ภาคผนวก ค

Source code C ของระบบ MicroBlaze

ค.1 โปรแกรมรู้จำเสียงพูดคำโดด (speech.c)

```
//Located in: microblaze_0/include/
#include "xparameters.h"
#include "xuartlite_l.h"
#include "xgpio.h"
#include "xtmrctr.h"
//================================
#define SQUARE(x) ((x)*(x))
#define SPEECH_SIZE 20000
#define FRAME_SIZE 236
#define OVERLAP 20
#define HAMMING_SIZE (FRAME_SIZE+OVERLAP)
#define NUM_FRAME (SPEECH_SIZE/FRAME_SIZE)
#define NP 256                  /* จำนวนจุดของ FFT */
#define TABLE_MAX 64            /* จำนวนค่า cos & sin ใน look-up table */
#define COEF_FB ((NP/2)+1)      /* จำนวน Power spectrum */
#define NUM_FB 8                /* จำนวน Mel-filter bank */
#define NUM_DET 6               /* จำนวนคู่ตำแหน่งของเสียงพูด */
#define NORM2MFCC 125           /* จำนวนค่าลักษณะสำคัญช่วงต้นและช่วงท้าย */
#define WB_SIZE 32328           /* จำนวน Weight & bias (8082 ค่า * 4 Byte) */
#include "preemphasis.h"
#include "feature.h"
#include "extraction.h"
#include "euclidean.h"
#include "detection.h"
#include "selection.h"
#include "neural.h"
#define CHANNEL 1
/* global variable */
XTmrCtr Timer;
XGpio GpioLED, GpioPUSH;        /* general-purpose I/O peripheral */
/*-------------------------------------------------------------------------*/
/* Main function */
/*-------------------------------------------------------------------------*/
int main(void) {
        unsigned long i;
        char choose = 0, dbg = 0, matlab = 0;

        print("\n\n\t\t-- Entering main() --\r\n");
        XStatus Sgpio = XGpio_Initialize(&GpioLED, XPAR_LEDS_DEVICE_ID);
        XGpio_mSetDataDirection(&GpioLED, XPAR_LEDS_DEVICE_ID, 0);
        Sgpio = XGpio_Initialize(&GpioPUSH, XPAR_PUSH_BUTTONS_DEVICE_ID);
        XStatus Stimer = XTmrCtr_Initialize(&Timer, XPAR_OPB_TIMER_1_DEVICE_ID);

        print("\t\t  ** Memory Map **\r\n");
```

```
        float (*XX)[2]; XX = (void *)XPAR_GENERIC_SDRAM_BASEADDR;
        xil_printf("Base addr of XX[0][0] is \t%X\r\n", XX);
        float (*X)[2]; X = XX+NP;
        xil_printf("Base addr of X[0][0] is \t%X\r\n", X);
        float (*x)[2]; x = X+NP;
        xil_printf("Base addr of x[0][0] is \t%X\r\n", x);
        unsigned char *recvBuf; recvBuf = (unsigned char *)(x+NP);
        xil_printf("Base addr of recvBuf[0] is \t%X\r\n", recvBuf);
        float *speech; speech = (float *)(recvBuf+SPEECH_SIZE);
        xil_printf("Base addr of speech[0] is \t%X\r\n", speech);
        float *emp_speech; emp_speech = speech+SPEECH_SIZE;
        xil_printf("Base addr of emp_speech[0] is \t%X\r\n", emp_speech);
        float *del; del = emp_speech+(NUM_FRAME*FRAME_SIZE);
        xil_printf("Base addr of del[0] is   \t%X\r\n", del);
        float *sp2en; sp2en = del+(NUM_FRAME*FRAME_SIZE);
        xil_printf("Base addr of sp2en[0] is   \t%X\r\n", sp2en);
        float *en; en = sp2en+NP;
        xil_printf("Base addr of en[0] is   \t%X\r\n", en);
        float *powSpec; powSpec = en+NUM_FRAME;
        xil_printf("Base addr of powSpec[0] is \t%X\r\n", powSpec);
        float *dct_coef; dct_coef = powSpec+((NP/2)+1);
        xil_printf("Base addr of dct_coef[0] is \t%X\r\n", dct_coef);
        float *mfcc; mfcc = dct_coef+NUM_FB;
        xil_printf("Base addr of mfcc[0] is \t%X\r\n", mfcc);
        float *ud; ud = mfcc+(NUM_FRAME*NUM_FB);
        xil_printf("Base addr of ud[0] is   \t%X\r\n", ud);
        float *MFCC; MFCC = ud+NUM_FRAME;
        xil_printf("Base addr of MFCC[0] is \t%X\r\n", MFCC);
        float *pre_out_f; pre_out_f = MFCC+NUM_FB*NUM_FRAME;
        xil_printf("Base addr of pre_out_f[0] is \t%X\r\n", pre_out_f);
        float *pre_out_b; pre_out_b = pre_out_f+NORM2MFCC;
        xil_printf("Base addr of pre_out_b[0] is \t%X\r\n", pre_out_b);
        float *result; result = pre_out_b+NORM2MFCC;
        xil_printf("Base addr of result[0] is \t%X\r\n", result);
        unsigned char *wb_Buf; wb_Buf = (unsigned char *)(result+7);
        xil_printf("Base addr of wb_Buf[0] is   \t%X\r\n", wb_Buf);
        float (*iw_f)[125]; iw_f = (void *)(wb_Buf);
        float (*iw_b)[125]; iw_b = iw_f+25;
        float (*lw21_f)[25]; lw21_f = (void *)(iw_b+25);
        float (*lw21_b)[25]; lw21_b = lw21_f+18;
        float (*lw32_f)[18]; lw32_f = (void *)(lw21_b+18);
        float (*lw32_b)[18]; lw32_b = lw32_f+16;
