สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

วงจรควบคุมอุณหภูมิแบบฟัซซี่ลอจิก

FUZZY LOGIC TEMPERATURE CONTROLLER

โดย

นาย คำรงศักดิ์ ทองสมพร เลขประจำตัว 40013169

นาย ปิยวิทย์ เจียรวรรณ์ เลขประจำตัว 40013178

อาจารย์ที่ปรึกษา

ผศ.ดร สมศักดิ์ ชุมช่วย

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2542

เลขหมู่.....
เลขทะเบียน 36944
วัน, เดือน, ปี 29 ส.ค. 254

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปริญญานิพนธ์ปีการศึกษา 2542

ภาควิชา อิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง วงจรควบคุมอุณหภูมิแบบฟัซซีลอจิก

ผู้จัดทำ

1. นาย ดำรงศักดิ์ ทองสมพร รหัสประจำตัว 40013169
2. นาย ปิยวิทย์ เจียรวรรณ์ รหัสประจำตัว 40013178

.....อาจารย์ที่ปรึกษา

(ผศ.ดร.สมศักดิ์ ชุมช่วย)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

วงจรควบคุมอุณหภูมิแบบฟัซซีลอจิก

FUZZY LOGIC TEMPERATURE CONTROLLER

นาย ดำรงศักดิ์ ทองสมพร รหัสประจำตัว 40013169

นาย ปิยวิทย์ เจียรวรรณ์ รหัสประจำตัว 40013178

โครงงานได้รับการตรวจสอบแล้ว พร้อมที่จะทำการสอบได้

.....

(ผศ.ดร.สมศักดิ์ ชุมช่วย)

อาจารย์ที่ปรึกษา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

วงจรควบคุมอุณหภูมิแบบฟัซซีลอจิก

นาย คำรงศักดิ์ ทองสมพร
นายปิยวิทย์ เจียรวรรณ์
ผศ.ดร.สมศักดิ์ ชุมช่วย อาจารย์ที่ปรึกษา
ปีการศึกษา 2542

บทคัดย่อ

วงจรควบคุมอุณหภูมิแบบฟัซซีลอจิก เป็นการประยุกต์ใช้งานของระบบควบคุมแบบดิจิตอล โดยจะเป็นการควบคุมแบบพีไอ (PI : Proportinal puls Integral control) ที่ใช้วิธีการทางฟัซซีลอจิก (Fuzzy Logic) ซึ่งระบบจะควบคุมอุณหภูมิแบบอัตโนมัติ โดยระบบจะคำนวนและปฏิบัติการด้วยระบบดิจิตอลทั้งหมด เนื่องจากเดิมทีนั้นระบบการควบคุมแบบพีไออย่างเดียวนั้น ไม่สามารถปรับระบบให้เหมาะสมกับการทำงานได้ทุกสภาวะ ฉะนั้นเมื่อเพิ่มส่วนที่เป็นฟัซซีลอจิกเข้าไปแล้ว ทำให้ปรับคุณลักษณะการทำงานของระบบพีไอให้เหมาะสมกับสภาพแวดล้อมต่างๆ จึงทำให้ระบบควบคุมทำงานได้อย่างมีประสิทธิภาพ โดยระบบทั้งหมดจะถูกสังเคราะห์ จะนำผลลัพธ์โปรแกรมลง เอฟพีจีเอ (FPGA) และนำมาต่อกับวงจรภายนอก เพื่อใช้งานควบคุมอุณหภูมิต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่1
บทนำ

การควบคุมอุณหภูมิให้มีประสิทธิภาพ คือ สามารถปรับอุณหภูมิให้คู่เข้าค่าใดค่าหนึ่งได้อย่าง รวดเร็วและมีความไว (Sensitivity) ในการปรับค่าอุณหภูมิ ของระบบเมื่อมีการเปลี่ยนแปลงอุณหภูมิ เกิดขึ้นซึ่งในโปรเจกนี้นั้นเป็นการประยุกต์ใช้งานระบบควบคุมอุณหภูมิแบบอัตโนมัติ โดยใช้ระบบควบคุมแบบดิจิตอล โดยเป็นการควบคุมอุณหภูมิแบบพีไอ (PI : Proportional Plus Integral Control) ซึ่งระบบควบคุมแบบพีไอนี้มีฟังก์ชันการถ่ายโอน (Transfer Function) เป็น

$$T(s) = Kp + Ki/s$$

ซึ่งคุณสมบัติต่างๆของระบบ เช่น ผลตอบสนองของระบบ การพุ่งเกินของระบบ (Overshoot), ช่วงเวลาขึ้น (Rise Time), ช่วงเวลาเข้าที่ (Settling Time) ขึ้นอยู่กับอัตราส่วนของสัมประสิทธิ์ของ Kp / Ki ดังนั้นการเปลี่ยนแปลงค่า Kp และ Ki จึงมีผลต่อเสถียรภาพของระบบด้วย จึงเป็นข้อจำกัดของการควบคุมลักษณะนี้คือ

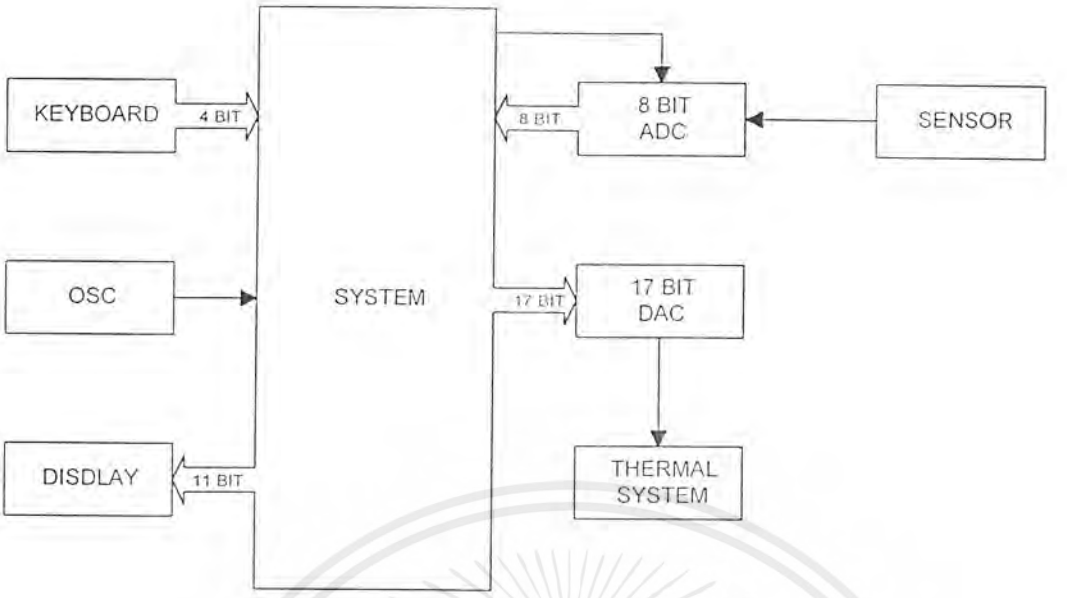
1. เมื่อเลือกอัตราส่วนของ Kp / Ki ค่ามากจะทำให้เกิด การพุ่งเกินของระบบ ขึ้นก่อนที่ระบบจะเข้าใกล้สถานะอยู่ตัว (Steady State)

2. เมื่อเลือกอัตราส่วนของ Kp / Ki ค่าน้อย ถึงแม้ว่าจะลดการพุ่งเกินของระบบได้ แต่ก็จะทำให้ความไวของวงจรลดลง

ดังนั้นเพื่อให้ได้อัตราส่วนของ Kp / Ki ที่เหมาะสม จึงใช้วิธีการ Fuzzy Logic ในการประมาณค่าของ อินพุทและเอาท์พุทของระบบ เพื่อให้ระบบมีการปรับคุณลักษณะการทำงานให้เหมาะสมกับสภาพแวดล้อมต่างๆ ได้อย่างมีประสิทธิภาพ

ระบบที่กล่าวถึงนั้นสามารถทำได้หลายวิธีการด้วยกัน เช่นการควบคุมโดยใช้ไมโครโปรเซสเซอร์ ซึ่งใช้งานกันอย่างแพร่หลาย ออกแบบได้ง่าย แต่มีสิ้นเปลืองสูง และความเร็วในการทำงานจะต่ำเมื่อเทียบกับการใช้ไอซีเฉพาะงาน ซึ่งนิยมใช้ในการผลิตจำนวนมาก (Mass Product) ซึ่งได้ ระบบจากการสังเคราะห์จากนั้นจะเป็นกระบวนการสร้างไอซี (Fabrication) ทำให้ระบบมีขนาดเล็ก และมีความเร็วสูง หรือ สามารถนำผลที่ได้จากการสังเคราะห์ไปโปรแกรม ลง FPGA เพื่อใช้งานต่อไป จากที่กล่าวมาข้างต้น สามารถออกแบบระบบได้ดังรูปที่ 1.1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่1.1 แสดงระบบควบคุมอุณหภูมิที่ต้องการ

จากรูป ระบบที่ต้องการจะออกแบบนั้นมีการรับอินพุทจากสิ่งแวดล้อมโดยใช้ ระบบเซ็นเซอร์อุณหภูมิส่งให้กับระบบ โดยผ่าน วงจร ADC ขนาด 8 บิต ที่เลือกใช้ ADC ขนาด8 บิตก็เนื่องจากต้องการความละเอียดในการคำนวณ และสามารถตั้งอุณหภูมิได้โดยผ่าน คีย์บอด มีส่วนแสดงผลซึ่งจะแสดงอุณหภูมิที่ตั้งจากคีย์บอด และอุณหภูมิขณะใดๆที่ได้จากระบบเซ็นเซอร์ และระบบจะส่งเอาท์พุทที่มีขนาด 17 บิต ผ่าน DAC ไปยังระบบเครื่องทำความเย็น ต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2
การออกแบบในเบื้องต้น

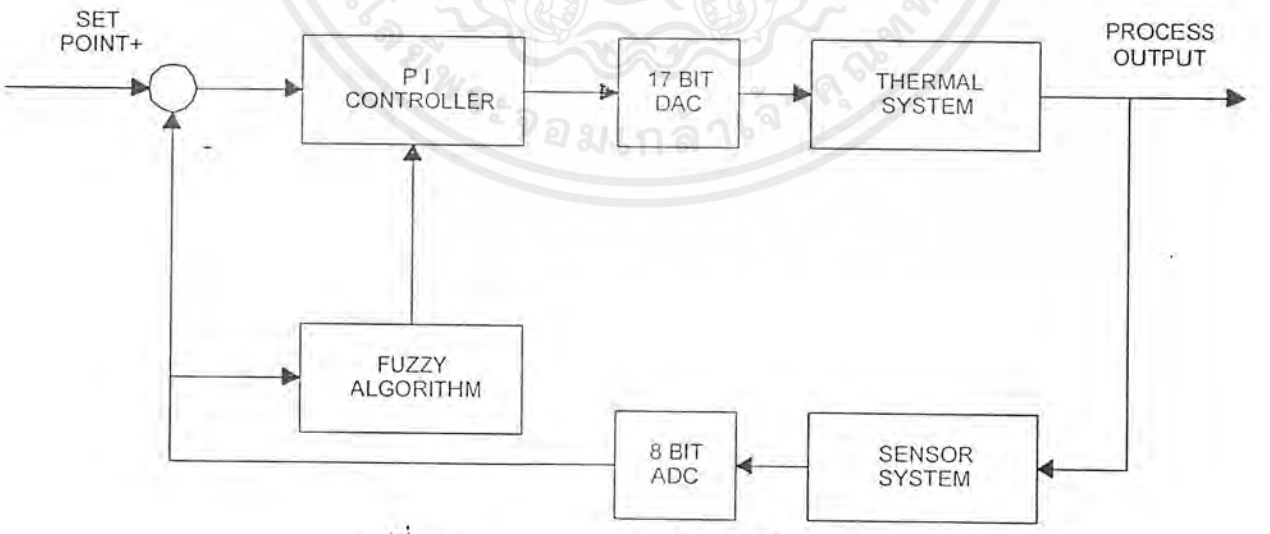
ในระบบการควบคุมอุณหภูมิแบบพีไอ (PI : Proportional Integral Control) นั้น โดยตัวของระบบเองจะมีคุณสมบัติเป็นวงจรกรองความถี่ต่ำ (Low Pass Filter) ที่พยายามจะทำให้ ความผิดพลาดในสถานะอยู่ตัว (Steady State Error) เป็น 0 ซึ่งปัญหาของเราก็คือ ทำอย่างไรเราจึงจะสามารถให้ระบบการควบคุมแบบพีไอ สามารถปรับตัวเองให้เหมาะสมกับการทำงานในสภาวะต่างๆ ได้อย่างมีประสิทธิภาพ ซึ่งคำตอบของเราจะมีพื้นฐานมาจากการตอบสนองความถี่ของระบบพีไอ

ระบบพีไอเป็นระบบที่ฟังก์ชั่นถ่ายโอนเป็น

$$T(s) = K_p + K_i/s \quad ;T(s) = \text{Transfer Function}$$
$$= [K_p s + K_i]/s$$

คุณสมบัติต่างๆของระบบอาทิเช่น ผลตอบสนองของระบบการ พุ่งเกินของระบบ ,ช่วงเวลาขึ้น (Rise Time),ช่วงเวลาเข้าที่ (Settling Time) ขึ้นอยู่กับอัตราส่วนของสัมประสิทธิ์ของ K_p/K_i ดังนั้นการเปลี่ยนแปลงค่า K_p และ K_i ก็จะมีผลต่อเสถียรภาพของระบบด้วย ฉะนั้นในการปรับค่า K_i และ K_p จะต้องคำนึงถึงหลักใหญ่ๆ อยู่ 2 ประการ คือ

1. ต้องไม่ทำให้ระบบเกิดออสซิเลท (Oscillate)
2. ต้องปรับให้ระบบมีประสิทธิภาพสูงสุด



(รูปที่ 2.1 แสดงระบบควบคุมอุณหภูมิ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากรูปที่ 2.1 จะเห็นว่า ถ้าเราจะหาฟังก์ชั่นถ่ายโอนแบบปิด (Close Loop Transfer Function) ของระบบรวมเป็นไปได้ค่อนข้างลำบาก เนื่องจากเราไม่สามารถทราบถึงฟังก์ชันถ่ายโอนของวงจร DAC (Digital to Analog Converter), ADC (Analog to Digital Converter), ระบบอุณหภูมิ (Thermal System), เซ็นเซอร์ (Sensor), Fuzzy Algorithm ดังนั้นในการกำหนดค่า K_p,K_i ซึ่งจะรักษาเสถียรภาพของระบบนั้น จึงต้องเป็นการทำแบบลองผิดลองถูกกับวงจร ซึ่งเราสามารถทำได้โดยใช้การเลียนแบบ (Simulation) ในคอมพิวเตอร์ เพื่อหาค่าที่เหมาะสมต่อไป แต่ในที่นี้ก็จะมีการกำหนดค่า K_p,K_i ขึ้นมาก่อน เพื่อให้เป็นค่าที่ใช้ลองผิดลองถูก ดังจะกล่าวถึงในหัวข้อต่อไป

2.1 FUZZIFICATION OF INPUT

เราจะนำอุณหภูมิด้านเข้า (Input Temperature ความละเอียดในการวัดอยู่ที่ 0.1 องศาเซลเซียส) ที่ได้มาหาค่าระดับขั้นการเป็นสมาชิก (Degree of Membership) โดยเราจะทำการแปลง อุณหภูมิด้านเข้า เป็นอินพุทของ Fuzzy

ระบบตรรก (Logic System) ได้ 2 ค่า คือ

1. อุณหภูมิผิดพลาด (Error temperature) E(t) = Set Temp – T(t)
2. การเปลี่ยนแปลงของอุณหภูมิผิดพลาด (Change of Error Temperature) $\Delta E = E(t) - E(t-1)$

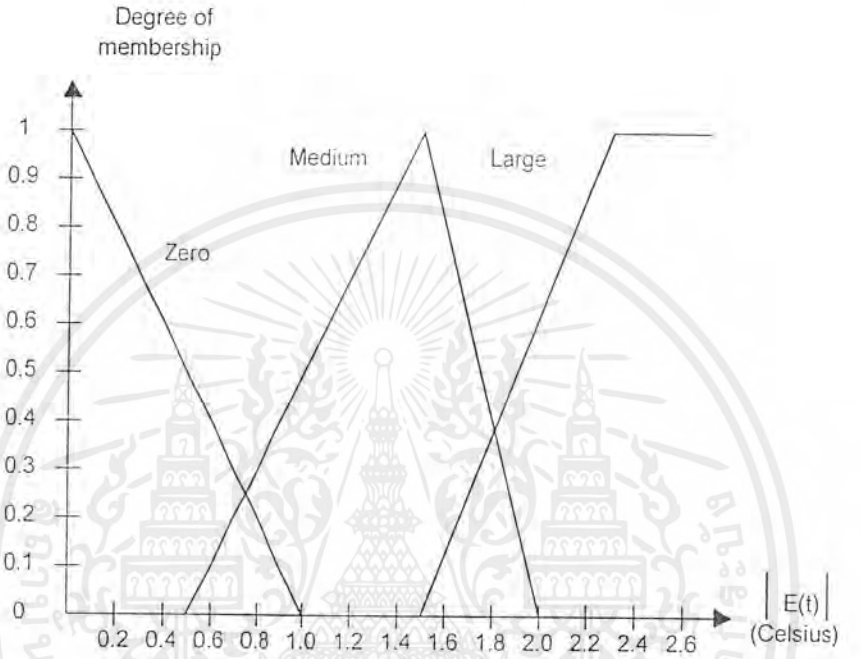
โดยที่

Set Temp	=	อุณหภูมิที่ตั้งไว้
T(t)	=	อุณหภูมิในขณะนั้น
E(t)	=	ค่า Error ในปัจจุบัน
E(t-1)	=	ค่า Error ก่อนหน้านั้น
ΔE	=	ค่าการเปลี่ยนแปลงของอุณหภูมิผิดพลาด

จากตัวแปร E(t) และ ΔE ดังกล่าวข้างต้น เราจะใช้เป็น Fuzzy Input ให้กับส่วน Fuzzification เพื่อหาค่าสัมประสิทธิ์ K_p,K_i ให้เหมาะสมตามการทำงานในแต่ละสภาวะ ซึ่งค่า E(t) และ ΔE มีโอกาสที่จะเป็นไปได้ทั้งค่าบวกและค่าลบ ซึ่งถ้าพิจารณาดูให้ดีจะเห็นว่า ค่า E(t) และ ΔE ที่เป็นลบ จะมีค่าสัมประสิทธิ์ K_p,K_i เหมือนกับค่า E(t) และ ΔE ค่าเท่ากันกับที่เป็นบวก เพราะว่าค่าประสิทธิ์ K_p,K_i นี้จะเป็นตัวกำหนดความเร็วในการเข้าสู่สภาวะ Steady State เท่านั้น ในการ Fuzzification เราจึงไม่ต้องสนใจเครื่องหมายของ E(t) และ ΔE ทำให้ตัวแปรที่จะใช้หาค่า Membership Function มีสงตัวคือ $|E(t)|$ และ $|\Delta E|$

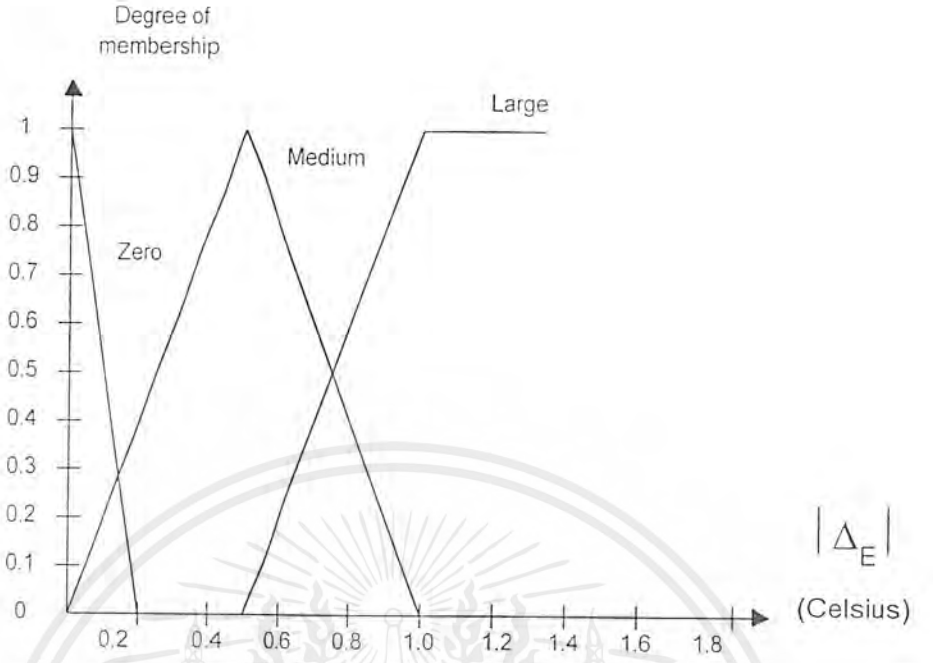
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โดยจากตัวแปรอินพุททั้งสองตัวนี้ $|E(t)|$ และ $|\Delta E|$ จะต้องนำมาเปรียบเทียบกราฟความเป็นสมาชิกที่กำหนดขึ้นมาก่อน ดังรูปที่ 2.2 และ 2.3 ซึ่งกราฟความเป็นสมาชิกนี้จะกำหนดขึ้นมาโดยอาศัยความต้องการและความรู้ของมนุษย์ในการเลือกค่าอุณหภูมิ แล้วเราต้องค่อยๆ ปรับกราฟนี้จนสอดคล้องกับความเป็นจริงมากที่สุด โดยรูปที่ 2.2 จะเป็นกราฟความเป็นสมาชิกของ Error Temperature และรูปที่ 2.3 จะเป็นกราฟความเป็นสมาชิกของ Change of Error Temperature



รูปที่ 2.2 ค่า Membership ของ $|E(t)|$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.3 แสดงค่า Membership ของ $|\Delta E|$

	$\|E(t)\|$	$\|\Delta E\|$	K_P	K_I	STATUS
1	ZERO	ZERO	20	12	A
2	ZERO	MEDIUM	20	12	A
3	ZERO	LARGE	16	10	B
4	MEDIUM	ZERO	16	10	B
5	MEDIUM	MEDIUM	20	10	C
6	MEDIUM	LARGE	20	10	C
7	LARGE	Any Value	24	8	D

ตารางที่ 2.1 แสดง Fuzzy IF-THEN Rule

การตั้งกฎเหล่านั้นขึ้นมาจะอาศัยหลักเกณฑ์ดังนี้ คือ

1. มี $|E(t)|$ และ $|\Delta E|$ เท่านั้นที่เป็นอินพุท
2. เอาท์พุททั้งหมดเป็น Integer
3. ค่า K_i , K_p จะเปลี่ยนไปพร้อมๆ กัน
4. เมื่อ Error Signal มาก K_p/K_i จะต้องมากเพื่อจะเพิ่มสัญญานควบคุม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5. เมื่อ Error Signal เริ่มลดลง K_p/K_i จะลดลงเพื่อ Overshoot

6. เมื่อ Error Signal น้อย K_p/K_i จะมีค่ามากเพื่อเพิ่มความไว (Sensitivity)

การคำนวณค่าตามกฎที่ตั้งไว้ โดยอาศัยความเป็นสมาชิกของตัวแปร ที่ได้จากการเปรียบเทียบค่า Fuzzy Input กับกราฟความเป็นสมาชิกเป็นตัวกำหนดความสำคัญของแต่ละกฎ โดยอาศัยเทคนิคของค่ากฎ Smallest ระหว่างตัวแปร Fuzzy Input ทั้ง 2 ตัว ในส่วนเงื่อนไข คือ Error และ Change of Error ตาม Fuzzy IF-THEN Rule ดังตารางที่ 2.1

จากตารางที่ 2.1 นั้นค่าผลลัพธ์ที่เกิดจากแต่ละเงื่อนไขของแต่ละตัว มีค่าตามตารางที่ 2.2

STATUS (TEMPERATURE)	ASSIGNED VALUE (WEIGHT)
A	1
B	2
C	3
D	4

ตารางที่ 2.2 ตารางแสดงค่าสภาวะ

2.2 DEFUZZIFICATION OF OUTPUT

เป็นขั้นตอนที่นำเอาผลลัพธ์ที่ได้จากขั้นตอน Rules Evaluation เพื่อหาค่าของ Fuzzy Output ซึ่งเมื่อทำการเปรียบเทียบตามกฎต่างๆ แล้ว จะเห็นได้ว่าถ้ามีการคำนวณ ตามกฎการควบคุม แล้วนำเพียงกฎใดกฎหนึ่งมาใช้จะทำให้ผลการควบคุมระบบผิดพลาดได้ จึงได้นำค่าเอาท์พุทของกฎต่างๆ มาหาค่าเอาท์พุทที่จะใช้ในการควบคุมโดยใช้การคำนวณแบบ Center of Gravity คำนวณผลลัพธ์ออกมา โดยค่าของจุดศูนย์ถ่วงสำหรับตัวแปรทางเอาท์พุทได้แสดงดังตารางที่ 2.2

$\lvert\Delta E\rvert$ / $\lvert E(t)\rvert$	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	≥1.0
0	1	1	1	1	1	1	1.5	1.5	1.5	1.5	2
0.1	1	1	1	1	1	1	1.5	1.5	1.5	1.5	2
0.2	1	1	1	1	1	1	1.5	1.5	1.5	1.5	2
0.3	1	1	1	1	1	1	1.5	1.5	1.5	1.5	2
0.4	1	1	1	1	1	1	1.5	1.5	1.5	1.5	2
0.5	1	1	1	1	1	1	1.5	1.5	1.5	1.5	2
0.6	1.5	1.75	2	2	2	2	2.25	2.25	2.25	2.25	2.5

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

0.7	1.5	1.75	2	2	2	2	2.25	2.25	2.25	2.25	2.5
0.8	1.5	1.75	2	2	2	2	2.25	2.25	2.25	2.25	2.5
0.9	1.5	1.75	2	2	2	2	2.25	2.25	2.25	2.25	2.5
1.0	2	2.5	3	3	3	3	3	3	3	3	3
1.1	2	2.5	3	3	3	3	3	3	3	3	3
1.2	2	2.5	3	3	3	3	3	3	3	3	3
1.3	2	2.5	3	3	3	3	3	3	3	3	3
1.4	2	2.5	3	3	3	3	3	3	3	3	3
1.5	2	2.5	3	3	3	3	3	3	3	3	3
1.6	3	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5
1.7	3	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5
1.8	3	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5
1.9	3	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5
≥ 2.0	4	4	4	4	4	4	4	4	4	4	4

ตารางที่ 2.3 ตารางแสดงค่าเอาท์พุทจากการคำนวณโดย Fuzzy Rules

หลังจากการที่ได้ทำการคำนวณตาม Rule Evaluation แล้วทำการหาค่า Center of Gravity แล้ว สามารถเขียนตารางของ Fuzzy Output สำหรับค่า Error และ Change of Error ต่างๆ ได้ตามตารางที่ 2.3

เมื่อได้ผลลัพธ์จากการทำ Defuzzification แล้วก็จะได้ค่าของการคำนวณออกมาในรูปของ Fuzzy Output ซึ่งมีค่าของ Fuzzy Output นี้ยังไม่สามารถไปใช้ในการควบคุมอุณหภูมิได้โดยตรงจึงต้องนำมาผ่านขบวนการที่จะทำการเปลี่ยนให้ค่าของ Fuzzy Output นั้นสามารถนำไปใช้ในการควบคุมเครื่องทำน้ำอุ่นได้ ซึ่งวิธีการก็คือ จะนำค่า Fuzzy Output ไปเปรียบเทียบกับตารางที่ 2.4 ซึ่งเป็นขบวนการกลับกันกับการหาค่า Fuzzy Input เพื่อหาสถานะของค่า K_p และ K_i เมื่อได้แล้วทำการเปลี่ยนค่า Fuzzy Output ให้เป็นค่าสัมประสิทธิ์ K_p/K_i ซึ่งจะได้สัมประสิทธิ์ของระบบพีไอที่ Error Temperature และ Change of Error Temperature ต่างๆ ดังแสดงในตารางที่ 2.5

FUZZY OUTPUT	STATUS	(Kp,Ki)
0.0 - 1.4	A	(20,12)
1.5 - 2.4	B	(16,10)
2.5 - 3.4	C	(20,10)
3.5 - 4.0	D	(24,8)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 2.4 แสดงค่าความสัมพันธ์ของ Fuzzy Output กับค่าสัมประสิทธิ์ K_p, K_i

จากขั้นตอนต่างๆ ที่กล่าวมาข้างต้น จะขอแสดงการหาค่า Fuzzy Output เพื่อเป็นตัวอย่าง 1 ค่า โดยในตอนแรกเราต้องสมมติค่าต่างๆ ดัง

Set Temperature = 32.7 C

T(t) = 31.0 C

T(t-1) = 32.9 C

นำค่าเหล่านี้เข้าสู่ขั้นตอน Fuzzification แปลงเป็น Fuzzy Input จะได้ค่า

$|\text{Error Temperature } (E(t))| = |\text{Set} - T(t)|$

$= 32.7 - 31.0$

$= 1.7$

$|\text{Change of Error Temperature}| = |E(t) - E(t\text{-}1)|$

$= |1.7 - (-0.2)|$

$= 1.9$

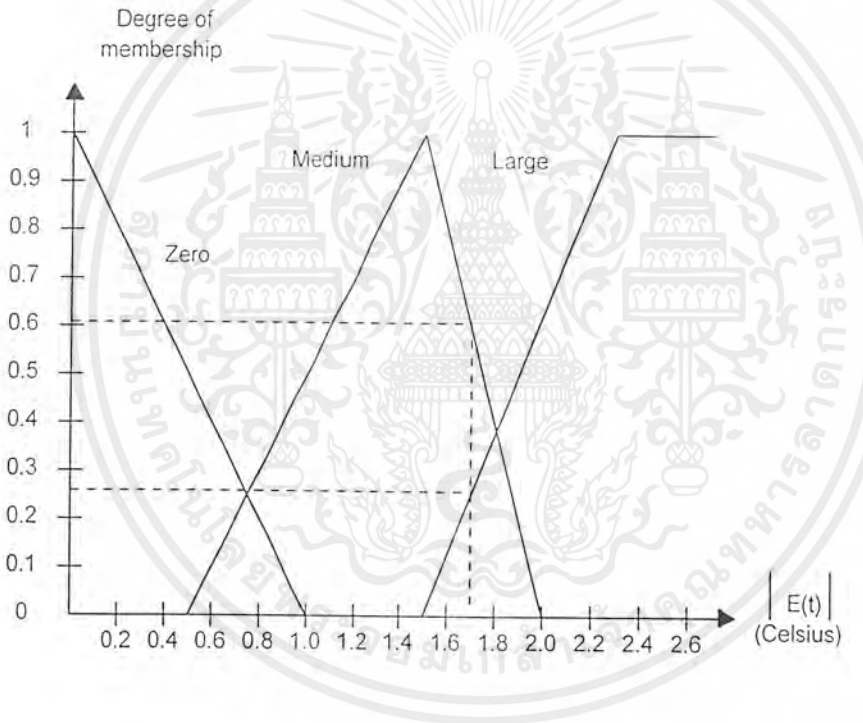
$\frac{\lvert\Delta E\rvert}{\lvert E(t)\rvert}$	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	≥1.0
0	A	A	A	A	A	A	B	B	B	B	B
0.1	A	A	A	A	A	A	B	B	B	B	B
0.2	A	A	A	A	A	A	B	B	B	B	B
0.3	A	A	A	A	A	A	B	B	B	B	B
0.4	A	A	A	A	A	A	B	B	B	B	B
0.5	A	A	A	A	A	A	B	B	B	B	B
0.6	B	B	B	B	B	B	B	B	B	B	C
0.7	B	B	B	B	B	B	B	B	B	B	C
0.8	B	B	B	B	B	B	B	B	B	B	C
0.9	B	B	B	B	B	B	B	B	B	B	C
1.0	B	C	C	C	C	C	C	C	C	C	C
1.1	B	C	C	C	C	C	C	C	C	C	C
1.2	B	C	C	C	C	C	C	C	C	C	C
1.3	B	C	C	C	C	C	C	C	C	C	C
1.4	B	C	C	C	C	C	C	C	C	C	C

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.5	B	C	C	C	C	C	C	C	C	C	C
1.6	C	C	D	D	D	D	D	D	D	D	D
1.7	C	C	D	D	D	D	D	D	D	D	D
1.8	C	C	D	D	D	D	D	D	D	D	D
1.9	C	C	D	D	D	D	D	D	D	D	D
≥2.0	D	D	D	D	D	D	D	D	D	D	D

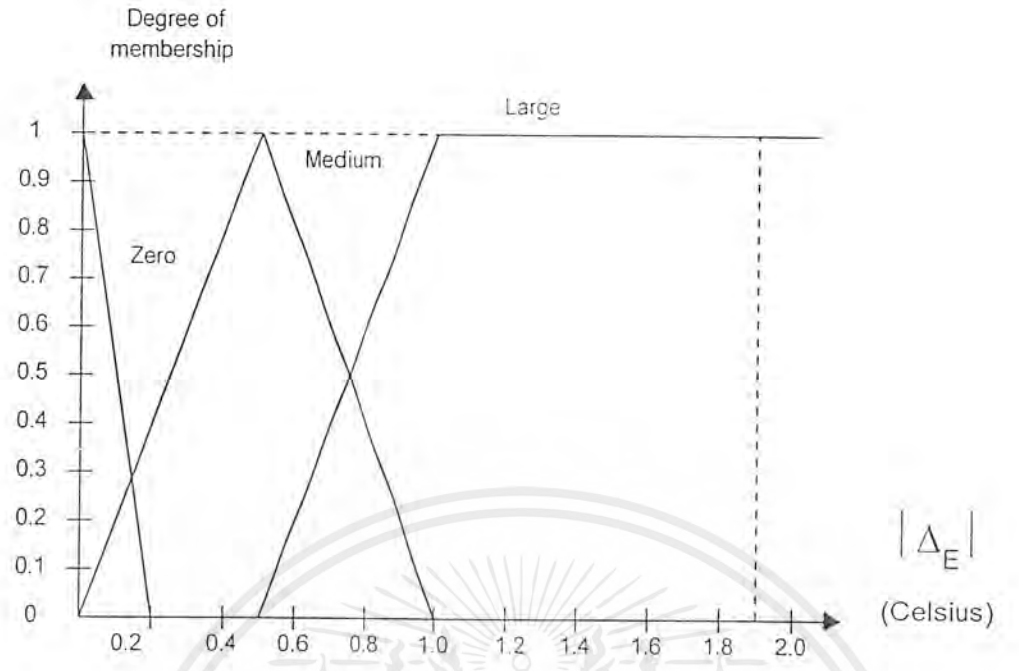
ตารางที่ 2.5 แสดง Fuzzy Output ที่นำไปใช้งานจริง

นำค่า $|E(t)|$ และ $|\Delta E|$ ไปหาค่า Degree of Membership จากกราฟรูปที่ 2.1 และ 2.2 ตามลำดับ



รูปที่ 2.4 แสดงการหาค่า Membership Function ที่ $|E(t)| = 1.7$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.5 แสดงการหาค่า Membership Function ที่ $|\Delta E| = 1.9$

แสดงการหาค่า Degree of Membership ของ E(t) และ ΔE ที่ $|E(t)| = 1.7$ และ $|\Delta E| = 1.9$ ตามลำดับ

จากกราฟจะเห็นว่าค่า $|E(t)|$ เท่ากับ 1.7 จะไปตัดกราฟ Medium และ Large ที่ค่า Membership เท่ากับ 0.25 สำหรับ Large และเท่ากับ 0.6 ที่ Medium ส่วนค่า $|\Delta E| = 1.9$ จะไปตัดกราฟ Large ที่ Membership Function เท่ากับ 1 เมื่อได้ค่าเหล่านี้แล้ว เราจะใช้กฎ Smallest หาค่าตัวแทนกลุ่ม คือ

$$\mu(|E(t)|, |\Delta E|) = \min[\mu(|E(t)|), \mu(|\Delta E|)]$$
$$= 0.25$$

หลังจากนั้นเราจะไปดูตารางที่ 2.1 เพื่อหากฎของ Fuzzy Rule ที่สอดคล้องกับเงื่อนไขทางอินพุทที่หามาได้ จะเห็นว่าจะสอดคล้องกฎที่ 6 และ7 ซึ่งมีค่าน้ำหนักเป็น 3 และ 4 ตามลำดับ แล้วนำค่านี้มาคำนวณแบบ Center of Gravity โดยมีสูตร

$$In = \frac{\sum_{i=1}^{n} (\mu_{In} \times u_{In})}{\sum_{i=1}^{n} \mu_{In}}$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โดยที่ In = Result of Centroid Method

μn = Degree of Membership

un = Status from IF-THEN Rules

$$In = \frac{0.25 \times 4) + (0.25 \times 3)}{(0.25+0.25)}$$

$$= 3.5$$

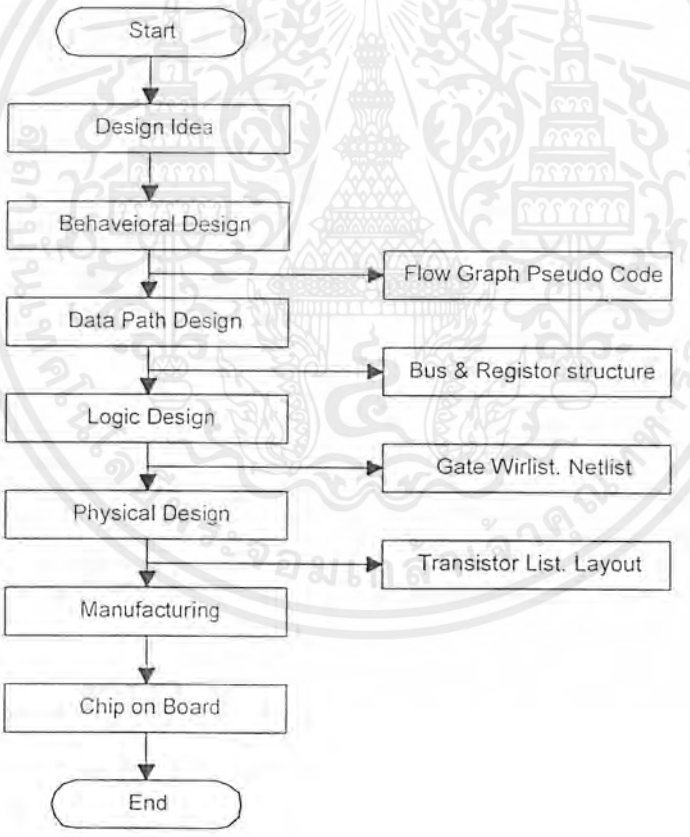
ซึ่งเป็นค่าที่ตรงกับตารางที่ .... และเมื่อนำค่าที่คำนวณได้นี้ไปเปรียบเทียบกับตา-รางที่ 2.4 จะได้ค่าของ Fuzzy Output คือ D ($K_p = 24$, $K_i = 8$) ซึ่งตรงกับค่าในตารางที่ 2.5 ที่ $|E(t)| = 1.7$ และ $|\Delta E| \geq 1.0$ ส่วนการหาค่า Fuzzy Output อื่นๆ ที่อุณหภูมิอื่นๆ ก็จะได้ในทำนองเดียวกัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3
ทฤษฎีและหลักการ

3.1 กล่าวนำ

เพื่อเป็นการแก้ปัญหาที่ยุ่งยากในการออกแบบวงจรเชิงเลขจึงได้นำเอาระบบคอมพิวเตอร์ช่วยออกแบบทางวิศวกรรมเข้ามาช่วยในการออกแบบ ทั้งยังช่วยลดความยุ่งยากและระยะเวลาในการออกแบบให้รวดเร็วขึ้น ภาษาสำหรับบรรยายอุปกรณ์ฮาร์ดแวร์เป็นภาษาหนึ่งที่ได้รับการพัฒนา และนำมาใช้ในการออกแบบระบบเชิงเลข โดยที่มีขั้นตอนตั้งแต่การกำหนดแนวความคิดในการออกแบบจนกระทั่งได้มาเป็นอุปกรณ์ ที่ถูกโปรแกรมการทำงานตามการวางแนวความคิดให้อุปกรณ์ทำงานได้ตามต้องการ

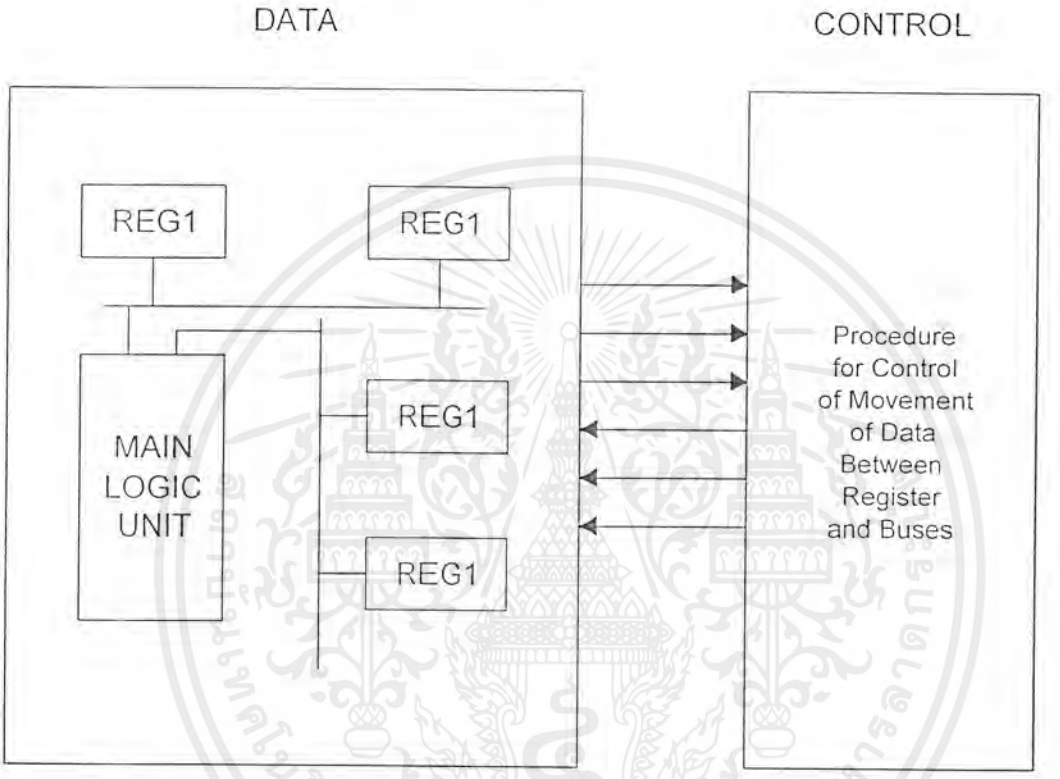


รูปที่ 3.1 ขั้นตอนการออกแบบระบบเชิงเลข

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขั้นตอนดังกล่าวนั้นสามารถอธิบายได้ดังรูปที่ 3.1 โดยที่ขั้นแรกผู้ออกแบบเริ่มกำหนดแนวความคิดในการออกแบบหลังจากนั้นออกแบบระบบในเชิงพฤติกรรมขึ้นมาเพื่อตรวจสอบการทำงาน ซึ่งอาจจะเป็นผังงาน, ผังแสดงแบบ, หรือคำสั่งเทียม (Pseudo Code) ก็ได้

ขั้นต่อมาคือ การออกแบบเส้นทางเดินของข้อมูลให้เชื่อมต่อกันอาจจะเชื่อมต่อกันด้วยระบบบัสหนึ่งหรือสองทิศทาง (Unidirectional or Bidirrectional Bus) เพื่อเชื่อมต่อริจิสเตอร์ (Registor) และวงจรตรรก (Logic) ให้เป็นระบบที่สมบูรณ์ดังรูปที่ 3.2



รูปที่ 3.2 การออกแบบระบบเส้นทางของข้อมูล

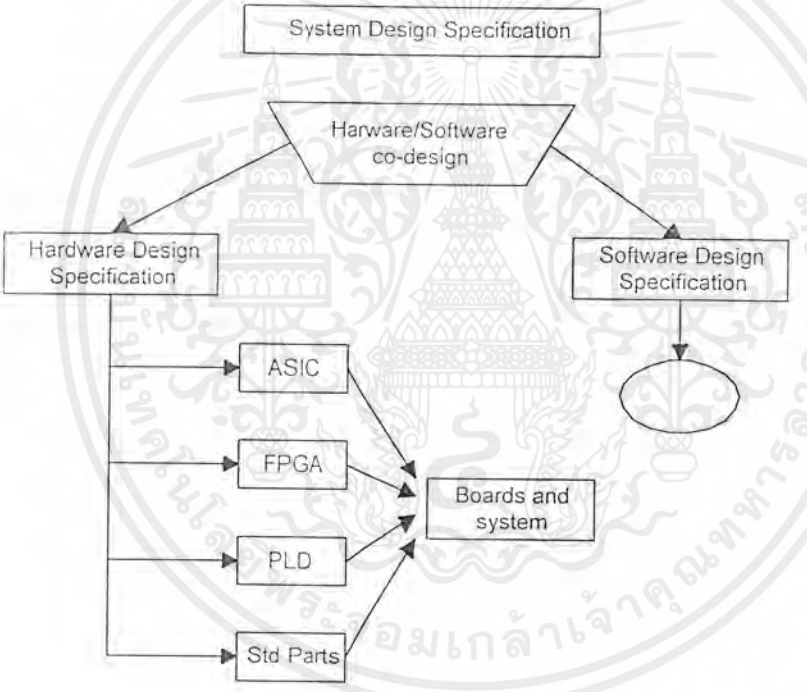
จากนั้นจะเป็นขั้นตอนของการเชื่อมโยงเกทพื้นฐานและฟลิปฟลอป จนมาถึงขั้นตอนการออกแบบทางกายภาพ คือเป็นการทำทรานซิสเตอร์ลิส และการเลเอาท์ (Transistor List and Layout) จนกระทั่งเป็นขั้นตอนการผลิตลงบน Chip ซึ่งอาจเป็นอุปกรณ์ FPGAs, PLD, หรือ Std Port หรืออาจจะเป็นการส่งระบบที่ออกแบบสมบูรณ์และไปทำการเจือสารที่โรงงานผลิตออกมาเป็นวงจรรวม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2 แนะนำภาษา VHDL

