Chapter #9: Finite State Machine Optimization

Contemporary Logic Design

Randy H. Katz
University of California, Berkeley

July 1993

Outline

- ***Procedures for optimizing implementation of an FSM***

  State Reduction

  State Assignment

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 register

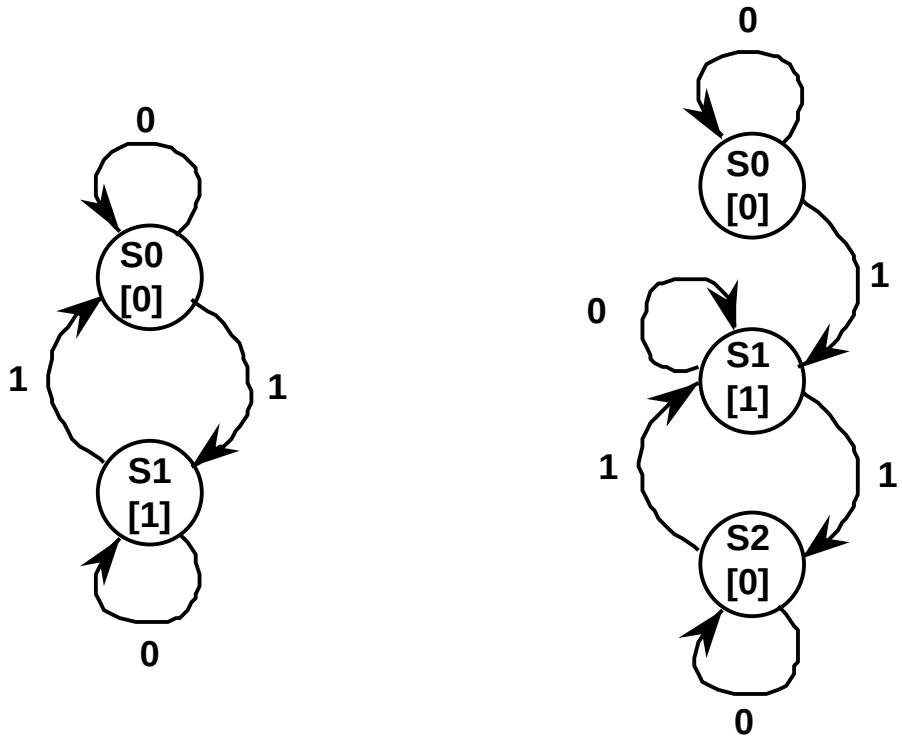
This Chapter!

(6) Implement the FSM

Next Chapter

Motivation

State Reduction



Odd Parity Checker: two alternative state diagrams

- **Identical output behavior on all input strings**
- **FSMs are *equivalent*, but require different implementations**
- **Design state diagram without concern for # of states, Reduce later**

Motivation

State Reduction (continued)

Implement FSM with fewest possible states

- **Least number of flipflops**
- **Boundaries are power of two number of states**
- **Fewest states usually leads to more opportunities for don't cares**
- **Reduce the number of gates needed for implementation**

State Reduction

Goal

Identify and combine states that have equivalent behavior

Equivalent States: **for all input combinations, states transition to the same or equivalent states**

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

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

Row Matching Method

Example FSM Specification:

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 . . .

Upper bound on FSM complexity:

Fifteen states (1 + 2 + 4 + 8)

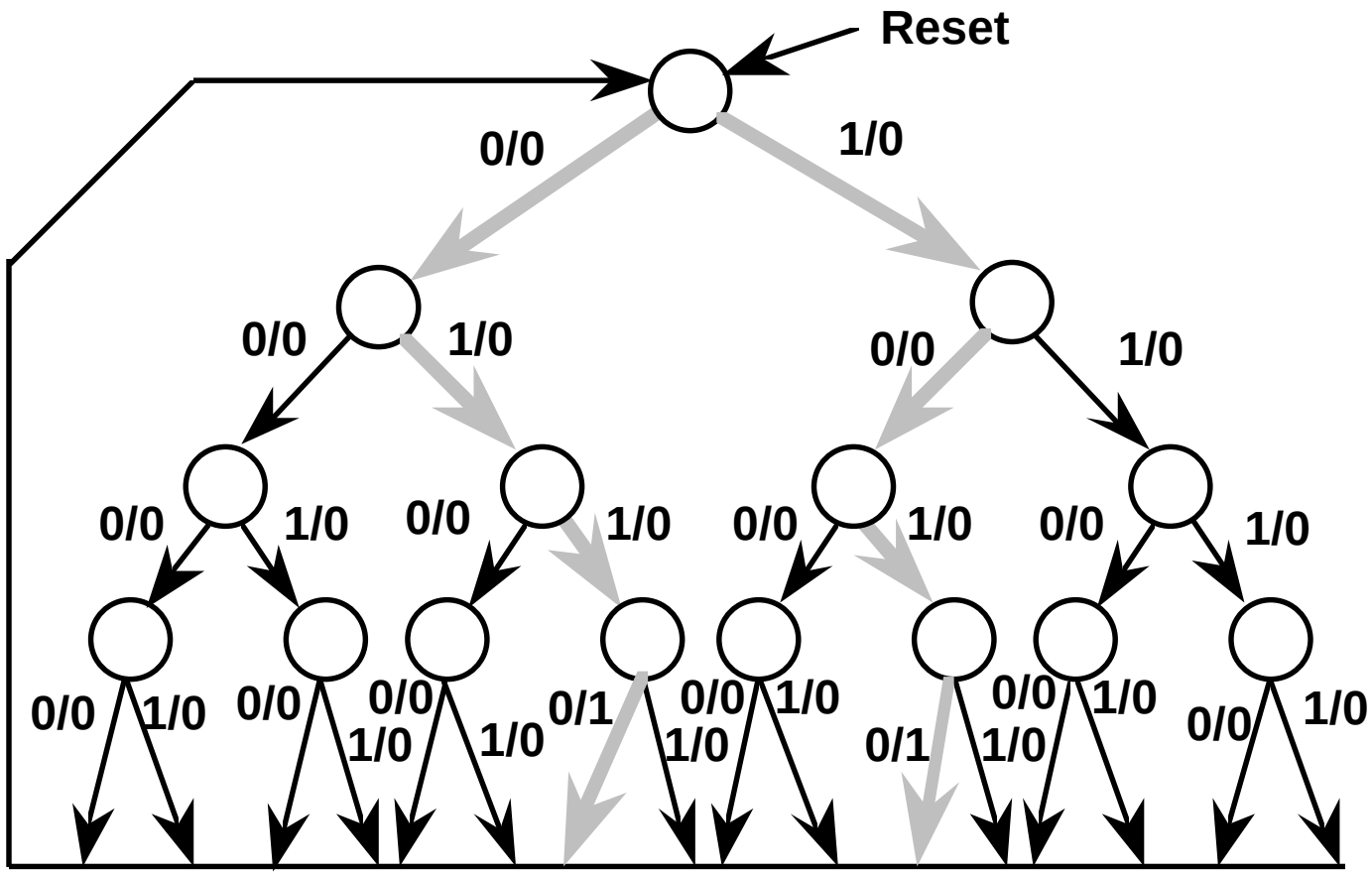
Thirty transitions (2 + 4 + 8 + 16)

