Project #1

1. Filter

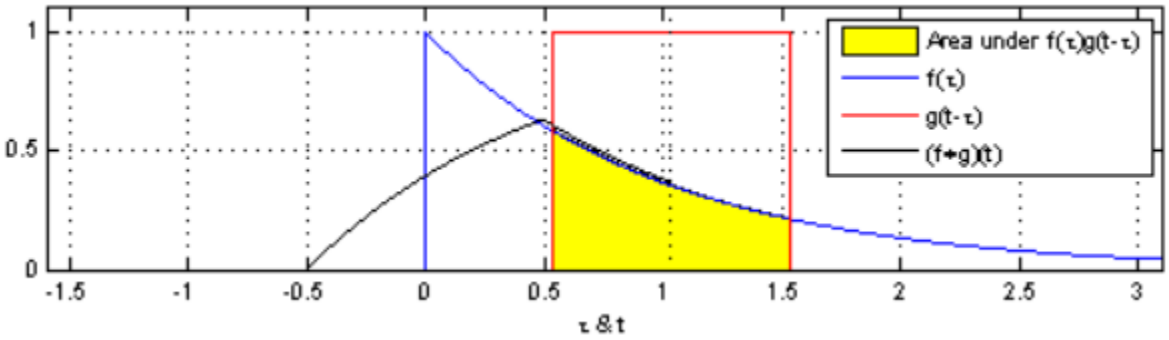
- 특정 주파수(Frequency)를 차단, 통과시키는 목적의 회로
  원하지 않는 신호(소리)인 잡음(노이즈)을 제거하기 위함

- 저역통과필터 (Low-pass Filter, LPF)
  : 낮은 대역의 주파수(저주파)는 통과시키고,
  높은 대역의 주파수(고주파)는 차단하는 필터
  A/D Converter의 샘플링(Sampling) 과정에 필수적으로 현존하는
  거의 모든 디지털 제품에 들어있음

- 응용 분야
  : 전자 회로, 전기 회로 및 음성과 영상 같은 신호 처리 시스템,
  통신 시스템

* Convolution

$$f(t) * g(t) \triangleq \underbrace{\int_{-\infty}^{\infty} f(\tau) g(t-\tau)\, d\tau}_{(f*g)(t)},$$



- Discrete-Time Convolution

$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]\ \ h[n-k]$$

Ex) x[k] = [3 1 2] h[k] = [3 2 1]

k:	-2	-1	0	1	2	3	4	5
x[k]:			3	1	2			
h[-k]:	1	2	3					
h[1-k]:		1	2	3				
h[2-k]:			1	2	3			
h[3-k]:				1	2	3		
h[4-k]:					1	2	3	
h[5-k]:						1	2	3

