อุปกรณ์ควบคุม สเตปปิ้งมอเตอร์ ด้วย FPGA

STEPPING MOTOR CONTROLLER WITH FPGA

นายณัฐพันธ์ เนาวเศรษฐ์
นายศักดิ์นรินทร์ เชาวน์ไชยันต์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมการวัดคุม
ภาควิชาวิศวกรรมการวัดคุม คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2548

STEPPING MOTOR CONTROLLER WITH FPGA

NATTAPAN NOUWASAT

SAKNARIN CHAOCHAIYAN

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN INSTRUMENTATION ENGINEERING

DEPARTMENT OF INSTRUMENTATION ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2005

ภาควิชาวิศวกรรมการวัดคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

หัวข้อปริญญานิพนธ์ อุปกรณ์ควบคุมสเตปปิ้งมอเตอร์ ด้วย FPGA

STEPPING MOTOR CONTROLLER WITH FPGA

นักศึกษาผู้จัดทำ นายณัฐพันธ์ เนาวเศรษฐ์ **รหัสประจำตัว** 46015439

นายศักดิ์นรินทร์ เชาวน์ไชยันต์ **รหัสประจำตัว** 46015461

ปริญญา วิศวกรรมศาสตรบัณฑิต

สาขาวิชา วิศวกรรมการวัดคุม

ปีการศึกษา 2548

อาจารย์ผู้ควบคุมปริญญานิพนธ์	**ลายมือชื่อ**
ผศ. เชื้อ นกอยู่	

ภาควิชารับรองแล้ว

(รศ.ประสิทธ์ จุลเสรีวงศ์)

หัวหน้าภาควิชาวิศวกรรมการวัดคุม

หัวข้อปริญญานิพนธ์	อุปกรณ์ควบคุมสเตปปิ้งมอเตอร์ ด้วย FPGA		
	STEPPING MOTOR CONTROLLER WITH FPGA		
นักศึกษาผู้จัดทำ	นายณัฐพันธ์ เนาวเศรษฐ์	รหัสประจำตัว	46015439
	นายศักดิ์นรินทร์ เชาวน์ไชยันต์	รหัสประจำตัว	46015461
อาจารย์ที่ปรึกษา	ผศ.เชื้อ นกอยู่		
ปีการศึกษา	2548		

บทคัดย่อ

โครงงานนี้เป็นการเสนอการนำ FPGA (Field-Programmable Gate Array) มาใช้ในการ Drive Stepping Motor เพื่อทำการควบคุมแรงบิดและความเร็วรอบ ในรูปแบบการ Drive แบบ Full Step และ Half Step ซึ่งความถี่ที่ใช้ในแบบ Full Step ใช้ในช่วง 10Hz-100Hz และแบบ Half Step ใช้ในช่วง 10Hz-500Hz ซึ่งการออกแบบโปรแกรมใช้ภาษาอธิบายพฤติกรรม (VHSIC Hardware Description Language) ของฮาร์ดแวร์ หรือเรียกว่า VHDL ซึ่ง FPGA ที่ใช้อยู่ในตระกูล XC3000 ของบริษัท Xilinx และใช้โปรแกรม ISE WebPACK เป็นโปรแกรมที่ใช้ในการออกแบบในการเขียนภาษา VHDL และใช้ Modelsim XE ในการจำลองการทำงานของโปรแกรมที่ทำการออกแบบบน ISE WebPACK เพื่อทำการเขียนโปรแกรมที่ผ่านการตรวจสอบเสร็จแล้วลง FPGA

Thesis Title	Stepping Motor Controller with FPGA	
Authors	Mr. Nattapan	Nouwasat
	Mr. Saknarin	Chaochaiyan
Thesis Advisor	Asst.Prof. Chuae	Nokyoo
Year	2005	

ABSTRACT

This project performs FPGA (Field-Programmable Gate Array) to use drive stepping motor for control torque and speed (rpm) in drive pattern Full step and Half step. Frequency use in pattern Full step use in range 10 Hz-100 Hz and pattern Half step use in range 10 Hz-500 Hz . Program design by using hardware description language of hardware or called VHDL . FPGA in use now is the family of XC3000 of the Xilinx Company and ISE Webpack Program designed in writing VHDL Language and using ModelSim XE in simulating the program. When this program is finished , it will pass FPGA.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยดีเพราะได้รับความเมตตาจาก ผู้ช่วยศาสตราจารย์เชื้อ นกอยู่ และ รองศาสตราจารย์ฟูศักดิ์ ชีวสุวิทย์ ที่ได้ให้คำแนะนำแก่ผู้วิจัยตลอดมา อีกทั้งยังเอื้อเฟื้ออุปกรณ์และเครื่องมือต่างๆ ในการทำปริญญานิพนธ์นี้ ผู้วิจัยรู้ซึกซาบซึ้งและขอกราบขอบพระคุณเป็นอย่างสูง

ขอขอบพระคุณ อาจารย์ภาควิชาวิศวกรรมการวัดคุมทุกท่าน ที่ให้คำแนะนำอันเป็นประโยชน์ต่อการทำปริญญานิพนธ์ฉบับนี้

และที่ลืมเสียมิได้คือ ขอกราบขอบพระคุณคุณแม่ อันเป็นทีรักยิ่ง ที่สนับสนุนและเป็นแรงบันดาลใจในการทำปริญญานิพนธ์ฉบับนี้

คุณค่าและประโยชน์อันพึงมีจากปริญญานิพนธ์ฉบับนี้ ผู้วิจัยขอมอบแด่ผู้มีพระคุณทุกท่าน

คณะผู้จัดทำ

สารบัญ

สารบัญ(ต่อ)

สารบัญ(ต่อ)

สารบัญตาราง

สารบัญภาพ

สารบัญภาพ(ต่อ)

สารบัญภาพ(ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและเหตุจูงใจของการวิจัย

การออกแบบวงจรทางดิจิตอลในอดีต ผู้ออกแบบต้องนำเกทดิจิตอลชนิดต่างๆ เช่น AND, OR, NOR, NOT และ FLIP FLOP ที่อยู่ในรูปของ IC สำเร็จรูปมาต่อร่วมกันเป็นวงจรที่ทำหน้าที่ตามต้องการ ด้วยวิธีการออกแบบในลักษณะนี้ หากวงจรที่ต้องการมีความซับซ้อนมากขึ้น จำนวนเกทดิจิตอลที่นำมาประกอบย่อมมีมากขึ้นตามไปด้วย ดังนั้นวงจรที่สร้างขึ้นจะมีขนาดใหญ่ นอกจากนี้ผู้ออกแบบยังต้องคำนึงถึงปัญหาของสัญญาณรบกวนที่จะเกิดขึ้นในการออกแบบทองแดงอีกด้วย จากความยุ่งยากดังกล่าวทำให้มีการพัฒนาวิธีการออกแบบวงจรทางดิจิตอลเรื่อยมา กระทั่งถึงปัจจุบันการออกแบบวงจรทางดิจิตอลจะเป็นการออกแบบในลักษณะที่ใช้ภาษาอธิบายพฤติกรรมของวงจร (Schematic) ผสมกับภาษาคอมพิวเตอร์แล้วใช้ซอฟต์แวร์สังเคราะห์ลอจิก (Logic Synthesis Tools) แปลงให้เป็นข้อมูลที่สามารถนำไปโปรแกรมลงในชิพไอซีที่สามารถโปรแกรมได้ เพื่อสร้างเป็นวงจรที่ต้องการ

1.2 วัตถุประสงค์ของปริญญานิพนธ์

1. ศึกษาสถาปัตยกรรมภายในของ FPGA
2. ศึกษาการออกแบบ FPGA โดยใช้ภาษา VHDL
3. ศึกษาและพัฒนา FPGA เพื่อนำไปควบคุม Stepping Motor

1.3 ขอบเขตของปริญญานิพนธ์

1. ทำการออกแบบจำลองโปรแกรมควบคุม Stepping Motor ด้วยภาษา VHDL
2. ทำการสร้างวงจรขับ Stepping Motor แบบสองแหล่งจ่าย
3. สามารถนำ FPGA ที่ทำการออกแบบ มาขับ Stepping Motor ได้ และบันทึกผลและเก็บค่าให้อยู่ในรูปของสัญญาณที่จุดต่างๆได้

1.4 ขั้นตอนการศึกษา

1. ศึกษาหลักการทำงานของ Stepping Motor และการขับสเตปของ Stepping Motor
2. สร้างวงจรสองแหล่งจ่ายเพื่อนำไปขับ Stepping Motor
3. ศึกษาสถาปัตยกรรมภายในของ FPGA

4. ศึกษาการออกแบบ FPGA โดยใช้ภาษา VHDL โดยใช้โปรแกรม ISE WebPACK เป็นตัวออกแบบในการเขียนภาษา VHDL และใช้ ModelsimXE ในการจำลองทำงานของโปรแกรมที่ทำการออกแบบบน ISE WebPACK

5. ทำการลงโปรแกรม ลงบน FPGA และนำไปขับ Stepping Motor

6. ทำการวัดสัญญาณสเตปที่ FPGA สร้างขึ้น ณ จุดต่างๆในรูปแบบของสัญญาณพัลส์

7. ทำการวัดค่าแรงบิท (Torque) และความเร็วรอบ (rpm) ที่ความถี่ต่างๆ แล้วนำไปทำการเขียนกราฟแสดงความสัมพันธ์ระหว่างค่าความถี่ แรงบิท และความเร็วรอบ ทั้งแบบ Full Step และแบบ Half Step

บทที่ 2
สเตปปิ้งมอเตอร์

2.1 กล่าวนำ

สเตปปิ้งมอเตอร์เป็นมอเตอร์ไฟฟ้ากระแสสลับที่มีโรเตอร์เป็นแม่เหล็กถาวรและใช้การกระตุ้นด้วยแม่เหล็กไฟฟ้าจากขดลวดที่สเตเตอร์ โดยแบ่งชุดขดลวดที่สเตเตอร์ออกเป็นเฟสต่าง ๆ

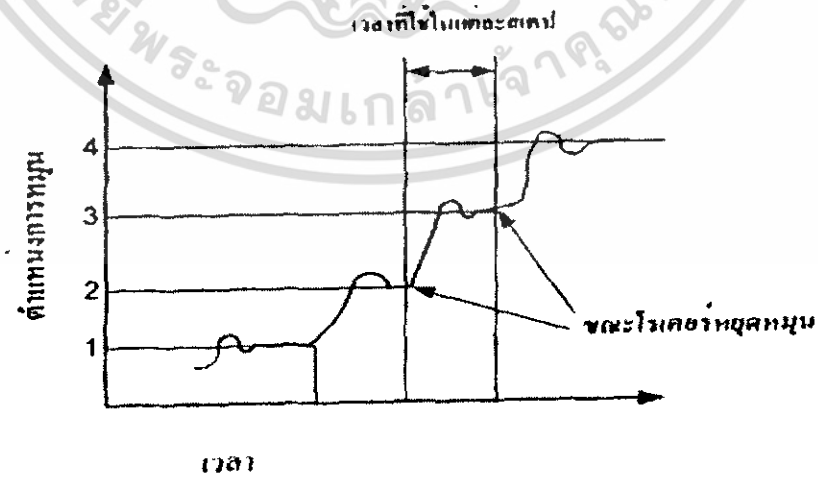
การขับสเตปปิ้งมอเตอร์ใช้วงจรขับควบคุมการจ่ายกระแสไฟฟ้าให้กับขดลวดตัวนำในแต่ละเฟสเพื่อสร้างสนามแม่เหล็กไฟฟ้าให้สอดคล้องกับการหมุนของโรเตอร์ที่เป็นแม่เหล็กชนิดถาวร

2.2 หลักการทำงานของสเตปปิ้งมอเตอร์

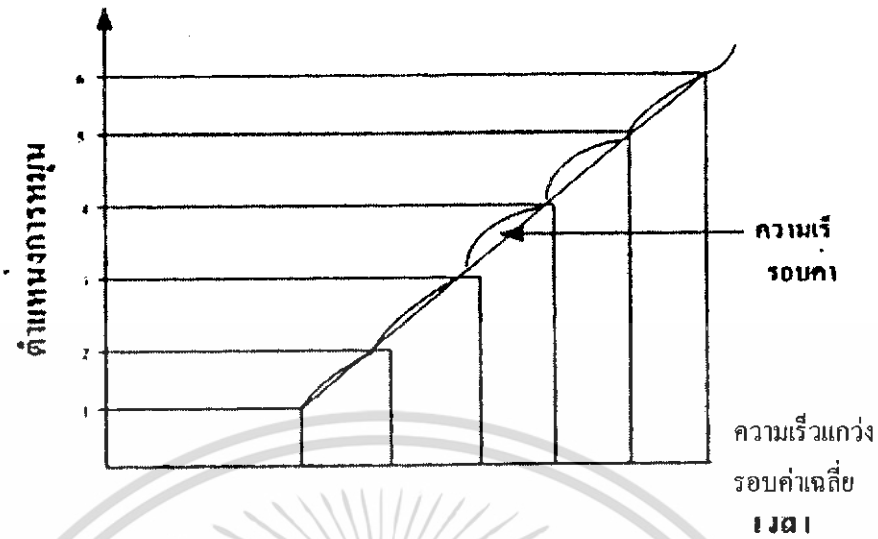
สเตปปิ้งมอเตอร์เป็นมอเตอร์ไฟฟ้ากระแสสลับที่เปลี่ยนสัญญาณดิจิตอลเป็นการเคลื่อนที่ทางกล จึงเหมาะสำหรับการเชื่อมต่อกับอุปกรณ์ทางดิจิตอลหรือคอมพิวเตอร์ โดยการทำงานของสเตปปิ้งมอเตอร์ โดยส่วนใหญ่จะขึ้นอยู่กับสัญญาณพัลส์ (PULSE) กระตุ้นที่ป้อนให้กับขดลวดของเฟสของมอเตอร์ในลำดับที่ถูกต้องด้วยวงจรลำดับลอจิกและกระแสที่เพียงพอเพื่อป้อนให้กับวงจรขับ

การทำงานของสเตปปิ้งมอเตอร์ใช้อัตราเร็วของแต่ละสเตปนั้น สามารถจะแบ่งออกเป็นลักษณะของ 2 โหมดการทำงานคือ

1. โหมดของการหมุนเป็นสเตป (Discrete Mode)
2. โหมดของการหมุนแบบต่อเนื่อง (Slewing Mode)



ภาพที่ 2.1 แสดงการหมุนเป็นสเตป



ภาพที่ 2.2 แสดงการหมุนแบบต่อเนื่อง

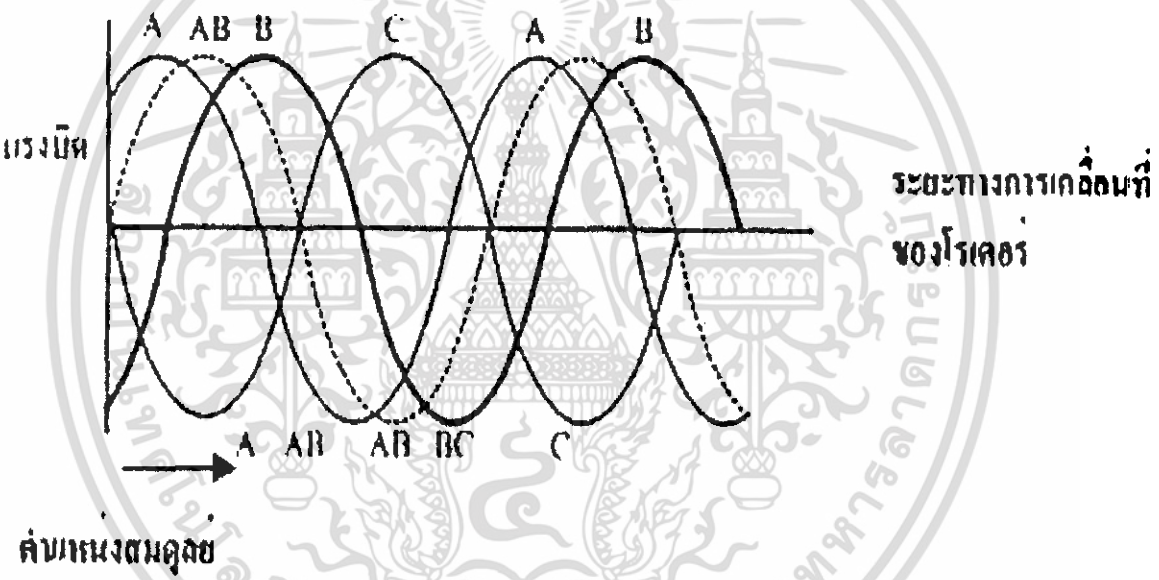
ลักษณะการหมุนแบบสเตปจะมีช่วงเวลาหยุดนิ่งของโรเตอร์ก่อนที่จะเปลี่ยนสเตปถัดไป ดังนั้นหากมีการเพิ่มอัตราเร็วในแต่ละสเตปให้เร็วขึ้นและเป็นไปอย่างต่อเนื่อง การหมุนของมอเตอร์ก็จะเปลี่ยนเป็นการหมุนแบบต่อเนื่อง

การที่นำสเตปปิ้งมอเตอร์ไปใช้งานไม่ว่าจะเป็นโหมดการทำงานแบบใดหรือเป็นมอเตอร์แบบไหนควรจะมีการศึกษาคุณลักษณะต่าง ๆ เช่น แรงบิด การตอบสนองต่อความถี่สูงสุดและความสัมพันธ์ระหว่างการเปลี่ยนแปลงของแรงบิดกับค่าของกระแสทั้งนี้เพื่อให้สามารถควบคุมการทำงานของสเตปปิ้งมอเตอร์ได้อย่าง มีประสิทธิภาพมากที่สุดซึ่งคุณลักษณะเหล่านี้สามารถแบ่งออกได้เป็นสองลักษณะคือ

1. คุณลักษณะทางสถิตย์ (Static Characteristics)
2. คุณลักษณะทางผลวัต (Dynamic Characteristics)

2.3 คุณลักษณะทางสถิตย์

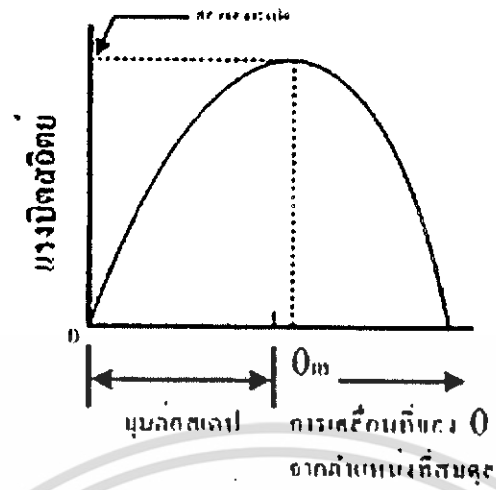
คุณลักษณะทางสถิตย์ หมายถึง คุณลักษณะในสภาวะที่มอเตอร์หยุดนิ่งไม่มีการเคลื่อนไหว ซึ่งจะเป็นตำแหน่งที่มีความสมดุลของการหมุนมากดังที่สุดกล่าวคือเป็นตำแหน่งที่ซี่ฟันของโรเตอร์และซี่ฟันของสเตเตอร์มีความสัมพันธ์กันอยู่ในแนวเดียวกัน และยังคงอยู่ในตำแหน่งนี้จนกว่าจะมีการกระตุ้นอีกครั้ง โดยที่ตำแหน่งสมดุลของโรเตอร์จะมีค่าของแรงบิดเท่ากับศูนย์ ซึ่งถ้าโรเตอร์มีการเคลื่อนที่ในทิศทางใดทิศทางหนึ่ง มอเตอร์ก็จะทำการสร้างสมดุลให้กับโหลดโดยการสร้างแรงบิดขึ้นมาในทิศทางตรงกันข้ามกับทิศทางการเคลื่อนที่ ผลของแรงบิดกับการเคลื่อนที่ของโรเตอร์จะมีลักษณะคล้ายกับรูปคลื่นซายน์



ภาพที่ 2.3 กราฟคุณลักษณะทางสถิตย์ของสเตปปิ้งขนาดสามเฟส

แรงบิดสูงสุดที่เกิดขึ้นอยู่กับกระแสที่จ่ายเฟสนั้น ๆ โดยที่แรงบิดนี้เรียกว่าสภาพคงแรงบิด (Holding Torpue) หรือแรงบิดสถิตย์ (Static Torque) โดยลักษณะความสัมพันธ์ระหว่างค่าของแรงบิดกับกระแสที่เปลี่ยนแปลงไปจะเป็นลักษณะเชิงเส้น แต่ก็จะมีขีดจำกัดอยู่ที่จุดอิ่มตัวแม่เหล็กของทั้งโรเตอร์และสเตเตอร์ซึ่งมีโครงสร้างเป็นแม่เหล็กโดยที่จุดอิ่มตัวคือจุดที่ไม่มีการเพิ่มขึ้นของแรงบิดแม้ว่าแสงความสัมพันธ์ดังต่อไปนี้

1. คุณลักษณะของแรงบิดกับมุมสเตป
2. คุณลักษณะของแรงบิดกับกระแส



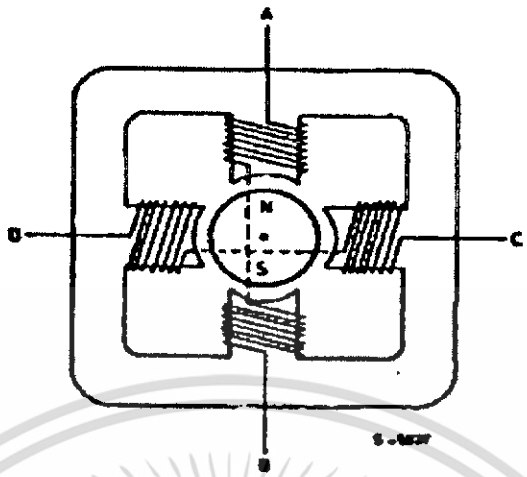
ภาพที่ 2.4 คุณลักษณะของแรงบิดต่อมุมสเตป

2.4 สเตปปิ้งมอเตอร์แบบแม่เหล็กถาวาร

สเตปปิ้งมอเตอร์ชนิดนี้จะใช้แม่เหล็กถาวรเป็นโรเตอร์โดยที่ตัวของโรเตอร์รูปร่างเป็นทรงกระบอก โรเตอร์และสเตอร์จะถูกแบ่งออกเป็นซี่ๆ หรือที่เรียกว่าเฟสล้อมรอบตามภาพที่ 2.5 โดยในแต่ละซี่นั้นจะมีขดลวดพันโดยรอบๆ เพื่อทำให้เกิดการกระตุ้นในการสร้างสนามแม่เหล็ก การให้สเตปปิ้งมอเตอร์แบบแม่เหล็กถาวรนั้นมีขนาดมุมสเตป(step)เล็กลงสามารถทำให้ได้โดยการเพิ่มจำนวนของขั้วแม่เหล็กของโรเตอร์และหรือจำนวนซี่ฟันของสเตเตอร์ แต่การเพิ่มความละเอียดของสเตปก็มีขีดจำกัดในการเพิ่มของจำนวนขั้วแม่เหล็กของโรเตอร์ ทั้งนี้เนื่องการสร้างแม่เหล็กถาวรให้มีโครงสร้างแบบมีขั้วแม่เหล็กหลายๆขั้วนั้นทำได้ยากจึงกลายเป็นข้อเสียของสเตปปิ้งมอเตอร์ ซึ่งในลักษณะนี้มีข้อเสียคือ

1. มีขนาดของมุมสเตปใหญ่จะทำให้มีความละเอียดของมุมสเตปต่อรอบน้อยมาก เนื่องจากว่าโครงสร้างของโรเตอร์เป็นแม่เหล็กถาวร

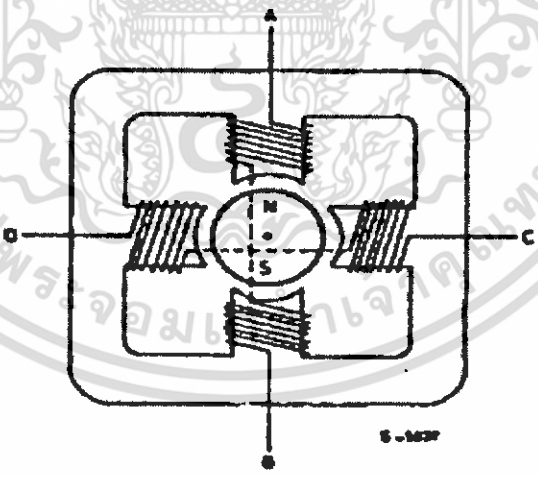
2. สเตปปิ้งมอเตอร์แบบแม่เหล็กถาวรส่วนใหญ่จะมีโครงสร้างขนาดเล็กทำให้ค่าของแรงบิดที่ได้ต่อหน่วยปริมาตรมีค่าต่ำ ถ้าต้องการปรับปรุงประสิทธิภาพในเรื่องของแรงบิดแม่เหล็กถาวรที่ใช้จะต้องทำมาจากสารแม่เหล็กที่มีสภาพความเป็นแม่เหล็กสูง



ภาพที่ 2.5 โครงสร้างของสเตปปิ้งมอเตอร์แบบแม่เหล็กถาวรสี่เฟส

2.5 การกระตุ้นสเตปปิ้งมอเตอร์

การกระตุ้นสเตปปิ้งมอเตอร์สามารถทำได้โดยการป้อนไฟให้กับขดลวดในแต่ละเฟสตามลำดับที่เหมาะสมจากภาพ 2. 6 โครงสร้างของสเตปปิ้งมอเตอร์



ภาพที่ 2.6 โครงสร้างของสเตปปิ้งมอเตอร์แบบแม่เหล็กถาวร

การกระตุ้นให้สเตปปิ้งมอเตอร์ทำงานสามารถทำได้ดังนี้

1. การกระตุ้นที่ละเฟส
2. การกระตุ้นแบบย่อยสเตป

ตารางที่ 2.1 การกระตุ้นทีละเฟสจะป้อนไฟกระตุ้นให้กับสเตปปิ้งมอเตอร์ทีละเฟส

ลำดับที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่4
1	◉			
2		◉		
3			◉	
4				◉
5	◉			
6		◉		
7			◉	
8				◉

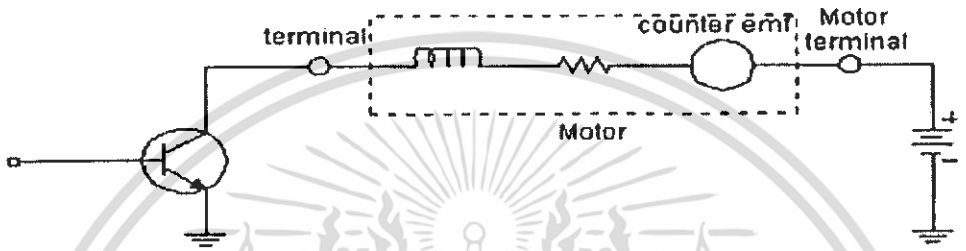
ตารางที่ 2.2 การกระตุ้นแบบย่อยสเตปจะป้อนไฟกระตุ้นให้กับสเตปปิ้งมอเตอร์

ลำดับที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่4
1	◉			
2	◉	◉		
3		◉		
4		◉	◉	
5			◉	
6			◉	◉
7				◉
8	◉			◉

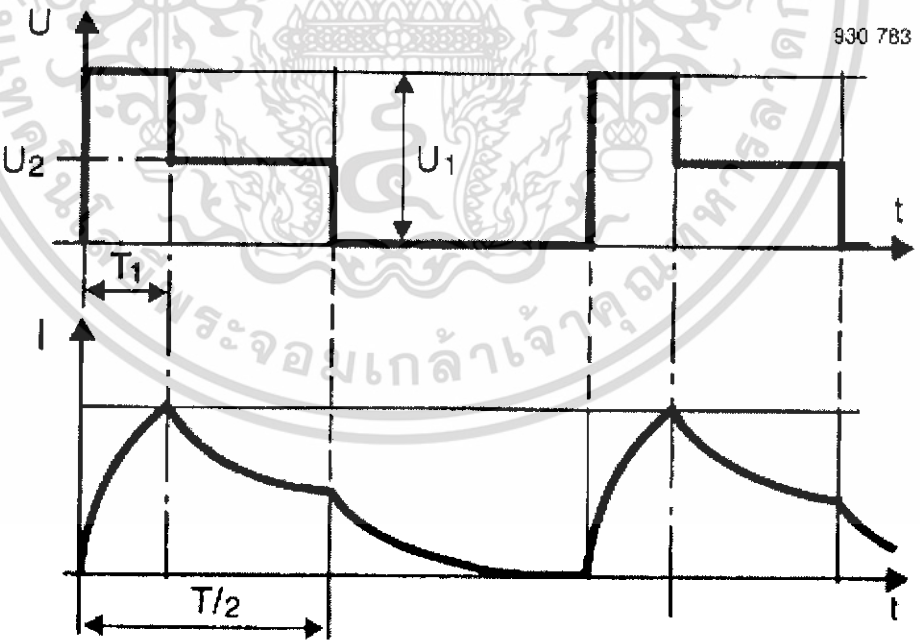
2.6 วงจรขับสำหรับสเตปปิ้งมอเตอร์

การพิจารณาถึงหลักการในการออกแบบวงจรขับให้กับขดลวดของสเตปปิ้งมอเตอร์ควรจะต้องคำนึงถึงองค์ประกอบหลายอย่างด้วยกันเนื่องจากการหมุนของสเตปปิ้งมอเตอร์จะทำให้เกิดแรงดันไฟฟ้าย้อนกลับ (Back emf) ซึ่งจะมีทิศทางตรงกันข้ามกับแหล่งจ่ายแรงดันไฟฟ้า

โดยสามารถเขียนรูปวงจรสมมูลย์ (Epuivalent circuit) ในหนึ่งเฟสของสเตปปิ้งมอเตอร์ได้แสดงดังในภาพที่ 2.7



