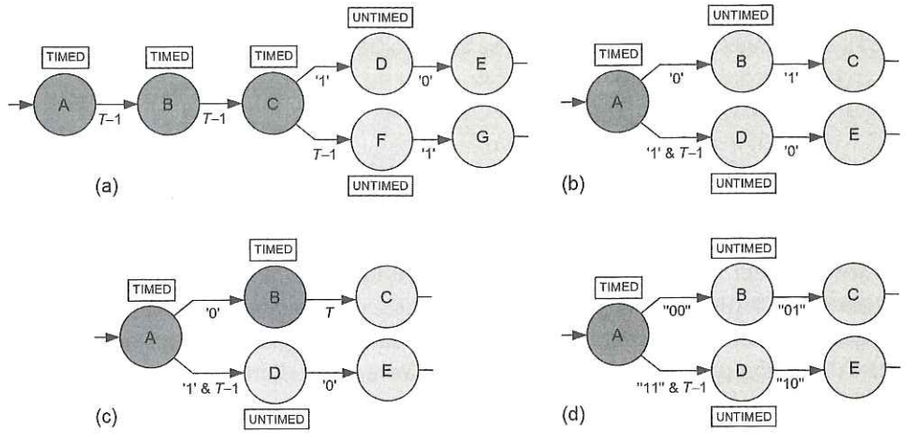

Figure 8.4
Four timed machines. (a) With only conditional and timed transitions. (b) With conditional and conditional-timed transitions but with state B untimed. (c) Same as b but with state B timed. (d) Same as b but with conditional values ("01" and "10") that might require the machine to remain in state A longer than T clock periods.

'1' means x = '1' and $T - 1$ means $t = T - 1$. As usual, it is assumed that a regular sequential counter running from $t = 0$ up to $t = T - 1$ is employed to build the timer.

8.5.1 Preliminary Analysis

A "tentative" strategy is assumed in this preliminary analysis, which consists of zeroing the timer after it reaches the monitored value ($t_{max} = T - 1$), with $t_{max} = 0$ adopted in the untimed states.

The machine in figure 8.4a has only conditional and timed transitions, so the timer always runs exactly up to t_{max}, after which the machine changes its state. Since it is assumed here that the timer is always zeroed after t_{max} occurs, the timer will always be cleared when the FSM enters a new state, causing it to work properly.

The machine in figure 8.4b has a conditional-timed transition. If x = '0' occurs before $t = t_{max}$, the machine moves from A to B with the timer at an unknown ($<t_{max}$) value. Consequently, the timer will not be zeroed here. However, because state B is untimed, so $t_{max} = 0$, the timer will be zeroed at the end of the first clock period after entering state B. As a result, the timer will be ready to operate properly even if state C is timed.

The case in figure 8.4c is similar to that in figure 8.4b, but state B is now timed. Because the machine will enter state B with $t < t_{max}$, the timer will span in state B only the number of clock cycles needed to complete state B's t_{max}. In summary, our tentative timer control strategy is not appropriate for this machine.

The case in figure 8.4d is also similar to that in figure 8.4b, with state B again untimed. However, note that there are values of x ("01" and "10") that might cause the machine to stay in state A even if t_{max} is reached. Because the timer is zeroed after t_{max} occurs, our tentative strategy does not work here either. A possible solution in this case is to stop the timer at t_{max}, zeroing it only when the machine changes state.

Based on the analysis above and that in section 8.4, two timer control strategies are proposed next.

8.5.2 Timer Control Strategy #1 (Generic)

A strategy that complies with all conditions described in section 8.4 and, consequently, with all conditions in the examples of figure 8.4, is summarized below.

For stopping the timer: Stop the timer when it reaches the monitored value (or a predefined value above that). Keep it so until the machine changes its state.
For zeroing the timer: Zero the timer whenever the machine changes state.

To apply the timer-zeroing technique above, we can compare *pr_state* to *nx_state*. If they are different, it means that the FSM will change its state at the next clock edge, so a flip-flop clearing command can be produced to zero the timer when such a transition occurs.

The advantages of this strategy are that it is generic, simple to understand, and simple to implement. The construction of state transition diagrams using it is simple and direct as well. Additionally, the timer does not need to be controlled in the untimed states because it will run only up to a certain value and will stop anyway, so power consumption is generally not a problem. Also, if one wants, a value greater than t_{max} can be employed (see comments in section 8.3.2), which can simplify the t-to-t_{max} comparator (recall that this comparator can be large; for example, to produce a 1 s delay from a 100 MHz clock, a 27-bit counter is needed); for instance, if T (= t_{max} + 1) is a power of 2, only a single bit (the MSB) needs to be monitored.

