change was worthwhile. However, we must remember that the new machine is different from the one in Table 9-7. Consider what happens if the machine ever gets into an unspecified state. In the original machine with fully specified output coding, there are no next-states for the $2^5 - 6 = 26$ unspecified states, so the state machine will always go to the state coded 00000 (SOK) from unspecified states. In the new machine, "unspecified" states aren't really unspecified; for example, the state coded 11111 actually matches five coded states, S1–S4 and SERR. The next state will actually be the "OR" of next-states for the matching coded states. (Read the box on the previous page to understand why these outcomes occur.) Again, you need to be careful.

RESETTING EXPECTATIONS

Reading the guessing-game program in Table 9-5, you would expect that the RESET input would force the machine to the SOK state, and it does. However, the moment that you have unspecified or partially coded states as in Tables 9-7 or 9-8, don't take anything for granted.

Referring to the box on the previous page, remember that transition statements in ABEL state machines augment the on-sets of state variables. If a particular, unused state combination does not match any of the states for which transition statements were written, then no on-sets will be augmented. Thus, the only transition from that state will be to the state with the all-0s coding.

For this reason, it is useful to code the reset state or a "safe" state as all 0s. If this is not possible, but the all-0s state is still unused, you can explicitly provide a transition from the all-0s state to a desired safe state.

9.1.5 Reinventing Traffic-Light Controllers

Our final example is from the world of cars and traffic. Traffic-light controllers in California, especially in the fair city of Sunnyvale, are carefully designed to *maximize* the waiting time of cars at intersections. An infrequently used intersection (one that would have no more than a "yield" sign if it were in Chicago) has the sensors and signals shown in Figure 9-5. The state machine that controls the traffic signals uses a 1 Hz clock and a timer and has four inputs:

NSCAR Asserted when a car on the north-south road is over either sensor on either side of the intersection.

EWCAR Asserted when a car on the east-west road is over either sensor on either side of the intersection.

TMLONG Asserted if more than five minutes has elapsed since the timer started; remains asserted until the timer is reset.

TMSHORT Asserted if more than five seconds has elapsed since the timer started; remains asserted until the timer is reset.

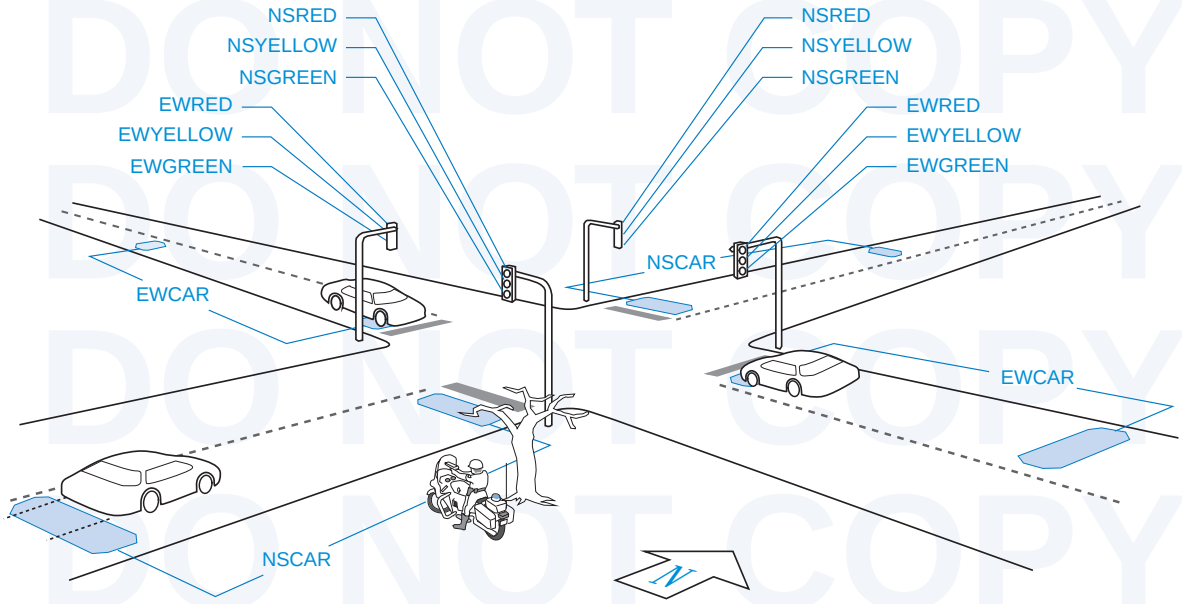

Figure 9-5 Traffic sensors and signals at an intersection in Sunnyvale, California.

The state machine has seven outputs:

NSRED, NSYELLOW, NSGREEN	Control the north-south lights.
EWRED, EWYELLOW, EWGREEN	Control the east-west lights.
TMRESET	When asserted, resets the timer and negates TMSHORT and TMLONG. The timer starts timing when TMRESET is negated.

