การควบคุมอินดักชั่นมอเตอร์ด้วยเทคนิคเวคเตอร์มอดูเลชั่น โดยใช้ FPGA

FPGA-BASED INDUCTION MOTORS CONTROL WITH VECTOR MODULATION TECHNIQUE

ประสูตร เดชสุวรรณ
PRASOOT DECHSUWAN

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์
บัณฑิตวิทยาลัย
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2549

ISBN 974-9309-17-0

การควบคุมอินดักชั่นมอเตอร์ด้วยเทคนิคเวคเตอร์มอดดูเลชั่น โดยใช้ FPGA

FPGA-BASED INDUCTION MOTORS CONTROL WITH VECTOR MODULATION TECHNIQUE

ประสูตร เดชสุวรรณ

PRASOOT DECHSUWAN

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2549

ISBN 974-8308-17-0

FPGA-BASED INDUCTION MOTORS CONTROL WITH VECTOR MODULATION TECHNIQUE

PRASOOT DECHSUWAN

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

MASTER OF ENGINEERING IN ELECTRONIC ENGINEERING

SCHOOL OF GRADUATE STUDIES

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2006

ISBN 974-8308-17-0

หัวข้อวิทยานิพนธ์	การควบคุมอินดักชั่นมอเตอร์ด้วยเทคนิคเวคเตอร์มอดดูเลชั่น โดยใช้ FPGA
นักศึกษา	นาย ประสูตร เดชสุวรรณ
รหัสประจำตัว	44611302
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมอิเล็กทรอนิกส์
พ.ศ.	2549
อาจารย์ผู้ควบคุมวิทยานิพนธ์	ผศ.พลผดุง ผดุงกุล

บทคัดย่อ

วิทยานิพนธ์นี้ นำเสนอการประยุกต์ใช้งาน FPGA (Field programmable gate array) Xilinx Spartan II (XC2S100) สร้างสัญญาณพีดับเบิลยูเอ็ม (PWM) ด้วยเทคนิค เวคเตอร์มอดดูเลชั่น สำหรับอินเวอร์เตอร์เพื่อควบคุมอินดักชั่นมอเตอร์เหนี่ยวนำกระแสสลับ โดยสามารถควบคุมความถี่ของสัญญาณไฟฟ้ากระแสสลับ ที่จ่ายให้กับมอเตอร์ได้ รวมถึงความถี่การสวิตช์สูงสุดที่ ใช้ในการทดสอบวงจรอินเวอร์เตอร์ที่ $30KHz$ ค่าดัชนีการมอดดูเลชั่นปรับได้ถึง 100% และ deadtime ได้ต่ำสุดที่ $330nS$

Thesis	FPGA-BASED INDUCTION MOTORS CONTROL WITH VECTOR MODULATION TECHNIQUE
Student	Mr.Prasoot Dechsuwan
Student ID	44611302
Degree	Master of Engineering
Program	Electronic Engineering
Year	2006
Thesis Advisor	Asst. Prof. Polphadung Phadungkul

ABSTRACT

This thesis presents an application of a Xilinx FPGA Device, Spartan II (XC2S100), in generating Pulse Width Modulation (PWM) signals using Space vector modulation (SVM) technique for insulated gate bipolar transistors (IGBTs) inverter to control Alternating current induction motors. This designed circuit can generate SVM PWM signal in many different fundamental frequencies , switching frequency ,modulation index and dead time.

กิตติกรรมประกาศ

วิทยานิพนธ์เล่มนี้สำเร็จลุล่วงได้ด้วยความกรุณาจาก ผศ.พลผดุง ผดุงกุล ซึ่งเป็นอาจารย์ผู้ควบคุมวิทยานิพนธ์ ผู้วิจัยรู้สึกซาบซึ้งในความอนุเคราะห์ของท่านที่ให้คำปรึกษา และสนับสนุนอุปกรณ์การทำวิจัย จึงขอขอบพระคุณเป็นอย่างสูง

ขอขอบพระคุณ คุณพ่อ คุณแม่ พี่ชาย และพี่สาวซึ่งเป็นที่รักและเคารพยิ่ง ที่คอยห่วงใยเป็นกำลังใจ และสนับสนุนด้านการศึกษาเป็นอย่างดีมาโดยตลอด

ขอขอบคุณ อาจารย์ เทียนไชย นกครุฑ อาจารย์ ไพบูลย์ ธานินทร์สุรัตน์ ดร.ธีรยศ เวียงทอง ดร.สมภพ ภูริวิกรัยพงศ์ อาจารย์ วิศิษฐ์ สุขจิตร อาจารย์ภาควิชาวิศวกรรมอิเล็กทรอนิกส์ มหาวิทยาลัยเทคโนโลยีมหานคร ที่ให้คำปรึกษา เครื่องมือ อุปกรณ์ทำวิจัย และช่วยเหลือเป็นอย่างดี

สุดท้ายนี้ขอขอบพระคุณ อาจารย์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุกท่านที่ให้คำปรึกษา และถ่ายทอดความรู้ให้แก่ข้าพเจ้าขณะที่ได้ศึกษาในสถาบันแห่งนี้

คุณค่า และประโยชน์อันพึงมีจากวิทยานิพนธ์นี้ ข้าพเจ้าขอมอบแต่ผู้มีพระคุณทุกท่าน

ประสูตร เดชสุวรรณ

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

สารบัญรูป(ต่อ)

รูปที่ หน้า

สารบัญรูป(ต่อ)

รูปที่ หน้า

สารบัญรูป(ต่อ)

รูปที่ หน้า

สารบัญรูป(ต่อ)

รูปที่ หน้า

สารบัญรูป(ต่อ)

รูปที่ หน้า

สารบัญรูป(ต่อ)

สารบัญรูป(ต่อ)

รูปที่ หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมา และความสำคัญของปัญหา

ในปัจจุบันมอเตอร์เหนี่ยวนำกระแสสลับ (Induction motor) ทั้งแบบ 1 เฟส และ 3 เฟส ได้ถูกนำมาใช้งานอย่างแพร่หลายทั้งในระบบการผลิตของโรงงานอุตสาหกรรม และในอุปกรณ์เครื่องใช้ไฟฟ้าสำหรับใช้งานในบ้านพักอาศัย อาทิเช่น ปั๊มน้ำ ระบบสายพานลำเลียง พัดลมเป่าอากาศ และคอมเพรสเซอร์ของระบบเครื่องปรับอากาศ เป็นต้น

การเลือกประเภทของมอเตอร์เหนี่ยวนำกระแสสลับ รวมไปถึงการเลือกวิธีการควบคุมการทำงานของมอเตอร์เป็นประเด็นที่จะต้องมีการพิจารณาถึงความเหมาะสมของการใช้งานทั้งในด้านวิศวกรรมศาสตร์ และเศรษฐศาสตร์ ที่เกี่ยวข้องกับการออกแบบ ต้นทุนการสร้าง และการประหยัดกำลังงาน เนื่องจากมีความหลากหลายของประเภทมอเตอร์เหนี่ยวนำกระแสสลับที่มีการจำหน่ายในปัจจุบัน ทั้งแบบ 1 เฟส และ 3 เฟส นอกจากนี้การควบคุมมอเตอร์ยังจะต้องคำนึงถึงปัจจัยต่างๆ ที่เกี่ยวข้องกับความต้องการของผู้ใช้ อาทิเช่น แรงบิด ความเร็วรอบของมอเตอร์ รวมไปถึงค่ากำลังงานที่จะต้องใช้ในการขับมอเตอร์ดังกล่าว โดยปัจจัยเหล่านี้จะขึ้นอยู่กับคุณสมบัติของมอเตอร์ ซึ่งมีความแตกต่างกันไป

นอกจากประเด็นที่ได้กล่าวในข้างต้นแล้ว ยังมีข้อปัญหาในเชิงเทคนิคที่จำเป็นที่จะต้องพิจารณาในเชิงลึก โดยเฉพาะปัญหาการสูญเสียกำลังงานบางส่วนในช่วงสภาวะเริ่มต้นของการขับมอเตอร์ ซึ่งเกิดจากการนำมอเตอร์เหนี่ยวนำกระแสสลับ ทำงานร่วมกับสวิตช์ตัดต่อวงจร เพื่อให้การทำงานของมอเตอร์เป็นไปในลักษณะแบบ ทำงาน และ หยุดทำงานเป็นช่วงๆสลับกัน หรือเรียกว่า การทำงานแบบเปิด/ปิด เช่นในกรณีเครื่องปรับอากาศ ซึ่งในระบบคอมเพรสเซอร์ ของเครื่องปรับอากาศ มีส่วนประกอบที่สำคัญคือ มอเตอร์เหนี่ยวนำกระแสสลับ ซึ่งการทำงานลักษณะเปิด/ปิด เช่นนี้ จะทำให้มีการสูญเสียกำลังงานบางส่วนในสภาวะเริ่มต้น เนื่องจากกระแสเฉลี่ยในช่วงเริ่มต้นมีขนาดสูงกว่าสภาวะปกติ และไม่สามารถควบคุมการทำงานได้อย่างต่อเนื่อง รวมทั้งการควบคุมความเร็วของมอเตอร์เหนี่ยวนำกระแสสลับนั้นจะขึ้นอยู่กับความถี่ของแรงดันไฟฟ้าที่ป้อนให้เป็นหลัก

จากปัญหาที่ได้กล่าวถึงในข้างต้น ทำให้วิทยานิพนธ์ฉบับนี้ได้นำเสนอการวิจัย ที่เกี่ยวข้องกับการออกแบบอินเวอร์เตอร์ เพื่อควบคุมมอเตอร์เหนี่ยวนำกระแสสลับ ให้มีการลดอัตราการสิ้นเปลืองกำลังงานไฟฟ้า โดยนำเสนอหลักการควบคุมความเร็วรอบของมอเตอร์ โดยใช้การเปลี่ยนแปลงความถี่ของแรงดันไฟสลับที่จ่ายให้กับมอเตอร์เหนี่ยวนำกระแสสลับ

1.2 ความมุ่งหมาย และวัตถุประสงค์ของการศึกษา

การวิจัยนี้ศึกษาการควบคุมมอเตอร์เหนี่ยวนำกระแสสลับ 3เฟส และ 1เฟส เพื่อจะควบคุมการทำงานของมอเตอร์ ด้วยอินเวอร์เตอร์วิธี Pulse Width Modulation (PWM) โดยใช้ FPGA (Field programmable gate array) เป็นตัวสร้างสัญญาณ PWM โดยใช้เทคนิค Space Vector Modulation (SVM)

ในงานวิจัยนี้ได้นำเสนอหลักการออกแบบวงจรควบคุมการทำงานของมอเตอร์เหนี่ยวนำกระแสสลับ 3เฟส ที่ควบคุมค่าความถี่แรงดันไฟสลับที่ใช้งานกับมอเตอร์ รวมทั้งค่าดรรชนีการมอดดูเลต (modulation index), ความถี่ในการสวิตช์ เพื่อสร้างสัญญาณ SVM PWM โดยในวิทยานิพนธ์ฉบับนี้มุ่งเน้นการควบคุมมอเตอร์เหนี่ยวนำกระแสสลับแบบ 3 เฟสเป็นหลัก อย่างไรก็ตามหลักการที่ได้นำเสนอ สามารถที่จะถูกนำไปประยุกต์ใช้งานกับระบบ 1เฟส

1.3 สมมติฐานของการศึกษา

ในการออกแบบอินเวอร์เตอร์ควบคุมมอเตอร์เหนี่ยวนำกระแสสลับ โดยใช้เทคนิค SVM มีหลายรูปแบบ อาทิเช่น

รูปแบบที่ 1 การใช้งานอุปกรณ์พื้นฐานทางดิจิตอล อาทิเช่น หน่วยความจำแบบ ROM ทำงานร่วมกับวงจรนับ [1] ซึ่งทำให้วงจรของทั้งระบบไม่ซับซ้อน โดยค่าเวลาในการสวิตช์ จะถูกคำนวณไว้ล่วงหน้าและนำไปเก็บไว้ใน ROM อย่างไรก็ตามรูปแบบนี้มีข้อด้อยในด้านการปรับเปลี่ยนค่าต่างๆทำได้ไม่สะดวกนัก เนื่องจากจะต้องนำอุปกรณ์ ROM ไปโปรแกรมใหม่

รูปแบบที่ 2 การใช้งานอุปกรณ์จำพวกไมโครคอนโทรลเลอร์ อาทิเช่น ไมโครคอนโทรลเลอร์ตระกูล PIC หรือ MCS 8051 ซึ่งเป็นที่นิยมใช้กันมาก [2] โดยมีข้อดีในด้านความสะดวกต่อการใช้งาน และปรับเปลี่ยนแก้ไขได้ โดยค่าเวลาในการสวิตช์ก็จะยังถูกคำนวณไว้ล่วงหน้า และเก็บไว้ในอุปกรณ์ ROM เช่นกัน แต่การปรับเปลี่ยนแก้ไขได้ง่ายกว่าแบบแรก อย่างไรก็ตาม ความเร็วของการสวิตช์ และการทำงานยังคงถูกจำกัดอยู่ เนื่องจากอัตราการสวิตช์ที่ควบคุมโดย MCS 8051 ถูกจำกัดอยู่ที่ประมาณ 2-3 KHz

รูปแบบที่ 3 การใช้งานอุปกรณ์ตัวประมวลสัญญาณ DSP (Digital Signal Processor) [3] ซึ่งทำให้ได้ความถี่ในการสวิตช์ที่สูงขึ้น [4] อย่างไรก็ตาม ต้นทุนเชิงเศรษฐศาสตร์โดยเฉพาะในแง่

ของราคาโดยรวมของระบบจะมีค่าสูงขึ้น อีกทั้งการพัฒนาระบบต้องใช้เวลาค่อนข้างนาน เนื่องจากจะต้องทำความเข้าใจในชุดคำสั่งที่มีจำนวนมาก รวมไปถึงตัวโครงสร้างของ DSP เองที่มีความซับซ้อนสูงกว่าทั้งสองแบบข้างต้นที่กล่าวมา นอกจากนี้ การใช้งานตัวประมวลสัญญาณ DSP กับวงจรอิเล็กทรอนิกส์กำลังนั้นยังประสบปัญหาด้านสัญญาณรบกวนสูง เนื่องจากในวงจร

อิเล็กทรอนิกส์กำลังก่อให้เกิดสัญญาณรบกวนจำนวนมาก ทำให้ระบบไม่เสถียรภาพ และทำให้เกิดความเสี่ยงที่จะทำให้ระบบโดยรวมเสียหาย

จากการศึกษาทั้งสามรูปแบบ ทำให้ผู้วิจัยได้แนวความคิด ที่จะใช้อุปกรณ์ฮาร์ดแวร์มาใช้ในการควบคุม ซึ่งจะทำให้ระบบมีเสถียรภาพ และมีความเร็วที่สูงกว่าการควบคุมระบบด้วยซอฟแวร์ โดยในวิทยานิพนธ์ฉบับนี้ อุปกรณ์ FPGA ได้ถูกเลือกนำมาใช้งานเป็นฮาร์ดแวร์ โดยมีจุดเด่นที่สามารถโปรแกรมได้ นอกจากนี้ในการออกแบบระบบสามารถที่ปรับเปลี่ยนฮาร์ดแวร์ได้ โดยไม่มีผลเสีย อีกทั้งใช้เวลาในการออกแบบระบบที่น้อยกว่า แม้ว่าในงานวิจัยด้านนี้ได้เคยนำเสนอมาแล้ว [5][6][7] แต่การเสนอดังกล่าวไม่สามารถที่จะปรับเฟสของสัญญาณ PWM และออกแบบมาเฉพาะงานเท่านั้น อาทิเช่น การควบคุมมอเตอร์เหนี่ยวนำกระแสสลับ 3 เฟส หรือ permanent magnet AC servo motor เพียงอย่างเดียว แต่ในวิทยานิพนธ์ฉบับนี้นำเสนอการออกแบบที่สามารถใช้งานได้หลากหลาย โดยระบบเป็นการออกแบบให้สามารถปรับเฟสของสัญญาณ PWM ให้ใช้ได้กับมอเตอร์เหนี่ยวนำกระแสสลับ 3 เฟส และ 1 เฟส โดยค่าพารามิเตอร์ เช่น ความถี่ในการสวิตช์ ค่า dead time ดรรชนีการมอดดูเลต และมุมเฟสสามารถปรับเปลี่ยนได้ และสามารถควบคุมไม่ให้เกิดการกระชากของกระแส ซึ่งเกิดขึ้นกับกรณีที่ใช้การ เปิด/ปิด มอเตอร์เหนี่ยวนำทำให้สามารถลดกำลังงานสูญเสียลงได้ โดยการปรับความถี่ของมอเตอร์เหนี่ยวนำเพื่อควบคุมความเร็วรอบจึงไม่จำเป็นต้องใช้วิธีการควบคุมการทำงานมอเตอร์เหนี่ยวนำแบบ เปิด/ปิด

ในวิทยานิพนธ์ฉบับนี้ การออกแบบสามารถทำได้โดยใช้ Single programmable hardware chip (Xilinx เบอร์XC2S100) ซึ่งไม่ต้องอาศัยหน่วยความจำภายนอก และจะทำให้วงจรมีขนาดเล็กและสะดวกต่อการใช้งาน

1.4 ขอบเขตการวิจัย

สำหรับรายละเอียดของวิทยานิพนธ์ฉบับนี้จะมีดังต่อไปนี้

<u>บทที่ 2</u> กล่าวถึงมอเตอร์เหนี่ยวนำกระแสสลับ 3 เฟส และ1เฟส โครงสร้างของมอเตอร์เหนี่ยวนำ3 เฟส และ1เฟส สนามแม่เหล็กหมุนของมอเตอร์เหนี่ยวนำ 3 เฟส ทฤษฎีสนามแม่เหล็กขวางของมอเตอร์เหนี่ยวนำ 1เฟส ค่าสลิปของมอเตอร์ การควบคุมอินเวอร์เตอร์แบบ PWM อินเวอร์เตอร์ 3 เฟส หลักการ SVM โครงสร้าง และ คุณสมบัติทั่วไปของ FPGA

<u>บทที่ 3</u> กล่าวถึงการออกแบบวงจรที่ใช้ในระบบ ซึ่งประกอบไปด้วย การออกแบบสัญญาณควบคุมมอเตอร์ การหา modulation index และมุมเฟส การออกแบบสัญญาณควบคุมสวิตช์ การออกแบบส่วนเชื่อมต่อระหว่างชุดรับค่าจากไมโครคอนโทรลเลอร์ กับชุด FPGA ที่ใช้ในการสร้างสัญญาณควบคุมสวิตช์ การออกแบบภาคขับสวิตช์ IGBT การออกแบบสร้างสัญญาณ PWM ที่ใช้ FPGA การออกแบบวงจรทวีแรงดัน

บทที่ 4 เป็นส่วนของการประยุกต์ใช้ FPGA เพื่อสร้างสัญญาณ SVM PWM โดยจะอธิบายการทำงานบล๊อคไดอะแกรม ของ FPGA ซึ่งจะประกอบไปด้วย บล๊อคของ input decoder parameter ซึ่งในรับค่าอินพุทจากไมโครคอนโทรลเลอร์ บล็อค MEM_ACCESS บล๊อค CAL_TIME บล็อค VEC_MOD บล็อค DEAD_TIME ของระบบมอเตอร์เหนี่ยวนำแบบ3 เฟส และ 1เฟส

บทที่ 5 แสดงผลการทดลองที่ได้จากการทดสอบระบบอินเวอร์เตอร์ ขับมอเตอร์เหนี่ยวนำแบบ 3 เฟส และ1เฟส พร้อมทำการทดสอบภาระทางกลของมอเตอร์เหนี่ยวนำ 3เฟส ขนาด 1 แรงม้า

บทที่ 6 สรุปการวิจัย และข้อเสนอแนะที่เกิดจากการทดลอง

ส่วนในภาคผนวก ก. จะเป็นรายละเอียดของ วงจรทั้งหมด ภาคผนวก ข. เป็นรายละเอียดของโปรแกรมในส่วนของ ไมโครคอนโทรลเลอร์ และ FPGA

บทที่2
ทฤษฎีที่เกี่ยวข้อง

2.1 มอเตอร์เหนี่ยวนำ 3 เฟส (Three Phase Induction Motor)

มอเตอร์เหนี่ยวนำเป็นมอเตอร์ที่นิยมใช้มากที่สุด ซึ่งมอเตอร์ที่เกิดจากการเหนี่ยวนำนี้อาจเป็นมอเตอร์เหนี่ยวนำเฟสเดียว หรือมอเตอร์เหนี่ยวนำหลายเฟส (Poly Phase Induction Motor) ก็ได้ มอเตอร์เหนี่ยวนำหลายเฟสนั้นโดยมากแล้วจะเป็นมอเตอร์เหนี่ยวนำ 3เฟส ซึ่งมีข้อดี และข้อด้อย คือ

ข้อดี (Advantage)

1. เป็นมอเตอร์ชนิดที่สร้างขึ้นได้ง่าย และ ทนทาน โดยเฉพาะชนิดกรงกระรอก (Squirrel-cage Type)
2. ราคาไม่แพง และไม่เสียง่าย
3. มีประสิทธิภาพที่สูงพอในสภาวะที่มอเตอร์หมุนปกติ ไม่มีแปรงถ่านดังนั้นการสูญเสีย เนื่องจากความฝืดจึงลดลงหรือมีค่าน้อยมาก และมีเพาเวอร์แฟกเตอร์ดี
4. ต้องการการดูแล และบำรุงรักษาต่ำ
5. สามารถที่จะเริ่มหมุน(Start) ได้ง่าย โดยเฉพาะชนิดกรงกระรอก

ข้อด้อย (Disadvantage)

1. ความเร็วรอบของมอเตอร์ไม่สามารถที่จะเปลี่ยนแปลงได้อันเนื่องมาจากความถี่หลักมูลของแรงดันไฟฟ้าสลับที่จ่ายให้กับมอเตอร์ไม่สามารถปรับเปลี่ยนได้
2. มีคุณสมบัติเหมือนกับมอเตอร์ไฟฟ้ากระแสตรงแบบชันต์ ความเร็วรอบจะลดลงหรือเพิ่มขึ้นจะขึ้นอยู่กับโหลด
3. แรงบิดในขณะเริ่มหมุนของมอเตอร์เหนี่ยวนำค่อนข้างต่ำกว่าแรงบิดขณะเริ่มหมุนของมอเตอร์ไฟฟ้ากระแสตรงแบบชันต์

2.2 โครงสร้างของมอเตอร์เหนี่ยวนำ 3 เฟส

มอเตอร์เหนี่ยวนำ 3เฟสประกอบด้วยส่วนประกอบหลัก 2 ส่วนด้วยกันคือ

2.2.1 สเตเตอร์หรือส่วนที่อยู่กับที่ (Stator)

2.2.2 โรเตอร์หรือส่วนที่หมุน(Rotor)

2.2.1 สเตเตอร์หรือส่วนที่อยู่กับที่

สเตเตอร์ของมอเตอร์เหนี่ยวนำ 3เฟสใช้หลักการเดียวกันกับของซิงโครนัสมอเตอร์ หรือเครื่องกำเนิดไฟฟ้ากระแสสลับ โดยทำมาจากแผ่นเหล็กบางๆอัดซ้อนเข้าด้วยกัน และทำเป็นช่องสลอตไว้บรรจุขดลวด และจำนวนขั้วแม่เหล็กจะเป็นตัวกำหนดความเร็วรอบของมอเตอร์เมื่อเราจ่ายไฟฟ้ากระแสสลับให้กับขดลวดที่สเตเตอร์ จะทำให้เกิดสนามแม่เหล็กที่คงที่ค่าหนึ่ง และสนามแม่เหล็กนี้จะหมุน (Revolves or Rotate) ด้วยความเร็วที่เรียกว่าความเร็วซิงโครนัส และหาได้โดย

$$N_S = \frac{120f}{P} \qquad (2.1)$$

N_S = ความเร็วซิงโครนัส

f = ความถี่หลักมูลของไฟฟ้ากระแสสลับ

P = จำนวนขั้วแม่เหล็กของมอเตอร์

สนามแม่เหล็กที่หมุนจะเหนี่ยวนำแรงดันไฟฟ้าขึ้นในโรเตอร์ ซึ่งเป็นไปตามกฎของการเหนี่ยวนำ

2.2.2 โรเตอร์ หรือส่วนที่หมุน

โรเตอร์ของมอเตอร์เหนี่ยวนำ 3เฟส แบ่งออกเป็น 2 ชนิดคือ

ก. โรเตอร์แบบกรงกระรอก มอเตอร์ที่ใช้โรเตอร์ชนิดนี้เราเรียกว่า มอเตอร์เหนี่ยวนำแบบกรงกระรอก โดยประมาณ 90% ของมอเตอร์เหนี่ยวนำจะใช้โรเตอร์เป็นแบบกรงกระรอก ทั้งนี้เป็นเพราะว่าโรเตอร์ชนิดนี้เป็นชนิดที่ทำได้ง่ายและทนทานที่สุด โรเตอร์ชนิดนี้ประกอบด้วยแผ่นเหล็กบางๆอัดซ้อนกันเป็นรูปทรงกระบอก และถูกทำให้เป็นช่องสลอตให้ขนานกันเพื่อสำหรับฝังหรือบรรจุตัวนำโรเตอร์ (Rotor Conductor) ลงในช่องสลอตนั้น ตัวนำที่ฝังนี้จะไม่มีลักษณะเป็นเส้นหรือเป็นสาย แต่จะเป็นแท่งทองแดงหรืออลูมินั่ม หรืออัลลอย (Copper Bar or Aluminum Bar or

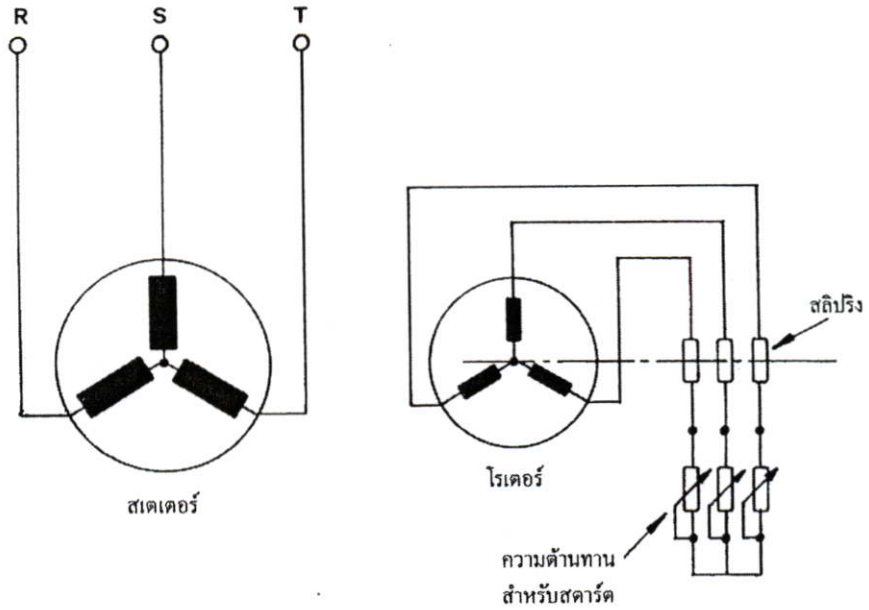
Alloy) โดยในหนึ่งสลอตจะบรรจุแท่งทองแดง หรือ อลูมิเนียมเพียง 1 แท่งเท่านั้น และที่ปลายสุดของแท่งตัวนำทั้งสองด้านนั้นในแต่ละด้านจะถูกต่อปลายลัดวงจรเข้าด้วยกันโดยการบัดกรี (Brazed) หรือเชื่อมด้วยไฟฟ้าโรเตอร์ของมอเตอร์แบบกรงกระรอกนี้แท่งตัวนำจะถูกลัดวงจรไว้อย่างถาวร ดังนั้นจึงไม่สามารถที่จะนำความต้านทานจากภายนอกมาต่ออนุกรมเข้ากับวงจรโรเตอร์เพื่อช่วยในการเริ่มหมุนได้ สลอตของโรเตอร์จะไม่อยู่ในลักษณะที่ขนานกับเพลา แต่จะวางให้มีลักษณะเฉียงเล็กน้อย เพื่อให้เกิดประโยชน์ได้2ทาง

1) จะช่วยให้มอเตอร์หมุนได้อย่างเร็ว โดยการลดการเกิดเส้นแรงแม่เหล็กฮัม (Magnetic Hum)
2) จะช่วยในการลดการเกิดยึด หรือล็อกของโรเตอร์อันเนื่องมาจากสนามแม่เหล็กที่ตกค้างอยู่ที่ฟัน(Teeth) ของสเตเตอร์กับโรเตอร์ทั้งสอง

ส่วนแบบอื่นๆของโรเตอร์ที่มีลักษณะคล้ายกันกับโรเตอร์แบบกรงกระรอกนั้นประกอบด้วยโซลิดไซลินเดอร์ (Solid Cylinder) ของแท่งเหล็ก (Steel) ซึ่งปราศจากสลอตสำหรับบรรจุตัวนำทั้งหมดมอเตอร์จะหมุนได้ขึ้นอยู่กับผลของการเกิดกระแสไหลวนในเหล็กของโรเตอร์

ข. โรเตอร์แบบพันขดลวดหรือเฟสวาวด์โรเตอร์ (Wound Rotor or Phase Wound Rotor)

มอเตอร์ที่ใช้โรเตอร์ชนิดนี้เรียกว่ามอเตอร์เหนี่ยวนำแบบโรเตอร์พันขดลวด หรือเฟสวาวล์มอเตอร์ หรือสลิปริงมอเตอร์ (Wound Rotor or Phase Wound Rotor or Slip-ring Motor) โรเตอร์ชนิดนี้จะพบมากในมอเตอร์เหนี่ยวนำ 3เฟส และมีการพันแบบชุดขดลวดสองชั้นเหมือนกับขดลวดที่ใช้ในเครื่องกำเนิดไฟฟ้ากระแสสลับ ในโรเตอร์ชนิดนี้ภายในจะต่อแบบสตาร์ และมีปลายสายออกมา 3 ปลายต่อเข้ากับสลิปริงที่ติดกับเพลาของโรเตอร์นั้น และโรเตอร์แบบโรเตอร์พันขดลวดสามารถที่จะนำความต้านทานที่ต่อแบบสตาร์ต่อเข้ากับสลิปริงของโรเตอร์ เพื่อช่วยในการเริ่มหมุนของมอเตอร์ เป็นการเพิ่มแรงบิดขณะเริ่มหมุนของมอเตอร์นั้น ความต้านทานที่นำมาต่อเข้าไปนี้มีลักษณะการต่อดังแสดงในรูปที่ 2.1 แต่เมื่อมอเตอร์เริ่มหมุนไปแล้ว และหมุนด้วยความเร็วปกติแล้วสลิปริงจะถูกลัดวงจรกลายสภาพเป็นโรเตอร์แบบกรงกระรอก

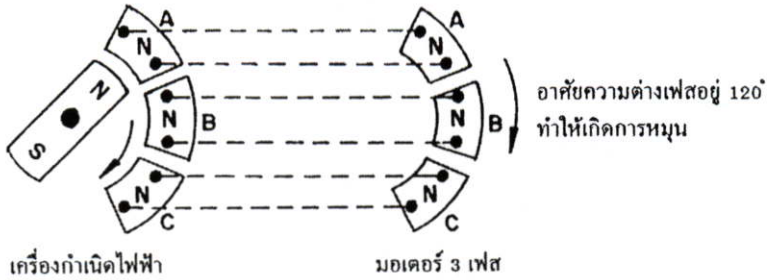


รูปที่ 2.1 แสดงการต่อความต้านทานจากภายนอกเข้ากับโรเตอร์แบบพันขดลวด

2.3 สนามแม่เหล็กหมุนของมอเตอร์เหนี่ยวนำ 3 เฟส

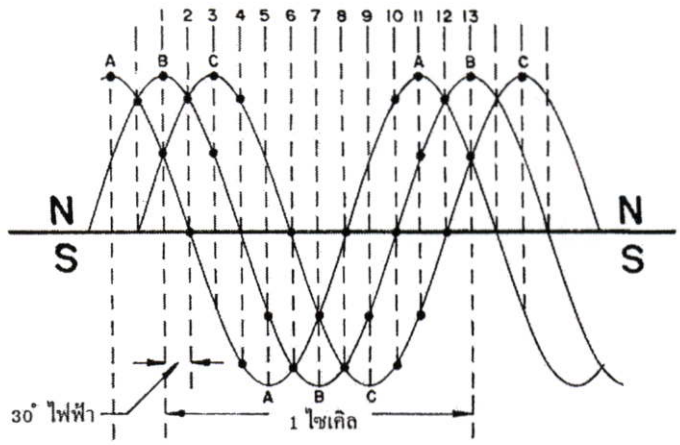
สนามแม่เหล็กของมอเตอร์เหนี่ยวนำ 3 เฟส เมื่อมีกระแสไฟฟ้าระบบ3เฟสจ่ายให้กับขดลวด 3 เฟส เป็นผลทำให้เกิดสนามแม่เหล็กหมุนในมอเตอร์นั้น สนามแม่เหล็กหมุนจะตัดกับตัวนำในโรเตอร์นั้นทำให้เกิดกระแสไฟฟ้าเหนี่ยวนำขึ้นในตัวนำที่ฝังอยู่ในโรเตอร์ และจะเกิดสนามแม่เหล็กขึ้นในโรเตอร์ เพราะที่โรเตอร์มีกระแสไฟฟ้าเหนี่ยวนำไหลอยู่ ซึ่งจะทำให้เกิดสนามแม่เหล็กเป็นขั้วเหนือและขั้วใต้เช่นเดียวกับที่สเตเตอร์ และสนามแม่เหล็กที่หมุนที่สเตเตอร์นั้นจะเกิดการผลัก และดูดกับขั้วแม่เหล็กที่เกิดขึ้นที่โรเตอร์ในทิศทางของสนามแม่เหล็กหมุน ผลที่ได้ของการดูด และผลักระหว่างขั้วแม่เหล็กบนสเตเตอร์และโรเตอร์ทำให้เกิดแรงบิดขึ้น

หลักการหมุนของสนามแม่เหล็กโดยกระแสไฟฟ้า 3เฟสจากรูปที่ 2.2 เป็นการแสดงให้เห็นว่าถ้าเราจ่ายกระแสไฟฟ้าในระบบ 3 เฟสให้กับขดลวดในสเตเตอร์ ในช่วงขณะหนึ่ง สมมติให้เป็นครึ่งไซเคิลบวกดังแสดงในรูปที่ 2.2 ด้านขวามือ โดยการต่อไฟฟ้าเฟส A เข้าการเฟส A ของมอเตอร์ และเฟส B เฟส C เข้ากับมอเตอร์ในเฟสถัดไป เมื่อกระแสไฟฟ้าในครึ่งไซเคิลบวกเฟส A ไหลเข้า

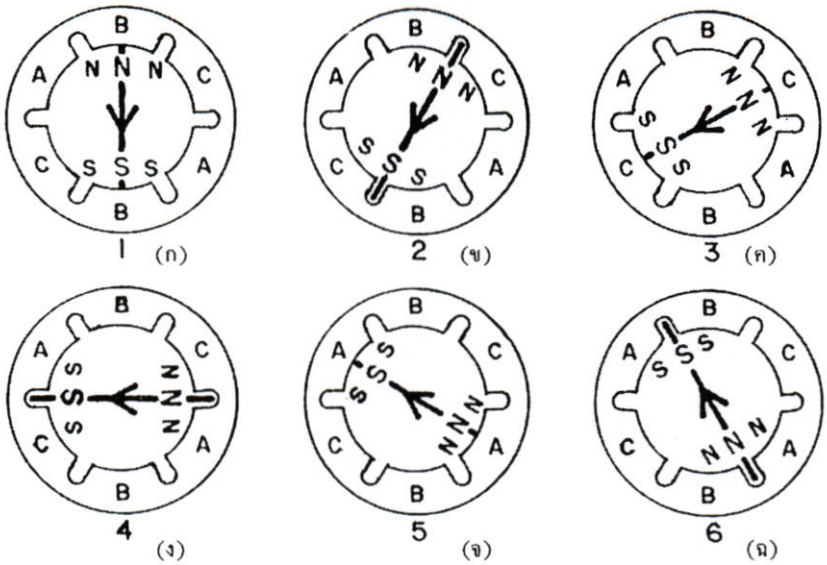


รูปที่ 2.2 แสดงการจ่ายแรงดันไฟฟ้า3เฟสให้กับมอเตอร์เหนี่ยวนำ3เฟส

ไปในขดลวดของเฟส A มอเตอร์ จะทำให้เกิดมีกระแสไฟฟ้าไหลผ่านขดลวดในเฟส A ของมอเตอร์นั้นทำให้เกิดขั้ว N ขึ้น และเมื่อแรงดันไฟฟ้าในเฟส A ที่จ่ายให้กับเฟส A มอเตอร์ค่อยๆ ลดลงอำนาจแม่เหล็กขั้ว N ก็จะค่อยๆลดอำนาจ หรือความเข้มลง และในขณะเดียวกันที่เฟสถัดไปก็จะมีอำนาจแม่เหล็กคล้ายๆกับเฟส A แต่ในเวลาถัดไปจนครบ 3 เฟสในหนึ่งขั้วแม่เหล็กของมอเตอร์ (คือเฟสA เฟส B และเฟส C) และเมื่อมีการเปลี่ยนแปลงของแรงดันไฟฟ้าในครึ่งไซเคิลบวกสิ้นสุดเรียบร้อยแล้ว ในครึ่งไซเคิลลบถัดไปที่ขั้วแม่เหล็กดังกล่าวข้างต้นก็จะเปลี่ยนสภาวะจากขั้ว N ไปเป็นขั้ว S และในอีกหนึ่งขั้วแม่เหล็กถัดไปก็มีลักษณะเช่นเดียวกันกับขั้วแม่เหล็กแรกที่กล่าวถึง ซึ่งลักษณะเช่นนี้เหมือนกับว่าสนามแม่เหล็กหมุนไปรอบๆสเตเตอร์ ซึ่งสนามแม่เหล็กที่เหมือนกับหมุนไปรอบๆนี้เรียกสั้นๆว่าสนามแม่เหล็กหมุน (Rotating Magnetic Field)



รูปที่ 2.3 แสดงรูปคลื่นไซน์ของสนามแม่เหล็กที่เกิดขึ้นโดยกระแสไฟฟ้า 3 เฟส



รูปที่ 2.4 แสดงการเกิดสนามแม่เหล็กหมุนของมอเตอร์ 3 เฟส 2 ขั้ว

จากรูปที่ 2.3 และรูปที่ 2.4 เป็นการแสดงถึงการเกิดสนามแม่เหล็กหมุนของมอเตอร์ 3 เฟส 2 ขั้ว ที่สามารถพิจารณาทีละขั้นได้คือ

ที่จุดที่1 บนรูปไซเคิลไซน์จะเห็นได้ว่าเฟส B อยู่ที่ตำแหน่งความเข้มสูงสุดของขั้วเหนือ เฟส A ก็เป็นขั้วเหนือ แต่ลดลงจากจุดสูงสุดและเฟส C เป็นขั้วเหนือและกำลังเพิ่มขึ้น ดังแสดงในรูปที่ 2.4 (ก) ของสเตเตอร์คือเฟส B เป็นขั้วเหนือมาก เฟส A และเฟส C เป็นขั้วเหนือน้อย

ที่จุดที่2 บนรูปคลื่นไซน์ เฟส A เป็น 0 (Zero) เฟส B เป็นขั้วเหนือแต่กำลังลดลง ส่วนเฟส C เป็นขั้วเหนือแต่กำลังเพิ่มขึ้นดังแสดงในรูปที่ 2.4 (ข) ของสเตเตอร์ เฟส A เป็น 0 เฟส B และเฟส C เป็นขั้วเหนือและมีความเข้มเท่ากัน ซึ่งจะเกิดขึ้นระหว่าง 2เฟสคือ เฟส B กับเฟส C ดังนั้นจากจุดที่ 1 ไปจุดที่ 2 ขั้วเหนือจะเคลื่อนที่ไป 30 องศาไฟฟ้า ในทิศทางตามเข็มนาฬิกา (30 Electrical Degree Clockwise)

ที่จุดที่ 3 บนรูปคลื่นไซน์ เฟส C จะมีความเข้มสูงสุดของขั้วเหนือ เฟส B ลดลงจากสูงสุด และเฟส A เพิ่มขึ้นแต่ เฟส A เป็นขั้วใต้ ดังแสดงในรูปที่ 2.4 (ค) ของสเตเตอร์เฟส C จะเป็นขั้วเหนือและมีความเข้มมาก ส่วนเฟส B มีความเข้มของขั้วเหนือน้อย และเฟส A เป็นขั้วใต้น้อย ดังนั้นจากจุดที่ 2 ไปจุดที่ 3 ขั้วเหนือจะเคลื่อนที่ไป 30 องศาไฟฟ้า ในทิศทางตามเข็มนาฬิกา

บนรูปคลื่นไซน์ที่จุดต่างๆในไซเคิลก็จะเกิดหมุนเวียนกันไปเช่นเดียวกับที่กล่าวมาแล้ว ตัวอย่างเช่น ที่จุดที่ 6 ในรูปที่ 2.3 บนรูปคลื่นไซน์ เฟส A และ เฟส B จะเป็นขั้วใต้ ส่วนเฟส C จะเป็น 0 ดังแสดงในรูปที่ 2.4 (ฉ) ของสเตเตอร์ขั้วเหนือจะเคลื่อนที่ไป 150องศาไฟฟ้าในทิศทางตามเข็มนาฬิกาจากจุดที่ 1 เป็นอันว่าครบไซเคิลของขั้วเหนือ หรือหมุนครบ 1 รอบ 360 องศา

2.4 สลิป (Slip, S)

ในทางปฏิบัตินั้นโรเตอร์ไม่สามารถหมุนได้เท่ากับความเร็วของสนามแม่เหล็กหมุนที่สเตเตอร์ โดยปกติแล้วความเร็วของโรเตอร์จะมีความเร็วน้อยกว่าความเร็วของสนามแม่เหล็กที่หมุนที่สเตเตอร์ ความแตกต่างของความเร็วนั้นจะขึ้นอยู่กับโหลดที่ต่ออยู่กับมอเตอร์นั้น

ความแตกต่างระหว่างความเร็วของสนามแม่เหล็กที่หมุนอยู่ที่สเตเตอร์ หรือความเร็วซิงโครนัส(N_S) และความเร็วรอบของโรเตอร์ขณะใช้งาน (Actual Speed : N) ของโรเตอร์เรียกว่าสลิป สลิปของมอเตอร์โดยปกติเป็นเปอร์เซ็นต์ ซึ่งเปอร์เซ็นต์สลิปสามารถหาได้จากสมการดังนี้คือ

$$S = \frac{N_S - N}{N_S} \tag{2.2}$$

$$\%Slip = \frac{N_S - N}{N_S} \times 100$$

แต่ในบางครั้ง $N_S - N$ ก็เรียกว่า ความเร็วสลิป (Slip Speed)

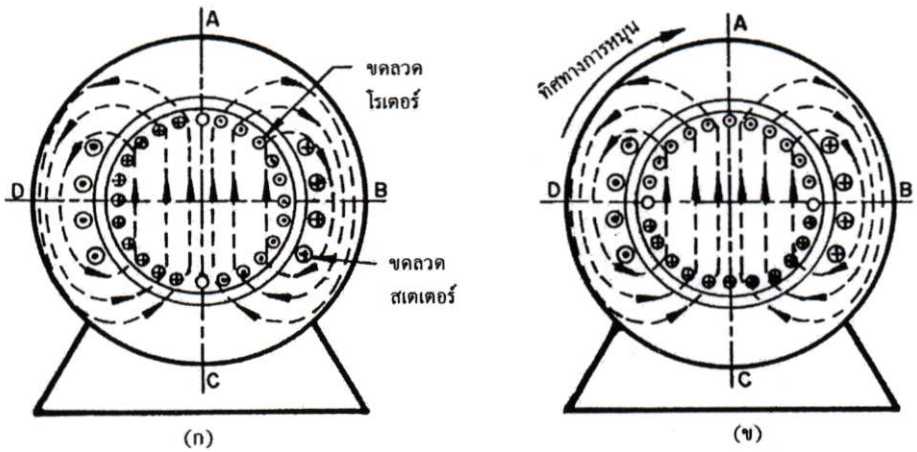
จะเห็นได้ว่าความเร็วของโรเตอร์ (มอเตอร์) คือ $N = N_S(1-S)$

2.5 มอเตอร์เหนี่ยวนำกระแสสลับ 1เฟส

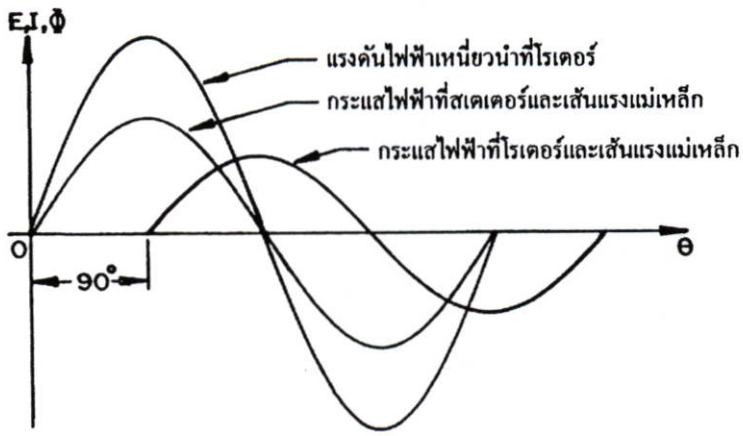
มอเตอร์เหนี่ยวนำกระแสสลับแบบหนึ่งเฟสมีโครงสร้างคล้ายกับมอเตอร์3เฟส แต่มีข้อแตกต่างกันตรงที่สเตเตอร์ของแบบ 1เฟสมีการพันขดลวดในลักษณะแบบเฟสเดียว และต่อเข้ากับแหล่งจ่ายไฟฟ้ากระแสสลับเพียง 1เฟส ดังนั้นสนามแม่เหล็กที่เกิดขึ้นที่สเตเตอร์จึงไม่ใช่สนามแม่เหล็กที่หมุนด้วยความเร็วซิงโครนัสเหมือนกับในกรณีของมอเตอร์ 2เฟส หรือ 3เฟสที่ได้รับแรงดันไฟฟ้ากระแสสลับ 2เฟส หรือ 3เฟส จึงเป็นเหตุให้สนามแม่เหล็กที่เกิดการกลับไปกลับมาอยู่ที่สเตเตอร์นั้นไม่สามารถทำให้มอเตอร์เกิดแรงบิด และหมุนขึ้นได้ในขณะที่โรเตอร์นั้นยังหยุดอยู่ ซึ่งเป็นสาเหตุว่ามอเตอร์เหนี่ยวนำ 1เฟสไม่สามารถเริ่มหมุนได้ด้วยตัวมันเอง แต่จะหมุนได้ก็ต่อเมื่อโรเตอร์ถูกทำให้หมุนด้วยกรรมวิธีใดวิธีหนึ่ง ซึ่งหมายถึงต้องทำให้เริ่มหมุนด้วยมือหรืออุปกรณ์อื่นๆ ก่อน และเมื่อมอเตอร์ถูกช่วยทำให้หมุนไปทิศทางใดทิศทางหนึ่งก่อนแล้ว จะทำให้เกิดแรงบิดและอัตราเร่งขึ้นในโรเตอร์นั้นจนกระทั่งได้ความเร็วเต็มพิกัดของมัน

2.6 ทฤษฎีสนามแม่เหล็กขวางของมอเตอร์เหนี่ยวนำ 1เฟส

จากรูปที่ 2.5(ก) ถ้าโรเตอร์อยู่ในสนามแม่เหล็กที่เกิดขึ้นเนื่องจากไฟฟ้ากระแสสลับ จะทำให้เกิดแรงดันไฟฟ้าเหนี่ยวนำขึ้นในโรเตอร์ (ลักษณะคล้ายกับกรณีของหม้อแปลงไฟฟ้า) พร้อมกับส่งผลให้เกิดกระแสไฟฟ้าไหลในโรเตอร์และเกิดสนามแม่เหล็กในโรเตอร์ โดยที่สนามแม่เหล็กที่เกิดขึ้นจากการเหนี่ยวนำนี้จะมีทิศทางตรงกันข้ามกับสนามแม่เหล็กที่สเตเตอร์และมีค่าเท่ากัน ผลลัพธ์ที่เกิดขึ้นที่โรเตอร์นั้นคือโรเตอร์จะไม่หมุน เนื่องจากการหักล้างกันระหว่างสนามแม่เหล็กที่สเตเตอร์กับโรเตอร์ ดังนั้นจึงไม่เกิดการหมุนหรือไม่เกิดแรงบิดเริ่มหมุนขึ้นในมอเตอร์ แต่แรงบิดจะเกิดขึ้นในโรเตอร์และโรเตอร์หมุนได้ โดยการที่ตัวนำ ในโรเตอร์ตัดกับสนามแม่เหล็กซึ่งทำให้เกิดแรงดันไฟฟ้าเหนี่ยวนำขึ้น กระแสไฟฟ้าจะไหลอยู่ในแท่งตัวนำในโรเตอร์ ถ้าตัวนำที่โรเตอร์ถูกทำให้หมุนไปจากตำแหน่ง A ไปยังตำแหน่ง B ซึ่งทำมุมฉากกับตำแหน่งเดิม ตัวนำโรเตอร์จะเกิดการตัดผ่านสนามแม่เหล็กที่สเตเตอร์ดังแสดงในรูปที่ 2.5(ข) ในขณะเดียวกันสนามแม่เหล็กก็จะแยกออกเป็น 2 ระบบคือต่างเฟสกันอยู่ 90 องศา และกระแสไฟฟ้าในสเตเตอร์และโรเตอร์ก็จะมีเฟสที่ต่างกัน อย่างไรก็ตามในความเป็นจริงแล้วกระแสไฟฟ้าที่โรเตอร์จะต่างเฟสกับกระแสไฟฟ้าที่สเตเตอร์ เพราะเนื่องจากว่าที่โรเตอร์นั้นความถี่เปลี่ยนไป โดยที่ความถี่ของกระแสไฟฟ้าที่โรเตอร์จะขึ้นอยู่กับความเร็วรอบของโรเตอร์ ซึ่งผลลัพธ์ที่ได้ของเส้นแรงแม่เหล็กทั้งสองนี้แสดงได้ดังรูปที่ 2.6



รูปที่ 2.5 แสดงทิศทางของการเกิดของกระแสไฟฟ้าและเส้นแรงแม่เหล็กในโรเตอร์ของมอเตอร์เหนี่ยวนำเฟสเดียว



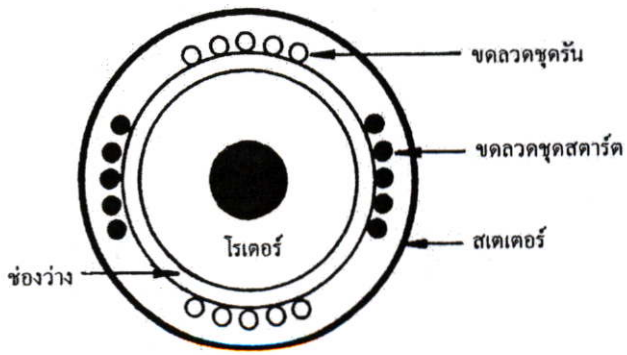
รูปที่ 2.6 แสดงรูปคลื่นของกระแสไฟฟ้าที่สเตเตอร์และโรเตอร์ของมอเตอร์เหนี่ยวนำ

มอเตอร์เหนี่ยวนำกระแสสลับชนิด เฟสเดียว เป็นมอเตอร์ชนิดทำงานโดยอาศัยแหล่งจ่ายไฟฟ้ากระแสสลับเฟสเดียว ซึ่งสามารถแบ่งได้เป็น 2 แบบ

ก. เครื่องกลชนิดกรงกระรอก ได้แก่ สปลิตเฟสมอเตอร์ คาปาซิเตอร์มอเตอร์ และเชดเดดโพลมอเตอร์(Shaded Pole Motor)

ข. เครื่องกลชนิดโรเตอร์พันขดลวด ได้แก่ ซีรีส์มอเตอร์ และรีพัลชั่นมอเตอร์

มอเตอร์เหนี่ยวนำเฟสเดียวจะประกอบด้วยขดลวด 2 ชุดคือ ขดลวดชุดสตาร์ตหรือขดลวดช่วย(Starting Winding or Auxiliary Winding) และขดลวดชุดรันหรือขดลวดหลัก(Running Winding or Main Winding) ขดลวดทั้งสองวางทำมุมกัน 90 องศาทางไฟฟ้า และต่อคร่อมเข้ากับแหล่งจ่ายไฟฟ้าเฟสเดียว ลักษณะตำแหน่งของขดลวดแสดงในรูปที่ 2.7



รูปที่ 2.7 ตำแหน่งการวางขดลวดของมอเตอร์เหนี่ยวนำเฟสเดียว

มอเตอร์เหนี่ยวนำเฟสเดียวนี้ อาศัยความต่างเฟสระหว่างกระแสไฟฟ้าที่ไหลในขดลวดทั้งสองบนสเตเตอร์ที่วางทำมุมกัน 90 องศา ดังนั้นมอเตอร์จะมีคุณสมบัติหรืออาการเช่นเดียวกับมอเตอร์ 2 เฟสกระแสไฟฟ้าที่ไหลในขดลวดทั้งสองจะทำให้เกิดสนามแม่เหล็กหมุน จึงทำให้มอเตอร์สามารถเริ่มหมุนด้วยตัวเองได้

2.7 คาปาซิเตอร์สตาร์ตและรันมอเตอร์ (Capacitor Start and Run Motor)

มอเตอร์ชนิดนี้เหมือนกันกับมอเตอร์แบบคาปาซิเตอร์สตาร์ตมอเตอร์ โดยจะมีคาปาซิเตอร์หรือตัวเก็บประจุต่ออนุกรมอยู่กับขดลวดชุดสตาร์ตตลอดเวลา ข้อดีของการต่อตัวเก็บประจุไว้อย่างถาวร ได้แก่

ก. เป็นการปรับหรือช่วยให้มอเตอร์ทำงานเกินกว่าโหลดปกติ

ข. ให้เพาเวอร์แฟกเตอร์ที่สูงกว่า

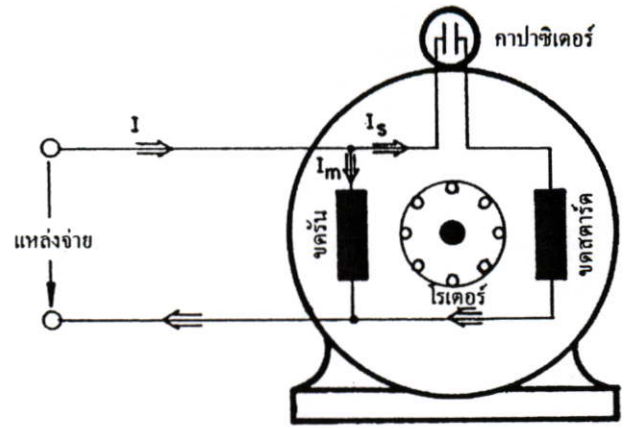
ค. เกิดประสิทธิภาพที่สูงกว่า

ง. ในขณะที่มอเตอร์หมุนอยู่จะไม่กำเนิดเสียงที่ดังมาก

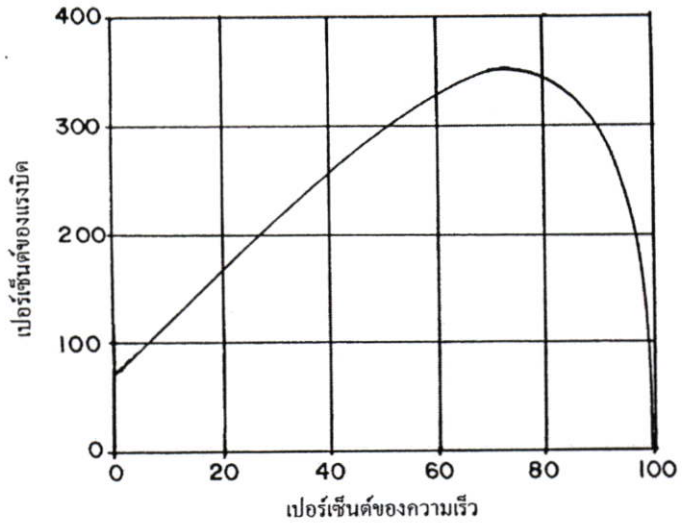
ในกรณีที่มอเตอร์ที่ใช้ตัวเก็บประจุทั้งในขดสตาร์ตและรันในตัวเดียวกันนี้เรียกว่าคาปาซิเตอร์รันมอเตอร์ชนิดค่าเดียว (Single Value Capacitor Run Motor) แต่ในกรณีที่มอเตอร์ใช้ตัวเก็บประจุ 2 ตัวโดยที่ขดสตาร์ตใช้ตัวเก็บประจุที่มีค่าความจุสูง แต่สำหรับเวลาที่หมุนปกติใช้ตัวเก็บประจุที่มีค่าความจุต่ำ โดยมอเตอร์แบบนี้ถูกเรียกว่าคาปาซิเตอร์มอเตอร์ชนิด 2 ค่า (Two Value Capacitor Run Motor)

2.8 คาปาซิเตอร์รันมอเตอร์ชนิดค่าเดียว

มอเตอร์ชนิดนี้จะมีขดรันและขดสตาร์ตอย่างละชุด โดยขดลวดชุดสตาร์ตจะมีตัวเก็บประจุต่ออนุกรมอยู่ดังแสดงในรูปที่ 2.8 โดยตัวเก็บประจุจะต่ออยู่กับวงจรขดลวดชุดสตาร์ตตลอดเวลา และจะไม่มีสวิตช์แรงเหวี่ยงหนีศูนย์ และตัวเก็บประจุนี้จะทำหน้าที่ทั้งสตาร์ตและรันต่อเนื่องกันไป ขนาดของความจุของตัวเก็บประจุที่ใช้จะมีค่าประมาณ 2-20 ไมโครฟารัด ซึ่งเป็นแบบน้ำมันหรือ แบบกระดาษ ซึ่งคุณลักษณะของมอเตอร์ดังแสดงในรูปที่ 2.9

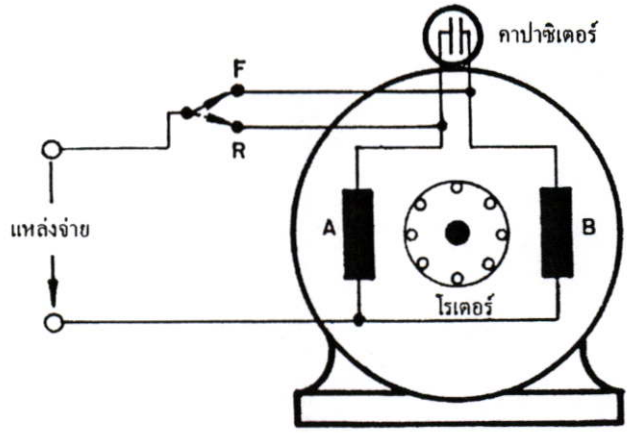


รูปที่ 2.8 แสดงวงจรของคาปาซิเตอร์รันมอเตอร์ชนิดค่าเดียว



รูปที่ 2.9 แสดงคุณลักษณะระหว่างแรงบิดกับเปอร์เซนต์ความเร็วของคาปาซิเตอร์รันมอเตอร์ชนิดค่าเดียว

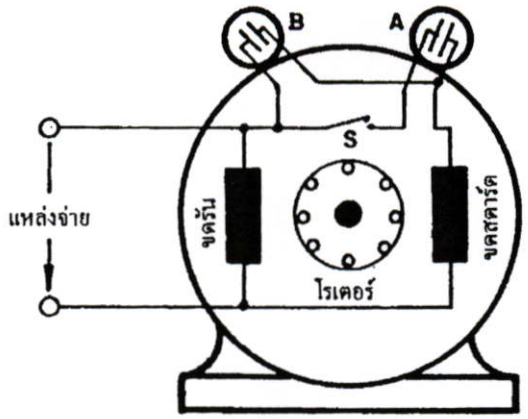
คาปาซิเตอร์รันมอเตอร์ชนิดค่าเดียวนี้สามารถที่จะทำให้กลับทางหมุนได้ง่าย ด้วยการใช้สวิตช์จากภายนอก แต่ขดรันและขดสตาร์ตจะต้องพันด้วยลวดขนาดเดียวกัน โดยมันจะผลัดกันทำหน้าที่เป็นขดรันและขดสตาร์ต เมื่อต้องการให้หมุนในทิศทางตามเข็มนาฬิกาก็ปรับสวิตช์ไปที่ตำแหน่ง F ขดลวดชุด B จะทำหน้าที่เป็นขดรัน ส่วนขด A จะทำหน้าที่เป็นขดสตาร์ต แต่ถ้าต้องการให้หมุนในทิศทางทวนเข็มนาฬิกาก็ปรับสวิตช์ไปที่ตำแหน่ง R ขดลวดชุด A จะทำหน้าที่เป็นขดรัน และขดลวด B จะทำหน้าที่เป็นขดสตาร์ต ดังแสดงในรูปที่ 2.10



รูปที่ 2.10 แสดงวงจรการกลับทิศทางการหมุนของคาปาซิเตอร์รันมอเตอร์ชนิดค่าเดียว

2.9 คาปาซิเตอร์รันมอเตอร์ชนิด 2 ค่า

คาปาซิเตอร์มอเตอร์ชนิดนี้จะใช้คาปาซิเตอร์ที่มีความจุมากต่ออนุกรมกับขดสตาร์ตใช้ทำหน้าที่สตาร์ต แต่ขณะหมุนปกติจะใช้คาปาซิเตอร์ที่มีความจุต่ำ โดยคาปาซิเตอร์ที่มีความจุต่ำนี้จะต่อคร่อมเข้ากับสวิตช์แรงเหวี่ยงหนีศูนย์ และคาปาซิเตอร์สตาร์ตดังในรูปที่ 2.11 คาปาซิเตอร์สตาร์ตคือคาปาซิเตอร์ A จะเป็นชนิดอิเล็กโตรไลต์มีค่าความจุสูง (ทำงานช่วงเวลาสั้น) และคาปาซิเตอร์ B เป็นแบบน้ำมันมีค่าความจุต่ำ(ทำงานต่อเนื่อง) โดยทั่วไปคาปาซิเตอร์ A จะมีค่าความจุประมาณ 10-15เท่าของคาปาซิเตอร์ B ขณะที่เริ่มสตาร์ตนั้นสวิตช์แรงเหวี่ยงหนีศูนย์จะคงปิดวงจรอยู่ทำให้คาปาซิเตอร์ทั้งสองต่อขนานกันอยู่ ดังนั้นค่าความจุรวมจะมาก หลังจากที่มอเตอร์หมุนไปประมาณ 75 เปอร์เซ็นต์ของความเร็วสูงสุดแล้ว สวิตช์แรงเหวี่ยงหนีศูนย์จะเปิดวงจรออก ยังคงเหลือคาปาซิเตอร์B ต่ออยู่กับวงจรของขดลวดชุดสตาร์ตเพียงตัวเดียว ซึ่งจะทำให้ขดลวดชุดสตาร์ตกลายเป็นขดลวดชุดรันด้วยอีกชุดหนึ่ง มอเตอร์ชนิดนี้จะสามารถสตาร์ตในขณะที่มีโหลดมากๆได้ และขณะทำงานจะเงียบทำให้ประสิทธิภาพสูง และมีเพาเวอร์แฟกเตอร์สูงด้วย



