ปีการศึกษา ๒๕๓๓

ระบบจำลองการทำงานของวงจรลอจิก

Logic Circuit Simulation System (LCSS)

โดย

นาย ประยุทธ เทพมังกร ๓๐๑๑๔๐

นาย ปราโมทย์ ภักดีศรีวงศ์ ๓๐๑๑๕๐

อาจารย์ที่ปรึกษา

รศ. มนัส สังวรศิลป์

อ. สุรพันธ์ เอื้อไพบูลย์

ปริญญานิพนธ์ปีการศึกษา ๒๕๓๓

ภาควิชา อิเลคทรอนิคส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง

เรื่อง ระบบจำลองการทำงานของวงจรลอจิก

ผู้จัดทำ

๑. นาย ประยุทธ เทพมังกร ๓๐๑๑๔๐

๒. นาย ปราโมทย์ ภักดิ์ศรีวงศ์ ๓๐๑๑๕๐

..... อาจารย์ที่ปรึกษา

(รศ. มนัส สังวรศิลป์)

..... อาจารย์ที่ปรึกษา

(อ. สุรพันธ์ เอื้อไพบูลย์)

ระบบจำลองการทำงานของวงจรลอจิก

Logic Circuit Simulation System (LCSS)

นาย ประยุทธ เทพมังกร ๓๐๑๑๔๐
นาย ปราโมทย์ ภักดิ์ศรีวงศ์ ๓๐๑๑๔๐
อาจารย์ที่ปรึกษา :
รศ. มนัส ลังวรศิลป์
อ. สุรพันธ์ เอื้อไพบูลย์
ปีการศึกษา ๒๕๓๓

บทคัดย่อ

ระบบจำลองการทำงานของวงจรลอจิก (*Logic Circuit Simulation System*) เป็นโปรแกรมที่ทำหน้าที่ จำลองการทำงานของวงจรทางลอจิก โดยการ ใช้เอดิเตอร์ (*editor*) ต่างๆเช่น sidekick, wordstar หรือจะเขียนด้วยเอดิเตอร์ภาษาต่างๆ (pascal, c, ฯลฯ) แทนส่วนของวงจรซึ่งประกอบด้วย อินพุท (*input*), เอ้าทพุท (*output*) และฟังก์ชันการทำงานของส่วนนั้น (AND,OR, NOR, NAND, ฯลฯ) เริ่มต้น โปรแกรมจะทำการโหลด (*load*) และแปลง (*translation*)ให้เป็นภาษาซี แล้วทำการลิงค์ (*link*) จนเป็นเอ็กซิคิวท์ ไฟล์ (*execute file*) จากนั้นจะนำสัญญาณอินพุทเข้ามาร่วมในการประมวลผลของวงจร จนได้เอ้าท์พุท ซึ่งจะนำไปแปลงให้เป็น แผนภูมิเวลา (*timing chart*) เพื่อความสะดวกในการวิเคราะห์การทำงานของวงจร

โปรแกรมที่เขียนขึ้น สามารถนำไปใช้จำลองการทำงานของอุปกรณ์ประเภทวงจรรวม (*IC, Integrated Circuit*) หรือ วงจรรวมย่อส่วนขนาดเล็ก (*VLSI, Very Large Scale Integrated Circuit*) เพื่อศึกษาการทำงานของอุปกรณ์เหล่านั้น หรือจะนำไปใช้จำลองการทำงานของวงจรจริงก็ได้

LOGIC CIRCUIT SIMULATION SYSTEM

(LCSS)

Mr. PRAYUT THEPMANGKORN 301140

Mr. PRAMOTE PAKSRIWONG 301150

Associated Professor

Mr. Manus Sangworasilp Advisor

Mr. Surapun Ua-piboon

1990.

Abstract

The Logic Circuit Simulation System (LCSS) is an intelligent computer program that can simulate the duty of logic circuit by using editors such as sidekick,wordstar or writing by editor of the computer language (pascal ,c, etc.) describe the circuit that include of input ,output point and working function of that part (or, nor, eor etc).

At the beginning , program have been loaded to translate file to C language and linked it to be the execute file. Input data file and program complied the circuit. After this the output is exist and will be translated to timing chart for easily analysis of the circuit.

This program can simulate the logic circuit such as Integrated Circuit (IC) or Vary Large Scale Integrate Circuit (VLSI) for study the working of this device.

สารบัญ

หน้า

สารบัญภาพ

สารบัญตาราง

หน้า

บทที่ ๑
บทนำ

ในช่วงระยะเวลาไม่กี่ปีมานี้ เทคโนโลยีทางด้านอิเลคทรอนิคส์เจริญขึ้นอย่างรวดเร็วจนแทบจะตามไม่ทัน การกำเนิดของไมโครโปรเซสเซอร์ ทำให้เกิดการเปลี่ยนแปลงครั้งยิ่งใหญ่ในวงการอิเลคทรอนิคส์ จากวงจรอิเลคทรอนิคส์ซึ่งเป็นฮาร์ดแวร์ มาเป็นวงจรไมโครโปรเซสเซอร์ซึ่งมีทั้งฮาร์ดแวร์ และซอฟต์แวร์ ปัจจุบันนี้เครื่องมืออิเลคทรอนิคส์ เกือบทุกชนิดจะมีไมโครโปรเซสเซอร์อยู่ภายใน

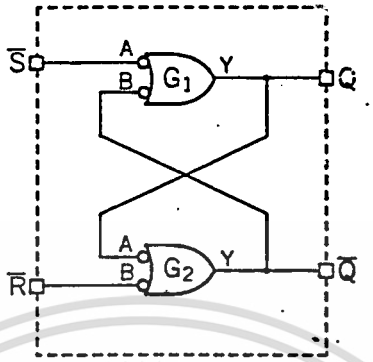
การพัฒนาเทคโนโลยีในการผลิตวงจรรวม (IC) ก็มีผลต่อวงการนักเล่นอิเลคทรอนิคส์เป็นอย่างมาก จากการประกอบวงจรด้วยทรานซิสเตอร์ และองค์ประกอบวงจรต่างๆ ก็เปลี่ยนมาเป็นการใช้วงจรรวมแทน และในที่สุดวงจรที่เรียกว่า ซับซ้อนมากอย่างเช่น นาฬิกา,ดิจิตอลมิเตอร์ ก็เหลือเพียง LSI เพียงชิพเดียวกับอุปกรณ์ประกอบเล็กๆ อีกเพียงไม่กี่ชิ้นเท่านั้น

การศึกษาทางอิเลคทรอนิคส์กำลังเปลี่ยนแปลงไป จากการเรียนรู้เทคนิคในการต่อวงจรด้วยอุปกรณ์เล็กๆ มาเป็นการเรียนรู้วิธีใช้วงจรรวม หรือ LSI ให้ถูกต้อง ซึ่งภายใน LSI ประกอบด้วย อุปกรณ์ต่างๆเป็นจำนวนมาก การที่จะสร้าง หรือศึกษาการทำงานของวงจร จำเป็นต้องนำอุปกรณ์มาทดลองต่อกันเองเป็นจำนวนมาก ทำให้เสียเวลาและสิ้นเปลืองโดยใช่เหตุ

จากเหตุผลต่างๆดังกล่าว จึงเกิดเป็นแรงดลใจในการคิดหาแนวทางที่จะเขียนโปรแกรมจำลองการทำงานทั้งหมดของวงจรต่างๆ โดยมุ่งเน้นไปที่วงจรทางลอจิก ซึ่งจะมีประโยชน์หลักในการจำลองการทำงานของ TTL และ CMOS

๑.๑ โปรแกรมอย่างง่าย

การเขียนโปรแกรมจำลองการทำงานของวงจรลอจิกอย่างง่ายสามารถทำได้โดยใช้ภาษาซี ต่อไปนี้จะเป็นตัวอย่างโปรแกรมที่เขียนขึ้นเพื่อจำลองการทำงานของ อาร์เอส ฟลิป ฟลอป โดยใช้ภาษาซีในการเขียน



ภาพที่ ๑ ก.วงจร อาร์เอส ฟลิบฟลอป

Time	S_	R_	Q	Q_
0	L	L	H	H
1	L	H	H	L
2	H	L	L	H
3	H	H	Q_n	Q_n

ข.ผลการทำงาน

```
/* RSFF.C */

#include <stdio.h>
#define  L  0
#define  H  1
struct _ND2 {
```

```
    char Y, Y__;
};

struct _RSFF {
    char Q, Q__;
    char Q_, Q___;
    struct _ND2 G1;
    struct _ND2 G2;
};

struct _RSFF RSFF_d;
int Now;

void main (void);
void RSFF_f (struct _RSFF *, char, char);
int RSFF_c (struct _RSFF *, char, char);
void ND2_f (struct _ND2 *, char, char);
int ND2_c (struct _ND2 *, char);

void main ()
{
    char S_, R_;
    Now = 0;
    printf ("Time   S_   R_   Q    Q_\n");
    while (scanf ("%d  %d", &S_, &R_) != EOF) {
        RSFF_f (&RSFF_d, S_, R_);
        printf ("%3d:   %d    %d    %d    %d\n",
                Now, S_, R_, RSFF_d.Q, RSFF_d.Q_);
```

```
        Now++;
    }
}

void RSFF_f (_p, S_, R_)
char S_, R_;
struct _RSFF *_p;
{
    if (! Now) {
        _p -> Q__ = H;
        _p -> Q___ = H;
    }
    do {
        ND2_f (&_p -> G1, S_, *(char *)&_p -> G2);
        ND2_f (&_p -> G2, R_, *(char *)&_p -> G1);
        _p -> Q = *(char *)&_p -> G1;
        _p -> Q_ = *(char *)&_p -> G2;
    } while (RSFF_c (_p, _p -> Q, _p ->Q_));
}

int RSFF_c (_p, Q, Q_)
struct _RSFF *_p;
char Q, Q_;
{
    int __f = 0;
    if (_p -> Q__ != Q) {
        _p -> Q__ = Q; __f = 1;
    }
```

```
    if (_p -> Q___ != Q_) {
        _p -> Q___ = Q_; __f = 1;
    }
    return (__f);
}

void ND2_f (_p, A, B)
struct _ND2 *_p;
char A, B;
{
    if (! Now) {
        _p -> Y__ = H;
    }
    do {
        _p -> Y = ~(A & B) & 1;
    } while (ND2_c (_p, _p -> Y));
}

int ND2_c (_p, Y)
struct _ND2 *_p;
char Y;
{
    int __f = 0;
    if (_p -> Y__ != Y) {
        _p -> Y__ = Y; __f = 1;
    }
    return (__f);
}
```

