Advanced Digital Circuit Design - State Reduction and Assignment

Prof. Dr. Berna Örs Yalçın

Istanbul Technical University

Faculty of Electrical and Electronics Engineering

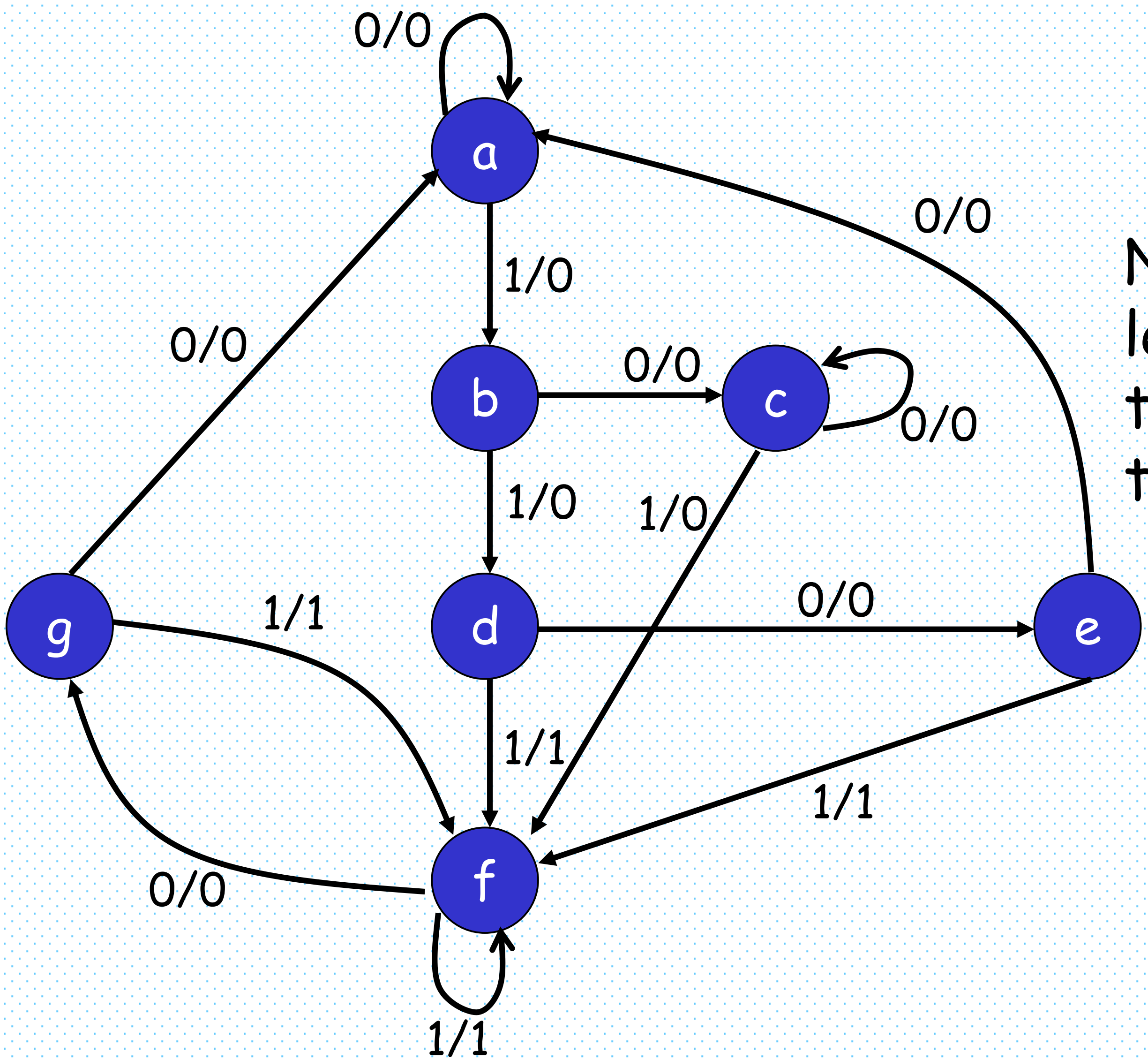
Department of Electronics and Communication Engineering

siddika.ors@itu.edu.tr

State Reduction and Assignment

- In the design process of sequential circuits certain techniques are useful in reducing the circuit complexity
  - state reduction
  - state assignment
- State reduction
  - Fewer states → fewer number of flip-flops
  - m flip-flops → 2^m states
  - Example: $m = 5$ → $2^m = 32$
    - If we reduce the number of states to 21 do we reduce the number of flip-flops?

Example: State Reduction



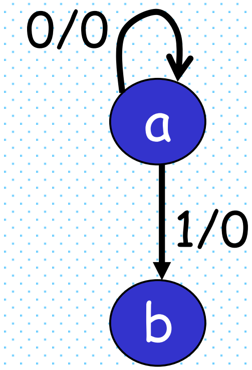
Note that we use letters to designate the states for the time being

Example: State Reduction

state	a	a	b	c	f	g	f	f	g	a	a		
input	0	1	0	1	0	1	1	0	0	0	0		
output	0	0	0	0	0	1	1	0	0	0			

- What is important
  - not the states
  - but the output values the circuit generates
- Therefore, the problem is to find a circuit
  - with fewer number of states,
  - but that produces the same output pattern for any given input pattern, starting with the same initial state

State Reduction Technique 1/7



- Step 1: get the state table

present state	next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	c	d	0	0
c	c	f	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

State Reduction Technique 2/7

- Step 2: Inspect the state table for equivalent states
  - Equivalent states: Two states,
    1. that produce exactly the same output
       - for each input combination
    2. whose next states are identical
       - for each input combination

State Reduction Technique 3/7

present state	next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	c	d	0	0
c	c	f	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

- States "e" and "g" are equivalent
- One of them can be removed

State Reduction Technique 4/7

present state	next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	c	d	0	0
c	c	f	0	0
d	e	f	0	1
e	a	f	0	1
f	e	f	0	1

- We keep looking for equivalent states

State Reduction Technique 5/7

present state	next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	c	d	0	0
c	c	d	0	0
d	e	d	0	1
e	a	d	0	1

