plemented with a *linear feedback shift register*—a shift register with feedback generated by a set of exclusive-OR gates. The placement of the XOR feedback terms is mathematically defined by a binary polynomial. Figure 9.5 shows scrambling logic used to encode and decode eight-bit data words using the function $F(X) = X^7 + X^4 + 1$. The mathematical theory behind such polynomials is based on *Galois fields*, discovered by Evariste Galois, a nineteenth century French mathematician. XOR gates are placed at each bit position specified by the polynomial exponents, and their outputs feed back to the shift register input to scramble and feed forward to the output to descramble.

This type of scrambling should not be confused with more sophisticated security and data protection algorithms. Data scrambled in this manner is done so for purposes of randomizing the bits on the communications channel to achieve an average DC value of 0. Polynomial scrambling works fairly well and is relatively easy to implement, but such schemes are subject to undesired cases in which the application of select repetitive data patterns can cause an imbalance in the number of 1s and 0s, thereby reducing the benefit of scrambling. The probability of settling into such cases is low, making scrambling a suitable coding mechanism for certain data links.

While shown schematically as a serial process, these algorithms can be converted to parallel logic by accumulating successive XOR operations over eight bits shifted through the polynomial register. In cases when the coding logic lies outside of the serdes in custom logic, it is necessary to convert this serial process into a parallel one, because data coming from the serdes will be in parallel form at a corresponding clock frequency. Working out the logic for eight bits at a time allows processing one byte per clock cycle. The serial to parallel algorithm conversion can be done over any number of bits that is relevant to a particular application. This process is conceptually easy, but it is rather tedious to actually work out the necessary logic.

A table can be formed to keep track of the polynomial code vector, C[6:0], and the output vector, Q[7:0], as functions of the input vector, D[7:0]. Table 9.1 shows the state of C and Q, assuming that the least-significant bit (LSB) is transmitted first, during each of eight successive cycles by listing terms that are XORed together.

The final column, D[7], and the bottom row, Q, indicate the final state of the code and output vectors, respectively. The code vector terms can be simplified, because some iterative XOR feedback terms cancel each other out as a result of the identity that $A \oplus A = 0$. Q[7:0] can be taken directly from the table because there are no duplicate XOR terms. The simplified code vector, C[6:0], is shown in Table 9.2.

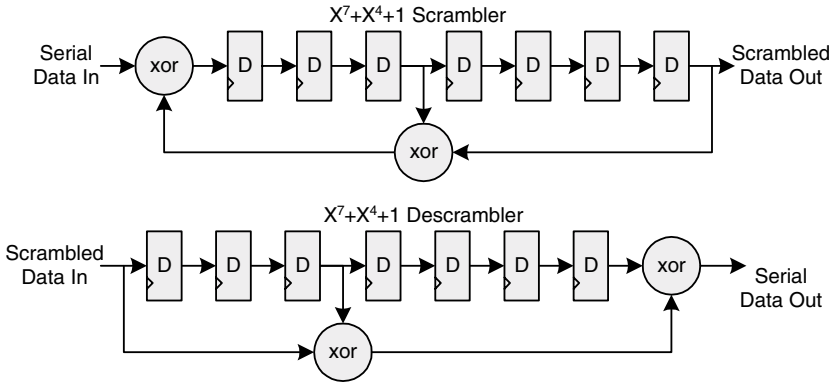

FIGURE 9.5 Eight-bit scrambling/descrambling logic.

TABLE 9.1 Scrambler Logic State Table

Code/Output Vector Bits	Input Vector Bits, One Each Clock Cycle							
	D0	D1	D2	D3	D4	D5	D6	D7
C6	C0 D0 C4	C1 D1 C5	C2 D2 C6	C3 D3 C0 D0 C4	C4 D4 C1 D1 C5	C5 D5 C2 D2 C6	C6 D6 C3 D3 C0 D0 C4	C0 D0 C4 D7 C4 D4 C1 D1 C5
C5	C6	C0 D0 C4	C1 D1 C5	C2 D2 C6	C3 D3 C0 D0 C4	C4 D4 C1 D1 C5	C5 D5 C2 D2 C6	C6 D6 C3 D3 C0 D0 C4
C4	C5	C6	C0 D0 C4	C1 D1 C5	C2 D2 C6	C3 D3 C0 D0 C4	C4 D4 C1 D1 C5	C5 D5 C2 D2 C6
C3	C4	C5	C6	C0 D0 C4	C1 D1 C5	C2 D2 C6	C3 D3 C0 D0 C4	C4 D4 C1 D1 C5
C2	C3	C4	C5	C6	C0 D0 C4	C1 D1 C5	C2 D2 C6	C3 D3 C0 D0 C4
C1	C2	C3	C4	C5	C6	C0 D0 C4	C1 D1 C5	C2 D2 C6
C0	C1	C2	C3	C4	C5	C6	C0 D0 C4	C1 D1 C5
Q	C0	C1	C2	C3	C4	C5	C6	C0 D0 C4

TABLE 9.2 Simplified Scrambling Code Vector Logic

Code Vector Bits	XOR Logic
C6	C0 D0 C1 D1 D4 C5 D7
C5	C0 D0 C3 D3 C4 C6 D6
C4	C2 D2 C5 D5 C6
C3	C1 D1 C4 D4 C5
C2	C0 D0 C3 D3 C4
C1	C2 D2 C6
C0	C1 D1 C5

Following the same process, code and output vectors for the associated descrambling logic can be derived as well, the results of which are shown in Table 9.3. In this case, the code vector, C[6:0], is easy, because it is simply the incoming scrambled data shifted one bit at a time without any XOR feedback terms. The output vector, Q[7:0], is also easier, because the XOR logic feeds forward without any iterative feedback terms.