Xilinx Zynq FPGA,TI DSP, MCU 기반의 프로그래밍 전문가 과정

날 짜 : 2018 . 5. 30

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 정한별
hanbulkr@gmail.com

signal (신호 및 시스템)

---

1)수학적 1개 이상의 독립변수의 함수를 표현된다.
2) 정보는 signal 이 변화하는 양상속에 담겨 있다.
3) signal 은 물리 현상 이나 동작, 성질 등을 표현한다. (대표적으로 영상, data 역시 signal로 표현된다.)

** 신호처리는 수학을 잘 써야 한다.
→ 변화율은 미분을 의미 한다. (환율 등)

***이산 신호는 불연속적인 신호이다.**
***주기, 비주기 (퓨리에 트렌스폼) ↔ (라플라스 트렌스폼)**
: 위의 두가지 트렌스폼을 왔다갔다 하면 번거로운 작업이 된다. **라플라스 적분**형태로 하면 번거로움을 줄일 수 있다.

- **시스템 프로그램 할 때, 라플라스 트렌스폼. → 제어기를 만들 때는 시스템이니까**
- **신호를 해석할 때, 퓨리에 트렌스폼 → 음성, 무선, 신호등에 씀.**

시간 지연기 : 입력 시간을 1 만큼 지역 시키는 동작을 한다. (미분을 생각하면됨)
곱셈기 : 곱셈기는 신호상수에 값을 곱해서 가중치를 준다.

신호처리를 위한 환경설정 업데이트.
1. sudo apt-get update
2. sudo apt-get install build-essential
3. sudo apt-get install freeglut3 greeglut3-dev
4. sudo apt-get install glew-utils glee-dev
5. sudo apt-get install libglew-dev

위의 업데이트를 해주면 기본적인 신호 " **처리 및 시스템** " 을 위한 업데이트 환경이 구축이 된다.

< openGL을 쓰는 것인데 옵션을 주어야 컴파일을 할 수 있다. >
-gcc "이름" -lGL -lglut -lGLU -lm

(프로그램을 만들어 보자)
ex) sin(wt)를 이산신호로 만들어 보자. [w= 2πf , f= w/2π]

---

openCV를 활용한 c언어 하기.

< 사인파 그리기 >

```
#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);
void sineWave(void);
void idle(void);

void display(void)
{
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

	originAxis();	// 축 설정.
	sineWave();	// 사이파장. 파장은 v=f*λ , c=f*λ

	//printf("%s\n", gluErrorString(glGetError()));
	glutSwapBuffers();	// 전면 버퍼와 후면 버퍼를 바꾼다.
}

void sineWave(void)
{
	float wavelength = 2;	//주기
	float amplitude = 1;	//진폭
	float inc = 0.05;//각주파수 = 증가폭(샘플링 주기)
	float k, x, y;
	glBegin(GL_LINES);
	glColor3f(1,1,1);
	for(x=-1;x<=1;x+=inc){
		k = 2 * 3.14 / wavelength;		// 2πf
		y = amplitude * sin(k * x);
		glVertex3f(x, y, 0);
	}
	glEnd();

}

void idle(void)
{
	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)
{
	glBegin(GL_LINES);
	glColor3f(1,0,0);
	glVertex3f(0,0,0);
	glVertex3f(1, 0, 0);
	glColor3f(0,1,0);
	glVertex3f(0,0,0);
	glVertex3f(0, 1, 0);
	glColor3f(0,0,1);
	glVertex3f(0,0,0);
	glVertex3f(0, 0, 1);
	glEnd();
}

int main(int argc, char **argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
	glutCreateWindow("Tutorial 2");

	glOrtho(-1, 1, -1, 1, -1, 1);
	glEnable(GL_DEPTH_TEST);

	glutDisplayFunc(display);
	glutIdleFunc(idle);
	glutMainLoop();

	return EXIT_SUCCESS;
}
```

