

# **BÀI GIẢNG XỬ LÝ SỐ TÍN HIỆU**

**Biên soạn: PGS.TS LÊ TIẾN THƯỜNG**

**Tp.HCM, 02-2005**

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

7.1. Dạng trực tiếp.

7.2. Dạng canonical.

7.3. Dạng Cascade.

7.4. Cascade sang canonical.

7.5. Cài đặt phần cứng và các bộ đệm vòng.

7.6. Các ảnh hưởng của quá trình lượng tử hóa trong mạch lọc số.

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.1. Dạng trực tiếp.

Xét một mạch lọc đơn giản có hàm truyền đạt như sau:

$$H(z) = \frac{N(z)}{D(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{a_0 + a_1 z^{-1} + a_2 z^{-2}} \quad (7.1.1)$$

Ta có phương trình sai phân vào/ra

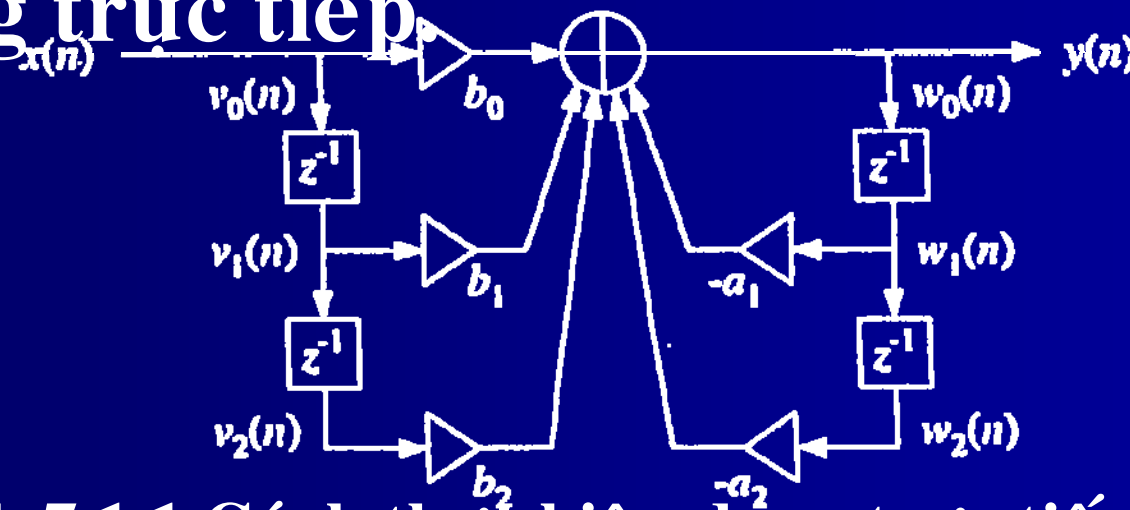
$$y_n = -a_1 y_{n-1} - a_2 y_{n-2} + b_0 x_n + b_1 x_{n-1} + b_2 x_{n-2} \quad (7.1.2)$$

Các biểu diễn dạng trực tiếp là sơ đồ khối biểu diễn phương trình sai phân này, được minh họa trong hình 7.1.1.

Dạng trực tiếp FIR của chương 4 có được từ một trường hợp đặt biệt của dạng này khi cho các hệ số hồi tiếp  $a_1 = a_2 = 0$ .

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.1. Dạng trực tiếp



Hình 7.1.1 Cách thực hiện dạng trực tiếp của mạch lọc IIR bậc 2.

Cách biểu diễn dạng trực tiếp có thể được tổng quát hóa cho trường hợp các tử và mẫu thức bất kỳ.

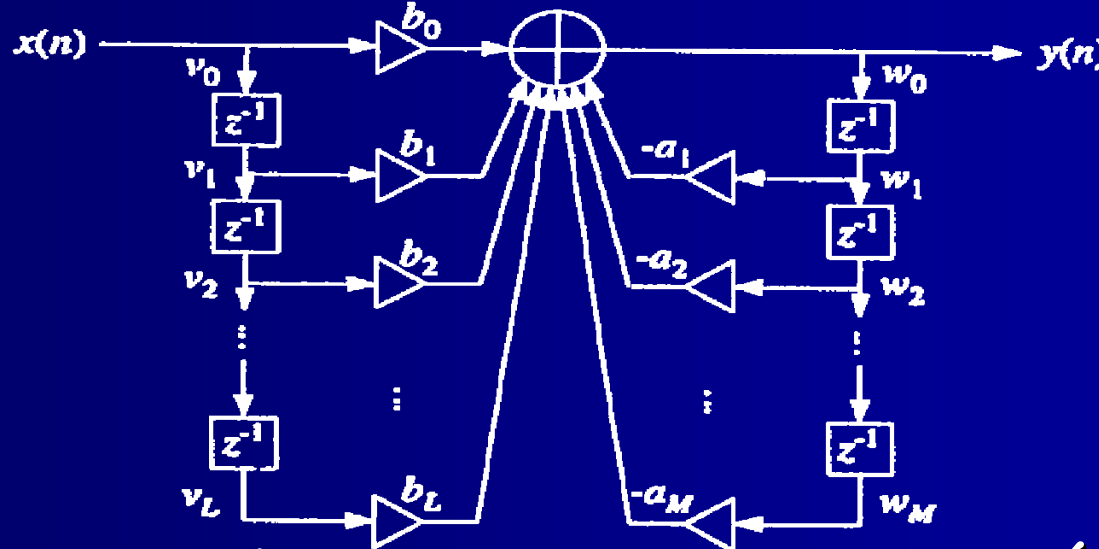
$$H(z) = \frac{N(z)}{D(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_L z^{-L}}{a_0 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_M z^{-M}} \quad (7.1.4)$$

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.1. Dạng trực tiếp.

Có tử bậc N và mẫu bậc M. Phương trình sai phân vào/ra tương ứng là:

$$y_n = -a_1 y_{n-1} - a_2 y_{n-2} - \dots - a_M y_{n-M} + b_0 x_n + b_1 x_{n-1} + b_2 x_{n-2} + \dots + b_L x_{n-L} \quad (7.1.5)$$



Hình 7.1.2: Cách thực hiện dạng trực tiếp của mạch lọc IIR bậc M

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.1. Dạng trực tiếp.

**Ví dụ 7.1.1:** Vẽ cách thực hiện dạng trực tiếp của mạch lọc sau  $H(z) = \frac{2 - 3z^{-1} + 4z^{-3}}{1 + 0.2z^{-1} - 0.3z^{-2} + 0.5z^{-4}}$  và xác định các phương trình sai phân và thuật toán xử lý mẫu.

**Giải:** Phương trình sai phân là:

$$y_n = -0.2y_{n-1} + 0.3y_{n-2} - 0.5y_{n-4} + 2x_n - 3x_{n-1} + 4x_{n-3}$$

**Các vector hệ số và trạng thái trong ví dụ này:**

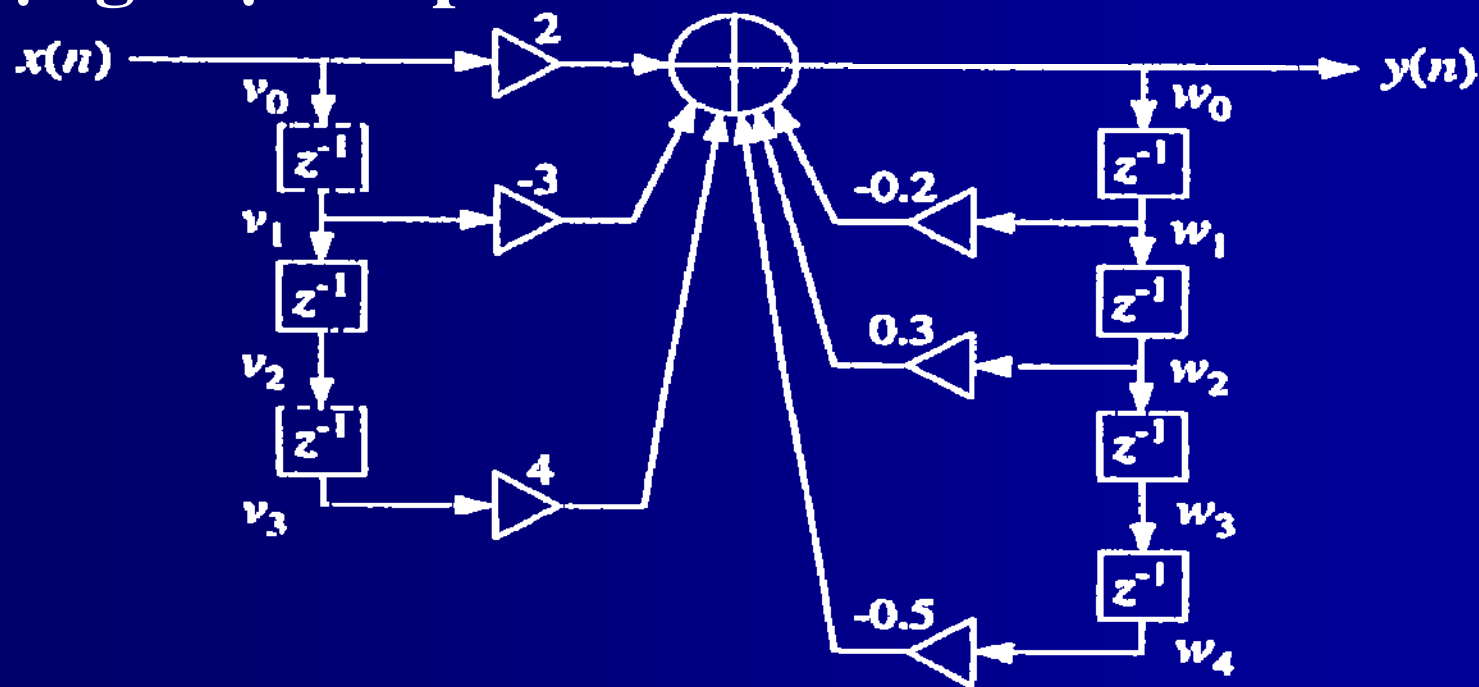
$$a = [a_0, a_1, a_2, a_3, a_4] = [1, 0.2, -0.3, 0.0, 0.5]$$

$$b = [b_0, b_1, b_2, b_3] = [2, -3, 0, 4]$$

$$w = [w_0, w_1, w_2, w_3, w_4], \quad v = [v_0, v_1, v_2, v_3]$$

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.1. Dạng trực tiếp.



