



ภาควิชาครุศาสตร์วิศวกรรม
คณะครุศาสตร์อุตสาหกรรม
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ใบรับรองปริญญานิพนธ์

ชื่อหัวข้อ ชุดฝึกดิจิทัลด้วย CPLDs
Digital Training Set with CPLDs

ชื่อนักศึกษา 1. นายสมโภช ธรรมโชติ รหัสประจำตัว 48035611
2. นายไพรัช ศรีโปฏก รหัสประจำตัว 48035615
3. นายกฤษณล ไม้ชู รหัสประจำตัว 48035619

หลักสูตร ครุศาสตร์อุตสาหกรรมบัณฑิต สาขาวิชา เทคโนโลยีการวัดคุมทางอุตสาหกรรม
อาจารย์ที่ปรึกษา อ.ปิยะ ศุภวาราสวัสดิ์
อาจารย์ที่ปรึกษาร่วม อ.พิชญ์สินี มะโน

คณะกรรมการสอบปริญญานิพนธ์	ลายมือชื่อ
1. อ.สุระชัย พิมพ์สาลี	
2. อ.ปิยะ ศุภวาราสวัสดิ์	
3. อ.ปิยะ จิตธรรมมาภิรมย์	
4. ผศ.สุทิน อาจหาญ	
5. อ.ไพบุลย์ พวงวงศ์ตระกูล	

วัน/เดือน/ปีที่สอบ วันวันศุกร์ที่ 4 เดือนพฤษภาคม พ.ศ. 2550 เวลา 11.00 น.

สถานที่สอบ ห้อง ค.310 คณะครุศาสตร์อุตสาหกรรม สจล.

ภาควิชารับรองแล้ว

ลงนาม.....

(รศ.สุรสิทธิ์ รัตรี)

หัวหน้าภาควิชาครุศาสตร์วิศวกรรม

วันที่ 30 เดือน พ.ค. พ.ศ. 50



<BT491522>

ชุดฝึกดิจิทัลด้วย CPLDs

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปริญญานิพนธ์

ชุดฝึกดิจิทัลด้วย CPLDs

DIGITAL TRAINING SET WITH CPLDs



ปพ.
๘ ๒๔๔๘
๙๕๔๙

เลขหมู่.....
เลขทะเบียน.....
วัน,เดือน,ปี.....

75174

24 ต.ค. 2550

b. 11816193
i.

ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรครุศาสตรบัณฑิต

สาขาวิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรม

ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานปีการศึกษา 2549 นั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปริญญานิพนธ์

เรื่อง ชุดฝึกดิจิทัลด้วย CPLDs
Digital Training Set with CPLDs

วัตถุประสงค์

1. เพื่อศึกษาหลักการออกแบบวงจรดิจิทัลด้วย CPLDs
2. เพื่อออกแบบชุดฝึกดิจิทัลด้วย CPLDs
3. เพื่อสร้างชุดฝึกดิจิทัลด้วย CPLDs เพื่อใช้ในการออกแบบวงจรดิจิทัล
4. เพื่อทดสอบการทำงานของชุดฝึกดิจิทัลด้วย CPLDs
5. เพื่อนำชุดฝึกดิจิทัลด้วย CPLDs ไปใช้ทำการทดลองและออกแบบวงจรดิจิทัล

ประโยชน์ที่คาดว่าจะได้รับ

1. ได้ความรู้จากการออกแบบวงจรดิจิทัลด้วย CPLDs ด้วยภาษา VHDL
2. ได้ต้นแบบของชุดฝึกดิจิทัลด้วย CPLDs
3. ได้ชุดฝึกดิจิทัลด้วย CPLDs
4. ได้ผลการทดลองของชุดฝึกดิจิทัลด้วย CPLDs
5. ได้ชุดฝึกดิจิทัลด้วย CPLDs ที่สามารถนำไปใช้งานได้จริง

ชื่อหัวข้อ	ชุดฝึกดิจิทัลด้วย CPLDs	
นักศึกษา	นายสมโภช	ธรรมโชติ
	นายไพรัช	ศรีโปฏก
	นายกฤษณล	ไผ่ชู
อาจารย์ที่ปรึกษา	อาจารย์ปิยะ	ศุภวาราสุวัฒน์
อาจารย์ที่ปรึกษาร่วม	อาจารย์พิชญ์สินี	มะโน
หลักสูตร	ครุศาสตร์อุตสาหกรรมบัณฑิต	
สาขาวิชา	เทคโนโลยีการวัดคุมทางอุตสาหกรรม	
ปีการศึกษา	2549	

บทคัดย่อ

ปรินญานินพนธ์ฉบับนี้นำเสนอชุดฝึกดิจิทัลด้วย CPLDs ซึ่งตัวชุดฝึกจะประกอบไปด้วยแผงวงจรอินเทอร์เฟสที่ครอบคลุมในการออกแบบวงจรดิจิทัล ได้แก่ วงจรแสดงผล วงจรควบคุม วงจรกำเนิดความถี่ และวงจรสวิตซ์เพื่อจำลองการทำงานระดับลอจิก และแผงวงจร CPLDs ได้ใช้ไอซีเบอร์ XC9572 เพื่อใช้ในการออกแบบวงจรดิจิทัล และเนื้อหาในงานจะใช้วิธีการออกแบบวงจรด้วยวิธีการเขียนภาษาอธิบายฮาร์ดแวร์ โดยจะใช้โปรแกรม Xilinx ISE 7.1i ในการออกแบบวงจรดิจิทัล

II

Thesis Title	Digital Training Set with CPLDs	
Students	Mr.Sompote	Tummachoti
	Mr.Pairat	Sripodok
	Mr.Kridsanon	Paichoo
Advisor	Mr.Piya	Supavarasuwat
Co - Advisor	Mrs.Pitsini	Mano
Education Level	Bachelor of Science in Industrial Education	
Program in	Industrial Instrument Technology	
Academic Year	2006	

ABSTRACT

This is Thesis Present Digital Training Set with CPLDs. This digital training has Board Interface inclusive design digital circuit. It's including Display circuit, control circuit, Frequency circuit and switch circuit for duplicate in the logic level. And Digital Training Set with CPLDs is use IC no. XC9572 for design digital circuit .This thesis will configure circuitry explanation about hardware by computer Software Xilinx ISE 7.1i in design Digital.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สามารถสำเร็จลุล่วงได้ด้วยดีนั้น เนื่องมาจากความร่วมมือร่วมใจของสมาชิกภายในกลุ่มทุกท่าน คณะผู้จัดทำต้องขอขอบคุณ อาจารย์ปิยะ ศุภวราสุวัฒน์ อาจารย์พิชญ์สินี มะโน อาจารย์สุรพงษ์ สิริพงศ์ดี และอาจารย์ประจำภาควิชาเศรษฐศาสตร์วิศวกรรมทุกท่านเป็นอย่างมากที่ได้กรุณาให้คำปรึกษาและให้คำแนะนำในการแก้ไขปัญหาต่างๆ ตลอดจนถึงข้อมูลและอุปกรณ์ที่เป็นประโยชน์ต่อการทดลองรวมถึงขั้นตอนต่างๆ ในการสร้างโครงงาน และในการจัดทำปริญญานิพนธ์ฉบับนี้ ขอขอบคุณห้องสมุดคณะครุศาสตร์อุตสาหกรรม ห้องสมุดคณะวิศวกรรมศาสตร์และสำนักหอสมุดกลางที่ช่วยอำนวยความสะดวกและเอื้อเฟื้อสถานที่ในการศึกษาและค้นคว้าหาข้อมูล

ขอกราบพระคุณบิดา มารดา และผู้มีพระคุณสำหรับที่กรุณาพวกเราที่ได้ให้การสนับสนุนทุกสิ่งทุกอย่างทางด้านการศึกษาดูมาจนถึงปัจจุบัน และสุดท้ายต้องขอขอบคุณเพื่อนๆ ที่เป็นกำลังใจให้พวกเราเสมอมา



สารบัญ

เรื่อง	หน้า
บทคัดย่อภาษาไทย	I
บทคัดย่อภาษาอังกฤษ	II
กิตติกรรมประกาศ	III
สารบัญ	IV
สารบัญตาราง	VII
สารบัญรูป	VIII
บทที่ 1 บทนำ	1
1.1 ความเป็นมาและความสำคัญ	1
1.2 จุดมุ่งหมายของโครงการ	1
1.3 สมมุติฐานของการจัดทำโครงการ	1
1.4 ขีดความสามารถของโครงการ	2
1.5 ขั้นตอนการทำโครงการ	2
1.6 เนื้อหาโดยสังเขป	2
บทที่ 2 ทฤษฎีและหลักการ	4
2.1 กล่าวนำ	4
2.2 ทำความรู้จักกับ FPGA	4
2.2.1 การโปรแกรมโดยการทำให้เกิดการเปลี่ยนแปลงทางกายภาพ	4
2.2.2 การโปรแกรมโดยใช้หน่วยความจำ	5
2.3 ปัจจัยที่ทำให้การออกแบบ FPGA ทำได้ง่ายและสะดวกรวดเร็ว	5
2.4 การออกแบบโดยใช้ภาษาอธิบายพฤติกรรมของฮาร์ดแวร์	6
2.4.1 การสังเคราะห์วงจร (Logic Synthesis)	6
2.4.2 การแบ่งวงจร (Partitioning)	7
2.4.3 การวางอุปกรณ์ (Placement)	7
2.4.4 การเชื่อมต่อสัญญาณ (Routing)	7
2.4.5 การหน่วงเวลา (Delay)	8
2.4.6 การจำลองการทำงานของวงจร (Simulation)	8
2.4.7 การโปรแกรมอุปกรณ์ FPGA (Configuration)	9
2.5 เครื่องมือสำหรับการออกแบบ FPGA	9

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

เรื่อง	หน้า
2.6 ภาษา VHDL	9
2.6.1 ความหมายของ VHDL	10
2.6.2 ที่มาของภาษา VHDL	10
2.7 โครงสร้างภาษา VHDL	11
2.7.1 ส่วนประกอบที่สำคัญของไฟล์ VHDL	11
2.7.2 แพกเกจในภาษา VHDL	11
2.7.3 โลบารีในภาษา VHDL	12
2.7.4 มาตรฐานของแพกเกจ	12
2.7.5 ชุดคำสั่ง USE	13
2.7.6 การเขียน Comments	13
2.7.7 การออกแบบ Entity	13
2.7.8 การเขียนโดยมีข้อมูลทางฟิลิกส์	15
2.7.9 การออกแบบส่วนที่ใช้เขียนบรรยายพฤติกรรมการทำงานของวงจร	15
2.7.10 การบรรยายพฤติกรรมแบบ Behavioural	17
2.7.11 การออกแบบการทำงานแบบลำดับขั้น	17
2.7.12 การออกแบบ Local Declaration	19
2.8 ประเภทของข้อมูล	19
2.8.1 ชนิดของข้อมูล (Type)	20
2.9 ข้อมูลประเภทอะเรย์ (Array)	23
2.10 รูปแบบคำสั่ง (Sequential)	24
2.10.1 การใช้คำสั่ง Process	24
2.10.2 ตัวอย่างการใช้งาน Process	24
2.10.3 การประมวลผลใน Process	25
2.10.4 กฎเกณฑ์การเขียน Process	26
2.10.5 การกำหนดตัวแปรของคำสั่ง Process	26
2.10.6 ชุดคำสั่งที่ใช้ใน Process	27
2.10.7 คำสั่ง If-Then-Else Statement	27
2.10.8 คำสั่ง Case Statement	28

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

เรื่อง	หน้า
2.10.9 คำสั่ง Loop Statement	28
2.10.10 คำสั่ง Wait Statement	30
บทที่ 3 การออกแบบ การสร้าง และการทำงาน	32
3.1 กล่าวนำ	32
3.2 คุณสมบัติทั่วไปของบอร์ดทดลองรุ่น CPLD Explorer XC9572XL	32
3.3 หลักการทำงานของบอร์ดทดลองรุ่น CPLD Explorer XC9572XL	32
3.4 การออกแบบใบงานของชุดฝึกดิจิทัลด้วย CPLDs	36
บทที่ 4 การทดลองและผลการทดลอง	38
4.1 กล่าวนำ	38
4.2 การทดลองการทำงานของแผงวงจร CPLDs	38
4.2.1 การทดลองติดต่อกับคอมพิวเตอร์	38
4.3 การทดลองชุดฝึกดิจิทัลด้วย CPLDs	38
บทที่ 5 บทสรุป	40
5.1 สรุป	41
5.2 ปัญหาและวิธีการแก้ไข	41
5.3 แนวทางการพัฒนา	41
บรรณานุกรม	42
ภาคผนวก ก เครื่องต้นแบบ	48
ภาคผนวก ข ใบงานการทดลอง	50
ภาคผนวก ค คู่มือการใช้งาน	154
ภาคผนวก ง แสดงรายละเอียดและคุณสมบัติของอุปกรณ์สำคัญที่ใช้ในโครงงาน	165
ประวัติผู้แต่ง	177

สารบัญตาราง

ตารางที่	หน้า
2.1 แพ็กเกจในภาษา VHDL	12
2.2 การกำหนดการใช้งาน Pre-defined Type	22
3.1 แสดงตำแหน่งขาของชิพที่ต่ออยู่กับฮาร์ดแวร์ภายนอกที่อยู่บนบอร์ด	35
3.1 (ต่อ) แสดงตำแหน่งขาของชิพที่ต่ออยู่กับฮาร์ดแวร์ภายนอกที่อยู่บนบอร์ด	36
4.1 ผลการทดลองโมดูล 7 Segment	43



สารบัญรูป

รูปที่	หน้า
2.1 การโปรแกรมลงในชิพ	5
2.2 การออกแบบด้วยภาษา VHDL	10
2.3 ส่วนประกอบที่สำคัญของไฟล์ VHDL	11
2.4 ตัวอย่างไลบรารี	12
2.5 ชุดคำสั่ง USE ใน Library	13
2.6 การเขียน Comments	13
2.7 รูปแบบการออกแบบ Entity	14
2.8 ตัวอย่างการเขียน Entity	14
2.9 การเขียนโดยมีข้อมูลทางฟิสิกส์	15
2.10 ตัวอย่างรูปแบบของภาษา VHDL	15
2.11 การบรรยายความสัมพันธ์ของข้อมูล	16
2.12 ตัวอย่างโครงสร้างที่ชื่อ BEHAVE	16
2.13 โมเดลของภาษา VHDL	16
2.14 ตัวอย่าง Behavioural Style	17
2.15 ตัวอย่าง การออกแบบวงจร Full Adder	18
2.16 การเขียน VHDL ในลักษณะของ Hierarchy (2)	18
2.17 การเขียน VHDL ในลักษณะของ Hierarchy (3)	19
2.18 Local Declarations	19
2.19 Signal ใน VHDL	20
2.20 แบ่ง Type ออกเป็นสองกลุ่มใหญ่ ๆ	20
2.22 การกำหนดชนิดข้อมูล	21
2.21 ตัวอย่างชนิดข้อมูลแบบ Standard Pre-defined	21
2.23 การใช้ Logic พื้นฐาน	22
2.24 ตัวอย่างข้อมูลต่างชนิดกัน	23
2.25 ตัวอย่างอะเรย์ std_ulogic	23
2.26 ตัวอย่างอะเรย์ std_l	23
2.27 ตัวอย่างอะเรย์ STRING หรือ BIT_VECTOR24	24
2.28 ตัวอย่างอะเรย์ โดยใช้คำสั่ง TYPE	24

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป (ต่อ)

รูปที่	หน้า
2.29 ตัวอย่างการใช้งาน Process	25
2.30 การประมวลผลใน Process (1)	25
2.31 การประมวลผลใน Process (2)	25
2.32 การประมวลผลใน Process	26
2.33 กฎเกณฑ์การเขียน Process	26
2.34 ตัวอย่างการใช้งาน Variable & Signal	27
2.35 ตัวอย่างคำสั่ง If-Then-Else Statement	27
2.36 ตัวอย่างกฎเกณฑ์ในการเขียนภาษา VHDL	28
2.37 ตัวอย่างการทำงาน Loop Statement	28
2.38 ตัวอย่างการทำงาน For-Loop Statement	28
2.39 ตัวอย่าง For-Loop	29
2.40 ตัวอย่างการทำงาน While Loop Statement	29
2.41 ตัวอย่าง While Loop	30
2.42 กฎเกณฑ์การเขียน Wait Statement	30
3.1 การจัดวางอุปกรณ์บน CPLD Explorer XC9572XL	33
3.2 วงจร Logic trainer และ Oscillator	33
4.1 โปรแกรมแอนเกด	38
4.2 การเข้าสู่ขั้นตอนการโปรแกรม	39
4.3 การค้นหาแผนวงจร	39
4.4 แสดงการผิดพลาดเมื่อหาแผนวงจรไม่พบ	40
4.5 แสดงผลว่าไม่พบแผนวงจร	40
4.6 การเลือกโปรแกรม	40
4.7 การบันทึกคำสั่ง	41
4.8 แสดงขั้นตอนการยืนยันการโหลดโปรแกรม	41
4.9 การประมวลผลการโหลดโปรแกรม	41
4.10 แสดงผลการโหลดเสร็จสมบูรณ์	42
4.11 โปรแกรม BCD 8421	43

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญ

เนื่องจากชุดฝึกดิจิทัลด้วย CPLDs ที่ภาควิชาวิศวกรรมอิเล็กทรอนิกส์ มหาวิทยาลัยเทคโนโลยีมหานครได้ออกแบบนั้นส่วนใหญ่จะออกแบบผังวงจรให้เป็นแบบอยู่ในผังวงจรเดียวกันและจำกัดขาเอาต์พุตทำให้ไม่สามารถออกแบบวงจรดิจิทัลที่มีความซับซ้อนได้เพราะขาเอาต์พุตที่เหลือนั้นมีไม่มากพอที่จะทำการทดลองวงจรดิจิทัลที่มีความซับซ้อนได้อย่างไม่เต็มที่และทางบริษัท ETT ได้ออกแบบชุดฝึกดิจิทัลด้วย CPLDs เป็นแบบแยกอิสระ คือ มีบอร์ดทดลองหลักและโมดูล CPLDs แต่ปัญหาที่พบนั้นตัวบอร์ดการทดลองหลักยังมีอุปกรณ์ที่ช่วยในการทดลองไม่มากพอที่จะทำการประยุกต์ในการออกแบบวงจรดิจิทัลที่มีความซับซ้อนและเนื้อหาในการทดลองหรือการออกแบบวงจรดิจิทัลไม่มีส่วนของการออกแบบการทดลองที่ใช้ภาษาอธิบายฮาร์ดแวร์ HDL (Hardware Description Language) ในการออกแบบวงจรดิจิทัลโดยจะเน้นไปทางวาดผังวงจร (Schematic) เพราะเหตุนี้จึงทำให้เกิดโครงการชุดฝึกดิจิทัลด้วย CPLDs โดยจะแยกออกเป็นสองบอร์ดโดยจะมีผังวงจรการทดลองหลักก็ คือ ผังวงจรที่มีอุปกรณ์ที่แสดงผลการทดลองทั้งอินพุตและเอาต์พุตโดยที่จะเพิ่มภาคการทดลองให้มากขึ้นเพื่อที่จะใช้ในการออกแบบวงจรดิจิทัลที่มีความซับซ้อนได้และอีกผังวงจร CPLDs ที่ใช้ เบอร์ XC9572 เพื่อใช้ในการออกแบบวงจรดิจิทัล

1.2 จุดมุ่งหมายของโครงการ

คณะผู้จัดทำได้สร้างชุดฝึกดิจิทัลด้วย CPLDs ขึ้นมาเพื่อให้ผู้ที่สนใจศึกษาการทำงานและการประยุกต์ใช้งานในการออกแบบวงจรดิจิทัลด้วย CPLDs ในด้านต่างๆ เช่น แอลอีดีแบบ 7 ส่วน 8 หลัก จอแสดงผลแบบสี่เหลี่ยม เซ็นเซอร์อุณหภูมิ รีเลย์ สแตมป์มอเตอร์ จะมีลักษณะเป็นไปตามใบงานการทดลองของชุดฝึกดิจิทัลด้วย CPLDs ผู้จัดทำได้ออกแบบการเรียนรู้ในแต่ละใบงานอย่างเป็นระบบ ซึ่งสามารถทำความเข้าใจได้ง่ายและเห็นถึงการนำไปใช้งานจริงโดยจะเรียงลำดับจากง่ายไปยาก

1.3 สมมุติฐานของการจัดทำโครงการ

เมื่อผ่านการเรียนและทำการทดลองตามใบงานในโครงการนี้แล้ว ผู้เรียนมีความรู้ความสามารถเกี่ยวกับการออกแบบวงจรดิจิทัลด้วยการเขียนภาษาอธิบายฮาร์ดแวร์ไปควบคุมอุปกรณ์ภายนอกแบบต่างๆ จนสามารถนำไปใช้งานจริงได้

1.4 ขีดความสามารถของโครงงาน

โครงงานนี้มีขีดความสามารถดังนี้

1. มีโมดูล 13 โมดูล เนื้อหาการทดลองแบบภาษาอธิบายฮาร์ดแวร์ (HDL) ดังนี้

- 1.1) การใช้งานโปรแกรม Xilinx ISE 7.1i
- 1.2) วงจรเกทพื้นฐาน
- 1.3) วงจรบวกเลข
- 1.4) วงจรลบเลข
- 1.5) วงจรเข้ารหัส
- 1.6) วงจรถอดรหัส
- 1.7) วงจรมัลติเพล็กซ์
- 1.8) วงจรดีมัลติเพล็กซ์
- 1.9) วงจรแปลงสัญญาณ BCD เป็น 7-Segment
- 1.10) วงจร ฟลิปฟลอป
- 1.11) วงจรนับ
- 1.12) วงจรเปรียบเทียบข้อมูล
- 1.13) วงจรเลื่อนข้อมูล

2. โมดูล CPLDs ใช้เบอร์ XC9572 ด้วยโปรแกรม Xilinx ISE 7.1i เพื่อทำการออกแบบวงจรดิจิทัลด้วยการใช้ภาษาอธิบายฮาร์ดแวร์ (VHDL)

1.5 ขั้นตอนการทำโครงงาน

โครงงานนี้ประกอบด้วยฮาร์ดแวร์และซอฟต์แวร์ ซึ่งการทำงานในระยะแรกจะเริ่มต้นจากการทำฮาร์ดแวร์ หลังจากนั้นเมื่อสร้างฮาร์ดแวร์ได้ระดับหนึ่งที่เหมาะสำหรับการเขียนโปรแกรมควบคุมได้ก็จะเริ่มเขียนโปรแกรมเพื่อทำการทดสอบ

1.6 เนื้อหาโดยสังเขป

เนื้อหาในปริิญญานิพนธ์ฉบับนี้แบ่งออกเป็นบทต่างๆ เพื่อสะดวกต่อการศึกษาและทำความเข้าใจในแต่ละบทจะประกอบด้วยเนื้อหาต่อไปนี้

บทที่ 1 กล่าวถึงความเป็นมาและความสำคัญของปริิญญานิพนธ์ขีดความสามารถของโครงงาน และเนื้อหาในบทต่างๆ โดยสังเขป

บทที่ 2 ประกอบด้วยทฤษฎีต่างๆ เกี่ยวกับอุปกรณ์ CPLDs และ FPGAs ด้วยภาษา VHDL ที่ใช้ในการออกแบบวงจรดิจิทัล

บทที่ 3 การออกแบบ การสร้าง และการทำงาน
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับอาจารย์ใช้ในการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4 การทดลองและผลการทดลองกล่าวถึงการทดลองที่ออกแบบด้วยการเขียนภาษาอธิบาย
ฮาร์ดแวร์ (VHDL)

บทที่ 5 เป็นการสรุปผลการจัดทำโครงงานปัญหาที่เกิดขึ้นและแนวทางแก้ไขรวมทั้งแนวทางการ
พัฒนา

ภาคผนวก ก เครื่องต้นแบบ

ภาคผนวก ข โปรแกรมการทดลอง

ภาคผนวก ค คู่มือการใช้งาน

ภาคผนวก ง แสดงรายละเอียดและคุณสมบัติของอุปกรณ์สำคัญที่ใช้ในโครงงาน



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2

ทฤษฎีและหลักการ

2.1 กล่าวนำ

ในช่วงก่อนทศวรรษ 1970 อุตสาหกรรมเซมิคอนดักเตอร์ได้ถูกปลุกให้ตื่นตัวขึ้นหลังจากที่มีการประดิษฐ์วงจรรวมหรือไอซีตัวแรกสำเร็จในยุคแรกนั้นไอซีขนาดเล็กหรือ SSI (Small-Scale Integration) ประกอบไปด้วยเกทดิจิทัลจำนวนไม่มากนัก (ประมาณ 1 ถึง 10 ตัว) ต่อมาได้มีการเพิ่มปริมาณของเกทดิจิทัลและฟังก์ชันทางลอจิกให้มากขึ้นจนเป็น MSI (Medium-Scale Integration) การพัฒนาไอซีเป็นไปอย่างต่อเนื่องจนมาถึงยุคของ LSI (Large-Scale Integration) ซึ่งเป็นยุคที่มีการสร้างไมโครโปรเซสเซอร์ตัวแรกขึ้นและในปัจจุบันเป็นยุคของ VLSI (Very Large-Scale Integration) ซึ่งเทคโนโลยีในการสร้างไอซีรุ่นหน้าจันสามารถสร้างไมโครโปรเซสเซอร์ขนาด 64 บิต ที่มีหน่วยความจำแคชกับหน่วยคำนวณทางคณิตศาสตร์ของฟลอยติงพอยท์ (Floating-Point Arithmetic Units) รวมอยู่ในตัวมันและเนื่องจากการปรับปรุงเทคโนโลยีของกระบวนการสร้างชิปที่มีมาอย่างต่อเนื่องทำให้ขนาดของทรานซิสเตอร์ที่บรรจุอยู่ในไอซีมีขนาดเล็กลงเรื่อยๆ จนบางคนโดยเฉพาะในญี่ปุ่นใช้คำว่า ULSI (Ultra Large Scale Integration) เพื่อใช้เรียกระดับของไอซีในปัจจุบันแต่คนส่วนมาก ยังมักนิยมเรียกเพียงแค่ VLSI จากการปรากฏตัวของ VLSI ในช่วงทศวรรษ 1980 ทำให้วิศวกรเริ่มมีการออกแบบไอซีตามความต้องการของลูกค้าซึ่งใช้ในระบบที่เจาะจงนอกเหนือจากการใช้ไอซีมาตรฐานเพียงอย่างเดียวโดยไอซีเหล่านี้มีชื่อเรียกว่า ASIC (Application-Specific Integrated Circuit) (ออกเสียงว่า เอ-ซิก) ซึ่งตัวอย่างของ ASIC ได้แก่ชิพไอซีที่ใช้สำหรับตุ๊กตาของเล่นพูดได้ ดาวเทียม และ ชิปที่ภายในบรรจุด้วยไมโครโปรเซสเซอร์กับอุปกรณ์ทางลอจิก

2.2 ทำความรู้จักกับ FPGA

ในปัจจุบันมี FPGA อยู่ 4 ชนิดที่วางขายอยู่ในท้องตลาดได้แก่ Symmetrical Array, Row-Based, Hierarchical PLD และ Sea-of-Gates ซึ่งแต่ละชนิดก็มีลักษณะการเชื่อมต่อภายในและการโปรแกรมที่แตกต่างกันไปนอกจากนี้ในการแบ่งประเภทของ FPGA อาจแบ่งได้ตามเทคโนโลยีที่ใช้ในการโปรแกรม ซึ่งมีอยู่ 2 แบบ คือ การโปรแกรมโดยการทำให้เกิดการเปลี่ยนแปลงทางกายภาพของตัวชิพ และการโปรแกรมโดยใช้หน่วยความจำ

2.2.1 การโปรแกรมโดยการทำให้เกิดการเปลี่ยนแปลงทางกายภาพ

Fuse เป็นวิธีการโปรแกรมที่สามารถทำได้เพียงครั้งเดียวซึ่งหลังจากที่โปรแกรมแล้วจุดเชื่อมต่อจะขาดจากกันส่วน Anti Fuse เป็นวิธีการโปรแกรมที่คล้ายกับแบบ Fuse แตกต่างกันที่หลังจากทำการโปรแกรมแล้วจุดเชื่อมต่อจะเชื่อมถึงกัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2.2 การโปรแกรมโดยใช้หน่วยความจำ

2.2.2.1 EEPROM Based FPGA

FPGA ที่ใช้การโปรแกรมแบบนี้มักเรียกว่า CPLD ซึ่งเทคโนโลยีที่ใช้จะเหมือนกับ EEPROM ทำให้มีความจุของเกตต่ำโดยทั่วไปจะน้อยกว่า 20,000 เกต แต่ข้อดีของ EEPROM Based FPGA คือสามารถเก็บข้อมูลที่โปรแกรมลงไปได้โดยไม่จำเป็นต้องมีไฟเลี้ยง และในการโปรแกรมจะใช้ทรานซิสเตอร์ 1 ตัวต่อ 1 บิต ซึ่งการโปรแกรมสามารถทำได้ประมาณ 10,000 ครั้ง

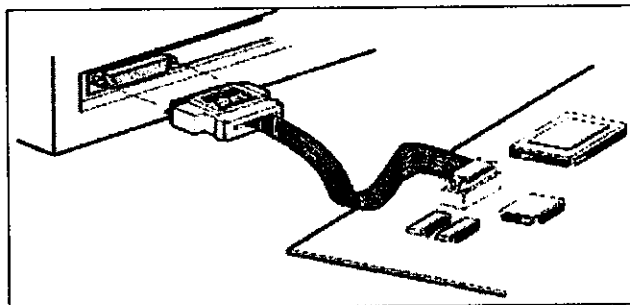
2.2.2.2 SRAM Based FPGA

FPGA แบบนี้จะใช้เทคโนโลยีในการโปรแกรมเหมือนกับ SRAM (Static RAM) ทำให้สามารถโปรแกรมซ้ำได้โดยไม่จำกัดจำนวนครั้งนอกจากนี้ยังมีความจุของเกตในระดับปานกลางถึงสูงมาก (ประมาณ 10,000 - 1,000,000 เกต) ซึ่งข้อดีของ SRAM Based FPGA คือ ใช้เวลาในการโปรแกรมน้อย (ระดับ nsec) การโปรแกรมทำได้ง่ายเทียบได้กับการเขียน SRAM ทั่วไป และเหมาะสำหรับการออกแบบวงจรที่มีความสลับซับซ้อน ส่วนข้อเสียคือไม่สามารถเก็บโปรแกรมในภาวะที่ไม่มีไฟเลี้ยงได้ ดังนั้น FPGA ชนิดนี้จึงมักใช้ควบคู่กับ ROM เพื่อเก็บโปรแกรมและทำการโหลดโปรแกรมลงในตัวชิปในขณะที่ยังมีไฟเลี้ยง

2.3 ปัจจัยที่ทำให้การออกแบบ FPGA ทำได้ง่ายและสะดวกรวดเร็ว

ผู้ออกแบบไม่จำเป็นต้องทราบถึงโครงสร้างภายในของตัวชิปเพียงแต่มีความรู้เกี่ยวกับขั้นตอนการออกแบบลอจิกก็เพียงพอแล้วต่างกับการใช้ไมโครโปรเซสเซอร์ซึ่งจำเป็นต้องศึกษาโครงสร้างภายในรวมถึงภาษา Assembly ของไมโครโปรเซสเซอร์ตัวนั้นด้วย

มีการออกแบบโดยใช้ภาษาในการอธิบายการทำงานของวงจรหรือ HDL (Hardware Description Language) เป็นเครื่องมือในการออกแบบซึ่งเป็นวิธีการที่มีความยืดหยุ่นสูงทำได้รวดเร็ว และไม่จำเป็นต้องทราบถึงลักษณะของวงจรที่ต้องการว่าจะเชื่อมต่อกันอย่างไรเพียงแต่กำหนดลักษณะการทำงานให้มันจากนั้นตัวซอฟต์แวร์จะทำ Synthesis and Optimize ให้ทั้งหมดนอกจากนี้ภาษาที่ใช้ยังเป็นมาตรฐานเดียวกันสามารถใช้ได้กับชิปทุกตัวและทุกบริษัท



รูปที่ 2.1 การโปรแกรมลงในชิป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การโปรแกรมสามารถทำได้เองและใช้เวลาไม่นานเพียงแค่ส่งข้อมูลผ่านสายดาวเทียมหรือทางพอร์ตของคอมพิวเตอร์ก็สามารถโปรแกรมตัวชิพขณะที่อยู่ในระบบได้โดยไม่ต้องถอดมาโปรแกรมข้างนอก ดังรูปที่ 2.1 และที่สำคัญสามารถโปรแกรมได้หลายครั้งจึงทำให้ง่ายในการแก้ไขและพัฒนาโดยไม่ต้องเสียค่าใช้จ่ายเพิ่มเติมอย่างใด

2.4 การออกแบบโดยใช้ภาษาอธิบายพฤติกรรมของฮาร์ดแวร์

ในการออกแบบวงจรดิจิทัลนั้นสามารถทำได้โดยการวาดวงจร (Schematic) หรือใช้ภาษาอธิบายพฤติกรรม (Hardware Description Language) ของฮาร์ดแวร์จากที่ได้กล่าวไปแล้วในบทที่ 1 ในกรณีของการออกแบบวงจรด้วย ASIC ชนิด Full Custom ผู้ออกแบบจะต้องเขียนวงจรด้วย Schematic จากนั้นจะนำวงจรที่ออกแบบไว้ไปทำการจำลองการทำงาน (Simulate) ซึ่งหากผลออกมาเป็นที่พอใจก็จะต้อง Layout เป็นชั้นสาร และในการออกแบบ ASIC ชนิดนี้ผู้ออกแบบจำเป็นจะต้องทราบถึงเทคโนโลยีที่ใช้ในการสร้างด้วย หลังจากได้ Layout ที่สมบูรณ์แล้วจึงจะส่งไปเข้ากระบวนการสร้างไอซีหรือ Fabrication เพื่อสร้างเป็นชิพ ไอซีออกมา แต่ในการออกแบบวงจรด้วย FPGA โดยการใช้ Schematic หรือใช้ภาษาอธิบายการทำงานของวงจรจะทำได้สะดวกกว่าเนื่องจากวิธีการนี้ผู้ออกแบบไม่จำเป็นต้องคำนึงถึงเทคโนโลยีที่จะใช้สร้างไอซีและที่สำคัญการออกแบบโดยวิธีนี้สามารถแก้ไขโมเดล (Model) หรือเปลี่ยนแปลงเทคโนโลยีได้สะดวกกว่าเพราะไม่ต้องวาดวงจรใหม่นั้นคือการออกแบบโดยใช้ภาษาอธิบายฮาร์ดแวร์จะทำให้โมเดลที่ได้ไม่ขึ้นกับเทคโนโลยีสำหรับภาษาที่ใช้ สำหรับอธิบายพฤติกรรมของฮาร์ดแวร์ที่ใช้กันก็มี VHDL, AHDL และ Verilog เป็นต้น ส่วนรายละเอียดของขั้นตอนในการออกแบบสามารถอธิบายได้ดังนี้

2.4.1 การสังเคราะห์วงจร (Logic Synthesis)

ในขั้นตอนนี้จะใช้ซอฟต์แวร์ในการสังเคราะห์วงจร (Synthesis Tools) ทำการสังเคราะห์พฤติกรรมของวงจรที่ได้จากการออกแบบด้วย Schematic หรือ VHDL ซึ่งต้องทำการตรวจสอบด้วยว่าซอฟต์แวร์นั้นสนับสนุนเทคโนโลยี FPGA (FPGA Library) ที่ต้องการหรือไม่ ตัวอย่าง เช่น FPGA ของบริษัท XILINX และบริษัท ALTERA จะมีซอฟต์แวร์หลายตัวที่สามารถใช้ได้ เช่น Max Plus II ในขั้นตอนนี้ซอฟต์แวร์สังเคราะห์วงจรจะทำการแปลงโค้ด VHDL และทำการ Optimize เพื่อให้ได้วงจรตามเทคโนโลยีที่เลือกใช้ในการสังเคราะห์วงจรนั้นวงจรระดับเกต (Gate Level) จะไม่เหมาะสมกับโครงสร้างที่มีอยู่ในอุปกรณ์ FPGA ดังนั้นในการ Optimize ซอฟต์แวร์สังเคราะห์วงจรจะต้องทำการ Optimize ให้ได้เป็นวงจรที่ประกอบด้วยกลุ่มของลอจิกที่เหมาะสมกับอุปกรณ์ FPGA นั่นๆจึงทำให้ผลที่ได้มีประสิทธิภาพและในขั้นตอนการสังเคราะห์วงจรนี้ ผู้ออกแบบสามารถกำหนดข้อบังคับสำหรับโมเดลแต่ละตัวได้ เช่น ข้อบังคับในเรื่องเวลา (Timing Constraints) หรือข้อบังคับในเรื่องของพื้นที่ (Area) หรือกำหนดชนิดและตำแหน่งของ I/O ซึ่งข้อบังคับเหล่านี้จะถูกนำไปใช้ในขั้นตอน Optimize เพื่อให้วงจรที่ได้เป็นไปตามที่กำหนดส่วนสำคัญในการ Optimize คือการเทียบ (Mapping) โมเดลให้เข้ากับเทคโนโลยีที่ใช้เพื่อให้ได้วงจรที่เหมาะสมกับโครงสร้าง

และสถาปัตยกรรมภายในอุปกรณ์ FPGA เมื่อทำการสังเคราะห์วงจรเสร็จแล้วซอฟต์แวร์การสังเคราะห์วงจรก็จะมีรายงานผลว่าโมเดลที่ออกแบบไปนั้นเป็นอย่างไร เช่น มีค่าความหน่วง (Delay) เท่าใดใช้ทรัพยากรต่างๆ ใน FPGA อะไรบ้างเมื่อมาถึงขั้นตอนนี้ผู้ออกแบบก็จะทราบว่าโมเดลเป็นไปตามข้อบังคับหรือไม่ถ้าไม่ก็สังเคราะห์ใหม่จนกว่าจะเป็นไปตามที่กำหนด

2.4.2 การแบ่งวงจร (Partitioning)

ขั้นตอนนี้เป็นการแบ่งวงจรที่ได้จากการสังเคราะห์ เป็นส่วนย่อยๆ สำหรับลงใน CLB, IOBs หรือองค์ประกอบอื่นๆ ภายในอุปกรณ์ FPGA สำหรับเทคนิคที่ใช้ในการแบ่งคือให้แต่ละส่วนที่จะแยกออกจากกันมีจำนวนสัญญาณที่เชื่อมต่อระหว่างกันน้อยที่สุดเท่าที่จะทำได้เพื่อลดความหนาแน่นในขั้นตอนการเชื่อมต่อสัญญาณ (routing) ในขั้นตอนนี้จะใช้ซอฟต์แวร์ทำโดยซอฟต์แวร์จะเทียบส่วนประกอบของวงจร เช่น เกต (gate), ฟลิป-ฟลอป (flip-flop) ลงในทรัพยากรต่างๆ ที่มีอยู่ในอุปกรณ์ FPGA หลังจากทำขั้นตอนนี้เสร็จแล้วผู้ออกแบบสามารถที่จะทราบว่าวงจรใช้จำนวนทรัพยากรภายในอุปกรณ์ FPGA ไปเท่าไร ส่วนข้อมูลทางเวลานั้นผู้ออกแบบจะทราบเฉพาะความหน่วงภายในแต่ละส่วนเท่านั้น หรือที่เรียกว่าความหน่วงลอจิก (logic delay) ส่วนซอฟต์แวร์จะรวมเอาซอฟต์แวร์ย่อยอื่นๆ อีกเพื่อให้การทำ PPR (Partitioning Placement & Routing) เป็นไปอย่างต่อเนื่อง

2.4.3 การวางอุปกรณ์ (Placement)

ขั้นตอนนี้เป็นการเลือกทำเลที่ตั้งของแต่ละส่วนของวงจรที่ผ่านการแบ่งวงจร (Partitioning) มาแล้ว ว่าควร จะอยู่ ณ ตำแหน่งไหนในอุปกรณ์ FPGA เพื่อให้ได้ผลลัพธ์ที่ดีที่สุด เช่น วงจรส่วนไหนควรอยู่ใกล้กันเพื่อจะได้ค้นหาเส้นทางได้ (route) ง่ายหรือช่วยลดความหน่วงจะเห็นได้ว่าตำแหน่งภายในอุปกรณ์ FPGA นั้นมีความสำคัญเพราะถ้าจัดวางวงจรลงในตำแหน่งที่ไม่เหมาะสมแล้วจะทำให้ความหน่วงเพิ่มขึ้นหรือ Router ทำการค้นหาเส้นทางสัญญาณซับซ้อนไม่หมดการวางอุปกรณ์ที่ดีควรวางส่วนต่างๆ ให้อยู่ใกล้กัน โดยเฉพาะส่วนที่มีการเชื่อมต่อสัญญาณด้วยกันนอกจากนั้นการกำหนดตำแหน่งขา I/O (I/O pin) ตามตำแหน่งขา I/O ของ FPGA บนแผ่น PCB ก็จะมีผลโดยตรงเลยคือซอฟต์แวร์จะวาง I/O ลงในตำแหน่งที่ผู้ออกแบบกำหนดซึ่งบางครั้งตำแหน่งที่กำหนดไปไม่เหมาะสม ดังนั้นการกำหนดขา I/O ควรกำหนดตำแหน่งให้เหมาะสมหรือไม่ก็ให้ซอฟต์แวร์จัดการเอง

2.4.4 การเชื่อมต่อสัญญาณ (Routing)

ในขั้นตอนนี้เป็นการเชื่อมต่อสัญญาณระหว่างองค์ประกอบต่างๆ ภายในอุปกรณ์ FPGA ขั้นตอนนี้จะทำต่อเนื่องจากการวางอุปกรณ์ในกรณีที่ทำการวางอุปกรณ์ไว้ไม่ดีซอฟต์แวร์ก็จะทำการเชื่อมต่อสัญญาณได้ไม่หมด (เนื่องจากจำนวนทรัพยากรสำหรับเชื่อมต่อสัญญาณนั้นมีอยู่จำกัด) หรือเกิดความหน่วงเกินค่าที่กำหนดในข้อบังคับผู้ออกแบบสามารถทำขั้นตอนนี้ได้โดยใช้ซอฟต์แวร์หรือผู้ออกแบบจะทำการเชื่อมต่อสัญญาณด้วยตนเองก็ได้ แต่ทางที่ดีควรใช้ซอฟต์แวร์ทำดีกว่านอกจากนั้นการกำหนดข้อบังคับทางเวลาจะช่วยให้ผลที่ได้จากการเชื่อมต่อสัญญาณดีขึ้นได้

2.4.5 การหน่วงเวลา (Delay)

ในการทำ FPGA นั้นความหน่วงที่เกิดขึ้นเป็นความหน่วงที่เกิดจากการวางตำแหน่ง (layout) อุปกรณ์ซึ่งผู้ออกแบบไม่สามารถเข้าไปแก้ไขได้ แต่สามารถทำให้มีความหน่วงน้อยที่สุดได้สำหรับความหน่วงที่เกิดขึ้นนั้นแยกได้เป็นสองประเภทคือความหน่วงลอจิก (Logic delay) เป็นความหน่วงภายในขององค์ประกอบของอุปกรณ์ FPGA เองความหน่วงที่เกิดจากการเชื่อมต่อสัญญาณ (Routing delay) เป็นความหน่วงที่เกิดจากการเชื่อมต่อสัญญาณระหว่างองค์ประกอบภายในอุปกรณ์ FPGA โดยปกติแล้วค่าความหน่วงลอจิกไม่ควรเกิน 50% ของค่าความหน่วงที่ยอมรับได้เพราะความหน่วงที่เกิดจากการเชื่อมต่อสัญญาณมักจะมีค่ามากกว่าค่าความหน่วงลอจิก ดังนั้นในการวางอุปกรณ์ และเชื่อมต่อสัญญาณ ผู้ออกแบบควรกำหนดข้อบังคับทางเวลาเพื่อให้ซอฟต์แวร์ได้ทำงานอย่างมีประสิทธิภาพเพิ่มขึ้น และเพื่อให้ได้ผลลัพธ์ที่ดีขึ้นค่าความหน่วงที่ได้หลังจากการวางอุปกรณ์ และเชื่อมต่อสัญญาณแล้วจะมีค่าความหน่วงที่ค่อนข้างแน่นอนซึ่งผู้ออกแบบสามารถทราบได้ว่าโมเดลที่ออกแบบนั้นเป็นไปตามข้อกำหนดหรือไม่

2.4.6 การจำลองการทำงานของวงจร (Simulation)

ในขั้นตอนนี้เป็นขั้นตอนที่สำคัญอีกขั้นตอนหนึ่งเพราะเป็นขั้นตอนที่ผู้ออกแบบตรวจสอบฟังก์ชันการทำงานของโมเดลว่าถูกต้องหรือไม่ มีข้อผิดพลาดตรงไหนเพื่อจะได้ทำการแก้ไขให้ถูกต้องในขั้นตอนนี้จะมีซอฟต์แวร์ที่ใช้สำหรับการจำลองการทำงานของวงจรที่ใช้อยู่ เช่น Model Sim ของบริษัท Model Technology หรือ Max Plus II ของบริษัท Altera ในการจำลองการทำงานของวงจรควรทำทุกครั้งหลังจากที่มีการทำแต่ละขั้นตอนหลักเสร็จแล้วเพื่อจะได้ทราบว่าข้อผิดพลาดของโมเดลเกิดขึ้นตอนไหนจะได้แก้ไขข้อผิดพลาดตรงขั้นตอนนั้นๆ ได้เลยไม่ต้องมาคอยตรวจหาขั้นตอนที่ทำให้เกิดข้อผิดพลาดนั้น คือ การทำการจำลองการทำงานของวงจรต้องทำทั้งหลังการเขียนโค้ด, การสังเคราะห์วงจร และการทำ PPR การจำลองการทำงานของวงจรหลังจากที่เขียนโค้ดเสร็จแล้วนั้นผู้ออกแบบสามารถทราบได้แค่โมเดลทำงานถูกต้องหรือไม่เท่านั้น (functional test) ยังไม่สามารถตรวจสอบการทำงานในเชิงเวลาได้ถูกต้องในการจำลองการทำงานของวงจรหลังจากที่สังเคราะห์เป็นวงจรแล้วเพื่อตรวจสอบว่าฟังก์ชันการทำงานยังคงถูกต้องหรือไม่ และค่าความหน่วงที่เกิดขึ้นเป็นไปตามข้อบังคับหรือไม่ มีข้อผิดพลาดเกิดขึ้นหรือไม่ถ้ามีจะแก้ไขให้ถูกต้องในการจำลองการทำงานของวงจรหลังจากที่ทำการวางอุปกรณ์การเชื่อมต่อสัญญาณ (post layout simulation) แล้วก็มีความสำคัญเช่นกันเพราะผลที่ได้จากการจำลองการทำงานของวงจรในตอนนี้จะเป็นผลลัพธ์ของโมเดลเลยซึ่งผู้ออกแบบนอกจากจะตรวจสอบฟังก์ชันการทำงานแล้วยังต้องตรวจสอบคุณสมบัติอื่นๆ เช่น ความหน่วงที่ได้จากการทำ PPR ในรูปแบบค่าความหน่วงมาตรฐาน (Standard Delay Format : SDF) ว่าตรงตามข้อกำหนดหรือไม่หรือตรวจสอบว่าวงจรรวมสามารถใช้งานที่ความถี่สูงสุดเท่าไรนั่นเอง

2.4.7 การโปรแกรมอุปกรณ์ FPGA (Configuration)

หลังจากที่โมเดลผ่านขั้นตอนต่างๆ จนกระทั่งผ่านการทำ PPR (Partitioning, Placement & Routing) แล้วนั้นถึงตอนนี้ก็สามารถที่จะดาวน์โหลด (download) ลงในอุปกรณ์ FPGA ได้แล้วในการ

ดาวนโหลดนี้ก่อนอื่นต้องแปลงแบบวงจรรวมที่ได้เป็นข้อมูลวงจร (configuration data) ซึ่งอยู่ในรูปของบิตสตรีม (bit stream) ก่อนแล้วจึงดาวนโหลดลงไปเพื่อให้อุปกรณ์ FPGA มีฟังก์ชันการทำงานตามโมเดลที่ผู้ออกแบบต้องการซึ่งในขั้นตอนนี้จะใช้วิธีที่แตกต่างกันออกไปสำหรับอุปกรณ์ FPGA ของแต่ละบริษัทผู้ผลิต คือ ในกรณีที่อุปกรณ์ FPGA ชนิดที่ต้องโปรแกรมโดยวิธี SRAM นั้นในการใช้งานผู้ออกแบบจะต้องเก็บข้อมูลวงจรไว้ในหน่วยความจำประเภท EPROM หรือ serial PROM ด้วยเพื่อจะใช้งานสะดวกขึ้น คือ ในการใช้งานโมเดลครั้งต่อไปไม่ต้องดาวนโหลดข้อมูลวงจรจากเครื่องคอมพิวเตอร์อีกเพราะมีข้อมูลวงจรเก็บอยู่ในหน่วยความจำอยู่แล้วแต่กรณีที่อุปกรณ์ FPGA เป็นชนิดที่โปรแกรมโดยวิธี EPROM หรือ Anti fuse ก็ไม่จำเป็นต้องมีหน่วยความจำสำหรับเก็บข้อมูลวงจรเพราะว่าอุปกรณ์ FPGA ชนิดนี้เมื่อดาวนโหลดข้อมูลวงจรลงไปแล้วข้อมูลก็ดาวนโหลดลงไปก็ยังคงอยู่ในอุปกรณ์ FPGA และครั้งต่อไปก็ใช้งานโมเดลที่ออกแบบไว้ได้เลย