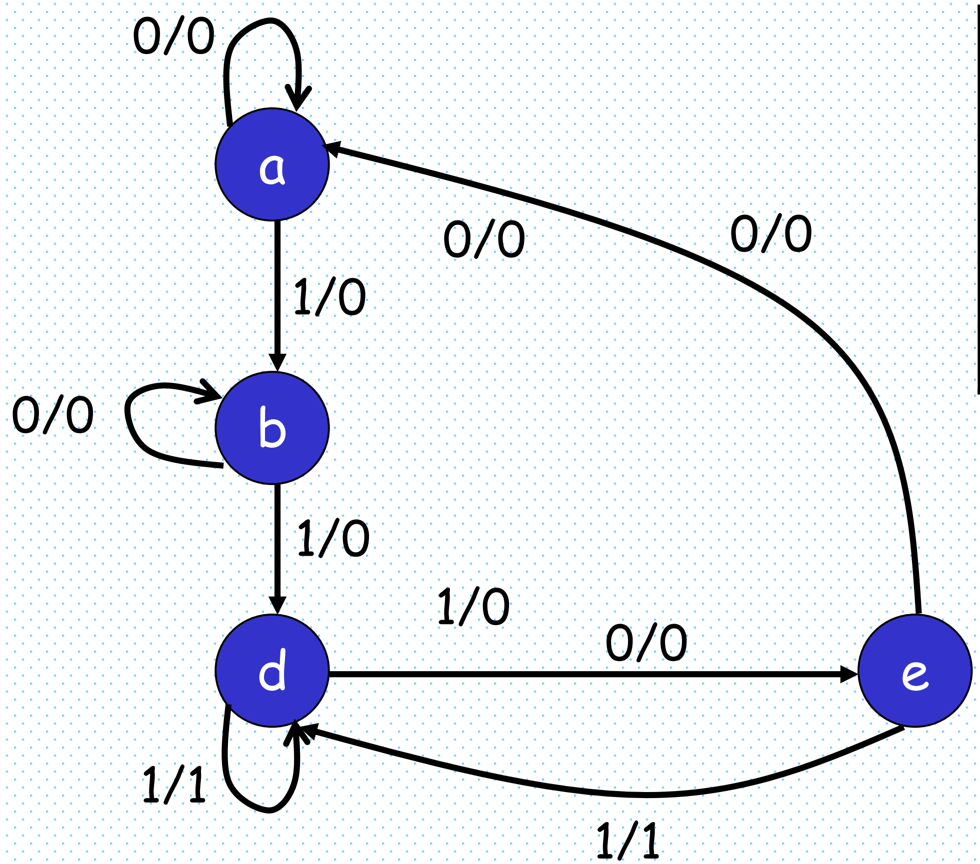
- We keep looking for equivalent states

State Reduction Technique 6/7

present state	next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	b	d	0	0
d	e	d	0	1
e	a	d	0	1

- We stop when there are no equivalent states

State Reduction Technique 7/7



present state	next state x = 0	next state x = 1	Output x = 0	Output x = 1
a	a	b	0	0
b	b	d	0	0
d	e	d	0	1
e	a	d	0	1

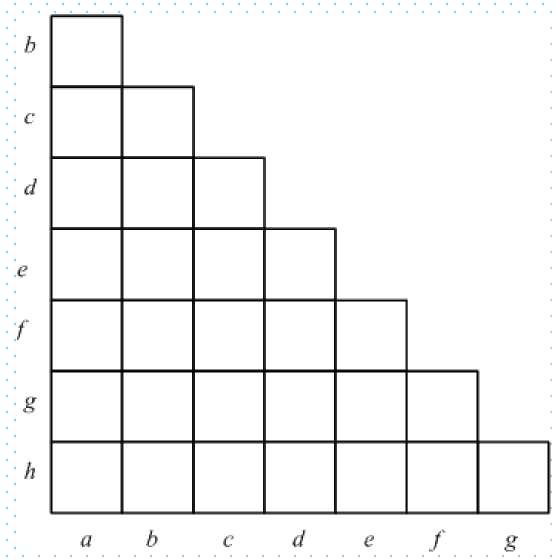
We need two flip-flops

state	a	a	b	b	d	e	d	d	e	a	a		
input	0	1	0	1	0	1	1	0	0	0	0		
output	0	0	0	0	0	1	1	0	0	0			

Implication Tables

- 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.

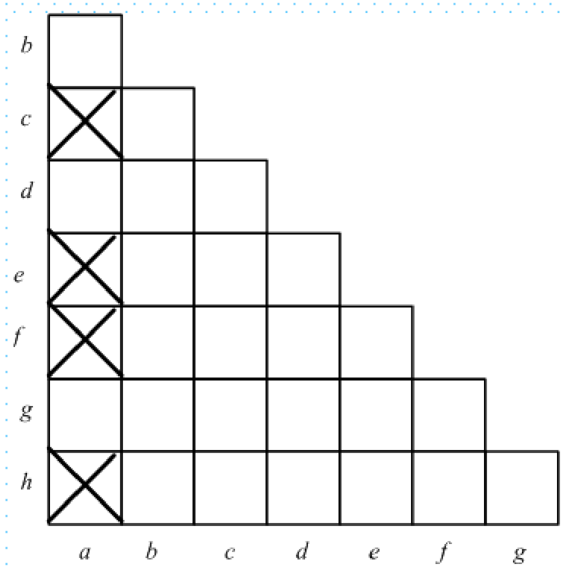
Present State	Next State $X = 0$	Next State $X = 1$	Present Output
a	*d*	*c*	0
b	*f*	*h*	0
c	*e*	*d*	1
d	*a*	*e*	0
e	*c*	*a*	1
f	*f*	*b*	1
g	*b*	*h*	0
h	*c*	*g*	1



Step 1

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

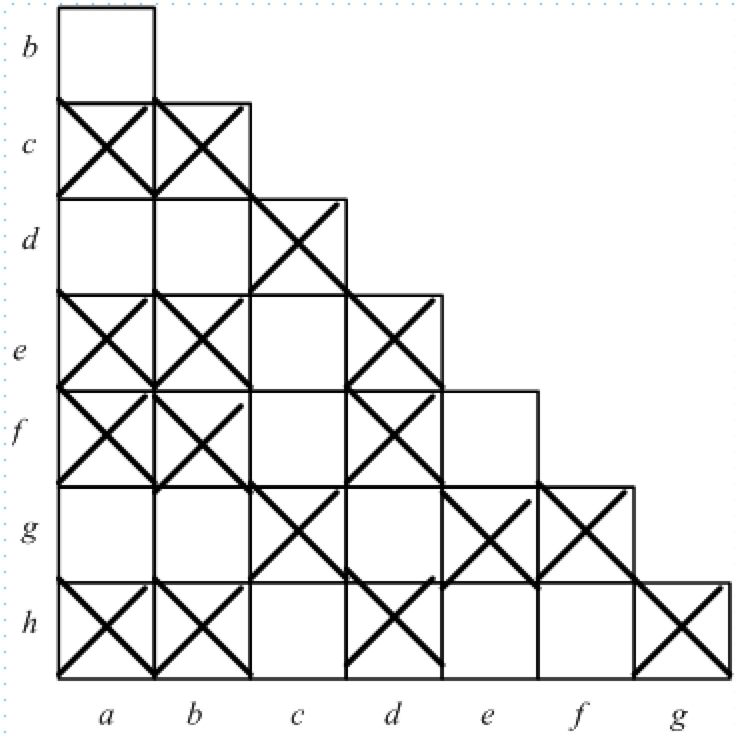
Present State	Next State $X = 0$	1	Present Output
a	d	c	0
b	f	h	0
c	e	d	1
d	a	e	0
e	c	a	1
f	f	b	1
g	b	h	0
h	c	g	1



