TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – GJ (박현우)
uc820@naver.com

목차

DSP C 프로그래밍

1) 삼각함수 합성

$$\sin A+\sin B=2\sin\frac{A+B}{2}\cos\frac{A-B}{2}$$
$$\sin A-\sin B=2\cos\frac{A+B}{2}\sin\frac{A-B}{2}$$
$$\cos A+\cos B=2\cos\frac{A+B}{2}\cos\frac{A-B}{2}$$
$$\cos A-\cos B=-2\sin\frac{A+B}{2}\sin\frac{A-B}{2}$$

$$\sin(\alpha+\beta)=\sin\alpha\cos\beta+\cos\alpha\sin\beta$$
$$\sin(\alpha-\beta)=\sin\alpha\cos\beta-\cos\alpha\sin\beta$$
$$\cos(\alpha+\beta)=\cos\alpha\cos\beta-\sin\alpha\sin\beta$$
$$\cos(\alpha-\beta)=\cos\alpha\cos\beta+\sin\alpha\sin\beta$$
$$\tan(\alpha+\beta)=\frac{\tan\alpha+\tan\beta}{1-\tan\alpha\tan\beta}$$
$$\tan(\alpha-\beta)=\frac{\tan\alpha-\tan\beta}{1+\tan\alpha\tan\beta}$$

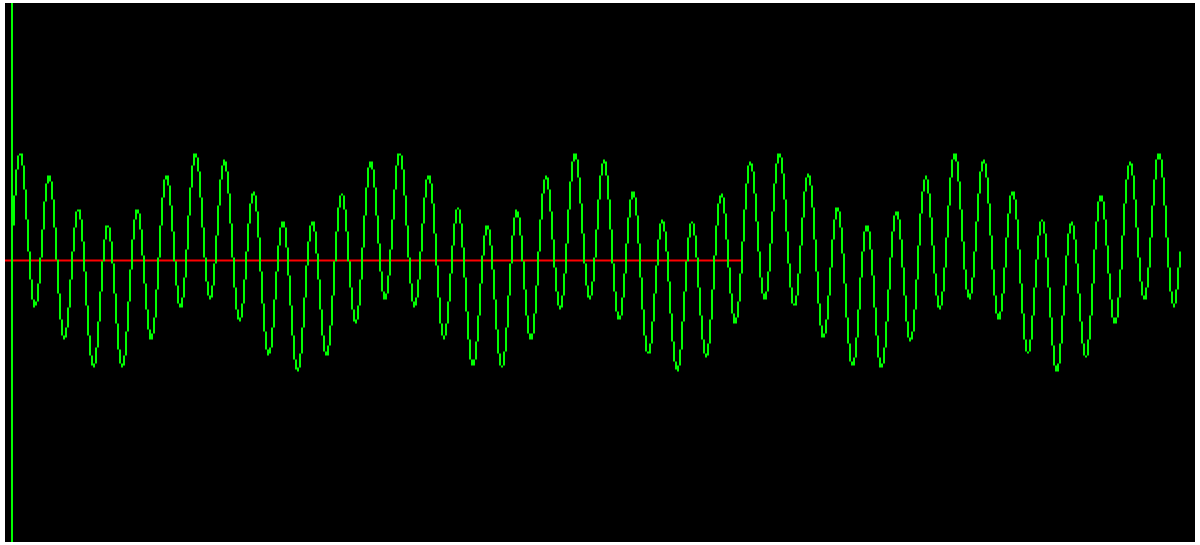
핵심 CODE

```
#define G5_PERIOD   1.0 / 5000000000.0
#define CALC_5G_2PI 10000000000 * M_PI
#define CALC_NOISE_2PI  1536000000 * M_PI
```

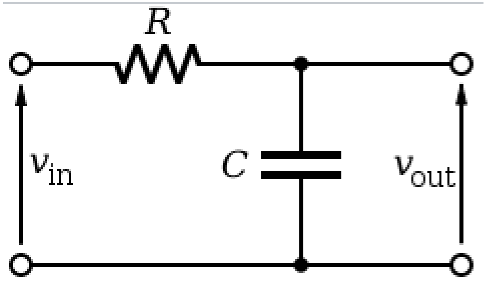
```
y2 = 10 * sin(CALC_5G_2PI * t) + 5 * cos(CALC_NOISE_2PI * t);
```