2.5 เครื่องมือสำหรับการออกแบบ FPGA

จะเห็นได้ว่าการออกแบบเพื่อทำ FPGA นั้นทำได้สะดวกกว่า ASIC มากเพราะใช้เวลาน้อยกว่ามากด้วยส่วนสำคัญที่ใช้ในการทำ FPGA คือ ซอฟต์แวร์ที่ใช้ตั้งแต่เขียนโค้ดอธิบายฮาร์ดแวร์จนกระทั่งดาวนโหลดลงในอุปกรณ์ FPGA ซึ่งซอฟต์แวร์ที่ใช้ต้องเป็นซอฟต์แวร์ที่ทำงานต่อเนื่องกันได้สำหรับซอฟต์แวร์ที่ใช้ทำการจำลองการทำงานของวงจรมันต้องสามารถใช้งานต่อเนื่องกับซอฟต์แวร์ที่ใช้ทั้งระบบเพราะโมเดลที่ได้จากการทำขั้นตอนต่างๆ (ด้วยซอฟต์แวร์ ต่างๆ ต้องเอามาจำลองการทำงานได้และในการจำลองการทำงานของวงจรใช้ซอฟต์แวร์ตัวเดียวกันตลอดทั้งระบบเพื่อจะได้เปรียบเทียบผลได้ง่ายในอดีตซอฟต์แวร์ส่วนใหญ่จะใช้งานอยู่บนคอมพิวเตอร์สมรรถนะสูงอย่างเวิร์กสเตชัน (Workstation) ในปัจจุบันมีการพัฒนาซอฟต์แวร์ที่ใช้บนพีซี (PC) มากขึ้นซึ่งสามารถลดค่าใช้จ่ายในด้านอุปกรณ์คอมพิวเตอร์

2.6 ภาษา VHDL

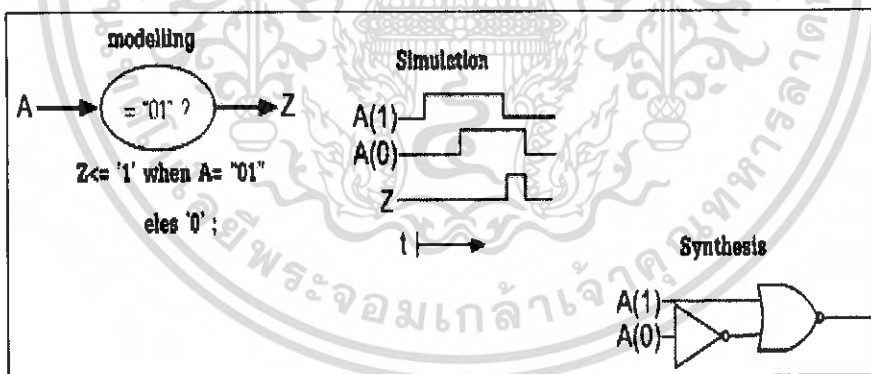
จากแนวความคิดในการออกแบบภาษาคอมพิวเตอร์ที่เป็นโครงสร้างนั้นทำให้เราสามารถออกแบบระบบที่มีความซับซ้อนได้ดี ดังนั้นจึงได้มีการสร้างภาษาขึ้นมาสำหรับบรรยายพฤติกรรมของฮาร์ดแวร์ (Hardware) เพื่อให้การออกแบบฮาร์ดแวร์เป็นไปอย่างมีระบบและสะดวกมากยิ่งขึ้นจนกระทั่งเมื่อกว่าทศวรรษมานี้ได้มีการพยายามสร้างมาตรฐานของภาษาบรรยายฮาร์ดแวร์ขึ้นมาเพื่อให้เป็นมาตรฐานอย่างเช่น Fortran, Pascal หรือ C เป็นต้นและในปี 1983 U.S. Department of Defense Sponsored ได้พัฒนาภาษาบรรยายความเร็วสูงมาก (Very High Speed Integrated Circuit Hardware Description Language : VHDL) ซึ่งต่อมารู้จักกันในนามของ VHDL แต่เนื่องจากการออกแบบโดยทั่วไปพบว่าการออกแบบด้วยภาษาดังกล่าวต้องรับอิทธิพลหลายๆ อย่างด้วยกันและแต่ละอย่างก็มีชนิดแตกต่างกันด้วยเหตุผลนี้ต่อมาได้มีการตกลงร่วมกันที่จะกำหนดคุณลักษณะของภาษาบรรยายขึ้นมาใหม่โดยในเดือน สิงหาคม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปี ค.ศ. 1985 ได้พัฒนาภาษาได้สมบูรณ์เป็นครั้งแรก (version 7.2) ในขณะนั้นต่อสมาคม IEEE เป็นผู้ส่งเสริมและผลักดันให้มีการพัฒนาภาษา VHDL และผลสุดท้ายได้พัฒนาปรับปรุงแก้ไขรูปแบบมาตรฐานของภาษาอย่างจริงจังและเสร็จสมบูรณ์ในเดือนพฤษภาคม ปี ค.ศ.1987 พร้อมทั้งทำคู่มืออ้างอิงภาษา (LRM : Language-Reference Manual) สำหรับให้อุตสาหกรรมตรวจสอบอีกครั้ง และในเดือน มิถุนายน ค.ศ. 1987 สมาชิกของ IEEE เห็นว่ามาตรฐานของภาษา VHDL ในเวอร์ชันนี้มีคุณสมบัติเหมาะสมและยอมรับให้เป็นเวอร์ชันที่เป็นมาตรฐานของ VHDL และในเดือน ธันวาคม ปี ค.ศ. 1987 สมาคม IEEE ได้นำมาตรฐานมาใช้ออกเป็นทางการและในช่วง 1988 ถึง 1991 ผู้ใช้ VHDL ได้เสนอแนะให้เปลี่ยนบางส่วนของภาษาและได้เปลี่ยนตามวัตถุประสงค์ดังกล่าวในปี ค.ศ. 1992 และผู้ใช้งานกลุ่มก็ได้ก่อตั้งเป็นบริษัทเกี่ยวข้องกับภาษานี้ขึ้นมาหลายบริษัทด้วยกัน

2.6.1 ความหมายของ VHDL

VHDL ย่อมาจาก VHSIC: Very High Speed IC: Hardware Description Language เริ่มต้นจากกระทรวงกลาโหมสหรัฐอเมริกา (US: DOD) ประมาณปี ค.ศ. 1981 ทำการพัฒนาโครงการที่มีชื่อว่า VHSIC ทำการพัฒนาโปรแกรมซึ่งจะจัดเป็นภาษาระดับสูงเช่นเดียวกับภาษา C หรือ PASCAL แต่สามารถบรรยายพฤติกรรมของ วงจรดิจิทัล หรือโครงสร้างของวงจรได้ ทั้งนี้เพื่อให้สามารถออกแบบ และสร้างวงจรดิจิทัลได้เร็วขึ้น



รูปที่ 2.2 การออกแบบด้วยภาษา VHDL

2.6.2 ที่มาของภาษา VHDL

เริ่มต้น โดย US: DOD ได้มอบหมายให้บริษัท IBM, Texas Instruments และ Intermarries เป็นผู้ศึกษาและพัฒนาภาษาที่ใช้ออกแบบวงจรดิจิทัลและได้ผลเป็นที่น่าพอใจ จนกระทั่งปี ค.ศ. 1985 ทาง ITAR ได้ยกเลิกข้อจำกัดในการถ่ายทอดเทคโนโลยีทางการทหาร ออกจากโครงการนี้ ดังนั้น VHDL จึงเริ่มรู้จักกันโดยทั่วไป จนกระทั่งทาง IEEE (Institute of Electrical and Electronic Engineering) ได้รับภาษานี้เข้ามาศึกษาและประมาณปี ค.ศ.1987 ได้ยอมรับกำหนดมาตรฐานของภาษา โดยให้ชื่อว่า IEEE 1076-1987 เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น เมื่ออนุญาตให้ไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

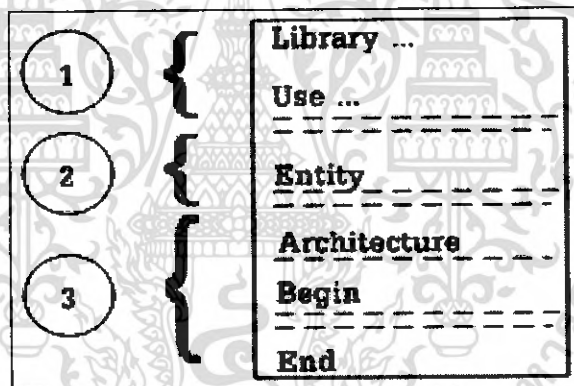
1976-1987 และมีชื่อเรียกว่า VHDL มาตรฐานนี้ได้รับการปรับปรุงจนปัจจุบันได้ชื่อว่า IEEE 1076-1993 หรือ VHDL 1993

2.6.3 ประโยชน์ของ VHDL

เป็นภาษาที่นำไปเขียนรูปแบบระบบดิจิทัล และมีคุณสมบัติที่คนและเครื่องสามารถเข้าใจได้ โดยไม่ต้องแปลอีก

1. เป็นภาษาที่สามารถเขียนขึ้นเพื่อจำลองการทำงานได้
2. สามารถนำไปเป็นเอกสารประกอบโครงการได้ในรายละเอียดต่อนี้จะเน้นถึงการใช้ภาษา VHDL ในการสังเคราะห์วงจร โดยเขียนเป็นรูปแบบภาษาที่สามารถอธิบายพฤติกรรม หรือ โครงสร้างของระบบดิจิทัล

2.7 โครงสร้างภาษา VHDL



รูปที่ 2.3 ส่วนประกอบที่สำคัญของไฟล์ VHDL

2.7.1 ส่วนประกอบที่สำคัญของไฟล์ VHDL

1. ส่วนที่แสดงถึง Library และ Package ที่จะเรียกใช้
2. Entity สำหรับกำหนดพอร์ตของวงจร ซึ่งเป็นได้ทั้ง in put, out put หรือ in out
3. Architecture สำหรับอธิบายการทำงานในลักษณะของทั้ง Behavioral Level หรือ Structural Level

Level

2.7.2 แพคเกจในภาษา VHDL

Package ข้อมูลต่างๆ ตลอดจนโปรแกรมย่อย (Subprogram) ที่เป็นประโยชน์ต่อการเขียนรูปแบบบรรยายระบบดิจิทัลสามารถเก็บไว้ในส่วนที่เรียกว่า Package ได้ และข้อมูลเหล่านี้สามารถถูกเรียกไปใช้ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โดย Entity design unit, Architecture design unit หรือจาก Package design unit อื่น ด้วยคำสั่ง USE statement

2.7.3 โลยารีในภาษา VHDL

Library ทำหน้าที่เก็บ Package ต่างๆ ไว้ ซึ่งใน Package จะเป็นทรัพยากรต่างๆ ที่ใช้งานร่วมกัน ดังนั้นก่อนใช้งานจะต้องกำหนดไว้ในตอนต้นของ VHDL ไฟล์ไว้ด้วย

ชุดคำสั่ง Library ดังในตัวอย่าง เป็นชุดคำสั่งทำให้ Library ที่มีชื่อสัญลักษณ์ว่า IEEE สามารถนำมาใช้ในส่วนที่เป็น entity ชื่อ test ได้ นั่นคือ library ชื่อ IEEE จะมองเห็นได้สำหรับ entity ชื่อ test

```
Library IEEE;
USE IEEE.std_logic_1164.and;
entity test is
```

รูปที่ 2.4 ตัวอย่างไลบรารี

2.7.4 มาตรฐานของแพ็กเกจ

จากตารางที่ 2.1 เป็นการบอกถึงลักษณะของแพ็กเกจมาตรฐานในการเขียนภาษา VHDL

ตารางที่ 2.1 แพ็กเกจในภาษา VHDL

IEEE	VHDL'87/VHDL'93	SYNOPSYS proprietary
std_logic_1164.vhd (ieee)	standard.vhd (std)	std_logic_arith.vhd (ieee)
numeric_std.vhd (ieee)	textio.vhd (std)	std_logic_misc.vhd (ieee)
numeric_bit.vhd (ieee)		std_logic_signed.vhd (ieee)
		std_logic_unsigned.vhd (ieee)
		std_logic_textio.vhd (ieee)
		attributes.vhd (synopsys)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.7.5 ชุดคำสั่ง USE

ชุดคำสั่ง USE ใน Library (กำหนดชื่อสัญลักษณ์) อาจจะประกอบด้วย package หลายๆ ชุด ดังนั้นจะต้องกำหนด package เฉพาะที่ต้องการใช้งาน ด้วยคำสั่ง USE ซึ่งมีรูปแบบดังรูป 2.5

```
USE library_name.package_name.element_of_package;  
  
Library IEEE;  
entity test is  
  
end test ;
```

รูปที่ 2.5 ชุดคำสั่ง USE ใน Library

2.7.6 การเขียน Comments

Comments คือการอธิบายความหมายของโปรแกรมในแต่ละบรรทัด

```
-----  
----- - this is a comments  
----- - each line must begin with -----  
----- - comment end with a new line  
-----  
Library IEEE;  
USE IEEE.std_logic_1164.all;  
Entity test is  
-----
```

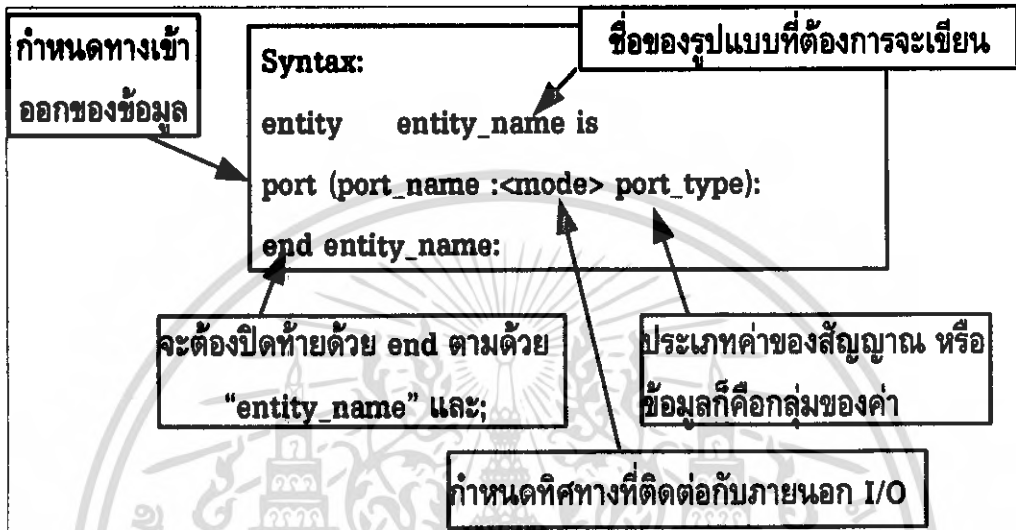
รูปที่ 2.6 การเขียน Comments

2.7.7 การออกแบบ Entity

Entity ใน VHDL จะบอกถึงพอร์ตที่จะใช้เชื่อมต่อกับส่วนอื่นๆ ของวงจรภายในวงเล็บโดยจะยังไม่มีการพูดถึงพฤติกรรมของวงจรเป็นส่วนที่ใช้สำหรับติดต่อระหว่างโลกภายนอกกับรูปแบบ (model) ที่จะเขียนขึ้นเป็นส่วนที่ใช้กำหนดจุดต่อ (connection Point) ของรูปแบบกำหนดทิศทางการไหลของสัญญาณ (mode)

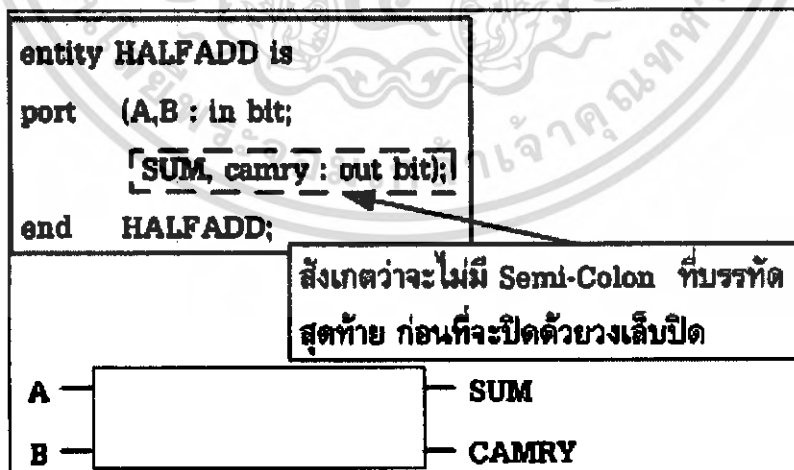
และประเภทของค่า (type of value) ที่สามารถกำหนดให้กับสัญญาณตามจุดต่างๆ (port) ของข้อมูลที่ไหลผ่านจุดต่อเหล่านั้น

2.7.7.1 หน่วยการออกแบบ Entity



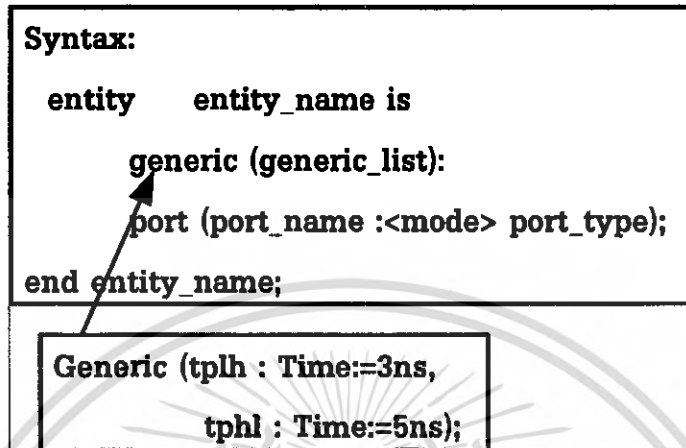
รูปที่ 2.7 รูปแบบการออกแบบ Entity

2.7.7.2 ตัวอย่างการเขียน Entity



รูปที่ 2.8 ตัวอย่างการเขียน Entity

2.7.8 การเขียนโดยมีข้อมูลทางฟิสิกส์

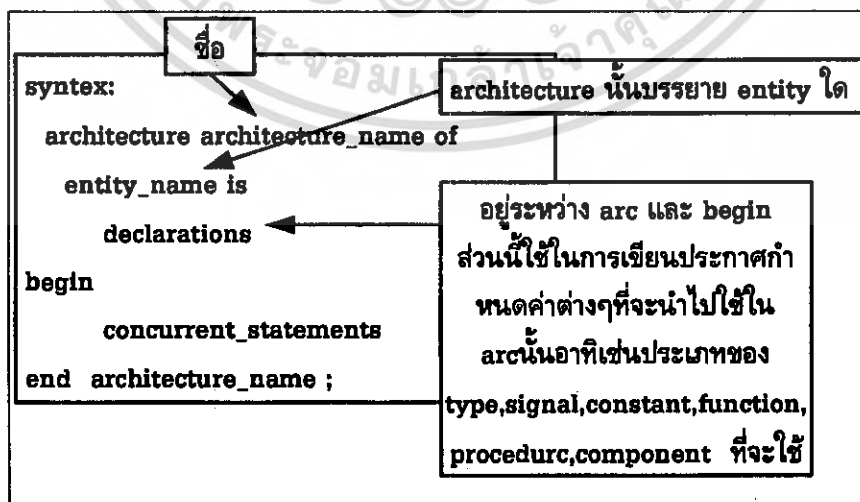


รูปที่ 2.9 การเขียนโดยมีข้อมูลทางฟิสิกส์

2.7.9 การออกแบบส่วนที่ใช้เขียนบรรยายพฤติกรรมการทำงานของวงจร

เป็นส่วนที่เขียนบรรยายพฤติกรรมของรูปแบบ (Model) ในมุมมองของการจำลองการทำงาน (simulation) พฤติกรรมที่บรรยายในส่วนนี้จะขึ้นอยู่กับข้อมูลที่ผ่านเข้าออกตรงช่องทางตลอดจนพารามิเตอร์ต่างๆ (port and generics) ที่กำหนดใน entity design unit Syntax: เริ่มต้นด้วยคำ Architecture ตามด้วยชื่อ และจบด้วย End ตามด้วยชื่อ ดังรูปที่ 2.10

2.7.9.1 Architecture Syntax



รูปที่ 2.10 ตัวอย่างรูปแบบของภาษา VHDL

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.7.9.2 รูปแบบการบรรยายพฤติกรรมของภาษา VHDL

Syntax:

```
architecture architecture_name of entity_name is
    declarations
begin
    concurrent_statements
end architecture_name ;
```

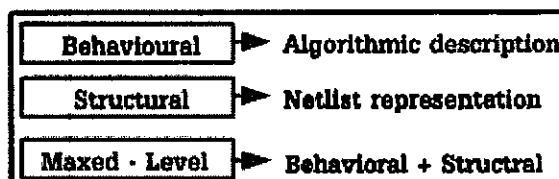
การบรรยายความสัมพันธ์ของข้อมูลที่ไหลเข้าออก model นั้นจะถูกบรรยายระหว่าง begin กับ end ชุดคำสั่งทุกคำสั่งที่อยู่ในบริเวณนี้จะเป็นชุดคำสั่งแบบขนาน (concurrent statement) เท่านั้น arc จะต้องปิดท้ายด้วย end และชื่อของ arc

รูปที่ 2.11 การบรรยายความสัมพันธ์ของข้อมูล

1. Architecture จะเป็นส่วนที่ใช้ในการอธิบายการทำงาน หรือโครงสร้างของวงจร
2. ตัวอย่าง เป็น Architecture ของ Half Adder (HA) ที่มีชื่อว่า "BEHAVE" ของ Entity ที่มีชื่อว่า "HALFADD" ซึ่งจะมีการอธิบายการทำงานระหว่าง Begin => End ดังรูปที่ 2.17
3. Architecture Styles

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

รูปที่ 2.12 ตัวอย่างโครงสร้างที่ชื่อ BEHAVE



รูปที่ 2.13 โมเดลของภาษา VHDL

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.7.10 การบรรยายพฤติกรรมแบบ Behavioural

เป็นการเขียนรูปแบบ VHDL ที่ใช้บรรยายการทำงานที่มีลักษณะคล้ายกับการเขียนโปรแกรมด้วยภาษา C หรือ PASCAL มากกว่าจะเป็นโครงสร้างของ hardware เช่นมีการใช้งานคำสั่งประเภท if-then-else, for-loop เป็นต้นรูปแบบ VHDL เช่นนี้จึงจัดให้อยู่ในประเภท behavioural

2.7.10.1 ตัวอย่าง Behavioural Style

```
architecture architecture of mux is
begin
    process (IN0,IN1,SEL)
    begin
        if (SEL='0') then
            OUTPUT<=IN0;
        else
            OUTPUT<=IN1;
        end if;
    end process;
end behaviour;
```

รูปที่ 2.14 ตัวอย่าง Behavioural Style

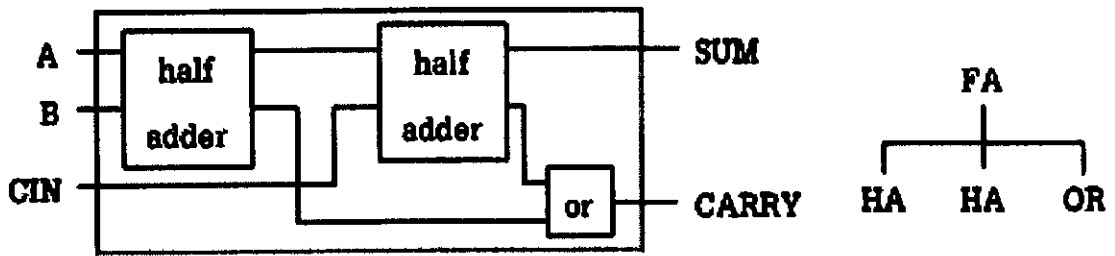
2.7.10.2 การบรรยายพฤติกรรมแบบ Structural

1. เป็นการเขียนเพื่อบรรยายลักษณะทางโครงสร้างของวงจร หรือระบบที่ประกอบด้วยอุปกรณ์หลายๆ ตัวต่อกัน
2. ในการเขียนเพื่ออธิบายนั้น จำเป็นต้องรู้ว่าขาของอุปกรณ์แต่ละตัวต่อกันอย่างไรทำให้เราสามารถมองเห็นหรือกำหนดโครงสร้างได้ เช่น มีการทำงานพร้อมๆ กัน (ขนาน) ของอุปกรณ์ตัวใดบ้าง
3. ตัวอย่าง เช่น การสร้างวงจรแบบ Pipeline เราจำเป็นต้องเขียนในลักษณะของ Structural

2.7.11 การออกแบบการทำงานแบบลำดับชั้น

Hierarchy เป็นการออกแบบที่เป็นลักษณะของลำดับชั้น ซึ่งจะช่วยให้ง่ายในการทำความเข้าใจ ตรวจสอบและแก้ไข

ตัวอย่าง เป็นการออกแบบวงจร Full Adder (FA) ซึ่งประกอบด้วย HA สองตัวและ OR gate เป็นลักษณะของ Hierarchy ดังแผนภาพที่ 2.15



รูปที่ 2.15 ตัวอย่าง การออกแบบวงจร Full Adder

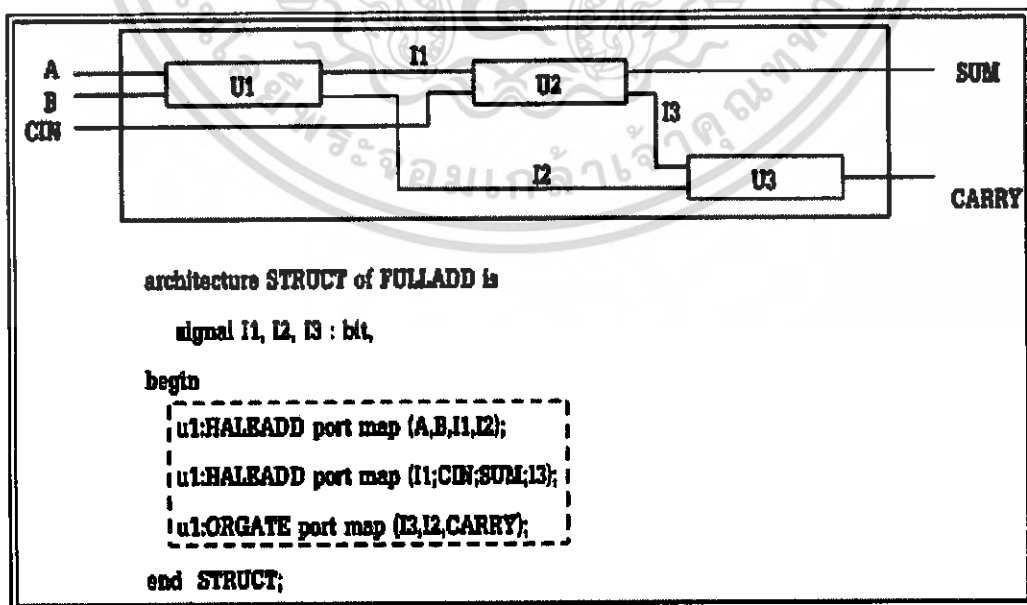
2.7.11.1 การเขียน VHDL ในลักษณะของ Hierarchy (1)

1. เราสามารถแสดงโครงสร้างการเชื่อมต่อกันของ HA และ OR ได้โดย 3 บรรทัดที่อยู่ระหว่าง Begin และ End
2. U1 และ U2 แทน HA และ U3 แทน OR โดยที่ทั้ง U1, U2, U3 เราเรียกว่า Instant Name
3. มีจุดเชื่อมต่อภายในวงจร 3 จุดด้วยกันคือ I1, I2, I3 ซึ่งใน VHDL ถือว่าเป็น Signal

2.7.11.2 การเขียน VHDL ในลักษณะของ Hierarchy (2) ในรูปที่ 2.16

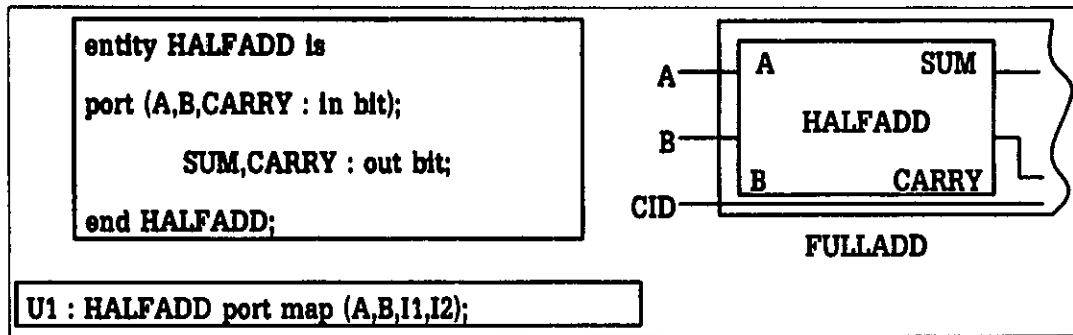
2.7.11.3 การเขียน VHDL ในลักษณะของ Hierarchy (3)

ตัวอย่างการเชื่อมต่อระหว่าง HA กับจุดต่างๆ โดยรอบเราจะใช้คำสั่ง Port map ซึ่งจะทำให้การเชื่อมต่อให้ตรงตามตำแหน่งการเรียงกันของพอร์ตที่กำหนดไว้ใน Entity ในรูปที่ 2.17



รูปที่ 2.16 การเขียน VHDL ในลักษณะของ Hierarchy (2)

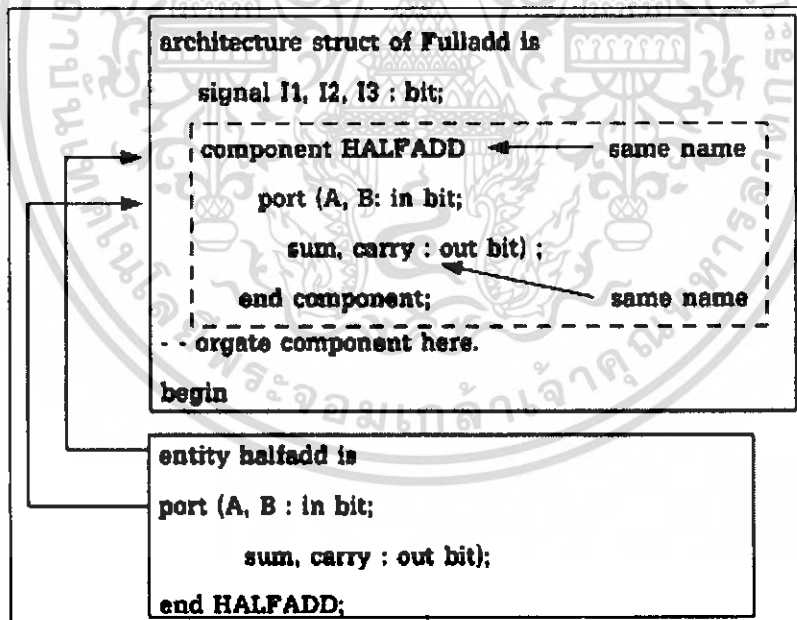
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.17 การเขียน VHDL ในลักษณะของ Hierarchy (3)

2.7.12 การออกแบบ Local Declaration

สำหรับการกำหนดว่าเราต้องการใช้ HA และ OR ในการสร้างจะต้องกำหนดไว้เป็นลักษณะของ Component ในบริเวณของ Local Declaration ชื่อของ Component และตำแหน่งของพอร์ตจะต้องเหมือนกับที่กำหนดไว้ใน Entity ในรูปที่ 2.18



รูปที่ 2.18 Local Declarations

2.8 ประเภทของข้อมูล

Signal ใน VHDL ที่ใช้ในการส่งผ่านค่าจากจุดหนึ่งไปยังอีกจุดหนึ่งนั้น มีได้หลายชนิด (Type) ซึ่งแต่ละ Type นั้นจะกำหนดชุดของค่าที่ Signal ถูกอนุญาตให้ใช้ได้ เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

entity FULLADD is
port (A, B, CIN : in bit;  ←
      SUM, CARRY, : out bit);
end FULLADD;
Architecture STUCT of FULLADD is
  Signal N_SUM : bit  ←
  -- other declaratons
Begin
  -- Code
end STRUCT;

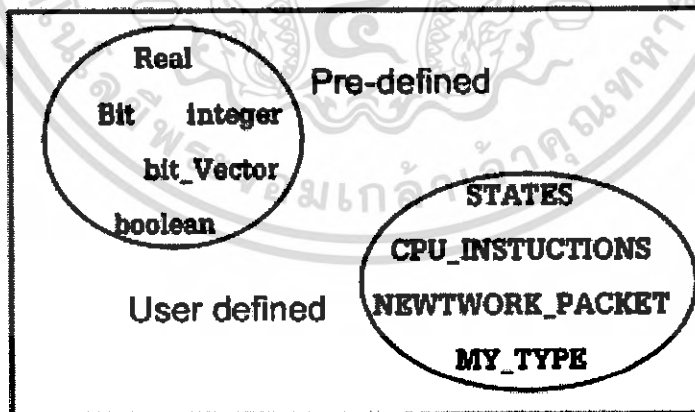
```

รูปที่ 2.19 Signal ใน VHDL

2.8.1 ชนิดของข้อมูล (Type)

Type แบ่งออกได้เป็นสองกลุ่มใหญ่ๆ คือ

1. Pre-Defined
2. User-Defined



รูปที่ 2.20 แบ่ง Type ออกเป็นสองกลุ่มใหญ่ ๆ

2.8.1.1 ข้อมูลแบบ Pre-defined

ใน Package ที่ชื่อว่า standard จะถูกโหลดเข้ามาใช้งานโดยอัตโนมัติโดยไม่ต้องกำหนดไว้ที่บรรทัด

ต้นไฟล์ ซึ่งมีชนิดต่างๆ ของ Type

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.8.1.2 ข้อมูลแบบ Standard Pre-defined

ตัวอย่างการเขียน ในรูปที่ 2.21

2.8.1.3 แนวคิดของการกำหนดชนิดข้อมูล

เมื่อเราต้องการกำหนด (Assign) ค่าให้กับ Signal ชนิดของ Type ทั้งด้านซ้ายและด้านขวาของตัวส่งผ่านค่า ($\leq$) จะต้องเป็นชนิดเดียวกัน ดังตัวอย่าง ในรูปที่ 2.22

```
Boolean : FALSE, TRUE
Bit : '0' , '1'
Bit_Vector : '00010' "111"...
Character: 'a' , 'b' , 'A' , 'Z' , '1' , '0' ....
String : "Hello" , "VHDL"...
Integer : 0, 1, - 45, 137 ,256 ,...
Real ; 1.02, - 37.4,...
Time : 10 ns, 200 fs, 2.5 ps,...
```

รูปที่ 2.21 ตัวอย่างชนิดข้อมูลแบบ Standard Pre-defined

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

```
SUM <= A xor B;
```

Types must match...

รูปที่ 2.22 การกำหนดชนิดข้อมูล

2.8.1.4 ชนิดข้อมูลแบบอื่นๆ ที่เป็นแบบ Pre-defined Type

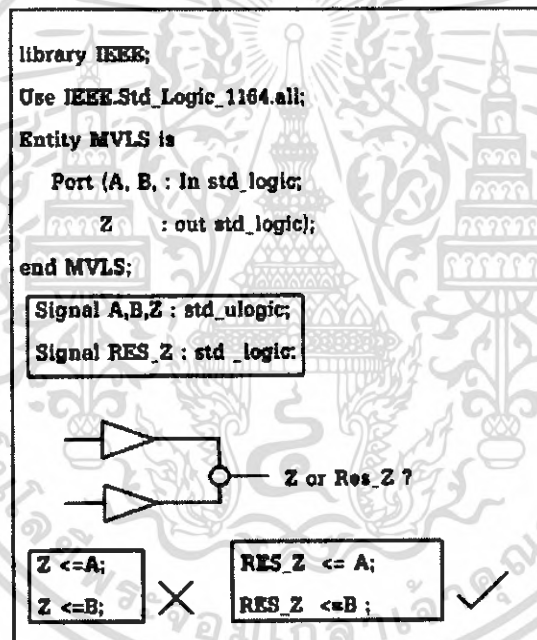
นอกจากนี้เรายังมี Type ที่กำหนดอยู่ใน Package อื่นๆ ที่ไม่ใช่ Standard package ดังตาราง ดังนั้นก่อนที่จะใช้งานต้องกำหนดการใช้งาน Package ไว้ตอนต้นด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 2.2 การกำหนดการใช้งาน Pre-defined Type

Package	Type
std_logic_1164	std_logic, std_logic_vector std_ulogic, std_ulogic_vector
std_logic_arith	signed, unsigned
'U' 'X' '0' '1' 'Z' 'W' 'L' 'H' '-' std_logic หรือ std_ulogic "1010" signed = -6 unsigned = 10	

2.8.1.5 การใช้ Standard Logic



รูปที่ 2.23 การใช้ Logic พื้นฐาน

2.8.1.6 การแปลงข้อมูลต่างประเภท

เนื่องจากทั้งด้านซ้าย และด้านขวาของตัวส่งผ่านค่า (<=) จะต้องเป็น Type เดียวกันถ้าต่างชนิดกัน จะทำอย่างไร ตัวอย่างในรูปที่ 2.24

2.8.1.7 การแปลงข้อมูลแบบ User-defined

1. เราสามารถที่จะกำหนดค่าต่างๆ ที่เป็นไปได้สำหรับ Type ที่เรากำหนดขึ้นมาเอง
2. การใช้งานจะต้องเป็นไปตามกฎนั้นคือ ด้านซ้ายและด้านขวาของตัวส่งผ่านค่าจะต้องเป็น

Type ชนิดเดียวกัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

ENTITY adder IS
PORT (op1, op2 : IN UNSIGNED(3 downto 0);
      result : OUT INTEGER);
END adder;

ARCHITECTURE maxpld OF adder IS
BEGIN
      result <= CONV_INTEGER(op1 + op2);
END maxpld;

```

รูปที่ 2.24 ตัวอย่างข้อมูลต่างชนิดกัน

2.8.1.8 การใช้งานข้อมูลแบบ User-defined

User-defined type อาจจะใช้ในการกำหนดชื่อของแต่ละ State ที่อยู่ใน State Machine ซึ่งเวลานำไปแปลด้วยเครื่องจะมีการเข้ารหัสเป็นเลขฐานสองเรียงตามลำดับที่กำหนดไว้

2.9 ข้อมูลประเภทอะเรย์ (Array)

ในภาษา VHDL สามารถใช้งานข้อมูลประเภทอะเรย์โดยที่ข้อมูลประเภทอะเรย์ จะเป็นกลุ่มชนิดข้อมูลประเภท Scalar ตัวอย่างเช่น การสร้างชนิดของอะเรย์ของ std_ulogic โดยอะเรย์ของข้อมูล std_ulogic คือ std_ulogic_vector สามารถสร้างได้โดยใช้คำสั่ง TYPE ในการข้อมูลประเภทอะเรย์ (รายละเอียดอยู่ใน package STD_LOGIC_1164 ดังแสดงในตัวอย่างดังรูป 2.25

```
TYPE std_ulogic_vector IS ARRAY (NATURAL RANGE <>) OF std_ulogic;
```

รูปที่ 2.25 ตัวอย่างอะเรย์ std_ulogic

เช่นเดียวกับข้อมูลประเภทอะเรย์ของ std_logic โดยอะเรย์ของข้อมูล std_logic คือ std_logic_vector

```
TYPE std_logic_vector IS ARRAY (NATURAL RANGE <>) OF std_logic;
```

รูปที่ 2.26 ตัวอย่างอะเรย์ std_logic

หรือเป็นประเภท STRING หรือ BIT_VECTOR ก็ตามใน package STANDARD ในรูปที่ 2.27

TYPE string IS ARRAY (POSITIVE RANGE <=>) OF CHARACTER;
TYPE BIT_VECTOR IS ARRAY (NATURAL RANGE <=>) OF BIT;

รูปที่ 2.27 ตัวอย่างอะเรย์ STRING หรือ BIT_VECTOR

แต่ในบางครั้งผู้ออกแบบสามารถสร้างข้อมูลประเภทอะเรย์ได้ โดยใช้คำสั่ง TYPE โดยมีรูปแบบดังรูปที่ 2.27

TYPE TYPE_NAME IS ARRAY (RANGE) OF element_type;

รูปที่ 2.28 ตัวอย่างอะเรย์ โดยใช้คำสั่ง TYPE

2.10 รูปแบบคำสั่ง (Sequential)

1. VHDL นั้นเป็นภาษาที่มีคุณสมบัติแบบแข่งขันกัน นั่นคือ ชุดคำสั่งภายในตัวโครงสร้างจะเป็นชุดคำสั่งแบบแข่งขันกัน
2. โครงสร้างที่ประกอบด้วย Sequential Statement เช่น if-then-else, case, for-loop, wait จะไปอยู่ส่วนไหนอย่างไรภายใน architecture ของ model? คำตอบก็คืออยู่ในชุดคำสั่ง Process
3. ใน architecture หนึ่ง สามารถมีได้หลายๆ Process ที่ทำงานขนานกัน

2.10.1 การใช้คำสั่ง Process

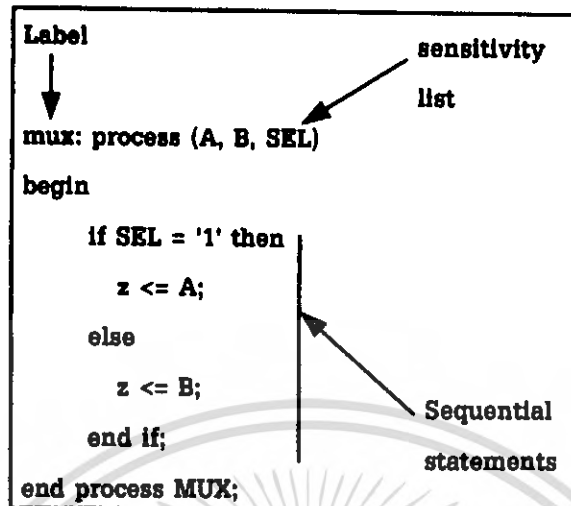
สำหรับบางคำสั่งที่เป็น Sequential Statement ต้องกำหนดให้อยู่ใน Process ซึ่งจะทำให้การแปลความหมายลงมาถึงบรรทัด

ใน Architecture หนึ่งๆ อาจจะมีหลายๆ Process ซึ่งทำงานพร้อมๆ กันเป็นลักษณะของ Concurrent

แต่ละ Process จะทำงานเมื่อ Signal ที่กำหนดไว้ใน Sensitivity List มีการเปลี่ยนแปลง

2.10.2 ตัวอย่างการใช้งาน Process

1. จากตัวอย่าง Label จะมีหรือไม่ก็ได้ ตามด้วย Process และชื่อ sensitivity List อยู่ในวงเล็บ
2. Sequential Statement จะอยู่ระหว่าง Begin และ End Process
3. จากตัวอย่างนี้ เมื่อนำไปสังเคราะห์เป็นวงจรจะได้เป็นวงจร Combinational Logic ชดเชยไม่ได้ ไม่ใช่ Sequential Logic รูปที่ 2.29



รูปที่ 2.29 ตัวอย่างการใช้งาน Process

2.10.3 การประมวลผลใน Process

การประมวลผลใน Process จะเรียงลำดับกันเป็นบรรทัดลงมา (Sequential) ซึ่งต่างกับ บริเวณที่อยู่ข้างนอก Process ซึ่งจะทำงานพร้อมๆ กัน (Concurrent) ดังนั้นถ้าเรากำหนดไว้ที่บริเวณต่างกัน ก็จะได้ลอจิกที่ต่างกันด้วยดังตัวอย่างต่อไปนี้

```

architecture sequential of multiple is
  signal z, a, b, c, d: std_logic;
begin
  Z <= A and B;
  Z <= C and D;
end concurrent;

```

รูปที่ 2.30 การประมวลผลใน Process (1)

```

architecture sequential of multiple is
  signal z, a, b, c, d: std_logic;
begin
  process(a, b, c, d)
  begin
    Z <= A and B;
    Z <= C and D;
  end process
end sequential;

```

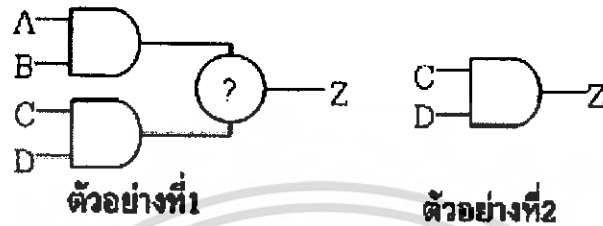
Annotation: **value assigned** points to the assignment `Z <= C and D;` inside the process.