Hình 7.1.3

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.2. Dạng Canonical

Dạng biểu diễn canonical, hay là dạng trực tiếp II, các hệ số recursive và các hệ số non-recursive, có nghĩa là:

$$y_n = (b_0x_n + b_1x_{n-1} + b_2x_{n-2}) + (-a_1y_{n-1} - a_2y_{n-2})$$

Việc nhóm lại tương ứng với việc chia một bộ cộng lớn của cách biểu diễn dạng trực tiếp của hình 7.1.1 thành 2 phần như trong hình 7.2.1.

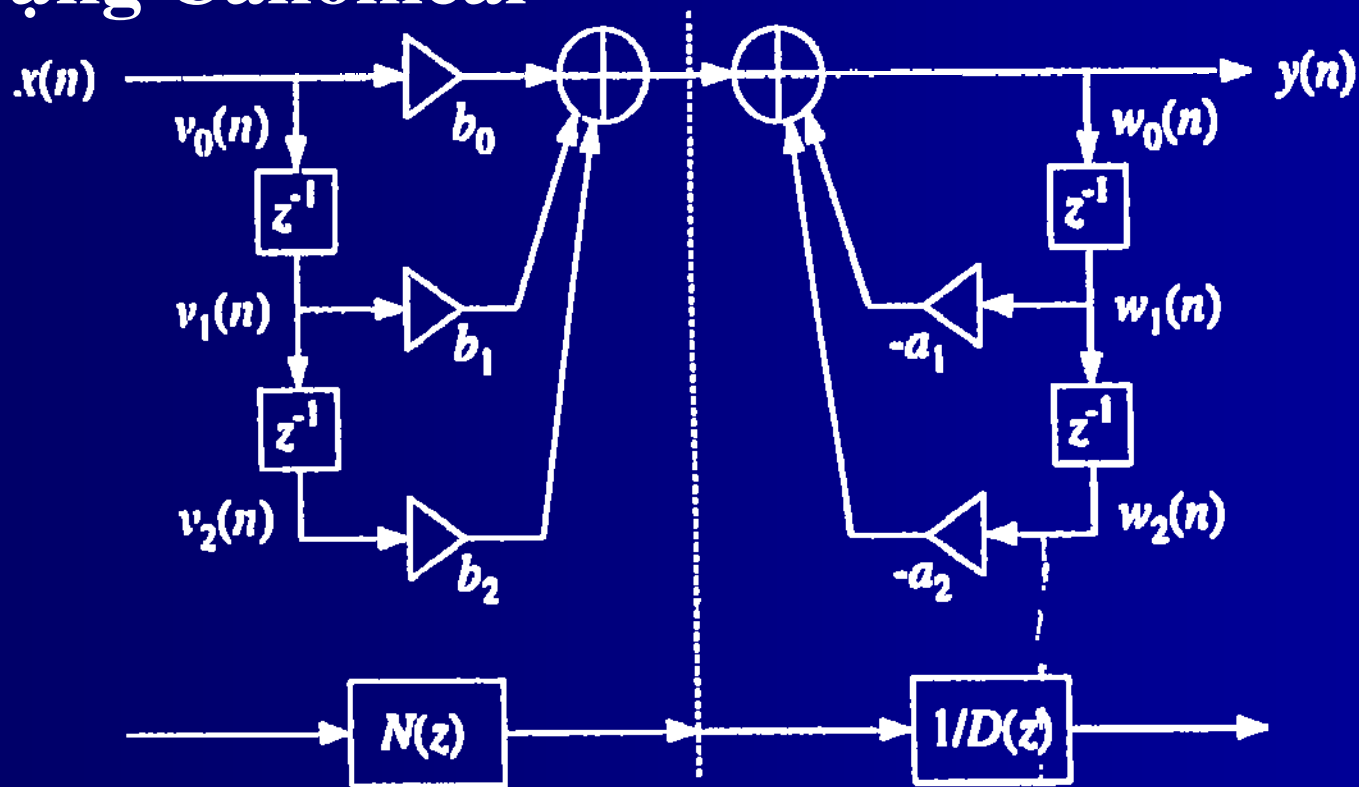
Cũng chính là hàm truyền đạt gốc được cho trong phương trình (7.1.1). Về mặt toán học, bậc của các hệ số cascade có thể được thay đổi sao cho