Step 1 continued

- Continue to remove output incompatible states

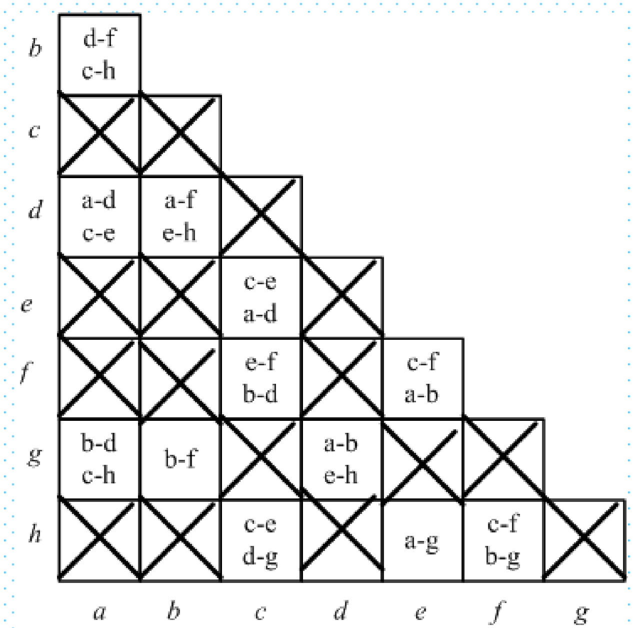
Present State	Next State X = 0	1	Present Output
a	d	c	0
b	f	h	0
c	e	d	1
d	a	e	0
e	c	a	1
f	f	b	1
g	b	h	0
h	c	g	1



Now what?

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

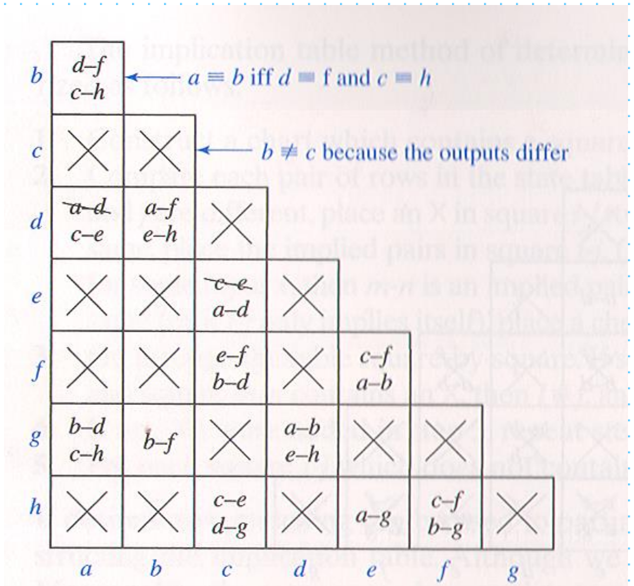
Present State	Next State X = 0	1	Present Output
a	*d*	*c*	0
b	*f*	*h*	0
c	*e*	*d*	1
d	*a*	*e*	0
e	*c*	*a*	1
f	*f*	*b*	1
g	*b*	*h*	0
h	*c*	*g*	1



Self redundant pairs

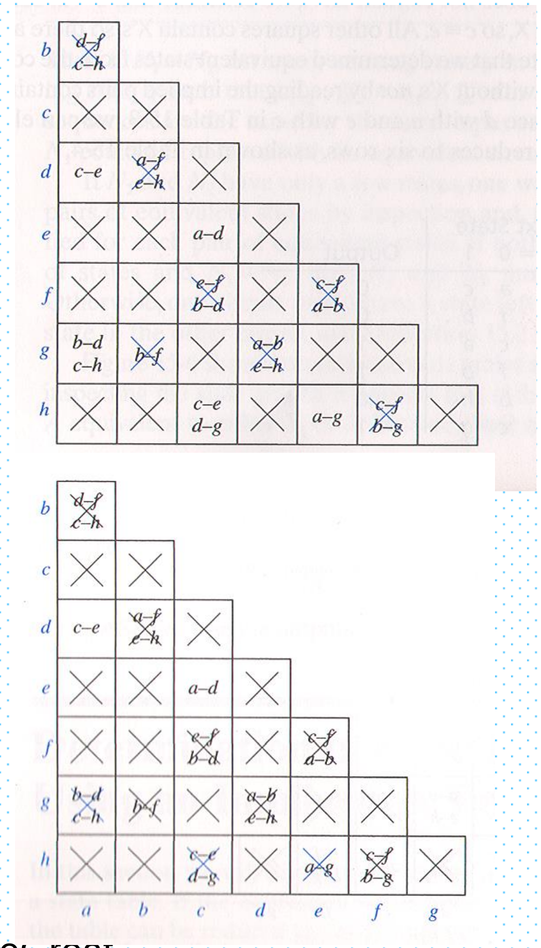
- Self redundant pairs are removed, i.e., in square a-d it contains a-d.

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



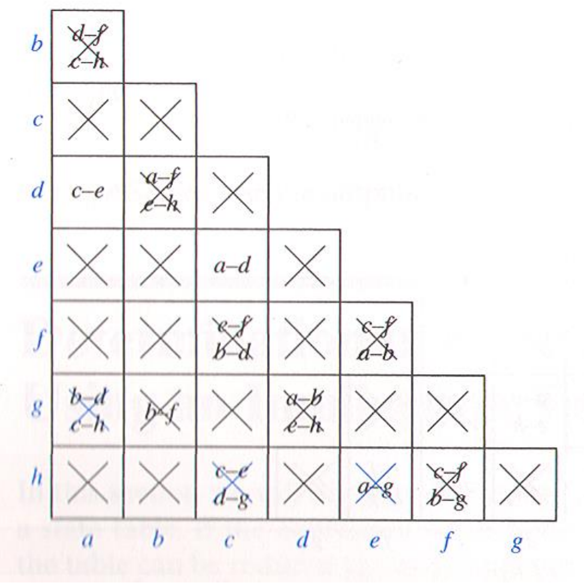
Next pass

- 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.

