ECEN 248 –Introduction to Digital Systems Design (Spring 2008)

(Sections: 501, 502, 503, 507)

Prof. Xi Zhang
ECE Dept, TAMU, 333N WERC
http://ece.tamu.edu/~xizhang/ECEN248

Section 8.6 State minimization

- Why do we need state minimization
  - When we design a more complex FSM, it is very likely that the initial attempt will result in a machine that has more states than is usually required.
  - Reducing the number of required flip-flops and thus the complexity of the combination circuit in the FSM

Equivalent states

- Definition

  Tow states S_i and S_j are said to be equivalent if and only if for every possible input sequence, the same output sequence will be produced regardless of whether S_i or S_j is the initial state.

- If the number of states in an FSM can be reduced, then some states in the original design must be equivalent to other states.
- Instead of trying to show that some states in a given FSM are equivalent, it is often easier to show some states are definitely not equivalent. This idea leads to a simple minimization procedure.

Portioning Minimization Procedure

- Consider a FSM with single input w.
- successor:
  - If w=0 is applied in state S_i and causes the state moving to S_u, S_u is a 0-successor of S_i.
  - If w=1 is applied in state S_i and it causes the FSM to state S_v, S_v is a 1-successor of S_i.
  - Generally, we use k-successors, where k is 0 or 1.
  - For multiple inputs, "k" of k-successors can be any element in the set of all possible combinations (valuations) of the (a number of) inputs.
- If S_i and S_j are equivalent, their corresponding k-successors (for all k) are also equivalent

Definition of Partition

- A partition consists of one or more blocks, where each block comprises a subset of states that may be equivalent, but the states in a given block are definitely not equivalent to the states in other blocks.

Partition procedures and algorithms

- $n = 1$.
- Step 1 (initial partition): $n = 1$
  - Assume that all states are equivalent, forming the initial partition P_1.
- Step 2: $n = 2$
  - Form the partition P_2 in which the set of states is partitioned into blocks such that the states in each block generate the same output values.
- Step 3: $n = 3$
  - Check each block in P_{n-1}. The states whose k-successors are in different blocks cannot be in one block.
  - Form new blocks for these states in new partition P_n.
- Step n:
  - Repeat the same procedure in Step 3 on P_n;
  - If $P_n = P_{n-1}$, we find the final partition and then the state minimization stop.

Procedures demonstrated by examples.

Present state	Next state		Output z
	$w = 0$	$w = 1$	
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

Figure 8.51. State table for Example 8.5.

- Initial partition:
  - P1 = (ABCDEFG)
- Step 2:
  - P2 = (ABD)(CEFG), because of different output

Procedures by example.

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

- Initial partition:
  - P1 = (ABCDEFG)
- Step 2:
  - P2 = (ABD)(CEFG),
- Step 3:
  - (ABD)

(ABD)

because their k-successors are in the same blocks in either (ABD) or (CEFG), respectively.

Procedures by example.

Present state	Next state $w = 0$	Next state $w = 1$	Output z
C	F	E	0
E	F	C	0
F	E	D	0
G	F	G	0

- Initial partition:
  - P1 = (ABCDEFG)
- Step 2:
  - P2 = (ABD)(CEFG),
- Step 3:
  - (ABD) (CEFG),

    (ABD) (CEG)(F),

    Because F yields different 1-sucessor as compared to C, E, and G.

Procedures by example.

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

- Initial partition:
  - P1 = (ABCDEFG)
- Step 2:
  - P2 = (ABD)(CEFG),
- Step 3:
  - P3 = (ABD)(CEG)(F).
- Step 4:
  - (ABD)
  - (AD)(B)
    Because B yields a different 1-sucessor

Procedures by example.

Present state	Next state $w = 0$	Next state $w = 1$	Output z
C	F	E	0
E	F	C	0
G	F	G	0

- Initial partition:
  - P1 = (ABCDEFG)
- Step 2:
  - P2 = (ABD)(CEFG),
- Step 3:
  - P3 = (ABD)(CEG)(F).
- Step 4:
  - (CEG)
  - (CEG)
    Because no further different successors → no further partitions

Procedures by example.

