Digital Filter Design & Implementation

Wonyong Sung

2 x 1.5 hrs

School of Electrical Engineering
Seoul National University

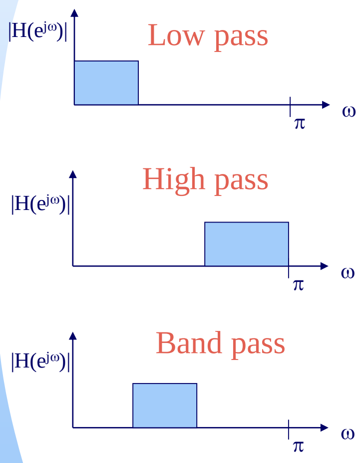
Digital and Analog Filters

- Analog filters are usually implemented in L(inductors) and C (capacitors). For high precision, crystal and ceramic, saw filters are used.
- Low frequency analog filters are bulky because of large inductance and capacitance needed. Analog filters have implementation advantages in high frequency.
- Digital filters are implemented in arithmetic operations, thus is very precise (5% accuracy for C, but 0.01% for digital)
- The number of arithmetic operations is proportional to the sampling frequency.

Digital Filters

- **Types of Digital Filters:**
  - low-pass, band-pass, high-pass, notch-filter, allpass, etc.
- **FIR and IIR Digital Filters**
- **Multiplierless filters**
- **Filters for sampling rate conversion**
- **Structures of Digital Filters**
  - Direct, cascade, parallel forms
  - State-space realizations
  - Orthogonal digital filter
- **Quantization Errors, Stability, accuracy**

Types of Digital Filters

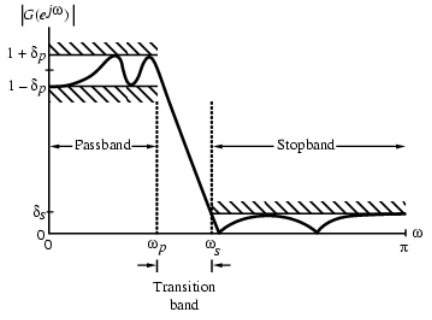


- **Usages:**
  - Low pass: anti-aliasing, smoothing, noise reduction
  - High pass: DC removal, baseline wander reduction
  - Band pass: noise reduction
- **Design:**
  - Choose FIR or IIR filter coefficients to approximate desired frequency response.
  - Usually the designed filter coefficients are not unique! Leaving large design space to be explored. Passband, stopband ripples.

Wonyong Sung
Multimedia Systems Lab SNU

Digital filter specifications

- For example the magnitude response $|G(e^{j\omega})|$ of a digital lowpass filter may be given as indicated below



* Transition bandwidth is important for filter order determination.

Digital filter specifications

- In practice, passband edge frequency F_p and stopband edge frequency F_s are specified in Hz
- For digital filter design, normalized bandedge frequencies need to be computed from specifications in Hz using

$$\omega_p = \frac{\Omega_p}{F_T} = \frac{2\pi F_p}{F_T} = 2\pi F_p T$$

$$\omega_s = \frac{\Omega_s}{F_T} = \frac{2\pi F_s}{F_T} = 2\pi F_s T$$

F_T is the sampling frequency

Digital filter specifications

Filter specification parameters

- ω_p - passband edge frequency
- ω_s - stopband edge frequency
- δ_p - peak ripple value **in the passband**
- δ_s - peak ripple value **in the stopband**

Digital filter specifications

- **Practical specifications are often given in terms of** loss function (in dB)
- $$\mathcal{G}(\omega) = -20\log_{10}\left|G(e^{j\omega})\right|$$
- Peak passband ripple

$$\alpha_p = -20\log_{10}(1-\delta_p) \quad \textbf{dB}$$

- Minimum stopband attenuation

$$\alpha_s = -20\log_{10}(\delta_s) \quad \textbf{dB}$$

Filter 복잡도

- **Wp 와 Ws의 차이를 transition bandwidth라 하며 이 사이가 좁을 수록 filter 의 차수가 높아야 한다 (복잡하다). 보통 0.05π 정도는 준다.**
- **Passband ripple은 0dB (이상적, 불가능한 값)이상으로 0.5 dB, 1 dB 등으로 준다. 이 값이 클 수록 필터의 ripple 은 크지만 차수가 낮아진다.**
- **Stopband ripple 또는 rejection 은 stopband 에서의 감쇄된 값으로 -40dB (amplitude로 1/100)에서 -80dB (1/10000)의 값을 가진다. 이 감쇄가 클 수록 차수가 높은 어려운 필터가 된다.**

Digital filter의 구성요소

- **Multiplier**
- **Adder**
- **Delay: z^-1 로 표현하며, 한 샘플 지연을 의미한다.**
- **이 때 delay의 개수를 가지고 몇차의 filter 라고 말한다.**

- **Analog filter: R, C, L, amplifier**
  - 이 때 C, L의 개수를 가지고 필터의 차수를 말한다.

