L10 – additional State Machine examples

States Machine Design

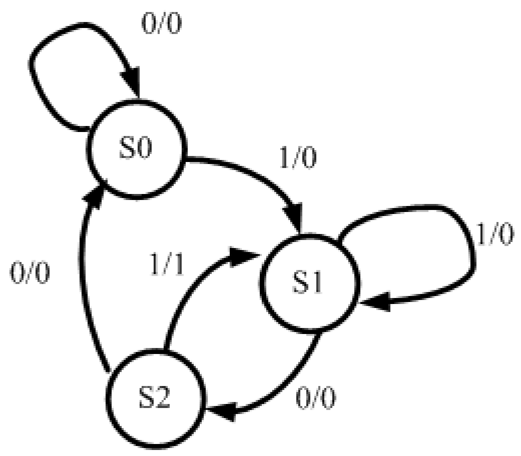
- Other topics on state machine design
  - Examples that are Equivalent
  - More one hot examples
- Ref: text Unit 15.4, 15.5, 15.8

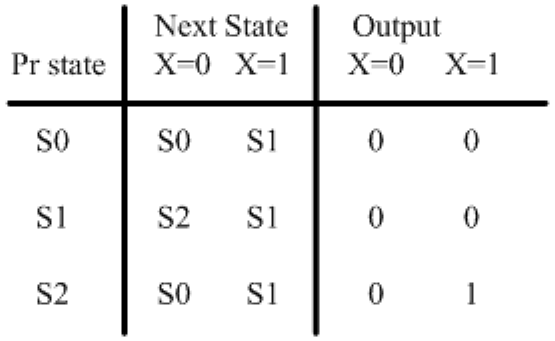
Equivalent State Machines

- So far have seen that equivalent states in the state table of a sequential machine are equivalent and can be removed.
- How about the equivalence between two sequential circuits?
  - *Two sequential circuits are equivalent if they are capable of doing the same work.*

Equivalence of two machines

- Consider a sequential machine
  - (ref lect 6) – a machine to detect 101 results in a state graph of with state table

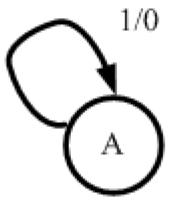




	Next State		Output	
Pr state	X=0	X=1	X=0	X=1
S0	S0	S1	0	0
S1	S2	S1	0	0
S2	S0	S1	0	1

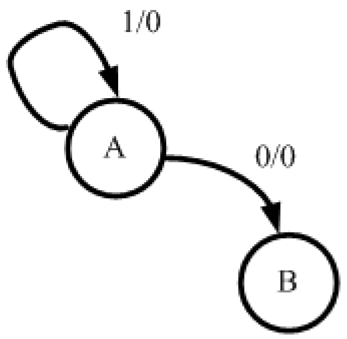
Now consider a second machine

- Consider a machine to detect 010, the complement of the other machine.
- Not yet develped - A is starting state where any number of 1’s has been received.