รูปที่ 2.31 การประมวลผลใน Process (2)

จากตัวอย่างที่ 1 เนื่องจาก Z ถูกกำหนดค่าพร้อมๆ กันเนื่องจากเป็น Concurrent ดังนั้นค่าของ Z จะต้องเลือกเอาค่าใด ค่าหนึ่งของเอชท์ทุกลอจิก ทั้งสองตัวที่เข้ามา

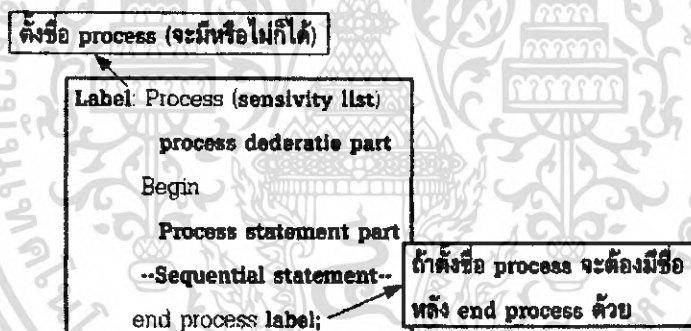
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำหรับตัวอย่างที่ 2 ค่าของ Z จะมีการเปลี่ยนแปลงค่า หรือถูกUpdate ด้วยค่าสุดท้ายที่เกิดขึ้นที่บรรทัดสุดท้าย บรรทัดแรกจะถูกมองข้ามไปเนื่องจากจะถูกแปลทับด้วยบรรทัดที่สอง เพราะว่าเป็นลักษณะการแปลความหมายเรียงกันลงมาที่ละบรรทัด



รูปที่ 2.32 การประมวลผลใน Process

2.10.4 กฎเกณฑ์การเขียน Process



รูปที่ 2.33 กฎเกณฑ์การเขียน Process

2.10.5 การกำหนดตัวแปรของคำสั่ง Process

สำหรับ Process ใน VHDL เราสามารถกำหนดตัวแปรเป็น Variable ได้ ซึ่งจะมีลักษณะเหมือนกับภาษาซีสูงโดยทั่วไปนั่นคือจะมีการ update ค่าทันทีซึ่งต่างกับ Signal ที่จะมีการ update ค่าในตอนสิ้นสุด Process เท่านั้น

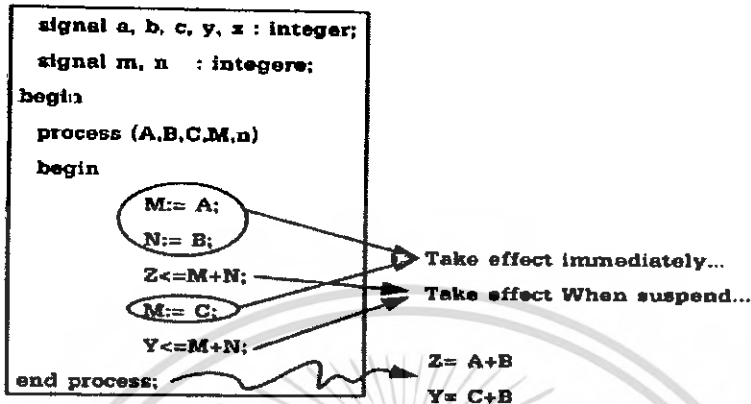
ในการกำหนดค่าให้ Variable จะใช้เครื่องหมาย: =

Variable จะรู้จักเฉพาะใน Process ที่กำหนดขึ้นมาเท่านั้น ซึ่งต่างกับ Signal ที่จะรู้จักตลอดทั้งใน Architecture

Variable เป็นเพียงตัวแปร ไม่มีความเกี่ยวข้องกับ Hardware แต่อย่างไรส่วน Signal นั้นเป็นตัวที่ทำหน้าที่เป็นพาหะ รับ-ส่ง ค่าของสัญญาณจึงเป็นส่วนของ Hardware

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.10.5.1 ตัวอย่างการใช้งาน Variable & Signal



รูปที่ 2.34 ตัวอย่างการใช้งาน Variable & Signal

2.10.6 ชุดคำสั่งที่ใช้ใน Process

1. เป็นชุดคำสั่งที่สำคัญที่สุดสำหรับการเขียนรูปแบบในลักษณะของการบรรยายพฤติกรรมของระบบดิจิทัล
2. ชุดคำสั่งเหล่านี้จะทำงานเป็นลำดับจากบนลงล่างเมื่อ Process ถูกกระตุ้นให้ทำงาน
3. จะเป็นตัวกำหนดเขตในส่วนของ Sequential statement
4. การทำงานภายใน Process จะไม่ขึ้นกับเวลานั้นหมายความว่าชุดคำสั่ง Process จะทำงานตลอดเวลา

2.10.7 คำสั่ง If-Then-Else Statement

กฎเกณฑ์ (Syntax) ในการเขียนภาษา VHDL ในรูปที่ 2.34

```

IF condition THEN
    {sequential-
    statement(s)}
    [( ELISIF comdition THEN)
    {sequential-
    {statement(s)}}]
    [ ELSE
END IF;

```

รูปที่ 2.35 ตัวอย่างคำสั่ง If-Then-Else Statement

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้สำหรับใช้ภายในเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.10.8 คำสั่ง Case Statement

กฎเกณฑ์ (Syntax) ในการเขียนภาษา VHDL ในรูปที่ 2.35

```
CASE expression IS
    WHEN choice => sequential-statement(s)
  [ WHEN choice => sequential-
statement(s)]
END CASE;
```

รูปที่ 2.36 ตัวอย่างกฎเกณฑ์ในการเขียนภาษา VHDL

คำสั่ง WHEN ใช้สำหรับกำหนดตัวเลือกที่จะนำมาเปรียบเทียบกับ expression ตัวเลือกใดเป็นไปตาม expression ข้างบน PROCESS จะเริ่มต้นทำงานที่ชุดคำสั่งลำดับที่ตามมาจนกระทั่งคำสั่งสุดท้ายของตัวเลือกนั้นๆ และออกจาก CASE statement โดยไม่กำหนดทางอื่นๆ ที่ยังเหลืออยู่

2.10.9 คำสั่ง Loop Statement

คือการทำงานในลักษณะที่เป็นวงรอบ (Loop) มีกฎเกณฑ์ในการเขียนดังรูปที่ 2.36

```
Label: iteration_scheme loop
    sequential-statement(s)
END loop label;
```

รูปที่ 2.37 ตัวอย่างการทำงาน Loop Statement

เราสามารถสร้าง Loop ได้ 3 ลักษณะด้วยกันคือ LOOP ธรรมดา, FOR-LOOP และ WHILE-LOOP

2.10.9.1 คำสั่ง For-Loop Statement

กลุ่มคำสั่ง For-loop จะทำงานเป็นวงรอบครบเท่าที่ค่าพารามิเตอร์ยังอยู่ในขอบข่าย (range)

```
Label: FOR loop_parament IN discent_range loop
    sequential-statement(s)
END loop label;
```

รูปที่ 2.38 ตัวอย่างการทำงาน For-Loop Statement

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้เพื่อการศึกษาเท่านั้น เมื่อผู้ใดได้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตัว `loop_parameter` เป็นพารามิเตอร์ที่แมงอยู่ในโครงสร้างที่ไม่ต้องประกาศหรือกำหนดขึ้น ค่าพารามิเตอร์นี้ห้ามถูกแก้ไขดัดแปลงด้วยคำสั่งใดๆ ทั้งจากภายในและจากภายนอก `for loop` ตัวอย่างในรูปที่ 2.38

```
b1: FOR index IN 0 to 7 loop  
      dout(index)<= din(index);  
END loop b1;
```

รูปที่ 2.39 ตัวอย่าง For-Loop

จากตัวอย่างเป็นการย้ายข้อมูลในรูปของ array ชื่อ `din` ไปยัง array ที่ชื่อ `dout` ตำแหน่งต่อตำแหน่งโดยอาศัย `loop_parameter` ชื่อ `index`

สิ่งที่พึงจำในการสร้างคือ Discrete range จะต้องเป็นค่าที่ลงตัวตัวเลขจำนวนเต็มสามารถใช้ได้หรือตัวอักษร เช่น `a to f`

2.10.9.2 คำสั่ง While Loop Statement

```
Label: while boolean_expression loop  
      sequential-statement(s)  
END loop Label;
```

รูปที่ 2.40 ตัวอย่างการทำงาน While Loop Statement

กลุ่มคำสั่งลำดับใน WHILE-LOOP จะทำงานเป็นวงรอบไปเรื่อยๆ ตราบเท่าที่ข้อแม้ของ `boolean_expression` ใน `while-loop` ยังคงมีค่าเป็น `true` จำนวนครั้ง (รอบ) ที่ทำจะถูกควบคุมจากในตัวของ loop เองนั่นคืออาการที่จะออกจากวงรอบได้ก็ต่อเมื่อผลลัพธ์ของ `boolean_expression` ได้ค่าเป็น `false` ข้อแม้ที่ใช้ควบคุมการทำงานนี้จะถูกตรวจสอบทุกครั้งก่อนการทำงานต่อไปในวงรอบ

ตัวอย่าง While Loop

```

P1: process (A)
  variable index : intage:=0;
  begin
    loop1 : while index< 8 loop
      dout(index <index+1;
    end loop Loop1;
  end process P1

```

รูปที่ 2.41 ตัวอย่าง While Loop

2.10.10 คำสั่ง Wait Statement

จากการที่ชุดคำสั่ง Process จะถูกกระตุ้นได้จากการที่สัญญาณใดสัญญาณหนึ่งใน sensitivity list เกิด event ขึ้นเท่านั้นหลังจากกระตุ้นแล้วคำสั่งภายในจะทำงานเป็นลำดับลงมาจนกระทั่งหมดและชุดคำสั่ง Process จะหยุดทำงานชั่วคราวจนกว่าสัญญาณใน list สัญญาณใดสัญญาณหนึ่งเกิด event ขึ้นอีก

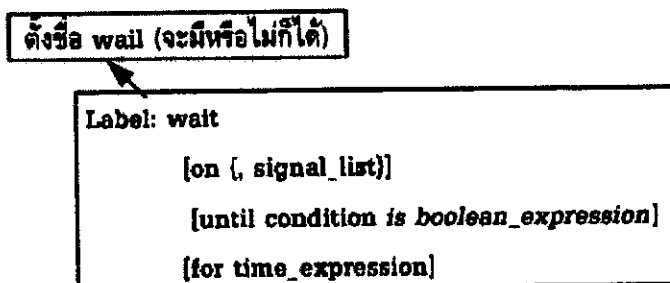
ถ้าในกรณีที่ภายในรายชื่อภายใน List ประกอบด้วยสัญญาณหลายๆ ตัว และเกิด event ขึ้นในเวลาเดียวกัน จะมีสัญญาณตัวเดียวจากทั้งหมดเท่านั้น ที่กระตุ้นการทำงานของชุดคำสั่ง process ซึ่งไม่สามารถที่จะบอกได้ว่าเป็นสัญญาณใดดังนั้นการใช้คำสั่ง process ร่วมกับ sensitivity list จึงมีขีดจำกัด

ใน VHDL จึงหลีกเลี่ยงปัญหานี้โดยใช้คำสั่งที่มีความอ่อนตัวมากกว่าได้แก่ชุดคำสั่ง wait statement

สามารถกำหนดลงตรงตำแหน่งใดๆ ภายในชุดคำสั่งลำดับได้เพื่อรับการดำเนินงานภายในชุดคำสั่ง Process ตรงตำแหน่งที่ wait statement อยู่

สามารถที่จะกำหนด Wait statement ได้หลายๆ อันในแต่ละตำแหน่งตามความต้องการโดยไม่จำกัด Wait statement ไม่สามารถใช้ร่วมกับ sensitivity list ภายใน process เดียวกันได้

2.10.10.1 กฎเกณฑ์การเขียน Wait Statement (1)



รูปที่ 2.42 กฎเกณฑ์การเขียน Wait Statement

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การใช้ On ร่วมกับ until การทำงานจะถูกหยุดจนกระทั่งสัญญาณในส่วนของ on มี event และ
ข้อแม้ในส่วนของ until เป็นจริง

การใช้ On ร่วมกับ for การทำงานจะถูกหยุดจนกระทั่งสัญญาณในส่วนของ on มี event หรือ เมื่อ
เวลาในส่วน for ผ่านไป

2.10.10.2 กฎเกณฑ์การเขียน Wait Statement (2)

การใช้ Until ร่วมกับ for การทำงานจะถูกหยุดจนกระทั่งข้อแม้ในส่วนของ until เป็นจริง หรือเมื่อ
เวลา for ผ่านไป

ถ้าชุดคำสั่ง Process ใดไม่มี sensitivity list และ wait statement ภายในโครงสร้าง process นั้น
จะเกิดการทำงานเป็นวัฏจักรที่ไม่มีจุดจบ



บทที่ 3

การออกแบบ การสร้าง และการทำงาน

3.1 กล่าวนำ

บอร์ดทดลองรุ่น CPLD Explorer XC9572XL จะมีส่วนของแสดงผลเป็นแบบ Logic Monitor 4 ดวง Slide Switch ส่วน I/O รับได้ทั้งแบบ 5V และ 3.3V Oscillator ที่ความถี่ 32.768 kHz เพื่อให้เหมาะสำหรับการทดลองออกแบบวงจรดิจิทัลและออกแบบไอซีพื้นฐาน บอร์ดทดลองนี้สามารถใช้เป็นเครื่องโปรแกรมชิพ CPLD ได้อีกด้วย

CPLD (Complex Programmable Logic Device) หรือ FPGA (Field Programmable Gate Arrays) ในแง่ของผู้ใช้งานทั่วไป ก็คือไอซีหรือชิพเนกประสงค์ที่สามารถโปรแกรมให้เป็นวงจรดิจิทัลอะไรก็ได้โดยวิธีการโปรแกรมแบบง่ายๆ และสามารถแก้ไขวงจรได้โดยการโปรแกรมซ้ำ ชิป CPLD จะเหมาะกับการออกแบบวงจรขนาดเล็กถึงขนาดกลาง วงจรที่โปรแกรมไว้ใน CPLD จะคงอยู่แม้จะไม่มีไฟเลี้ยงก็ตาม

3.2 คุณสมบัติทั่วไปของบอร์ดทดลองรุ่น CPLD Explorer XC9572XL

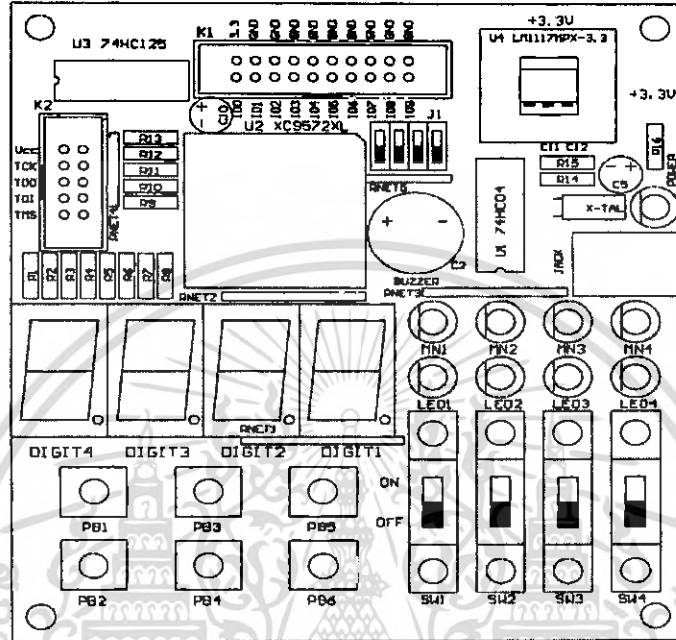
1. CPLD เบอร์ XC9572XL (1,600 เกต) แบบ PLCC44 ขา Speed Grade -10
2. 7-Segment จำนวน 4 หลัก
3. LED แสดงผล 2 สถานะ จำนวน 4 ดวง
4. Logic Monitor ที่เป็น LED แสดงผล 3 สถานะ จำนวน 4 ดวง (ใช้ร่วมกับ K1)
5. Buzzer จำนวน 1 ตัว
6. Slide switch 4 บิต (ใช้ร่วมกับ Push button switch)
7. Push button switch 6 ตัว
8. พอร์ต K1 เป็น I/O 10 บิตที่สามารถใช้กับ I/O 5V และ 3.3V ได้
9. Oscillator 32.768 kHz

3.3 หลักการทำงานของบอร์ดทดลองรุ่น CPLD Explorer XC9572XL

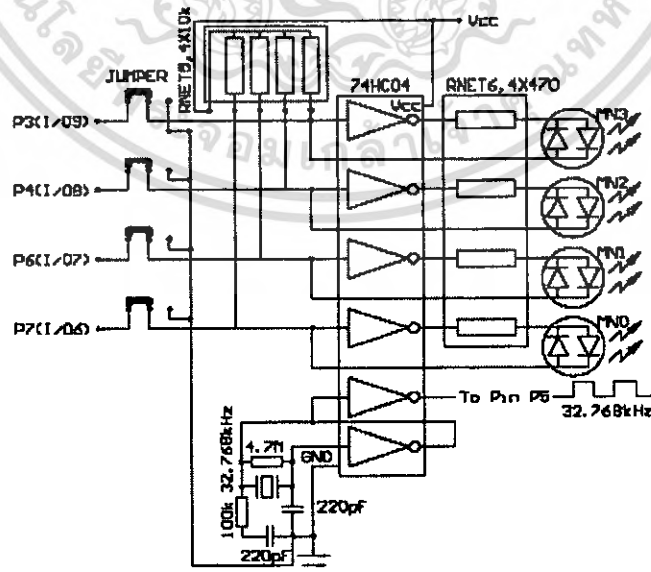
จากรูปที่ 3.1 บอร์ดทดลองรุ่น CPLD Explorer XC9572XL จะมีส่วนของแสดงผลเป็นแบบ Logic Monitor 4 ดวง Slide Switch ส่วน I/O รับได้ทั้งแบบ 5V และ 3.3V Oscillator ที่ความถี่ 32.768 kHz ทางด้านเอาต์พุตของบอร์ดทดลองนี้มี LED แบบ 2 สถานะ 4 ดวง คือ L1-L4 โดยต่อขาแคโทด (Cathode) ลงกราวด์และต่อขาแอโนดเข้ากับขา I/O ของ CPLD โดยมีตัวต้านทาน RNET2 470 โอห์มต่ออนุกรมอยู่เพื่อจำกัดกระแส ดังนั้นถ้า CPLD ส่งลอจิก "1" จะทำให้ LED ดวงนั้นติดสว่าง Logic Monitor

เอกสารนี้เป็นเอกสารทบทวนเนื้อหาสำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ทางการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมีเหตุดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ของ CPLD ที่ K1 (I/O6-I/O9) มีรายละเอียดดังรูปที่ 2 ถ้า I/O ของ CPLD เป็นลอจิก "1" ทำให้ LED ติดเป็นสีเขียว ถ้าลอจิก "0" ทำให้ LED ติดเป็นสีแดง แต่ถ้าเป็น High impedance LED จะดับดังรูปที่ 3.2



รูปที่ 3.1 การจัดวางอุปกรณ์บน CPLD Explorer XC9572XL



รูปที่ 3.2 วงจร Logic trainer และ Oscillator

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับใช้ภายในเท่านั้น ไม่ควรเผยแพร่ภายนอกโดยไม่ได้รับอนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตัวแสดงผล 7-Segment จำนวน 4 ตัว คือ DIGIT4-DIGIT1 โดนใช้เทคนิคการสแกนในการแสดงตัวเลขบนตัวแสดงผลทั้ง 4 ตัวพร้อมๆ กันเพื่อประหยัดสายสัญญาณ โดยมีหลักการคือ ทำการส่งตัวเลขไปที่หลักแรกแล้วทำการเลือกให้หลักแรกแสดงผล ถ้าเปลี่ยนเป็นส่งตัวเลขของหลักที่สองแล้วเปลี่ยนไปเลือกหลักที่สองให้แสดงผล จากนั้นก็วนไปจนครบทั้ง 4 หลักแล้วกลับมาหลักแรกใหม่ จะทำให้ผู้ใช้มองเห็นตัวเลขติดพร้อมกัน แต่มีข้อแม้ว่าการสแกนหลักในการแสดงผลต้องทำให้เร็วกว่าที่ตาคนเราจะมองได้ทัน ซึ่งโดยทั่วไปตาคนเราจะมองแยกได้ทีประมาณ 25-30 ครั้งต่อวินาที ดังนั้นควรตั้งสแกนโดยใช้ความเร็วไม่น้อยกว่า 120 ครั้งต่อวินาที (30 ครั้ง x 4 หลัก-120 ครั้งต่อวินาที) จึงมองไม่เห็นการกระพริบ ตัวแสดงผล 7-Segment ที่ใช้เป็นแบบแคโทดรวมนั้นถ้าเราจะให้ 7-Segment โดของตัวแสดงผลติดต้องให้ลอจิก "1" โดยมีตัวต้านทาน R1- R8 ขนาด 100 โอห์มต่ออนุกรมอยู่เพื่อจำกัดกระแส ดังนั้นหากต้องการความสว่างมากขึ้นให้ลดค่าตัวต้านทานนี้ลง และถ้าต้องการให้หลักใดติดก็ต้องให้ขาคอมมอนของหลักนั้นเป็นลอจิก "0" โปรดสังเกตว่าตัวแสดงผล 7-Segment ตัวที่ 3 และ 4 หรือสองตัวสุดท้ายทางขวามือถูกออกมาให้มีการกลับตัวแสดงผลโดยการหมุน 180 องศา เพื่อให้จุดของตัวที่ 3 และ 4 ขึ้นไปอยู่ด้านบนเพื่อประโยชน์ในการแสดงเครื่องหมาย ":" ในการทำนาฬิกาหรือแสดงเป็นองศาในงานที่เกี่ยวข้องกับอุณหภูมิแต่การแสดงผลตัวเลขต่างๆ ยังคงใช้สายสัญญาณเดียวกับสองตัวแรก

พอร์ตต่อสายสัญญาณไปยังภายนอกอีก 10 เส้น คือ I/O0-I/O9 โดยต่ออยู่ที่คอนเนคเตอร์ K1 เพื่อเชื่อมต่อสัญญาณไปยังอุปกรณ์ภายนอกหรือรับสัญญาณจากเซนเซอร์ต่างๆ การต่อสายสัญญาณ I/O ถ้าต้องการใช้ทำงานที่ความถี่สูงๆ ได้ถึง 50 MHz โดยใช้สายแพรขนาด 20 เส้นที่ความยาวสัญญาณไม่ควรยาวเกิน 10-15 เซนติเมตร (เมื่อโปรแกรม CPLD ใช้ I/O เป็นแบบ Slow Slew Rate เพื่อลดสัญญาณรบกวนข้ามช่องและลดการสะท้อนของสัญญาณในสายแพร) โดยที่สายกราวด์ทุกเส้นจะต้องต่อลงกราวด์ทั้งสองด้านและควรออกแบบ PCB เป็นแบบมีกราวด์เพลน

Buzzer จำนวน 1 ตัว ที่ต่อขาข้างหนึ่งลงกราวด์และต่อขาอีกข้างซึ่งเป็นขั้วบวกกับขาอินพุต เอาต์พุตของ CPLD ดังนั้นถ้า CPLD ส่งลอจิก "1" มาจะทำให้ Buzzer ดัง หากจะให้ Buzzer ดับก็ส่งเป็นลอจิก "0" โดยไม่จำเป็นต้องส่งเป็นสัญญาณพัลส์

ทางด้านอินพุตบอร์ดทดลองนี้มีสวิตช์กดติดปล่อยดับอยู่ 6 ตัวคือ PB1-PB6 ต่ออยู่กับขาของ CPLD โดยปกติหากไม่มีการกดจะมีลอจิก "1" และหากกดปุ่มจะทำให้ได้ลอจิก "0" เนื่องจากมีตัวต้านทาน RNET1 ต่อพูลอัพอยู่ อีกทั้งบนบอร์ดยังมีสวิตช์แบบเลื่อน 4 ตัว หากเลือกสวิตช์ลงไปที่ OFF จะทำให้ได้ลอจิก "1" และหากเลื่อนสวิตช์ขึ้นไป ON จะทำให้ได้ลอจิก "0" เพียงแต่ว่าสวิตช์ PB3-PB6 จะใช้สัญญาณร่วมกับสวิตช์เลื่อนตัวที่ 1-4 ตามลำดับ ดังนั้นหากกดสวิตช์ PB3 ค้างไว้

จะเหมือนกับการเลื่อนสวิตช์ 1 ขึ้นไปที่ ON บอร์ดทดลองนี้ใช้ไอซี 74HC04 ติดตั้งบนซอกเกิดเพื่อประกอบเป็นออสซิลเลเตอร์บนความถี่ 32.768 kHz

บอร์ดทดลองนี้ใช้ไฟเลี้ยง 4.5-12 VDC โดยต่ออะแดปเตอร์ที่มีสายด้านในเป็นไฟบวกและด้านนอกเป็นกราวด์เพื่อลดระดับแรงดันให้คงที่เหลือ 3.3 V โดยใช้ชิปเบอร์ LM1117MPX-3.3 และหากมีไฟเลี้ยงเข้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งยังมีหลอดแบบส่องเนอ้า และต้องอ้างอิงถึงเงาของเอกสารทุกครั้งที่มีการนำใบใช้

บอร์ดจะทำให้ LED Power บนบอร์ดสว่างด้วย สำหรับตัวเก็บประจุต่างๆ ที่ติดตั้งอยู่บนบอร์ดมีไว้เพื่อรักษา ระดับแรงดันไฟเลี้ยงให้คงที่ที่ 3.3V โดยไม่มีการกระเพื่อมและช่วยกำจัดสัญญาณรบกวนบนบอร์ด

JTAG คอนเนคเตอร์ K1 ใช้สำหรับต่อสายดาวนโหลดเข้ากับพอร์ตขนานของคอมพิวเตอร์เพื่อ โปรแกรมข้อมูลลง จร ลง CPLD โดยขาสัญญาณของ JTAG ทุกเส้นจะต่อผ่านไอซี TTL ที่เป็นบัฟเฟอร์เบอร์ 74HC125 เพื่อป้องกันสัญญาณรบกวนในสาย JTAG โดยมีตัวต้านทาน R9 และ RNET4 ต่อพูลอัพ ส่วน R10-R13 ต่อไว้เพื่อลดสัญญาณสะท้อน สาย JTAG จะใช้สายรบบินสีเทาขนาด 10 เส้นยาว 1.5 เมตร

การโปรแกรมบอร์ดทดลองจะใช้โปรแกรมวงจรดิจิทัลต่างๆ ลงบอร์ดโดยทำการต่อสาย JTAG และต่อสายไฟเลี้ยง 6-9VDC โดยการต่ออะแดปเตอร์ที่มีสายด้านในเป็นไฟบวกและด้านนอกเป็นกราวด์ เข้า บอร์ดและจะเห็นแอลอีดีไฟาเวอร์สว่าง หลังจากนั้นจึงทำการดาวนโหลดวงจรที่ต้องการลงสู่ชิพ CPLD ใน ขั้นตอนก่อนการโปรแกรมจะต้องมีการกำหนดขาเป็นตามตารางที่ 3.1

ตารางที่ 3.1 แสดงตำแหน่งขาของชิพที่ต่ออยู่กับฮาร์ดแวร์ภายนอกที่อยู่บนบอร์ด

LED		7-Segment	
I/O Name	Pin No	I/O Name	Pin No.
L1	P38	A	P27
L2	P37	B	P26
L3	P36	C	P25
L4	P35	D	P24
Push Button Sw		E	P22
I/O Name	Pin No.	F	P20
PB1	P39	G	P18
PB2	P40	Dot	P19
PB3	P42	DIGIT4(SEG1)	P28
PB4	P43	DIGIT3(SEG2)	P29
PB5	P44	DIGIT2(SEG3)	P33
PB6	P1	DIGIT1(SEG4)	P34

ตารางที่ 3.1 (ต่อ) แสดงตำแหน่งขาของชิพที่ต่ออยู่กับฮาร์ดแวร์ภายนอกที่อยู่บนบอร์ด

SLIDE SW		I/O ของ K1		REMARK
I/O Name	Pin No.	I/O Name	Pin No.	
SLIDE SW	P42	IO0	P14	
SLIDE SW	P43	IO1	P13	
SLIDE SW	P44	IO2	P12	
SLIDE SW	P1	IO3	P11	
Misc		IO4	P9	
I/O Name	Pin No.	IO5	P8	
Buzzer	P2	IO6	P7,MN1	J1-1
		IO7	P6,MN2	J2-2
OSC = 32.768 kHz	P5	IO8	P4,MN3	J3-3
		IO9	P3,MN4	J4-4

3.4 การออกแบบโรงงานของชุดฝึกดิจิทัลด้วย CPLDs

ในการออกแบบโรงงานนั้น จะแบ่งโรงงานออกด้วยกันทั้งหมด 13 โรงงาน โดยในแต่ละโรงงานจะเป็นการออกแบบวงจรดิจิทัลด้วยภาษาอธิบายฮาร์ดแวร์ คือการเขียนชุดคำสั่งในแต่ละโรงงาน และเอาชุดคำสั่งที่ได้ทำการออกแบบนำไปโปรแกรมลงตัวชิพ เพื่อนำไปทดลองกับบอร์ดลงรุ่น CPLD Explorer XC9572XL โดยที่โรงงานทั้ง 13 โรงงาน จะมีดังนี้

โรงงานที่ 1 การใช้โปรแกรม Xilinx 7.1i เมื่อเริ่มต้น ในโรงงานนี้จะประกอบด้วยขั้นตอนการใช้งานโปรแกรมในการออกแบบวงจรดิจิทัลด้วยโปรแกรม Xilinx 7.1i ตั้งแต่การเข้าโปรแกรม การเขียนชุดคำสั่ง และตรวจสอบความผิดพลาดของชุดคำสั่ง แล้วสุดท้ายจะเป็นการโปรแกรมชุดคำสั่งลงชิพ CPLDs

โรงงานที่ 2 คือ โรงงานที่สอนหลักการออกแบบวงจรลอจิกเกตต่างๆ ในการออกแบบวงจรดิจิทัล

โรงงานที่ 3 คือ โรงงานที่สอนการออกแบบวงจรบวก (Adder) แบบ Half Adder และแบบ Full Adder

โรงงานที่ 4 คือ โรงงานที่สอนการออกแบบวงจรลบ (Subtractor) แบบ Half Subtractor และแบบ Full Subtractor

โรงงานที่ 5 คือ โรงงานการออกแบบวงจรเข้ารหัส (Encoder)

เอกสารนี้เป็นโรงงานที่ 6 คือ โรงงานการออกแบบวงจรถอดรหัส (Decoder) ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- ใบงานที่ 7 คือ ใบงานการออกแบบวงจรมัลติเพล็กซ์เซอร์ (Multiplexer)
 ใบงานที่ 8 คือ ใบงานการออกแบบวงจรดี-มัลติเพล็กซ์เซอร์ (D- Multiplexer)
 ใบงานที่ 9 คือ ใบงานการออกแบบวงจรถอดรหัส BCD ออกทาง 7-Segment
 ใบงานที่ 10 คือ ใบงานการออกแบบวงจร ฟลิป ฟลอป (Flip-Flop)
 ใบงานที่ 11 คือ ใบงานการออกแบบวงจรนับ (Countor)
 ใบงานที่ 12 คือ ใบงานการทดลองเลื่อนข้อมูล (Shift Register Circuit)
 ใบงานที่ 13 คือ ใบงานการทดลองวงจรเปรียบเทียบข้อมูล (Comparator)



บทที่ 4

การทดลองและผลการทดลอง

4.1 กล่าวนำ

ในส่วนของบทนี้จะกล่าวถึงการทดลองและผลการทดลองของชุดฝึกดิจิทัลด้วย CPLDs ไม่ว่าจะเป็นส่วนของแผงวงจรอินเทอร์เฟส แผงวงจร CPLDs ซึ่งจะใช้โปรแกรม Xilinx ISE 7.1i ในการทดลอง

4.2 การทดลองการทำงานของแผงวงจร CPLDs

หน้าที่หลักการทำงานของชุดฝึกดิจิทัลด้วย CPLDs คือ การรับข้อมูลจากคอมพิวเตอร์และส่งข้อมูลทางแผงวงจรเชื่อมต่อ โดยจะทำการติดต่อผ่านทางพอร์ตอนุกรม ดังนั้นในการทดลองแผงวงจรได้ดังนี้

4.2.1 การทดลองติดต่อกับคอมพิวเตอร์

4.2.1.1 ขั้นตอนการทดลอง

1. เชื่อมต่อชุดฝึกดิจิทัลด้วย CPLDs กับพอร์ตอนุกรมของเครื่องคอมพิวเตอร์
2. เปิดโปรแกรม Xilinx ISE 7.1i เพื่อดูผลการเชื่อมต่อ

4.2.1.2 ผลการทดลอง

1. ถ้าหากแผงวงจรชุดฝึกดิจิทัลด้วย CPLDs สามารถใช้งานได้ก็จะสามารถทำการโหลดโปรแกรมใบงานต่างๆ ได้

4.3 การทดลองชุดฝึกดิจิทัลด้วย CPLDs

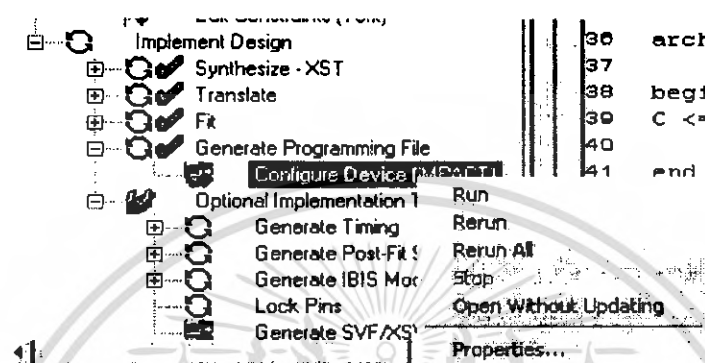
ทำการเขียนโปรแกรมขึ้นมา เช่น โปรแกรมแอนเกตเพื่อที่จะทำการทดลองชุดฝึกโดยพิมพ์ชุดคำสั่งดังรูปที่ 4.1

```
19 -----
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 entity And is
31     (P1A,P1B:IN std_logic=>'U';
32      P1Y:OUT std_logic);
33
34 end And;
35
36 architecture Behavioral of And is
37
38     begin
39         P1Y<=P1A and P1B;
40
41     end Behavioral;
42
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
รูปที่ 4.1 โปรแกรมแอนเกต
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

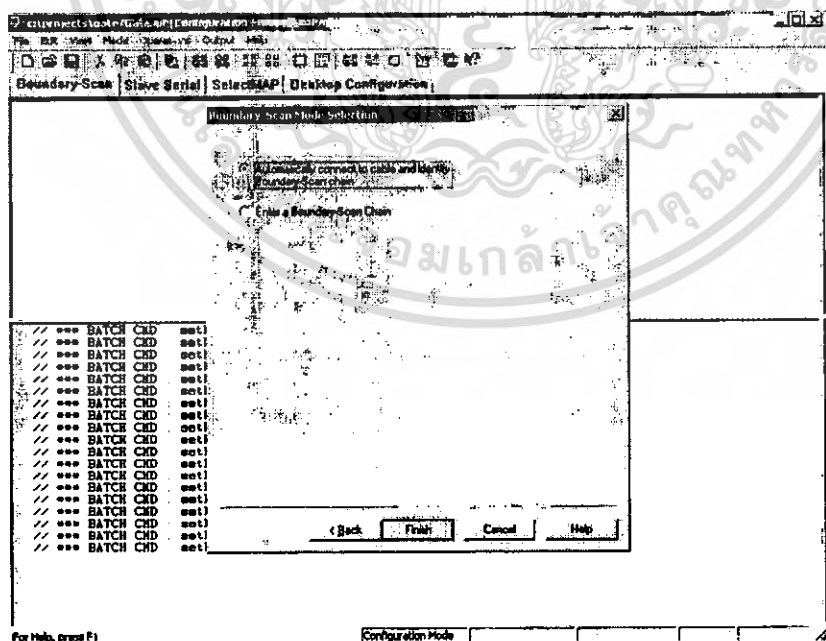
เมื่อพิมพ์ชุดคำสั่งเสร็จแล้วจึงทำการโหลดโปรแกรมลงชุดฝึกเพื่อทำการทดสอบโมดูลต่าง ๆ ที่อยู่บนแผงวงจรโดยทำการโหลดโปรแกรมที่พิมพ์ขึ้นเมื่อข้างต้นลงบนแผงวงจรชุดฝึกดิจิทัล โดยมีขั้นตอนดังนี้

คลิกที่ข้อความ Configure Device เพื่อเข้าสู่ขั้นตอนการโปรแกรม ดังรูปที่ 4.2



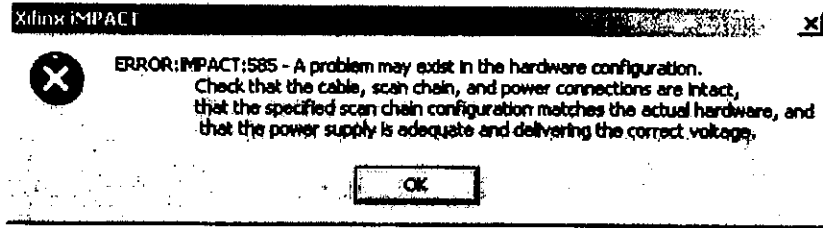
รูปที่ 4.2 การเข้าสู่ขั้นตอนการโปรแกรม

1. จากนั้นโปรแกรมจะเริ่มทำการหาแผงวงจรชุดฝึกดิจิทัลโดยจะมีคำสั่งให้เลือกค้นหาแบบอัตโนมัติและแบบค้นหา ให้เลือกแบบค้นหาอัตโนมัติ ดังรูปที่ 4.3



รูปที่ 4.3 การค้นหาแผงวงจร

เอกสารที่ 2 ตัวโปรแกรมจะทำการค้นหาและเชื่อมต่อแผงวงจรชุดฝึกดิจิทัลถ้าค้นหาไม่พบโปรแกรมจะเกิดการแจ้งเตือน Error โดยโปรแกรมจะแสดงผล ดังรูปที่ 4.4 ลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



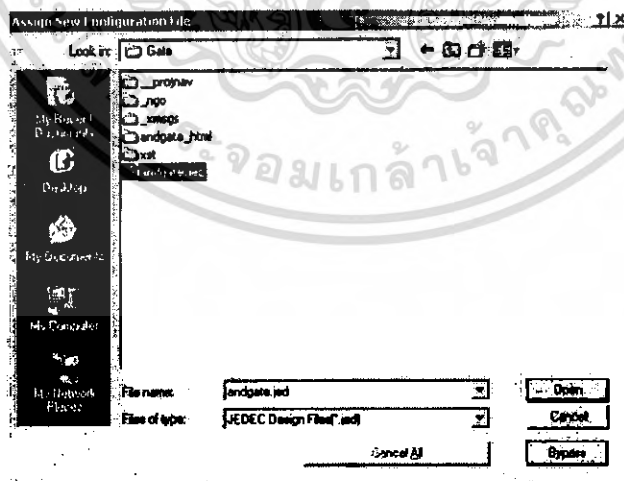
รูปที่ 4.4 แสดงการผิดพลาดเมื่อหาแผงวงจรไม่พบ

3. เมื่อคลิกที่ ok จะขึ้นข้อความว่าหาอุปกรณ์การเชื่อมต่อไม่พบ ดังรูปที่ 4.5 แก้ได้โดยกาตรวจสอบที่ใช้สายเชื่อมต่อระหว่างแผงวงจรกับคอมพิวเตอร์ว่าหลวมหรือไม่



รูปที่ 4.5 แสดงผลว่าไม่พบแผงวงจร

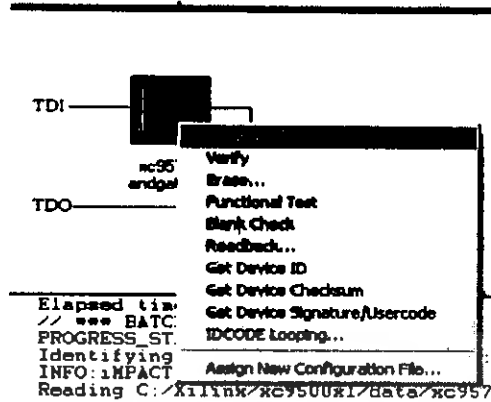
4. เมื่อโปรแกรมสามารถหาแผงวงจรพบจะขึ้นหน้าจอเพื่อให้เลือกโปรแกรมในการทดสอบดังรูปที่ 4.6



รูปที่ 4.6 การเลือกโปรแกรม

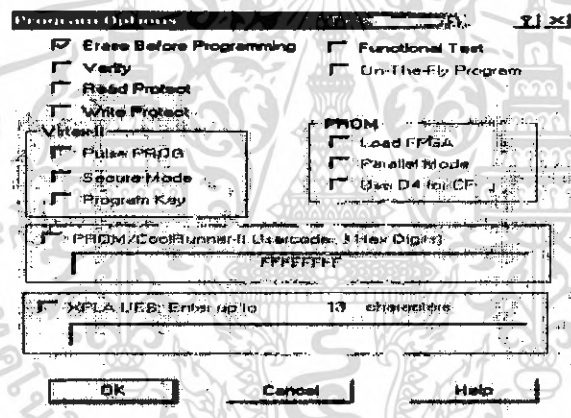
5. หลังจากทำการเลือกโปรแกรมแล้วก็ทำการส่งบันทึกโปรแกรมลงแผงวงจรชุดฝึกติดจอสั่งรูปที่ 4.7

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



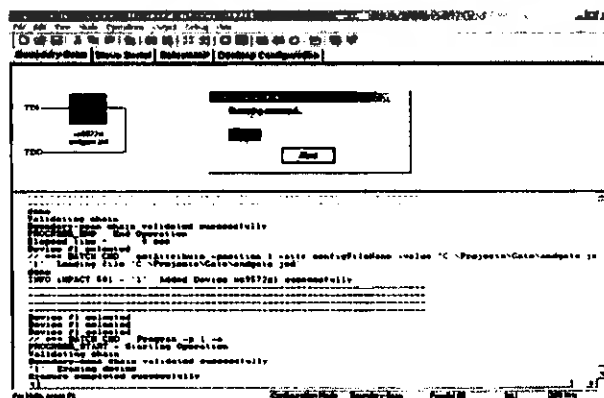
รูปที่ 4.7 การบันทึกคำสั่ง

6. แสดงขั้นตอนการยืนยันการโหลดโปรแกรม ดังรูปที่ 4.8



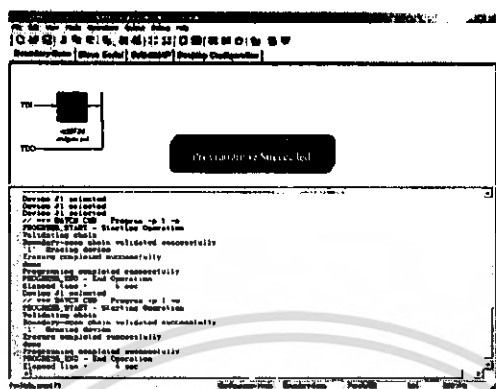
รูปที่ 4.8 แสดงขั้นตอนการยืนยันการโหลดโปรแกรม

7. การประมวลผลการโหลดโปรแกรม ดังรูปที่ 4.9



เอกสารนี้เป็นเอกสารที่สงวนไว้รูปที่ 4.9 การประมวลผลการโหลดโปรแกรม อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จึงเริ่มทำการทดลองแผงวงจรชุดฝึกดิจิตอลแต่ละโมดูล



รูปที่ 4.10 แสดงผลการไหลตเสร็จสมบูรณ์

เมื่อโยก Slide Switch ขึ้นทั้งสองตัว LED Logic-Monitor จะติดซึ่งแสดงว่าไมดูลทั้งสองสามารถใช้งานได้ตามปกติ

1. การทดสอบโมดูล LED Logic-Monitor ซึ่งสามารถทดสอบควบคู่ไปกับโมดูล Slide Switch ได้ในเวลาเดียวกัน โดยใช้โปรแกรมของ แอนเกด ที่พิมพ์ไว้ดังรูปที่ 4.2 ในการทดสอบโดยใช้ Slide Switch เป็นอินพุตและ LED Logic-Monitor เป็นเอาต์พุตในการแสดงผลเพื่อการทดสอบ

ผลการทดลอง

เมื่อโยก Slide Switch ขึ้นทั้งสองตัว LED Logic-Monitor จะติดซึ่งแสดงว่าไมโครทั้งสองสามารถใช้งานได้ตามปกติ

2. การทดสอบโมดูล LED Logic-Monitor แบบสองสถานะซึ่งสามารถทดสอบควบคู่ไปกับโมดูล

Switch แบบกดติดปล่อยดับได้ในเวลาเดียวกัน โดยใช้โปรแกรมของ แอนเกด ที่พิมพ์ไว้ดังรูปที่ 4.2 ในการทดสอบโดยใช้ Switch กดติดปล่อยดับเป็นอินพุตและ LED Logic-Monitor แบบ 2 สถานะเป็นเอาต์พุตในการแสดงผลเพื่อการทดสอบ ซึ่งหลอดแสดงผลแบบ 2 สถานะปกติจะเป็นสีแดงและเมื่อมีสัญญาณมากกระตุ้นจะเปลี่ยนเป็นสีเขียว

ผลการทดลอง

เมื่อกดสวิทช์กดติดปล่อยดับค้างไว้ทั้งสองตัวหลอด LED จะเปลี่ยนสถานะจากสีแดงเป็นสีเขียวแสดงว่า
โมดูลทั้งสองสามารถใช้งานได้ตามปกติ

3. Segment x Digit โมเดล เป็นโมเดลที่ใช้แสดงผลเป็นตัวเลขหรือตัวอักษรได้ ทำการทดลองโมเดลชุดนี้

โดยใช้โปรแกรม BCD 8421 และใช้ Dip Switch ช่วยในการกำหนดสัญญาณทดลองโดยพิมพ์โปรแกรมดังรูปที่ 4.11

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity bcd7 is
  port(
    D:in STD_LOGIC_VECTOR(3 downto 0);
    Y:out STD_LOGIC_VECTOR(6 downto 0));
  end bcd7;
architecture Behavioral of bcd7 is
begin
  with D Select
    Y<="1111001" when "0001",
      "0100100" when "0010",
      "0110000" when "0011",
      "0011001" when "0100",
      "0010010" when "0101",
      "0000011" when "0110",
      "1111000" when "0111",
      "0000000" when "1000",
      "0011000" when "1001",
      "1000000" when others;
end Behavioral;

```

รูปที่ 4.11 โปรแกรม BCD 8421

ผลการทดลอง

7 Segment x 4 Digit แสดงผลได้ตามปกติโดยมีการแสดงผลตามตารางที่ 4.2

ตารางที่ 4.1 ผลการทดลองไมโคร 7 Segment

D	C	B	A	a	b	c	d	e	f	g
0	0	0	0	ดับ	ดับ	ดับ	ดับ	ดับ	ดับ	ติด
0	0	0	1	ติด	ดับ	ดับ	ติด	ติด	ติด	ติด
0	0	1	0	ดับ	ดับ	ติด	ดับ	ดับ	ติด	ดับ
0	0	1	1	ดับ	ดับ	ดับ	ดับ	ติด	ติด	ดับ
0	1	0	0	ติด	ดับ	ดับ	ติด	ติด	ดับ	ดับ
0	1	0	1	ดับ	ติด	ดับ	ดับ	ติด	ดับ	ดับ
0	1	1	0	ติด	ติด	ดับ	ดับ	ดับ	ดับ	ดับ
0	1	1	1	ดับ	ดับ	ดับ	ติด	ติด	ติด	ติด
1	0	0	0	ดับ	ดับ	ดับ	ดับ	ดับ	ดับ	ดับ
1	0	0	1	ดับ	ดับ	ดับ	ติด	ติด	ดับ	ดับ

4. Buzzer โมดูล เป็นโมดูลที่ใช้กำเนิดเสียงเมื่อมีสัญญาณป้อนให้กับโมดูลนี้จะเกิดเสียงดังขึ้น

ผลการลอง

เมื่อใช้ Slide Switch ร่วมกับโปรแกรม แอนเกต โดยให้ Slide Switch เป็นตัวป้อนสัญญาณให้กับ Buzzer เมื่อโยก Slide Switch ขึ้นพร้อมกันทั้งสองตัวสัญญาณจะถูกส่งไปที่ Buzzer ทำให้เกิดเสียงดังขึ้นแสดงว่า Buzzer ใช้งานได้ตามปกติ



บทที่ 5

บทสรุป

5.1 สรุป

ชุดฝึกดิจิทัลด้วย CPLDs เป็นชุดฝึกที่ประกอบด้วย ชุดทดลองที่ใช้สำหรับการออกแบบวงจรดิจิทัล โปรแกรมที่ใช้ในการออกแบบวงจรดิจิทัล และไมงานการทดลองชุดฝึกดิจิทัลด้วย CPLDs จะใช้บอร์ดทดลองรุ่น CPLD Explorer XC9572XL ซึ่งภายในบอร์ดจะมีส่วนที่ใช้ในการทดลองแบ่งออกเป็น 2 ส่วน คือ ส่วนทางอินพุต และส่วนเอาต์พุต โดยที่ส่วนอินพุต จะมีสวิตช์แบบ Slide switch 4 ตัว Switch Button 6 ตัว ส่วนทางเอาต์พุตมี 7-Segment แบบคอมมอนแคโทด 4 หลัก LED Logic Monitor แบบ 1 สถานะ และแบบ 2 สถานะ และส่วนไมงานการทดลองแบ่งออกเป็น 13 ไมงาน ประกอบด้วย ชุดการใช้งานโปรแกรมจำนวน 1 ไมงาน ชุดการแสดงผลทางแอลอีดีจำนวน 7 ไมงาน ชุดแสดงผลโดยการจำลองการทำงานทางโปรแกรมจำนวน 5 ไมงาน

5.2 ปัญหาและวิธีการแก้ไข

จากการดำเนินการสร้างและทดสอบโครงงานปรากฏว่ามีปัญหาเกิดขึ้นหลายประการ ซึ่งสามารถสรุปได้ดังนี้

1. เนื่องจากช่วงแรกได้ใช้โปรแกรมในการออกแบบเป็น ISE Foundation 2.1i แล้วเกิดปัญหาคือไม่สามารถทำงานร่วมกับ Windows XP ได้
วิธีการแก้ไข โดยการเปลี่ยนโปรแกรม ด้วย Xilinx ISE 7.1i ในการออกแบบวงจรดิจิทัลแทน
2. ความล่าช้าในการศึกษาโปรแกรม Xilinx ISE 7.1i เนื่องจากเป็นโปรแกรมลิขสิทธิ์ ไม่มีหนังสือที่ใช้สำหรับการค้นคว้าที่เป็นภาษาไทยต้องแปลจากคู่มือการใช้งานจากเว็บไซต์ผู้ผลิต
วิธีการแก้ไข ทำการศึกษาจากเว็บไซต์ของผู้ผลิตโปรแกรม และจากเว็บไซต์อื่น แล้วทำการฝึกปฏิบัติให้ละเอียดทีละขั้นตอน จากขั้นตอนการใช้โปรแกรมเบื้องต้น จนถึงขั้นตอนการโปรแกรมลงตัวไอซี เมื่อมีความชำนาญแล้วทำคู่มือการใช้โปรแกรมขึ้นมาเองเพื่อที่จะได้สะดวกกับผู้ที่ศึกษาการใช้โปรแกรมนี้
3. การเขียนชุดคำสั่งในแต่ละไมงานบางส่วนยังไม่สัมพันธ์กับส่วนของการแสดงผลเนื่องจากผลลัพธ์ที่ได้ยังไม่ตรงไปตามตารางความจริงในทางทฤษฎี
วิธีการแก้ไข ทำการปรับปรุงโปรแกรมในส่วนที่มีปัญหาให้สามารถทำงานกับส่วนของการแสดงผลได้

4. การจำลองการทำงานในส่วนของโปรแกรมที่ทำงานนั้น เมื่อทำการจำลองการทำงานแล้วปรากฏว่ามีการ Error เกิดขึ้น วิธีการแก้ไข โดยการแก้ไขในส่วนที่ Error เนื่องจากทำการประกาศตัวแปรซ้ำกับในส่วนของโปรแกรมที่มีอยู่แล้ว

5.3 แนวทางการพัฒนา

1. ชุดฝึกดิจิทัลด้วย CPLDs นี้ ใช้ไอซีเบอร์ XC9572 ซึ่งมีจำนวนเกตไม่มากพอที่จะออกแบบวงจรที่ซับซ้อนได้ เช่น วงจรนาฬิกา การทำ CPU
2. โปรแกรมการทดลองต้องเพิ่มในส่วนของการแสดงผลทาง ตรวจสอบสัญญาณ สแต็กบิงมอเตอร์ การควบคุมอุปกรณ์ด้วยรีเลย์ เป็นต้น
3. โปรแกรม Xilinx ISE ตอนนี้ทางผู้ผลิตได้พัฒนาโปรแกรมไปถึงเวอร์ชัน Xilinx ISE 9.2i จะสามารถนำไปออกแบบวงจรดิจิทัลได้หลากหลายมากขึ้นเพราะโปรแกรมได้เพิ่มรูปแบบการทำงานที่ยืดหยุ่นมากขึ้น



บรรณานุกรม

ขรรค์ชัย ตูลละสกุล. 2546 . การใช้งานโปรแกรมออกแบบวงจรรวมดิจิทัล. กรุงเทพฯ :

บริษัท ซีเอ็ดดูเคชั่น จำกัด (มหาชน).

ขรรค์ชัย ตูลละสกุล. 2546 . การออกแบบวงจรดิจิทัลภายใน FPGAs และ CPLDs. กรุงเทพฯ :

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี.

ชำนาญ ปัญญาโส และวัชรกร หนูทอง. 2547. ภาษา VHDL สำหรับการออกแบบวงจรดิจิทัล. กรุงเทพฯ :

บริษัท ซีเอ็ดดูเคชั่น จำกัด (มหาชน).

