Chapter 11: Scaling and Round-off Noise

Keshab K. Parhi

Outline

- Introduction
- Scaling and Round-off Noise
- State Variable Description of Digital Filters
- Scaling and Round-off Noise Computation
- Round-off Noise Computation Using State Variable Description
- Slow-Down, Retiming, and Pipelining

Introduction

- **In a fixed-point digital filter implementation, the overall input-output behavior is non-ideal. The quantization of signals and coefficients using finite word-lengths and propagation of roundoff noises to the output are the sources of noise.**
- **Other undesirable behavior include limit-cycle oscillations where undesirable periodic components are present at filter output even in the absence of any input. These may be caused due to internal rounding or overflow.**
- **Scaling is often used to constrain the dynamic range of the variables to a certain word-length**
- **State variable description of a linear filter: provides a mathematical formulation for studying various structures. These are most useful to compute quantities that depend on the internal structure of the filter. Power at each internal node and the output round-off noise of a digital FIR/IIR filter can be easily computed once the digital filter is described in state variable form**

Scaling and Round-off Noise

Scaling Operation

- Scaling: A process of readjusting certain internal gain parameters in order to constrain internal signals to a range appropriate to the hardware with the constraint that the transfer function from input to output should not be changed
- Illustration:
  - The filter in Fig.11.1(a) with unscaled node x has the transfer function
$$H(z) = D(z) + F(z)G(z) \qquad (11.1)$$
  - To scale the node x, we divide F(z) by some number β and multiply G(z) by the same number as in Fig.11.1(b). Although the transfer function does not change by this operation, the signal level at node x has been changed

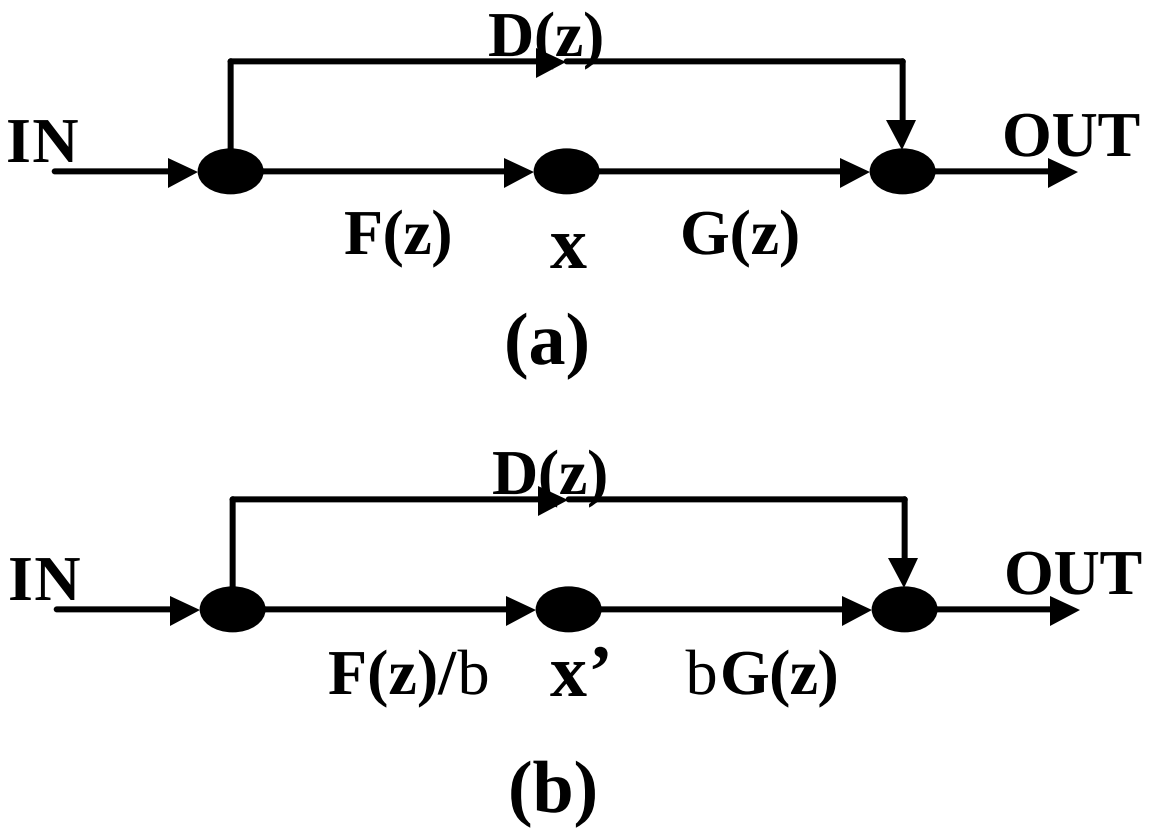

Fig.11.1 (a) A filter with unscaled node x, (b) A filter with scaled node x’

– The scaling parameter β can be chosen to meet any specific scaling rule such as

$$\begin{cases} l_1 - scaling: & b = \sum_{i=0}^{\infty} |f(i)|, \quad (11.2) \\ l_2 - scaling: & b = d\sqrt{\sum_{i=0}^{\infty} |f^2(i)|}, \quad (11.3) \end{cases}$$

  - where f(i) is the unit-sample response from input to the node x and the parameter δ can be interpreted to represent the number of standard deviations representable in the register at node x if input is unit-variance white noise

– If the input is bounded by $|u(n)| \le 1$, then

$$|x(n)| = \left|\sum_{i=0}^{\infty} f(i)u(n-i)\right| \le \sum_{i=0}^{\infty} |f(i)| \quad (11.4)$$

  - Equation (11.4) represents the true bound on the range of x and overflow is completely avoided by l_1 scaling in (11.2), which is the most stringent scaling policy

– Input can be generally assumed to be white noise.For unit-variance white noise input, variance at node ***x*** is given by:

$$E\left[x^2(n)\right] = \sum_{i=0}^{\infty} f^2(i) \qquad (11.5)$$

- l_2-scaling is commonly used because most input signals can be assumed to be white noise
- (11.5) is a variance (not a strict bound). So, we can increase δ in (11.3) to prevent possible overflow. But increasing δ will decrease SNR (signal-to-noise ratio). Thus, there is a trade-off between overflow and round-off noise

Scaling and Round-off Noise(cont'd)

Round-off Noise

- Round-off Noise: Product of two *W*-bit fixed-point fractions is a (2W-1) bit number. This product must eventually be quantized to W-bits by rounding or truncation, which results in round-off noise.
- Example:
  - Consider the 1st-order IIR filter shown in Fig. 11.2. Assume that the input wordlength W=8 bits, and the multiplier coefficient wordlength is also 8 bits. To maintain full precision in the output, we need to increase the output wordlength by 8 bits per iteration. This is clearly infeasible. Thus, the result needs to be rounded or truncated to its nearest 8-bit representation. This introduces a round-off noise e(n) (see Fig. 11.3).

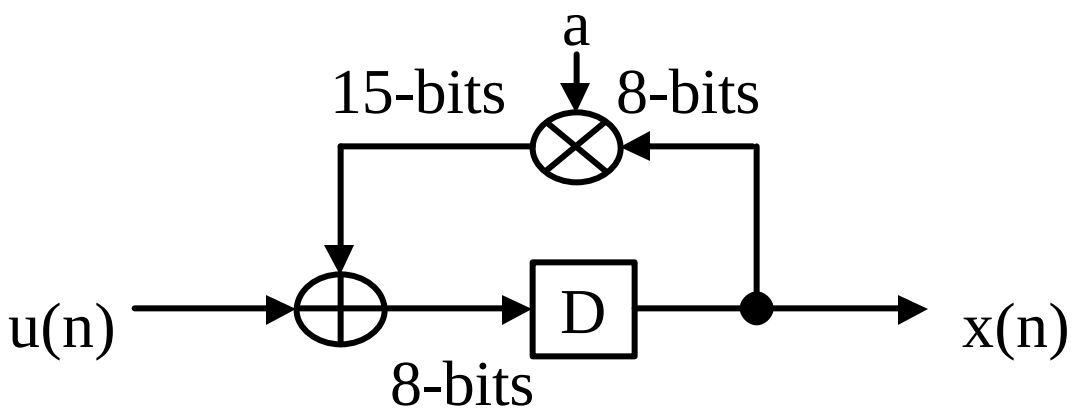

Fig.11.2 A 1ST-order IIR filter (W=8)