ภาพที่ 2.7 แสดงวงจรสมมูลย์ของสเตปปิ้งมอเตอร์



ภาพที่ 2.8 กราฟแรงดันและกระแสแบบสองแหล่งจ่าย

หลักการแก้ไขปัญหาด้านแรงบิดที่มีผลต่อความถี่ โดยเงื่อนไขแรงบิดของสเตปปิ้งมอเตอร์นั้นขึ้นอยู่กับกระแสที่ไหลผ่านขดลวดเมื่อสเตปปิ้งมอเตอร์ทำงานอยู่ที่ความถี่ต่ำนั้นค่าของกระแสที่ไหลผ่านขดลวดนั้นจะถึงระดับที่สเตปปิ้งมอเตอร์ต้องการ ทำให้ได้สเตปปิ้งมอเตอร์อยู่ในสภาพคงแรงบิดที่ T2 แต่เมื่อทำการเพิ่มความถี่ให้มีค่าสูงขึ้นค่าเวลาก็จะมีค่าน้อยลงทำให้ไม่เพียงพอต่อการที่จะทำให้กระแสที่ไหลผ่านขดลวดนั้นจะถึงระดับที่ต้องการของสเตปปิ้งมอเตอร์ จึงทำให้ได้ค่ากระแสเฉลี่ยที่น้อยลง

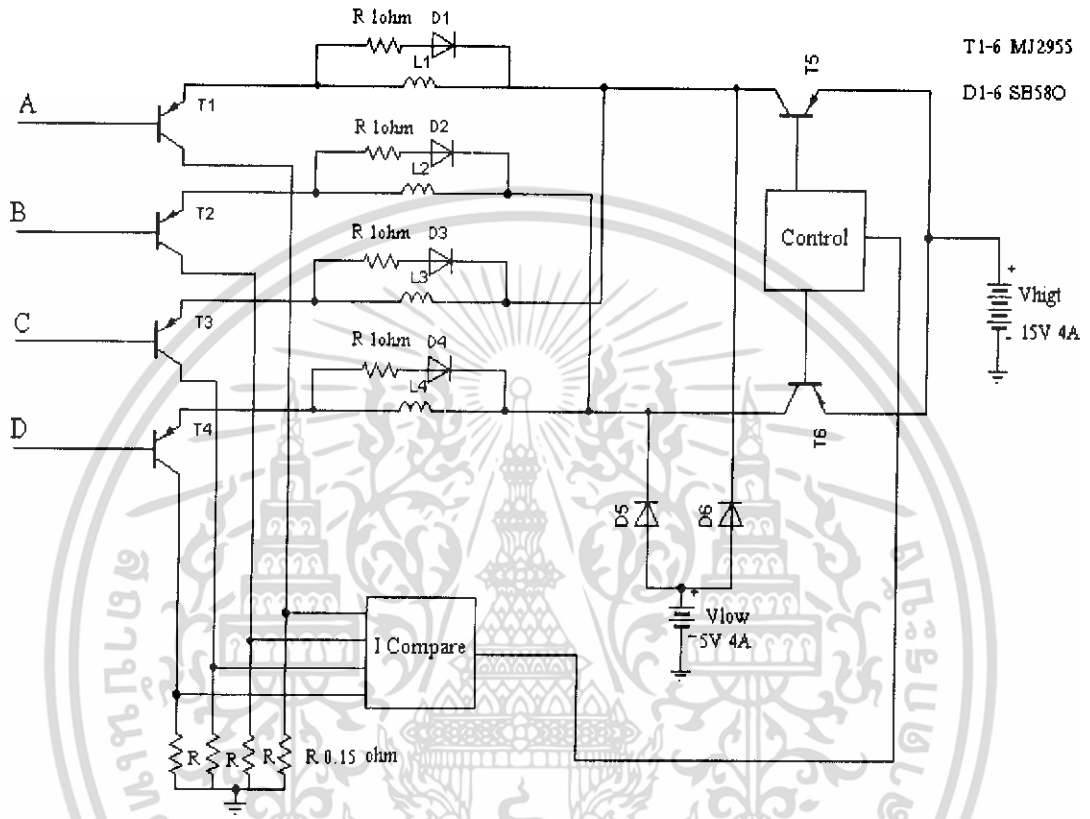
โดยการแก้ไขแบบสองแหล่งจ่ายนั้นจะทำโดยการจ่ายแรงดันสองระดับให้แก่วงจร โดยในสภาวะแรกจะทำการจ่ายแรงดัน V(high) ก่อนเพื่อให้กระแสถึงระดับที่สเตปปิ้งมอเตอร์ต้องการ เมื่อกระแสถึงระดับที่สเตปปิ้งมอเตอร์ต้อการแล้วก็จะทำการตัดแหล่งจ่าย V(high) ออกจากวงจรแล้วทำการจ่ายแรงดัน V(low) ข้ามาเพื่อรักษาให้สเตปปิ้งมอเตอร์อยู่ในสภาพคงแรงบิดต่อไป ซึ่งการเพิ่มแหล่งจ่ายอีกชุดเข้ามาจะช่วยยกระดับของกระแสให้สูงขึ้นและจะมีผลทำให้กระแสถึงระดับที่สเตปปิ้งมอเตอร์ต้องการเร็วขึ้นนั่นเอง ดูจากค่าเวลาที่ T1 ซึ่งตรงเส้นสีแดงจะเป็นระดับของกระแสที่สเตปปิ้งมอเตอร์ต้องการเมื่อทำการเปรียบเทียบ T2 แล้วค่าเวลา T1 จะมีค่าเวลาน้อยกว่า จึงช่วยให้สเตปปิ้งมอเตอร์ทำงานที่ย่านความถี่ที่สูงขึ้นได้

โดยทั่วไปสเตปปิ้งมอเตอร์ได้ถูกออกแบบให้ทนความร้อนได้สูงถึงประมาณที่ 100 องศา แต่ในการใช้งานจริงจะถูกใช้งานที่เงื่อนไขต่ำกว่าจุดอิ่มตัวที่กำหนดมา โดยทั่วไปการขับมอเตอร์นั้นจะต้องเลือกใช้อุปกรณ์ที่กินกระแสสูง เนื่องจากชุดขดลวดในแต่ละเฟสของสเตปปิ้งมอเตอร์จะต้องมีการนำและหยุดนำกระแสอยู่ตลอดเวลา ดังนั้นจึงจำเป็นอย่างยิ่งต้องออกแบบเพื่อป้องกันความเสียหายที่อาจเกิดขึ้นกับทรานซิสเตอร์กำลังจากแรงดันยอดแหลม (Spike Voltage) ที่เกิดจากการเหนี่ยวนำของกระแสในขดลวด (Inductive Turn Off Spike Voltage) และการเสียสภาพฉับพลันของแรงดันซึ่งการออกแบบวงจรขับและวงจรป้องกันสามารถทำได้ดังนี้

2.7 วงจรขับสองแหล่งจ่าย

เป็นวิธีการขับที่มีประสิทธิภาพกว่ารูปแบบของการต่อความต้านทานอนุกรม ซึ่งใช้แรงดันไฟฟ้าระดับสูงสำหรับยกระดับกระแสและติดไปยังแรงดันไฟฟ้าระดับต่ำ เมื่อถึงอัตรากระแสที่ต้องการหรือเวลาที่กำหนดกระแสที่อยู่ในขดลวดนั้นจะไหลผ่านไดโอดโดยแรงดันไฟฟ้าระดับต่ำจะทำงานแทน วิธีการนี้เรียกว่าการขับแรงดันไฟฟ้าคู่หรือการแบบสองระดับเพื่อพิจารณาการลดลงของเวลาในการเปลี่ยนระดับกระแสของระบบการขับแบบสองระดับเปรียบเทียบกับแบบการต่อความต้านทานอนุกรม แรงดันไฟฟ้าตกคร่อมขดลวดจะยังคงอยู่ทั้งหมดจนกระทั่งกระแสเพิ่มขึ้นจนถึงระดับกระแสที่ต้องการ

แม้ว่าค่าเวลาคงตัวจะไม่ลดลงเหมือนในแบบการต่อความต้านทานทางอนุกรม ค่าการยกระดับกระแสข้อนข้างจะมีลักษณะเป็นเชิงเส้น หลังจากแรงดันไฟฟ้าระดับสูงจะตัดต่อไปสู่แรงดันไฟฟ้าระดับต่ำกระแสจะตกลงและแรงดันย้อนกลับ



ภาพที่ 2.9 วงจรขับแบบสองแหล่งจ่าย

บทที่ 3
คุณลักษณะโดยทั่วไปของอุปกรณ์ประเภท FPGA

ในบทนี้จะกล่าวถึงคุณสมบัติทั่วไปที่มีอยู่ในตัวอุปกรณ์ FPGA ซึ่งได้อ้างอิงกับอุปกรณ์ที่เป็นผลิตภัณฑ์ของ Xilinx ตระกูล XC3000 เนื่องจากในการวิจัยนี้อาศัยอุปกรณ์ในตระกูลนี้มาใช้งาน จัดเป็นตระกูลในประเภท LCA (Logic Cell Array) ที่มีประสิทธิภาพและความจุสูง สถาปัตยกรรมโครงสร้างภายในอุปกรณ์ประกอบด้วยส่วนโครงสร้างสำคัญ 3 ส่วนด้วยกันคือ I/O Bocks (IOBs), Configurable Logic Bocks (CLBs) และ resources สำหรับการเชื่อมต่อระหว่างกัน ลักษณะโครงสร้างโดยทั่วไปของอุปกรณ์ LCA แสดงดังภาพที่ 3.1

การใช้งานหน้าที่ทางตรรกและการเชื่อมต่อระหว่างกันของ LCA ถูกกำหนดด้วยข้อมูลโปรแกรมโครงแบบ (Configuration program data) ที่เก็บอยู่ในเซลล์หน่วยความจำแบบสถิตภายใน (static memory cells) ซึ่งโปรแกรมนี้สามารถสั่งให้ทำงานในลักษณะต่างๆตามความเหมาะสมของแต่ละระบบที่ต้องการข้อมูลโปรแกรมอาจอยู่ในหน่วยความจำภายนอก เช่น EEPROM, EPROM หรือ ROM ที่อยู่บนแผงวงจร หรือเก็บอยู่ในฟลอปปี้ดิสหรือฮาร์ดดิสก์ได้ เมื่อทำการเปิดเครื่องหรือป้อนแหล่งจ่ายไฟเข้าไป ข้อมูลโปรแกรมจะถูกดึงลงมายังอุปกรณ์ LCA ในทันที

คุณสมบัติเด่นของอุปกรณ์ตระกูล XC3000 คือมีความสามารถและความจุทางตรรกที่มากมาย รูปร่างภายนอกของชิ้นอุปกรณ์หลากหลาย อีกทั้งช่วงอุณหภูมิการทำงานและความเร็วสูง

3.1 สถาปัตยกรรมโครงสร้าง

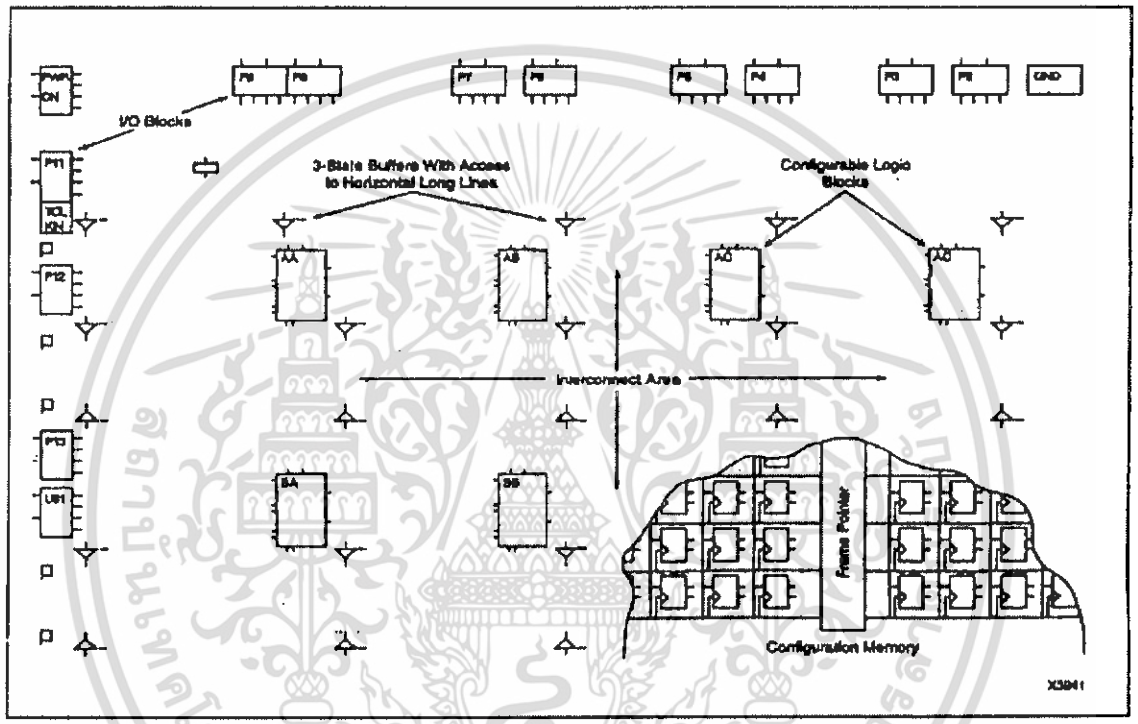
บริเวณโดยรอบกรอบสี่เหลี่ยมของ configurable IOBs จัดการเกี่ยวกับการอินเตอร์เฟสระหว่างแถวลำดับตรรก (logic array) ภายในกับขาต่างๆ ของอุปกรณ์ ส่วนแถวของ CLBs ให้การทำงานทางตรรกตามที่ผู้ใช้งานกำหนด และส่วน resources การเชื่อมต่อระหว่างกันถูกใช้ในการสร้างโครงข่ายและนำพาสัญญาณตรรกระหว่างบล็อกคล้ายกับลายเส้นทองแดงในแผงวงจร

บล็อกการทำงานทางตรรกสามารถทำงานให้เป็นผลจริงได้โดยลักษณะการโปรแกรมที่เป็นแบบตารางเปิดดูส่วนการทำงานอื่นเพิ่มเติมทำได้โดยโปรแกรมควบคุมตัวมัลติเพลกเซอร์และการเชื่อมต่อระหว่างกันในโครงข่ายระหว่างบล็อกต่างๆ ทำได้โดยการเชื่อมต่อส่วนที่เป็นโลหะด้วยโปรแกรมที่ควบคุมผ่านทางตัวทรานซิสเตอร์

การทำงานของอุปกรณ์ LCA จะมีความถูกต้องเที่ยงตรง เมื่อดึงเอาโปรแกรมโครงแบบเข้ามาไว้ภายใน และกระจายออกไปตามเซลล์หน่วยความจำโครงแบบต่างๆ การดึงหรือโหลดเอาโปรแกรมโครงแบบเข้ามานี้จะเกิดขึ้นทันทีที่ป้อนแหล่งจ่ายที่เข้าไปหรือจากคำสั่งให้โหลดข้อมูล

โดยตรงใน LCA จะประกอบด้วยสัญญาณตรรกและสัญญาณควบคุมที่เป็นแบบธรรมดาและแบบอัตโนมัติ

ข้อมูลโปรแกรมอาจอยู่ในรูปแบบขนานหรืออนุกรมต่อๆกันก็ได้ ระบบพัฒนาที่มีชื่อเรียกว่า XACT จะช่วยในการสร้างหรือกำเนิดขบวนบิตข้อมูลของโปรแกรมที่ใช้สำหรับเป็นโครงแบบหรือกำหนดการทำงานให้กับอุปกรณ์ LCA ซึ่งขบวนการทำงานดึงหรือเก็บเกี่ยวกับหน่วยความจำจะขึ้นอยู่กับการใช้งานในหน้าที่ต่างๆ ไม่เกี่ยวข้องกัน



ภาพที่ 3.1 แสดงโครงสร้างโดยทั่วไปของอุปกรณ์ FPGA

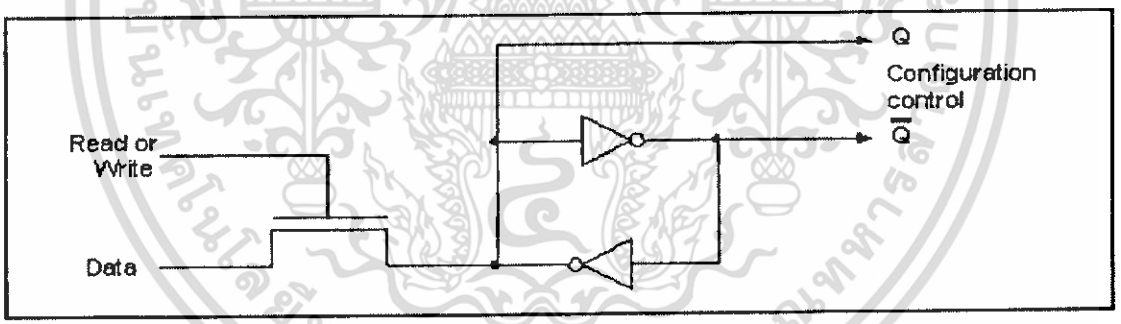
3.1.1 หน่วยความจำโครงแบบ

ใน LCA ส่วนของ หน่วยความจำโครงแบบ จะใช้เซลล์หน่วยความจำแบบสถิต ซึ่งได้รับการออกแบบขึ้นมาให้มีความเชื่อถือได้สูง อีกทั้งปราศจากสัญญาณรบกวนหรือแปลกปลอมเข้ามาแทรกแซง ความแน่นอนของหน่วยความจำโครงแบบของอุปกรณ์ LCA บนพื้นฐานการออกแบบนี้ทำให้สามารถรับรองได้ว่าจะไม่เกิดเหตุการณ์ที่อยู่นอกเหนือจากเงื่อนไขที่ต้องการเปรียบเทียบกับอุปกรณ์ในประเภทที่โปรแกรมได้อย่างอื่น พบว่าหน่วยความจำแบบสถิตนี้ให้ผลดีที่สุดทั้งในด้านความจุ ประสิทธิภาพ และมีความเชื่อถือได้สูง ดังแสดงในภาพที่ 3.2 เซลล์หน่วยความจำพื้นฐานประกอบด้วย CMOS อินเวอร์เตอร์ 2 ตัว รวมกับทรานซิสเตอร์ที่ใช้สรหรับการเขียน-อ่านเซลล์ข้อมูล ซึ่งเซลล์จะสามารถถูกเขียนได้อย่างเดียวในระหว่างทำการสร้างและอ่านอย่างเดียวใน

ระหว่างทำการดึงกลับออกมา ในภาวการณ์ทำงานปกติ เซลล์จะให้การควบคุมเป็นไปอย่างต่อเนื่องและตัวทรานซิสเตอร์จะไม่ทำงานและไม่มีผลกระทบกับความเสถียรภาพของเซลล์แต่อย่างใด ซึ่งค่อนข้างแตกต่างไปจากอุปกรณ์อื่นโดยทั่วไปในด้านที่เซลล์มักจะมีการอ่านและเขียนเข้าอยู่เป็นประจำ

ขาออกของเซลล์หน่วยความจำ Q และ $\overline{Q}$ จะใช้ระดับของ กราวด์ และ Vcc ซึ่งให้เป็นแบบต่อเนื่องและควบคุมโดยตรง ความสามารถเพิ่มเติมเกี่ยวกับโหลดพร้อมกับที่ไม่มีการถอดรหัสตำแหน่งและความรวดเร็วของตัวขยายช่วยให้เกิดความเสถียรแก่เซลล์สูง และจากโครงแบบของเซลล์หน่วยความจำโครงแบบ มันไม่มีผลกระทบที่เนื่องมาจากการเปลี่ยนแปลงอย่างกะทันหันของแหล่งจ่ายไฟหรือพลังงานรังสี ในการทดสอบพบว่าไม่มีข้อผิดพลาดใดเกิดขึ้นมาให้เห็นในขณะที่ทำงานภาพใต้บริเวณที่มีพลังงานรังสีสูงๆ

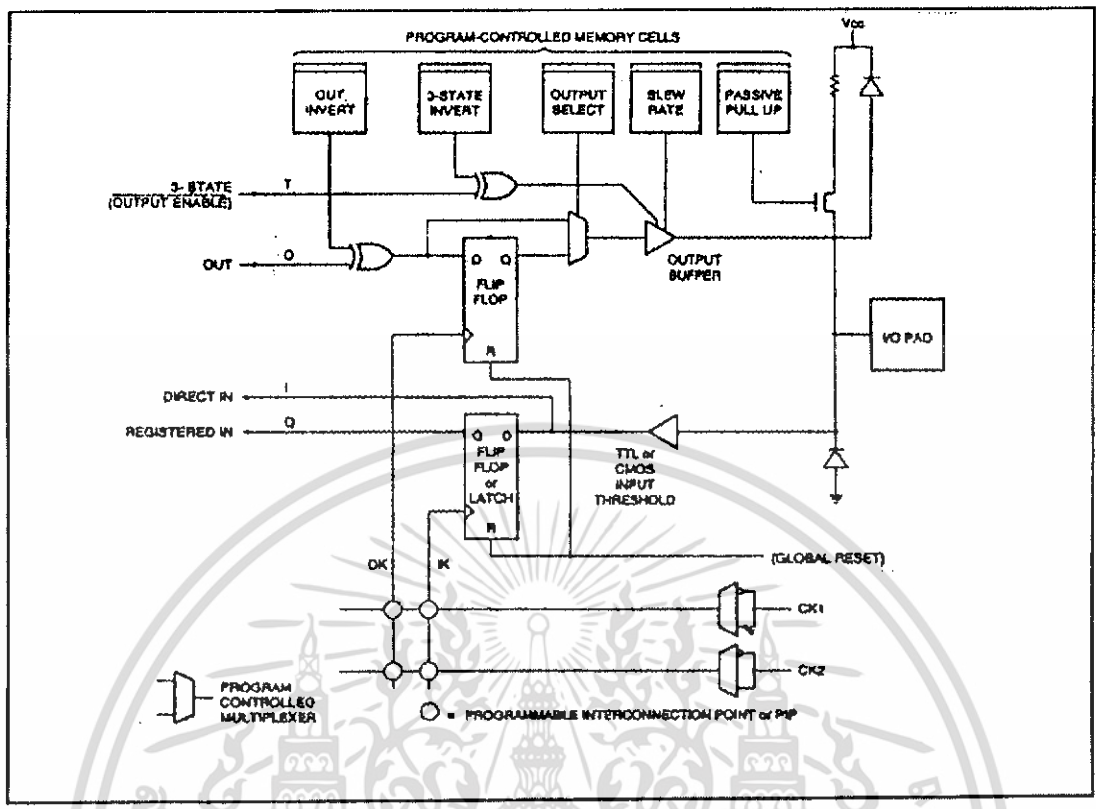
กรรมวิธีการโหลดข้อมูลโครงแบบมีหลายวิธีให้เลือกใช้ โดยปกติแล้วจะใช้ระบบพัฒนา XACT ช่วยในข้อมูลโปรแกรมเพื่อส่งตรงไปยังเซลล์หน่วยความจำโครงแบบข้อมูลที่เป็นแบบอนุกรมและนับตามความยาว ช่วยให้เกิดความเข้ากันได้สำหรับการผสมใช้งานอุปกรณ์ LCA หลายๆ ตัวในแบบซิงโคนัส, อนุกรม หรือแบบลูกโซ่



ภาพที่ 3.2 เซลล์หน่วยความจำโครงแบบแบบสถิต

3.1.2 I/O Block

ในโครงสร้างของ IOB แสดงดังภาพที่ 3.3 ทำหน้าที่เชื่อมต่อระหว่างขาภายนอกของอุปกรณ์เข้ากับการใช้ตรรกที่อยู่ภายในตัวอุปกรณ์ ในแต่ละ IOB จะมีทั้งการรีจิสเตอร์และเส้นทางตรงอินพุต และให้ 3-สเตต เอาท์พุตบัฟเฟอร์ ที่อาจถูกขับโดยการรีสเตอร์หรือสัญญาณเอาท์พุตโดยตรง ในโครงแบบยอมให้มีการกลับ (inversion) ในแต่ละ IOB ควบคุมอัตราการสลูว์ (slew rate) และ อิมพิแดนซ์สูง และในแต่ละทางเข้าของวงจรจะมีตัว แคมป์ปิ้งไดโอด เพื่อป้องกันเกี่ยวกับไฟฟ้าสถิต และวงจรป้องกันการเกิด latch-up เนื่องจากกระแสที่เข้ามา



ภาพที่ 3.3 บล็อกอินพุต-เอาต์พุท (IOB)

ส่วนของอินพุต-บัฟเฟอร์ ในแต่ละ IOB จะให้ขอบเขตหรือจัดระดับของการตรวจจับเกี่ยวกับการแปลงหรือพาสัญญาณจากภายนอกที่ขาของอุปกรณ์ไปยังระดับตรรกภายใน ค่าระดับหรือขอบเขตที่ตัว อินพุต-บัฟเฟอร์ สามารถโปรแกรมได้ให้อยู่ในระดับ TTL หรือ CMOS สัญญาณที่เข้ามาอยู่ในบัฟเฟอร์ ทำหน้าที่ส่งหรือขับข้อมูลที่เข้ามาของส่วนที่เก็บข้อมูล ซึ่งอาจจะอยู่ในรูปของฟลิปฟลอปหรือแลตซ์ ในส่วนของขั้วสัญญาณนาฬิกาที่ใช้สามารถทำการโปรแกรมได้ไม่ว่าจะต้องการใช้ขอบด้านใดของสัญญาณนาฬิกาก็ตามในการเป็นสัญญาณกระตุ้นให้ส่วนอื่นเริ่มทำงาน เมื่อมีการใช้เส้นสัญญาณนาฬิกาไปกระตุ้นให้กับฟลิปฟลอปหรือแลตซ์ต้องมีการชดเชยสำหรับความแตกต่างของสัญญาณนาฬิกาด้วยเสมอ ชิ้นส่วนเก็บข้อมูล I/O จะถูกรีเซ็ตในขณะที่ทำโครงแบบหรือโดยสัญญาณสั่งให้รีเซตสัญญาณอินพุตทั้งคู่ (ขา I และขา Q ของ IOB) สามารถเชื่อมต่อระหว่างกันได้

เพื่อความเชื่อถือได้ในขณะทำงาน อินพุตที่เข้ามาจะต้องมีค่าเวลาในการเปลี่ยนแปลงระดับต่ำกว่า 100 nSec. และต้องไม่มีการเหลือตกค้างอยู่ ซึ่งอาจทำให้เกิดการผิดพลาดเป็นสัญญาณรบกวนในระบบได้ในแต่ละ IOB ที่ใช้จะมีตัวต้านทานอินพีแดนซ์สูง ซึ่งเลือกโปรแกรมได้เพื่อใช้สำหรับกรณีที่ไม่มีการขับไปยังขาของอุปกรณ์

เวลาประวิงในลูปสำหรับ IOB และฟลิปฟลอปจะมีค่าประมาณ 3 nSec. ซึ่งถือว่าสั้นมาก ทำให้เกิดประสิทธิภาพสูงในการทำงานภายใต้เงื่อนไขของข้อมูลและสัญญาณนาฬิกาที่เป็นแบบอะซิงโครนัส และจากค่าเวลาประวิงที่ต่ำนี้ทำให้อุปกรณ์ LCA สามารถนำสัญญาณซิงโครนัสจากภายนอกมาใช้งานได้

เซลล์หน่วยความจำที่ถูกโปรแกรมควบคุมในภาพที่ 3.4 ใช้เลือกควบคุมหน้าที่ต่างๆดังนี้

- การผกผันตรรกของ อินพุท จะถูกควบคุมโดยโปรแกรมโครงแบบเพียง 1 บิตต่อ IOB

- ตรรกควบคุม 3-สเตต ของแต่ละ IOB เอาท์พุทบัฟเฟอร์ จะถูกกำหนดโดยสถานะของบิตโปรแกรมโครงแบบที่จะให้บัฟเฟอร์ปิดหรือเปิด หรือเลือกเป็น 3-สเตต (ที่ขา T ของ IOB) เมื่อสัญญาณควบคุม เอาท์พุตของ IOB เป็นระดับสูงหรือตรรก '1' ตัวบัฟเฟอร์จะอยู่ในสภาวะใช้งานไม่ได้ และที่ขาภายนอกจะเป็นอิมพีแดนซ์สูง แต่ถ้าสัญญาณควบคุมนี้เป็นระดับต่ำหรือตรรก '0' ก็หมายถึงบัฟเฟอร์ที่ใช้งานได้และมีสัญญาณออกไปที่ขาอุปกรณ์

- Direct หรือ registered เอาท์พุต สามารถเลือกได้ในแต่ละ IOB ตัวรีจิสเตอร์จะใช้ขอบที่เป็นบวกของสัญญาณนาฬิกา ซึ่งสัญญาณนาฬิกานี้อาจมาจากเส้นใดเส้นหนึ่งในจำนวน 2 เส้นที่มี

- การเพิ่มความเร็วของการเปลี่ยนระดับเอาท์พุตสามารถทำการเลือกความเร็วนี้ให้เหมาะสมกับค่าเวลาที่จำเป็นต้องใช้ ถ้าเพิ่มความเร็วนี้ขึ้นมากก็จะเกิดสัญญาณรบกวนมากตาม

- ตัวต้านทานที่เป็นอิมพีแดนซ์สูงภายในใช้ป้องกันกรณีที่ไม่มีการใช้งานหรือไม่ได้เชื่อมต่ออินพุตไม่ให้เกิดการรบกวนเข้าไปยังส่วนอื่นของวงจร

