CS221: Digital Design

http://jatinga.iitg.ernet.in/~asahu/cs221

FSM: Optimization and State Encoding

A. Sahu

Dept of Comp. Sc. & Engg.

Indian Institute of Technology Guwahati

Outline

- FSM State Optimization : RM and IC
- FSM State Encoding
- FSM + Data Path
- ASM

FSM State Minimization

FSM State Minimization

- Minimizing number of state reduce
  - Requirement of bigger size state register
  - Possibly reduce the CCC

Some Definitions

- **State Equivalence**: S1 and S2 are equivalent if for every input sequence applied to machine goes to same NS and Output
  - If S1(t+1)=S2(t+1) and Z1=Z2 then S1=S2

- **Distinguishable States**: Two states S1 and S2 are Distinguishable iff there exist at least one finite input sequence which produce different outputs from S1 and S2

Methods

- Row Matching Method or Partitioning Method
  - Completely specified machine (n^2 edges)
  - Partially specified machine
- Implication Chart Method

Implication Chart Methods

FSM Reduction: Implication Chart Method

Problem:

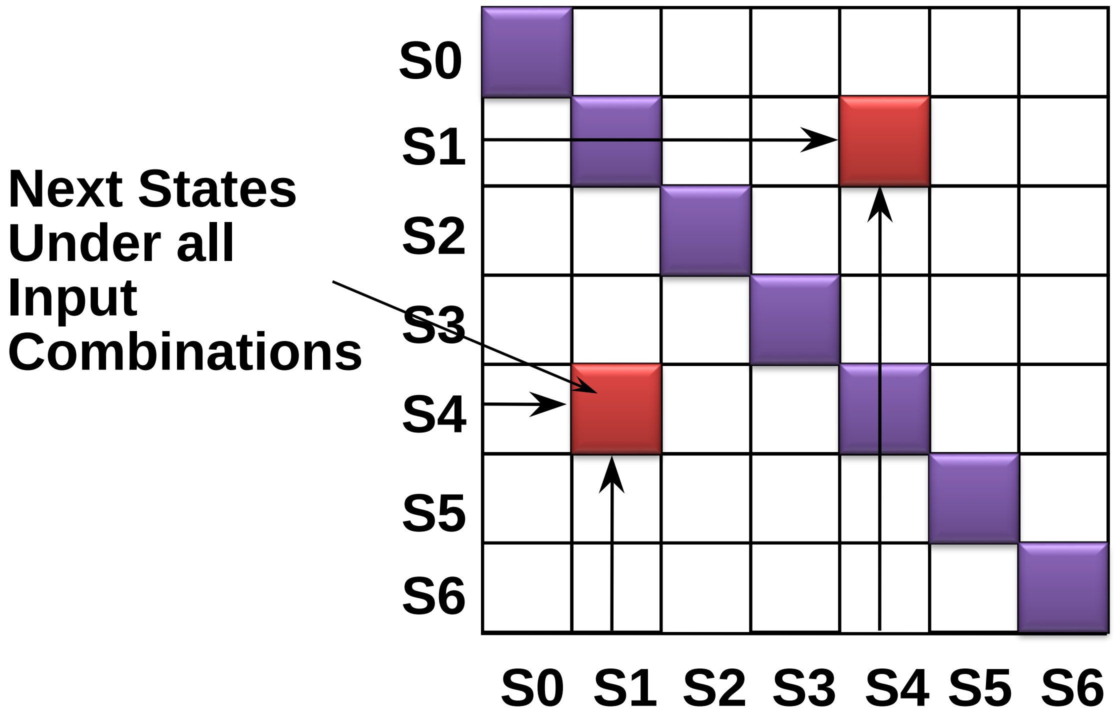
Single input X, Single output Z

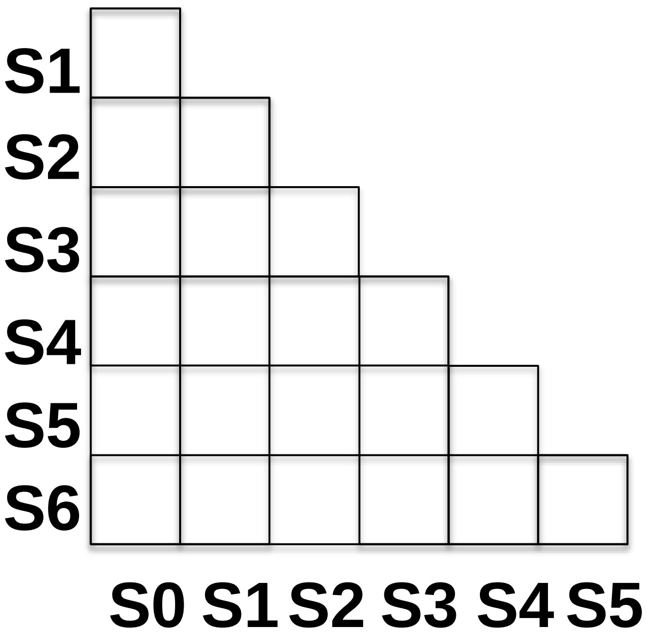
Output a 1 whenever the serial sequence 010 or 110 has been observed at the inputs

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

Implication Chart Method

Enumerate all possible combinations of states taken two at a time





Implication Chart

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

Implication Chart Method

Filling in the Implication Chart

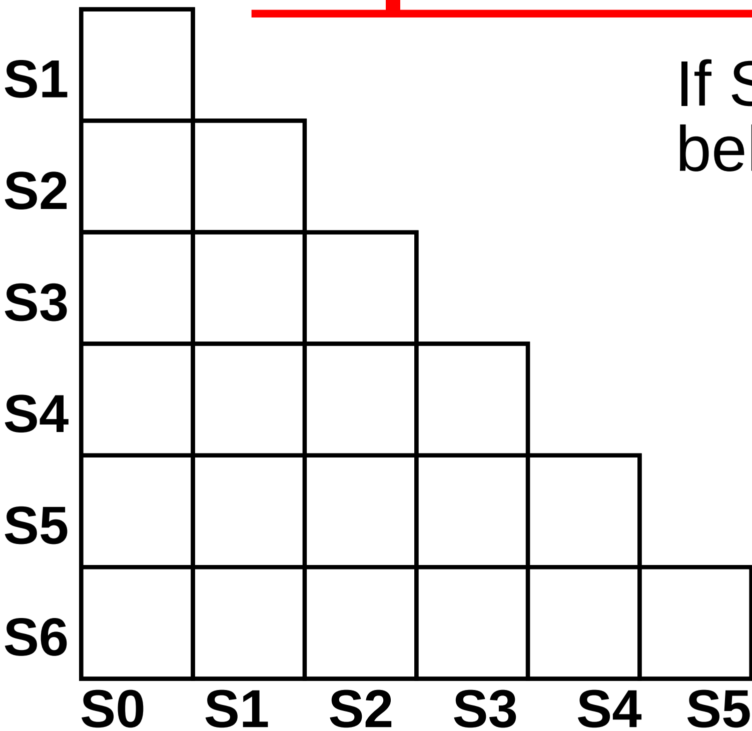
Entry $\mathbf{X_{ij}}$ — Row is $\mathbf{S_i}$, Column is $\mathbf{S_j}$