๑.๒ การทำงานของวงจรอย่างง่าย

จากโปรแกรม rsff.c สามารถอธิบายการทำงานของโปรแกรมได้ดังนี้

เริ่มต้นด้วยการกำหนดค่า L เท่ากับ 0 และ H เท่ากับ 1 โดยกำหนดโปรแกรมโครงสร้าง ๒ โปรแกรมคือ struct_ND2 และ struct_RSFF โดยที่ struct_ND2 มีตัวแปรเป็นตัวอักษร คือ Y, Y_ และให้ struct_RSFF มีตัวแปรเป็นตัวอักษรเช่นกันคือ Q, Q_, Q__, Q___ และสร้าง struct_ND2 ที่มีชื่อ G1, G2 แล้วกำหนด struct_RSFF ชื่อ RSFF_d และในโปรแกรมหลักมีโปรแกรมย่อยอีก ๔ โปรแกรมคือ RSFF_f, RSFF_c, ND2_f, ND2_c

ในโปรแกรมหลัก กำหนดตัวแปรเป็นตัวอักษรคือ S_, R_ และกำหนด Now เป็นเลขจำนวนเต็มมีค่าเท่ากับ 0 โดยให้นำค่าของ S_, R_ ไปเก็บไว้ในหน่วยความจำ และทำการเช็คว่าถ้ายังไม่สิ้นสุดไฟล์ก็ให้ทำต่อไปจนจบ ก่อนที่จะเข้าสู่โปรแกรมย่อย RSFF_f โดยมีพอยน์เตอร์ชี้ไปที่โปรแกรมย่อย RSFF_d และนำค่า S_, R_ เข้ามาแล้วเช็คว่าค่า Now เท่ากับ 0 หรือไม่ ถ้าไม่ก็ให้พอยน์เตอร์ ชี้ไปที่ Q__, Q___ และให้เท่ากับ 1 จากนั้นจึงเข้าสู่โปรแกรมย่อย ND2_f โดยจะทำหน้าที่เปลี่ยนค่า Y ให้เท่ากับ ~(A & B) & 1 โดยครั้งแรก ให้ A เท่ากับ S_ และ B เท่ากับ ค่าของ G1 แล้วผลลัพธ์เก็บไว้ที่ G2 ส่วนครั้งที่สอง ให้ A เท่ากับ R_ และ B เท่ากับ ค่าของ G2 แล้วผลลัพธ์ไปเก็บไว้ที่ G1 ส่วนโปรแกรมย่อย ND2_c นั้นทำหน้าที่ แปลงค่า Y__ ให้เท่ากับค่า Y จากนั้นก็จะกลับสู่โปรแกรมย่อย RSFF_f อีก แล้วทำการถ่ายค่าผลลัพธ์ G1 ให้กับค่า Q และ ค่าผลลัพธ์ G2 ให้กับค่า Q__ ส่วนโปรแกรมย่อย RSFF_c นั้นทำหน้าที่ เช็คค่า Q__ ให้เท่ากับ ค่าของ Q และ ค่า Q___ ให้มีค่าเท่ากับ ค่า Q_ จากนั้นก็จะทำการพิมพ์ค่าของ Now, S_, R_, RSFF_d.Q, RSFF_d.Q_ โดยกำหนดการพิมพ์ด้วยเลขฐานสิบ หลังจากนั้นจะเพิ่มค่า Now ขึ้น 1 แล้วกลับไปนำค่าของ S_ และ R_ เข้ามาอีกจนกว่าจะพบสัญญาณสิ้นสุดไฟล์แล้วให้ออกจากลูปเป็นการสิ้นสุดโปรแกรม

เมื่อทำการ รัน โดยป้อนข้อมูลแบบต่างๆ จะได้ผลตามตารางในหน้าถัดไป

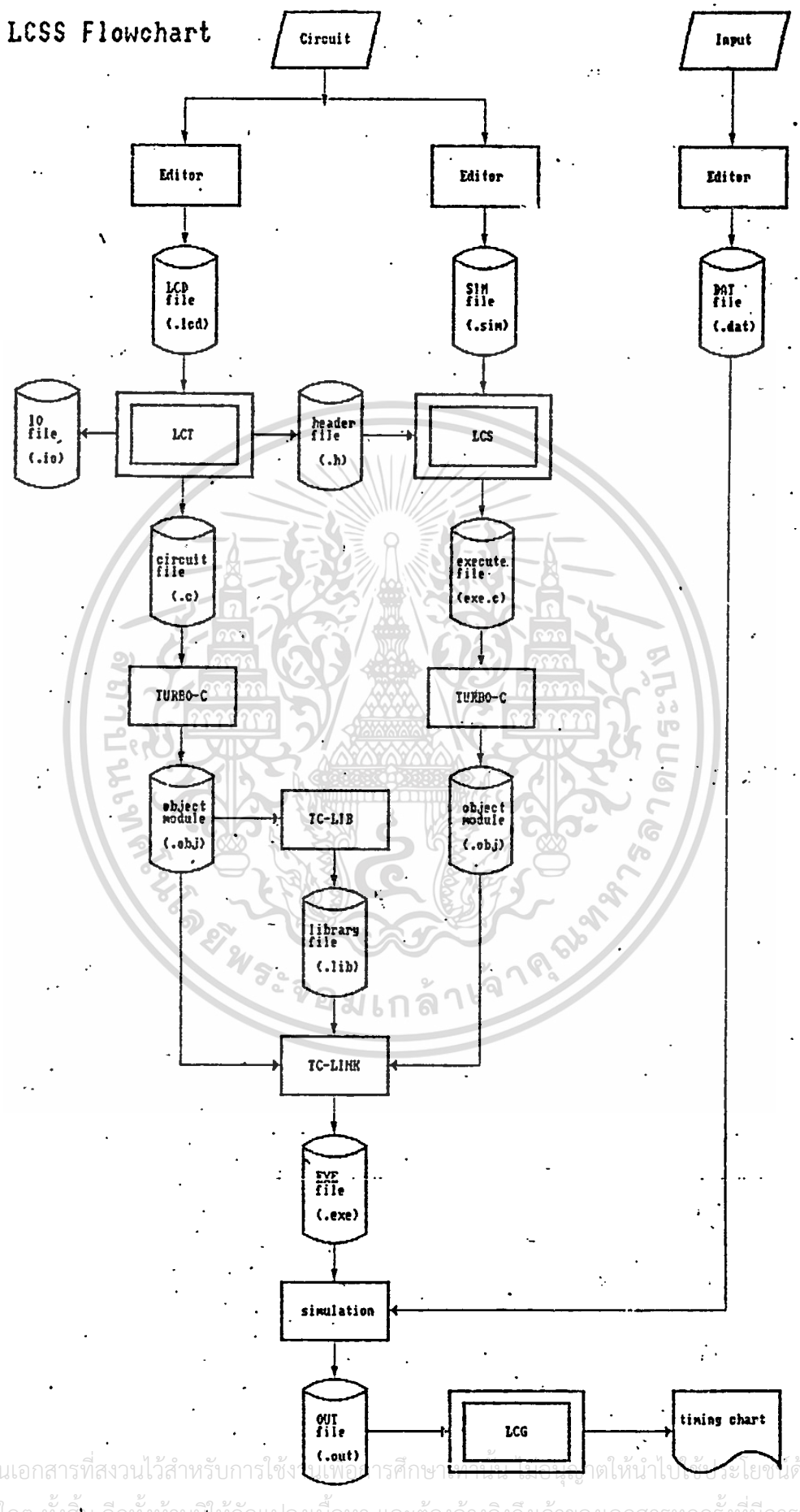
/*RSFF.OUT*/

Time	S_	R_	Q	Q_
0	0	0	1	1
1	0	1	1	0
2	1	0	0	1
3	1	1	0	1

ภาพที่ ๑-๒ ผลการทำงานของโปรแกรมจำลองการทำงานของ อาร์เอส ฟลิบฟลอป

ในการใช้งานจริงจะต้องทำการ ลิงค์ โปรแกรมให้เป็นเอ็กซีคิวท์ ไฟล์ เพื่อ ให้สามารถ รัน ได้โดยไม่ต้องเข้าภาษาซีก่อน ทำให้ไฟล์ที่ได้มีขนาดใหญ่มากทั้งที่สามารถจำลองการทำงานของอุปกรณ์เพียงตัวเดียว ในกรณีที่ต้องการจำลองการทำงานของอุปกรณ์ตัวอื่นๆด้วย จะต้องเขียนโปรแกรมเพิ่มต่างหาก ทำให้ยุ่งยากในทางปฏิบัติ

โปรแกรมที่เขียนขึ้นจริงมีลักษณะเป็นโมดูล กล่าวคือจะแบ่งเป็นหลายส่วนย่อยๆ แต่ละส่วนจะทำหน้าที่เฉพาะตัวต่างกัน ซึ่งจะมีประโยชน์ในการขยายขอบเขตการทำงาน โดยเขียนเฉพาะส่วนที่เพิ่มเข้าไปเท่านั้น ไม่ต้องเขียนใหม่ทั้งหมดดังจะได้กล่าวในรายละเอียดต่อไป



