Unit - V

Sequential Circuits

Finite State Machine:-

→ A sequential circuit is referred to as a finite state machine (FSM).

→ A finite state machine is an abstract model that describes the synchronous sequential machine.

→ In a sequential circuit the output depends on the present input as well as on the past inputs, i.e. on the past histories and since a machine might have an infinite varieties of possible histories, it would need an infinite capacity for storing them.

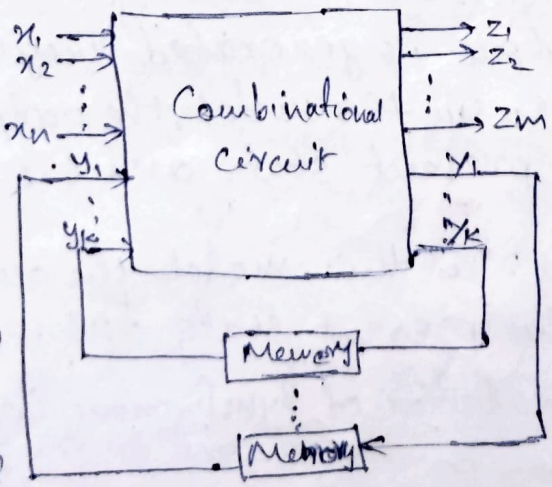
→ Since it is impossible to implement machines which have infinite ~~Capacity for~~ storage capabilities, we consider only finite state machines.

→ Finite state machines are sequential circuits whose past histories can affect their future behaviour in only a finite number of ways, i.e., they are machines with a fixed number of states.

→ These machines can distinguish among a finite number of classes of input histories.

→ These classes of input histories are referred to as the internal states of the machine.

→ Every finite state machine therefore contains a finite number of memory devices.



Block diagram of a finite state model

→ The block diagram of a finite state model x_1, x_2, . . x_n are inputs.

→ z_1, z_2 - - z_m are outputs

→ y_1, y_2 - - y_k are state variables, and Y_1, Y_2 - - - Y_k represent the next states.

Capabilities and Limitations of Finite State Machines

1) Periodic sequence of finite states: With n-state machine, we

can generate a periodic sequence of n states or smaller than n states.

→ For example, in a 6-state machine, we can have a maximum periodic sequence as 0, 1, 2, 3, 4, 5, 0, 1...

2) No infinite sequence: Consider an infinite sequence such that the output is 1 if and only if the number of inputs received so far is equal to $P(P+1)/2$ for P = 1, 2, 3... i.e., the desired input-output sequence has the following form:

Input :	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Output:	1	0	1	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	

→ Such an infinite sequence cannot be produced by a finite state machine.

3) Limited memory: The finite state machine has a limited memory and due to limited memory, it cannot produce certain outputs.

→ Consider a binary multiplier circuit for multiplying two arbitrarily large binary numbers.

→ If we implement this with a finite state machine capable of performing serial multiplication, we can find that it is not possible to multiply certain numbers.

→ Such a limitation does occur due to the limited memory available to the machine. This memory is not sufficient to store arbitrarily large partial products resulted during multiplication.

→ Finite state machines are of two types. They differ in the way the output is generated. They are:

1) Mealy type model: In this model, the output is a function of the present state and the present input

2) Moore type model: In this model, the output is a function of the present state only.

Mathematical Representation of Synchronous Sequential Machine

→ The relation between the present state S(t), present input x(t), and next state S(t+1) can be given as $S(t+1) = f\{S(t), x(t)\}$

→ The value of output $z(t)$ can be given as

$z(t) = g\{S(t), x(t)\}$ for Mealy model

$z(t) = g\{S(t)\}$ for Moore model

Comparision between the Moore machine and Mealy machine

Moore Machine	Mealy Machine
1) Its output is a function of present state only. $z(t) = g\{S(t)\}$	1) Its output is a function of present state as well as present input $z(t) = g\{S(t), x(t)\}$
2) Input changes do not affect the output.	2) Input changes may affect the output of the circuit
3) It requires more number of states for implementing same function	3) It requires less number of states for implementing same function

Distinguishable states and Distinguishing Sequences (Next page)

Minimization of Completely Specified Sequential Machines using Partition Technique

→ A procedure for the state minimization and determination of n equivalence is as follows:

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

Step 1: Partition the states into subsets such that all states in the same subset are 1-equivalent.