< 사각파 그리기 >

```
#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);
void tanWave(void);
void idle(void);

void display(void)
{
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

	originAxis();	// 축 설정.
	tanWave();
	//printf("%s\n", gluErrorString(glGetError()));
	glutSwapBuffers();	// 전면 버퍼와 후면 버퍼를 바꾼다.
}

void tanWave(void)
{
	float wavelength = 2;	//주기
	float amplitude = 1;	//진폭
	float inc = 0.005;	//각주파수 = 증가폭(샘플링 주기)
	float k, x, y;
	int n;

	glBegin(GL_LINES);
	glColor3f(1,1,1);

	for(x=-30;x<=30;x+=inc){
	//	k = 2 * 3.14 / wavelength;	// 2πf
		y=0.5;
		for(n=1 ; n<6000; n++)
			y += (1/(n*3.14))*(1-cos(n*3.14))*sin(n*x);
//		for(n= 1; n< 5000 ; n++){
//			if(n%2) y += 4/(n*3.14)*sin(n*x);
//			else y+=0;
//		}

		glVertex3f(x, y, 0);
```

```
	}
	glEnd();

}

void idle(void)
{
	float wavelength = 2;
	float amplitude = 1;
	float inc = 0.02;
	float k, x, y;
	int n;
	for (x = -50; x <= 50; x += inc){
		glBegin(GL_POINTS);
		glPointSize(150);
		glColor3f(0, 1, 0);
		//k = 2 * 3.14 / wavelength;
		for(n= 1; n< 1000 ; n++){
			if(n%2) y += 4/(n*3.14)*sin(n*x);
			else y+=0;
		}
		glVertex3f(x, y, 0);
		glEnd();
	}

	glutPostRedisplay();
}

void originAxis(void)
{
	glBegin(GL_LINES);
	glColor3f(1,0,0);
	glVertex3f(0,0,0);
	glVertex3f(30, 0, 0);
	glColor3f(0,1,0);
	glVertex3f(0,0,0);
	glVertex3f(0, 30, 0);
	glColor3f(0,0,1);
	glVertex3f(0,0,0);
	glVertex3f(0, 0, 30);
	glEnd();
}

int main(int argc, char **argv)
{
	glutInit(&argc, argv);    //openGL의 초기값을 설정.내장그래픽 세팅
	glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
	// RGD 색을 쓰기 위한 옵션.
	// double에서 back buffer에서 미리 그림을 그리고 1:모니터가 그 후에 그리는 모드이다.
```

```
	// depth는 2D 시스템에서 3D를 구현하기 위해 필요한 옵션.
	glutInitWindowSize(500, 500);
	glutInitWindowPosition(100, 100);
	glutCreateWindow("Tutorial 2"); // 만든 윈도우 창의 제목 설정.

	glOrtho(-30, 30, -30, 30, -30, 30); // 원점을 기준으로 1,1,1,1,육각형을 만듬
	glEnable(GL_DEPTH_TEST); // 모니터안에 깊이값을 설정해준다.

	glutDisplayFunc(display); // 이제 화면에 뿌려.
	glutIdleFunc(idle); // 일반적인 상황에서 아이들이 돌아가라.
	glutMainLoop(); // while문인것 같다.

	return EXIT_SUCCESS;
}
```

< 삼각파 그리기 >

```
#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);
void tanWave(void);
void idle(void);

void display(void)
{
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

	originAxis();	// 축 설정.
	tanWave();
	//printf("%s\n", gluErrorString(glGetError()));
	glutSwapBuffers();	// 전면 버퍼와 후면 버퍼를 바꾼다.
}

void tanWave(void)
{
	float wavelength = 2;	//주기
	float amplitude = 1;	//진폭
	float inc = 0.001;	//각주파수 = 증가폭(샘플링 주기)
	float k, x, y;
```

```
	int n;

	glBegin(GL_LINES);
	glColor3f(1,1,1);
	for(x=-30;x<=30;x+=inc){
		y=3.14/2;
		for(n=1 ; n<1500; n++)
			y -= (4/(3.14*pow(2*n-1,2)))*cos((2*n-1)*x);

		glVertex3f(x, y, 0);
	}
	glEnd();

}

void idle(void)
{
	float wavelength = 2;
	float amplitude = 1;
	float inc = 0.02;
	float k, x, y;
	int n;
	for (x = -50; x <= 50; x += inc){
		glBegin(GL_POINTS);
		glPointSize(150);
		glColor3f(0, 1, 0);
		//k = 2 * 3.14 / wavelength;
		for(n= 1; n< 1000 ; n++){
			if(n%2) y += 4/(n*3.14)*sin(n*x);
			else y+=0;
		}
		glVertex3f(x, y, 0);
		glEnd();
	}

	glutPostRedisplay();
}

void originAxis(void)
{
	glBegin(GL_LINES);
	glColor3f(1,0,0);
	glVertex3f(0,0,0);
	glVertex3f(30, 0, 0);
	glColor3f(0,1,0);
	glVertex3f(0,0,0);
	glVertex3f(0, 30, 0);
	glColor3f(0,0,1);
	glVertex3f(0,0,0);
```

```
	glVertex3f(0, 0, 30);
	glEnd();
}

int main(int argc, char **argv)
{
	glutInit(&argc, argv);    //openGL의 초기값을 설정.내장그래픽 세팅
	glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
	// RGD 색을 쓰기 위한 옵션.
	// double에서 back buffer에서 미리 그림을 그리고 1:모니터가 그 후에 그리는 모드이다.
	// depth는 2D 시스템에서 3D를 구현하기 위해 필요한 옵션.
	glutInitWindowSize(500, 500);
	glutInitWindowPosition(100, 100);
	glutCreateWindow("Tutorial 2"); // 만든 윈도우 창의 제목 설정.

	glOrtho(-30, 30, -30, 30, -30, 30); // 원점을 기준으로 1,1,1,1,육각형을 만듬
	glEnable(GL_DEPTH_TEST); // 모니터안에 깊이값을 설정해준다.

	glutDisplayFunc(display); // 이제 화면에 뿌려.
	glutIdleFunc(idle); // 일반적인 상황에서 아이들이 돌아가라.
	glutMainLoop(); // while문인것 같다.

	return EXIT_SUCCESS;
}
```

파형이 끊기지 않게 만들기.

```
#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); // 이거를 안하면 잔상이 남는다.
        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);	// 라디안에 맞게 변경해주는거
	calc_period(&freq, &period);	// 주기 구함.
	calc_angular_velocity(&freq, &omega); // 각속도 각주파수 omega = 2파이f
#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); // 스케일을 바꾸어 준다.
			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); // 화면 크기를 바꿀때 다시 그려주는 동작이 빨리된다.
	//glutKeyboardFunc(keyboard);
	glutMainLoop();

	return 0;
}
```