**Table 7-4** Transition/output and state/output tables for the state machine in Figure 7-43.

(a)

| | XY | | | | |
|---|---|---|---|---|---|
| Q2 Q1 Q0 | 00 | 01 | 10 | 11 | Z1 Z2 |
| 000 | 000 | 100 | 001 | 001 | 10 |
| 001 | 001 | 001 | 011 | 011 | 10 |
| 010 | 010 | 110 | 000 | 000 | 10 |
| 011 | 011 | 011 | 010 | 010 | 00 |
| 100 | 101 | 101 | 101 | 101 | 11 |
| 101 | 001 | 001 | 001 | 001 | 10 |
| 110 | 111 | 111 | 111 | 111 | 11 |
| 111 | 011 | 011 | 011 | 011 | 11 |
| | Q2* Q1* Q0* | | | | |

(b)

| | XY | | | | |
|---|---|---|---|---|---|
| S | 00 | 01 | 10 | 11 | Z1 Z2 |
| A | A | E | B | B | 10 |
| B | B | B | D | D | 10 |
| C | C | G | A | A | 10 |
| D | D | D | C | C | 00 |
| E | F | F | F | F | 11 |
| F | B | B | B | B | 10 |
| G | H | H | H | H | 11 |
| H | D | D | D | D | 11 |
| | S* | | | | |

A transition table based on these equations is shown in Table 7-4(a). Reading the logic diagram, we can write two output equations:

$$Z1 = Q2 + Q1' + Q0'$$
$$Z2 = Q2 \cdot Q1 + Q2 \cdot Q0'$$

The resulting output values are shown in the last column of (a). Assigning state names A–H, we obtain the state/output table shown in (b).

A state diagram for the example machine is shown in Figure 7-44. Since our example is a Moore machine, the output values are written with each state. Each arc is labeled with a *transition expression*; a transition is taken for input combinations for which the transition expression is 1. Transitions labeled "1" are always taken.

*transition expression*

**Figure 7-44** State diagram corresponding to Table 7-4.

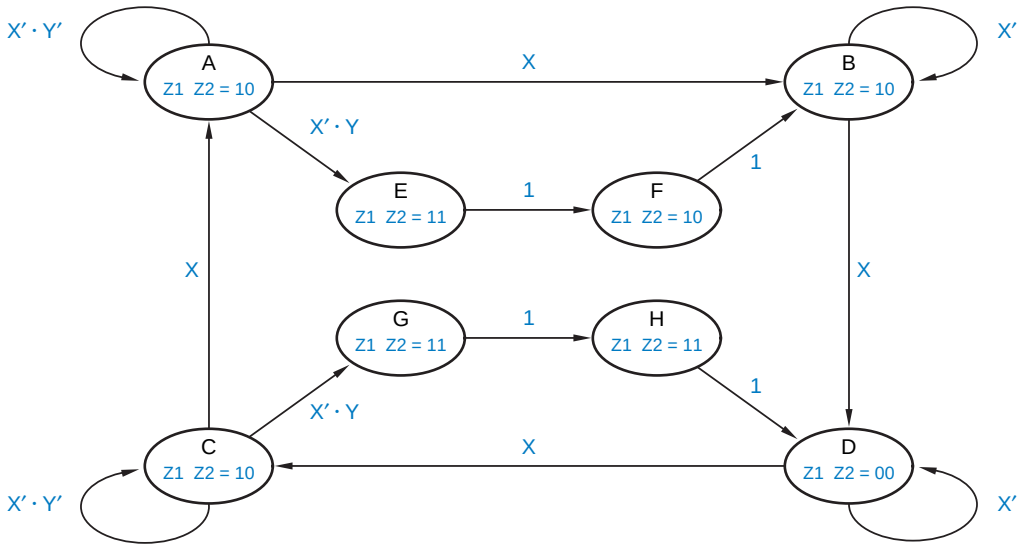


**Table 7-5** Transition/output and state/output tables for the state machine in Figure 7-45.

(a)

| | XY | | | |
|---|---|---|---|---|
| Q1 Q0 | 00 | 01 | 10 | 11 |
| 00 | 00, 0 | 10, 1 | 01, 0 | 10, 1 |
| 01 | 01, 0 | 11, 0 | 10, 0 | 11, 0 |
| 10 | 10, 0 | 00, 0 | 11, 0 | 00, 0 |
| 11 | 11, 0 | 10, 0 | 00, 1 | 10, 1 |
| | Q1* Q0*, Z | | | |