        float (*lw43_f)[16]; lw43_f = (void *)(lw32_b+16);
        float (*lw43_b)[16]; lw43_b = lw43_f+7;
        float *b1_f; b1_f = (float *)(lw43_b+7);
        float *b1_b; b1_b = b1_f+25;
        float *b2_f; b2_f = b1_b+25;
        float *b2_b; b2_b = b2_f+18;
        float *b3_f; b3_f = b2_b+18;
        float *b3_b; b3_b = b3_f+16;
        float *b4_f; b4_f = b3_b+16;
        float *b4_b; b4_b = b4_f+7;
        float *use_mem; use_mem = b4_b+7;

        xil_printf("Total addrs of used memory is \t%X - %X\r\n",
XPAR_GENERIC_SDRAM_BASEADDR, use_mem);
```

```
	print("\nSend your Weight & bias to MicroBlaze\r\n");
	XGpio_DiscreteWrite(&GpioLED, CHANNEL, 8);
	i = 0;
	while (i < WB_SIZE) {
		register short k = 3;
		while(k > -1) {
			register char recv = XUartLite_RecvByte(XPAR_RS232_BASEADDR);
			if (recv == 10) { //แก้ bug ในกรณีรับค่า 10 แล้วได้ค่า 13 ด้วย (ascii ของ Enter)
				if (wb_Buf[i+k+1] = 13) {
					wb_Buf[i+k+1] = (char)10;
					k++;
				}
			}
			else {
				wb_Buf[i+k] = recv;
			}
			k--;
		}
		i += 4;
	}
	XGpio_DiscreteWrite(&GpioLED, CHANNEL, 9);

	while(choose != 27) {

		char c;
		char mode = XGpio_mGetDataReg(XPAR_PUSH_BUTTONS_BASEADDR,
CHANNEL);
		//mode 1 - ตรวจสอบการทำงานของระบบ , mode 0 - ไม่ตรวจสอบการทำงานของระบบ
		if (mode) {
			print("\nDo you want to debug?\r\n1 -> Yes, Other -> No\r\n");
			dbg = XUartLite_RecvByte(XPAR_RS232_BASEADDR);
			switch (dbg) {
				case '1':
					print(">> Print all value for debugger\r\n");
					break;
				default :
					print(">> Do not print any value\r\n");
			}
			print("\nWant to send all data to MATLAB?\r\n1 -> Yes, Other -> No\r\n");
			matlab = XUartLite_RecvByte(XPAR_RS232_BASEADDR);
			switch (matlab) {
				case '1':
					print(">> Let MATLAB store the speech data\r\n");
					break;
				default :
					print(">> Do not send the speech data to MATLAB\r\n");
			}
			print("\nPress switch and speak command (2 second) ...(((@,@))\r\n");
		}

		for(i=0; i < SPEECH_SIZE; i++) { //รับสัญญาณเสียง
			recvBuf[i] = XUartLite_RecvByte(XPAR_RS232_BASEADDR);
			if (i == 100) XGpio_DiscreteWrite(&GpioLED, CHANNEL, 0);
		}
		XGpio_DiscreteWrite(&GpioLED, CHANNEL, 9);
```

```
		if (mode) { //ตรวจสอบการทำงาน
			if (matlab == 0x31) {
				c = 0;
				while (c != 13) { //รอรับ Enter จาก MATLAB
					c = XUartLite_RecvByte(XPAR_RS232_BASEADDR);
				}
				i = 0;
				while (i < SPEECH_SIZE) { //ส่งข้อมูลให้ MATLAB
					XUartLite_SendByte(XPAR_RS232_BASEADDR,
recvBuf[i]);
					i += 1;
				}
				XGpio_DiscreteWrite(&GpioLED, CHANNEL, 0);
			}
			c = 0;
			while (c != 13) {
				c = XUartLite_RecvByte(XPAR_RS232_BASEADDR);
			}
		}

		/*************** แปลงข้อมูล ****************/
		for(i=0; i < SPEECH_SIZE; i++) {
			speech[i] = (float)(*(recvBuf+i));
			speech[i] = (speech[i]-128)/128;
		}
		/*******************************************/

		if (dbg == 0x31) { //ตรวจสอบค่าของ speech
			for (i=0; i<10; i++)
				xil_printf(" Speech(%d) = %7d\r\n", i, (int)(speech[i]*1000000));
			for (i=SPEECH_SIZE-10; i<SPEECH_SIZE; i++)
				xil_printf(" Speech(%d) = %7d\r\n", i, (int)(speech[i]*1000000));
			print("\tPress any key to continue...\r\n");
			c = XUartLite_RecvByte(XPAR_RS232_BASEADDR);
		}

		XTmrCtr_Start(&Timer, 0);
		emphasis(speech, emp_speech);
		XTmrCtr_Stop(&Timer, 0);
		Xuint32 time = XTmrCtr_GetValue(&Timer, 0);
		if (mode) { //ตรวจสอบการทำงาน
			xil_printf("Preemphasis TIME = %7d\r\n", time);
			if (dbg == 0x31) {
				for (i=NUM_FRAME*FRAME_SIZE-10;
i<NUM_FRAME*FRAME_SIZE; i++)
					xil_printf(" emp_speech[%d] = %7d\r\n", i,
(int)(emp_speech[i]*1000000));
				print("PROCESS-Preemphasis PASS..\r\n");
				print("\tPress any key to continue...\r\n");
				c = XUartLite_RecvByte(XPAR_RS232_BASEADDR);
			}
		}

		XTmrCtr_Start(&Timer, 0);
		delta(emp_speech, del, speech);
		XTmrCtr_Stop(&Timer, 0);
```

```
		time = XTmrCtr_GetValue(&Timer, 0);
		if (mode) { //ตรวจสอบการทำงาน
			xil_printf("Delta addition TIME = %7d\r\n", time);
			if (dbg == 0x31) {
				for (i=NUM_FRAME*FRAME_SIZE-10;
i<NUM_FRAME*FRAME_SIZE; i++)
					xil_printf(" Delta_speech[%d] = %7d\r\n", i,
(int)(speech[i]*1000000));
				print("PROCESS-Delta addition PASS..\r\n");
				print("\tPress any key to continue...\r\n");
				c = XUartLite_RecvByte(XPAR_RS232_BASEADDR);
			}
		}

		XTmrCtr_Start(&Timer, 0);
