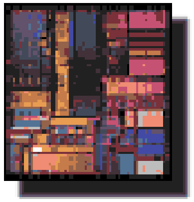
ECE 595Z
Digital VLSI Design Automation

Module 7 (Lectures 24-26): Sequential Logic Optimization

Lecture 25



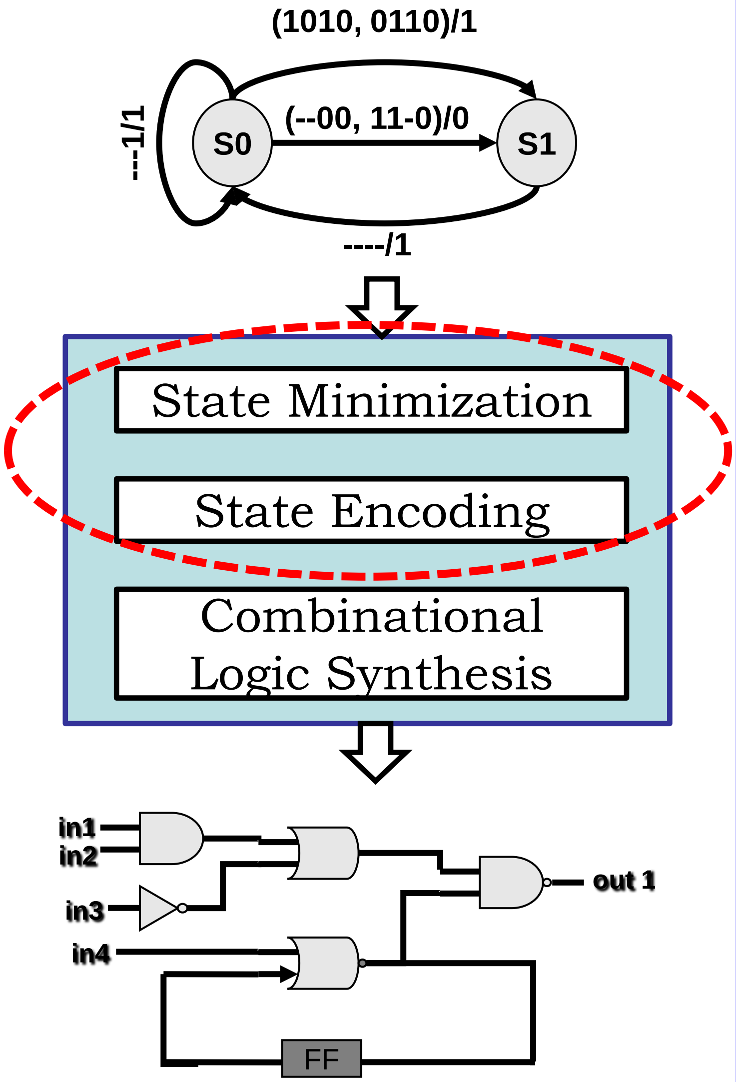
Anand Raghunathan

MSEE 318

raghunathan@purdue.edu

FSM Synthesis - Overview

- Given FSM specification, synthesize optimized implementation (gates + FFs)
  - State minimization
  - State encoding
  - Derive next-state, output functions & apply combinational logic minimization techniques