Present state	Next state		Output
	$w = 0$	$w = 1$	z
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

- Initial partition:
  - P1 = (ABCDEFG)
- Step 2:
  - P2 = (ABD)(CEFG),
- Step 3:
  - P3 = (ABD)(CEG)(F).
- Step 4:
  - P4 = (AD)(B)(CEG)(F)
- Step 5:
  - As P4 = P5, and thus we complete partition.

Procedures by example.

Present state	Next state $w = 0$	Next state $w = 1$	Output z
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

Present state	Nextstate w = 0	Nextstate w = 1	Output z
A	B	C	1
B	A	F	1
C	F	C	0
F	C	A	0

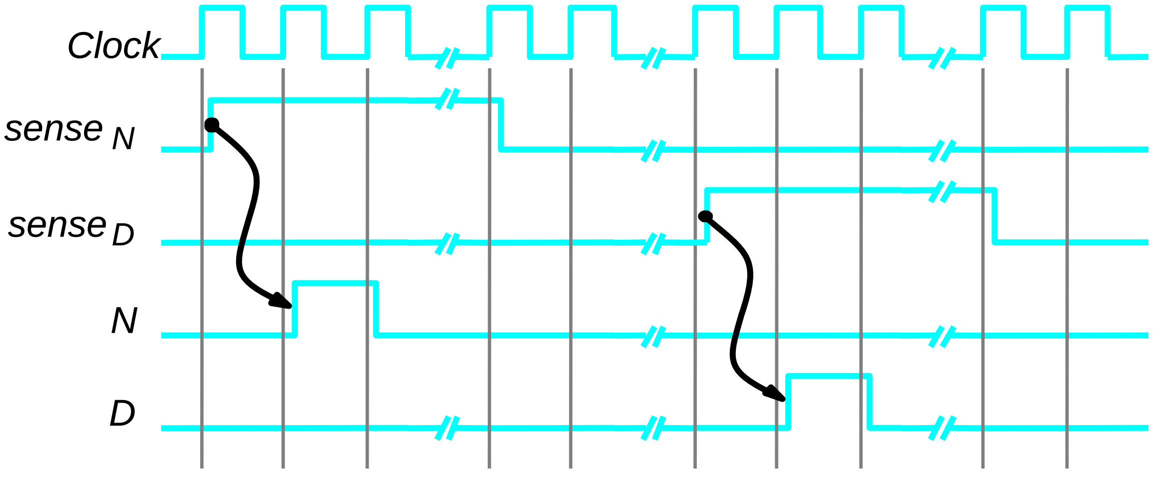
Figure 8.52. New Minimized state table for Example 8.5.

- Initial partition:
  - P1 = (ABCDEFG)
- Step 2:
  - P2 = (ABD)(CEFG),
- Step 3:
  - P3 = (ABD)(CEG)(F).
- Step 4:
  - P4 = (AD)(B)(CEG)(F)
  - P4 = P5, and thus we complete partition.

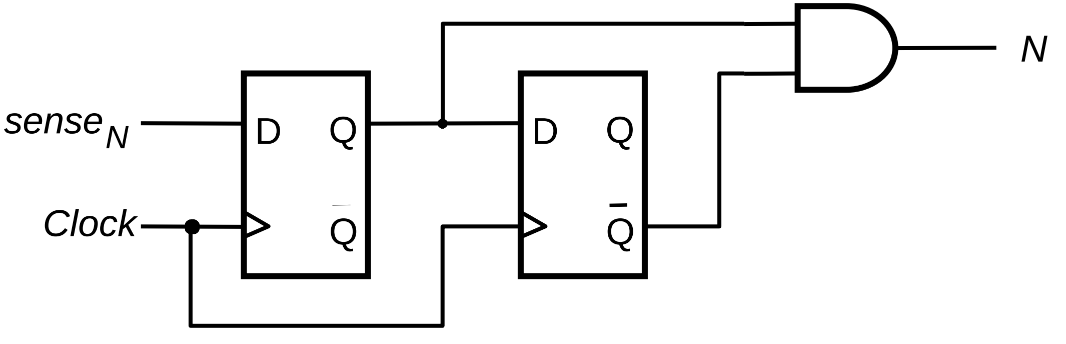
Example 8.5: Vending Machine

- Suppose that a coin-operated vending machine dispenses candy under the following conditions:
  - The machine accepts nickels and dimes
  - It takes 15 cents for a piece of candy to be released from the machine.
  - If 20 cents is deposited, the machine will not return the change, but it will credit the buyer with 5 cents and wait for the buyer to make a second purchase.