รูปที่ 2.11 แสดงวงจรของคาปาซิเตอร์รันมอเตอร์ชนิด 2 ค่า

2.10 การควบคุมอินเวอร์แบบ PWM (Pulse Width Modulate)

2.10.1 การเฉลี่ยเฉพาะที่

แรงดันเอาท์พุท V_o ของวงจรทอนระดับสัญญาณดังรูปที่ 2.12(ก)สามารถ เปลี่ยนแปลงได้ระหว่างค่า 0 ถึงค่า V_s โดยที่วัฏจักรงาน D มีค่าระหว่าง 0 ถึง 1 ถ้าเราให้ D เป็นฟังก์ชั่นของเวลา $d(t)$ โดยที่ $d(t)$ มีการเปลี่ยนแปลงค่าอย่างช้าๆ เมื่อเทียบกับคาบของการสวิตช์ ซึ่งถ้าเป็นในกรณีเช่นนี้เราสามารถสังเคราะห์แรงดันเอาท์พุท V_o ซึ่งมีค่าเฉลี่ยจะเป็นฟังก์ชั่นของเวลา และมีค่าเท่ากับ $d(t)\,V_s$ อย่างไรก็ตามเวลาในการเฉลี่ยค่า V_o จำเป็นต้องเป็นช่วงเวลาที่นานเมื่อเทียบกับคาบเวลาของการสวิตช์ T แต่คาบเวลาดังกล่าวจะต้องสั้นกว่าเมื่อเทียบกับคาบเวลาของ $d(t)$

การเฉลี่ยค่าแรงดันเอาท์พุท V_o ในที่นี้หมายถึงการเฉลี่ยเฉพาะที่ (local averaging) และจะใช้สัญลักษณ์ $\overline{V_o}(t)$ สำหรับสัญลักษณ์ $<V_o>$ หมายถึงค่าเฉลี่ยจริงซึ่งเป็นค่าคงตัว หรืออาจกล่าวอีกนัยหนึ่งได้ว่า ค่าเฉลี่ยดังกล่าวเป็นค่าเฉลี่ยเฉพาะที่ได้จากการกรองสัญญาณแรงดันเอาท์พุทด้วยวงจรกรองความถี่ต่ำผ่าน โดยขจัดความถี่สูงออกและคงเหลือไว้แต่สัญญาณความถี่ต่ำๆ เช่น องค์ประกอบหลักมูล (fundamental component) ในขณะที่ถ้าส่วนค่าเฉลี่ยได้จากการกรองทุกความถี่ออก จะคงเหลือไว้แต่องค์ประกอบไฟตรง

สำหรับวงจรทอนระดับสัญญาณแสดงในรูปที่ 2.12(ก) ถ้าเราให้วัฏจักรงาน $d(t)$ เป็นฟังก์ชั่นไซนูซอยด์ เราจะได้แรงดันเอาท์พุทที่เป็นองค์ประกอบไฟตรงบวกกับไซนูซอยด์ ได้ดังนี้

ถ้า $\quad d(t) = 0.5 + m_a \sin \omega_a t$

$$V_o(t) = 0.5V_s + m_a V_s \sin \omega_a t \quad ; m_a \le 0.5 \qquad (2.3)$$

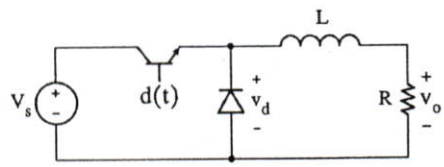
m_a คือ amplitude modulation

ω_a คือ frequency modulation

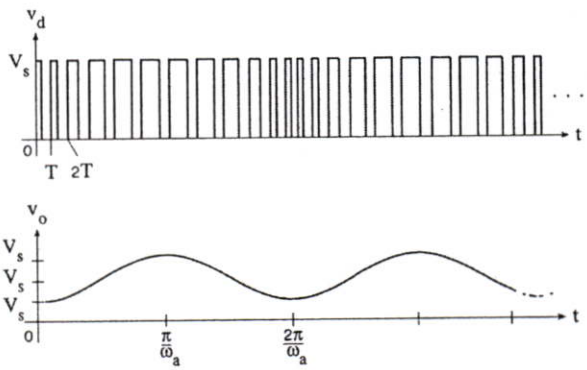
โดยมีเงื่อนไขดังนี้

$$T \ll \frac{L}{R} \ll \frac{2\pi}{\omega_a} \tag{2.4}$$

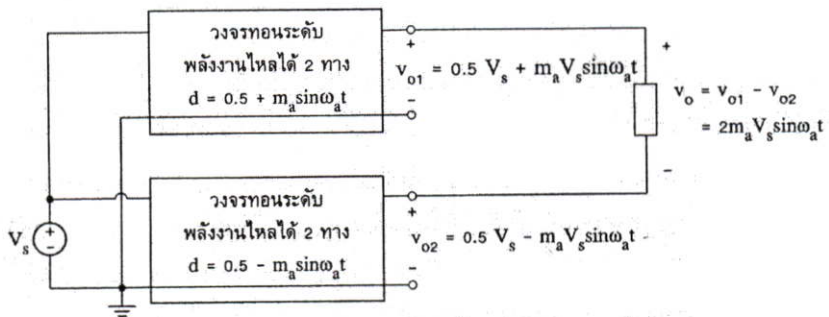
แรงดัน V_d เป็นพัลส์ที่มีความถี่เท่ากับ $1/T$ แต่มีความกว้างพัลส์ที่ไม่คงตัว เราเรียกรูปคลื่นที่วัฏจักรงานเป็นฟังก์ชั่นของเวลาว่ารูปคลื่นสัญญาณ PWM ค่าเฉลี่ยเฉพาะที่ของรูปคลื่น PWM หรือ $\bar{V}_d$ จะเป็นฟังก์ชั่นของเวลา ถ้าเราทำการกรอง V_d ด้วยวงจรกรองความถี่ต่ำผ่านที่ประกอบไปด้วยอุปกรณ์ L และ R โดยมีพารามิเตอร์เป็นไปตามเงื่อนไขสมการที่ (2.4) ดังนั้นค่าความถี่การสวิตช์จะถูกกรองออกไปแต่ความถี่ ω_a ไม่ถูกลดทอน



(ก) วงจร



(ข) รูปคลื่น : $\bar{v}_d = v_o$ (ถ้า $T \ll L/R \ll 2\pi/\omega_a$)



(ค) การต่อวงจรทอนระดับ 2 วงจร เพื่อกำจัดองค์ประกอบไฟตรง

รูปที่ 2.12 การใช้เทคนิค PWM กับวงจรทอนระดับ

ในรูปที่ 2.12(ข) V_o คือค่าเฉลี่ยเฉพาะที่ของรูปคลื่น PWM เนื่องจาก $d(t)$ เป็นฟังก์ชั่นไซนูซอยด์ ค่าเฉลี่ยเฉพาะที่ของรูปคลื่น PWM จึงเป็นรูปไซนูซอยด์ด้วย แต่ $d(t)$ อาจเป็นฟังก์ชั่นของเวลาใดๆ (ที่เป็นไปตามเงื่อนไขสมการที่ (2.4)) $V_o(t)$ ก็จะเป็นฟังก์ชั่นที่เหมือนกับ $d(t)$

อย่างไรก็ดี แรงดันเอาท์พุท V_o ของวงจรทอนระดับสัญญาณยังมีองค์ประกอบไฟตรงผสมอยู่ (เป็นไปตามสมการที่ (2.3) และรูปที่ 2.12 (ข)) ดังนั้นการกำจัดองค์ประกอบไฟตรงสามารถทำได้โดยการต่อวงจรทอนระดับ 2 วงจร โดยที่สัญญาณมอดดูเลตของวงจรทั้งสองมีเฟสต่างกัน 180^o (ดังแสดงในรูปที่ 2.12 (ค)) เมื่อนำสัญญาณด้านเอาท์พุทมาลบกับ จะส่งผลให้แรงดันไฟตรงก็จะหักล้างกันไปเหลือแต่แรงดันไฟสลับ อย่างไรก็ดี เนื่องจากกระแสด้านออก i_o เป็นกระแสสลับวงจรทอนระดับที่ใช้จะต้องเป็นชนิดที่กำลังงานไหลได้ 2 ทิศทาง กล่าวคือ แต่ละวงจรใช้สวิตช์ ที่กระแสไหลได้ 2 ทางจำนวน 2 ตัวถ้าให้สัญญาณมอดดูเลตเป็นฟังก์ชั่นใดๆ แต่องค์ประกอบความถี่สูงสุดยังเป็นไปตามเงื่อนไขสมการที่ (2.4) แรงดันด้านออกจะเป็นฟังก์ชั่นเดียวกับสัญญาณมอดดูเลต เพียงแต่จ่ายกำลังได้มากขึ้น ในแง่นี้วงจรทำหน้าที่เป็นวงจรขยายกำลังแบบวีธีสวิตช์ (Switch mode power amplifier)

$$m_f = \frac{f}{f_a} \tag{2.5}$$

m_f = อัตราการมอดดูเลตความถี่

f = ความถี่การสวิตช์

f_a = ความถี่การมอดดูเลต

2.11 อินเวอร์เตอร์ 3 เฟส

2.11.1 วงจรและรูปคลื่นสัญญาณ

อินเวอร์เตอร์ 3 เฟสเหมาะสำหรับกรณีที่ต้องการกำลังไฟฟ้าสูง ซึ่งโครงสร้าง และรูปคลื่นสัญญาณของวงจรอินเวอร์เตอร์ 3 เฟสแสดงอยู่ในรูปที่ 2.13 (ก)

เทคนิค PWM สามารถใช้ได้กับวงจรอินเวอร์เตอร์ 3 เฟส ไม่ว่าจะเป็นเทคนิคไซน์ตัดสามเหลี่ยม หรือเทคนิคการกำจัดฮาร์มอนิกต่ำ หรือการควบคุมรูปคลื่นของกระแส

รูปที่ 2.13(ข) แสดงหลักการของเทคนิคไซน์ตัดสามเหลี่ยม เรามีรูปสามเหลี่ยม V_T ซึ่งมีความถี่เท่ากับความถี่การสวิตช์ และมีสัญญาณอ้างอิงรูปไซน์ 3 สัญญาณได้แก่คือ V_{rA}, V_{rB} และ V_{rC} ซึ่งมีมุมเฟสต่างกัน 120^o โดยมีจุดตัดระหว่าง V_T กับ V_{rA} (V_{rB} และ V_{rC}) เป็นตัวกำหนดการตัดต่อวงจรของสวิตช์ Q_1 Q_4 (Q_3 Q_6 และ Q_5 Q_2 ตามลำดับ) รูปคลื่นของแรงดัน V_{AG}, V_{BG} (G คือสายลบของแหล่งแรงดันไฟตรง) และ V_{AB} แสดงอยู่ในรูปที่ 2.13(ค) สังเกตได้ว่าองค์ประกอบหลัก

มูลของ V_{AG} หรือ V_{AG1} มีแอมพลิจูดเท่ากับค่าที่แสดงในสมการที่(2.6) (V_{AG} ต่างกับ V_{AO} เพียงค่าแรงดันไฟตรง $V_s/2$)

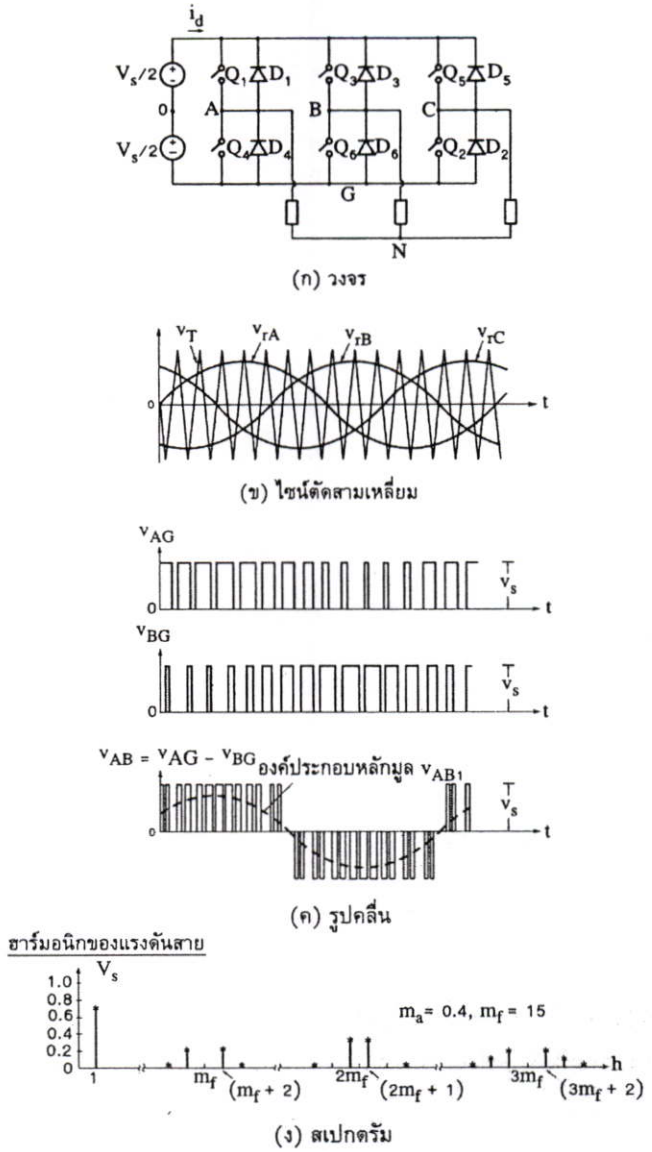
$$V_{AG1} = m_a V_s \qquad ;m_a \le 0.5 \qquad (2.6)$$

แรงดันสาย V_{AB} มีองค์ประกอบหลักมูลเท่ากับ V_{AB1} แอมพลิจูดของ V_{AB1} เท่ากับ $\sqrt{3}$ ของแอมพลิจูดของ V_{AG1} เนื่องจาก V_{AG} และ V_{BG} มีเฟสต่างกัน 120° และ V_{AB1} มีเฟสล้าหลัง V_{AG1} อยู่ 30° (แสดงในรูปที่ 2.13(ค)) ดังนั้น

$$V_{AB1} = \sqrt{3} m_a V_s \qquad ;m_a \le 0.5 \qquad (2.7)$$

ในการออกแบบระบบ โดยทั่วไปนิยมเลือกอัตราการมอดดูเลตความถี่ m_f เป็นเลขคี่และเป็นพหุคูณของ 3 และให้ความลาดชันของ V_T และ V_r ณ จุดที่แรงดันทั้งสองมีค่าเป็นศูนย์มีเครื่องหมายตรงกันข้ามกัน(แสดงในรูปที่ 2.13 (ข)) ในกรณีนี้ถ้า $m_a \le 0.5$ สเปกตรัมของ V_{AB} จะเป็นดังแสดงในรูปที่2.13 (ง) นอกจากนี้สังเกตได้ว่าฮาร์มอนิกต่ำสุดเกิดขึ้นที่ความถี่ $(m_f \pm 2k) f_a$ ตารางที่ 2.1 แสดงฮาร์มอนิกของแรงดันสายของอินเวอร์เตอร์ 3เฟส สำหรับอินเวอร์เตอร์ 3เฟสเราอาจจะออกแบบให้วงจรทำงานถึงในย่านการมอดดูเลตเกิน เพื่อให้ได้แรงดันเอาท์พุทที่มีค่าสูงขึ้น ส่วนข้อเสียที่เนื่องมาแต่ฮาร์มอนิกที่เพิ่มขึ้นมักจะพอยอมรับได้

ถ้าโหลดมีลักษณะเป็นตัวเหนี่ยวนำโดยมีค่าคงตัวเวลาที่เป็นไปตามเงื่อนไขสมการที่ (2.4) กระแสโหลดจะเป็นรูปไกล้เคียงไซน์โดย ทิศทางของกระแสโหลดเป็นตัวกำหนดการนำกระแสของไดโอด เช่น กรณีที่เฟสของโหลดเท่ากับ 30° (ตามล้าหลัง) รูปคลื่นจะเป็นดังแสดงในรูปที่ 2.7 โดยสังเกตได้ว่าในช่วงเวลาที่ $i_c > 0$ และ $V_{CG} = 0$ (Q_2 หรือ D_2 นำกระแส) แต่ทิศทางของ i_c แสดงว่า D_2 นำกระแส (แสดงในรูปที่2.13 (ก))

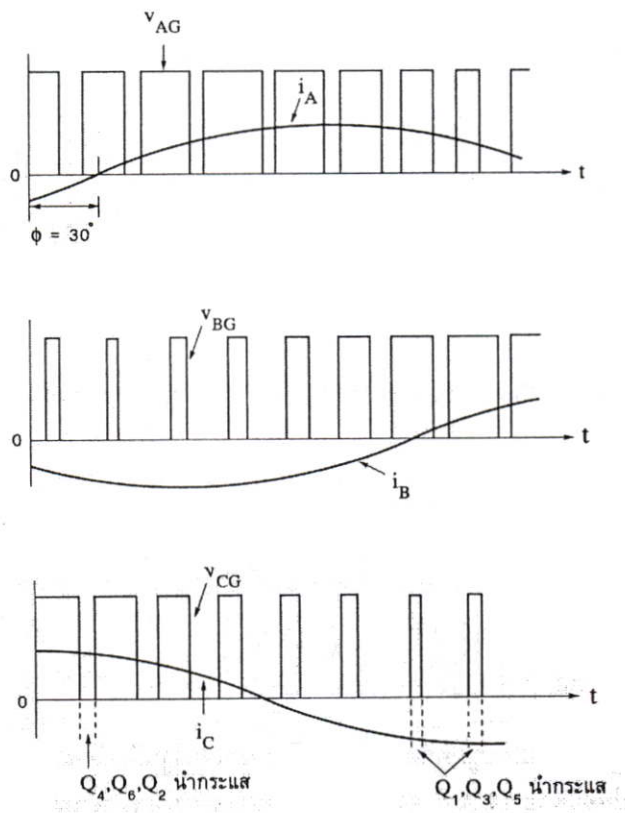


รูปที่ 2.13 วงจรและรูปคลื่นสัญญาณของอินเวอร์เตอร์ 3 เฟสแบบ PWM

เมื่อพิจารณารูปคลื่นสัญญาณในรูปที่ 2.13 จะพบว่า ในบางช่วงเวลาดังแสดงในรูปล่าง สวิตช์จะต่อกิ่งทั้งสามของโหลดเข้ากับขั้วลบของแหล่งจ่าย V_s (เช่น เมื่อ Q_4,Q_6,Q_2 นำกระแส) และบางช่วงเวลาสวิตช์จะต่อกิ่งทั้งสามของโหลดเข้ากับขั้วบวกของแหล่ง V_s (เช่น เมื่อ Q_1,Q_3,Q_6 นำกระแส) ซึ่งในช่วงเวลาดังกล่าว กระแสอินพุทของอินเวอร์เตอร์ (i_d ในรูปที่ 2.13(ก)) มีค่าเป็นศูนย์ หมายความว่ากำลังงานไม่ไหลจากแหล่งจ่ายสู่โหลด ซึ่งระยะเวลาของการควบคุมช่วงเวลาดังกล่าวเป็นการควบคุมกำลังงานที่จ่ายให้แก่โหลด หรือควบคุมค่าองค์ประกอบหลักมูลของแรงดันเอาท์พุทนั่นเอง

ตารางที่ 2.1 อัตราส่วนระหว่างค่า RMS ของฮาร์มอนิกกับ V_S ของแรงดันสายของอินเวอร์เตอร์ 3 เฟส

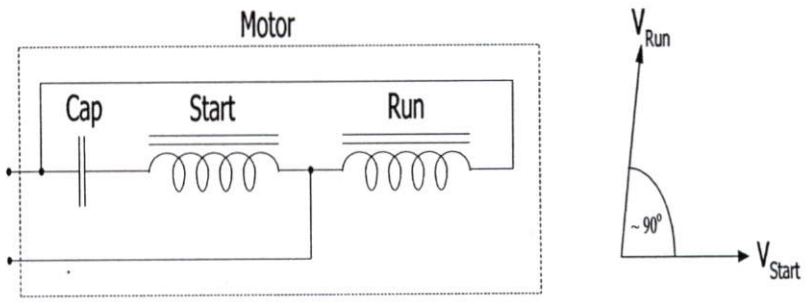
h	m_a	0.1	0.2	0.3	0.4	0.5
1		0.122	0.245	0.367	0.490	0.612
$m_f \pm 2$		0.010	0.037	0.080	0.135	0.195
$m_f \pm 4$					0.005	0.011
$2m_f \pm 1$		0.116	0.200	0.227	0.192	0.111
$2m_f \pm 5$					0.008	0.020
$3m_f \pm 2$		0.027	0.085	0.124	0.108	0.038
$3m_f \pm 4$			0.007	0.029	0.064	0.096
$4m_f \pm 1$		0.100	0.096	0.005	0.064	0.042
$4m_f \pm 5$				0.021	0.051	0.073
$4m_f \pm 7$					0.010	0.030



รูปที่ 2.14 รูปคลื่นของแรงดัน และกระแสของอินเวอร์เตอร์ 3 เฟส แสดงช่วงกำลังงานไม่ไหลสู่โหลด

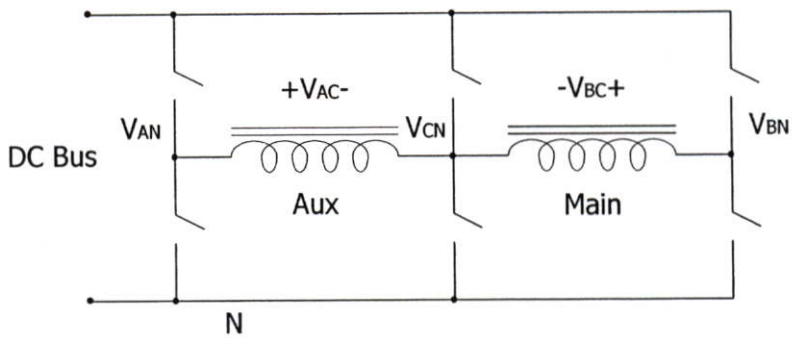
2.12 หลักการของการควบคุมทางเวคเตอร์ของมอเตอร์เหนี่ยวนำไฟฟ้ากระแสสลับ 1 เฟส

เนื่องจากมอเตอร์เหนี่ยวนำแบบคาปาซิเตอร์มอเตอร์ จะมีชุดขดลวด 2 ชุด คือ ขดหลัก (Main , Run) ขดช่วย(Auxiliary , Start) ดังแสดงรูปที่ 2.15 โดยมีตัวเก็บประจุต่อร่วมกับขดรัน เพื่อให้เกิดมุม θ ระหว่างขดลวดทั้งสอง และเพื่อให้เกิดแรงบิดในการขับเคลื่อนมอเตอร์

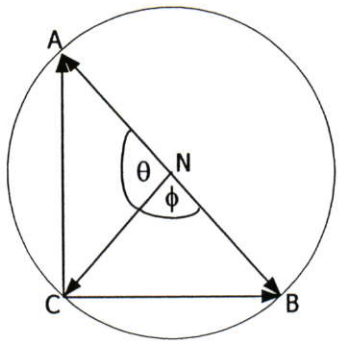


รูปที่ 2.15 โครงสร้างมอเตอร์และเวคเตอร์แรงดันระหว่างขด Run กับขด Start

แต่ในการควบคุม เราจะออกแบบให้มอเตอร์มีประสิทธิภาพสูงสุดโดยให้มุมของแรงดันระหว่างทั้งสองขดมีขนาด 90° โดยใช้โครงสร้างของสวิตช์ 3 เฟส เพื่อสร้างแรงดัน 2 ชุด ดังกล่าว โดยไม่ต้องดัดแปลงโครงสร้างของมอเตอร์เดิมดังรูปที่ 2.16



รูปที่ 2.16 โครงสร้างสวิตช์ 3 เฟส



รูปที่ 2.17 เวคเตอร์แสดงแรงดันการควบคุมมอเตอร์

จากรูปที่ 2.17 ขนาดของ V_{AC} และ V_{BC} จะได้จากการวัดแล้วนำมาทำมุมกัน 90^{o} ต่อมาเราทำการสร้างเวคเตอร์ V_{AN}, V_{BN} และ V_{CN} โดยที่จุด N จะอยู่กึ่งกลางระหว่าง A กับ B จากนั้นเราทำการปรับมุม θ เพื่อให้ได้ ผลลัพธ์ของมุมระหว่าง V_{AC} และ V_{BC} ยังมีค่า 90^{o} ซึ่ง V_{AN}, V_{BN} และ V_{CN} จะเป็นแรงดันที่ถูกสร้างขึ้นมาจากการมอดดูเลชั่นเพื่อไปจ่ายให้กับมอเตอร์ 1 เฟส

2.13 Space Vector Modulation (SVM)

หลักการของวิธี Space vector modulation (SVM) มีความแตกต่างจากหลักการของวิธี PWM โดยอินเวอร์เตอร์ที่ใช้วิธี PWM จะมีชุดขับแบบ push-pull 3 ชุด ซึ่งแต่ละชุดสร้างสัญญาณควบคุมที่แยกอิสระกัน ในขณะที่วิธี SVM จะพิจารณาอินเวอร์เตอร์เป็นชุดขับเพียงหนึ่งชุดเท่านั้น โดยที่อินเวอร์เตอร์สามารถที่จะถูกขับได้ถึง 8 สเตต ดังแสดงในตารางที่ 2.2 การเรียงสวิตช์เป็นไปตามรูปที่ 2.19

ตารางที่ 2.2 ช่วงเวลาการสวิตช์ ของอินเวอร์เตอร์ไฟสลับ 3 เฟส

State	State No.	Switch States	v_{ab}	v_{bc}	v_{ca}	Space Vector
S_1, S_2, and S_6 are on and S_4, S_5, and S_3 are off	1	100	V_S	0	$-V_S$	$\mathbf{V}_1 = 1 + j0.577 = 2/\sqrt{3}\ \angle 30^{o}$
S_2, S_3, and S_1 are on and S_5, S_6, and S_4 are off	2	110	0	V_S	$-V_S$	$\mathbf{V}_2 = j1.155 = 2/\sqrt{3}\ \angle 90^{o}$
S_3, S_4, and S_2 are on and S_6, S_1, and S_5 are off	3	010	$-V_S$	V_S	0	$\mathbf{V}_3 = -1 + j0.577 = 2/\sqrt{3}\ \angle 150^{o}$
S_4, S_5, and S_3 are on and S_1, S_2, and S_6 are off	4	011	$-V_S$	0	V_S	$\mathbf{V}_4 = -1 - j0.577 = 2/\sqrt{3}\ \angle 210^{o}$
S_5, S_6, and S_4 are on and S_2, S_3, and S_1 are off	5	001	0	$-V_S$	V_S	$\mathbf{V}_5 = -j1.155 = 2/\sqrt{3}\ \angle 270^{o}$
S_6, S_1, and S_5 are on and S_3, S_4, and S_2 are off	6	101	V_S	$-V_S$	0	$\mathbf{V}_6 = 1 - j0.577 = 2/\sqrt{3}\ \angle 330^{o}$
S_1, S_3, and S_5 are on and S_4, S_6, and S_2 are off	7	111	0	0	0	$\mathbf{V}_7 = 0$
S_4, S_6, and S_2 are on and S_1, S_3, and S_5 are off	8	000	0	0	0	$\mathbf{V}_0 = 0$

ในการควบคุมการสวิตช์ของวิธี SVM นั้นทำในระบบดิจิตอล ซึ่งกล่าวได้ว่า SVM เป็นเทคนิคการมอดดูเลตแบบดิจิตอล โดยมีเป้าหมายในการสร้างแรงดันไฟฟ้า PWM ให้กับโหลด โดยที่ค่าแรงดันไฟฟ้าดังกล่าวที่จ่ายให้กับโหลดจะเป็นค่าเฉลี่ย ซึ่งการสร้างสัญญาณ PWM ดังกล่าวจะเสร็จสิ้นภายในแต่ละคาบเวลาสุ่ม (sampling period) โดยพิจารณาจากการเลือกสเตตการสวิตช์และการคำนวณคาบเวลาของสเตตที่เหมาะสม ซึ่งคำนวณได้จากการแปลงเวคเตอร์ปริภูมิ (space vector transformation)

การแปลงปริภูมิ

พิจารณาฟังก์ชั่นทางเวลา3 ฟังก์ชั่นที่มีคุณสมบัติเป็นไปตามสมการ

$$u_a(t)+u_b(t)+u_c(t)=0 \tag{2.8}$$

การอธิบายเวคเตอร์ดังกล่าวสามารถแสดงในปริภูมิสองมิติ ซึ่งพิกัดจะเหมือนกับแรงดันไฟฟ้า 3 เฟส อาทิเช่น เวคเตอร์ $[u_a \ 0 \ 0]^T$ วางอยู่บนแกน X เวคเตอร์ $[0 \ u_b \ 0]^T$ มีเฟสต่างไป 120^0 และเวคเตอร์ $[0 \ 0 \ u_c]^T$ มีเฟสต่าง 240^0 จากเวคเตอร์ตัวแรก ดังแสดงในรูปที่ 2.18

เวคเตอร์ SV ในรูปแบบของจำนวนเชิงซ้อนแสดงได้ตามสมการ

$$u(t)=\frac{2}{3}[u_a+u_b e^{j(2/3)\pi}+u_c e^{-j(2/3)\pi}] \tag{2.9}$$

โดยที่ 2/3 เป็น scaling factor สมการ (2.9) สามารถเขียนให้อยู่ในรูปองค์ประกอบของจำนวนจริงและจำนวนจินตภาพในโดเมน $x-y$ ได้ดังนี้

$$u(t)=u_x+ju_y \tag{2.10}$$

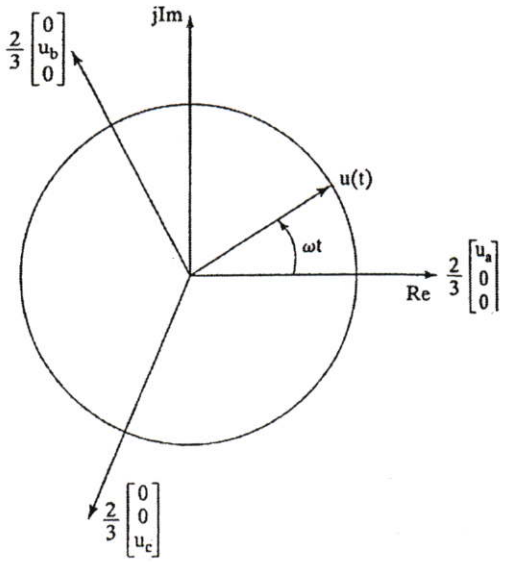
พิจารณาสมการ (2.9) และ (2.10) เราสามารถแปลงพิกัดจาก $a-b-c$ ไปเป็น $x-y$ ดังนี้

$$\begin{pmatrix} u_x \\ u_y \end{pmatrix} = \frac{2}{3}\begin{pmatrix} 1 & \frac{-1}{2} & \frac{-1}{2} \\ 0 & \frac{\sqrt{3}}{2} & \frac{-\sqrt{3}}{2} \end{pmatrix}\begin{pmatrix} u_a \\ u_b \\ u_c \end{pmatrix} \tag{2.11}$$

ซึ่งสามารถเขียนเป็น

$$u_x = \frac{2}{3}\left[v_a - 0.5(v_b + v_c)\right] \tag{2.12a}$$

$$u_y = \frac{\sqrt{3}}{3}(v_b - v_c) \tag{2.12b}$$



รูปที่ 2.18 พิกัดเวคเตอร์ในระบบสามเฟส และ space vector $\mathbf{u}(t)$

การแปลงจากแกน $x-y$ ไปเป็นแกน $\alpha-\beta$ ได้โดยทำการหมุนแกน $x-y$ ดังกล่าวด้วย ωt โดยที่ ω เป็นความเร็วเชิงมุม

$$\begin{pmatrix} u_\alpha \\ u_\beta \end{pmatrix} = \begin{pmatrix} \cos(\omega t) & \cos(\frac{\pi}{2}+\omega t) \\ \sin(\omega t) & \sin(\frac{\pi}{2}+\omega t) \end{pmatrix}\begin{pmatrix} u_x \\ u_y \end{pmatrix} = \begin{pmatrix} \cos(\omega t) & -\sin(\omega t) \\ \sin(\omega t) & \cos(\omega t) \end{pmatrix}\begin{pmatrix} u_x \\ u_y \end{pmatrix} \tag{2.13}$$

จากสมการ (2.9) เราสามารถแปลงกลับได้เป็น

$$u_a = \mathrm{Re}(\mathbf{u}) \tag{2.14}$$
$$u_b = \mathrm{Re}(\mathbf{u}e^{-j(2/3)\pi})$$
$$u_c = \mathrm{Re}(\mathbf{u}e^{j(2/3)\pi})$$

ในระบบ 3 เฟสแบบสมดุล แรงดันไฟฟ้า u_a u_b u_c สามารถเขียนได้เป็น

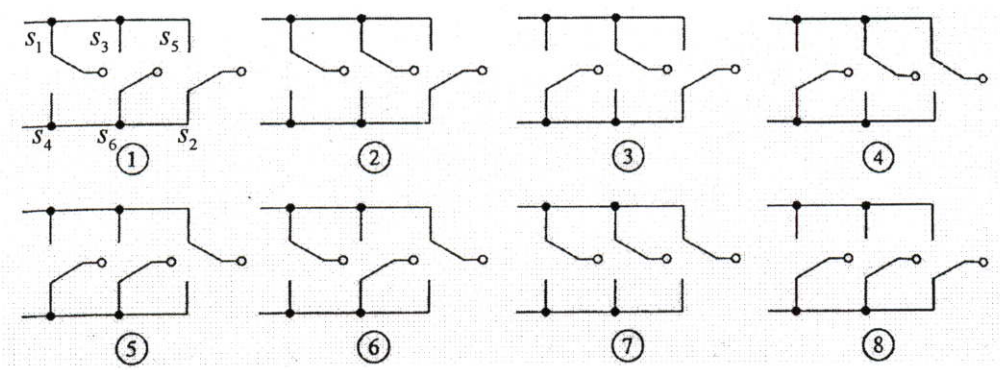
$$u_a = V_m \sin(\omega t) \tag{2.15}$$
$$u_b = V_m \sin(\omega t - 2\pi/3)$$
$$u_c = V_m \sin(\omega t + 2\pi/3)$$

โดยที่ V_m มีค่าเป็นยอดแรงดันไฟฟ้า

จากสมการ (2.9) เวคเตอร์ SV สามารถเขียนได้เป็น

$$u(t) = V_m e^{j\theta} = V_m e^{j\omega t} \tag{2.16}$$

จากสมการข้างต้น อธิบายถึงเวคเตอร์ที่มีขนาด V_m หมุนด้วยความเร็วคงที่ ω



รูปที่ 2.19 ช่วงเวลาการสวิตช์ของอินเวอร์เตอร์

เวคเตอร์ปริภูมิ (SV)

สเตตการสวิตช์ของอินเวอร์เตอร์ สามารถแสดงในรูปแบบของค่าเลขฐานสอง q_1, q_2, q_3, q_4, q_5 และ q_6 โดยที่ $q_k = 1$ เมื่อสวิตช์ ปิดวงจร และ $q_k = 0$ เมื่อสวิตช์เ ปิดวงจร

เมื่อคู่ q_1q_4, q_3q_6 และ q_5q_2 เป็นคู่ที่มีค่าตรงข้ามกัน ดังนั้น $q_4 = 1 - q_1, q_6 = 1 - q_3$ และ $q_2 = 1 - q_5$ โดยสเตตของการเปิด/ปิด สวิตช์แสดงในรูปที่ 2.19

พิจารณาการแปลงจาก 3 เฟสไปเป็น 2 เฟส ตามสมการที่ (2.11) และ พิจารณาแรงดันไฟฟ้าในสาย($\sqrt{3}$ แรงดันเฟส) เป็นค่าอ้างอิง ทำให้องค์ประกอบของ $\alpha - \beta$ ซึ่งเป็นเวคเตอร์แรงดันไฟฟ้าแบบ rms (root mean square) สามารถแสดงในรูปแบบฟังก์ชั่นของ q_1, q_3, q_5

$$\begin{pmatrix} V_{L\alpha} \\ V_{L\beta} \end{pmatrix} = \frac{2}{3}\sqrt{\frac{3}{2}}V_s \begin{pmatrix} 1 & \frac{-1}{2} & \frac{-1}{2} \\ 0 & \frac{\sqrt{3}}{2} & \frac{-\sqrt{3}}{2} \end{pmatrix} \begin{pmatrix} q_1 \\ q_3 \\ q_5 \end{pmatrix} \qquad (2.17)$$

เมื่อ V_s = แหล่งจ่ายแรงดันไฟตรง

ค่า $\sqrt{2}$ ใช้ในการแปลงค่าแรงดันไฟฟ้าแบบ rms เป็นค่าแรงดันสูงสุด สำหรับค่าสูงสุดของแรงดันสายมีค่า $2V_s/\sqrt{3}$ ในขณะที่ค่าสูงสุดของแรงดันเฟสมีค่า $V_{p(peak)} = V_s/\sqrt{3}$

เมื่อพิจารณาให้แรงดันเฟส $\mathbf{V_a}$ เป็นแรงดันอ้างอิง เวคเตอร์แรงดันในสาย $\mathbf{V_{ab}}$ จะนำหน้าเฟสเวคเตอร์อยู่ $\pi/6$ และค่าสูงสุดของแรงดันสายที่ถูกนอร์แมลไลซ์ สามารถแสดงได้ตามสมการ

$$\mathbf{V_n} = \frac{\sqrt{2}\times\sqrt{2}}{\sqrt{3}} e^{j(2n-1)\pi/6} = \frac{2}{\sqrt{3}}\left[\cos\left(\frac{(2n-1)\pi}{6}\right) + j\sin\left(\frac{(2n-1)\pi}{6}\right)\right] \qquad (2.18)$$

โดยที่ $n = 0, 1, 2, 6$ เป็นหมายเลขของเวคเตอร์แรงดันในสาย

พิจารณาหกเวคเตอร์ที่ไม่ไช่เวคเตอร์ศูนย์, $\mathbf{V_1}$ ถึง $\mathbf{V_6}$, และ $\mathbf{V_0}$ และ $\mathbf{V_7}$ ที่เป็นเวคเตอร์ศูนย์ ดังแสดงในรูปที่ 2.20 ทำให้เราสามารถกำหนดเวคเตอร์ **U** เป็นฟังก์ชั่นเวลาทางของการอินทิเกรท $\mathbf{V_n}$ ดังแสดงตามสมการ

$$\mathbf{U} = \int \mathbf{V}_n dt + \mathbf{U}_0 \qquad (2.19)$$

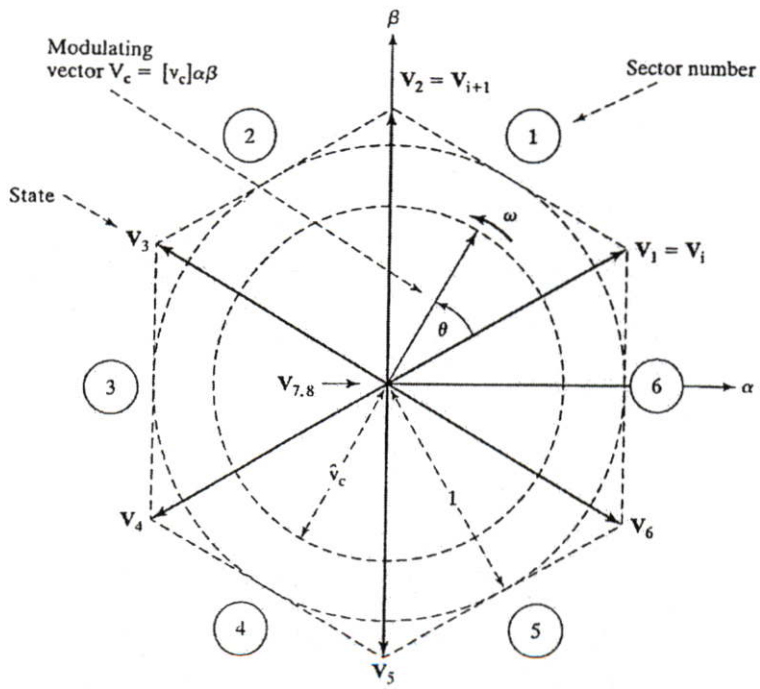
โดยที่ $\mathbf{U}_0$ เป็นสภาวะเริ่มต้น

จากสมการที่ (2.19) กายภาพของฟังก์ชั่น U แสดงได้เป็นรูปทรงเชิงเรขาคณิตหกเหลี่ยมที่คำนวณได้จากขนาด และคาบเวลาของเวคเตอร์แรงดันไฟฟ้า ถ้าแรงดันไฟฟ้าเอาท์พุทเป็นสัญญาณไซนูซอยด์ที่สมบูรณ์แบบ ดังนั้นเราสามารถพิจารณาฟังก์ชั่น U ได้ในรูปแบบ

$$\mathbf{U}^* = Me^{j\theta} = Me^{j\omega t} \qquad (2.20)$$

โดยที่ M เป็นดรรชนีการมอดดูเลต ($0 < M < 1$) สำหรับการควบคุมขนาดของแรงดันไฟฟ้าเอาท์พุท และ ω เป็นความถี่เอาท์พุท

กายภาพของฟังก์ชั่น $\mathbf{U}^*$ แสดงได้เป็นรูปทรงเชิงเรขาคณิตวงกลม ดังแสดงในรูปที่ 2.20 ตามเส้นประวงกลมที่มีรัศมี $M = 1$ และ ถูกพิจารณาเป็นเวคเตอร์อ้างอิง $\mathbf{V}_r$



รูปที่ 2.20 แสดงเวคเตอร์ปริภูมิ

กายภาพรูปทรงเชิงเรขาคณิตของฟังก์ชั่น $\mathbf{U}$ สามารถควบคุมได้โดยการเลือกเวคเตอร์ $\mathbf{V}_n$ และการปรับค่าความกว้างของเวคเตอร์ $\mathbf{V}_n$ ให้มีรูปทรงเข้าใกล้ฟังก์ชั่น $\mathbf{U}^*$ มากที่สุด ซึ่งวิธีการนี้เรียกว่า "quasi-circular"

การมอดดูเลตเวคเตอร์อ้างอิง

จากสมการที่ (2.11) และ (2.12) เวคเตอร์ของสัญญาณมอดดูเลตแรงดันสาย 3 เฟส $[v_r]_{abc} = [v_{ra} v_{rb} v_{rc}]^T$ สามารถแสดงโดยเวคเตอร์จำนวนเชิงซ้อน $\mathbf{U}^* = \mathbf{V}_r = [v_r]_{\alpha\beta} = [v_{r\alpha} v_{r\beta}]^T$ โดยได้ดังนี้

$$v_{r\alpha} = \frac{2}{3}[v_{ra} - 0.5(v_{rb} + v_{cr})] \tag{2.21}$$

$$v_{r\beta} = \frac{\sqrt{3}}{3}(v_{rb} - v_{rc}) \tag{2.22}$$

ถ้าสัญญาณมอดดูเลตแรงดันสาย $[v_r]_{abc}$ เป็นสัญญาณไซนูซอยด์ของระบบ 3 เฟสสมดุล ที่มีขนาด $A_c = 1$ และ ω เป็นความถี่เชิงมุม โดยพิจารณาในระนาบไม่เคลื่อนที่ $\alpha - \beta$ ที่ไม่เคลื่อนที่ สัญญาณมอดดูเลต $\mathbf{V}_c = [v_r]_{\alpha\beta}$ จะมีขนาดคงที่ $MA_c (= M)$ และหมุนด้วยความถี่ ω ดังแสดงเป็นวงกลมจุดไข่ปลาที่มีรัศมี M ในรูปที่ 2.20

การสวิตช์ชิ่งแบบ SV

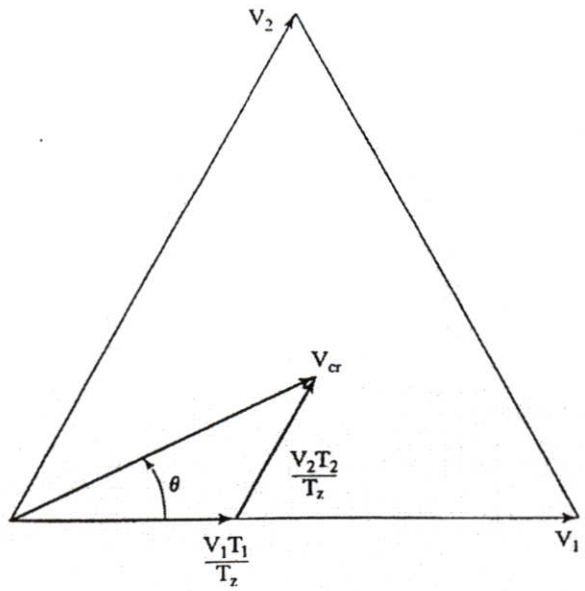
จุดประสงค์ของสวิตช์ชิ่งแบบ SV เป็นการสร้างสัญญาณมอดดูเลต $\mathbf{V}_r$ ในสายให้มีความใกล้เคียงกับสัญญาณไซนูซอยด์ $\mathbf{V}_r$ ซึ่งมี 8 SV ($\mathbf{V}_n, n = 0, 2, ..., 7$) อย่างไรก็ตามถ้าสัญญาณการมอดดูเลต $\mathbf{V}_c$ ถูกวางให้อยู่ระหว่างเวคเตอร์ $\mathbf{V}_n$ และ $\mathbf{V}_{n+1}$ จะส่งผลให้เวคเตอร์ศูนย์จำนวนสองเวคเตอร์ และ SV เวคเตอร์ที่เป็นศูนย์ ($\mathbf{V}_z = \mathbf{V}_0$ หรือ $\mathbf{V}_7$) จะถูกใช้เพื่อให้ให้ได้แรงดันในสายสูงสุดที่ตกคร่อมโหลด พร้อมกับทำให้ความถี่การสวิตช์มีค่าน้อยที่สุด

เวคเตอร์ $\mathbf{V}_r$ ของส่วนที่หนึ่ง (แสดงในรูปที่ 2.20) สามารถพิจารณาได้จากเวคเตอร์ $\mathbf{V}_1$ และ $\mathbf{V}_2$ และ หนึ่งเวคเตอร์ศูนย์ ($\mathbf{V}_0$ หรือ $\mathbf{V}_7$) ในความหมายอีกนัยหนึ่ง สเตตของ $\mathbf{V}_1$ แอคทีฟเป็นช่วงเวลา T_1 แอคทีฟเป็นช่วงเวลา ในขณะที่ $\mathbf{V}_2$ แอคทีฟเป็นช่วงเวลา T_2 สำหรับ T_z เป็นช่วงเวลาแอคทีฟของเวคเตอร์ศูนย์ ($\mathbf{V}_0$ หรือ $\mathbf{V}_7$)

เมื่อพิจารณาความถี่การสวิตช์ที่สูงเพียงพอสำหรับการทำงาน เวคเตอร์อ้างอิง $\mathbf{V}_r$ สามารถถูกอนุมานให้มีค่าคงที่ตลอดหนึ่งคาบเวลาของการสวิตช์ และจากสาเหตุที่เวคเตอร์ $\mathbf{V}_1$ และ $\mathbf{V}_2$ มีค่าคงที่ และ เวคเตอร์ $\mathbf{V}_z$ มีค่าเป็นศูนย์ ทำให้เราสามารถกำหนดให้เวลาของการสวิตช์ได้ตามสมการ

$$\mathbf{V}_r \times T_s = \mathbf{V}_1 \times T_1 + \mathbf{V}_2 \times T_2 + \mathbf{V}_z \times T_z \qquad (2.23)$$

สมการข้างต้นอธิบายถึงหลักการของ SVM ซึ่งอาศัยเวคเตอร์ SV จำนวนสองเวคเตอร์ที่อยู่ใกล้กันที่มีวงรอบเวลาการทำงานที่เหมาะสม โดยกายภาพของวิธี SVM แสดงในรูปที่ 2.21



รูปที่ 2.21 การหาค่าเวลาสเตตของ SVM

เมื่อพิจารณาเวคเตอร์ SV ในระบบพิกัดเชิง สมการที่ (2.20) สามารถเขียนใหม่ได้เป็น

$$T_s\mathbf{M}\begin{pmatrix}\cos\left(\frac{\pi}{6}+\theta\right)\\ \sin\left(\frac{\pi}{6}+\theta\right)\end{pmatrix} = T_1\frac{2}{\sqrt{3}}\begin{pmatrix}\cos\left(\frac{\pi}{6}\right)\\ \sin\left(\frac{\pi}{6}\right)\end{pmatrix} + T_2\frac{2}{\sqrt{3}}\begin{pmatrix}\cos\left(\frac{\pi}{2}\right)\\ \sin\left(\frac{\pi}{2}\right)\end{pmatrix} + T_z 0 \qquad (2.24)$$

เมื่อพิจารณาส่วนจำนวนจริง และจำนวนจินตภาพ ทั้งสองข้างของสมการข้างต้น

$$T_s M \cos\left(\frac{\pi}{6}+\theta\right) = T_1 \frac{2}{\sqrt{3}} \cos\left(\frac{\pi}{6}\right) + T_2 \frac{2}{\sqrt{3}} \cos\left(\frac{\pi}{2}\right)$$

$$T_s M \sin\left(\frac{\pi}{6}+\theta\right) = T_1 \frac{2}{\sqrt{3}} \sin\left(\frac{\pi}{6}\right) + T_2 \frac{2}{\sqrt{3}} \sin\left(\frac{\pi}{2}\right)$$

ค่าคาบเวลา T_1 และ T_2 คำนวณได้จาก

$$T_1 = T_s M \frac{\sqrt{3}}{2} \frac{\cos\left(\frac{\pi}{6}+\theta\right)}{\cos\left(\frac{\pi}{6}\right)} = T_s M \cos\left(\frac{\pi}{6}+\theta\right) = T_s M \sin\left(\frac{\pi}{3}-\theta\right) \qquad (2.25)$$

$$T_2 = T_s M \frac{\sqrt{3}}{2} \frac{\sin(\theta)}{\sin\left(\frac{\pi}{6}\right)} = T_s M \sin(\theta) \qquad (2.26)$$

$$T_0 = T_z = T_s - T_1 - T_2 \qquad (2.27)$$

โดยที่ M เป็นดรรชนีการมอดดูเลต

θ เป็นมุมระหว่าง $\mathbf{V}_r$ และ $\mathbf{V}_n$

T_S เป็นคาบเวลาของการสวิตช์รวม

T_0 เป็นคาบเวลาของเวคเตอร์แรงดัน $\mathbf{V}_0, \mathbf{V}_7$

T_1 เป็นคาบเวลาของเวคเตอร์แรงดัน $\mathbf{V}_1$

T_2 เป็นคาบเวลาของเวคเตอร์แรงดัน $\mathbf{V}_2$

สำหรับส่วนที่ 2 ถึง 6 เราสามารถที่จะพิจารณาได้เหมือนกับกรณีส่วนที่หนึ่ง เพื่อคำนวณหาเวลาสเตต โดยสมมุติให้อินเวอร์เตอร์ทำงานที่ความถี่คงที่ และ T_S ยังคงมีค่าคงที่

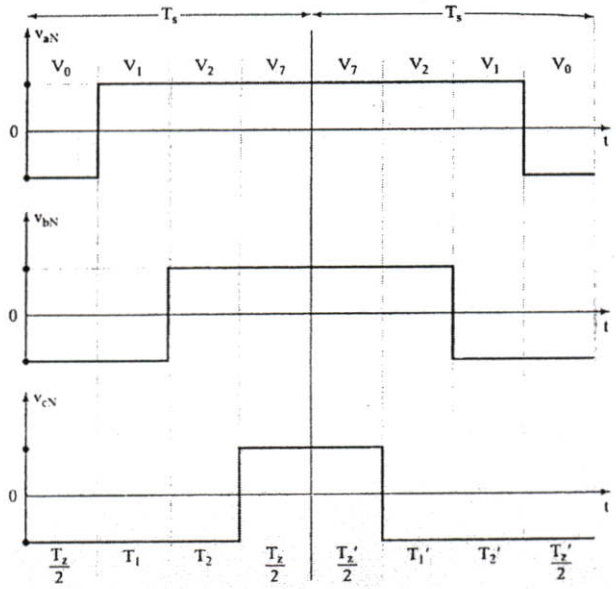
ลำดับการสวิตช์ SV

ลำดับการสวิตช์ SV ถูกนำมาพิจารณาเพื่อให้แรงดันไฟฟ้าในสายมีความสมมาตรและลดผลของฮาร์มอนิกคู่ นอกจากนี้ในการที่จะลดค่าความถี่ของการสวิตช์ จำเป็นที่ต้องทำการจัดเรียงลำดับการสวิตช์ในทิศทางไปสู่ลำดับต่อไปโดยการสวิตช์เพียงหนึ่งกิ่งของชุดขับ ณ.เวลาหนึ่ง ถึงแม้ว่าการทำงานในลักษณะแบบนี้จะไม่ใช่แนวทางสมมาตรในการสร้างลำดับการสวิตช์ SV

โดยสถานะของลำดับการสวิตช์เป็น $\mathbf{V}_z, \mathbf{V}_n, \mathbf{V}_{n+1}\ \mathbf{V}_z$ (โดยที่ $\mathbf{V}_z$ เป็นตัวเลือกระหว่าง $\mathbf{V}_0$ และ $\mathbf{V}_7$)

จากรูปที่ 2.20 พิจารณาเวคเตอร์อ้างอิงในส่วนที่หนึ่ง ลำดับของการสวิตช์เป็น $\mathbf{V}_0, \mathbf{V}_1, \mathbf{V}_2$, $\mathbf{V}_7$, $\mathbf{V}_7, \mathbf{V}_2, \mathbf{V}_1, \mathbf{V}_0$ ช่วงเวลา $T_z(=T_0=T_7)$ สามารถแยกและกระจายที่ได้จุดเริ่มต้นและจุดสิ้นสุดของคาบเวลาการสุ่ม T_S

รูปที่ 2.22 แสดงลำดับการสวิตช์ และส่วนประกอบของแรงดันไฟฟ้าสามเฟสเอาท์พุทในช่วงสองคาบเวลาการสุ่ม โดยทั่วไปช่วงเวลาของเวคเตอร์ศูนย์จะมีการกระจายที่เท่าๆกัน ด้วยคาบเวลา $T_z/2$ ทั้งที่จุดเริ่มต้นและจุดสิ้นสุด



รูปที่ 2.22 รูปแบบของ SVM

แรงดันไฟฟ้าเฟส ณ. เวลาใดเวลาหนึ่งสามารถพิจารณาได้จากค่าเวลาเฉลี่ยของ SV ตลอดช่วงหนึ่งคาบเวลาการสวิตช์สำหรับส่วนที่หนึ่ง ดังแสดงได้ตามสมการ

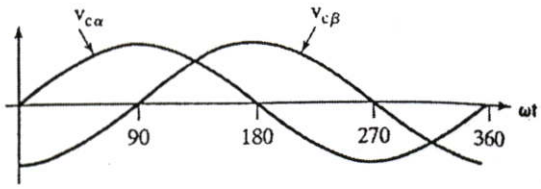
$$v_{aN} = \frac{V_s}{2T_s}\left(\frac{-T_z}{2} + T_1 + T_2 + \frac{T_z}{2}\right) = \frac{V_s}{2}\sin\left(\frac{\pi}{3} + \theta\right) \qquad (2.28a)$$

$$v_{bN} = \frac{V_s}{2T_s}\left(\frac{-T_z}{2} - T_1 + T_2 + \frac{T_z}{2}\right) = V_s\frac{\sqrt{3}}{2}\sin\left(\theta - \frac{\pi}{6}\right) \qquad (2.28b)$$

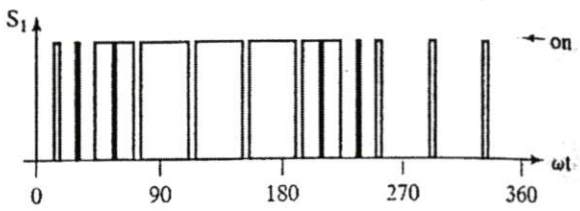
$$v_{cN} = \frac{V_s}{T_s}\left(\frac{-T_z}{2} - T_1 - T_2 + \frac{T_z}{2}\right) = -V_{aN} \qquad (2.28c)$$

ในการที่จะลดผลของฮาร์มอนิกที่ไม่พึงประสงค์จากการมอดดูเลตโดยวิธี SVM ค่าความถี่สุ่มที่ถูกนอร์แมลไลซ์ f_{sn} จะต้องเป็นเลขจำนวนเต็มที่คูณด้วย 6 โดยที่ $T \geq 6nT_S$ สำหรับ $n = 0,1,2,3,\ldots$ ซึ่งการที่จะทำให้แรงดันเอาท์พุทในสายมีความสมมาตร จำเป็นที่ส่วนทั้งหกจะต้องถูกใช้เท่าๆกันในหนึ่งคาบเวลา

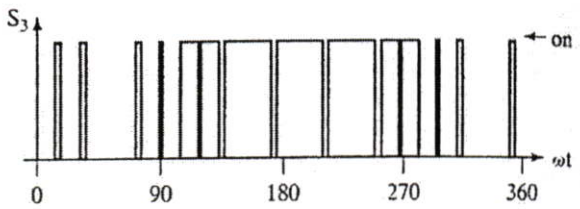
รูปที่ 2.23 แสดงรูปแบบทั่วไปของสัญญาณการมอดดูเลต SV สำหรับ $f_{sn} = 18$ และ $M =$ 0.8



(ก) สัญญาณมอดดูเลต

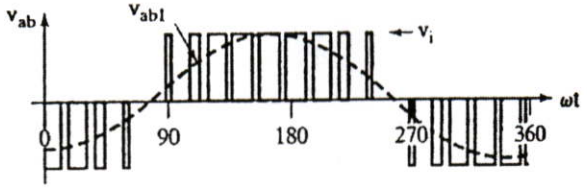


(ข) ช่วงเวลาการสวิตช์ของสวิตช์ S_1



(ค) ช่วงเวลาการสวิตช์ของสวิตช์ S_3

รูปที่ 2.23 รูปสัญญาณ3เฟสของการมอดดูเลต space vector ($M = 0.8, f_{sn} = 18$)



(ง) สเปคตรัมของแรงดันไฟฟ้าสลับที่ขาออกของวงจร

รูปที่ 2.23(ต่อ) รูปสัญญาณ3เฟสของการมอดดูเลต space vector ($M = 0.8, f_{sn} = 18$)

การมอดดูเลตเกิน (overmodulation)

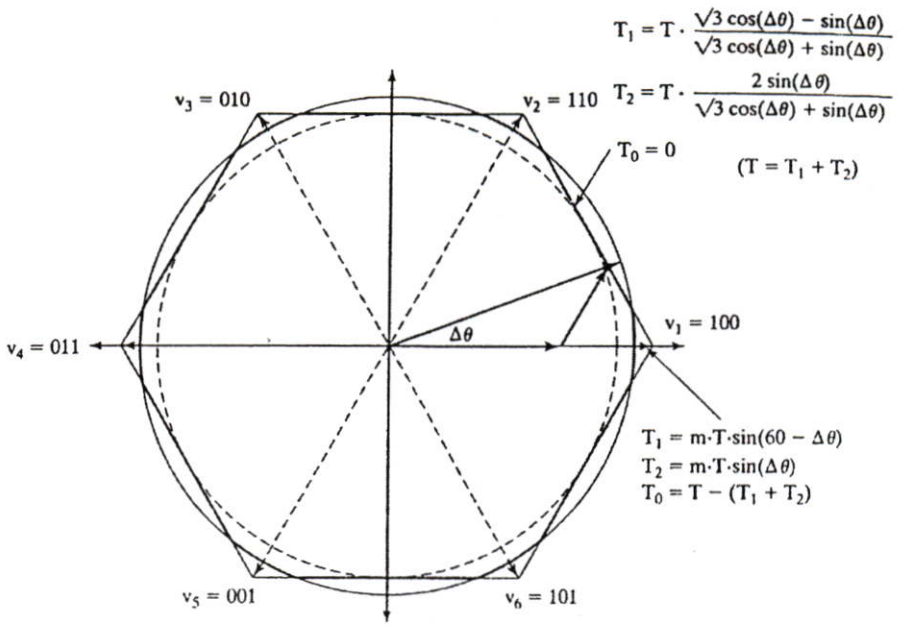
ในกรณีของการมอดดูเลตเกิน เวคเตอร์อ้างอิงจะตามแนววิถีวงกลม ซึ่งขยายไปจากขอบเขตเดิมของรูปทรงหกเหลี่ยม โดยที่ส่วนของวงกลมที่อยู่ภายในรูปทรงหกเหลี่ยมสามารถที่จะใช้สมการ SVM สำหรับการคำนวณหาเวลาสเตต T_n, T_{n+1} และ T_z ตามสมการ (2.25) ถึง (2.27) อย่างไรก็ตามในส่วนของวงกลมที่อยู่ภายนอกรูปทรงหกเหลี่ยมจะถูกจำกัดโดยขอบของรูปทรงหกเหลี่ยมดังแสดงในรูปที่ 2.24 โดยเวลาสเตต T_n และ T_{n+1} ที่สอดคล้องกับกรณี สามารถคำนวณได้จาก

$$T_n = T_s \frac{\sqrt{3}\cos(\theta) - \sin(\theta)}{\sqrt{3}\cos(\theta) + \sin(\theta)} \qquad (2.29a)$$

$$T_{n+1} = T_s \frac{2\sin(\theta)}{\sqrt{3}\cos(\theta) + \sin(\theta)} \qquad (2.29b)$$

$$T_z = T_s - T_1 - T_2 = 0 \qquad (2.29c)$$

ค่าดัชนีการมอดดูเลตสูงสุด M สำหรับ SVM มีค่า $M_{\max} = 2/\sqrt{3}$ สำหรับกรณี $0 < M \le 1$ อินเวอร์เตอร์ทำงานในโหมด SVM ปกติ และกรณี $M \ge 2/\sqrt{3}$ อินเวอร์เตอร์ทำงานในโหมดเอาท์พุทหกขั้น โดยการทำงานของโหมดเอาท์พุทหกขั้น จะทำการสวิตช์อินเวอร์เตอร์ตามตารางที่ 2.2



รูปที่ 2.24 การมอดดูเลชั่นเกิน

ด้วยวิธีนี้ ทำให้จำนวนการสวิตช์ ณ.เวลาหนึ่งๆ มีจำนวนลดลง สำหรับกรณี $1 < M < 2/\sqrt{3}$ อินเวอร์เตอร์จะทำงานในโหมดมอดดูเลตเกิน ซึ่งโดยปกติแล้วถูกใช้เป็นขั้นตอนของการเปลี่ยนแปลงจากเทคนิคของ SVM ไปสู่การทำงานในโหมดหกขั้น ถึงแม้ว่าการมอดดูเลตเกินจะยินยอมให้มีการใช้แรงดันไฟฟ้าอินพุทดีซีในปริมาณที่มากกว่าเทคนิค SVM แต่จะส่งผลให้แรงดันไฟฟ้าเอาท์พุทมีลักษณะไม่เป็นแบบไซนูซอยด์ โดยเฉพาะที่ความถี่เอาท์พุทต่ำๆ

