DO NOT SHARE
SLIDES AND CLASS MATERIALS
ON ONLINE SITES

Advanced Logic Design
Finite State Machine

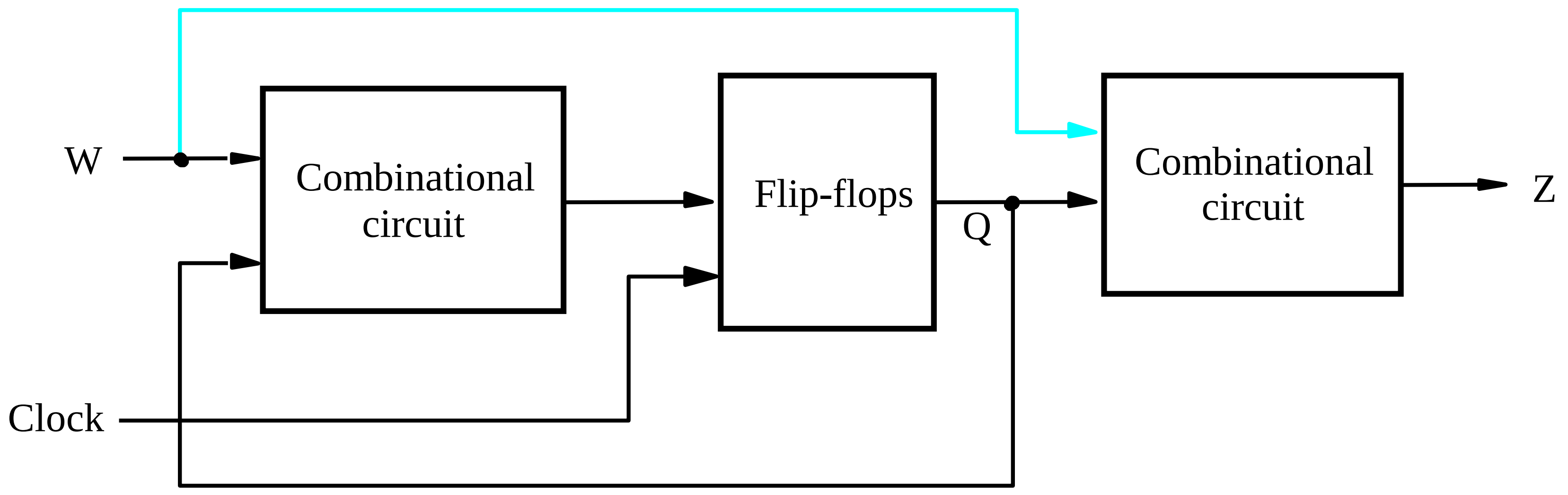
Mingoo Seok
Columbia University

BV: Secs. 6.1-6.3, 6.5, 6.7

Sequential Circuits (aka FSM)

- In a combinational circuit, the values of the outputs are determined solely by the present values of its inputs
- In a sequential circuit, the values of the outputs depend on the past behavior of the circuit, as well as the present values of its inputs
- A sequential circuit has states, which in conjunction with the present values of inputs determine its behavior.
- Sequential circuits can be:
  - Synchronous – where flip-flops are used to implement the states, and a clock signal is used to control the operation
  - Asynchronous – where no clock is used
- Sequential circuit is formally called a finite state machine (FSM) since it has a finite number of states
- It controls the operation (sequencing) of digital circuits

General Form of a Sync. FSM



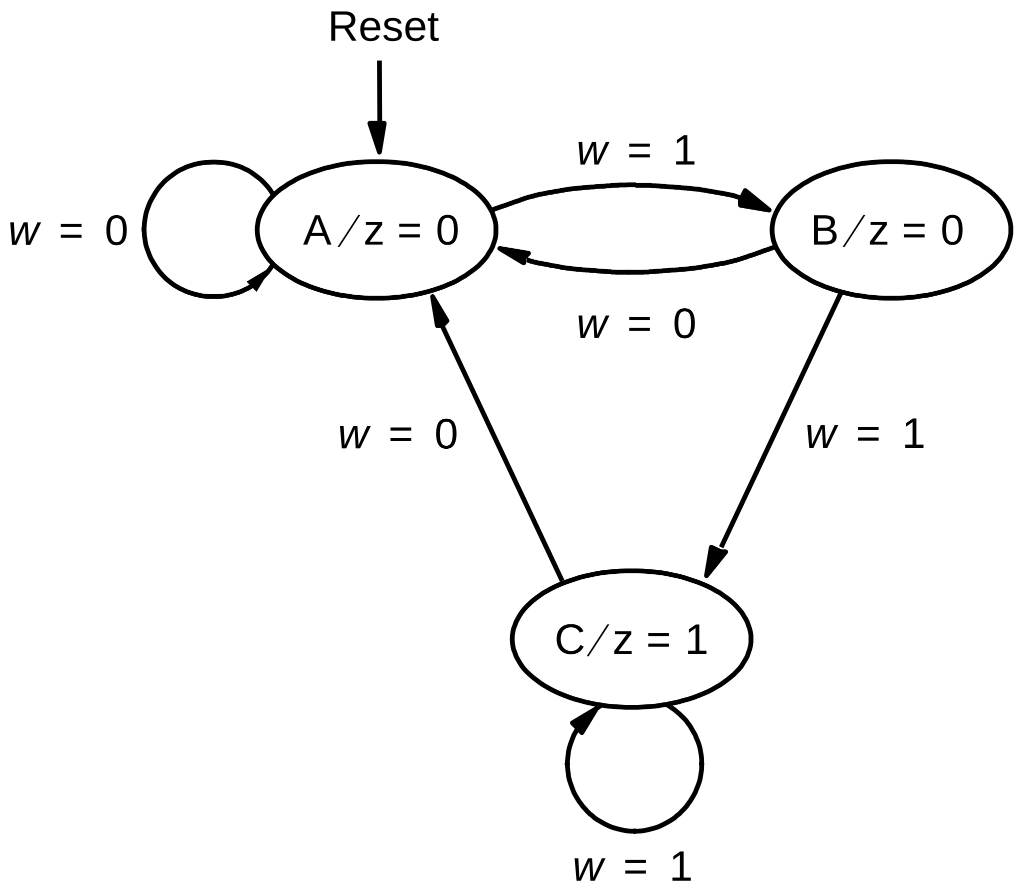
- w/o the blue line: Moore FSM
- w/ the blue line: Mealy FSM.