Signals for the vending machine



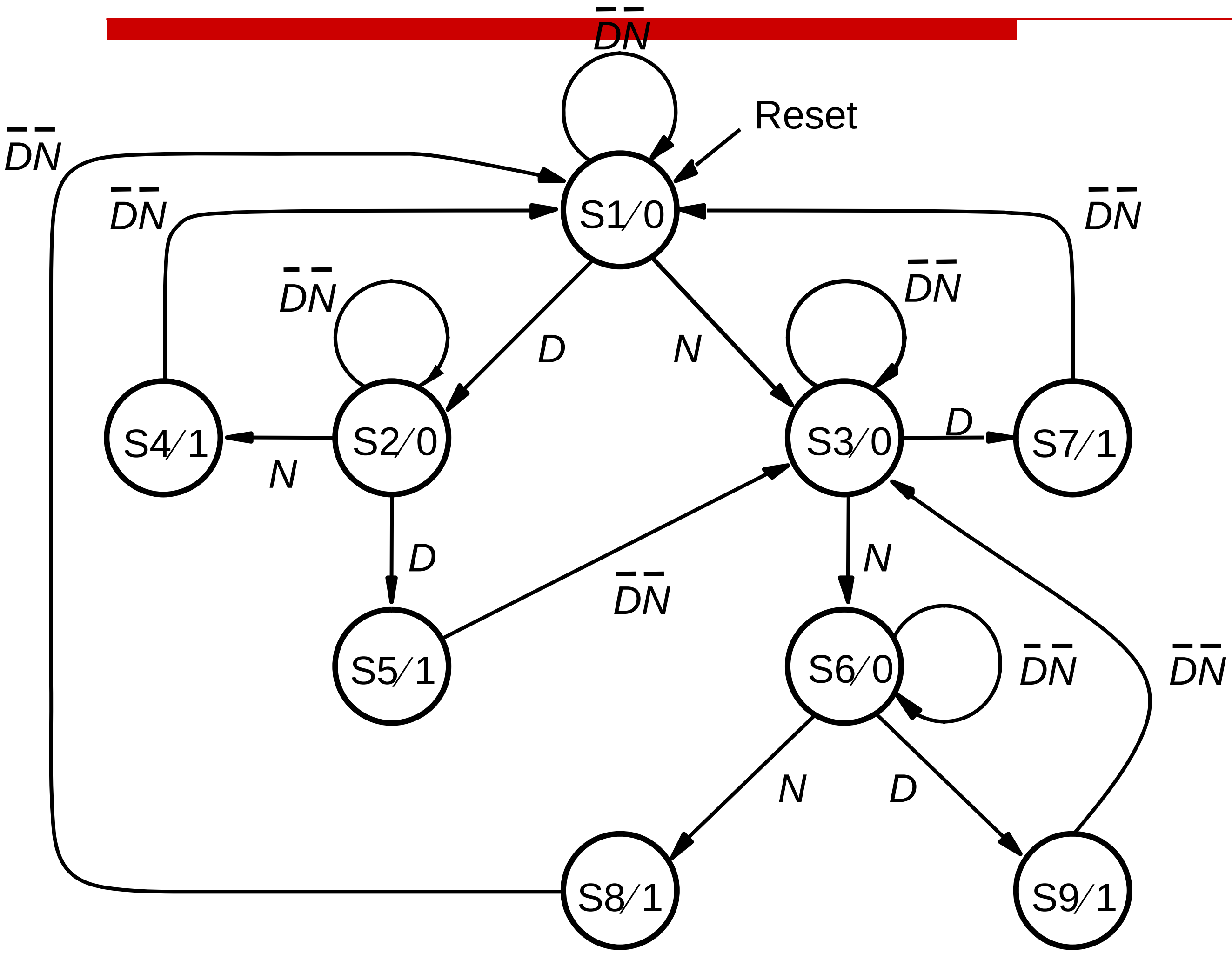
(a) Timing diagram



(b) Circuit that generates *N* (sensed a nickel)

Figure 8.53. Signals for the vending machine.

