สำนักหอสมุดกลาง_ พระจอมเกล้าลาดกระบัง

การสร้าง G-CODE PROCESSOR ด้วย อัลกอริทึม CORDIC

G-CODE PROCESSOR IMPLEMENTATION USING CORDIC ALGORITHM

อมร ช่วยชู

AMORN CHUAICHU

วท.
อ่ยน ค
2545

เลขหมู่.....
เลขทะเบียน 44053
วัน, เดือน, ปี 25 ต.ค. 2545

b 11255985
i 12240894

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2545

ISBN 974-324-037-3

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

G-CODE PROCESSOR IMPLEMENTATION USING CORDIC ALGORITHM

AMORN CHUAICHU

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING

SCHOOL OF GRADUATE STUDIES

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2002

ISBN 974-324-037-3

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

COPYRIGHT 2002

SCHOOL OF GRADUATE STUDIES

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หัวข้อวิทยานิพนธ์	การสร้าง G-Code Processor ด้วยอัลกอริทึม CORDIC
นักศึกษา	นายอมร ช่วยชู
รหัสประจำตัว	41061135
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมศาสตรไฟฟ้า
พ.ศ.	2545
อาจารย์ผู้ควบคุมวิทยานิพนธ์	ผศ. บรรจง ปิยธำรง

บทคัดย่อ

ในปัจจุบันได้ใช้ระบบเครื่องจักรซีเอ็นซีในงานอุตสาหกรรมอย่างกว้างขวางซึ่งเครื่องจักรซีเอ็นซีเหล่านี้ต้องสามารถรับคำสั่งควบคุมการทำงานของเครื่องจักรที่เรียกว่า จี-โค้ดได้ ในการพัฒนาระบบซีเอ็นซีโดยทั่วไปได้ใช้โปรเซสเซอร์ เช่น ไมโครคอนโทรเลอร์ ไมโครโปรเซสเซอร์ หรือ ไมโครคอมพิวเตอร์ ในการควบคุมการทำงานเครื่องจักร แต่ชุดคำสั่ง(Instruction Set) ของโปรเซสเซอร์เหล่านี้ไม่เหมาะสมกับคำสั่งจี-โค้ด ทำให้การพัฒนาโปรแกรมเป็นไปได้ยากและซับซ้อน

ในงานวิจัยนี้เป็นการออกแบบโปรเซสเซอร์ที่รับชุดคำสั่งเป็นจี-โค้ด หรือเรียกว่า "จี-โค้ดโปรเซสเซอร์" ซึ่งแบ่งส่วนประกอบเป็น 4 ส่วน คือส่วนของการคำนวณโดยใช้ CORDIC อัลกอริทึม ส่วนควบคุมใช้หลักการไมโครโปรแกรม ส่วนการเก็บข้อมูล และ ส่วนเชื่อมต่อสัญญาณ ซึ่งการวิจัยนี้ได้แสดงให้เห็นถึงโครงสร้างการทำงาน, อัลกอริทึม, วงจร และ ผลการทำงาน ในส่วนการคำนวณของจี-โค้ดโปรเซสเซอร์เป็นหลัก

คำสำคัญ : สถาปัตยกรรมคอมพิวเตอร์, เครื่องจักรซีเอ็นซี, คอร์ดิคอัลกอริทึม, จี-โค้ด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Thesis Title	G-Code Processor Implementation Using CORDIC Algorithm
Student	Mr. Amorn Chuaichu
Student ID	41061135
Degree	Master of Engineering
Programme	Electriccal Engineering
Year	2002
Thesis Advisor	Assoc.Prof.Bunjong Piyathumrong

Abstract

Nowadays, CNC Mechanism, which can receive command to control in the format "G-Code" in order the process of machine, is widely used in Industry. This system is normally developed by using processors, such as micro-controllers, microprocessors or microcomputer, but the instruction set of such processors are not suitable with G-Code commands; This leads to a difficulty and complication when we want to develop a G-Code program on this processors.

This thesis concerns a design of processor which can receive command in the form of G-Code, so we called this processor a "G-Code Processor" and it consists of 4 components. They are Arithmetic Unit using CORDIC algorithm, Control Unit using Micro-programs, data paths and registers, and an interface unit respectively. This research investigates an architecture algorithm, circuit, and Arithmetic Unit of the G-Code processor.

Keyword : Computer Architecture, CNC Machine, CORDIC Algorithm, G-Code

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงได้อย่างดี ด้วยคำแนะนำและคำปรึกษาเกี่ยวกับแนวทางการทำวิทยานิพนธ์ในทุกด้านจาก ผศ.บรรจง ปิยธำรง ซึ่งเป็นอาจารย์ผู้ควบคุมวิทยานิพนธ์ ผู้วิจัยรู้สึกซาบซึ้งในความอนุเคราะห์จากท่านและขอกราบขอบพระคุณเป็นอย่างสูง

อมร ช่วยชู

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

หน้า

บทคัดย่อภาษาไทย I

บทคัดย่อภาษาอังกฤษ II

กิตติกรรมประกาศ III

สารบัญ IV

สารบัญตาราง VII

สารบัญรูป VIII

บทที่ 1. บทนำ 1

1.1 ความเป็นมา 1

1.2 วัตถุประสงค์ในการศึกษา 1

1.3 ทฤษฎีและแนวคิดในงานวิจัย 2

1.4 ขอบเขตการวิจัย 2

1.5 ประโยชน์ที่ได้จากงานวิจัยนี้ 2

บทที่ 2 งานวิจัยที่เกี่ยวข้องและหลักการของ จี-โค้ด โปรเซสเซอร์ 4

2.1 บทนำ 4

2.2 งานวิจัยที่เกี่ยวข้อง 4

2.3 โครงสร้างของการทำงาน จี-โค้ดโปรเซสเซอร์ 10

2.4 ส่วนการทำงานต่างๆของ จี-โค้ด โปรเซสเซอร์ 11

2.4.1 การทำงานของส่วนประมวลผล 11

2.4.2 การทำงานของส่วนควบคุมภายใน 11

2.4.3 การทำงานของส่วนเชื่อมต่ออุปกรณ์ภายนอก 13

2.4.4 การทำงานหน่วยเก็บข้อมูลและเส้นทางข้อมูล 15

2.5 สรุปและแนวทางการวิจัย 16

บทที่ 3 การทำงานของ คอร์ดิคอัลกอริทึม 17

3.1 บทนำ 17

3.2 หลักการของ CORDIC (COordinate Rotation Digital Computing) 17

3.3 โครงสร้างการทำงานของ Block Diagram พื้นฐานของคอร์ดิค 20

3.4 การทำงานของ วงจรคอร์ดิค 21

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

หน้า

3.5 การปรับปรุงการคำนวณค่าคอร์ดิค 21
3.5.1 การทำงานของวงจร PreCORDIC 21
3.5.2 การทำงานของวงจร PostCORDIC 22
3.6 สรุป 23

บทที่ 4 การคำนวณค่าของหน่วยประมวลผลใน จี-โค๊ด โปรเซสเซอร์ 24
4.1 บทนำ 24
4.2 สมการหน่วยประมวลผลของ จี-โค๊ด โปรเซสเซอร์ 24
4.2.1 การคำนวณทางคณิตศาสตร์ 24
4.2.2 การคำนวณด้วย คอร์ดิคอัลกอริทึม 25
4.3 การสร้างวงจรจากสมการหน่วยประมวลผล 26
4.4 โครงสร้างหน่วยประมวลผลที่ใช้หลักการ คอร์ดิคอัลกอริทึม 28
4.5 สรุป 29

บทที่ 5 อุปกรณ์และการทดลอง 30
5.1 บทนำ 30
5.2 การทดสอบความถูกต้องของการคำนวณทางคณิตศาสตร์ 30
5.2.1 การจำลองทางคณิตศาสตร์ตามหลักการ คอร์ดิคอัลกอริทึม 30
5.2.2 การจำลองทางคณิตศาสตร์หน่วยประมวลผล จีโค๊ด โปรเซสเซอร์ 36
5.2.3 การจำลองทางคณิตศาสตร์ ตามรูปแบบการใช้งานจริง 38
5.3 การทดสอบทางด้านความเร็วในการทำงาน 39
4.4 สรุป 41

บทที่ 6 สรุปผลการวิจัยและข้อเสนอแนะ 42
6.1 สรุปผลการวิจัย 42
6.2 แนวทางในการพัฒนา 43

บรรณานุกรม 45

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

หน้า

ภาคผนวก ก. โปรแกรมภาษา C ที่ใช้กับตัวประมวลผล MCH 51 เพื่อคำนวณ
คำสั่ง G00 และ G01 47

ประวัติผู้เขียน 49

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

สารบัญตาราง

ตารางที่ หน้า

2.1 เปรียบเทียบคำสั่ง จี-โค้ด กับชุดคำสั่งของ จี-โค้ด โปรเซสเซอร์ 10

2.2 ชุดคำสั่งไมโครโปรแกรมของ Control Unit 12

3.1 การทำงานรูปแบบต่างๆของคอร์ดิค 18

3.2 การทำงานรูปแบบต่างๆของคอร์ดิค หลังการปรับปรุงค่าแล้ว 19

3.3 ค่าคงที่ที่ใช้ในสมการของคอร์ดิค 19

4.1 การแทนส่วนต่างๆของหลักการแก้สมการหน่วยประมวลผลด้วยวงจรดิจิตอล 27

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป

หน้า

รูปที่

2.1 โครงสร้างระบบควบคุมของเครื่องจักรซีเอ็นซี ที่ใช้หลักการ FMS 4
2.2 โครงสร้างระบบควบคุมของเครื่องจักรซีเอ็นซี ที่ใช้หลักการ HFMS 5
2.3 การทำงานของเครื่องจักรซีเอ็นซี ที่ใช้ระบบสื่อสาร Fieldbus 5
2.4 โครงสร้างระบบควบคุม ที่ใช้ระบบไมโครคอมพิวเตอร์ กับตัวประมวลผลเชิงเลข 6
2.5 โครงสร้างระบบควบคุมของเครื่องจักรซีเอ็นซี ที่พัฒนาในส่วนโปรแกรม 6
2.6 โครงสร้างระบบควบคุมของเครื่องจักรซีเอ็นซี ที่เป็นงานวิจัยในประเทศ 7
2.7 การทำงานของระบบควบคุมการเคลื่อนที่โดยใช้วงจรรวม HCTL-1000 7
2.8 การทำงานของระบบควบคุมการเคลื่อนที่โดยใช้การ์ดควบคุมสำเร็จรูป PMD5650 8
2.9 การทำงานของระบบควบคุมการเคลื่อนที่โดยใช้วงจรรวม HCTL-1100 8
2.10 การทำงานของระบบเครื่องจักรซีเอ็นซี โดยใช้ จี-โค้ด โปรเซสเซอร์ 9
2.11 โครงสร้างสถาปัตยกรรม จี-โค้ด โปรเซสเซอร์ 10
2.12 โครงสร้างของส่วนควบคุมภายใน 12
2.13 วงจรการทำงานในส่วนเชื่อมต่อกับอุปกรณ์ภายนอก 14
2.14 สัญญาณในการอ่านค่าจากอุปกรณ์ภายนอก และเป็นสัญญาณในการเขียนอุปกรณ์ภายนอก 14
2.15 โครงสร้างของเส้นทางข้อมูลและส่วนเก็บข้อมูล 15
3.1 Block Diagram CORDIC 20
3.2 วงจรคอร์ดิค 21
3.3 วงจร PreCORDIC 22
3.4 วงจร PostCORDIC 23
4.1 ค่าต่างๆตามสมการที่ 4.3 เพื่อทำการประมาณเส้นโค้งด้วยเส้นตรง 25
4.2 หลักการแก้สมการที่ 4.2 ด้วย COEDIC อัลกอริทึม 25
4.3 การแก้สมการที่ 4.3 ด้วย CORDIC อัลกอริทึม 26
4.4 วงจรดิจิตอลตามหลักการคำนวณการเคลื่อนที่เป็นเส้นตรง 27
4.5 วงจรดิจิตอลตามหลักการคำนวณการเคลื่อนที่เป็นเส้นโค้งส่วนของวงกลม 28
4.6 วงจรหน่วยประมวลผลของ จีโค้ด โปรเซสเซอร์ 29
5.1 การทดสอบการทำงานของ Pre-CORDIC 31
5.2 การทดสอบการทำงานของ Post-CORDIC 32

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป (ต่อ)

หน้า

รูปที่

5.3 แสดงเอาท์พุตและค่าที่เกิดจากการคำนวณผิดพลาดของ อัลกอริทึม CORDIC-R 34
5.4 แสดงเอาท์พุตและค่าที่เกิดจากการคำนวณผิดพลาดของ อัลกอริทึม CORDIC-V 35
5.5 แสดงค่าเอาท์พุตของการจำลองการทำงาน G00G01 37
5.6 แสดงค่าความผิดพลาดของการจำลองการทำงานคิดเป็นร้อยละ 37
5.7 แสดงค่าคู่อันดับ (Px,Py) และค่าผิดพลาด ที่ได้จากการทดลองโปรแกรม G02 38
5.8 แสดงเส้นทางการเคลื่อนที่ของหัวเครื่องจักรตามแนวระนาบ x,y 39
5.9 ผลการวัดการประมวลผลจากระบบจริงที่ใช้ MCS-51 ความถี่ระบบ 8 MHz 40
5.10 ผลการจำลองวงจรประมวลผล ด้วยโปรแกรม Foundation 1.5 40

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

บทที่ 1
บทนำ

1.1 ความเป็นมา

ในปัจจุบันในประเทศไทยมีการใช้เครื่องจักรซีเอ็นซี (Computer Numerical Control Machine : CNC Machine) มาใช้ในงานอุสาหกรรมอย่างมากส่งผลให้เกิดมีการทำงานวิจัยทางด้านนี้กันอย่างกว้างขวาง เช่น งานวิจัยที่ใช้ไมโครคอนโทรเลอร์ [1] หรือใช้การประมวลผลสัญญาณดิจิตอล (Digital Signal Processor : DSP) [2,3] มาควบคุมการเคลื่อนที่ (Motion Control) ของเครื่องจักรซีเอ็นซี นอกจากนี้ยังมีงานวิจัยที่ใช้บอร์ดควบคุมการเคลื่อนที่สำเร็จรูป (Motion Control Board) มาใช้ในการควบคุมการเคลื่อนที่ของเซอร์โวมอเตอร์แบบไฟฟ้ากระแสตรง (DC) และกระแสสลับ (AC) โดยใช้ชิพควบคุมการเคลื่อนที่ (Motion Control Chip) HCTL-1000 [4] หรือบอร์ด PMD-5650 และใช้ชิพ PMD [5] จากงานวิจัยที่กล่าวมาทั้งหมดนี้ นอกจากการทำการควบคุมการเคลื่อนที่แล้วเพื่อที่จะพัฒนาเครื่องจักรซีเอ็นซียังจำเป็นต้องมีงานที่เกี่ยวกับคำสั่งจี-โค้ด(G-Code) เช่น G00, G01, G02 และ G03 เป็นคำสั่งในการเคลื่อนที่เป็นเส้นตรง (Linear Interpolation) และเส้นโค้งส่วนของวงกลม (Circular Interpolation) ในการประมวลผลงานส่วนนี้งานวิจัยที่กล่าวถึงข้างต้น [2][4-6] นั้นได้แยกการทำงานออกจากชิพที่ใช้ในการควบคุมการเคลื่อนที่ โดยใช้ชิพอีกตัว เช่น ไมโครคอนโทรเลอร์ ไมโครโปรเซสเซอร์ DSP หรือไมโครคอมพิวเตอร์ ช่วยทำงานในส่วนนี้แทน เช่นงานวิจัย [2,3][5] จะใช้การ์ดการควบคุมการเคลื่อนที่ เสียบลงในสล็อตของเครื่องไมโครคอมพิวเตอร์ ซึ่งจะใช้ไมโครคอมพิวเตอร์ทำงานในส่วนของคำสั่ง จี-โค้ด หรือ งานวิจัยที่ใช้ไมโครคอนโทรเลอร์ 8 บิต [6] ทำงานในส่วนของคำสั่ง จี-โค้ด

1.2 วัตถุประสงค์ในการศึกษา

วัตถุประสงค์ในงานวิจัยนี้เป็นแนวทางในการสร้าง วงจรรวมเฉพาะงาน Application Specific Integrated Circuit (ASIC) เพื่อรองรับงานทางด้านคำสั่ง จี-โค้ด แทนการทำงานของไมโครคอนโทรเลอร์ ไมโครโปรเซสเซอร์ DSP และ ไมโครคอมพิวเตอร์ ที่ไม่ได้ออกแบบชุดคำสั่ง (Instruction Set) มาสำหรับการทำงานกับคำสั่ง จี-โค้ด โดยเฉพาะ ดังนั้นงานวิจัยนี้ได้ออกแบบ จี-โค้ดโปรเซสเซอร์ที่มีชุดคำสั่งเป็น จี-โค้ด ทำให้มีความเหมาะสมในการทำงานมากกว่า ช่วยให้การพัฒนาระบบเครื่องจักรซีเอ็นได้ง่ายขึ้น ผู้พัฒนาระบบไม่จำเป็นต้องเขียนโปรแกรมเพื่อรองรับคำสั่ง จี-โค้ด แต่จะส่งคำสั่งมาให้ จี-โค้ดโปรเซสเซอร์ทำงานแทน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.3 ทฤษฎีและแนวคิดในงานวิจัย

การทำงานของวงจรในงานวิจัยนี้จะแบ่งออกได้เป็น 4 ส่วน

1) วงจรในการประมวลผล (Arithmetic Unit) เปรียบเทียบได้กับ ALU (Arithmetic and Logic Unit) ของโปรเซสเซอร์ทั่วไปแต่แตกต่างในส่วนฟังก์ชั่นการคำนวณที่ซับซ้อน เพื่อรองรับการคำนวณการเคลื่อนที่เป็นเส้นตรงและเส้นโค้งส่วนของวงกลม (Linear & Circular Interpolation) ซึ่งการทำงานในส่วนนี้จะแตกต่างจากโปรเซสเซอร์ทั่วไปมากที่สุด คือวงจรในส่วนนี้ต้องมีความสามารถในการแก้สมการที่มีฟังก์ชั่น Cos( ), Sin( ), Tan^{-1}( ) และ Sqrt( ) ซึ่งในงานวิจัยนี้ใช้หลักการของ CORDIC (COordinate Rotation Digital Computer) [7-11] เพื่อใช้ในการคำนวณหาฟังก์ชั่นดังกล่าว

2) วงจรในการควบคุมภายใน (Control Unit) ใช้หลักการของไมโครโปรแกรม [12] ในการสร้างสัญญาณเพื่อนำไปควบคุมส่วนต่างๆ มีคำสั่งไมโครโปรแกรม 16 คำสั่ง การกระโดด การเรียกโปรแกรมย่อย และการวนรอบ

3) ส่วนของรีจิเตอร์ที่ใช้เก็บข้อมูล และเส้นทางของข้อมูล (Register and Data Path) เป็นการอ้างถึงตำแหน่งข้อมูลแบบพื้นฐานคือ การเข้าหาข้อมูลทางตรง (Direct Mode) ซึ่งจะเป็นแนวทางในการออกแบบการเข้าหาข้อมูลในรูปแบบอื่นๆได้

4) ส่วนของการเชื่อมต่อกับอุปกรณ์ภายนอก (Interface Unit) เป็นการออกแบบให้มีขาสัญญาณส่งสัญญาณออกมาติดต่อกับอุปกรณ์ภายนอกได้เหมือน ไมโครคอนโทรเลอร์ตระกูล MCS51

1.4 ขอบเขตการวิจัย

โดยเนื้อหาหลักของงานวิจัยนี้สนใจรายละเอียดของการทำงานในส่วนที่ 1 คือส่วนของวงจรในการประมวลผล ซึ่งเป็นการทำงานที่ซับซ้อน แตกต่างไปจากรูปแบบของหน่วยประมาณผลทั่วไป และยังเป็นตัวบ่งชี้ในการวัดประสิทธิภาพของโปรเซสเซอร์ได้เป็นอย่างดี ทั้งทางด้าน ความถูกต้องและทางด้านความเร็วในการทำงาน โดยในงานวิจัยนี้ได้แสดงการคำนวณและผลการคำนวณของส่วนการประมวลผล โดยแบ่งส่วนประมวลผลออกเป็นส่วนย่อยๆ เพื่อตรวจสอบความถูกต้อง ในการคำนวณ

1.5 ประโยชน์ที่ได้จากงานวิจัยนี้

ประโยชน์ที่ได้รับจากงานวิจัยโดยแบ่งเป็นข้อๆมีดังต่อไปนี้

1) วิจัยนี้เป็นแนวทางส่วนหนึ่งในการสร้าง จี-โค้ดโปรเซสเซอร์ ซึ่งงานวิจัยนี้ได้แสดงหลักการใหม่ในการออกแบบหน่วยประมวลผลของ จี-โค้ดโปรเซสเซอร์ โดยนำเสนอหลักการของ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การต่อวงจรรวมกันโดยใช้คอร์ดิคอัลกอริทึม [7-11] เพื่อแก้สมการการคำนวณที่ต้องใช้ฟังก์ชั่น Cos( ), Sin( ), Tan^{-1}( ) และ Sqrt( ) ซึ่งการคำนวณนี้เป็นการคำนวณที่ซับซ้อนกว่าการคำนวณของโปรเซสเซอร์ทั่วไป โดยในส่วนการต่อวงจรรวมกันนี้เป็นการประยุกต์จากหลักการของการต่อวงจรรวมกันของคอร์ดิคอัลกอริทึม ที่ได้ถูกใช้แก้ปัญหาสมการทางด้านอื่นมาแล้ว

2) แสดงถึงหลักการโดยละเอียดของ คอร์ดิคอัลกอริทึม สมการทางการคำนวณ, ความถูกต้องของการทำงาน, ช่วงของการตอบสนอง และการขยายช่วงในการตอบสนองโดยใช้หลักการ PreCORDIC และ PostCORDIC ผลวิเคราะห์ส่วนสำคัญต่างๆที่มีผลต่อการคำนวณของหลักการคอร์ดิคอัลกอริทึม โดยทั้งหมดนี้เป็นการรวบรวมผลมาจากงานวิจัยต่างๆ ซึ่งเป็นประโยชน์ในการประยุกต์ใช้หลักการคอร์ดิคอัลกอริทึมกับงานทางด้านอื่น