การเปรียบเทียบเทคนิค PWM

เทคนิคการมอดดูเลตถูกนำมาใช้ในการสร้างรูปสัญญาณความถี่หรือรูปสัญญาณแรงดันไฟฟ้าที่มีการเปลี่ยนแปลงตามเวลา ซึ่งเป็นที่ทราบกันดีว่าสัญญาณมอดดูเลต PWM ที่ใช้ในการสวิตช์อินเวอร์เตอร์นั้นถูกสร้างมาจากการเปรียบเทียบระหว่างสัญญาณพาห์รูปคลื่นสามเหลี่ยมความถี่สูงและสัญญาณไซนูซอยด์อ้างอิงสามสัญญาณ ซึ่งหลักการพื้นฐานนี้ได้ถูกนำมาพัฒนาสร้างทั้งทางแอนาลอกและดิจิตอลสำหรับการแปลงกำลังงาน

สัญญาณ PWM ได้รับความนิยมในการประยุกต์ใช้งานกับระบบไฟฟ้าสามเฟส เนื่องจากองค์ประกอบฮาร์มอนิกที่สามที่ถูกขจัดออก และ การใช้แรงดันไฟฟ้ากระแสตรงในการสร้างสัญญาณ PWM ในขณะที่วิธี SV ไม่ได้พิจารณาแยกส่วนในแต่ละเฟสแรงดันไฟฟ้าเหมือนกับกรณีของ PWM แต่แรงดันไฟฟ้าทั้งสามเฟสซึ่งถูกพิจารณาภายในระบบอ้างอิงสองมิติ (ระนาบ α - β) และ เวคเตอร์อ้างอิงจะถูกประมวลผลเป็นชุดเดียวกัน นอกจากนี้วิธี SVM ยังมีข้อดีในแง่ฮาร์มอนิก

ที่ต่ำ และ ดรรชนีการมอดดูเลตที่สูง โดยเฉพาะการพัฒนาในเชิงดิจิตอลที่ใช้ไมโครโปรเซสเซอร์ชิฟเดี่ยว จากคุณสมบัติที่ หยืดหยุ่นในการควบคุม ทำให้วิธี SVM ได้รับความนิยมในการประยุกต์ใช้งานด้านการแปลงกำลังงาน และการควบคุมมอเตอร์

ตารางที่ 2.3 แสดงผลรวมของการมอดดูเลตประเภทต่างๆสำหรับอินเวอร์เตอร์สามเฟส ที่ $M = 1$

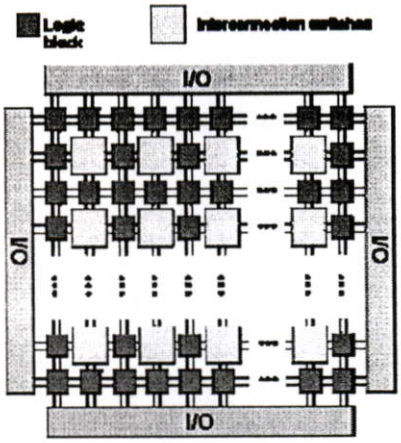
Modulation Type	Normalized Phase Voltage, V_P/V_S	Normalized Line Voltage, V_L/V_S	Output Waveform
Sinusoidal PWM	0.5	$0.5 \times \sqrt{3} = 0.8666$	Sinusoidal
60° PWM	$1/\sqrt{3} = 0.57735$	1	Sinusoidal
Third-harmonic PWM	$1/\sqrt{3} = 0.57735$	1	Sinusoidal
SVM	$1/\sqrt{3} = 0.57735$	1	Sinusoidal
Overmodulation	Higher than the value for $M = 1$	Higher than the value for $M = 1$	Nonsinusoidal
Six-step	$\sqrt{2/3} = 0.4714$	$\sqrt{(2/3)} = 0.81645$	Nonsinusoidal

2.14 FPGA (Field Programmable Gate Array)

เอฟพีจีเอ(FPGA) คือวงจรรวมชนิดหนึ่งที่พัฒนา หรือขยายความสามารถให้มากกว่า PLA/PLD (Programmable logic array/Programmable logic device) ซึ่งภายในประกอบไปด้วยเกตจำนวนมาก และความเร็วที่เพิ่มขึ้น สำหรับการรองรับการใช้งานในระบบที่มีความซับซ้อนมากขึ้นกว่า PLA/PLD ตัว FPGA นั้นเป็นส่วนผสมระหว่างฮาร์ดแวร์และซอฟท์แวร์ นั่นคือสามารถนำมาใช้งานเป็นฮาร์ดแวร์ที่สามารถโปรแกรมได้ (Programmable) หรือเปลี่ยนลักษณะของการทำงานได้ (Reconfigurable) เหมือนซอฟท์แวร์ ทำให้มีความยืดหยุ่นในการใช้งานสูง ทำให้ FPGA เริ่มเป็นที่นิยมมากขึ้นในปัจจุบัน นอกจากความง่ายใช้การออกแบบแล้ว ยังสามารถกำจัดความเสี่ยงในการออกแบบ เพราะสามารถนำมาโปรแกรมใหม่ได้ตลอดเวลา ต่างจากวงจรที่เป็นฮาร์ดแวร์ซึ่งไม่สามารถแก้ไขวงจรที่ออกแบบได้อีก โดย FPGA ในปัจจุบันมีความจุเกตเพิ่มขึ้นมากกว่าช่วงเริ่มแรกที่มีเกตไม่กี่ร้อยตัว ซึ่งปัจจุบันสามารถทำได้หลายล้านตัวในชิฟ FPGA เพียงตัว ทำให้สามารถทำการโปรแกรมได้วงจรที่ซับซ้อน และใหญ่ขึ้น

ซึ่งคุณสมบัติของ FPGA ส่วนใหญ่แล้วภายในจะประกอบไปด้วย ลอจิกเซลล์ที่สามารถโปรแกรมได้ หน่วยความจำ RAM , DSP block มีพอร์ต I/O เป็นจำนวนมาก และยังสามารถส่งข้อมูลแบบอนุกรมที่ความสูงมากกว่า 600Mb/s (Multi-gigabit transceiver) เมื่อนำไปเปรียบเทียบ

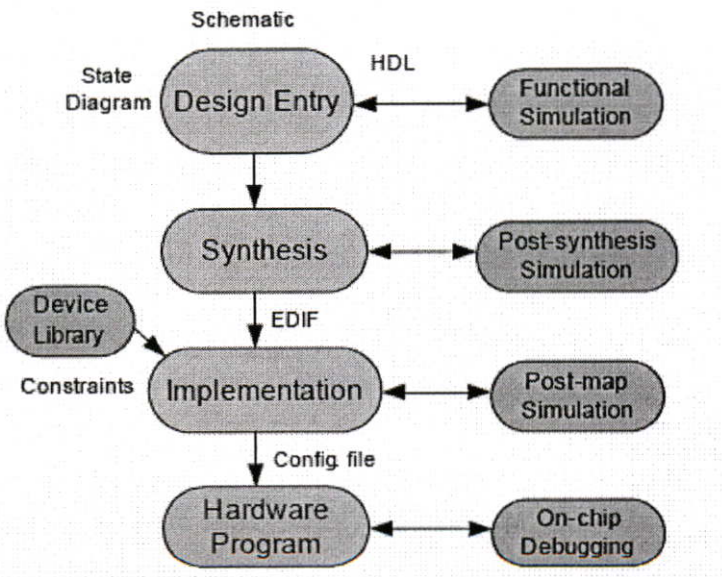
กับไมโครคอนโทรลเลอร์เช่น MSC-51 หรือ ตระกูล PIC แล้ว FPGA มีข้อได้เปรียบมากกว่าตรงที่สามารถนำเอาวงจรที่ออกแบบทั้งหมด นำมาโปรแกรมไว้ใน FPGA ได้โดยไม่ต้องต่อลอจิกเกตภายนอกเพิ่มเติม ทั้งยังสามารถจำลองไมโครคอนโทรลเลอร์ไว้ภายในตัวด้วย อีกทั้งยัง



รูปที่ 2.25 แสดงโครงสร้างภายใน FPGA

มีความเร็วของสัญญาณนาฬิกาที่สูงกว่ามาก เช่น FPGA เบอร์ XC2S100 ที่ใช้มีสัญญาณนาฬิกา 160MHz

ส่วน MCS-51 และ PIC16F72 จะมีสัญญาณนาฬิกา 20MHz ส่วนข้อด้อยกว่าของ FPGA คือราคาเมื่อเทียบกับ ไมโครคอนโทรลเลอร์แล้วยังแพงกว่า แต่เมื่อเทียบกับระบบรวมแล้วที่ไม่ต้องต่อลอจิกเกต ภายนอก ไม่ต้องทำแผ่นวงจรพิมพ์ PCB ในส่วนของภาคดิจิตอลที่ออกแบบเพิ่ม จะทำให้ราคาของระบบมีราคาใกล้เคียง หรือถูกกว่า



รูปที่ 2.26 แสดงขั้นตอนออกแบบบน FPGA

ส่วนในการออกแบบวงจรดิจิตอลด้วย FPGA นั้นต่างจากไมโครคอนโทรลเลอร์ในลักษณะเป็นการออกแบบฮาร์ดแวร์ หรือวงจรดิจิตอล แทนที่จะเป็นซอฟท์แวร์ หรือตัวโปรแกรมโดยตรง ซึ่งทั่วไปมีองค์ประกอบ 4 ส่วนหลักๆด้วยกันคือ

Design Entry เป็นขั้นตอนของการรับข้อมูลของการออกแบบเข้าไปในระบบ โดยทั่วไปสามารถทำได้หลายทาง เช่นโดยใช้ Schematic design entry ดึงอุปกรณ์มาจากไลบารีของ FPGA หรือโดยใช้ภาษา HDL อย่างเช่น VHDL หรือ Verilog ซึ่งออกแบบโดยใช้ภาษาชั้นสูงนั้นมีของดีกว่าคือ ไม่ขึ้นกับเทคโนโลยี หรือตัวชิฟที่จะถูกโปรแกรม ดังนั้นผู้ออกแบบไม่จำเป็นต้องรู้ถึงลักษณะการเชื่อมต่อของวงจร ตัวอย่างเช่นการสร้างตัวบวก(adder)ที่ทำการบวกข้อมูล a กับb เข้าด้วยกัน ก็เพียงแค่เพิ่มบรรทัด z = a+b; เข้าไปเท่านั้น ส่วนการทดสอบความถูกต้องเป็นในลักษณะการตรวจสอบระดับฟังก์ชั่นการทำงาน (Function simulation) โดยใช้ซอฟต์แวร์สำหรับจำลองการทำงาน ซึ่งจะไม่มีข้อมูลเกี่ยวกับดีเลย์ให้เห็น

Synthesis ขั้นตอนนี้จะเกี่ยวข้องกับการแปลงแบบที่ได้ออกแบบจาก Schematic หรือ HDL ให้เป็นวงจรลอจิก โดยจะมีขั้นตอนย่อยๆคือการสังเคราะห์วงจร (Logic synthesis) ซึ่งโดยปกติแล้วจะมีขั้นตอนการ optimization ด้วยเพื่อให้ได้วงจรที่ใช้ทรัพยากรน้อยที่สุด หรือทำงานได้เร็วที่สุด ผลลัพธ์จากขั้นตอนนี้จะเป็น EDIF (Electronic design intermediate form) ซึ่งเป็นไฟล์ที่อธิบายการเชื่อมต่อ (Netlist) และยังไม่ขึ้นกับเทคโนโลยี นั่นหมายถึงว่าเป็นมาตราฐานที่ตัวเครื่องมือในการออกแบบ (Design tools) ส่วนมากจะสามารถเข้าใจไฟล์ชนิดนี้ได้ การจำลองการทำงานในขั้นตอนนี้จะเป็นลักษณะของ Post-synthesis simulation ซึ่งจะไม่มีข้อมูลเกี่ยวกับดีเลย์ให้เห็นเช่นกัน แต่ผลการจำลองจะได้มาจาก EDIF ไฟล์

Implementation หลังจากได้ Netlist file มาแล้วจะเป็นการแมป(MAP) ให้เข้ากับเทคโนโลยี หรือตัวชิฟที่จะใช้งาน(Technology mapping) เช่นโดยส่วนมากแล้ว FPGA จะเป็นเทคโนโลยีแบบ LUT (Look-up table) และหลังจากนั้นก็จะทำการวางตำแหน่ง (Placement) ของลอจิกต่างๆ แล้วทำการเชื่อมต่อสายสัญญาณ (Routing)ก่อนที่จะได้ไฟล์สำหรับโปรแกรมลงชิป (Reconfiguration file) การจำลองการทำงานในระดับนี้จะสามารถดูดีเลย์ตามจุดต่างๆได้หลังจากมีการสร้างขึ้นจริงๆตามเทคโนโลยี หรือตัวชิฟที่จะใช้งาน

Hardware Program การโปรแกรมอุปกรณ์หรือชิฟ FPGA นั้นเราสามารถทำเองได้ ได้ทันที รวมทั้งการปรับเปลี่ยนแก้ไขได้ตลอดเวลา โดยกลับไปทำตั้งแต่ต้น ก่อนที่จะทำการโปรแกรม

ทดสอบการทำงานจริงๆได้โดย ไม่มีค่าใช้จ่าย หรือความเสียหายเกิดขึ้น ซึ่งเหมือนกับไมโครคอนโทรลเลอร์นั่นเอง ดังนั้นชิฟ FPGA จึงอาจถูกเรียกว่าฮาร์ดแวร์ที่สามารถโปรแกรมได้ (Reprogrammable hardware)

ความสามารถของ FPGA นั้นสูงกว่าไมโครคอนโทรลเลอร์มาตามที่กล่าวมาแล้ว โดยตัวชิฟ FPGA ทำงานที่ความถี่ 300MHz นั้นเร็วกว่า 1GHz 32bit processor (ที่ต้องทำการประมวลผลโปรแกรมทีละบรรทัด) หลายเท่าตัว เนื่องจากการทำงานบน FPGA นั้นเป็นลักษณะของฮาร์ดแวร์ที่สามารถทำงานได้พร้อมๆกันเป็นแบบขนาน ส่วนราคาจะสูงตามประสิทธิภาพ

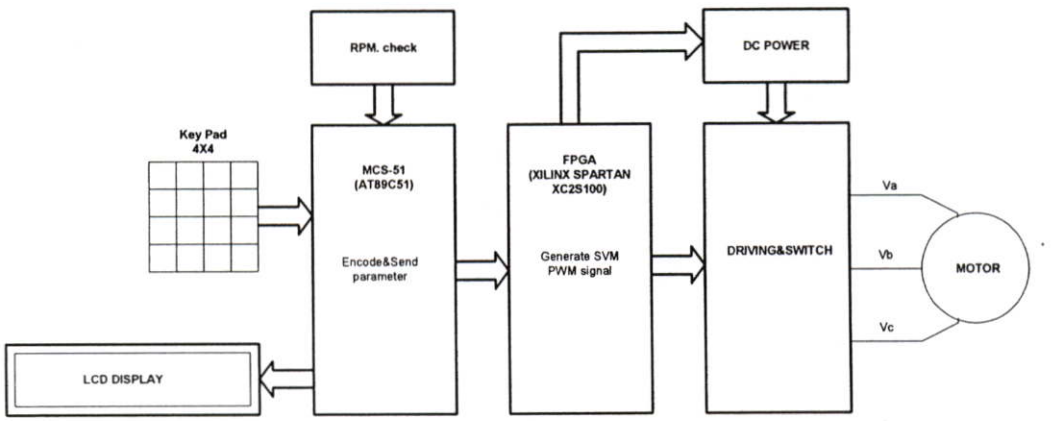
บทที่3

หลักการทำงาน และการออกแบบ

โครงสร้างของส่วนฮาร์ดแวร์ประกอบไปด้วย 2ส่วนหลักได้แก่

-ส่วนวงจรขับมอเตอร์

-ส่วนวงจรควบคุม และสร้างสัญญาณ SVM PWM



รูปที่ 3.1 บล็อกไดอะแกรมของอินเวอร์เตอร์ SVM PWM

การดำเนินงานวิจัยนี้แบ่งออกเป็น 2 ส่วน ได้แก่ ชุดควบคุมอินเวอร์เตอร์สำหรับมอเตอร์เหนี่ยวนำ 3 เฟส ส่วนออกแบบ และพัฒนาซอฟต์แวร์ควบคุม

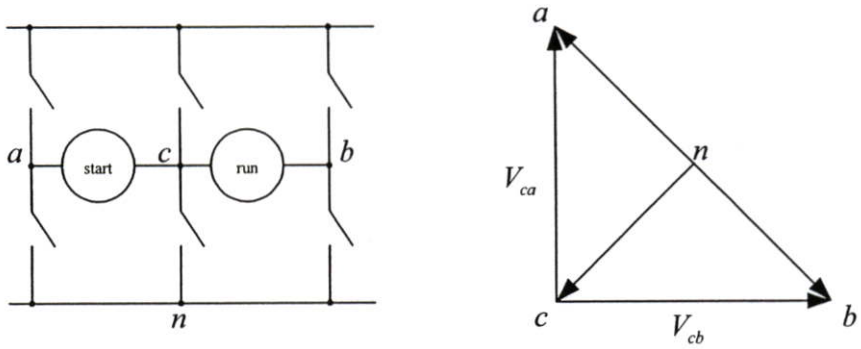
จากรูปที่ 3.1 ระบบของอินเวอร์เตอร์ SVM PWM แบ่งออกเป็นสองส่วนหลักคือ

ส่วนวงจรขับมอเตอร์ ประกอบไปด้วย วงจรภาคขับสวิตช์ IGBT วงจรแปลงแรงดัน และวงจรป้องกันการลัดวงจรของสวิตช์ วงจรทวีแรงดัน

ส่วนออกแบบ และพัฒนาซอฟต์แวร์ควบคุมซึ่งจะประกอบไปด้วย ส่วนของไมโครคอนโทรลเลอร์ MCS-51(AT89C51) ทำหน้าที่รับค่าจากคีย์บอร์ด และส่งสถานะการทำงานแสดงผลผ่านแอลซีดี อีกส่วนคือ FPGA ทำหน้าที่รับค่าตัวแปรพารามิเตอร์จาก MCS-51 เพื่อการคำนวณ และสร้างสัญญาณ SVM PWM

3.1 การออกแบบหาดรรชนีการมอดดูเลต และมุมสำหรับมอเตอร์เหนี่ยวนำไฟฟ้ากระแสสลับ 1เฟส

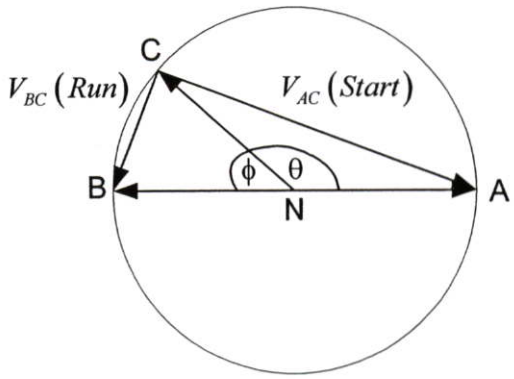
ชุดควบคุมอินเวอร์เตอร์หนึ่งเฟส ที่พัฒนาขึ้นใช้หลักการของการควบคุมทางเวคเตอร์ ดังโครงสร้างที่แสดงในรูปที่ 3.2



รูปที่ 3.2 สวิตช์และเวคเตอร์

เราทำการออกแบบสัญญาณ PWM ซึ่งถูกสร้างขึ้นเพื่อควบคุมแรงดันไฟฟ้าที่จุด a ให้ได้ค่า V_{an} และ V_{bn} ที่เหมาะสมกับขดลวดของมอเตอร์ จากนั้นควบคุมค่า V_{cn} เพื่อให้ได้ผลเฟสของ V_{ca} และ V_{cb} ที่ตั้งฉากกัน 90 องศา

แรงดันไฟฟ้าที่วัดได้จากขดสตาร์ต และขดรันที่มอเตอร์ถูกนำมาสร้างเป็นเวคเตอร์ดังแสดงในรูปที่ 3.3



รูปที่ 3.3 เวคเตอร์สำหรับหา ดรรชนีการมอดดูเลต

วัด $V_{AC} = 343V_{rms}$

$V_{BC} = 230V_{rms}$

$V_{AB} = \sqrt{343^2 + 230^2}$

$= 413V_{rms}$

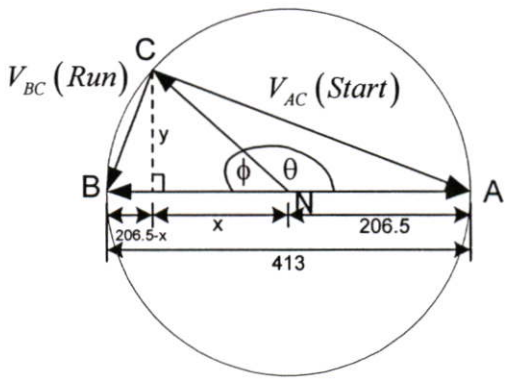
$V_{AN} = V_{BN} = V_{CN} = \frac{413}{2}$

$= 206.48V_{rms}$

$\therefore \quad V_p = 206.48 \times \sqrt{2} = 292V$

%ดรรชนีการมอดดูเลต $= \frac{292}{310} \times 100$

$= 94.2\%$



รูปที่ 3.4 เวคเตอร์การหามุม

$$y^2 + (206.5 - x)^2 = 230^2 \tag{3.1}$$

$$y^2 + (x + 206.5)^2 = 343^2 \tag{3.2}$$

คำนวณหาค่า x จากสมการ (3.1) และ (3.2)

$$x = \frac{343^2 - 230^2}{2(413)} \tag{3.3}$$

$= 78.4$

$y = \sqrt{230^2 - (206.5 - 78.4)^2}$

$= 191$

ค่ามุม θ คำนวณได้จาก

$$\phi = \tan^{-1}\frac{191}{78.4} \qquad (3.4)$$

$$= 67.68^0$$

$$\theta = 180 - 67.68 = 112.32^0$$

3.2 การออกแบบวงจรภาคขับสวิตช์ IGBT

โครงสร้างของชุดขับมอเตอร์เหนี่ยวไฟฟ้ากระแสสลับทั้งแบบ 1เฟส และ3เฟสใช้โครงสร้างแบบเดียวกันโดยต่างกันตรงการต่อมอเตอร์เข้ากับชุดขับ โดยแสดงดังรูปที่ 3.5 และ 3.6 ตามลำดับ

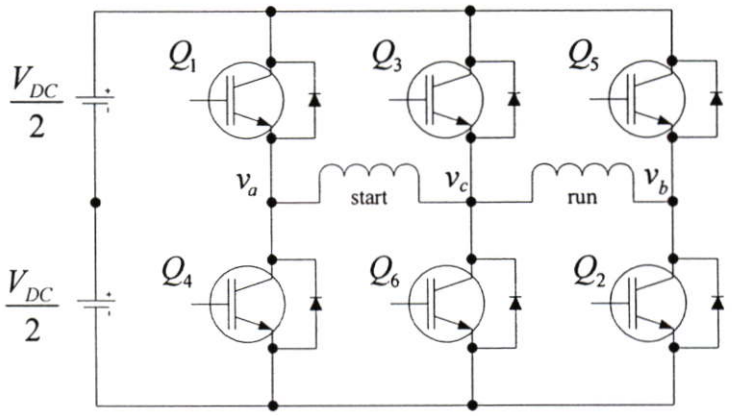
การออกแบบ วงจรขับสวิตช์ IGBT จะใช้ Opto Isolate เบอร์ TLP250 เพื่อแยกแรงดันไฟต่ำกับแรงดันไฟสูงออกจากกัน โดย R3U ในรูปที่ 3.7 จะใช้ค่าอยู่ระหว่าง 10Ω ถึง 100Ω เพื่อป้องกันการเกิดการแกว่งของสัญญาณที่ขาเกทของ IGBT ซึ่งในงานวิจัยนี้เลือกใช้ค่า R3U = 10Ω ซึ่งการแกว่งของสัญญาณเกิดจากตัวเก็บประจุที่ขาเกทของ IGBT มีค่า $2800pF$ หากเรามองจากเอาท์พุทของ Opto Isolate จะประกอบไปด้วยตัวเหนี่ยวนำแฝงที่เกิดจากลายวงจร และตัวเก็บประจุที่ขาเกทของ IGBT เป็นวงจร LC ซึ่งจะทำให้เกิดความถี่แกว่ง (damping frequency) เนื่องจากสัญญาณที่เอาท์พุทของ Opto Isolate เป็นสัญญาณพัลส์ PWM ที่ประกอบด้วยฮาร์มอนิกของสัญญาณไซน์หลายความถี่ ซึ่งอาจทำให้เกิดการออสซิลเลท ได้จึงจำเป็นต้องใส่ตัวต้านทาน R3U เพื่อควบคุมไม่ให้เกิดการแกว่งของสัญญาณโดยจะทำให้ผลของ L ในลายวงจรลดลงจะมีผลของ R และ C เท่านั้น โดยการเพิ่มขึ้นของแรงดัน และลดลงของระดับแรงดันพัลส์ของสัญญาณ PWM จะขึ้นกับค่าเวลาคงตัวของ R3U และ C ที่ขาเกทของ IGBT โดยมีค่าที่ $\tau = 28nS$

$$\tau = R_{3U} \cdot C_{GATE} \qquad (3.5)$$

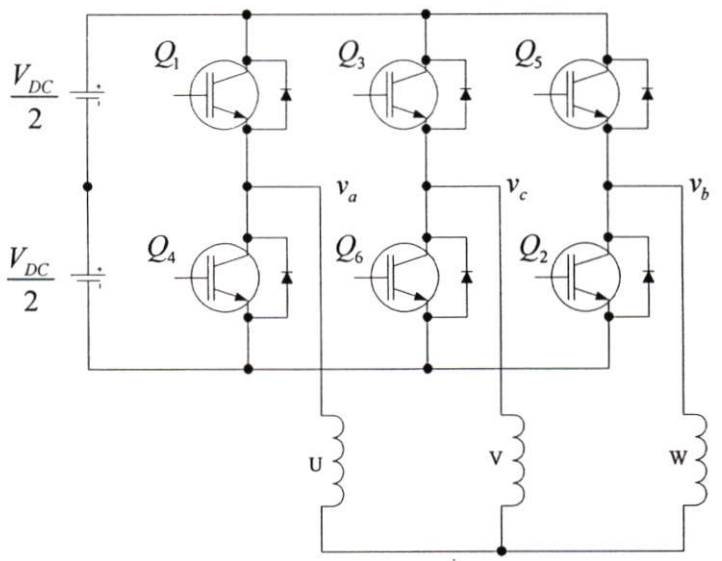
τ = ค่าเวลาคงตัว

R_{3U} = ค่าตัวต้านทาน

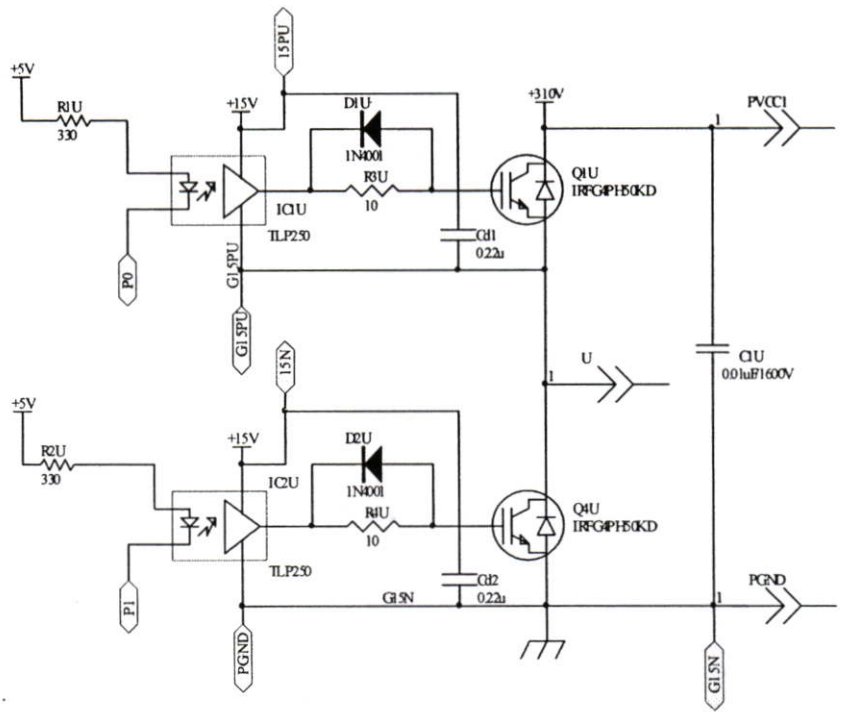
C_{GATE} = ค่าตัวเก็บประจุที่ขาเกทของ IGBT



รูปที่ 3.5 โครงสร้างของชุดขับสวิตช์ IGBT สำหรับมอเตอร์เหนี่ยวนำ 1เฟส



รูปที่ 3.6 โครงสร้างของชุดขับสวิตช์ IGBT สำหรับมอเตอร์เหนี่ยวนำ 3เฟส



รูปที่ 3.7 วงจรภาคขับสวิตช์ IGBT

ส่วนไดโอด D1U มีหน้าที่ช่วยในการคายประจุของตัวเก็บประจุที่ขาเกทของ IGBT ขณะสั่งให้ IGBT เป็นสภาวะ turn off โดยดึงประจุผ่านไดโอดผ่าน Opto Isolate ลง Ground ของวงจร

การออกแบบ R1U กำหนดให้กระแสที่ไหลผ่าน I_D มีค่าเท่ากับ $10mA$, $V_D = 1.6V$ โดยที่ $V_{CC} = 5V$

จะได้ว่า

$$R_{1U} = \frac{V_{CC} - V_D}{I_D} \tag{3.6}$$

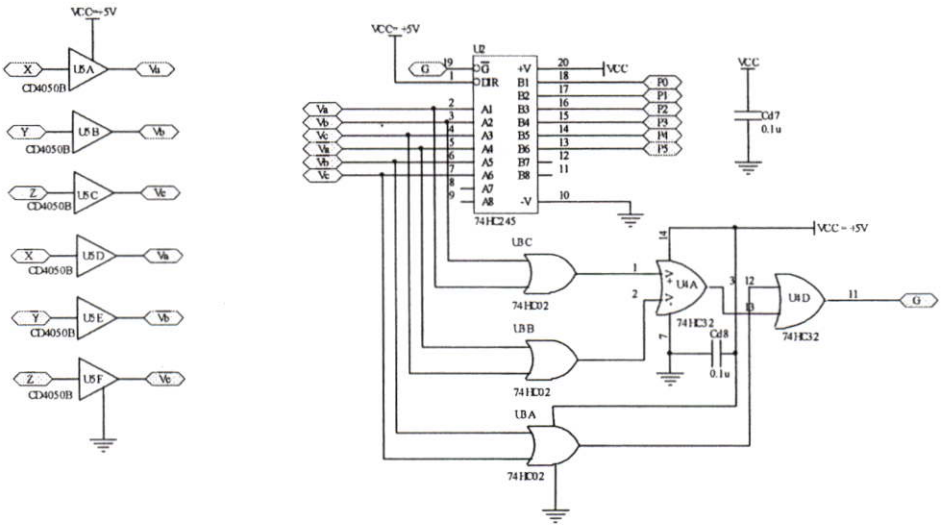
$$= \frac{5V - 1.6V}{10mA}$$

$$= 340\Omega$$

ในงานวิจัยนี้เลือกใช้ค่า 330 Ω ซึ่งจะได้กระแสประมาณ $10.3mA$ วงจรยังสามารถทำงานได้

3.3 การออกแบบวงจรแปลงแรงดัน และป้องกันการลัดวงจรของสวิตช์

จากรูปที่ 3.8 อุปกรณ์ FPGA ทำหน้าที่สร้างสัญญาณ SVM PWM โดยการคำนวณค่า และส่งผ่านวงจรแปลงแรงดันไฟฟ้าจาก 3.3V ไปเป็น 5V (CD4050B) และเชื่อมต่อกับวงจรบัฟเฟอร์ (74HC245) และเชื่อมต่อกับส่วนขับวงจร Opto isolate สำหรับนอร์เกตถูกนำมาเชื่อมต่อไว้เพื่อป้องกันกรณีที่สวิตช์ 2 ตัวที่อยู่ในกิ่งเดียวกันเกิดการทำงานพร้อมกัน ส่งผลให้เกิดการลัดวงจรขึ้น และทำความเสียหายให้กับตัวสวิตช์ โดยสาเหตุเกิดจากการที่ FPGA หยุดทำงาน พร้อมกับทำให้สภาวะที่พอร์ทเป็น 0 ทั้งคู่ ดังนั้นอุปกรณ์นอร์เกตถูกนำมาต่อเพื่อป้องกันปัญหาดังกล่าว



รูปที่ 3.8 วงจรแปลงระดับแรงดัน และป้องกันการลัดวงจรของสวิตช์

3.4 การออกแบบวงจรทวีแรงดัน

วงจรทวีแรงดันออกแบบสำหรับเมื่อต้องการเพิ่มแรงดันให้กับมอเตอร์เหนี่ยวนำเมื่อกรณีทำการเพิ่มความถี่หลักมูลสูงกว่าความถี่หลักมูลทำงานของมอเตอร์ โดยปกติจะมีความถี่หลักมูลทำงานที่ $50Hz - 60Hz$

ซึ่งหากเราทำการปรับความถี่ จะทำให้มีผลกับค่าความต้านทานของขดลวด(รีแอคแตนซ์) โดย

$$X_L = 2\pi fL \qquad (3.7)$$

X_L = ค่าความต้านทานของขดลวด (รีแอคแตนซ์)

f = ความถี่หลักมูล

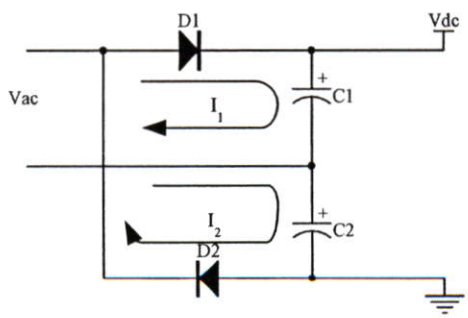
L = ค่าของตัวเหนี่ยวนำ

จากสมการที่ (3.7) จะเห็นได้ว่าเมื่อเปลี่ยนแปลงความถี่ให้มีค่าสูงขึ้นจะทำให้ค่า X_L มีค่าสูงขึ้นตามความถี่ ซึ่งจะทำให้กระแสที่จ่ายให้กับมอเตอร์ลดลง หากต้องการให้มอเตอร์ทำงานที่พิกัดกำลังเดิมจำเป็นต้องเพิ่มแรงดันให้กับมอเตอร์สูงขึ้น จึงมีความจำเป็นที่จะต้องมีวงจรทวีแรงดันเพื่อเพิ่มแรงดันให้กับมอเตอร์โดยทำการปรับขนาดแรงดันได้โดยการปรับจากค่าดรรชนีการมอดดูเลชั่น โดยปรับจากโปรแกรมในส่วนของ FPGA

ในทางกลับกันหากเราลดความถี่ลงจะทำให้ค่าของ X_L ลดลงตามจะทำให้กระแสที่จ่ายให้กับมอเตอร์เพิ่มมากขึ้นแต่ ไม่จำเป็นต้องปรับแรงดัน เนื่องจากค่าพิกัดกำลังงานของมอเตอร์จะขึ้นอยู่กับกระแสที่มอเตอร์ใช้ขณะนั้น ซึ่งต่างจากการทำงานที่ความถี่สูงเนื่องจาก กระแสจะถูกจำกัดโดยค่าของ X_L จำเป็นต้องเพิ่มแรงดันที่จ่ายให้มอเตอร์แทน

วงจรเร็กติไฟเออร์ และทวีแรงดันในตัว ที่ได้ออกแบบไว้แสดงในรูปที่ 3.9 การทำงานของวงจรพิจารณาได้จาก ถ้าแรงดันไฟฟ้ากระแสสลับ(V_{ac})มีค่าเป็นช่วงบวก จะส่งผลให้อุปกรณ์ไดโอด D1 ทำงาน(ON) แต่ D2 ไม่ทำงาน(OFF) โดยมีกระแส I_1 ไหลไปประจุที่ตัวเก็บประจุ C1 ทำให้แรงดันไฟฟ้าตกคร่อมมีค่าเท่ากับ V_{C1} และเมื่อแรงดันไฟฟ้ากระแสสลับ(V_{ac})มีค่าเป็นช่วงลบจะส่งผลให้อุปกรณ์ไดโอด D2 ทำงาน(ON) แต่ D1 ไม่ทำงาน(OFF) โดยมีกระแส I_2 ไหลไปประจุที่ตัวเก็บประจุ C2 ทำให้แรงดันไฟฟ้าตกคร่อมมีค่าเท่ากับ V_{C2}

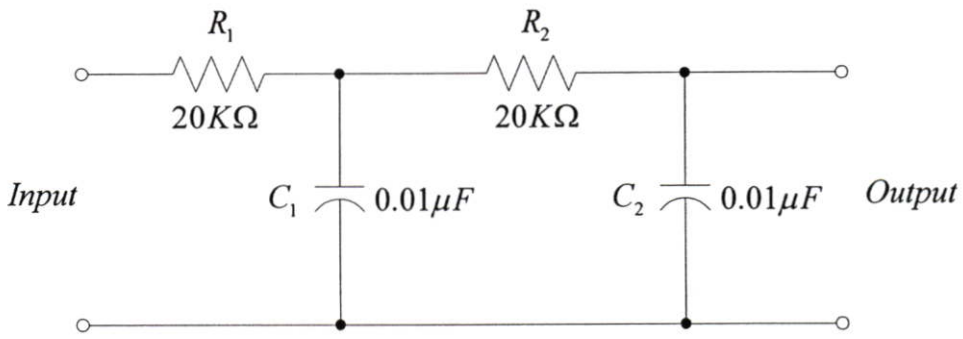
ถ้าพิจารณาในกรณีที่ $V_{C1} = V_{C2} = V_{ac}$ ทำให้แรงดันไฟฟ้าที่ตกคร่อมระหว่างตัวเก็บประจุทั้ง 2 ตัวมีค่าเท่ากับ $2V_{ac}$ และเป็นไฟกระแสตรง



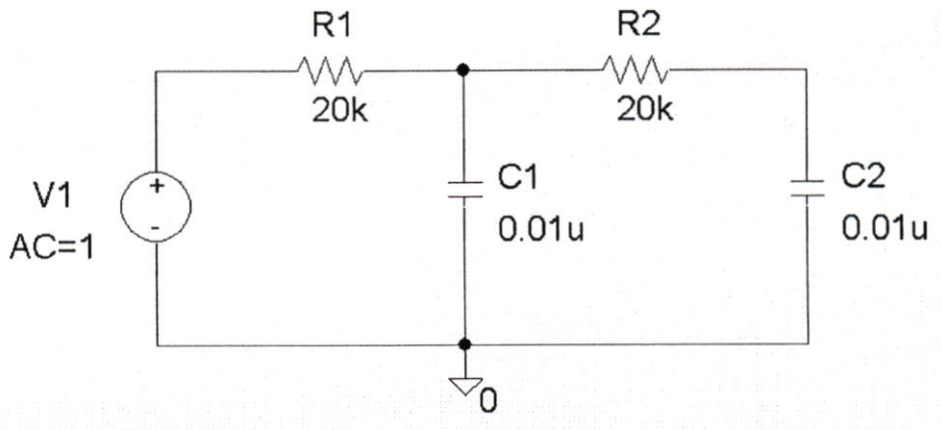
รูปที่ 3.9 วงจรเร็กติไฟเออร์และทวีแรงดัน

3.5 วงจรกรองความถี่ต่ำผ่าน

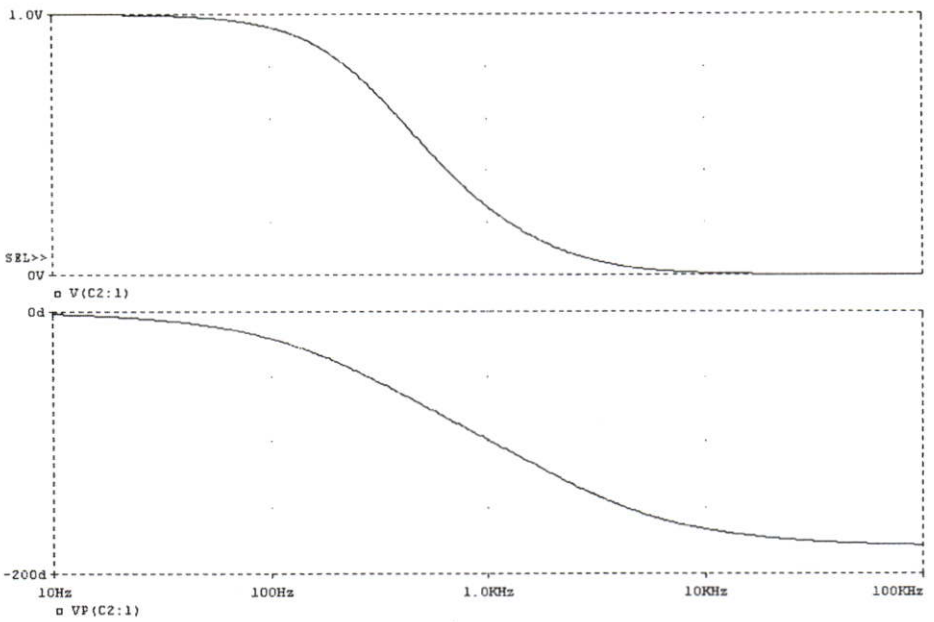
วงจรกรองความถี่ต่ำผ่านที่ใช้ในการวัดแรงดันระหว่างเฟส v_{AB} , v_{BC} และ v_{CA}



รูปที่ 3.10 วงจรกรองความถี่ต่ำผ่าน



รูปที่ 3.11 วงจรกรองความถี่ต่ำผ่านที่ใช้โปรแกรม PSPICE จำลองการทำงาน



รูปที่ 3.12 ผลตอบสนองความถี่ของวงจรกรองความถี่ต่ำผ่าน

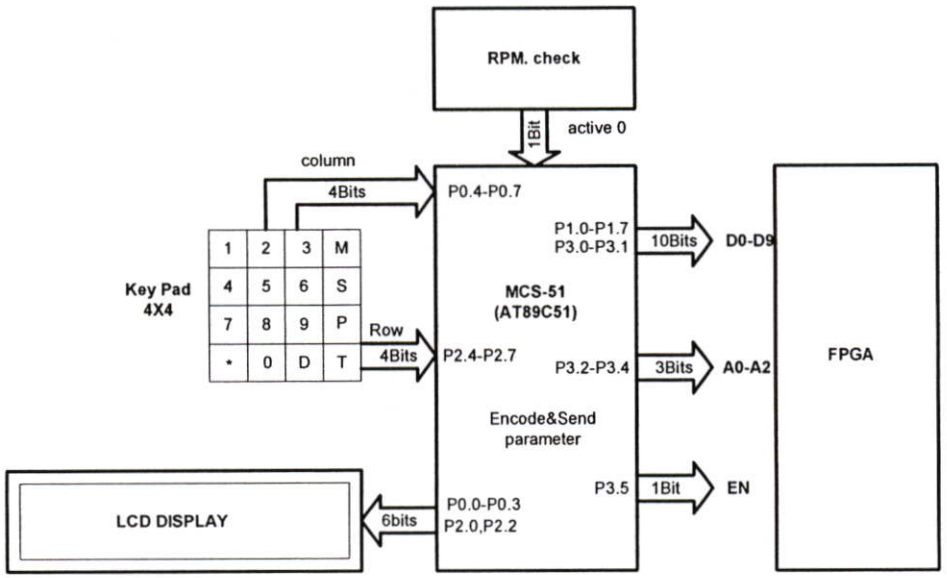
การจำลองการทำงานโดยโปรแกรม PSPICE สามารถวัดความถี่ตัดของวงจรกรองความถี่ต่ำผ่านได้ที่ $297Hz$

ตารางที่ 3.1 ตารางผลตอบสนองความถี่ของวงจรกรองความถี่ต่ำผ่าน โดย $v_{input} = 1V_P$

ความถี่อินพุท(Hz)	แรงดันเอาท์พุท (mV)	ความต่างเฟส(องศา)
20	997	-4.275
25	996	-5.443
30	994	-6.522
35	993	-7.549
40	991	-8.656
45	988	-9.679
50	986	-10.723
55	983	-11.796
60	980	-12.916

3.6 การออกแบบไมโครคอนโทรลเลอร์เพื่อรับค่าพารามิเตอร์

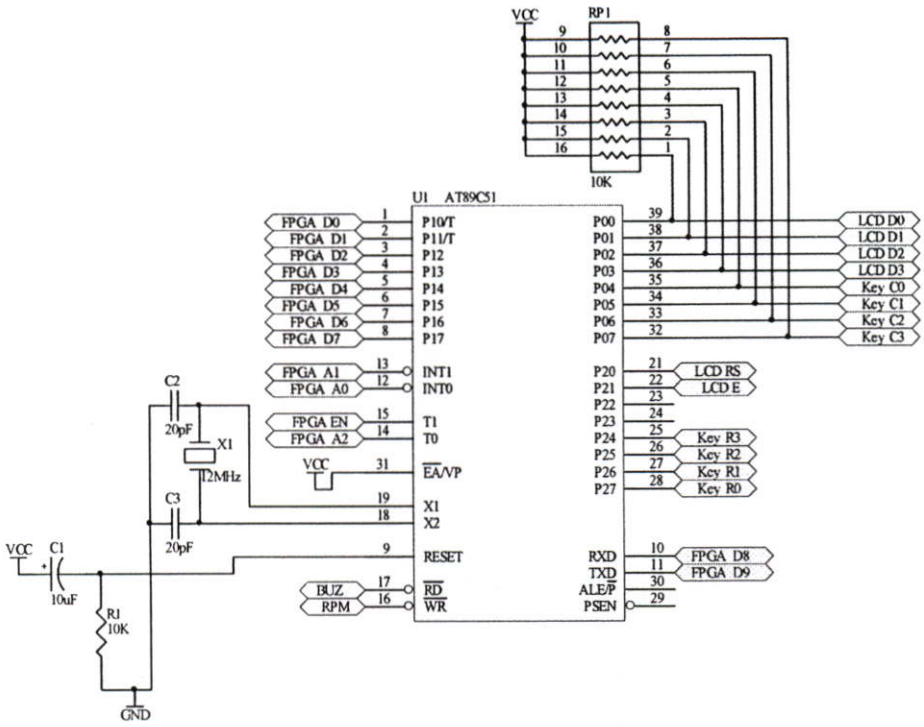
ดังแสดงในรูปที่ 3.13 เป็นการออกแบบส่วนรับค่าพารามิเตอร์เพื่อส่งให้ FPGA โดยในส่วนนี้จะทำการรับค่าจากคีย์บอร์ด ซึ่งมีจำนวน 16 คีย์ เพื่อใช้ในการตั้งค่าพารามิเตอร์ต่างๆ อาทิ เช่น Dead time , ดรรชนีการมอดดูเลต , ความถี่ในการสวิตช์ และแสดงผลผ่านจอแอลซีดี



รูปที่ 3.13 บล็อกไดอะแกรมส่วนรับ-ส่งค่าพารามิเตอร์เพื่อส่งให้ FPGA

3.7 การออกแบบส่วน FPGA

ในส่วนนี้จะเป็นการออกแบบ FPGA เพื่อทำหน้าที่สร้างสัญญาณ SVM PWM เพื่อใช้ควบคุมชุดขับ IGBT จะใช้ขาสัญญาณทั้งหมด 12 เส้นโดยแบ่งเป็นสัญญาณ SVM PWM ของมอเตอร์เหนี่ยวนำกระแสสลับแบบ 3เฟส 6เส้น และสัญญาณ SVM PWM ของมอเตอร์เหนี่ยวนำกระแสสลับแบบ 1เฟส อีก 6เส้น โดยจะมีการสร้างสัญญาณ 6 เส้น สำหรับขับมอเตอร์เหนี่ยวนำกระแสสลับแบบ 3เฟสคือ $Va,Vb,Vc,\overline{Va},\overline{Vb},\overline{Vc}$ โดยมีมุม $0^0,120^0,240^0$ ตามลำดับโดยขาสัญญาณที่ออกจาก FPGA คือขา 3 5 7 4 6 8 ตามลำดับ โดย $\overline{Va},\overline{Vb},\overline{Vc}$ เป็นสัญญาณกลับสถานะจาก Va,Vb,Vc โดย Va,Vb,Vc ใช้ขับ IGBT ชุดบน Q_1,Q_3,Q_5 ทั้งหมดและ $\overline{Va},\overline{Vb},\overline{Vc}$ ใช้ขับ IGBT ชุดล่าง Q_4,Q_6,Q_2 ทั้งหมด และอีก 6 เส้น สำหรับขับมอเตอร์เหนี่ยวนำกระแสสลับแบบ 1เฟสคือ $Va,Vb,Vc,\overline{Va},\overline{Vb},\overline{Vc}$ โดยมีมุม $0^0,\theta^0,180^0$ ตามลำดับโดยขาสัญญาณที่ออกจาก FPGA คือขา 13 15 17 14 16 18 ตามลำดับ และใช้ FPGA เป็นพอร์ทสำหรับการรับค่าอินพุทจากไมโครคอนโทรลเลอร์ MSC-51 จำนวน 14 พอร์ท และใช้เป็นส่วนของสัญญาณPWM ที่สร้าง



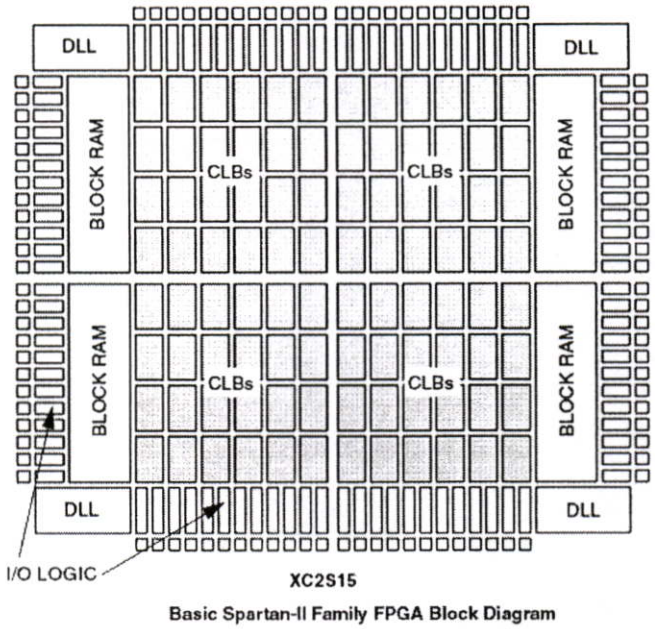
รูปที่ 3.14 วงจรไมโครคอนโทรลเลอร์ AT89C51

ตารางที่ 3.2 แสดงคุณสมบัติของ Spartan-II FPGA XC2S100

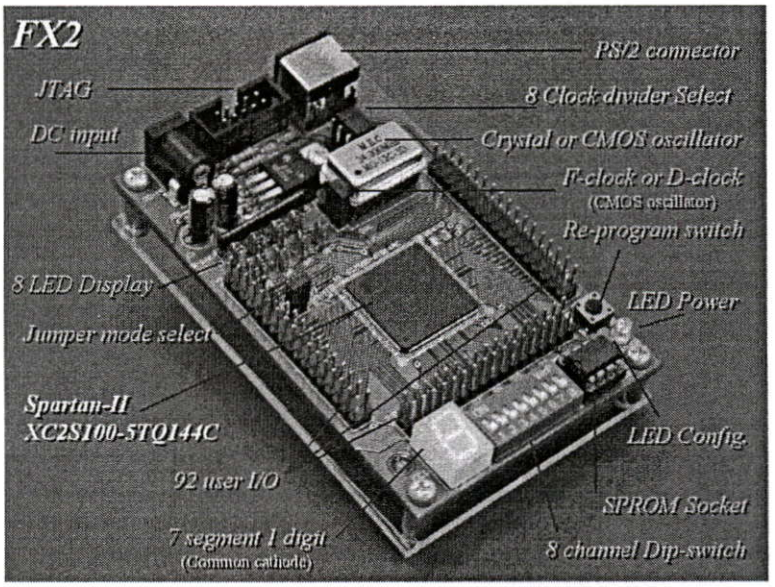
Logic cell	2700 cells
System Gates(Logic and RAM)	100000
CLB Array(R xC)	20x30
Total CLBs	600
Maximum Available User I/O	176
Total Distributed RAM Bits	38400
Total Block Bits	40K
Clock frequency operate	Up to 200MHz

ออกมาเพื่อขับ IGBT จำนวน 6 พอร์ทดังรูปที่ 3.14 โดยในส่วนนี้ได้เลือกใช้งานอุปกรณ์ FPGA ของ XILINX SPATAN II เบอร์ XC2S1000 ดังแสดงในรูปที่ 3.15 ส่วนรูปของบอร์ด FPGA แสดงในรูปที่ 3.16

คุณสมบัติของชิฟไอซี XILINX SPATAN II XC2S100 แสดงตามตารางที่ 3.2 และโครงสร้างภายในแสดงในรูปที่ 3.15



รูปที่ 3.15 แสดงบล็อกไดอะแกรมโครงสร้างภายใน

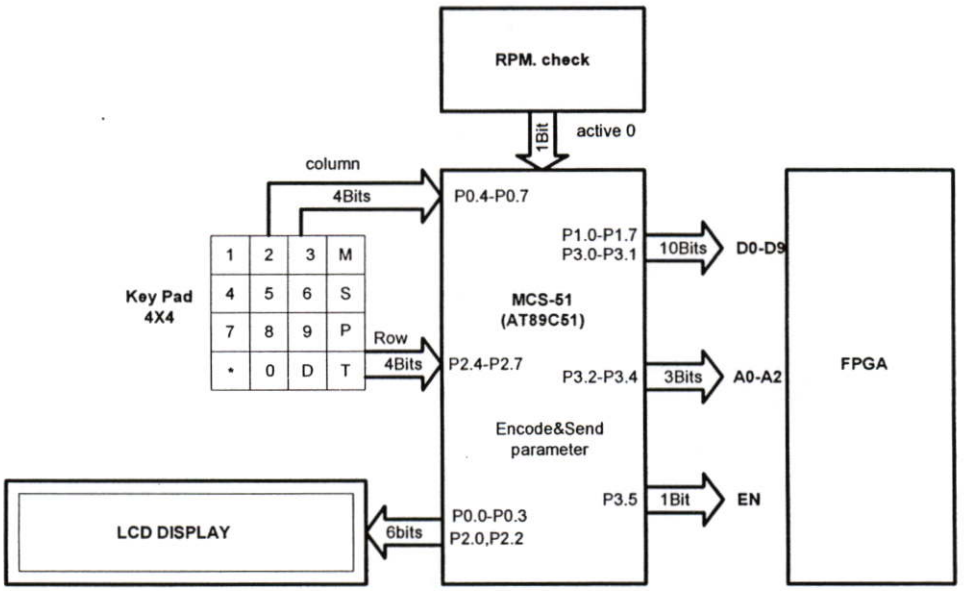


รูปที่ 3.16 บอร์ดอเนกประสงค์ที่ใช้สร้างสัญญาณ PWM

บทที่4
การออกแบบโปรแกรม

ในส่วนการออกแบบโปรแกรมเพื่อสร้างสัญญาณ SVM PWM นั้นจะประกอบไปด้วยสองส่วนได้แก่ ส่วนของไมโครคอนโทรลเลอร์สำหรับรับค่าอินพุทตัวแปร และส่วนของ FPGA ที่ถูกใช้งานด้านการคำนวณ และสร้างสัญญาณ SVM PWM เพื่อขับมอเตอร์

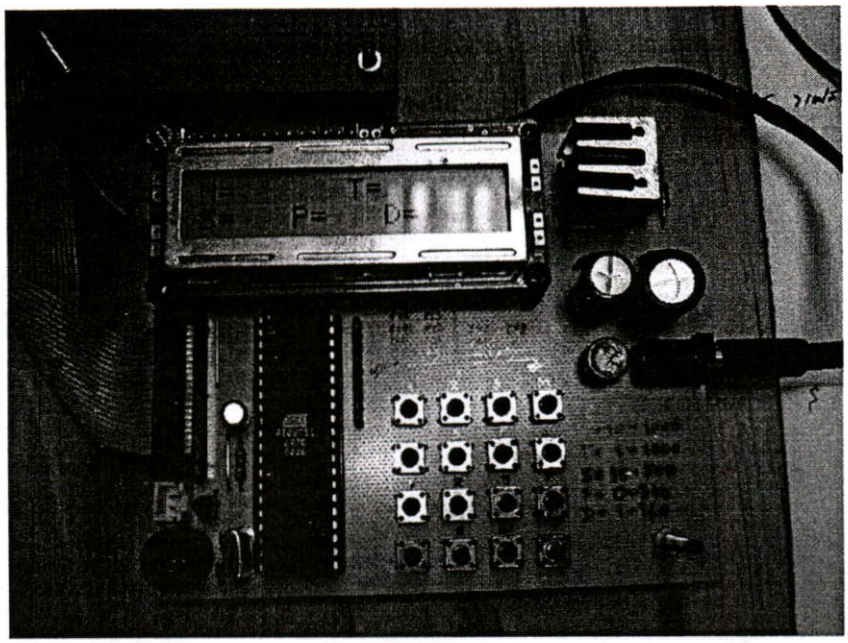
4.1 การออกแบบส่วนไมโครคอนโทรลเลอร์ส่งค่าตัวแปรให้ FPGA



รูปที่ 4.1 บล็อกไดอะแกรมส่วนรับ-ส่งค่าตัวแปรเพื่อส่งให้ FPGA

ไมโครคอนโทรลเลอร์ MCS-51(AT89C51) ใช้ในการติดต่อกับผู้ใช้งานโดยการตั้งค่าตัวแปรผ่านทางคีย์บอร์ด เพื่อตั้งค่าตัวแปรต่างๆที่ใช้สำหรับการควบคุมการทำงานของมอเตอร์ อาทิ เช่น ดรรชนีการมอดดูเลต , ความถี่มอเตอร์ , ความถี่การสวิตช์ และ Dead time โดยแสดงผลของการตั้งค่าบนจอแอลซีดี โดยค่าตัวแปรที่ทำการปรับตั้งค่าจะถูกส่งไปให้กับ FPGA เพื่อใช้ในการคำนวณเพื่อสร้างสัญญาณ SVM PWM โดยจะแบ่งข้อมูลที่ส่งให้ FPGA เป็น 3 ส่วนคือ

- ส่วนของค่าช่วงตัวแปร(DATA) โดยแสดงค่าช่วงตัวแปรดังตารางที่ 4.1 จำนวน 10 Bits
- ส่วนของการเลือกตัวแปร(ADDRESS) โดยแสดงดังตารางที่ 4.2 จำนวน 3 Bits



รูปที่ 4.2 บอร์ดไมโครคอนโทรลเลอร์ MCS-51 สำหรับรับ-ส่งค่าตัวแปร

ส่วนของสัญญาณ Enable เพื่อให้ FPGA รับข้อมูลที่ส่งจากไมโครคอนโทรลเลอร์ MCS-51 จำนวน 1 Bit

ตารางที่ 4.1 กำหนดตัวแปรที่แสดงบนจอแอลซีดี และช่วงค่าตัวแปร

ตัวแปรใน MCS-51	ตัวแปรใน FPGA	ค่าช่วงตัวแปร ที่MCS-51 ส่งให้ FPGA	ความหมายตัวแปร
M	m_index	1-1000	ตรรชนีการมอดดูเลต
S	step	1-500	ความถี่มอเตอร์
T	Ts	1-1000	ความถี่การสวิตช์
P	pshift	1-360	มุมแรงดันระหว่างเฟส Aและ B
D	dtime	1-100	Dead time

การกำหนดพอร์ตข้อมูล

จากตารางที่ 4.1

1) ค่าช่วงตัวแปร ที่ส่งให้กับ FPGA อยู่ในช่วง 0-1000 โดยใช้ 10 Bits ดังนั้นจำเป็นต้องใช้พอร์ตของ MCS-51 ทั้งหมด 10พอร์ตโดยที่ใช้งานพอร์ต P1.0 ถึง P1.7 และ P3.0 P3.1 ใน MCS-51 ดังรูปที่ 4.1 โดย P1.0 เป็น Bit ค่าน้อยสุด หรือ LSB และ P3.1 เป็น Bit ค่ามากสุด หรือ MSB

2) ค่าตัวแปรที่ปรับค่าได้มี 5 ตัวแปรดังนั้น กำหนดการเลือกปรับค่าตัวแปรแต่ละตัวตามค่าเลขฐานสองโดยใช้ทั้งหมด 3 Bits จำนวน 3 พอร์ตโดยที่ใช้งานพอร์ต P3.2 ถึง P3.4 ใน MCS-51 ดังรูปที่ 4.1 โดย P3.2 เป็น Bit ค่าน้อยสุด หรือ LSB และ P3.4 เป็น Bit ค่ามากสุด หรือ MSB

ตารางที่ 4.2 ตัวแปรเทียบกับค่าเลขฐานสอง

ตัวแปร	ค่าเลขฐานสอง
m_index	000
step	001
pshift	010
T	011
dtime	100

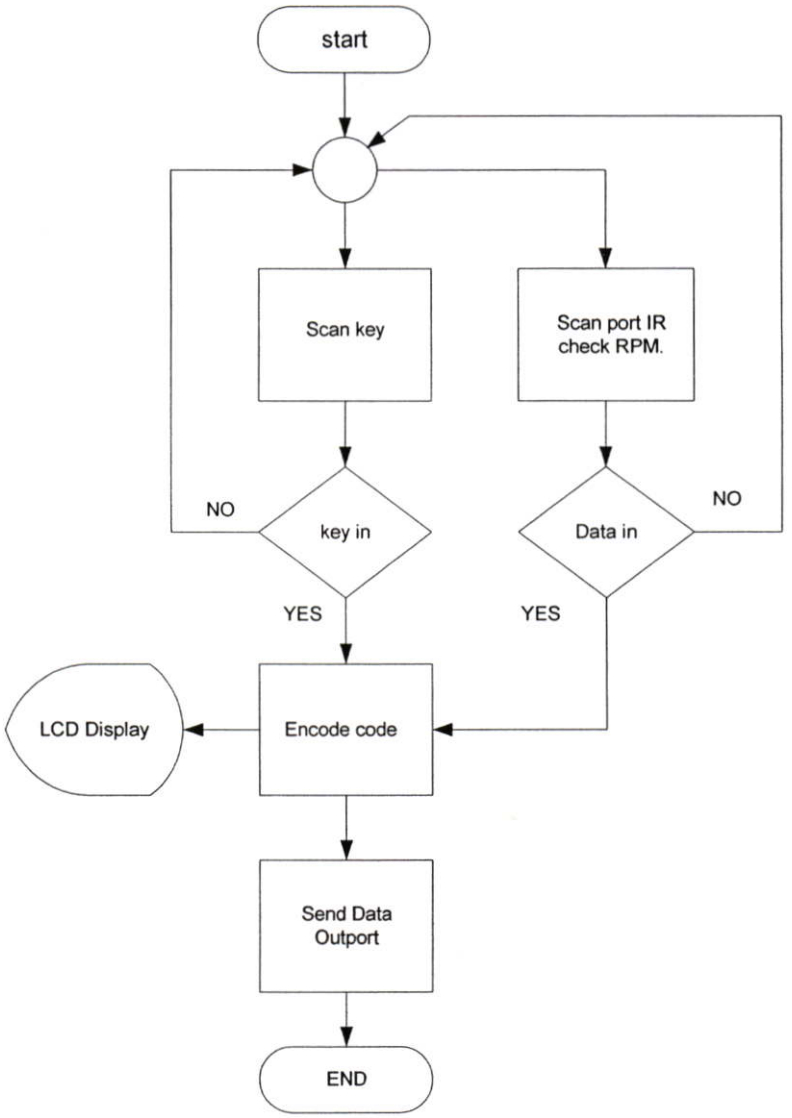
3) บิท Enable สำหรับการตรวจสอบค่าที่ส่งมาถูกต้องหรือไม่ โดยปกติจะมีค่าสภาวะลอจิกเป็น 1 และเมื่อมีการเปลี่ยนแปลงค่าตัวแปร และค่าช่วงตัวแปร จึงจะมีค่าสภาวะลอจิกเป็น 0 เพียงช่วงเวลา 1ms เท่านั้นแล้วจะกลับมาเป็นสภาวะลอจิก 1 เหมือนเดิมโดย กำหนดให้ FPGA ทำการอ่านค่าที่พอร์ตนี้ โดยกำหนด เป็น active 0 เท่านั้น FPGA จึงจะรับค่าตัวแปรเข้าไปคำนวณโดยการออกแบบ จะให้ FPGA จะทำการวนอ่านค่าที่พอร์ตนี้ ถ้าสภาวะลอจิกเป็น "0" ก็จะดำเนินการรับค่าจาก พอร์ต data 10 Bits และ select 3 bits เพื่อเก็บค่าเข้าไปคำนวณเพื่อสร้างสัญญาณ SVM PWM โดยที่ใช้งานพอร์ต P3.5 ของ MCS-51 เป็นบิท Enable ดังรูปที่ 4.1