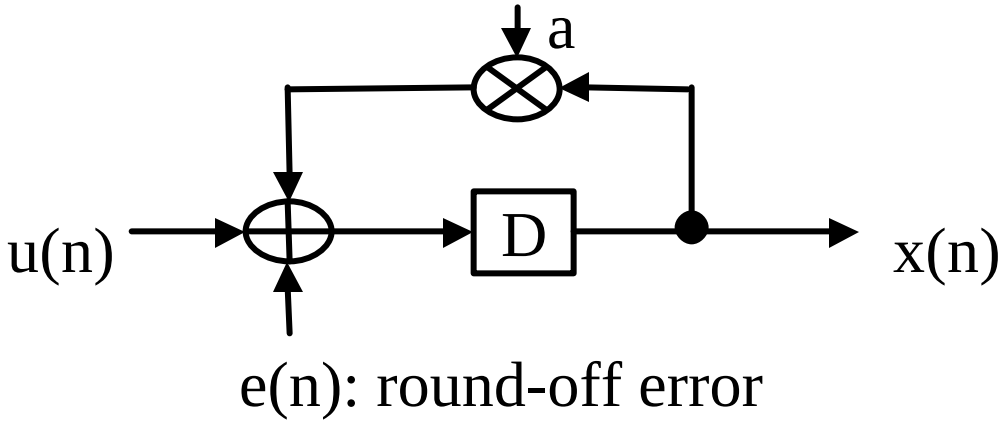

Fig.11.3 Model of Round-off Error

- Round-off Noise Mathematical Model: usually modeled as an infinite precision system with an external error input (see Fig.11.3)
- Rounding is a nonlinear operation, but its effect at the output can be analyzed using linear system theory with the following assumptions about e(n)
  - 1.e(n) is uniformly distributed white noise
  - 2. e(n) is a wide-sense stationary random process (mean & covariance of e(n) are independent of the time index n)
  - 3. e(n) is uncorrelated to all other signals such as input and other noise signals
- Let the wordlength of the output be W-bits, then the round-off error e(n) can be given by

$$-\frac{2^{-(W-1)}}{2} \le e(n) \le \frac{2^{-(W-1)}}{2} \qquad (11.6)$$

  - The error is assumed to be uniformly distributed over the interval in (11.6), the corresponding probability distribution is shown in Fig.11.4, where Δ is the length of the interval and $\Delta = 2^{-(W-1)}$

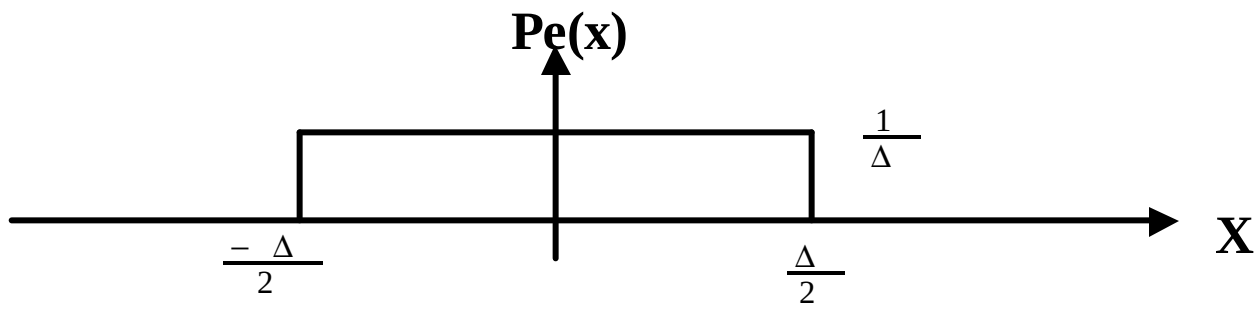

Fig.11.4 Error probability distribution

- The mean $E[e(n)]$ and variance $E[e^2(n)]$ of this error function:

$$\begin{cases} E[e(n)] = \int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} x P_e(x)dx = \frac{1}{\Delta}\frac{x^2}{2}\Big|_{\frac{-\Delta}{2}}^{\frac{\Delta}{2}} = 0 & (11.7) \\ E[e^2(n)] = \int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} x^2 P_e(x)dx = \frac{1}{\Delta}\frac{x^3}{3}\Big|_{\frac{-\Delta}{2}}^{\frac{\Delta}{2}} = \frac{\Delta^2}{12} = \frac{2^{-2W}}{3} & (11.8) \end{cases}$$

  - (11.8) can be rewritten as (11.9), where s_e^2 is the variance of the round-off error in a finite precision W-bit wordlength system

$$s_e^2 = 2^{-2W}/3 \qquad (11.9)$$

  - The variance is proportional to 2^{-2W}, so, increase in wordlength by 1 bit decreases the error by a factor of 4.
- Purpose of analyzing round-off noise: determine its effect at the output
  - If the noise variance at output is not negligible in comparison to the output signal level, the wordlength should be increased or some low-noise structure should be used.
  - We need to compute the SNR at the output, not just the noise gain to the output
  - In noise analysis, we use a double-length accumulator model: rounding is performed after two (2W-1)-bit products are added. Notice: multipliers are the sources for round-off noise

State Variable Description of Digital Filters

- Consider the signal flow graph (SFG) of an N-th order digital filter in Fig.11.5. We can represent it in the following recursive matrix form:

$$\begin{cases} \underline{x}(n+1) = \underline{\underline{A}} \cdot \underline{x}(n) + \underline{b} \cdot u(n), & (11.10) \\ y(n) = \underline{c}^{T} \cdot \underline{x}(n) + d \cdot u(n) & (11.11) \end{cases}$$

  - where $\underline{\boldsymbol{x}}$ is the state vector, $\boldsymbol{u}$ is the input, and $\boldsymbol{y}$ is the output of the filter; $\underline{\boldsymbol{x}}$, $\underline{\boldsymbol{b}}$ and $\underline{\boldsymbol{c}}$ are $\boldsymbol{N \times 1}$ column vectors; $\underline{\underline{A}}$ is $\boldsymbol{N \times N}$ matrix; $\boldsymbol{d}$, $\boldsymbol{u}$ and $\boldsymbol{y}$ are scalars.
- Let $\{f_i(n)\}$ be the unit-sample response from the input *u(n)* to the state $\underline{x}_i(n)$ and let $y_i(n)$ be the unit-sample response from the state $\underline{x}_i(n)$ to the output $\{g_i(n)\}$. It is necessary to scale the inputs to multipliers in order to avoid internal overflow

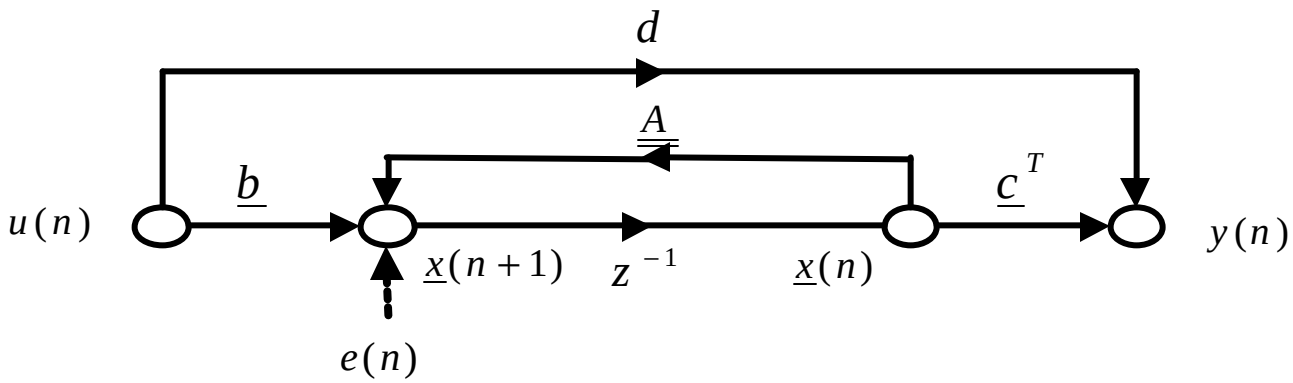

Fig.11.5 Signal flow graph of IIR filter

- Signals $\underline{x}(n)$ are input to the multipliers in Fig11.5. We need to compute $f(n)$ for scaling. Conversely, to find the noise variance at the output, it is necessary to find the unit-sample response from the location of the noise source $e(n)$ to $y(n)$. Thus g(n) represents the unit-sample response of the noise transfer function
- From the SFG of Fig.11.15, we can write:

$$\frac{\underline{X}(z)}{U(z)} = \frac{\underline{b} \cdot z^{-1}}{\underline{\underline{I}} - z^{-1} \cdot \underline{\underline{A}}} \qquad (11.12)$$

