L8 – Reduction of State Tables

Reduction of states

- Given a state table reduce the number of states.
- Eliminate redundant states

- Ref: text Unit 15

Objective

- Reduce the number of states in the state table to the minimum.
  - Remove redundant states
  - Use don't cares effectively
- Reduction to the minimum number of states reduces
  - The number of F/Fs needed
  - Reduces the number of next states that has to be generated → Reduced logic.

An example circuit

- From 14.3, example 1
  - A sequential circuit has one input X and one output Z. The circuit looks at the groups of four consecutive inputs and sets Z=1 if the input sequence 0101 or 1001 occurs. The circuit returns to the reset state after four inputs. Design the Mealy machine.
- Typical sequence
  - X = 0101 0010 1001 0100
  - Z= 0001 0000 0001 0000

A state table for this

- Set up a table for all the possible input combinations (versus rationalizing the development of a state graph).
- For the two sequences when the 4th input completes a sequence, return to reset with Z=1.

Input Sequence	Present State	Next State X = 0	Next State X = 1	Present Output X = 0	Present Output X = 1
reset	A	B	C	0	0
0	B	D	E	0	0
1	C	F	G	0	0
00	D	H	I	0	0
01	E	J	K	0	0
10	F	L	M	0	0
11	G	N	P	0	0
000	H	A	A	0	0
001	I	A	A	0	0
010	J	A	A	0	1
011	K	A	A	0	0
100	L	A	A	0	1
101	M	A	A	0	0
110	N	A	A	0	0
111	P	A	A	0	0

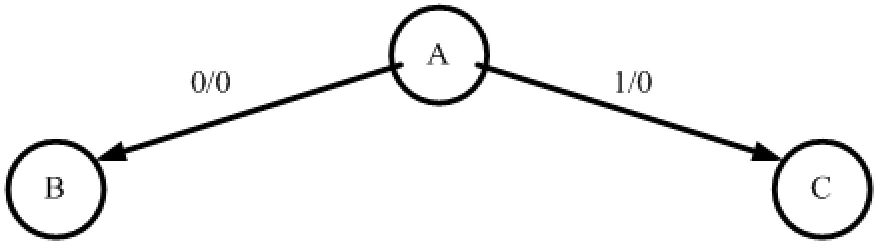
Notes on state table generation

- When generated by looking at all combinations of inputs the state table is far from minimal.

- First step is to remove redundant states.
  - There are states that you cannot tell apart
    - Such as H and I – both have next state A with Z=0 as output.
    - State H is equivalent to State I and state I can be removed from the table.
    - Examining table shows states K, M, N and P are also the same as I was – they can be deleted.
    - States J and L are also equivalent.