$\mathbf{S_i}$ is equivalent to $\mathbf{S_j}$ if outputs are the same and next states are equivalent

$\mathbf{X_{ij}}$ contains the next states of $\mathbf{S_i}$, $\mathbf{S_j}$ which must be equivalent if S_i and S_j are equivalent

If $\mathbf{S_i}$, $\mathbf{S_j}$ have different output behavior, then $\mathbf{X_{ij}}$ is crossed out

Implication Chart Method

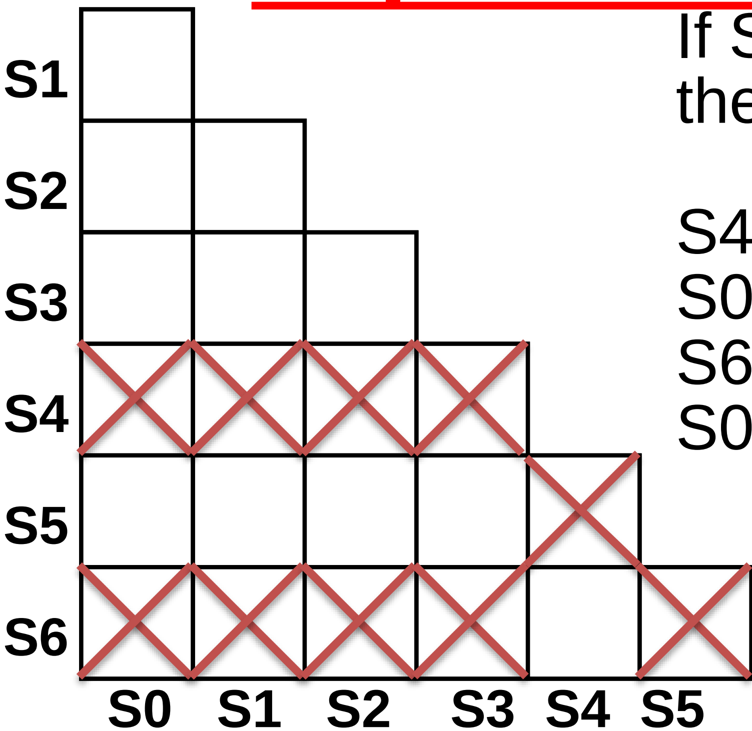


Starting Implication Chart

If S_i, S_j have different output behavior, then X_{ij} is crossed out

Input	P State	NS X=0	NS 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

Implication Chart Method



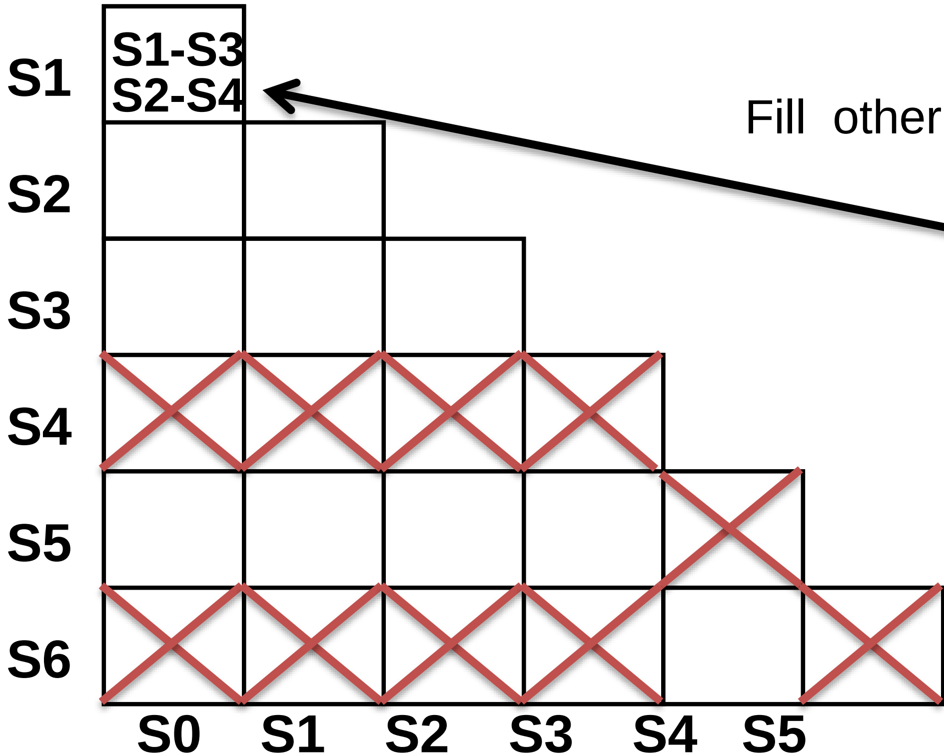
Starting Implication Chart

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

S4 have different out put behavior with S0, S1, S 2, S3, S5
S6 have different out put behavior with S0, S1, S 2, S3, S5

Input	P State	NS X=0	NS 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

Implication Chart Method



Starting Implication Chart

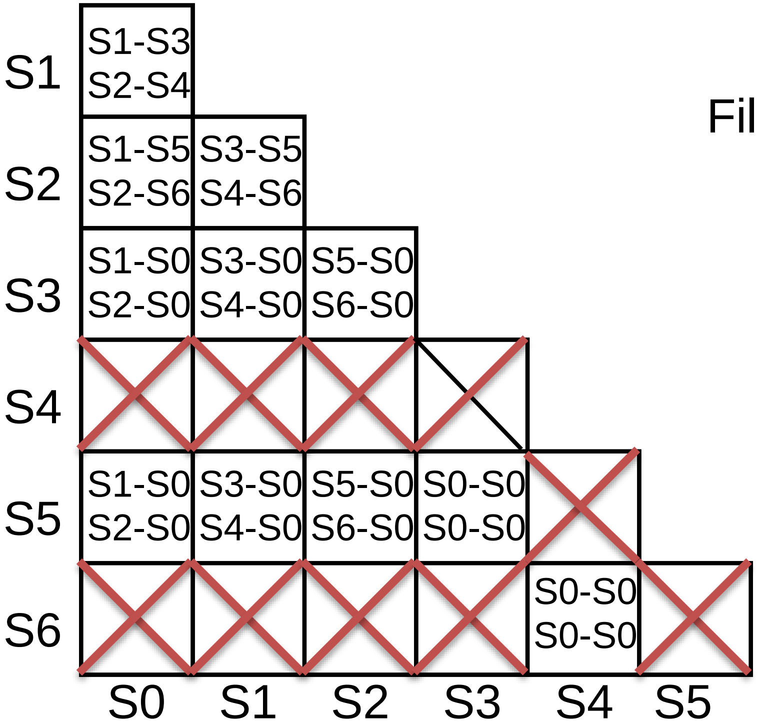
Fill other Sij Based on Next State Transaction