FSM Design

Clockcycle:	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
w:	0	1	0	1	1	0	1	1	1	0	1
z:	0	0	0	0	0	1	0	0	1	1	0

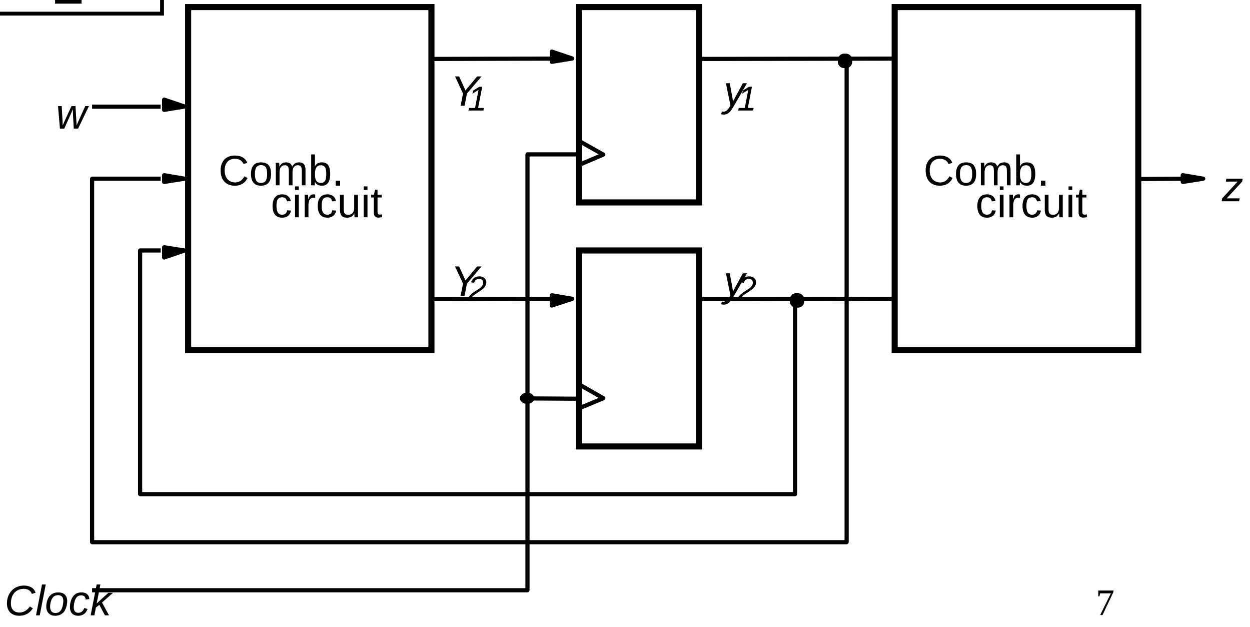
- Example: consider an application where the speed of an automatically-controlled vehicle has to be regulated
  - Input w is 1 if the speed is higher than the target; 0 otherwise
  - The output z is equal to 1 if during two immediately preceding clock cycles the input w was equal to 1; 0 otherwise.
  - Output z is 1 to decelerate
  - All changes in the logic circuit occur on the positive edge of the clock signal

FSM Design



State Table

Present state	Next state		Output
	$w = 0$	$w = 1$	z
A	A	B	0
B	A	C	0
C	A	C	1

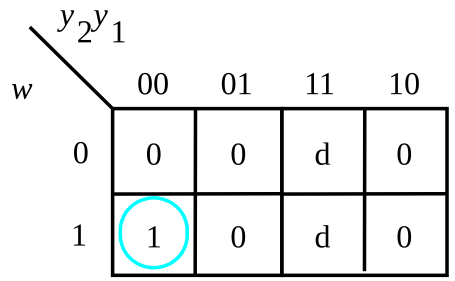


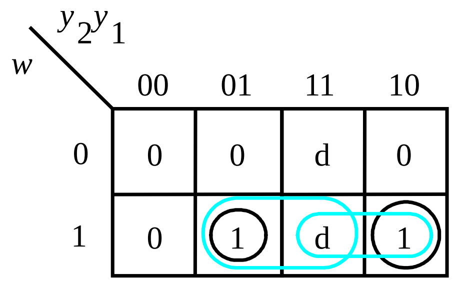
State Table

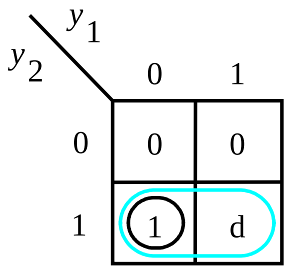
Present state	Next state		Output z
	$w = 0$	$w = 1$	
A	A	B	0
B	A	C	0
C	A	C	1

	Present state	Next state		Output z
		$w = 0$	$w = 1$	
	$y_2 y_1$	$Y_2 Y_1$	$Y_2 Y_1$	
A	00	00	01	0
B	01	00	10	0
C	10	00	10	1
	11	*dd*	*dd*	*d*

