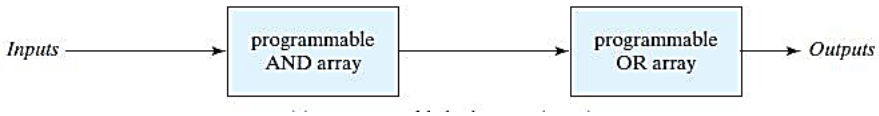
EE6301 – DIGITAL LOGIC CIRCUITS
UNIT – IV
ASYNCHRONOUS SEQUENTIAL CIRCUITSAND PROGRAMMABLE LOGIC DEVICES
PART – A

1. What is deadlock condition? [N/D'14]
   A condition resulting when one task is waiting to access a resource that another is holding, and vice versa.

2. Draw the block diagram of PLA. [N/D'14]



3. State the difference between static – 0 and static – 1 hazard.[A/M'15]
   Static-1 Hazard: the output is currently 1 and after the inputs change, the output momentarily changes
   to 0 before settling on 1
   Static-0 Hazard: the output is currently 0 and after the inputs change, the output momentarily changes to 1 before settling on 0

4. What is a PROM? [A/M'15] , [N/D'15]
   PROM- Programmable Read Only Memory is a device that contains Fixed AND and Programmable
   OR functions. IT contains fuses inact giving all 1`s in the stored bits and blown fuses by applying high voltage defining 0 states.

5. Compare pulsed mode and fundamental mode asynchronous circuit.[N/D'15]

S.No.	Pulsed mode asynchronous circuit	Fundamental mode asynchronous circuit
1.	Inputs are levels.	Inputs are pulses.
2.	Memory elements are either clocked flip – flops or time delay elements.	Memory elements are clocked flip – flops
3.	More difficult to dsign.	Easy to design.

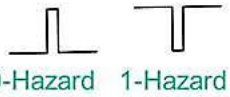
6. What are the two types of asynchronous sequential circuits? [M/J'16]
   The two types of asynchronous sequential circuits are:
   a) Fundamental mode circuits
   b) Pulse mode circuits

7. State the difference between PROM, PLA and PAL. [M/J'16], [A/M'17]

Types of PLD s	AND array	OR array
PROM	Fixed	Programmable
PLA	Programmable	Programmable
PAL	Programmable	Fixed

8. What is static hazard and dynamic hazard?[N/D'16]

- Static Hazard – Output value the same after input change



0-Hazard 1-Hazard

- Dynamic Hazard – Output value different after input change

9. Define races in asynchronous sequential circuits.[N/D'16]
When 2 or more binary state variables change their value in response to a change in an input variable, race condition occurs in an asynchronous sequential circuit. In case of unequal delays, a race condition may cause the state variables to change in an unpredictable manner.

10. What is a flow table? Give example. [A/M'17]
A state transition table with its internal state being symbolised with letters.
Examples:

y \ x	0	1
a	(a)	b
b	c	(b)
c	(c)	d
d	a	(d)

(a) Four states with one input

x_1x_2	00	01	11	10
a	(a), 0	(a), 0	(a), 0	b, 0
b	a, 0	a, 0	(b), 1	(b), 0

(b) Two states with two inputs and one output

PART – B

1. Explain the various types of hazards in sequential circuit design and the methods to eliminate them. Give suitable examples. (16) [N/D'14]

   Hazards in any system are obviously an un-desirable effect caused by either a deficiency in the system or external. Influences. In digital logic hazards are usually referred to in one of three ways:

   a) Static Hazards
   b) Dynamic Hazards
   c) Function Hazards

   These logic hazards are all subsets of the same problem: - When changes in the input variables do not change the output due to some form of delay caused by logic elements (NOT, AND, OR gates etc), this results in the logic not performing its function properly.

   This is however a temporary problem, and the logic will finally come to the desired function. Despite the logic arriving at the correct output, it is imeritive that hazards be eliminated as they can have an effect on other systems. Imagine hazards like this in a piece of hospital equipment.

   Static Hazards

   Definition:- *"When one input variable changes, the output changes momentarily when it shouldn't"*

   This particular type of hazard is usually due to a NOT gate within the logic. We can see the effects of the delay in the circuit from the following flash animation.

   The hazard can be dealt with in two ways:

   1. Insert another (additional) delay to the circuit. This then eliminates the static hazard.
   2. Eliminate the hazard by inserting more logic to counteract the effects (Note this makes assumptions that the logic will fail)

   - The first case is the most used of the two options. This is because it does not make assumptions about the logic, instead the method adds redundancy to overcome the hazard.
   - To solve the hazard we shall use our previous example and apply a theory that 'Huffman' discovered.
   - The insertion of a redundant loop can eliminate a static hazard.

   In the next example, it will also be evident that there will not be a situation where a static '0' occurs. A static '0' hazard is one which briefly goes to '1' when it should remain at '0'. A static '1' hazard is the reverse of this situation, i.e. the output should remain at '1' yet under some condition it briefly changes state to '0' (something we shall see in the following example)..

   Example of Static Hazards

   The Static '1' Hazard.

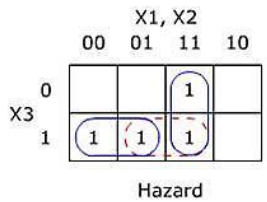
   Let us consider an imperfect circuit that suffers from a delay in the physical logic elements i.e. AND gates etc.

   The simple circuit performs the function:

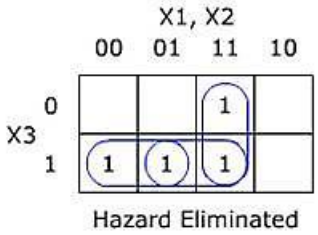
   f = X1.X2 + X1'.X3

   and the logic diagram can be shown as follows:

   At the starting diagram, it is clear that if no delays were to occur, then the circuit would function normally. However since this is not a perfect circuit, and an error occurs when the input changes from 111 to 011. i.e. When X1 changes state.



