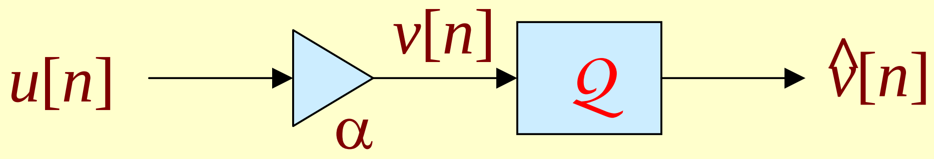
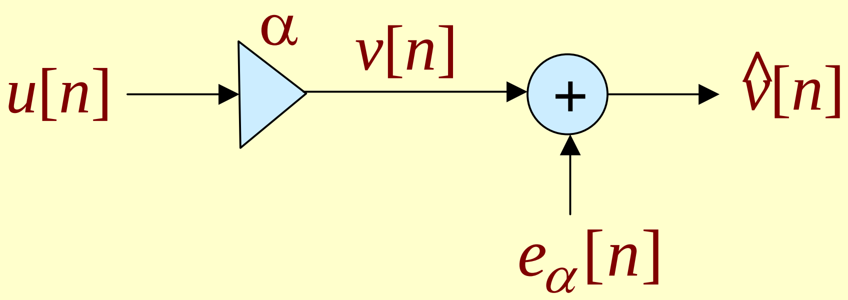
Analysis of Arithmetic Round-Off Errors

- In the fixed-point implementation of a digital filter only the result of the multiplication operation is quantized
- The representation of a practical multiplier with the quantizer at its output is shown below



Analysis of Arithmetic Round-Off Errors

- The statistical model of the multiplier with the quantizer at its output is as shown below



- The output $v[n]$ of the ideal multiplier is quantized to a value $\hat{v}[n]$, where

$$\hat{v}[n] = v[n] + e_\alpha[n]$$

Analysis of Arithmetic Round-Off Errors

- For analysis purposes, the following assumptions are made:

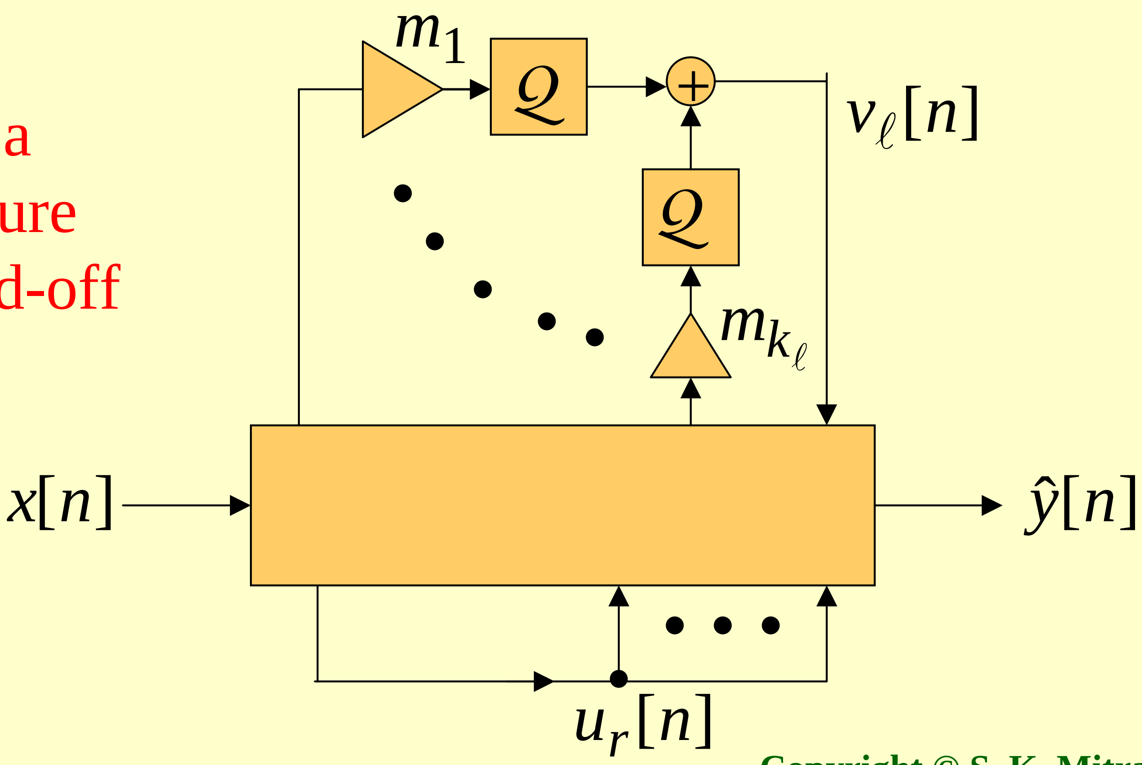
  (1) The error sequence $\{e_\alpha[n]\}$ is a sample sequence of a stationary white noise process, with each sample $e_\alpha[n]$ being uniformly distributed over the range of the quantization error

  (2) The error sequence $\{e_\alpha[n]\}$ is uncorrelated with the sequence $\{v[n]\}$, the input sequence $\{x[n]\}$, and all other quantization noise sources

Analysis of Arithmetic Round-Off Errors

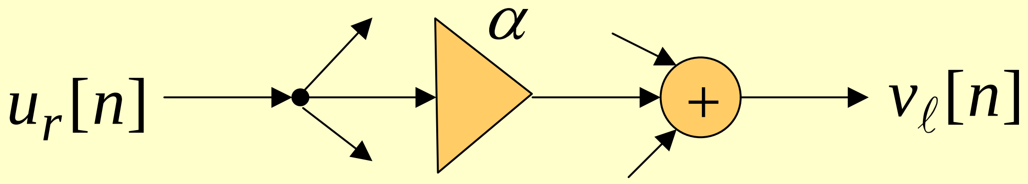
- Recall that the assumption of $\{e_\alpha[n]\}$ being uncorrelated with $\{v[n]\}$ holds only for rounding and two's-complement truncation

Representation of a digital filter structure with product round-off before summation



Analysis of Arithmetic Round-Off Errors

- The noise analysis model also shows the internal r-th branch node associated with the signal variable $u_r[n]$ that needs to be scaled to prevent overflow at this node
- These nodes are typically the inputs to the multipliers as indicated below