4) คีย์บอร์ดใช้สำหรับป้อนค่าช่วงตัวแปรและตัวแปร โดยตัวคีย์บอร์ดมีจำนวน 16 คีย์และมีโครงสร้างในรูปแบบ 4 แถว และ 4 หลัก ซึ่ง MCS-51 จะต้องใช้พอร์ตจำนวน 8 พอร์ตในการเชื่อมต่อกับคีย์บอร์ดโดยที่ใช้งานพอร์ต P0.4 ถึง P0.7 เป็นตำแหน่งหลัก และพอร์ต P2.4 ถึง P2.7 เป็นตำแหน่งแถว ของ MCS-51 ดังรูปที่ 4.1

5) แอลซีดี ใช้แสดงผลของการตั้งค่าตัวแปร โดยเชื่อมต่อกับพอร์ตของ MCS-51 จำนวน 6 พอร์ต โดยที่ใช้งานพอร์ต P0.0 ถึง P0.3 เป็น LCD data(ข้อมูลที่ส่งให้ LCD) และP2.0 เป็น LCDRS(เป็นสัญญาณการเขียนและอ่าน LCD) P2.1 เป็น LCDE (เป็นสัญญาณ Enable จอ LCD) ดังรูปที่ 4.1

4.2 โปรแกรมส่วนของไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์ MCS-51 ถูกนำมาใช้ในการติดต่อกับผู้ใช้งาน โดยแสดงดังรูปที่ 4.3 เริ่มต้นการทำงานของโปรแกรมจะทำการรอรับค่าจากคีย์บอร์ด เมื่อมีการป้อนค่าโดยให้ป้อนค่าตัวแปร แล้วตามด้วยค่าช่วงของตัวแปร โดยป้อนเป็นเลขฐาน 10 เมื่อรับค่าจะคีย์บอร์ดแล้ว ไมโครคอนโทรลเลอร์จะทำการเปลี่ยนข้อมูลที่ป้อนเป็นเลขฐานสอง เพื่อส่งให้กับ FPGA และจะแสดงค่าตัวแปร และค่าช่วงตัวแปรที่เป็นเลขฐาน10 ที่ทำการตั้งค่าแสดงบนจอ LCD ด้วย

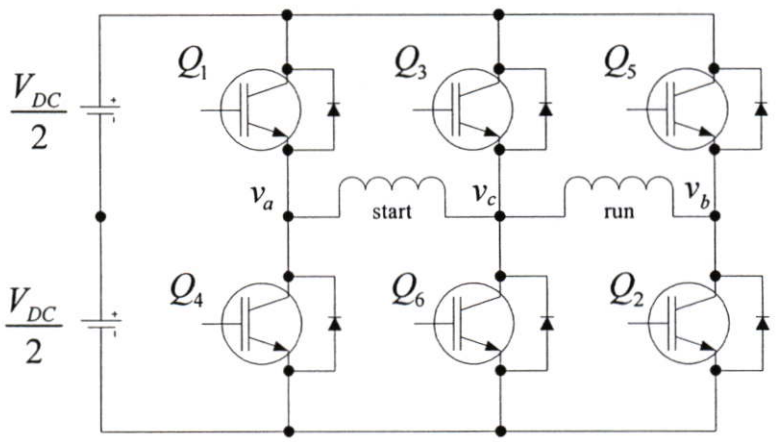


รูปที่ 4.3 แผนภาพแสดงการทำงานของโปรแกรมในส่วน MCS-51

4.3 การออกแบบสัญญาณ SVM PWM ของ FPGA

4.3.1 การขับมอเตอร์เหนี่ยวนำกระแสสลับชนิด 1 เฟสโดยอินเวอร์เตอร์ 3 เฟส

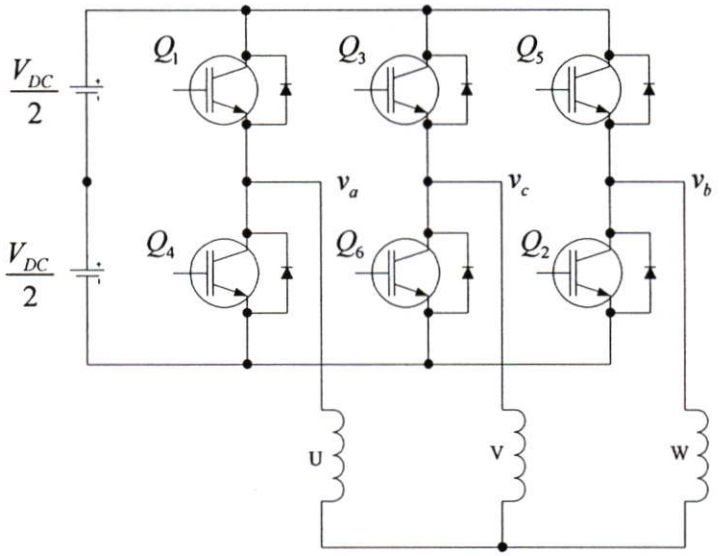
โครงสร้างวงจรอินเวอร์เตอร์ 3เฟส ที่ใช้ขับมอเตอร์เหนี่ยวนำกระแสสลับชนิด 1เฟส ดังแสดงในรูปที่ 4.4 โดย แรงดัน v_a คือแรงดันเฟสมีมุม 0^0 , v_b คือแรงดันเฟสมีมุม 180^0, v_c คือแรงดันเฟสมีมุม θ^0 และ IGBT แต่ละตัวถูกควบคุมด้วยสัญญาณ SVM PWM จาก FPGA โดยชุดบน Q_1, Q_3, Q_5 คือ a, b, c และชุดล่าง Q_4, Q_6, Q_2 คือ $\bar{a}, \bar{b}, \bar{c}$



รูปที่ 4.4 วงจรขับอินเวอร์เตอร์ สำหรับมอเตอร์เหนี่ยวนำกระแสสลับ 1 เฟส

4.3.2 การขับมอเตอร์เหนี่ยวนำกระแสสลับชนิด 3 เฟสโดยอินเวอร์เตอร์ 3 เฟส

โครงสร้างวงจรอินเวอร์เตอร์ 3เฟส ที่ใช้ขับมอเตอร์เหนี่ยวนำกระแสสลับชนิด 3เฟส ดังแสดงในรูปที่ 4.5 โดย แรงดัน v_a คือแรงดันเฟสมีมุม 0^0 , v_b คือแรงดันเฟสมีมุม 120^0, v_c คือแรงดันเฟสมีมุม 240^0 และ IGBT แต่ละตัวถูกควบคุมด้วยสัญญาณ SVM PWM จาก FPGA โดยชุดบน Q_1, Q_3, Q_5 คือ a, b, c และชุดล่าง Q_4, Q_6, Q_2 คือ $\bar{a}, \bar{b}, \bar{c}$

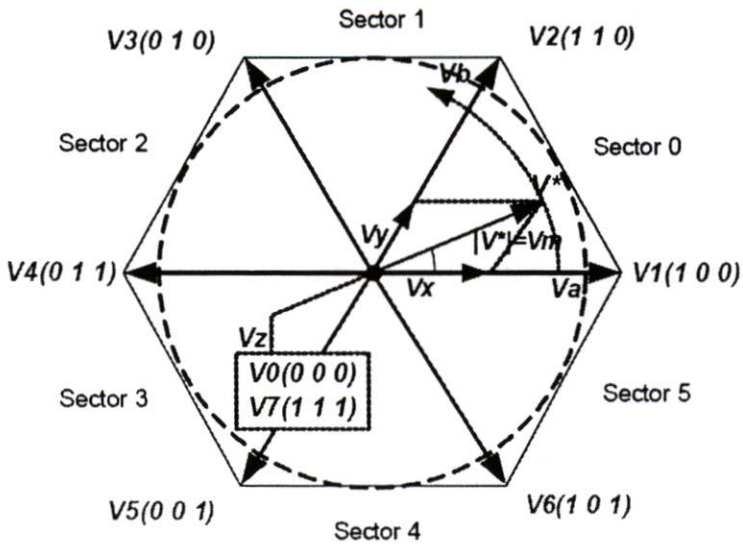


รูปที่ 4.5 วงจรขับอินเวอร์เตอร์ สำหรับมอเตอร์เหนี่ยวนำกระแสสลับ 3 เฟส

4.3.3 หลักการควบคุมแบบ Vector Modulation

หลักการทำงานของ Space vector PWM อธิบายได้โดย เวคเตอร์แรงดัน (**V0,..,V7**) แสดงในรูปที่ 4.6 ซึ่งประกอบไปด้วยเวคเตอร์ที่มีค่าจำนวนหกเวคเตอร์ (**V1,..,V6**) และสองเวคเตอร์ศูนย์ (**V0,V7**) ซึ่งยังคงใช้ในส่วนของ เวลาการสวิตช์(Ts) พื้นฐานของเทคนิกการควบคุมแบบ Vector Modulation เป็นการการบวกของเวคเตอร์ศูนย์กับเวคเตอร์อีก 2 ตัวใกล้เคียง โดยจะได้ผลเป็นเวคเตอร์ตัวใหม่ $\mathbf{V}^*$ ดังแสดงในรูปที่ 4.6 โดยค่าของ $\mathbf{V}^*$ จะมีค่าคงที่ตลอดช่วงเวลาคาบการสุ่ม T

ตัวอย่างเช่นเวคเตอร์ $\mathbf{V}^*$ ใน Sector 0 จะประกอบไปด้วยเวคเตอร์ศูนย์ **Vz** และเวคเตอร์ที่มีค่า **Vx,Vy** ซึ่งเป็นองค์ประกอบของ **Va** และ **Vb** ตามลำดับ ดังนั้นใน Sector 0 จะมีค่าดังนี้ **V1(100), V2(110)** โดยตัวเลข 1 และ 0 แสดงสถานการณ์ เปิด ปิดตามลำดับ ของชุด IGBT ชุดบน Q_1, Q_3, Q_5 ของวงจรอินเวอร์เตอร์ 3 เฟส



รูปที่ 4.6 เวคเตอร์แรงดันไฟฟ้าที่สร้างโดยอินเวอร์เตอร์

เวคเตอร์ $\mathbf{V}^*$ อธิบายได้ดังสมการ

$$\mathbf{V}^* = \mathbf{Va}\frac{ta}{T} + \mathbf{Vb}\frac{tb}{T} + \mathbf{Vz}\frac{tz}{T} \tag{4.1}$$

โดยที่ ta, tb, tz เป็นเวลาในการสร้างสัญญาณเวลาการสวิตช์เวคเตอร์แรงดัน $\mathbf{Va}, \mathbf{Vb}, \mathbf{Vz}$ ผลรวมของเวลาจะเป็นคาบเวลาการสวิตช์

$$ta + tb + tz = T = \frac{Ts}{2} \tag{4.2}$$

เมื่อ $\mathbf{Vz} = 0$ และ $\mathbf{V}^* = \mathbf{Vx} + \mathbf{Vy}$ สมการของ $\mathbf{V}^*$ เขียนใหม่ได้เป็น

$$\mathbf{V}^* = \mathbf{Vx} + \mathbf{Vy} = (\mathbf{Va}\frac{ta}{T} + \mathbf{Vb}\frac{tb}{T}) \tag{4.3}$$

ดังนั้น

$$ta = \frac{|\mathbf{Vx}|}{|\mathbf{Va}|}T \quad , tb = \frac{|\mathbf{Vy}|}{|\mathbf{Vb}|}T \tag{4.4}$$

จากตรีโกณมิติ ความสัมพันธ์ของ $\mathbf{Vx}, \mathbf{Vy}$ สามารถแสดงได้เป็น

$$\mathbf{Vx} = V_m(\cos\alpha - \frac{1}{\sqrt{3}}\sin\alpha) \qquad (4.5)$$

$$\mathbf{Vy} = \frac{2}{\sqrt{3}}V_m \sin(\alpha)$$

ถ้า $|\mathbf{Va}| = |\mathbf{Vb}| = \frac{2}{3}Vdc$ [1] , เมื่อทำการแทนค่า **Vx, Vy** ลงในสมการ (4.4) ค่าของ ta, tb, tz จะมีค่าดังนี้

$$ta = \frac{3}{4}M(\cos\alpha - \frac{1}{\sqrt{3}}\sin\alpha)T$$

$$tb = \frac{\sqrt{3}}{2}M(\sin\alpha)T \qquad (4.6)$$

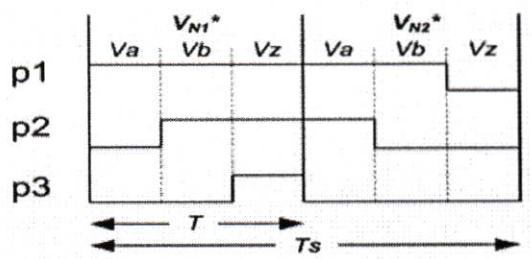
$$tz = T - ta - tb$$

เมื่อ M = modulation index = $V_m/(V_{dc}/2)$

$T = ta + tb + tz = Ts/2$ (Ts = switching time)

α = vector angle in each sector , $0^\circ < \alpha < 60^\circ$

วิธีการลดทอนผลของการสูญเสียในสวิตช์ เป็นวิธีที่ได้รับความนิยมแพร่หลายในการออกแบบวงจรอินเวอร์เตอร์ โดยอาศัยคุณสมบัติของ เวคเตอร์ศูนย์ **V0**(000) และ **V7**(111) จากตัวอย่างในรูปที่ 4.7 เวคเตอร์ศูนย์ **V7** มีค่ามากกว่า **V0** ซึ่งเป็นการใช้สร้าง $\mathbf{V_{N1}}$. ใน Sector 0 โดยสังเกตได้ว่าจะมีเฉพาะ P3 ที่สวิตช์จากสภาวะลอจิกต่ำเป็นลอจิกสูง

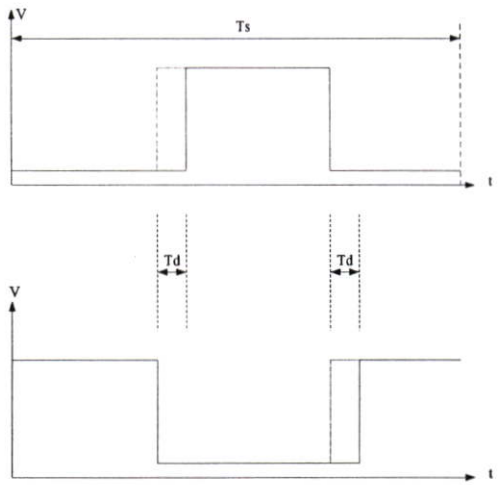


รูปที่ 4.7 กราฟแสดงการสวิตช์

4.4 ออกแบบและพัฒนาส่วนซอฟต์แวร์ควบคุมวงจรอินเวอร์เตอร์

4.4.1 Dead Time

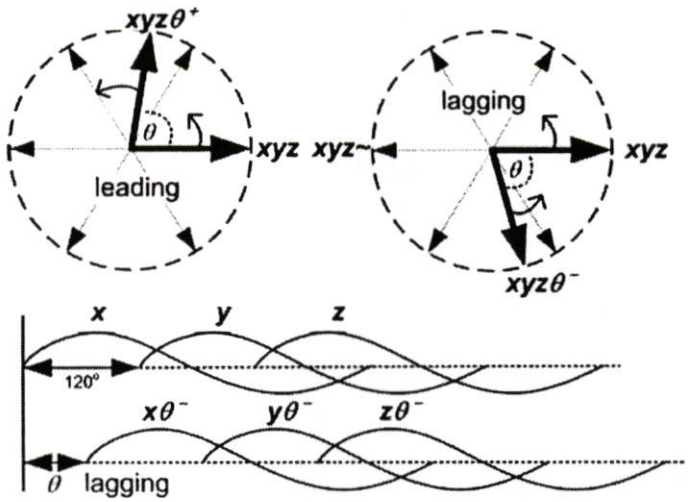
เงื่อนไขการสวิตช์ในทางอุดมคตินั้น สัญญาณควบคุม IGBT ที่กิ่งเดียวกันจะมีค่าตรงกันข้ามกัน (Complementary) เช่นกิ่งสวิตช์ Q_1, Q_4 ในรูปที่ 4.5 แต่ในทางปฏิบัตินั้นเป็นไปไม่ได้ เนื่องจาก IGBT หรือสวิตช์กำลังทางไฟฟ้านั้นจะมีช่วงเวลาในการทำงาน และหยุดทำงาน ช่วงหนึ่ง โดยช่วงเวลาเริ่มการทำงานของสวิตช์จะมีค่าน้อยกว่าช่วงเวลาที่ให้หยุดทำงาน ถ้าหากให้สัญญาณควบคุม สวิตช์ Q_1 หยุดทำงาน และ Q_4 เริ่มทำงานที่เวลาเดียวกัน ในทางปฏิบัติจะทำให้มีช่วงเวลาที่สวิตช์ทั้งสองยังคงทำงานอยู่ จะทำให้เกิดการลัดวงจรของแหล่งจ่ายกำลังงานได้จะทำให้เกิดความเสียหายกับสวิตช์ หรือแหล่งจ่ายได้ ดังนั้นเราจึงต้องป้องกันการลัดวงจรดังกล่าวด้วยการหน่วงเวลาของสัญญาณเพื่อให้ Q_1 หยุดทำงาน และเริ่มให้ Q_4 ทำงานหลังจากหน่วงเวลาดังรูปที่ 4.8



รูปที่ 4.8 สัญญาณ SVM PWM ที่มี delay time

4.4.2 ออกแบบและพัฒนาส่วนซอฟต์แวร์ควบคุมวงจรอินเวอร์เตอร์สำหรับมอเตอร์ 1เฟส

การสร้างสัญญาณ เลื่อนเฟส SVM PWM นั้นใช้วิธี Vector Modulation จากรูปที่ 4.9 เมื่อ Sector เริ่มต้นของเวคเตอร์ที่เลื่อนเฟส $(xyz\theta)$ คำนวณโดยพิจารณาจากเงื่อนไขการนำหน้า และตามหลังเฟส เทียบกับเวคเตอร์เริ่มต้น โดย $xyz, xyz\sim, xyz\theta$ มีค่าเฟสของเวคเตอร์ $0^\circ, 180^\circ, \theta^\circ$ ตามลำดับ



รูปที่ 4.9 Vector modulation และการเลื่อนเฟส

ดังแสดงในรูปที่ 4.10 เป็นฟังก์ชั่นบล็อคในระบบ ซึ่งทั้งหมดเขียนโดยใช้ภาษามาตราฐานของ Verilog โดยค่าตัวแปรอินพุทที่ต้องการประกอบไปด้วย Dead time (DTIME), ดรรชนีการมอดดูเลต (MOD) ระยะเวลาการสุ่ม (Ts) มุมของเฟสเลื่อน (θ°) Memory incremental step (STEP) โดยการออกแบบนี้ ค่าดรรชนีการมอดดูเลต และ มุมของเฟสเลื่อน นั้นสามารถหาความสัมพันธ์กับระบบที่ออกแบบ สำหรับค่าตัวแปรอินพุท ซึ่งได้แก่คือ Ts , STEP, dead time สามารถคำนวณได้จาก

$$T_S = \frac{F_{clk}}{2 \times Fs} \tag{4.7}$$

$$STEP = \frac{F_{fund} \times D_{mem} \times 6}{2 \times Fs}$$

$$td = \frac{2 \cdot dtime}{f_{clk}}$$

เมื่อ T_S = คาบเวลาของความถี่การสวิตช์

F_{clk} = สัญญาณนาฬิกาของ FPGA

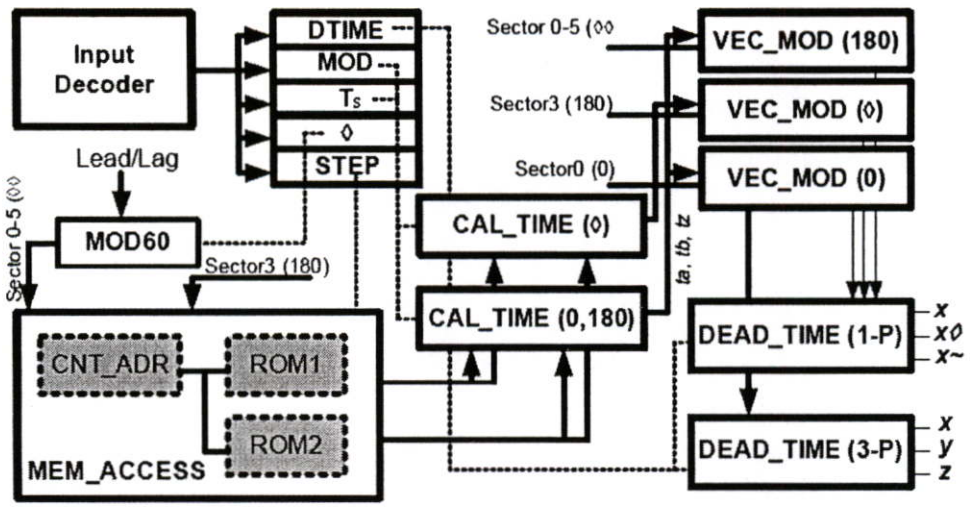
Fs = ความถี่การสวิตช์

F_{fund} = ความถี่หลักมูลของไฟฟ้าสลับที่จ่ายให้กับมอเตอร์

D_{mem} = ความละเอียดของ ROM = 512 ระดับ (9 Bit)

td = ช่วงเวลาของ dead time

$dtime$ = ตัวแปรสำหรับปรับค่า dead time โดยมีค่าช่วง 0-100



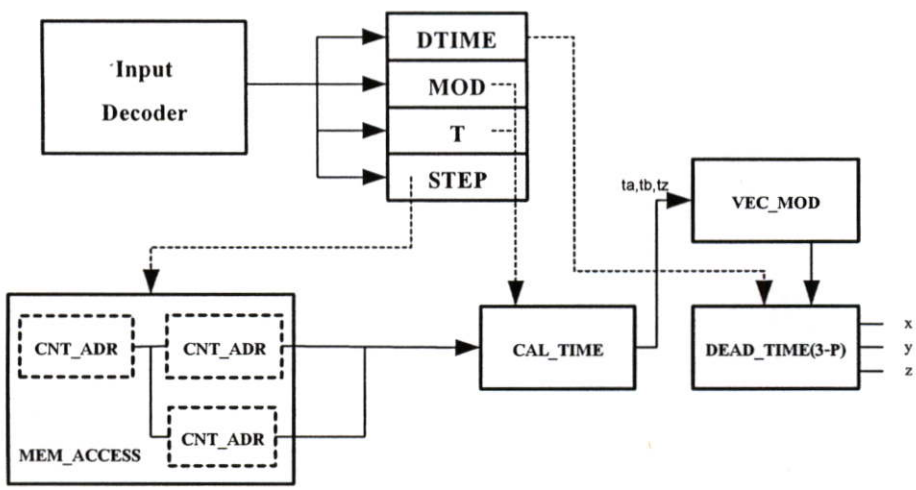
รูปที่ 4.10 ฟังก์ชั่นบล็อคในการออกแบบระบบ

MOD60 ใช้ในการหา Start sector ของเวคเตอร์เฟสเลื่อน ซึ่งขึ้นอยู่กับค่า Lag/Lead ที่กำหนดโดยตัวแปร θ° เมื่อทราบ Start sector แล้วก็จะทำการหาค่า ตำแหน่งAddress ที่มีความสัมพันธ์กับค่า STEP ซึ่งROM1 และ ROM2 จะเก็บค่า $\frac{3}{4}(\cos\alpha - \frac{1}{\sqrt{3}}\sin\alpha)$ และ $\frac{\sqrt{3}}{2}\sin(\alpha)$ ตามลำดับ ($0 < \alpha < 60$) โดยค่าคงที่ ที่ได้จะนำไปใช้ในการคำนวณ ta, tb, tz ตามสมการที่ (4.6) หน่วยความจำจะส่งค่าของเวคเตอร์ $xyz, xyz\sim, xyz\theta$ โดย Multiplex มากับ Address จากนั้น CAL_TIME จะทำการคำนวณค่า ta, tb, tz แยกกันสำหรับแต่ละเวคเตอร์ $xyz, xyz\sim, xyz\theta$ ต่อจากนั้น VEC_MOD ทั้งสามส่วนจะทำการสร้างรูปแบบสัญญาณ SVM PWM สำหรับแต่ละเฟส สุดท้ายก่อนจะส่งสัญญาณ SVM PWM ไปควบคุม transistor ในวงจรอินเวอร์เตอร์ 3 เฟส จะทำการแทรก Dead time เข้าไปเพื่อป้องกันการลัดวงจร ในรูปที่ 4.10 สัญญาณ SVM PWM (x, y, z) ใช้ควบคุมมอเตอร์ 3 เฟส และสัญญาณ SVM PWM ($x, x\theta, x\sim$)ใช้ควบคุมมอเตอร์ 1 เฟส

4.4.3 ออกแบบและพัฒนาส่วนซอฟต์แวร์ควบคุมวงจรอินเวอร์เตอร์สำหรับมอเตอร์ 3เฟส

จากหลักการของวิธี SVM PWM จะเห็นได้ว่าการสร้างเวคเตอร์รับที่เกิดจากเวคเตอร์แรงดันสองตัวจะทำให้สามารถควบคุมแรงดัน 3เฟสได้โดยการแปลงเวคเตอร์ให้เป็นคาบเวลาในการสร้างสัญญาณ SVM PWM โดยจะได้สัญญาณควบคุมแรงดัน 3เฟส ได้พร้อมกัน โดยการทำงานในส่วนของโปรแกรมที่สร้างสัญญาณ SVM PWM สำหรับมอเตอร์ 3เฟสแสดงดังรูปที่ 4.11 การทำงานคล้ายคลึงกับโปรแกรมในส่วนของการควบคุมมอเตอร์1เฟส แตกต่างกันเพียงไม่ต้อง

สร้างสัญญาณ SVM PWM 3ชุด แบบใน 1เฟส โดยอธิบายการทำงานของโปรแกรมในระบบ3เฟส ได้ดังนี้ ในส่วนของ Input Decoder ใช้สำหรับรับค่าตัวแปร และค่าช่วงตัวแปรจากไมโครคอนโทรลเลอร์ และทำการเข้ารหัสเพื่อปรับตัวแปรที่ต้องการโดย ตัวแปร DTIME ใช้สำหรับการปรับค่า Deadtime ในส่วนของบล๊อค DEAD_TIME(3-P) เพื่อแทรก delay time เข้าไปกับสัญญาณ SVM PWM ในส่วนของตัวแปร MOD ใช้สำหรับปรับค่า modulation index ในส่วนของตัวแปร Ts ใช้ในการปรับค่าความถี่ในการสวิตช์โดยเป็นการปรับคาบเวลาของสัญญาณที่ได้มาจากบล๊อค MEM_ACCESS ส่วนตัวแปร STEP ใช้ในการปรับค่าความถี่ของมอเตอร์และความละเอียดของมุมที่เวคเตอร์เลื่อนโดย บล๊อค MEM_ACCESS มีหน้าที่ในการคำนวณค่าคงที่ และการหาค่า ตำแหน่งAddress ที่มีความสัมพันธ์กับค่า STEP ซึ่งROM1 และ ROM2 จะเก็บค่า $\frac{3}{4}(\cos\alpha - \frac{1}{\sqrt{3}}\sin\alpha)$ และ $\frac{\sqrt{3}}{2}\sin(\alpha)$ ตามลำดับ $(0 < \alpha < 60)$ โดยค่าคงที่ ที่ได้จะนำไปใช้ในการคำนวณ ta, tb, tz ตามสมการที่ (4.6) ซึ่งจะคำนวณโดยบล๊อค CAL_TIME จากนั้นสัญญาณสัญญาณ ta, tb, tz ที่ได้จะส่งเข้าไปยังบล๊อค VEC_MOD เพื่อทำการกำหนด Sector เริ่มต้นและสร้างรูปแบบการสวิตช์ตามแต่ละ Sector ที่กำหนดไว้



รูปที่ 4.11 ฟังก์ชั่นบล็อคในการออกแบบระบบ 3เฟส

บทที่5

ผลการทดลอง

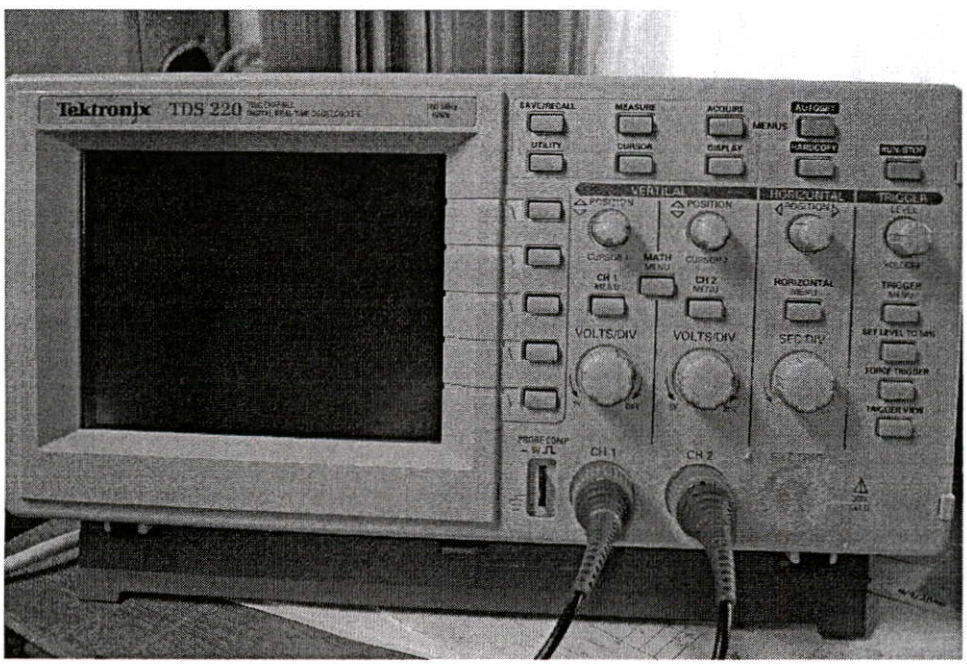
ผลการทดลองของวงจรอินเวอร์เตอร์ควบคุมมอเตอร์เหนี่ยวนำกระแสสลับจะแบ่งออกเป็น 2ส่วนหลักๆตามชนิดของมอเตอร์เหนี่ยวนำซึ่งจะประกอบไปด้วย

5.1 ผลการทดลองมอเตอร์เหนี่ยวนำกระแสสลับแบบ 3เฟส ซึ่งจะประกอบไปด้วย 2ส่วนคือ

ก) ส่วนของสัญญาณทางเวลา และสัญญาณทางความถี่ ภายในอินเวอร์เตอร์ ที่ทำการวัดด้วย Digital Storage Scope ยี่ห้อ Tektronix รุ่น TDS220

ข) ส่วนของผลการทดสอบอินเวอร์เตอร์กับภาระโหลดที่เป็นมอเตอร์เหนี่ยวนำ 3 เฟส ขนาด 1 แรงม้า ที่มีการจ่ายภาระทางกล ด้วยชุดทดสอบยี่ห้อ SIEMENS รวมทั้งประสิทธิภาพของอินเวอร์เตอร์ และประสิทธิภาพของมอเตอร์เหนี่ยวนำเมื่อมีภาระทางกลในแต่ละย่านของความถี่หลักมูล

5.2 ผลการทดลองมอเตอร์เหนี่ยวนำกระแสสลับแบบ 1เฟส ซึ่งทำการทดลองในเฉพาะ ส่วนของสัญญาณทางเวลาและสัญญาณทางความถี่ ของชุดอินเวอร์เตอร์ โดยทำการวัดด้วย Digital Storage Scope ยี่ห้อ Tektronix รุ่น TDS220



รูปที่ 5.1 เครื่องมือวัดสัญญาณ Digital Storage Scope ยี่ห้อ Tektronix รุ่น TDS220

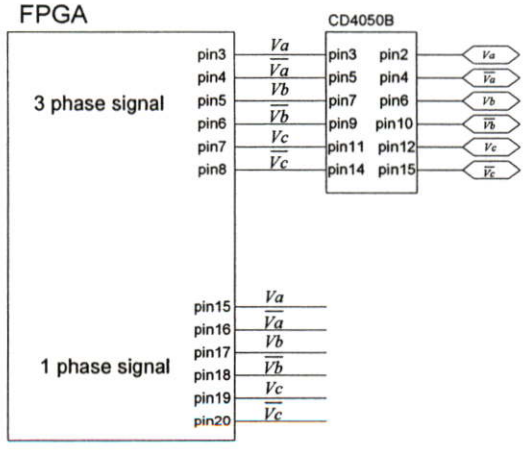
5.1 ผลการทดลองมอเตอร์เหนี่ยวนำกระแสสลับแบบ 3เฟส

5.1.1 ผลการทดลองในส่วนของสัญญาณทางเวลา และสัญญาณความถี่ของอินเวอร์เตอร์ สำหรับมอเตอร์เหนี่ยวนำ 3 เฟส

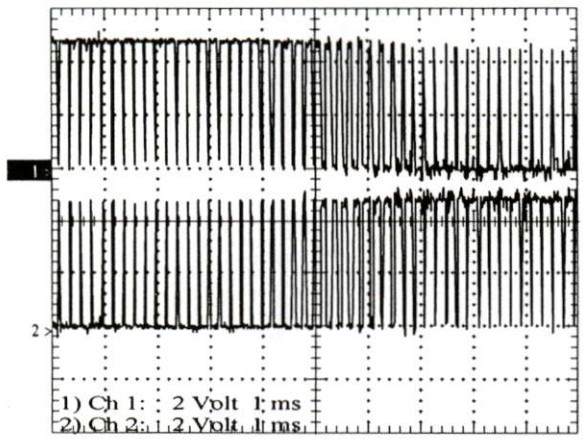
ผลการทดลองจะประกอบไปด้วยสัญญาณ SVM PWM ที่ได้จาก FPGA โดยแปลงระดับแรงดันของสัญญาณจาก 3.3V เป็นระดับ 5V โดยใช้ IC CD4050B โดยเป็นสัญญาณ SVM PWM แรงดันเฟส Va, Vb, Vc และสัญญาณตรงกันข้ามเฟส $\overline{Va}, \overline{Vb}, \overline{Vc}$ โดยสัญญาณ Va, Vb, Vc ควบคุมสวิตช์ IGBT Q_1, Q_3, Q_5 ตามลำดับ และสัญญาณ $\overline{Va}, \overline{Vb}, \overline{Vc}$ ควบคุมสวิตช์ IGBT Q_4, Q_6, Q_2 ตามลำดับ และสัญญาณ SVM PWM ที่ผ่านวงจร opto isolate ที่ป้อนให้กับขาเกตของ IGBT แต่ละตัว พร้อมทั้งวัดสัญญาณ SVM PWM ที่ขา C เทียบ E ของ IGBT แต่ละตัว โดยแหล่งจ่ายแรงดันไฟตรงได้มาจากวงจรทวีแรงดัน 2เท่า ทำให้วัดแรงดันที่ขา C เทียบ E ของ IGBT แต่ละตัวได้แรงดันสูงกว่าขนาดของแรงดันไฟสลับอินพุทเท่าตัว (ขนาดของแรงดันไฟสลับ $220V_{rms}$ และแรงดันไฟตรงหลังจากผ่านวงจรทวีแรงดันได้เป็นแรงดันไฟตรงขนาด $620V_{DC}$)

5.1.1.1 สัญญาณ PWM จาก FPGA หลังผ่านวงจรแปลงระดับแรงดัน

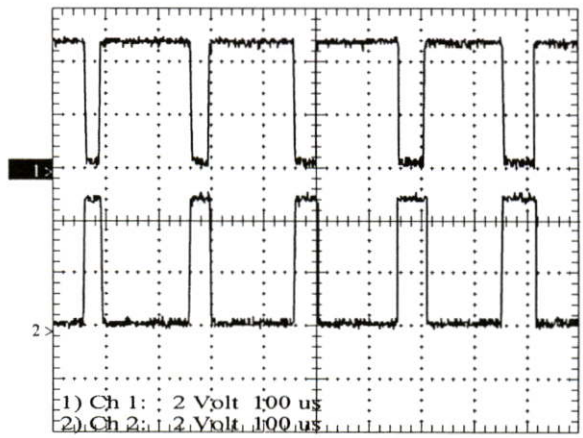
โดยสัญญาณที่วัดจากชุดแปลงระดับแรงดันจาก FPGA แรงดัน 3.3V โดยผ่าน IC CD4050B ทำการแปลงระดับแรงดันเอ้าท์พุทเป็น 5V โดยสัญญาณ SVM PWM (Active Low) มีเงื่อนไข ความถี่หลักมูล $50Hz$ ความถี่การสวิตช์ $5KHz$ และ Dead time $2.5\mu S$



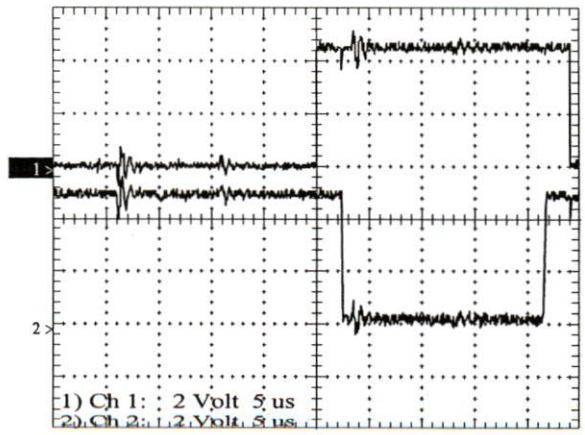
รูปที่ 5.2 วงจรแปลงระดับแรงดัน



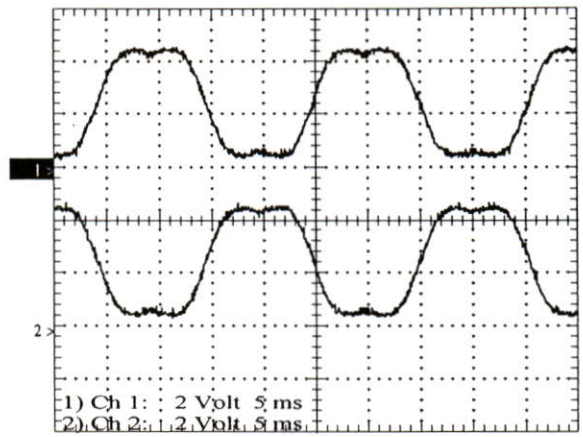
รูปที่ 5.3 รูปสัญญาณ PWM เอ้าท์พุตของ CD4050B ที่ Va (CH1) และ $\overline{Va}$ (CH2)



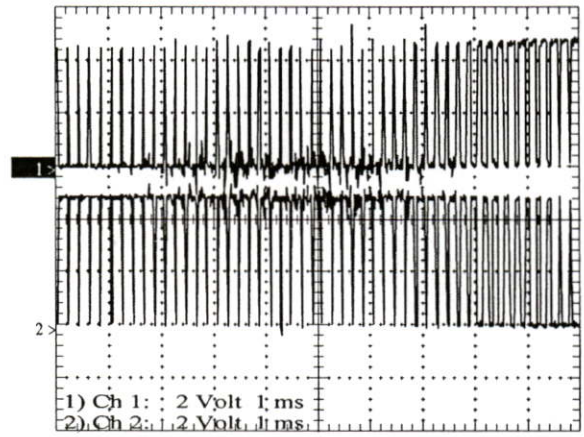
รูปที่ 5.4 รูปขยายสัญญาณ PWM ที่ เอ้าท์พุตของ CD4050B ที่ Va (CH1) และ $\overline{Va}$ (CH2) แสดงความถี่การสวิตช์ $5KHz$



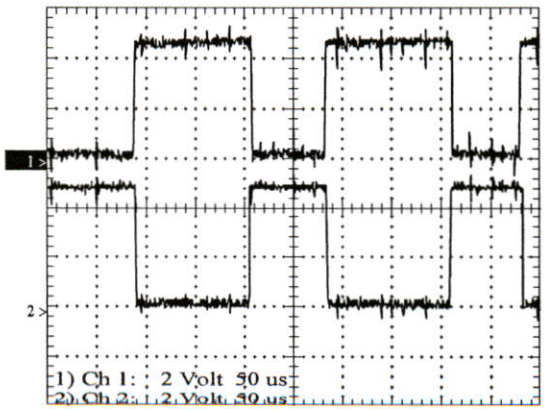
รูปที่ 5.5 รูปขยายสัญญาณ PWM ที่ เอ้าท์พุตของ CD4050B ที่ Va (CH1) และ $\overline{Va}$ (CH2) แสดง dead time $2.5\mu S$ active low



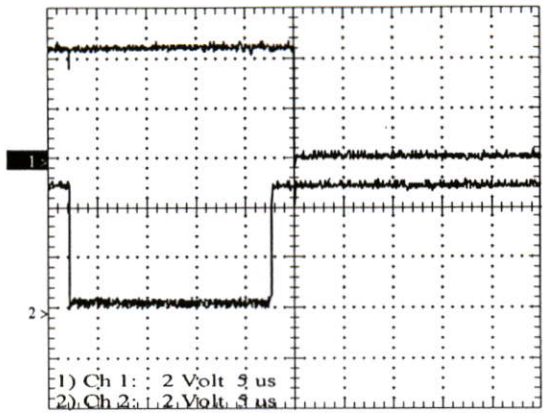
รูปที่ 5.6 รูปสัญญาณ PWM ที่ เอ้าท์พุตของ CD4050B ที่ Va (CH1) และ $\overline{Va}$ (CH2) ผ่านวงจรกรองความถี่ต่ำผ่าน



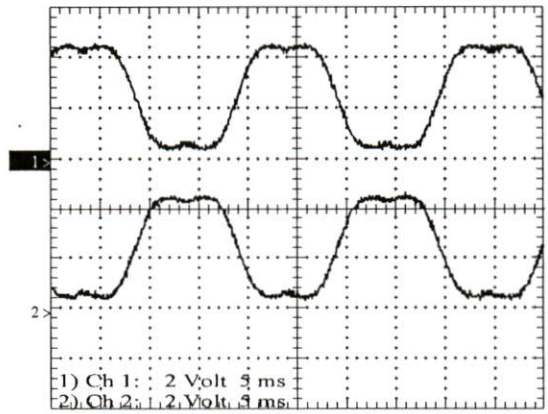
รูปที่ 5.7 รูปสัญญาณ PWM ที่ เอ้าท์พุตของ CD4050B ที่ Vb (CH1)และ $\overline{Vb}$ (CH2)



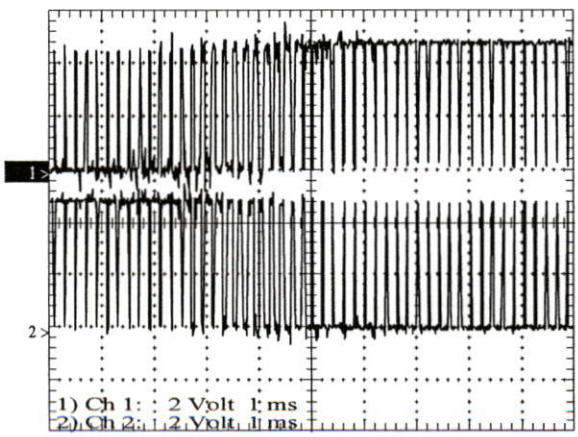
รูปที่ 5.8 รูปขยายสัญญาณ PWM ที่ เอ้าท์พุตของ CD4050B ที่ Vb (CH1) และ $\overline{Vb}$ (CH2) แสดงความถี่การสวิตช์ $5KHz$



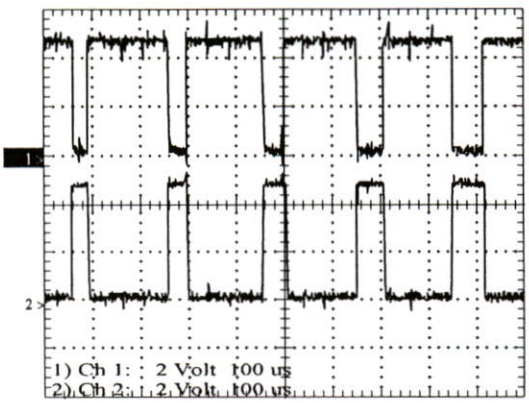
รูปที่ 5.9 รูปขยายสัญญาณ PWM ที่ เอ้าท์พุตของ CD4050B ที่ Vb (CH1) และ $\overline{Vb}$ (CH2) แสดง dead time $2.5\mu S$ active low



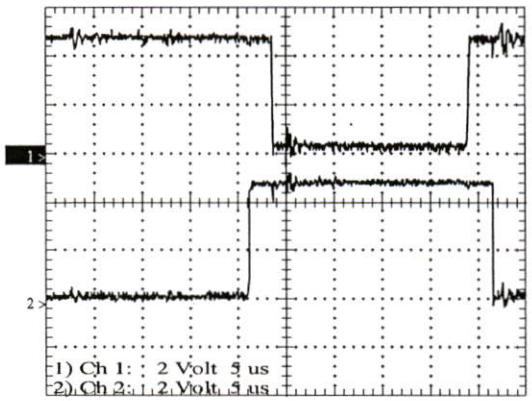
รูปที่ 5.10 รูปสัญญาณ PWM ที่ เอ้าท์พุตของ CD4050B ที่ Vb (CH1) และ $\overline{Vb}$ (CH2) ผ่านวงจรกรองความถี่ต่ำผ่าน



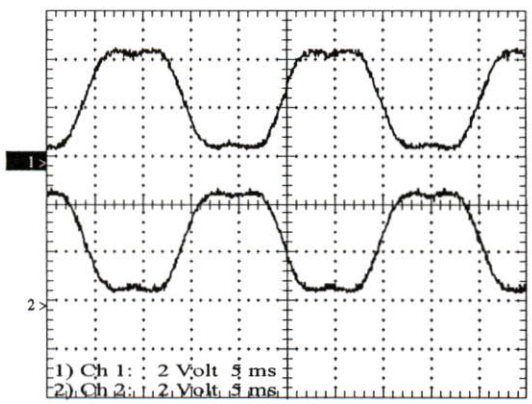
รูปที่ 5.11 รูปสัญญาณ PWM ที่ เอ้าท์พุตของ CD4050B ที่ Vc (CH1) และ $\overline{Vc}$ (CH2)



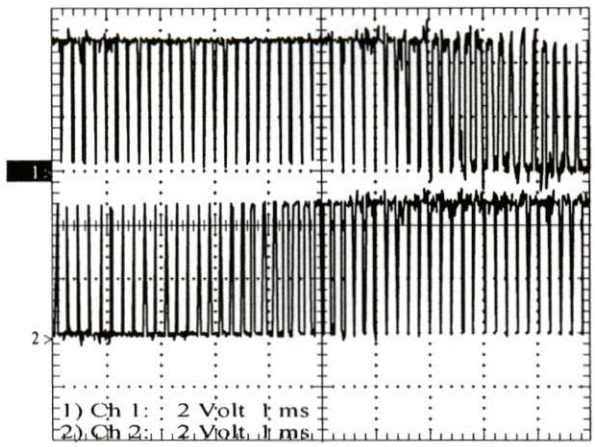
รูปที่ 5.12 รูปขยายสัญญาณ PWM ที่ เอ้าท์พุตของ CD4050B ที่ Vc (CH1) และ $\overline{Vc}$ (CH2) แสดงความถี่การสวิตช์ $5KHz$



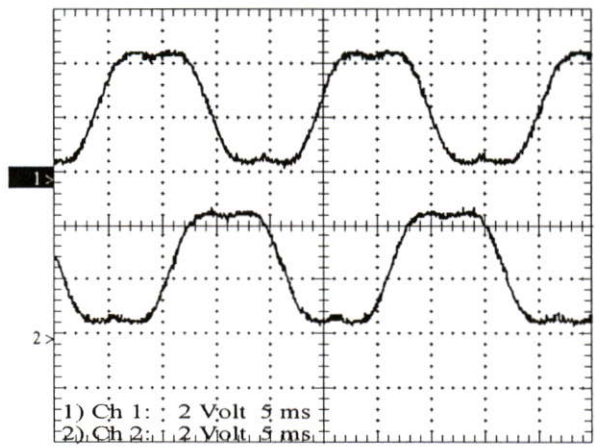
รูปที่ 5.13 รูปขยายสัญญาณ PWM ที่ เอ้าท์พุตของ CD4050B ที่ Vc (CH1) และ $\overline{Vc}$ (CH2) แสดง dead time $2.5\mu S$ active low



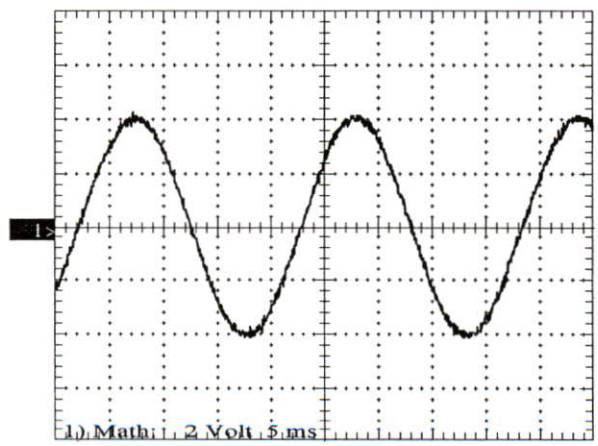
รูปที่ 5.14 รูปสัญญาณ PWM ที่ เอ้าท์พุตของ CD4050B ที่ Vc (CH1) และ $\overline{Vc}$ (CH2) ผ่านวงจรกรองความถี่ต่ำผ่าน



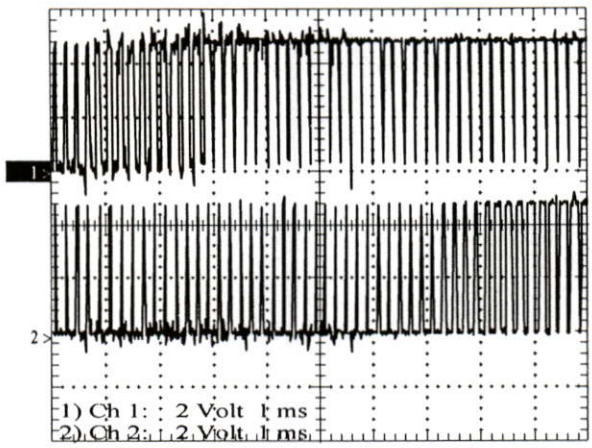
รูปที่ 5.15 รูปสัญญาณ PWM ที่ เอ้าท์พุตของ CD4050B ที่ *Va* (CH1) และ *Vb* (CH2) เทียบกัน



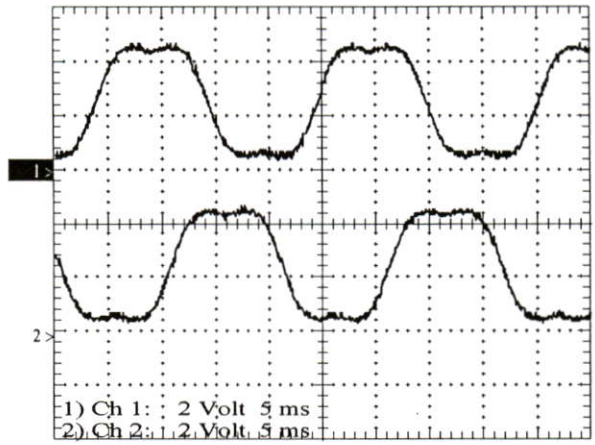
รูปที่ 5.16 รูปสัญญาณ PWM ที่ เอ้าท์พุตของ CD4050B ที่ *Va* (CH1) และ *Vb* (CH2) เทียบกันโดยผ่านวงจรกรองความถี่ต่ำผ่าน



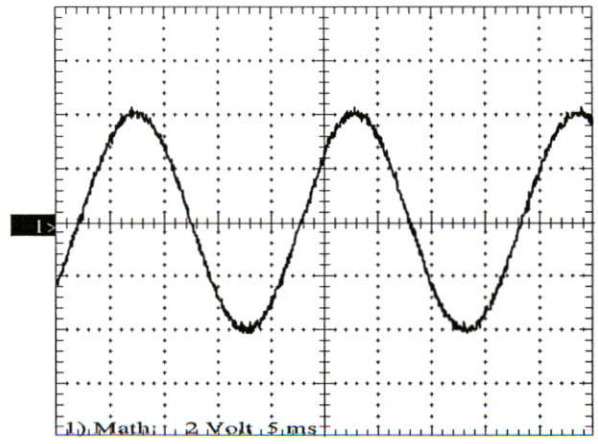
รูปที่ 5.17 รูปสัญญาณ ที่ เอ้าท์พุตของ CD4050B ที่ *Va* (CH1) ลบ *Vb* (CH2)



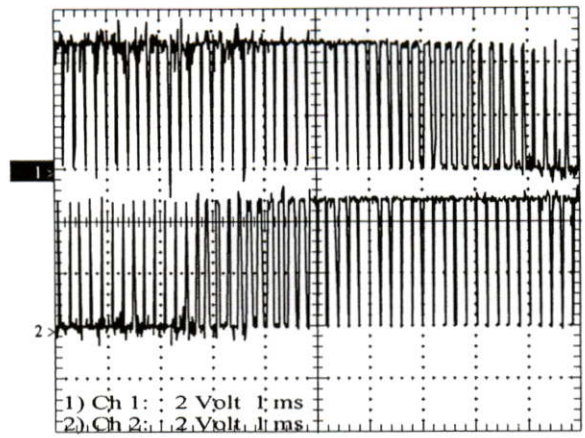
รูปที่ 5.18 รูปสัญญาณ PWM ที่ เอ้าท์พุตของ CD4050B ที่ *Vb* (CH1) และ *Vc* (CH2) เทียบกัน



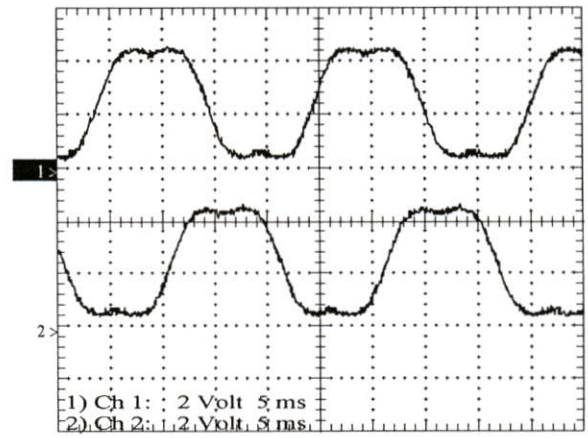
รูปที่ 5.19 รูปสัญญาณ PWM ที่ เอ้าท์พุตของ CD4050B ที่ *Vb* (CH1) และ *Vc* (CH2) ผ่านวงจรกรองความถี่ต่ำผ่าน



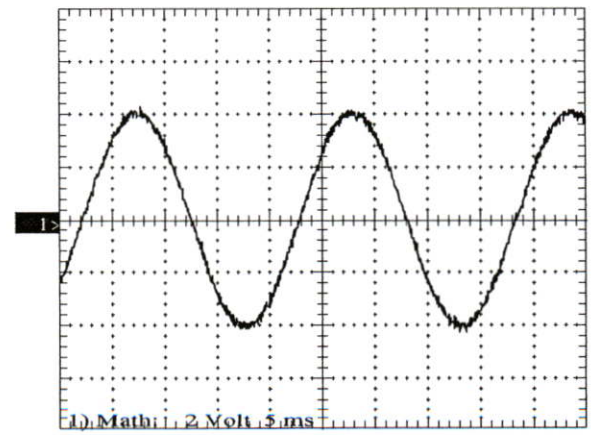
รูปที่ 5.20 รูปสัญญาณ PWM ที่ เอ้าท์พุตของ CD4050B ที่ *Vb* (CH1) ลบ *Vc* (CH2)



รูปที่ 5.21 รูปสัญญาณ PWM ที่ เอ้าท์พุตของ CD4050B ที่ Vc (CH1) และ Va (CH2) เทียบกัน



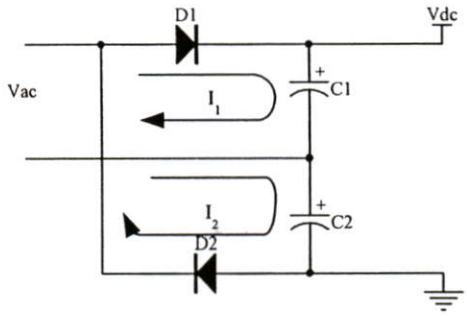
รูปที่ 5.22 รูปสัญญาณ PWM ที่ เอ้าท์พุตของ CD4050B ที่ Vc (CH1) และ Va (CH2) ผ่านวงจรกรองความถี่ต่ำผ่าน



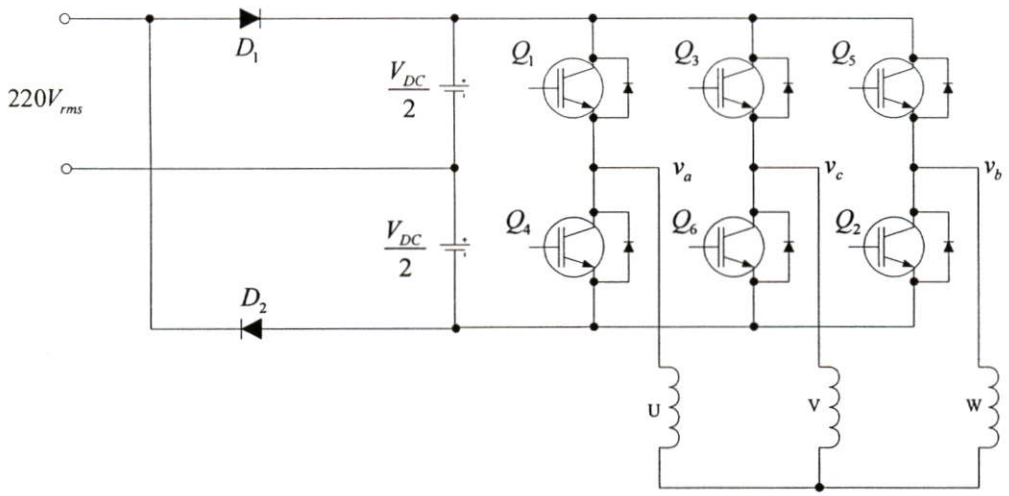
รูปที่ 5.23 รูปสัญญาณ PWM ที่ เอ้าท์พุตของ CD4050B ที่ Vc (CH1) ลบ Va (CH2)

5.1.1.2 สัญญาณ PWM วัดที่ขาเกตของ IGBT ที่จุดต่างๆ

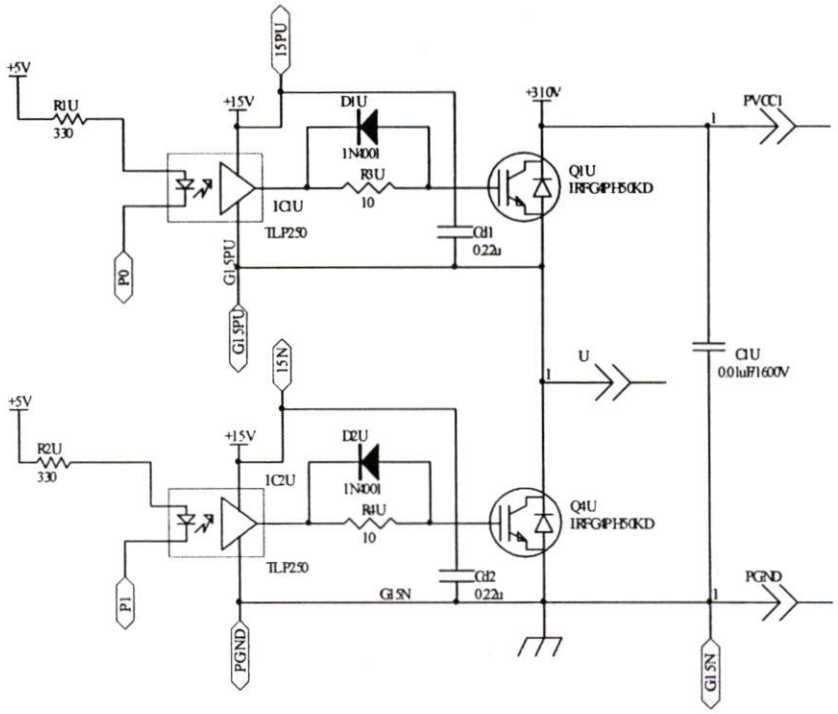
โดยในส่วนนี้จะเป็นการวัดสัญญาณ SVM PWM ที่ขาเกตของ IGBT ที่แสดงรูปวงจรขับสัญญาณดังรูปที่ 5.25 และ5.26



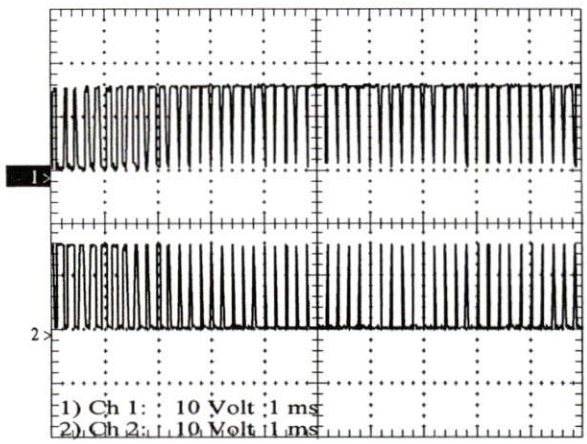
รูปที่ 5.24 วงจรเร็กติไฟเออร์และทวีแรงดัน



