TI DSP, MCU 및 Xilinx Zynq FPGA
프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 하성용
accept0108@naver.com

71 일차 (신호처리)

신호처리//수학 약한사람은 사실상 못함
MCU 나 FPGA 복습이 나은부분중 하나임

signal(신호)란?

1. Signal 은 수학적으로 1 개 이상의 독립변수의 함수로 표현된다.
2. 정보는 signal 이 변화하는 양상 속에 담겨 있다.
3. signal 은 physical Phenomenon 동작 또는 성질을 표현한다.

대표적인 예로 image(영상) data 역시 signal 로서 표현될 수있다.

그리고 RLC 회로의 경우에도 sin 파 형태를 띄는 signal 을 보여준다.

mp3 에서 나오는 음악 = 음성신호
지금 말하는것 = 음성신호
눈에보이는것 = 영상신호
눈에보이지않는것 =무선신호

디지털신호처리라 부르는이유는
아날로그신호를 해석하기에 컴퓨터는 가장안좋은단점이있음
디지털이라는거
데이터값이 이산적이라는거

실제로 플로팅값을 가져온다해도 떳다내려갔다할때 신호가 하나 수집이되는거
ADC 가 그래서 샘플링, 떳다내려갈때 수집을하게되는데
ADC 사용하겠다면 샘플링제어를 적절히해주어야함

- -
알고리즘이란 무엇인가?
우선 알고리즘이란 임의의 값을 Input 으로 받아서
또 다른 값을 Output 으로 내놓는 잘 정의된 계산 절차라고 한다.

이 Algorithm 을 용어적으로 알고 있기보다는

데이터를 어떻게 표현할것인가가 필요
프로그램은 두가지가 결합되야나느데 자료구조랑 알고리즘
데이터표현→ 자료구조(트리로 표현할것인가 스택으로 표현할것인가 섞어서쓸것인가 등)
알고리즘→ 거기에 있는 정보를 활용해서 문제를 해결하는것

신호처리는 알고리즘에 속함

데이터는 자료구조

순차적으로 들어오는데 들어온것을 기점으로 돌려야함
단순히 그렇게끝나는게 아니라 우선순위 큐방식으로 처리를 하게됨

- -
환율도 컴퓨터입장에선 전부 신호
신호의 변화량
연도에 따른 달러의 환율 변화율을 볼수있음
즉 변화율이니까 미분을 쓸수있다는것.
미분은 변화율
그러면 컴퓨터에서 신호처리를 할수있다는것
미분과 적분을 하기위해 나온게 신호처리

주기성 ex)sin x

푸리에 트랜스폼이 안통하는건 라플라스

시스템설계는 라플라스 트랜스폼

푸리에트랜스폼은 언제씀? 신호자체를 보려고할때
시그널 에스 시스템(책?)? 신호는 푸리에급수 시스템은 라플라스 트랜스폼

//기호 사진
제트트랜스폼에 -1 은 무엇을 의미?

이미 복소도메인에 있다는것
복소평면에 있다는것

시간이 하나밀려있음 그랫 시간징ㄱ이됨

- -

신호처리하기위한 준비과정

```
sudo apt-get update
sudo apt-get install build-essential
sudo apt-get install freeglut3 freeglut3-dev
sudo apt-get install glew-utils glee-dev
sudo apt-get install libglew-dev
```

문제)
sin(wt//오메가트리)를 이산신호로 만들어보라
오메가값은 적절히 줄것

$$\begin{cases} w=2\pi f \\ f=\dfrac{w}{2\pi} \end{cases}$$

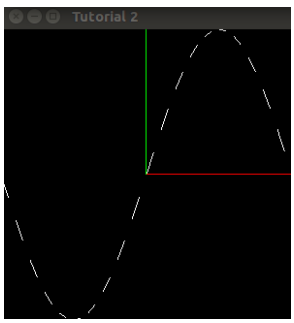
```
cd
cd /Homework/sanghoonlee/lec/dsp
yong@yong-Z20NH-AS51B5U:~/git/Homework/sanghoonlee/lec/dsp$ ls
non_anim_sin.c
```

non_anim_sin.c

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <GL/glut.h>
#include <GL/glu.h>
#include <GL/gl.h>
#include <GL/freeglut.h>
```

```
void originAxis(void); //좌표
void sineWave(void); //파장
void idle(void);

void display(void)//
{
// glClear : 설정해둔거 싹날리는거]
//  GL_DEPTH : 이전창에 설정되있는거 날려보내는거
             GL_DEPTH :
            glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);


            originAxis();
            sineWave();