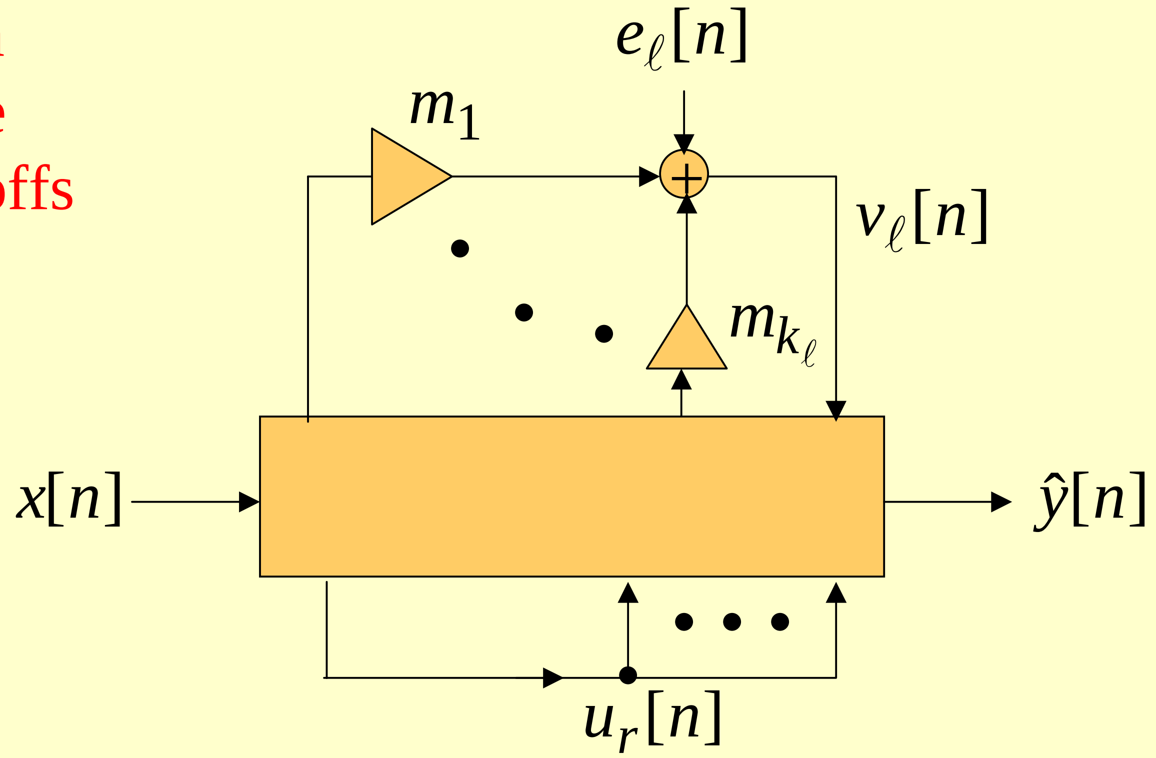


Analysis of Arithmetic Round-Off Errors

- In digital filters employing two's-complement arithmetic, these nodes are outputs of adders forming sums of products, as here the sums will still have the correct values even though some of the products and/or partial sums overflow
- It is assumed the error sources are statistically independent of each other and thus, each error source develops a round-off noise at the output of the digital filter

Analysis of Arithmetic Round-Off Errors

Statistical model of a digital filter structure with product round-offs before summation



Analysis of Arithmetic Round-Off Errors

- Notations:
- $f_r[n]$ - impulse response from the digital filter input to the *r*-th branch node
- $g_\ell[n]$ - impulse response from the input of the ℓ-th adder to the digital filter output
- $F_r(z) = \mathcal{Z}\{f_r[n]\}$ - *z*-transform of $f_r[n]$, called the **scaling transfer function**
- $G_\ell(z) = \mathcal{Z}\{g_\ell[n]\}$ - *z*-transform of $g_\ell[n]$, called the **noise transfer function**

Analysis of Arithmetic Round-Off Errors

- If σ_o^2 denotes the variance of each individual noise source at the output of each multiplier, the variance of $e_\ell[n]$ is simply $k_\ell \sigma_o^2$
- Variance of the output noise caused by $e_\ell[n]$ is then given by

$$\sigma_o^2 \cdot \left[k_\ell \left(\frac{1}{2\pi j} \oint_C G_\ell(z) G_\ell(z^{-1}) z^{-1} dz \right) \right]$$

$$= \sigma_o^2 \cdot \left[k_\ell \left(\frac{1}{2\pi} \int_{-\pi}^{\pi} \left| G_\ell(e^{j\omega}) \right|^2 d\omega \right) \right]$$

Analysis of Arithmetic Round-Off Errors

- If there are L such adders in the digital filter structure, the total output noise power due to all product round-offs is given by

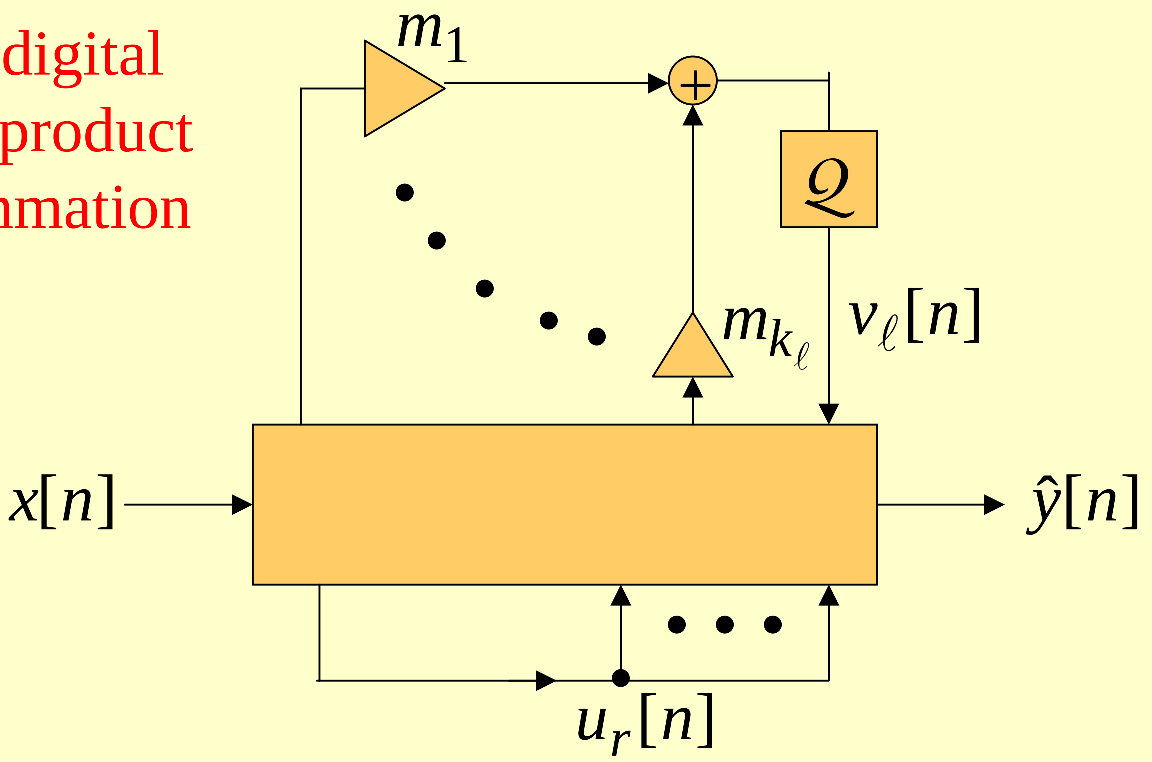
$$\sigma_{\gamma}^{2} = \sigma_{o}^{2} \sum_{\ell=1}^{L} k_{\ell} \left(\frac{1}{2\pi j} \oint_{C} G_{\ell}(z) G_{\ell}(z^{-1}) z^{-1} dz \right)$$

