 0	F, 1
F	B, 0	C, 0

(A, B) – 1 Distinguishable

(A, E) – 3 Distinguishable

Seq - 111

k-equivalent – The states that are not *k*-distinguishable are said to be k-equivalent

Also *r-equivalent* r<k

Distiguishable States

K- equivalent

K- indistinguishable.

State Equivalence

- States S_i and S_j of machine **M** are said to be equivalent if and only if, for every possible input sequence, the same output sequence will be produced regardless of whether S_i or S_j is the initial state

n = no. of States in state m/c

- States that are k-equivalent for all k < n-1, are equivalent

- $S_i \equiv S_j$, and $S_j \equiv S_k$, then $S_i \equiv S_k$

State Equivalence

- The set of states of a machine M can be partitioned into disjoint subsets, known as equivalence classes
- Two states are in the same equivalence class if and only if they are equivalent, and are in different classes if and only if they are distinguishable

Property: If S_i and S_j are equivalent states, their corresponding X-successors, for all X, are also equivalent

State Minimization Procedure

1. Partition the states of M into subsets s.t. all states in same subset are *1-equivalent*
2. Two states are 2-equivalent iff they are 1-equivalent and their I_i successors, for all possible I_i, are also 1-equivalent

MOORE MINIMIZATION

PS	NS, z	
	X = 0	X = 1
A	E, 0	D, 1
B	F, 0	D, 0
C	E, 0	B, 1
D	F, 0	B, 0
E	C, 0	F, 1
F	B, 0	C, 0

P0 = (ABCDEF)

State Minimization Procedure

1. Partition the states of M into subsets s.t. all states in same subset are *1-equivalent*
2. Two states are 2-equivalent iff they are 1-equivalent and their I_i successors, for all possible I_i, are also 1-equivalent

<table>
<tr><th rowspan="2">PS</th><th colspan="2">NS, z</th></tr>
<tr><th>X = 0</th><th>X = 1</th></tr>
<tr><td>A</td><td>E, 0</td><td>D, 1</td></tr>
<tr><td>B</td><td>F, 0</td><td>D, 0</td></tr>
<tr><td>C</td><td>E, 0</td><td>B, 1</td></tr>
<tr><td>D</td><td>F, 0</td><td>B, 0</td></tr>
<tr><td>E</td><td>C, 0</td><td>F, 1</td></tr>
<tr><td>F</td><td>B, 0</td><td>C, 0</td></tr>
</table>

P0 = (ABCDEF)

P1 = (ACE), (BDF)

State Minimization Procedure

1. Partition the states of M into subsets s.t. all states in same subset are *1-equivalent*
2. Two states are 2-equivalent iff they are 1-equivalent and their I_i successors, for all possible I_i, are also 1-equivalent

PS	NS, z	
	X = 0	X = 1
A	E, 0	D, 1
B	F, 0	D, 0
C	E, 0	B, 1
D	F, 0	B, 0
E	C, 0	F, 1
F	B, 0	C, 0

P0 = (ABCDEF)

P1 = (ACE), (BDF)

P2 = (ACE), (BD), (F)

State Minimization Procedure

1. Partition the states of M into subsets s.t. all states in same subset are *1-equivalent*
2. Two states are 2-equivalent iff they are 1-equivalent and their I_i successors, for all possible I_i, are also 1-equivalent

PS	NS, z	
	X = 0	X = 1
A	E, 0	D, 1
B	F, 0	D, 0
C	E, 0	B, 1
D	F, 0	B, 0
E	C, 0	F, 1
F	B, 0	C, 0

P0 = (ABCDEF)

P1 = (ACE), (BDF)

P2 = (ACE), (BD), (F)

P3 = (AC), (E), (BD), (F)

State Minimization Procedure

1. Partition the states of M into subsets s.t. all states in same subset are *1-equivalent*
2. Two states are 2-equivalent iff they are 1-equivalent and their I_i successors, for all possible I_i, are also 1-equivalent

PS	NS, z	
	X = 0	X = 1
A	E, 0	D, 1
B	F, 0	D, 0
C	E, 0	B, 1
D	F, 0	B, 0
E	C, 0	F, 1
F	B, 0	C, 0

P0 = (ABCDEF)

P1 = (ACE), (BDF)

P2 = (ACE), (BD), (F)

P3 = (AC), (E), (BD), (F)

P4 = (AC), (E), (BD), (F)

State Minimization Procedure

1. Partition the states of M into subsets s.t. all states in same subset are *1-equivalent*
2. Two states are 2-equivalent iff they are 1-equivalent and their I_i successors, for all possible I_i, are also 1-equivalent

PS	NS, z	
	X = 0	X = 1
A	E, 0	D, 1
B	F, 0	D, 0
C	E, 0	B, 1
D	F, 0	B, 0
E	C, 0	F, 1
F	B, 0	C, 0

P0 = (ABCDEF)

P1 = (ACE), (BDF)

P2 = (ACE), (BD), (F)

P3 = (AC), (E), (BD), (F)

P4 = (AC), (E), (BD), (F)

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

PS	NS, z	
	X = 0	X = 1
AC - α	β, 0	γ, 1
E - β	α, 0	δ, 1
BD - γ	δ, 0	γ, 0
F - δ	γ, 0	α, 0

φ

State Equivalence - Example

Machine M2

PS	NS, z	
	X = 0	X = 1
A	E, 0	C, 0
B	C, 0	A, 0
C	B, 0	G, 0
D	G, 0	A, 0
E	F, 1	B, 0
F	E, 0	D, 0
G	D, 0	G, 0

P0 = (ABCDEFG)

P1 = (ABCDFG) (E)

P2 = (AF) (BCDG) (E)

P3 = (AF) (BD) (CG) (E)

P4 = (A) (F) (BD) (CG) (E)

P5 = (A) (F) (BD) (CG) (E)

5

Thank You