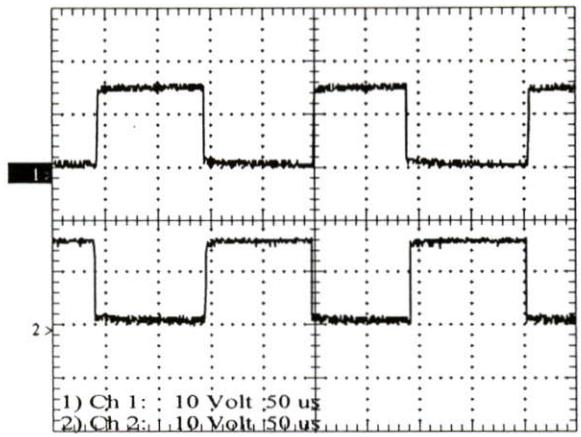
รูปที่ 5.25 วงจรทวีเรียงดันที่ใช้เป็นแหล่งจ่ายไฟตรงที่จ่ายให้กับชุดสวิตช์ IGBT



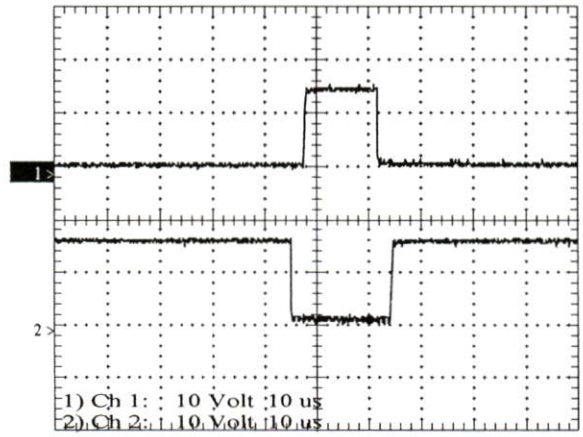
รูปที่ 5.26 วงจรภาคขับสวิตช์ IGBT ประกอบไปด้วยชุด opto isolate และชุดสวิตช์ IGBT



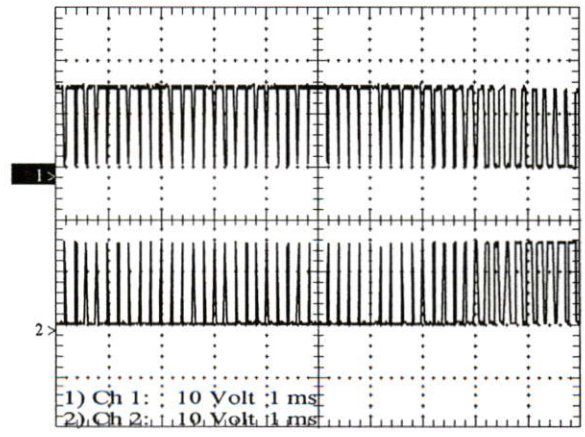
รูปที่ 5.27 รูปสัญญาณ PWM ที่ขาเกตของสวิตช์ Q1U(CH1) และ Q4U(CH2)



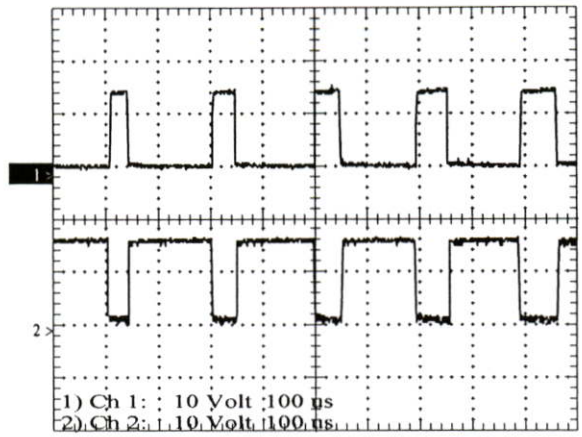
รูปที่ 5.28 รูปขยายสัญญาณ PWM ที่ขาเกตของสวิตช์ Q1U(CH1) และ Q4U(CH2)



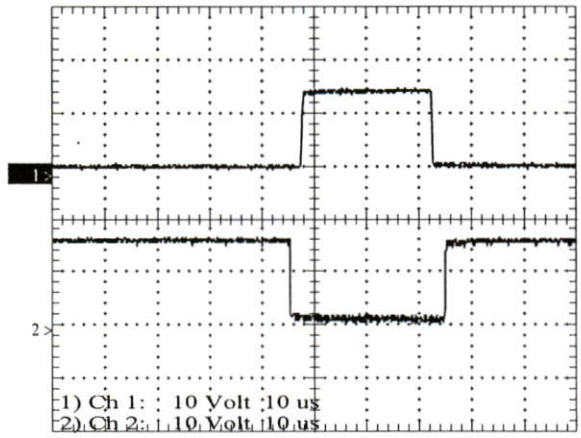
รูปที่ 5.29 รูปขยายสัญญาณ PWM ที่ขาเกตของสวิตช์ Q1U(CH1) และ Q4U(CH2) แสดง Dead time



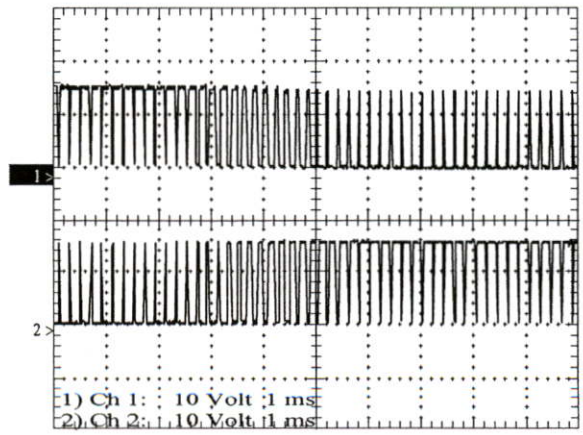
รูปที่ 5.30 รูปสัญญาณ PWM ที่ขาเกตของสวิตช์ Q3V(CH1) และ Q6V(CH2)



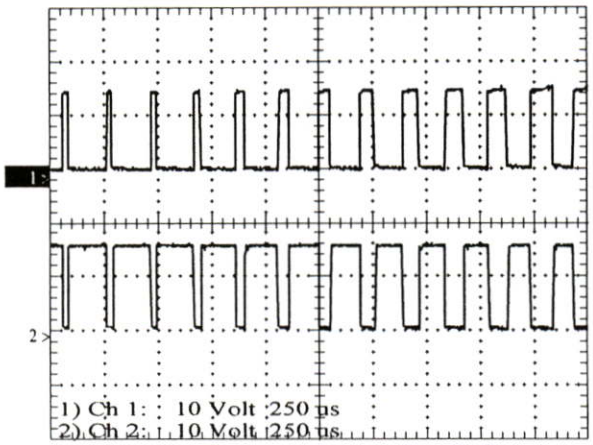
รูปที่ 5.31 รูปขยายสัญญาณ PWM ที่ขาเกตของสวิตช์ Q3V(CH1) และ Q6V(CH2)



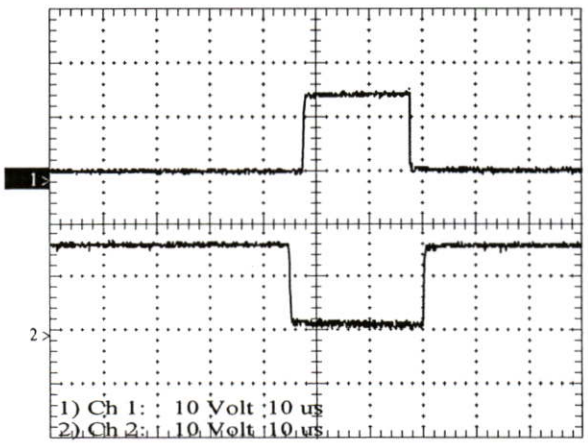
รูปที่ 5.32 รูปขยายสัญญาณ PWM ที่ขาเกตของสวิตช์ Q3V(CH1) และ Q6V(CH2) แสดง Deadtime



รูปที่ 5.33 รูปสัญญาณ PWM ที่ขาเกตของสวิตช์ Q5W(CH1) และ Q2W(CH2)



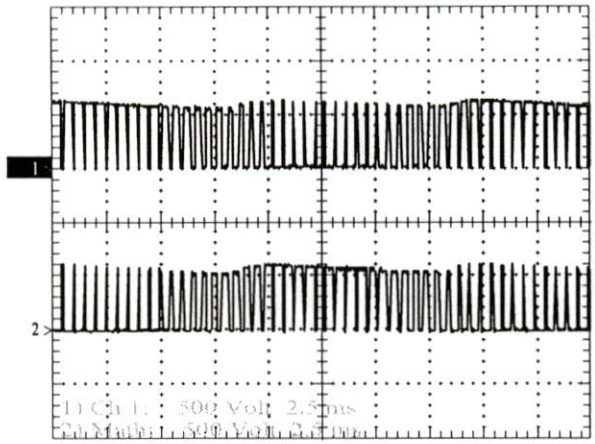
รูปที่ 5.34 รูปขยายสัญญาณ PWM ที่ขาเกตของสวิตช์ Q5W(CH1) และ Q2W(CH2)



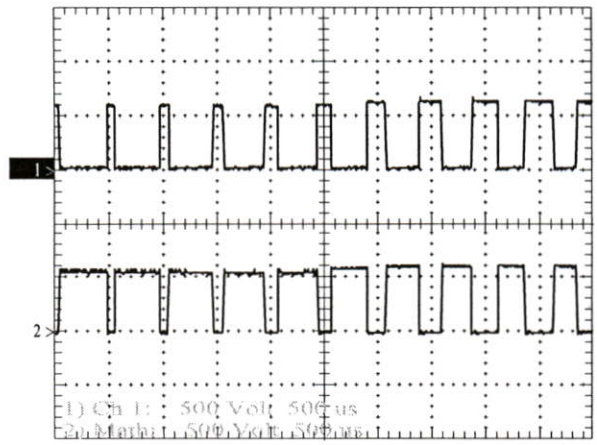
รูปที่5.35 รูปขยายสัญญาณ PWM ที่ขาเกตของสวิตช์ Q5W(CH1) และ Q2W(CH2) แสดง Deadtime

5.1.1.3 แสดงรูปสัญญาณที่ขา C, E ของสวิตช์

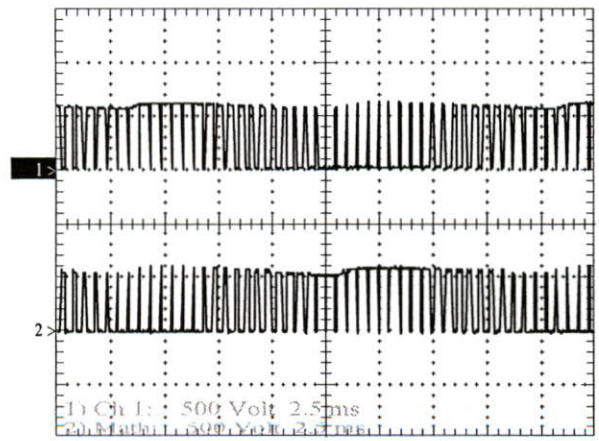
สัญญาณที่วัดเป็นสัญญาณ SVM PWM วัดที่ขา C และE ของ IGBT ซึ่งเป็นสัญญาณ SVM PWM ที่ควบคุมการสวิตช์โดยขาเกตของ IGBT โดยสัญญาณที่ได้จะใช้เป็นสัญญาณในการขับมอเตอร์เหนี่ยวนำโดยขนาดแรงดันมีค่า 600 โวล์ต



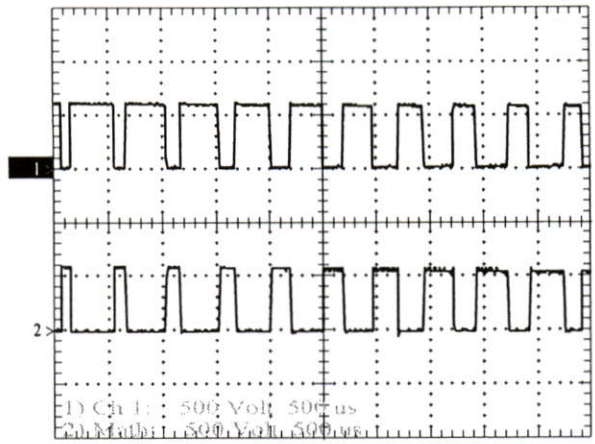
รูปที่ 5.36 รูปสัญญาณที่ขา C, E ของสวิตช์ Q1U(CH1) และ Q4U(CH2)



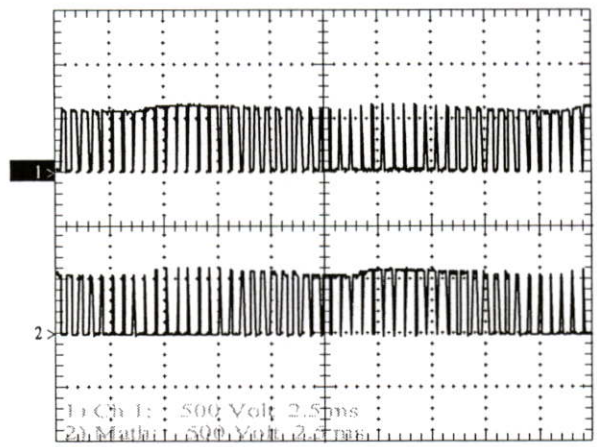
รูปที่ 5.37 รูปขยายสัญญาณที่ขา C, E ของสวิตช์ Q1U(CH1) และ Q4U(CH2)



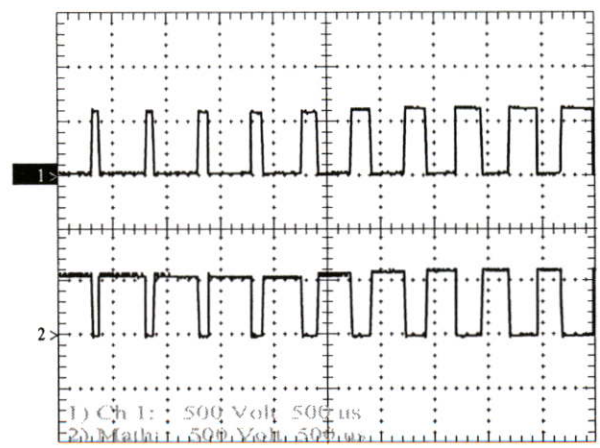
รูปที่ 5.38 รูปสัญญาณที่ขา C, E ของสวิตช์ Q3V(CH1) และ Q6V(CH2)



รูปที่ 5.39 รูปขยายสัญญาณที่ขา C, E ของสวิตช์ Q3V(CH1) และ Q6V(CH2)



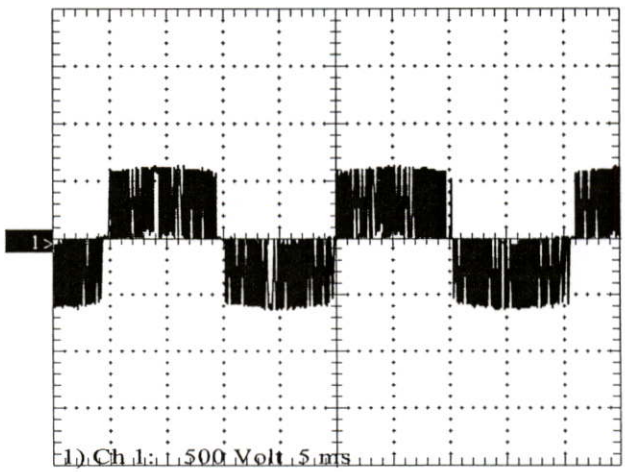
รูปที่ 5.40 รูปสัญญาณที่ขา C, E ของสวิตช์ Q5W(CH1) และ Q2W(CH2)



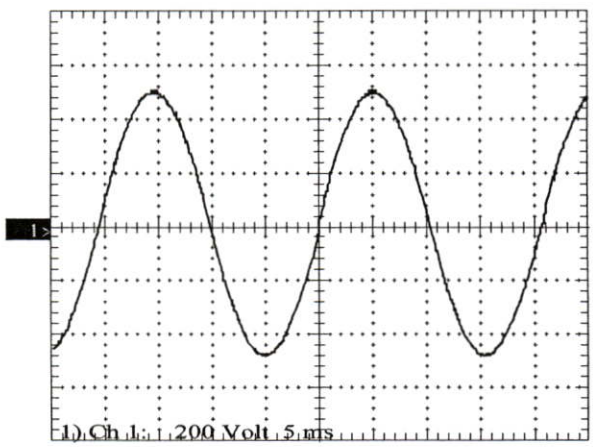
รูปที่ 5.41 รูปขยายสัญญาณที่ขา C, E ของสวิตช์ Q5W(CH1) และ Q2W(CH2)

5.1.1.4 สัญญาณ Out put PWM ที่จ่ายให้กับมอเตอร์เหนี่ยวนำ 3เฟส

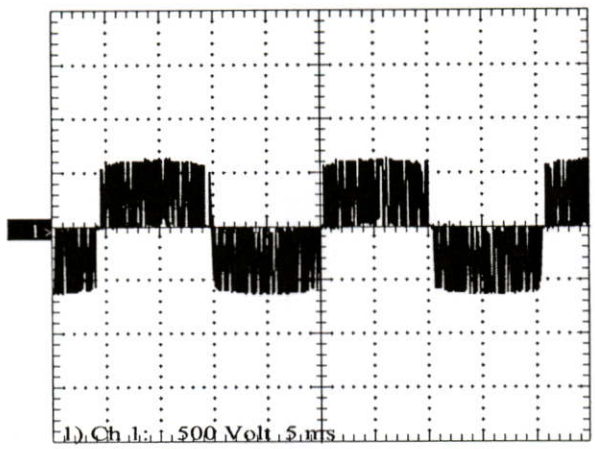
สัญญาณดังแสดงในรูปที่ 5.42 ถึงรูปที่ 5.47 เป็นสัญญาณแรงดันเอ้าท์พุตที่จุด $v_a(U), v_b(W), v_c(V)$ ดังรูปที่ 5.25 โดยในรูปที่ 5.42 เป็นแรงดันที่วัดระหว่างสาย $v_a(U)$และ $v_c(V)$ ในรูปที่ 5.43 เป็นสัญญาณของแรงดันในรูปที่ 5.42 หลังจากวัดผ่านวงจรกรองความถี่ต่ำผ่าน โดยในรูปที่ 5.44 จนถึง 5.45 วัดแรงดันสาย $v_c(V)$ กับ $v_b(W)$ และรูปที่ 5.46 จนถึงรูปที่ 5.47 วัดแรงดันสาย $v_b(W)$ กับ $v_a(U)$



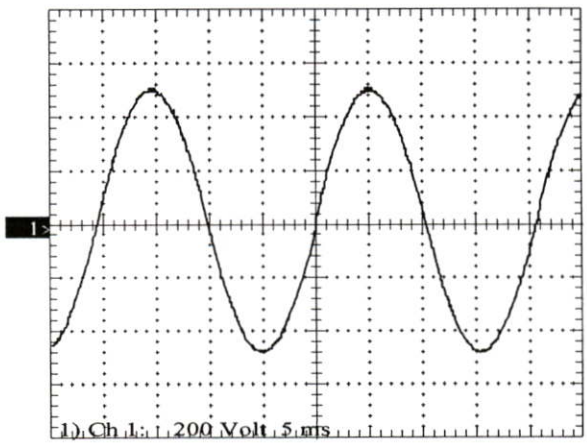
รูปที่ 5.42 สัญญาณ Out put PWM แรงดันสายที่ U และV



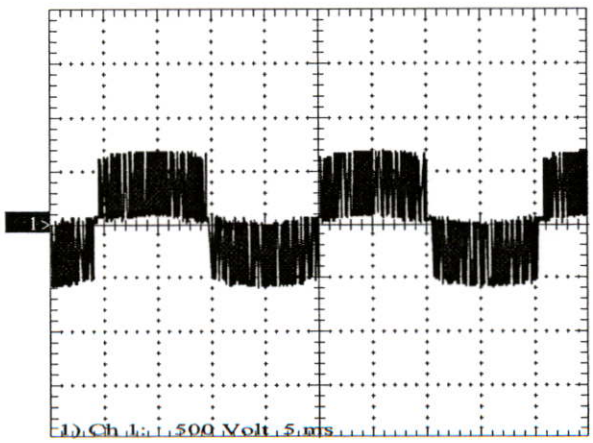
รูปที่ 5.43 สัญญาณ Out put PWM แรงดันสายที่ U และV ผ่านวงจรกรองความถี่ต่ำผ่าน



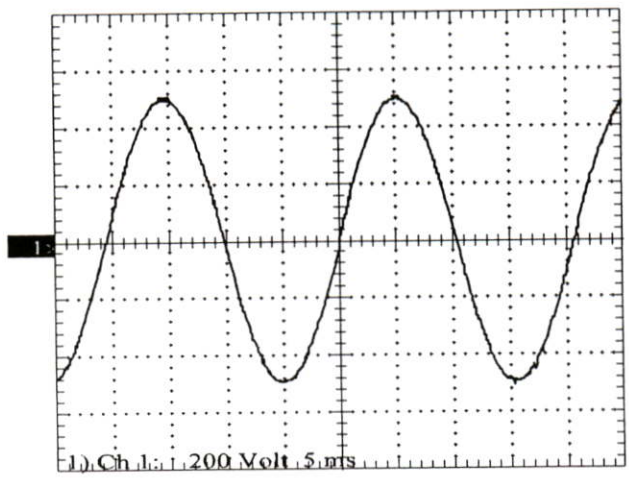
รูปที่ 5.44 สัญญาณ Out put PWM แรงดันสายที่ V และW



รูปที่ 5.45 สัญญาณ Out put PWM แรงดันสายที่ V และW ผ่านวงจรกรองความถี่ต่ำผ่าน



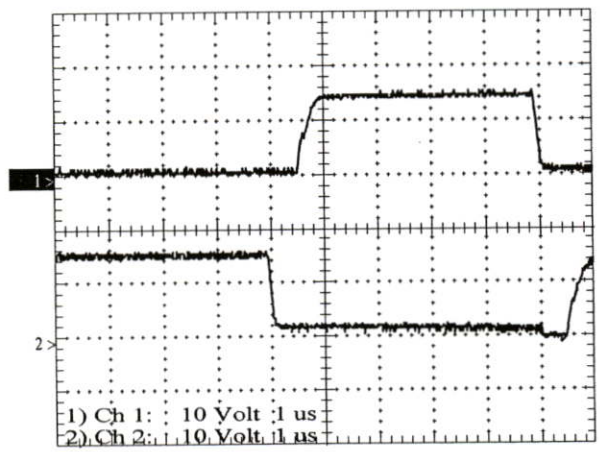
รูปที่ 5.46 สัญญาณ Out put PWM แรงดันสายที่ W และU



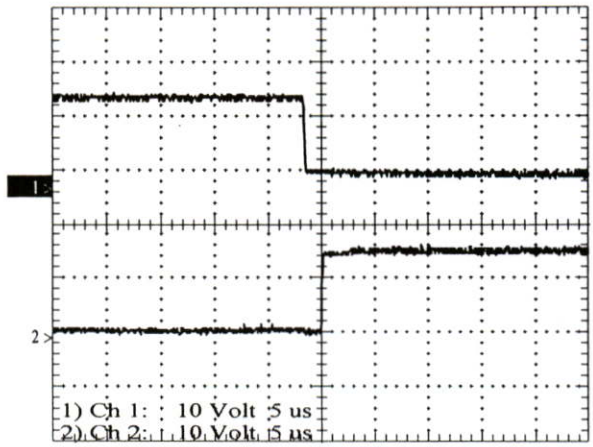
รูปที่ 5.47 สัญญาณ Out put PWM แรงดันสายที่ W และU ผ่านวงจรกรองความถี่ต่ำผ่าน

5.1.1.5 การปรับค่าเวลา dead time ค่าต่างๆ

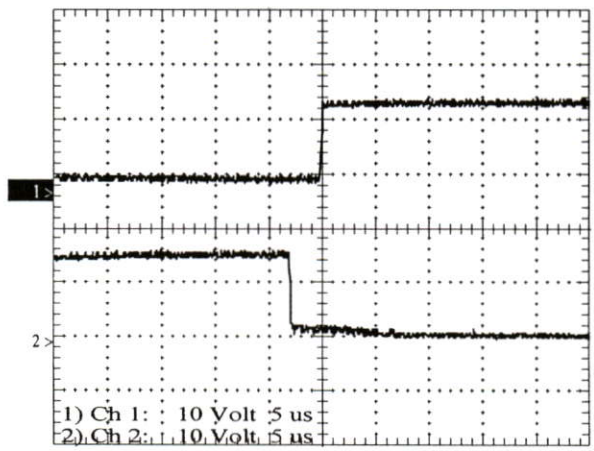
การกำหนดเวลา dead time เพื่อเป็นการหน่วงเวลาการสวิตช์เพื่อไม่ให้เกิดการสวิตช์พร้อมกันในแต่ละกิ่งคือ กิ่ง $v_a, \overline{v}_a$ กิ่ง $v_b, \overline{v}_b$ และ กิ่ง $v_c, \overline{v}_c$ ซึ่งหากสวิตช์พร้อมกันจะเป็นการลัดวงจรของแหล่งจ่ายแรงดันไฟตรง 600 โวลต์ กับกราวน์ ซึ่งจะทำให้สวิตช์เสียหายได้ การกำหนดช่วงเวลา dead time จำเป็นต้องทราบว่าช่วงเวลา turn on และ turn off ของ IGBT เพื่อกำหนดช่วงเวลา dead time ในเหมาะสม แต่หากช่วงเวลา dead time กว้างมากไปจะทำให้ค่าเฉลี่ยแรงดันลดลงด้วย



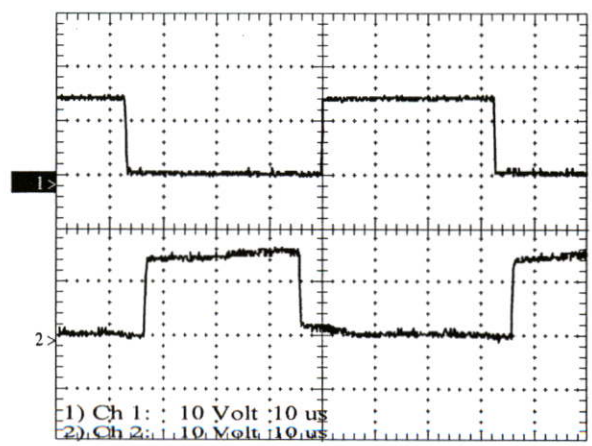
รูปที่ 5.48 รูปขยายสัญญาณแสดง Dead time $1\mu S$ ที่ขาเกตของ Q1U(CH1) และ Q4U(CH2)



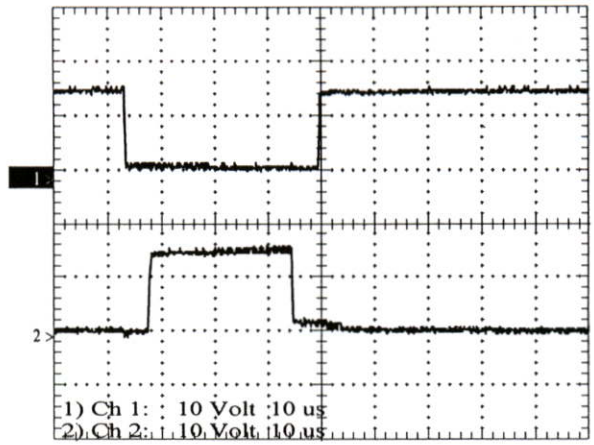
รูปที่ 5.49 รูปขยายสัญญาณแสดง Dead time $2\mu S$ ที่ขาเกตของ Q1U(CH1) และ Q4U(CH2)



รูปที่ 5.50 รูปขยายสัญญาณแสดง Dead time $3\mu S$ ที่ขาเกตของ Q1U(CH1) และ Q4U(CH2)



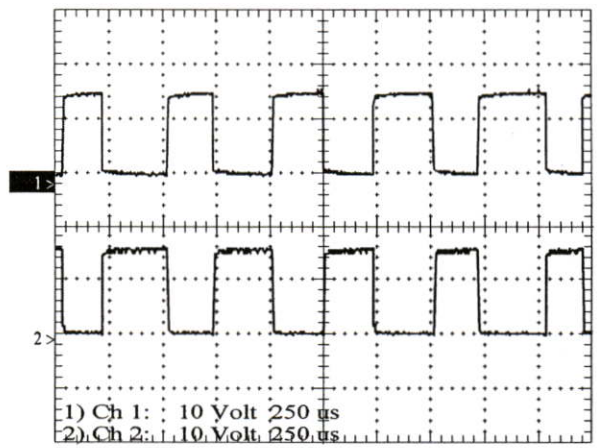
รูปที่ 5.51 รูปขยายสัญญาณแสดง Dead time $4\mu S$ ที่ขาเกตของ Q1U(CH1) และ Q4U(CH2)



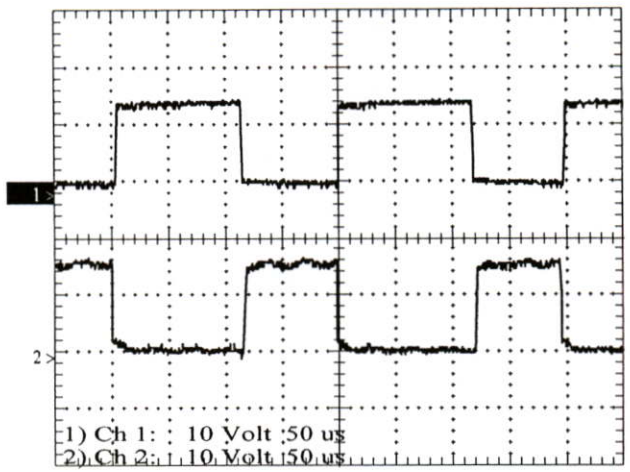
รูปที่ 5.52 รูปขยายสัญญาณแสดง Dead time $5\mu S$ ที่ขาเกตของ Q1U(CH1) และ Q4U(CH2)

5.1.1.6 การปรับความถี่การสวิตช์ IGBT

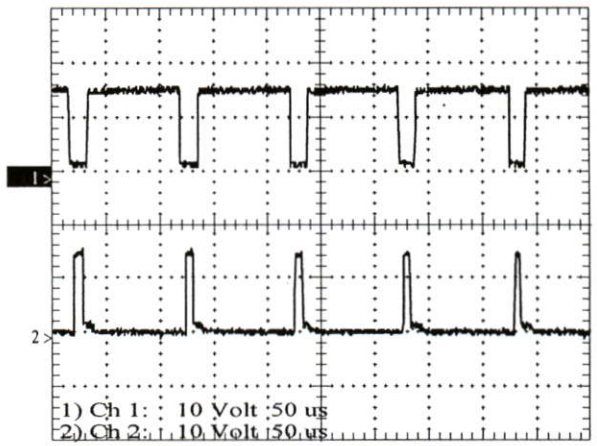
เป็นการปรับความถี่ของความถี่การสวิตช์ของ IGBT ซึ่งไม่มีผลกับความถี่หลักมูลของสัญญาณแรงดันไฟฟ้าสลับ โดยทั่วไปมีผลอยู่หลายประการเช่น หากใช้ในวงจรแหล่งจ่ายแบบสวิตช์ชิ่งจะทำให้ขนาดของหม้อแปลงมีขนาดลดลง และออกแบบวงจรกรองได้ง่ายขึ้น ข้อเสียคือเกิดผลของการสูญเสียกำลังที่เกิดจากการสวิตช์ชิ่ง แต่ในการใช้ขับมอเตอร์เหนี่ยวนำ ผลของความถี่การสวิตช์จะลดผลของเสียงที่เกิดจากมอเตอร์ลงได้ ในการทดลองได้ทำการทดลองปรับความถี่สวิตช์หลายค่า



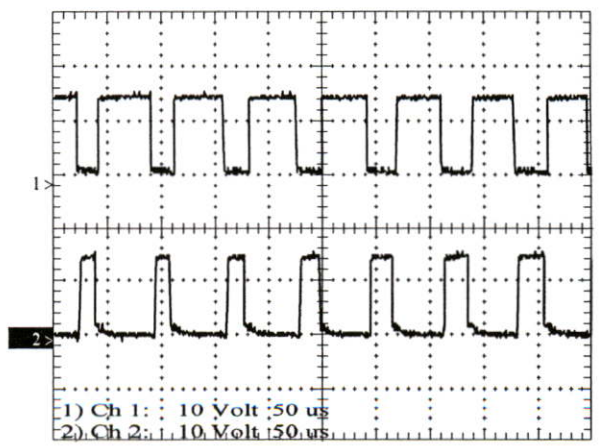
รูปที่ 5.53 รูปแสดงสัญญาณ ความถี่การสวิตช์ $2KHz$ ที่ขาเกตของ Q1U(CH1) และ Q4U(CH2)



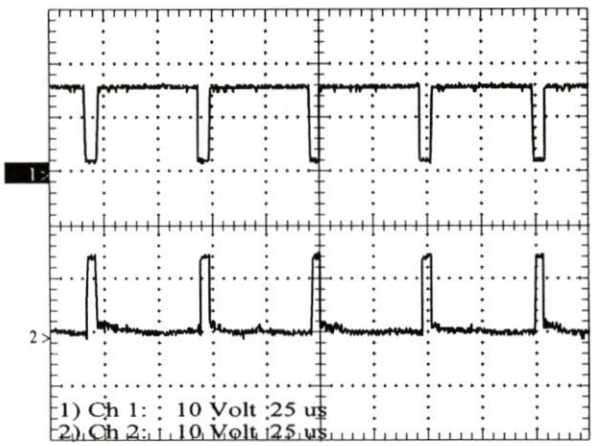
รูปที่ 5.54 รูปแสดงสัญญาณ ความถี่การสวิตช์ 5*KHz* ที่ขาเกตของ Q1U(CH1) และ Q4U(CH2)



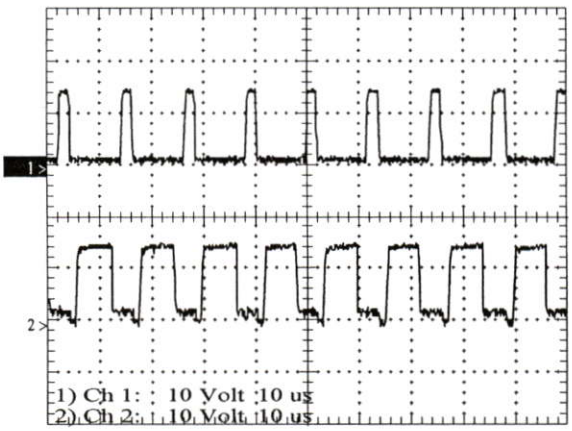
รูปที่ 5.55 รูปแสดงสัญญาณ ความถี่การสวิตช์ 10*KHz* ที่ขาเกตของ Q1U(CH1) และ Q4U(CH2)



รูปที่ 5.56 รูปแสดงสัญญาณ ความถี่การสวิตช์ 15*KHz* ที่ขาเกตของ Q1U(CH1) และ Q4U(CH2)



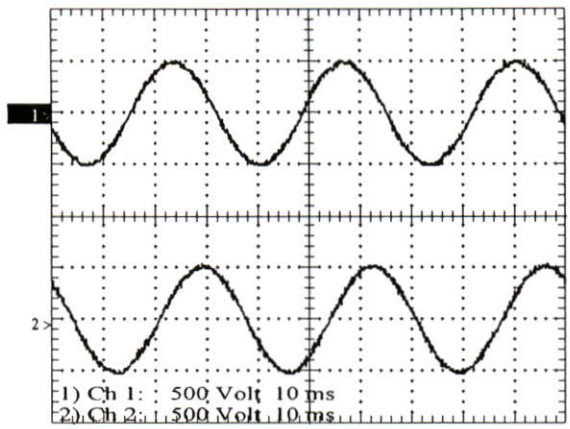
รูปที่ 5.57 รูปแสดงสัญญาณ ความถี่การสวิตช์ $20KHz$ ที่ขาเกตของ Q1U(CH1) และ Q4U(CH2)



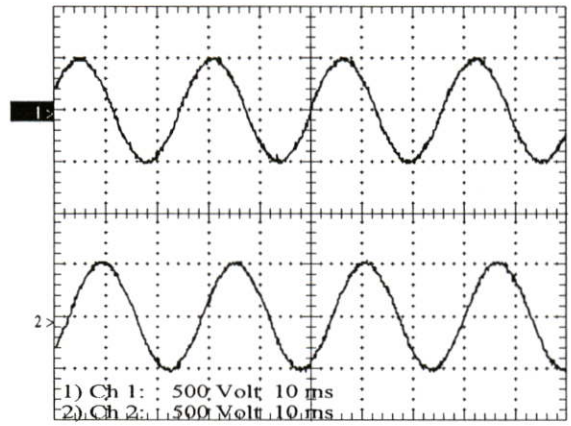
รูปที่ 5.58 รูปแสดงสัญญาณ ความถี่การสวิตช์ $80KHz$ ที่ขาเกตของ Q1U(CH1) และ Q4U(CH2)

5.1.1.7 การปรับความถี่หลักมูลแรงดันไฟสลับที่จ่ายให้กับมอเตอร์เหนี่ยวนำ 3 เฟส

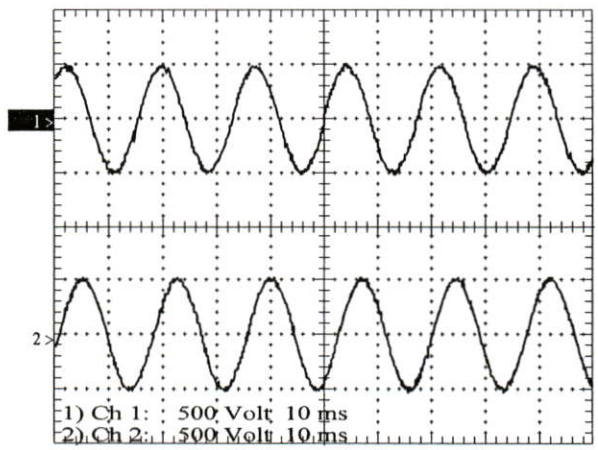
การปรับความถี่หลักมูลของไฟฟ้าสลับที่จ่ายให้กับมอเตอร์เหนี่ยวนำจะมีผลโดยตรงกับความเร็วรอบของมอเตอร์เหนี่ยวนำกระแสสลับตามสมการที่ 2.1



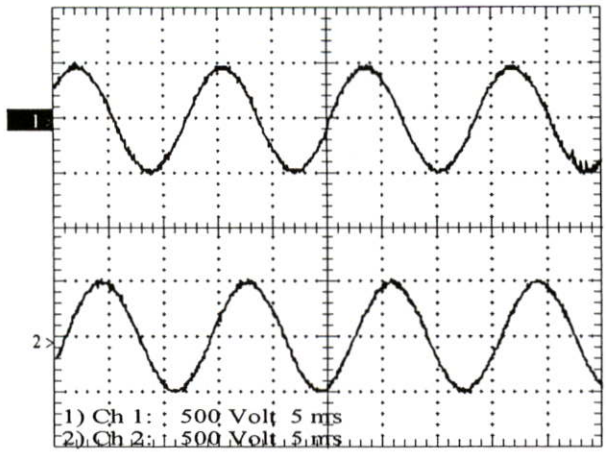
รูปที่ 5.59 รูปสัญญาณไฟสลับความถี่ $30Hz$ โดยวัดที่ v_{UV} (CH1) และ v_{WV} (CH2)



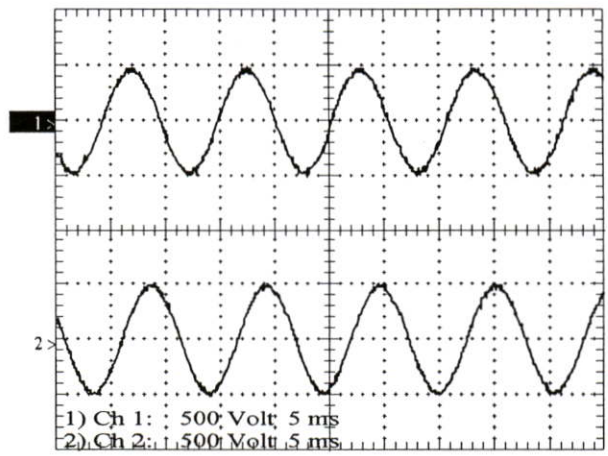
รูปที่ 5.60 รูปสัญญาณไฟสลับความถี่ $40Hz$ โดยวัดที่ v_{UV} (CH1) และ v_{WV} (CH2)



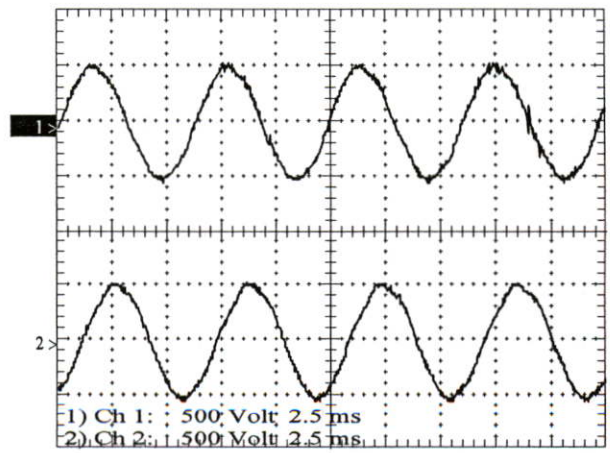
รูปที่ 5.61 รูปสัญญาณไฟสลับความถี่ $60Hz$ โดยวัดที่ v_{UV} (CH1) และ v_{WV} (CH2)



รูปที่ 5.62 รูปสัญญาณไฟสลับความถี่ $75Hz$ โดยวัดที่ v_{UV} (CH1) และ v_{WV} (CH2)



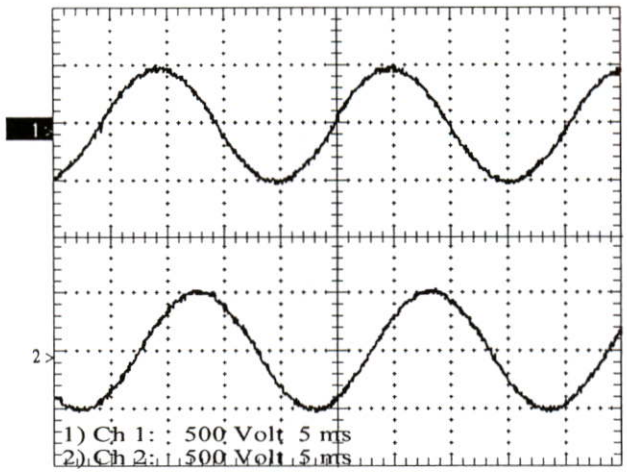
รูปที่ 5.63 รูปสัญญาณไฟสลับความถี่ $100Hz$ โดยวัดที่ v_{UV} (CH1) และ v_{WV} (CH2)



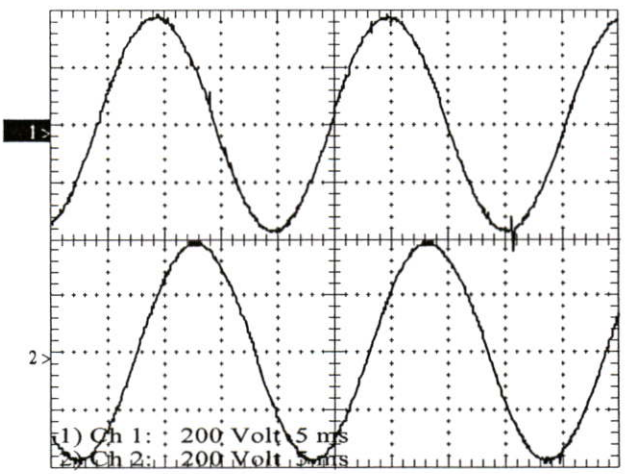
รูปที่ 5.64 รูปสัญญาณไฟสลับความถี่ $160Hz$ โดยวัดที่ v_{UV} (CH1) และ v_{WV} (CH2)

5.1.1.8 การปรับ Modulation Index

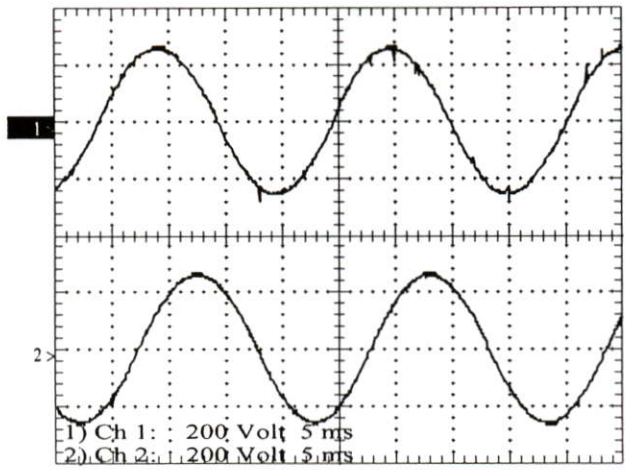
การปรับดรรชนีการมอดดูเลชั่นจะมีผลกับขนาดแรงดันของสัญญาณไฟฟ้าสลับที่ป้อนให้กับมอเตอร์เหนี่ยวนำกระแสสลับ โดยวัดสัญญาณที่เงื่อนไข ความถี่แรงดันไฟสลับ $50Hz$ ความถี่การสวิตช์ $5KHz$ และ Dead time $3\mu S$



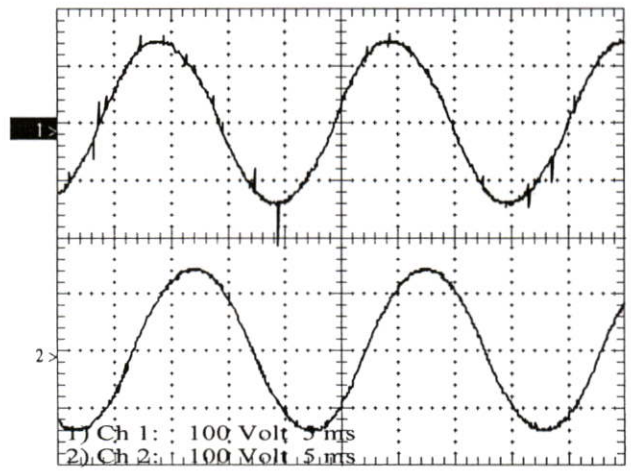
รูปที่ 5.65 รูปสัญญาณไฟสลับวัด ที่ v_{UV} (CH1) และ v_{WV} (CH2) , Modulation Index = 100%



รูปที่ 5.66 รูปสัญญาณไฟสลับวัด ที่ v_{UV} (CH1) และ v_{WV} (CH2) , Modulation Index = 75%



รูปที่ 5.67 รูปสัญญาณไฟสลับวัด ที่ v_{UV} (CH1) และ v_{WV} (CH2) , Modulation Index = 50%



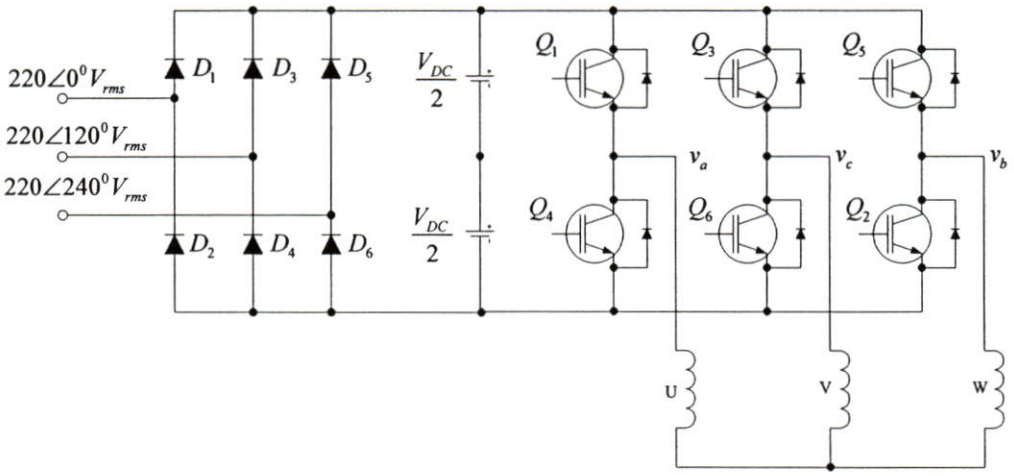
รูปที่ 5.68 รูปสัญญาณไฟสลับวัด ที่ v_{UV} (CH1) และ v_{WV} (CH2) , Modulation Index = 25%

5.1.2 ผลทดสอบอินเวอร์เตอร์กับภาระโหลดที่เป็นมอเตอร์เหนี่ยวนำ 3 เฟส ขนาด 1 แรงม้า ที่มีการจ่ายภาระทางกล ด้วยชุดทดสอบยี่ห้อ SIEMENS

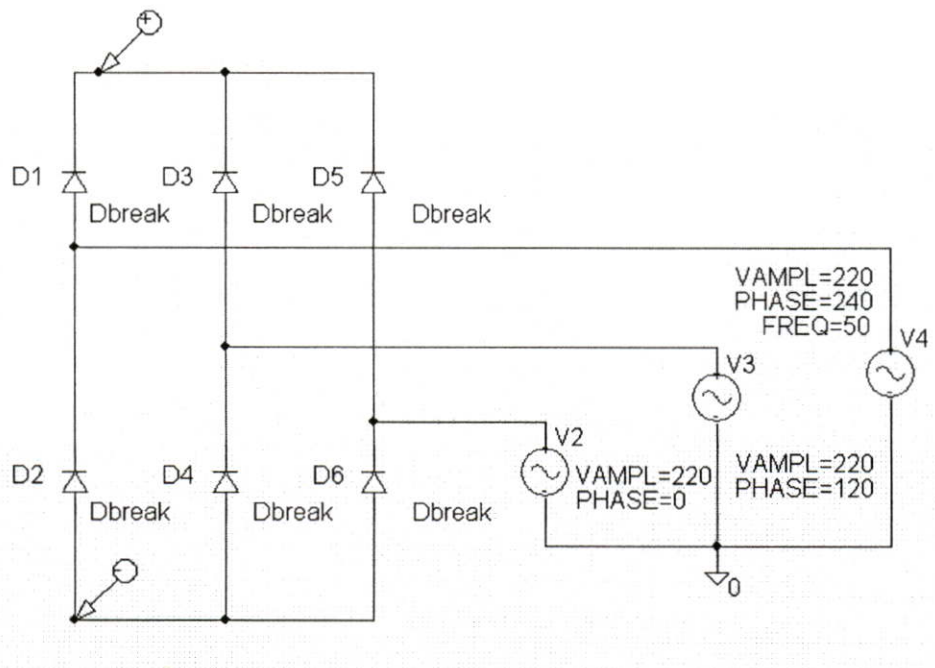
5.1.2.1 วงจรชุดขับสวิตช์ และวงจรเรียงกระแสแบบ 3เฟสที่ใช้เป็นแหล่งจ่ายไฟตรง

จากรูปที่5.69 แสดงวงจรชุดสวิตช์ IGBT โดย Q_1 Q_4 ใช้ขับขดลวด U ของมอเตอร์เหนี่ยวนำ 3เฟส และ Q_3 Q_6 ใช้ขับขดลวด V ของมอเตอร์เหนี่ยวนำ 3เฟส และ Q_5 Q_2 ใช้ขับขดลวด W ของมอเตอร์เหนี่ยวนำ 3เฟส โดยแหล่งจ่ายแรงดันไฟตรงที่จ่ายให้กับชุดสวิตช์ IGBT ใช้วงจร

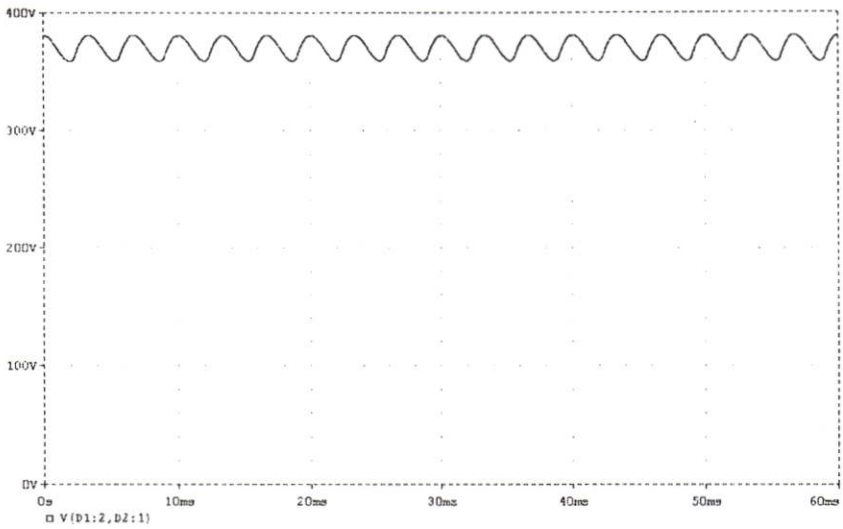
เรียงกระแสแบบ 3เฟส 6ลูกคลื่นเพื่อให้ได้แรงดันกระเพื่อม (ripple voltage) ที่ต่ำเมื่อมอเตอร์ดึงกระแสสูง และลดผลของความถี่แรงดันกระเพื่อม (ripple frequency)



รูปที่ 5.69 วงจรชุดสวิตช์ และวงจรเรียงกระแส 3เฟส



รูปที่ 5.70 วงจรเรียงกระแส 3เฟสที่ใช้ในการจำลองการทำงานโดย Pspice



รูปที่ 5.71 ผลการจำลองการทำงานวงจรเรียงกระแส 3เฟส

รูปไดอะแกรมการต่อการทดสอบอินเวอร์เตอร์ทำการทดสอบด้วยชุดทดสอบมอเตอร์ยี่ห้อ SIEMENS โดยบล็อกไดอะแกรมของระบบที่ใช้ในการทดสอบแสดงในรูปที่ 5.81

<u>เครื่องมือและอุปกรณ์ที่ใช้ในการทดสอบ</u>

- อินเวอร์เตอร์ 3 เฟสที่นำเสนอ 1 ชุด
- Induction Motor 3 Phase 1 HP 4 Pole 380 V 1 ตัว
- SIEMENS DC Generator 1 ตัว
- SIEMENS Measurement and Analyzing Unit 1 ตัว
- Linear Load 1000W 1 ชุด
- 3 Phase Digital Power Meter 2 ตัว
- Variac 1 Phase 15 A 4 ตัว
- Bridge Rectifier 1 ชุด
- Fuse 10 A 1 ชุด

<u>ผลการทดสอบประกอบด้วย</u>

ตารางที่ 5.1 ตารางบันทึกผลการทดสอบแรงบิดขณะเปลี่ยน Load

ตารางที่ 5.2 ตารางบันทึกผลการทดสอบการเปลี่ยนความถี่การสวิตช์

ตารางที่ 5.3 ผลทดสอบมอเตอร์เหนี่ยวนำ 3 เฟส ขณะต่อตรงกับแหล่งจ่ายไฟสลับ 3เฟส ไม่ผ่าน Inverter

การทดลองในตารางที่ 5.1 ตารางบันทึกผลการทดสอบแรงบิดขณะเปลี่ยน Load เป็นการทดสอบมอเตอร์เหนี่ยวนำโดยทำการทดสอบแบ่งเป็น 3ช่วงคือ ขณะไม่ต่อ Load และต่อ Load 50% และ100% ตามลำดับ โดยการต่อ Load แต่ละครั้งจะมีการปรับความถี่ไฟฟ้าสลับเป็นค่า 50Hz 40Hz 30Hz และ 20Hz โดยในส่วนของ 20Hz จะทำการปรับดรรชนีมอดดูเลชั่นเพิ่มขึ้นเพื่อเพิ่มแรงดัน โดยที่ในความถี่ 50Hz 40Hz 30Hz และ 20Hz จะปรับค่ามอดดูเลชั่น 100% 80% 60% และ40% ตามลำดับเพื่อให้อัตราส่วนระหว่าง แรงดันต่อความถี่คงที่ โดยจะเห็นได้ว่า ความถี่สูงขึ้นมีผล กับกำลังงาน และความเร็วรอบของมอเตอร์เพิ่มขึ้นตามความถี่ และการปรับดรรชนีมอดดูเลชั่นเป็นการเพิ่มขนาดแรงดันที่มอเตอร์มีผลให้ความเร็วรอบ และกำลังงานที่มอเตอร์เพิ่มขึ้นได้เช่นกัน

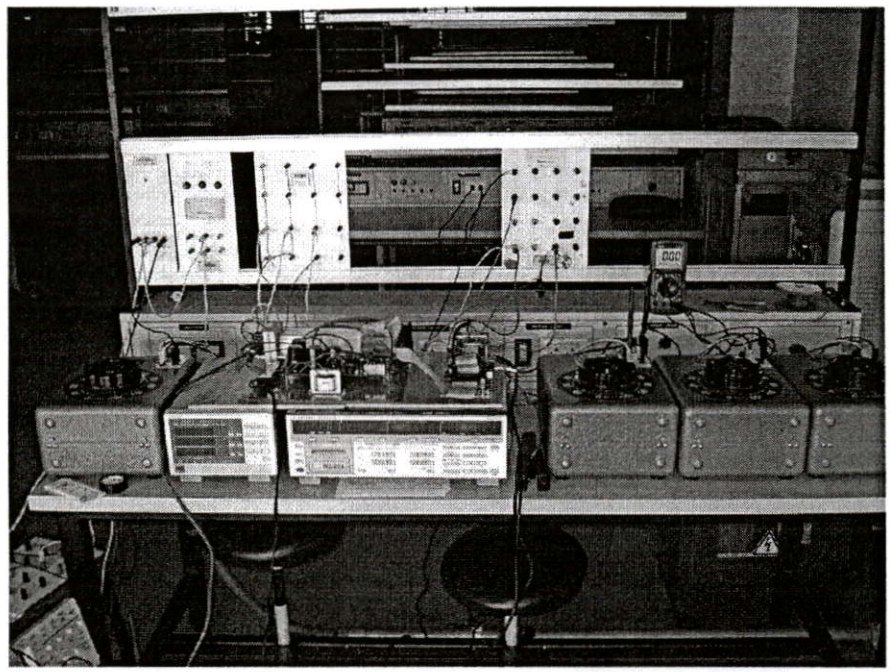
การทดลองในตารางที่ 5.2 ผลการทดสอบการเปลี่ยนความถี่การสวิตช์ มีผลให้ความเร็วรอบของมอเตอร์ลดลงเนื่องจากค่าเฉลี่ยแรงดันที่มอเตอร์จะลดลงด้วย และยิ่งมีการสวิตช์ความถี่สูงขึ้นจะทำให้เกิดการสูญเสียกำลังงานที่สวิตช์มากขึ้น

การทดลองในตารางที่ 5.3 เป็นการทดสอบมอเตอร์เหนี่ยวนำ ขณะที่ไม่ได้ขับมอเตอร์ผ่านชุดอินเวอร์เตอร์ นั่นคือเป็นการขับโดยตรง เพื่อหาคุณสมบัติของมอเตอร์ที่ใช้ทดลอง โดยจะเห็นได้ว่าขณะที่ต่อ Load ให้กับมอเตอร์ 100% ความเร็วรอบจะลดลงจากกรณีที่ Load มีค่าต่ำ

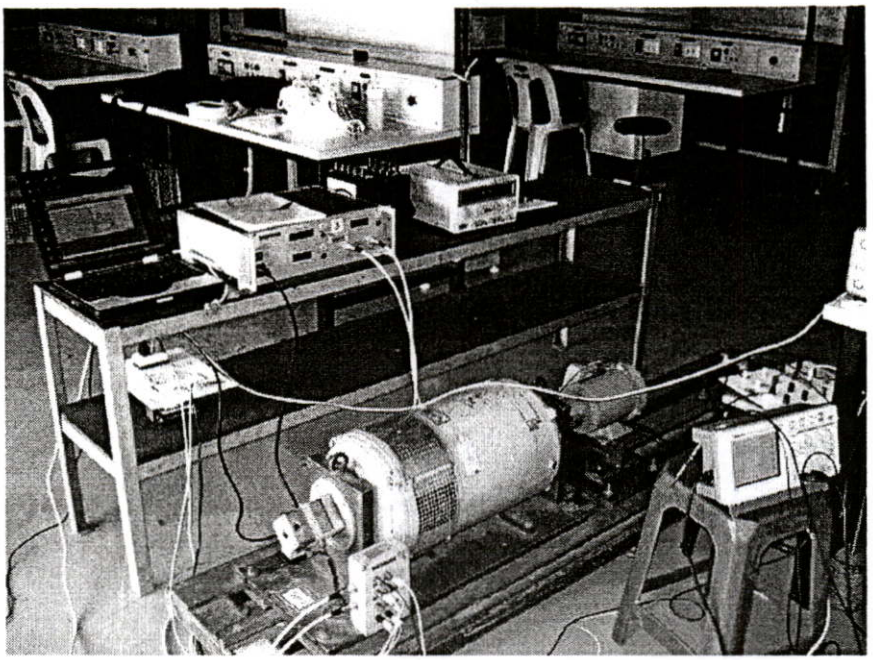
<u>ความหมายของตัวแปรในตารางการทดสอบมอเตอร์</u>

f_{MOTOR} (Motor frequency)	หมายถึง ความถี่หลักมูลของไฟฟ้าสลับที่ป้อนให้กับมอเตอร์ หน่วยคือ Hz(Hertz)
f_s (Switching frequency)	หมายถึง ความถี่การสวิตช์ของ IGBT หน่วยคือ Hz(Hertz)
Deadtime	หมายถึง ช่วงเวลาการหน่วงสัญญาณของสวิตช์ขับในกิ่งเดียวกันเพื่อป้องกันการลัดวงจร หน่วยคือ S(second)
Step	หมายถึง ตัวแปรใน FPGA ที่ใช้ในการปรับค่าความถี่หลักมูลของแรงดันไฟสลับที่ให้กับมอเตอร์
Command	หมายถึง เงื่อนไขการทำงานของอินเวอร์เตอร์โดยเป็นการกำหนดความถี่มอเตอร์ และดรรชนีการมอดดูเลชั่น
Freq (Frequency)	หมายถึง ความถี่สัญญาณไฟฟ้าสลับที่ป้อนให้กับมอเตอร์ หน่วยคือ Hz(Hertz)
Mod(Modulation index)	หมายถึง ดรรชนีการมอดดูเลชั่น หน่วยคือ %
Vin(Input voltage)	หมายถึง แรงดันที่ป้อนเข้าชุดอินเวอร์เตอร์ หน่วยคือ Vrms
Iin(Input current)	หมายถึง กระแสที่ป้อนเข้าชุดอินเวอร์เตอร์ หน่วยคือ Arms

