# การขับสเตปปิงมอเตอร์แบบหลายโหมดด้วย FPGA

# MULTI-MODE EXCITING OF STEPPING MOTOR BASED ON FPGA

นายเกษมศักดิ์ คุณพิชิตชัย

นายประสิทธิ์ เกื้อทวีกุล



ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมการวัดคุม

ภาควิชาวิศวกรรมการวัดคุม คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2550

# MULTI-MODE EXCITING OF STEPPING MOTOR BASED ON FPGA

**KASEMSAK KHUNPICHITCHAI**

**PRASIT KUETHAWEEKUN**

**A THESIS SUBMITTED IN PARTIAL FULFILLMENT**

**OF THE REQUIREMENT FOR THE DEGREE OF**

**BACHELOR OF ENGINEERING IN INSTRUMENTATION ENGINEERING**

**DEPARTMENT OF INSTRUMENTATION ENGINEERING**

**FACULTY OF ENGINEERING**

**KING MONGKUT'S INSTITUE OF TECHNOLOGY LARDKRABANG**

**2007**

**ภาควิชาวิศวกรรมการวัดคุม**

**คณะวิศวกรรมศาสตร์**

**สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง**

**ใบรับรองปริญญานิพนธ์**

.....

| | |
|---|---|
| **หัวข้อปริญญานิพนธ์** | การขับสเตปปิงมอเตอร์แบบหลายโหมดด้วย FPGA<br>MULTI-MODE EXCITING OF STEPPING<br>MOTOR BASED ON FPGA |
| **นักศึกษาผู้จัดทำ** | นายเกษมศักดิ์ คุณพิชิตชัย **รหัสนักศึกษา** 47012050 |
| | นายประสิทธิ์ เกื้อทวีกุล **รหัสนักศึกษา** 47012066 |
| **ปริญญา** | วิศวกรรมศาสตรบัณฑิต |
| **สาขาวิชา** | วิศวกรรมการวัดคุม |
| **ปีการศึกษา** | 2550 |

| อาจารย์ผู้ควบคุมปริญญานิพนธ์ | ลายมือชื่อ |
|---|---|
| รองศาสตราจารย์ ไสว พงศ์สวัสดิ์ | |

**ภาควิชารับรองแล้ว**

**(รศ.ประภาษ อุดคกิมาพันธุ์)**

**หัวหน้าภาควิชาวิศวกรรมการวัดคุม**

| | | | |
|---|---|---|---|
| **หัวข้อปริญญานิพนธ์** | การขับสเตปปิงมอเตอร์แบบหลายโหมดด้วย FPGA<br>MULTI-MODE EXCITING OF STEPPING<br>MOTOR BASED ON FPGA | | |
| **นักศึกษาผู้จัดทำ** | นายเกษมศักดิ์ | คุณพิชิตชัย | รหัสนักศึกษา 47012050 |
| | นายประสิทธิ์ | เกื้อทวีกุล | รหัสนักศึกษา 47012066 |
| **อาจารย์ที่ปรึกษา** | รองศาสตราจารย์ไสว | พงศ์สวัสดิ์ | |
| **ปีการศึกษา** | 2550 | | |

## บทคัดย่อ

โครงงานนี้ได้นำเสนอการประยุกต์ใช้ FPGA ในการสร้างสัญญาณขับและการควบคุมสเตปปิงมอเตอร์ ด้วยสัญญาณกระตุ้นแบบหลายโหมดที่สามารถปรับเปลี่ยนโหมดการกระตุ้นให้สอดคล้องกับความเร็วรอบที่ต้องการ สัญญาณกระตุ้นที่ออกแบบไว้มี 3 โหมด คือ Full step Half step และ Sinusoidal ministep ซึ่งสัญญาณขับในแต่ละโหมดจะสร้างให้มีจำนวนบิตเท่ากับจำนวนเฟสของสเตปปิงมอเตอร์และจัดเรียงเป็นข้อมูลดิจิตอลแบบอนุกรม 1 บิตต่อ 1 เฟสการกระตุ้น จัดเก็บไว้ในหน่วยความจำ ROM ในโครงงานจะใช้สัญญาณกระตุ้นทั้งหมด 3 โหมดเพื่อขับสเตปปิงมอเตอร์แบบ 4 เฟส ที่ทำการปรับเปลี่ยนโหมดการกระตุ้นแบบอัตโนมัติ เพื่อให้สอดคล้องกับตามความถี่อินพุตที่ป้อน และเหมาะสมกับช่วงตอบสนองทางความเร็วของแต่ละโหมด อีกทั้งได้มีการตรวจจับการเกิดสลิปของมอเตอร์ ด้วยการตรวจจับค่ากระแสกระตุ้นในวงจรขับเพื่อป้องกันความเสียหายที่จะเกิดขึ้นในวงจรขับเมื่อมอเตอร์เกิดการสลิป

**Thesis Title** Multi-Mode Exciting of Stepping Motor Based on FPGA

**Authors** Mr.Kasemsak Khunpichitchai

Mr.Prasit Kuethaweekun

**Thesis Advisor** Assoc. Prof. Sawai Pongswatd

**Year** 2007

## ABSTRACT

This project presents an application of FPGA in order to generate multi-mode exciting signals and control the stepping motor. The project designs exciting signals as 1 bit serial data and group to 3 modes, Full step, Half step, and Sinusoidal mini step respectively. Each mode of exciting signals is automatically selected depend on actual rotating speed. The output bits of each mode are equal to phase number of stepping motor. The experiment performs with 4-phase stepping motor that is driven by automatic selection. The test results show the exciting signals and the speed response of 3-mode. Moreover, the driver circuit can detect the exciting current for slip check and driver protection.