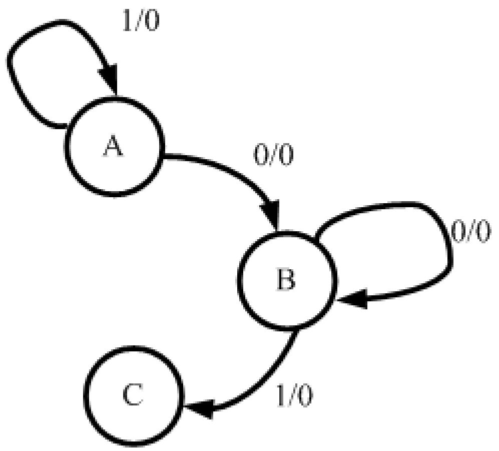


State A and state B

- State A is starting state.
  - On a 1 stay there
  - On a 0 go to state B

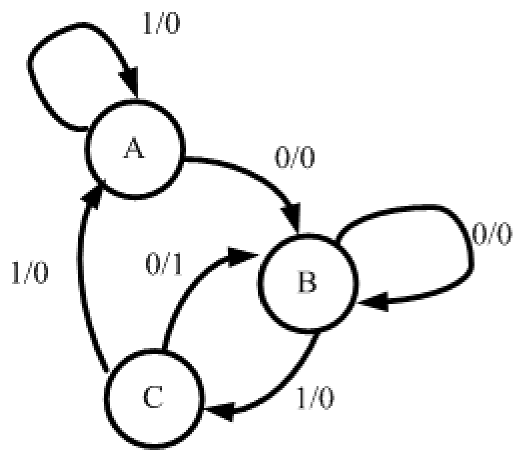


- State B
  - On a 0 stay there
  - On a 1 now have 01 so go to C



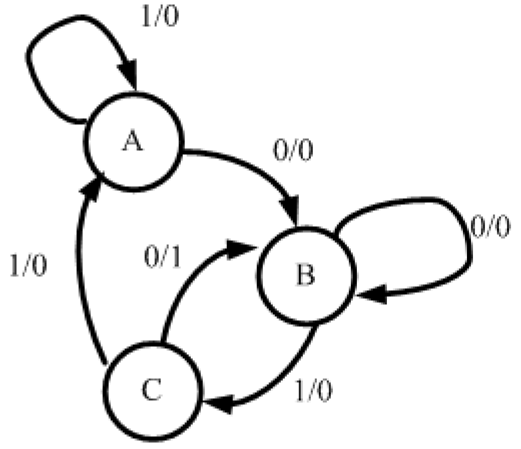
State C

- In state C
  - On a 0 have 010 as last 3, output a 1 and go back to B.
  - On a 1 have a 1 as last input and return to A



Machine and state table

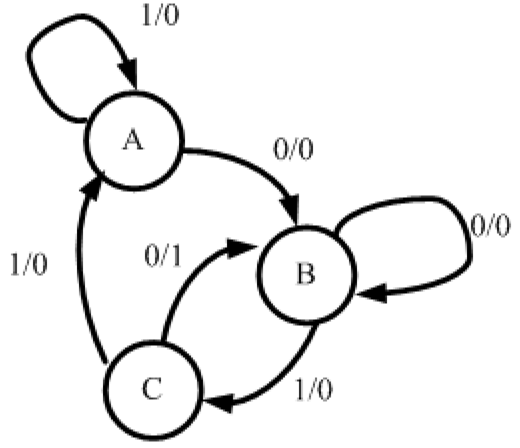
- Can generate a state table for this 010 sequential machine

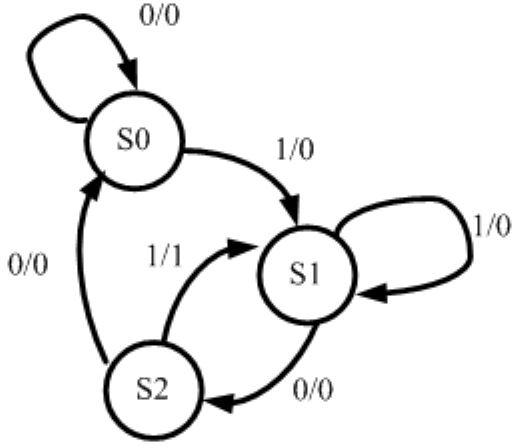


Pr state	Next State X=0	Next State X=1	Output X=0	Output X=1
A	B	A	0	0
B	B	C	0	0
C	B	A	1	0

Question

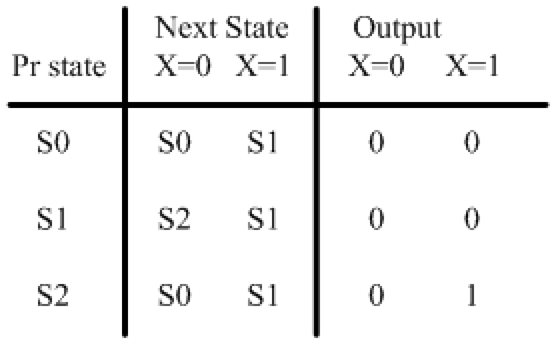
- Are the sequential circuits to detect 010 and 101 as the last 3 equivalent?



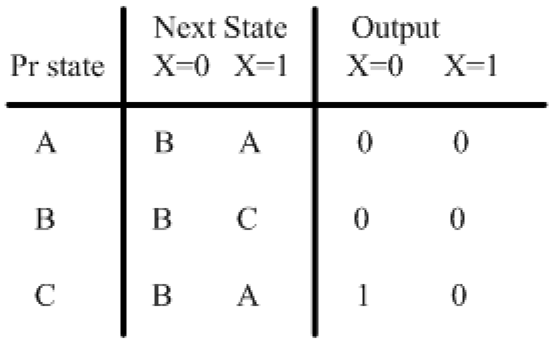


Look at next state tables

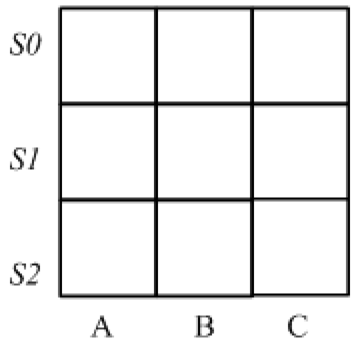
- The tables for the two machines are and develop an equivalence implication table



Pr state	Next State X=0	Next State X=1	Output X=0	Output X=1
S0	S0	S1	0	0
S1	S2	S1	0	0
S2	S0	S1	0	1