ธวัชชัย เลื่อนฉวี และอนุรักษ์ เกื่อนศิริ. 2543. ดิจิตอลเทคนิค เล่ม 1. กรุงเทพฯ :

มิตรนราการพิมพ์.

ธวัชชัย เลื่อนฉวี. 2543. ดิจิตอลเทคนิค เล่ม 2. กรุงเทพฯ : มิตรนราการพิมพ์.

วัชรินทร์ เคารพ. 2546 . เรียนรู้และเข้าใจ CPLD XC95108. กรุงเทพฯ : บริษัท อีทีที จำกัด.



ภาคผนวก ข ใบงานการทดลอง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ใบงานที่ 1

การใช้งานโปรแกรม Xilinx ISE 7.1i เบื้องต้น

วัตถุประสงค์

1. เพื่อให้นักศึกษาสามารถใช้โปรแกรม Xilinx ISE 7.1i ได้
2. เพื่อให้นักศึกษาสามารถจำลองการทำงานของวงจรที่ทำการออกแบบได้โดยโปรแกรม Xilinx ISE 7.1i ได้
3. เพื่อให้นักศึกษาสามารถทำการโปรแกรมไอซีด้วยโปรแกรม Xilinx ISE 7.1i ได้

เครื่องมือสำหรับการทดลอง

1. ระบบไมโครคอมพิวเตอร์
2. โปรแกรม Xilinx ISE 7.1i
3. สายดาร์วินโหลด
4. บอร์ดทดลอง CPLD เบอร์ XC9572

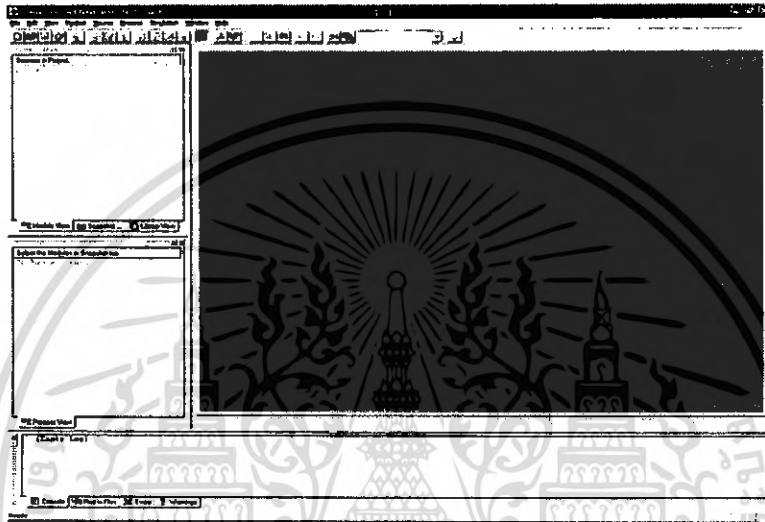
ทฤษฎีพื้นฐาน

โปรแกรม Xilinx ISE 7.1i เป็นวงจรที่ใช้ออกแบบวงจรรวมแบบดิจิทัล สามารถพัฒนางจรได้ด้วยการเลือกตัวลอจิกเกิดนำมาต่อกันแล้วลากสายเชื่อมต่อวงจรในลักษณะของการเขียนผังรูปวงจร (Schematic) การเขียนภาษาอธิบายฮาร์ดแวร์ หรือที่เรียกกันว่าเฮชดีแอล (HDL: Hardware Description Language) และการเขียนไดอะแกรมการเปลี่ยนสถานะ (SFM: Specify Finite State Maching) โปรแกรม Xilinx ISE 7.1i มีความสามารถในการพัฒนางจรรวมแบบวงจรดิจิทัลร่วมกับชิพไอซี CPLDs และชิพไอซี FPGAs ของบริษัท Xilinx

ขั้นตอนการทดลอง

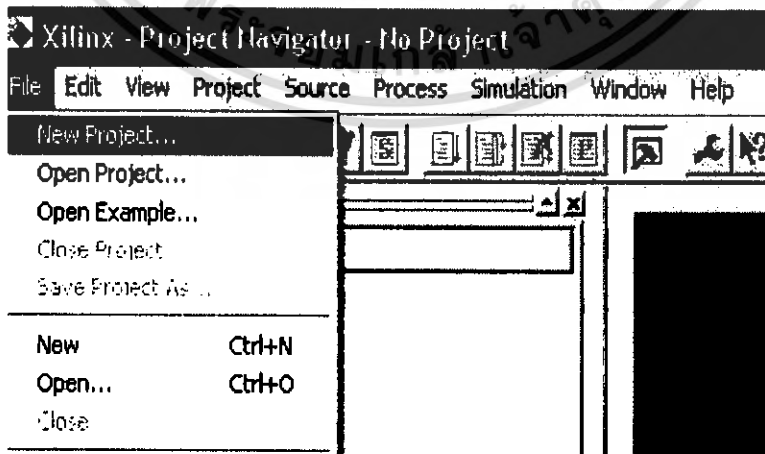


1. ดับเบิลคลิกที่ไอคอน Xilinx ISE 7.1i ที่หน้าจอคอมพิวเตอร์ จากนั้นโปรแกรมจะเข้าสู่หน้าจอหลักของโปรแกรม Xilinx ISE 7.1i ดังรูปที่ ข.1



รูปที่ ข.1 หน้าจอหลักของโปรแกรม Xilinx ISE 7.1i

2. คลิกที่ File เลือกที่ New Project ดังรูปที่ ข.2



รูปที่ ข.2 การสร้าง New Project

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3. ให้ตั้งชื่อ Project Name: เลือกที่ Next ดังรูปที่ ข.3

New Project

Enter a Name and Location for the Project

Project Name: Project Location: ...

Select the type of Top-Level module for the Project

Top-Level Module Type:

< Back Next > Cancel Help

รูปที่ ข.3 หน้าตา New Project

4. ให้ตั้งค่า New Project ดังรูปที่ ข.4 แล้วทำการ Next

New Project

Select the Device and Design Flow for the Project

Property Name	Value
Device Family	xc9500 CPLDs
Device	xc9572
Package	PC44
Speed Grade	-15
Top-Level Module Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	Modelsim
Generated Simulation Language	VHDL

< Back Next > Cancel Help

รูปที่ ข.4 การตั้งค่าในส่วนของ Select the Device and Design Flow for the Project

5. ขั้นตอน Create a New Source ดังรูปที่ ข.5 ให้ทำการ Next

	Source File	Type	New Source...
1			Remove

Create a new source to add to the project (optional). Only one new source can be specified now. Additional new sources can be added after project creation using the "Project->New Source" command.

Existing sources can be added on the next page.

< Back Next > Cancel Help

รูปที่ ข.5 Create a New Source

6. ขั้นตอน Add Existing Sources ดังรูปที่ ข.6 ให้ทำการ Next เพื่อไปขั้นตอนต่อไป

	Source File	Type	Copy to Project		Add Source...
1			☐		Remove
2			☐		
3			☐		
4			☐		

Add existing sources to the project (optional). Additional sources can be added after project creation using the "Project->Add Source" or "Project->Add Copy of Source" commands.

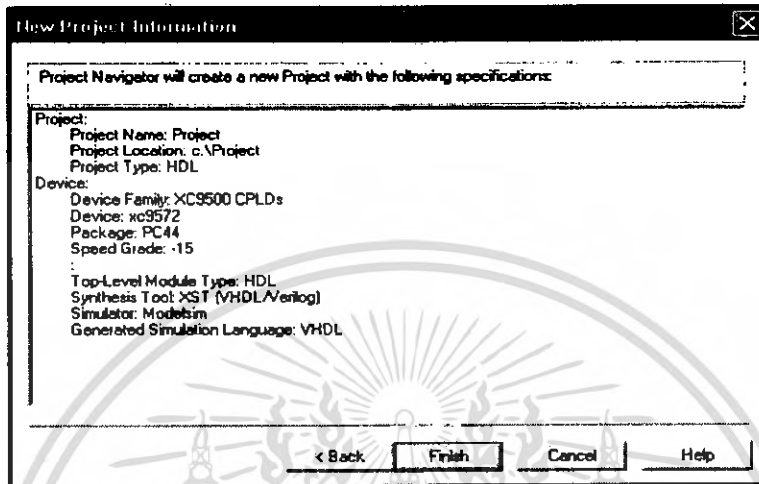
< Back Next > Cancel Help

รูปที่ ข.6 Add Existing Sources

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

7. ขั้นตอน Project Navigator will create a new Project with the following specification:

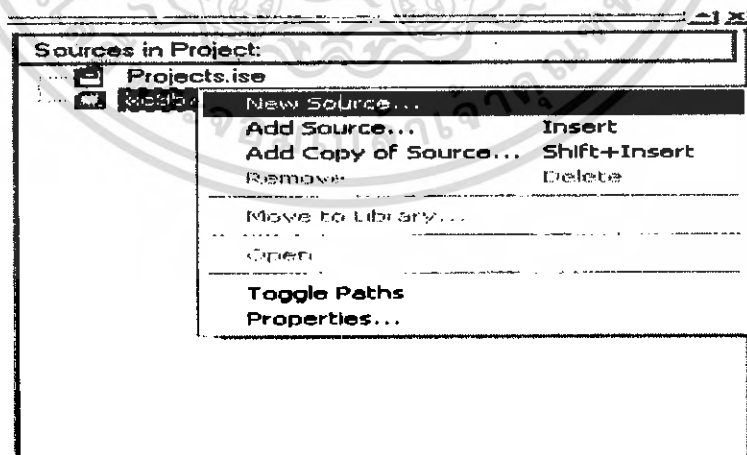
ดังรูปที่ ข.7 ให้ทำการ Finish เพื่อจบขั้นตอนการสร้าง New Project



รูปที่ ข.7 Project Navigator will create a new Project with the following specification:

ขั้นตอนการสร้าง Module ของภาษา VHDL

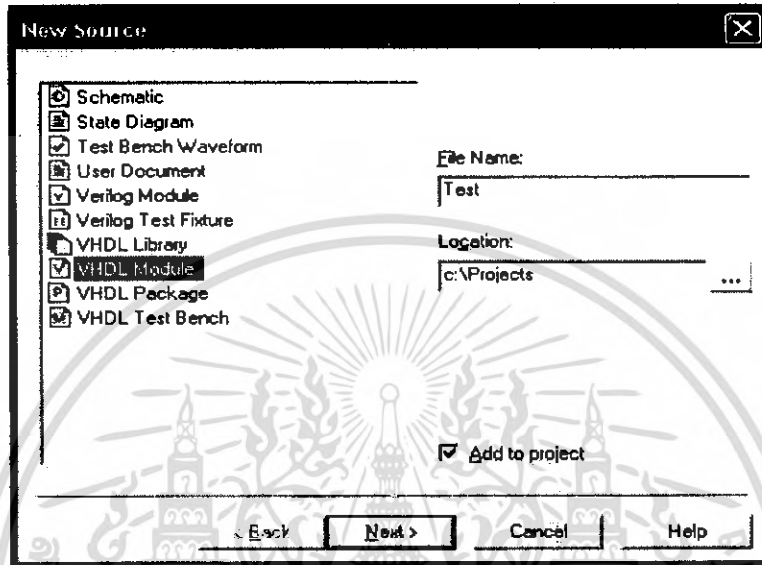
1. ให้ทำการคลิกขวาที่ XC9572-15pc44 เลือกที่ New Source... เพื่อที่จะทำการสร้าง Module ภาษา VHDL ดังรูปที่ ข.8



รูปที่ ข.8 ขั้นตอนการสร้าง Module ของภาษา VHDL

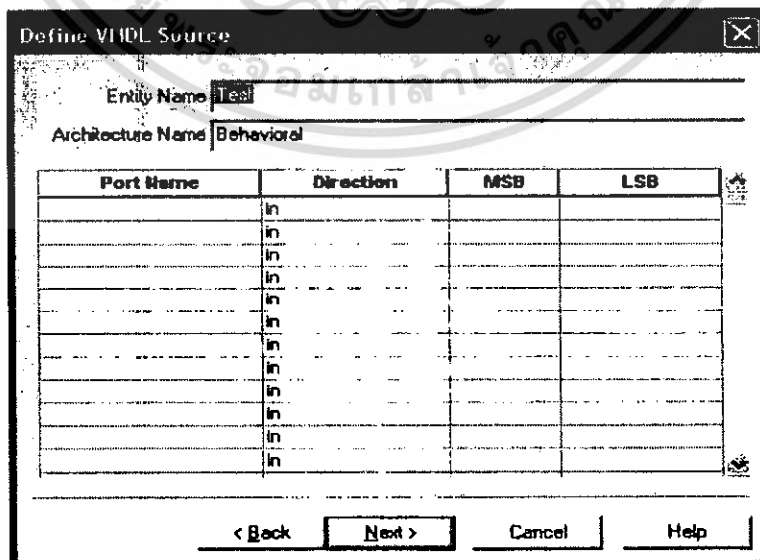
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. จากรูปที่ ข.9 ให้ทำการเลือกที่ VHDL Module ส่วน File Name ให้ตั้งชื่ออะไรก็ได้ ถ้าจะให้จำง่ายควรตั้งชื่อให้เหมาะสมกับงานที่ทำ แล้วทำการ Next

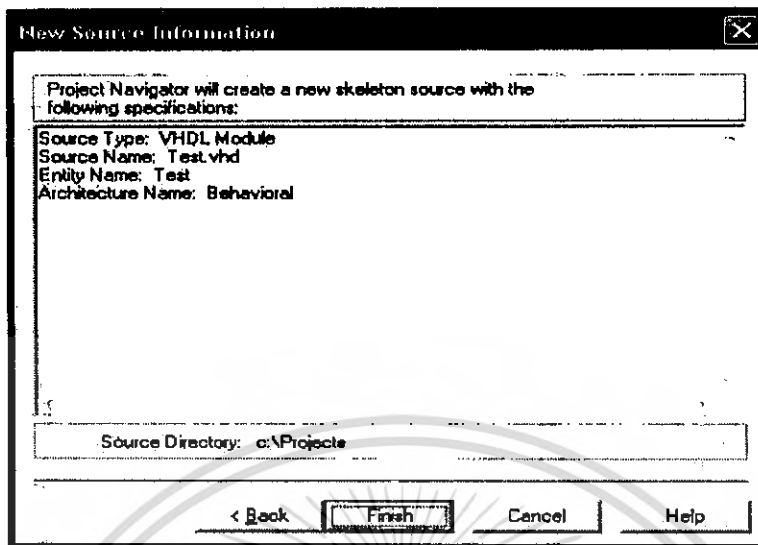


รูปที่ ข.9 แสดงขั้นตอนการสร้าง Module VHDL

3. จากรูปที่ ข.10 เป็นขั้นตอนการกำหนดขา อินพุต และ เอาต์พุต ขั้นตอนนี้ให้ทำการ Next โดยที่ยังไม่ต้องกำหนด ส่วนรูปที่ ข.11 เป็นการสิ้นสุดขั้นตอนการสร้าง Module VHDL ให้ทำการ Finish



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับรูปที่ ข.10 ขั้นตอนการกำหนดขา อินพุต และ เอาต์พุต นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ ข.11 ขั้นตอนสุดท้ายการสร้าง Module VHDL

4. ตัวโปรแกรมจะเปิดหน้าต่างเพื่อทำการพิมพ์ภาษา VHDL ดังรูปที่ ข.12 ส่วนรูปที่ ข.13 เป็นส่วนที่ต้องเขียนโปรแกรมเพิ่มเข้า

```

1
2
3 -- Company:
4 -- Engineer:
5
6 -- Create Date:    17:30:09 03/31/07
7 -- Design Name:
8 -- Module Name:    Test - Behavioral
9 -- Project Name:
10 -- Target Device:
11 -- Tool versions:
12 -- Description:
13
14 -- Dependencies:
15
16 -- Revision:
17 -- Revision 0.01 - File Created
18 -- Additional Comments:
19
20
21 library IEEE;
22 use IEEE.STD_LOGIC_1164.ALL;
23 use IEEE.STD_LOGIC_ARITH.ALL;
24 use IEEE.STD_LOGIC_UNSIGNED.ALL;
25
26 -- Uncomment the following library declaration if instantiating
27 -- any Xilinx primitives in this code.
28 --library UNISIM;
29 --use UNISIM.VComponents.all;
30
31
32 --entity Test is
33 --end Test;
34
35
36 --architecture Behavioral of Test is
37
38
39 begin
40
41
42 end Behavioral;
43

```

รูปที่ ข.12 หน้าต่างเพื่อทำการพิมพ์ภาษา VHDL

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.vComponents.all;

entity Test
    ส่วนที่ต้องพิมพ์เพิ่ม
end Test;

architecture Behavioral of Test is
    begin
        ส่วนที่ต้องพิมพ์เพิ่ม
    end Behavioral;

```

รูปที่ ๑.13 ส่วนของโปรแกรมที่ต้องพิมพ์เพิ่ม

ขั้นตอนการ วิเคราะห์ชุดคำสั่ง

1. ทำการเขียนโปรแกรม VHDL ลงไปในส่วนที่ต้องพิมพ์เพิ่ม ในที่นี้จะยกตัวอย่าง VHDL จำลองการทำงานของลอจิกเกต ชนิดแอนด์เกตดังรูปที่ ๑.14

