TI DSP,MCU 및 Xilinux Zynq FPGA

프로그래밍 전문가 과정

이름	문지희
학생 이메일	mjh8127@naver.com
날짜	2018/6/11
수업일수	71 일차
담당강사	Innova Lee(이상훈)
강사 이메일	gcccompil3r@gmail.com

목차

신호처리할 때 필요한 프로그램

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

신호처리 프로그래밍

문제) sin(wt)를 이산신호로 만들어 보라
(w = 2*pi*f)

gcc ~~~.c -lGL -lglut -lGLU -lm 로 컴파일하기

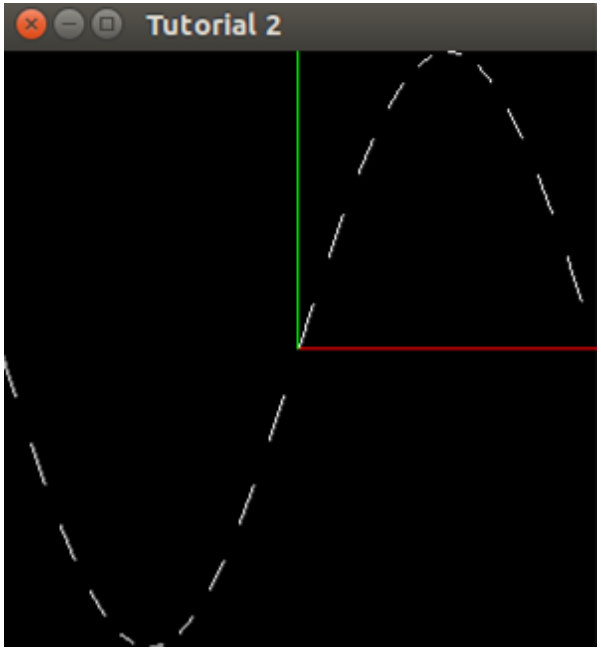
```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <GL/glut.h>
#include <GL/glu.h>
#include <GL/gl.h>
#include <GL/freeglut.h>

void originAxis(void);
void sineWave(void);
void idle(void);

void display(void)
{
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        /*비트맵 픽셀로 구성되어 있으니 clear 시켜라,
```

```
	*/
	originAxis();//좌표축설정
	sineWave();

	//printf("%s₩n", gluErrorString(glGetError()));
	glutSwapBuffers(); //다음장으로 넘기기
}

void sineWave(void)
{
	float wavelength = 2; //한 주기가 2, 파장 : 주파수*람다....
	float amplitude = 1; //진폭
	float inc = 0.005; //증가폭 (샘플링)
	float k, x, y;
	glBegin(GL_LINES); //그림
	glColor3f(1,1,1); //흰색으로 그림
	for(x=-1;x<=1;x+=inc){
		k = 2 * 3.14 / wavelength; //각주파수, f=0.5
		y = amplitude * sin(k * x);
		glVertex3f(x, y, 0);
	}
	glEnd(); //그리는것 끝

}

void idle(void) //sinwave 랑 똑같음 , 아무것도 지정하지 않았을 때 동작하는 함수
```

```
{
	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);//선을 그리는거 시작

	glColor3f(1,0,0);
	glVertex3f(0,0,0);
	glVertex3f(1, 0, 0);
	//0~1 까지 빨간선그림
```

```
        glColor3f(0,1,0);
        glVertex3f(0,0,0);
        glVertex3f(0, 1, 0);
        //0~1 까지 초록색선그림

        glColor3f(0,0,1);
        glVertex3f(0,0,0);
        glVertex3f(0, 0, 1);
        //0~1 까지  파란색선그림

        glEnd();//그리는거 끝
}

int main(int argc, char **argv)
{
        glutInit(&argc, argv);//openGL clear : 그래픽카드를 셋팅하는 역할
        glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
        /*(RGB 컬러사용,
        백버퍼에 다음에 출력할걸 저장해서  끊김없이 화면을 보여줌,
        2d 시스템에서 3d 를 구현해야하기에)*/

        glutCreateWindow("Tutorial 2"); // 제목설정

        glOrtho(-1, 1, -1, 1, -1, 1); //직교좌표계만듦, 그램슈미트
        glEnable(GL_DEPTH_TEST);//깊이값 설정
```

```
        glutDisplayFunc(display);//화면에 출력시킴
        glutIdleFunc(idle);//화면에 그림 ?? 위에거랑 먼차이야
        glutMainLoop();

        return EXIT_SUCCESS;
}
```



```
#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); // 1.0 : RGBA -a 는 알파값. 투명도를
나타냄(불투명하다)
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); //잔상이 보이지
않도록 clear 해줌
	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();
	//100~-100 범위를 그림

	 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;
```

