TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

2018-06-14 (73 회차)

강사 - Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 - 정유경

** 중요**
LPF, HPF, FFT 는 알고 넘어가자!

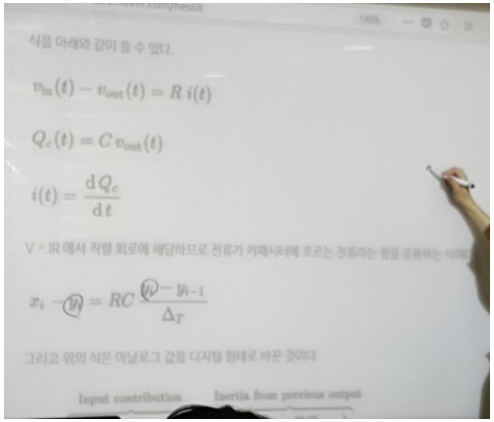
1. 5G 768Mhz 가 합쳐진 신호를 만들어 보자
일반적인 상황에서는 합쳐져 있음을 알수가 없다.
2. FFT 로 주파수 스펙트럼을 분석해야 한다.
3. 수신하려는 신호가 어떤신호인지에 따라 적절한 필터를 만들어야 한다.
필터를 만들기 위해서는 라플라스 변환이 필요하다.
센서가 잡음을 타기 때문에 그 잡음을 방지하기 위해서는 필터를 달아야 한다.
(mcu, fpga, dsp 모두 할줄 알아야 하는 이유가 이것이다)
우선 두개의 합성신호를 만들자