# กิตติกรรมประกาศ

คณะผู้จัดทำขอขอบพระคุณ รศ. ไสว พงศ์สวัสดิ์ ที่ได้ให้คำแนะนำในการจัดทำโครงงานชิ้นนี้ และให้คำปรึกษาเพื่อเป็นแนวทางในการแก้ไข และดูแลตรวจสอบจนโครงงานวิศวกรรมฉบับนี้เสร็จลุล่วงไปได้ด้วยดี และขอขอบพระคุณผู้ที่เกี่ยวข้องกับโครงงานนี้ทุกท่านรวมทั้งผู้ที่ไม่ได้กล่าวนาม หากโครงงานนี้มีข้อผิดพลาดประการใด ทางคณะผู้จัดทำต้องขออภัยไว้ ณ ที่นี้ด้วย

คณะผู้จัดทำ

# สารบัญ

หน้า

# สารบัญ (ต่อ)

หน้า

# สารบัญตาราง

**ตารางที่** **หน้า**

# สารบัญรูป

**รูปที่** **หน้า**

# สารบัญรูป (ต่อ)

**รูปที่** **หน้า**

# สารบัญรูป (ต่อ)

**รูปที่** **หน้า**

# บทที่ 1
# บทนำ

## 1.1 ความสำคัญของปริญญานิพนธ์

ในอดีตการออกแบบวงจรดิจิตอลจะต้องนำ IC มาทำการเชื่อมต่อลงในแผ่นปริ้นซึ่งในกรณีที่วงจรที่ทำการออกแบบมีความซับซ้อนทำให้การแก้ไขข้อผิดพลาดต่าง ๆ เป็นไปได้ยากใช้เวลานานและมีค่าใช้จ่ายสูงดังนั้นด้วยข้อจำกัดดังกล่าวจึงได้มีการพัฒนาการออกแบบระบบดิจิตอลให้สามารถนำวงจรทั้งหมดเก็บลงอยู่ในวงจรรวม (IC) เพียงหนึ่งตัวและทำการแก้ไขได้โดยการโปรแกรมซ้ำภายใน IC นั้นซึ่งจะทำให้ประหยัดเวลาและค่าจ่ายในการแก้ไขวงจรดิจิตอลที่ออกแบบไว้

ปัจจุบันฮาร์ดแวร์ที่สามารถโปรแกรมได้ หรือที่เรียกกันว่า FPGA (Field Programmable Gate Array) มีความนิยมในการใช้งานในการออกแบบระบบดิจิตอลสูงมากขึ้นเรื่อย ๆ เนื่องจากความง่ายในการออกแบบและความรวดเร็วในการผลิตและสร้างโดยที่สามารถกระทำได้โดยใช้เครื่องคอมพิวเตอร์มาตรฐานทั่ว ๆ ไปนอกจากนี้สิ่งที่ทำให้แตกต่างจากระบบฮาร์ดแวร์เดิม ๆ เช่น ASIC (Application Specific IC) คือความสามารถในการโปรแกรมหรือใช้งานได้หลาย ๆ ครั้งโดยไม่มีความเสียหายหรือค่าใช้จ่ายเพิ่มเติมเกิดขึ้นซึ่งชิฟประเภท FPGA นี้ทำให้มีการเปลี่ยนแปลงที่สำคัญของการออกแบบระบบดิจิตอลในปัจจุบัน

ยกตัวอย่างการประยุกต์ใช้ FPGA โดยนำ FPGA มาใช้ในการควบคุมและสร้างสัญญาณกระตุ้นหลายรูปแบบในการขับ Stepping Motor คือ Full step Half step และ Sinusoidal ministep ซึ่ง ในการทดสอบป้อนความถี่จะได้ผลตอบสนองทางความเร็วที่แตกต่างกันจึงได้ประยุกต์นำสัญญาณขับ 3 โหมดมาจัดเรียงบิตภายใน เพื่อทำให้ Stepping Motor สามารถหมุนด้วยความเร็วที่ต่อเนื่องและให้ผลตอบสนองทางความเร็วได้กว้างขึ้นดดยการรวมโหมดการกระตุ้นทั้งสามเข้าด้วยกันรวมทั้งสามารถหยุดการทำงานของมอเตอร์เมื่อเกิดการสลิปของมอเตอร์

## 1.2 วัตถุประสงค์ของปริญญานิพนธ์

1. เพื่อศึกษาการออกแบบวงจรดิจิตอลโดยใช้ FPGA ในการออกแบบ
2. เพื่อศึกษาโครงสร้างของสเตปปิงมอเตอร์
3. ศึกษาสัญญาณในรูปแบบต่าง ๆ ที่ใช้ในการขับสเตปปิงมอเตอร์ และลักษณะกระแสขณะใช้งาน