3) ผลการทำงานของคอร์ดิคอัลกอริทึมในส่วนต่างๆ แสดงค่าอินพุตตัวอย่าง, ค่าเอาท์พุต และค่าความผิดพลาด ที่เกิดขึ้นจากการทำงานของหลักการคอร์ดิคอัลกอริทึม เพื่อเป็นข้อมูลในการออกแบบนำหลักการคอร์ดิคอัลกอริทึมไปประยุกต์ใช้งาน และทำความเข้าใจการทำงานในแต่ละส่วนของการทำงานตามหลักการคอร์ดิคอัลกอริทึม

4) แสดงวงจรดิจิตอลที่สร้างจากหลักการคอร์ดิคอัลกอริทึม และวงจรดิจิตอลที่ได้จากการประยุกต์ใช้หลักการคอร์ดิคอัลกอริทึม เพื่อเป็นแนวทางในการสร้างวงจรทั้งส่วนหลักการคอร์ดิคอัลกอริทึม และ ส่วนประมวลผล จี-โด้ดโปรเซสเซอร์ รวมถึงสามารถนำวงจรที่สร้างจากหลักการคอร์ดิคอัลกอริทึมไปใช้ในการประยุกต์ทางด้านอื่น และใช้งานจริง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

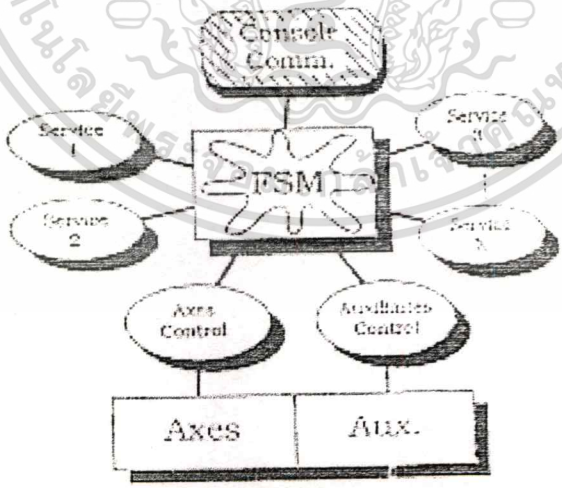
บทที่ 2
งานวิจัยที่เกี่ยวข้องและหลักการของ จี-โค้ดโปรเซสเซอร์

2.1 บทนำ

ในบทนี้กล่าวถึงงานวิจัยที่เกี่ยวข้องทั้งงานวิจัยในประเทศไทยและงานวิจัยของต่างประเทศ โดยแสดงถึงระบบต่างๆที่ได้มีการออกแบบไว้ของงานทางด้านเครื่องจักรซีเอ็นซี รวมถึงในบทนี้ได้นำเสนอโครงสร้างของการ จี-โค้ดโปรเซสเซอร์ เพื่อใช้งานในระบบเครื่องจักรซีเอ็นซี ในส่วนของการตอบสนองต่อคำสั่ง จี-โค้ด

2.2 งานวิจัยที่เกี่ยวข้อง

ในงานวิจัยทางด้านเครื่องจักรซีเอ็นซี ในช่วงแรกอยู่ในปี ค.ศ.1991 โดย J.D. Decotignie และ Michel Moreaux [13] โดยผลงานวิจัยดังกล่าว เป็นระบบที่เมื่อรับคำสั่งจากทางผู้ใช้งานแล้ว หลังจากนั้นนำคำสั่งต่างๆ ผ่านเข้าในส่วนกลาง ที่ใช้หลักการการออกแบบทางวงจรดิจิตอล คือ FMS (Finite State Machine) หลังจากนั้นระบบส่วนกลาง ทำการกระตุ้นการทำงานส่วนต่างๆของเครื่องจักร ซึ่งเห็นได้ว่าการทำงานดังกล่าวทำให้วงจรการทำงานในส่วนกลางมีขนาดใหญ่และซับซ้อน แสดงการทำงานดังรูป 2.1

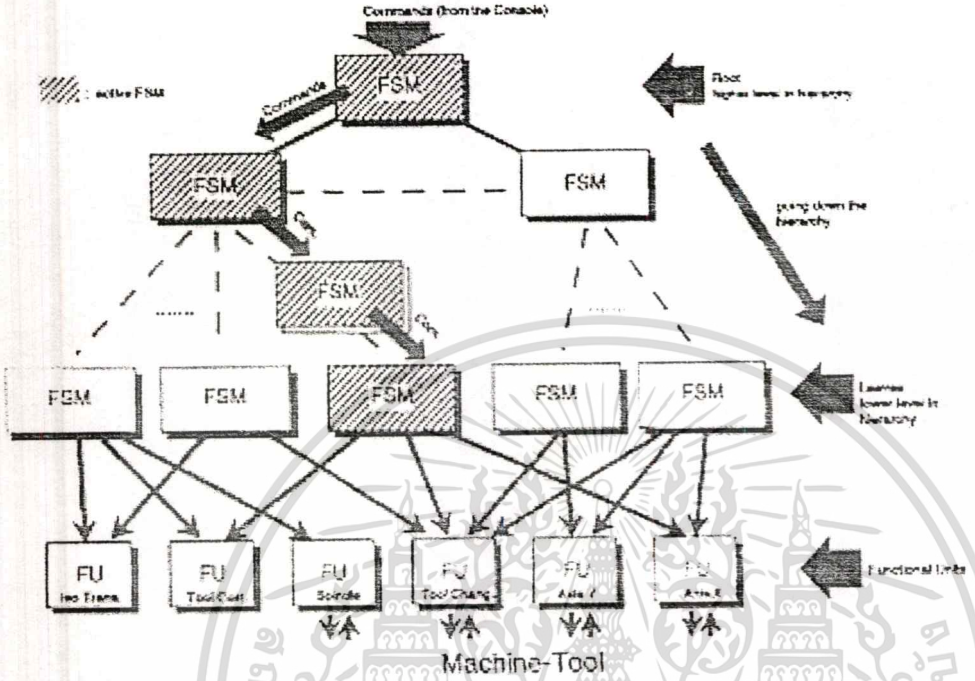


รูปที่ 2.1 โครงสร้างระบบควบคุมของเครื่องจักรซีเอ็นซี ที่ใช้หลักการ FMS

หลังจากนั้นในปี ค.ศ.1993 Michel Moreaux [14] ก็ได้เสนอหลักการ HFMS (Hierarchical Finite State Machine) โดยเมื่อมีคำสั่งจากผู้ใช้งานระบบ หลังจากนั้นทำการแตกการทำงานออกเป็นส่วนๆ โดยการทำงานส่วนแรกทำการกระตุ้นการทำงานในส่วนต่อไป จนสุดท้ายเป็นการกระตุ้นการ

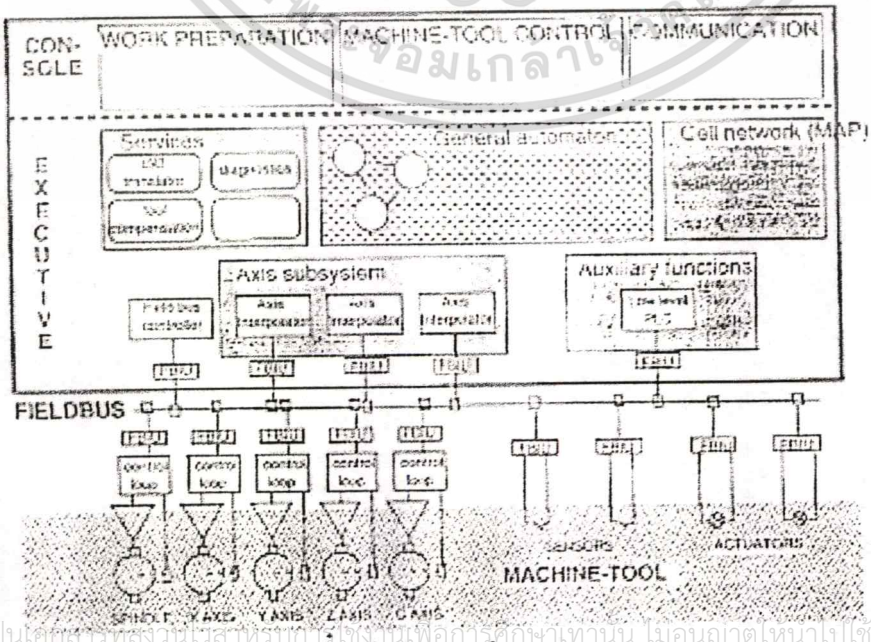
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ทำงานส่วนต่างๆของเครื่องจักร ทำให้การออกแบบวงจรในส่วนนี้มีความซับซ้อนน้อยลง แต่ก็ยังมีความยากในการออกแบบอยู่มาก แสดงในรูปที่ 2.2



รูปที่ 2.2 โครงสร้างระบบควบคุมของเครื่องจักรซีเอ็นซี ที่ใช้หลักการ HFMS

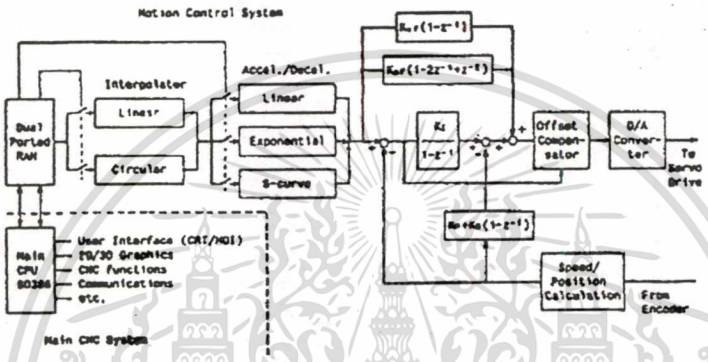
หลังจากนั้นในปี ค.ศ.1994 J.D. Decotignie ได้ปรับปรุงระบบโดยการใช้ระบบสื่อสาร ฟิลบัส (Fieldbus) ในการเชื่อมต่อกันระหว่างส่วนควบคุมหลัก และ ส่วนของเครื่องจักร ดังแสดงรูปที่ 2.3



รูปที่ 2.3 การทำงานของเครื่องจักรซีเอ็นซี ที่ใช้ระบบสื่อสาร Fieldbus

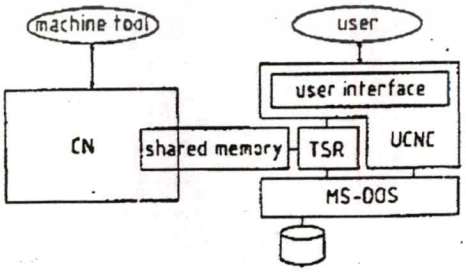
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในงานวิจัยทางด้านเครื่องจักรซีเอ็นซี ได้มีแนวคิดในการสร้างระบบอีกแบบโดยการใช้เครื่องไมโครคอมพิวเตอร์ (MicroComputer) ร่วมกับตัวประมวลผลสัญญาณดิจิตอล (DSP, Digital Signal Processor) เช่น ในปี ค.ศ. 1991 ได้มีผู้วิจัย Dong-Il Kim, Jin-I1 และ Sungkwun Kim [7][16,17] ได้นำเสนอผลงานวิจัยโดยใช้ เครื่องไมโครคอมพิวเตอร์ที่มีหน่วยประมวลผลเป็น CPU80386 เชื่อมต่อกับ หน่วยความจำแบบ 2 ทาง (Dual Ported Ram) โดยด้านหนึ่งต่อเข้ากับระบบบัสของเครื่องไมโครคอมพิวเตอร์ และอีกด้านหนึ่งต่อเข้ากับ ตัวประมวลผลสัญญาณดิจิตอล TMS 32030 โดยได้แสดงโครงสร้างดังรูปที่ 2.3



รูปที่ 2.4 โครงสร้างการควบคุม ที่ใช้ระบบไมโครคอมพิวเตอร์ กับตัวประมวลผลสัญญาณดิจิตอล

นอกจากนี้ยังมีงานวิจัยทางด้าน โปรแกรมของเครื่องจักรซีเอ็นซี โดยใช้หลักการเชิงวัตถุ (Object-Oriented) เพื่อพัฒนาความสามารถของเครื่องจักรซีเอ็นซี ซึ่งโครงสร้างของระบบได้ใช้หน่วยประมวลผลสองส่วน ส่วนแรกเหมือนเครื่องไมโครคอมพิวเตอร์ และส่วนที่สองใช้เหมือนตัวประมวลผลสัญญาณดิจิตอลในงานวิจัย [7][16-17] ซึ่งวิธีดังกล่าวได้ตีพิมพ์ปี ค.ศ. 1998 ผู้วิจัย F.Bulera, B.Fontenella,P.Nesi และ M.Perfetti [18] แสดงโครงสร้างดังรูปที่ 2.5

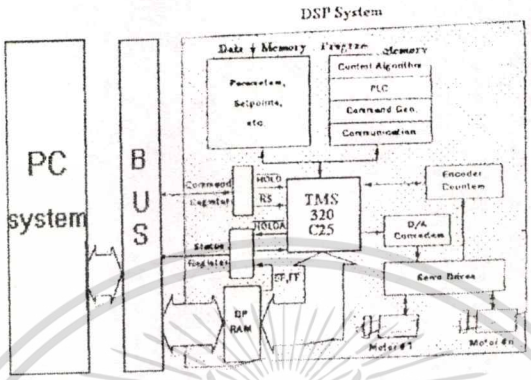


รูปที่ 2.5 โครงสร้างระบบควบคุมของเครื่องจักรซีเอ็นซี ที่พัฒนาในส่วนโปรแกรม

ในส่วนของงานวิจัยในประเทศไทยในปี พ.ศ. 2536 สินชัย กมลภิวงค์ ได้พัฒนาระบบเครื่องจักรซีเอ็นซีชนิดแกนเดียว [1] โดยใช้ไมโครคอนโทรเลอร์เป็นหลัก หลังจากนั้น พ.ศ. 2536 สมหญิง ไทยนิมิต และ วโรดม ตู้จินดา [2] ได้พัฒนาระบบซีเอ็นซีสำหรับฝึกอบรม โดยมี

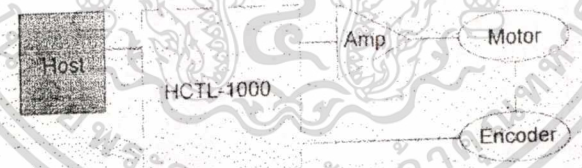
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่อนุญาตให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเอกสารทุกครั้งที่มีการนำไปใช้

โครงสร้างเป็นเครื่องไมโครคอมพิวเตอร์เชื่อมต่อกับตัวประมวลผลสัญญาณดิจิตอล (DSP, Digital Signal Processor) ซึ่งเป็นโครงสร้างที่ใกล้เคียงกับงานวิจัย [7][16-17] ในปี พ.ศ. 2537 วโรดม ตู้จินดา[3] ได้ตีพิมพ์ในวิทยานิพนธ์ การทำงานในส่วนของตัวประมวลผลสัญญาณดิจิตอล แสดงในรูปที่ 2.6



รูปที่ 2.6 โครงสร้างระบบควบคุมของเครื่องจักรซีเอ็นซี ที่เป็นงานวิจัยในประเทศ

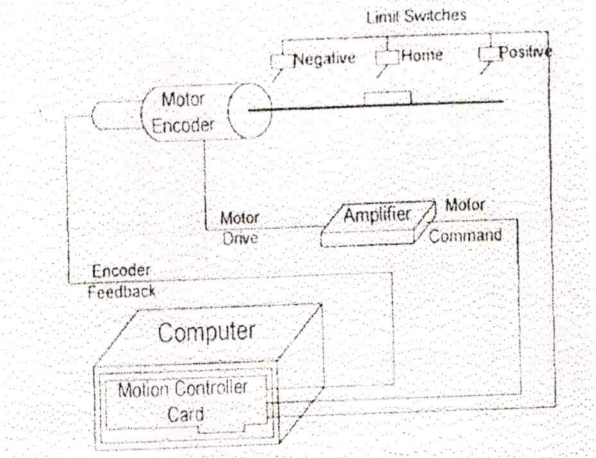
ในปี พ.ศ.2537 เริ่มมีการใช้วงจรรวมในการควบคุมการเคลื่อนที่ (Motion Control Chip) HCTL-1000 [4] เพื่อใช้แทนในส่วนของตัวประมวลผลสัญญาณดิจิตอล แสดงการทำงานของระบบดังรูปที่ 2.7



รูปที่ 2.7 การทำงานของระบบควบคุมการเคลื่อนที่โดยใช้วงจรรวม HCTL-1000

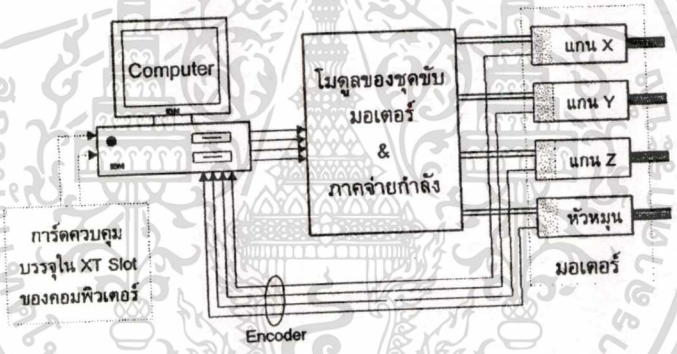
ในปี พ.ศ. 2540 มีงานวิจัยทางด้านเครื่องจักรซีเอ็นซี [5] เริ่มมีการใช้ การ์ดควบคุมสำเร็จรูป PMD5650 โดยมีโครงสร้างการใช้งานดัง รูปที่2.8 และในปีนี้ได้มีพัฒนาทางด้านโปรแกรมคอมพิวเตอร์เพื่อนำข้อมูลภาพจากการออกแบบมาควบคุมการเคลื่อนที่ 2 แกน [19]

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.8 การทำงานของระบบควบคุมการเคลื่อนที่โดยใช้การ์ดควบคุมสำเร็จรูป PMD5650

ในปี พ.ศ.2541 ได้มีการใช้โครงสร้างที่ใช้ไมโครคอมพิวเตอร์และวงจรรวม HCTL-1100 ดังรูปที่ 2.9



รูปที่ 2.9 การทำงานของระบบควบคุมการเคลื่อนที่โดยใช้วงจรรวม HCTL-1100

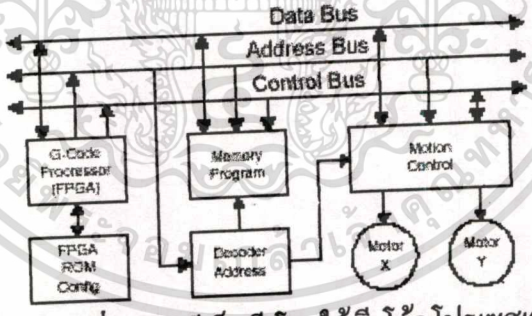
จากงานวิจัยทั้งในประเทศ และ ต่างประเทศ เพื่อสร้างเครื่องจักรซีเอ็นซีนั้น มีงานจำเป็นหลักอย่างหนึ่งของระบบคือ การรับคำสั่งจากผู้ใช้เพื่อบ่งบอกความต้องการ ซึ่งได้มีการกำหนดชุดคำสั่ง จี-โค้ด เพื่อเป็นตัวกลางในสื่อสารกันระหว่างผู้ใช้กับเครื่องจักรซีเอ็นซี โดยงานวิจัย [13-15] ใช้หลักการ FSM หรือ HFSM ในการแปลงคำสั่ง จี-โค้ด ส่วนงานวิจัยที่ใช้เครื่องไมโครคอมพิวเตอร์ [1-5][7][16-20] ได้ใช้โปรแกรมที่ทำงานบนเครื่องไมโครคอมพิวเตอร์ในการแปลงคำสั่ง จี-โค้ด แล้วส่งให้ส่วนควบคุมการเคลื่อนที่ทำงาน โดยมีโครงสร้างทั้งเป็น DSP, HCTL-1000, HCTL-1100 หรือ PMD5650 เพื่อทำการควบคุมเครื่องจักรต่อไป

ได้มีการสำรวจแนวทางของการใช้โปรเซสเซอร์และวงจรรวมดิจิตอลที่เกี่ยวข้องกับงานทางด้านการควบคุมการเคลื่อนที่ [21] โดยในส่วนแรกของการสำรวจได้กล่าวถึงการใช้ไมโครโปรเซสเซอร์และวงจรรวมดิจิตอลเช่นตัวประมวลผลสัญญาณดิจิตอล, ไมโครคอนโทรเลอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

และรวมถึงการออกแบบวงจรรวมเฉพาะทาง เพื่อนำมาประยุกต์ใช้งานทางด้านการควบคุมการเคลื่อนที่ รวมถึงได้กล่าวถึงแนวทางในการพัฒนาในอนาคตของโปรเซสเซอร์ที่ใช้ในงานควบคุมการเคลื่อนที่ โดยเป็นการรวมความสามารถในการทำงานต่างๆของระบบการควบคุมการเคลื่อนที่ให้อยู่ในโปรเซสเซอร์ตัวเดียวเพื่อสะดวกในการใช้งานมากขึ้น