$$H(z) = \frac{1}{D(z)} N(z)$$



# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

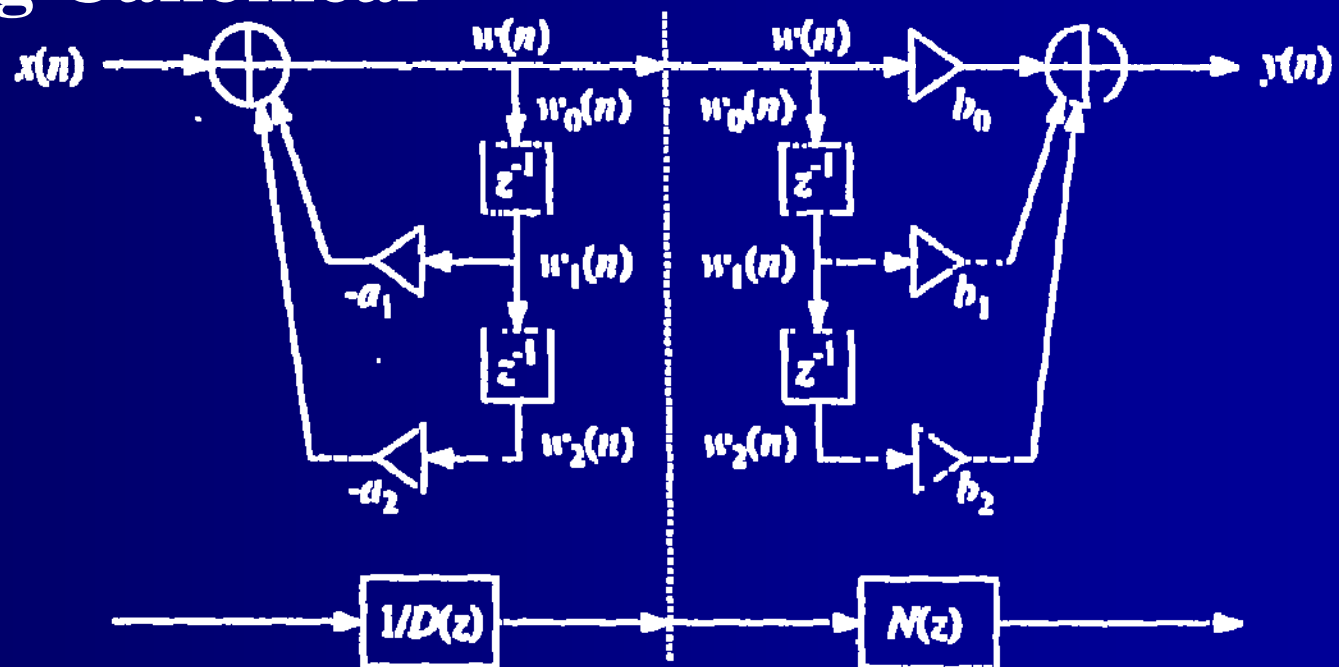
## 7.2. Dạng Canonical



Hình 7.2.1

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

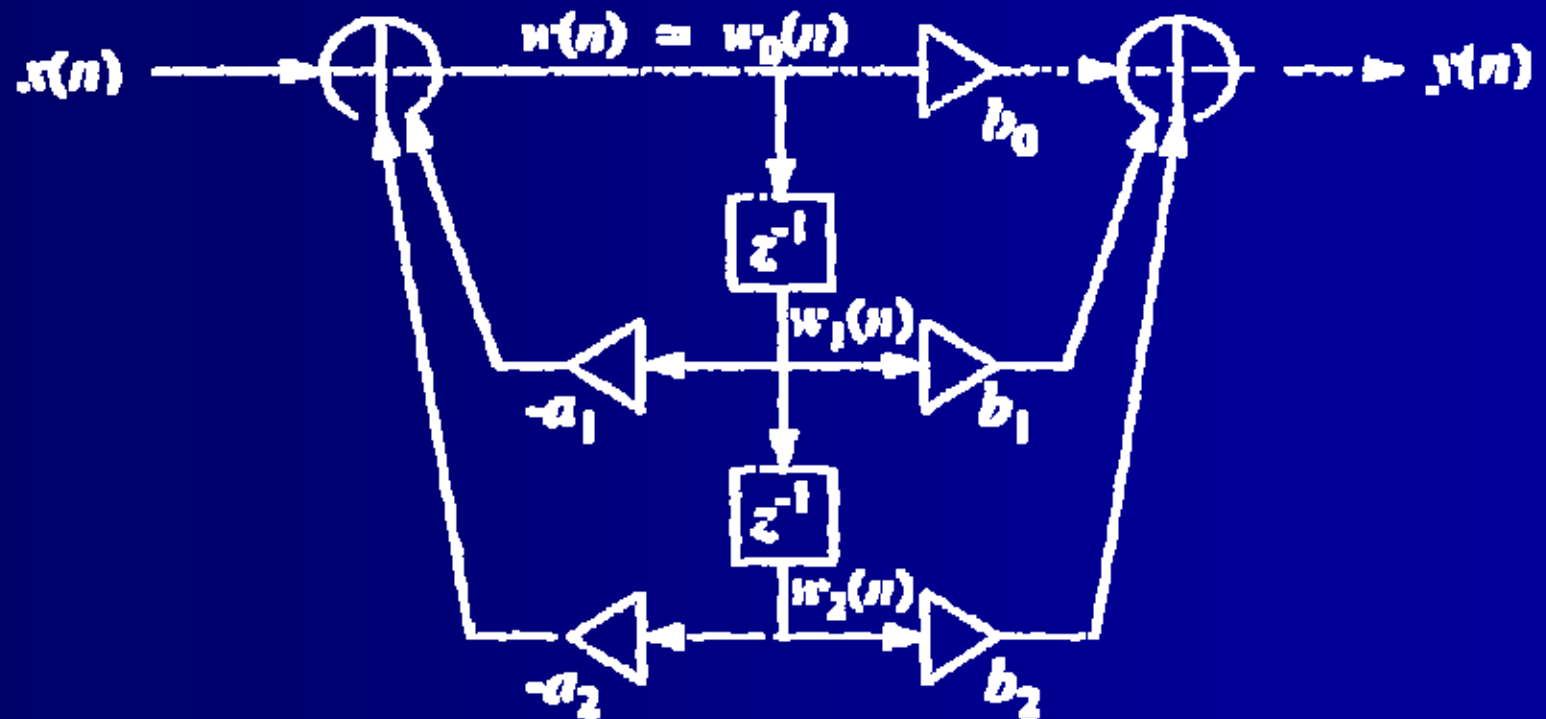
## 7.2. Dạng Canonical



Hình 7.2.2 Thay đổi  $N(z)$  và  $1/D(z)$ .

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

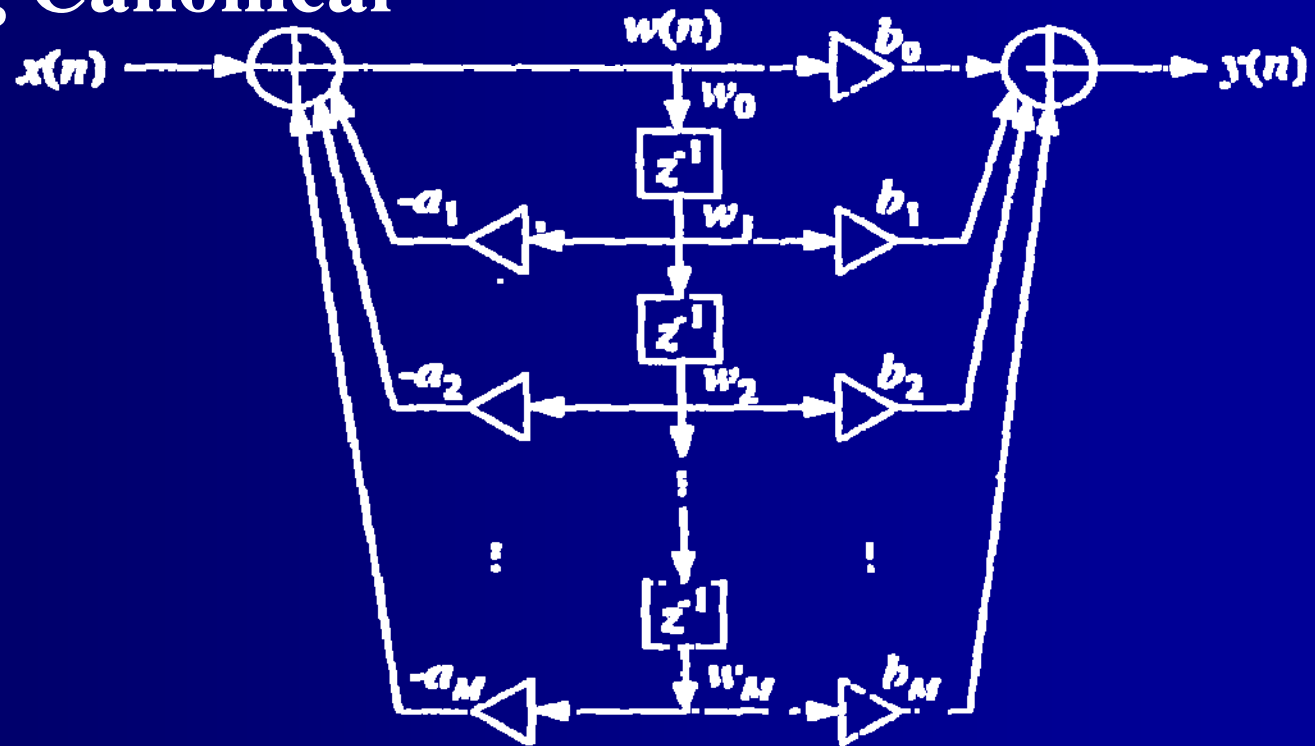
## 7.2. Dạng Canonical



Hình 7.2.3: Dạng chính tắc của bộ lọc IIR bậc hai

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.2. Dạng Canonical



Hình 7.2.4: Thực hiện canonical của mạch lọc IIR bậc M

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.2. Dạng Canonical

So sánh các hình 7.1.2 và 7.2.4, ta lưu ý rằng:

- a) Dạng trực tiếp đòi hỏi lượng khâu trễ gấp đôi.
- b) Cả hai đều có cùng các hệ số nhân.
- c) Dạng trực tiếp chỉ có một bộ cộng mà đáp ứng của nó là đáp ứng của hệ thống.
- d) Dạng canonical có hai bộ cộng, một ở đầu vào và một ở đầu ra.

Trong hai dạng này, dạng canonical được sử dụng rộng rãi hơn trong thực tế.

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.2. Dạng Canonical

Lưu ý: đối với các mạch lọc FIR có đa thức mẫu  $D(z) = 1$  và do đó các dạng trực tiếp và canonical tương tự với dạng trực tiếp của chương 4.

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.3. Dạng Cascade

Dạng biểu diễn cascade của một hàm truyền tổng quát giả sử hàm truyền là tích các phần bậc hai như thế:

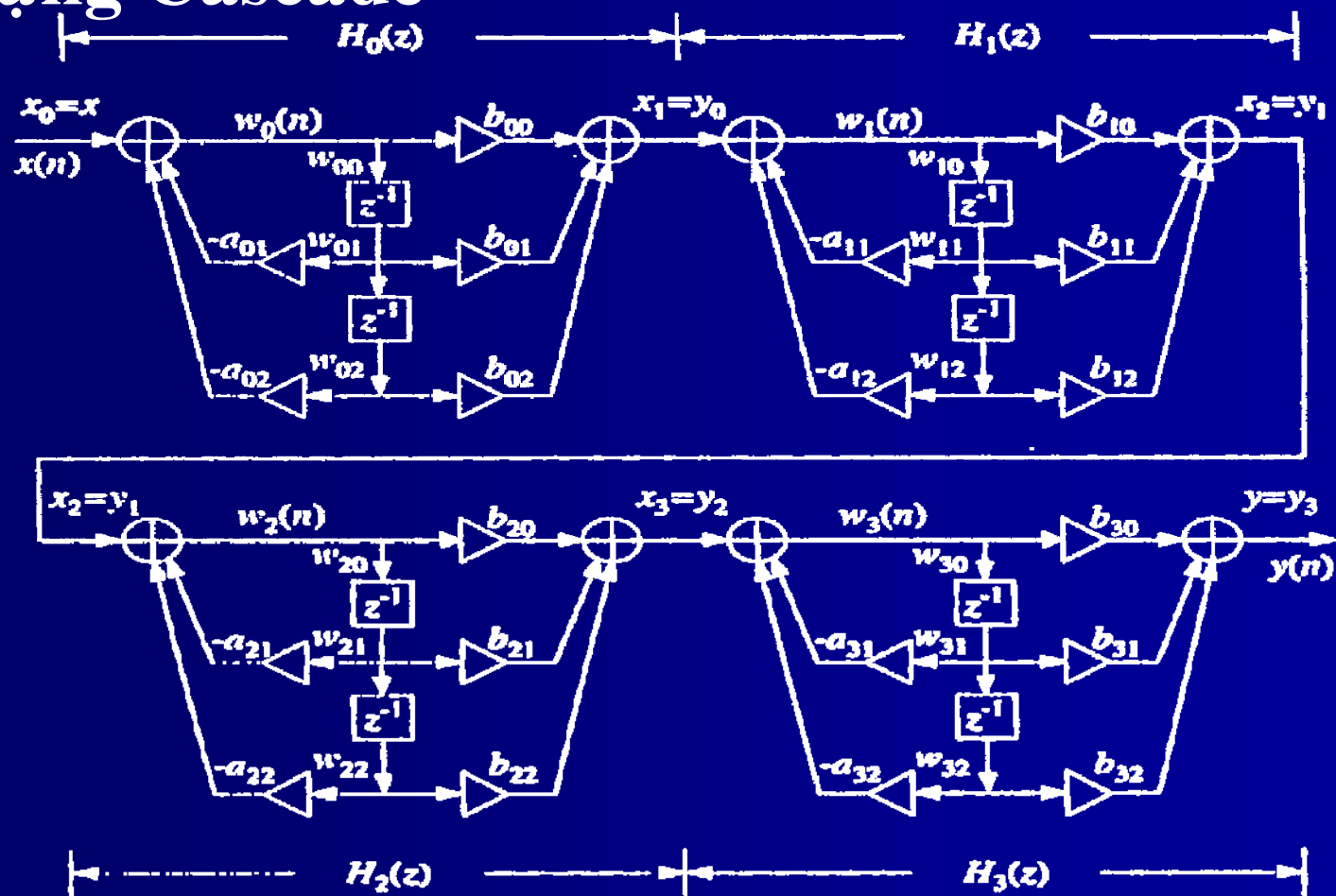
$$H(z) = \prod_{i=0}^{K-1} H_i(z) = \prod_{i=0}^{K-1} \frac{b_{i0} + b_{i1}z^{-1} + b_{i2}z^{-2}}{1 + a_{i1}z^{-1} + a_{i2}z^{-2}} \quad (7.3.1)$$