```

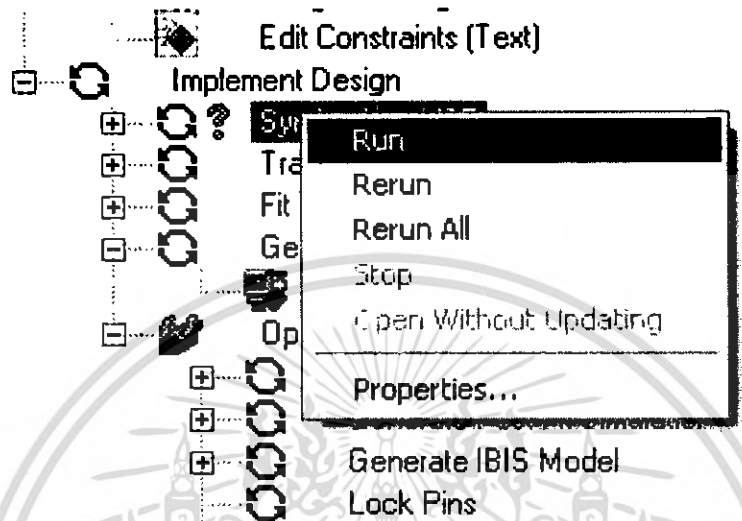
29
30 entity Test is
31
32     Port (P1A,P1B: IN std_logic:='U';
33          P1Y: OUT std_logic);
34
35 end Test;
36
37 architecture Behavioral of Test is
38
39 begin
40
41     P1Y<=P1A and P1B;
42
43 end Behavioral;

```

รูปที่ ๑.14 ส่วนที่ทำการพิมพ์โปรแกรม จำลองการทำงาน แอนด์เกต

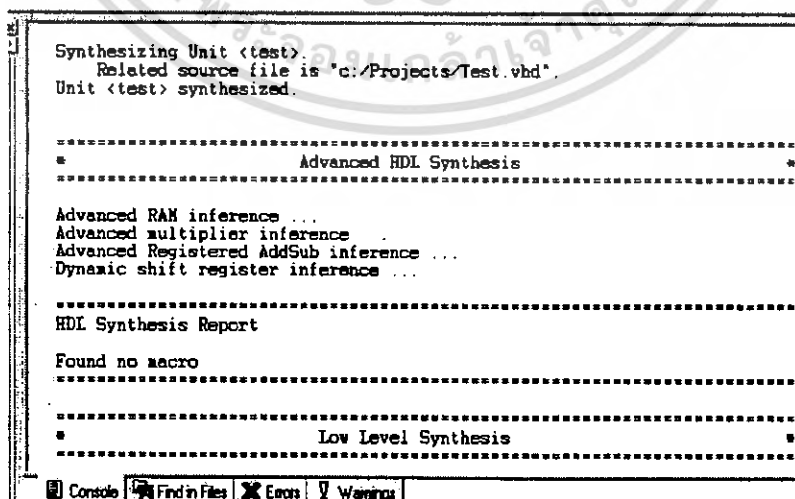
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. เมื่อทำการพิมพ์ชุดคำสั่งเพิ่มเข้าไปแล้วขั้นตอนต่อไปคือทำการ Synthesize-XST เพื่อทำการตรวจสอบชุดคำสั่งที่ทำการพิมพ์เพิ่มว่าถูกต้องหรือไม่โดยทำการคลิกขวาแล้วคลิก Run ดังรูปที่ ข.15



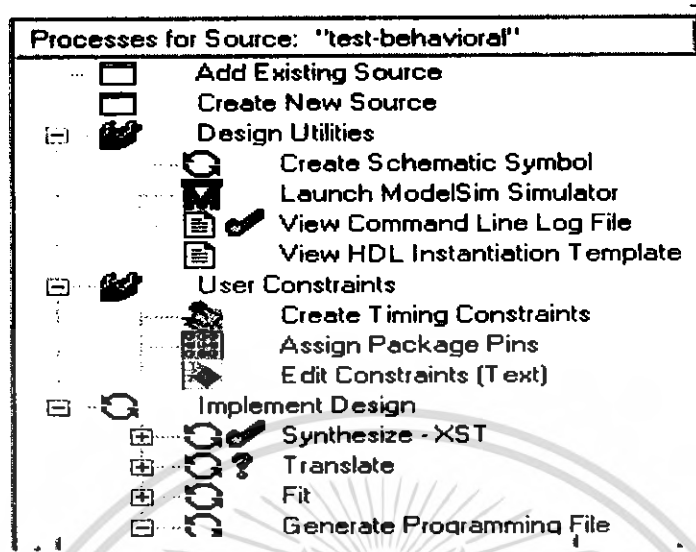
รูปที่ ข.15 ขั้นตอนการทำ Synthesize-XST

3. เมื่อทำการ Synthesize-XST จะปรากฏขั้นตอนการวิเคราะห์ชุดคำสั่งว่ามีถูกต้องหรือไม่ ถ้าไม่ปรากฏข้อความผิดพลาดเพื่อให้ทำการเช็ค ถ้าชุดคำสั่งถูกต้อง จะมีเครื่องหมายถูกปรากฏที่หน้า Synthesize-XST ดังรูปที่ ข.16 และรูปที่ ข.17



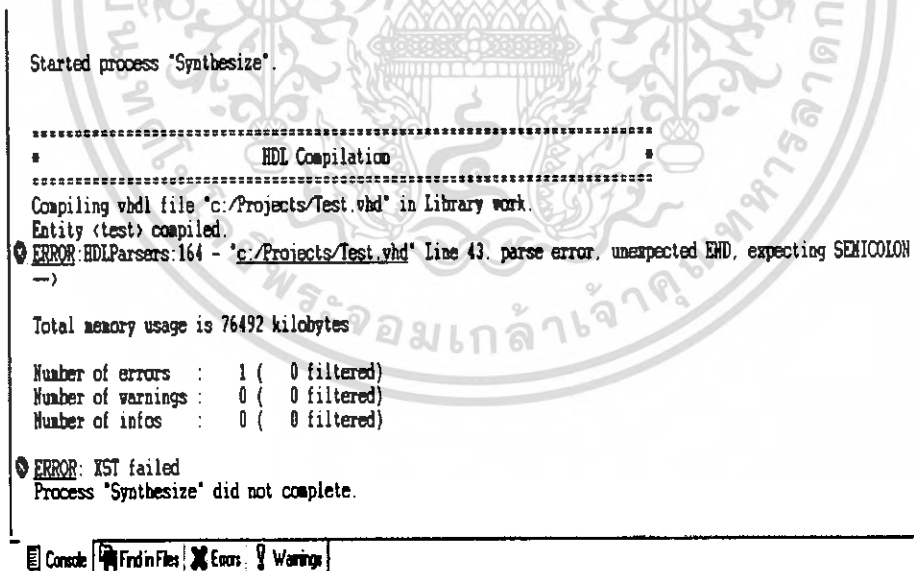
รูปที่ ข.16 ขั้นตอนการวิเคราะห์ชุดคำสั่ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่นิยมนำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

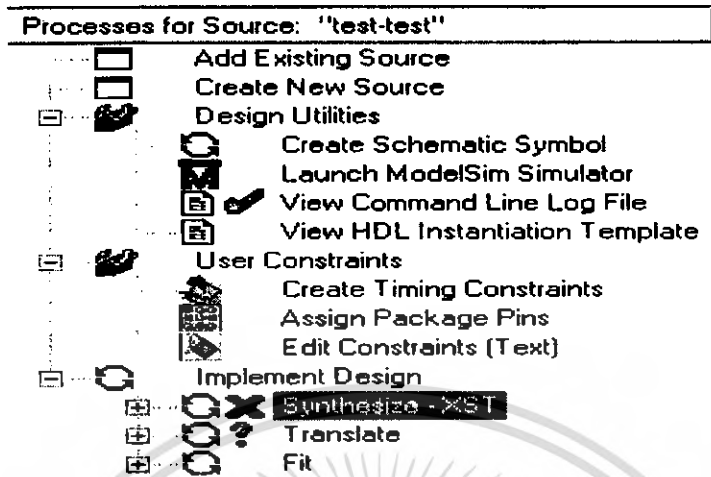


รูปที่ ข.17 เมื่อไม่มีข้อผิดพลาดจะมีเครื่องหมายถูกปรากฏขึ้น

4. ส่วนรูปที่ ข.18 และรูปที่ ข.19 จะแสดงขั้นตอนวิเคราะห์ที่มีความผิดพลาดเกิดขึ้น



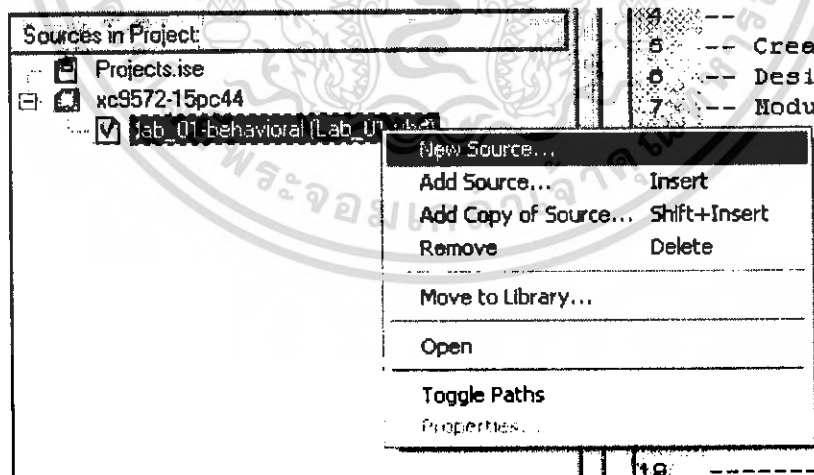
รูปที่ ข.18 การวิเคราะห์ชุดคำสั่งและข้อผิดพลาดเกิดขึ้น



รูปที่ ข.19 เมื่อพบข้อผิดพลาดจะมีเครื่องหมายกากบาทปรากฏขึ้น

ขั้นตอนจำลองการทำงาน

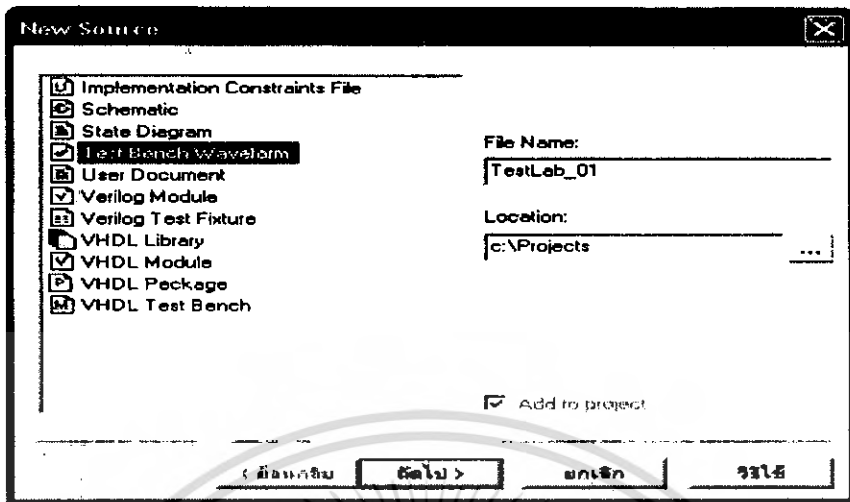
1. ในขั้นตอนนี้จะกล่าวถึงการจำลองการทำงานของโปรแกรมก่อนที่จะทำการโปรแกรมลงตัวไอซี ทำได้ดังนี้
ให้คลิกขวาที่โปรเจกต์ที่เราต้องการที่จะจำลองการทำงานแล้วเลือกที่ New source... ดังรูปที่ ข.20



รูปที่ ข.20 ขั้นตอนการทำ New source...

2. ทำการเลือก Test Bench Waveform แล้วตั้งชื่อโปรเจกต์ให้เหมาะสมกับงานที่ทำดังรูปที่ ข.21
แล้วทำการกด next

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



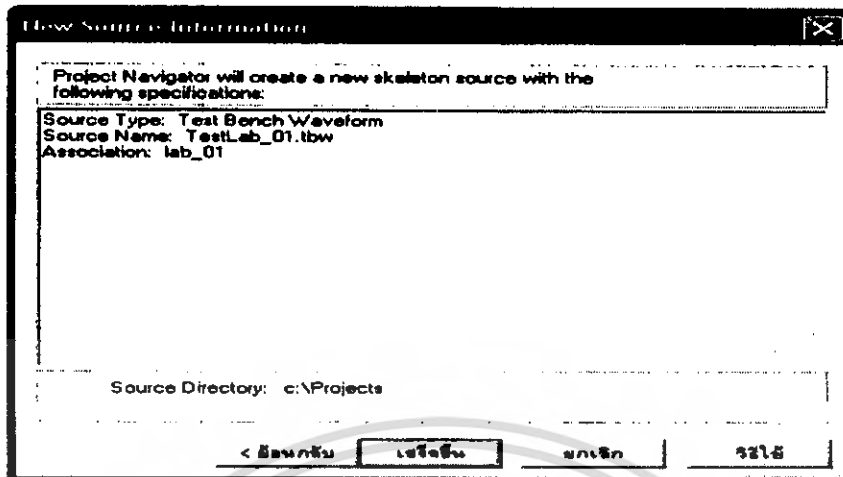
รูปที่ ข.21 การเลือก Test Bench Waveform เพื่อทำการจำลองการทำงาน

3. จากรูปที่ ข.22 ให้ทำการ Next เพื่อไปขั้นตอนต่อไป



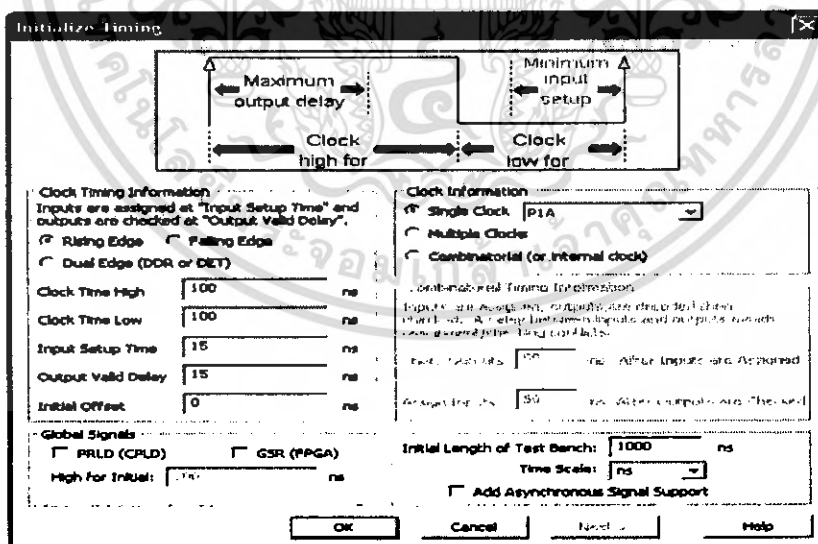
รูปที่ ข.22 ขั้นตอนเลือก Source File เลือกที่ Next

4. จากรูปที่ ข.23 เป็นขั้นตอนสุดท้ายที่แสดงรายละเอียดในการทำการจำลองการทำงานให้ทำการคลิก Finish เพื่อเป็นการสิ้นสุดการตั้งค่าในส่วนนี้



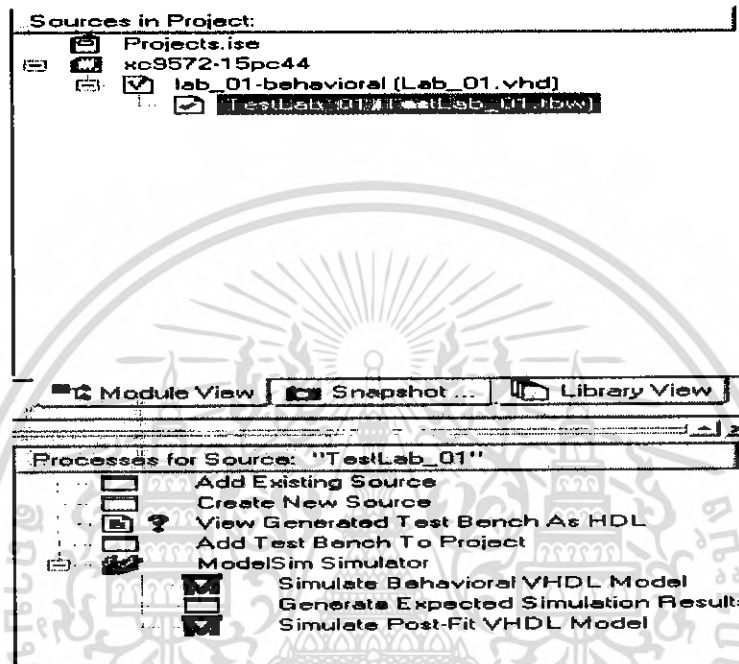
รูปที่ ข.23 ขั้นตอนสุดท้ายในการกำหนดค่าการจำลองการทำงาน

5. จากรูปที่ ข.23 เป็นการเลือกสัญญาณ อินพุตที่จะทำการป้อนเข้ามา ถ้าเลือก Single Clock หมายถึงจะใช้สัญญาณ Clock สัญญาณเดียวเข้าที่ขา a แต่ในกรณีที่ไม่ต้องป้อนสัญญาณ Clock เข้ามาที่ขา a โดยให้เลือกที่ Combinatorial ดังรูปที่ ข.24

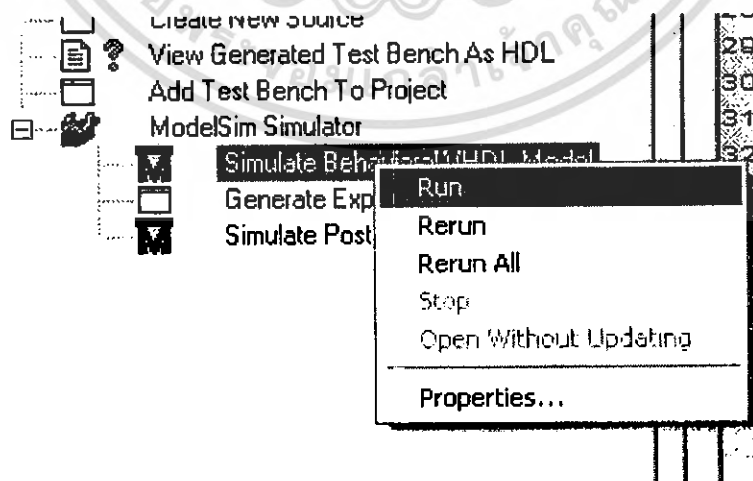


รูปที่ ข.24 การเลือก Single Clock

7. ขั้นตอนต่อไปให้ทำการคลิก ให้แถบนำเงินตรง TestLab แล้วให้สังเกตข้างล่างจะเห็นได้ว่าจะมี ส่วนที่จะให้ทำการจำลองการทำงานคือ Simulate Behavioral VHDL Model จากรูปที่ ข.27 ส่วนรูปที่ ข.28 เป็นการแสดงถึงขั้นตอนการคลิก Run เพื่อสั่งให้โปรแกรมเริ่มการจำลองการทำงาน



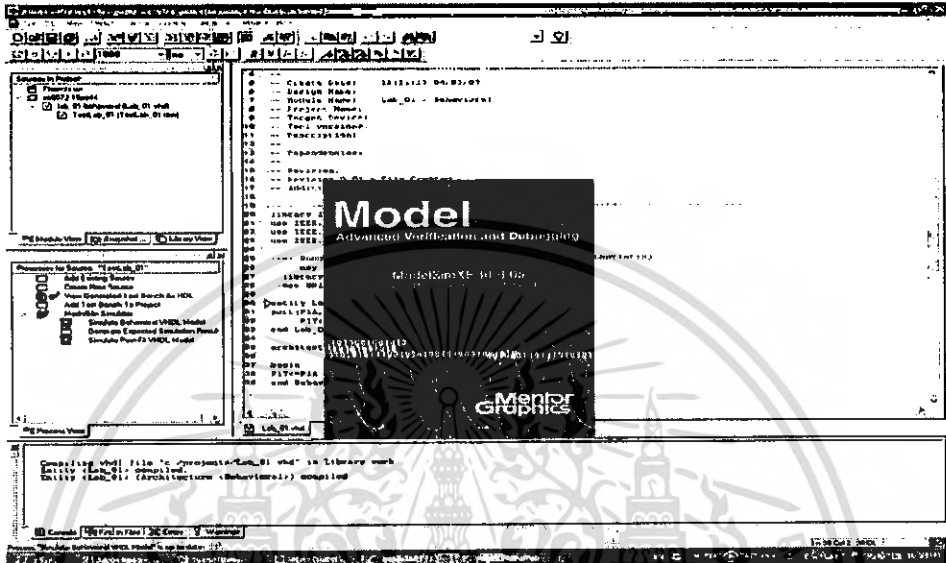
รูปที่ ข.27 ขั้นตอนการสั่งการจำลองการทำงาน



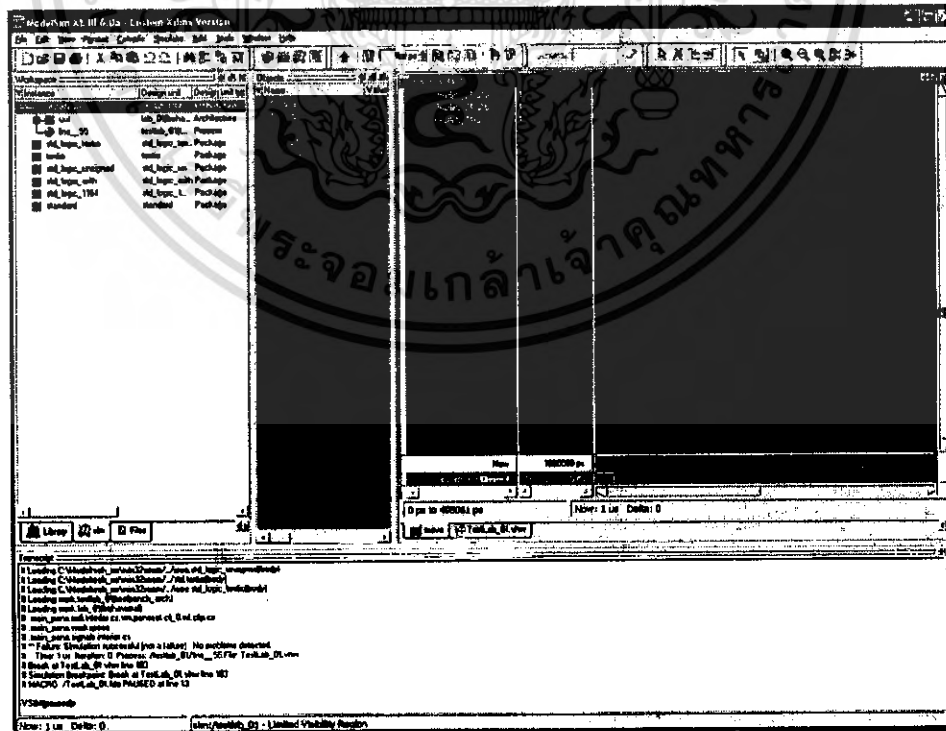
รูปที่ ข.28 ขั้นตอนการสั่ง Run ให้โปรแกรมเริ่มการจำลองการทำงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

8. รูปที่ ข.29 โปรแกรมเริ่มทำการเข้าสู่การจำลองการทำงานโดยจะเข้าจำลองการทำงานในตัวโปรแกรม ModelSim ส่วนรูปที่ ข.30 แสดงถึงรูปคลื่นที่ได้ออกมาจากการทำการตั้งค่า



รูปที่ ข.29 โปรแกรมเริ่มทำการจำลองการทำงาน

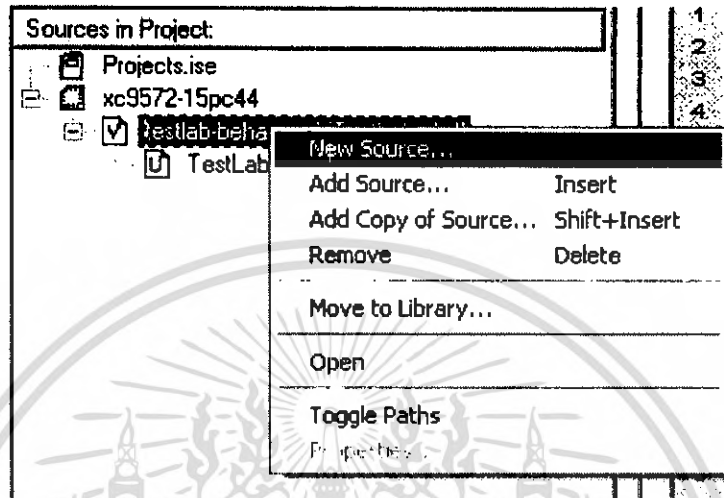


รูปที่ ข.30 รูปคลื่นที่ได้จากการจำลองการทำงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

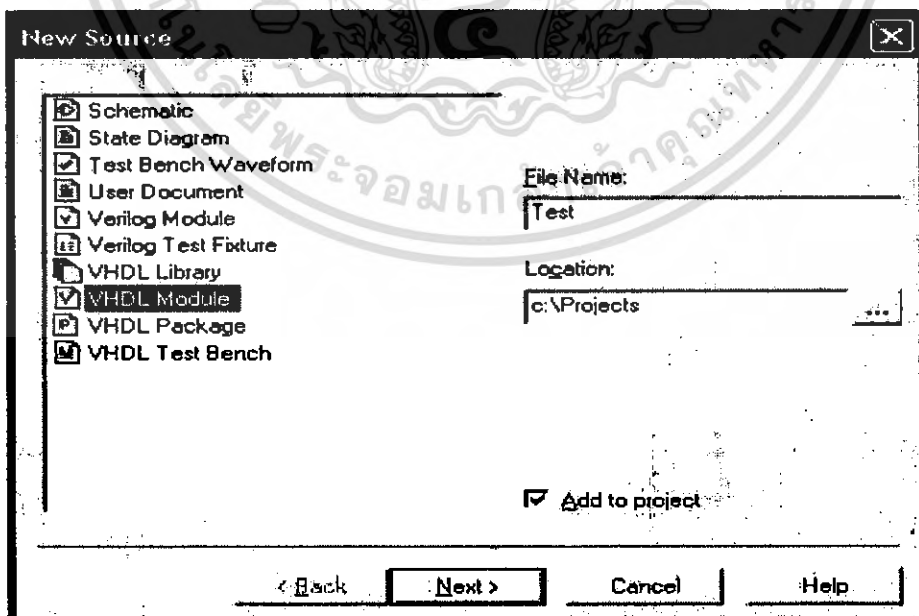
ขั้นตอนการกำหนดขาไอซีและโปรแกรมลงตัวไอซี

1. ให้ทำการคลิกขวา Source in Project เลือกที่ New Source...ดังรูปที่ ข.31



รูปที่ ข.31 ขั้นตอนเข้าไปกำหนดขาอินพุตและเอาต์พุต

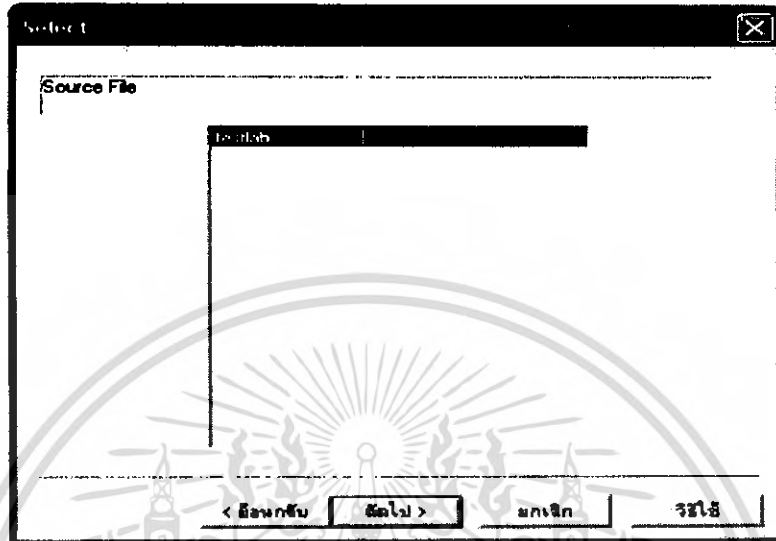
2. รูปที่ ข.32 เป็นการเลือกหัวข้อที่จะทำการกำหนดขาอินพุตและเอาต์พุต โดยเลือกที่หัวข้อ Implementation Constraints File แล้วทำการตั้งชื่อที่ File Name: แล้ว Next



รูปที่ ข.32 การกำหนด Complementation Constraints File

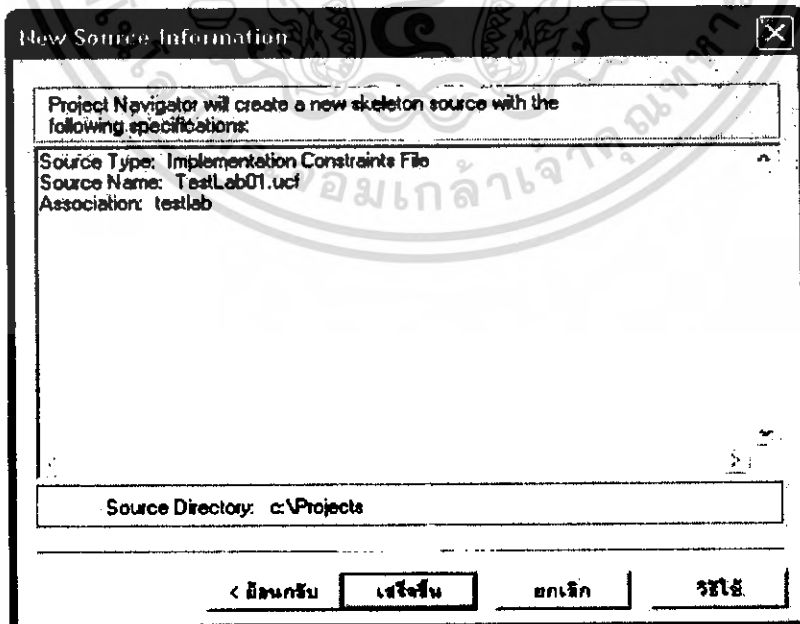
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3. รูปที่ ข.33เป็นการเลือก Select Source File ให้ทำการ Next



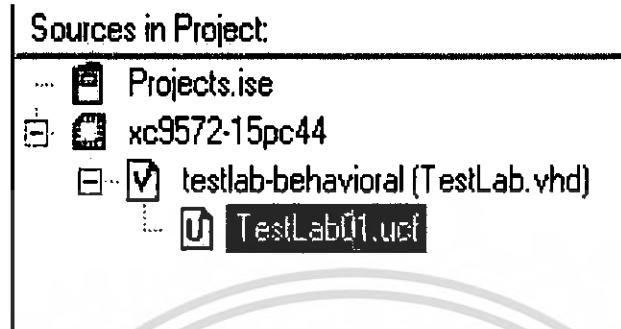
รูปที่ ข.33 การเลือก Select Source File

4.รูปที่ ข.34 New Source Information คือการสรุปการดำเนินการให้คลิก Finish



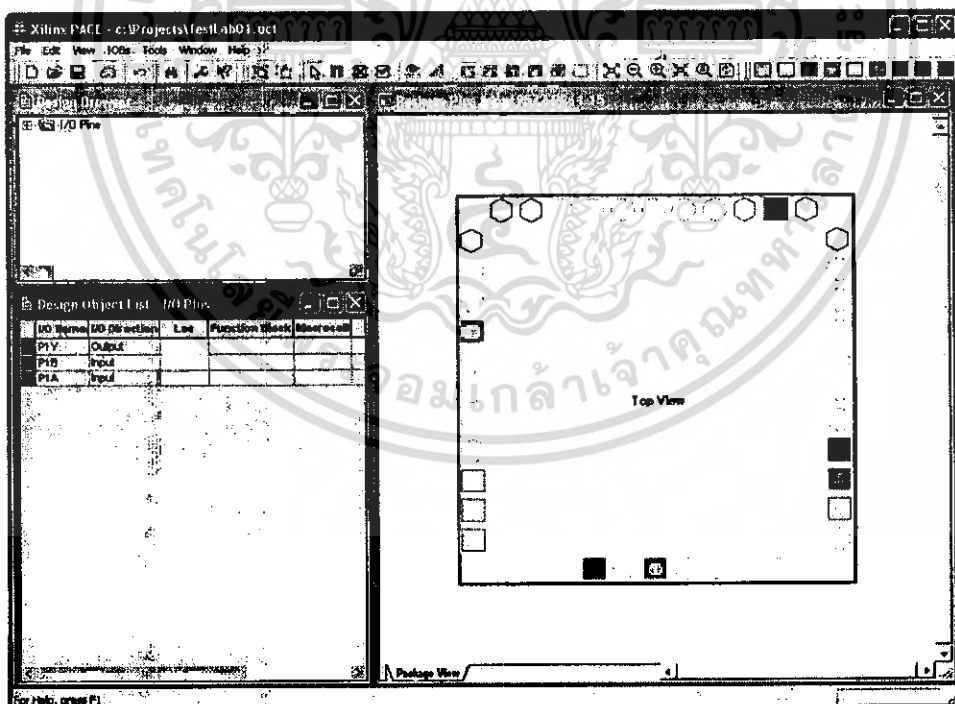
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับรูปที่ ข.34 ขั้นตอน New Source Information ให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5. จะขึ้นชื่อโปรเจกต์ที่ทำการตั้งชื่อ ดังรูปที่ ข.35



รูปที่ ข.35 ชื่อโปรเจกต์ที่ทำการสร้าง

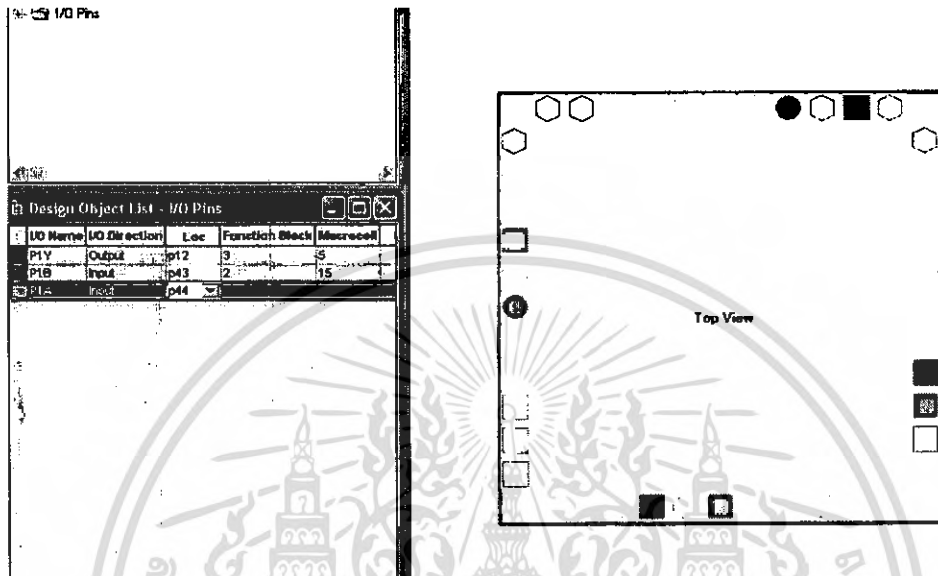
6. ดับเบิลคลิกที่ชื่อโปรเจกต์ที่ทำการสร้าง เพื่อจะดำเนินการกำหนดขาอินพุตและเอาต์พุตให้กับตัวไอซีดังรูปที่ ข.36



รูปที่ ข.36 ปรากฏหน้าต่างเพื่อที่จะทำการกำหนดขาอินพุตและเอาต์พุต

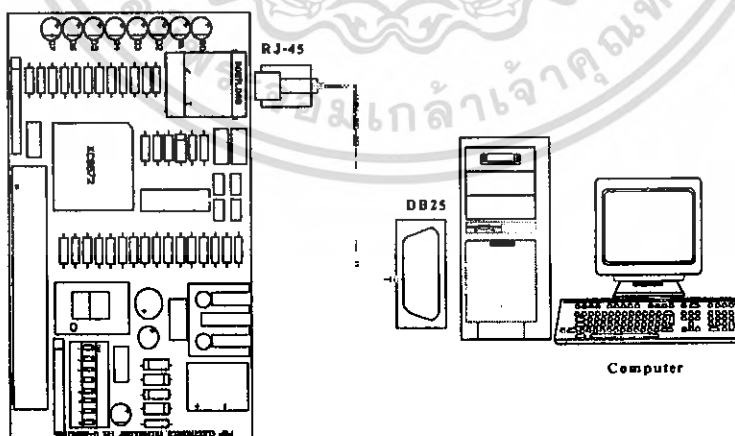
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

7. ให้ทำการกำหนดขาโดยจะดู คู่มือบอร์ดประกอบเมื่อได้ตำแหน่งขาอินพุตและเอาต์พุตแล้วให้ทำการบันทึกแล้วปิดหน้าต่างนี้ลง ดังรูปที่ ข.37



รูปที่ ข.37 การกำหนดขาอินพุตและเอาต์พุต

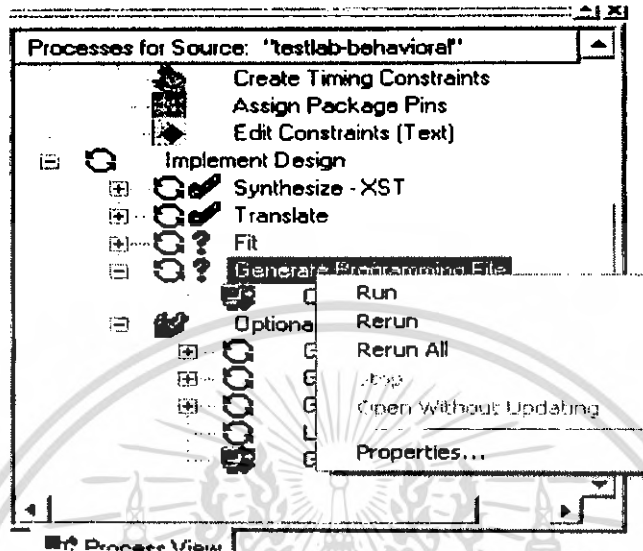
8. เป็นขั้นตอนการโปรแกรมลงตัวไอซีให้ทำการเชื่อมต่อบอร์ด CPLDs XC9572 กับคอมพิวเตอร์ดังรูปที่ ข.38



รูปที่ ข.38 การเชื่อมต่อตัวบอร์ดเข้าคอมพิวเตอร์

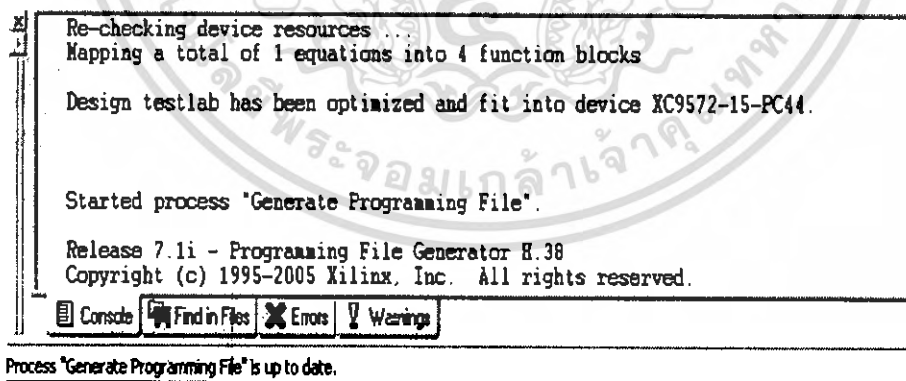
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

9. ทำการคลิกขวา Generate Programming File เลือกที่ Run ดังรูปที่ ข.39



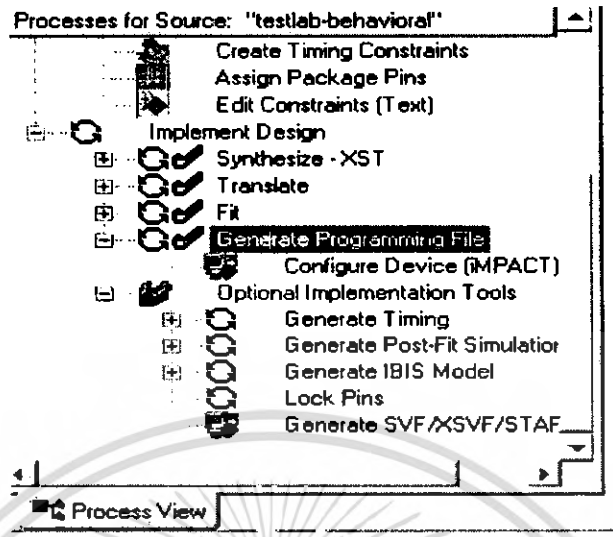
รูปที่ ข.39 ขั้นตอนการ Generate Programming File

10. โปรแกรมจะทำการสังเคราะห์ถ้าไม่พบข้อผิดพลาดจะมีเครื่องหมายถูกที่ข้างหน้า ดังรูปที่ ข.40 และรูปที่ ข.41



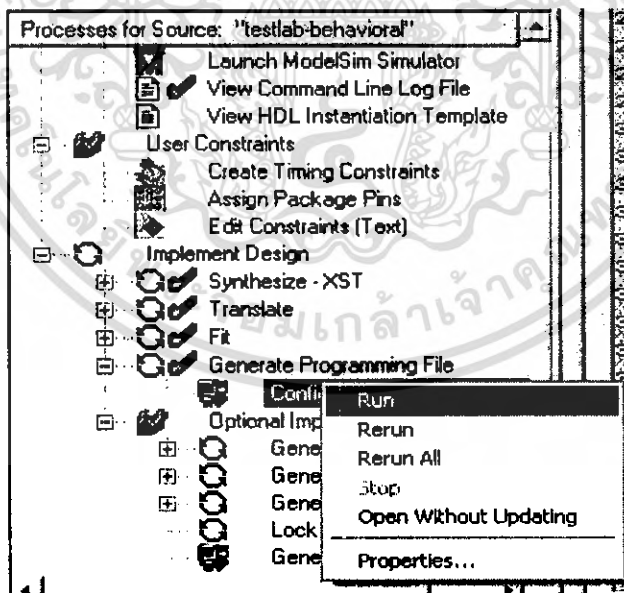
รูปที่ ข.40 ขั้นตอนการสังเคราะห์ก่อนทำการโปรแกรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



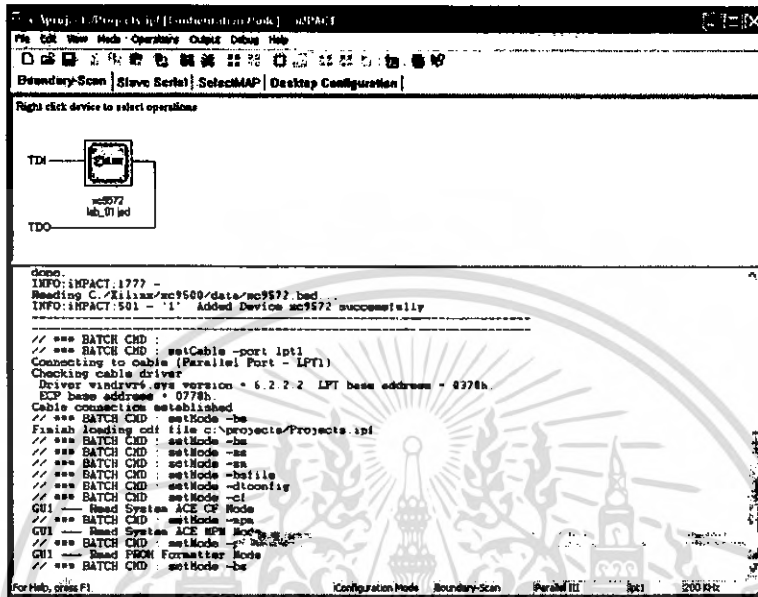
รูปที่ ข.41 เมื่อไม่พบข้อผิดพลาดจะปรากฏเครื่องหมายถูก

11. ขั้นตอนการโปรแกรมลงตัวไอซี ให้ทำการคลิกขวา Configure Device(iMPACT) เลือกที่ Run ดังรูปที่ ข.42



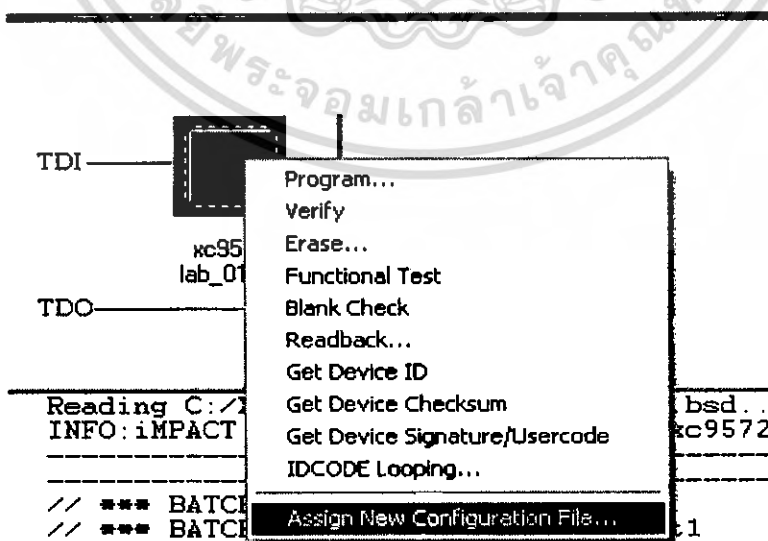
รูปที่ ข.42 การสั่ง Run เพื่อทำการโปรแกรมลงตัวไอซี

12. โปรแกรมจะปรากฏหน้าต่างเพื่อจะทำการขั้นตอนโปรแกรมลงตัวไอซี ดังรูปที่ ข.43



รูปที่ ข.43 หน้าต่างเพื่อที่จะทำโปรแกรมลงตัวไอซี

13. รูปที่ ข.44 ให้ทำการคลิกที่ Assign New Configuration File... เพื่อทำการเลือก File ที่ทำการทำการโปรแกรมลงตัวไอซี

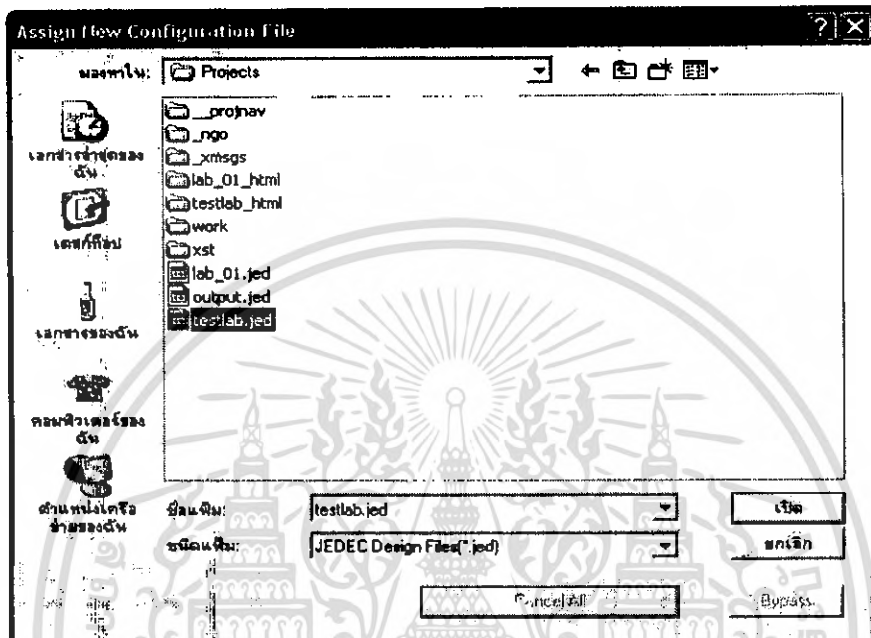


รูปที่ ข.44 Assign New Configuration File...

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

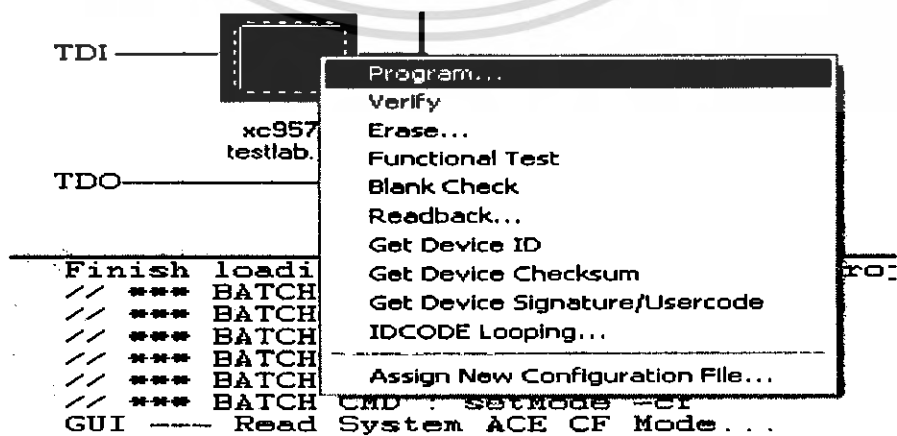
14. รูปที่ ข.45 แสดงถึงการเลือก File นามสกุล .jed ที่ได้ทำการ Implementation Constraints

File



รูปที่ ข.45 การเลือก File เพื่อที่จะทำการโปรแกรมลงตัวไอซี

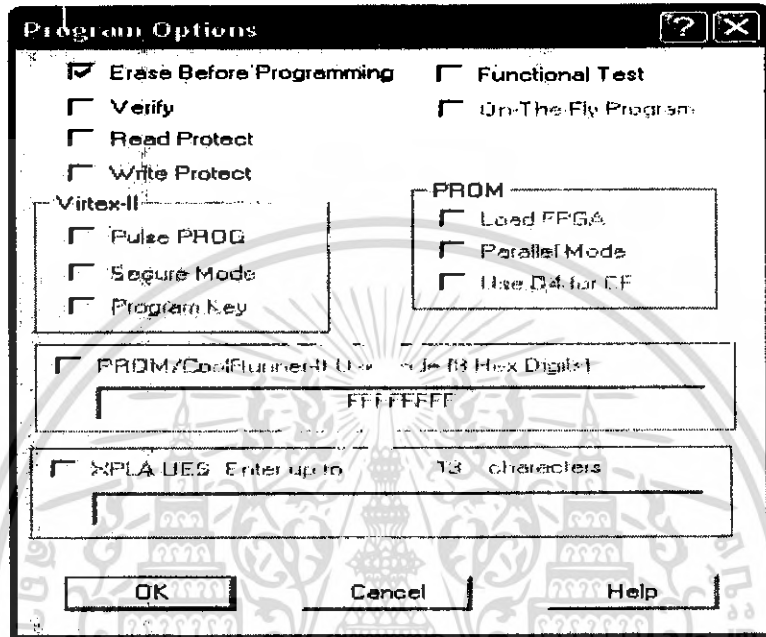
15. รูปที่ ข.46 ให้ทำการเลือก Program... เพื่อที่จะทำการโปรแกรมลงตัวไอซี



รูปที่ ข.46 การโปรแกรมลงตัวไอซี

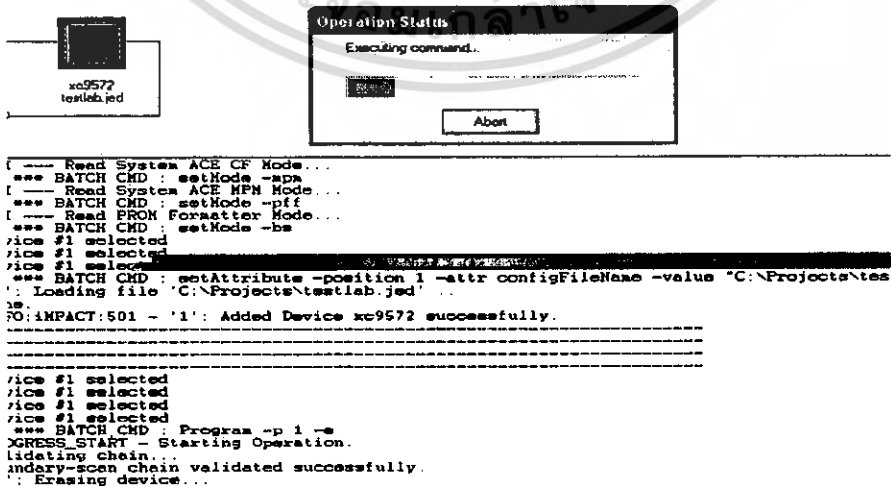
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับกรณที่หน่วยงานที่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

16. รูปที่ ข.47 ให้ทำการเลือกช่อง Erase Before Programming ให้มีเครื่องหมายถูกเกิดขึ้น แล้วคลิก OK เพื่อที่จะทำการโปรแกรมลงตัวไอซี



รูปที่ ข.47 การโปรแกรมลงตัวไอซี (ต่อ)

17. รูปที่ ข.48 โปรแกรมจะตรวจสอบว่ามีการเชื่อมต่อสายจากบอร์ดเข้าคอมพิวเตอร์หรือยังถ้าไม่มีอะไรผิดพลาด จะปรากฏคำว่า Programming Succeeded ดังรูปที่ ข.49



เอกสารนี้เป็นเอกสารที่รูปที่ ข.48 การวิเคราะห์การเชื่อมต่อระหว่างบอร์ดกับคอมพิวเตอร์ไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



Programming Succeeded

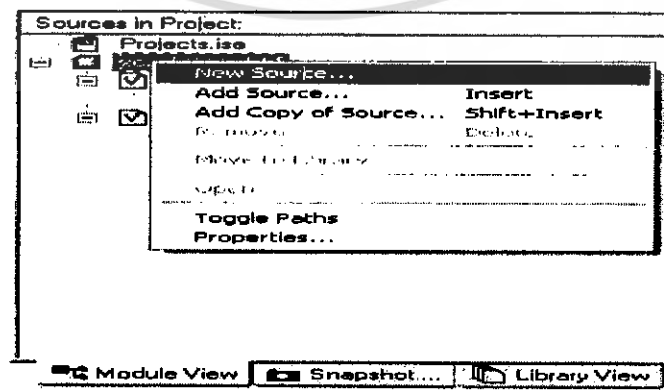
```
Device #1 selected
Device #1 selected
Device #1 selected
Device #1 selected
// *** BATCH CMD : Program -p 1 -e
PROGRESS_START - Starting Operation.
Validating chain...
Boundary-scan chain validated successfully.
'1': Erasing device...
Erasure completed successfully.
done.
Programming completed successfully.
PROGRESS_END - End Operation.
Elapsed time = 13 sec.
Device #1 selected
// *** BATCH CMD : Program -p 1 -e
PROGRESS_START - Starting Operation.
Validating chain...
Boundary-scan chain validated successfully.
'1': Erasing device...
Erasure completed successfully.
done.
Programming completed successfully.
PROGRESS_END - End Operation.
Elapsed time = 13 sec.
```

รูปที่ ข.49 เมื่อไปพบข้อผิดพลาดจะแสดงข้อความ Programming Succeeded

18. เมื่อโปรแกรมทำการโปรแกรมลงตัวไอซีเรียบร้อยแล้วให้ทำการเปิดหน้าต่าง Configure Device [iMPACT] เป็นอันเสร็จสิ้นขั้นตอนการโปรแกรมลงตัวไอซี

การโปรแกรมลงบอร์ด FPGAs Mini Discovery XC2S50

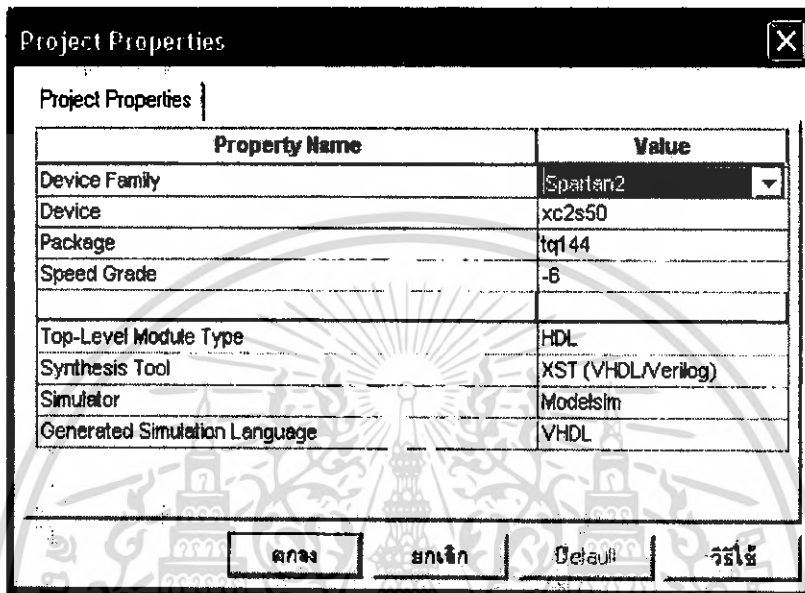
1. ดับเบิลคลิกที่ Sources in Project: เลือกที่ New Sources ดังรูป ข.50 เพื่อที่จะทำการกำหนดอุปกรณ์ ให้ตรงกับบอร์ดที่ใช้งาน



รูปที่ ข.50 เลือก New Sources เพื่อเข้ากำหนดอุปกรณ์ใหม่

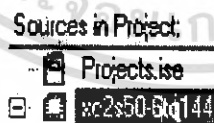
เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้เพื่อการศึกษาค้นคว้าเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. ขั้นตอนนี้จะให้การกำหนดค่าดังรูปที่ ข.51 เพื่อเปลี่ยนไปใช้บอร์ด FPGAs Mini Discovery XC2S50



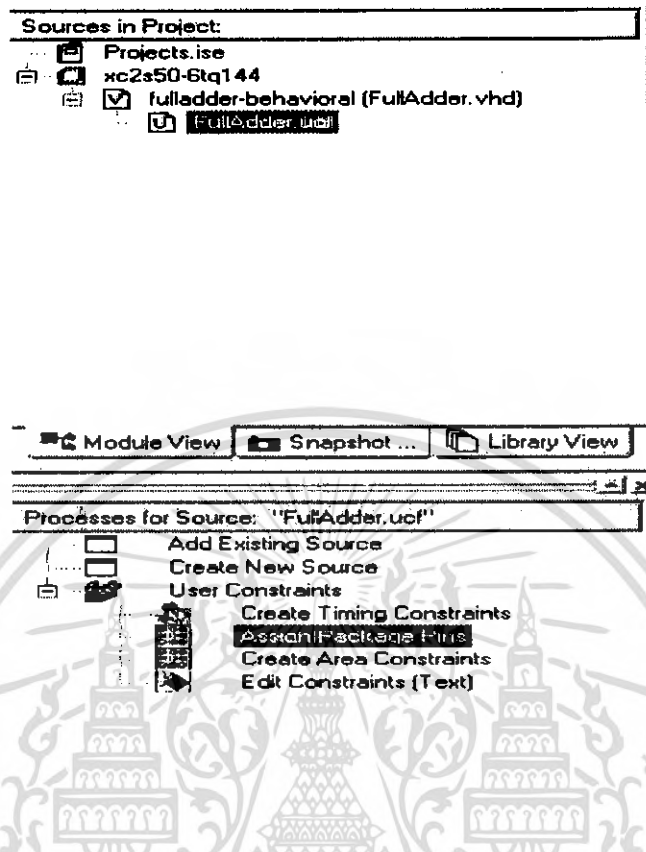
รูปที่ ข.51 การกำหนดค่าของบอร์ด FPGAs Mini Discovery XC2S50

3. รูปที่ ข.52 แสดงบอร์ด ไอซี ที่ใช้ร่วมกับบอร์ด FPGAs Mini Discovery XC2S50 ที่ได้ทำการตั้งค่า



รูปที่ ข.52 บอร์ดไอซี ที่ใช้ร่วมกับบอร์ด FPGAs Mini Discovery XC2S50

4. เมื่อได้ชุดคำสั่งไว้แล้วให้เข้าไปกำหนดขาอุปกรณ์โดยคลิกที่ โครงการที่ได้เขียนชุดคำสั่งแล้วตรง Processes for Source จะปรากฏ Assign Package Pins ดังรูปที่ ข.53 ให้ทำการคลิก แล้วโปรแกรมจะขึ้นหน้าต่างเพื่อกำหนดขา อินพุต และ เอาต์พุต ดังรูปที่ ข.54



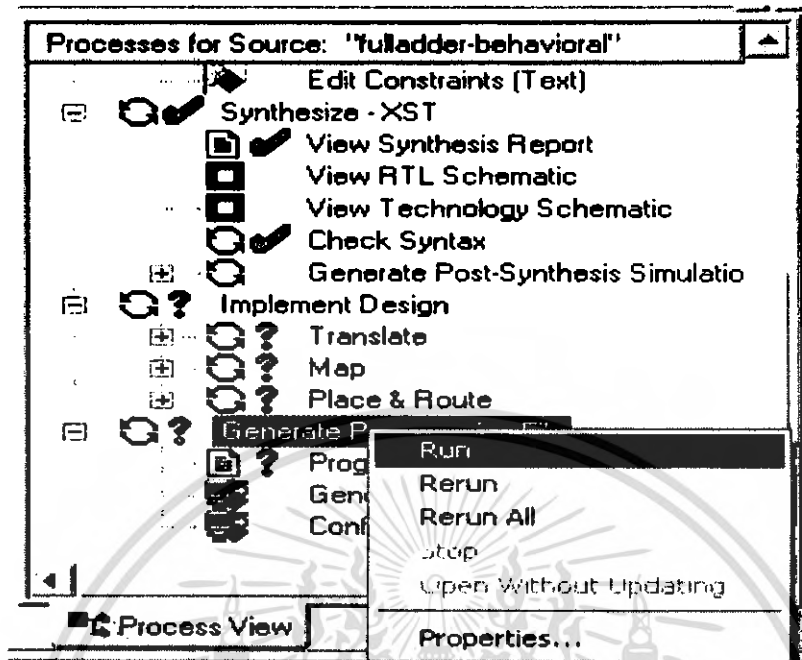
รูปที่ ข.53 ขั้นตอนการกำหนดขา อินพุต และ เอาต์พุต

Design Object List - I/O Pins						
I/O Name	I/O Direction	Loc	Bank	I/O Std.	Vref	
a	Input	p129	BANK7			
b	Input	p130	BANK7			
ci	Input	p131	BANK7			
co	Output	p116	BANK6			
sum	Output	p115	BANK6			

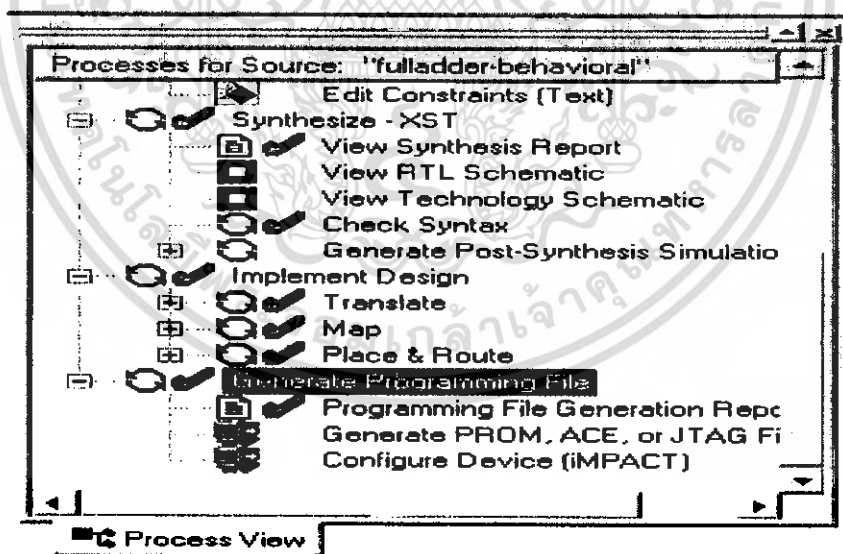
รูปที่ ข.54 การกำหนดขาอินพุต และ เอาต์พุต

5. ทำการ Generate Programming File โดยคลิกขวาแล้วคลิก Run ดังรูปที่ ข.55

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษเท่านั้น ไม่อนุญาตให้นำไปเผยแพร่โดยไม่ได้รับอนุญาต หากมีข้อผิดพลาดจะมีเครื่องถูกปรากฏขึ้นดังรูปที่ ข.56



รูปที่ ข.55 การทำ Generate Programming File

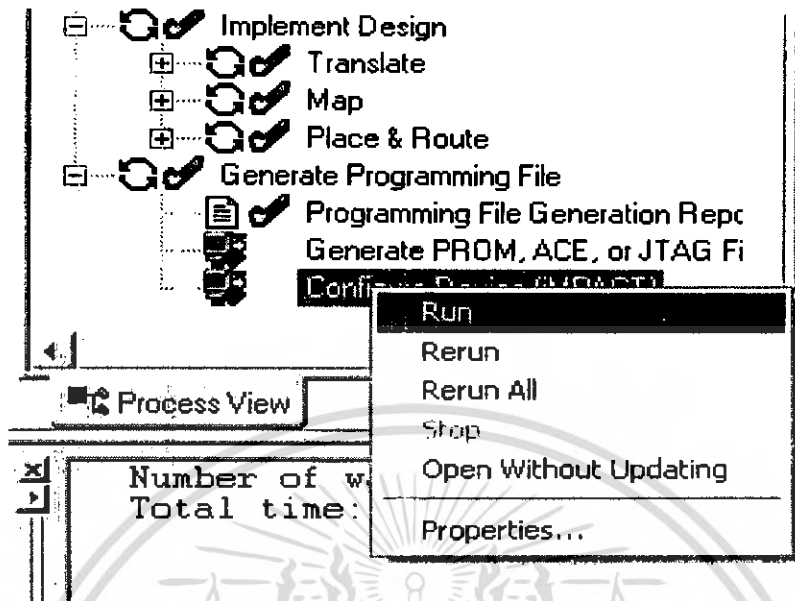


รูปที่ ข.56 เมื่อไม่มีข้อผิดพลาดจะมีเครื่องหมายถูกปรากฏ

6. ทำการคลิก Configure Device [iMPACT] เพื่อเข้าสู่การโปรแกรมลงตัวไอซี ดัง

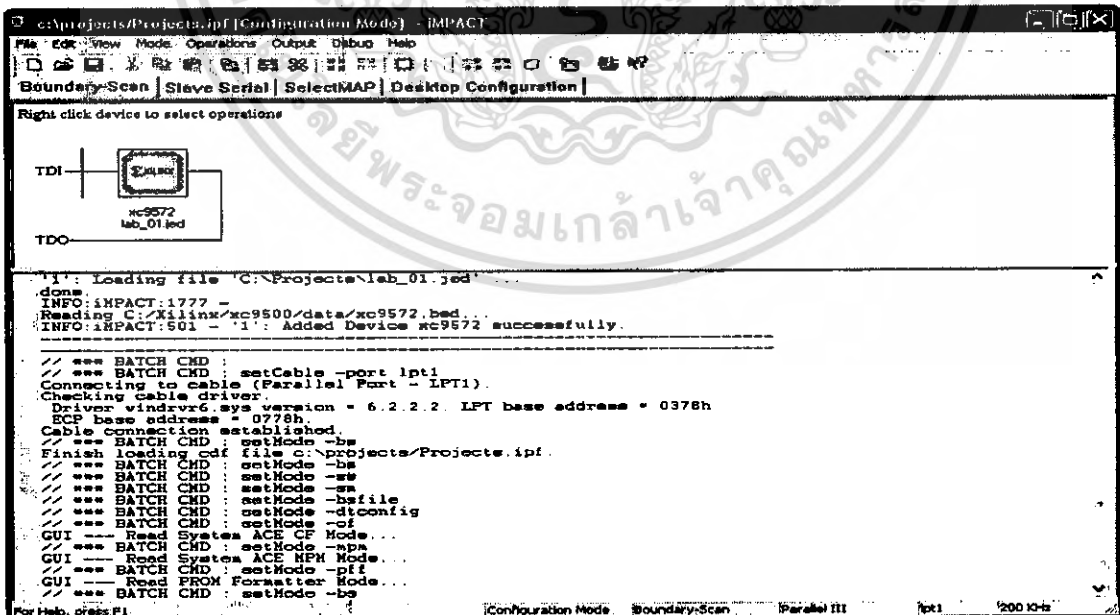
รูปที่ ข.57

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



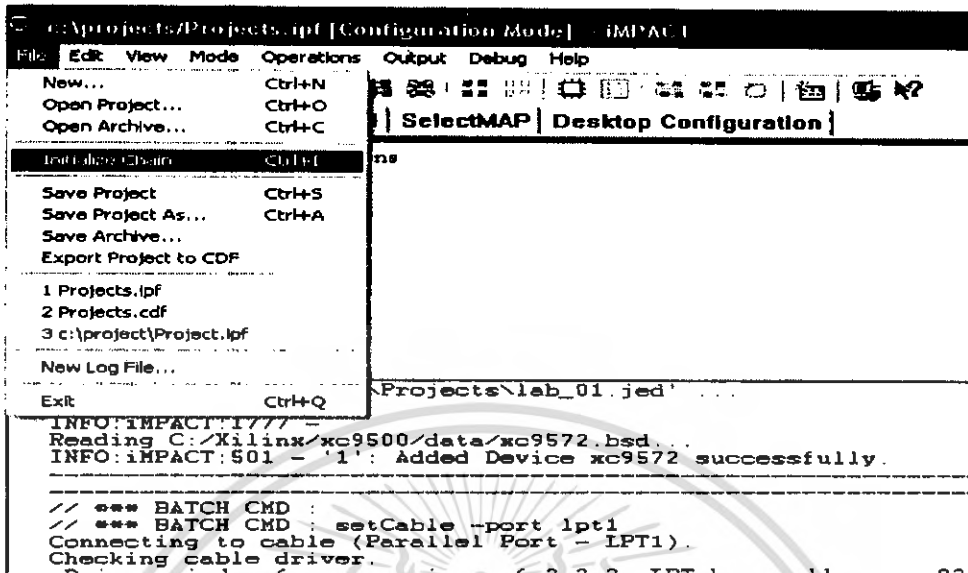
รูปที่ ๕.๕๗ การเข้าสู่การโปรแกรมลงตัวไอซี

7. โปรแกรมจะเข้าสู่หน้าต่างในการโปรแกรมลงตัวไอซีดังรูปที่ ๕.๕๘ ส่วนรูปที่ ๕.๕๙ คือการกำหนดตัวไอซีใหม่อีกครั้งโดยการคลิกที่ File เลือกที่ Initialize Chainเพื่อให้ตรงกับบอร์ดที่ใช้งาน



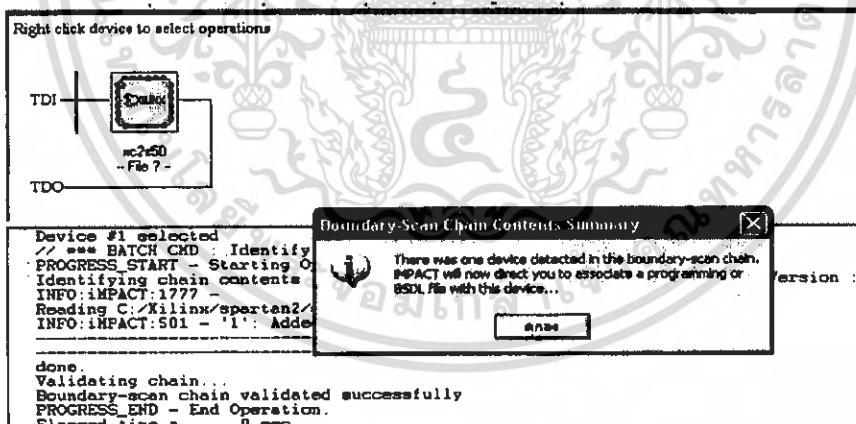
รูปที่ ๕.๕๘ หน้าต่างที่จะโปรแกรมลงตัวไอซี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



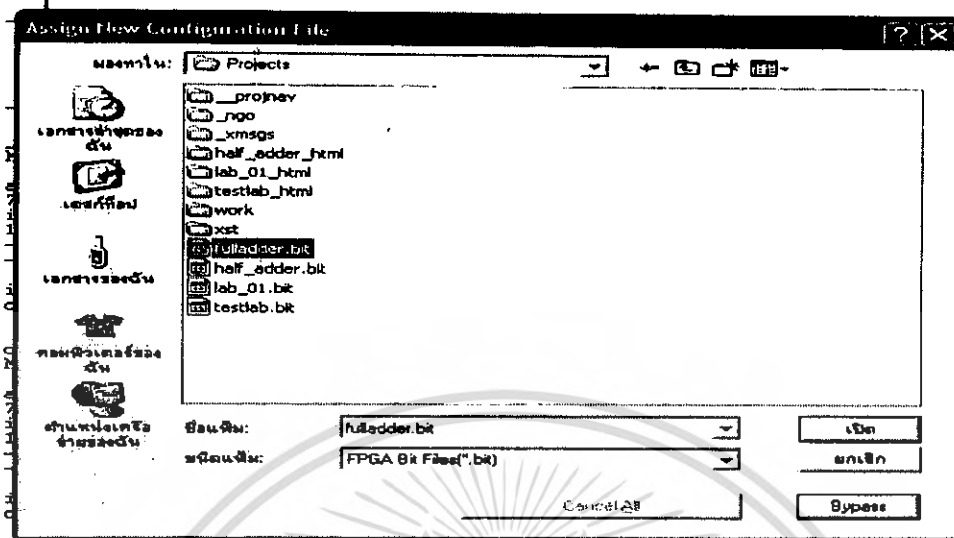
รูปที่ ข.59 Initialize Chain เพื่อทำการเปลี่ยนไอซี

8. รูปที่ ข.60 แสดงถึงการเปลี่ยนเบอร์ไอซีเมื่อทำการ Initialize Chain กด OK



รูปที่ ข.60 เมื่อทำการ Initialize chain จะปรากฏเบอร์ไอซีตัวใหม่เกิดขึ้น

9. ทำการเลือก File ที่จะทำการโปรแกรมลงตัวไอซีแต่ในที่นี้จะเป็นอุปกรณ์แบบ FPGAs File ที่ใช้จะเป็นนามสกุล .bit เมื่อเลือก File แล้วคลิก Open ดังรูปที่ ข.61



รูปที่ ข.61 เลือก File ที่จะทำการโปรแกรมลงตัว FPGAs

10. ให้ทำการโหลด โปรแกรมที่ทำการเลือกลงตัวไอซี FPGAs ขั้นตอนการโปรแกรมลงตัวไอซีจะเหมือนกับการโปรแกรม CPLDs ทุกประการ

ใบงานที่ 2

การทำงานของลอจิกเกต

วัตถุประสงค์

1. เพื่อให้นักศึกษาใช้โปรแกรม Xilinx Foundation 7.1i ได้
2. เพื่อให้นักศึกษาจำลองการทำงานของลอจิกเกตเบื้องต้นโดยโปรแกรม Xilinx ISE 2.1i ได้
3. เพื่อให้นักศึกษานำวงจรดิจิทัลที่ออกแบบมาสร้างจริงโดยการโปรแกรมอุปกรณ์ CPLDs ได้

เครื่องมือสำหรับการทดลอง

1. ระบบไมโครคอมพิวเตอร์
2. โปรแกรม Xilinx ISE 7.1i
3. สายดาร์วินโหลด
4. บอร์ดทดลอง CPLD เบอร์ XC9572PC44

ทฤษฎีพื้นฐาน

โปรแกรม Xilinx ISE 2.1i เป็นโปรแกรมที่ช่วยในการพัฒนาการออกแบบวงจรดิจิทัล ซึ่งสามารถออกแบบได้ทั้งการใช้วิธีเขียนวงจร วิธีบรรยายพฤติกรรมวงจรด้วยภาษา ABEL VHDL และ Verilog โดยเฉพาะอย่างยิ่งเครื่องมือที่จะสร้างไฟล์เพื่อโปรแกรมลงบนอุปกรณ์ CPLDs และ Configuration ข้อมูลให้อุปกรณ์ FPGAs ได้ ซึ่งในใบงานที่ 1 นี้กำหนดให้ผู้ออกแบบใช้โปรแกรม Xilinx Foundation 2.1I ในการออกแบบเพื่อแสดงการทำงานของเกตเบื้องต้น เช่น AND GATE, OR GATE, NOT GATE, NAND GATE และ NOR GATE ด้วยวิธีเขียนวงจรพร้อมทั้งการเลียนแบบการทำงานของวงจรเพื่อตรวจสอบความถูกต้องก่อนที่จะโปรแกรมลงบนอุปกรณ์ CPLD

ขั้นตอนการทดลอง

1. ทำการพิมพ์ชุดคำสั่ง And Gate ดังรูปที่ ข.62

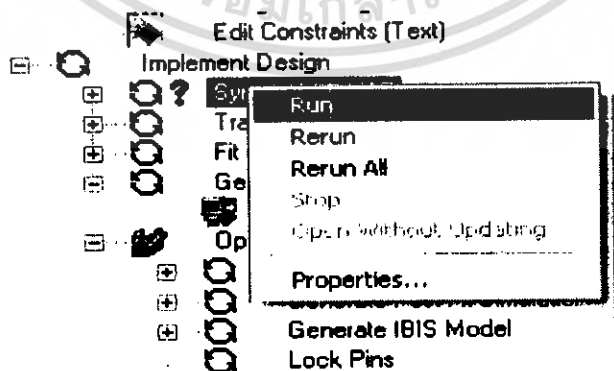
```

19 -----
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 entity And is
31     (P1A,P1B:IN std_logic:= 'U';
32      P1Y:OUT std_logic);
33
34 end And;
35
36 architecture Behavioral of And is
37
38 begin
39     P1Y<=P1A and P1B;
40
41 end Behavioral;
42