y[0] = 3 x 3 = 9

y[1] = 3 x 2 + 3 x 1 = 9

y[2] = 3 x 1 + 1 x 2 + 2 x 3= 11

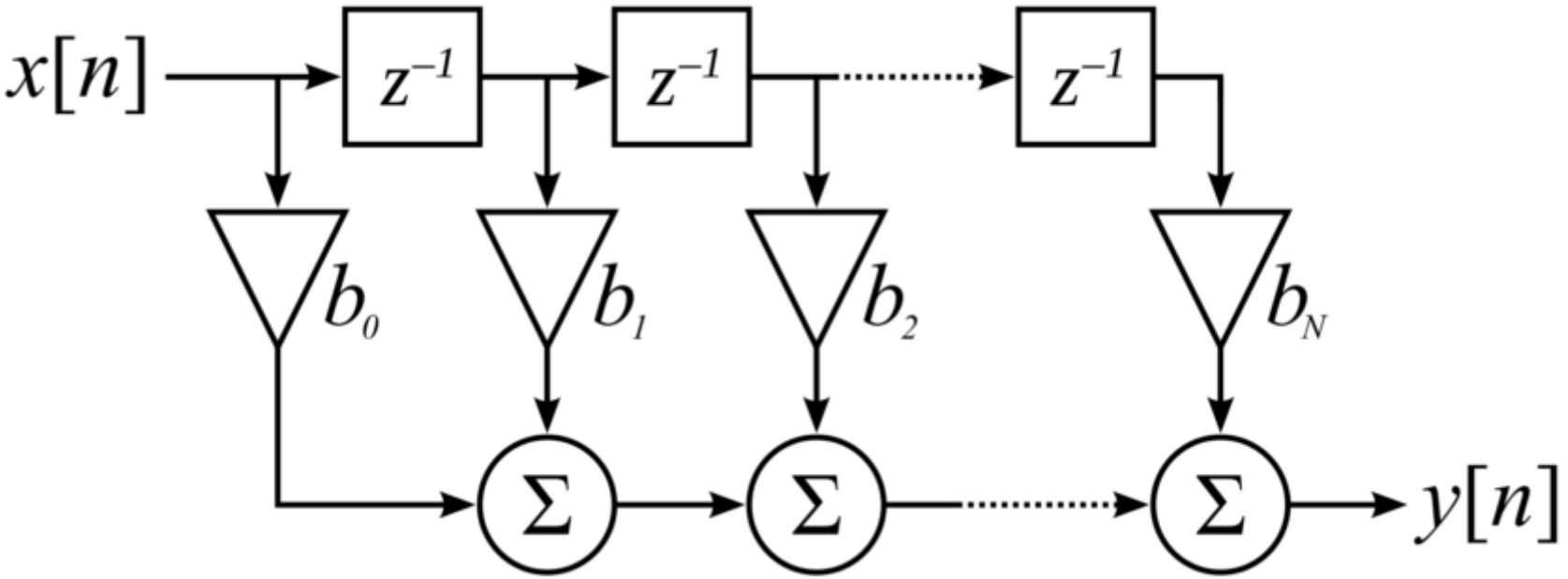
y[3] = 1 x 1 + 2 x 2 = 5

y[4] = 2 x 1 = 2

y[5] = 0 (no overlap)