→ The first partition P_1 can be obtained by placing those states having the same outputs under all inputs, in the same block. The partitioning gives two subsets.

1) (A, C, E): Their outputs under 0 and 1 inputs are 0 and 0 respectively.

2) (B, D, F): Their outputs under 0 and 1 inputs are 1 and 0 respectively.

$$P_1 = (A, C, E)(B, D, F)$$

Step 2: Partition the states into subsets such that all states in the same subset are 2-equivalent.

→ Two states are 2-equivalent if and only if they are 1-equivalent and their T_i-successors of all possible T_i are also 1-equivalent.

→ In other words, two states are placed in the same block of partition P_2 if and only if they are in the same block of P_1 and for each possible I_i, their I_i-successors are also contained in the same block of P_1.

1) The 0-successors of (A,C,E) are (C,E,D). They are in different blocks of P_1. Therefore, block (A,C,E) must be split into (A,C) and (E).
2) The 1-successors of (B,D,F) are (F,E,B). They are in different blocks of P_1. Therefore, (B,D,F) must be split into (BF) and (D).
3) The 1-successors of (A,C,E) are (F,B,B). They are in same block of P_1. The 0-successors of (B,D,F) are (D,B,D). They are in the same block of P_1. So, no partitioning is possible.

$\therefore P_2 = (A,C)(E)(B,F)(D)$

Step3: Partition the states into subsets such that all states in the same block are 3-equivalent.

→ For this, consider the states which are 2-equivalent, i.e, blocks of P_2.

1) The 0-successors of (A,C) are (C,E). They are in differed blocks of P_2. So partition (A,C) into (A)(C).
2) The 1-successors of (A,C) are (F,B). They are in same block of P_2. And also the 0- and 1- successors of (B,F) are in the same block of P_2. So further partitioning of states is not possible.

$\therefore P_3 = (A)(C)(E)(B,F)(D)$

→ The states in the same blocks of P_3. So states B and F are equivalent

→ One of them is redundant and can be eliminated. So remove F and replace F by B in the other places in the table.

PS	NS, Z	
	X=0	X=1
A	C, 0	B, 0
B	D, 1	B, 0
C	E, 0	B, 0
D	B, 1	E, 0
E	D, 0	B, 0

Minimized state table

Distinguishable states and Distinguishing Sequences

→ Two states A and B of a sequential machine are distinguishable, if and only if there exists at least one finite input sequence which when applied to the sequential machine causes different output sequences depending on whether A or B is the initial state.

→ The sequence which distinguishes these states is called a distinguishing sequence of the pair (A, B)

→ Consider states A and B in this state table.

→ When input X=0, their outputs are 0 and 1 respectively and therefore, states A and B are called 1-distinguishable.

→ Cosider states A and E. The output sequence is as follows;

State Table

PS	NS, Z	
	X=0	X=1
A	C, 0	F, 0
B	D, 1	F, 0
C	E, 0	B, 0
D	B, 1	E, 0
E	D, 0	B, 0
F	D, 1	B, 0

$$X=0 \begin{cases} A \rightarrow C, 0 \text{ and } E \rightarrow D, 0 & \text{: outputs are the same} \\ C \rightarrow E, 0 \text{ and } D \rightarrow B, 1 & \text{: outputs are different} \end{cases}$$

→ Here the outputs are different after 2-state transitions and hence states A and E are 2-distinguishable.