This Karnaugh Map shows the circuit. The two gates are shown by solid rings, and the hazard can be seen under the dashed ring. The theory proved by Huffman tells us that by adding a redundant loop 'X2X3' this will elimate the hazard. So the resulting logic is of the form shown in the next figure.

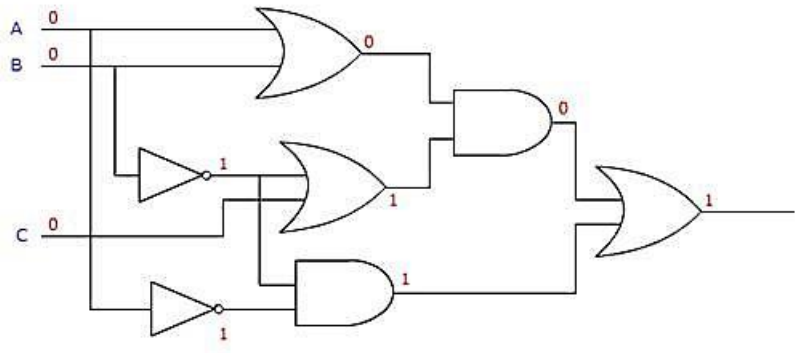


So our original function is now: f =X1.X2 + X1'.X3 + X2.X3
It is observed that even with imperfect logic elements, this example will not show signs of hazards when X1 changes state. This theory can be applied to any logic system.

Dynamic Hazards

- Definition:- *"A dynamic hazard is the possibility of an output changing more than once as a result of a single input change"*
- Dynamic hazards often occur in larger logic circuits where there are different routes to the output (from the input). If each route has a different delay, then it quickly becomes clear that there is the potential for changing output values that differ from the required / expected output. e.g. A logic circuit is meant to change output state from '1' to '0', but instead changes from '1' to '0' then '1' and finally rests at the correct value '0'. This is a dynamic hazard.
- Dynamic hazards take a more complex method to resolve. The below example shows how a dynamic hazard can occur but now how to solve it.

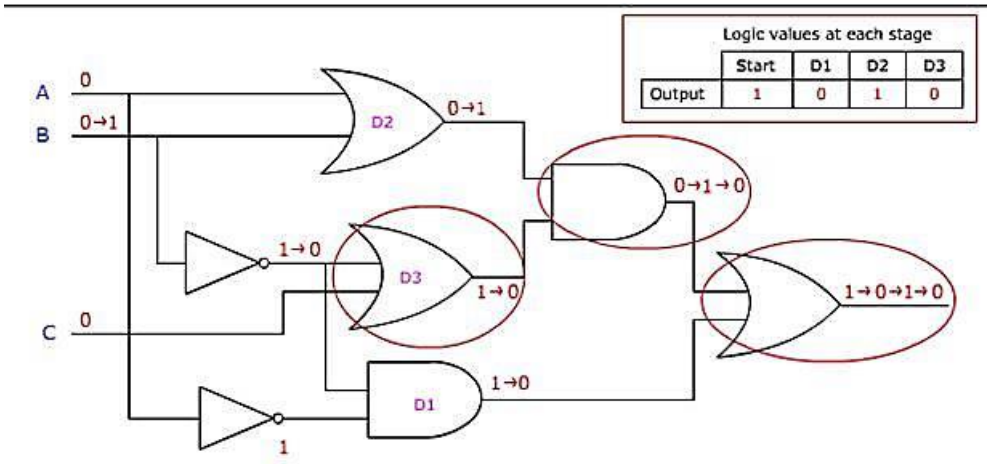
Let us take the circuit above, and see the proper logic output with the logic values above



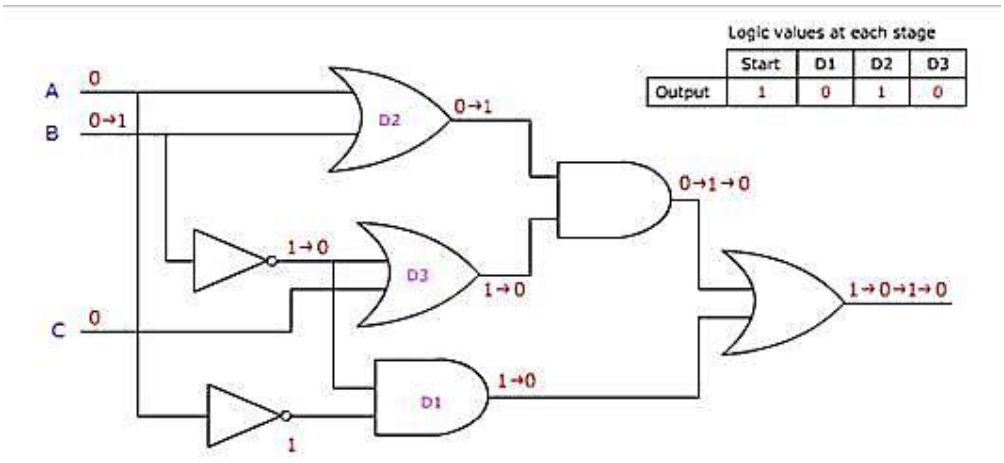
Let artificial delays be introduced in some of the elements. Marked D1, D2 and D3.

- Consider D1 < D2 < D3.
  i.e. The delay in D1 is less than the delay in D2, and the delay in D3 is greater than the delay in D2.