Pr state	Next State X=0	Next State X=1	Output X=0	Output X=1
A	B	A	0	0
B	B	C	0	0
C	B	A	1	0



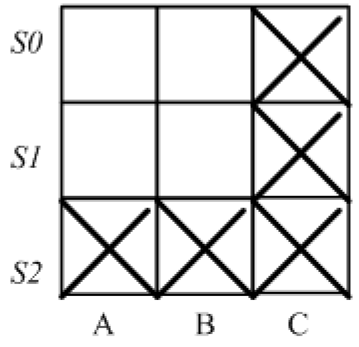
S0			
S1			
S2			
	A	B	C

Start on table

- Mark output incompatable states
  - S2-A,B,C   C-S0,S1,S2

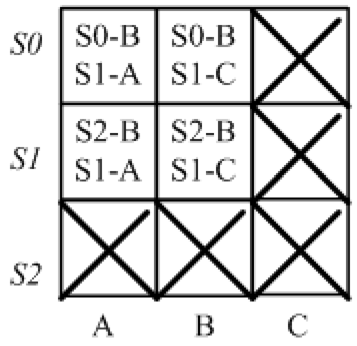
Pr state	Next State X=0	Next State X=1	Output X=0	Output X=1
S0	S0	S1	0	0
S1	S2	S1	0	0
S2	S0	S1	0	1

Pr state	Next State X=0	Next State X=1	Output X=0	Output X=1
A	B	A	0	0
B	B	C	0	0
C	B	A	1	0

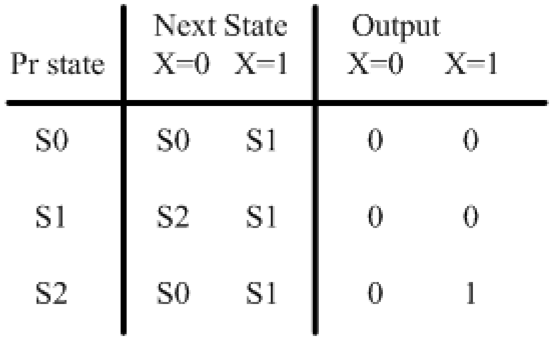


Now the other blocks

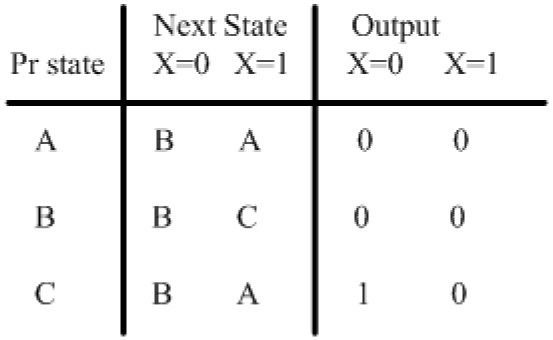
- Implied next states



S0	S0-B S1-A	S0-B S1-C	✗
S1	S2-B S1-A	S2-B S1-C	✗
S2	✗	✗	✗
	A	B	C



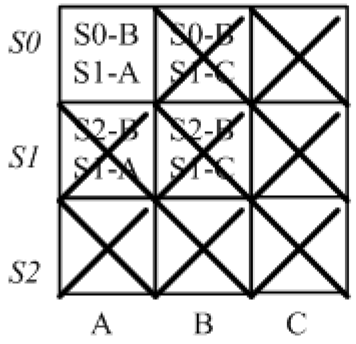
Pr state	Next State X=0	Next State X=1	Output X=0	Output X=1
S0	S0	S1	0	0
S1	S2	S1	0	0
S2	S0	S1	0	1



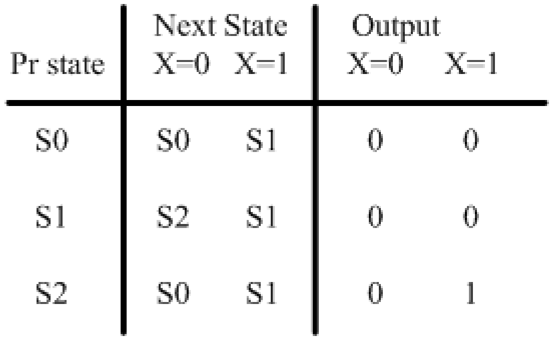
Pr state	Next State X=0	Next State X=1	Output X=0	Output X=1
A	B	A	0	0
B	B	C	0	0
C	B	A	1	0