## 1.3 ขอบเขตของปริญญานิพนธ์

1. ประยุกต์ใช้ FPGA ในการสร้างสัญญาณกระตุ้น Full step Half step และ Sinusoidal ministepได้
2. สามารถที่จะเปลี่ยนรูปแบบสัญญาณกระตุ้นของ Stepping Motor ตามความถี่ที่ป้อนแบบ Manual และ Auto
3. สามารถหยุดการกระตุ้นของสัญญาณที่ป้อนให้กับ Stepping Motor ได้เมื่อเกิดการสลิป
4. สามารถหมุนทวนเข็มหรือตามเข็มนาฬิกาได้

## 1.4 ขั้นตอนการศึกษา

1. ศึกษาโครงสร้าง และวิธีการออกแบบวงจรดิจิตอลด้วย FPGA
2. ศึกษาโครงสร้างของสเตปปิงมอเตอร์
3. ทำการออกแบบโปรแกรมที่ใช้สร้างสัญญาณขับสเตปปิงมอเตอร์
4. สร้าง DRIVER ที่ใช้ในการขับสเตปปิงมอเตอร์
5. ทำการทดลองและบันทึกผล

## 1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. สามารถที่จะออกแบบวงจรดิจิตอลด้วย FPGA ได้
2. สามารถที่จะควบคุมการหมุนของสเตปปิงมอเตอร์ได้
3. สามารถนำสัญญาณที่ใช้ในการขับสเตปปิงมอเตอร์ไปประยุกต์ใช้ในงานต่าง ๆ ได้

# บทที่ 2
# ทฤษฎี

## 2.1 ความรู้เกี่ยวกับ FPGA

### 2.1.1 FPGA คืออะไร

FPGA ย่อมาจาก Field Programmable Gate Array เป็นวงจรรวมทางดิจิตอลที่สามารถโปรแกรมวงจรหรือฟังค์ชันการทำงานลงไปภายในตัวชิฟได้เองเหมาะสำหรับการออกแบบวงจรและการออกแบบชิฟต้นแบบของวงจรทางดิจิตอล ข้อดีคือเปรียบเทียบกับการออกแบบวงจรดิจิตอลโดยการใช้ IC Gates หรือใช้ IC TTL หลาย ๆ ตัวบนแผ่น PCB เราสามารถออกแบบวงจรการเชื่อมต่อและคุณสมบัติต่างๆด้วย Software ได้ จากนั้นเมื่อทดลอง Simulate ได้ผลน่าพอใจแล้วจึงโปรแกรมลงบนชิฟ FPGA จะเห็นว่าการแก้ไขทำได้ง่ายเพียงแก้บน Software (เสมือนอุปกรณ์ดิจิตอลของคุณอยู่ในรูปของ Software แก้ไขง่ายและแลกเปลี่ยนกันใช้ได้) และทำการโปรแกรมใหม่ (โปรแกรมซ้ำได้) ลดความยุ่งยากจากการเปลี่ยนอุปกรณ์ใหม่การนำ IC จำนวนมากมาต่อกันการออกแบบ PCB ใหม่ และความผิดพลาดที่อาจเกิดขึ้นได้จากลายวงจรลดการเกิดสัญญาณรบกวนจากการออกแบบ PCB และการใช้อุปกรณ์มาก ๆ ได้ เปรียบเทียบกับการออกแบบโดย ASIC การออกแบบวงจรรวม (IC) ต้นแบบโดย ASIC พัฒนาได้ยากเนื่องจากการแก้ไขวงจรแต่ละครั้งหมายถึงการเริ่มต้นขบวนการใหม่ทั้งหมดเช่น การออกแบบ Layout และการทำบน Silicon wafer เป็นต้นรวมถึงทรัพยากรทั้ง Hardware และ Software ในการออกแบบมีราคาแพง ดังนั้น การนำ FPGA ไปช่วยในการออกแบบทำให้การพัฒนาและการแก้ไขทำได้สะดวกและประหยัดขึ้นการออกแบบทำได้โดยเขียนวงจร Schematics ประกอบกับการเขียนภาษาอธิบายลักษณะพฤติกรรมหรือ Hardware Description Language จากนั้นทำการสังเคราะห์และโปรแกรมลงบนชิพ FPGA ด้วย Software เช่น MAX+PLUS II ผู้ใช้สามารถออกแบบและแก้ไขวงจรได้ง่ายจะเห็นว่าเทคโนโลยี FPGA จะช่วยให้นักศึกษาและผู้ที่สนใจสามารถออกแบบ IC ของตนเองได้นอกจากนี้เมื่อนักออกแบบสร้าง IC ของตนเองขึ้นมาแล้วยังสามารถป้องกันการลอกเลียนแบบได้อีกด้วย

**2.1.2 การแบ่งชนิดของ FPGA**

การแบ่งหมวดของ FPGA นั้นสามารถแบ่งแยกตามคุณลักษณะได้ดังนี้

**2.1.2.1 โครงสร้างสถาปัตยกรรมภายในของอุปกรณ์**

การแบ่งโครงสร้างสถาปัตยกรรมของ FPGA แบ่งได้เป็น 2 ลักษณะคือ

- Coarse-grained โครงสร้างสถาปัตยกรรมแบบ Coarse-grained จะประกอบด้วยลอจิกบล็อกขนาดใหญ่ เช่น มักประกอบด้วย LUT และ Flip-Flops 2 อัน หรือมากกว่าตัวอย่าง FPGA ได้แก่ Xilinx ตระกูล XC4K, Spartan, Virtex หรือ Altera ตระกูล FLEX, APEX