ภาพที่ ๒-๑ บล็อกไดอะแกรมของระบบจำลองการทำงานของวงจรลอจิก

บทที่ ๒

หลักการทำงาน

การทำงานของโปรแกรมสามารถอธิบายได้ด้วยผังโปรแกรมดังภาพที่ ๒-๑ จากภาพจะเห็นได้ว่าในการใช้งานต้องเขียนไฟล์ขึ้นมา ๓ ไฟล์ ด้วยกัน คือ LCD ไฟล์, SIM ไฟล์, และ DAT ไฟล์ เมื่อเริ่มใช้งานโปรแกรมจะโหลด *(load)* LCD ไฟล์ และ SIM ไฟล์ ขึ้นมา แล้วทำการแปลง *(translation)* ให้เป็นภาษาซี โดยใช้ส่วนที่เรียกว่า LCT *(LCD to C Translator)* ในการแปลง LCD ไฟล์ และ LCS *(LCD Simulation Code Generator)* ในการแปลง SIM ไฟล์ ขั้นต่อไป เทอร์โบ ซี *(Turbo C)* จะเปลี่ยนให้เป็นออบเจ็ค ไฟล์ *(.obj)* แล้วจึงทำการลิงค์ *(link)* ออบเจ็ค ไฟล์ ทั้งสองเข้าด้วยกันจนได้เป็นเอ็กซิคิวท์ ไฟล์ *(.exe)* จากนั้นจะนำสัญญาณที่ใช้ทดสอบการทำงานของวงจร คือ DAT ไฟล์ เข้ามาร่วมในการประมวลผลของวงจร จนได้ เอ้าท์พุท ซึ่งยังไม่สะดวกในการวิเคราะห์ เนื่องจากอยู่ในรูปของไบนารี่โค้ด *(binary code)* ในขั้นตอนสุดท้ายส่วนที่เรียกว่า LCG *(LCD Graphic Utility)* จะนำไปแปลงให้เป็น แผนภูมิเวลา*(timing chart)* และทำการพิมพ์ออกมาเพื่อความสะดวกในการวิเคราะห์การทำงานของวงจร

ซึ่งในแต่ละไฟล์มีโครงสร้างและรายละเอียดดังต่อไปนี้

๒.๑ โครงสร้างของ LCD ไฟล์

LCD ไฟล์ *(Logic Circuit Description File)* เป็นไฟล์ที่เขียนบรรยายส่วนประกอบของวงจร ซึ่งจะกล่าวถึง ชื่อวงจร, จุดต่อสัญญาณทางเข้า - ออก, และฟังก์ชันการทำงานภายในวงจร โดยที่มีโครงสร้างดังนี้

NAME	xxxx;	ชื่อวงจร
INPUT	xxxx;	จุดต่อสัญญาณทางเข้า
OUTPUT	xxxx;	จุดต่อสัญญาณทางออก
DEFINE		กำหนดฟังก์ชั่นที่ใช้ในโปรแกรม

```
{
    EXT   xx;              ชื่อโปรแกรมย่อยภายนอกที่เรียกใช้
    LOC   xx;              ชื่ออุปกรณ์ภายในที่เรียกใช้โปรแกรม
                           ย่อยภายนอก
    U1 = FUNC (x,x);       ฟังก์ชั่นของอุปกรณ์ภายใน
    G1 = A & B;
       :
       :

}
END;                       สิ้นสุดไฟล์
```

๒.๒ โครงสร้างของ SIM ไฟล์

SIM ไฟล์ (*Simulation File*) เป็นไฟล์ที่เขียนขึ้นเพื่อกำหนดจุดทางเข้า - ออก ของสัญญาณที่ใช้ทดสอบ โดยที่มีโครงสร้างดังนี้

```
TEST      xxxx;        ชื่อวงจร
INPUT     xxxx;        จุดต่อสัญญาณทางเข้า
OUTPUT    xxxx;        จุดต่อสัญญาณทางออก
TESTIN    xxxx;        จุดทางเข้าของสัญญาณที่ใช้ทดสอบ
TESTOUT   xxxx;        จุดทางออกของสัญญาณไปแสดงผล
END;                   สิ้นสุดไฟล์
```

๒.๓ โครงสร้างของ DAT ไฟล์

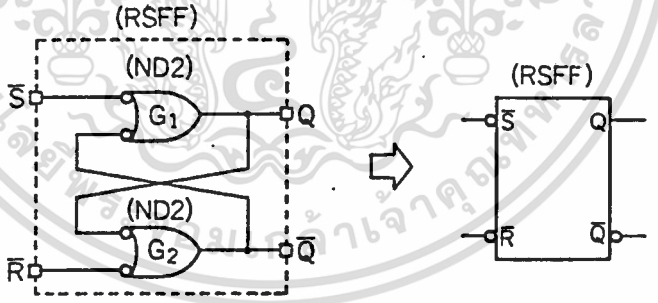
DAT ไฟล์ (*Data File*) เป็นไฟล์ที่เขียนขึ้นเพื่อกำหนดรูปแบบของสัญญาณที่ใช้ทดสอบ โดยที่มีโครงสร้างดังนี้

```
PATTERN                รูปแบบของข้อมูลที่ใช้ทดสอบ
```

```
{
0000 : x x                    ค่าของข้อมูลชุดที่หนึ่ง
0010 : x x                    ค่าของข้อมูลชุดที่สอง
     :
     :

}
END;                          จบชุดข้อมูล
```

จากที่กล่าวมาข้างต้น อาจไม่กระจ่างเท่าที่ควร ต่อไปจะยกตัวอย่างการจำลองการทำงานของวงจร อาร์เอส ฟลิบฟลอป เพื่อเปรียบเทียบกับการใช้ภาษาซีในตัวอย่างแรก



ภาพที่ ๒-๒ ก.วงจร อาร์เอส ฟลิบฟลอป ข.วงจรสมมูลย์

จากภาพเป็นวงจร RS ฟลิปฟลอป ซึ่งภายในประกอบด้วย NAND เกท จำนวนสองตัว คือ G_1 และ G_2 โดยมี S_ และ R_ เป็นจุดต่อสัญญาณทางเข้า ส่วน Q และ Q_ เป็นจุดต่อสัญญาณทางออก โดยที่ ND2 *(NAND gate 2 input)* เป็นฟังก์ชั่นที่เขียนไว้ในโปรแกรมย่อยภายนอก ซึ่งเรียกใช้โดย G1 และ G2

และจากวงจรสมมูลย์ เราจะทำการป้อนสัญญาณที่ใช้ทดสอบเข้าทาง S_ , R_ และนำสัญญาณจากจุดทั้งสี่คือ S_. R_, Q และ Q_ ไปแสดงผล โดยการกำหนดรูปแบบสัญญาณที่ใช้ทดสอบแบบต่าง ๆ ขึ้นมา ๔ แบบ จะเขียนไฟล์ทั้งสามได้ดังนี้

/*RSFF.LCD*/

```
NAME     RSFF;
INPUT    S_, R_;
OUTPUT   Q, Q_;
DEFINE
{
     EXT  ND2;
     LOC  G1, G2;

     G1 = ND2 (S_, G2);
     G2 = ND2 (R_, G1);
     Q  = G1;
     Q_ = G2;
}
END;
```

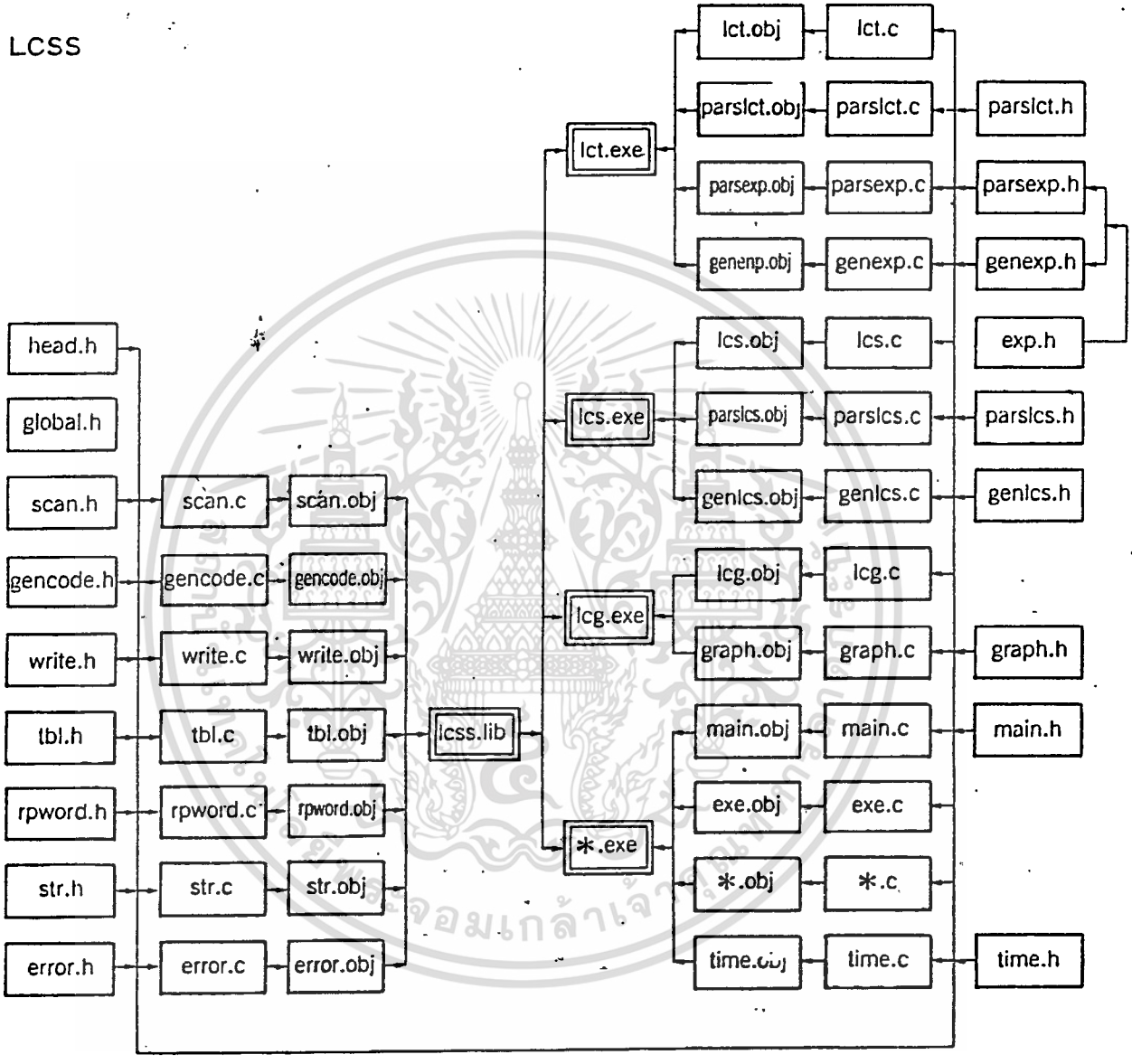
/*RSFF.SIM*/

```
1: TEST     RSFF;
2: TESTIN   R_, S_;
3: TESTOUT  R_, S_, Q, Q_;
4: END;
```

