สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

เครื่องทดลอง เอฟพีจีเอ ด้วยภาษา วีเอชดีแอล

FPGA EXPERIMENT WITH VHDL

T046456

โดย

นาย ณัฐพล แก่นศิริ

นาย ไพฑูลย์ โตพานิชย์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรอุตสาหกรรมศาสตรบัณฑิต

สาขาเทคโนโลยีโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

ฝน.
ฐ๑ ๓๔๔ ก
ร๖๔๒

เลขหมู่.....
เลขทะเบียน 46456
วัน, เดือน, ปี - 2 เม.ย. 2546

.b.....
.i.....

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หัวข้อปริญญานิพนธ์	เครื่องทดลอง เอฟพีจีเอ ด้วยภาษา วีเอชดีแอล	
TITLE	FPGA EXPERIMENTAL WITH VHDL	
นักศึกษา	นาย ณัฐพล แก่นศิริ	รหัสประจำตัว 43015812
	นาย ไพฑูลย์ โตพานิชย์	รหัสประจำตัว 43015828
อาจารย์ผู้ควบคุมปริญญานิพนธ์	ผศ.ไพศาล สิทธิโยภาสกุล	
	อาจารย์ กฤดากร กล่อมการ	
ภาควิชา	เทคนิคอุตสาหกรรม	
ปีการศึกษา	2544	

ปริญญานิพนธ์ฉบับนี้ได้รับการอนุมัติให้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรอุตสาหกรรมศาสตร์บัณฑิต คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

คณะกรรมการสอบปริญญานิพนธ์

.....ประธานกรรมการ

(.....)

.....กรรมการ

(.....)

.....กรรมการ

(.....)

.....กรรมการ

(.....)

.....กรรมการ

(.....)

ลิขสิทธิ์ของคณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หัวข้อปริญญานิพนธ์	เครื่องทดลอง เอฟพีจีเอ ด้วยภาษา วีเอชดีแอล
นักศึกษา	นายณัฐพล แก่นศิริ รหัสประจำตัว 43015812
	นายไพฑูลย์ โตพานิชย์ รหัสประจำตัว 43015828
อาจารย์ผู้ควบคุมปริญญานิพนธ์	ผศ.ไพศาล สิทธิโยภาสกุล
	อาจารย์ กฤดากร กล่อมการ
ระดับการศึกษา	ปริญญาอุตสาหกรรมศาสตรบัณฑิต
	สาขาวิชาเทคโนโลยีโทรคมนาคม
ภาควิชา	เทคนิคอุตสาหกรรม
ปีการศึกษา	2544

บทคัดย่อ

ปริญญานิพนธ์นี้เป็นการศึกษาการออกแบบวงจรรวมด้วยภาษาวีเอชดีแอล(VHSIC (Very High speed Intergreated Circuit) Hardware Description Language) มากขึ้นแต่ผู้ที่ออกแบบไม่ได้รับความสะดวกในการออกแบบเท่าที่ควรอาจจะไม่มีบอร์ดทดลองให้มาทดสอบโปรแกรมที่ออกแบบมาและเป็นการยากต่อผู้ที่ทำการเริ่มศึกษาการออกแบบวงจรรวมด้วยภาษาวีเอสดีแอล เพราะว่าต้องทำวงจรขึ้นมาทดสอบเองซึ่งต้องเสียเวลามากดังนั้นโครงงานนี้จึงได้ทำการศึกษาและทำชุดทดลองเอฟพีจีเอขึ้นมาโดยมีจุดประสงค์เพื่อให้ผู้ที่ต้องการศึกษา การออกแบบวงจรโดยใช้ภาษาวีเอสดีแอลได้ทดลองออกแบบและสามารถเรียนรู้ได้ด้วยตนเองการออกแบบโครงงานนี้ตัวบอร์ดทดลองจะมีการแบ่งเป็นโมดุลต่างๆ คือ โมดูลเอฟพีจีเอ,แสดงผลแบบ แอลอีดี 7 ส่วน,พอร์ต แสดงผลแอลซีดี,การติดต่อกับคีย์บอร์ด,สวิทช์,การแสดงผลแบบแอลอีดี,พอร์ตอาร์เอส232,มอเตอร์เสต็ปและมอเตอร์ ดีซี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

PROJECT TITLE	**FPGA EXPERIMENTAL WITH VHDL**	
STUDENT	**Mr. Nattapon Kansiri**	**No. 43015812**
	Mr. Paitoon Topanit	**No. 43015828**
ADVISOR	**Ass.Prof. Paisan Sittiyopasakul**	
	Mr. Kitdakorn Kaomkarn	
COURSE	**Bachelor of Industrial Technology Telecommunication**	
DEPARTMENT	**Industrial Technology**	
YEAR	**2001**	

ABSTRACT

This thesis present the development Integrated Circuit(IC) by using VHDL (VHSIC (Very High Speed Integrated Circuit) hardware Description Language) language. But designer and developer rough because they don't have experiment board for test his program. Design and development experiment board for only one purpose is consume more time and money. So the purposes of this project are design and develop FPGA Experimental With VHDL Board for anyone who wants to study and design circuit .In FPGA Experimental With VHDL Board had partition into several modules for comfort and easy to learning. This board include FPGA, 7-Segment Display, port LCD Display, keyboard Switch, LED port, RS-232 port, Stepping Motor port and DC-Motor port

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

โครงงานนี้ได้จัดทำขึ้นเป็นผลสำเร็จ ทางคณะผู้จัดทำต้องขอขอบคุณ บิดา มารดา ที่คอยช่วยเหลือ และให้กำลังใจ ขอขอบพระคุณบูรพาจารย์ทั้งหลายท่านผู้เขียนเอกสารและตำราอ้างอิงต่างๆทุกท่าน โดยเฉพาะอาจารย์ที่ปรึกษา อาจารย์ ผศ.ไพศาล สิทธิโยภาสกุล ที่ให้คำปรึกษาที่มีประโยชน์อย่างมากและคอยกระตุ้นให้พวกเราขยันทำงาน, อาจารย์ กฤดากร กล่อมการ ที่คอยให้คำชี้แนะดีๆ สำหรับชีวิตการเรียนและในอนาคตการทำงาน และพี่ๆที่ NECTEC ฝ่ายออกแบบไอซี ที่กรุณาให้ยืมเครื่องมืออุปกรณ์ และคำแนะนำดีๆในการทำงานและเพื่อนทุกคน ที่คอยให้คำปรึกษา

ทั้งนี้คณะผู้จัดทำต้องขอบพระคุณภาควิชาเทคนิคอุตสาหกรรม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ได้ให้คณะผู้จัดทำได้มีโอกาสเข้ามาศึกษา ณ สถาบันแห่งนี้

สุดท้ายนี้ทางคณะผู้จัดทำขอขอบพระคุณ คณะอาจารย์ทุกท่านที่กรุณาประสิทธิ์ประสาทวิชาให้แก่กลุ่มผู้จัดทำ จนทำให้ปริญญานิพนธ์ฉบับนี้สำเร็จลงได้ด้วยดี

คณะผู้จัดทำ

นายณัฐพล แก่นศิริ รหัสประจำตัว 43015812

นายไพฑูลย์ โตพานิชย์ รหัสประจำตัว 43015828

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

หน้า

บทคัดย่อภาษาไทย

บทคัดย่อภาษาอังกฤษ

กิตติกรรมประกาศ

สารบัญ I

สารบัญรูป V

บทที่ 1. บทนำ 1

1.1 จุดมุ่งหมายและวัตถุประสงค์ของโครงงาน 1

1.2 ขอบเขตโครงงาน 2

1.3 รายละเอียดของโครงงาน 2

บทที่ 2. ภาษา วีเอชดีแอล 3

2.1 วีเอชดีแอล คืออะไร 3

2.2 วีเอชดีแอล แอดวานซ์เตจ 4

2.3 ลักษณะการใช้งานภาษา วีเอชดีแอล 4

2.3.1 Document Language 4

2.3.2 Design Language 5

2.3.3 Verification Language 5

2.3.4 Test Language 5

2.3.5 Synthesis Language 5

2.4 วิธีการออกแบบระบบอิเล็กทรอนิกส์ 5

2.5 การออกแบบแบบบนลงล่าง 6

2.6 พื้นฐานการเขียนภาษา วีเอชดีแอล 7

2.6.1 ทั่วไป 7

2.6.2 การเขียน Comment 8

2.7 โครงสร้างภาษา วีเอชดีแอล 8

2.7.1 การ Declaration of VHDL Objects 8

2.7.2 รูปแบบของ Entity และ Architecture 9

2.7.2.1 Entity 10

2.7.2.2 Achitecture 10

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ(ต่อ)

	หน้า
2.7.3 Entity Port mode	10
2.7.4 Component Declaration	11
2.7.5 Component Instantiation	12
2.7.6 การ Configuration	14
2.7.7 Package	15
2.7.7.1 Package Declaration	16
2.7.7.2 Package Body	16
2.7.8 Library	16
2.8 ชนิดของข้อมูล	17
2.8.1 Constant	17
2.8.2 Signal	18
2.8.3 Variable	18
บทที่ 3 เอฟพีจี และการออกแบบวงจรดิจิตอลเพื่อใช้งานกับเอฟพีจีเอ	19
3.1 ฟิลด์โปรแกรมเมเบิล(Field Progarmmable)	19
3.1.1 พีแอลดี (PLD : Programmable Logic Device)	20
3.1.2 เอฟพีจีเอ (FPGA : Field Programmable Gate Array)	20
3.2 สถาปัตยกรรมภายในของเอฟพีจีเอตระกูล XC4000	21
3.2.1 CLB(Configuration Logic Block)	22
3.2.2 IOB(Input Output Block)	22
3.2.3 Interconnect	23
3.3 คุณสมบัติโดยทั่วไปของ เอฟพีจีเอ ตระกูล XC4000	24
3.3.1 เป็นอุปกรณ์รุ่นที่ 3 ของ เอฟพีจีเอ	24
3.3.2 มีสถาปัยกรรมภายในที่ยืดหยุ่น	24
3.3.3 คุณลักษณะทาง System-Oriented	25
3.4 การโปรแกรมเอฟพีจีเอตระกูล XC4000	25
3.4.1 โหมดการโปรแกรม	25
3.4.1.1 โหมดหลัก (Master mode)	26
3.4.1.2 เพอริเฟอรัลโหมด (Peripheral modes)	27

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ(ต่อ)

หน้า

3.4.1.3 โหมดรองแบบอนุกรม(Slave Serial modes) 27

3.5 การใช้ความสามารถของแรมในเอฟพีจีเอตระกูล XC4000 28

3.6 การออกแบบวงจรดิจิตอลเพื่อใช้งานกับอุปกรณ์ FPGA 28

3.6.1 วิธีการออกแบบวงจรดิจิตอล 28

3.6.2 ขั้นตอนการออกแบบวงจรโดยใช้ภาษา HDL 29

3.6.2.1 การออกแบบวงจรให้อยู่ในรูปของโค้ดภาษา HDL 30

3.6.2.2 การตรวจสอบความถูกต้องของวงจรที่สร้างขึ้นในระดับพฤติกรรม 30

3.6.3 การสังเคราะห์วงจรจากดีไซน์ในภาษา HDL 31

3.6.4 การแปลงดีไซน์ที่สังเคราะห์แล้วเพื่อใช้กับอุปกรณ์ FPGA 31

3.6.5 การทำซิมมูเลชั่นหลังผ่านขั้นตอนการวางปละเชื่อมเส้นทาง 31

3.6.6 สิ่งที่ต้องคำนึงถึงเมื่อใช้ภาษา HDL สำหรับการออกแบบวงจร 31

บทที่ 4 การออกแบบโครงงาน และการใช้ซอฟแวร์โปรแกรมลง เอฟพีจีเอ 33

4.1 การออกแบบโครงงาน 33

4.1.1 โมดูลสายดาวน์โหลด (Download Cable Module) 35

4.1.2 โมดูลบอร์ดดาวน์โหลด (Download Board Module) 36

4.1.3 โมดูลบอร์ดแสดงผล (Output Board Module) 36

4.1.3.1 ชุดแสดงผล 37

4.1.3.2 ชุดควบคุม 41

4.2 การใช้โปรแกรม Exemplar Leonardo Spectrum เพื่อการสังเคราะห์วงจร 43

4.2.1-4.2.15 ขั้นตอนการใช้โปรแกรม 43-58

4.3 การใช้โปรแกรม XILINX FOUNDATION M series เพื่อการ Place & Route วงจรลง FPGA 58

4.3.1-4.3.4 ขั้นตอนการใช้โปรแกรม 58-61

4.4 การ Translate & Map 62

4.5 การ Place & Route 62

4.6 การ Configure 63

4.6.1-4.6.2 ขั้นตอนการใช้โปรแกรม 63-64

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ(ต่อ)

หน้า

4.7 การใช้งาน Parallel Cable และการใช้โปรแกรม Hardware Debugger 65

4.7.1-4.7.2 ขั้นตอนการใช้โปรแกรม 65-68

4.8 การเซ็ต Dipswitch M0 M1 M2 ในการกำหนดการ Download ลง FPGA ใช้กับ XC4000E Series 68

4.9 ตัวอย่างการทดลองแบบพื้นฐาน และผลการทดลอง 71

4.9.1 การทดลอง ออกแบบ Shift Register 71

4.9.2 Code VHDL โดยออกแบบวงจรเป็น Shift Register 71

4.9.3 ผลการทดลอง 73

4.9.3.1 ผลการทดลองออกแบบวงจร Shift Register 73

4.9.3.2 ผลการทดลองออกแบบวงจร Control Stepping Motor 75

4.9.3.3 ผลการทดลองออกแบบวงจร Counter 77

บทที่ 5 บทสรุป 79

5.1 วิธีการดำเนินงาน 79

5.2 ปัญหา และอุปสรรคในการทำงาน 79

5.3 สิ่งที่ต้องแก้ไขปรับปรุง 79

ภาคผนวก 80

บรรณานุกรม 85

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป

รูปที่	หน้า
2.1 การออกแบบระบบอิเล็กทรอนิกส์	5
2.2 การออกแบบบนลงล่าง(Top-Down design)	6
2.3 ตัวอย่างการใช้ Statement	7
2.4 การประกาศ Signal หรือ Variable หลายอัน	7
2.5 การประกาศสัญญาณ	8
2.6 ความสัมพันธ์ของส่วนประกอบต่างๆ	9
2.7 รูปการเขียน Entity	9
2.8 รูปการเขียน Architecture	10
2.9 การประกาศใช้ Port ใน Entity	10
2.10 การประกาศ Component	11
2.11 การประกาศ Component ใน Architecture	12
2.12 การทำ Component Instantiation	12
2.13 การ Map สัญญาณเข้ากับ Port โดยใช้ตำแหน่ง	13
2.14 การ Map สัญญาณเข้ากับ Port โดยใช้ชื่อ	14
2.15 การใช้งาน Configuration	14
2.17 การใช้ Configuration ซ้อน Configuration	15
2.18 การใช้งานคำสั่ง Use	15
2.19 การประกาศ Library และ Package	16
2.20 รูปแบบของ Package declaration	16
2.21 รูปแบบการประกาศ Constant	17
2.22 การประกาศ Signal	18
2.23 รูปแบบการประกาศ Variable	18
3.1 ผังแสดงการแบ่งกลุ่มของวงจรรวม ASIC	19
3.2 แสดงวงจรพื้นฐานของอุปกรณ์พีแอลดีซึ่อยู่ในรูปผลคูณร่วมบวก	20
3.3 แสดงวงจรพื้นฐานภายในXilinx เอฟพีจีเอ	22
3.4 แสดงผังวงจรภายของซีแอลบีของเอฟพีเจเอตระกูล XC4000	22
3.5 แสดงผังวงจรของไอโอบีภายในเอฟพีจีเอตระกูลXC4000	23
3.6 รูปแบบของโหมดต่างๆในการโปรแกรมเอฟพีจีเอตระกูล XC4000	25

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป(ต่อ)

รูปที่	หน้า
3.7 แสดงผังวงจรการเชื่อมต่อเอฟพีจีเอในฟโหมดหลักแบบขนาน	26
3.8 แสดงผังวงจรการเชื่อมต่อในโหมดเพอริเฟอรัล แบบอะซิงโครนัส	27
3.9 แสดงผังวงจรการเชื่อมต่อในโหมดรองแบบอนุกรม	27
4.1 แสดงรูปจำลองของบอร์ดทดลอง	33
4.1 Sythesis Design Flow	33
4.2 แสดง Flow chat การทำงาน	34
4.3 วงจรของโมดูลสายดาวน์โหลด	35
4.4 วงจรของโมดูลบอร์ดดาวน์โหลด	36
4.5 วงจรรวมของบอร์ดแสดงผล	37
4.6 วงจรชุดแสดงผลการควบคุมมอเตอร์	38
4.7 ชุดแสดงผลการควบคุม LED	39
4.8 ชุดแสดงผลการควบคุม LED 7 ส่วน	40
4.9 ชุดสำรองแสดงผลการควบคุม LCD	40
4.10 ชุดควบคุม Scankey	41
4.11 ชุดควบคุม Dipswitch และ Switch กด ปล่อย	41
4.12 ชุดสำรองพอร์ตอนุกรม	42
4.13 Synthesis Design Flow	43
4.14 หน้าต่างของโปรแกรม Exemplar Leonadospectrum 2000.1a	44
4.15 หน้าต่างการเลือกเทคโนโลยีการสังเคราะห์	46
4.16 หน้าต่างการเลือกเบอร์ของเอฟพีจีเอ	47
4.17 หน้าต่างการเลือกไฟล์ในการสังเคราะห์	47
4.18 หน้าต่างการเลือกไฟล์ในการสังเคราะห์	48
4.19 หน้าต่างในการเลือกไฟล์ในการสังเคราะห์	48
4.20 หน้าต่างในการเลือกไฟล์ในการสังเคราะห์	49
4.21 หน้าต่างการเซตความเร็วของวงจรที่ถูกสังเคราะห์	49
4.22 หน้าต่างการเซตเอ้าพุตไฟล์ที่ได้จากการสังเคราะห์	50
4.23 หน้าต่างโปรแกรมที่แสดงค่าต่างๆ ในการสังเคราะห์วงจร	51
4.24 แสดงผลของการสังเคราะห์วงจร	52

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป(ต่อ)

รูปที่	หน้า
4.25 แสดง RTL ของวงจรที่ถูกสังเคราะห์ขึ้น	52
4.26 แสดง Schematic ในระดับTechnolgy cell ของวงจรที่ถูกสังเคราะห์ขึ้น	53
4.27 แสดงหน้าต่างของการ Optimize วงจร	54
4.28 แสดงการเลือกไฟล์เอ้าพุตในการสังเคราะห์เพื่อนไป Place & Route	55
4.29 แสดงหน้าต่างของ Design Manager	58
4.30 แสดงหน้าต่างของ New Proct	59
4.31 แสดงหน้าต่างของ New Version	59
4.32 แสดงหน้าต่างของ New Project Complete	60
4.33 แสดงหน้าต่างของการเลือก Option ในการ Place & Route	60
4.34 แสดงหน้าต่างของการเลือก Option ในการ Place & Route	61
4.35 แสดงหน้าต่างของ Flow Engine ก่อนการ run Flow Engine	61
4.36 แสดงหน้าต่างของ Flow EngineหลังการrunFlow Engine	63
4.37 แสดงหน้าต่างของ Report Browsers	64
4.38 แสดงหน้าต่างของ Xilinx FPGA Editor	64
4.39 ตัวอย่างพื้นฐานบางส่วนของ Layout เมื่อขยายภาพให้ใหญ่ขึ้น	65
4.40 JTAG Cable	65
4.41 โปรแกรม Hardware Debugger	66
4.42 Load Design ที่ต้องการ Configure สู่ FPGA	66
4.43 เลือก Parallel และกำหนด Post ที่ Cable ที่ต่ออยู่	67
4.44 สถานะการติดต่อ cable	67
4.45 แสดงสถานะการ Configure การออกแบบสู่ FPGA	67
4.46 เมื่อ Configure เสร็จสิ้น	68
4.47 โหมดในการเซต	68
4.48 เซ็ตโหมด มาสเตอร์	69
4.49 เซ็ตโหมด สลาฟ	69
4.50 เซ็ตโหมดพาราเรล อัพ	70
4.51 เซ็ตโหมดพาราเรล ดาวน์	70
4.52 Gate Level Schematic ของวงจร Shift Register	73

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป(ต่อ)

รูปที่	หน้า
4.53 แสดงขาของ FPGA ของวงจร Shift Register	74
4.54 Gate Level Schematic ของวงจร Control Stepping Motor	75
4.55 แสดงขาของ FPGA ของวงจร Control Stepping Motor	76
4.56 Gate Level Schematic ของวงจร Counter	77
4.57 แสดงขาของ FPGA ของวงจร Counter	78
4.58 ภาพจริงของโครงงาน	80
4.59 ลาย PCB ของโมดูลสายดาวน์โหลด ด้านบน	80
4.60 ลาย PCB ของโมดูลสายดาวน์โหลด ด้านล่าง	81
4.61 ลาย PCBของโมดูลบอร์ดดาวน์โหลดด้านบน	81
4.62 ลาย PCBของโมดูลบอร์ดดาวน์โหลด ด้านล่าง	82
4.63 ลาย PCBของโมดูลบอร์แสดงผลด้านบน	83
4.64 ลายPCBของโมดูลบอร์แสดงผลด้านล่าง	84

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1

บทนำ

ความเป็นมาและเหตุผลการสร้าง

ในทุกวันนี้การออกแบบวงจรอิเล็กทรอนิกส์ และดิจิตอลนั้นมีความสำคัญมากซึ่งเป็นจุดสำคัญของการเจริญเติบโตของอุตสาหกรรม และเศรษฐกิจของประเทศเรา ดังนั้นเทคโนโลยีในการออกแบบวงจรอิเล็กทรอนิกส์ และดิจิตอลจึงก้าวหน้าไปอย่างรวดเร็ว และสะดวกมากขึ้น และเทคโนโลยีที่น่าสนใจอย่างหนึ่งก็คือ การออกแบบวงจรดิจิตอลด้วย ชิฟ เอฟพีจีเอ (FPGA: Field Programmable Gate Array) โดยใช้ภาษาบรรยายอุปกรณ์ฮาร์ดแวร์(HDL: Hardware Discription Language) ในการโปรแกรมลงตัวชิฟนี้ ซึ่งเป็นเทคโนโลยีที่สะดวกและมีความยืดหยุ่นสูง และผู้ผลิตนิยมใช้งานจริง ดังนั้นกลุ่มผู้จัดทำจึงเล็งเห็นความสำคัญตรงจุดนี้ และเห็นว่าภาควิชาเรายังขาดเครื่องมือที่จะให้นักศึกษาที่สนใจในเรื่องนี้ได้ทดลอง และสามารถศึกษาได้ด้วยตนเอง กลุ่มผู้จัดทำจึงตัดสินใจสร้างเครื่องต้นแบบ เครื่องทดลอง FPGA ด้วยภาษา VHDL(FPGA Experimental With VHDL) ขึ้นมาซึ่งน่าจะเป็นประโยชน์แก่นักศึกษาที่ต้องการศึกษาในเรื่องนี้

1.1 จุดมุ่งหมายและวัตถุประสงค์ของโครงงาน

กลุ่มของเรามีจุดมุ่งหมาย เพื่อสร้างเครื่องต้นแบบ เครื่องทดลอง FPGA ด้วยภาษา VHDL เพื่อให้ภาควิชาของเราได้นำไปใช้ในการเรียนการสอนได้จริง สำหรับนักศึกษาที่สนใจในแขนงวิชานี้ ซึ่งชุดทดลองของเรานี้จะมีชุดแสดงผลพื้นฐาน เช่น LED 7 Segment ,LED ,port LCD,Stepper motor,DC Motor และ Keyboard 4*4 เพื่อให้ง่ายสำหรับการศึกษา ของผู้ที่เริ่มต้นศึกษาและออกแบบในเรื่องนี้โดยโครงงานนี้ได้กำหนดจุดประสงค์คือเพื่อศึกษารูปแบบการเขียนภาษา VHDL ในการออกแบบระบบฮาร์ดแวร์ดิจิตอลซึ่งเริ่มตั้งแต่การออกแบบ แก้ไขตรวจสอบ จำลองการทำงานจนถึงขั้นผลิตวงจรหรือ สังเคราะห์วงจร (Synthesis)เพื่อศึกษาลักษณะการทำงาน การควบคุมฟังก์ชั่นการใช้งานต่างๆของ เอฟพีจีเอเพื่อศึกษาโปรแกรมที่ต้องใช้งานจริงในการออกแบบฮาร์ดแวร์ดิจิตอลคือ โปรแกรม ModleSim ,Leonadospectrum และ Xilinx Foundationเพื่อสร้างเครื่องทดลองมาใช้งานได้จริง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.2 ขอบเขตโครงงาน

โครงงานนี้ แยกออกเป็น 3 ส่วน คือ สร้างสายดาวน์โหลดต้นทุนต่ำ ,บอร์ดดาวน์โหลด และชุดแสดงผลการทดลอง โดยรวมกันแล้วได้เครื่องทดลอง FPGA โดยใช้ภาษาวีเอชดีแอล ในการเขียนโปรแกรมออกแบบวงจรดิจิตอลซึ่งสามารถใช้งานได้จริง

1.3 รายละเอียดของโครงงาน

มีเนื้อหาทั้งหมดรวม 6 บท คือ

บทที่1. บทนำ

บทที่2. ภาษาวีเอชดีแอล

บทที่3. เอฟพีจีเอ และการออกแบบวงจรดิจิตอลเพื่อใช้งานกับเอฟพีจีเอ

บทที่4. การออกแบบโครงงาน และการใช้ซอฟแวร์โปรแกรมลง เอฟพีจีเอ

บทที่5. บทสรุป

ภาคผนวก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่2

ภาษาวีเอชดีแอล

ปัจจุบันนี้ธุรกิจทางด้านเกี่ยวกับไมโครอิเล็กทรอนิกส์จัดเป็นธุรกิจที่ค่อนข้างจะมาแรงมาก เนื่องจากเป็นธุรกิจที่มีผลกำไรค่อนข้างสูง ดังจะเห็นได้จากการแข่งขันกันระหว่างบริษัทผู้ผลิตไมโครโปรเซสเซอร์ เช่น บริษัทอินเทล และ บริษัทเอเอ็มดี เป็นต้น เคยมีความสงสัยหรือไม่ว่าทำไมบริษัทนั้นสามารถผลิตผลิตภัณฑ์แต่ละชิ้นออกสู่ท้องตลาดได้รวดเร็ว บริษัททั้ง สองนี้วิธีการออกแบบอย่างไรที่สามารถทำให้ออกแบบได้รวดเร็ว เนื่องจากไมโครโปรเซสเซอร์แต่ละตัวจะประกอบด้วยทรานซิสเตอร์จำนวน มหาศาลและมีความซับซ้อนของวงจรสูง การที่บริษัททั้งสองนี้สามารถผลิตไมโครโปรเซสเซอร์ออกมาแข่งขันกันได้ในท้องตลาด นี้ก็เนื่องจากว่ามีเครื่องมือหรือซอฟท์แวร์ที่ช่วยในการออกแบบที่ดี ทำให้การออกแบบวงจรอิเล็กทรอนิกส์สามารถทำได้คล้ายกับ การเขียนโปรแกรม ซึ่งจะมีภาษาที่ทำหน้าที่บรรยายพฤติกรรมของวงจร (Hardware Description Language: HDL) ที่ต้องการออกแบบ ดังนั้นจึงทำให้การออกแบบวงจรสามารถทำได้ง่ายและมีความรวดเร็วมาก ภาษา VHDL ก็จัดเป็นภาษาที่ทำหน้าที่ บรรยายพฤติกรรมของวงจรภาษาหนึ่งที่ช่วยให้การออกแบบสามารถทำได้ง่าย รวดเร็วและมีประสิทธิภาพสูง

2.1 วีเอชดีแอล คืออะไร

ภาษา วีเอชดีแอล (VHDL)หรือ VHSIC (Language Very High Speed Integrated Circuit) Hardware Description ซึ่งเป็นภาษาที่พัฒนาขึ้นโดยกระทรวงกลาโหมของสหรัฐอเมริกาในช่วงปี ค.ศ. 1980 โดยเป้าหมายของโครงการนี้ก็เพื่อพัฒนา ขีดความสามารถในการออกแบบวงจรรวมให้สูงขึ้นและสามารถทำได้ง่ายมากยิ่งขึ้น เป้าหมายของหลักของพัฒนาภาษา VHDL มี 2 ประการคือ

1. เนื่องจากนักออกแบบวงจรรวมมีความต้องการภาษาที่สามารถรองรับการออกแบบวงจรที่มีความซับซ้อน
2. นักออกแบบต้องการภาษาที่เป็นมาตรฐานหรือเป็นภาษากลางที่ทำให้สามารถเผยแพร่ผลงานการออกแบบกันภายใน กลุ่มนักออกแบบด้วยกันได้

ในปี 1986 ภาษา VHDL ได้เริ่มมีการปรับปรุงภาษา เพื่อให้สามารถกำหนดเป็นมาตรฐานของ IEEE โดยสามารถประกาศเป็นมาตรฐานได้ในเดือนธันวาคมปี1987โดยอยู่ในหมวด IEEE 1076-1987 หลังจากนั้นก็ได้มีการพัฒนาปรับปรุงอย่างต่อเนื่อง โดยได้มีประกาศปรับปรุงอีกครั้งในปี1993โดยได้มีการเพิ่มเติม Syntax พิเศษเพื่อให้ผู้ใช้สามารถใช้งานได้สะดวกมากยิ่งขึ้น สำหรับขีดความ

เอกสารนี้สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สามารถในการออกแบบโดยใช้ภาษา วีเอชดีแอลนั้นสามารถออกแบบได้เฉพาะวงจรที่มีลักษณะเป็นดิจิตอลเท่านั้นส่วนวงจรที่เป็นอนาล็อกในขณะนี้ยังไม่สามารถออกแบบได้โดย

2.2 วีเอชดีแอล แอดวานซ์เตจ

ลักษณะที่เป็นประโยชน์ของ VHDL มีดังนี้คือ

- **Standard VHDL** เป็นมาตรฐานของ IEEE ทำให้มี Tools และบริษัทที่สนับสนุนการทำงานมากมาย นอกจากนี้วงจรที่ออกแบบโดย VHDL ก็จะใช้งานได้นานเนื่องจากมีความเข้ากันได้ของภาษากับวงจรที่ได้รับการออกแบบใหม่
- **Government Support** เนื่องจาก ได้รับการพัฒนาโดยกระทรวงกลาโหมของสหรัฐอเมริกา ดังนั้นการออกแบบวงจรโดยใช้ภาษา VHDL จึงได้รับการสนับสนุนจากรัฐบาลสหรัฐอเมริกา
- **Industrial Support** เนื่องจากภาษา VHDL เป็นภาษาที่เป็นมาตรฐานของ IEEE จึงมีอุตสาหกรรมจำนวนมากที่รองรับการออกแบบที่ใช้ภาษา VHDL
- **Portability** การออกแบบโดยใช้ภาษา VHDL สามารถนำไปจำลองการทำงานหรือสังเคราะห์ด้วยซอฟท์แวร์ตัวใดก็ได้ที่รองรับภาษา VHDL จึงทำให้การออกแบบด้วยภาษา VHDL จึงเป็นการออกแบบที่ไม่ยึดติดกับซอฟท์แวร์ที่ใช้ในการออกแบบ
- **Modeling capability** ผู้ออกแบบวงจรสามารถออกแบบวงจรโดยใช้ภาษา VHDL ได้หลายระดับตั้งแต่ระดับ Electronic boxes ถึงระดับทรานซิสเตอร์ และสามารถออกแบบวงจรที่มีความซับซ้อนสูงและมีขนาดใหญ่มากได้
- **Reusability** วงจรที่ออกแบบโดยภาษา VHDL สามารถนำกลับมาใช้ใหม่ได้ง่าย เนื่องจากสามารถเปลี่ยนแปลงแก้ไขวงจรได้ง่าย
- **Documentation VHDL** เป็นภาษาในรูปแบบบรรยายพฤติกรรม ทำให้เราสามารถอธิบายการทำงานของวงจรภายในการออกแบบได้ทันที

2.3 ลักษณะการใช้งานภาษา วีเอชดีแอล

การใช้งานภาษา VHDL อาจจำแนกออกเป็น 5 ประเภทดังนี้คือ

2.3.1 Document Language: ใช้สำหรับบรรยายรายละเอียดการทำงานของวงจรที่ออกแบบ

2.3.2 Design Language: ใช้สำหรับออกแบบวงจรที่มีความซับซ้อนสูงเพื่อใช้สำหรับการจำลองการทำงาน

2.3.3 Verification Language: ใช้ตรวจสอบการทำงานของวงจรที่ออกแบบว่ามีความถูกต้องหรือไม่

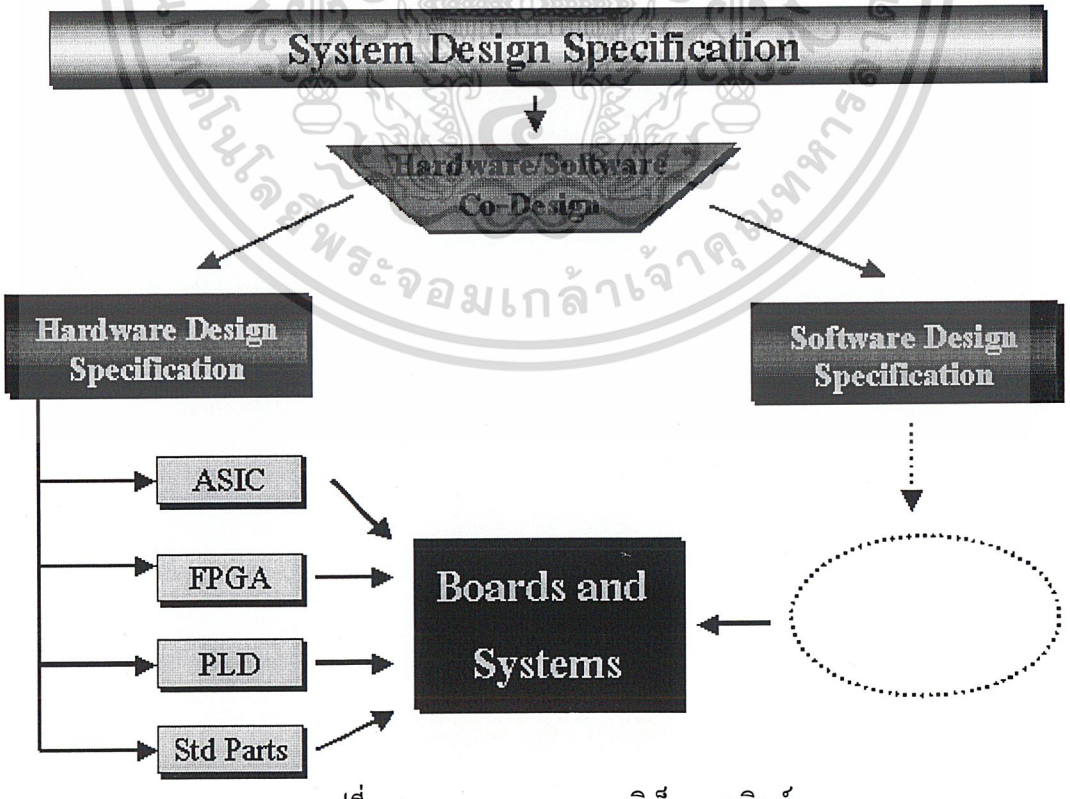
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.3.4 Test Language: ใช้สำหรับสร้าง Test Vector เพื่อใช้สำหรับเป็นข้อมูลที่ใช้สำหรับทดสอบการทำงานของวงจรที่ออกแบบ

2.3.5 Synthesis Language: ใช้สำหรับสร้างวงจรจริงเพื่อนำไป Implement เป็น Hardware ต่อไป

2.4 วิธีการออกแบบระบบอิเล็กทรอนิกส์

การออกแบบระบบอิเล็กทรอนิกส์ จะเริ่มต้นจากการกำหนดหน้าที่การทำงานของระบบอิเล็กทรอนิกส์ที่ต้องการซึ่งโดยส่วนใหญ่แล้ว ระบบอิเล็กทรอนิกส์มักจะประกอบด้วย 2 ส่วนคือ ส่วนที่เป็น Hardware และส่วนที่เป็น Software โดยทั่วไปแล้วการออกแบบทั้ง 2 ส่วนดังกล่าวนี้จะต้องทำไปพร้อมๆ กัน หลังจากนั้นจะเป็นขั้นตอนการกำหนดหน้าที่การทำงานของ Hardware และ Software โดยสำหรับการออกแบบทางด้าน Hardware สามารถเลือกได้ว่าต้องการออกแบบเป็น ASIC, FPGA, PLD หรือจาก Standard Component ที่มีขายอยู่ทั่วไปตามท้องตลาด สำหรับการออกแบบทาง Software ผู้ออกแบบสามารถเลือกใช้ภาษา Programming ต่างๆ เช่น ภาษา C/C++ ภาษา Assembly เป็นต้น สำหรับในการออกแบบ Firmware และ Application โดยในระหว่างการออกแบบนั้น ผู้ออกแบบทั้งทางด้าน Hardware และ Software จะต้องมีการทดสอบการทำงานร่วมกัน

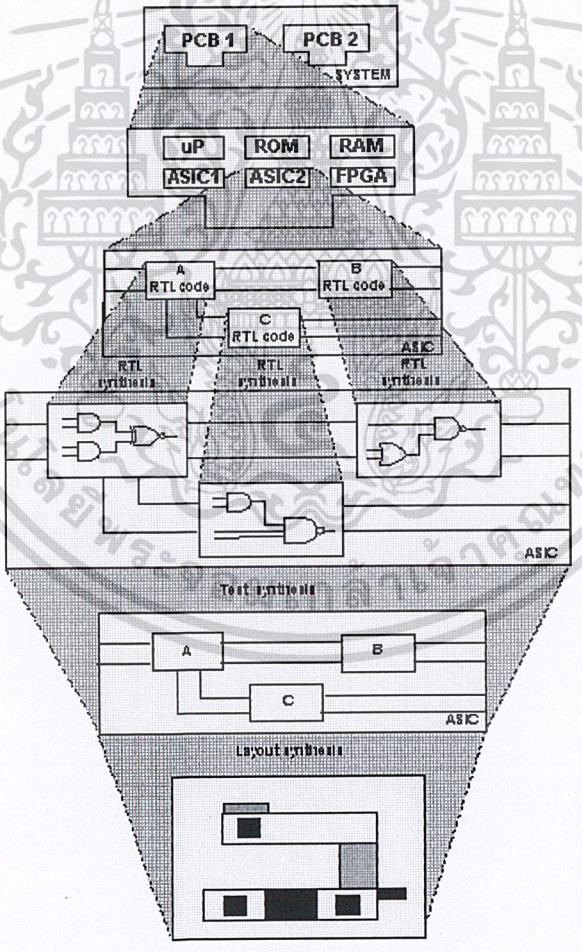


รูปที่ 2.1 การออกแบบระบบอิเล็กทรอนิกส์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.5 การออกแบบแบบบนลงล่าง (Top-Down Design)

เป็นการออกแบบที่เน้นความถูกต้องและประสิทธิภาพในการทำงานของวงจรที่ออกแบบ โดยจะใช้การออกแบบ ในลักษณะที่เรียกว่า High-level behavioral description จะมีการแบ่งวงจรทั้งหมดที่ต้องการออกแบบออกเป็นส่วน แล้วทำการออกแบบในแต่ละส่วนแล้วจำลองการทำงานของวงจรในแต่ละส่วนแยกอิสระออกจากกันเพื่อตรวจสอบความถูกต้องของหน้าที่การทำงานของวงจรและประสิทธิภาพการทำงานของวงจร ขั้นต่อไปก็จะนำการออกแบบไปสังเคราะห์ เพื่อให้ได้เป็นวงจรในระดับ Gate Level แล้วทำการจำลองการทำงาน เพื่อตรวจสอบความล่าช้าของการทำงานภายในวงจร และทดสอบประสิทธิภาพการทำงานของวงจรก่อนที่จะนำไป Implement เป็น Hardware