		feature_ex(speech, sp2en, en, x, X, XX, powSpec, dct_coef, mfcc);
		XTmrCtr_Stop(&Timer, 0);
		time = XTmrCtr_GetValue(&Timer, 0);
		if (mode) { //ตรวจสอบการทำงาน
			xil_printf("Feature extraction TIME = %7d\r\n", time);
			if (dbg == 0x31) {
				for (i=0; i<NUM_FB; i++)
					xil_printf(" Energy(%d) = %7d\r\n", i,
(int)(en[i]*1000000));
				for (i=NUM_FRAME*NUM_FB-10;
i<NUM_FRAME*NUM_FB; i++)
					xil_printf(" mfcc[%d] = %7d\r\n", i,
(int)(mfcc[i]*1000000));
				print("PROCESS-Feature extraction PASS..\r\n");
				print("\tPress any key to continue...\r\n");
				c = XUartLite_RecvByte(XPAR_RS232_BASEADDR);
			}
		}

		XTmrCtr_Start(&Timer, 0);
		euclid(mfcc, ud);
		XTmrCtr_Stop(&Timer, 0);
		time = XTmrCtr_GetValue(&Timer, 0);
		if (mode) { //ตรวจสอบการทำงาน
			xil_printf("Euclidean distance of MFCC TIME = %7d\r\n", time);
			if (dbg == 0x31) {
				for (i=0; i<NUM_FB; i++)
					xil_printf(" Euclidean(%d) = %7d\r\n", i,
(int)(ud[i]*1000000));
				print("PROCESS-Euclidean distance of MFCC PASS..\r\n");
				print("\tPress any key to continue...\r\n");
				c = XUartLite_RecvByte(XPAR_RS232_BASEADDR);
			}
		}

		unsigned long numMFCC;

		XTmrCtr_Start(&Timer, 0);
		numMFCC = detect(en, mfcc, ud, MFCC);
		XTmrCtr_Stop(&Timer, 0);
		time = XTmrCtr_GetValue(&Timer, 0);
```

```
		if (mode) { //ตรวจสอบการทำงาน
			xil_printf("Endpoint detection TIME = %7d\r\n", time);
			if (dbg == 0x31) {
				for (i=numMFCC-10; i<numMFCC; i++)
					xil_printf(" MFCC(%d) = %7d\r\n", i,
(int)(MFCC[i]*1000000));
				print("PROCESS-Endpoint detection PASS..\r\n");
				print("\tPress any key to continue...\r\n");
				c = XUartLite_RecvByte(XPAR_RS232_BASEADDR);
			}
			xil_printf("\n\tNumber of detected MFCC = %4d\r\n\n", numMFCC);
		}

		if (numMFCC > NUM_FB) {

			XTmrCtr_Start(&Timer, 0);
			size_norm(MFCC,numMFCC,pre_out_f,pre_out_b);
			XTmrCtr_Stop(&Timer, 0);
			time = XTmrCtr_GetValue(&Timer, 0);
			if (mode) { //ตรวจสอบการทำงาน
				xil_printf("Normalized feature size TIME = %7d\r\n", time);
				if (dbg == 0x31) {
					for (i=NORM2MFCC-65; i<NORM2MFCC-57; i++)
						xil_printf(" pre_out_f(%3d) = %8d\r\n", i,
(int)(pre_out_f[i]*1000000));
					for (i=57; i<65; i++)
						xil_printf(" pre_out_b(%3d) = %8d\r\n", i,
(int)(pre_out_b[i]*1000000));
					print("PROCESS-Normalized feature size PASS..\r\n");
					print("\tPress any key to continue...\r\n");
					c = XUartLite_RecvByte(XPAR_RS232_BASEADDR);
				}
			}

			XTmrCtr_Start(&Timer, 0);
			neural_net(iw_f, iw_b, lw21_f, lw21_b, lw32_f, lw32_b, lw43_f, lw43_b,
					b1_f, b1_b, b2_f, b2_b, b3_f, b3_b, b4_f, b4_b,
					pre_out_f, pre_out_b, result);
			XTmrCtr_Stop(&Timer, 0);
			time = XTmrCtr_GetValue(&Timer, 0);
			if (mode) { //ตรวจสอบการทำงาน
				xil_printf("Recognition TIME = %7d\r\n", time);
				if (dbg == 0x31) print("PROCESS-Recognition PASS..\r\n");
			}

			float max_result = 0;
			unsigned long output;
			for (i=0; i<7; i++) {
				if (dbg == 0x31)
					xil_printf(" Result(%d) = %d\r\n", i,
(int)(result[i]*1000000));
				if (result[i] > max_result) {
					max_result = result[i];
					output = i;
				}
			}
```

```
                switch (output) {
                        case 0 :
                                XGpio_DiscreteWrite(&GpioLED, CHANNEL, 1);
                                if (mode) xil_printf(" << LEFT >> %d %%\r\n",
(int)(max_result*100));
                        break;
                        case 1 :
                                XGpio_DiscreteWrite(&GpioLED, CHANNEL, 2);
                                if (mode) xil_printf(" << RIGHT >> %d %%\r\n",
(int)(max_result*100));
                        break;
                        case 2 :
                                XGpio_DiscreteWrite(&GpioLED, CHANNEL, 3);
                                if (mode) xil_printf(" << FORWARD >> %d %%\r\n",
(int)(max_result*100));
                        break;
                        case 3 :
                                XGpio_DiscreteWrite(&GpioLED, CHANNEL, 4);
                                if (mode) xil_printf(" << BACK >> %d %%\r\n",
(int)(max_result*100));
                        break;
                        case 4 :
                                XGpio_DiscreteWrite(&GpioLED, CHANNEL, 5);
                                if (mode) xil_printf(" << SPEED >> %d %%\r\n",