S0 : S1, S2 when X=0, 1

S1 : S3, S4 when X=0,1

Input	P State	NS X=0	NS 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

Implication Chart Method

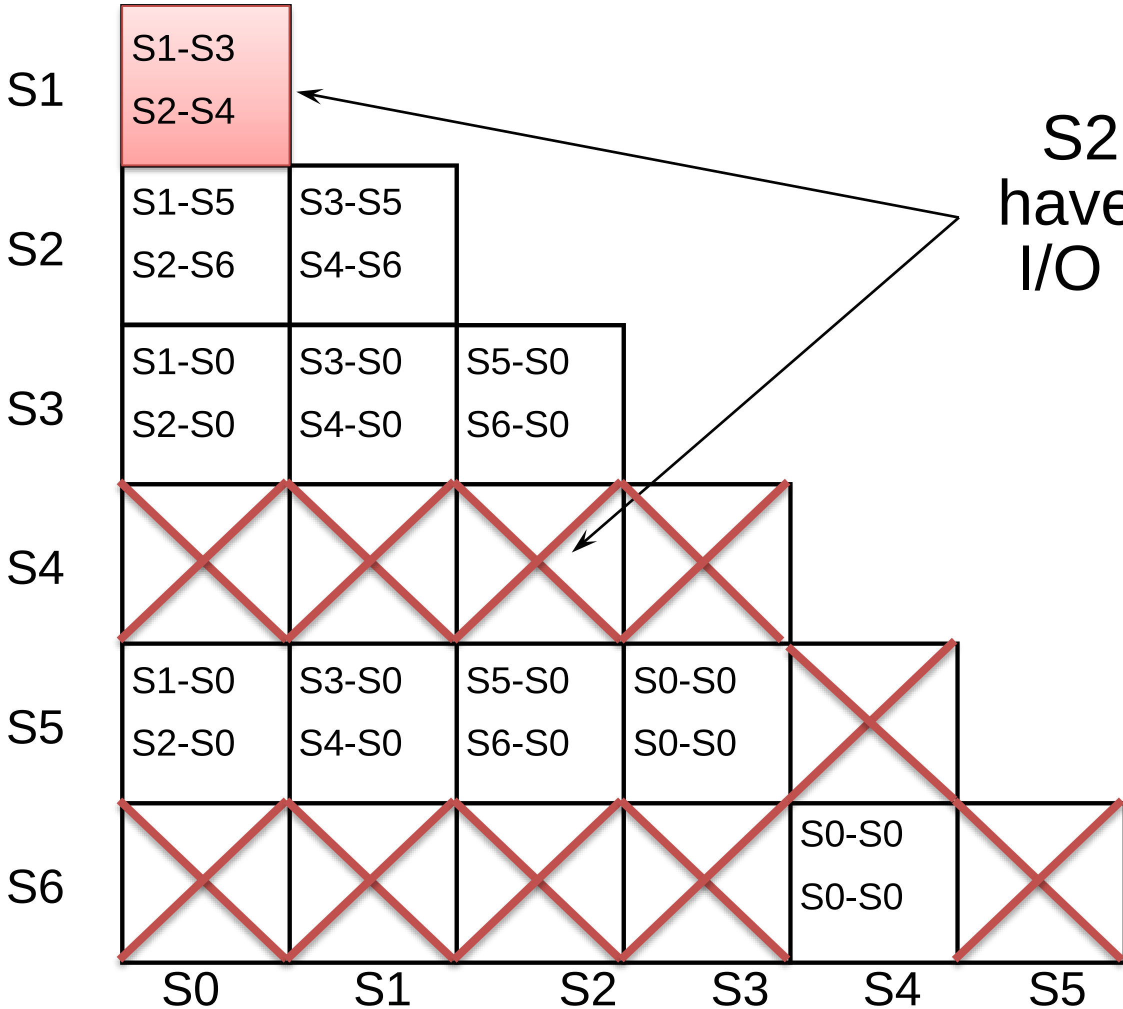


Starting Implication Chart

Fill others Sij Based on Next State Transaction

Input	P State	NS X=0	NS 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

Implication Chart Method

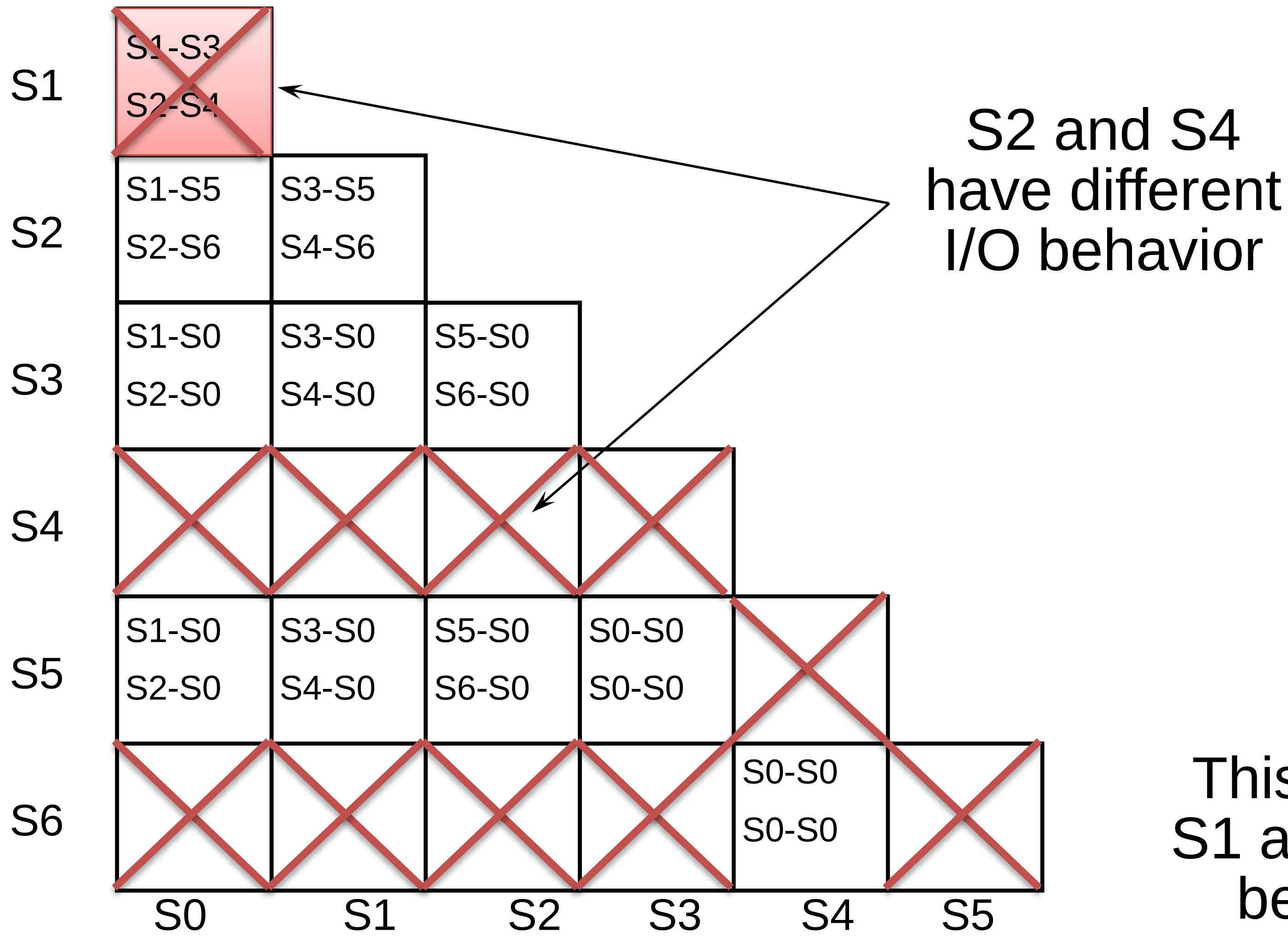