- Fine-grained ประกอบด้วยลอจิกบล็อกแบบง่ายจำนวนมากลอจิกบล็อกประกอบด้วย ลอจิกฟังก์ชันหรือ 2 อินพุทหรือมัลติเพล็กเซอร์แบบ 4-1 และฟลิปฟล็อป ตัวอย่าง FPGA แบบ fine-grained เช่น ตระกูล ACT ของบริษัท ACTEL

**2.1.2.2 เทคโนโลยีการโปรแกรม (Programming Technology)**

เทคนิคการโปรแกรมไอซี FPGAแบ่งได้ดังนี้

- FPGA ชนิด SRAM นั้นสามารถทำการโปรแกรมซ้ำ ๆ ได้หลายครั้งแต่ในการใช้งานในภาคสนามจำเป็นต้องใช้ไอซี สำหรับเก็บข้อมูลคอนฟิกกูเรชันของวงจรการโหลดข้อมูลจากตัวเก็บบิตสามารถเกิดขึ้นได้โดยอัตโนมัติเมื่อเปิดสวิตซ์หรืออาจโหลดโปรแกรมผ่านไมโครโปรเซสเซอร์ก็ได้ เช่นโหลดโปรแกรมด้วย PC ผ่านสายดาวน์โหลด ข้อดี FPGA นอกจากจะมีขนาดของเกตสูงแล้วยังสามารถโปรแกรมซ้ำได้บ่อยตามที่ต้องการ

- FPGA ชนิด Fuse หรือ Anti-fuse จะสามารถทำการโปรแกรมได้เพียงครั้งเดียว (OTP) และไม่สามารถแก้ไขหรือโปรแกรมซ้ำได้อีกแต่ข้อมูลการโปรแกรมไม่สูญหายเมื่อวงจรถูกตัดแหล่งจ่ายไฟ การโปรแกรมไอซี FPGA ชนิดนี้จะต้องใช้เครื่องโปรแกรมไอซี

- FPIC (Field Programmable Interconnect device) FPIC จริง ๆ แล้วไม่ได้ไอซีทางด้านลอจิกแต่จะเป็นอุปกรณ์ที่สามารถโปรแกรมส่วนที่เชื่อมต่อ(wiring) เช่นการเชื่อมตัวให้ไอซีหนึ่งตัวสามารถเลือกได้ว่าให้เชื่อมต่อเข้ากับไอซีตัวไหนได้บริษัทที่ผลิตอุปกรณ์ประเภทนี้ได้แก่ไอซี Digital Cross paint Switch ของบริษัท I-Cube Digital cross point device ของบริษัท Lattice หรืออุปกรณ์ Hardware emulator ของบริษัท Aptix

**2.1.3 การออกแบบด้วย FPGA**

การออกแบบวงจรดิจิตอลด้วย FPGA โดยทั่วไปมีองค์ประกอบ 3 ส่วน

**2.1.3.1 ซอฟต์แวร์ Design Entry**

- โดยใช้ Schematic Design Entry ใช้ไลบารีของ FPGA

- ใช้ภาษา HDL เช่น VHDL, Verilog, การออกแบบโดยใช้ภาษาชั้นสูงนั้นการทดสอบของวงจรนั้นยังไม่ขึ้นกับเทคโนโลยีเป้าหมาย (Technology independent) ผู้ออกแบบไม่จำเป็นต้องกังวลถึงค่าความหน่วงทางเวลาของอุปกรณ์ที่มากับเทคโนโลยีนั้นการทดสอบความถูกต้องเป็นในลักษณะการตรวจสอบระดับฟังก์ชันการทำงานโดยใช้ซอฟต์แวร์สำหรับจำลองการทำงาน (Simulation)

**2.1.3.2 Design Implementation**

ขั้นตอนนี้ ต่อเนื่องจากขั้นตอนที่ 1 ซึ่งเกี่ยวข้องกับการแปลงแบบที่ได้ออกแบบจาก Schematic หรือ HDL ให้เป็นลอจิกซึ่งอาจใช้ซอร์ฟแวร์สำหรับสังเคราะห์วงจร (Logic synthesis) แล้วทำการแบ่งลอจิกเป็นส่วน ๆ (Partitioning) และวางตำแหน่ง (Placement) ของลอจิกทำการเชื่อมต่อสายสัญญาณ (routing) สุดท้ายเป็นการสร้างไฟล์สำหรับโปรแกรมลงชิป (bit file)