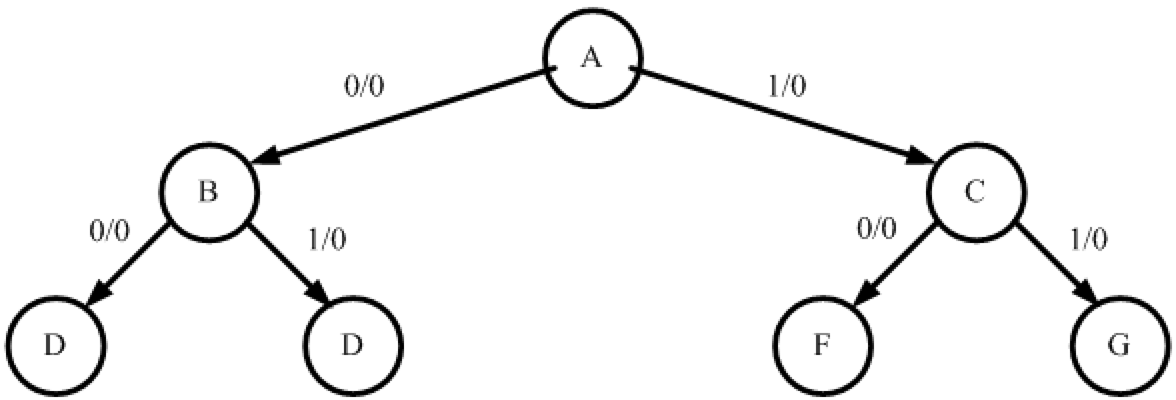
Can take state table to graph

- Reset and states B and C
- Will also be able to see redundancies in graph



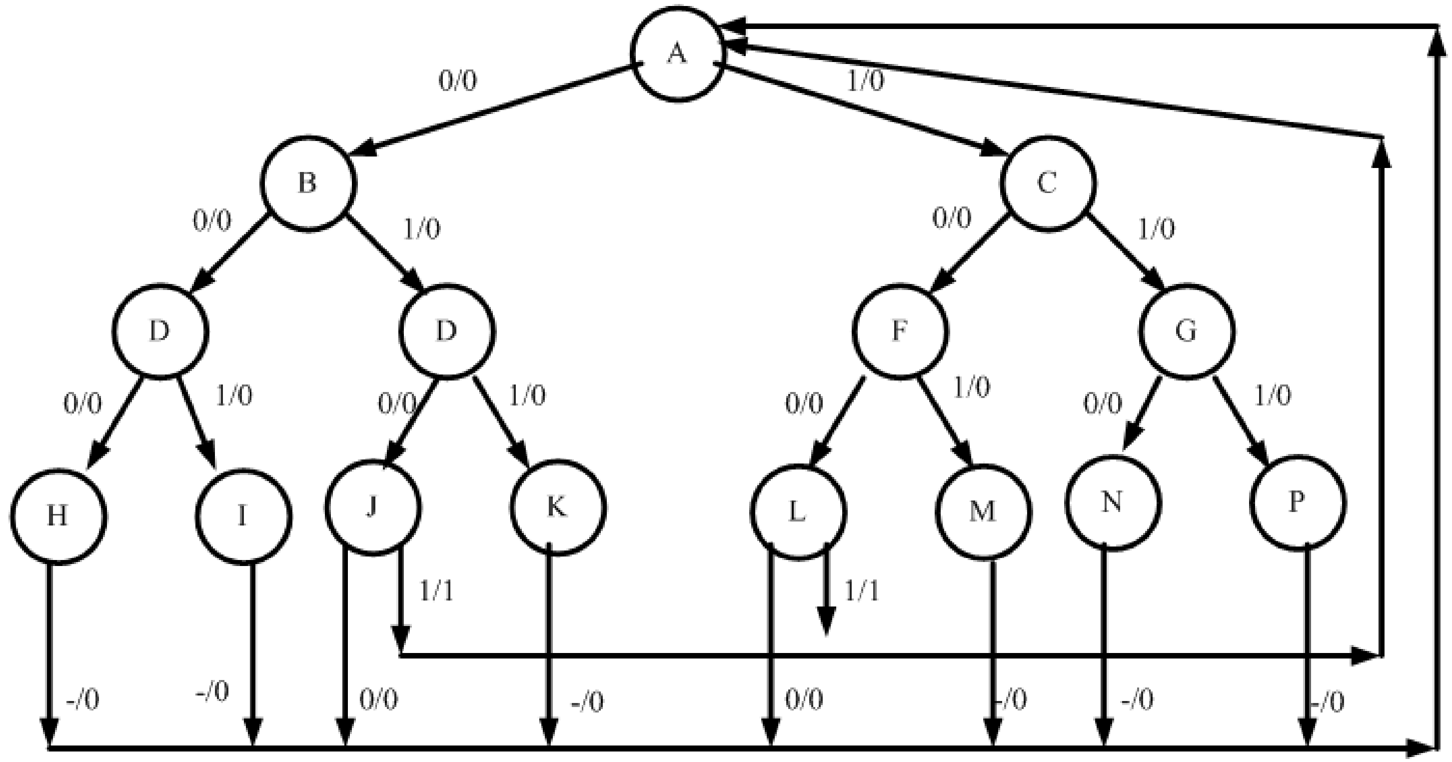
The next level

- Now add D, E,F, G



And the final level

- Adding state H,I,J,K,L,M,N,P



1st state reduction

- First need to indicate that H, I, K, M, N and P are the same
- AND J and L are the same
- So remove all but H and J