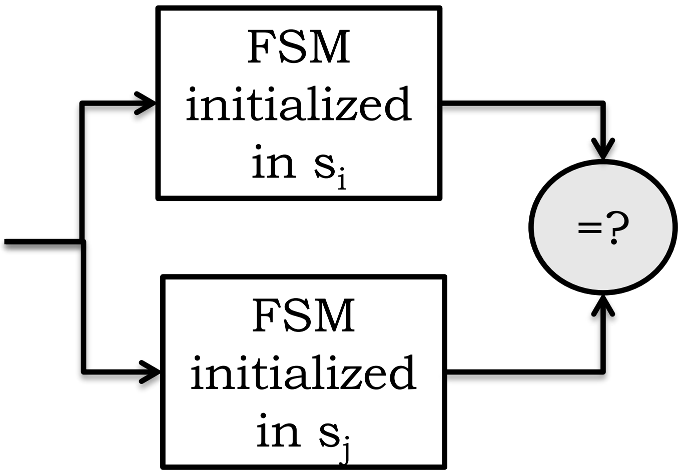


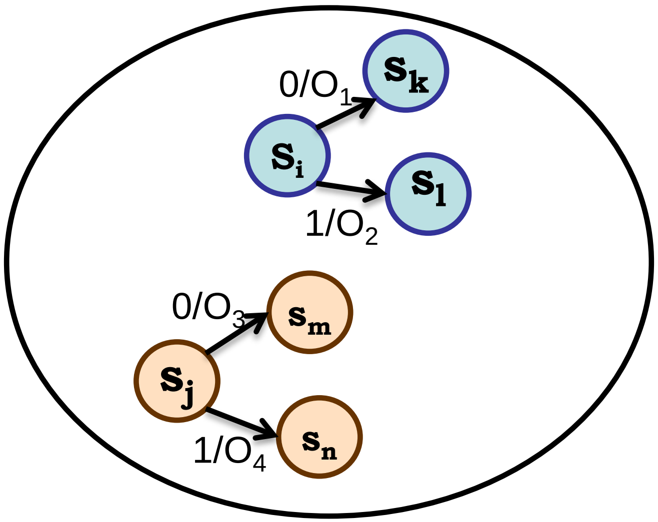
State Minimization

- Multiple states in an FSM may be equivalent
  - Equivalent states may be merged into a single state without affecting functionality
  - Often reduces complexity of implementation, but not always
- Definition of equivalence is different for completely and incompletely specified FSMs

Equivalent States in Completely Specified FSMs

- **Definition**: Two states are equivalent iff the output sequences of the FSM initialized in the two states are equal for any input sequence
- **Theorem**: Two states of a completely specified FSM are equivalent iff, for every input, the outputs are identical and the corresponding next states are equivalent