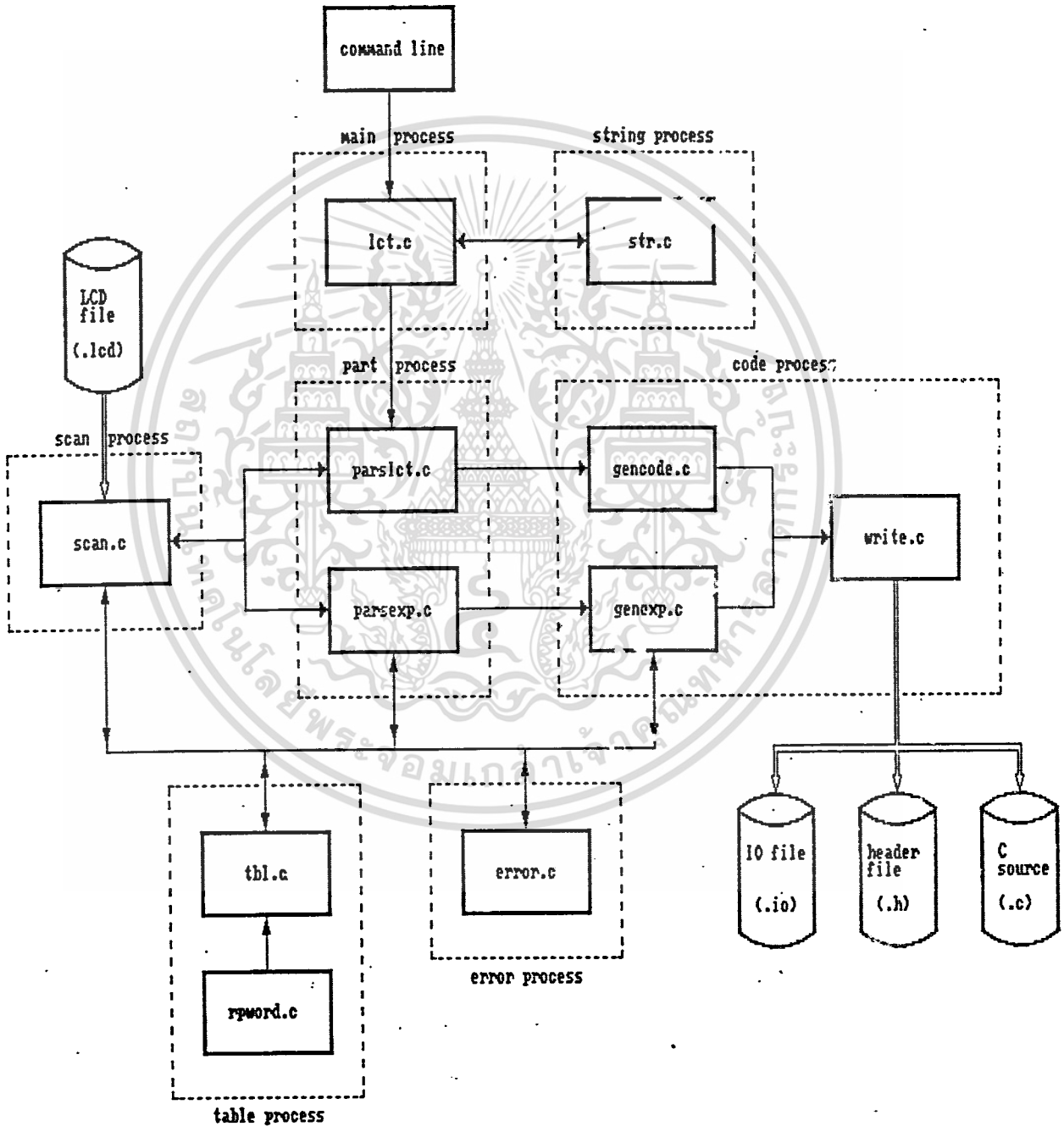
/*RSFF.DAT*/

```
PATTERN
{
/* S_  R_ */
:  0   0
:  0   1
:  1   0
:  1   1
:  0   0
}
END;
```

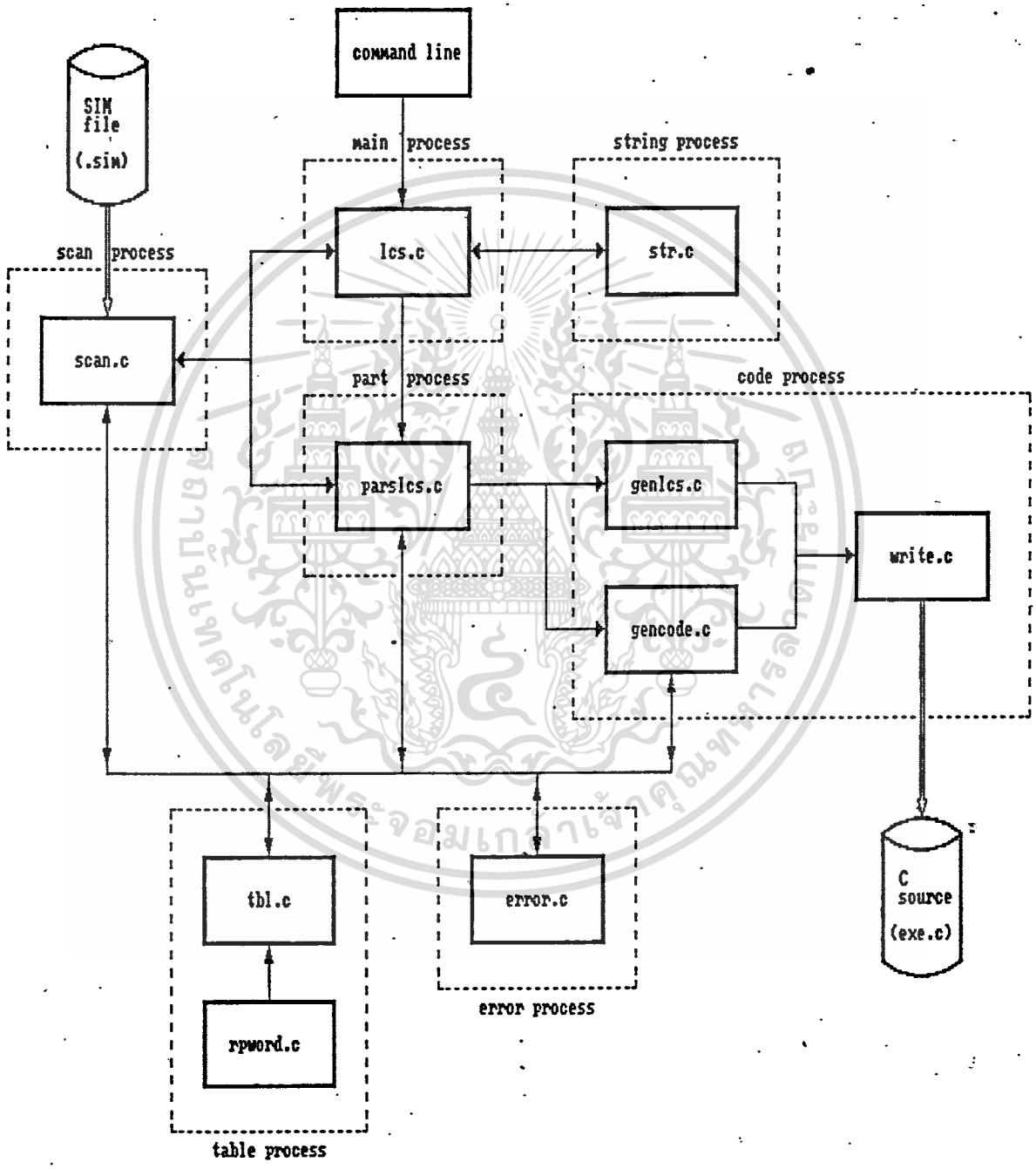
จากที่กล่าวมาข้างต้น จะเห็นได้ว่า จากเดิมที่เคยใช้ภาษาซีในการเขียนโปรแกรมจำลองการทำงานของวงจร ซึ่งมีความยาวมาก และยังเข้าใจยากด้วย สามารถนำมาเขียนในรูปแบบที่ง่ายต่อการตรวจสอบ แก้ไข หรือเพิ่มเติมบางส่วนเข้าไป นอกจากนั้นยังมีขนาดเล็กกว่าอีกด้วย



