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) OPENGL INSTALL & COMPILE

OPENGL INSTALL

```
sudo apt-get update
sudo apt-get install build-eessential
```

```
sudo apt-get install glew-utils glee-dev
sudo apt-get install libglew-dev
```

```
sudo apt-get install freeglut3 freeglut3-dev
```

COMPILE

```
gcc sin.c -lm -lGL -lglut -lGLU
```

2) FOURIER TRANSFORM으로 SIN 함수 만들기 1

CODE

```
#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); // 모니터 픽셀 초기화

    originAxis();   // 좌표축 설정.
    sineWave();

    //printf("%s\n", gluErrorString(glGetError()));
    glutSwapBuffers(); // swap 다 그렸으니 다음장면으로 넘김.
}

void sineWave(void)
{
    float wavelength = 2;   // V = f ( lamda )  주기 2
    float amplitude = 1;  // 진폭
    float inc = 0.05;   //  증가폭  ( sampling )
    float k, x, y;
    glBegin(GL_LINES);   // 시작
    glColor3f(1,1,1); // R G B   흰색
    for(x=-1;x<=1;x+=inc){
        k = 2 * 3.14 / wavelength;
        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);   // 선 그리기
    glColor3f(1,0,0);  // R G B
    glVertex3f(0,0,0);    // 0 to 1  Red
    glVertex3f(1, 0, 0);
    glColor3f(0,1,0);  // R G B
    glVertex3f(0,0,0);
    glVertex3f(0, 1, 0);
    glColor3f(0,0,1);   // R G B
    glVertex3f(0,0,0);
    glVertex3f(0, 0, 1);
    glEnd();
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);  //내장 그래픽 설정
    glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
    // RGB , DOUBLE -- monitor (glut_one), DEPTH--2d, 3d 설정

    glutCreateWindow("Tutorial 2");

    glOrtho(-1, 1, -1, 1, -1, 1); // 직교 좌표계 만들기.
    glEnable(GL_DEPTH_TEST);  // 깊이 활성화

    glutDisplayFunc(display); // 화면엔 뿌리기
    glutIdleFunc(idle);  // 일반적 상황에 작동
    glutMainLoop();  // 그림 그리기 시작

    return EXIT_SUCCESS;
}
```

