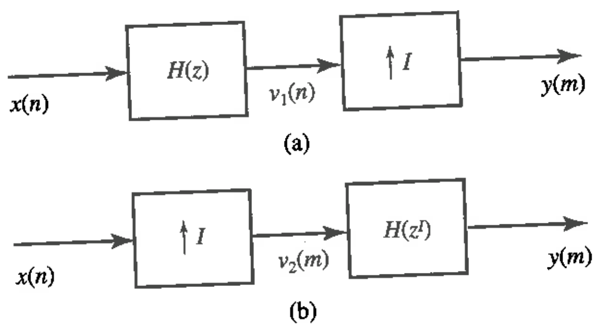


**Figure 11.5.4** Two equivalent upsampling systems (second noble identity).

A similar identity can be shown to hold for upsampling. To start, we recall that the input/output description of an upsampler is

$$y(n) = \begin{cases} x\left(\frac{n}{I}\right), & n = 0, \pm I, \pm 2I, \ldots \\ 0, & \text{otherwise} \end{cases} \overset{z}{\longleftrightarrow} Y(z) = X(z^I) \tag{11.5.9}$$

The output of the system in Figure 11.5.4(a) can be written as

$$Y(z) = H(z^I)V_1(z) = H(z^I)X(z^I) \tag{11.5.10}$$

because $V_1(z) = X(z^I)$. The output of the system in Figure 11.5.4(b) is given by

$$Y(z) = V_2(z^I) = H(z^I)X(z^I) \tag{11.5.11}$$

which is equal to (11.5.10). This shows that the two systems in Figure 11.5.4 are identical.

In conclusion, we have shown that it is possible to interchange the operation of linear filtering and downsampling or upsampling if we properly modify the system function of the filter.

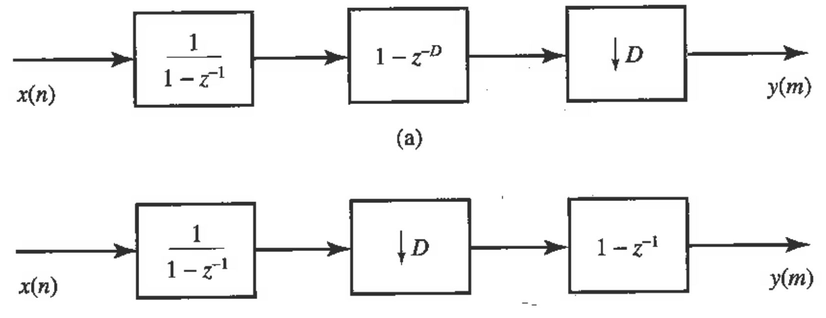
## 11.5.3 Sampling Rate Conversion with Cascaded Integrator Comb Filters

The hardware implementation of the lowpass filter required for sampling rate conversion can be significantly simplified if we choose a comb filter with system function (see Section 5.4.5)

$$H(z) = \sum_{k=0}^{M-1} z^{-k} = \frac{1 - z^{-M}}{1 - z^{-1}} \tag{11.5.12}$$

This system can be implemented by cascading either the "integrator" $1/(1-z^{-1})$ with the comb filter $(1 - z^{-M})$ or vice versa. This leads to the name *cascaded integrator comb* (CIC) filter structure. The CIC structure does not require any multiplications or storage for the filter coefficients.

To obtain an efficient decimation structure, we start with an integrator-comb CIC filter followed by the downsampler and then we apply the first noble identity, as shown in Figure 11.5.5. For the interpolation case, we use an upsampler followed



(b)

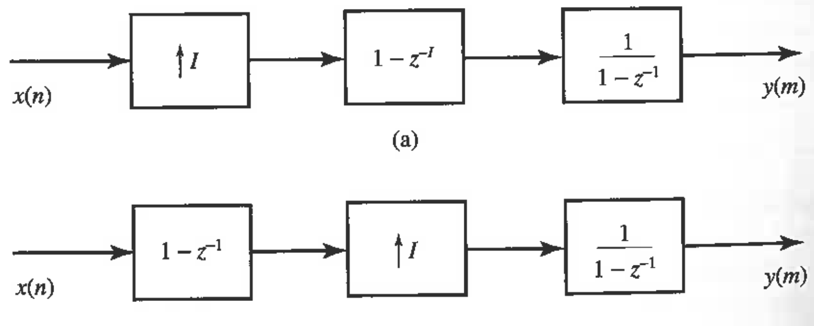
**Figure 11.5.5** Using the first noble identity to obtain an efficient CIC filter structure for decimation.