- If product round-off is carried out after the summation of products, then

$$\sigma_{\gamma}^{2} = \sigma_{o}^{2} \sum_{\ell=1}^{L} \left(\frac{1}{2\pi j} \oint_{C} G_{\ell}(z) G_{\ell}(z^{-1}) z^{-1} dz \right)$$

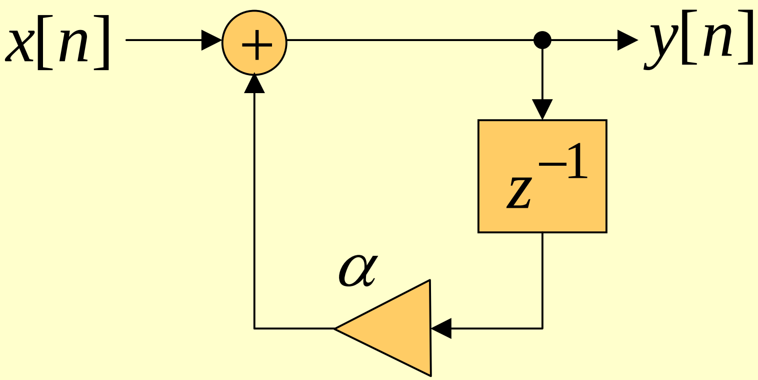
Analysis of Arithmetic Round-Off Errors

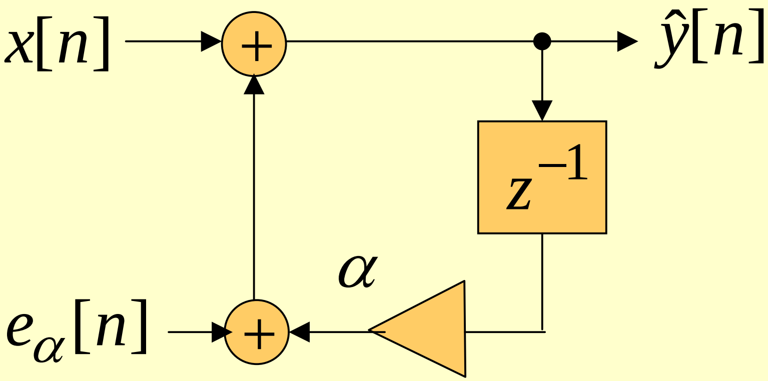
Representation of a digital filter structure with product round-offs after summation



Analysis of Arithmetic Round-Off Errors

- Example - For the first-order digital filter structure shown below on the left, the model for the product round-off error analysis is shown on the right





Analysis of Arithmetic Round-Off Errors

- From the noise analysis model it can be seen that the noise transfer function $G_\alpha(z)$ is the same as the filter transfer function $H(z)$, i.e.,

$$G_\alpha(z) = H(z) = \frac{z}{z-\alpha}$$

- Thus, the output noise variance due to the product round-off is same as that due to input quantization computed earlier:

$$\sigma_\gamma^2 = \frac{\sigma_o^2}{1-\alpha^2}$$

Analysis of Arithmetic Round-Off Errors

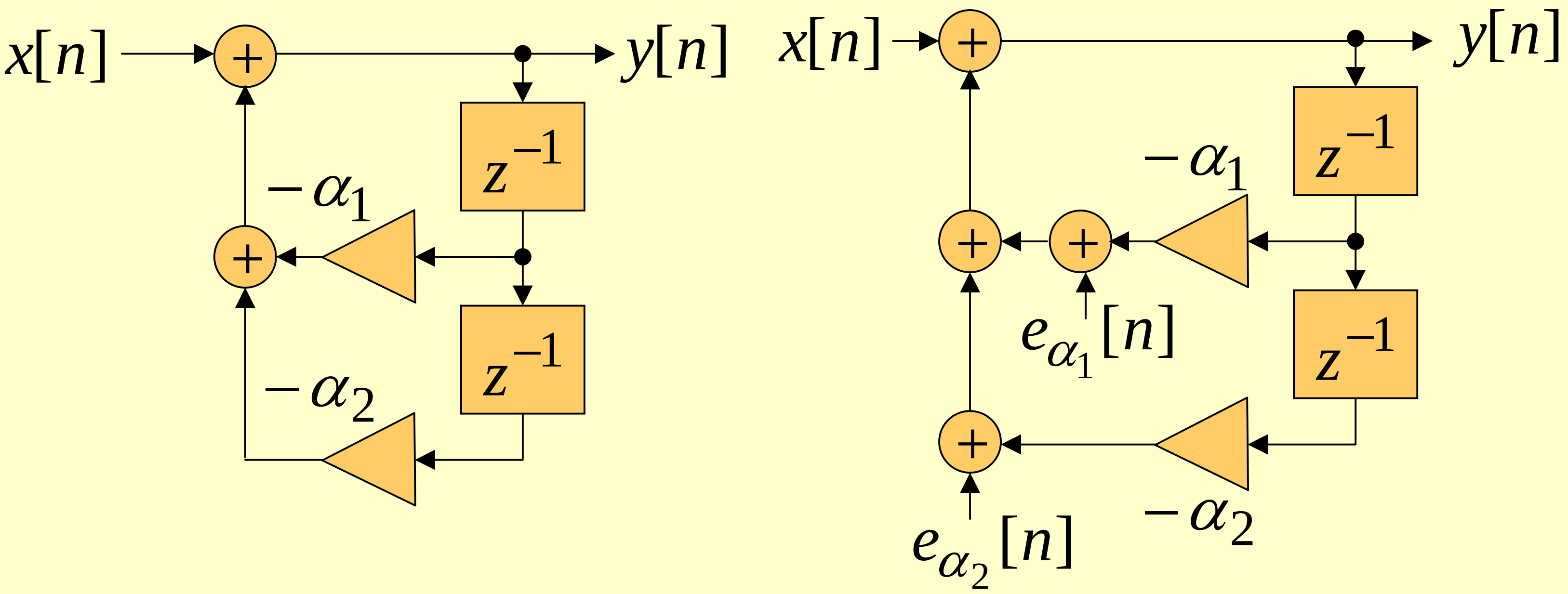
- The quantity $\sigma_{\gamma,n}^2 = \sigma_\gamma^2 / \sigma_o^2$ is called the **noise gain** or the **normalized round-off noise variance**

- Example - We now evaluate the output noise power of the direct form II realization of a causal second-order transfer function:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{z^2}{z^2 + \alpha_1 z + \alpha_2}$$

Analysis of Arithmetic Round-Off Errors

- The direct form II realization is shown below on the left and the model for error analysis is shown on the right