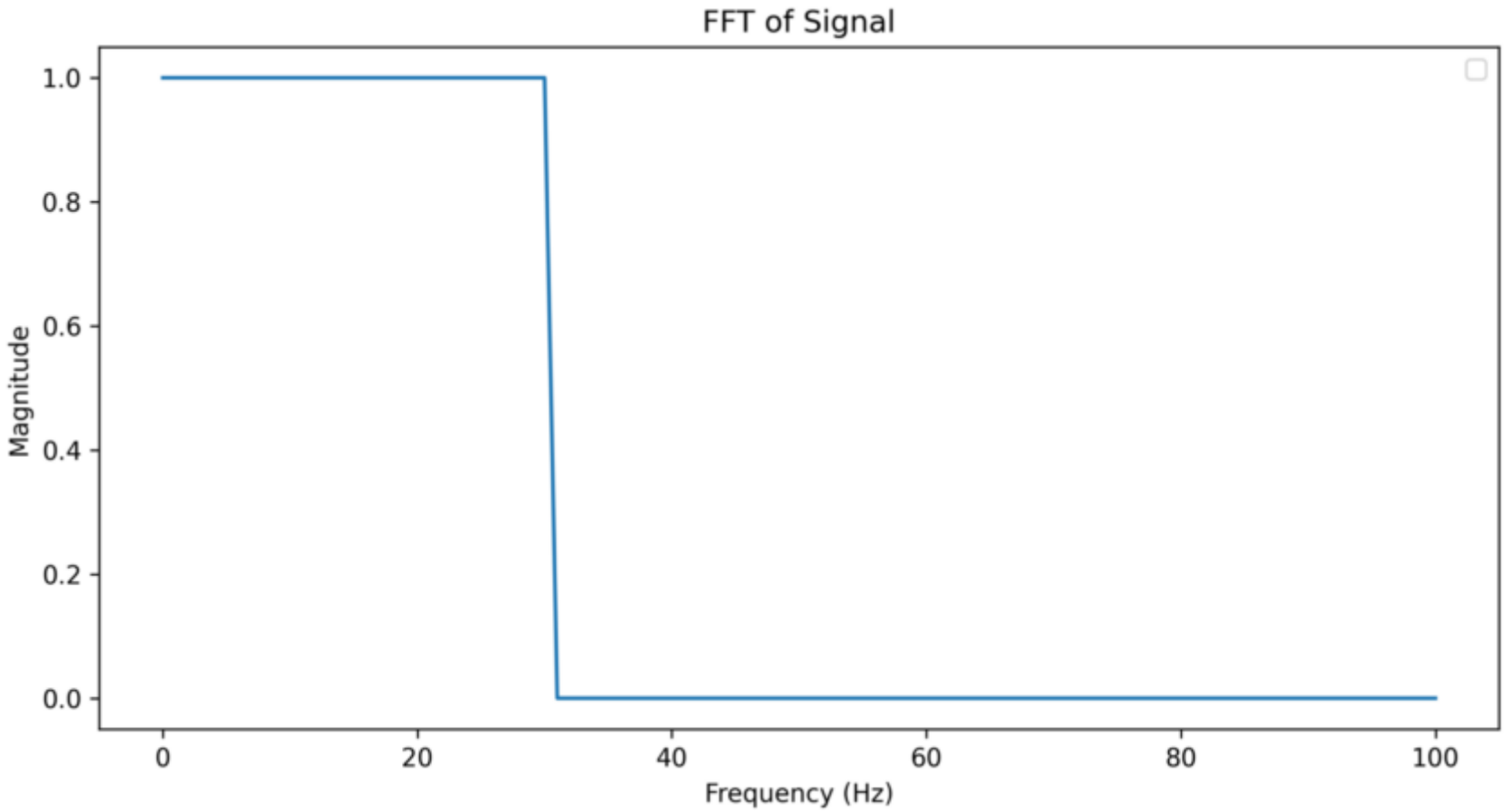
$$y[n] = \{9 \quad 9 \quad 11 \quad 5 \quad 2 \quad 0\}$$