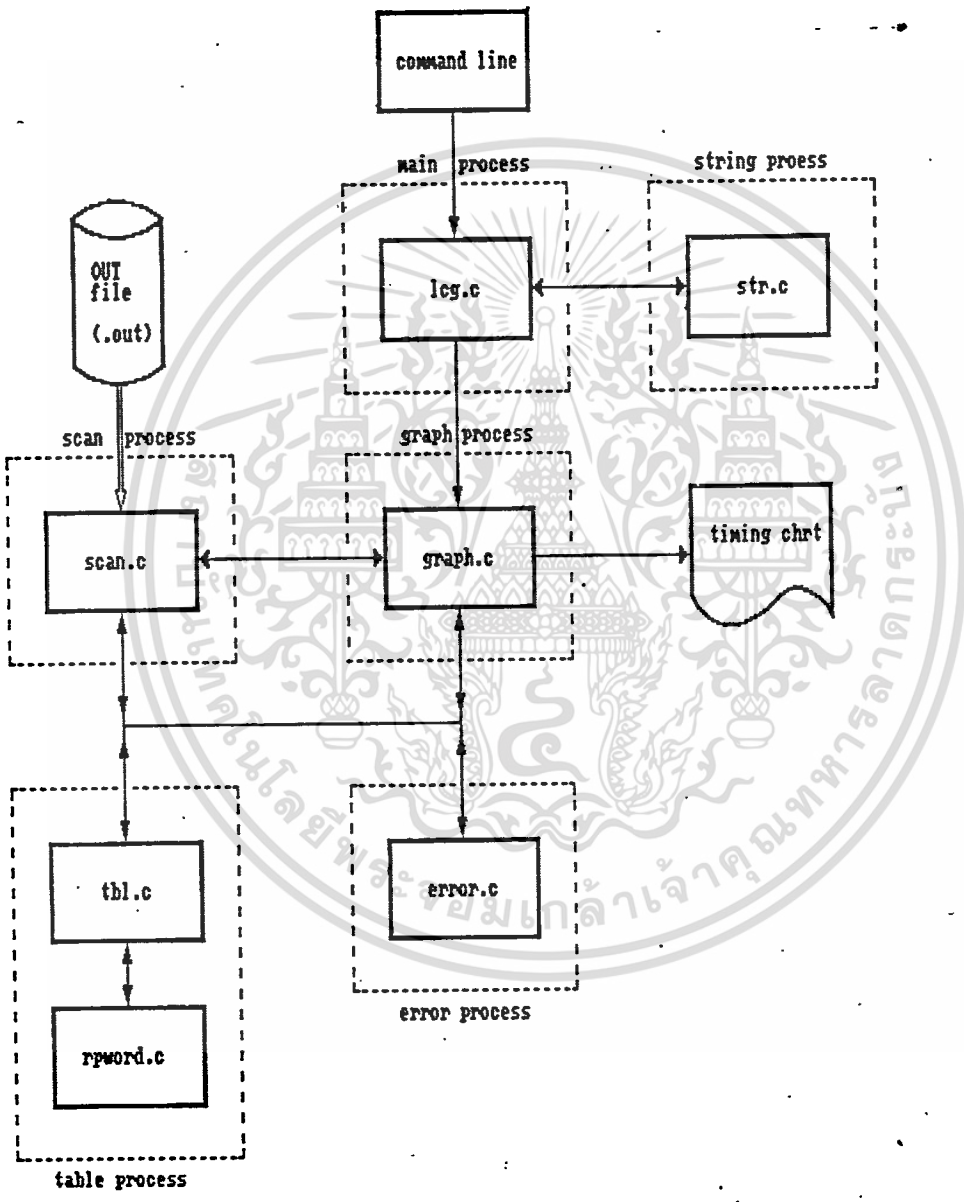
ภาพที่ ๓-๑ บล็อกไดอะแกรมโครงสร้างของระบบ



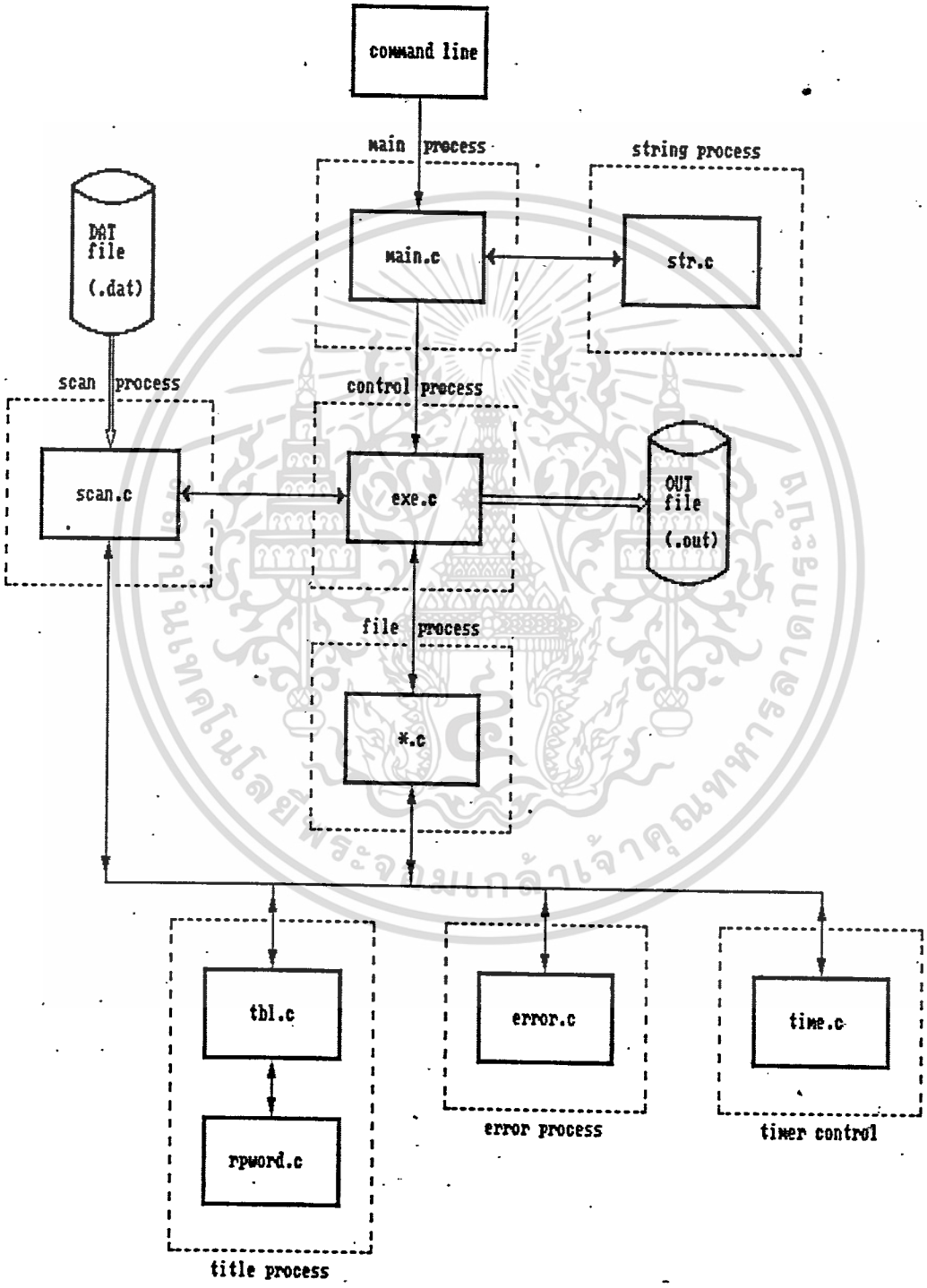
ภาพที่ ๓-๒ บล็อกไดอะแกรมการทำงานของ LCT



ภาพที่ ๓-๓ บล็อกไดอะแกรมการทำงานของ LCS



ภาพที่ ๓-๔ บล็อกไดอะแกรมการทำงานของ EXE



ภาพที่ ๓-๔ บล็อกไดอะแกรมการทำงานของ EXE

บทที่ ๓
โครงสร้างของระบบ

ในบทนี้จะได้กล่าวถึงโครงสร้างของระบบ ซึ่งประกอบด้วยส่วนหลัก ๓ ส่วนด้วยกันคือ LCT, LCS และ LCG ดังแสดงในภาพที่ ๓-๑ . รวมทั้งการทำงานอย่างคร่าว ๆ ของแต่ละส่วนดังนี้

๓.๑ LCT (*LCD to C translator*)

ส่วนนี้มีหน้าที่หลักในการแปลง LCD ไฟล์ ที่เขียนขึ้นเพื่อบรรยายส่วนประกอบของวงจร โดยใช้เอดิเตอร์ต่าง ๆ ให้เป็นภาษาซี ขั้นตอนการทำงานของ LCT แสดงไว้ในภาพที่ ๓-๒ สามารถอธิบายได้คร่าว ๆ ดังต่อไปนี้

จาก LCD ไฟล์ ที่เขียนไว้ สแกน โปรเซส (scan process) จะทำการอ่านข้อมูล แล้วส่ง ต่อไปให้ส่วน พาร์ท โปรเซส (part process) และ เทเบิล โปรเซส (table process)ซึ่งมีการตรวจจับความผิดพลาดด้วยส่วน เออร์เรอร์ โปรเซส (error process) ทำการตีความหมาย จากนั้นจะถูกส่งต่อไปยังส่วน โค้ดโปรเซส (code process)เพื่อสร้างเป็น อินพุทเอาท์พุทไฟล์ (io file), เฮดเดอร์ไฟล์ (header file) และ ซอร์ส ไฟล์ (source file) (*.c)

๓.๒ LCS (*LCD Simulation Code Generator*)

ส่วนนี้มีหน้าที่หลักในการแปลง SIM ไฟล์ ที่เขียนขึ้นเพื่อบรรยายส่วนประกอบของวงจร ซึ่งจะกล่าวถึง ชื่อวงจร, จุดต่อสัญญาณทางเข้า - ออก, และฟังก์ชั่นการทำงานภายในวงจร โดยใช้เอดิเตอร์ต่าง ๆ ให้เป็นภาษาซี ขั้นตอนการทำงานของ LCS แสดงไว้ในภาพที่ ๓-๓ สามารถอธิบายได้คร่าว ๆ ดังต่อไปนี้

จาก SIM ไฟล์ ที่เขียนไว้ กระบวนการสแกน จะทำการอ่านข้อมูล แล้วส่งต่อไปให้ส่วน พาร์ท โปรเซส และ เทเบิล โปรเซส ซึ่งมีการตรวจจับความผิดพลาดด้วยส่วน เออร์เรอร์ โปรเซส ทำการตีความหมาย จากนั้นจะถูก

ส่งต่อไปยังส่วน โค้ด โปรเซส เพื่อสร้างเป็น ซอร์ท ไฟล์อีกอันหนึ่ง (exe.c)