Analysis of Arithmetic Round-Off Errors

- The noise transfer functions $G_{\alpha_1}(z)$ and $G_{\alpha_2}(z)$ are same as the transfer function $H(z)$ of the digital filter
- A direct partial-fraction expansion of $H(z)$ is

$$H(z) = 1 + \frac{-\alpha_1 z - \alpha_2}{z^2 + \alpha_1 z + \alpha_2}$$

Analysis of Arithmetic Round-Off Errors

- Using the algebraic computation outlined earlier we get

$$\sigma_{\gamma,n}^2 = 2\left[1+\frac{(\alpha_1^2+\alpha_2^2)(1-\alpha_2^2)-2(\alpha_1\alpha_2-\alpha_1\alpha_2^2)\alpha_1}{(1-\alpha_2^2)^2+2\alpha_2\alpha_1^2-(1+\alpha_2^2)\alpha_1^2}\right]$$

$$= 2\left(\frac{1+\alpha_2}{1-\alpha_2}\right)\left(\frac{1}{1+\alpha_2^2+2\alpha_2-\alpha_1^2}\right)$$

Analysis of Arithmetic Round-Off Errors

- In terms of the pole locations $re^{\pm j\theta}$, we have $\alpha_1 = -2r\cos\theta$ and $\alpha_2 = r^2$
- Substituting these values in the expression for $\sigma_{\gamma,n}^2$ we get

$$\sigma_{\gamma,n}^2 = 2\left(\frac{1+r^2}{1-r^2}\right)\left(\frac{1}{1+r^4-2r^2\cos 2\theta}\right)$$

Analysis of Arithmetic Round-Off Errors

- If the poles are close to the unit circle, i.e., $r = 1 - \varepsilon$ where ε is a very small positive number, we can express $\sigma^2_{\gamma,n}$ as

$$\sigma^2_{\gamma,n} \cong \frac{1}{2\sin^2\theta}\left(\frac{1-\varepsilon}{\varepsilon}\right)\left(\frac{1}{1-2\varepsilon}\right)$$

- Thus, as the poles get closer to the unit circle, $\varepsilon \to 0$, the total output noise power increases rapidly

Dynamic Range Scaling

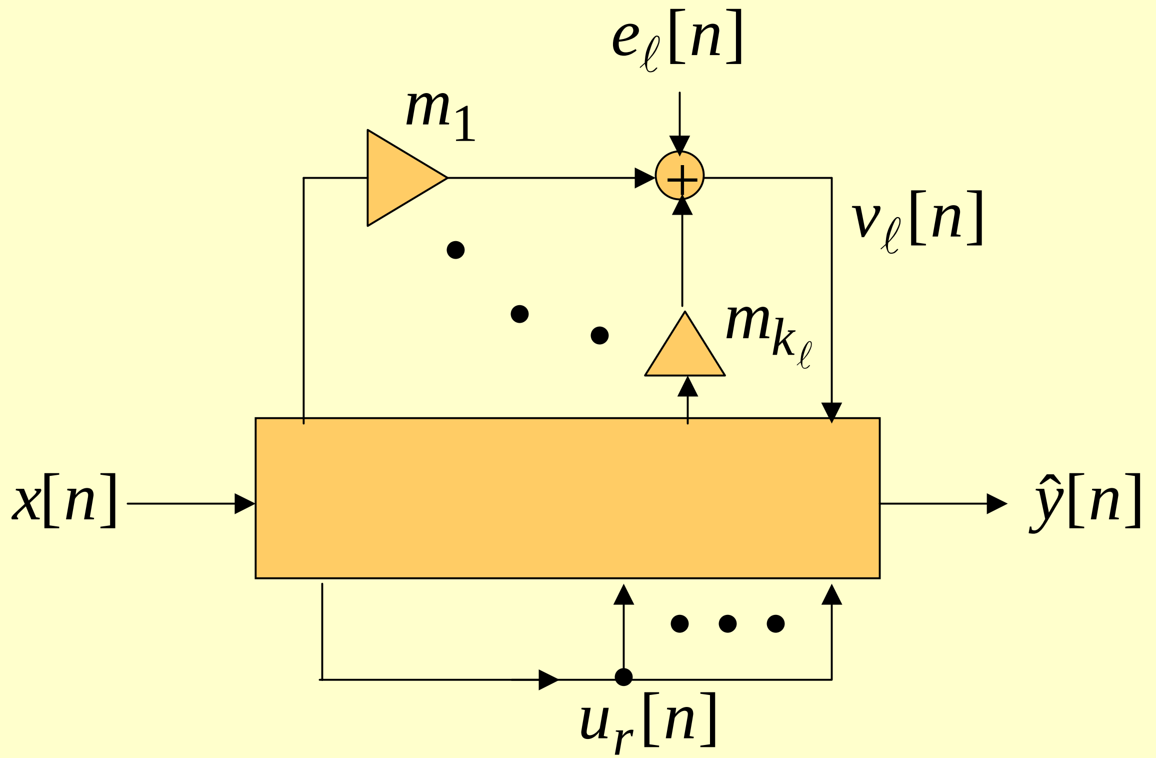
- In a digital filter implemented using fixed-point arithmetic, overflow may occur at certain internal nodes such as inputs to multipliers and/or the adder outputs
- Occurrence of overflows may lead to large amplitude oscillations at the filter output causing unsatisfactory operations

Dynamic Range Scaling

- Probability of overflow can be minimized significantly by properly scaling the internal signal levels with the aid of scaling multipliers
- In many cases, most of these multipliers can be absorbed with existing multipliers in the structure, thus reducing the total number of multipliers needed to implement the scaled filter

Dynamic Range Scaling

- To understand the basic concepts involved in scaling, consider the structure given below showing explicitly the *r*-th node variable $u_r[n]$ that needs to be scaled



Dynamic Range Scaling

- All fixed-point numbers are assumed to be represented as binary fractions
- Input sequence is assumed to be bounded by unity, i.e.,

$$|x[n]| \leq 1, \quad \text{for all values of } n$$

- Objective of scaling is to ensure that

$$|u_r[n]| \leq 1, \quad \text{for all } r \text{ and for all values of } n$$

Dynamic Range Scaling

- Three different conditions can be derived to ensure that $u_r[n]$ satisfies the above bound
- An Absolute Bound -
- Now

$$u_r[n] = \sum_{r=-\infty}^{\infty} f_r[k]x[n-k]$$

- From the above we get

$$|u_r[n]| = \left| \sum_{r=-\infty}^{\infty} f_r[k]x[n-k] \right| \leq \sum_{k=-\infty}^{\infty} |f_r[k]|$$

Dynamic Range Scaling

- Thus the condition $|u_r[n]| \leq 1$ is satisfied if

$$\sum_{k=-\infty}^{\infty} |f_r[k]| \leq 1, \;\; \text{for all } r$$

- The above condition is both necessary and sufficient to guarantee no overflow
- If this condition is not satisfied in the unscaled realization, the input signal can be scaled with a multiplier K of value

$$K = \frac{1}{\max_r \sum_{k=-\infty}^{\infty} |f_r[k]|}$$

Dynamic Range Scaling

- The scaling rule developed is based on a worst case bound and does not fully utilize the dynamic range of all adder output registers → significant reduction in SNR
- It is difficult to compute the value of K analytically
- Approximate value can be computed by replacing the infinite sum with a finite sum for a stable filter