Input Sequence	Present State	Next State X = 0	Next State X = 1	Present Output X = 0	Present Output X = 1
reset	A	B	C	0	0
0	B	D	E	0	0
1	C	F	G	0	0
00	D	H	~~I~~ H	0	0
01	E	J	~~K~~ H	0	0
10	F	~~L~~ J	~~M~~ H	0	0
11	G	~~N~~ H	~~P~~ H	0	0
000	H	A	A	0	0
~~001~~	~~I~~	~~A~~	~~A~~	~~0~~	~~0~~
010	J	A	A	0	1
~~011~~	~~K~~	~~A~~	~~A~~	~~0~~	~~0~~
~~100~~	~~L~~	~~A~~	~~A~~	~~0~~	~~1~~
~~101~~	~~M~~	~~A~~	~~A~~	~~0~~	~~0~~
~~110~~	~~N~~	~~A~~	~~A~~	~~0~~	~~0~~
~~111~~	~~P~~	~~A~~	~~A~~	~~0~~	~~0~~

Reduction continued

- Having made these reductions move up to the D E F G section where the next state entries have been changed.
- Note that State D and State G are equivalent.
- State E is equivalent to F.
- The result is a reduced state table.

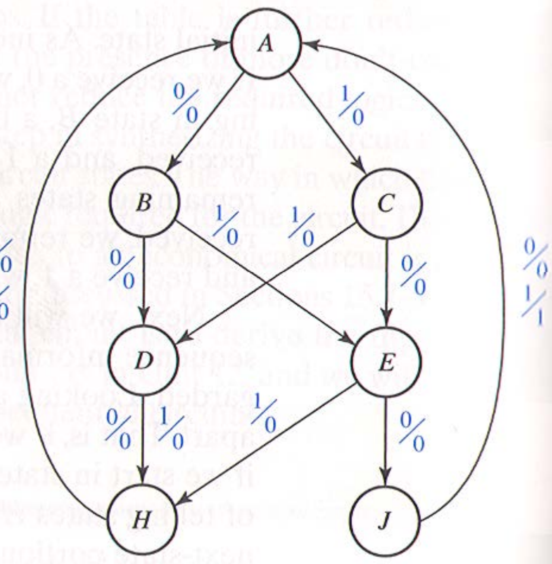
Present State	Next State X = 0	Next State X = 1	Present Output X = 0	Present Output X = 1
A	B	C	0	0
B	D	E	0	0
C	~~F~~ E	~~G~~ D	0	0
D	H	~~I~~ H	0	0
E	J	~~K~~ H	0	0
~~F~~	~~L J~~	~~M H~~	~~0~~	~~0~~
~~G~~	~~N H~~	~~R H~~	~~0~~	~~0~~
H	A	A	0	0
~~I~~	~~A~~	~~A~~	~~0~~	~~0~~
J	A	A	0	1
~~K~~	~~A~~	~~A~~	~~0~~	~~0~~
~~L~~	~~A~~	~~A~~	~~0~~	~~1~~
~~M~~	~~A~~	~~A~~	~~0~~	~~0~~
~~N~~	~~A~~	~~A~~	~~0~~	~~0~~
~~P~~	~~A~~	~~A~~	~~0~~	~~0~~

The result

- Reduced state table and graph

Present State	Next State X = 0	Next State X = 1	Output X = 0	Output X = 1
A	B	C	0	0
B	D	E	0	0
C	E	D	0	0
D	H	H	0	0
E	J	H	0	0
H	A	A	0	0
J	A	A	0	1