sufficient to recognize any binary string of length four!

State Reduction

Row Matching Method

State Diagram for Example FSM:



State Reduction

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

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

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

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

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

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

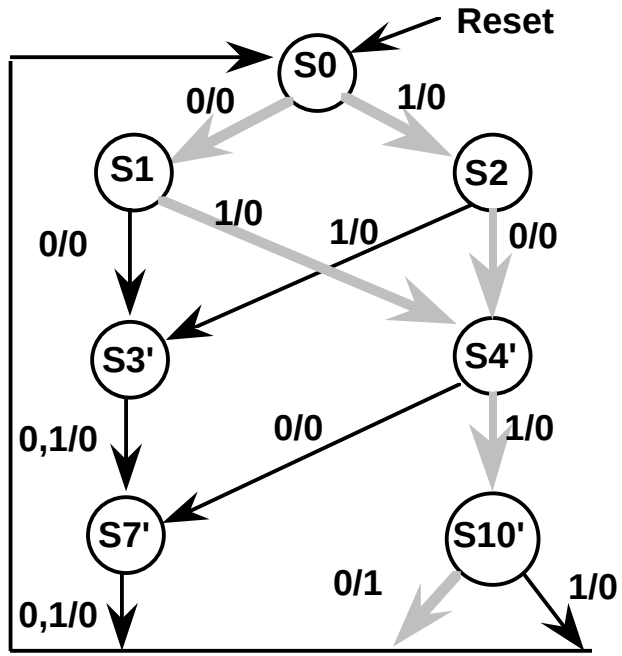
State Reduction

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



State Reduction

Row Matching Method

- **Straightforward to understand and easy to implement**
- **Problem: does not allows yield the most reduced state table!**

Example: 3 State Odd Parity Checker

Present State	Next State X=0	Next State X=1	Output
S_0	S_0	S_1	0
S_1	S_1	S_2	1
S_2	S_2	S_1	0

No way to combine states S0 and S2 based on Next State Criterion!

State Reduction

Implication Chart Method

New example FSM:

Single input X, Single output Z

Output a 1 whenever the serial sequence 010 or 110 has been observed at the inputs

State transition table:

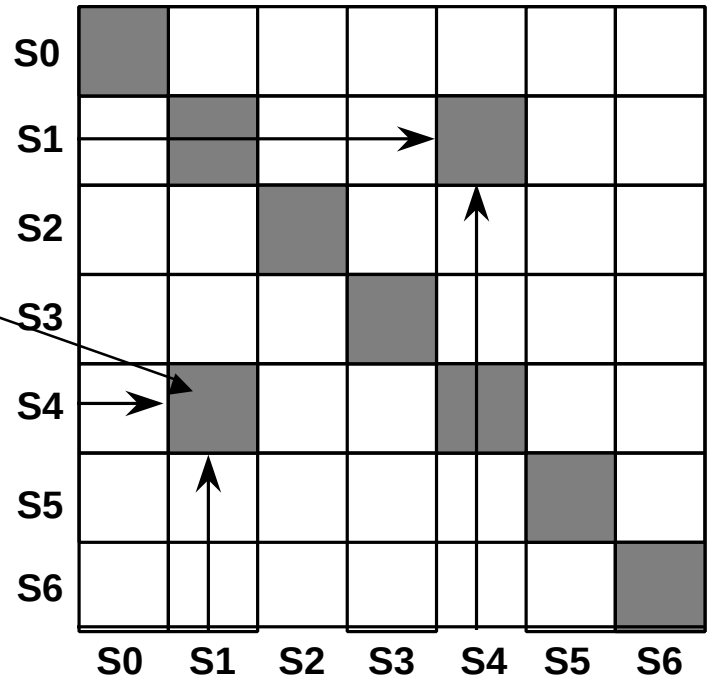
Input Sequence	Present State	Next State X=0	Next State X=1	Output X=0	Output X=1
Reset	S_0	S_1	S_2	0	0
0	S_1	S_3	S_4	0	0
1	S_2	S_5	S_6	0	0
00	S_3	S_0	S_0	0	0
01	S_4	S_0	S_0	1	0
10	S_5	S_0	S_0	0	0
11	S_6	S_0	S_0	1	0