S2 and S4 have different I/O behavior

This implies that S1 and S0 cannot be combined

Marked with Cross

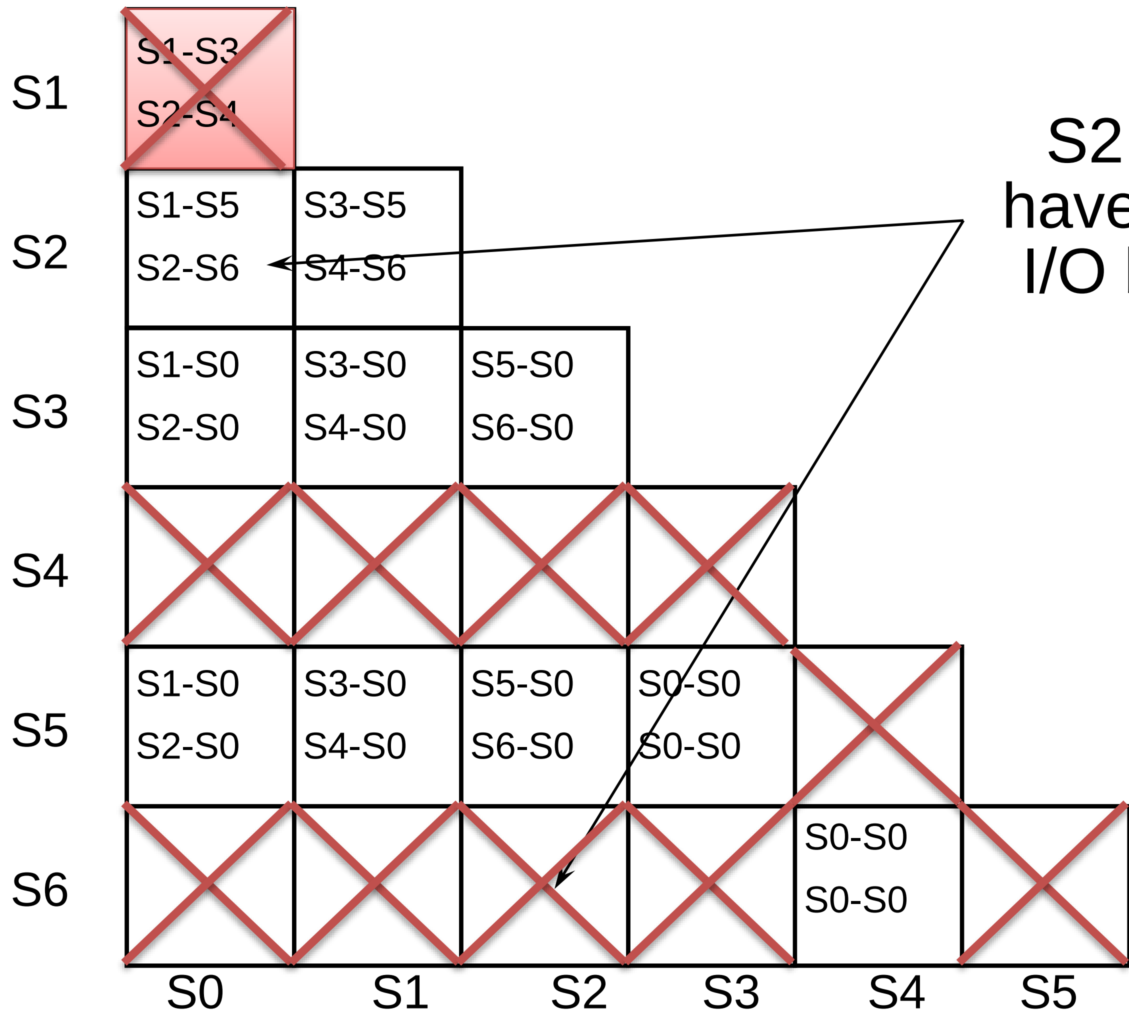
Implication Chart Method



This implies that S1 and S0 cannot be combined

Marked with Cross

Implication Chart Method

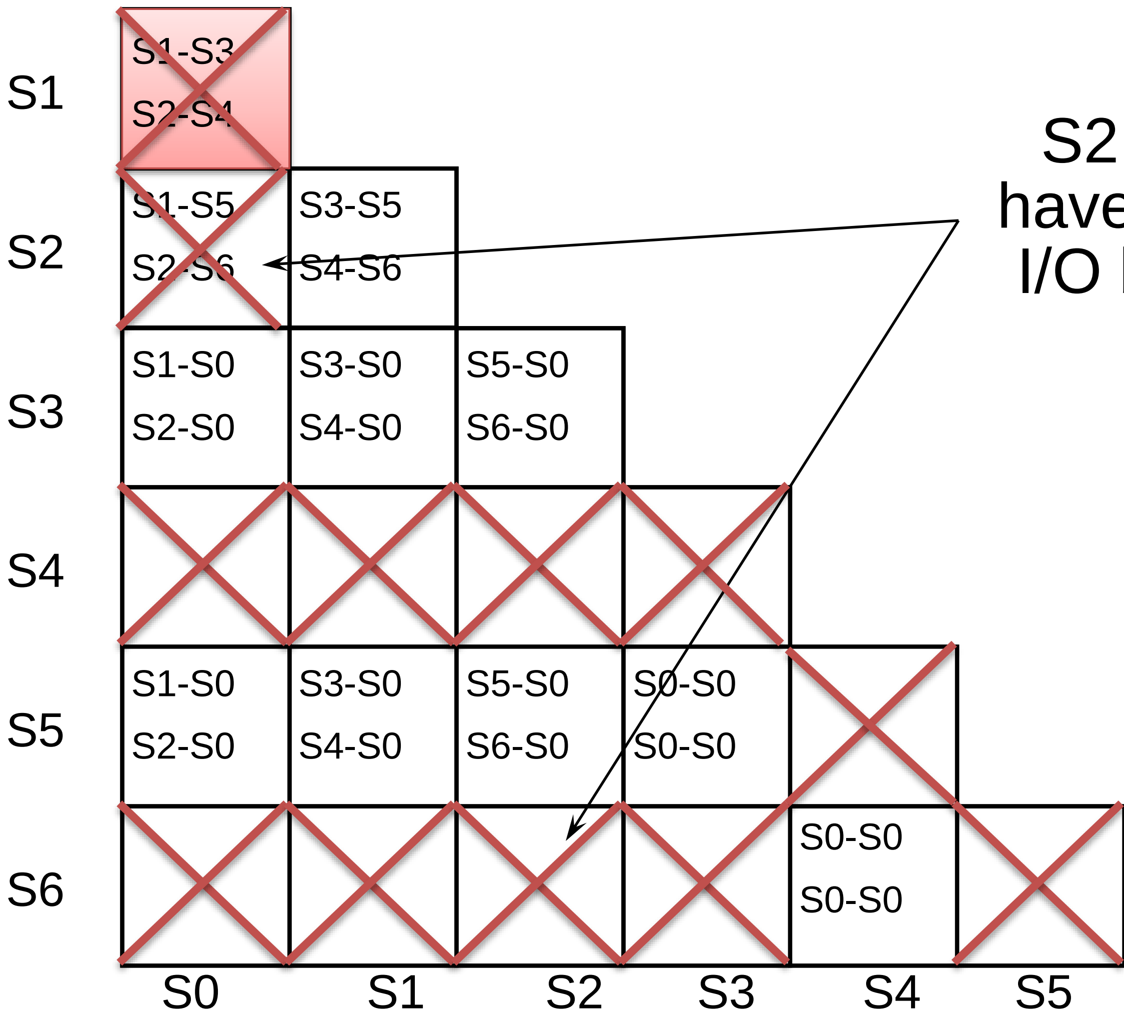


S2 and S6 have different I/O behavior

This implies that S1 and S0 cannot be combined

Marked with Cross

Implication Chart Method