VHDL เป็นภาษาบรรยายทางฮาร์ดแวร์สำหรับการออกแบบในระดับสูงทางอิเล็กทรอนิกส์ (Electronics) มีความสามารถในการเลียนแบบ (Simulation), การสังเคราะห์ (Synthesis), และการทดสอบ (Testing) ในการจำลองระบบอิเล็กทรอนิกส์ดิจิตอล ลักษณะการทำงานของ VHDL จะทำงานไปพร้อมๆ กัน (Concurrent) และเป็นลำดับ (Sequential Statements) ซึ่งหมายถึงทุกๆ คำสั่ง องค์ประกอบเกท หรือวงจรตรรกจะนำมาปฏิบัติทั้งหมด จึงดูเหมือนว่าได้มีการปฏิบัติไปพร้อมๆ กัน

3.3 VHDL กับการออกแบบอิเล็กทรอนิกส์

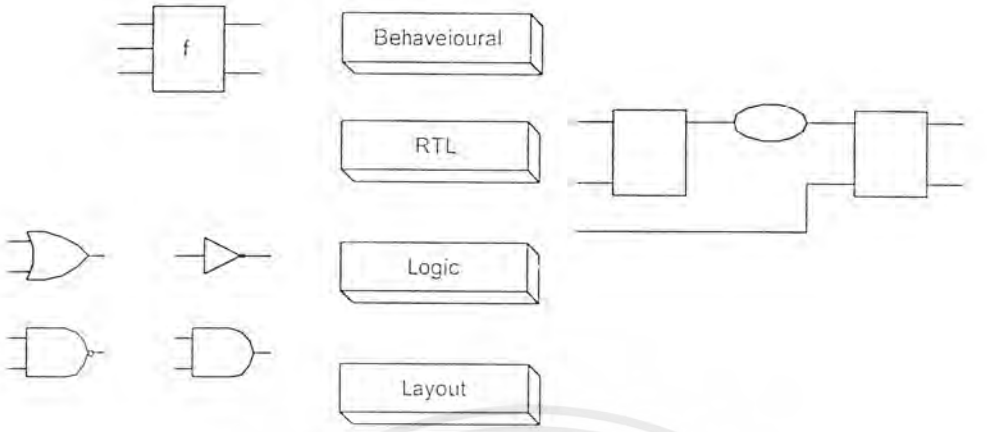


รูปที่ 3.3 การออกแบบระบบอิเล็กทรอนิกส์

จากรูปที่ 3.3 แสดงกระบวนการการทำงานของภาษา VHDL จากการออกแบบทางฮาร์ดแวร์ผ่านกระบวนการแปลภาษา (Compile) การจำลองการทำงาน (Simulate) เพื่อตรวจสอบดูว่าผลการทำงานเป็นไปตามจุดประสงค์หรือไม่ จากนั้นนำไปสังเคราะห์ เพื่อให้ได้เป็นวงจรแสดงการเชื่อมต่อซึ่งประกอบด้วยเกท (Gate), ฟลิปฟลอป (Flip-Flop), และอุปกรณ์พื้นฐานต่างๆ แล้วนำไปโปรแกรมลงบนอุปกรณ์ประเภท ASIC, FPGA, PLD หรือ Std part จนเป็นวงจรรวมแบบสมบูรณ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.3.1 ขั้นตอนในการออกแบบ



รูปที่ 3.4 ขั้นตอนการออกแบบ

1. Behavioral เป็นขั้นตอนในการสร้างแบบจำลองการบรรยายพฤติกรรมในขั้นตอนแรก
2. RTL (Register Transfer Logic) การบรรยายการถ่ายโอน Logic
3. Logic เป็นขั้นตอนการสังเคราะห์ให้ได้เป็นวงจรแสดงการเชื่อมต่อซึ่งประกอบด้วยเกทต่างๆ
4. Layout เป็นขั้นตอนของการเจือสาร

ในส่วนของภาษา VHDL ครอบคลุมในขั้นตอน Behavioral, RTL, และ Logic

วัตถุประสงค์ในขั้นตอนของ Behavioral มาถึงขั้น RTL ก็เพื่อ

1. ช่วยในการกำหนดการทำงานและการแก้ไขจุดผิดพลาดของโปรแกรม
2. ช่วยในการวิเคราะห์ปัญหาของโปรแกรม
3. เป็นวิธีในการระบุรายละเอียดและการแยกส่วนการทำงานของโปรแกรม
4. เป็นการตรวจสอบและเปรียบเทียบผลลัพธ์ของโปรแกรม

3.3.2 ข้อได้เปรียบของการใช้ VHDL

1. การออกแบบมีคุณภาพที่สูงกว่า
2. ออกแบบวงจรที่มีความซับซ้อนมากๆ ได้
3. ใช้เวลาในการออกแบบสั้นกว่า
4. ค้นหาข้อผิดพลาดและเปลี่ยนแปลงแก้ไขได้ง่าย

ภาษา VHDL สามารถใช้ในการบรรยายได้ 3 ลักษณะคือ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

3.4 การบรรยายเชิงพฤติกรรม

เป็นการบรรยายลักษณะการทำงานของอุปกรณ์ฮาร์ดแวร์ในเชิงพฤติกรรม เป็นการบรรยายลักษณะการเปลี่ยนแปลงของข้อมูลในรูปแบบของอัลกอริธึม สำหรับการคำนวณผลลัพธ์ที่เกิดขึ้นสืบเนื่องมาจากการเปลี่ยนแปลงสภาวะของข้อมูลที่เข้ามา โดยไม่คำนึงถึงว่าลักษณะโครงสร้างหรือความสัมพันธ์ของอุปกรณ์ที่อยู่ภายใน

3.5 การบรรยายเชิงข้อมูล

เป็นการบรรยายถึงการเคลื่อนไหลของข้อมูลผ่านรีจิสเตอร์ และบัสของระบบเป็นระดับขั้นของการบรรยายที่อยู่ตรงกลางระหว่างการบรรยายเชิงพฤติกรรม และการบรรยายเชิงโครงสร้างเครื่องมือที่ใช้ในการควบคุมการไหลเลื่อน ได้แก่ Conditional, Selected และ Guarded

3.6 การบรรยายเชิงโครงสร้าง

เป็นการบรรยายการทำงานของระบบในเชิงโครงสร้างจะต้องแสดงรายการของอุปกรณ์ทั้งหมดที่ใช้ในระบบ และต้องกำหนดการเชื่อมต่อระหว่างอุปกรณ์ต่างๆ ด้วยเพราะว่าการบรรยายในระดับนี้เป็นการบรรยายที่ใกล้เคียงลักษณะของฮาร์ดแวร์มากที่สุด ภาษา VHDL ได้จัดเตรียมเครื่องมือ และลักษณะโครงสร้างของการบรรยายในลักษณะนี้ไว้ที่สำคัญ 4 ลักษณะคือ

1. ความสามารถในการเลือกหรือกำหนดอุปกรณ์ที่ต้องการได้จากไลบรารี่
2. การสร้างไลบรารี่เพื่อเก็บอุปกรณ์ที่ผู้ใช้ออกแบบไว้เองได้
3. กลไกในการเชื่อมต่อระหว่างอุปกรณ์
4. โครงสร้างของการกำหนดอุปกรณ์ชนิดเดียวกันซ้ำๆกัน

3.7 ส่วนประกอบหลักของภาษา VHDL

3.7.1 โครงสร้าง

จากเอกสารของ DOD (The United State Department of Defense) ได้กล่าวไว้ว่าภาษาบรรยายทางฮาร์ดแวร์ เป็นภาษาที่ต้องการกำหนดรูปแบบทางโครงสร้างเพราะการกำหนดโครงสร้างนี้จะช่วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้
36944

ในการออกแบบ ทั้งระบบที่ง่ายไปจนถึงระบบที่ซับซ้อน ซึ่งระบบที่ว่านี้ถือเป็นมาตรฐานของภาษาบรรยายทางฮาร์ดแวร์

3.7.2 การทำงานที่พร้อมกัน

ในวงจรอิเล็กทรอนิกส์ อุปกรณ์ทุกๆ ตัวจะอยู่ในสภาพเตรียมพร้อมเสมอและจะมีเรื่องของเวลาเข้ามาควบคุมการทำงานเสมอ ภาษา VHDL ได้รับการออกแบบมาเพื่อบรรยาย รูปแบบทางดิจิตอล และการป้องกันของเวลา สำหรับการทำงานของอุปกรณ์ได้อย่างถูกต้อง การบรรยายการทำงานที่อยู่ภายในส่วนของสถาปัตยกรรม (Architecture) จะมีการทำงานที่พร้อมเพรียงกันเสมอ หรือแม้แต่โปรเซสซึ่งมีการทำงานภายในเป็นแบบลำดับคำสั่งก็ตาม หากมีหลายโปรเซสอยู่ภายในโครงสร้างเดียวกันทุกโปรเซสจะทำงานไปพร้อมกันด้วย

3.7.3 ลำดับคำสั่ง

ถึงแม้ว่าลักษณะเฉพาะของภาษาบรรยายทางฮาร์ดแวร์ จะสนับสนุนการปฏิบัติตามคำสั่งอย่างเป็นลำดับเป็นกระบวนการ ตัวภาษา VHDL ก็ยังได้จัดเตรียมลักษณะการควบคุมแบบลำดับคำสั่งเอาไว้ เมื่อนักออกแบบได้บรรยายหน้าที่การทำงานซึ่งเป็นรายละเอียดในของแต่ละองค์ประกอบได้ในลักษณะเดียวกับการเขียนโปรแกรม ที่ประกอบด้วยโครงสร้างแบบ if-then-else และ loop การบรรยายแบบลำดับคำสั่ง ทำให้การออกแบบหน้าที่การทำงานของอุปกรณ์กระทำได้สะดวกและง่ายขึ้น แต่อย่างไรก็ตามการทำงานของภาษา VHDL ก็ยังเป็นการทำงานแบบพร้อมเพรียงกันอยู่

3.7.4 เวลาที่ใช้ในการควบคุม

ภาษา VHDL เป็นภาษาที่อนุญาตให้ผู้ออกแบบ สามารถกำหนดเวลาในการส่งผ่านข้อมูลหรือสัญญาณได้ การตรวจสอบการออกแบบเกท หรือการหน่วงเวลาสามารถกระทำได้โดยกำหนดช่วงเวลาที่แน่นอนหรือกำหนดให้มีการรอคอยเหตุการณ์ นอกจากนี้ก็ยังสามารถกำหนดรูปแบบของสัญญาณนาฬิกาได้

3.8 องค์ประกอบในการเขียนภาษา VHDL

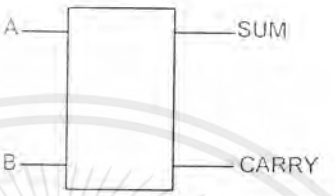
สามารถแบ่งองค์ประกอบในการเขียนภาษา VHDL ได้ Entity, Architecture, Process, Configuration, Package, และ Library

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.8.1Entity

เป็นส่วนที่ใช้ในการประกาศ อธิบายการเชื่อมต่อการติดต่อกับองค์ประกอบภายนอก แต่ยังมิได้กำหนดการกระทำใดๆ ทั้งสิ้น ซึ่งในส่วนประกอบนี้จะต้องทำกำหนดชื่อ (ห้ามซ้ำกับชื่อสงวนของภาษา VHDL) แล้วตามด้วย is ตัวอักษรในภาษา VHDL ทั้งตัวใหญ่ตัวเล็กมีค่าท่ากันดังตัวอย่างแสดงในรูปที่ 3.5

```
entity HALFADD is
port (A,B:in bit;
      SUM,CARRY:out bit);
end HALFADD;
```



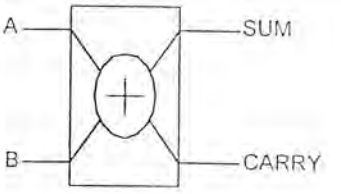
รูปที่ 3.5 โปรแกรมการใช้งานและลักษณะการติดต่อใช้งานของ entity

จากรูปที่ 3.5 จะเห็นว่ามีการประกาศชื่อ entity HALFADD ตาม is แล้วมีการกำหนดสัญญาณอินพุทและเอาท์พุท โดยมีวงเล็บเปิดและวงเล็บปิด ในแต่ละบรรทัดจะถูกปิดท้ายด้วย (;) และใช้ในการกำหนดสัญญาณให้กับตัวแปรและจบ entity ด้วยคำสั่ง end ตามด้วยชื่อของ entity

3.8.2 Architecture

เป็นส่วนที่ใช้การกำหนดการทำงานภายในของ entity ต้องเป็นการเชื่อมโยงกันของ entity จำเพาะซึ่ง entity เดียวสามารถมีได้หลาย Architecture จะต้องมีการกำหนดชื่อของ Architecture นั้นๆ ด้วยว่าเป็นของ entity ไหน การเขียน Architecture จะตามหลัง begin เสมอ และจบด้วยคำสั่ง end ตามด้วยชื่อ Architecture นั้น ดังตัวอย่างแสดงในรูปที่ 3.6

```
architecture BEHAVE of HALFADD is
begin
   SUM <= A xor B;
   CARRY <= A and B;
end BEHAVE;
```



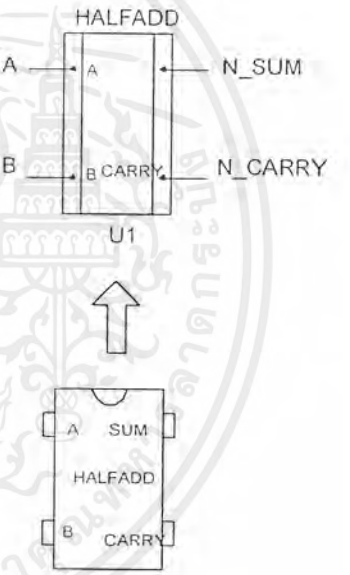
รูปที่ 3.6 โปรแกรมการใช้งานและลักษณะการติดต่อใช้งานของ Architecture

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.8.3 Configuration

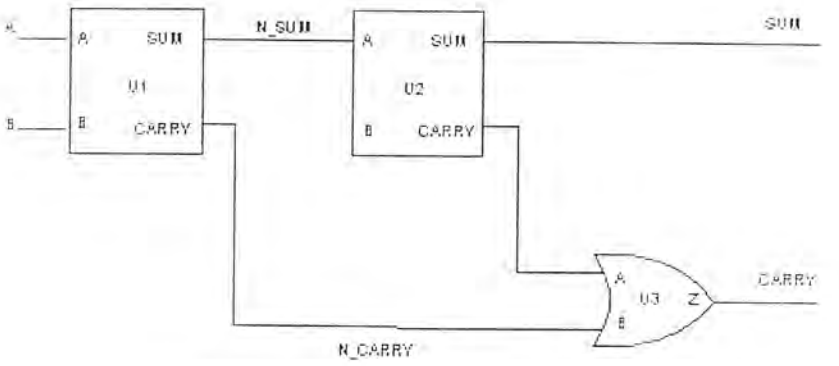
ในภาษา VHDL สามารถสร้างอุปกรณ์เก็บเอาไว้ได้ และสามารถจำลอง Socket ของอุปกรณ์แต่ละตัว เพื่อความสะดวกเหมือนกับการใช้งานจริงในการเชื่อมต่ออุปกรณ์ดังรูปที่ 3.7 ul ถูกสร้างขึ้นเป็น Socket ให้กับอุปกรณ์ โดยให้กับอุปกรณ์ โดยการนำไปใช้งานนั้นจะอาศัย Component ตามด้วยชื่อของอุปกรณ์มีการกำหนดสัญญาณอินพุทเป็นแบบบิทให้กับตัวแปร A และ B ส่วนเอาท์พุทก็เป็นสัญญาณแบบบิทที่กำหนดค่าให้กับตัวแปร SUM และ CARRY เหมือนกับ entity HALFADD ทุกประการ และจบด้วยคำสั่ง end component; สำหรับการเชื่อมต่อจะใช้คำสั่ง port map โดยเป็นการให้กับทางขวามือ คือ ul: HALFADD port map (A, B, N_SUM,N_CARRY); โดยการเขียนนี้ต้องอยู่หลังจาก begin ดังตัวอย่างแสดงในรูปที่ 3.8 และรูปที่ 3.9

```
component HALFADD
  port (A,B : in bit;
        SUM,CARRY : out bit);
  end component;
  ...
begin
  ul:HALFADD port map (A,B,N_SUM,N_CARRY);
  ...

DEFAULT
CONFIGURATION

entity HALFADD is
port (A,B : in bit;
      SUM,CARRY : out bit);
end HALFADD;
```



รูปที่ 3.7 โปรแกรมและการติดต่อใช้งานของ Configuration

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.8 การเชื่อมต่อของอุปกรณ์ภายใน

```
entity FULLADD is
port (A,B,CIN :in bit);
      SUM,CARRY : out bit);
end FULLADD;
architecture STRUCTURAL of FULLADD is
       signal N_SUM,N_CARRY1,N_CARRY2 : bit;
       component HALFADD
       port (A,B : in bit;
             SUM,CARRY: out bit);
       end component;
begin
       u1 : HALFADD port map (A,B,N_SUM,N_CARRY1);
       u2 : HALFADD port map (N_SUM,CIN,SUM,N_CARRY2);
       u3 : ORGATE port map (N_ CARRY2,N_CARRY1,CARRY);
end STRUCTURAL;
```

รูปที่ 3.9 โปรแกรมการเชื่อมต่ออุปกรณ์ภายในและการประกาศสายสัญญาณ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากรูปที่ 3.9 มีการประกาศสัญญาณภายในอุปกรณ์คือ Signal N_CARRY1, N_CARRY2 ให้มีสัญญาณเป็นแบบบิท จะสังเกตเห็นว่าวงจรมีอุปกรณ์ภายใน 3 ตัว คือ HALFADD 2 ตัวและ ORGATE 1 ตัวจึงประกอบด้วย u1, u2, u3 และสัญญาณที่เขียนใน port map จะเรียงสัญญาณเข้าจนถึงสัญญาณออกตามลำดับและจบโปรแกรมด้วยคำสั่ง end STRUCTURAL ตามชื่อของArchitecture เสมอส่วนของ component เป็นส่วนหนึ่งของ Architecture

3.8.4 Process

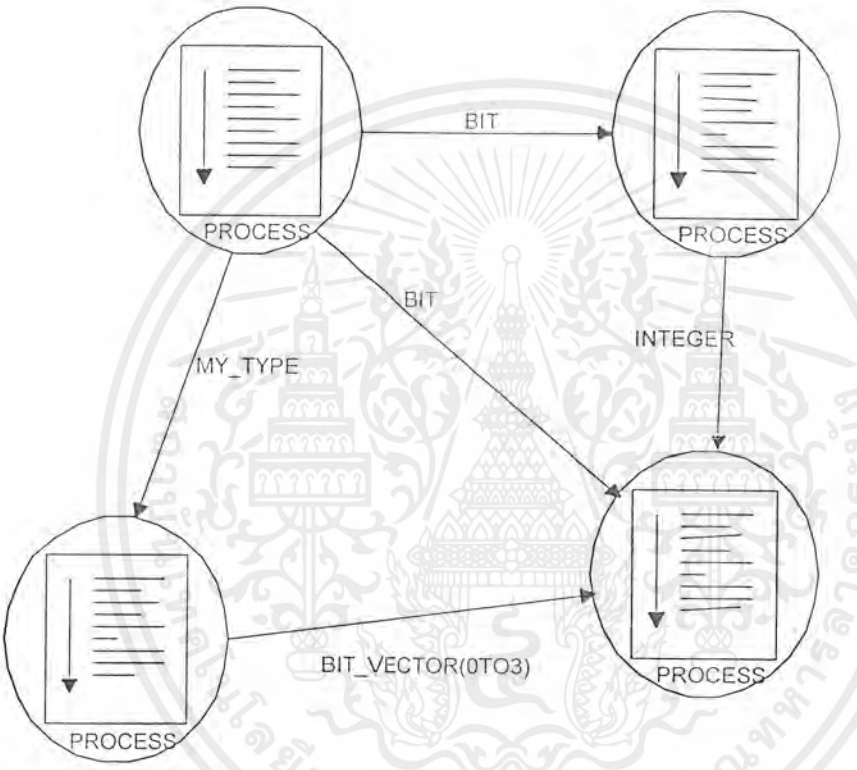
ส่วนนี้เป็นการบรรจุคำสั่งลำดับที่ต้องเขียนอยู่ภายใน Architecture ซึ่งโปรเซสหลายๆโปรเซสจะมีการทำงานที่พร้อมกัน (concurrent) ตัวแปรทางขวามือจะถูกกำหนดค่าให้จากสัญญาณทางซ้ายมือ (<=) ในการเริ่มโปรแกรมใช้คำสั่ง PROCESS และจบด้วยคำสั่ง END PROCESSส่วนประกอบของการบรรยายโปรเซสประกอบด้วยส่วนของประกาศตัวแปรที่ต้องใช้และส่วนของการปฏิบัติตามคำสั่งเพื่อให้ได้ผลลัพธ์ตามที่ต้องการ ดังตัวอย่างที่ต้องการในรูปที่ 3.10

```
entity AND_OR is
        port (A,B :in bit ;
              Z_OR,Z_AND :out bit);
end AND_OR;
architecture HEHAVE of AND_OR is
begin
        AND_OR_FUNC: process(A,B)
        Begin
                if (A='1' orB='1') then
                        Z_OR <='1';
                Else
                        Z_OR <='0';
                End if;
                If(A='1' and B='1') then
                        Z_AND <='1';
                Else
                        Z_AND <='0';
                End if;
        end process AND_OR_FUNC ;
end BEHAVE ;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 3.10 โปรแกรมการทำงานของโปรเซส

จากรูปที่ 3.10 ชื่อของ Process ชื่อของโปรเซสคือ AND_OR_FUNC มีสัญญาณอินพุทเป็น A และ B โดยใช้คำสั่ง if-then-else ในการลำดับการทำงานดังนี้ ถ้า A=1 หรือ B=1 จะให้ Z_OR มีค่าเป็น 1 เงื่อนไขนอกจากนั้น Z_OR จะมีค่าเป็น 0 จบการทำงานด้วย end if ; และถ้า A=1 และ B=1 ดังนั้น Z_AND จะมีค่าเป็น 1 เงื่อนไขนอกจากนั้น Z_AND จะมีค่าเป็น 0 และจบ Process ด้วย end process AND_OR_FUNC ; และจบ Architecture ด้วยคำสั่ง end BEHAVE;



รูปที่ 3.11 การเชื่อมต่อหลายๆ process ที่สามารถเชื่อมต่อกันได้ด้วยสัญญาณที่ต่างกัน

3.8.5 Package

เป็นส่วนของโปรแกรมที่แปล (compile) เรียบร้อยแล้ว และดึงเอามาใช้เท่านั้น หรืออาจกล่าวได้ว่า package คือกลุ่มชนิดของข้อมูล โปรแกรมย่อยหรืออุปกรณ์ต่างๆ ที่ได้ออกแบบไว้แล้วนำมารวบรวมไว้เป็นกลุ่มๆ อยู่ภายในเพื่อให้ผู้ออกแบบเรียกใช้ได้สะดวกดังรูปที่ 3.12

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

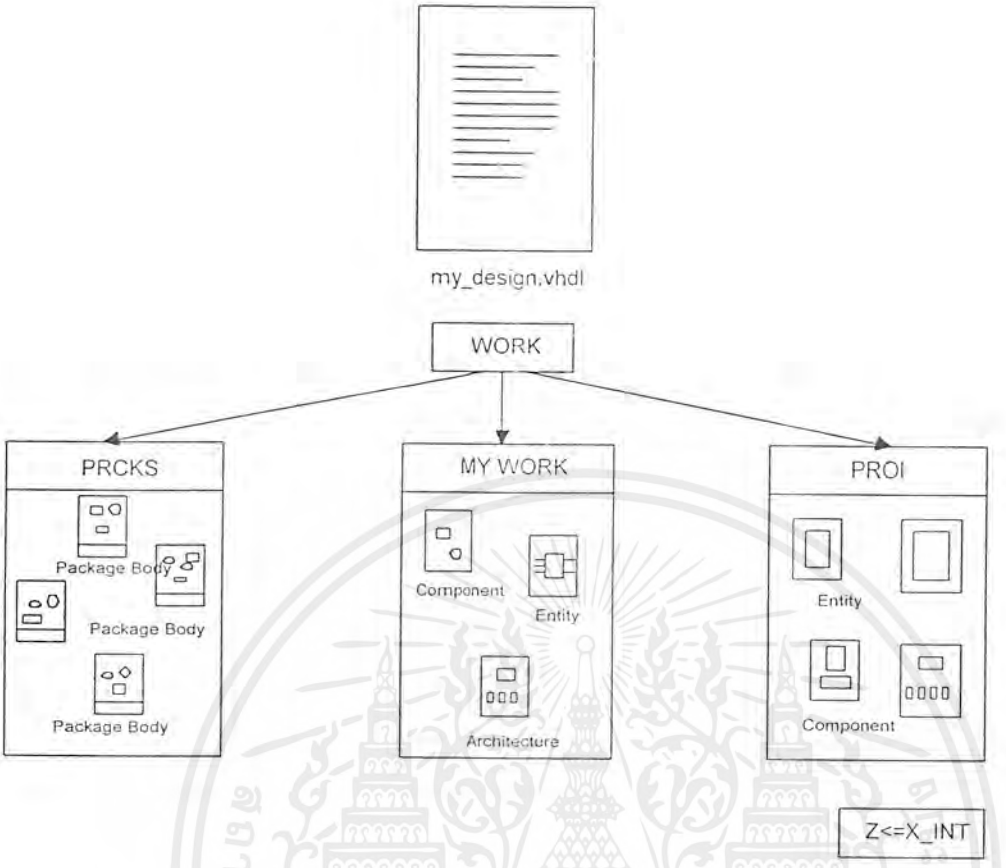
```
Package DEMO_PACK is
    - constants
    - data types
    - components
    -subprogram
end DEMO_PACK
package body DEMO_PACK is
    - constant values
    - subprogram definition
end DEMO_PCCK;
```

รูปที่ 3.12 โครงสร้างของ Package

3.8.6 Library

เป็นส่วนของการเก็บโปรแกรมต่างๆ ที่สมบูรณ์ซึ่งภายใน library จะประกอบด้วย entity, architecture, package, configulation จะมีลักษณะการใช้งานดังรูปที่ 3.13

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.13 ส่วนของการเก็บโปรแกรมต่างๆ ที่สมบูรณ์

จากรูปที่ 3.13 ก่อนการเรียกใช้ work จะต้องทำการแปล (compile) ที่ packs, mywork และ proj ก่อน แล้วจึงสามารถเรียกใช้ได้

3.9 สัญญาณและชนิดของข้อมูลในภาษา VHDL

สัญญาณมีลักษณะเป็นเหมือนตัวกลางฮาร์ดแวร์ที่ใช้ในการส่งผ่านข้อมูลและมีเรื่องของเวลาเข้ามาด้วย การกำหนดค่าให้สัญญาณจะใช้สัญลักษณ์ <= ในการส่งค่าและสามารถใช้คำสั่ง AFTER เพื่อกำหนดช่วงเวลาในการส่งผ่านค่าของสัญญาณเช่น w <= AFTER 12 nS หมายถึงการกำหนดค่าของสัญญาณ a ให้กับ w หลังจากเวลาผ่านไป 12 nS ในทางตรงกันข้ามตัวแปรมีลักษณะเป็นเสมือนตัวกลางซอฟแวร์ที่ใช้ในการส่งผ่านข้อมูลและไม่มีเรื่องของเวลาเข้ามาเกี่ยวข้องด้วย การกำหนดค่าให้กับ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตัวแปรจะใช้สัญลักษณ์ := ตัวแปรจะถูกใช้ในส่วนที่มีการทำงานเป็นแบบลำดับคำสั่งในฟังก์ชัน โพรซีเจอร์ และโพรเซส

3.9.1 ลักษณะข้อมูลมาตรฐาน

1. กลุ่มของข้อมูลชนิดต่างๆต้องสามารถนำมาประเมินค่าได้
2. กลุ่มของข้อมูลมาตรฐานสามารถให้คำจำกัดความได้
3. ผู้ใช้สามารถที่จะนำข้อมูลเหล่านี้ไปใช้ได้ถูกต้อง ต้องดูที่คำจำกัดความของข้อมูลชนิดนั้น เพื่อไม่ให้เกิดความผิดพลาด

3.9.2 ชนิดของข้อมูลมาตรฐาน

1. FALSE, TRUE นำไปใช้ในสมการ BOOLEAN
2. BIT เช่น "0", "1"
3. BITVECTOR เช่น "10001", "10100101"
4. CHARACTER เช่น "A", "C", "R", "2" " ∃"
5. ทศนิยม เช่น "1.02", "0.436", "1.0e-10", "-4.356e45"
6. กำหนดเป็นข้อความ เช่น "% Error 56.3", "message string"
7. แบบจำนวนเต็ม เช่น "1", "254", "-456"
8. แบบกำหนดค่าเวลา เช่น "40 nS", "5.5 pS"

ข้อมูลเหล่านี้จะถูกเก็บอยู่ใน package ดังรูปที่ 3.14

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
package  STANDARD is
        type BOOLEAN is (FALSE,TRUE);
        type BIT is ('0','1');
        type CHARRACTER is  (- ascii set);
        type INTEGER is range
                        implementation_defined;
        type REAL is range
                        implementation_defined;
        BIT_VECTOR,STRING,TIME
End STANDARD;
```

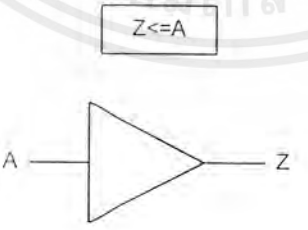
รูปที่ 3.14 การเขียนโปรแกรมของข้อมูลภายใน package

3.10 สัญญาณและการกระตุ้น (signal and driver)

```
signal A,B,Z : bit;
signal X_INT : interer;
```

รูปที่ 3.15 การกระตุ้นสัญญาณให้ A, B, Z

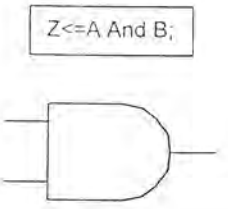
จากรูปที่ 3.15 เป็นการกระตุ้นสัญญาณให้ A, B, Z เป็นสัญญาณแบบ Bit, X_INT เป็น integer



รูปที่ 3.16 การส่งค่า A ให้ Z

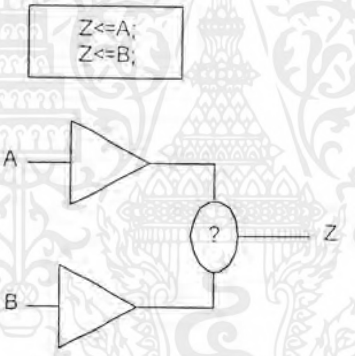
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากรูปที่ 3.16 ตัวกระทำทางด้านขวาคือ A ส่งค่าให้ Z



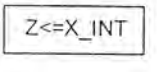
รูปที่ 3.17 การให้ค่า A And B แก่ Z

จากรูปที่ 3.17 เป็นการให้ค่าทางขวามือคือ A And B แก่ Z และยังมีรูปแบบใช้ไม่ได้ เช่น



รูปที่ 3.18 การส่งค่า A และ B ที่ไม่สามารถส่งให้แก่ Z ได้

จากรูปที่ 3.18 ค่า A และ B ไม่สามารถส่งให้แก่ Z ได้ เพราะอยู่ในสภาวะที่ไม่สามารถตัดสินใจได้ หรือรูปแบบในการกำหนดข้อมูลคนละชนิดให้แก่ตัวแปรก็ไม่สามารถทำได้



รูปที่ 3.19 การส่งค่า X_INT ให้แก่ Z

จากรูปที่ 3.19 Z ถูกกำหนดให้มีค่าเป็น Bit ส่วน X_INT ถูกกำหนด integer

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.11 ข้อมูลแบบเป็นชุด

เป็นข้อมูลเหมือนกับข้อมูลทั่วไปเพียงแต่มีลักษณะของข้อมูลเป็นชุด ซึ่งข้อมูลจะถูกกำหนดค่าข้อมูลเรียงตามลำดับ ดังตัวอย่างในรูปที่ 3.20

```
signal A_BUS,B_BUS,Z_BUS : bit_vector (3 downto 0)
```

รูปที่ 3.20 การกำหนดค่าข้อมูลแบบเป็นชุด

จากรูปที่ 3.20 เป็นการกำหนดค่า bit_vector 4 ค่า คือจาก 0-3 โดยค่าถูกกำหนดจาก 3, 2, 1,0

```
Z_BUS <= A_BUS;
```

```
Z_BUS(3) <= A_BUS(3)
Z_BUS(2) <= A_BUS(2)
Z_BUS(1) <= A_BUS(1)
Z_BUS(0) <= A_BUS(0)
```

รูปที่ 3.21 การให้ค่า A_BUS แก่ Z-BUS

จากรูปที่ 3.21 ค่าจะเรียงตามลำดับ Z_BUS(3) ถูกกำหนดค่าให้จาก A_BUS(3) ตามลำดับ

```
Z_BUS(3) <= A_BUS(0)
Z_BUS(2) <= A_BUS(1)
Z_BUS(1) <= A_BUS(2)
Z_BUS(0) <= A_BUS(3)
```

รูปที่ 3.22 การกำหนดค่า A_BUS(0) ให้แก่ Z_BUS(3)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากรูปที่ 3.22 เป็นการกำหนดค่า Z_BUS(3) ถูกกำหนดค่าให้จาก A_BUS(0)

สิ่งที่ต้องคำนึงเมื่อให้ข้อมูลแบบเป็นชุด คือ ขนาดของ Array ข้างซ้ายและข้างขวาจะต้องเท่ากัน และสมาชิกแต่ละตัวจะถูกกำหนดไว้โดยตำแหน่ง ดังรูปที่ 3.23

```
signal A_BUS,B_BUS,Z_BUS : bit_vector (3 downto 0);
signal A_BIT,B_BIT,C_BIT,D_BIT : bit;
signal BYTE : bit_vector (7 downto 0);
```

```
Z_BUS <= (A_BIT,B_BIT,C_BIT,D_BIT);
```

```
BYTE <= (7 => '1',5 downto 1 => '1',6 => B_BIT,others => '0');
```

รูปที่ 3.23 แสดงข้อมูลเป็นกลุ่ม (Aggregates)

3.12 ตัวดำเนินการภาษา VHDL (Operators)

การบรรยายเชิงพฤติกรรมในภาษา VHDL ก็มีตัวกระทำทางลอจิกและคณิตศาสตร์เช่นเดียวกับภาษาซอฟแวร์ทั่วไปดังตารางที่ 3.1

ตารางที่ 3.1 ตัวดำเนินการภาษา VHDL

PREDEFINED OPERATORS
LOGICAL OPERATORS : NOT AND OR NAND NOR XOR
OPERAND TYPE : BIT BOOLEAN
RESULT TYPE : BIT BOOLEAN
RELATIONAL OPERATORS : = / = < <= > >=
OPERAND TYPE : any type
RESULT TYPE : Boolean
ARITMETIC OPERATORS : + - * / MOD REN ABS

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

OPERAND TYPE : INTEGER REAL Physical

RESULT TYPE : INTEGER REAL Physical

CONCENTENATION OPERATORS : &

OPERAND TYPE : array of any type

RESULT TYPE : array of any type

3.13 คำสั่งการทำซ้ำ (Sequential Statements)

3.13.1คำสั่ง If-then-else

```
if CONDITION then
  - sequential statements
end if;
```

```
if CONDITION then
  - sequential statements
else
  - sequential statements
end if;
```

```
if CONDITION then
  - sequential statements
elsif CONDITION then
  - sequential statements
elsif CONDITION then
  - sequential statements
else
  - sequential statements
end if;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 3.24 รูปแบบของคำสั่ง if-then-else

```
brary IEEE;
use IEEE.Std_Logic_1164.all;
entity IF_EXAMPLE is
port (A,B,C,X : in  std_ulogic_vector(3 downto o);
      Z       : out std_ulogic_vector(3 downto o);
end IF_EXAMPLE is
begin
   process (A,B,C,X)
   begin
        if (x = "0000") then
            Z <= A;
        Elsif (X <= "0101") then
            Z <= B;
        Else
            Z <= C;
        End if
   End process;
end A;
```

รูปที่ 3.25 ตัวอย่างการใช้คำสั่ง If-then-else

จากรูปแบบของคำสั่ง If-then-else ในรูปที่ 3.24 นำมาเขียนตัวอย่างดังรูปที่ 3.25 คำสั่ง If จะกระทำภายใน Architecture และอยู่ภายใน Process โดยที่คำสั่ง If 2 สเตจ โดยเริ่มจากคำสั่ง If ตามด้วย else if และ else จบคำสั่ง if ที่ end if

3.13.2 คำสั่ง case

การใช้คำสั่ง case ต้องใช้ให้ครบของความเป็นไปได้ของ expression และต้องเรียงจากน้อยสุดไปหามากสุด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
case expression is
    when VALUE_2 | VALUE_4 =>
        sequential statements
    when VALUE_M to VALUE_N =>
        sequential statements
    when others =>
        sequential statements
end case;
```

รูปที่ 3.26 รูปแบบของคำสั่ง Case

```
entity CASE_EXAMPLE is
port (A,B,X :in integer range 0 to 15;
      Z     out integer range 0 to 15;
end CASE_EXAMPLE;
Architecture A of CASE_EXAMPLE is
Begin
  Process (A,B,X)
  Begin
      case X is
        when 0 to 4 =>
          Z <= B;
        when 7 | 9 =>
          Z <= A;
        when others =>
          X <= 0;
      end case;
  end process
end A;
```

รูปที่ 3.27 ตัวอย่างการใช้คำสั่ง Case

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากรูปแบบคำสั่ง Case ในรูปที่ 3.26 นำมาเขียนตัวอย่างได้ดังรูปที่ 3.27 case จะกำหนดค่าเป็น x ให้ตัวแปร B เป็นค่า 0-4 ให้ตัวแปร C เป็นค่า 5 ตัวแปร A เป็นค่า 7,9 และถ้าเป็นค่าอื่น Z จะมีค่าเป็น 0 จบด้วย end case;

3.13.3 คำสั่ง For

```
for loop_parameter in discrete_range loop
    - sequential statements
end loop;
```

รูปที่ 3.28 รูปแบบของคำสั่ง For

```
Entity  FOR_LOOP is
Port (A : in integer range 0 to 3;
    Z : out std_ulogic_vector (3 downto 0));
End FOR_LOOP;
Achitecture A of FOR_LOOP is
Begin
  process (a)
  begin
    Z <= "0000";
    for I in 0 to 3 loop
      if (A = I) then
        Z(I) <= '1';
      End if;
    end loop;
  end process;
end A;
```

รูปที่ 3.29 ตัวอย่างการใช้คำสั่ง For

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

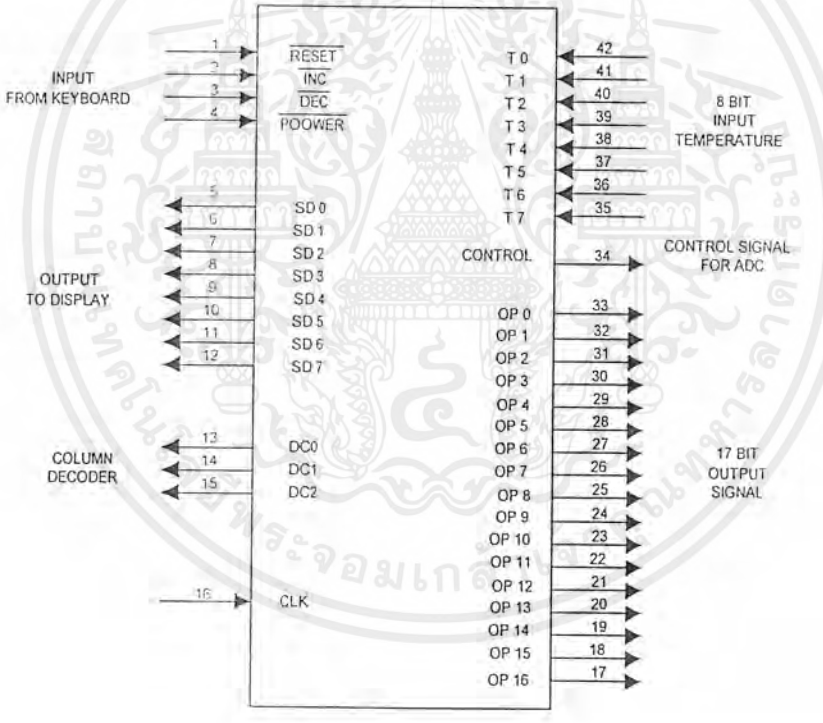
จากรูปแบบของคำสั่ง For ดังรูปที่ 3.28 นำมาเขียนตัวอย่างได้ดังรูปที่ 3.29 คำสั่ง for จะถูกกำหนดให้วนการทำงานใน loop 3 ครั้ง และจบด้วย end loop ซึ่งคำสั่ง for loop นี้ก็อยู่ในกระบวนการของ process

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4
การออกแบบทางด้าน HARDWARE

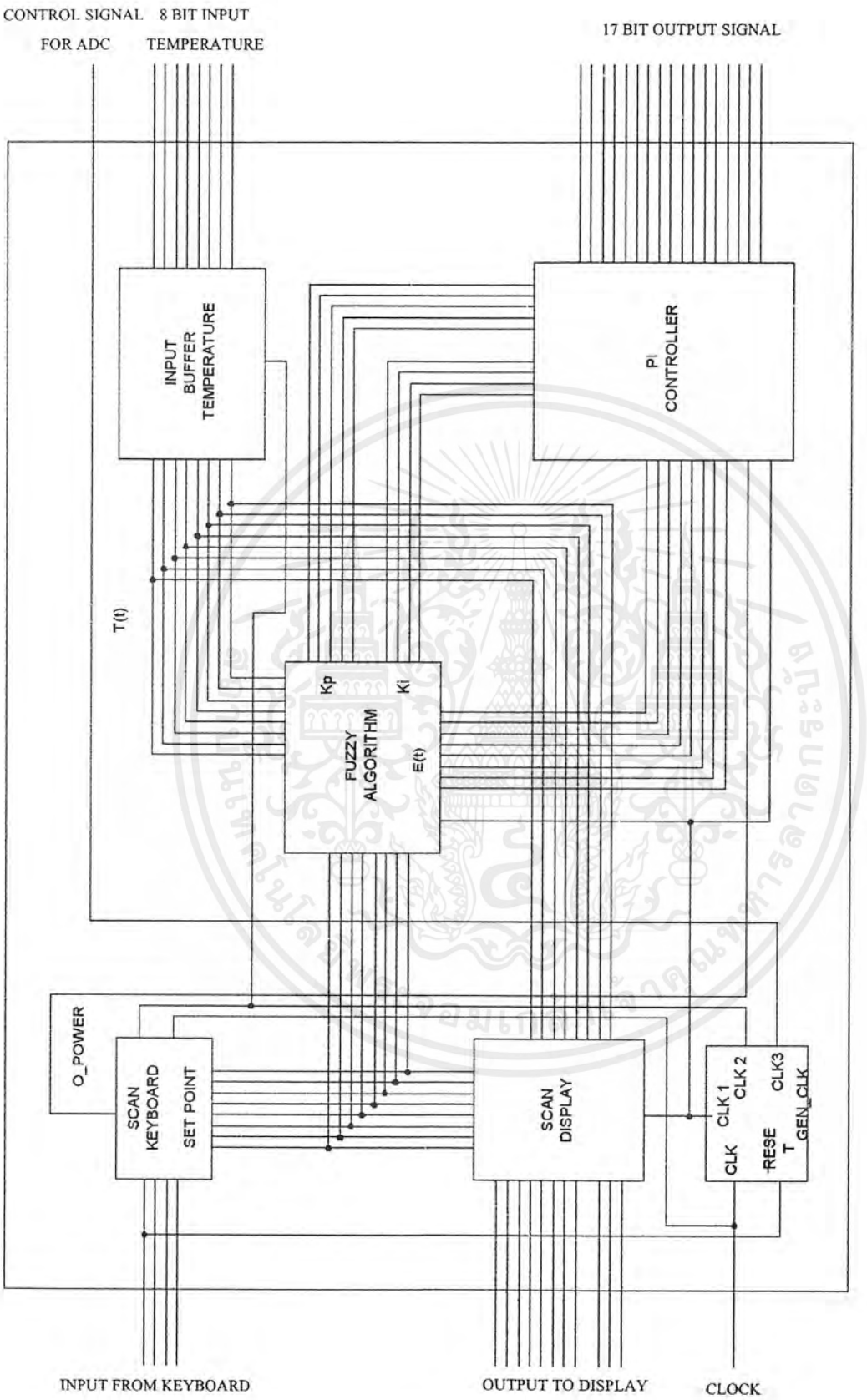
หลังจากที่ได้ศึกษาถึงวิธีการคำนวณหาเงื่อนไข และวิธีใช้โปรแกรมภาษา VHDL ในเบื้องต้นแล้ว ต่อไปสิ่งที่ต้องทำก็คือ การออกแบบ Hardware สำหรับวงจรที่เราต้องการในเบื้องต้นมีดังนี้

1. รับอินพุทจากมนุษย์ในการตั้งค่าอุณหภูมิ
2. รับอินพุทจากเซนเซอร์อุณหภูมิที่เป็นข้อมูลดิจิตอลแล้ว
3. ส่วนแสดงผล อุณหภูมิปัจจุบันและอุณหภูมิที่ตั้งไว้
4. ส่วนคำนวณแบบ Fuzzy logic
5. สัญญาณควบคุมทางด้านเอาท์พุท



รูปที่ 4.1 แสดงวงจรที่ต้องการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.2 แสดงรายละเอียดของระบบที่ต้องการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