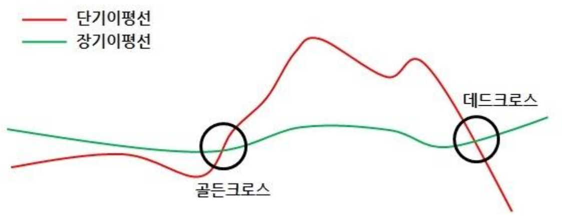
간단한 digital filtering의 예

- 지금부터 과거의 신호를 10개 합쳐서 결과로 낸다.
- 즉 $y[n] = x[n] + x[n-1] + \ldots + x[n-9]$

- 이렇게 지금부터 과거의 신호를 합쳐서 내는 것을 moving average라 한다. 이 때 더하는 것의 개수를 늘릴 수록 더 강력한 lowpass filter 가 된다. 즉 높은 주파수 성분이 없어진다.
- 주식값: 이동평균
  - 5일, 20일, 60일, 120일 이동평균
- Moving average filter: 매우 간단한 lowpass filter이고 구현하는데 계산이 매우 간단하다 (곱셈 없다).

Wonyong Sung
Multimedia Systems Lab SNU

Golden cross, dead cross



간단한 filter

- **y[n] = x[n] – x[n-1]**
- **이 필터는 어떤 특징을 가지는가?**
- **DC를 집어 넣으면, 즉 x[n] = x[n-1] 이 경우는 y[n]으로 0 이 나온다.**
- **X[n] = cos(πn)을 넣는다. (매우 높은 주파수 신호)**
- **y[n] = cos(πn)- cos(π(n-1))**
  = cos(πn)- cos(πn-π) = 2 cos(πn)

High pass filter 가 된다.

Wonyong Sung
Multimedia Systems Lab SNU

일반적인 구조의 FIR filter

- **FIR filter (Finite Impulse Response) filter: input signal 을 delay한 후 각각을 계수로 multiply 한 후 모두 더한다.**
- **The basic FIR Filter equation is**

$$y[n] = \sum h[k].x[n-k] \quad \text{for } k = 0 \text{ to } M\text{-}1$$

where h[k] is an array of constants (filter coefficients)