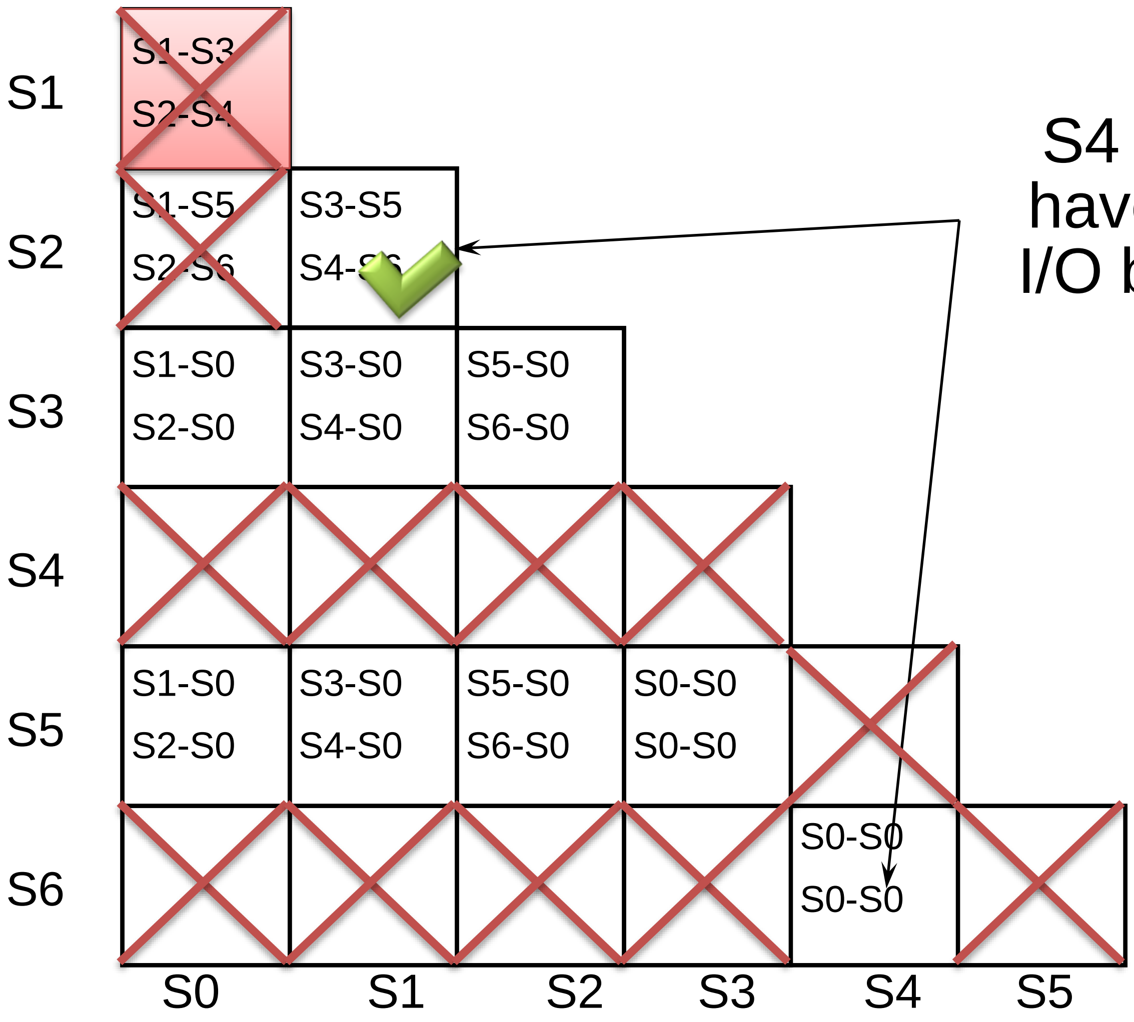


S2 and S6 have different I/O behavior

This implies that S2 and S0 cannot be combined

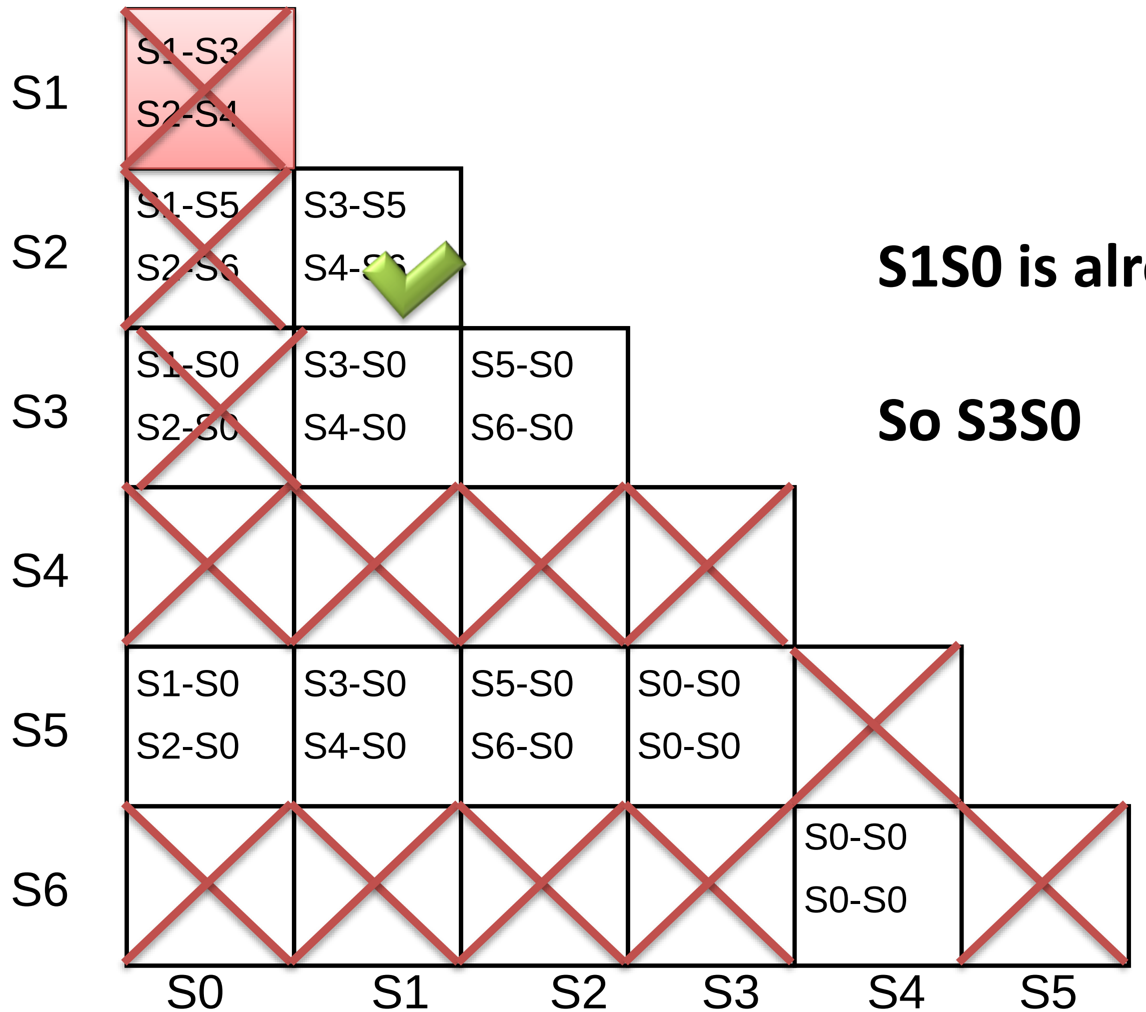
Marked with Cross

Implication Chart Method



S4 and S6 have **Same** I/O behavior

Implication Chart Method



S1S0 is already crossed

So S3S0

Implication Chart Method

	S0	S1	S2	S3	S4	S5
S1	S1-S3 S2-S4					
S2	S1-S5 S2-S6	S3-S5 S4-S6				
S3	S1-S0 S2-S0	S3-S0 S4-S0	S5-S0 S6-S0			