- Then, we can write the z-transform of ***f***(n), ***F***(z) as,

$$\underline{F}(z) = \underline{X}(z)/U(z) = (\underline{\underline{I}} + \underline{\underline{A}}z^{-1} + \underline{\underline{A}}^2 z^{-2} + \cdots)\underline{b}z^{-1}, \qquad (11.13)$$

$$\Rightarrow \underline{f}(n) = \underline{\underline{A}}^{n-1} \cdot \underline{b}, \quad n \geq 1. \qquad (11.14)$$

  - We can compute f(n) by substituting u(n) by δ(n) and using the recursion (11.15) and initial condition *f(0)=0:*

$$\underline{f}(n+1) = \underline{\underline{A}} \cdot \underline{f}(n) + \underline{b} \cdot d(n) \qquad (11.15)$$

  - The unit-sample response *g(n)* from the state ***x****(n)* to the output *y(n)* can be computed similarly with *u(n)=0.* The corresponding SFG is shown in Fig.11.6, which represents the following transfer function **G(z)**,

$$\underline{G}(z) = \frac{\underline{c}^T}{\underline{\underline{I}} - \underline{\underline{A}} \cdot z^{-1}}, \qquad (11.16)$$

$$\Rightarrow \underline{g}(n) = \underline{c}^T \cdot \underline{\underline{A}}^n, \quad n \geq 0 \qquad (11.17)$$

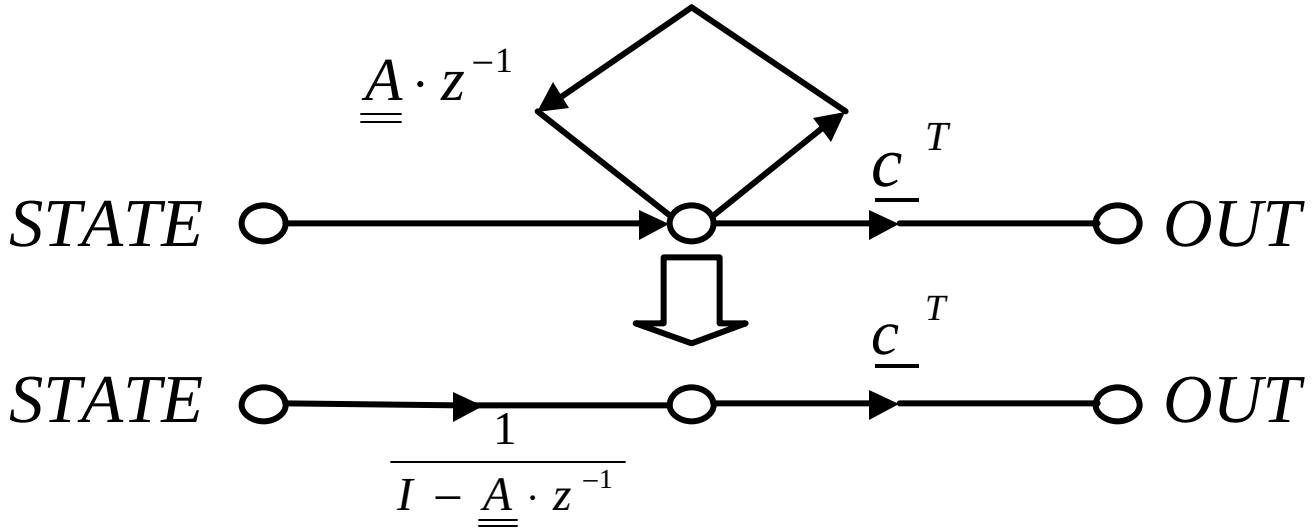

Fig.11.6 Signal flow graph of g(n)

- State covariance matrix **K**:

$$\underline{\underline{K}} \equiv E\left\{\underline{x}(n) \cdot \underline{x}^T(n)\right\} \qquad (11.18)$$

  - Because $\underline{X}$ is an N×1 vector, $\underline{K}$ is an N×N matrix
  - $\underline{K}$ is a measure of error power at various states (the diagonal element K_{ii} is the energy of the error signal at state x_i due to the input white noise)

- Express $\underline{K}$ in a form that reflects the error properties of the filter:
  - State vector $\underline{X}(n)$ can be obtained by the convolution of $u(n)$ and $f(n)$, by using (11.14) for $f(n)$, we get:

$$\underline{x}(n) = [x_1(n), x_2(n), \cdots, x_N(n)]^T \tag{11.19}$$

$$= \underline{f}(n) * u(n) = \sum_{l=0}^{\infty} \underline{\underline{A}}^l \underline{b} \cdot u(n-l-1) \tag{11.20}$$

  - Therefore

$$\underline{\underline{K}} = E\left\{ \sum_{l=0}^{\infty} (\underline{\underline{A}}^l \underline{b}) u(n-l-1) \sum_{m=0}^{\infty} u(n-m-1)(\underline{\underline{A}}^m \underline{b})^T \right\}$$

$$= E\left\{ \sum_{l=0}^{\infty} \sum_{m=0}^{\infty} \underline{\underline{A}}^l \underline{b} u(n-l-1) u(n-m-1)(\underline{\underline{A}}^m \underline{b})^T \right\}$$

$$= \sum_{l=0}^{\infty} \sum_{m=0}^{\infty} \underline{\underline{A}}^l \underline{b} E\left[u(n-l-1)u(n-m-1)\right](\underline{\underline{A}}^m \underline{b})^T \tag{11.21}$$

  - Assume u(n) is zero-mean unit-variance white noise, so we have:

$$\begin{cases} E[u^2(n)] = 1 & (11.22) \\ E[u(n)u(n-k)] = 0, \quad k \neq 0 & (11.23) \end{cases}$$

– Substituting (11.22) & (11.23) into (11.21), we obtain:

$$\underline{\underline{K}} = \sum_{l=0}^{\infty}\sum_{m=0}^{\infty} \underline{\underline{A}}^{l}\underline{b}\cdot d_{lm}\cdot(\underline{\underline{A}}^{m}\underline{b})^{T} = \sum_{l=0}^{\infty} f(l)f^{T}(l) = \sum_{l=0}^{\infty} \underline{\underline{A}}^{l}\underline{b}\cdot(\underline{\underline{A}}^{l}\underline{b})^{T}$$

$$= \underline{b}\underline{b}^{T} + \sum_{l=1}^{\infty} \underline{\underline{A}}^{l}\underline{b}\cdot(\underline{\underline{A}}^{l}\underline{b})^{T} = \underline{b}\underline{b}^{T} + \sum_{K=0}^{\infty} \underline{\underline{A}}^{K+1}\underline{b}\cdot(\underline{\underline{A}}^{K+1}\underline{b})^{T}$$

$$= \underline{b}\underline{b}^{T} + \sum_{K=0}^{\infty} \underline{\underline{A}}\left[\underline{\underline{A}}^{K}\underline{b}\cdot(\underline{\underline{A}}^{K}\underline{b})^{T}\right]\underline{\underline{A}}^{T} = \underline{b}\underline{b}^{T} + \underline{\underline{A}}\left[\sum_{K=0}^{\infty} \underline{\underline{A}}^{K}\underline{b}\cdot(\underline{\underline{A}}^{K}\underline{b})^{T}\right]\underline{\underline{A}}^{T} \quad (11.24)$$

– Finally, we get the Lyapunov equation:

$$\Rightarrow \underline{\underline{K}} = \underline{b}\cdot\underline{b}^{T} + \underline{\underline{A}}\cdot\underline{\underline{K}}\cdot\underline{\underline{A}}^{T} \quad (11.25)$$

- If for some state x_i, $E[x_i^2]$ has a higher value than other states, then x_i needs to be assigned more bits, which leads to extra hardware and irregular design.
  – By scaling, we can ensure that all nodes have equal power, and the same word-length can be assigned to all nodes.