Logic Synthesis

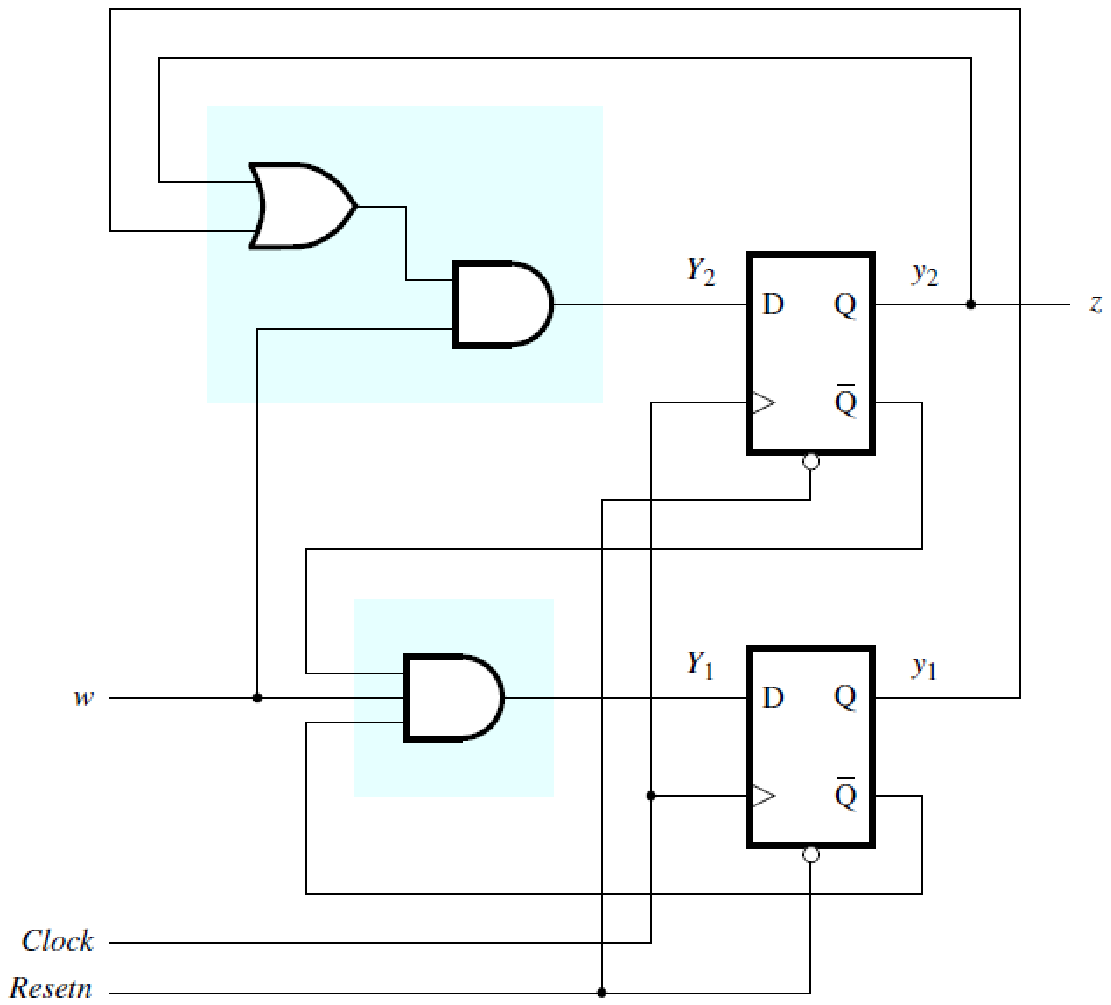






Ignoring don't cares	Using don't cares
$Y_1 = w\bar{y}_1\bar{y}_2$	$Y_1 = w\bar{y}_1\bar{y}_2$
$Y_2 = wy_1\bar{y}_2 + w\bar{y}_1y_2$	$Y_2 = wy_1 + wy_2$ $= w(y_1 + y_2)$
$z = \bar{y}_1y_2$	$z = y_2$

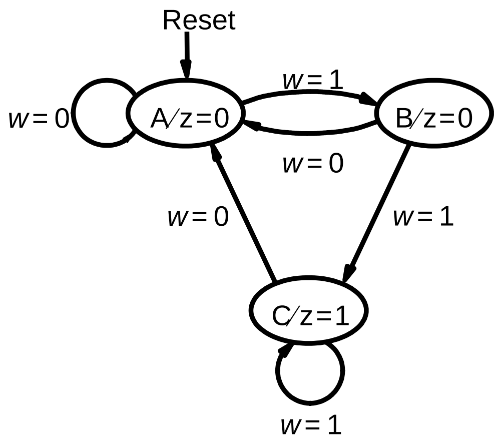
Logic Synthesis

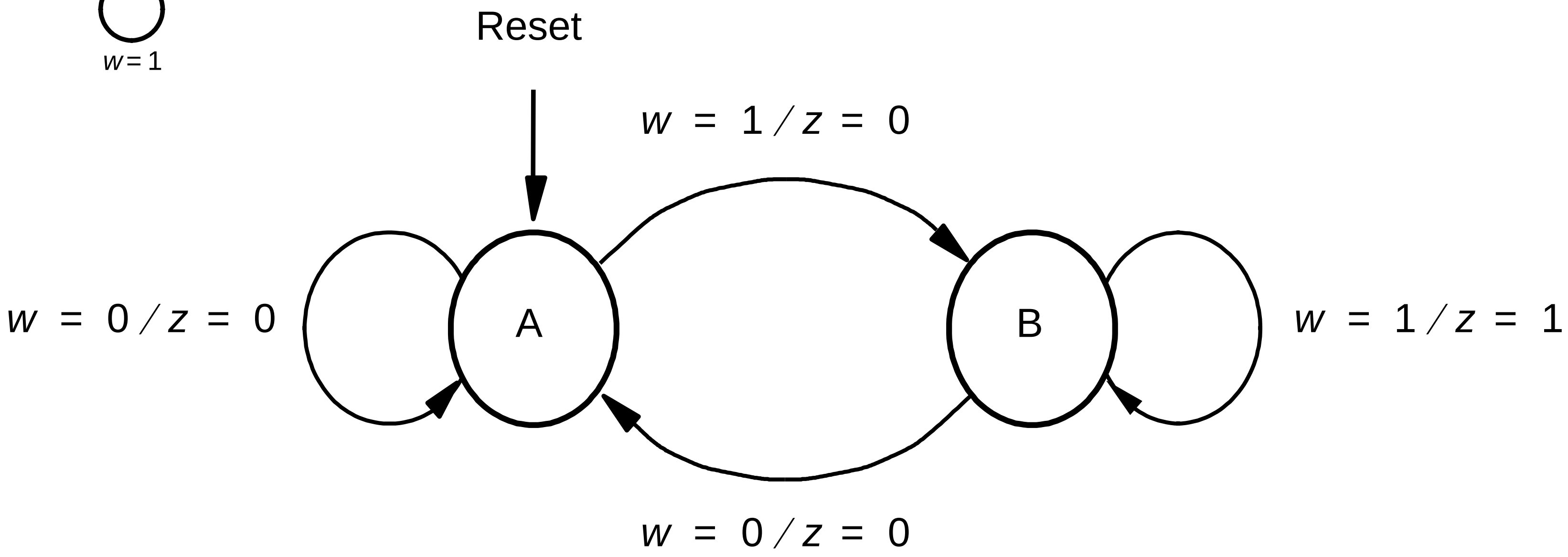


Summary of Design Steps

1. Obtain the specification of the desired circuit.
2. Derive a state diagram.
3. Derive the corresponding state table.
4. Reduce the number of states if possible.
5. Decide on the number of state variables.
6. Choose the type of flip-flops to be used.
7. Derive the logic expressions needed to implement the circuit.

Moore to Mealy



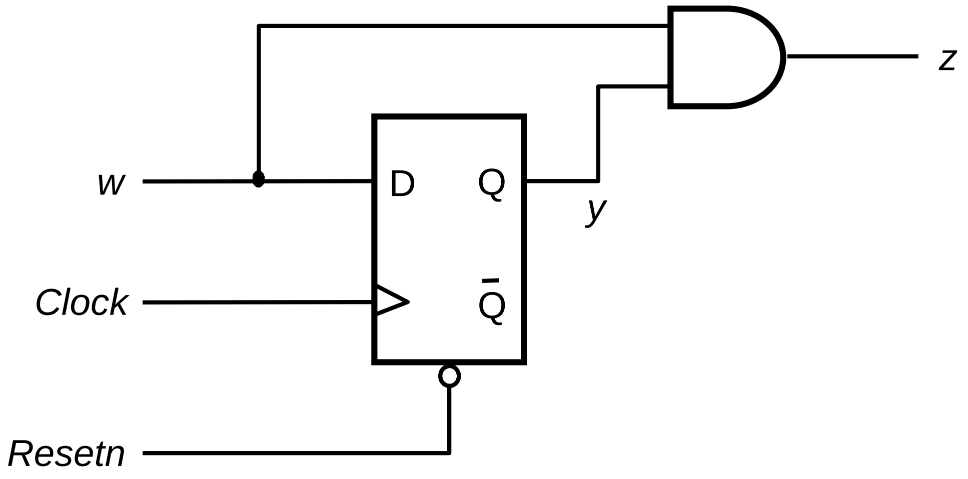


State Table

Present state	Next state		Output z	
	$w = 0$	$w = 1$	$w = 0$	$w = 1$
A	A	B	0	0
B	A	B	0	1