Check implied next states

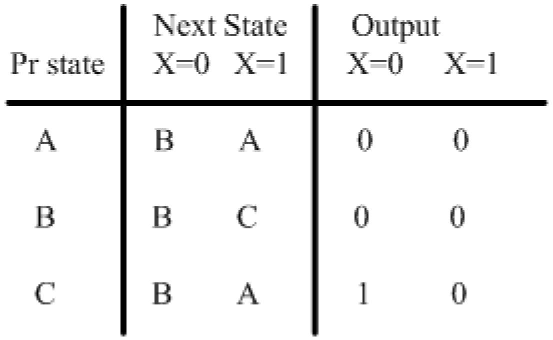
- On pass 1 remove
  - S1-C and S2-B



S0	S0-B S1-A	~~S0-B S1-C~~	✗
S1	~~S2-B S1-A~~	~~S2-B S1-C~~	✗
S2	✗	✗	✗
	A	B	C



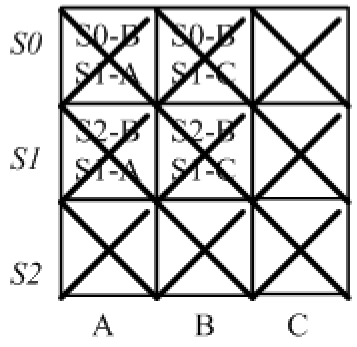
Pr state	Next State X=0	Next State X=1	Output X=0	Output X=1
S0	S0	S1	0	0
S1	S2	S1	0	0
S2	S0	S1	0	1



Pr state	Next State X=0	Next State X=1	Output X=0	Output X=1
A	B	A	0	0
B	B	C	0	0
C	B	A	1	0

Check implied next states

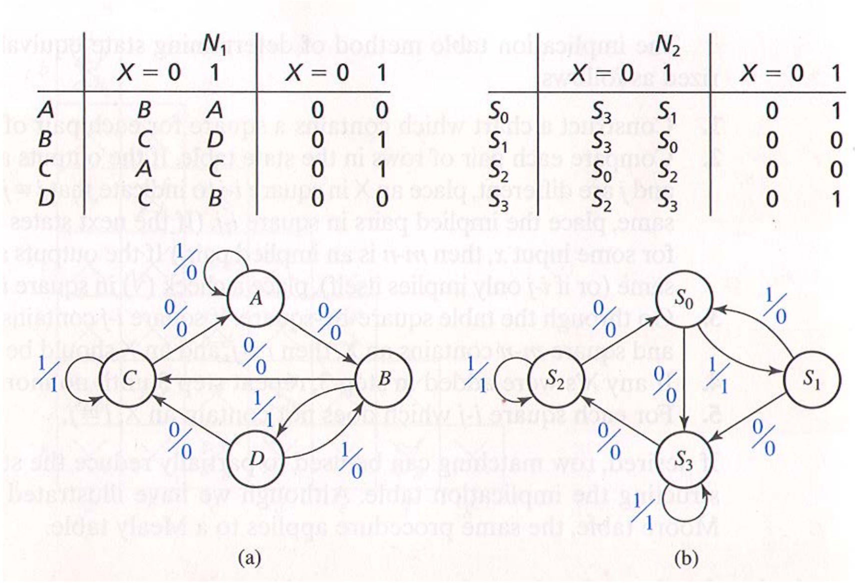
- On pass 2 remove
  - S1-A
  - Now all blocks are Xed



- What does this tell us?
  - The machines are incompatible.
  - To be compatible one block in each row and one block in each column need to be un-Xed and only one.
  - If more than one – says machine incompatible or one or both not minimal

An example

- State tables and state graphs of two sequential machines. Figure 15-6 from the text.
- Equivalent?



N_1

	$X = 0$	1	$X = 0$	1
A	B	A	0	0
B	C	D	0	1
C	A	C	0	1
D	C	B	0	0

N_2

	$X = 0$	1	$X = 0$	1
S_0	S_3	S_1	0	1
S_1	S_3	S_0	0	0
S_2	S_0	S_2	0	0
S_3	S_2	S_3	0	1

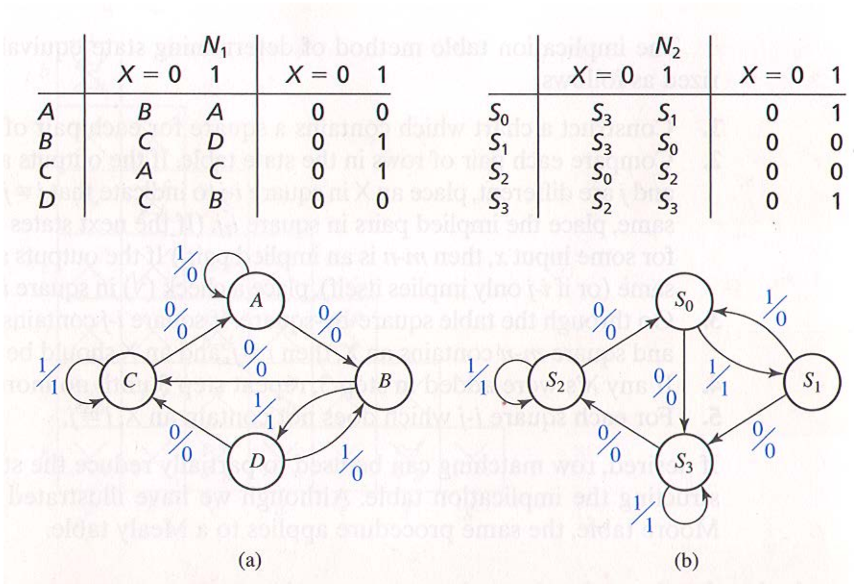
(a) (b)