(b)

| | XY | | | |
|---|---|---|---|---|
| S | 00 | 01 | 10 | 11 |
| A | A, 0 | C, 1 | B, 0 | C, 1 |
| B | B, 0 | D, 0 | C, 0 | D, 0 |
| C | C, 0 | A, 0 | D, 0 | A, 0 |
| D | D, 0 | C, 0 | A, 1 | C, 1 |
| | S*, Z | | | |

Substituting into the characteristic equation for J-K flip-flops, we obtain the transition equations:

$$
\begin{aligned}
Q0^* &= J0 \cdot Q0' + K0' \cdot Q0 \\
&= X \cdot Y' \cdot Q0' + (X \cdot Y' + Y \cdot Q1)' \cdot Q0 \\
&= X \cdot Y' \cdot Q0' + X' \cdot Y' \cdot Q0 + X' \cdot Q1' \cdot Q0 + Y \cdot Q1' \cdot Q0 \\
Q1^* &= J1 \cdot Q1' + K1' \cdot Q1 \\
&= (X \cdot Q0 + Y) \cdot Q1' + (Y \cdot Q0' + X \cdot Y' \cdot Q0)' \cdot Q1 \\
&= X \cdot Q1' \cdot Q0 + Y \cdot Q1' + X' \cdot Y' \cdot Q1 + Y' \cdot Q1 \cdot Q0' + X' \cdot Q1 \cdot Q0 + Y \cdot Q1 \cdot Q0
\end{aligned}
$$

A transition table based on these equations is shown in Table 7-5(a). Reading the logic diagram, we can write the output equation:

$$Z = X \cdot Q1 \cdot Q0 + Y \cdot Q1' \cdot Q0'$$

The resulting output values are shown in each column of (a) along with the next state. Assigning state names A–D, we obtain the state/output table shown in (b). A corresponding state diagram that uses transition expressions is shown in Figure 7-46.

