Digital Logic Design
ECEN 3233
Module 9a – FSM Optimization: State Minimization

Weihua Sheng
School of Electrical and Computer Engineering
Oklahoma State University

Spring 2007

Motivation

Basic FSM Design Procedure:

(1) Understand the problem

(2) Obtain a formal description

(3) Minimize number of states

(4) Encode the states

(5) Choose FFs to implement state registers (D FFs are usually more popular than JK, T)

(6) Implement the FSM

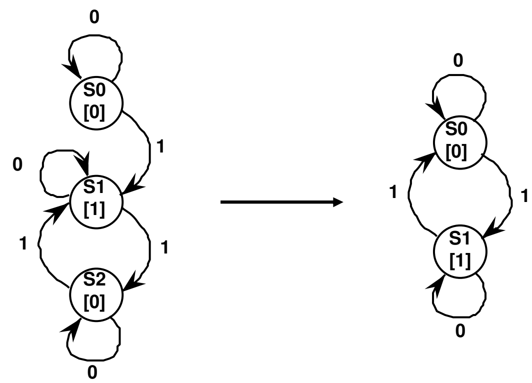
Finite state machine optimization

- State minimization
  - fewer states require fewer state bits
  - fewer bits require fewer logic equations
- Encodings: state, inputs, outputs
  - state encoding with fewer bits has fewer equations to implement
    - however, each may be more complex
  - state encoding with more bits (e.g., one-hot) has simpler equations
  - input/output encoding may or may not be under designer control

Algorithmic approach to state minimization

- Goal – identify and combine states that have equivalent behavior
- Equivalent states: two states are equivalent if they have the same output behavior and the same next state behavior
  - same output
  - for all input combinations, states transition to same or equivalent states

Example 1



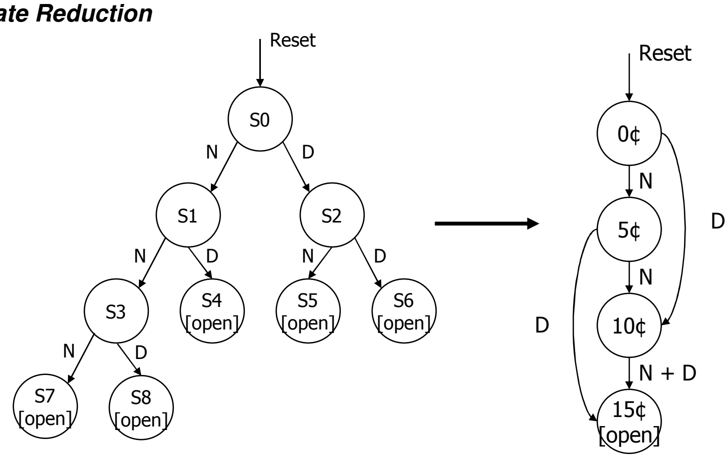
Odd Parity Checker: **S0, S2 are equivalent states**

Both output a 0

Both transition to S1 on a 1 and self-loop on a 0

Example 2

State Reduction



Vending Machine

State Reduction

Goal

Identify and combine states that have equivalent behavior

Algorithmic Approach

- **Start with state transition table**
- **Identify states with same output behavior**
- **If such states transition to the same next state, they are equivalent**
- **Combine into a single new renamed state**
- **Repeat until no new states are combined**

State reduction using row matching method: example

4-bit sequence detector:

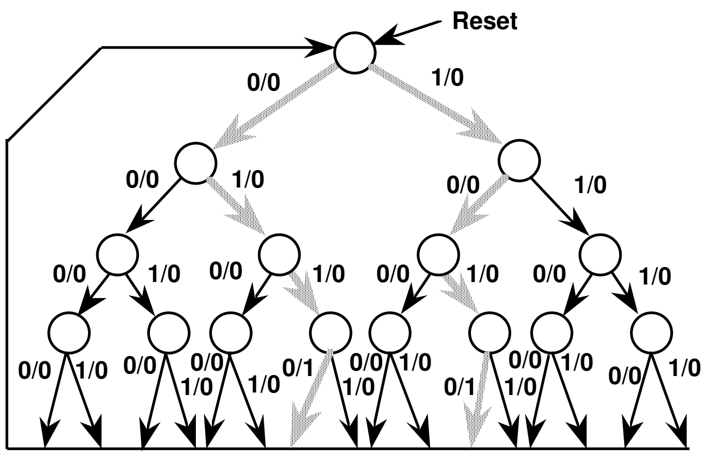
Single input X, output Z
Taking inputs grouped four at a time, output 1 if last four inputs were the string 1010 or 0110

Example I/O Behavior:

X = 0010 0110 1100 1010 0011 . . .
Z = 0000 0001 0000 0001 0000 . . .