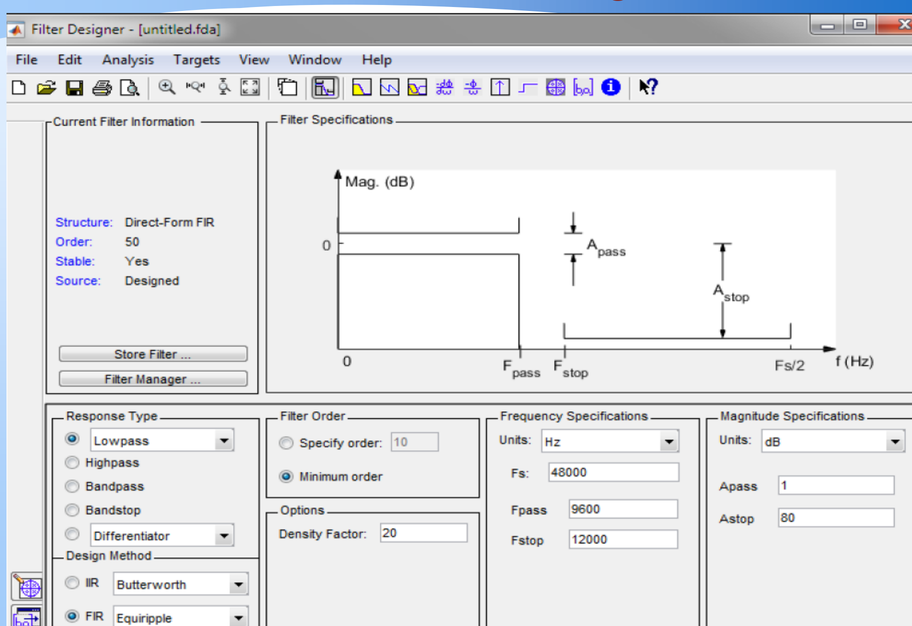
M=3 인 경우

y[0] = h0*x[0]+h1*x[-1] +h2*x[-2]
y[1] = h0*x[1]+h1*x[0] +h2*x[-1]
y[2] = h0*x[2]+h1*x[1] +h2*x[0]
y[3] = h0*x[3]+h1*x[2] +h2*x[1]

```
//N 개의 출력 계산 y[0] ~ y[N-1]
For (n=0; n<N; n++) {
    y[n]=0;
    For (k = 0;k<M;k++)
    //inner loop
    y[n] = y[n] + h[k]*x[n-k];}
```

Wonyong Sung
Multimedia Systems Lab SNU

Matlab 의 filter designer



Multimedia Systems Lab SNU

Filter design specification

- Filter structure: FIR filter direct form (무난)
- Filter의 종류: lowpass, high pass, bandpass 등
- Design method: Equiripple (CAD 방법)
- Sampling frequency: 현재 사용하는 system의 sampling frequency를 주어도 되고, Fs =1 로 두고 설계해도 됨. Fs =1에서 Fpass = 0.1인 것과 Fs = 10KHz에서 Fpass = 1KHz인 것은 동일한 digital filter
- Passband ripple (여기에서는 1dB), stopband attenuation (여기에서는 80dB)를 준다.
- 그러면 금방 filter order M 을 정하고 계수를 준다.

FIR Digital Filter Order Estimation

Kaiser's Formula:

$$N \cong \frac{-20\log_{10}(\sqrt{\delta_p \delta_s})}{14.6(\omega_s - \omega_p)/2\pi}$$

- ie ***N*** is inversely proportional to transition band width and not on transition band location
- 설계된 **filter** 의 **order** 가 너무 크다 (즉 계산 많이 든다). 그래서 줄여야 한다. 그러면 **transition bandwidth**를 늘리고, **attenuation** 을 작게 해야 한다 **(80dB=> 60dB =>40dB)**

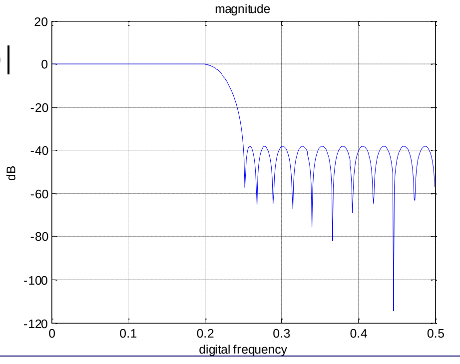
Example: Low Pass Filter

Passband $f = 0.2$ (fs = 1 기준)

Stopband $f = 0.25$ with attenuation 40dB

Choose order N=40/(22*(0.25-0.20))=37

$|H(f)|$



Almost 40dB!!!

f

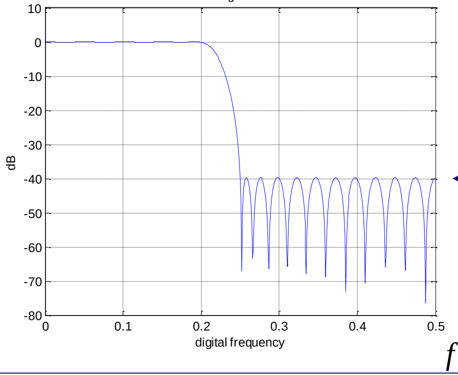
Example: Low Pass Filter

Passband $f = 0.2$

Stopband $f = 0.25$ with attenuation 40dB

Choose order N=40 > 37

$|H(f)|$



OK!!!