Dynamic Range Scaling

- More practical and easy to use scaling rules can be derived in the frequency domain if some information about the input signals is known a priori
- Define the $\mathcal{L}_p$ -norm ($p \geq 1$) of a Fourier transform $F(e^{j\omega})$ as

$$\|F\|_p \triangleq \left(\frac{1}{2\pi} \int_{-\pi}^{\pi} \left| F(e^{j\omega}) \right|^p d\omega \right)^{1/p}$$

Dynamic Range Scaling

- $\|F\|_2$, the $\mathcal{L}_2$-norm, is the root-mean-squared (rms) value of $F(e^{j\omega})$ over $[-\pi, \pi]$
- $\|F\|_1$, the $\mathcal{L}_1$-norm, is the mean absolute value of $F(e^{j\omega})$ over $[-\pi, \pi]$
- $\lim_{p\to\infty}\|F\|_p$ exists for a continuous $F(e^{j\omega})$ and is given by

$$\|F\|_\infty = \max_{-\pi\le\omega\le\pi}\left|F(e^{j\omega})\right|$$

Dynamic Range Scaling

- A more realistic bound is derived next assuming that the input $x[n]$ is a deterministic signal with a DTFT $X(e^{j\omega})$
- $\mathcal{L}_\infty$-Bound
- Now from $u_r[n] = \sum_{k=-\infty}^{\infty} f_r[k]x[n-k]$ we get

$$U_r(e^{j\omega}) = F_r(e^{j\omega})X(e^{j\omega})$$

where $U_r(e^{j\omega})$ and $F_r(e^{j\omega})$ are the DTFTs of $u_r[n]$ and $f_r[n]$, respectively

Dynamic Range Scaling

- The inverse Fourier transform of

$$U_r(e^{j\omega}) = F_r(e^{j\omega})X(e^{j\omega})$$

yields

$$u_r[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} F_r(e^{j\omega})X(e^{j\omega})e^{j\omega n}d\omega$$

- Thus,

$$|u_r[n]| \le \frac{1}{2\pi}\int_{-\pi}^{\pi} |F_r(e^{j\omega})||X(e^{j\omega})|d\omega$$

$$\le \|F_r\|_\infty \left[\frac{1}{2\pi}\int_{-\pi}^{\pi} |X(e^{j\omega})|d\omega\right] \le \|F_r\|_\infty \|X\|_1$$

Dynamic Range Scaling

- Thus, if $\|X\|_1 \leq 1$, the dynamic range constraint $|u_r[n]| \leq 1$ is satisfied if

$$\|F_r\|_\infty \leq 1$$

- Hence, if the mean absolute value of the input spectrum is bounded by unity, then there will be no adder overflow if the peak gains from the filter input to all adder outputs are scaled to satisfy $\|F_r\|_\infty \leq 1$
- In general, this scaling rule is rarely used since in practice $\|X\|_1 \leq 1$ does not hold

Dynamic Range Scaling

- $\mathcal{L}_2$ - Bound
- Applying the Schwartz inequality to

$$u_r[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} F_r(e^{j\omega}) X(e^{j\omega}) e^{j\omega n} d\omega$$

we get

$$\left|u_r[n]\right|^2$$

$$\leq \left(\frac{1}{2\pi}\int_{-\pi}^{\pi} \left|F_r(e^{j\omega})\right|^2 d\omega \right)\left(\frac{1}{2\pi}\int_{-\pi}^{\pi} \left|X(e^{j\omega})\right|^2 d\omega \right)$$

Dynamic Range Scaling

- or equivalently,

$$|u_r[n]| \leq \|F_r(e^{j\omega})\|_2 \cdot \|X(e^{j\omega})\|_2$$

- Thus, if the input to the filter has finite energy bounded by unity, i.e., $\|X\|_2 \leq 1$, then the adder overflow can be prevented by scaling the filter such that the rms values of all scaling transfer functions from the input to all adder outputs are bounded by unity, i.e.,

$$\|F_r\|_2 \leq 1, \quad r = 1, 2, \ldots, R$$

Dynamic Range Scaling

- A General Scaling Rule -
- Obtained using Holder's inequality is given by

$$|u_r[n]| \le \|F_r\|_p \cdot \|X\|_q$$

where $p, q \ge 1$ satisfying $\frac{1}{p} + \frac{1}{q} = 1$

- Note: $\mathcal{L}_\infty$-bound is obtained when $p = \infty$ and $q = 1$

  and $\mathcal{L}_2$-bound is obtained when $p = 2$ and $q = 2$

- Another useful scaling rule, $\mathcal{L}_1$-bound is obtained when $p = 1$ and $q = \infty$

Dynamic Range Scaling

- After scaling, the scaling transfer functions become $\breve{F}_r(z)$ and the scaling constants should be chosen such that

$$\left\| \breve{F}_r \right\|_p \leq 1, \quad r = 1, 2, \ldots, R$$

- In many structures, all scaling multipliers can be absorbed into the existing feedforward multipliers without any increase in the total number of multipliers, and hence, noise sources

Dynamic Range Scaling

- In some cases, the scaling process may introduce additional multipliers in the system
- If all scaling multipliers are b-bit units, then

$$\left\| \breve{F}_r \right\|_p \leq 1, \quad r = 1, 2, \ldots, R$$

can be satisfied with an equality sign, providing a full utilization of the dynamic range of each adder output and thus yielding a maximum SNR