5G_768M_signal.c

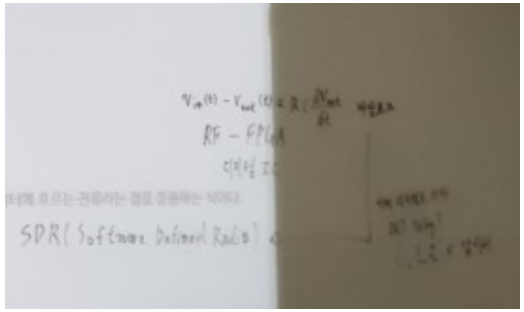
```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>

#include <GL/glut.h>

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

void draw_omega_sin(void);

float common_angles[5] = {15.0, 30.0, 45.0, 60.0, 75.0};
float freq_table[5] = {1000.0, 2400.0, 5000.0, 24000.0, 77000.0};

float theta = 0.0;

void display(void)
{
	glClearColor(0.0, 0.0, 0.0, 1.0);
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glLoadIdentity();

	//gluLookAt(0.0, 0.0, 3.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

	glColor3f(1, 0, 0);

	glBegin(GL_LINE_LOOP);
	glVertex3f(100.0, 0.0, 0.0);
    glVertex3f(-100.0, 0.0, 0.0);
    glEnd();

    glColor3f(0.0, 1.0, 0.0);

    glBegin(GL_LINE_LOOP);
    glVertex3f(0.0, 100.0, 0.0);
    glVertex3f(0.0, -100.0, 0.0);
    glEnd();

	draw_omega_sin();
	glutSwapBuffers();
}

#if 0
void reshape(int w, int h)
{
	glViewport(0, 0, (GLsizei)w, (GLsizei)h);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluPerspective(60, (GLfloat)w / (GLfloat)h, 0.1, 100.0);
	glMatrixMode(GL_MODELVIEW);
}
#endif

void reshape(int w, int h)
{
    GLfloat n_range = 100.0f;

    if(h == 0)
        h = 1;

    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    if(w <= h)
        glOrtho(-n_range, n_range, -n_range * h / w, n_range * h /
w, -n_range, n_range);
    else
        glOrtho(-n_range * w / h, n_range * w / h, -n_range,
n_range, -n_range, n_range);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}

void keyboard(unsigned char key, int x, int y)
{
	switch(key)
	{
		case 27:
			exit(0);
			break;
	}
}

void set_rand_amplitude(float *amp)
{
	*amp = rand() % 3 + 3;
}

void set_angle_with_common_angles(float *angle)
{
	*angle = common_angles[rand() % 5];
}

void angle2radian(float *angle, float *radian)
{
	*radian = *angle * M_PI / 180.0;
}

void radian2angle(float *angle, float *radian)
{
	*angle = *radian * 180.0 / M_PI;
}

void set_rand_frequency(float *freq)
{
	*freq = freq_table[rand() % 5];
}

void calc_period(float *freq, float *period)
{
	*period = 1 / (*freq);
}

void calc_angular_velocity(float *freq, float *ang_vel)
{
	*ang_vel = 2 * M_PI * (*freq);
}

float get_step(float slice, float period)
{
	return period / slice;
}
void draw_omega_sin(void)
{
	float amp, angle, period, freq, rad, omega, t, step = 0.0;
	float radius = 3.0;
	float x = 0, x2 = 0, y2, cx, cy;
	float tmp;
	int cache = 0;

	srand(time(NULL));

#if 0
	set_rand_amplitude(&amp);
	set_angle_with_common_angles(&angle);
	angle2radian(&angle, &rad);
	set_rand_frequency(&freq);
	calc_period(&freq, &period);
	calc_angular_velocity(&freq, &omega);
#endif

#if 1
	amp = 10;
	angle = 45.0;
	freq = 100.0;

	angle2radian(&angle, &rad);
	calc_period(&freq, &period);
	calc_angular_velocity(&freq, &omega);
#endif

#if 0
	printf("amplitude = %f\n", amp);
```

```
		printf("angle = %f degree\n", angle);
		printf("radian = %f\n", rad);
		printf("frequency = %f\n", freq);
		printf("period = %f\n", period);
		printf("angular_velocity = %f\n", omega);
#endif

		//t = step = get_step(SLICE, period);
		step = G5_PERIOD / 32.0;
		t = 0.0;

		//printf("t = %f\n", t);
#if 1
		if(t > 40 * G5_PERIOD)
			t = 0.0;
#endif

		glBegin(GL_LINES);
		for(; ; t += step)
		{
			if(t > 40 * G5_PERIOD)
			{
				break;
				t = 0.0;
			}

			//float rad_angle = angle * (M_PI / 180.0);
			//x2 += x;			// time += step;
			//x2 += 0.1;
			//y2 = amp * sin(CALC_5G_2PI * t);
			y2 = 10 * sin(CALC_5G_2PI * t) + 5 *
cos(CALC_NOISE_2PI * t);
			//y2 = radius * sin((double)rad_angle);

#if 1
			if(cache)
			{
				glVertex2f(cx * 20000000000.0, cy);
				glVertex2f(t * 20000000000.0, y2);
			}
#endif

#if 0
			glVertex2f(t * 40000000000.0, y2);
#endif
			cache = 1;
			cx = t;
			cy = y2;
			//printf("t = %f, y2 = %f\n", t * 4000, y2);
		}
		glEnd();
}

int main(int argc, char **argv)
{
		float amplitude, angle, period, frequency, radian,
angular_velocity;
		float step = 0.0;

		glutInit(&argc, argv);
		glutInitDisplayMode(GLUT_DOUBLE);
		glutInitWindowSize(800, 600);
		glutInitWindowPosition(0, 0);
		glutCreateWindow("Digital Signal Processing");

#if 0
		srand(time(NULL));

		set_rand_amplitude(&amplitude);
		set_angle_with_common_angles(&angle);
		angle2radian(&angle, &radian);
		set_rand_frequency(&frequency);
		calc_period(&frequency, &period);
		calc_angular_velocity(&frequency, &angular_velocity);

		printf("amplitude = %f\n", amplitude);
		printf("angle = %f degree\n", angle);
		printf("radian = %f\n", radian);
		printf("frequency = %f\n", frequency);
		printf("period = %f\n", period);
		printf("angular_velocity = %f\n", angular_velocity);

		cos_sim(amplitude, angular_velocity, period);
		sin_sim(amplitude, angular_velocity, period);
#endif

		glutDisplayFunc(display);
		//glutIdleFunc(display);
		glutReshapeFunc(reshape);
		//glutKeyboardFunc(keyboard);
		glutMainLoop();

		return 0;
}
```

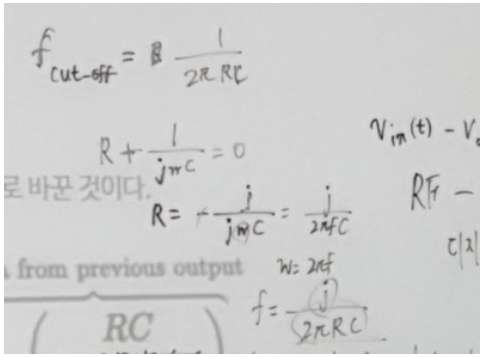
2. 일반적인 RC LPF 를 보면 다음과 같이 쓸 수 있다.