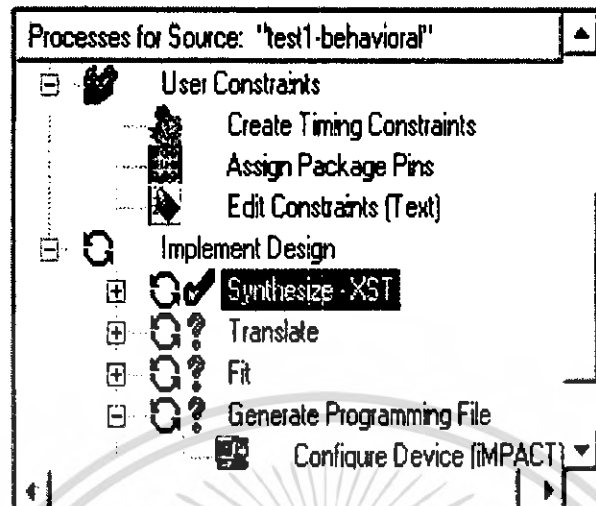
```

รูปที่ ข.62 ชุดคำสั่ง And Gate

2. ทำการ Synthesize-SXT ชุดคำสั่งที่เราทำการพิมพ์ในขั้นตอนที่หนึ่งเพื่อเป็นการตรวจสอบชุดคำสั่งว่ามีการผิดพลาดหรือไม่ โดยการคลิกขวาที่คำว่า Synthesize-SXT และคลิกคำว่า Run ถ้าชุดคำสั่งไม่มีอะไรผิดพลาดจะมีเครื่องหมาย ถูกขึ้นที่คำว่า Synthesize-SXT ดังรูปที่ ข.63

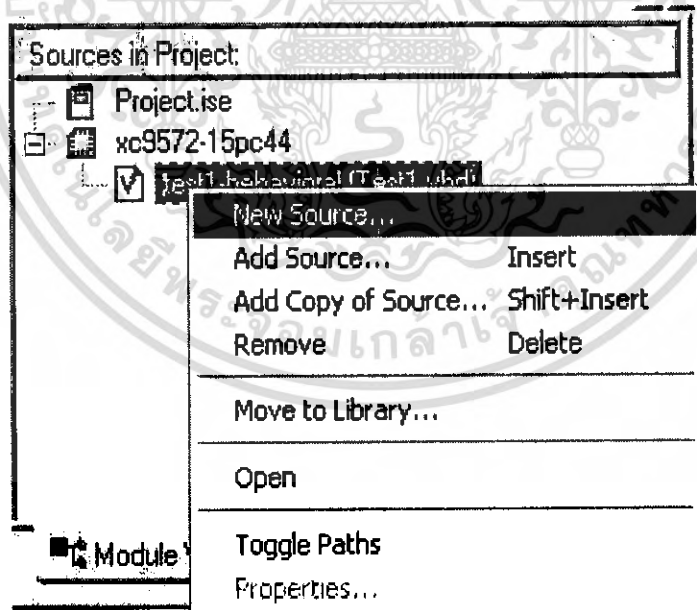


รูปที่ ข.63 ขั้นตอนการทำ Synthesize-SXT



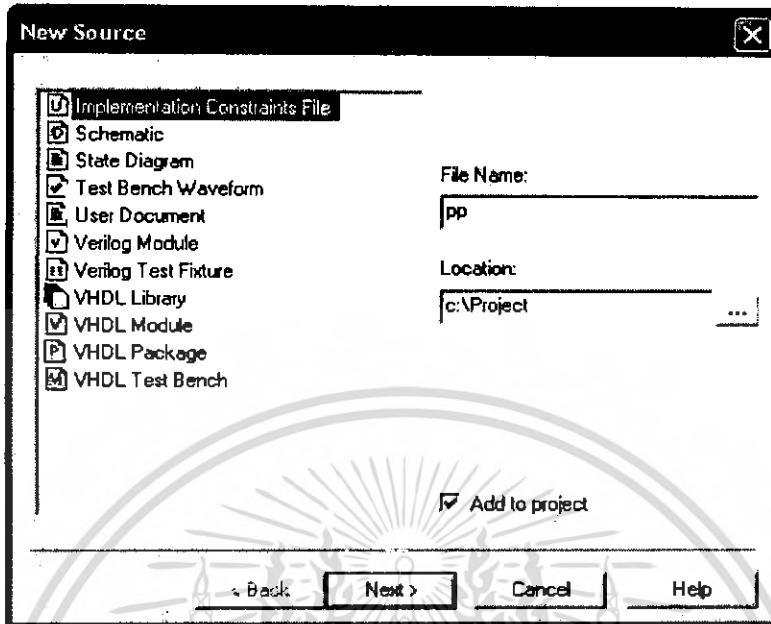
รูปที่ ข.64 เมื่อไม่พบข้อผิดพลาด

3. ทำการกำหนดค่าให้ IC เพื่อที่จะนำโปรแกรมไปใช้งานโดยที่ไปคลิกขวาที่ชื่อของโปรเจกต์ที่เรากำหนดไว้หลังจากนั้นจะมีแถบคำสั่งขึ้นมาให้เลือกที่ New Source จะมีแถบขึ้นมาให้เลือกดังรูปที่ ข.65



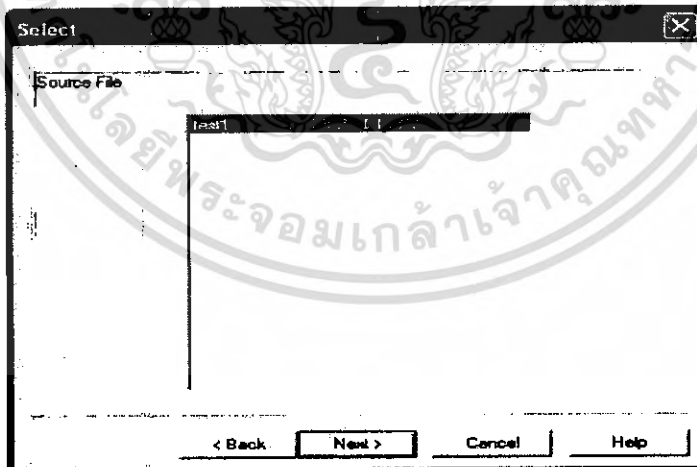
รูปที่ ข.65 เปิดโปรเจกต์ใหม่

4. เมื่อเลือกคลิกที่ New Source แล้วจะมีหน้าต่าง New Source ขึ้นมาให้เลือกที่ Implementation Constraints File และให้กำหนดชื่อในช่อง File Name: ดังรูปที่ ข.66 ใต้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



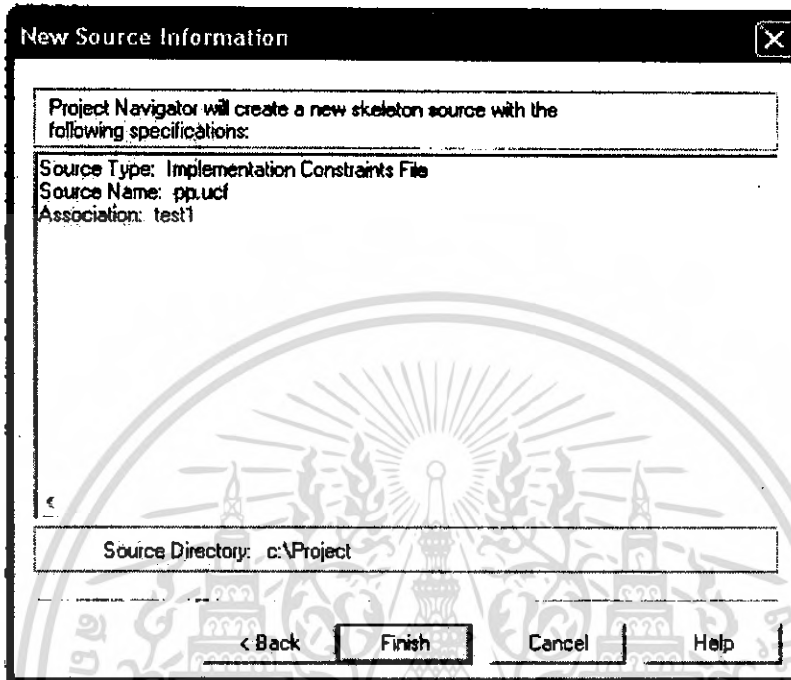
รูปที่ ข.66 Implementation Constraints File

5. เมื่อกำหนดครบแล้วให้คลิกคำว่า Next จะขึ้นหน้าต่าง Select ดังรูปที่ ข.67



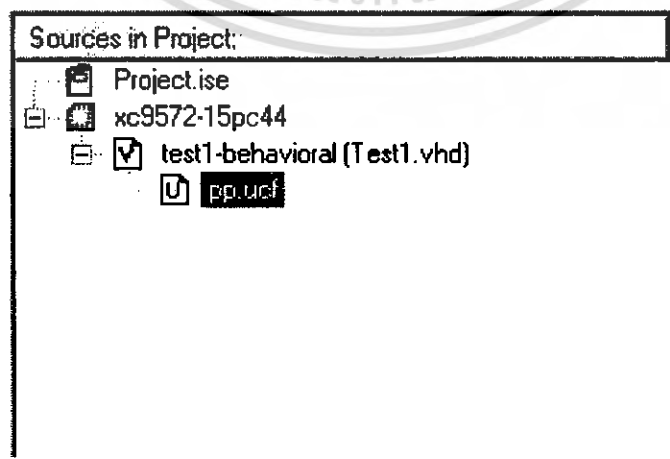
รูปที่ ข.67 Source File ที่ทำการสร้าง

6. ให้คลิกที่คำว่า Next ต่อได้เลยหน้าต่างจะเปลี่ยนเป็น New Source Information ดังรูปที่ ข.68



รูปที่ ข.68 การยืนยันโปรเจคใหม่

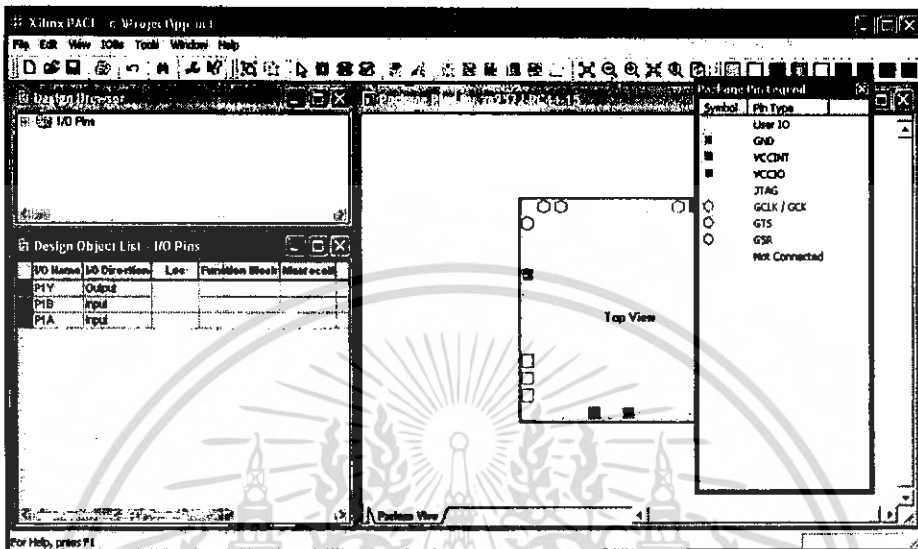
7. คลิกที่ Finish จะมีแถบคำเพิ่มขึ้นมาที่หน้าต่าง Sources in Project: หลังจากนั้นให้ดับเบิลที่คำที่เพิ่มขึ้นมาใหม่เพื่อเรียกหน้าต่างจะทำการกำหนดขาขึ้นมา ดังรูปที่ ข.69



รูปที่ ข.69 โปรเจคใหม่ที่สร้างขึ้น

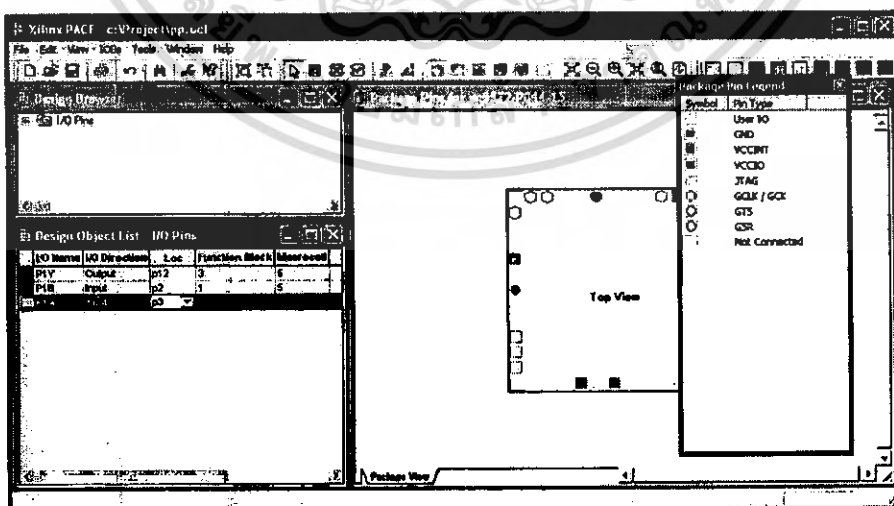
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษานี้เท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

8. หลังจากนั้นหน้าต่างที่จะทำการกำหนดขาของ IC ก็จะดังขึ้นมาดังรูปที่ ข.70



รูปที่ ข.70 หน้าต่างการกำหนดขาอุปกรณ์

9. กำหนดขาของ IC ในตาราง Design Object List - I/O Pins โดยให้ P1Y เป็น Output และ P1Acและ P1B เป็น Input เพื่อให้แสดงผลที่หลอด LED หลอดที่หนึ่ง ดังรูปที่ ข.71 และทำการโปรแกรม ลงตัวไอซี ขั้นตอนการโปรแกรมลงตัวไอซีได้จากใบงานที่ 1



รูปที่ ข.71 การกำหนดขาของอุปกรณ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ผลการทดลองของชุดคำสั่ง แอน เกต ดังตารางที่ ข.1

ตารางที่ ข.1 ตารางการทำงานของ แอนเกต

P1A	P1B	P1Y
0	0	0
0	1	0
1	0	0
1	1	1

10. ทำการพิมพ์ชุดคำสั่ง ออ เกต ดังรูปที่ ข.72

```

19 -----
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 entity Test1 is
31 port(P1A,P1B: in std_logic:='U';
32      P1Y:out std_logic);
33 end Test1;
34
35 architecture Behavioral of Test1 is
36
37 begin
38
39   P1Y<=P1A or P1B;
40 end Behavioral;
41

```

รูปที่ ข.72 ชุดคำสั่งของ ออ เกต

11. เมื่อพิมพ์ชุดคำสั่งของ ออ เกตเสร็จสมบูรณ์แล้ว ให้ทำการ Synthesize-XST เพื่อเป็นการตรวจสอบข้อผิดพลาดเมื่อไม่มีข้อผิดพลาดแล้วจึงทำการกำหนดค่าของไอซีดังเช่นไปงานที่ ง.1 และจึงทำการ
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
บันทึกผลการทดลองของ ออเกต ดังตารางที่ ง.2
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ ข.2 ตารางการทำงานของ ออ เกต

P1A	P1B	P1Y
0	0	0
0	1	1
1	0	1
1	1	1

13. ทำการพิมพ์ชุดคำสั่ง นอร์ เกต ดังรูปที่ ข.73

```

19 -----
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 entity Test1 is
31 port (P1A,P1B: in std_logic:='U';
32       P1Y:out std_logic);
33 end Test1;
34
35 architecture Behavioral of Test1 is
36
37 begin
38
39     P1Y<=P1A nor P1B;
40 end Behavioral;
41

```

รูปที่ ข.73 ชุดคำสั่งของ นอร์เกต

14. เมื่อพิมพ์ชุดคำสั่งของ Nor Gate เสร็จสมบูรณ์แล้ว ให้ทำการ Synthesize-XST เพื่อเป็นการตรวจสอบข้อผิดพลาดเมื่อไม่มีข้อผิดพลาดแล้วจึงทำการกำหนดขาของไอซีดังเช่นใบงานที่ 1 และจึงทำการบันทึกผลการทดลองของ นอร์ เกต ดังตารางที่ ข.3

เอกสารนี้เป็นเอกสารลิขสิทธิ์ของสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง การศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ ข.3 ตารางการทำงานของ นอร์เกต

P1A	P1B	P1Y
0	0	1
0	1	1
1	0	1
1	1	0

15. ทำการพิมพ์ชุดคำสั่ง เนน เกต ดังรูปที่ ข.74

```

19 -----
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 entity Test1 is
31 port (P1A,P1B: in std_logic='U';
32       P1Y:out std_logic);
33 end Test1;
34
35 architecture Behavioral of Test1 is
36
37 begin
38
39     P1Y<=P1A nand P1B;
40 end Behavioral;
41

```

รูปที่ ข.74 เนน เกต

เมื่อพิมพ์ชุดคำสั่งของ เนน เกต เสร็จสมบูรณ์แล้ว ให้ทำการ Synthesize-XST เพื่อเป็นการตรวจสอบข้อผิดพลาดเมื่อไม่มีข้อผิดพลาดแล้วจึงทำการกำหนดค่าของไอซีดังเช่นใบงานที่ 1 และจึงทำการบันทึกผลการทดลองของ เนน เกต ดังตารางที่ ข.4

ตารางที่ ๓.4 ตารางการทำงานของ แนน เกต

P1A	P1B	P1Y
0	0	0
0	1	1
1	0	1
1	1	0

16. ทำการพิมพ์ชุดคำสั่ง เอ็กซ์อ เกต ดังรูปที่ ๓.75

```
19 -----
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 entity Test1 is
31   port(P1A,P1B: in std_logic:='U';
32         P1Y:out std_logic);
33 end Test1;
34
35 architecture Behavioral of Test1 is
36
37 begin
38
39   P1Y<=P1A xor P1B;
40 end Behavioral;
41
```

รูปที่ ๓.75 เอ็กซ์อ เกต

เมื่อพิมพ์ชุดคำสั่งของ เอ็กซ์อ เกตอ เสร็จสมบูรณ์แล้ว ให้ทำการ Synthesize-XST เพื่อเป็นการตรวจสอบข้อผิดพลาดเมื่อไม่มีข้อผิดพลาดแล้วจึงทำการกำหนดขาของไอซีดังเช่นใบบางที่ 1 และจึงทำการบันทึกผลการทดลองของ เอ็กซ์อ เกต ดังตารางที่ ๓.5

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ ข.5 ตารางการทำงานของ เอ็กซ์ออ เกต

P1A	P1B	P1Y
0	0	1
0	1	0
1	0	0
1	1	0

17. ทำการพิมพ์ชุดคำสั่ง เอ็กซ์นอร์ เกต ดังรูปที่ ข.76

```

19 -----
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 entity Test1 is
31 port(P1A,P1B: in std_logic:= 'U';
32      P1Y:out std_logic);
33 end Test1;
34
35 architecture Behavioral of Test1 is
36
37 begin
38
39     P1Y<=P1A xnor P1B;
40 end Behavioral;
41

```

รูปที่ ข.76 เอ็กซ์นอร์ เกต

เมื่อพิมพ์ชุดคำสั่งของ Xnor Gate เสร็จสมบูรณ์แล้ว ให้ทำการ Synthesize-XST เพื่อเป็นการตรวจสอบข้อผิดพลาดเมื่อไม่มีข้อผิดพลาดแล้วจึงทำการกำหนดขาของไอซีดังเช่นในงานที่ 1 และจึงทำการบันทึกผลการทดลองของ เอ็กนอ เกต ดังตารางที่ ข.6

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ ข.6 ตารางการทำงานของ เอ็กนอร์ เกต

P1A	P1B	P1Y
0	0	1
0	1	0
1	0	0
1	1	1

บันทึกผลการทดลอง

ตารางที่ ข.7 ตารางผลการทดลองของ แอน เกต

P1A	P1B	P1Y
0	0	
0	1	
1	0	
1	1	

ตารางที่ ข.8 ตารางผลการทดลองของ ออ เกต

P1A	P1B	P1Y
0	0	
0	1	
1	0	
1	1	

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ ข.9 ตารางผลการทดลองของ แนน เกต

P1A	P1B	P1Y
0	0	
0	1	
1	0	
1	1	

ตารางที่ ข.10 ตารางผลการทดลองของ นอร์ เกต

P1A	P1B	P1Y
0	0	
0	1	
1	0	
1	1	

ตารางที่ ข.11 ตารางผลการทดลองของ เอ็กอ เกต

P1A	P1B	P1Y
0	0	
0	1	
1	0	
1	1	

ตารางที่ ข.12 ตารางผลการทดลองของ เอ็กนอร์ เกต

P1A	P1B	P1Y
0	0	
0	1	
1	0	
1	1	

สรุปผลการทดลอง

ใบงานที่ 3

การออกแบบวงจร Adder

วัตถุประสงค์

1. เพื่อให้นักศึกษาใช้โปรแกรม Xilinx 7.1i ได้
2. เพื่อให้นักศึกษาจำลองการทำงานของวงจร Adder โดยโปรแกรม Xilinx 7.1i ได้
3. เพื่อให้นักศึกษานำวงจรดิจิทัลที่ออกแบบมาสร้างจริงโดยการโปรแกรมอุปกรณ์ CPLD ได้

เครื่องมือสำหรับการทดลอง

1. ระบบไมโครคอมพิวเตอร์
2. โปรแกรม Xilinx 7.1i
3. สายดาวน์โหลด
4. บอร์ดทดลอง CPLD เบอร์ XC9572PC44

ทฤษฎีพื้นฐาน

วงจร Adder คือวงจรที่รับตัวเลข Binary 2 bits เข้าด้วยกัน ซึ่งวงจรจะมีอยู่ 2 ประเภทคือ

1. วงจร Half Adder โดยสมการของวงจรมีดังนี้

$$\text{Sum} = A \oplus B$$

$$\text{Carry} = A \cdot B$$
2. วงจร Full Adder โดยสมการของวงจรมีดังนี้

$$\text{Sum} = \overline{A}BC_i + A\overline{B}C_i + AB\overline{C}_i + ABC_i$$

$$\text{Carry} = AB + AC_i + BC_i$$

จากสมการทั้งสองวงจรมีสามารถนำไปเขียนเป็นวงจรดิจิทัลที่ Schematic editor โดยใช้ลอจิกเกตพื้นฐานมาประกอบกัน

วงจรบวกเลข แบบไม่คิดตัวทด

วงจร Half Adder เป็นวงจรรับบวกเลขฐานสอง 2 บิต เข้าด้วยกันซึ่งเป็นพื้นฐานของการบวกเลขฐานสองหลายบิต และนำไปใช้ในเครื่องคำนวณทั่วไป ตารางการบวกเลข 2 บิต

ตารางที่ ข.13 ตารางการทำงานของ Half Adder

INPUT		OUTPUT	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Sum คือ ผลรวม

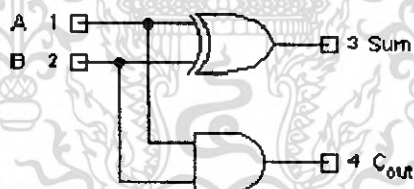
Carry คือ ตัวทด

จากตารางเราสามารถเขียนเป็นสมการลอจิกได้ดังนี้

$$\text{Sum} = \bar{A}B + A\bar{B}$$

$$\text{Carry} = A \bullet B$$

จากสมการลอจิกที่ได้นำมาเขียนเป็นวงจรได้ดังนี้



หรือบางครั้งอาจจะพบเห็นสัญลักษณ์ของวงจรบวก Half Adder ได้ดังนี้



การใช้ชุดคำสั่งของ Half Adder

1. เขียนชุดคำสั่งของ Half adder ดังรูปที่ ข.77 เมื่อพิมพ์ชุดคำสั่งของ Half adder เสร็จสมบูรณ์แล้ว ให้ทำการ Synthesize-XST เพื่อเป็นการตรวจสอบข้อผิดพลาดเมื่อไม่มีข้อผิดพลาดแล้วจึงทำการกำหนดขาของไอซีดังเช่นใบงานที่ 1 และจึงทำการกำหนดขาเพื่อแสดงเอาต์พุตของชุดคำสั่งออกทาง Ic ทำได้โดยทำการดับเบิลคลิก Assign Package Pins เพื่อเรียกหน้าต่างที่จะทำการกำหนดขาขึ้นมา ดังรูปที่ ข.78

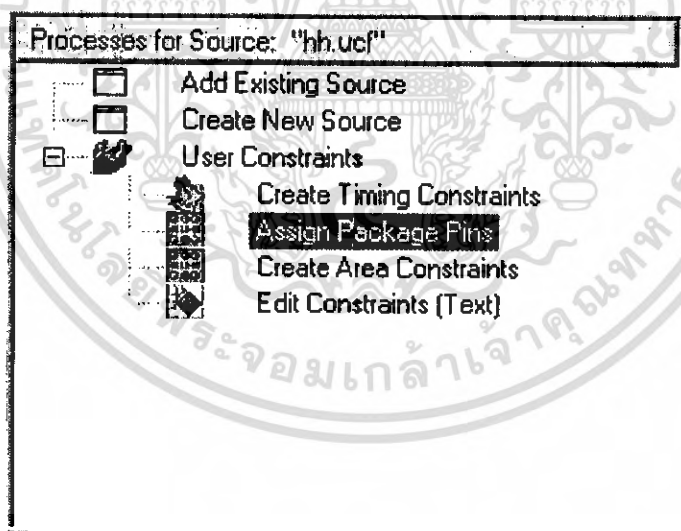
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

19 -----
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 library IEEE;
31 use IEEE.STD_LOGIC_1164.ALL;
32
33 entity half_adder is
34 Port ( A : in STD_LOGIC;
35       B : in STD_LOGIC;
36       SUM : out STD_LOGIC;
37       Cout : out STD_LOGIC);
38 end half_adder;
39
40 architecture Behavioral of half_adder is
41 begin
42
43     SUM <= A xor B;
44     Cout <= A and B;
45
46 end Behavioral;
47

```

รูปที่ ข.77 ชุดคำสั่งของ Half adder



รูปที่ ข.78 ทำการ Synthesize-XST

2. เมื่อหน้าต่างขึ้นมาแล้วจึงทำการกำหนดขาที่จะนำไปใช้งานดังรูปที่ ข.79

Design Object List - I/O Pins						
	I/O Name	I/O Direction	Loc	Bank	I/O Std.	Vref V
	A	Input	p129	BANK		
	B	Input	p130	BANK		
	Cout	Output	p115	BANK		
	SUM	Output	p116	BANK		

รูปที่ ข.79 การกำหนดขา Ic

3. หลังจากนั้นจึงทำการดาวน์โหลดชุดคำสั่งลงไปยัง IC เพื่อทำการทดลองเพื่อบันทึกผลการทดลองตามตารางที่ ค.14

ตารางที่ ข.14 ตารางผลการทดลองของ Half Adder

INPUT		OUTPUT	
A	B	Sum	Carry
0	0		
0	1		
1	0		
1	1		

วงจรบวกเลขฐานสองแบบคิตตัวท (Full Adder)

วงจร Full Adder เป็นวงจรที่ใช้บวกเลขฐานสอง 2 บิต และตัวทเข้า (Ci) อีก 1 บิต รวมเป็น 3 บิต เข้าด้วยกัน และจะมีเอาต์พุต 2 ตัวคือ ผลรวม (Sum) และ ตัวท (Carry out:Co) สามารถเขียนเป็นตารางการบวกได้ดังนี้

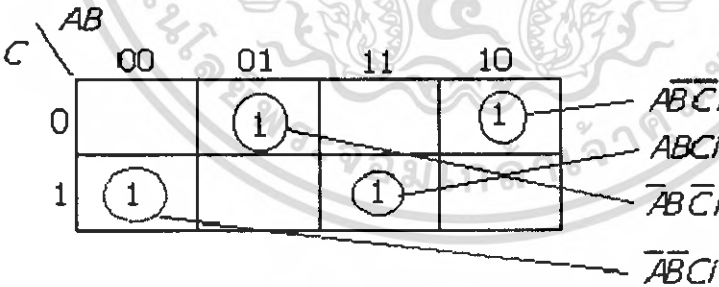
ตารางที่ ๕.15 ตารางการทำงานของ Full Adder

INPUT			OUTPUT	
A	B	Ci	Sum	Co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

จากตารางเราสามารถเขียนเป็นสมการลอจิกได้ดังนี้

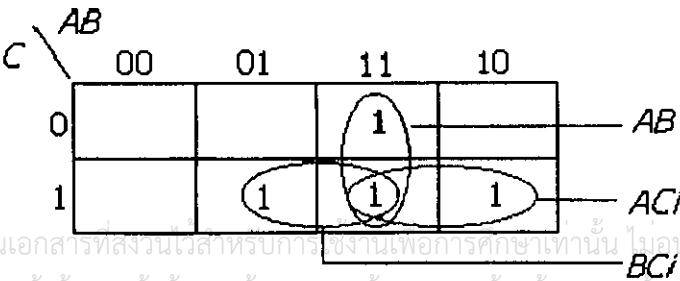
Sum = $Sum = \overline{A} \overline{B} C_i + \overline{A} B \overline{C_i} + A \overline{B} \overline{C_i} + A B C_i$

ทำการลดรูปฟังก์ชันโดยใช้ตารางคาร์นอร์

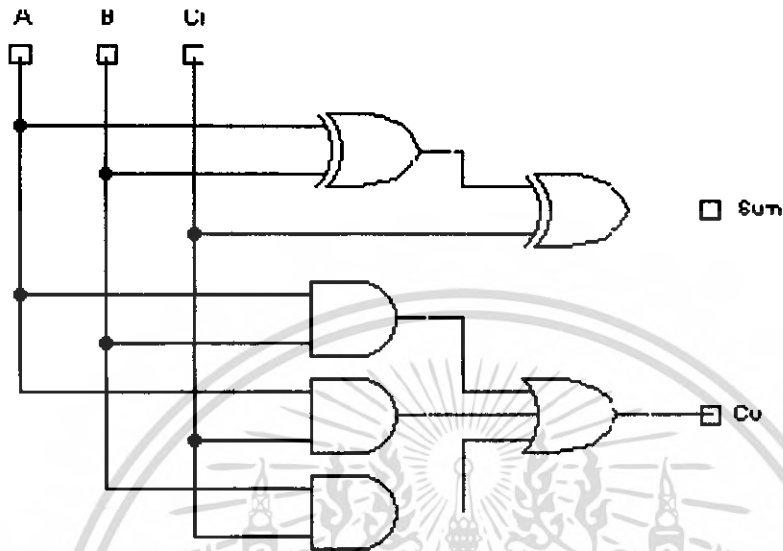


Carry = $Co = \overline{A} B C_i + A \overline{B} C_i + A B \overline{C_i} + A B C_i$

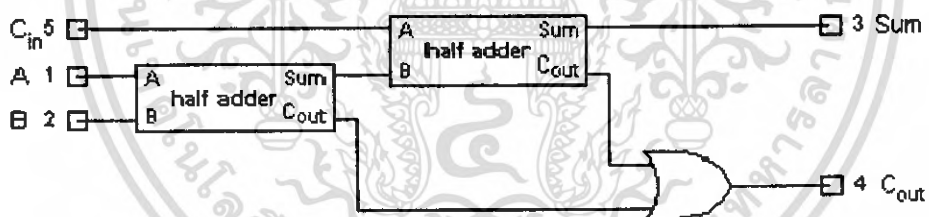
ทำการลดรูปฟังก์ชันโดยใช้ตารางคาร์นอร์



จากสมการลอจิกที่ได้นำมาเขียนเป็นวงจรได้ดังนี้



หรือบางครั้งอาจจะพบเห็นสัญลักษณ์ของวงจรบวก Full Adder ได้ดังนี้



การใช้ชุดคำสั่งของ Full adder

1.เขียนชุดคำสั่งของ Half adder ดังรูปที่ ก.80

เมื่อพิมพ์ชุดคำสั่งของ Full Adder เสร็จสมบูรณ์แล้ว ให้ทำการ Synthesize-XST เพื่อเป็นการตรวจสอบข้อผิดพลาดเมื่อไม่มีข้อผิดพลาดแล้วจึงทำการกำหนดขาของไอซีดังเช่นใบงานที่ 1 และจึงทำการกำหนดขาเพื่อแสดงเอาต์พุตของชุดคำสั่ง โดยทำขั้นตอนเช่นเดียวกับชุดคำสั่ง full adder แล้วจึงทำการดาวน์โหลดชุดคำสั่งลงไอซีเพื่อบันทึกผลการทดลองตามตารางที่ ข.16

```
19 -----
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 entity FullAdder is
31     Port(a,b,ci: in bit;
32          sum,co: out bit);
33
34 end FullAdder;
35
36 architecture Behavioral of FullAdder is
37     Signal s1,s2,s3: bit;
38     Constant gate_delay: time:=5 ns;
39
40 begin
41     L1:s1<=(a xor b) after gate_delay;
42     L2:s2<=(ci and s1) after gate_delay;
43     L3:s3<=(a and b) after gate_delay;
44     L4:sum<=(s1 xor ci) after gate_delay;
45     L5:co<=(s2 or s3) after gate_delay;
46
47 end Behavioral;
```

รูปที่ ข.80 ชุดคำสั่งของ Half adder

ตารางที่ ข.16 ค่าที่ได้จากการทดลอง Full Adder

INPUT			OUTPUT	
A	B	Ci	Sum	Co
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

สรุปผลการทดลอง

.....

.....

.....

.....

.....

.....

.....



ใบงานที่ 4

การออกแบบวงจร Subtractor

วัตถุประสงค์

1. เพื่อให้นักศึกษาใช้โปรแกรม Xilinx ISE 7.1i ได้
2. เพื่อให้ศึกษาลองการทำงานของวงจร Subtractor โดยโปรแกรม Xilinx ISE 7.1i ได้
3. เพื่อให้ศึกษานำวงจรดิจิทัลที่ออกแบบมาสร้างจริงโดยการโปรแกรมอุปกรณ์ CPLD ได้

เครื่องมือสำหรับการทดลอง

1. ระบบไมโครคอมพิวเตอร์
2. โปรแกรม Xilinx 7.1i
3. สายดาว์นโหลด
4. บอร์ดทดลอง CPLD เบอร์ XC9572PC44

ทฤษฎีพื้นฐาน

วงจร Subtractor คือวงจรที่รับตัวเลข Binary 2 bits เข้าด้วยกัน ซึ่งวงจรจะมีอยู่ 2 ประเภทคือ

3. วงจร Half Subtractor โดยสมการของวงจรมีดังนี้

$$\text{Difference} = A \oplus B$$

$$\text{Borrow} = \overline{A} \cdot B$$

Half Subtractor

- Subtracting a single-bit binary value Y from another X (I.e. $X - Y$) produces a difference bit D and a borrow out bit B-out.
- This operation is called half subtraction and the circuit to realize it is called a half subtractor.

Half Subtractor Truth Table

Inputs		Outputs	
X	Y	D	B-out
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

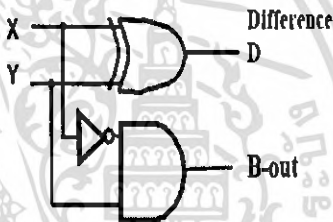
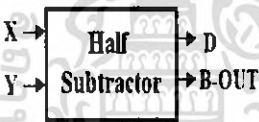
$D(X,Y) = \Sigma(1,2)$

$D = X'Y + XY'$

$D = X \oplus Y$

$B-out(x, y, C-in) = \Sigma(1)$

$B-out = X'Y$

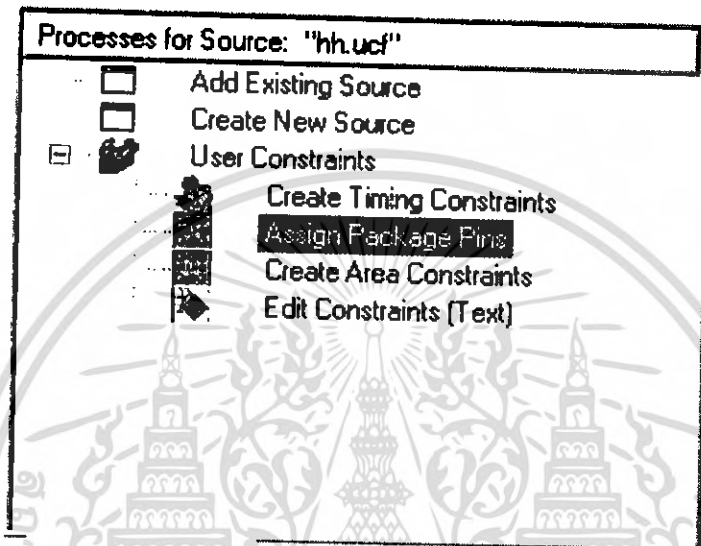


การทดลอง

1.ชุดคำสั่ง Half Subtractor ดังรูป ข.81

```
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30
31 entity HSub is
32 port(X,Y: in std_logic:= 'U';
33      D,BO: out std_logic);
34 end HSub;
35
36 architecture Behavioral of HSub is
37 begin
38   D<= X xor Y;
39   BO<=not(X)and Y;
40 end Behavioral;
```

2. เมื่อพิมพ์ชุดคำสั่งของ Half Subtractor เสร็จสมบูรณ์แล้ว ให้ทำการ Synthesize-XST เพื่อเป็นการตรวจสอบข้อผิดพลาดเมื่อไม่มีข้อผิดพลาดแล้วจึงทำการกำหนดขาของไอซี ดังเช่นใบงานที่ 1 และจึงทำการกำหนดขาเพื่อแสดงเอาต์พุตของชุดคำสั่งออกจาก Ic ทำได้โดยทำการดับเบิลคลิก Assign Package Pins เพื่อเรียกหน้าต่างที่จะทำการกำหนดขาขึ้นมา ดังรูปที่ ข.82



รูปที่ ข.82 การทำ Synthesize-XST

3. เมื่อหน้าต่างขึ้นมาแล้วจึงทำการกำหนดขาที่จะนำไปใช้งานดังรูปที่ ข.83

Design Object List - I/O Pins						
I/O Name	I/O Direction	Loc	Bank	I/O Std.	Vref	V
A	Input	p129	BANK			
B	Input	p130	BANK			
Cout	Output	p115	BANK			
SUM	Output	p116	BANK			

รูปที่ ข.83 การกำหนดขา

4. หลังจากนั้นจึงทำการโหลดชุดคำสั่งลงไปยัง IC เพื่อทำการทดลองเพื่อบันทึกผลการทดลองตามตารางที่ ข.17

ตารางที่ ข.17 ผลการทดลองของ Half Subtractor

X	Y	D	B-out
0	0		
0	1		
1	0		
1	1		

Difference = $\overline{A}B\overline{B}i + \overline{A}B\overline{B}i + \overline{A}B\overline{B}i + \overline{A}B\overline{B}i$

Borrow = $\overline{A}B + \overline{A}Bi + B\overline{B}i$

วงจร Full Subtractor

1. จากสมการทั้งสองวงจรมีสามารถนำไปเขียนเป็นวงจรดิจิทัลที่ Schematic editor โดยใช้ลอจิกเกตพื้นฐานมาประกอบกัน นอกจาก 2 วงจรนี้แล้วเรายังสามารถลบเลข binary ที่มีค่ามากกว่า 2 บิต โดยใช้ LogiBLOX Module Generator เป็นตัวสร้างโมดูลที่ใช้เลขแบบนี้มา

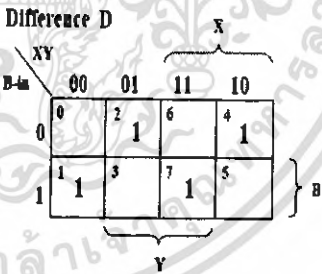
Full Subtractor

Subtracting two single-bit binary values, Y, B-in from a single-bit value X produces a difference bit D and a borrow out B-out bit. This is called full subtraction.

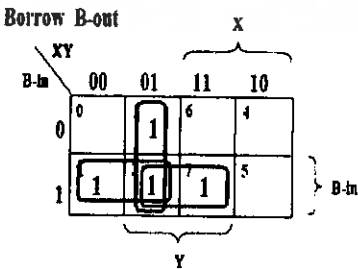
Full Subtractor Truth Table

Inputs			Outputs	
X	Y	B-in	D	B-out
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$S(X,Y,C-in) = \Sigma(1,2,4,7)$
 $C-out(x,y,C-in) = \Sigma(1,2,3,7)$



$S = X'Y'(B-in) + XY'(B-in)' + XY'(B-in) + XY(B-in)$
 $S = X \oplus Y \oplus (C-in)$



$B-out = X'Y + X'(B-in) + Y(B-in)$

2. ชุดคำสั่ง Half Subtractor ดังรูปที่ ข.84

```

20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 ---library UNISIM;
28 ---use UNISIM.VComponents.all;
29
30 entity FSub is
31   Port(X,Y,Bi: in std_logic='U';
32         Bo,D: out std_logic);
33 end FSub;
34
35 architecture Behavioral of FSub is
36 begin
37   D<=(X xor Y) xor Bi;
38   Bo<= (X and Y) or (X and Bi) or (Y and Bi);
39 end Behavioral;
40
41
42

```

รูปที่ ข.84 ชุดคำสั่ง Half Subtractor

3. เมื่อพิมพ์ชุดคำสั่งของ Half Subtractor เสร็จสมบูรณ์แล้ว ให้ทำการ Synthesize-XST เพื่อเป็นการตรวจสอบข้อผิดพลาดเมื่อไม่มีข้อผิดพลาดแล้วจึงทำการกำหนดค่าของไอซี ดังเช่นใบงานที่ 1 และจึงทำการกำหนดค่าเพื่อแสดงเอาต์พุตของชุดคำสั่งออกจาก LCD และทำการบันทึกผลการทดลองตามตารางที่ ค.

ตารางที่ ข.18 ตารางการทดลองของ Full Subtractor

A	B	Ci	Sum	Co
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

สรุปผลการทดลอง

ใบงานที่ 5

การออกแบบวงจร Encoder

วัตถุประสงค์

1. เพื่อให้นักศึกษาใช้โปรแกรม Xilinx ISE 7.1i ได้
2. เพื่อให้นักศึกษาออกแบบวงจร Encoder โดยโปรแกรม ISE 7.1i ได้
3. เพื่อให้นักศึกษานำวงจรดิจิทัลที่ออกแบบมาสร้างจริงโดยการโปรแกรมอุปกรณ์ CPLDs ได้

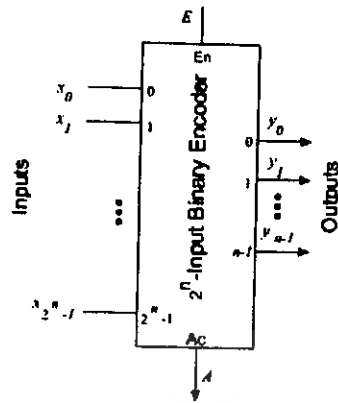
เครื่องมือสำหรับการทดลอง

1. ระบบไมโครคอมพิวเตอร์
2. โปรแกรม Xilinx ISE 7.1i
3. สายดาว์นโหลด
4. บอร์ดทดลอง CPLDs เบอร์ XC9572PC44

ทฤษฎีพื้นฐาน

ตัวเข้ารหัสไบนารี (Encoder)

ตัวเข้ารหัสไบนารี 2^n อินพุตคือวงจรที่มีไบนารีอินพุต $(x_{2n-1}, \dots, x_0)$ และมีไบนารีเอาต์พุต n $(y_{n-1}, \dots, y_0)$ โดยมันจะแปลงรหัส 1 ของรหัส 2^n ไปเป็นรหัสไบนารี โดยอินพุตมีค่า "1" บิตเดียว ดังนั้น '1' ตรงกับข้อใดของอินพุตก็จะได้อาต์พุตเป็น Binary มีค่าตรงกับของอินพุต นั้น เช่นถ้าอินพุตเป็น "1" ที่ข้อที่สองจะได้เอาต์พุต (10_2) เป็นต้น



รูปที่ ข.85 แสดงลักษณะของตัวเข้ารหัส

จงออกแบบวงจรเข้ารหัสคำสั่งของเครื่องคำนวณซึ่งมีคีย์ฟังก์ชัน 4 คีย์คือ Adder Subtrac multiply และ divide โดยกำหนดให้กดคีย์เดียวเท่านั้น หน้าที่และรหัสกำหนดได้ดังตารางที่ ข.19

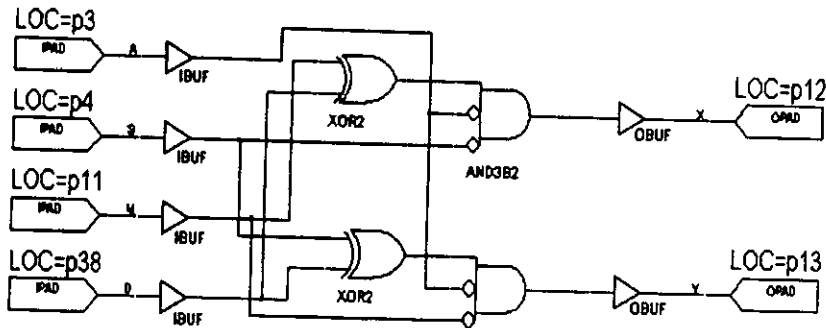
ตารางที่ ข.19 ตารางความจริงของ Encoder แบบเข้า 4 ออก 2

INPUT				OUTPUT	
A	S	M	D	X	Y
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

จากตารางที่ ค.19 สามารถเขียนสมการเพื่อสร้างวงจรลอจิกได้ดังนี้

$$X = \overline{A}S(A \oplus D)$$

$$Y = \overline{A}M(S \oplus D)$$



รูปที่ ข.86 แสดงวงจรเข้ารหัสที่ได้จากสมการในข้อ 1

1.ทำการพิมพ์ชุดคำสั่งของ Encoder แบบเข้า 4 ออก 2 ดังรูป ข.87

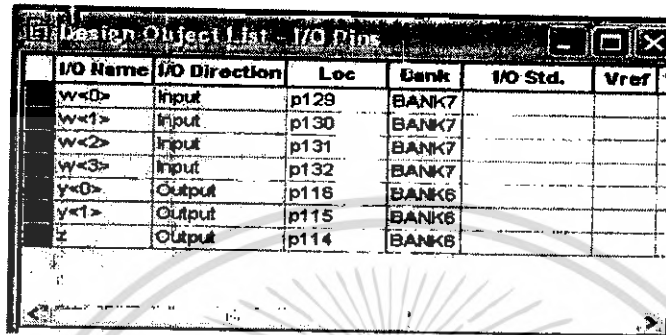
```

20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29 ENTITY priority IS
30 PORT ( w : IN STD_LOGIC_VECTOR(3 DOWNTO 0) ;
31       y : OUT STD_LOGIC_VECTOR(1 DOWNTO 0) ;
32       z : OUT STD_LOGIC ) ;
33 END priority ;
34 ARCHITECTURE Behavior OF priority IS
35 BEGIN
36 y <= "11" WHEN w(3) = '1' ELSE
37     "10" WHEN w(2) = '1' ELSE
38     "01" WHEN w(1) = '1' ELSE
39     "00" ;
40 z <= '0' WHEN w = "0000" ELSE '1' ;
41 END Behavior ;
42

```

รูปที่ ข.87 ชุดคำสั่งของ Encoder

2. ทำการ Synthesize-SXT ชุดคำสั่งที่เราทำการพิมพ์ในขั้นตอนที่หนึ่งเพื่อเป็นการตรวจสอบชุดคำสั่งว่ามีการผิดพลาดเกิดขึ้นหรือไม่และเมื่อชุดคำสั่งไม่ผิดพลาดจึงทำการกำหนดค่าเอาต์พุตเพื่อที่จะแสดงผลการทำงานดังรูป ข.88



I/O Name	I/O Direction	Loc	Bank	I/O Std.	Vref
w<0>	Input	p129	BANK7		
w<1>	Input	p130	BANK7		
w<2>	Input	p131	BANK7		
w<3>	Input	p132	BANK7		
y<0>	Output	p116	BANK6		
y<1>	Output	p115	BANK6		
z	Output	p114	BANK6		

รูปที่ ข.88 การกำหนดค่า

3. เมื่อกำหนดค่าของไอซี เสร็จแล้วดังรูปที่ ข.94 แล้วจึงทำการโปรแกรมลงไอซีและทำการบันทึกผลการทดลองตามตารางที่ ข.20

ตารางที่ ข.20 ตารางผลการทดลอง

A	B	C	D	X	Y
1	0	0	0		
0	1	0	0		
0	0	1	0		
0	0	0	1		

สรุปผลการทดลอง

.....

.....

.....

.....

.....

.....

.....