A typical, municipally approved algorithm for controlling the traffic lights is embedded in the ABEL program of Table 9-9. This algorithm produces two frequently seen behaviors of "smart" traffic lights. At night, when traffic is light, it holds a car stopped at the light for up to five minutes, unless a car approaches on the cross street, in which case it stops the cross traffic and lets the waiting car go. (The "early warning" sensor is far enough back to change the lights before the approaching car reaches the intersection.) During the day, when traffic is heavy and there are always cars waiting in both directions, it cycles the lights every five seconds, thus minimizing the utilization of the intersection, and maximizing everyone's waiting time and creating public outcry for more taxes to fix the problem.

The equations for the TMRESET output are worth noting. This output is asserted during the "double-red" states, NSDELAY and EWDELAY, to reset the timer in preparation for the next green cycle. The desired output signal could be generated on a combinational output pin by decoding these two states, but we have chosen instead to generate it on a registered output pin by decoding the *predecessors* of these two states.

Table 9-9 Sunnyvale traffic-lights program.

```
module svaletl
title 'State Machine for Sunnyvale, CA, Traffic Lights'
SVALETL device 'P16V8R';

" Input and output pins
CLOCK, !OE                      pin 1, 11;
NSCAR, EWCAR, TMSHORT, TMLONG   pin 2, 3, 8, 9;
Q0, Q1, Q2, TMRESET_L           pin 17, 16, 15, 14 istype 'reg';

" Definitions
LSTATE  = [Q2,Q1,Q0];           " State variables
NSGO    = [ 0, 0, 0];           " States
NSWAIT  = [ 0, 0, 1];
NSWAIT2 = [ 0, 1, 1];
NSDELAY = [ 0, 1, 0];
EWGO    = [ 1, 1, 0];
EWWAIT  = [ 1, 1, 1];
EWWAIT2 = [ 1, 0, 1];
EWDELAY = [ 1, 0, 0];

state_diagram LSTATE
state NSGO:                     " North-south green
   IF (!TMSHORT) THEN NSGO      " Minimum green is 5 seconds.
   ELSE IF (TMLONG) THEN NSWAIT " Maximum green is 5 minutes.
   ELSE IF (EWCAR & !NSCAR)     " If E-W car is waiting and no one
         THEN NSGO              "  is coming N-S, make E-W wait!
   ELSE IF (EWCAR & NSCAR)      " Cars coming in both directions?
         THEN NSWAIT            " Thrash!
   ELSE IF (!NSCAR)             " Nobody coming N-S and not timed out?
         THEN NSGO              " Keep N-S green.
   ELSE NSWAIT;                 " Else let E-W have it.

state NSWAIT: GOTO NSWAIT2;     " Yellow light is on for two ticks for safety.
state NSWAIT2: GOTO NSDELAY;    " (Drivers go 70 mph to catch this turkey green!)
state NSDELAY: GOTO EWGO;       " Red in both directions for added safety.

state EWGO: " East-west green; states defined analogous to N-S
   IF (!TMSHORT) THEN EWGO
   ELSE IF (TMLONG) THEN EWWAIT
   ELSE IF (NSCAR & !EWCAR) THEN EWGO
   ELSE IF (NSCAR & EWCAR) THEN EWWAIT
   ELSE IF (!EWCAR) THEN EWGO  ELSE EWWAIT;

state EWWAIT: GOTO EWWAIT2;
state EWWAIT2: GOTO EWDELAY;
state EWDELAY: GOTO NSGO;

equations

LSTATE.CLK = CLOCK;  TMRESET_L.CLK = CLOCK;
!TMRESET_L := (LSTATE == NSWAIT2)  " Reset the timer when going into
            + (LSTATE == EWWAIT2); "   state NSDELAY or state EWDELAY.
end svaletl
```