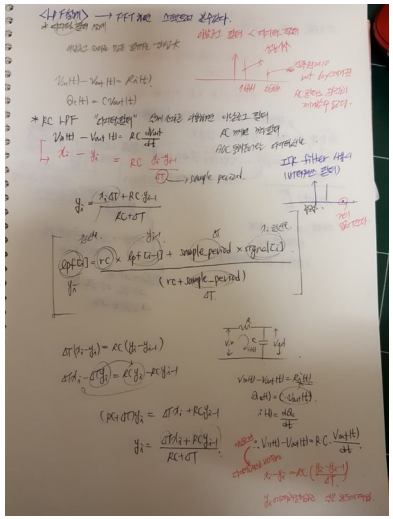
바로 위의 식이 아날로그 필터보다 성능이 매루 뛰어난 디지털 필터가 된다.
이를 yi 에 대하여 정리한다

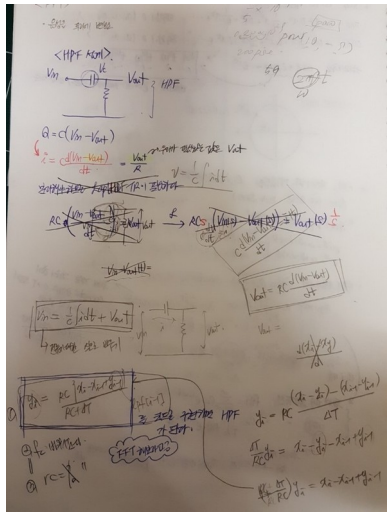


차단주파수는 아래와 같이 계산한다.



필터 식 계산하기





RC 필터를 라플라스 변환 하면 시스템분석이 가능하다.
SDR(software defined radio) 무선분야에서 사용하는 소프트웨어 필터
아날로그 필터보다 성능이 뛰어난 디지털 필터
즉, SW 로 필터설계 하려면 FPGA 를 사용해야 함
FPGA 가 TR 을 프로그래밍 할수 있는 언어이기 때문
C,L 은 주변환경의 간섭에 매우 민감하다.
제기능을 할수 없다.
위성통신 300Ghz.....저항조차도 제기능을 못하는 고주파

과거에는 이 문제를 해결하기 위해 PLL 을 이용했다.
PLL 은 기본적으로 피드백 제어를 함
PID 제어기와 같은 원리
오차 보정하라고 Feedback 넣어줌

fc cutoff freq
f = 1/ 2 pi rc

FFT 하는 이유 ** 중요 **
섞여있는 신호를 보고 육안으로 어떤 주파수가 들어있는지 알수 없다.
FFT 를 통해서 어떤 주파수가 들어있는지 알수 있다
5G 768Mhz 합쳐놓고 FFT 를 해보니까 주파수 분리가 되었다
FFT 잘 안되면 DFT 라도 쓰자. 근데 시간이 엄청 많이 걸린다.

LPF, HPF, BPF 어떤걸 쓸건지 결정하기 위해 푸리에 트랜스폼을 해준 것이다.
정 힘들면 LPf + HPF 하면 BPF 되고 노치 필터와 유사하나 성능은 두배 떨어진다.
IIR, FIR 을 하면 훨씬 깨끗하게 나올것

5G + 768M + 2.4G 있을때
LPF + HPF 사용해서 2.4G 를 가장 크게 살려놓으면 된다.

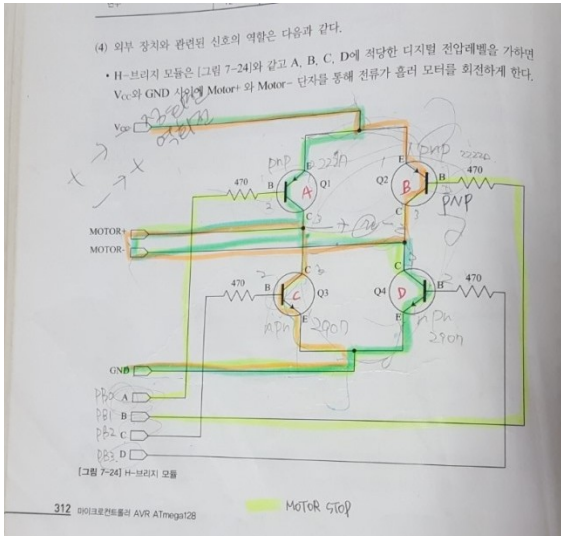
추가적으로 다음을 참고하자
http://cafe.naver.com/plduser/14620
IIR N 차 Butterworth LPF 설계 및 구현

lpf_signal.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <math.h>

#include <GL/glut.h>

int glob = 2;
int count = 1;