Proving equivalence

- Again will use an implication table.
  - Only this time, it is the full square.
  - Along bottom are the states of one machine
  - Along the side are the states of the second.
- Start by removing output incompatible states.

The equivalence implication table

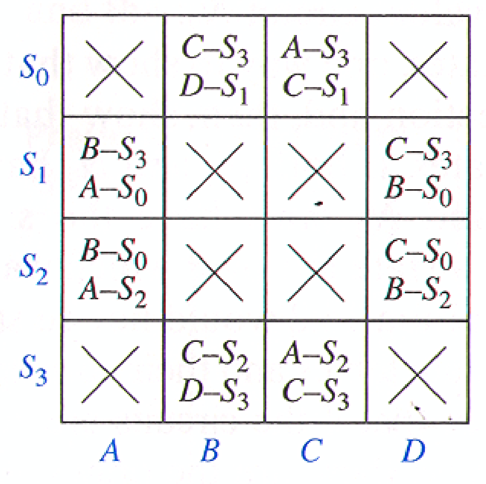
- X squares where the outputs are incompatible
- Enter implied equivalence pairs for remaining states.



	N_1 $X=0$	1	$X=0$	1
A	B	A	0	0
B	C	D	0	1
C	A	C	0	1
D	C	B	0	0

	N_2 $X=0$	1	$X=0$	1
S_0	S_3	S_1	0	1
S_1	S_3	S_0	0	0
S_2	S_0	S_2	0	0
S_3	S_2	S_3	0	1

(a) (b)

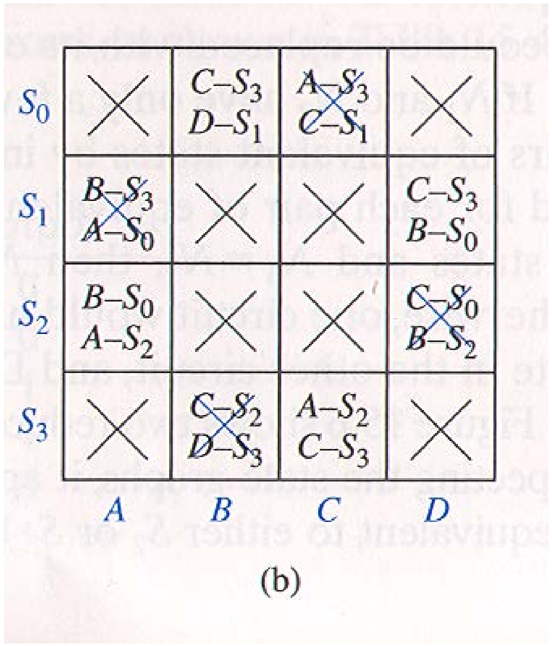


S_0	X	C–S_3 D–S_1	A–S_3 C–S_1	X
S_1	B–S_3 A–S_0	X	X	C–S_3 B–S_0
S_2	B–S_0 A–S_2	X	X	C–S_0 B–S_2
S_3	X	C–S_2 D–S_3	A–S_2 C–S_3	X
	A	B	C	D

(a)

Step 2

- Go back through and remove the implied equivalence pairs that were Xed on the first pass. Continue until no further Xs are entered.
- If there is one square not Xed in each row and each column, the state machines are equivalent. (When both are minimal)
- Consider problem 15-17 in text Does this work if the state tables are of different size?