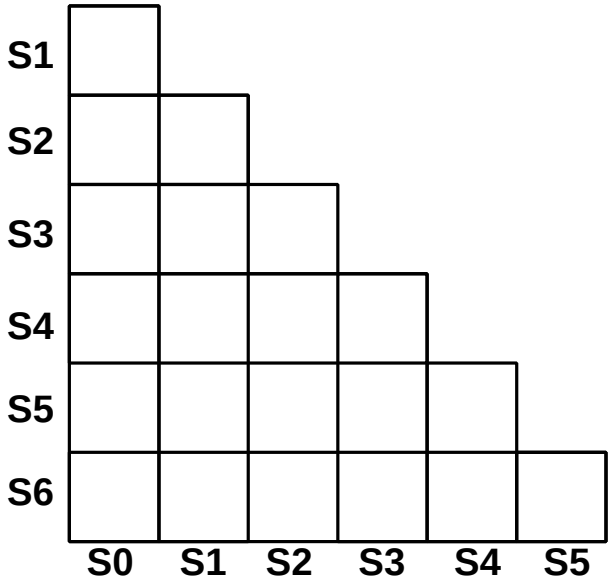
State Reduction

Implication Chart Method

Enumerate all possible combinations of states taken two at a time

Next States Under all Input Combinations





Implication Chart

Naive Data Structure:
Xij will be the same as Xji
Also, can eliminate the diagonal

State Reduction

Implication Chart

Filling in the Implication Chart

Entry Xij — Row is Si, Column is Sj

Si is equivalent to Sj if outputs are the same and
next states are equivalent

Xij contains the next states of Si, Sj which must be equivalent if
Si and Sj are equivalent

If Si, Sj have different output behavior, then Xij is crossed out

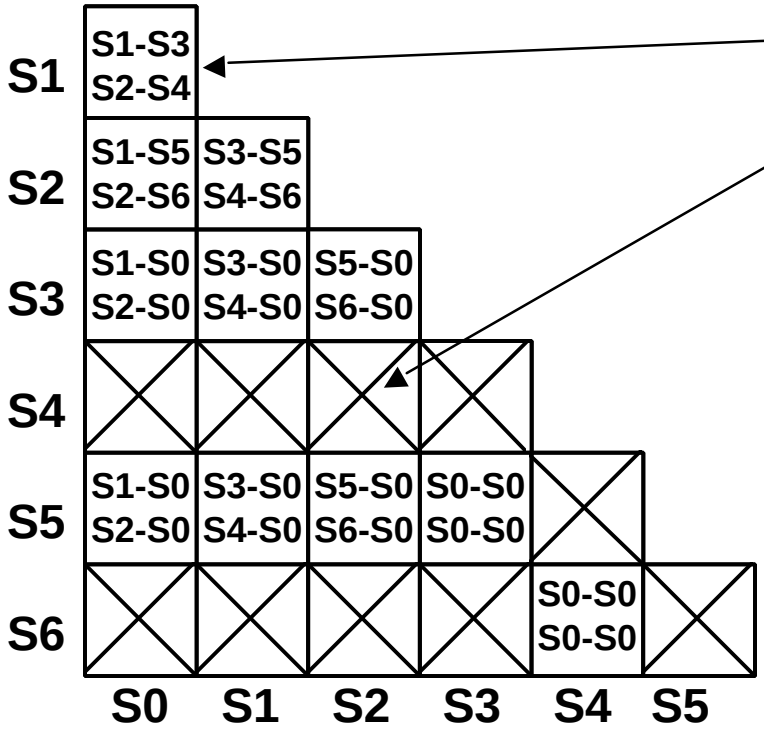
Example:
S0 transitions to S1 on 0, S2 on 1;
S1 transitions to S3 on 0, S4 on 1;

So square X<0,1> contains entries S1-S3 (transition on zero)
S2-S4 (transition on one)