**2.1.3.3 Device Programming**

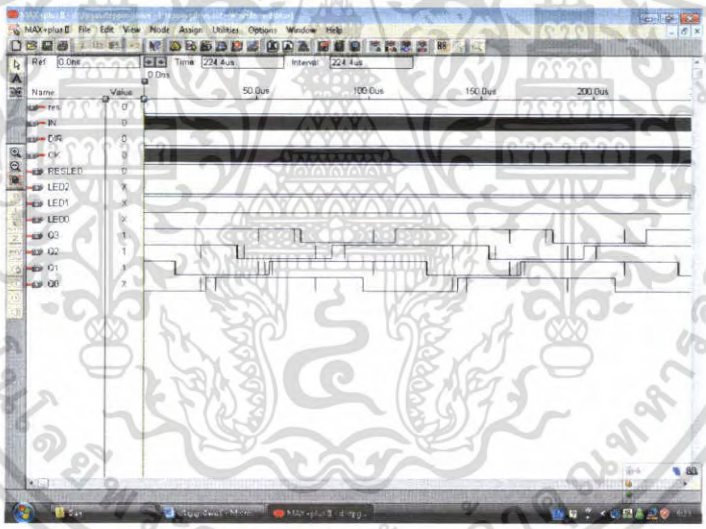
การโปรแกรมอุปกรณ์หรือชิป FPGA นั้น มีเทคนิคหรือวิธีใหญ่ ๆ 3 ลักษณะ ทั้งนี้ตัวชิปจะต้องสนับสนุนการทำงานในโหมดของการโปรแกรมเหล่านี้ด้วย

- การโปรแกรมโดยผ่านสายดาวน์โหลด หรือผ่าน JTAG หรือผ่าน ISP
- การโปรแกรมโดยใช้ ตัวเก็บข้อมูลไฟล์บิต
- การโปรแกรมโดยใช้เครื่องโปรแกรมไอซี

**2.1.4 โปรแกรม MAX+PLUS II คืออะไร**

โปรแกรม MAX+PLUS II เป็นโปรแกรมของบริษัท Altera Corporation ซึ่งใช้สำหรับการออกแบบ การสังเคราะห์ และการโปรแกรมลงบนชิฟ นั่นคือผู้ออกแบบสามารถทำทุกขั้นตอนด้วยโปรแกรม MAX+PLUS II โปรแกรมเดียว ผู้ที่สนใจสามารถดาวโหลดได้จาก www.altera.com ขนาดประมาณ 50 MBytes และในชุดพัฒนา/ชุดทดลอง CPLD/FPGA ของบริษัท ASTRON LOGIC R&D ทุกชุดจะแนบโปรแกรม MAX+PLUS II Baseline ให้ไปด้วย ดังรูปที่ 2.1


**รูปที่ 2.1** โปรแกรม MAXPLUSII



**รูปที่ 2.2** สัญญาณที่ได้จากการ SIMULATE

### 2.1.5 FPGA ต่างจาก CPLD อย่างไร

CPLD ย่อมาจาก Complex Programmable Logic Device ซึ่งเทคโนโลยีที่ใช้จะเหมือนกับ EEPROM ทำให้มีความจุของเกทต่ำโดยทั่วไปจะน้อยกว่า 20,000 เกทแต่ข้อดีของ EEPROM Based FPGA คือสามารถเก็บข้อมูลที่โปรแกรมลงไปได้โดยไม่จำเป็นต้องมีไฟเลี้ยง และในการโปรแกรมจะใช้ทรานซิสเตอร์ 1 ตัวต่อ 1 บิท ซึ่งการโปรแกรมสามารถทำได้ประมาณ 10,000 ครั้ง FPGA ย่อมาจาก Field Programmable Gate Array ใช้เทคโนโลยีในการโปรแกรมเหมือนกับ SRAM (Static RAM) ทำให้สามารถโปรแกรมซ้ำได้โดยไม่จำกัดจำนวนครั้งนอกจากนี้ยังมีความจุของเกทในระดับปานกลางถึงสูงมาก (ประมาณ 10,000 – 1,000,000 เกท) ซึ่งข้อดีของ SRAM Based FPGA คือใช้เวลาในการโปรแกรมน้อย (ระดับ nsec) การโปรแกรมทำได้ง่ายเทียบได้กับการเขียน SRAM ทั่วไป และเหมาะสำหรับที่มีความสลับซับซ้อนส่วนข้อเสียคือไม่สามารถเก็บโปรแกรมในภาวะที่ไม่มีไฟเลี้ยงได้ดังนั้น FPGA ชนิดนี้จึงมักใช้ควบคู่กับ ROM เพื่อเก็บโปรแกรมและทำการโหลดโปรแกรมลงในตัวชิพในขณะที่เริ่มต้นใช้งานข้อสังเกตประการหนึ่งคือ CPLD จะมีความจุต่ำกว่า FPGA รวมถึงมีคุณสมบัติพิเศษอื่น ๆ น้อยกว่า เช่น ไม่มี RAM เป็นต้นดังนั้นในการใช้งานออกแบบวงจรที่ซับซ้อนจึงแนะนำว่าควรใช้ FPGA จะเหมาะสมกว่าและควรศึกษาข้อมูลของ FPGA แต่ละตระกูลด้วยเนื่องจากมักจะมีคุณสมบัติพิเศษต่างกัน