S4						
S5	S1-S0 S2-S0	S3-S0 S4-S0	S5-S0 S6-S0	S0-S0 S0-S0		
S6					S0-S0 S0-S0	

S4S0 is already crossed

So S3S1

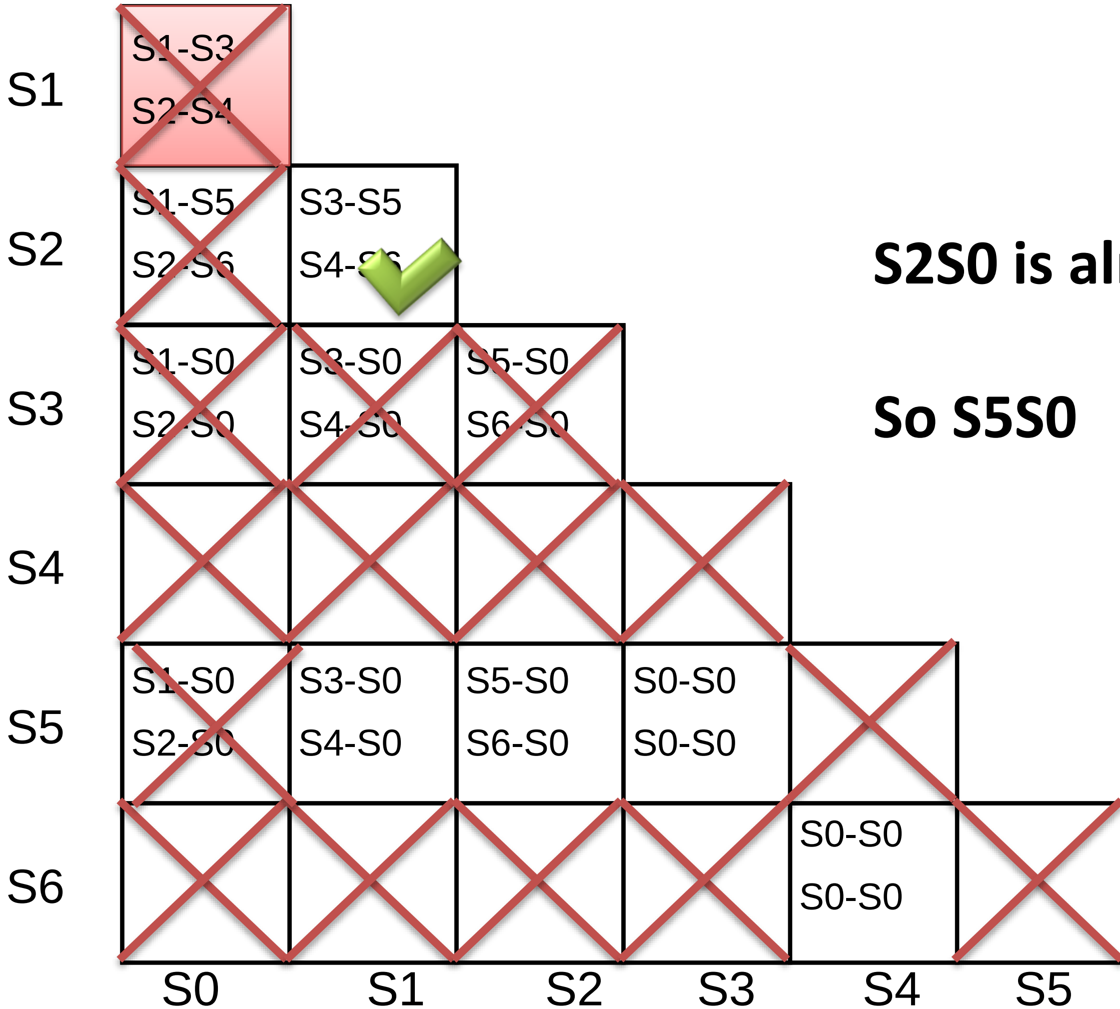
Implication Chart Method

	S0	S1	S2	S3	S4	S5
S1	~~S1-S3~~ ~~S2-S4~~					
S2	~~S1-S5~~ ~~S2-S6~~	S3-S5 S4-S6				
S3	~~S1-S0~~ ~~S2-S0~~	~~S3-S0~~ ~~S4-S0~~	~~S5-S0~~ ~~S6-S0~~			
S4						
S5	S1-S0 S2-S0	S3-S0 S4-S0	S5-S0 S6-S0	S0-S0 S0-S0		
S6					S0-S0 S0-S0	

