the state sequences in Fig. 9.20b we find logic relation like $Y = QR'$ or $Y = RS'$ or $Y = ST'$, etc. can also be used for decoding purpose as they generate $Y = 1$ only once during $2N$ clock cycles. Note that for ring counter we don't need any decoding gate and clock pulse count can directly be obtained from any one flip-flop output. We shall discuss other counter design techniques in Chapter 10, which require less number of flip-flops for a particular modulo number. But, there decoding complexity increases with increasing number of flip-flops. For example, a modulo-8 counter is possible to design with $\log_2 8 = 3$ number of flip-flops but we need a 3 input AND gate to decode the counter. Similarly, modulo-16 counter requires 4 flip-flops and 4 input AND gate for decoding.

There is another important issue related with ring counter and switched tail counter. An n-bit register has 2^n different combination of states. But, the counter is to be initialized with one of the valid state of the counting sequence on which the design is based. Otherwise, the counter will follow a completely different state sequence (mutually exclusive) and decoding will not be proper. Solve problem 9.25 to get an idea on what happens if circuit in Fig. 9.20a is initialized with a word outside the state sequence appearing in Fig. 9.20b.

Sequence Generator and Sequence Detector

Sequence generator is useful in generating a sequence pattern repetitively. It may be the synchronizing bit pattern sent by a digital data transmitter or it may be a control word directing repetitive control task. Sequence detector checks binary data stream and generates a signal when a particular sequence is detected.

Figure 9.21a gives the basic block diagram of a sequence generator where shift register is presented as pipe full of data and each flip-flop represents one compartment of it. The leftmost flip-flop is connected to serial data in and rightmost provides serial data out. The clock is implied and data transfer takes place only when a clock trigger arrives. Note that the shift register is connected like a ring counter and with triggering of clock the binary word stored in the clock comes out sequentially from serial out but does not get lost as it is fed back as serial in to fill the register all over again. Sequence generated for binary word 1011 is shown in the figure and for any n-bit long sequence to be generated for this configuration we need to store the sequence in an n-bit shift register.

The circuit that can detect a 4-bit binary sequence is shown in Fig. 9.21b. It has one register to store the binary word we want to detect from the data stream. Input data stream enters a shift register as serial data in and leaves as serial out. At every clocking instant, bit-wise comparisons of these two registers are done through Ex-NOR gate as shown in the figure. Two input Ex-NOR gives logic high when both inputs are low or both of them are high i.e. when both are equal. The final output is taken from a four input AND gate, which becomes 1 only when all its inputs are 1, i.e. all the bits are matched. Figure 9.21b shows a situation when data received so far is 0111 and word to be matched is 1011. The first two bits are mismatched and corresponding Ex-NOR outputs are low, so also final output Y. Now, as the next bit in the serial data stream is 1 when a clock trigger comes the first flip-flop of the shift-register stores 1 and 011 gets shifted to 2nd to 3rd flip-flops. With this both registers store 1011 and the first flip-flop of the shift-register stores 1 and 011 gets shifted to 2nd to 3rd flip-flops and $Y = 1$ completing sequence detection.

Note that Fig. 9.21b can be used as a *programmable sequence detector*, i.e. if we want to change the binary word to be detected we simply load that in the bottom register. For a fixed sequence detector, we can reduce hardware cost by removing bottom register and directly connect

Ex-NOR input to $+V_{CC}$ or GND depending on whether we need a 1 or a 0 to be detected in a particular position.