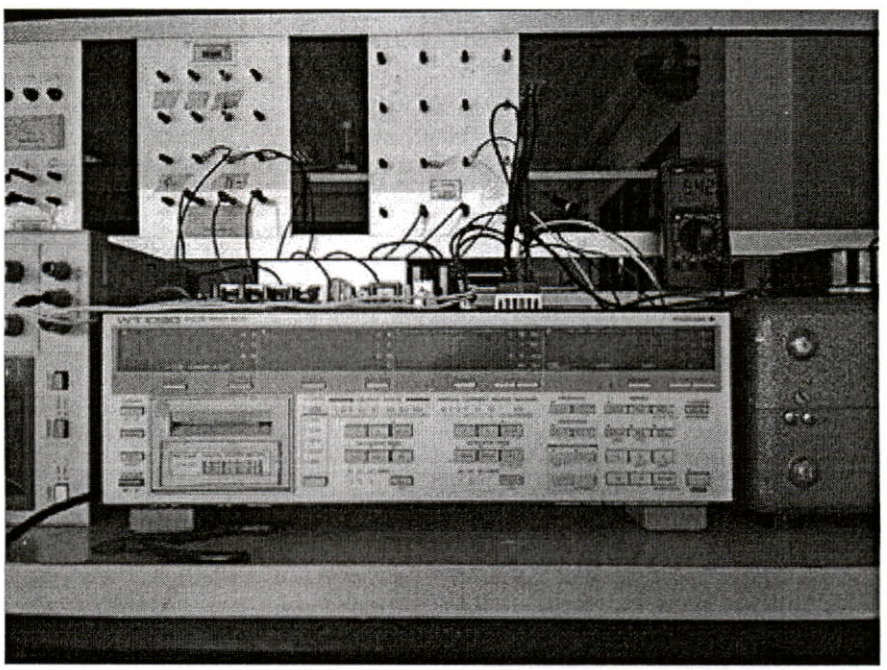
Pin(Power input)	หมายถึง กำลังงานที่ป้อนเข้าชุดอินเวอร์เตอร์ หน่วยคือ KW (Kilo Watts)
V(Output voltage)	หมายถึง แรงดันที่ออกจากชุดอินเวอร์เตอร์ หน่วยคือ Vrms
I(Output current)	หมายถึง กระแสที่ออกจากชุดอินเวอร์เตอร์ หน่วยคือ Arms
Po(Power output)	หมายถึง กำลังงานที่ออกจากชุดอินเวอร์เตอร์ หน่วยคือ KW (Kilo Watts)
Eff(Efficiency)	หมายถึง ค่าประสิทธิผลของปริมาณกำลังงานขาออกเทียบกับขาเข้า
Tq(Torque)	หมายถึง แรงบิด หน่วยคือ nM(newton Metre)
Speed	หมายถึง ความเร็วรอบการหมุนของมอเตอร์เป็นจำนวนรอบต่อนาที หน่วยคือ r.p.m.(revolution per minute)
Pload(load power)	หมายถึง กำลังงานที่วัดที่ Load(Motor) หน่วยคือ W (watt)



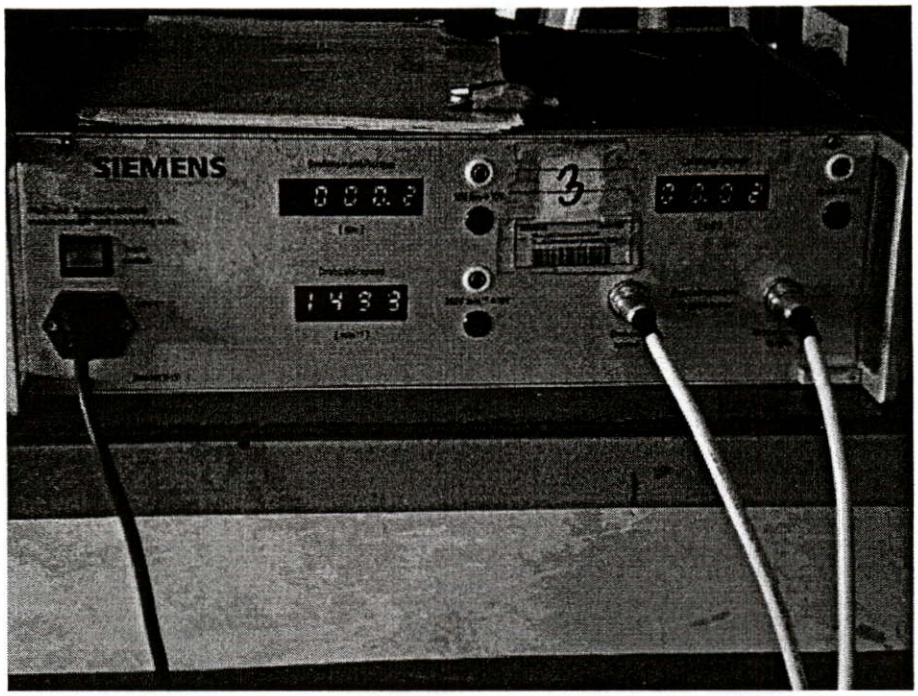
รูปที่ 5.72 ชุดขับอินเวอร์เตอร์ 3เฟสขณะต่ออยู่กับ Variac 1เฟส 3ตัว ที่ทำหน้าที่เป็นตัวจ่ายแรงดัน 3เฟสให้กับชุดไดโอดเรียงกระแส 3เฟส แบบ6คลื่น



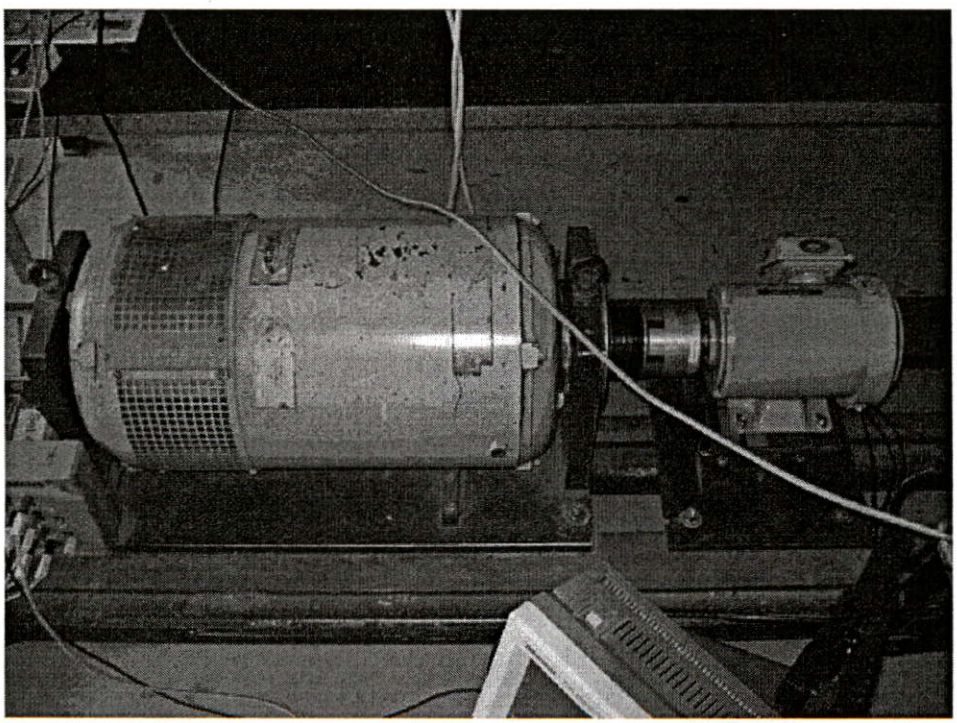
รูปที่ 5.73 อุปกรณ์ที่ใช้ในการทดสอบอินเวอร์เตอร์กับการจ่ายโหลดมอเตอร์เหนี่ยวนำ 3 เฟส ที่มีภาระทางกล



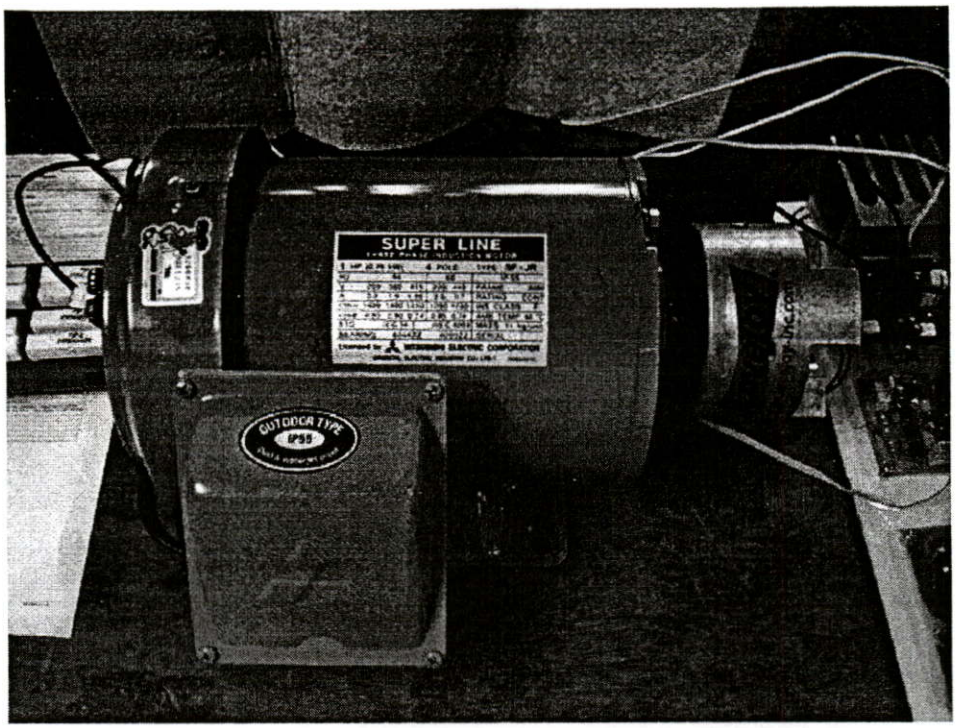
รูปที่ 5.74 Digital Power Meter 3 Phase ยี่ห้อ YOKOKAWA รุ่น WT1030 ที่ใช้บันทึกผล



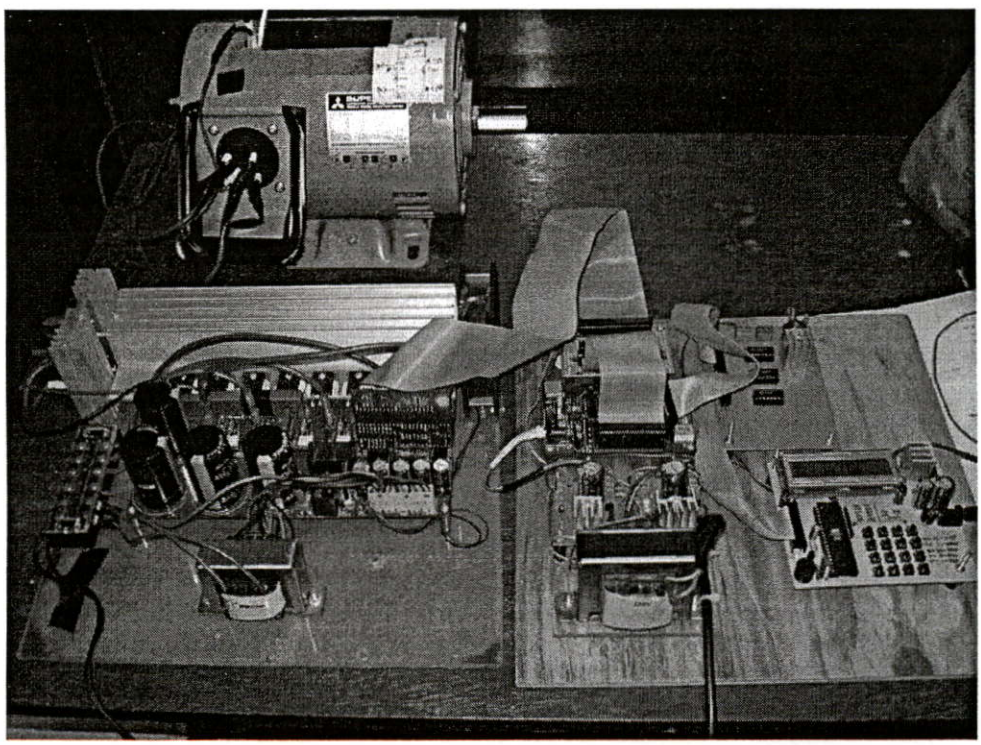
รูปที่ 5.75 Siemens Measurement and Analyzing Unit (วัดทอร์ค, วัดความเร็วรอบ, วัดกำลังที่จ่ายโหลด จากการทำงานของมอเตอร์)



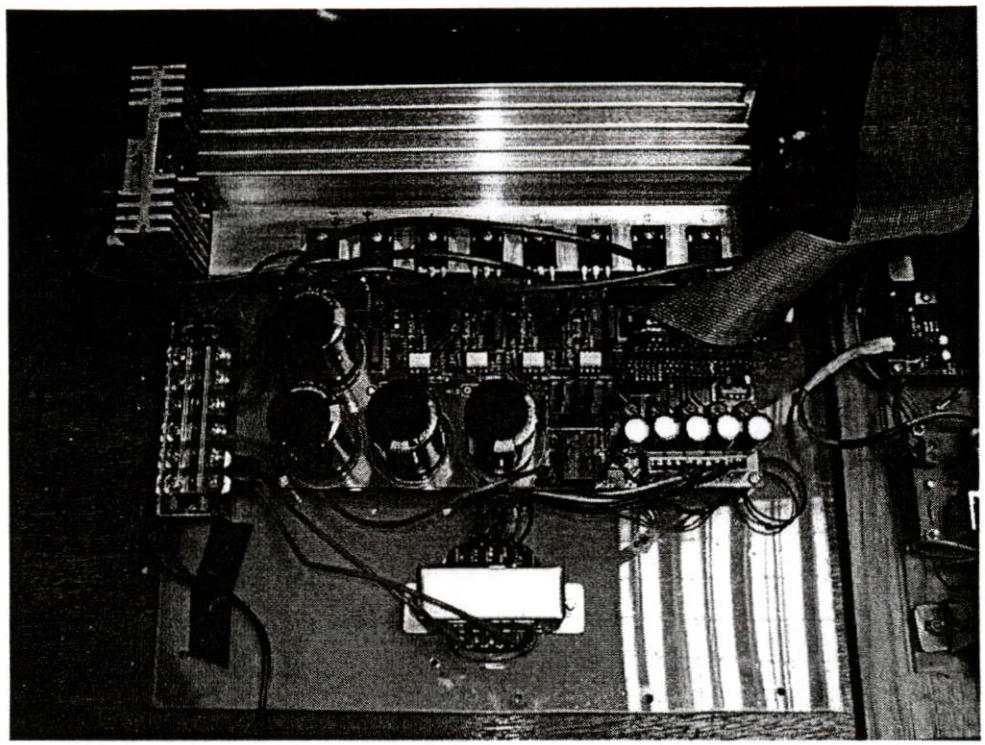
รูปที่ 5.76 การต่อ Motor เข้ากับ Generator ผ่าน Coupling



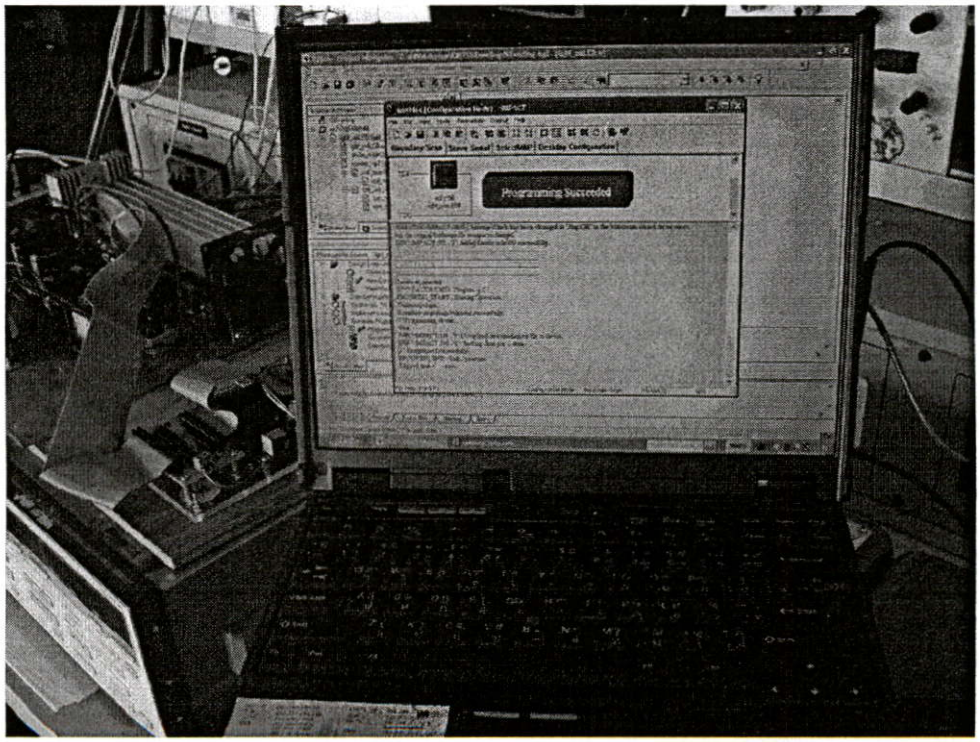
รูปที่ 5.77 มอเตอร์เหนี่ยวนำ 3เฟส



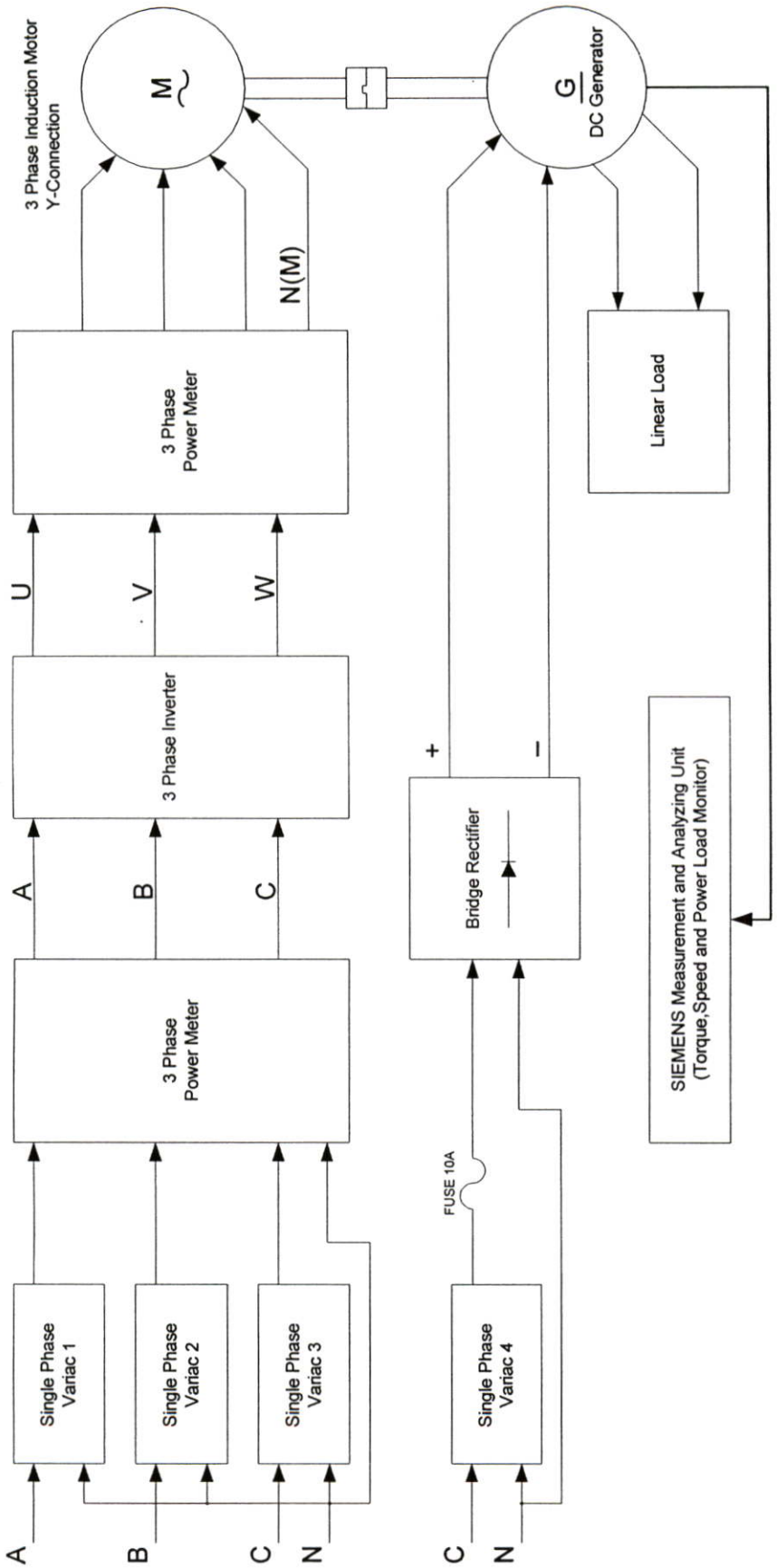
รูปที่ 5.78 ชุดควบคุมอินเวอร์เตอร์ 3เฟส ที่ประกอบไปด้วย บอร์ดชุดขับ IGBT บอร์ด FPGA บอร์ดคอนโทรลเลอร์



รูปที่ 5.79 บอร์ดอินเวอร์เตอร์ในส่วนของชุดสวิตช์ IGBT



รูปที่ 5.80 ภาพหน้าจอคอมพิวเตอร์ขณะทำการโปรแกรม FPGA



รูปที่ 5.81 Block Diagram การต่ออุปกรณ์ต่างๆที่ใช้ในการทดสอบแรงบิดของมอเตอร์เหนี่ยว

ตารางที่ 5.1 ตารางบันทึกผลการทดสอบแรงบิดขณะเปลี่ยน Load

No load , f_s = 15KHz , dead time $2\mu S$												
Command		Input			Output				Motor			
Freq(Hz)	Mod(%)	Vin(Vrms)	Iin(Arms)	Pi(KW)	V(Vrms)	I(Arms)	Po(KW)	Eff(%)	Tq(nM)	Speed	Pload(W)	Eff(%)
50	100	220	0.6	0.156	230	0.865	0.131	83.9	0.2nM	1410	20	15.26
40	80	220	0.52	0.13	206	0.85	0.104	80	0.2nM	1127	20	19.23
30	60	220	0.42	0.1	179.7	0.83	0.076	76	0.2nM	844	10	13.15
20	40	220	0.334	0.078	147.92	0.81	0.052	66.6	0.2nM	561	10	19.23
20	50	220	0.455	0.105	165.43	1.15	0.078	74.28	0.2nM	564	10	12.82
20	60	220	0.709	0.184	180.17	1.814	0.148	80.43	0.2nM	565	10	6.7
20	80	220	2.35	0.768	204.43	4.12	0.698	90.88	0.2nM	565	10	1.432
50% load , f_s = 15KHz , dead time $2\mu S$												
50	100	220	1.54	0.49	227.3	1.253	0.48	97.95	2.3nM	1354	320	66
40	80	220	1.4	0.44	202.36	1.27	0.4	90.9	2.3nM	1065	235	58
30	60	220	1.15	0.35	177.2	1.28	0.31	88.57	2.3nM	772	190	61.29
20	40	220	0.93	0.254	143	1.3	0.22	86.6	2.3nM	470	110	50
20	50	220	0.945	0.263	163	1.276	0.228	86.66	2.3nM	516	120	52
20	60	220	1.09	0.303	178.3	1.663	0.27	89.1	2.3nM	536	120	44
20	80	220	2.46	0.810	203.5	3.9	0.734	90.6	2.3nM	549	120	16
100% load , f_s = 15KHz , dead time $2\mu S$												
50	100	220	3.3	1.2	225	2.51	1.14	95	5.5nM	1225	700	61.4
40	80	220	2.89	1.016	200	2.57	0.94	92	5.5nM	917	510	54.5
30	60	220	2.5	0.87	176.7	2.979	0.81	93	5.5nM	530	290	35.8
20	40	220	1.42	0.423	141.58	2.13	0.38	89.8	3.5nM	330	120	31.57
20	50	220	1.826	0.59	161.36	2.41	0.54	91.52	5.5nM	376	210	38.88
20	60	220	1.712	0.67	176.34	2.012	0.615	91.79	5.5nM	476	260	42.27
20	80	220	2.67	0.89	201.84	3.44	0.82	92	5.5nM	520	290	35.365

ตารางที่ 5.2 ตารางบันทึกผลการทดสอบการเปลี่ยนความถี่การสวิตช์

f_{MOTOR} = 50Hz , f_s = 20KHz , dead time $2\mu S$, step = 4												
Command		Input			Output				Motor			
Load(%)	Mod(%)	Vin(Vrms)	Iin(Arms)	Pi(KW)	V(Vrms)	I(Arms)	Po(KW)	Eff(%)	Tq(nM)	Speed	Pload(W)	Eff(%)
0	100	220	0.65	0.153	233	0.785	0.12	78.4	0.2nM	1492	20	16.6
50	100	220	1.77	0.55	231	1.314	0.51	92.7	2.3nM	1419	330	64.7
100	100	220	3.6	1.28	230	2.706	1.21	94.53	5.5nM	1237	700	57.8
f_{MOTOR} = 58.59Hz , f_s = 30KHz , dead time $2\mu S$, step = 3												
Command		Input			Output				Motor			
Load(%)	Mod(%)	Vin(Vrms)	Iin(Arms)	Pi(KW)	V(Vrms)	I(Arms)	Po(KW)	Eff(%)	Tq(nM)	Speed	Pload(W)	Eff(%)
0	100	220	0.6	0.164	240.5	0.69	0.12	73.1	0.2nM	1646	30	25
50	100	220	1.84	0.620	235.5	1.378	0.565	91.1	2.3nM	1543	360	63.7
100	100	220	4.17	1.6	232.5	3.59	1.377	86	5.5nM	1140	600	43.57

ตารางที่ 5.3 ผลทดสอบมอเตอร์เหนี่ยวนำ 3 เฟส ขณะต่อตรงกับแหล่งจ่ายไฟสลับ 3เฟส ไม่ผ่าน Inverter

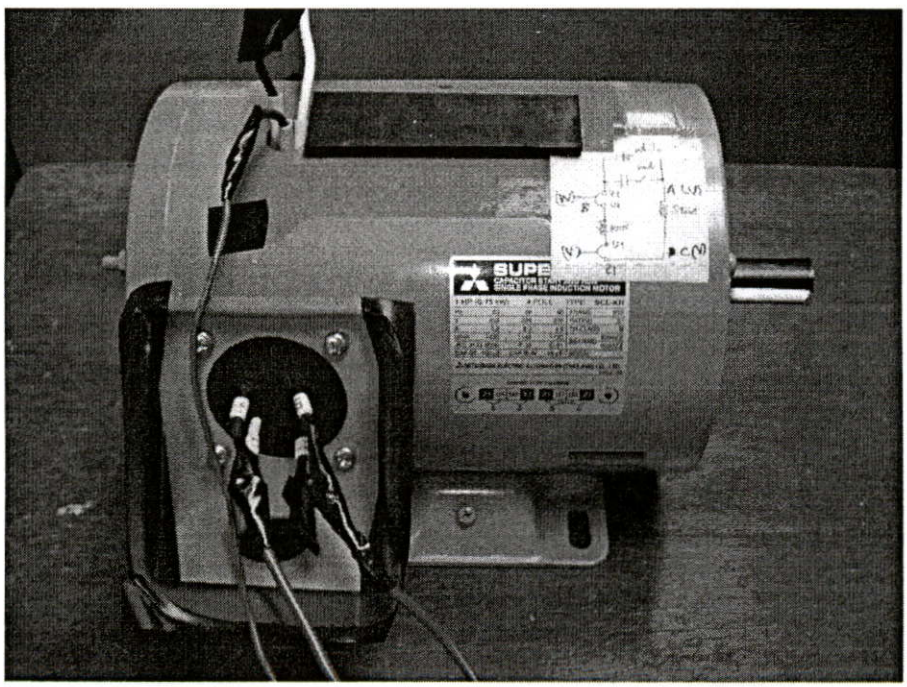
f_{MOTOR} = 50Hz ,Direct Drive to AC Line 220V												
Command		Input			Output				Motor			
Load(%)	Mod(%)	Vin(Vrms)	Iin(Arms)	Pi(KW)	V(Vrms)	I(Arms)	Po(KW)	Eff(%)	Tq(nM)	Speed	Pload(W)	Eff(%)
0	-	220	1.071	0.15	219.6	1.013	0.15	100	0.2nM	1497	0	0
50	-	220	1.35	0.562	219.6	1.338	0.566	100	2.3nM	1447	370	65.37
100	-	220	2	1.097	219	1.99	1.0925	99.58	5.5nM	1376	750	68.64

5.2 ผลการทดลองมอเตอร์เหนี่ยวนำกระแสสลับ 1เฟส

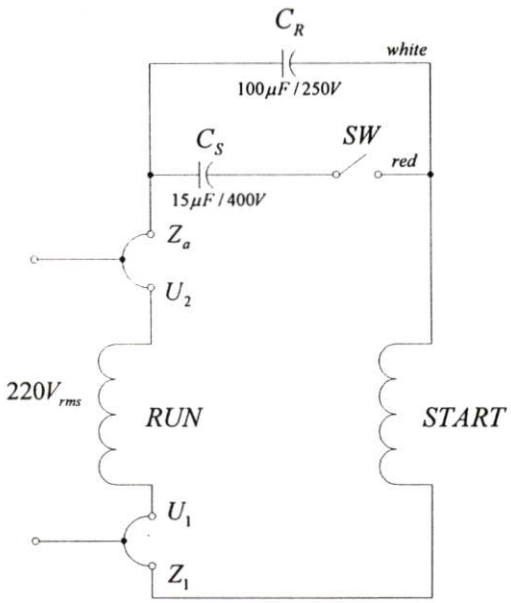
การทดลองในส่วนของมอเตอร์เหนี่ยวนำแบบ 1เฟสนั้นยังคงใช้วงจรอินเวอร์เตอร์เดิม เช่นเดียวกับแบบ 3เฟสเพียงแต่มีการเปลี่ยนสัญญาณควบคุม SVM PWM จาก FPGA โดยเลือกใช้สัญญาณในส่วนของการขับมอเตอร์เหนี่ยวนำแบบ 1เฟส แทนดังแสดงการต่อในรูปที่ 5.2 และการต่อมอเตอร์เหนี่ยวนำแบบ 1เฟสเข้ากับชุดอินเวอร์เตอร์ ดังแสดงในรูปที่ 3.5 โดยการทดลองในส่วนของมอเตอร์เหนี่ยวนำแบบ 1เฟส จะมีเพียงส่วนของการวัดสัญญาณทางเวลา และสัญญาณทางความถี่เท่านั้น ในส่วนของการต่อภาระทางกลไม่ได้ทำการทดสอบ

5.2.1 ผลการวัดค่าแรงดัน และกระแสของมอเตอร์เหนี่ยวนำกระแสสลับ 1เฟสขณะต่อตรงกับแหล่งจ่ายไฟสลับโดยไม่ผ่านอินเวอร์เตอร์

โดยการวัดสัญญาณจะวัดที่แรงดัน $220V, 50Hz$ การวัดสัญญาณจะแบ่งเป็น 2ส่วนใหญ่ๆ คือวัดขณะที่มอเตอร์เหนี่ยวนำแบบ 1เฟส ต่อตรงกับแรงดันไฟฟ้าสลับ $220V, 50Hz$ โดยไม่ผ่านวงจรอินเวอร์เตอร์ เพื่อวัดสัญญาณแรงดัน และกระแสขณะเริ่มทำงาน และขณะทำงานปกติ โดยอีกส่วนจะการวัดสัญญาณโดยต่อผ่านวงจรอินเวอร์เตอร์

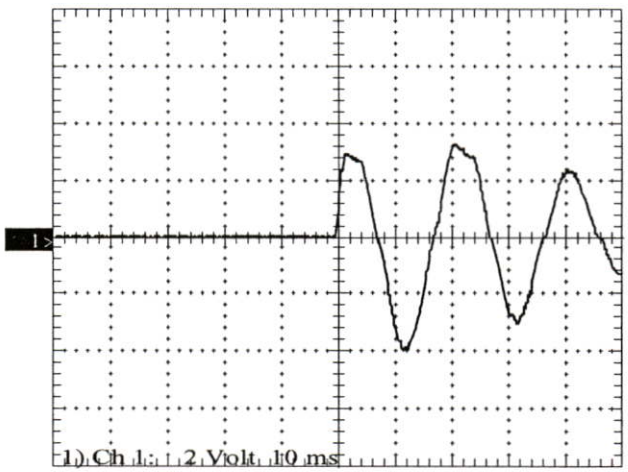


รูปที่ 5.82 มอเตอร์เหนี่ยวนำไฟฟ้ากระแสสลับ 1เฟส ขนาด 1แรงม้า (750W)

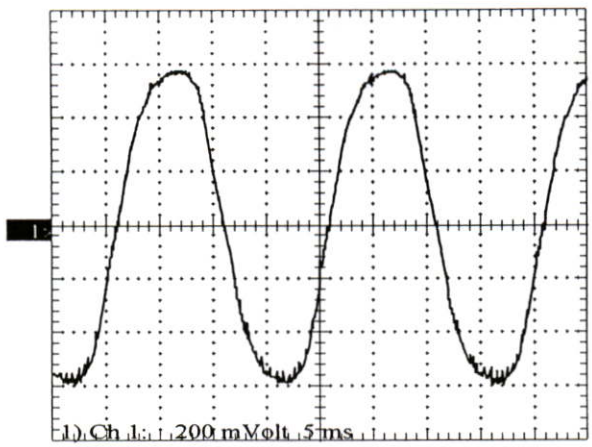


รูปที่ 5.83 แผนภาพการต่อภายในมอเตอร์เหนี่ยวนำกระแสสลับ1เฟส

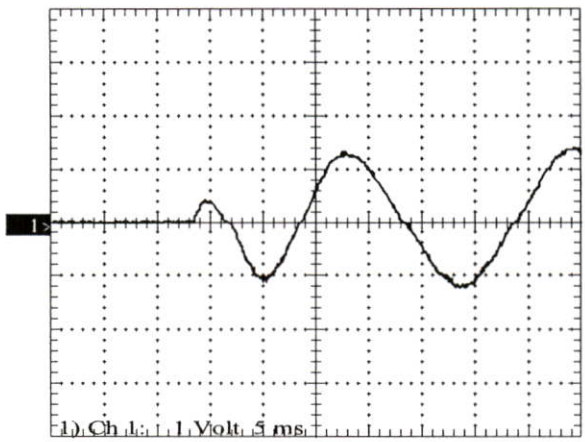
ตั้งค่าโปรบวัดกระแส $100mV / A$



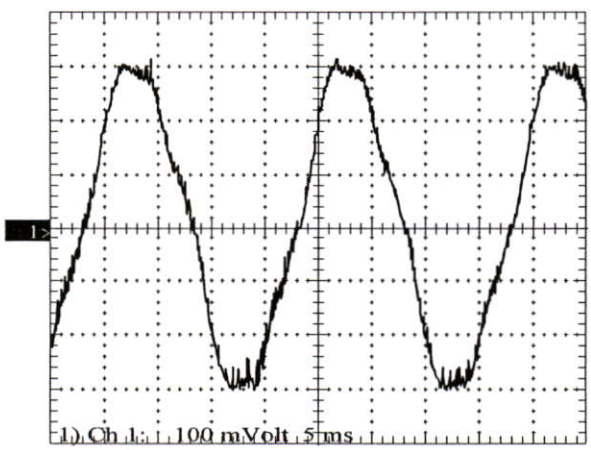
รูปที่ 5.84 กระแสที่ขด Run ขณะมอเตอร์เริ่มทำงาน



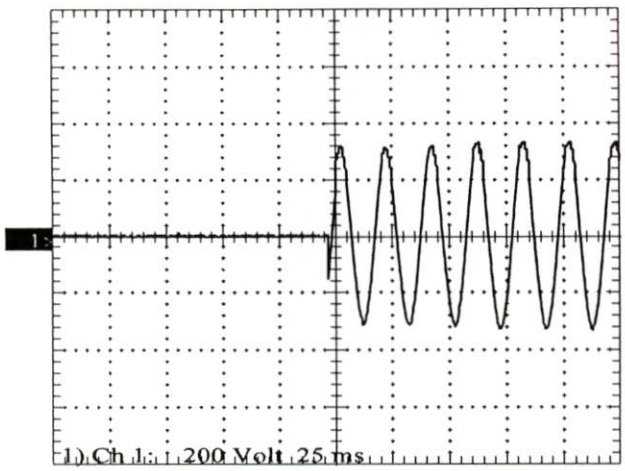
รูปที่ 5.85 กระแสที่ขด Run ขณะทำงานปกติ



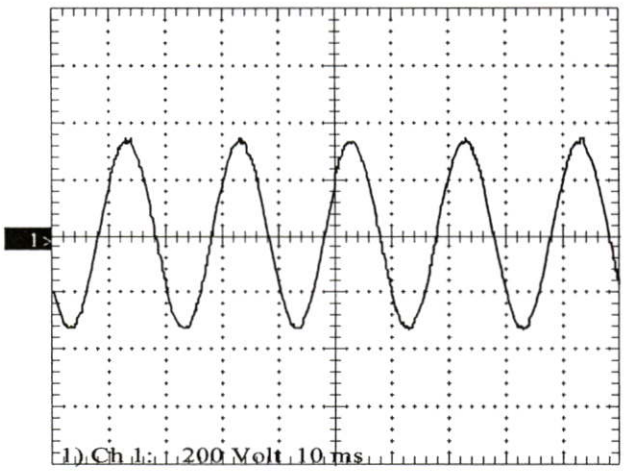
รูปที่ 5.86 กระแสที่ขด Start ขณะเริ่มทำงาน



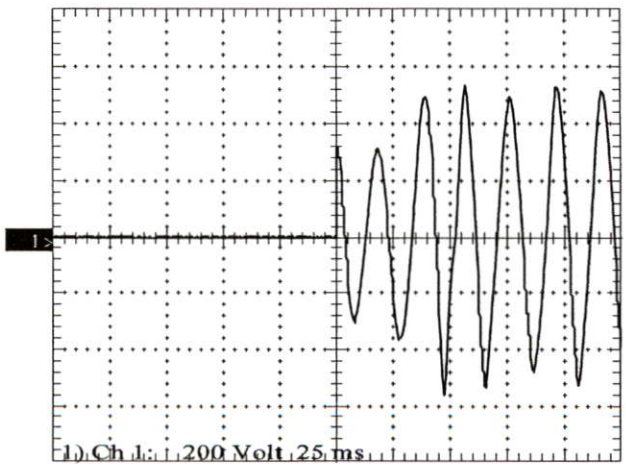
รูปที่ 5.87 กระแสที่ขด Start ขณะทำงานปกติ



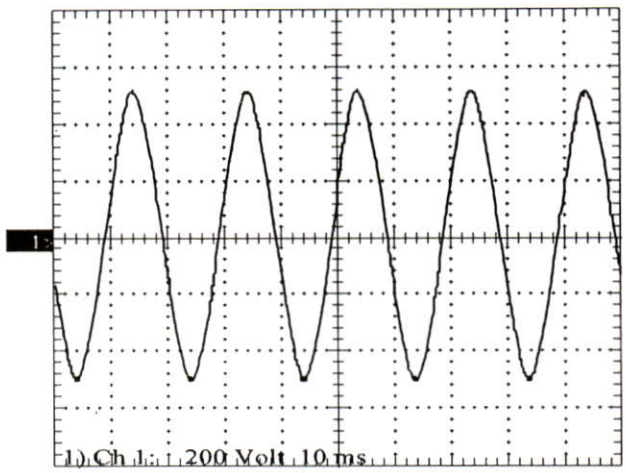
รูปที่ 5.88 แรงดันที่ขด Run ขณะเริ่มทำงาน



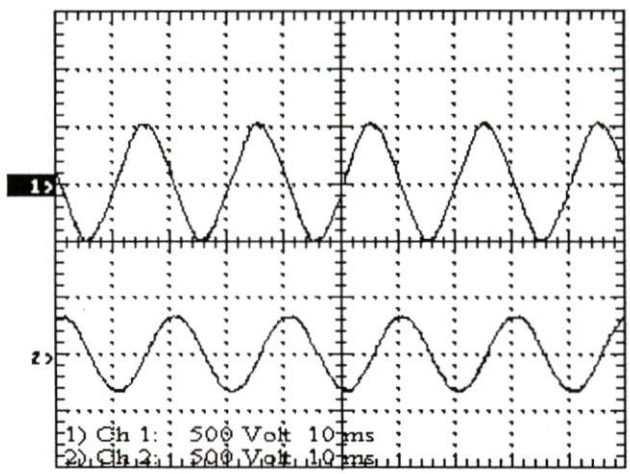
รูปที่ 5.89 แรงดันที่ขด Run ขณะทำงานปกติ



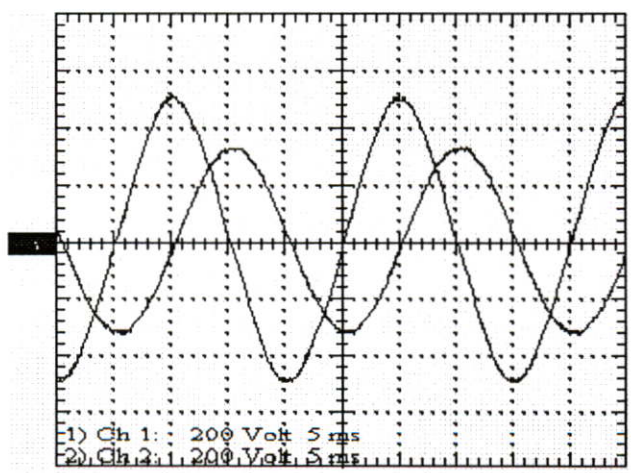
รูปที่ 5.90 แรงดันที่ขด Start ขณะเริ่มทำงาน



รูปที่ 5.91 แรงดันที่ขด Start ขณะทำงานปกติ



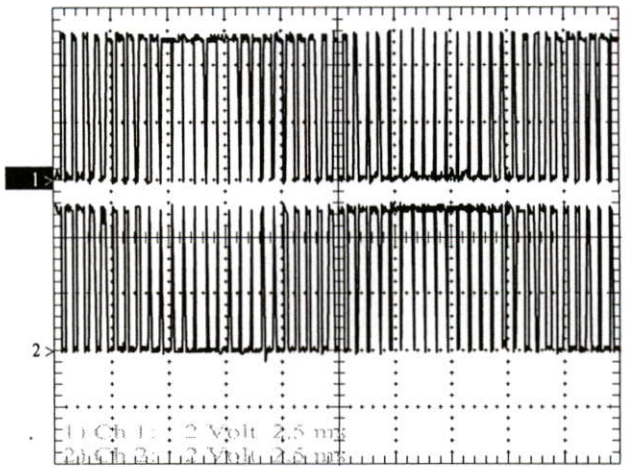
รูปที่ 5.92 แรงดันที่ขด Start (Ch1) เทียบกับที่ขด Run (Ch2)



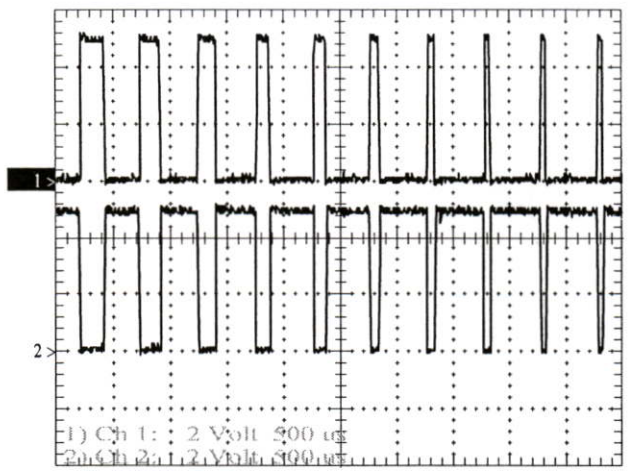
รูปที่ 5.93 แรงดันที่ขด Start (Ch1) เทียบกับที่ขด Run (Ch2)

5.2.2 ผลการวัดค่าแรงดัน และกระแสของมอเตอร์เหนี่ยวนำกระแสสลับ 1เฟสขณะต่อ ผ่านอินเวอร์เตอร์

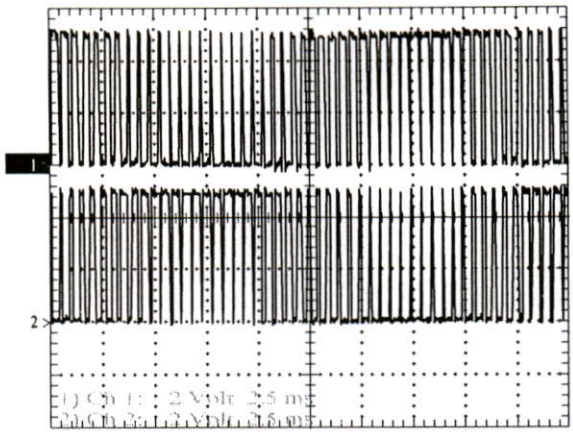
5.2.2.1 สัญญาณ PWM จาก FPGA หลังผ่านวงจรแปลงระดับแรงดันที่จุดต่างๆ



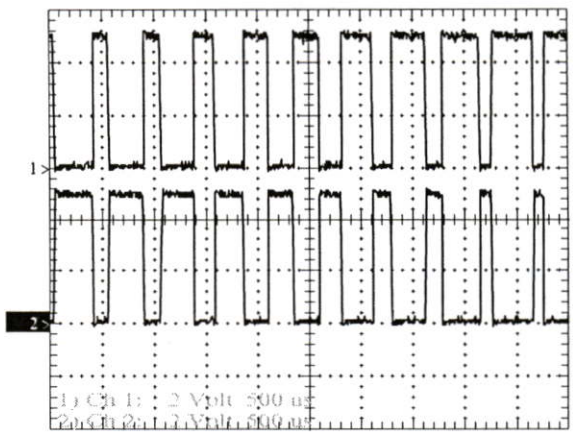
รูปที่ 5.94 รูปสัญญาณ PWM เอ้าท์พุตของ CD4050B ที่ Va (CH1) และ $\overline{Va}$ (CH2)



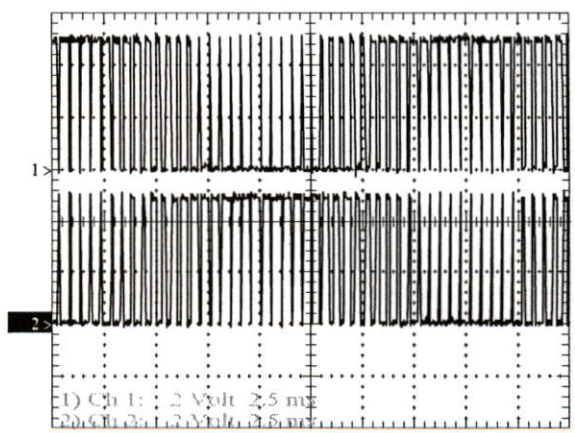
รูปที่ 5.95 รูปขยายสัญญาณ PWM ที่ เอ้าท์พุตของ CD4050B ที่ Va (CH1) และ $\overline{Va}$ (CH2) แสดงความถี่การสวิตช์ $2KHz$



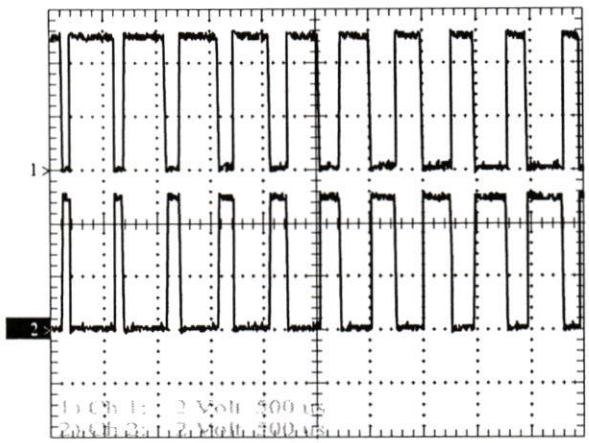
รูปที่ 5.96 รูปสัญญาณ PWM ที่ เอ้าท์พุตของ CD4050B ที่ Vb (CH1)และ $\overline{Vb}$ (CH2)



รูปที่ 5.97 รูปขยายสัญญาณ PWM ที่ เอ้าท์พุตของ CD4050B ที่ Vb (CH1) และ $\overline{Vb}$ (CH2) แสดงความถี่การสวิตช์ $2KHz$

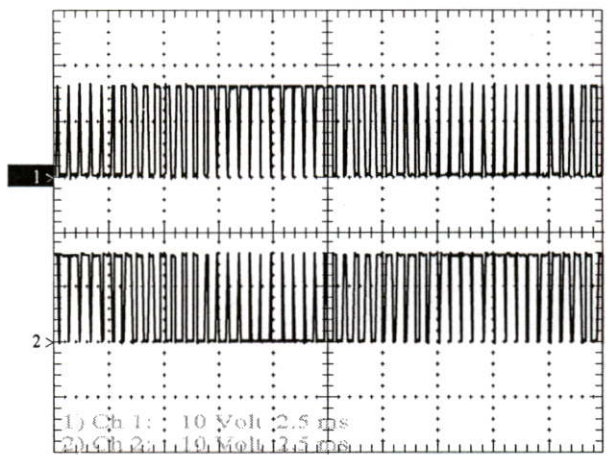


รูปที่ 5.98 รูปสัญญาณ PWM ที่ เอ้าท์พุตของ CD4050B ที่ Vc (CH1) และ $\overline{Vc}$ (CH2)

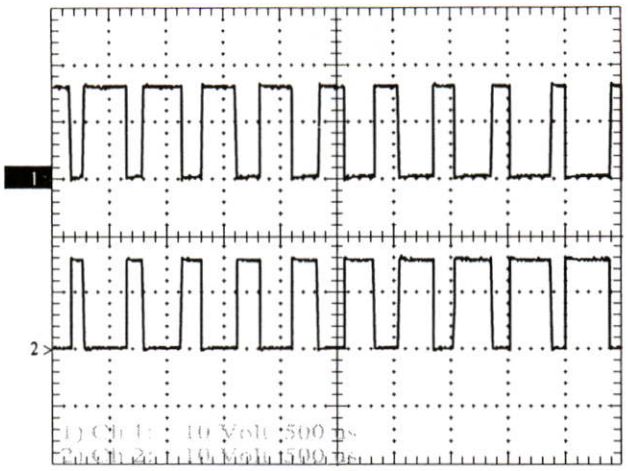


รูปที่ 5.99 รูปขยายสัญญาณ PWM ที่ เอ้าท์พุตของ CD4050B ที่ Vc (CH1) และ $\overline{Vc}$ (CH2) แสดงความถี่การสวิตช์ $2KHz$

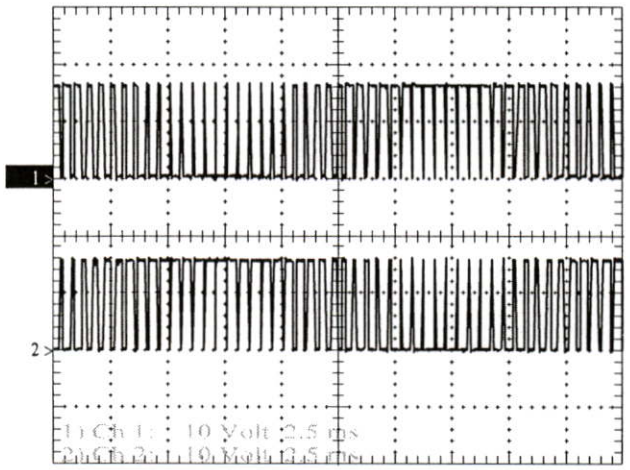
5.2.2.2 สัญญาณ PWM วัดที่ขาเกตของ IGBT ที่จุดต่างๆ



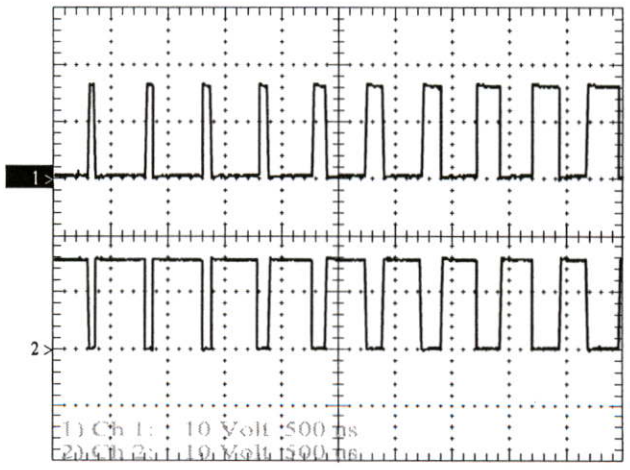
รูปที่ 5.100 รูปสัญญาณ PWM ที่ขาเกตของสวิตช์ Q1U(CH1) และ Q4U(CH2)



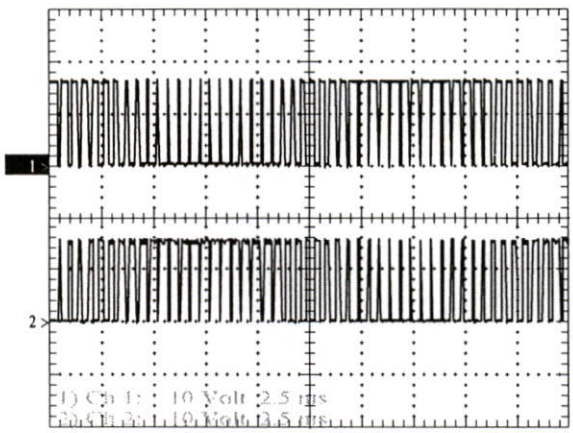
รูปที่ 5.101 รูปขยายสัญญาณ PWM ที่ขาเกตของสวิตช์ Q1U(CH1) และ Q4U(CH2)



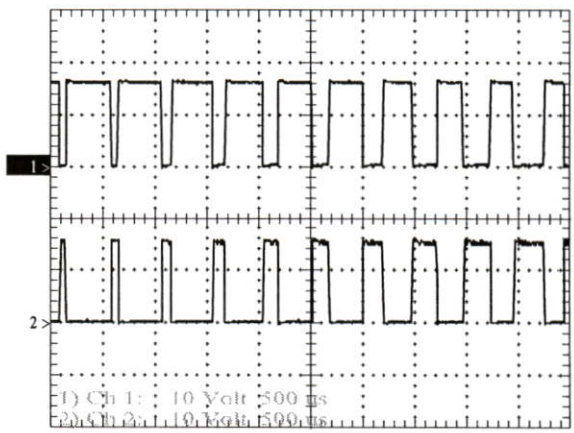
รูปที่ 5.102 รูปสัญญาณ PWM ที่ขาเกตของสวิตช์ Q3V(CH1) และ Q6V(CH2)



รูปที่ 5.103 รูปขยายสัญญาณ PWM ที่ขาเกตของสวิตช์ Q3V(CH1) และ Q6V(CH2)

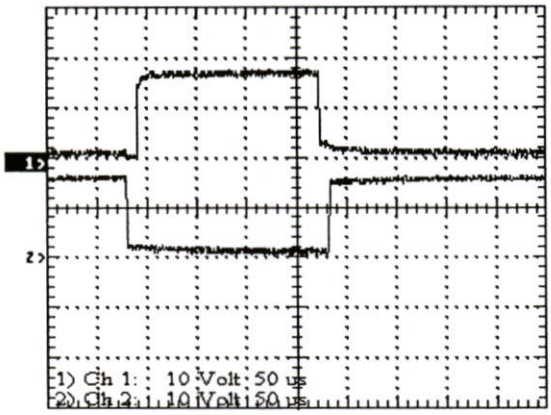


รูปที่ 5.104 รูปสัญญาณ PWM ที่ขาเกตของสวิตช์ Q5W(CH1) และ Q2W(CH2)

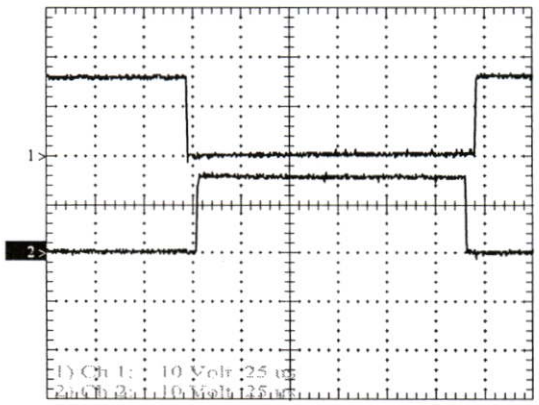


รูปที่ 5.105 รูปขยายสัญญาณ PWM ที่ขาเกตของสวิตช์ Q5W(CH1) และ Q2W(CH2)

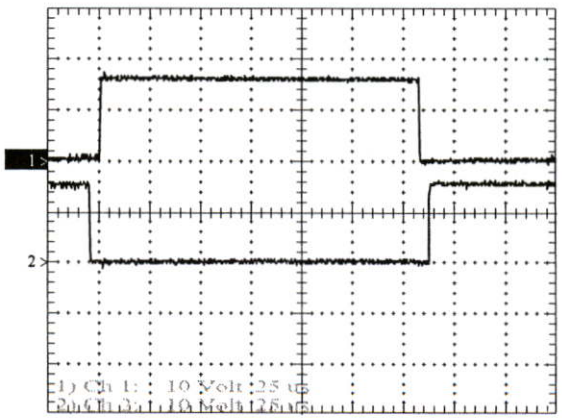
5.2.2.3 แสดง Dead time ของรูปสัญญาณที่ขาเกต ของ IGBT



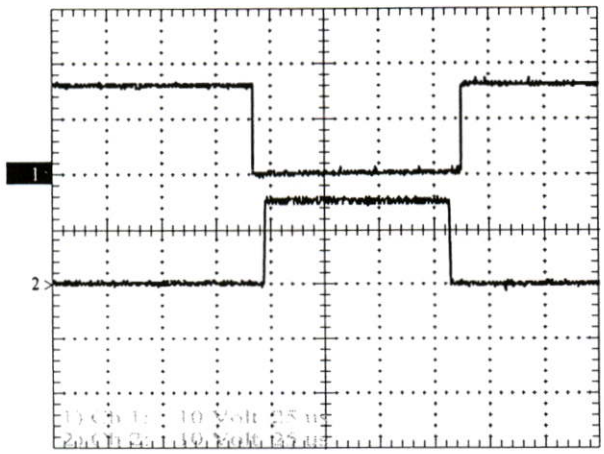
รูปที่ 5.106 รูปขยายสัญญาณแสดง Dead time $10\mu S$ ที่ขาเกตของ Q1U(CH1) และ Q4U(CH2)



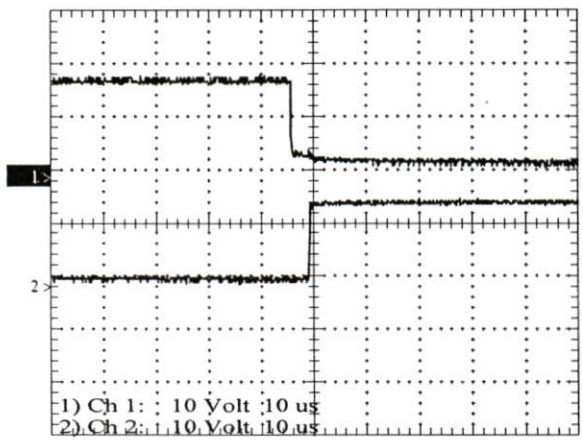
รูปที่ 5.107 รูปขยายสัญญาณแสดง Dead time $5\mu S$ ที่ขาเกตของ Q1U(CH1) และ Q4U(CH2)



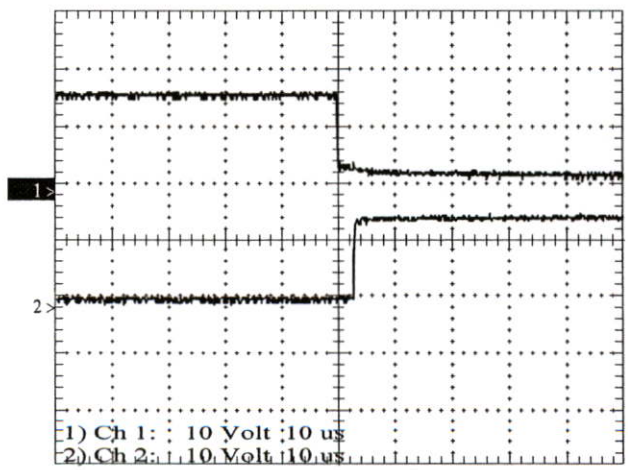
รูปที่ 5.108 รูปขยายสัญญาณแสดง Dead time $5\mu S$ ที่ขาเกตของ Q3V(CH1) และ Q6V(CH2)



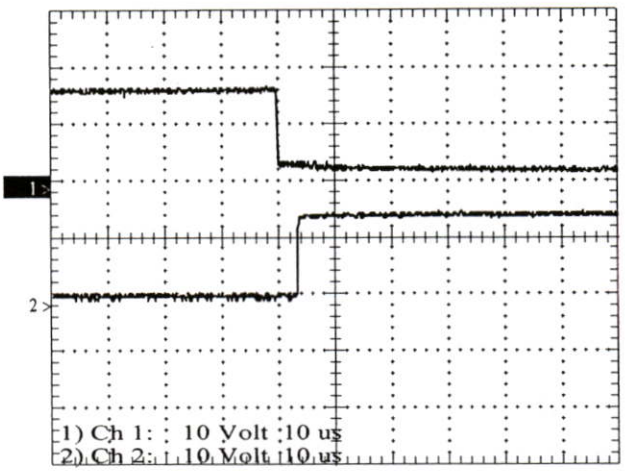
รูปที่ 5.109 รูปขยายสัญญาณแสดง Dead time $5\mu S$ ที่ขาเกตของ Q5W(CH1) และ Q2W(CH2)



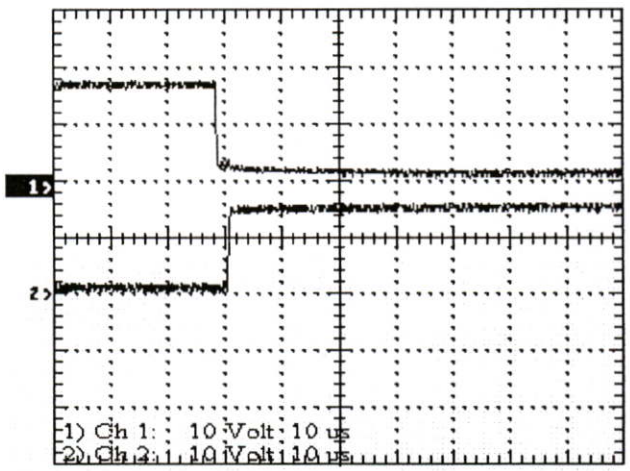
รูปที่ 5.110 รูปขยายสัญญาณแสดง Dead time $3\mu S$ ที่ขาเกตของ Q1U(CH1) และ Q4U(CH2)



รูปที่ 5.111 รูปขยายสัญญาณแสดง Dead time $3\mu S$ ที่ขาเกตของ Q3V(CH1) และ Q6V(CH2)