by a comb-integrator CIC filter and then we use the second noble identity, as shown in Figure 11.5.6. To improve the lowpass frequency response required for sampling rate conversion, we can cascade $K$ CIC filters. In this case, we order all integrators on one side of the filter and the comb filters on the other side, and then we apply the noble identities as in the single-stage case. The integrator $1/(1-z^{-1})$ is an unstable system. Therefore, its output may grow without limits, resulting in overflow when the integrator section comes first, as in the decimation structure shown in Figure 11.5.5(b). However, this overflow can be tolerated if the entire filter is implemented using two's-complement fixed-point arithmetic. If $D \neq M$ or $I \neq M$, the comb filter $1-z^{-1}$ in Figures 11.5.5(a) and 11.5.6(b) should be replaced by $1-z^{-M/D}$ or $1-z^{-M/I}$, respectively. A detailed treatment of CIC filters for decimation and interpolation can be found in Hogenauer (1981). Finally, we note that CIC filters are special cases of the frequency sampling structure discussed in Section 10.2.3.

If the CIC filter order is a power of 2, that is, $M = 2^K$, we can decompose the system function (11.5.12) as follows:

$$H(z) = (1+z^{-1})(1+z^{-2})(1+z^{-4})\ldots(1+z^{-2^{K-1}}) \qquad (11.5.13)$$

Using this decomposition we can develop decimator structures using nonrecursive



(b)

**Figure 11.5.6** Using the second noble identity to obtain an efficient CIC filter structure for interpolation.

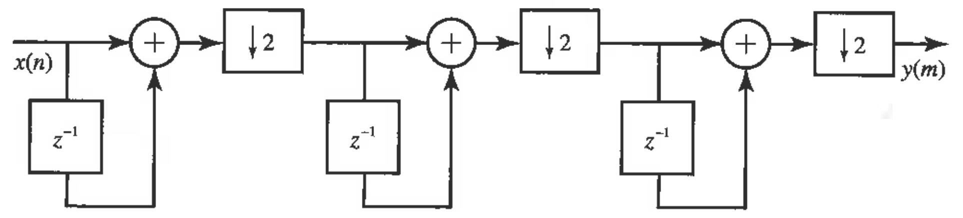


**Figure 11.5.7** Efficient filter structure for decimation by $D = 8$ using comb filters.

CIC filters. Figure 11.5.7 shows an example for a decimator with $D = M = 8$. Cascading of $N$ CIC filters can be obtained by providing $M$ first-order sections $(1-z^{-1})$ between each decimation stage. The constraint $M = 2^K$ can be relaxed by factoring $M$ into a product of prime numbers, as shown in Jang and Yang (2001).

### 11.5.4 Polyphase Structures for Decimation and Interpolation Filters

To develop a polyphase structure for decimation, we start with the straightforward implementation of the decimation process shown in Figure 11.5.8. The decimated sequence is obtained by passing the input sequence $x(n)$ through a linear filter and then downsampling the filter output by a factor $D$. In this configuration, the filter is operating at the high sampling rate $F_x$, while only one out of every $D$ output samples is actually needed. A logical solution would be to find a structure where only the needed samples are computed. We will develop such an efficient implementation by exploitating the polyphase structure in Figure 11.5.1. Since downsampling commutes with addition, combining the structures in Figures 11.5.8 and 11.5.1 yields the structure in Figure 11.5.9(a). If we next apply the identity in Figure 11.5.3, we obtain the desired implementation structure shown in Figure 11.5.9(b). In this filtering structure, only the needed samples are computed and all multiplications and additions are performed at the lower sampling rate $F_x/D$. Thus, we have achieved the desired efficiency. Additional reduction in computation can be achieved by using an FIR filter with linear phase and exploiting the symmetry of its impulse response.

In practice it is more convenient to implement the polyphase decimator using a *commutator model* as shown in Figure 11.5.10. The commutator rotates *counterclockwise* starting at time $n = 0$ and distributes a block of $D$ input samples to the polyphase filters starting at filter $i = D - 1$ and continuing in reverse order until $i = 0$. For every block of $D$ input samples, the polyphase filters receive a new input and their outputs are computed and summed to produce one sample of the output signal $y(m)$. The operation of this realization can be also understood by a careful inspection of Figure 11.1.3.

Next, let us consider the efficient implementation of an interpolator, which is realized by first inserting $I - 1$ zeros between successive samples of $x(n)$ and then

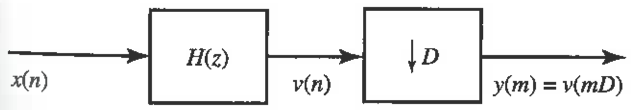


**Figure 11.5.8** Decimation system.