- Orthogonal filter structure: All internal variables are uncorrelated and have unit variance assuming a white-noise input, it satisfies the following:

$$\underline{\underline{K}} = \underline{\underline{I}} = \underline{\underline{A}} \cdot \underline{\underline{A}}^T + \underline{b} \cdot \underline{b}^T \qquad (11.26)$$

- The advantages of orthogonal filter structure:
  - The scaling rule is automatically satisfied
  - The round-off noise gain is low and invariant under frequency transformations
  - Overflow oscillations are impossible
- Similarly, define the output covariance matrix $\underline{\underline{W}}$ as follows:

$$\underline{\underline{W}} = \sum_{n=0}^{\infty} \underline{g}^T(n)\underline{g}(n) = \sum_{n=0}^{\infty} (\underline{c}^T \underline{\underline{A}}^n)^T \underline{c}^T \underline{\underline{A}}^n \qquad (11.27)$$

- Proceeding in a similar manner as before, we can get

$$\underline{\underline{W}} = \underline{\underline{A}}^T \cdot \underline{\underline{W}} \cdot \underline{\underline{A}} + \underline{c} \cdot \underline{c}^T \qquad (11.28)$$

Scaling and Round-off Noise Computation

Scaling Operation

- The same word-length can be assigned to all the variables of the system only if all the states have equal power. This is achieved by scaling
- The state vector is pre-multiplied by inverse of the scaling matrix $\underline{\underline{T}}$.
  - If we denote the scaled states by $\underline{x}_S$, we can write,

$$\underline{x}_S(n) = \underline{\underline{T}}^{-1} \cdot \underline{x}(n) \quad \Rightarrow \quad \underline{x}(n) = \underline{\underline{T}} \cdot \underline{x}_S(n) \tag{11.29}$$

  - Substituting for $\underline{x}$ from (11.29) into (11.10) and solving for $\underline{x}_S$, we get

$$\underline{\underline{T}} \cdot \underline{x}_S(n+1) = \underline{\underline{A}} \cdot \underline{\underline{T}} \cdot \underline{x}_S(n) + \underline{b} \cdot u(n) \tag{11.30}$$

$$\Rightarrow \quad \underline{x}_S(n+1) = \underline{\underline{T}}^{-1} \cdot \underline{\underline{A}} \cdot \underline{\underline{T}} \cdot \underline{x}_S(n) + \underline{\underline{T}}^{-1} \cdot \underline{b} \cdot u(n) \tag{11.31}$$

$$\Rightarrow \quad \underline{x}_S(n+1) = \underline{\underline{A}}_S \cdot \underline{x}_S(n) + \underline{b}_S \cdot u(n) \tag{11.32}$$

– where $\left(\underline{\underline{A}}_S = \underline{\underline{T}}^{-1} \cdot \underline{\underline{A}} \cdot \underline{\underline{T}}, \quad \underline{b}_S = \underline{\underline{T}}^{-1} \cdot \underline{b}\right)$

- Similarly, the output equation (11.11) can be derived as follows

$$y(n) = \underline{c}^T \cdot \underline{\underline{T}} \cdot \underline{x}_S(n) + d \cdot u(n)$$

$$= \underline{c}_S^T \cdot \underline{x}_S(n) + d_S \cdot u(n)$$

$$\Rightarrow \left\{\underline{c}_S^T = \underline{c}^T \cdot \underline{\underline{T}}, \quad d_S = d\right\} \qquad (11.33)$$

- The scaled ***K*** matrix is given by

$$\underline{\underline{K}}_S = E[\underline{x}_S \cdot \underline{x}_S^T] = E[\underline{\underline{T}}^{-1} \underline{x} \cdot \underline{x}^T (\underline{\underline{T}}^{-1})^T] = \underline{\underline{T}}^{-1} E\left(\underline{x} \cdot \underline{x}^T\right) (\underline{\underline{T}}^{-1})^T$$

$$\Rightarrow \quad \underline{\underline{K}}_S = \underline{\underline{T}}^{-1} \cdot \underline{\underline{K}} \cdot (\underline{\underline{T}}^{-1})^T \qquad (11.34)$$

- It is desirable to have equal power at all states, so the transformation matrix ***T*** is chosen such that the ***Ks*** matrix of the scaled system has all diagonal entries as 1.

- Further assume T to be diagonal, i.e.,

$$\underline{\underline{T}} = diag\,[t_{11}, t_{22}, \cdots\cdots t_{NN}], \tag{11.35}$$

$$\Rightarrow \underline{\underline{T}}^{-1} = diag\,[\frac{1}{t_{11}}, \frac{1}{t_{22}}, \cdots\cdots \frac{1}{t_{NN}}] = (\underline{\underline{T}}^{-1})^{T} \tag{11.36}$$

  - From (11.34) and (11.35) and let $(K_S)_{ii} = 1$, we can obtain:

$$(K_S)_{ii} = \frac{K_{ii}}{t_{ii}^2} = 1, \tag{11.37}$$

$$\Rightarrow t_{ii} = \sqrt{K_{ii}} \tag{11.38}$$

- Conclusion: By choosing i-th diagonal entry in ***T*** to be equal to the square root of the i-th diagonal element of ***K*** matrix, all the states can be guaranteed to have equal unity power
- Example (Example 11.4.1,pp.387) Consider the unscaled 2nd-order filter shown in Fig.11.7, its state variable matrices are (see the next page):

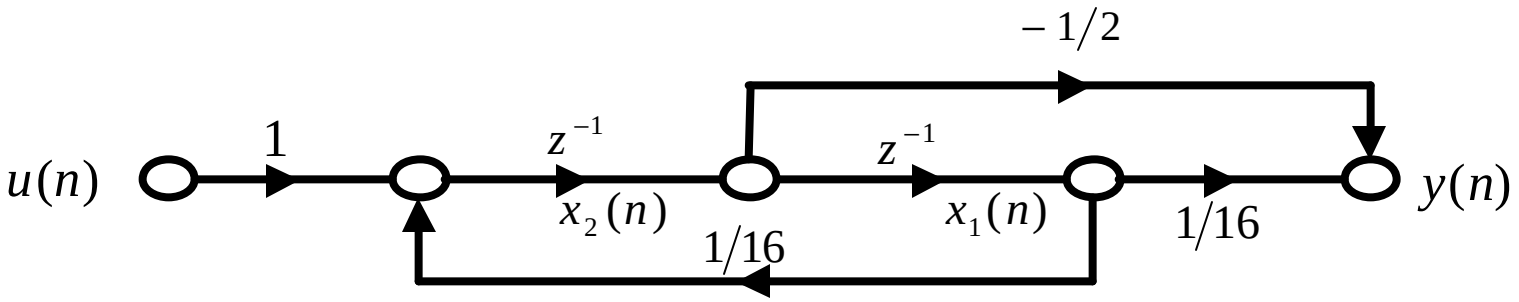

Fig.11.7 An SFG of an unscaled 2nd-order filter

– Example (cont'd)

$$\underline{\underline{A}} = \begin{bmatrix} 0 & 1 \\ \frac{1}{16} & 0 \end{bmatrix}, \quad \underline{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \underline{c} = \begin{bmatrix} \frac{1}{16} \\ -\frac{1}{2} \end{bmatrix}, \quad d = 0$$

– The state covariance matrix $\underline{\underline{\boldsymbol{K}}}$ can be computed using (11.25) as

$$\begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{1}{16} & 0 \end{bmatrix} \cdot \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \cdot \begin{bmatrix} 0 & \frac{1}{16} \\ 1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} K_{22} & \frac{1}{16} K_{21} \\ \frac{1}{16} K_{12} & \frac{1}{256} K_{11} + 1 \end{bmatrix}$$

– Thus, we get: $\left\{K_{11} = K_{22} = \frac{256}{255}, \quad K_{12} = K_{21} = 0\right\}$

– For l_2 scaling with δ=1, the transformation matrix is

$$\underline{\underline{T}} = \begin{bmatrix} \frac{16}{\sqrt{255}} & 0 \\ 0 & \frac{16}{\sqrt{255}} \end{bmatrix}$$

– Thus the scaled filter is described as below and is shown in Fig.11.8

$$\underline{\underline{A}}_S = \underline{\underline{T}}^{-1} \cdot \underline{\underline{A}} \cdot \underline{\underline{T}} = \begin{bmatrix} 0 & 1 \\ \frac{1}{16} & 0 \end{bmatrix}, \quad \underline{b}_S = \underline{\underline{T}}^{-1} \cdot \underline{b} = \begin{bmatrix} 0 \\ \frac{\sqrt{255}}{16} \end{bmatrix},$$

$$\underline{c}_S = \underline{\underline{T}}^{T} \cdot \underline{c} = \begin{bmatrix} \frac{1}{\sqrt{255}} \\ -\frac{8}{\sqrt{255}} \end{bmatrix}, \qquad d_S = 0$$

– Note: the state covariance matrix $\underline{\underline{\boldsymbol{K}}}s$ of the scaled filter is

$$\underline{\underline{K}}_S = \begin{bmatrix} \frac{\sqrt{255}}{16} & 0 \\ 0 & \frac{\sqrt{255}}{16} \end{bmatrix} \begin{bmatrix} \frac{256}{255} & 0 \\ 0 & \frac{256}{255} \end{bmatrix} \begin{bmatrix} \frac{\sqrt{255}}{16} & 0 \\ 0 & \frac{\sqrt{255}}{16} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