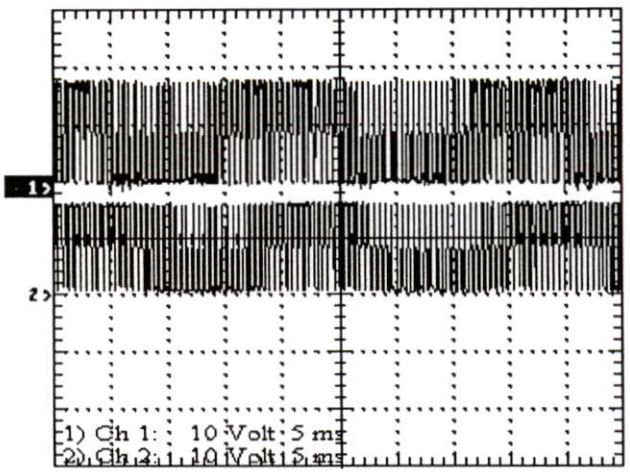


รูปที่ 5.112 รูปขยายสัญญาณแสดง Dead time $3\mu S$ ที่ขาเกตของ Q5W(CH1) และ Q2W(CH2)

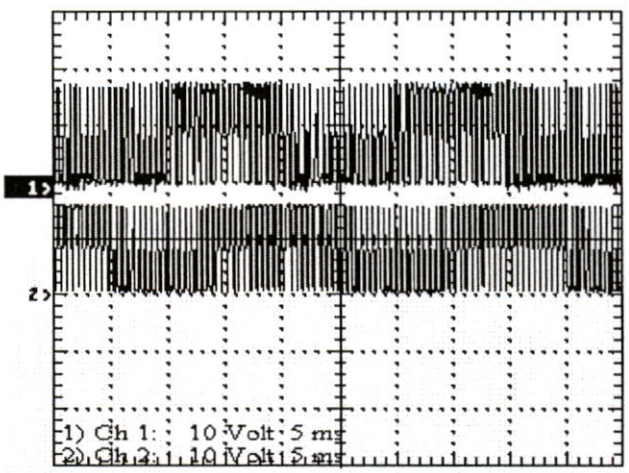


รูปที่ 5.113 รูปขยายสัญญาณแสดง Dead time $2\mu S$ ที่ขาเกตของ Q1U(CH1) และ Q4U(CH2)

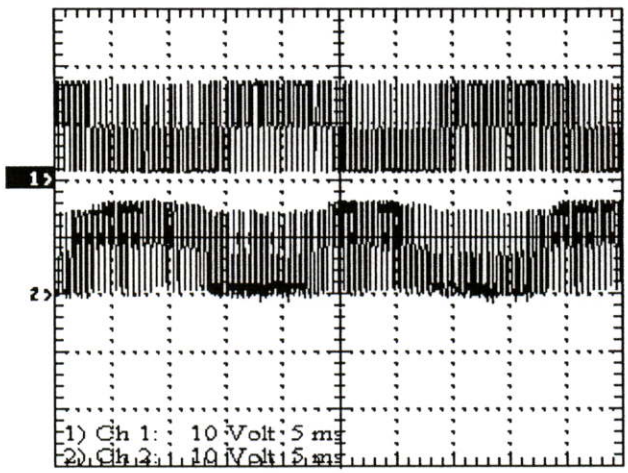
5.2.2.4 แสดงเฟสต่างของสัญญาณที่ ขาเกตของ Q1U เทียบ ขาเกตของ Q3V



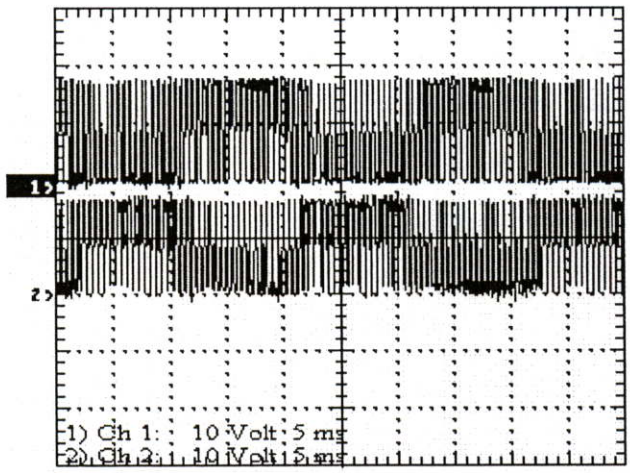
รูปที่ 5.114 เฟสต่าง 45 องศา ของสัญญาณที่ ขาเกตของ Q1U เทียบ ขาเกตของ Q3V



รูปที่ 5.115 เฟสต่าง 90 องศา ของสัญญาณที่ ขาเกตของ Q1U เทียบ ขาเกตของ Q3V

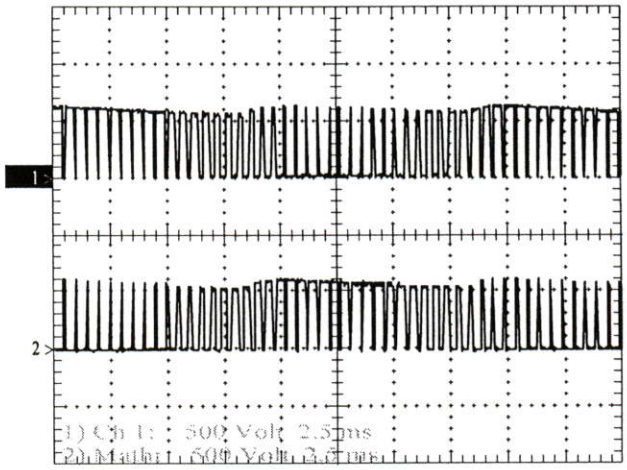


รูปที่ 5.116 เฟสต่าง 135 องศา ของสัญญาณที่ ขาเกตของ Q1U เทียบ ขาเกตของ Q3V

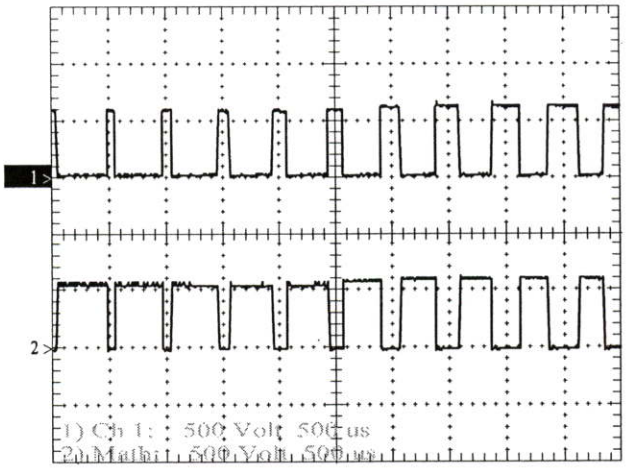


รูปที่ 5.117 เฟสต่างของสัญญาณที่ ขาเกตของ Q1U เทียบ ขาเกตของ Q3V จะมีค่าต่างกัน180องศา

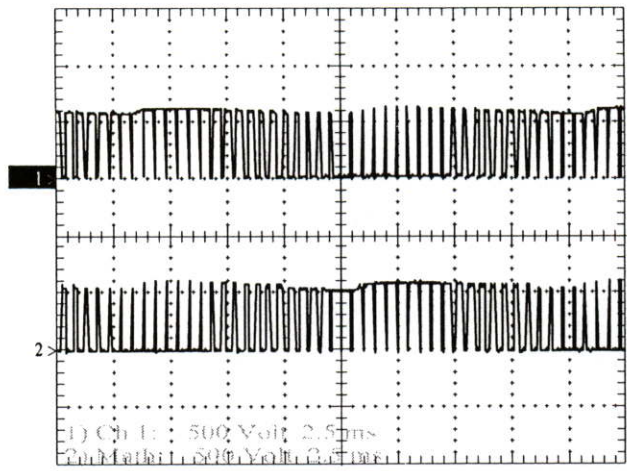
5.2.2.5 แสดงรูปสัญญาณที่ขา C, E ของสวิตช์



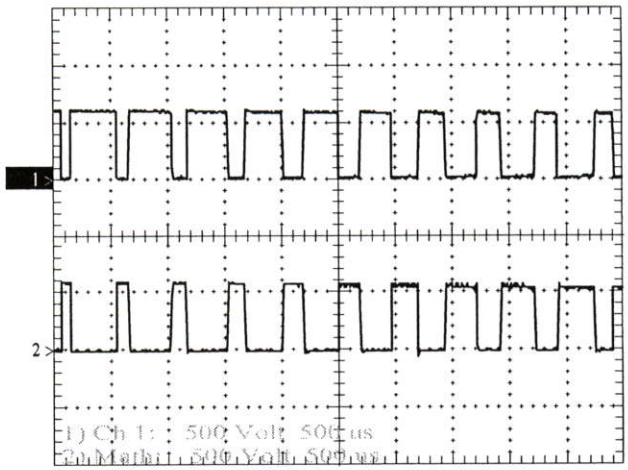
รูปที่ 5.118 รูปสัญญาณที่ขา C, E ของสวิตช์ Q1U(CH1) และ Q4U(CH2)



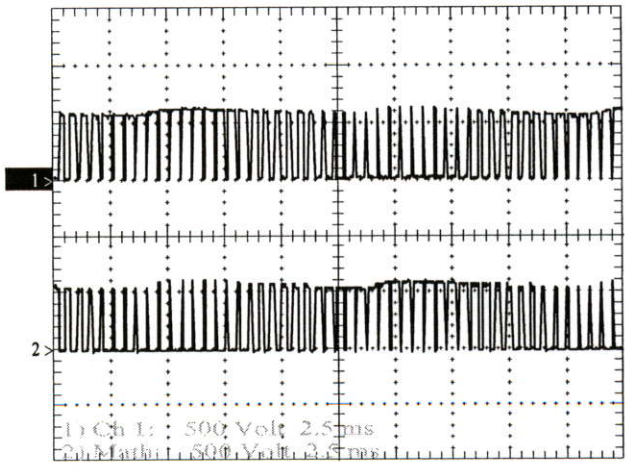
รูปที่ 5.119 รูปขยายสัญญาณที่ขา C, E ของสวิตช์ Q1U(CH1) และ Q4U(CH2)



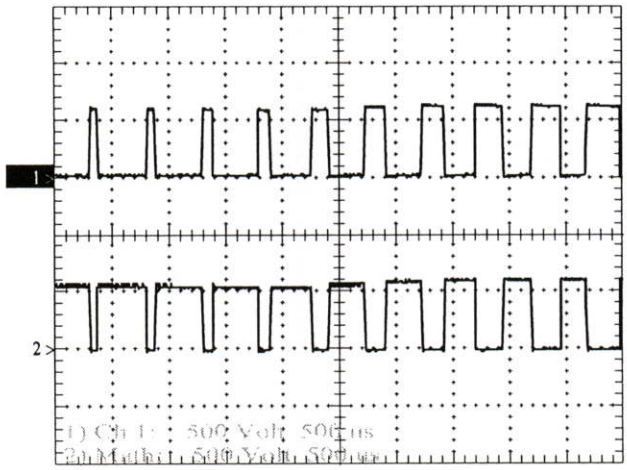
รูปที่ 5.120 รูปสัญญาณที่ขา C, E ของสวิตช์ Q3V(CH1) และ Q6V(CH2)



รูปที่ 5.121 รูปขยายสัญญาณที่ขา C, E ของสวิตช์ Q3V(CH1) และ Q6V(CH2)

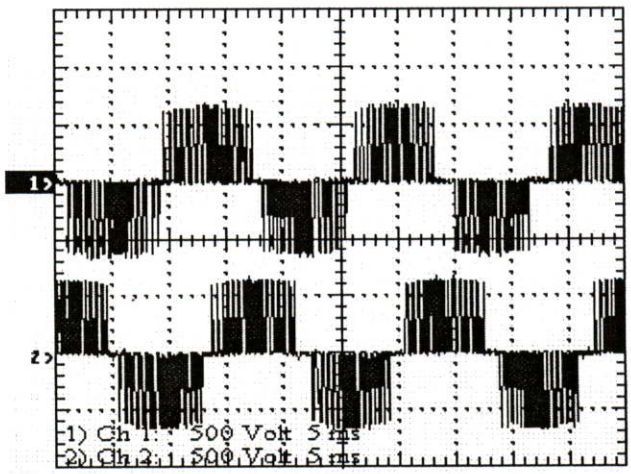


รูปที่ 5.122 รูปสัญญาณที่ขา C, E ของสวิตช์ Q5W(CH1) และ Q2W(CH2)

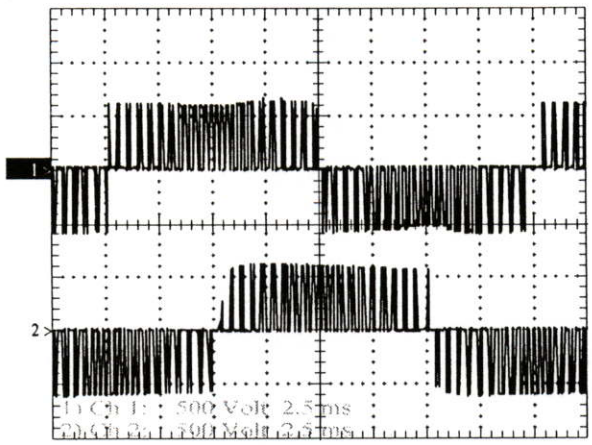


รูปที่ 5.123 รูปขยายสัญญาณที่ขา C, E ของสวิตช์ Q5W(CH1) และ Q2W(CH2)

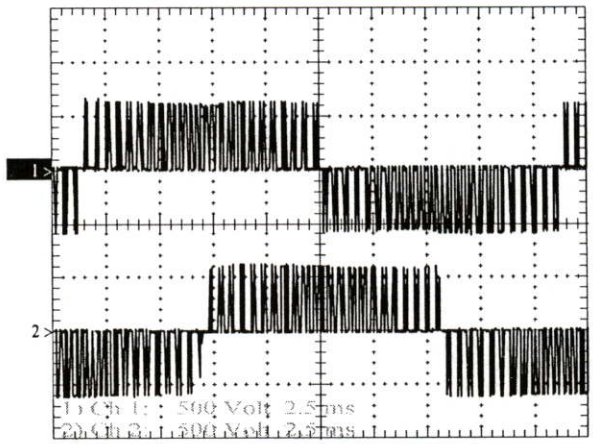
5.2.2.6 แสดงรูปสัญญาณ SVM PWM แรงดันที่ขด Start และขด Run ที่ความถี่ค่าต่างๆ



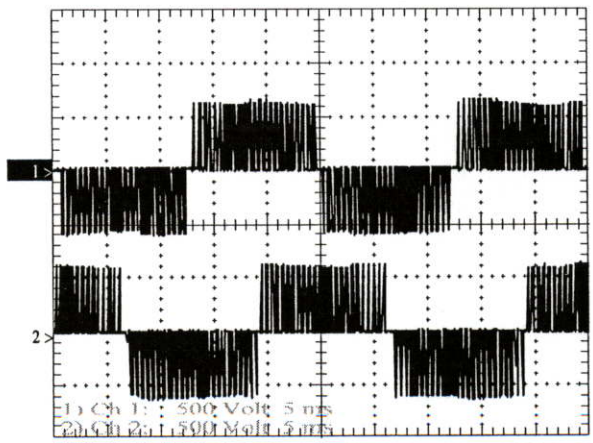
รูปที่ 5.124 รูปสัญญาณแรงดันที่ขด Start(CH1) และขด Run(CH2) f = 60 Hz



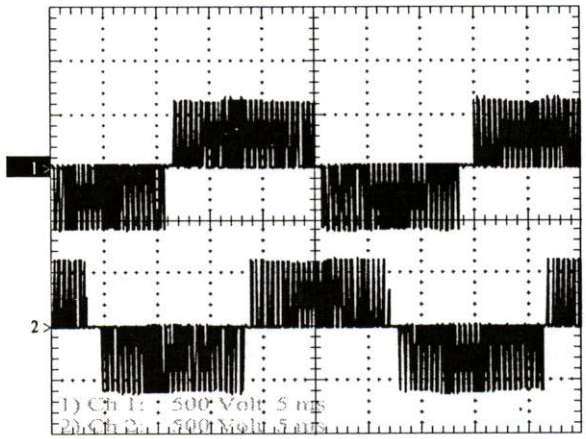
รูปที่ 5.125 รูปสัญญาณแรงดันที่ขด Start(CH1) และขด Run(CH2) f = 50 Hz



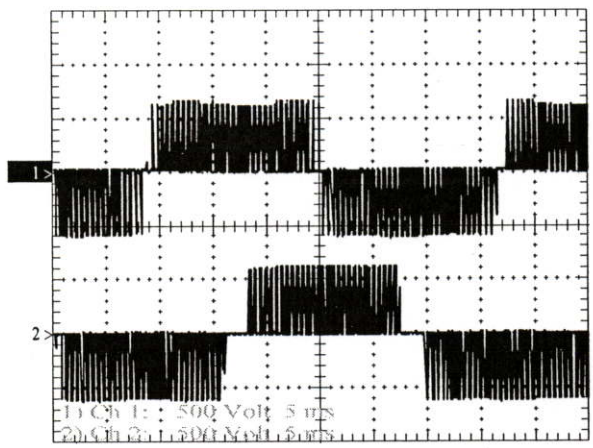
รูปที่ 5.126 รูปสัญญาณแรงดันที่ขด Start(CH1) และขด Run(CH2) f = 45 Hz



รูปที่ 5.127 รูปสัญญาณแรงดันที่ขด Start(CH1) และขด Run(CH2) f = 40 Hz

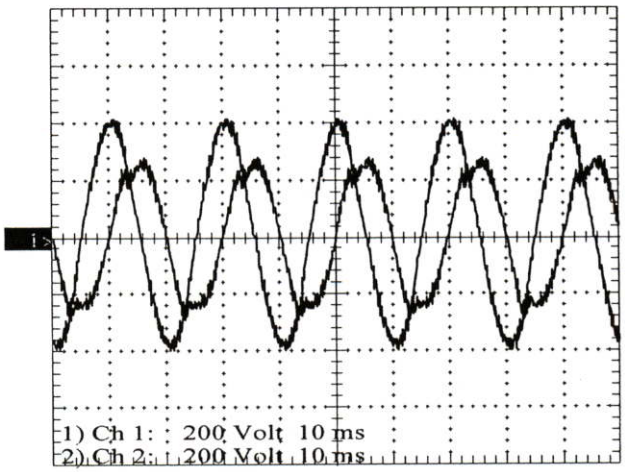


รูปที่ 5.128 รูปสัญญาณแรงดันที่ขด Start(CH1) และขด Run(CH2) f = 35 Hz

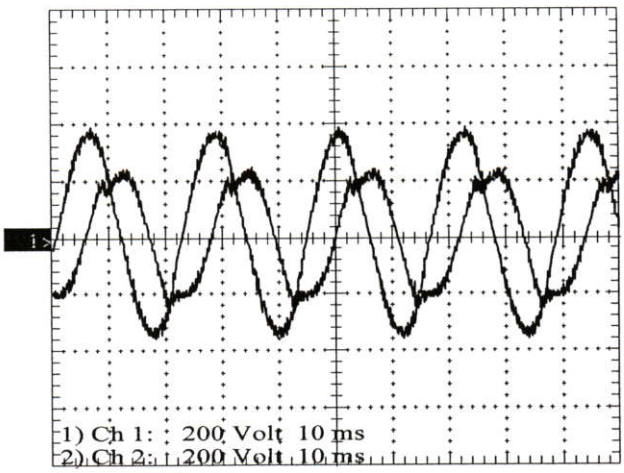


รูปที่ 5.129 รูปสัญญาณแรงดันที่ขด Start(CH1) และขด Run(CH2) f = 30 Hz

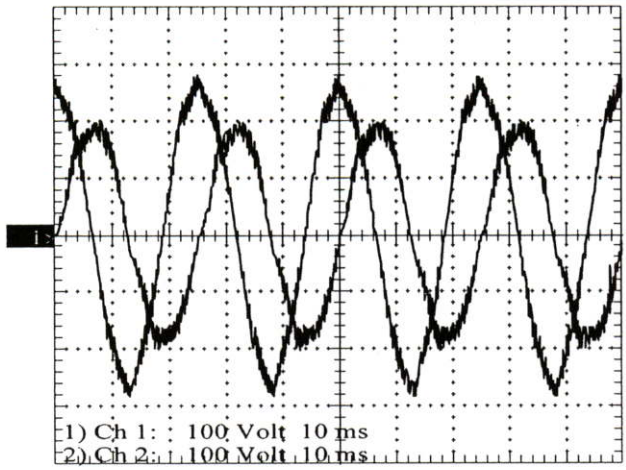
5.2.2.7 แสดงรูปสัญญาณ SVM PWM แรงดันที่ขด Start และ ขด Run โดยวัดสัญญาณ ผ่านวงจรกรองความถี่ต่ำผ่าน



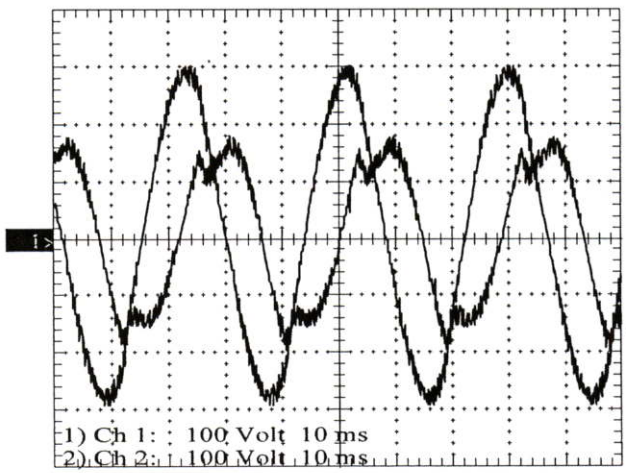
รูปที่ 5.130 รูปแรงดันหลักมูล ที่ขด Start(Ch1) และขด Run(Ch2) f = 50 Hz (วัดผ่านวงจรกรองสัญญาณความถี่ต่ำผ่าน)



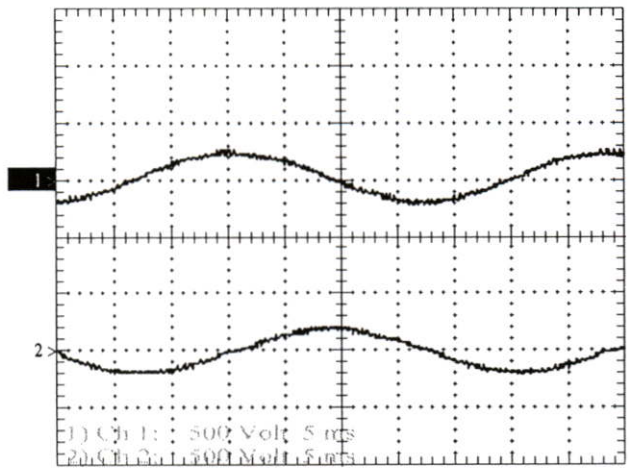
รูปที่ 5.131 รูปแรงดันหลักมูล ที่ขด Start(Ch1) และขด Run(Ch2) f = 45 Hz (วัดผ่านวงจรกรองสัญญาณความถี่ต่ำผ่าน)



รูปที่ 5.132 รูปแรงดันหลักมูล ที่ขด Start(Ch1) และขด Run(Ch2) f = 40 Hz
(วัดผ่านวงจรกรองสัญญาณความถี่ต่ำผ่าน)

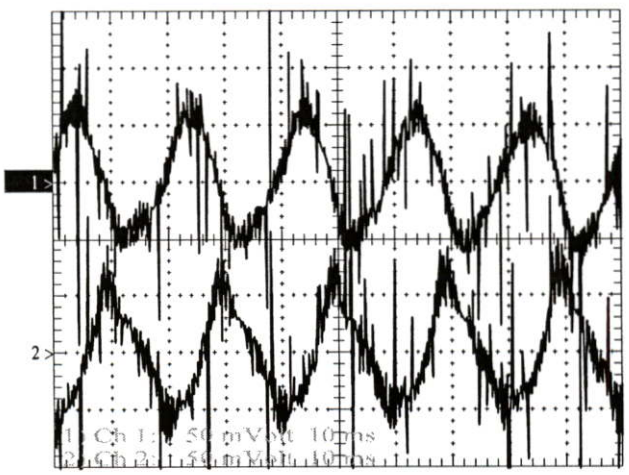


รูปที่ 5.133 รูปแรงดันหลักมูล ที่ขด Start(Ch1) และขด Run(Ch2) f = 35 Hz
(วัดผ่านวงจรกรองสัญญาณความถี่ต่ำผ่าน)

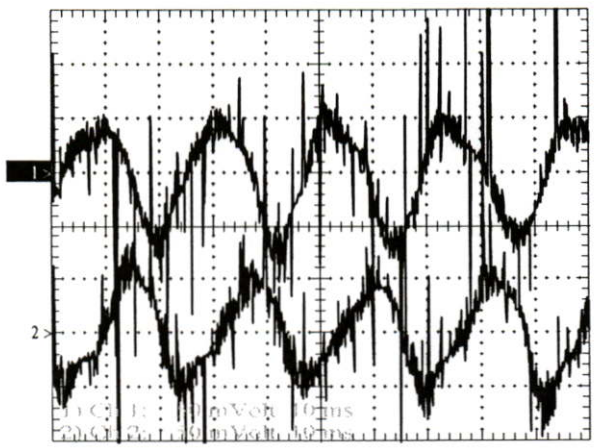


รูปที่ 5.134 รูปแรงดันหลักมูล ที่ขด Start(Ch1) และขด Run(Ch2) f = 30 Hz
(วัดผ่านวงจรกรองสัญญาณความถี่ต่ำผ่าน)

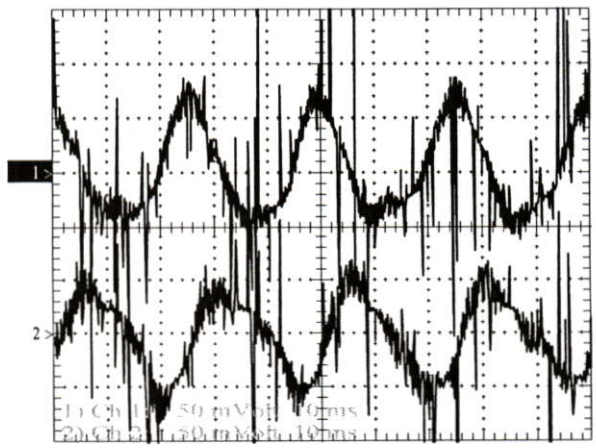
5.2.2.8 แสดงรูปสัญญาณกระแสที่ขด Start และ ขด Run



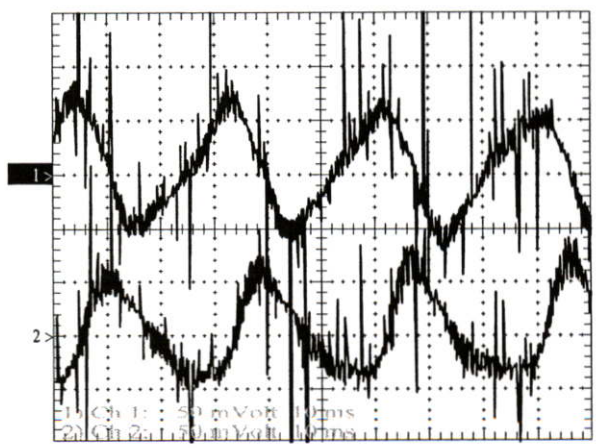
รูปที่ 5.135 รูปสัญญาณกระแสที่ขด Start(Ch1) และขด Run(Ch2) f = 50 Hz



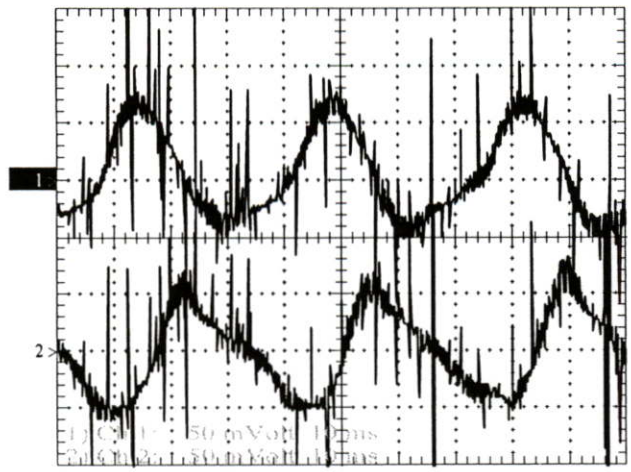
รูปที่ 5.136 รูปสัญญาณกระแสที่ขด Start(Ch1) และขด Run(Ch2) f = 45 Hz



รูปที่ 5.137 รูปสัญญาณกระแสที่ขด Start(Ch1) และขด Run(Ch2) f = 40 Hz



รูปที่ 5.138 รูปสัญญาณกระแสที่ขด Start(Ch1) และขด Run(Ch2) f = 35 Hz



รูปที่ 5.139 รูปสัญญาณกระแสที่ขด Start(Ch1) และขด Run(Ch2) f = 30 Hz

5.2.2.9 ความเร็วรอบของมอเตอร์เหนี่ยวนำกระแสสลับ 1 เฟส ที่ความถี่ต่างๆ

ตารางที่ 5.4 แสดงค่าความเร็วรอบ ที่ความถี่แรงดันไฟฟ้าสลับ ค่า 50,45,40,35 และ30 Hz. โดยวัดค่าด้วย TACHOMETER YOKOGAYA MODEL3631

ตารางที่ 5.4 แสดงความเร็วรอบของมอเตอร์เหนี่ยวนำ 1เฟสที่ความถี่หลักมูลค่าต่างๆ

ความถี่(*Hz*)	ความเร็วรอบ(*RPM*)
50	1450
45	1314
40	1152
35	998
30	825

จากการทดลองในส่วนของอินเวอร์เตอร์ขับมอเตอร์เหนี่ยวนำ 1เฟสมีเทคนิคแตกต่างจากการขับมอเตอร์เหนี่ยว 3เฟสอยู่ที่การเขียนโปรแกรม(ดังอธิบายในหัวข้อ 4.4.2) และการต่อมอเตอร์เหนี่ยวนำ 1เฟสเข้ากับวงจรอินเวอร์เตอร์ที่ใช้โครงสร้างเดียวกันกับอินเวอร์เตอร์สำหรับขับมอเตอร์เหนี่ยวนำ 3เฟสซึ่งจากการทดลองสามารถปรับค่าตัวแปรต่างๆสำหรับควบคุมมอเตอร์เหนี่ยวนำ 1เฟสได้เช่นเดียวกันกับมอเตอร์เหนี่ยวนำ 3เฟส แต่ผลของแรงดันดังในหัวข้อ 5.2.2.7 นั้นจะเห็นได้ว่า ไม่ได้เป็นรูปสัญญาณซานย์สมบรูณ์ ซึ่งมีความผิดเพี้ยนอยู่บ้างอันเนื่องจะการหักล้างของแรงดันเฟส ไม่สามารถหักล้างฮาร์มอนิกประกอบได้หมด

บทที่6
สรุปผลการวิจัย และข้อเสนอแนะ

เทคนิค SVM (Space Vector Modulation) ได้ถูกนำมาใช้ในงานวิจัย และออกแบบอินเวอร์เตอร์ 3เฟส เพื่อสร้างสัญญาณ PWM ควบคุมมอเตอร์เหนี่ยวนำกระแสสลับแบบ 3เฟสอย่างแพร่หลาย ในวิทยานิพนธ์นี้ได้อาศัยหลักการเทคนิค SVM (Space Vector Modulation) เช่นเดียวกันกับในงานวิจัยอินเวอร์เตอร์ 3 เฟส ที่กล่าวมาแล้ว โดยในวิทยานิพนธ์นี้จะประยุกต์ใช้เทคนิค SVM PWM บน FPGA โดยอินพุทพารามิเตอร์ ที่ป้อนให้กับอินเวอร์เตอร์ เพื่อควบคุมมอเตอร์ สามารถควบคุมได้ด้วย ระบบดิจิตอลทั้งหมด ซึ่งทำให้สามารถนำไปใช้กับมอเตอร์เหนี่ยวนำกระแสสลับได้ทั้ง 1เฟส และ 3เฟส โดยการใช้งานกับมอเตอร์ 3เฟสนั้นสัญญาณ SVM PWM ที่ได้จะมีลักษณะคล้ายคลึงกับวิธีบวกฮาร์มอนิกส์ที่3 ดังรูปที่ 5.14 โดยเมื่อนำสัญญาณแรงดันเฟสมาลบกันจะได้สัญญาณแรงดันสายที่มีลักษณะเป็นซานย์ ดังรูปที่ 5.43 ซึ่งผ่านวงจรกรองความถี่ต่ำผ่านแล้ว ในขณะเดียวกันหากนำวิธี SVM PWM ไปใช้ควบคุมมอเตอร์เหนี่ยวนำกระแสสลับแบบ 1เฟสโดยการควบคุมมุมเฟสระหว่างขดรัน และขดช่วยให้มีมุมต่างเฟส 90องศา นั้นก็สามารถใช้ควบคุมมอเตอร์ 1เฟสได้ แต่จะเกิดฮาร์มอนิกตัวอื่นประกอบมาด้วย เนื่องจากสัญญาณ SVM PWM ที่ขดทั้งสองไม่สามารถหักล้างฮาร์มอนิกที่ไม่ต้องการได้หมด ดังนั้นวิธี SVM PWM จึงเหมาะกับการควบคุมมอเตอร์เหนี่ยวนำแบบ 3เฟส มากกว่าที่จะนำไปควบคุมมอเตอร์เหนี่ยวนำแบบ 1เฟส แต่ในงานวิจัยที่เคยมีผู้นำเสนอหลักการ SVM เพื่อควบคุมมอเตอร์เหนี่ยวนำที่ผ่านมานั้นสามารถควบคุมได้เพียงมอเตอร์เหนี่ยวนำ 3เฟสเพียงอย่างเดียว

เทคนิค SVM PWM สามารถให้ค่าแรงดันสายสูงเท่ากับแรงดันแหล่งจ่ายไฟตรง ซึ่งวิธี SIN PWM นั้นให้ค่าแรงดันสายสูง 86.66% ของแรงดันแหล่งจ่ายไฟตรง หากต้องการให้ได้แรงดันสายมีค่าสูงเป็น 100% จำเป็นต้องมีการทวีแรงดัน ทำให้ต้องใช้สวิตช์ IGBT ที่ต้องทนแรงดันได้สูงกว่าแบบ SVM 2 เท่า รวมทั้งค่าตัวเก็บประจุในวงจรเรียงกระแสก็ต้องทนแรงดันสูงขึ้นด้วย ทำให้ชุดขับอินเวอร์เตอร์มีราคาแพงขึ้น

ส่วนเทคนิคการสร้างสัญญาณ PWM แบบบวกฮาร์มอนิกที่3 ของสัญญาณซานย์หลักมูล (Third-harmonic PWM) เป็นเทคนิคที่ใช้การบวกสัญญาณฮาร์มอนิกที่3 ของสัญญาณซานย์หลักมูลเป็นอีกเทคนิคหนึ่งที่ใช้กันอย่างกว้างขวาง ซึ่งข้อดีคือสร้างได้ไม่ยุ่งยาก ง่ายต่อการไปประยุกต์ใช้งานทางดิจิตอล และสามารถให้แรงดันสายเอ้าท์พุทมีประสิทธิภาพ 100% ของแรงดันแหล่งจ่ายไฟตรง จึงไม่จำเป็นที่จะต้องอาศัยวงจรทวีแรงดัน โดยรูปสัญญาณแรงดันเฟสจะมีลักษณะคล้ายคลึงกันกับวิธี SVM PWM แต่มีข้อด้อยกว่าแบบ SVM PWM ในเรื่องของการสูญเสียกำลังงานไปการสวิตช์มากกว่าแบบ SVM PWM

การสร้างสัญญาณ SVM PWM สร้างโดยใช้ FPGA (Field Programmable Gate Array) ซึ่งมีความเร็วสูง และมีความยืดหยุ่นในการออกแบบมากกว่า ไมโครคอนโทรลเลอร์ และ DSPs ดังอธิบายในหัวข้อ 1.3 และ 2.8 โดยการสร้างสัญญาณ PWM ของ FPGA นั้นบล๊อกการทำงานที่ออกแบบไว้จะมีลักษณะเป็นการทำงานของวงจรฮาร์ดแวร์ที่มีการโปรแกรมได้ โดยสามารถทำงานได้อิสระพร้อมๆกัน จึงมีความเร็วในการทำงานสูงกว่า ไมโครคอนโทรลเลอร์ และ DSPs โดยการสร้างสัญญาณ PWM นั้นจะอาศัยหลักการเคลื่อนที่ของเวคเตอร์แรงดันเฟส สองตัว รวมกับเวคเตอร์ศูนย์ที่สร้างขึ้น โดยทำการแปลงเวกเตอร์ทั้งสามตัวไปเป็นคาบเวลาในการควบคุมการสวิตช์ของแรงดันเฟสทั้งสามตัวในแบบ 3เฟส โดยเทคนิค SVM มีความเหมาะสมมากที่จะนำไปประยุกต์ใช้กับวงจรควบคุมแบบดิจิตอล อีกทั้งเทคนิค SVM ยังสามารถสร้างสัญญาณ PWM ที่มีรูปแบบการจัดเรียงการสวิตช์ เพื่อลดการสวิตช์ที่ไม่จำเป็นได้อีกด้วย ทำให้สามารถลดผลของการสูญเสียกำลังงานอันเนื่องจากการสวิตช์ได้ ซึ่งเทคนิคการสร้างสัญญาณ PWM แบบอื่นไม่สามารถทำได้

โดยเทคนิค SVM PWM สามารถให้แรงดันสายเอ้าท์พุทมีประสิทธิภาพ 100% ของแรงดันแหล่งจ่ายไฟตรง จึงไม่จำเป็นที่จะต้องอาศัยวงจรทวีแรงดัน แต่ในงานวิจัยนี้มีการต่อวงจรทวีแรงดันเพิ่มเข้าไปทำให้สามารถปรับแรงดันสายที่เอ้าท์พุทได้สูงถึงเท่าตัวของแรงดันอินพุทที่เข้ามา ซึ่งมีประโยชน์มากในการขับมอเตอร์เหนี่ยวนำที่ความถี่สัญญาณไฟสลับสูงกว่าความถี่ใช้งานปกติของมอเตอร์(50Hz) โดยหากต้องการเพิ่มหรือลดความเร็วรอบของมอเตอร์เหนี่ยวนำสามารถทำได้โดยการปรับความถี่ไฟสลับ หากเราทำการปรับความถี่ จะทำให้มีผลกับค่าความต้านทานของขดลวด(X_L :รีแอคแตนซ์) จะเห็นได้ว่าเมื่อเปลี่ยนแปลงความถี่ให้มีค่าสูงขึ้นจะทำให้ค่า X_L มีค่าสูงขึ้นตามความถี่ ซึ่งจะทำให้กระแสที่จ่ายให้กับมอเตอร์ลดลง หากต้องการให้มอเตอร์ทำงานที่พิกัดกำลังเดิมจำเป็นต้องเพิ่มแรงดันให้กับมอเตอร์สูงขึ้น จึงมีความจำเป็นที่จะต้องมีวงจรทวีแรงดันเพื่อเพิ่มแรงดันให้กับมอเตอร์โดยทำการปรับขนาดแรงดันได้โดยการปรับจากค่าดรรชนีการมอดดูเลชั่น โดยปรับจากโปรแกรมในส่วนของ FPGA

วิทยานิพนธ์นี้ได้ออกแบบวงจรอินเวอร์ขับมอเตอร์เหนี่ยวนำกระแสสลับ 3เฟส และ 1เฟส ขนาด 1แรงม้า(750W)โดยใช้โครงสร้างของอินเวอร์เตอร์ 3 เฟสดังรูปที่ 4.3 ซึ่งสามารถควบคุม ค่าดรรชนีการมอดูเลชั่นได้ตั้งแต่ 1-100% ความถี่แรงดันไฟฟ้าสลับของมอเตอร์ได้ตั้งแต่ 20-100Hz , ค่าความถี่การสวิตซ์ได้ตั้งแต่ 1.5KHz จนถึง 1.5MHz และ dead time ได้น้อยสุด 333nS ที่สัญญาณนาฬิกา 6MHz ซึ่งหากทำการปรับเปลี่ยนสัญญาณนาฬิกาให้สูงขึ้นสามารถที่จะเพิ่มความถี่ไฟฟ้าสลับ และความถี่สวิตช์ได้สูงขึ้น โดย FPGA เบอร์ที่ใช้สามารถใช้ได้กับสัญญาณนาฬิกาสูงสุดถึง 120MHz ทั้งนี้การปรับค่าพารามิเตอร์ต่างๆขึ้นอยู่กับ อุปกรณ์และลักษณะการใช้งาน

การพัฒนา ในการนำไปใช้งานในงานที่ต้องมีการปรับเปลี่ยนความเร็วรอบหลายๆค่าเช่น ในระบบเครื่องปรับอากาศแบบอินเวอร์เตอร์ ซึ่งมอเตอร์ก็คือคอมเพรสเซอร์ของเครื่องปรับอากาศ สามารถทำได้โดยการเพิ่มชุดตรวจวัดอุณหภูมิ เพื่อเป็นการป้อนกลับเข้ามาเพื่อปรับความถี่ไฟสลับ

ที่ป้อนให้กับคอมเพรสเซอร์ ซึ่งสามารถทำงานได้โดยไม่จำเป็นต้องมีผู้ควบคุมระบบ หรือหากต้องการปรับความถี่ใช้งานให้สูงกว่าความถี่ไฟสลับที่ทำงานปกติก็สามารถทำได้โดยการเพิ่มชุดตรวจวัดความเร็วรอบเพื่อวัดความเร็วมอเตอร์เหนี่ยวนำ ซึ่งมอเตอร์เหนี่ยวนำที่ใช้มีความเร็วรอบสูงสุดที่ 1500รอบต่อนาทีที่ 50Hz แต่เราสามารถเพิ่มให้สูงกว่าได้โดยการปรับความถี่ไฟสลับจากชุดอินเวอร์เตอร์ และทำการปรับดรรชนีมอดดูเลชั่นเพื่อเพิ่มแรงดันให้กับมอเตอร์เหนี่ยวนำเพื่อรักษาพิกัดกำลังงานไว้ สามารถทำได้โดยการเขียนโปรแกรมเพิ่มเติมให้ปรับค่าดรรนีมอดดูเลชั่นโดยอัติโนมัติ โดยเขียนเป็นลักษณะเก็บค่ามอดดูเลชั่นที่สัมพันธ์กับความเร็วรอบที่ต้องการคงพิกัดกำลังงานไว้เปรียบเทียบกับความเร็วที่เปลี่ยนไป

เอกสารอ้างอิง

[1] J.Rodriquez, E. Wiechmann, I. Holtz "**IGBT Inverter with Vector Modulation**" , IEEE International Symposium on industrial Electronics, p. 131-6,1994.

[2] H. van de Broeck, H.C.Skudelny and G.V.stanke "**Analysis and Realization of a Pulse Width Modulation Based on Voltage Space Vectors**", IEEE Trans. Ind. Appl.vol24,pp 142-150,Jan/Feb 1987.

[3] J.Holtz, "**Pulse Width Modulation-A Survey**" ,Proceedings of the PESC'92,Espana 1992.

[4] P.Yedamale "**Bidirectional VF control for single and 3-phase Induction Motor Using PIC16F72**" , Application note AN967 , Microchip Technology Inc.

[5] S. Fukuda, Y. Iwaji, H. Hasegawa "**PWM Technique for Invertor with Sinusoidal output current**" IEEE Trans., Vol.5, No.1, Nov 1990.

[6] Y. Tzou, H. Hsu "**FPGA Realization of Space-Vector control IC for Three-phase PWM Inverters**" IEEE Trans. On Power Electronics,Vol.12 ,No.6, November 1997.

[7] Y. Tzou, T. Kuo "**Design and implementation of an FPGA-Based Motor Control IC for Permanent Magnet AC Servo Motors**" IEEE Trans. On Industrial Electronic, Vol.38, No.2, April 1996.

[8] T.Wiangtong, W.Sangchai, P.Lumyong "**FPGA-Based IC Design for Inverter with Vector Modulation Technique**", International Symposium on Circuits and System ISCAS,May,2000.

[9] Muhammad H. Rashid. **POWER ELECTRONICS CIRCUIT,DEVICES AND APPLICATION**. Third Editions. : Prentice Hall

[10] T.Wiangtong, P.Dechsuwan, P.Taninsurat "**Universal High Speed Inverter Controller for AC Induction Motors**" , Proceedings of the 28th Electrical Engineering Conference

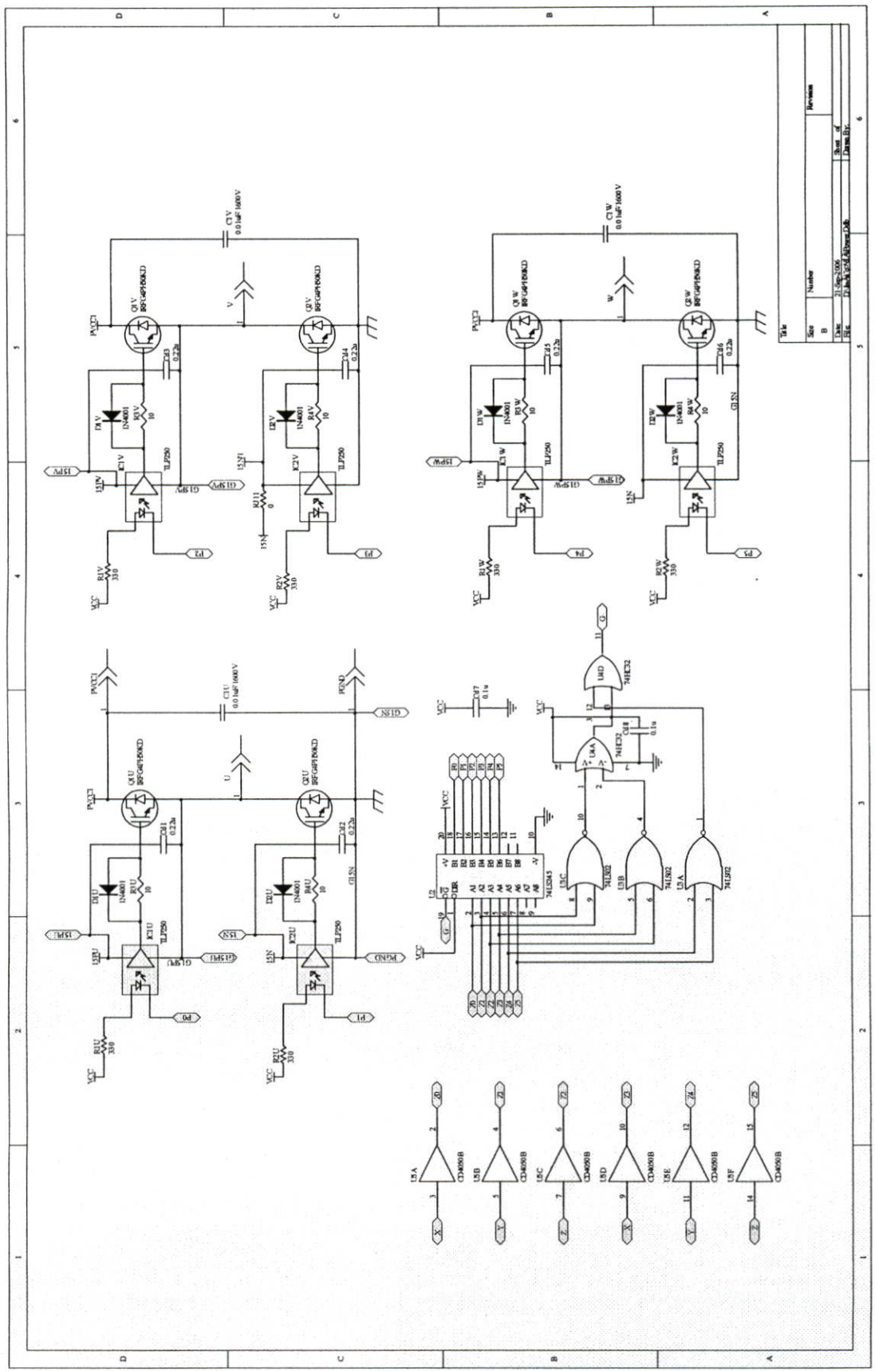
[11] T.Wiangtong, P.Dechsuwan "**Unified Motor Controller Based on Space Vector Modulation Technique**", IEEE International Symposium on Circuits and Systems ISCAS, May, 2006

[12] ไชยชาญ หินเกิด. **เครื่องกลไฟฟ้า2** .กรุงเทพมหานคร: พิมพ์ครั้งที่3 สำนักพิมพ์สมาคมส่งเสริมเทคโนโลยี ไทย- ญี่ปุ่น, พ.ศ.2543

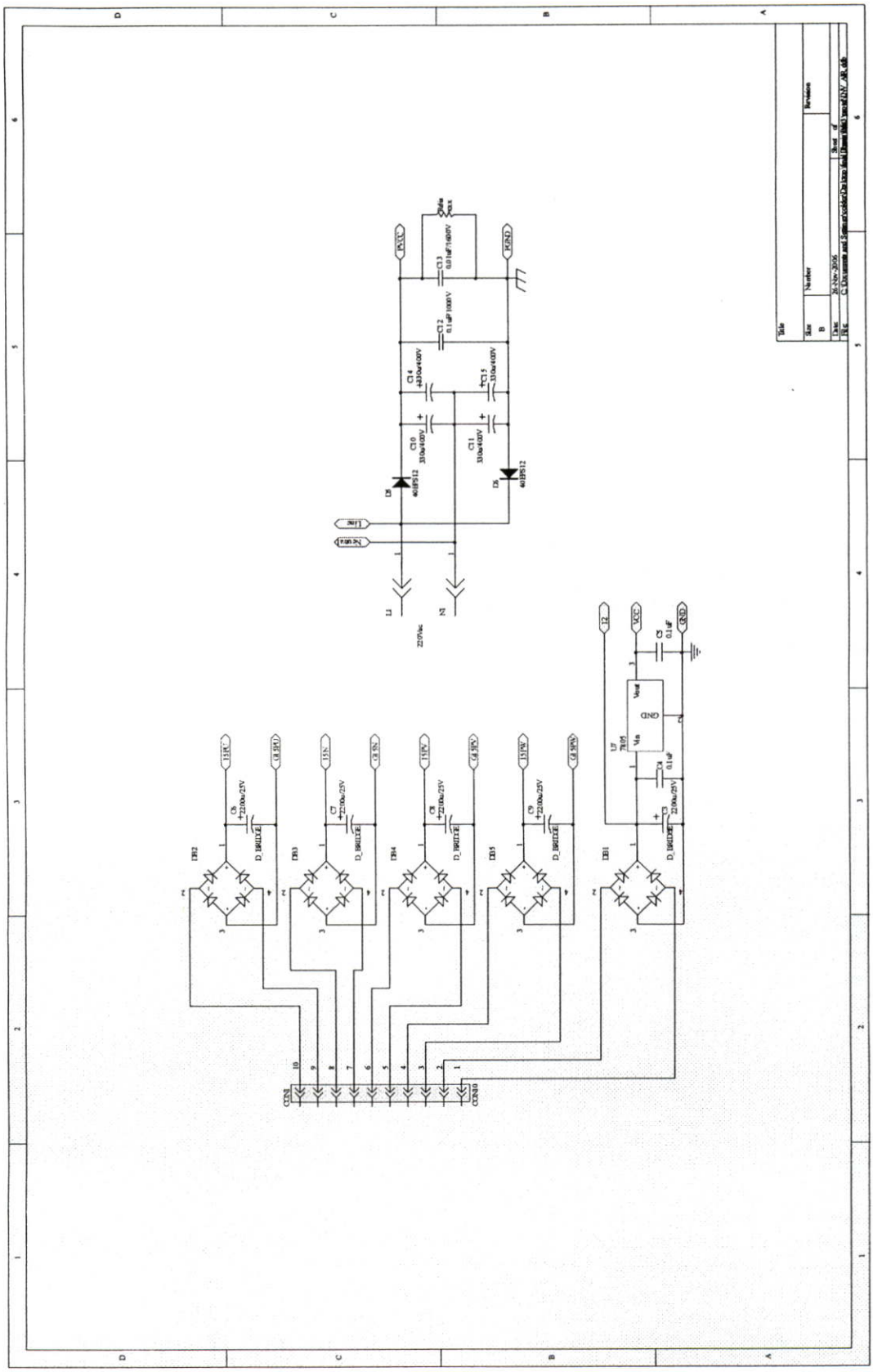
[13] โคทม อารียา. **อิเล็กทรอนิกส์กำลัง 2** .กรุงเทพมหานคร:ซีเอ็ดยูเคชั่น,พ.ศ.2544

[14] D.Rathnakumar, J.LakshmanaPerumal, T.Srinivasan "**A New Software Implementation of Space Vector PWM**" " IEEE Trans., pp 131-136,2005

ภาคผนวก ก.
วงจรสมบูรณ์ของระบบอินเวอร์เตอร์



รูปที่ ก.1 วงจรภาคขับสัญญาณของอินเวอร์เตอร์



รูปที่ ก.2 วงจรภาคแหล่งจ่ายไฟของอินเวอร์เตอร์

ภาคผนวก ข.