	Present state	Next state		Output	
		$w = 0$	$w = 1$	$w = 0$	$w = 1$
	y	Y	Y	z	z
A	0	0	1	0	0
B	1	0	1	0	1

Logic Synthesis



- Significantly simpler than the Moore implementation
- Faster
- A bit more complex to analyze

Better State Assignment

	Present state y_2y_1	Next state $w=0$ Y_2Y_1	Next state $w=1$ Y_2Y_1	Output z
A	00	00	01	0
B	01	00	10	0
C	**10**	00	10	1
	11	*dd*	*dd*	*d*

	Present state y_2y_1	Next state $w=0$ Y_2Y_1	Next state $w=1$ Y_2Y_1	Output z
A	00	00	01	0
B	01	00	11	0
C	11	00	11	1
	10	*dd*	*dd*	*d*

$$Y_1 = w\bar{y}_1\bar{y}_2$$

$$Y_2 = w(y_1 + y_2)$$

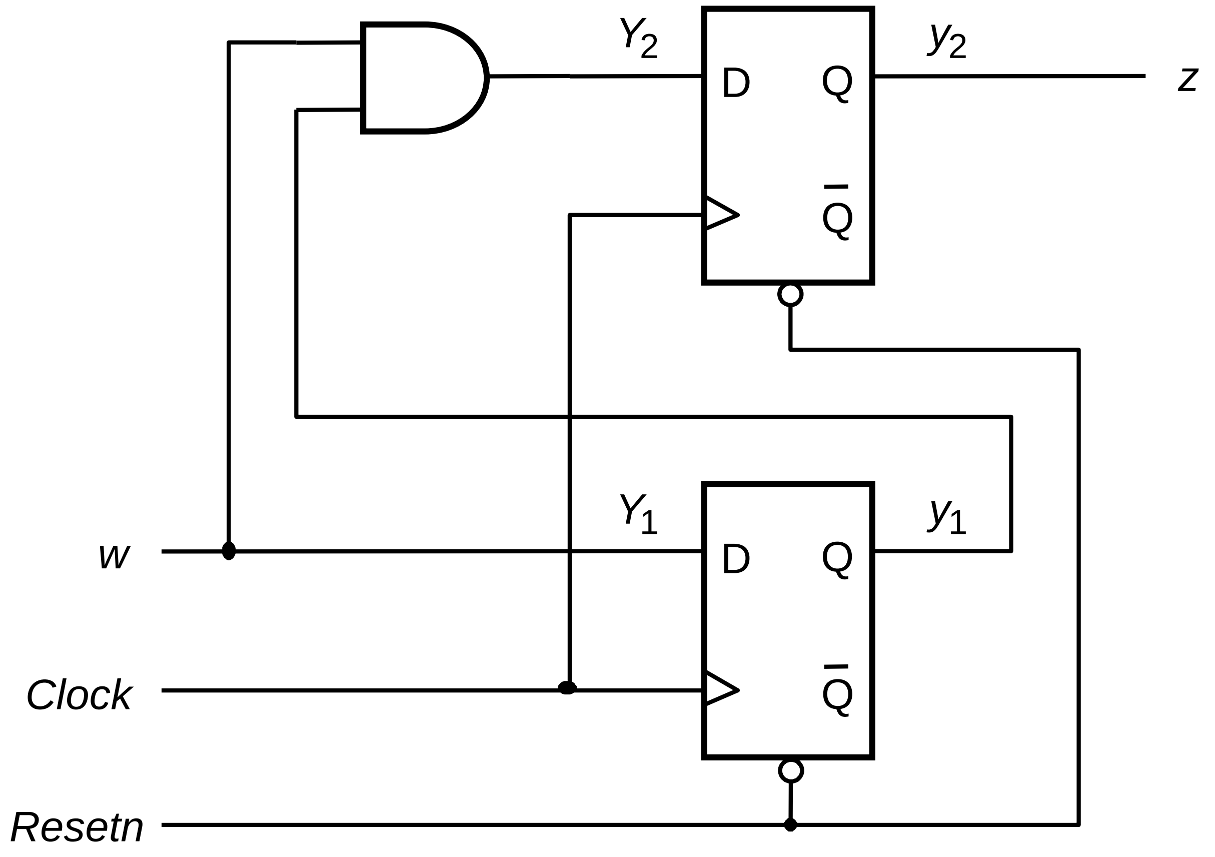
$$z = y_2$$

$$Y_1 = D_1 = w$$

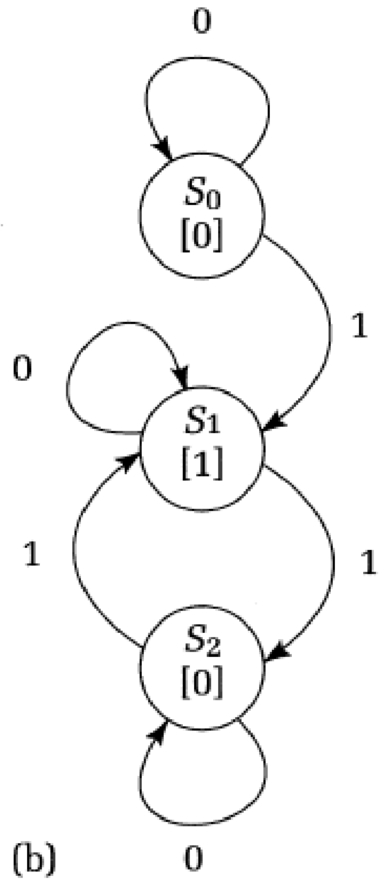
$$Y_2 = D_2 = wy_1$$

$$z = y_2$$

Better State Assignment



FSM Optimization



- The smaller # of states reduces # of flip-flops, logic circuits complexity, and delay
- S_0 and S_2 are equivalent
  - Same outputs
  - Same state transitions
- Row matching method
- Implication chart method