Hàm truyền bất kỳ có dạng (7.1.4) có thể được phân tích thành các thừa số bậc hai với các hệ số thực, nếu phương trình (7.1.4) có các hệ số thực.

Để theo dõi các hệ số của các phần và các trạng thái trong, ta xếp chúng thành các ma trận  $K \times 3$  có hàng thứ  $i$  là các thông số tương ứng của phần thứ  $i$ . Ví dụ, nếu  $K=4$  như trong hình 7.3.1, ta định nghĩa:

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.3. Dạng Cascade



Hình 7.3.1 Cascade của 2 khâu bậc 2



# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.3. Dạng Cascade

$$A = \begin{bmatrix} 1 & a_{01} & a_{02} \\ 1 & a_{11} & a_{12} \\ 1 & a_{21} & a_{22} \\ 1 & a_{31} & a_{32} \end{bmatrix}, \quad B = \begin{bmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \\ b_{20} & b_{21} & b_{22} \\ b_{30} & b_{31} & b_{32} \end{bmatrix}, \quad W = \begin{bmatrix} w_{00} & w_{01} & w_{02} \\ w_{10} & w_{11} & w_{12} \\ w_{20} & w_{21} & w_{22} \\ w_{30} & w_{31} & w_{32} \end{bmatrix}$$

Hàng thứ  $i$  của các ma trận này là các vector hệ số 3 chiều và các trạng thái của phần thứ  $i$ , có nghĩa là:

$$a_i = [1, a_{i1}, a_{i2}] \quad (7.3.4)$$

$$b_i = [b_{i0}, b_{i1}, b_{i2}] \quad i = 0, 1, \dots, K-1$$

$$w_i = [w_{i0}, w_{i1}, w_{i2}]$$

**Ví dụ 7.3.1:** Vẽ các cách thực hiện dạng cascade và canonical của các mạch lọc sau:

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.3. Dạng Cascade

$$\begin{aligned} H(z) &= \left[ \frac{3 - 4z^{-1} + 2z^{-2}}{1 - 0.4z^{-1} + 0.5z^{-2}} \right] \left[ \frac{3 + 4z^{-1} + 2z^{-2}}{1 + 0.4z^{-1} + 0.5z^{-2}} \right] = H_0(z)H_1(z) \\ &= \frac{9 - 4z^{-2} + 4z^{-4}}{1 - 0.84z^{-2} + 0.25z^{-4}} \end{aligned}$$

**Viết phương trình sai phân I/O tương ứng và các thuật toán xử lý mẫu.**

**Giải:** Các ma trận trạng thái và hệ số trong hàm cas trong trường hợp này là:

$$A = \begin{bmatrix} 1 & -0.4 & 0.5 \\ 1 & 0.4 & 0.5 \end{bmatrix}, \quad B = \begin{bmatrix} 3 & -4 & 2 \\ 3 & 4 & 2 \end{bmatrix}, \quad W = \begin{bmatrix} w_{00} & w_{01} & w_{02} \\ w_{10} & w_{11} & w_{12} \end{bmatrix}$$

**Trong trường hợp canonical, t có các vector hệ số cho tử thức và mẫu thức là:**

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.3. Dạng Cascade

$$b = [9, 0, -4, 0, 4], \quad a = [1.00, 0.00, 0.84, 0.00, 0.25]$$

Phương trình sai phân tại các bộ cộng đầu vào và đầu ra của hình 7.3.3:

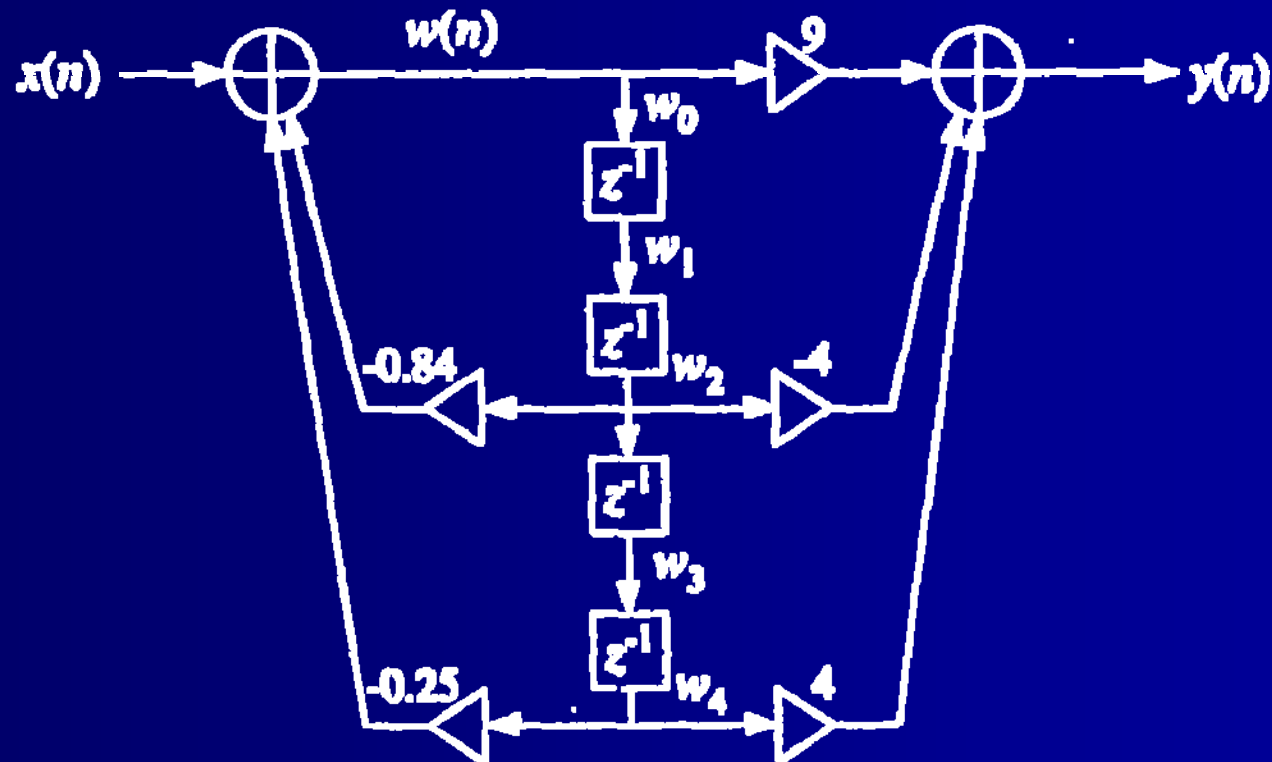
$$w(n) = x(n) - 0.84w(n-2) - 0.25w(n-4)$$

$$y(n) = 9w(n) - 4w(n-2) + 4w(n-4)$$

**Tổng số trạng thái trong trong các cách thực hiện dạng trực tiếp và canonical là giống nhau, đều bằng 4.**

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.3. Dạng Cascade



Hình 7.3.3 Cách thực hiện dạng canonical của ví dụ 7.3.1.

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.4. Cascade sang canonical

Để chuyển từ cách biểu diễn trực tiếp sang canonical, phương trình (7.1.4), sang cách biểu diễn cascade, phương trình (7.3.1), đòi hỏi việc phân tích các tử thức và mẫu thức thành các thừa số bậc hai.

Điều này có thể được thực hiện bằng cách tìm các nghiệm của các đa thức này rồi nhóm chúng thành từng cặp là liên hợp phức với nhau.

$$\begin{aligned} D(z) &= 1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_M z^{-M} \\ &= (1 - p_1 z^{-1})(1 - p_2 z^{-1}) \cdots (1 - p_M z^{-1}) \end{aligned}$$

Các thừa số nghiệm của các nghiệm thực bất kỳ có thể được bỏ qua khi chúng kết hợp thành từng cặp.