- Say that input B changes from 0 to 1. Working with one delay at a time the output can be determined.
- There could be a different output value for every different delay.
- The NOT gate shown changes from 1 to 0. Remembering that D1 is the shortest of the three delays, the AND gate shown will also change from 1 to 0.
- Because the other delays (D2, D3) are longer than D1, and because the other gates have no delay (or neglible delay), our output changes from 1 to 0. (The first effect of the hazard).
- The next delay to occur is D2. So the OR gate shown implements the change (0 OR 1 = 1). Now the AND gate has only seen the change in one of its inputs due to the delay D3 being longer than the other delays. So momentarily, both inputs are logic 1 which means the AND gates output goes to logic 1 hence changing the output of the entire circuit (1 OR 0 = 1).



- The next delay to occur is D3. The OR gate reacts slower than D1 to the change in input, but now the OR gate implements the change (0 OR 0 = 0). There is a knock on effect to the AND gate (as 0 AND 1 = 0) and then again to the output of the entire circuit. The output rests at 0.
- This is the final stage as there are no more delays in the circuit and therefore no more changes of state due to input B changing.
  D3(1 OR 0 = 1).
- The circuit finally rests at the correct logical value for an ideal circuit. However the output has changed values twice before coming to the correct result on the third change. If this was an ideal circuit then the output should have changed only once.



2. Describe with reasons the effect of races in asynchronous sequential circuit design. Explain its type with illustrations. Show the method of race – free state assignments with examples. (16) [N/D'14]
   - A race condition or race hazard is the behaviour of an electronic, software, or other system where the output is dependent on the sequence or timing of other uncontrollable events.
   - A race condition may occur in a system of logic gates where inputs vary. If a given output depends on the state of the inputs it may only be defined for steady-state signals. As the inputs change state a small delay will occur before the output changes due to the physical nature of the electronic system. The output may, for a brief period, change to an unwanted state before settling back to the designed state. Certain systems can tolerate such glitches but if this output functions as a clock signal for further systems that contain memory, for example, the system can rapidly depart from its designed behaviour..

Types:

❖ **Critical and non-critical forms**

a) A *critical race condition* occurs when the order in which internal variables are changed determines the eventual state that the state machine will end up in.

b) A *non-critical race condition* occurs when the order in which internal variables are changed does not determine the eventual state that the state machine will end up in.

❖ **Static, dynamic, and essential forms**

a) A *static race condition* occurs when a signal and its complement are combined together.

b) A *dynamic race condition* occurs when it results in multiple transitions when only one is intended. They are due to interaction between gates. It can be eliminated by using no more than two levels of gating.

c) An *essential race condition* occurs when an input has two transitions in less than the total feedback propagation time. Sometimes they are cured using inductive delay line elements to effectively increase the time duration of an input signal.

Design techniques such as Karnaugh maps encourage designers to recognize and eliminate race conditions before they cause problems. Often logic redundancy can be added to eliminate some kinds of races.

RACE -FREE STATE ASSIGNMENT

- Once a reduced flow table has been derived for an asynchronous sequential circuit, the next step in the design is to assign binary variables to each stable state. This assignment results in the transformation of the flow table into its equivalent transition table.
- The primary objective in choosing a proper binary state assignment is the prevention of critical races. Critical races can be avoided by making a binary state assignment in such a way that only one variable changes at any given time when a state transition occurs in the flow table.

Three-Row Flow-Table Example

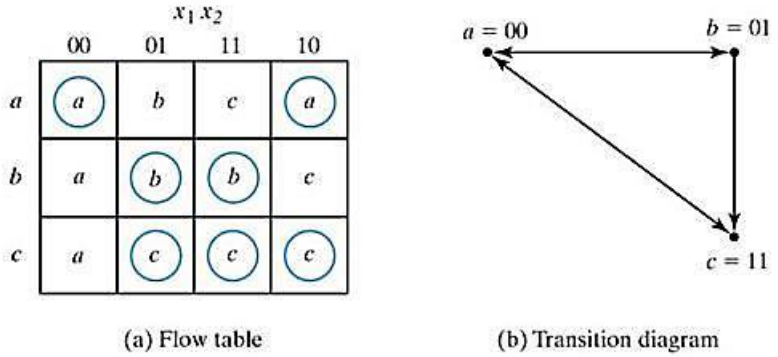


Fig: Three row flow table example

- To avoid critical races, we must find a binary state assignment such that only one binary variable changes during each state transition.
- An attempt to find such an assignment is shown in the transition diagram.
- State a is assigned binary 00, and state c is assigned binary 11.
- This assignment will ca use a critical race during the transition from a to c because there are two changes in the binary state variables and the transition from a to c may occur directly or pass through b.
- Note that the transition from c to a also ca uses a race condition, but it is noncritical because the transition does not pass through other states.
- A race-free assignment can be obtained if we add an extra row to the flow table. The use of a fourth row does not increase the number of binary state variables, but it allows the formation of cycles between two stable states.
- The transition table corresponding to the flow table with the indicated binary state assignment is shown in Fig. The two dashes in row d represent unspecified states that can be considered don't-care conditions. However, care must be taken not to assign 10 to these squares, in order to avoid the possibility of an unwanted stable state being established in the fourth row.