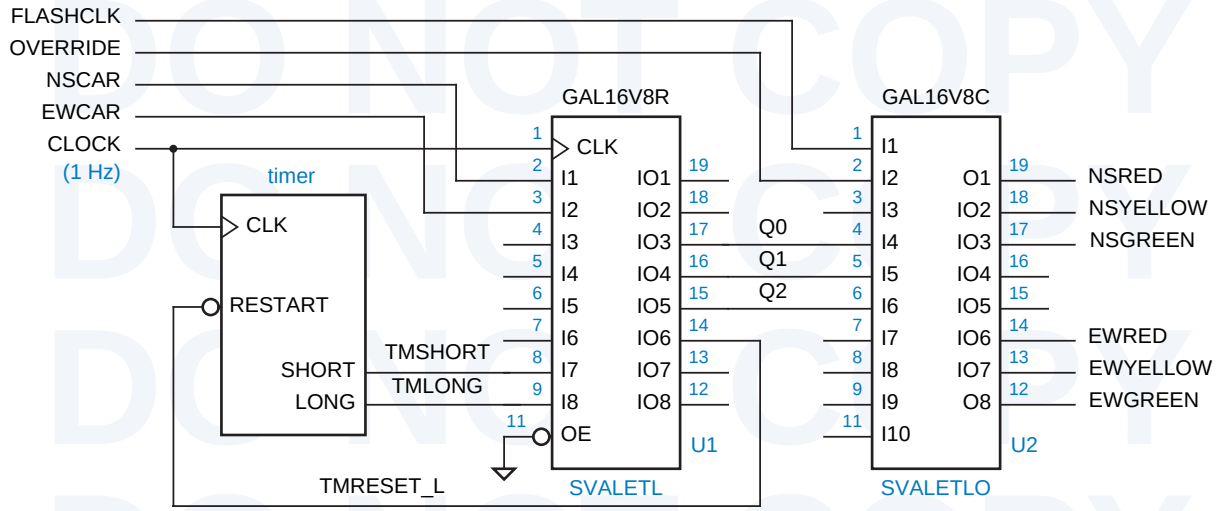

Figure 9-6 Sunnyvale traffic-light controller using two PLDs.

Table 9-10 Output logic for Sunnyvale traffic lights.

```
module svaletlo
title 'Output logic for Sunnyvale, CA, Traffic Lights'
"SVALETLO device 'P16V8C';

" Input pins
FLASHCLK, OVERRIDE, Q0, Q1, Q2   pin 1, 2, 4, 5, 6;

" Output pins
NSRED, NSYELLOW, NSGREEN         pin 19, 18, 17 istype 'com';
EWRED, EWYELLOW, EWGREEN         pin 14, 13, 12 istype 'com';

" Definitions (same as in state machine SVALETL)
...

equations

NSRED = !OVERRIDE & (LSTATE != NSGO) & (LSTATE != NSWAIT) & (LSTATE != NSWAIT2)
      # OVERRIDE & FLASHCLK;
NSYELLOW = !OVERRIDE & ((LSTATE == NSWAIT) # (LSTATE == NSWAIT2));
NSGREEN  = !OVERRIDE & (LSTATE == NSGO);

EWRED = !OVERRIDE & (LSTATE != EWGO) & (LSTATE != EWWAIT) & (LSTATE != EWWAIT2)
      # OVERRIDE & FLASHCLK;
EWYELLOW = !OVERRIDE & ((LSTATE == EWWAIT) # (LSTATE == EWWAIT2));
EWGREEN  = !OVERRIDE & (LSTATE == EWGO);

end svaletlo
```

The ABEL program in Table 9-9 defines only the state variables and one registered Moore output for the traffic controller. Six more Moore outputs are needed for the lights, more than remain on the 16V8. Therefore, a separate combinational PLD is used for these outputs, yielding the complete design shown in Figure 9-6 on the preceding page. An ABEL program for the output PLD is given in Table 9-10. We've taken this opportunity to add an OVERRIDE input to the controller. This input may be asserted by the police to disable the controller and put the signals into a flashing-red mode (at a rate determined by FLASHCLK), allowing them to manually clear up the traffic snarls created by this wonderful invention.

A traffic-light state machine including output logic can be built in a single 16V8, shown in Figure 9-7, if we choose an output-coded state assignment. Only the definitions in the original program of Table 9-9 must be changed, as shown in Table 9-11. This PLD does not include the OVERRIDE input and mode, which is left as an exercise (1.30).

Table 9-11 Definitions for Sunnyvale traffic-lights machine with output-coded state assignment.

```
module svaletlb
title 'Output-Coded State Machine for Sunnyvale Traffic Lights'
"SVALETLB device 'P16V8R';

" Input and output pins
CLOCK, !OE                          pin 1, 11;
NSCAR, EWCAR, TMSHORT, TMLONG       pin 2, 3, 8, 9;
NSRED, NSYELLOW, NSGREEN            pin 19, 18, 17 istype 'reg';
EWRED, EWYELLOW, EWGREEN            pin 16, 15, 14 istype 'reg';
TMRESET_L, XTRA                     pin 13, 12 istype 'reg';

" Definitions
LSTATE  = [NSRED,NSYELLOW,NSGREEN,EWRED,EWYELLOW,EWGREEN,XTRA]; " State vars
NSGO    = [   0,       0,      1,    1,       0,      0,   0]; " States
NSWAIT  = [   0,       1,      0,    1,       0,      0,   0];
NSWAIT2 = [   0,       1,      0,    1,       0,      0,   1];
NSDELAY = [   1,       0,      0,    1,       0,      0,   0];
EWGO    = [   1,       0,      0,    0,       0,      1,   0];
EWWAIT  = [   1,       0,      0,    0,       1,      0,   0];
EWWAIT2 = [   1,       0,      0,    0,       1,      0,   1];
EWDELAY = [   1,       0,      0,    1,       0,      0,   1];
```