* FIR(Finite Impulse Response) filter



$$y[n] = b_0 x[n] + b_1 x[n-1] + \cdots + b_N x[n-N]$$

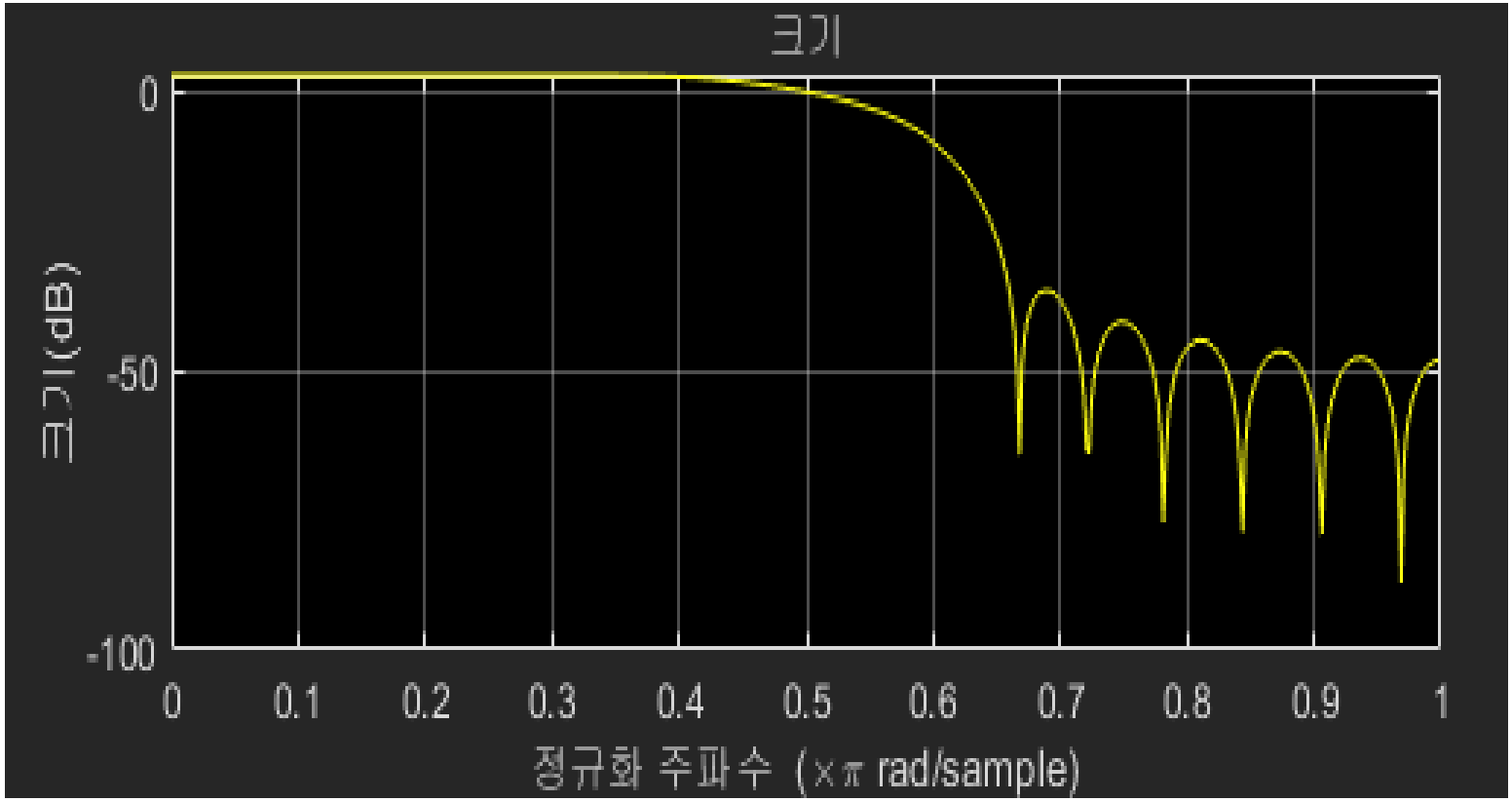
- Low Pass Filter



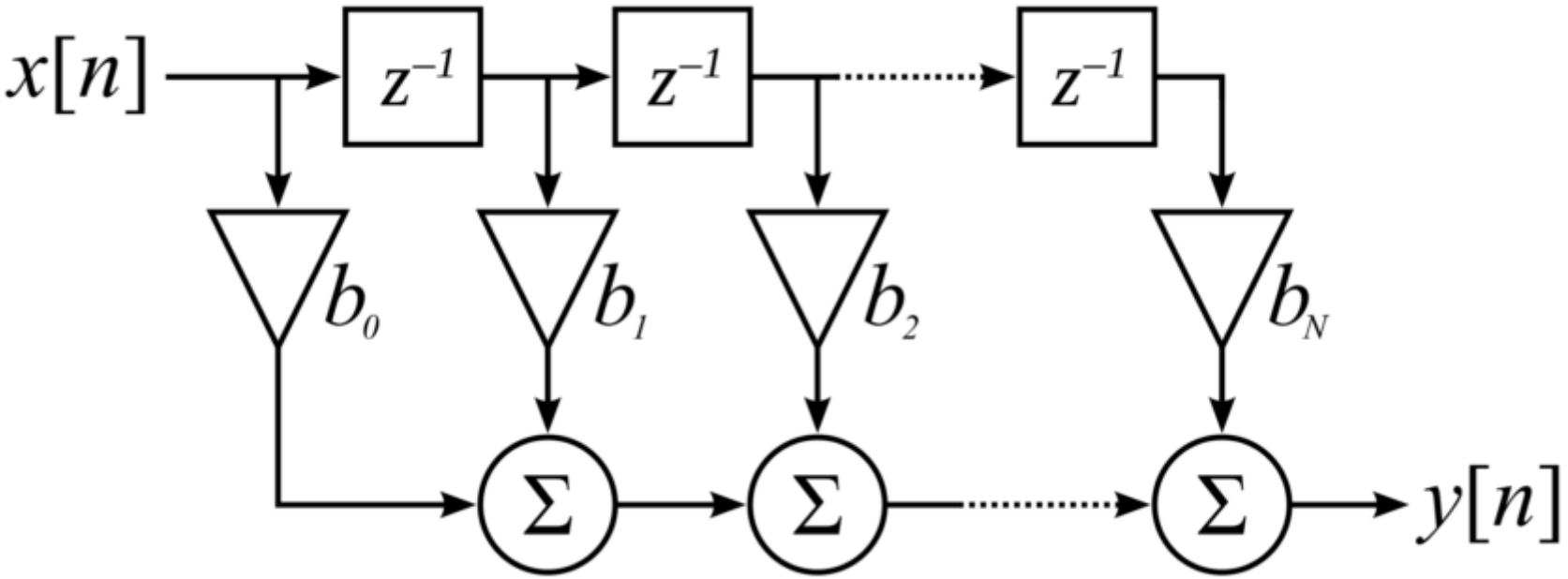
* RRC(Root Raised Cosine) filter

RRC filter coefficient (floating)																
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
-0.0006	-0.0021	0.0036	-0.0013	-0.0047	0.0061	0.0013	-0.0077	0.0084	-0.0013	-0.0234	0.0318	0.0400	-0.1086	-0.0530	0.4351	0.7651
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	
0.4351	-0.0530	-0.1086	0.0400	0.0318	-0.0234	-0.0013	0.0084	-0.0077	0.0013	0.0061	-0.0047	-0.0013	0.0036	-0.0021	-0.0006	

RRC filter coefficient (fixed : <1.8>)																
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	-1	1	0	-1	2	0	-2	2	0	-6	8	10	-28	-14	111	196
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	
111	-14	-28	10	8	-6	0	2	-2	0	2	-1	0	1	-1	0	