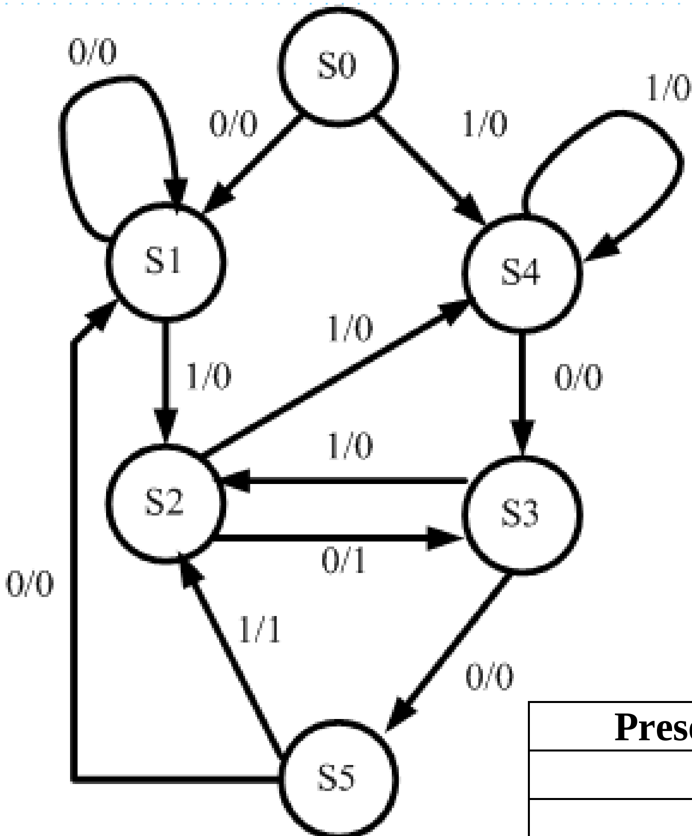
Present State	Next State X = 0	1	Present Output
a	d	c	0
b	f	h	0
c	e	d	1
d	a	e	0
e	c	a	1
f	f	b	1
g	b	h	0
h	c	g	1

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.
3. Remove redundant pairs. If the implied pair is the same place a check mark as $i \equiv j$.
4. Go through the implied pairs and X the square when an implied pair is incompatible.
5. Repeat until no more Xs are added.
6. For any remaining squares not Xed, $i \equiv j$.

Another example

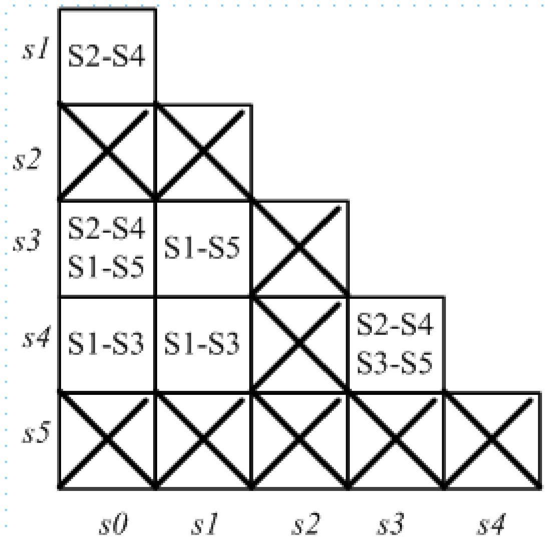


	NEXT STATE		OUTPUT	
Present State	**X=0**	**X=1**	**X=0**	**X=1**
S0	S1	S4	0	0
S1	S1	S2	0	0
S2	S3	S4	1	0
S3	S5	S2	0	0
S4	S3	S4	0	0
S5	S1	S2	0	1

Set up Implication Chart

- And remove output incompatible states

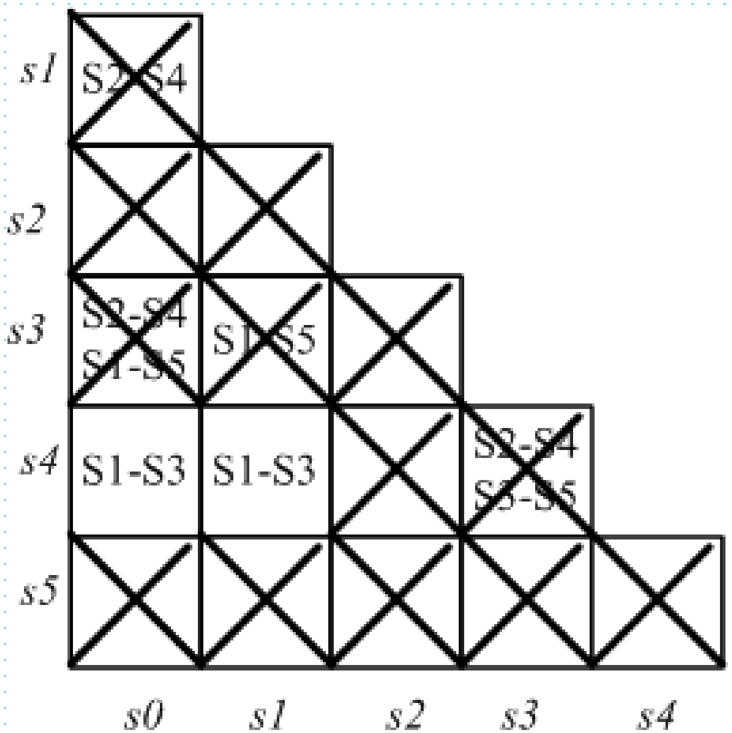
	NEXT STATE		OUTPUT	
Present State	**X=0**	**X=1**	**X=0**	**X=1**
S0	S1	S4	0	0
S1	S1	S2	0	0
S2	S3	S4	1	0
S3	S5	S2	0	0
S4	S3	S4	0	0
S5	S1	S2	0	1