S6S0 is already crossed

So S3S2

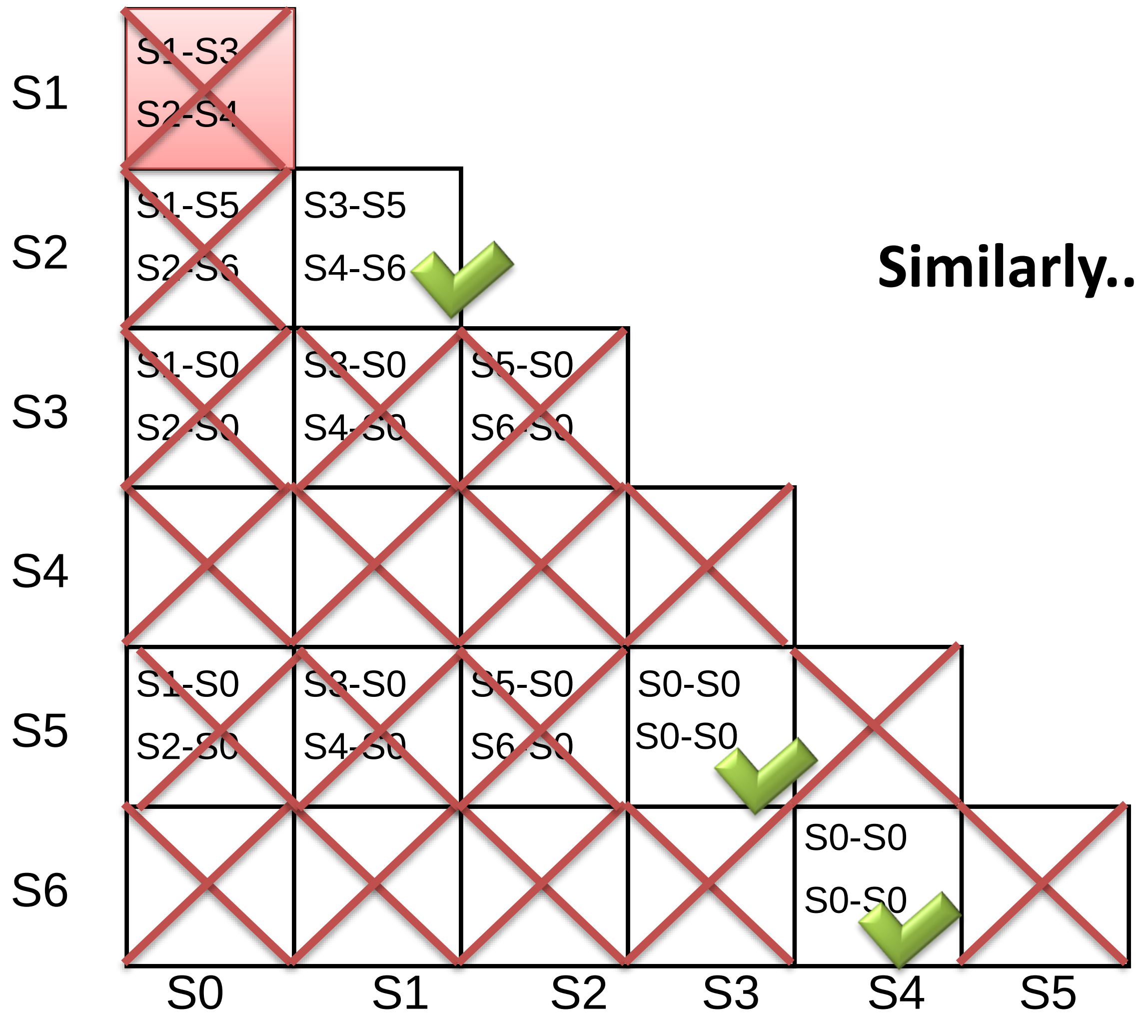
Implication Chart Method



S2S0 is already crossed

So S5S0

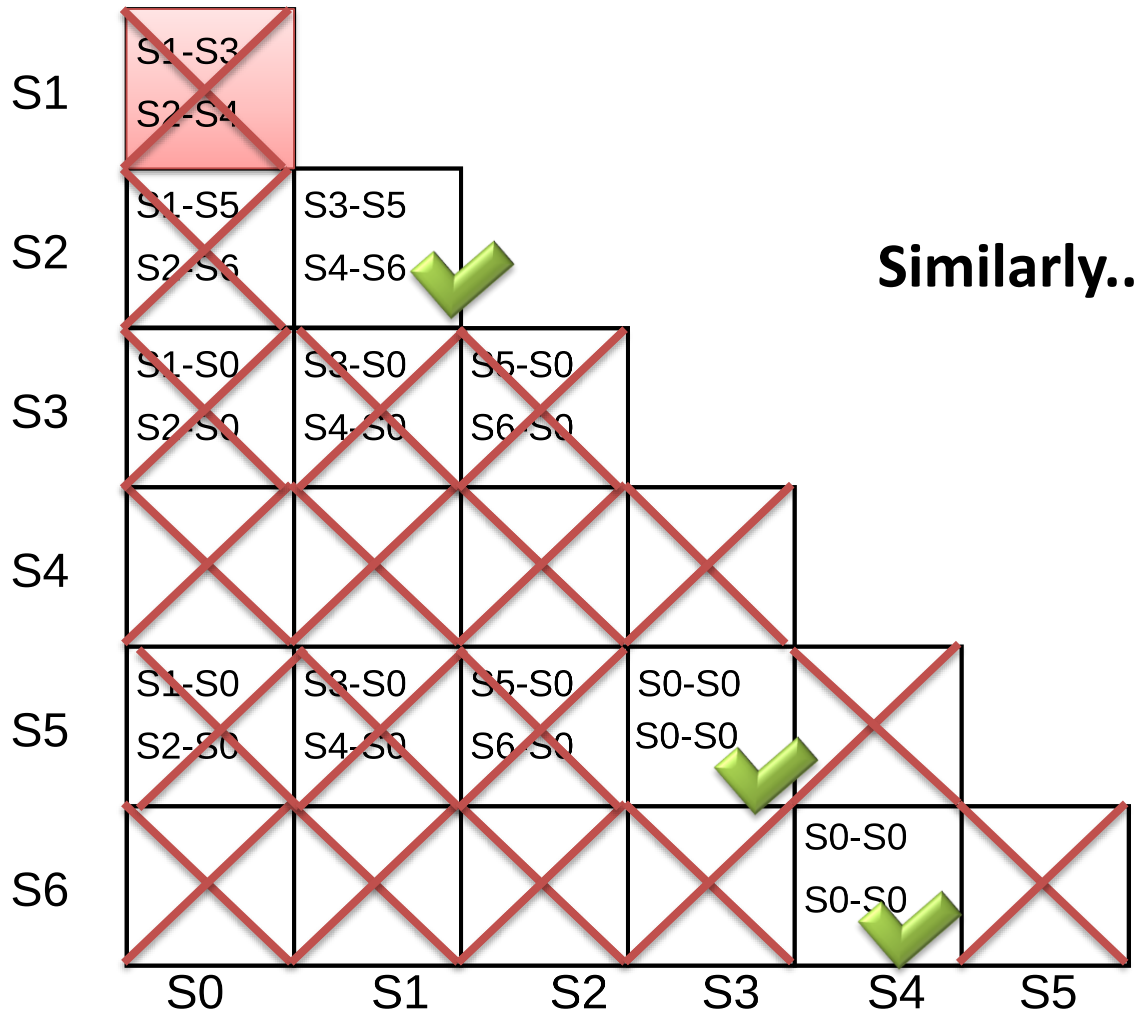
Implication Chart Method



Similarly..

Second Pass Adds No New Information

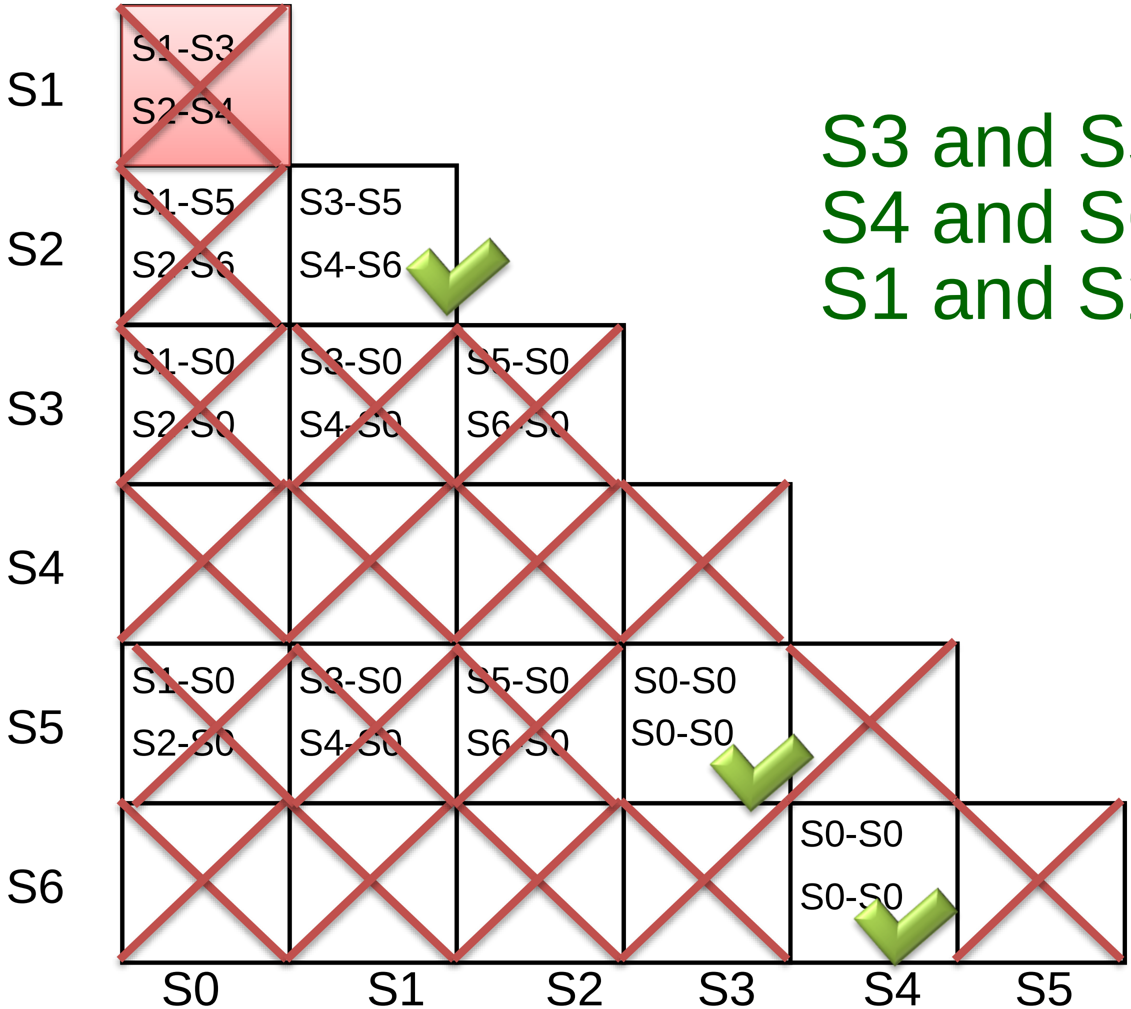
Implication Chart Method



Similarly..

Second Pass Adds No New Information

Implication Chart Method



S3 and S5 are equivalent
S4 and S6 are equivalent
S1 and S2 are equivalent

Second Pass Adds No New Information

Final : Reduce State Tabe

- Reduces State table

Input	P State	NS X=0	NS 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

Input	PS	NS		Output	
		X=0	X=1	X=0	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

State Encoding/Assignment

State assignment

- Since we don't care about the actual flip-flop values for each state we can assign each state to any binary number we like **as long as** each state is assigned a unique binary number
- Suppose a FSM have 5 states
- If we use 3 bits to encode the 5 states, we have

$$\binom{8}{5} = \frac{8!}{5!(8-5)!}$$

possible encodings

State assignment

state	Encoding 1 (binary)	Encoding 2 (Gray)	Encoding 3
a	000	000	000
b	001	001	100
c	010	011	010
d	011	010	101
e	100	110	011

State Encoding

- Hamming/Edit Distance
  - HD(0000,0100)=1, HD(0011,1100)=4, HD(1010,1111)=2
- Binary and Gray encoding use the minimum number of bits for state register
- Gray and Johnson code: Two adjacent codes differ by only one bit
  - Reduce simultaneous switching
  - Reduce crosstalk, Reduce glitch

One-hot encoding

- One flip-flop per state encoding
- Leads to greater number of flip-flops than binary encoding but possibly to simpler o/p logic