EC 3352 – DIGITAL SYSTEM DESIGN

UNIT – IV : ASYNCHRONOUS SEQUENTIAL CIRCUITS

4.5 ANALYSIS EXAMPLE FUNDAMENTAL MODE

Asynchronous sequential circuits can be constructed with the use of SR latches with or without external feedback paths. Of course, there is always a feedback loop within the latch itself. The analysis of a circuit with latches will be demonstrated by means of a specific example from which it will be possible to generalize the procedural steps necessary to analyze other, similar circuits.

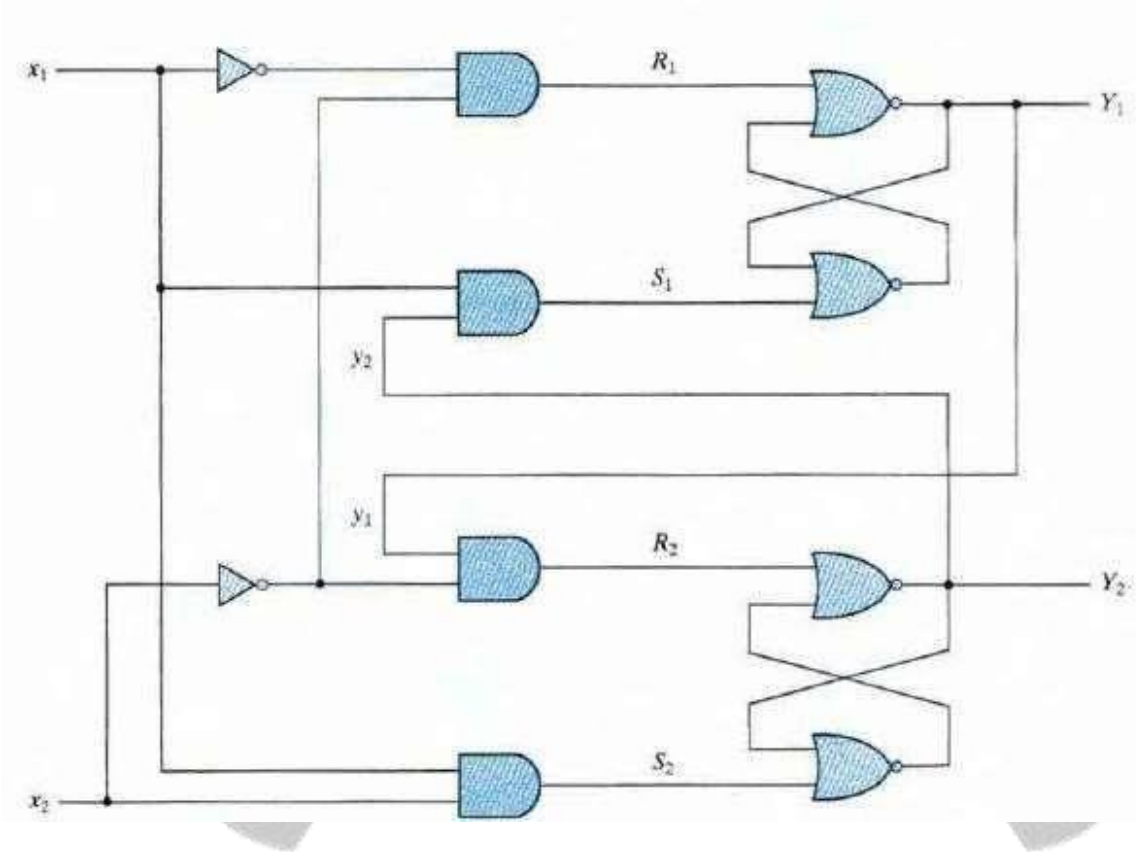


Fig: 4.8 - Examples of a circuit with SR latches

Image source from Digital Design by Moris Mano (Page No. 428)