State diagram for 4-bit sequence detector

Upper bound on FSM complexity:
- **-Fifteen states (1 + 2 + 4 + 8)**
- **-Thirty transitions (2 + 4 + 8 + 16)**
- **-Sufficient to recognize any binary string of length four!**



State reduction using row-matching method

Initial State Transition Table:

Input Sequence	Present State	Next State X=0	Next State X=1	Output X=0	Output X=1
Reset	S_0	S_1	S_2	0	0
0	S_1	S_3	S_4	0	0
1	S_2	S_5	S_6	0	0
00	S_3	S_7	S_8	0	0
01	S_4	S_9	S_{10}	0	0
10	S_5	S_{11}	S_{12}	0	0
11	S_6	S_{13}	S_{14}	0	0
000	S_7	S_0	S_0	0	0
001	S_8	S_0	S_0	0	0
010	S_9	S_0	S_0	0	0
011	S_{10}	S_0	S_0	1	0
100	S_{11}	S_0	S_0	0	0
101	S_{12}	S_0	S_0	1	0
110	S_{13}	S_0	S_0	0	0
111	S_{14}	S_0	S_0	0	0

State reduction using row-matching method

Initial State Transition Table:

Input Sequence	Present State	Next State X=0	Next State X=1	Output X=0	Output X=1
Reset	S_0	S_1	S_2	0	0
0	S_1	S_3	S_4	0	0
1	S_2	S_5	S_6	0	0
00	S_3	S_7	S_8	0	0
01	S_4	S_9	S_{10}	0	0
10	S_5	S_{11}	S_{12}	0	0
11	S_6	S_{13}	S_{14}	0	0
000	S_7	S_0	S_0	0	0
001	S_8	S_0	S_0	0	0
010	S_9	S_0	S_0	0	0
011	S_{10}	S_0	S_0	1	0
100	S_{11}	S_0	S_0	0	0
101	S_{12}	S_0	S_0	1	0
110	S_{13}	S_0	S_0	0	0
111	S_{14}	S_0	S_0	0	0

State reduction using row-matching method

Input Sequence	Present State	Next State X=0	Next State X=1	Output X=0	Output X=1
Reset	S_0	S_1	S_2	0	0
0	S_1	S_3	S_4	0	0
1	S_2	S_5	S_6	0	0
00	S_3	S_7	S_8	0	0
01	S_4	S_9	S'_{10}	0	0
10	S_5	S_{11}	S'_{10}	0	0
11	S_6	S_{13}	S_{14}	0	0
000	S_7	S_0	S_0	0	0
001	S_8	S_0	S_0	0	0
010	S_9	S_0	S_0	0	0
011 or 101	S'_{10}	S_0	S_0	1	0
100	S_{11}	S_0	S_0	0	0
110	S_{13}	S_0	S_0	0	0
111	S_{14}	S_0	S_0	0	0

State reduction using row-matching method

Input Sequence	Present State	Next State X=0	Next State X=1	Output X=0	Output X=1
Reset	S_0	S_1	S_2	0	0
0	S_1	S_3	S_4	0	0
1	S_2	S_5	S_6	0	0
00	S_3	S_7	S_8	0	0
01	S_4	S_9	S'_{10}	0	0
10	S_5	S_{11}	S'_{10}	0	0
11	S_6	S_{13}	S_{14}	0	0
000	S_7	S_0	S_0	0	0
001	S_8	S_0	S_0	0	0
010	S_9	S_0	S_0	0	0
011 or 101	S'_{10}	S_0	S_0	1	0
100	S_{11}	S_0	S_0	0	0
110	S_{13}	S_0	S_0	0	0
111	S_{14}	S_0	S_0	0	0