๓.๓ LCG (*LCD Graphic Utility*)

ส่วนนี้มีหน้าที่หลักในการแปลง OUT ไฟล์ที่ได้จากการนำสัญญาณอินพุทเข้ามาร่วมในการประมวลผลของวงจร จนได้เอ้าท์พุท ซึ่งจะนำไปแปลงให้เป็นแผนภูมิเวลา *(timing chart)* เพื่อความสะดวกในการวิเคราะห์การทำงานของวงจรขั้นตอนการทำงานของ LCG แสดงไว้ในภาพที่ ๓-๔ สามารถอธิบายได้คร่าว ๆ ดังต่อไปนี้

จาก OUT ไฟล์ ที่ได้ สแกน โปรเซส จะทำการอ่านข้อมูล แล้วส่งต่อไปให้กับส่วน เทเบิล โปรเซส ซึ่งมีการตรวจจับความผิดพลาดด้วยส่วน เออร์เรอร์ โปรเซส ทำการตีความหมาย จากนั้นจะถูกส่งต่อไปยังส่วน กราฟ โปรเซส เพื่อสร้างเป็น แผนภูมิเวลา *(timing chart)*

ยังมีอีกส่วนที่ไม่ได้กล่าวไว้ในส่วนหลัก ๓ ส่วน แต่มีความสำคัญมาก คือ การคอมไพล์ ซอร์ท ไฟล์ทั้งสอง (*.c และ exe.c) และทำการลิงค์ ออพ-เจ็ค ไฟล์ทั้งสองเข้าด้วยกันเป็น เอ็กซิคิวท์ ไฟล์ ซึ่งสามารถนำไปประมวลผลการทำงานโดยป้อนอินพุทแบบต่าง ๆ เข้าไป จะขอเรียกส่วนนี้ว่า EXE โดยมีขั้นตอนการทำงานของ แสดงไว้ในภาพที่ ๓-๕ สามารถอธิบายได้คร่าว ๆ ดังต่อไปนี้

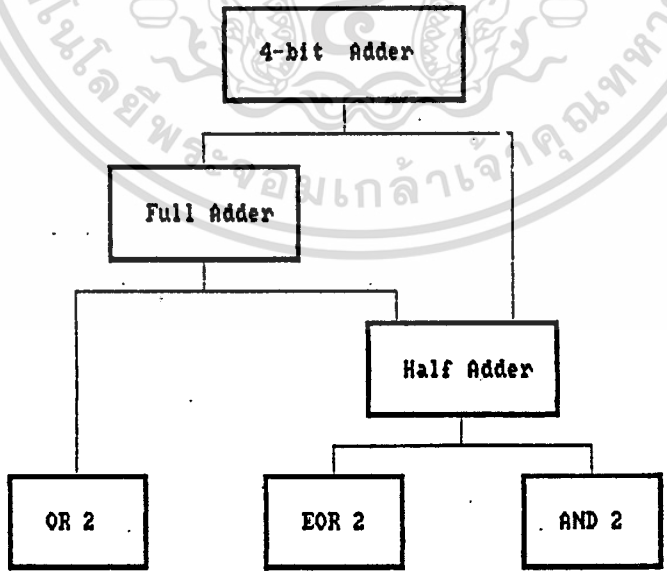
จาก DAT ไฟล์ ที่เขียนไว้ สแกน โปรเซส จะทำการอ่านข้อมูล แล้วส่งต่อไปให้ส่วน เทเบิล โปรเซส ซึ่งมีการตรวจจับความ ผิดพลาดด้วยส่วน เออร์เรอร์ โปรแกรม เช่นเดียวกับ LCT และ LCS แต่จะมีส่วน ไทเมอร์ คอนโทรล เพิ่มขึ้นมา เนื่องจากในการประมวลผล จะมีสัญญาณนาฬิกา (clock) เข้ามาเกี่ยวข้องด้วย ขณะเดียวกัน สแกน โปรเซส จะส่งข้อมูลให้กับ คอนโทรล โปรเซส ซึ่งทำงานร่วมกับ ไฟล์ โปรเซส (file process) โดยถูก เมน โปรเซส (main process) ควบคุมอยู่ จากนั้นจะได้ผลการประมวลผลสัญญาณออกมาเป็น เอาท์พุท ไฟล์

ในบทหน้าจะได้กล่าวถึงการขยายขอบเขต การทำงานของโปรแกรมออกไปอีก สำหรับวงจรที่มีความซับซ้อนมากขึ้น

บทที่ ๔

การขยายขอบเขตการทำงาน

ในบทนี้จะกล่าวถึงการทำงานในลักษณะเป็นโมดูล *(module)* กล่าวคือ จะแบ่งการทำงานของวงจรออกเป็นส่วนย่อยหลาย ๆ ส่วนประกอบกัน ซึ่งมีประโยชน์ในการจำลองการทำงานของวงจรที่มีขนาดใหญ่ ตัวอย่างที่จะกล่าวถึงต่อไปนี้เป็นการทำงานของวงจรบวกเลขสี่หลัก ซึ่งสามารถแบ่งออกเป็น ๒ ส่วนใหญ่ ๆ คือ วงจรบวกเลขแบบมีตัวทดเข้า *(Full Adder)* และ วงจรบวกเลขแบบไม่มีตัวทดเข้า *(Half Adder)* โดยภายในวงจรบวกเลขแบบไม่มีตัวทดเข้า จะใช้ฟังก์ชั่น AND และ EOR ส่วนในวงจรบวกเลขแบบมีตัวทดเข้า จะประกอบด้วย วงจรบวกเลขแบบไม่มีตัวทดเข้า และ ใช้ฟังก์ชั่น OR โดยมีผังโปรแกรมดังนี้



ภาพที่ ๔-๑ บล็อกไดอะแกรมของวงจรบวกเลขฐานสองสี่หลัก

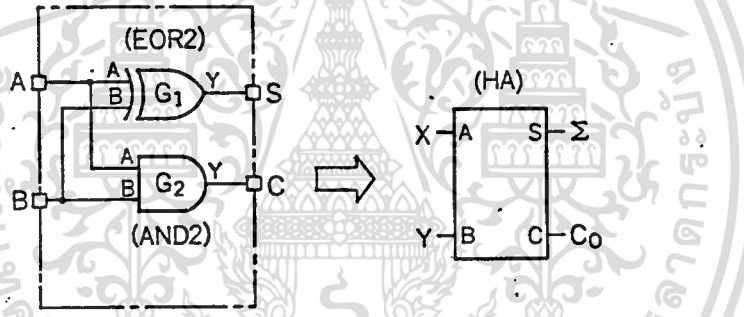
เริ่มต้นจะทำการกำหนดฟังก์ชั่นต่าง ๆ ของ เกท (gate) ที่ใช้ คือ OR, AND และ EOR (ซึ่งจะถูกเรียกใช้โดยอุปกรณ์ภายใน) รวมกันไว้ภายในไฟล์ที่ชื่อ GATE1.LCD ดังนี้

```
/*GATE1.LCD*/

1:  NAME      OR2;
2:  INPUT     A, B;
3:  OUTPUT    Y;
4:  DEFINE
5:  {
6:        Y     = A # B;
7:  }
8:  END;
9:
10: NAME      AND2;
11: INPUT     A, B;
12: OUTPUT    Y;
13: DEFINE
14: {
15:       Y     = A & B;
16: }
17: END;
18:
19: NAME      EOR2;
20: INPUT     A, B;
21: OUTPUT    Y;
22: DEFINE
```

```
23: {
24:       Y     = A ^ B;
25: }
26: END;
```

ภายในวงจรบวกเลขแบบไม่มีตัวทดเข้า ประกอบด้วย AND เกท และ EOR เกท อย่างละ ๑ ตัว ซึ่งมีวงจรดังภาพที่ ๔-๒ ก. และวงจรสมมูลย์ ใน ภาพ ข.



ภาพที่ ๔-๒ ก. วงจรบวกเลขฐานสองแบบไม่คิดตัวทดเข้า ข. วงจรสมมูลย์

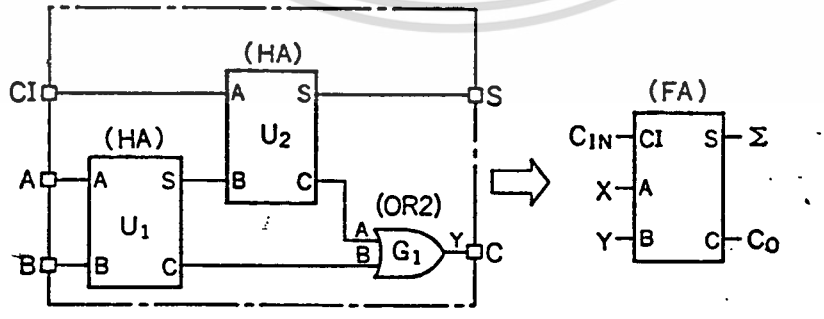
จากวงจรในภาพข้างต้น สามารถสร้าง LCD ไฟล์ ชื่อ HA.LCD แทนตัววงจร โดยมี A, B เป็นจุดต่อทางเข้าของสัญญาณที่นำมาบวกกัน และ S เป็นจุดต่อทางออกของสัญญาณที่ได้จากการบวก ส่วน C เป็นจุดต่อทางออกของตัวทดที่ได้จากการบวก ดังนี้

```
/*HA.LCD*/

1:  NAME    HA;
2:  INPUT   A, B;
```

```
3:  OUTPUT  C, S;
4:  DEFINE
5:  {
6:      EXT EOR2, AND2;
7:      LOC G1, G2;
8:
9:      G1  = EOR2 (A, B);
10:     G2  = AND2 (A, B);
11:     S   = G1;
12:     C   = G2;
13: }
14: END;
```

ภายในวงจรบวกเลขแบบมีตัวทดเข้า ประกอบด้วย วงจรบวกเลขแบบไม่มีตัวทดเข้า ๒ วงจร และ OR เกท ๑ ตัว ซึ่งมีวงจรดังภาพที่ ๔-๓ ก. และวงจรสมมูลย์ ในภาพ ข.