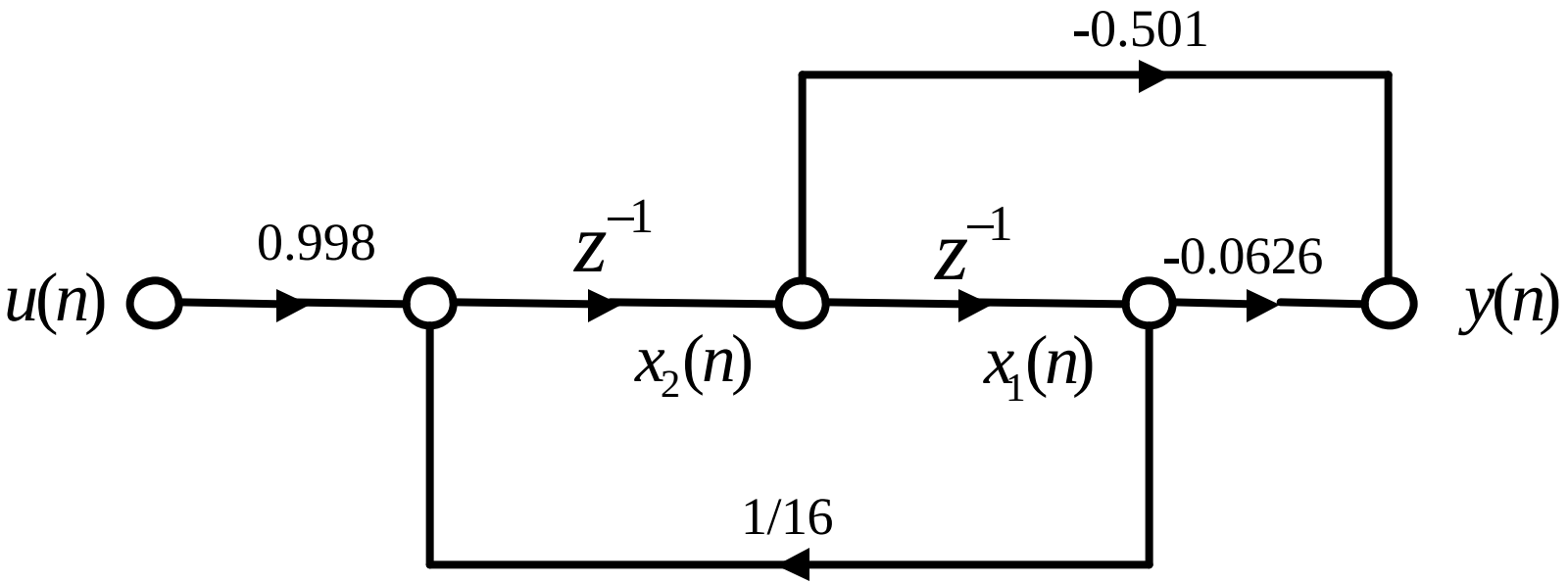

Fig.11.8 A SFG of a scaled 2nd-order filter

Scaling and Round-off Noise Computation (cont'd)

Round-off Noise Computation

- Computation: Let $e_i(n)$ be the error due to round-off at state x_i. Then the output round-off noise $y_i(n)$, due to this error, can be written as the convolution of the error input $e_i(n)$ with the state-to-output unit-sample response $g_i(n)$:

$$y_i(n) = e_i(n) * g_i(n) = \sum_{l=0}^{\infty} e_i(l) g_i(n-l) \qquad (11.39)$$

  - Consider the mean and the variance of $y_i(n)$. Since $e_i(n)$ is white noise with zero mean, so we have:

$$E[y_i(n)] = 0, \qquad (11.40)$$

$$E[y_i^2(n)] = E\left[\sum_l e_i(l) g_i(n-l) \sum_m e_i(m) g_i(n-m)\right]$$

$$= \sum_l \sum_m g_i(n-l) E[e_i(l) e_i(m)] g_i(n-m)$$

$$let \quad s_e^2 = E[e_i^2(n)] = \text{var iance}[e_i(n)]$$

– (cont'd) $$E\left[y_i^2(n)\right] = \sum_l \sum_m g_i(n-l)\, s_e^2 \cdot d_{lm}\, g_i(n-m)$$

$$= s_e^2 \sum_l g_i^2(n-l) = s_e^2 \sum_n g_i^2(n) \qquad (11.41)$$

– Expand $\underline{\underline{W}}$ in its explicit matrix form, we can observe that all its diagonal entries are of the form $\sum_n g_i^2(n)$:

$$\underline{\underline{W}} = \sum_n \underline{g}^T(n)\underline{g}(n) = \sum_n \begin{bmatrix} g_1(n), \\ \cdots \\ g_N(n) \end{bmatrix} \cdot \left[g_1(n), \quad \cdots, \quad g_N(n) \right] \qquad (11.42)$$

$$= \begin{bmatrix} \sum_n g_1^2(n) & \sum_n g_1(n)g_2(n) & \cdots & \sum_n g_1(n)g_N(n) \\ \sum_n g_2(n)g_1(n) & \sum_n g_2^2(n) & \cdots & \sum_n g_2(n)g_N(n) \\ \cdots & \cdots & \cdots & \cdots \\ \sum_n g_N(n)g_1(n) & \sum_n g_N(n)g_2(n) & \cdots & \sum_n g_N^2(n) \end{bmatrix} \qquad (11.43)$$

– Using (11.41), we can write the expression for the total output round-off noise in terms of trace of $\underline{\underline{W}}$:

$$total\_roundoff\_noise = s_e^2 \sum_{i=1}^{N} \sum_n g_i^2(n) = s_e^2 \sum_{i=1}^{N} W_{ii} = s_e^2 Trace(\underline{\underline{W}}) \quad (11.44)$$

– Note: (11.44) is valid for all cases. But when there is no round-off operation at any node, then the W_{ii} corresponding to that node should not be included while computing noise power

– (11.44) can be extended to compute the total round-off noise for the scaled system, which will simply be the trace of the scaled $\underline{\underline{W}}$ matrix:

$$\text{total round-off noise (scaled system)} = s_e^2 Trace(\underline{\underline{W}}_S) \quad (11.45)$$

– Replacing the filter parameters with the scaled parameters in (11.27), we can show:

$$\underline{\underline{W}}_S = \underline{\underline{T}}^T \cdot \underline{\underline{W}} \cdot \underline{\underline{T}} \quad (11.46)$$

– Also, for a diagonal $\underline{\underline{T}}$ we can write:

$$Trace\left(\underline{\underline{W}}_S\right) = \sum_{i=1}^{N} \left(W_S\right)_{ii} = \sum_{i=1}^{N} \left(t_{ii}^2 \cdot W_{ii}\right) \quad (11.47)$$

– (11.47) can be rewritten as follows because $t_{ii} \equiv \sqrt{K_{ii}}$

$$Trace\left(\underline{\underline{W}}_S\right) = \sum_{i=1}^{N}\left(K_{ii}\cdot W_{ii}\right) \quad \Rightarrow$$

total round-off noise (scaled system) $= s_e^2 \sum_{i=1}^{N}\left(K_{ii}\cdot W_{ii}\right)$ (11.48)

– **<u>Conclusion:</u>** The round-off noise of the scaled system can be computed using (11.48), i.e., using $\left\{K_{ii},\quad W_{ii}\right\}$

• Example (Example 11.4.2, p.390) To find the output round-off noise for the scaled filter in Fig.11.8, $\underline{\underline{\boldsymbol{W}}}$ can be calculated using (11.28) as

$$\begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{16} \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 \\ \frac{1}{16} & 0 \end{bmatrix} + \begin{bmatrix} \frac{1}{255} & \frac{-8}{255} \\ \frac{-8}{255} & \frac{64}{255} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{256}W_{22} + \frac{1}{255} & \frac{1}{16}W_{21} - \frac{8}{255} \\ \frac{1}{16}W_{12} - \frac{8}{255} & W_{11} + \frac{64}{255} \end{bmatrix}$$

– Thus $\begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix} = \begin{bmatrix} 0.0049 & -0.0332 \\ -0.0332 & 0.2559 \end{bmatrix}$

