**TABLE 9.3 Descrambling Logic Code and Output Vector Logic**

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

Data that has been encoded must be decoded before framing information can be extracted. Therefore, the serdes' receiving shift register must begin by performing a simple serial-to-parallel conversion until framing information can be extracted after decoding. Prior to the detection of framing information, the output of the parallel shift register will be arbitrary data at an arbitrary alignment, because there is no knowledge of where individual bytes of words begin and end in the continuous data stream. Once a framing sequence has been detected, the shift register can be "snapped" into correct alignment, and its output will be properly formatted whole bytes or words.

When reconstructing decoded data, the desired byte alignment will likely span two consecutive bytes as they come straight from the descrambling logic. The most significant bits of a descrambled byte logically follow the LSB of the next descrambled byte because of the order in which bits are shifted through the scrambler and descrambler. Therefore, when data arrives misaligned at the receiver/descrambler, bytes are reassembled by selecting the correct bits from the most significant bits (MSB) of descrambled byte N and the LSB of descrambled byte N + 1 as shown in Fig. 9.6.

Framing information is not conveyed by the scrambled coding and must therefore be extracted at a higher level. For this reason, certain serdes components that are used in scrambled coding systems

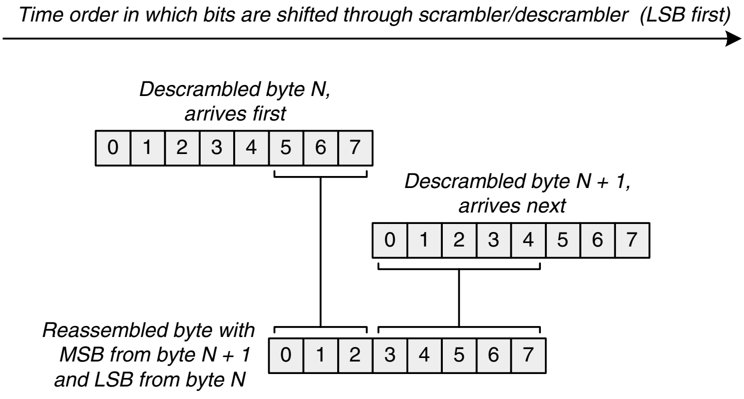
Time order in which bits are shifted through scrambler/descrambler (LSB first)
Descrambled byte N, arrives first
0 1 2 3 4 5 6 7
Descrambled byte N + 1, arrives next
0 1 2 3 4 5 6 7
Reassembled byte with MSB from byte N + 1 and LSB from byte N
0 1 2 3 4 5 6 7

**FIGURE 9.6** Reassembly of descrambled data.

do not support any descrambling or framing detection logic, because that logic is application specific. These functions must be implemented by external data processing logic. There are specific cases in which a common class of applications use standard scrambling and framing techniques. Serdes ICs designed for these applications do support the decoding and framing functions. An example of such an application is the transport of serial digital video in a TV studio setting. Companies such as Gennum Corporation manufacture serdes components with logic that can descramble 10-bit video data words and recognize framing sequences so that the resulting data stream is cleanly word aligned and directly usable without further manipulation.

## 9.5 8B10B CODING

An alternative, more robust coding method is *8B10B coding,* which maps byte values to 10-bit codes. Unlike polynomial coding that operates on a continuous stream of bits, 8B10B processes each byte individually. Each coded value is guaranteed to contain either five 0s and five 1s, four 0s and six 1s, or six 0s and four 1s. Therefore, a coded value contains a 0/1 imbalance of at most one bit. Some raw byte values are mapped to a perfectly balanced coded value. Those that map to imbalanced codes are given two mappings: one with four 0s and six 1s and one with six 0s and four 1s. A concept called c*urrent running disparity* (CRD) is used to keep track of whether the last code contained a positive (1) or negative (0) imbalance. If a byte passes through the 8B10B encoder and is mapped to a code with four 0s and six 1s, a CRD state bit is set. The next time a byte passes through the encoder that has two possible mappings, the negatively imbalanced code will be chosen, and CRD will be cleared. This mechanism ensures that there are an equal number of 1s and 0s over time.

8B10B guarantees a minimum frequency of 0/1/0 transitions within the coded data stream, enabling reliable recovery of the serial bit clock at the receiver and guaranteeing an average DC value of 0 across the data stream. In mapping the 256 unique byte values to 10-bit codes, not all 1,024 code words are used. Some of these words are undesirable, because they contain greater than a single bit of 0/1 imbalance. However, some code words are left over that contain valid sequences of 0s and 1s. Rather than leave them unused, these code words are used to carry special values called *special characters* that can assist in the framing of data on a link. In particular, three special characters contain a unique *comma pattern*, where the first seven bits of a word contain two bits of one polarity followed by five bits of the opposite polarity. This comma pattern is guaranteed not to occur within any other data words, making it ideal as a marker with which data can be aligned within a serdes. A serdes that is 8B10B coding aware can search for comma patterns in the data stream and then realign the data stream such that comma patterns show up in the seven most significant bits of an incoming word. If the data link is properly encoded and is sufficiently free of disruptive noise, this alignment process should have to occur only once and thereafter will be transparent, because the comma patterns will already be in the proper bit positions. The benefits of 8B10B coding come with a 20 percent overhead penalty, because every eight data bits require ten coded bits. The cost is justified in many communications systems, because the electrical benefits of a balanced coding scheme enable more usable bandwidth to be extracted from a medium with a corresponding low bit error rate.

Serdes devices that are 8B10B coding aware contain a framing detection logic block that implements the 8B10B comma alignment function. The actual encoding and decoding may or may not be implemented in the serdes according to the parameters of the specific device. The comma detection and alignment function does not require the more complex encoding/decoding functions, because only a simple pattern match is required. A benefit of 8B10B coding is that low-level alignment operations can be performed by generic serdes logic without regard to the actual type of network traffic passing through the device. For example, Gigabit Ethernet and Fibre Channel (a storage area net-