1) 삼각함수 합성

5G + 768M 합성 - SIMULATION



2) LPF



$$f_c = \frac{1}{2\pi\tau} = \frac{1}{2\pi RC}$$

$$\omega_c = \frac{1}{\tau} = \frac{1}{RC}$$

$$v_{in}(t) - v_{out}(t) = R\, i(t)$$

$$Q_c(t) = C\, v_{out}(t)$$

$$i(t) = \frac{\mathrm{d}\,Q_c}{\mathrm{d}\,t}$$

$$v_{in}(t) - v_{out}(t) = RC\frac{\mathrm{d}\,v_{out}}{\mathrm{d}\,t}$$

$$x_i - y_i = RC\,\frac{y_i - y_{i-1}}{\Delta_T}$$

$$y_i = \overbrace{x_i\left(\frac{\Delta_T}{RC+\Delta_T}\right)}^{\text{Input contribution}} + \overbrace{y_{i-1}\left(\frac{RC}{RC+\Delta_T}\right)}^{\text{Inertia from previous output}}.$$

```
y[i] := y[i-1] + α * (x[i] - y[i-1])
```

핵심 CODE

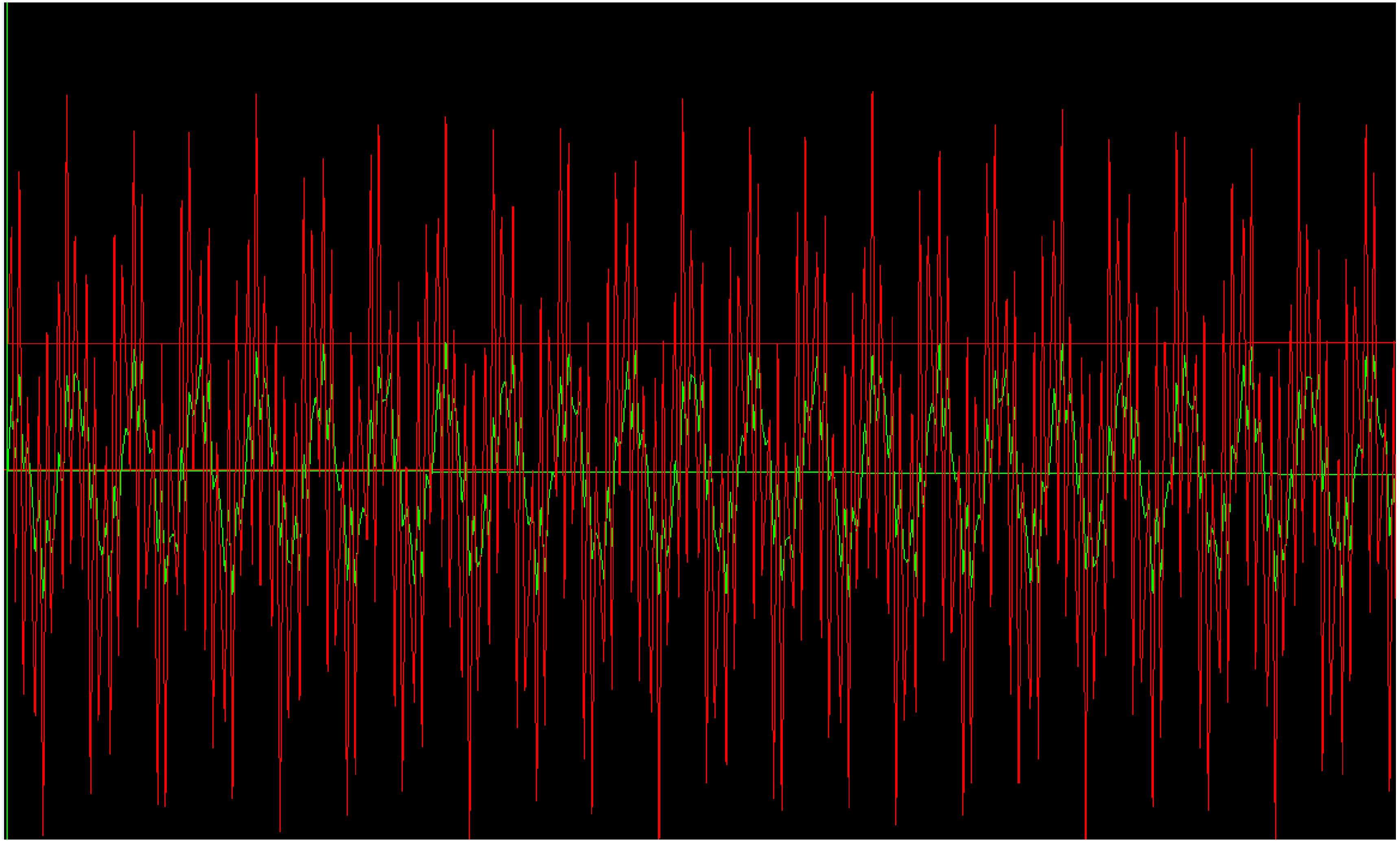
```
#define G5_PERIOD   1.0 / 5000000000.0
#define CALC_5G_2PI 10000000000 * M_PI
#define CALC_NOISE_2PI  1536000000 * M_PI
```

```
for(i = 0; i < SLICE; t += SAMPLE_PERIOD, i++)
{
    signal[i] = 10 * sin(CALC_5G_2PI * t) + 5 * cos(CALC_NOISE_2PI * t);
    printf("signal[%d] = %lf\n", i, signal[i]);
}
```