รูปที่ 2.2 การออกแบบบนลงล่าง (Top-down Design)

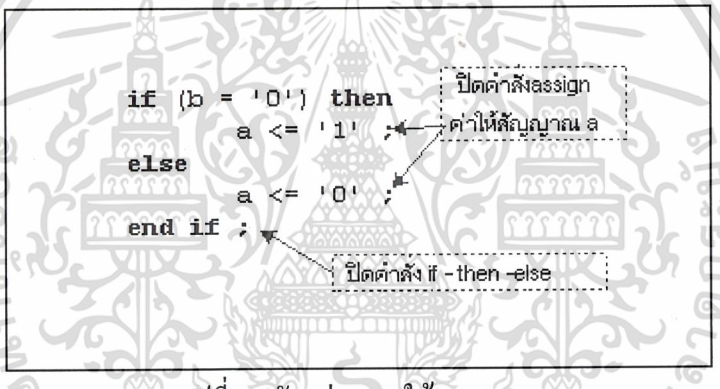
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.6 พื้นฐานการเขียนภาษาวีเอชดีแอล

2.6.1 ทั่วไป

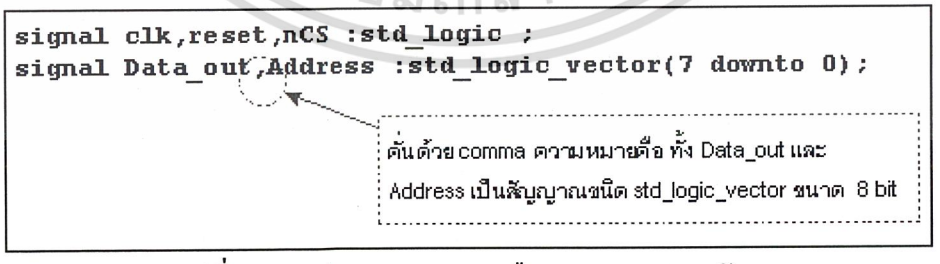
- การใช้ตัวอักษรในภาษา VHDL นั้นอักษรตัวเล็กหรือใหญ่จะมีความหมายเหมือนกัน ผู้เขียนสามารถกำหนดรูปแบบการใช้ขนาดตัวอักษรได้เอง เช่น signal, variable, port name ใช้อักษรตัวเล็ก Constant หรือ signal ที่กำหนดขึ้นมาเองใช้อักษรตัวใหญ่ เพื่อให้สะดวกในการอ่าน code แล้วสามารถแยกแยะได้ทันทีว่า code แต่ละส่วนมีความหมายยังไง

- การ assign ค่า signal ใช้เครื่องหมายน้อยกว่าหรือเท่ากับ (<=)แต่ละ statement จะต้องจบด้วยเครื่องหมาย semicolon (;) สามารถเขียนหลายๆ statement อยู่บนบรรทัดเดียวกันได้ โดยมี semicolon คั่นทุกๆ statement ในบาง statement อาจมี statement ย่อยๆซ้อนอยู่ภายในได้ เช่น if.... Then....else



รูปที่2.3 ตัวอย่างการใช้ Statement

- การประกาศ signal หรือ variable หลายๆ อัน ใช้เครื่องหมาย comma (,) คั่นแต่ละ signal , variable



รูปที่2.4 การประกาศ Signal หรือ Variable หลายอัน

- '--' ก่อนเขียน comment และจะจบการ comment ที่เมื่อขึ้นบรรทัดใหม่ end of line (เคาะ enter)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
signal clk    :std_logic ;    -- 50Mhz clock signal
signal Reset  :std_logic ;    -- Gobal Reset signal
```

รูปที่ 2.5 การประกาศสัญญาณ

2.6.2 การเขียน Comment

2.6.2.1 Identifiers

พยัญชนะ A - Z , a – z

ตัวเลข 0,1,2,3,4,5,6,7,8,9

เครื่องหมายขีดเส้นใต้ (Underscore) " – "

ต้องขึ้นต้นด้วยพยัญชนะเสมอ

ชื่อสามารถประกอบด้วยพยัญชนะ ตัวเลข และเครื่องหมายขีดเส้นใต้จำนวนไม่จำกัด

การใช้เครื่องหมายขีดเส้นใต้ทุกครั้ง ต้องนำหน้าด้วยพยัญชนะหรือตัวเลขและตามด้วยพยัญชนะหรือตัวเลขเช่นกัน

พยัญชนะตัวเล็กหรือตัวใหญ่มีความหมายเหมือนกัน

ห้ามใช้คำสงวน (reserved word) ของภาษา VHDL

2.7 โครงสร้างภาษา วีเอชดีแอล

ภาษา VHDL เป็นภาษาที่มีลักษณะเป็นโครงสร้าง โดยจะประกอบด้วยส่วนต่างๆดังนี้

- Entity

- Architecture

- Component

- Package

- Library

- Configuration

แต่ละส่วนหรือแต่ละelementเหล่านี้จะมีหน้าที่ต่างๆกันออกไป ในการออกแบบด้วยภาษา VHDL จะต้องนำแต่ละส่วนเหล่านี้มาประกอบกัน โดยจะกล่าวถึงรายละเอียดต่อไป

2.7.1 การ Declaration of VHDL Objects

นอกจาก Elements ต่างๆที่กล่าวมา ในภาษา VHDL ยังมีส่วนประกอบอีกประเภทหนึ่ง เรียกว่า Objects ถ้าเปรียบเทียบกับภาษาในการเขียนโปรแกรมทางซอฟแวร์ Objects จะเปรียบเทียบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ได้กับ ตัวแปรและค่าคงที่ หรือเปรียบได้เหมือนภาชนะไว้สำหรับเก็บค่าต่างๆ สามารถแบ่งออกได้ 3 class คือ

- **Constant** เป็น Object ที่กำหนดค่าเริ่มต้นไว้ แล้วไม่สามารถเปลี่ยนแปลงค่าได้
- **Signal** เป็น Object ที่สามารถกำหนดค่าที่สัมพันธ์กับเวลาให้ได้
- **Variable** เป็น Object ที่สามารถกำหนดค่าให้ได้และสามารเปลี่ยนแปลงค่าได้ตลอดการจำลองการทำงานในส่วนรายละเอียดในจะกล่าวของแต่ละ Object จะกล่าวถึงในบทที่ 3

จะเห็นได้ว่าในภาษา VHDL จะประกอบไปด้วยส่วนประกอบต่างๆหลายส่วนด้วยกันและแต่ละส่วนก็จะมีตำแหน่งการสร้าง และตำแหน่งการประกาศค่าต่างกันออกไปดังนี้

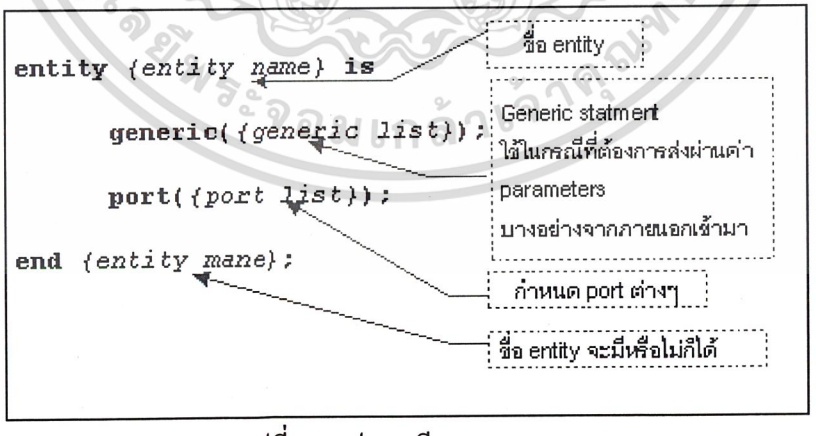
	Entity	Architecture	Process/Subprogram	Package
Subprogram	X	X	X	X
Component		X		X
Configuration		X		
Constant	X	X	X	X
Data type	X	X	X	X

รูปที่ 2.6 ความสัมพันธ์ของส่วนประกอบต่างๆ

2.7.2 รูปแบบของ Entity และ Architecture

2.7.2.1 Entity

เป็นส่วนที่แสดงพอร์ตต่างๆของวงจร รวมทั้งการส่งผ่านค่าพารามิเตอร์บางอย่างระหว่างวงจรกับภายนอก รูปแบบการเขียนเป็นดังนี้

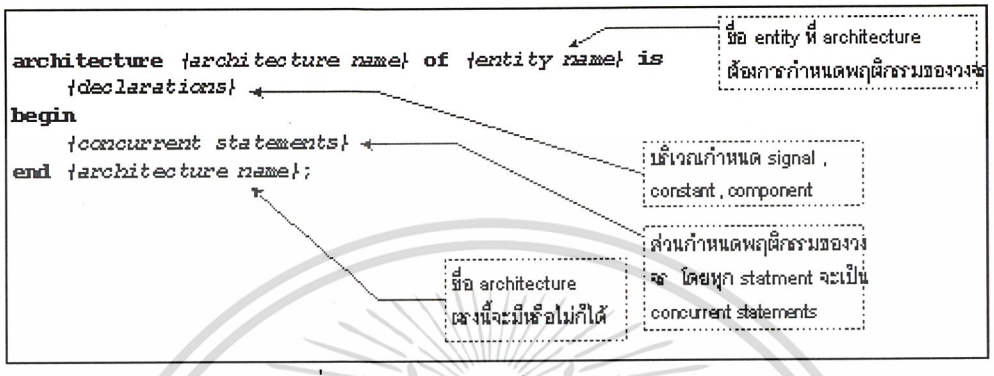


รูปที่ 2.7รูปการเขียน Entity

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.7.2.2 Architecture

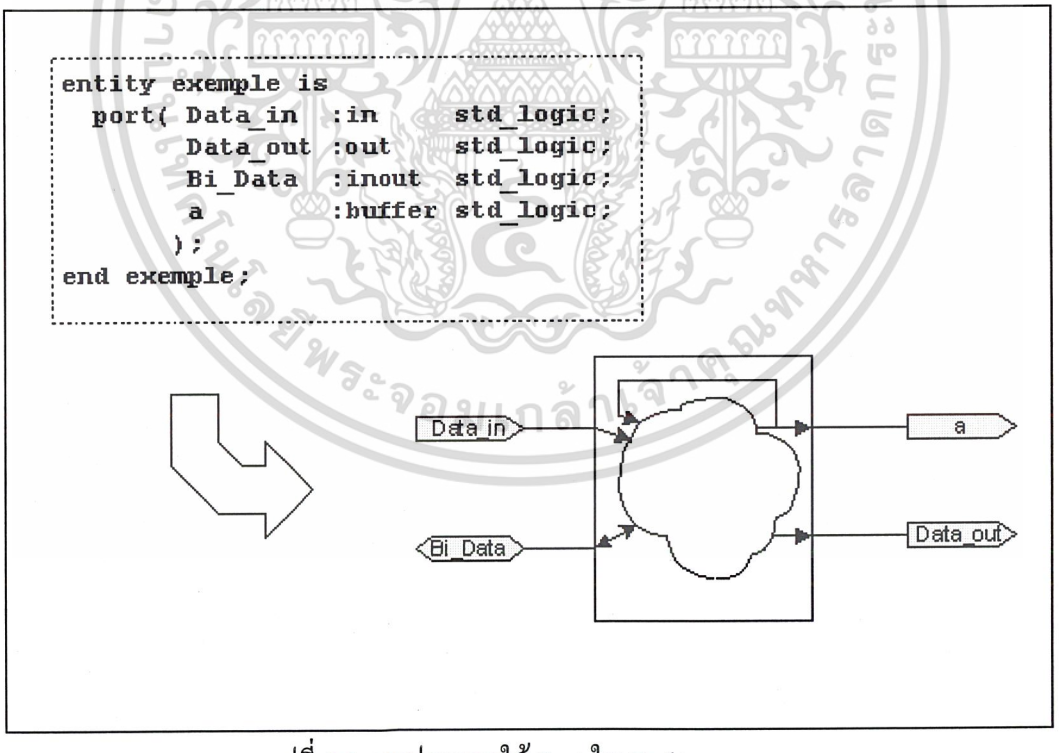
โดยเป็นส่วนสำหรับเขียน statement ต่างๆเพื่อกำหนดพฤติกรรมการทำงานของวงจรที่ต้องการออกมีพอร์ตติดต่อกับภายนอกตามที่กำหนดไว้ใน Architecture มีรูปแบบการเขียนดังนี้



รูปที่ 2.8 รูปแบบการเขียน Achitecture

2.7.3 Entity port mode

ประเภทของ port ที่ประกาศใน entity ประกอบด้วย



รูปที่ 2.9 การประกาศใช้ Port ใน Entity

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

input : เป็น port ทิศทางเดียว ที่นำค่าสัญญาณจากภายนอก เข้ามาภายใช้ภายในวงจรที่แสดงโดย Entity นั้นสามารถนำมา Source ให้กับสัญญาณอื่นได้ แต่ไม่สามารถเป็น Destination ใน Assign statement ได้จากตัวอย่าง entity ด้านบนนำมาพิจารณาตัวอย่างได้ดังนี้

output : เป็น port ทิศทางเดียว นำค่าสัญญาณภายในออกไปสู่ภายนอกไม่สามารถเป็น Source ให้กับสัญญาณใดๆใน Architectureได้ เพราะมีทิศทางออกไปภายนอกทิศทางเดียว

inout : เป็น port 2 ทิศทาง สามารถส่งสัญญาณเข้า และออกได้

buffer : เป็น port output ที่สามารถ Feedback สัญญาณที่ port กลับเข้ามาในวงจรได้

จากตัวอย่างด้านบนแสดงให้เห็นถึงลักษณะความเป็น Hierachy ของภาษา VHDL เป็นการสร้างวงจร FULL ADDER ขนาด 1 bit โดย Block บนสุดประกอบด้วยวงจรต่างๆ แบ่งเป็น 3 Block เป็น Half adder(U1,U2) และ OR gate สามารถเขียนเป็นภาษา VHDL ได้ตาม Entity ที่ชื่อว่า FULLADD ซึ่งการเขียนลักษณะนี้เรียกว่าการเขียนแบบ Structural คือเป็นการแสดงการเชื่อมต่อแต่ละ Module เข้าด้วยกัน จะไม่มีการแสดงพฤติกรรมทางวงจรแต่อย่างใด และในแต่ละ Block ย่อยๆก็สามารถสร้างเป็น Entity และ Architecture ใหม่แยกต่างหากได้

2.7.4 Component Declaration

เป็นการประกาศว่า Module ย่อยที่ต้องการนำมาเชื่อมต่อกัน ในการเขียนแบบStructuralนั้นมี Port อะไร,เข้าออกอย่างไรบ้าง โดยทั่วไปในขั้นตอนการออกแบบจะทำการออกแบบวงจร Module ย่อยๆก่อน โดยสร้างเป็น Entity และ Architecture ไว้ แล้วจากนั้นก็สร้าง Entity และ Architecture อีกอันเพื่อแสดงการเชื่อมต่อ Module ย่อยๆเหล่านั้น ใน Module นี้จะต้องมีการสร้าง Component Declaration เพื่อแสดงถึงPortของวงจรย่อยๆเหล่านั้น เพื่อที่จะสร้างการเชื่อมต่อกันด้วย Signal ต่อไป รูปแบบการเขียนเป็นดังนี้

```
component {component_name}
    generic( {generic_list} );
    port( {port list} );
end component;
```

รูปที่ 2.10 การประกาศ Component

การทำ Component Declaration สามารถสร้างไว้ใน Architecture โดยจะอยู่บริเวณเดียวกันกับ Signal Declaration คือก่อนถึง Begin ของ Architecture ดังในตัวอย่างต่อไป และนอกจากนี้ยังสามารถสร้างไว้ใน Package โดยจะกล่าวต่อไปเรื่องของ Package

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ได้กับ ตัวแปรและค่าคงที่ หรือเปรียบได้เหมือนภาชนะไว้สำหรับเก็บค่าต่างๆ สามารถแบ่งออกได้ 3 class คือ

- **Constant** เป็น Object ที่กำหนดค่าเริ่มต้นไว้ แล้วไม่สามารถเปลี่ยนแปลงค่าได้
- **Signal** เป็น Object ที่สามารถกำหนดค่าที่สัมพันธ์กับเวลาให้ได้
- **Variable** เป็น Object ที่สามารถกำหนดค่าให้ได้และสามารเปลี่ยนแปลงค่าได้ตลอดการจำลองการทำงานในส่วนรายละเอียดในจะกล่าวของแต่ละ Object จะกล่าวถึงในบทที่ 3

จะเห็นได้ว่าในภาษา VHDL จะประกอบไปด้วยส่วนประกอบต่างๆหลายส่วนด้วยกันและแต่ละส่วนก็จะมีตำแหน่งการสร้าง และตำแหน่งการประกาศค่าต่างกันออกไปดังนี้

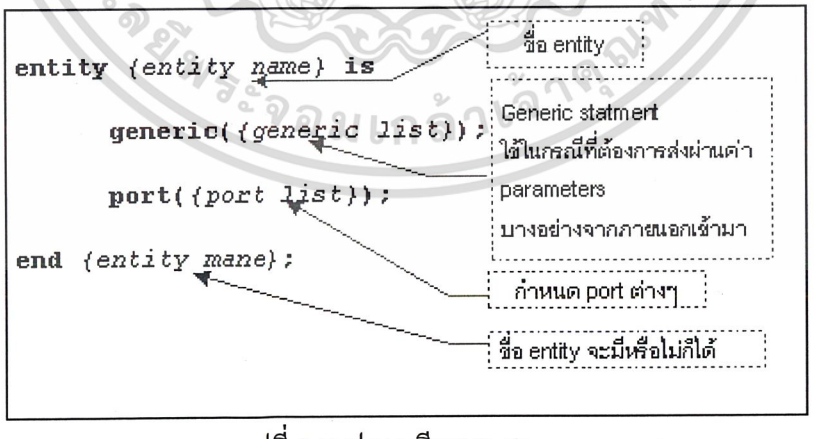
	Entity	Architecture	Process/Subprogram	Package
Subprogram	X	X	X	X
Component		X		X
Configuration		X		
Constant	X	X	X	X
Data type	X	X	X	X

รูปที่ 2.6 ความสัมพันธ์ของส่วนประกอบต่างๆ

2.7.2 รูปแบบของ Entity และ Architecture

2.7.2.1 Entity

เป็นส่วนที่แสดงพอร์ตต่างๆของวงจร รวมทั้งการส่งผ่านค่าพารามิเตอร์บางอย่างระหว่างวงจรกับภายนอก รูปแบบการเขียนเป็นดังนี้

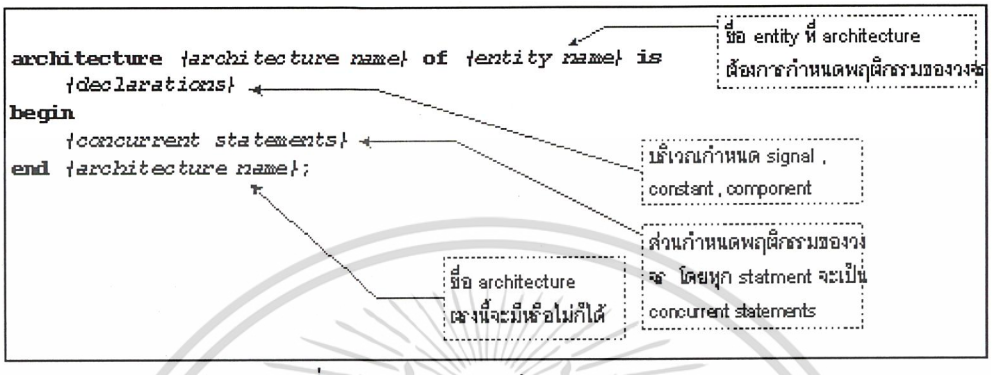


รูปที่ 2.7รูปการเขียน Entity

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.7.2.2 Architecture

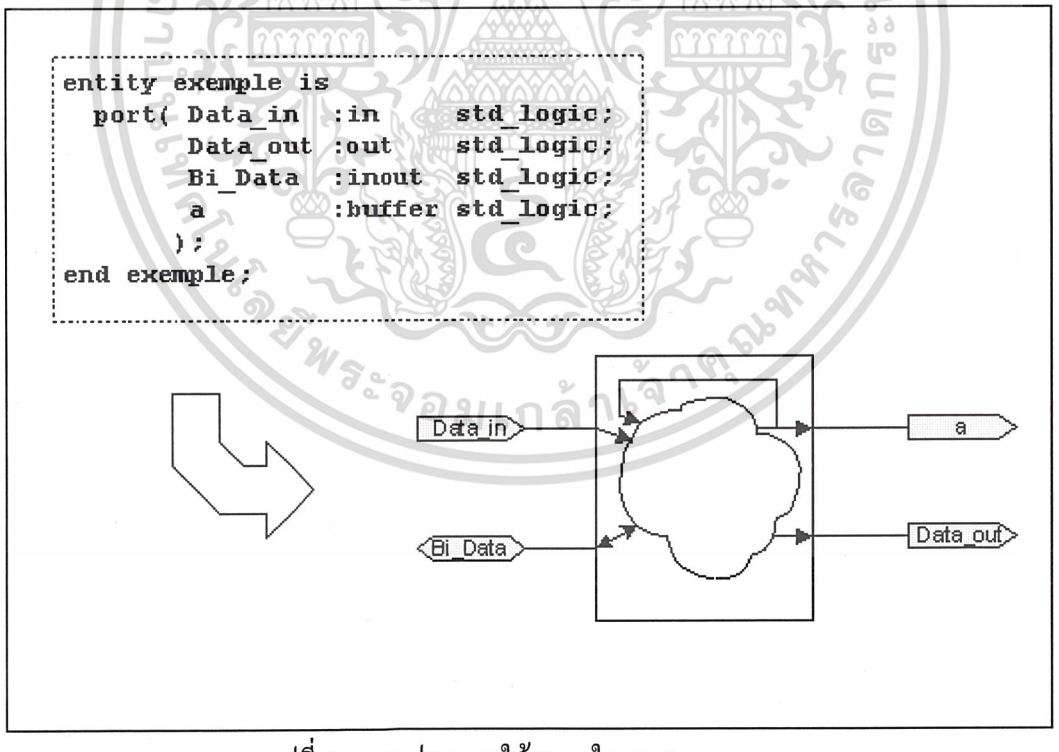
โดยเป็นส่วนสำหรับเขียน statement ต่างๆเพื่อกำหนดพฤติกรรมการทำงานของวงจรที่ต้องการออกมีพอร์ตติดต่อกับภายนอกตามที่กำหนดไว้ใน Architecture มีรูปแบบการเขียนดังนี้



รูปที่ 2.8 รูปแบบการเขียน Achitecture

2.7.3 Entity port mode

ประเภทของ port ที่ประกาศใน entity ประกอบด้วย



รูปที่ 2.9 การประกาศใช้ Port ใน Entity

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

input : เป็น port ทิศทางเดียว ที่นำค่าสัญญาณจากภายนอก เข้ามาภายใช้ภายในวงจรที่แสดงโดย Entity นั้นสามารถนำมา Source ให้กับสัญญาณอื่นได้ แต่ไม่สามารถเป็น Destination ใน Assign statement ได้จากตัวอย่าง entity ด้านบนนำมาพิจารณาตัวอย่างได้ดังนี้

output : เป็น port ทิศทางเดียว นำค่าสัญญาณภายในออกไปสู่ภายนอกไม่สามารถเป็น Source ให้กับสัญญาณใดๆใน Architecture ได้ เพราะมีทิศทางออกไปภายนอกทิศทางเดียว

inout : เป็น port 2 ทิศทาง สามารถส่งสัญญาณเข้า และออกได้

buffer : เป็น port output ที่สามารถ Feedback สัญญาณที่ port กลับเข้ามาในวงจรได้

จากตัวอย่างด้านบนแสดงให้เห็นถึงลักษณะความเป็น Hierachy ของภาษา VHDL เป็นการสร้างวงจร FULL ADDER ขนาด 1 bit โดย Block บนสุดประกอบด้วยวงจรต่างๆ แบ่งเป็น 3 Block เป็น Half adder(U1,U2) และ OR gate สามารถเขียนเป็นภาษา VHDL ได้ตาม Entity ที่ชื่อว่า FULLADD ซึ่งการเขียนลักษณะนี้เรียกว่าการเขียนแบบ Structural คือเป็นการแสดงการเชื่อมต่อแต่ละ Module เข้าด้วยกัน จะไม่มีการแสดงพฤติกรรมทางวงจรแต่อย่างใด และในแต่ละ Block ย่อยๆก็สามารถสร้างเป็น Entity และ Architecture ใหม่แยกต่างหากได้

2.7.4 Component Declaration

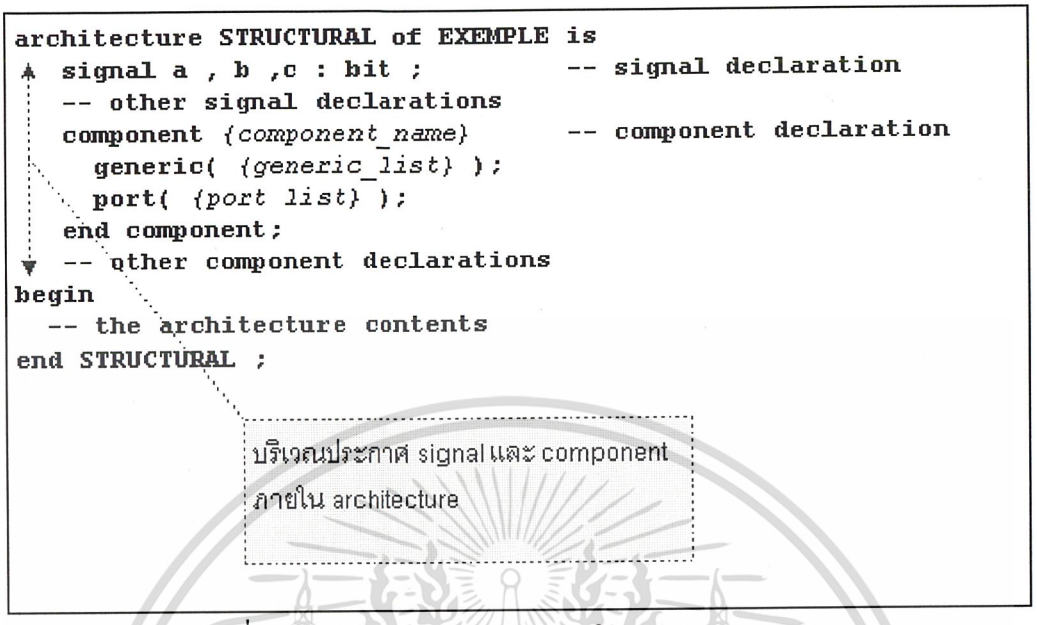
เป็นการประกาศว่า Module ย่อยที่ต้องการนำมาเชื่อมต่อกัน ในการเขียนแบบStructuralนั้นมี Port อะไร,เข้าออกอย่างไรบ้าง โดยทั่วไปในขั้นตอนการออกแบบจะทำการออกแบบวงจร Module ย่อยๆก่อน โดยสร้างเป็น Entity และ Architecture ไว้ แล้วจากนั้นก็สร้าง Entity และ Architecture อีกอันเพื่อแสดงการเชื่อมต่อ Module ย่อยๆเหล่านั้น ใน Module นี้จะต้องมีการสร้าง Component Declaration เพื่อแสดงถึงPortของวงจรย่อยๆเหล่านั้น เพื่อที่จะสร้างการเชื่อมต่อกันด้วย Signal ต่อไป รูปแบบการเขียนเป็นดังนี้

```
component {component_name}
    generic( {generic_list} );
    port( {port list} );
end component;
```

รูปที่ 2.10 การประกาศ Component

การทำ Component Declaration สามารถสร้างไว้ใน Architecture โดยจะอยู่บริเวณเดียวกันกับ Signal Declaration คือก่อนถึง Begin ของ Architecture ดังในตัวอย่างต่อไป และนอกจากนี้ยังสามารถสร้างไว้ใน Package โดยจะกล่าวต่อไปเรื่องของ Package

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.11 การประกาศ Component ใน Architecture

สำหรับ Component นั้นถ้าเปรียบเทียบกับการออกแบบวงจรบน PCB (Print Circuit Bord) นั้นจะเปรียบเสมือนว่าเป็นซ็อกเก็ตว่างๆ สำหรับเสียบ IC ลงไป โดยซ็อกเกตมีหน้าที่เพียงเชื่อมต่อสัญญาณเท่านั้นไม่มีหน้าที่สร้างฟังก์ชั่นการทำงาน Component ในภาษา VHDL ก็เช่นเดียวกัน จะไม่มีการแสดงถึงฟังก์ชันการทำงาน แต่มีหน้าที่เพียงเชื่อมต่อสัญญาณต่างๆเข้ากับ Port ของ Component นั้น ๆ

2.7.5 Component Instantiation

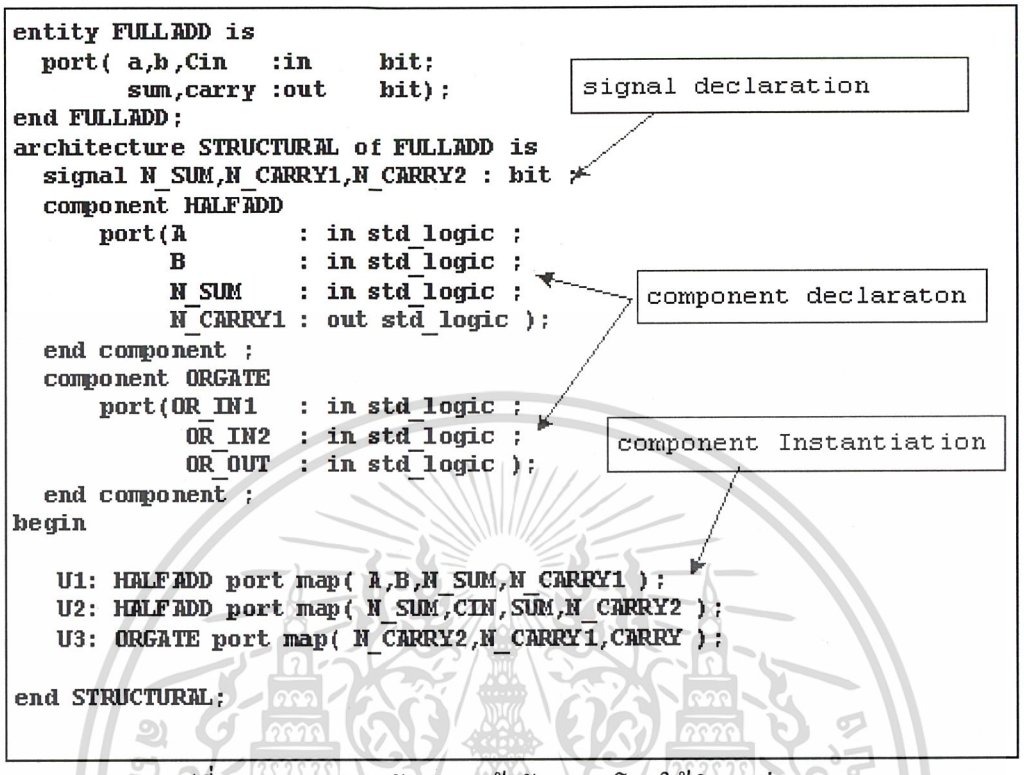
ในการเขียน Code ในรูปแบบ structural เมื่อประกาศ component แล้ว จะต้องทำการเชื่อมต่อสัญญาณเข้ากับ port ต่างๆ ของ component ที่ประกาศไว้ โดยเขียนอยู่ในรูปแบบดังนี้

```
instance_label : component_name
        generic map( generic_association_list )
        port map( port_association_list );
```

รูปที่ 2.12 การทำ Component Instantiation

component Instantiation นี้เป็นconcurrent statement จะอยู่ภายใต้ Architecture คืออยู่ระหว่าง Begin และ End ของ Architecture เท่านั้น ดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.13 การ Map สัญญาณเข้ากับ Port โดยใช้ตำแหน่ง

ในการ map สัญญาณเข้ากับ port จะทำได้ 2 รูปแบบคือ map สัญญาณเข้ากับ port โดยพิจารณาจากตำแหน่งของพอร์ต และตำแหน่งสัญญาณใน port_association_list ที่ตรงกันจะมีความหมายว่าต่อถึงกันอยู่ ไม่ต้องมีการอ้างชื่อพอร์ตเพียงแต่วางตำแหน่งให้ตรงกันก็พอ ดังในตัวอย่างด้านบน พิจารณาที่ ORGATE จะหมายความว่า พอร์ต OR_IN1 ต่ออยู่กับสัญญาณ N_CARRY2 , OR_IN2 ต่อกับสัญญาณ N_CARRY1 และ OR_OUT ต่อกับสัญญาณ CARRY

อีกรูปแบบจะเป็นการ mapด้วยชื่อของ port และชื่อของสัญญาณ จะไม่พิจารณาตำแหน่ง แต่ต้องอ้างถึงชื่อของพอร์ตด้วย ซึ่งต่างจากรูปแบบแรก ดังตัวอย่างต่อไปนี้ เป็นวงจรเดียวกับตัวอย่างก่อนหน้า แต่ใช้วิธี component instantiation อีกแบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
architecture STRUCTURAL of FULLADD is
            :
            :
begin

   U1: HALFADD port map( A         => A,
                         B         => B,
                         N_SUM     => NSUM,
                         N_CARRY1  => N_CARRY1 );

   U2: HALFADD port map( A          => N_SUM ,
                         B          => CIN ,
                         N_SUM      => SUM,
                         N_CARRY2   => N_CARRY2);

   U3: ORGATE port map( OR_IN1  => N_CARRY2,
                        OR_IN2  => N_CARRY1,
                        OR_OUT  => CARRY );

end STRUCTURAL;
```

signal name

port name

Instance label

component name

รูปที่ 2.14 การ Map สัญญาณเข้ากับ Port โดยใช้ชื่อ

2.7.6 การ Configuration

ในการออกแบบวงจรด้วย VHDL ต้องมีการสร้าง Entity และภายใต้ Entity ก็มีการสร้าง Architecture ซึ่งสามารถสร้าง Architecture ได้มากกว่า 1 Architecture ต่อ Entity นอกจากนี้ภายใน Architecture ยังมี component ที่อ้างถึง Entity อื่นๆ อีก สิ่งที่ใช้บอกให้ simulator ทราบว่า Architecture ใดอยู่ภายใต้ Entity อันไหน และ component แต่ละตัวอ้างถึง Entity อันไหนเรียกว่า Configuration โดยมีรูปแบบดังนี้

```
configuration config_name of entity_name is
  for architecture_name
    for instance_label:component_name
       use entity library_name.entity_name(arch_name);
          for arch_name
            ...
             low-level configuration specifications
            ...
          end for;
    end for;
  end for;
end config_name;
```

รูปที่ 2.15 การใช้งาน Configuration

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

นอกจากในรูปแบบข้างต้นแล้ว ยังสามารถใช้ configuration ซ้อน configuration ได้ดังรูปแบบด้านล่างนี้

```
configuration config_name of entity_name is
  for architecture_name
    for instance_label:component_name
       use configuration library_mane.cfg_name;
    end for;
  end for;
end config_name;
```

รูปที่ 2.17 การใช้ Configuration ซ้อน Configuration

Configuration ส่วนใหญ่จะไม่สามารถนำไปใช้กับ Synthesis tools ได้ จะใช้เฉพาะขั้นตอนการ Simulation เท่านั้น เพราะฉะนั้นในขั้นตอนการ Synthesis จะต้องให้มี 1 Entity และ 1 Architecture เท่านั้น และ component และ Entity ที่อ้างถึงจะต้องให้ชื่อตรงกัน และมี port ตรงกัน วิธีนี้เรียกว่า Default configuration

2.7.7 Package

เป็นส่วนเก็บข้อมูล Subprogram หรือสิ่งที่มีประโยชน์ในการเขียนบรรยายพฤติกรรมของวงจร ข้อมูลใน package สามารถเรียกใช้ได้โดย Entity, Architecture หรือด้วย Package ด้วยกันเอง สิ่งที่สามารถสร้างไว้ใน Package ได้แก่

Subprogram

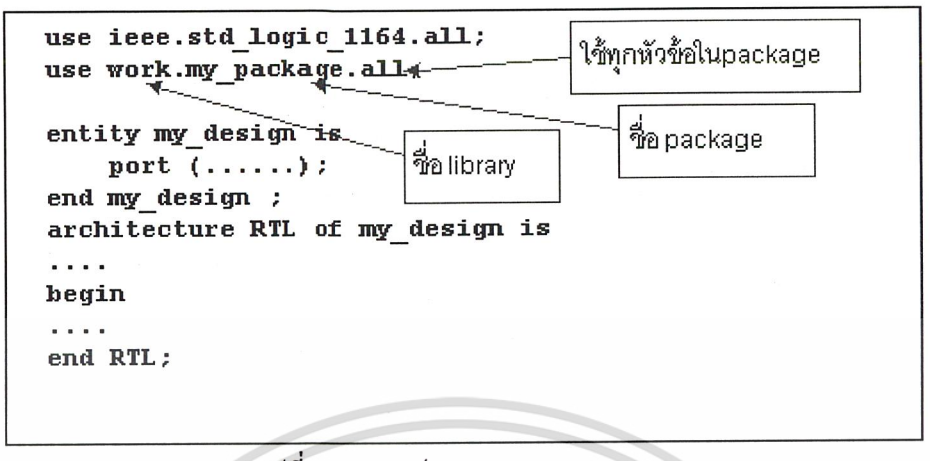
- Types
- Constants
- Signals
- Aliases
- Attributes
- Component
- Disconnection Specification

โดยใช้คำสั่ง USE กำหนดไว้ที่ต้นไฟล์ร่วมกับคำสั่ง library เพื่อประกาศชื่อ library ที่มี package ที่ต้องการใช้งานอยู่

```
use library_name.package_name.item;
```

รูปที่ 2.18 การใช้งานคำสั่ง Use

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.19 การประกาศ Library และ Package

ในตัว package เองนั้นจะประกอบไปด้วย 2 ส่วนคือ

2.7.7.1 Package Declaration สำหรับนำไปใช้ภายนอกตัวของ package เปรียบเทียบได้กับการประกาศพอร์ตใน Entity มีรูป1. Package declaration จะเป็นส่วนที่กำหนดชื่อ(identifier) ของสิ่งที่ประกาศอยู่ภายใน แบบดังนี้

```
package package_name is
    declarations
end package_name ;
```

รูปที่ 2.20 รูปแบบของ Package declaration

2.7.7.2 Package body โครงสร้างที่ประกอบด้วยคำสั่งต่างๆ ในรูปของคำสั่งลำดับ (sequential statement) ที่ใช้บรรยายฟังก์ชันการทำงานของโปรแกรมย่อย (subprogram) ที่ได้ประกาศไว้ใน Package declaration แล้ว

2.7.8 Library

เป็นส่วนที่เก็บ design unit ต่างๆ ที่ถูกทำการ Compiled แล้ว เพื่อให้พร้อมที่จะนำไปใช้ใน design unit อื่นๆที่อ้างถึงได้ สิ่งที่เก็บไว้ใน library ได้แก่