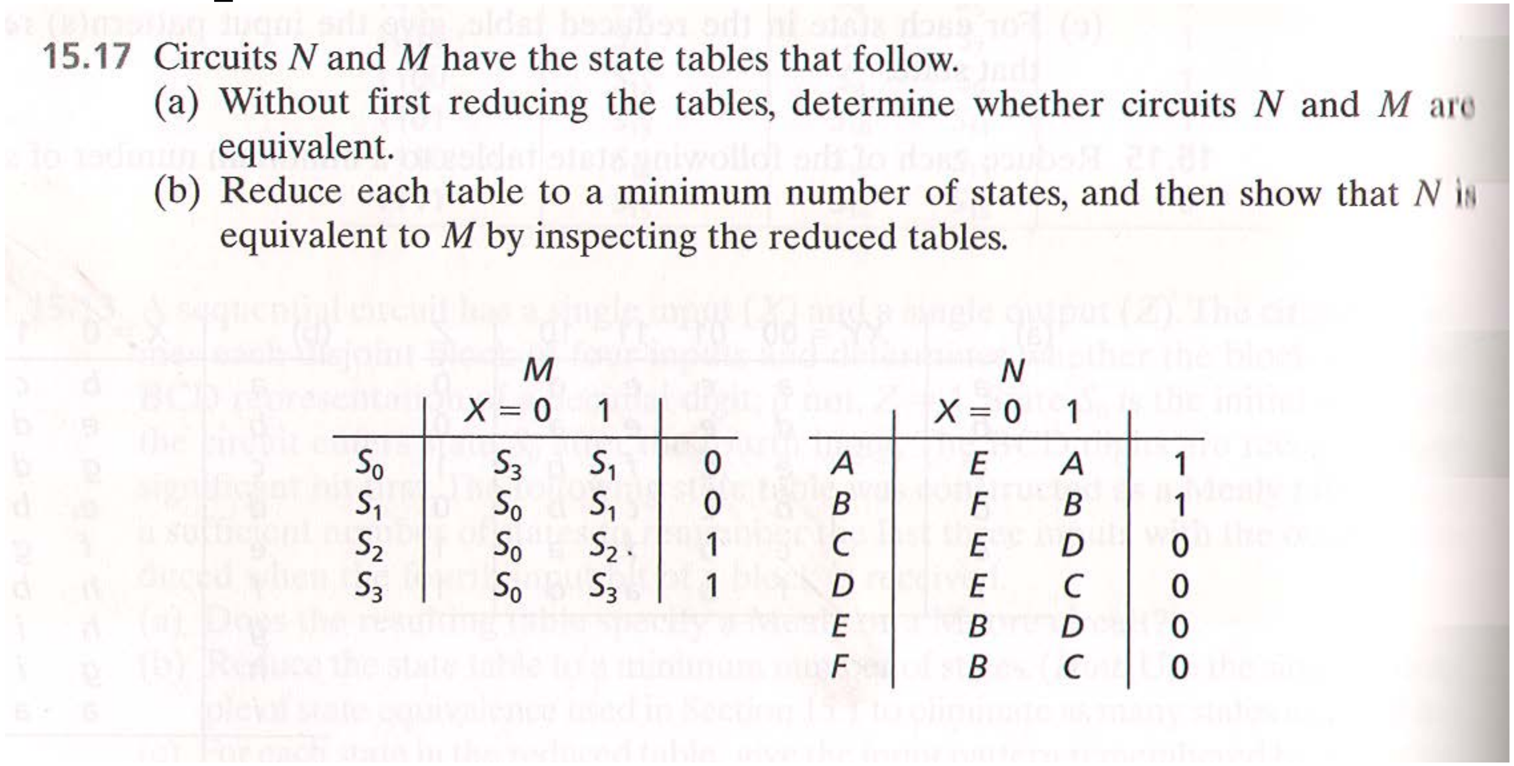


S_0	X	$C-S_3$ $D-S_1$	~~$A-S_3$ $C-S_1$~~	X
S_1	~~$B-S_3$ $A-S_0$~~	X	X	$C-S_3$ $B-S_0$
S_2	$B-S_0$ $A-S_2$	X	X	~~$C-S_0$ $B-S_2$~~
S_3	X	~~$C-S_2$ $D-S_3$~~	$A-S_2$ $C-S_3$	X
	A	B	C	D

(b)

Problem 15.17

- The problem statement



15.17 Circuits N and M have the state tables that follow.

(a) Without first reducing the tables, determine whether circuits N and M are equivalent.

(b) Reduce each table to a minimum number of states, and then show that N is equivalent to M by inspecting the reduced tables.