2) FOURIER TRANSFORM으로 SIN 함수 만들기 2

SIN - SIMULATION

3) FOURIER TRANSFORM으로 SQUARE WAVE 함수 만들기 1 – 1

CODE

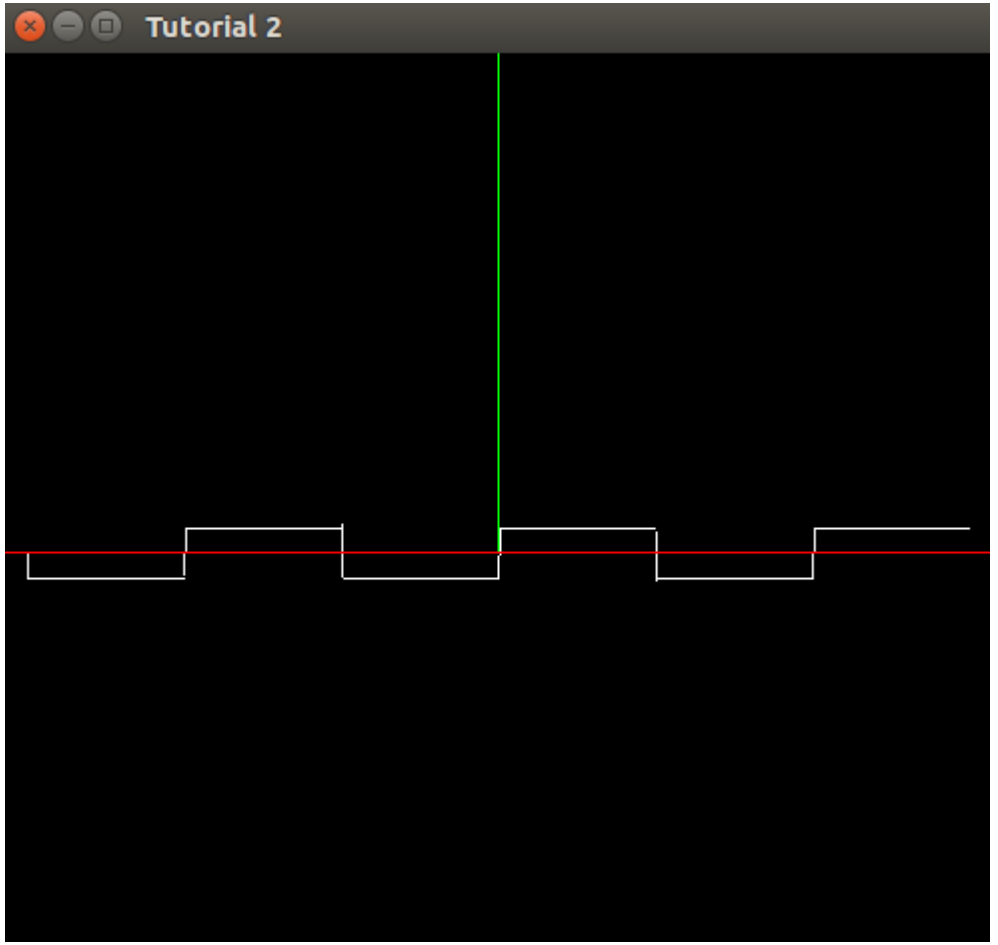
```c
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <GL/glut.h>
#include <GL/glu.h>
#include <GL/gl.h>
#include <GL/freeglut.h>

#define PI 3.14159265358979323846264338327 9

void originAxis(void);
void rectangulWave(void);
void idle(void);

void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); // 모니터 픽셀 초기화

    originAxis();   // 좌표축 설정.
    rectangulWave();

    //printf("%s\n", gluErrorString(glGetError()));
    glutSwapBuffers(); // swap 다 그렸으니 다음장면으로 넘김.
}

void rectangulWave(void)
{
    float wavelength = 2;   // V = f ( lamda )  주기 2
    float amplitude = 1;  // 진폭
    float inc = 0.01;   //  증가폭  ( sampling )
    float k, x, y;
    int n;

    glBegin(GL_LINES);   // 시작
    glColor3f(1,1,1); // R G B   흰색
    for(x= -3 *PI; x<= 3 * PI; x+=inc){

    y = 0;

        for( n = 1; n<=1000; n++){  // sigma   100개의 합
            y += (float)(1/2) + (1 - cos(n*PI)) * sin(n * x) / ( n* PI);
        }
#if 0
        for( n = 1; n<100; n++){  // sigma   100개의 합
            y += sin( (2*n -1) * x ) * ( 8 / ((2*n -1) * PI));
        }
#endif
        glVertex3f(x, y, 0);
    }
    glEnd();
```

```c
void idle(void)
{
    float inc = 0.01;
    float k, x, y;
    int n;

    for (x = -3 * PI; x <= 3 * PI; x += inc){
        glBegin(GL_POINTS);
        glPointSize(200);
        glColor3f(0, 1, 0);
        y = 0;
        for( n = 1; n<=1000; n++){  // sigma   100개의 합
            y += (float)(1/2) + (1 - cos(n*PI)) * sin(n * x) / ( n* PI);
        }
#if 0
        for( n = 1; n<100; n++){  // sigma   100개의 합
            y += sin( (2*n -1) * x ) * 8 / ((2*n -1) * PI);
        }
#endif
//      y = amplitude * cos(k * x);
        glVertex3f(x, y, 0);
        glEnd();
    }

    glutPostRedisplay();
}
```

```c
void originAxis(void)
{
    glBegin(GL_LINES);   // 선 그리기
    glColor3f(1,0,0);  // R G B
    glVertex3f(-10,0,0);    // 0 to 1  Red
    glVertex3f(10, 0, 0);
    glColor3f(0,1,0);  // R G B
    glVertex3f(0,0,0);
    glVertex3f(0, 10, 0);


    glColor3f(0,0,1);   // R G B
    glVertex3f(0,0,0);
    glVertex3f(0, 0, 10);
    glEnd();
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);  //내장 그래픽 설정
    glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
    // RGB , DOUBLE -- monitor (glut_one), DEPTH--2d, 3d 설정

    glutInitWindowSize(500, 500);
    glutCreateWindow("Tutorial 2");

    glOrtho(-10, 10, -10, 10, -10, 10); // 직교 좌표계 만들기.
    glEnable(GL_DEPTH_TEST);  // 깊이 활성화
    glutDisplayFunc(display); // 화면엔 뿌리기
    glutIdleFunc(idle);  // 일반적 상황에 작동
    glutMainLoop();  // 그림 그리기 시작

    return EXIT_SUCCESS;
}
```