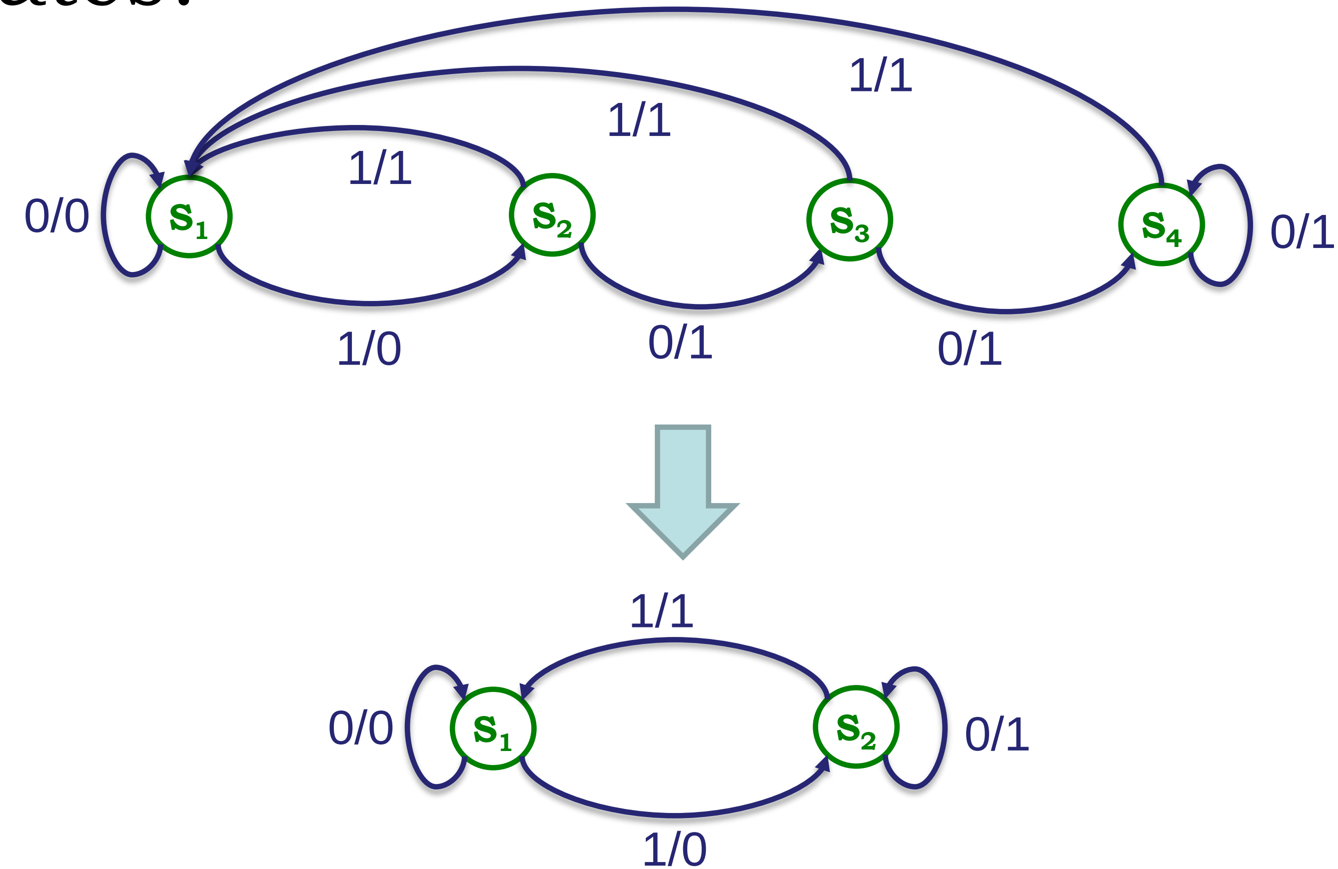




$s_i = s_j$ if ______________________

Example

- Does this FSM contain equivalent states?



Distinguishable States

- **Definition**: Two states, s_i and s_j of FSM M are ***distinguishable*** if and only if there exists a **finite input sequence** which when applied to M **causes different output sequences** depending on whether M started in s_i or s_j.
  - Such a sequence is called a ***distinguishing sequence*** for (s_i, s_j)
  - If there exists a distinguishing sequence of length k for (s_i, s_j), they are said to be ***k-distinguishable***.

Example:

PS	NS, z x=0	x=1
A	E, 0	D, 1
B	F, 0	D, 0
C	E, 0	B, 1
D	F, 0	B, 0
E	C, 0	F, 1
F	B, 0	C, 0

A and B are ____________________

A and E are ____________________

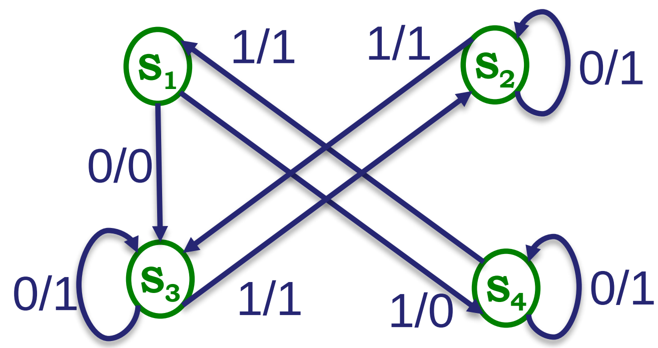
Identifying Equivalent States

- Partition set of states so that two states are in the same group/partition iff they are equivalent
- Compute this iteratively
  1. Start with all states in one group
  2. Split states for which applying the same input leads to different outputs
  3. Split states whose next states when applying the same input are in different groups
  4. Repeat step 3 until no further change

The above procedure is guaranteed to terminate in n_s steps, where n_s is the number of states in the FSM. The Complexity is $O(n_s^2)$.
An $O(n_s \log(n_s))$ algorithm exists.