(a)



(b)

- Original – 15 states – reduced to 7 states

Equivalence

- Two states are equivalent if there is no way of telling them apart through observation of the circuit inputs and outputs.
- Formal definition
  - Let N_1 and N_2 be sequential circuits (not necessarily different). Let $\underline{X}$ represent a sequence of inputs of arbitrary length. Then state *p* in N_1 is equivalent to state *q* in N_2 iff $\lambda_1(p,\underline{X}) = \lambda_2(q,\underline{X})$ for every possible input sequence $\underline{X}$.
- The definition is not practical to apply in practice.

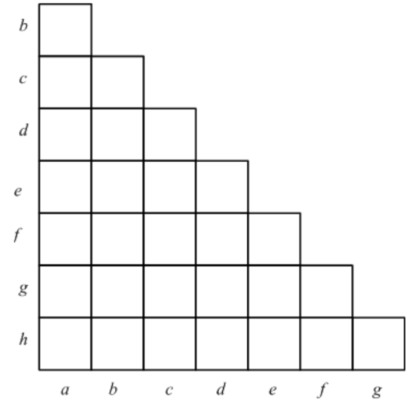
As not practical

- Theorem 15.1
  - Two states *p* and *q* of a sequential circuit are equivalent iff for every single input X, the outputs are the same and the next states are equivalent, that is, $\lambda(p,\underline{X}) = \lambda(q,\underline{X})$ and $\delta(p,\underline{X}) \equiv \delta(q,\underline{X})$ where $\lambda(p,\underline{X})$ is the output given present state *p* and input X, and $\delta(p,\underline{X})$ is the next state given the present state *p* and input X.
- So the outputs have to be the same and the next states equivalent.

Implication Tables

- Now a procedure for finding all the equivalent states in a state table.
- Use an implication table – a chart that has a square for each pair of states.

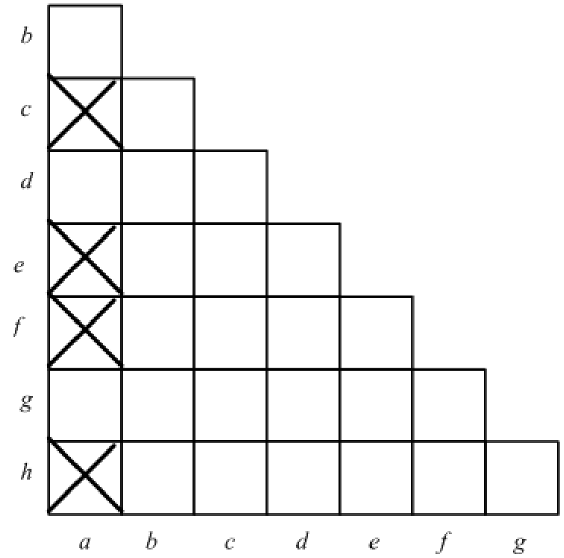
Present State	Next State $X = 0$	1	Present Output
a	*d*	*c*	0
b	*f*	*h*	0
c	*e*	*d*	1
d	*a*	*e*	0
e	*c*	*a*	1
f	*f*	*b*	1
g	*b*	*h*	0
h	*c*	*g*	1



Step 1

- Use a X in the square to eliminate output incompatible states.
- 1st output of a differes from c, e, f, and h