Four-Row Flow-Table Example

A flow table with four rows requires a minimum of two state variables. Although a race-free assignment is sometimes possible with only two binary state variables, in many cases the requirement of extra rows to avoid critical races will dictate the use of three binary state variables

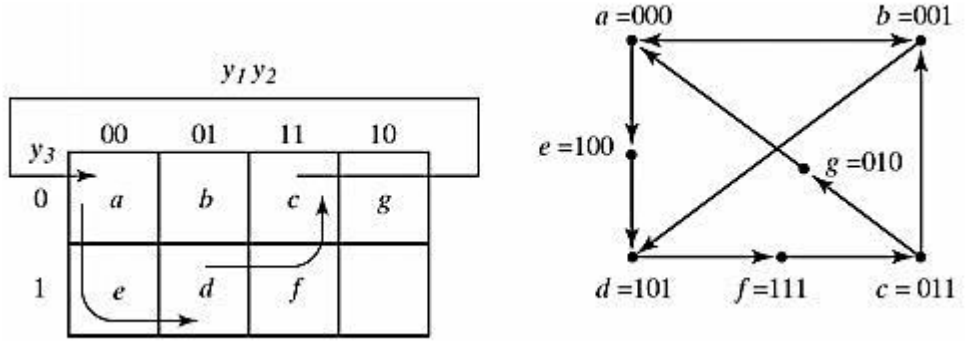


Fig: Four-row flow-table example

- The following figure shows a state assignment map that is suitable for any four-row flow table. States a, b, c and d are the original states and e, f and g are extra states.
- The transition from a to d must be directed through the extra state e to produce a cycle so that only one binary variable changes at a time.
- Similarly, the transition from c to a is directed through g and the transition from d to c goes through f. By using the assignment given by the map, the four-row table can be expanded to a seven-row table that is free of critical races.

	00	01	11	10
000 = *a*	*b*	(*a*)	*e*	(*a*)
001 = *b*	(*b*)	*d*	(*b*)	*a*
011 = *c*	(*c*)	*g*	*b*	(*c*)
010 = *g*	–	*a*	–	–
110 –	–	–	–	–
111 = *f*	*c*	–	–	*c*
101 = *d*	*f*	(*d*)	(*d*)	*f*
100 = *e*	–	–	*d*	–

Fig: State assignment to modified flow table

Note that although the flow table has seven rows there are only four stable states. The uncircled states in the three extra rows are there merely to provide a race-free transition between the stable states.

Multiple-Row Method

- The method for making race-free stale assignments by adding extra rows in the flow table is referred to as the shared-row method.
- A second method called the multiple-row method is not as efficient, but is easier to apply.
- In multiple- row assignment each state in the original row table is replaced by two or more combinations of state variables.

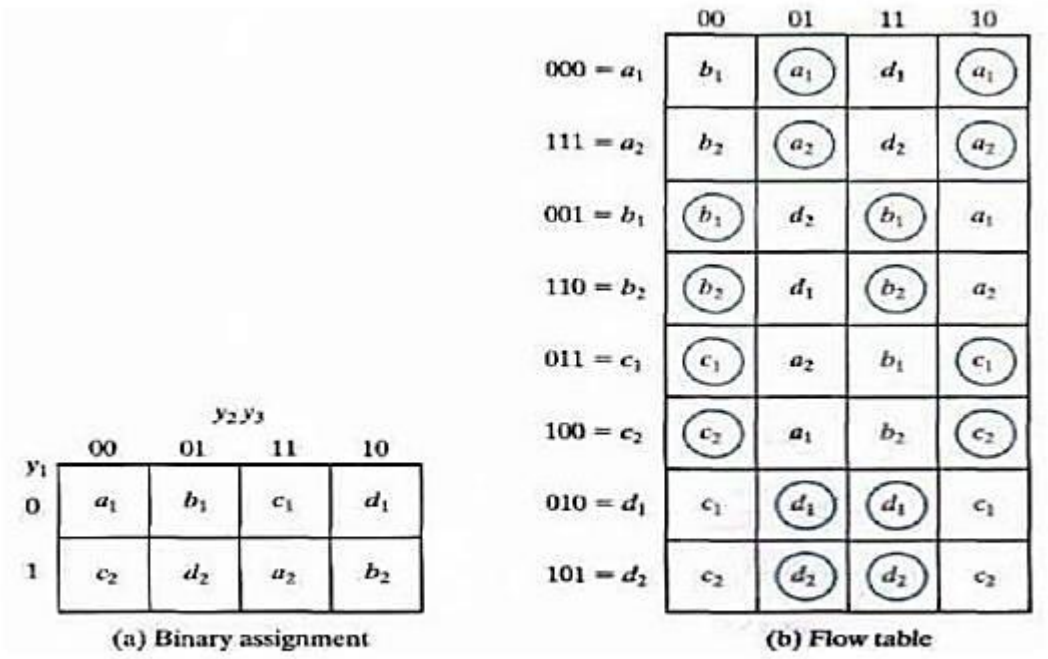


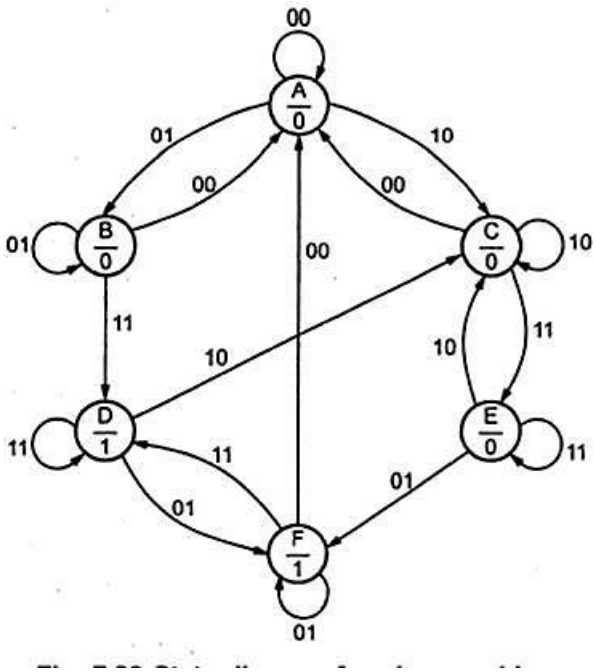
Fig: Multiple row assignment

Fig: Multiple row assignment

- There are two binary state variables for each stable state, each variable being the logical complement of the other. For example, the original state a is replaced with two equivalent states a1 =000 and a2 = 111.
- The output values, not shown here must be the same in a1 and a2. Note that a1 is adjacent to b1, c2 and d1, and a2 is adjacent to c1, b2 and d2, and similarly each state is adjacent to three states with different letter designations.
- The expanded table is formed by replacing each row of the original table with two rows. In the multiple-row assignment, the change from one stable state 10 another will always cause a change of only one binary state variable.
- Each stable stale has two binary assignments with exactly the same output.