Identifying Equivalent States: Example

1. Start with all states in one group
2. Split states for which applying the same input leads to different outputs
3. Split states whose next states when applying the same input are in different groups
4. Repeat step 3 until no further change



	output	
PS	0	1
s_1	0	1
s_2	1	1
s_3	1	1
s_4	1	0

$\{s_1,s_2,s_3,s_4\}$

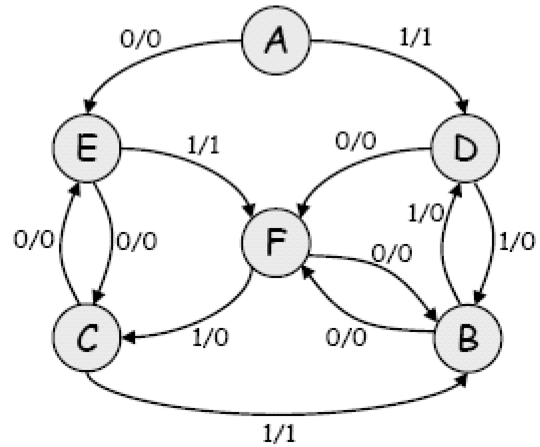
$\Downarrow$

$\{s_1\},\{s_2,s_3\},\{s_4\}$

$\Downarrow$

$\{s_1\},\{s_2,s_3\},\{s_4\}$ No change!

Identifying Equivalent States: Example



PS	NS, PO x=0	NS, PO x=1
A	E, 0	D, 1
B	F, 0	D, 0
C	E, 0	B, 1
D	F, 0	B, 0
E	C, 0	F, 1
F	B, 0	C, 0

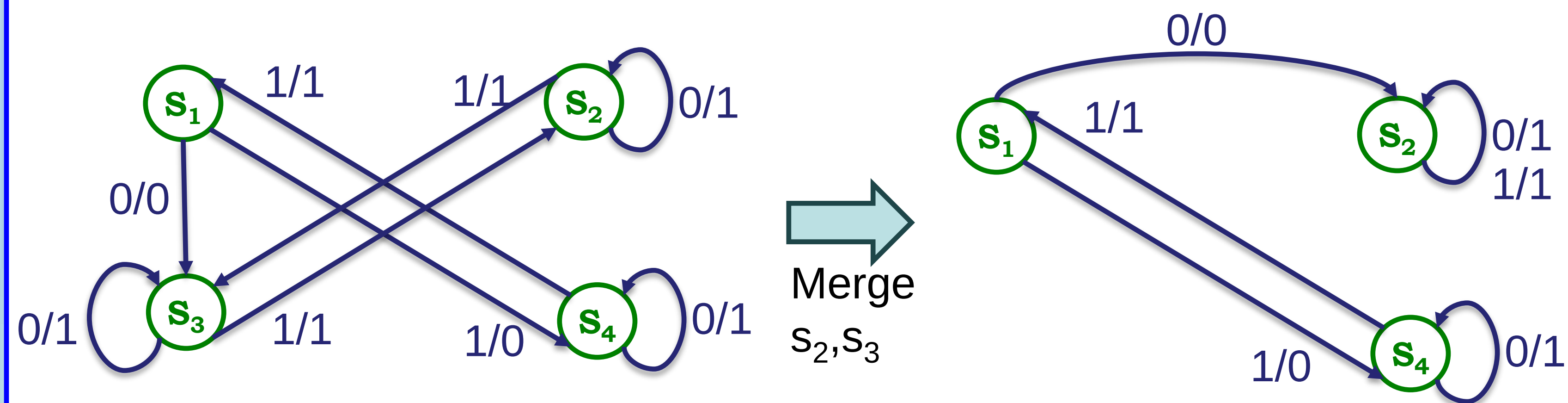
1. Start with all states in one group
2. Split states for which applying the same input leads to different outputs
3. Split states whose next states when applying the same input are in different groups
4. Repeat step 3 until no further change

{A,B,C,D,E,F}
⇩
{A,C,E}, {B,D,F}
⇩
{A,C,E}, {B,D}, {F}
⇩
{A,C}, {E}, {B,D}, {F}
⇩
{A,C}, {E}, {B,D}, {F}

No change!