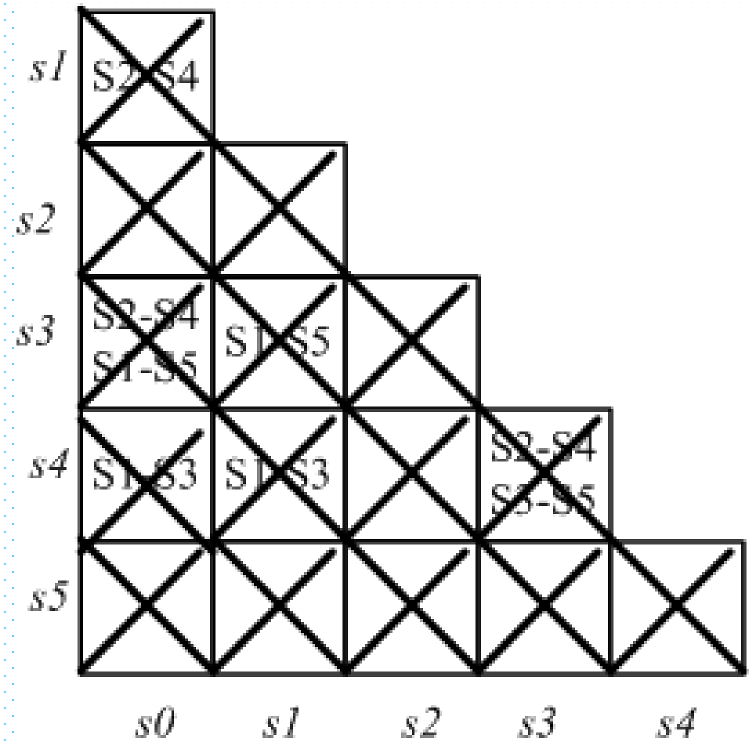


- Also indicate implied pairs

Step 2

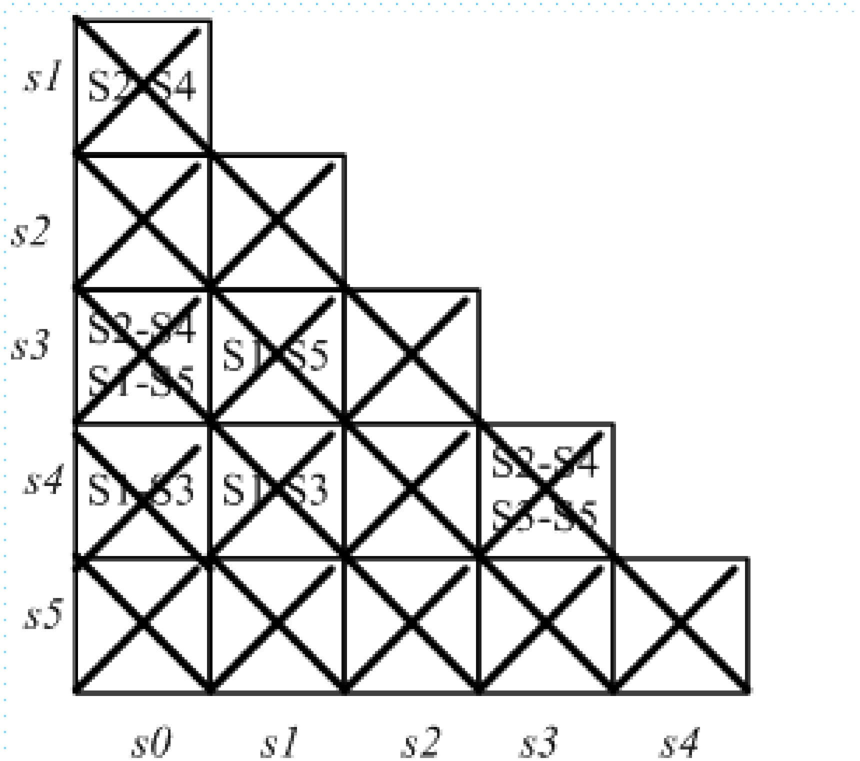
- Check implied pairs and X
- 1st pass and 2nd pass



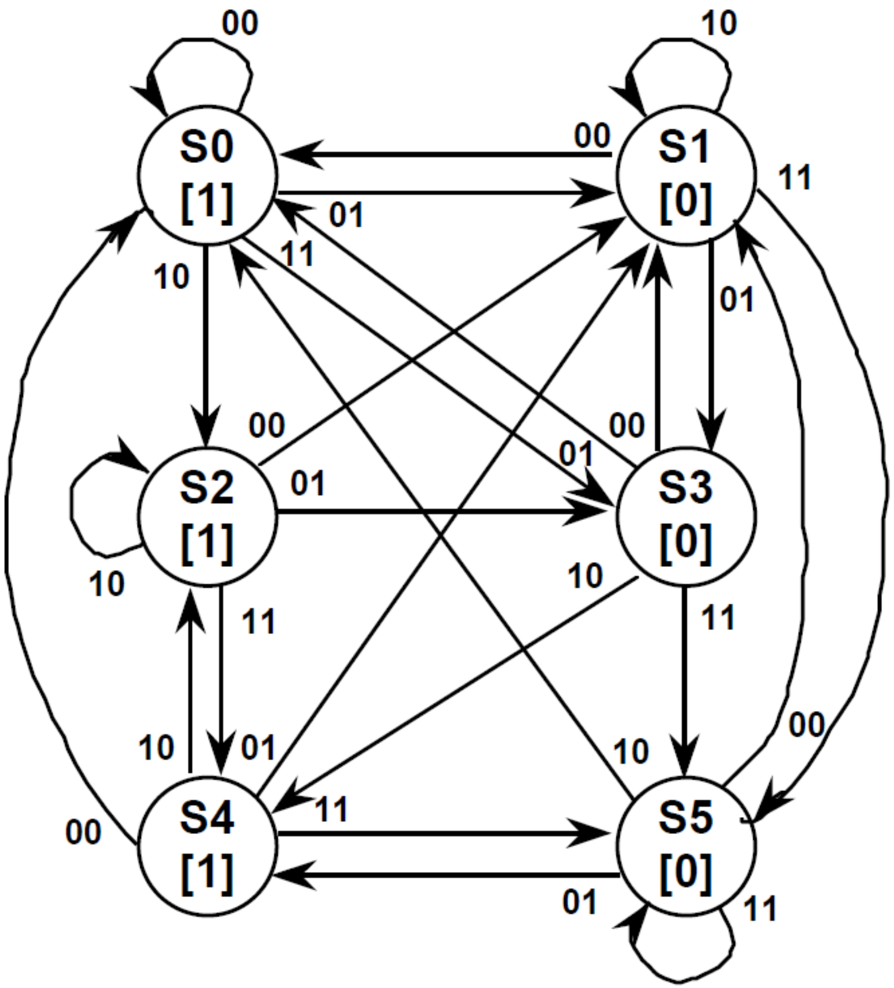


What does it tell you?