Primary units

Entity declarations

Package declarations

Configuration specifications

Secondary units

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

Architecture bodies

Package bodies

ส่วน Secondary units เป็นส่วนที่ต้องอ้างถึง Primary units ดังนั้น จะต้องคอมไพล์ในส่วน Primary units ก่อนเสมอ เช่น คอมไพล์ package declarations ก่อนแล้วจึงคอมไพล์ Package bodies, คอมไพล์ Entity ก่อน Architecture

ก่อนที่จะเรียกใช้ design units ต่างๆหลังจากคอมไพล์เก็บไว้ใน library แล้ว จะต้องประกาศชื่อ Library นั้นก่อนเสมอ

2.8 ชนิดของข้อมูล

มีอยู่ 3แบบ คือ

2.8.1 Constant

เป็นการกำหนดค่าคงที่ เมื่อประกาศค่าไว้แล้ว จะไม่สามารถเปลี่ยนแปลงค่าได้ นำไปใช้ในการกำหนดพฤติกรรมวงจร

```
constant constant_name :type := value;
```

รูปที่ 2.21 รูปแบบการประกาศ Constant

การประกาศ Constant จะประกาศ ไว้ส่วนต้นของ Architecture , package declaration ,package body , Entity หรือในส่วน process declaration ซึ่งจะมี scope ต่างกันออกไปขึ้นอยู่กับว่าประกาศไว้ที่ไหน

Entity : ประกาศโดยใช้ Generic statement ซึ่งจะต่างกับการประกาศในที่อื่น โดยค่า constant ที่ประกาศในบริเวณนี้สามารถนำไปใช้ในทุกๆ architecture ที่อ้างถึง Entity นั้นๆ

Architecture: ประกาศไว้บริเวณเดียวกับ signal สามารถนำไปใช้ได้ทุกที่ใน Architecture

Package :การประกาศใน Pacakge ทำได้ 2 แบบคือประกาศใน Package Declaration อย่างเดียว หรือ ประกาศไว้ใน Package Declaration แล้วกำหนดค่าไว้ใน Pacakge body ดังตัวอย่าง

Process :การประกาศใน Process นี้ค่าที่ประกาศไว้สามารถใช้ได้เฉพาะใน Process นั้นเท่านั้น จะใช้นอก process ที่ประกาศไว้ไม่ได้ และไม่จำเป็นต้องใส่ constant ไว้ใน sensitivity list เพราะเป็นค่าคงที่ ไม่มีการเปลี่ยนแปลงค่า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

46456

2.8.2 Signal

มีหน้าที่ส่งผ่านข้อมูลระหว่าง concurrent statement ภายใน Architecture และใช้ใน sequential statement ภายใน process เพื่อกำหนดพฤติกรรมการทำงานของวงจร signal จะถูกประกาศไว้ที่บริเวณต่างๆ ได้เช่นเดียวกับ constant คือใน package ,package body, Entity, Architecture และมี scope เช่นเดียวกับ constant

```
signal signal_name : type ;
```

```
signal signal_name : type := initial_value ;
```

รูปที่ 2.22 การประกาศ Signal

การประกาศ signal โดยมีการให้ค่าเริ่มต้น (intitial_value) นี้เมื่อนำไปสังเคราะห์เป็นวงจร ค่า intitial_valueนี้จะไม่ถูกนำไปสังเคราะห์ การ Assign ค่าให้กับ signal จะใช้เครื่องหมายน้อยกว่าหรือเท่ากับ " <= " ในความหมายคือการ Assign ค่าจากขวามือของเครื่องหมายให้กับ Signal ทางซ้ายมือ ดังตัวอย่างต่อไปนี้

2.8.3 Variable

เป็นตัวแปรชนิด Local ที่กำหนดไว้ใน process และ subprogram เพื่อกำหนดพฤติกรรมแบบ sequential การประกาศ Variable จะถูกประกาศไว้ในส่วนต้นๆของ Process และจะถูกใช้ใน Process นั้นเท่านั้น ไม่สามารถนำ Variable ไปใช้นอก Process ได้ มีคุณสมบัติต่างจาก signal ตรงที่สามารถ update ค่าได้ทันทีที่ถูก assign ค่าให้ ซึ่ง signal ที่ถูกใช้ภายใน process จะ updateค่าหลังจากจบ process แล้วเท่านั้น

```
variable variable_name : type ;
```

```
variable variable_name : type := initial_value ;
```

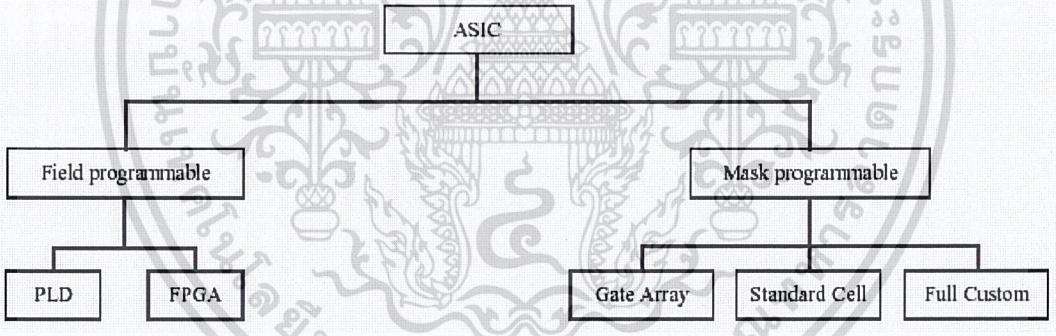
รูปที่2.23รูปแบบการประกาศ Variable

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่3

เอฟพีจีเอ และการออกแบบวงจรดิจิตอลเพื่อใช้งานกับเอฟพีจีเอ

ความก้าวหน้าของอุตสาหกรรมอิเล็คทรอนิกส์ปัจจุบันทำให้เกิดการพัฒนาความสามารถของอุปกรณ์ต่างๆ มากมายซึ่งทำให้เกิดการลดค่าใช้จ่าย การสิ้นเปลืองพลังงานและขนาด ในขณะเดียวกันก็มีการเพิ่มประสิทธิภาพและระดับความเชื่อถือได้ของวงจรรวมที่สูงขึ้นเห็นได้ชัดจากเทคโนโลยีไมโครโพรเซสเซอร์และหน่วยความจำปัจจุบันทุกๆ ครั้งที่มีการพัฒนาขึ้นทำให้เกิดช่องวงจรรวมและไอซีมาตรฐานมากขึ้นในการพัฒนาเพิ่มความหนาแน่น และจำนวน ฟังก์ชั่นลอจิกที่เหมาะสมนักออกแบบอุปกรณ์ทางด้านดิจิตอลได้พิจารณาถึงการผลิตให้ขนาดมากและการผลิตวงจรรวม (ASIC: Application Specific Integrated Circuit)ซึ่งวงจรรวม จะแบ่งตามการสร้างออกเป็น 2 กลุ่มคือ Field programmable และ Mask programmableดังแสดงในรูปที่ 3.1



รูปที่ 3.1 ผังแสดงการแบ่งกลุ่มของวงจรรวม ASIC

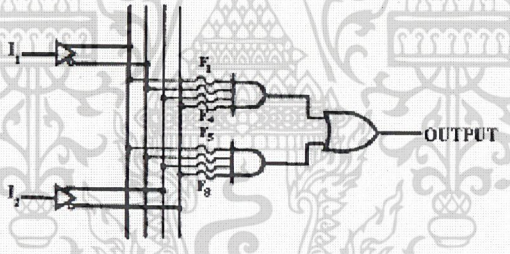
3.1 ฟิลด์ โปรแกรมเมเบิล (Field Programmable)

อุปกรณ์วงจรรวม ASIC แบบ field programmable มีอยู่มากมายหลายชนิด แต่มีลักษณะการสร้างหรือกำหนดการทำงานของวงจรที่เหมือนกัน กล่าวคือ ผู้ใช้งานสามารถออกแบบและสร้างวงจรที่ต้องการใช้ลงในตัวอุปกรณ์ได้เองโดยไม่ต้องไปโรงงานเพื่อผลิต โดยเฉพาะอย่างยิ่งในปัจจุบันนี้มีเครื่องมือที่ใช้ช่วยในการออกแบบ และสร้างวงจรร่วมกับไมโครคอมพิวเตอร์ที่มีความสามารถสูงในการพัฒนาตั้งแต่ขั้นการออกแบบ การจำลองการทำงานจนถึงจัดสร้างวงจรลงในอุปกรณ์ รวมทั้งอุปกรณ์ field programmable เหล่านี้ สามารถหาซื้อได้ง่ายทำให้การสร้างวงจรอิเล็กทรอนิกส์จนถึงระบบไมโครโพรเซสเซอร์หันมาใช้อุปกรณ์จำพวกนี้เป็นอุปกรณ์ประกอบในงานแทนอุปกรณ์ย่อยๆ แยกชิ้น (Discrete component) มาก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.1.1 พีแอลดี (PLD: Programmable Logic Device)

ภายในอุปกรณ์พีแอลดีถูกเตรียมเป็นวงจรพื้นฐานทางด้านลอจิกต่อกันอยู่เป็นกลุ่มมีทั้งวงจรคอมบิเนชั่น (Combination) และซีเควนเชียล (Sequential) ซึ่งมีส่วนประกอบเป็นวงจรภายในเทคโนโลยีของวงจรที่ใช้สร้างพีแอลดีมีทั้ง ทีทีแอล (TTL) อีซีแอล (ECL) และซีมอส(CMOS) ตามความเหมาะสมของแต่ละระบบอุปกรณ์อีแอลดีทุกชนิดมีหลักการพื้นฐานของวงจรภายในที่เหมือนกันโดยมีวงจรหลักเป็นวงจรคอมบิเนชั่นที่ให้ผลเป็นผลคูณร่วมบวก (Sum of produce) ดังรูปที่ 3.2 ประกอบไปด้วยชุดของแอนด์เกตที่ต่อร่วมกับออเกตการโปรแกรมคือ การเลือกว่าจะให้มีการต่ออินพุตในของแอนด์เกตกับสัญญาณอินพุตใดบ้างซึ่งมีทั้งจากภายนอกและสัญญาณป้อนกลับจากเอาต์พุตภายในเอง การติดต่ออินพุตของออเกตกับเอาต์พุตของ แอนด์เกต ตัวต่างๆ วิธีการเลือกหรือการโปรแกรมทาบงกายภาพ อินพุตต่างๆ ของอุปกรณ์ทุกตัสตะถูกต่อผ่านฟิวส์เข้ากับแหล่งสัญญารณ ซึ่งถ้าไม่ต้องการใช้สัญญาณใดจะตัดฟิวส์ทำให้สามารถโปรแกรมได้ครั้งเดียว อุปกรณ์พีแอลดีบางชนิดใช้มอสทรานซิสเตอร์แทนฟิวส์ทำให้สามารถโปรแกรมโดยให้กระแสไฟฟ้า และสามารถลบและโปรแกรมใหม่เข้าไปได้อีก



รูปที่ 3.2 แสดงวงจรพื้นฐานของอุปกรณ์พีแอลดีซึ่งอยู่ในรูปผลคูณร่วมบวก

3.1.2 เอฟพีจีเอ (FPGA: Field Programmable Gate Array)

เป็นอุปกรณ์ที่ถูกพัฒนาต่อจากอุปกรณ์เอลซีเอของบริษัทไซลิงซ์ (XILINX Inc.)โดยมีประสิทธิภาพการทำงานและมีปริมาณความหนาแน่นของเกตสูง สามารถจะกำหนดฟังก์ชั่นการทำงานได้ความต้องการผู้ใช้โดยผ่านการโปรแกรมเอฟพีจีเอได้รวบรวมข้อดีทั้งหมดของการทำ

คัสตัมวีแอลเอสไอ(Custom VLSI) มารวมไว้ทั้งหมาได้แก่ การออกแบบการผลิต, ลดเวลาที่จะส่งตัวผลิตภัณฑ์ออกตลาด ซึ่งเป็นประโยชน์ต่อการผลิตวงจรเป็นอย่างมาก นักออกแบบเพียงกำหนดฟังก์ชั่นการทำงานของวงจร ดังนั้นการออกแบบโดยใช้ เกตอาเรย์ จึงใช้เวลาหลายสัปดาห์ การเปลี่ยนแปลงแก้ไขแบบก็เช่นเดียวกันจากประโยชน์ของเอฟพีจีเอ ดังกล่าวมา ทำให้เกิดการประหยัดค่าใช้จ่ายเป็นอย่างมาก เพราะได้ความเสี่ยงในการที่จะต้องแก้ไขตัววงจร การเลื่อนเวลาการออกผลิตภัณฑ์ ลดค่าเอ็นอาร์อี (NRE: Nonrecurring Engineering Cost) ลงไปด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

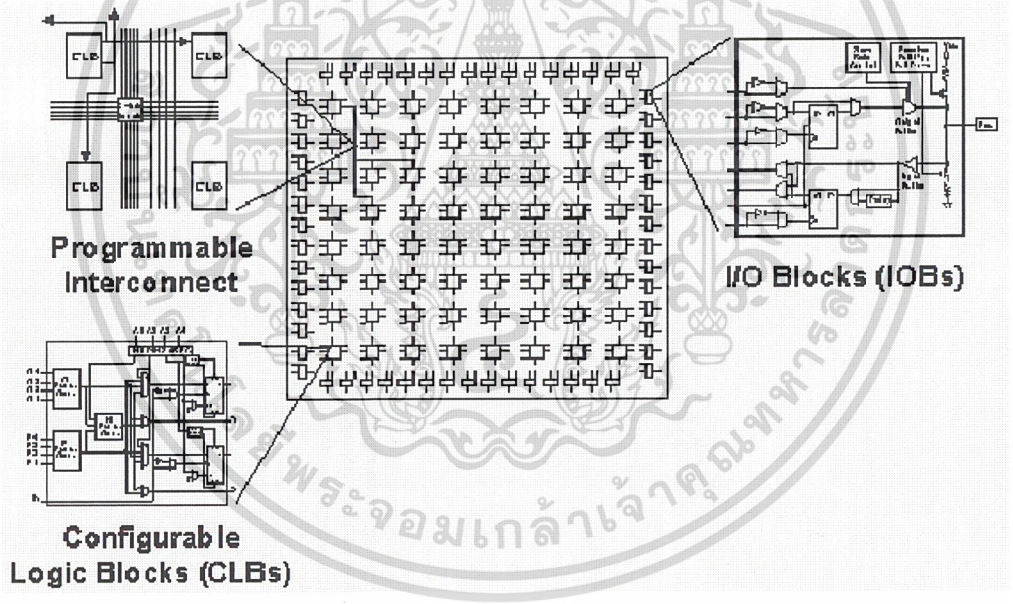
3.2 สถาปัตยกรรมภายในของเอฟพีจีเอตระกูล XC4000

FPGA (Field Programmable Gate Array) เป็นชิพที่มีสนามวงจรขนาดใหญ่อยู่ภายในที่เราสามารพนำมาใช้และออกแบบวงจรต่างๆ ได้ตามที่เราต้องการ โดยมีโปรแกรมสำเร็จรูปที่ใช้ในการออกแบบอยู่แล้ว มีความเร็วสูง มีสถาปัตยกรรมการออกแบบคล้าย CPLD (Complex Programmable Logic Device) แต่มีส่วนประกอบที่ซับซ้อน และมีประสิทธิภาพมากกว่า ในการออกแบบวงจรสามารถทำได้ง่าย ทั้งในการเชื่อมต่อและการแก้ไข ดังนั้น FPGA จึงเหมาะสมสำหรับการออกแบบวงจรเป็นอย่างมาก สถาปัตยกรรรมภายในของ FPGA แบ่งเป็น 3 ส่วนคือ

3.2.1 CLB (Configuration Logic Block)

3.2.2 IOB (Input Output Block)

3.2.3 Interconnect



รูปที่ 3.3 แสดงวงจรพื้นฐานภายในXilinx เอฟพีจีเอ

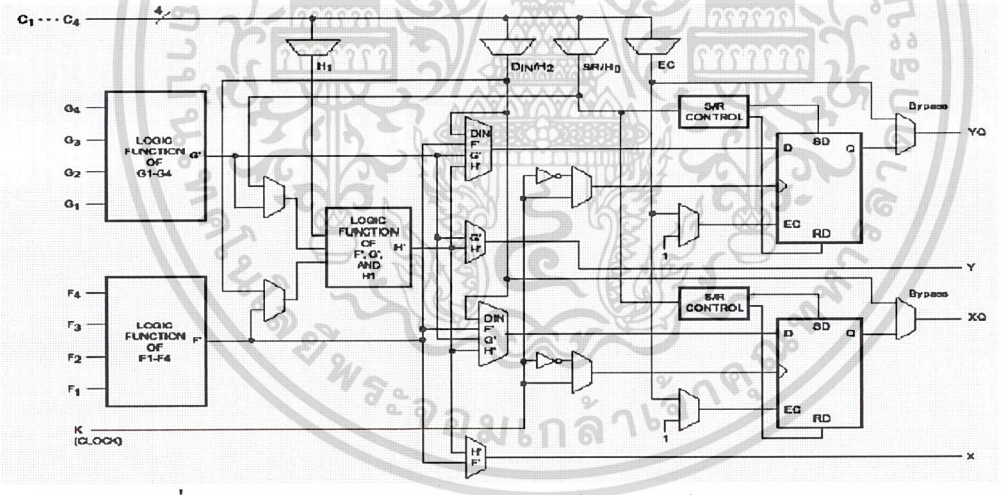
ซึ่งภายในมีสถาปัตยกรรมของ FPGA จะมีลักษณะเป็นตารางของลอจิกบล็อก (Logic block) และล้อมรอบไปด้วยบล็อกการเชื่อมต่อของไอโอ (I/O Interface block) การเชื่อมต่อระหว่างซีแอลบี (CLB: Configuration Logic Block) และไอโอบี (IOB: Input Output Block) ทำได้โดยผ่านช่องที่ว่างพาดผ่านระหว่างแถว(Row) และคอลัมน์(Column)มีการทำงาน เหมือนกันไมโครโพรเซสเซอร์ตัวแอลซีเอจะทำงานได้ต้องใช้ Program – driven logic device หน้าที่ของซีแอลบีและไอโอแต่ละตัว การเชื่อมต่อภายใน (Interconnection) ถูกกำหนดไว้ในโปรแกรมคอนฟิกคูเรชั่น (Configuration

เอกสารนี้มีการป้องกันลิขสิทธิ์ ไม่อนุญาตให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้ ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

program) หรือเก็บไว้ในอีพรอม (EPROM) ภายในแอลซีเอ (LCA: Logic Cell Array)โปรแกรมจะถูกโหลดเข้าสู่แอลซีเอเมื่อมีการจ่ายไฟ (Power-up) โดยทางคำสั่ง (Command) ซึ่งเป็นส่วนหนึ่งของการเริ่มต้นระบบ (System initialization) ประสิทธิภาพของแอลซีเอกำหนดโดย ความเร็วของลอจิกส่วนประกอบหน่วยความจำและการโปรแกรมการเชื่อมต่อต่างๆ ความเร็วของอัตราของระบบสัญญาณนาฬิกา (System clock rate) ถูกกำหนดด้วย ทอกเกลฟลิปฟลอฟ สำหรับการประยุกต์ใช้โดยทั่วไปจะอยู่ที่ประมาณ 1/3 ถึง1/2 ค่าสูงสุดของทอกเกิลเกต (Maximum toggle gate)

3.2.1 CLB (Configuration Logic Block)

ภายใน LCA (Logic Cell Array) คือเมทริกซ์ของซีแอลบีแต่ละตัวประกอบด้วยหน่วยของคอมเบนิชั่นลอจิกที่สามารถโปรแกรมได้ (Programmable combination logic) ส่วนของเรจิสเตอร์เก็บข้อมูล (Storage register) ส่วนของวงจรคอมเบนิชั่นลอจิกสามารถใช้สร้างวงจรทางด้านฟังกชั่นบูลีนของอินพุต ส่วนเรจิสเตอร์รับค่าจากส่วนคอมบิเนชั่นหรือโดยตรงจากเอาต์พุตของซีแอลบีสามารถขับวงจรคอมบิเนชั่นลอจิกโดยตรงผ่านเส้นทางเดินย้อนกลับ (Feedback path)

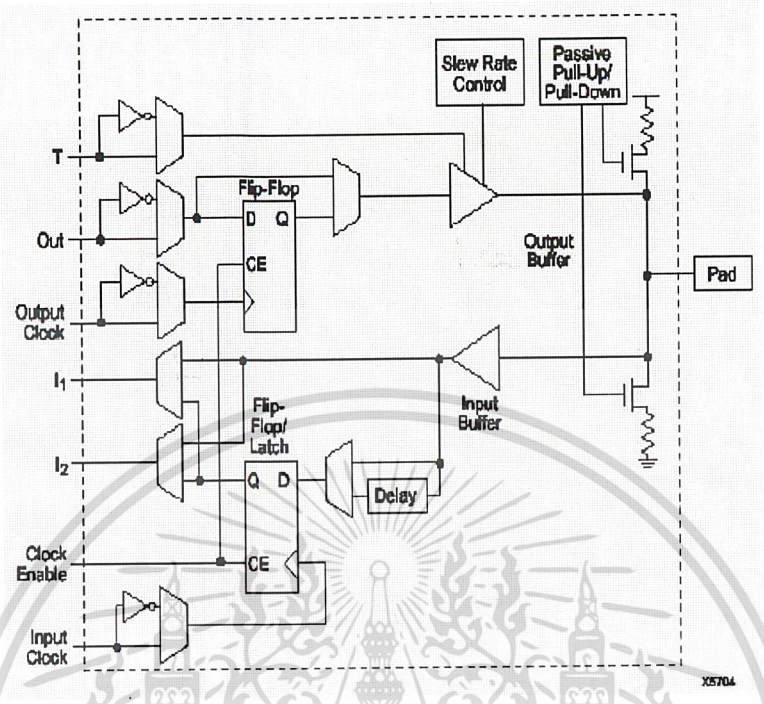


รูปที่3.4 แสดงผังวงจรภายในของซีแอลบีของเอฟพีจเอตระกูล XC4000

3.2.2 IOB (Input Output Block)

เป็นส่วนติดต่อกับวงจรภายนอกของแอลซีเอสร้างมาจากส่วนของอุปกรณ์อินพุต/เอาต์พุตที่สามารถโปรแกรมได้ (Programmable Input/Output device) แต่ละตัวสามารถโปรแกรมได้อย่างอิสระโดยจะให้เป็นอินพุต/เอาต์พุตแบบ 3 สถานะหรือไอโอแบบสองทิศทางก็ได้ โดยอินพุตสามารถโปรแกรมให้รู้จักทั้ระดับสัญญาณทีทีแอลและซีมอสเทรดโฮลของไอโอบี แต่ละตัวมีฟลิปฟลอปสามารถใช้เป็นบัฟเฟอร์สำหรับอินพุตและเอาต์พุต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



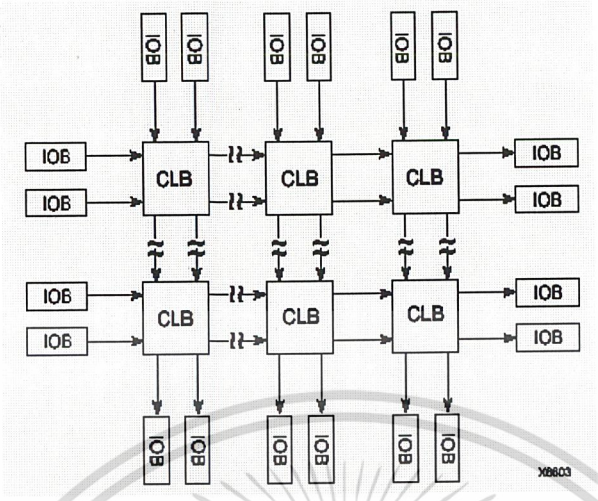
รูปที่3.5 แสดงผังวงจรของไอโอบีภายในเอฟพีจีเอตระกูล XC4000

3.2.3 Interconnect

ความยืดหยุ่นของการใช้แอลซีเอมาทำเป็นอุปกรณ์ขึ้นอยู่กับการโปรแกรม ทรัพยากรต่างๆ ที่อยู่ภายในเข้าด้วยกันกับการที่จะควบคุมการเชื่อมต่อระหว่างจุดสองจุดภายในชิป เหมือนกับ

เกตอาเรย์ทั่วๆ ไป การเชื่อมตอภายในเอลซีเอประกอบด้วยเน็ตเวิก 2 ทิศทางคือ ทางแถวและคอลัมน์ซึ่งวางอยู่ระหว่าง CLB programmable switch จะทำการเชื่อมต่ออินพุตและเอาต์พุตของไอโอบีและซีแอลบีที่จุดต่อร่วมระหว่างแถวกับคอลัมน์สามารถสลับสัญญาณจาก เส้นทางไปยังส่วนต่าง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.6 แสดง Interconnect ระหว่างไอโอบีกับซีแอลบีของเอฟพีจีเอตระกูล XC4000

3.3 คุณสมบัติโดยทั่วไปของ เอฟพีจีเอ ตระกูล XC4000

3.3.1 เป็นอุปกรณ์รุ่นที่3 ของเอฟพีจีเอ

- มีฟลิปฟลอปเป็นจำนวนมาก
- ในการผลิตฟังก์ชั่นของการทำงานมีความยืดหยุ่นสูง
- มีจำนวนเกตภายในจำนวน 2,000-10,000 เกต
- เพิ่มความสามารถพิเศษของเรจิสเตอร์และอินพุต/เอาต์พุต
- มีค่าแฟนเอาต์ (fan-out) สูง
- มีบัสภายใน 3 สถานะ
- ทำงานกับสัญญาณทีทีแอลและซีมอส
- มีแรมภายในความเร็วสูง (< 25 Ns)
- ใช้กับงานที่ต้องการความเร็วสูง (ใช้งานได้ที่ความถี่ 70/100/125 MHz)
- มี Wide edge decoder
- เส้นทางการเชื่อมต่อ (Interconnect line) เป็นแบบลำดับชั้น
- มีการกระจาย กำลังงานของสัญญาณต่ำ

3.3.2 มีสถาปัตยกรรมภายในที่ยืดหยุ่น

- มีลอจิกบล็อกและไอโอบล็อคที่สามารถโปรแกรมได้
- มีอินเตอร์คอนเน็คและ Wide decoder ที่โปรแกรมได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.3.3 คุณลักษณะทาง System-Oriented

- รองรับ มาตราฐาน IEEE 1149.1 ในการทำ boundary-scan logic
- สามารถโปรแกรมค่า output slew rate ได้
- สามารถโปรแกรมให้อินพุตมีลักษณะพูลอัพ (Pull-up) หรือ พูลดาวน์ (Pull-down) รีจิเตอร์ได้
- ให้กระแส เอาต์พุตได้ตั้งแต่ 12-24 มิลลิแอมป์ (ขึ้นอยู่แต่ละรุ่น)
- ทำการโหลดเอาแฟ้มข้อมูลประเภทไบนารี
- ไม่จำกัดจำนวนครั้งในการโปรแกรมซ้ำ
- มีโหมดในการโปรแกรมให้เลือก 6 โหมด

3.4 การโปรแกรมเอฟพีจีเอตระกูล XC4000

คือกระบวนการในการโหลดข้อมูลในโปรแกรมไปยังแอลซีเอ เพื่อกำหนดหน้าที่ของการทำงานในแต่ละบล็อกภายในการเชื่อมต่อ ซึ่งเอฟพีจีเอตระกูล XC4000 จะต้องใช้ข้อมูลเกี่ยวกับการโปรแกรมประมาณ 350 บิตต่อซีแอลบี โดยแต่ละบิตจะบอกถึงสถานะของหน่วยความจำสเตติกที่ควบคุมบิตในการควบคุมตารางฟังก์ชั่น (function table bit) และ มัลติเพล็กซ์อินพุตหรือการเชื่อต่อกันระหว่างทรานซิสเตอร์

3.4.1 โหมดการโปรแกรม

เอฟพีจีเอในตระกูล XC4000 มีโหมดการโปรแกรม 6 โหมด ซึ่งแต่ละโหมดจะถูกกำหนดโดยจากบิตโหมด ได้แก่บิต M0, M1 และ M2 โดยมีโหมดการโปรแกรมดังรูปที่ 3.6

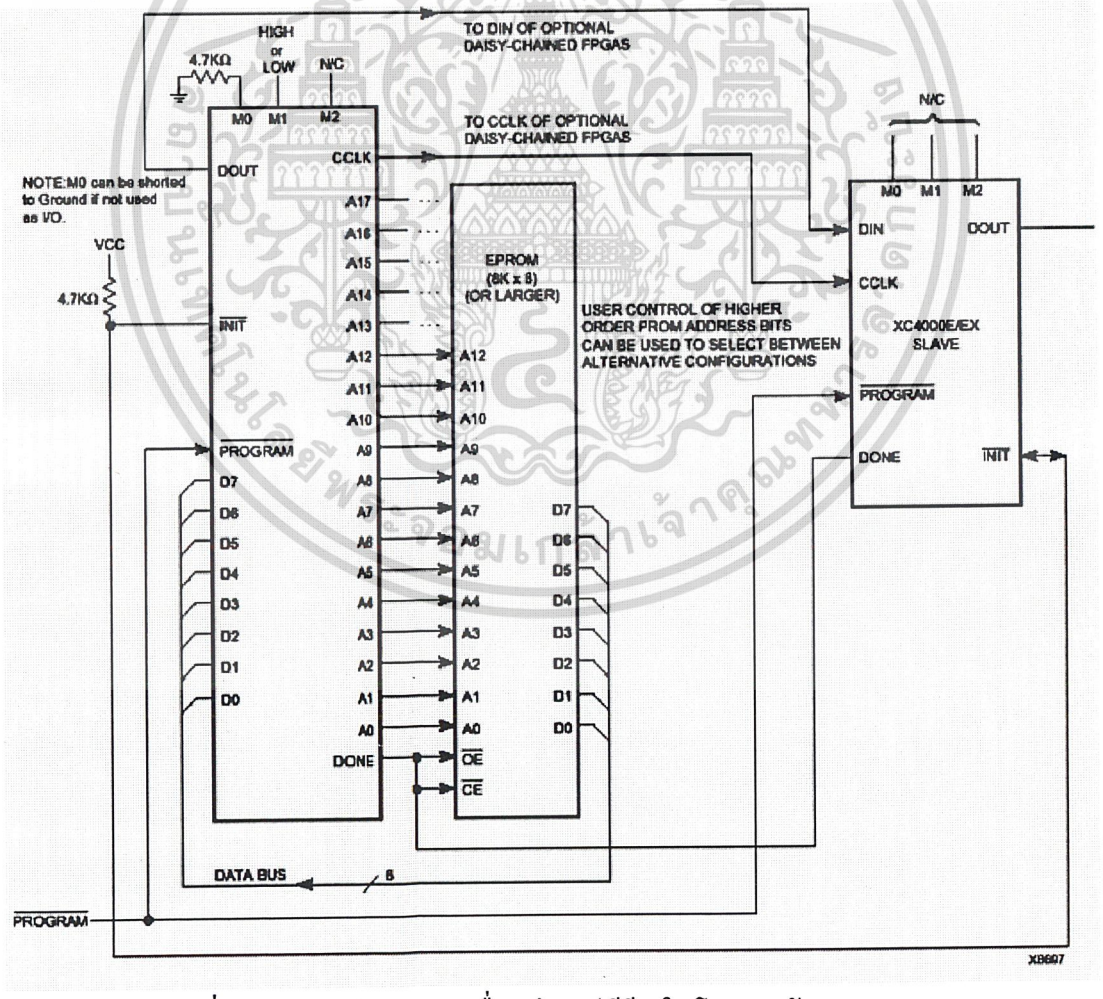
Mode	M2	M1	M0	Clock	Data
Master Serial	0	0	0	Output	Bit-serial
Slave Serial	1	1	1	Input	Bit-serial
Master parallel up	1	0	0	Output	Byte-Wide,00000
Master parallel down	1	1	0	Output	Byte-Wide,3FFFF
Peripheral Synch.	0	1	1	Input	Byte-Wide
Peripheral Asynch.	1	0	1	Output	Byte-Wide

รูปที่ 3.6 รูปแบบของโหมดต่างๆ ในการโปรแกรมเอฟดีจีเอตระกูล XC4000

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.4.1.1โหมดหลัก (Master modes)

ในการทำงานในโหมดนี้ตัวแอลซีเอจะถูกโหลดข้อมูลโครงแบบ (Configuration data) จากหน่วยความจำภายนอกเข้ามาโดยอัตโนมัติ มีโหมดที่แตดต่างกัน 8 โหมดโดยใช้ช่วงเวลาภายในจ่าย ให้ซีคล็อก (CCLK: Configuration clock) เพื่อที่จะเป็นฐานเวลาในการนำข้อมูลที่เข้ามาทางโหมดหลักแบบอนุกรม (serial master mode) และรับข้อมูลโครงแบบเข้ามาทางขาสัญญาณดีอิน (DIN: Data in) จากแหล่งสัญญาณซิงโครนัสเช่น Xilinx serial configuration PROM, parallel master low and master high mode โดยจะรับข้อมูลแบบขนานมาจากบิต D0-D7 โดยสัมพันธ์กับแอดเดรสที่กำหนดโดยแอลซีเอดังรูปที่ 3.4จะเป็นการแสดงตัวอย่างการเชื่อมต่อในโหมดหลักแบบขนาน (Master Parallel mode) โดยเริ่มต้นที่แอดเดรส 0000 ไบต์ข้อมูล (data byte) จะถูกอ่านเข้ามาแบบขนานทุก ๆ สัญญาณอาร์คล็อก (RCLK: Read Clock) และส่งเข้าไปภายในแบบอนุกรมโดยอาศัยสัญญาณนาฬิกาโครงแบบ (Configuration clock)

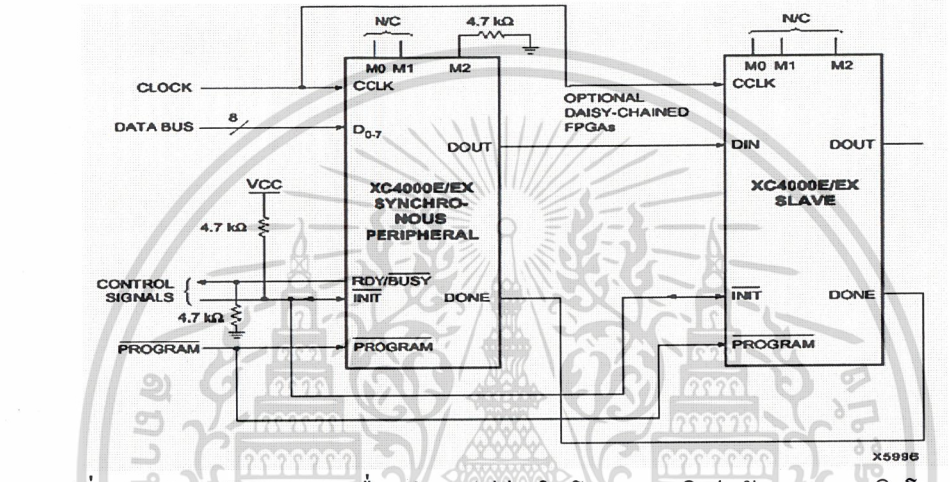


รูปที่ 3.7 แสดงผังวงจรการเชื่อมต่อเอฟพีจีเอในโหมดหลักแบบขนาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.4.1.2 เพอริเฟอรัลโหมด (Peripheral modes)

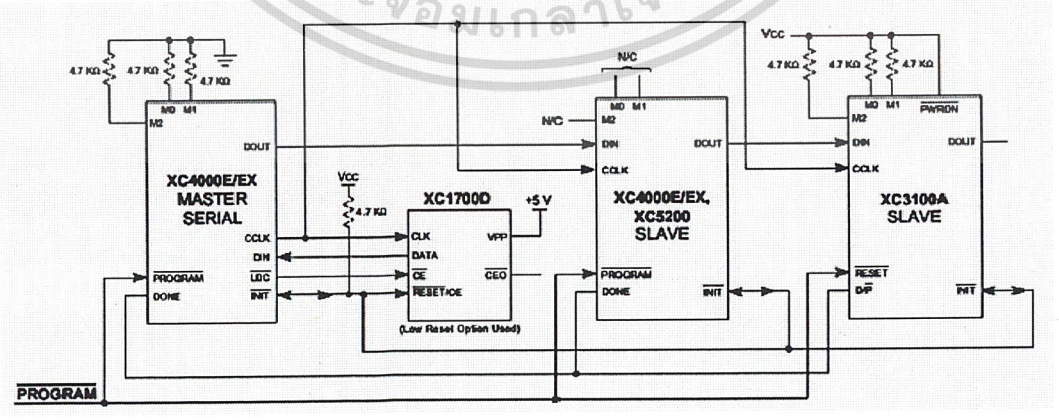
โหมดนี้จะรับข้อมูลเป็นไบต์จากบัส สถานะพร้อม (Ready) /กำลังทำงาน (Busy) จะมีอยู่เพื่อทำให้สัญญาณต่างๆสื่อสารกันได้มีประสิทธิภาพมากขึ้นในอะซิงโครนัสโหมดนั้นออสซิเลเตอร์ภายในจะสร้างสัญญาณ CCLK เพื่อรับข้อมูลที่เป็นไบต์ให้อยู่ในรูปแบบที่เหมาะสม และในซิงโครนัสโหมดนั้นสัญญาณนาฬิกาภายนอกจะเป็นตัวควบคุมการรับข้อมูลที่เป็นไบต์



รูปที่ 3.8 แสดงผังวงจรการเชื่อมต่อเอฟพีจีเอในโหมดเพอริเฟอรัล แบบ อะซิงโครนัส

3.4.1.3 โหมดรองแบบอนุกรม (Slave Serial modes)

ในโหมดนี้แอลซีเอจะรับข้อมูลโครงแบบที่เป็นอนุกรมในขาขึ้นของสัญญาณ CCLK และหลังจากรับข้อมูลมาแล้วจะส่งข้อมูลเพิ่มเติมออกไปด้วยและแอลซีเอก็จะถูกทำการซิงโครไนซ์ในขาลงถัดไปของสัญญาณ CCLK



รูปที่ 3.9 แสดงผังวงจรการเชื่อมต่อเอฟพีจีเอในโหมดรองแบบอนุกรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.5 การใช้ความสามารถของแรมในเอฟพีจีเอตระกูล XC4000

แอลซีเอทำงานโดยใช้ตารางการค้นหา (Look-up table) ซึ่งจะทำการเก็บตารางที่ว่านี้ในสแตติกแรม ซึ่งจะถูกเขียนในระหว่างการโปรแกรมโครงแบบลงบนแอลซีเอและจะถูกอ่านในการโอเปอ-เรชั่น ดังนั้นแรมภายในจึงควรถูกรวมไว้ในการออกแบบของผู้ใช้ด้วยหน้าที่ของแรมในเอฟพีจีเอตระกูล XC4000 มีหน้าที่คล้ายแรมโดยทั่วๆ ไป เช่น เอฟไอเอฟโอ (FIFO: First in First Out) แอลไอเอฟโอ (LIFO: Last in First Out) เรจิสเตอร์ไฟล์รวมทั้งแอปพลิเคชั่น อย่างเช่น รีจิสเตอร์เลื่อนข้อมูล (Shift register) แรมของเอฟพจีเอตระกูล XC4000 มีความเร็วสูงเสมือนเอสแรม (SRAM) จึงไม่จำเป็นต้องคำนึงถึงเวลาหน่วงของการเชื่อมต่อ (Interconnection delay)