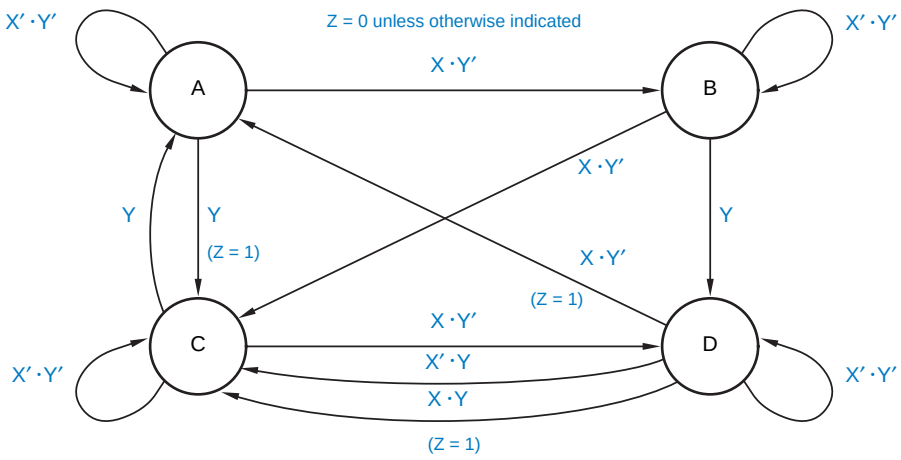


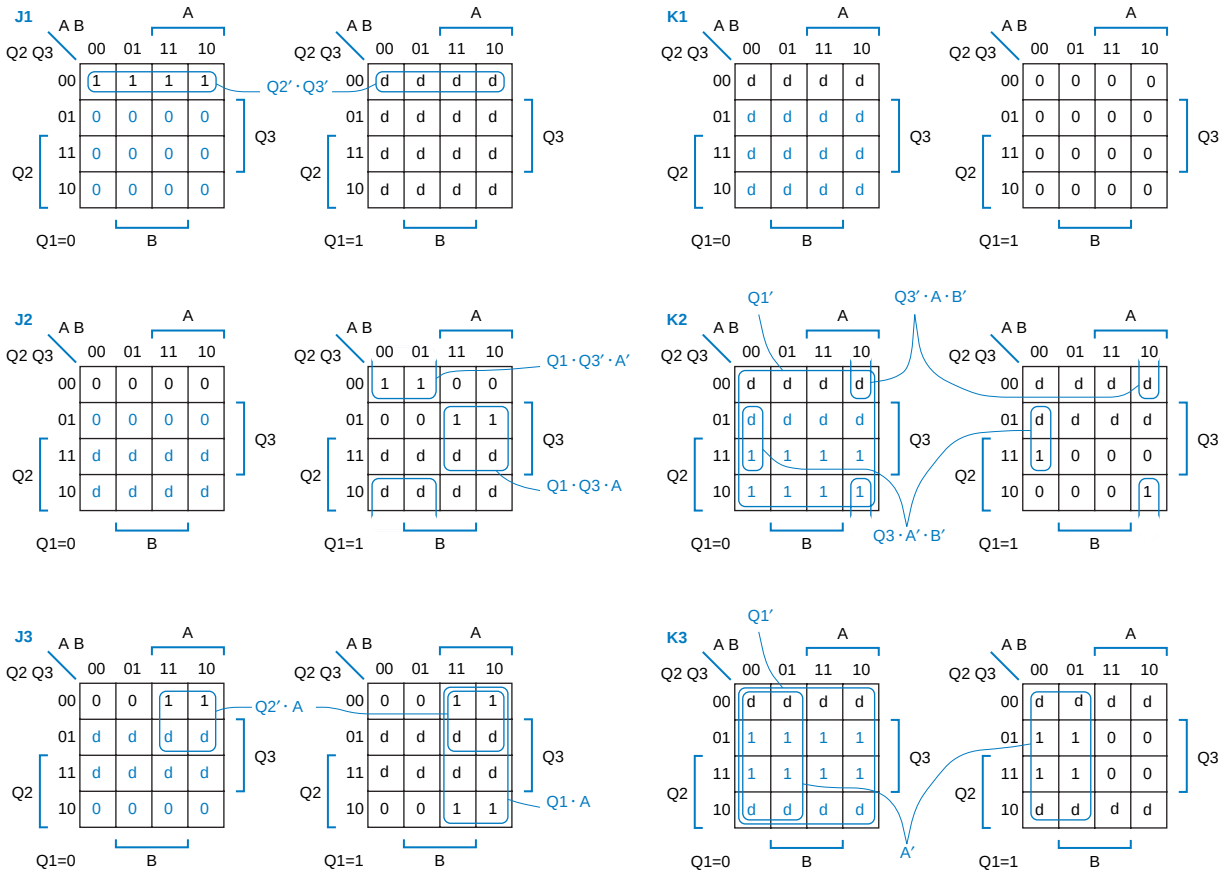

**Figure 7-46** State diagram corresponding to the state machine of Table 7-5.

corresponding pair of J and K values from the application table. For the transition table in Table 7-8 on page 573, these substitutions produce the excitation table in Table 7-11. For example, in state 100 under input combination 00, Q1 is 1 and the required Q1* is 1; therefore, "d0" is entered for J1 K1. For the same state/input combination, Q2 is 0 and Q2* is 1, so "1d" is entered for J2 K2. Obviously, it takes quite a bit of patience and care to fill in the entire excitation table (a job best left to a computer).

As in the D synthesis example of the preceding subsection, the excitation table is *almost* a truth table for the excitation functions. This information is transferred to Karnaugh maps in Figure 7-55.

The excitation table does not specify next states for the unused states, so once again we must choose between the minimal-risk and minimal-cost approaches. The colored entries in the Karnaugh maps result from taking the minimal-risk approach.