Its main disadvantage is that the *pr_state*-to-*nx_state* comparator can be a large circuit, because the number of bits in these two signals can be large, particularly when the number of states is high and one-hot encoding is employed (sequential or gray encoding is suggested when using strategy #1).

The following procedure is recommended: Use strategy #1, which is generic, to draw the state transition diagram. After completing it, check whether it complies with condition 1 or 2 described below for strategy #2. If it does, strategy #2 too can be used to build the timer.

There are only few cases in which strategy #1 cannot be applied completely, but the required adjustments are simple to handle. Such cases will be illustrated in sections 8.7 and 8.11.8.

When using VHDL or SystemVerilog, one of the following codes can be used to implement the timer using strategy #1. Note the use of $t \neq t_{max}$, which can be a slightly smaller comparator circuit than $t < t_{max}$, but either one is fine.

```
--Timer for strategy #1------------------------

--VHDL-----------------------------------------
process (clk, rst)
begin
   if rst='1' then
      t <= 0;
   elsif rising_edge(clk) then
      if pr_state /= nx_state then
         t <= 0;
      elsif t /= tmax then --see comment
         t <= t + 1;
      end if;
   end if;
end process;
-----------------------------------------------

--SystemVerilog---------------------------------
always_ff @(posedge clk, posedge rst)
   if (rst) t <= 0;
   else if (pr_state != nx_state) t <= 0;
   else if (t != tmax) t <= t + 1; --see comment
-----------------------------------------------
```

8.5.3 Timer Control Strategy #2 (Nongeneric)

This strategy is not generic because it cannot be employed in any timed machine. For example, it only works properly for machines a and b of figure 8.4. The procedure is summarized below.

For stopping the timer: Do not stop the timer.
For zeroing the timer: Zero the timer after it reaches $t_{max} = T - 1$. In the untimed states, adopt $t_{max} = 0$ (timer stopped at zero).

This strategy can be applied in the following cases:

1) To any timed machine without conditional-timed transitions (figure 8.4a, for example).
2) To timed machines with conditional-timed transitions but only if no state has more than one value for T, if no state can last longer than T clock periods, and if any transition that might last less than T clock cycles goes to an untimed state (figure 8.4b, for example).

The advantage of this strategy is that it avoids the *pr_state*-to-*nx_state* comparator, which can be a large circuit.

The disadvantages are that it is not generic and that the resulting circuit is not guaranteed to be smaller than that for strategy #1. Because here the value of t_{max} must be specified in all states (with $t_{max} = 0$ in the untimed states) when the machine has conditional-timed transitions, the t-to-t_{max} comparator (which also can be large) is more complex.

Since strategy #2 is not generic, the suggested procedure is to draw the state transition diagram using strategy #1, checking next if it complies with condition 1 or 2 above in order to determine whether strategy #2 can be used as well.

When using VHDL or SystemVerilog, one of the codes below can be used to implement the timer for strategy #2. Note the use of $t < t_{max}$ instead of $t \neq t_{max}$, needed to guarantee that the timer will be zeroed if the FSM leaves a timed state before the timer has reached t_{max} (entering therefore an untimed state). However, such a situation can only occur if the machine has conditional-timed transitions (figure 8.4b, for example); if the machine does not have conditional-timed transitions (figure 8.4a, for example), then $t \neq t_{max}$ is fine, too.

```
--Timer for strategy #2------------------------

--VHDL-----------------------------------------
process (clk, rst)
begin
   if rst='1' then
      t <= 0;
   elsif rising_edge(clk) then
      if t < tmax then --see comment
         t <= t + 1;
      else
         t <= 0;
      end if;
   end if;
end process;
-----------------------------------------------

--SystemVerilog---------------------------------
always_ff @(posedge clk, posedge rst)
   if (rst) t <= 0;
   else if (t < tmax) t <= t + 1; --see comment
   else t <= 0;
-----------------------------------------------
```

8.5.4 Time Behavior of Strategies #1 and #2

Figure 8.5 shows an example of FSM that in spite of having a conditional-timed transition can be implemented using any of the timer control strategies proposed above (note that this machine falls in the category depicted in figure 8.4b). The purpose of this example is to illustrate the differences in terms of time behavior between strategies #1 and #2.