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



State Reduction: *Multiple Input State Diagram Example (1/)*

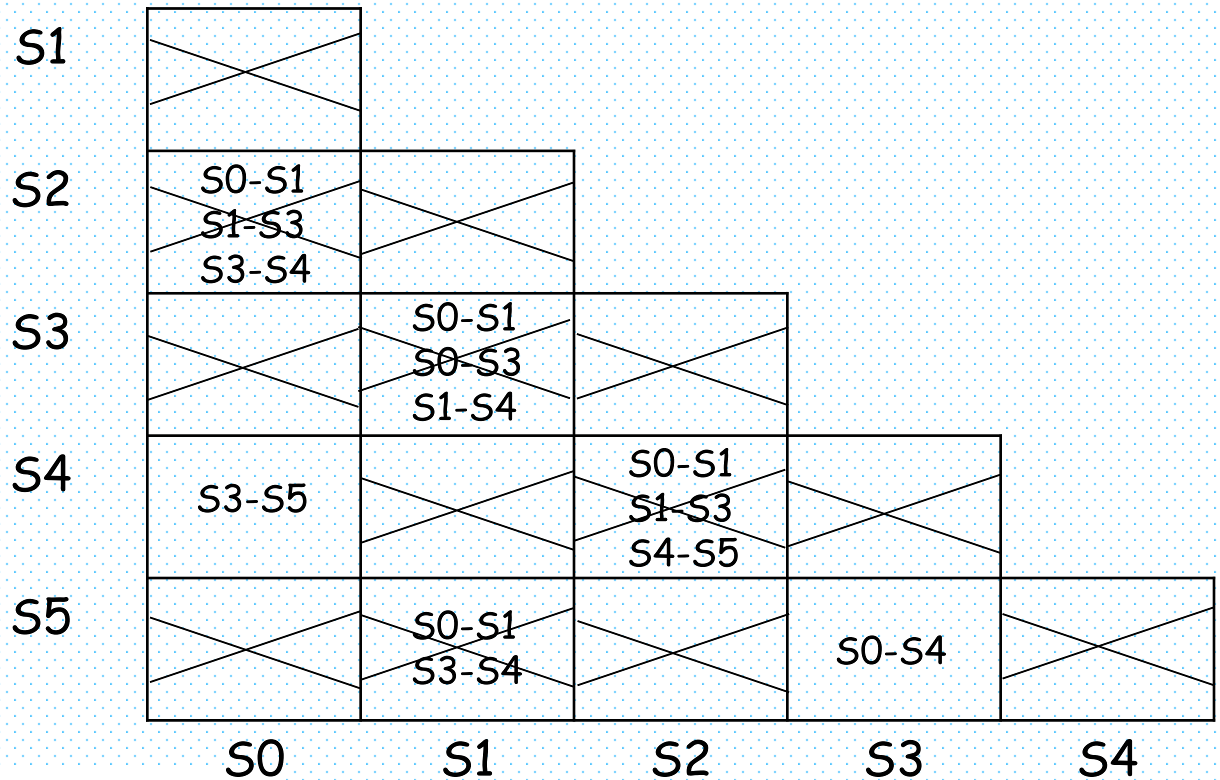


State Diagram

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

State Table

State Reduction: *Multiple Input State Diagram Example (2/)*



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

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

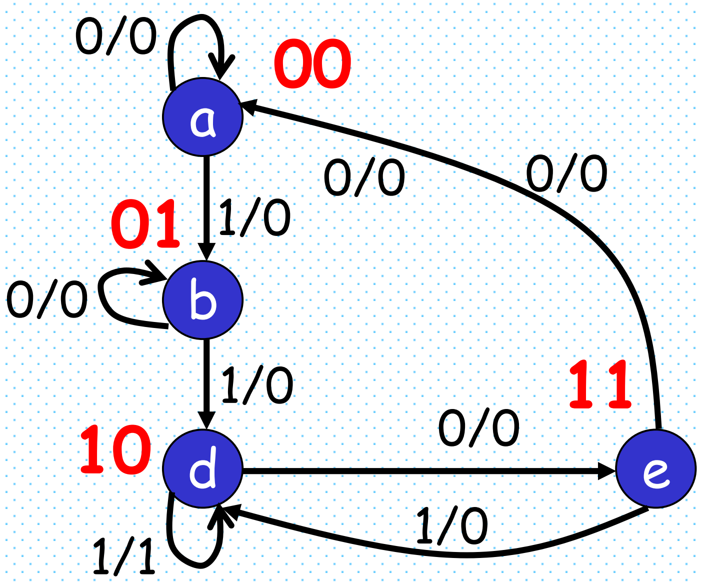
Minimised State Table

State Assignments

- We have to assign binary values to each state
- If we have m states, then we need codes of n bits, where $n = \lceil \log_2 m \rceil$
- There are different ways of encoding
- <u>Example</u>: Five states: S_0, S_1, S_2, S_3, S_4

state	binary	gray	one-hot
S_0	000	000	00001
S_1	001	001	00010
S_2	010	011	00100
S_3	011	010	01000
S_4	100	110	10000

Binary State Encoding



y_1	y_2	x	Y_1	Y_2	J_1	K_1	J_2	K_2	z
0	0	0	0	0	0	X	0	X	0
0	0	1	0	1	0	X	1	X	0
0	1	0	0	1	0	X	X	0	0
0	1	1	1	0	1	X	X	1	0
1	0	0	1	1	X	0	1	X	0
1	0	1	1	0	X	0	0	X	1
1	1	0	0	0	X	1	X	1	0
1	1	1	1	0	X	0	X	1	0

Simplified flip-flop input equations

x \ y_1y_2	00	01	11	10
0	0	0	0	1
1	0	1	1	1

$D_1 = y_1y_2' + y_2x$

x \ y_1y_2	00	01	11	10
0	0	1	0	1
1	1	0	0	1

$D_2 = y_1y_2' + y_1'y_2x' + y_1'y_2'x$

2 2-input AND, 2 3-input AND, 1 2-input OR, 1 3-input OR gates

x \ y_1y_2	00	01	11	10
0	0	0	X	X
1	0	1	X	X

$J_1 = x+y_2$

x \ y_1y_2	00	01	11	10
0	X	X	0	1
1	X	X	0	0

$K_1 = x'+y_2'$

x \ y_1y_2	00	01	11	10
0	0	X	X	1
1	1	X	X	0

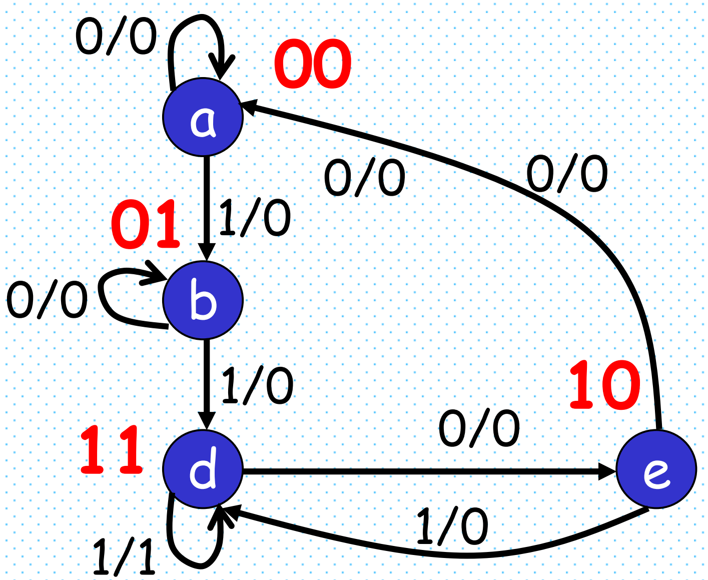
$J_2 = (y_1+x)(y_1'+x')$

x \ y_1y_2	00	01	11	10
0	X	0	1	X
1	X	1	1	X

$K_2 = y_1+x$

4 2-input OR, 1 2-input AND gates

Gray State Encoding



y_1	y_2	x	Y_1	Y_2	J_1	K_1	J_2	K_2	z
0	0	0	0	0	0	X	0	X	0
0	0	1	0	1	0	X	1	X	0
0	1	0	0	1	0	X	X	0	0
0	1	1	1	1	1	X	X	0	0
1	0	0	0	0	X	1	0	X	0
1	0	1	1	1	X	0	1	X	0
1	1	0	1	0	X	0	X	1	0
1	1	1	1	1	X	0	X	0	1