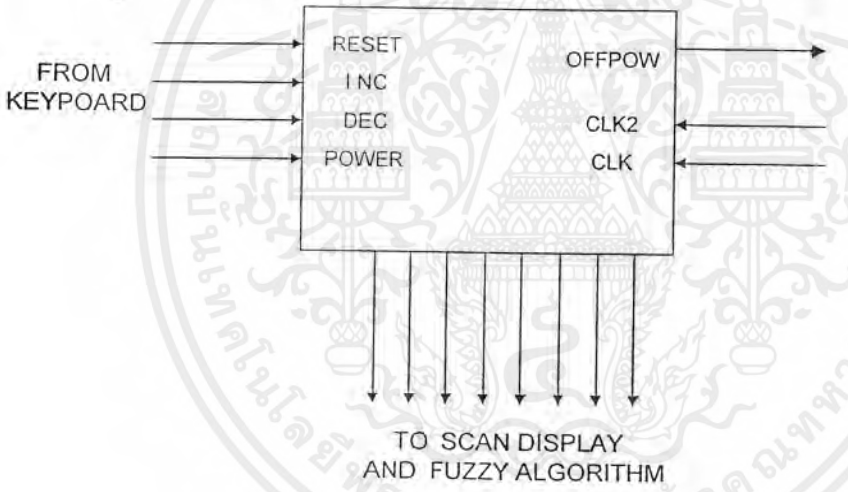
เพื่อความสะดวกในการนำไปสร้างโดยใช้ภาษา VHDL จึงแยกพิจารณาเป็นแต่ละส่วนๆไป ดังนี้

4.1 ส่วนสแกนคีย์บอร์ด (SCAN KEYBOARD)

เป็นส่วนที่รับค่าอินพุทจากสวิทช์ซึ่งเป็นการตั้งค่าอุณหภูมิที่ต้องการ โดยจะมีสวิทช์ดังต่อไปนี้

- POWER ใช้สำหรับเปิด – ปิดระบบ
- RESET ใช้รีเซตระบบเพื่อตั้งค่าอุณหภูมิใหม่
- INC เป็นสวิทช์ที่ใช้เพิ่มค่าอุณหภูมิ
- DEC เป็นสวิทช์ที่ใช้ลดค่าอุณหภูมิ

ซึ่งจะได้เอาท์พุทที่ต้องการในส่วนพื้นที่เป็น 8 บิท เพื่อใช้ในการตั้งค่าอุณหภูมิ โดยทั่วๆไปจะใช้ได้ในช่วง 20-40 องศาเซลเซียสสำหรับเครื่องปรับอากาศ

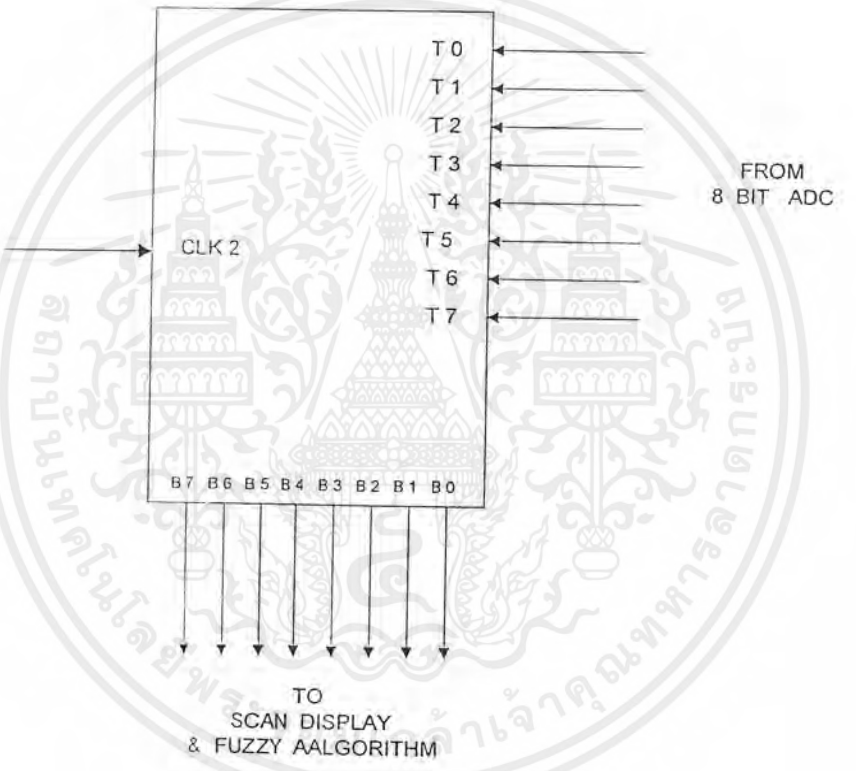


รูปที่ 4.3 ส่วนสแกนคีย์บอร์ด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2 ส่วน BUFFER TEMPERATURE

ส่วนนี้จะเป็นส่วนที่รับอุณหภูมิจากเซนเซอร์ แต่เนื่องจากการออกแบบระบบเป็นการออกแบบที่เป็นดิจิตอล ไม่สามารถที่จะใช้ต่อกับเซนเซอร์ได้โดยตรง จึงต้องรับค่าอินพุทจาก ADC (Analog to Digital Converter) ที่แปลงสัญญาณให้เป็นดิจิตอลแล้ว ในส่วนนี้เราใช้อินพุท 8 บิท เนื่องจากต้องการความละเอียด สัญญาณเอาท์พุทจาก BUFFER นี้จะถูกส่งไปแสดงที่ DISPLAY และคำนวณที่ส่วนของ FUZZY ALGORITHM แสดงให้เห็นดังรูป 4.4

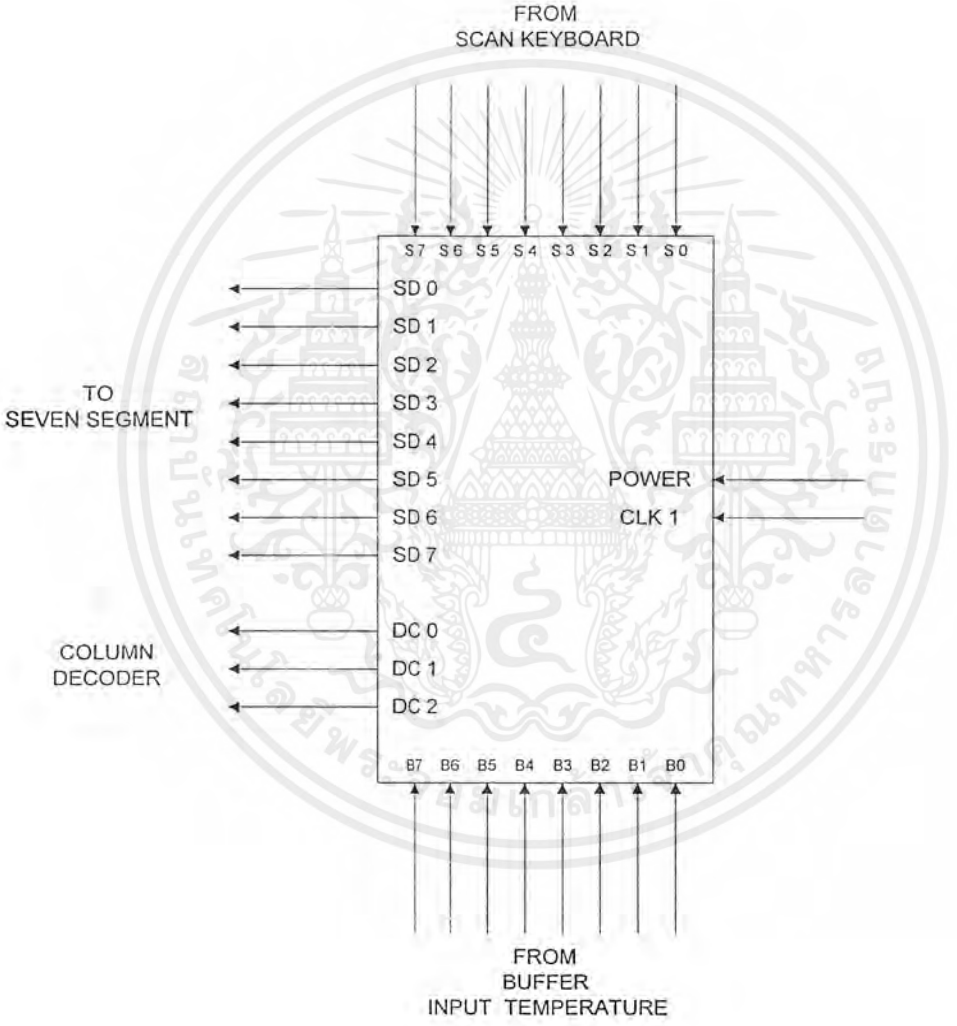


รูปที่ 4.4 ส่วน Buffer Temperature

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.3 ส่วนสแกนดิสเพลย์ (SCAN DISPLAY)

ในส่วนนี้จะรับอินพุทจากส่วนของเอาท์พุทสแกนคีย์บอร์ด และ Buffer Temperature ซึ่งเป็นอินพุทขนาด 8 บิท เพื่อนำมาแปลงเป็นเลข BCD แล้วส่งไปแสดงที่ Seven Segment อีกทีหนึ่ง และจะมีสัญญาณ Decode ขนาด 3 บิทใช้สำหรับเลือก Column ของ Seven Segment ที่จะใช้แสดงผล ดังแสดงไว้ในรูป



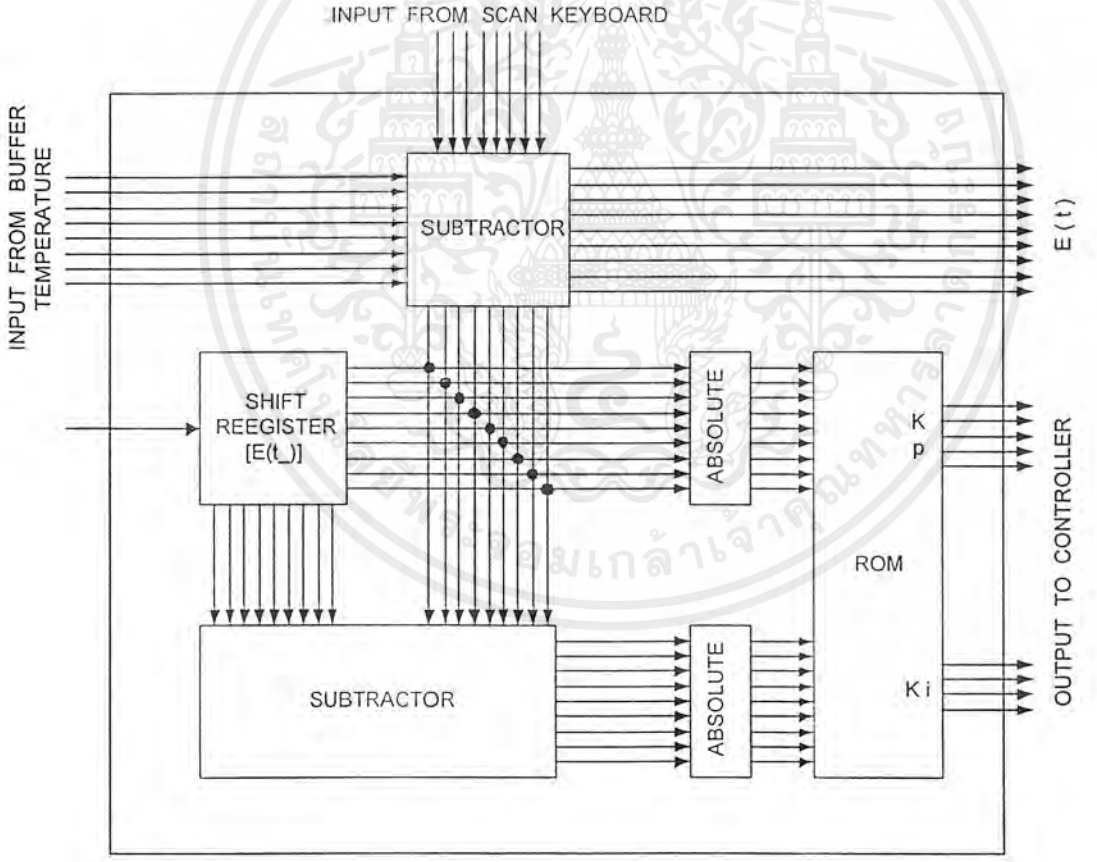
รูปที่ 4.5 ส่วน Scan Display

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.4 ส่วน Fuzzy Algorithm

ส่วนนี้เป็นส่วนที่สำคัญเพราะเป็นส่วนประมวลผลกลาง ดังนั้น จึงสามารถแบ่งย่อยออกเป็นส่วนๆ ดังรูปที่ 4.6 โดยให้ค่าอินพุทเป็น ค่า T(t) และ Set Temp ซึ่งจะนำค่า 2 ค่านี้มาคำนวณหาค่า E (t) แล้วนำค่าที่ได้ไปคำนวณหาค่า ΔE อีกทีโดยที่ E(t-1) ได้มาจาก Shift Register ซึ่งค่า ΔE และ E(t) จะต้องนำไปผ่านวงจร Absolute เพื่อที่จะเลือกเฉพาะค่าที่เป็นบวก แล้วนำค่า $|\Delta E|$ และ $|E(t)|$ ไปหาค่า K_p และ K_i ที่เหมาะตาม Algorithm Fuzzy โดยที่ใช้เทคนิค Look up Table ในการหาค่า

ค่าเอาท์พุทจากส่วนนี้คือ ค่า K_p, K_i และ E(t) จะถูกส่งออกไปในรูปของเลขฐานสองขนาด 5, 4, และ 8 บิท ตามลำดับ ก็เพราะว่าค่า K_p สูงสุดคือ 24 และค่า K_i สูงสุดเท่ากับ 12 นั่นเอง ดังนั้นจึงได้เอาท์พุทส่วนนี้ทั้งหมด 17 บิท ดังแสดงในรูปที่ 4.6



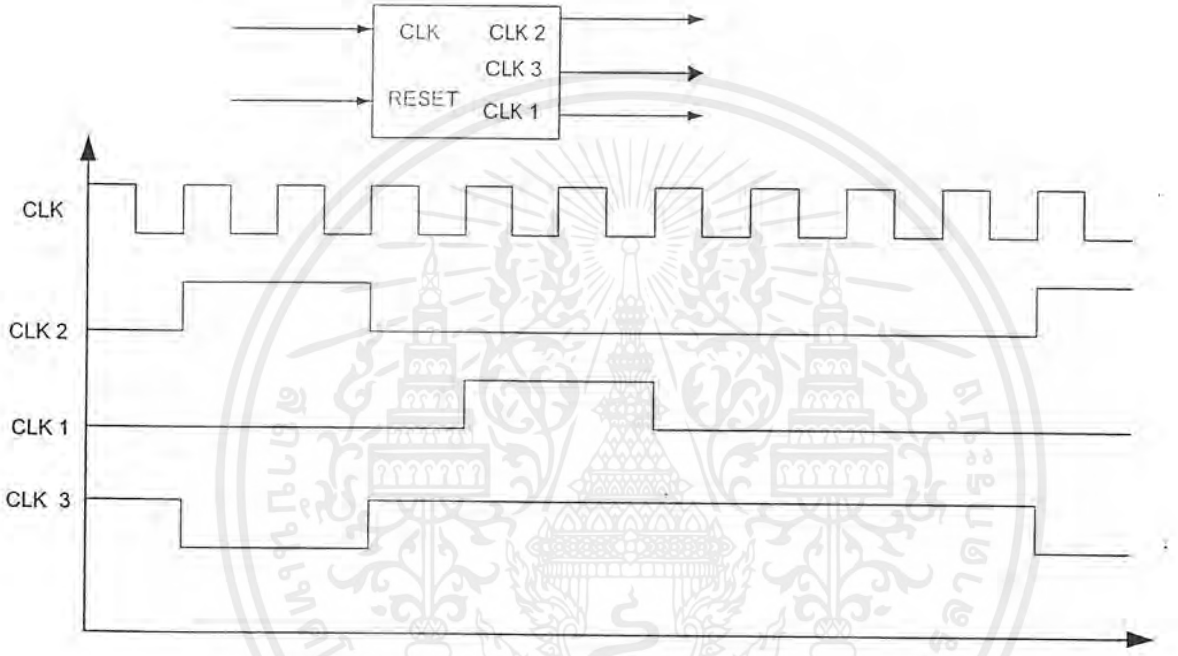
รูปที่ 4.6 แสดงส่วนรายละเอียดของ Fuzzy Algorithm

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.5 ส่วนควบคุมแบบพีไอ (PI CONTROLLER)

ในส่วนนี้ เราต้องหาฟังก์ชั่นการถ่ายของพีไอ ซึ่งเป็นตัวแปรในโดเมน S ฉะนั้นจะต้องมีการเปลี่ยนไปเป็นโดเมน Z เพื่อที่จะทำให้สามารถทำการคำนวณแบบดิจิตอล

4.6 ส่วนสร้างสัญญาณนาฬิกา 3 เฟส



รูปที่ 4.7 แสดงวงจรและสัญญาณนาฬิกาของวงจรสร้างนาฬิกา 3 เฟส

สาเหตุสำคัญที่ต้องใช้สัญญาณนาฬิกา 3 เฟส เพื่อทำให้การทำงานของระบบสอดคล้องกัน โดยที่อินพุทจาก Keyboard เข้ามา ส่วน Scan Keyboard ก็จะรับข้อมูลเข้ามาในช่วงขอบขาลงของ CLK2 เช่นเดียวกัน โดยที่สัญญาณ CLK2 จะเป็นสัญญาณที่ใช้ในการควบคุม ADC ค่าที่ได้จาก Sensor และ Keyboard นี้จะส่งไปที่ส่วน Fuzzy Algorithm ในช่วงขอบขาขึ้นของ CLK1 เพื่อหาค่า K_p และ K_i ที่เหมาะสม แล้วจากนั้นนำค่า K_p และ K_i นี้ไปใช้ในการคำนวณหาค่าเอาท์พุท ตามการคำนวณแบบดิจิตอล แล้วส่งออกไปที่ เอาท์พุทในช่วงขอบขาลงของ CLK1 ทั้งนี้ก็เพื่อที่จะให้การทำงานของแต่ละภาคมีความสัมพันธ์ที่สอดคล้องกัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

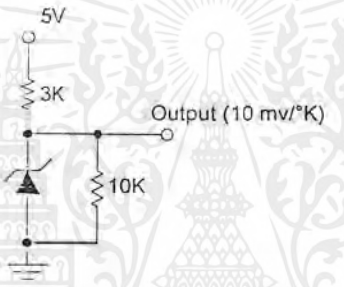
บทที่ 5

ส่วนประกอบของวงจรภายนอก

การออกแบบวงจร SENSOR and ADC

5.1 เซนเซอร์ (Sensor)

ในการออกแบบวงจร โดยใช้ LM355 เป็น Temperature Sensors ซึ่งมีลักษณะเช่นเดียวกับ Zener โดยจะเปลี่ยนแรงดันตกคร่อม ± 10 MV/$^{\circ}$K ซึ่งมีความต้านทานน้อยกว่า 1 Ω ที่กระแส 400 μ A-5mA ดังนั้นในการ Calibrated ที่ 25 $^{\circ}$C จะมีแรงดันตกคร่อม Sensors 2.982 V



รูปที่ 5.1 Calibrated Sensors

5.2 เปลี่ยนสัญญาณอนาลอกเป็นดิจิตอล (Analog to Digital)

จะกำหนดให้ที่ 19°C นั้น ADC จะแสดงค่า output เท่ากับ "00000000" และที่ 20°C จะแสดง output ของ ADC คือ "00001010"

20°C - 19°C = 1°C

00001010 - 00000000 = 00001010

จะเห็นว่าที่อุณหภูมิแตกต่างกัน 1°C ทำให้ตัวเลข output ของ ADC เปลี่ยนไปเท่ากับ "10"นั่นคือ ที่อุณหภูมิเปลี่ยนไป 0.1°C จะทำให้ตัวเลข output ของ ADC เปลี่ยนไปเท่ากับ "1" เช่น

20°C $\Rightarrow$ "00001010"

20.1°C $\Rightarrow$ "00001011"

20.2°C $\Rightarrow$ "00001100"

เมื่อ Sensor อุณหภูมิที่ใช้งานนั้น linear ที่อุณหภูมิ -40°C - 100°C โดยมีค่าผิดพลาด $\pm$20°C ค่าแรงดัน output ของ Sensor เท่ากับ 10 mV/$^{\circ}$K (0 mV ที่ 0°K ดังนั้นที่ 25 เท่ากับ 2.982V)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากคุณสมบัติดังกล่าวจะพบว่าเมื่ออุณหภูมิเปลี่ยนไป 0.1°C ทำให้แรงดัน output เปลี่ยนไป 1 mV และทำให้ ADC มี output เปลี่ยนไป 1 bit

ตารางที่ 5.1 อุณหภูมิ กับ output ADCที่ต้องการ

Temperature (°C)	ADC Output
19	0000 0000
20	0000 1010
21	0001 0100
22	0001 1110
23	0010 1000
24	0011 0010
25	0011 1100
26	0100 0110
27	0101 0000
28	0101 1010
29	0110 0100
30	0110 1110
31	0111 1000
32	1000 0010
33	1000 1100
34	1001 0110
35	1010 0000
36	1010 1010
37	1011 0100
38	1011 1110
39	1100 1000
40	1101 0010

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.3 วงจรทดสอบ ADC 0804

จากรูปที่ 5.2 เป็นวงจรทดสอบ ADC โดยการต่อ output กับ LED เพื่อแสดงผลและให้ input เป็น DC Voltage ปรับค่าได้ตั้งแต่ 0-5 V ซึ่งกำหนด $V_{REF}/2$ เท่ากับ 2.5 Volt ในที่นี้เราใช้ MX 584 เป็น Voltage Reference จะได้ output ที่ input 0 Volt เท่ากับ 0000 0000 ที่ 2.5 Volt เท่ากับ 1000 0000 และที่ 5 Volt เท่ากับ 1111 1111 โดยมีความถี่ในการทำงานของ Ic ที่ Clock เท่ากับ

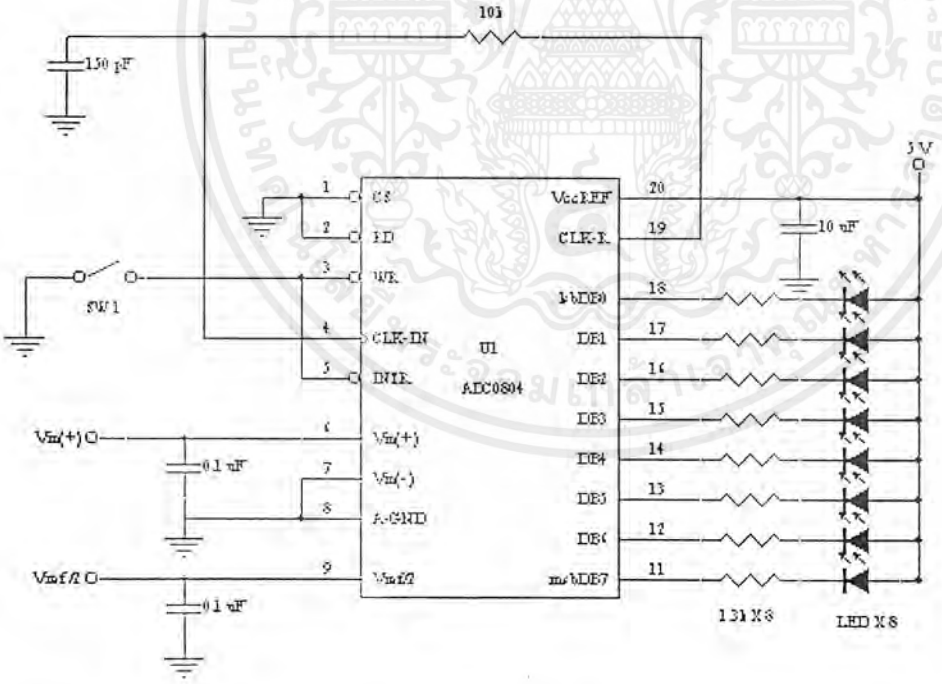
$$f_{CLK} = \frac{1}{1.1RC}$$

ในที่นี้เราใช้ R = 10 k และ C = 150 pF

$$f_{CLK} = \frac{1}{1.1\times 10\times 10^{3}\times 150\times 10^{-12}}$$

$$= 606.1KH_{z}$$

แต่เนื่องจาก Sensor ที่ใช้ใน output 2.982 Volt ที่ 25°C แต่ต้องการตั้งแต่ 19°C-40°C และให้ output



ตั้งแต่ 2.862 Volt ถึง 3.282 Volt จึงต้องสร้างวงจรใหม่ขึ้นมา

รูปที่ 5.2 วงจรทดสอบ ADC 0804

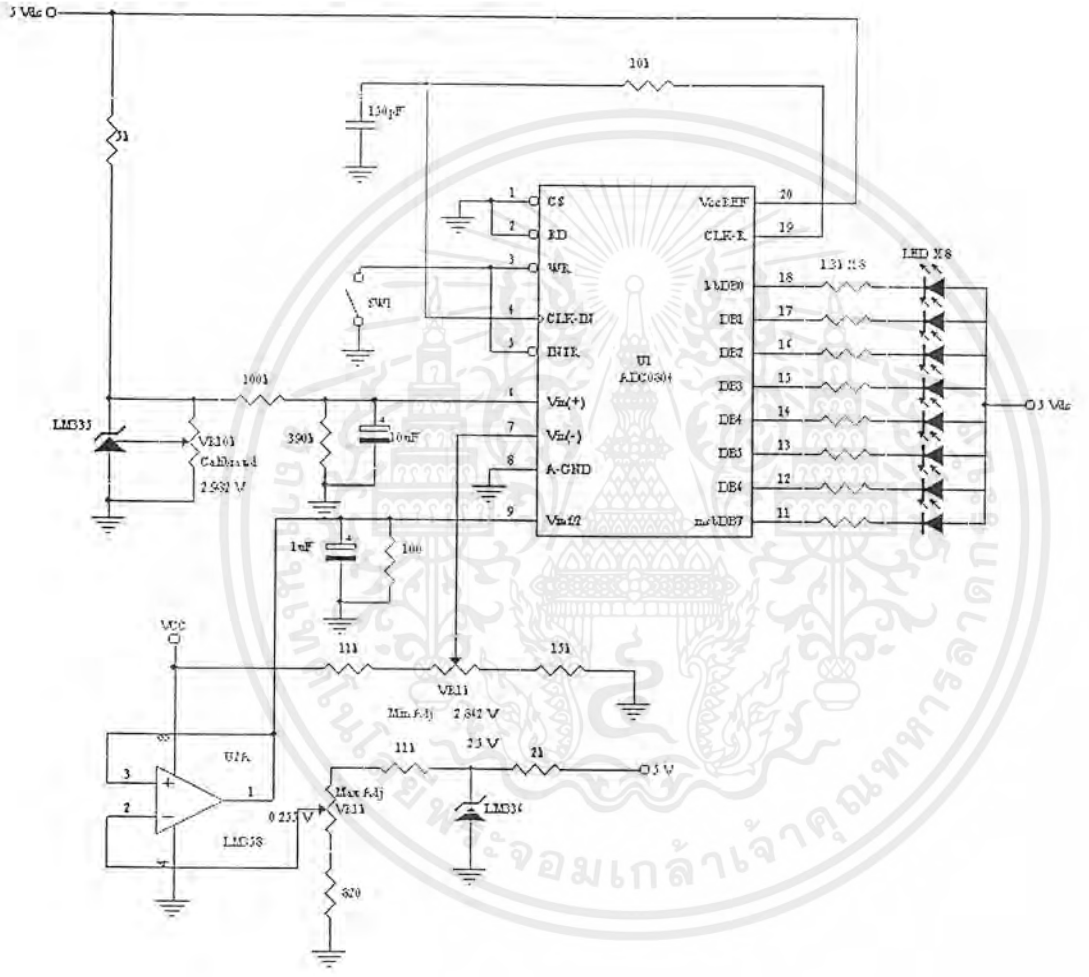
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.4 วงจรใช้งานจริง

ในวงจรที่ใช้งานจริงนี้ มีส่วนที่เพิ่มเติม คือ วงจร Reference Voltage 2 ชุด คือ

Vin- เป็นการตั้งแรงดัน input ต่ำสุดที่ 2.826 Volt

$V_{REF}/2$ เป็นการตั้ง Reference ในที่นี้ต้องการ 256 mV เพื่อให้ได้ output ที่เปลี่ยนแปลง 1 bit ที่ input เปลี่ยนไป 1 mV ดังนั้น Voltage ในส่วนนี้เท่ากับ 128 mV



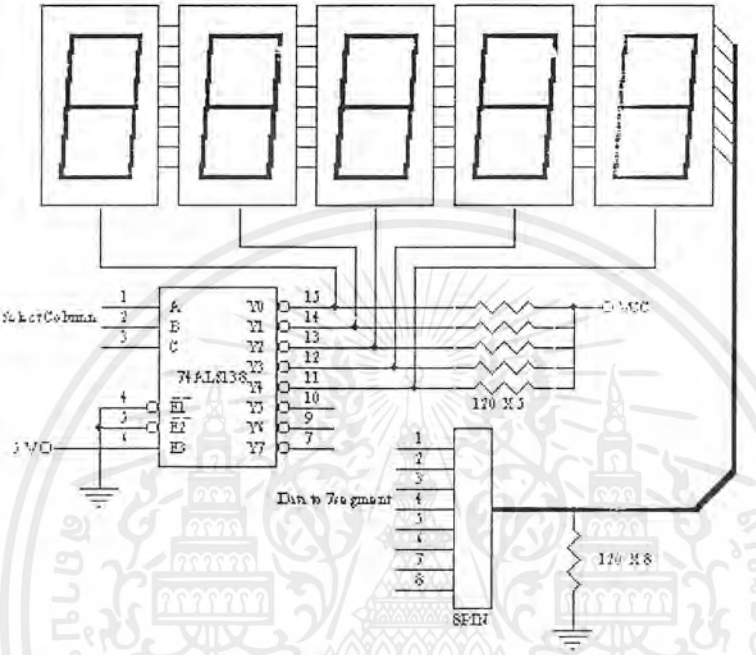
รูปที่ 5.3 วงจรใช้งานจริง

วงจรนี้จะรับ input จาก Sensor อุณหภูมิโดยตรง โดยที่ต่อ R 10 K คร่อม LM 335 เมื่อ Calibrate ที่ 25°C เท่ากับ 2.982 V

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.5 ส่วนแสดงผล (Scan Display)

ในส่วนนี้จะใช้ Seven Segment แสดงผลเพราะระบบควบคุมอุณหภูมินี้จะส่งค่าข้อมูลออกไปแสดง โดยใช้เวลาที่เหมาะสมในการ Scan Display ที่ความถี่ที่สายตาไม่สามารถมองเห็นการกระพริบของ LED ได้ ลักษณะการต่อจะแสดงดังรูปที่ 5.4



รูปที่ 5.4 แสดงการเชื่อมต่อกับ Seven Segment

จากรูปจะเห็นว่ามี Seven Segment ทั้งหมด 5 ตัว ซึ่งจะแบ่งออกเป็น 2 ชุด คือ ชุดแรกประกอบด้วย Seven Segment 3 ตัว ส่วนนี้จะแสดงค่าของอุณหภูมิที่วัดได้จาก Sensor โดยจะมีค่าเป็นทศนิยมหนึ่งตำแหน่ง ส่วนที่สองคือที่ใช้สำหรับเซตค่าอุณหภูมิที่ต้องการ ซึ่งจะเป็นจำนวนเต็ม ใช้ Seven Segment เพียง 2 ตัว

การ Scan จะ IC 74LS138 เป็นตัวสแกนให้ Seven Segment แต่ละทำงานไม่พร้อมกันที่ตาเราไม่สามารถมองเห็น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.6 ส่วนแสดงผลเอาท์พุท

ส่วนที่1

ในส่วนนี่จะเป็นส่วนที่แปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอล ทั้งนี้เพราะในการใช้งานจริงๆ จะนำสัญญาณอนาลอกไปใช้ การแปลงสัญญาณทำได้โดยใช้ไอซีหรือจะใช้วงจรตัวต้านทานแลดเดอร์ (ladder) โดยจะมีลักษณะดังรูปที่ 5.5 ในวงจรนี้จะใช้ขนาด 16 บิท ค่าแรงดันที่คร่อมตัวต้านทานจะมีค่าลดลงเป็นครึ่งหนึ่งของแรงดันคร่อมตัวต้านทานก่อนหน้านี้ จากกฎของโอห์ม ลักษณะของแรงดัน คำนวณได้จากสูตรโดยจะมีค่าเท่ากับเลขฐานสองของบิทแต่ละบิทที่มีค่าเท่ากับ "1" บวกกัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ส่วนที่2

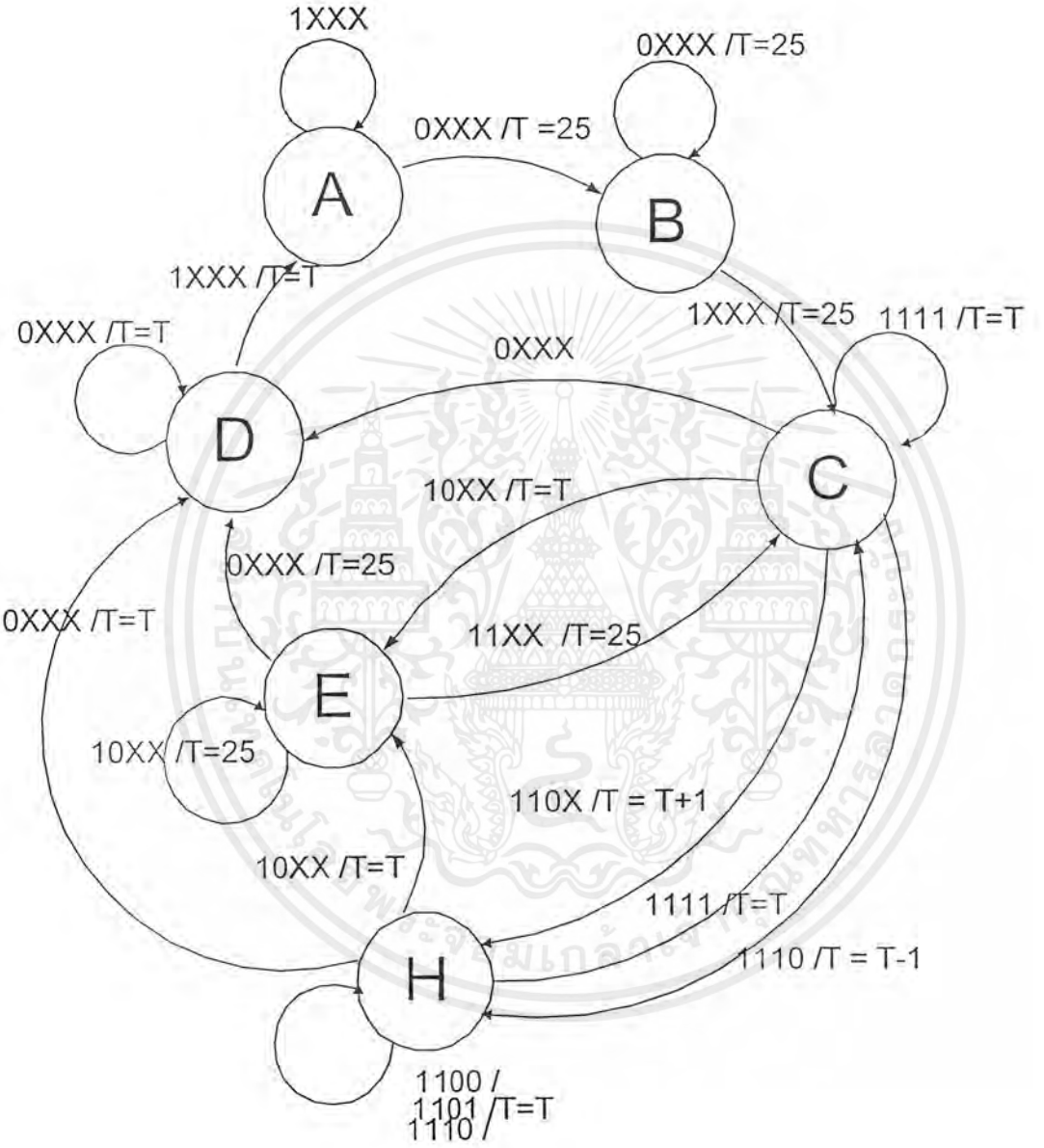
เป็นส่วนของวงจรที่รับค่าสัญญาณเอาท์พุทจากส่วนของดีเอซี (DAC) ซึ่งจะเป็นค่าแรงดันไฟฟ้าขนาน 0 ถึง 10 โวลท์ แล้วเปลี่ยนให้เป็นสัญญาณความถี่ที่มีค่าความถี่ตั้งแต่ 30 – 90 เฮิตท์ สัญญาณที่ได้นี่จะเป็นสัญญาณอินพุทให้กับวงจรส่วนที่นำไปขับความเร็วมอเตอร์ของคอมเพลสเซอร์อีกทีหนึ่ง ในที่นี้ไม่ได้แสดงวงจรส่วนนี้เพราะถือว่าเป็นงานโปรเจคอีกตัวหนึ่งก็ได้และอาจจะทำให้งานไม่เสร็จสมบูรณ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 6

โปรแกรมและการทดสอบ

6.1 ส่วนโปรแกรมสแกนคีย์บอร์ด(SCAN KEYBOARD)



รูปที่ 6.1 แสดง State Machine ของส่วนสแกนคีย์บอร์ด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในส่วนนี้จะรับค่าจาก KEYBOARD อยู่ 4 ค่าคือ

- POWER เป็นสวิทช์ ควบคุมเปิด ปิดระบบ
- RESET เป็นสวิทช์ ในการรีเซ็ตระบบ
- INC เป็นสวิทช์ ใช้เพิ่มค่าอุณหภูมิ
- DEC เป็นสวิทช์ ใช้ลดค่าอุณหภูมิ

ในการเขียน STATE MACMINE นั้น ในการกด สวิทช์จะพิจารณาว่าการกดสวิทช์ 2 อันพร้อมกันไม่ได้ เนื่องจากความเป็นจริงมันไม่มีโอกาสที่จะพร้อมกัน ลำดับความสำคัญของสวิทช์ ก็คือ POWER, RESET, INC และ DEC โดยที่เมื่อเริ่มต้นการทำงานนั้นเราจะ SET ค่าอุณหภูมิเริ่มต้นที่ 27 $^{\circ}$C เสมอ เพื่อเป็นค่ากลางให้กับระบบ โดยจะสามารถตั้งค่าอุณหภูมิในช่วง 20-40 $^{\circ}$C

- STATE MACMINE ของ SCAN KEYBOARD

สถานะ A เป็นสถานะที่ระบบถูกปิดอยู่ และรอการกดคีย์ POWER

สถานะ B เป็นสถานะที่ระบบถูกเปิด และเซ็ตการปล่อยคีย์ POWER

สถานะ C เป็นสถานะที่ระบบรอรับการกดคีย์ต่างๆ

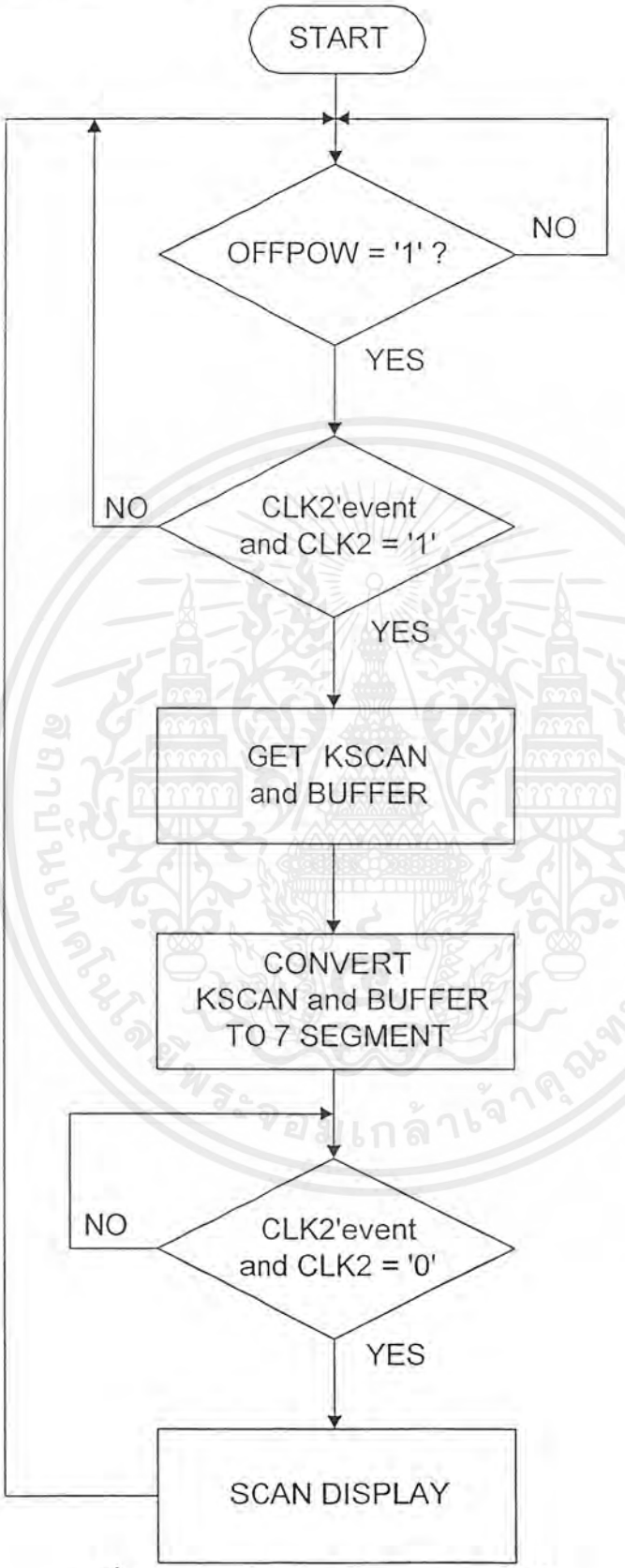
สถานะ D เป็นสถานะที่ระบบถูกปิดและเซ็ตการปล่อยคีย์ POWER

สถานะ E เป็นสถานะที่ระบบถูกรีเซ็ต และรอการปล่อยคีย์ POWER

สถานะ H เป็นสถานะที่ระบบเซ็ตการปล่อยคีย์ INC หรือ DEC

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.2 ส่วนโปรแกรม SCAN DISPLAY

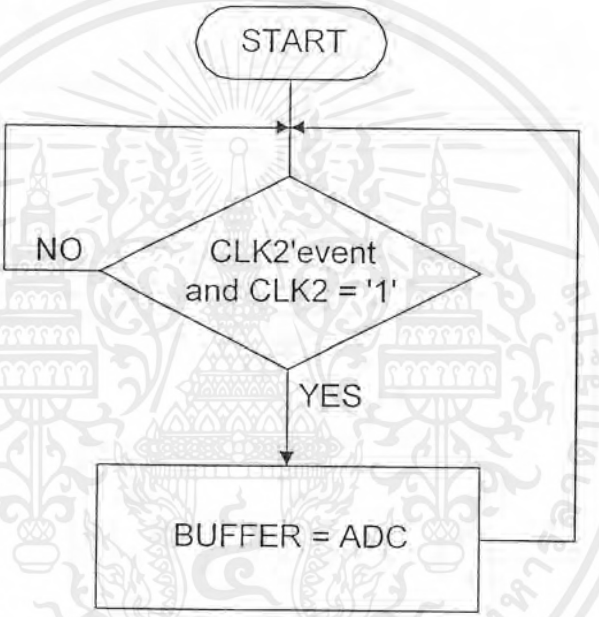


รูปที่ 6.2 แสดงการทำงานส่วนสแกนดิสเพลย์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในส่วนนี้จะทำหน้าที่รับค่าอินพุทจากส่วน SCAN KEYBOARD และส่วน BUFFER TEMP ที่ขอบขาหน้าของ CLK1 มาแปลงเป็น BCD แล้วนำไปแสดงผลที่ SEVEN SEGMENT อีกทีหนึ่ง โดยแบ่งสัญญาณเป็น 2 กลุ่ม คือ ส่วนหนึ่งเป็น DATA 8 BIT ใช้เป็นข้อมูลของ SEVEN SEGMENT และส่วนที่สองคือ สัญญาณ 3 BIT เป็น DECODE เพื่อเลือก COLUME ของ SEVEN SEGMENT ที่ใช้แสดงผล โดยโปรแกรมทำงานสัมพันธ์กับ CLK1 ดังนี้ ที่ขอบขาหน้าของ CLK1 จะรับข้อมูลจาก SCAN KEYBOARD และ BUFFER TEMP ในขณะที่ขอบขาหลังของ CLK1 จะส่งข้อมูลออกไป SCAN DISPLAY ที่เป็นแบบ SEVEN SEGMENT

6.3 โปรแกรมส่วน BUFFER TEMPERATURE

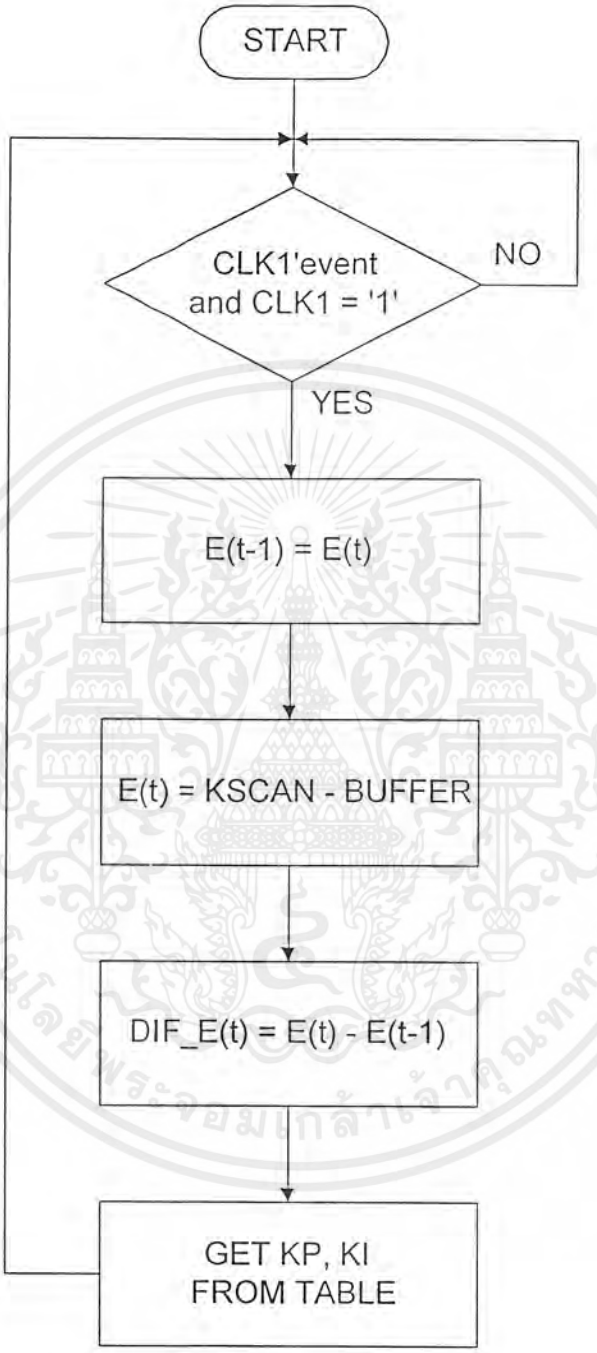


รูปที่ 6.3 แสดงการทำงานของส่วน BUFFER TEMPERATURE

เป็นส่วนที่รับค่าอุณหภูมิจากภายนอกที่เป็นสัญญาณ DIGITAL ขนาด 8 BIT เข้ามาโดยใช้สัญญาณ CLK2 ในการควบคุม โดยระบบจะส่งสัญญาณ CLK3ไป LATCH ค่าที่ ADC ก่อน จากนั้นที่ขอบขาลงของ CLK2 ก็จะรับสัญญาณค่าอุณหภูมิภายนอกเข้ามาแล้วส่งค่าที่ได้ไปยังส่วน SCAN DISPLAY และส่วน FUZZY ALGORITHM