S0	S1-S3 S2-S4
S1	

State Reduction

Implication Chart Method



Starting Implication Chart

S2 and S4 have different I/O behavior

This implies that S1 and S0 cannot be combined

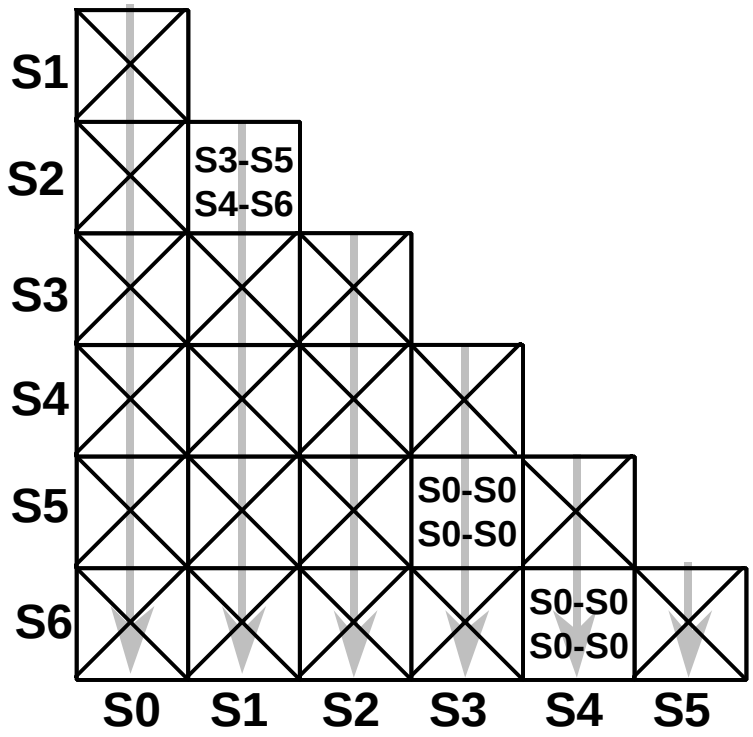
State Reduction

Implication Chart Method

Results of First Marking Pass

Second Pass Adds No New Information

- **S3 and S5 are equivalent**
- **S4 and S6 are equivalent**
- **This implies that S1 and S2 are too!**

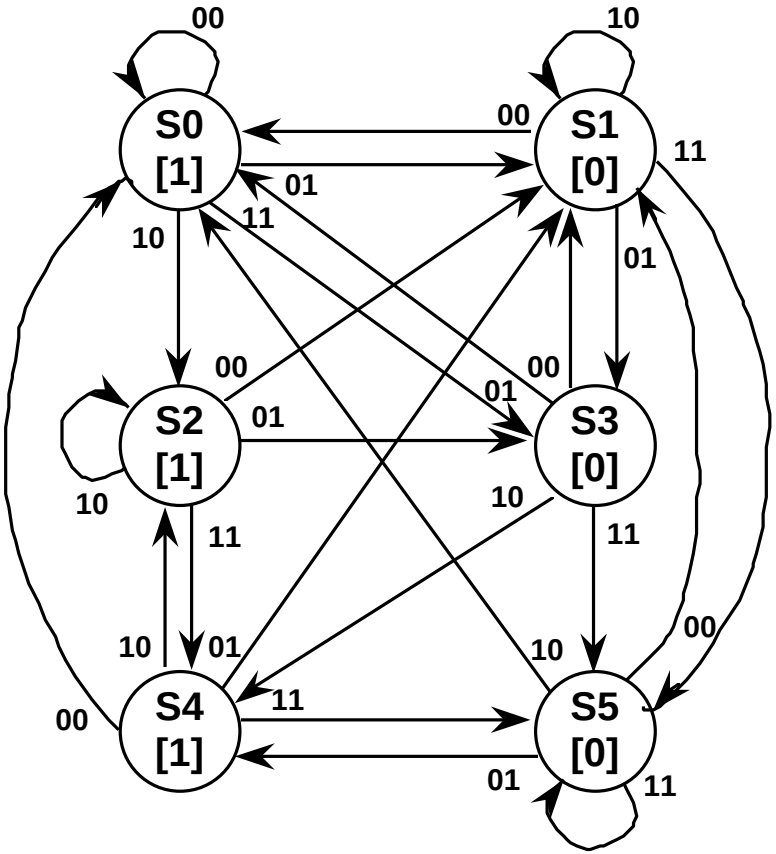


Reduced State Transition Table

Input Sequence	Present State	Next State X=0	Next State X=1	Output X=0	Output X=1
Reset	S_0	S'_1	S'_1	0	0
0 or 1	S'_1	S'_3	S'_4	0	0
00 or 10	S'_3	S_0	S_0	0	0
01 or 11	S'_4	S_0	S_0	1	0