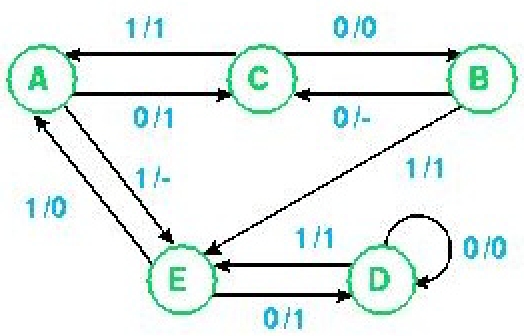
State Minimization

- Given equivalent groups of states
  - Merge all states in a group of equivalent states into a single state
  - All transitions going in/out of all states in the group go in/out of the merged state



How about Incompletely Specified FSMs?

- **Definition**: Two states are **compatible** iff they agree on the outputs when they are all specified & corresponding next states are compatible when both are specified

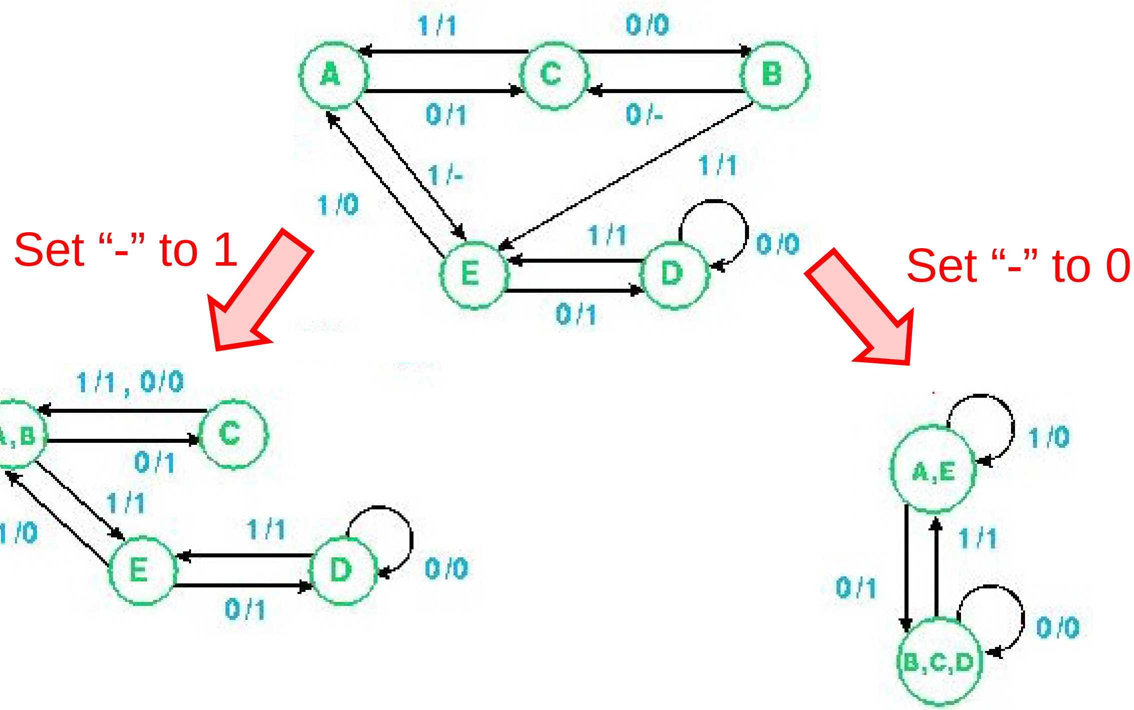


A ~ B if
    1/- from A is made 1/1
    0/- from B is made 0/1
B ~ C if
    0/- from B is made 0/0 AND A ~ E

- Important difference between equivalence and compatibility
  - Equivalence is transitive (A=B and B=C → A=C)
  - Compatibility is NOT! (A~B and B~C ↛ A~C)