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.4 โปรแกรมส่วน FUZZY ALGORITHM



รูปที่ 6.4 แสดงการทำงานของส่วน FUZZY ALGORITHM

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

อินพุทของส่วนนี้ก็คือ สัญญาณ T(t) ซึ่งก็คือ ค่าอุณหภูมิจาก SENSOR และสัญญาณ Set Temp แล้วนำมาคำนวณหาค่า E(t) แล้วนำค่า E(t) ไปคำนวณค่า ΔE อีกที โดยที่ $\Delta E = E(t) - E(t-1)$ได้สัญญาณ E(t-1) ได้มาจาก SHIFT REGISTER ซึ่งค่า ΔE และ E(t) จะผ่านส่วนวงจร ABSOLUTEB เอาเฉพาะค่าบวกไปเลือก KP และ KI ที่เหมาะสมตาม ALGORITHM โดยใช้วิธีการ Look up Table ในการหาค่า โดยค่าที่ส่งออกไปมี 3 ส่วนคือ E(t) ขนาด 9 BIT KP ขนาด 5 BIT และ KI ขนาด 4 BIT ซึ่งจะนำไปคำนวณอีกครั้งหนึ่ง

6.5 ส่วนควบคุมแบบพีไอ (PI CONTROLLER)

ในส่วนนี้เราจะทำการคำนวณค่าฟังก์ชันถ่ายโอนแบบพีไอซึ่งเป็นตัวแปรมนโดรเมน S ฉะนั้นจะต้องมีการเปลี่ยนแปลงไปเป็นโดรเมน Z เพื่อที่จะทำให้สามารถ ทำการคำนวณแบบดิจิตอลได้ (ใช้หลักการ DIGITAL SIGNAL PROCESSOR) โดยในการเปลี่ยนตัวแปรนี้ จะใช้การส่งแบบ BILINEAR TRANSFORM ซึ่งจะให้ผลการเปลี่ยนแปลงมีค่าใกล้เคียงกับฟังก์ชันถ่ายโอนเดิมที่มีความถี่ไม่สูงมากนัก

หลักจากที่ได้ฟังก์ชันถ่ายโอนในโดรเมนจากการแปลงแบบ BILINEAR มาแล้วก็จะสามารถเขียนโครงสร้างของระบบควบคุมแบบพีไอได้หลายอย่าง เช่น แบบโดยตรง , แบบสับเปลี่ยน เป็นต้น ทั้งนี้การคัดเลือกโครงสร้าง ในแต่ละแบบจะมีข้อดี ข้อเสียที่แตกต่างกันไป ซึ่งในที่นี้เลือกใช้โครงสร้างแบบโดยตรง เพราะว่าไม่มีการป้อนกลับจากเอาท์พุททำให้โอกาสเกิด OVERFLOW น้อยกว่าแบบอื่นๆ ซึ่งสามารถแสดงได้ดังสมการต่อไปนี้

โดยที่ X(nT) คือ E(t) ที่ถูกสุ่มขึ้นมาด้วยความถี่เท่ากับส่วนของคาบเวลา เนื่องจากเราใช้ในการคำนวณแบบดิจิตอล จึงต้องมีการสุ่มสัญญาณ ซึ่งค่าความถี่ในการสุ่มสัญญาณนี้จะต้องมากพอที่จะรักษารูปร่างของสัญญาณไว้ไม่ให้ผิดเพี้ยน ในที่นี้สัญญาณ E(t) เป็นค่า ERROR ของอุณหภูมิ ซึ่งมีการเปลี่ยนแปลงค่อนข้างช้า ดังนั้นเราจึงไม่จำเป็นต้องใช้ความถี่ในการสุ่มสูงมาก โดยในที่นี้จะใช้ค่าความถี่ในการสุ่มสัญญาณเท่ากับ 1 KHZ ในขณะที่ความถี่ของสัญญาณนาฬิกามีค่าเท่ากับ 10 KHZ

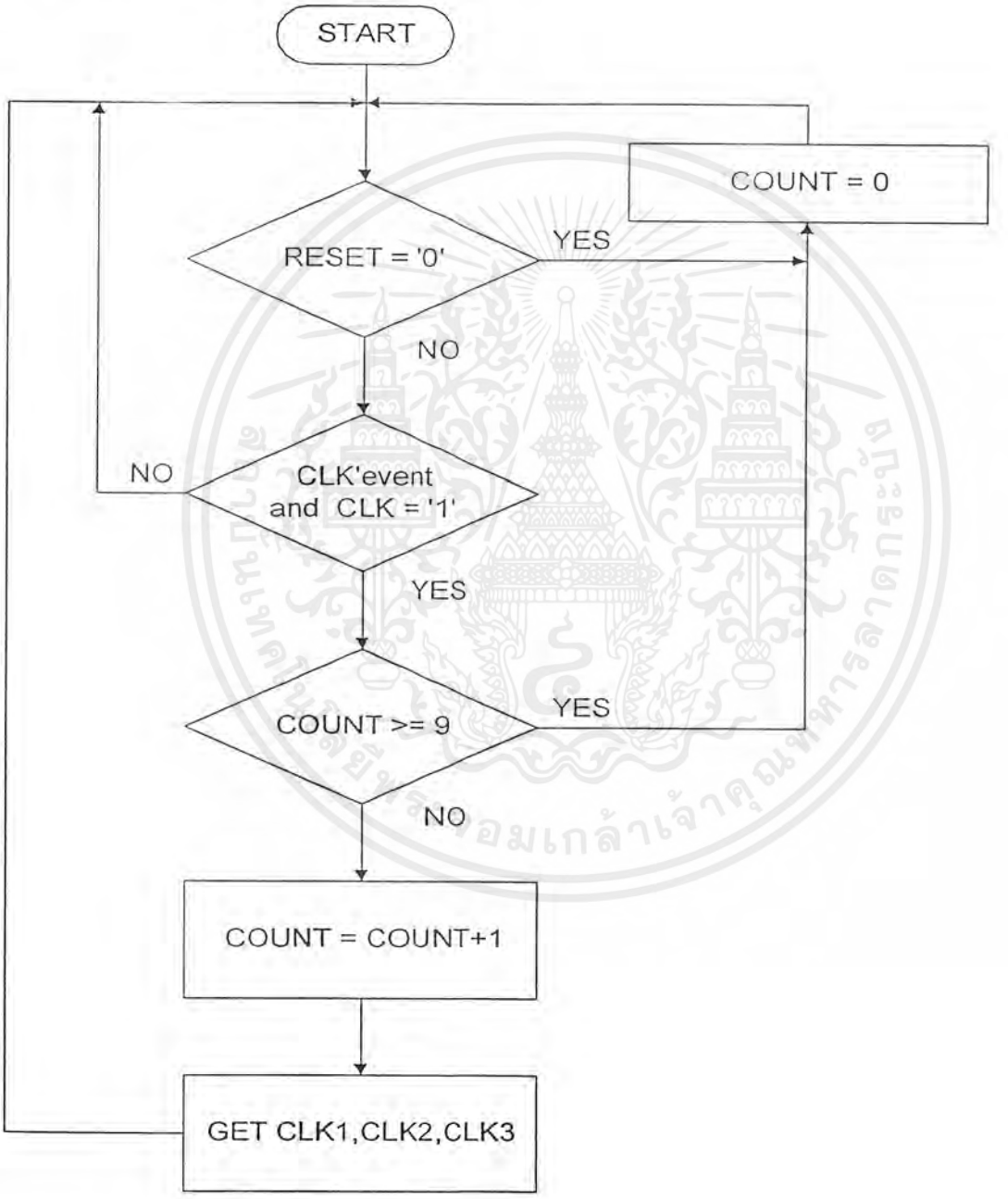
6.6 การทำงานโดยรวมของระบบ

เมื่อมีอินพุทจาก KEYBOARD เข้ามาโดยส่วน SCAN KEYBOARD จะรับข้อมูลเข้ามาในช่วงขอบขาลงมา CLK2 ซึ่งเป็นขณะเดียวกันกับ BUFFER ก็รับค่าอุณหภูมิจากสิ่งแวดล้อมที่ขอบขาของ CLK2 เช่นกัน โดยที่ก่อนหน้านี้ระบบได้ส่ง CLK3 ออกไป LATCH ค่าที่ ADC ก่อนแล้ว และค่าสัญญาณอินพุทสองทางคือ จาก KEYBOARD และจาก SENSOR จะส่งให้กับ SCAN DISPLAY และ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

FUZZY ALGORITHM ส่วน SCAN DISPLAY จะนำค่าทั้งสองไปแสดงผลภายนอก ในรูปของข้อมูลสำหรับ SEVEN SEGMENT และ สัญญาณ SCAN COLUME ส่วนสัญญาณที่เข้าไปยัง FUZZY ALGORITHM จะทำให้ได้ค่า E(t) เป็น ERROR และค่า KP และ KI โดยสัมพันธ์กับขาขอบขาขึ้นของ CLK1 ส่งไปควบคุม PI CONTROLLER ทำให้ได้สัญญาณ OUTPUT ออกควบคุมภายนอก

6.7 ส่วนสร้างสัญญาณนาฬิกา 3 เฟส

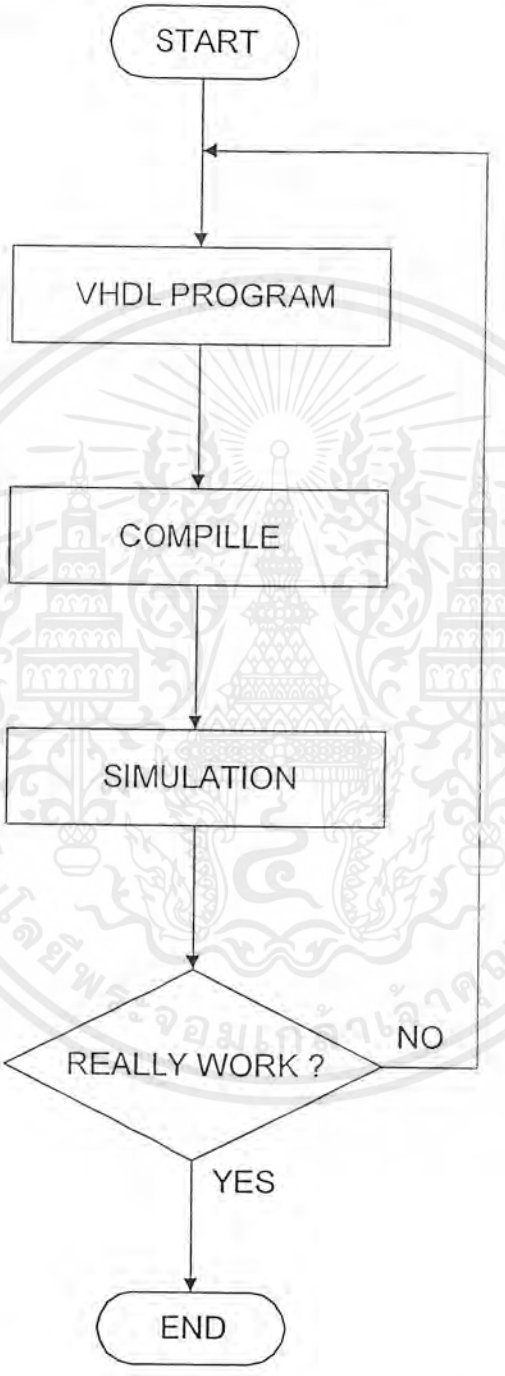


รูปที่ 6.5 แสดงการทำงานส่วนสร้างสัญญาณนาฬิกา 3 เฟส

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.8การทดสอบโปรแกรม (SIMULATION)

เมื่อเขียนโปรแกรมของวีเอชดีแอล (VHDL) ได้แล้ว จะต้องนำมา Compile ให้ผ่าน จากนั้นจึงนำมาทดสอบโปรแกรม โดยการป้อนค่าอินพุทที่เหมาะสมให้กับระบบที่ต้องการทดสอบ ซึ่งวิธีที่นิยมใช้โดยทั่วไปก็คือการเขียนโปรแกรมวีเอชดีแอลขึ้นมาอีกโปรแกรมหนึ่ง เพื่อสร้างอินพุทขึ้นมา



รูปที่ 6.6 แสดงขั้นตอนการเขียนโปรแกรมภาษา วีเอชดีแอล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ให้กับโปรแกรมที่เราต้องการทดสอบ ซึ่งโปรแกรมนี้เรียกว่า Test Bench ซึ่งจะสามารถสรุปเป็นโฟลว์ชาร์ตได้ดังรูปที่ 6.6

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 7

การสังเคราะห์วงจรและการทดลองในระดับ Time Sim

ในบทที่ 6 นั้น เราได้เขียนโปรแกรมภาษาวีเอชดีแอล ซึ่งบรรยายระบบในเชิงพฤติกรรมไว้แล้ว โดยผ่านการทดสอบโปรแกรมด้วยการทดสอบ ในขั้นของ Source Code แต่ในการสร้างวงจรโปรแกรมที่ผ่านการทดสอบแล้วอาจจะไม่สามารถสังเคราะห์เป็นวงจรดิจิตอลได้ ทั้งนี้ก็เนื่องมาจากขึ้นอยู่ว่าในการสังเคราะห์นั้นเราจะเลือกสังเคราะห์โดยใช้เทคโนโลยีของ FPGAs ชนิดไหน ของบริษัทอะไร เบอร์อะไร มีโครงสร้างภายในเป็นอย่างไร ในบทนี้จึงได้จัดลำดับขั้นตอน การออกแบบ ดังนี้

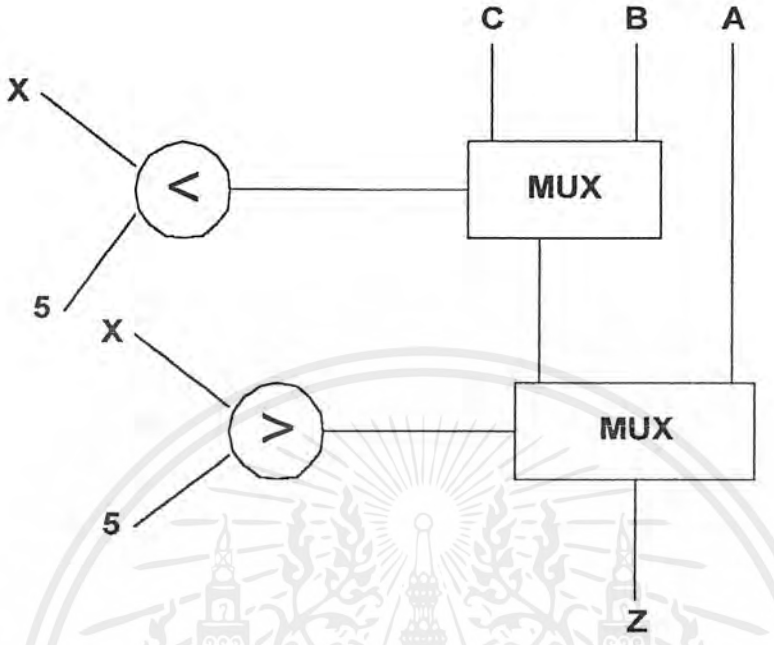
1. การเลือกใช้ FPGAs จึงต้องพิจารณาดูว่าจะเลือกใช้งาน FPGAs ที่มีสถาปัตยกรรมเป็น Multiplexer หรือเป็น สถาปัตยกรรมของ Decoder ซึ่งในการเขียนโปรแกรมภาษาวีเอชดีแอล นั้นก็จะมีความแตกต่างกันด้วย ดังเช่นตัวอย่าง การเปรียบเทียบระหว่างการเขียนแบบ If Statement และ Case Statement

```
If Statement
if (X>5) then
   Z<= A;
else if (X<5) then
   Z <= B;
else
   Z <= C ;
end if ;
```