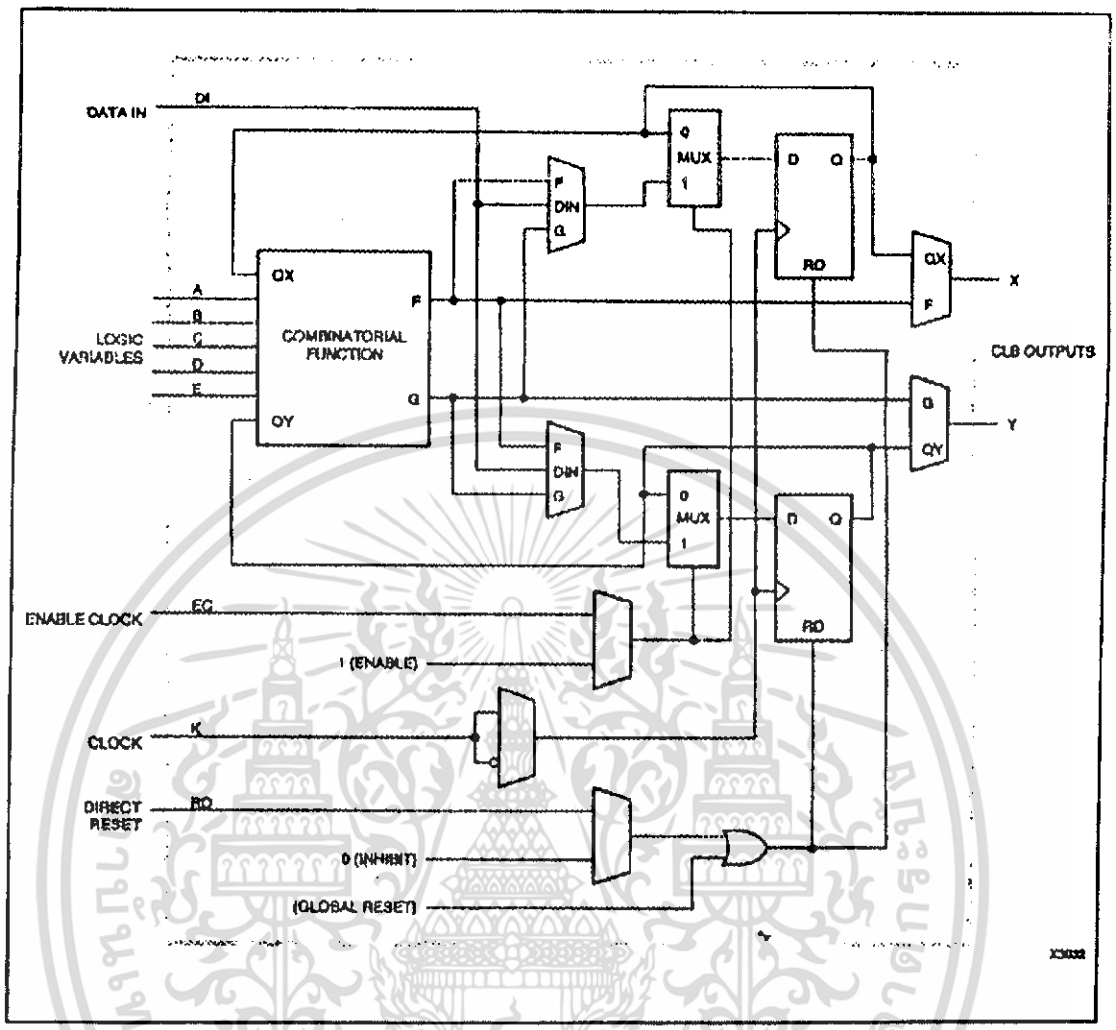
ดังนั้นการเลือกในส่วนของ I/O สามารถสรุปได้ดังนี้

- ในด้าน อินพุต
  - Direct
  - Flip-flop หรือ latch
  - CMOS หรือ TTL threshold (chip inputs)
  - Pull-up resistor หรือ open circuit
- ในด้าน เอาท์พุต
  - Direct หรือ registered
  - inverted หรือ not
  - 3-state หรือ on หรือ off
  - Full speed หรือ slow limited
  - 3-state หรือ เอาท์พุต enable (inverse)

3.1.3 Configurable Logic Block (CLB)

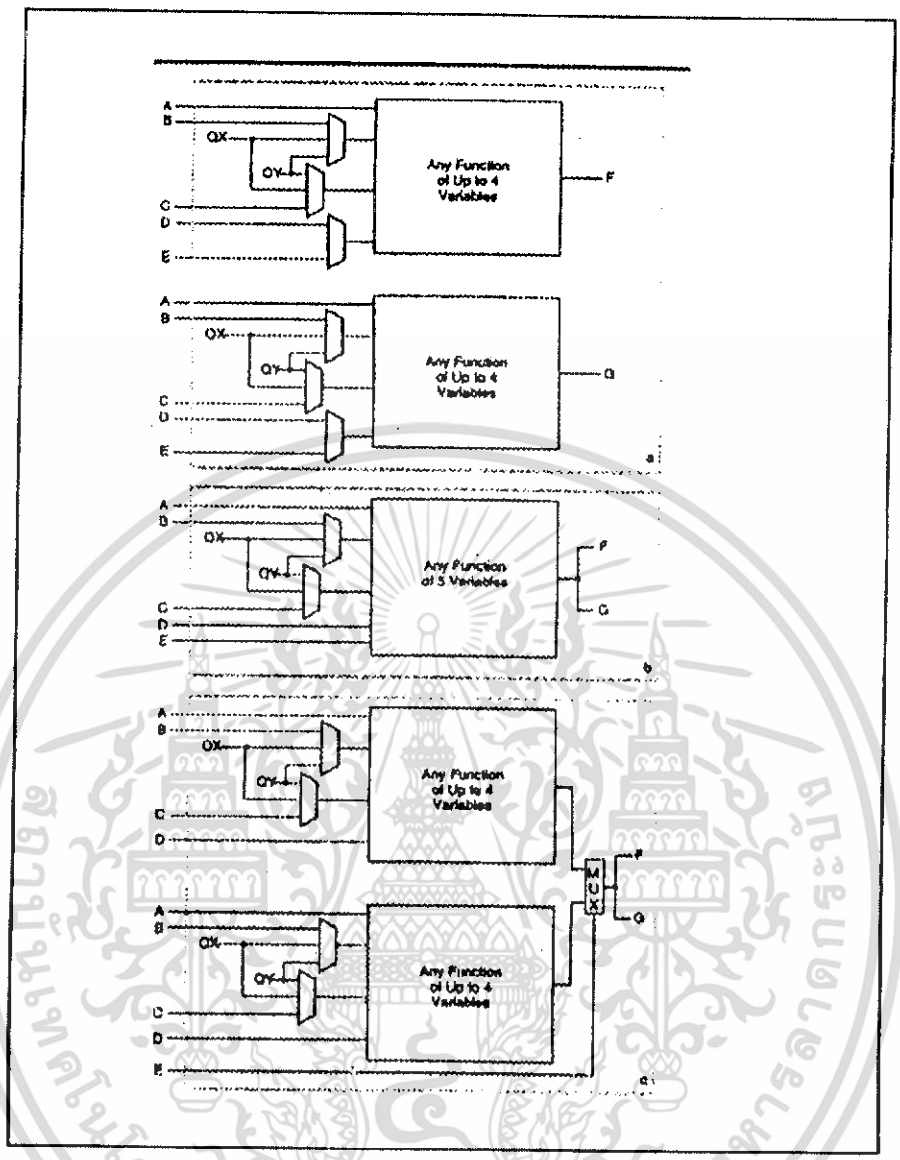
แถวลำดับของ CLB ให้การทำงานตามที่ผู้ใช้กำหนดหรือสร้างขึ้นมา โดยบล็อกตรรกเหล่านี้ถูกจัดเรียงในลักษณะของเมตริกซ์อยู่ภายในกรอบของ IOBs ตัวอย่างเช่นตัว XC3020 จะมีทั้งหมด 64 บล็อก จัดเรียงเป็น 8 แถว และ 8 ช่อง และใช้ระบบพัฒนา XACT ในการคอมไพล์ข้อมูลโครงแบบที่จะถูกนำไปสู่หน่วยความจำโครงแบบภายในเพื่อใช้กำหนดหน้าที่การทำงานและการเชื่อมต่อระหว่างกันของแต่ละบล็อก ผู้ใช้หรือผู้ออกแบบสามารถกำหนดเกี่ยวกับ CLBs และการเชื่อมต่อในโครงข่ายในแบบที่แปลงจากรูปวงจรแบบอัตโนมัติหรือเลือกใช้จากคลังข้อมูลที่ติดตั้งอยู่ หรือโดยผู้ใช้กำหนดเองก็ได้

แต่ละ CLB จะมีทั้งส่วนที่รวมตรรก, ฟลิปฟลอป 2 ตัว และส่วนควบคุมภายในดังแสดงในภาพที่ 3.4 คือ มี 5 ตรรกที่เป็นอินพุต (A, B, C, D และ E) สัญญาณนาฬิการ่วมที่เข้ามา (K), อะซิงโครนัสรีเซ็ตที่เข้ามาโดยตรง (RD) และ สัญญาณนาฬิกาให้ทำงาน (EC) ซึ่งทั้งหมดจะถูกขับจาก resources การเชื่อมต่อที่อยู่ใกล้ๆ กับบล็อก แต่ละ CLB จะมีเอาท์พุต อยู่ 2 เส้นคือ X กับ Y ซึ่งอาจใช้ส่งไปยังโครงข่ายที่มีการเชื่อมต่อถึงกัน

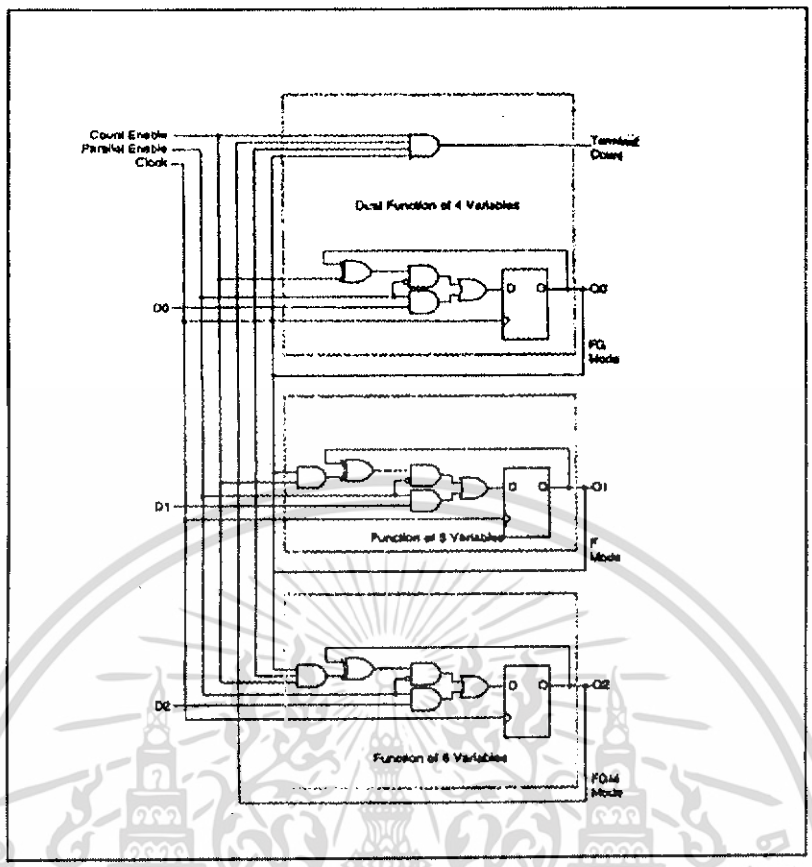


ภาพที่ 3.4 โครงสร้างของบล็อกตรรก

ส่วนของตรรกรวมใน CLB จะใช้ตารางเปิดดู แบบ 32 ต่อ 1 ในการแทนหน้าที่การทำงานตามสมการแบบบูล การเลือกที่เปลี่ยนแปลงได้จากทั้ง 5 อินพุต และฟลิปฟลอปทั้ง 2 ตัวภายในจะถูกใช้เป็นตำแหน่งของตาราง ค่าเวลาประวิงโดยรวมที่เกิดขึ้นในตลอดโครงข่ายจะขึ้นอยู่กับหน้าที่การทำงานที่สร้างขึ้นมาแต่ละหน้าที่และแยกอิสระจากกันสำหรับอินพุตที่เปลี่ยนไปแต่ละตัว ทำให้สามารถสร้างหน้าที่การทำงานทางตรรกที่เป็นอิสระต่อกัน 2 อย่างแบบมี 4 ตัวแปรดังภาพที่ 3.5(a) หรือหน้าที่เดียวแต่มี 6 ตัวแปรดังภาพที่ 3.5(b) หรือหน้าที่บางอย่างที่มี 7 ตัวแปรดังภาพที่ 3.5(c) ส่วนในภาพที่ 3.6 แสดงถึงตัวนับแบบ Modulo-8 ซึ่งใช้ตรรกรวมเพียงบล็อกเดียวในแต่ละส่วน



ภาพที่ 3.5 ลักษณะการจัดรวมตรรกให้มีหน้าที่การทำงานสำหรับจำนวนตัวแปรที่ต่างกัน

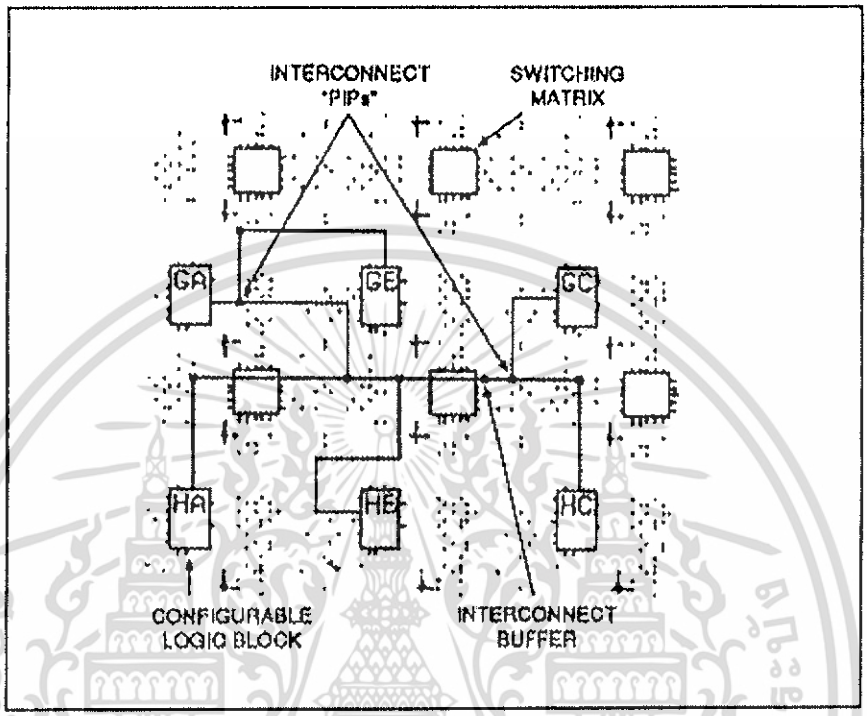


ภาพที่ 3.6 ตัวนับแทน modulo-8 ที่ใช้บล็อกตรรกรวมเดียวในแต่ละส่วน

3.1.4 การเชื่อมต่อระหว่างกันแบบโปรแกรมได้ (Programmable Interconnect)

ทรัพยากรสำหรับใช้เชื่อมต่อระหว่างกันที่อยู่ใน LCA เป็นเส้นทางการเชื่อมต่อถึงกันของ IOBs และ CLBs เข้ากันเป็นโครงสร้างข่ายผ่านทางเส้นที่เป็นโลหะออกแบบกำหนดผ่านทาง ตัวทรานซิสเตอร์ ซึ่งแต่ละตัวถูกควบคุมโดยบิตของโปรแกรมโครงแบบจากจุดเชื่อมต่อระหว่างกันที่โปรแกรมเลือกได้ (PIPs) และตัวสวิตซ์เมตริกซ์ จะใช้ในกรณีที่จำเป็นสำหรับการเชื่อมต่อระหว่างเส้นโลหะที่เลือกใช้กับขาของบล็อก ในภาพที่ 3.7 แสดงตัวอย่างของเส้นทางเชื่อมต่อโครงข่าย สำหรับระบบพัฒนา XACT จะถูกใช้เมื่อต้องการให้เส้นทางการเชื่อมต่อและตำแหน่งการวางเป็นไปแบบอัตโนมัติ อินพุตของ IOBs และ CLBs เป็นตัวมัลติเพลกเซอร์ ซึ่งสามารถถูกโปรแกรมให้เลือกโครงข่ายอินพุตจากส่วนการเชื่อมต่อที่อยู่ใกล้กันและเนื่องจากการเชื่อมต่อไปยังบล็อกอินพุต สามารถเลือกได้ทั้งสองทิศทางเหมือนกับทางบล็อกเอาท์พุตจะใช้เพียงเฉพาะที่เป็นการต่อบล็อกอินพุตเท่านั้นและไม่เกี่ยวกับการทำเส้นทาง ดังภาพที่ 3.8 เป็นตัวอย่างแสดงการทำเส้นทางของบล็อกตรรกอินพุต, ส่วนควบคุมและบล็อกเอาต์พุตและส่วนของโลหะ 3 ชนิดที่จัดเตรียมหรือมีไว้ให้กับความต้องการของการเชื่อมต่อระหว่างกันในโครงข่ายแบบต่างๆ คือ

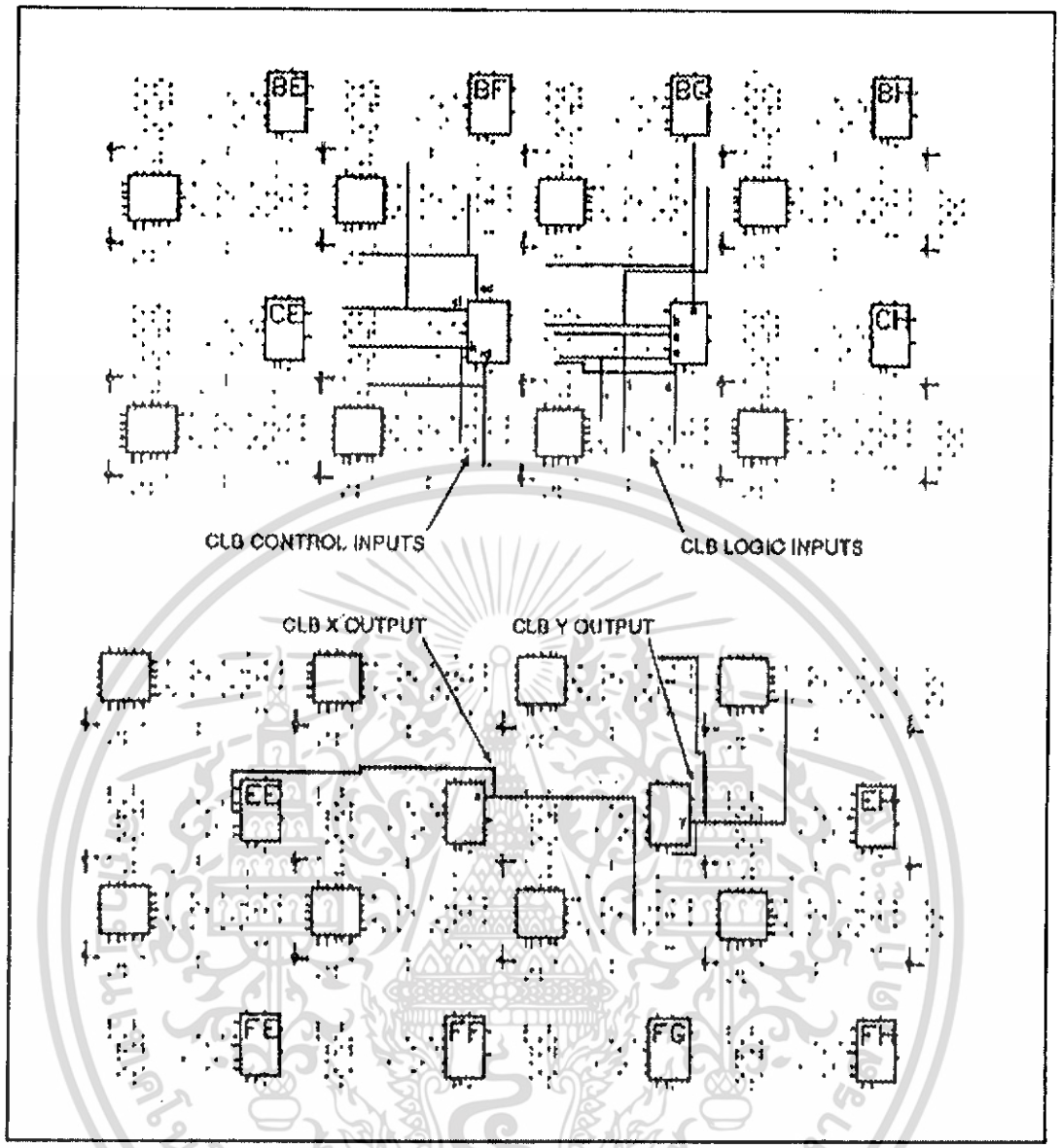
- การเชื่อมต่อระหว่างกันตามวัตถุประสงค์ทั่วไป
- การต่อตรง
- เส้นยาว (longlines)



ภาพที่ 3.7 ภาพที่ได้จากระบบพัฒนา XACT ในส่วนการใช้ routing resources

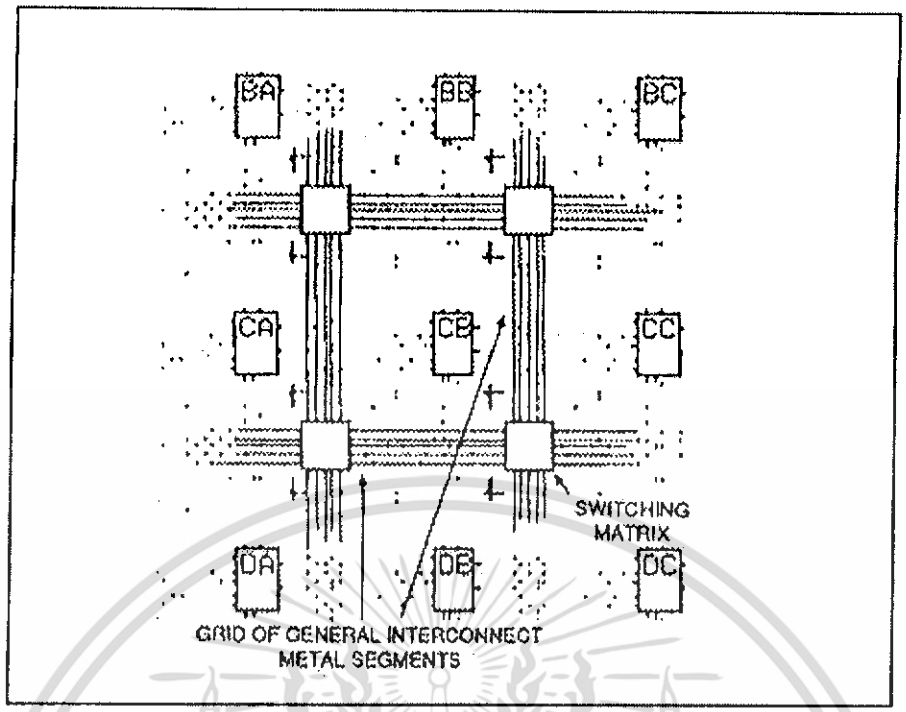
3.1.5 การเชื่อมต่อระหว่างกันตามวัตถุประสงค์ทั่วไป

ดังแสดงในภาพที่ 10 ประกอบด้วยตะแกรงโลหะ 5 แถวทางแนวนอน และ 5 แถวทางแนวตั้ง วางไว้ในตำแหน่งระหว่างช่องของตรรกและ IOBs แต่ละส่วนจะมีความกว้างและสูงตามบล็อกตรรก ตัวสวิตซ์เมตริกซ์จะเป็นตัวเชื่อมตรงปลายของส่วนต่างๆและยอมให้เลือกการเชื่อมต่อระหว่างส่วนตะแกรงโลหะทางแนวนอนและแนวตั้งที่อยู่ติดกัน ถ้าสวิตซ์ตัวไหนไม่ถูกโปรแกรมเลือกก็จะไม่มีการเชื่อมต่อ ซึ่งการเลือกเส้นทางเชื่อมต่อนี้อาจกระทำได้โดยใช้โปรแกรม

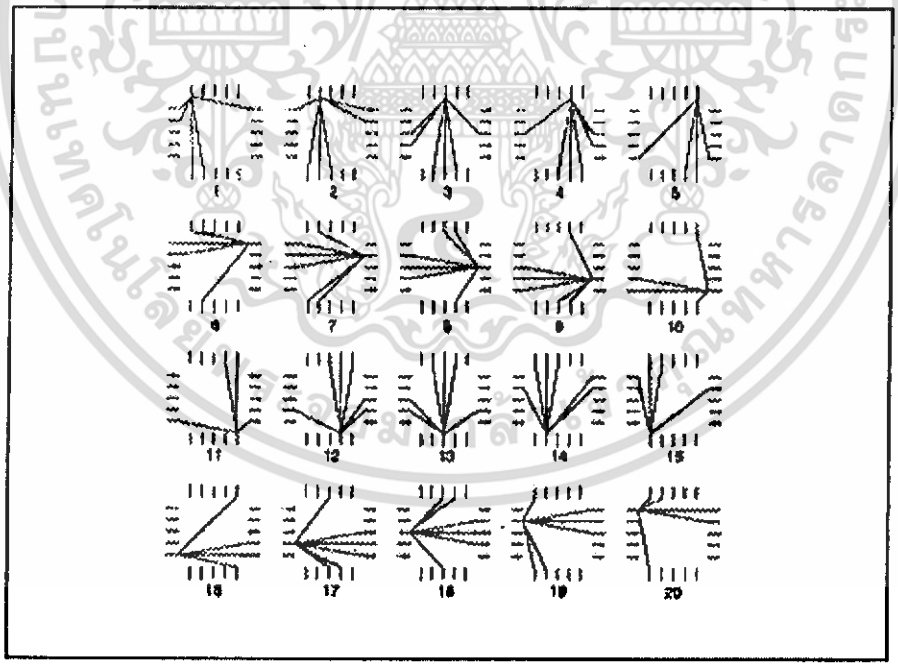


ภาพที่ 3.8 ภาพที่ได้จากระบบพัฒนา XACT ในส่วนของตำแหน่งและการเชื่อมต่อระหว่างกัน

ซอฟต์แวร์เข้าช่วยในการเลือกขาของเมตริกซ์ที่จะถูกต่อหรือไม่ต่อ ในภาพที่ 3.10 แสดงถึงตัวอย่างสวิตซ์เมตริกซ์สำหรับแต่ละขาที่ถูกต้องและอาจใช้คำสั่ง Show–Matrix ที่มีอยู่ในระบบ XACT เพื่อแสดงภาพให้ชัดเจนยิ่งขึ้นได้



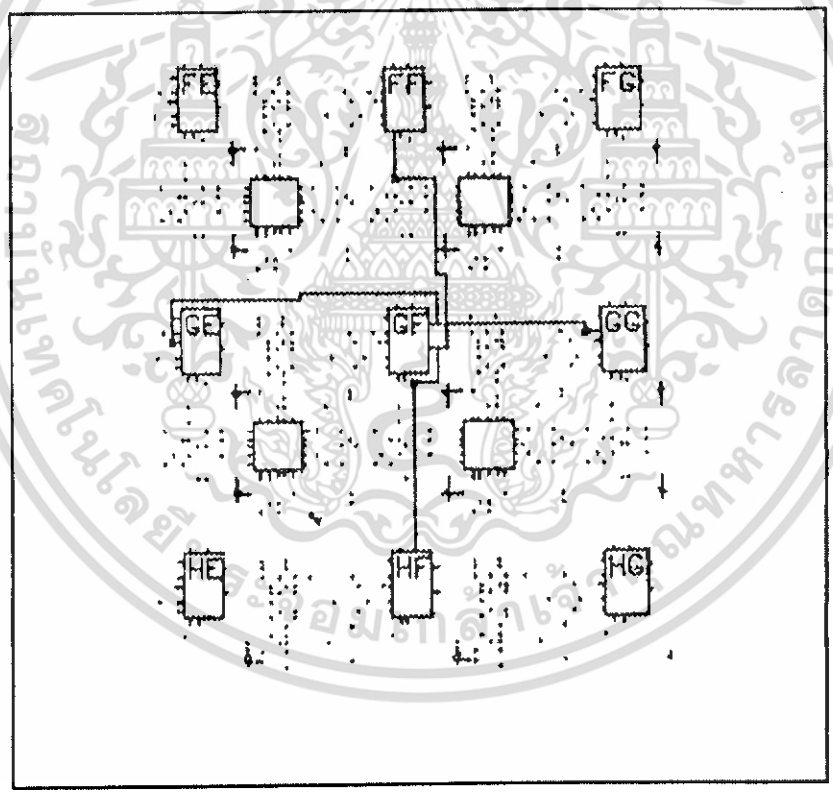
ภาพที่ 3.9 การเชื่อมต่อถึงกันภายในอุปกรณ์ FPGA ตามแบบทั่วไป