#define SLICE                   (1024)
#define HALF_SLICE              (SLICE >> 1)
#define QUAD_SLICE              (SLICE >> 2)
#define CALC_ORDER              ((HALF_SLICE) + 1)
#define CALC_HEIGHT             (SLICE - 3)

#define SAMPLE_FREQ             (12000000000.0)
#define SAMPLE_PERIOD           (1.0 / SAMPLE_FREQ)

#define CALC_5G_2PI             (10000000000 * M_PI)
//#define CALC_2_4G_2PI         (4800000000 * M_PI)
#define CALC_NOISE_2PI          (1536000000 * M_PI)

#define G5_PERIOD               (1.0 / 5000000000.0)

typedef struct complex
{
        double re;
        double im;
} c;

void draw_omega_sin(void);
void draw_spectrum(void);
void low_pass_filter(double *);
void spectrum_analysis(double *);

float common_angles[5] = {15.0, 30.0, 45.0, 60.0, 75.0};
float freq_table[5] = {1000.0, 2400.0, 5000.0, 24000.0, 77000.0};

float theta = 0.0;

void display(void)
{
        double lpf_signal[SLICE] = {0};

        glClearColor(0.0, 0.0, 0.0, 1.0);
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glLoadIdentity();

        //gluLookAt(0.0, 0.0, 3.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

        glColor3f(1, 0, 0);

        glBegin(GL_LINE_LOOP);
        glVertex3f(100.0, 0.0, 0.0);
        glVertex3f(-100.0, 0.0, 0.0);
        glEnd();

        glColor3f(0.0, 1.0, 0.0);

        glBegin(GL_LINE_LOOP);
        glVertex3f(0.0, 100.0, 0.0);
        glVertex3f(0.0, -100.0, 0.0);
        glEnd();

                //draw_omega_sin();
                //draw_spectrum();
                low_pass_filter(lpf_signal);
                spectrum_analysis(lpf_signal);
                glutSwapBuffers();
}

void reshape(int w, int h)
{
        GLfloat n_range = 100.0f;

        if(h == 0)
                h = 1;

        glViewport(0, 0, w, h);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();

        if(w <= h)
                glOrtho(-n_range, n_range, -n_range * h / w, n_range * h /
w, -n_range, n_range);
        else
                glOrtho(-n_range * w / h, n_range * w / h, -n_range,
n_range, -n_range, n_range);

        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
}

void keyboard(unsigned char key, int x, int y)
{
        switch(key)
        {
                case 27:
                        exit(0);
                        break;
        }
}

void set_rand_amplitude(float *amp)
{
        *amp = rand() % 3 + 3;
}

void set_angle_with_common_angles(float *angle)
{
        *angle = common_angles[rand() % 5];
}

void angle2radian(float *angle, float *radian)
{
        *radian = *angle * M_PI / 180.0;
}

void radian2angle(float *angle, float *radian)
{
        *angle = *radian * 180.0 / M_PI;
}

void set_rand_frequency(float *freq)
{
        *freq = freq_table[rand() % 5];
}

void calc_period(float *freq, float *period)
{
        *period = 1 / (*freq);
}

void calc_angular_velocity(float *freq, float *ang_vel)
{
        *ang_vel = 2 * M_PI * (*freq);
}

float get_step(float slice, float period)
{
        return period / slice;
}

static double rt_hypotd_snf(double u0, double u1)
{
        double y;
        double a;
        double b;
        a = fabs(u0);
        b = fabs(u1);
        if (a < b) {
                a /= b;
                y = b * sqrt(a * a + 1.0);
        } else if (a > b) {
                b /= a;
                y = a * sqrt(b * b + 1.0);
        //} else if (rtIsNaN(b)) {
        } else if(b == 0.0){
                y = b;
        } else {
                y = a * 1.4142135623730951;
        }

        return y;
}

void find_frequency(c X[SLICE], double R[CALC_ORDER], double
f[HALF_SLICE])
{
        double P2[SLICE];
        c y[SLICE];
        int k;
        for (k = 0; k < SLICE; k++) {
```

```
#if 0
                        if (X[k].im == 0.0) {
                                y[k].re = X[k].re / 256.0;
                                y[k].im = 0.0;
                        } else if (X[k].re == 0.0) {
                                y[k].re = 0.0;
                                y[k].im = X[k].im / 256.0;
                        } else {
#endif
                        y[k].re = X[k].re / SLICE;
                        y[k].im = X[k].im / SLICE;
//                      }