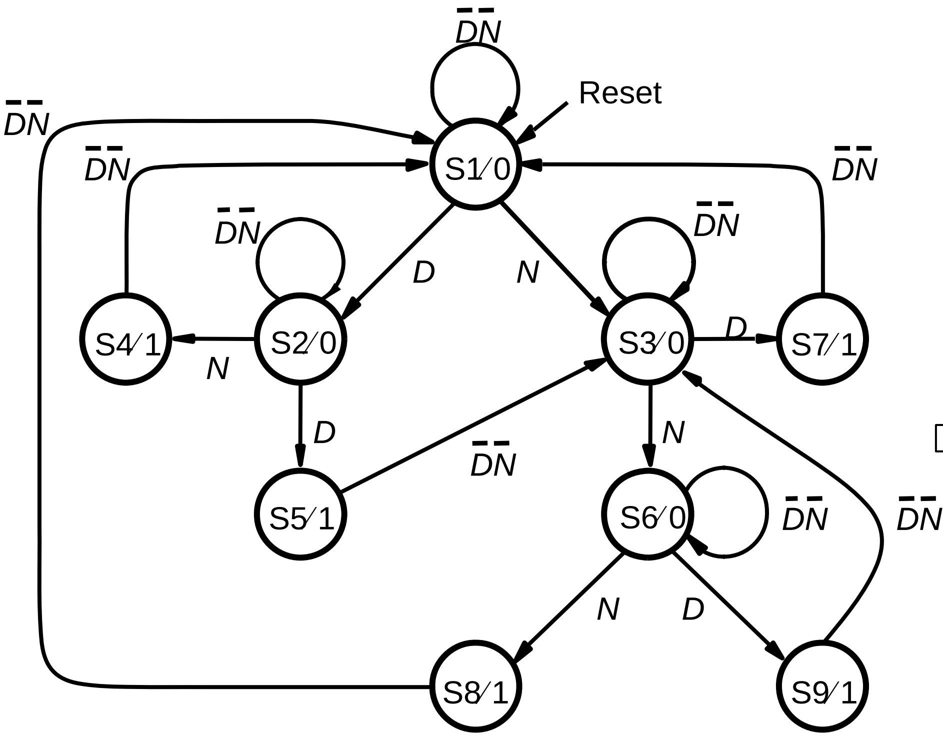
State diagram for the vending machine (Moore)



- S1: starting state, no money.
- S2: Insert dime.
- S3: Insert nickel.
- S4: S2 + nickel
- S5: S2 + dime
- S6: S3 + nickel
- S7: S3 + dime
- S8: S6 + nickel
- S9: S6 + dime

Figure 8.54. State diagram for Example 8.6. (9 states defined)

State table for the vending machine (Moore)



Present state	Next state				Output z
	DN =00	01	10	11	
S1	S1	S3	S2	–	0
S2	S2	S4	S5	–	0
S3	S3	S6	S7	–	0
S4	S1	–	–	–	1
S5	S3	–	–	–	1
S6	S6	S8	S9	–	0
S7	S1	–	–	–	1
S8	S1	–	–	–	1
S9	S3	–	–	–	1

Figure 8.55. State table for Example 8.6 (contains don't care conditions in state table which is called incompletely specified FSM)

Partition minimization for the vending machine (Moore)

Present state	Next state DN =00	01	10	11	Output z
S1	S1	S3	S2	–	0
S2	S2	S4	S5	–	0
S3	S3	S6	S7	–	0
S4	S1	–	–	–	1
S5	S3	–	–	–	1
S6	S6	S8	S9	–	0
S7	S1	–	–	–	1
S8	S1	–	–	–	1
S9	S3	–	–	–	1

- P1 = (S1, S2, S3, S4, S5, S6, S7, S8, S9)
- P2 = (S1, S2, S3, S6)(S4, S5, S7, S8, S9)
- P3 = (S1)(S3)(S2, S6)(S4, S5, S7, S8, S9)
- P4 = (S1)(S3)(S2, S6)(S4, S7, S8)(S5, S9)
- P5 = (S1)(S3)(S2, S6)(S4, S7, S8)(S5, S9)

Minimized state table for the vending machine (Moore)

P5 = (S1)(S3)(S2, S6)(S4, S7, S8)(S5, S9)