→ Consider states A and C. The output sequence is as follows:

$$X=0 \begin{cases} A \rightarrow C, 0 \text{ and } C \rightarrow E, 0 & \text{: outputs are the same} \\ C \rightarrow E, 0 \text{ and } E \rightarrow D, 0 & \text{: outputs are the same} \\ E \rightarrow D, 0 \text{ and } D \rightarrow B, 1 & \text{: outputs are different.} \end{cases}$$

→ Here the outputs are different after 3 state-transitions and hence states A and C are 3-distinguishable.

→ In general, if two states have a distinguishable sequence of length k, the states are said to be k-distinguishable

→ The concept of k-distinguishability leads directly to the definition of k-equivalence.

→ States that are not k-distinguishable are said to be k-equivalent.

Simplification of Incompletely Specified Machines

→ In sequential circuits, if the state transitions or output

variables are not completely specified, they are called incompletely specified machines.

→ When the state transitions are not specified, the sequential machine is not predictable.

→ It is desirable to avoid such situations either by specifying the input sequence so that no next state is unspecified or by assigning next states that are not contrary to the desired results.

PS	NS, Z	
	X=0	X=1
S_1	S_3, 0	–
S_2	–, 1	S_3, 0
S_3	S_1, 0	S_2, 1

Incompletely Specified State Table

PS	NS, Z	
	X=0	X=1
S_1	S_3, 0	T, –
S_2	T, 1	S_3, 0
S_3	S_1, 0	S_2, 1
T	T, –	T, –

Reduced State Table

→ In reduced state table, state T is added to describe specified behaviour of the sequential machine in which all state transitions are specified and only the outputs are partially specified.

→ Unspecified outputs can be assigned without any effect on the state ~~machine~~ sequence. However, it is advantageous to leave outputs unspecified as long as possible during the state reduction process. This provides additional flexibility in the state reduction process.

→ When reducing incompletely specified state table, we use the term state compatibility instead of state equivalence.

→ The state compatibility is defined as;

"States S_i and S_j are said to be compatible states, if and only if for every input sequence that affects the two states, the same output sequence occurs whenever both outputs are specified and regardless of whether S_i or S_j is the initial state"

State Table

PS	NS, Z	
	X=0	X=1
S_0	S_1, 0	S_2, 0
S_1	S_0, –	S_2, 0
S_2	S_3, –	S_4, 1
S_3	S_1, –	S_2, –
S_4	S_2, –	S_4, 1

→ In this state table, compatible states are:

1) $S_0 = S_3$, if the output Z for S_3 when X=0 and X=1 is changed from a don't care to a 0.

2) $S_2 = S_4$, if the output Z for S_2 and S_4 when X=0 is changed from a don't care to either a 0 or a 1.

Partition Technique (Minimization of completely Specified Sequential Machine)

→ The equivalent partition is unique.

→ If two states, S_i and S_j of sequential machine, M are distinguishable, then they are distinguishable by a sequence of length n-1 or less, where n is the number of states in M.

Machine Equivalence: Two machines, M_1 and M_2 are said to be equivalent if and only if for every state in M_1, there is a corresponding equivalent state in M_2 and Viceversa.

Merger Chart Methods

Merger Graphs

→ The merger graph is a state reducing tool used to reduce states in the incompletely specified machine.

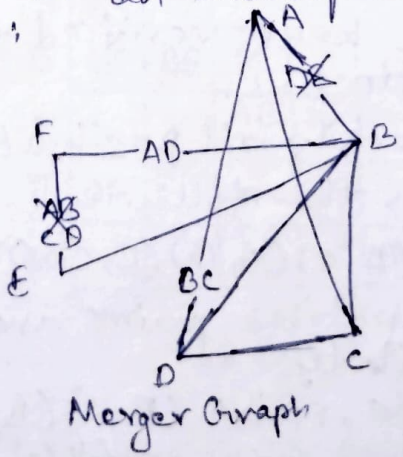
→ The merger graph is an undirected graph, defined as follows:

1) Each state in the state table is represented by a vertex in the merger graph. So it contains the same number of vertices as the state table contains states.
2) Each compatible state pair is indicated by an unbroken line drawn between the two state vertices.
3) Each potentially compatible state pair with non-conflicting outputs but with different next states is connected by a broken line. The implied states are written in the line break between the two potentially compatible states.
4) If two states are incompatible, no connecting line is drawn.