            //printf("%s\n", gluErrorString(glGetError()));
            glutSwapBuffers(); // 다그렸으면 다음장면 계속 돌아가는것
}

void sineWave(void) //파장그려라
{
            float wavelength = 2; //주기 2
            float amplitude = 1; //진폭 1
            float inc = 0.05; //증가폭(샘플링)=0.05
            float k, x, y;//k=오메가(각주파수), x=계수 t 값,
            glBegin(GL_LINES); //그리겠다
            glColor3f(1,1,1); //흰색 3 원색 = 흰색
            for(x=-1;x<=1;x+=inc){
                                    //0.05 씩 증가
                        k = 2 * 3.14 / wavelength;
//전체구간에대해서 그려주겠다(증가폭 이거로잡음) → 2*3.14/π)
                        y = amplitude * sin(k * x);//증폭(작게주면 작게진동 크게주면 크게진동)
//x[0] x[1] x[2] ... x[n]
//y[0] y[1] y[2] ... y[n]
//이구간이 짧으면 이어지게보임
                        glVertex3f(x, y, 0);
            }
            glEnd();//여기서 그만두겠다

}

void idle(void) //아무것도안했을때 동작하는거
                        //코드가 사인웨이브랑 똑같음
{
            float wavelength = 2;
            float amplitude = 1;
            float inc = 0.05;
            float k, x, y;
            for (x = -1; x <= 1; x += inc){
                        glBegin(GL_POINTS);
                        glPointSize(200);
                        glColor3f(0, 1, 0);
                        k = 2 * 3.14 / wavelength;
                        y = amplitude * sin(k * x);
                        glVertex3f(x, y, 0);
                        glEnd();
            }

            glutPostRedisplay();
}
```

```
void originAxis(void) //그림의 시작과 끝
{
	glBegin(GL_LINES); //여기부터 선을
	glColor3f(1,0,0); //빨강색선을
	glVertex3f(0,0,0);
	glVertex3f(1, 0, 0); ~여기까지
	glColor3f(0,1,0); //초록색선을
	glVertex3f(0,0,0);
	glVertex3f(0, 1, 0); ~여기까지
	glColor3f(0,0,1); 파랑색선을
	glVertex3f(0,0,0);
	glVertex3f(0, 0, 1); //여기까지 그려라
	glEnd();
}


int main(int argc, char **argv)
{
	glutInit(&argc, argv); //OpenGL 초기화, 내장그래픽카드 세팅
	glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
	// GLUT_RGB : 색깔,

	// GLUT_DOUBLE : 백버퍼
	// 백버퍼란? 모니터=GLUT1, 앞에서 무엇을 그리고 있을때 뒤에 나올 화면을 미리 그려놓
	// 지않으면 버벅임. 백퍼퍼는 그려야할 내용을 그려놓고 있음다가 해당 프레임이 나오게되
	// 면 백버퍼에 그려둔게 나와서 끊김이없게됨

	//  GLUT_DEPTH : 우리가 바라보는 영상자체, 모니터 = 2D 시스템
	//                       2D 시스템에서 구현하기위해서 GLUT_DEPTH 가 필요함

	glutCreateWindow("Tutorial 2"); // 윈도우 창 제목 설정

	glOrtho(-1, 1, -1, 1, -1, 1); // 원점을 기준으로 1,1,1,1 육각형을 만들어내는것
	//우리가 바라보는 모니터에 육각형이 있다 라는걸 좌표로 만들어주는것

	glEnable(GL_DEPTH_TEST); //모니터안에 깊이값 설정

	glutDisplayFunc(display); //화면에 출력
	glutIdleFunc(idle); //일반적인 상황에서 idle 이 동작하라
	glutMainLoop(); //while 문

	return EXIT_SUCCESS;
}
```




```
gcc non_anim_sin.c -lGL -lglut -lGLU -lm
./a.out
```

끝낼때는 터미널창에서 ctrl+c

sign 파

signal_plot.c

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

#include <GL/glut.h>

#define SLICE        360

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 cos_sim(float amplitude, float ang_vel, float period)
{
	int cnt = 0;
	float step, t = 0.0;

	t = step = get_step(SLICE, period);

	while(cnt++ < 36)
	{
		printf("%.1fcos(%f * %.8f) = %f\n", amplitude, ang_vel,
			t, amplitude * cos(ang_vel * t));
		t += step;
	}
}

void sin_sim(float amplitude, float ang_vel, float period)
{
	int cnt = 0;
	float step, t = 0.0;

	t = step = get_step(SLICE, period);

	while(cnt++ < 36)
```

```
		{
			printf("%.1fsin(%f * %.8f) = %f\n", amplitude, ang_vel,
				t, amplitude * sin(ang_vel * t));
			t += step;
		}
}

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);

		//printf("t = %f\n", t);
#if 1
		if(t > period)
			t = 0.0;
```

```
#endif

		glBegin(GL_LINES);
		for(; ; t += step)
		{
			if(t > 3 * period)
			{
				break;
				t = 0.0;
			}

			//float rad_angle = angle * (M_PI / 180.0);
			//x2 += x;		// time += step;
			//x2 += 0.1;
			y2 = amp * sin(omega * t);
			//y2 = radius * sin((double)rad_angle);

			if(cache)
			{
				glVertex2f(cx * 4000, cy);
				glVertex2f(t * 4000, y2);
			}

			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(1200, 800);
		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;
}
```