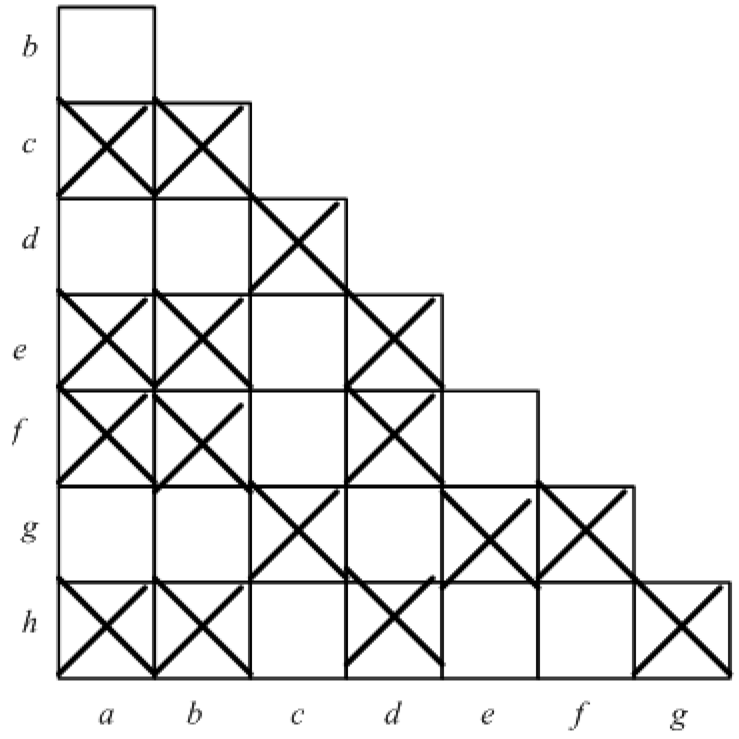
Present State	Next State $X = 0$	1	Present Output
a	*d*	*c*	0
b	*f*	*h*	0
c	*e*	*d*	1
d	*a*	*e*	0
e	*c*	*a*	1
f	*f*	*b*	1
g	*b*	*h*	0
h	*c*	*g*	1



Step 1 continued

- Continue to remove output incompatible states

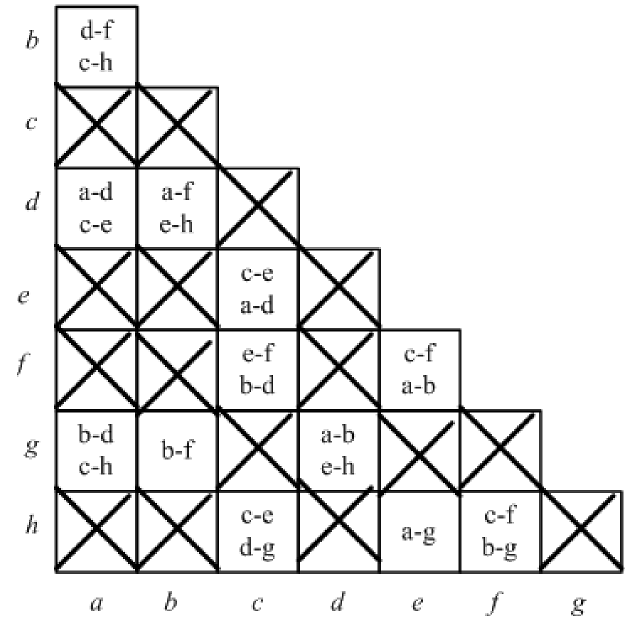
Present State	Next State $X = 0$	1	Present Output
a	*d*	*c*	0
b	*f*	*h*	0
c	*e*	*d*	1
d	*a*	*e*	0
e	*c*	*a*	1
f	*f*	*b*	1
g	*b*	*h*	0
h	*c*	*g*	1



Now what?

- *Implied pair* are now entered into each non X square.
- Here a≡b iff d≡f and c≡h