Simplified flip-flop input equations

$x \backslash y_1y_2$	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$D_1 = y_1y_2+y_2x+y_1x$

$x \backslash y_1y_2$	00	01	11	10
0		1		
1	1	1	1	1

$D_2 = y_1'y_2+x$

4 2-input AND, 1 3-input OR, 1 2-input OR gates

$x \backslash y_1y_2$	00	01	11	10
0	0	0	X	X
1	0	1	X	X

$J_1 = xy_2$

$x \backslash y_1y_2$	00	01	11	10
0	X	X	0	1
1	X	X	0	0

$K_1 = x'y_2'$

$x \backslash y_1y_2$	00	01	11	10
0	0	X	X	0
1	1	X	X	1

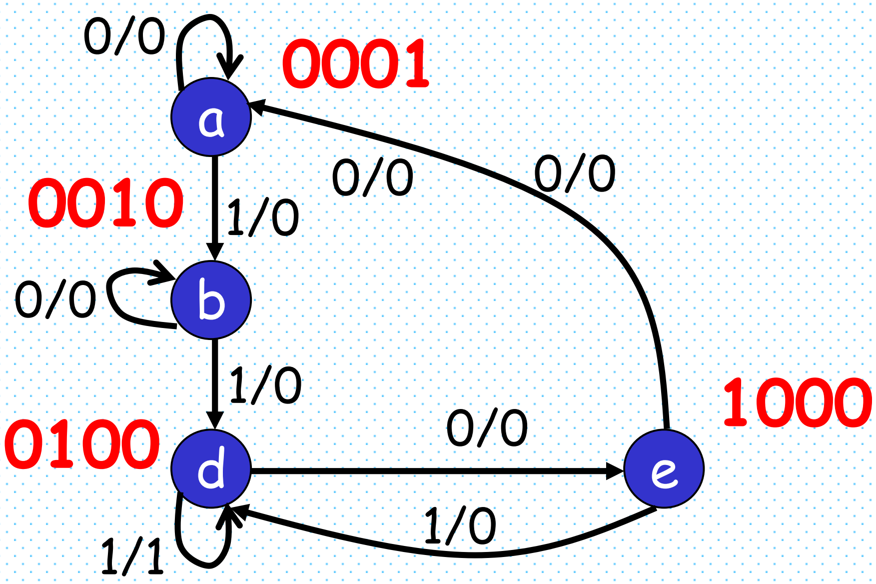
$J_2 = x$

$x \backslash y_1y_2$	00	01	11	10
0	X	0	1	X
1	X	0	0	X

$K_2 = y_1x'$

3 2-input AND

One-Hot State Encoding



y_1	y_2	y_3	y_4	x	Y_1	Y_2	Y_3	Y_4	J_1	K_1	J_2	K_2	J_3	K_3	J_4	K_4	z
0	0	0	1	0	0	0	0	1	0	X	0	X	0	X	X	0	0
0	0	0	1	1	0	0	1	0	0	X	0	X	1	X	X	1	0
0	0	1	0	0	0	0	1	0	0	X	0	X	X	0	0	X	0
0	0	1	0	1	0	1	0	0	0	X	1	X	0	1	0	X	0
0	1	0	0	0	1	0	0	0	1	X	X	1	0	X	0	X	0
0	1	0	0	1	0	1	0	0	0	X	X	0	0	X	0	X	1
1	0	0	0	0	0	0	0	1	X	1	0	X	0	X	1	X	0
1	0	0	0	1	0	1	0	0	X	1	1	X	0	X	0	X	0

Simplified flip-flop input equations

$D_1 = y_2x'$ $D_2 = y_3x$ $D_3 = y_4x+y_3x'$ $D_4 = y_4x'+y_1x'$

6 2-input AND, 2 2-input OR gates

$J_1 = y_2x'$ $K_1 = 1$

$J_2 = y_3x+y_1x$ $K_2 = y_2x'$

$J_3 = y_4x$ $K_3 = y_3x$

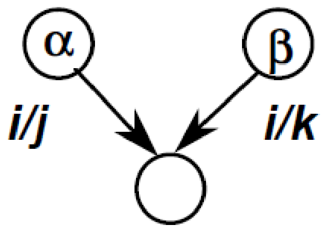
$J_4 = y_1x'$ $K_4 = y_4x$

5 2-input AND, 1 2-input OR gates

State Assignment

Paper & Pencil Methods

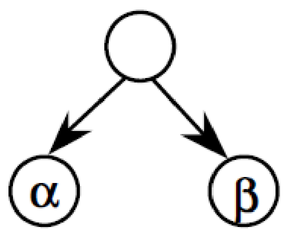
Alternative heuristics based on input and output behavior as well as transitions:



Highest Priority

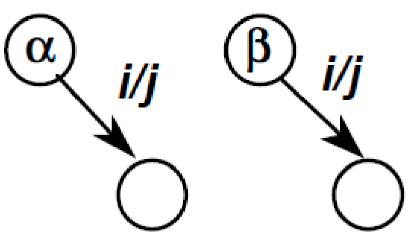
Adjacent assignments to:

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)



