can also be written as

```
(r_reg+1)=unsigned(m)
```

Since the r_req+1 operation is needed for incrementing operation, we can use it in comparison and eliminate the decrementor. The revised VHDL code is shown in Listing 8.16.

Listing 8.16 More efficient description of a programmable mod-m counter

```
architecture two_seg_effi_arch of prog_counter is
   signal r_reg: unsigned(3 downto 0);
   signal r_next, r_inc: unsigned(3 downto 0);
begin
   -- register
   process(clk,reset)
   begin
      if (reset='1') then
         r_reg <= (others=>'0');
      elsif (clk'event and clk='1') then
         r_reg <= r_next;
      end if;
   end process;
   -- next-state logic
   r_inc <= r_reg + 1;
   r_next <= (others=>'0') when r_inc=unsigned(m) else
             r_inc;
   -- output logic
   q <= std_logic_vector(r_reg);
end two_seg_effi_arch;
```

Note that we employ a separate statement for the shared expression:

```
r_inc <= r_reg + 1;
```

and use the r_inc signal for both comparison and incrementing. The diagram of the revised code is shown in Figure 8.14(b).

8.6 TIMING ANALYSIS OF A SYNCHRONOUS SEQUENTIAL CIRCUIT

The timing of a combinational circuit is characterized primarily by the propagation delay, which is the time interval required to generate a stable output response from an input change. The timing characteristic of a sequential circuit is different because of the constraints imposed by memory elements. The major timing parameter in a sequential circuit is the *maximal clock rate*, which embeds the effect of the propagation delay of the combination circuit, the clock-to-q delay of the register and the setup time constraint of the register. Other timing issues include the condition to avoid hold time violation and I/O-related timing parameters.

8.6.1 Synchronized versus unsynchronized input

Satisfying the setup and hold time constraints is the most crucial task in designing a sequential circuit. One motivation behind synchronous design methodology is to group all FFs together and control them with the same clock signal. Instead of considering the constraints

of tens or hundreds of FFs, we can treat them as one memory component and deal with the timing constraint of a *single* register.

The conceptual diagram of Figure 8.5 can be considered as a simplified block diagram for all synchronous sequential circuits. In this diagram, FFs and registers are grouped together as the state register. The input of this register is the `state_next` signal. It is generated by next-state logic, which is a combinational logic with two inputs, including the external input and the output of the state register, `state_reg`. To study the timing constraint of the state register, we need to examine the impact of the two inputs of the next-state logic. Our discussion considers the following effects:

- The effect of the `state_reg` signal.
- The effect of *synchronized* external input.
- The effect of *unsynchronized* external input.

Since the `state_reg` signal is the output of the state register, it is synchronized by the same clock. A closed feedback loop is formed in the diagram through this signal. The timing analysis of a synchronous sequential circuit focuses mainly on this loop and is discussed in Section 8.6.2 .

A *synchronized* external input means that the generation of the input signal is controlled by the same clock signal, possibly from a subsystem of the same design. The timing analysis is somewhat similar to the closed-loop analysis describe above, and is discussed in Section 8.6.5.

An *unsynchronized* external input means that the input signal is generated from an external source or an independent subsystem. Since the system has no information about the unsynchronized external input, it cannot prevent timing violations. For this kind of input, we must use an additional *synchronization circuit* to synchronize the signal with the system clock. This issue is be discussed in Chapter 16.

8.6.2 Setup time violation and maximal clock rate

In Figure 8.5, the output of the register is processed via next-state logic, whose output becomes the new input to the register. To analyze the timing, we have to study the operation of this closed feedback loop and examine the `state_reg` and `state_next` signals. The `state_reg` signal is the output of the register, and it also serves as the input to the next-state logic. The `state_next` signal is the input of the register, and it is also the output of the next-state logic.

Maximal clock rate The timing diagram in Figure 8.15 shows the responses of the `state_reg` and `state_next` signals during one clock cycle. At time t_0, the clock changes from '0' to '1'. We assume that the `state_next` signal has stabilized and doesn't change within the setup and hold time periods. After the clock-to-q delay (i.e., T_{cq}), the register's output, `state_reg`, becomes available at time t_1, which is $t_0 + T_{cq}$. Since `state_reg` is the input of the next-state logic, the next-state logic responds accordingly. We define the propagation delays of the fastest and slowest responses as $T_{next(min)}$ and $T_{next(max)}$ respectively. In the timing diagram, the `state_next` signal changes at t_2, which is $t_1 + T_{next(min)}$, and becomes stabilized at t_3, which is $t_1 + T_{next(max)}$. At time t_5, a new rising clock edge arrives and the current clock cycle ends. The `state_next` is sampled at t_5 and the process repeats again. t_5 is determined by the period (T_c) of the clock signals, which is $t_0 + T_c$.