Dynamic Range Scaling

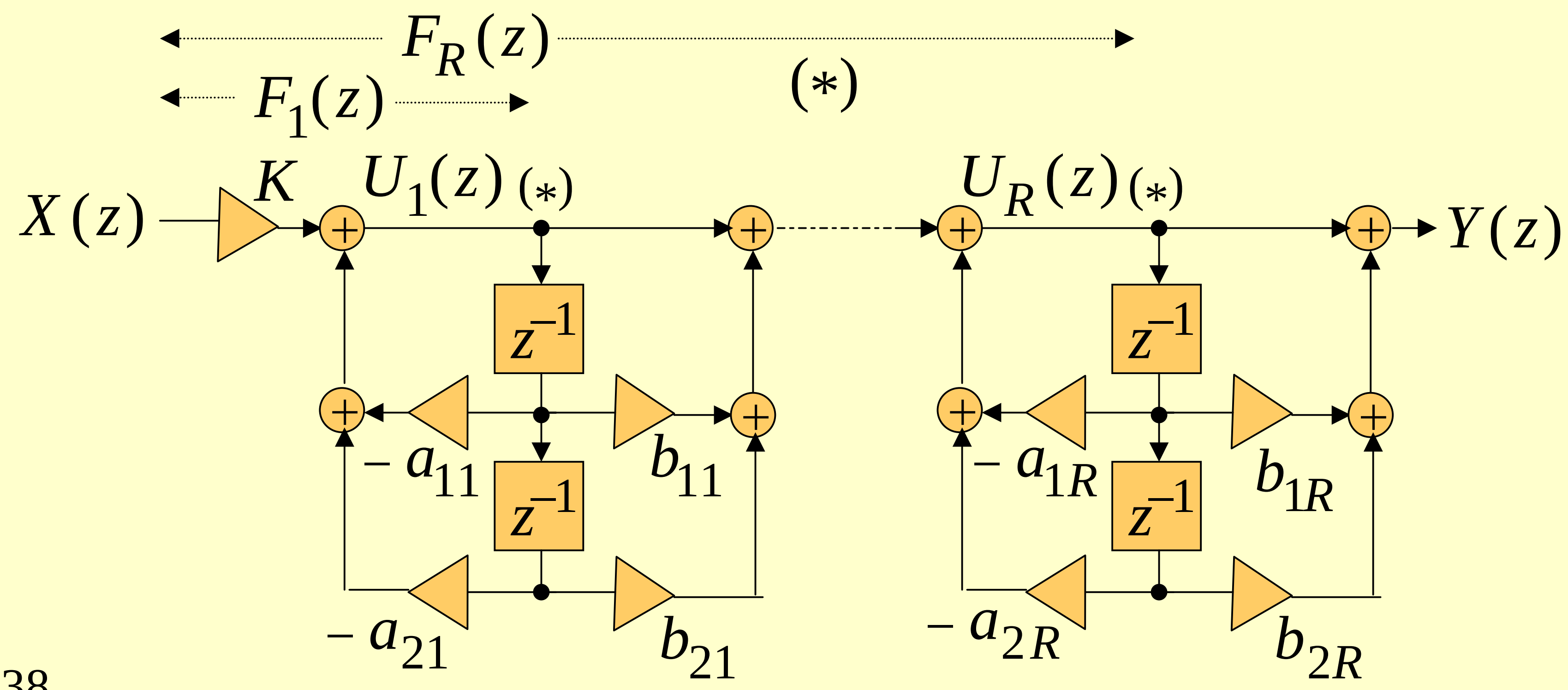
- An attractive option from a hardware point of view is to make as many unabsorbed scaling multipliers as possible in the scaled structure take a value that is a power of 2
- In which case, these scaling multipliers can be implemented simply by a shift operation
- The norm of the scaling transfer function for these multipliers then satisfies

$$\frac{1}{2} < \left\| \breve{F}_r \right\|_p \le 1$$

with a slight decrease in the SNR

Scaling of a Cascade Form IIR Digital Filter Structure

- Consider the unscaled structure consisting of R second-order IIR sections realized in direct form II



Scaling of a Cascade Form IIR Digital Filter Structure

- Its transfer function is given by

$$H(z) = K \prod_{i=1}^{R} H_i(z)$$

where

$$H_i(z) = \frac{B_i(z)}{A_i(z)} = \frac{1 + b_{1i} z^{-1} + b_{2i} z^{-2}}{1 + a_{1i} z^{-1} + a_{2i} z^{-2}}$$

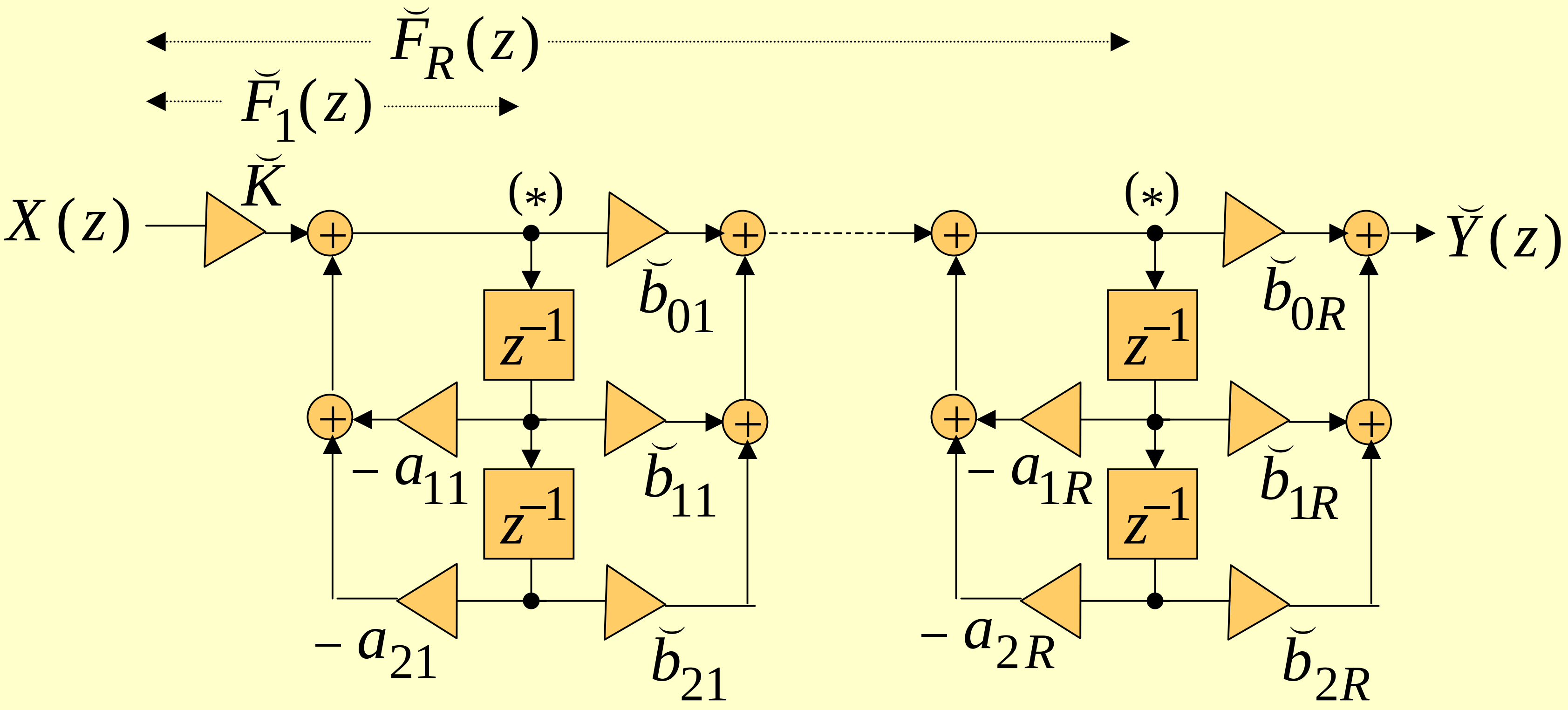
Scaling of a Cascade Form IIR Digital Filter Structure

- The branch nodes to be scaled are marked by $(*)$ which are seen to be the inputs to the multipliers in each second-order section
- The scaling transfer functions are given by

$$F_r(z) = \frac{K}{A_r(z)} \prod_{\ell=1}^{r-1} H_\ell(z), \; r = 1, 2, \ldots, R$$

Scaling of a Cascade Form IIR Digital Filter Structure

- The scaled version of the cascade structure is shown below



Scaling of a Cascade Form IIR Digital Filter Structure

- The scaling process has introduced a new multiplier $\breve{b}_{0\ell}$ in each second-order section
- If the zeros of the transfer function H(z) are on the unit circle, as is usually the case, then $b_{2\ell} = \pm 1$
- In which case we can choose $\breve{b}_{0\ell} = \breve{b}_{2\ell} = 2^{-\nu}$ to reduce the total number of multipliers in the final scaled structure