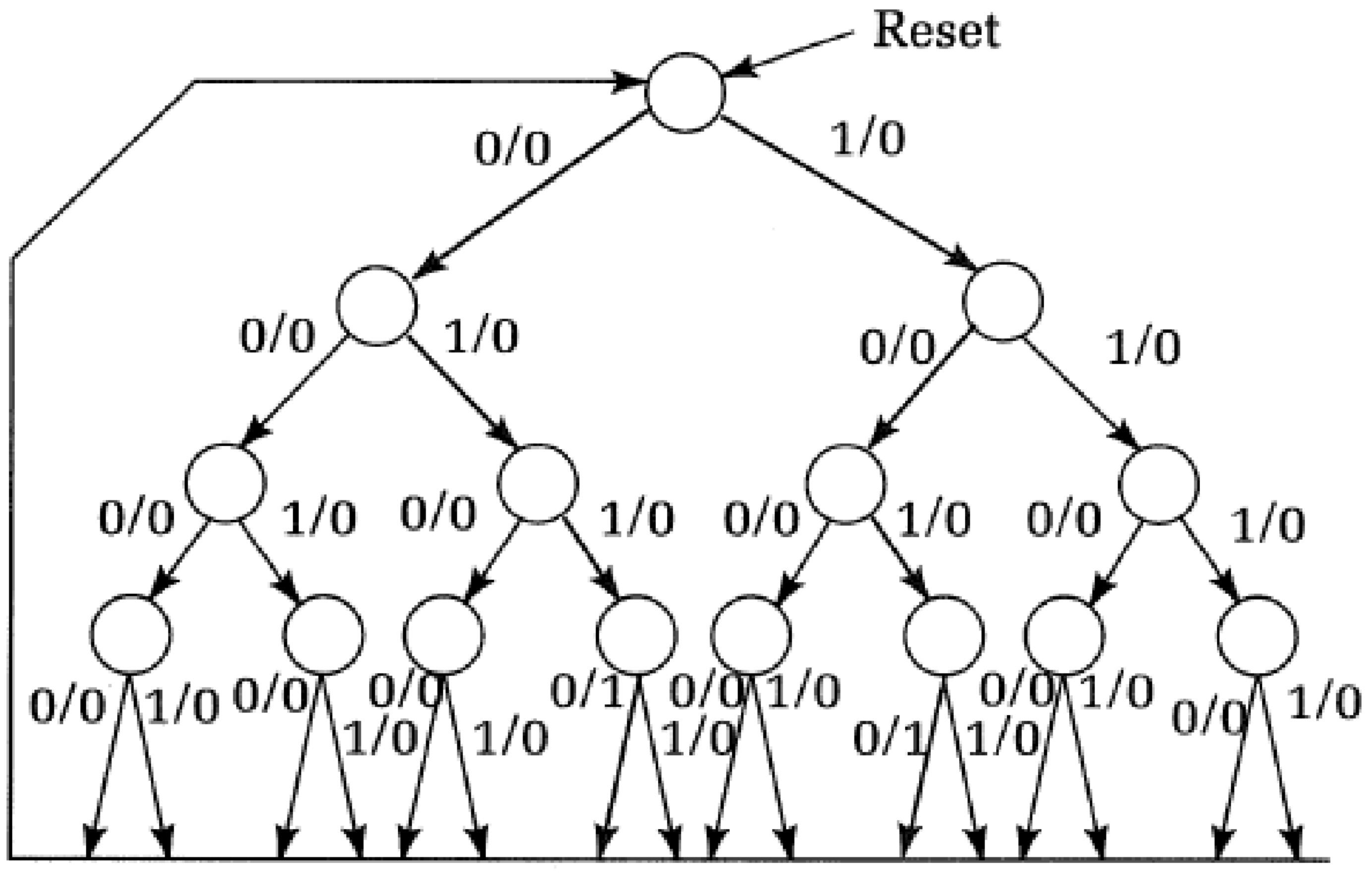
Row Matching Method

$$X = 0010\ 0110\ 1100\ 1010\ 0011\ \ldots$$

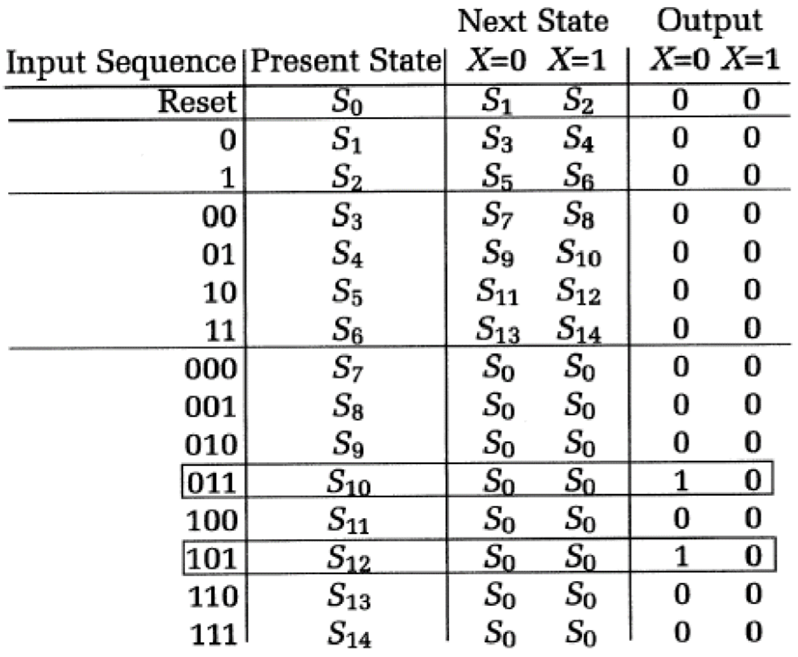
$$Z = 0000\ 0001\ 0000\ 0001\ 0000\ \ldots$$

- Input X and Output Z
- Output is asserted if each 4-bit input sequence is either 0110 or 1010
- Row matching method: quick but not the optimum