There are two inputs, x_1 and x_2, and two external feedback loops giving rise to the secondary variables, y_1 and y_2. Note that this circuit resembles a conventional sequential circuit with latches behaving like flip-flops without clock pulses. The analysis of the circuit requires that we first obtain the Boolean function for the S and R inputs in each latch:

$$S_1 = x_1 y_2 \qquad S_2 = x_1 x_2$$
$$R_1 = x_1' x_2' \qquad R_2 = x_2' y_1$$

We then check whether the condition SR = 0 is satisfied to ensure proper operation ofthe circuit:

$$S_1 R_1 = x_1 y_2 x_1' x_2' = 0$$
$$S_2 R_2 = x_1 x_2 x_2' y_1 = 0$$

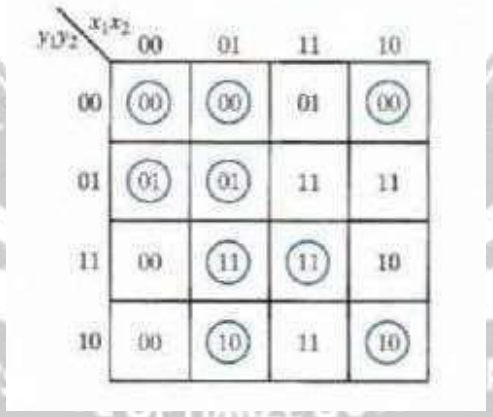


Fig: 4.9 - Transition table

Image source from Digital Design by Moris Mano (Page No. 429)

The procedure for analyzing an asynchronous sequential circuit with SR latches can be summarized as follows:

1. Label each latch output with Y_i and its external feedback path (if any) with Yi for i

   = 1,

   2, ... k.
2. Derive the Boolean functions for the S_i and R_i inputs in each latch.
3. Check whether SR = 0 for each NOR latch or whether S'R' = 0 for each NAND latch. If either of these conditions is not satisfied, there is a possibility that the circuit may not operate properly.
4. Evaluate Y = S + R'y for each NOR latch or Y = S' + Ry for each NAND latch.
5. Consruct a map, with the y's representing the rows and the x inputs representing the colwnns.
6. Plot the value of $Y= Y_1Y_2.....Y_K$ in the map.
7. Circle all stable states such that Y = y. The resulting map is then the transition table.

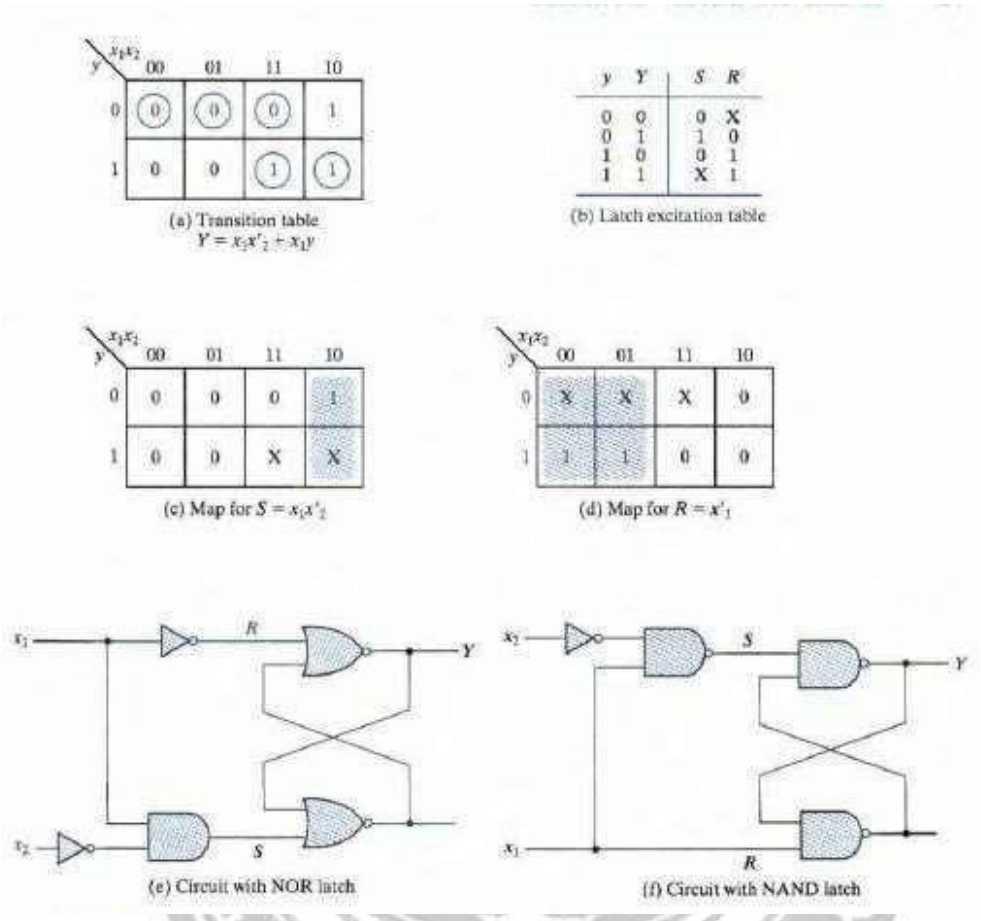


Fig: 4.10 - Derivation of a Latch Circuit fromTransition table

Image source from Digital Design by Moris Mano (Page No. 431)