**Figure 7-55** Excitation maps for J1, K1, J2, K2, J3, and K3, assuming that unused states go to state 000.



| | X Y | | | | |
|---|---|---|---|---|---|
| Q1 Q2 | 00 | 01 | 11 | 10 | Z |
| 00 | 00 | 01 | 11 | 01 | 1 |
| 01 | 01 | 11 | 10 | 11 | 0 |
| 11 | 11 | 10 | 00 | 10 | 0 |
| 10 | 10 | 00 | 01 | 00 | 0 |
| | Q1* Q2* or D1 D2 | | | | |

**Table 7-13** Transition/excitation and output table for 1s-counting machine.

The 1s-counting machine can use two state variables to code its four states, with no unused states. In this case, there are only 4! possible assignments of coded states to named states. Still, we'll try only one of them. We'll assign coded states to the named states in Karnaugh-map order (00, 01, 11, 10) for two reasons: in this state table, it minimizes the number of state variables that change for most transitions, potentially simplifying the excitation equations; and it simplifies the mechanical transfer of information to excitation maps.

A transition/excitation table based on our chosen state assignment is shown in Table 7-13. Since we're using D flip-flops, the transition and excitation tables are the same. Corresponding Karnaugh maps for D1 and D2 are shown in Figure 7-57. Since there are no unused states, all of the information we need is in the excitation table; no choice is required between minimal-risk and minimal-cost approaches. The excitation equations can be read from the maps, and the output equation can be read directly from the transition/excitation table:

$$D1 = Q2 \cdot X' \cdot Y + Q1' \cdot X \cdot Y + Q1 \cdot X' \cdot Y' + Q2 \cdot X \cdot Y'$$
$$D2 = Q1' \cdot X' \cdot Y + Q1' \cdot X \cdot Y' + Q2 \cdot X' \cdot Y' + Q2' \cdot X \cdot Y$$
$$Z = Q1' \cdot Q2'$$

A logic diagram using D flip-flops and AND-OR or NAND-NAND excitation logic can be drawn from these equations.

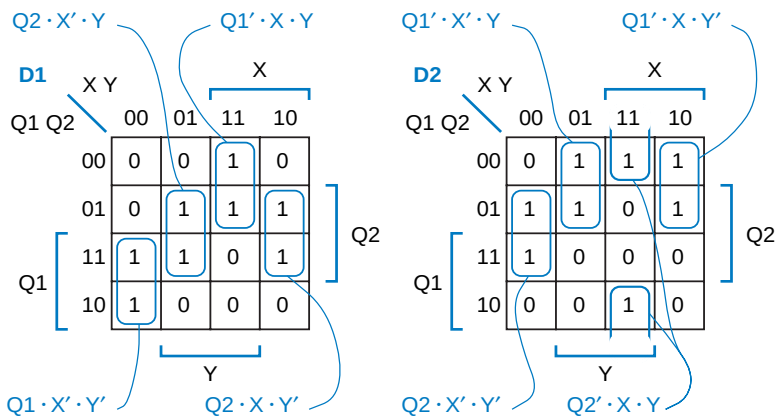



**Figure 7-57** Excitation maps for D1 and D2 inputs in 1s-counting machine.

| | X | |
|---|---|---|
| Q1 Q2 Q3 | 0 | 1 |
| 000 | 001, 01 | 000, 00 |
| 001 | 001, 00 | 010, 01 |
| 010 | 001, 00 | 011, 01 |
| 011 | 100, 01 | 000, 00 |
| 100 | 001, 00 | 101, 01 |
| 101 | 001, 00 | 110, 01 |
| 110 | 100, 00 | 111, 01 |
| 111 | 001, 11 | 000, 00 |
| | Q1* Q2* Q3*, UNLK HINT | |

