11 Recursive (Category 3) State Machines

The name "recursive" for category 3 is due to the fact that when an output depends on a previous output value that value is generally from that same output, so a recursive

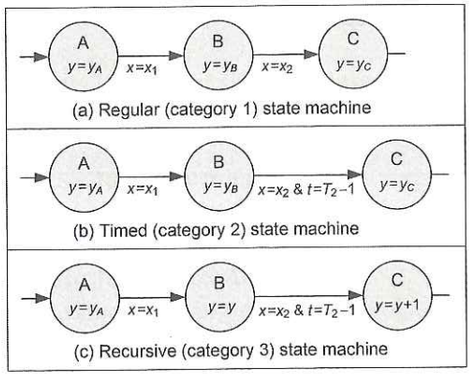

Figure 11.1
State machine categories (from a hardware perspective).

equation results (i.e., the output is a function of itself). For example, $y = y$, $y = y'$, and $y = y + 1$ mean that y (which is an output) should keep in the present state the same value that it had in the previous state, or the complement of that value, or the incremented version of that value, respectively. Equivalently, one could write $y_{new} = y_{old}$, $y_{new} = y_{old}'$, and $y_{new} = y_{old} + 1$. Occasionally, an output might be a function of a past value of another signal, like $y = z$ (same as $y_{new} = z_{old}$).

The two fundamental decisions that must be made before starting a design are then the following:

1) The state machine category (regular, timed, or recursive).
2) The state machine type (Moore or Mealy).

It is important to recall, however, that regardless of the machine category and type, the state transition diagram must fulfill three fundamental requisites (seen in section 1.3):

1) It must include all possible system states.
2) All state transition conditions must be specified (unless a transition is unconditional) and must be truly complementary.
3) The list of outputs must be exactly the same in all states (standard architecture).

11.2 Recursive (Category 3) State Machines

Figure 11.2 shows two examples of very special circuits. In figure 11.2a a simplified flowchart for a memory-write procedure is shown in which an address is set, the data to be stored at that address is presented, then a write-enable pulse is applied to store the data. Note the presence of an incrementer (gray block), responsible for setting the

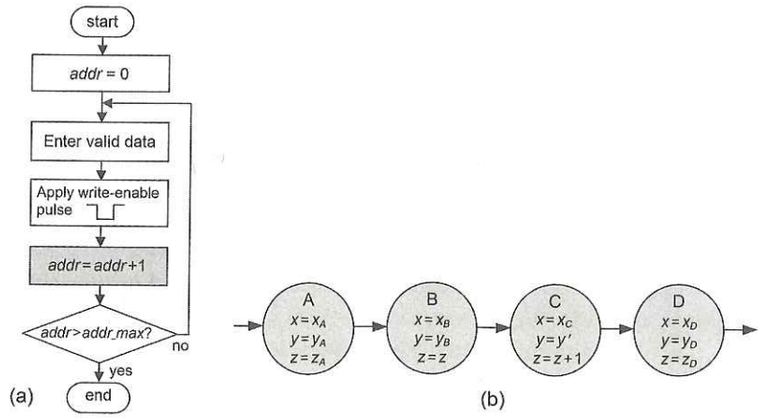

Figure 11.2
Examples of category 3 state machines.

next memory address. Because the expression *addr* = *addr* + 1 is not a constant but, rather, depends on the previous value of *addr*, this flowchart cannot be implemented in hardware without some sort of auxiliary memory (to hold the value of *addr*), which must be provided along with the corresponding FSM (note that this is different—and more complex—than a "similar" implementation in software).

The second example (figure 11.2b) consists of a state machine with three outputs. Note that the list of outputs is exactly the same in all states (as required for hardware implementations using the standard architecture; otherwise latches would be inferred), but again not all output values are deterministic: in state B, z must keep the same value that it had when the machine left state A; in state C, y must exhibit the complement of the value that it had in the previous state, while z must be incremented. Recall that we cannot simply write $z = z_A$ in state B because z_A might have changed; for the same reason, we cannot write $y = y_B'$ and $z = z_A + 1$ in state C. Consequently, an extra memory (to hold the values of y and z) is again needed.

11.3 Architectures for Recursive (Category 3) Machines

The general architecture for category 3 machines is summarized in figure 11.3a. This representation follows the style of figures 3.1b and 3.1d, but the style of figures 3.1a and 3.1c could be used equivalently. Note that the timer is optional, but at least one auxiliary register is necessary.

In this illustration, only for the signal that produces *output2* an auxiliary register is needed, so for that output the optional output register (figure 11.3b) is never required (the dashed lines indicate that *output2* can be either the unregistered or the registered

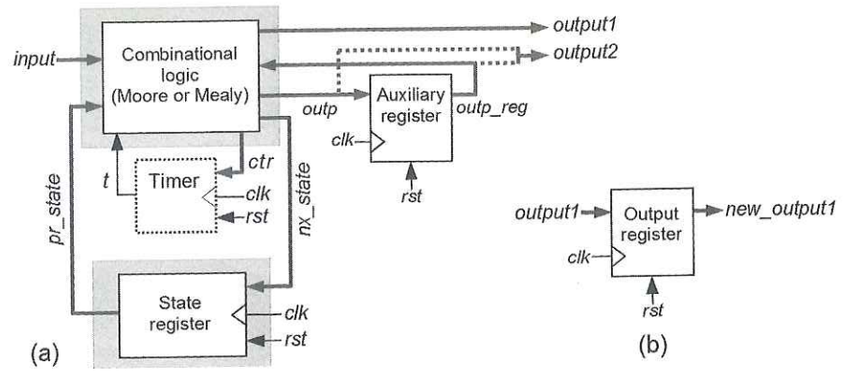

Figure 11.3
(a) General architecture for category 3 machines (timer is optional, but auxiliary register is compulsory). (b) Optional output register (only for outputs not processed by an auxiliary register).

version of *outp*). On the other hand, *output1* is not registered, so depending on the application, for it the optional output register might be needed. The resulting implementations are described below.

Recursive Moore machine: The circuit of figure 11.3a is used, with the input (if it exists) connected only to the logic block for the next state, as in figure 5.2a, and with unregistered output. Regarding the options for the output, see the comments above.

Recursive Mealy machine: Again, the circuit of figure 11.3a is used, but this time with the input connected to both logic blocks (for output and for next state), as in figure 5.2b. Regarding the options for the output, see the comments above.