3. Design an asynchronous sequential circuit that has two inputs X_2 and X_1 and one output Z. When $X_1 = 0$, the output Z is 0.The first change in X_2 that occurs while X_1 is 1 will cause output Z to be 1. The output Z will remain 1 until X_1 returns to 0.(16) [A/M'15]

State diagram



Primitive flow table constructed from state diagram

Present state	Next state, Output Z for X_2X_1 Inputs			
	00	01	11	10
A	(A), 0	B, –	–, –	C, –
B	A, –	(B), 0	D, –	–, –
C	A, –	–, –	E, –	(C), 0
D	–, –	F, –	(D), 1	C, –
E	–, –	F, –	(E), 0	C, –
F	A, –	(F), 1	D, –	–, –

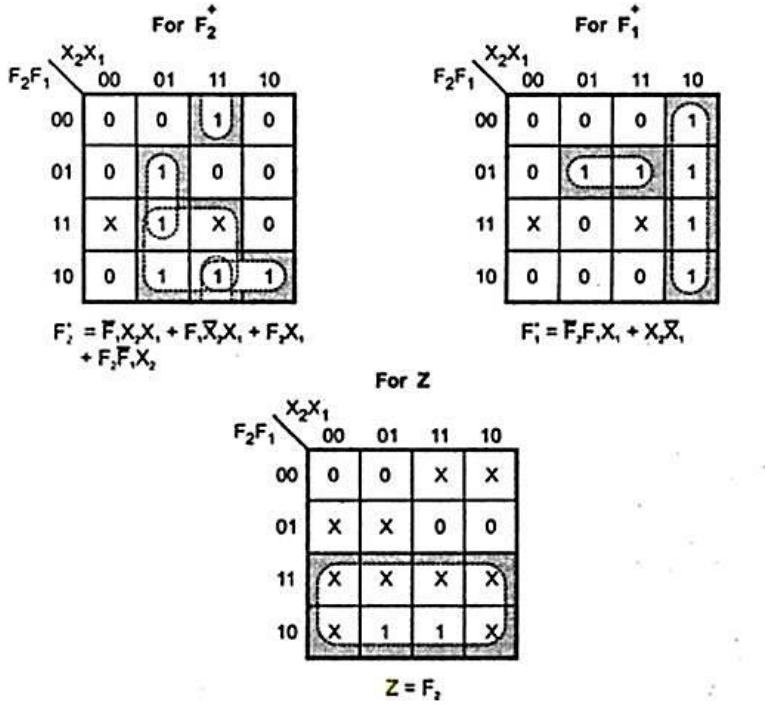
Flow table with state assignment

Present state F_2 F_1	Next state, Output Z for X_2X_1 Inputs			
	00	01	11	10
S_0 → 0 0	(S_0), 0	(S_0), 0	S_2, –	S_1, –
S_1 → 0 1	S_0, –	S_3, –	(S_1), 0	(S_1), 0
S_2 → 1 0	S_0, –	(S_2), 1	(S_2), 1	S_3, –
S_3 → 1 1	–, –	S_2, –	–, –	S_1, –

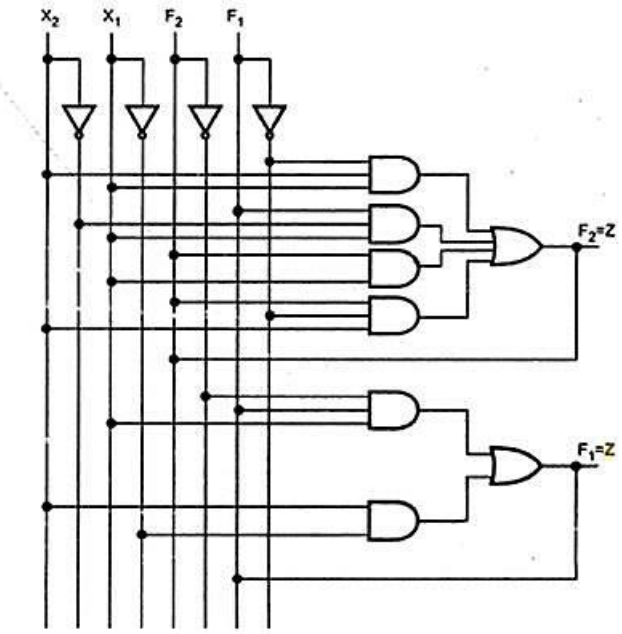
Flow table converted into a transition table

Present state		Next state, Output Z for X_2X_1 Inputs			
F_2	F_1	00	01	11	10
0	0	00 , 0	00 , 0	10 , –	01 , –
0	1	00 , –	11 , –	01 , 0	01 , 0
1	0	00 , –	10 , 1	10 , 1	11 , –
1	1	– , –	10 , –	– , –	01 , –