**Table 7-15** Transition/excitation table for combination-lock machine.

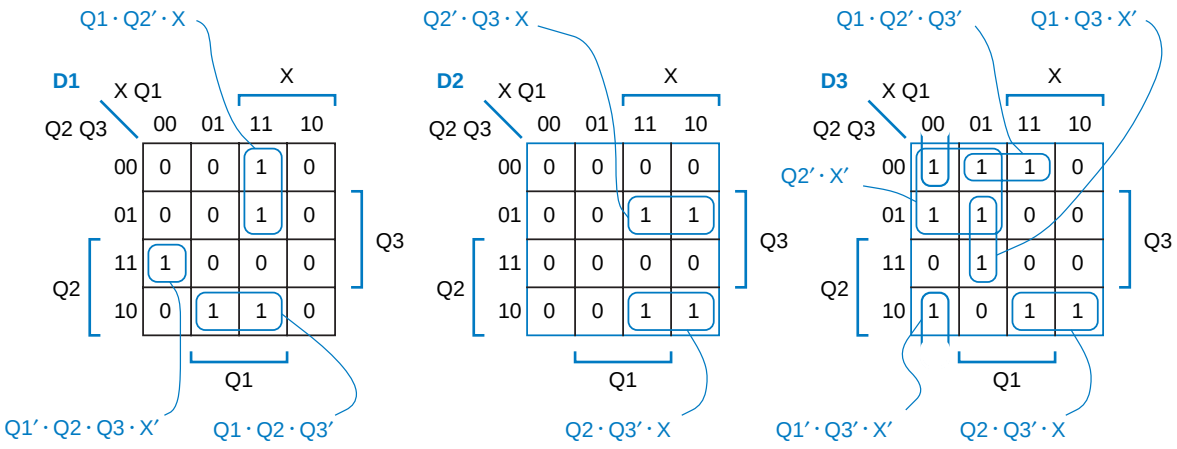
The combination lock's eight states can be coded with three state variables, leaving no unused states. There are 8! state assignments to choose from. To keep things simple, we'll use the simplest, and assign the states in binary counting order, yielding the transition/excitation table in Table 7-15. Corresponding Karnaugh maps for D1, D2, and D3 are shown in Figure 7-58. The excitation equations can be read from the maps:

$$D1 = Q1 \cdot Q2' \cdot X + Q1' \cdot Q2 \cdot Q3 \cdot X' + Q1 \cdot Q2 \cdot Q3'$$

$$D2 = Q2' \cdot Q3 \cdot X + Q2 \cdot Q3' \cdot X$$

$$D3 = Q1 \cdot Q2' \cdot Q3' + Q1 \cdot Q3 \cdot X' + Q2' \cdot X' + Q3' \cdot Q1' \cdot X' + Q2 \cdot Q3' \cdot X$$

**Figure 7-58** Excitation maps for D1, D2, and D3 in combination-lock machine.



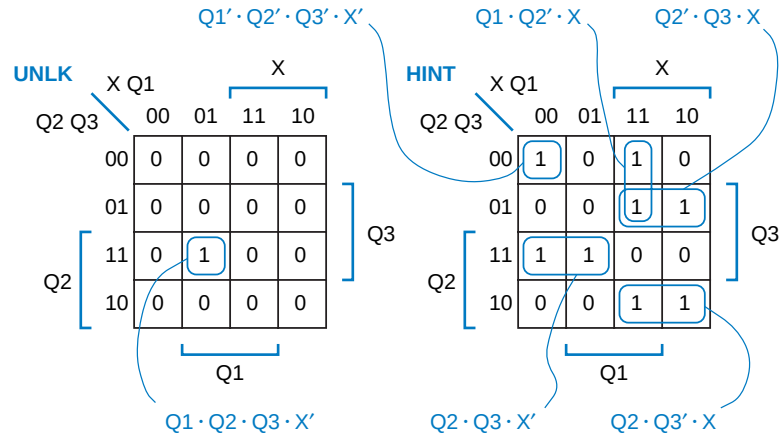



**Figure 7-59** Karnaugh maps for output functions UNLK and HINT in combination-lock machine.

The output values are transferred from the transition/excitation and output table to another set of maps in Figure 7-59. The corresponding output equations are:

$$\text{UNLK} = Q1 \cdot Q2 \cdot Q3 \cdot X'$$

$$\text{HINT} = Q1' \cdot Q2' \cdot Q3' \cdot X' + Q1 \cdot Q2' \cdot X + Q2' \cdot Q3 \cdot X + Q2 \cdot Q3 \cdot X' + Q2 \cdot Q3' \cdot X$$

Note that some product terms are repeated in the excitation and output equations, yielding a slight savings in the cost of the AND-OR realization. If we went through the trouble of performing a formal multiple-output minimization of all five excitation and output functions, we could save two more gates (see Exercise 7.55).