Now let us examine the impact of the setup time constraint. The setup time constraint indicates that the `state_next` signal must be stabilized at least T_{setup} before the next

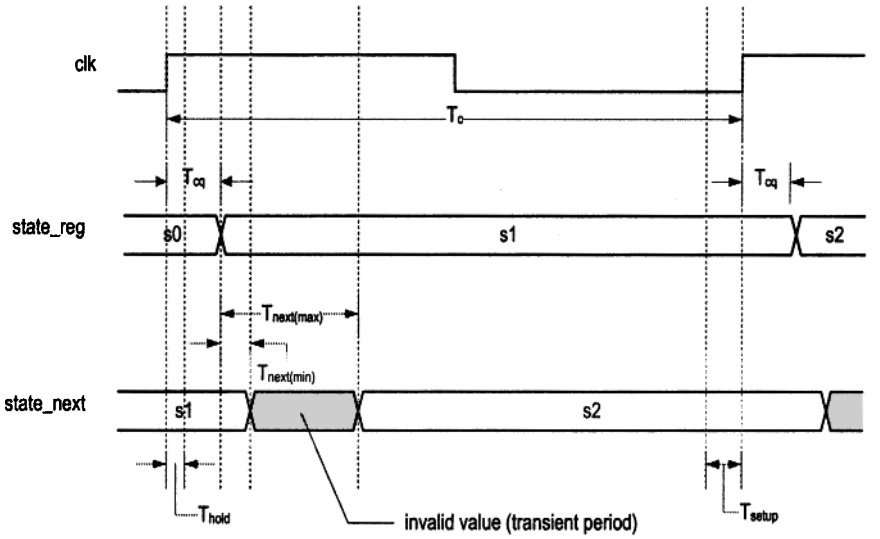

Figure 8.15 Timing analysis of a basic sequential circuit.

sampling edge at t_5. This point is labeled t_4 in the timing diagram. To satisfy the setup time constraint, the `state_next` signal must be stabilized before t_4. This requirement translates into the condition

$$t_3 < t_4$$

From the timing diagram, we see that

$$t_3 = t_0 + T_{cq} + T_{next(max)}$$

and

$$t_4 = t_5 - T_{setup} = t_0 + T_c - T_{setup}$$

We can rewrite the inequality equation as

$$t_0 + T_{cq} + T_{next(max)} < t_0 + T_c - T_{setup}$$

which is simplified to

$$T_{cq} + T_{next(max)} + T_{setup} < T_c$$

This shows the role of the clock period on a sequential circuit. To avoid setup time violation, the minimal clock period must be

$$T_{c(min)} = T_{cq} + T_{next(max)} + T_{setup}$$

The clock period is the main parameter to characterize the timing and performance of a sequential circuit. We commonly use the maximal clock rate or frequency, the reciprocal of the minimal period, to describe the performance of a sequential circuit, as in a 500-MHz counter or 2-GHz processor.

Clock rate examples For a given technology, the T_{cq} and T_{setup} of a D FF are obtained from the data sheet. We can determine the maximal clock rate of a sequential circuit once the propagation delay of the next-state logic is known. This information can only be determined after synthesis and placement and routing. However, we can calculate and estimate the rate of some simple examples.

Assume that we use the technology discussed in Section 6.2.6, and T_{cq} and T_{setup} of its D FF cell are 1 and 0.5 ns respectively. The delay information of combinational components can be obtained from Table 6.2. Let us first consider the free-running shift register of Section 8.5.2. The next-state logic of the shift register only involves the routing of the input and output signals. If we assume that the wiring delay is negligible, its propagation delay is 0. The minimal clock period and maximal clock rate become

$$T_{c(min)} = T_{cq} + T_{setup} = 1.5 \text{ ns}$$

$$f_{max} = \frac{1}{T_{cq} + T_{setup}} = \frac{1}{1.5 \text{ ns}} \approx 666.7 \text{ MHz}$$

Clearly, this is the maximal clock rate that can be achieved with this particular technology.

The second example is an 8-bit free-running binary counter, similar to the 4-bit version of Section 8.5.4. The next-state logic of this circuit is the incrementor, as shown in Figure 8.12. If we choose the incrementor that is optimized for area, the clock rate for this 8-bit binary counter is

$$f_{max} = \frac{1}{T_{cq} + T_{8\_bit\_inc(area)} + T_{setup}} = \frac{1}{1 \text{ ns} + 2.4 \text{ ns} + 0.5 \text{ ns}} \approx 256.4 \text{ MHz}$$