State reduction using row-matching method

Input Sequence	Present State	Next State X=0	Next State X=1	Output X=0	Output X=1
Reset	S_0	S_1	S_2	0	0
0	S_1	S_3	S_4	0	0
1	S_2	S_5	S_6	0	0
00	S_3	S'_7	S'_7	0	0
01	S_4	S'_7	S'_{10}	0	0
10	S_5	S'_7	S'_{10}	0	0
11	S_6	S'_7	S'_7	0	0
not (011 or 101)	S'_7	S_0	S_0	0	0
011 or 101	S'_{10}	S_0	S_0	1	0

State reduction using row-matching method

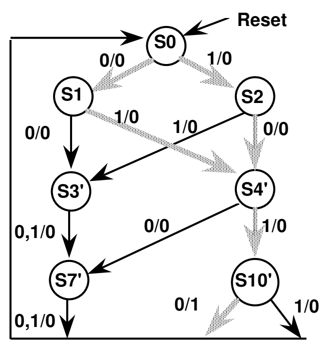
Input Sequence	Present State	Next State X=0	Next State X=1	Output X=0	Output X=1
Reset	S_0	S_1	S_2	0	0
0	S_1	S_3	S_4	0	0
1	S_2	S_5	S_6	0	0
00	S_3	S_7'	S_7'	0	0
01	S_4	S_7'	S_{10}'	0	0
10	S_5	S_7'	S_{10}'	0	0
11	S_6	S_7'	S_7'	0	0
not (011 or 101)	S_7'	S_0	S_0	0	0
011 or 101	S_{10}'	S_0	S_0	1	0

State reduction using row-matching method

Final Reduced State Transition Table

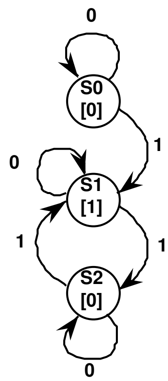
Input Sequence	Present State	Next State X=0	Next State X=1	Output X=0	Output X=1
Reset	S0	S1	S2	0	0
0	S1	S3'	S4'	0	0
1	S2	S4'	S3'	0	0
00 or 11	S3'	S7'	S7'	0	0
01 or 10	S4'	S7'	S10'	0	0
not (011 or 101)	S7'	S0	S0	0	0
011 or 101	S10'	S0	S0	1	0

Corresponding State Diagram



Discussion on row matching method

- Row matching method does not always yield the most-reduced state table.



Present State	Next State X=0	Next State X=1	Output
S0	S0	S1	0
S1	S1	S2	1
S2	S2	S1	0

State reduction using implication chart method

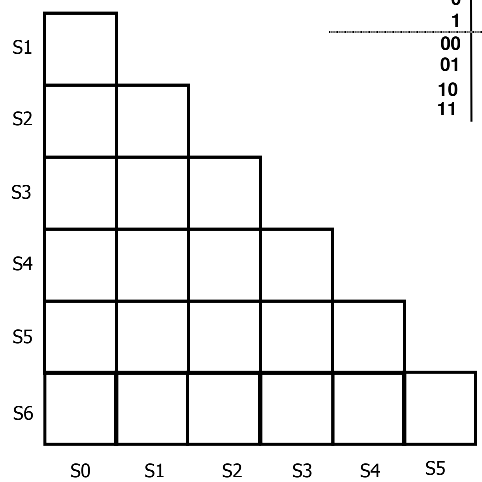
3-bit sequence detector: detect 010 or 110

Input Sequence	Present State	Next State X=0	Next State X=1	Output X=0	Output X=1
Reset	S0	S1	S2	0	0
0	S1	S3	S4	0	0
1	S2	S5	S6	0	0
00	S3	S0	S0	0	0
01	S4	S0	S0	1	0
10	S5	S0	S0	0	0
11	S6	S0	S0	1	0

Initial state transition table for the 3-bit sequence detector

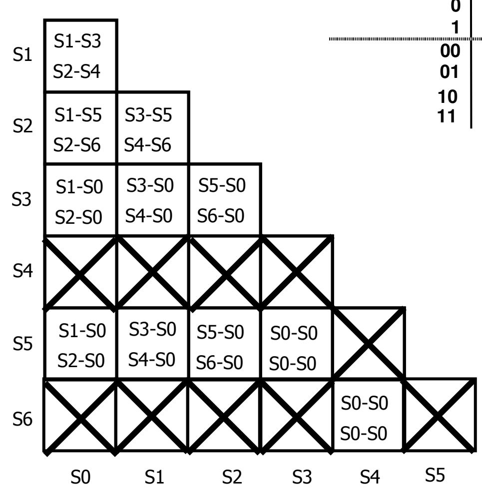
State reduction using implication chart method