f

Linear phase filter - symmetric FIR

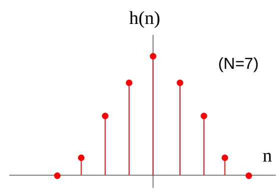
- **h(n) = h(-n)**
- **Evaluate the frequency response (assuming that N is odd) and h(n) is real-valued**

$$H(z) = \sum_{n=n_1}^{n_2} h(n)\ z^{-n} = \sum_{n=-\frac{N-1}{2}}^{\frac{N-1}{2}} h(n)\ z^{-n}$$

if $h(n) = h(-n)$ we get

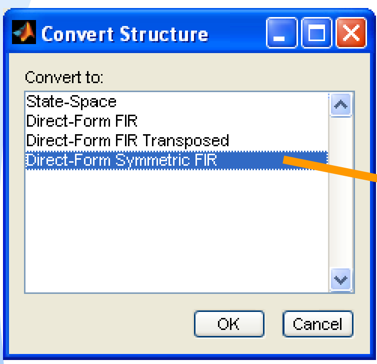
$$H(e^{j2\pi\Omega}) = h(0) + \sum_{n=1}^{\frac{N-1}{2}} h(n)\ (e^{-j2\pi n\Omega} - e^{+j2\pi n\Omega})$$

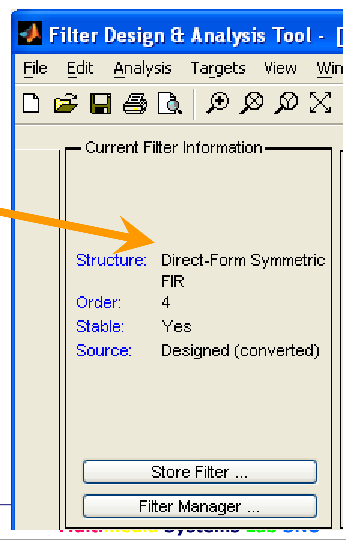
$$H(e^{j2\pi\Omega}) = h(0) + 2\sum_{n=1}^{\frac{N-1}{2}} h(n)\ \cos[2\pi n\Omega]$$



The **frequency response is real**: phase shift is 0 or 180 degrees

Edit, Convert Structure …

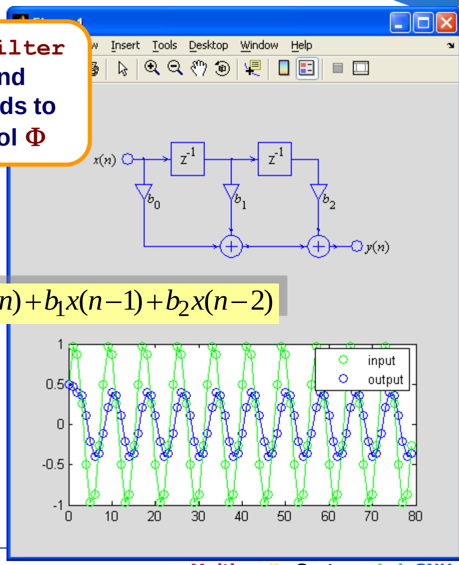




MATLAB example 1

```
N = 80; k = 0:(N-1);
b0 = 1;
b1 = -1;
b2 = 1;
B = [b0 b1 b2];
f = 1/8;
x = sin(2*pi*f*k+pi/6);
y = filter(B,1,x);
subplot(2,1,1)
systemFIR(0,0,4,5,10,'b')
subplot(2,1,2)
plot(k,x,'go', k,y,'bo',...
    k,x,'g-', k,y,'b-')
legend('input','output')
```

MATLAB `filter` command corresponds to the symbol Φ

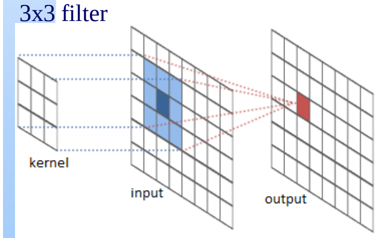


$$y(n) = b_0 x(n) + b_1 x(n-1) + b_2 x(n-2)$$

2D FIR filter (2D convolution)

- 2 dimensional filtering: image 신호처리에 사용함.
- Input x[n1, n2] -> h[n1, n2] -> y[n1, n2]

3x3 filter



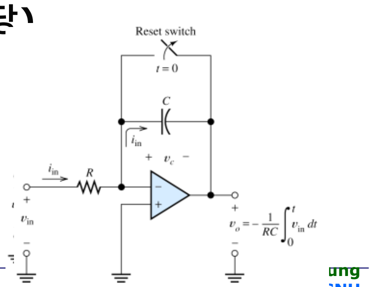

```
//N² 개의 출력 계산 y[0,0] ~ y[N-1, N-1]
For (n1=0; n1<N; n1++) {
  For (n2=0; n2<N; n2++) {
    y[n1, n2]=0.0;
    For (k1 = 0; k1<M; k1++) {
      For (k2 = 0;k2<M;k2++)
      //inner loop
      y[n1, n2] = y[n1, n2] + h[k1, k2]*x[n1-k1, n2-k2];}}}
```