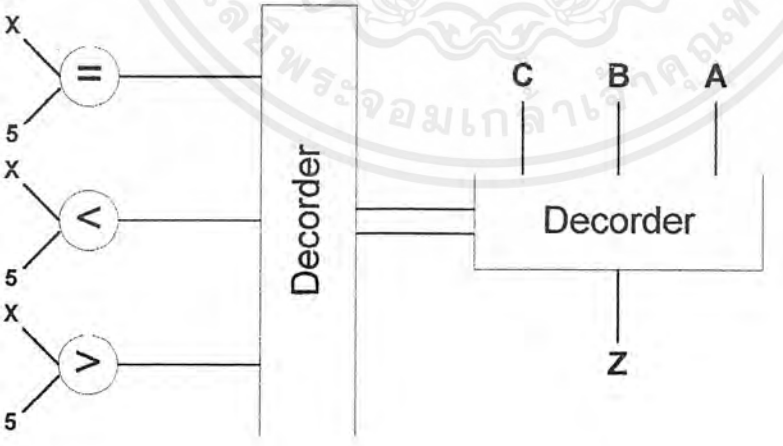
ซึ่งจากโปรแกรมของIf Statement เรามองรูปแบบของวงจรที่ได้ดังรูป 7.1 นั่นคือ จะใช้สถาปัตยกรรมของ MUX หรือ Multiplexer นั่นเอง

```
Case Statement
case X is
   when 0 to 4 =>
   Z <= B ;
when 5 =>
   Z <= C ;
when others =>
   Z <= A ;
end case ;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 7.1 สถาปัตยกรรมของ MUX



รูปที่ 7.2 สถาปัตยกรรมของ Decoder

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากโปรแกรมของ Case Statement รูปแบบของวงจรที่ได้ดังรูป 7.2 นั่นคือจะได้สถาปัตยกรรมของ Decoder จะเห็นว่าการเขียนโปรแกรมทั้งสองแบบมีความแตกต่างกัน วงจรที่ได้จากการสังเคราะห์ก็ต่างกัน แต่การทำงานของวงจรทั้งสองนั้นได้ผลลัพธ์ที่เหมือนกัน จะแตกต่างกันอยู่บ้าง ในเรื่องของความเร็วในการทำงานของวงจรซึ่งวงจรแบบแรกเป็นสถาปัตยกรรมของ Multiplexer นั้นจะทำงานช้ากว่า และในการทำโปรเจคในครั้งนี้ได้เลือกใช้ FPGAsของบริษัท XILINX ทั้งนี้ก็เนื่องมาจากได้รับความนิยมเป็นอย่างมาก ทำให้สามารถหาอุปกรณ์และตัว FPGAs ได้ง่ายด้วย เราต้องพิจารณาด้วยว่าหลังจากได้วงจรจากการสังเคราะห์แล้ว เราต้องใช้โปรแกรม ของบริษัทผู้ผลิตในการจัดรูปแบบของวงจรและแปลงวงจรให้อยู่ในรูปแบบเลขฐานสอง (Binary Files) เพื่อนำไปโปรแกรมตัว FPGAs เพื่อใช้งานต่อไป ส่วนการเลือกเบอร์ของ FPGAs จะขึ้นอยู่กับขนาดของโปรแกรมที่เขียนว่าจะต้องใช้ทรัพยากรต่างๆภายในตัวFPGAsมากขนาดไหนซึ่งความหมายของทรัพยากรที่ว่า ตัวอย่างเช่น จำนวนพอร์ต อินพุท เอาท์พุท จำนวนวงจรลอจิกที่จะใช้ จำนวนขาทั้งหมดของ FPGAs ซึ่งจะต้องพิจารณาเป็นสำคัญว่าจะสามารถหาบอร์ดที่สามารถโปรแกรม FPGAs เบอร์ดังกล่าว ได้หรือไม่

สรุปในโปรเจคครั้งนี้ได้เลือกเบอร์ XC4010e PC84 โดยสามารถยืมอุปกรณ์ต่างๆรวมทั้งตัว FPGAs และบอร์ดสำหรับโปรแกรม ได้จากศูนย์อิเลคทรอนิคส์และคอมพิวเตอร์แห่งชาติ (NECTEC)

2. การสังเคราะห์วงจร กระบวนการสังเคราะห์วงจรจะมี อินพุท ก็คือ โปรแกรมภาษาวีเอชดีแอล ที่เขียนบรรยายระบบในเชิงพฤติกรรมที่ผ่านการ ทดสอบ ในระดับ Source Code แล้วและเอาท์พุทของกระบวนการก็คือ วงจรดิจิตอล ที่ในระดับเกทซึ่งอาจจะประกอบด้วย FlipFlop, Multiplexer , Decoder, Adder, Buffer และ เกทชนิดต่างๆ โดยเครื่องมือที่ใช้สังเคราะห์วงจรมีหลายแบบ ตัวอย่างเช่น Leonado Spectrum และ Simplify ซึ่งในแต่ละแบบจะมีข้อดี ข้อเสียที่แตกต่างกัน ในการทำโปรเจคครั้งนี้ได้เลือกใช้ Leonado Spectrum Version 1999 ซึ่งเป็นของบริษัท Exemplar เหตุผลที่เลือกใช้เนื่องจากเป็นโปรแกรมที่ใช้กันอย่างแพร่หลาย และสามารถยืมโปรแกรมและส่วน Hard Lock ได้จาก NECTEC

วิธีการสังเคราะห์วงจร

2.1 การเซ็ตค่าต่างๆก่อนทำการสังเคราะห์

- การเลือกใช้อุปกรณ์โดยจะเลือกใช้ CPLD หรือ FPGAs ทำการเลือกบริษัทผู้ผลิต FPGAs ในที่นี่คือ XILINX จากนั้นเลือกตระกูลของ FPGAs ซึ่งก็คือ XC4000E ทำการเลือกเบอร์ FPGAs ในที่นี้คือ XC4010e PC84 สุดท้ายจะเลือกความเร็ว (Speed) ของ FPGAs
- การเลือกแฟ้มข้อมูลเอาท์พุท (Output Files) ซึ่งจะมีอยู่หลายแบบเช่น EDIF File, XNF File และ VHDL Fileทั้งนี้ขึ้นอยู่กับจุดประสงค์การนำเอาท์พุทที่ได้ไปใช้งานในขั้นตอนต่อไป หากสังเคราะห์วงจรแล้วจะนำผลลัพธ์ที่ได้ไปทดสอบแบบ gate sim ก็จะต้องเลือกเอาท์พุทเป็น VHDL File หากต้องการนำผลลัพธ์ที่ได้ไปทำขั้นตอนต่อไปแล้วโปรแกรมลงบน FPGAs ก็จะต้องเลือกเอาท์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

พุทเป็น XNF File หากต้องการนำผลลัพธ์ที่ได้ไปสร้าง Layout เพื่อทำการสร้างวงจรรวมก็จะต้องเลือกเอาท์พุทเป็น EDIF File

- การเลือก Contraint ซึ่งเป็นการเลือกความถี่ใช้งานของวงจรที่ได้จากการสังเคราะห์
- การเลือกแฟ้มข้อมูลอินพุทที่จะทำการสังเคราะห์

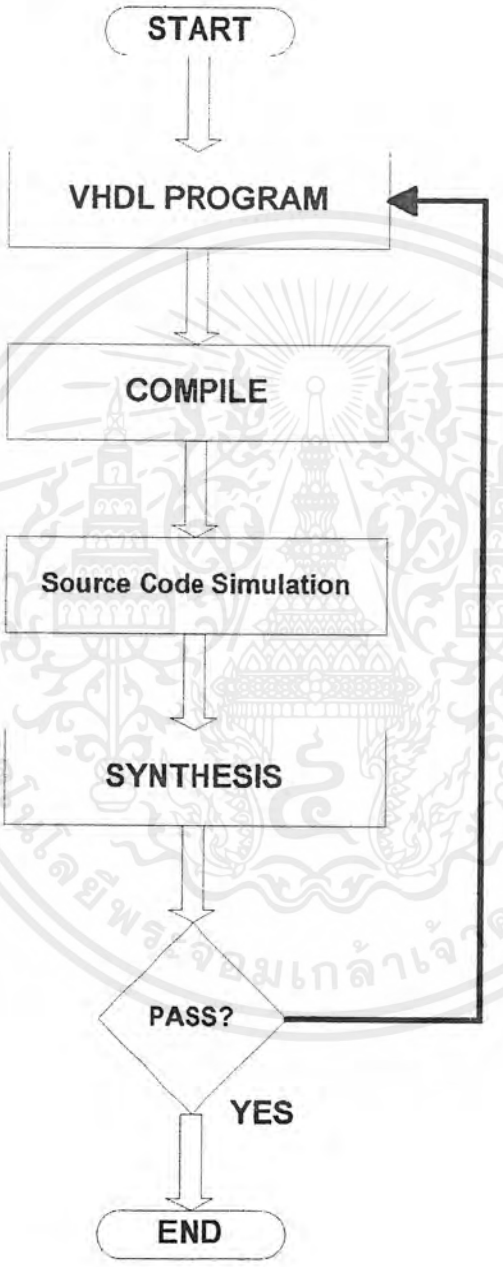
2.2 การสังเคราะห์วงจร แฟ้มข้อมูลอินพุทที่เลือก ซึ่งก็คือ ส่วนที่จะทำการสร้างเป็นวงจร ซึ่งผ่านการคอมไพล์ และการทดสอบในระดับsource code แล้วจากนั้นนำทำการสังเคราะห์ หากไม่สามารถสังเคราะห์เป็นวงจรได้ โปรแกรมก็จะปรากฏข้อผิดพลาดขึ้น จะต้องทำการแก้ไข โปรแกรมภาษาวีเอชดีแอลใหม่ซึ่งขั้นทั้งหมดเป็นไปตาม

ไดอะแกรมดังรูปที่ 7.3

3. การ Place and Routing จะมีหน้าที่นำวงจรที่ได้จากการสังเคราะห์จัดวางตำแหน่งลงบนส่วนต่างๆของ FPGAs แล้วจะทำหน้าที่เลือกเส้นทางการเชื่อมต่อกันของส่วนต่างๆในวงจรเข้าด้วยกัน โดยมีอินพุท คือ แฟ้มข้อมูลของวงจรที่ได้จากการสังเคราะห์ (XNF File) และเอาท์พุทของกระบวนการสามารถนำไปโปรแกรมตัว FPGAs ได้ ก่อนที่จะกระทำในกระบวนการนี้ หากเราต้องการนำเอาท์พุทไปทดสอบ อีกครั้ง ก็จะต้องเลือก ModelSim VHDL ซึ่งกระบวนการนี้จะสร้างแฟ้มข้อมูลสองแฟ้ม คือ timesim.VHD และ timesimd.SDF ซึ่งแฟ้มข้อมูลทั้งสองแฟ้ม จะนำไปทดสอบในระดับ TimeSim โปรแกรมที่ใช้ทำกระบวนการ นี้ก็คือ โปรแกรม Design Manager ของ XILINX FOUNDATION

4. การทดสอบโปรแกรมแบบ Time Sim เป็นการนำเอาท์พุทจากกระบวนการ Place and Routing คือ แฟ้มข้อมูล timesim.VHD และ timesim.SDF ไปทดสอบโปรแกรมบน ModelSim การทดสอบโปรแกรมแบบ TimeSim มีความแตกต่างจากการทดสอบโปรแกรมระดับ Source Code เนื่องจากมีการจำลองแทนวงจรส่วนต่างๆด้วยอุปกรณ์มาตรฐาน โดยอุปกรณ์เหล่านี้จะถูกเก็บอยู่ใน Library ในส่วนโปรแกรมการทดสอบ ซึ่งจะมีคุณสมบัติคล้ายกับวงจรลอจิกจริงๆ ซึ่งจะมีการหน่วงเวลา (Delay Time) เกิดขึ้นในวงจรด้วย ซึ่งผลลัพธ์ที่ได้จากการทดสอบ จึงเปรียบเสมือนการต่อวงจรภายนอกจริงๆ เป็นการจำลองการทำงานของ FPGAs หลังจากถูกโปรแกรมแล้ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



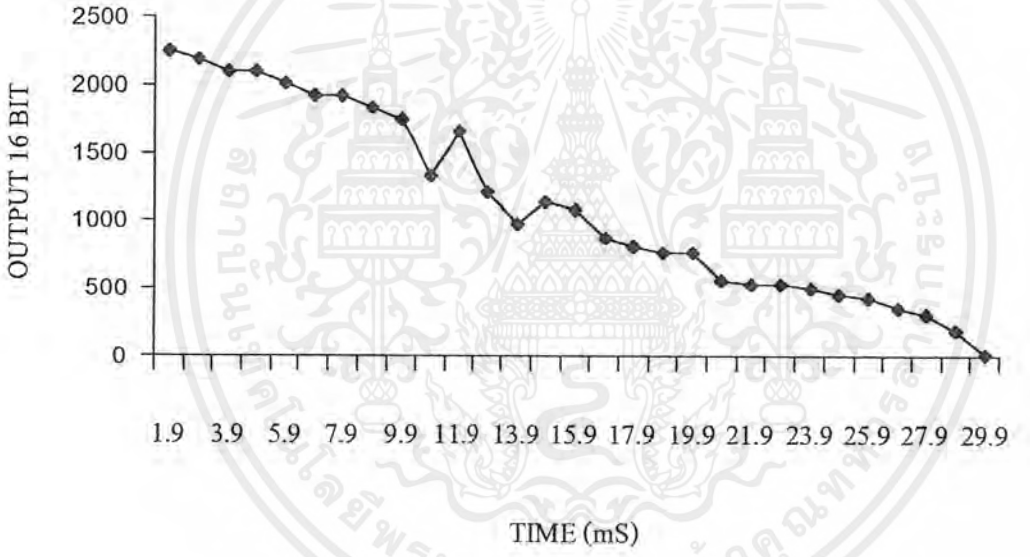
รูปที่ 7.3 การสังเคราะห์วงจร

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรมการทำงานของวงจรของโปรเจคได้ออกแบบเป็น4แบบ โดยทำการเปลี่ยนแปลงค่า Kp และ Ki ของ Fuzzy Algorithm ดังต่อไปนี้

แบบที่ 1

STATUS	Kp	Ki	Kp/Ki
D	30	15	2
C	21	12	1.75
B	15	10	1.5
A	10	8	1.25

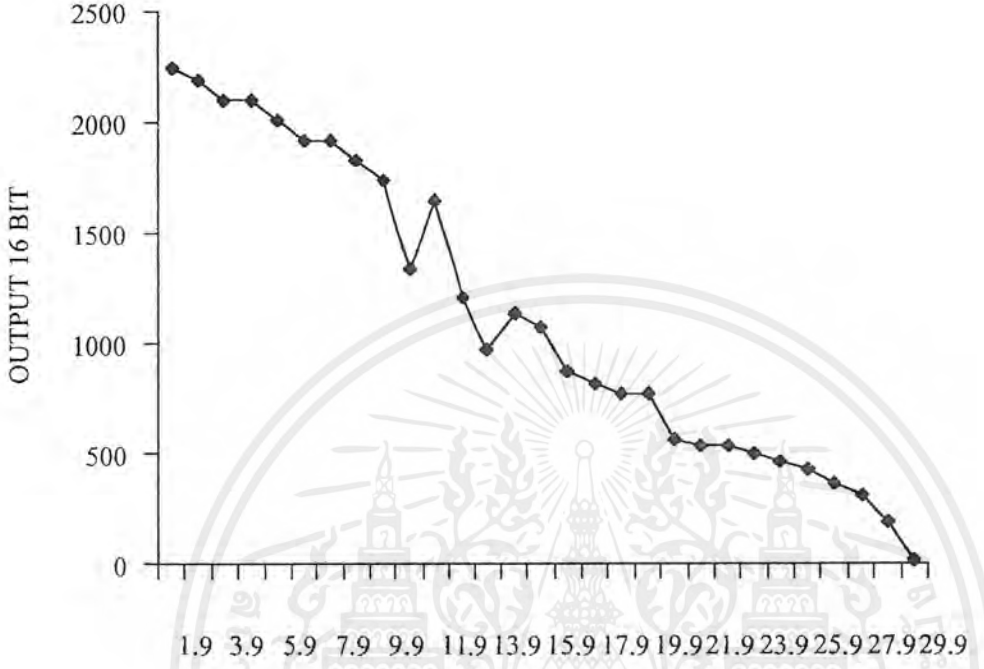


รูปที่ 7.4 การ Simulate แบบที่ 1

แบบที่ 2

STATUS	Kp	Ki	Kp/Ki
D	24	8	3
C	20	10	2
B	16	10	1.6
A	20	12	1.67

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

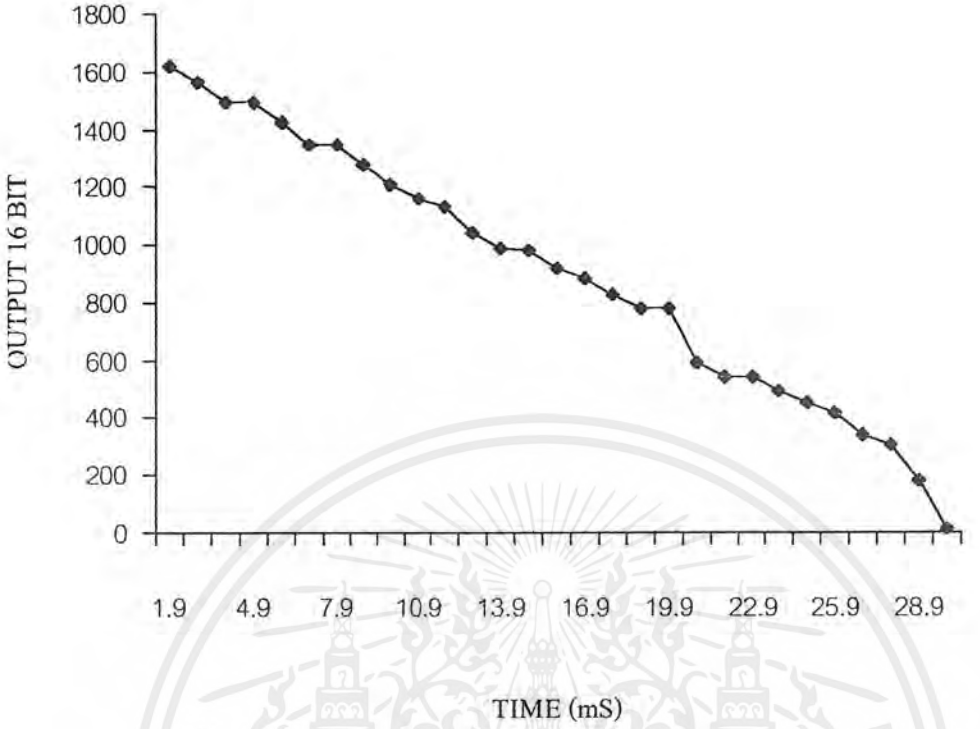


รูปที่ 7.5 การ Simulate แบบที่ 2

แบบที่ 3

STATUS	Kp	Ki	Kp/Ki
D	27	9	3
C	20	10	2
B	16	10	1.6
A	15	8	1.875

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

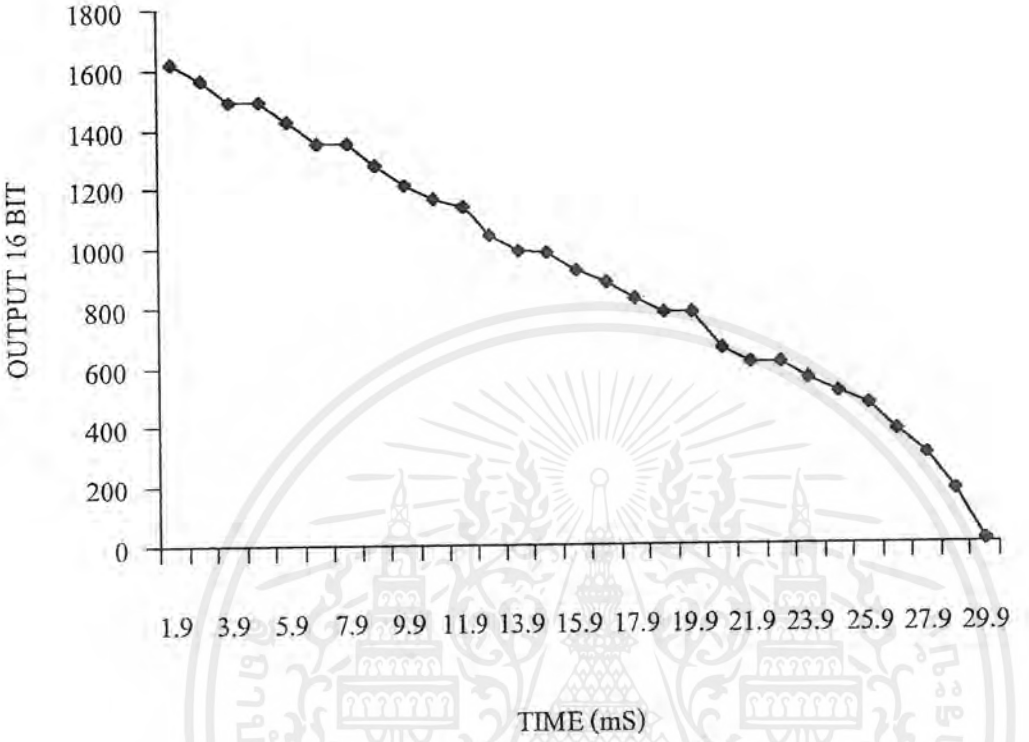


รูปที่ 7.6 การ Simulate แบบที่ 3

แบบที่ 4

STATUS	Kp	Ki	Kp/Ki
D	27	9	3
C	20	10	2
B	16	10	1.6
A	16	9	1.78

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 7.7 การ Simulate แบบที่ 4

ผลของการทดสอบแสดงในกราฟรูปที่ 7.4,7.5,7.6และ 7.7 ซึ่งเอาท์พุทที่ได้จากกราฟจะนำมาวิเคราะห์ คุณสมบัติของวงจรในระบบควบคุมอุณหภูมิ

หากผลลัพธ์ที่ได้จากกระบวนการทดสอบ TimeSim ไม่ถูกต้องจะต้องย้อนกลับไปทำตั้งแต่ขั้นตอนแรก คือการเขียนภาษาวีเอชดีแอลใหม่ และทำขั้นตอนต่างๆตามลำดับต่อไป

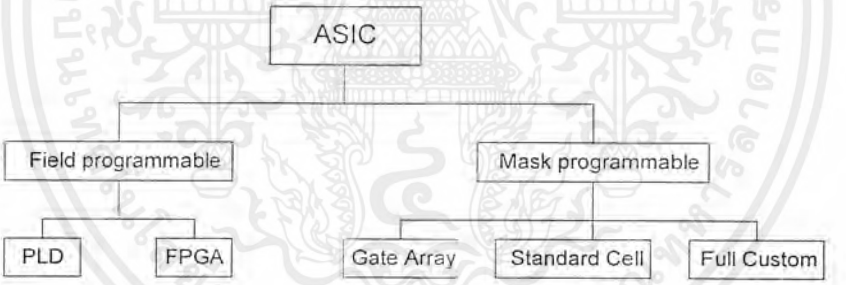
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 8

โครงสร้าง FPGA และการ Implementation

FPGA (Field Programmable Gate Array)

ความก้าวหน้าของอุตสาหกรรมอิเล็กทรอนิกส์ปัจจุบันทำให้เกิดการพัฒนาความสามารถของอุปกรณ์ต่างๆ มากมายซึ่งทำให้เกิดการลดค่าใช้จ่าย การสิ้นเปลืองพลังงานและขนาด ในขณะเดียวกันก็มีการเพิ่มประสิทธิภาพและระดับความเชื่อถือได้ของวงจรรวมที่สูงขึ้นเห็นได้ชัดจากเทคโนโลยีไมโครโพรเซสเซอร์และหน่วยความจำปัจจุบัน ทุกๆ ครั้งที่มีการพัฒนาขึ้นทำให้เกิดช่องว่างวงจรรวมและไอซีมาตรฐานมากขึ้น ในการพัฒนาเพิ่มความหนาแน่นและจำนวน ฟังก์ชั่นลอจิกที่เหมาะสม นักออกแบบอุปกรณ์ทางด้านดิจิตอลได้พิจารณาถึงการผลิตให้ขนาดมากๆ และการผลิตวงจรรวม (ASIC : Application Specific Integrated Circuit) ซึ่งวงจรรวม จะแบ่งตามการสร้างออกเป็น 2 กลุ่มคือ Filed programmable และ Mask programmable ดังแสดงในภาพประกอบที่ 8.1



รูปที่ 8.1 ผังแสดงการแบ่งกลุ่มของวงจรรวม ASIC

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

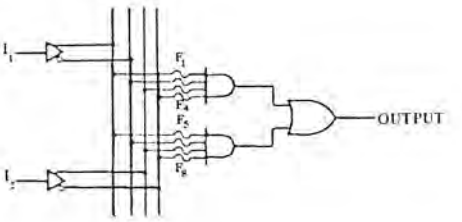
8.1 Filed programmable

อุปกรณ์วงจรรวม ASIC แบบ Filed programmable มีอยู่มากมายหลายชนิด แต่มีลักษณะการสร้างหรือกำหนดการทำงานของวงจรที่เหมือนกัน กล่าวคือ ผู้ใช้งานสามารถออกแบบและสร้างวงจรที่ต้องการใช้ลงในตัวอุปกรณ์ได้เองโดยไม่ต้องไปโรงงานเพื่อผลิต โดยเฉพาะอย่างยิ่งในปัจจุบันนี้มีเครื่องมือที่ใช้ช่วยในการออกแบบ และสร้างวงจรร่วมกับไมโครคอมพิวเตอร์ที่มีความสามารถสูงในการพัฒนาตั้งแต่ขั้นการออกแบบ การจำลองการทำงาน จนถึงจัดสร้างวงจรลงในอุปกรณ์ รวมทั้งอุปกรณ์ Filed programmable เหล่านี้สามารถหาซื้อได้ง่ายทำให้การสร้างวงจรอิเล็กทรอนิกส์จนถึงระบบไมโครโพรเซสเซอร์ทันมาใช้อุปกรณ์จำพวกนี้ เป็นอุปกรณ์ประกอบในวงจรแทนอุปกรณ์ย่อยๆ แยกชิ้น (Discrete componnt) มาก

8.1.1 พีแอลดี (PLD : Programmable Logic Device)

ภายในอุปกรณ์ พีแอลดีถูกเตรียมเป็นวงจรพื้นฐานทางด้านลอจิกต่อกันอยู่เป็นกลุ่มมีทั้งวงจรคอมบิเนชั่น (Combination) และแบบซีเควนเชียล (Sequential) ซึ่งมีส่วนประกอบเป็นวงจรภายใน เทคโนโลยีของวงจรที่ใช้สร้างพีแอลดีมีทั้ง ทีทีแอล (TTL) อีซีแอล (ECL) และ ซีมอส (CMOS) ตามความเหมาะสมของแต่ละระบบ อุปกรณ์พีแอลดีทุกชนิดมีหลักการพื้นฐานของวงจรภายในที่เหมือนกัน โดยมีวงจรหลักเป็นวงจรคอมบิเนชั่นที่ให้ผลเป็นผลคูณร่วมบวก (Sum of product) ประกอบไปด้วยชุดของแอนด์เกตที่ต่อร่วมกับออร์เกตการโปรแกรมคือ การเลือกว่าจะให้มีการต่ออินพุทภายในของแอนด์เกตกับสัญญาณอินพุทใดบ้างซึ่งมีทั้งจากภายนอกและสัญญาณป้อนกลับจากเอาท์พุทภายในเอง การติดต่ออินพุทของออเกตกับเอาท์พุทของ แอนด์เกต ตัวต่างๆ วิธีการเลือกหรือการโปรแกรมทางกายภาพ อินพุทต่างๆ ของอุปกรณ์ทุกตัวจะถูกต่อผ่านฟิวส์เข้ากับแหล่งสัญญาณ ซึ่งถ้าไม่ต้องการใช้สัญญาณใดจะตัดฟิวส์ทำให้สามารถโปรแกรมได้ครั้งเดียวอุปกรณ์พีแอลดีบางชนิดใช้มอสทรานซิสเตอร์แทนฟิวส์ทำให้สามารถโปรแกรมโดยใช้กระแสไฟฟ้าและสามารถลบและโปรแกรมใหม่เข้าไปได้อีก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 8.2 แสดงวงจรพื้นฐานของอุปกรณ์พีแอลดีซึ่งอยู่ในรูปผลคูณร่วมบวก

8.1.2 เอฟพีจีเอ (FPGA : Field Programmable Gate Array)

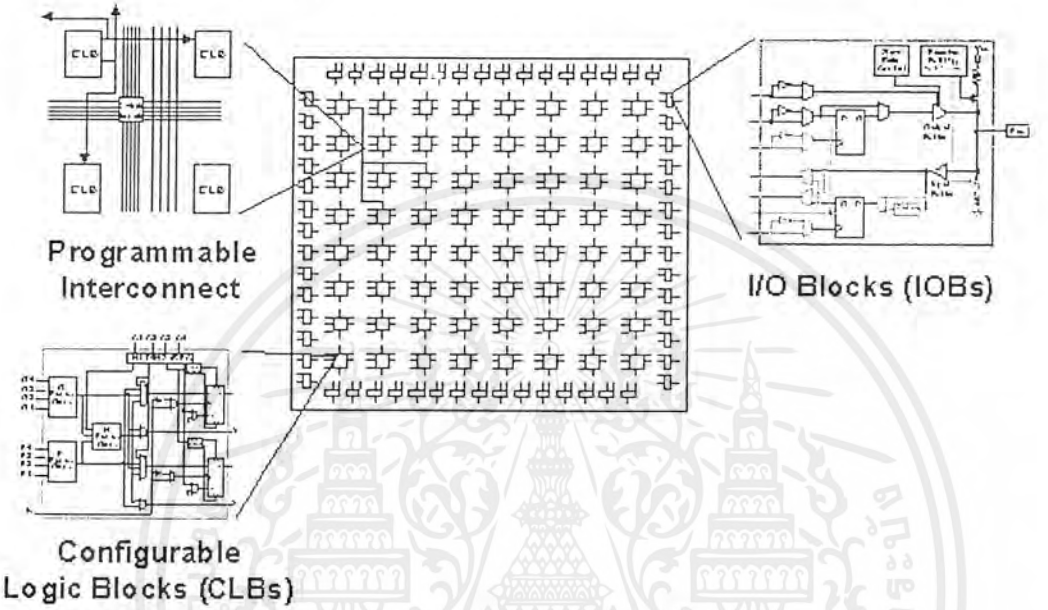
เป็นอุปกรณ์ที่ถูกพัฒนาต่อจากอุปกรณ์แอลซีเอของบริษัทไซลิงซ์ (XILING Inc.) โดยมีประสิทธิภาพการทำงานและมีปริมาณความหนาแน่นของเกตสูง สามารถจะกำหนดฟังก์ชั่นการทำงานได้ความต้องการของผู้ใช้โดยผ่านการโปรแกรมเอฟพีจีเอ ได้รวบรวมข้อดีทั้งหมดของการทำคัสตัมวีแอลเอสไอ (Custom VLSI) มารวมไว้ทั้งหมดได้แก่ การออกแบบการผลิต , ลดเวลาที่จะส่งตัวผลิตภัณฑ์ออกตลาด ซึ่งเป็นประโยชน์ต่อการผลิตวงจรเป็นอย่างมาก นักออกแบบเพียงกำหนดฟังก์ชันการทำงานของวงจร ดังนั้นการออกแบบวงจรโดยใช้เอฟพีจีเอ สามารถออกแบบและทดสอบภายในเวลาเพียง 2-3 วัน เท่านั้น ตรงกันข้ามกับการออกแบบโดยใช้เกตอาเรย์ ซึ่งใช้เวลาหลายสัปดาห์การเปลี่ยนแปลงแก้ไขแบบก็เช่นเดียวกัน จากประโยชน์ของเอฟพีจีเอ ดังกล่าวมา ทำให้เกิดการประหยัดค่าใช้จ่ายเป็นอย่างมาก เพราะได้ความเสี่ยงในการที่จะต้องแก้ไขตัววงจร การเลื่อนเวลาการออกผลิตภัณฑ์ ลดค่าเอ็นอาร์อี (NRE : Nonrecurring Engincering Cost) ลงไปด้วย

8.2 สถาปัตยกรรมภายในของเอฟพีจีเอตระกูล XC 4000

FPGA (Field Programmable Gate Array) เป็นชิปที่มีสนามวงจรขนาดใหญ่อยู่ภายใน ที่เราสามารถนำมาใช้และออกแบบวงจรต่างๆ ได้ตามที่เราต้องการ โดยมีโปรแกรมสำเร็จรูปที่ใช้ในการออกแบบอยู่แล้ว มีความเร็วสูง มีสถาปัตยกรรมการออกแบบคล้ายๆ CPLD (Complex Programmable Logic Device) แต่มีส่วนประกอบที่ซับซ้อน และมีประสิทธิภาพมากกว่า ในการออกแบบวงจรสามารถทำได้ง่าย ทั้งในการเชื่อมต่อและการแก้ไข ดังนั้น FPGA จึงเหมาะสมสำหรับการออกแบบวงจรเป็นอย่างมาก สถาปัตยกรรมภายในของ FPGA แบ่งเป็น 3 ส่วน คือ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- *CLB (Configuration Logic Block)*
- *IOB (Input Output Block)*
- *Interconnect*



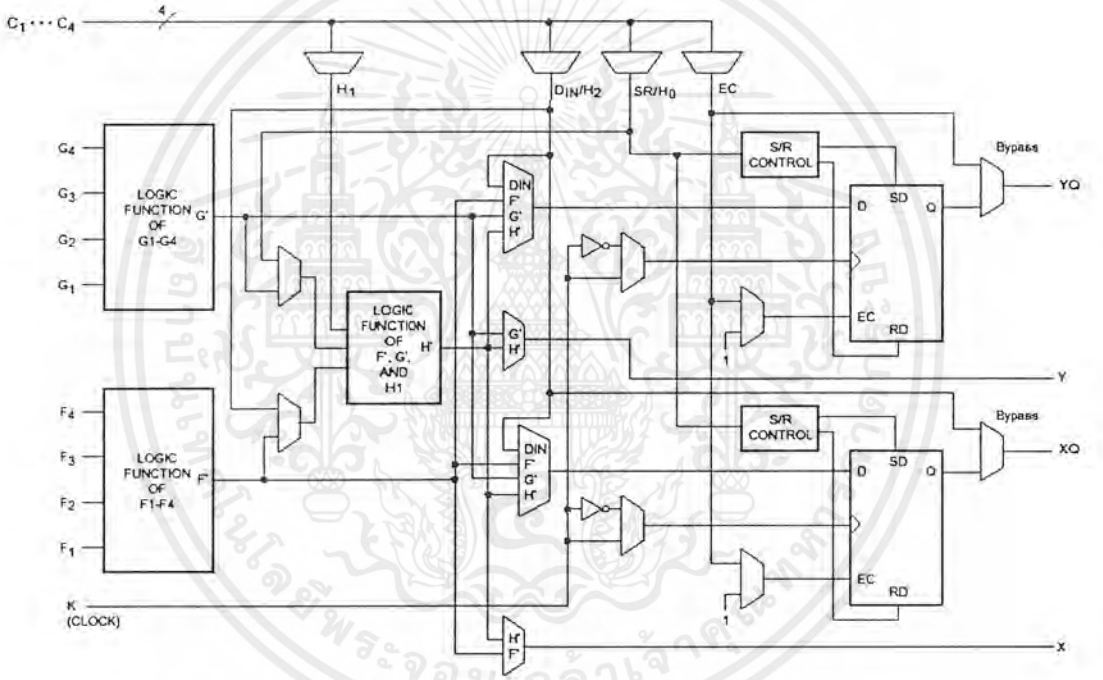
รูปที่ 8.3 แสดงวงจรพื้นฐานภายในไซลิ้งเอฟพีจีเอ

ซึ่งภายในมีสถาปัตยกรรมของ FPGA จะมีลักษณะเป็นตารางของลอจิกบล็อก (Logic block) และล้อมรอบไปด้วยบล็อกการเชื่อมของไอโอ (I/O Interface block) การเชื่อมต่อระหว่างซีแอลบี (CLB : Configuration Logic Block) และไอโอบี (IOB : Input Output Block) ทำได้โดยผ่านช่องที่ว่างพาดผ่านระหว่างแถว (Row) และคอลัมน์ (Column) มีการทำงาน เหมือนกันไมโครโพรเซสเซอร์ ตัวแอลซีเอจะทำงานได้ต้องใช้ Program-driven logic device หน้าที่ของซีแอลบีและไอโอบีแต่ละตัว การเชื่อมต่อภายใน (Interconnection) ถูกกำหนดไว้ในโปรแกรมคอนฟิกคูเรชัน (Configuration program) หรือเก็บไว้ในอีพรอม (EPROM) ภายในแอลซีเอ (LCA : Logic Cell Array) โปรแกรมจะถูกโหลดเข้าสู่แอลซีเอเมื่อมีการจ่ายไฟ (Power-up) โดยทางคำสั่ง (Command) ซึ่งเป็นส่วนหนึ่งของการเริ่มต้นระบบ (System initialization) ประสิทธิภาพของแอลซีเอกำหนดโดย ความเร็วของลอจิกส่วนประกอบหน่วยความจำและการโปรแกรมเชื่อมต่อต่างๆ ความเร็วของอัตราของระบบสัญญาณนาฬิกา (System clock rate) ถูกกำหนดด้วย ทอกเกิลฟลิปฟลอปสำหรับการประยุกต์ใช้โดยทั่วไปจะอยู่ที่ประมาณ 1/3 ถึง 1/2 ค่าสูงสุดของทอกเกิลเกต (Maximum toggle gate)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

8.2.1 ซีแอลบี (CLB : Configuration Logic Block)

ภายใน LCA (Logic Cell Array) คือเมทริกซ์ของซีแอลบีแต่ละตัวประกอบด้วยหน่วยของคอมบิเนชั่นลอจิกที่สามารถโปรแกรมได้ (program combination logic) และส่วนของเรจิสเตอร์เก็บข้อมูล (Storage register) ส่วนของวงจรคอมบิเนชั่นลอจิกสามารถใช้สร้างวงจรทางด้านฟังก์ชั่นบูลีนของดินพุท ส่วนเรจีสเตอร์รับค่าจากส่วนคอมบิเนชั่นหรือโดยตรงจากเอาท์พุทของซีแอลบี สามารถขับวงจรคอมบิเนชั่นลอจิกโดยตรงผ่านเส้นทางเดินย้อนกลับ (Feedback path)

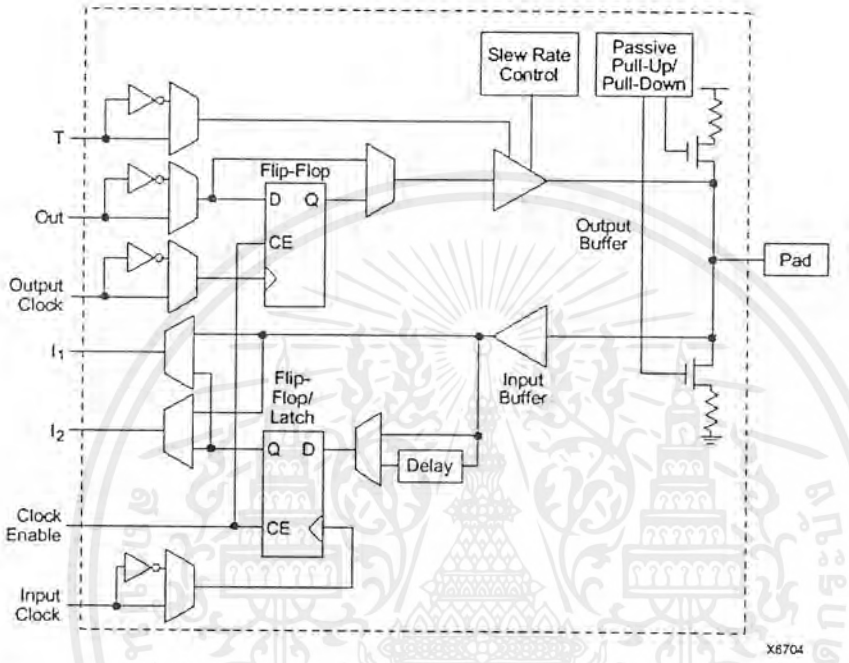


รูปที่ 8.4 แสดงผังวงจรภายในของซีแอลบีของเอฟพีจีตระกูล XC4000

8.2.2 ไอโอบี (IOB : Input Output Block)

เป็นส่วนติดต่อกับวงจรภายนอกของแอลซีเอสร้างมาจากส่วนของอุปกรณ์อินพุท/เอาท์พุทที่สามารถโปรแกรมได้ (Programmable Input/Output device) แต่ละตัวสามารถโปรแกรมได้อย่างอิสระโดยจะให้เป็นอินพุท/เอาท์พุท 3 สถานะ หรือไอโอแบบสองทิศทางก็ได้ โดยอินพุทสามารถโปรแกรมให้รู้จักทั้งระดับสัญญาณทีทีแอลและซีมอสเทรดโฮล ของไอโอบี แต่ละตัวมีฟลิปฟลอปสามารถใช้เป็นบัฟเฟอร์สำหรับอินพุทและเอาพุท

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

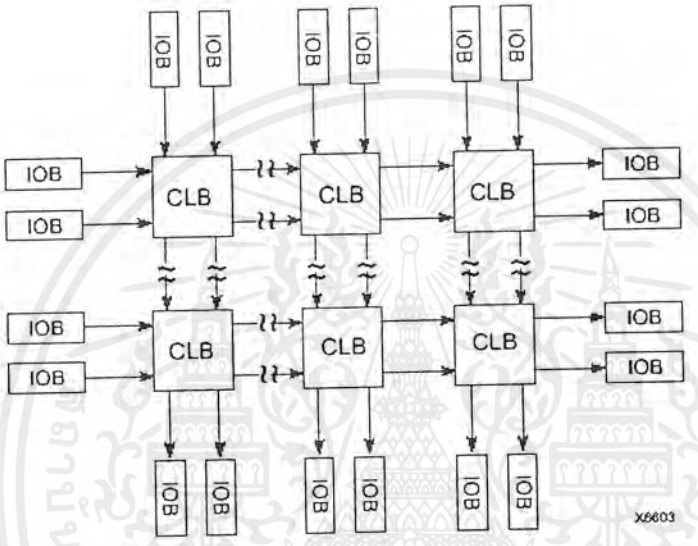


รูปที่ 8.5 แสดงผังวงจรของไอโอบีภายในเอฟพีจีเอตระกูล XC4000

8.2.3 อินเตอร์คอนเน็ค (Interconnect)

ความยืดหยุ่นของการใช้แอลซีเอมาทำเป็นอุปกรณ์ขึ้นอยู่กับการโปรแกรม ทรัพยากรต่างๆ ที่อยู่ภายในเข้าด้วยกันการที่จะควบคุมการเชื่อมต่อระหว่างจุดสองจุดภายในชิปเหมือนกับเกตอาเรย์ทั่วๆ ไป การเชื่อมต่อภายในแอลซีเอประกอบด้วยเน็ตเวิร์ค 2 ทิศทาง แถวละคอลัมน์ซึ่งวางอยู่ระหว่าง CLB programmable switch จะทำการเชื่อมต่ออินพุทและเอาท์พุทของไอโอบีและซีแอลบีที่จุดต่อร่วมระหว่างแถวกับคอลัมน์สามารถสลับสัญญาณจาก เส้นทางไปยังส่วนต่างๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 8.6 แสดง Interconnect ระหว่างไอโอบีกับซีแอลบีของเอฟพีจีเอตระกูล XC4000

คุณสมบัติทั่วไปของเอฟพีจีเอตระกูล XC4000

1. เป็นอุปกรณ์รุ่นที่สามของเอฟพีจีเอ
   - มีฟลิปฟลอปเป็นจำนวนมาก
   - ในการผลิตฟังก์ชั่นของการทำงานความยืดหยุ่นสูง
   - มีจำนวนเกตภายในจำนวน 2,000-10,000 เกต
   - เพิ่มความสามารถพิเศษของเรจีสเตอร์และอินพุท/เอาท์พุท
   - มีค่าแฟนเอาท์ (fan-out) สูง
   - มีบัสภายใน 3 สถานะ
   - ทำงานกับสัญญาณทีทีแอลและซีมอส

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- มีแรมภายในความเร็วสูง (<25 Ns)
- ใช้กับงานที่ต้องการความเร็วสูง (ใช้งานได้ที่ความถี่ 70/100/125 MHz)
- มี Wide edge decoder
- เส้นทางการเชื่อมต่อ (Interconnect line) เป็นแบบลำดับชั้น
- มีการกระจายกำลังงานของสัญญาณต่ำ

2. มีสถาปัตยกรรมภายในที่ยืดหยุ่น
   - มีลอจิกบล็อคและไอโอบล็อกที่สามารถโปรแกรมได้
   - มีอินเตอร์คอนเน็คและ Wide decoder ที่โปรแกรมได้
3. ทำกระบวนการซับไมครอนชนิดซีมอสได้
   - มีลอจิกและอินเตอร์คอนเน็คที่มีความเร็วสูง
   - ใช้กำลังงานต่ำ
4. คุณลักษณะทาง System-Oriented
   - รองรับมาตรฐาน IEEE 1149.1 ในการทำ boundary-scan logic
   - สามารถโปรแกรมค่า output slew rate ได้
   - สามารถโปรแกรมให้อินพุทมีลักษณะพูลอัพ (Pull-up) หรือพูลดาวน์ (Pull-down) เรจิสเตอร์ได้
   - ให้กระแสเอาท์พุทได้ตั้งแต่ 12-24 มิลลิแอมป์ (ขึ้นอยู่กับแต่ละรุ่น)
5. ทำการโหลดเอาแฟ้มข้อมูลประเภทไบนารี
   - ไม่จำกัดจำนวนครั้งในการโปรแกรมซ้ำ
   - มีโหมดในการโปรแกรมให้เลือก 6 โหมด
6. มีโปรแกรมช่วยพัฒนาได้แก่ XACT Development System (ปัจจุบัน Foundation series) ที่ทำงานบนคอมพิวเตอร์รุ่นต่างๆ เช่น '486/Pentiums,NEC PC,Apollo,Sun-4,HP700
   - สามารถติดต่อกับโปรแกรมอื่นๆได้ เช่น Viewlogic,Mentor Graphic และ OrCAD เป็นต้น
   - มีโปรแกรมการวางและเชื่อมโยงอุปกรณ์ภายในแบบอัตโนมัติ (automatic place and routing) ที่ครบสมบูรณ์
   - มี Interactive Design Editor ที่ใช้สำหรับการทำ optimization
   - มี 288 มาโคร 34 ฮาร์ดมาโคร และ แรม/รอมคอมพายเลอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 8.1 แสดงรายละเอียดของอุปกรณ์ภายในเอฟพีจีเอตระกูล XC4000

Device	Max Logic Gate (No RAM)	Max.RAM Bits (No Logic)	Typical Gate Range (Logic and RAM)*	CLB Matrix	Total Logic Block	Number Of Flip-Flops	Max. Decode Input Per side	Max. User I/O
XC4003E	3,000	3,200	2,000-5,000	10×10	100	360	30	80
XC4005E/L	5,000	6,272	3,000-9,000	14×14	196	616	42	112
XC4006E	6,000	8,192	4,000-12,000	16×16	256	768	48	128
XC4008E	8,000	10,368	6,000-15,000	18×18	324	936	54	144
XC4010E/L	10,000	12,800	7,000-20,000	20×20	400	1,120	60	160
XC4013E/L	13,000	18,432	10,000-30,000	24×24	576	1,536	72	192
XC4020E	20,000	25,088	13,000-40,000	28×28	784	2,016	84	224
XC4025E	25,000	32,768	15,000-45,000	32×32	1,024	2,560	96	256
XC4028EX/XL	28,000	32,768	18,000-50,000	32×32	1,024	2,560	96	256
XC4036EX/XL	36,000	41,472	22,000-65,000	36×36	1,296	3,168	108	288
XC4044EX/XL	44,000	51,200	27,000-80,000	40×40	1,600	3,840	120	320
XC4052XL	52,000	61,952	33,000-100,000	44×44	1,936	4,576	132	352
XC4062XL	62,000	73,728	40,000-130,000	48×48	2,304	5,376	144	384
	Larger Device Available in the First Half of 1997							

*Max value of Typical Gate Range inclide 20-30% of CLBs used as RAM

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

8.3 การโปรแกรมเอฟพีจีเอตระกูล XC4000

คือกระบวนการในการโหลดข้อมูลในโปรแกรมไปยังแอลซีเอ เพื่อกำหนดหน้าที่ของการทำงานในแต่ละบล็อกภายใน และการเชื่อมต่อ ซึ่งเอฟพีจีเอตระกูล XC4000 จะต้องใช้ข้อมูลเกี่ยวกับการดปรแกรมประมาณ 350 บิตต่อซีแอลบี โดยแต่ละบิตจะบอกถึงสถานะของหน่วยความจำสเตติกที่ควบคุมบิตในการควบคุมตารางฟังก์ชั่น (function table bit) และมัลติเพลกซ์อินพุทหรือการเชื่อมต่อกันระหว่างทรานซิสเตอร์

8.3.1 โหมดการโปรแกรม

เอฟพีจีเอในตระกูล XC4000 มีโหมดการโปรแกรม 6 โหมด ซึ่งแต่ละโหมดจะถูกกำหนดโดยจากบิตโหมด ได้แก่ บิต M0, M1 และ M2 โดยมีโหมดการโปรแกรมดังตารางที่ 8.2

ตารางที่ 8.2 แสดงรูปแบบของโหมดต่างๆ ในการโปรแกรมเอฟพีจีเอตระกูล XC4000

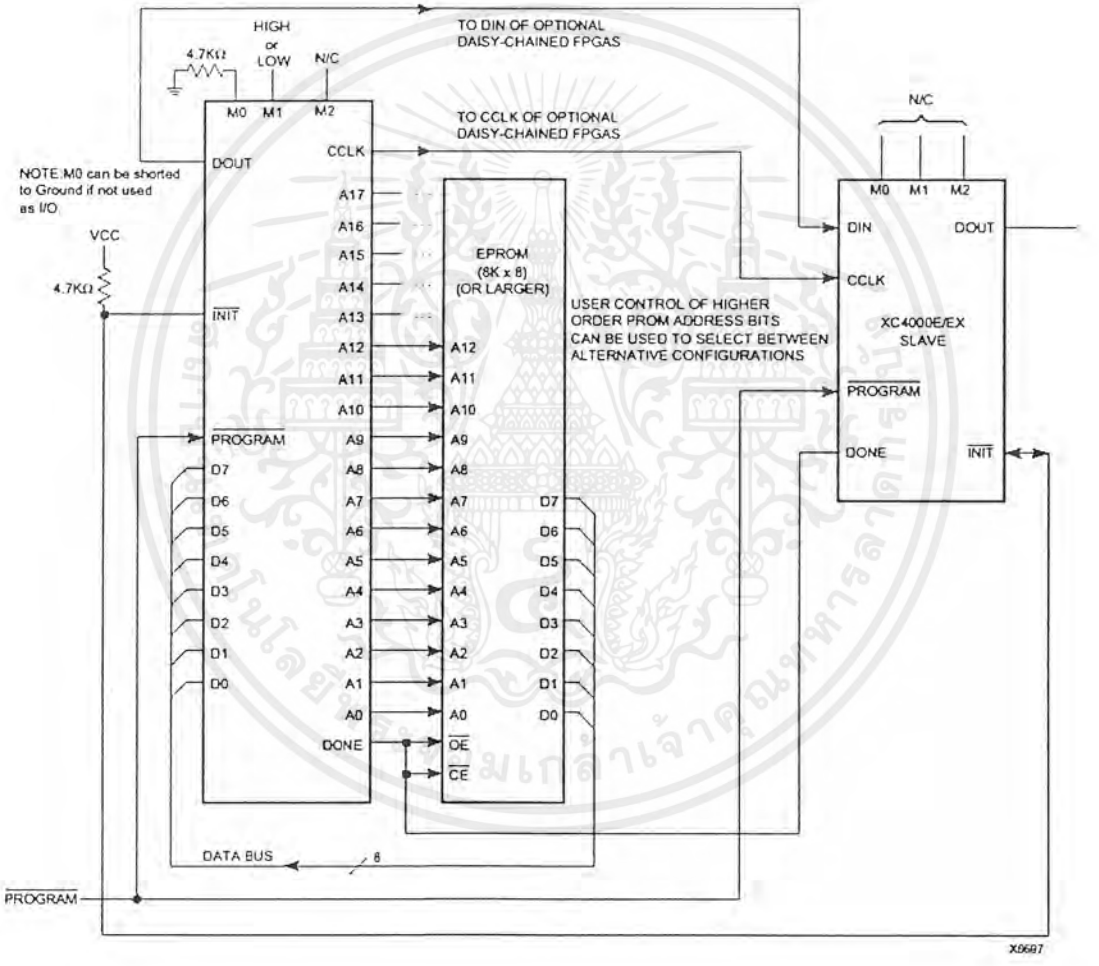
Mode	M2	M1	M10	Clock	Data
Master Serial	0	0	0	Output	Bit-Serial
Slave Serial	1	1	1	Input	Bit-Serial
Master parallel up	1	0	0	Output	Byte-Wide,00000
Master parallel down	1	1	0	Output	Byte-Wide,3FFFF
Peripheral Synch	0	1	1	Input	Byte-Wide
Peripheral Asynch	1	0	1	Output	Byte-Wide

8.3.1.1 โหมดหลัก (Master modes)

ในการทำงานในโหมดนี้ตัวแอลซีเอจะถูกโหลดข้อมูลโครงแบบ (configuration data) จากหน่วยความจำภายนอกเข้ามาโดยอัตโนมัติ มีโหมดที่แตกต่างกัน 8 โหมดโดยใช้ ช่วงเวลาภายในจ่ายให้ซีคล็อก (CCLK : Configuration clock) เพื่อที่จะเป็นฐานเวลาในการนำข้อมูลที่เข้ามาทางโหมดหลักแบบอนุกรม (serial master mode) และรับข้อมูลโครงแบบเข้ามาทางขาสัญญาณดีอิน (DIN : Data in) จากแหล่งสัญญาณซิงโครนัสเช่น Xilink serial configguration PROM, parallel master low and master high mode โดยจะรับข้อมูลแบบขนานมาจากบิต D0-D7 โดยสัมพันธ์กับแอดเดรสที่กำหนดโดยแอลซีเอดังภาพประกอบที่ 3.4 จะเป็นการแสดงตัวอย่างการเชื่อมต่อในโหมดหลักแบบขนาน (Master Parallel mode) โดยเริ่มต้นที่แอดเดรส 0000 ไบต์ข้อมูล (data byte)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จะถูกอ่านเข้ามาแบบขนานทุกๆ สัญญาณอาร์คคล็อก (RCLK : Read Clock) และส่งเข้าไปภายในแบบอนุกรมโดยอาศัยสัญญาณนาฬิกาโครงแบบ (Configuration clock)



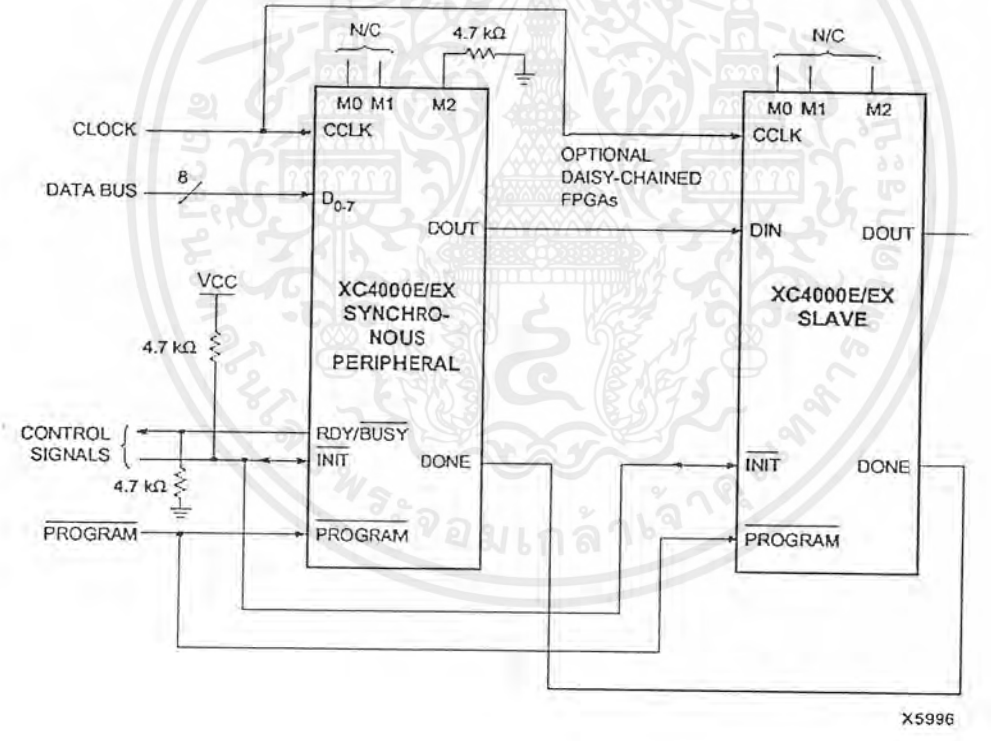
รูปที่ 8.7 แสดงผังวงจรการเชื่อมต่อเอฟพีจีเอในโหมดหลักแบบขนาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

8.3.1.2 เพอริเฟอรัลโหมด (Peripheral modes)