Present state	Next state *DN* =00	01	10	11	Output *z*
S1	S1	S3	S2	–	0
S2	S2	S4	S5	–	0
S3	S3	S6	S7	–	0
S4	S1	–	–	–	1
S5	S3	–	–	–	1
S6	S6	S8	S9	–	0
S7	S1	–	–	–	1
S8	S1	–	–	–	1
S9	S3	–	–	–	1

Present state	Next state *DN* =00	01	10	11	Output *z*
S1	S1	S3	S2	–	0
S2	S2	S4	S5	–	0
S3	S3	S2	S4	–	0
S4	S1	–	–	–	1
S5	S3	–	–	–	1

Figure 8.56. Minimized state table (5 states after minimization) for Example 8.6.

Minimized state diagram for the vending machine (Moore)

Present state	Next state				Output
	DN =00	01	10	11	*z*
S1	S1	S3	S2	–	0
S2	S2	S4	S5	–	0
S3	S3	S2	S4	–	0
S4	S1	–	–	–	1
S5	S3	–	–	–	1

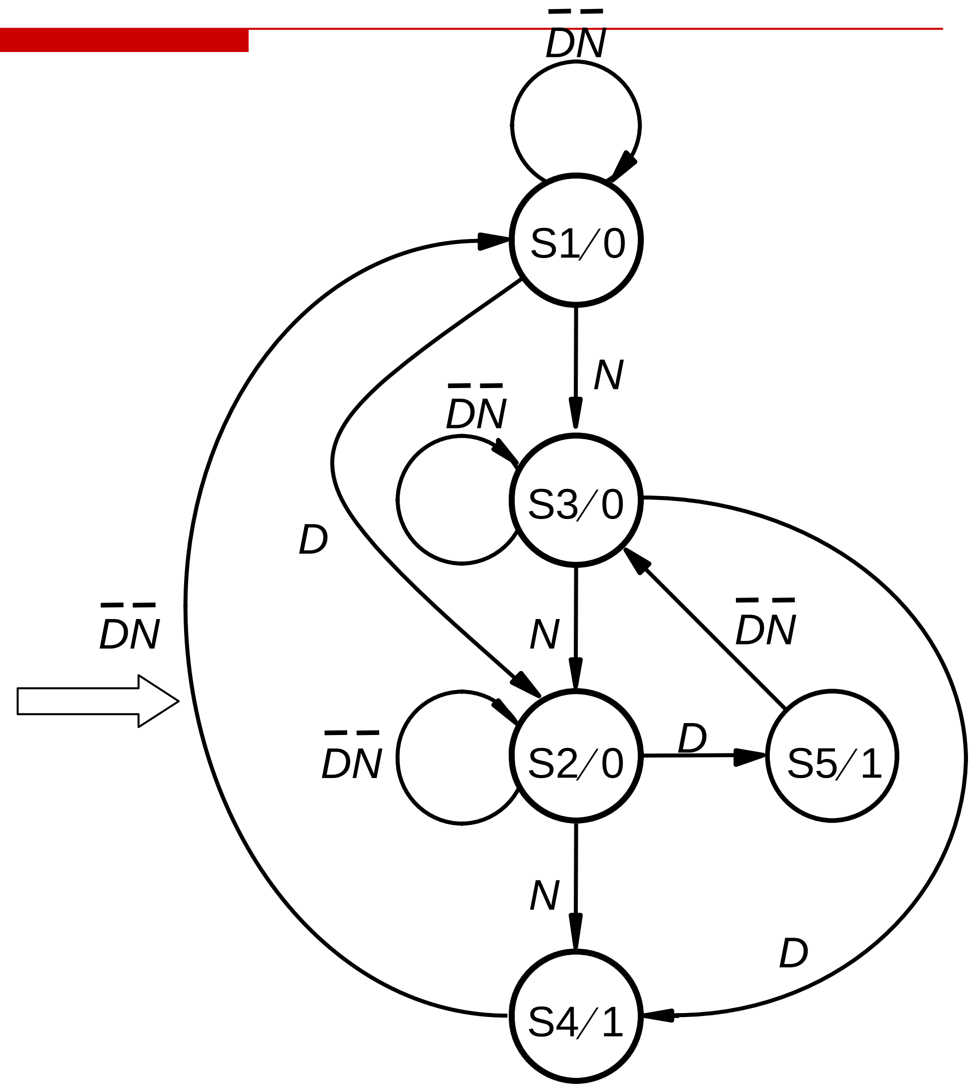


Figure 8.57. Minimized state diagram for Example 8.6.