State Reduction

Multiple Input State Diagram Example



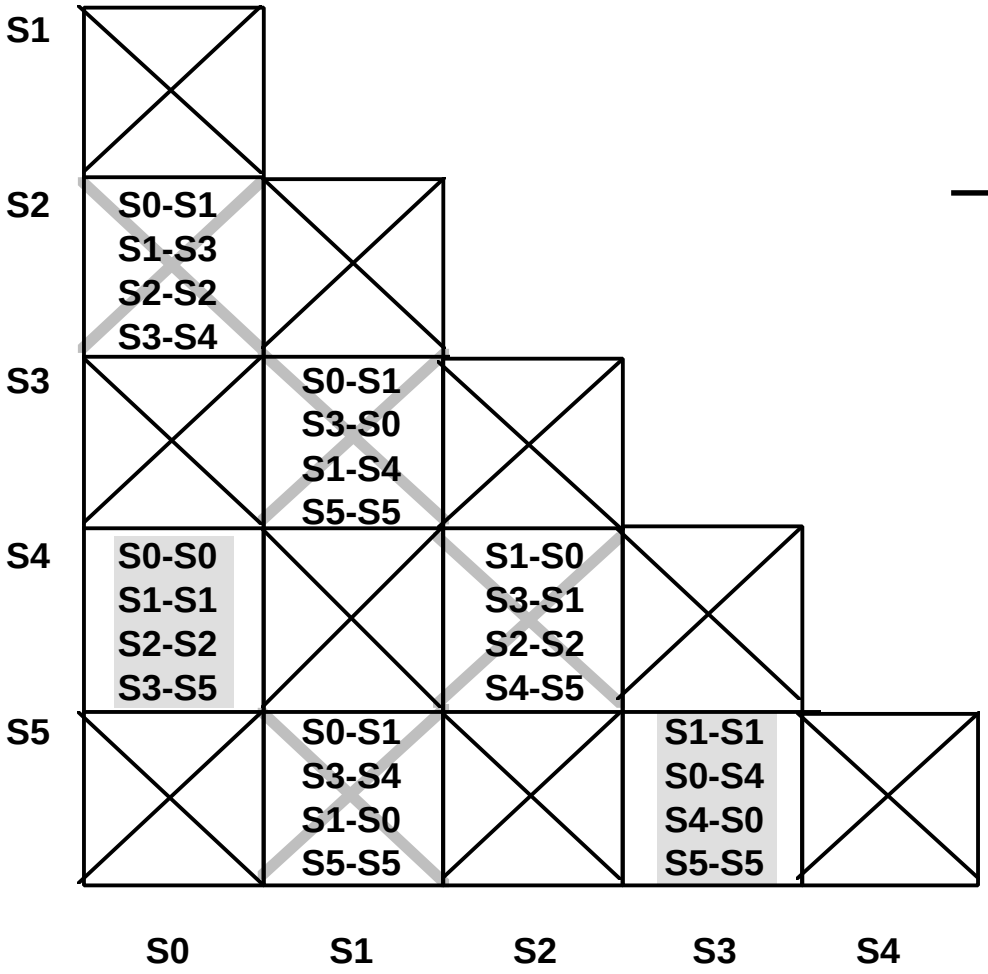
State Diagram

Present State	Next State 00	01	10	11	Output
S_0	S_0	S_1	S_2	S_3	1
S_1	S_0	S_3	S_1	S_5	0
S_2	S_1	S_3	S_2	S_4	1
S_3	S_1	S_0	S_4	S_5	0
S_4	S_0	S_1	S_2	S_5	1
S_5	S_1	S_4	S_0	S_5	0

Symbolic State Diagram

State Reduction

Multiple Input Example



Implication Chart

Present State	Next State 00	01	10	11	Output
S'_0	S_0'	S_1	S_2	S_3'	1
S_1	S_0'	S_3'	S_1	S_3'	0
S_2	S_1	S_3'	S_2	S_0'	1
S'_3	S_1	S'_0	S'_0	S'_3	0

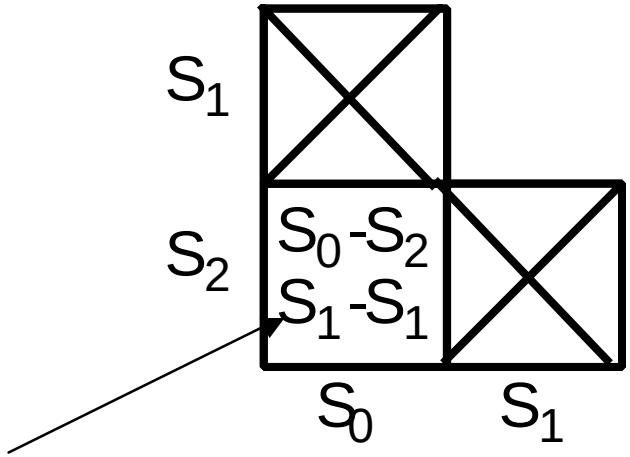
Minimized State Table

State Reduction

Implication Chart Method

Does the method solve the problem with the odd parity checker?

Implication Chart



S0 is equivalent to S2
since nothing contradicts this assertion!

State Reduction

Implication Chart Method

The detailed algorithm:

1. **Construct implication chart, one square for each combination of states taken two at a time**

2. **Square labeled Si, Sj, if outputs differ than square gets "X". Otherwise write down implied state pairs for all input combinations**

3. **Advance through chart top-to-bottom and left-to-right. If square Si, Sj contains next state pair Sm, Sn and that pair labels a square already labeled "X", then Si, Sj is labeled "X".**

4. **Continue executing Step 3 until no new squares are marked with "X".**

5. **For each remaining unmarked square Si, Sj, then Si and Sj are equivalent.**

State Assignment

When FSM implemented with gate logic, number of gates will depend on mapping between symbolic state names and binary encodings

4 states = 4 choices for first state, 3 for second, 2 for third, 1 for last
= 24 different encodings (4!)