ในระยะหลังในการเขียนชุดคำสั่ง จี-โค้ด เพื่อสั่งงานเครื่องจักรซีเอ็นซีนั้น ได้มีการใช้โปรแกรมคอมพิวเตอร์ช่วยออกแบบ (CAD/CAM) เป็นตัวสร้างชุดคำสั่งแทนการเขียนของคน ซึ่งผลลัพธ์ของโปรแกรมช่วยออกแบบนี้ยังเป็นคำสั่ง จี-โค้ด แต่เป็นคำสั่งที่ไม่หลากหลาย เหมือนในอดีตที่ใช้คนในการเขียนคำสั่ง โดยคำสั่งส่วนใหญ่ประกอบไปด้วย คำสั่ง G00, G01, G02 และ G03 ซึ่งเป็นคำสั่งพื้นฐานในการบอกลักษณะการเคลื่อนที่ของเครื่องจักรซีเอ็นซี ดังนั้นในงานวิจัยนี้ได้นำเสนอแนวของการสร้างหน่วยประมวลผลเฉพาะทาง จี-โค้ดโปรเซสเซอร์ เพื่อทำการแปลงชุดคำสั่ง จี-โค้ด โดยรองรับชุดคำสั่งที่สร้างจากโปรแกรมคอมพิวเตอร์ช่วยในการออกแบบ ตัดส่วนที่เป็นความสามารถในการสื่อสารกับผู้ใช้ลง ด้วยเหตุนี้ส่งผลทำให้สามารถลดขนาดของระบบโดยไม่จำเป็นต้องใช้ไมโครคอมพิวเตอร์ [1-5][7][16-20] และลดความยากในการออกแบบระบบเครื่องจักรซีเอ็นซี [13-15] ทำให้ผู้พัฒนาระบบเครื่องจักรซีเอ็นซี สะดวกและรวดเร็วขึ้น เป็นทางเลือกในการพัฒนาระบบที่ไม่ได้รองรับคำสั่งจากผู้ใช้โดยตรง แต่เป็นการรับคำสั่งจากโปรแกรมคอมพิวเตอร์ช่วยออกแบบแทน ซึ่งได้แสดงโครงสร้างการใช้งานของ จี-โค้ดโปรเซสเซอร์ ดังรูปที่2.10

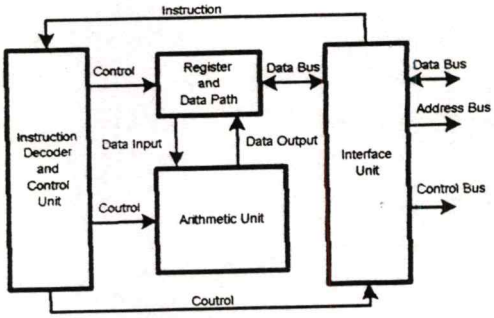


รูปที่ 2.10 การทำงานของระบบเครื่องจักรซีเอ็นซี โดยใช้ จี-โค้ดโปรเซสเซอร์

จากรูปที่ 2.10 เป็นการสร้าง จี-โค้ดโปรเซสเซอร์ โดยการใช้ FPGA แล้วนำไปต่อกับส่วนของ หน่วยความจำที่ใช้ในการเก็บคำสั่ง จี-โค้ด และ ส่วนควบคุมการเคลื่อนที่ ซึ่งหน่วยความจำที่ใช้เก็บคำสั่ง จี-โค้ด นี้อาจออกแบบให้เป็น ROM หรือ หน่วยความจำแบบ 2 ทาง (Dual Ported Ram) ทั้งหน่วยความจำและส่วนควบคุมการเคลื่อนที่จะต่ออยู่บนบัสระบบเดียวกันแต่จะถูกแยกออกจากกันโดยใช้วงจร Decoder Address

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.3 โครงสร้างของการทำงาน จี-โค้ดโปรเซสเซอร์



รูปที่ 2.11 โครงสร้างสถาปัตยกรรม จี-โค้ดโปรเซสเซอร์

โครงสร้างของ จี-โค้ดโปรเซสเซอร์ตามรูปที่ 2.11 การทำงานส่วนใหญ่จะเหมือนกับโปรเซสเซอร์ทั่วไป แต่การคำนวณในส่วนของ Arithmetic Unit จะมีการคำนวณที่ซับซ้อนกว่า รองรับชุดคำสั่งแบ่งเป็น 2 ชุดคือชุดคำสั่งที่ใช้ในการเคลื่อนย้ายข้อมูล เช่น Mov XI,100 และชุดคำสั่งในการคำนวณหาค่าเช่น G00, G02 และ G03 ดังนั้นจากคำสั่ง จี-โค้ด เช่น G00 X10 Y10 ซึ่งแสดงถึงคำสั่งในการเคลื่อนที่จากตำแหน่งปัจจุบันไปอยู่ที่ตำแหน่ง X = 10 และ Y = 10 เทียบได้ตามตารางที่ 2.1

ตารางที่ 2.1 เปรียบเทียบคำสั่ง จี-โค้ด กับชุดคำสั่งของ จี-โค้ดโปรเซสเซอร์

คำสั่ง จี-โค้ด	คำสั่งของ จี-โค้ด โปรเซสเซอร์
G00 X0 Y0	Mov XI ,0
	Mov YJ ,0
	G00
G02 X100 Y100 I50 J50	Mov XI,50
	Mov YJ,50
	Mov Dz,XX (Compiler Calculate)
	Mov N,XX (Compiler Calculate)
	G02

การทำงานของรูปที่ 2.11 เป็นโครงสร้างหลักของ จี-โค้ดโปรเซสเซอร์ ระบบนี้เริ่มการทำงานจากการถูก Reset หลังจากนั้นวงจรควบคุมภายใน สั่งให้ส่วนเชื่อมต่อกับภายนอก อ่านค่าคำสั่งเข้ามาจากหน่วยความจำ หน่วยควบคุมภายในทำการตีความหมายและทำตามคำสั่งนั้นๆ ถ้าเป็นคำสั่งที่ไม่ต้องการการคำนวณก็เป็นการทำงานโอนถ่ายค่าในส่วนของรีจิสเตอร์อย่างเดียว แต่ถ้าคำสั่งที่รับเขามามีการประมวลผลก็จะใช้หน่วยคำนวณผลทำงานด้วย เมื่อทำเสร็จก็ทำการอ่านคำสั่งใหม่เข้ามา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากการทำงานดังกล่าว คอร์ดิคอัลกอริทึม ได้ถูกมาใช้ในการแก้ปัญหาชุดคำสั่งที่ใช้ในการคำนวณการเคลื่อนที่ ซึ่งการทำงานในส่วนของชุดคำสั่งการคำนวณการเคลื่อนที่นี้เป็นการคำนวณทางคณิตศาสตร์ที่ใช้แก้สมการการแตกแกนของเวคเตอร์อินพุต ให้อยู่ในรูปของเวคเตอร์ในแนวแกนนอนและแนวแกนตั้ง โดยสมการการแตกแกนของเวคเตอร์นี้เพื่อแก้สมการแล้วจำเป็นต้องใช้ ฟังก์ชั่น Sin( ), Cos( ) และ Tan^{-1}( ) ดังนั้น คอร์ดิคอัลกอริทึม ที่มีคุณสมบัติในการคำนวณหาค่าของฟังก์ชั่น Sin( ), Cos( ) และ Tan^{-1}( ) และมีคุณสมบัติที่สามารถนำมาสร้างเป็นวงจรดิจิตอลได้ง่าย จึงถูกนำมาใช้ในงานวิจัยนี้ โดยใช้ในส่วนของวงจรในการประมวลผล ซึ่งได้กล่าวไว้ในส่วนต่อไป

2.4 ส่วนการทำงานต่างๆของ จี-โค้ด โปรเซสเซอร์

ในการทำงานของ จี-โค้ดโปรเซสเซอร์แบบออกได้เป็น 4 ส่วน 1)วงจรในการประมวลผล (Arithmetic Unit) เปรียบเทียบได้กับ ALU (Arithmetic and Logic Unit) 2) วงจรในการควบคุมภายใน (Control Unit) 3)ส่วนของการเก็บข้อมูล และเส้นทางของข้อมูล (Register and Data Path) และ 4) ส่วนของการเชื่อมต่อกับอุปกรณ์ภายนอก (Interface)

2.4.1 การทำงานของส่วนประมาณผล

การทำงานของ Arithmetic Unit แบ่งการทำงานเป็น 2ส่วน ส่วนแรกเป็นการคำนวณการเคลื่อนที่ในแบบเส้นตรง (Linear Interpolation) โดยมีอินพุตเป็นตำแหน่งปลายทางในการเคลื่อนอัตราเร็วและอัตราเร่ง โดยมีเอาท์พุทเป็นค่าของความเร็วและความเร่ง ในส่วนที่ 2 เป็นการคำนวณการเคลื่อนที่ในแบบเส้นโค้งส่วนของวงกลม (Circular Interpolation) โดยรับค่ามุมกับจุดศูนย์กลางเพื่อคำนวณหาค่าตำแหน่งต่อไปในการประมาณค่า แล้วนำตำแหน่งที่คำนวณนี้ส่งให้การทำงานในส่วนแรกเพื่อหาค่า ความเร็วและความเร่ง [5][7] โดยแสดงรายละเอียดในบทที่ 4

2.4.2 การทำงานของส่วนควบคุมภายใน

การทำงานในส่วนนี้เป็นการสร้างสัญญาณควบคุมภายในของโปรเซสเซอร์ ทั้งหมดโดยอาศัยหลักการ ไมโครโปรแกรม [12] ซึ่งงานนี้ได้ออกแบบวงจรเพื่อลองรับชุดคำสั่งไมโครโปรแกรมจำนวน 16 คำสั่งตามตารางที่ 2.2

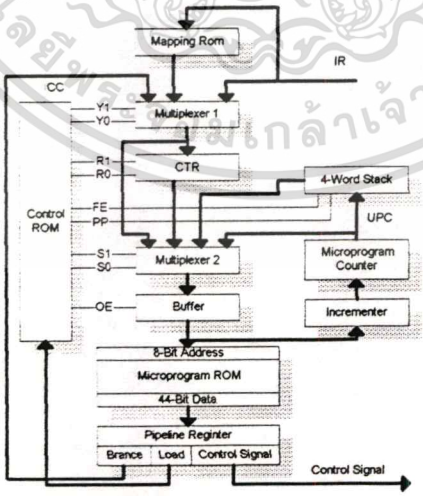
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 2.2 ชุดคำสั่งไมโครโปรแกรมของ Control Unit

OpCode/ Mnemonic	Name	Description
0000. JZ	Jump Zero	กระโดดไปตำแหน่ง 0 เพื่อทำการอ่านคำสั่งใหม่
0001. CJS	Condition Jump Subroutine	ตรวจสอบบิต CC ตัดสินใจเรียกโปรแกรมย่อย หรือทำงานตำแหน่งต่อไป
0010. JMAP	Jump Map	กระโดดตามค่าใน Mapping Rom
0011. CJP	Condition Jump PL	ตรวจสอบบิต CC ตัดสินใจกระโดดตาม PL หรือทำงานตำแหน่งต่อไป
0100. PUSH	Push & Conditional Load CTR	เก็บค่า μPC ลง Stack และ นำค่า PL ใส่ให้กับ CTR
0101. JSRP	Jump Subroutine CTR/PL	ตรวจสอบบิต CC ตัดสินใจเรียกโปรแกรมย่อยตาม PL หรือเรียกโปรแกรมย่อยตาม CTR
0110. CJV	Conditional Jump Vector	ตรวจสอบบิต CC ตัดสินใจกระโดดตามชุดคำสั่งที่เข้ามา หรือ ทำงานตำแหน่งต่อไป
0111. JRP	Conditional Jump CTR/PL	ตรวจสอบบิต CC ตัดสินใจกระโดดไปตำแหน่ง CTR หรือ ทำงานตำแหน่ง PL
1000. RFCT	Repeat Loop CTR ≠0	ตรวจสอบบิต CTR ตัดสินใจกระโดดไปตำแหน่ง Top Stack หรือ ทำงานตำแหน่งต่อไป
1001. RPCT	Repeat PL CTR ≠0	ตรวจสอบบิต CTR ตัดสินใจกระโดดไปตำแหน่ง PL หรือ ทำงานตำแหน่งต่อไป
1010. CRTN	Conditional Return	ตรวจสอบบิต CC ตัดสินใจPop Stack แล้วกระโดดไปตำแหน่ง Top Stack หรือ ทำงานตำแหน่งต่อไป
1011. CJPP	Conditional Jump PL and POP	ตรวจสอบบิต CC ตัดสินใจทำการ Pop Stack แล้วกระโดดไปตำแหน่ง PL หรือ ทำงานตำแหน่งต่อไป
1100. LDCT	Load CTR and Continue	ใส่ค่า PL ให้กับ CTR
1101. LOOP	Test End Loop	ตรวจสอบบิต CC ตัดสินใจกระโดดไปตำแหน่ง Top Stack หรือ ทำงานตำแหน่งต่อไป
1110. CONT	Continue	ทำงานในตำแหน่งต่อไป
1111. TWB	Three-Way Branch	ตรวจสอบบิต CC ตัดสินใจ ทำการ Pop Stack แล้วกระโดดไปตำแหน่ง Top Stack หรือ กระโดดไปที่ตำแหน่ง PL

* CTR = Counter Register, PL = Pipeline Register

โครงสร้างของส่วนควบคุมภายใน



รูปที่ 2.12 โครงสร้างของส่วนควบคุมภายใน

จากรูปที่ 2.12 เป็นโครงสร้างของหน่วยควบคุมภายในประกอบด้วยส่วนต่างๆดังต่อไปนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1) Mapping Rom เป็นรอมที่ใช้ในการแปลง Instruction ให้เป็นตำแหน่งที่เก็บชุดคำสั่ง ไมโครโปรแกรม เพื่อใช้ทำงานกับคำสั่งที่เข้ามานั้นๆ

2) Multiplexer1 เป็นตัวเลือกข้อมูล เพื่อส่งให้ CTR และ Multiplexer2

3) Counter Register (CTR) เป็นวงจรนับการวนรอบต่างๆ

4) Multiplexer2 เป็นตัวเลือกข้อมูลจาก Multiplexer1, CTR, 4-Word Stack หรือ μPC เพื่อส่งให้ Buffer

5) Buffer เป็นรีจีสเตอร์ที่ใช้ในการเก็บค่าเพื่ออ้างแอดเดรสให้กับ μROM

6) Microprogram ROM (μROM)เป็นส่วนที่เก็บไมโครโปรแกรมต่างๆ ซึ่งจะเป็นชุดคำสั่งที่ได้จากการเรียงของคำสั่งในตารางที่ 1 ซึ่งที่แอดเดรส 0 เป็น ไมรโครโปรแกรมที่ทำงานอ่านข้อมูลคำสั่งจากภายนอก

7) Pipeline Register (PL) เป็นรีจีสเตอร์ข้อมูลที่ได้จาก μROM ซึ่งข้อมูลจะประกอบด้วยข้อมูล 3 ส่วน คือ Load ประกอบด้วยค่า 4 บิต เป็น OpCode ตามตารางที่ 1 เพื่อเป็นโปรแกรมในการทำงาน, Branch เป็นข้อมูลที่ใช้ในกรณีที่มีการกระโดดของไมโครโปรแกรม และ Control Signal เป็นสัญญาณควบคุมภายในของโปรเซสเซอร์ทั้งหมด

8) Incrementer เป็นวงจรที่ใช้ในการบวกค่าของอินพุทขึ้นไปอีก 1

9) Microprogram Counter (μPC)เป็นตัวกำหนดตำแหน่งของไมโครโปรแกรมที่กำลังทำงานอยู่ในขณะนั้น

10)4-Word Stack เป็นรีจีสเตอร์ขนาด 4 ไบท์ เพื่อทำงานในส่วนของการ Push, Pop, การเรียกโปรแกรมย่อย

11)Control Rom เป็นตัวสร้างสัญญาณควบคุมการทำงานของ Control Unit โดยมีอินพุทเป็น OpCode ที่ได้จาก PL และ CC เพื่อสร้างสัญญาณ ในการควบคุมต่างๆ

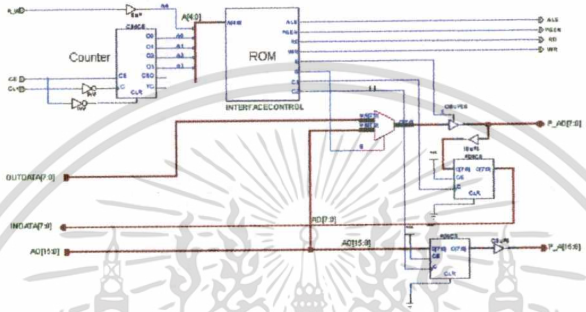
การทำงานจะเริ่มจากการที่ ระบบถูก Reset ค่าของ Buffer มีค่าเป็น 0 ทำให้ μROM ถูกอ้างที่ตำแหน่ง 0 ซึ่งเป็นชุดคำสั่งไมโครโปรแกรมเพื่อทำงานดึงค่าคำสั่งจากภายนอก หลังจากที่ได้ค่าคำสั่งภายนอกมาไว้ใน IRแล้ว ก็จะเริ่มทำตามคั่งสั่งใหม่ที่ดึงเข้ามานี้โดยส่งผ่านไปที่ Mapping ROM เพื่อกำหนดตำแหน่งของชุดคำสั่งที่อยู่ μROM โดยชุดคำสั่งที่เก็บอยู่ในตำแหน่งที่อ้างนี้จะเป็นไมโครโปรแกรมที่ทำงานตามคำสั่งที่ดึงเข้ามานั้น เมื่อเสร็จแล้วก็กระโดมาที่ μROM ในตำแหน่ง 0 ใหม่เพื่อทำการดึงคำสั่งใหม่เข้ามาทำงาน

2.4.3 การทำงานของส่วนเชื่อมต่ออุปกรณ์ภายนอก

วงจรในส่วนนี้เป็นการสร้างสัญญาณเพื่อนำออกไปควบคุมอุปกรณ์ภายนอกโดยนำเสนอการออกแบบให้ สัญญาณต่างๆ เหมือนกับโปรเซสเซอร์ Intal MCS 51 มีขาสัญญาณ บัสแอดเดรส

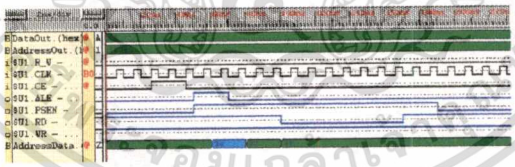
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขนาด 16 บิต, บัสข้อมูลขนาด 8 บิต ซ้อนทับอยู่บน บัสแอดเดรส, สัญญาณ ALE, สัญญาณ PSEN, สัญญาณ WR และสัญญาณ RD โดยมีอินพุทคือ สัญญาณ R/W, สัญญาณ Clock และสัญญาณ CE เป็นสัญญาณควบคุม โดยในการออกแบบได้ใช้รอมในการสร้างสัญญาณทั้งรูปแบบในการอ่านและรูปแบบในการเขียนโดยใช้สัญญาณ R/W ในการเลือกว่าใช้รูปแบบการอ่านหรือรูปแบบการเขียน สัญญาณ CE เป็นตัวปิดวงจรนับเพื่อสร้างค่า แอดเดรสที่เพิ่มขึ้นให้กลับรอมแล้วนำ ข้อมูลที่ได้จากรอมออกมาเป็นสัญญาณเอาท์พุทต่างๆ และข้อมูลอีกส่วนหนึ่งก็นำมาเป็นสัญญาณในการควบคุมภายใน ตามรูปที่ 2.13

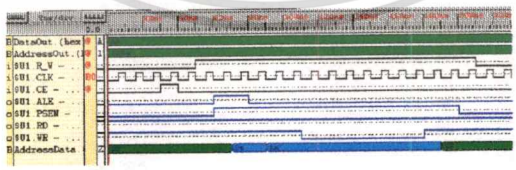


รูปที่ 2.13 วงจรการทำงานในส่วนเชื่อมต่อกับอุปกรณ์ภายนอก

วงจรในการเชื่อมต่อกับอุปกรณ์ภายนอกจะแบ่งออกเป็นสองส่วนคือการอ่านข้อมูลจากอุปกรณ์ภายนอก และการเขียนข้อมูลไปยังอุปกรณ์ภายนอก ทดลองทาง Founction Diagram ได้ผลการทำงานตามรูปที่ 2.14



(a)



(b)

รูปที่ 2.14 (a) เป็นสัญญาณในการอ่านค่าจากอุปกรณ์ภายนอก
(b) เป็นสัญญาณในการเขียนอุปกรณ์ภายนอก

จากรูปที่ 2.14 สัญญาณที่เกิดขึ้นทั้งการอ่านและการเขียนจะเริ่มจากเกิดสัญญาณ CE เป็นตัวเริ่มต้น หลังจากนั้นเอาท์พุทจะสร้างสัญญาณ ALE เพื่อให้อุปกรณ์ภายนอกรู้ว่าค่าแอดเดรส ได้ถูกส่งออกมาที่บัส หลังจากนั้นสัญญาณ WR จะทำงานถ้าเป็นกรณีที่ต้องการเขียนข้อมูลไปยังอุปกรณ์

เอกสารนี้มีลิขสิทธิ์ ห้ามนำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภายนอกแล้วจึงส่งข้อมูลที่ต้องการเขียนไปยังอุปกรณ์ภายนอกโดยใช้ยังบัสแอดเดรสดาต้า แต่ถ้าในกรณีอ่านค่าจากอุปกรณ์ภายนอก สัญญาณ RD จะทำงานแล้วจึงทำการอ่านข้อมูลจากบัสแอดเดรสดาต้า

2.4.4 การทำงานหน่วยเก็บข้อมูลและเส้นทางข้อมูล

วงจรการทำงานในส่วนนี้เป็นส่วนที่เชื่อมต่อส่วนต่างๆเข้าด้วยกันรวมถึงการจัดเก็บข้อมูลเพื่อใช้ในการคำนวณและเก็บผลลัพธ์ที่ได้ ซึ่งรีจิสเตอร์ที่สำคัญใน จี-โค้ด โปรเซสเซอร์ มีดังนี้

1) XI เป็นรีจิสเตอร์ที่เก็บค่าของ ตำแหน่งปลายทางตามแกน X ในการคำนวณการเคลื่อนที่เป็นเส้นตรง และใช้เป็นค่าจุดศูนย์กลางแนวแกน X ในการคำนวณการเคลื่อนเป็นเส้นโค้งส่วนของวงกลม

2) YJ เป็นรีจิสเตอร์ที่ทำงานเหมือน XI แต่จะเก็บข้อมูลในแนวแกน Y

3) Vmax เป็นรีจิสเตอร์ ที่เก็บค่าอัตราเร็วเพื่อใช้ในการคำนวณ

4) Amax เป็นรีจิสเตอร์ ที่เก็บค่าอัตราเร่ง

5) Dz เป็นรีจิสเตอร์ที่ใช้เก็บค่าของมุมที่ใช้ในการคำนวณตาม สมาการที่ (3)

6) N เป็นรีจิสเตอร์ที่ใช้เก็บค่าการ วนรอบคำนวณการเคลื่อนที่เป็นเส้นโค้งส่วนของวงกลม