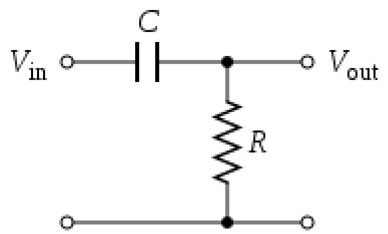
```
for(i = 1; i < SLICE; i++)
{
    lpf[i] = (rc * lpf[i - 1] + SAMPLE_PERIOD * signal[i]) / (rc + SAMPLE_PERIOD);
```

2) LPF

LPF - SIMULATION



3) HPF



$$\begin{cases} V_{out}(t) = I(t)\,R & \text{(V)} \\ Q_c(t) = C\,(V_{in}(t) - V_{out}(t)) & \text{(Q)} \\ I(t) = \frac{\mathrm{d}\,Q_c}{\mathrm{d}\,t} & \text{(I)} \end{cases}$$

$$V_{out}(t) = \overbrace{C\left(\frac{\mathrm{d}\,V_{in}}{\mathrm{d}\,t} - \frac{\mathrm{d}\,V_{out}}{\mathrm{d}\,t}\right)}^{I(t)} R = RC\left(\frac{\mathrm{d}\,V_{in}}{\mathrm{d}\,t} - \frac{\mathrm{d}\,V_{out}}{\mathrm{d}\,t}\right)$$

$$y_i = RC\left(\frac{x_i - x_{i-1}}{\Delta_T} - \frac{y_i - y_{i-1}}{\Delta_T}\right)$$

$$y_i = \overbrace{\frac{RC}{RC + \Delta_T} y_{i-1}}^{\text{Decaying contribution from prior inputs}} + \overbrace{\frac{RC}{RC + \Delta_T}(x_i - x_{i-1})}^{\text{Contribution from change in input}}$$

```
y[i] := α * (y[i-1] + x[i] - x[i-1])
```