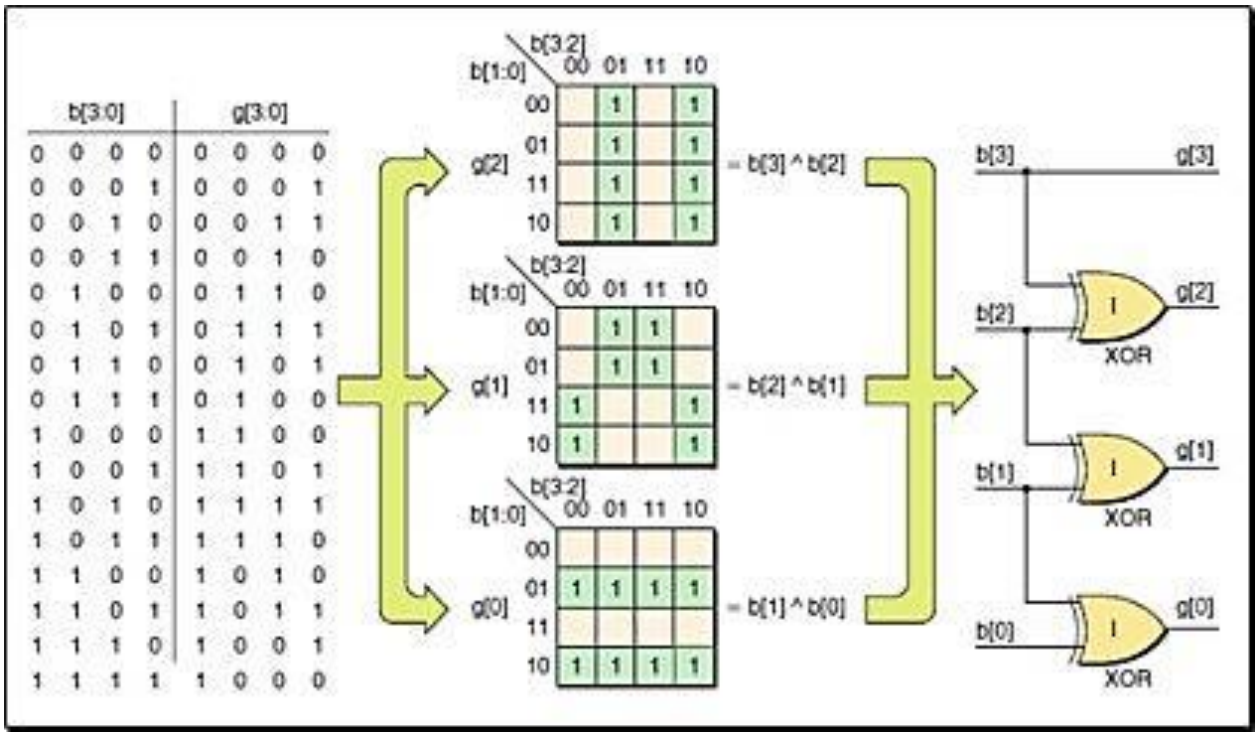
K-map simplification



Logic diagram



4. Show how to program the fusible links to get a 4 bit Gray code from the binary inputs using PLA and PAL and compare the design requirements with PROM. (16) [N/D'15]
   Gray code generator using PROM



PROM based binary to gray code converter

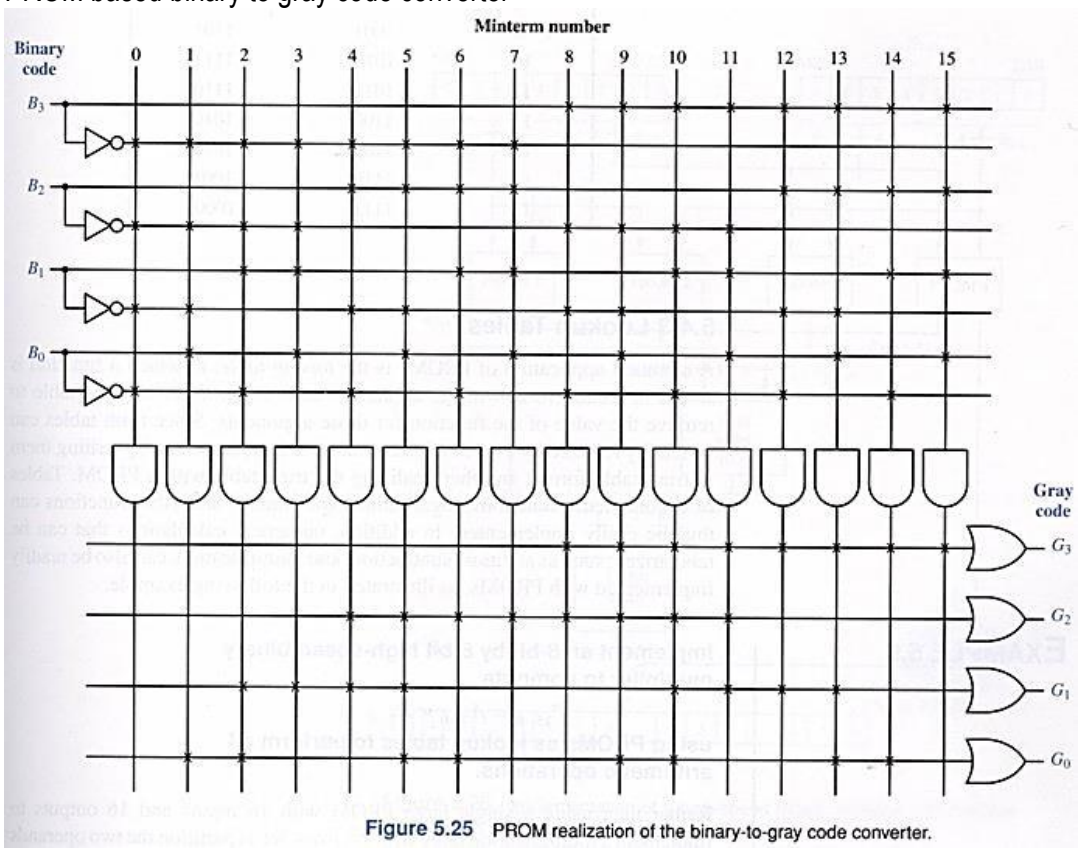
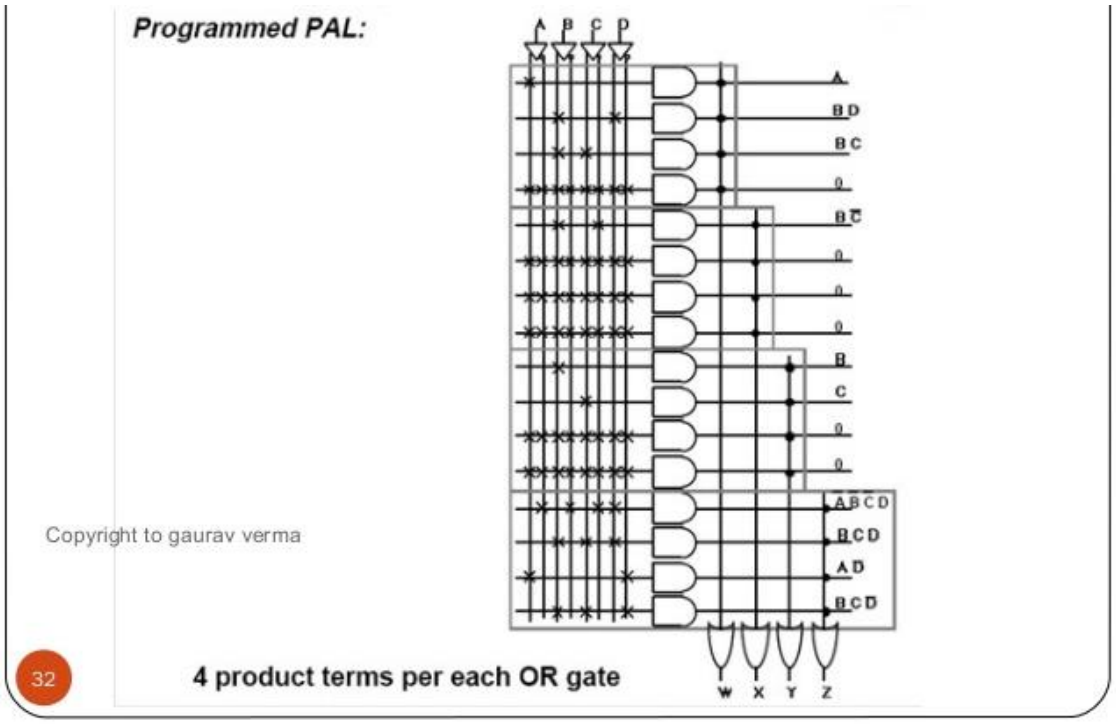