7) X0 เป็นรีจิสเตอร์เก็บค่าของตำแหน่งต่อไปที่คำนวณได้ในแนวแกน X

8) Y0 เป็นรีจิสเตอร์เก็บค่าของตำแหน่งต่อไปที่คำนวณได้ในแนวแกน Y

9) Vx รีจิสเตอร์ที่เก็บค่าอัตราเร็วที่คำนวณได้ ในแนวแกน X

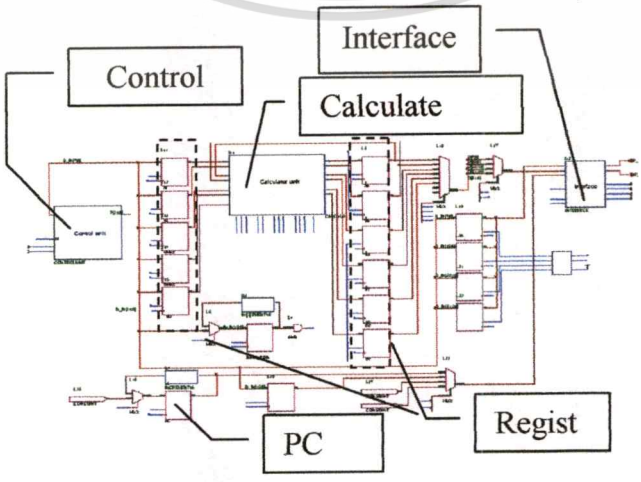
10) Vy รีจิสเตอร์ที่เก็บค่าอัตราเร็วที่คำนวณได้ ในแนวแกน Y

11) Ax รีจิสเตอร์ที่เก็บค่าอัตราเร่ง ในแนวแกน X

12) Ay รีจิสเตอร์ที่เก็บค่าอัตราเร่ง ในแนวแกน Y

13) PC รีจิสเตอร์ เป็น โปรแกรมเค้าเตอร์ของโปรเซสเซอร์

ในส่วนของรีจิสเตอร์ในการเก็บข้อมูลยังมี รีจิสเตอร์ที่ใช้ในการทำงานของ Addressing Mode ต่างๆด้วยซึ่งในงานนี้จะใช้แสดงเป็นDirect Mode เพียงอย่างเดียว โดยมีโครงสร้างตามรูปที่ 2.15



รูปที่ 2.15 โครงสร้างของเส้นทางข้อมูลและส่วนเก็บข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.5 สรุปและแนวทางการวิจัย

ในบทนี้แสดงถึงสถาปัตยกรรม จี-โค้ดโปรเซสเซอร์ เพื่อแก้ไขความไม่เหมาะสมกันระหว่างชุดคำสั่ง จี-โค้ด กับชุดคำสั่งของโปรเซสเซอร์ทั่วไป โดยแสดงโครงสร้างที่มีส่วนสำคัญทั้งหมด 4 ส่วน คือส่วนประมวลผล, ส่วนควบคุมภายใน ที่ใช้หลักการสร้างสัญญาณควบคุมโดยใช้ไมโครโปรแกรม, การแสดงโครงสร้างของหน่วยเก็บข้อมูลกับเส้นทางข้อมูล และ การแสดงวงจรในส่วนเชื่อมต่อกับอุปกรณ์ภายนอก

แนวทางการวิจัย เห็นได้ว่าโครงสร้างของ จี-โค้ดโปรเซสเซอร์ใกล้เคียงกับโปรเซสเซอร์ทั่วไป แต่มีข้อแตกต่างอย่างมากอยู่ที่ส่วนประมวลผล ซึ่งจำเป็นต้องมีการคำนวณที่ซับซ้อนกว่า รวมถึงต้องมีความถูกต้อง สิ่งนี้เป็นสิ่งสำคัญในการวัดประสิทธิภาพของ จี-โค้ดโปรเซสเซอร์ ดังนั้นในการวิจัยได้นำเสนอ หลักการการทำงาน, โครงสร้างของหน่วยประมาณผล, วงจรการทำงาน, การวิเคราะห์ผล และวัดผลทั้งทางด้านความถูกต้อง ความเร็วในการทำงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3
การทำงานของ คอร์ดิคอัลกอริทึม

3.1 บทนำ

จากที่ได้กล่าวมาแล้ว ส่วนที่แตกต่างกันมากที่สุดของไมโครโปรเซสเซอร์ กับ จี-โค้ด โปรเซสเซอร์ ซึ่งแตกต่างและน่าสนใจในการทำวิจัยคือส่วนของวงจรในการประมวลผลซึ่งในบทนี้กล่าวถึงหลักการทางคณิตศาสตร์และวงจรพื้นฐานเพื่อนำไปสู่การประยุกต์ใช้ แก้ปัญหาส่วนของการประมวลผลของ จี-โค้ดโปรเซสเซอร์

3.2 หลักการของ CORDIC (COordinate Rotation Digital Computing)

หลักการทำงานของคอร์ดิค (CORDIC) [8-11] เป็นหลักการที่เกิดจาก คณิตศาสตร์ พื้นฐานที่ใช้การหมุนเวคเตอร์ เพื่อหาค่าผลลัพธ์ ซึ่งหลักการนี้เป็นหลักการที่เหมาะสมกับการนำมาใช้กับการคำนวณในคอมพิวเตอร์เป็นอย่างมาก เนื่องจากหลักการของคอร์ดิค สุดท้ายในการคำนวณใช้แค่การบวกและการเลื่อนค่าข้อมูล ส่งผลให้สามารถนำไปสร้างเป็นวงจรทางอิเล็กทรอนิกส์ ได้ง่ายด้วย

$$\begin{aligned} x_{i+1} &= k_m(x_i - m\sigma_i 2^{-S(m,i)} y_i) \\ y_{i+1} &= k_m(y_i + \sigma_i 2^{-S(m,i)} x_i) \\ z_{i+1} &= z_i - \sigma_i \alpha_{m,i} \end{aligned} \qquad \text{.....(3.1)}$$

$$\sigma_i = sign(z_i) \quad OR \quad \sigma_i = -sign(y_i) \qquad \text{.....(3.2)}$$

$$\alpha_{m,i} = \frac{1}{\sqrt{m}} \tan^{-1}(\sqrt{m}\,2^{-S(m,i)}) = \begin{cases} \tan^{-1}(2^{-S(m,i)}) & ;m=1 \\ 2^{-S(m,i)} & ;m=0 \\ \tanh^{-1}(2^{-S(m,i)}) & ;m=-1 \end{cases} \qquad \text{.....(3.3)}$$

สมการที่ 3.1 เป็นสมการ พื้นฐานของคอร์ดิค โดยมีความหมายต่อไปนี้ x, y และ z มีค่าตามตารางที่ 3.1 แสดงถึงอินพุต และ เอาท์พุต ของระบบ, m มีค่าเป็น {-1,0,1} เพื่อกำหนดระบบการทำงานของคอร์ดิค ว่าเป็นการทำงานในรูปแบบใดตามตารางที่ 3.1, σ_i มีค่าเป็น{-1,1}เพื่อกำหนดทิศทางของการหมุนเป็นไปตามสมการที่ 3.2, $s(m,i)$ มีค่าตามตารางที่ 3.3 เป็นการเลื่อนบิตในรูปแบบของเลขฐาน 2 , $\alpha(m,i)$ มีค่าตามสมการที่ 3.3 เป็นตัวปรับเปลี่ยนขนาดของมุมในการหมุน และ i

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เป็นตัวนับจำนวนรอบในการคำนวณค่า สมการที่ 3.1 นี้เริ่มการทำงานจากการใส่ค่าอินพุต x_0 , y_0 และ z_0 หลังจากนั้นทำการวนลูปเปลี่ยนค่า i จาก 0 ไปจนถึงค่า n โดยผลลัพธ์ที่ได้จากการทำงานนี้จะขึ้นกับ การเลือกค่า m และการเลือกใช้งานค่าของ σ_i ซึ่งถ้า $\sigma_i = sign(z_i)$ ส่งผลทำให้ $z_n \to 0$ ส่วนถ้าใช้ $\sigma_i = -sign(y_i)$ ส่งผลทำให้ $y_n \to 0$ ซึ่งผลลัพธ์ของการทำงานทั้งหมดแสดงตามตารางที่ 3.1

ตารางที่ 3.1 การทำงานรูปแบบต่างๆของคอร์ดิค

	$z_n \to 0$ (การหมุนเวคเตอร์(CR))	$y_n \to 0$ (การแปลงเวคเตอร์(CV))
$m=-1$	$x_n = k_{-1}(x_0\cosh(z_0)+y_0\sinh(z_0))$ $y_n = k_{-1}(y_0\cosh(z_0)+x_0\sinh(z_0))$ $z_n = 0$	$x_n = k_{-1}\sqrt{x_0^{\,2}+y_0^{\,2}}$ $y_n = 0$ $z_n = z_0+\tanh^{-1}(y_0/x_0)$
$m=0$	$x_n = x_0$ $y_n = y_0+z_0x_0$ $z_n = 0$	$x_n = x_0$ $y_n = 0$ $z_n = z_0+y_0/x_0$
$m=1$	$x_n = k_1(x_0\cos(z_0)-y_0\sin(z_0))$ $y_n = k_1(y_0\cos(z_0)+x_0\sin(z_0))$ $z_n = 0$	$x_n = k_1\sqrt{x_0^{\,2}+y_0^{\,2}}$ $y_n = 0$ $z_n = z_0\tan^{-1}(y_0/x_0)$

จากตารางที่ 3.1 การทำงานของคอร์ดิค แบบการทำงานออกเป็นสองรูปแบบคือการหมุนเวคเตอร์ (Rotation) และการแปลงเวคเตอร์ (Vectoring) โดยใช้ค่าของ m เป็นตัวเลือกรายละเอียดของการทำงาน อย่างไรก็ตามค่าที่อยู่ในตารางที่ 3.1 บางค่ายังติดค่าคงที่ k_m อยู่ดังนั้นจึงนำตารางที่ 3.1 มาปรับปรุงค่าโดยปรับปรุงเฉพาะค่าของ x_n และค่าของ y_n โดยใช้สมการที่ 3.4 และ สมการที่ 3.5

$$x_{j+1} = x_j + \gamma(m,j)2^{-\tau(m,j)}x_j$$
$$y_{j+1} = y_j + \gamma(m,j)2^{-\tau(m,j)}y_j \qquad \text{.....(3.4)}$$

$$\gamma(m,j) = sign(a(m,j))$$
$$\tau(m,j) = |a(m,j)| \qquad \text{.....(3.5)}$$

ทำการปรับปรุงค่าโดยการกำหนดให้ $x_j = x_n$ และ $y_j = y_n$ เมื่อ $j = 0$ เพื่อนำไปกำหนดค่าเริ่มต้นของสมการที่ 3.4 ทำการวนลูปค่า j เท่ากับ 0 จนถึง N สรุปการทำงานของสมการที่ 3.1 และสมการ

เอกสารนี้เป็นการเผยแพร่ผลงานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้ใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ที่ 3.4ได้ผลลัพธ์ตามตารางที่ 3.2

ตารางที่ 3.2 การทำงานรูปแบบต่างๆของคอร์ดิค หลังการปรับปรุงค่าแล้ว

	การหมุนเวคเตอร์(CR)	การแปลงเวคเตอร์(CV)
$m=-1$	$x = x_0\cosh(z_0) + y_0\sinh(z_0)$ $y = y_0\cosh(z_0) + x_0\sinh(z_0)$ $z = 0$	$x = \sqrt{x_0^{\,2} + y_0^{\,2}}$ $y = 0$ $z = z_0 + \tanh^{-1}(y_0/x_0)$
$m=0$	$x = x_0$ $y = y_0 + z_0 x_0$ $z = 0$	$x = x_0$ $y = 0$ $z = z_0 + y_0/x_0$
$m=1$	$x = x_0\cos(z_0) - y_0\sin(z_0)$ $y = y_0\cos(z_0) + x_0\sin(z_0)$ $z = 0$	$x = \sqrt{x_0^{\,2} + y_0^{\,2}}$ $y = 0$ $z = z_0\tan^{-1}(y_0/x_0)$

จากสมการที่ 3.1 และสมการที่ 3.4 มีค่าคงที่ยังไม่ทราบค่าอีกสองตัวคือ $s(m,i)$ และ $a(m,j)$ ซึ่งค่าดังกล่าวนี้จะมีผลกับความถูกต้องของการคำนวณซึ่งมีตัวอย่างของค่า [11] ตามตารางที่ 3.3a และตารางที่ 3.3b ตามลำดับ

ตารางที่ 3.3 ค่าคงที่ใช้ในสมการของคอร์ดิค

i	$S(m,i)$	i	$S(m,i)$	i	$S(m,i)$	i	$S(m,i)$	i	$S(m,i)$	i	$S(m,i)$
1	1	6	3	11	7	16	12	21	16	26	21
2	2	7	4	12	8	17	13	22	17	27	22
3	2	8	5	13	9	18	13	23	18	28	23
4	2	9	6	14	10	19	14	24	19	29	24
5	2	10	6	15	11	20	15	25	20	-	-

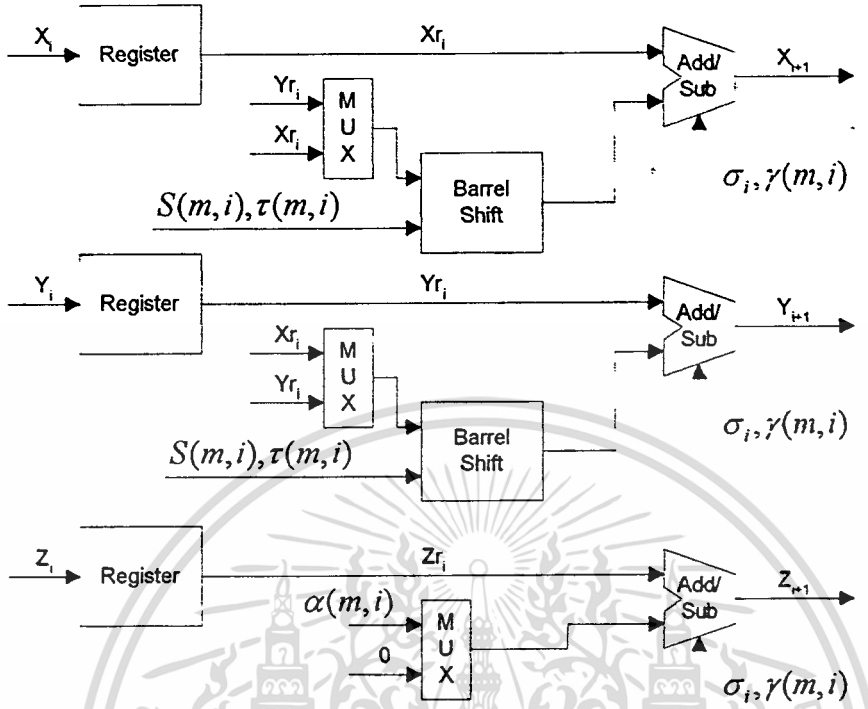
(a)

m	$a(m,1)$	$a(m,2)$	$a(m,3)$	$a(m,4)$	$a(m,5)$	$a(m,6)$	$a(m,7)$	$a(m,8)$
1	-2	4	-5	6	0	17	-20	0
0	0	0	0	0	0	0	0	0
-1	2	4	0	6	-6	0	-20	-21

(b)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.3 โครงสร้างการทำงานของ Block Diagram พื้นฐาน ของ คอร์ดิค

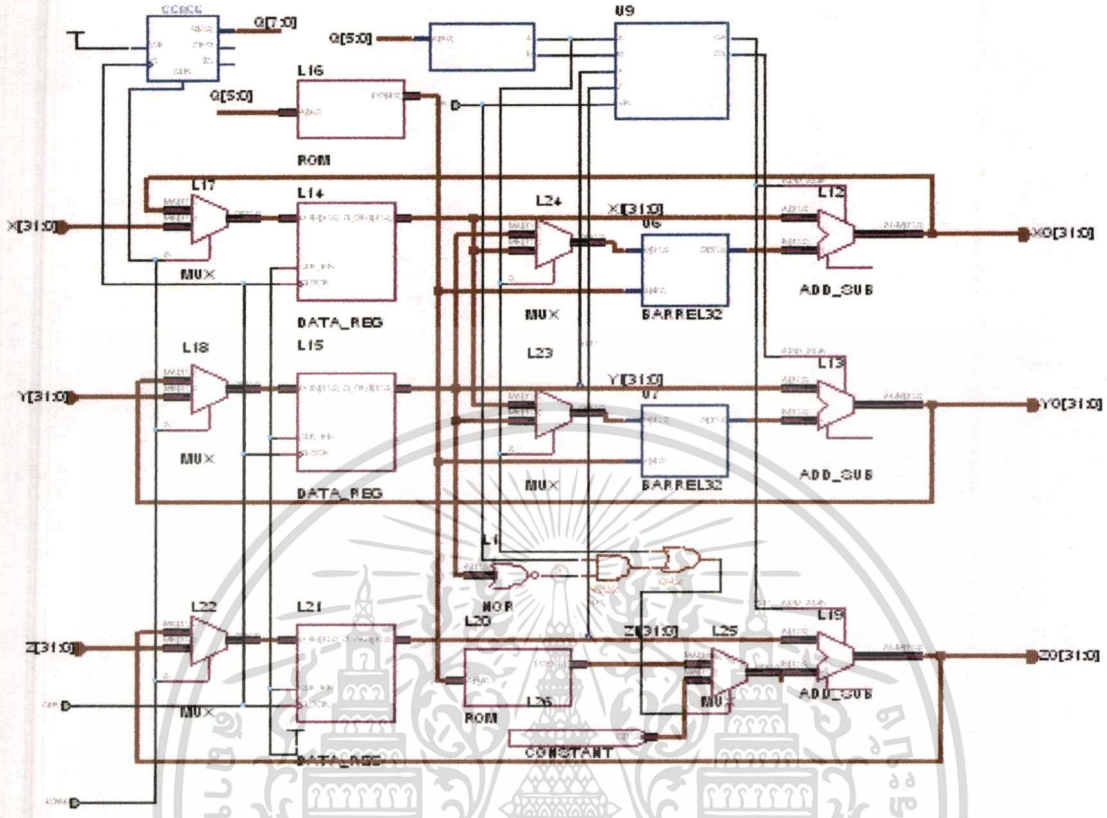


รูปที่ 3.1 Block Diagram CORDIC

จาก รูปที่ 3.1 เป็นโครงสร้างพื้นฐานเริ่มจากการกำหนดค่า $i=0$ กำหนดค่า m ตามรูปแบบการทำงานที่ต้องการ และกำหนดให้วงจร MUX เลือกค่าข้อมูลด้านบนเป็นเอาท์พุต ซึ่งทั้งหมดเป็นค่าเริ่มต้นในการทำงาน โดยมีค่า x_0 , y_0 และ z_0 เป็นค่าอินพุตของระบบ นำค่าอินพุตเก็บไว้ในตัว Register ทำให้เกิดค่า x_{r0} , y_{r0} และ z_{r0} จากนั้นนำค่า x_{r0} ไปบวกหรือลบกับค่า $y_{r0}2^{-S(m,0)}$ ที่ได้จาก Barrel Shift โดยกำหนดการทำงานเป็นบวกหรือลบได้จาก σ_i ตามสมการที่ 3.2 จากการทำงานข้างต้นสามารถนำไปประยุกต์ใช้กับการทำงานเช่นเดียวกันในส่วนของ y ซึ่งค่าผลลัพธ์มีค่าเป็น $y_1' = y_{r0} + \sigma_0 x_{r0} 2^{-S(m,0)}$, $x_1 = x_{r0} - \sigma_0 y_{r0} 2^{-S(m,0)}$ และในส่วนของ z ได้ค่าผลลัพธ์เป็น $z_1 = z_0 - \sigma_0 \alpha(m,0)$ ในขั้นตอนต่อไปนำค่า $i=1$ นำค่า x_1 , y_1 และ z_1 ไปเป็น อินพุตของระบบเพื่อหาค่า x_2, y_2 และ z_2 ทำการวนรอบเป็นจำนวน n รอบ ผลลัพธ์ ที่ได้จะเป็นไปตามตารางที่ 3.1 เพื่อปรับปรุงค่าจากตารางที่ 3.1 ให้ได้ค่าตามตารางที่ 3.2 โดยกำหนดให้วงจร MUX เลือกข้อมูลด้านล่างเป็นเอาท์พุต, Barrel Shift ใช้ข้อมูล $\tau(m,j)$ เป็นค่าที่ใช้กำหนดการเลื่อนบิตข้อมูล และการกำหนดค่าการบวกหรือลบให้ค่าของ $\gamma(m,j)$ ทำการวนรอบกำหนดค่า $j=0$ ถึง N ค่าของเอาท์พุตของการทำงานนี้เป็นค่าตามตารางที่ 3.2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.4 การทำงานของ วงจรคอร์ดิค



รูปที่ 3.2 วงจรคอร์ดิค

รูปที่ 3.2 เป็นวงจรของคอร์ดิค ซึ่งมีวงจรสำคัญคือ วงจร Register, Adder and Subtactor, Multiplax, ROM และวงจร Barral shift

3.5 การปรับปรุงการคำนวณค่าคอร์ดิค

ในการคำนวณค่าของคอร์ดิค ที่ผ่านมา อินพุตของระบบยังไม่รองรับค่าที่เป็นไปได้ทั้งหมด โดยยอมรับค่าเวคเตอร์ (x_0, y_0) ที่อยู่ในช่วง $\pi/2$ ถึง $-\pi/2$ เท่านั้นดังนั้นเพื่อให้การคำนวณตอบสนองทุกค่าที่เป็นไปได้ จึงได้มีการกำหนดการทำงานของ วงจรในส่วน PreCORDIC และวงจรในส่วน PostCORDIC [9] โดยมีหลักการทำงานดังต่อไปนี้