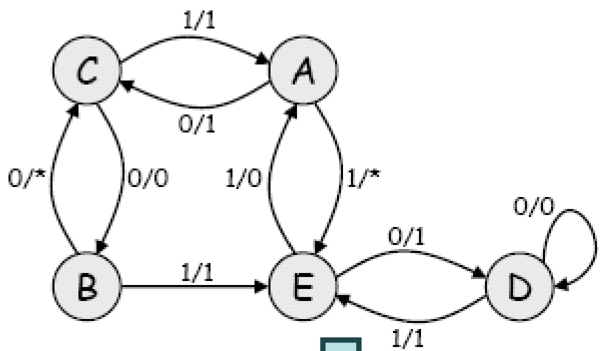
Minimizing Incompletely Specified FSMs

- How about specifying don't cares to 0/1 and using technique for minimizing completely specified FSMs?
  - Huge number of possible don't care assignments
  - Setting the don't care values differently can lead to drastically different results!



Minimizing Incompletely Specified FSMs

- Overview



PS	NS, PO x=0	NS, PO x=1
A	C, 1	E, *
B	C, *	E, 1
C	B, 0	A, 1
D	D, 0	E, 1
E	D, 1	A, 0

	Pairs	Implied Pairs
Compatible	{A, B}	
Compatible	{A, E}	{C, D}
Compatible	{B, C}	{A, E}
Compatible	{B, D}	{C, D}
Compatible	{C, D}	{B, D}, {A, E}
Incompatible	{A, C}	
Incompatible	{A, D}	
Incompatible	{B, E}	
Incompatible	{C, E}	
Incompatible	{D, E}	

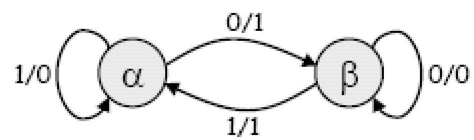
Find compatible pairs

Find larger compatible sets

Compatibles	Implied Classes	
{A, B}		
{A, E}	{C, D}	α
{B, C, D}	{A, E}	β

Closed set (all implied classes are contained within it)

Complete set (all states are included)



Choose minimum subset

Minimizing Incompletely Specified FSMs

- Algorithm to minimize incompletely specified FSMs
  1. Find the pairs of compatible states
  2. Find the maximal compatibles
  3. Find the remaining prime compatibles
  4. Select the minimum number of prime compatibles such that they form a closed and complete cover
     - Binate covering problem!
  5. Construct the reduced FSM

Finding Maximal Compatibles

- Maximal compatibles: sets of compatible states that are not strictly contained in any other set of compatible states
- Step 2: Find the maximal compatibles
  a) Associate a variable to a state s_i. 1 means that s_i belongs to the maximal compatibles
  b) Take each incompatible pair s_i,s_j and make a clause with s_i' and s_j'
  c) Take a product of all the clauses (POS form)
  d) Multiply it out to get SOP expression
  e) Identify a maximal compatible from each product term in the SOP expression

Example

Inputs

States	x_1	x_2	x_3	x_4	x_5	x_6	x_7
a	a,0	-	d,0	e,1	b,0	a,-	-
b	b,0	d,1	a,-	-	a,-	a,1	-
c	b,0	d,1	a,1	-	-	-	g,0
d	-	e,-	-	b,-	b,0	-	a,-
e	b,-	e,-	a,-	-	b,-	e,-	a,1
f	b,0	c,-	-,1	h,1	f,1	g,0	-
g	-	c,1	-	e,1	-	g,-	f,0
h	a,1	e,0	d,1	b,0	b,-	e,-	a,1

State transition table

b	a,d						
c	X	~					
d	b,e	a,b d,e	d,e a,g				
e	a,b a,d	d,e a,b a,e	X	~			
f	X	X	c,d	X	X		
g	~	X	c,d f,g	X	X	e,h	
h	X	X	X	~	a,b a,d	X	X
	a	b	c	d	e	f	g

Compatible pairs

POS expression representing compatibles

(a'+c')(a'+f')(a'+h')(b'+f')(b'+g')(b'+h')(c'+e')
(c'+h')(d'+f')(d'+g')(e'+f')(e'+g')(f'+h')(g'+h')