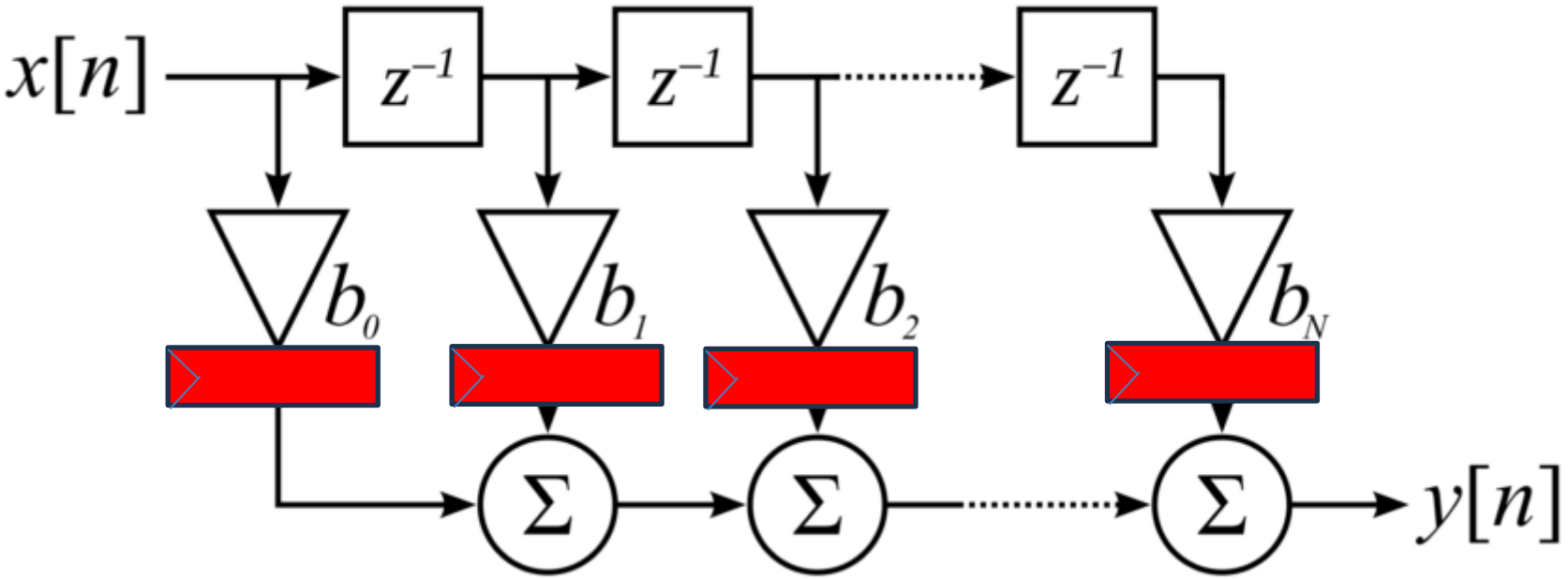

크기
0
-50
-100
크기(dB)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
정규화 주파수 (×π rad/sample)

* Example 1 (RRC original)



- Timing violated (Setup) @ 1.25 GHz

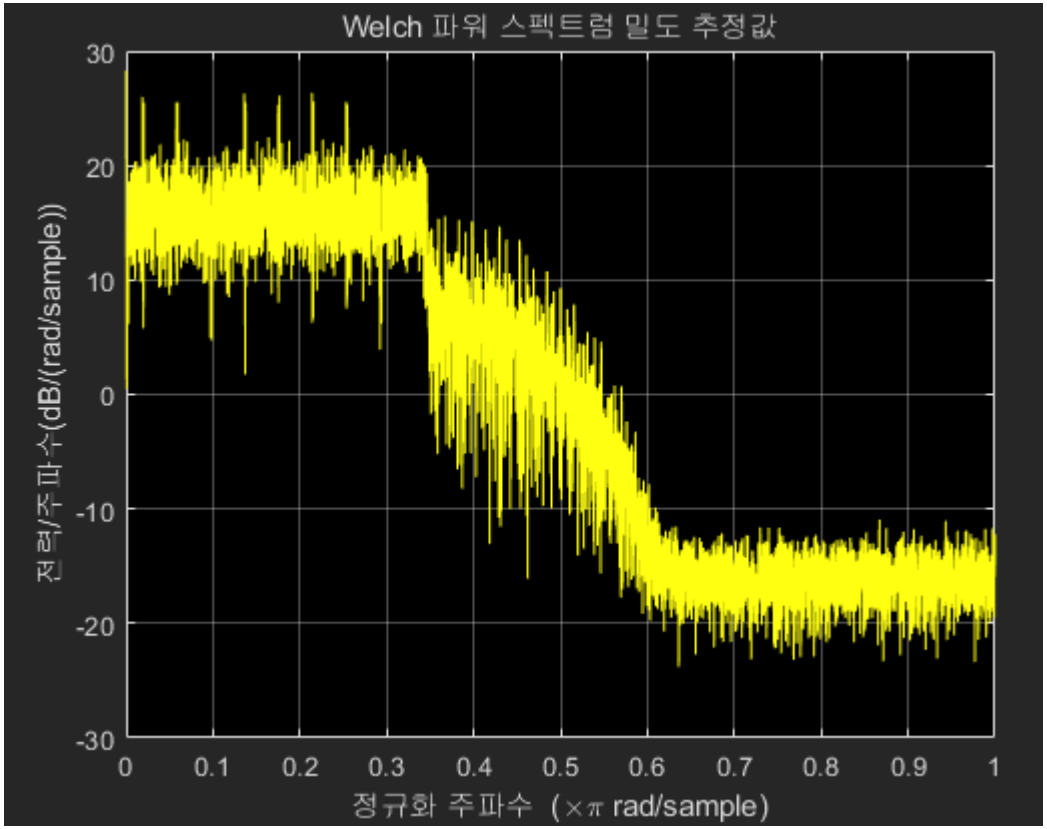
* Example 2 (RRC pipe)