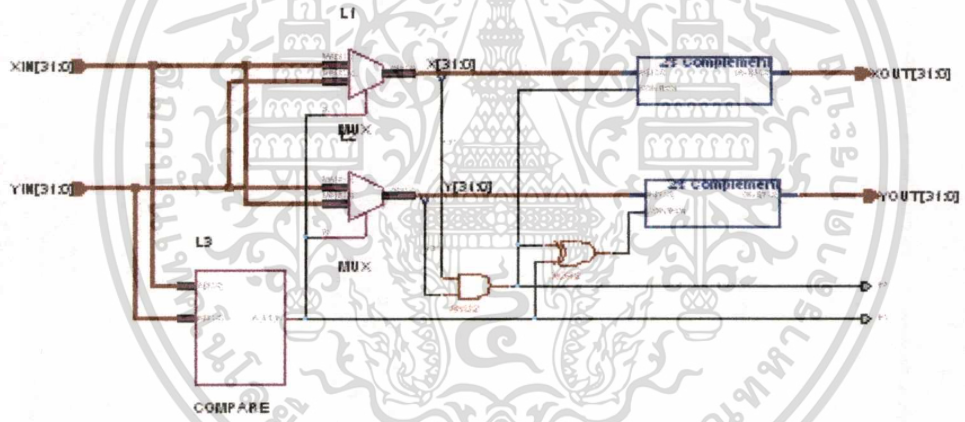
3.5.1 การทำงานของวงจร PreCORDIC

การทำงานของวงจร PreCORDIC มีค่าอินพุต เป็นค่าของเวคเตอร์ (x_i, y_i) ใดๆ และมีค่าเอาท์พุตเป็นค่า f1,f2 และค่า (x_o, y_o)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
If yi >= xi
  f1 = 1;   xm = yi; ym = -xi;
else
  f1 = 0; xm = xi; ym = yi;
end
if sign(xm) == -1
  f2 = 1; xo = -xm; yo = -ym;
else
  f2 = 0; xo = xm; yo = ym;
end
```

จากขั้นตอนการทำงานด้านบนนำมาสร้างเป็นวงจรได้ตามรูปที่ 3.3



รูปที่ 3.3 วงจร PreCORDIC

รูปที่ 3.3 เมื่อมีอินพุตเวคเตอร์ (x_i, y_i) เข้ามาในระบบ หลังจากนั้น ทำการเปรียบเทียบค่าทั้งสองเพื่อที่หาค่าของ f1 ซึ่งค่าของ f1 นี้ถูกส่งไปให้วงจร MUX เพื่อกำหนดการเปลี่ยนค่ากันระหว่างค่า y กับค่าของ x หลังจากนั้นนำเอาเครื่องหมายของทั้งสองมาคำนวณเพื่อกำหนดการทำงานของ วงจร 2's Comprement เพื่อหาค่าของ (x_o, y_o)

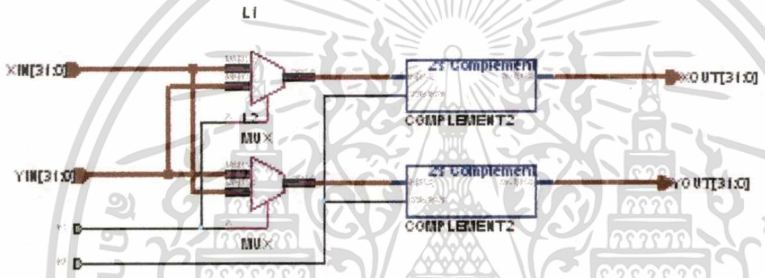
3.5.2 การทำงานของวงจร PostCORDIC

การทำงานของวงจร PostCORDIC มีค่าอินพุต เป็นค่าของเวคเตอร์ (x_i, y_i) และค่า f1,f2 มีค่าเอาท์พุตเป็นค่า (x_o, y_o)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
if f1 == 1
  xm = -yi; ym = xi;
else
  xm = xi; ym = yi;
end
if f2 == 1
  xo = -xm; yo = -ym;
else
  xo = xm; yo = ym;
end
```

จากขั้นตอนการทำงานด้านบนนำมาสร้างเป็นวงจรได้ตามรูปที่ 3.4



รูปที่ 3.4 วงจร PostCORDIC

รูปที่ 3.4 เมื่อมีอินพุต f1, f2 และ เวคเตอร์ (x_i, y_i) เข้ามาในระบบ f1 ถูกส่งสัญญาณให้กับวงจร MUX เพื่อกำหนดการเปลี่ยนค่ากันระหว่างค่า y กับค่าของ x ส่วนค่า f2 นำมากำหนดค่าการทำงานของ วงจร 2's Comprement เพื่อหาค่าของ (x_o, y_o)

3.6 สรุป

ในบทนี้ได้แสดงหลักการของคอร์ดิค เพื่อใช้ในการคำนวณค่าของ ฟังก์ชั่น Cos( ), Sin( ), Tan^{-1}( ) และ Sqrt( ) ซึ่งหลักการดังกล่าวประกอบไปด้วยวงจรพื้นฐาน Register, Adder and Subtactor, Multiplax, ROM และวงจร Barral shift ทำให้สามารถนำหลักการคอร์ดิคมาสร้างเป็นวงจรดิจิตอลได้ แต่อย่างไรก็ตามหลักการของคอร์ดิค ตอบสนองการทำงานในช่วง $\pi/2$ ถึง $-\pi/2$ ซึ่งยังไม่พอเพียงที่ใช้กับงานวิจัยนี้ จึงต้องใช้หลักการ PreCORDIC และ PostCORDIC เพื่อช่วยขยายช่วงการตอบสนองของหลักการคอร์ดิค ทำให้ตอบสนองค่าในช่วงที่กว้างขึ้นคืออยู่ในช่วง π ถึง $-\pi$ ซึ่งหลักการทั้ง คอร์ดิค, PreCORDIC และ PosCORDIC ได้นำมาประยุกต์ใช้ในบทต่อไป เพื่อสร้างส่วนประมวลผลของ จี-โค้ดโปรเซสเซอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4
การคำนวณค่าของหน่วยประมวลผลใน จี-โค้ดโปรเซสเซอร์

4.1 บทนำ

ในบทนี้กล่าวถึงหลักการใหม่ในการต่อวงจรร่วมกันของวงจรที่สร้างจาก คอร์ดิค อัลกอริทึม เพื่อใช้แก่สมการของหน่วยประมาณผล จี-โค้ดโปรเซสเซอร์ โดยนำวงจรที่สร้างจากคอร์ดิคอัลกอริทึมนี้มาต่อเรียงลำดับกับเพื่อแก้สมการให้เป็นไปตามรูปแบบการคำนวณที่ จี-โค้ดโปรเซสเซอร์ต้องการ ซึ่งวิธีการต่อวงจรเรียงลำดับกันนี้เป็นการประยุกต์จากหลักการของการต่อวงจรรวมกันของคอร์ดิคอัลกอริทึม ที่ได้ถูกใช้แก้ปัญหาสมการทางด้านอื่น [7-11] มาแล้ว

4.2 สมการหน่วยประมวลผลของ จี-โค้ดโปรเซสเซอร์

การทำงานในส่วนของหน่วยประมาณผลเพื่อทำงานชุดคำสั่ง G00, G01, G02 และ G03 แบ่งการทำงานเป็น 2ส่วน ส่วนแรก G00 และ G01 เป็นการคำนวณการเคลื่อนที่ในแบบเส้นตรง (Linear Interpolation) โดยมีอินพุตเป็นตำแหน่งปลายทางในการเคลื่อน อัตราเร็วและอัตราเร่ง และมีเอาท์พุทเป็นค่าของความเร็วและความเร่ง ในส่วนที่ 2 G02 และ G03 เป็นการคำนวณการเคลื่อนที่ในแบบเส้นโค้งส่วนของวงกลม (Circular Interpolation) โดยรับค่ามุม $\Delta\theta$ กับจุดศูนย์กลาง เพื่อคำนวณหาค่าตำแหน่งต่อไปในการประมาณค่า แล้วนำตำแหน่งที่คำนวณได้นี้ส่งให้การทำงานในส่วนแรกเพื่อหาค่า ความเร็วและความเร่ง [5][7]

4.2.1 การคำนวณทางคณิตศาสตร์

การคำนวณในการเคลื่อนที่เป็นเส้นตรง (Linear Interpolation) หรือคำสั่ง G00 และ G01 เป็นไปตามสมการที่ 4.1 และสมการที่ 4.2 กำหนดให้ Tmin = 2Vmax/Amax,Vmax เป็นอัตราเร็วในแนวการเคลื่อนที่, Vx ความเร็วในแนวแกน X, Vy ความเร็วในแนวแกน Y, Ax เป็นค่าความเร่งในแนวแกน X, Ay เป็นค่าความเร่งในแนวแกน Y, Amax เป็นค่าอัตราเร่ง, (X0,Y0) ตำแหน่งเริ่มต้นการเคลื่อนที่ และ (X1,Y1) ตำแหน่งปลายทาง

$$Vx=\frac{Vmax(X_1-X_0)}{\sqrt{(X_1-X_0)^2+(Y_1-Y_0)^2}} \quad Vy=\frac{Vmax(Y_1-Y_0)}{\sqrt{(X_1-X_0)^2+(Y_1-Y_0)^2}}$$

$$Ax=\frac{0.5\cdot Amax(X_1-X_0)}{\sqrt{(X_1-X_0)^2+(Y_1-Y_0)^2}} \quad Ay=\frac{0.5\cdot Amax(Y_1-Y_0)}{\sqrt{(X_1-X_0)^2+(Y_1-Y_0)^2}} \qquad \text{.....(4.1)}$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เขียนสมการที่ 4.1 ให้อยู่ในรูป ฟังก์ชั่น Sin( ), Cos( ) ได้ในสมการที่ 4.2

$$\theta=\tan^{-1}\frac{(y_1-y_0)}{(x_1-x_0)}$$

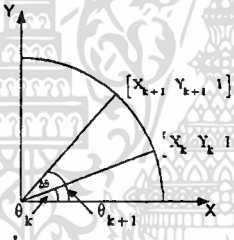
$$Vx=Vmax\cdot Cos(\theta \qquad Vy=Vmax\cdot Sin(\theta$$

$$Ax=0.5\ Amax\cdot Cos(\theta \qquad Ay=0.5\ Amax\cdot Sin(\theta) \qquad \text{..... (4.2)}$$

การคำนวณในการเคลื่อนที่เป็นเส้นโค้งส่วนของวงกลม (Circular Interpolation) หรือคำสั่ง G02 และ G03 เขียนสมการในการคำนวณหาจุดต่างๆด้วย สมการที่ 4.3 กำหนดให้ (I,J) เป็นจุดศูนย์กลางของวงกลมที่ใช้ในการประมาณค่า, $\Delta\theta$ เป็นมุมที่ใช้ในการประมาณค่าวงกลมด้วยเส้นตรง และ (Xk, Yk) เป็นจุดใดๆบนส่วนโค้ง

$$\begin{bmatrix} X_{k-1} & Y_{k-1} & 1 \end{bmatrix}=\begin{bmatrix} X_k & Y_k & 1 \end{bmatrix}\cdot\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -I & -J & 1 \end{bmatrix}\cdot\begin{bmatrix} \cos\Delta\theta & -\sin\Delta\theta & 0 \\ \sin\Delta\theta & \cos\Delta\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}\cdot\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ I & J & 1 \end{bmatrix}$$

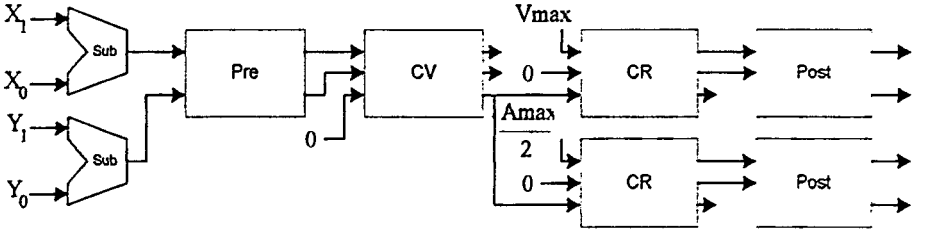
$$\theta_{k-1}=\theta_k-\Delta\theta \qquad \text{..... (4.3)}$$



รูปที่ 4.1 ค่าต่างๆตามสมการที่ 4.3 เพื่อทำการประมาณเส้นโค้งด้วยเส้นตรง

จากสมการที่ 4.2 และ 4.3 งานวิจัยนี้ได้เลือกใช้หลักการของ คอร์ดิคที่ได้กล่าวถึงในบทที่ 3 [8-11] เพื่อทำการคำนวณหาค่า เนื่องจาก คอร์ดิคใช้วงจรในการทำงานเป็นแค่การบวกค่า(Adder) กับการเลื่อนค่า(Shift) ทำให้ง่ายต่อการออกแบบและพัฒนา

4.2.2 การคำนวณด้วย คอร์ดิคอัลกอริธึม

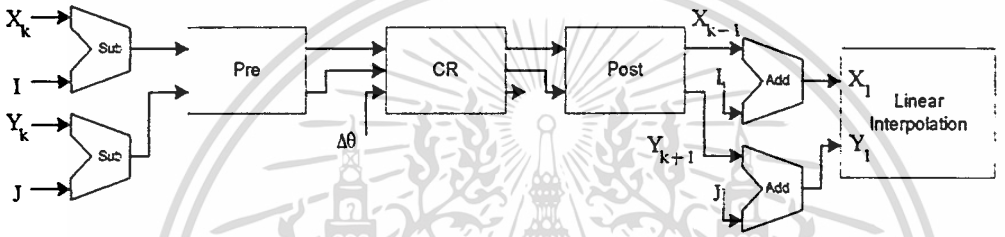


รูปที่ 4.2 หลักการแก้สมการที่ 4.2 ด้วย คอร์ดิคอัลกอริธึม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากรูปที่ 4.2 เป็นการแสดง อัลกอริธึม เพื่อแก้สมการที่ 4.2 โดยมีอินพุท(X0,Y0) เป็นค่าของจุดเริ่มต้น, (X1,Y1) เป็นค่าของจุดปลายทาง เริ่มจากการหาค่าความแตกต่าง วงจรลบเลข ปรับค่าที่ได้จากการลบให้อยู่ในช่วง $-\pi/2$ ถึง $\pi/2$ ด้วย Pre(Pre-CORDIC) เพื่อให้อยู่ในช่วงที่ CV(CORDIC Vectoring) ทำงานได้ ผลที่ได้จาก CV จะเป็นค่ามุม θ ตามรูป เมื่อนำ θ ส่งให้ CR(CORDIC Rotation) เพื่อสร้างค่า Sin(θ), Cos(θ) มาคูณกับ Vmax และ Amax /2 เพื่อหาค่าความเร็วและความเร่ง ทำการปรับค่าที่ได้ให้อยู่ในตำแหน่งที่ถูกต้องด้วย Post(Post-CORDIC)

การนำหลักการ คอร์ดิคมาประมวลผล สมการที่ 4.3 ได้ดังโครงสร้างรูปที่ 4.3



รูปที่ 4.3 การแก้สมการที่ 4.3 ด้วย คอร์ดิคอัลกอริธึม

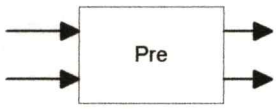
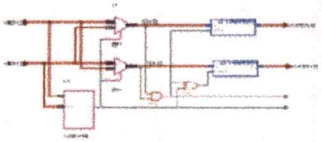
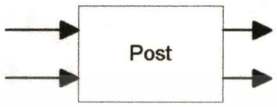
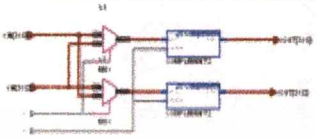
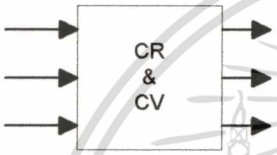
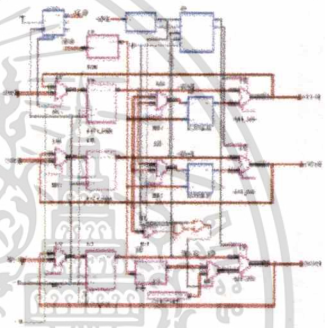
รูปที่ 4.3 เป็นการแสดง อัลกอริธึมเพื่อแก้สมการที่ 4.3 โดยใส่ค่าอินพุท เป็น (Xk,Yk) กับ (I,J) เข้ามาในระบบให้ Pre ปรับค่าให้อยู่ในช่วงที่กำหนด ทำการ CR และ Post หมุนตำแหน่ง (Xk,Yk) ให้เป็น (Xk+1,Yk+1) แล้วทำการรวมกับค่า (I,J) หลังจากนั้นส่งค่านี้ไปยัง (Xi,Yi) ของการคำนวณการเคลื่อนที่แบบเส้นตรงตามสมการที่ 4.2 เพื่อทำการหาค่า ความเร็วและความเร่ง

4.3 การสร้างวงจรจากสมการหน่วยประมวลผล

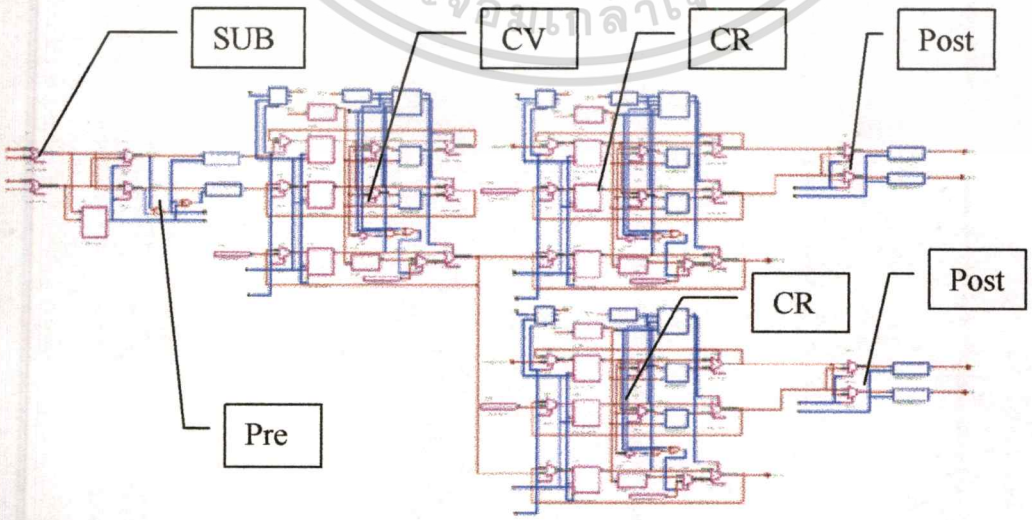
ในการสร้างวงจรจากหลักการที่แสดงตามรูปที่ 4.2 และรูปที่ 4.3 เริ่มจากการแทนส่วนของ Pre, Post, CR และ CV ด้วยวงจรดิจิตอลที่แสดงในบทที่ 3 ซึ่งได้สรุปผลการแทนเป็นไปตามตารางที่ 4.1 หลักจากการแทนหลักการในส่วนต่างๆด้วยวงจรดิจิตอลแล้ว ทำการเชื่อมต่อวงจรในส่วนต่างๆให้เป็นไปตามลำดับที่แสดงในหลักการของการแก้สมการหน่วยประมวลผล จี-โค้ด โปรเซสเซอร์ ผลการแทนวงจรแสดงดังรูปที่ 4.4 และรูปที่ 4.5

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ตารางที่ 4.1 การแทนส่วนต่างๆของหลักการแก้สมการหน่วยประมวลผลด้วยวงจรดิจิตอล

Block Diagram	วงจรดิจิตอล
	 จากรูปที่ 3.3
	 จากรูปที่ 3.4
	 จากรูปที่ 3.2

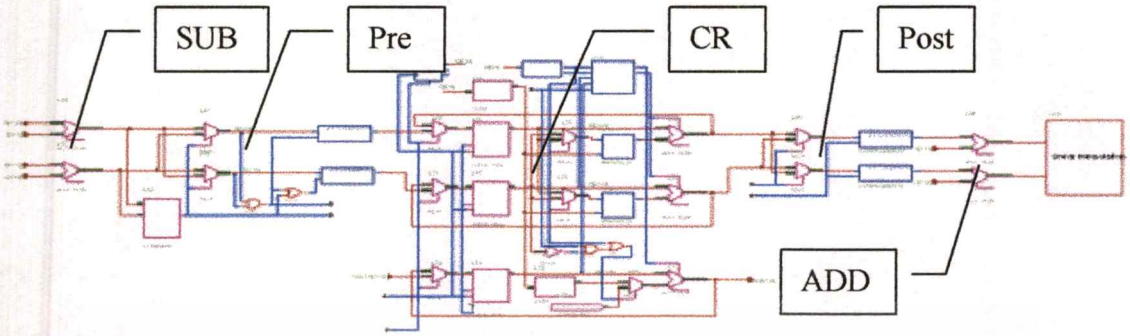
เมื่อพิจารณาหลักการคำนวณการเคลื่อนที่เป็นเส้นตรงตามรูปที่ 4.2 แปรงเป็นวงจรดิจิตอล แล้ว ได้แสดงดังรูปที่ 4.4



รูปที่ 4.4 วงจรดิจิตอลตามหลักการคำนวณการเคลื่อนที่เป็นเส้นตรง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

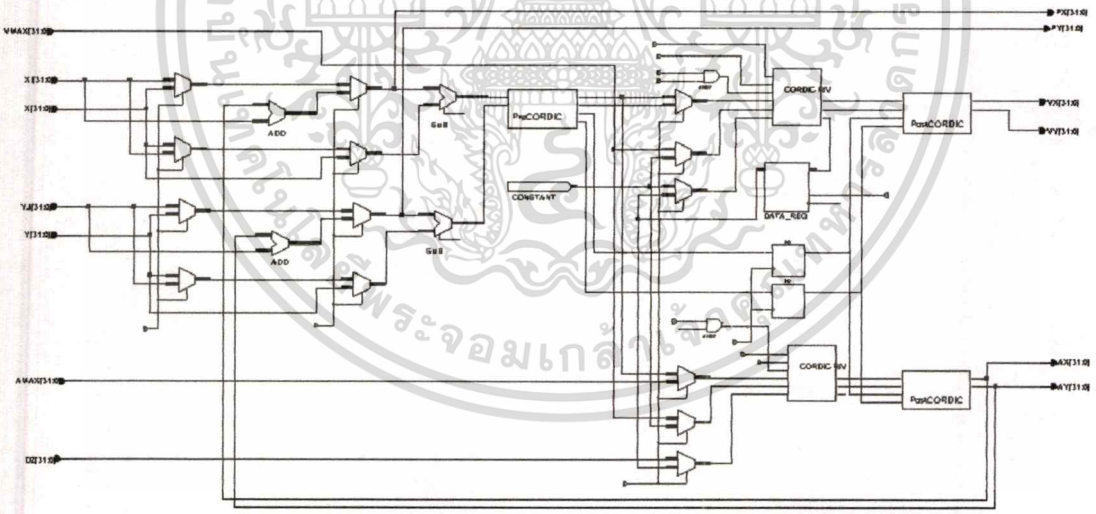
เมื่อพิจารณาหลักการคำนวณการเคลื่อนที่เป็นเส้นโค้งส่วนของวงกลมตามรูปที่ 4.3 แปรงเป็นวงจรดิจิตอลแล้วได้แสดงดังรูปที่ 4.5