### 2.1.6 ภาษา VERILOG HDL

Verilog หรือชื่อเต็ม ๆ คือ Verilog HDL เป็นภาษาที่ใช้ในการอธิบายลักษณะการทำงานของฮาร์ดแวร์ (Hardware description language) ภาษาหนึ่งที่ถูกสร้างขึ้นมาในช่วง ค.ศ. 1984-1985 โดย Philip Moorby ที่ต้องการภาษาที่ง่ายและมีประสิทธิภาพในการอธิบายลักษณะของวงจรดิจิตอลสำหรับการโมเดล การจำลองการทำงานและการวิเคราะห์การทำงานภาษานี้ได้กลายเป็นลิขสิทธิ์ของบริษัท Design GatewayAutomation ซึ่งต่อมาได้รวมกับบริษัท Cadence Design Systems และตั้งแต่ปี ค.ศ. 1990 เป็นต้นมาทางCadence ได้เปิดให้เป็นภาษาที่ทุกคนสามารถใช้งานได้ ทั้งนี้เพื่อให้ง่ายในการกำหนดมาตรฐานและพัฒนาร่วมกันและในที่สุดภาษานี้ได้เข้าเป็นภาษามาตรฐานของ IEEE ในปี ค.ศ. 1995 (IEEE Standard 1364)คุณลักษณะที่สำคัญในภาษานี้ได้แก่ Universal: ภาษา Verilog อนุญาตให้การออกแบบทั้งระบบทำได้ภายใต้สภาวะแวดล้อมของการออกแบบ (Design environment) เดียวกันที่ประกอบด้วยการวิเคราะห์และการตรวจสอบ(Analysis and verification) อย่างไรก็ตาม Verilog อาจไม่เหมาะกับระดับของการออกแบบที่เป็น Complex system design ที่ต้องการภาษาชั้นสูงกว่านี้ที่อาจต้องการความสามารถในการอธิบายการทำงานได้ทั้งซอฟท์แวร์และฮาร์ดแวร์ Extensibility: มาตรฐาน IEEE std 1364 ประกอบด้วย Verilog PLI (Programming Language Interface) ซึ่งช่วยขยายความสามารถของภาษา Verilog โดยจะประกอบด้วยวิธีการที่จะเพิ่มเติมฟังก์ชันต่าง ๆ ขึ้นมาหรือการติดต่อกับภาษาอื่นเป็นต้น

Industrial support: ภาษา Verilog ได้รับการต้อนรับเป็นอย่างดีจากผู้ออกแบบวงจรประเภทASIC (Application specific circuit) เนื่องจากเป็นภาษาที่ง่ายในการเรียนรู้ และช่วยให้การจำลองการทำงานหรือตรวจสอบระบบทำได้อย่างรวดเร็วจากข้อมูลของนิตยสาร EE Timesได้พูดถึงเกี่ยวกับการใช้งานภาษา Verilog ในการออกแบบ ASIC ในปี 1993 นั้นมีถึง 85%ของการออกแบบทั้งหมด

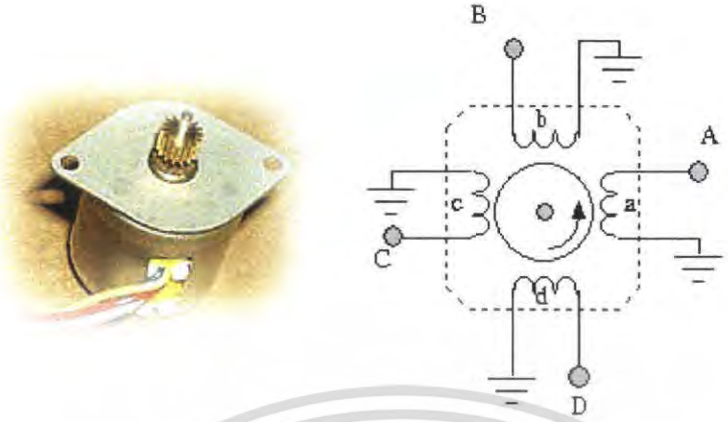
### 2.1.7 Verilog เปรียบเทียบกับ VHDL

ผู้ออกแบบหลาย ๆ คนมักตั้งคำถามกับตัวเองว่า แล้วภาษา HDL ภาษาไหนควรนำมาใช้ในการออกแบบดี VHDL หรือ Verilog คำตอบก็คือทั้งสองภาษาต่างมีข้อได้เปรียบเสียเปรียบหรือยากง่ายต่างกันแล้วแต่ความถนัดหรือความชอบของผู้ใช้งาน ดังนั้นเมื่อเปรียบเทียบกันทั้งสองภาษาถือว่าไม่มีผู้ชนะความได้เปรียบของภาษา Verilog คือความง่ายในการทำความเข้าใจและใช้งาน โครงสร้างไวยากรณ์เหมือนภาษา C และไม่จุกจิก ถือว่ามีความยืดหยุ่นในการเขียนมากกว่า VHDL ดังนั้นจึงค่อนข้างเป็นที่นิยมในการใช้งานในการออกแบบทั่วไปในอุตสาหกรรมวงจร โดยเฉพาะทางอเมริกา และญี่ปุ่น อย่างไรก็ตามภาษา Verilog ค่อนข้างจะด้อยในด้านความสามารถในการกำหนดการทำงานของระบบในระดับที่สูงขึ้น(System level specification)สำหรับภาษา VHDL จะซับซ้อนกว่า มีคุณสมบัติด้านต่าง ๆ (Feature) ที่เยอะกว่า (ซึ่งบางทีไม่จำเป็นในงานการออกแบบโดยทั่วไป) จึงมีไวยากรณ์ของภาษาหรือกฎต่าง ๆ มากกว่า ทำให้ค่อนข้างยากในการเรียนรู้ และใช้งานแต่ข้อดีคือมีความยืดหยุ่นในการใช้งานสูงเนื่องจากสามารถใช้ในการเขียนออกแบบรูปแบบต่าง ๆ มากมาย (Permissible coding styles) ดังนั้น VHDL จึงเหมาะสำหรับการออกแบบระบบที่ซับซ้อนทำให้ได้รับความนิยมมากจากนักออกแบบวงจรดิจิตอลโดยทั่วไปโดยเฉพาะแถบทางด้านยุโรป