Example for State Assignment: Traffic Light Controller

HG	HY	FG	FY
00	01	10	11
00	01	11	10
00	10	01	11
00	10	11	01
00	11	01	10
00	11	10	01
01	00	10	11
01	00	11	10
01	10	00	11
01	10	11	00
01	11	00	10
01	11	10	00

HG	HY	FG	FY
10	00	01	11
10	00	11	01
10	01	00	11
10	01	11	00
10	11	00	01
10	11	01	00
11	00	01	10
11	00	10	01
11	01	00	10
11	01	10	00
11	10	00	01
11	10	01	00

Inputs			Present State	Next State	Outputs		
C	TL	TS	Q_1 Q_0	P_1 P_0	ST	H_1 H_0	F_1 F_0
0	X	X	HG	HG	0	00	10
X	0	X	HG	HG	0	00	10
1	1	X	HG	HY	1	00	10
X	X	0	HY	HY	0	01	10
X	X	1	HY	FG	1	01	10
1	0	X	FG	FG	0	10	00
0	X	X	FG	FY	1	10	00
X	1	X	FG	FY	1	10	00
X	X	0	FY	FY	0	10	01
X	X	1	FY	HG	1	10	01

24 state assignments for the traffic light controller

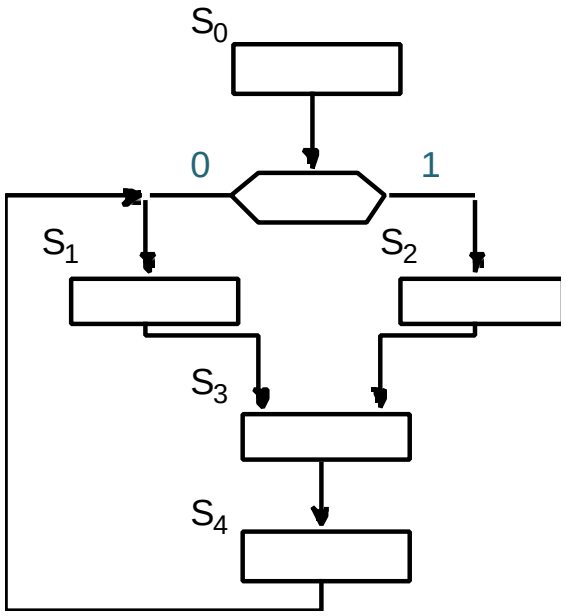
Symbolic State Names: HG, HY, FG, FY

State Assignment

Pencil & Paper Heuristic Methods

State Maps: similar in concept to K-maps

If state X transitions to state Y, then assign "close" assignments to X and Y

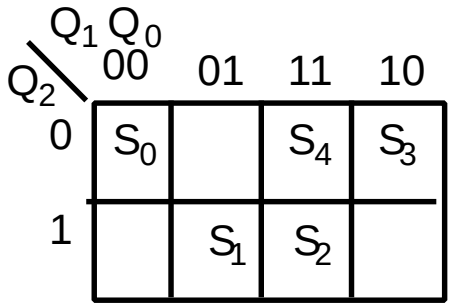


State Name	Assignment Q_2	Q_1	Q_0
S_0	0	0	0
S_1	1	0	1
S_2	1	1	1
S_3	0	1	0
S_4	0	1	1

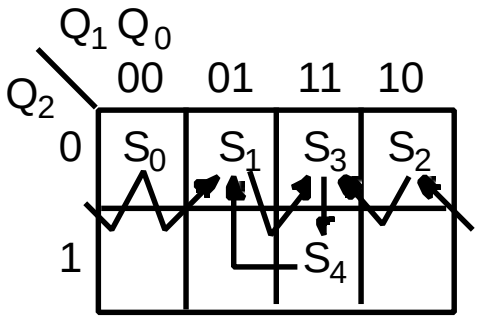
Assignment

State Name	Assignment Q_2	Q_1	Q_0
S_0	0	0	0
S_1	0	0	1
S_2	0	1	0
S_3	0	1	1
S_4	1	1	1

Assignment



State Map



State Map

State Assignment

Paper and Pencil Methods

Minimum Bit Distance Criterion

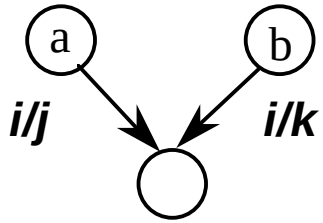
Transition	*First Assignment Bit Changes*	*Second Assignment Bit Changes*
S0 to S1:	2	1
S0 to S2:	3	1
S1 to S3:	3	1
S2 to S3:	2	1
S3 to S4:	1	1
S4 to S1:	2	2
	13	7

Traffic light controller: HG = 00, HY = 01, FG = 11, FY = 10
yields minimum distance encoding but not best assignment!

State Assignment

Paper & Pencil Methods

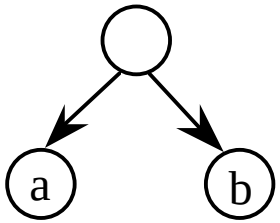
Alternative heuristics based on input and output behavior as well as transitions:



Highest Priority

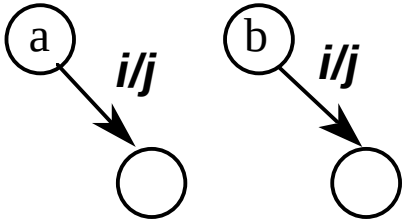
Adjacent assignments to:

states that share a common next state (group 1's in next state map)



Medium Priority

states that share a common ancestor state (group 1's in next state map)