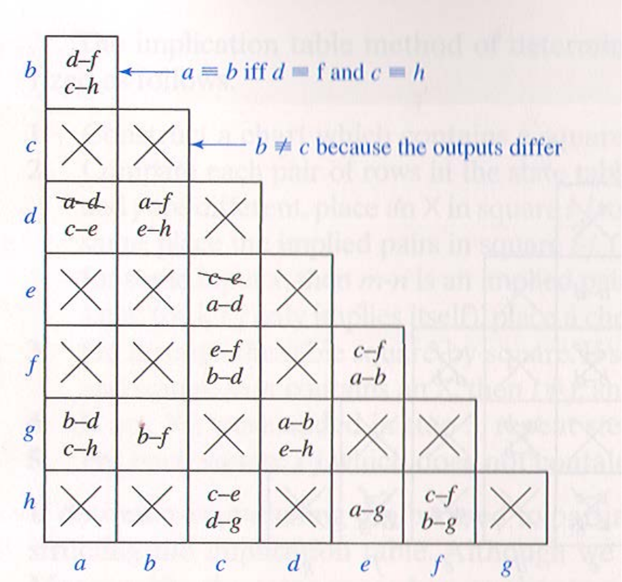
Present State	Next State X = 0	1	Present Output
a	d	c	0
b	f	h	0
c	e	d	1
d	a	e	0
e	c	a	1
f	f	b	1
g	b	h	0
h	c	g	1



Self redundant pairs

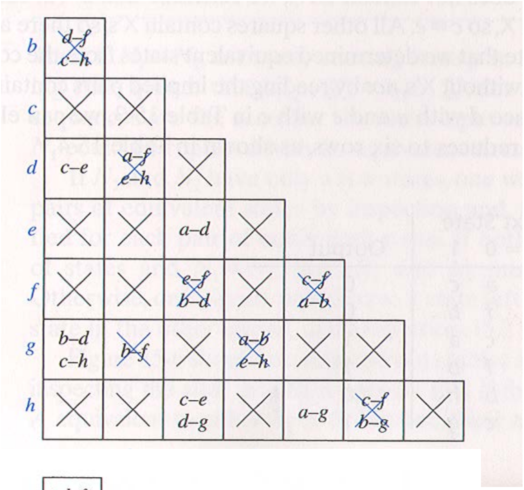
- Self redundant pairs are removed, i.e., in square a-d it contains a-d.

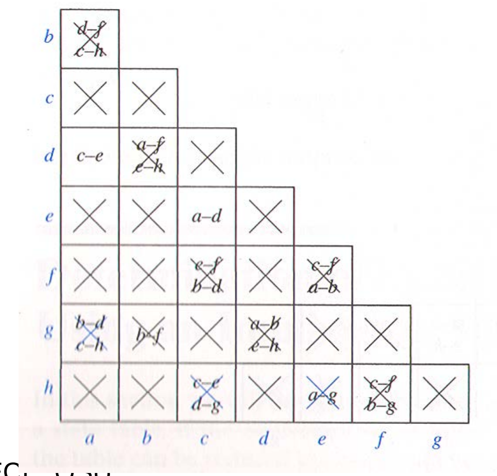
Present State	Next State X = 0	1	Present Output
a	d	c	0
b	f	h	0
c	e	d	1
d	a	e	0
e	c	a	1
f	f	b	1
g	b	h	0
h	c	g	1



Next pass

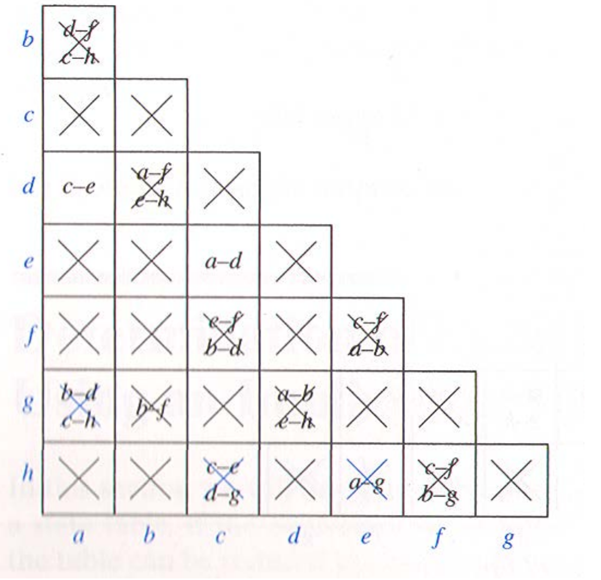
- X all squares with implied pairs that are not compatible.
- Such as in a-b have d-f which has an X in it.
- Run through the chart until no further X's are found.