Recursive filter

- ❖ FIR filte는 입력 x[n]을 지연시킨 것을 조합하여 y[n]을 만들었다.
- ❖ Recursive filter는 x[n]을 지연시킨 것과 함께, y[n]을 지연시킨 것을 조합하여 y[n]을 만든다. IIR (infinite impulse response) filter 라고도 한다.

IIR filter 의 예: y[0] =0.0으로 초기화,

- ❖ y[n] = y[n-1] + x[n], for n=1, 2, 3, …..
  적분기 (accumulator 에 해당)

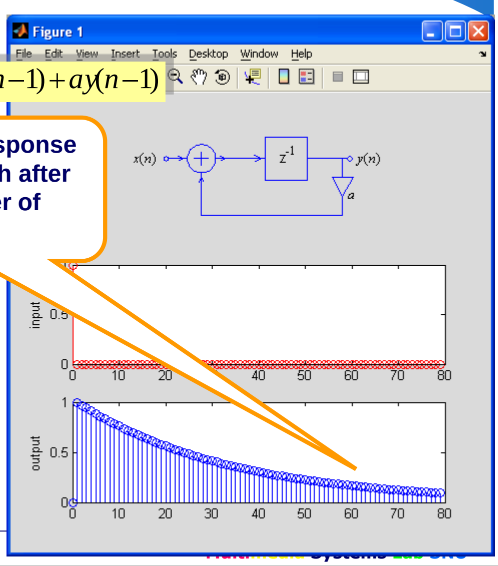


MATLAB example 3

```
N = 80; k = 0:(N-1);
a = 0.97;
B = [0 1];
A = [1 -a];
x = (k==0);
y = filter(B,A,x);
subplot(3,1,1)
draw1stIIR(0,0,4,5,10,'b')
subplot(3,1,2)
stem(k,x,'r')
ylabel('input')
subplot(3,1,3)
stem(k,y,'b')
ylabel('output')
```

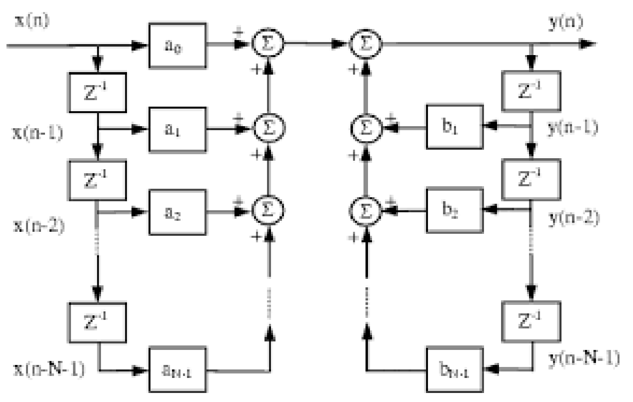
$$y(n) = x(n-1) + ay(n-1)$$

The impulse response does not vanish after finite number of samples



Recursive filter (Nth order)

- feed-forward feed-back path



z^{-1} means 1 sample delay

FIR, IIR digital filters

- **Let {h[n}: impulse response**
  {x(n)}: input, {y(n)}: output
- **Finite impulse response (FIR) filter:**

$$y(n) = \sum_{j=0}^{J-1} h(j)x(n-j)$$

- **Usually, implemented as a feed-forward type.**
- **Infinite impulse response (IIR) filter**