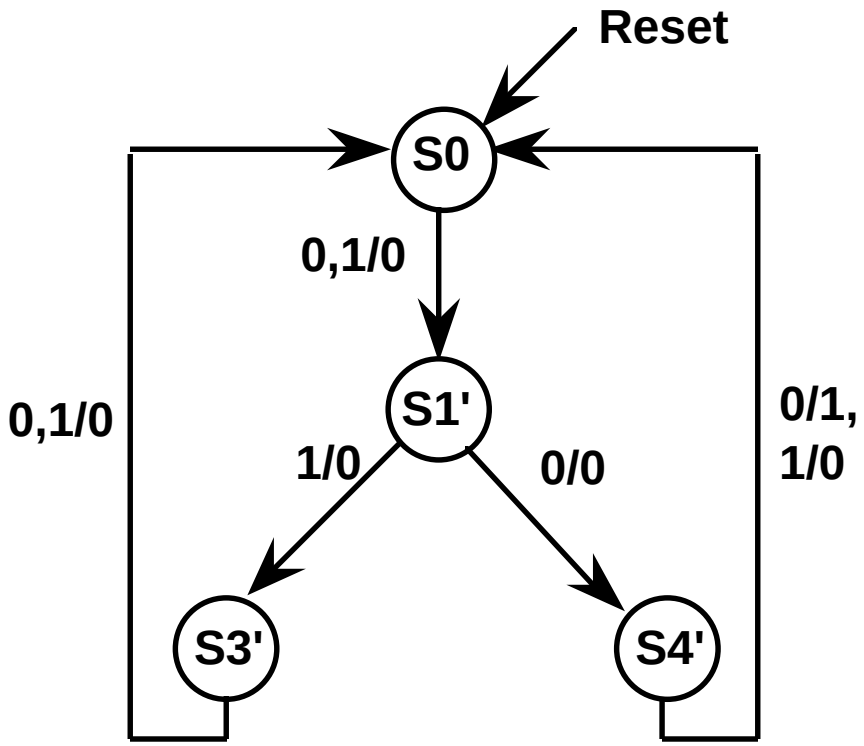
Lowest Priority

states that have common output behavior (group 1's in output map)

State Assignment

Pencil and Paper Methods

Example: 3-bit Sequence Detector



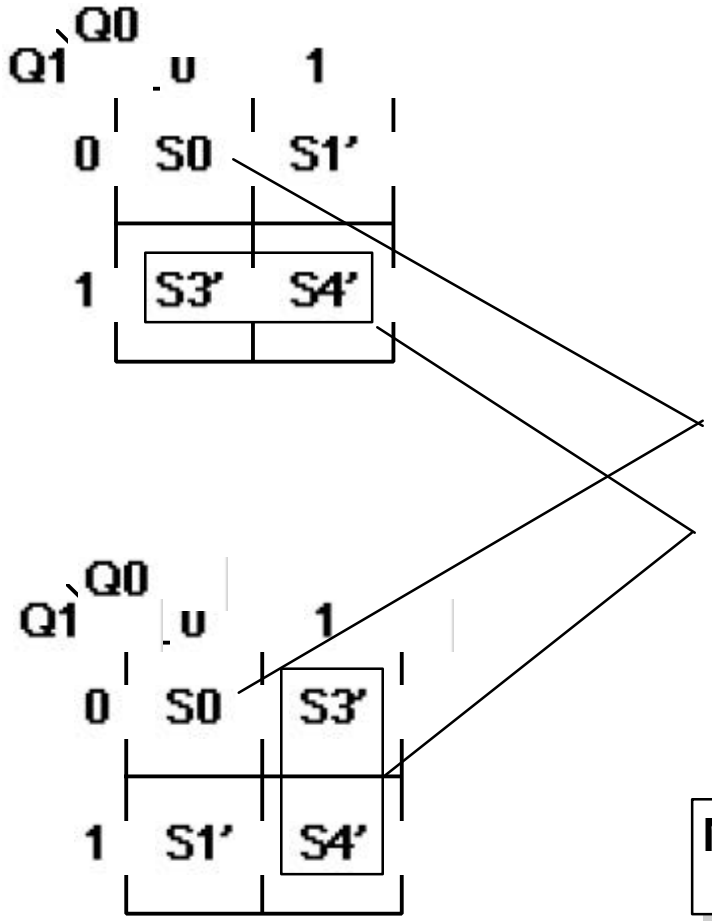
Highest Priority: (S3', S4')

Medium Priority: (S3', S4')

Lowest Priority:
0/0: (S0, S1', S3')
1/0: (S0, S1', S3', S4')

State Assignment

Paper and Pencil Methods



Reset State = 00

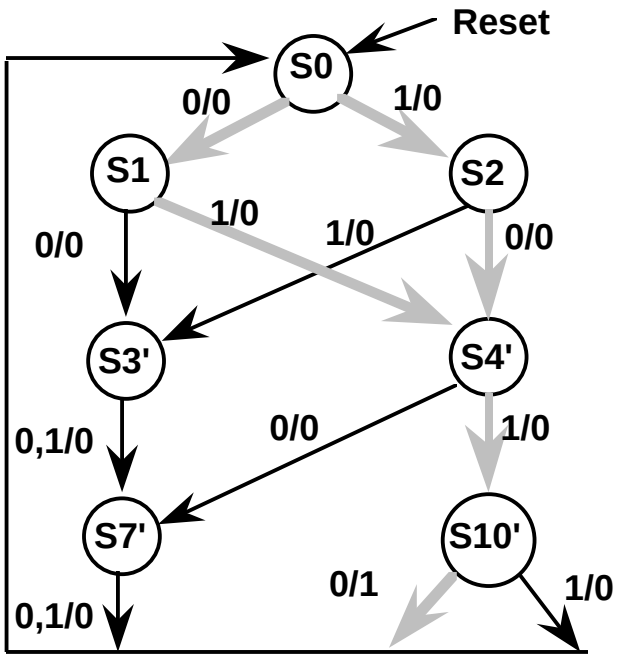
Highest Priority Adjacency

Not much difference in these two assignments

State Assignment

Paper & Pencil Methods

Another Example: 4 bit String Recognizer



Highest Priority: (S3', S4'), (S7', S10')