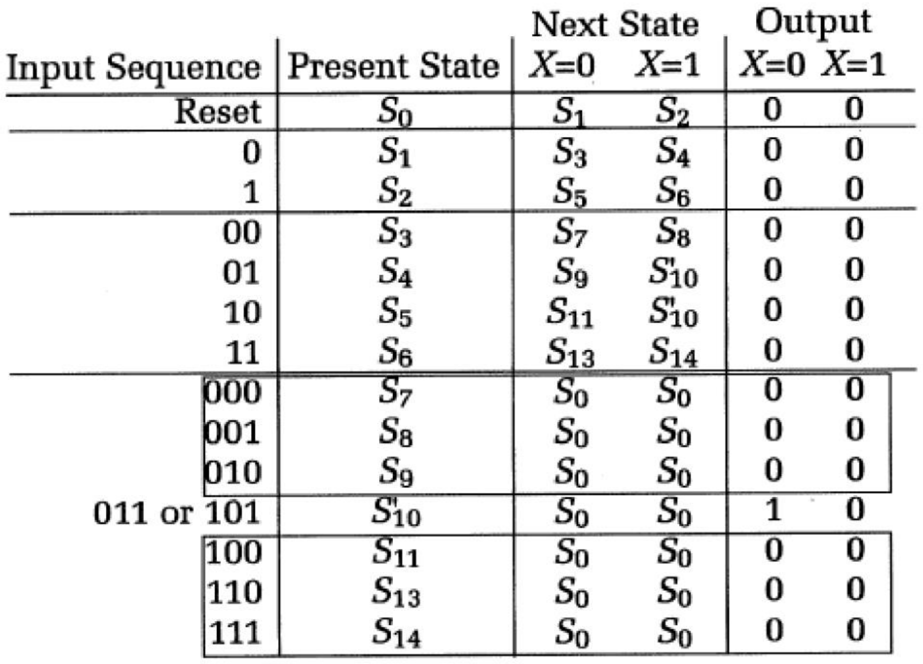
Row Matching Method



Row Matching Method

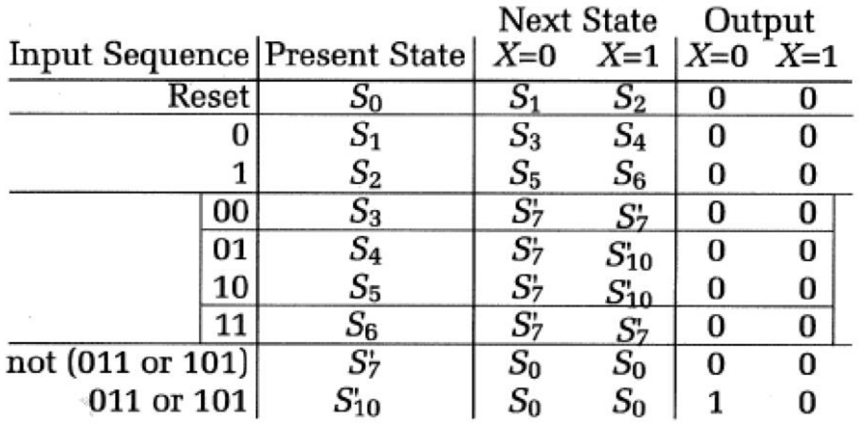


		Next State		Output	
Input Sequence	Present State	X=0	X=1	X=0	X=1
Reset	S_0	S_1	S_2	0	0
0	S_1	S_3	S_4	0	0
1	S_2	S_5	S_6	0	0
00	S_3	S_7	S_8	0	0
01	S_4	S_9	S_{10}	0	0
10	S_5	S_{11}	S_{12}	0	0
11	S_6	S_{13}	S_{14}	0	0
000	S_7	S_0	S_0	0	0
001	S_8	S_0	S_0	0	0
010	S_9	S_0	S_0	0	0
011	S_{10}	S_0	S_0	1	0
100	S_{11}	S_0	S_0	0	0
101	S_{12}	S_0	S_0	1	0
110	S_{13}	S_0	S_0	0	0
111	S_{14}	S_0	S_0	0	0

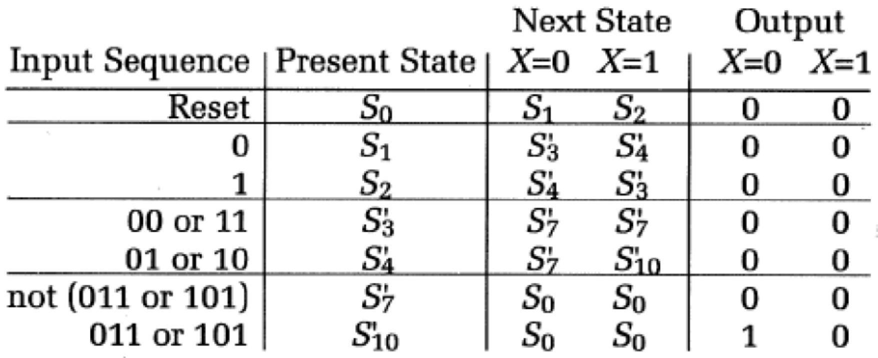


		Next State		Output	
Input Sequence	Present State	X=0	X=1	X=0	X=1
Reset	S_0	S_1	S_2	0	0
0	S_1	S_3	S_4	0	0
1	S_2	S_5	S_6	0	0
00	S_3	S_7	S_8	0	0
01	S_4	S_9	S'_{10}	0	0
10	S_5	S_{11}	S'_{10}	0	0
11	S_6	S_{13}	S_{14}	0	0
000	S_7	S_0	S_0	0	0
001	S_8	S_0	S_0	0	0
010	S_9	S_0	S_0	0	0
011 or 101	S'_{10}	S_0	S_0	1	0
100	S_{11}	S_0	S_0	0	0
110	S_{13}	S_0	S_0	0	0
111	S_{14}	S_0	S_0	0	0

Row Matching Method

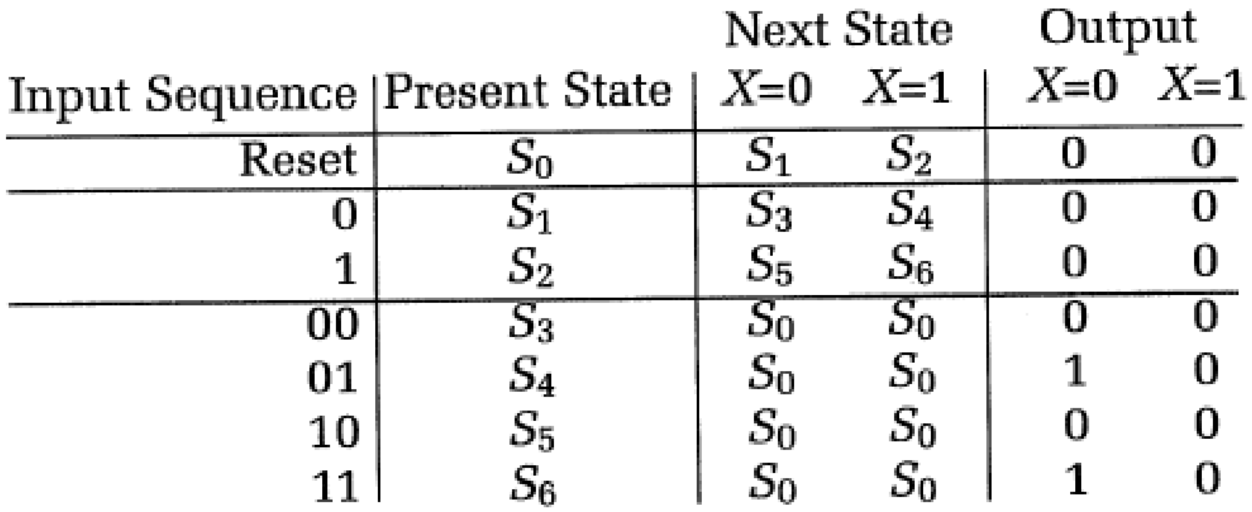


Input Sequence	Present State	Next State X=0	Next State X=1	Output X=0	Output X=1
Reset	S_0	S_1	S_2	0	0
0	S_1	S_3	S_4	0	0
1	S_2	S_5	S_6	0	0
00	S_3	S'_7	S'_7	0	0
01	S_4	S'_7	S'_{10}	0	0
10	S_5	S'_7	S'_{10}	0	0
11	S_6	S'_7	S'_7	0	0
not (011 or 101)	S'_7	S_0	S_0	0	0
011 or 101	S'_{10}	S_0	S_0	1	0