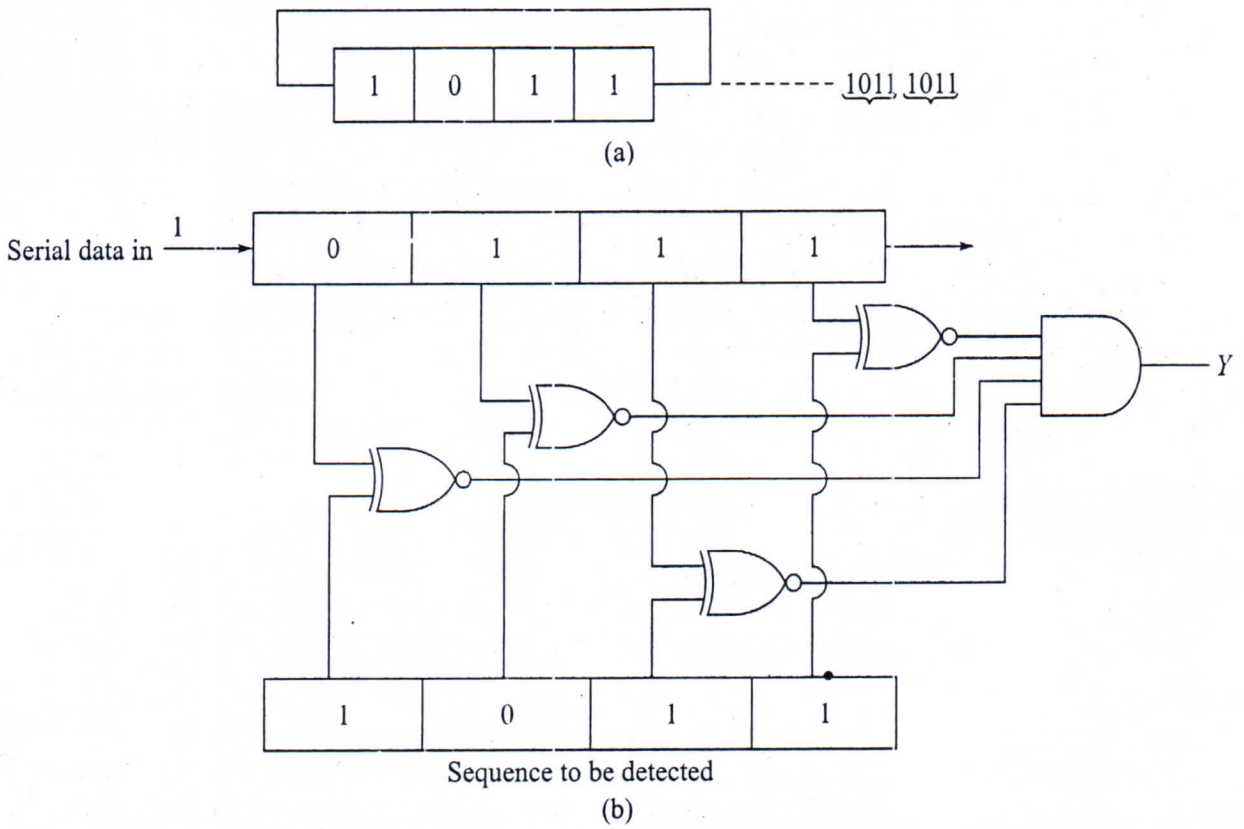

Fig. 9.21 *(a) 4-bit sequence generator. (b) 4-bit programmable sequence detector.*

Serial Adder

The addition operation and full adder (FA) circuit is discussed in detail in Chapter 6. We have seen for 8-bit addition we need 8 FA units (Fig. 6.6). There the addition is done in parallel. Using shift register we can convert this parallel addition to serial one and reduce number of FA units to only one. The benefit of this technique is more pronounced if the hardware unit that's needed to be used in parallel is very costly. Figure 9.22 shows how serial addition takes place in a time-multiplexed manner and also provides a snapshot of the register values at 3rd clock cycle.

Two 8-bit numbers, to be added ($A_7A_6...A_1A_0$ and $B_7B_6...B_1B_0$) are loaded in two 8-bit shift registers A and B. The LSB of each number appears in the rightmost position in two registers. Serial data out of A and B are fed to data inputs of full adder. The carry-in is fed from its own carry output delayed by one clock period by a D flip-flop, which is initially cleared. Both registers and D flip-flop are triggered by same clock. The sum (S) output of FA is fed to serial data in of Shift Register A.

The serial addition takes place like this. The LSBs of two numbers (A_0 and B_0) appearing at serial out of respective registers are added by FA during 1st clock cycle and generate sum (S_0) and carry (C_0). S_0 is available at serial data input of register A and C_0 at input of D flip-flop. At NT of clock shift registers shift its content to right by one unit. S_0 becomes MSB of A and C_0 appears at D flip-flop output. Therefore in the second clock cycle FA is fed by second bit (A_1 and B_1) of two numbers and previous carry (C_0). In second clock cycle, S_1 and C_1 are generated and made available at serial