If we increase the size of the counter, a wider incrementor must be utilized, and the propagation delay of the incrementor is increased accordingly. The clock rate of a 16-bit binary counter is reduced to

$$f_{max} = \frac{1}{T_{cq} + T_{16\_bit\_inc(area)} + T_{setup}} = \frac{1}{1 \text{ ns} + 5.5 \text{ ns} + 0.5 \text{ ns}} \approx 142.9 \text{ MHz}$$

and the clock rate of a 32-bit counter is reduced to

$$f_{max} = \frac{1}{T_{cq} + T_{32\_bit\_inc(area)} + T_{setup}} = \frac{1}{1 \text{ ns} + 11.6 \text{ ns} + 0.5 \text{ ns}} \approx 76.3 \text{ MHz}$$

To increase the performance of a binary counter, we must reduce the value of $T_{cq} + T_{next(max)} + T_{setup}$. Since T_{cq} and T_{setup} are determined by the intrinsic characteristics of FFs, they cannot be altered unless we switch to a different device technology. The only way to increase performance is to reduce the propagation delay of the incrementor. If we replace the incrementors that are optimized for delay, the clock rates of the 8-, 16- and 32-bit binary counters are increased to

$$f_{max} = \frac{1}{T_{cq} + T_{8\_bit\_inc(delay)} + T_{setup}} = \frac{1}{1 \text{ ns} + 1.5 \text{ ns} + 0.5 \text{ ns}} \approx 333.3 \text{ MHz}$$

$$f_{max} = \frac{1}{T_{cq} + T_{16\_bit\_inc(delay)} + T_{setup}} = \frac{1}{1 \text{ ns} + 3.3 \text{ ns} + 0.5 \text{ ns}} \approx 208.3 \text{ MHz}$$

and

$$f_{max} = \frac{1}{T_{cq} + T_{32\_bit\_inc(delay)} + T_{setup}} = \frac{1}{1 \text{ ns} + 7.5 \text{ ns} + 0.5 \text{ ns}} \approx 111.1 \text{ MHz}$$

respectively.

8.6.3 Hold time violation

The impact of the hold time constraint is somewhat different from the setup time constraint. Hold time, T_{hold}, is the time period that the input signal must be stabilized after the sampling edge. In the timing diagram of Figure 8.15, it means that the `state_next` must be stable between t_0 and t_h, which is $t_0 + T_{hold}$. Note that the earliest time that `state_next` changes is at time t_2. To satisfy the hold time constraint, we must ensure that

$$t_h < t_2$$

From the timing diagram, we see that

$$t_2 = t_0 + T_{cq} + T_{next(min)}$$

and

$$t_h = t_0 + T_{hold}$$

The inequality becomes

$$t_0 + T_{hold} < t_0 + T_{cq} + T_{next(min)}$$

which is simplified to:

$$T_{hold} < T_{cq} + T_{next(min)}$$

$T_{next(min)}$ depends on the complexity of next-state logic. In some applications, such as the shift register, the output of one FF is connected to the input of another FF, and the propagation delay of the next-state logic is the wire delay, which can be close to 0. Thus, in the worst-case scenario, the inequality becomes

$$T_{hold} < T_{cq}$$

Note that both parameters are the intrinsic timing parameters of the FF, and the inequality has nothing to do with the next-state logic. Manufacturers usually guarantee that their devices satisfy this condition. Thus, we need not worry about the hold time constraint unless the clock edge cannot arrive at all FFs at the same time. We discuss this issue in Chapter 16.

8.6.4 Output-related timing considerations

The closed feedback diagram in Figure 8.5 is the core of a sequential system. In addition, there are also external inputs and outputs. Let us first consider the output part of the circuit. The output signal of a sequential circuit can be divided into the *Moore-typed output* (or just *Moore output*) and *Mealy-typed output* (or just *Mealy output*). For Moore output, the output signal is a function of *system state* (i.e., the output of the register) only. On the other hand, for Mealy output, the output signal is a function of *system state and the external input.* The two types of output can coexist, as shown in Figure 8.16. The main timing parameter for both types of outputs is T_{co}, the time required to obtain a valid output signal after the rising edge of the clock. The value of T_{co} is the summation of T_{cq} and T_{output} (the propagation delay of the output logic); that is,

$$T_{co} = T_{cq} + T_{output}$$

For Mealy output, there exists a path in which the input can affect the output directly. The propagation delay from input to output is simply the combinational propagation delay of output logic.

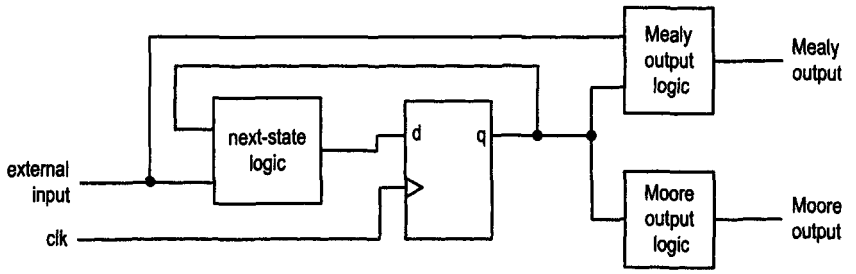

Figure 8.16 Output circuits of a sequential circuit.