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.4. Cascade sang canonical

Ví dụ, khi cả  $p_1$  và  $p_2$  đều là thực, ta có thể kết hợp chúng thành SOS với các hệ số thực:

$$(1 - p_1 z^{-1})(1 - p_2 z^{-1}) = 1 - (p_1 + p_2)z^{-1} + p_1 p_2 z^{-2}$$

Nếu một nghiệm nào đó là phức, chúng phải là một cặp liên hợp phức, ví dụ, nếu  $p_1$  là một nghiệm phức, thế thì  $p_2 = p_1^*$  cũng phải là một nghiệm. Kết hợp các thừa số nghiệm của các cặp nghiệm liên hợp thành một SOS với các hệ số thực, ví dụ:

$$\begin{aligned}(1 - p_1 z^{-1})(1 - p_1^* z^{-1}) &= 1 - (p_1 + p_1^*)z^{-1} + p_1 p_1^* z^{-2} \\ &= 1 - 2 \operatorname{Re}(p_1)z^{-1} + |p_1|^2 z^{-2}\end{aligned}$$

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.4. Cascade sang canonical

Cách này cũng đã được sử dụng trong chương 5. Dùng cách biểu diễn số phức trong tọa độ cực  $p_1 = R_1 e^{j\theta_1}$ , ta có  $Re(p_1) = R_1 \cos \theta_1$  và  $|p_1|^2 = R_1^2$ , ta có thể viết lại phương trình trên như sau:

$$\begin{aligned}(1 - p_1 z^{-1})(1 - p_1^* z^{-1}) &= 1 - 2Re(p_1)z^{-1} + |p_1|^2 z^{-2} \\ &= 1 - 2R_1 \cos(\theta_1)z^{-1} + R_1^2 z^{-2}\end{aligned}$$

Khi các tử thức và mẫu thức đã được phân tích thành các thừa số bậc hai, mỗi thừa số bậc hai trên tử có thể đi cặp với một thừa số bậc hai dưới mẫu để tạo thành một phân bậc hai.

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.4. Cascade sang canonical

Việc ghép cặp các thừa số tử thức và mẫu thức và việc sắp xếp các SOS không phải là duy nhất, nhưng hàm truyền tổng sẽ giống nhau. Đối với các đa thức bậc cao, ta phải sử dụng đến các hàm tìm nghiệm được cho trong các package phần mềm toán học như là Matlab hay Mathematica. Trong một vài trường hợp đặc biệt của đa thức bậc cao, ta có thể tính toán được nghiệm bằng tay, như ví dụ sau

**Ví dụ 7.4.1:** Xác định dạng thực hiện cascade của mạch lọc:

$$H(z) = \frac{1 - 1.5z^{-1} + 0.48z^{-2} - 0.33z^{-3} + 0.9376z^{-4} - 0.5328z^{-5}}{1 + 2.2z^{-1} + 1.77z^{-2} + 0.52z^{-3}}$$



# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.4. Cascade sang canonical

**Giải:** Dùng MATLAB, ta tìm được 5 nghiệm của tử thức:  $z = 0.9$ ,  $z = -0.5 \pm 0.7j$ ,  $z = 0.8 \pm 0.4j$

Chúng dẫn đến các thừa số nghiệm, theo cặp liên hợp phức:

$$(1 - 0.9z^{-1})$$
$$(1 - (-0.5 + 0.7j)z^{-1})(1 - (-0.5 - 0.7j)z^{-1}) = (1 + z^{-1} + 0.74z^{-2})$$
$$(1 - (0.8 + 0.4j)z^{-1})(1 - (0.8 - 0.4j)z^{-1}) = (1 - 1.6z^{-1} + 0.8z^{-2})$$

**Tương tự, ta tìm các nghiệm của mẫu thức:**

$p = -0.8$ ,  $p = -0.7 \pm 0.4j$  cho ta các thừa số nghiệm

$$(1 + 0.8z^{-1})$$
$$(1 - (-0.7 + 0.4j)z^{-1})(1 - (-0.7 - 0.4j)z^{-1}) = (1 + 1.4z^{-1} + 0.65z^{-2})$$

**Do đó, tất cả các cách ghép cặp/xếp thứ tự của các khâu SOS cho  $H(z)$ :**

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.4. Cascade sang canonical

$$H(z) = \frac{1 - 0.9z^{-1}}{1 + 0.8z^{-1}} \cdot \frac{1 + z^{-1} + 0.74z^{-2}}{1 + 1.4z^{-1} + 0.65z^{-2}} \cdot (1 - 1.6z^{-1} + 0.8z^{-2})$$

Các ma trận hệ số A và B cần thiết cho việc lập trình cho mạch lọc này bằng hàm cas sẽ là:

$$A = \begin{bmatrix} 1 & 0.8 & 0 \\ 1 & 1.4 & 0.65 \\ 1 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & -0.9 & 0 \\ 1 & 1 & 0.74 \\ 1 & -1.6 & 0.8 \end{bmatrix}$$

Khâu bậc nhất của thể được xem như là một trường hợp đặc biệt của một SOS có dạng (7.1.1) với hệ số  $z^{-2}$  bằng zero, có nghĩa là  $b_2 = a_2 = 0$ . Tương tự, thừa số bậc hai cuối cùng là trường hợp đặc biệt của một SOS FIR, có nghĩa là với  $a_1 = a_2 = 0$  (nhưng  $a_0 = 1$ ).

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.4. Cascade sang canonical

**Ví dụ 7.4.3:** Một trường hợp đặc biệt khác, xác định dạng thực hiện cascade của mạch lọc:  $H(z) = \frac{1 - z^{-8}}{1 - 0.0625z^{-8}}$

**Giải:** Nghiệm của tử thức là 8 nghiệm của phương trình

$$1 + z^{-8} = 0 \Rightarrow z^{-8} = -1 = e^{j\pi} = e^{j\pi} e^{2\pi jk} = e^{j(2k+1)\pi}$$

trong đó  $k$  là số tự nhiên. Lấy căn bậc 8 của 2 vế

$$z_k = e^{j(2k+1)\pi/8}, \quad k = 0, 1, \dots, 7$$

Ta có các cặp nghiệm liên hợp phức sau, như trong hình 7.4.1.  $\{z_0, z_7\}, \{z_1, z_6\}, \{z_2, z_5\}, \{z_3, z_4\}$

Chúng dẫn đến các thừa số nghiệm, theo cặp liên hợp phức:

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.4. Cascade sang canonical

$$(1 - z_0 z^{-1})(1 - z_7 z^{-1}) = 1 - 2 \cos\left(\frac{\pi}{8}\right) z^{-1} + z^{-2} = 1 - 1.8478 z^{-1} + z^{-2}$$

$$(1 - z_1 z^{-1})(1 - z_6 z^{-1}) = 1 - 2 \cos\left(\frac{3\pi}{8}\right) z^{-1} + z^{-2} = 1 - 0.7654 z^{-1} + z^{-2}$$

$$(1 - z_2 z^{-1})(1 - z_5 z^{-1}) = 1 - 2 \cos\left(\frac{5\pi}{8}\right) z^{-1} + z^{-2} = 1 + 0.7654 z^{-1} + z^{-2}$$

$$(1 - z_3 z^{-1})(1 - z_4 z^{-1}) = 1 - 2 \cos\left(\frac{7\pi}{8}\right) z^{-1} + z^{-2} = 1 + 1.8478 z^{-1} + z^{-2}$$

**Tương tự, ta tìm các nghiệm của mẫu thức:**

$$1 - 0.0625 z^{-8} = 0 \Rightarrow z^8 = -0.0625 = 0.0625 e^{2\pi j k} = (0.5)^4 e^{j(2k+1)\pi}$$

**Lấy căn bậc 8 hai vế ta được:**  $p_k = \sqrt[4]{0.5} e^{2\pi j k / 8}, \quad k = 0, 1, \dots, 7$

**Trong các nghiệm đó, là thực và có thể được ghép cặp với nhau thành một SOS. Các nghiệm còn lại là phức và được ghép thành từng cặp liên hợp:**

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.4. Cascade sang canonical

$$\{p_0, p_7\}, \{p_1, p_6\}, \{p_2, p_5\}, \{p_3, p_4\}$$

**Chúng dẫn đến các thừa số nghiệm bậc 2:**

$$(1 - p_0 z^{-1})(1 - p_7 z^{-1}) = (1 - \sqrt{0.5} z^{-1})(1 - \sqrt{0.5} z^{-1}) = 1 - 0.5 z^{-2}$$

$$(1 - p_1 z^{-1})(1 - p_6 z^{-1}) = 1 - \sqrt{2} \cos\left(\frac{2\pi}{8}\right) z^{-1} + 0.5 z^{-2} = 1 - z^{-1} + 0.5 z^{-2}$$

$$(1 - p_2 z^{-1})(1 - p_5 z^{-1}) = 1 - \sqrt{2} \cos\left(\frac{4\pi}{8}\right) z^{-1} + 0.5 z^{-2} = 1 + 0.5 z^{-2}$$

$$(1 - p_3 z^{-1})(1 - p_4 z^{-1}) = 1 - \sqrt{2} \cos\left(\frac{6\pi}{8}\right) z^{-1} + 0.5 z^{-2} = 1 + z^{-1} + 0.5 z^{-2}$$