รูปที่ 4.5 วงจรดิจิตอลตามหลักการคำนวณการเคลื่อนที่เป็นเส้นโค้งส่วนของวงกลม

4.4 โครงสร้างหน่วยประมาณผลที่ใช้หลักการ คอร์ดิคอัลกอริธึม

จากหัวข้อ 4.3 สามารถปรับปรุงโดยการรวมส่วนที่ทำงานซ้ำกันของ การคำนวณการเคลื่อนที่แบบเส้นตรง(Linear Interpolation) และ การคำนวณการเคลื่อนที่เป็นเส้นโค้งส่วนของวงกลม(Circular Interpolation) สรุปโครงสร้างได้ตามรูปที่ 4.4



รูปที่ 4.6 วงจรหน่วยประมาณผลของ จีโค้ด โปรเซสเซอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.5 สรุป

ในบทนี้ได้กล่าวถึงสมการที่ใช้ในการคำนวณของชุดคำสั่ง G00, G01, G02 และ G03 เพื่อใช้สร้างหน่วยประมวลผลของ จี-โค้ดโปรเซสเซอร์ โดยได้นำเสนอหลักการที่ใช้ คอร์ดิคอัลกอริธึม มาเรียงลำดับ เพื่อแก่สมการของหน่วยประมาณผล จี-โค้ดโปรเซสเซอร์ ซึ่งได้แสดงดังรูปที่ 4.2 และ รูปที่ 4.3 เป็นการคำนวณการเคลื่อนที่เป็นเส้นตรง และการคำนวณการเคลื่อนที่เป็นเส้นโค้งส่วนของวงกลม ในบทนี้ยังได้แสดงวิธีการเปลี่ยนการเรียงลำดับของหลักการคอร์ดิคอัลกอริธึม ที่ใช้แก้สมการของหน่วยประมาณผล จี-โค้ดโปรเซสเซอร์ มาให้อยู่ในรูปแบบของวงจรดิจิตอล โดยการแทนวงจรดิจิตอลลงในส่วนต่างๆตามตารางที่ 4.1 และนำส่วนต่างๆมาต่อร่วมกันตามลำดับซึ่งได้วงจรแสดงอยู่ในรูปที่ 4.4 และรูปที่ 4.5 หลังจากนั้นได้ปรับปรุงวงจรโดยลดความซ้ำซ้อนและแสดงในรูปที่ 4.6 ในบทต่อไปเป็นการวิเคราะห์ ผลความถูกต้อง และความเร็วในการทำงาน

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

บทที่ 5
อุปกรณ์และการทดลอง

5.1 บทนำ

ในบทนี้แสดงการทดสอบ จี-โค้ดโปรเซสเซอร์ โดยการเน้นที่ส่วนของ หน่วยประมวลผล เนื่องจากโครงสร้างการทำงานของส่วนนี้เป็นส่วนที่แตกต่างจาก โครงสร้างการทำงานของโปรเซสเซอร์อื่นๆมากที่สุด รวมถึงในส่วนนี้ยังเป็นตัวชี้วัดในการแสดงถึงความเร็วในการประมวลผล เนื่องจาก เวลาที่ใช้ในการทำงานส่วนใหญ่ของ จี-โค้ดโปรเซสเซอร์ ขึ้นอยู่กับการทำงานในส่วนนี้ โดยมีการแบ่งการทดสอบผลออกเป็น สองรูปแบบ คือ การทดสอบความถูกต้องของการคำนวณโดยการจำลองทางคณิตศาสตร์ด้วยโปรแกรม MathLab และการทดลองที่สอง เป็นการทดสอบทางด้านความเร็วในการทำงานโดยวัดจำนวนสัญญาณนาฬิกาที่ใช้ในการคำนวณ จากระบบที่ออกแบบ กับค่าที่วัดผลได้จากระบบจริง [7] ที่ทำงานในงานเดียวกัน

5.2 การทดสอบความถูกต้องของการคำนวณทางคณิตศาสตร์

ในขั้นตอนแรกในการออกแบบหน่วยประมาณผล ใน จี-โค้ดโปรเซสเซอร์ผู้วิจัยได้ใช้เครื่องมือทางคณิตศาสตร์คือโปรแกรม MathLab จำลองรูปแบบทางคณิตศาสตร์ ออกเป็นสองรูปแบบเพื่อผลทางการเปลี่ยนเทียบ โดยรูปแบบทางคณิตศาสตร์แรกเป็นสมการที่สร้างขึ้นตามหลักการของ คอร์ดิคอัลกอริธึม ที่ได้กล่าวมาในบทที่ 3 และ บทที่ 4 และในส่วนที่สองเป็นสมการของหน่วยประมาณผลที่สร้างจากสมการพื้นฐาน ซึ่งการสร้างสมการทั้งสองนี้ขึ้นมาเป็นการบ่งบอกความถูกต้องของหลักการในการคำนวณของหน่วยประมาณผล

5.2.1 การจำลองทางคณิตศาสตร์ตามหลักการ คอร์ดิคอัลกอริธึม

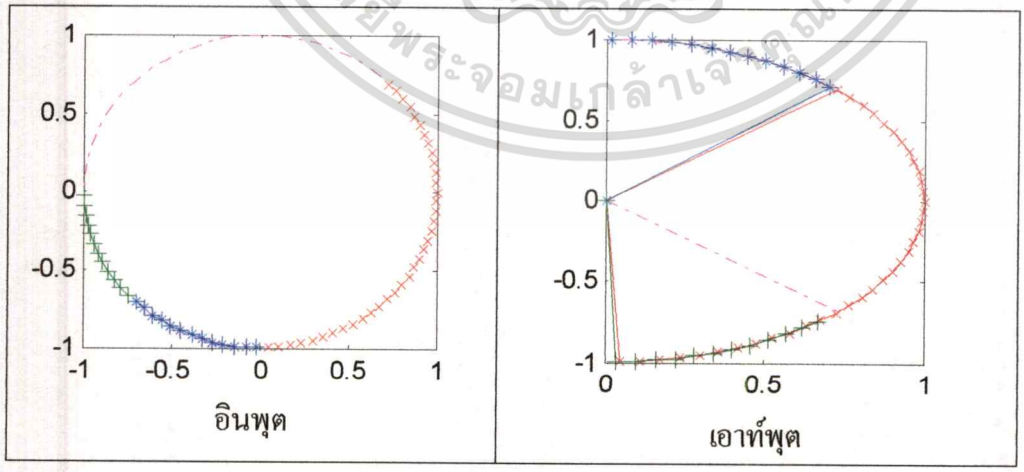
ในการจำลองทางคณิตศาสตร์ตามหลักการ CORDIC อัลกอริธึม เริ่มจากการแบ่งการจำลองออกเป็นส่วนย่อยๆ คือมีส่วนของ Pre-CORDIC (Pre-Processing CORDIC), CORDIC-R (CORDIC Rotation, CR), CORDIC-V (CORDIC Vectoring, CV) และ ส่วนของ Post-CORDIC (Post-Processing CORDIC) แล้วทำการทดสอบความถูกต้อง หลังจากที่ทำการจำลองและทดสอบในแต่ละส่วนย่อยๆ แล้ว เริ่มทำการประกอบส่วนย่อยๆ ให้ได้ตามระบบที่ได้นำเสนอในบทที่ 4 ทำการทดสอบผลอีกครั้ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

Pre-CORDIC มีคำสั่งในการจำลองการทำงานดังนี้

```
function [xo,yo,f1,f2]=PreCordic(xi,yi)
   xm =0; ym =0;
   if yi >= xi
      f1 = 1; xm = yi; ym = -xi;
   else
      f1 = 0; xm = xi; ym = yi;
   end
   if sign(xm) == -1
      f2 = 1; xo = -xm; yo = -ym;
   else
      f2 = 0; xo = xm; yo = ym;
   end
end;
```

ทำการทดสอบการทำงานโดยใส่ค่า $Xi = \cos(\theta)$ และ $Yi = \sin(\theta)$ ผลการทดลองทางด้านเอาท์พุต ค่าที่ได้อยู่ในช่วง $-\frac{\pi}{2}$ ถึง $\frac{\pi}{2}$ โดยมีค่า f1 และ f2 เป็นตัวเก็บข้อมูล ที่ใช้เพื่อเปลี่ยนค่าทางด้านเอาท์พุต กลับมาเป็นข้อมูลทางด้านอินพุตได้อย่างถูกต้องหลังจากที่ทำการคำนวณเสร็จแล้ว โดยใช้ประกอบกับการทำงานในส่วนของ Post-CORDIC แสดงดังรูปที่ 5.1



รูปที่ 5.1 การทดสอบการทำงานของ Pre-CORDIC

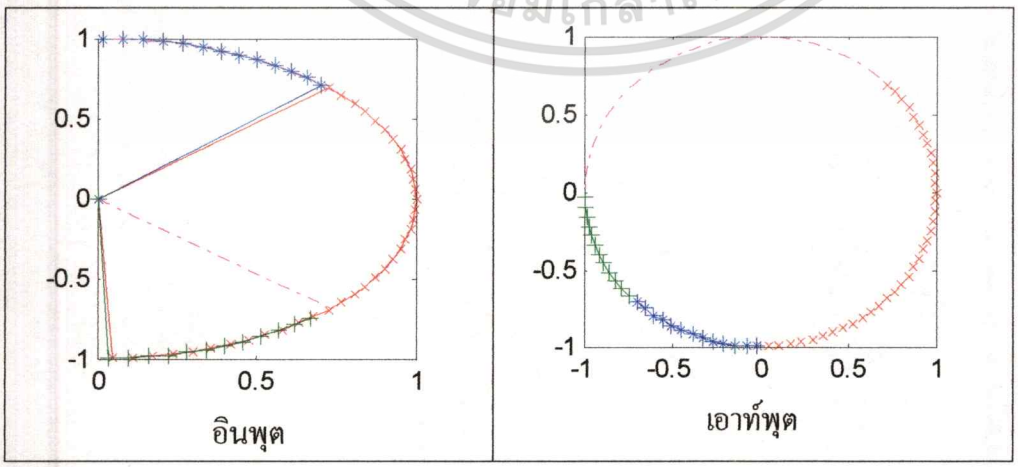
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 5.1 ทางด้านอินพุตเป็นที่ใช้ทดสอบหลักการ Pre-CORDIC โดยมีการกำหนดสีเพื่อแสดงให้เห็นถึงค่าที่แตกต่างเพื่อช่วยในการแสดงความสัมพันธ์ระหว่างอินพุตและเอาท์พุต โดยไม่มีผลในการคำนวณทางคณิตศาสตร์ ส่วนทางด้านเอาท์พุตเป็นค่าของ Xo และ Yo และใช้สัญญาลักษณ์ที่แตกต่างกันของรูป แสดงถึงค่าที่แตกต่างกันของ f1 และ f2

Post-CORDIC มีคำสั่งในการจำลองการทำงานดังนี้

```
function [xo,yo]=PosCordic(xi,yi,f1,f2)
  xm =0; ym =0;
  if f1 == 1;
    xm = -yi; ym = xi;
  else
    xm = xi; ym = yi;
  end
  if f2 == 1
    xo = -xm; yo = -ym;
  else
    xo = xm; yo = ym;
  end
end;
```

ทำการทดลองการทำงานโดยการนำเอาท์พุตของการทดลอง Pre-CORDIC ทั้งค่า Xo, Yo, f1 และ f2 เป็นอินพุตให้การทดลองของ Post-CORDIC เพื่อเป็นการเปลี่ยนค่าข้อมูล Xo, Yo กลับเป็นค่า Xi, Yi โดยการใช้ข้อมูล f1 และ f2 แสดงดังรูปที่ 5.2



รูปที่ 5.2 การทดสอบการทำงานของ Post-CORDIC

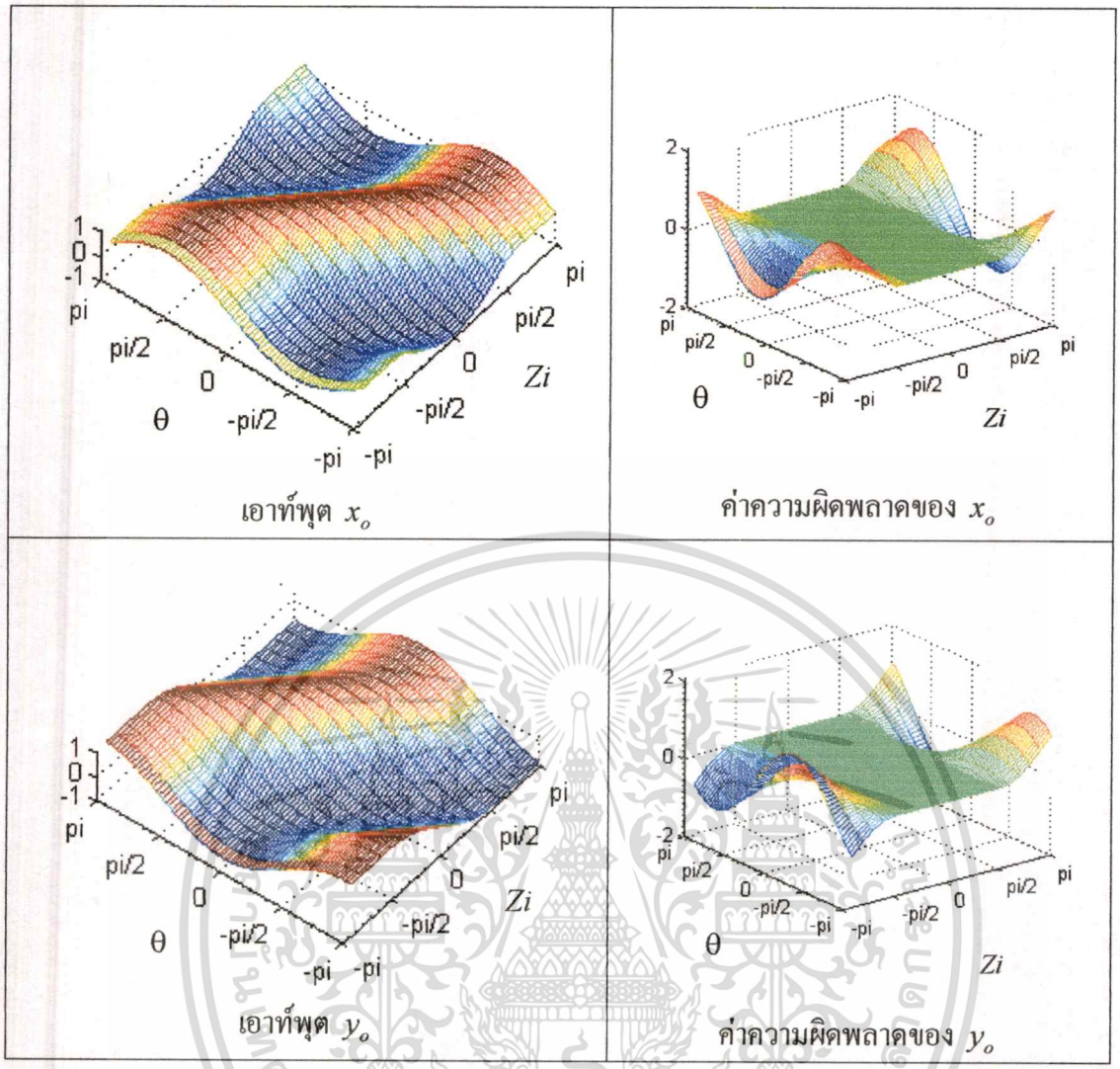
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

CORDIC-R มีคำสั่งในการจำลองการทำงานดังนี้

```
function [xo.yo,zo]= CordicR(xi,yi,zi);
   s=[1 2 2 2 2 3 4 5 6 6 7 8 9 10 11 12 13 13 14 15 16 17 18 19 20 21 22 23 24];
   l=30; r=linspace(0,0,l); x=linspace(0,0,l); y=linspace(0,0,l); z=linspace(0,0,l);
   x(1)=xi; y(1)=yi; z(1)=zi;
   for i=1:l-1
      c(i)=atan(2^(-s(i))); r(i)=sign(z(i));
      x(i+1)=x(i)-r(i)*2^(-s(i))*y(i);
      y(i+1)=y(i)+r(i)*2^(-s(i))*x(i);
      z(i+1)=z(i)-r(i)*c(i);
  end;
  xo=CordicScal(x(l)); yo=CordicScal(y(l)); zo=z(l);
end;
function u= CordicScal(a);
   s=[-2 4 -5 6 0 17 -20];
   l=8; r=linspace(0,0,l); x=linspace(0,0,l); x(1)=a;
   for i=1:l-1
      r(i)=sign(s(i));
      x(i+1)=x(i)+r(i)*2^(-abs(s(i)))*x(i);
   end;
   u=x(l);
end;
```

โปรแกรมด้านบนเป็นโปรแกรมจำลองการคำนวณตามหลักการ CORDIC Rotation เป็นไปตามหลักการในบทที่ 3 โดยค่าเอาท์พุตของหลักการนี้ $x_o = x_i\cos(z_i) - y_i\sin(z_i)$, $y_o = y_i\cos(z_i) + x_i\sin(z_i)$, $z_o = 0$ เพื่อทดสอบความถูกต้องของโปรแกรมจำลองการทำงานตามหลักการ CORDIC Rotation กำหนดให้ $x_i = \cos(\theta)$, $y_i = \sin(\theta)$ โดยกำหนดให้ $-\pi \le \theta \le \pi$ และ $-\pi \le z_i \le \pi$ เป็นอินพุตของโปรแกรม เอาท์พุตแสดงดังรูป 5.3

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า



รูปที่ 5.3 แสดงเอาท์พุตและค่าที่เกิดจากการคำนวณผิดพลาดของ อัลกอริธึม CORDIC-R

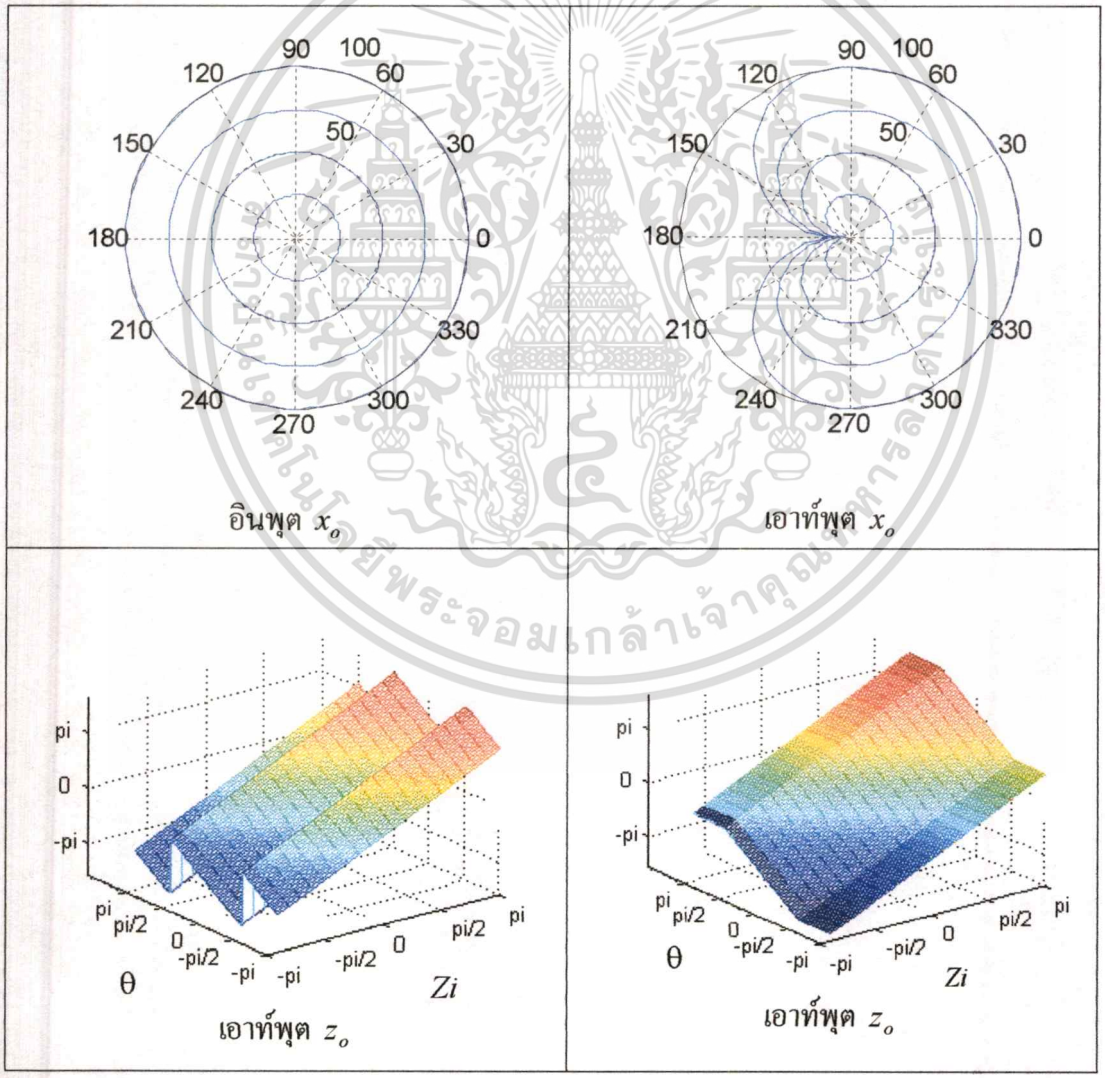
CORDIC-V มีคำสั่งในการจำลองการทำงานดังนี้

```
function [xo,yo,zo]= CordicV(xi,yi,zi);
  s=[1 2 2 2 3 4 5 6 6 7 8 9 10 11 12 13 13 14 15 16 17 18 19 20 21 22 23 24];
  l=30; r=linspace(0,0,l); x=linspace(0,0,l); y=linspace(0,0,l); z=linspace(0,0,l);
  x(1)=xi; y(1)=yi; z(1)=zi;
  for i=1:l-1
    c(i)=atan(2^(-s(i))); r(i)=-sign(y(i));
    x(i+1)=x(i)-r(i)*2^(-s(i))*y(i);
    y(i+1)=y(i)+r(i)*2^(-s(i))*x(i);
    z(i+1)=z(i)-r(i)*c(i);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  end;
  xo=CordicScal(x(l)); yo=CordicScal(y(l)); zo=z(l);
end;
```