โหมดนี้จะรับข้อมูลเป็นไบต์จากบัส สถานะพร้อม (Ready) / กำลังทำงาน (Busy) จะมีอยู่เพื่อทำให้สัญญาณต่างๆ สื่อสารกันได้มีประสิทธิภาพมากขึ้น

ในอะซิงโครนัสโหมดนั้นออสซิสเตอร์ภายในจะสร้างสัญญาณ CCLK เพื่อรับข้อมูลที่เป็นไบต์ให้อยู่ในรูปแบบที่เหมาะสม และในซิงโครนัสโหมดนั้นสัญญาณนาฬิกาภายนอกจะเป็นตัวควบคุมการรับข้อมูลที่เป็นไบต์



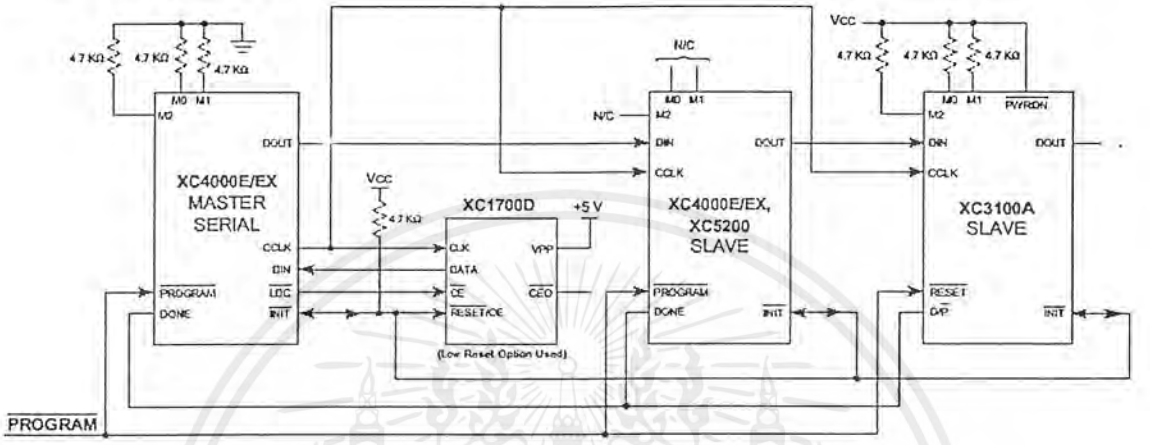
รูปที่ 8.3 แสดงผังวงจรการเชื่อมต่อเอฟพีจีเอในโหมดเพอริเฟอรัลแบบอะซิงโครนัส

(Asynchronous Peripheral mode)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

8.3.1.3 โหมดรองแบบอนุกรม (Slave Serial modes)

ในโหมดนี้แอลซีเอจะรับข้อมูลโครงแบบที่เป็นอนุกรมในขาขึ้นของสัญญาณ CCLK และหลังจากรับข้อมูลมาแล้วจะส่งข้อมูลเพิ่มเติมออกไปด้วยและแอลซีเอก็จะถูกการซิงโครไนซ์ในขาถัดไปของสัญญาณ CCLK



รูปที่ 8.9 แสดงผังวงจรการเชื่อมต่อเอฟพีจีเอในโหมดรองแบบอนุกรม

8.4 การใช้ความสามารถของแรมในเอฟพีจีเอตระกูล XC4000

แอลซีเอทำงานโดยใช้ตารางการค้นหา (Look-up table) ซึ่งจะทำการเก็บตารางที่ว่านี้ในสแตติกแรม ซึ่งจะถูกเขียนในระหว่างการโปรแกรมโครงแบบลงบนแอลซีเอและจะถูกอ่านในการโอเปอ-เรชั่น ดังนั้นแรมภายในจึงควรถูกรวมไว้ในการออกแบบของผู้ใช้ด้วย

หน้าที่ของแรมในเอฟพีจีเอตระกูล XC4000 มีหน้าที่คล้ายแรมโดยทั่วๆ ไป เช่น เอฟไอเอฟโอ (FIFO : First In First Out) แอลไอเอฟโอ (LIFO : Last In First Out) เรจิสเตอร์ไฟล์รวมทั้งแอปพลิเคชั่นบางอย่างเช่น เรจิสเตอร์เลื่อนข้อมูล (Shift register) แรมของเอฟพีจีเอตระกูล XC4000 มีความเร็วสูงเสมือนเอสแรม (SRAM) จึงไม่จำเป็นต้องคำนึงถึงเวลาหน่วงของการเชื่อมต่อ (Interconnection delay)

ตารางที่ 8.3 จำนวนของแรมภายในเอฟพีจีเอตระกูล XC4000

RAM Module	Equivalent Logic	XC4003	XC4005	XC4010
16×1	4-input Function Generator (F or G)	200	392	800
32×1	Two-4-input Function Generator and One 3-input Function Generator (F+G+H)	100	196	400

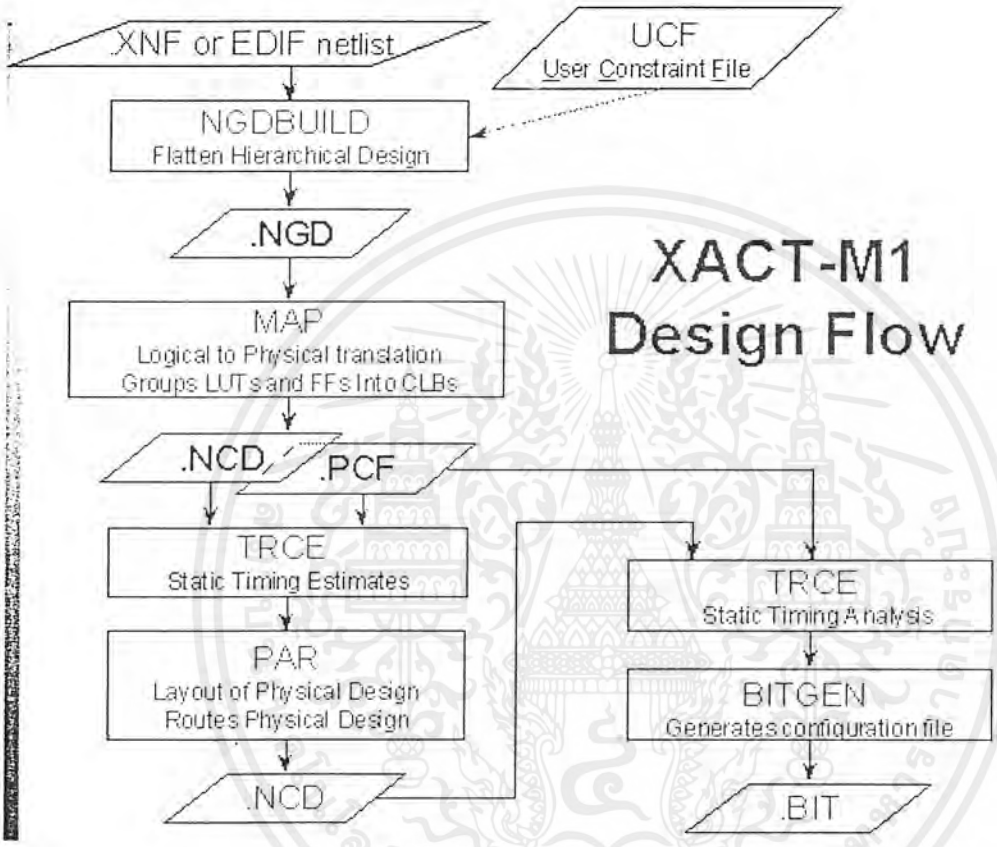
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 8.4 รายละเอียดของขาอุปกรณ์เอฟพีจีเอเบอร์ XC4010EPC84

Pad Name	Pin number	Pad Name	Pin number
VCC	P2	O(M1)	P30
I/O(A8)	P3	GND	P31
I/O(A9)	P4	I(M0)	P32
I/O(A10)	P5	VCC	P33
I/O(A11)	P6	I(M2)	P34
I/O(A12)	P7	I/O,PGCK2	P35
I/O(A13)	P8	I/O(HDC)	P36
I/O(A14)	P9	I/O(LDC)	P36
I/O,SGCK1(A15)	P10	I/O	P37
VCC	P11	I/O	P38
GND	P12	I/O	P39
I/O,PGCK1(A16)	P13	I/O	P40
I/O(A17)	P14	I/O(INIT)	P41
I/O,TDI	P15	VCC	P42
I/O,TCK	P16	GND	P43
I/O,TMS	P17	I/O	P44
I/O	P18	I/O	P45
I/O	P19	I/O	P46
I/O	P20	I/O	P47
GND	P21	I/O	P48
VCC	P22	I/O	P49
I/O	P23	I/O	P50
I/O	P24	I/O,SGCK3	P51
I/O	P25	GND	P52
I/O	P26	DONE	P53
I/O	P27	VCC	P54
I/O	P28	$\overline{\text{PROGRAM}}$	P55
I/O,SCGK	P29	I/O(D7)	P56
O(M1)	P30	I/O,PGCK	P57

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

8.5 Implementation Flow Overview

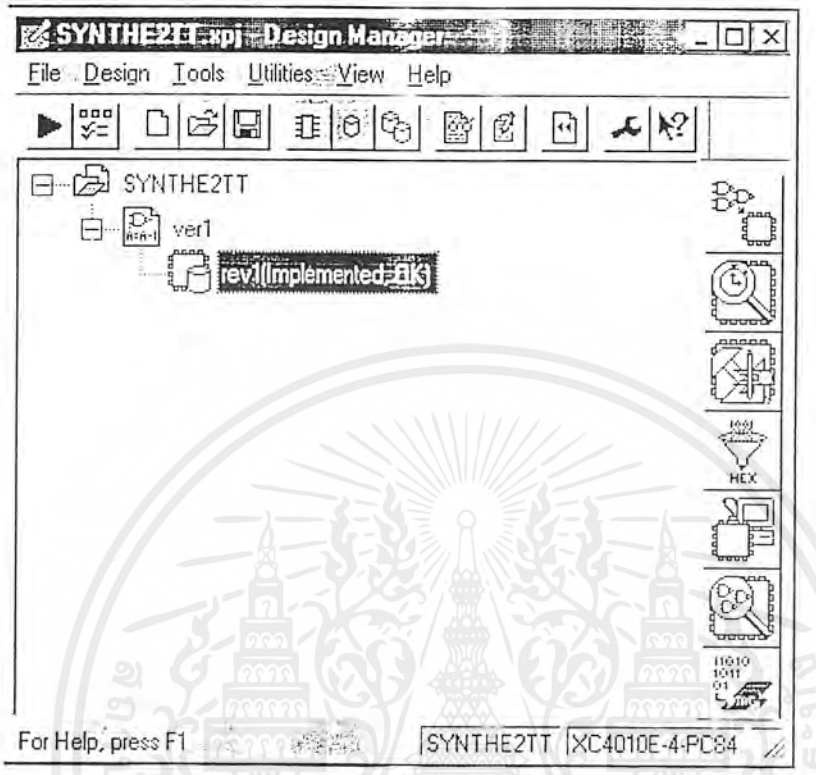


รูปที่ 8.10 แสดงขั้นตอนการออกแบบวงจรโดยใช้ FPGA

ขั้นตอน Place & Route โดยใช้ซอฟต์แวร์ของ XILINX Foundation 2.Li เมื่อทำการสังเคราะห์วงจรจากโปรแกรม Leonardo Spectrum 1999.1g โดยผลของการสังเคราะห์จะถูกเขียนไว้ในรูปของเน็ตลิสต์แบบ XNF (Xilinx Netlist Format) เพื่อที่จะนำไปผ่านขั้นตอน Place & Route ด้วยโปรแกรม XILINX Foundation M2.Li โดยมีขั้นตอนดังต่อไปนี้

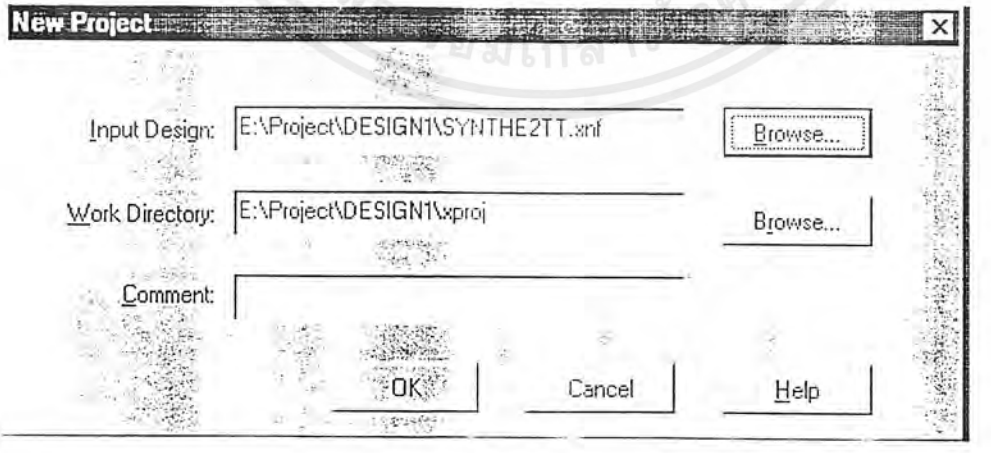
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1. เรียก Design Manager ของ XILINX Foundation 2.Li โดยคลิ๊กไอคอน Design Manager



รูปที่ 8.11 หน้าต่างของ Design Manager

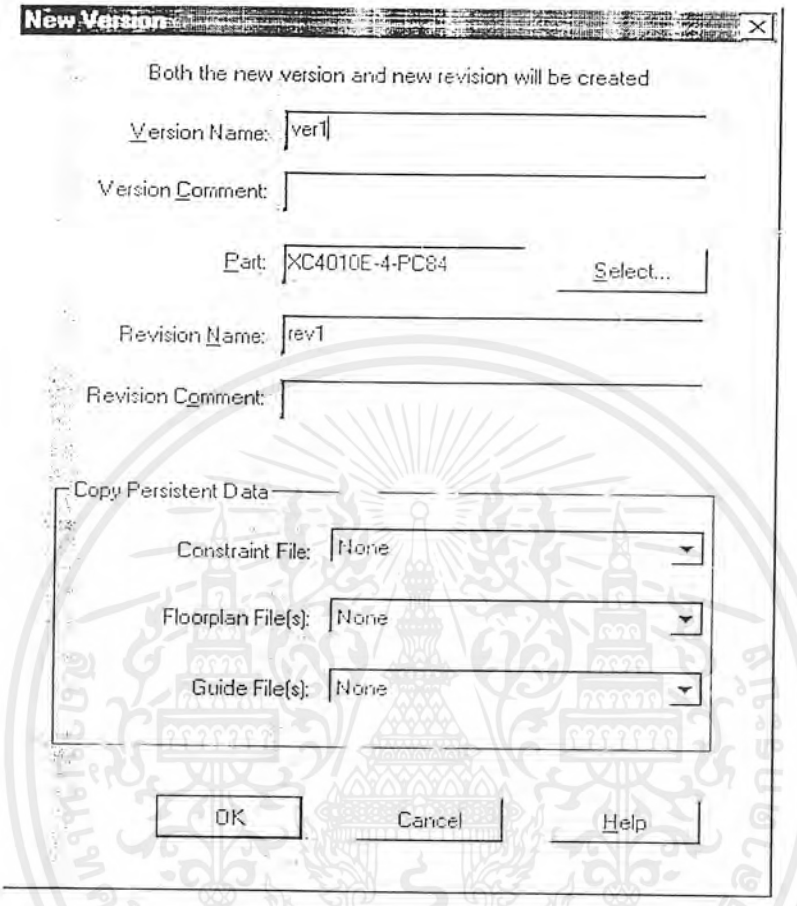
2. สร้างโปรเจคใหม่สำหรับดีไซน์ **File > New Project..**



รูปที่ 8.12 หน้าต่างของ New Project เมื่อสร้างโปรเจคใหม่

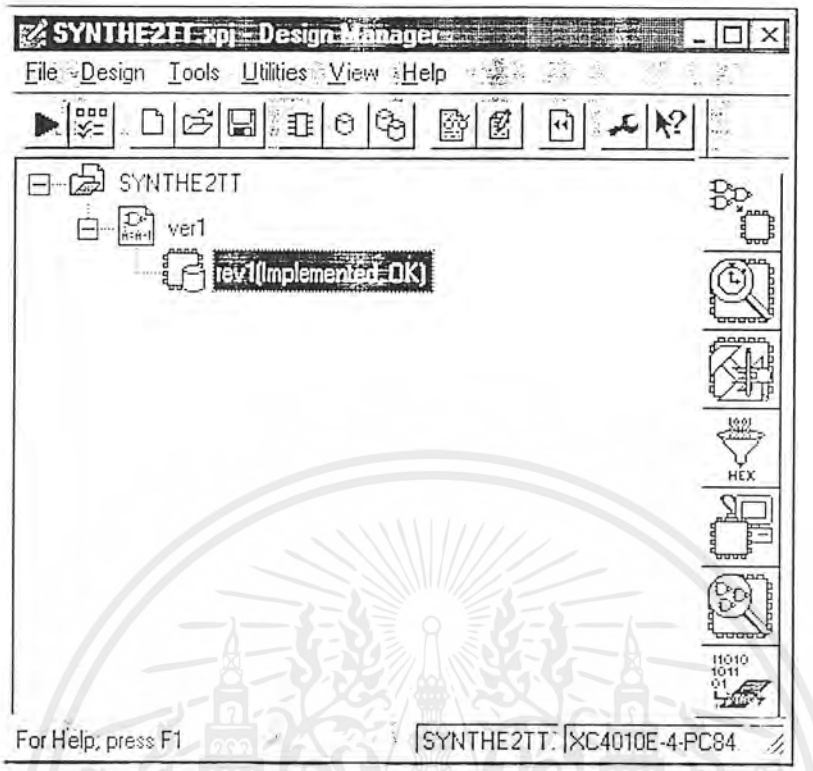
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตัวอย่างนี้ไฟล์ XNF อยู่ในไดทรอรี่ C:\Project\Mydesign.xnf



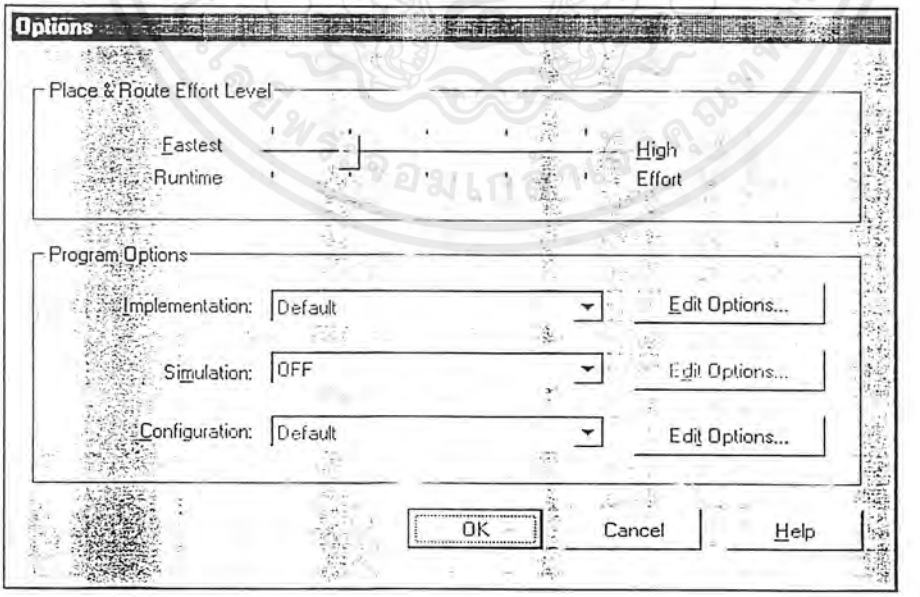
รูปที่ 8.13 หน้าต่างของการกำหนด Version และเบอร์ของ FPGA ที่ใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



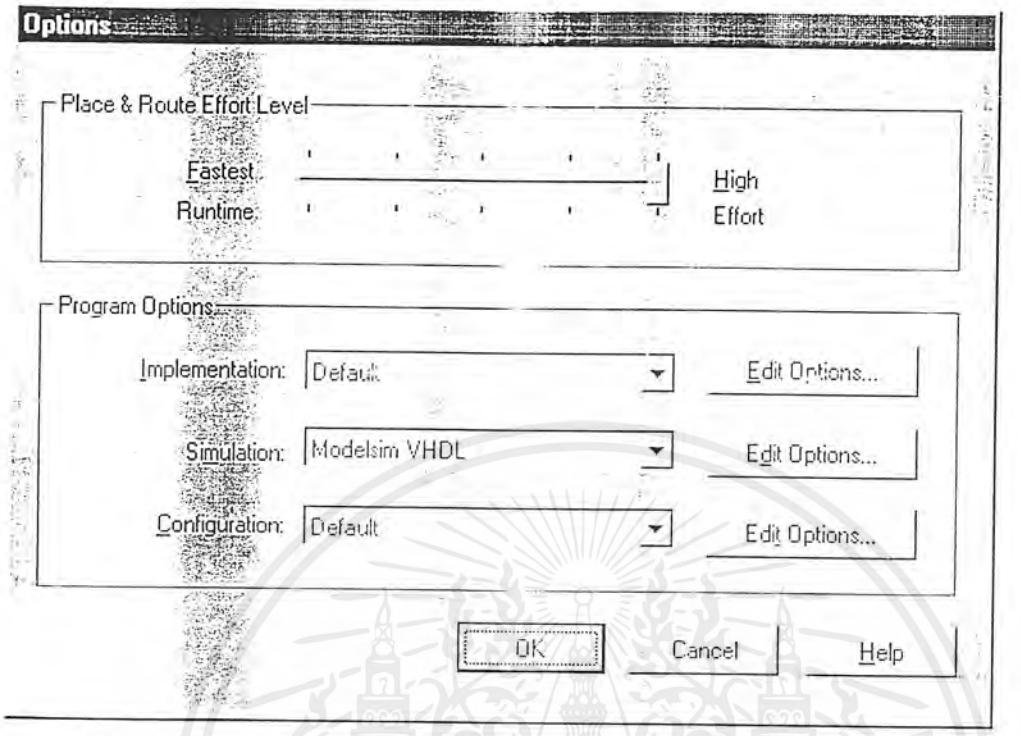
รูปที่ 8.14 หน้าต่าง Project ที่ทำการ New Version Complete

3. คลิ๊กเลือก Set option



รูปที่ 8.15 หน้าต่าง Set option

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 8.16 หน้าต่าง Set option ที่ถูกกำหนดค่าต่างๆ

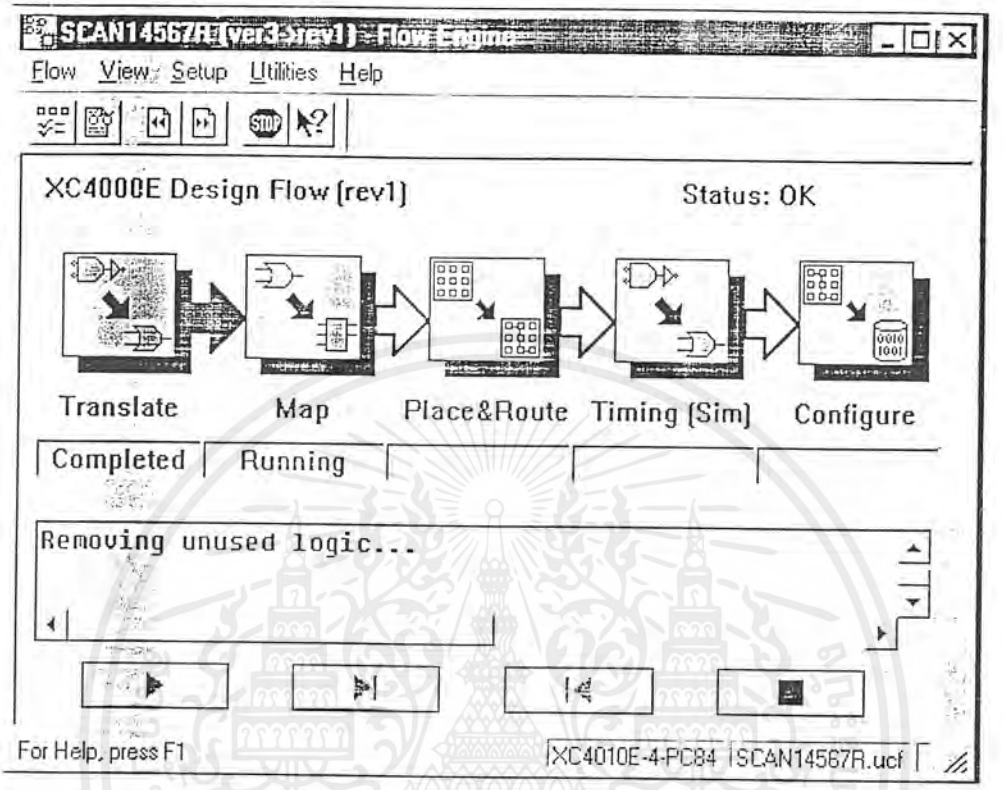
โดยทำการเลือก **Place & Route Effort Level > High Effort**

Program Options > Simulation > Modelsim VHDL

เพื่อนำไปจำลองการทำงานในการตรวจสอบมาตรฐานเวลาจากการทำ Place & Route ต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4. คลิ๊กเลือก Flow Engine



รูปที่ 8.17 หน้าต่าง Flow Engine

Flow Engine จะประกอบด้วยขั้นตอนย่อยดังต่อไปนี้ **Translate -> Map -> Place & Route -> Timing (Sim) -> Configure** และจะดำเนินการไปตามลำดับ โดยคลิ๊กเริ่มทำงานที่ปุ่มรัน โปรแกรมก็จะเริ่มทำงานตามขั้นตอนข้างต้น

Translate & Map

Translate คือ ขั้นตอนที่โปรแกรมอ่านเน็ตลิสต์ไฟล์แบบ XNF ของดีไซน์ตามที่ได้กำหนดไว้แล้วแปลงให้อยู่ในรูปแบบข้อมูลของ XILINX เอง (ไฟล์.NGO) โดยใช้คำสั่ง NGDBuild ซึ่งจะแปลงให้เป็นเน็สลิสต์แบบ .NGO (Native Generic Database) แล้วผ่านต่อไปยังขั้นตอน Map ที่เกี่ยวข้องกับการจัดการฮาร์ดแวร์ภายในของอุปกรณ์เป้าหมาย (Resource Allocation) รวมถึงการจัดแบ่งลอจิกในลอจิกบล็อค อาทิเช่น CLB (Configuration Logic Block) และ IOB (Input Output Block) และผลการทำงานที่ได้จะถูกเก็บบันทึกไว้ในไฟล์ที่ชื่อว่า **map.ncd** เพื่อนำไปใช้ในขั้นตอน Place & Route ต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หลังจากผ่านขั้นตอน Translate และ Map แล้วอาจจะมีการทำ Timing Analysis เพื่อทราบรายละเอียดเกี่ยวกับความล่าช้า (Delay time) ของสัญญาณอย่างคร่าวๆ ซึ่งเป็นค่าประมาณเท่านั้น ความล่าช้าของสัญญาณอาจแบ่งออกได้เป็นสองจำพวกคือ ความล่าช้าที่เกิดจากการแบ่งลอจิกบล็อค (Block Delay) และความล่าช้าที่เกิดจากการเชื่อมโยงเส้นทางสัญญาณระหว่างบล็อคเหล่านั้น (Routing Delay) ในกรณีที่ได้มีการกำหนดข้อบังคับในเรื่องระยะเวลาให้แก่เส้นทางสัญญาณภายในวงจร ค่าความล่าช้าของสัญญาณที่คำนวณได้หลังจากได้ทำขั้นตอน Map (Post-map Delay) ไม่ควรเกิน 50 เปอร์เซ็นต์ของความล่าช้ารวมอาจจะเกินค่าที่ต้องการก็ได้ และถ้าความล่าช้าจริงมากกว่าค่าที่กำหนดไว้ก็มีทางเลือกสองทาง คือ เปลี่ยนแปลงวงจรใหม่เพื่อให้ได้เส้นทางสัญญาณวิกฤต (Critical path) ที่สั้นลง หรืออีกทางหนึ่งก็คือ ยอมรับค่าความล่าช้าของสัญญาณที่ได้จริง

Place & Route

ขั้นตอนการทำ Place & Route เริ่มจากการอ่านข้อมูลของวงจรไฟล์ map.ncd และไฟล์ที่ลงท้ายด้วย .PCF (Physical Constrant File) ซึ่งภายในมีข้อบังคับต่างๆ ที่โปรแกรม PAR จะต้องคำนึงถึงเมื่อทำขั้นตอน Place & Route งานขั้นแรก คือ การวางลอจิกบล็อคลงใน FPGA ก่อนซึ่งเป็นหน้าที่ของ Placer และถ้ามีการกำหนดข้อบังคับในเรื่องเวลาแล้วโปรแกรม Placer ก็จะพยายามวางบล็อคเหล่านั้นให้อยู่ใกล้ๆ กันเพื่อลดค่าความล่าช้าของสัญญาณระหว่างบล็อคต่างๆ และผลการกระทำในแต่ละครั้งจะมีแต้มคะแนน (Placer Score) และครั้งที่มีแต้มคะแนนน้อยที่สุดจะถือว่าดีที่สุด และจบท้ายด้วยการเชื่อมโยงบล็อคต่างๆ ที่ถูกวางลงในตำแหน่งที่เหมาะสมแล้วด้วย Router ซึ่งขั้นตอนนี้จะมีหลายครั้งเพื่อพยายามเชื่อมโยงเส้นทางภายในของวงจรให้ได้ตามข้อกำหนด

Timing Analysis

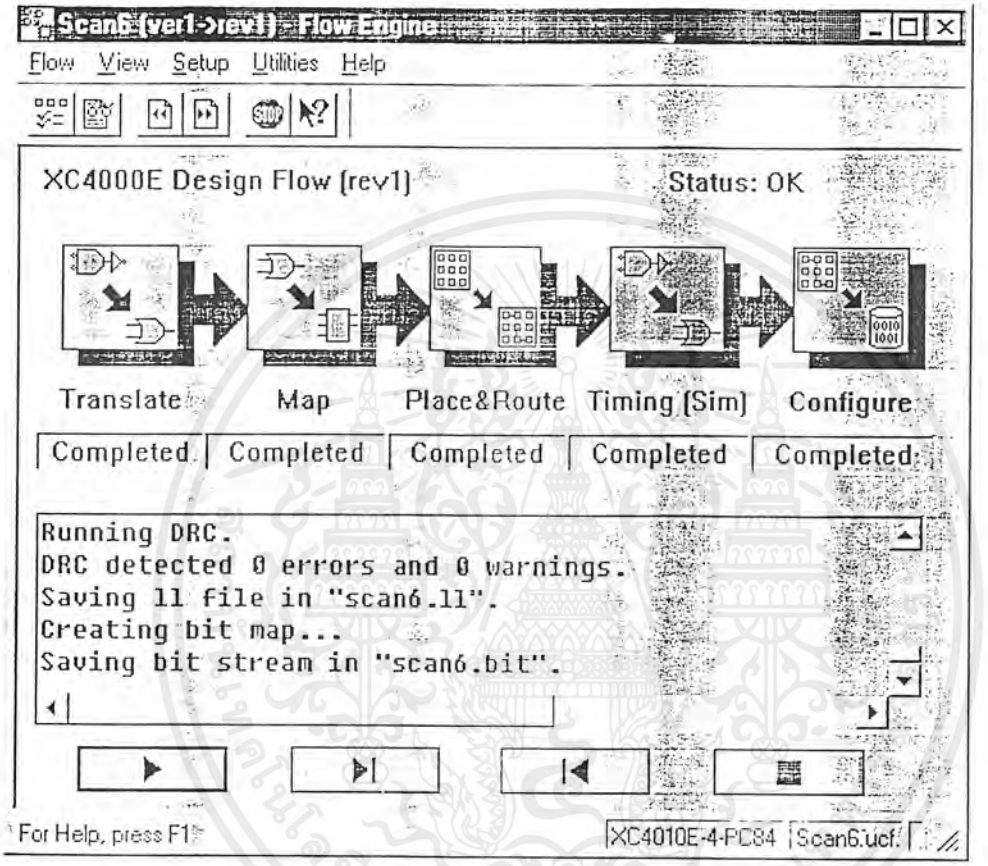
ในขั้นตอนนี้ โปรแกรมจะทำการวิเคราะห์วงจรเพื่อหาค่าความล่าช้าของเส้นทางสัญญาณต่างๆ หลังจากผ่านขั้นตอน Place & Route แล้วซึ่งค่าความล่าช้าที่คำนวณได้นี้ ค่อนข้างแม่นยำและถูกต้อง ใกล้เคียงกับค่าความล่าช้าจริงเมื่อวงจรทำงานภายใน FPGA

Configure

เมื่อผ่านขั้นตอน Place & Route แล้วถือว่าการออกแบบสร้างดีไซน์สำหรับ FPGA ได้เสร็จสิ้นลงแล้ว และเหลือเพียงขั้นตอน Configure ซึ่งใช้ในการแปลงข้อมูลของวงจรที่ได้ให้อยู่ในรูปแบบของบิตสตรีม (Bit Stream) เพื่อนำไป Download ลงใน FPGA ต่อไปโดยข้อมูลนี้จะถูกบันทึกลงในไฟล์ที่มีนามสกุล .BIT

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

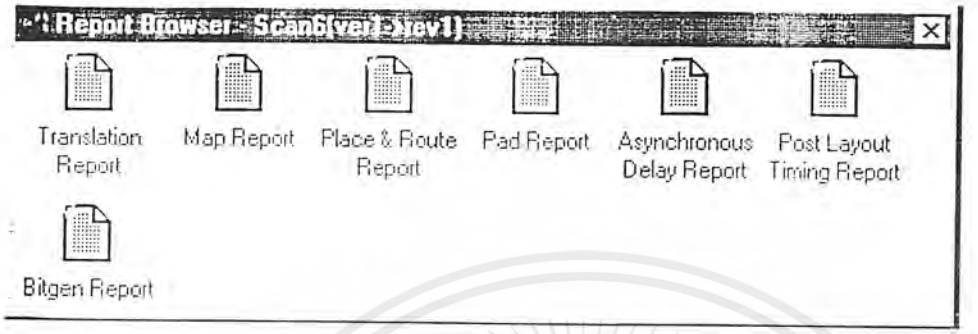
ในระหว่างการทำขั้นต้นแต่ละขั้นเริ่มตั้งแต่ Translate ไปจนถึง Configure ผู้ใช้สามารถหยุดการทำงานของ Flow Engine อย่างชั่วคราวได้ และสามารถสั่งให้ทำต่อไปได้ถ้าต้องการ โดยกดปุ่ม **stop** หรือ **run** ตามลำดับ



รูปที่ 8.18 หน้าต่าง Flow Engine ที่ทำครบทุกขั้นตอน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5. ผลการทำงานของขั้นตอนต่างๆ ใน Flow Engine สามารถตรวจสอบได้จาก คลิ๊กปุ่ม Browe. Report โดยรายงานผลจะถูกแบ่งออกตามขั้นตอนดังรูปที่ 8.22

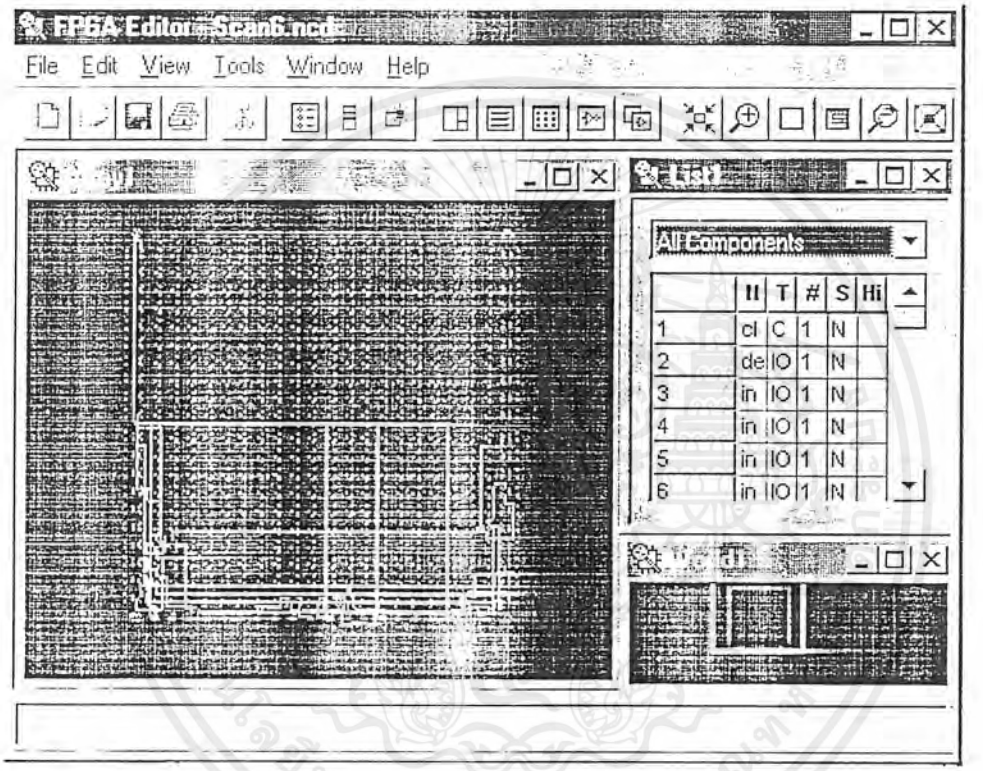


รูปที่ 8.19 หน้าต่างสำหรับเลือกแสดงผลการทำงานในขั้นตอนต่างๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

8.6 การใช้ FPGA Editor เพื่อตรวจสอบวงจรภายใน FPGA

หลังจากที่ได้ขั้นตอนต่างๆ ภายใน Flow Engine แล้วเราสามารถเรียกใช้งานโปรแกรมย่อยอีกตัวหนึ่งก็คือ FPGA Editor โดยคลิ๊กปุ่ม.....เพื่อใช้ดู Layout ของวงจรภายใน FPGA โดยจะแสดงให้เห็นตำแหน่งของลอจิกบล็อกต่างๆ พร้อมเส้นทางของสัญญาณที่เชื่อมโยงภายในวงจร มีลักษณะดังรูปที่ 8.20



รูปที่ 8.20 หน้าต่าง FPGA Editor และ Layout ของวงจรภายใน FPGA

การจำลองฐานเวลาของวงจร (Simulation) หลังจากทำขั้นตอน Place & Route

หลังจากที่ผ่านขั้นตอน Place & Route แล้ว จะต้องมีการตรวจสอบความถูกต้องของวงจรที่ได้อีกครั้ง เพื่อที่จะแน่ใจว่า วงจรที่เราจะนำไปใช้งานกับอุปกรณ์กับFPGA นั้นทำงานได้ถูกต้องทั้งในเรื่องของฟังก์ชั่นและเวลา เราเรียกขั้นตอนนี้ว่า Post-Layout Functional & Timing simulation เนื่องจากเราได้กำหนดตัวเลือกให้โปรแกรมสร้างเน็ตลิสต์ไฟล์แบบ VHDL สำหรับการจำลองฐานเวลาของวงจรไว้แล้ว ดังนั้นเราก็แค่นำสำเนาไฟล์ดังกล่าวไปใส่ไว้ในไดเรคทรอรี่ที่เราต้องการ ซึ่งมีอยู่สองไฟล์ด้วยกันคือ **time_sim.vhd** (VHDL) และ **time_sim.sdf** (SDF:Standard Format) และ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สามารถนำไปการจำลองฐานของวงจรโดยใช้ Modelsim ได้ แต่ก่อนที่จะทำการจำลองฐานเวลาของวงจรกับเน็ตลิสต์ที่สร้างจาก XILINX ได้ จะต้องมีการคอมไพล์สามไฟล์ตามลำดับคือ

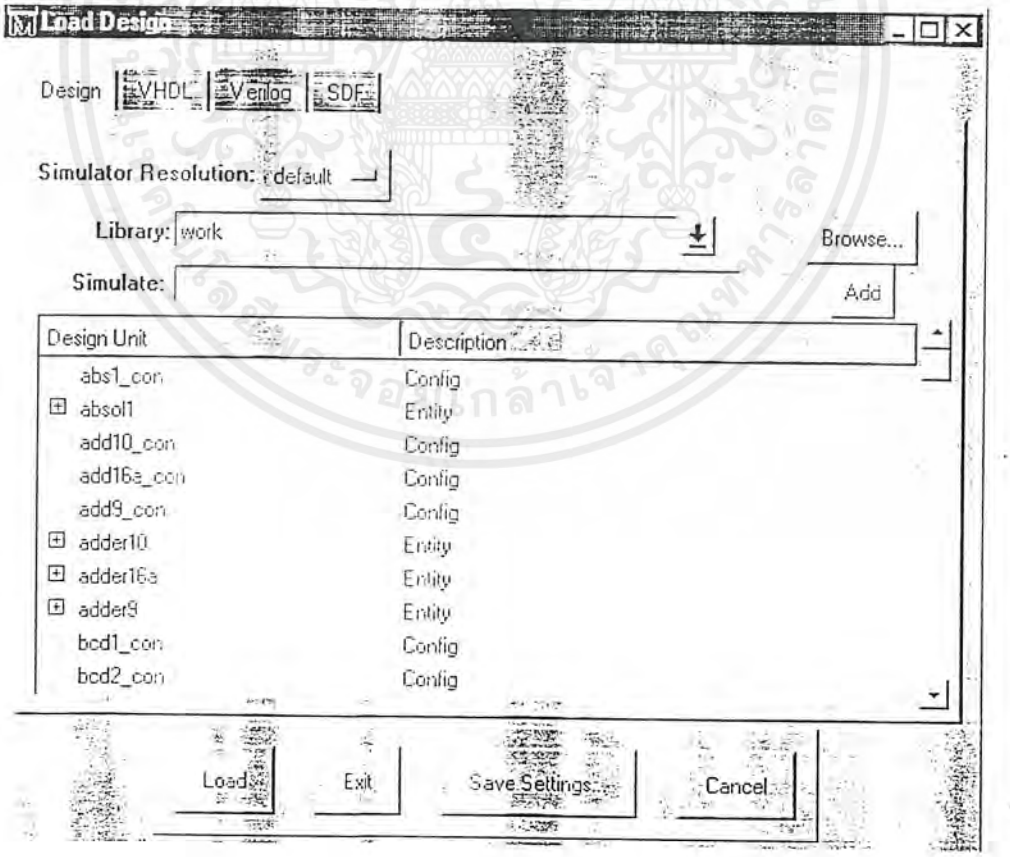
<XILINX>\vhdl\src\simprims**simprim_Vcomponent.vhd**

<XILINX>\vhdl\src\simprims**simprim_Vpackage.vhd**

<XILINX>\vhdl\src\simprims**simprim_VITAL.vhd**

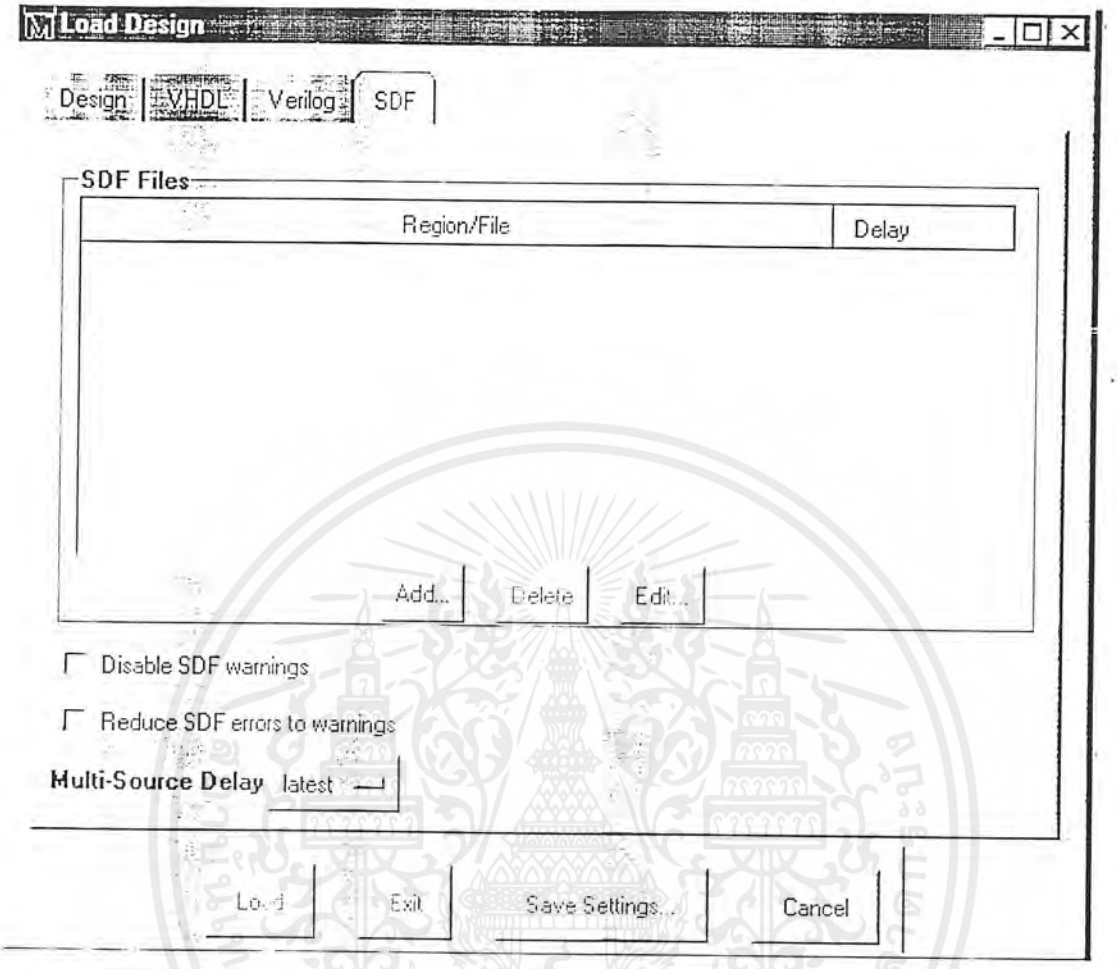
และจะต้องคอมไพล์เก็บไว้ในไลบรารี่ที่มีชื่อ **simprim**

หลังจากที่ได้สร้างไลบรารี่ simprim และคอมไพล์ไฟล์ทั้งสามเก็บไว้ในไลบรารี่ดังกล่าวภายใน Project Directory และจากนั้นทำการคอมไพล์ไฟล์ **time_sim.vhd** เก็บลงในไลบรารี่ work และทำการ recompile Tesbench ของดีไซน์ในครั้งที่ทำ Behavioral Simulation ให้ทำตามขั้นตอนเหมือนที่เคยทำเพียงแต่ในคราวนี้จะต้องมีการกำหนดไฟล์ SDF ที่ชื่อ **time_sim.sdf** เพิ่มขึ้นมา



รูปที่ 8.21 หน้าต่าง Load Design ภายในโปรแกรม Modelsim

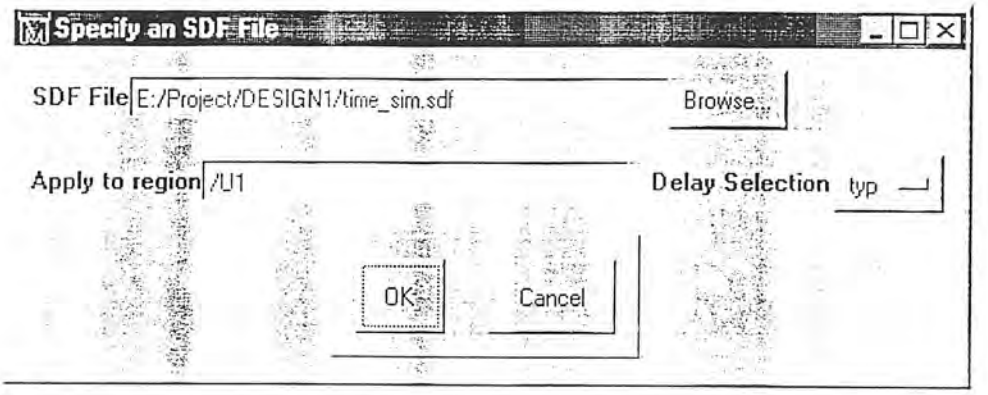
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



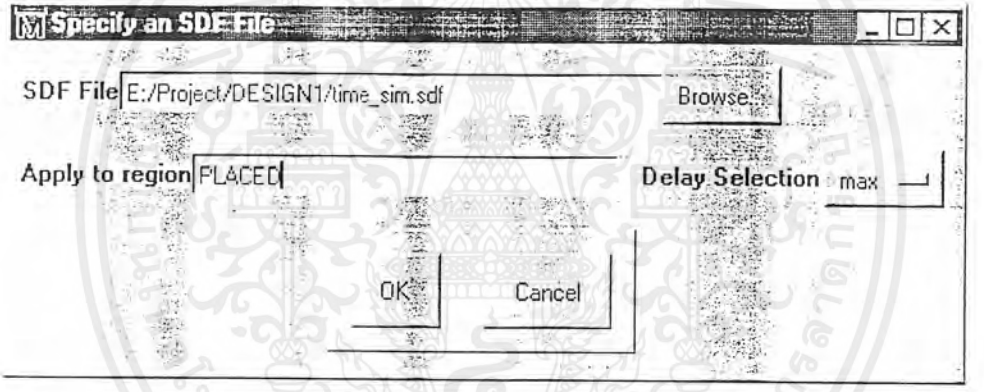
รูปที่ 8.21.1 หน้าต่างสำหรับเลือกไฟล์ SDF ในการซิมมูลเลชั่น

โดยเริ่มจากคลิ๊กปุ่ม Add เพื่อเลือกไฟล์ SDF เมื่อคลิ๊กปุ่ม Add จะปรากฏหน้าต่าง รูปที่ 21.1 ในตัวเลือกไฟล์ **C:Project\time_sim.sdf** และ **U1** เป็นชื่อ instsnce เมื่อเรียกใช้ Mydesign โมดูลที่ออกแบบใน Test bench (TB_ Mydesign.vhd)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

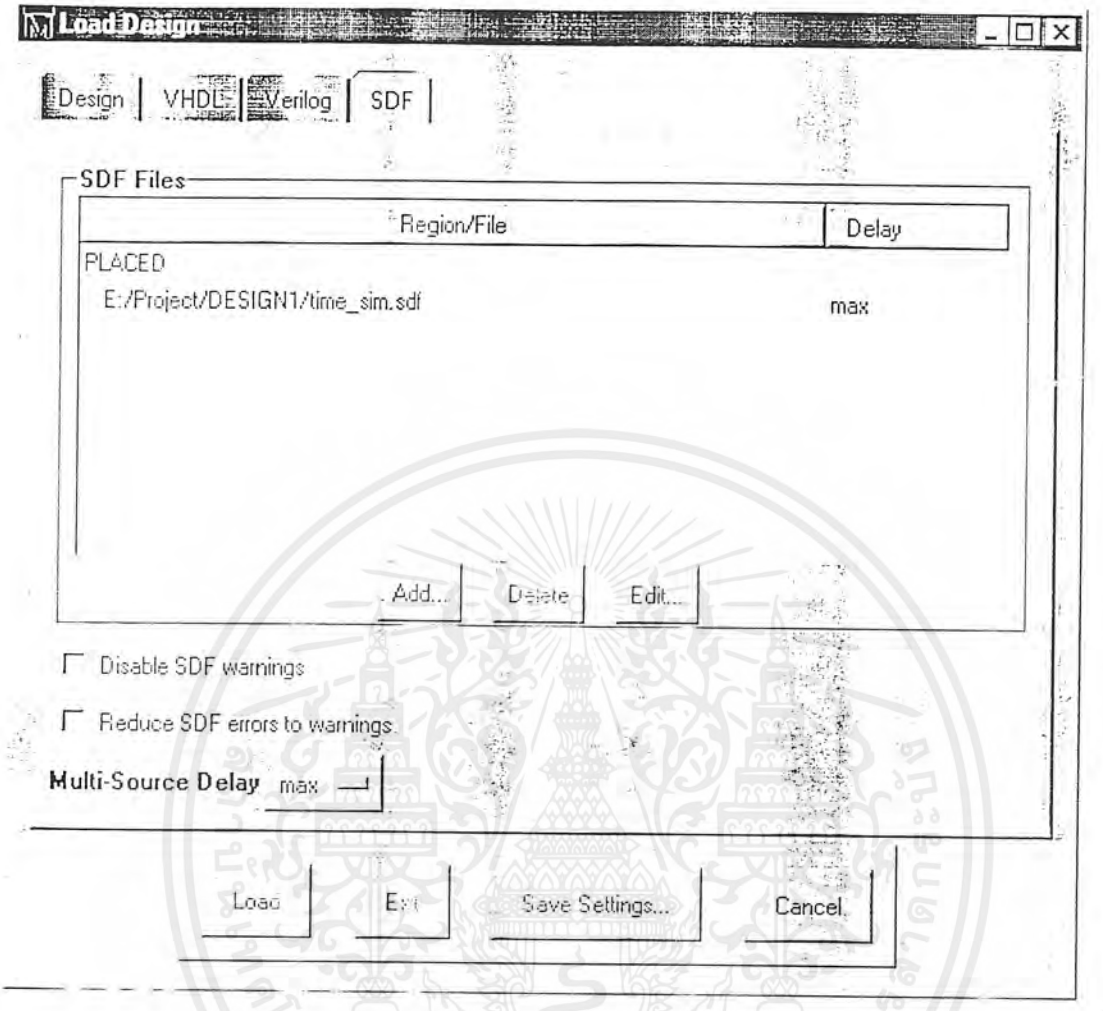


รูปที่ 8.21.2 หน้าต่างสำหรับเลือกไฟล์ SDF ในการซิมมูลเลชั่น
เลือก Delay Selection เป็น MAX ในการจำลองฐานเวลาของวงจร



รูปที่ 8.21.3 หน้าต่างสำหรับเลือกไฟล์ SDF ในการซิมมูลเลชั่น

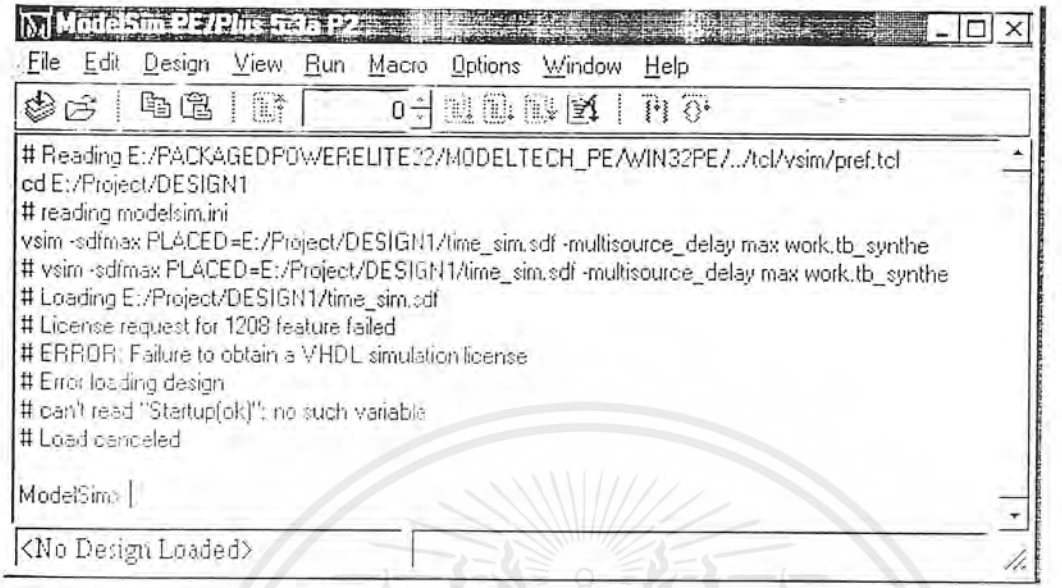
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 8.21.4 หน้าต่างสำหรับเลือกไฟล์ SDF ในการซิมมูลเลชั่น

เมื่อทำครบทุกขั้นตอนให้คลิ๊กปุ่ม Load เพื่อทำการ Load Test bench ขึ้นมาซิมมูลเลชั่นในระดับ Timing ต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



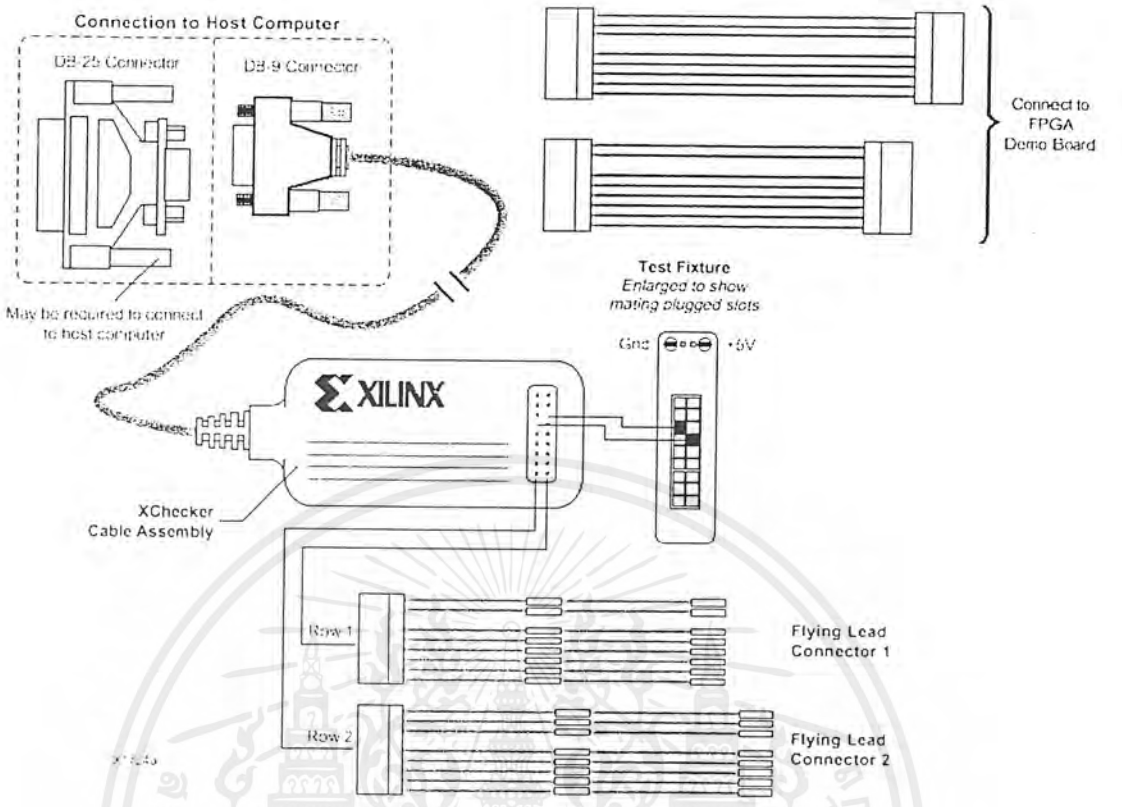
รูปที่ 8.22 หน้าต่างในการ Load Test bench พร้อมกับ SDF เรียบร้อย

จากนั้นทำการเปรียบเทียบดูสัญญาณ (waveform) ว่าตรงกับที่จำลองการทำงานในระดับฟังก์ชั่น (Functional) หรือไม่ เพื่อเป็นการยืนยันว่าการทำงานยังถูกต้องจากการที่นำไปสังเคราะห์

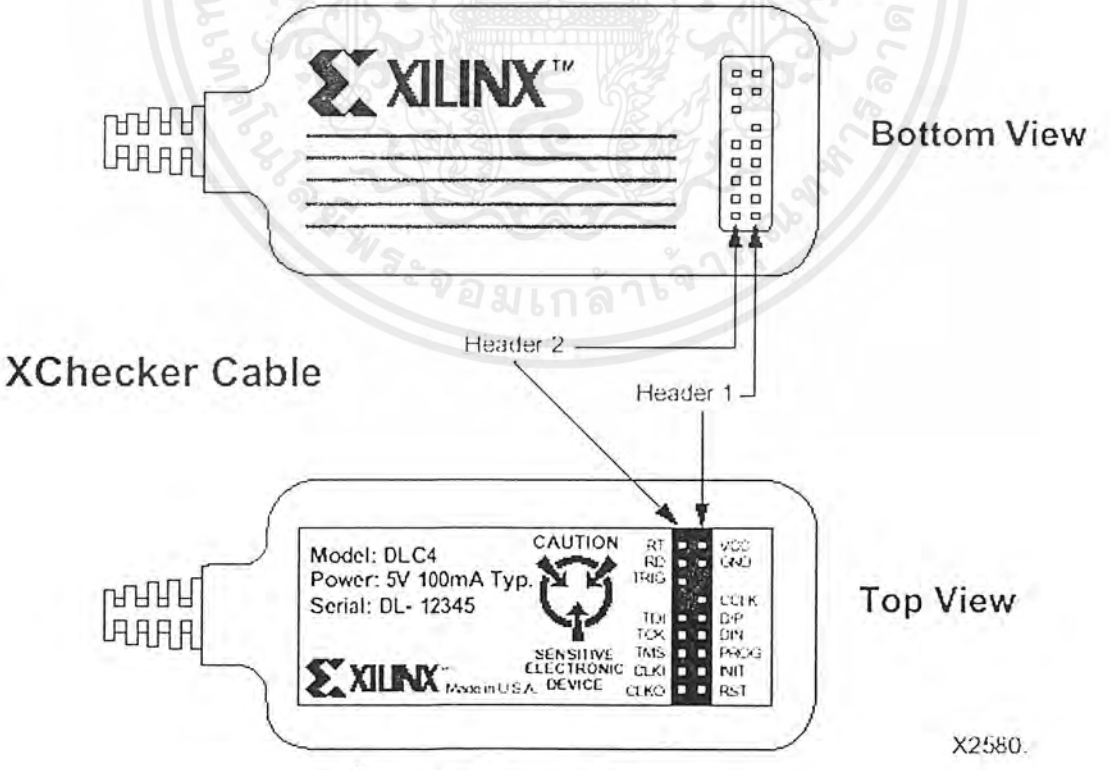
8.7 การทดสอบวงจรกับฮาร์ดแวร์บาง XILINX84 FPGA Demo board

เมื่อผ่านขั้นตอนต่างๆ จาก Flow Engine แล้วผู้ใช้สามารถนำวงจรที่ได้สร้างขึ้นสำหรับ FPGA ไปทดสอบกับฮาร์ดแวร์จริงได้โดยการ Download ไฟล์ โดยผ่านโปรแกรม Hardware Debugger โดยคลิ๊กปุ่มในโปรแกรม Design Manager เพื่อ Download ข้อมูลใน FPGA บนบอร์ดทดสอบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

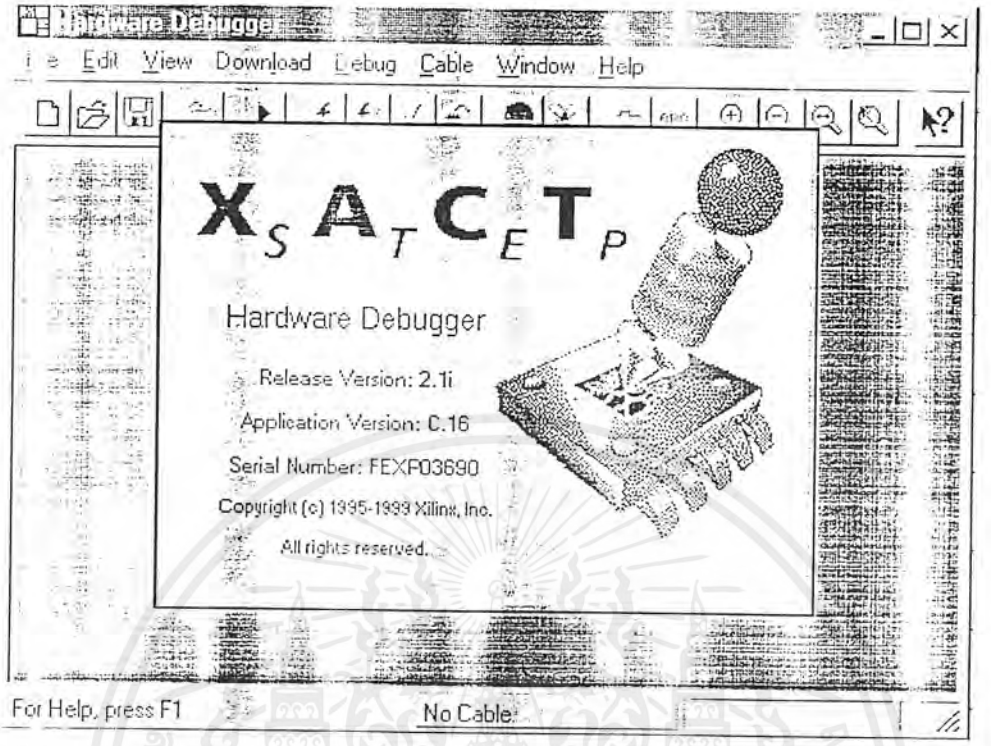


รูปที่ 8.23 สาย X-Checker พร้อมแถบสายต่อไปยังบอร์ดทดสอบ

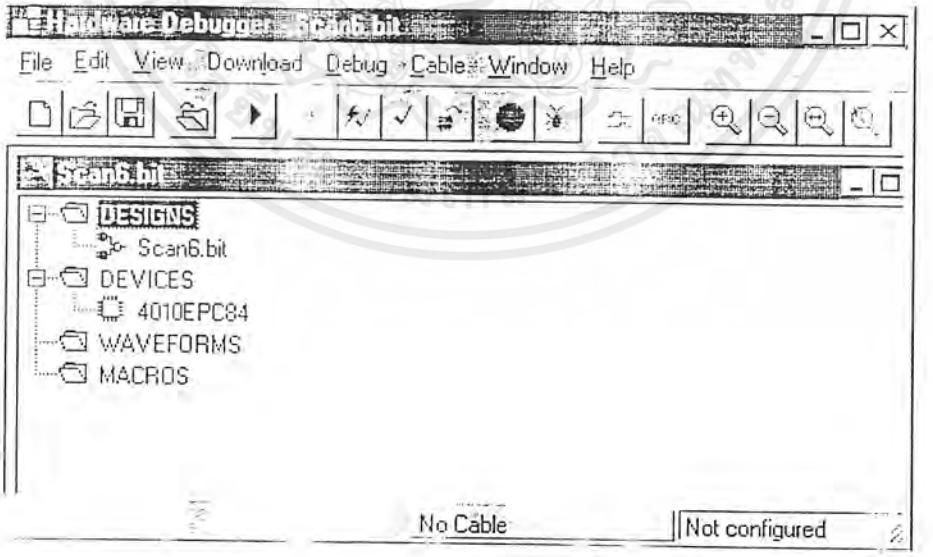


รูปที่ 8.24 ส่วนหัวของสาย X-Checker

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



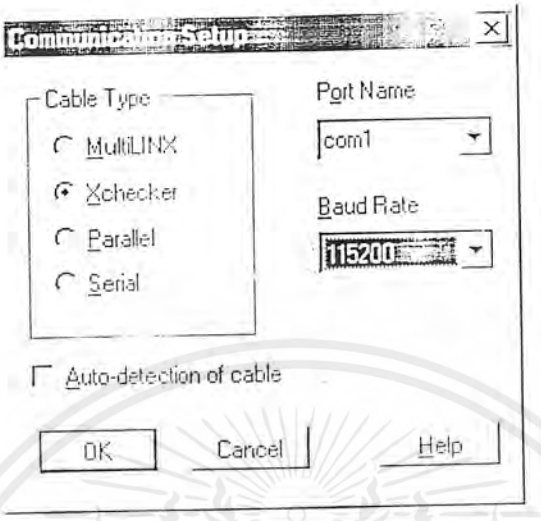
รูปที่ 8.25 หน้าต่างโปรแกรม Hardware Debugger



รูปที่ 8.26 หน้าต่างโปรแกรม Hardware Debugger ในการเปิด Design project

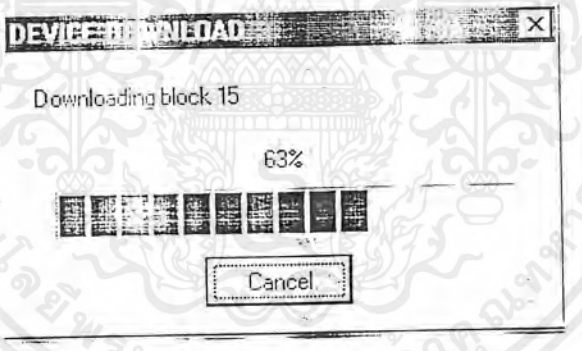
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เลือก Cable>Communications...เพื่อทำการ Setup การเชื่อมต่อของ X-Checker cable กับเครื่องคอมพิวเตอร์ที่ใช้

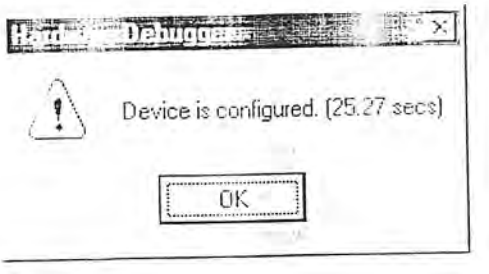


รูปที่ 8.27 หน้าต่าง Communication Setup

เมื่อเชื่อมทุกอย่างเรียบร้อย จากนั้นทดลอง Download วงจรสู่ FPGA โดยคลิ๊กปุ่ม ⊕



รูปที่ 8.28 หน้าต่างแสดงการ Download วงจรสู่ FPGA

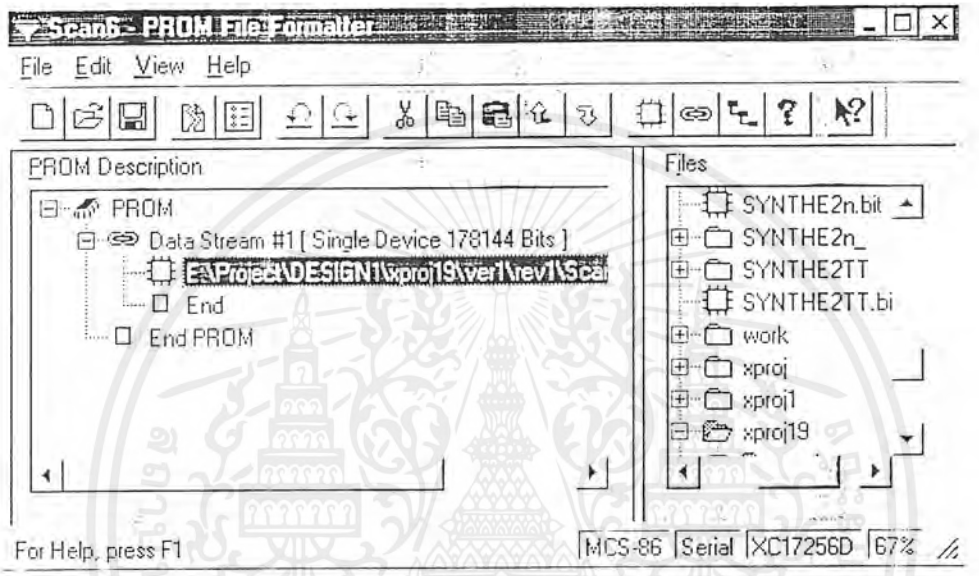


รูปที่ 8.29 หน้าต่างแสดงการ Download วงจรเสร็จสมบูรณ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

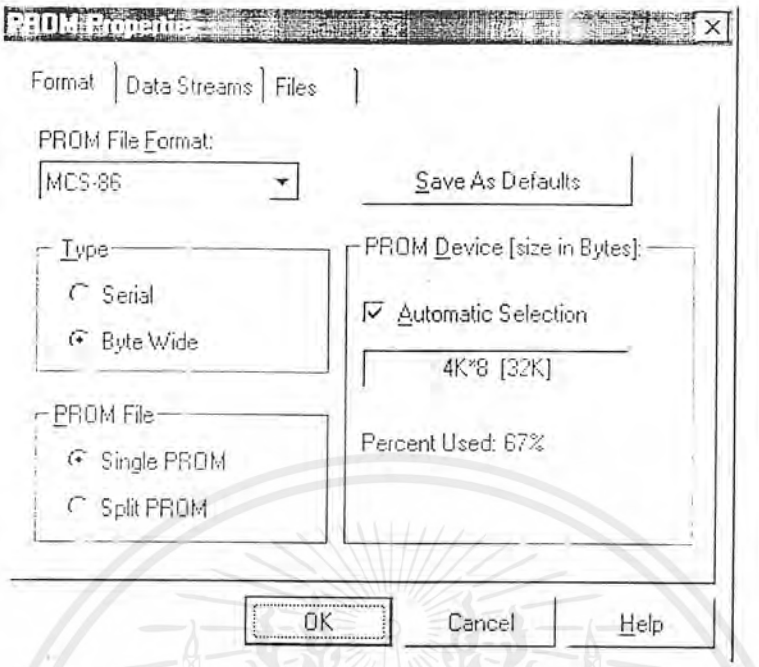
1.8 Configuration PROM

เป็นขั้นตอนการแปลง BIT ไฟล์ที่ใช้ในการ Download ผ่าน X-Checker เพื่อใช้เขียนลงสู่ Serial PROM ในการ Communication แบบ Master Series mode แทนการใช้ X-Checker cable ซึ่งเป็นแบบ Slave series mode ได้โดยการคลิ๊กปุ่ม PROM File Fomatter ใน Design Manager

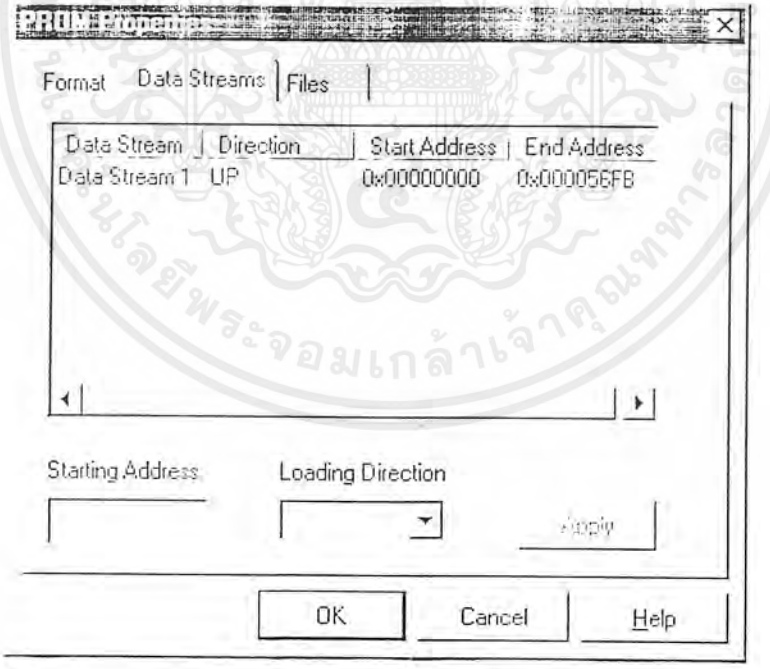


รูปที่ 8.30 หน้าต่างโปรแกรม PROM File Fomatter

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



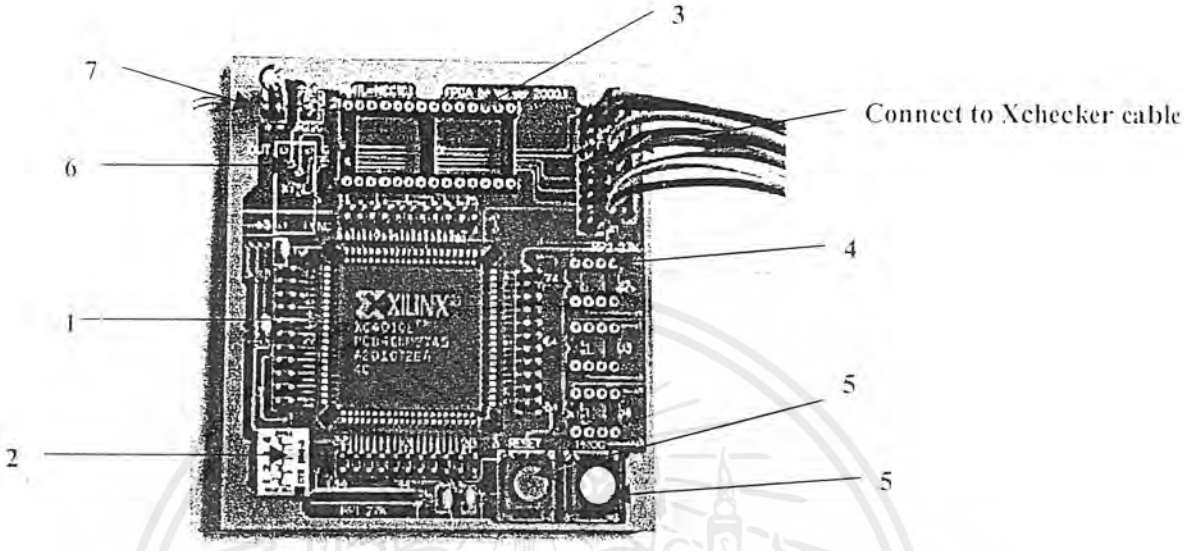
รูปที่ 8.31หน้าต่าง PROM Properties



รูปที่ 8.32 หน้าต่าง PROM Properties ในการเริ่มเขียนแบบ UP Direction

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

8.9 วิธีการใช้รายละเอียด XILINX84 Demo board

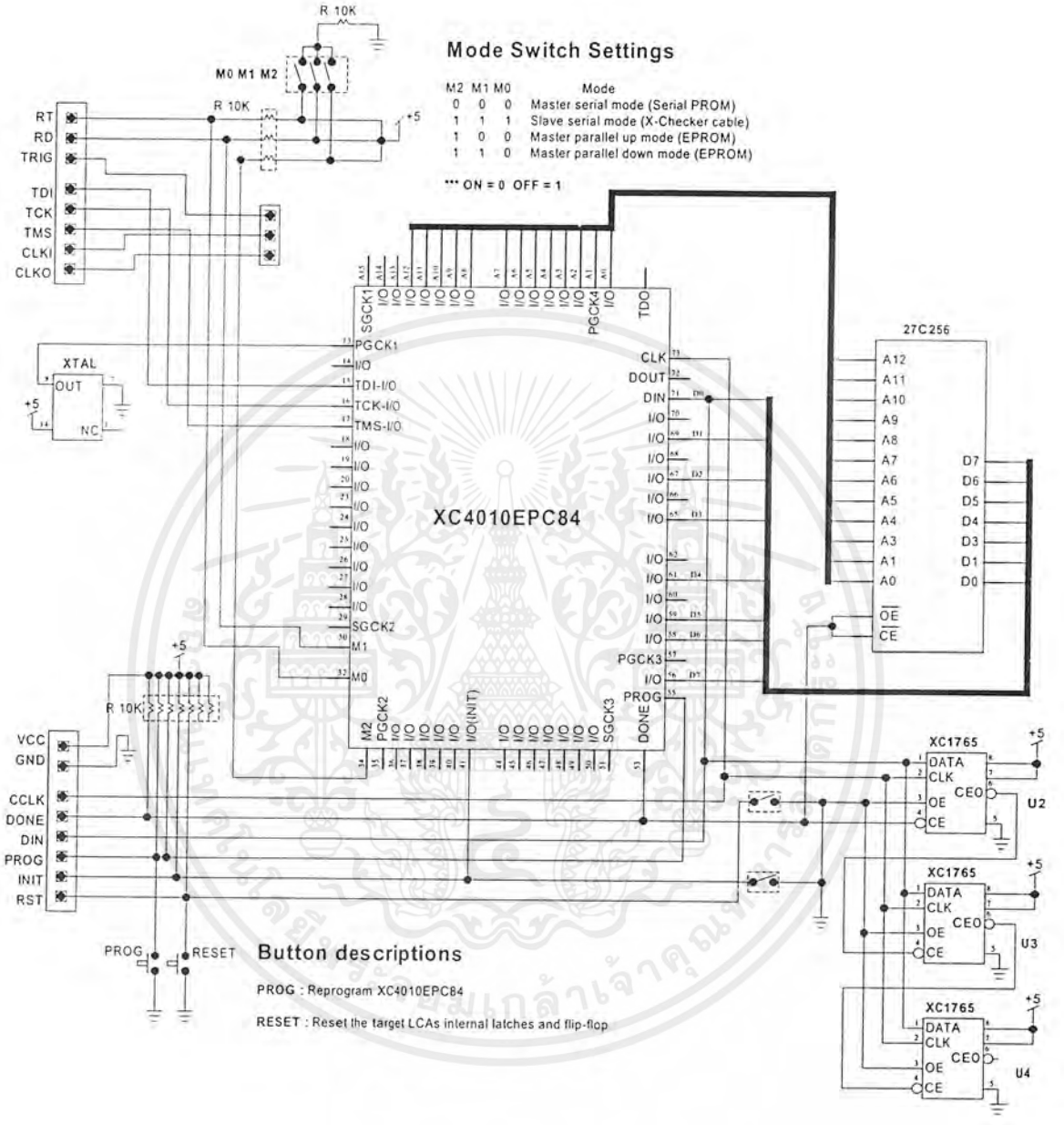


รูปที่ 8.33 XILINX84 Demo board

จากรูปที่ 8.33 มีรายละเอียดของบอร์ดและการเชื่อมต่ออุปกรณ์ต่างๆ โดยแต่ละส่วนประกอบหลักๆ ดังนี้

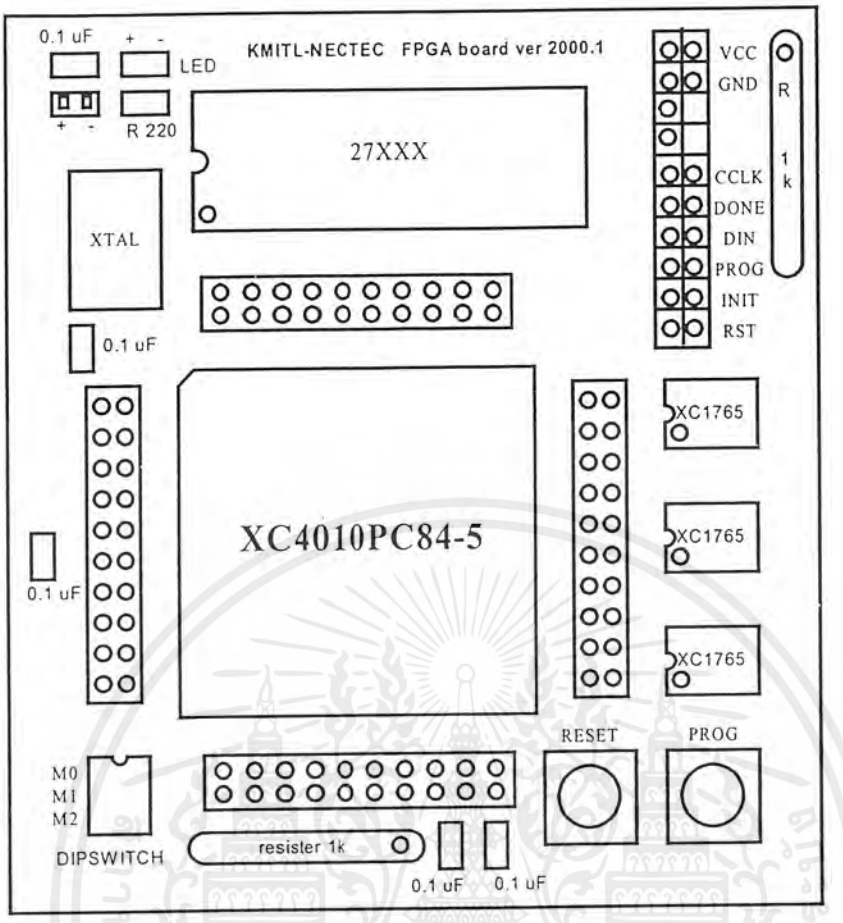
1. SOCKET 84 ขาไว้สำหรับใส่อุปกรณ์ FPGA 1 ชุด
2. DIPSWITCH 3 rows ไว้สำหรับกำหนดโหมดในการโปรแกรมอุปกรณ์ FPGA 1 ชุด
3. SOCKET 28 ขาไว้สำหรับใส่ EPROM 27C256 1 ชุด (สำหรับ Master parallel mode)
4. SOCKET 8 ขาไว้สำหรับใส่ Serial PROM เบอร์ XC1765 3 ชุด (สำหรับ Master serial mode)
5. SWITCH กดสำหรับ PROG และ RESET อย่างละ 1 ชุด
6. SOCKET XTAL สำหรับใส่ XTAL ภายนอก (ถ้าต้องการ)
7. Connector with power supply 5V

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

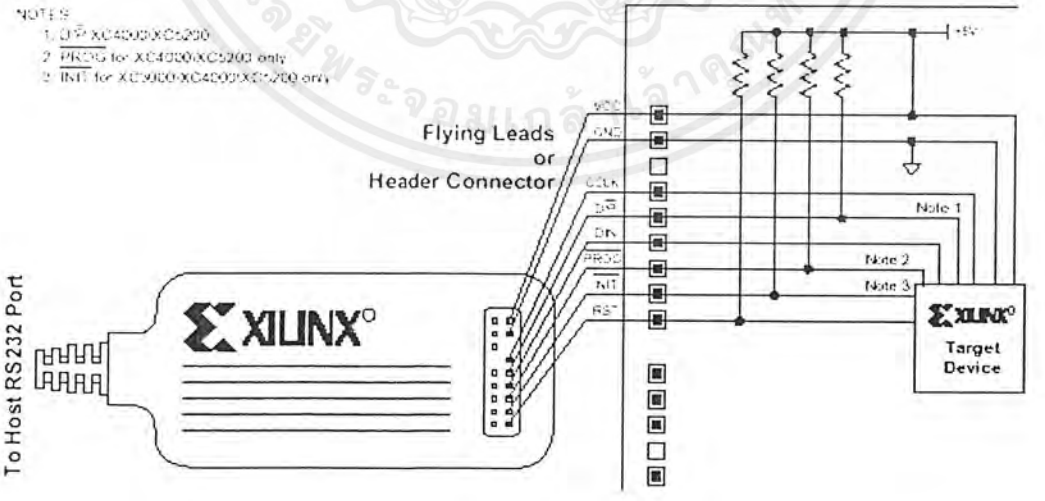


รูปที่ 8.34 โครงสร้างบอร์ด XILINX84 Demo board

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 8.35 โครงสร้าง XILINX84 Demo board



รูปที่ 8.36 การเชื่อมระหว่าง X-Checker กับ XILINX84 Demo board

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 9
สรุปและวิจารณ์การทำงาน

9.1 สรุปและวิจารณ์

จากการศึกษาและทำการทดลองออกแบบระบบดิจิตอลด้วยภาษาวีเอชดีแอลนั้นจะพบว่าขั้นตอนในการออกแบบมีดังนี้

1. แนวความคิดในการออกแบบระบบ
2. แสดงแนวความคิดออกมาในรูปแบบบล๊อคไดอะแกรมและโฟลว์ชาร์ต
3. การออกแบบด้วยโปรแกรมวีเอชดีแอล
   3.1 เขียนอธิบาย การทำงานของแต่ละบล๊อคไดอะแกรมด้วยภาษาวีเอชดีแอลโดยที่แต่ละบล็อคในไดอะแกรมอาจจะมีบล๊อคไดอะแกรมย่อยๆอยู่ภายใน
   3.2 นำโปรแกรมในแต่ละบล๊อคไดอะแกรมมาต่อรวมกันกลายเป็นโปรแกรมของระบบรวม
4. การทดสอบโปรแกรมด้วยอินพุทที่เหมาะสมในระดับ Source Code
   4.1 ทดสอบโปรแกรมในแต่ละบล๊อคไดอะแกรมย่อยๆซึ่งเขียนไว้เป็นentity
   4.2 เมื่อโปรแกรมผ่านการทดสอบในแต่ละบล๊อคไดอะแกรม ทำการรวมโปรแกรมเข้าด้วยกันเป็นโปรแกรมรวมของระบบ และทำการทดสอบอีกครั้ง
5. การสังเคราะห์วงจรออกมาในรูปแบบที่เหมาะสมกับ FPGAs
   5.1 การสังเคราะห์เพื่อนำผลลัพธ์ที่ได้ไปทดสอบในระดับ gate sim
   5.2 การสังเคราะห์เพื่อนำผลลัพธ์ที่ได้ไปโปรแกรมตัว FPGAs
6. การจัดวางตำแหน่งและเลือกเส้นทางเดินข้อมูล(Place and Routing) ของวงจรที่ได้จากการสังเคราะห์ เพื่อสร้างข้อมูลที่เหมาะสมสำหรับการโปรแกรมตัว FPGAs
7. การโปรแกรมตัว FGPAs ด้วยแฟ้มข้อมูลที่ได้จากการจัดวางตำแหน่งและเลือกเส้นทางเดินข้อมูล
8. การออกแบบส่วนประกอบภายนอกและสร้างเครื่องต้นแบบ
9. การจำลองและการทดสอบในขั้นสุดท้าย

ในขั้นตอนการออกแบบด้วยโปรแกรมวีเอชดีแอล นั้นสามารถทำได้หลายวิธีซึ่งแสดงให้เห็นถึงความยืดหยุ่นของภาษา โดยรูปแบบต่างๆดังกล่าวสามารถแบ่งได้ดังนี้

1. รูปแบบ Top-Down Design รูปแบบนี้มีลักษณะคล้ายกับการออกแบบซอฟต์แวร์ โดยการแบ่งระบบออกเป็นโมเดล ซึ่งจะออกแบบระบบทั้งหมดก่อน โดยการมอง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โมเดลหนึ่งๆนั้นเป็นBlack Block ที่ไม่ทราบการทำงานภายใน แต่ทราบว่าโมเดลนั้นมีอินพุท เอาท์พุทอย่างไร จากนั้นจึงทำการออกแบบส่วนของโมเดลต่างๆ

2. รูปแบบ Bottom-Up Design รูปแบบนี้จะกลับกันกับแบบ Top-Down Design เพราะจะออกแบบทีละโมเดลก่อน จากนั้นนำโมเดลต่างๆมาประกอบเข้าด้วยกันเป็นระบบ
3. รูปแบบ Flattem design รูปแบบนี้จะไม่มีการแบ่งแยกการทำงานออกเป็นโมเดล แต่จะมองระบบทั้งหมด แล้วทำการออกแบบ ระบบนี้จะเหมาะสมกับระบบที่ทำงานไม่ซับซ้อน

9.2 วิเคราะห์ผลงานและปัญหาในการทำงาน

1. เนื่องจากโปรแกรมวีเอชดีแอลนั้น เป็นโปรแกรมที่ไม่ได้ใช้กันแพร่หลายโดยทั่วไปจึงหาโปรแกรมมาใช้งานยากมาก อีกทั้งโปรแกรมบางชนิดก็ต้องใช้ Hard Lock ด้วย ในการออกครั้งนี้ได้ใช้โปรแกรม V-System ซึ่งเป็น Version เก่า มาจากภาควิชาคอมพิวเตอร์ แต่หนังสือคู่มือที่ใช้เป็นคู่มือการเรียนการสอน การเขียนโปรแกรมภาษาวีเอชดีแอลของบริษัท Esperan ซึ่ง NECTEC ได้ใช้ในการอบรมในหลักสูตรการออกแบบเบื้องต้น
2. โปรแกรมภาษาวีเอชดีแอลนี้ค่อนข้างมีรายละเอียดในการใช้งานมากทำให้ยุ่งยากในการศึกษาในเบื้องต้น แต่หลังจากเริ่มเข้าใจระบบการออกแบบและโครงสร้างต่างๆของภาษา จะทำให้การออกแบบทำได้ง่านมากขึ้นรวมทั้งโปรแกรมเดิมมักจะมีโปรแกรมบางส่วนถูกสร้างไว้แล้วจะสะดวกในการนำมาใช้งาน แต่อย่าลืมว่าจะต้องทดสอบโปรแกรมส่วนนั้นให้ละเอียดเพื่อหลีกเลี่ยงข้อผิดพลาดที่จะเกิดขึ้น
3. เราสามารถสร้างโปรแกรมย่อยเก็บไว้ใช้งานได้ ซึ่งจะสะดวกมากในการศึกษาในลักษณะที่ทำกันเป็นกลุ่ม โดยแบ่งกันออกแบบแต่ละส่วนย่อยแล้วนำมาสร้างเป็นระบบที่มีขนาดรวมใหญ่
4. โปรแกรมภาษาวีเอชดีแอลที่ผ่านการทดสอบแบบ Source Code แล้วบางครั้ง ไม่สามารถสังเคราะห์เป็นวงจรได้ ซึ่งทำให้ยุ่งยากมากสำหรับการเขียนจะต้องเรียบเรียงโครงสร้างของโปรแกรม เพื่อให้เหมาะสมสำหรับการสังเคราะห์
5. วงจรที่ได้จากการสังเคราะห์ เมื่อนำมาทดสอบวงจรแบบ gate sim อาจจะทำงานไม่ถูกต้อง จะต้องกลับไปแก้ไข โปรแกรมภาษาวีเอชดีแอลใหม่ แล้วทำตามลำดับขั้นต่างๆใหม่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6. วงจรที่ผ่านการทดสอบแบบ gate sim และผ่านกระบวนการ Place and Routing แล้ว เมื่อนำมาทดสอบโปรแกรมแบบ Time Sim ก็อาจจะมีข้อผิดพลาดขึ้นได้เนื่องจากจะมีส่วนการหน่วงเวลาเกิดขึ้นมีอุปกรณ์ต่างๆ
7. การทดสอบแบบ Time Sim ควรที่จะแยกวงจรออกเป็นส่วนย่อยๆแล้วทำการทดสอบจากนั้นจึงจะรวมวงจรเข้าด้วยกันและทดสอบวงจรรวมอีกครั้งหนึ่ง
8. วงจรที่ผ่านการทดสอบแบบ Time Sim ในแต่ละส่วน เมื่อนำมารวมกันแล้วทำการทดสอบ อาจจะมีข้อผิดพลาดเกิดขึ้นได้เนื่องจาก เวลาที่ถูกหน่วงในแต่ละส่วนเมื่อนำมาต่อเข้าด้วยกัน ก็จะถูกรวมกันด้วย
9. วงจรที่ผ่านการทดสอบแบบ Time Sim เมื่อนำไปโปรแกรม FPGAs อาจจะมีข้อผิดพลาดได้ สาเหตุที่อาจจะเกิดขึ้นได้เช่น ลักษณะของ fan out ของสัญญาณอินพุทที่มาจากภายนอกและสัญญาณนาฬิกา อาจจะมีมากเกิน จะสามารถแก้ไขได้ด้วยการใส่ Buffer โดยการเขียนโปรแกรมเพิ่มเติมเข้าไปในระบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก

โปรแกรมภาษาวีเอชดีแอลของระบบ

ผ.1 โปรแกรมส่วนสแกนคีย์บอร์ด