DESIGN PROCEDURE

The design of an asynchronous sequential circuit starts from the statement of the problem and culminates in a logic diagram. There are a number of design steps that must be carried out in order to minimize the complexity of the circuit and to produce a stable circuit without critical races. Briefly, the design steps are as follows: A primitive flow table is obtained from the design specifications. The flow table is then reduced to a minimum number of states. Next, the states are given a binary assignment from which we obtain the transition table. Finally, from the transition table, we derive the logic diagram as a combinational circuit with feedback or as a circuit with SR latches.

DESIGN EXAMPLE

It is necessary to design a gated latch circuit with two inputs G (gate) and D (data) and one output Q. Binary information present at the D input is transferred to the Q output when G is equal to 1. The Q output will follow the D input as long as G = 1. When G goes to 0, the information that was present at the D input at the time the transition occurred is retained at the Q output. The gated latch is a memory element that accepts the value of D when G = 1 and retains this value after G goes to 0. Once G = 0, a change in D does not change the value of the output Q.

Primitive Flow Table

A primitive flow table is a flow table with only one stable total state in each row. Remember that a total stale consists of the internal state combined with the input. The derivation of the primitive flow table can be facilitated if we first form a table with all possible total states in the system.

Inputs *DG*

States	00	01	11	10
a	*c*, –	(*a*), 0	*b*, –	–, –
b	–, –	*a*, –	(*b*), 1	*e*, –
c	(*c*), 0	*a*, –	–, –	*d*, –
d	*c*, –	–, –	*b*, –	(*d*), 0
e	*f*, –	–, –	*b*, –	(*e*), 1
f	(*f*), 1	*a*, –	–, –	*e*, –

Fig: 4.10 - Primitive flow table

Image source from Digital Design by Moris Mano (Page No. 434)

The primitive flow table for the gated latch has one row for each state and one column for each input combination. First, we fill in one square in each row belonging to the stable state in that row. These entries are determined from Table. For example, Slate Q is stable and the output is 0 when the input is 01. This information is entered into the flow table in the first row and second column. Similarly, the other five stable stales together with their output are entered into the corresponding input columns. Next, we note that since both inputs are not allowed to change simultaneously, we can enter dash marks in each row that differs in two or more variables from the input variables associated with the stable state. For example, the first row in the flow table shows a stable state with an input of 01. Since only one input can change at any given time, it can change to 00 or 11, but not to 10. Therefore, we enter two dashes in the 10 column of row a. This will eventually result in a don't-care condition for the next state and output in this square. Following the same procedure, we fill in a second square in each row of the primitive flow table.

Next, it is necessary to find values for two more squares in each row. The comments listed in Table may help in deriving the necessary information. For example, state c is associated with input 00 and is reached after a change in input from state a or d. Therefore, an unstable state c is shown in column 00 and rows a and d in the flow table. The output is marked with a dash to indicate a don't-care condition. The interpretation of this situation is that if the circuit is in stable state a and the input changes from 01 to 00, the circuit first goes to an unstable next state c. which changes

the present-slate value from a to c, causing a transition to the third row and first column of the table. The unstable state values for the other square s are determined in a similar manner. All outputs associated with unstable states are marked with a dash to indicate don't -care conditions.

Reduction of the Primitive flow table

The primitive flow table has only one stable state in each row. The table can be reduced to a smaller number of rows if two or more stable states are placed in the same row. The grouping of stable states from separate rows into one common row is called merging. Merging a number of stable states in the same row means that the binary state variable ultimately assigned to the merged row will not change when the input variable changes. This is because, in a primitive flow table, the stale variable changes every time the input changes, but in a reduced flow table, a change of input will not cause a change in the state variable if the next stable state is in the same row.

Two or more rows in the primitive flow table can be merged into one row if there are non conflicting stales and outputs in each of the columns. Whenever one state symbol and don't -care entries are encountered in the same column, the state is listed in the merged row.

Moreover, if the stale is circled in one of the rows, it is also circled in the merged row. The output value is included with each stable slate in the merged row. Because the merged stales have the same output, the state cannot be distinguished on the basis of the output.