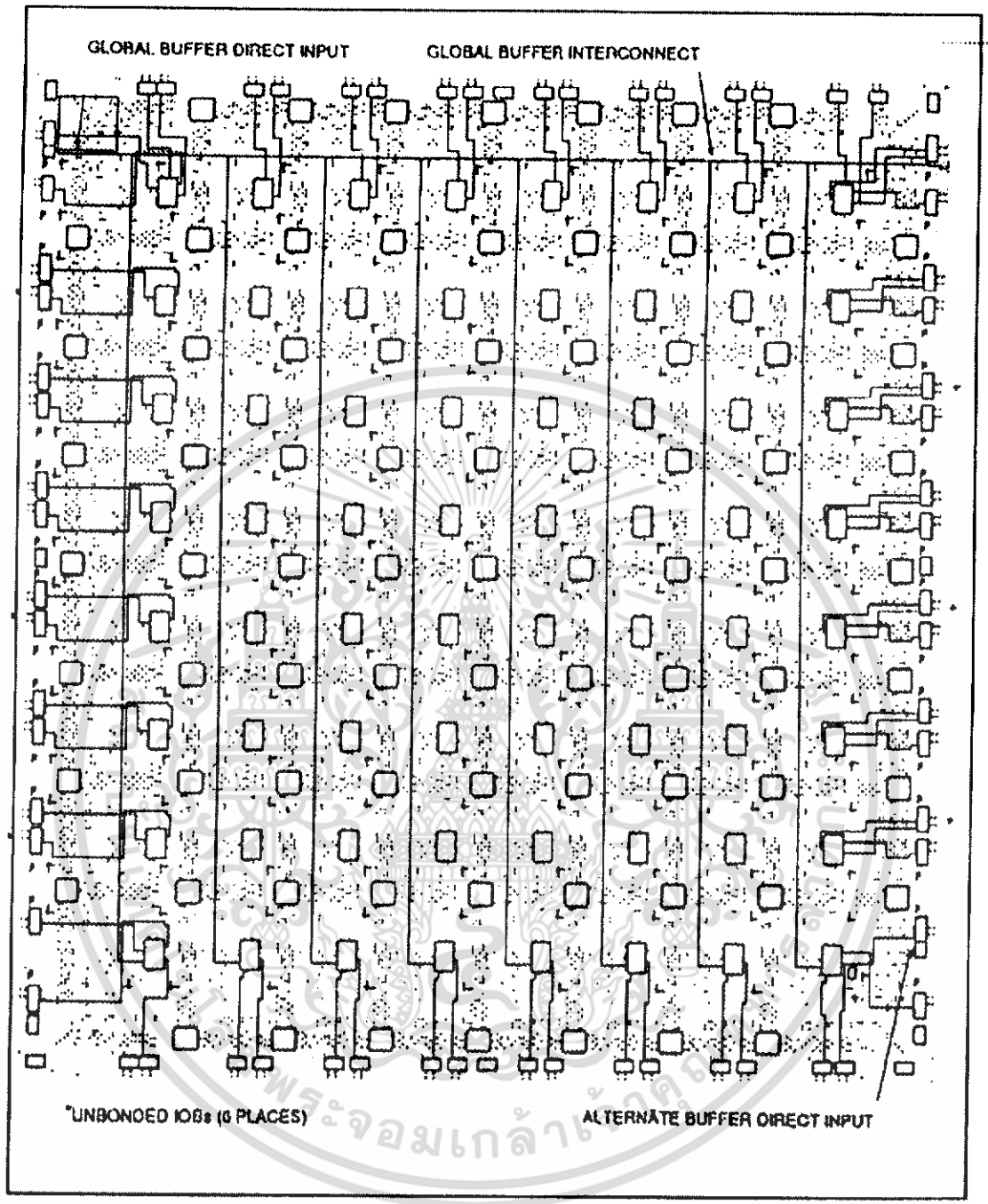
ภาพที่ 3.10 ทางเลือกของการเชื่อมต่อถึงกันของสวิตซ์เมตริกซ์สำหรับแต่ละขา

3.1.6 การต่อตรงระหว่างกัน

ดังแสดงในภาพที่ 3.11 ซึ่งเป็นการแทนโครงข่ายที่มีประสิทธิภาพสูงในการเชื่อมต่อระหว่างบล็อกของ IOBs กับ CLBs ที่อยู่ใกล้กัน สัญญาณจะถูกส่งผ่านจากบล็อกถึงบล็อกได้โดยตรง ทำให้ไม่ต้องใช้เส้นสำหรับการเชื่อมต่อทั่วไปที่มีอยู่ สำหรับในแต่ละ CLB จุดเอาต์พุต X อาจต่อตรงถึงอินพุต B ของ CLB ทางด้านขวาและต่อไปยังจุดอินพุต C ของ CLB ทางด้านซ้าย ส่วนจุดเอาต์พุต Y สามารถต่อตรงไปยังจุดอินพุต D ของบล็อกที่อยู่ข้างบนและจุดอินพุต A ของบล็อกที่อยู่ด้านล่าง การต่อตรงในลักษณะนี้จะถูกนำมาใช้เมื่อมีความต้องการความเร็วที่สูงสุดในส่วนของตรรกตำแหน่งสถานที่บล็อกตรรกที่อยู่ใกล้กับ IOB การต่อตรงจะเลือกได้ทั้ง 4 มุมทางด้านขวาจะให้ การต่อตรงจากเอาท์พุต ของ CLBs ไปยัง IOBs ที่อยู่ใกล้กัน ดังตัวอย่างที่แสดงในภาพที่ 3.12



ภาพที่ 3.11 แสดงการต่อถึงกันโดยตรงระหว่างเอาท์พุต X และ Y ของ CLB

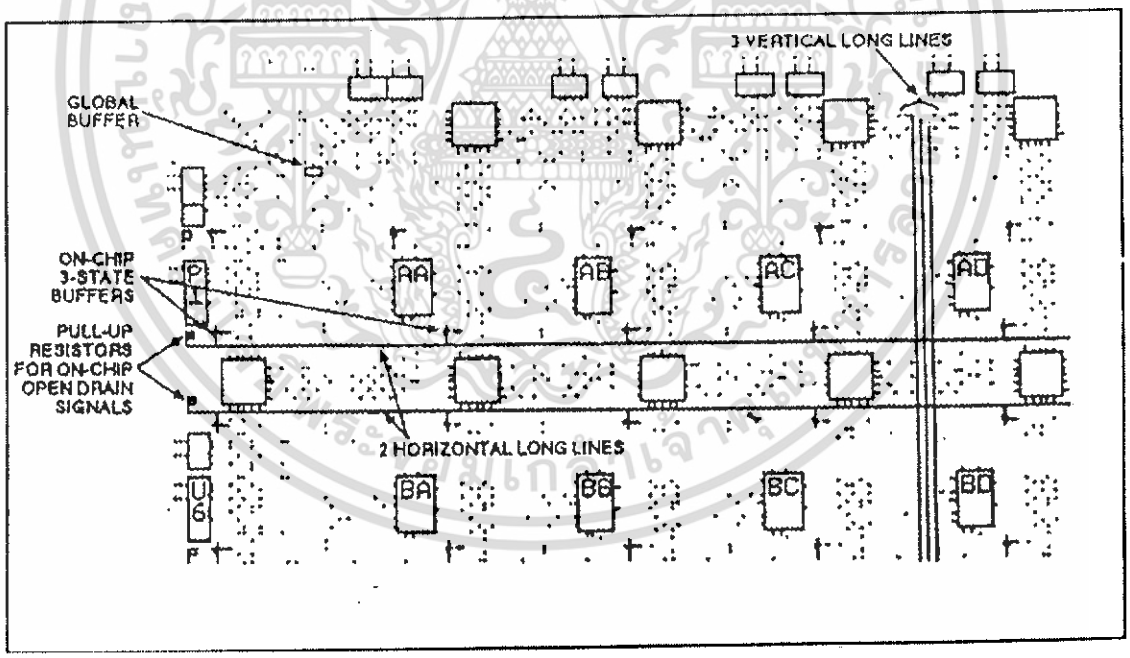


ภาพที่ 3.12 ตัวอย่างการต่อถึงกันโดยตรงระหว่าง CLB ที่อยู่ใกล้กันภายในอุปกรณ์ XC3020

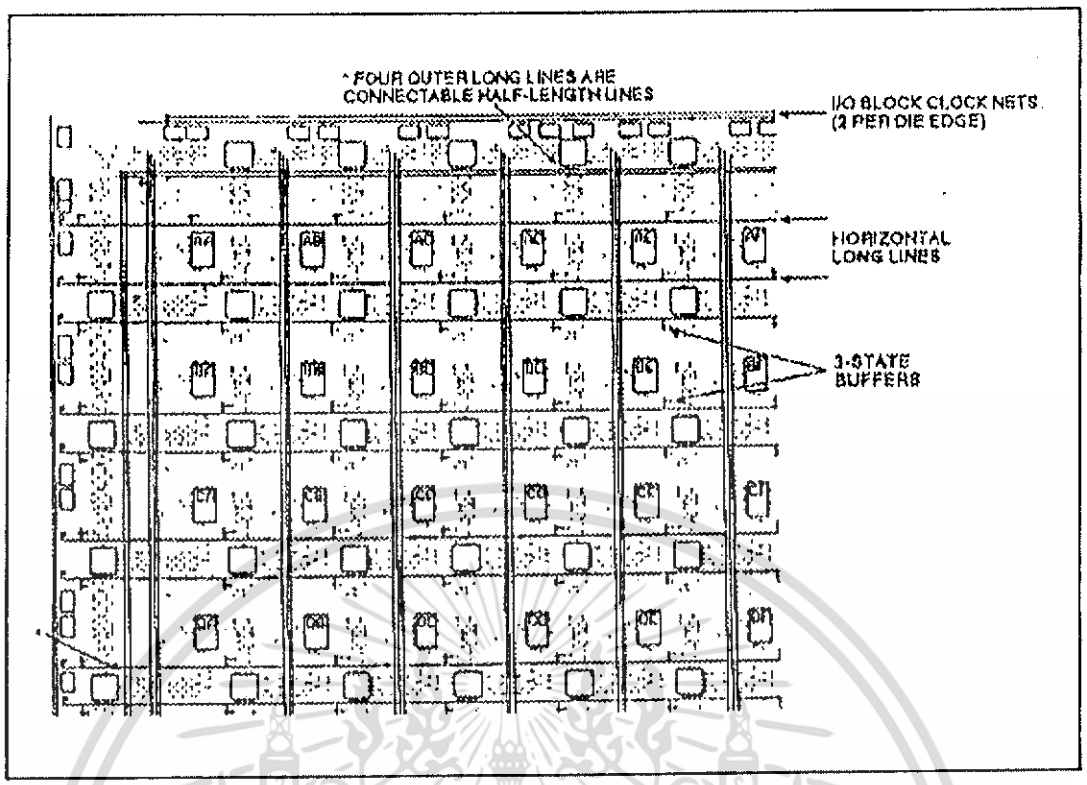
3.1.7 เส้นยาว

ใช้สำหรับเป็นถนนให้กับสวิตซ์เมตริกซ์และสำหรับการเดินทางของสัญญาณที่ต้องใช้ระยะมาก หรือต้องมีจำนวนการผ่านน้อยสุดในระหว่างจุดปลายทาง เส้นยาวแสดงให้เห็นดังภาพที่ 3.14 มีเส้นวิ่งทั้งทางแนวนอนแนวตั้งตามขนาดความสูงและความกว้างของบริเวณการเชื่อมต่อ ในแต่ละช่องจะมีเส้นแนวตั้ง 3 เส้นและระหว่างแถวจะมีเส้นทางแนวนอนอยู่ 2 เส้นที่เป็นเส้นยาว และมีอีก 2 เส้นอยู่ภายนอกของกลุ่มสวิตซ์เมตริกซ์

เส้นยาว สามารถใช้งานโดยบล็อกตรรก หรือเอาต์พุต ของ IOB แบบช่องต่อช่อง การใช้งานด้วยเส้นยาวแบบนี้ทำให้ลดจำนวนเส้นของสัญญาณนาฬิกาหรือสัญญาณควบคุมภายในแต่ละช่องของบล็อกตรรกลง ลักษณะการเชื่อมต่อดังแสดงในภาพที่ 3.14 โดยแยกบัฟเฟอร์ออกที่แต่ละอินพุตไปยังเส้นยาว ตัวบัฟเฟอร์ทางด้านมุมซ้ายของชิฟ LCA จะส่งไปตลอดทั่วทั้งหมดทุกอินพุต K ของบล็อกตรรก และด้วยการใช้บัฟเฟอร์ทั่วไปนี้สำหรับเป็นสัญญาณนาฬิกาจะทำให้สามารถมีค่าแฟนเอาต์ (fan-out) สูง และใช้สัญญาณนาฬิกาเข้าจังหวะพร้อมกันทั้งหมดได้ทุกๆอันทั้งหมดใน IOBs และ CLBs ส่วนบัฟเฟอร์ที่อยู่มุมด้านขวาจะขับเส้นขาวทางแนวแนวนอน ซึ่งสามารถส่งการเชื่อมต่อไปได้ยังเส้นทางแนวตั้งได้ในแต่ละช่องของการเชื่อมต่อระหว่างกัน



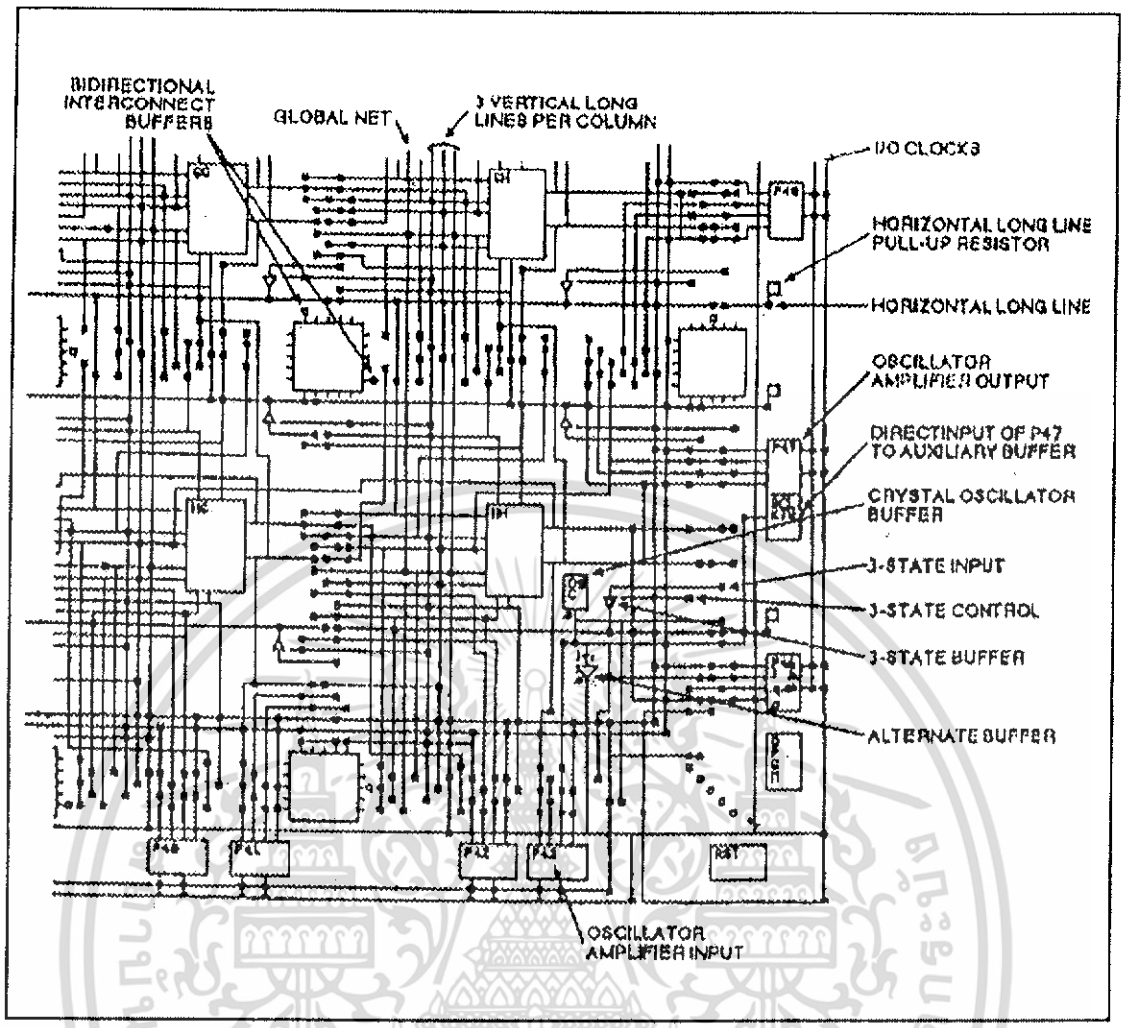
ภาพที่ 3.13 แสดงเส้นยาว (longlines) ทางแนวนอนและแนวตั้ง



ภาพที่ 3.14 แสดงตัวอย่างการเลือกจุดต่อถึงกันของเส้นยาว

3.1.8 บัสภายใน

คู่ของบัฟเฟอร์แบบ 3-สเตต ที่วางไว้ในตำแหน่งใกล้กับ แต่ละ CLB จะให้ตรรกะที่ใช้ขับเส้นยาวทางแนวนอน การทำงานตรรกของบัฟเฟอร์แบบ 3-สภาวะนี้ควบคุมให้ทำหน้าที่เป็นการผสมสัญญาณทั่วไป ซึ่งทุกๆอินพุตสามารถเลือกได้โดยใช้ระดับสัญญาณตรรกต่ำบนเส้นควบคุมของ 3-สเตต สำหรับการส่งขับไปยังเส้นยาวทางแนวนอน เมื่อผู้ใช้ตัดการหลีกเลี่ยงการขัดแย้งซึ่งสามารถเป็นผลมาจากการที่มีตัวขับหลายตัว และมีระดับของตรรกที่ไม่เหมือนกัน ต้องควบคุมให้อินพุต ที่จะออกไปขับตัวบัฟเฟอร์เป็นระดับสัญญาณเดียวกันหมด ถ้าเป็นระดับสูงหมดก็จะอยู่ในสภาวะอิมพีแดนซ์สูง คือไม่ได้ใช้งาน แต่ถ้าเป็นระดับตรรกต่ำหมดก็คือใช้บัฟเฟอร์ขับไปยังเส้นทางที่ต่ออยู่ ส่วนตัวต้านทานแบบพูลอัพ (pull-up) ที่อยู่ตรงปลายของเส้นยาวจะให้เอาต์พุตเป็นตรรกสูง เมื่อไม่ได้ใช้ตัวบัฟเฟอร์ ภาพที่ 3.16 แสดงให้เห็นถึงการเชื่อมต่อระหว่างกันของตัวบัฟเฟอร์แบบ 3-สเตต, เส้นยาวและตัวต้านทานแบบพูลอัพที่เป็นไปได้โดยนำมาเฉพาะส่วนที่อยู่มุมขวาด้านล่างของตัว XC3020

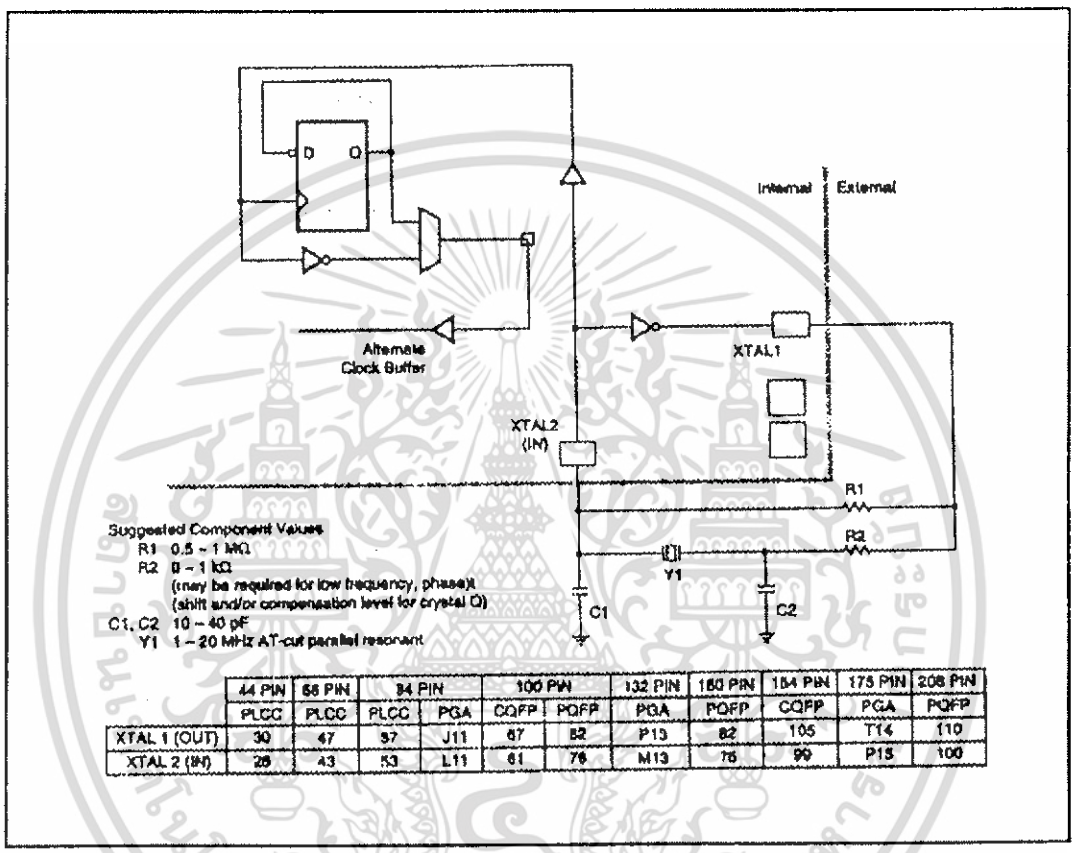


ภาพที่ 3.15 ตัวอย่างการเชื่อมต่อระหว่างกันภายใน (เฉพาะมุมขวาล่าง) ของอุปกรณ์ XC3020 ที่ได้จากระบบพัฒนา XACT

3.1.9 คริสตอลออสซิลเลเตอร์ (Crystal Oscillator)

ภาพที่ 3.16 แสดงถึงตัวขยายแบบผกผันความเร็วสูงที่มีอยู่ภายในอุปกรณ์ ซึ่งอาจใช้เป็นตัวคริสตอลออสซิลเลเตอร์บนชิพได้ ซึ่งใช้บัฟเฟอร์ช่วยในมุมขวาด้านล่าง เมื่อมีการสร้างตัวออสซิลเลเตอร์ขึ้นมาโดยโปรแกรม Make Bits และการเชื่อมต่อใช้เป็นแหล่งจ่ายสัญญาณจะมีการใช้ IOBs พิเศษ 2 ตัว สำหรับเชื่อมต่อตัวขยายออสซิลเลเตอร์กับส่วนประกอบที่เป็นคริสตอลออสซิลเลเตอร์ภายนอก ดังในภาพที่ 3.17 ซึ่งต้องแบ่งทั้ง 2 ส่วนให้สมมาตรกัน วงจรออสซิลเลเตอร์จะเริ่มต้นทำงานก่อนขบวนการอื่น เพื่อความเสถียรภาพของระบบการเชื่อมต่อจริงภายในจะถูกประวิงเอาไว้จนกว่าขบวนการสร้างจะเสร็จสมบูรณ์ในภาพที่ 3.16 ตัวต้านทานป้อนกลับ R1 ที่อยู่ระหว่างอินพุตและเอาต์พุต จะเป็นไบอัสให้กับตัวขยาย, การผกผันของตัวขยายร่วมกับวงจร R-C จะให้การเลื่อนเฟสไป 360 องศา ส่วนตัวต้านทานที่อยู่อนุกรมอยู่ R2 อาจ

นำมาต่อใช้เมื่อต้องการควบคุมการเลื่อนเฟส หรือควบคุมขนาดของสัญญาณที่ออกมาจากตัวขยาย ระดับแรงดันป้อนกลับที่สูงไปอาจทำให้ถูกต้องได้โดยใช้อัตราส่วนของ C2 ต่อ C1 ตัวขยายถูกออกแบบให้ใช้งานตั้งแต่ความถี่ 1 MHz ไปจนถึงประมาณครึ่งหนึ่งของความถี่ท็อกเกิล (toggle) ของตัว CLBs



	44 PIN	68 PIN	84 PIN		100 PIN		132 PIN	160 PIN	164 PIN	175 PIN	208 PIN
	PLCC	PLCC	PLCC	PGA	CQFP	PQFP	PGA	PQFP	CQFP	PGA	PQFP
XTAL 1 (OUT)	30	47	57	J11	67	82	P13	82	105	T14	110
XTAL 2 (IN)	29	43	53	L11	61	76	M13	76	99	P15	100

ภาพที่ 3.16 แสดงลักษณะการใช้คริสตอลออสซิลเลเตอร์

3.2 การโปรแกรม

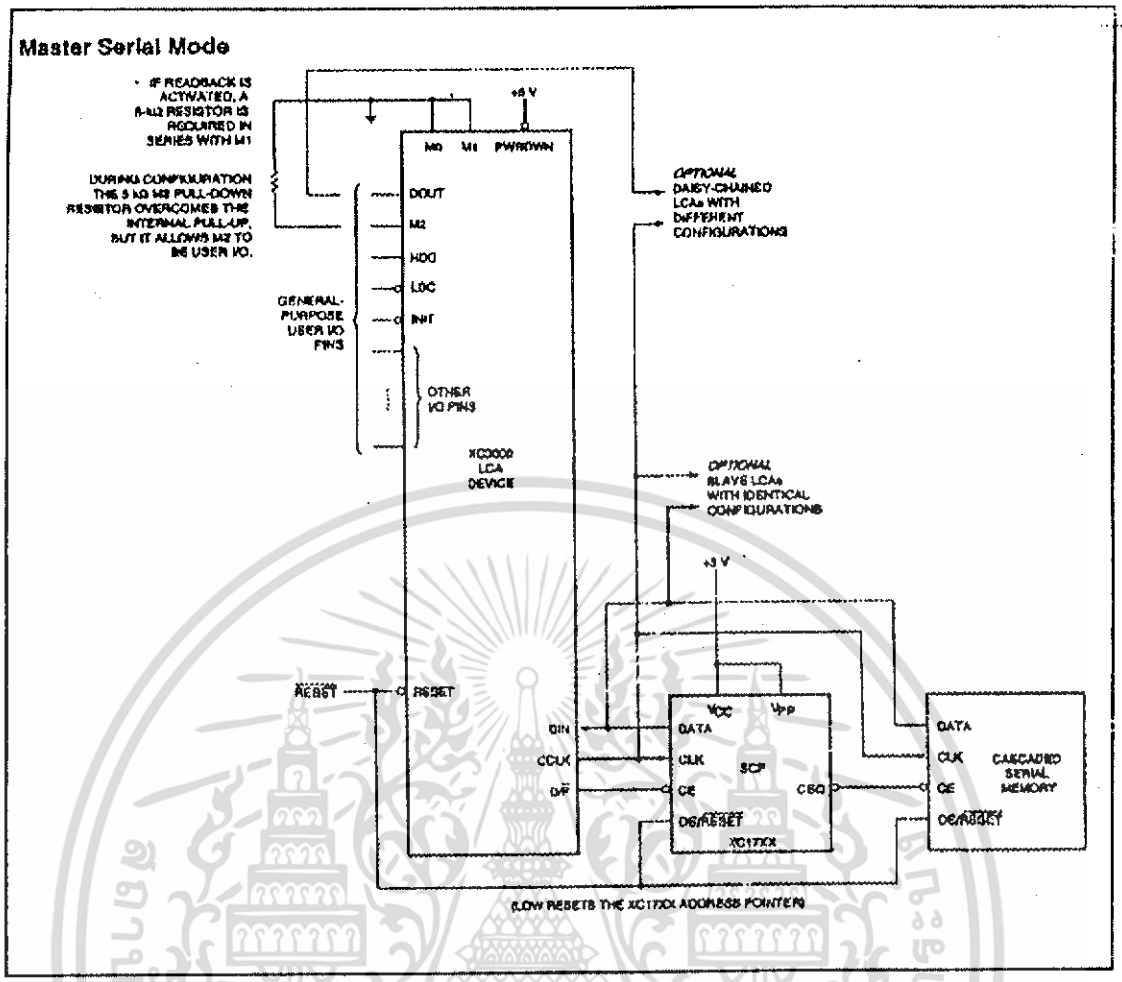
เมื่อมีแรงไฟ Vcc มาถึงส่วนของอุปกรณ์ LCA จะเริ่มต้นทำงาน (ปกติใช้ 2.5 ถึง 3 V) ค่าเวลาประวิงจะมีขึ้นจนกว่าแรงไฟของแหล่งจ่ายจะคงที่อยู่ในสภาพที่เสถียรและมีช่วงเวลา timeout ของการเริ่มต้นปกติประมาณ 11 ถึง 33 mSec. ในตารางที่ 3.1 แสดงให้เห็นรูปแบบโหมดการทำงาน 5 อย่างให้เลือกใช้โดยกำหนดที่ระดับอินพุตของขาโหมดทั้ง 3 คือ M0, M1 และ M2 โดยแบ่งโหมดการทำงานออกเป็น 3 โหมดใหญ่ๆ คือ master, peripheral และแบบ slave

ตารางที่ 3.1 การเลือกใช้โหมดการทำงานของอุปกรณ์ FPGA ที่สามารถกระทำได้

M0	M1	M2	CCLK	Mode	Data
0	0	0	เอาต์พุต	Master	Bit Serial
0	0	1	เอาต์พุต	Master	Byte Wide Addr. = 0000 up
0	1	0	-	Reserved	-
0	1	1	เอาต์พุต	Master	Byte Wide Addr. = FFFF up
1	0	0	-	Reserved	-
1	0	1	เอาต์พุต	Peripheral	Byte Wide
1	1	0	-	Reserved	-
1	1	1	อินพุต	Slave	Bit Serial

ในโครงสร้างแบบ master นั้น อุปกรณ์ LCA จะกลายเป็นแหล่งจ่ายของ Configulation Clock (CCLK) ส่วนการเริ่มต้นของอุปกรณ์ในโครงสร้างที่เป็น peripheral หรือ slave นั้นจะถูกประวิงเวลาเอาไว้นานเท่าที่จำเป็นให้เพียงพอต่อการเริ่มต้นที่สมบูรณ์ในอุปกรณ์ LCA ที่ถูกเลือกใช้เป็นโหมดแบบ Master นั้น สภาวะเริ่มต้นจะถูกเพิ่มขึ้นเป็น 4 เท่าตัวคือประมาณ 43 ถึง 130 mSec เพื่อให้แน่ใจว่าอุปกรณ์ตัวลูกที่ต่อพ่วงอยู่ทั้งหมดอยู่ในสภาพพร้อมที่จะทำงานได้แล้ว และหลังจากที่เป็นข้อมูลโครงสร้างถูกโหลดเข้าไปแล้วและเปรียบเทียบจำนวนความยาวเรียบร้อย สภาพของขา I/O ก็พร้อมใช้งานได้