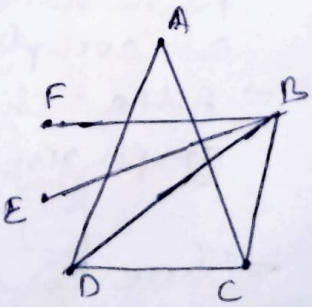
→ Consider a state table of an incompletely specified machine shown below:

PS	NS, Z			
	I_1	I_2	I_3	I_4
A	–	E, 1	B, 1	–
B	–	D, 1	–	F, 1
C	F, 1	–	–	–
D	–	–	C, 1	–
E	C, 0	–	A, 0	F, 1
F	D, 0	A, 1	B, 0	–

State Table



Merger Graph



Simplified Merger Graph

→ States A and B have non-conflicting outputs, but the successors (next states) under input I_2 are compatible only if implied states D and E are compatible. So draw a broken line from A to B with DE written in between.
→ States A and C are compatiable because the next states and output entries of states A and C are not conflicting. Therefore, a line is drawn between nodes A and C.
→ States A and D have non-conflicting outputs but the successors under I_3 are B and C. Hence join A and D by a broken line with BC entered in between.
→ States A and E are non-compatible and hence no line is drawn between A and E.
→ States A and F also have conflicting outputs under input I_3. So states A and F are non-compatible and hence no line is drawn between A and F.
→ In a similar way, the merger graph is drawn for all possible pairs of states.
→ A pair of states is compatible only if its implied pair is compatible. Therefore, it is necessary to check whether the implied pairs are also compatible.
→ If the implied states are incompatible, they are crossed and the corresponding line is ignored.
→ For example, implied states D and E are incompatible. So States A and B are also incompatible.
→ Next, it is necessary to check whether the incompatibility of A and B does not invalidate any other broken line.
→ On observing, the state E and F are also incompatible because the implied pair AB is incompatible.
→ The lines which remain in the graph after all the implied pairs have been verified to be compatible are regarded as complete lines.
→ After checking all possibilites of incompatibility, the merger graph gives the following seven compatible pairs:

(A, C) (A, D) (B, C) (B, D) (C, D) (B, E) (B, F)

→ These compatible pairs are further checked for compatibility.
→ For example, pairs (B, C) (B, D) (C, D) are compatible. So (B, C, D) is also compatible.
→ Also pairs (A, C) (A, D) (C, D) are compatible. So (A, C, D) is also compatible.

→ In this way the entire set of compatibles of sequential machine can be generated from its compatible pairs.
→ To find the minimal set of compatibles for state reduction, it is useful to find the maximum compatibles.
→ A set of compatible state pairs is said to be maximal, if it is not completely covered by any other set of compatible state pairs.
→ The maximum compatibles can be found by looking at the merger graph for polygons which are not contained within any higher order complete polygons.
→ For example in simplified merger graph only triangles (A, C, D) and (B, C, D) are of higher order.
→ The set of maximal compatibles for this sequential machine is given as:

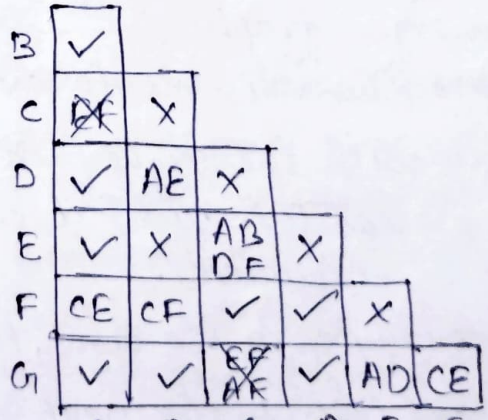
(A, C, D) (B, C, D) (B, E) (B, F)