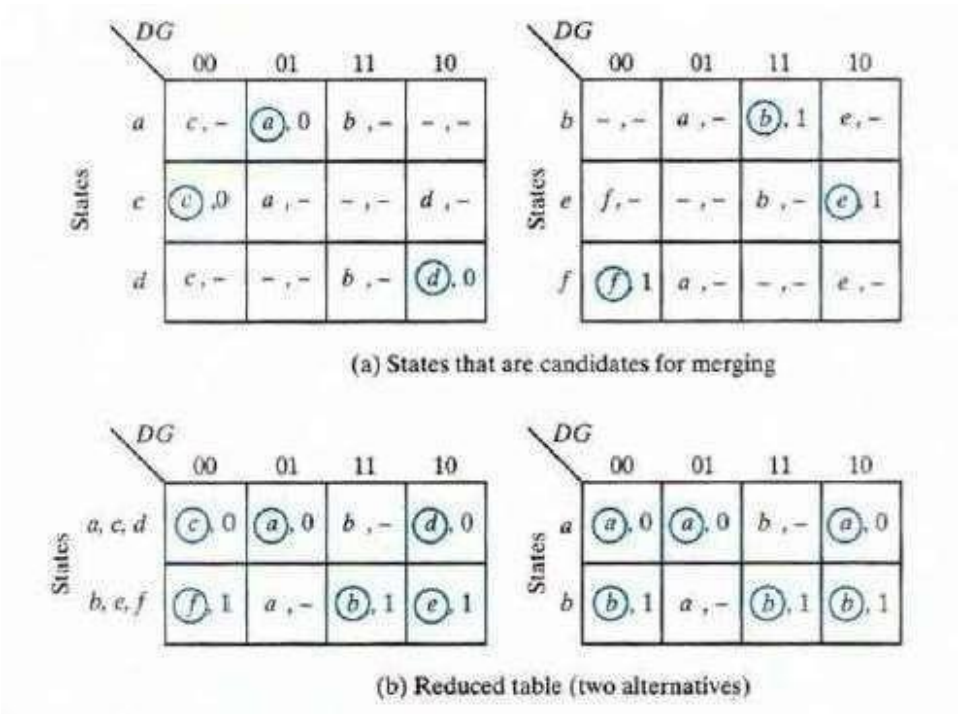


Fig: 4.11 - Reduction of primitive flow tables

Image source from Digital Design by Moris Mano (Page No. 436)

Transition table and Logic Diagram

In order to obtain the circuit described by the reduced flow table, it is necessary to assign a distinct binary value to each state. This assignment converts the flow table into a transition table. In the general case, a binary state assignment must be made to ensure that the circuit will be free of critical races. Fortunately, there can be no critical races in a two-row flow table; therefore, we can finish the design of the gated latch prior to studying that section.Assigning 0 to state a and 1 to state b in the reduced flow table of Fig. (b), we obtain the transition table of Fig. (a). The transition table is, in effect, a map for the excitation variable Y. The simplified Boolean function for Y is then obtained from the map as

Y = DC + C'y

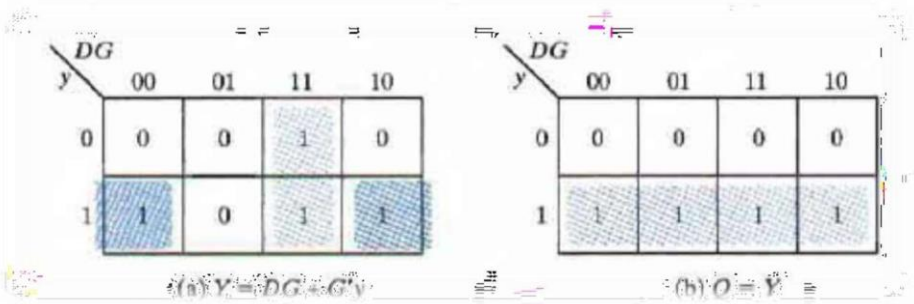


Fig: 4.12 - Transition table and output map for a gated latch

Image source from Digital Design by Moris Mano (Page No. 437)

Fig. Transition Table and output amp for gated latch

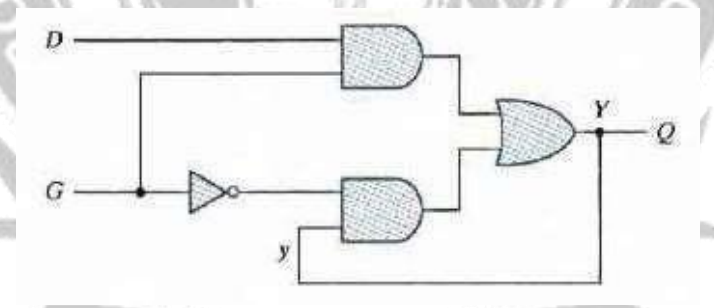


Fig : 4.13 : Gated-latch logic diagram

Image source from Digital Design by Moris Mano (Page No. 437)