Final step

- Note that a-d is not Xed – can conclude that a≡d. The same for c-e, i.e., c≡e.



Reduced table

- Removing equivalent states.

Present State	Next State X = 0	1	Present Output
a	*d*	*c*	0
b	*f*	*h*	0
c	*e*	*d*	1
d	*a*	*e*	0
e	*c*	*a*	1
f	*f*	*b*	1
g	*b*	*h*	0
h	*c*	*g*	1

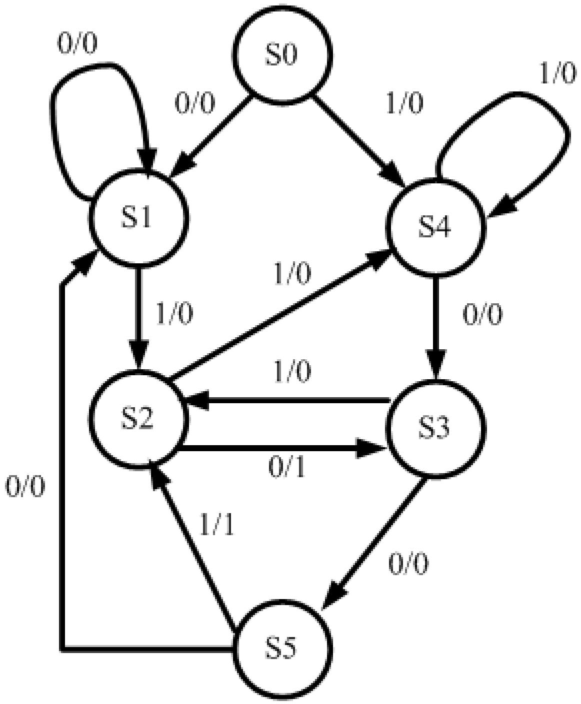
Present State	Next State X = 0	1	Output
a	*a*	*c*	0
b	*f*	*h*	0
c	*c*	*a*	1
f	*f*	*b*	1
g	*b*	*h*	0
h	*c*	*g*	1

Summary of method

- 1. construct a chart with a square for each pair of states.
- 2. Compare each pair of rows in the state table. X a square if the outputs are different. If the output is the same enter the implied pairs. Remove redundant pairs. If the implied pair is the same place a check mark as i≡j.
- 3. Go through the implied pairs and X the square when an implied pair is incompatible.
- 4. Repeat until no more Xs are added.
- 5. For any remaining squares not Xed, i≡j.

Another example

- Consider a previous circuit

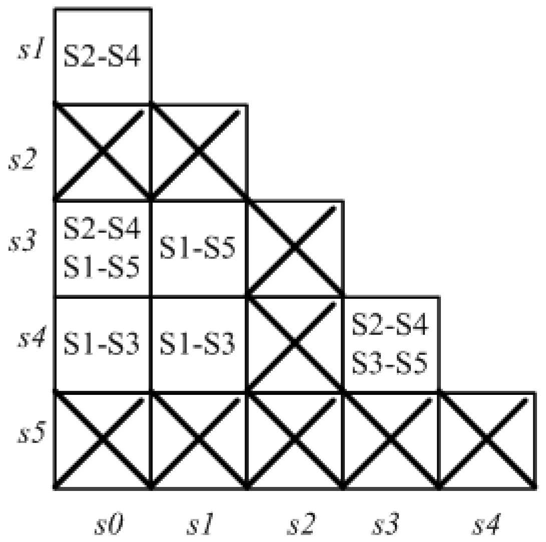


Present State	NEXT STATE X=0	NEXT STATE X=1	OUTPUT X=0	OUTPUT X=1
S0	S1	S4	0	0
S1	S1	S2	0	0
S2	S3	S4	1	0
S3	S5	S2	0	0
S4	S3	S4	0	0
S5	S1	S2	0	1