3.6 การออกแบบวงจรดิจิตอลเพื่อใช้งานกับอุปกรณ์ FPGA

3.6.1วิธีการออกแบบวงจรดิจิตอล

ตามปกติแล้ว การออกแบบวงจรดิจิตอลเพื่อใช้งานร่วมกับอุปกรณ์ประเภทเอฟพีจีเอ (FPGA : Field Programmable Gate Arrays) นั้นจะมีขั้นตอนที่ไม่แตกต่างจากการออกแบบวงจรดิจิตอลประเภทเอสิก (ASIC – Application Specific integrated Circuits) มากนัก โดยเฉพาะอย่างยิ่ง การใช้ภาษาธิบายฮาร์ดแวร์ในการออกแบบวงจร หรือที่เรียกกันว่า เอชดีแอล (HDL – Hardware Description Language) เช่น ภาษา VHDL หรือ Verilog เป็นต้น ซึ่งสามารถเขียนให้อยู่ในรูปของโค้ดภาษาอธิบายฮาร์ดแวร์ แล้วนำไปผ่านการสังเคราะห์วงจร (HDL Synthesis) โดยที่เราสามารถเลือกเทคโนโลยีเป้าหมาย (Target technology) ได้ตามต้องการ เช่น เลือกเทคโนโลยีเป้าหมายเป็น Standard Cell ASIC หรือ FPGA เป็นต้น แต่จะต้องเลือกใช้ไลบรารีสำหรับการสังเคราะห์อย่างเมาะสมด้วย เนื่องจากปัญหาสำคัญส่วนหนึ่งของการออกแบบอยู่ตรงที่การกำหนดเทคโนโลยีเป้าหมาย ดังนั้นอาจจะมีบางจุดในโค้ดที่เราได้เขียนขึ้นต้องได้รับการเปลี่ยนแปลงแก้ไขก่อนเพื่อเหมาะสมและถูกต้องเมื่อต้องการจะเปลี่ยนจากเทคโนโลยีเป้าหมายหนึ่งไปยังเทคโนโลยีอื่น ๆ แต่ก็ทำได้ไม่ยาก ซึ่งไม่เหมือนกับวิธการออกแบบที่ใช้สคิมเมติก (Schematic) เพราะสำหรับวิธีหลังนี้ ดีไซน์จะถูกเขียนอธิบายให้อยู่ในรูปของสัญลักษณ์ทางกราฟิกที่ประกอบขึ้นจากองค์ประกอบพื้นฐาน (Primitives) และมีอยู่ในไลบรารีสำหรับเทคโนโลยีเป้าหมาย (Target Library) แล้วนำมาต่อเข้าด้วยกันเป็นบล็อกย่อยหรือโมดูล และสามารถนำมาประกอบเข้าเป็นวงจรที่ซับซ้อนอีกได้หลายระดับขึ้นไปจนกลายเป็นวงจรที่เสร็จสมบรูณ์ ซึ่งการใช้ซีมเมติกนี้เป็นการอธิบายวงจรในเชิงโครงสร้าง (Structural Description) นั้นเอง จุดด้อยของวิธีการออกแบบโดยใช้ซีมเมติกนี้ก็คือ การเปลี่ยนแปลงแก้ไขดีไซน์นั้นทำได้ค่อนข่างยากลำบากและใช้เวลามาก และอีก

เอกสารนี้สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้ทำซ้ำ ดัดแปลง เผยแพร่ หรือนำไปใช้ประโยชน์ทางการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ประการหนึ่งที่สำคัญคือ ดีไซน์ที่ถูกสร้างขึ้นนั้นจะขึ้นอยู่กับไลบรารี่เป้าหมายที่ได้เลือกใช้ ถ้าจะเปลี่ยนไลบรารี่ก็ย่อมทำได้ลำบากโดยเฉพาะอย่างยิ่งวงจรที่มีขนาดใหญ่ เพราะจะต้องมาแก้ไขซีมเมติกใหม่ ดังนั้นจะขอกล่าวถึงการออกแบบโดยใช้ภาษาอธิบายฮาร์ดแวร์เท่านั้น

ไม่ว่าจะเป็นการออกแบบโดยใช้ซีมเมติก หรือ เอชดีแอลร่วมกับการสังเคราะห์ก็ตาม แต่ผลที่ได้ก็คือ เน็ตลิสต์ในระดับเกต (Gate-level Netlist) ซึ่งมีฟอร์แมทมาตราฐานที่นิยมใช้คือ EDIF (Electronic Data Interchange Format) และ XNF (Xilinx Netlist Format) สำหรับการออกแบบวงจรให้ใช้งานได้กับอุปกรณ์เอฟพีจีเอของ Xilinxนั้น เราจะใช้ซอฟต์แวร์ที่มีชื่อว่า Xilinx Foundation M Series หรือ Xilinx Alliance M Series ซอฟต์แวร์ตัวนี้จะอ่านเน็ตลิสต์ (EDIF หรือ XNF) ของดีไซน์ที่ได้รับการออกแบบเป็นที่เรียบร้อยแล้ว จากนั้นจึงจะเปลี่ยนให้อยู่ในรูปแบบที่เหมาะสมตามสถาปัตยกรรมภายในของอุปกรณ์ ประกอบด้วยลอจิกบล็อกย่อย (Logic Blocks) ที่สามารถวางลงในเอฟพีจีเอได้ และเมื่อผ่านขั้นตอนการวาง (Placement) บล็อกย่อยเหล่านั้นอย่างมีประสิทธิภาพแล้ว ก็จะทำการเชื่อมโยงบล็อกเหล่านั้น (Routing) เข้าด้วยกัน และท้ายสุดก็คือการเขียนข้อมูลเกี่ยวกับ Device Configuration ให้อยู่ในรูปของบิตสตรีม (Bitstream) เพื่อนำไปใช้โปรแกรมอุปกรณ์ (Device Programming) ให้ทำงานตามแผนวงจรที่ได้ออกแบบไป

ขั้นตอนต่างๆ ที่ได้กล่าวไปเป็นเพียงขั้นตอนการออกแบบอย่างคร่าวๆ เท่านั้น ซึ่งตามความเป็นจริงแล้ว จะมีขั้นตอนย่อยอีกหลายขั้นตอนที่จะกล่าวถึงต่อไป

3.6.2 ขั้นตอนการออกแบบวงจรโดยใช้ภาษา HDL

ดังที่ได้กล่าวไปแล้ว ในปัจจุบันแนวโน้มการออกแบบวงจรดิจิตอบขนาดใหญ่จะหันเหไปทางการใช้งานภาษา อธิบายฮาร์ดแวร์ ซึ่งมีบทบาทมากขึ้นตามลำดับด้วยเหตุผลหลายประการ อาทิเช่น ความสามารถในการใช้อธิบายการทำงานของวงจรที่มีขนาดใหญ่มาก ความสามารถในการนำกลับไปใช้งานได้อีกของดีไซน์ที่ถูกสร้างขึ้น ความสามารถในการนำไปใช้งานได้กับหลายๆ เทคโนโลยีเป้าหมาย โดยโค้ดที่ถูกเขียนจะไม่ขึ้นอยู่กับเทคโนโลยีเหล่านั้น หรือเป็นเพียงส่วนน้อยเท่านั้น นอกจากนั้นยังมีซอฟต์แวร์ที่มีประสิทธิภาพอีกมากมายสำหรับใช้ในการสังเคราะห์วงจ ในปัจจุบันภาษาอธิบายฮาร์ดแวร์ที่นิยมใช้และเป็นมาตราฐานในทางอุตสาหกรรมด้วยคือ ภาษา VHDL และ Verilog HDL แม้ว่าจะมีซอฟต์แวร์หลายๆ ตัวที่สนับสนุนทั้งสองภาษาและภาษาทั้งสองก็มีจุดเด่นจุดด้อยและประสิทธิภาพในการใช้งานไม่แตกต่างกันมากนัก แต่ที่จะกล่าวถึงในที่นี้จะเป็นภาษา VHDL เท่านั้น ซึ่งรวมไปถึงดีไซน์ตัวอย่างทั้งหมดด้วย ดังนั้นผู้อ่านจะต้องมีความรู้ความเข้าใจภาษา VHDL นี้ด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.6.2.1 การออกแบบวงจรให้อยู่ในรูปของโค้ดภาษา HDL

การออกแบบวงจรโดยใช้ภาษา VHDL (เช่นเดียวกับ Verilog) จะเริ่มต้นที่การเขียนโค้ดหลังจากที่ได้มีการออกแบบสถาปัตยกรรมของวงจรอย่างดีแล้ว เช่น อาจจะเริ่มจากการแบ่งวงจรรวมออกเป็นวงจรย่อย เพื่อให้ง่ายต่อการออกแบบตามหลักการ Top-down Design Methodology ถ้าวงจรหรือบล็อกย่อยไม่ซับซ้อนจนเกินไปและง่ายต่อการสังเคราะห์วงจรในภายหลัง ก็สามารถเขียนอธิบายพฤติกรรมการทำงาน ของวงจรเหล่านั้น (Behavioral Description) เป็นโมเดลในภาษา VHDL และเมื่อวงจรย่อยเหล่านั้นเสร็จแล้ว จึงนำนำมาประกอบเข้าด้วยกันโดยการเรียกใช้ส่วนประกอบ (Component Instantiation) และมีสายสัญญาณเชื่อมต่อ (Signals) อย่างเหมาะสม ดังนั้นการอธิบายวงจรในระดับนี้จึงเป็นการอธิบายในเชิงโครงสร้าง (Structural Description) การเขียนโค้ดภาษา VHDL เช่นเดียวกับภาษาสูงอื่นๆ สามารถใช้โปรแกรมจำพวก Text Editor ทั่วๆ ไปได้แต่อย่างไรก็ตาม ยังมีซอฟต์แวร์หลายตัวที่สนับสนุนการเขียนโค้ดภาษา VHDL ซึ่งทำให้ง่ายขึ้นเวลาเขียนโค้ดในภาษา VHDL รวมทั้งการคอมไพล์โค้ดเพื่อตรวจเช็คความถูกต้องตามหลักไวยากรณ์ด้วย ตัวอย่างของคอมไพล์เลอร์ภาษา VHDL สำหรับเครื่องพีซี ก็เช่น ซอฟต์แวร์ชื่อ Modelsim Model Technology,Inc. เป็นต้น

3.6.2.2 การตรวจสอบความถูกต้องของวงจรที่สร้างขึ้นในระดับพฤติกรรม

ขั้นตอนอีกขั้นตอนหนึ่งที่สำคัญในการออกแบบวงจร คือ การตรวจสอบว่า โมเดลในภาษา VHDL ที่เราได้สร้างขึ้นนั้นสามารถทำหน้าที่ตามที่ได้กำหนดไว้หรือไม่ ซึ่งสามารถทำได้โดยการทำซิมมูเลชั่นเพื่อดูพฤติกรรมการทำงานของโมเดลเหล่านั้น (Behavioral Simulation) และเพื่อจุดประสงค์นี้เอง จะต้องมีการเขียนโมเดลพิเศษสำหรับการทำซิมมูเลชั่นขึ้นอีก ที่เรียกว่า VHDL Testbench ซึ่งภายในจะมีการเรียกใช้โมเดลหรือส่วนประกอบที่ต้องการจะตรวจสอบพฤติกรรมการทำงาน และมีการกำหนดรูปแบบของสัญญาณขาเข้า (Input Signal) ที่เรียกว่า Stimulus และเมื่อทำซิมมูเลชั่นสัญญาณขาออก (Output Signal) ของโมเดล ก็จะปรากฏให้เห็นบนจอภาพ และสามารถนำไปเหรียบเทียบกับสัญญาณขาเข้าเพื่อตรวจดูว่า การทำงานของโมเดลหรือดีไซน์ที่ถูกตรวจสอบ (Design Under Test หรือ DUT) เป็นไปตามที่คาดหวังไว้หรือไม่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.6.3 การสังเคราะห์วงจรจากดีไซน์ในภาษา HDL

เมื่อได้ตรวจสอบเป็นที่แน่ใจแล้วว่าโมเดลต่างๆ ที่ได้สร้างขึ้นสามารถทำงานได้อย่างถูกต้อง ขั้นตอนต่อไปก็คือ การผ่านโค้ดของโมเดลเหล่านั้นไปทำการสังเคราะห์วงจรเพื่อให้ได้เน็ตลิสต์ในระดับเกตออกมา เครื่องมือที่ใช้ในการสังเคราะห์โค้ดภาษา VHDL (HDL Synthesis Tool) ที่ทาง TMEC ใช้อยู่ก็คือ ซอฟต์แวร์ของ Exemplar Logic lnc.ที่ชื่อLeonardoSpectrum ซึ่งสามารถใช้ในการสังเคราะห์วงจรให้เป็นเอสิกหรือเอฟพีจีเอก็ได้เพราะมีคลังไลบรารี่ให้เลือกใช้ได้จากหลายผู้ผลิต เนื่องจากเราจะสังเคราะห์วงจรสำหรับใช้งานกับเอฟพีจีเอของ Xilinx ดังนั้นไลบารี่ที่ใช้ก็จะเป็นของ Xilinx เช่น ไลบารี่ของตระกูล XC4000E เป็นต้น

3.6.4 การแปลงดีไซน์ที่สังเคราะห์แล้วเพื่อใช้กับอุปกรณ์ FPGA

ผลที่ได้จากการสังเคราะห์นั้นจะเป็นเน็ตลิสต์ในระดับเกต โดยสามารถนำไปผ่านขั้นตอนการแปลงให้เป็นลอจิกบล็อกและรวมถึงการวางและเชื่อมเส้นทาง (Place & Route) ภายในอุปกรณ์เอฟพีจีเอ โดยใช้ซอฟต์แวร์ Alliance Series Implementation Tool

3.6.5 การทำซิมมูเลชั่นหลังผ่านขั้นตอนการวางและเชื่อมเส้นทาง

การทำซิมมูเลชั่นในระดับพฤติกรรมหรือหลังจากการสังเคราะห์วงจรแต่ยังมิได้ผ่านขั้นตอนการวางและเชื่อมเส้นทางนั้นจะใช้ในการตรวจสอบความถูกต้องทางลอจิกของวงจรเท่านั้น แต่ยังไม่มีรายละเอียดที่แน่นอนเกี่ยวกับเรื่องของเวลาหรือไทม์มิ่ง (Timing) มากนัก เช่น ความล่าช้าของสัญญาณไฟฟ้าตามเส้นทางต่างๆ ภายในวงจร ระยะคาบของสัญญาณเวลา (Clock Period) น้อยที่สุดเท่าที่จะสามารถใช้กับวงจรได้ เป็นต้น เพราะจะต้องให้ผ่านขั้นตอนการวางและเชื่อมเส้นทางก่อน ดังนั้นการทำซิมมูเลชั่นเพื่อตรวจสอบการทำงานของวงจรในอุปกรณ์เอฟพีจีเอโดยคำนึงถึงเรื่องไทม์มิ่งจะกระทำเป็นขั้นตอนสุดท้าย

3.6.6 สิ่งที่ต้องคำนึงถึงเมื่อใช้ภาษา HDL สำหรับออกแบบวงจร

ในช่วงที่ผ่านมา การออกแบบวงจรดิจิตอลสำหรับอุปกรณ์แบบเอฟพีจีเอจะเป็นแบบผสม (Hybrid Design Approach) คือจะประกอบด้วยซีมเมติกที่ใช้สัญลักษณ์จากไลบรารี่ของผู้ผลิตเอฟพีจีเอเป็นส่วนใหญ่ รวมถึงการใช้โมดูลพิเศษที่อยู่ในรูปของแมคโคร (macro) เช่น XBLOX และ LPMของ Xilinx เป็นต้น และอาจจะมีการใช้ภาษาระดับสูงร่วมด้วย อาทิ เช่น VHDL, Verilog หรือ ABELแต่ในปัจจุบันแนวโน้มของการออกแบบจะเน้นไปทางการใช้ภาษาอธิบายฮาร์ดแวร์มากขึ้นเพราะความสะดวก และความสามารถของอุปกรณ์เอฟพีจีเอในยุคปัจจุบันได้เพิ่มมากขึ้นซึ่งสามารถนำไปใช้ในการออกแบบวงจรดิจิตอลที่มีความซับซ้อนในระดับสูงได้ ดังนั้นเพื่อให้เป็นไปในทิศทางเดียวกันกับการออกแบบวงจรเอสิกที่ปกติใช้สำหรับวงจรขนาดใหญ่มาก จึงส่งผลให้การใช้ภาษาอธิบายฮาร์ดแวร์ในการออกแบบวงจรสำหรับอุแกรณ์เอฟพีจีเอมีความสำคัญมากขึ้นตาม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ลำดับแต่อย่างไรก็ตาม การใช้ภาษาอธิบายฮาร์ดแวร์เพียงอย่างเดียว (Completely HDL Based Design Approach) ไม่ว่าจะเป็น VHDL หรือ Verilog ก็ตาม ยังมีข้อจำกัดในการนำไปใช้งานจริงอยู่บ้าง ยกตัวอย่างเช่น การเรียกใช้องค์ประกอบ (Component Instantiation) โดยตรงจากไลบรารีของผู้ผลิตอุปกรณ์เอฟพีจีเอซึ่งอาจจะเป็นองค์ประกอบพื้นฐานหรือองค์ประกอบรวม (คล้ายกับการใช้องค์ประกอบจากไลบรารี่ในรูปสัญลักษณ์ทางกราฟฟิกภายในซีมเมติก) เช่น โมเดลที่เป็นหน่วยความจำแบบRAMเป็นต้น ถ้าได้มีการเรียกใช้องค์ประกอบนี้ภายในโค้ดภาษาอธิบายฮาร์ดแวร์ องค์ประกอบเหล่านั้นก็จะเป็นเสมือนกล่องดำหรือ Block Box คือมีแต่โครงภายนอกที่มีกำหนดแค่พอร์ตต่างๆ ขององค์ประกอบหรือโมเดล แต่ไม่ทราบฟังก์ชันหรือโครงสร้างภายใน การทำเช่นนี้มีข้อดีคือ องค์ประกอบจากไลบรารี่เหล่านั้นได้ถูกออกแบบให้เหมาะสมกับเทโนโลยีที่ได้เลือกใช้แล้วโดยผู้ผลิต นักออกแบบวงจรไม่จำเป็นต้องคำนึงถึงเรื่องการที่จะต้องมาออกแบบองค์ประกอบเหล่านั้นเองและอย่างไรบ้าง ซึ่งเป็นการช่วยลดระยะเวลาในการออกแบบ แต่ในขณะเดียวกันวิธีการดังกล่าวก็มีข้อเสียดังนี้คือ ดีไซน์จะขึ้นกับเทคโนโลยีเป้าหมายมากขึ้นเพราะได้มีการเลือกใช้องค์ประกอบจากไลบรารี่สำหรับเทคโนโลยีตัวใดตัวหนึ่ง ละอีกประการหนึ่งก็คือ เมื่อต้องการจะซิมมูเลชั่นจะไม่สามารถทำซิมมูเลชั่นได้อย่างสมบรูณ์ถ้าไม่มีไลบรารี่ที่มีองค์ประกอบเหล่านั้นสำหรับการทำซิมมูเลชั่นโดยเฉพาะ จะมีก็เพียงแต่ไลบรารี่สำหรับการสังเคราะห์วงจรเท่านั้น ในกรณีดังกล่าวการทำซิมมูเลชั่นในระดับพฤติกรรมของวงจรจึงต้องข้ามไป ซึ่งเป็นการเสี่ยงอย่างยิ่งเพราะเราไม่อาจจะแน่ใจได้ว่าโมเดลที่ได้เขียบขึ้นจะทำงานได้ถูกต้องหรือไม่ และถ้าโมเดลไม่ถูกต้องตั้งแต่เริ่มแรกแล้วการสังเคราะห์และการทำขั้นตอนอื่นต่อๆ ไปก็ไม่มีประโยชน์

ต่อมาภายหลังผู้ผลิตเอฟพีจีเอซึ่งเป็นผู้สร้างไลบรารี่สำหรับเทคโนโลยีของตนเองนั้นได้เริ่มหันมาสนใจกับปัญหานี้มากขึ้นอย่างเช่นXilinxก็ได้สร้างไลบรารี่สำหรับองค์ประกอบพื้นฐาน (SIMPRIM)เพื่อให้สามารถทำซิมมูเลชั่นในระดับพฤติกรรมหรือระดับเกตได้อย่าง สมบรูณ์แบบ นอกจากนั้นก็ได้พยายามผลิตซอฟต์แวร์ตัวใหม่ออกมาเพื่อการออกแบบวงจรสำหรับอุปกรณ์ของตนเองโดยเฉาะ โดยคลอบคลุมแผนขั้นตอนการออกแบบเกือบทั้งหมด และนักออกแบบก็ไมจำเป็นต้องใช้ซอฟต์แวร์ต่างผู้ผลิตเหมือนอย่างในอดีตที่ผ่านมา ซึ่งเมื่อหลายปีก่อน Xilinx จะทำซอฟต์แวร์เฉพาะใช้สำหรับขั้นตอนในลำดับท้ายตามแผนขั้นตอนการออกแบบเท่านั้น

อย่างไรก็ตาม การใช้ภาษาอธิบายฮาร์แวร์ในการออกแบบวงจรดิจิตอลสำหรับอุปกรณ์เอฟพีจีเอจะส่งผลให้ดีไซน์ที่ถูกสร้างขึ้นไม่ขึ้นอยู่กับเทคโนโลยีเป้าหมายมากเกินไป แต่ในขณะเดียวกันนักออกแบบก็ต้องคำนึงถึงการใช้งานอุปกรณ์ (Device Utilization)ให้ได้ประสิทธิภาพมากที่สุด ดังนั้นในระหว่างการออกแบบวงจร ก็มีความจำเป็นต้องคำนึงถึงสถาปัตยกรรมภายในและเทคโนโลยี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่4

การออกแบบโครงงาน และการใช้ซอฟแวร์โปรแกรมลง เอฟพีจีเอ

บทนี้เป็นการบรรยายการออกแบบโครงงาน และใช้ ซอฟแวร์เพื่อโปรแกรมลงเอฟพีจีเอโดย

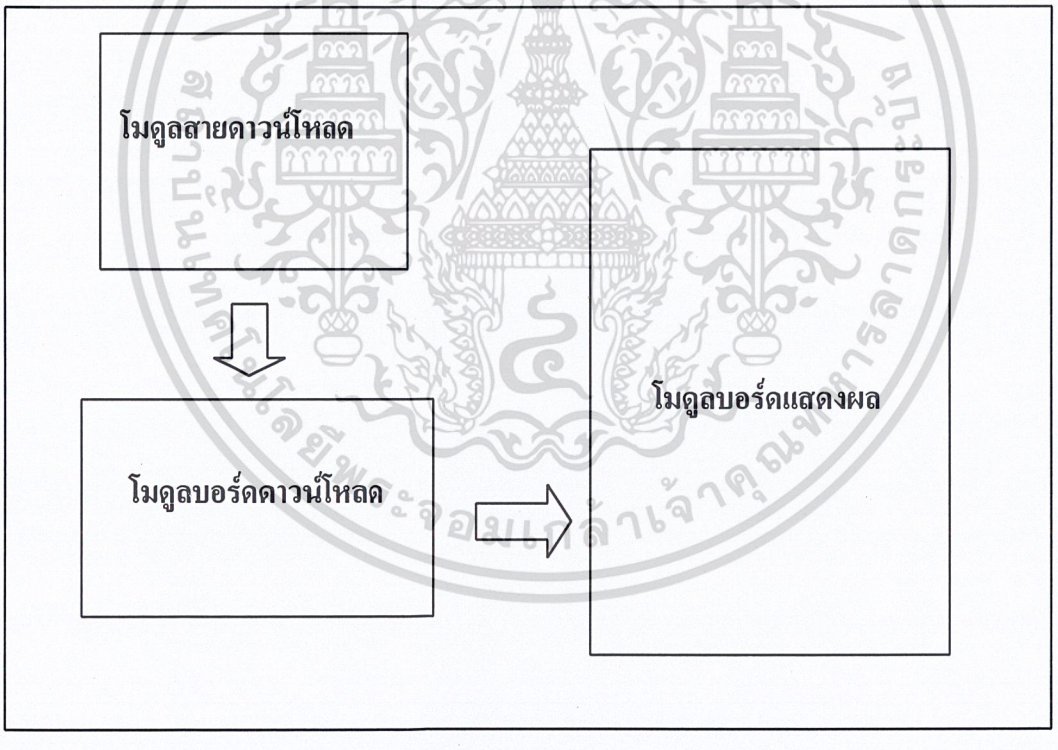
4.1 การออกแบบโครงงาน

การออกแบบงานเราได้แบ่งงานออกเป็น 3 ส่วนคือ

โมดูลสายดาวน์โหลด (Download Cable Module)

โมดูลบอร์ดดาวน์โหลด (Download Board Module)

โมดูลบอร์ดแสดงผล (Output Board Module)



รูปที่4.1 แสดงรูปจำลองของบอร์ดทดลอง

ลักษณะขั้นตอนการทำงาน

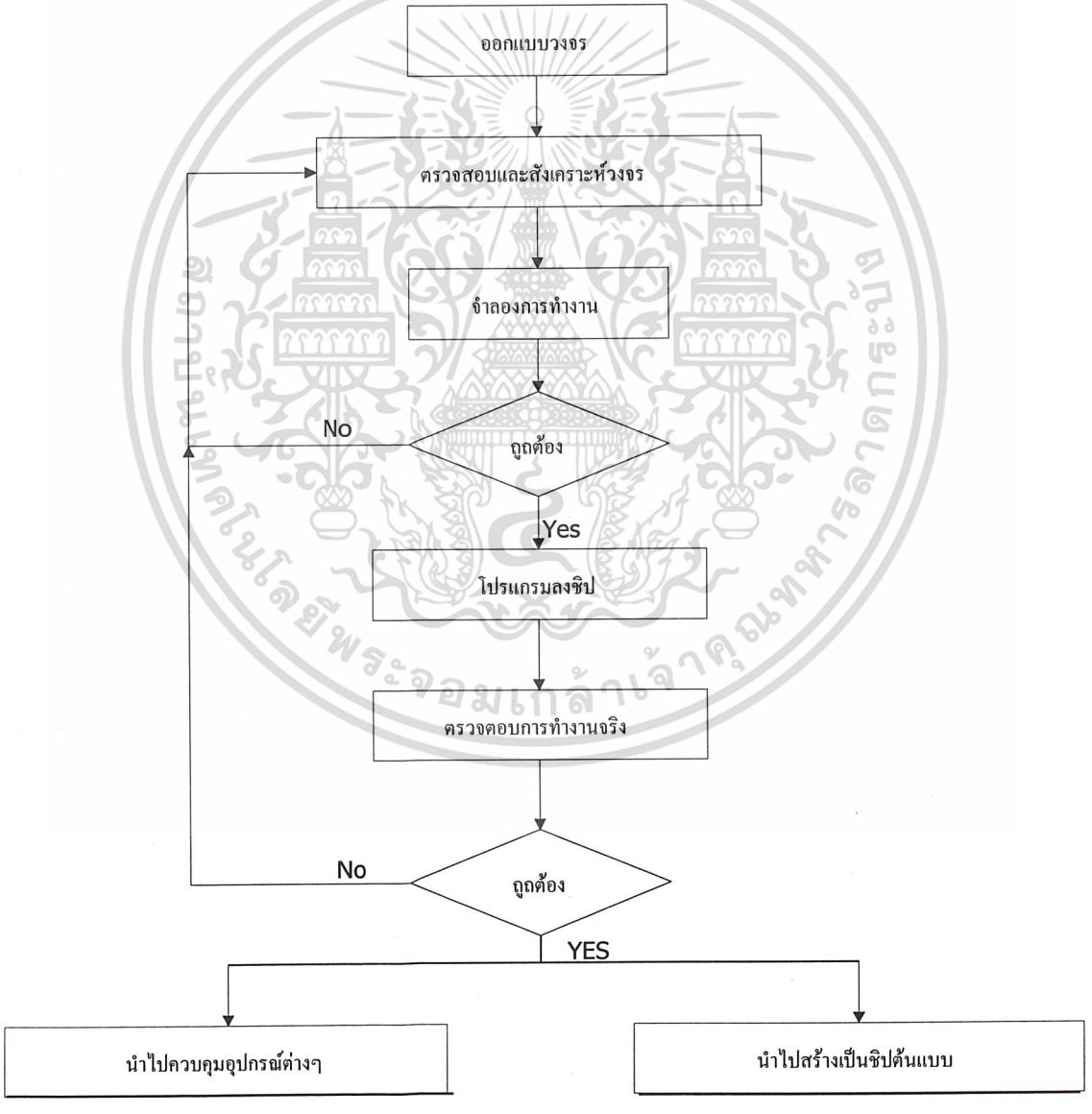
1. เขียนโปรแกรมโดยภาษาวีเอชดีแอล
2. ทำการคอมไพล์และซิมมูเลท เพื่อตรวจสอบความถูกต้องของไวยกรณ์ในภาษาวีเอชดีแอลที่เขียนขึ้นโดยใช้โปรแกรม ModelSim ตามวิธีที่เขียนไปแล้วข้างต้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3. เสร็จจากการคอมไพล์แล้วก็จะเป็นการสังเคราะห์ไฟล์ของภาษาวีเอชดีแอลเพื่อให้เป็นไฟล์เน็ท ลิสต์โดยการใช้โปรแกรม Leonadospectrum

พอสังเคราะห์ไฟล์แล้วเราจะใช้โปรแกรม Xilinx Foundation แปลงข้อมูลจากไฟล์เน็ทลิสต์เพื่อที่จะดาวน์โหลดลง เอฟพีจีเอ ด้วยสาย ดาวน์โหลดที่สร้างขึ้นต่อไปสู่บอร์ดดาวน์โหลดต่อไป

4. บอร์ดดาวน์โหลดจะรับค่าและเก็บลงสู่ เอฟพีจีเอ แล้วส่งผลไปเพื่อแสดง ซึ่งเราแสดงกระบวนการทั้งหมดออกมาเป็น โฟรชาดร์ดั้งนี้



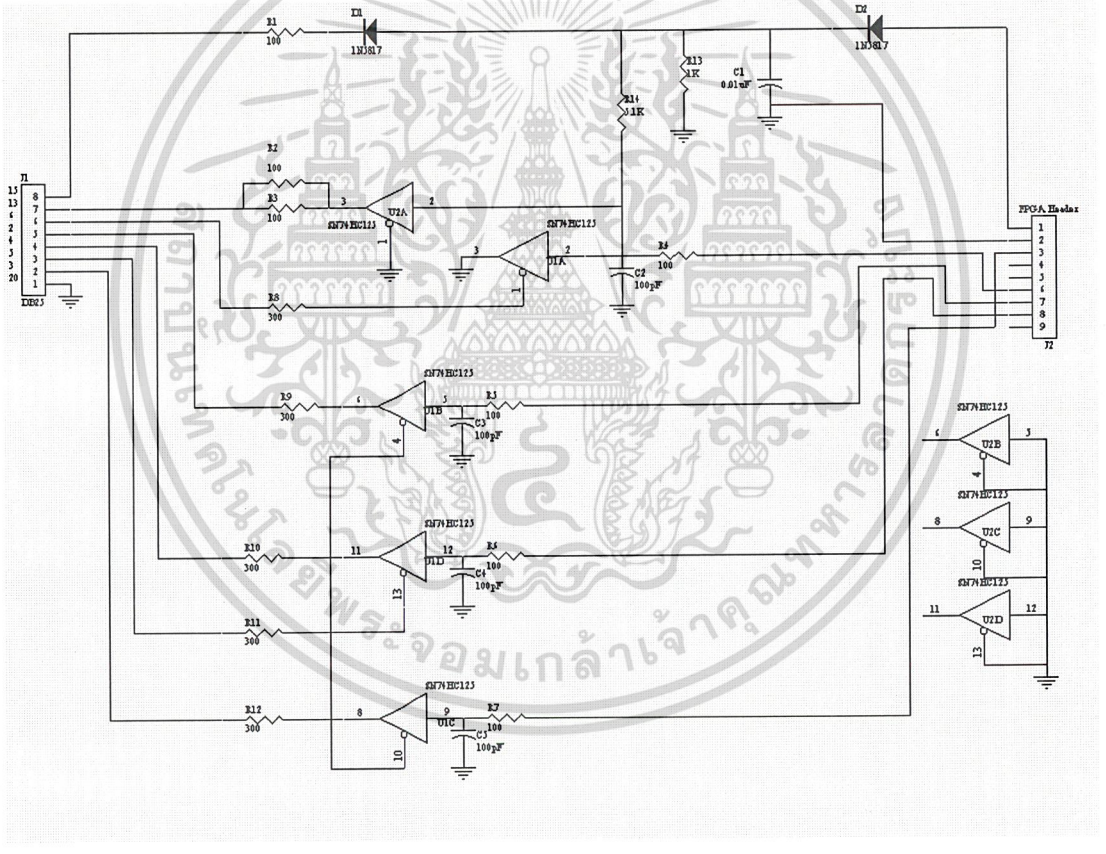
รูปที่ 4.2 แสดง flow chat การทำงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.1.1 โมดูลสายดาวน์โหลด (Download Cable Module)

เป็นโมดูลที่สำคัญอย่างมากส่วนหนึ่งเพราะถ้าขาดไป บอร์ดก็ไม่สามารถทำงานได้ และเป็นที่น่าภูมิใจมากที่โมดูลส่วนนี้ของเราได้สร้างขึ้น ด้วยต้นทุนที่ต่ำมากงบประมาณไม่เกิน 200 บาท ซึ่งราคาของสายดาวน์โหลดซึ่งเป็นชนิด X Checker ที่ขายในท้องตลาดจะมีราคาแพงมากจึงเป็นเรื่องยากที่จะหามาใช้ได้สำหรับผู้เริ่มต้น หน้าที่ของโมดูลนี้ จะเป็นตัวเชื่อมต่อกับ คอมพิวเตอร์เพื่อนำไฟล์ที่ออกแบบโดยภาษาวีเอชดีแอลและได้รับการสังเคราะห์เป็นระดับเกต lavel เพื่อมาดาวน์โหลดลง ชิฟ เอฟพีจีเอ ตามกระบวนการออกแบบดังนี้

ลายวงจร ของโมดูลสายดาวน์โหลด



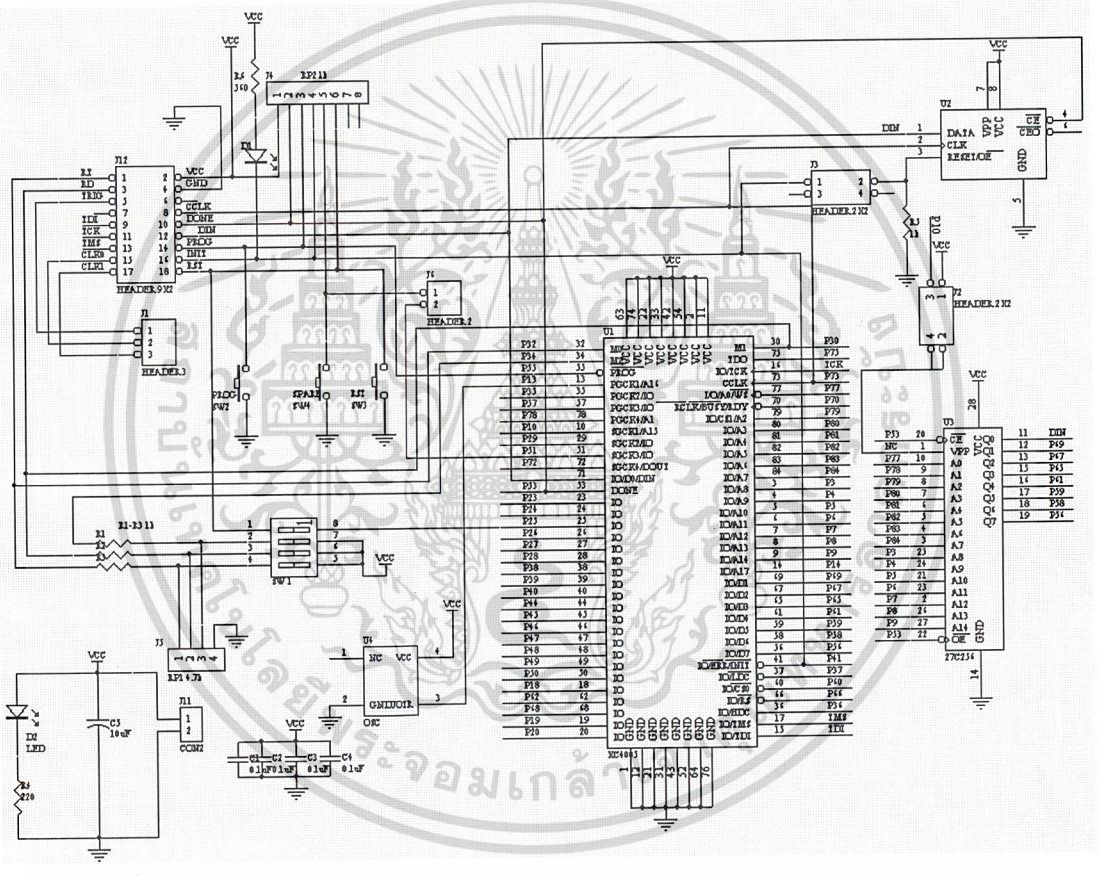
รูปที่ 4.3 วงจรของโมดูลสายดาวน์โหลด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.1.2 โมดูลบอร์ดดาวน์โหลด (Download Board Module)

โมดูลส่วนนี้ก็มีความสำคัญไม่แพ้ส่วนของสายดาวน์โหลด เพราะจะใช้งานร่วมกันซึ่งเมื่อสายดาวน์โหลดส่งข้อมูลมาให้ตัวบอร์ดนี้ก็จะทำการแปลงไฟล์ และเก็บข้อมูลใส่ลงในตัวชิฟ เอฟพีจีเอ สำหรับงานชิ้นนี้ จะสร้างคอนเน็คเอ้าพุทของตัว เอฟพีจีเอ เพื่อไปแสดงผลจากการโปรแกรมลงตัว เอฟพีจีเอ เพื่อแสดงการทำงานของโปรแกรมที่เขียนขึ้น

ลายวงจรของโมดูลบอร์ดดาวน์โหลด จะเป็นดังนี้

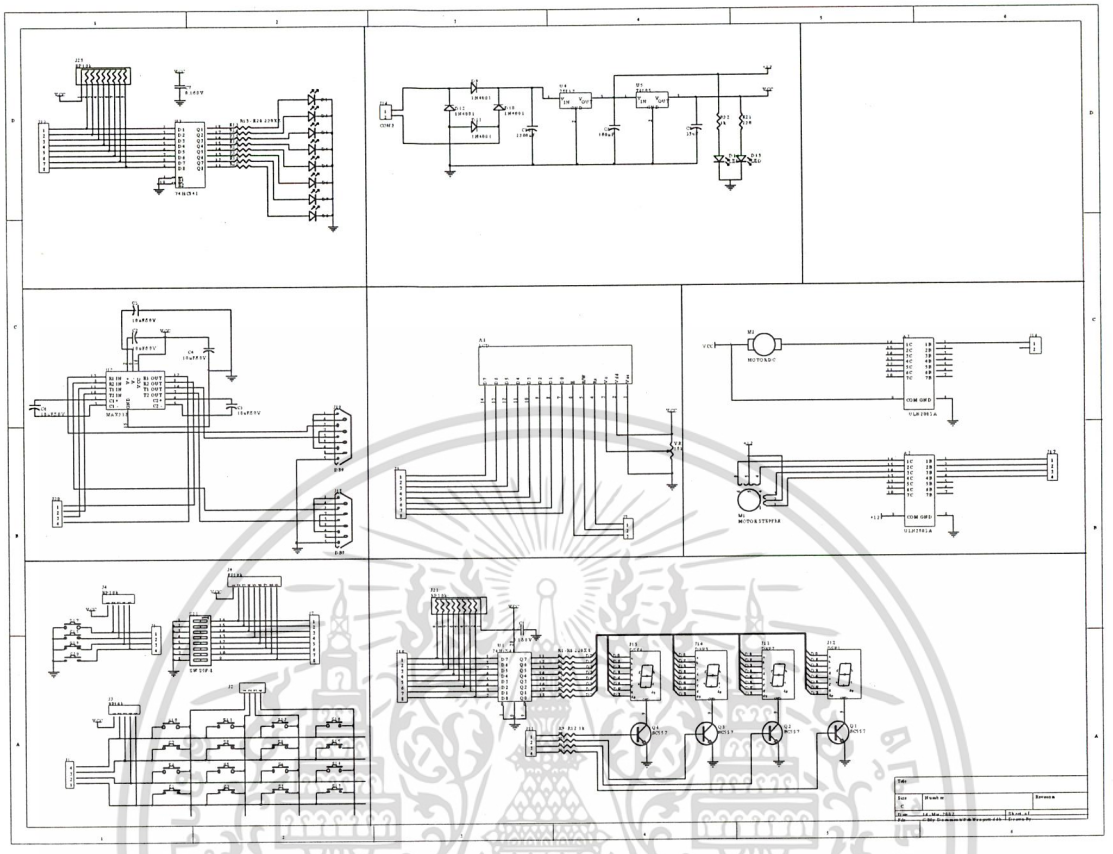


รูปที่ 4.4 วงจรของโมดูลบอร์ดดาวน์โหลด

4.1.3 โมดูลบอร์ดแสดงผล (Output Board Module)