(int)(max_result*100));
                        break;
                        case 5 :
                                XGpio_DiscreteWrite(&GpioLED, CHANNEL, 6);
                                if (mode) xil_printf(" << SLOW >> %d %%\r\n",
(int)(max_result*100));
                        break;
                        default :
                                XGpio_DiscreteWrite(&GpioLED, CHANNEL, 7);
                                if (mode) xil_printf(" << STOP >> %d %%\r\n",
(int)(max_result*100));
                }
                if (max_result < 0.01) {
                        XGpio_DiscreteWrite(&GpioLED, CHANNEL, 0);
                        if (mode) print(" (Likely) << Undefined Word >>\r\n");
                }

                XTmrCtr_Start(&Timer, 0);
                for (i = 0; i<300000000; i++) { ; } //delay
                XTmrCtr_Stop(&Timer, 0);
                time = XTmrCtr_GetValue(&Timer, 0);
                if (mode) { //ตรวจสอบการทำงาน
                        xil_printf("Delay TIME = %7d\r\n", time);
                        print("\n\tPress any key to Continue. Esc to Exit ..\r\n");
                        choose = XUartLite_RecvByte(XPAR_RS232_BASEADDR);
                }
        }
        else {
                if (mode) { //ตรวจสอบการทำงาน
                        print("\nERROR: Can not detect MFCC!\r\n");
                        print(" 1. Number of MFCC less than or equal NUM_FB\r\n");
                        print(" 2. Number of speech segment more than NUM_DET\r\n");
                }
```

```
                        XGpio_DiscreteWrite(&GpioLED, CHANNEL, 15);
                        XTmrCtr_Start(&Timer, 0);
                        for (i = 0; i<300000000; i++) { ; } //delay
                        XTmrCtr_Stop(&Timer, 0);
                        time = XTmrCtr_GetValue(&Timer, 0);
                        if (mode) { //ตรวจสอบการทำงาน
                                xil_printf("Delay TIME = %7d\r\n", time);
                                print("\n\tPress any key to Continue. Esc to Exit ..\r\n");
                                choose = XUartLite_RecvByte(XPAR_RS232_BASEADDR);
                        }
                }

        } //end while

        print("\t\t-- Exiting main() --\r\n");
        return 0;
}
/*---------------------------------------------------------------------------*/
```

ค.2 ฟังก์ชัน Preemphasis และเพิ่มค่า Delta ให้สัญญาณ (preemphasis.h)

```
/*-----------------------------------------------------------*/
void emphasis(float *speech, float *emp_speech)
{
        register unsigned long i;
        float max_sp = 0;
        for (i=0; i<SPEECH_SIZE; i++) {
                if (speech[i] >= max_sp)    max_sp = speech[i];
        }
        emp_speech[0] = (float)0;
        for (i=0; i<(NUM_FRAME*FRAME_SIZE); i++) {
                speech[i] = speech[i]/max_sp;
                if (i>0)   emp_speech[i] = speech[i] - (float)0.95*speech[i-1];
        }
}
/*-----------------------------------------------------------*/
/*-----------------------------------------------------------*/
void delta(float *emp_speech, float *del, float *speech_del)
{
        register unsigned long i;
        float mu = 0.25;
        for (i=0; i<FRAME_SIZE; i++) {
                del[i] = mu*( 1*emp_speech[FRAME_SIZE+i] +
2*emp_speech[2*FRAME_SIZE+i] );
                del[FRAME_SIZE+i] = mu*( (-1)*emp_speech[i] +
1*emp_speech[2*FRAME_SIZE+i] + 2*emp_speech[3*FRAME_SIZE+i] );
                del[FRAME_SIZE*(NUM_FRAME-2)+i] = mu*( (-
2)*emp_speech[(NUM_FRAME-4)*FRAME_SIZE+i] + (-1)*emp_speech[(NUM_FRAME-
3)*FRAME_SIZE+i] + 1*emp_speech[(NUM_FRAME-1)*FRAME_SIZE+i] );
                del[FRAME_SIZE*(NUM_FRAME-1)+i] = mu*( (-
2)*emp_speech[(NUM_FRAME-3)*FRAME_SIZE+i] + (-1)*emp_speech[(NUM_FRAME-
2)*FRAME_SIZE+i] );
        }
```

```
	register unsigned long curFrame;
	for (curFrame=3; curFrame<=NUM_FRAME-2; curFrame++) {
		for (i=0; i<FRAME_SIZE; i++)
			del[(curFrame-1)*FRAME_SIZE+i] = mu*( (-2)*emp_speech[(curFrame-
2)*FRAME_SIZE-FRAME_SIZE+i] + (-1)*emp_speech[(curFrame-1)*FRAME_SIZE-
FRAME_SIZE+i] + 1*emp_speech[(curFrame+1)*FRAME_SIZE-FRAME_SIZE+i] +
2*emp_speech[(curFrame+2)*FRAME_SIZE-FRAME_SIZE+i] );
	}

	for (i=0; i<NUM_FRAME*FRAME_SIZE; i++) speech_del[i] = emp_speech[i] + del[i];
}