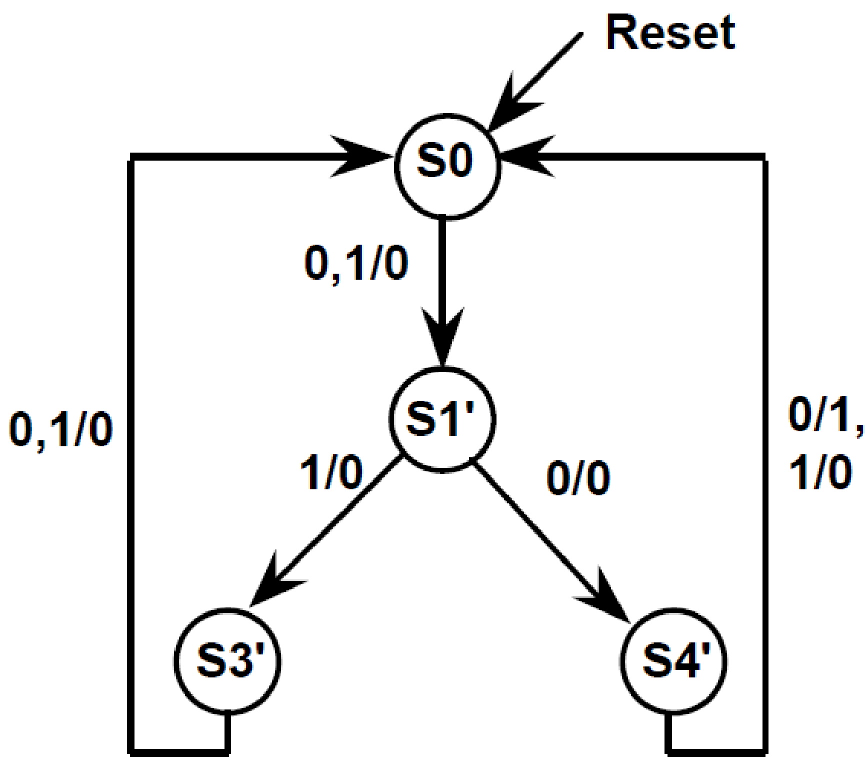
Lowest Priority

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')

Example 1: State Assignment

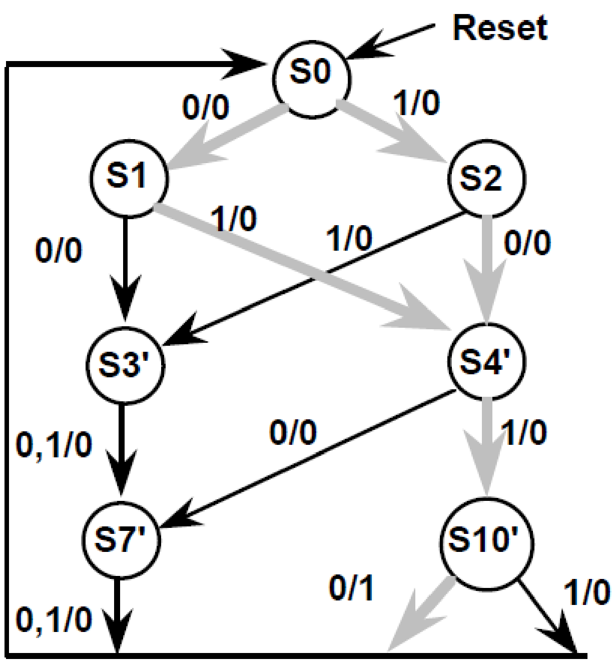
- Reset State: S0
- Highest Priority: S3, S4
- Medium Priority: S3, S4
- Lowest Priority:
  - S0, S1
  - S1, S3
  - S1, S4
  - S3, S0
  - S4, S0

S0: 00	S0: 00
S3: 01	S3: 11
S4: 11	S4: 10
S1: 10	S1: 01

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')

Example 2: State Assignment

- Reset State: S0
- Highest Priority:
  - S1, S2
  - S3, S4
  - S7, S10
- Medium Priority:
  - S1, S2
  - S3, S4
  - S7, S10
- Lowest Priority:
  - S0, S1
  - S0, S2
  - S1, S3
  - S1, S4
  - S2, S3
  - S2, S4
  - S7, S0
  - S10,S0

S0: 000

S1: 001

S2: 011

S3: 010

S4: 110

S7: 100

S10: 101

Example 2: State Assignment

y_1	y_2	y_3	x	Y_1	Y_2	Y_3	J_1	K_1	J_2	K_2	J_3	K_3	z
0	0	0	0	0	0	1	0	X	0	X	1	X	0
0	0	0	1	0	1	1	0	X	1	X	1	X	0
0	0	1	0	0	1	0	0	X	1	X	X	1	0
0	0	1	1	1	1	0	1	X	1	X	X	1	0
0	1	0	0	1	0	0	1	X	X	1	0	X	0
0	1	0	1	1	0	0	1	X	X	1	0	X	0
0	1	1	0	1	1	0	1	X	X	0	X	1	0
0	1	1	1	0	1	0	0	X	X	0	X	1	0
1	0	0	0	0	0	0	X	1	0	X	0	X	0
1	0	0	1	0	0	0	X	1	0	X	0	X	0
1	0	1	0	0	0	0	X	1	0	X	X	1	1
1	0	1	1	0	0	0	X	1	0	X	X	1	0
1	1	0	0	1	0	0	X	0	X	1	0	X	0
1	1	0	1	1	0	1	X	0	X	1	1	X	0
1	1	1	0	X	X	X	X	X	X	X	X	X	X
1	1	1	1	X	X	X	X	X	X	X	X	X	X

Simplified flip-flop input equations

y_3x \ y_1y_2	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	1	0	X	0
10	0	1	X	0

$D_1 = y_2y_3' + y_1'y_2'y_3x + y_2y_3x'$

y_3x \ y_1y_2	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	1	1	X	0
10	1	1	X	0

$D_2 = y_1'y_2'x + y_1'y_3$

y_3x \ y_1y_2	00	01	11	10
00	1	0	0	0
01	1	0	1	0
11	0	0	X	0
10	0	0	X	0

$D_2 = y_1'y_2'y_3' + y_1y_2x$

Simplified flip-flop input equations

y_3x \ y_1y_2	00	01	11	10
00	0	1	X	X
01	0	1	X	X
11	1	0	X	X
10	0	1	X	X

$J_1 = y_2y_3' + y_2'y_3x + y_2x'$

y_3x \ y_1y_2	00	01	11	10
00	X	X	0	1
01	X	X	0	1
11	X	X	X	1
10	X	X	X	1

$K_1 = y_2'$

y_3x \ y_1y_2	00	01	11	10
00	0	X	X	0
01	1	X	X	0
11	1	X	X	0
10	1	X	X	0

$J_2 = y_1'(y_3+x)$

y_3x \ y_1y_2	00	01	11	10
00	X	1	1	X
01	X	1	1	X
11	X	0	X	X
10	X	0	X	X

$K_2 = y_3'$

y_3x \ y_1y_2	00	01	11	10
00	1	0	0	0
01	1	0	1	0
11	X	X	X	X
10	X	X	X	X

$J_3 = y_1'y_2' + y_1y_2x$

y_3x \ y_1y_2	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	1	1	X	1
10	1	1	X	1

$K_3 = 1$