สำหรับในการนำอุปกรณ์ LCA หรือ FPGA มาใช้งานในการทดลองวิจัยนี้ได้เลือกใช้โครงสร้างเป็นแบบ Master Serial Made ซึ่งเป็นรูปแบบหนึ่งใน Master Mode ดังแสดงตัวอย่างในภาพที่ 18 โดยลักษณะทำงานแบบนี้ อุปกรณ์ LCA หรือ FPGA จะทำการโหลดข้อมูลโครงแบบจากอุปกรณ์หน่วยความจำภายนอกอย่างอัตโนมัติ เมื่อมีการป้อนแหล่งจ่ายไฟเข้าไปหรือจากคำสั่งโหลดข้อมูลโดยตรง ซึ่งในลักษณะของ Master Serial Mode นี้จะใช้ข้อมูลโครงสร้างแบบอนุกรมเป็น Data-in (Din) มาจากแหล่งจ่ายอนุกรมแบบซิงโครนัส เช่น อุปกรณ์ Xilinx Serial Configuration PROM หรืออุปกรณ์หน่วยความจำประเภทอื่นที่มีลักษณะในทำนองเดียวกันนี้



ภาพที่ 3.17 ลักษณะการใช้งานอุปกรณ์ FPGA ให้มีการทำงานแบบ Master Serial Mode

บทที่ 4
ภาษาวีเอชดีแอล

วีเอชดีแอล ย่อมาจากคำว่า VHDL Hardware Description Langquage (VHSIC: Very Hight Speed Integrated Circuit) เป็นภาษาโปรแกรมระดับสูง (high level language) ที่ใช้ในการออกแบบฮาร์ดแวร์ในระบบดิจิตอล ตัวของภาษาสามารถบรรยายพฤติกรรมการทำงานในรูปของลำดับชั้น (hierarchy) ได้ และสามารถที่จะเขียนได้หลายรูปแบบซึ่งจะกล่าวต่อไป จึงทำให้ภาษาวีเอชดีแอลเป็นเครื่องมือที่ใช้ออกแบบตั้งแต่ขั้นตอนบนสุด คือ แนวความคิดที่จะแก้ปัญหาลงไปทีละขั้นจนถึงขั้นตอนของการสร้างวงจรจริงและตัวภาษาสามารถเปิดโอกาสให้วิศวกรได้พัฒนาและจำลองการทำงานของรูปแบบฟังก์ชันการทำงานของวงจรอย่างสังเขป โดยที่ยังไม่ต้องไปคำนึงถึงรายละเอียดเกี่ยวกับโครงสร้างวงจรจริง นอกจากนั้นวีเอชดีแอลยังเป็นภาษาที่สนับสนุนลักษณะต่างๆของระบบดิจิตอลที่มีความซับซ้อนได้ทั้งหมดจึงเป็นภาษาที่น่าสนใจในการศึกษาและนำไปใช้งาน

4.1 ประวัติความเป็นมาของภาษาวีเอชดีแอล

วิวัฒนาการของภาษาวีเอชดีแอล นั้นเริ่มต้นประมาณปี ค.ศ. 1981โดยที่กระทรวงกลาโหมสหรัฐอเมริกา หรือ ดีโอดี (DoD: Department of Defense) มองเห็นว่าอุปกรณ์อิเล็กทรอนิกส์และคอมพิวเตอร์ที่ใช้ในกิจการทหารเป็นอุปกรณ์ที่ได้รับการพัฒนามาเมื่อประมาณ 20 ปีก่อนเพราะเทคโนโลยีในขณะนั้นทำให้การพัฒนาอุปกรณ์อิเล็กทรอนิกส์เป็นไปอย่างล่าช้า ซึ่งเป็นสภาพที่ไม่อาจยอมรับได้ในปัจจุบัน เพราะเทคโนโลยีทางด้านไมโครอิเล็กทรอนิกส์ ได้รับการพัฒนาไปอย่างรวดเร็ว ดังที่จะเห็นได้ว่ามีวงจรดิจิตอลอิเล็กทรอนิกส์หลายวงจร ที่แต่เดิมถูกสร้างขึ้นมาจากชิ้น ถูกนำประกอบกันอยู่บนแผงวงจรไฟฟ้าที่มีขนาดใหญ่ แต่ในปัจจุบันสามารถที่จะใช้เทคโนโลยีการออกแบบและผลิตวงจรรวมขนาดใหญ่มาก (VLSI: Very Large Scale Integration) รวมอุปกรณ์ต่างๆ เหล่านั้นให้อยู่บนชิ้นอุปกรณ์สารกึ่งตัวนำที่มีขนาดประมาณ 1-2 ตร.ซม ได้ ซึ่งเป็นผลให้ประสิทธิภาพในการทำงานของวงจรสูงขึ้น (ความเร็วในการทำงานของวงจร) ตลอดจนความน่าเชื่อถือในการทำงานและความคงทนต่อสภาพแวดล้อมสูง ขณะเดียวกันนั้นในวงการทหารได้มีการนำระบบคอมพิวเตอร์และอิเล็กทรอนิกส์มาใช้ในระบบอาวุธอย่างแพร่หลาย ดังนั้นอุปกรณ์ที่มีใช้อยู่จึงไม่เหมาะสมกับเทคโนโลยีด้านอาวุธของประเทศคู่แข่งขัน การที่จะเปลี่ยนอุปกรณ์ใหม่เป็นสิ่งที่ต้องใช้งบประมาณมากและก็จะประสบกับปัญหาเช่นเดิมคือ อุปกรณ์ได้รับการพัฒนามานานแล้วเช่นกัน เพราะในขณะนั้นขั้นตอนของการออกแบบการผลิตและการตรวจสอบวงจรต้นแบบเป็นขบวนการที่ต้องใช้วิศวกรและเวลาสำหรับการดำเนินการมาก ฉะนั้น

การดำเนินการมาก ฉะนั้นทางดีโอดีตั้งโครงการขึ้นมาเพื่อศึกษาวิธีการที่จะช่วยพัฒนาวงจรอิเล็กทรอนิกส์ โดยเฉพาะอย่างยิ่งวงจรระบบดิจิตอลให้สามารถนำไปผลิตได้เร็วขึ้นและโครงการดังกล่าวมีชื่อว่า " Very High Speed Integrated Circuits " หรือ วีเอชเอสไอซี (VHSIC) ในระยะแรกนั้นโครงการเป็นความลับทางด้านความมั่นคงของประเทศและอยู่ในความดูแลควบคุมของ United States Inernational Traffic and Arms Regulation หรือ ไอทีเออาร์ (ITAR) ในปี ค.ศ. 1983 ตามคำแนะนำของคณะทำงาน ("Woods Hole" workshop) ทาง ดีโอดี ได้ออกความต้องการมาตรฐานของภาษาที่ใช้สำหรับบรรยายพฤติกรรมของวงจรหรือฮาร์ดแวร์ของระบบสำหรับโครงการวีเอชเอสไอซี ซึ่งมีสาระสำคัญพอสรุปได้ดังนี้

- ต้องเป็นภาษาที่นำไปเขียนรูปแบบระบบดิจิตอล และมีคุณสมบัติที่สามารถจะเข้าใจได้ทั้งคน และเครื่องโดยไม่ต้องมีการแปลหรือเปลี่ยแปลงอีก
- สามารถนำไปใช้เป็นเอกสารประกอบโครงการได้
- ต้องเป็นภาษาที่เขียนขึ้นสำหรับใช้จำลองการทำงานของวงจร

ฉะนั้นภาษดังกล่าวนี้จึงจัดเป็นภาษาโปรแกรมระดับสูง เช่นเดียวกับภาษาปาสคาล หรือภาษาซี ซึ่งในทางวิศวกรรมการออกแบบฮาร์ดแวร์เรียกว่า "Hardware Description Langquage" หรือ เอชดีแอล (HDL) เริ่มต้นโครงการดีโอดีได้มอบหมายให้บริษัทไอบีเอ็มและบริษัทเท็กซัลอินสตูลเมนท์ และ บริษัทเมทริกซ์เป็นผู้ศึกษาและพัฒนาการดำเนินการได้กระทำไปอย่างต่อเนื่องและได้เป็นผลที่น่าพอใจ จนกระทั่ง ค.ศ. 1985 ทางไอทีเออาร์ได้ยกเลิกข้อจำกัดในการถ่ายทอดเทคโนโลยีทางทหารออกจากโครงการนี้ ดังนั้นภาษาวีเอชดีแอลจึงเริ่มเป็นที่รู้จักกันโดยทั่วไป จนกระทั่งทางไออีอีอี (IEEE) จึงได้รับภาษานี้เข้ามาศึกษาและประมาณปี ค.ศ. 1987 ได้ยอมรับกำหนดมาตรฐานของภาษา โดยให้ชื่อว่า IEEE 1076-1987 และมีชื่อเรียกว่าวีเอชดีแอลมาตรฐานนี้ก็ได้รับการปรับปรุงจนปัจจุบัน ได้ชื่อเรียกว่า IEEE 1076-1993 หรือ วีเอชดีแอล 1993 การที่ทางดีโอดีในขณะนั้นเป็นลูกค้ารายใหญ่ของอุตสาหกรรมอิเล็กทรอนิกส์และคอมพิวเตอร์ จึงมีผู้รับโครงการต่างๆ จากดีโอดี ไปดำเนินการด้านวิจัยและพัฒนามากเพื่อที่จะให้เป็นมาตรฐานเดียวกันหมดทางดีโอดีจึงกำหนดว่าในการส่งโครงการนั้นจะต้องเขียนอยู่ในรูปของภาษาวีเอชดีแอลเท่านั้น ซึ่งทำให้เกิดข้อดีต่อดีโอดีเองที่เป็นมาตรฐานเดียวกันสามารถนำไปจำลองกับเครื่องคอมพิวเตอร์ได้หลายๆระบบ

4.2 ส่วนประกอบต่างๆของภาษาวีเอชดีแอล

ในการเขียนรูปแบบบรรยายระบบดิจิตอลในมุมมองของการออกแบบลักษณะบนลงล่าง จะต้องทำความเข้าใจในเรื่องของโครงสร้างและส่วนประกอบต่างๆ ของรูปแบบภาษาวีเอชดีแอล เสียก่อน ซึ่งส่วนประกอบที่สำคัญและเป็นพื้นฐานของการเขียนมี 4 หน่วยคือ

1. หน่วยการออกแบบเอนทิตี (Entity Design Unit)
2. หน่วยการออกแบบสถาปัตยกรรม (Architecture Design Unit)
3. หน่วยการออกแบบแพ็กเก็ต (Package Design Unit)
4. หน่วยการออกแบบโครงแบบ (Configuration Design Unit)

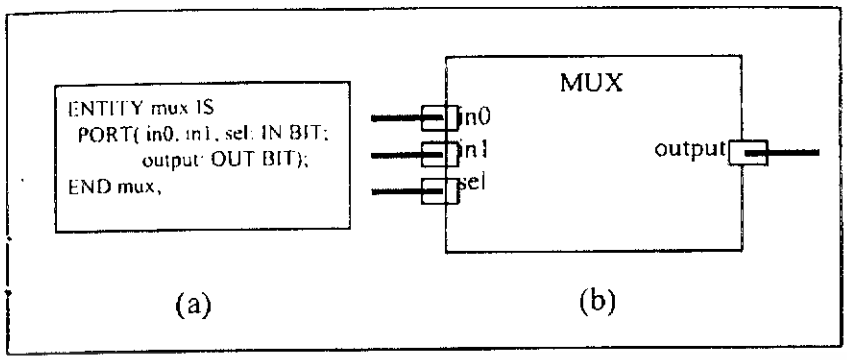
4.2.1 หน่วยการออกแบบเอนทิตี

หน่วยการออกแบบนี้เป็นส่วนที่ใช้สำหรับติดต่อระหว่างโลกภายนอกกับรูปแบบที่เขียนขึ้นที่เรียกว่า หน่วยการออกแบบเอนทิตี ในส่วนนี้ใช้กำหนดจุดเชื่อมต่อของรูปแบบกำหนดทิศทางการไหลของสัญญาณ และประเภทของค่าที่สามารถกำหนดให้กับสัญญาณตามจุดต่างๆ ของข้อมูลที่ไหลผ่านจุดต่อเหล่านั้น ภาพที่ 4.1 แสดงให้เห็นโครงสร้างอย่างง่ายๆ ของหน่วยการออกแบบเอนทิตี

```
ENTITY component_name IS
    Input and output ports
    Physical and other parameters
END [component_name];
```

ภาพที่ 4.1 แสดงโครงสร้างทั่วไปของหน่วยการออกแบบเอนทิตี

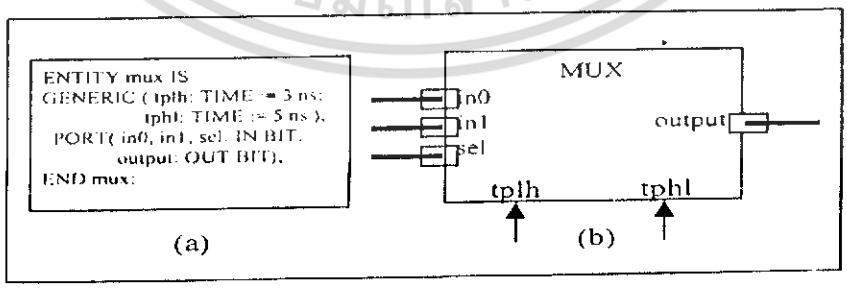
ส่วนนี้จะขึ้นต้นด้วยคำ ENTITY และ IS ระหว่างคำทั้งสองเป็นส่วนสำหรับชื่อของรูปแบบที่ต้องการจะเขียน (component_name) สำหรับการตั้งชื่อนั้นต้องเป็นไปตามกฎเกณฑ์ของภาษาหลังจากนั้นจะตามด้วยส่วนที่ใช้กำหนดช่องทางเข้าและออกของข้อมูล (input-output) รวมทั้งพารามิเตอร์อื่นๆ ส่วนนี้เรียกว่าส่วนหัว (entity header) และที่สำคัญคือหน่วยการออกแบบเอนทิตีจะต้องปิดท้ายด้วยคำว่า END และเครื่องหมายอัฒภาคเสมอ (;)



ภาพที่ 4.2 แสดงรูปแบบของมัลติเพลกซ์ (a) หน่วยการออกแบบเอนทิตีในรูปของวีเอชดีแอล (b) มุมมองของตัวเชื่อมประสาน (Interfacing)

ในภาพที่ 4.2 เป็นหน่วยการออกแบบเอนทิตี ที่บรรยายอุปกรณ์ที่มีชื่อว่ามัลติเพลก หรือ MUX ในส่วนหัวของเอนทิตี มีการกำหนดจุดต่อ 4 จุดภายใต้ชุดคำสั่ง PORT โดยที่ 3 จุดแรกเป็นจุดให้ข้อมูลไหลผ่านเข้า ได้แก่ in0 , in1 , sel ซึ่งกำหนดด้วยทิศทางการติดต่อกับภายนอกเป็นการไหลเข้าของข้อมูล (IN) ที่แสดงด้วยรูปสี่เหลี่ยมโปร่งในภาพที่ 4.2 ส่วนจุดเอาต์พุตเป็นจุดให้ข้อมูลไหลออก ซึ่งกำหนดด้วยทิศทางการติดต่อกับภายนอกเป็นการไหลออก (OUT) ที่แสดงด้วยรูปสี่เหลี่ยมทึบในภาพที่ 4.2 ส่วนประเภทของข้อมูลที่จะไหล เข้าและออก นั้นเป็นประเภท BIT ที่สามารถมีค่าได้เพียงสองค่าคือ '0' และ '1' เท่านั้น

นอกจากนั้นผู้ออกแบบยังสามารถกำหนดค่าพารามิเตอร์ทางฟิสิกส์ที่เป็นข้อมูลเพิ่มเติมอื่นๆ ลงในส่วนหัวของเอนทิตีได้อีก เช่นข้อมูลเกี่ยวกับความเร็วในการทำงานของอุปกรณ์อันได้แก่ ค่าเวลาหน่วงแพร่กระจาย (Propagation delay time) พารามิเตอร์เหล่านี้ เรียกว่า เจนเนริก (Generic) ที่กำหนดด้วยคำสั่ง GENERIC จากตัวอย่างในภาพที่ 4.3



ภาพที่ 4.3 รูปแบบมัลติเพลกซ์ที่ประกอบด้วยข้อมูลค่าเวลาหน่วงแพร่กระจาย (a) หน่วยการออกแบบเอนทิตีในรูปของวีเอชดีแอล (b) มุมมองของตัวเชื่อมประสาน

ในบางกรณีสามารถที่จะใช้ภาษาวีเอชดีแอล สร้างรูปแบบที่ปราศจากช่องทางไหลเข้าและออกของข้อมูลได้ ซึ่งส่วนใหญ่จะพบในการสร้างรูปแบบสำหรับตรวจสอบการทำงานของอีกรูปแบบหนึ่ง คือ วีเอชดีแอลสำหรับการทดสอบเปรียบเทียบ (Test bench)

```
ENTITY test_bench IS

END test_bench;
```

ภาพที่ 4.4 หน่วยการออกแบบเอนทิตีที่ไม่มีการกำหนดช่องทางที่ต่อกับภายนอก

4.2.2 หน่วยการออกแบบสถาปัตยกรรม

คือส่วนที่ใช้เขียนบรรยายพฤติกรรมของรูปแบบในมุมมองของการจำลองการทำงานพฤติกรรมต่างๆ ที่บรรยายในส่วนนี้ขึ้นอยู่กับข้อมูลที่ผ่านเข้าและออกตรงช่องทางตลอดจนพารามิเตอร์ต่างๆ ที่กำหนดในหน่วยการออกแบบเอนทิตี ภาพที่ 4.5 แสดงให้เห็นถึงโครงสร้างอย่างง่ายๆ ของหน่วยการออกแบบสถาปัตยกรรม

```
ARCHITECTURE identifier OF component_name IS
[declaration]
BEGIN
   specification of the functionality of the
   component in terms of its input lines and as
   influenced by physical and other parameters
END [identifier];
```

ภาพที่ 4.5 แสดงโครงสร้างโดยทั่วไปของหน่วยการออกแบบสถาปัตยกรรม

ส่วนของหน่วยการออกแบบสถาปัตยกรรมเริ่มต้นด้วยคำ ARCHITECTURE และตามด้วยชื่อ (identifier) สิ่งที่ต้องการกำหนดลงไปได้แก่ สิ่งที่แสดงให้เห็นว่า ARCHITECTURE นั้นใช้บรรยายหน่วยการออกแบบเอนทิตีใดๆ (OF <entity design unit> IS) ส่วนที่อยู่ระหว่าง ARCHITECTURE และ BEGIN เป็นพื้นที่ส่วนประกาศหน่วยของสถาปัตยกรรมกำหนด (architecture declarative area) ที่เป็นเพียงส่วนเผื่อเลือก (option) ในบริเวณนี้สามารถใช้เขียนประกาศกำหนดค่าต่างๆ ที่จะนำไปใช้ภายในสถาปัตยกรรมนั้นได้ อาทิเช่นประเภท (type) ต่างๆ

(ตัวอย่างเช่น bit, bit_vector), สัญญาณ (signal), ตัวคงที่ (constant), โปรแกรมย่อย (ได้แก่ function และ procedure) และอุปกรณ์ (component) ส่วนที่ใช้บรรยายความสัมพันธ์ระหว่างข้อมูลที่ไหลเข้า และไหลออกของรูปแบบ (สัญญาณที่กำหนดในชุดคำสั่ง PORT) นั้นจะถูกบรรยายในบริเวณเนื้อที่ระหว่างคำว่า BEGIN กับ END ของหน่วยการออกแบบสถาปัตยกรรม และนอกจากนั้นชุดคำสั่งทุกคำสั่งที่อยู่ภายในบริเวณนี้จะเป็นชุดคำสั่งแบบแข่งขนาน (concurrent statement) เท่านั้น หน่วยการออกแบบสถาปัตยกรรมนั้นๆ ที่เป็นส่วนเผื่อเลือกโดยทั่วไปการเขียนรูปแบบระบบดิจิตอลด้วยภาษาวีเอชดีแอล สามารถเขียนได้ในลักษณะต่างๆ ดังนี้

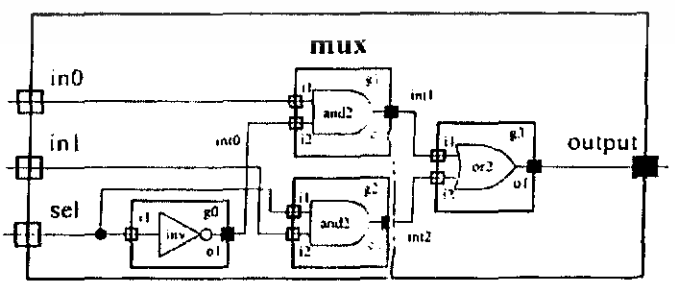
- ประเภทการไหลของข้อมูล (Dataflow description)
- ประเภทฟฤติกรรม (Behavioral description)
- ประเภทโครงสร้าง (Structure description)
- ประเภทผสม (Mixed model description)

```
ARCHITECTURE data_flow OF mux IS
BEGIN
   output <= ((NOT sel) AND in0) OR (sel AND in1);
END data_flow;
```

ภาพที่ 4.6 แสดงหน่วยการออกแบบสถาปัตยกรรมของมัลติเพลกซ์ตามฟังก์ชันบูลีน

Output = (sel.in0)+(sel.in1)

ภาพที่ 4.6 ส่วนที่บรรยายความสัมพันธ์ระหว่างข้อมูลที่ไหลเข้า (in0, in1) กับข้อมูลที่ไหลออก (output) ประกอบด้วยชุดคำสั่งแบบแข่งขนานเพียงชุดเดียว ซึ่งเขียนเป็นประเภทการไหลของข้อมูลของมัลติเพลกซ์ หรือ ระดับการถ่ายโอนข้อมูลระหว่างเรจิสเตอร์ (RTL: Register Transfer Level)



ภาพที่ 4.7 แสดงโครงสร้างภายในสถาปัตยกรรมของมัลติเพลกซ์

ภาพที่ 4.7 เป็นหน่วยการออกแบบสถาปัตยกรรมของมัลติเพลกซ์ประเภทโครงสร้าง โดยใช้อินเวอร์เตอร์ (inv ที่ตำแหน่ง g0), แอนด์เกต 2 อินพุตจำนวน 2 ตัว (and ที่ตำแหน่ง g1 และ g2) และออร์เกต 2 อิพุต (or2 ที่ตำแหน่ง g3) มาสร้างตามฟังก์ชันบูลีนของภาพที่ 4.6

```
ARCHITECTURE struc OF mux IS
  COMPONENT inv
  PORT ( i1 : IN BIT ; o1 : OUT BIT );
   COMPONENT and2
   PORT ( i1, i2 : IN BIT ; o3 : OUT BIT );
   COMPONENT or2
   PORT ( i1, i2 : IN BIT ; o1 : OUT BIT );
END COMPONENT;
   SIGNAL int0, int1, int2 : BIT;
BEGIN
   g0 : inv   PORT MAP (i1 => sel, o1 => int0);
   g1 : and2 PORT MAP (i1 => in0, i2 => int0, o1 => int1);
   g3 : or2   PORT MAP (i1 => int1, i2 => int2, o1 => ouput);
END struc;
```

ภาพที่ 4.8 หน่วยการออกแบบสถาปัตยกรรมของมัลติเพลกซ์ประเภทโครงสร้าง

```
ARCHITECTURE behav OF mux IS
BEGIN
     PROCESS (in0, in1, sel)
     BEGIN
          IF (sel = '0') THEN output <= in0;
          ELSE output <= in1;
          END IF;
   END PROCESS;
END behav;
```

ภาพที่ 4.9 หน่วยการออกแบบสถาปัตยกรรมของมัลติเพลกซ์ประเภทพฤติกรรม

ไม่ว่าเขียนบรรยายส่วนของสถาปัตยกรรมของมัลติเพลกซ์ในลักษณะของ ประเภทพฤติกรรม ประเภทการไหลของข้อมูล ประเภทโครงสร้างหรือประเภทผสมที่นำเอาแต่ละประเภทมาเขียนไว้ในส่วนของสถาปัตยกรรม ก็ตามต่างก็มีพฤติกรรมเดียวกัน และจะให้ผลลัพธ์จากการจำลองการทำงานที่เหมือนกัน ซึ่งนี้ก็เป็นข้อดีของภาษาวีเอชดีแอล

4.2.3 หน่วยการออกแบบแพ็กเก็ต

ข้อมูลต่างๆ ตลอดจนโปรแกรมย่อย ที่เป็นประโยชน์ต่อการเขียนรูปแบบบรรยายระบบดิจิตอล สามารถเก็บไว้ในส่วนของแพ็กเก็ตได้และข้อมูลเหล่านี้สามารถเรียกไปใช้ได้โดยหน่วยการออกแบบเอนทิตี หน่วยการออกแบบสถาปัตยกรรม หรือจากหน่วยการออกแบบแพ็กเก็ตอื่นๆ นอกจากนั้นสิ่งที่นิยมทำกันมากคือรูปแบบมาตรฐานต่างๆ เช่นอุปกรณ์มาตรฐาน (เช่น ไอซีตระกูล 78XX เป็นต้น) จะถูกเก็บไว้ในแพ็กเก็ต ที่ทุกคนสามารถเข้าถึง โดยปกติแล้วแพ็กเก็ตจะแบ่งออกเป็น 2 ส่วนคือ การประกาศแพ็กเก็ต (Package declaration) และ ส่วนของบอดี้แพ็กเก็ต (Package body) เนื่องจาก แพ็กเก็ตถูกสร้างขึ้นเป็นส่วนแยกต่างหากออกจากรูปแบบที่กำลังเขียนอยู่ ฉะนั้นการที่นำแพ็กเก็ตไปใช้นั้นจะต้องมีการเชื่อมโยงหรืออ้างอิงเสียก่อน ซึ่งในภาษาวีเอชดีแอลสามารถกระทำได้ด้วยชุดคำสั่ง USB

4.2.3.1 PACKAGE DECLARATION

ส่วนที่มีความสำคัญที่สุดของแพ็กเก็ต (ถ้ามองในแง่ของการนำไปใช้จากภายนอก) ได้แก่ส่วนการประกาศแพ็กเก็ต เพราะจะเป็นส่วนที่กำหนดชื่อของสิ่งที่ประกาศอยู่ภายในแพ็กเก็ตสำหรับนำไปใช้ภายนอกตัวของแพ็กเก็ตเอง สิ่งใดๆ ถูกประกาศในส่วนของ ส่วนบอดี้แพ็กเก็ตแต่ไม่ถูกประกาศในส่วนการประกาศแพ็กเก็ตจะไม่สามารถถูกนำค่าและพฤติกรรมไปใช้ส่วนนอกได้ ซึ่งสามารถเปรียบเทียบได้กับสิ่งที่ประกาศไว้ในส่วนของการประกาศเอนทิตีคือ จุดเชื่อมต่อ หรือ พอร์ต ที่มีหน้าที่ติดต่อกับโลกภายนอก ฉะนั้นโดยทั่วไปแล้วแพ็กเก็ตสามารถสร้างขึ้นได้โดยไม่จำเป็นต้องมีส่วนบอดี้ และยังสามารถถูกนำไปใช้จากรูปแบบภายนอกได้เช่น ใช้สำหรับประกาศ ชนิด (Type) หรือ สัญญาณ เช่นเดียวกันกับส่วนบอดี้แพ็กเก็ตที่ไม่จำเป็นต้องมีส่วนของการประกาศแพ็กเก็ต แต่แพ็กเก็ตนั้นจะไม่สามารถถูกนำไปใช้จากรูปแบบอื่นได้

```
PACKAGE package_name IS
   Package_declarative_part
END package_name;
```

ภาพที่ 4.10 แสดงโครงสร้างโดยทั่วไปของส่วนการประกาศแพ็กเก็ต

4.2.3.2 PACKAGE BODY

โครงสร้างที่ประกอบด้วยคำสั่งต่างๆ ในรูปของคำสั่งลำดับที่ใช้บรรยายฟังก์ชันการทำงานของโปรแกรมย่อย (Subprogram) ทั้งหลายที่ชื่อของโปรแกรมย่อยนั้นๆ ที่ถูกประกาศไปในส่วนของการประกาศแพ็กเก็ตแล้วจะถูกเก็บไว้ในส่วนบอดี้แพ็กเก็ต ทั้งนี้รวมทั้งการกำหนดค่าคงที่ต่างๆ อันได้แก่ตัวคงที่ที่ถูกประกาศชื่อก่อนในส่วนของการประกาศแพ็กเก็ต แต่ถูกกำหนดค่าในส่วนของบอดี้แพ็กเก็ต ฉะนั้นส่วนบอดี้แพ็กเก็ตจึงไม่จำเป็นต้องมี ถ้าในส่วนของการประกาศแพ็กเก็ตไม่มีการประกาศชื่อที่เป็นโปรแกรมย่อย หรือ ค่าคงที่ การเขียนบอดี้แพ็กเก็ตนั้นเป็นไปตามกฎเกณฑ์ที่แสดงในภาพที่ 4.11

```
PACKAGE BODY package_name IS
        declarative part
END package_name;
```