เป็นส่วนแสดงผลเพื่อแสดงว่าการออกแบบของเราเป็นไปตามการโปรแกรมและความต้องการที่เราออกแบบหรือไม่ ซึ่งโครงงานชิ้นนี้ ได้แสดงส่วนของเอ้าท์พุทออกมา 5 แบบ และส่วนควบคุม 3 แบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่4.5 วงจรรวมของบอร์ดแสดงผล

4.1.3.1 ชุดแสดงผล

ประกอบด้วย

- ชุดแสดงผลการควบคุม DC motor และชุดแสดงผลการควบคุม Stepping motor
- ชุดแสดงผลการควบคุม LED
- ชุดแสดงผลการควบคุม LED 7ส่วน
- ชุดสำรองแสดงผล LCD

4.1.3.2 ชุดควบคุม

ประกอบด้วย

- ชุด Scankey
- ชุด Dipswitch และ Switch กด ปล่อย
- ชุดสำรองพอร์ตอนุกรม

ซึ่งแต่ละชุดจะมีวงจร Schematic ดังนี้

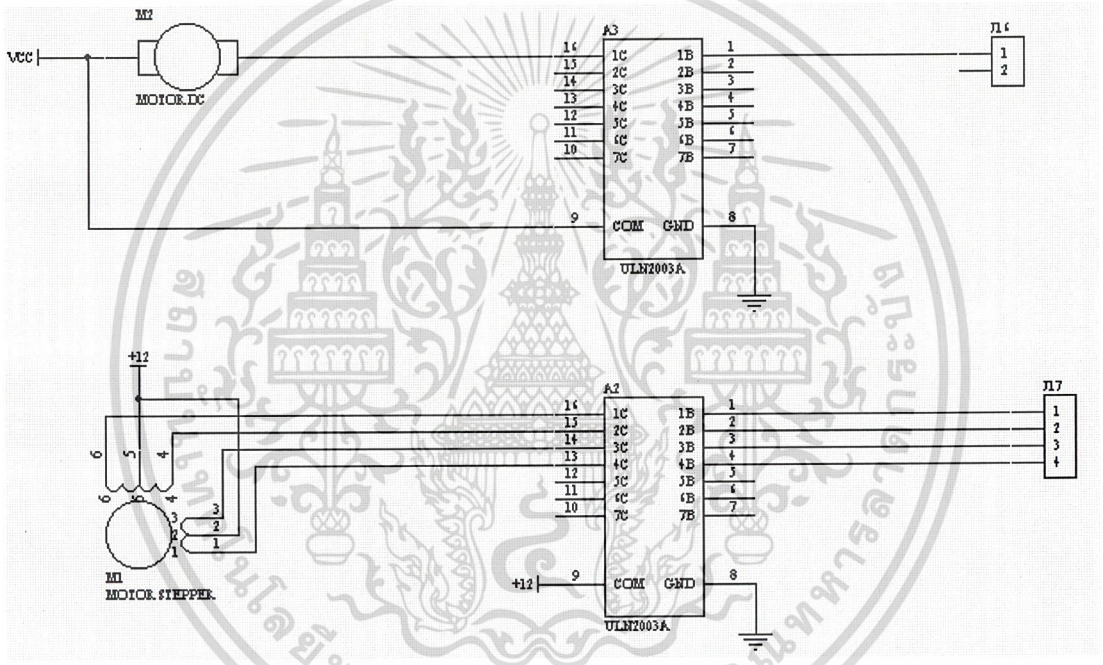
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โดยแต่ละส่วนจะมีรายละเอียดดังนี้

4.1.3.1 ชุดแสดงผล

- ชุดแสดงผลการควบคุม DC motor และชุดแสดงผลการควบคุม Stepping motor

ภายในวงจรการควบคุมการแสดงผลนี้จะประกอบด้วย ไอซีไดร์เวอร์ ULN2003 เป็นตัวขับ4บิท ในส่วนของ stepping motor ซึ่งทำงานโดย ลอจิก 0 เข้าที่ อินพุทของไอซี และในส่วนของ DC motor ควบคุมเพียง 1 บิท แต่ป้อน ลอจิก 0 เหมือนกัน ดังรูปของวงจรดังนี้

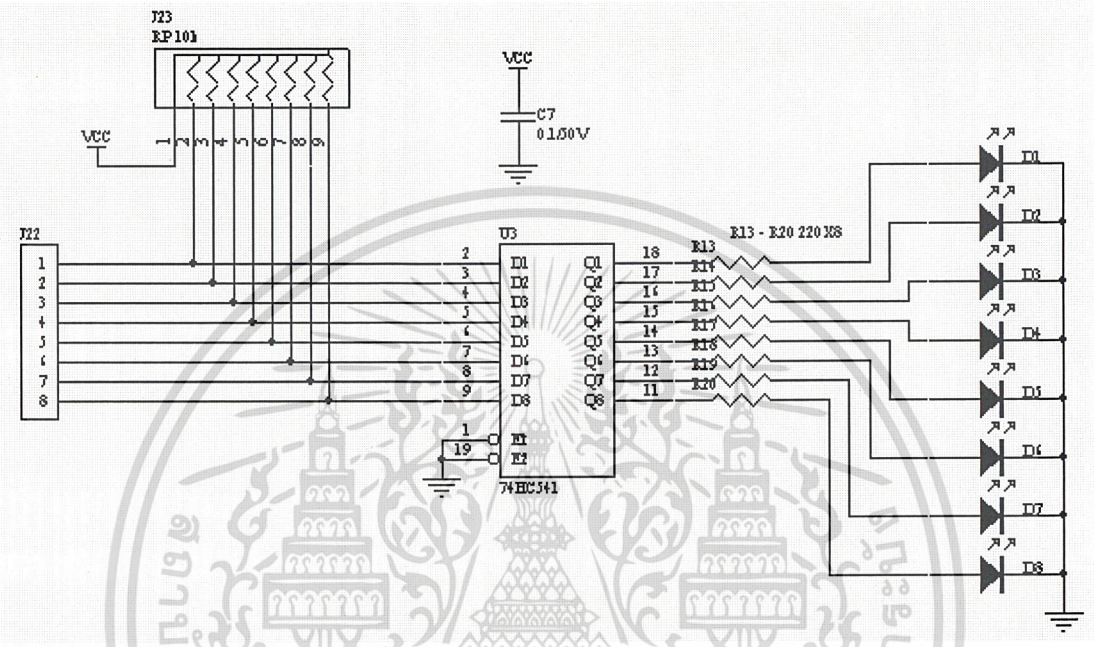


รูปที่ 4.6 วงจรชุดแสดงผลการควบคุมมอเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- ชุดแสดงผลการควบคุม LED

จะใช้ ไอซีแลตช์ขนาด 8บิททำหน้าที่ขับ LED ให้สว่างตามข้อมูลที่ส่งเข้ามาทางอินพุท และใช้ ตัวต้านทานขนาดจำกัดกระแส ขนาด R101-R108ป้องกัน LED

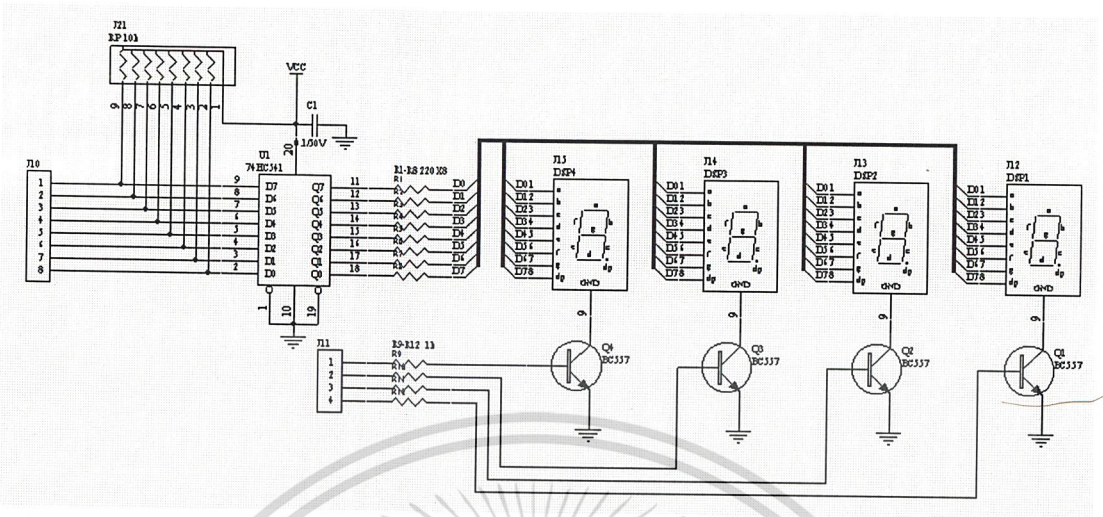


รูปที่4.7 ชุดแสดงผลการควบคุม LED

- ชุดแสดงผลการควบคุม LED 7 ส่วน

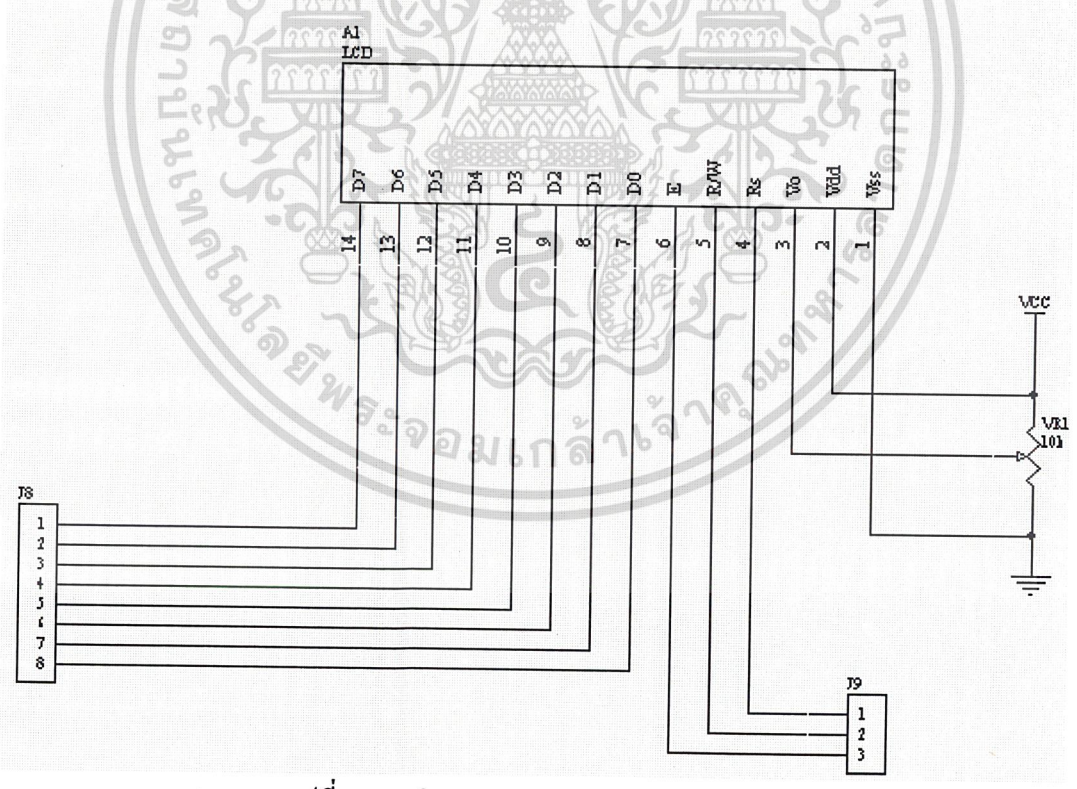
ประกอบด้วยไอซี เบอร์ 74C541 เป็น ไอซีบัฟเฟอร์ขนาด 8 บิททำหน้าที่รับข้อมูลจากอินพุทแล้วส่งไปขับในแต่ละ segment ,R201-R208,ตัวต้านทานไบแอสทรานซิสเตอร์ R209-R210, ทรานซิสเตอร์ Q201-Q202 และ LED 7ส่วน การต่อ LED 7 ส่วนทำในลักษณะมัลติเพล็กนำขาเซ็กเม้นทั้ง8 มาต่อรวมกัน ส่วนขาร่วมจะแยกเพื่อใช้ในการควบคุมการ ติดดับการจะให้หลอดติดสว่างต้องป้อน ลอจิก 1ส่วนลอจิกที่ใช้ในดารควบคุมที่ขาร่วมคือ ลอจิก 0 ตามรูป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่4.8 ชุดแสดงผลการควบคุม LED 7 ส่วน

- ชุดสำรองแสดงผลการควบคุมLCD



รูปที่4.9 ชุดสำรองแสดงผลการควบคุมLCD

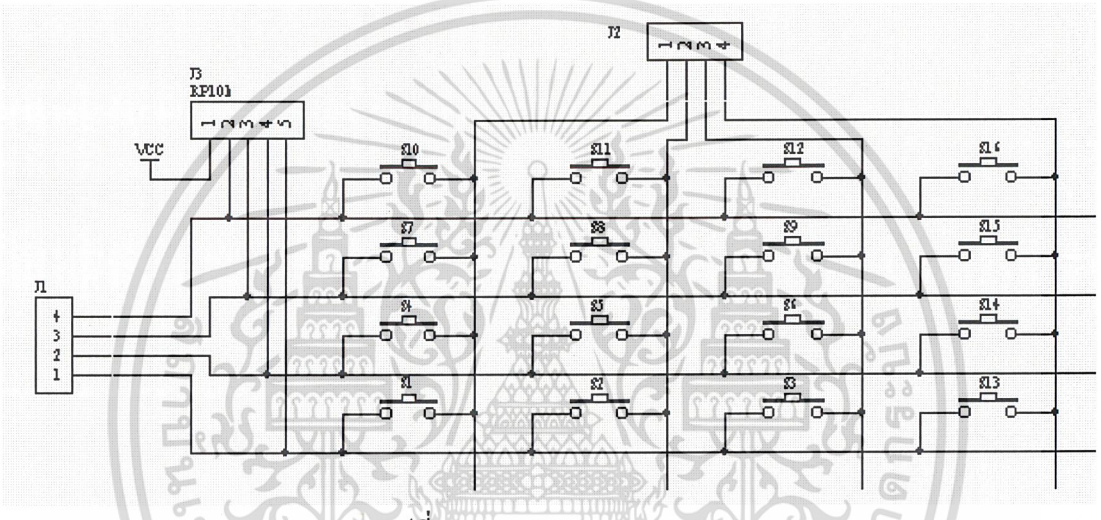
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.1.3.2 ชุดควบคุม

ประกอบด้วย

-ชุด Scankey

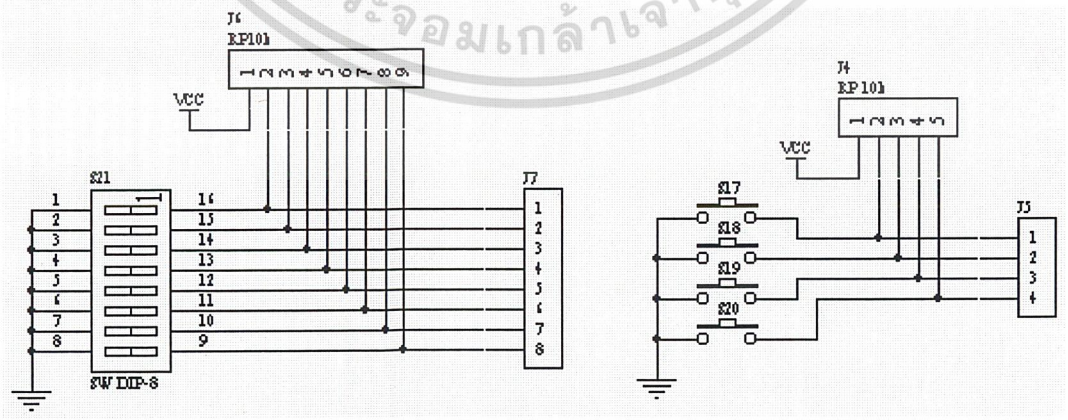
การทำงานเมื่อสวิตซ์ถูกกด ทำให้เกิดลอจิก 0แต่ถ้าไม่ต่อวงจร จะมีสถานะทางลอจิกเป็น 1 เป็นผลมาจากการต่อ ตัวต้านทานพูลอัพที่ต่อไว้



รูปที่4.10 ชุดควบคุม Scankey

-ชุด Dipswitch และ Switch กด ปล่อย

การทำงานเหมือนกับหัวข้อ 4.8.3.2.1

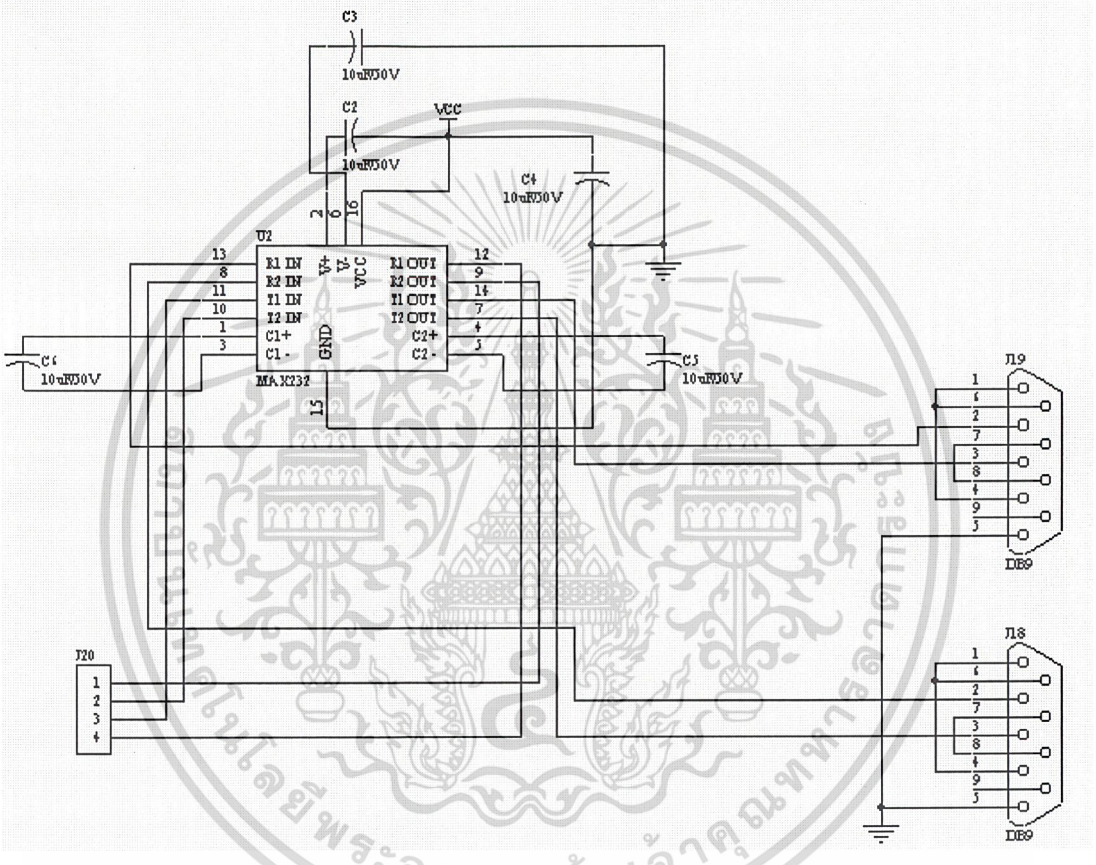


รูปที่4.11 ชุดควบคุมชุด Dipswitch และ Switch กด ปล่อย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

-ชุดสำรองพอร์ตอนุกรม

เนื่องจาก FPGA ใช้ไฟเลี้ยง +5 V จึงไม่สามารถเชื่อมต่อกับ พอร์ตอนุกรมของคอมพิวเตอร์ได้ เนื่องจากแรงดันในพอร์ตจะอยู่ในระหว่าง +3 - +12 V ดังนั้นในการเชื่อมต่อจะต้องใช้ ไอซีไดร์เวอร์ เฉพาะในวงจร Max232 ทำให้การรับส่งข้อมูลกับ พอร์ตอนุกรมอยู่ใต้ระดับแรงดัน ทีทีแอลเหมือนกัน

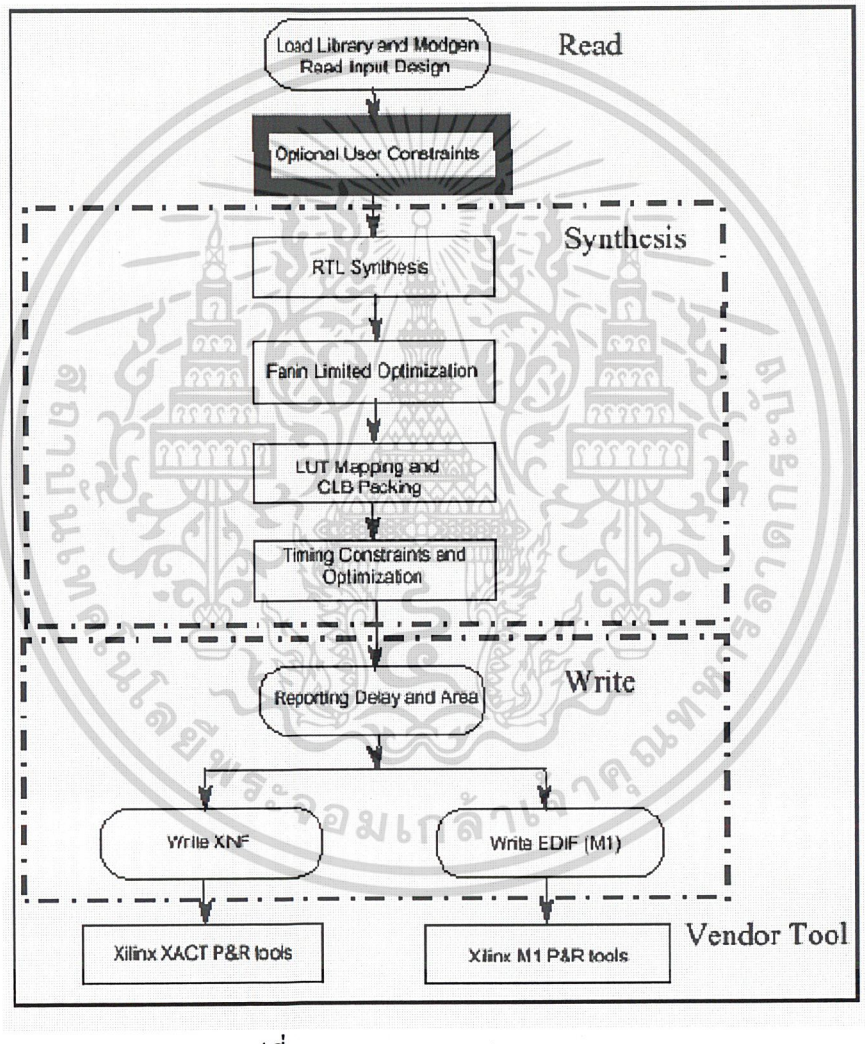


รูปที่4.12 ชุดสำรองพอร์ตอนุกรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2 การใช้โปรแกรม Exemplar Leonardo Spectrum เพื่อการสังเคราะห์วงจร

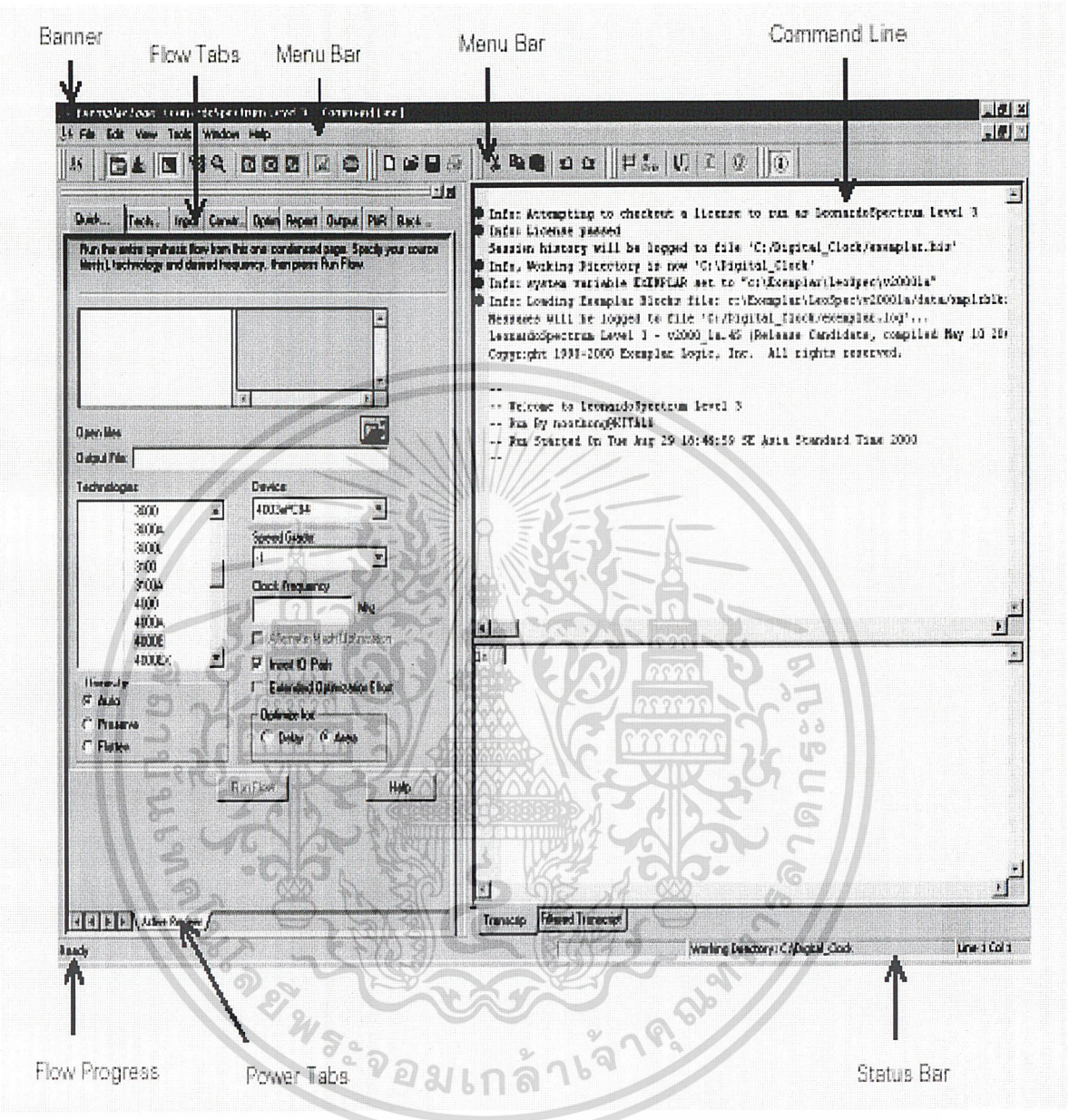
เมื่อได้ออกแบบวงจรดิจิตอลโดยใช้ภาษา VHDL และรวมถึงการตรวจสอบวงจรในระดับฟังก์ชัน (Functional) เรียบร้อยแล้ว ขั้นตอนต่อไปคือ การนำไฟล์ภาษา VHDLของดีไซน์ไปสังเคราะห์วงจร (Synthesis)โดยใช้โปรแกรมช่วยในการสังเคราะห์ (Synthesis tool) ที่มีชื่อว่า Exemplar LeonardoSpectrum ซึ่งมีขั้นตอนต่างๆ ดังรูปที่ 4.13



รูปที่ 4.13 Synthesis Design Flows

4.2.1 เรียกโปรแกรม Exemplar LeonardoSpecctrum 2000.1a

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.14 หน้าต่างของโปรแกรม Exemplar LeonardoSpecctrum 2000.1a

***ตัวอย่างจะทดลองสังเคราะห์ดีไซน์ที่เขียนจากภาษา VHDL เชื่อว่า DISMUX.VHDL โดยเลือกใช้เทคโนโลยี FPGA ของ Xilinx เบอร์ XC4010EPC84-4 โดยใช้ Exemplar Logic LeonaroSpectrum

DISMUX.VHB

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Library IEEE;
use IEEE.Std_Logic_1164.all;

entity DISPLAY_MUX is
port ( ALARM_TIME, CURRENT_TIME  : in std_ulogic_vector (3 downto 0);
       SHOW_A                    : in std_ulogic;
       SOUND_ALARM               : out std_ulogic;
       DISPLAY_TIME              : out std_ulogic_vector (3 downto 0)
     );
end DISPLAY_MUX;

architecture RTL of DISPLAY_MUX is

begin

   process (ALARM_TIME,CURRENT_TIME,SHOW_A)
   begin
            if (SHOW_A='1') then
                  DISPLAY_TIME <= ALARM_TIME;
            else
                  DISPLAY_TIME <= CURRENT_TIME;
            end if;

            if (ALARM_TIME = CURRENT_TIME) then
                  SOUND_ALARM <= '1';
            else
                  SOUND_ALARM <= '0';
            end if;

   end process;