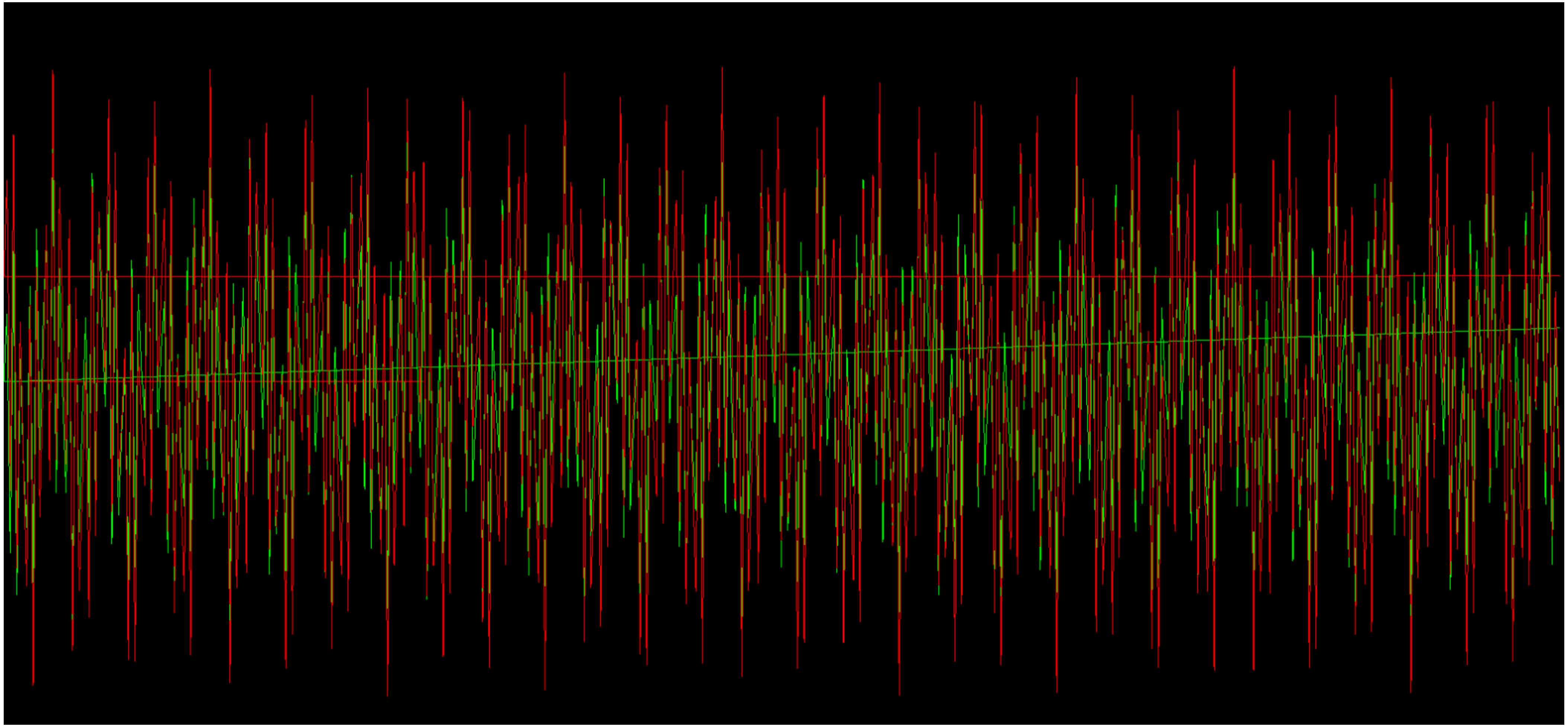
핵심 CODE

```
#define G5_PERIOD   1.0 / 5000000000.0
#define CALC_5G_2PI 10000000000 * M_PI
#define CALC_NOISE_2PI  1536000000 * M_PI
```

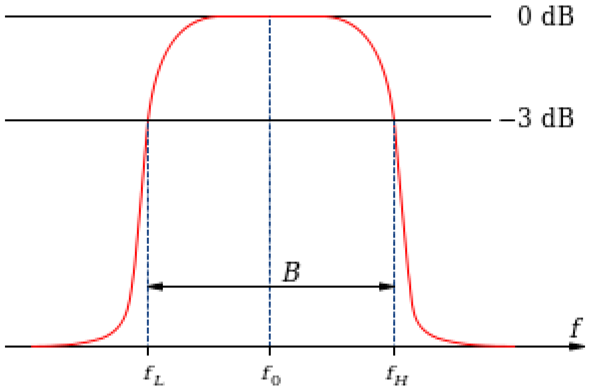
```
for(i = 1; i < SLICE; i++)
{
    hpf[i] = ( rc / (rc + SAMPLE_PERIOD) ) * ( lpf[i - 1] + signal[i] - signal[i-1] );
```

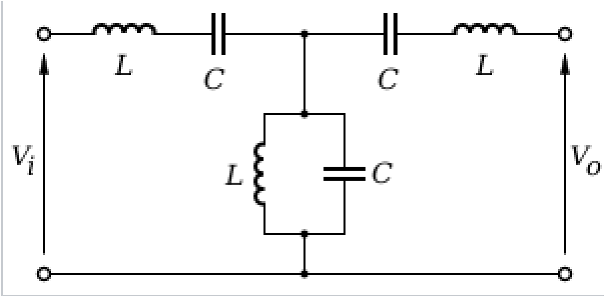
3) HPF

HPF - SIMULATION



4) BPF





A medium-complexity example of a band-pass filter.