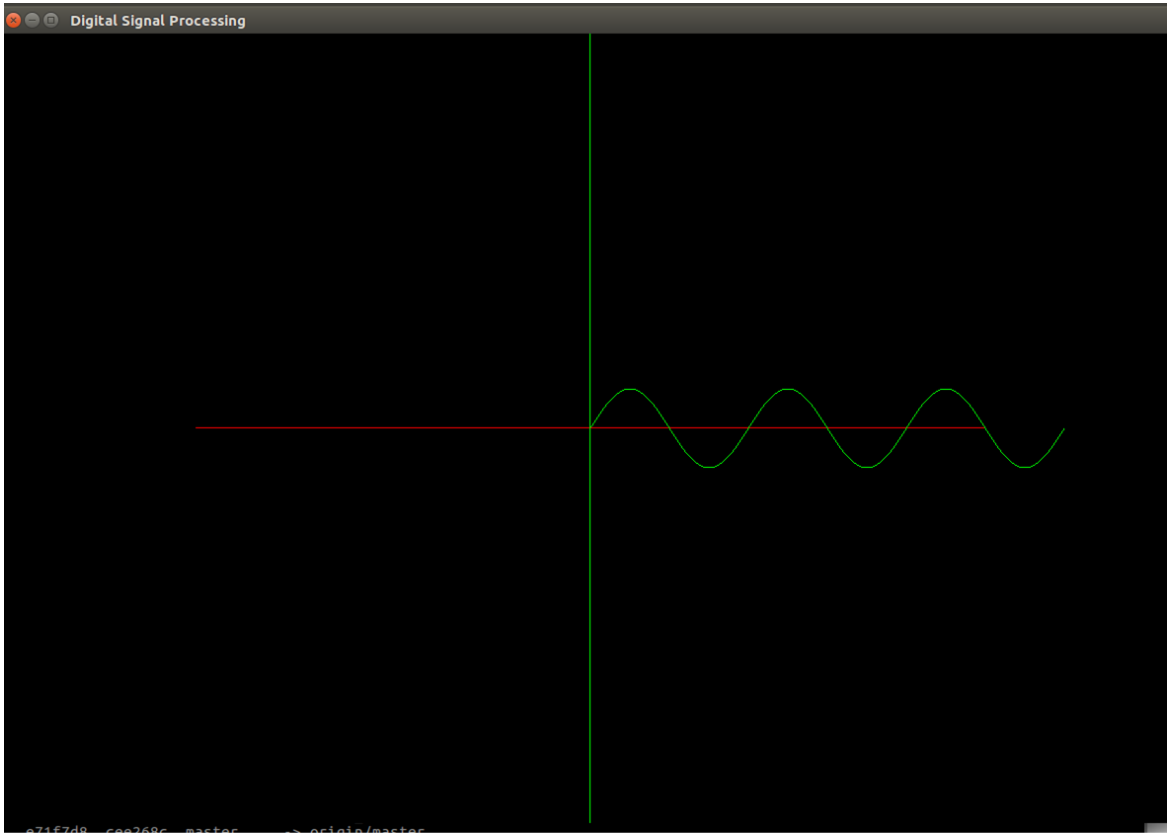
```
        }
}

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; //진폭
        angle = 45.0; //위상
        freq = 100.0; //주파수
```

```
        angle2radian(&angle, &rad); //각도값을 라디안으로 변환
        calc_period(&freq, &period); //주파수를 주기정보로 변환
        calc_angular_velocity(&freq, &omega); //각속도 구하기 w=2*pi*f
#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); // 전체 주기를 등분. (샘플링타임)

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

        glBegin(GL_LINES);
        for(; ; t += step)
        {
```

```
                if(t > 3 * period) //파형이 3 번 진동하고 끝날 때
                {
                        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); //사이즈 크기를 잡아줌
	glutInitWindowPosition(0, 0); //0,0 을 기준으로 하겠다 600,400 위치
	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); // display 함수를 그려야함
        //glutIdleFunc(display);
        glutReshapeFunc(reshape); // 사이즈를 바꿀 때 동작을 빠르게 할 수 있는
        //glutKeyboardFunc(keyboard);
        glutMainLoop(); //동작을 하기 시작함

        return 0;
}
```


Digital Signal Processing