Set up Implication Chart

- And remove output incompatible states

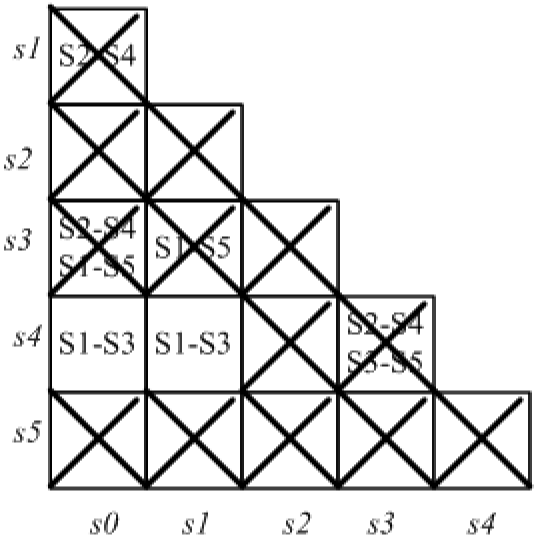
	NEXT STATE		OUTPUT	
Present State	**X=0**	**X=1**	**X=0**	**X=1**
S0	S1	S4	0	0
S1	S1	S2	0	0
S2	S3	S4	1	0
S3	S5	S2	0	0
S4	S3	S4	0	0
S5	S1	S2	0	1

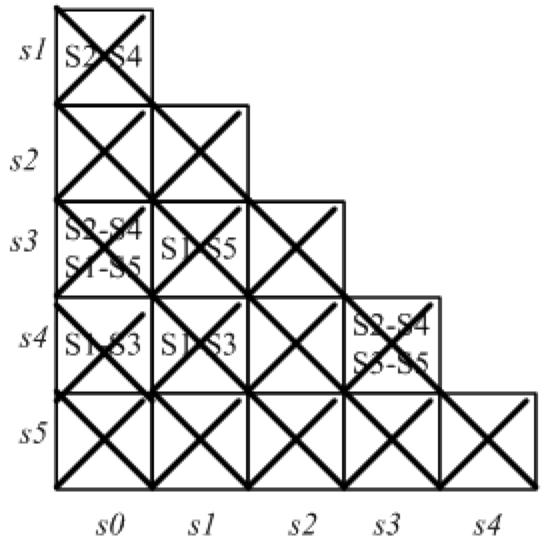


- Also indicate implied pairs

Step 2

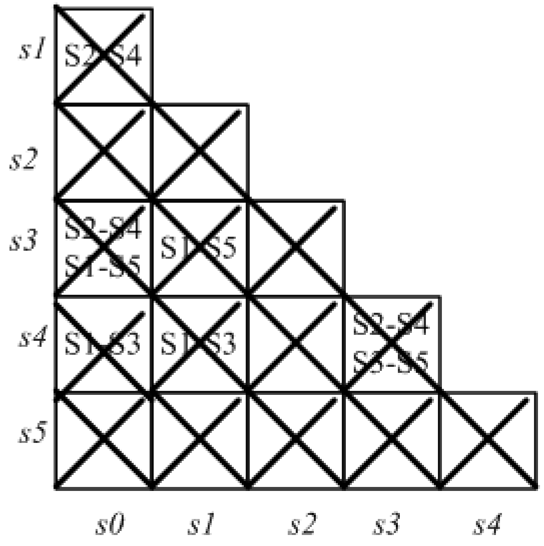
- Check implied pairs and X
- 1st pass and 2nd pass





What does it tell you?

- In this case, the state table is minimal as no state reduction can be done.



Lecture summary

- Have covered the method for removal of redundant states from state tables.
- **Work problem 14.26 by enumerating all the possible states and then doing state reduction. See web page.**
- Look at 15.2 through 15.8 (answers in text)