Figure 5.25 PROM realization of the binary-to-gray code converter.

PAL based binary to gray code converter



Binary to gray code converter Boolean expression are concluded as

$$Y_3 = A$$

$Y_2 = A\bar{B} + \bar{A}B$

$Y_1 = B\bar{C} + \bar{B}C$

$Y_0 = C\bar{D} + \bar{C}D$

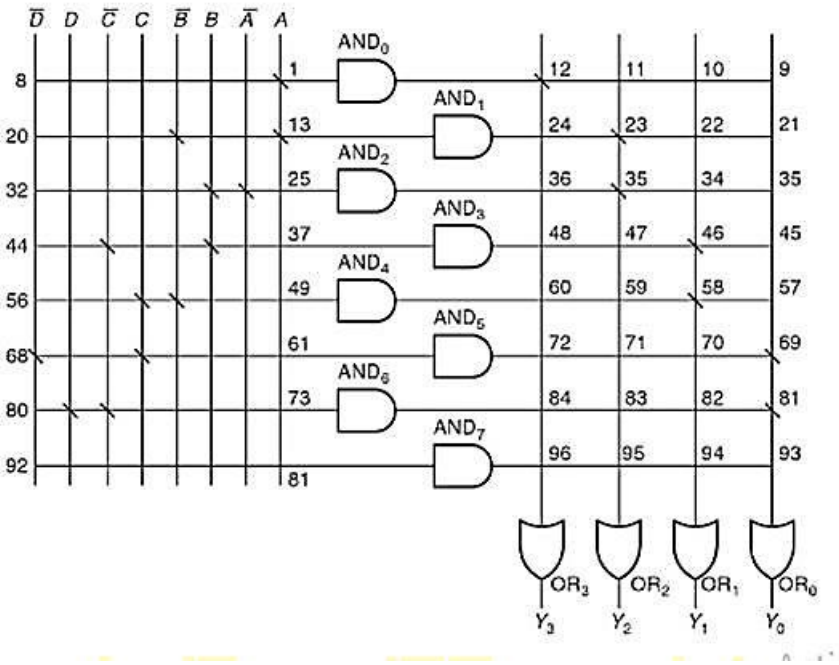
It is noted that 4 x 7 x 4 PLA is needed to implement gray code converter and same for PAL also.

Total number of AND OR gates links are

96 – PLA

56 – PAL

64 – PROM



5. What are static – 0 and static – 1 hazard? Explain the removal of hazards using hazard covers in k-map. (8) [M/J'16]

Static Hazards

A static hazard occurs when a single input variable change should cause no change in the output of a combinational logic circuit, but a short glitch of the incorrect logic level occurs.

The problem occurs because real physical implementations of logic functions have finite propagation times which are variable, and if two inputs to a gate should theoretically change simultaneously, one will actually change before the other.

If more than one input variable changes "simultaneously" there is no way to guarantee that such glitches will not occur.

Types of Static Hazards

Static – 1 hazard :- A static 1 hazard may occur in a two level sum of products (SOP) implementation.

Static – 0 hazard :- A static 0 hazard may occur in a two level product of sums (POS)implementation.