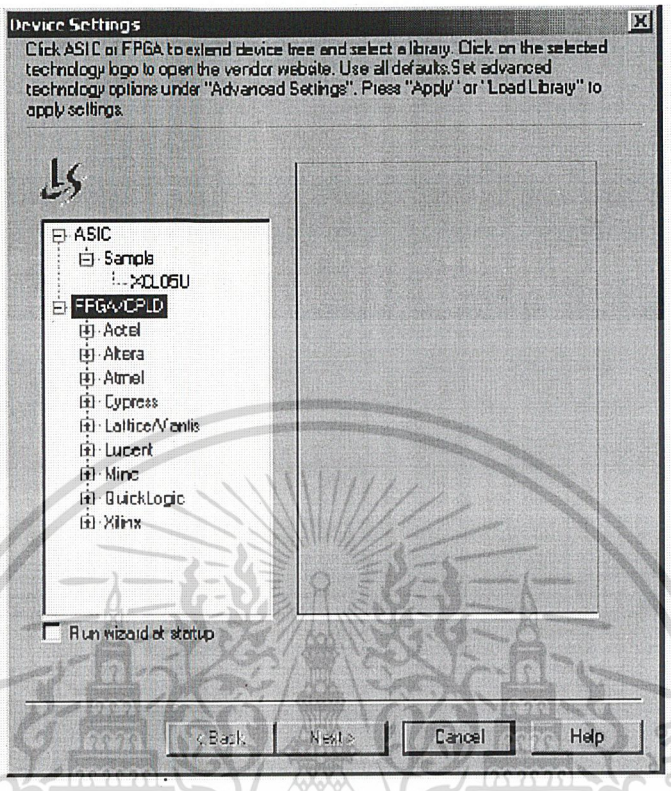
end RTL;
```

4.2.2 คลิก Synthesis Wizard  เพื่อเลือกอุปกรณ์ (Device) จากไลบรารี่ ซึ่งมีให้เลือก 2 ไลบรารี่คือ

- ASIC Library เป็นไลบรารี่ของอุปกรณ์ ASIC (Application Speciffic Integrated Circuits) เทคโนโลยี

- FPGA/CPLD Library เป็นไลบรารี่ของอุปกรณ์ FPGA(Field Programmable Gate Array) หรือ CPLD (Complex Programmable Logic Device)ตระกูลต่างๆ ดังรูป

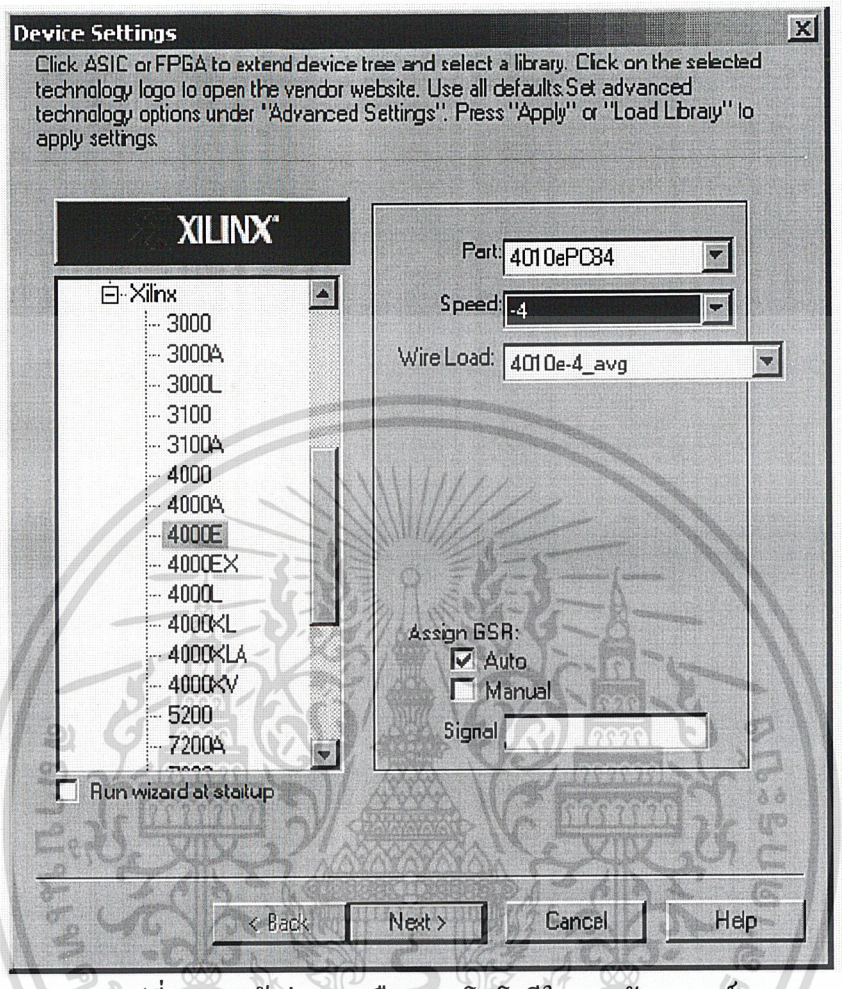
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.15 หน้าต่างการเลือกเทคโนโลยีในการสังเคราะห์

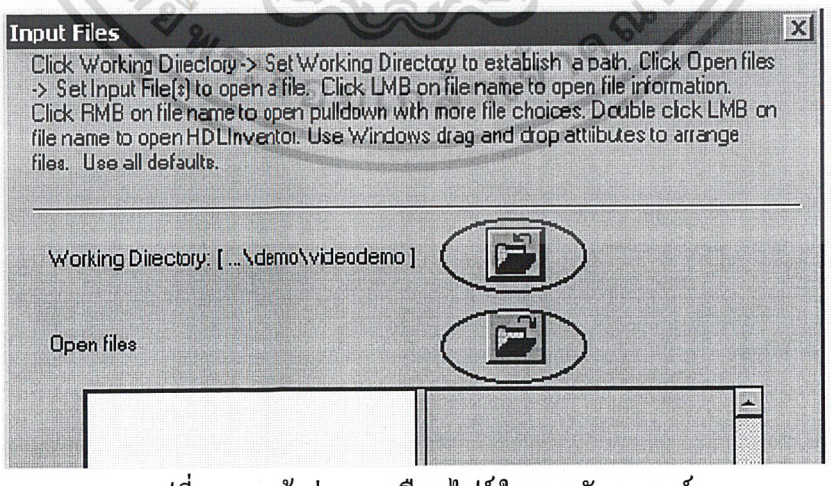
4.2.3 ให้เลือกเทคโนโลยีของ FPGA/CPLDโดยให้เลือกตระกูล FPGAที่ต้องการใช้งานโดยในที่นี้เลือก FPGAจาก บริษัท Xilinx เบอร์ XC4010ePC84 Speed เท่ากับ –4 ดังรูปที่4.16

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.16 หน้าต่างการเลือกเทคโนโลยีในการสังเคราะห์

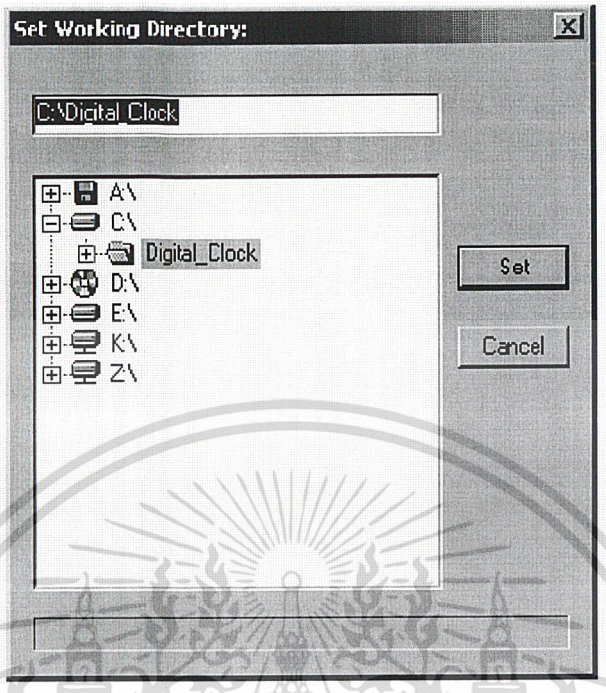
4.2.4 คลิก Next > เพื่อกำหนด Working Directory และไฟล์ภาษา VHDL ที่ต้องการสังเคราะห์



รูปที่ 4.17 หน้าต่างการเลือกไฟล์ในการสังเคราะห์

4.2.5 เลือก Working Directory เป็น C:/Digital_Clock โดยคลิก

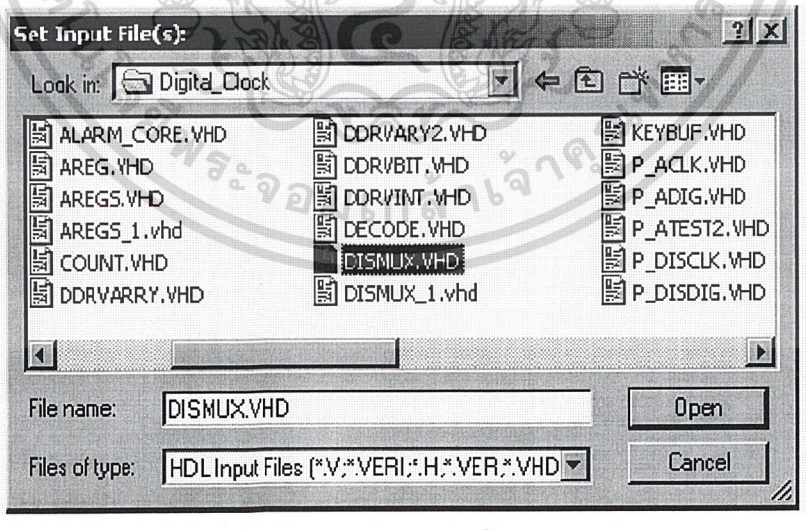
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.18 หน้าต่างการเลือกไฟล์ในการสังเคราะห์

4.2.6 คลิก 

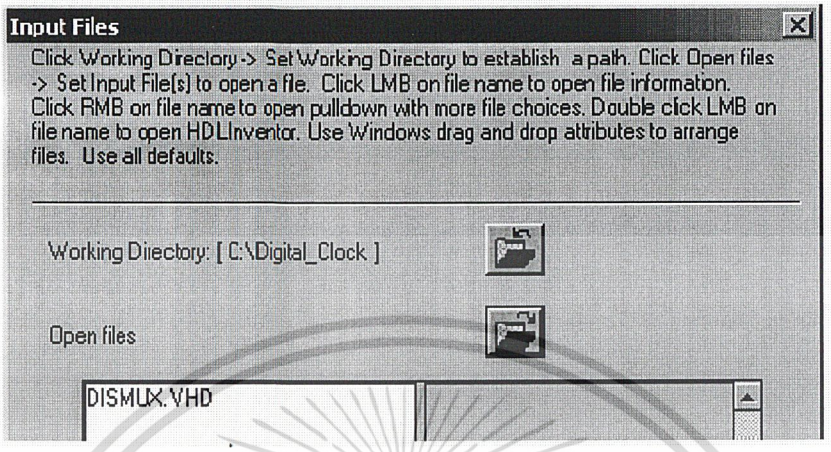
เพื่อเปลี่ยน Working Directory จากนั้นให้เลือกไฟล์ที่ต้องการสังเคราะห์ โดย เลือก DISMUX.VHK



รูปที่ 4.19 หน้าต่างการเลือกไฟล์ในการสังเคราะห์

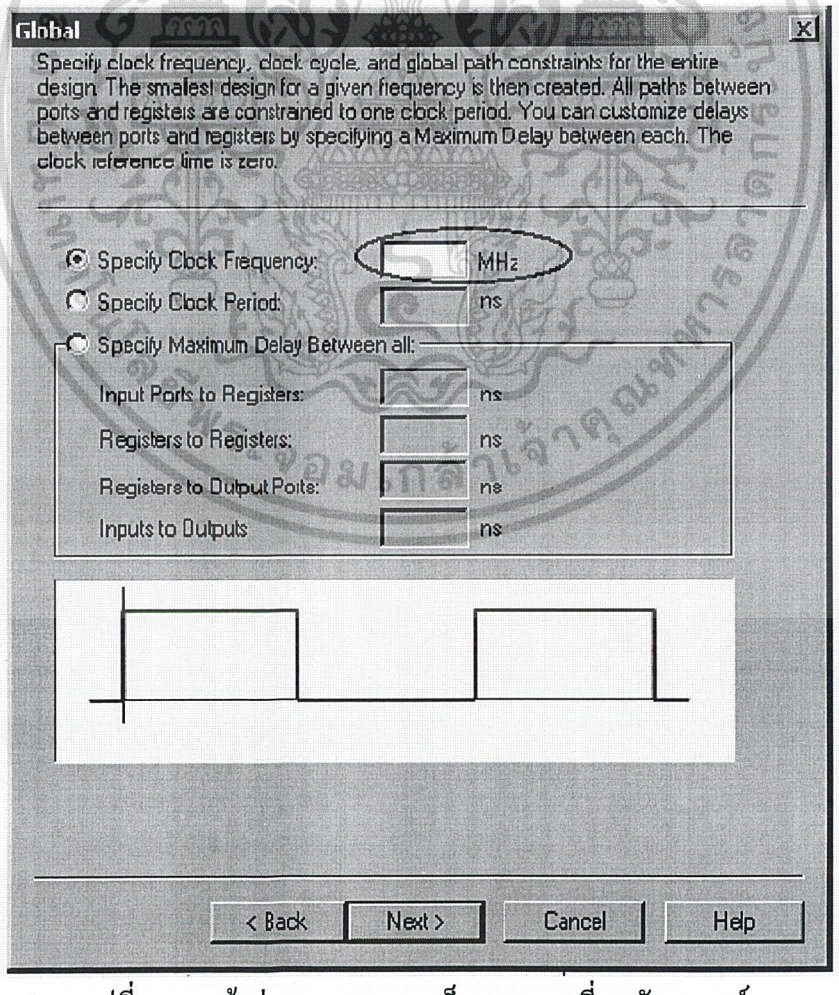
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2.7 คลิก Open เพื่อ Load ไฟล์ที่ต้องการสังเคราะห์



รูปที่4.20 หน้าต่างการเลือกไฟล์ในการสังเคราะห์

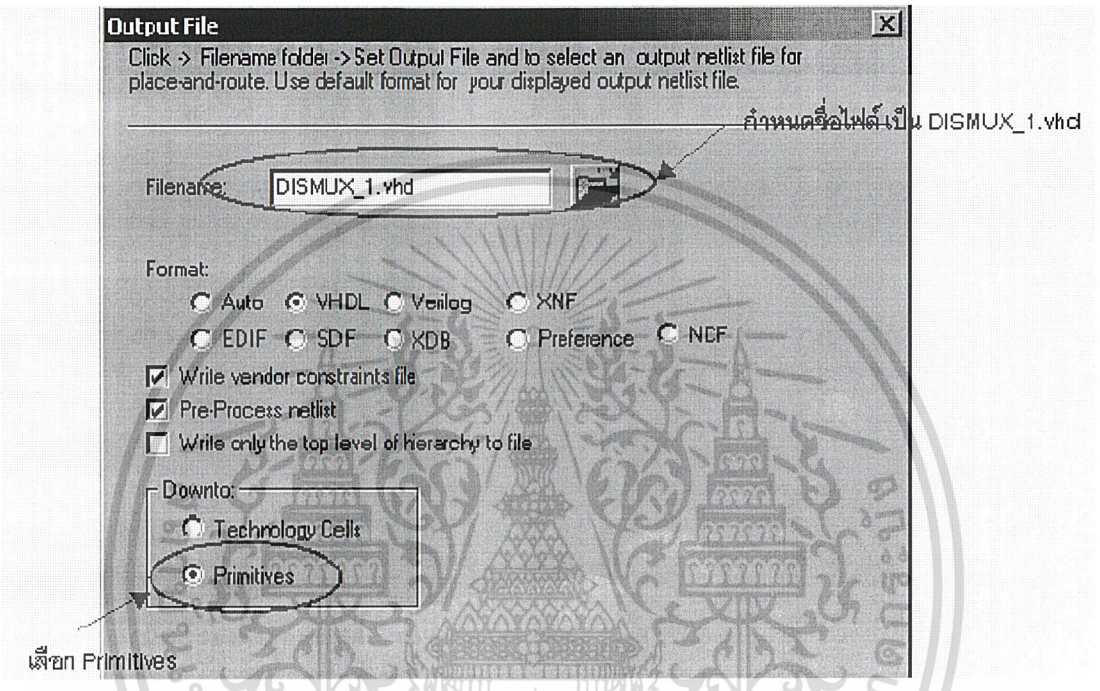
4.2.8 คลิก Next > เพื่อกำหนดความเร็วของวงจรที่จะทำการสังเคราะห์



รูปที่ 4.21 หน้าต่างการเซตความเร็วของวงจรที่ถูกสังเคราะห์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2.9 คลิก Next > เพื่อกำหนดไฟล์เอาต์พุต โดยให้เลือก Format เป็น VHDL ไฟล์เพื่อนำไปกลับซิมมูลเลชั่น ในระดับ Gate Level โดยใช้โปรแกรม ModelSim ต่อไปเพื่อตรวจสอบความถูกต้องของการสังเคราะห์

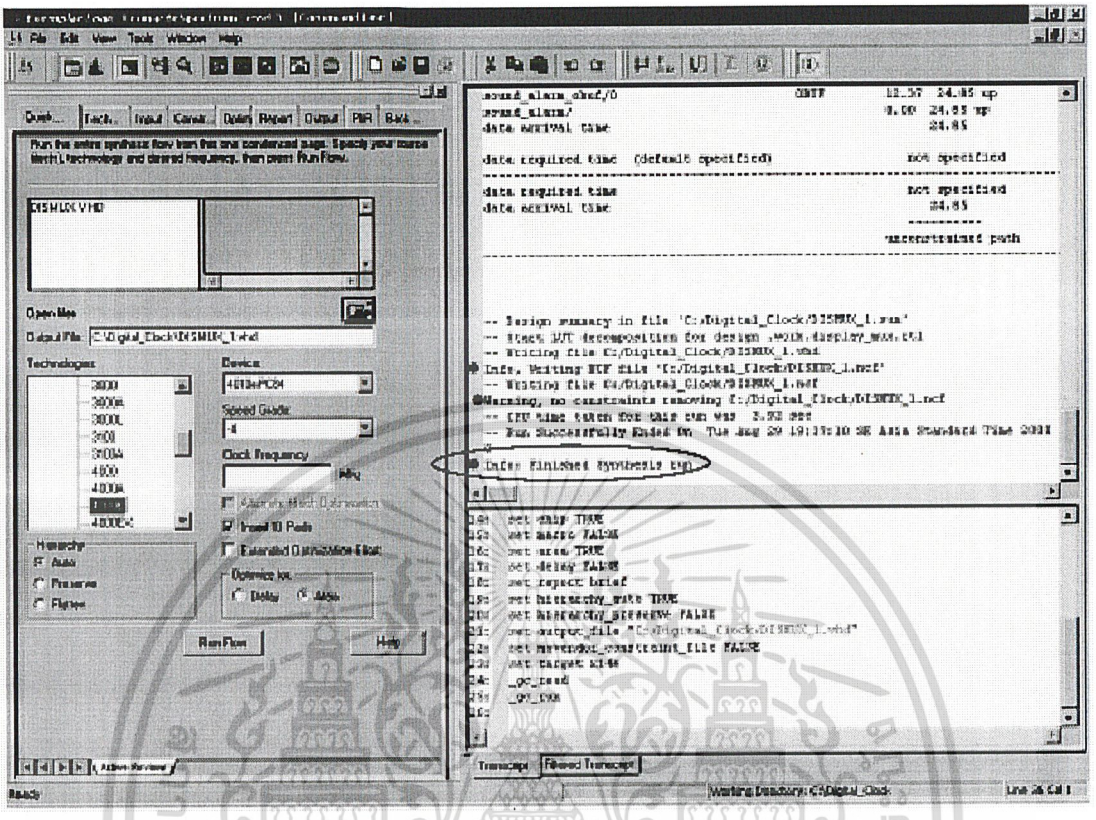


รูปที่ 4.22 หน้าต่างการเซตเอาต์พุตไฟล์ที่ได้จากการสังเคราะห์

โดยใช้เลือกOptionของ Downto: เป็น Primitives ในการเลือก Format แบบ VHDL และเลือก Option ของ Downto: เป็นTechnology Cellsในการเลือก Format แบบ EDIF หรือ XNF

4.2.10 คลิก Finish เพื่อให้โปรแกรมเริ่มทำการสังเคราะห์ดีไซน์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.23หน้าต่างโปรแกรมที่แสดงค่าต่างๆ ในการสังเคราะห์วงจร

ถ้าผลการสังเคราะห์ดีไซน์ที่ออกแบบไม่พบปัญหาใด โปรแกรมจะรายงานการสังเคราะห์วงจรที่ได้ เช่น ขนาดของวงจร ที่สังเคราะห์ ความเร็วของวงจร (ถ้ามี) แต่ถ้าพบปัญหาในการสังเคราะห์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรมจะทำการแจ้งและหยุดให้เราทำการแก้ไขต่อไป

```
*******************************************************

Cell: display_mux     View: rtl     Library: work

*******************************************************

Total accumulated area :
 Number of FG Function Generators :       6
 Number of H Function Generators :        1
 Number of IBUF :                         9
 Number of OBUF :                         5
 Number of Packed CLBs :                  3

 Number of ports :                       14
 Number of nets :                        30
 Number of instances :                   21
 Number of references to this view :      0

            Cell         Library  References     Total Area
F3_LUT                   xi4e     4 x     1     4 FG Function Generators
F4_LUT                   xi4e     2 x     1     2 FG Function Generators
H2_LUT                   xi4e     1 x     1     1 H Function Generators
  IBUF                   xi4e     9 x     1     9 IBUF
  OBUF                   xi4e     5 x     1     5 OBUF

*********************************************
Device Utilization for 4010ePC84
*********************************************
Resource                Used    Avail   Utilization
---------------------------------------------
IOs                     14      61        22.95%
FG Function Generators  6       800        0.75%
H Function Generators   1       400        0.25%
CLB Flip Flops          0       800        0.00%

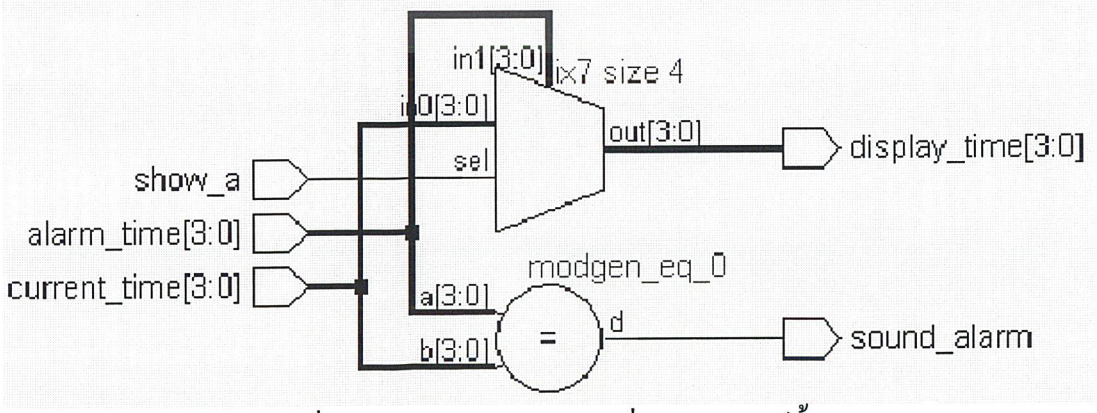
---------------------------------------------
Using wire table: 4010e-4_avg
```

อุปกรณ์ต่างที่ใช้ออกแบบวงจรที่สังเคราะห์ขึ้น

ชิพที่เลือกใช้

รูปที่ 4.24 แสดงผลของการสังเคราะห์วงจร

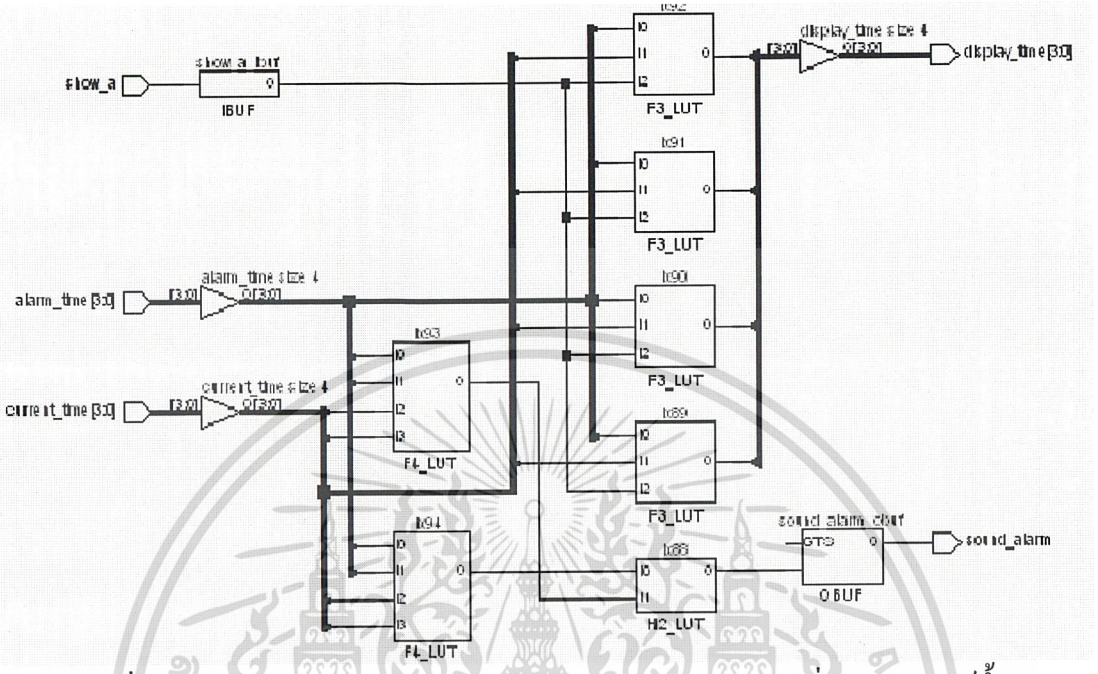
4.2.11 คลิก  เพื่อดูหน้าตาของวงจรในระดับ RTL ได้ (รูปแสดง RTL ของ DISMUX)



รูปที่ 4.25 แสดง RTL ของวงจรที่ถูกสังเคราะห์ขึ้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2.12 คลิก  เพื่อดูหน้าตาของวงจรในระดับ Technology ของ FPGA ได้

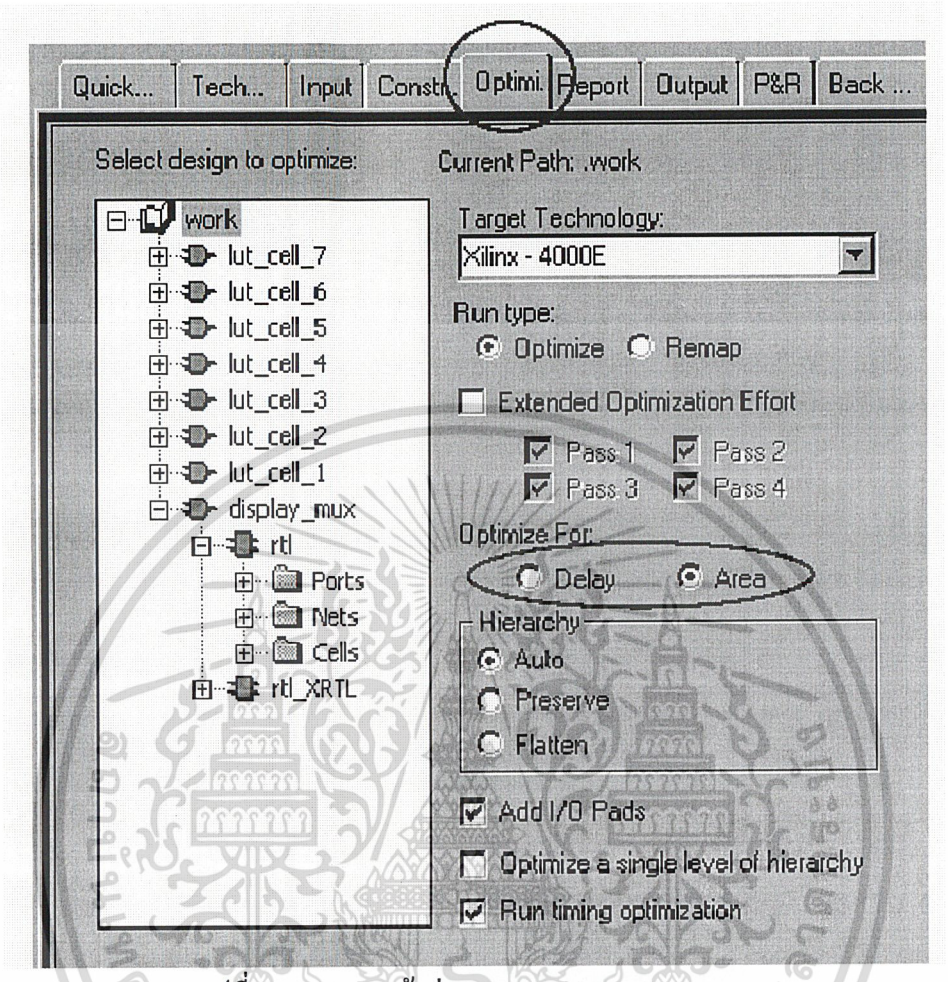


รูปที่ 4.26 แสดง Schematic ในระดับ Technology cells ของวงจรที่ถูกสังเคราะห์ขึ้น

4.2.13 คลิก  เพื่อแสดง Critical path(เส้นทางเดินของวงจรที่อาจทำให้วงจรวิกฤต) ที่สังเคราะห์ได้

4.2.14 ทำการ Optimizeวงจรที่ออกแบบเพื่อให้ผลการสังเคราะห์ที่ได้ออกมาเป็นไปตามที่ต้องการมากที่สุด โดยรูปแบบการ Optimize สามารถเลือกได้ 2 แบบคือ Optimize เพื่อให้วงจรที่ได้มีความเร็วสูงสุด (Delay)ส่วนอีกรูปแบบหนึ่งก็คือการ Optimize เพื่อให้ได้วงจรที่มีขนาดเล็กที่สุด (Area)

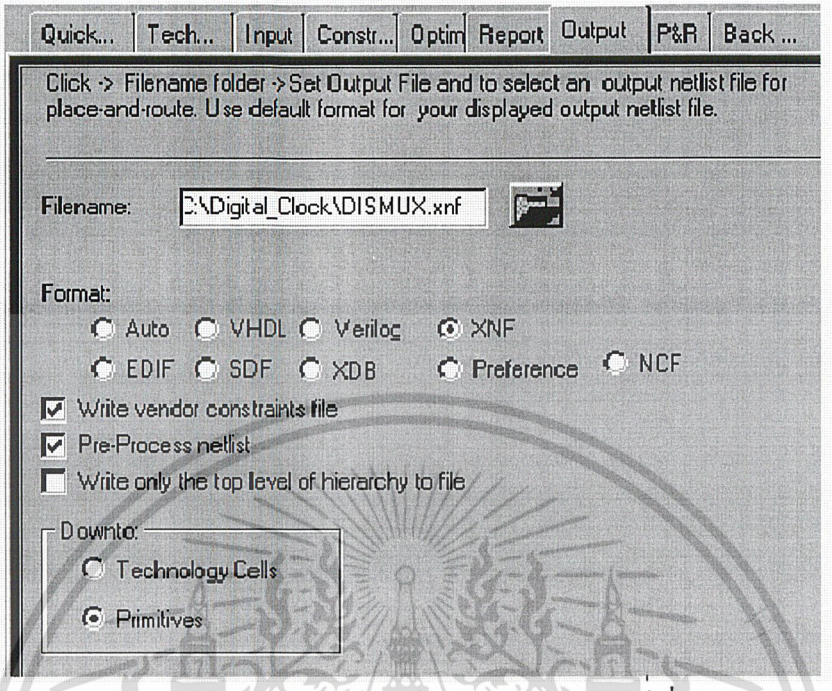
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.27 แสดงหน้าต่างของการ Optimize วงจร

4.2.15 หลังจากที่สังเคราะห์วงจรเรียบร้อยแล้วสามารถกำหนดรูปแบบเอาต์พุต Netlist ของวงจรเพื่อนำไป Place & Route สู่ชิพต่อไป โดยรูปแบบที่ใช้กันอย่างแพร่หลายจะมี 2 รูปแบบคือ รูปแบบแรกคือ EDIF (Electronics Data Interchange Format) อีกรูปแบบก็คือ XNF (Xilinx Netlist Format) โดยในที่นี้จะใช้แบบ XNF เพราะเราจะไปโปรแกรมลงชิพ FPGA ของ XILINX นอกจากนี้โปรแกรม Exemplar LeonardoSpectrum ยังสามารถเขียนวงจรที่ผ่านการสังเคราะห์เรียบร้อยแล้วกลับเป็น VHDL หรือ Verilog และเขียนข้อมูล Delay ภายในรูปแบบของ SDF (Standard Delay Format) เพื่อใช้ในการ Simulation ในระดับ Timing ต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่4.28 แสดงหน้าต่างของการเลือกไฟล์เอาต์พุตในการสังเคราะห์เพื่อนำไป Place & Route

DISMUX.XNF