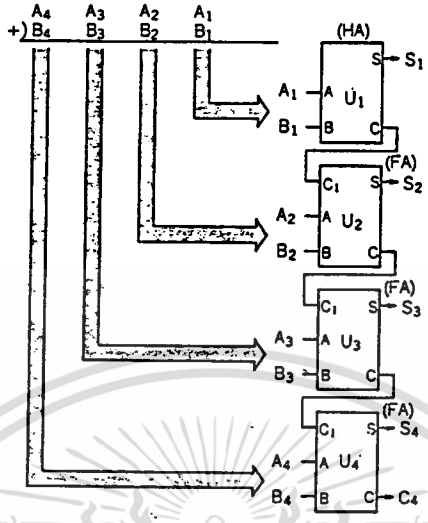
ภาพที่ ๔-๓ ก. วงจรบวกเลขฐานสองแบบคิดตัวทดเข้า ข. วงจรสมมูลย์

จากวงจรในภาพข้างต้น สามารถสร้าง LCD ไฟล์ ชื่อ FA.LCD แทนตัววงจร โดยมี A, B เป็นจุดต่อทางเข้าของสัญญาณที่นำมาบวกกัน C, แทนจุดต่อตัวทดเข้าจากหลักที่อยู่ต่ำกว่า และ S เป็นจุดต่อทางออกของสัญญาณที่ได้จากการบวก ส่วน C เป็นจุดต่อทางออกของตัวทดที่ได้จากการบวก ดังนี้

```
/*FA.LCD*/

1:  NAME    FA;
2:  INPUT   C1, A, B;
3:  OUTPUT  C, S;
4:  DEFINE
5:  {
6:      EXT EOR2, HA;
7:      LOC G1, U1, U2;
8:
9:      G1  = OR2 (U1.C, U2.C);
10:     U1  = HA (A, B);
11:     U2  = HA (C1, U1.S);
12:     S   = U2.S;
13:     C   = G1;
14: }
15: END;
```

สุดท้าย ทำการรวมวงจรบวกเลขแบบไม่มีตัวทดเข้า และวงจรบวกเลขแบบไม่มีตัวทดเข้า โดยให้หลักแรกบวกแบบไม่มีตัวทด ส่วนหลักที่อยู่ถัดขึ้นมาบวกแบบคิดตัวทดจากหลักที่อยู่ก่อนหน้า โดยต่อตัวทดออกของหลักที่อยู่ต่ำกว่า ไปเข้าที่จุดต่อตัวทดเข้าของหลักที่อยู่สูงกว่าถัดไปเรื่อย ดังภาพที่ ๔-๔



ภาพที่ ๔-๔ วงจรสมบูรณ์ของวงจรบวกเลขฐานสองสี่หลัก

จากวงจรในภาพข้างต้น สามารถสร้าง LCD ไฟล์ ชื่อ FA.LCD แทนตัววงจร โดยมี A_1, A_2, A_3, A_4, B_1, B_2, B_3, B_4 เป็นจุดต่อทางเข้าของสัญญาณที่นำมาบวกกัน C_4 แทนจุดต่อตัวทดออกที่ได้จากการบวกทั้งสี่หลัก และ S_1 เป็นจุดต่อทางออกของสัญญาณที่ได้จากการบวกทั้งสี่หลัก ดังนี้

```
/*PA4.LCD*/

1:  NAME     PA4;
2:  INPUT    A1, A2, A3, A4, B1, B2, B3, B4;
3:  OUTPUT   S1, S2, S3, S4, C4;
4:  DEFINE
5:  {
6:       EXT  HA, FA;
7:  LOC  U1, U2, U3, U4;
```

```
8:
9:        U1   = HA (A1, B1);
10:       U2   = FA (U1.C, A2, B2);
11:       U3   = FA (U2.C, A3, B3);
12:       U4   = FA (U3.C, A4, B4);
13:       S1   = U1.S;
14:       S2   = U2.S;
15:       S3   = U3.S;
16:       S4   = U4.S;
17:       C4   = U4.C;
18: }
19: END;
```

บทที่ ๔

ผลการทำงาน

การทำงานของระบบเริ่มด้วย การใช้คำสั่ง LCT ตามด้วยชื่อของ LCD ไฟล์และถ้ามีการเรียกใช้โปรแกรมย่อยภายนอก จะต้องตามด้วยชื่อไฟล์ที่โปรแกรมย่อยนั้นอยู่ โดยมีรูปแบบดังนี้

```
LCT ( *.LCD ) ( LIB.LCD )
```

เมื่อใช้ LCTแปลง RSFF.LCD ที่เขียนขึ้นเพื่อจำลองการทำงานของวงจรอาร์เอส ฟลิปฟลอบโดยใช้คำสั่ง LCT RSFF G จะได้ เอาท์พุท ไฟล์, เฮดเดอร์ ไฟล์และ SOURCE ไฟล์ ดังนี้

```
/* RSFF.H */

struct _AND2 {
char Y, Y__;
char A__, B__;
long delay;
};

void AND2_f (struct _AND2 *,  char,  char);

struct _ND2 {
char Y, Y__;
char A__, B__;
long delay;
};
```

```
void ND2_f (struct _ND2 *,  char,  char);

struct _OR2 {
char Y, Y__;
char A__, B__;
long delay;
};

void OR2_f (struct _OR2 *,  char,  char);

struct _NOR2 {
char Y, Y__;
char A__, B__;
long delay;
};

void NOR2_f (struct _NOR2 *,  char,  char);

struct _EOR2 {
char Y, Y__;
char A__, B__;
long delay;
};

void EOR2_f (struct _EOR2 *,  char,  char);
struct _RSFF {
char Q, Q__, Q_, Q___;
char S___, R___;
```

```
long delay;
struct _ND2 G1;
struct _ND2 G2;
};

void RSFF_f (struct _RSFF *,  char,  char);
```

```
    /* RSFF.IO */

AND2 (A, B)
       -> Y
ND2 (A, B)
       -> Y
OR2 (A, B)
       -> Y
NOR2 (A, B)
       -> Y
EOR2 (A, B)
       -> Y
RSFF (S_, R_)
       -> Q, Q_
```

```
    /* RSFF.C */

#include<stdio.h>
#include"\h\head.h"
```

```
#include"\h\global.h"
#include"\h\main.h"
#include"\h\time.h"

struct _AND2 {
char Y, Y__;
char A__, B__;
long delay;
};

void AND2_f (struct _AND2 *,  char,  char);
void AND2_f (_p, A, B)
struct _AND2 *_p;
char A, B;
{

if (! Now) {
_p -> delay = 0;
_p -> A__ = INDEF_VAL;
_p -> B__ = INDEF_VAL;
_p -> Y = INDEF_VAL;
}
AND2_ci (_p, A, B);
if (_p -> delay == Now || !_p -> delay) {
_p -> Y = ((char)(_p -> A__ & MASK) & (char)
(_p -> B__ & MASK));
_p -> delay = 0;
}
```

```
}
```



```
static AND2_ci (_p, A, B)
struct _AND2 *_p;
char A, B;
{
int __f = 0;

if (_p -> A__ != A) { _p -> A__ = A; __f = 1; }
if (_p -> B__ != B) { _p -> B__ = B; __f = 1; }
return (__f);
}

struct _ND2 {
char Y, Y__;
char A__, B__;
long delay;
};

void ND2_f (struct _ND2 *,  char,  char);
void ND2_f (_p, A, B)
struct _ND2 *_p;
char A, B;
{
if (! Now) {
_p -> delay = 0;
_p -> A__ = INDEF_VAL;
_p -> B__ = INDEF_VAL;
_p -> Y = INDEF_VAL;
```

```
}
ND2_ci (_p, A, B);
if (_p -> delay == Now || !_p -> delay) {
_p -> Y = ( ~ (((char)(_p -> A__ & MASK) & (char)
(_p -> B__ & MASK))) & MASK);
_p -> delay = 0;
}
}
static ND2_ci (_p, A, B)
struct _ND2 *_p;
char A, B;
{
int __f = 0;

if (_p -> A__ != A) { _p -> A__ = A; __f = 1; }
if (_p -> B__ != B) { _p -> B__ = B; __f = 1; }
return (__f);
}

struct _OR2 {
char Y, Y__;
char A__, B__;
long delay;
};
void OR2_f (struct _OR2 *,  char,  char);
void OR2_f (_p, A, B)
struct _OR2 *_p;
char A, B;
```

```
{
if (! Now) {
_p -> delay = 0;
_p -> A__ = INDEF_VAL;
_p -> B__ = INDEF_VAL;
_p -> Y = INDEF_VAL;
}
OR2_ci (_p, A, B);
if (_p -> delay == Now || !_p -> delay) {
_p -> Y = ((char)(_p -> A__ & MASK) |
(char)(_p -> B__ & MASK));
_p -> delay = 0;
}
}
static OR2_ci (_p, A, B)
struct _OR2 *_p;
char A, B;
{
int __f = 0;

if (_p -> A__ != A) { _p -> A__ = A; __f = 1; }
if (_p -> B__ != B) { _p -> B__ = B; __f = 1; }
return (__f);
}
struct _NOR2 {
char Y, Y__;
char A__, B__;
```

```
long delay;
};

void NOR2_f (struct _NOR2 *,  char,  char);
void NOR2_f (_p, A, B)
struct _NOR2 *_p;
char A, B;
{

if (! Now) {
_p -> delay = 0;
_p -> A__ = INDEF_VAL;
_p -> B__ = INDEF_VAL;
_p -> Y = INDEF_VAL;
}
NOR2_ci (_p, A, B);
if (_p -> delay == Now || !_p -> delay) {
_p -> Y = ( ~ (((char)(_p -> A__ & MASK) | (char)
(_p -> B__ & MASK))) & MASK);
_p -> delay = 0;
}
}
static NOR2_ci (_p, A, B)
struct _NOR2 *_p;
char A, B;
{
int __f = 0;
```

```
if (_p -> A__ != A) { _p -> A__ = A; __f = 1; }
if (_p -> B__ != B) { _p -> B__ = B; __f = 1; }
return (__f);
}

struct _EOR2 {
char Y, Y__;
char A__, B__;
long delay;
};

void EOR2_f (struct _EOR2 *,  char,  char);
void EOR2_f (_p, A, B)
struct _EOR2 *_p;
char A, B;
{
if (! Now) {
_p -> delay = 0;
_p -> A__ = INDEF_VAL;
_p -> B__ = INDEF_VAL;
_p -> Y = INDEF_VAL;
}
EOR2_ci (_p, A, B);
if (_p -> delay == Now || !_p -> delay) {
_p -> Y = ((char)(_p -> A__ & MASK) ^
(char)(_p -> B__ & MASK));
_p -> delay = 0;
}
```

```
}
static EOR2_ci (_p, A, B)
struct _EOR2 *_p;
char A, B;
{
int __f = 0;

if (_p -> A__ != A) { _p -> A__ = A; __f = 1; }
if (_p -> B__ != B) { _p -> B__ = B; __f = 1; }
return (__f);
}

struct _RSFF {
char Q, Q__, Q_, Q___;
char S___, R___;
long delay;
struct _ND2 G1;
struct _ND2 G2;
};

void RSFF_f (struct _RSFF *,  char,  char);
void RSFF_f (_p, S_, R_)
struct _RSFF *_p;
char S_, R_;
{
int __c = 0;
if (! Now) {
_p -> delay = 0;
```

```
_p -> S___ = INDEF_VAL;
_p -> R___ = INDEF_VAL;
_p -> Q = INDEF_VAL;
_p -> Q_ = INDEF_VAL;
}
RSFF_ci (_p, S_, R_);
if (_p -> delay == Now || !_p -> delay) {
do {
ND2_f (&_p -> G1, (char)(_p -> S___ & MASK),
*(char *)&_p -> G2);
ND2_f (&_p -> G2, (char)(_p -> R___ & MASK),
*(char *)&_p -> G1);
_p -> Q = *(char *)&_p -> G1;
_p -> Q_ = *(char *)&_p -> G2;
if (__c++ >= Count) {
exe_error (); break;
}
} while (RSFF_co (_p, _p -> Q, _p -> Q_));
_p -> delay = 0;
}
}
static RSFF_ci (_p, S_, R_)
struct _RSFF *_p;
char S_, R_;
{
int __f = 0;
if (_p -> S___ != S_) { _p -> S___ = S_; __f = 1; }
if (_p -> R___ != R_) { _p -> R___ = R_; __f = 1; }
```

```
return (__f);
}
static RSFF_co (_p, Q, Q_)
struct _RSFF *_p;
char Q, Q_;
{
int __f = 0;

if (_p -> Q__ != Q) { _p -> Q__ = Q; __f = 1; }
if (_p -> Q___ != Q_) { _p -> Q___ = Q_; __f = 1; }
return (__f);
}
```