Medium Priority:
(S1, S2), 2x(S3', S4'), (S7', S10')

Lowest Priority:
0/0: (S0, S1, S2, S3', S4', S7')
1/0: (S0, S1, S2, S3', S4', S7')

State Assignment

Paper & Pencil Methods

State Map

Q2 \ Q1 Q0	00	01	11	10
0	S0			
1				

Q2 \ Q1 Q0	00	01	11	10
0	S0		S3'	
1			S4'	

Q2 \ Q1 Q0	00	01	11	10
0	S0		S3'	S7'
1			S4'	S10'

Q2 \ Q1 Q0	00	01	11	10
0	S0	S1	S3'	S7'
1		S2	S4'	S10'

(a)

Q2 \ Q1 Q0	00	01	11	10
0	S0			
1				

Q2 \ Q1 Q0	00	01	11	10
0	S0			
1	S7'			S10'

Q2 \ Q1 Q0	00	01	11	10
0	S0		S3'	
1	S7'		S4'	S10'

Q2 \ Q1 Q0	00	01	11	10
0	S0	S1	S3'	
1	S7'	S2	S4'	S10'

(b)

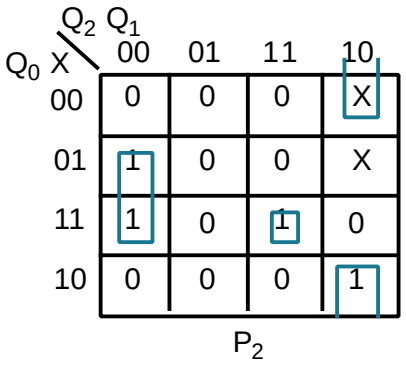
00 = Reset = S0

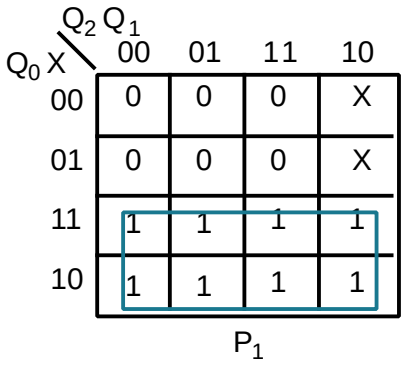
(S1, S2), (S3', S4'), (S7', S10') placed adjacently

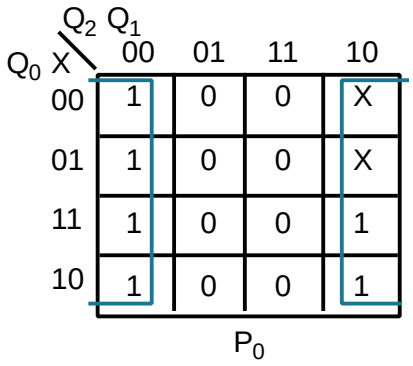
State Assignment

Effect of Adjacencies on Next State Map

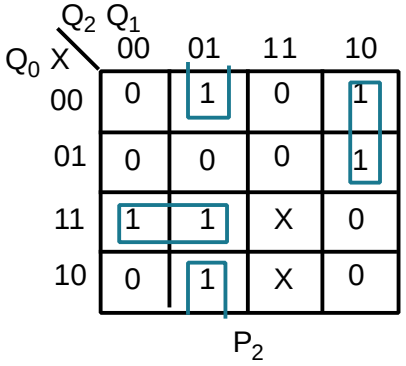
Current State	Next State X = 0	X = 1
(S_0) 000	001	101
(S_1) 001	011	111
(S_2) 101	111	011
(S'_3) 011	010	010
(S'_4) 111	010	110
(S'_7) 010	000	000
(S'_{10}) 110	000	000

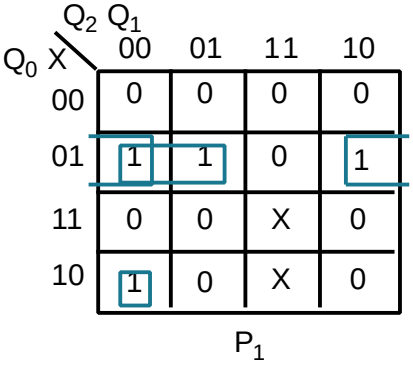


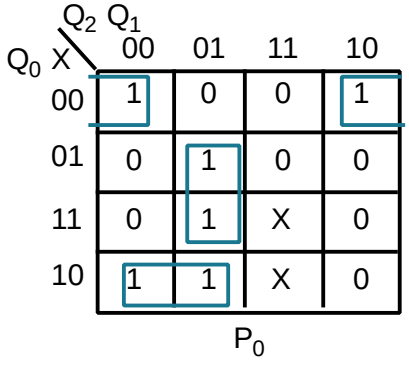




Current State	Next State X = 0	X = 1
(S_0) 000	001	010
(S_1) 001	011	100
(S_2) 010	100	011
(S'_3) 011	101	101
(S'_4) 100	101	110
(S'_7) 101	000	000
(S'_{10}) 110	000	000







First encoding exhibits a better clustering of 1's in the next state map