Finding Maximal Compatibles

Example

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
a	a,0	-	d,0	e,1	b,0	a,-	-
b	b,0	d,1	a,-	-	a,-	a,1	-
c	b,0	d,1	a,1	-	-	-	g,0
d	-	e,-	-	b,-	b,0	-	a,-
e	b,-	e,-	a,-	-	b,-	e,-	a,1
f	b,0	c,-	-,1	h,1	f,1	g,0	-
g	-	c,1	-	e,1	-	g,-	f,0
h	a,1	e,0	d,1	b,0	b,-	e,-	a,1

State transition table

b	a,d						
c	X	~					
d	b,e	a,b d,e	d,e a,g				
e	a,b a,d	d,e a,b a,e	X	~			
f	X	X	c,d	X	X		
g	~	X	c,d f,g	X	X	e,h	
h	X	X	X	~	a,b a,d	X	X
	a	b	c	d	e	f	g

Compatible pairs

POS expression representing compatibles

(a'+c')(a'+f')(a'+h')(b'+f')(b'+g')(b'+h')(c'+e')
(c'+h')(d'+f')(d'+g')(e'+f')(e'+g')(f'+h')(g'+h')

= (f' + a'b'd'e'h')(h' + a'b'c'g')(g' + b'd'e')(c' + a'e')

= c'f'g'h' + a'e'f'g'h' + b'c'd'e'f'h' + a'b'c'f'g' + a'b'd'e'h

Maximal Compatibles: abde, bcd, ag, deh, cfg

Finding Additional Prime Compatibles

- Step 3: Find the remaining prime compatibles
- Let C1 be a compatible set and let Γ1 be the corresponding set of implied classes. C1 is prime iff there does not exist C2 ⊃ C1 such that Γ2 ⊆ Γ1

Compatible pairs

b	a,d						
c	X	~					
d	b,e	a,b d,e	d,e a,g				
e	a,b a,d	d,e a,b a,e	X	~			
f	X	X	c,d	X	X		
g	~	X	c,d f,g	X	X	e,h	
h	X	X	X	~	a,b a,d	X	X
	a	b	c	d	e	f	g

Maximal compatibles	Implied classes
{a,b,d,e}	{ }
{b,c,d}	{{a,b}, {a,g}, {d,e}}
{c,f,g}	{{c,d}, {e,h}}
{d,e,h}	{{a,b}, {a,d}}
{a,g}	{ }

Other Prime compatibles	
{b,c}	{ }
{c,d}	{{a,g}, {d,e}}
{c,f}	{{c,d}}
{c,g}	{{c,d}, {f,g}}
{f,g}	{{e,h}}
{d,h}	{ }
{f}	{ }

Finding a Minimum Cover

- Step 4: Select a minimum set of prime compatibles that
  - Forms a closed cover
  - Is a complete cover

Maximal compatibles	Implied classes
{a,b,d,e}	{ }
{b,c,d}	{{a,b}, {a,g}, {d,e}}
{c,f,g}	{{c,d}, {e,h}}
{d,e,h}	{{a,b}, {a,d}}
{a,g}	{ }
Other Prime compatibles	
{b,c}	{ }
{c,d}	{{a,g}, {d,e}}
{c,f}	{{c,d}}
{c,g}	{{c,d}, {f,g}}
{f,g}	{{e,h}}
{d,h}	{ }
{f}	{ }

Minimum cover $\Rightarrow$

A = {a, b, d, e}
B = {d, e, h}
C = {b, c}
D = {f, g}

Constructing the Reduced FSM

- Same as completely specified case, except specify don't cares as necessary

Example:

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
a	a,0	-	d,0	e,1	b,0	a,-	-
b	b,0	d,1	a,-	-	a,-	a,1	-
c	b,0	d,1	a,1	-	-	-	g,0
d	-	e,-	-	b,-	b,0	-	a,-
e	b,-	e,-	a,-	-	b,-	e,-	a,1
f	b,0	c,-	-,1	h,1	f,1	g,0	-
g	-	c,1	-	e,1	-	g,-	f,0
h	a,1	e,0	d,1	b,0	b,-	e,-	a,1

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
A							
B							
C							
D							

A = {a, b, d, e}
B = {d, e, h}
C = {b, c}
D = {f, g}