```
LCANET, 5
PART, 4010ePC84-4
USER, LIBRARY_NAME, work
USER, CELL_NAME, display_mux
USER, VIEW_NAME, rtl
PROG, LeonardoSpectrum Level 3, v2000_1a.45, "09/02/00 14:14:45"
EXT, alarm_time<3>, I,
EXT, alarm_time<2>, I,
EXT, alarm_time<1>, I,
EXT, alarm_time<0>, I,
EXT, current_time<3>, I,
EXT, current_time<2>, I,
EXT, current_time<1>, I,
EXT, current_time<0>, I,
EXT, show_a, I,
EXT, sound_alarm, O,
EXT, display_time<3>, O,
EXT, display_time<2>, O,
EXT, display_time<1>, O,
EXT, display_time<0>, O,
SYM, show_a_ibuf, IBUF, LIBVER=2.0.0, SCHNM=IBUF, FILE=ibuf.xnf
PIN, I, I, show_a,
PIN, O, O, show_a_int, 3.375000
END
SYM, current_time<0>_ibuf, IBUF, LIBVER=2.0.0, SCHNM=IBUF, FILE=ibuf.xnf
PIN, I, I, current_time<0>,
PIN, O, O, current_time<0>_int, 3.375000
END
SYM, current_time<1>_ibuf, IBUF, LIBVER=2.0.0, SCHNM=IBUF, FILE=ibuf.xnf
PIN, I, I, current_time<1>,
PIN, O, O, current_time<1>_int, 3.375000
END
SYM, current_time<2>_ibuf, IBUF, LIBVER=2.0.0, SCHNM=IBUF, FILE=ibuf.xnf
PIN, I, I, current_time<2>,
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
PIN, O, O, current_time<2>_int, 3.375000
END
SYM, current_time<3>_ibuf, IBUF, LIBVER=2.0.0, SCHNM=IBUF, FILE=ibuf.xnf
PIN, I, I, current_time<3>,
PIN, O, O, current_time<3>_int, 3.375000
END
SYM, alarm_time<0>_ibuf, IBUF, LIBVER=2.0.0, SCHNM=IBUF, FILE=ibuf.xnf
PIN, I, I, alarm_time<0>,
PIN, O, O, alarm_time<0>_int, 3.375000
END
SYM, alarm_time<1>_ibuf, IBUF, LIBVER=2.0.0, SCHNM=IBUF, FILE=ibuf.xnf
PIN, I, I, alarm_time<1>,
PIN, O, O, alarm_time<1>_int, 3.375000
END
SYM, alarm_time<2>_ibuf, IBUF, LIBVER=2.0.0, SCHNM=IBUF, FILE=ibuf.xnf
PIN, I, I, alarm_time<2>,
PIN, O, O, alarm_time<2>_int, 3.375000
END
SYM, alarm_time<3>_ibuf, IBUF, LIBVER=2.0.0, SCHNM=IBUF, FILE=ibuf.xnf
PIN, I, I, alarm_time<3>,
PIN, O, O, alarm_time<3>_int, 3.375000
END
SYM, display_time<0>_obuf, OBUF, LIBVER=2.0.0, SCHNM=OBUF, FILE=obuf.xnf
PIN, O, O, display_time<0>, 11.475000
PIN, I, I, display_time_dup0<0>,
END
SYM, display_time<1>_obuf, OBUF, LIBVER=2.0.0, SCHNM=OBUF, FILE=obuf.xnf
PIN, O, O, display_time<1>, 11.475000
PIN, I, I, display_time_dup0<1>,
END
SYM, display_time<2>_obuf, OBUF, LIBVER=2.0.0, SCHNM=OBUF, FILE=obuf.xnf
PIN, O, O, display_time<2>, 11.475000
PIN, I, I, display_time_dup0<2>,
END
SYM, display_time<3>_obuf, OBUF, LIBVER=2.0.0, SCHNM=OBUF, FILE=obuf.xnf
PIN, O, O, display_time<3>, 11.475000
PIN, I, I, display_time_dup0<3>,
END
SYM, sound_alarm_obuf, OBUF, LIBVER=2.0.0, SCHNM=OBUF, FILE=obuf.xnf
PIN, O, O, sound_alarm, 11.475000
PIN, I, I, sound_alarm_dup0,
END
SYM, ix88, EQN, EQN=((I0*I1)), LIBVER=2.0.0
PIN, I0, I, nx83,
PIN, I1, I, nx82,
PIN, O, O, sound_alarm_dup0,
END
SYM, ix89, EQN, EQN=((I0*I2)+(I1*~I2)), LIBVER=2.0.0
PIN, I0, I, alarm_time<3>_int,
PIN, I1, I, current_time<3>_int,
PIN, I2, I, show_a_int,
PIN, O, O, display_time_dup0<3>,
END
SYM, ix90, EQN, EQN=((I0*I2)+(I1*~I2)), LIBVER=2.0.0
PIN, I0, I, alarm_time<2>_int,
PIN, I1, I, current_time<2>_int,
PIN, I2, I, show_a_int,
PIN, O, O, display_time_dup0<2>,
END
SYM, ix91, EQN, EQN=((I0*I2)+(I1*~I2)), LIBVER=2.0.0
PIN, I0, I, alarm_time<1>_int,
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
PIN, I1, I, current_time<1>_int,
PIN, I2, I, show_a_int,
PIN, O, O, display_time_dup0<1>,
END
SYM, ix92, EQN, EQN=((I0*I2)+(I1*~I2)), LIBVER=2.0.0
PIN, I0, I, alarm_time<0>_int,
PIN, I1, I, current_time<0>_int,
PIN, I2, I, show_a_int,
PIN, O, O, display_time_dup0<0>,
END
SYM, ix93, EQN, EQN=((I0*I1*I2*I3)+(~I0*I1*~I2*I3)+(I0*~I1*I2*~I3)+
(~I0*~I1*~I2*~I3)), LIBVER=2.0.0
PIN, I0, I, alarm_time<3>_int,
PIN, I1, I, alarm_time<2>_int,
PIN, I2, I, current_time<3>_int,
PIN, I3, I, current_time<2>_int,
PIN, O, O, nx82,
END
SYM, ix94, EQN, EQN=((I0*I1*I2*I3)+(~I0*I1*~I2*I3)+(I0*~I1*I2*~I3)+
(~I0*~I1*~I2*~I3)), LIBVER=2.0.0
PIN, I0, I, alarm_time<1>_int,
PIN, I1, I, alarm_time<0>_int,
PIN, I2, I, current_time<1>_int,
PIN, I3, I, current_time<0>_int,
PIN, O, O, nx83,
END
SYM, HMAP_0, HMAP, LIBVER=2.0.0
PIN, I1, I, nx82
PIN, I2, I, nx83
PIN, O, I, sound_alarm_dup0
END
SYM, FMAP_1, FMAP, LIBVER=2.0.0
PIN, I1, I, alarm_time<3>_int
PIN, I2, I, current_time<3>_int
PIN, I3, I, show_a_int
PIN, O, I, display_time_dup0<3>
END
SYM, FMAP_2, FMAP, LIBVER=2.0.0
PIN, I1, I, alarm_time<2>_int
PIN, I2, I, current_time<2>_int
PIN, I3, I, show_a_int
PIN, O, I, display_time_dup0<2>
END
SYM, FMAP_3, FMAP, LIBVER=2.0.0
PIN, I1, I, alarm_time<1>_int
PIN, I2, I, current_time<1>_int
PIN, I3, I, show_a_int
PIN, O, I, display_time_dup0<1>
END
SYM, FMAP_4, FMAP, LIBVER=2.0.0
PIN, I1, I, alarm_time<0>_int
PIN, I2, I, current_time<0>_int
PIN, I3, I, show_a_int
PIN, O, I, display_time_dup0<0>
END
SYM, FMAP_5, FMAP, LIBVER=2.0.0
PIN, I1, I, alarm_time<3>_int
PIN, I2, I, alarm_time<2>_int
PIN, I3, I, current_time<3>_int
PIN, I4, I, current_time<2>_int
PIN, O, I, nx82
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
END
SYM, FMAP_6, FMAP, LIBVER=2.0.0
PIN, I1, I, alarm_time<1>_int
PIN, I2, I, alarm_time<0>_int
PIN, I3, I, current_time<1>_int
PIN, I4, I, current_time<0>_int
PIN, O, I, nx83
END
EOF
```

4.3 การใช้โปรแกรม XILINX FOUNDATION M Series เพื่อการ Place &Route วงจรลง FPGA

เมื่อทำการสังเคราะห์วงจรจากโปรแกรม Exemplar logic Leonadospectrum โดยผลการสังเคราะห์จะถูกเขียนไว้ในรูปของ Netlist แบบ XFT (Xilinx Netlist Format) ขั้นตอนต่อไปจะนำไฟล์เน็ตลิสต์ไป place & route โดยใช้โปรแกรม Xilinx Alliance M series โดยมีขั้นตอนต่อไปนี้

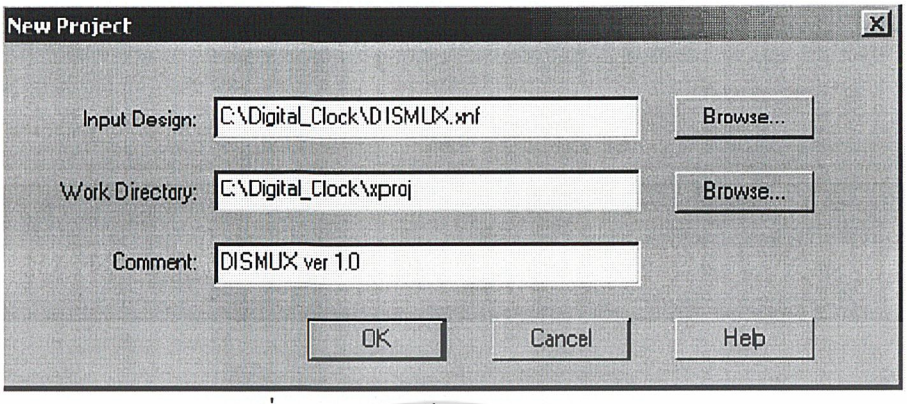
4.3.1 เรียก Design Manager ของ Xilinx Foundation M series  โดยคลิกไอคอน Design Manager



รูปที่4.29 แสดงหน้าต่างของ Design Manager

4.3.2 สร้างโปรเจคใหม่สำหรับดีไซน์ที่จะนำไป Place &Route โดยเลือก File>New Project

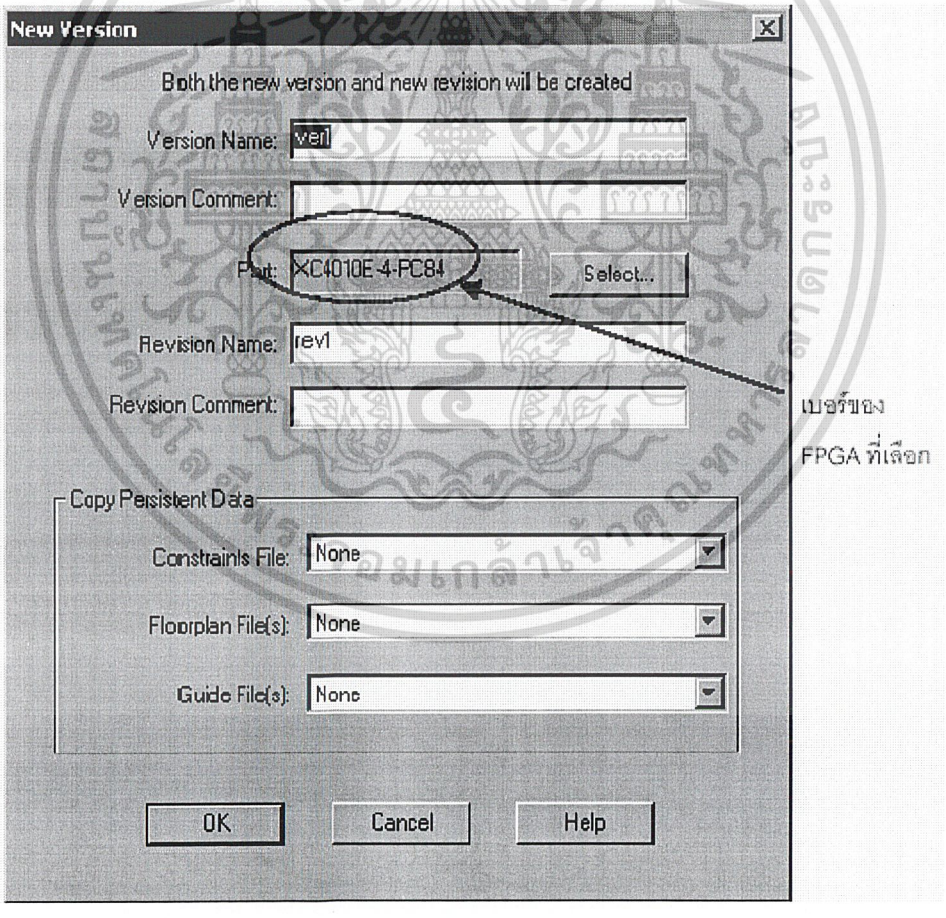
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่4.30 แสดงหน้าต่างของ New Project

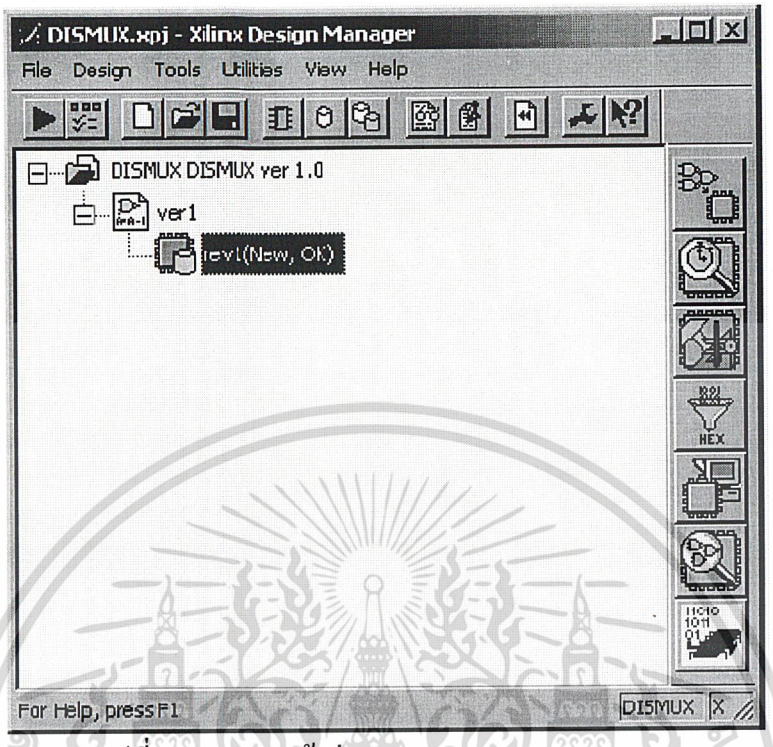
โดยเลือก Directory>c:\Digital_Clock\DISMUX.xnf

โดยโปรแกรมจะสร้าง directory>c:\Digital_Clock\xproj



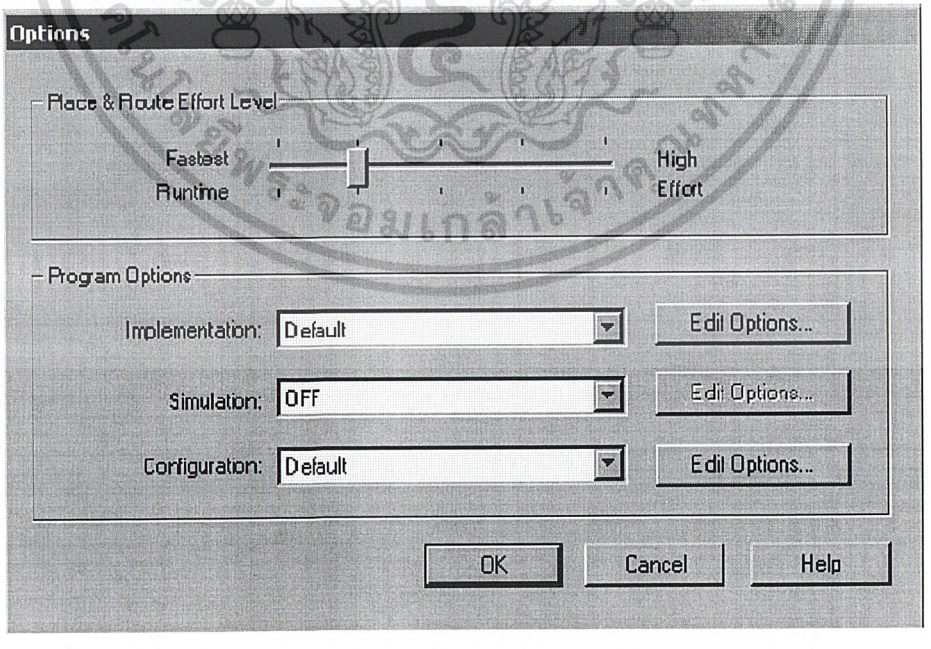
รูปที่4.31 แสดงหน้าต่างของ New Version

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



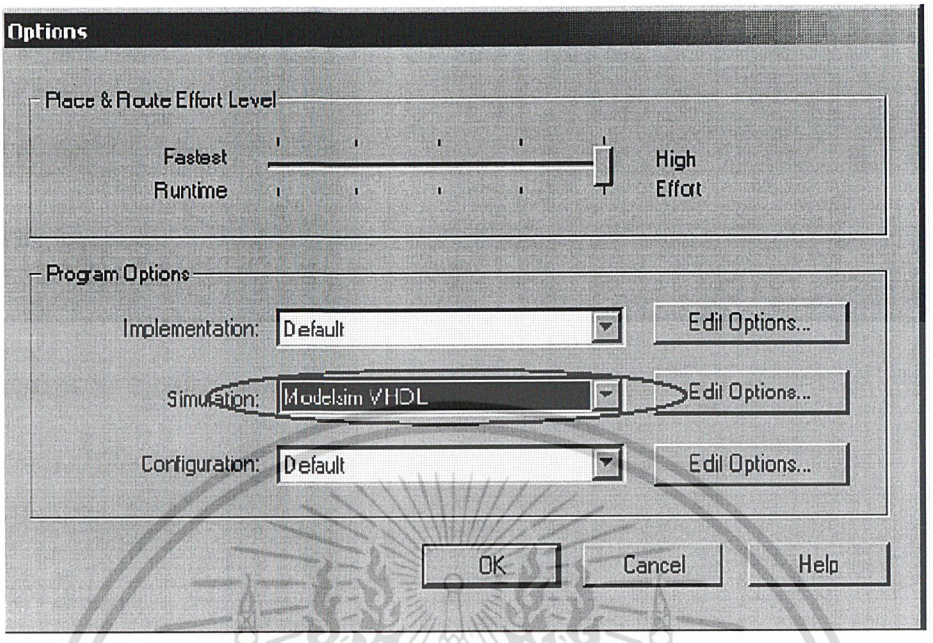
รูปที่4.32 แสดงหน้าต่างของ New Project Complete

4.3.3 คลิกเลือก Set option 



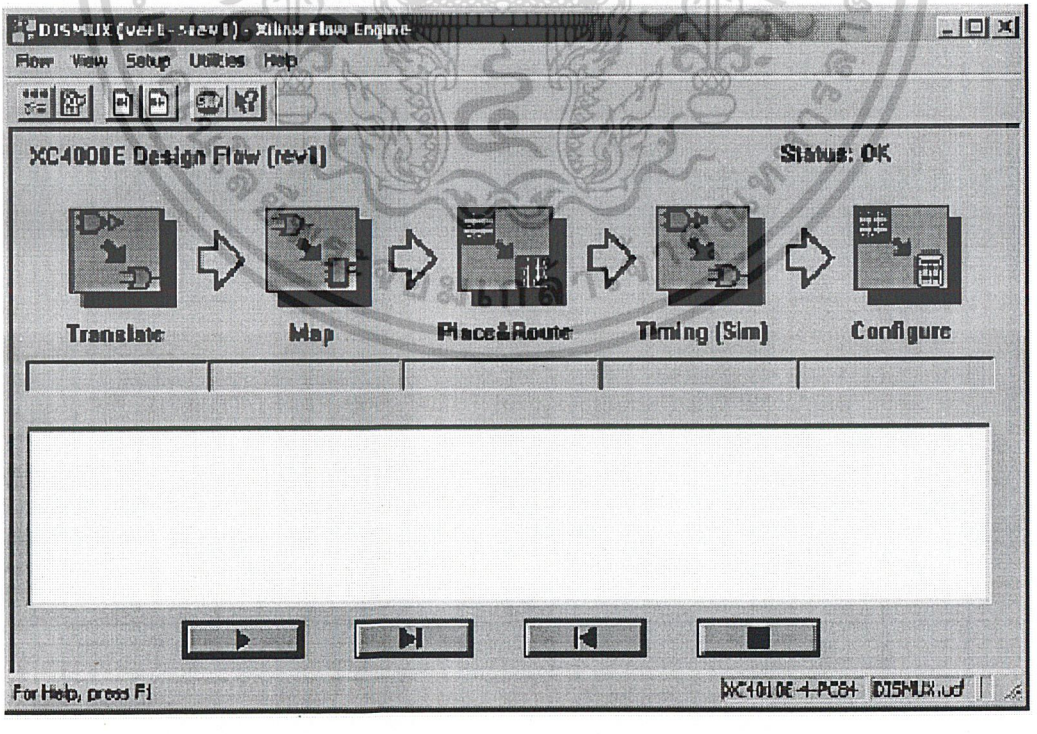
รูปที่4.33 แสดงหน้าต่างของการเลือก Options ในการ Place &Route

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



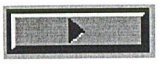
รูปที่4.34 แสดงหน้าต่างของการเลือก Options ในการ Place &Route

4.3.4 คลิกเลือกFlow Engine 



รูปที่4.35 แสดงหน้าต่างของ Flow Engine ก่อนการ runFlow Engine

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จะประกอบด้วยขั้นตอนย่อยดังต่อไปนี้ Translate > Map > Place &Route > Timisng(Sim) > Configure และจะดำเนินการไปตามลำดับ โดยคลิกเริ่มทำงานปุ่มรัน  โปรแกรมก็จะเริ่มทำงานตามขั้นตอนข้างต้น

4.4 การ Translate &Map

Translate คือขั้นตอนที่โปรแกรมอ่านเน็ตลิสต์ไฟล์แบบ XNF ของดีไซน์ตามที่ได้กำหนดไว้แล้วแปลงให้อยู่ในรูปแบบข้อมูลของ XILIINX เอง(ไฟล์ .NGO) โดยใช้คำสั่ง NGDBuild ซึ่งจะแปลงให้เป็นเน็ตลิสต์ แบบ .NGO แล้วผ่านต่อยังขั้นตอน Map ที่เกี่ยวข้องกับการจัดการฮาดแวร์ภายในอุปกรณ์เป้าหมายรวมถึงการจัดแบ่งลอจิกในลอจิกบล็อก อาทิ เช่น CLB (configuration logic block) และ IOB (input output block) และผลการทำงานที่ได้จะถูกเก็บบันทึกไว้ในไฟล์ที่ชื่อว่า map.ncd เพื่อนำไปใช้ในขั้นตอน place & route ต่อไป

หลังจากผ่านขั้นตอน Translateและ Map แล้วอาจจะมีการทำ Timig Analysis เพื่อทราบรายละเอียดเกี่ยวกับความล่าช้า(Delay Time) ของสัญญานอย่างคร่าวๆซึ่งเป็นค่าประมาณเท่านั้น ความล่าช้าของสัญญานอาจแบ่งออกได้เป็นสองจำพวกคือ ความล่าช้าที่เกิดจากการแบ่งลอจิกบล็อก (Block Delay)และความล่าช้าที่เกิดจากการเชื่อมโยงเส้นของสัญญาณระหว่างบล็อกเหล่านั้นในกรณีที่มีการกำหนดข้อบังคับในเรื่องระยะเวลาให้แก่เส้นทางสัญญาณภายในวงจร ค่าความล่าช้าของสัญญาณที่คำนวนได้หลังได้ขั้นตอน Map (post-Map-Deley) ไม่ควรเกิน 50 เปอร์เซนต์ของความล่าช้ารวมอาจจะเกินค่าที่ต้องการก็ได้ และถ้าความล่าช้าจริงมากกว่าค่าที่กำหนดไว้ก็มีทางเลือกสองทางคือ เปลี่ยนแปลงวงจรใหม่ เพื่อให้ได้เส้นทางสัญญาณวิกฤต (Critical path) ที่สั้นลง หรืออีกทางหนึ่งก็คือยอมรับค่าความล่าช้าของสัญญาณที่ได้

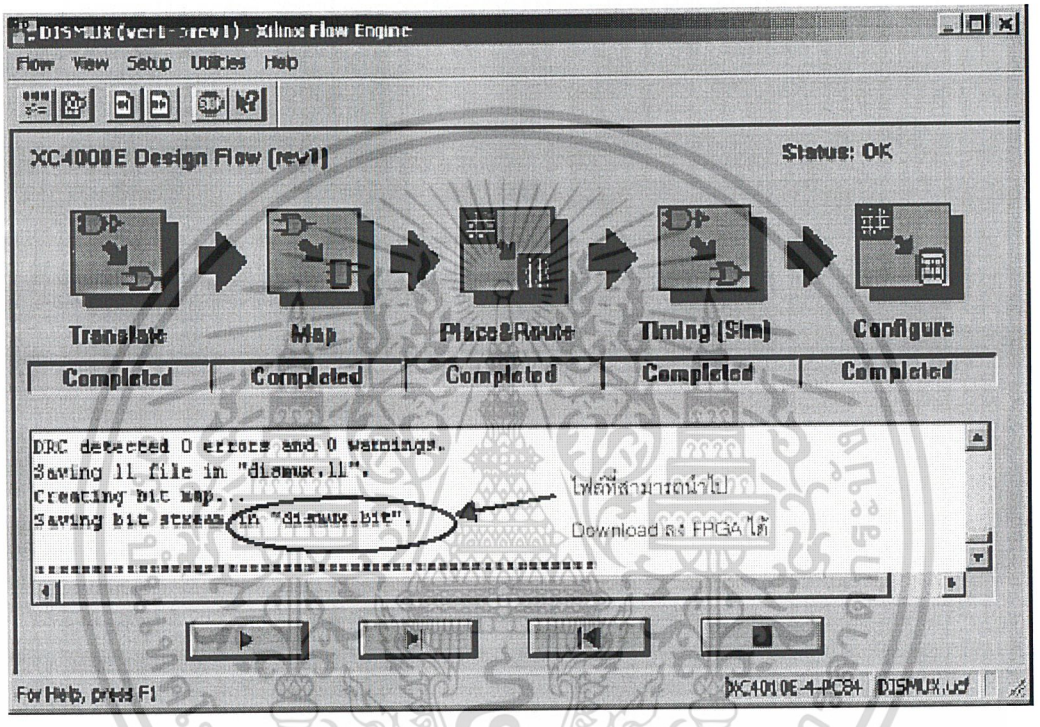
4.5 การ Place & Route

ขั้นตอนการทำ Place & Route เริ่มจากการอ่านข้อมูลของวงจรไฟล์ map.ncd และไฟล์ที่ลงท้ายด้วย .pcf (physical constrain file) ซึ่งภายในมีข้อบังคับต่างๆ ที่โปรแกรม PAR จะต้องคำนึงถึงเมื่อทำขั้นตอน Place & Route ขั้นแรก คือ วางลอจิกบล็อกลงใน FPGA ก่อนซึ่งเป็นหน้าที่ของ placer ส่วน Router จะทำการเชื่อมโยงบล็อกต่างๆ ให้อยู่ในตำแหน่งที่เหมาะสม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.6 การ Configure

เมื่อผ่านขั้นตอน Place & Route เราก็ถือว่าการดีไซน์สำหรับ FPGA เสร็จแล้ว เหลือเพียงขั้นตอนเพียง configure ซึ่งใช้ในการแปลงข้อมูลของวงจรที่ได้ให้อยู่ในรูปแบบของบิทสตรีม (bit stream) เพื่อนำไป download ลง FPGA ต่อไปโดยข้อมูลนี้จะถูกบันทึกลงในไฟล์ที่มีนามสกุล .Bit



รูปที่ 4.36 แสดงหน้าต่างของ Flow Engine หลังการ run

4.6.1 ผลการทำงานของขั้นตอนต่างๆ ใน Flow Engine สามารถตรวจสอบได้จาก คลิกปุ่ม Browse Report  โดยรายงานผลจะถูกแบ่งออกเป็น

Translation Report

Map Report

Place & Route Report

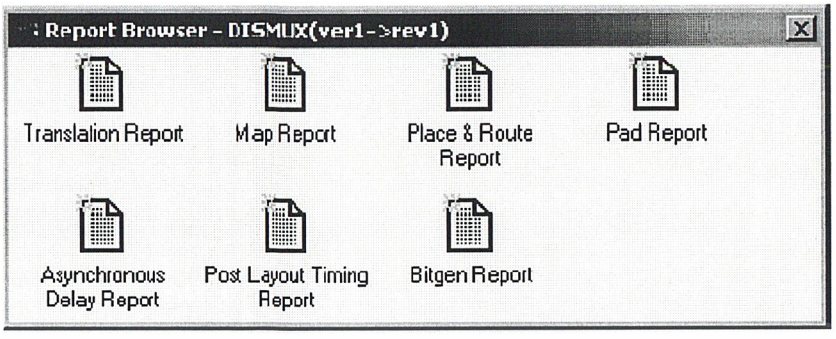
Pad Report

Asynchronous Delay Report

Post Layout Timing Report

Bitgen Report

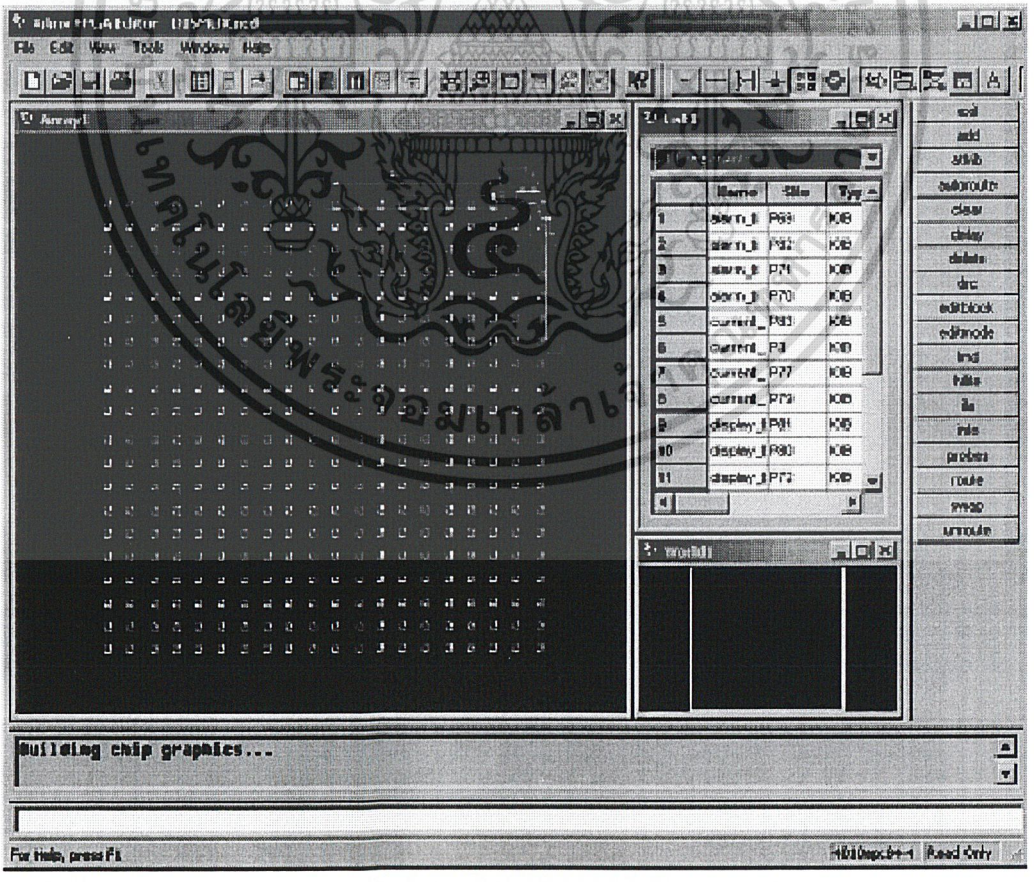
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.37 แสดงหน้าต่างของ Report Browsers

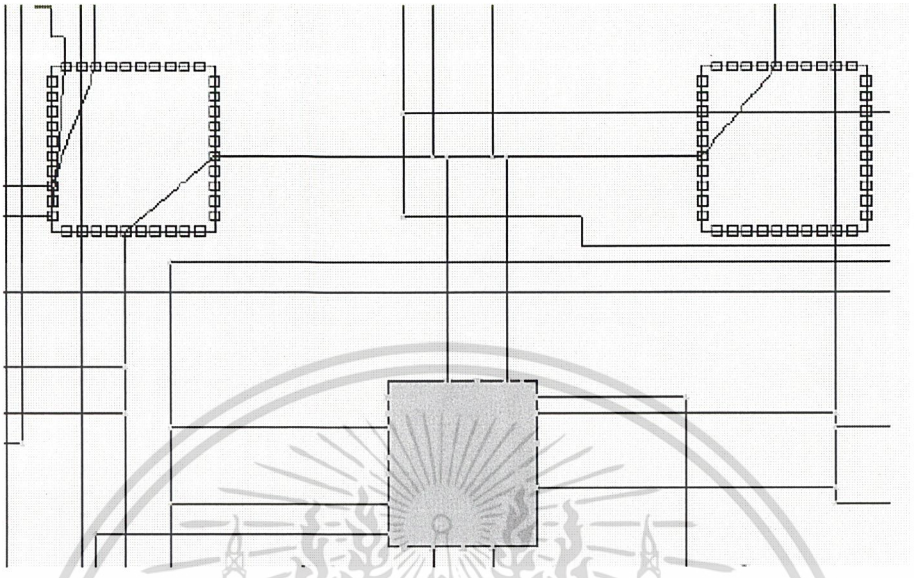
4.6.2 การใช้โปรแกรม FPGA Editor เพื่อตรวจสอบวงจรภายใน FPGA

หลังจากที่ได้ขั้นตอนต่าง ภายใน Flow Engine แล้ว เราสามารถเรียกใช้งานโปรแกรมย่อย อีกตัวหนึ่งก็คือ FPGA Editor โดยคลิกปุ่ม  เพื่อใช้ดู Layout ของวงจรภายใน FPGA แสดงให้เห็นตำแหน่งของลอจิกบล็อก ต่างๆ และ เส้นทางสัญญาณที่เชื่อมโยงภายในวงจร



รูปที่ 4.38 แสดงหน้าต่างของXilinx FPGA Editor

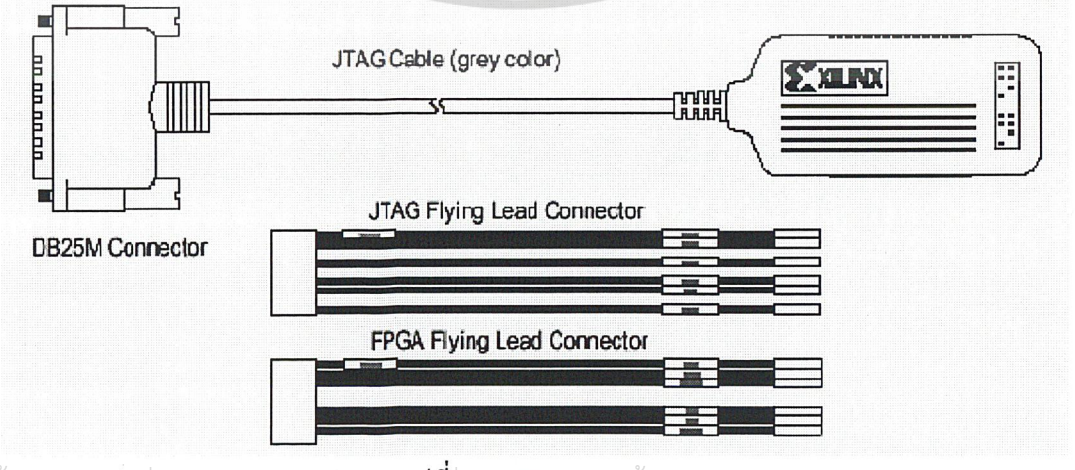
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่4.39 ตัวอย่างพื้นที่บางส่วนของ Layout เมื่อขยายภาพให้ใหญ่ขึ้น

4.7 การใช้งาน Parallel Cable และการโปรแกรม Hardware Debugger

เมื่อผ่านขั้นตอนต่างๆ จาก Flow Engine แล้วผู้ใช้สามารถนำวงจรที่ได้สร้างขึ้นสำหรับ FPGA ไปทดสอบฮาร์ดแวร์จริงได้โดยการ Download ไฟล์ โดยผ่านโปรแกรม Hardware Debugger โดยคลิกปุ่ม  ในโปรแกรม Design Manager เพื่อ Download ข้อมูลลงใน FPGA บนบอร์ดทดสอบ

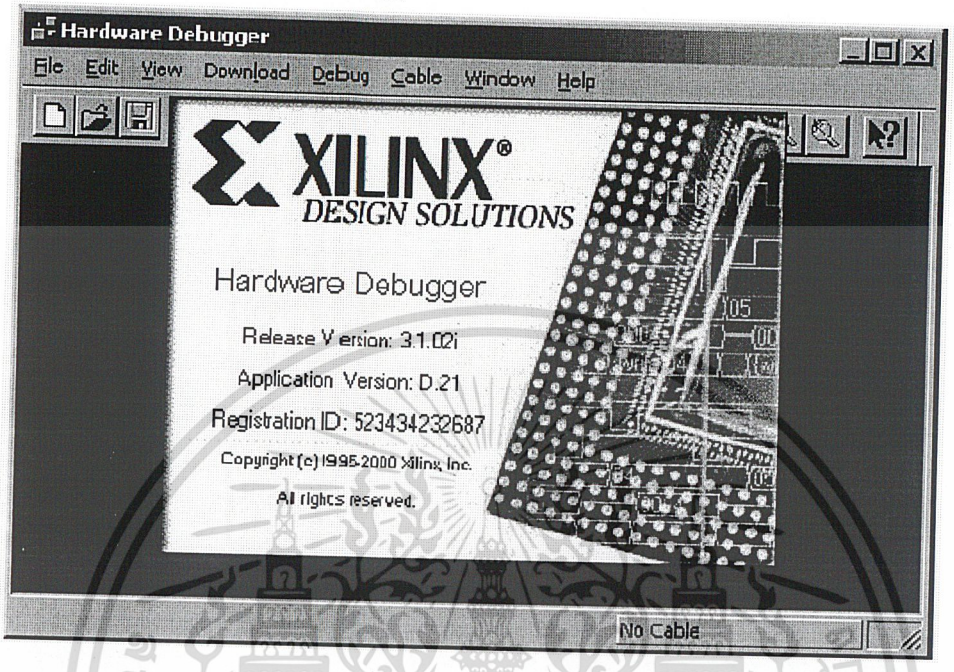


รูปที่ 4.40 JTAG Cable

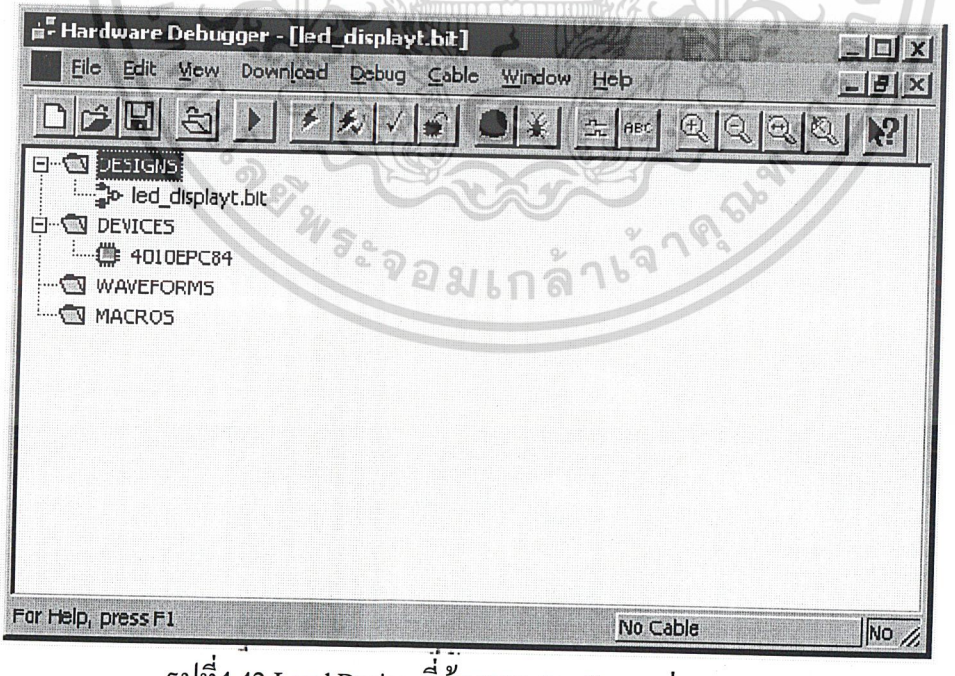
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

-การใช้ Parallel Cable 3 จะต้องมีการเลือก Option การ Load Parallel Cable ในขณะ Install ดังรูป

4.7.1 เรียกโปรแกรม Hardware Debugger

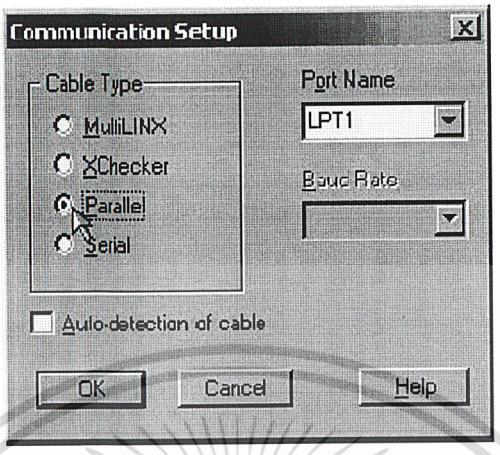


รูปที่4.41 โปรแกรม Hardware Debugger



รูปที่4.42 Load Design ที่ต้องการ Configure สู่ FPGA

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

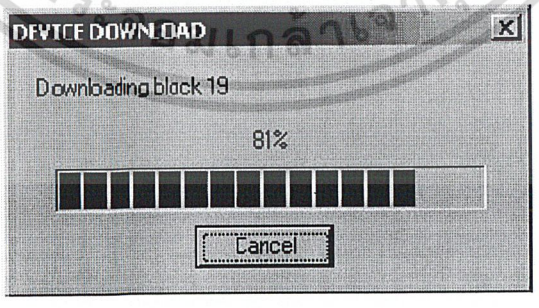


รูปที่4.43 เลือก Parallelและกำหนด Post ที่ Cable ต่ออยู่



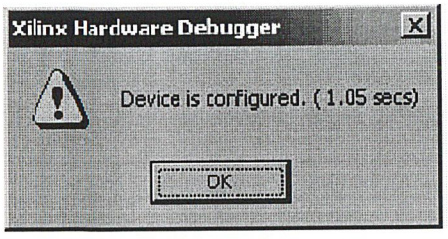
รูปที่4.44 เมื่อ Software สามารถ ติดต่อกับ Cable ได้จะปรากฏสถานการณ์เชื่อมต่อที่มุมล่างด้านขวาของ Hardware Debugger

4.7.2 ต่อสายต่างๆของสาย JTAG Checker กับบอร์ด จากนั้นทดลองทำการ Download วงจรลงสู่ FPGA โดยคลิ๊กปุ่ม  ดังรูป



รูปที่4.45 แสดงสถานะการ Configure การออกแบบสู่ FPGA

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.46 เมื่อ Configure เสร็จสิ้น

ถ้าหากไม่พบปัญหาใดๆโปรแกรมจะแจ้งเวลาที่ใช้ในการ Download วงจรลงสู่อุปกรณ์ FPGA

4.8 การเซ็ต Dipswitch M0 M1 M2 ในการกำหนดการ Download วงจรลง FPGA -ใช้กับ XC4000E Series

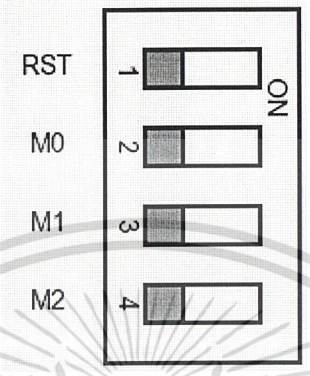
Mode	M2	M1	M0	CCLK	Data
Master Serial	0	0	0	output	Bit-Serial
Slave Serial	1	1	1	input	Bit-Serial
Master Parallel Up	1	0	0	output	Byte-Wide, increment from 00000
Master Parallel Down	1	1	0	output	Byte-Wide, deccrement from 3FFFF
Peripheral Synchronous*	0	1	1	input	Byte-Wide
Peripheral Asynchronous	1	0	1	output	Byte-Wide
Express(XC4000EX only)	0	1	0	Input	Byte-Wide
Reserved	0	0	1	-	-

Note * Peripheral Synchronous can be considered byte-wide Slave Parallel

รูปที่ 4.47 โหมดในการเซ็ต

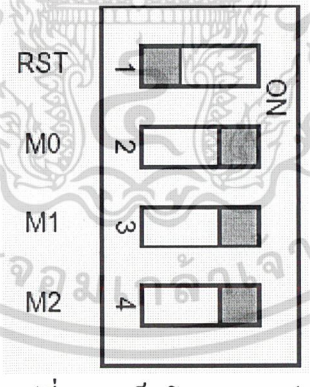
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

-Mater serial mode ซึ่งเป็นโหมดที่ใช้งานร่วมกับ Serial PROM ให้ทำการเซ็ต SW DIPSWITCH M M1 M2 เป็น 000 (0FF=0) เพื่อกำหนดให้ FPGA รับข้อมูลจาก Serial PROM



รูปที่ 4.48 เซ็ตโหมด มาสเตอร์

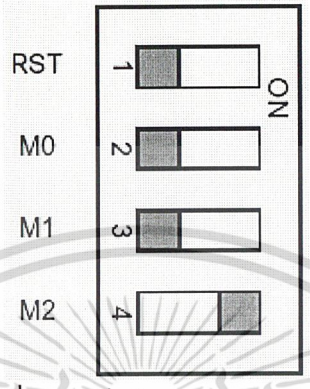
-Slave serial mode ซึ่งเป็นโหมดที่ใช้งานร่วมกับ X-Checker cable ที่ใช้ Download วงจรที่ออกแบบผ่านทาง Serial PORT (COM1,COM2) ให้ทำการเซ็ต SW DIPSWITCH M0 M1 M2 เป็น 111 (ON=1) เพื่อกำหนดให้ FPGA รับข้อมูลจาก X-Checker



รูปที่ 4.49 เซ็ตโหมดสลาฟ

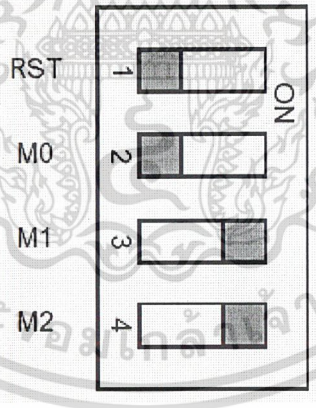
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

-Master parallel up mode ซึ่งเป็นโหมดที่ใช้งานร่วมกับ EPROM เบอร์ 27C256 หรือ 27C512 โดยทำการเซ็ต DIPSWITCH M0 M1 M2 1เป็น 001 (OFF=0,ON=1) เพื่อกำหนดให้ FPGA รับข้อมูลจาก EPROM ที่ถูกเขียนจากแอดเดรส 0x00000 - 0x3FFFF



รูปที่ 4.50 เซ็ตโหมดพาราเรล อัพ

-Master parallel down mode ซึ่งเป็นโหมดที่ใช้งานร่วมกับ EPROM เบอร์ 27C256 หรือ 27C512 โดยทำการเซ็ต DIPSWITCH M0 M1 M2 1เป็น 011 (OFF=0,ON=1) เพื่อกำหนดให้ FPGA รับข้อมูลจาก EPROM ที่ถูกเขียนจากแอดเดรส 0x3FFFF -0x00000



รูปที่ 4.51 เซ็ตโหมดพาราเรล ดาวน์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.9 ตัวอย่างการทดลองแบบพื้นฐาน และผลการทดลอง

4.9.1 การทดลอง ออกแบบ Shift Register

ขั้นตอนการทดลอง

1. เขียน Code VHDL การออกแบบวงจรร่วมวงจร Shift Register
2. ทำการคอมไพล์ โดยใช้โปรแกรม Modelsim
3. เมื่อทำการคอมไพล์จากโปรแกรม Modelsim ผ่านแล้ว นำ Code VHDL ที่ได้ไปทำการสังเคราะห์ โดยใช้โปรแกรม Leonardospectrum เพื่อให้ได้เป็นไพล์ XNF
4. นำไพล์ XNF ไปทำการ place & Route ด้วยโปรแกรม Xilinx Foundation หลังจากทำการ place & Route เราจะได้เป็นไพล์ Bit
5. นำไพล์ Bit ที่ไปทำการ Download ลงอุปกรณ์ FPGA โดยใช้โปรแกรม Hardware Debugger
6. ดูผลการทดลอง จากชุดแสดงผล LED

4.9.2 Code VHDL โดยออกแบบวงจรเป็น Shift Register

```
library ieee;
use ieee.std_logic_1164.all;
entity counter is
port (          LED :out std_logic_vector (7 downto 0));
end counter;
architecture RTL of counter is
component OSC4
                port (  F8M  :  out     std_logic;
                F500K :  out     std_logic;
                F16K  :  out     std_logic;
                F490  :  out     std_logic;
                F15   :  out     std_logic );
end component;
        signal count  : integer range 0 to 15;
        signal q :integer range 0 to 15 ;
```

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

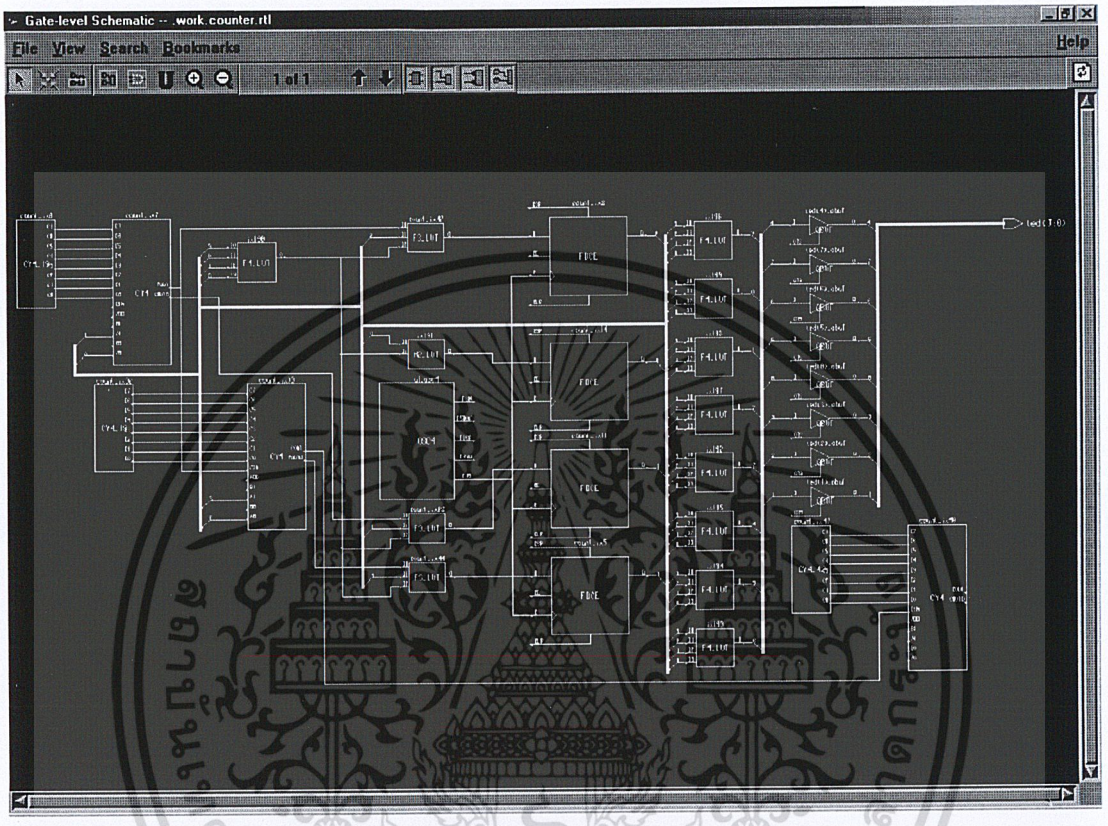
```
        signal OSC15Hz : std_logic;

begin
        U1_OSC4 : OSC4 port map (F15 => OSC15Hz);
        process (OSC15Hz)
        begin
                if OSC15Hz'event and OSC15Hz = '1' then
                  if (count >= 7) then
                        count <= 0;
                  else
                        count <= count +1;
                  end if ;
                end if;
 end process;
 process (count)
        begin
                case count is
                  when 1 => LED <= "00000010";
                  when 2 => LED <= "00000100";
                  when 3 => LED <= "00001000";
                  when 4 => LED <= "00010000";
                  when 5 => LED <= "00100000";
                  when 6 => LED <= "01000000";
                 when 7 => LED <= "10000000";
                  when others => LED <= "00000001";
                end case;
        end process;
end RTL ;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.9.3 ผลการทดลอง

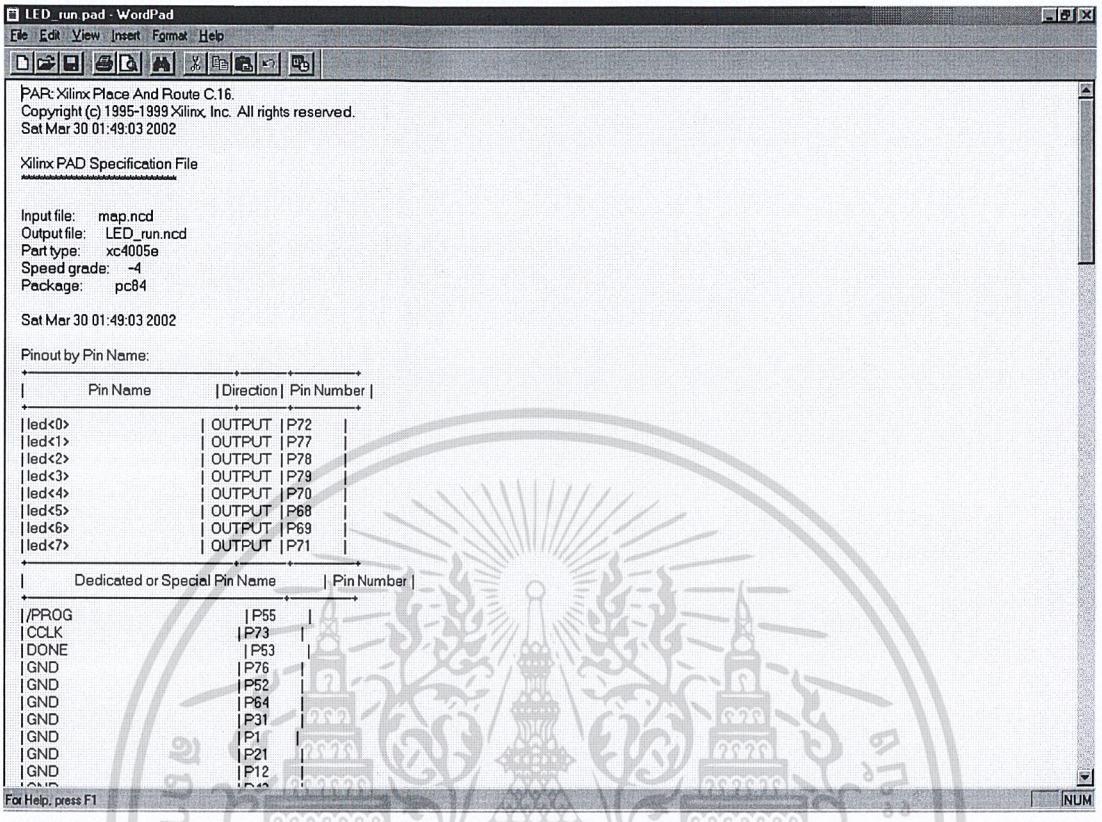
4.9.3.1 ผลการทดลองออกแบบวงจร Shift Register