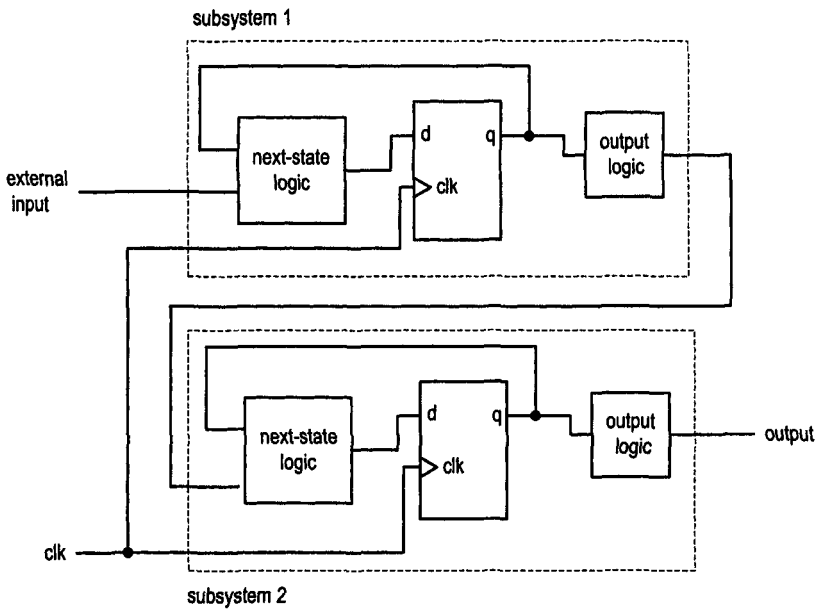

Figure 8.17 Input timing of two synchronous subsystems.

8.6.5 Input-related timing considerations

In a large design, a system may contain several synchronous subsystems. Thus, it is possible that an input comes from a subsystem that is controlled and synchronized by the same clock. The block diagram of this situation is shown in Figure 8.17. Note that the two subsystems are controlled by the same clock and thus are synchronous. At the rising edge of the clock, the register of subsystem 1 samples a new input value. After $T_{co(system1)}$, its new output, which is the input for the next-state logic of subsystem 2, becomes available. At this point the timing analysis is identical to that in Section 8.6.2. To avoid setup time violation, the timing of the two circuits must satisfy the following condition:

$$T_{co(system1)} + T_{next(max)} + T_{setup} < T_c$$

Note that $T_{next(max)}$, the propagation delay of next-state logic, is somewhat different from the calculation used in Section 8.6.2. The $T_{next(max)}$ here is the propagation delay

from the external input to `state_next`, whereas $T_{next(max)}$ used in earlier minimal clock period calculation in Section 8.6.2 is the propagation delay from the internal register output (i.e., `state_reg`) to `state_next`. To be more accurate, we should separate the two constraints. The constraint for the closed loop is

$$T_{cq} + T_{next(max\ of\ state\_reg-to-state\_next)} + T_{setup} < T_{c1}$$

and the constraint for the external input is

$$T_{co(system1)} + T_{next(max\ of\ ext\_input-to-state\_next)} + T_{setup} < T_{c2}$$

We usually determine the clock period based on the calculation of T_{c1}. If T_{c2} turns out to be greater than T_{c1}, we normally redesign the I/O buffer rather than slowing down the clock rate of the entire system. For example, we can employ an extra input buffer for the external input of subsystem 2. Although this approach delays the external input by one clock cycle, it reduces the $T_{co(system1)}$ to T_{cq} in the second constraint.

8.7 ALTERNATIVE ONE-SEGMENT CODING STYLE

So far, all VHDL coding follows the basic block diagram of Figure 8.5 and separates the memory elements from the rest of the logic. Alternatively, we can describe the memory elements and the next-state logic in a single process segment. For a simple circuit, this style appears to be more compact. However, it becomes involved and error-prone for more complex circuits. In this section, we use some earlier examples to illustrate the one-segment VHDL description and the problems associated with this style.

8.7.1 Examples of one-segment code

D FF with enable Consider the D FF with an enable signal in Listing 8.7. It can be rewritten in one-segment style, as in Listing 8.17.

Listing 8.17 One-segment description of a D FF with enable

```
architecture one_seg_arch of dff_en is
begin
   process(clk,reset)
   begin
      if (reset='1') then
         q <='0';
      elsif (clk'event and clk='1') then
         if (en='1') then
            q <= d;
         end if;
      end if;
   end process;
end one_seg_arch;
```

The code is similar to a regular D FF except that there is an if statement inside the elsif branch:

```
if (en='1') then
   q <= d;
end if;
```