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'_4	S'_3	0	0
00 or 11	S'_3	S'_7	S'_7	0	0
01 or 10	S'_4	S'_7	S'_{10}	0	0
not (011 or 101)	S'_7	S_0	S_0	0	0
011 or 101	S'_{10}	S_0	S_0	1	0

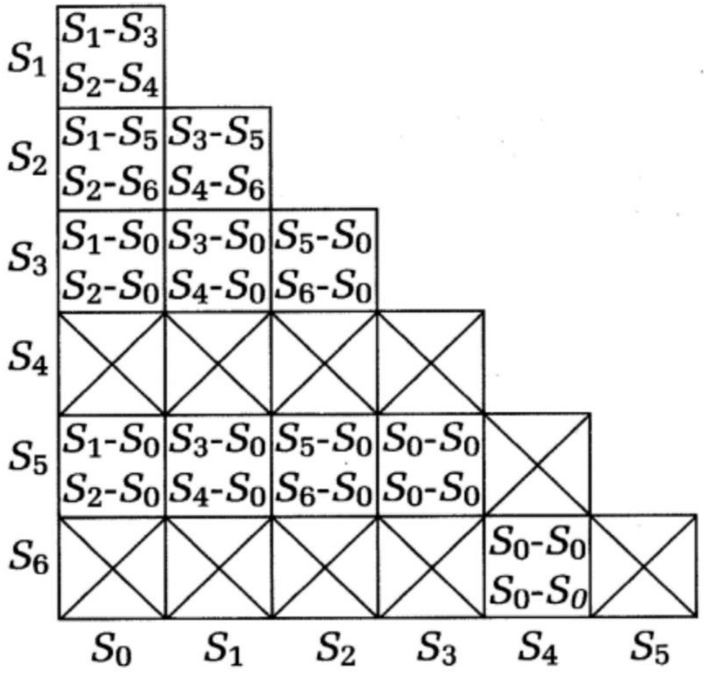
Implication Chart Method

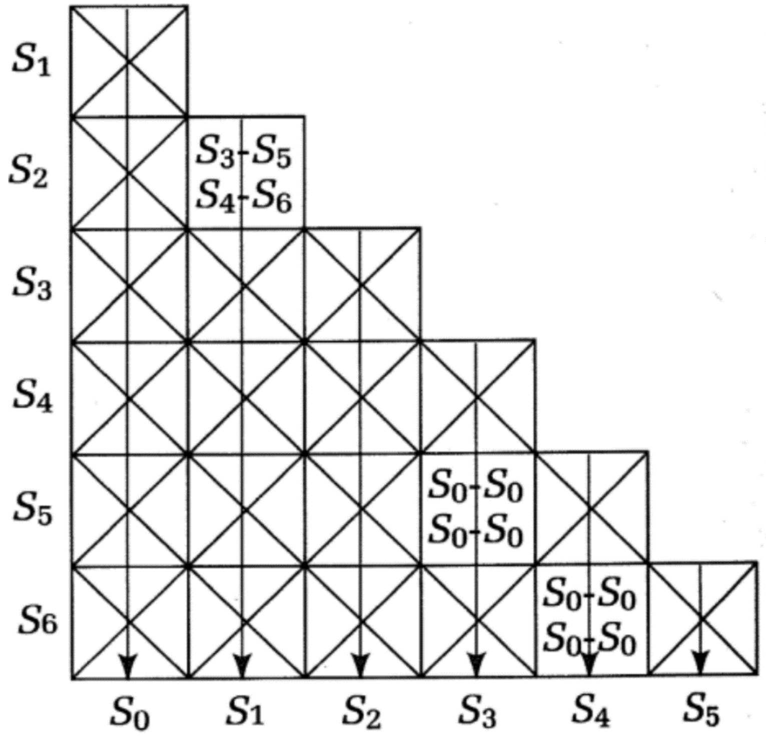


Input Sequence	Present State	Next State		Output	
		$X=0$	$X=1$	$X=0$	$X=1$
Reset	S_0	S_1	S_2	0	0
0	S_1	S_3	S_4	0	0
1	S_2	S_5	S_6	0	0
00	S_3	S_0	S_0	0	0
01	S_4	S_0	S_0	1	0
10	S_5	S_0	S_0	0	0
11	S_6	S_0	S_0	1	0

- 3-bit sequence detector that will output a 1 whenever the machine has observed the serial sequence 010 or 110 at the inputs