M

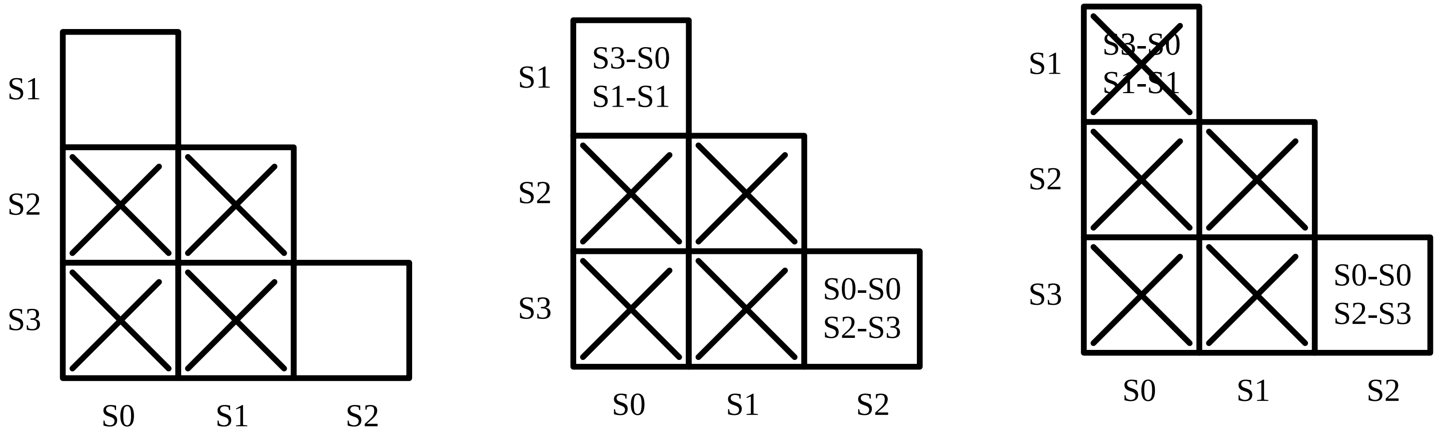
	$X = 0$	1	
S_0	S_3	S_1	0
S_1	S_0	S_1	0
S_2	S_0	S_2	1
S_3	S_0	S_3	1

N

	$X = 0$	1	
A	E	A	1
B	F	B	1
C	E	D	0
D	E	C	0
E	B	D	0
F	B	C	0

Minimize Sx machine

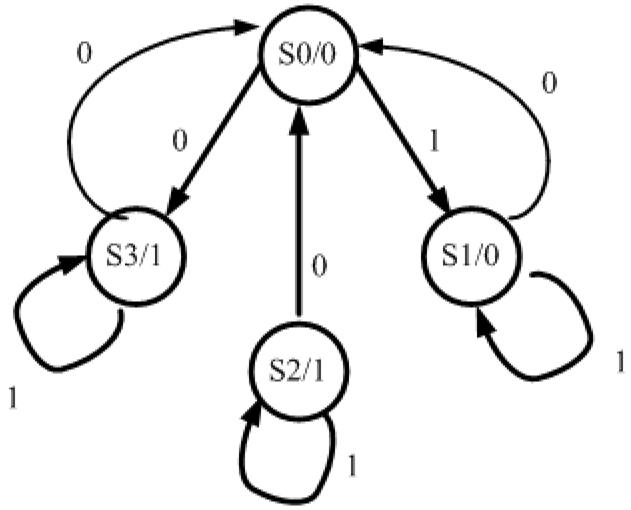
- Can it be reduced?



- YES

Reduced machine

- States S2 and S3 are equivalent – in fact S2 is not reachable unless the machine comes up in that state at startup and it can never reach S2 again.



Significance

- Now consider the equivalence implication table. What is the implication if S2 replaces state S2 and S3?

	A	B	C	D	E	F
S0	✗	✗	~~S3 – E S1 – D~~	~~S3 – E S1 – C~~	S3 – B S1 – D	S3 – B S1 – C
S1	✗	✗	S0 – E S1 – D	S0 – E S1 – C	~~S0 – B S1 – D~~	~~S0 – B S1 – C~~
S2	S0 – E S2 - A	S0 – F S2 – B	✗	✗	✗	✗
S3	S0 – E S3 - A	S0 – F S3 – B	✗	✗	✗	✗

Incompletely Specified

- Incompletely Specified State Tables
  - State tables that contain don’t cares.
  - Results in reduced logic

- Determining the best way to fill in the don’t cares is another of the *n-p complete* problems.
- For this course do the most logical approach.

One Hot

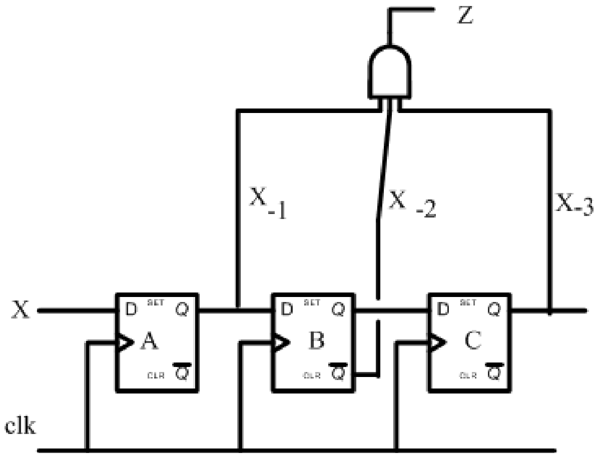
- CPLDs and FPGAs have a good number of F/Fs onboard. The F/Fs are there whether they are used or not, so a circuit with the minimum number of F/Fs is not the ultimate objective.
- For these devices the objective is to reduce the total number of logic cells used and the interconnection between cells.
- One hot encoding is one approach to have shorter signal paths and reduce logic cells.