### 2.1.8 การทำงานของโปรแกรม

เมื่อมีสัญญาณความถี่เข้ามา Counter จะทำการนับความถี่ Input ค่าที่นับจะถูกส่งให้ ROM เพื่อทำการเลือก Address ซึ่งภายใน Address ของ ROM แต่ละตัวจะมีบิตที่ใช้สร้างสัญญาณขับสเตปปิงมอเตอร์สัญญาณที่ได้ออกมาจากROMแต่ละตัวนั้นจะผ่านการเลือกสัญญาณโดยใช้ช่วงความถี่ Input เป็นเงื่อนไขในการเลือกโหมดเมื่อมอเตอร์เกิดการสลิป Current Sensor จะให้ค่าแรงดันที่ทำให้ FPGA เกิดการ ACTIVE เมื่อเกิดการ ACTIVE จะทำให้ Counter และ ROM จะไม่ทำงานทำให้ไม่มีการสร้างสัญญาณในการขับสเตปปิงมอเตอร์

## 2.2 สเตปปิงมอเตอร์




**รูปที่ 2.3** สเตปปิงมอเตอร์

สเตปปิงมอเตอร์ที่ใช้เป็นสเตปปิงมอเตอร์แบบ 4 เฟส 5 เส้น 24 V และมีกระแส 0.18 A ต่อเฟส และ 1.8 องศาต่อสเตป โครงสร้างของ Stepping Motor มีลักษณะดังรูป ซึ่งประกอบด้วยขดลวด stator 4 ขด ล้อมรอบแกนหมุนหลักการทำงาน คือเมื่อจ่ายกระแสไฟฟ้าให้กับขดลวด Stator Coil a, b, c, d ไม่พร้อมกันนั่นคือ ถ้าเราจ่ายกระแสให้ a ก่อนโดยไม่จ่ายให้ขดอื่นแล้วตามด้วย b, c และ d เรียงตามลำดับจะทำให้เกิดสนามแม่เหล็ก หมุนวนในลักษณะทวนเข็มนาฬิกา ซึ่งส่วนของ Rotor ที่เป็นแม่เหล็กถาวรก็จะหมุนตามสนามแม่เหล็กไปด้วย คือทวนเข็มนาฬิกาในทำนองเดียวกัน ถ้าเราจ่ายกระแสให้ขด a, d, c, b, a ..... ก็จะให้สนามแม่เหล็กหมุนในทิศทางตามเข็มนาฬิกาซึ่งส่งผลให้ Rotor หมุนตามเข็มนาฬิกา ด้วยการกำหนดความเร็วของ Stepping Motor ทำได้โดยการเปลี่ยนแปลง ความเร็วของการเปลี่ยนการจ่ายกระแสจากขดลวดขดหนึ่งไปยังขดหนึ่งให้เร็วขึ้นอีก การกำหนดความเร็วของ Stepping Motor ทำได้โดยการเปลี่ยนแปลงความเร็วของ การเปลี่ยนการจ่ายกระแสจากขดลวดขดหนึ่งไปยังอีกขดหนึ่งให้เร็วขึ้น

### 2.2.1 สเต็ปมอเตอร์ที่พบในปัจจุบัน

#### 2.2.1.1 แบบแม่เหล็กถาวร(PERMANENT MAGNET_PM)

สเต็ปปิงมอเตอร์แบบ PM จะมีสเตเตอร์ (STATOR) ที่พันขดลวดไว้หลาย ๆ โพล โดยมีโรเตอร์ (ROTOR) เป็นรูปทรงกระบอกฟันเลื่อยและโรเตอร์ทำด้วยแม่เหล็กถาวรเพื่อป้อนไฟกระแสตรงให้กับขดสเตเตอร์จะทำให้เกิดแรงแม่เหล็กไฟฟ้าผลักต่อโรเตอร์ทำให้มอเตอร์หมุน มอเตอร์แบบ PM จะเกิดแรงฉุดยึดให้โรเตอร์หยุดอยู่กับที่แม้จะไม่ได้ป้อนไฟเข้าขดลวด

