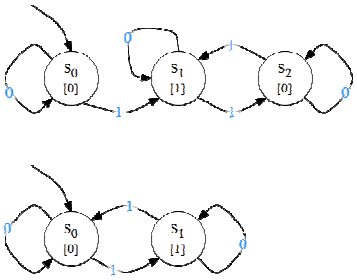
Lecture 21

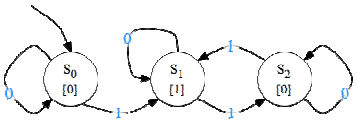
- State minimization via implication charts

Row matching not optimal

- Two FSMs for odd parity checking



No rows match!



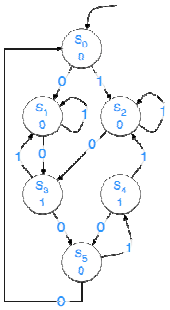
Present State	Next State		Output
	X=0	X=1	
S_0	S_0	S_1	0
S_1	S_1	S_2	1
S_2	S_2	S_1	0

FSM minimization

- Row matching
  - Easier to do by hand
  - Misses minimization opportunities
- Implication chart
  - Guaranteed to find the most reduced FSM
  - More complicated algorithm (but still relatively easy to write a program to do it)

Implication chart method

- Row matching will not work



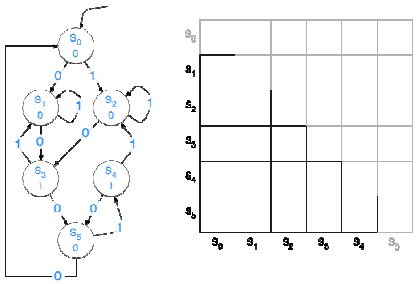
Implication chart method

- Basic Idea: Assume that all states are grouped together and only split state pairs that must be split

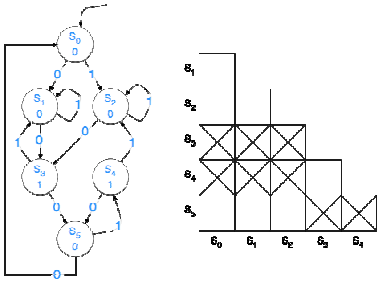
Implication chart method

1. Draw a table with a box for every pair of states
2. Cross out boxes for pairs of states with different outputs
3. Fill rest of table with transition pairs
   - For each box and possible input, list pairs of destination states, one per source state
4. Repeat until no more boxes can be crossed out:
   - If box (Si,Sj) contains some transition pair Sk-Sl corresponding to an already crossed-out box then cross out box (Si,Sj)
5. Combine all pairs of states that are left

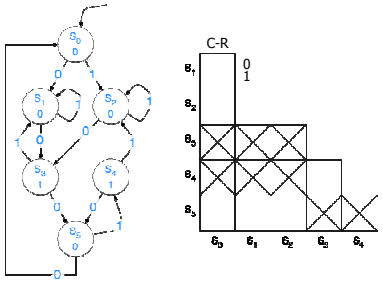
Step 1: Draw table of state pairs



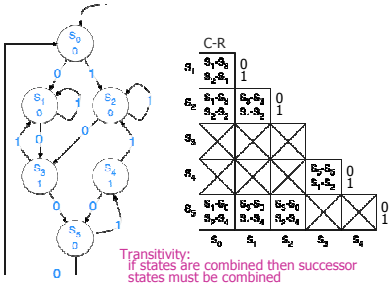
Step 2: Consider the outputs



Step 3: Add transition pairs



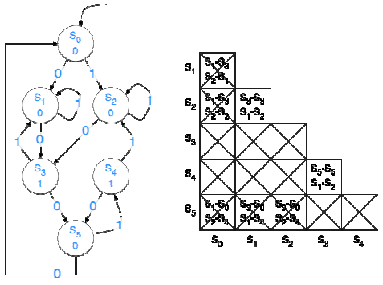
Step 3: Add transition pairs



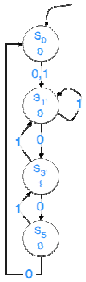
Transitivity:
if states are combined then successor states must be combined

⇒ if successor states cannot be combined then states cannot be combined

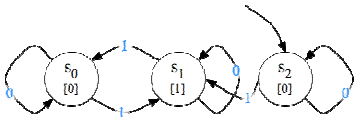
Step 4: Consider transitions



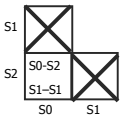
Final reduced FSM



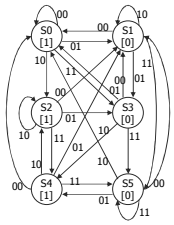
Odd parity checker revisited



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



More complex minimization



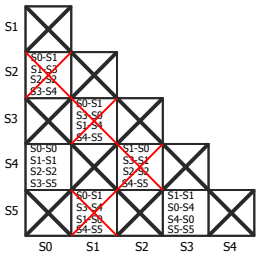
inputs here

present 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

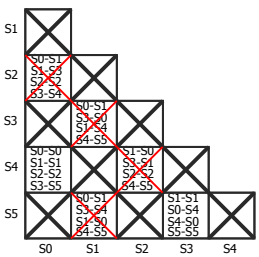
symbolic state transition table

Implication chart

present 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



Minimized FSM



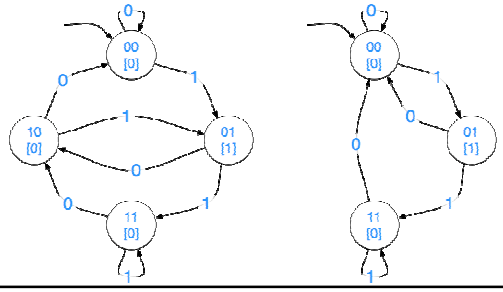
present state	00	01	10	11	output
S0	S0	S1	S2	S3	1
S1	S0	S3	S1	S4	0
S2	S1	S3	S2	S4	1
S3	S1	S0	S4	S5	0
S4	S0	S1	S2	S5	1
S5	S1	S4	S0	S5	0

present 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

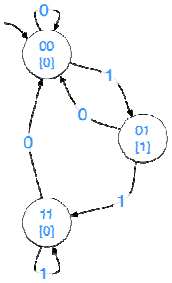
minimized state table (S0==S4) (S3==S5)

Minimizing not always good

- Two FSMs for 0 → 1 edge detection



Minimizing not always good



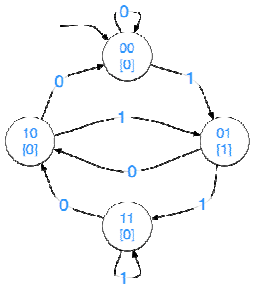
In	Q_1	Q_0	Q_1^+	Q_0^+
0	0	0	0	0
0	0	1	0	0
0	1	1	0	0
1	0	0	0	1
1	0	1	1	1
1	1	1	1	1
–	1	0	–	–

$Q_1^+ = \text{In}\ Q_0$

$Q_0^+ = \text{In}$

$\text{Out} = Q_1' Q_0$

Minimizing not always good



In	Q_1	Q_0	Q_1^+	Q_0^+
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	1	0
1	0	0	0	1
1	0	1	1	1
1	1	0	0	1
1	1	1	1	1

$Q_1^+ = Q_0$

$Q_0^+ = \text{In}$

$\text{Out} = Q_1' Q_0$

A little perspective

- These kinds of optimizations are what CAD (Computer Aided Design) / EDA (Electronic Design Automation) is all about
- The interesting problems are almost always computationally intractable to solve optimally
- People **really** care about the automation of the design of billion-transistor chips