Merger Table

→ The merger table method is also called Paull-Unger method or implication chart method.
→ This method is more convenient than the merger graph method to find the compatible pairs and their implications while performing the state reduction of machines having a large number of states.
→ Consider the state table for an incompletely specified sequential machine shown below:

PS	NS, Z			
	00	01	11	10
A	E, 0	–	–	–
B	–	F, 1	E, 1	A, 1
C	F, 0	–	A, 0	F, 1
D	–	–	A, 1	–
E	–	C, 0	B, 0	D, 1
F	C, 0	C, 1	–	–
G	E, 0	–	–	A, 1

State Table



Merger Table

→ Each cell of merger table corresponds to the compatible pair represented by the insertion of the row and column headings.

→ The incompatibility of two states is indicated by placing a cross mark X in the corresponding cell.
→ For example, states B and C are incompatible because their outputs are conflicting and hence the cell corresponding to them contains a cross mark X.
→ Similarly states B,E ; D,E ; E,F are incompatible, Hence put a x mark in the corresponding cells.
→ States A and B are compatible and hence the cell corresponding to them contains the check mark ✓.
→ Similarly, cells corresponding to states A,D ; A,E ; A,G ; B,G ; C,F ; D,F ; D,G are also compatible. So a check mark ✓ is put in those cells also.
→ The implied pairs of pairs corresponding to the state pair are written within the cell.
→ In a similar way, the entire merger table is written.
→ Now it is necessary to check whether the implied pairs are compatible or not by observing the merger table.
→ The implied states are incompatible if the corresponding cell contains a X.
→ For example, implied pair E,F is incompatible because cell EF contains a X.
→ Similarly, implied pairs EF,AF are incompatible because EF contains a X. So it is indicated by a X.
→ Once the merger table is completed, the set of all maximal compatibles can be formed by procedure which is the counterpart to that of finding a complete polygon in the merger graph.
→ The procedure is as follows:

1) Begin with the right most column in the merger table and proceed left until a column containing a compatible pair is encountered.
→ For example, in the table, the pair FG is the starting one.

2) Proceed left to the next column containing at least one compatible pair. If the state to which this column corresponds is incompatible with all states in the set of previously determined compatible states, then add this state to that set of compatible states to form a larger compatible. If this state is not compatible

states of previously determined set, but is compatible with some of them and/or with some other states, form a new set of compatible states.

→ For example, state E is not compatible with both F and G which are the previously determined set of states. However it is compatible with state G. Therefore, it is not added to the previous set (F,G) but a new set (E,G) is formed.

→ State D is compatible with states F and G of the previous set. So form a set (D, F, G) but D is not compatible with states E and G of the other previous set. So leave it as it is.

3) Repeat step 2 until the leftmost column is reached.

→ After application of the above mentioned procedure, the merger table gives the following set of maximal compatibles.

Column F: (F,G)

Column E: (F,G) (E,G)

Column D: (D,F,G) (E,G)

Column C: (C,E) (C,F) (E,G) (D,F,G)

Column B: (B,D,F,G) (C,E) (C,F) (E,G)

Column A: (A,B,D,F,G) (A,E) (C,E) (C,F) (C,G)

→ Therefore the set of maximal compatibles is (A,B,D,F,G) (A,E) (C,E) (C,F) (C,G).

Concept of Minimal Cover Table

→ Bothe the merger graph and merger table gives us the maximal compatible pairs and we can cover (implement) the sequential machine using the number of states equal to the number of maximal compatible pairs.