                        P2[k] = rt_hypotd_snf(y[k].re, y[k].im);
                        //printf("P2[%d] = %lf\n", k, P2[k]);
                }

                memcpy(&R[0], &P2[0], CALC_ORDER * sizeof(double));
                for (k = 0; k < HALF_SLICE - 1; k++) {
                        R[1 + k] = 2.0 * P2[1 + k];
                        //printf("R[%d] = %lf\n", k + 1, R[k + 1]);
                }

                for(k = 0; k < CALC_ORDER; k++)
                {
                        f[k] = SAMPLE_FREQ * k / SLICE;
                        //printf("f[%d] frequency = %lf\n", k, f[k]);
                }
}

void low_pass_filter(double *lpf)
{
        int i;
        double t = 0.0;
        double signal[SLICE] = {0};
        double fc = 800000000.0;
        double rc = 1.0 / (2 * M_PI * fc);

        printf("Original Signal\n");
        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]);
        }

        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);
                printf("lpf[%d] = %lf\n", i, lpf[i]);
        }

#if 0
        for(i = 0; i < SLICE; i++)
                fout[i] = signal[i + 1] + fc * (signal[i + 1] -
signal[i]) / SAMPLE_PERIOD;
#endif
}

void spectrum_analysis(double *lpf)
{
        double t = 0, period, freq = SLICE, step = 0.0;
        double temp_re, temp_im, twid_re, twid_im;
        double x = 0, x2 = 0, y2, cx, cy, orig_y;
        double dv0[CALC_ORDER] = {0};
        double dv1[CALC_ORDER] = {0};
        double rf[CALC_ORDER] = {0};
        double f[CALC_ORDER] = {0};
        double signal[SLICE] = {0};
        int cache;

        c y[SLICE] = {0};
        int ix = 0, ju = 0, iy = 0, tst, iheight, istart, ihi, i, j;

        //step = 2 * M_PI / SLICE;
        step = G5_PERIOD / SLICE;

        for(i = 0; i < SLICE; i++)
                printf("lpf[%d] = %lf\n", i, lpf[i]);

        for(i = 0; i < SLICE; t += SAMPLE_PERIOD, i++)
        signal[i] = 10 * sin(CALC_5G_2PI * t) + 5 *
cos(CALC_NOISE_2PI * t);

        t = 0.0;
        i = 0;

        glBegin(GL_LINES);
        for(; ; t += step)
        {
#if 1
#if 0
                if(t > 40 * G5_PERIOD)
                {
                        t = 0.0;
                        break;
                }
#endif
                if(i > 1023)
                {
                        t = 0.0;
                        break;
                }

                if(cache)
                {
                glColor3f(0.0, 1.0, 0.0);
                        glVertex2f(cx * 4000000000000.0, cy
* 5);
                        glVertex2f(t * 4000000000000.0, lpf[i]
* 5);

                glColor3f(1.0, 0.0, 0.0);
                        glVertex2f(cx * 4000000000000.0,
orig_y * 5);
                        glVertex2f(t * 4000000000000.0,
signal[i] * 5);
                }

                cache = 1;
                cx = t;
                cy = lpf[i];

                orig_y = signal[i];

                //printf("lpf[%d] = %lf\n", i, lpf[i]);
                i++;
#endif
        }
        glEnd();
        glob = 2;
        i = 0;
}

int main(int argc, char **argv)
{
        float amplitude, angle, period, frequency, radian,
angular_velocity;
        float step = 0.0;

        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_DOUBLE);
        glutInitWindowSize(400, 200);
        glutInitWindowPosition(0, 0);
        glutCreateWindow("Digital Signal Processing");

        glutDisplayFunc(display);
        glutReshapeFunc(reshape);
        glutMainLoop();

        return 0;
}
```

3. H Bridge 모듈은 DC 모터의 방향전환에 쓰인다.
BLDC 모터에는 다른 방법이 쓰임



(4) 외부 장치와 관련된 신호의 역할은 다음과 같다.

• H-브리지 모듈은 [그림 7-24]와 같고 A, B, C, D에 적당한 디지털 전압레벨을 가하면 V_{CC}와 GND 사이에 Motor+ 와 Motor- 단자를 통해 전류가 흘러 모터를 회전하게 한다.

[그림 7-24] H-브리지 모듈