ภาพที่ 4.11 ในโครงสร้างของบอดี้แพ็กเก็ต

4.2.4 หน่วยการออกแบบโครงแบบ

ดังที่ทรายกันแล้วว่ารูปแบบหนึ่งของระบบดิจิตอลไม่ว่าจะเป็นอะไร จะมีหน่วยการออกแบบเอนทิตีได้เพียงหนึ่งเดียวเท่านั้น แต่ในขณะที่หน่วยการออกแบบเอนทิตีหนึ่งหน่วยนี้อาจจะมีสถาปัตยกรรมที่เป็นหน่วยรองได้หลายหน่วย ดังนั้นจะต้องมีหน่วยการออกแบโครงแบบมาเพื่อกำหนดการใช้โครงแบบ (Configuration) ประกอบเอนทิตีกับหน่วยการออกแบบสถาปัตยกรรมหน่วยไหนเข้าด้วยกัน

```
CONFIGURATION identifier OF entity_name IS
        Configuration_declarative_part
END ;
```

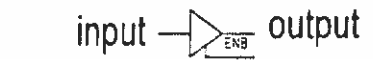
ภาพที่ 4.12 โครงสร้างโดยทั่วไปของหน่วยการออกแบบโครงแบบ

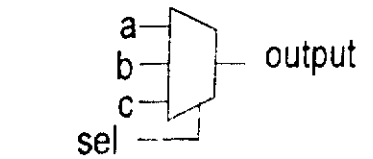
4.3 ภาษาวีเอชดีแอลเพื่อการสังเคราะห์

ภาษาวีเอชดีแอล เป็นภาษาที่เขียนเพื่อจำลองการทำงานของวงจร ซึ่งในบางรูปแบบการเขียนไม่สามารถที่จะนำไปสังเคราะห์ได้ทั้งหมด ดังนั้นถ้าต้องการเขียนเพื่อนำไปสังเคราะห์ ควรหลีกเลี่ยงรูปแบบต่างๆ ที่ไม่สามารถนำไปสังเคราะห์ได้ ในที่นี้ขึ้นอยู่กับความสามารถของโปรแกรมที่ใช้สังเคราะห์แต่ละโปรแกรม ดังนั้นในหัวข้อนี้จะแสดงตัวอย่างของการเขียนโมเดลในรูปแบบต่างที่สามารถนำไปสังเคราะห์ซึ่งยึดหลักการเขียนตาม ViewSynthesis User's Guide ของโปรแกรม Viewlogic ซึ่งเป็นโปรแกรมที่ใช้ในการสังเคราะห์วงจรทั้งหมดในการออกแบบไทโครคอนโทรลเลอร์โดยแบ่งออกเป็น 2 กลุ่ม คือ เกตและฟลิปฟลอปประเภทต่างๆ

4.3.1 ตัวอย่างรูปแบบการเขียนเกตพื้นฐาน

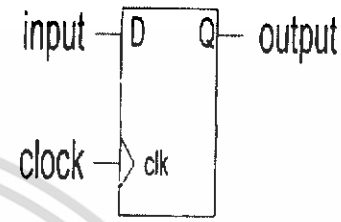
```
SIGNAL a, b, c, d, input, output : vlbit_1d(3 DOWNTO 0);
SIGNAL sel  : vlbit_1d(1 DOWNTO 0);
SIGNAL enb : vlbit;
...
output <= a AND b;
...
output <= a OR b;
...
output <= a XOR b;
...
output <= NOT input;
...
output <= input WHEN enb='1' ELSE "ZZ";
...
WITH sel SELECT
        output <=    a WHEN "00" ELSE
                     b WHEN "01" ELSE
                     c;
```

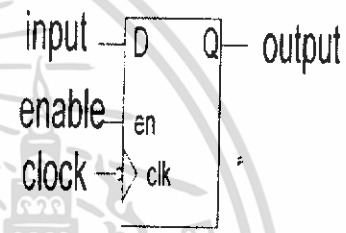


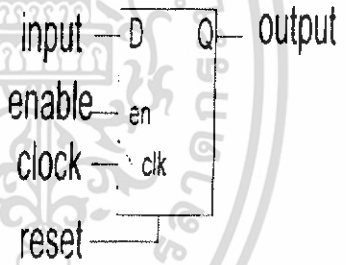


4.3.2 ตัวอย่างรูปแบบการเขียนฟลิปฟลอปพื้นฐาน

```
SIGNAL input, output : vlbit_1d(3 DOWNTO 0);
SIGNAL clock, enable, reset : vlbit;
...

PROCESS BEGIN
        WAIT UNTIL PRISING(clock);
                output <= input;
END PROCESS;
...

PROCESS BEGIN
        WAIT UNTIL PFALLING(clock);
                output <= input;
END PROCESS;
...

PROCESS BEGIN
WAIT UNTIL PRISING(clock) OR (reset='1');
 IF (reset = '1') THEN output <= "00";
 ELSIF (enble = '1') THEN ouput <= input;
 END IF;
END PROCESS;
```

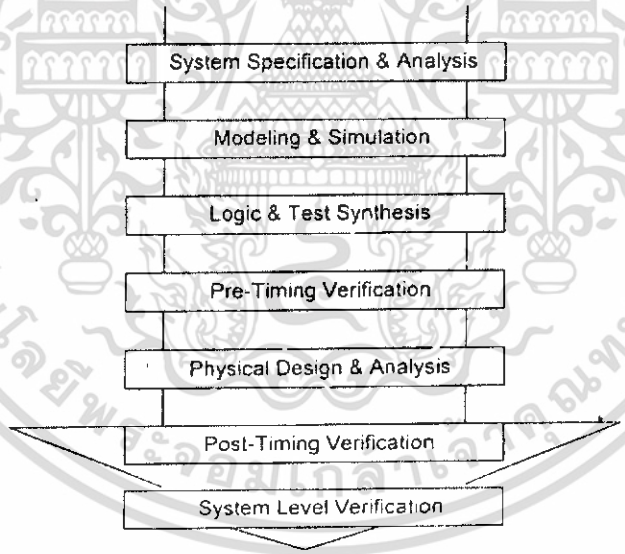






4.4 การออกแบบจากบนลงล่าง

ในการพัฒนาวงจรรวมดิจิตอลขนาดใหญ่ที่มีความซับซ้อน เช่น วงจรรวม (ASIC: Application Specific Integrated Circuit) วิศวกรหรือผู้ออกแบบมักจะมองการออกแบบให้อยู่ในรูปของของการบล็อกไดอะแกรมเสียก่อน ก่อนที่จะวิเคราะห์ให้ลึกถึงรายละเอียดต่อไป ซึ่งภาษาวีเอชดีแอลนั้นอนุญาตให้อธิบายการทำงานของแต่ละบล็อกและวิเคราะห์การทำงานแก้ไขและปรับปรุงการทำงานจากผลที่วิเคราะห์เพื่อให้ได้การทำงานตามที่ต้องการ และเพิ่มเติมในรายละเอียดทีละชั้นนี้คือ หลักการออกแบบจากบนลงล่าง (Top-Down Design) ถ้าทดลองเปรียบเทียบกับการออกแบบจากล่างขึ้นบน (Botom-Up Design) จะเห็นได้ว่าการออกแบบจากล่างขึ้นบนจะใช้เวลาการออกแบบมากกว่า 90% เพราะเป็นการวาดวงจรด้วยอุปกรณ์ต่างๆ (Schematic capture) ที่ประกอบกันเข้าเป็นวงจรที่ต้องการออกแบบจำลองการทำงานตรวจสอบความถูกต้อง ซึ่งใช้เวลามากและถ้าวงจรที่ต้องการออกแบบมีความซับซ้อนก็จะเป็นเรื่องที่ยากมากให้การออกแบบในลักษณะนี้ ดังนั้นการใช้ภาษาวีเอชดีแอลกับหลักการออกแบบจากบนลงล่างจึงเป็นทางออกให้กับวิศวกรออกแบบที่จะสามารถออกแบบและพัฒนาวงจรที่มีความซับซ้อนได้มากขึ้น และช่วยลดเวลาและค่าใช้จ่ายในการออกแบบ



ภาพที่ 4.13 ขั้นตอนการออกแบบจากบนลงล่าง

จากภาพที่ 4.13 แสดงให้ถึงขั้นตอนการออกแบบจากบนลงล่าง ทั้งนี้ในทางปฏิบัติอาจจะมีข้อแตกต่างไปจากนี้บ้างเล็กน้อย ก็เนื่องจากขั้นตอนของการผลิต (Implementation) สามารถกระทำได้หลายๆ เทคโนโลยี เช่น พีแอลดี (PLD: Programmable Logic Device) อันได้แก่

พีแอลเอ (PLA: Programmable Logic Array), เอฟพีจีเอ (FPGA: Filed Programmable Gate Array), ซีพีแอลดี (CPLD: Cell Programmable Logic Device) เป็นต้น นอกนั้นยังมีเซมิคัสตัมไอซี (Semi Custom IC) ได้แก่ เกตอะเรย์ (Gate array), เซลล์มาตรฐาน (Standard Cell) ขั้นตอนของการออกแบบจากบนลงล่างมีรายละเอียดดังนี้

1. ขั้นตอนการสร้างข้อกำหนดของความต้องการ และวิเคราะห์ระบบเพื่อหาแนวความคิดและหลักการ (Idea and concept) ในการแก้ปัญหา

2. ขั้นตอนการการเขียนรูปแบบของระบบที่ต้องการออกแบบโดยใช้ภาษาวีเอชดีแอล หรือ ภาษาเอชดีแอลอื่นๆ สำหรับบรรยายพฤติกรรมการทำงานพร้อมทั้งจำลองการทำงานเพื่อเปรียบเทียบและตรวจสอบความถูกต้องกับข้อกำหนด

3. หลังจากที่ได้หลักการขั้นต้นพร้อมกับแนวความคิดที่ผ่านการตรวจสอบแล้ว หลักการนี้จะถูกเพิ่มเติมในรายละเอียดลงมาเป็นลำดับขั้นที่สองจนกระทั่งอยู่ในระดับที่จะนำไปผลิตวงจร หรือ สังเคราะห์ในขั้นตอนนี้เองเทคโนโลยีที่จะมารองรับวงจรออกแบบจะถูกกำหนดขึ้น และระบบช่วยการออกแบบจะสังเคราะห์วงจรที่ได้จากรูปแบบที่เขียนขึ้นให้อยู่ในรูปของวงจรที่ประกอบด้วยอุปกรณ์อิเล็กทรอนิกส์หรือวงจรในระดับเกต และการเชื่อมต่อระหว่างกันของอุกรณ์เหล่านั้น หรือไม่ก็อยู่ในรูปของเน็ทลิสท์ (Netlist) ที่สามารถนำไปผลิตลงบนอุปกรณ์อื่นได้

4. หลังจากการสังเคราะห์วงจรให้อยู่ในรูประดับเกตหรือเน็ทลิสท์แล้วข้อมูลที่ได้จากผู้ผลิตอุปกรณ์วงจรนั้น นอกจากจะเป็นข้อมูลสำหรับจำลองการทำงานในเรื่องของความถูกต้องของฟังก์ชันแล้วยังมีข้อมูลที่เกี่ยวกับเวลาด้วย ซึ่งเป็นความจริงที่ว่าอุปกรณ์ทางอิเล็กทรอนิกส์ทุกชิ้นจะมีเวลาหน่วงของการแพร่กระจาย (Propagation delay time) เสมอถึงแม้จะเป็นเวลาที่น้อยมากในระดับนาโนวินาที แต่ถ้าภายในวงจรหนึ่งประกอบด้วยเกตของฟังก์ชันต่างๆ จำนวน 10,000 เกตขึ้นไป เวลาดังกล่าวนี้จะสะสมกันมากขึ้นจนอาจจะทำให้การทำงานของวงจรรวมทั้งหมดผิดไป หรือไม่สามารถทำงานในย่านความถี่สัญญาณนาฬิกาที่สูงได้

5. ขั้นตอนของการผลิตเป็นวงจรจริง (Technology and device mapping) โดยนำข้อมูลที่ได้จากการสังเคราะห์มาผลิตซึ่งอาจจะอยู่ในรูปของแผงวงจรไฟฟ้าที่ประกอบด้วยอุปกรณ์หลายๆ ชิ้น หรืออยู่ในรูปของวงจรรวม (ASIC)

6. หลังจากที่ได้วงจรจริงมาแล้วยังต้องมีความจำเป็นที่ต้องตรวจสอบการทำงานที่คำนึงถึงเวลาด้วย เพื่อความถูกต้องของวงจรครั้งสุดท้ายก่อนที่จะนำไปรวมเข้ากับอุปกรณ์อื่นๆให้เป็นระบบดิจิตอลเพราะในขั้นตอนนี้วงจรที่ออกแบบจะประกอบด้วยอินพุตและเอาต์พุตแพด (Pad) ซึ่งเป็นจุดต่อสำหรับรับและส่งสัญญาณกับภายนอก

7. หลังจากที่นำวงจรที่ออกแบบรวมเข้ากับอุปกรณ์อื่นๆให้เป็นระบบดิจิตอลแล้วนั้นจะต้องทดสอบการทำงานรวมทั้งระบบร่วมกับอุปกรณ์อื่นๆอีกครั้งเป็นการควบคุมคุณภาพของผลิตภัณฑ์

บทที่ 5

การทดลองและผลการทดลอง

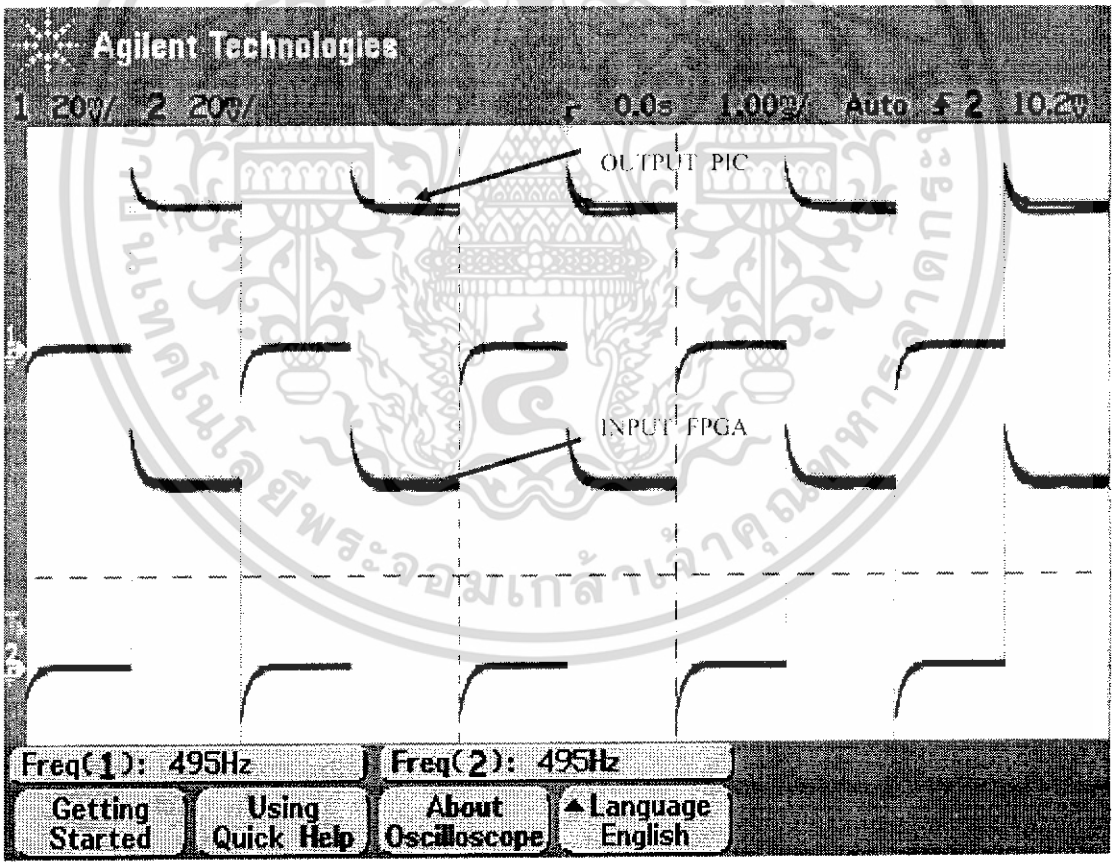
5.1 จุดประสงค์ในการทดลอง

1. เพื่อทดสอบการทำงานของตัว FPGA ได้ทำงานตามค่า ความถี่และเวลา ที่ป้อนให้ได้จริงและถูกต้อง

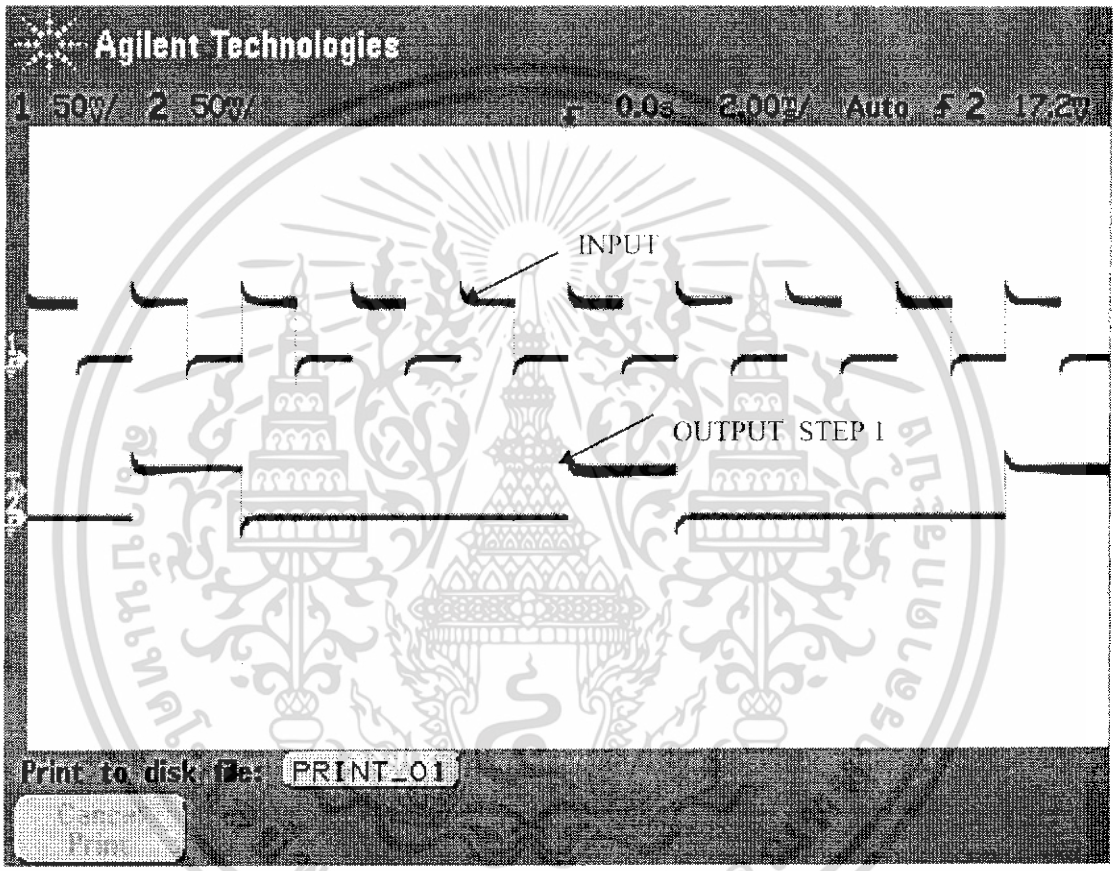
2. เพื่อทำการทดลองสัญญาณ และดูรูปแบบของสัญญาณ ณ จุดต่างๆที่ตั้งไว้ว่าได้ออกมาตามค่า ความถี่และเวลา ที่ได้ตั้งไว้ได้จริงและถูกต้อง

3. เพื่อทำการวัดหาค่าแรงบิดและความเร็วรอบ

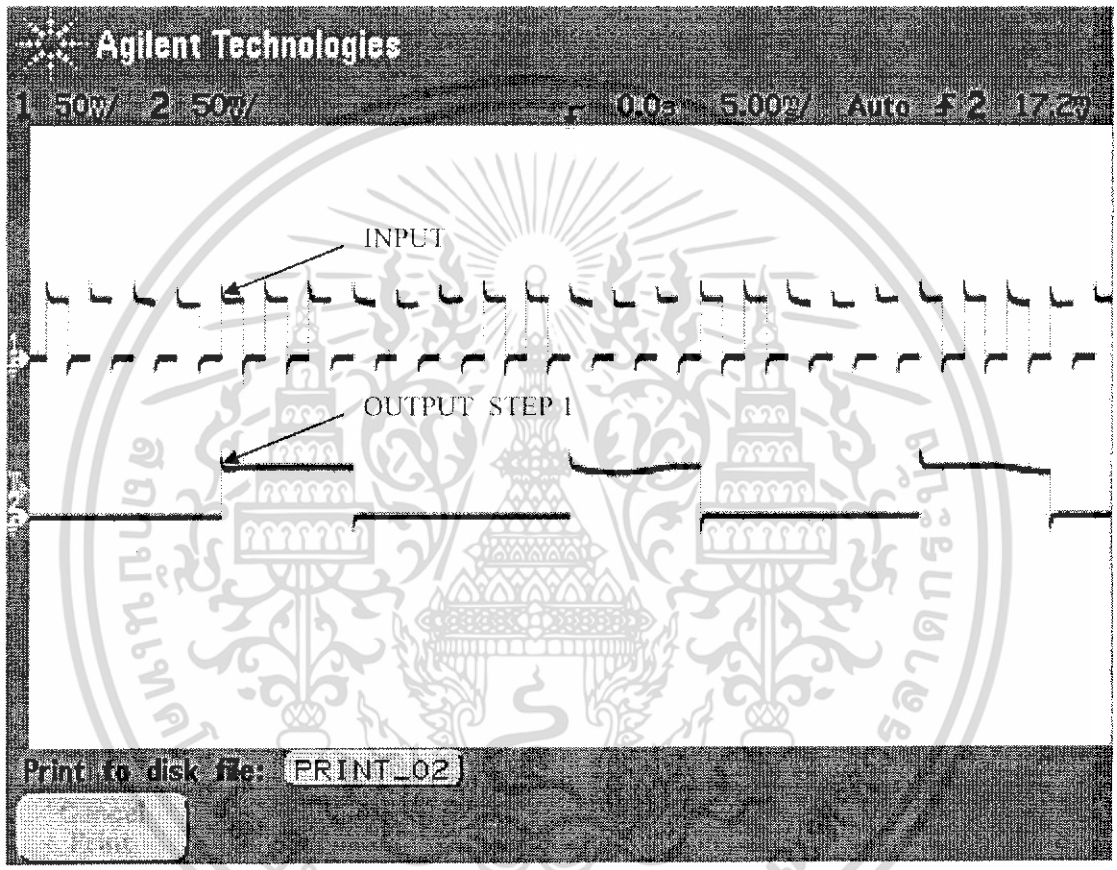
5.2 ผลการทดลอง



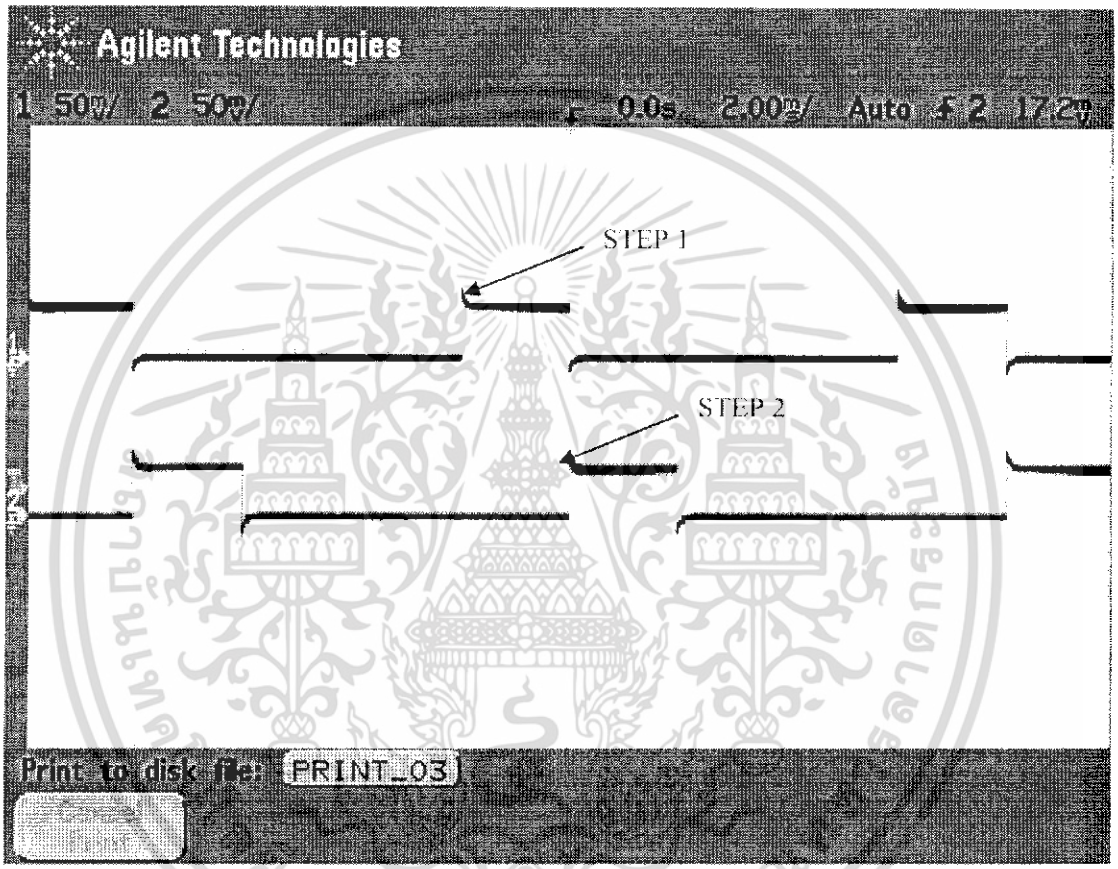
ภาพที่ 5.1 แสดงการเปรียบเทียบสัญญาณระหว่าง OUTPUT ของ PIC กับ INPUT ของ FPGA



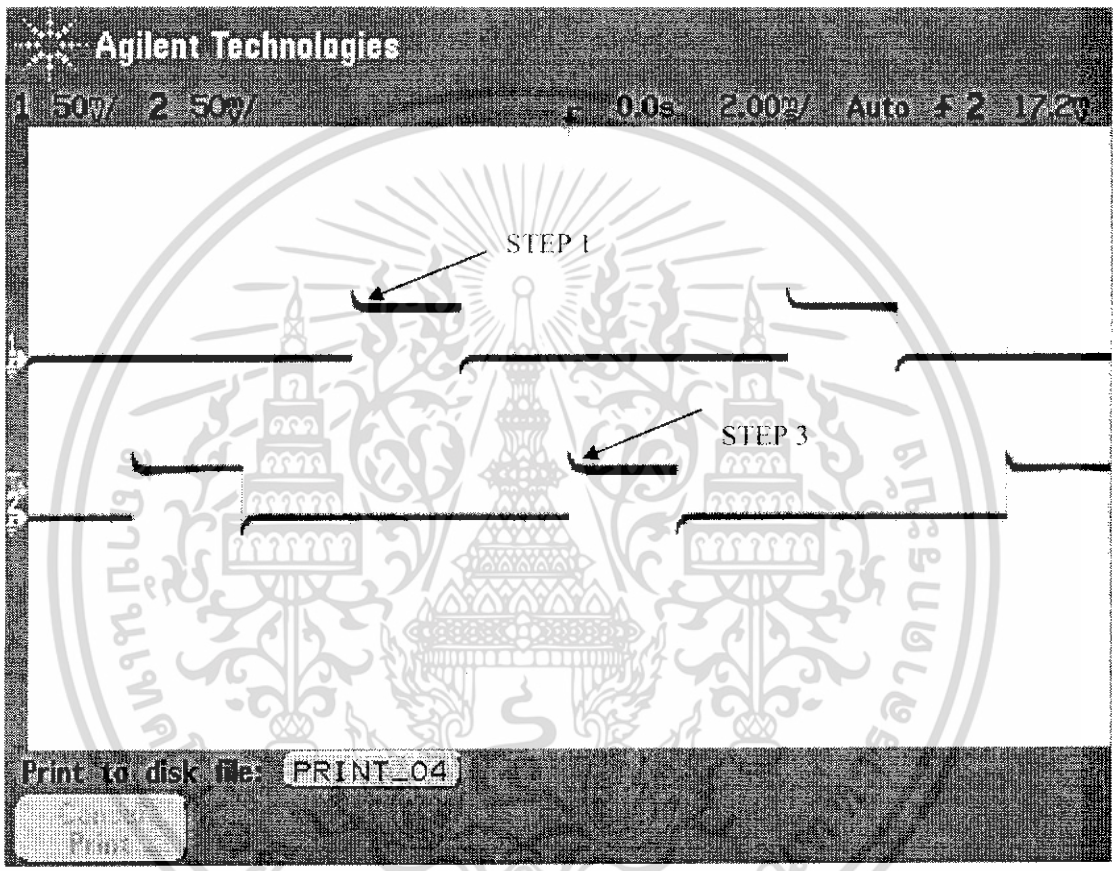
ภาพที่ 5.2 แสดงการเปรียบเทียบสัญญาณระหว่างสัญญาณ INPUT กับสัญญาณ OUTPUT ที่ STEP ที่1 ของ FPGA แบบ Full Step