รูปที่ 4.52 Gate Level Schematic ของวงจร Shift Register

จากรูปเป็นผลที่ได้จากการทดลองจากการสังเคราะห์ได้จากการเขียน Code VHDL วงจรแบบ Shift Register เป็นรูป Gate Level Schematic ซึ่งจะแสดงการทำงานของวงจร โดยจะใช้ Soft Ware ของ Leonardospectrum ในการทำการสังเคราะห์ ซึ่งเราจะได้เป็นไพล์ XNF เพื่อที่ไพล์ที่ดังกล่าวไปทำการ Place & Route เพื่อที่จะหาขาของการออกแบบวงจรใน FPGA และก็ทำการ Download ลง FPGA ต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.53 แสดงขาของ FPGA ของวงจร Shift Register

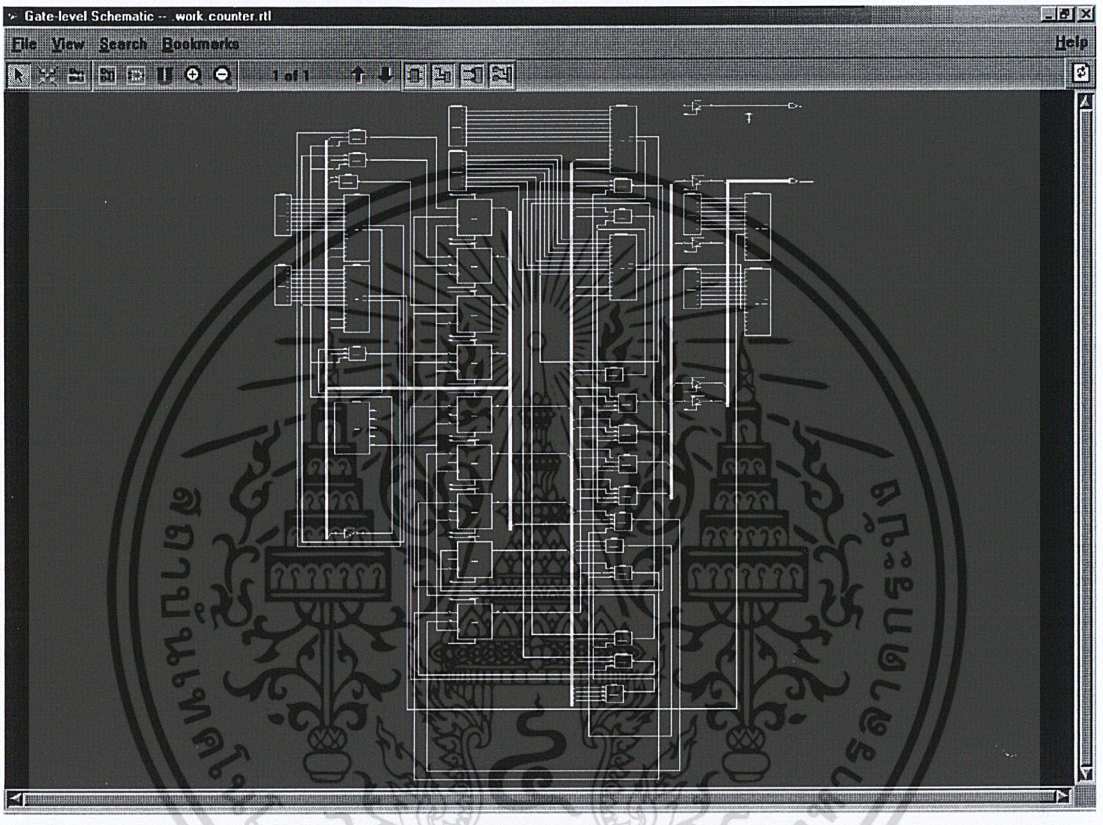
จากรูปเป็นหน้าของการบอกตำแหน่งของขาที่จะทำการ Download ลงสู่ FPGA โดยหน้านี้จะทำการเปิดได้จากโปรแกรม Design Manager เมื่อทำการ Download แล้วเลือก Report Browser จะบอกลายระเอียด ของการ Download แล้วทำการเลือก Pad Report ซึ่งในหน้านี้จะบอกลายระเอียด ต่อสายตามรูปแล้วทำการ Download ลง FPGA ด้วยโปแกรม Hardware Debugger และแสดงผลการทดลอง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.9.3.2 ผลการทดลองออกแบบวงจร Control Stepping Motor

ขั้นตอนการทดลอง

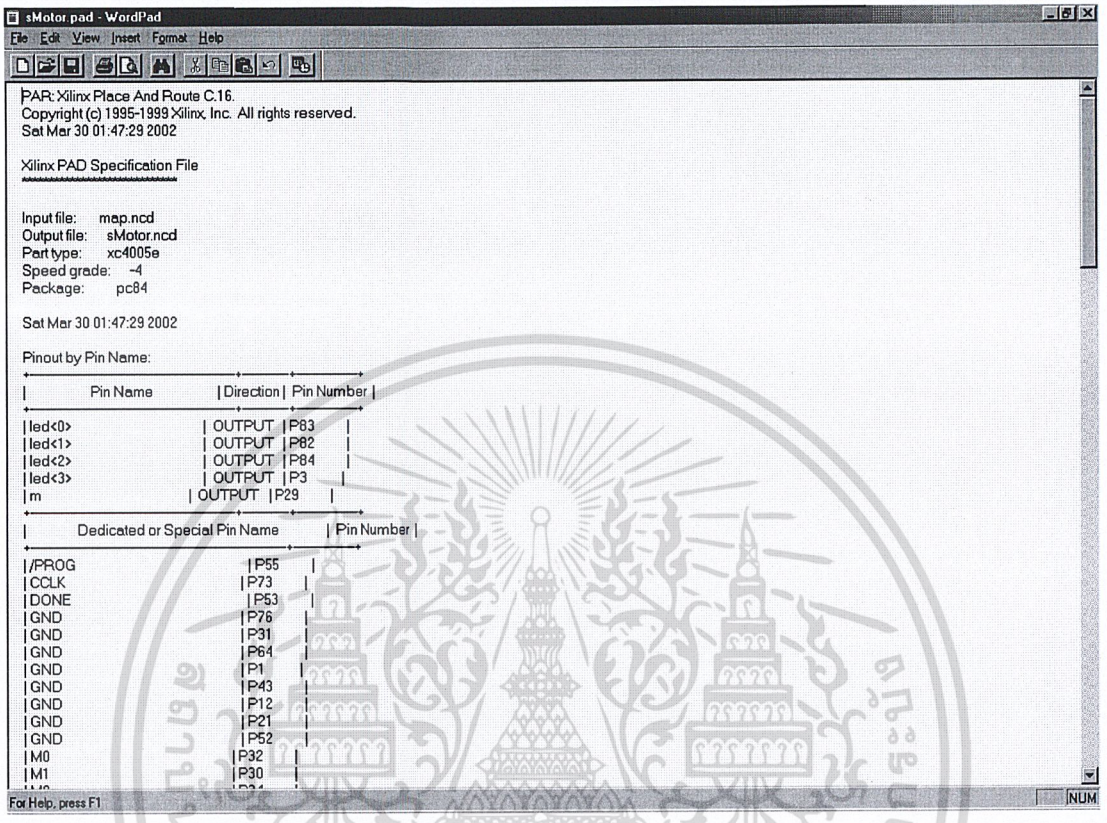
-ทำการทดลองเหมือนขั้นต้น



รูปที่ 4.54 Gate Level Schematic ของวงจร Control Stepping Motor

จากรูปเป็นผลที่ได้จากการทดลองจากการสังเคราะห์ได้จากการเขียน Code VHDL วงจรแบบการ Control Stepping Motor เป็นรูป Gate Level Schematic ซึ่งจะแสดงการทำงานของวงจรโดยจะใช้ Soft Ware ของ Leonardospectrum ในการทำการสังเคราะห์ ซึ่งเราจะได้เป็นไพล์ XNF เพื่อที่ไพล์ที่ดังกล่าวไปทำการ Place & Route เพื่อที่จะหาขาของการออกแบบวงจรใน FPGA และก็ทำการ Download ลง FPGA ต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



sMotor.pad - WordPad

File Edit View Insert Format Help

PAR: Xilinx Place And Route C.16.
Copyright (c) 1995-1999 Xilinx, Inc. All rights reserved.
Sat Mar 30 01:47:29 2002

Xilinx PAD Specification File

Input file: map.ncd
Output file: sMotor.ncd
Part type: xc4005e
Speed grade: -4
Package: pc84

Sat Mar 30 01:47:29 2002

Pinout by Pin Name:

Pin Name	Direction	Pin Number
led<0>	OUTPUT	P83
led<1>	OUTPUT	P82
led<2>	OUTPUT	P84
led<3>	OUTPUT	P3
m	OUTPUT	P29

Dedicated or Special Pin Name	Pin Number
/PROG	P55
CCLK	P73
DONE	P53
GND	P76
GND	P31
GND	P64
GND	P1
GND	P43
GND	P12
GND	P21
GND	P52
M0	P32
M1	P30

For Help, press F1 NUM

รูปที่ 4.55 แสดงขาของ FPGA ของวงจร Control Stepping Motor

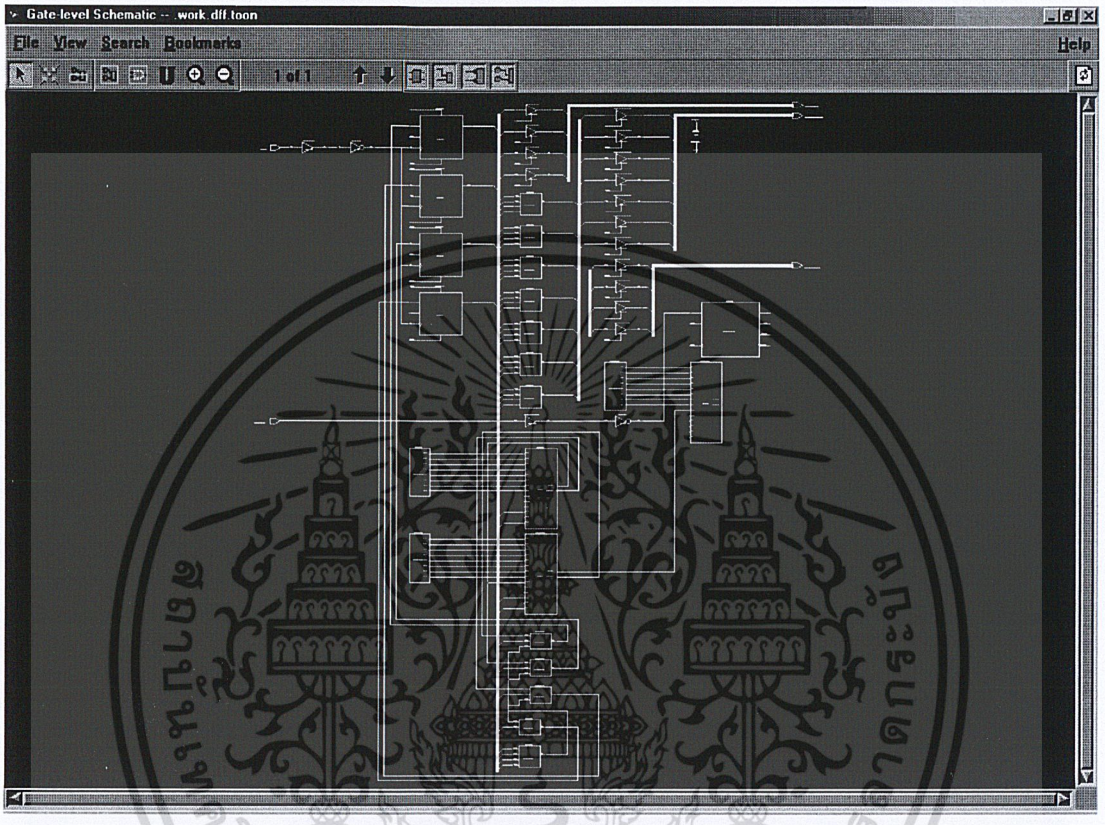
จากรูปเป็นหน้าของการบอกตำแหน่งของขาที่จะทำการ Download ลงสู่ FPGA โดยหน้านี้จะทำการเปิดได้จากโปรแกรม Design Manager เมื่อทำการ Download แล้วเลือก Report Browser จะบอกลายระเอียด ของการ Download แล้วทำการเลือก Pad Report ซึ่งในหน้านี้จะบอกลายระเอียด ต่อสายตามรูปแล้วทำการ Download ลง FPGA ด้วยโปแกรม Hardware Debugger และแสดงผลการทดลอง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.9.3.3 ผลการทดลองออกแบบวงจร Counter

ขั้นตอนการทดลอง

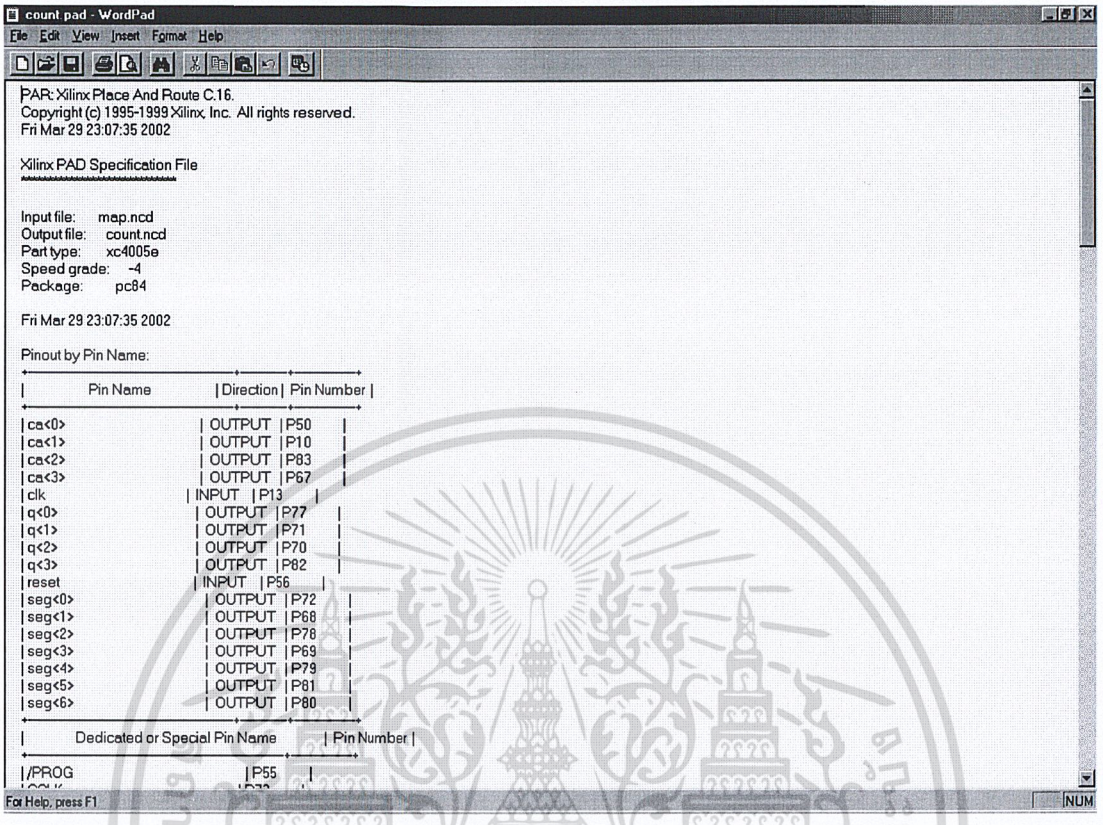
-ทำการทดลองเหมือนขั้นต้น



รูปที่ 4.56 Gate Level Schematic ของวงจร Counter

จากรูปเป็นผลที่ได้จากการทดลองจากการสังเคราะห์ได้จากการเขียน Code VHDL วงจรแบบการ Counter เป็นรูป Gate Level Schematic ซึ่งจะแสดงการทำงานของวงจร โดยจะใช้ Soft Ware ของ Leonardospectrum ในการทำการสังเคราะห์ ซึ่งเราจะได้เป็นไพล์ XNF เพื่อที่ไพล์ที่ดังกล่าวไปทำการ Place & Route เพื่อที่จะหาขาของการออกแบบวงจรใน FPGA และก็ทำการ Download ลง FPGA ต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.57 แสดงขาของ FPGA ของวงจร Counter

จากรูปเป็นหน้าของการบอกตำแหน่งของขาที่จะทำการ Download ลงสู่ FPGA โดยหน้านี้จะทำการเปิดได้จากโปรแกรม Design Manager เมื่อทำการ Download แล้วเลือก Report Browser จะบอกลายระเอียด ของการ Download แล้วทำการเลือก Pad Report ซึ่งในหน้านี้จะบอกลายระเอียด ต่อสายตามรูปแล้วทำการ Download ลง FPGA ด้วยโปแกรม Hardware Debugger และแสดงผลการทดลอง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่5
บทสรุป

ผลงานที่ได้ออกมาก็คิดว่าทำได้ดีระดับหนึ่งแต่ก็ยังไม่ดีที่สุด เพราะเหตุผลหลายๆอย่าง เช่นงบประมาณ,ระยะเวลา ,เครื่องมืออุปกรณ์ที่อำนวยความสะดวกในการทำงาน รวมทั้งตัวผู้สร้างเองก็ยังขาดประสบการณ์ในการทำงานด้านนี้ซึ่งเป็นการเริ่มต้นใหม่ด้วยกันทั้งสองคนจึงมักเจอปัญหาที่เราสองคนไม่คาดคิด และใช้เวลามากในการค้นหาและแก้ไขปัญหาที่เกิดขึ้นแต่ก็ทำให้เราได้ประสบการณ์ใหม่ๆที่ไม่เคยเจอ ซึ่งถ้ามองในภาพรวมแล้วผลงานที่ได้ก็เป็นที่น่าพอใจ และตรงตามจุดประสงค์ที่เราได้ตั้งไว้เราจึงหวังอย่างยิ่งว่า ชุดเครื่องทดลอง FPGA ด้วยภาษา VHDL ของเรานี้น่าจะมีประโยชน์กับผู้ที่สนใจในด้านนี้พอควร

5.1 วิธีการดำเนินงาน

ผู้จัดทำได้แบ่งงานออกเป็น 3 ส่วนดังที่ได้กล่าวไปแล้วข้างต้นเพื่อให้ง่ายต่อการแก้ไขปัญหาเมื่อเจอปัญหาโดยงานจะประกอบด้วย

1. โมดูล สายดาวน์โหลด
2. โมดูลบอร์ด ดาวน์โหลด
3. โมดูลบอร์ด บอร์ดแสดงผล

5.2 ปัญหาและอุปสรรคในการทำงาน

ปัญหาส่วนใหญ่พบในการทำงานคือการใช้โปรแกรมต่างๆที่ใช้ในการทำงาน และมักจะเกิดปัญหาที่คอมพิวเตอร์ Hank

5.3 สิ่งที่ต้องแก้ไขปรับปรุง

ขนาดของ เครื่องทดลองมีขนาดใหญ่ และความซับซ้อนของโปรแกรมการทดลองเป็นเพียงพื้นฐานผู้ที่จะพัฒนาต่อควรสร้าง Application ที่มีความซับซ้อนมากขึ้น

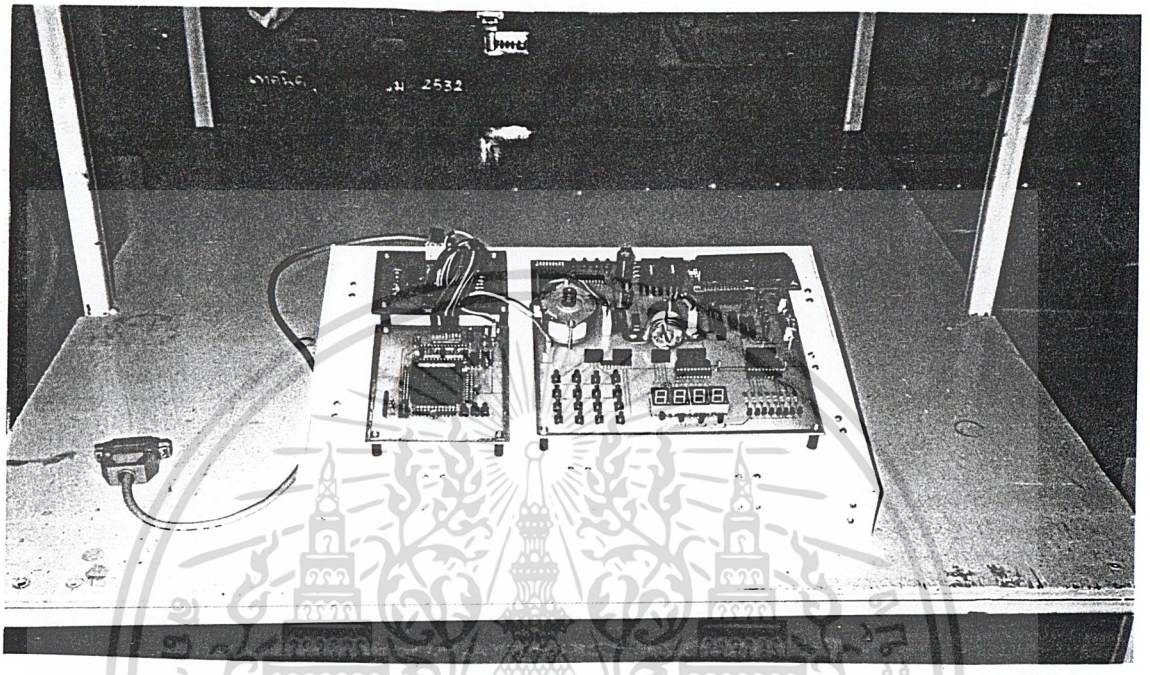
สรุป

ผลงานที่ออกมาก็ตรงตามจุดประสงค์ที่ตั้งไว้ แต่ก็ยังมีจุดบกพร่องที่ยังต้องแก้ไขและพัฒนาให้ดีและสมบูรณ์ยิ่งขึ้นผู้จัดทำจึงหวังว่าผู้ที่สนใจในเรื่องนี้จะพัฒนาต่อหรือสร้างแอปพลิเคชั่นใหม่ๆขึ้นเพื่อเป็นประโยชน์แก่ภาควิชา และประเทศของเราต่อไป และหวังว่าสิ่งที่ทำลงไปจะมีประโยชน์แก่ผู้ที่สนใจขอบคุณครับ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก

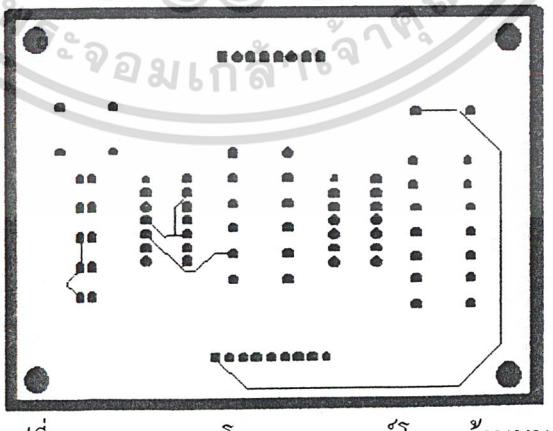
1.ภาพจริงของโครงงาน



รูปที่4.52 ภาพจริงของโครงงาน

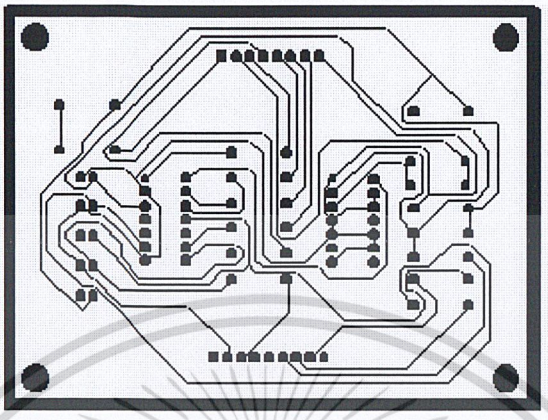
2. ลาย PCB ของโครงงาน

2.1 ลายPCBของโมดูลสายดาวน์โหลด



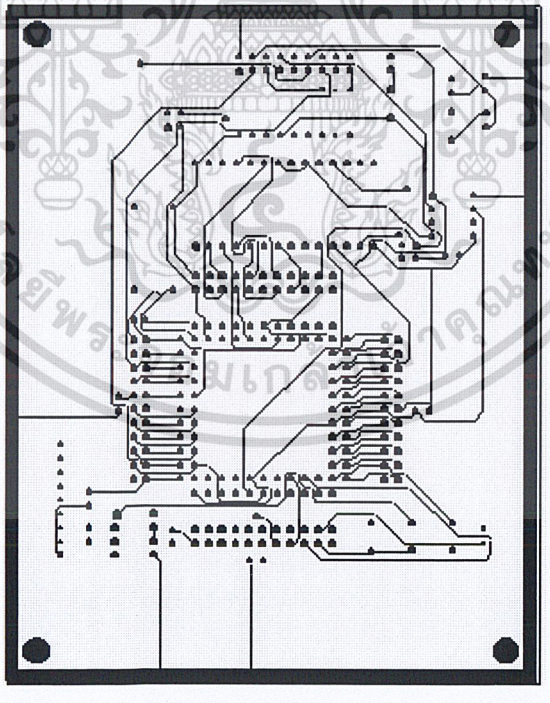
รูปที่ 4.53 ลาย PCB โมดูลสายดาวน์โหลดด้านบน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



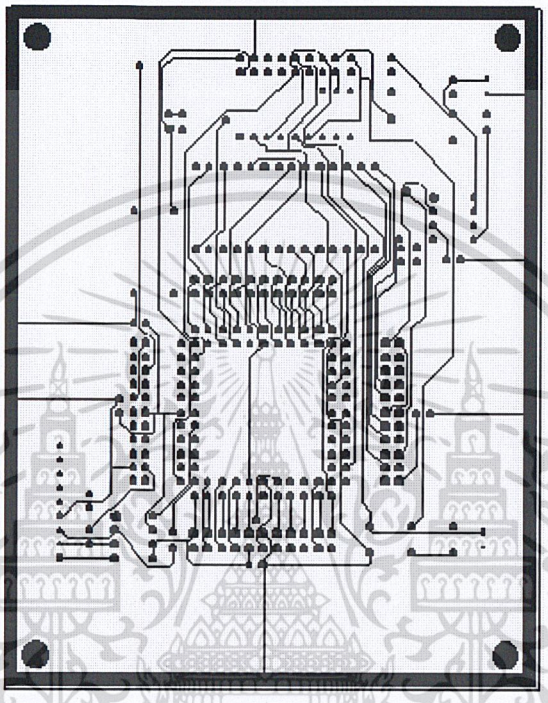
รูปที่ 4.54 ลาย PCB โมดูลสายดาวน์โหลดด้านล่าง

2.2 ลายPCBของโมดูลบอร์ดดาวน์โหลด



รูปที่ 4.55 ลาย PCB โมดูลบอร์ดดาวน์โหลดด้านบน

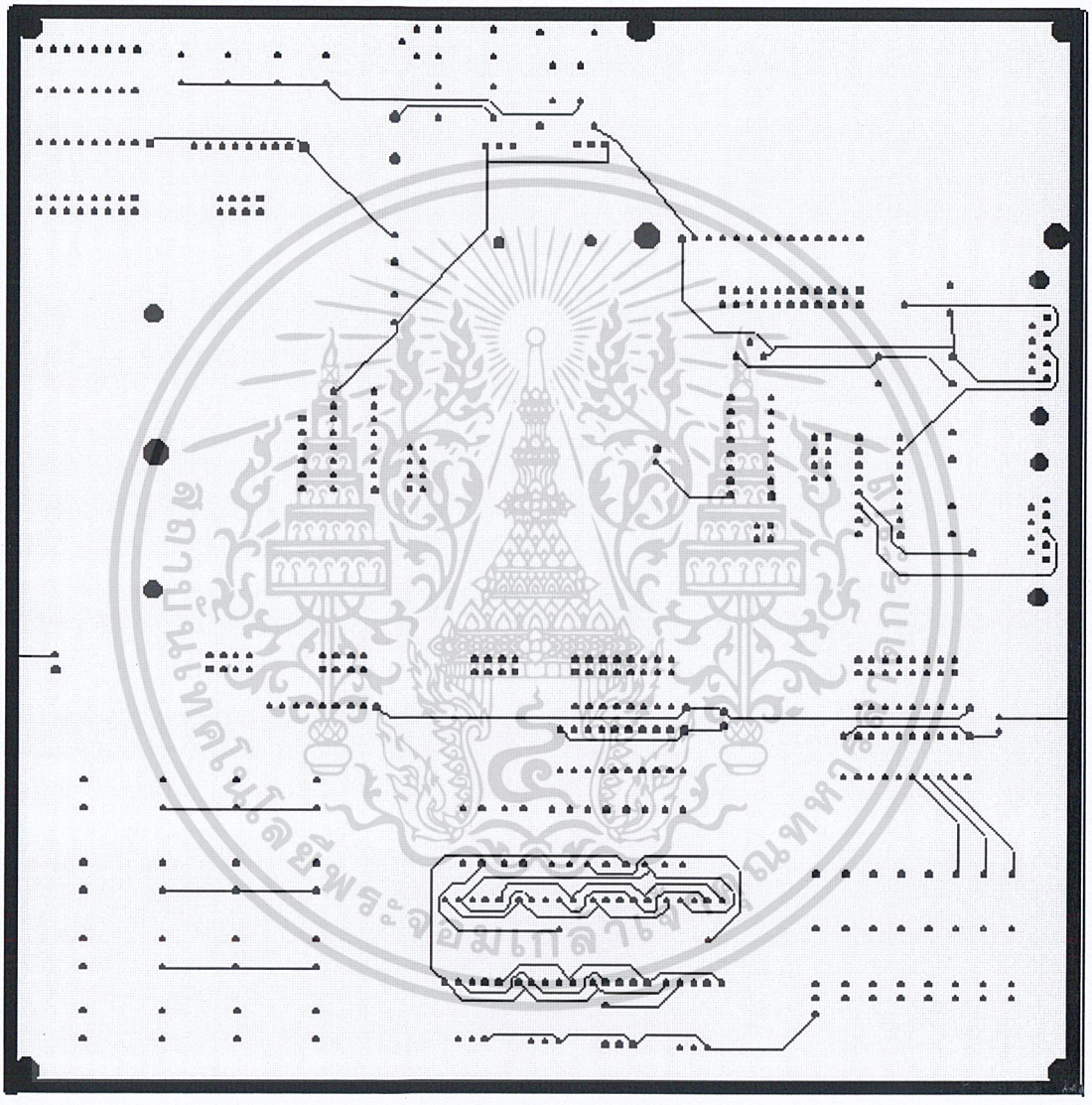
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.56 ลาย PCB โมดูลบอร์ดดาวน์โหลดด้านล่าง

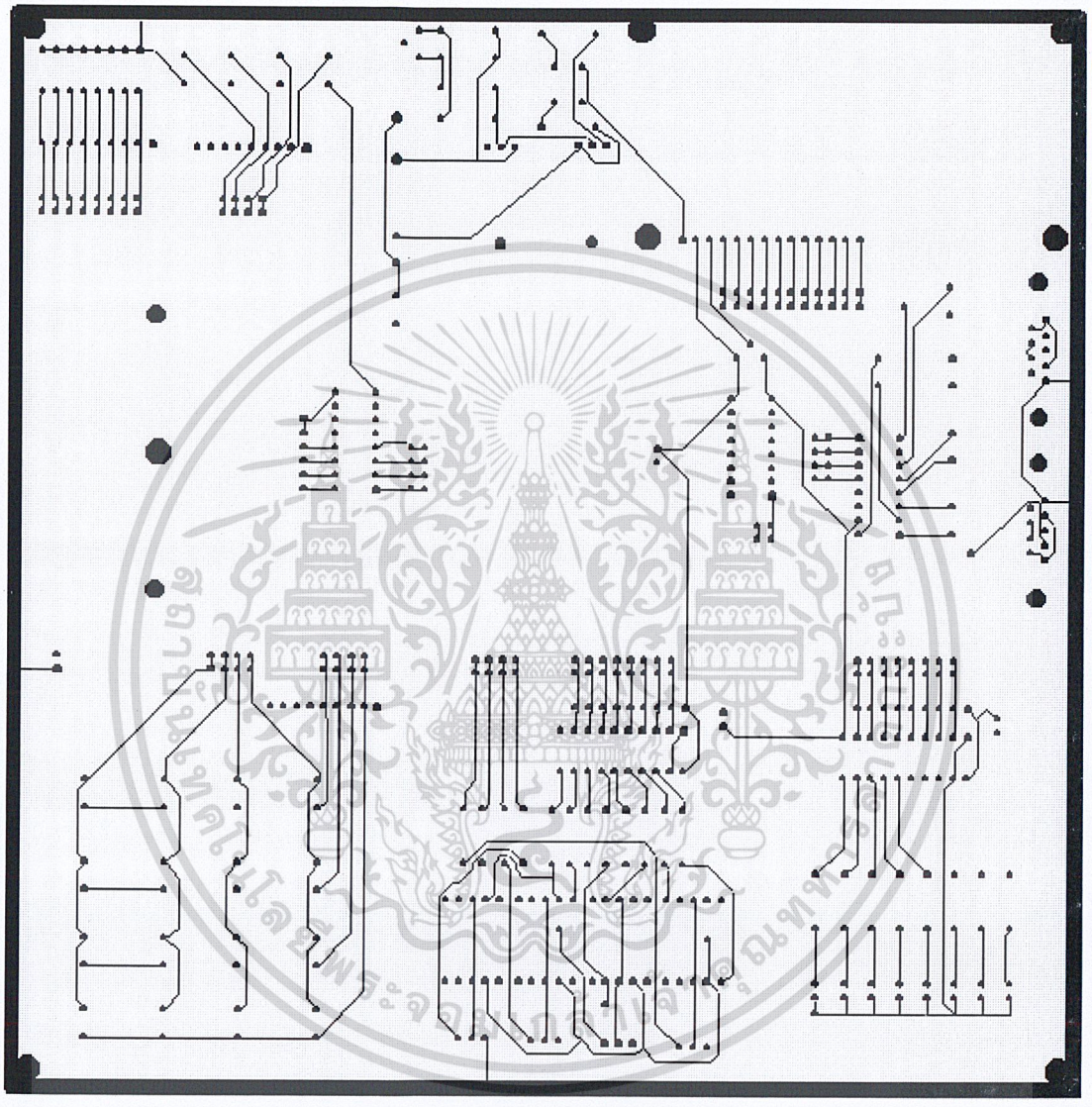
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.3 ลายPCBของโมดูลบอร์ดแสดงผล



รูปที่ 4.57 ลาย PCB โมดูลบอร์ดแสดงผลด้านบน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.58 ลาย PCB โมดูลบอร์ดแสดงผลด้านล่าง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรม Counter

```
library ieee;
use ieee.std_logic_1164.all;
entity dff is
 port (  clk, reset : in std_logic;
         q: out integer range 0 to 15;
         seg : out std_logic_vector ( 6 downto 0);
         ca : out std_logic_vector ( 3 downto 0));
end dff;

architecture toon of dff is
 signal count: integer range 0 to 15;

begin
        process(CLK, RESET)
            begin
            if (reset = '0')then
               count <= 0;
            else
              if (clk'event and clk='0') then
                 if (count >= 9) then
                          count <= 0;
                 else
                          count <= count + 1;
                 end if;
              end if;
           end if;
        end process;
        q <= count;
        ca <= "1110";
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
process (count)
begin
        case count is
                when 0 => seg <= "1111110";
                when 1 => seg <= "0110000";
                when 2 => seg <= "1101101";
                when 3 => seg <= "1111001";
                when 4 => seg <= "0110011";
                when 5 => seg <= "1011011";
                when 6 => seg <= "1011111";
                when 7 => seg <= "1110000";
                when 8 => seg <= "1111111";
                when 9 => seg <= "1111011";
                when others => seg <= "1111110";
        end case;
    end process;
end toon;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรม Control Stepping Motor

```
library ieee;
use ieee.std_logic_1164.all;

entity counter is
port (      M :out std_logic;
            LED :out std_logic_vector (3 downto 0));
end counter;
architecture RTL of counter is
component OSC4
 port (   F8M  : out      std_logic;
          F500K : out     std_logic;
          F16K  : out     std_logic;
          F490  :  out    std_logic;
          F15   :  out    std_logic );
end component;
  signal count  : integer range 0 to 15;
  signal OSC15Hz : std_logic;
  signal a : integer range 0 to 15;
  signal b : integer range 0 to 15;
begin
  U1_OSC4 : OSC4 port map (F15 => OSC15Hz);
  process (OSC15Hz,a,b)
  begin
          if OSC15Hz'event and OSC15Hz = '1' then
            if a = 0 then
              if (count >= 3) then
                  count <= 0;
              else
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
count <= count +1;
if(count = 0) then
  if  b >= 8 then
        a <= 1;
        b <= 0;
  else
        a <= 0;
        b <= b+1;
  end if;
 end if ;
 end if;
end if;
if a = 1 then
 if (count >= 3)then
        count <= 0;
 else
        count <= count +1;
        if(count = 0) then
         if  b >= 8 then
                a <= 0
                b <= 0;
         else
                a <= 1;
                b <= b+1;
          end if;
        end if ;
   end if;
  end if;
 end if;
end process;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
process (count)
begin
        if a = 0 then
                M <= '1';
                case count is
                  when 1 => LED <= "0001";
                  when 2 => LED <= "0010";
                  when 3 => LED <= "0100";
                  when others => LED <= "1000";
                end case;
        elsif a = 1 then
                case count is
                  when 1 => LED <= "1000";
                  when 2 => LED <= "0100";
                  when 3 => LED <= "0010";
                  when others => LED <= "0001";
                end case;
        end if;
end process;
end RTL ;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บรรณานุกรม

1. Milos D. Ercegovac ,Tomas Lang , Jaime H. Moreno : Introduction to Digital Systems :John Wiley & Sons, Inc
2. Jonh F. Wakerly : DIGITAL DESIGN Principles and Practices : Prentice Hall International, Inc.
3. Xilinx Inc : XACT Hardware and Peripherals Guide: Xilinx Inc. 1994
4. Xilinx Inc : The Programmable Logic Data Book : Xilinx Inc. 1994
5. ฝ่ายออกแบบวงจรรวม ศูนย์วิจัยและพัฒนาเทคโนโลยีไมโครอิเล็กทรอนิกส์ ศูนย์เทคโนโลยีอิเล็กทรอนิกส์ และคอมพิวเตอร์แห่งชาติ : FPGA Design Workshop :2540

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้