## 7.5 Designing State Machines Using State Diagrams

Aside from planning the overall architecture of a digital system, designing state machines is probably the most creative task of a digital designer. Most people like to take a graphical approach to design—you've probably solved many problems just by doodling. For that reason, state diagrams are often used to design small- to medium-sized state machines. In this section we'll give examples of state-diagram design and describe a simple procedure for synthesizing circuits from the state diagrams. This procedure is the basis of the method used by CAD tools that can synthesize logic from graphical or even text-based "state diagrams."

Designing a state diagram is much like designing a state table, which, as we showed in Section 7.4.1, is much like writing a program. However, there is one fundamental difference between a state diagram and a state table, a difference that makes state-diagram design simpler but also more error prone:

- A state table is an exhaustive listing of the next states for each state/input combination. No ambiguity is possible.

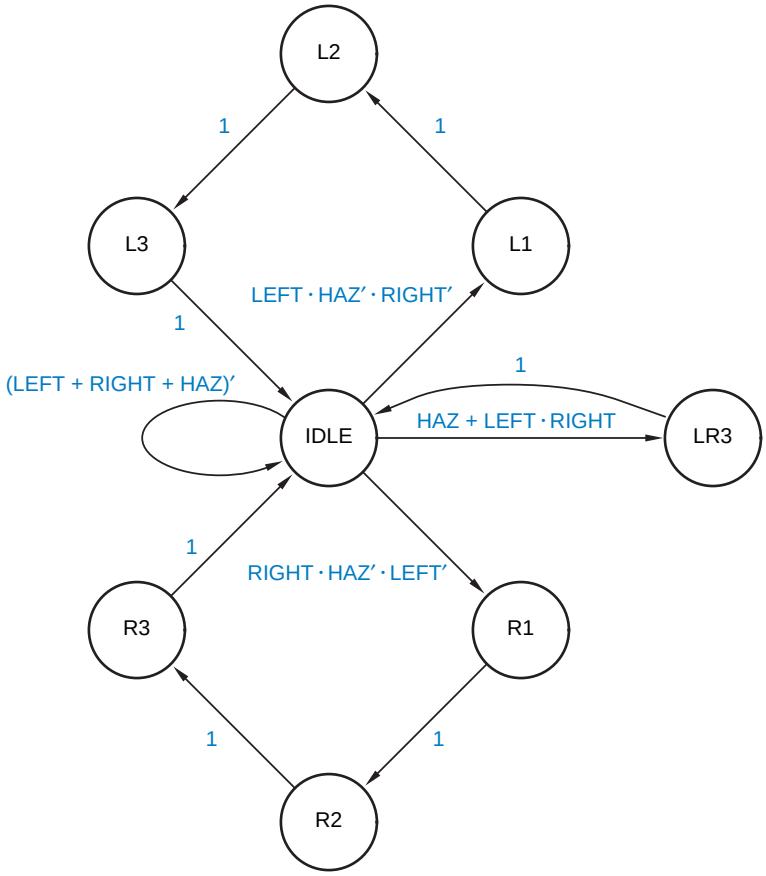



**Figure 7-63**
Corrected state diagram for T-bird tail lights.

If there are many transitions leaving each state, these steps, especially the first one, are very difficult to perform. However, typical state machines, even ones with lots of states and inputs, don't have many transitions leaving each state, since most designers can't dream up such complex machines in the first place. This is where the tradeoff between state-table and state-diagram design occurs. In state-table design, the foregoing steps are not required, because the structure of a state table guarantees mutual exclusion and all inclusion. But if there are a lot of inputs, the state table has *lots* of columns.

Verifying that a state diagram is unambiguous may be difficult in principle, but it's not too bad in practice for small state diagrams. In Figure 7-63, most of the states have a single arc with a transition expression of 1, so verification is trivial. Real work is needed only to verify the IDLE state, which has four transitions leaving it. This can be done on a sheet of scratch paper by listing the eight combinations of the three inputs and checking off the combinations covered by each transition expression. Each combination should have exactly one check. As another example, consider the state diagrams in Figures 7-44 and 7-46 on pages 560 and 562; both can be verified mentally.