โปรแกรมควบคุมระบบ

****โปรแกรมส่วนรับค่า(MCS-51)**********

```
// u3.c
#pragma code
#include <reg51.h>
#include "define.h"
//-------------------------------------
#define KEYMINDEX       0x0a
#define KEYSTEP         0x0b
#define KEYPSHIFT       0x0c
#define KEYT 0x0d
#define KEYDTIME        0x0e
#define KEYENTER        0x0f
#define CURSON          0x0f
#define CURSOFF         0x0c
// Global variable--------------------
uchar       dM3,dM2,dM1,dM0,dT3,dT2,dT1,dT0;
uchar       dS2,dS1,dS0,dP2,dP1,dP0,dD2,dD1,dD0;
uint        dM,dT,dS,dP,dD;
// Constants--------------------------
// String for LCD display -------------
//          "0123456789ABCDEF";
code uchar LINE0[] = "M=     T=     ";
code uchar LINE1[] = "S=   P=   D=   ";
// M(modulation index) 10-1000 ;(1.0-100.0)%
// T(for cal. freq.) 1-1000 ;(3k-3M)Hz
// S(freq. step) 10-500 ;(1.0-50.0)Hz
// P(phase shift) 0-360 degree
// D(dead time) 1-160 ;(0.1-16.0)us
//-------------------------------------
void main(void)
{           uchar k;
            initial();
            while(1)
            {       while((k=scankey())!=0xff)
                    {       if(k==0x0a) { peep1(); fill_m(); }
                            else if(k==0x0b) { peep1(); fill_s(); }
                            else if(k==0x0c) { peep1(); fill_p(); }
                            else if(k==0x0d) { peep1(); fill_t(); }
                            else if(k==0x0e) { peep1(); fill_d(); }
                    }//end while((k=scankey()!=0xff)
            }//end while(1)
}

//-------------------------------------
void initial(void)
{           peep1(); lcdini(); lcdwi(CURSOFF); lcdlc(0x80,LINE0);
lcdlc(0xc0,LINE1);
}
//-------------------------------------
void fill_m(void)           //10-1000 (1.0-100.0)%
{           uchar k,b3,b2,b1,b0;
            uint bint;
            lcdwi(0x82); lcdwd(' '); lcdwd(' '); lcdwd(' '); lcdwd(' ');
lcdwi(0x82); lcdwi(CURSON);
            while((k=scankey())>1); b3=k; shownum(b3); peep1();
            while((k=scankey())>9); b2=k; shownum(b2); peep1();
            while((k=scankey())>9); b1=k; shownum(b1); peep1();
            while((k=scankey())>9); b0=k; shownum(b0); peep1();
            bint=(b3*1000)+(b2*100)+(b1*10)+b0;
            if((bint<=1000)&&(bint>=10)) { dM3=b3; dM2=b2; dM1=b1;
dM0=b0; txdata(0x00,bint); }
            else { lcdwi(0x82); shownum(dM3); shownum(dM2);
shownum(dM1); shownum(dM0); peep2(); }
            lcdwi(CURSOFF);
}
//-------------------------------------
void fill_t(void)           //1-1000 (3k-3M)Hz
{           uchar k,b3,b2,b1,b0;
            uint bint;
            lcdwi(0x8a); lcdwd(' '); lcdwd(' '); lcdwd(' '); lcdwd(' ');
lcdwi(0x8a); lcdwi(CURSON);
            while((k=scankey())>1); b3=k; shownum(b3); peep1();
            while((k=scankey())>9); b2=k; shownum(b2); peep1();
            while((k=scankey())>9); b1=k; shownum(b1); peep1();
            while((k=scankey())>9); b0=k; shownum(b0); peep1();
            bint=(b3*1000)+(b2*100)+(b1*10)+b0;
            if((bint<=1000)&&(bint>=1)) { dT3=b3; dT2=b2; dT1=b1;
dT0=b0; txdata(0x03,bint); }
            else { lcdwi(0x8a); shownum(dT3); shownum(dT2);
shownum(dT1); shownum(dT0); peep2(); }
            lcdwi(CURSOFF);
}
//-------------------------------------
void fill_s(void)           //10-500 (1.0-50.0)Hz
{           uchar k,b2,b1,b0;
            uint bint;
            lcdwi(0xc2); lcdwd(' '); lcdwd(' '); lcdwd(' '); lcdwi(0xc2);
lcdwi(CURSON);
            while((k=scankey())>5); b2=k; shownum(b2); peep1();
            while((k=scankey())>9); b1=k; shownum(b1); peep1();
            while((k=scankey())>9); b0=k; shownum(b0); peep1();
            bint=(b2*100)+(b1*10)+b0;
            if((bint<=500)&&(bint>=10)) { dS2=b2; dS1=b1; dS0=b0;
txdata(0x01,bint); }
            else { lcdwi(0xc2); shownum(dS2); shownum(dS1);
shownum(dS0); peep2(); }
            lcdwi(CURSOFF);
}
//-------------------------------------
void fill_p(void)           //0-360 degree
{           uchar k,b2,b1,b0;
            uint bint;
            lcdwi(0xc7); lcdwd(' '); lcdwd(' '); lcdwd(' '); lcdwi(0xc7);
lcdwi(CURSON);
            while((k=scankey())>3); b2=k; shownum(b2); peep1();
            while((k=scankey())>9); b1=k; shownum(b1); peep1();
            while((k=scankey())>9); b0=k; shownum(b0); peep1();
            bint=(b2*100)+(b1*10)+b0;
            if(bint<=360) { dP2=b2; dP1=b1; dP0=b0; txdata(0x02,bint); }
            else { lcdwi(0xc7); shownum(dP2); shownum(dP1);
shownum(dP0); peep2(); }
            lcdwi(CURSOFF);
}
//-------------------------------------
void fill_d(void)           //1-160 (0.1-16.0)us
{           uchar k,b2,b1,b0;
            uint bint;
            lcdwi(0xcc); lcdwd(' '); lcdwd(' '); lcdwd(' '); lcdwi(0xcc);
lcdwi(CURSON);
            while((k=scankey())>1); b2=k; shownum(b2); peep1();
            while((k=scankey())>9); b1=k; shownum(b1); peep1();
            while((k=scankey())>9); b0=k; shownum(b0); peep1();
```

```
                bint=(b2*100)+(b1*10)+b0;
                if((bint<=160)&&(bint>=1)) { dD2=b2; dD1=b1; dD0=b0;
txdata(0x04,bint); }
                else { lcdwi(0xcc); shownum(dD2); shownum(dD1);
shownum(dD0); peep2(); }
                lcdwi(CURSOFF);
}
//-------------------------------------
void txdata(uchar a,uint d)
{               uchar x;
                intbyte dd;
                dd.intx=d; P1=dd.bytex[1];
                x=dd.bytex[0]; P3=((a<<2)|(x&0xe3));
                ENFPGA=0; delaym(1); ENFPGA=1;
}
//-------------------------------------
```

```
// define.h
//-------------------------------------
typedef unsigned char uchar;
typedef unsigned int uint;
//-------------------------------------
typedef union
{               uint intx;
                uchar bytex[2];
} intbyte;
//-------------------------------------
// port declaration
// P0.0-P0.3 = LCD data(output)
// P0.4-P0.7 Key Column Input
sbit LCDRS = P2^0;          //output
sbit LCDE = P2^1;           //output
// P2.2-P2.3 reserve
sbit ROW0 = P2^7;           //Output to key
sbit ROW1 = P2^6;           //Output to key
sbit ROW2 = P2^5;           //Output to key
sbit ROW3 = P2^4;           //Output to key
// P1.0-P1.7 data low out to FPGA
// P3.0-P3.1 data high out to FPGA
sbit ADDR0 = P3^2;          //output to FPGA select Address
sbit ADDR1 = P3^3;          //output to FPGA select Address
sbit ADDR2 = P3^4;          //output to FPGA select Address
sbit ENFPGA = P3^5;         //output to FPGA Enable
sbit RPM = P3^6;            //input
sbit BUZ = P3^7;            //output
//-------------------------------------
// delay.c
void delayu(uint count);
void delaym(uint count);
//void delays(uint count);
void peep(void);
void peep1(void);
void peep2(void);
//-------------------------------------

// key.c
uchar scankey(void);
//-------------------------------------
```

```
// lcd.c
void lcdwi(uchar ins);
void lcdwd(uchar dat);
void lcdlc(uchar addr,uchar *msptr);
void lcdini(void);
void shownum(uchar y);
//void showbcd(uchar y);
//-------------------------------------
// u3.c
void initial(void);
void fill_m(void);
void fill_t(void);
void fill_s(void);
void fill_p(void);
void fill_d(void);
void txdata(uchar a,uint d);
//-------------------------------------
```

```
//-------------------------delay-part---------------
// delay.c
#include <reg51.h>
#include "define.h"
//for 89C51 @12MHz,12clk
//-------------------------------------
void delayu(uint count)
{               uchar x,y,z;
                x=count/240; y=(count%240)/2;
                while(x) { for(z=120;z>0;z--); }
                for(z=y;z>0;z--);
}
//-------------------------------------
void delaym(uint count)
{               uint x;
                uchar y;
                for(x=count;x>0;x--)
                {               for(y=250;y>0;y--); for(y=250;y>0;y--); }
}
//-------------------------------------
//void delays(uint count)
//{             uint x;
//              for(x=count;x>0;x--) { delaym(1000); }
//}
//-------------------------------------
void peep(void)             //@2.5kHz
{               uint x;
                for(x=0;x<400;x++) { BUZ=~BUZ; delayu(200); }
}
//-------------------------------------
void peep1(void)
{               peep(); delaym(200);
}
//-------------------------------------
void peep2(void)
{               peep(); delaym(100); peep();
}
//-------------------------------------
```

```
// lcd.c
#include <reg51.h>
#include <string.h>
#include "define.h"
//-------------------------------------
void lcdwi(uchar ins)
{       P0=(ins>>4)|(0xf0);
        LCDRS=0; LCDE=1; LCDE=0; delaym(2);  // lcd instruction
        P0=(ins)|(0xf0);
        LCDRS=0; LCDE=1; LCDE=0; delaym(2); // lcd instruction
}
//-------------------------------------
void lcdwd(uchar dat)
{       P0=(dat>>4)|(0xf0);
        LCDRS=1; LCDE=1; LCDE=0; delaym(2); // lcd data
        P0=(dat)|(0xf0);
        LCDRS=1; LCDE=1; LCDE=0; delaym(2); // lcd data
}
//-------------------------------------
void lcdlc(uchar addr,uchar *msptr)
{       uchar len;
        len=strlen(msptr); lcdwi(addr);
        while(len--) { lcdwd(*msptr); msptr++; }
}
//-------------------------------------
void lcdini(void)
{       LCDE = 0;
        lcdwi(0x33);   /* 8 bits */
        lcdwi(0x32);   /* 4 bits */
        lcdwi(0x28);   /* function set */
        lcdwi(0x06);   /* entry mode set */
        lcdwi(0x0f);   /* display on/off */
        lcdwi(0x01);   /* clear display */
        delaym(10);
}
//-------------------------------------
void shownum(uchar y)
{       uchar x;
        if(y<=0x09) x=y+'0'; else x=(y-0x0a+'A');
        lcdwd(x);
}
//-------------------------------------
//void showbcd(uchar y)
//{       uchar x;
//      x=y>>4; x=x&0x0f; shownum(x);
//      x=y&0x0f; shownum(x);
//}
//-------------------------------------
```

```
// key.c
#include <reg51.h>
#include "define.h"
//-------------------------------
// return 0xff if has no key press
uchar scankey(void)
{       uchar col;
        P2=0x7f; col=P0&0xf0; //row 0
        if(col != 0xf0)
                switch(col)
                {       case 0x70: return(0x01); break; //
r0 c0
                        case 0xb0: return(0x02); break; //
r0 c1
                        case 0xd0: return(0x03); break; //
r0 c2
                        case 0xe0: return(0x0a); break; //
r0 c3
                };
        P2=0xbf; col=P0&0xf0; //row 1
        if(col != 0xf0)
                switch(col)
                {       case 0x70: return(0x04); break; //
r1 c0
                        case 0xb0: return(0x05); break; //
r1 c1
                        case 0xd0: return(0x06); break; //
r1 c2
                        case 0xe0: return(0x0b); break; //
r1 c3
                };
        P2=0xdf; col=P0&0xf0; //row 2
        if(col != 0xf0)
                switch(col)
                {       case 0x70: return(0x07); break; //
r2 c0
                        case 0xb0: return(0x08); break; //
r2 c1
                        case 0xd0: return(0x09); break; //
r2 c2
                        case 0xe0: return(0x0c); break; //
r2 c3
                };
        P2=0xef; col=P0&0xf0; //row 3
        if(col != 0xf0)
                switch(col)
                {       case 0x70: return(0x0f); break; //
r3 c0
                        case 0xb0: return(0x00); break; //
r3 c1
                        case 0xd0: return(0x0e); break; //
r3 c2
                        case 0xe0: return(0x0d); break; //
r3 c3
                };
        return(0xff);   // no key press
}
//-------------------------------
```

```
********โปรแกรมส่วนของ FPGA**********
// Author
// Version 2
// Created 22-Jun-2005
//
// Copyright (c) 2005, QuIC.  All Rights Reserved.
// There is no warranty with respect to
// the operation and/or functionality of the design files.
// No use for commercial purpose without authorisation/permission from the
auther!

// Descriptions
================================================
// Top-level module without LCD user interface
// New system_vp2.v

module igbt_noLCD(clk ,reset, lag, a, b, c, x, y, z, dbg);

        input clk, lag, reset;
        //PWM output
        output [1:0] a,b,c;          //3phase inverter
        output [1:0] x,y,z;          //1phase inverter

        //debugging ports
        output [3:0] dbg;

        //system_vp interface
        wire         [7:0] dtime;
        wire         [8:0] step;
        wire         [8:0] pshift;
        wire         [9:0] T, m_index;
        wire [3:0] dbg;

        wire start;
        assign start = reset ;

        //clock divider
        wire clksrc, clkdiv2;
        wire clkdiv1024, blink;
        defparam div2.n = 1;
        div2e_n div2(.clkin(clk), .clkout(clkdiv2));
        defparam div1024.n = 10;
        div2e_n div1024(.clkin(clksrc), .clkout(clkdiv1024));
        defparam div1024x.n = 10;
        div2e_n div1024x(.clkin(clkdiv1024), .clkout(blink));

        //for probing with external logic analyser
        assign dbg[0] = a[1]; //ref pwm
        assign dbg[1] = z[1]; //shifted pwm
        assign dbg[2] = ~reset;
        assign dbg[3] = blink;


        /*
        //input and display control module
        input_ctrl InputControl(
        .clk(clkdiv1024),
        .row(row),
        .col(col),
        .rw(rw),
        .en(en),
        .rs(rs),
        .db(db),
        .enter(enter),
        .dtime(dtime),
        .step(step),
        .pshift(pshift),
        .T(T),
        .m_index(m_index)
        );
        */

        assign clksrc = clkdiv2;        //use 'clk' as sampling clk

        assign m_index = 1000;  //0-1000
        assign step = 3;                //0-500
        assign dtime = 6;               //0-100 * 1/fclk
        assign pshift = 96;  //0-360 degree
        assign T = 50;   //0-1000

        defparam System.sm1.VecIni = 3'b000;      //0
        defparam System.sm2.VecIni = 3'b000;      //pshift
        defparam System.sm3.VecIni = 3'b111;      //180

        //Useful Formula
================================================
        //Fc = Operating Frequency
        //Fs = Sampling Frequency
        //Fm = Motor Frequency (Speed)
        //
        //Fs = 1/Ts = 1/(2T*Tc) = Fc/(2T)
        //Fm = 1/Tm = 1/(512/step * 6 * Ts/2) = (step)/(1536*Ts) =
(Fs*step)/1536
        //Deadtime = dtime * 1/Fc
        //=========================================
==========================

        //core module
        system_vp2 System(
        .clk(clksrc),
        .dt_clk(clksrc),
        .cal_clk(clksrc),
        .reset(reset),
        .dtime(dtime),
        .m_index(m_index),
        .step(step),   //0..511
        .pshift(pshift),  //0..360 degree
        .lag(lag),    // phase shift lead => lag = '0' : lag => lag = '1'
        .T(T),
        .a(a),  .b(b),  .c(c),          //for 3 phase induction motor
        .x(x),  .y(y),  .z(z)           //for 1 phase induction motor
        );

        endmodule
```

```
// Author
// Version 1
// Created 09-Jun-2005
//
// Copyright (c) 2005, QuIC.  All Rights Reserved.
// There is no warranty with respect to
// the operation and/or functionality of the design files.
// No use for commercial purpose without authorisation/permission from the
auther!

// Descriptions
================================================

// Divided by 2^n
// n is parameterisable

module div2e_n(clkin, clkout);
   input clkin;
   output clkout;

parameter n = 8;

reg [n-1:0] divff;
always @ (posedge clkin)
                divff <= divff + 1;

assign clkout = divff[n-1];

endmodule
```

```
// Author
// Version 2
// Created 22-Jun-2005
//
// Copyright (c) 2005, QuIC.  All Rights Reserved.
// There is no warranty with respect to
// the operation and/or functionality of the design files.
// No use for commercial purpose without authorisation/permission from the
auther!

// Descriptions
================================================

// PWM for 1-phase and 3-phase inverter
// This system can vary phase from 0 - 360 degree with 1 degree resolution
// However, we can increase the resultion to 0.2 degree.
// Outputs are  for both 3-phase IGBT (abc) also 1-phase inverter (xyz)
// Differences:
// Change Phasc10n (36states 112slices) to Vector_gen (4 state 50 slices!)
// Can be added for 3 inverters simultaneously
// NEW---------
// New cnt_address with loop_num3 for 180 degree

module system_vp2(
                clk,
                dt_clk,
                cal_clk,
                reset,
                dtime,
                m_index,
                step,   //0..511
                pshift, //0..360 degree
                lag,    // phase shift lead => lag = '0' : lag => lag = '1'
                T,
                a, b, c,        //for 3 phase induction motor
                x, y, z         //for 1 phase induction motor

);

input           clk;
input           dt_clk;
input           cal_clk;
input           reset;
input           [7:0] dtime;
input           [8:0] step;
input           [8:0] pshift;
input           lag;
input           [9:0] T, m_index;
output  [1:0] a, b, c;
output  [1:0] x, y, z;

wire            check_wire1, check_wire2;
wire cal_wire;
wire [2:0] mod_wire;
wire [8:0] shift;
wire            [2:0] pwm_ref_wire, pwm_lag_wire, pwm_180_wire;
wire            [2:0] sector1, sector2, sector3;
wire            [9:0] ta1_wire,tb1_wire,to1_wire;
wire            [9:0] ta2_wire,tb2_wire,to2_wire;
wire            [9:0] MemA1, MemB1;
wire            [9:0] MemA2, MemB2;
```

```
wire [2:0] st_pc1, st_pc2;
wire [2:0] st_cal1, st_cal2;
wire [10:1] cnt1, cnt2, tmon1, tmon2;
wire [1:0] delta1, delta2;

//find sector and shifted degree of  shifted PWM
mod60 modulus_60(.indeg(pshift), .lag(lag), .mod(mod_wire), .shift(shift));

//access to the vlaues of vector stored in ROM
cnt_adr_vp2 mem_access( .clk(clk), .reset(reset), .sector_ini(mod_wire),
.step(step), .pshift(shift),
                .start(check_wire1 | check_wire2), .A1(MemA1), .B1(MemB1),
.A2(MemA2), .B2(MemB2),
                .ready(cal_wire), .loop_num1(sector1), .loop_num2(sector2),
.loop_num3(sector3));

//calculate ta, tb, tx of reference PWM
cal_time
                cal_time1(.clk(cal_clk),.cal(cal_wire),.m(m_index),.mema(MemA
1),.memb(MemB1),
                .t(T),.ta(ta1_wire),.tb(tb1_wire),.tx(to1_wire), .state(st_cal1));

//calculate ta, tb, tx of shifted PWM
cal_time
                cal_time2(.clk(cal_clk),.cal(cal_wire),.m(m_index),.mema(MemA
2),.memb(MemB2),
                .t(T),.ta(ta2_wire),.tb(tb2_wire),.tx(to2_wire), .state(st_cal2));

//generate a new vector of reference PWM regarding to ta, tb, tx
//defparam sm1.VecIni = 3'b000;
vector_gen      sm1(.clk(clk),.reset(reset),.sector(sector1),
                .ta(ta1_wire),.tb(tb1_wire),.tx(to1_wire),
                .check(check_wire1),.abc(pwm_ref_wire), .state(st_pc1),
.cnt(cnt1), .tmon(tmon1));

//generate a new vector of shifted PWM regarding to ta, tb, tx
//defparam sm2.VecIni = 3'b000;
vector_gen      sm2(.clk(clk),.reset(reset),.sector(sector2),
                .ta(ta2_wire),.tb(tb2_wire),.tx(to2_wire),
                .check(check_wire2),.abc(pwm_lag_wire), .state(st_pc2),
.cnt(cnt2), .tmon(tmon2));

//generate a new vector of 180 PWM regarding to ta, tb, tx
//defparam sm3.VecIni = 3'b111;
vector_gen      sm3(.clk(clk),.reset(reset),.sector(sector3),
                .ta(ta1_wire),.tb(tb1_wire),.tx(to1_wire),
                .check(),.abc(pwm_180_wire), .state(), .cnt(), .tmon());

//insert deadtime for 3 phase inverter
defparam three_phase.e = 1'b0; //falling edge deferring (for active low driver)
dtime6x         three_phase(.clk(dt_clk),
.ina1(~pwm_ref_wire[2]),.ina2(pwm_ref_wire[2]),  // 0 degree
.inb1(~pwm_ref_wire[1]),.inb2(pwm_ref_wire[1]),  // 120 degree
.inc1(~pwm_ref_wire[0]),.inc2(pwm_ref_wire[0]),  // 240 degree
.dtime(dtime),
.outa1(a[1]),.outa2(a[0]),
.outb1(b[1]),.outb2(b[0]),
.outc1(c[1]),.outc2(c[0]));

//insert deadtime for 1 phase inverter          (can be added for 3 inverters
simultaneously)
defparam one_phase.e = 1'b0; //falling edge deferring (for active low driver)
dtime6x         one_phase(.clk(dt_clk),
.ina1(~pwm_ref_wire[2]),.ina2(pwm_ref_wire[2]), // 0 degree (ref PWM)
.inb1(~pwm_lag_wire[2]),.inb2(pwm_lag_wire[2]), // pshift degree
.inc1(~pwm_180_wire[2]),.inc2(pwm_180_wire[2]), // 180 degree
.dtime(dtime),
.outa1(x[1]),.outa2(x[0]),
.outb1(y[1]),.outb2(y[0]),
.outc1(z[1]),.outc2(z[0]));

assign delta1 = x[1]-z[1];
assign delta2 = y[1]-z[1];

endmodule
```

```
// Author
// Version 1
// Created 09-Jun-2005
//
// Copyright (c) 2005, QuIC.  All Rights Reserved.
// There is no warranty with respect to
// the operation and/or functionality of the design files.
// No use for commercial purpose without authorisation/permission from the
auther!

// Descriptions
================================================

// Calculate Ta, Tb and Tx for ref PWM and shift PWM independently
// Shared multiplier to cal Ta, Tb and Tx respectively
// Make non-zero value for ta, tb to prevent missing VecOld stored at rising edge
of cnt_rst in vector_gen.v


module cal_time(clk, cal, m, t, mema, memb, ta, tb, tx, state);
  input clk, cal;
  input [9:0] m;
  input [9:0] t;
  input [9:0] mema;
  input [9:0] memb;
  output [9:0] ta;
  output [9:0] tb;
  output [9:0] tx;
  output [2:0] state;

  wire [19:0] mult_result;
  reg [9:0] mult_dataa;
  reg [9:0] mult_datab;
  reg [9:0] aff;
  reg [9:0] bff;
  reg [9:0] mtff;
  reg [2:0] ss_d;
  reg [2:0] ss_q;
  reg [9:0] a;
  reg [9:0] b;
  reg [9:0] mt;
  reg clka, clkb, clkmt, ldx;

  always @ (posedge clkmt)
   mtff <= mt;

  always @ (posedge clka)
  if (a==0)
            aff <= 1; //to prevent missing VecOld stored at rising edge of
cnt_rst in vector_gen.v
  else
   aff <= a;

  always @ (posedge clkb)
  if (b==0)
   bff <= 1; //to prevent missing VecOld stored at rising edge of cnt_rst in
vector_gen.v
  else
   bff <= b;

  always @ (posedge clk)
   ss_q <= ss_d;

// Sub Module Section
  cal_time_lpm_mult0  mult (.dataa(mult_dataa), .datab(mult_datab),
            .result(mult_result));

  always @(ss_q or m or t or mema or mtff or memb or mult_result
  or clk or cal) begin
            {clkmt, clka, clkb} = 3'b000;
            ldx = 1'b0;
   casex (ss_q)
   3'b000: begin
             if (cal) begin
               ss_d = 3'b001;
             end else begin
               ss_d = 3'b000;
             end
           end
   3'b001: begin
             mult_dataa <= m;
             mult_datab <= t;
             mt <= mult_result[19:10];
             ss_d = 3'b010;
             clkmt = !clk;
           end
   3'b010: begin
             mult_dataa <= mtff;
             mult_datab <= mema;
             a <= mult_result[19:10];
             ss_d = 3'b011;
             clka = !clk;
           end
   3'b011: begin
             mult_dataa <= mtff;
             mult_datab <= memb;
             b <= mult_result[19:10];
             ss_d = 3'b100;
             clkb = !clk;
           end
   3'b100: begin
             if (!cal) begin
               ss_d = 3'b000;
             end
           end
    endcase
  end

   assign ta = aff;
   assign tb = bff;
   assign tx= t - (aff + bff);

  assign state = ss_q;

endmodule


module cal_time(clk, cal, m, t, mema, memb, ta, tb, tx, state);
  input clk, cal;
  input [9:0] m;
  input [9:0] t;
  input [9:0] mema;
```

```
input [9:0] memb;
output [9:0] ta;
output [9:0] tb;
output [9:0] tx;
output [2:0] state;

wire [19:0] mult_result;
reg [9:0] mult_dataa;
reg [9:0] mult_datab;
reg [9:0] aff;
reg [9:0] bff;
reg [9:0] mtff;
reg [2:0] ss_d;
reg [2:0] ss_q;
reg [9:0] a;
reg [9:0] b;
reg [9:0] mt;
reg clka, clkb, clkmt, ldx;

always @ (posedge clkmt)
 mtff <= mt;

always @ (posedge clka)
if (a==0)
        aff <= 1; //to prevent missing VecOld stored at rising edge of
cnt_rst in vector_gen.v
else
 aff <= a;

always @ (posedge clkb)
if (b==0)
 bff <= 1; //to prevent missing VecOld stored at rising edge of cnt_rst in
vector_gen.v
else
 bff <= b;

always @ (posedge clk)
 ss_q <= ss_d;

// Sub Module Section
cal_time_lpm_mult0  mult (.dataa(mult_dataa), .datab(mult_datab),
        .result(mult_result));

always @(ss_q or m or t or mema or mtff or memb or mult_result
or clk or cal) begin
        {clkmt, clka, clkb} = 3'b000;
        ldx = 1'b0;
  casex (ss_q)
  3'b000: begin
          if (cal) begin
            ss_d = 3'b001;
          end else begin
            ss_d = 3'b000;
          end
        end
  3'b001: begin
          mult_dataa <= m;
          mult_datab <= t;
          mt <= mult_result[19:10];
          ss_d = 3'b010;
          clkmt = !clk;
        end
  3'b010: begin
          mult_dataa <= mtff;
          mult_datab <= mema;
          a <= mult_result[19:10];
          ss_d = 3'b011;
          clka = !clk;
        end
  3'b011: begin
          mult_dataa <= mtff;
          mult_datab <= memb;
          b <= mult_result[19:10];
          ss_d = 3'b100;
          clkb = !clk;
        end
  3'b100: begin
          if (!cal) begin
            ss_d = 3'b000;
          end
        end
  endcase
end

assign ta = aff;
assign tb = bff;
assign tx= t - (aff + bff);

assign state = ss_q;

endmodule
```

```
// Author
// Version 1
// Created 28-Apr-2005
//
// Copyright (c) 2005, QuIC.  All Rights Reserved.
// There is no warranty with respect to
// the operation and/or functionality of the design files.
// No use for commercial purpose without authorisation/permission from the
auther!

// Descriptions
================================================

// Calculate Time Ta, Tb, Tx
// Referenced from module: cal_time


module cal_time_lpm_mult0(dataa, datab, result);
  input [9:0] dataa;
  input [9:0] datab;
  output [19:0] result;

  wire [19:0] res;


// Start of original equations
  assign res = dataa * datab;
  assign result = res;
endmodule
```

```
// Author
// Version 2
// Created 22-Jun-2005
//
// Copyright (c) 2005, QuIC.  All Rights Reserved.
// There is no warranty with respect to
// the operation and/or functionality of the design files.
// No use for commercial purpose without authorisation/permission from the
auther!

// Descriptions
================================================


// Acess data in ROMs for ref PWM and shift PWM independently
// Use 2 ROMS synchronise with posedge of clk signal
// Multiplex adrr to access memory
// Modified to be used in system_vp1.v
// that has initial values of "sector" and "phase shift" loaded when 'reset'
// NEW----------
// Access mem for 0, pshift, 180 (loop_num3)

module cnt_adr_vp2( clk, reset, sector_ini, step, pshift, start, A1, B1, A2, B2,
ready, loop_num1, loop_num2, loop_num3);
  input clk, reset;
  input [2:0] sector_ini;
  input [8:0] step, pshift;
  input start;
  output ready;
  output [9:0] A1, B1, A2, B2;
  output [2:0] loop_num1, loop_num2, loop_num3;

  reg ld1, ld2; //for loading rom outputs of two diff addresses (ref signal and
shifted signal)
  reg ready;
  wire lp_cnt1, lp_cnt2;
  reg [2:0] loop_num1, loop_num2, loop_num3;
  reg [1:0] ss__d;
  reg [1:0] ss__q;
  reg [8:0] adr;
  reg [8:0] adr1;
  reg [8:0] adr2;
  reg [9:0] A1, B1, A2, B2;
  wire [9:0] A, B;

  assign state = ss__q;

            always @(posedge clk)
            if (reset)
            ss__q <= 0;
            else
            ss__q <= ss__d;

  always @(ss__q or start or adr or adr1 or adr2 or step or pshift)
            begin
            ld1 <= 1'b0;
            ld2 <= 1'b0;
            ready <= 1'b0;
  case (ss__q)
  2'b00: begin
              ready <= 1'b0;
              if (start) begin
```

```
        ss_d <= 2'b01;
      end else begin
        ss_d <= 2'b00;
      end
    end
    2'b01: begin
        ld1 <= 1'b1;
                ready <= 1'b0;
                ss_d <= 2'b10;
    end
    2'b10: begin
                ld2 <= 1'b1;
                ready <= 1'b0;
                ss_d <= 2'b11;
    end
    2'b11: begin
                ready <= 1'b1;
                if (!start)
        ss_d <= 2'b00;
                else
                ss_d <= 2'b11;
    end
    endcase
    end


    always @ (posedge start  or posedge reset)
    begin
      if (reset)  //load ini value
      begin adr1 <= 0;
          adr2 <= pshift;
      end
      else          //regular count
      begin
                adr1 <= adr1 + step;
                adr2 <= adr2 + step;
      end
    end


//mux adr1 and adr2 to roms' address
    always @ ( ld1 or  ld2)
    begin
      if (ld1) adr <= adr1;
      else if (ld2) adr <= adr2;
    end

    //read data from rom at falling edge of clk
    syn_rom_0     rom0(.clk(~clk), .address(adr),.q(A));
    syn_rom_1     rom1(.clk(~clk), .address(adr),.q(B));
    //rom0          rom0(.addr(adr), .clk(clk), .dout(A));
    //rom1          rom1(.addr(adr), .clk(clk), .dout(B));

    //load read data at falling edge of ld1 and ld2
    always @ (negedge ld1)
    begin
                A1 <= A;
                B1 <= B;        end
    always @ (negedge ld2)
    begin
                A2 <= A;
                B2 <= B;        end


    assign lp_cnt1 = ~adr1[8];
    assign lp_cnt2 = ~adr2[8];

    always @ (posedge lp_cnt1 or posedge reset )
    begin
                if ( reset) loop_num1 <= 0;    //reference PWM
starts with sector_ini 0
                else
                 if (loop_num1 < 5)
                        loop_num1 <= loop_num1 + 1;
                 else
                        loop_num1 <= 0;
    end

    always @ (posedge lp_cnt2 or posedge reset )
    begin
                if (reset) loop_num2 <= sector_ini;  //shifted
PWM starts with 'sector_ini'
                else
                 if (loop_num2 < 5)
                        loop_num2 <= loop_num2 + 1;
                 else
                        loop_num2 <= 0;
    end

    always @ (posedge lp_cnt1 or posedge reset )
    begin
                if ( reset) loop_num3 <= 3;    //180 PWM starts
with sector_ini 3
                else
                 if (loop_num3 < 5)
                        loop_num3 <= loop_num3 + 1;
                 else
                        loop_num3 <= 0;
    end

endmodule
```

```
// Author
// Version 1
// Created 28-Apr-2005
//
// Copyright (c) 2005, QuIC.  All Rights Reserved.
// There is no warranty with respect to
// the operation and/or functionality of the design files.
// No use for commercial purpose without authorisation/permission from the
auther!

// Descriptions
================================================
// Attention! use negedge of clk

module syn_rom_0(clk,address,q);
input clk;
input [8:0] address;
output [9:0] q;
reg [9:0] q;


always @(posedge clk)
begin
   case (address)
0       :       q       =       766     ;
1       :       q       =       765     ;
2       :       q       =       764     ;
3       :       q       =       763     ;
4       :       q       =       762     ;
5       :       q       =       761     ;
6       :       q       =       760     ;
7       :       q       =       759     ;
8       :       q       =       758     ;
9       :       q       =       758     ;
10      :       q       =       757     ;
11      :       q       =       756     ;
12      :       q       =       755     ;
13      :       q       =       754     ;
14      :       q       =       753     ;
15      :       q       =       752     ;
16      :       q       =       751     ;
17      :       q       =       750     ;
18      :       q       =       749     ;
19      :       q       =       748     ;
20      :       q       =       747     ;
21      :       q       =       746     ;
22      :       q       =       745     ;
23      :       q       =       744     ;
24      :       q       =       743     ;
25      :       q       =       742     ;
26      :       q       =       741     ;
27      :       q       =       740     ;
28      :       q       =       739     ;
29      :       q       =       738     ;
30      :       q       =       737     ;
31      :       q       =       736     ;
32      :       q       =       735     ;
33      :       q       =       734     ;
34      :       q       =       733     ;
35      :       q       =       732     ;
36      :       q       =       731     ;
37      :       q       =       730     ;
38      :       q       =       729     ;
39      :       q       =       728     ;
40      :       q       =       727     ;
41      :       q       =       726     ;
42      :       q       =       725     ;
43      :       q       =       724     ;
44      :       q       =       723     ;
45      :       q       =       722     ;
46      :       q       =       721     ;
47      :       q       =       720     ;
48      :       q       =       719     ;
49      :       q       =       718     ;
50      :       q       =       716     ;
51      :       q       =       715     ;
52      :       q       =       714     ;
53      :       q       =       713     ;
54      :       q       =       712     ;
55      :       q       =       711     ;
56      :       q       =       710     ;
57      :       q       =       709     ;
58      :       q       =       708     ;
59      :       q       =       707     ;
60      :       q       =       706     ;
61      :       q       =       705     ;
62      :       q       =       703     ;
63      :       q       =       702     ;
64      :       q       =       701     ;
65      :       q       =       700     ;
66      :       q       =       699     ;
67      :       q       =       698     ;
68      :       q       =       697     ;
69      :       q       =       696     ;
70      :       q       =       695     ;
71      :       q       =       693     ;
72      :       q       =       692     ;
73      :       q       =       691     ;
74      :       q       =       690     ;
75      :       q       =       689     ;
76      :       q       =       688     ;
77      :       q       =       687     ;
78      :       q       =       685     ;
79      :       q       =       684     ;
80      :       q       =       683     ;
81      :       q       =       682     ;
82      :       q       =       681     ;
83      :       q       =       680     ;
84      :       q       =       679     ;
85      :       q       =       677     ;
86      :       q       =       676     ;
87      :       q       =       675     ;
88      :       q       =       674     ;
89      :       q       =       673     ;
90      :       q       =       672     ;
91      :       q       =       670     ;
92      :       q       =       669     ;
93      :       q       =       668     ;
94      :       q       =       667     ;
95      :       q       =       666     ;
96      :       q       =       664     ;
97      :       q       =       663     ;
```

98	:	q	=	662	;	159	:	q	=	584	;
99	:	q	=	661	;	160	:	q	=	582	;
100	:	q	=	660	;	161	:	q	=	581	;
101	:	q	=	658	;	162	:	q	=	580	;
102	:	q	=	657	;	163	:	q	=	578	;
103	:	q	=	656	;	164	:	q	=	577	;
104	:	q	=	655	;	165	:	q	=	575	;
105	:	q	=	654	;	166	:	q	=	574	;
106	:	q	=	652	;	167	:	q	=	573	;
107	:	q	=	651	;	168	:	q	=	571	;
108	:	q	=	650	;	169	:	q	=	570	;
109	:	q	=	649	;	170	:	q	=	569	;
110	:	q	=	647	;	171	:	q	=	567	;
111	:	q	=	646	;	172	:	q	=	566	;
112	:	q	=	645	;	173	:	q	=	564	;
113	:	q	=	644	;	174	:	q	=	563	;
114	:	q	=	642	;	175	:	q	=	562	;
115	:	q	=	641	;	176	:	q	=	560	;
116	:	q	=	640	;	177	:	q	=	559	;
117	:	q	=	639	;	178	:	q	=	557	;
118	:	q	=	637	;	179	:	q	=	556	;
119	:	q	=	636	;	180	:	q	=	555	;
120	:	q	=	635	;	181	:	q	=	553	;
121	:	q	=	634	;	182	:	q	=	552	;
122	:	q	=	632	;	183	:	q	=	550	;
123	:	q	=	631	;	184	:	q	=	549	;
124	:	q	=	630	;	185	:	q	=	547	;
125	:	q	=	629	;	186	:	q	=	546	;
126	:	q	=	627	;	187	:	q	=	545	;
127	:	q	=	626	;	188	:	q	=	543	;
128	:	q	=	625	;	189	:	q	=	542	;
129	:	q	=	623	;	190	:	q	=	540	;
130	:	q	=	622	;	191	:	q	=	539	;
131	:	q	=	621	;	192	:	q	=	537	;
132	:	q	=	620	;	193	:	q	=	536	;
133	:	q	=	618	;	194	:	q	=	535	;
134	:	q	=	617	;	195	:	q	=	533	;
135	:	q	=	616	;	196	:	q	=	532	;
136	:	q	=	614	;	197	:	q	=	530	;
137	:	q	=	613	;	198	:	q	=	529	;
138	:	q	=	612	;	199	:	q	=	527	;
139	:	q	=	610	;	200	:	q	=	526	;
140	:	q	=	609	;	201	:	q	=	524	;
141	:	q	=	608	;	202	:	q	=	523	;
142	:	q	=	606	;	203	:	q	=	521	;
143	:	q	=	605	;	204	:	q	=	520	;
144	:	q	=	604	;	205	:	q	=	518	;
145	:	q	=	602	;	206	:	q	=	517	;
146	:	q	=	601	;	207	:	q	=	516	;
147	:	q	=	600	;	208	:	q	=	514	;
148	:	q	=	598	;	209	:	q	=	513	;
149	:	q	=	597	;	210	:	q	=	511	;
150	:	q	=	596	;	211	:	q	=	510	;
151	:	q	=	594	;	212	:	q	=	508	;
152	:	q	=	593	;	213	:	q	=	507	;
153	:	q	=	592	;	214	:	q	=	505	;
154	:	q	=	590	;	215	:	q	=	504	;
155	:	q	=	589	;	216	:	q	=	502	;
156	:	q	=	588	;	217	:	q	=	501	;
157	:	q	=	586	;	218	:	q	=	499	;
158	:	q	=	585	;	219	:	q	=	498	;

220 : q = 496 ; 281 : q = 401 ;
221 : q = 495 ; 282 : q = 399 ;
222 : q = 493 ; 283 : q = 398 ;
223 : q = 492 ; 284 : q = 396 ;
224 : q = 490 ; 285 : q = 395 ;
225 : q = 489 ; 286 : q = 393 ;
226 : q = 487 ; 287 : q = 391 ;
227 : q = 486 ; 288 : q = 390 ;
228 : q = 484 ; 289 : q = 388 ;
229 : q = 483 ; 290 : q = 386 ;
230 : q = 481 ; 291 : q = 385 ;
231 : q = 480 ; 292 : q = 383 ;
232 : q = 478 ; 293 : q = 382 ;
233 : q = 477 ; 294 : q = 380 ;
234 : q = 475 ; 295 : q = 378 ;
235 : q = 473 ; 296 : q = 377 ;
236 : q = 472 ; 297 : q = 375 ;
237 : q = 470 ; 298 : q = 373 ;
238 : q = 469 ; 299 : q = 372 ;
239 : q = 467 ; 300 : q = 370 ;
240 : q = 466 ; 301 : q = 368 ;
241 : q = 464 ; 302 : q = 367 ;
242 : q = 463 ; 303 : q = 365 ;
243 : q = 461 ; 304 : q = 363 ;
244 : q = 460 ; 305 : q = 362 ;
245 : q = 458 ; 306 : q = 360 ;
246 : q = 457 ; 307 : q = 359 ;
247 : q = 455 ; 308 : q = 357 ;
248 : q = 453 ; 309 : q = 355 ;
249 : q = 452 ; 310 : q = 354 ;
250 : q = 450 ; 311 : q = 352 ;
251 : q = 449 ; 312 : q = 350 ;
252 : q = 447 ; 313 : q = 349 ;
253 : q = 446 ; 314 : q = 347 ;
254 : q = 444 ; 315 : q = 345 ;
255 : q = 442 ; 316 : q = 344 ;
256 : q = 441 ; 317 : q = 342 ;
257 : q = 439 ; 318 : q = 340 ;
258 : q = 438 ; 319 : q = 339 ;
259 : q = 436 ; 320 : q = 337 ;
260 : q = 435 ; 321 : q = 335 ;
261 : q = 433 ; 322 : q = 334 ;
262 : q = 431 ; 323 : q = 332 ;
263 : q = 430 ; 324 : q = 330 ;
264 : q = 428 ; 325 : q = 328 ;
265 : q = 427 ; 326 : q = 327 ;
266 : q = 425 ; 327 : q = 325 ;
267 : q = 424 ; 328 : q = 323 ;
268 : q = 422 ; 329 : q = 322 ;
269 : q = 420 ; 330 : q = 320 ;
270 : q = 419 ; 331 : q = 318 ;
271 : q = 417 ; 332 : q = 317 ;
272 : q = 416 ; 333 : q = 315 ;
273 : q = 414 ; 334 : q = 313 ;
274 : q = 412 ; 335 : q = 312 ;
275 : q = 411 ; 336 : q = 310 ;
276 : q = 409 ; 337 : q = 308 ;
277 : q = 408 ; 338 : q = 306 ;
278 : q = 406 ; 339 : q = 305 ;
279 : q = 404 ; 340 : q = 303 ;
280 : q = 403 ; 341 : q = 301 ;

342 : q = 300 ; 403 : q = 194 ;
343 : q = 298 ; 404 : q = 192 ;
344 : q = 296 ; 405 : q = 190 ;
345 : q = 295 ; 406 : q = 188 ;
346 : q = 293 ; 407 : q = 187 ;
347 : q = 291 ; 408 : q = 185 ;
348 : q = 289 ; 409 : q = 183 ;
349 : q = 288 ; 410 : q = 181 ;
350 : q = 286 ; 411 : q = 179 ;
351 : q = 284 ; 412 : q = 178 ;
352 : q = 283 ; 413 : q = 176 ;
353 : q = 281 ; 414 : q = 174 ;
354 : q = 279 ; 415 : q = 172 ;
355 : q = 277 ; 416 : q = 171 ;
356 : q = 276 ; 417 : q = 169 ;
357 : q = 274 ; 418 : q = 167 ;
358 : q = 272 ; 419 : q = 165 ;
359 : q = 271 ; 420 : q = 163 ;
360 : q = 269 ; 421 : q = 162 ;
361 : q = 267 ; 422 : q = 160 ;
362 : q = 265 ; 423 : q = 158 ;
363 : q = 264 ; 424 : q = 156 ;
364 : q = 262 ; 425 : q = 155 ;
365 : q = 260 ; 426 : q = 153 ;
366 : q = 258 ; 427 : q = 151 ;
367 : q = 257 ; 428 : q = 149 ;
368 : q = 255 ; 429 : q = 147 ;
369 : q = 253 ; 430 : q = 146 ;
370 : q = 251 ; 431 : q = 144 ;
371 : q = 250 ; 432 : q = 142 ;
372 : q = 248 ; 433 : q = 140 ;
373 : q = 246 ; 434 : q = 138 ;
374 : q = 245 ; 435 : q = 137 ;
375 : q = 243 ; 436 : q = 135 ;
376 : q = 241 ; 437 : q = 133 ;
377 : q = 239 ; 438 : q = 131 ;
378 : q = 238 ; 439 : q = 129 ;
379 : q = 236 ; 440 : q = 128 ;
380 : q = 234 ; 441 : q = 126 ;
381 : q = 232 ; 442 : q = 124 ;
382 : q = 231 ; 443 : q = 122 ;
383 : q = 229 ; 444 : q = 121 ;
384 : q = 227 ; 445 : q = 119 ;
385 : q = 225 ; 446 : q = 117 ;
386 : q = 224 ; 447 : q = 115 ;
387 : q = 222 ; 448 : q = 113 ;
388 : q = 220 ; 449 : q = 112 ;
389 : q = 218 ; 450 : q = 110 ;
390 : q = 217 ; 451 : q = 108 ;
391 : q = 215 ; 452 : q = 106 ;
392 : q = 213 ; 453 : q = 104 ;
393 : q = 211 ; 454 : q = 103 ;
394 : q = 209 ; 455 : q = 101 ;
395 : q = 208 ; 456 : q = 99 ;
396 : q = 206 ; 457 : q = 97 ;
397 : q = 204 ; 458 : q = 95 ;
398 : q = 202 ; 459 : q = 94 ;
399 : q = 201 ; 460 : q = 92 ;
400 : q = 199 ; 461 : q = 90 ;
401 : q = 197 ; 462 : q = 88 ;
402 : q = 195 ; 463 : q = 86 ;

```
464     :     q     =     85     ;
465     :     q     =     83     ;
466     :     q     =     81     ;
467     :     q     =     79     ;
468     :     q     =     77     ;
469     :     q     =     76     ;
470     :     q     =     74     ;
471     :     q     =     72     ;
472     :     q     =     70     ;
473     :     q     =     68     ;
474     :     q     =     66     ;
475     :     q     =     65     ;
476     :     q     =     63     ;
477     :     q     =     61     ;
478     :     q     =     59     ;
479     :     q     =     57     ;
480     :     q     =     56     ;
481     :     q     =     54     ;
482     :     q     =     52     ;
483     :     q     =     50     ;
484     :     q     =     48     ;
485     :     q     =     47     ;
486     :     q     =     45     ;
487     :     q     =     43     ;
488     :     q     =     41     ;
489     :     q     =     39     ;
490     :     q     =     38     ;
491     :     q     =     36     ;
492     :     q     =     34     ;
493     :     q     =     32     ;
494     :     q     =     30     ;
495     :     q     =     28     ;
496     :     q     =     27     ;
497     :     q     =     25     ;
498     :     q     =     23     ;
499     :     q     =     21     ;
500     :     q     =     19     ;
501     :     q     =     18     ;
502     :     q     =     16     ;
503     :     q     =     14     ;
504     :     q     =     12     ;
505     :     q     =     10     ;
506     :     q     =     9      ;
507     :     q     =     7      ;
508     :     q     =     5      ;
509     :     q     =     3      ;
510     :     q     =     1      ;
511     :     q     =     0      ;

  endcase
end


endmodule
```

```
// Author
// Version 1
// Created 28-Apr-2005
//
// Copyright (c) 2005, QuIC.  All Rights Reserved.
// There is no warranty with respect to
// the operation and/or functionality of the design files.
// No use for commercial purpose without authorisation/permission from the
auther!

// Descriptions
//------------------------------------------------
// Attention! use negedge of clk

module syn_rom_1(clk,address,q);
input clk;
input [8:0] address;
output [9:0] q;
reg [9:0] q;

always @(posedge clk)
begin
   case (address)
0   :   q   =   1   ;
1   :   q   =   3   ;
2   :   q   =   5   ;
3   :   q   =   7   ;
4   :   q   =   9   ;
5   :   q   =   10  ;
6   :   q   =   12  ;
7   :   q   =   14  ;
8   :   q   =   16  ;
9   :   q   =   18  ;
10  :   q   =   19  ;
11  :   q   =   21  ;
12  :   q   =   23  ;
13  :   q   =   25  ;
14  :   q   =   27  ;
15  :   q   =   28  ;
16  :   q   =   30  ;
17  :   q   =   32  ;
18  :   q   =   34  ;
19  :   q   =   36  ;
20  :   q   =   38  ;
21  :   q   =   39  ;
22  :   q   =   41  ;
23  :   q   =   43  ;
24  :   q   =   45  ;
25  :   q   =   47  ;
26  :   q   =   48  ;
27  :   q   =   50  ;
28  :   q   =   52  ;
29  :   q   =   54  ;
30  :   q   =   56  ;
31  :   q   =   57  ;
32  :   q   =   59  ;
33  :   q   =   61  ;
34  :   q   =   63  ;
35  :   q   =   65  ;
36  :   q   =   66  ;
37  :   q   =   68  ;
38  :   q   =   70  ;
39  :   q   =   72  ;
40  :   q   =   74  ;
41  :   q   =   76  ;
42  :   q   =   77  ;
43  :   q   =   79  ;
44  :   q   =   81  ;
45  :   q   =   83  ;
46  :   q   =   85  ;
47  :   q   =   86  ;
48  :   q   =   88  ;
49  :   q   =   90  ;
50  :   q   =   92  ;
51  :   q   =   94  ;
52  :   q   =   95  ;
53  :   q   =   97  ;
54  :   q   =   99  ;
55  :   q   =   101 ;
56  :   q   =   103 ;
57  :   q   =   104 ;
58  :   q   =   106 ;
59  :   q   =   108 ;
60  :   q   =   110 ;
61  :   q   =   112 ;
62  :   q   =   113 ;
63  :   q   =   115 ;
64  :   q   =   117 ;
65  :   q   =   119 ;
66  :   q   =   121 ;
67  :   q   =   122 ;
68  :   q   =   124 ;
69  :   q   =   126 ;
70  :   q   =   128 ;
71  :   q   =   129 ;
72  :   q   =   131 ;
73  :   q   =   133 ;
74  :   q   =   135 ;
75  :   q   =   137 ;
76  :   q   =   138 ;
77  :   q   =   140 ;
78  :   q   =   142 ;
79  :   q   =   144 ;
80  :   q   =   146 ;
81  :   q   =   147 ;
82  :   q   =   149 ;
83  :   q   =   151 ;
84  :   q   =   153 ;
85  :   q   =   155 ;
86  :   q   =   156 ;
87  :   q   =   158 ;
88  :   q   =   160 ;
89  :   q   =   162 ;
90  :   q   =   163 ;
91  :   q   =   165 ;
92  :   q   =   167 ;
93  :   q   =   169 ;
94  :   q   =   171 ;
95  :   q   =   172 ;
96  :   q   =   174 ;
97  :   q   =   176 ;
98  :   q   =   178 ;
```

99	:	q	=	179	;	160	:	q	=	286	;
100	:	q	=	181	;	161	:	q	=	288	;
101	:	q	=	183	;	162	:	q	=	289	;
102	:	q	=	185	;	163	:	q	=	291	;
103	:	q	=	187	;	164	:	q	=	293	;
104	:	q	=	188	;	165	:	q	=	295	;
105	:	q	=	190	;	166	:	q	=	296	;
106	:	q	=	192	;	167	:	q	=	298	;
107	:	q	=	194	;	168	:	q	=	300	;
108	:	q	=	195	;	169	:	q	=	301	;
109	:	q	=	197	;	170	:	q	=	303	;
110	:	q	=	199	;	171	:	q	=	305	;
111	:	q	=	201	;	172	:	q	=	306	;
112	:	q	=	202	;	173	:	q	=	308	;
113	:	q	=	204	;	174	:	q	=	310	;
114	:	q	=	206	;	175	:	q	=	312	;
115	:	q	=	208	;	176	:	q	=	313	;
116	:	q	=	209	;	177	:	q	=	315	;
117	:	q	=	211	;	178	:	q	=	317	;
118	:	q	=	213	;	179	:	q	=	318	;
119	:	q	=	215	;	180	:	q	=	320	;
120	:	q	=	217	;	181	:	q	=	322	;
121	:	q	=	218	;	182	:	q	=	323	;
122	:	q	=	220	;	183	:	q	=	325	;
123	:	q	=	222	;	184	:	q	=	327	;
124	:	q	=	224	;	185	:	q	=	328	;
125	:	q	=	225	;	186	:	q	=	330	;
126	:	q	=	227	;	187	:	q	=	332	;
127	:	q	=	229	;	188	:	q	=	334	;
128	:	q	=	231	;	189	:	q	=	335	;
129	:	q	=	232	;	190	:	q	=	337	;
130	:	q	=	234	;	191	:	q	=	339	;
131	:	q	=	236	;	192	:	q	=	340	;
132	:	q	=	238	;	193	:	q	=	342	;
133	:	q	=	239	;	194	:	q	=	344	;
134	:	q	=	241	;	195	:	q	=	345	;
135	:	q	=	243	;	196	:	q	=	347	;
136	:	q	=	245	;	197	:	q	=	349	;
137	:	q	=	246	;	198	:	q	=	350	;
138	:	q	=	248	;	199	:	q	=	352	;
139	:	q	=	250	;	200	:	q	=	354	;
140	:	q	=	251	;	201	:	q	=	355	;
141	:	q	=	253	;	202	:	q	=	357	;
142	:	q	=	255	;	203	:	q	=	359	;
143	:	q	=	257	;	204	:	q	=	360	;
144	:	q	=	258	;	205	:	q	=	362	;
145	:	q	=	260	;	206	:	q	=	363	;
146	:	q	=	262	;	207	:	q	=	365	;
147	:	q	=	264	;	208	:	q	=	367	;
148	:	q	=	265;		209	:	q	=	368	;
149	:	q	=	267	;	210	:	q	=	370	;
150	:	q	=	269	;	211	:	q	=	372	;
151	:	q	=	271	;	212	:	q	=	373	;
152	:	q	=	272	;	213	:	q	=	375	;
153	:	q	=	274	;	214	:	q	=	377	;
154	:	q	=	276	;	215	:	q	=	378	;
155	:	q	=	277	;	216	:	q	=	380	;
156	:	q	=	279	;	217	:	q	=	382	;
157	:	q	=	281	;	218	:	q	=	383	;
158	:	q	=	283	;	219	:	q	=	385	;
159	:	q	=	284	;	220	:	q	=	386	;

221 : q = 388 ; 282 : q = 484 ;
222 : q = 390 ; 283 : q = 486 ;
223 : q = 391 ; 284 : q = 487 ;
224 : q = 393 ; 285 : q = 489 ;
225 : q = 395 ; 286 : q = 490 ;
226 : q = 396 ; 287 : q = 492 ;
227 : q = 398 ; 288 : q = 493 ;
228 : q = 399 ; 289 : q = 495 ;
229 : q = 401 ; 290 : q = 496 ;
230 : q = 403 ; 291 : q = 498 ;
231 : q = 404 ; 292 : q = 499 ;
232 : q = 406 ; 293 : q = 501 ;
233 : q = 408 ; 294 : q = 502 ;
234 : q = 409 ; 295 : q = 504 ;
235 : q = 411 ; 296 : q = 505 ;
236 : q = 412 ; 297 : q = 507 ;
237 : q = 414 ; 298 : q = 508 ;
238 : q = 416 ; 299 : q = 510 ;
239 : q = 417 ; 300 : q = 511 ;
240 : q = 419 ; 301 : q = 513 ;
241 : q = 420 ; 302 : q = 514 ;
242 : q = 422 ; 303 : q = 516 ;
243 : q = 424 ; 304 : q = 517 ;
244 : q = 425 ; 305 : q = 518 ;
245 : q = 427 ; 306 : q = 520 ;
246 : q = 428 ; 307 : q = 521 ;
247 : q = 430 ; 308 : q = 523 ;
248 : q = 431 ; 309 : q = 524 ;
249 : q = 433 ; 310 : q = 526 ;
250 : q = 435 ; 311 : q = 527 ;
251 : q = 436 ; 312 : q = 529 ;
252 : q = 438 ; 313 : q = 530 ;
253 : q = 439 ; 314 : q = 532 ;
254 : q = 441 ; 315 : q = 533 ;
255 : q = 442 ; 316 : q = 535 ;
256 : q = 444 ; 317 : q = 536 ;
257 : q = 446 ; 318 : q = 537 ;
258 : q = 447 ; 319 : q = 539 ;
259 : q = 449 ; 320 : q = 540 ;
260 : q = 450 ; 321 : q = 542 ;
261 : q = 452 ; 322 : q = 543 ;
262 : q = 453 ; 323 : q = 545 ;
263 : q = 455 ; 324 : q = 546 ;
264 : q = 457 ; 325 : q = 547 ;
265 : q = 458 ; 326 : q = 549 ;
266 : q = 460 ; 327 : q = 550 ;
267 : q = 461 ; 328 : q = 552 ;
268 : q = 463 ; 329 : q = 553 ;
269 : q = 464 ; 330 : q = 555 ;
270 : q = 466 ; 331 : q = 556 ;
271 : q = 467 ; 332 : q = 557 ;
272 : q = 469 ; 333 : q = 559 ;
273 : q = 470 ; 334 : q = 560 ;
274 : q = 472 ; 335 : q = 562 ;
275 : q = 473 ; 336 : q = 563 ;
276 : q = 475 ; 337 : q = 564 ;
277 : q = 477 ; 338 : q = 566 ;
278 : q = 478 ; 339 : q = 567 ;
279 : q = 480 ; 340 : q = 569 ;
280 : q = 481 ; 341 : q = 570 ;
281 : q = 483 ; 342 : q = 571 ;

343 : q = 573 ;
344 : q = 574 ;
345 : q = 575 ;
346 : q = 577 ;
347 : q = 578 ;
348 : q = 580 ;
349 : q = 581 ;
350 : q = 582 ;
351 : q = 584 ;
352 : q = 585 ;
353 : q = 586 ;
354 : q = 588 ;
355 : q = 589 ;
356 : q = 590 ;
357 : q = 592 ;
358 : q = 593 ;
359 : q = 594 ;
360 : q = 596 ;
361 : q = 597 ;
362 : q = 598 ;
363 : q = 600 ;
364 : q = 601 ;
365 : q = 602;
366 : q = 604 ;
367 : q = 605 ;
368 : q = 606 ;
369 : q = 608 ;
370 : q = 609 ;
371 : q = 610 ;
372 : q = 612 ;
373 : q = 613 ;
374 : q = 614 ;
375 : q = 616 ;
376 : q = 617 ;
377 : q = 618 ;
378 : q = 620 ;
379 : q = 621 ;
380 : q = 622 ;
381 : q = 623 ;
382 : q = 625 ;
383 : q = 626 ;
384 : q = 627 ;
385 : q = 629 ;
386 : q = 630 ;
387 : q = 631 ;
388 : q = 632 ;
389 : q = 634 ;
390 : q = 635 ;
391 : q = 636 ;
392 : q = 637 ;
393 : q = 639 ;
394 : q = 640 ;
395 : q = 641 ;
396 : q = 642 ;
397 : q = 644 ;
398 : q = 645 ;
399 : q = 646 ;
400 : q = 647 ;
401 : q = 649 ;
402 : q = 650 ;
403 : q = 651 ;
404 : q = 652 ;
405 : q = 654 ;
406 : q = 655 ;
407 : q = 656 ;
408 : q = 657 ;
409 : q = 658 ;
410 : q = 660 ;
411 : q = 661 ;
412 : q = 662 ;
413 : q = 663 ;
414 : q = 664 ;
415 : q = 666 ;
416 : q = 667 ;
417 : q = 668 ;
418 : q = 669 ;
419 : q = 670 ;
420 : q = 672 ;
421 : q = 673 ;
422 : q = 674 ;
423 : q = 675 ;
424 : q = 676 ;
425 : q = 677 ;
426 : q = 679 ;
427 : q = 680 ;
428 : q = 681 ;
429 : q = 682 ;
430 : q = 683 ;
431 : q = 684 ;
432 : q = 685 ;
433 : q = 687 ;
434 : q = 688 ;
435 : q = 689 ;
436 : q = 690 ;
437 : q = 691 ;
438 : q = 692 ;
439 : q = 693 ;
440 : q = 695 ;
441 : q = 696 ;
442 : q = 697 ;
443 : q = 698 ;
444 : q = 699 ;
445 : q = 700 ;
446 : q = 701 ;
447 : q = 702 ;
448 : q = 703 ;
449 : q = 705 ;
450 : q = 706 ;
451 : q = 707 ;
452 : q = 708 ;
453 : q = 709 ;
454 : q = 710 ;
455 : q = 711 ;
456 : q = 712 ;
457 : q = 713 ;
458 : q = 714 ;
459 : q = 715 ;
460 : q = 716 ;
461 : q = 718 ;
462 : q = 719 ;
463 : q = 720 ;
464 : q = 721 ;

```
465 : q = 722 ;
466 : q = 723 ;
467 : q = 724 ;
468 : q = 725 ;
469 : q = 726 ;
470 : q = 727 ;
471 : q = 728 ;
472 : q = 729 ;
473 : q = 730 ;
474 : q = 731 ;
475 : q = 732 ;
476 : q = 733 ;
477 : q = 734 ;
478 : q = 735 ;
479 : q = 736 ;
480 : q = 737 ;
481 : q = 738 ;
482 : q = 739 ;
483 : q = 740 ;
484 : q = 741 ;
485 : q = 742 ;
486 : q = 743 ;
487 : q = 744 ;
488 : q = 745 ;
489 : q = 746 ;
490 : q = 747 ;
491 : q = 748 ;
492 : q = 749 ;
493 : q = 750 ;
494 : q = 751 ;
495 : q = 752 ;
496 : q = 753 ;
497 : q = 754 ;
498 : q = 755 ;
499 : q = 756 ;
500 : q = 757 ;
501 : q = 758 ;
502 : q = 758 ;
503 : q = 759 ;
504 : q = 760 ;
505 : q = 761 ;
506 : q = 762 ;
507 : q = 763 ;
508 : q = 764 ;
509 : q = 765 ;
510 : q = 766 ;
511 : q = 767 ;
  endcase
end


endmodule
```

```
// Author
// Version 1
// Created 28-Apr-2005
// Copyright (c) 2005, QuIC.  All Rights Reserved.
// There is no warranty with respect to
// the operation and/or functionality of the design files.
// No use for commercial purpose without authorisation/permission from the
auther!

// Descriptions
================================================
// top level of dead time inserter contains 6 'deadtime's

module dtime6x(
                ina1,
                clk,
                inc1,
                inc2,
                inb1,
                inb2,
                ina2,
                dtime,
                outa1,
                outc1,
                outc2,
                outb1,
                outb2,
                outa2
);

parameter e = 1'b1;

input       ina1;
input       clk;
input       inc1;
input       inc2;
input       inb1;
input       inb2;
input       ina2;
input       [8:1] dtime;
output      outa1;
output      outc1;
output      outc2;
output      outb1;
output      outb2;
output      outa2;

defparam b2v_1.f = e;
deadtime    b2v_1(.clk(clk),.in(ina1),.dtime(dtime),.out(outa1));
defparam b2v_2.f = e;
deadtime    b2v_2(.clk(clk),.in(ina2),.dtime(dtime),.out(outa2));
defparam b2v_3.f = e;
deadtime    b2v_3(.clk(clk),.in(inb1),.dtime(dtime),.out(outb1));
defparam b2v_4.f = e;
deadtime    b2v_4(.clk(clk),.in(inb2),.dtime(dtime),.out(outb2));
defparam b2v_5.f = e;
deadtime    b2v_5(.clk(clk),.in(inc1),.dtime(dtime),.out(outc1));
defparam b2v_6.f = e;
deadtime    b2v_6(.clk(clk),.in(inc2),.dtime(dtime),.out(outc2));
endmodule
```

```
// Author
// Version 1
// Created 28-Apr-2005

// Copyright (c) 2005, QuIC.  All Rights Reserved.
// There is no warranty with respect to
// the operation and/or functionality of the design files.
// No use for commercial purpose without authorisation/permission from the
auther!

// Descriptions
================================================

// insert deadtime by delaying rising edges of all signals
// determine values of dtime and has independent clk to count the delay

module deadtime(clk, dtime, in, out);
  parameter f = 1'b1;
  input clk;
  input [7:0] dtime;
  input in;
  output out;

  wire out_clk;
  wire [7:0] cnt;
  wire cnt1_clk_ctrl;
  reg out_d, out_q;
  reg [7:0] cnt_d;
  reg [7:0] cnt_q;


  assign out = out_q;
  always @(posedge out_clk)
    out_q <= out_d;

  always @(posedge cnt1_clk_ctrl)
    cnt_q <= cnt_d;

// Start of original equations
  assign out_clk = clk;
  assign cnt1_clk_ctrl = clk;


  always @(out_q or cnt_q or dtime or in) begin
    out_d = out_q;
    cnt_d = 8'b0000_0000;
    if (dtime == 8'b0000_0000) begin
                out_d = in;
    end else begin
                if (f)
                        casex (out_q)
                        1'b0: begin
                           if (in == 1'b1) begin
                                   cnt_d = cnt_q + 8'b0000_0001;
                           end
                           if (cnt_q == dtime) begin
                                   out_d = 1'b1;
                                   cnt_d = 8'b0000_0000;
                           end else begin
                                   out_d = 1'b0;
                           end
                        end
                        1'b1: begin
                           if (in == 1'b0) begin
                                   out_d = 1'b0;
                           end
                        end
                        endcase
                else
                        casex (out_q)
                        1'b1: begin
                           if (in == 1'b0) begin
                                   cnt_d = cnt_q + 8'b0000_0001;
                           end
                           if (cnt_q == dtime) begin
                                   out_d = 1'b0;
                                   cnt_d = 8'b0000_0000;
                           end else begin
                                   out_d = 1'b1;
                           end
                        end
                        1'b0: begin
                           if (in == 1'b1) begin
                                   out_d = 1'b1;
                           end
                        end
                        endcase
    end
  end
endmodule
```

```
// Author
// Version 1
// Created 28-Apr-2005

// Copyright (c) 2005, QuIC.  All Rights Reserved.
// There is no warranty with respect to
// the operation and/or functionality of the design files.
// No use for commercial purpose without authorisation/permission from the
auther!

// Descriptions
================================================

// find sector and shift degree which shifted PWN will be locate at the beginning
// sector = indeg mod 60
// phase shift = remainder

module mod60(indeg,lag,mod,shift);

input [8:0] indeg;
input lag;
output [2:0] mod;
output  [8:0] shift;

wire [8:0] degree;
reg [2:0] mod;
reg [5:0] rem;


assign degree = lag ? 360-indeg: indeg ;

always @ (degree)
begin
        if (degree > 360)                       begin
                mod <= 3'bxxx;
                rem <= 6'bxxxxxx;               end
        else if (degree >= 300)         begin
                mod <= 5;
                rem <= degree-300;              end
        else if (degree >= 240)         begin
                mod <= 4;
                rem <= degree-240;              end
        else if (degree >= 180)         begin
                mod <= 3;
                rem <= degree-180;              end
        else if (degree >= 120)         begin
                mod <= 2;
                rem <= degree-120;              end
        else if (degree >= 60)          begin
                mod <= 1;
                rem <= degree-60;               end
        else                                                    begin
                mod <= 0;
                rem <= degree;                                  end
end

assign shift = (rem * 17) >> 1; // scaled by 512/60 = 8.5 = 17/2

endmodule
// Author P
// Version 2
// Created 09-Jun-2005
//
// Copyright (c) 2005, QuIC.  All Rights Reserved.
// There is no warranty with respect to
// the operation and/or functionality of the design files.
// No use for commercial purpose without authorisation/permission from the
auther!

// Descriptions
================================================

// new version of vector generator
// reduced from 36 state machine to only 5 state machines (2states at initial)
// capable to start at any sector and any phase shift
// make it own decision of the next vector to minimize switching activity

module vector_gen(cnt, ta, tb, tx, clk, sector, reset, abc, check, state, tmon);
parameter VecZ0 =               3'b000 ;
parameter VecZ1 =  3'b111 ;
parameter VecIni =  3'b000 ;
  input [10:1] ta;
  input [10:1] tb;
  input [10:1] tx;
  input clk;
  input [3:1] sector;
  input reset;
  output [3:1] abc;
  output check;
  output [3:1] state;
  output [10:1] cnt;
  output [10:1] tmon;

  reg check;
  reg [10:1] cnt;
  reg cnt_rst;
  reg [3:1] st_q, st_d;
  reg [3:1] abc, abc_st;
  reg [10:1] tZero, tOdd, tEven;
  reg [10:1] Tab, Tz;  //store period of ta, tb, tx for comparison in state
machine
  reg [3:1]  VecA, VecB, VecOdd, VecEven, VecAB, VecZ; //classify vector
to be odd or even (no of '1')
  reg [3:1]  VecOld;            //store previous value to select the next vector
with mim switching

  //debugging port
  assign state = st_q;
  assign tmon = Tab;

  //select vector A and B to be combined
  always @(sector)
  begin
        case (sector)
                0 : begin VecA <= 3'b100; VecB <= 3'b110; end
                1 : begin VecA <= 3'b110; VecB <= 3'b010; end
                2 : begin VecA <= 3'b010; VecB <= 3'b011; end
                3 : begin VecA <= 3'b011; VecB <= 3'b001; end
                4 : begin VecA <= 3'b001; VecB <= 3'b101; end
                5 : begin VecA <= 3'b101; VecB <= 3'b100; end
                default : begin VecA <= 3'b000; VecB <=
3'b000; end
```

```
            endcase
end

//Categorise Odd or Even type when 'check' signal, for the next vector
always @ (check or VecA or VecB or ta or tb or tx)
begin
if (check) begin
            tZero <= tx;
            if (^VecA)
             begin
                        VecOdd <= VecA;
                        VecEven <= VecB;
                        tOdd <= ta;
                        tEven <= tb;
             end
            else
             begin
                        VecOdd <= VecB;
                        VecEven <= VecA;
                        tOdd <= tb;
                        tEven <= ta;
            end
end
end

//store previous vector
always @ (posedge cnt_rst)
            VecOld <= abc;

//select the right vector to minimize switching
wire [3:1] NextVecAB, NextVecZ;
wire [10:1] NextTab;
assign NextVecAB = ^VecOld ? VecEven : VecOdd;
assign NextTab = ^VecOld ? tEven : tOdd;
assign NextVecZ          = ^VecOld ? VecZ0 : VecZ1;

//register the right vector
always @ (negedge cnt_rst)
begin
            VecAB <= NextVecAB;
            Tab <= NextTab;
            VecZ <= NextVecZ;
            Tz <= tZero;
end

//counter ta, tb or tx with reset
always @ (posedge clk)
if (cnt_rst) cnt <= 0;
else cnt <= cnt+1;

//state change with reset
=================================================
always @(posedge clk )
  if (reset)
            st_q <= 0;
  else
            st_q <= st_d;

//state combinational functions
always @(st_q  or Tab or Tz or VecAB or VecZ or cnt )
begin
cnt_rst <= 0;
case (st_q)
            0: begin        //initial state
              check <= 1'b0;
              abc_st <= 0;
              cnt_rst <= 1;
              st_d <= 7;
             end
             7: begin        //start with gen 'check' pulse
              check <= 1'b1;
              abc_st <= 0; //initial value
              if (cnt == 15) //waiting for calculating ta, tb, tx
              begin
               cnt_rst <= 1;
                        st_d <= 1;      //change state
              end
              else
               st_d <= 7;   //same state
             end
 1: begin    //period 1
              check <= 1'b0;
              abc_st <= VecAB;
              if (cnt == Tab)  //change state
              begin
                cnt_rst <= 1'b1;
                st_d <= 2;
              end
              else
                st_d <= 1; //same state
             end
             2: begin        //period 2
              check <= 1'b0;
              abc_st <= VecAB;
              if (cnt == Tab)  //change state
              begin
              cnt_rst <= 1'b1;
              st_d <= 3;
              end
              else
                st_d <= 2; //same state
             end
  3: begin    //period 3
              check <= 1'b1;
              abc_st <= VecZ;
              if (cnt == Tz)  //change state
              begin
              cnt_rst <= 1'b1;
              st_d <= 1;
              end
              else
                st_d <= 3; //same state
             end
  endcase

end

//sync output
always @ (posedge clk)
  abc <= abc_st;

endmodule
```

ประวัติผู้เขียน

ชื่อ-นามสกุล	ประสูตร เดชสุวรรณ
วัน เดือน ปีเกิด	19 กุมภาพันธ์ 2516 ที่ กรุงเทพ มหานครฯ
ที่อยู่	222 หมู่2 ถนน บางพูน ตำบล ประชาธิปัตย์ อำเภอ ธัญบุรี จังหวัด ปทุมธานี 12130
ประวัติการศึกษา	2539 วิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมอิเล็กทรอนิกส์ มหาวิทยาลัยเทคโนโลยีมหานคร
ประสบการณ์การทำงาน และผลงานวิจัย	
พ.ศ.2541-ปัจจุบัน	อาจารย์ประจำมหาวิทยาลัยเทคโนโลยีมหานคร