What is one hot?

- One hot is a method where a flip flop is used for each state in the state machine. A state machine with *n* states will require *n* flip flops in its realization.
- One hot realization is excellent for controllers that step through a set sequence of linear steps.
- Text gives example of a multiplier controller state graph which is not linear.

The full circuit

- Desire Z=1 when X_{-1} X_{-2} X_{-3} is 101.
- Simply construct the combinational logic with inputs from the F/F outputs.

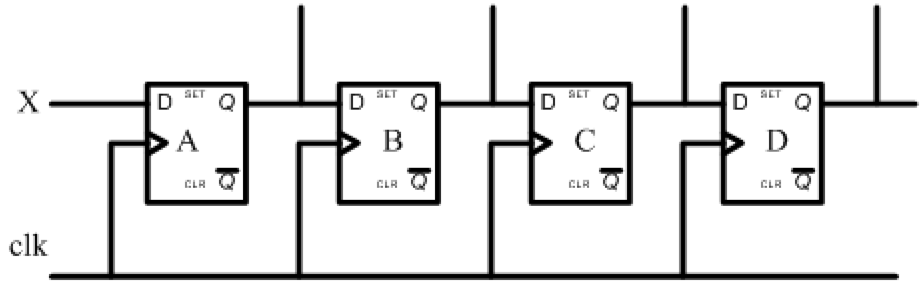


Compare the gates

- Traditional implementation for sequence detector (from text)
  - 2 F/Fs
  - 2 2-input AND gates
  - 1 INV
- One hot implementation
  - 3 F/Fs
  - 1 3-input AND gate

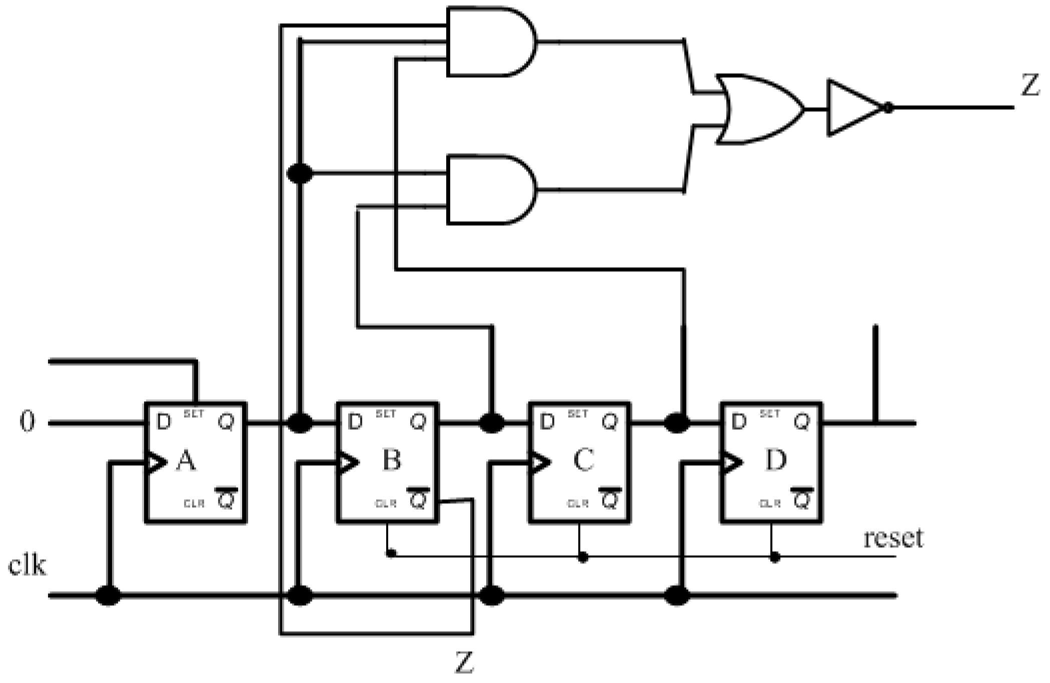
Another example

- Design a circuit to detect when the value represents a BCD digit
- Could also detect when it is not a valid BCD digit, i.e., 10,11,12,13,14,15 which is
  - 1010,1011,1100,1101,1110,1111 or
  - 101x,11xx
  - As last 4 inputs



BCD digit detector

- The logic
- In groups of 4
- Detect when
  - 101x
  - 11xx
- Z=1 valid BCD



Lecture summary

- Have again looked at state machine equivalence.
- One hot encoding and it is interesting for many implementations.