→ The compatibility graph can be used to get minimum closed covering giving further reduction of states if possible.

Compatibility Graph

→ The compatibility graph is a directed graph with the number of vertices equal to the number of compatible pairs.

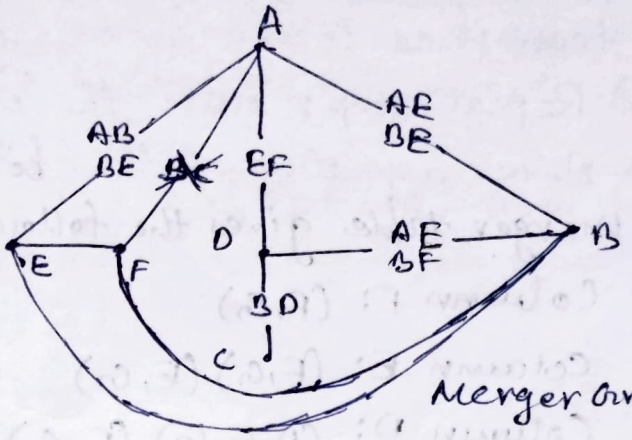
→ Each of its vertices correspond to one compatible pair.
→ A directed line (arc) leads from vertex (s_i, s_j) to vertex (s_p, s_q) if and only if (s_i, s_j) implies (s_p, s_q).
→ The compatibility graph can be easily drawn from the merger graph or merger table.

Steps to construct Compatibity graph from merger graph:

→ Consider a sequential machine specified by the state table below:

PS	NS, Z			
	00	01	10	11
A	A,0	-	E,-	B,1
B	E,-	C,1	B,-	-
C	-	B,0	-,1	D,0
D	A,0	~~B,0~~	F,1	B,-
E	B,0	-	B,0	-
F	-	C,1	-,0	C,1

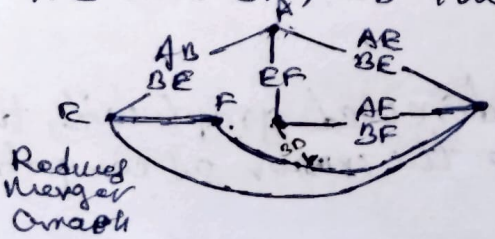
State Table

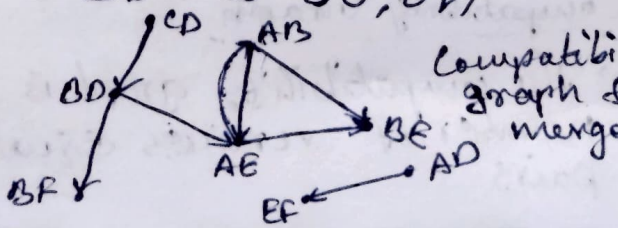


Merger Graph

Step 1: Identify and draw vertices: Mark the vertices corresponding to all compatible pairs. In merger graph, each compatible state pair is indicated by a line drawn between the two vertices.
→ For example in the reduced merger graph, there are lines between A and, A and D, A and E, B and D, B and E, B and F, C and D, and E and F. This gives eight vertices - AB, AD, AE, BD, BE, BF, CD and EF in the compatibility graph.

Step 2: Draw arcs: Draw the arcs that lead from vertex (s_i, s_j) to vertex (s_p, s_q) if and only if the compatible pair (s_i, s_j) implies the pair (s_p, s_q).
→ The compatible pair AB implies AE and BE, AD implies EF, AE implies AB and BE, BD implies AE and BF, CD implies BD and so, on.





Compatibility graph for merger graph

Steps to construct compatibility graph from merger table:

For the given state table, first draw the merger table.

Step 1: Identify and draw vertices: Mark the vertices corresponding to all compatible pairs in the merger table. In the merger table, each cell of the table except those marked x corresponds to the compatible pair defined by the intersection of the row and column headings.