– The total output round-off noise for the scaled filter is

$$\left(W_{11}+W_{22}\right)\cdot s_e^2 = 0.2608 s_e^2$$

– For the unscaled filter in Fig.11.7:

$$\begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{16} \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 \\ \frac{1}{16} & 0 \end{bmatrix} + \begin{bmatrix} \frac{1}{256} & \frac{-1}{32} \\ \frac{-1}{32} & \frac{1}{4} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{256}W_{22} + \frac{1}{256} & \frac{1}{16}W_{21} - \frac{1}{32} \\ \frac{1}{16}W_{12} - \frac{1}{32} & W_{11} + \frac{1}{4} \end{bmatrix}$$

– Thus $\begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix} = \begin{bmatrix} 0.0049 & -0.0333 \\ -0.0333 & 0.2549 \end{bmatrix}$

– The total output round-off noise for the unscaled filter is

$$(W_{11}+W_{22})\cdot s_e^2 = 0.2598 s_e^2$$

– Notice: The scaled filter suffers from larger round-off noise, which can also be observed by comparing the unscaled and scaled filter structure:

$$\frac{roundoff\_noise(unscaled)}{roundoff\_noise(scaled)} = \frac{0.2598}{0.2608} < 1$$

- In the scaled filter, the input is scaled down by multiplying 0.998 to the input to avoid overflow (See Fig.11.8). Therefore, to keep the transfer functions the same in both filters, the output path of the scaled filter should have a gain which is 1/0.998 times the gain of the output path of the unscaled filter. Thus the round-off noise of the scaled filter is $1/0.998^2$ times that of the unscaled filter
- The above observation represents the tradeoff between overflow and round-off noise: More stringent scaling reduces the possibility of overflow but increases the effect of round-off noise

– Notice: (11.48) can be confirmed by:

$$(K_{11}W_{11}+K_{22}W_{22})_{unscaled} = \frac{256}{255}(0.0049+0.2549) = 0.2608 = (W_{11}+W_{22})_{scaled}$$

Round-off Noise Computation Using State Variable Description

Algorithms for Computing $\underline{K}$ and $\underline{W}$

- Parseval's relation and Cauchy's residue theorem are useful for finding signal power or round-off noise of digital filters. But, they are not useful for complex structures.
- The power at each internal node and the output round-off noise of a complex digital filter can be easily computed once the digital filter is described in state variable form
- Algorithm for computing $\underline{K}$
  - Using (11.24), K can be computed efficiently by the following algorithm: (see it on next page)

- Algorithm for computing $\underline{\underline{K}}$ (cont'd)
  - 1. Initialize: $\underline{\underline{F}} \leftarrow \underline{\underline{A}}, \quad \underline{\underline{K}} \leftarrow \underline{b} \cdot \underline{b}^T$
  - 2. Loop: $\underline{\underline{K}} \leftarrow \underline{\underline{F}} \cdot \underline{\underline{A}} \cdot \underline{\underline{F}}^T, \quad \underline{\underline{F}} \leftarrow \underline{\underline{F}}^2$
  - 3. Computation continues until $\underline{\underline{F}} = 0$
- Algorithm analysis:
  - After the 1st-loop iteration:

$$\begin{cases} \underline{\underline{K}} = \underline{\underline{A}} \cdot (\underline{b}\underline{b}^T) \cdot \underline{\underline{A}}^T + \underline{b}\underline{b}^T \\ \underline{\underline{F}} = \underline{\underline{A}}^2 \end{cases} \qquad (11.49)$$

  - After the 2nd-loop iteration:

$$\begin{cases} \underline{\underline{K}} = \underline{\underline{A}}^3 \underline{b}\underline{b}^T (\underline{\underline{A}}^3)^T + \underline{\underline{A}}^2 \underline{b}\underline{b}^T (\underline{\underline{A}}^2)^T + \underline{\underline{A}}\underline{b}\underline{b}^T \underline{\underline{A}}^T + \underline{b}\underline{b}^T, \\ \underline{\underline{F}} = \underline{\underline{A}}^4 \end{cases} \qquad (11.50)$$

- Thus, each iteration doubles the number of terms in the sum of (11.24). The above algorithm converges as long as the filter is stable (because the eigen-values of the matrix $\underline{\underline{\boldsymbol{A}}}$ are the poles of the transfer function)
- This algorithm can be used to compute $\underline{\underline{\boldsymbol{W}}}$ after some changes

- Algorithm for Computing $\underline{\underline{W}}$
  - 1. Initialize: $\underline{\underline{F}} \leftarrow \underline{\underline{A}}^T, \quad \underline{\underline{W}} \leftarrow \underline{c} \cdot \underline{c}^T$
  - 2. Loop: $\underline{\underline{W}} \leftarrow \underline{\underline{F}} \cdot \underline{\underline{W}} \cdot \underline{\underline{F}}^T, \quad \underline{\underline{F}} \leftarrow \underline{\underline{F}}^2$
  - 3. Computation continues until $\underline{\underline{F}} = 0$
- Example (Example 11.6.1, p.404) Consider the scaled-normalized lattice filter in Fig.11.9. We need to compute the signal powers at node 1, 2 and 3:
  - Because there are 3 states (1—3), the dimensions of the matrix $\underline{\underline{\boldsymbol{A}}}$, $\underline{\boldsymbol{b}}$, $\underline{\boldsymbol{c}}$ and $\boldsymbol{d}$ are 3×3, 3×1, 3×1, and 1×1, respectively. From Fig.11.9, the state equations can be written as (see next page)

$$\begin{cases} x_1(n+1) = 0.4944x_1(n) - 0.1915x_2(n) + 0.0443x_3(n) + 0.8467u(n), \\ x_2(n+1) = 0.3695x_1(n) + 0.9054x_2(n) - 0.2093x_3(n) \\ x_3(n+1) = 0.2252x_1(n) + 0.9743x_3(n) \\ y(n) = 0.0184x_1(n) + 0.1035x_2(n) + 0.3054x_3(n) + 0.0029u(n) \end{cases}$$

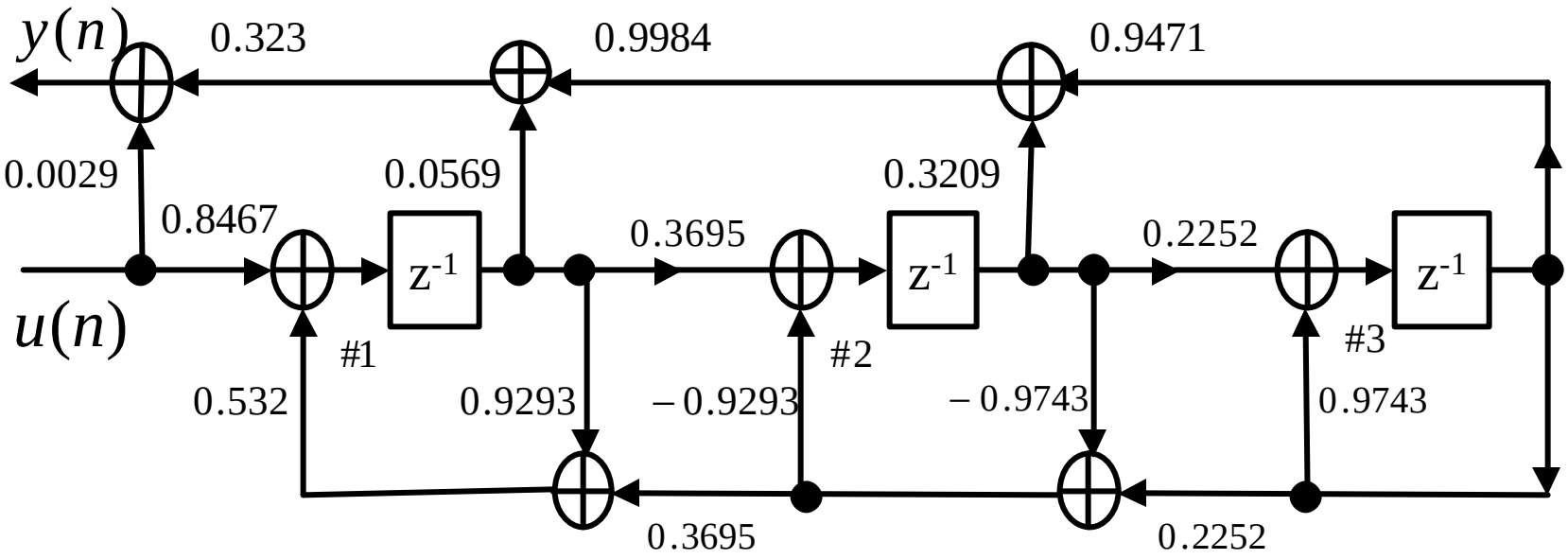

Fig.11.9 A 3rd-order scaled-normalized lattice filter
(also see Fig.11.18, p.403, Textbook)