/*-----------------------------------------------------------*/
```

ค.3 ฟังก์ชันที่ใช้ในการดึงค่าลักษณะสำคัญ (feature.h)

```
/*-----------------------------------------------------------*/
float sumf(unsigned long vector_size, float *vector)
{
	register unsigned long i;
	float sum_vector = 0;
	for (i=0; i<vector_size; i++) sum_vector = vector[i]+sum_vector;
	return sum_vector;
}
/*-----------------------------------------------------------*/
/*-----------------------------------------------------------*/
void fft_rec(unsigned long N, unsigned long offset, unsigned long delta, float (*x)[2], float (*X)[2],
float (*XX)[2], float *cosTablPtr, float *sinTablPtr) /* FFT recursion */
{
	unsigned long N2 = N/2;
	unsigned long k00, k01, k10, k11;
	if (N != 2)
	{
		fft_rec(N2, offset, 2*delta, x, XX, X, cosTablPtr, sinTablPtr);
		fft_rec(N2, offset+delta, 2*delta, x, XX, X, cosTablPtr, sinTablPtr);
		unsigned char k;
		// รวม N/2-point เป็น N-point
		for(k=0; k<N2; k++)
		{
			k00 = offset + k*delta;    k01 = k00 + N2*delta;
			k10 = offset + 2*k*delta;  k11 = k10 + delta;

			register float cs, sn;
			register unsigned long p = k*delta;
			if (p > (NP/4)) { // หาค่า cos และ sin จาก table
				register unsigned long q = p-(NP/4);
				cs = -(*((cosTablPtr+TABLE_MAX)-q));
				sn = *((sinTablPtr-TABLE_MAX)+q);
			} else if (p == (NP/4)) {
				cs = *(sinTablPtr); sn = *(cosTablPtr);
			} else {
				cs = *(cosTablPtr+p); sn = *(sinTablPtr-p);
			}

			register float tmp0, tmp1;
```

```
                        tmp0 = cs * XX[k11][0] + sn * XX[k11][1];
                        tmp1 = cs * XX[k11][1] - sn * XX[k11][0];
                        X[k01][0] = XX[k10][0] - tmp0;
                        X[k01][1] = XX[k10][1] - tmp1;
                        X[k00][0] = XX[k10][0] + tmp0;
                        X[k00][1] = XX[k10][1] + tmp1;
                }
        }
        else // หาค่า 2-point DFT
        {
                k00 = offset; k01 = k00 + delta;
                X[k01][0] = x[k00][0] - x[k01][0];
                X[k01][1] = x[k00][1] - x[k01][1];
                X[k00][0] = x[k00][0] + x[k01][0];
                X[k00][1] = x[k00][1] + x[k01][1];
        }
}
/*-------------------------------------------------------------*/
/*-------------------------------------------------------------*/
void fft(unsigned long N, float (*x)[2], float (*X)[2], float (*XX)[2], float *cosTablPtr, float
*sinTablPtr)
{
        fft_rec(N, 0, 1, x, X, XX, cosTablPtr, sinTablPtr);
}
/*-------------------------------------------------------------*/
/*-------------------------------------------------------------*/
void pow_cmpx(float (*X)[2], float *powSpec)
{
        register unsigned long i;
        for (i=0; i<COEF_FB; i++) {
                powSpec[i] = ( SQUARE(X[i][0]) + SQUARE(X[i][1]) )/NP;
                //ปรับระดับ power spectrum
                if ((i >= 3)&(i <= 9)) {
                        powSpec[i] *= (float)1.2;
                }
                else if ((i >= 10)&(i <= 19)) {
                        powSpec[i] *= (float)1.4;
                }
                else if ((i >= 20)&(i <= 39)) {
                        powSpec[i] *= (float)1.3;
                }
                else if ((i >= 40)&(i <= 59)) {
                        powSpec[i] *= (float)1.2;
                }
                else if ((i >= 60)&(i <= 89)) {
                        powSpec[i] *= (float)1.1;
                }
        }
}
/*-------------------------------------------------------------*/
/*-------------------------------------------------------------*/
void mel_fft(float *powSpec, float *fb1TablPtr, float *fb2TablPtr, float *fb3TablPtr, float *fb4TablPtr,
float *fb5TablPtr, float *fb6TablPtr, float *fb7TablPtr, float *fb8TablPtr)
{
        register unsigned long i, j;
        float *tmp_mul;
        tmp_mul = (float *)(XPAR_GENERIC_SDRAM_BASEADDR); //temp addr
```

```
	i = 0;
	for (j=1; j<=10; j++) {
		tmp_mul[j] = powSpec[j]*(*(fb1TablPtr+i)); i += 1;
	}
	powSpec[0] = sumf(10,&tmp_mul[1]);
	i = 0;
	for (j=5; j<=17; j++) {
		tmp_mul[j] = powSpec[j]*(*(fb2TablPtr+i)); i += 1;
	}
	powSpec[1] = sumf(13,&tmp_mul[5]);
	i = 0;
	for (j=11; j<=26; j++) {
		tmp_mul[j] = powSpec[j]*(*(fb3TablPtr+i)); i += 1;
	}
	powSpec[2] = sumf(16,&tmp_mul[11]);
	i = 0;