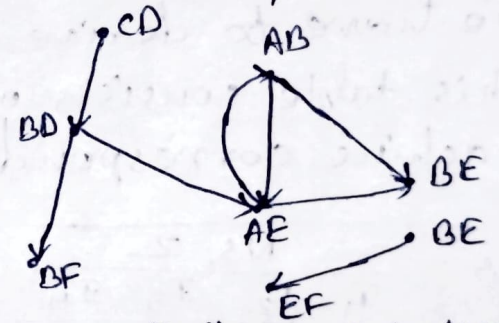
→ In the merger table from right to left and top to bottom the compatible pairs are EF, CD, BD, BE, BF, AB, AD and AE. This gives us eight vertices in the compatibility graph.

Step 2: Draw arcs: Draw the arcs that lead from vertex (S_i, S_j) to vertex (S_p, S_q) if and only if the compatible pair (S_i, S_j) implies (S_p, S_q)

→ For example, compatible pair CD implies BD, pair BD implies AE and BF, pair AB implies AE and BE and so on.

	A	B	C	D	E
B	AE BE				
C	x	x			
D	EF	AE BF	BD		
E	AB BE	✓	x	x	
F	~~BC~~	✓	x	x	✓

Merger Table



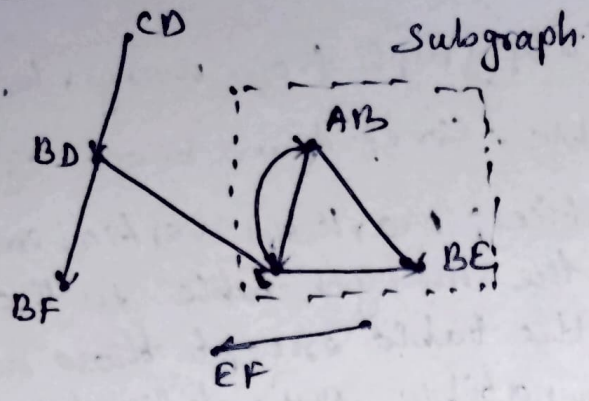
Compatibility Graph for Merger Table.

Subgraph of Compatibility Graph

→ Any part of a compatibility graph is called the subgraph of the compatibility graph.

→ A subgraph of a compatibility graph is said to be closed if for every vertex in the subgraph, all outgoing arcs and their terminating vertices also belong to the subgraph.

→ Each vertex in the subgraph belongs to one state. Such a subgraph forms a closed covering for the corresponding machine.



→ It includes three vertices: AB, AE, and BE. Therefore, the corresponding sequential machine can be covered by a three-state machine and this is a minimal closed covering.

→ The set of maximal compatibles derived from the merger graph contains four members {(A,B,D) (A,B,E) (B,E,F) (C,D)}. Therefore, the merger graph reduces the number of states required to implement the machine up to four.

→ The same machine can be implemented using only three states as indicated by the subgraph of the compatibility graph.

Minimal Cover Table

→ Once we obtain the minimal closed covering, then we have to derive the minimal cover table.

→ This table consists of states of the minimal state machine corresponding to the sequential machine.

PS	NS, Z			
	I_1	I_2	I_3	I_4
A	A,0	—	E,—	B,1
B	E,—	C,1	B,—	—
C	—	B,0	—,1	D,0
D	A,0	—	F,1	B,—
E	B,0	—	B,0	—
F	—	C,1	—,0	C,1

State Table

PS	NS, Z			
	I_1	I_2	I_3	I_4
(A,B)→X	Y,0	—	Z,—	Z/X,1
(A,E)→Y	X,0	—	Z,0	Z/X,1
(B,E)→Z	Z,0	—	Z/X,0	—

Minimal Cover Table

→ For the case under consideration there are three states and we assign them as x, Y and Z.

Therefore (A,B) → X ; (A,E) → Y ; (B,E) → Z

→ Using these assignments, we can write the minimal cover table.