3) FOURIER TRANSFORM으로 SQUARE WAVE 함수 만들기 1 – 2

SQUARE WAVE- SIMULATION



3) FOURIER TRANSFORM으로 SQUARE WAVE 함수 만들기 2 – 1

CODE

```
#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();
    sineWave();

    glutSwapBuffers();
}
```

```
void sineWave(void)
{
    float wavelength = 2.0 * M_PI;
    float amplitude = 1;
    float inc = 2.0 * M_PI / 1024.0;
    float k, x, y, yp = 0, y2, y2p = 0, cx, cy, cy2;
    int i, cache = 0;

    glBegin(GL_LINES);
    glColor3f(1,1,0);
    for(x=-3*M_PI;x<=3*M_PI;x+=inc)
    {
        yp = 0;

        for(i = 1; i < 10; i++)
            yp += ((1.0 - cos(i * M_PI)) / (i * M_PI)) * sin(i * x);

        y = yp + 0.5;

        if(cache)
        {
            glVertex2f(cx, cy);
            glVertex2f(x, y);
        }

        cache = 1;
        cx = x;
        cy = y;
    }
    glEnd();

    cache = 0;

    glBegin(GL_LINES);
    glColor3f(1,0,1);
    for(x=-3*M_PI;x<=3*M_PI;x+=inc)
    {
        yp = 0;

        for(i = 1; i < 10000; i++)
            yp += ((1.0 - cos(i * M_PI)) / (i * M_PI)) * sin(i * x);

        y = yp + 0.5;

        if(cache)
        {
            glVertex2f(cx, cy);
            glVertex2f(x, y);
        }

        cache = 1;
        cx = x;
        cy = y;
    }
    glEnd();

    cache = 0;

    glBegin(GL_LINES);
    glColor3f(1,0,1);
    for(x=-3*M_PI;x<=3*M_PI;x+=inc)
    {
        yp = 0;

        for(i = 1; i < 10000; i++)
            yp += ((1.0 - cos(i * M_PI)) / (i * M_PI)) * sin(i * x);

        y = yp + 0.5;

        if(cache)
        {
            glVertex2f(cx, cy);
            glVertex2f(x, y);
        }

        cache = 1;
        cx = x;
        cy = y;
    }
    glEnd();
}
```

3) FOURIER TRANSFORM으로 SQUARE WAVE 함수 만들기 2 – 2

CODE

```
void originAxis(void)
{
    glBegin(GL_LINES);
    glColor3f(0,0,1);
    glVertex3f(-100,0,0);
    glVertex3f(100, 0, 0);
    glColor3f(1,0,0);
    glVertex3f(0,-100,0);
    glVertex3f(0, 100, 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);
    glutInitWindowSize(800, 800);
    glutCreateWindow("Fourier Series(Rectangular Wave)");

    glOrtho(-1.1 * M_PI, 1.1 * M_PI, -0.5, 1.3, -1.0, 1.0);
    glEnable(GL_DEPTH_TEST);

    glutDisplayFunc(display);
    glutMainLoop();

    return EXIT_SUCCESS;
}
```

3) FOURIER TRANSFORM으로 SQUARE WAVE 함수 만들기 2 – 3

SQUARE WAVE- SIMULATION