	for (j=18; j<=38; j++) {
		tmp_mul[j] = powSpec[j]*(*(fb4TablPtr+i)); i += 1;
	}
	powSpec[3] = sumf(21,&tmp_mul[18]);
	i = 0;
	for (j=27; j<=53; j++) {
		tmp_mul[j] = powSpec[j]*(*(fb5TablPtr+i)); i += 1;
	}
	powSpec[4] = sumf(27,&tmp_mul[27]);
	i = 0;
	for (j=39; j<=72; j++) {
		tmp_mul[j] = powSpec[j]*(*(fb6TablPtr+i)); i += 1;
	}
	powSpec[5] = sumf(34,&tmp_mul[39]);
	i = 0;
	for (j=54; j<=96; j++) {
		tmp_mul[j] = powSpec[j]*(*(fb7TablPtr+i)); i += 1;
	}
	powSpec[6] = sumf(43,&tmp_mul[54]);
	i = 0;
	for (j=73; j<=127; j++) {
		tmp_mul[j] = powSpec[j]*(*(fb8TablPtr+i)); i += 1;
	}
	powSpec[7] = sumf(55,&tmp_mul[73]);
}
/*-----------------------------------------------------------*/
/*-----------------------------------------------------------*/
void ln_fft(float *feature)
{
	register unsigned long i;
	for (i=0; i<NUM_FB; i++) feature[i] = logf(feature[i]);
}
/*-----------------------------------------------------------*/
/*-----------------------------------------------------------*/
void dct(float *feature, float *dct_coef)
{
	register unsigned long i;
	float s07, s12, s34, s56, d12, d56, d07, d34, tmp1, tmp2, tmp3, tmp4;
	float C4 = 0.707106781186548;
	float C6 = 0.38268343236509;
```

```
	float S6 = 0.923879532511287;
	float C3 = 0.831469612302545;
	float S3 = 0.555570233019602;
	float C1 = 0.98078528040323;
	float S1 = 0.195090322016128;
	float half = 2;

	s07 = feature[0] + feature[7];
	s12 = feature[1] + feature[2];
	s34 = feature[3] + feature[4];
	s56 = feature[5] + feature[6];

	d12 = feature[1] - feature[2];
	d56 = feature[5] - feature[6];
	d07 = feature[0] - feature[7];
	d34 = feature[3] - feature[4];

	tmp1 = s07+s34;
	tmp2 = s12+s56;
	dct_coef[0] = ( (tmp1+tmp2)*(C4) )/half;
	dct_coef[4] = ( (tmp1-tmp2)*(C4) )/half;

	tmp1 = d12-d56;
	tmp2 = s07-s34;
	dct_coef[2] = ( tmp1*(C6) + tmp2*(S6) )/half;
	dct_coef[6] = ( tmp1*(-S6) + tmp2*(C6) )/half;

	tmp1 = (C4)*(s12-s56);
	tmp2 = (C4)*(d12+d56);
	tmp3 = d07-tmp1;
	tmp4 = d34-tmp2;
	dct_coef[3] = ( (C3)*tmp3 - (S3)*tmp4 )/half;
	dct_coef[5] = ( (S3)*tmp3 + (C3)*tmp4 )/half;

	tmp3 = d07+tmp1;
	tmp4 = d34+tmp2;
	dct_coef[1] = ( (C1)*tmp3 + (S1)*tmp4 )/half;
	dct_coef[7] = ( (S1)*tmp3 - (C1)*tmp4 )/half;
}
/*-------------------------------------------------------------*/
```

ค.4 ฟังก์ชันการดึงค่าลักษณะสำคัญ (extraction.h)

```
/*-------------------------------------------------------------*/
void feature_ex(float *emp_speech, float *sp2en, float *en, float (*x)[2], float (*X)[2], float (*XX)[2],
float *powSpec, float *dct_coef, float *mfcc)
{
	float *hammingPtr; hammingPtr = (float *)XPAR_TABLE_0_AR0_BASEADDR;
	float *cosTablPtr; cosTablPtr = hammingPtr+0x80;
	float *sinTablPtr; sinTablPtr = cosTablPtr+0x40;
	float *fb1TablPtr; fb1TablPtr = sinTablPtr+0x1;
	float *fb2TablPtr; fb2TablPtr = fb1TablPtr+0xa;
	float *fb3TablPtr; fb3TablPtr = fb2TablPtr+0xd;
	float *fb4TablPtr; fb4TablPtr = fb3TablPtr+0x10;
```

```
	float *fb5TablPtr; fb5TablPtr = fb4TablPtr+0x15;
	float *fb6TablPtr; fb6TablPtr = fb5TablPtr+0x1b;
	float *fb7TablPtr; fb7TablPtr = fb6TablPtr+0x22;
	float *fb8TablPtr; fb8TablPtr = fb7TablPtr+0x2b;
	register unsigned long i, j;
	for (i=0; i<(HAMMING_SIZE/2); i++) {
		x[i][0] = emp_speech[i]*(*(hammingPtr+i));
		x[i][1] = 0;
		sp2en[i] = SQUARE(x[i][0]);
	}
	j = 1;
	for (i=(HAMMING_SIZE/2); i<HAMMING_SIZE; i++) {
		x[i][0] = emp_speech[i]*(*(hammingPtr+(HAMMING_SIZE/2)-j));
		x[i][1] = 0;
		sp2en[i] = SQUARE(x[i][0]);
		j += 1;
	}
	en[0] = sumf(HAMMING_SIZE, sp2en);

	fft(NP, x, X, XX, cosTablPtr, sinTablPtr);
	pow_cmpx(X, powSpec);
	mel_fft(powSpec, fb1TablPtr, fb2TablPtr, fb3TablPtr, fb4TablPtr, fb5TablPtr, fb6TablPtr,
fb7TablPtr, fb8TablPtr);
	ln_fft(powSpec);
	dct(powSpec, dct_coef);

	for (i=0; i<NUM_FB; i++) mfcc[i] = dct_coef[i];

	unsigned long curFrame;
	for (curFrame=2; curFrame<=NUM_FRAME; curFrame++) {
		register unsigned long start = (curFrame-1)*FRAME_SIZE-OVERLAP;
		for (i=0; i<(HAMMING_SIZE/2); i++) {
			x[i][0] = emp_speech[start+i]*(*(hammingPtr+i));
			x[i][1] = 0;
			sp2en[i] = SQUARE(x[i][0]);
		}
		j = 1;
		for (i=(HAMMING_SIZE/2); i<HAMMING_SIZE; i++) {
			x[i][0] = emp_speech[start+i]*(*(hammingPtr+(HAMMING_SIZE/2)-j));
			x[i][1] = 0;
			sp2en[i] = SQUARE(x[i][0]);
			j += 1;
		}
		en[curFrame-1] = sumf(HAMMING_SIZE, sp2en);