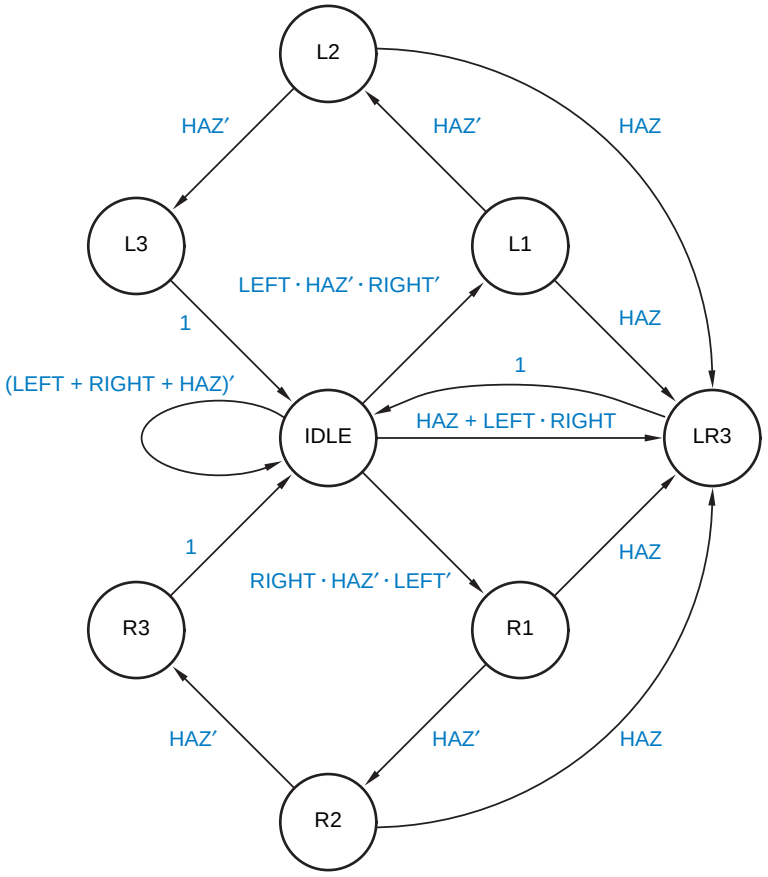



**Figure 7-64** Enhanced state diagram for T-bird tail lights.

Returning to the T-bird tail-lights machine, we can now synthesize a circuit from the state diagram if we wish. However, if we want to change the machine's behavior, now is the time to do it, before we do all the work of synthesizing a circuit. In particular, notice that once a left- or right-turn cycle has begun, the state diagram in Figure 7-63 allows the cycle to run to completion, even if HAZ is asserted. While this may have a certain aesthetic appeal, it would be safer for the car's occupants to have the machine go into hazard mode as soon as possible. The state diagram is modified to provide this behavior in Figure 7-64.

Now we're finally ready to synthesize a circuit for the T-bird machine. The state diagram has eight states, so we'll need a minimum of three flip-flops to code the states. Obviously, many state assignments are possible (8! to be exact); we'll use the one in Table 7-16 for the following reasons:

1. An initial (idle) state of 000 is compatible with most flip-flops and registers, which are easily initialized to the 0 state.
2. Two state variables, Q1 and Q0, are used to "count" in Gray-code sequence for the left-turn cycle (IDLE→L1→L2→L3→IDLE). This minimizes the