$$y(n) = \sum_{i=1}^{P} a(i)y(n-i) + \sum_{k=0}^{Q} b(k)x(n-k)$$

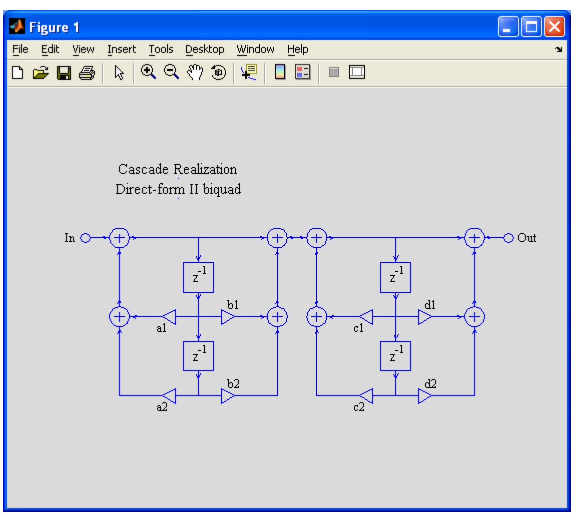
- **Recursive formula will impact on computation methods (feedback).**
- 일반적으로 같은 주파수 특성을 가지는 **filter** 를 만드는계 계산량이 덜 든다.
- **Stability concerns:**
  - The magnitude of y(n) may become infinity even all x(n) are finite!
  - Coefficient values,
  - Quantization error

FIR or IIR filter

- 일반적으로 비슷한 주파수 특성을 가지는 filter 를 만드는데 IIR filter의 차수가 적게 필요하다. IIR filter의 차수는 보통 10차 이내인데 비해서 FIR filter 의 경우는 100차가 넘는 경우도 많다.
- 대략적인 계산량은
- FIR filter: output sample의 수*filter order
- IIR filter: output sample 의 수 *(feedback 부분 order (P)+ feedforward 부분 order (Q))
- Image processing 에는 꼭 Linear phase FIR filter 만 사용한다.

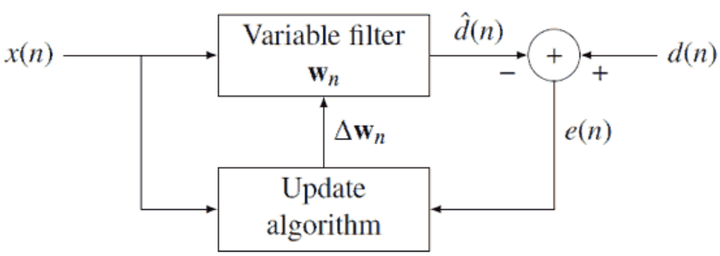
$$y(n) = \sum_{i=1}^{P} a(i)y(n-i) + \sum_{k=0}^{Q} b(k)x(n-k)$$

Cascade direct form II



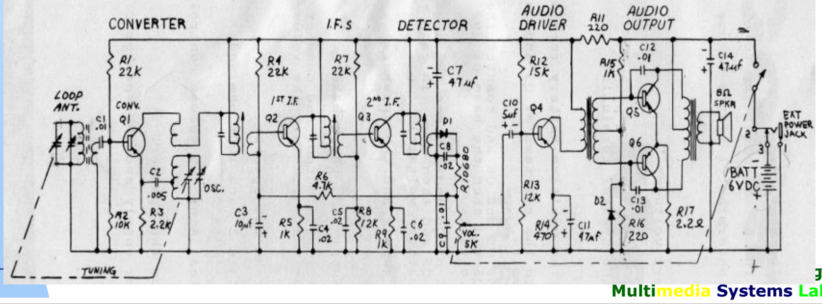
Adaptive filter

- ❖ **Filter 의 계수를 신호에 따라서 바꾼다. 즉, 환경 조건에 따라서 필터의 계수를 자동으로 바꾼다.**
- ❖ **나중에 비슷한 알고리즘을 neural network 에서 다룬다.**



SW defined Radio

RF analog -> 고속 ADC ->
(SW: bandpass filtering -> AM demodulation->
lowpass filter) ->
audio DAC -> audio amp & speaker



Wonyong Sung
Multimedia Systems Lab SNU

Summary

- Digital signal processing 은 digital 연산(승, 가산)과 memory (지연)를 이용하여 신호처리를 수행한다.
- SW를 이용하여 구현할 경우 매우 쉽게 시스템의 parameter 등을 바꿀 수 있으므로 유연한 시스템 설계가 가능하다.
- Sampling frequency 가 높은 고속 신호처리에는 계산량이 매우 많이 필요하였으나, 현재 고속의 컴퓨터를 이용하기 때문에 SW를 이용한 시스템 구현이 가능하다.
- Digital filter 는 FIR filter 와 IIR filter 등이 있다. 계수를 환경 조건에 따라 바꿀 수 있는 adaptive filter 도 있다.