โปรแกรมด้านบนเป็นโปรแกรมจำลองการคำนวณตามหลักการ CORDIC Vectoring เป็นไปตามหลักการในบทที่ 3 โดยค่าเอาท์พุตของหลักการนี้ $x_o = \sqrt{(x_i^2 + y_i^2)}$, $y_o = 0$, $z_o = z_i + \tan^{-1}(y_i / x_i)$ เพื่อทดสอบความถูกต้องของโปรแกรมจำลองการทำงานตามหลักการ CORDIC Vectoring โดยการทดสอบค่า x_o กำหนดให้ $x_i = r \cdot \cos(\theta)$, $y_i = r \cdot \sin(\theta)$ โดยกำหนดให้ $-\pi \le \theta \le \pi$ และ $r = \{0,20,40,60,80,100\}$ เป็นอินพุตของโปรแกรม เอาท์พุตแสดงดังรูป 5.4 บน ส่วนการทดลองค่า z_o กำหนดให้ $x_i = \cos(\theta)$, $y_i = \sin(\theta)$ โดย $-\pi \le \theta \le \pi$ และ $-\pi \le z_i \le \pi$



รูปที่ 5.4 แสดงเอาท์พุตและค่าที่เกิดจากการคำนวณผิดพลาดของ อัลกอริธึม CORDIC-V

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.2.2 การจำลองทางคณิตศาสตร์หน่วยประมวลผล จี-โค้ดโปรเซสเซอร์

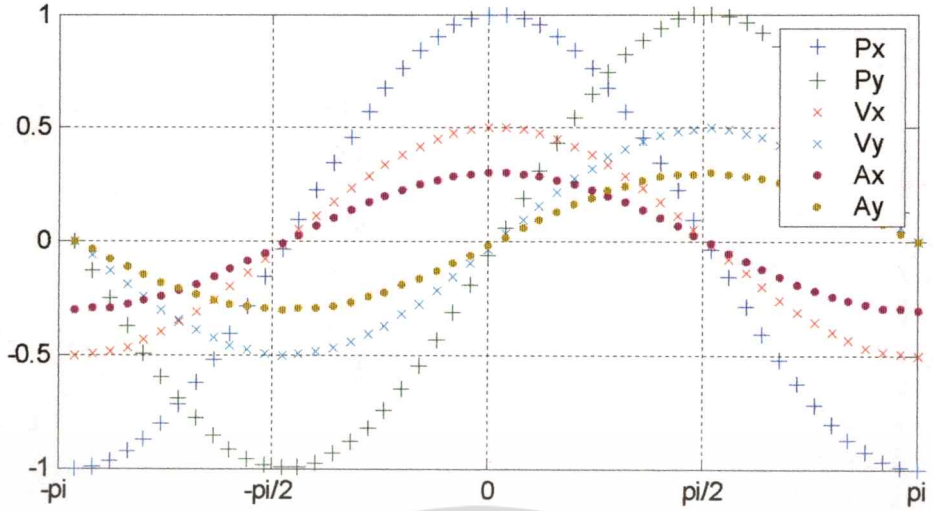
ในการจำลองทางคณิตศาสตร์ของหน่วยประมาณผลของ จี-โค้ดโปรเซสเซอร์ แบ่งการทดสอบออกเป็นสองการทดลอง โดยการการทดลองแรกเป็นการทดสอบการทำงานของคำสั่ง G00 และ G01 และในส่วนที่สองเป็นการทดสอบการทำงานของ G02 และ G03

G00 G01 มีคำสั่งในการจำลองการทำงานดังนี้

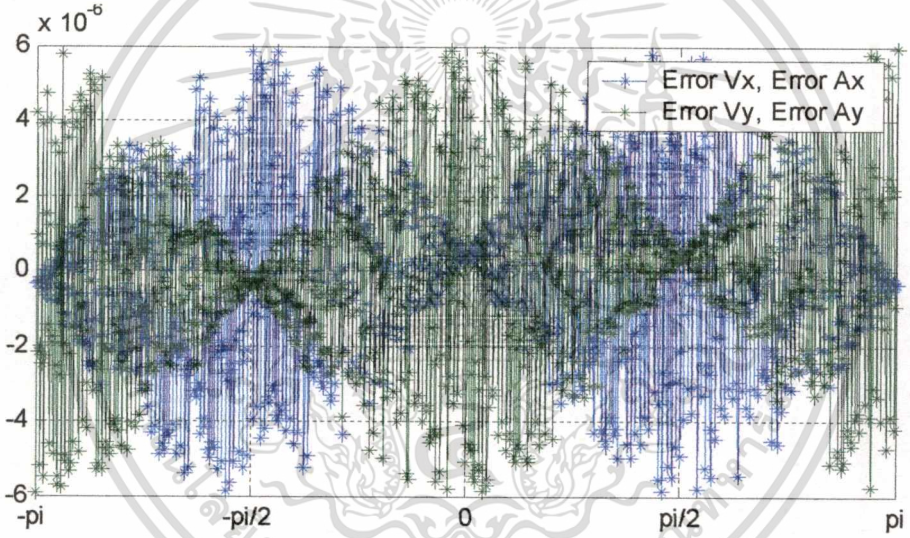
```
function [Px,Py,Vx,Vy,Ax,Ay]=G00G01(X1,Y1,X0,Y0,V,A);
   [Tx,Ty,F1,F2]=PreCordic(X1-X0,Y1-Y0);
   [Tx,Ty,Z]=CordicV(Tx,Ty,0);
   Tz=Z;
   [Vx,Vy,Tz]=CordicR(V,0,Tz);
   [Vx,Vy]=PosCordic(Vx,Vy,F1,F2);
   Tz=Z;
   [Ax,Ay,Tz]=CordicR(A,0,Tz);
   [Ax,Ay]=PosCordic(Ax,Ay,F1,F2);
   Px=X1; Py=Y1;
end;
```

โปรแกรมด้านบนเป็นโปรแกรมจำลองการคำนวณในการเคลื่อนที่เป็นเส้นตรง (Linear Interpolation) เป็นไปตามหลักการในบทที่ 4 โดยค่าเอาท์พุตมีค่าดังต่อไปนี้ $Px = X1$, $Py = Y1$, $Vx = V \cdot \cos\left(\tan^{-1}\left(\frac{Y1-Y0}{X1-X0}\right)\right)$, $Vy = V \cdot \sin\left(\tan^{-1}\left(\frac{Y1-Y0}{X1-X0}\right)\right)$, $Ax = A \cdot \cos\left(\tan^{-1}\left(\frac{Y1-Y0}{X1-X0}\right)\right)$, $Ay = A \cdot \sin\left(\tan^{-1}\left(\frac{Y1-Y0}{X1-X0}\right)\right)$ เพื่อทดสอบความถูกต้องของโปรแกรมจำลองการทำงาน กำหนดให้ $X1 = \cos(\theta)$, $Y1 = \sin(\theta)$, $X0 = 0$, $Y0 = 0$, $V = 0.5$ และ $A = 0.3$ ผลการทดสอบแสดงดังรูปที่ 5.5 และรูปที่ 5.6

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า



รูปที่ 5.5 แสดงค่าเอาท์พุตของการจำลองการทำงาน G00G01



รูปที่ 5.6 แสดงค่าความผิดพลาดของการจำรองการทำงานคิดเป็นร้อยละ

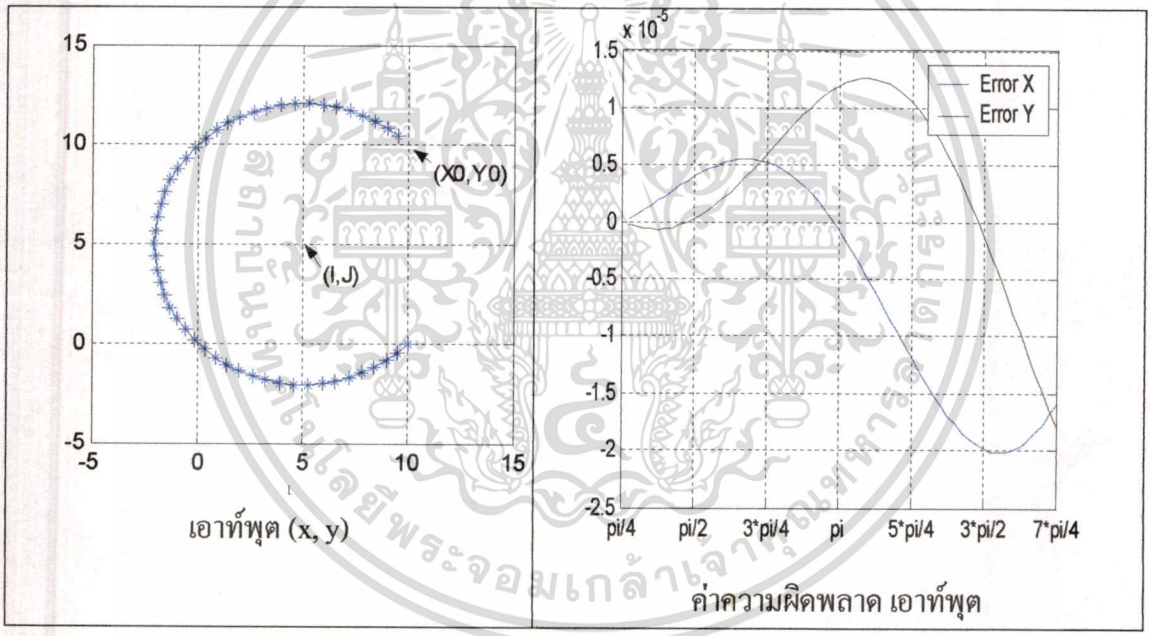
G02 G03 มีคำสั่งในการจำลองการทำงานดังนี้

```
function [Px,Py,Vx,Vy,Ax,Ay]=G02(I,J,X0,Y0,Sthe,n,V,A);
   Vx=1:n; Vy=1:n; Ax=1:n; Ay=1:n;
   x=1:n+1; y=1:n+1;
   Dz=Sthe/n;     %G03 used Dz=-Sthe/n;
   x(1)=X0; y(1)=Y0;
   for i=1:n
     [Vx(i),Vy(i),F1,F2]=PreCordic(x(i)-I,y(i)-J);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    [Ax(i),Ay(i),Z]=CordicR(Vx(i),Vy(i),Dz);
    [x(i+1),y(i+1)]=PosCordic(Ax(i),Ay(i),F1,F2);
    x(i+1)=x(i+1)+I;y(i+1)=y(i+1)+J;
    [Px(i),Py(i),Vx(i),Vy(i),Ax(i),Ay(i)]=G01(x(i+1),y(i+1),i,j,V,A);
  end
end;
```

โปรแกรมด้านบนเป็นโปรแกรมจำลองการคำนวณในการเคลื่อนที่ในแบบเส้นโค้งส่วนของวงกลม (Circular Interpolation) เป็นไปตามหลักการในบทที่ 4 ทดสอบการทำงานโดยการกำหนดให้ I=5, J=10, X0=10, Y0=10, Sthe= 3*Pi/2,n= 50, V และ V เป็นค่าคงที่นำค่า Px และ Py มาแสดงผลดังรูปที่ 5.7

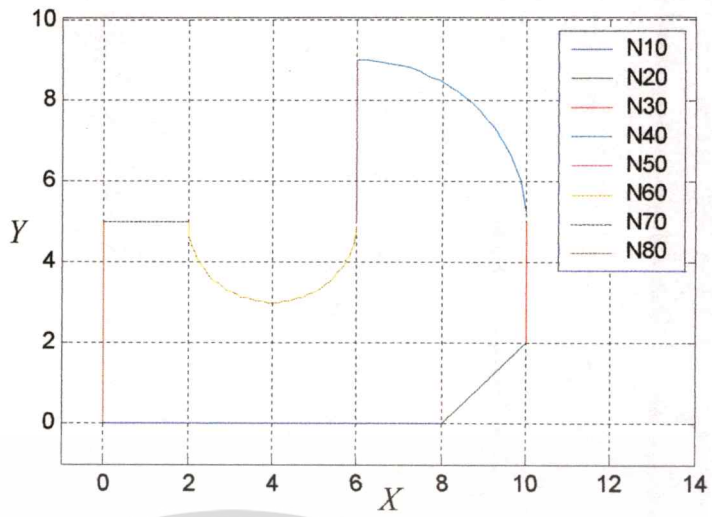


รูปที่ 5.7 แสดงค่าคู่อันดับ(Px,Py) และค่าผิดพลาด ที่ได้จากการทดลองโปรแกรม G02

5.2.3 การจำลองทางคณิตศาสตร์ ตามรูปแบบการใช้งานจริง

ในการทดลองนี้เป็นการจำลองการชุดคำสั่ง G-Code เพื่อใช้เคลื่อนหัวเครื่องจักรในแนวระนาบ (x, y) ตามรูปที่ 5.8

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

N10 G00 X8 Y0 N20 G00 X10 Y2 N30 G00 X10 Y5 N40 G02 I6 J5 X6 Y9 N50 G01 X6 Y5 N60 G03 I6 J5 X6 Y5 N70 G01 X0 Y5 N80 G01 X0 Y0	

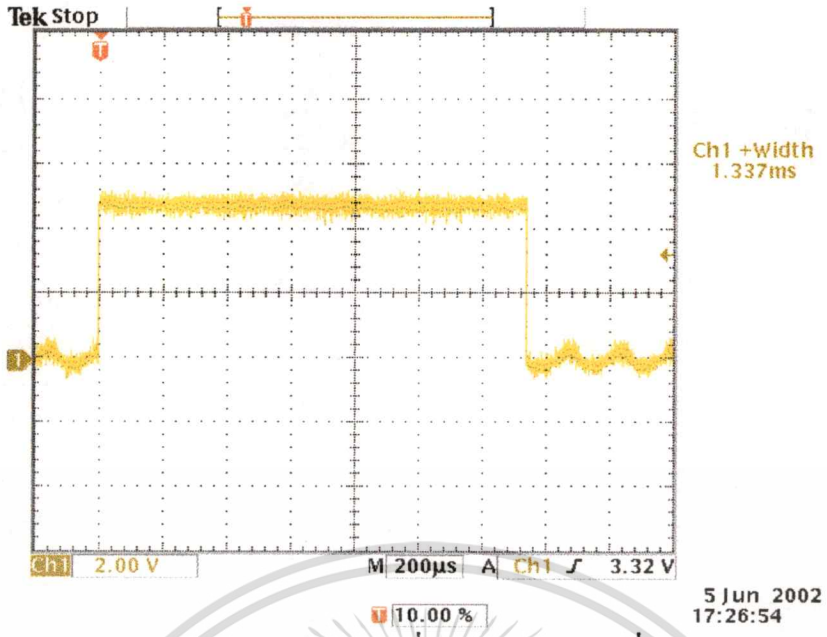
รูปที่ 5.8 แสดงเส้นทางการเคลื่อนที่ของหัวเครื่องจักรตามแนวระนาบ x,y

จากรูปที่ 5.8 ทางด้านซ้าย เป็นคำสั่ง G-Code โดยกำหนดให้สถานะปัจจุบันของเครื่องจักรอยู่ที่ตำแหน่ง (0,0) คำสั่งแรก N10 G01 X8 Y0 เป็นคำสั่งให้หัวเครื่องจักรเคลื่อนที่เป็นเส้นตรงไปยังตำแหน่ง (8,0) เช่นเดียวกับคำสั่งที่ N20, N30, N50, N70 และ N80 ทำงานเหมือน คำสั่ง N10 แตกต่างที่ตำแหน่งปลายทางการเคลื่อนที่ ส่วนคำสั่ง N40 และ N60 เป็นคำสั่งเคลื่อนที่ในรูปแบบส่วนของวงกลม โดยมีจุดศูนย์กลางอยู่ที่ตำแหน่งคู่อันดับ (I,J) และมีจุดปลายทางอยู่ที่คู่อันดับ (X,Y)

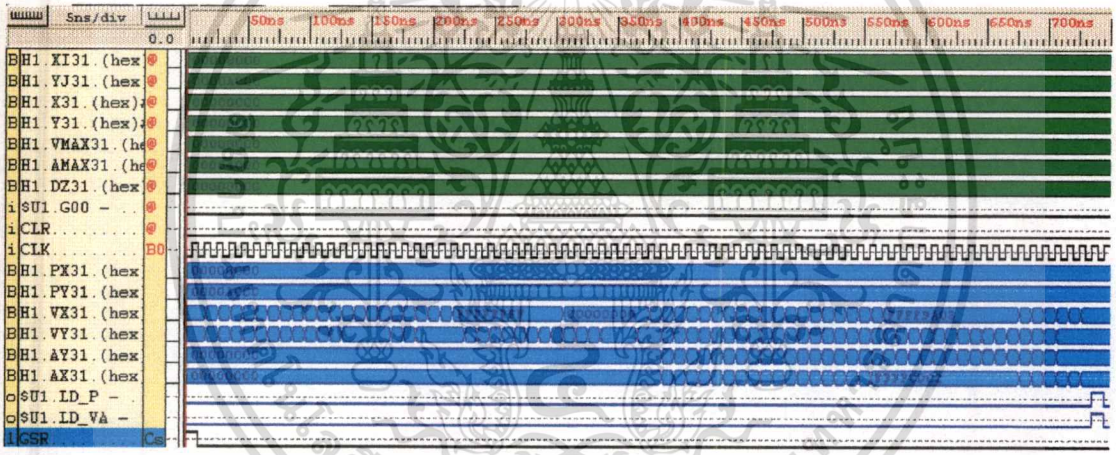
5.3 การทดสอบทางด้านความเร็วในการทำงาน

ในการทดสอบนี้ใช้ระบบที่มีอยู่เดิม [6] ซึ่งมีตัวประมวลผลเป็น MCS-51 ความถี่ระบบ 8 MHz ทดสอบโดยให้คำนวณหาค่าในการเคลื่อนที่เป็นเส้นตรง (Linear Interpolation) เป็นไปตามหลักการคำนวณในบทที่ 4 โดยมีโปรแกรมชุดคำสั่งภาษา C ตามภาคผนวก ก. แล้ววัดสัญญาณ ที่ขาตัวประมวลผล ซึ่งใช้เวลาในการประมาณผล 1.337 *ms* แสดงดังรูปที่ 5.9 เมื่อคำนวณเป็นจำนวนสัญญาณที่ใช้ในการทำงานได้ 10696 สัญญาณ เมื่อเปรียบเทียบกับรูปที่ 5.10 การจำลองจากโปรแกรม Xilinx Foundation F1.5 ซึ่งเป็นโปรแกรมใช้ในการออกแบบ FPGA ใช้สัญญาณนาฬิกาจำนวน 74 สัญญาณ ซึ่งทำงานเร็วขึ้นประมาณ 99.3%

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.9 ผลการวัดการประมวลผลจากระบบจริงที่ใช้ MCS-51 ความถี่ระบบ 8 MHz



รูปที่ 5.10 ผลการจำลองวงจรประมวลผล ด้วยโปรแกรม Foundation 1.5

ในส่วนของการพิจารณาเวลาของระบบทั้งหมดนั้นเมื่อพิจารณาโดยใช้กรณีตัวอย่างเครื่องเย็บรองเท้าซึ่งเป็นเครื่องจักรชนิด 3 แกน คือแกน X, Y, และ Z โดยแกน X และแกน Y เป็นการเคลื่อนที่ในระนาบ 2 มิติ และแกน Z เป็นการเคลื่อนที่ของเข็มเย็บรองเท้าโดยเมื่อมอเตอร์หมุนหนึ่งรอบ ทำให้เข็นเย็บรองเท้าขึ้นลงหนึ่งครั้ง หรือเรียกว่าเป็นการเย็บหนึ่งฝีเข็ม ในการทำการเย็บหนึ่งฝีเข็มเห็นได้ว่าเข็มจะจมอยู่ในเนื้อผ้า เป็นเวลาหนึ่งในสามของเวลาทั้งหมดที่ใช้ในการเย็บหนึ่งฝีเข็ม ซึ่งช่วงเวลาที่เข็มจมอยู่ในเนื้อผ้านี้เอง ทางด้านแกน X และ แกน Y ต้องหยุดอยู่กับที่เพื่อป้องกันมิให้เข็มเย็บหักเสียหาย ในกรณีตัวอย่างเครื่องเย็บรองเท้าที่พัฒนาขึ้น [6] ใช้ความเร็วมอเตอร์ในแกน Z ความเร็วอยู่ที่ประมาณ 3000 รอบต่อนาที มอเตอร์หมุนหนึ่งรอบใช้เวลา 20 *ms* ส่วนทางด้านแกน X และแกน Y หมุนหนึ่งรอบทำให้แกนเคลื่อนที่ 20 *mm* ดังนั้นเพื่อทำการเคลื่อนที่ในระยะ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปกติของฝีเข็ม 3 *mm* จึงจำเป็นต้องใช้เวลา 2/3 ของ 20 *ms* หรือประมาณ 13.33 *ms* ดังนั้นความเร็วรอบของมอเตอร์แกน X และแกน Y อยู่ที่ประมาณ 675 รอบต่อนาที เมื่อเราพิจารณาความเร็วของการประมวลผลในการคำนวณการเคลื่อนที่ดังนั้นตัวประมวลผลจึงจำเป็นต้องคำนวณให้เสร็จภายในเวลา 20 *ms* ซึ่งเป็นค่าความเร็วของมอเตอร์แกน Z เคลื่อนที่หนึ่งรอบ เมื่อใช้ตัวประมวลผล MCS-51 ความถี่ระบบ 8 MHz ต้องใช้เวลาในการประมาณผล 1.337 *ms* เวลาที่ใช้ในการประมวลผลคิดเป็น 6.68% ของเวลาที่ใช้ได้ทั้งหมด ดังนั้นความเร็วรอบสูงสุดของแกน Z ที่ใช้ตัวประมวลผล MCS-51 เป็นตัวประมวลผลอยู่ที่ประมาณน้อยกว่า 44,876 รอบต่อนาที เพราะต้องใช้เวลาส่วนหนึ่งกับการทำงานทางด้านอื่นของเครื่องจักร ส่วนความเร็วรอบสูงสุดของแกน Z ที่ใช้ตัวประมวลผล จี-โค้ดโปรเซสเซอร์ อยู่ที่ประมาณน้อยกว่า 6,486,486 รอบต่อนาที