Input Sequence	Present State	Next State X=0	Next State X=1	Output X=0	Output X=1
Reset	S0	S1	S2	0	0
0	S1	S3	S4	0	0
1	S2	S5	S6	0	0
00	S3	S0	S0	0	0
01	S4	S0	S0	1	0
10	S5	S0	S0	0	0
11	S6	S0	S0	1	0



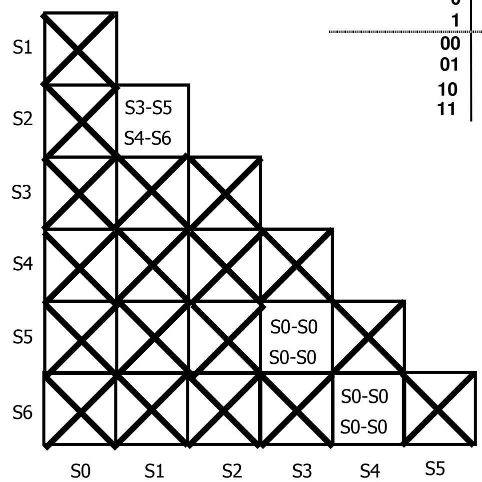
State reduction using implication chart method

Input Sequence	Present State	Next State X=0	Next State X=1	Output X=0	Output X=1
Reset	S0	S1	S2	0	0
0	S1	S3	S4	0	0
1	S2	S5	S6	0	0
00	S3	S0	S0	0	0
01	S4	S0	S0	1	0
10	S5	S0	S0	0	0
11	S6	S0	S0	1	0



State reduction using implication chart method

Input Sequence	Present State	Next State X=0	Next State X=1	Output X=0	Output X=1
Reset	S0	S1	S2	0	0
0	S1	S3	S4	0	0
1	S2	S5	S6	0	0
00	S3	S0	S0	0	0
01	S4	S0	S0	1	0
10	S5	S0	S0	0	0
11	S6	S0	S0	1	0



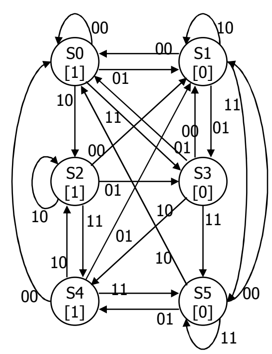
State reduction using implication chart method

Final reduced state transition table for 3-bit sequence detector

Input Sequence	Present State	Next State X=0	Next State X=1	Output X=0	Output X=1
Reset	S0	S1'	S1'	0	0
0 or 1	S1'	S3'	S4'	0	0
00 or 10	S3'	S0	S0	0	0
01 or 11	S4'	S0	S0	1	0

Multiple input example

- Multiple input example

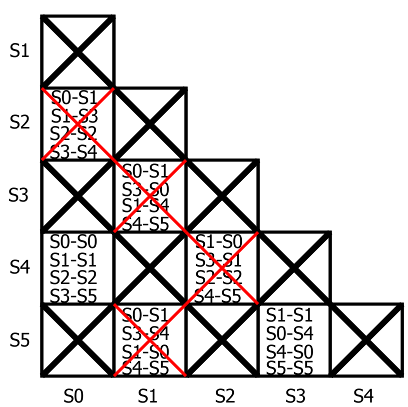


inputs here

present state	next state 00	01	10	11	output
S0	S0	S1	S2	S3	1
S1	S0	S3	S1	S4	0
S2	S1	S3	S2	S4	1
S3	S1	S0	S4	S5	0
S4	S0	S1	S2	S5	1
S5	S1	S4	S0	S5	0

symbolic state transition table

Multiple input example



present state	next state 00	01	10	11	output
S0'	S0'	S1	S2	S3'	1
S1	S0'	S3'	S1	S3'	0
S2	S1	S3'	S2	S0'	1
S3'	S1	S0'	S0'	S3'	0

minimized state table
(S0==S4) (S3==S5)

FSM optimization summary

- State minimization
  - Row matching method
  - Implication chart method
- Next: state assignment