Implication Chart Method

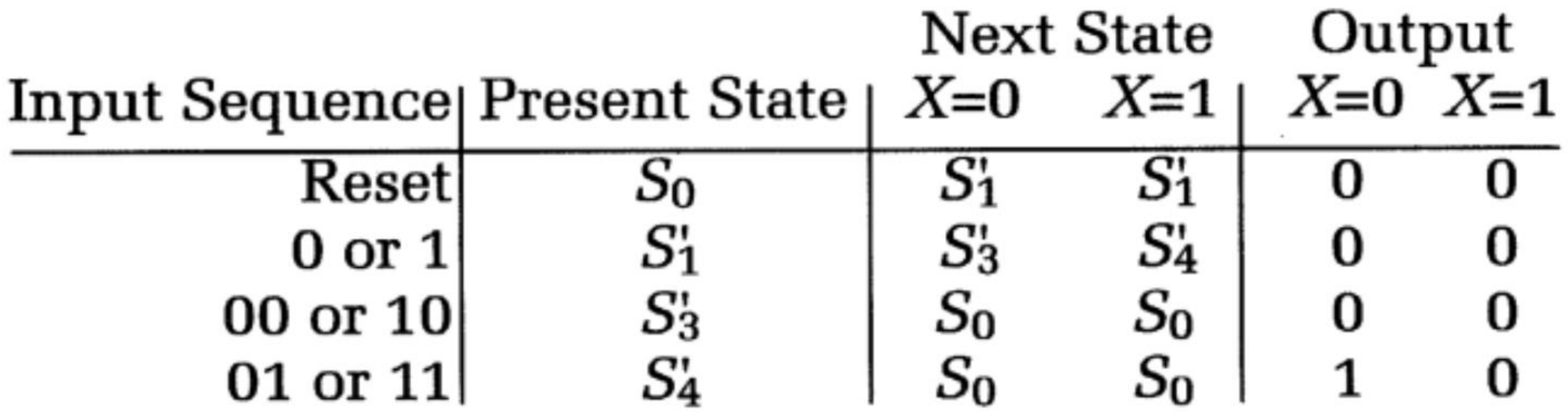




Write the possible equivalent state pairs in each slot

Eliminate the slots that have been proved definitely not equivalent

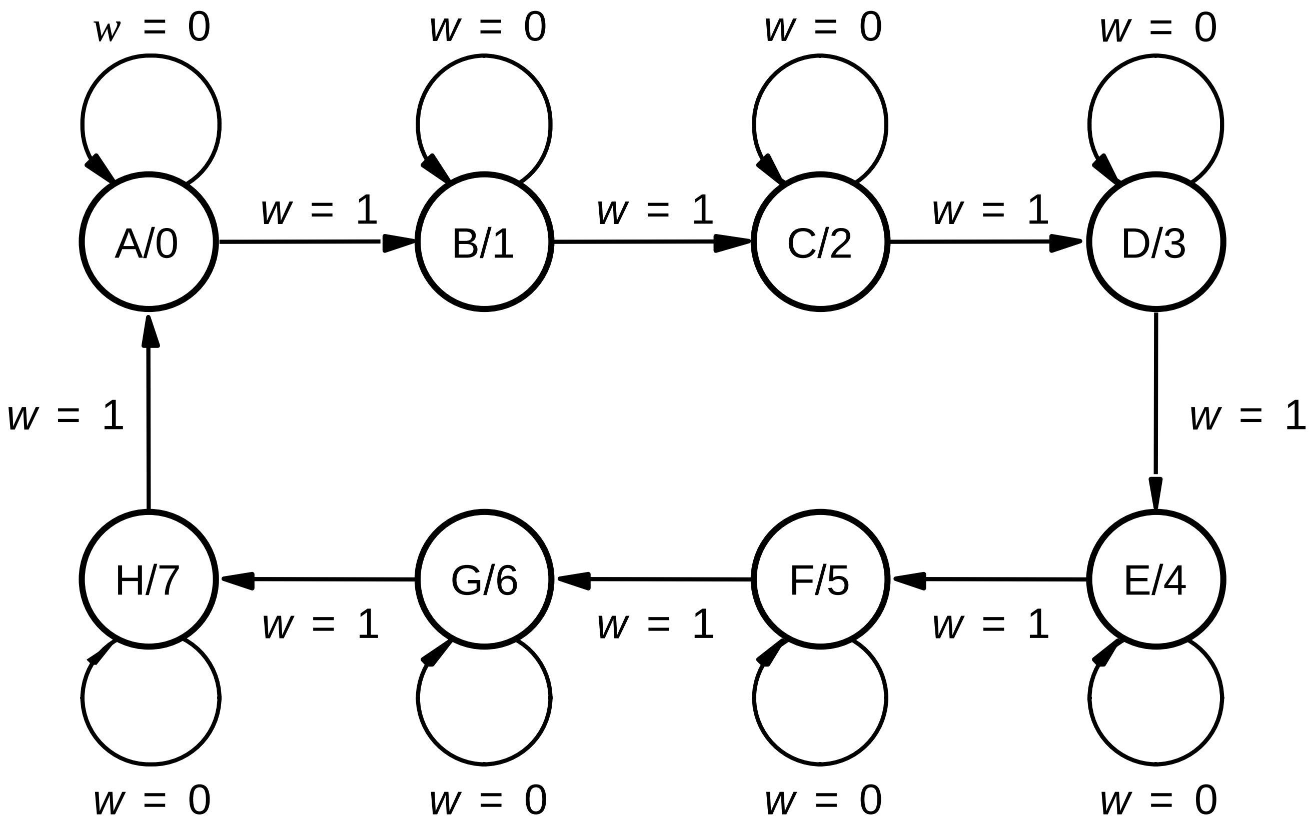
Implication Chart Method



		Next State		Output	
Input Sequence	Present State	X=0	X=1	X=0	X=1
Reset	S_0	S'_1	S'_1	0	0
0 or 1	S'_1	S'_3	S'_4	0	0
00 or 10	S'_3	S_0	S_0	0	0
01 or 11	S'_4	S_0	S_0	1	0

- State # reduces from 7 to 4
- 3 flip-flops to 2 flip-flops
- Associated logic reduction

Counter as an FSM: Modulo-8 Counter



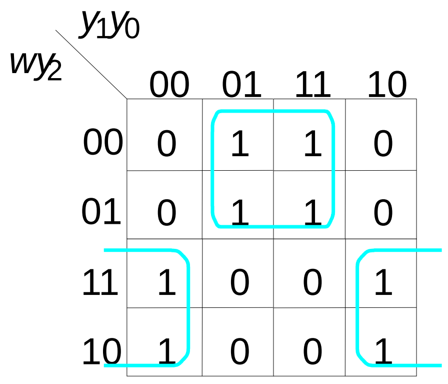
Modulo-8 Counter

Present state	Next state		Output
	$w = 0$	$w = 1$	
A	A	B	0
B	B	C	1
C	C	D	2
D	D	E	3
E	E	F	4
F	F	G	5
G	G	H	6
H	H	A	7

Modulo-8 Counter

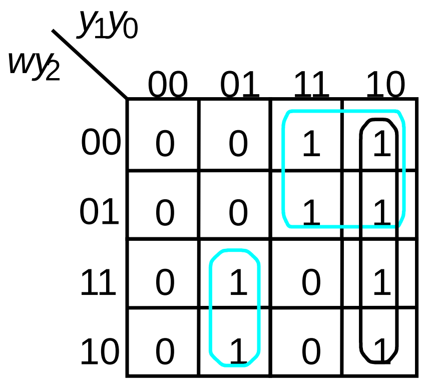
	Present state $y_2y_1y_0$	Next state		Count $z_2z_1z_0$
		$w = 0$	$w = 1$	
		$Y_2Y_1Y_0$	$Y_2Y_1Y_0$	
A	000	000	001	000
B	001	001	010	001
C	010	010	011	010
D	011	011	100	011
E	100	100	101	100
F	101	101	110	101
G	110	110	111	110
H	111	111	000	111

Modulo-8 Counter



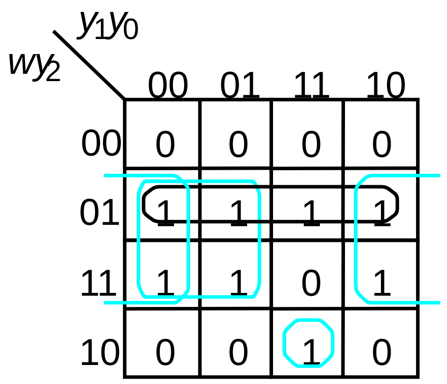
$wy_2 \backslash y_1y_0$	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	1	0	0	1
10	1	0	0	1

$Y_0 = \bar{w}y_0 + w\bar{y}_0$



$wy_2 \backslash y_1y_0$	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	0	1	0	1
10	0	1	0	1

$Y_1 = \bar{w}y_1 + y_1\bar{y}_0 + wy_0\bar{y}_1$



$wy_2 \backslash y_1y_0$	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	1	1	0	1
10	0	0	1	0

$Y_2 = \bar{w}y_2 + \bar{y}_0y_2 + \bar{y}_1y_2 + wy_0y_1\bar{y}_2$