Scaling of a Cascade Form IIR Digital Filter Structure

- From the scaled structure it can be seen that

$$\breve{F}_r(z) = \frac{\breve{K}}{A_r(z)} \prod_{\ell=1}^{r-1} \breve{H}_\ell(z),$$

$$\breve{H}(z) = \breve{K} \prod_{\ell=1}^{R} \breve{H}_\ell(z)$$

where

$$\breve{H}_\ell(z) = \frac{\breve{b}_{0\ell} + \breve{b}_{1\ell} z^{-1} + \breve{b}_{2\ell} z^{-2}}{1 + a_{1\ell} z^{-1} + a_{2\ell} z^{-2}}$$

Scaling of a Cascade Form IIR Digital Filter Structure

- Denote

$$\|F_r\|_p \stackrel{\Delta}{=} \alpha_r, \quad r = 1, 2, \ldots, R$$

$$\|H\|_p \stackrel{\Delta}{=} \alpha_{R+1}$$

and choose the scaling constants as

$$\breve{K} = \beta_0 K; \quad \breve{b}_{\ell r} = \beta_r b_{\ell r}, \quad \ell = 0, 1, 2; \; r = 1, 2, \ldots, R$$

Scaling of a Cascade Form IIR Digital Filter Structure

- Then

$$\breve{F}_r(z) = \frac{\beta_0 K}{A_r(z)} \prod_{\ell=1}^{r-1} \beta_\ell H_\ell(z)$$

$$= \left(\prod_{\ell=0}^{r-1} \beta_\ell \right) F_r(z), \quad r = 1, 2, \ldots, R$$

$$\breve{H}(z) = \beta_0 K \prod_{\ell=0}^{R} \beta_\ell H_\ell(z) = \left(\prod_{\ell=0}^{R} \beta_\ell \right) H(z)$$

Scaling of a Cascade Form IIR Digital Filter Structure

- After scaling we require

$$\left\|\breve{F}_r\right\|_p = \left(\prod_{\ell=0}^{r-1} \beta_\ell\right)\left\|F_r\right\|_p = \alpha_r\left(\prod_{\ell=0}^{r-1} \beta_\ell\right) = 1, \; r = 1, 2, \ldots, R$$

$$\left\|\breve{H}\right\|_p = \left(\prod_{\ell=0}^{R} \beta_\ell\right)\left\|H\right\|_p = \alpha_{R+1}\left(\prod_{\ell=0}^{R} \beta_\ell\right) = 1$$

- Solving the above we get

$$\beta_0 = \frac{1}{\alpha_1}, \quad \beta_r = \frac{\alpha_r}{\alpha_{r+1}}, \quad r = 1, 2, \ldots, R$$

Dynamic Range Scaling Using MATLAB

- Dynamic range scaling using the $\mathcal{L}_2$-norm rule can be easily carried out using MATLAB by simulating the digital filter structure
- Denote the impulse response from the input to the *r*-th branch node as $\{f_r[n]\}$
- Assume that the branch nodes have been ordered in accordance with their precedence relations with increasing r

Dynamic Range Scaling Using MATLAB

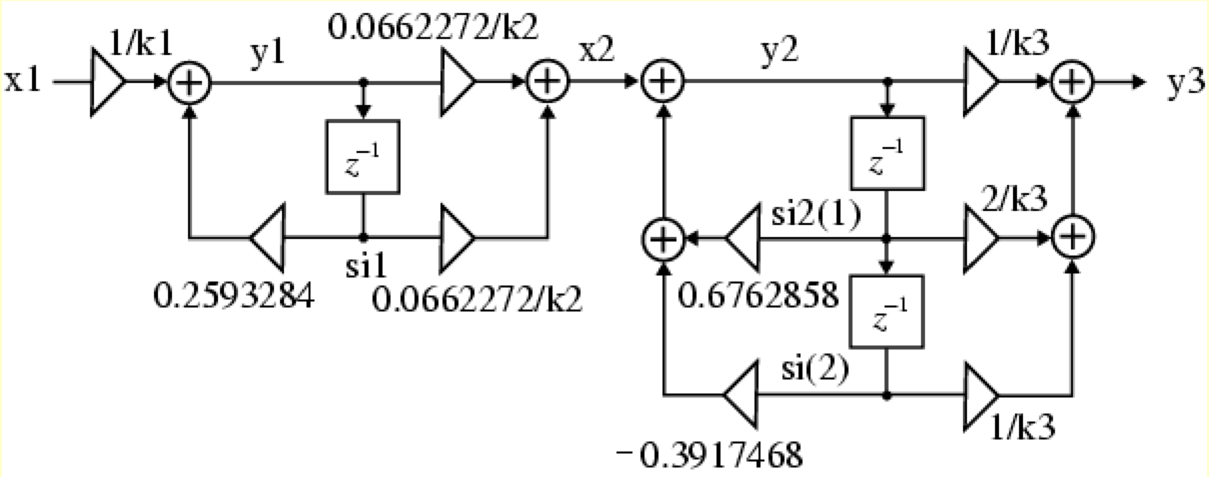
- Compute first the $\mathcal{L}_2$-norm $\|F_1\|_2$ of $\{f_1[n]\}$ and scale the input by a multiplier $k_1 = \|F_1\|_2$
- Next, compute the $\mathcal{L}_2$-norm $\|F_2\|_2$ of $\{f_2[n]\}$ and scale the multipliers feeding into then second adder by dividing with a constant $k_2 = \|F_2\|_2$
- Continue the process until the output node has been scaled to yield an $\mathcal{L}_2$-norm of unity

Dynamic Range Scaling Using MATLAB

- Example - Consider the cascade realization of

$$H_1(z) = \frac{0.0662272(1+z^{-1})}{1-0.2593284z^{-1}}$$

$$H_2(z) = \frac{1+2z^{-1}+z^{-2}}{1-0.6762858z^{-1}+0.3917468z^{-2}}$$



Dynamic Range Scaling Using MATLAB

```
k1 = 1; k2 = 1; k3 = 1;
x1 = 1/k1;
si1 = 0; si2 = [0 0];
varnew = 0; k= 1
while k > 0.0001
    y1 = 9.2593284*si1 + x1;
    x2 = (0.0662272/k2) *(y1 + si1);
    si1 = y1;
    y2 = 0.6762858*si2(1) - 0.3917468*si2(2) + x2;
    si2(2) = si2(1); si2(1) = y2;
    varold = varnew;
    varnew = varnew + abs(y1)*abs(y1);
    k = varnew - varold;
    x1 = 0;
end
```

Dynamic Range Scaling Using MATLAB

- The MATLAB program simulating the cascaded structure is given by Program 9_6 in text
- The program is first run with all scaling constants set to unity, i.e., k1 = k2 = k3 =1
- In the statement computing the approximate value of the $\mathcal{L}_2$-norm, the output variable is chosen as y1
- The program computes the square of the $\mathcal{L}_2$-norm at node y1 as 1.07210002757252