- From these equations, matrices ***A***, ***b***, ***c*** and ***d*** can be obtained directly. By substituting them into the K-computing algorithm, we get

$$\underline{\underline{K}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Since $\{K_{11} = K_{22} = K_{33} = 1\}$, so no scaling is needed for nodes 1—3. In addition, the K matrix shows that the signals at nodes 1—3 are orthogonal to each other since all off-diagonal elements are zeros
- By the W-computing algorithm, we obtain:

$$\{W_{11} = 0.1455, W_{22} = 0.2952, W_{33} = 0.3096\}$$

- Conclusion:
  - Using state variable description method, we can compute signal power or round-off noise of a digital filter easily and directly. However, it can not be used on the nodes that are not connected to unit-delay branches because these nodes do not appear in the state variable description

Slow-Down, Retiming, and Pipelining

Introduction

- Many useful realizations contains roundoff nodes that are not connected to unit-delay branches. Thus these nodes (variables) do not appear in a state variable description and the scaling and roundoff noise computation methods can not be applied directly.
- The SRP (slow-down and retiming/pipelining) transformation technique can be used as a preprocessing step to overcome this difficulty
  - Slow-down: every delay element ($\mathbf{Z}$) in the original filter is changed into M delay element (Z^M)
  - Retiming and Pipelining (Please see Chapters 4 and 3 for details)

- Slow-down: Consider the filter in Fig.11.10(b) which is obtained by applying slow-down transformation (M=3) to the filter in Fig.11.10(a). By 3 slow down transformation, every *Z*-variable in Fig.11.10(a) is changed into Z^3. Thus the transfer function of the transformed filter H'(Z) is related to the original transfer function H(Z) as (11.51):

$$H'(z) = F'(z)G'(z) = F(z^3)G(z^3) = H(z^3) \qquad (11.51)$$

  - Thus, if the unit-sample response from the input to the internal node x in Fig.11.10(a) is defined by:

$$f(n) = \{f(0),\ f(1),\ f(2),\cdots\}, \qquad (11.52)$$

  - Then, the unit-sample response from the input to the internal node x' in Fig.11.10(b) is:

$$f'(n) = \{f(0),0,0,\ f(1),0,0,\ f(2),0,0,\cdots\}, \qquad (11.53)$$

  - We can get:

$$K'_{xx} = \sum_n [f'(n)]^2 = \sum_n f(n)^2 = K_{xx} \qquad (11.54)$$

  - Similarly it can be shown that:

$$W'_{xx} = W_{xx} \qquad (11.55)$$

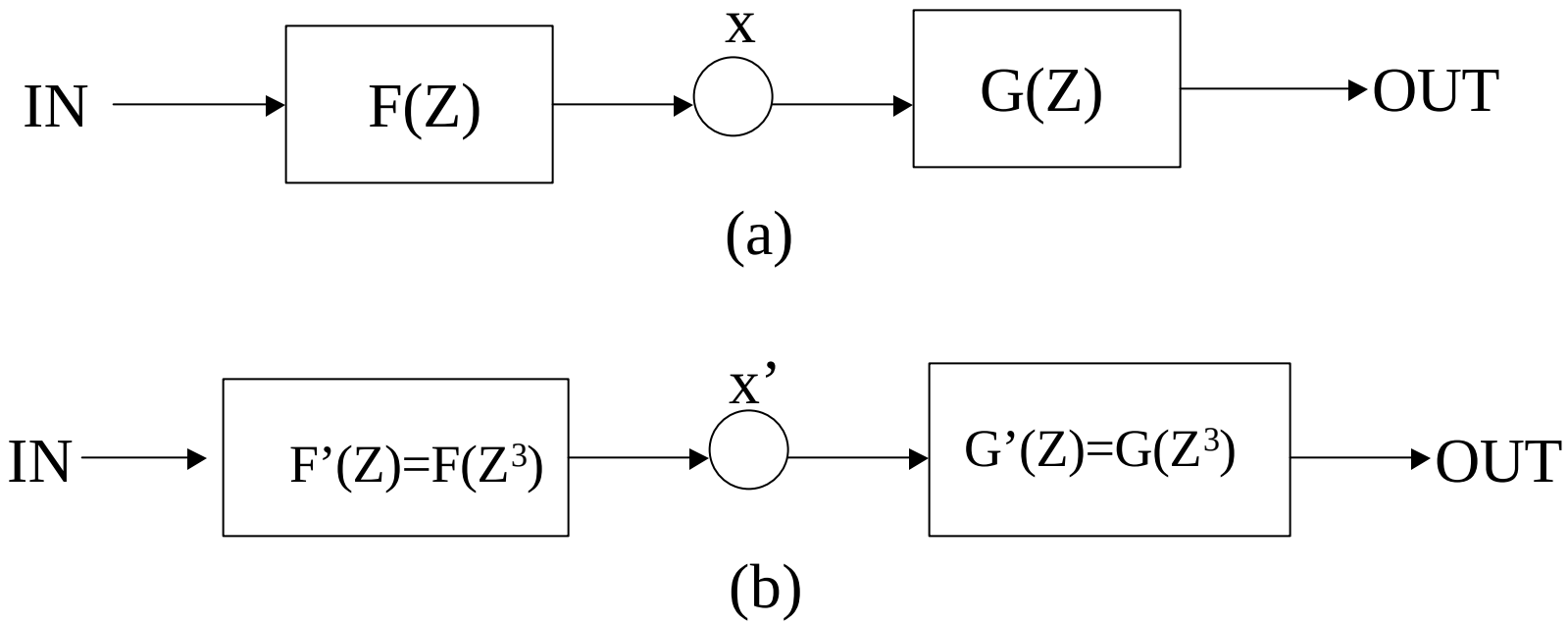

Figure 11.10 (a) A filter with transfer function H(z)=F(Z)G(Z). (b) Transformed filter obtained by 3 slow-down transformation H'(Z)=F(Z^3)G(Z^3).

- The foregoing analysis shows that slow-down transformation does not change the finite word-length behavior

- Pipelining:
  - Consider the filter in Fig.11.11(a), which has a non-state variable node x on the feed-forward path. It is obvious that the non-state variable node cannot be converted into the state variable node by slow-down transformation.
  - However, since x is on the feed-forward path, a delay can be placed on a proper cut-set location as shown in Fig.11.11(b). This pipelining operation converts the non-state variable node x into state variable node. The output sequence of the pipelined filter is equal to that of the original filter except one clock cycle delay.
  - So, the pipelined filter undergoes the same possibility of overflow and the same effect of round-off noise as in the original filter. Thus it is clear that pipelining does not change the filter finite word-length behavior