ในปัจจุบันในการทำงานของเครื่องเย็บรองเท้า ที่ใช้ตัวประมวลผล MCS-51 เป็นตัวประมวลผล เพื่อควบคุมการหมุนของแกน Z โดยสามารถควบคุมความเร็วได้สูงสุดที่ 44,876 รอบต่อนาที ก็พอเพียงกับการใช้ควบคุมมอเตอร์แกน Z ที่ใช้กับเครื่องเย็บรองเท้าในปัจจุบันแล้ว แต่อย่างไรก็ตามได้เริ่มมีการวิจัยเพื่อเพิ่มความเร็วมอเตอร์ [22] ซึ่งความเร็วมอเตอร์อยู่ที่ประมาณ 20,000-200,000 ซึ่งตัวประมวลผล จี-โค้ดโปรเซสเซอร์ เป็นทางเลือกอีกวิธีหนึ่งในอนาคต

5.4 สรุป

ในบทนี้แสดงการทดสอบความถูกต้องในการคำนวณค่าของ หน่วยประมวลผล จี-โค้ดโปรเซสเซอร์ โดยแสดงเป็นแผนภูมิภาพตามความเหมาะสมในแต่ละขั้นตอนของการประมวลผล และ ในบทนี้ยังได้แสดงผลการเปรียบเทียบทางด้านความเร็วของระบบที่มีการพัฒนาอยู่เดิม [6] ที่ใช้ตัวประมวลผลเป็น MCH51 โดยการนับจำนวนสัญญาณที่ใช้ในการคำนวณระหว่างการใช้งานจริง เปรียบเทียบกับ การจำลองการคำนวณค่าของหน่วยประมวลผล จี-โค้ดโปรเซสเซอร์ เพื่อนับจำนวนสัญญาณ และในบทนี้ยังได้ยกตัวอย่างกรณีศึกษาของเครื่องเย็บรองเท้า ที่มีข้อจำกัดทางด้านเวลาของการประมวลผลอยู่ที่ความเร็วรอบของเข็มเย็บผ้า โดยเปรียบเทียบที่ความเร็วสูงสุดที่ตัวประมวลผล MCH51 ทำได้กับความเร็วสูงสุดที่ตัวประมวลผล จี-โค้ดโปรเซสเซอร์ทำได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

บทที่ 6
สรุปผลการวิจัยและข้อเสนอแนะ

6.1 สรุปผลการวิจัย

การใช้งานเครื่องจักรซีเอ็นซีในปัจจุบันนั้น เริ่มมีการเปลี่ยนรูปแบบการใช้งานจากการที่ผู้ใช้ทำการ ออกแบบเส้นทางการเคลื่อนของเครื่องจักรในกระดาษ แล้วใช้ความชำนาญ เปลี่ยนข้อมูลที่อยู่ในกระดาษนั้นให้เป็นคำสั่ง จี-โค้ด แล้วนำคำสั่ง จี-โค้ด ที่สร้างขึ้นมานั้นไปป้อนให้กับเครื่องจักร ซึ่งวิธีนี้เห็นได้ว่าเกิดจากในอดีตเทคโนโลยีทางด้านคอมพิวเตอร์ช่วยออกแบบ ยังไม่มีการใช้งานกันมากนัก ทำให้ผู้พัฒนาระบบเครื่องจักรซีเอ็นซี ต้องออกแบบชุดคำสั่ง จี-โค้ด ให้ทำงานที่ซับซ้อนขึ้นเพื่อสะดวกต่อผู้ใช้งาน แต่อย่างไรก็ตาม เมื่อมีการใช้โปรแกรมคอมพิวเตอร์ช่วยออกแบบมากขึ้นผู้ใช้ไม่จำเป็นต้องเขียนคำสั่ง จี-โค้ดด้วยตัวเองอีก ดังนั้นชุดคำสั่ง จี-โค้ด ที่ซับซ้อนและไม่เป็น มาตรฐานของเครื่องจักร แต่ละบริษัทก็มีความจำเป็นในการใช้งานน้อยลง แนวคิดของจี-โค้ดโปรเซสเซอร์ จึงได้เกิดขึ้นเพื่อลดขนาดของระบบเครื่องจักรซีเอ็นซี [1-5][7][16-20] เพิ่มความเร็วในการประมวลผลให้มากขึ้น และง่ายแก่การออกแบบและพัฒนาระบบเครื่องจักรซีเอ็นซี โดยตัดส่วนที่ใช้ติดต่อกับผู้ใช้ออกแล้วรองรับชุดคำสั่ง จากโปรแกรมคอมพิวเตอร์ช่วยออกแบบแทน

ในงานวิจัยนี้ได้แสดงแนวความคิดในการออกแบบ จี-โค้ดโปรเซสเซอร์ โดยรองรับคำสั่งการเคลื่อนที่ของเครื่องจักร คือคำสั่ง G00, G01, G02 และ G03 ซึ่งได้แบ่งโครงสร้างหลักออก เป็น 4 ส่วน

1) วงจรในการประมวลผล (Arithmetic Unit)

2) วงจรในการควบคุมภายใน (Control Unit)

3) ส่วนของการเก็บข้อมูล และเส้นทางของข้อมูล (Register and Data Path)

4) ส่วนของการเชื่อมต่อกับอุปกรณ์ภายนอก (Interface)

ซึ่งในแต่ละส่วนเหมือนกับการออกแบบโปรเซสเซอร์ทั่วไปยกเว้นในส่วนที่ 1 ที่มีความซับซ้อนในการประมวลผลมากกว่า โดยได้แสดงอัลกอริธึม และหลักการโดยละเอียด เริ่มจากหลักการทางคณิตศาสตร์ ของ คอร์ดิคอัลกอริทึม ซึ่งเป็นหลักการทางคณิตศาสตร์เพื่อคำนวณหาค่าทาง ฟังค์ชันตรีโกณมิติ ซึ่งหลักการนี้เป็นหลักการที่เหมาะสมในการนำมาสร้างเป็นวงจรทำงานจริง โดยมีผู้เสนอผลงานวิจัยในส่วนนี้จำนวนมากตัวอย่างเช่น [8-11]

งานวิจัยนี้ได้วิเคราะห์ ความถูกต้องของหลักการ คอร์ดิคอัลกอริทึม ซึ่งแสดงผลให้เห็นว่ามีการตอบสนองค่าที่ถูกต้องอยู่ในช่วง $-\pi/2$ ถึง $\pi/2$ เพื่อขยายช่วงการทำงานให้ครบตั้งแต่ $-\pi$ ถึง π จึงได้นำหลักการ PreCORDIC และ PostCORDIC [9] เข้ามาแก้ปัญหานี้ ทำให้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตอบสนองค่าได้ตามที่ต้องการ หลังจากนั้นงานวิจัยนี้ได้ แสดงวิธีการและการวิเคราะห์ผลในการนำหลักการ คอร์ดิคอัลกอริทึม มาต่อรวมกันเพื่อแก้สมการของการหาค่า ความเร็วและความเร่ง ซึ่งเป็นคุณสมบัติของ หน่วยประมวลผล โดยผลของการวิเคราะห์ มีค่าเป็นความผิดพลาดอยู่ช่วงที่ยอมรับได้ คืออยู่ในช่วง $\pm 2.5\times10^{-5}$ งานวิจัยนี้ยังได้ทดสอบการออกแบบทาง อัลกอริธึม โดยจำลองการทำงานในส่วนของการคำนวณค่าการเคลื่อนที่เป็นเส้นตรงจากจุดหนึ่งไปอีกจุดหนึ่ง ซึ่งการทำงานใช้สัญญาณนาฬิการะบบ 74สัญญาณ ทั้งนี้ไม่รวมเวลาในการโอนถ่ายข้อมูลระหว่างหน่วยความจำเข้ามาในโปรเซสเซอร์และการโอนข้อมูลในตัวโปรเซสเซอร์เพื่อจัดเตรียมการประมวลผล เปรียบเทียบกับระบบเดิม [6] ที่ใช้ตัวประมวลผล MCH51 ใช้สัญญาณนาฬิการะบบ 10696 สัญญาณ โดยนับรวมการโอนข้อมูลในตัวโปรเซสเซอร์เพื่อจัดเตรียมการประมวลผล แต่ไม่นับรวมการโอนถ่ายข้อมูลระหว่าง หน่วยความจำเข้ามาในโปรเซสเซอร์ เนื่องจากได้กำหนดให้ข้อมูลอยู่ในหน่วยความจำในตัวโปรเซสเซอร์แล้ว ซึ่งเมื่อเปรียบเทียบผลแล้วมีความเร็วในการทำงานเพิ่มขึ้นมาก ประมาณ 99%

เมื่อพิจารณาวิเคราะห์ประสิทธิภาพส่วนคอร์ดิคอัลกอริทึม ที่เป็นอัลกอริทึมพื้นฐาน เพื่อใช้สร้างส่วนประมวลผล ของจี-โค้ดโปรเซสเซอร์ ความถูกต้องในการคำนวณขึ้นกับส่วนสำคัญ 3ส่วน คือ

1) กับลำดับของตัวเลข $S(m,i)$ และ $a(m,i)$ ที่แสดงอยู่ในตารางที่ 3.3 เป็นตัวเลขในการกำหนดการลู่เข้าของ สมการคอร์ดิคอัลกอริทึม ในส่วนนี้เป็นส่วนที่ทำให้เกิดข้อผิดพลาดมากที่สุด
2) วงจรบวกลบตัวเลข เกิดขึ้นเมื่อทำการบวกลบตัวเลขที่เป็นจุดทศนิยม หลังจากที่ทำการบวกลบค่าเสร็จเรียบร้อยแล้วต้องทำการปรับ เลขยกกำลังซึ่งทำให้เกิดค่าผิดพลาดในกรณีที่ตัวเลขทั้งสองมีค่าที่แตกต่างกันมากๆ
3) ค่าที่เราประมาณด้วยตัวเลขในตารางค่าของ $\alpha_{m.i} = \tan^{-1}(2^{-s(m.i)})$ เมื่อแทนด้วยตัวเลขฐานสองทำให้เกิดความผิดพลาดในส่วนของจุดทศนิยม

แต่อย่างไรก็ตามค่าความผิดพลาดของการทำงานตามหลักการคอร์ดิคอัลกอริทึม ก็อยู่ในช่วงที่ยอมรับได้ เพราะมีค่าความผิดพลาดน้อยมากเมื่อเปรียบเทียบกับความผิดพลาดของเครื่องจักร

6.2 แนวทางในการพัฒนา

แนวทางการพัฒนา แบ่งพิจารณาเป็นส่วนๆ โดยเริ่มจากส่วนของตัว จี-โค้ดโปรเซสเซอร์ โดยเพิ่มการทำงานดังนี้

- ทำการรวมส่วนของการควบคุมมอเตอร์เข้ามาอยู่ในส่วนของ จี-โค้ดโปรเซสเซอร์ เพื่อความสะดวกในการเชื่อมต่อและควบคุม มอเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- รวมการทำงานของระบบ PLC (Program Logic Control) เข้ามาอยู่ใน จี-โค้ด โปรเซสเซอร์ เพื่อขยายความสามารถของ ตัวโปรเซสเซอร์ให้ควบคุมเครื่องจักรได้หลายรูปแบบขึ้น
- รวมการทำงานของ เอ็ม-โค้ด บางตัวเพื่อรองรับคำสั่งเฉพาะเครื่องจักรซีเอ็นซี ประเภทต่างๆเฉพาะรูปแบบของเครื่องจักรซีเอ็นซีนั้นๆ

แนวทางในการพัฒนาส่วนของคอร์ดิคอัลกอริทึม เป็นการใช้เทคนิค Pipelined Architecture เพื่อเพิ่มความเร็วในการทำงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

บรรณานุกรม

[1] สินชัย กมลภิวงศ์ และ ทศพร กมลภิวงศ์. "การพัฒนาระบบควบคุมเครื่องพับเหล็กชนิดไฮโดรลิกเพื่ออุตสาหกรรม." การประชุมวิชาการทางไฟฟ้าครั้งที่ 16, พ.ศ.2536, หน้า 590-594

[2] สมหญิง ไทยนิมิต และ วโรดม ตู้จินดา. "การพัฒนาเครื่องกลึงซีเอ็นซีสำหรับฝึกอบรม." การประชุมวิชาการประจำปี ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ, พ.ศ.2537, หน้า 113-119

[3] วโรดม ตู้จินดา. "DSP based multi-axis servo controllers." วิทยานิพนธ์ระดับปริญญาโท คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์, พ.ศ.2536

[4] กวิน สนธิเพิ่มพูน และ คณะ. "การสร้างต้นแบบส่วนควบเครื่องกัดแนวดิ่งซีเอ็นซี และการสร้างชุดฝึกเครื่องกัดแนวดิ่ง ของเครื่องกลึงซีเอ็นซี สำหรับกองวิทยาลัยเทคนิค กรมอาชีวะศึกษา." การประชุมวิชาการประจำปีศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ, พ.ศ. 2537, หน้า 120-125

[5] พิสุทธิ์ อภิชยกุล กวิน สนธิเพิ่มพูน และ โยธินเปรมปราณีรัชต์. "การสร้างต้นแบบคอมพิวเตอร์ควบคุมเครื่องกลึง." การประชุมวิชาการทางไฟฟ้าครั้งที่ 20, พ.ศ.2540, หน้า 586-591.

[6] กฤษณ์ จงสฤษดิ์ และ สุธี ผู้เจริญชนะชัย. "การพัฒนาต้นแบบคอมพิวเตอร์ควบคุมเครื่องจักรกลซีเอ็นซี." การประชุมวิชาการประจำปีศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ, พ.ศ.2542

[7] Dong-Il Kim,Jin-Il Song and Sungkwun Kim. "Design of digital signal processor system for CNC systems." Industrial Electronics Control and Instrumentation, 1991. Proceedings. IECON '91., 1991 International Conference on, vol.3, 1991 , pp1861 -1866

[8] D.timmermann, H.Hahn, B.J. Hosticka, and G.Schmidt. "A Programmable CORDIC Chip for Digital Signal." Processing Application IEEE Journal of Solid-State Circuits, Vol 26, No 9, September 1991

[9] Gerben J. Hekstra and Ed F.A. Deprettere. "Floating Point Cordic." Computer Arithmetic, 1993. Proceedings., 11th Symposium on, 1993, pp130 -137

[10] Helmut Hahn, Dirk Timmermann, Bedrich J.Hosticka and Bernold Rix. "A Unified and Division-Free CORDIC Argument Reduction Method With Unlimited Convergrnce Domain Includeing Inverse Hyperbolic Function." IEEE Transaction on Computer, Vol 43, No 11, November 1994

เอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

[11] D.timmermann, B.Rix, H.Hahn and B.J. Hosticka. "A CMOS Floating-Point Vector-Arithmetic Unit." IEEE Journal of Solid-State Circuit, Vol 29, No 5, May 1994

[12] John W. Carter. "Microprocessor Architecture and Microprogramming." A State Machine Approach, ISBN 0-13-192253-X, 1995

[13] M.Moreaux and Dr.J.D. Decotignie. "CNC and PLC: are they one and the same for Machine-Tool?" Industrial Electronics Control and Instrumentation, IECON '91. 1991, pp924-927

[14] Michel MOREAUX. "A CNC Model Well-Suited for the Requirement of CNC Soft Construction Environment." Control System, IEEE, 1993, pp176-181

[15] J.D. Decotignie. "Fieldbus and CNC Architecture: towards a CNC software factory." Symposium on Emerging Technologies & Factory Automation, IEEE, 1994, pp338-344

[16] Dong-Il Kim and Chang Hyuk Yim. "All Digital High Performance Controller for Spindle Motor in CNC Machine Tool" MC2 IEEE, 1997,pp2.1-2.3

[17] Dong-Il Kim,Jin-Il Song and Sungkwun Kim. "Dependence of Machining Accuracy on Acceleration/Decelation and Interpolation Methods in CNC Machine Tools." IEEE, 1994, pp1898-1905

[18] F. Butera, B. Fontanella,P. Nesi and M. Perfetti "Reengineering a Computerized Numerical Control Towards Object-Oriented." IEEE, 1998, pp224-227

[19] ศักดิ์ชัย พิทักษ์เสรีสกุล, ทศัย ตันไถง และ สุรพนธ์ ตุ้มนาค. "การควบคุมชุดขับเคลื่อน 2 แกน โดยใช้ฐานข้อมูลภาพจากซอฟท์แวร์ช่วยออกแบบ." การประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 20, พ.ศ.2540, หน้า 100-105

[20] รวิ อุดตมธนินทร์. "การสร้างต้นแบบส่วนควบคุมเครื่องกัดแนวดิ่งซีเอ็นซี." วิทยานิพนธ์วิศวกรรมศาสตร์มหาบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า บัณฑิตวิทยาลัย, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, พ.ศ.2541

[21] Hoang Le-Huy. "Microprocessor and Digital IC's for Motion Control." Proceedings of The IEEE, Vol 82, No 8, August 1994, pp1140-1163

[22] Saari J. "Thermal Analysis of High-Speed Induction Machines." Acta Polytechnica Scandinavica Electrical Engineering Series No90, ISBN 952-5148-43-2, 1998, pp73

[23] อมร ช่วยชู และ บรรจงปิยธำรง. "การออกแบบสถาปัตยกรรม G-Code โปรเซสเซอร์." National Computer Science and Engineering Conference NCSEC 2000, November 2000, pp197-204

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ก.

โปรแกรมภาษา C ที่ใช้กับ ตัวประมวลผล MCH51 เพื่อคำนวณคำสั่ง G00 และ G01

```
#include <reg320.H>
#include <stdio.h>
#include <absacc.h>
#include <ctype.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <intrins.h>

// define 8255 address1
#define o_port1   XBYTE[0x7F04]
#define o_port2   XBYTE[0x7F05]
#define o_port3   XBYTE[0x7F06]
#define pgmem_op  XBYTE[0x7C00]

// define 8255 address2
#define in_port1   XBYTE[0x7F00]
#define in_port2   XBYTE[0x7F01]
#define in_port3   XBYTE[0x7F02]

sbit pmd0_rdy = 0x90;          // p1.0  input bit
sbit all_signal_en = 0x94;     // p1.4  input bit

//-------------------------------------
// prototype declare   AxpBkup.c
//------------------ --------------------
signed long jab(signed long a)
{
   signed long ans=1;

    if(a < 0) ans =-1;

   return(ans);
}


void main(void)
{
char count =0;
signed long ddx = 1,ddy = 1;
signed long rr1 = 5,rr2 = 4;
signed long rr3 = 7,rr4 = 8;
signed long sg1,sg2;
float reg_flt=0.0,reg_flt1;
float vx1 = 1.0,vy1 = 1.0;
float ax1 = 1.0,ay1 = 1.0;
float VMAX = 1.0,AMAX = 1.0;
unsigned long ct=0;

        EA =0;
        CKCON = CKCON&0xD7;
        PX0 = 1;
        IT0 = 0;          // Start timer0
        TMOD = 0x01;
        EA =0;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    EX0 = 0;
    ET0 = 0;
    all_signal_en=0;
  while((in_port1 & 0x40) == 0) { }
    o_port3 = 0x96;
    o_port1 = 0xff;
    o_port2 = 0xff;

    while(1)
    {

            pmd0_rdy = 1;           // Stat Calculating
            sg1 = jab(rr2 - rr1);
            sg2 = jab(rr4 - rr3);

            ddx = labs(rr2 - rr1);
            ddy = labs(rr4 - rr3);


            reg_flt1 = ddx/1000.0;
            reg_flt =  ddy/1000.0;
            vx1 = sqrt(reg_flt1*reg_flt1 + reg_flt*reg_flt);
            reg_flt = reg_flt / vx1;
            reg_flt1 = reg_flt1 / vx1;
            vy1 = VMAX * reg_flt;
            vx1 = VMAX * reg_flt1;
            ay1 = AMAX * reg_flt;
            ax1 = AMAX * reg_flt1;
            pmd0_rdy = 0;           // End Calculating

            for(ct =0;ct < 50000;ct++){}
            reg_flt1 = reg_flt1 + 0.0001;
            reg_flt = reg_flt + 0.0001;
            rr2 = rr2 + 1;
            rr1 = rr1 + 1;
            rr3 = rr3 + 1;
            rr4 = rr4 + 1;
            VMAX = VMAX + 0.0;
             AMAX = AMAX + 0.0;
    }
}

//---------- END MAIN ----------
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ประวัติผู้เขียน

ชื่อผู้เขียน	นาย อมร ช่วยชู
วันเดือนปีเกิด	วันที่ 21 มีนาคม 2517
สถานที่เกิด	จังหวัดกรุงเทพฯ
วุฒิการศึกษาระดับปริญญาตรี	วิศวกรรมศาสตร์บัณฑิต(เกียรตินิยมอันดับสอง) สาขาวิชาวิศวกรรมศาสตร์คอมพิวเตอร์ สถาบันเทคโนโลยีเพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีที่สำเร็จการศึกษา	ปีการศึกษา 2539
ผลงานทางวิชาการที่ได้รับการตีพิมพ์	- การสร้างภาพเสมือนจริง การประชุมทางวิชาการวิศวกรรมไฟฟ้าครั้งที่ 20พ.ศ.2540 - การออกแบบสถาปัตยกรรม จี-โค้ด โปรเซสเซอร์ National Computer Science and Engineering Conference (NESEC 2000)
ประสบการณ์การทำงาน	ผู้ช่วยนักวิจัย ในโครงการ พัฒนาเครื่องจักร CNC หน่วยวิจัย Computer and Automation Technology Lab(CTL) ศูนย์เทคโนโลยีอีเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ (NECTEC) พ.ศ.2540 ถึง พ.ศ.2545

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้