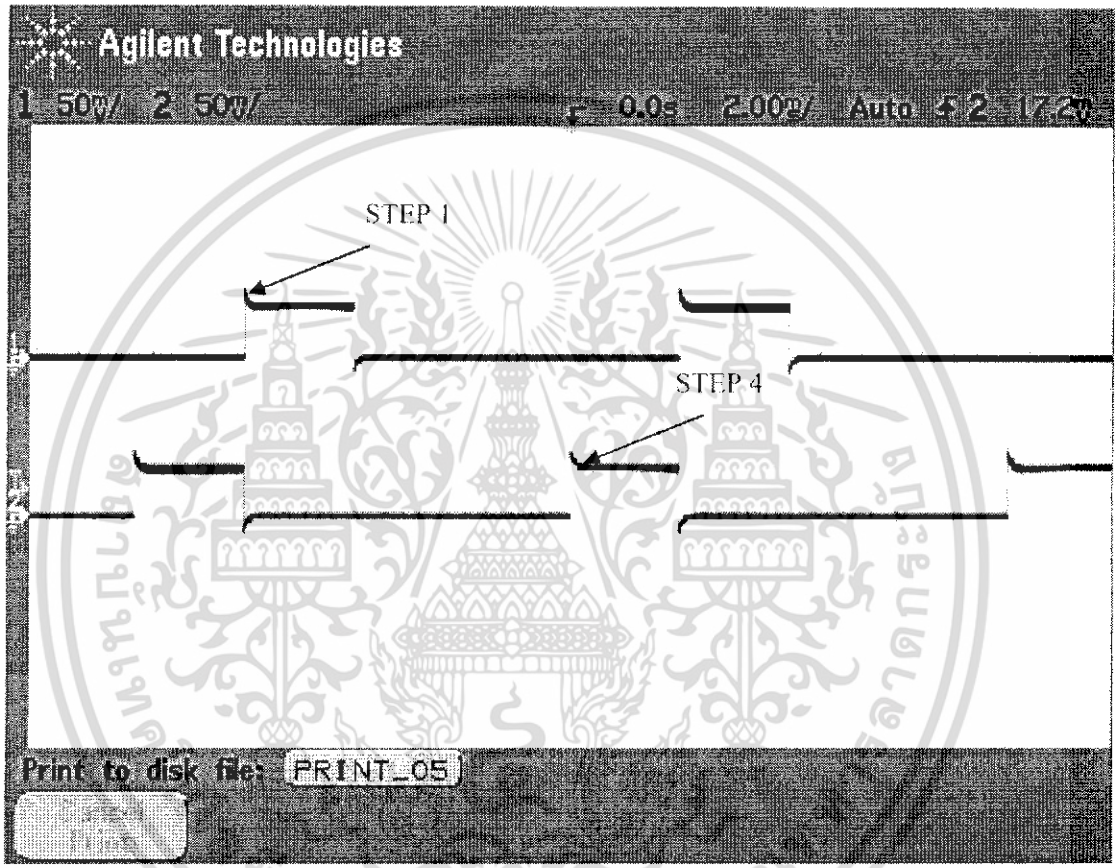
ภาพที่ 5.3 แสดงการเปรียบเทียบสัญญาณระหว่างสัญญาณ INPUT กับสัญญาณ OUTPUT ที่ STEP ที่1 ของ FPGA แบบ Half Step



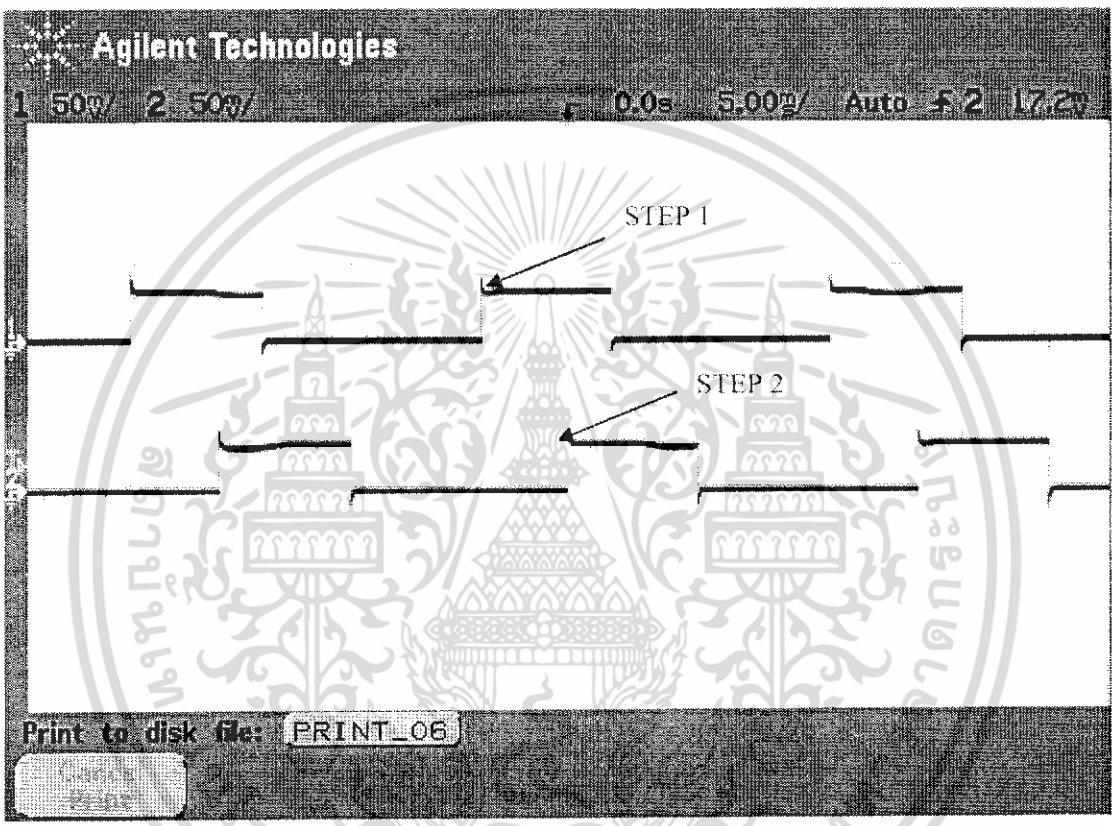
ภาพที่ 5.4 แสดงการเปรียบเทียบระหว่างสัญญาณ STEP ที่ 1 และ STEP ที่ 2 แบบ Full Step



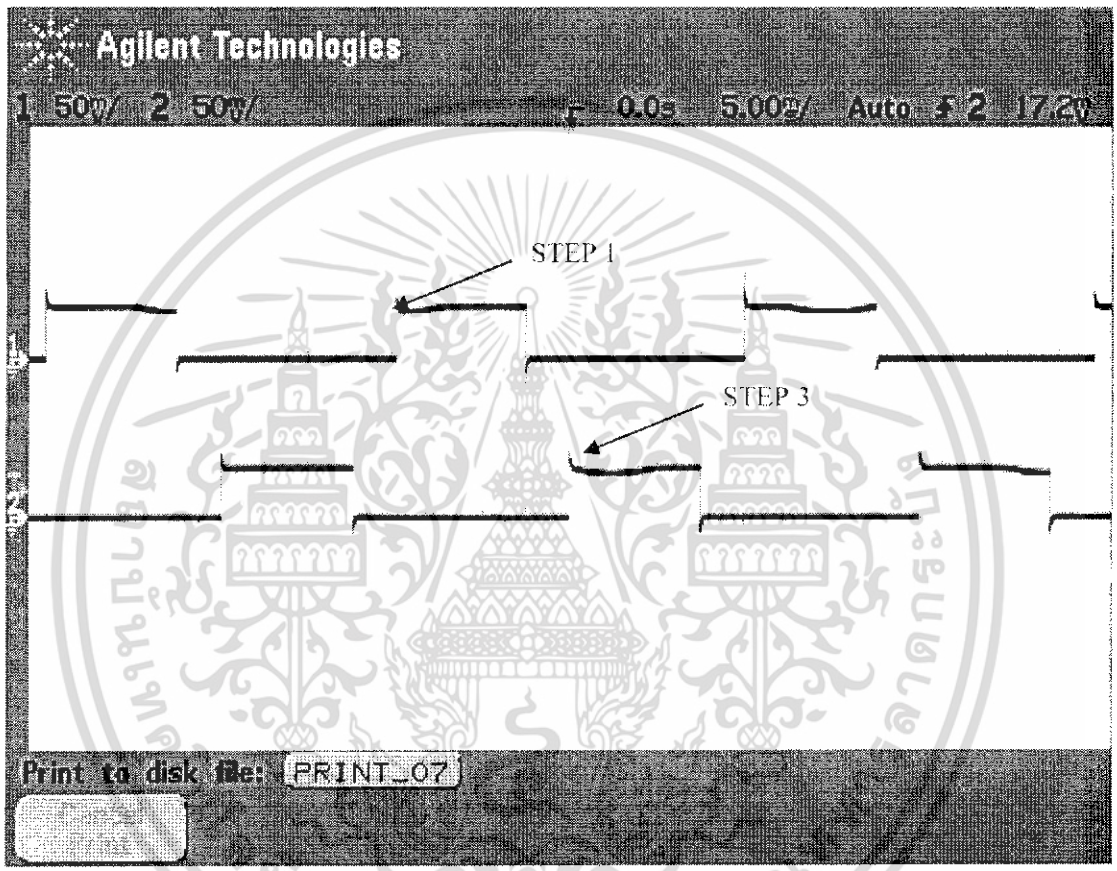
ภาพที่ 5.5 แสดงการเปรียบเทียบระหว่างสัญญาณ STEP ที่ 1 และ STEP ที่ 3 แบบ Full Step



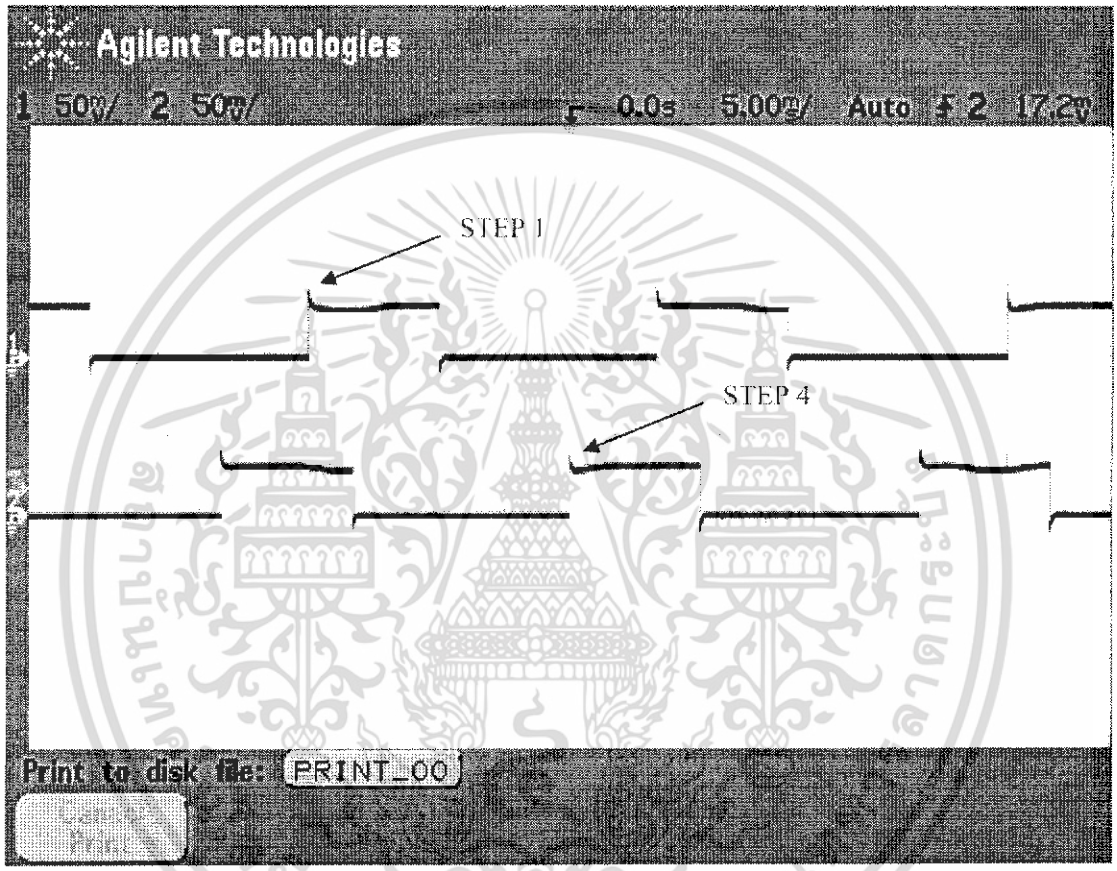
ภาพที่ 5.6 แสดงการเปรียบเทียบระหว่างสัญญาณ STEP ที่ 1 และ STEP ที่ 4 แบบ Full Step



ภาพที่ 5.7 แสดงการเปรียบเทียบระหว่างสัญญาณ STEP ที่ 1 และ STEP ที่ 2 แบบ Half Step



ภาพที่ 5.8 แสดงการเปรียบเทียบระหว่างสัญญาณ STEP ที่ 1 และ STEP ที่ 3 แบบ Half Step



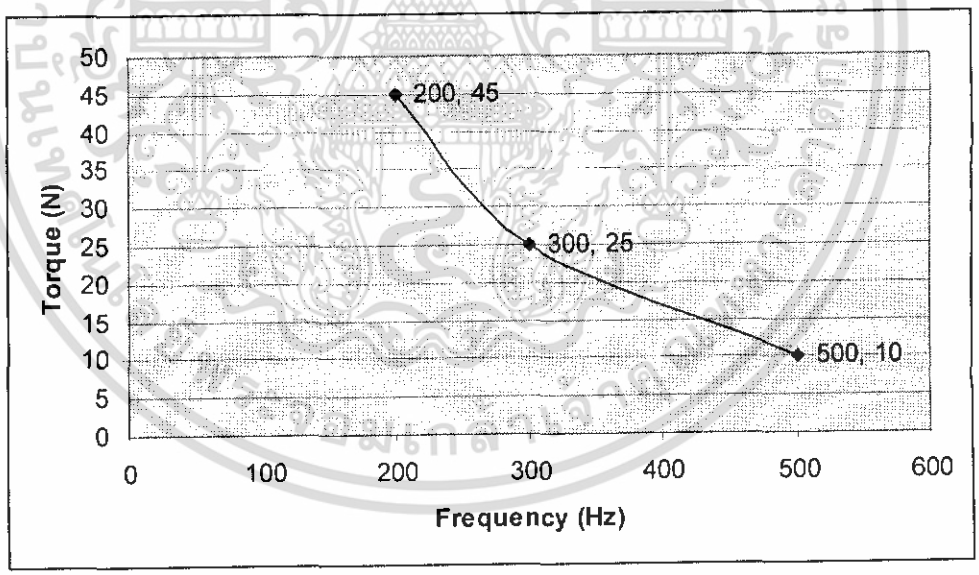
ภาพที่ 5.9 แสดงการเปรียบเทียบระหว่างสัญญาณ STEP ที่ 1 และ STEP ที่ 4 แบบ Half Step

ตารางที่ 5.1 ผลการทดลองแบบ Half Step

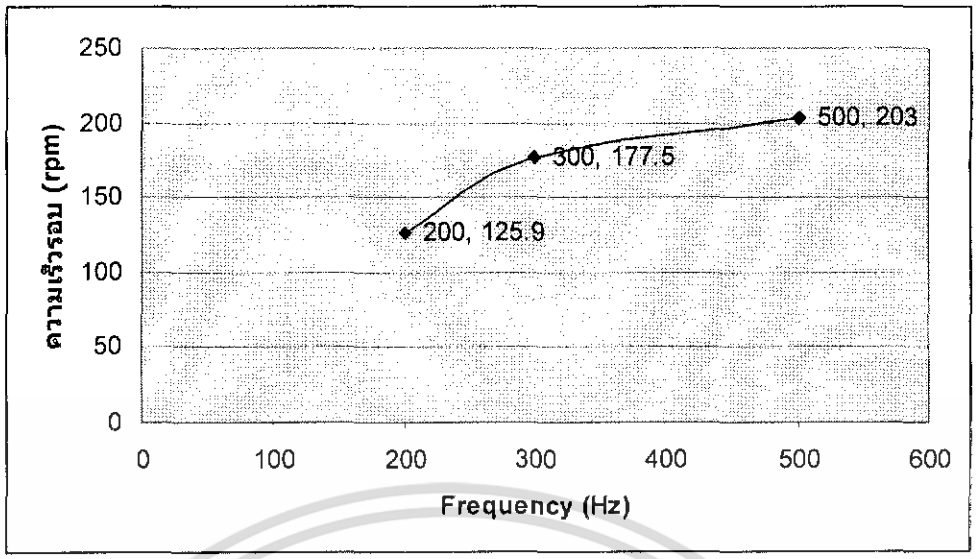
Frequency (Hz)	Torque (N)	rpm
500	10	203
300	25	177.5
200	45	125.9

ตารางที่ 5.2 ผลการทดลองแบบ Full Step

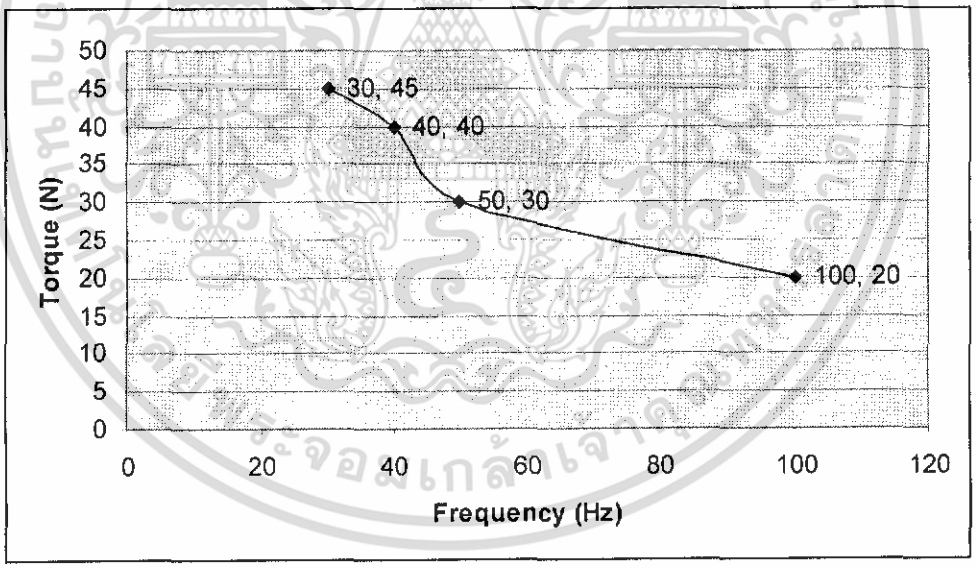
Frequency (Hz)	Torque (N)	rpm
100	20	125
50	30	55
40	40	47.9
30	45	31



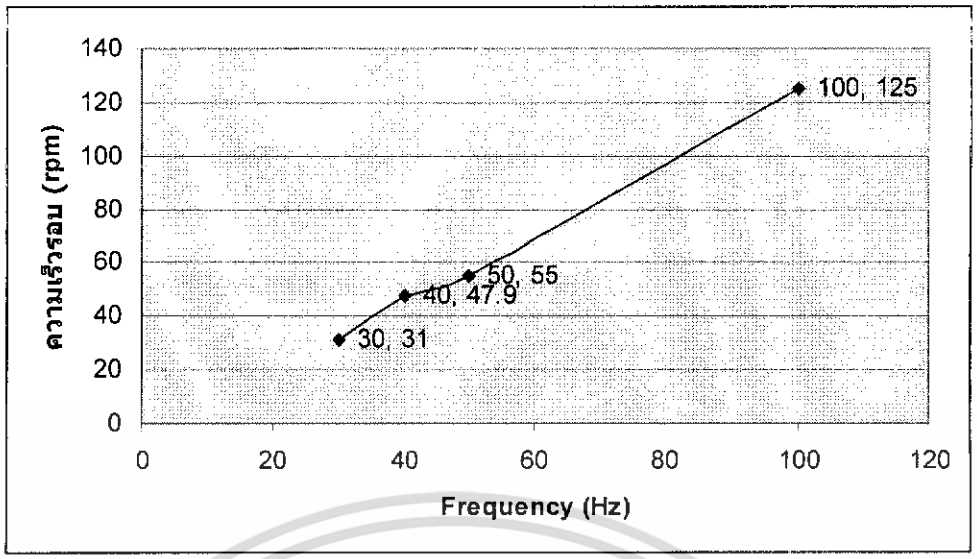
ภาพที่ 5.10 แสดงความสัมพันธ์ระหว่าง Frequency และ Torque แบบ Half Step



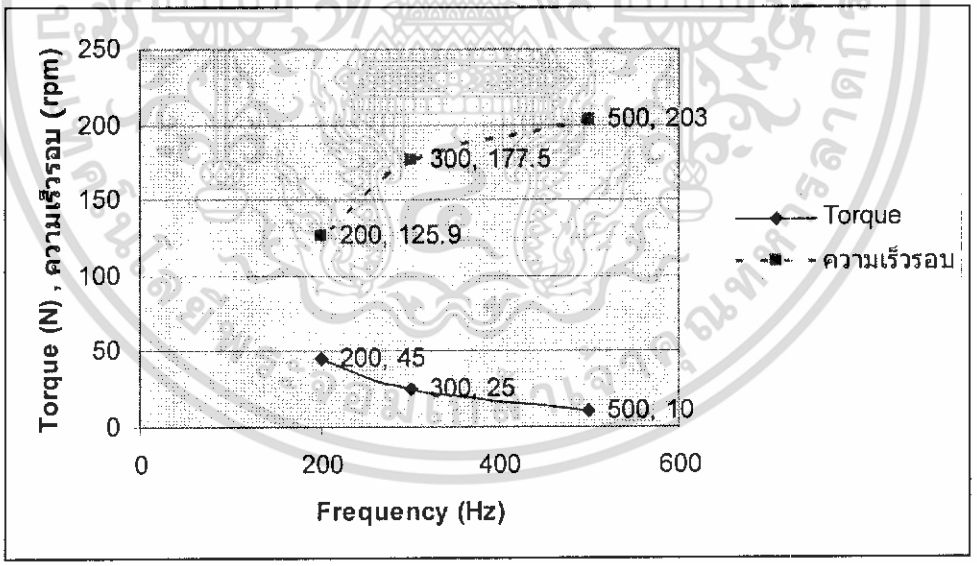
ภาพที่ 5.11 แสดงความสัมพันธ์ระหว่าง Frequency และ ความเร็วรอบ แบบ Half Step



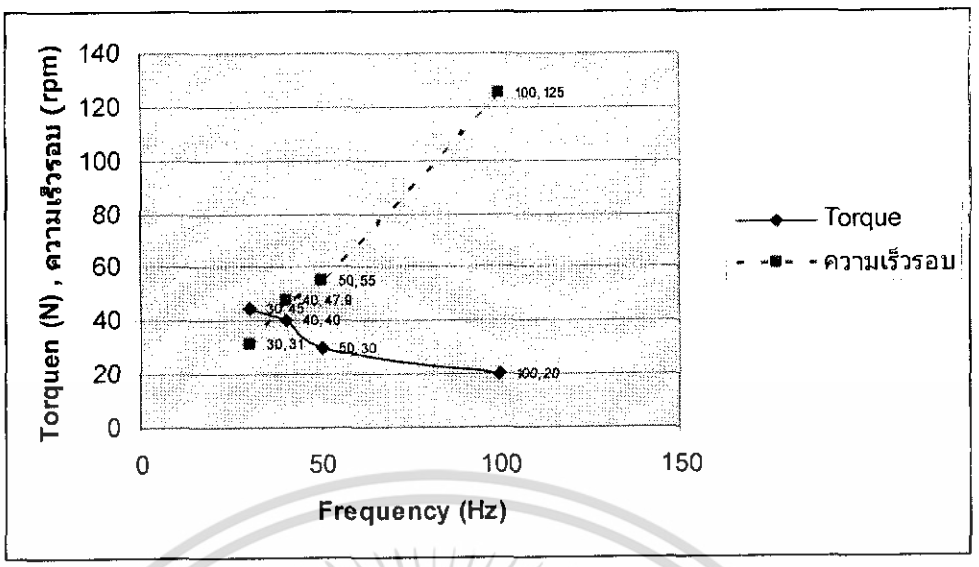
ภาพที่ 5.12 แสดงความสัมพันธ์ระหว่าง Frequency และ Torque แบบ Full Step



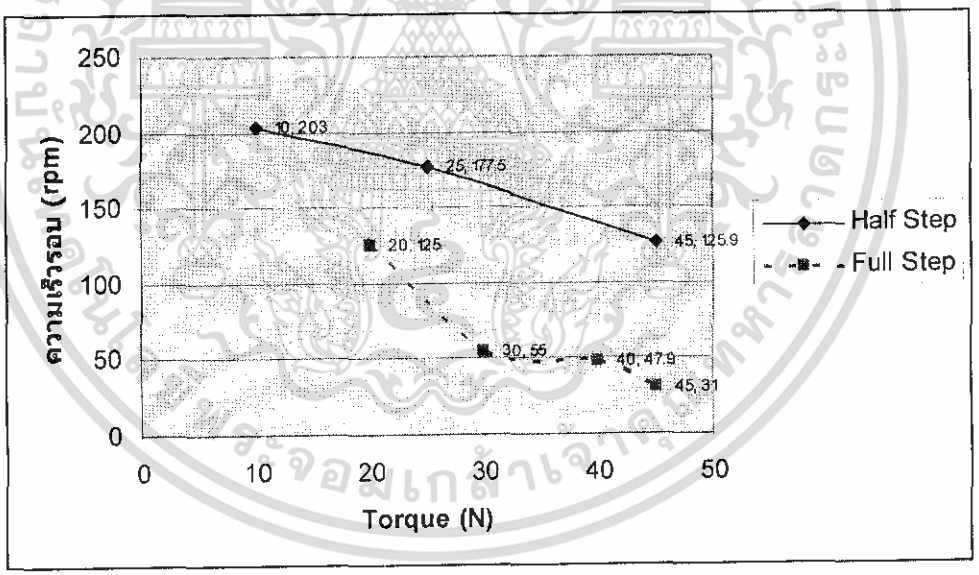
ภาพที่ 5.13 แสดงความสัมพันธ์ระหว่าง Frequency และ ความเร็วรอบ แบบ Full Step



ภาพที่ 5.14 แสดงความสัมพันธ์ระหว่าง Frequency , Torqueและ ความเร็วรอบ แบบ Half Step



ภาพที่ 5.15 แสดงความสัมพันธ์ระหว่าง Frequency , Torqueและ ความเร็วรอบ แบบ Full Step



ภาพที่ 5.16 แสดงความสัมพันธ์ระหว่าง Torque และ ความเร็วรอบ

5.3 สรุปผลการทดลอง

1. เมื่อทำการทดสอบตัว FPGA สามารถสร้างสัญญาณ STEP ที่จะนำไปใช้ในการสั่งงาน Stepping Motor ได้ตามรูปแบบการ Drive ทั้งแบบ Full Step และ Half Step ที่ถูกต้องได้

2. เมื่อทำการทดสอบทางด้านค่าความถี่ และเวลา FPGA สามารถสร้าง Step ได้ตรงความถี่และ ค่าเวลา ที่ตั้งได้อย่างถูกต้อง

3. สามารถทำการควบคุมแรงบิดและความเร็วรอบให้อยู่ในระดับที่ต้องการควบคุมได้ตามต้องการ ซึ่งความถี่ที่ใช้ในการทดลองนั้นจะเริ่มจากความถี่สูงสุดที่ Stepping Motor สามารถเกิดแรงบิดขับโหลดได้มาจนถึงระดับความถี่ที่ Stepping Motor สามารถเกิดแรงบิดขับโหลดได้สูงสุดถึงที่ 45 N แล้วทำการวัดค่าความเร็วรอบที่ความถี่ต่างๆที่ได้จากการวัดแรงบิด ทั้งในแบบ Full Step และแบบ Half Step จะได้ค่าตามตารางการทดลองและกราฟ

บทที่ 6
สรุปผลการวิจัยและแนวทางการพัฒนา

6.1 บทสรุป

ในการวิจัยครั้งนี้เป็นการศึกษาโครงสร้าง รูปแบบการทำงานของ FPGA และการเขียนโปรแกรมโดยใช้ภาษา VHDL เพื่อนำไปควบคุมการทำงานของ Stepping Motor จากการวิจัยสามารถใช้ FPGA Drive Stepping Motor เพื่อทำการควบคุมความเร็วรอบและแรงบิด ของการ Drive แบบ Full Step และแบบ Half Step

6.2 แนวทางการพัฒนา

แนวทางการพัฒนาต่อทางคณะผู้จัดทำเห็นว่า FPGA เป็นอุปกรณ์ที่มีความสามารถในการทำงานทางด้านเวลาดีกว่า Microcontroller ที่มีขายตามท้องตลาด แต่ยังมีข้อจำกัดด้านเทคนิคบางอย่างที่ตัว FPGA ไม่สามารถทำงานได้เหมือน Microcontroller ซึ่งยังต้องรอผู้ผลิตพัฒนาประสิทธิภาพของ FPGA ให้มีความสามารถมากยิ่งขึ้นและใช้งานได้ง่ายขึ้น

6.3 ปัญหาสำหรับการวิจัย

1. เนื่องจาก FPGA เป็นอุปกรณ์ชนิดใหม่จึงต้องใช้เวลาในการศึกษา FPGA และภาษา VHDL เป็นเวลานาน และแหล่งข้อมูลที่ใช้ในการศึกษายังมีไม่มากนัก

2. วงจรและอุปกรณ์ที่ให้ในการต่อวงจรขับ Stepping Motor แบบสองแหล่งจ่าย ยังไม่สมบูรณ์พอจึงต้องมีการพัฒนาให้มีประสิทธิภาพที่ดีกว่านี้

3. อุปกรณ์ที่ใช้ในการทดลองดังเช่น Load ที่ใช้วัดแรงบิดไม่ละเอียด จึงทำให้การทดลองได้ค่าที่ไม่ละเอียด

บรรณานุกรม

1. ภาษา VHDL สำหรับการออกแบบวงจรดิจิตอล,ชาญ ปัญญาใส, วัชราภร หนูทอง
2. เปิดโลก FPGA กับ WIZARD PLD-A01,Astron Logic Research & Development Co.,LTD.
3. การประยุกต์ใช้ FPGA ในการออกแบบอินเตอร์เฟสของฮาร์ดดิสคอนโทรเลอร์สำหรับเครื่องเอ็กซ์เรย์คอมพิวเตอร์,นายสีหชาติ สดับธรรมารักษ์,สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
4. การออกแบบแบะสร้างไมโครคอนโทรลเลอร์โดยใช้เอฟพีจีเอ, นายวัชราภร หนูทอง, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ภาคผนวก