- Timing met (Setup) @ 1.25 GHz

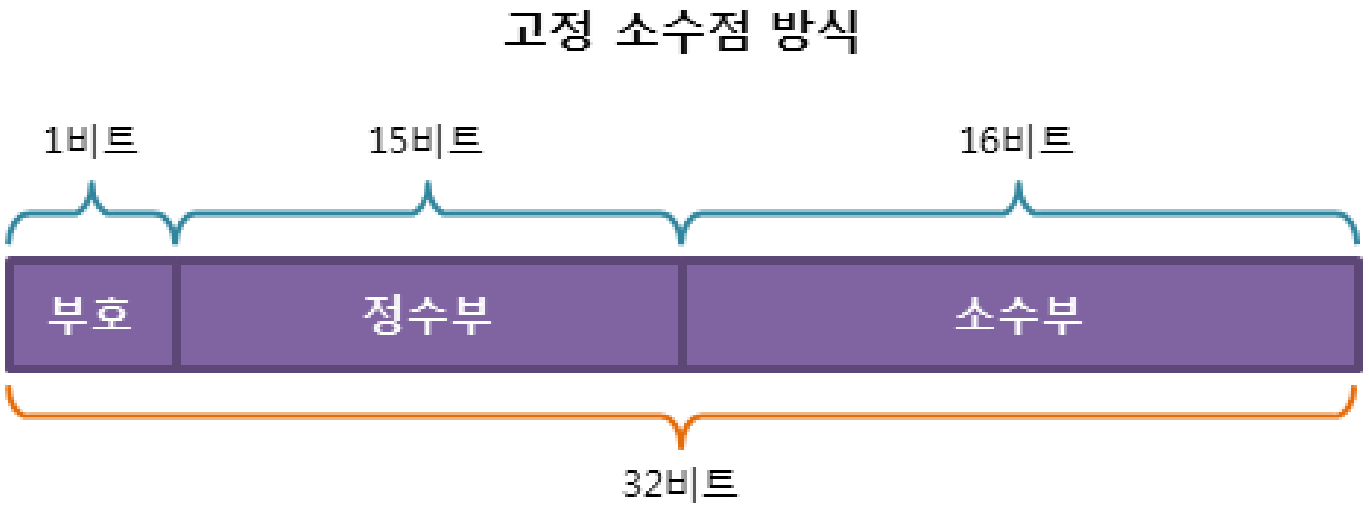
* Simulation Result (RRC original)