Example of Static Hazards

The Static '1' Hazard.

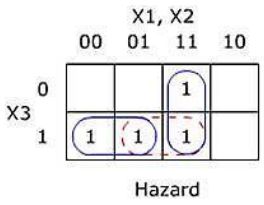
Let us consider an imperfect circuit that suffers from a delay in the physical logic elements i.e. AND gates etc.

The simple circuit performs the function:

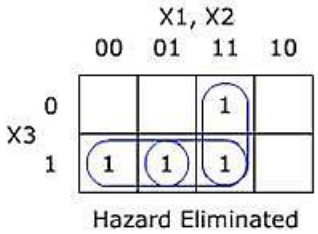
f = X1.X2 + X1'.X3

and the logic diagram can be shown as follows:

At the starting diagram, it is clear that if no delays were to occur, then the circuit would function normally. However since this is not a perfect circuit, and an error occurs when the input changes from 111 to 011. i.e. When X1 changes state.



This Karnaugh Map shows the circuit. The two gates are shown by solid rings, and the hazard can be seen under the dashed ring. The theory proved by Huffman tells us that by adding a redundant loop 'X2X3' this will elimate the hazard. So the resulting logic is of the form shown in the next figure.



So our original function is now: f =X1.X2 + X1'.X3 + X2.X3

It is observed that even with imperfect logic elements, this example will not show signs of hazards when X1 changes state. This theory can be applied to any logic system.

6. Explain cycles and races in asynchronous sequential circuits. (8) [M/J'16]

Races in asynchronous sequential circuits

- Race condition: two or more binary state variables will change value when one input variable changes Cannot predict state sequence if unequal delay is encountered
- Non-critical race: The final stable state does not depend on the change order of state variables Critical race: The change order of state variables will result in different stable states.

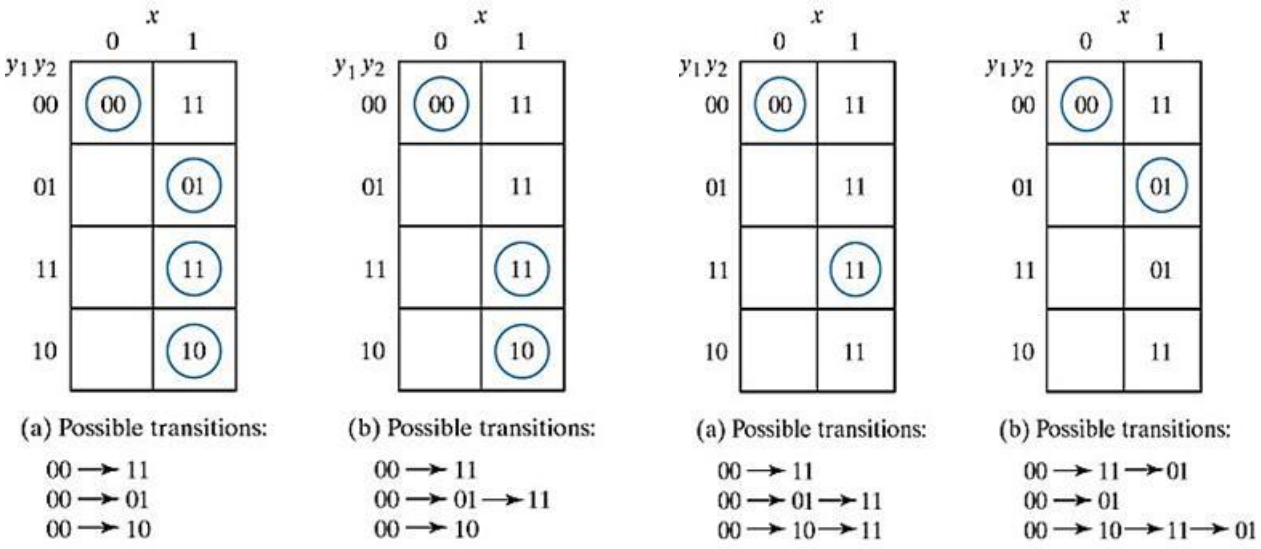


Fig. 9-7 Examples of Critical Races

Fig. 9-6 Examples of Noncritical Races

Cycles in asynchronous sequential circuits

- A cycle occurs when an asynchronous circuit makes a transition through a series of unstable states.
- When a state assignment is made so that it introduces cycles.
- Care must be taken so that each cycle terminates on a stable state.
- If a cycle does not contain a stable state, the circuit will go from one unstable state to another, until the inputs are changed.

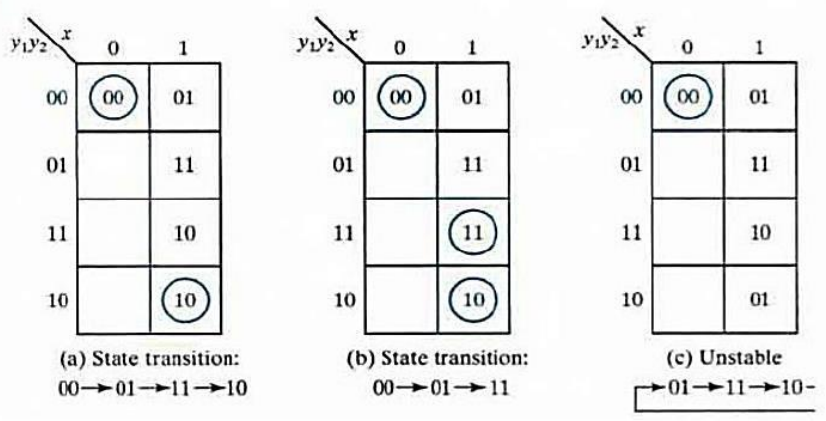


Fig: Examples of cycles