```
   -- SCAN4 FUNCTION
library IEEE ;
use IEEE.Std_Logic_1164.all ;
Library EXEMPLAR;
use EXEMPLAR.EXEMPLAR.all;
entity SCAN4 is
port  ( DEC  :  in std_logic ;
      IND  : in integer range  20 to 40 ;
      OUTD : out  integer range 15 to 40 );
 end SCAN4 ;
architecture SCAN4  of  SCAN4  is
begin
 process (DEC,IND)
 begin
          if (DEC'event and DEC = '0') then
           OUTD <= IND - 1 ;
          end if;
 end process;
end SCAN4;
configuration  SCAN4_CON   of  SCAN4 is
   for  SCAN4
   end for ;
end SCAN4_CON ;


   -- SCAN5 FUNCTION
library IEEE ;
use IEEE.Std_Logic_1164.all ;
Library EXEMPLAR;
use EXEMPLAR.EXEMPLAR.all;
entity SCAN5 is
port  ( INC  :  in std_logic ;
      IND  : in integer range  20 to 40 ;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
      OUTD : out  integer range 20 to 45 );
 end SCAN5 ;
architecture SCAN5  of  SCAN5  is
begin
 process (INC,IND)
 begin
          if (INC'event and INC = '0') then
           OUTD <= IND + 1 ;
          end if;
 end process;
end SCAN5;
configuration  SCAN5_CON   of  SCAN5 is
   for  SCAN5
   end for ;
end SCAN5_CON ;

 --  SCAN6 FUNCTION
library IEEE ;
use IEEE.Std_Logic_1164.all ;
Library EXEMPLAR;
use EXEMPLAR.EXEMPLAR.all;
entity SCAN6 is
port  ( CLK2  :  in std_logic ;
      POWER , RESET,INC,DEC  : in std_logic;
      IN1  : in integer range  20 to 45 ;          --INC
      IN2  : in integer range  15 to 40 ;          --DEC
      IN3  : in integer range  20 to 40 ;          -- Z1
      OUTD : out  integer range  15 to 45 );
 end SCAN6 ;
architecture SCAN6  of  SCAN6  is
begin
 process (CLK2,POWER,RESET,INC,DEC,IN1,IN2,IN3)
 begin
          if (CLK2'event and CLK2 = '1') then
           if (POWER = '0' or RESET = '0') then
             OUTD <= 25 ;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        elsif ( INC = '0' ) then
                OUTD <= IN1 ;
        elsif ( DEC = '0' ) then
         OUTD <= IN2 ;
        else
         OUTD <= IN3 ;
        end if;
       end if;
 end process;
end SCAN6;
configuration SCAN6_CON of SCAN6 is
   for SCAN6
   end for ;
end SCAN6_CON ;

   -- SCAN7 FUNCTION
library IEEE ;
use IEEE.Std_Logic_1164.all ;
Library EXEMPLAR;
use EXEMPLAR.EXEMPLAR.all;
entity SCAN7 is
port  ( CLK2  : in std_logic ;
      IND  : in integer range 20 to 40 ;          -- INDEX
      OUTD : out integer range 20 to 40 ); --Z1
 end SCAN7 ;
architecture SCAN7 of SCAN7 is
begin
 process (CLK2,IND)
 begin
        if (CLK2'event and CLK2 = '0') then
         OUTD <= IND ;
        end if;
 end process;
end SCAN7;
configuration SCAN7_CON of SCAN7 is
   for SCAN7
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    end for ;
end SCAN7_CON ;

   -- MAIN PROGRAM  FOR  SCANKEY FUNCTION
library IEEE ;
use IEEE.Std_Logic_1164.all ;
Library EXEMPLAR;
use EXEMPLAR.EXEMPLAR.all;
entity SCAN14567 is
port  ( CLK2  :  in std_logic ;
        POWER , RESET , INC , DEC  : in std_logic;
        OFFPOW  : out std_logic ;
        T : out bit_vector ( 7 downto 0 )) ;
end SCAN14567 ;
architecture  SCAN14567  of  SCAN14567  is
 component SCAN4
 port  ( DEC  : in std_logic ;
        IND  : in integer range  20 to 40 ;
        OUTD : out integer range 15 to 40 );
 end component;
 component SCAN5
 port  ( INC  :  in std_logic ;
        IND  : in integer range  20 to 40 ;
        OUTD : out integer range 20 to 45 );
 end component;

 component SCAN6
 port  ( CLK2  :  in std_logic ;
        POWER , RESET,INC,DEC  : in std_logic;
        IN1  : in integer range  20 to 45 ;          --INC
        IN2  : in integer range  15 to 40 ;          --DEC
        IN3  : in integer range  20 to 40 ;          -- Z1
        OUTD : out integer range  15 to 45 );
 end component;
 component SCAN7
 port  ( CLK2  :  in std_logic ;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        IND  : in integer range  20 to 40 ;          -- INDEX
        OUTD : out integer range  20 to 40 ); --Z1
 end component;
 signal BU1 : integer range  20 to 40 ;
 signal BU2 : integer range  20 to 45 ;
 signal BU3 : integer range  15 to 40 :
 signal OFF1 : std_logic;
 signal INDEX :  integer range  20 to 40 ;
 subtype   TEMPW  is bit_vector( 7 downto 0);
 type  TEMPTB  is array( 20 to 40) of  TEMPW;
 constant   TEMP : TEMPTB  := TEMPTB'(
        "00001010",     -- TEMPERATURE  20 C
        "00010100",     -- TEMPERATURE  21 C
        "00011110",     -- TEMPERATURE  22 C
        "00101000",     -- TEMPERATURE  23 C
        "00110010",     -- TEMPERATURE  24 C
        "00111100",     -- TEMPERATURE  25 C
        "01000110",     -- TEMPERATURE  26 C
        "01010000",     -- TEMPERATURE  27 C
        "01011010",     -- TEMPERATURE  28 C
        "01100100",     -- TEMPERATURE  29 C
        "01101110",     -- TEMPERATURE  30 C
        "01111000",     -- TEMPERATURE  31 C
        "10000010",     -- TEMPERATURE  32 C
        "10001100",     -- TEMPERATURE  33 C
        "10010110",     -- TEMPERATURE  34 C
        "10100000",     -- TEMPERATURE  35 C
        "10101010",     -- TEMPERATURE  36 C
        "10110100",     -- TEMPERATURE  37 C
        "10111110",     -- TEMPERATURE  38 C
        "11001000",     -- TEMPERATURE  39 C
        "11010010");    -- TEMPERATURE  40 C
begin
 S4 : SCAN4
  port map(DEC => DEC,IND => BU1,OUTD => BU3);
 S5 : SCAN5
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  port map(INC => INC,IND => BU1,OUTD => BU2);
 S6 : SCAN6
  port map(CLK2 => CLK2,POWER => POWER,RESET => RESET,
                INC => INC,DEC => DEC,IN1 => BU2,
                        IN2 => BU3,IN3 => BU1,OUTD => INDEX );
 S7 : SCAN7
  port map(CLK2 => CLK2,IND => INDEX,OUTD => BU1);
CLOCK  :  process ( CLK2,OFF1 )
begin
 if ( CLK2'event  and  CLK2 = '0' ) then
   T  <=  TEMP( INDEX ) ;
          OFFPOW <= OFF1 ;                                    --OK--
 end if ;
end process CLOCK ;
PP : process ( POWER,OFF1 )
begin
 if ( POWER'event and POWER = '0' ) then
          OFF1 <= not OFF1 ;
        end if;
end process PP ;
end SCAN14567 ;
configuration  KEY1_CON   of  SCAN14567 is
 for  SCAN14567

  for S4 : SCAN4 use configuration work.SCAN4_CON ;
  end for;
  for S5 : SCAN5 use configuration work.SCAN5_CON ;
  end for;
  for S6 : SCAN6 use configuration work.SCAN6_CON ;
  end for;
  for S7 : SCAN7 use configuration work.SCAN7_CON ;
  end for;
 end for;
end KEY1_CON ;
```

ผ.2 โปรแกรมส่วน BUFFER TEMPERATURE

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
-- INPUT  BUFFER  TEMPERATURE --

Library  IEEE;

use IEEE.Std_Logic_1164.all;

entity TBUFF is

port  ( CLK2 : in std_logic ;

     ADC : in bit_vector ( 7 downto 0 ) ;
     BUF : out bit_vector ( 7 downto 0 )) ;
end TBUFF;
architecture TBUFF of TBUFF is
begin
   process ( CLK2 )
   begin
      if ( CLK2'event  and  CLK2 = '0' ) then
         BUF <= ADC ;
      end if ;
   end process ;
end TBUFF ;
configuration BUFF_CON of TBUFF is
   for  TBUFF
   end for;
end BUFF_CON ;
```