**Cuối cùng, ta có**

$$H(z) = \left[ \frac{1 - 1.8478 z^{-1} + z^{-2}}{1 - 0.5 z^{-2}} \right] \left[ \frac{1 - 0.7654 z^{-1} + z^{-2}}{1 - z^{-1} + 0.5 z^{-2}} \right] \cdot \left[ \frac{1 + 0.7654 z^{-1} + z^{-2}}{1 + 0.5 z^{-2}} \right] \left[ \frac{1 + 1.8478 z^{-1} + z^{-2}}{1 + z^{-1} + 0.5 z^{-2}} \right]$$

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.4. Cascade sang canonical

Các ma trận hệ số A và B trong trường hợp này sẽ là:

$$A = \begin{bmatrix} 1 & 0 & -0.5 \\ 1 & -1 & 0.5 \\ 1 & 0 & 0.5 \\ 1 & 1 & 0.5 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & -1.8478 & 1 \\ 1 & -0.7654 & 1 \\ 1 & 0.7654 & 1 \\ 1 & 1.8478 & 1 \end{bmatrix}$$

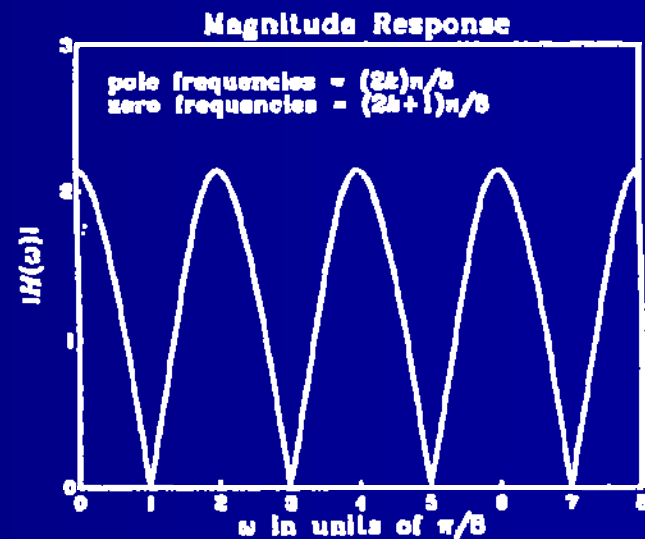
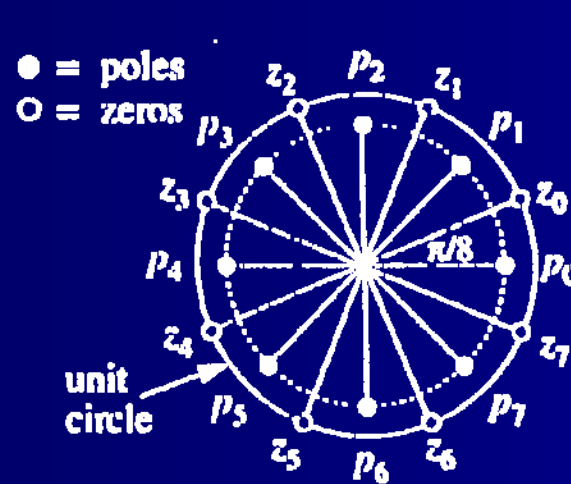
Mạch lọc này đóng vai trò như một mạch lọc notch /comb. Trong đó các dip zero được dịch  $\pi/8$  khi so sánh với các đỉnh cực. Các mẫu cực zero và đáp ứng biên độ  $|H(\omega)|$  được cho trong hình 7.4.1.

Các ví dụ này chỉ nhằm mục đích minh họa quá trình phân tích thành nhân tử. Cách thực hiện dạng canonical khá hiệu quả hơn cách cascade, vì nó chỉ liên quan đến một bộ nhân và một khâu trễ 8-fold

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.4. Cascade sang canonical

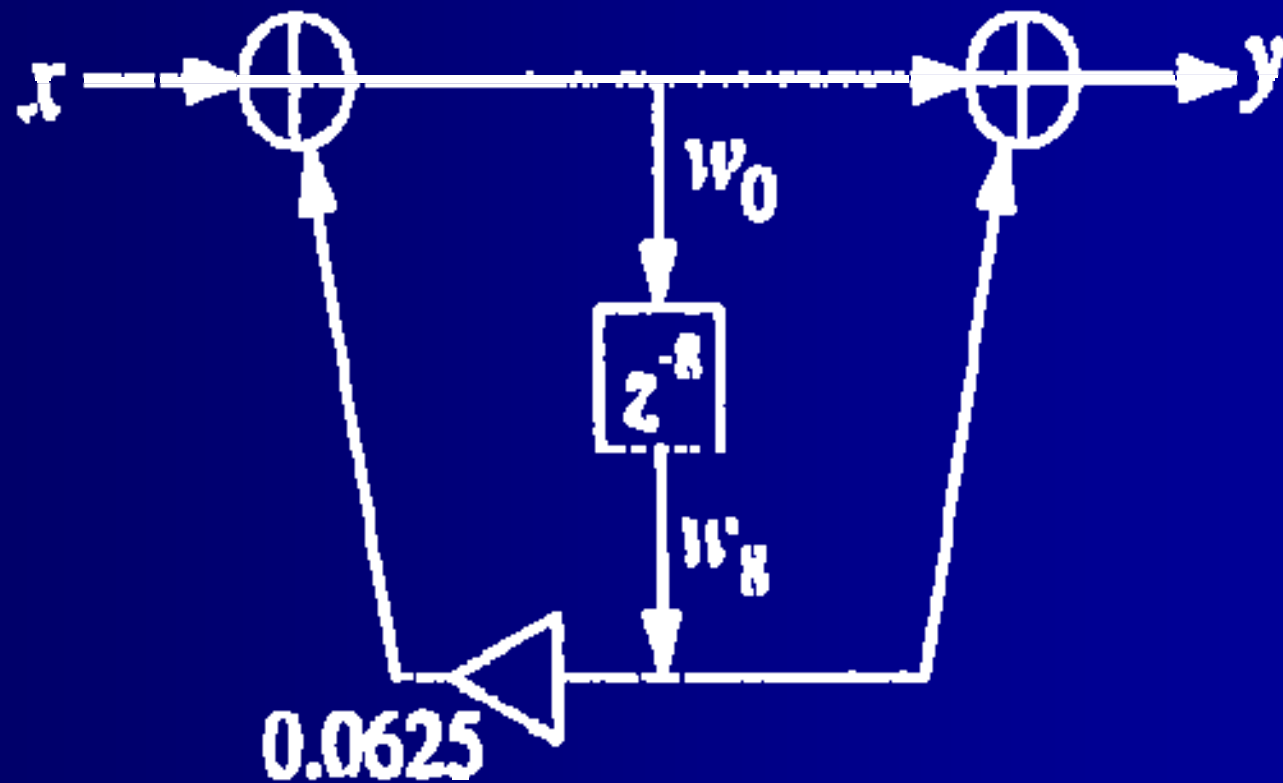
Các thực hiện canonical và thuật toán xử lý mẫu tương ứng được cho trong hình 7.4.2. ở đây,  $\omega = [\omega_0, \omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \omega_6, \omega_7, \omega_8]$  là vector trạng thái trong 9 chiều.



Hình 7.4.1 Mẫu cực/zero và đáp ứng biên độ của ví dụ 7.4.3

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.4. Cascade sang canonical





# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.4. Cascade sang canonical

*for each input sample  $x$  do :*

$$w_0 = x + 0.0625 w_8$$

$$y = w_0 + w_8$$

$$\text{delay}(8, w)$$

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

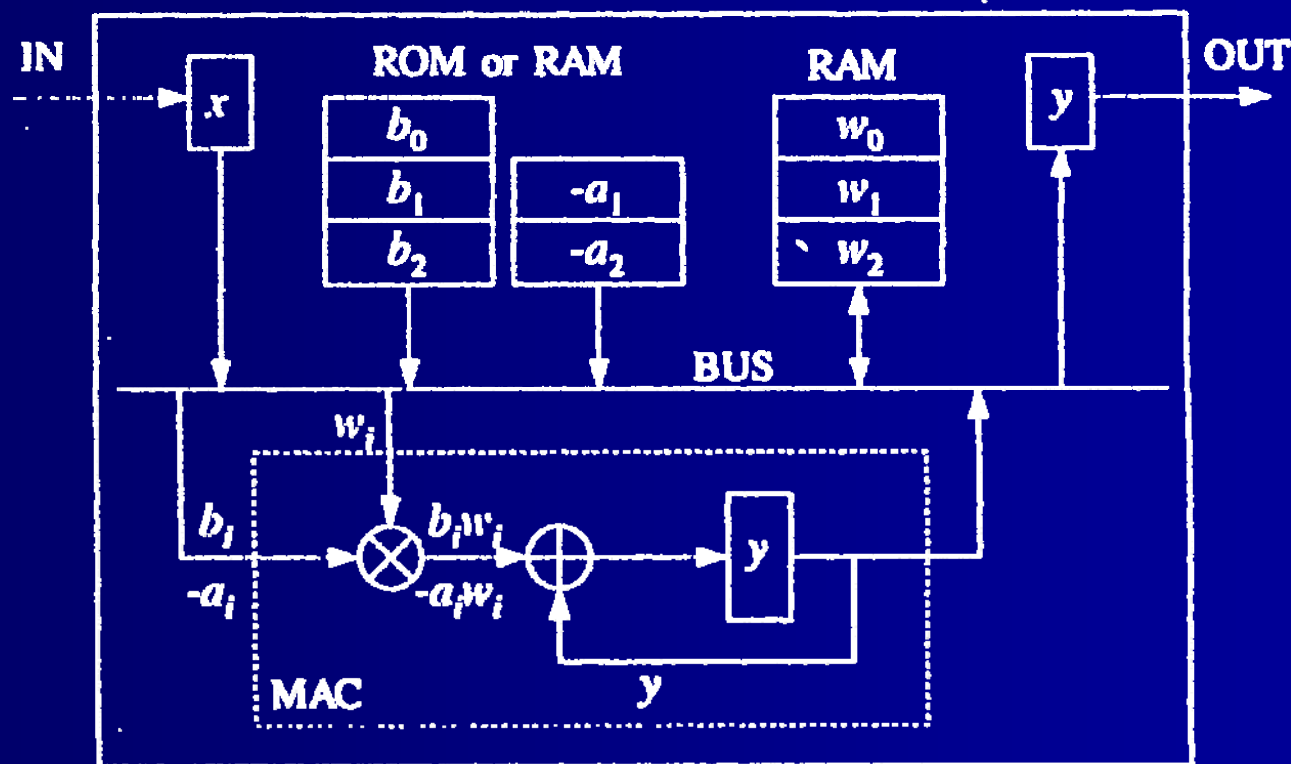
## 7.5. Cài đặt phần cứng và các bộ đệm vòng

Việc cài đặt phần cứng của các mạch lọc FIR với các chip DSP đã được trình bày trong phần 4.2.4. Các mạch lọc IIR cũng có thể được thiết kế một cách tương tự.