		fft(NP, x, X, XX, cosTablPtr, sinTablPtr);
		pow_cmpx(X, powSpec);
		mel_fft(powSpec, fb1TablPtr, fb2TablPtr, fb3TablPtr, fb4TablPtr, fb5TablPtr,
fb6TablPtr, fb7TablPtr, fb8TablPtr);
		ln_fft(powSpec);
		dct(powSpec, dct_coef);

		for (i=0; i<NUM_FB; i++) mfcc[i+(NUM_FB*(curFrame-1))] = dct_coef[i];
	}
}
/*-----------------------------------------------------------*/
```

ค.5 ฟังก์ชันการหาค่า Euclidean distance ของสัมประสิทธิ์ (euclidean.h)

```
/*-------------------------------------------------------------*/
void euclid(float *mfcc, float *ud)
{
	register unsigned long i;
	float *tmp; tmp = (float *)(XPAR_GENERIC_SDRAM_BASEADDR); //temp addr
	float *mfc_mean; mfc_mean = tmp+NUM_FB;
	float *sil1; sil1 = mfc_mean+NUM_FB;
	float *sil2; sil2 = sil1+NUM_FB;
	float *sil3; sil3 = sil2+NUM_FB;
	float *sil_lst1; sil_lst1 = sil3+NUM_FB;
	float *sil_lst2; sil_lst2 = sil_lst1+NUM_FB;
	float *sil_lst3; sil_lst3 = sil_lst2+NUM_FB;
	for (i=0; i<NUM_FB; i++) {
		sil1[i] = mfcc[i + 0*NUM_FB];
		sil2[i] = mfcc[i + 1*NUM_FB];
		sil3[i] = mfcc[i + 2*NUM_FB];
		sil_lst1[i] = mfcc[(NUM_FRAME-3)*NUM_FB + i];
		sil_lst2[i] = mfcc[(NUM_FRAME-2)*NUM_FB + i];
		sil_lst3[i] = mfcc[(NUM_FRAME-1)*NUM_FB + i];
		mfc_mean[i] = (sil1[i]+sil2[i]+sil3[i]+sil_lst1[i]+sil_lst2[i]+sil_lst3[i])/6;
	}
	register unsigned long j;
	for (j=0; j<NUM_FRAME; j++) {
		for (i=0; i<NUM_FB; i++) tmp[i] = SQUARE( mfcc[i+j*NUM_FB]-mfc_mean[i] );
		ud[j] = sumf(NUM_FB,tmp);
	}
}
/*-------------------------------------------------------------*/
```

ค.6 ฟังก์ชันการตรวจหาขอบเขตของคำ (detection.h)

```
/*-------------------------------------------------------------*/
unsigned long detect(float *en, float *mfcc, float *ud, float *MFCC)
{
	register unsigned long i;
	unsigned long *bin_boundary;
	bin_boundary = (unsigned long *)(XPAR_GENERIC_SDRAM_BASEADDR); //temp addr
	long *bip_boundary;
	bip_boundary = (long *)(bin_boundary+NUM_FRAME);
	float Esil, Dsil, D_ref1, D_ref2, E_ref1, E_ref2, E_ref3;
	for (i=0; i<NUM_FRAME; i++) {
		Esil=(en[0]+en[1]+en[2]+en[NUM_FRAME-3]+en[NUM_FRAME-
2]+en[NUM_FRAME-1])/6+0.5;
		Dsil=(ud[0]+ud[1]+ud[2]+ud[NUM_FRAME-3]+ud[NUM_FRAME-
2]+ud[NUM_FRAME-1])/6;

		D_ref1 = Dsil*(float)3; E_ref1 = Esil*(float)0.75; //condition1
		D_ref2 = Dsil; E_ref2 = Esil*(float)1.1; //condition2
		E_ref3 = Esil*(float)1.3; //condition3
```

```
        if ( ( (en[i]>E_ref1) & (ud[i]>D_ref1) ) | ( (en[i]>E_ref2) & (ud[i]>D_ref2) ) |
        ( (en[i]>E_ref3) ) ) {
                bin_boundary[i] = 1;
        }
        else bin_boundary[i] = 0;
}

unsigned long *be_frm, *en_frm, k_be = 0, k_en = 0;
be_frm = (unsigned long *)(bip_boundary+(NUM_FRAME-1));
en_frm = be_frm+NUM_DET;
for (i=0; i<NUM_FRAME-1; i++) {
        bip_boundary[i] = bin_boundary[i+1]-bin_boundary[i];
        if (bip_boundary[i] == 1) {
                be_frm[k_be] = i;
                k_be += 1;
        }
        if (bip_boundary[i] == -1) {
                en_frm[k_en] = i;
                k_en += 1;
        }
}

if (k_be > k_en) k_be -= 1;
if (k_en > k_be) {
        register unsigned long s = 0;
        while (s < k_en-1) {
                en_frm[s] = en_frm[s+1];
                s += 1;
        }
        k_en -= 1;
}

unsigned long numMFCC = 0;
if ( (k_be != 0) & (k_en != 0) ) {
        if ( k_be <= NUM_DET ) {
                register unsigned long s = k_be-1;
                while (s > 0) {
                        if (be_frm[s]-en_frm[s-1] == 1) {
                                en_frm[s-1] = en_frm[s];
                                if (s == (NUM_DET-2)) {
                                        be_frm[s] = be_frm[s+1];
                                        en_frm[s] = en_frm[s+1];
                                } else if (s == (NUM_DET-3)) {
                                        be_frm[s] = be_frm[s+1];
                                        en_frm[s] = en_frm[s+1];
                                        be_frm[s+1] = be_frm[s+2];
                                        en_frm[s+1] = en_frm[s+2];
                                } else if (s == (NUM_DET-4)) {
                                        be_frm[s] = be_frm[s+1];
                                        en_frm[s] = en_frm[s+1];
                                        be_frm[s+1] = be_frm[s+2];