-. Matlab plot



* Fixed point

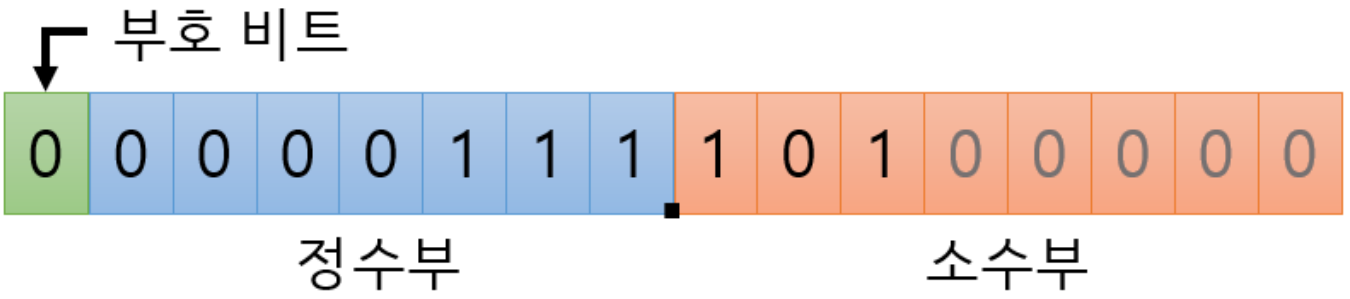
- 32 bit CPU



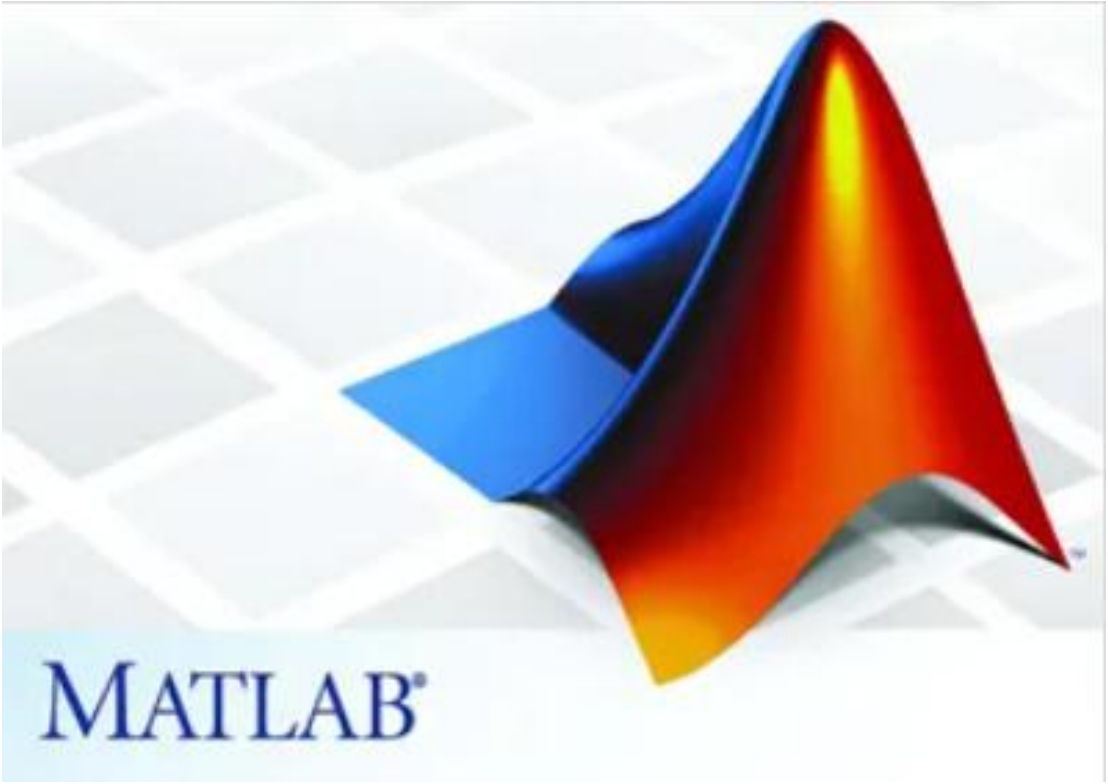
- Example (floating : 7.625 => fixed : 0000011110100000 <8.8>)

$$7.625_{(10)} = 2^2 \cdot 1 + 2^1 \cdot 1 + 2^0 \cdot 1 + 2^{-1} \cdot 1 + 2^{-2} \cdot 0 + 2^{-3} \cdot 1$$

$$= 111.101_{(2)}$$



* Matlab



- Company : Mathworks co.
- SW Script I/F => Programming Language (m-file)
- Various function and task provided
- Slower than C/C++ based coding
- Matrix based (if fully used, speed up)
- Generous compile, Easy to use

- matlab_basic.m

```
clc;

A = [1 2; 3 4];
B = [8 7; 6 5];

C = A + B; % Matrix Addition
D = A - B; % Matrix Subtraction
E = A * B; % Matrix Multiplication
F = A ^ 2; % Matrix Square
G = A'; % Matrix Transpose

E_1 = A .* B; % Element Multiplication
E_2 = A ./ B; % Element Division
F_1 = A .^ 2; % Element Square

[rows, cols] = size(A);
tot_sum = sum(A(:));
row_sum = sum(A,2);
col_sum = sum(A,1);
tot_mean = mean(A(:));
row_mean = mean(A,2);
col_mean = mean(A,1);
```