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ใบงานที่ 6

การออกแบบวงจร Decoder

วัตถุประสงค์

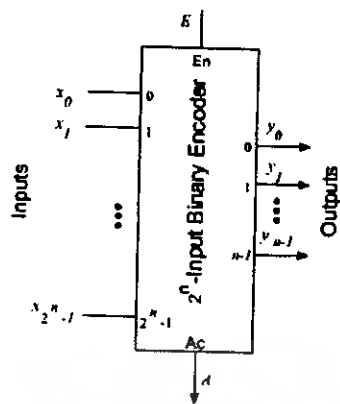
1. เพื่อให้นักศึกษาใช้โปรแกรม Xilinx ISE 7.1i ได้
2. เพื่อให้นักศึกษาออกแบบวงจร Decoder โดยโปรแกรม Xilinx ISE 7.1i ได้
3. เพื่อให้นักศึกษานำวงจรดิจิทัลที่ออกแบบมาสร้างจริงโดยการโปรแกรมอุปกรณ์ CPLD ได้

เครื่องมือสำหรับการทดลอง

1. ระบบไมโครคอมพิวเตอร์
2. โปรแกรม Xilinx ISE 7.1i
3. สายดาว์นโหลด
4. บอร์ดทดลอง CPLD เบอร์ XC9572PC44

ทฤษฎีพื้นฐาน

ตัวถอดรหัสคือตัวที่ทำการเปลี่ยนเอาต์พุตตัวเข้ารหัส n บิตเป็นเอาต์พุต 2^n บิตของตัวถอดรหัส ดังนั้นตัวแปรminatของอินพุต n บิตจะได้เอาต์พุตเท่ากับ 2^n บิต การออกแบบตัวถอดรหัสจะตรงกันข้ามกับการออกแบบตัวเข้ารหัส เพราะตัวถอดรหัส n อินพุต ส่วนตัวเข้ารหัสมี 2^n อินพุต ดังนั้นตารางเท็จจริงของตัวเข้ารหัสจะมี 2^n แถว



รูปที่ ๑.๘๙ แสดงลักษณะของตัวถอดรหัส

ออกแบบตัวถอดรหัส 2 อินพุต 4 เอาท์พุท จากตัวเข้ารหัสในตัวอย่างก่อนหน้านี้

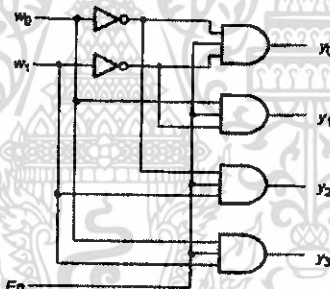
- 1. วิเคราะห์โจทย์ว่ามี 2 อินพุต ให้เป็น x,y และ 4 เอาท์พุทให้เป็น A S M D
- 2. เขียนตารางเท็จจริงได้ดังตาราง

En	w ₁	w ₀	y ₃	y ₂	y ₁	y ₀
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1
0	x	x	0	0	0	0

(a) Truth table



(b) Graphical symbol



(c) Logic circuit

1. ทำการพิมพ์ชุดคำสั่งของ Decoder ดังรูป ข.90

```

20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29 ENTITY dec2to4 IS
30 PORT ( w : IN STD_LOGIC_VECTOR(1 DOWNTO 0) ;
31       En : IN STD_LOGIC ;
32       y : OUT STD_LOGIC_VECTOR(0 TO 3) ) ;
33 END dec2to4 ;
34 ARCHITECTURE Behavior OF dec2to4 IS
35 SIGNAL Enw : STD_LOGIC_VECTOR(2 DOWNTO 0) ;
36 BEGIN
37   Enw <= En & w ;
38   WITH Enw SELECT
39   y <= "1000" WHEN "100",
40       "0100" WHEN "101",
41       "0010" WHEN "110",
42       "0001" WHEN "111",
43       "0000" WHEN OTHERS ;
44 END Behavior ;

```

รูปที่ ข.90 ชุดคำสั่งของ Decoder

2. ทำการ Synthesize-SXT ชุดคำสั่งที่เราทำการพิมพ์ในขั้นตอนที่หนึ่งเพื่อเป็นการตรวจสอบชุดคำสั่งว่ามีการผิดพลาดเกิดขึ้นหรือไม่และเมื่อชุดคำสั่งไม่ผิดพลาดจึงทำการกำหนดขาเอาต์พุตเพื่อที่แสดงผลการทำงานดังรูป ข.91

I/O Name	I/O Direction	Loc	Bank	I/O Std.	Vref
En	Input	p129	BANK7		
w<0>	Input	p130	BANK7		
w<1>	Input	p131	BANK7		
y<0>	Output	p116	BANK6		
y<1>	Output	p115	BANK6		
y<2>	Output	p114	BANK6		
y<3>	Output	p113	BANK6		

รูปที่ ข.91 การกำหนดขาอุปกรณ์

3. เมื่อกำหนดขาของไอซี เสร็จแล้วดังรูปที่ ข.90 แล้วจึงทำการโปรแกรมลงไอซีและทำการบันทึกผลการทดลองตามตารางที่ ข.21

ตารางที่ ข.21 ตารางผลการทดลองวงจร Decoder

X	Y	Z	Y1	Y2	Y3	Y4
1	0	0				
1	0	1				
1	1	0				
1	1	1				
0	X	X				

สรุปผลการทดลอง

ใบงานที่ 7

การออกแบบวงจร Multiplex

วัตถุประสงค์

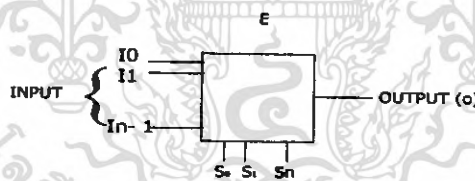
1. เพื่อให้นักศึกษาใช้โปรแกรม Xilinx ISE 7.1i ได้
2. เพื่อให้นักศึกษาออกแบบวงจร Multiplex โดยโปรแกรม Xilinx ISE 7.1i ได้
3. เพื่อให้นักศึกษานำวงจรดิจิทัลที่ออกแบบมาสร้างจริงโดยการโปรแกรมอุปกรณ์ CPLDs ได้

เครื่องมือสำหรับการทดลอง

1. ระบบไมโครคอมพิวเตอร์
2. โปรแกรม Xilinx ISE 7.1i
3. สายดาว์นโหลด
4. บอร์ดทดลอง CPLDs เบอร์ XC9572PC44

ทฤษฎีพื้นฐาน

หลักการของมัลติเพล็กซ์ (MUX) หลักการของมัลติเพล็กซ์แสดงดังรูปที่ ข.97



รูปที่ ข.92 แสดงไดอะแกรมของตัวมัลติเพล็กซ์

จากรูปที่ 8.1 สามารถแสดงความสัมพันธ์ต่างๆได้ดังนี้

$$O = m_0 I_0 + m_1 I_1 + \dots + m_{n-1} I_{n-1} \quad (1)$$

ในการใช้งานจริงอาจมีขา E (Enable) เพิ่มเข้ามาดังนั้น

$$O = m_0 I_0 E + m_1 I_1 E + \dots + m_{n-1} I_{n-1} E \quad (2)$$

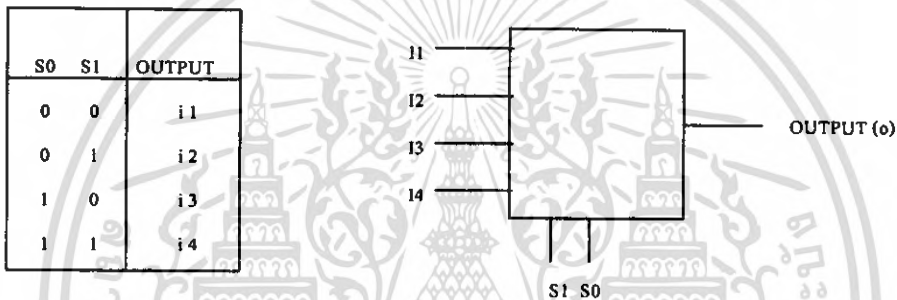
โดยที่ m_i คือมินเทอมตัวที่ i ของตัวแปรประกอบ ของรหัสเลือกเมื่อ $i = 0$ ถึง $N-1$ และ

$$m_0 = S_{n-1} S_{n-2} \dots S_1 S_0$$

$$\begin{aligned}
 m_1 &= S_{n-1}S_{n-2}\dots S_1S_0 \\
 m_2 &= S_{n-1}S_{n-2}\dots S_1S_0 \\
 m_n &= S_{n-1}S_{n-2}\dots S_1S_0
 \end{aligned}
 \tag{3}$$

การทดลองที่ 8.1 ออกแบบวงจรมัลติเพล็กซ์ที่มี 4 อินพุต และ 1 เอาท์พุต เราทราบว่า 4 อินพุตเขียนในรูป $2n$ ได้ 22 ดังนั้น $n = 2$ แสดงว่ามีตัวเลือก 2 เส้น เราเขียนตารางที่จริงได้ดังตารางที่ ข.22 และแสดงไดอะแกรมดังรูปที่ ข.93

ตารางที่ ข.22 แสดงตารางที่จริงของไดอะแกรมวงจรมัลติเพล็กซ์ 4 อินพุต 1 เอาท์พุต



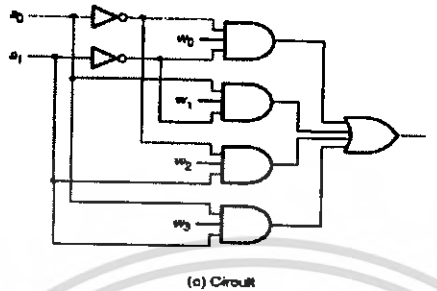
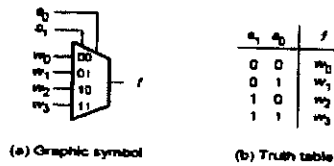
จากตารางเราจะได้

$$O = i_0m_0 + i_1m_1 + i_2m_2 + i_3m_3$$

เพราะฉะนั้น

$$O = i_1\overline{S_0}S_1 + i_2\overline{S_0}S_1 + i_3S_0\overline{S_1} + i_4S_0S_1 \tag{4}$$

จากสมการบูลีนเอาท์พุตสามารถเขียนแผนผังเกทได้ดังรูปที่ ข.99



รูปที่ ข.93 แผนผังเกท

1. ทำการพิมพ์ชุดคำสั่งของ Multiplex ดังรูป ข.94

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

ENTITY mux4to1 IS
PORT ( w0, w1, w2, w3 : IN STD_LOGIC ;
      s : IN STD_LOGIC_VECTOR(1 DOWNTO 0) ;
      f : OUT STD_LOGIC ) ;
END mux4to1 ;

ARCHITECTURE Behavior OF mux4to1 IS
BEGIN
WITH s SELECT
    f <= w0 WHEN "00",
        w1 WHEN "01",
        w2 WHEN "10",
        w3 WHEN OTHERS ;
END Behavior ;

```

รูปที่ ข.94 คำสั่งของ Multiplex

2. ทำการ Synthesize-SXT ชุดคำสั่งที่เราทำการพิมพ์ในขั้นตอนที่หนึ่งเพื่อเป็นการตรวจสอบชุดคำสั่งว่ามีการผิดพลาดเกิดขึ้นหรือไม่และเมื่อชุดคำสั่งไม่ผิดพลาดจึงทำการกำหนดค่าเอาต์พุตเพื่อที่จะแสดงผลการทำงานดังรูป ข.95

I/O Name	I/O Direction	Loc	Bank	I/O Std.	Vref
f	Output	p116	BANK6		
s<0>	Input	p129	BANK7		
s<1>	Input	p130	BANK7		
w0	Input	p131	BANK7		
w1	Input	p132	BANK7		
w2	Input	p133	BANK7		
w3	Input	p134	BANK7		

รูปที่ ข.95 การกำหนดขา

3. เมื่อกำหนดขาของไอซี เสร็จแล้วดังรูปที่ ข.101 แล้วจึงทำการโปรแกรมลงไอซี

ตามตารางที่ ข.22 ตารางแสดงผลการทดลองของวงจร Multiplex

สรุปผลการทดลอง

S1	S0	I0	I2	I2	I3	Y
0	0	0	0	0	0	
0	1	0	0	0	1	
1	0	0	0	1	0	
1	1	0	0	1	1	

ใบงานที่ 8

การออกแบบวงจร Demultiplex

วัตถุประสงค์

1. เพื่อให้นักศึกษาใช้โปรแกรม Xilinx ISE 7.1i ได้
2. เพื่อให้นักศึกษาออกแบบวงจร Demultiplex เบื้องต้นโดยโปรแกรม Xilinx ISE 7.1i ได้
3. เพื่อให้นักศึกษานำวงจรดิจิทัลที่ออกแบบมาสร้างจริงโดยการโปรแกรมอุปกรณ์ CPLDs ได้

เครื่องมือสำหรับการทดลอง

1. ระบบไมโครคอมพิวเตอร์
2. โปรแกรม Xilinx ISE 7.1i
3. สายดาว์นโหลด
4. บอร์ดทดลอง CPLDs เบอร์ XC9572

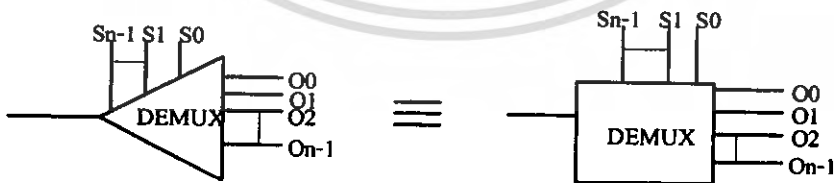
ทฤษฎีพื้นฐาน

ดีมัลติเพล็กซ์ (DEMUX)

เป็นวงจรที่มี 1 อินพุตแต่มีหลายเอาต์พุต ทำงานโดยการส่ง 1 อินพุตไปยังเอาต์พุตที่ถูกเลือก อาจเทียบได้กับ data router ถ้าหากเราใช้อินพุตจำนวน I ขั้ว และ เอาต์พุตเป็น N ขั้ว ($O_0, O_1, \dots, O_{n-1}$) ดังนั้น ขั้วออกเท่ากับ $N = 2^n$ ขั้ว แล้วต้องใช้อินพุตเลือก n บิตคือ $S_{n-1} \dots S_1 S_0$ โดยที่ $O = ($ มินิเทอมของ i ของตัวแปร $S) I$ จากสมการ จะเขียนเป็นไดอะแกรมได้ดังรูปที่ ข.102

$$O = m_i I = S_{n-1} S_{n-2} \dots S_1 S_0 I$$

$$O = m_i I = S_{n-1} S_{n-2} \dots S_1 S_0 I$$



รูปที่ ข.96 แสดงไดอะแกรมของตัวดีมัลติเพล็กซ์

1. ทำการพิมพ์ชุดคำสั่งของ DMultiplex ดังรูป ข.97

```

20      library IEEE;
21      use IEEE.STD_LOGIC_1164.ALL;
22      use IEEE.STD_LOGIC_ARITH.ALL;
23      use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25      ---- Uncomment the following library declaration if instantiating
26      ---- any Xilinx primitives in this code.
27      --library UNISIM;
28      --use UNISIM.VComponents.all;
29
30      entity Dmux is
31      Port (S0,S1,Din: IN std_logic:='U';
32           D0,D1,D2,D3:OUT std_logic);
33      end Dmux;
34
35      architecture Behavioral of Dmux is
36
37      begin
38      D0<=not(S0)and not(s1)and Din;
39      D1<=not(S1)and(S0)and Din;
40      D2<=S1 and not(S0)and Din;
41      D3<=S1 and S0 and Din;
42      end Behavioral;

```

รูปที่ ข.97 ชุดคำสั่งของ DMultiplex

2. ทำการ Synthesize-SXT ชุดคำสั่งที่เราทำการพิมพ์ในขั้นตอนที่หนึ่งเพื่อเป็นการตรวจสอบชุดคำสั่งว่ามีการผิดพลาดเกิดขึ้นหรือไม่และเมื่อชุดคำสั่งไม่ผิดพลาดจึงทำการกำหนดขาเอาต์พุตเพื่อที่จะแสดงผลการทำงาน ดังรูปที่ ข.98

I/O Name	I/O Direction	Loc	Bank	I/O Std.	Vref	V
Din	Input	P131				
D0	Output	P116	BANK			
D1	Output	P115	BANK			
D2	Output	P114	BANK			
D3	Output	P113	BANK			
S0	Input	P129	BANK			
S1	Input	P130	BANK			

รูปที่ ข.98 การกำหนดขา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3. เมื่อกำหนดขาของไอซี เสร็จดังรูป ข.104 แล้วจึงทำการโปรแกรมลงบนไอซีและทำการบันทึกผลการทดลอง

ตารางที่ ข.23 ตารางผลการทดลองของวงจร DMultiplex

S0	S1	Din	D0	D1	D2	D3
0	0	1				
0	1	1				
1	0	1				
1	1	1				

สรุปผลการทดลอง

.....

.....

.....

.....

.....

.....

.....

.....

ใบงานที่ 9

การออกแบบวงจร BCD to 7-Segment

วัตถุประสงค์

1. เพื่อให้นักศึกษาใช้โปรแกรม Xilinx ISE 7.1i ได้
2. เพื่อให้นักศึกษาออกแบบวงจร BCD to 7-Segment โดยโปรแกรม Xilinx ISE 7.1i ได้
3. เพื่อให้นักศึกษานำวงจรดิจิทัลที่ออกแบบมาสร้างจริงโดยการโปรแกรมอุปกรณ์ CPLDs ได้

เครื่องมือสำหรับการทดลอง

1. ระบบไมโครคอมพิวเตอร์
2. โปรแกรม Xilinx ISE 7.1i
3. สายดาวน์โหลด
4. บอร์ดทดลอง CPLDs เบอร์ XC9572

ทฤษฎีพื้นฐาน

วงจรถอดรหัส BCD เป็นตัวเลข 7 ส่วน (BCD to 7-Segment Decoder) ทำหน้าที่แปลงรหัส BCD 8421 ให้เป็นตัวเลขฐานสิบ โดยแสดงผลทางแอลอีดีแบบเจ็ดส่วน (7-Segment LED) ที่อยู่ 2 ประเภท คือประเภท Common Anode และ ประเภท Common Cathode

7-Segment LED ภายในประกอบไปด้วย LED จำนวนทั้งหมด 8 ดวงคือดวง a,b,c,d,e,f,g และ dp หรือ dot

การออกแบบวงจรถอดรหัส BCD เป็นตัวเลข 7 ส่วน (BCD to 7-Segment Decoder) การออกแบบวงจรจะต้องสัมพันธ์กับประเภทของ 7-Segment ที่ใช้งาน ดังตารางความจริงที่ ข.24

ตารางที่ ๑.24 ตารางการทำงาน BCD to 7-Segment Decoder

D	C	B	A	a	b	c	d	e	f	g
0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	1	1	1	1
0	0	1	0	0	0	1	0	0	1	0
0	0	1	1	0	0	0	0	1	1	0
0	1	0	0	1	0	0	1	1	0	0
0	1	0	1	0	1	0	0	1	0	0
0	1	1	0	1	1	0	0	0	0	0
0	1	1	1	0	0	0	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	1	1	0	0

1. ทำการพิมพ์ชุดคำสั่งของวงจรถอดรหัส BCD เป็นตัวเลข 7 ส่วน แบบ Anode ดังรูปที่ ๑.99

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity bcd7 is
    port(
        D:in STD_LOGIC_VECTOR(3 downto 0);
        Y:out STD_LOGIC_VECTOR(6 downto 0));
    end bcd7;
architecture Behavioral of bcd7 is
begin
    with D Select
        Y<="1111001" when "0001",
            "0100100" when "0010",
            "0110000" when "0011",
            "0011001" when "0100",
            "0010010" when "0101",
            "0000011" when "0110",
            "1111000" when "0111",
            "0000000" when "1000",
            "0011000" when "1001",
            "1000000" when others;
end Behavioral;

```

รูปที่ ๑.99 ชุดคำสั่งของวงจรถอดรหัส BCD แบบ แอนโนด

2 เมื่อพิมพ์ชุดคำสั่งเสร็จแล้วจึงทำการบันทึกผลการทดลองดังตารางที่ ข.25

ตารางที่ ข.25 ผลการทดลอง วงจรถอดรหัส BCD แบบ แอนได

D	C	B	A	a	b	c	D	e	f	g
0	0	0	0							
0	0	0	1							
0	0	1	0							
0	0	1	1							
0	1	0	0							
0	1	0	1							
0	1	1	0							
0	1	1	1							
1	0	0	0							
1	0	0	1							

3. ทำการพิมพ์ชุดคำสั่งของวงจรถอดรหัส BCD เป็นตัวเลข 7 ส่วน แบบ แคโทด ดังรูปที่ ข.100

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity bcd_7 is
port(
D:in STD_LOGIC_VECTOR(3 downto 0);
Y:out STD_LOGIC_VECTOR(6 downto 0));
end bcd_7;
architecture Behavioral of bcd_7 is
begin
with D Select
Y<="0000110" when "0001",
"1011011" when "0010",
"1001111" when "0011",
"1100110" when "0100",
"1101101" when "0101",
"1111100" when "0110",
"0000111" when "0111",
"1111111" when "1000",
"1100111" when "1001",
"0111111" when others;
end Behavioral;

```

รูปที่ ข.100 ชุดคำสั่งของวงจรถอดรหัส BCD แบบ แคโทด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4. เมื่อพิมพ์ชุดคำสั่งเสร็จแล้วจึงทำการบันทึกผลการทดลองดังตารางที่ ข.26

ตารางที่ ข.26 ผลการทดลอง วงจรถอดรหัส BCD แบบ แคโทด

D	C	B	A	a	b	c	d	e	f	g
0	0	0	0							
0	0	0	1							
0	0	1	0							
0	0	1	1							
0	1	0	0							
0	1	0	1							
0	1	1	0							
0	1	1	1							
1	0	0	0							
1	0	0	1							

สรุปผลการทดลอง

ใบงานที่ 10

การทำงานของ Flip Flop

วัตถุประสงค์

1. เพื่อให้นักศึกษาใช้โปรแกรม Xilinx ISE 7.1i ได้
2. เพื่อให้นักศึกษา ศึกษาการทำงานของ Flip Flop โดยโปรแกรม Xilinx ISE 7.1i ได้
3. เพื่อให้นักศึกษานำวงจรดิจิทัลที่ออกแบบมาสร้างจริงโดยการโปรแกรมอุปกรณ์ CPLDs ได้

เครื่องมือสำหรับการทดลอง

1. ระบบไมโครคอมพิวเตอร์
2. โปรแกรม Xilinx ISE 7.1i
3. สายดาว์นโหลด
4. บอร์ดทดลอง CPLDs เบอร์ XC9572

ทฤษฎีพื้นฐาน

ฟลิป ฟลอป (Flip Flop) เป็น multivibrator ชนิด Bistable คือเอาต์พุตลงที่ 2 สภาวะ สำหรับเอาต์พุตทั้งสองจะตั้งเงื่อนไขว่าเอาต์พุตหนึ่งจะเป็น Complement ของอีกเอาต์พุตหนึ่ง ในกรณีใดก็ตามหากเอาต์พุตหนึ่งไม่เป็น Complement ของอีกเอาต์พุตหนึ่งแล้ว เราจะไม่ยอมให้เกิดขึ้น เอาต์พุตทั้งสองดังกล่าวจะคงอยู่ในสภาวะอันใดอันหนึ่งจนกว่าจะมีอินพุต ฟลัส มากระตุ้นให้เปลี่ยนสภาวะไป การที่เอาต์พุตมีสภาวะคงตัว 2 สภาวะและคงอยู่ในสภาวะใดสภาวะหนึ่งโดยไม่จำกัดเวลา เราจึงกล่าวได้ว่ามันสามารถจดจำสภาวะที่เกิดขึ้นเมื่อ Pulse อันสุดท้ายเมื่อป้อนเข้ามากระตุ้นที่อินพุตได้

วงจร Flip Flop ที่พบในอุปกรณ์ Digital นั้นมีหลายแบบด้วยกัน แต่ละแบบก็ให้เอาต์พุตที่มีสภาวะคงตัว 2 สภาวะเหมือนกันหมด แต่แตกต่างกันตรงที่ต้องการสภาวะทางอินพุตเพื่อให้เอาต์พุตเปลี่ยนสภาวะคงตัวจากอันหนึ่งไปยังอีกอันหนึ่งแตกต่างกัน Flip Flop ในโปรแกรม Xilinx ISE 7.1i จะมีอยู่ 4 ชนิดด้วยกันคือ

1. D Flip Flop
2. T Flip Flop
3. JK Flip Flop
4. RS Flip Flop

ขั้นตอนการทดลอง

1. D Flip Flop

1.1 D Flip Flop ให้ทำการพิมพ์ชุดคำสั่งของ D Flip Flop ดังรูปที่ ข.101

```
entity d_ff is
  port (
    d:in std_logic;
    clk:in std_logic;
    q:out std_logic;
    q_bar:out std_logic
  );
end d_ff;

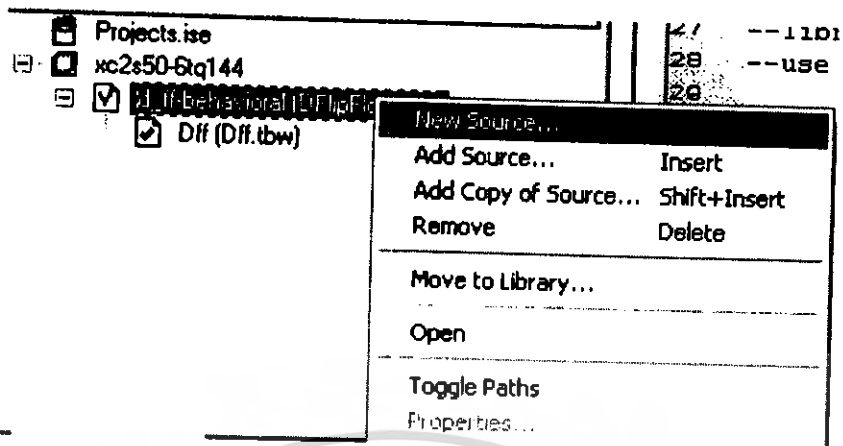
architecture Behavioral of d_ff is
  signal temp: std_logic;
begin
  process(clk)
  begin
    if clk'event and clk='1' then
      temp <= d;
    end if;
  end process;
  q <= temp;
  q_bar <= not temp;
end Behavioral;
```

รูปที่ ข.101 ชุดคำสั่งของ D Flip Flop

1.2 ทำการ Synthesize-SXT ชุดคำสั่งที่เราทำการพิมพ์ในขั้นตอนที่หนึ่งเพื่อเป็นการตรวจสอบชุดคำสั่งว่ามีการผิดพลาดเกิดขึ้นหรือไม่และเมื่อชุดคำสั่งไม่ผิดพลาดจึงทำการทดลองโดยการทดลองการทำงานด้วย ModelSim XE III 6.0a ในการจำลองการทำงาน

1.3 ทำการสร้างโปรเจกขึ้นมาใหม่โดยทำการคลิกขวาที่โปรเจก เลือกที่ New Source...ดังรูปที่

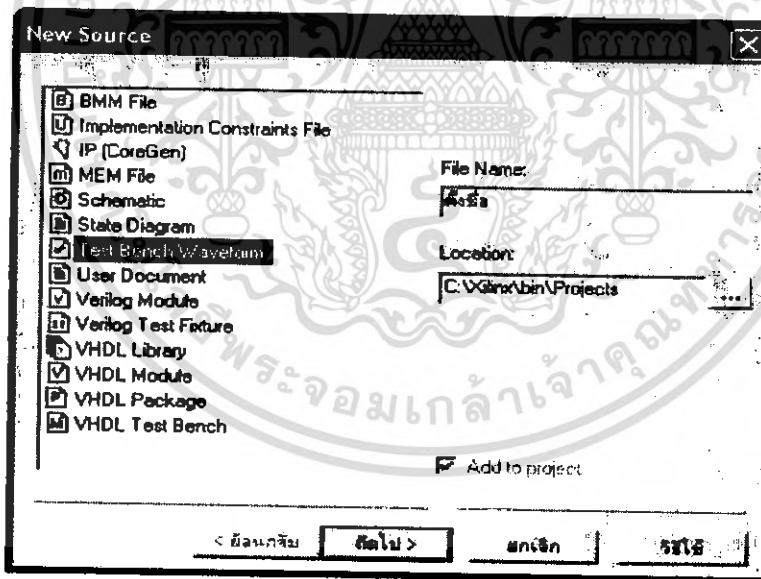
ข.103



รูปที่ ข.102 New Source...

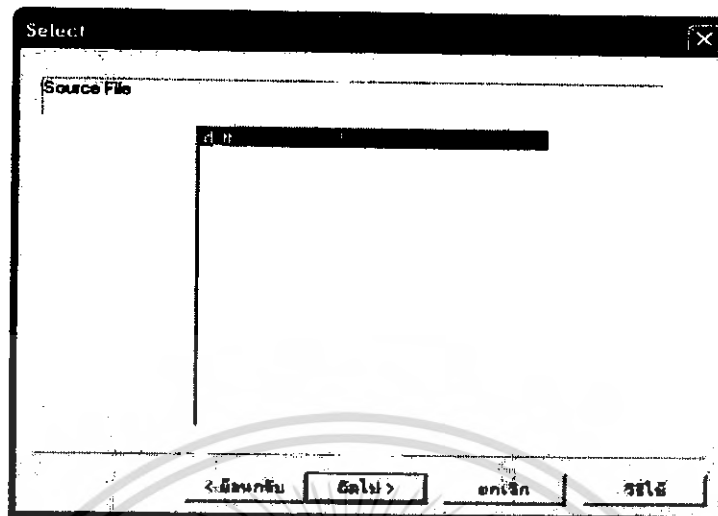
ถัดไป

1.4 ทำการกำหนดชื่อให้กับชุดคำสั่งเพื่อที่จะทดลองผลการทำงาน รูปที่ ข.103 และคลิกที่คำว่า



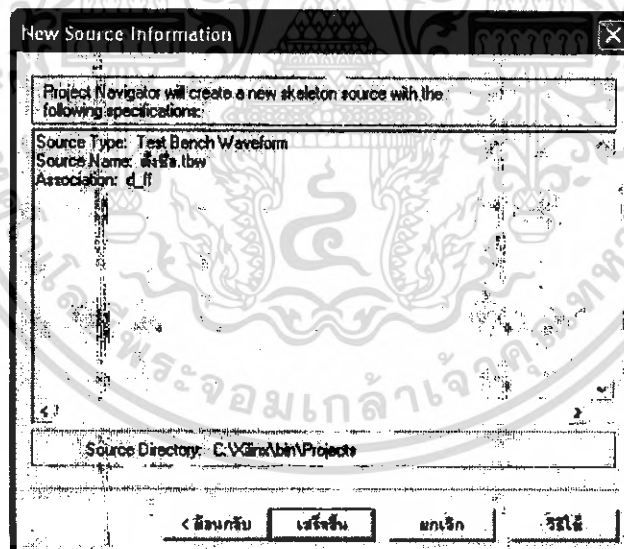
รูปที่ ข.103 การตั้งชื่อ

1.5 กำหนดการเข้าไปอยู่ในโปรเจกต์หลักดังรูปที่ ข.104 และคลิกที่คำว่าถัดไปเพื่อยืนยัน



รูปที่ ข.104 การเลือกบันทึกของหัวข้อโปรเจกใหม่

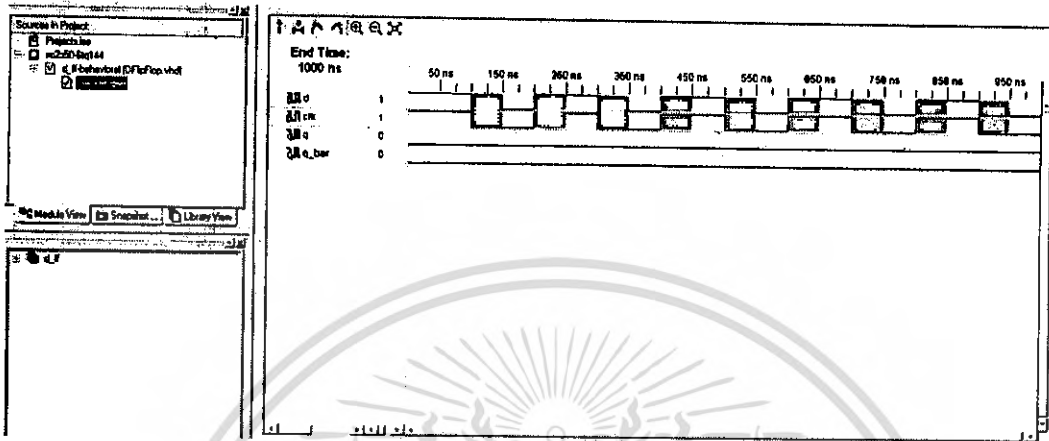
1.6 คลิกคำสั่ง เสร็จสิ้นเพื่อทำการยืนยันดังรูป ข.1065



รูปที่ ข.105 การยืนยันการสร้างโปรเจกใหม่

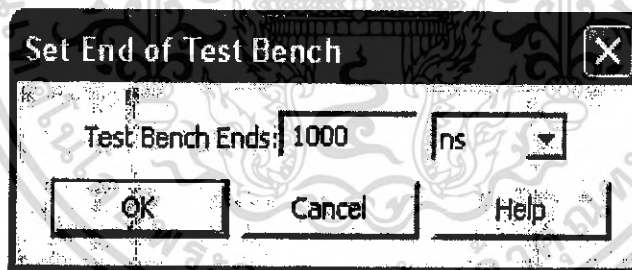
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.7 กำหนดสัญญาณนาฬิกาที่ขาอินพุต ดังรูปที่ ข.106



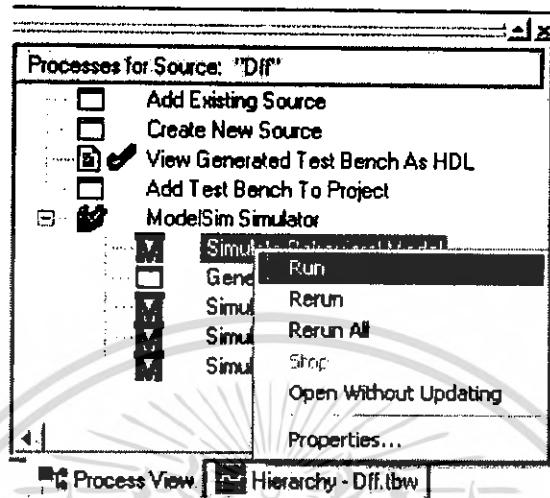
รูปที่ ข.106 การกำหนดสัญญาณอินพุต

1.8 กำหนดการสิ้นสุดของขั้นตอนการจำลองสัญญาณนาฬิกา ดังรูปที่ ข.107



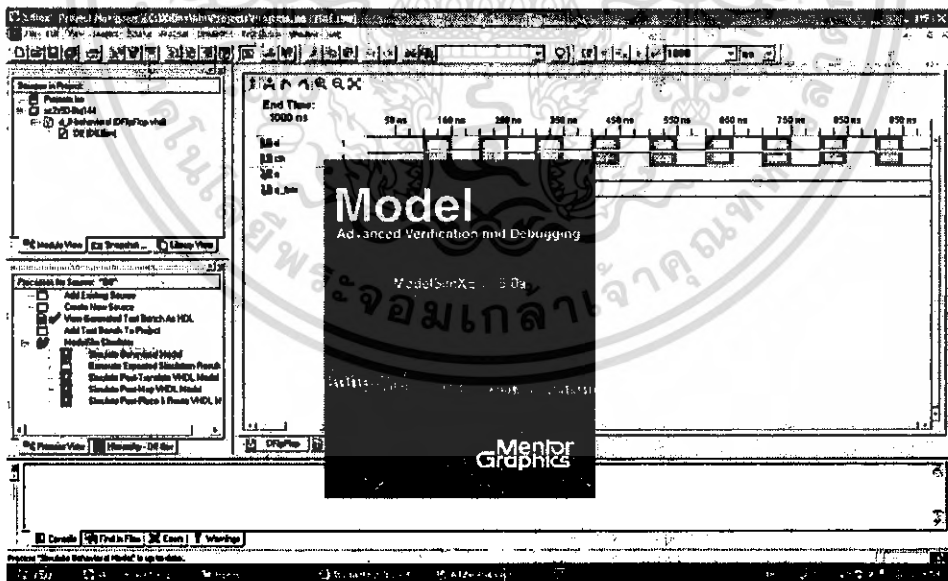
รูปที่ ข.107 การยืนยันการสิ้นสุดการกำหนดสัญญาณ

1.9 การเรียกเข้าขั้นตอนการ ModelSim ดังรูปที่ ข.108



รูปที่ ข.108 ขั้นตอนการ Modlesim

1.10 แสดงหน้าจอการเข้า Modlesim ดังรูปที่ ข.109



รูปที่ ข.109 หน้าจอการเข้า Modlesim

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. RS Flip Flop ให้ทำการพิมพ์ชุดคำสั่งของ RS Flip Flop ดังรูปที่ ข.111

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity RS is
port (clk:in std_logic;
      s:in std_logic;
      r:in std_logic;
      q:out std_logic;
      q_bar:out std_logic
    );
end RS;

architecture Behavioral of RS is
    signal temp : std_logic;
    signal aa: std_logic_vector(1 downto 0);
begin
    aa <= r & s;

```

รูปที่ ข.111 ชุดคำสั่งของ RS Flip Flop

```

    process (clk,r,s)
    begin
        if clk'event and clk='1'then
            if(aa = "10")then
                temp <= '0';
            elsif(aa = "01")then
                temp <= '1';
            elsif(aa = "00") then
                temp <= temp;
            end if;
        end if;
    end process;

    q <= temp;
    q_bar<=not temp;

end Behavioral;

```

รูปที่ ข.112 ชุดคำสั่งของ RS Flip Flop (ต่อ)

บันทึกผลการทดลอง

.....

.....

.....

.....

.....

.....

.....

.....

3. JK Flip Flop ให้ทำการพิมพ์ชุดคำสั่งของ JK Flip Flop ดังรูปที่ ข.113

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity JK is
port (j:in std_logic;
      clk:in std_logic;
      k:in std_logic;
      q:out std_logic;
      q_bar: out std_logic);
end JK;

architecture Behavioral of JK is
    signal temp: std_logic;
    signal aa: std_logic_vector(1 downto 0);
begin
    aa <= ( j & k);
    process (clk)
    begin

```

รูปที่ ข.113 ชุดคำสั่งของ JK Flip Flop

```

        if clk'event and clk='1' then

            if(aa = "11") then
                temp <= not temp;
            elsif(aa = "10") then
                temp <= '1';
            elsif(aa = "01") then
                temp <= '0';
            else
                temp <= temp;
            end if;
        end if;
    end process;
    q <= temp;
    q_bar <= not temp;
end Behavioral;

```

รูปที่ ข.114 ชุดคำสั่งของ JK Flip Flop (ต่อ)

บันทึกผลการทดลอง

4. T Flip Flop ให้ทำการพิมพ์ชุดคำสั่งของ T Flip Flop ดังรูปที่ ข.115

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity t is
port (clk:in std_logic;
      q:out std_logic;
      q_bar:out std_logic
      );
end t;

architecture Behavioral of t is
    signal temp: std_logic := '0';
begin
    process (clk)
    begin
        if clk'event and clk='1' then
            temp <= not temp;
        end if;
    end process;
    q <= temp;
    q_bar<=not temp;
end Behavioral;

```

รูปที่ ข.115 ชุดคำสั่งของ T Flip Flop

บันทึกผลการทดลอง

[illegible]

สรุปผลการทดลอง

.....

.....

.....

.....

.....

.....

.....



ใบงานที่ 11

การออกแบบวงจร Counter

วัตถุประสงค์

1. เพื่อให้นักศึกษาใช้โปรแกรม Xilinx ISE 7.1i ได้
2. เพื่อให้นักศึกษาออกแบบวงจร Counter โดยโปรแกรม Xilinx ISE 7.1i ได้
3. เพื่อให้นักศึกษานำวงจรดิจิทัลที่ออกแบบมาสร้างจริงโดยการโปรแกรมอุปกรณ์ CPLD ได้

เครื่องมือสำหรับการทดลอง

1. ระบบไมโครคอมพิวเตอร์
2. โปรแกรม Xilinx ISE 7.1i
3. สายดาวน์โหลด
4. บอร์ดทดลอง CPLDs เบอร์ XC9572

ทฤษฎีพื้นฐาน

วงจรมีสามารถแบ่งออกเป็น 2 ประเภทใหญ่ๆ ตามลักษณะการทำงานและการต่อสัญญาณนาฬิกาเข้าสู่วงจรคือ วงจรมีแบบอิงโครนัส และวงจรมีแบบอิงโครนัส

วงจรมีแบบอิงโครนัสเป็นวงจรมีที่ทำการนับเป็นระลอก (Rip pie) วงจรมีประเภทนี้สามารถตั้งค่านับได้โดยการใช้สัญญาณป้อนกลับ (Feedback) จากเอาต์พุตของการนับมา Clear วงจรมีให้เริ่มต้นการนับใหม่ อาจเรียกวงจรมีในลักษณะนี้ว่าวงจรมีแบบโมดูลัส หรือที่เรียกสั้น ๆ ว่าวงจรมีแบบ MOD-N ตัวอย่างของวงจรมีแบบ MOD-N สามารถยกตัวอย่างได้ คือ วงจรมี MOD-N จะเป็นวงจรมี 0-5 และสภาวะที่นับ 6 (110_2) จะนำมาใช้ในการออกแบบวงจรมีป้อนกลับมา Clear วงจรมีให้เอาต์พุตทุกตัวเป็น 0 หมด

วงจรมีแบบอิงโครนัสเป็นวงจรมีที่วงจรมีลืปฟอปทุกตัวทำงานพร้อมๆ กันเพราะว่าวงจรมีประเภทนี้จะรับสัญญาณนาฬิกาที่ตำแหน่งเดียวกัน วงจรมีแบบอิงโครนัสสามารถนำไปใช้งานที่มีการทำงานในความถี่สูง ๆ ได้ดีกว่าวงจรมีแบบอิงโครนัส เพราะวงจรมีแบบอิงโครนัส เกิดจากการนำวงจรมีลืปฟอปแต่ละวงจรมีมาต่ออันดับกัน การเปลี่ยนสภาวะของวงจรมีลืปฟอปแต่ละวงจรมี จะต้องรอการเปลี่ยนสภาวะจากวงจรมีลืปฟอปที่ต่อเป็นวงจรมีหน้าก่อน การทำงานในลักษณะนี้จึงไม่สามารถทำงานในวงจรมีที่มีการทำงานในความถี่สูงๆ ได้

1. ทำการพิมพ์ชุดคำสั่งนับดังรูปที่ ข.116

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity CounterProcess is
    Port ( CLK_1Hz : in std_logic;
          SevenSegment : out std_logic_vector(6 downto 0);
          Common : out std_logic
        );
end CounterProcess;

architecture Behavioral of CounterProcess is
    signal Counter : std_logic_vector(3 downto 0);
begin
    process (CLK_1Hz)
    begin
        if (CLK_1Hz'event and CLK_1Hz = '1') then
            if (Counter >= 9) then
                Counter <= "0000";
            else
                Counter <= Counter + 1;
            end if;
        end if;
    end process;
end architecture;

```

รูปที่ ข.116 ชุดคำสั่งนับ

```

end if;
end if;
end process;

process(Counter)
begin
    case Counter is
        when "0001" => SevenSegment <= "0000110";
        when "0010" => SevenSegment <= "1011011";
        when "0011" => SevenSegment <= "1001111";
        when "0100" => SevenSegment <= "1100110";
        when "0101" => SevenSegment <= "1101101";
        when "0110" => SevenSegment <= "1111101";
        when "0111" => SevenSegment <= "0000111";
        when "1000" => SevenSegment <= "1111111";
        when "1001" => SevenSegment <= "1101111";
        when others => SevenSegment <= "0111111";
    end case;
end process;

Common <= '0';

end Behavioral;

```

รูปที่ ข.117 ชุดคำสั่งนับ (ต่อ)

บันทึกผลการทดลอง

สรุปผลการทดลอง

.....

.....

.....

.....

.....

.....

.....



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ใบงานที่12

การออกแบบวงจร Shift Register circuit

วัตถุประสงค์

1. เพื่อให้นักศึกษาใช้โปรแกรม Xilinx ISE 7.1i ได้
2. เพื่อให้นักศึกษาออกแบบวงจร Decoder โดยโปรแกรม Xilinx ISE 7.1i ได้
3. เพื่อให้นักศึกษานำวงจรดิจิทัลที่ออกแบบมาสร้างจริงโดยการโปรแกรมอุปกรณ์ CPLD ได้

เครื่องมือสำหรับการทดลอง

1. ระบบไมโครคอมพิวเตอร์
2. โปรแกรม Xilinx ISE 7.1i
3. สายคาร์โนไลต์
4. บอร์ดทดลอง CPLD เบอร์ XC9572

ทฤษฎีพื้นฐาน

วงจรเลื่อนข้อมูล (Shift Register circuit) เป็นวงจรที่ทำหน้าที่เลื่อนข้อมูลจากจุดหนึ่งไปยังอีกจุดหนึ่ง วงจรเลื่อนข้อมูลในใบงานนี้จะมียู 2 ประเภทด้วยกันได้แก่

1. วงจรเลื่อนข้อมูลแบบอนุกรมเข้า-อนุกรมออก (Serial In-Parallel Out: SISO)
2. วงจรเลื่อนข้อมูลแบบอนุกรมเข้า-ขนานออก (Parallel In-Serial Out: SISO)

1. ทำการพิมพ์ชุดคำสั่งของ Shift Register circuit แบบอนุกรมเข้าอนุกรมออกดังรูปที่ ข.118

```

library ieee;
use ieee.std_logic_1164.all;
entity shift is
port(
C, SI, CE : in std_logic;
SO : out std_logic
);
end shift;
architecture arch1 of shift is
signal tmp: std_logic_vector(7 downto 0);
begin
process (C)
begin
if (C'event and C='0') then
if (CE='1') then
for i in 0 to 6 loop
tmp(i+1) <= tmp(i);
end loop;
tmp(0) <= SI;
end if;
end if;
end process;
SO <= tmp(7);
end arch1;

```

รูปที่ ข.118 ชุดคำสั่งของ Shift Register circuit แบบอนุกรมเข้าอนุกรมออก

บันทึกผลการทดลอง

.....

.....

.....

.....

.....

.....

.....

.....

2. ทำการพิมพ์ชุดคำสั่งของ Shift Register circuit แบบอนุกรมเข้า-ขนานออกดังรูป ข.119

```

library ieee;
use ieee.std_logic_1164.all;
entity shift_2 is
port (
C, SI : in std_logic;
PO : out std_logic_vector(7 downto 0)
);
end shift_2;
architecture arch1 of shift_2 is
signal tmp: std_logic_vector(7 downto 0);
begin
process (C)
begin
if (C'event and C='1') then
tmp <= tmp(6 downto 0) & SI;
end if;
end process;
PO <= tmp;
end arch1;

```

รูปที่ ข.119 ชุดคำสั่งของ Shift Register circuit แบบอนุกรมเข้า-ขนานออก

บันทึกผลการทดลอง

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

สรุปผลการทดลอง

.....

.....

.....

.....

.....

.....

.....



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ใบงานที่ 13

การออกแบบวงจร Comparator

วัตถุประสงค์

1. เพื่อให้นักศึกษาใช้โปรแกรม Xilinx ISE 7.1i ได้
2. เพื่อให้นักศึกษาออกแบบวงจร Decoder โดยโปรแกรม Xilinx ISE 7.1i ได้
3. เพื่อให้นักศึกษานำวงจรดิจิทัลที่ออกแบบมาสร้างจริงโดยการโปรแกรมอุปกรณ์ CPLD ได้
4. ออกแบบวงจรเปรียบเทียบข้อมูล (Comparator) ด้วยโปรแกรม Xilinx ISE 7.1i ได้

เครื่องมือสำหรับการทดลอง

1. ระบบไมโครคอมพิวเตอร์
2. โปรแกรม Xilinx ISE 7.1i
3. สายดาร์วินโฮลด
4. บอร์ดทดลอง CPLD เบอร์ XC9572

ทฤษฎีพื้นฐาน

วงจรเปรียบเทียบข้อมูล (Comparator) เป็นวงจรที่ทำหน้าที่นำเอาข้อมูลจากอินพุตทั้ง 2 ชุด ของวงจรมาเปรียบเทียบกันผลลัพธ์จากการเปรียบเทียบอาจจะแสดงอยู่ในรูปแบบของการ เท่ากัน, ไม่เท่ากัน หรือ มากกว่าน้อยกว่า เป็นต้น

จากวงจรด้านบนเป็นตัวอย่างของวงจรเปรียบเทียบข้อมูลแบบ 2 บิตโดยจะเป็นการเปรียบเทียบกันแบบบิตต่อบิต คือบิตนัยสำคัญสูง A1 กับ B1 และบิตล่าง A0 กับ B0 ซึ่งเอาต์พุตของวงจรนี้จะแสดงผลเพียงการเท่ากัน หรือ ไม่เท่ากัน โดยให้ลอจิกเอาต์พุตเป็น "1" เมื่อผลลัพธ์เท่ากัน และ ลอจิก "0" เมื่อผลลัพธ์ไม่เท่ากัน

1. พิมพ์ชุดคำสั่งของวงจร Comparator ดังรูปที่ ข.120

```
10 -----
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 entity Comparator2 is
31 port (a,b,c,d : in std_logic := 'U';
32       q : out std_logic);
33 end Comparator2;
34
35 architecture Behavioral of Comparator2 is
36
37 begin
38   q <= not(a xor b) and not(c xor d);
39
40 end Behavioral;
41
```

รูปที่ ข.120 วงจร Comparator

บันทึกผลการทดลอง

ตารางที่ ข.27 บันทึกผลการทดลองของวงจร Comparator

A0	B0	A1	B1	Q
0	0	0	0	
0	0	0	1	
0	0	1	1	
0	1	0	1	

สรุปผลการทดลอง

.....

.....

.....

.....

.....

.....

.....

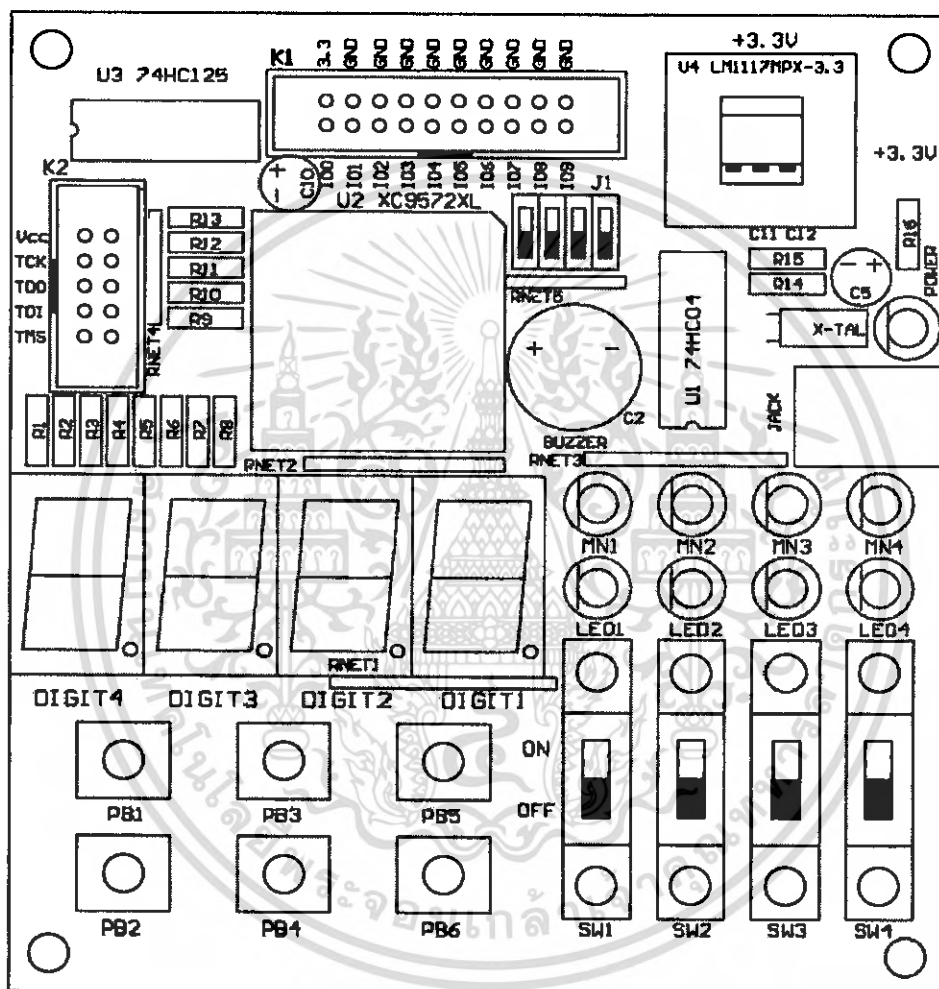




ภาคผนวก ค

คู่มือการใช้งาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



คณะครุศาสตร์อุตสาหกรรม

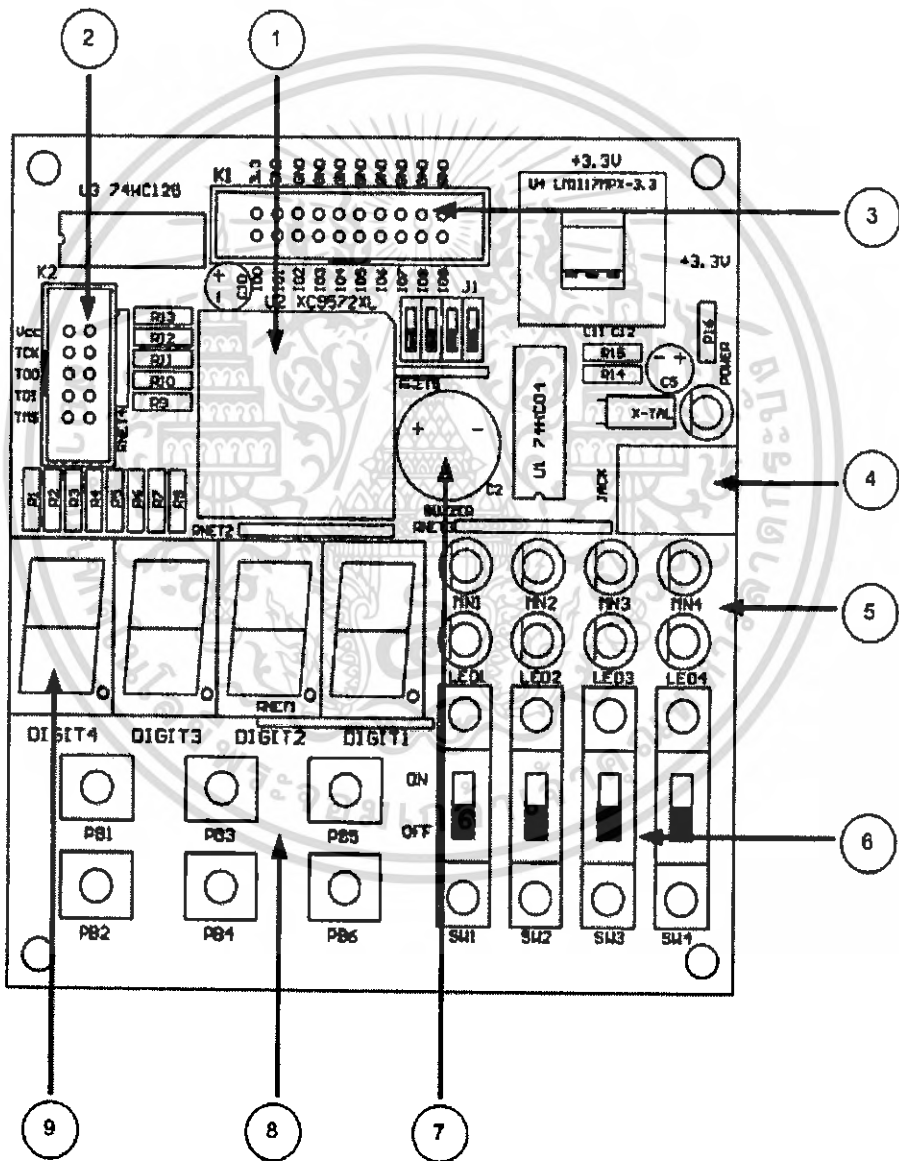
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษานานาชาติ ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ปีการศึกษา 2549
 ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1. คำแนะนำเบื้องต้น

ก่อนที่จะลงใช้งานชุดฝึกดิจิทัลด้วย CPLDs ด้วยตนเองนั้น ควรทำการศึกษาการใช้งานจากคู่มือให้เข้าใจ เพื่อการใช้งานที่ถูกต้อง และเป็นเป็นการป้องกันความเสียหายที่อาจเกิดขึ้นกับชุดฝึกดิจิทัลด้วย CPLDs

2. ส่วนประกอบของบอร์ดทดลองรุ่น CPLD Explorer XC9572XL



รูปที่ ค.1 ส่วนประกอบของบอร์ดทดลองรุ่น CPLD Explorer XC9572XL

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากรูปที่ ค.1 มีรายละเอียดต่างๆ ดังนี้

- 1 ไอซีเบอร์ XC9572XL
- 2 คอนเนคเตอร์เชื่อมต่อกับเครื่องคอมพิวเตอร์
- 3 จุดเชื่อมต่อเพื่อนำไปใช้ประยุกต์กับวงจรภายนอก
- 4 อะแดปเตอร์ 9VDC
- 5 LED Logic Monitor
- 6 Slide Switch
- 7 Buzzer
- 8 Botton Switch
- 9 7-Segment 4 หลัก

3. การติดตั้งและใช้งาน

- 1.1 ทำการเสียบปลั๊กเข้าที่อะแดปเตอร์ของบอร์ดทดลองรุ่น CPLD Explorer XC9572XL
- 1.2 ทำการต่อสายดาวินโฮลด์เข้าที่ตัวบอร์ดและเครื่องคอมพิวเตอร์
- 1.3 ทำการเปิดโปรแกรม Xilinx ISE 7.1i ที่หน้าเดสก์ทอปดังรูปที่ ค.2



รูปที่ ค.2 ไอคอน Xilinx ISE 7.1i

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.4 ให้ทำการทดสอบการเชื่อมต่อโดยการพิมพ์ชุดคำสั่ง แอนเกต ดังรูปที่ ค.3