핵심 CODE

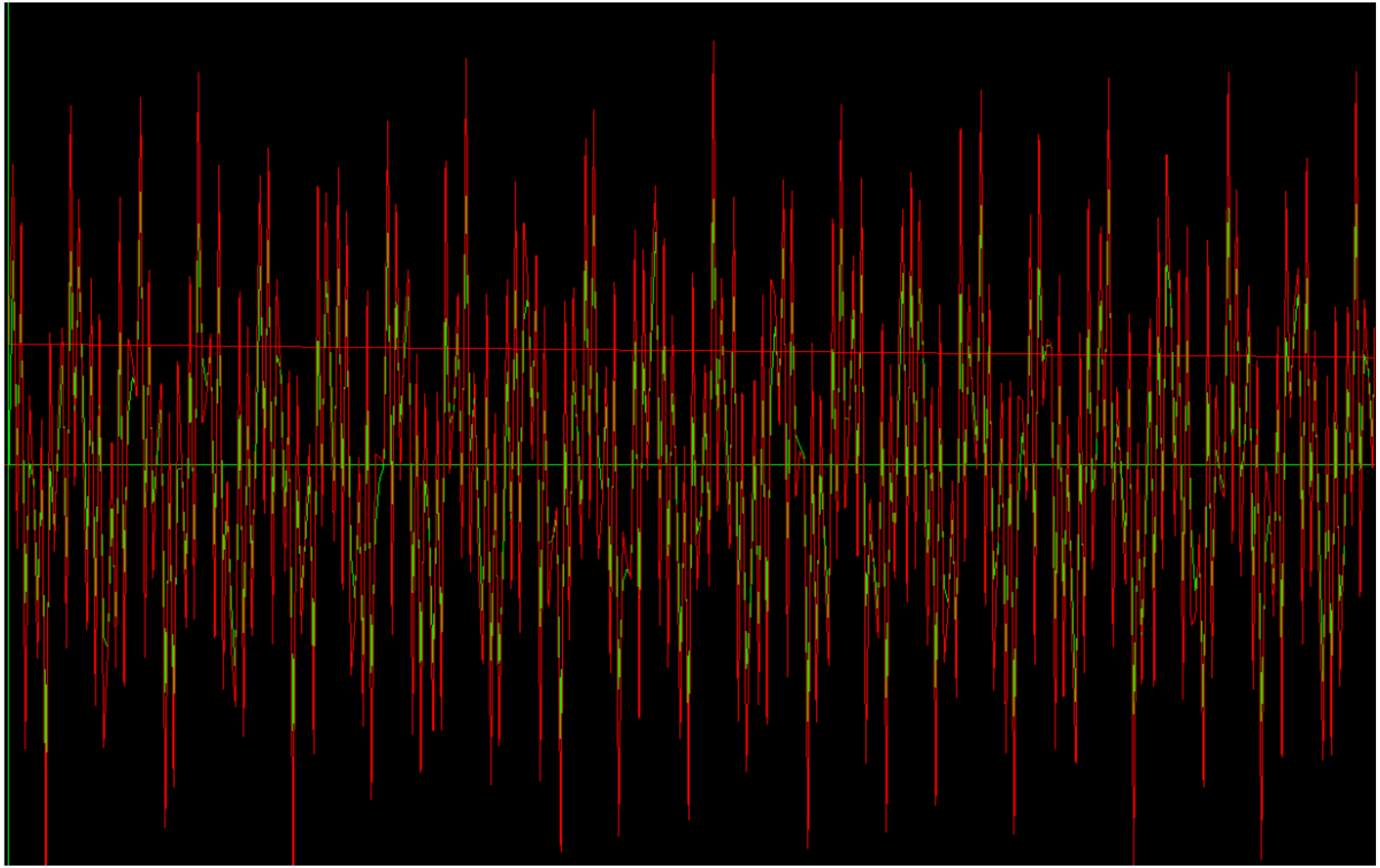
```
#define G5_PERIOD   1.0 / 5000000000.0
#define CALC_5G_2PI 10000000000 * M_PI
#define CALC_NOISE_2PI  1536000000 * M_PI


for(i = 0; i < SLICE; t += SAMPLE_PERIOD, i++)
{
    signal[i] = 10 * sin(CALC_5G_2PI * t) + 5 * cos(CALC_NOISE_2PI * t) + 3 * sin( CALC_2_4G_2PI * t);
    printf("signal[%d] = %lf\n", i, signal[i]);
}

printf("RC Low Pass Filter\n");
for(i = 1; i < SLICE; i++)
{
    lpf[i] = (rc * lpf[i - 1] + SAMPLE_PERIOD * signal[i]) / (rc + SAMPLE_PERIOD);
    hpf[i] = ( rc / (rc + SAMPLE_PERIOD) ) * ( lpf2[i - 1] + lpf[i] -lpf[i-1] );
```

4) BPF

BPF - SIMULATION



5) Fourier Transform과 Filter가 필요한 이유

Fourier Transform을 하는 이유

합성 주파수에서 어떤 주파수가 있는가를 알기 위해서 사용.

LPF나 HPF 를 써서 원하는 주파수 성분만 뽑아냄.