Assigning outputs to unstable states

The procedure for making the assignment to outputs associated with unstable states can be summarized as follows:

1. Assign a 0 to an output variable associated with an unstable state which is a transient state between two stab le states that have a 0 in the corresponding output variable.
2. Assign a 1 to an output variable associated with an unstable state which is a transient state between two stable states that have a 1 in the corresponding output variable.
3. Assign a don' t-care condition to an output variable associated with an unstable state which is a transient state between two stable states that have different value s (0 and 1, or 1 and 0) in the corresponding output variable .

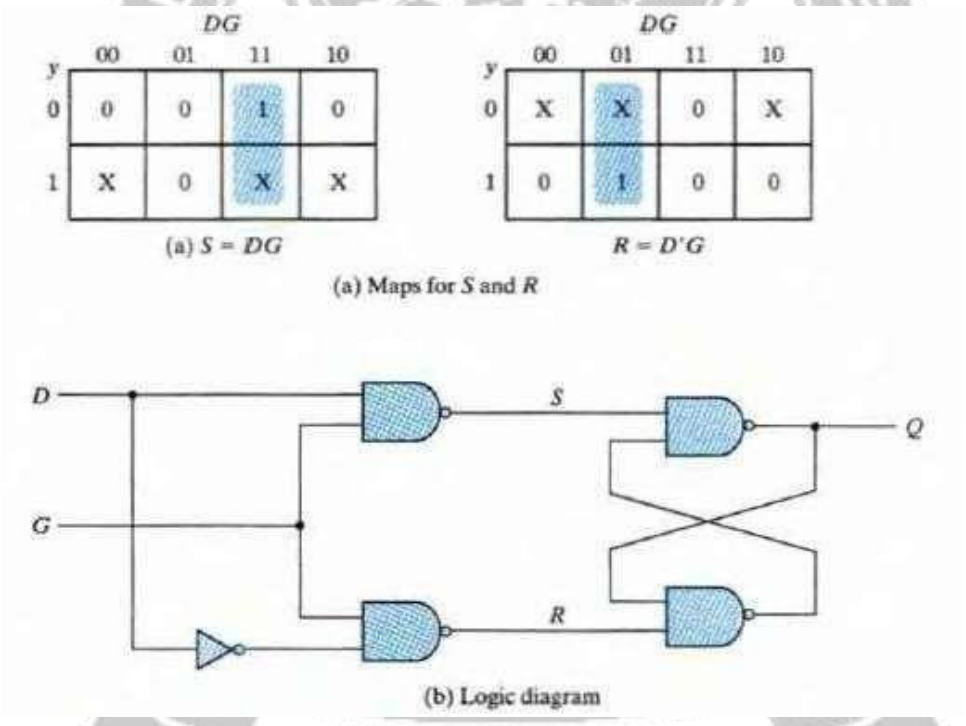
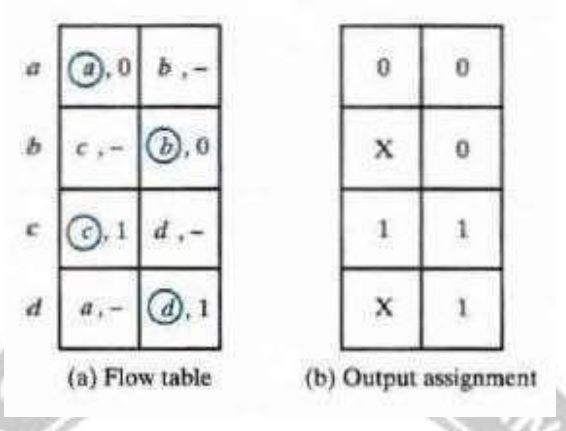


Fig 4.14 – Circuit with SR Latches

Image source from Digital Design by Moris Mano (Page No. 438)



a	(a), 0	b, –
b	c, –	(b), 0
c	(c), 1	d, –
d	a, –	(d), 1

(a) Flow table

0	0
X	0
1	1
X	1

(b) Output assignment

Fig 4.15 – Assigning output values to unstable states

Image source from Digital Design by Moris Mano (Page No. 438)