หลังจากนั้นก็จะใช้คำสั่ง LCS ตามด้วยชื่อของ SIM ไฟล์โดยมีรูปแบบดังนี้

LCS (*.SIM)

เมื่อใช้ LCS แปลง RSFF.SIM ที่เขียนขึ้นเพื่อกำหนดจุดทางเข้า - ออก ของสัญญาณ ที่ใช้ทดสอบการทำงานของวงจร อาร์เอส ฟลิปฟลอบ โดยใช้คำสั่ง LCS RSFF จะได้ ซอร์ท ไฟล์ ดังนี้

```
/* EXE.C */

#include<stdio.h>
#include"\h\head.h"
#include"\h\global.h"
#include"\h\main.h"
#include"\h\time.h"
```

```
#include"r.h"
#defineMAX 4
struct _RSFF RSFF_d;

void program ()
{
char S_, R_;
char __0[MAX], __1[MAX];
long t0 = -1, t1;
int i;

fputs ("\tName'RSFF'\n", stderr);
fputs ("  NAME   RSFF\n", Out_fp);
for (i = 0; i < MAX; i++) {
__0[i] = 0;
}
fputs ("  TIME: R_    S_    Q     Q_    \n", Out_fp);
for (; ;) {
R_ = get_digit (); insymbol ();
S_ = get_digit (); insymbol ();
t1 = time_set ();
push_time (t1);
fprintf (stderr,"\tTime\t'%1d'\n", t1);
do {
if ((Now = pop_time ()) == t0 && Time_begin == Time_end){
break;
}
t0 = Now;
```

```
RSFF_f (&RSFF_d, S_, R_);
__1[0] = R_;
__1[1] = S_;
__1[2] = RSFF_d.Q;
__1[3] = RSFF_d.Q_;
pr_chk (__0, __1, MAX);
} while (Time_begin -> time < t1);
if ((Atr == EOF || Atr == END_STATE) && Now) {
insymbol ();
expect (';');
break;
}
}
}
```

ต่อไปเป็นการ คอมไพล์ ซอร์ท ไฟล์ทั้งลองแล้วทำการ ลิงค์ ออพเจ็ค ไฟล์ทั้งลองเข้าด้วยกันเป็น เอ็กซิคิวท์ ไฟล์ โดยใช้โปรแกรม TCC ในการคอม-ไพล์ และ TLINK ในการ ลิงค์ ตามคำสั่ง

TCC (*.c) EXE LCSS.LIB

โดยที่ *.c เป็น โปรแกรมที่เราต้องการจะคอมไพล์ และ ลิงค์ ส่วน EXE.C ได้จากการแปลง SIM ไฟล์ และ LCSS.LIB เป็น ไลเบอร์รี่ (library) ที่เก็บคำสั่งต่าง ๆ ที่ใช้ใน EXE.C และ *.C

จากนี้จะได้ เอ็กซิคิวท์ ไฟล์ ขั้นตอนต่อไปจะนำข้อมูลใน DAT ไฟล์มาร่วมประมวลผลด้วยโดยใช้คำสั่ง

(*.EXE) (*.DAT)

หลังจากใช้คำสั่ง RSFF RSFF จะได้ เอาท์พุท ไฟล์ ดังนี้

```
/* RSFF.OUT */

NAME   RSFF

TIME:  R_    S_    Q     Q_
    0:  0     0     1     1
    1:  0     1     0     1
    2:  1     0     1     0
    3:  1     1     1     0
    4:  0     0     1     1
```

ขั้นตอนสุดท้ายเป็นการแปลง เอาท์พุท ไฟล์ ที่ได้ใช้เป็นแผนภูมิเวลาโดยใช้คำสั่ง

```
LCG ( *.OUT )
```

เมื่อขั้นตอนสุดท้ายเสร็จสิ้นก็เป็นการสิ้นสุด การทำงานของระบบที่สร้างขึ้น

บทที่ ๔
วิจารณ์และสรุปผล

โครงงานนี้มีวัตถุประสงค์เพื่อที่ศึกษาการจำลองวงจรอย่างง่าย โดยใช้ภาษาซีในการเขียนโปรแกรม พบปัญหาที่เกิดขึ้นมากมายตั้งแต่การใช้คำสั่งต่างๆ, ใช้โปรแกรมโครงสร้าง ตลอดจนการลิงค์โปรมแกรมย่อยเข้าด้วย ทำให้ล่าช้ามาก แต่ก็ได้รับการปรับปรุงแก้ไข จนใช้งานได้หลายส่วน แต่ก็ยังเหลือบางส่วนที่ยังแก้ไขไม่สำเร็จ นั่นคือ การแปลง OUT ไฟล์ ให้เป็นแผนภูมิเวลา LCD ไฟล์ที่เขียนขึ้น เพื่อจำลองการทำงานของวงจรต่างๆที่ยกตัวอย่างขึ้นมา จึงยังไม่มีผลการทำงาน

ในอนาคตถ้าได้ทำการปรับปรุงให้โปรแกรมใช้งานได้จริง จะมีประโยชน์มากในการศึกษาการทำงานของวงจรลอจิกต่างๆ รวมทั้งถ้าได้รับการพัฒนาต่อไปอีกให้ใช้กับวงจรอนาลอกได้ด้วย จะยิ่งสมบูรณ์แบบมากขึ้น

กิตติกรรมประกาศ

การทดลองวิจัยสร้าง ระบบจำลองการทำงานของวงจรลอจิกนี้ ได้รับความกรุณาเป็นอย่างดี ในการจัดหา อุปกรณ์ เครื่องมือ แหล่งข้อมูล ตลอดจนเวลา สถานที่ และ คำปรึกษา จากอาจารย์ มนัส สังวรศิลป์ อาจารย์ที่ปรึกษา นอกจากนี้ยังได้รับคำแนะนำ ความช่วยเหลือ และอำนวยความสะดวกอย่างมาก จากอาจารย์ สุรพันธุ์ เอื้อไพบูลย์, อาจารย์ พิชัย คูศิริวานิชกร ตลอดจนรุ่นพี่และเพื่อน ๆ อีกหลายคน จึงขอขอบพระคุณและ ขอบคุณมา ณ ที่นี้ด้วย

หนังสืออ้างอิง

เอกสารอ้างอิงเป็นหนังสือภาษาไทยและภาษาอังกฤษ จัดเรียงตามลำดับดังนี้

๑.บุญเลิศ เอี่ยมทัศนา, ยืน ภู่วรวรรณ, สมนึก คีรีโต, "โปรแกรมคอมพิวเตอร์ ภาษาซี",บริษัท ซีเอ็ดยูเคชั่น จำกัด,พิมพ์ครั้งที่ ๑, ๒๕๒๙

๒.กฤษดา วิศวธีรานนท์,"เรียน/เล่น/ใช้ ไอซีดิจิตอล",บริษัท ซีเอ็ดยูเคชั่นจำกัด, พิมพ์ครั้งที่ ๒, ๒๕๓๑

๓."คู่มือ / เทียบเบอร์ ไอซี TTL", บริษัท ซีเอ็ดยูเคชั่น จำกัด ,พิมพ์ครั้งที่ ๖, ๒๕๓๑

๔."TURBO C", Borland International, Inc., First Printing, 1987

๕.Herbert Schildt, "Advanced C", McGraw-Hill , Inc., 1986

๖."STRETCHING TURBO C ", Kent Portor , A Bandy Book Distribute by Prentice Hall Trade , New York, 1989