Mealy-type FSM for the vending machine

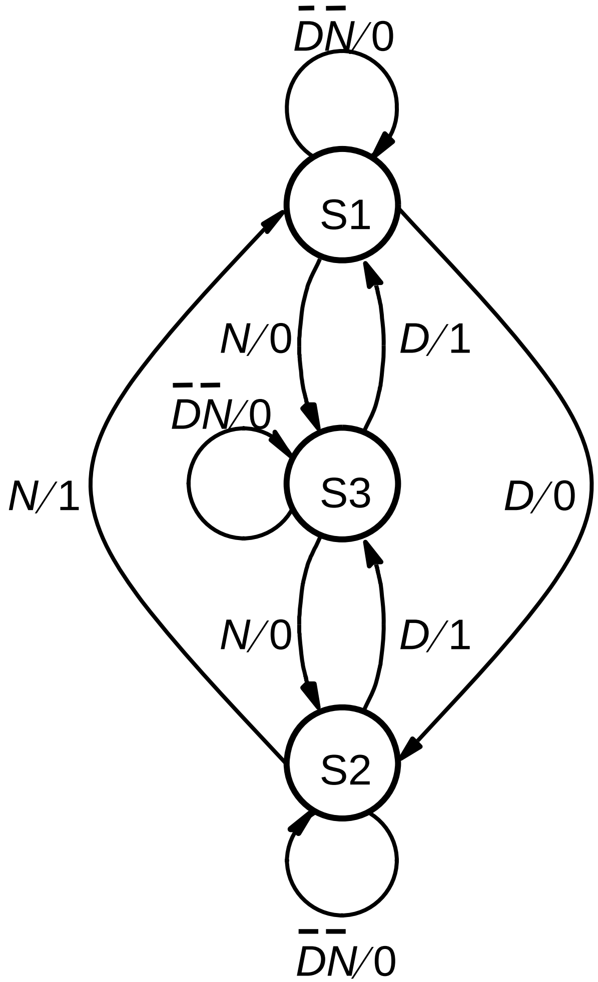


Figure 8.58. Mealy-type FSM for Example 8.6.

Incompletely specified state table

- State table include “don’t care” condition.
- “don’t care” for state transition (we can use the same partition strategy as previously)
- “don’t care” for output (we need to virtually specify output first)

Present state	Next state		Output z	
	$w = 0$	$w = 1$	$w = 0$	$w = 1$
A	B	C	0	0
B	D	–	0	–
C	F	E	0	1
D	B	G	0	0
E	F	C	0	1
F	E	D	0	1
G	F	–	0	–

Figure 8.59. Incompletely specified state table for Example 8.7.

Partition for Incompletely specified state table (Example 8.7)

Present state	Next state		Outputz	
	$w = 0$	$w = 1$	$w = 0$	$w = 1$
A	B	C	0	0
B	D	–	0	–
C	F	E	0	1
D	B	G	0	0
E	F	C	0	1
F	E	D	0	1
G	F	–	0	–

- Both unspecified outputs are assumed to be 0, then we have:
  - P1 = (ABCDEFG)
  - P2 = (ABDG)(CEF)
  - P3 = (AB)(D)(G)(CE)(F)
  - P4 = (A)(B)(D)(G)(CE)(F)
  - P5 = P4

  Leading to totally 6 states after minimization

Partition for Incompletely specified state table (Example 8.7)

Present state	Next state		Output z	
	$w = 0$	$w = 1$	$w = 0$	$w = 1$
A	B	C	0	0
B	D	–	0	–
C	F	E	0	1
D	B	G	0	0
E	F	C	0	1
F	E	D	0	1
G	F	–	0	–

- Both unspecified outputs are assumed to be 1, then we have:.
  - P1 = (ABCDEFG)
  - P2 = (AD)(BCEFG)
  - P3 = (AD)(B)(CEFG)
  - P4 = (AD)(B)(CEG)(F)
  - P5 = P4

  Leading to totally 4 states after minimization