Dynamic Range Scaling Using MATLAB

```
k1 = sqrt(1.07210002757252); k2 = 1; k3 = 1;
x1 = 1/k1;
si1 = 0; si2 = [0 0];
varnew = 0; k= 1
while k > 0.0001
    y1 = 9.2593284*si1 + x1;
    x2 = (0.0662272/k2) *(y1 + si1);
    si1 = y1;
    y2 = 0.6762858*si2(1) - 0.3917468*si2(2) + x2;
    si2(2) = si2(1); si2(1) = y2;
    varold = varnew;
    varnew = varnew + abs(y1)*abs(y1);
    k = varnew - varold;
    x1 = 0;
end
```

Dynamic Range Scaling Using MATLAB

- For the next run of the program, we set k1 = $= \sqrt{1.07210002757252}$ with other scaling constants still set to unity
- A second run of the program shows the $\mathcal{L}_2$-norm of the impulse response at node y1 as 1.0 verifying the success of scaling the input
- In the second step, in the statement computing the approximate value of the $\mathcal{L}_2$-norm, the output variable is chosen as y2

Dynamic Range Scaling Using MATLAB

```
k1 = sqrt(1.07210002757252); k2 = 1; k3 = 1;
x1 = 1/k1;
si1 = 0; si2 = [0 0];
varnew = 0; k= 1
while k > 0.0001
    y1 = 9.2593284*si1 + x1;
    x2 = (0.0662272/k2) *(y1 + si1);
    si1 = y1;
    y2 = 0.6762858*si2(1) - 0.3917468*si2(2) + x2;
    si2(2) = si2(1); si2(1) = y2;
    varold = varnew;
    varnew = varnew + abs(y2)*abs(y2);
    k = varnew - varold;
    x1 = 0;
end
```

Dynamic Range Scaling Using MATLAB

- The program yields the square of the $\mathcal{L}_2$-norm of the impulse response at node y2 as 0.02679820762398, which is used to set k2 $= \sqrt{0.02679820762398}$ with k3 still set to unity

Dynamic Range Scaling Using MATLAB

```
k1 = sqrt(1.0721002757252);
k2 = sqrt(0.2679820762398); k3 = 1;
x1 = 1/k1;
si1 = 0; si2 = [0 0];
varnew = 0; k= 1
while k > 0.0001
      y1 = 9.2593284*si1 + x1;
      x2 = (0.0662272/k2) *(y1 + si1);
      si1 = y1;
      y2 = 0.6762858*si2(1) - 0.3917468*si2(2) + x2;
      si2(2) = si2(1); si2(1) = y2;
      varold = varnew;
      varnew = varnew + abs(y3)*abs(y3);
      k = varnew - varold;
      x1 = 0;
end
```

Dynamic Range Scaling Using MATLAB

- The process is repeated for node y3, resulting in $k3 = \sqrt{11.96975400608943}$

- The final value of the $\mathcal{L}_2$ -norm of the impulse response at node y3 is 0.99999683

Dynamic Range Scaling Using MATLAB

```
k1 = sqrt(1.07210002757252);
k2 = sqrt(0.2679820762398); k3 = sqrt(11.9675400608943);
x1 = 1/k1;
si1 = 0; si2 = [0 0];
varnew = 0; k= 1
while k > 0.0001
      y1 = 9.2593284*si1 + x1;
      x2 = (0.0662272/k2) *(y1 + si1);
      si1 = y1;
      y2 = 0.6762858*si2(1) - 0.3917468*si2(2) + x2;
      si2(2) = si2(1); si2(1) = y2;
      varold = varnew;
      varnew = varnew + abs(y3)*abs(y3);
      k = varnew - varold;
      x1 = 0;
end
```

Product Round-Off Noise Calculation Using MATLAB

- Program 9_6 can be easily modified to calculate the product round-off noise variance at the output of the scaled structure
- To this end, we set the digital filter input to zero and apply an impulse at the input of the first adder
- This is equivalent to setting x1 = 1 in the program
- The normalized output noise variance due to a single noise source is 1.077209663042567

Product Round-Off Noise Calculation Using MATLAB

- Next, we apply an impulse at the input of the second adder with the digital filter input set to zero
- This is achieved by replacing x2 in the calculation of y2 with x1
- The program yields the normalized output noise variance due to a single error source at the second adder as 1.26109014071707

Product Round-Off Noise Calculation Using MATLAB

- The total normalized output noise variance, assuming all products to be quantized before addition, is

$$2 \times 1.07209663042567 + 4 \times 1.26109014071707 + 3$$

$$= 10.18855382371962$$

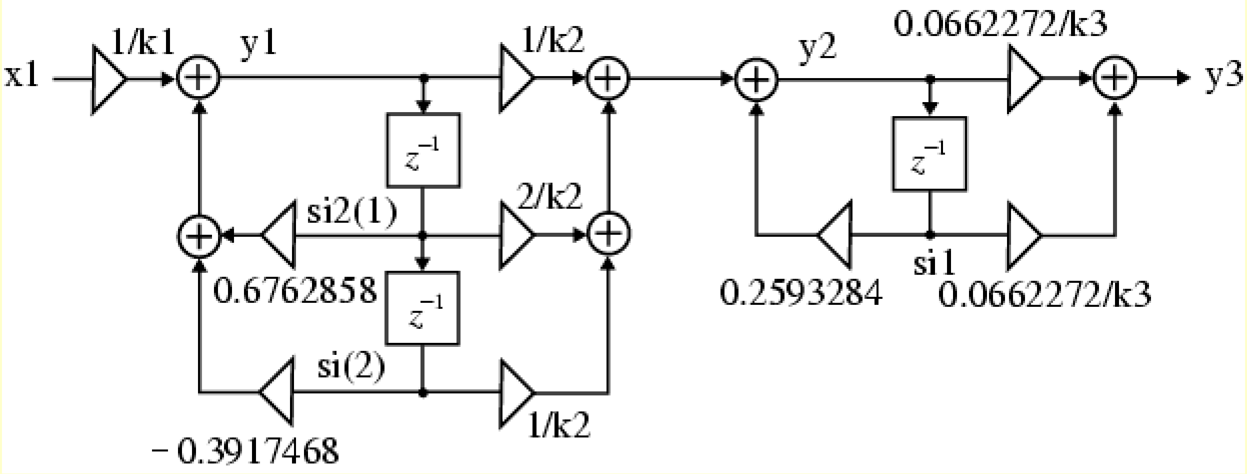
- On the the hand, for quantization after addition of products, the total normalized output noise variance is

$$1.07209663042567 + 1.26109014071707 + 1$$

$$= 3.3318677114274$$

Product Round-Off Noise Calculation Using MATLAB

- Example - We interchange the locations of the two sections in the cascade



Product Round-Off Noise Calculation Using MATLAB

- In this case, the total normalized output noise variance, assuming all products to be quantized before addition, is

$$3\times 1.5465221 + 4\times 0.7693895 + 2 = 9.7171242$$

- On the the hand, for quantization after addition of products, the total normalized output noise variance is

$$1.5465221 + 0.7693895 + 1 = 3.3159116$$