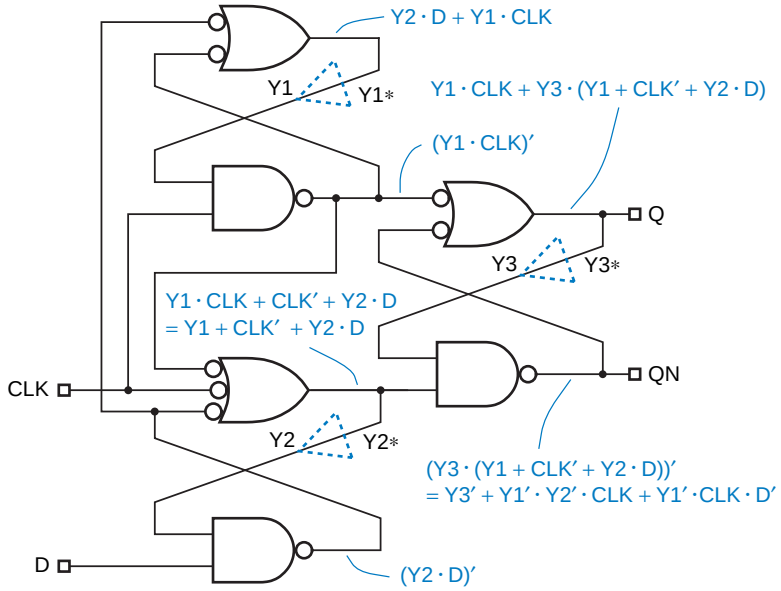



**Figure 7-78**
Simplified positive edge-triggered D flip-flop for analysis.

The following excitation and output equations can be derived from the logic diagram in Figure 7-78:

$$Y1* = Y2 \cdot D + Y1 \cdot CLK$$
$$Y2* = Y1 + CLK' + Y2 \cdot D$$
$$Y3* = Y1 \cdot CLK + Y1 \cdot Y3 + Y3 \cdot CLK' + Y2 \cdot Y3 \cdot D$$
$$Q = Y1 \cdot CLK + Y1 \cdot Y3 + Y3 \cdot CLK' + Y2 \cdot Y3 \cdot D$$
$$QN = Y3' + Y1' \cdot Y2' \cdot CLK + Y1' \cdot CLK \cdot D'$$

**Figure 7-79**
Transition table for the D flip-flop in Figure 7-78.

| | CLK D | | | |
|---|---|---|---|---|
| Y1 Y2 Y3 | 00 | 01 | 11 | 10 |
| 000 | 010 | 010 | (000) | (000) |
| 001 | 011 | 011 | 000 | 000 |
| 010 | (010) | 110 | 110 | 000 |
| 011 | (011) | 111 | 111 | 000 |
| 100 | 010 | 010 | 111 | 111 |
| 101 | 011 | 011 | 111 | 111 |
| 110 | 010 | (110) | 111 | 111 |
| 111 | 011 | (111) | (111) | (111) |
| | Y1* Y2* Y3* | | | |

| Y1 Y2 | P R = 00 | P R = 01 | P R = 11 | P R = 10 | Z |
|---|---|---|---|---|---|
| 00 | (00) | (00) | 10 | 01 | 0 |
| 01 | (01) | 00 | 11 | (01) | 1 |
| 11 | — | — | 10 | — | — |
| 10 | 00 | 00 | (10) | (10) | 0 |
| | Y1* Y2* | | | | |

**Figure 7-102**
Transition table for the pulse-catching circuit, exhibiting an essential hazard.

The only way to avoid this erroneous behavior in general is to ensure that changes in P arrive at the inputs of all the excitation circuits before any changes in state variables do. Thus, the inevitable difference in input arrival times, called *timing skew*, must be less than the propagation delay of the excitation circuits and feedback loops. This timing requirement can generally be satisfied only by careful design *at the electrical circuit level*.

*timing skew*

In the example circuit, it would appear that the hazard is easily masked, even by non-electrical engineers, since the designer need only ensure that a straight wire has shorter propagation delay than an AND-OR structure—easy in most technologies.

Still, many feedback sequential circuits, such as the TTL edge-triggered D flip-flop in Figure 7-19, have essential hazards in which the input skew paths include inverters. In such cases, the input inverters must be guaranteed to be faster than the excitation logic; that's not so trivial in either board-level or IC design. For example, if the excitation circuit in Figure 7-101 were physically built using AND-OR-INVERT gates, the delay from input changes to Y1_L could be very short indeed, as short as the delay through a single inverter.

Essential hazards can be found in most but not all fundamental-mode circuits. There's an easy rule for detecting them; in fact, this is the definition of "essential hazard" in some texts:.

- A fundamental-mode flow table contains an essential hazard for a stable total state S and an input variable X if, starting in state S, the stable total state reached after three successive transitions in X is different from the stable total state reached after one transition in X.

**THESE HAZARDS ARE, WELL, ESSENTIAL!**

Essential hazards are called "essential" because they are inherent in the flow table for a particular sequential function and will appear in any circuit realization of that function. They can be masked only by controlling the delays in the circuit. Compare with static hazards in combinational logic, where we could eliminate hazards by adding consensus terms to a logic expression.