```
entity AndGate is
    Port ( A : in std_logic;
          B : in std_logic;
          C : out std_logic);
end AndGate;

architecture Behavioral of AndGate is
begin
    C <= A and B;
end Behavioral;
```

รูปที่ ค.3 ชุดคำสั่งในการทดสอบการเชื่อมต่อ

1.5 ให้ทำการบันทึกแล้วทำการ Synthesize - XST เพื่อเช็คข้อผิดพลาดของโปรแกรม

1.6 ให้ทำการกำหนดขาอินพุตและเอาต์พุตให้ตรงตามดังตารางที่ ค.1

ตารางที่ ค.1 แสดงตำแหน่งขาของชิปที่ต่ออยู่กับฮาร์ดแวร์ภายนอกที่อยู่บนบอร์ด

LED		7-Segment	
I/O Name	Pin No	I/O Name	Pin No.
L1	P38	A	P27
L2	P37	B	P26
L3	P36	C	P25
L4	P35	D	P24
Push Button Sw		E	P22
I/O Name	Pin No.	F	P20
PB1	P39	G	P18
PB2	P40	Dot	P19
PB3	P42	DIGIT4(SEG1)	P28
PB4	P43	DIGIT3(SEG2)	P29
PB5	P44	DIGIT2(SEG3)	P33
PB6	P1	DIGIT1(SEG4)	P34

ตารางที่ ค.1 (ต่อ) แสดงตำแหน่งขาของชิพที่ต่ออยู่กับฮาร์ดแวร์ภายนอกที่อยู่บนบอร์ด

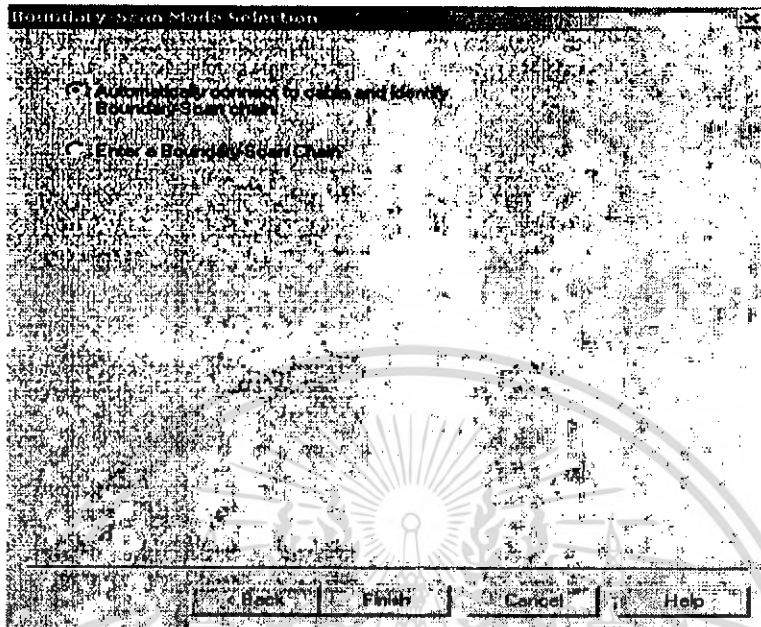
SLIDE SW		I/O ของ K1		REMARK
I/O Name	Pin No.	I/O Name	Pin No.	
SLIDE SW	P42	IO0	P14	
SLIDE SW	P43	IO1	P13	
SLIDE SW	P44	IO2	P12	
SLIDE SW	P1	IO3	P11	
Misc		IO4	P9	
I/O Name	Pin No.	IO5	P8	
Buzzer	P2	IO6	P7,MN1	J1-1
		IO7	P6,MN2	J2-2
OSC = 32.768 kHz	P5	IO8	P4,MN3	J3-3
		IO9	P3,MN4	J4-4

1.7 เมื่อทำการกำหนดขาอินพุตและเอาต์พุตแล้วหลังจากนั้นให้ทำการ Configure Device [iMPACT] ดังรูปที่ ค.4 เพื่อทำการโปรแกรมลงตัวไอซี

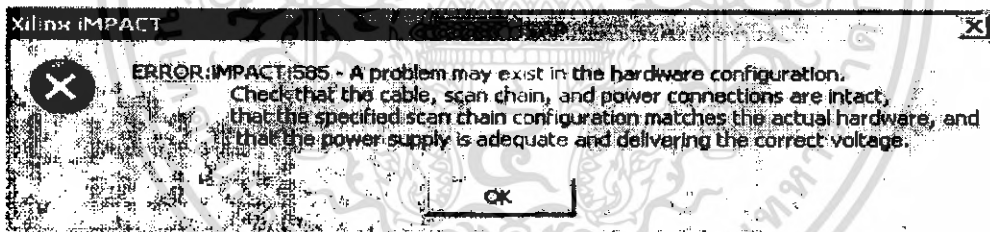


รูปที่ จ.4 Configure Device [iMPACT]

1.8 จากรูปที่ ค.5 ให้ทำการเลือก Automatically connect to cable and identify Boundary-Scan chain แล้วทำคลิก Finish ส่วนรูปที่ ค.6 กรณีที่ไม่มีการเชื่อมต่อสายระหว่างบอร์ดทดลองกับเครื่องคอมพิวเตอร์จะปรากฏข้อความผิดพลาดเพื่อแจ้งให้ทราบว่าไม่ได้มีการเชื่อมต่อสายให้ทำการตรวจสอบว่าได้ทำการเชื่อมต่อสายหรือไม่เมื่อทำการแก้ไขแล้วให้ทำตามขั้นตอนต่อไป

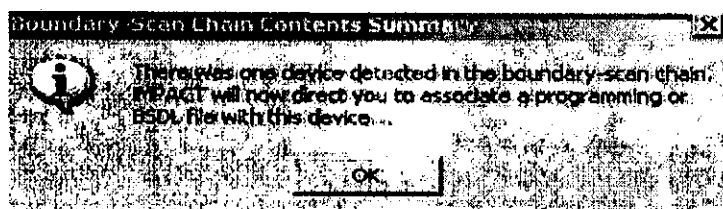


รูปที่ ค.5 เลือก Automatically connect to cable and identify Boundary-Scan chain



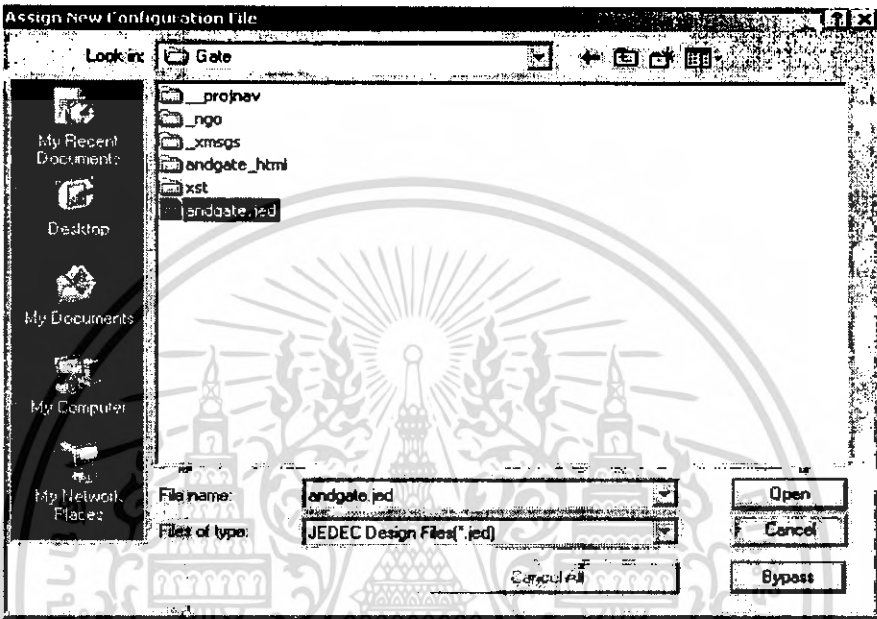
รูปที่ ค.6 เมื่อไม่มีการเชื่อมต่อสายระหว่างบอร์ดทดลองกับคอมพิวเตอร์จะปรากฏข้อความผิดพลาด

1.9 เมื่อทำการตรวจสอบสายเชื่อมต่อถูกต้องแล้วจะปรากฏข้อความดังรูปที่ จ.7



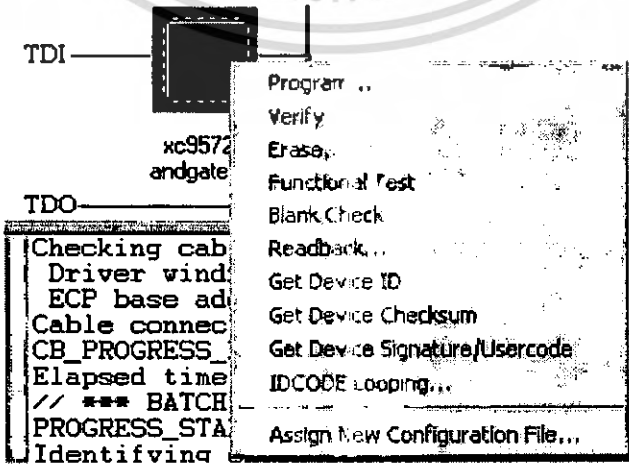
เอกสารนี้เป็นเอกสารที่สงวนไว้รูปที่ ค.7 เมื่อไม่พบข้อผิดพลาดในการเชื่อมต่อสายสัญญาณไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.10 ให้ทำการเลือกโปรแกรมที่ได้ทำการออกแบบดังรูปที่ ค.8 แล้วทำการคลิก Open



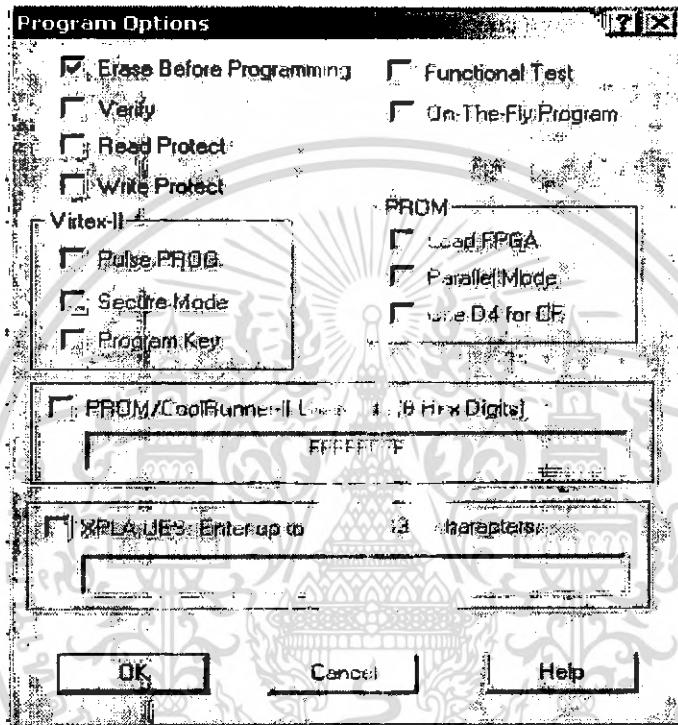
รูปที่ ค.8 เลือกโปรแกรมที่ได้ทำการออกแบบ

1.11 ให้ทำการคลิกขวาที่ตัวไอซีแล้วเลือก Program... ดังรูปที่ ค.9



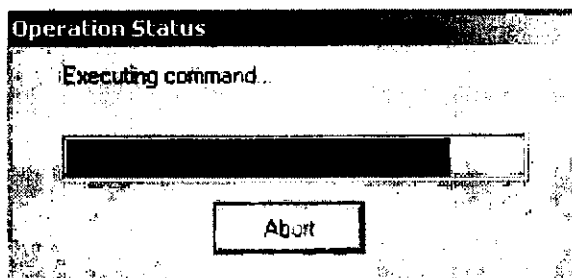
รูปที่ ค.9 การโปรแกรมลงตัวไอซี

1.12 เมื่อทำการคลิกที่ Program... แล้วจะปรากฏหน้าต่าง Program Options ให้ทำการคลิกให้ เกิดเครื่องหมายถูกที่หน้า Erase Before Programming แล้วคลิก OK ดังรูปที่ ค.10



ดังรูปที่ ค.10 Program Options

1.13 เมื่อทำการคลิก OK แล้วโปรแกรมจะทำการโหลดชุดคำสั่งลงตัวไอซีดังรูปที่ ค.11 ส่วนรูปที่ ค.12 จะปรากฏข้อความเมื่อทำการโปรแกรมลงตัวไอซีเสร็จเรียบร้อยแล้ว



รูปที่ ค.11 แสดงการโหลดชุดคำสั่งลงตัวไอซี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับใช้ภายในเพื่อการศึกษาเท่านั้น เมื่อผู้ใดได้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Programming Succeeded

รูปที่ ค.12 ข้อความเมื่อทำการโปรแกรมลงตัวไอซีเสร็จเรียบร้อยแล้ว

1.14 จากขั้นตอนทั้งหมดเมื่อมาปรากฏข้อผิดพลาดอันใดหมายถึงบอร์ดสามารถทำงานได้ตามปกติ

4. การแก้ปัญหาเบื้องต้น

เมื่อท่านประสบปัญหาในการใช้บอร์ดทดลองรุ่น CPLD Explorer XC9572XL สามารถตรวจสอบแนวทางแก้ไขปัญหเบื้องต้นได้จากข้างล่างนี้

ตารางที่ ค.2 การแก้ปัญหาเบื้องต้น

อาการ	สาเหตุและ/หรือวิธีการแก้ไข
ไม่มีไฟเข้าสู่บอร์ดทดลอง	ตรวจสอบที่อะแดปเตอร์ว่าทลวมหรือไม่
ไม่สามารถโหลดโปรแกรมลงตัวไอซีได้	ตรวจสอบสายเชื่อมต่อว่าได้มีการเชื่อมต่อหรือไม่
เมื่อโหลดโปรแกรมแล้วไม่สามารถใช้งานได้	ตรวจสอบไอซีเบอร์ XC9572XL อาจจะเสียหายได้

5. การดูแลรักษาและข้อควรระวัง

5.1 การดูแลรักษา

- 1. ตรวจสอบขั้วต่อสายเชื่อมต่อให้อยู่ในสภาพพร้อมใช้งาน
- 2. ควรมีการตรวจสอบหรือทดลองการทำงานก่อนทุกครั้งเมื่อมีการใช้งาน

5.2 ข้อควรระวัง

เอกสารนี้เป็น 1. ทรัพย์สินทางปัญญาของบริษัทผู้จัดทำเอกสารฉบับนี้ อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- 2. ควรวางตัวเครื่องใกล้กับเครื่องคอมพิวเตอร์เพื่อสะดวกในการใช้งาน
- 3. การเคลื่อนย้ายควรระมัดระวังอย่าให้มีการกระแทกเพื่อป้องกันความเสียหายของบอร์ดทดลอง

6. ข้อมูลจำเพาะ

ตารางที่ ค.3 ข้อมูลจำเพาะ

คุณสมบัติ	รายละเอียด
การเชื่อมต่อคอมพิวเตอร์	เชื่อมต่อคอมพิวเตอร์ผ่านพอร์ตขนาน
แหล่งจ่ายพลังงาน	6VDC-12VDC
ตัวแสดงผลแบบ 7-Segment	คอมมอนแคโทด

ภาคผนวก ง

แสดงรายละเอียดและคุณสมบัติของอุปกรณ์สำคัญที่ใช้ในโครงการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



Spartan-IIE 1.8V FPGA Family: Introduction and Ordering Information

DS077-1 (v1.0) November 15, 2001

Preliminary Product Specification

Introduction

The Spartan™-IIE 1.8V Field-Programmable Gate Array family gives users high performance, abundant logic resources, and a rich feature set, all at an exceptionally low price. The five-member family offers densities ranging from 50,000 to 300,000 system gates, as shown in Table 1. System performance is supported beyond 200 MHz.

Spartan-IIE devices deliver more gates, I/Os, and features per dollar than other FPGAs by combining advanced process technology with a streamlined architecture based on the proven Virtex™-E platform. Features include block RAM (to 64K bits), distributed RAM (to 98,304 bits), 19 selectable I/O standards, and four DLLs (Delay-Locked Loops). Fast, predictable interconnect means that successive design iterations continue to meet timing requirements.

The Spartan-IIE family is a superior alternative to mask-programmed ASICs. The FPGA avoids the initial cost, lengthy development cycles, and inherent risk of conventional ASICs. Also, FPGA programmability permits design upgrades in the field with no hardware replacement necessary (impossible with ASICs).

Features

- Second generation ASIC replacement technology
 - Densities as high as 6,912 logic cells with up to 300,000 system gates
 - Streamlined features based on Virtex-E architecture
 - Unlimited in-system reprogrammability
 - Very low cost
- System level features
 - SelectRAM+™ hierarchical memory:
 - 16 bits/LUT distributed RAM
 - Configurable 4K-bit true dual-port block RAM
 - Fast interfaces to external RAM
 - Fully 3.3V PCI compliant to 64 bits at 66 MHz and CardBus compliant
 - Low-power segmented routing architecture
 - Full readback ability for verification/observability
 - Dedicated carry logic for high-speed arithmetic
 - Efficient multiplier support
 - Cascade chain for wide-input functions
 - Abundant registers/latches with enable, set, reset
 - Four dedicated DLLs for advanced clock control
 - Four primary low-skew global clock distribution nets
 - IEEE 1149.1 compatible boundary scan logic
- Versatile I/O and packaging
 - Low cost packages available in all densities
 - Family footprint compatibility in common packages
 - 19 high-performance interface standards, including LVDS and LVPECL
 - Up to 120 differential I/O pairs that can be input, output, or bidirectional
 - Zero hold time simplifies system timing
- Fully supported by powerful Xilinx ISE development system
 - Fully automatic mapping, placement, and routing
 - Integrated with design entry and verification tools

Table 1: Spartan-IIE FPGA Family Members

Device	Logic Cells	Typical System Gate Range (Logic and RAM)	CLB Array (R x C)	Total CLBs	Maximum Available User I/O	Maximum Differential I/O Pairs	Distributed RAM Bits	Block RAM Bits
XC2S50E	1,728	23,000 - 50,000	16 x 24	384	182	84	24,576	32K
XC2S100E	2,700	37,000 - 100,000	20 x 30	600	202	86	38,400	40K
XC2S150E	3,888	52,000 - 150,000	24 x 36	864	263	114	55,296	48K
XC2S200E	5,292	71,000 - 200,000	28 x 42	1,176	289	120	75,264	56K
XC2S300E	6,912	93,000 - 300,000	32 x 48	1,536	329	120	98,304	64K

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้คัดลอกเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

General Overview

The Spartan-IIE family of FPGAs have a regular, flexible, programmable architecture of Configurable Logic Blocks (CLBs), surrounded by a perimeter of programmable Input/Output Blocks (IOBs). There are four Delay-Locked Loops (DLLs), one at each corner of the die. Two columns of block RAM lie on opposite sides of the die, between the CLBs and the IOB columns. These functional elements are interconnected by a powerful hierarchy of versatile routing channels (see Figure 1).

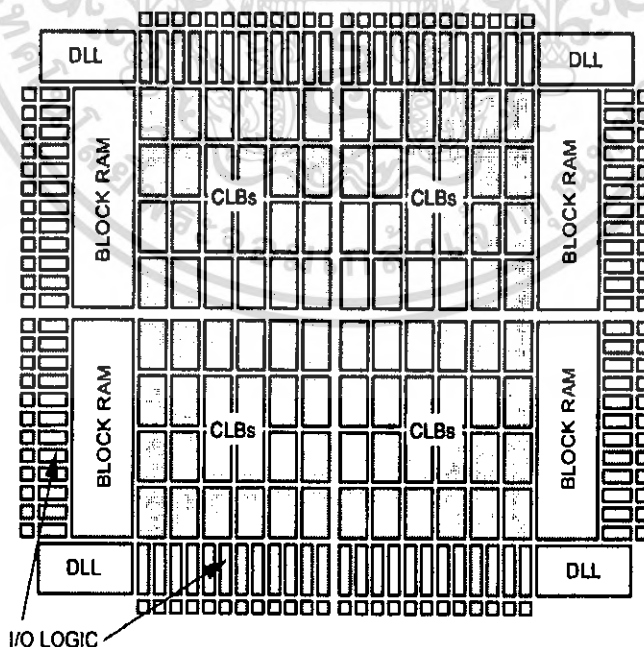
Spartan-IIE FPGAs are customized by loading configuration data into internal static memory cells. Unlimited reprogramming cycles are possible with this approach. Stored values in these cells determine logic functions and interconnections implemented in the FPGA. Configuration data can be read from an external serial PROM (master serial mode), or written into the FPGA in slave serial, slave parallel, or Boundary Scan modes. The Xilinx XC17S00A PROM family is recommended for serial configuration of Spartan-IIE FPGAs. The XC18V00 reprogrammable PROM family is recommended for parallel or serial configuration.

Spartan-IIE FPGAs are typically used in high-volume applications where the versatility of a fast programmable solution adds benefits. Spartan-IIE FPGAs are ideal for shortening product development cycles while offering a cost-effective solution for high volume production.

Spartan-IIE FPGAs achieve high-performance, low-cost operation through advanced architecture and semiconductor technology. Spartan-IIE devices provide system clock rates beyond 200 MHz. Spartan-IIE FPGAs offer the most cost-effective solution while maintaining leading edge performance. In addition to the conventional benefits of high-volume programmable logic solutions, Spartan-IIE FPGAs also offer on-chip synchronous single-port and dual-port RAM (block and distributed form), DLL clock drivers, programmable set and reset on all flip-flops, fast carry logic, and many other features.

Spartan-IIE Family Compared to Spartan-II Family

- Higher density and more I/O
- Higher performance
- Unique pinouts in cost-effective packages
- Differential signaling
 - LVDS, Bus LVDS, LVPECL
- $V_{CCINT} = 1.8V$
 - Lower power
 - 5V tolerance with 100 Ω external resistor
 - 3V tolerance directly
- PCI, LVTTTL, and LVCMOS2 input buffers powered by V_{CCO} instead of V_{CCINT}
- Unique larger bitstream



03677-01 162201

Figure 1: Basic Spartan-IIE Family FPGA Block Diagram



Spartan-IIe Product Availability

Table 2 shows the package and speed grades available for user I/Os available on the device and the number of user I/Os available for each device/package combination.

Table 2: Spartan-IIe Package and Speed Grade Availability

Device	Pins	144	208	256	456
	Type	Plastic TQFP	Plastic PQFP	Fine Pitch BGA	Fine Pitch BGA
	Code	TQ144	PQ208	FT256	FG456
XC2S50E	-6	C, I	C, I	C, I	-
	-7	(C)	(C)	(C)	-
XC2S100E	-6	C, I	C, I	C, I	C, I
	-7	(C)	(C)	(C)	(C)
XC2S150E	-6	-	(C, I)	(C, I)	(C, I)
	-7	-	(C)	(C)	(C)
XC2S200E	-6	-	C, I	C, I	C, I
	-7	-	(C)	(C)	(C)
XC2S300E	-6	-	C, I	C, I	C, I
	-7	-	(C)	(C)	(C)

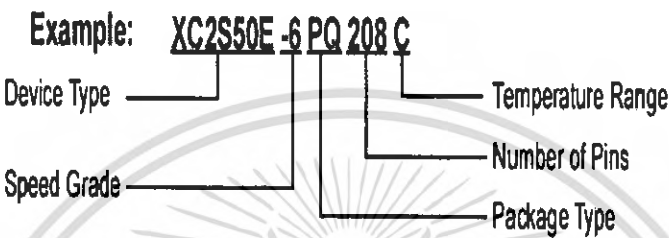
- Notes:
- 1. C = Commercial, T_J = 0° to +85°C; I = Industrial, T_J = -40°C to +100°C
 - 2. Parentheses indicate product not yet released. Contact sales for availability.

Table 3: Spartan-IIe User I/O Chart

Device	Maximum User I/O	Available User I/O According to Package Type			
		TQ144	PQ208	FT256	FG456
XC2S50E	182	102	146	182	-
XC2S100E	202	102	146	182	202
XC2S150E	263	-	146	182	263
XC2S200E	289	-	146	182	289
XC2S300E	329	-	146	182	329



Ordering Information



Device Ordering Options

Device	Speed Grade		Package Type / Number of Pins		Temperature Range (T _J)	
XC2S50E	-6	Standard Performance	TQ144	144-pin Plastic Thin QFP	C = Commercial	0°C to +85°C
XC2S100E	-7	Higher Performance	PQ208	208-pin Plastic QFP	I = Industrial	-40°C to +100°C
XC2S150E			FT256	256-ball Fine Pitch BGA		
XC2S200E			FG456	456-ball Fine Pitch BGA		
XC2S300E						

Revision History

Version No.	Date	Description
1.0	11/15/01	Initial Xilinx release.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



XC9572 In-System Programmable CPLD

December 4, 1998 (Version 3.0)

Product Specification

Features

- 7.5 ns pin-to-pin logic delays on all pins
- f_{CNT} to 125 MHz
- 72 macrocells with 1,600 usable gates
- Up to 72 user I/O pins
- 5 V in-system programmable (ISP)
 - Endurance of 10,000 program/erase cycles
 - Program/erase over full commercial voltage and temperature range
- Enhanced pin-locking architecture
- Flexible 36V18 Function Block
 - 90 product terms drive any or all of 18 macrocells within Function Block
 - Global and product term clocks, output enables, set and reset signals
- Extensive IEEE Std 1149.1 boundary-scan (JTAG) support
- Programmable power reduction mode in each macrocell
- Stew rate control on individual outputs
- User programmable ground pin capability
- Extended pattern security features for design protection
- High-drive 24 mA outputs
- 3.3 V or 5 V I/O capability
- Advanced CMOS 5V FastFLASH technology
- Supports parallel programming of more than one XC9500 concurrently
- Available in 44-pin PLCC, 84-pin PLCC, 100-pin PQFP and 100-pin TQFP packages

Description

The XC9572 is a high-performance CPLD providing advanced in-system programming and test capabilities for general purpose logic integration. It is comprised of four 36V18 Function Blocks, providing 1,600 usable gates with propagation delays of 7.5 ns. See Figure 2 for the architecture overview.

Power Management

Power dissipation can be reduced in the XC9572 by configuring macrocells to standard or low-power modes of operation. Unused macrocells are turned off to minimize power dissipation.

Operating current for each design can be approximated for specific operating conditions using the following equation:

$$I_{CC} \text{ (mA)} = MC_{HP} (1.7) + MC_{LP} (0.9) + MC (0.006 \text{ mA/MHz}) f$$

Where:

MC_{HP} = Macrocells in high-performance mode

MC_{LP} = Macrocells in low-power mode

MC = Total number of macrocells used

f = Clock frequency (MHz)

Figure 1 shows a typical calculation for the XC9572 device.

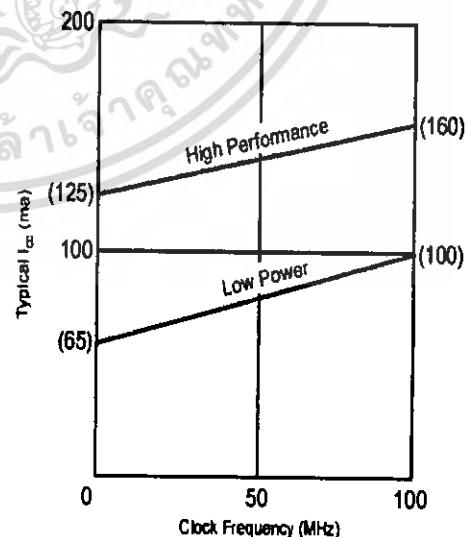
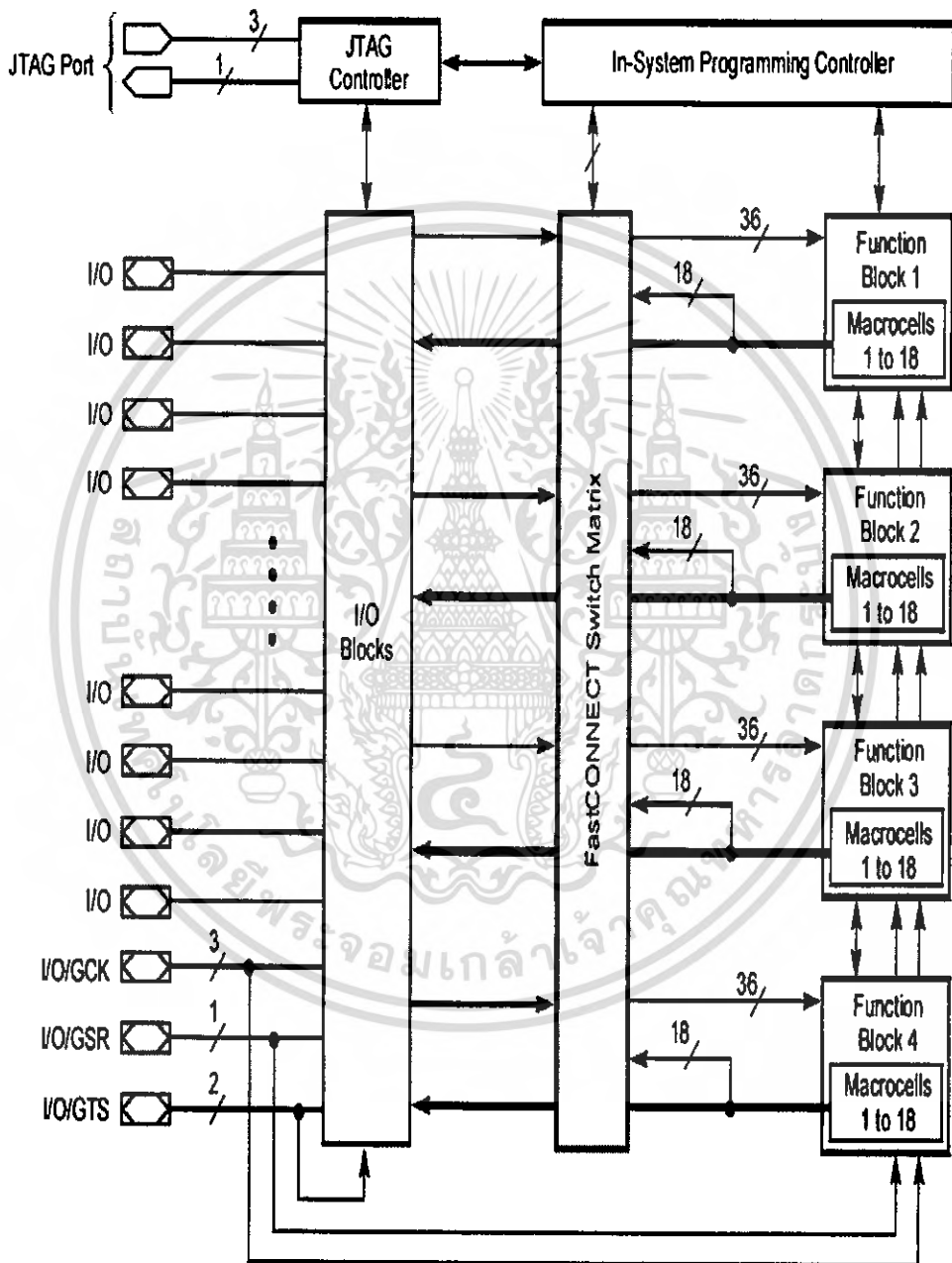


Figure 1: Typical I_{CC} vs. Frequency for XC9572

XC9572 In-System Programmable CPLD



X5921

Figure 2: XC9572 Architecture

Note: Function Block outputs (indicated by the bold line) drive the I/O Blocks directly

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

XC9572 In-System Programmable CPLD

Absolute Maximum Ratings

Symbol	Parameter	Value	Units
V _{CC}	Supply voltage relative to GND	-0.5 to 7.0	V
V _{IN}	DC input voltage relative to GND	-0.5 to V _{CC} + 0.5	V
V _{TS}	Voltage applied to 3-state output with respect to GND	-0.5 to V _{CC} + 0.5	V
T _{STG}	Storage temperature	-65 to +150	°C
T _{SOL}	Max soldering temperature (10 s @ 1/16 in = 1.5 mm)	+260	°C

Warning: Stresses beyond those listed under Absolute Maximum Ratings may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those listed under Recommended Operating Conditions is not implied. Exposure to Absolute Maximum Rating conditions for extended periods of time may affect device reliability.

Recommended Operation Conditions

Symbol	Parameter	Min	Max	Units
V _{CCINT}	Supply voltage for internal logic and input buffer	4.75 (4.5)	5.25 (5.5)	V
V _{CCIO}	Supply voltage for output drivers for 5 V operation	4.75 (4.5)	5.25 (5.5)	V
	Supply voltage for output drivers for 3.3 V operation	3.0	3.6	V
V _{IL}	Low-level input voltage	0	0.80	V
V _{IH}	High-level input voltage	2.0	V _{CCINT} + 0.5	V
V _O	Output voltage	0	V _{CCIO}	V

Note: 1. Numbers in parenthesis are for industrial temperature range versions.

Endurance Characteristics

Symbol	Parameter	Min	Max	Units
t _{DR}	Data Retention	20	-	Years
N _{PE}	Program/Erase Cycles	10,000	-	Cycles

XC9572 In-System Programmable CPLD

DC Characteristics Over Recommended Operating Conditions

Symbol	Parameter	Test Conditions	Min	Max	Units
VOH	Output high voltage for 5 V operation	IOH = -4.0 mA VCC = Min	2.4		V
	Output high voltage for 3.3 V operation	IOH = -3.2 mA VCC = Min	2.4		V
VOL	Output low voltage for 5 V operation	IOL = 24 mA VCC = Min		0.5	V
	Output low voltage for 3.3 V operation	IOL = 10 mA VCC = Min		0.4	V
IIL	Input leakage current	VCC = Max VIN = GND or VCC		±10.0	µA
IIH	I/O high-Z leakage current	VCC = Max VIN = GND or VCC		±10.0	µA
CIN	I/O capacitance	VIN = GND f = 1.0 MHz		10.0	pF
ICC	Operating Supply Current (low power mode, active)	VI = GND, No load f = 1.0 MHz	65 (Typ)		ma

AC Characteristics

Symbol	Parameter	XC9572-7		XC9572-10		XC9572-15		Units
		Min	Max	Min	Max	Min	Max	
tPD	I/O to output valid		7.5		10.0		15.0	ns
tSU	I/O setup time before GCK	4.5		6.0		8.0		ns
tH	I/O hold time after GCK	0.0		0.0		0.0		ns
tCO	GCK to output valid		4.5		6.0		8.0	ns
fCNT ¹	16-bit counter frequency	125.0		111.1		95.2		MHz
fSYSTEM ²	Multiple FB internal operating frequency	83.3		66.7		55.6		MHz
tPSU	I/O setup time before p-term clock input	0.5		2.0		4.0		ns
tPH	I/O hold time after p-term clock input	4.0		4.0		4.0		ns
tPCO	P-term clock to output valid		8.5		10.0		12.0	ns
tOE	GTS to output valid		5.5		6.0		11.0	ns
tOD	GTS to output disable		5.5		6.0		11.0	ns
tPOE	Product term OE to output enabled		9.5		10.0		14.0	ns
tPOD	Product term OE to output disabled		9.5		10.0		14.0	ns
tWLH	GCK pulse width (High or Low)	4.0		4.5		5.5		ns

Note: 1. fCNT is the fastest 16-bit counter frequency available, using the local feedback when applicable.
fCNT is also the Export Control Maximum flip-flop toggle rate, fTOG.
2. fSYSTEM is the internal operating frequency for general purpose system designs spanning multiple FBs.

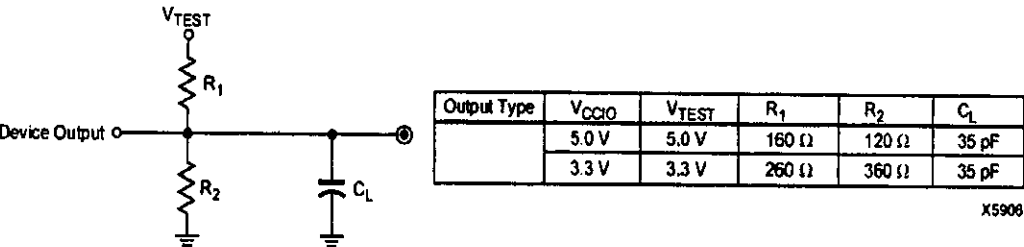


Figure 3: AC Load Circuit

XC9572 In-System Programmable CPLD

Internal Timing Parameters

Symbol	Parameter	XC9572-7		XC9572-10		XC9572-15		Units
		Min	Max	Min	Max	Min	Max	
Buffer Delays								
t _{IN}	Input buffer delay		2.5		3.5		4.5	ns
t _{GCK}	GCK buffer delay		1.5		2.5		3.0	ns
t _{GSR}	GSR buffer delay		4.5		6.0		7.5	ns
t _{GTS}	GTS buffer delay		5.5		6.0		11.0	ns
t _{OUT}	Output buffer delay		2.5		3.0		4.5	ns
t _{EN}	Output buffer enable/disable delay		0.0		0.0		0.0	ns
Product Term Control Delays								
t _{PTCK}	Product term clock delay		3.0		3.0		2.5	ns
t _{PTSR}	Product term set/reset delay		2.0		2.5		3.0	ns
t _{PTTS}	Product term 3-state delay		4.5		3.5		5.0	ns
Internal Register and Combinatorial delays								
t _{PDI}	Combinatorial logic propagation delay		0.5		1.0		3.0	ns
t _{SUI}	Register setup time	1.5		2.5		3.5		ns
t _{HI}	Register hold time	3.0		3.5		4.5		ns
t _{COI}	Register clock to output valid time		0.5		0.5		0.5	ns
t _{AOI}	Register async. S/R to output delay		6.5		7.0		8.0	ns
t _{RAI}	Register async. S/R recovery before clock	7.5		10.0		10.0		ns
t _{LOGI}	Internal logic delay		2.0		2.5		3.0	ns
t _{LOGILP}	Internal low power logic delay		10.0		11.0		11.5	ns
Feedback Delays								
t _F	FastCONNECT matrix feedback delay		8.0		9.5		11.0	ns
t _{LF}	Function Block local feedback delay		4.0		3.5		3.5	ns
Time Adders								
t _{PTA} ³	Incremental Product Term Allocator delay		1.0		1.0		1.0	ns
t _{SLEW}	Slew-rate limited delay		4.0		4.5		5.0	ns

Note: 3. t_{PTA} is multiplied by the span of the function as defined in the family data sheet.

XC9572 In-System Programmable CPLD

XC9572 I/O Pins

Function Block	Macrocell	PC 44	PC 84	PQ 100	TQ 100	BScan Order	Notes	Function Block	Macrocell	PC 44	PC 84	PQ 100	TQ 100	BScan Order	Notes
1	1	–	4	18	16	213		3	1	–	25	43	41	105	
1	2	1	1	15	13	210		3	2	11	17	34	32	102	
1	3	–	6	20	18	207		3	3	–	31	51	49	99	
1	4	–	7	22	20	204		3	4	–	32	52	50	96	
1	5	2	2	16	14	201		3	5	12	19	37	35	93	
1	6	3	3	17	15	198		3	6	–	34	55	53	90	
1	7	–	11	27	25	195		3	7	–	35	56	54	87	
1	8	4	5	19	17	192		3	8	13	21	39	37	84	
1	9	5	9	24	22	189	[1]	3	9	14	26	44	42	81	
1	10	–	13	30	28	186		3	10	–	40	62	60	78	
1	11	6	10	25	23	183	[1]	3	11	18	33	54	52	75	
1	12	–	18	35	33	180		3	12	–	41	63	61	72	
1	13	–	20	38	36	177		3	13	–	43	65	63	69	
1	14	7	12	29	27	174	[1]	3	14	19	36	57	55	66	
1	15	8	14	31	29	171		3	15	20	37	58	56	63	
1	16	–	23	41	39	168		3	16	–	45	67	65	60	
1	17	9	15	32	30	165		3	17	22	39	60	58	57	
1	18	–	24	42	40	162		3	18	–	–	61	59	54	
2	1	–	63	89	87	159		4	1	–	46	68	66	51	
2	2	35	69	96	94	156		4	2	24	44	66	64	48	
2	3	–	67	93	91	153		4	3	–	51	73	71	45	
2	4	–	68	95	93	150		4	4	–	52	74	72	42	
2	5	36	70	97	95	147		4	5	25	47	69	67	39	
2	6	37	71	98	96	144		4	6	–	54	78	76	36	
2	7	–	76	5	3	141	[2]	4	7	–	55	79	77	33	
2	8	38	72	99	97	138		4	8	26	48	70	68	30	
2	9	39	74	1	99	135	[1]	4	9	27	50	72	70	27	
2	10	–	75	3	1	132		4	10	–	57	83	81	24	
2	11	40	77	6	4	129	[1]	4	11	28	53	76	74	21	
2	12	–	79	8	6	126		4	12	–	58	84	82	18	
2	13	–	80	10	8	123		4	13	–	61	87	85	15	
2	14	42	81	11	9	120	[3]	4	14	29	56	80	78	12	
2	15	43	83	13	11	117		4	15	33	65	91	89	9	
2	16	–	82	12	10	114		4	16	–	62	88	86	6	
2	17	44	84	14	12	111		4	17	34	66	92	90	3	
2	18	–	–	94	92	108		4	18	–	–	81	79	0	

Notes: [1] Global control pin

[2] Global control pin GTS1 for PC84, PQ100, and TQ100

[3] Global control pin GTS1 for PC44

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

XC9572 In-System Programmable CPLD

XC9572 Global, JTAG and Power Pins

Pin Type	PC44	PC84	PQ100	TQ100
I/O/GCK1	5	9	24	22
I/O/GCK2	6	10	25	23
I/O/GCK3	7	12	29	27
I/O/GTS1	42	76	5	3
I/O/GTS2	40	77	6	4
I/O/GSR	39	74	1	99
TCK	17	30	50	48
TDI	15	28	47	45
TDO	30	59	85	83
TMS	16	29	49	47
V _{CCINT} 5 V	21,41	38,73,78	7,59,100	5,57,98
V _{CCIO} 3.3 V/5 V	32	22,64	28,40,53,90	26,38,51,88
GND	10,23,31	8,16,27,42, 49,60	2,23,33,46,64,71, 77,86	100,21,31,44,62,69, 75, 84
No Connects	-	—	4,9,21,26,36,45,48, 75, 82	2,7,19,24,34,43,46, 73, 80

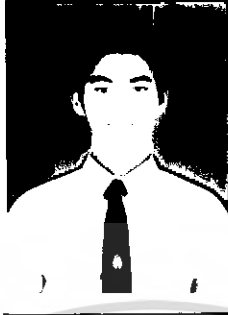
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ประวัติผู้แต่ง



ชื่อ-สกุล	นายสมโภช ธรรมโชติ
วัน เดือน ปีเกิด	6 สิงหาคม 2525
ภูมิลำเนา	68/10 หมู่ 15 ถ.สุขุมวิท 103 แขวงบางนา เขตบางนา จังหวัดกรุงเทพมหานคร โทรศัพท์เคลื่อนที่ 086-003-1626
ประวัติการศึกษา	
ประถมศึกษา	โรงเรียนรุ่งเรืองอุปถัมภ์ จังหวัดกรุงเทพมหานคร
มัธยมศึกษาตอนต้น	โรงเรียนรุ่งเรืองอุปถัมภ์ จังหวัดกรุงเทพมหานคร
ประกาศนียบัตรวิชาชีพ	โรงเรียนเทคโนโลยีกรุงเทพ จังหวัดกรุงเทพมหานคร
ประกาศนียบัตรวิชาชีพชั้นสูง	โรงเรียนเทคโนโลยีกรุงเทพ จังหวัดกรุงเทพมหานคร
ปริญญาตรี	สาขาวิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรม ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม สจล.
คติพจน์	พลังอันยิ่งใหญ่ มาจากความรับผิดชอบอันใหญ่ยิ่ง

ประวัติผู้แต่ง



ชื่อ-สกุล	นายไพรัช ศรีโปฏก
วัน เดือน ปีเกิด	26 มกราคม 2527
ภูมิลำเนา	99/5 หมู่ 2 แขวงลาดกระบัง เขตลาดกระบัง จังหวัดกรุงเทพมหานคร 10520 โทรศัพท์เคลื่อนที่ 086-773-6677
ประวัติการศึกษา	
ประถมศึกษา	โรงเรียนเซนต์จิมคิลส์
มัธยมศึกษาตอนต้น	โรงเรียนพรตพิทยพยัต
ประกาศนียบัตรวิชาชีพ	วิทยาลัยเทคนิคกาญจนาภิเษก มหานคร
ประกาศนียบัตรวิชาชีพชั้นสูง	วิทยาลัยเทคนิคมีนบุรี
ปริญญาตรี	สาขาวิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรม ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม สจล.
คติพจน์	ผิดเป็นครู

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ประวัติผู้แต่ง



ชื่อ-สกุล	นายกฤษณล ไผ่ชู
วัน เดือน ปีเกิด	13 ตุลาคม พ.ศ. 2526
ภูมิลำเนา	201/247 หมู่ 11 ต.บางพลีใหญ่ อ.บางพลี จังหวัดสมุทรปราการ โทรศัพท์เคลื่อนที่ 089-184-6299
ประวัติการศึกษา	
ประถมศึกษา	โรงเรียนสุขเจริญผล
มัธยมศึกษาตอนต้น	โรงเรียนบางพลีราษฎร์บำรุง
ประกาศนียบัตรวิชาชีพ	โรงเรียนเทคโนโลยีกรุงเทพ
ประกาศนียบัตรวิชาชีพชั้นสูง	โรงเรียนเทคโนโลยีกรุงเทพ
ปริญญาตรี	สาขาวิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรม ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม สจล.
คติพจน์	อดีตช่างมันปัจจุบันสำคัญที่สุด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้