                                        en_frm[s+1] = en_frm[s+2];
                                        be_frm[s+2] = be_frm[s+3];
                                        en_frm[s+2] = en_frm[s+3];
                                } else if (s == (NUM_DET-5)) {
                                        be_frm[s] = be_frm[s+1];
                                        en_frm[s] = en_frm[s+1];
```

```
                                        be_frm[s+1] = be_frm[s+2];
                                        en_frm[s+1] = en_frm[s+2];
                                        be_frm[s+2] = be_frm[s+3];
                                        en_frm[s+2] = en_frm[s+3];
                                        be_frm[s+3] = be_frm[s+4];
                                        en_frm[s+3] = en_frm[s+4];
                                }
                                k_be -= 1;
                                k_en -= 1;
                        }
                        s -= 1;
                }

                unsigned long p_be[k_be],p_en[k_en];
                for (i=0; i<k_be; i++) {
                        p_be[i] = ( (be_frm[i]+1)*NUM_FB-1 )+1;
                        p_en[i] =   (en_frm[i]+1)*NUM_FB-1;
                        register unsigned long j;
                        for (j=0; j<=p_en[i]-p_be[i]; j++) {
                                MFCC[numMFCC] = mfcc[p_be[i]+j];
                                numMFCC++;
                        }
                }
        }
    }
    return numMFCC;
}
/*-------------------------------------------------------------*/
```

ค.7 ฟังก์ชันปรับจำนวน MFCC ให้เท่ากัน (selection.h)

```
/*-------------------------------------------------------------*/
void size_norm(float *MFCC, unsigned long numMFCC, float *pre_out_f, float *pre_out_b)
{
    register unsigned long i;
    unsigned long tag = numMFCC/2, ovl = tag/2, len = tag+ovl;
    for (i=0; i<NORM2MFCC; i++) {
        pre_out_f[i] = MFCC[i];
        pre_out_b[(NORM2MFCC-1)-i] = MFCC[(numMFCC-1)-i];
        if (len < NORM2MFCC) {
            if (i >= len) {
                pre_out_f[i] = 0;
                pre_out_b[(NORM2MFCC-1)-i] = 0;
            }
        }
    }
}
/*-------------------------------------------------------------*/
```

ค.8 ฟังก์ชันการรู้จำของโครงข่ายประสาทเทียม (neural.h)

```
/*-------------------------------------------------------*/
void neural_net(float (*iw_f)[125],float (*iw_b)[125], float (*lw21_f)[25],float (*lw21_b)[25],
float (*lw32_f)[18],float (*lw32_b)[18], float (*lw43_f)[16],float (*lw43_b)[16],
float *b1_f,float *b1_b,float *b2_f,float *b2_b, float *b3_f,float *b3_b,float *b4_f,float *b4_b,
float *pre_out_f, float *pre_out_b, float *result)
{
	float *hid1_f; hid1_f = (float *)(XPAR_GENERIC_SDRAM_BASEADDR); //temp addr
	float *hid1_b; hid1_b = hid1_f+25;
	float *hid2_f; hid2_f = hid1_b+25;
	float *hid2_b; hid2_b = hid2_f+18;
	float *hid3_f; hid3_f = hid2_b+18;
	float *hid3_b; hid3_b = hid3_f+16;
	float *rst_f; rst_f = hid3_b+16;
	float *rst_b; rst_b = rst_f+7;
	float *tmp_mul_f; tmp_mul_f = rst_b+7;
	float *tmp_mul_b; tmp_mul_b = tmp_mul_f+NORM2MFCC;
	register unsigned short i,j;
	for (j=0; j<25; j++) {
		for (i=0; i<NORM2MFCC; i++) {
			tmp_mul_f[i] = pre_out_f[i]*iw_f[j][i];
			tmp_mul_b[i] = pre_out_b[i]*iw_b[j][i];
		}
		hid1_f[j] = 1/(1 + expf(-(sumf(NORM2MFCC,tmp_mul_f) + b1_f[j])));
		hid1_b[j] = 1/(1 + expf(-(sumf(NORM2MFCC,tmp_mul_b) + b1_b[j])));
	}
	for (j=0; j<18; j++) {
		for (i=0; i<25; i++) {
			tmp_mul_f[i] = hid1_f[i]*lw21_f[j][i];
			tmp_mul_b[i] = hid1_b[i]*lw21_b[j][i];
		}
		hid2_f[j] = 1/(1 + expf(-(sumf(25,tmp_mul_f) + b2_f[j])));
		hid2_b[j] = 1/(1 + expf(-(sumf(25,tmp_mul_b) + b2_b[j])));
	}
	for (j=0; j<16; j++) {
		for (i=0; i<18; i++) {
			tmp_mul_f[i] = hid2_f[i]*lw32_f[j][i];
			tmp_mul_b[i] = hid2_b[i]*lw32_b[j][i];
		}
		hid3_f[j] = 1/(1 + expf(-(sumf(18,tmp_mul_f) + b3_f[j])));
		hid3_b[j] = 1/(1 + expf(-(sumf(18,tmp_mul_b) + b3_b[j])));
	}
	for (j=0; j<7; j++) {
		for (i=0; i<16; i++) {
			tmp_mul_f[i] = hid3_f[i]*lw43_f[j][i];
			tmp_mul_b[i] = hid3_b[i]*lw43_b[j][i];
		}
		rst_f[j] = 1/(1 + expf(-(sumf(16,tmp_mul_f) + b4_f[j])));
		rst_b[j] = 1/(1 + expf(-(sumf(16,tmp_mul_b) + b4_b[j])));
		result[j] = rst_f[j]*rst_b[j];
	}
}
/*-------------------------------------------------------*/
```