Xét SOS (7.1.1) được biểu diễn theo dạng canonical như hình 7.2.3. Một cách thiết kế phần cứng bằng một chip DSP điển hình được cho trong hình 7.5.1. Các hệ số mạch lọc được chứa trong RAM hay ROM trên chip. Các trạng thái trong được chứa trong RAM.

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.5. Cài đặt phần cứng và các bộ đếm vòng



Hình 7.5.1 Cách thực hiện một khâu bậc hai bằng chip DSP điển hình.

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.5. Cài đặt phần cứng và các bộ đệm vòng

Như trong phần 4.2.4, thuật toán xử lý mẫu (7.2.2) có thể được viết theo dạng mô phỏng các lệnh của một chip DSP, như là các lệnh MAC và các lệnh dịch dữ liệu:

*for each input sample x do :* (7.5.1)

$$w_0 := x - a_1 w_1$$

$$w_0 := w_0 - a_2 w_2$$

$$w_2 := w_1, \quad y := y + b_1 w_1$$

$$w_1 := w_0, \quad y := y + b_0 w_0$$

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.5. Cài đặt phần cứng và các bộ đệm vòng

Trong chip DSP hiện đại, mỗi dòng trong thuật toán trên có thể được cài đặt bằng một lệnh loại-MAC; do đó, mỗi một SOS có thể được cài đặt với 5 dòng lệnh cho mỗi mẫu vào.

Lưu ý: các trạng thái  $w_1$  và  $w_2$  không được cập nhật cho đến khi nào  $w_0$  đã được tính toán xong. Các lệnh MAC để tính  $w_0$  thực hiện theo chiều tới, có nghĩa là từ hệ số ai thấp nhất đến cao nhất. Điều này là thuận tiện bởi vì khi  $w_0$  vừa được tính xong, các lệnh dịch dữ liệu/MAC kết hợp để tính  $y$  có thể bắt đầu, nhưng sẽ thực hiện lùi từ hệ số bi cao nhất đến thấp nhất.

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.5. Cài đặt phần cứng và các bộ đếm vòng

Trong trường hợp tổng quát, ta có thể viết lại phương trình (7.2.5) với giả sử  $L = M$ :

*for each input sample  $x$  do :*

$w_0 := x$

*for  $i = 1, 2, \dots, M$  do :*

$w_0 := w_0 - a_1 w_1$

$y := b_M w_M$  (7.5.2)

*for  $i = M - 1, \dots, 1$  do :*

$w_{i+1} := w_i$

$y := y + b_i w_i$

$N_{instr} = (M + 1) + C$  (mạch lọc FIR bậc  $M$ ) (7.5.3)

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.5. Cài đặt phần cứng và các bộ đệm vòng

Thời gian tổng cần thiết cho một quá trình xử lý một mẫu đầu vào sẽ là  $T_{proc} = N_{instr} T_{instr}$ . (7.5.5)

Trong đó  $T_{instr}$  là thời gian cho một lệnh cơ bản, như MAC hay MACD. Cũng trong phần 4.2.4,  $T_{instr}$  khoảng  $30 \div 80$  ns, tương ứng với lệnh có tốc độ  $f_{instr} = 1/T_{instr} = 12.5 \div 33.3$  MIPS (*Million Instructions Per Second*). Thời gian xử lý một mẫu áp đặt giới hạn trên của tốc độ lấy mẫu mà mạch lọc có thể hoạt động được:

$$f_s = \frac{1}{T_{proc}} = \frac{1}{N_{instr} T_{instr}} = \frac{f_{instr}}{N_{instr}} \quad (7.5.6)$$

Trong đó  $1/T_{proc}$  là tốc độ tính toán của chip, tức là số mẫu có thể được xử lý trong 1s.

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.5. Cài đặt phần cứng và các bộ đệm vòng

Ta không thể tính toán số lệnh không phụ thuộc vào bộ xử lý cho một mạch lọc cụ thể. Số lệnh cũng như thời gian xử lý  $T_{\text{proc}}$  trong 1s tùy thuộc vào kiến trúc, tập lệnh, cách truy xuất bộ nhớ của chip DSP.

Đối với một cascade của SOS, để tìm thời gian xử lý tổng ta phải tính toán thời gian phải chi phí để xử lý một SOS rồi nhân cho số lượng khâu. Theo phương trình (7.5.1) ta thấy rằng cần khoảng 5 lệnh cho một SOS; do đó, thời gian xử lý cho một SOS có thể xấp xỉ bằng (bỏ qua overhead).



# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.5. Cài đặt phần cứng và các bộ đệm vòng

$$T_{\text{SOS}} \approx 5T_{\text{instr}} \quad (7.5.7)$$

Khi có K SOS được nối cascade hay song song, nhưng đều được thiết kế với cùng một DSP, thì tổng số lệnh sẽ là:

$$N_{\text{instr}} = 5K + C \quad (\text{Mạch lọc IIR khâu } K) \quad (7.5.7)$$

Trong đó là overhead cho mạch lọc K-tầng. Do đó thời gian xử lý tổng là:

$$T_{\text{proc}} = T_{\text{instr}}(5K + C) = KT_{\text{SOS}} + Ct_{\text{instr}} \quad (7.59)$$

Bỏ qua hệ số overhead tương đối nhỏ, ta tìm tốc độ lấy mẫu tối đa  $f_s$  cho K khâu bậc hai.

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.5. Cài đặt phần cứng và các bộ đệm vòng

$$f_s = \frac{1}{T_{proc}} = \frac{1}{KT_{SOS}} = \frac{f_{instr}}{5K} \quad (7.5.10)$$

Đối với các mạch song song (xem bài tập 5.18), ta có thể nâng tốc độ thông lượng bằng cách dùng K chip DSP khác nhau hoạt động song song, mỗi chip được dùng để thực hiện phép lọc SOS trong  $T_{SOS}$  giây. Trong trường hợp này, thời gian xử lý tổng là  $T_{SOS}$  vì tất cả các DSP kết thúc đồng thời, và do đó, tốc độ thông lượng sẽ nhanh hơn K lần trong trường hợp 1 DSP:

$$T_{proc} = T_{SOS} \Rightarrow f_s = \frac{1}{T_{proc}} = \frac{1}{T_{SOS}} \quad (7.5.11)$$

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

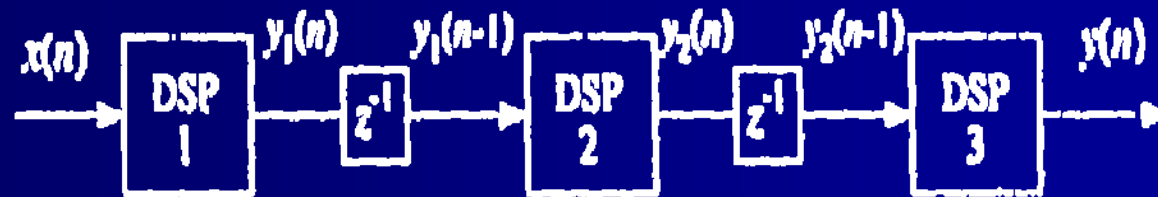
## 7.5. Cài đặt phần cứng và các bộ đệm vòng

Đối với các mạch cascade, ta cũng có thể sử dụng K chip DSP – mỗi chip cho một SOS – để tăng tốc độ xử lý. Tuy nhiên, vì đầu ra của khâu này là đầu vào của khâu kế tiếp nên không thể chạy K chip DSP đồng thời. Mỗi DSP phải chờ  $T_{SOS}$  giây cho DSP trước nó kết thúc.

Một giải pháp là pipeline các phép lọc của các khâu liên tiếp, do đó tất cả các DSP hoạt động với nhau, nhưng mỗi DSP xử lý đầu vào từ thời điểm lấy mẫu trước đó. Việc này có thể được thực hiện bằng cách chèn các khâu trễ đơn vị giữa các DSP, như trong hình 7.5.2.