ผ.3 โปรแกรมส่วนสแกนดิสเพลย์

```
 --8BIT TO 3 DIGIT BCD--

Library IEEE;

use IEEE.Std_Logic_1164.all;

entity BITTOBCD1 is
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
port (CLK,CLK1 : in std_logic;
      IND  : in bit_vector(7 downto 0);
      OUTT : out integer range 0 to 15 ;
      OUTU : out integer range 0 to 15 ;
      OUTP : out integer range 0 to 15 );
end  BITTOBCD1;
architecture  BITTOBCD1 of BITTOBCD1 is
 signal MES,LES : integer range 0 to 25;
 signal POINT,UNIT,HES : integer range 0 to 15;
 signal A,B,C,D,E,F,G,H : integer range 0 to 1 ;
begin
 AA : process (CLK1,CLK,IND)
 begin
        if (CLK1 = '0') then
        if (CLK'event and CLK = '1')        then
   if(IND(0) = '1')          then
                  A <= 1 ;
          else
           A <= 0 ;
          end if;
          if(IND(1) = '1')   then
                  B <= 1 ;
          else
           B <= 0 ;
          end if;
          if(IND(2) = '1')   then
                  C <= 1 ;
          else
           C <= 0 ;
          end if;

          if(IND(3) = '1')   then
                  D <= 1 ;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        else
         D <= 0 ;
        end if;
        if(IND(4) = '1')  then
                E <= 1 ;
        else
         E <= 0 ;
        end if;
        if(IND(5) = '1')  then
                F <= 1 ;
        else
         F <= 0 ;
        end if;
        if(IND(6) = '1')  then
                G <= 1 ;
        else
         G <= 0 ;
        end if;
        if(IND(7) = '1')  then
                H <= 1 ;
        else
         H <= 0 ;
        end if;
 end if;
      end if;
end process AA ;
BB : process (CLK1,LES,MES,A,B,C,D,E,F,G,H)
begin
        LES <= A + (2 * B) + (4 * C) + (8 * D) + (6 * E) + (2 * F) + (4 * G) + (8 * H);
        if ( LES < 10 ) then
                POINT <= LES;
         MES <= 9 + E + (3 * F) + (6 * G) + (2 * H);
         if ( MES < 10 ) then
                UNIT <= MES;
           HES <= 1 + H ;
         elsif ( MES < 20 ) then
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                UNIT <= MES - 10 ;
            HES <= 2 + H ;
          else
            UNIT <= MES - 20 ;
            HES <= 3 + H ;
          end if;  -- IF MES
        elsif ( LES < 20 ) then
              POINT <= LES - 10 ;
              MES <= 10 + E + (3 * F) + (6 * G) + (2 * H);
          if ( MES < 10 ) then
              UNIT <= MES;
            HES <= 1 + H ;
          elsif ( MES < 20 ) then
                UNIT <= MES - 10 ;
            HES <= 2 + H ;
          else
            UNIT <= MES - 20 ;
            HES <= 3 + H ;
          end if;  -- IF MES
        else
          POINT <= LES - 20 ;
          MES <= 11 + E + (3 * F) + (6 * G) + (2 * H);
          if ( MES < 10 ) then
              UNIT <= MES;
            HES <= 1 + H ;
          elsif ( MES < 20 ) then
                UNIT <= MES - 10 ;
            HES <= 2 + H ;
          else
            UNIT <= MES - 20 ;
            HES <= 3 + H ;

          end if;  -- IF MES
        end if;   -- IF LES
 end process BB;
CC : process (CLK,CLK1,HES,UNIT,POINT)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
begin
        if (CLK1 = '1') then
         if (CLK'event and CLK = '1') then
          OUTT <= HES ;
    OUTU <= UNIT ;
          OUTP <= POINT ;
   end if;
        end if;
 end process CC ;
end BITTOBCD1 ;
configuration BCD1_CON of BITTOBCD1 is
  for BITTOBCD1
  end for;
end BCD1_CON ;

 --8BIT TO 2 DIGIT BCD PROGRAM
Library IEEE;
use IEEE.Std_Logic_1164.all;
entity BITTOBCD2 is
port (CLK,CLK1 : in std_logic;
    IND  : in bit_vector(7 downto 0);
    OUTT : out integer range 0 to 15 ;
    OUTU : out integer range 0 to 15 );
end  BITTOBCD2;
architecture  BITTOBCD2 of BITTOBCD2 is
 signal MES,LES : integer range 0 to 25;
 signal UNIT,HES : integer range 0 to 15;
 signal A,B,C,D,E,F,G,H : integer range 0 to 1 ;
begin

 AA : process (CLK1,CLK,IND)
 begin
        if (CLK1 = '0') then
         if (CLK'event and CLK = '1')          then
   if(IND(0) = '1')          then
                  A <= 1 ;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
else
 A <= 0 ;
end if;
if(IND(1) = '1')  then
        B <= 1 ;
else
 B <= 0 ;
end if;
if(IND(2) = '1')  then
        C <= 1 ;
else
 C <= 0 ;
end if;
if(IND(3) = '1')  then
        D <= 1 ;
else
 D <= 0 ;
end if;
if(IND(4) = '1')  then
        E <= 1 ;
else
 E <= 0 ;
end if;
if(IND(5) = '1')  then
        F <= 1 ;
else
 F <= 0 ;
end if;

if(IND(6) = '1')  then
        G <= 1 ;
else
 G <= 0 ;
end if;
if(IND(7) = '1')  then
        H <= 1 ;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        else
          H <= 0 ;
        end if;
 end if;
      end if;
end process AA ;
BB : process (CLK1,LES,MES,A,B,C,D,E,F,G,H)
begin
        LES <= A + (2 * B) + (4 * C) + (8 * D) + (6 * E) + (2 * F) + (4 * G) + (8 * H);
        if ( LES < 10 ) then
         MES <= 9 + E + (3 * F) + (6 * G) + (2 * H);
         if ( MES < 10 ) then
               UNIT <= MES;
           HES <= 1 + H ;
         elsif ( MES < 20 ) then
                 UNIT <= MES - 10 ;
           HES <= 2 + H ;
         else
           UNIT <= MES - 20 ;
           HES <= 3 + H ;
         end if;  -- IF MES
        elsif ( LES < 20 ) then
               MES <= 10 + E + (3 * F) + (6 * G) + (2 * H);
         if ( MES < 10 ) then
               UNIT <= MES;
           HES <= 1 + H ;
         elsif ( MES < 20 ) then
                 UNIT <= MES - 10 ;
           HES <= 2 + H ;
         else
           UNIT <= MES - 20 ;
           HES <= 3 + H ;
         end if;  -- IF MES
        else
         MES <= 11 + E + (3 * F) + (6 * G) + (2 * H);
         if ( MES < 10 ) then
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                UNIT <= MES;
              HES <= 1 + H ;
            elsif ( MES < 20 ) then
                    UNIT <= MES - 10 ;
              HES <= 2 + H ;
            else
              UNIT <= MES - 20 ;
              HES <= 3 + H ;
            end if;  -- IF MES
          end if;   -- IF LES
  end process BB;
 CC : process (CLK,CLK1,HES,UNIT)
 begin
        if (CLK1 = '1') then
          if (CLK'event and CLK = '1') then
           OUTT <= HES ;
      OUTU <= UNIT ;
    end if;
          end if;
  end process CC ;
end BITTOBCD2 ;
configuration BCD2_CON of BITTOBCD2 is
   for BITTOBCD2
   end for;
end BCD2_CON ;


--5 DIGIT BCD TO SEVEN SEGMENT AND TDMSCAN
Library IEEE;
use IEEE.Std_Logic_1164.all;
entity TDMSEV is
port (CLK,CLK1 : in std_logic;
          OFFPOW : in std_logic;
    IN1 : in integer range 0 to 15 ;
    IN2 : in integer range 0 to 15 ;
    IN3 : in integer range 0 to 15 ;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    IN4 : in integer range 0 to 15 ;
    IN5 : in integer range 0 to 15 ;
    DATASEV : out bit_vector (7 downto 0 );
    COLSCAN : out bit_vector (2 downto 0 ) );
end TDMSEV;
architecture  TDMSEV of TDMSEV is
 signal PSTATE,NSTATE : integer range 0 to 7 ;
 signal DA1,DA2,DA3,DA4,DA5 : bit_vector(7 downto 0);
begin
 AA : process(CLK1,IN1,IN2,IN3,IN4,IN5)
 begin
          if ( CLK1'event and CLK1 = '0') then
   case  IN1  is   --A,B,C,D,E,F,G,H--     --SEGMENT OUTPUT--
    when  0 =>
     DA1 <= ('1','1','1','1','1','1','0','0'); --(FCh)
    when  1 =>
     DA1 <= ('0','1','1','0','0','0','0','0'); --(60h)
    when  2 =>
     DA1 <= ('1','1','0','1','1','0','1','0'); --(DAh)
    when  3 =>
     DA1 <= ('1','1','1','1','0','0','1','0'); --(F2h)
    when  4 =>
     DA1 <= ('0','1','1','0','0','1','1','0'); --(66h)
    when  5 =>
     DA1 <= ('1','0','1','1','0','1','1','0'); --(B6h)
    when  6 =>
     DA1 <= ('1','0','1','1','1','1','1','0'); --(BEh)
    when  7 =>
     DA1 <= ('1','1','1','0','0','0','0','0'); --(E0h)
    when  8 =>
     DA1 <= ('1','1','1','1','1','1','1','0'); --(FEh)
    when  9 =>
     DA1 <= ('1','1','1','1','0','1','1','0' ); --(F6h)
    when  others =>
     DA1 <= ('0','0','0','0','0','0','0','0' ); --(00h)
   end case;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
case IN2 is   --A,B,C,D,E,F,G,H--     --SEGMENT OUTPUT--
 when 0 =>
  DA2 <= ('1','1','1','1','1','1','0','0'); --(FCh)
 when 1 =>
  DA2 <= ('0','1','1','0','0','0','0','0'); --(60h)
 when 2 =>
  DA2 <= ('1','1','0','1','1','0','1','0'); --(DAh)
 when 3 =>
  DA2 <= ('1','1','1','1','0','0','1','0'); --(F2h)
 when 4 =>
  DA2 <= ('0','1','1','0','0','1','1','0'); --(66h)
 when 5 =>
  DA2 <= ('1','0','1','1','0','1','1','0'); --(B6h)
 when 6 =>
  DA2 <= ('1','0','1','1','1','1','1','0'); --(BEh)
 when 7 =>
  DA2 <= ('1','1','1','0','0','0','0','0'); --(E0h)
 when 8 =>
  DA2 <= ('1','1','1','1','1','1','1','0'); --(FEh)
 when 9 =>
  DA2 <= ('1','1','1','1','0','1','1','0' ); --(F6h)
 when others =>
  DA2 <= ('0','0','0','0','0','0','0','0' ); --(00h)
end case;

case IN3 is   --A,B,C,D,E,F,G,H--     --SEGMENT OUTPUT--
 when 0 =>
  DA3 <= ('1','1','1','1','1','1','0','0'); --(FCh)
 when 1 =>
  DA3 <= ('0','1','1','0','0','0','0','0'); --(60h)
 when 2 =>
  DA3 <= ('1','1','0','1','1','0','1','0'); --(DAh)
 when 3 =>
  DA3 <= ('1','1','1','1','0','0','1','0'); --(F2h)
 when 4 =>
  DA3 <= ('0','1','1','0','0','1','1','0'); --(66h)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
when 5 =>
 DA3 <= ('1','0','1','1','0','1','1','0'); --(B6h)
when 6 =>
 DA3 <= ('1','0','1','1','1','1','1','0'); --(BEh)
when 7 =>
 DA3 <= ('1','1','1','0','0','0','0','0'); --(E0h)
when 8 =>
 DA3 <= ('1','1','1','1','1','1','1','0'); --(FEh)
when 9 =>
 DA3 <= ('1','1','1','1','0','1','1','0' ); --(F6h)
when others =>
 DA3 <= ('0','0','0','0','0','0','0','0' ); --(00h)
end case;
case IN4 is   --A,B,C,D,E,F,G,H--     --SEGMENT OUTPUT--
when 0 =>
 DA4 <= ('1','1','1','1','1','1','0','0'); --(FCh)
when 1 =>
 DA4 <= ('0','1','1','0','0','0','0','0'); --(60h)
when 2 =>
 DA4 <= ('1','1','0','1','1','0','1','0'); --(DAh)
when 3 =>
 DA4 <= ('1','1','1','1','0','0','1','0'); --(F2h)
when 4 =>
 DA4 <= ('0','1','1','0','0','1','1','0'); --(66h)
when 5 =>
 DA4 <= ('1','0','1','1','0','1','1','0'); --(B6h)
when 6 =>
 DA4 <= ('1','0','1','1','1','1','1','0'); --(BEh)
when 7 =>
 DA4 <= ('1','1','1','0','0','0','0','0'); --(E0h)
when 8 =>
 DA4 <= ('1','1','1','1','1','1','1','0'); --(FEh)
when 9 =>
 DA4 <= ('1','1','1','1','0','1','1','0' ); --(F6h)
when others =>
 DA4 <= ('0','0','0','0','0','0','0','0' ); --(00h)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  end case;
  case IN5 is   --A,B,C,D,E,F,G,H--     --SEGMENT OUTPUT--
   when 0 =>
    DA5 <= ('1','1','1','1','1','1','0','0'); --(FCh)
   when 1 =>
    DA5 <= ('0','1','1','0','0','0','0','0'); --(60h)
   when 2 =>
    DA5 <= ('1','1','0','1','1','0','1','0'); --(DAh)
   when 3 =>
    DA5 <= ('1','1','1','1','0','0','1','0'); --(F2h)
   when 4 =>
    DA5 <= ('0','1','1','0','0','1','1','0'); --(66h)
   when 5 =>
    DA5 <= ('1','0','1','1','0','1','1','0'); --(B6h)
   when 6 =>
    DA5 <= ('1','0','1','1','1','1','1','0'); --(BEh)
   when 7 =>
    DA5 <= ('1','1','1','0','0','0','0','0'); --(E0h)
   when 8 =>
    DA5 <= ('1','1','1','1','1','1','1','0'); --(FEh)
   when 9 =>
    DA5 <= ('1','1','1','1','0','1','1','0' ); --(F6h)
   when others =>
    DA5 <= ('0','0','0','0','0','0','0','0' ); --(00h)
  end case;
        end if;
end process AA;
BB : process(CLK,OFFPOW,PSTATE,DA1,DA2,DA3,DA4,DA5)
begin
        if ( CLK'event and CLK = '1' ) then
          PSTATE <= NSTATE ;
        end if;
 if ( CLK'event  and  CLK = '0' ) then
  if ( OFFPOW = '1') then
            case  PSTATE  is
    when 0 =>      -- COLUME OUTPUT --
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
      DATASEV <= DA1 ;    --OUTT of STEMP
      COLSCAN <= "000" ;  --(0h)
      NSTATE <= 1 ;
     when 1 =>
      DATASEV <= DA2 ;            --OUTU of STEMP
      COLSCAN <= "001" ;  --(1h)
      NSTATE <= 2 ;
     when 2 =>
      DATASEV <=DA3 ;             --OUTP of STEMP
      COLSCAN <= "010" ;  --(2h)
      NSTATE <= 3 ;
     when 3 =>
      DATASEV <= DA4 ;            --OUTT of KTEMP
      COLSCAN <= "011" ;  --(3h)
      NSTATE <= 4 ;
     when 4 =>
      DATASEV <= DA5 ;            --OUTU of STEMP
      COLSCAN <= "100" ;  --(4h)
      NSTATE <= 0 ;
     when others =>
      DATASEV <= "00000000";
      COLSCAN <= "111" ;
                  NSTATE <= 0 ;
    end case ;
   else
    DATASEV <= "00000000";      -- OFF DISPLAY
    COLSCAN <= "111";
    NSTATE <= 0;
   end if ;          -- IF OFFPOW
  end if ;           -- IF CLK
 end process BB;
end TDMSEV ;
configuration TDM_CON of TDMSEV is
  for TDMSEV
  end for;
end TDM_CON ;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
-- MAIN PROGRAM FOR SCANDISPLAY FUNCTION
Library IEEE;
use IEEE.Std_Logic_1164.all;
entity SCANDIS is
port (CLK,CLK1  : in std_logic;
    OFFPOW : in std_logic ;
    KDATA  : in bit_vector ( 7 downto 0 ) ;
    SDATA  : in bit_vector ( 7 downto 0 ) ;
    DATASEV  : out bit_vector ( 7 downto 0 ) ;
    COLSCAN  : out bit_vector ( 2 downto 0 )) ;
end SCANDIS ;
 architecture SCANDIS of SCANDIS is
 component BITTOBCD1
 port (CLK,CLK1 : in std_logic;
     IND  : in bit_vector(7 downto 0);
     OUTT : out integer range 0 to 15 ;
     OUTU : out integer range 0 to 15 ;
     OUTP : out integer range 0 to 15 );
 end component;
 component BITTOBCD2
 port (CLK,CLK1 : in std_logic;
     IND  : in bit_vector(7 downto 0);
     OUTT : out integer range 0 to 15 ;
     OUTU : out integer range 0 to 15 );
 end component;
 component TDMSEV
 port (CLK,CLK1 : in std_logic;
     OFFPOW : in std_logic;
     IN1 : in integer range 0 to 15 ;
     IN2 : in integer range 0 to 15 ;
     IN3 : in integer range 0 to 15 ;
     IN4 : in integer range 0 to 15 ;
     IN5 : in integer range 0 to 15 ;
     DATASEV : out bit_vector (7 downto 0 );
     COLSCAN : out bit_vector (2 downto 0 ) );
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
 end component;
 signal BU1,BU2,BU3,BU4,BU5 : integer range 0 to 15 ;
begin
 BCD1 : BITTOBCD1
  port map( CLK => CLK,CLK1 => CLK1,IND => SDATA,
            OUTT => BU1,OUTU => BU2,OUTP => BU3 );
 BCD2 :  BITTOBCD2
  port map( CLK => CLK,CLK1 => CLK1,IND => KDATA,
            OUTT => BU4,OUTU => BU5);
 OSCAN : TDMSEV
  port map( CLK => CLK,CLK1 => CLK1,OFFPOW => OFFPOW,
            IN1 => BU1,IN2 => BU2,IN3 => BU3,IN4 => BU4,
                  IN5 => BU5,DATASEV => DATASEV,COLSCAN => COLSCAN);
end SCANDIS ;
configuration  DIS_CON  of  SCANDIS  is
 for SCANDIS
  for BCD1 : BITTOBCD1 use configuration work.BCD1_CON ;
  end for;
  for BCD2 : BITTOBCD2 use configuration work.BCD2_CON ;
  end for;
  for OSCAN : TDMSEV use configuration work.TDM_CON ;
  end for;
 end  for ;
end DIS_CON ;
```

ผ.4 โปรแกรมส่วนสร้างสัญญาณนาฬิกา 3 เฟส

```
   -- CLOCK GENERATOR --

Library IEEE;

use IEEE.Std_Logic_1164.all;

entity  CLKSYN  is
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
port ( CLK  :  in std_logic ;
       RESET :  in std_logic ;
       CLK1 , CLK2 , CLK3 :  out  std_logic ) ;
end CLKSYN ;
architecture CLKSYN of CLKSYN  is
   signal  COUNT  :  integer range 0 to 15 ;
begin
   COUNTER : process( CLK , RESET )
   begin
      if  ( RESET = '0' ) then
          COUNT <= 0 ;
      elsif  ( CLK'event  and  CLK = '1' ) then
          if ( COUNT >= 9 ) then
             COUNT <= 0 ;
          else
             COUNT <= COUNT + 1 ;
          end if ;
      end if ;
   end process COUNTER ;
          CLOCK : process(CLK,COUNT)
   begin
    if  ( CLK'event  and  CLK = '0' ) then

                    case COUNT is
       when 0 to 1 =>
        CLK1 <= '0';
        CLK2 <= '0';
        CLK3 <= '1';
       when 2 to 3 =>
        CLK1 <= '0';
        CLK2 <= '1';
        CLK3 <= '0';
       when 4 =>
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        CLK1 <= '0';
        CLK2 <= '0';
        CLK3 <= '1';
       when 5 to 6 =>
        CLK1 <= '1';
        CLK2 <= '0';
        CLK3 <= '1';
       when others =>
        CLK1 <= '0';
        CLK2 <= '0';
        CLK3 <= '1';
      end case;
        end if;
         end process  CLOCK ;
end CLKSYN ;
configuration  CLKS_CON  of  CLKSYN  is
   for  CLKSYN
   end for ;
end  CLKS_CON ;
```

ผ.5 โปรแกรมส่วน FUZZY ALGORITHM

```
-- Kp and Ki TABLE --

package KPKIT1 is

 type ARR_2D1 is array(0 to 20,0 to 10) of integer range 0 to 40;

 type ARR_2D2 is array(0 to 20,0 to 10) of integer range 0 to 40;
  --TABLE KP--
  constant KPTB : ARR_2D1 :=
   ((16,16,16,16,16,16,16,16,16,16,16),
    (16,16,16,16,16,16,16,16,16,16,16),
    (16,16,16,16,16,16,16,16,16,16,16),
    (16,16,16,16,16,16,16,16,16,16,16),
    (16,16,16,16,16,16,16,16,16,16,16),
    (16,16,16,16,16,16,16,16,16,16,16),
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
(16,16,16,16,16,16,16,16,16,16,20),
(16,16,16,16,16,16,16,16,16,16,20),
(16,16,16,16,16,16,16,16,16,16,20),
(16,16,16,16,16,16,16,16,16,16,20),
(16,20,20,20,20,20,20,20,20,20,20),
(16,20,20,20,20,20,20,20,20,20,20),
(16,20,20,20,20,20,20,20,20,20,20),
(16,20,20,20,20,20,20,20,20,20,20),
(16,20,20,20,20,20,20,20,20,20,20),
(16,20,20,20,20,20,20,20,20,20,20),
(20,20,27,27,27,27,27,27,27,27,27),
(20,20,27,27,27,27,27,27,27,27,27),
(20,20,27,27,27,27,27,27,27,27,27),
(20,20,27,27,27,27,27,27,27,27,27),
(27,27,27,27,27,27,27,27,27,27,27));

--TABLE KI--
constant KITB : ARR_2D2 :=
(( 9, 9, 9, 9, 9, 9,10,10,10,10,10),
 ( 9, 9, 9, 9, 9, 9,10,10,10,10,10),
 ( 9, 9, 9, 9, 9, 9,10,10,10,10,10),
 ( 9, 9, 9, 9, 9, 9,10,10,10,10,10),
 ( 9, 9, 9, 9, 9, 9,10,10,10,10,10),
 ( 9, 9, 9, 9, 9, 9,10,10,10,10,10),
 (10,10,10,10,10,10,10,10,10,10,10),
 (10,10,10,10,10,10,10,10,10,10,10),
 (10,10,10,10,10,10,10,10,10,10,10),
 (10,10,10,10,10,10,10,10,10,10,10),
 (10,10,10,10,10,10,10,10,10,10,10),
 (10,10,10,10,10,10,10,10,10,10,10),
 (10,10,10,10,10,10,10,10,10,10,10),
 (10,10,10,10,10,10,10,10,10,10,10),
 (10,10,10,10,10,10,10,10,10,10,10),
 (10,10,10,10,10,10,10,10,10,10,10),
 (10,10, 9, 9, 9, 9, 9, 9, 9, 9, 9),
 (10,10, 9, 9, 9, 9, 9, 9, 9, 9, 9),
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    (10,10, 9, 9, 9, 9, 9, 9, 9, 9, 9),
    (10,10, 9, 9, 9, 9, 9, 9, 9, 9, 9),
    ( 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9));
end KPKIT1 ;

 --MAIN Kp and Ki PROGRAM--
Library IEEE;
use IEEE.Std_Logic_1164.all;
use work.KPKIT1.all;
entity KPKI1N  is
port ( ET,DE : in bit_vector(8 downto 0);
       KP : out bit_vector(4 downto 0);
       KI : out bit_vector(3 downto 0) );
end KPKI1N;
architecture KPKI1N of KPKI1N is
 signal CO,RO : integer range 0 to 511 ;
 signal KPDEC,KIDEC :integer range 0 to 40 ;
 signal COL :integer range 0 to 20 ;
 signal ROW :integer range 0 to 10 ;
 procedure  BITTODEC (X : in bit_vector(8 downto 0);
      signal DA1 : out  integer range 0 to 511) is
  variable HAM,MAM,DAM,COW1,COW2,HES,MES,LES : integer range 0 to 20;
          variable VAL : integer range 0 to 511 ;
 begin
  HES := 0 ;
  MES := 0 ;
  LES := 0 ;
  for J in 0 to 7 loop  -- THE 8TH BIT IS A SIGN BIT--
   case J  is
    when  0 =>
     HAM := 0 ; MAM := 0 ; DAM := 1 ;
    when  1 =>
     HAM := 0 ; MAM := 0 ; DAM := 2 ;
    when  2 =>
     HAM := 0 ; MAM := 0 ; DAM := 4 ;
    when  3 =>
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    HAM := 0 ; MAM := 0 ; DAM := 8 ;
   when  4 =>
    HAM := 0 ; MAM := 1 ; DAM := 6 ;
   when  5 =>
    HAM := 0 ; MAM := 3 ; DAM := 2 ;
   when  6 =>
    HAM := 0 ; MAM := 6 ; DAM := 4 ;
   when  7 =>
    HAM := 1 ; MAM := 2 ; DAM := 8 ;
   when  others =>
    HAM := 0 ; MAM := 0 ; DAM := 1 ;
  end case ;

  if (  X( J ) = '1' ) then
   LES := LES + DAM ;
    if ( LES >= 10 ) then
     LES := LES - 10 ;
     COW1 := 1 ;
    else
     COW1 := 0 ;
    end if ;
   MES := MES + MAM + COW1 ;
    if ( MES >= 10 ) then
     MES := MES - 10 ;
     COW2 := 1 ;
    else
     COW2 := 0 ;
    end if ;
   HES := HES +HAM + COW2 ;
  end if ;
 end loop ;
  VAL := (HES*100)+(MES*10)+LES ;
  DA1 <= VAL ;
end BITTODEC ;
begin
  BITTODEC( ET , CO );
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
BITTODEC( DE , RO );
COLROW : process(CO,RO)
begin
    if ( CO > 20 ) then
        COL <= 20;
    else
        COL <= CO;
    end if;
    if ( RO > 10 ) then
        ROW <= 10;
    else
        ROW <= RO;
    end if;
end process COLROW ;
KPDEC <= KPTB(COL,ROW);
KIDEC <= KITB(COL,ROW);
GETKPKI : process(KPDEC,KIDEC)
begin
    case KPDEC is
        when 27 =>
            KP <= "11011";
        when 20 =>
            KP <= "10100";
        when 16 =>
            KP <= "10000";
        when others =>
            KP <= "00000";
    end case;
    case KIDEC is
        when 9 =>
            KI <= "1001";
        when 10 =>
            KI <= "1010";
        when others =>
            KI <= "0000";
    end case;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    end process GETKPK1 ;
end KPKI1N ;
configuration KPKI1N_CON of KPKI1N is
    for KPKI1N
    end for;
end KPKI1N_CON ;

-- SHIFT REGISTER PROGRAM --
Library IEEE;
use IEEE.Std_Logic_1164.all;
entity SHREG1 is
port ( CLK1 : in std_logic;
     IND : in bit_vector( 8 downto 0 );
     OUTD : out bit_vector( 8 downto 0 ));
end SHREG1 ;
architecture SHREG1 of SHREG1 is
begin
 process( CLK1 )
 begin
  if ( CLK1'event and CLK1 = '1') then
   OUTD <= IND;
  end if;
 end process;
end SHREG1;
configuration SHIFT1_CON of SHREG1 is
    for SHREG1
    end for;
end SHIFT1_CON ;

 -- SUBTRACTIVE PROGRAM --
library IEEE;
use ieee.std_logic_1164.all;
library EXEMPLAR;
use EXEMPLAR.EXEMPLAR.all;
entity SUBTR3 is
port(INN1,INN2 : in bit_vector(8 downto 0);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    OUT2 : out bit_vector(8 downto 0));
end SUBTR3;
architecture SUBTR3 of SUBTR3 is
  signal OUT1 : bit_vector(9 downto 0);
begin
 process ( INN1,INN2,OUT1 )
 begin
   OUT1 <= sub2(INN1,INN2);
   OUT2(0) <= OUT1(0);
   OUT2(1) <= OUT1(1);
   OUT2(2) <= OUT1(2);
   OUT2(3) <= OUT1(3);
   OUT2(4) <= OUT1(4);
   OUT2(5) <= OUT1(5);
   OUT2(6) <= OUT1(6);
   OUT2(7) <= OUT1(7);
   OUT2(8) <= OUT1(8);
 end process;
end SUBTR3;
configuration SUB3_CON of SUBTR3 is
   for SUBTR3
   end for;
end SUB3_CON;

 -- ABSOLUTE VALUE PROGRAM --
Library IEEE;
use IEEE.Std_Logic_1164.all;
Library EXEMPLAR;
use EXEMPLAR.EXEMPLAR.all;
entity ABSOL1 is
port ( SUBT : in bit_vector(8 downto 0);
     ABST : out bit_vector(8 downto 0));
end ABSOL1;
architecture ABSOL1 of ABSOL1 is
begin
   process ( SUBT )
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    begin
      if ( SUBT(8) = '1') then
         ABST <= comp2( SUBT );
      else
         ABST <= SUBT;
      end if;
    end process;
end ABSOL1;
configuration ABS1_CON of ABSOL1 is
    for ABSOL1
    end for;
end ABS1_CON;


 --FUZZY ALGORITHM PROGRAM--
Library EXEMPLAR ;
use EXEMPLAR.EXEMPLAR.all ;
Library IEEE ;
use IEEE.Std_Logic_1164.all ;
entity FUZZALG1N is
port ( CLK1: in std_logic ;
     STEMP,BTEMP :in bit_vector(7 downto 0);
     KP : out bit_vector(4 downto 0);
     KI : out bit_vector(3 downto 0);
     ERTEMP : out bit_vector(8 downto 0));
end FUZZALG1N;
architecture FUZZALG1N of FUZZALG1N is
 component SUBTR3
 port ( INN1,INN2 : in bit_vector(8 downto 0);
      OUT2 : out bit_vector(8 downto 0));
 end component;
 component SHREG1
 port ( CLK1 : in std_logic ;
      IND : in bit_vector(8 downto 0);
      OUTD : out bit_vector(8 downto 0));
 end component;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
component ABSOL1
port ( SUBT : in bit_vector(8 downto 0);
     ABST : out bit_vector(8 downto 0));
end component;
component KPKI1N
port ( ET,DE : in bit_vector(8 downto 0);
     KP : out bit_vector(4 downto 0);
     KI : out bit_vector(3 downto 0));
end component;
signal  STEMP1,BTEMP1,ETEMP,ETEMP1,
     DETEMP,ABSET,ABSDET : bit_vector(8 downto 0);
begin


ERTEMP <= ETEMP ;
SUB1 : SUBTR3
 port map( INN1 => STEMP1,INN2 => BTEMP1,OUT2 => ETEMP);
REG : SHREG1
 port map(CLK1 => CLK1,IND => ETEMP,OUTD => ETEMP1);
SUB2 : SUBTR3
 port map( INN1 => ETEMP,INN2 => ETEMP1,OUT2 => DETEMP);
ABS1 : ABSOL1
 port map( SUBT => ETEMP,ABST => ABSET);
ABS2 : ABSOL1
 port map( SUBT => DETEMP,ABST => ABSDET);
GETKPKI : KPKI1N
 port map( ET => ABSET,DE => ABSDET,KP => KP,KI => KI);
SYNC : process (CLK1)
begin
 if (CLK1'event and CLK1 = '1')then
   STEMP1(0) <= STEMP(0);
   STEMP1(1) <= STEMP(1);
   STEMP1(2) <= STEMP(2);
   STEMP1(3) <= STEMP(3); ----------------------
   STEMP1(4) <= STEMP(4); -- EXTEND INPUT DATA --
   STEMP1(5) <= STEMP(5); -- 8 BIT TO 9 BIT -----
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    STEMP1(6) <= STEMP(6); ----------------------
    STEMP1(7) <= STEMP(7);
    STEMP1(8) <= '0';
    BTEMP1(0) <= BTEMP(0);
    BTEMP1(1) <= BTEMP(1);
    BTEMP1(2) <= BTEMP(2);
    BTEMP1(3) <= BTEMP(3);
    BTEMP1(4) <= BTEMP(4);
    BTEMP1(5) <= BTEMP(5);
    BTEMP1(6) <= BTEMP(6);
    BTEMP1(7) <= BTEMP(7);
    BTEMP1(8) <= '0';
   end if;
  end process SYNC ;
end FUZZALG1N;
configuration FUZZY1N_CON of FUZZALG1N is
 for FUZZALG1N
  for SUB1 : SUBTR3 use configuration work.SUB3_CON ;
  end for;
  for REG : SHREG1 use configuration work.SHIFT1_CON ;
  end for;
  for SUB2 : SUBTR3 use configuration work.SUB3_CON ;
  end for;
  for ABS1 : ABSOL1 use configuration work.ABS1_CON ;
  end for;
  for ABS2 : ABSOL1 use configuration work.ABS1_CON ;
  end for;
  for GETKPKI : KPKI1N use configuration work.KPKI1N_CON ;
  end for;
 end for;
end FUZZY1N_CON ;

 ------------------------------------
```

ผ.6 โปรแกรมส่วน พีไอคอนโทรล

```
-- SHIFT REGISTER PROGRAM FOR PI CONTROLLER --
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Library EXEMPLAR ;
use EXEMPLAR.EXEMPLAR.all ;
Library IEEE;
use IEEE.Std_Logic_1164.all;
entity PISREG1N is
port ( CLK,CLK1 : in std_logic;
     IND : in bit_vector( 9 downto 0 );
     OUTD : out bit_vector( 8 downto 0 ));
end PISREG1N ;
architecture PISREG1N of PISREG1N is
 signal A,B,C,D,E : bit;
 signal BB1 : bit_vector( 8 downto 0 );
begin
 INDATA : process( CLK )
 begin
  if ( CLK'event and CLK = '1') then
          BB1(0) <= IND(0);
    BB1(1) <= IND(1);
    BB1(2) <= IND(2);
    BB1(3) <= IND(3);
    BB1(4) <= IND(4);
    BB1(5) <= IND(5);
    BB1(6) <= IND(6);
    BB1(7) <= IND(7);
    BB1(8) <= IND(9);
    A <= IND(9);
    B <= IND(8);
    C <= IND(7);
    D <= IND(6);
    E <= IND(5);
        end if;
 end process INDATA;

 OUTDATA : process( CLK1 )
 begin
  if ( CLK1'event and CLK1 = '0') then
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    if A = '1' then
      if (B='1' and C='1' and D='1' and E='1') then
       OUTD <= BB1;
      else
       OUTD <= "111100001"; -- (-31 DEC) MAXIMUM IN
      end if;                    -- NEGATIVE
    else
      if (B='0' and C='0' and D='0' and E='0') then
       OUTD <= BB1;
      else
       OUTD <= "000011111"; -- (31 DEC)  MAXIMUM IN
      end if;                                  -- POSITIVE
    end if;
  end if;
 end process OUTDATA;
end PISREG1N;
configuration PIREG1N_CON of PISREG1N is
   for PISREG1N
   end for;
end PIREG1N_CON ;
-- 17BIT TO 16BIT CONVERTER AND CONTROL SETTING PROGRAM --
          -- ( CBS : CHANGE BEFORE SEND )
Library IEEE;
use IEEE.Std_Logic_1164.all;
entity CBS1 is
port ( CLK : in std_logic;
     OFFPOW : in std_logic;
     IND : in bit_vector(16 downto 0);
     OUTD : out bit_vector(15 downto 0));
end CBS1 ;
architecture CBS1 of CBS1 is
 signal A,B : bit;
 signal AA1 : bit_vector(16 downto 0);
begin
 INDATA : process( CLK )
 begin
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  if (CLK'event and CLK = '1') then
    AA1 <= IND;
    A <= IND(16);
    B <= IND(15);
          end if;
 end process INDATA;
OUTDATA : process(CLK,OFFPOW)
 begin
  if (OFFPOW = '1') then
   if (CLK'event and CLK = '0') then
    if A = '1' then
      if B ='1' then
       OUTD(0) <= not AA1(0);
       OUTD(1) <= not AA1(1);
       OUTD(2) <= not AA1(2);
                  OUTD(3) <= not AA1(3);
       OUTD(4) <= not AA1(4);
       OUTD(5) <= not AA1(5);
       OUTD(6) <= not AA1(6);
       OUTD(7) <= not AA1(7);
       OUTD(8) <= not AA1(8);
       OUTD(9) <= not AA1(9);
       OUTD(10) <= not AA1(10);
       OUTD(11) <= not AA1(11);
       OUTD(12) <= not AA1(12);
       OUTD(13) <= not AA1(13);
       OUTD(14) <= not AA1(14);
       OUTD(15) <= AA1(16);
                          else
       OUTD <= "1111111111111111";
      end if;
    else
      if B ='0' then
       OUTD(0) <= not AA1(0);
       OUTD(1) <= not AA1(1);
       OUTD(2) <= not AA1(2);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                OUTD(3) <= not AA1(3);
          OUTD(4) <= not AA1(4);
          OUTD(5) <= not AA1(5);
          OUTD(6) <= not AA1(6);
          OUTD(7) <= not AA1(7);
          OUTD(8) <= not AA1(8);
          OUTD(9) <= not AA1(9);
          OUTD(10) <= not AA1(10);
          OUTD(11) <= not AA1(11);
          OUTD(12) <= not AA1(12);
          OUTD(13) <= not AA1(13);
          OUTD(14) <= not AA1(14);
          OUTD(15) <= AA1(16);
        else
         OUTD <= "0000000000000000";
        end if;
      end if; -- IF OF A
    end if; -- IF CLK
    else
     OUTD <= "0000000000000000";
    end if;
  end process OUTDATA;
end CBS1;
configuration CBS1_CON of CBS1 is
    for CBS1
    end for;
end CBS1_CON ;

       -- MULTIPLY  11 BIT  X  5 BIT  --
Library  IEEE;
use IEEE.Std_Logic_1164.all;
Library EXEMPLAR;
use EXEMPLAR.EXEMPLAR.all;
entity MTP115 is
port  ( QUE1 : in bit_vector ( 10 downto 0 ) ;
     QUE2 : in bit_vector ( 4 downto 0 ) ;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
      ANS : out bit_vector ( 15 downto 0 )) ;
end MTP115;
architecture  MTP115 of MTP115 is
begin
  ANS <= mult2(QUE1,QUE2);
end MTP115 ;
configuration M115_CON of MTP115 is
   for  MTP115
   end for;
end M115_CON ;

       -- MULTIPLY  9 BIT  X  6 BIT  --
Library  IEEE;
use IEEE.Std_Logic_1164.all;
Library EXEMPLAR;
use EXEMPLAR.EXEMPLAR.all;
entity MTP96 is
port  ( QUE1 : in bit_vector ( 8.downto 0 ) ;
      QUE2 : in bit_vector ( 5 downto 0 ) ;
      ANS : out bit_vector ( 14 downto 0 )) ;
end MTP96;
architecture  MTP96 of MTP96 is
begin
  ANS <= mult2(QUE1,QUE2);
end MTP96 ;
configuration M96_CON of MTP96 is
   for  MTP96
   end for;
end M96_CON ;

       -- 16 BIT and 15 BIT ADDER  --
Library  IEEE;
use IEEE.Std_Logic_1164.all;
Library EXEMPLAR;
use EXEMPLAR.EXEMPLAR.all;
entity ADDER16A is
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
port ( QUE1 : in bit_vector ( 14 downto 0 ) ;
       QUE2 : in bit_vector ( 15 downto 0 ) ;
       ANS : out bit_vector ( 16 downto 0 )) ;
end ADDER16A ;
architecture ADDER16A of ADDER16A is
begin
  ANS <= add2(QUE1,QUE2);
end ADDER16A ;
configuration ADD16A_CON of ADDER16A is
   for ADDER16A
   end for;
end ADD16A_CON ;


       -- 10 BIT ADDER --
Library IEEE;
use IEEE.Std_Logic_1164.all;
Library EXEMPLAR;
use EXEMPLAR.EXEMPLAR.all;
entity ADDER10 is
port ( QUE1 : in bit_vector( 9 downto 0 );
       QUE2 : in bit_vector( 8 downto 0 );
       ANS : out bit_vector( 10 downto 0 ));
end ADDER10 ;
architecture ADDER10 of ADDER10 is
begin
  ANS <= add2(QUE1,QUE2);
end ADDER10 ;
configuration ADD10_CON of ADDER10 is
   for ADDER10
   end for;
end ADD10_CON ;


       -- 9 BIT ADDER --
Library IEEE;
use IEEE.Std_Logic_1164.all;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Library EXEMPLAR;
use EXEMPLAR.EXEMPLAR.all;
entity ADDER9 is
port  ( QUE1 : in bit_vector ( 8 downto 0 ) ;
        QUE2 : in bit_vector ( 8 downto 0 ) ;
        ANS : out bit_vector ( 9 downto 0 )) ;
end ADDER9 ;
architecture  ADDER9 of ADDER9 is
begin
   ANS <= add2(QUE1,QUE2);
end ADDER9 ;
configuration ADD9_CON of ADDER9 is
    for  ADDER9
    end for; end ADD9_CON ;

 -- PI CONTROLLER PROGRAM--
Library EXEMPLAR ;
use EXEMPLAR.EXEMPLAR.all ;
Library IEEE ;
use IEEE.Std_Logic_1164.all ;
entity PICONTRN is
port ( CLK,CLK1: in std_logic;
      OFFPOW : in std_logic;
      KP : in bit_vector(4 downto 0);
      KI : in bit_vector(3 downto 0);
      ERTEMP :in bit_vector(8 downto 0);
      OUT16 : out bit_vector(15 downto 0));
end PICONTRN;
architecture PICONTRN of PICONTRN is
 component PISREG1N
 port ( CLK.CLK1 : in std_logic;
       IND : in bit_vector( 9 downto 0 );
       OUTD : out bit_vector( 8 downto 0 ));
 end component;
 component ADDER9
 port  ( QUE1 : in bit_vector ( 8 downto 0 ) ;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        QUE2 : in bit_vector ( 8 downto 0 ) ;
        ANS : out bit_vector ( 9 downto 0 )) ;
 end component;
 component ADDER10
 port  ( QUE1 : in bit_vector(9 downto 0);
        QUE2 : in bit_vector(8 downto 0);
        ANS  : out bit_vector(10 downto 0));
 end component;
 component ADDER16A
 port  ( QUE1 : in bit_vector(14 downto 0);
        QUE2 : in bit_vector(15 downto 0);
        ANS  : out bit_vector(16 downto 0));
 end component;
 component MTP96
 port  ( QUE1 : in bit_vector(8 downto 0);
        QUE2 : in bit_vector(5 downto 0);
        ANS  : out bit_vector(14 downto 0));
 end component;
 component MTP115
 port  ( QUE1 : in bit_vector(10 downto 0);
        QUE2 : in bit_vector(4 downto 0);
        ANS  : out bit_vector(15 downto 0));
 end component;
 component CBS1
 port(CLK : in std_logic;
    OFFPOW : in std_logic;
    IND : in bit_vector(16 downto 0);
    OUTD : out bit_vector(15 downto 0));
 end component;
 signal ETEMP : bit_vector(8 downto 0);
 signal KPS : bit_vector(5 downto 0);
 signal KIS : bit_vector(4 downto 0);
 signal BU1 : bit_vector(14 downto 0);
 signal BU2 : bit_vector(9 downto 0);
 signal BU3 : bit_vector(8 downto 0);
 signal BU4 : bit_vector(10 downto 0);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
 signal BU5 : bit_vector(15 downto 0);
 signal BU6 : bit_vector(16 downto 0);
begin
 ADD9B : ADDER9
  port map( QUE1 => ETEMP,QUE2 => BU3,ANS => BU2);
 SHIFT : PISREG1N
  port map(CLK => CLK,CLK1 => CLK1,IND => BU2,OUTD => BU3);
 ADD10B : ADDER10
  port map( QUE1 => BU2,QUE2 => BU3,ANS => BU4);
 MULKI : MTP115
  port map( QUE1 => BU4,QUE2 => KIS,ANS => BU5);
 MULKP : MTP96
  port map( QUE1 => ETEMP,QUE2 => KPS,ANS => BU1);
 ADD16B : ADDER16A
  port map( QUE1 => BU1,QUE2 => BU5,ANS => BU6);
 CONSIG : CBS1
  port map( CLK => CLK,OFFPOW => OFFPOW,IND => BU6,OUTD => OUT16);
 SYNC : process (CLK1)
 begin
  if (CLK1'event and CLK1 = '0')then
   ETEMP <= ERTEMP;
   KPS(0) <= KP(0);
   KPS(1) <= KP(1);  -------------------------
   KPS(2) <= KP(2);  -- EXTEND INPUT KP TO ---
   KPS(3) <= KP(3);  -- 6 BIT FOR SIGN DATA --
   KPS(4) <= KP(4);  -------------------------
   KPS(5) <= '0';
   KIS(0) <= KI(0);
   KIS(1) <= KI(1);  -------------------------
   KIS(2) <= KI(2);  -- EXTEND INPUT KI TO ---
   KIS(3) <= KI(3);  -- 5 BIT FOR SIGN DATA --
   KIS(4) <= '0';    -------------------------
  end if;
 end process SYNC ;
end PICONTRN;
configuration PIN_CON of PICONTRN is
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  for PICONTRN
   for ADD9B : ADDER9 use configuration work.ADD9_CON ;
   end for;
   for ADD10B : ADDER10  use configuration work.ADD10_CON ;
   end for;
   for ADD16B : ADDER16A use configuration work.ADD16A_CON ;
   end for;
   for SHIFT : PISREG1N use configuration work.PIREG1N_CON ;
   end for;
   for MULKP : MTP96 use configuration work.M96_CON ;
   end for;
   for MULKI : MTP115 use configuration work.M115_CON ;
   end for;
   for CONSIG : CBS1 use configuration work.CBS1_CON ;
   end for;
  end for;
end PIN_CON ;
```

ผ.7 โปรแกรมหลักของระบบ

```
□
 -- MAIN PROGRAM FOR TOTAL SYSTEM --
□
Library EXEMPLAR ;
□
use EXEMPLAR.EXEMPLAR.all ;
Library IEEE ;
use IEEE.Std_Logic_1164.all ;
entity UNIFY is
port (CLK : in std_logic;
    POWER , RESET , INC , DEC  : in std_logic;
    ADC : in bit_vector(7 downto 0);
    DATASEV : out bit_vector(7 downto 0);
    COLSCAN : out bit_vector(2 downto 0);
    OUT16 : out bit_vector(15 downto 0);
    OFFPOW : out std_logic;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    TRIG : out std_logic);
end UNIFY;
architecture UNIFY of UNIFY is
 component SCAN14567
  port (CLK2 : in std_logic ;
       POWER,RESET,INC,DEC : in std_logic ;
       OFFPOW : out std_logic ;
       T : out bit_vector(7 downto 0));
 end component;
 component SCANDIS
 port (CLK,CLK1  : in std_logic;
     OFFPOW : in std_logic ;
     KDATA,SDATA : in bit_vector(7 downto 0);
     DATASEV : out bit_vector(7 downto 0);
     COLSCAN : out bit_vector(2 downto 0));
 end component;
 component CLKSYN
 port (CLK : in std_logic ;
     RESET: in std_logic ;
     CLK1,CLK2,CLK3 : out std_logic);
 end component;
 component TBUFF
 port (CLK2 : in std_logic ;
     ADC : in bit_vector(7 downto 0);
     BUF : out bit_vector(7 downto 0));
 end component;
 component OBUFF
 port (CLK : in std_logic ;
     IND1: in bit_vector(7 downto 0);
     IND2: in bit_vector(2 downto 0);
     OUTD1 : out bit_vector(7 downto 0);
     OUTD2 : out bit_vector(2 downto 0));
 end component;
 component FUZZALG1N
 port ( CLK1: in std_logic ;
     STEMP,BTEMP :in bit_vector(7 downto 0);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
      KP : out bit_vector(4 downto 0);
      KI : out bit_vector(3 downto 0);
      ERTEMP : out bit_vector(8 downto 0));
  end component;
  component PICONTRN
  port ( CLK,CLK1: in std_logic;
      OFFPOW : in  std_logic;
      KP : in bit_vector(4 downto 0);
      KI : in bit_vector(3 downto 0);
      ERTEMP :in bit_vector(8 downto 0);
      OUT16 : out bit_vector(15 downto 0));
  end component;
  component BUFGP
   port (I : in std_logic ;
      O : out std_logic );
  end component;
  signal BU1,BU2,BU15 : bit_vector(7 downto 0);
  signal BU3 : bit_vector(8 downto 0);
  signal BU16 : bit_vector(2 downto 0);
  signal BU4 : bit_vector(4 downto 0);
  signal BU5 : bit_vector(3 downto 0);
  signal BU6 : std_logic;
  signal BU7,BU8,BU9,BU10,BU11,BU12,BU14 : std_logic;
begin
  TRIG  <= BU14;
  OFFPOW <= BU6;
  SCANKEY : SCAN14567
   port map(CLK2=>BU8,POWER=>POWER,RESET=>RESET,
       INC=>INC,DEC=>DEC,OFFPOW=>BU6,T=>BU1);
  SCANDI : SCANDIS
   port map(CLK => BU9,CLK1=>BU7,OFFPOW=>BU6,KDATA=>BU1,SDATA=>BU2,
       DATASEV=>BU15,COLSCAN=>BU16);
  CLKGEN : CLKSYN
   port map(CLK=>BU9,RESET=>RESET,CLK1=>BU10,CLK2=>BU11,CLK3=>BU12);
  ADCBUF : TBUFF
   port map(CLK2=>BU8,ADC=>ADC,BUF=>BU2);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

ขอขอบพระคุณ คุณพ่อ คุณแม่ อาจารย์
ที่ให้โอกาส แนวคิด แนวทาง
สร้างฝันให้เป็นจริง
พี่ๆ เพื่อนๆ ทุกคน ที่ให้กำลังใจในการทำงาน
และขอขอบคุณพี่ๆที่ทำงานที่เนคเทคทุกคน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
 OBUF : OBUFF
  port map(CLK=>BU9,IND1=>BU15,IND2 => BU16,OUTD1=>DATASEV,OUTD2 => COLSCAN);
 FUZZY : FUZZALG1N
  port map(CLK1=>BU7,STEMP=>BU1,BTEMP=>BU2,KP=>BU4,
        KI=>BU5,ERTEMP=>BU3);
 PICON : PICONTRN
  port map(CLK=>BU9,CLK1=>BU7,OFFPOW=>BU6,KP=>BU4,
        KI=>BU5,ERTEMP=>BU3,OUT16=>OUT16);
 GOL0  : BUFGP
  port map(I => CLK,O => BU9);
 GOL1  : BUFGP
  port map(I => BU10,O => BU7);
 GOL2  : BUFGP
  port map(I => BU11,O => BU8);
 GOL13  : BUFGP
  port map(I => BU12,O => BU14);
end UNIFY;

configuration UNIFY_CON of UNIFY is
 for UNIFY
  for SCANKEY : SCAN14567 use configuration work.KEY1_CON ;
  end for;
  for SCANDI : SCANDIS  use configuration work.DIS_CON ;
  end for;
  for CLKGEN : CLKSYN use configuration work.CLKS_CON ;
  end for;
  for ADCBUF : TBUFF use configuration work.BUFF_CON ;
  end for;
  for OBUF : OBUFF use configuration work.OBUFF_CON ;
  end for;
  for FUZZY : FUZZALG1N use configuration work.FUZZY1N_CON ;
  end for;
  for PICON : PICONTRN use configuration work.PIN_CON ;
  end for;
 end for;
end UNIFY_CON;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

ขอขอบพระคุณ คุณพ่อ คุณแม่ อาจารย์
ที่ให้โอกาส แนวคิด แนวทาง
สร้างฝันให้เป็นจริง
พี่ๆ เพื่อนๆ ทุกคน ที่ให้กำลังใจในการทำงาน
และขอขอบคุณพี่ๆที่ทำงานที่เนคเทคทุกคน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บรรณานุกรม

1. คาวีร์ ขันธะกาด และ ดำรงพล เลิศพิรุฬห์วงศ์ "การสังเคราะห์วงจรควบคุมอุณหภูมิ แบบฟัซซี่ลอจิกโดยวีเอชดีเอล" ปริญญานิพนธ์ ภาควิชาอิเล็กทรอนิกส์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง พ.ศ. 2539

2. ไพรสิทธิ์ ศรีวัฒนานุกูลกิจ และ วิบูลย์ ปิยวัฒนาเมชา "การนำฟัซซี่ลอจิกมาประยุกต์ใช้งานในเครื่องทำน้ำอุ่น" ปริญญานิพนธ์ ภาควิชาอิเล็กทรอนิกส์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง พ.ศ. 2536

3. Bo Yeon ,"Fuzzy logic : Basic concept and Application , 1996.

4. Mohammad Jamshida , Nader Vadiee and Timothy . Ross ,"Fuzzy Logic and Control Software snd Hardware Application", Prentice-Hall , 1993

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้