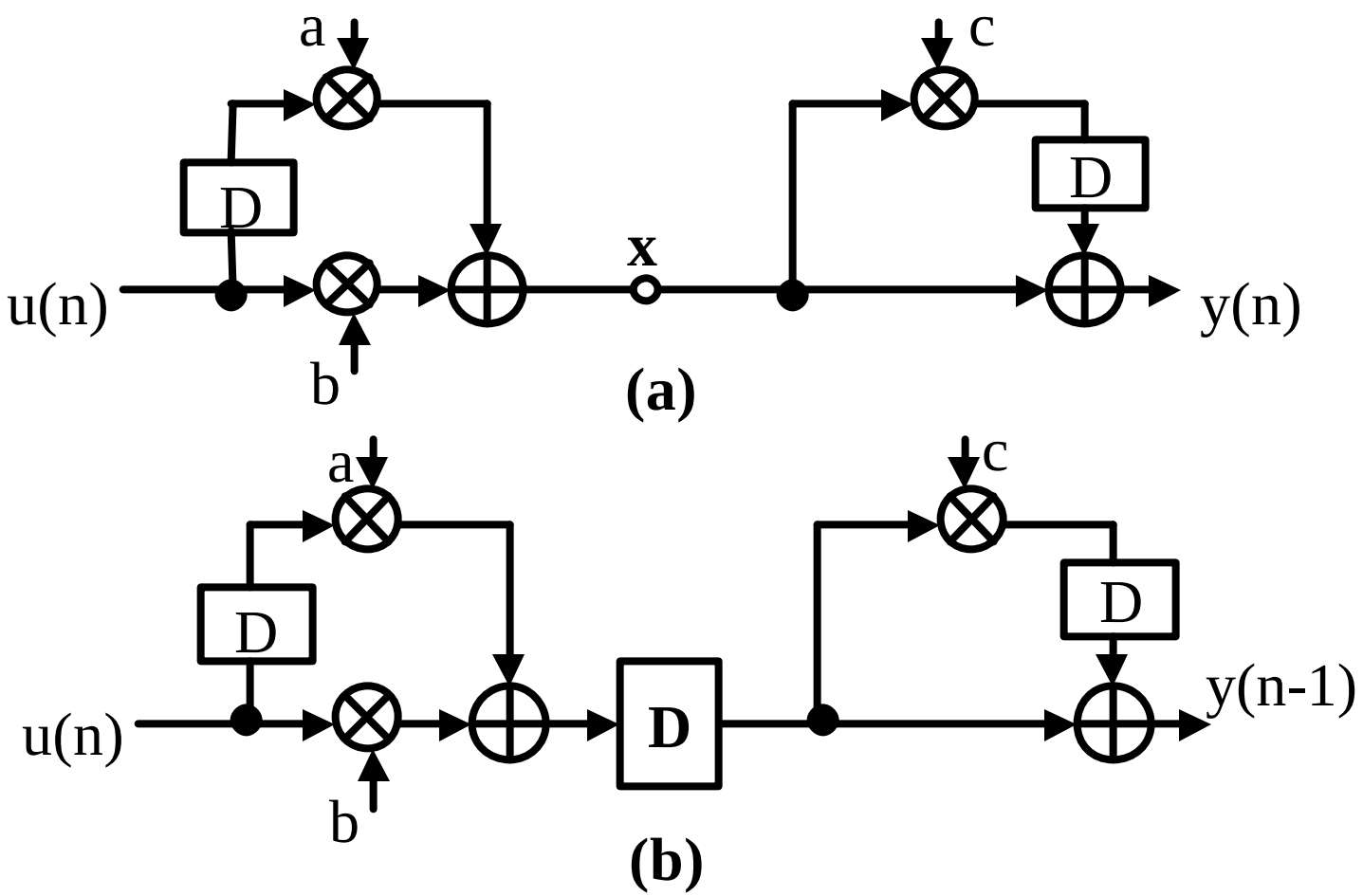

Fig.11.11 (a) A filter with a non-state variable node on a feed-forward path (b) Non-state variable node is converted into state variable node by pipelining

- Retiming:
  - In a linear array, if either all the left-directed or all the right-directed edges between modules carry at least 1 delay on each edge, the cut-set localization procedure can be applied to transfer some delays or a fraction of a delay to the opposite directed edges (see Chapter 4) — This is called **retiming**
- SRP transformation technique is summarized as follows:
  - 1. Apply slow-down transformation by a factor of M to a linear array, i.e., replace Z by Z^M. Also, apply pipelining technique to appropriate locations.
  - 2. Distribute the additional delays to proper locations such that non-state variable nodes are converted to state variable nodes
  - 3. Apply the scaling and noise computation method using state variable description
- Example (Example 11.7.1, p.407) Consider the filter shown in Fig.11.12, same as the 3rd-order scaled-normalized lattice filter in Fig.11.9 except that it has five more delays. The SFG in Fig.11.12 is obtained by using a 2-slow transformation and followed by retiming or cut-set transformation. (cont'd)

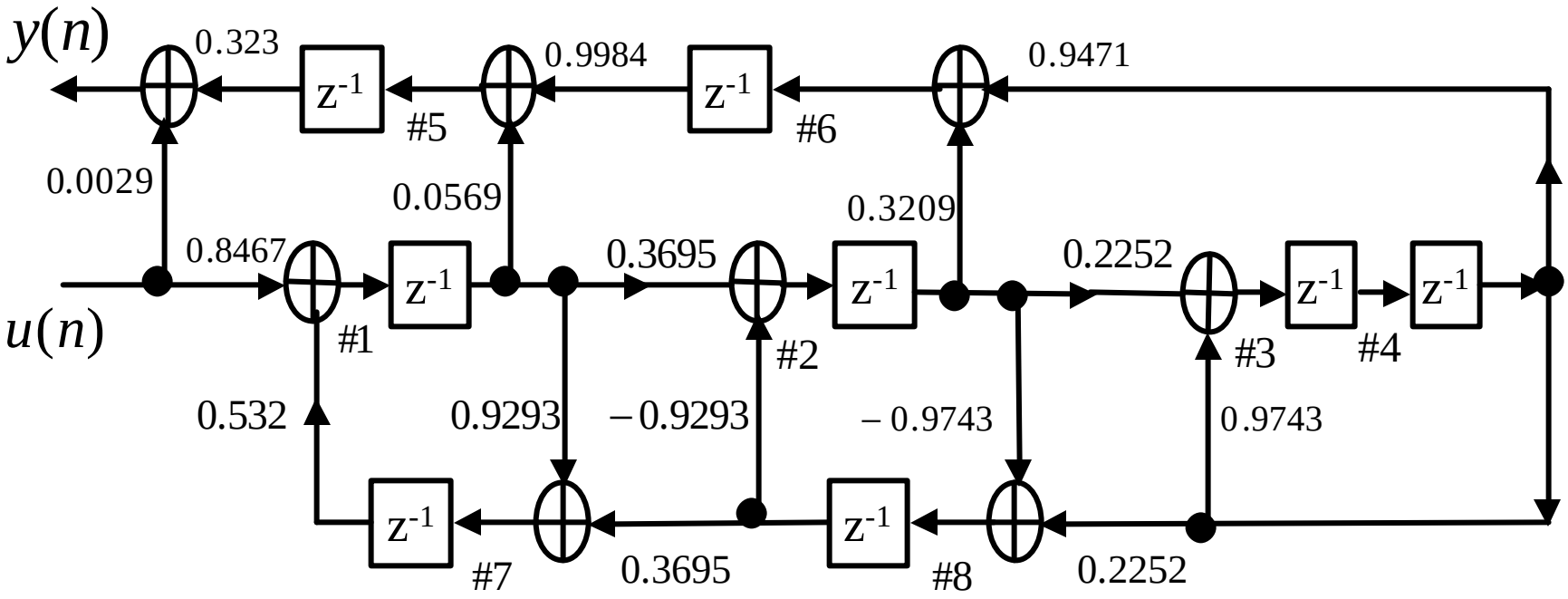

Fig.11.12 A transformed filter of the 3rd-order scaled-normalized lattice filter in Fig.11.9 (also see Fig.11.21,pp.407)

– (cont'd) Notice that signal power or round-off noise at every internal node in this filter can be computed using state variable description since each node is connected to a unit delay branch. Since there are 8 states, the dimensions of the matrices $\underline{\boldsymbol{A}}$, $\underline{\boldsymbol{b}}$, $\underline{\boldsymbol{c}}$, and d are 8×8, 8×1, 8×1, and 1×1, respectively. From Fig.11.12, state equations can be written as follows:

$$\begin{cases} x_1(n+1) = 0.532\,x_7(n) + 0.8467\,u(n), \\ x_2(n+1) = 0.3695\,x_1(n) - 0.9293\,x_8(n), \\ x_3(n+1) = 0.2252\,x_2(n) + 0.9743\,x_4(n), \\ x_4(n+1) = x_3(n), \\ x_5(n+1) = 0.0569\,x_1(n) + 0.9984\,x_6(n), \\ x_6(n+1) = 0.3209\,x_2(n) + 0.9471\,x_4(n), \\ x_7(n+1) = 0.9293\,x_1(n) + 0.3695\,x_8(n), \\ x_8(n+1) = -0.9743\,x_2(n) + 0.2252\,x_4(n), \\ y(n) = 0.323\,x_5(n) + 0.0029\,u(n) \end{cases}$$

- From the above equations, matrices ***A***, ***b***, ***c***, and d can be obtained directly. Using the K-computing algorithm, we obtain $\{K_{ii} = 1, \quad i = 1,2,\cdots,8\}$, which means that every internal node is perfectly scaled. Similarly, we get $W_{11},\cdots,W_{88} = \{0.1455, 0.2952, 0.3096, 0.3096, 0.1043, 0.104, 0.0412, 0.1912\}$
- Thus, the total output round-off noise is: $= s_e^2 \sum_i K_{ii} W_{ii} = 1.191\, s_e^2$
- Note: no round-off operation is associated with node 4 or state x_4. Therefore, W_{44} is not included in Trace(***W***) for round-off noise computation

- Example (omitted, study at home)
  - For details, please see Example 11.7.2, p.408 of textbook