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.5. Cài đặt phần cứng và các bộ đệm vòng



Hình 7.5.2 Pipeline hoạt động của nhiều bộ xử lý DSP. Tại thời điểm thứ  $n$ , khi DSP-1 đang xử lý mẫu đầu vào hiện hành  $x(n)$ , DSP-2 đang xử lý mẫu  $y_1(n-1)$  được DSP xuất vào thời điểm trước đó và được lưu trong một thanh ghi chờ, và DSP-3 đang xử lý mẫu  $y_2(n-1)$  được DSP-2 tạo ra trước đó,... Tác động của việc phát sinh các thời gian trễ này chỉ là một thời gian trễ chung trong đầu ra. Ví dụ, trong trường hợp được chỉ ra trong hình 7.5.2, hàm truyền tổng hợp thay đổi từ  $H(z) = H_1(z) H_2(z) H_3(z)$  thành:

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.5. Cài đặt phần cứng và các bộ đệm vòng

$$H(z) = H_1(z)z^{-1}H_2(z)z^{-1}H_3(z) = z^{-2}H_1(z)H_2(z)H_3(z)$$

tương ứng với việc làm trễ đầu ra 2 xung đơn vị. Đối với K khâu, thời gian trễ tổng cộng là  $z^{-(K-1)}$ .

Ví dụ 7.5.1: Chip DSP dấu chấm động AT&T DSP 32 các [87,88] có thể thực thi các lệnh loại -MAC cơ bản trong 4 chu kỳ xung nhịp, tức là  $T_{instr} = 4T_{clock}$ . Do đó, tần số lệnh của nó là  $f_{instr} = f_{clock}/4$ . Một lệnh MAC điển hình biểu diễn 2 phép toán dấu chấm động: một phép nhân và một phép cộng. Do đó, chip có tốc độ tính toán là  $f_{Flop} = 2f_{instr} = f_{clock}/2$ .

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.5. Cài đặt phần cứng và các bộ đệm vòng

Ở tốc độ xung nhịp  $f_{\text{Clock}} = 50 \text{ MHz}$ , nó có tốc độ lệnh là  $f_{\text{instr}} = f_{\text{clock}}/4 = 12.5 \text{ MHz}$  và tốc độ tính toán là  $f_{\text{Flop}} = f_{\text{clock}}/2 = 25 \text{ MHz}$ . Thời gian cho một lệnh là  $T_{\text{instr}} = 1/f_{\text{instr}} = 80 \text{ ns}$ .

Một mạch lọc FIR bậc  $M$  có thể được cài đặt với

$$N_{\text{instr}} = (M + 1) + 11 = M + 12 \text{ (lệnh/mẫu)}$$

Do đó, thời gian xử lý 1 mẫu là  $T_{\text{proc}} = (M + 12)T_{\text{instr}}$ .

Đối với một mạch lọc FIR 100-tap ( $M=99$ ) dùng DSP 32 các đang chạy ở tốc độ 50 MHz, ta có  $T_{\text{proc}} = (M + 12)T_{\text{instr}} = (99 + 1)80 = 8.9 \mu\text{s}$ , có tốc độ thông lượng tối đa là  $f_s = 1/T_{\text{proc}} = 112.4 \text{ kHz}$ .

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.5. Cài đặt phần cứng và các bộ đệm vòng

Một mạch lọc IIR K-khâu được cài đặt

$$N_{\text{instr}} = 5K + 10 \text{ (Lệnh/mẫu)}$$

Nó đòi hỏi số trạng thái chờ chu kỳ – máy là:

$$N_{\text{wait}} = 2K + 1 \text{ (Trạng thái chờ/mẫu)}$$

Do đó, thời gian xử lý tổng cho K khâu sẽ là:

$$T_{\text{proc}} = N_{\text{instr}} T_{\text{instr}} + N_{\text{wait}} T_{\text{clock}}$$

Viết  $T_{\text{instr}} = 4T_{\text{clock}} = 4/f_{\text{clock}}$  ta có

$$T_{\text{proc}} = (4N_{\text{instr}} + N_{\text{wait}})/f_{\text{clock}} = (22K + 41)/f_{\text{clock}}$$

Đối với một SOS,  $K=1$ , và xung nhịp 50 MHz, ta tìm được  $T_{\text{proc}} = 1.26 \mu\text{s}$ , hay tần số lấy mẫu cực đại là:

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.5. Cài đặt phần cứng và các bộ đệm vòng

$$f_s = 1/T_{\text{proc}} = 793.6 \text{ kHz}$$

Đối với một mạch lọc 5 khâu,  $K = 1$ , ta tìm được  $T_{\text{proc}} = 3.02 \mu\text{s}$ , hay tần số lấy mẫu cực đại là  $f_s = 33.1 \text{ kHz}$ . Và cho một mạch lọc 10 khâu,  $K=10$ , ta có  $T_{\text{proc}} = 5.22 \mu\text{s}$ , hay tần số lấy mẫu cực đại là  $f_s = 191.6 \text{ kHz}$ .



# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.6. Các ảnh hưởng của quá trình lượng tử hóa trong mạch lọc số

Có hai loại ảnh hưởng do lượng tử hóa trong các mạch lọc số bên cạnh việc lượng tử hóa các tín hiệu đầu vào và đầu ra: các sai số roundoff trong các phép tính trong của mạch lọc và lượng tử hóa hệ số.

Lượng tử hóa hệ số thực hiện khi các hệ số mạch lọc được làm tròn từ giá trị chính xác của nó đến một số hữu hạn các chữ số (hay, các bit cho các cài đặt phần cứng). Cách biểu diễn trực tiếp và canonical có xu hướng rất nhạy với các việc làm tròn này, trong khi đó cách biểu diễn cascade thì rất tốt.

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.6. Các ảnh hưởng của quá trình lượng tử hóa trong mạch lọc số

Các mạch lọc bậc cao có các cực phân bố hẹp trong mặt phẳng  $z$ , những thay đổi nhỏ trong các hệ số mẫu thức có thể gây ra dịch lớn vị trí của các cực. Nếu bất kỳ điểm cực nào di chuyển ra ngoài đường tròn đơn vị, mạch lọc sẽ trở nên không ổn định, làm cho nó không thể sử dụng được. Nhưng cho dù các điểm cực không di chuyển ra ngoài, các khoảng dịch lớn có thể làm méo dạng đáp ứng tần số của mạch lọc và do đó nó không còn thỏa mãn các thông số thiết kế.

Trong thực tế, ta phải luôn kiểm tra rằng tính ổn định và các thông số của mạch lọc được bảo đảm khi các hệ số được làm tròn.

# CHƯƠNG 7: THỰC HIỆN MẠCH LỘC SỐ

## 7.6. Các ảnh hưởng của quá trình lượng tử hóa trong mạch lọc số

Khi dùng các package phần mềm để thiết kế một mạch lọc, ta phải luôn giữ các thông số thiết kế với số chữ số bảo đảm cho các yêu cầu. Bài tập 7.20 và 7.21 khảo sát các ảnh hưởng lượng tử hóa trên và một số nguy cơ thường gặp.

Điều này cũng không có nghĩa là các dạng trực tiếp và canonical không được sử dụng; thực tế, ta thấy trong ví dụ 7.4.3 và 7.4.4, các dạng canonical đơn giản hơn nhiều khi cài đặt so với các dạng cascade, và cũng rất tốt trong lượng tử hóa hệ số.

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.6. Các ảnh hưởng của quá trình lượng tử hóa trong mạch lọc số

Tóm lại, dạng cascade được khuyến cáo sử dụng trong các kỹ thuật thiết kế các mạch lọc IIR thông thấp, thông dải, thông cao, băng hẹp bậc cao, như là phương pháp biến đổi song tuyến tính, cho các kết quả thiết kế ở dạng cascade.

Có các dạng thực hiện khác, như là các khâu bậc 2 cascade trong dạng chuyển vị, các dạng song song, và các dạng dàn, hoạt động rất tốt trong quá trình lượng tử hóa hệ số.

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.6. Các ảnh hưởng của quá trình lượng tử hóa trong mạch lọc số

Sai số roundoff xảy ra trong các phép nhân và các phép cộng trong, ví dụ  $y := y + aw$ . Tích  $aw$  đòi hỏi hai lần số bit của mỗi thừa số để có thể biểu diễn một cách chính xác. Sai số roundoff sẽ xảy ra nếu tích này được làm tròn đến kích thước ban đầu của hai thừa số. Các sai số roundoff như thế có thể bị đưa vào các vòng hồi tiếp của các mạch lọc recursive và có thể bị khuếch đại, gây méo dạng rất lớn trong đáp ứng mong muốn.

- Các thực hiện trạng thái đặc biệt và kỹ thuật của định hình nhiễu lượng tử hóa (còn được gọi là định hình phổ sai số – *error spectrum shaping*) có thể được sử dụng để tối thiểu hóa tích lũy sai số roundoff.

# CHƯƠNG 7: THỰC HIỆN MẠCH LỌC SỐ

## 7.6. Các ảnh hưởng của quá trình lượng tử hóa trong mạch lọc số

Để không tràn các kết quả trung gian trong các phép tính của mạch lọc, các hệ số định tỷ lệ thích hợp có thể được đưa ra tại các điểm khác nhau trong các tầng lọc. Các chip DSP hiện đại giải quyết vấn đề ảnh hưởng do lượng tử hóa theo hai hướng: dùng các từ dài, như là 32 bit, để lưu trữ các hệ số, và dùng các bộ cộng dồn độ chính xác-kép để thực hiện nhiều phép toán MAC mà không bị sai số roundoff trước khi kết quả cuối cùng được làm tròn.