ภาคผนวก ก
รายละเอียดโปรแกรมที่ใช้ควบคุม

โปรแกมที่ใช้ในการควบคุม

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

-- Uncomment the following lines to use the declarations that are
-- provided for instantiating Xilinx primitive components.
--library UNISIM;
--use UNISIM.VComponents.all;

entity TEST3 is
   Port ( sw : in std_logic;
          clk : in std_logic;
          reset : in std_logic;
          resetb : in std_logic;
                   input : in std_logic;
          output : out std_logic_vector(3 downto 0));
end TEST3;

architecture Behavioral of TEST3 is


type  state_type is (s0,s1,s2,s3,s4,s5,s6,s7);
signal state : state_type;

begin

state_p: PROCESS (clk,reset,resetb,input)
```

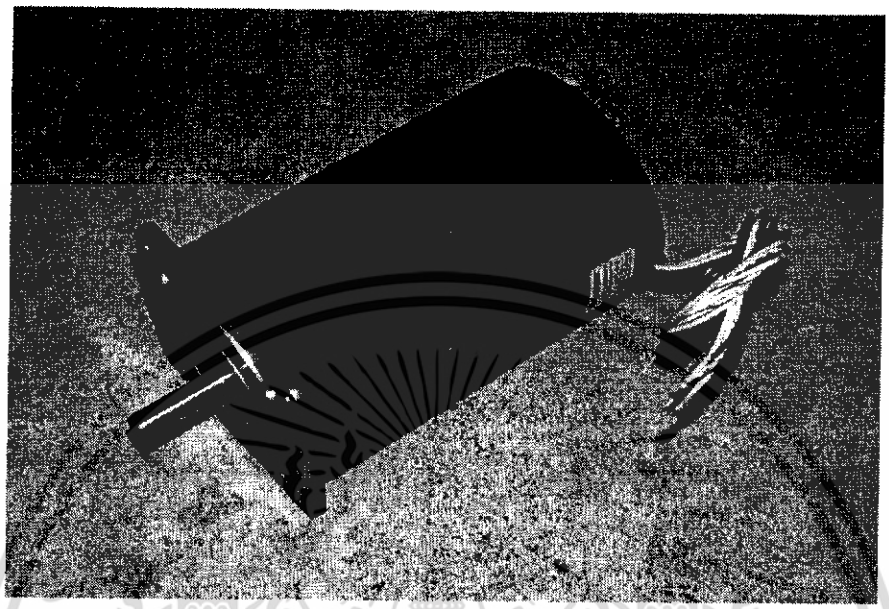
```
BEGIN

IF (not reset = '1')or(not resetb = '1') THEN
        state  <= s0;

ELSIF Rising_edge(clk)  THEN

        CASE state  IS

          WHEN  s0 =>
          IF (input = '1')  THEN
          state <= s1;
          END IF;

          WHEN s1 =>
          IF (input = '0')  THEN
          state <= s2;
          END IF;

          WHEN s2 =>
          IF (input = '1')  THEN
          state <= s3;
          END IF;

          WHEN s3 =>
          IF (input = '0')  THEN
          state <= s4;
          END IF;

          WHEN  s4 =>
          IF (input = '1')  THEN
```

```
            state <= s5;
            END IF;

            WHEN s5 =>
            IF (input = '0')  THEN
            state <= s6;
            END IF;

            WHEN s6 =>
            IF (input = '1')  THEN
            state <= s7;
            END IF;

            WHEN s7 =>
            IF (input = '0')  THEN
            state <= s0;
            END IF;

        END CASE;
  END IF;
END PROCESS  state_P;

output_P: PROCESS (sw,state)
        BEGIN

        IF (sw = '1') THEN
        CASE state IS
                WHEN  s0 =>  output(3 downto 0)  <= "0001";
                WHEN  s1 =>  output(3 downto 0)  <= "0010";
                WHEN  s2 =>  output(3 downto 0)  <= "0100";
                WHEN  s3 =>  output(3 downto 0)  <= "1000";
```

```
            WHEN  s4 =>  output(3 downto 0)  <= "0001";
            WHEN  s5 =>  output(3 downto 0) <= "0010";
            WHEN  s6 =>  output(3 downto 0)  <= "0100";
            WHEN  s7 =>  output(3 downto 0)  <= "1000";
        END CASE;
       ELSIF (sw = '0') THEN
        CASE state IS
            WHEN  s0 =>  output(3 downto 0)  <= "0001";
            WHEN  s1 =>  output(3 downto 0)  <= "0011";
            WHEN  s2 =>  output(3 downto 0)  <= "0010";
            WHEN  s3 =>  output(3 downto 0)  <= "0110";
            WHEN  s4 =>  output(3 downto 0)  <= "0100";
            WHEN  s5 =>  output(3 downto 0)  <= "1100";
            WHEN  s6 =>  output(3 downto 0)  <= "1000";
            WHEN  s7 =>  output(3 downto 0)  <= "1001";
        END CASE;
    END IF;
END PROCESS output_P;

end Behavioral;
```

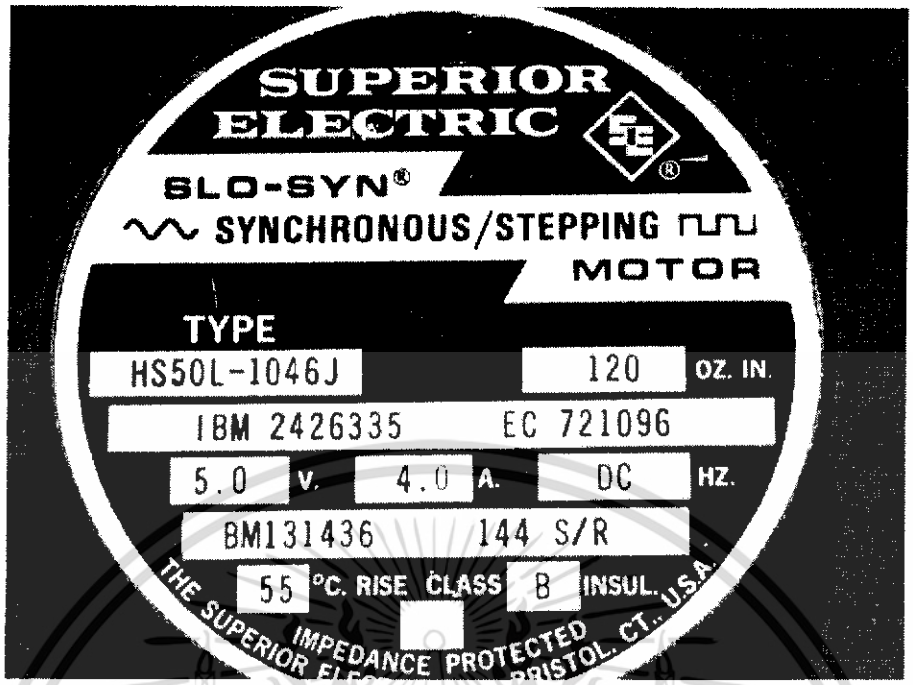
ภาคผนวก ข

ลักษณะของมอเตอร์ที่ใช้ในการวิจัย



ภาพที่ ข1. มอเตอร์ที่ใช้ในการวิจัย

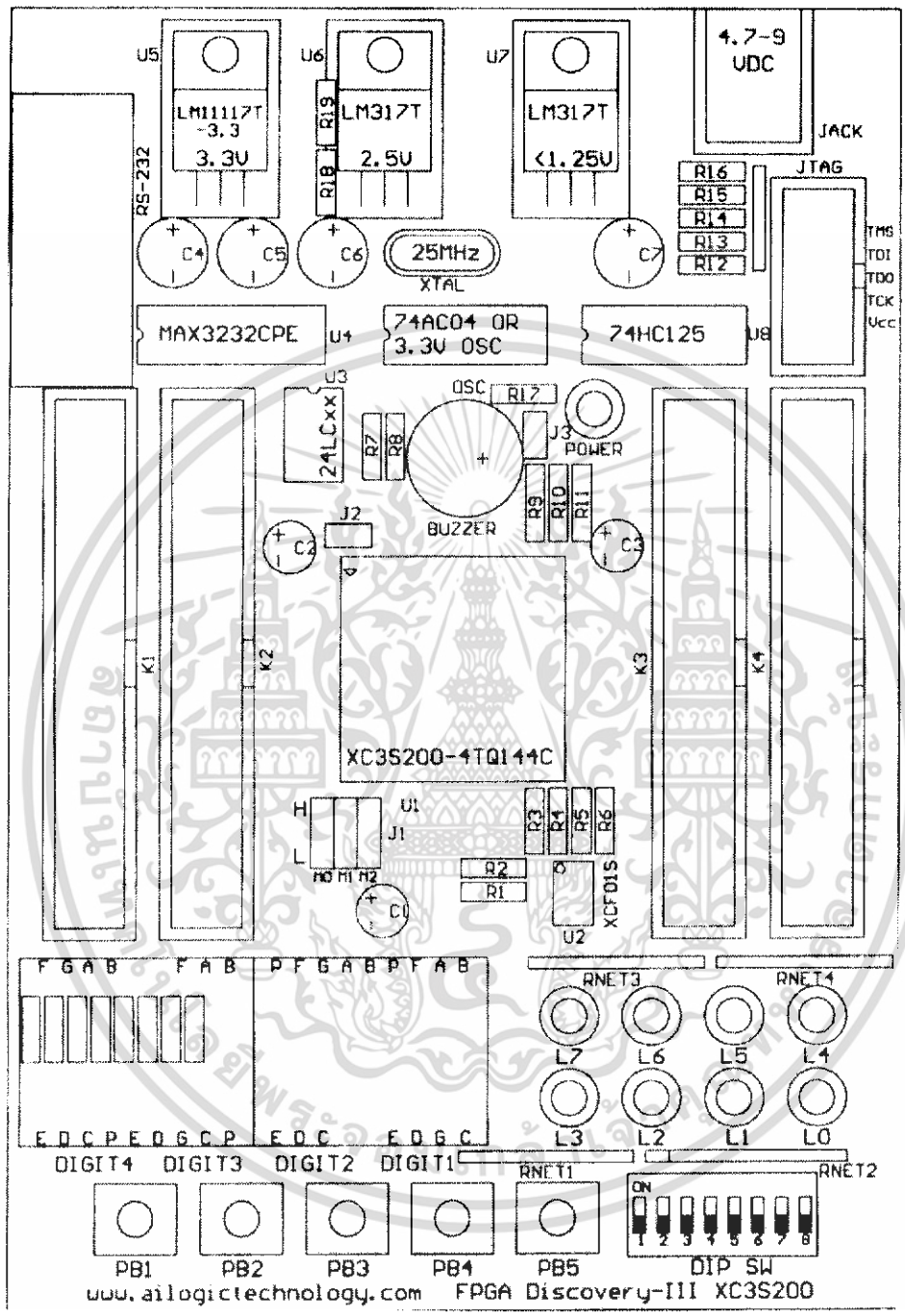
ภาพที่ ข2. มอเตอร์ที่ใช้ในการวิจัย



ภาพที่ ข3. มอเตอร์ที่ใช้ในการวิจัย

ภาคผนวก ค

FPGA Discovery-III XC3S200 Board Manual



1. **Connector and Jumper**

1.1 Expansion connector (K1 – K4)

เป็นหัวต่อที่ใช้เชื่อมสัญญาณ I/O จาก FPGA ไปยังบอร์ดหรืออุปกรณ์ภายนอกที่มี I/O เป็น 3.3V โดยจะต่ออยู่กับขา CPLD ดังตารางด้านท้าย ในกรณีที่ I/O ของ FPGA เป็น Output สามารถต่อออกจากบอร์ดไปขับ Input

ของอุปกรณ์ที่เป็นระบบ 3.3V และ 5V ได้โดยตรง แต่ถ้า I/O ของ FPGA เป็น Input นั้นจะรับได้เฉพาะ Input ที่เป็นระบบ 3.3V เท่านั้น (ถ้ารับมาจากระบบ 2.5V ต้องใช้ตัวความต้านทาน (R) มา Pull up) แต่ถ้า Input เป็นระบบ 5V จะต้องใช้บัฟเฟอร์ที่เป็นระบบ 3.3V มากั้นเพื่อป้องกันไม่ให้ I/O ของ FPGA ได้รับความเสีย บัฟเฟอร์ที่เป็นระบบ 3.3V อาจใช้ไอซีตระกูล 74HCxx หรือ 74ACxx (เร็วกว่า) มากินไฟเลี้ยง (Vcc) **3.3VDC** และต้องต่อความต้านทาน 200 โอห์มที่ Input ของบัฟเฟอร์ทุกตัว (เพื่อจำกัดกระแสไม่ให้ Input ของบัฟเฟอร์เสียหาย คือ < 10 mA)

1.2 JTAG connector

เป็นหัวต่อที่ใช้ต่อกับสายที่ใช้โปรแกรมข้อมูลลงตัว FPGA และ PROM โดยผ่ายทาง JTAG Cable

1.3 J1

จัมเปอร์ J1 ประกอบด้วย M0 , M1 , M2 โดยปกติให้เซทไว้ที่ลอจิก " L " หรืออยู่ในโหมด Master serial เนื่องจากเราสามารถโปรแกรม PROM หรือ FPGA โดยใช้สาย JTAG ได้อยู่แล้วโดยไม่ต้องสนใจตำแหน่งของจัมเปอร์แต่อย่างใด

1.4 J2

จัมเปอร์ที่ใช้ควบคุมให้ FPGA ทำการ Pull up I/O ของ FPGA ทุกขาเมื่อใส่จัมเปอร์ และจะเป็น Hi Impedance เมื่อถอดจัมเปอร์ออก

1.5 J3

จัมเปอร์ที่ใช้ตัด BUZZER ออกจาก FPGA เมื่อถอดจัมเปอร์ออก

2. Input

2.1 DIP switch (DIP SW)

เป็นชุดของสวิทซ์เลื่อนขนาดเล็กที่ใช้ป้อนข้อมูลเข้าสู่ FPGA โดยถ้าเลื่อนลง (**Off**) จะเป็น "1"

 ถ้าเลื่อนขึ้น (**On**) จะเป็น "0"  โดยเชื่อมต่อกับขาของ FPGA ดังตารางด้านท้าย DIP SW ทุกตัวจึงทำงานแบบ Active Low

2.2 Push button switch (PB1 – PB5)

เป็นสวิทซ์กดติดปล่อยดับที่ให้สัญญาณเอ้าท์พุทเป็นระดับลอจิก "0" เมื่อกดสวิทซ์ และเป็นระดับลอจิก "1" เมื่อปล่อยสวิทซ์ โดยจะต่ออยู่กับขา FPGA ดังตารางด้านท้าย Push button switch ทุกตัวจึงทำงานแบบ Active Low

2.3 Changeable oscillator (OSC)

เป็นตัวกำเนิดสัญญาณนาฬิกาที่สามารถเปลี่ยนค่าความถี่ที่ต้องการได้ โดยการถอดเปลี่ยนออสซิลเลเตอร์เดิม (3.3V) ที่ให้มาบนบอร์ดออก แล้วใส่ตัวใหม่(3.3V) เข้าไปแทนที่ที่ซอคเก็ตไอซีเบอร์ 74AC04 โดยที่เอาท์พุทของ OSC จะต่ออยู่กับขา FPGA ดังตารางด้านท้าย ซึ่งเป็นขา Global clock เหมาะสำหรับวงจรที่ต้องการความถี่ในการทำงานสูงๆ

3. Output

3.1 7-Segment (DIGIT1 – DIGIT4)

เป็นตัวแสดงผลเจ็ดส่วนจำนวน 4 หลักที่สามารถถอดออกได้ (หากต้องการใช้ I/O ที่ Connector K1 และ K2 ส่วนที่แชร์ I/O อยู่กับตัวแสดงผลเจ็ดส่วนทั้ง 4 หลัก) โดยเรียงจากซ้ายไปขวาคือ DIGIT4, DIGIT3, DIGIT2 และ DIGIT1 โดยตัวที่ 2 และ 1 จะทำการกลับตัวแสดงผลเจ็ดส่วนให้เพื่อใช้จุด (Dot) ในการทำนาฬิกาหรือ แสดงองศาในการวัดอุณหภูมิ เช่น 11:39 หรือ 20°

ตัวแสดงผลเจ็ดส่วนทั้งหมดจะต่อขาคาต้าเข้าด้วยกันโดยมีขาไฟร่วม (Common cathode) แยกกันสี่ขา ดังนั้นผู้ใช้จึงจำเป็นต้องใช้เทคนิคในการสแกน (Scan) เพื่อให้ตัวแสดงผลทั้งเจ็ดส่วนสามารถแสดงผลพร้อมกันได้ทั้งหมด และตัวแสดงผลทั้งหมดเป็นแบบไฟร่วม (Common cathode) โดยจะต่ออยู่กับขา FPGA ดังตารางด้านท้าย

3.2 LED แสดงผล

LED แสดงผล L0 – L7 จะต่อแชร์กับ I/O ของ Connecter K3 และ K4 โดยที่ L2 , L3 , L6 และ L7จะต่อแชร์กับ I/O ของ Connecter K3 โดยมี Resister " RNET3" 8P4R 470 Ohm จำกัดกระแส และ L0 , L1 , L4 และ L5 จะต่อแชร์กับ I/O ของ Connecter K4 โดยมี Resister " RNET3" 8P4R 470 Ohm จำกัดกระแส

3.3 Buzzer

เป็นออดความถี่เสียง (Buzzer) โดยที่จะมีเสียงดังเมื่อป้อนสัญญาณเป็น High "1" โดยจะต่ออยู่กับขา FPGA ดังตารางด้านท้าย กรณีที่ต้องการใช้ I/O ของ Connecter K4 ที่แชร์อยู่กับออดหรือไม่ต้องการใช้ออดให้ถอดจัมเปอร์ J3 ออก

4. Misc

4.1 Jack สำหรับ DC Adaptor

เป็นหัวต่อไฟเลี้ยงเพื่อป้อนให้แก่บอร์ดในการทำงาน ต่ออยู่กับอแดปเตอร์ที่มีไฟออกมาเป็น 4.7V – 9V โดยมีขั้วด้านในเป็น บวก "+" ด้านนอกเป็น ลบ "-"

4.2 Power LED (POWER)

เป็นไดโอดเปล่งแสงว่าในขณะนั้นๆ มีไฟเลี้ยงบอร์ดอยู่หรือไม่

4.3 RS-232C Port

เป็นพอร์ต RS-232C ซึ่งหากไม่ต้องการใช้พอร์ต RS-232C แต่ต้องการใช้เป็น I/O ที่ Connector K1 และ K2 (ส่วนที่แชร์ I/O อยู่กับพอร์ต RS-232C) ให้ถอดไอซี MAX3232CPE ออกจาก Socket

หมายเหตุ ขา 15 ของไอซี MAX3232CPE ต้องบัดกรีลงกราวนด์ด้วย

4.4 Platform Flash PROM เบอร์ XCF01S

เป็น Serial PROM ที่สามารถโปรแกรมได้โดยตรงผ่านทางสาย JTAG สามารถโปรแกรมซ้ำได้ประมาณ 20,000 ครั้ง

4.5 I2C Socket

เป็น Socket สำหรับใส่ไอซีแบบ I2C Serial EEPROM เบอร์ 24LCxx เช่น 24LC256 เป็นต้น ซึ่งสามารถถอดออกได้ (หากต้องการใช้ I/O ที่ Connector K1 และ K2 ส่วนที่แชร์ I/O อยู่กับ I2C Serial EEPROM) I/O 2 ขานี้จะมี Pull up resister 4.7kOhm ต่ออยู่

5. ตาราง I/O ของ FPGA

ดูจากไฟล์ K1 , K2 , K3 , K4 และ 7-Segment การดูขาของ Connector K1-K4 ให้สังเกตด้านรอยบากลึกรูปสี่เหลี่ยมจะเป็นแถวของขาที่เป็นเลขคี่ทั้งหมดเริ่มจาก 1 ถึง 39 โดยที่ด้านซ้ายสุดเป็นขา 1 ซึ่งจะมี Mark รูปสามเหลี่ยมปรากฏอยู่ด้านข้าง

หมายเหตุ ข้อมูล UPDATE ต่างๆ สอบถามหรือดูรายละเอียดได้ที่บริษัท เอเพก อินสตรูเมนต์ จำกัด 77/9 ซอยลาดพร้าว1 แขวงลาดยาว เขตจตุจักร กทม 10900 โทร. 0 – 293 – 2084 หรือ 0 – 1833 – 7279 หรือดาวน์โหลดได้ที่ http://www.ailogictechnology.com

7-Segment	FPGA Pinout	Descriptions
a	p40	a
b	p35	b
c	p32	c
d	p30	d
e	p27	e
f	p25	f
g	p23	g
dp	p20	Decimal Point
DG1	p31	DIGIT1 , COMMON CATHODE
DG2	p33	DIGIT2 , COMMON CATHODE
DG3	p36	DIGIT3 , COMMON CATHODE
DG4	p41	DIGIT4 , COMMON CATHODE

Push Botton	FPGA Pinout	Descriptions
PB1	p44	Push Botton No. 1
PB2	p46	Push Botton No. 2
PB3	p47	Push Botton No. 3
PB4	p50	Push Botton No. 4
PB5	p51	Push Botton No. 5

EEPROM	FPGA Pinout	Descriptions
I2C-SCL	p128	24LCXX
I2C-SDA	p129	24LCXX

RS-232	FPGA Pinout	Descriptions
TX	p131	MAX3232CPE
RX	p132	MAX3232CPE

LED	FPGA Pinout	Descriptions
L0	p70	L0
L1	p77	L1
L2	p69	L2
L3	p76	L3
L4	p74	L4
L5	p79	L5
L6	p73	L6
L7	p78	L7

Dip SW	FPGA Pinout	Description
1	p52	Dip Switch No.1
2	p53	Dip Switch No.2
3	p55	Dip Switch No.3
4	p56	Dip Switch No.4
5	p59	Dip Switch No.5
6	p60	Dip Switch No.6
7	p63	Dip Switch No.7
8	p68	Dip Switch No.8

Oscillator	FPGA Pinout	Descriptions
OSC	p127	25MHz , GCLK6

BUZZER	FPGA Pinout	Descriptions
BUZZER	p124	BUZZER

K1 Pinout	FPGA Pinout	Descriptions
1	p40	I/O , a
2		3.3 V
3	p35	I/O , b
4		GND
5	p32	I/O , c
6		GND
7	p30	I/O , d
8		GND
9	p27	I/O , e
10		GND
11	p25	I/O , f
12		GND
13	p23	I/O , g
14		GND
15	p20	I/O , dp
16		GND
17	p17	I/O
18		GND
19	p14	I/O
20		GND

k1 Pinout	FPGA Pinout	Descriptions
21	p12	I/O
22		GND
23	p10	I/O
24		GND
25	p7	I/O
26		GND
27	p5	I/O
28		GND
29	p2	I/O
30		GND
31	p141	I/O
32		GND
33	p137	I/O
34		GND
35	p132	I/O, RS-232 (RX)
36		GND
37	p130	I/O
38		GND
39	p128	I/O , GCLK7 , I2C-SCL
40		GND

K2 Pinout	FPGA Pinout	Descriptions
1	p41	I/O , DIGIT4
2		3.3 V
3	p36	I/O , DIGIT3
4		GND
5	p33	I/O , DIGIT2
6		GND
7	p31	I/O , DIGIT1
8		GND
9	p28	I/O
10		GND
11	p26	I/O
12		GND
13	p24	I/O
14		GND
15	p21	I/O
16		GND
17	p18	I/O
18		GND
19	p15	I/O
20		GND

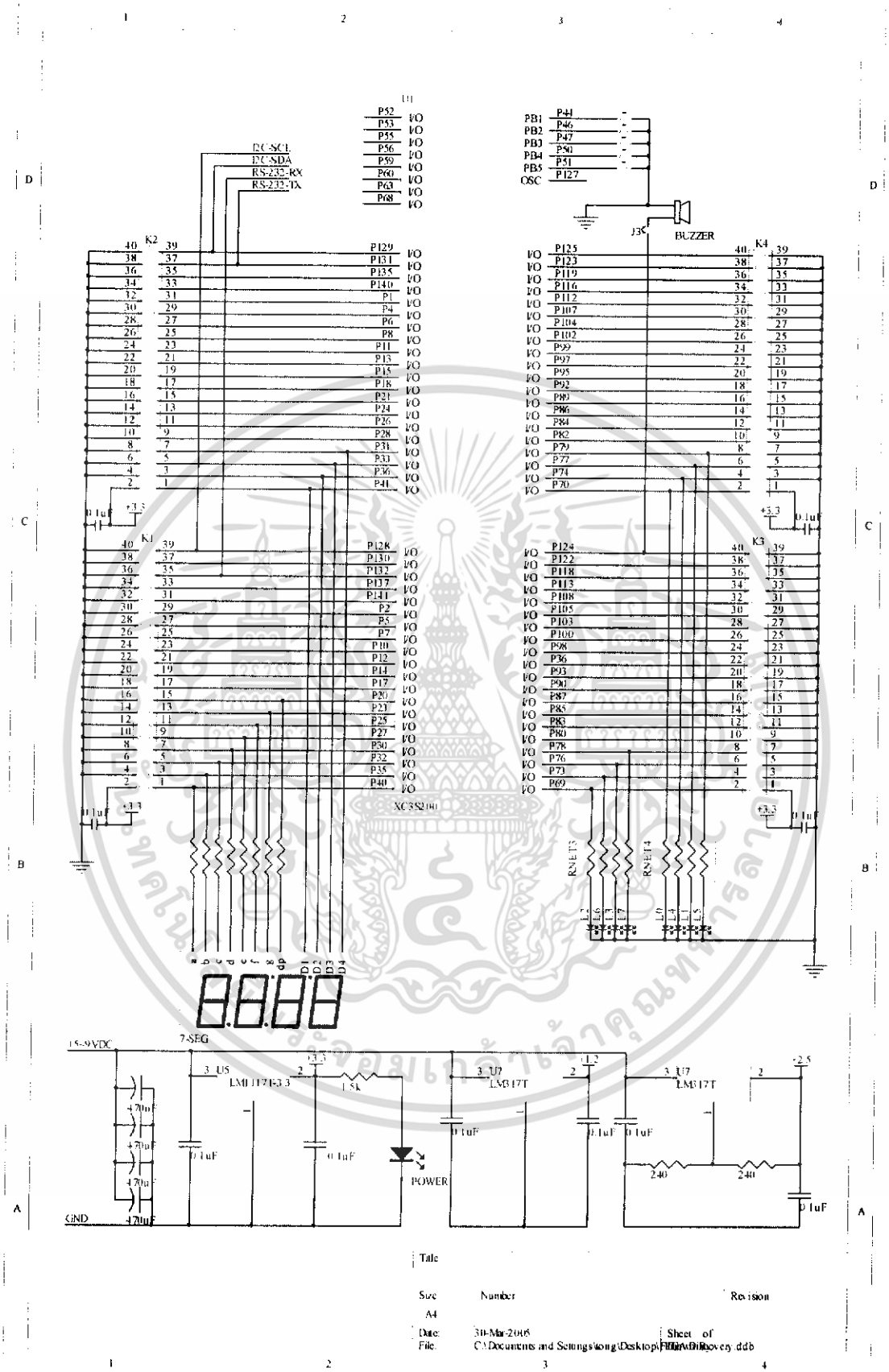
k2 Pinout	FPGA Pinout	Descriptions
21	p13	I/O
22		GND
23	p11	I/O
24		GND
25	p8	I/O
26		GND
27	p6	I/O
28		GND
29	p4	I/O
30		GND
31	p1	I/O
32		GND
33	p140	I/O
34		GND
35	p135	I/O
36		GND
37	p131	I/O , RS-232 (TX)
38		GND
39	p129	I/O , I2C-SDA
40		GND

K3 Pinout	FPGA Pinout	Descriptions
1		3.3 V
2	p69	I/O , L2
3		GND
4	p73	I/O , L6
5		GND
6	p76	I/O , L3
7		GND
8	p78	I/O , L7
9		GND
10	p80	I/O
11		GND
12	p83	I/O
13		GND
14	p85	I/O
15		GND
16	p87	I/O
17		GND
18	p90	I/O
19		GND
20	p93	I/O

k3 Pinout	FPGA Pinout	Descriptions
21		GND
22	p96	I/O
23		GND
24	p98	I/O
25		GND
26	p100	I/O
27		GND
28	p103	I/O
29		GND
30	p105	I/O
31		GND
32	p108	I/O
33		GND
34	p113	I/O
35		GND
36	p118	I/O
37		GND
38	p122	I/O
39		GND
40	p124	I/O , GCLK4 , BUZZER

K4 Pinout	FPGA Pinout	Descriptions
1		3.3 V
2	p70	I/O , L0
3		GND
4	p74	I/O , L4
5		GND
6	p77	I/O , L1
7		GND
8	p79	I/O , L5
9		GND
10	p82	I/O
11		GND
12	p84	I/O
13		GND
14	p86	I/O
15		GND
16	p89	I/O
17		GND
18	p92	I/O
19		GND
20	p95	I/O

k4 Pinout	FPGA Pinout	Descriptions
21		GND
22	p97	I/O
23		GND
24	p99	I/O
25		GND
26	p102	I/O
27		GND
28	p104	I/O
29		GND
30	p107	I/O
31		GND
32	p112	I/O
33		GND
34	p116	I/O
35		GND
36	p119	I/O
37		GND
38	p123	I/O
39		GND
40	p125	I/O , GCLK5



หมายเหตุ ขา 36 ของ FPGA ที่ต่อกับ ขา 22 ของ Connector K3 แก้เป็นขา 96