**2.2.1.2 แบบแปรค่ารีลักแตนซ์ (VARIABLE RELUCTANCE- VR)**

สเต็ปปิงมอเตอร์แบบ VR จะมีการหมุนโรเตอร์ได้อย่างอิสระแม้จะไม่ได้จ่ายไฟให้โรเตอร์ทำจากสารเฟอร์โรแมกเนติกกำลังอ่อนมีลักษณะเป็นฟันเลื่อยรูปทรงกระบอกโดยจะมีความสัมพันธ์โดยตรงกับจำนวนโพลในสเตเตอร์แรงบิดที่เกิดขึ้นจะไปหมุนโรเตอร์ไปในเส้นทางของอำนาจแม่เหล็กที่มีค่ารีลักแตนท์ต่ำที่สุดตำแหน่งที่จะเกิดแน่นอนและมีเสถียรภาพแต่จะเกิดขึ้นได้หลาย ๆ จุดดังนั้นเมื่อป้อนไฟเข้าขดลวดต่าง ๆ ในมอเตอร์แตกต่างขดกันไปก็ทำให้มอเตอร์หมุนไปตำแหน่งต่าง ๆ กันโรเตอร์ของ VR จะมีความเฉื่อยของโรเตอร์น้อยจึงมีความเร็วรอบสูงกว่ามอเตอร์แบบ PM

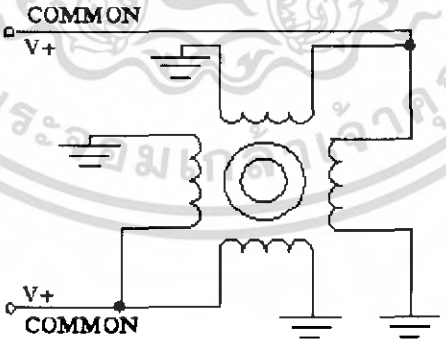
**2.2.1.3 แบบผสม(HYBRID-H)**

สเต็ปปิงมอเตอร์แบบ H จะเป็นลูกผสมของ VR กับ PM โดยจะมีสเตเตอร์คล้ายกับที่ใช้ใน VR โรเตอร์มีหมวกหุ้มปลายซึ่งมีลักษณะของสารแม่เหล็กที่มีกำลังสูงโดยการควบคุมขนาดรูปร่างของหมวกแม่เหล็กอย่างดีทำให้ได้มุมการหมุนและครั้งน้อยและแม่นยำข้อดีก็คือให้แรงบิดสูงและมีขนาดกระทัดรัดและให้แรงฉุดยึดโรเตอร์นิ่งกับที่ตอนไม่จ่ายไฟ

**2.2.2 การตรวจสอบหาสาย COMMON และ สาย GROUND ของ STEPPING**

**2.2.2.1. ชนิดที่เป็น COMMON ภายนอก Stepping Motor**

แบบนี้มีสายอยู่ 6 เส้น คือ 2 เส้นเป็น สาย COMMON และอีก 4 เส้นเป็นสาย GROUND ดังรูปที่ 2.4

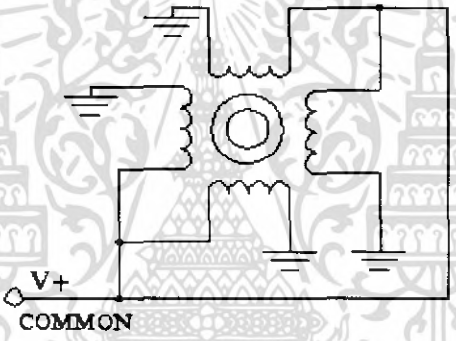



**รูปที่ 2.4** โครงสร้างและสายต่อที่มี 6 เส้น

จากรูปที่ 2.4 สาย COMMON 1 เส้นจะต่อร่วมกับสาย GROUND 2 เส้นซึ่งในการตรวจเช็คจะใช้โอห์มมิเตอร์วัดหาสายที่เป็น COMMON ก่อนโดยการตั้งย่านการวัดของมิเตอร์ที่ Rx1 จับที่สายทีละคู่ ซึ่งถ้าหากทำการวัดสาย COMMON เทียบกับสาย GROUND ได้คู่ที่ถูกต้องค่าความต้านทานที่อ่านได้จะน้อย แต่ถ้าวัดผิดสายคือวัดสาย GROUND เทียบกับ GROUND ค่าความต้านทานที่อ่านได้จะสูงกว่าแต่ถ้าวัดสาย COMMON เทียบกับสาย GROUND ที่ไม่ใช่คู่กันแล้ว เข็มมิเตอร์ก็จะไม่กระดิกให้ทดลองวัดเปรียบเทียบกันทีละคู่ก็จะทราบว่าสายใดเป็นสาย COMMON สายใดเป็นสาย GROUND

#### 2.2.2.2 ชนิดที่เป็น COMMON ภายใน Stepping Motor

แบบนี้มีสายอยู่ 5 เส้นคือ 1 เส้นเป็นสาย COMMON และอีก 4 เส้นเป็นสาย GROUND ดังรูปที่ 2.5




**รูปที่ 2.5** โครงสร้างและสายต่อที่มี 5เส้น

ในการวัดให้ทำแบบเดียวกับการวัด Stepping Motor ชนิด COMMON ภายนอกแตกต่างกันเพียงแบบ COMMON ภายในสาย COMMON 1 เส้นต่อร่วมกับสาย GROUND 4 เส้นดังนั้นหากสายเส้นใดเมื่อวัดเทียบกับสายเส้นอื่นแล้วมีค่าความต้านทานน้อยที่สุดสายเส้นนั้นเป็นสาย COMMON และที่เหลืออีก 4 เส้นจะเป็นสาย GROUND

### 2.2.3 การเรียงเฟสของ STEPPING MOTOR

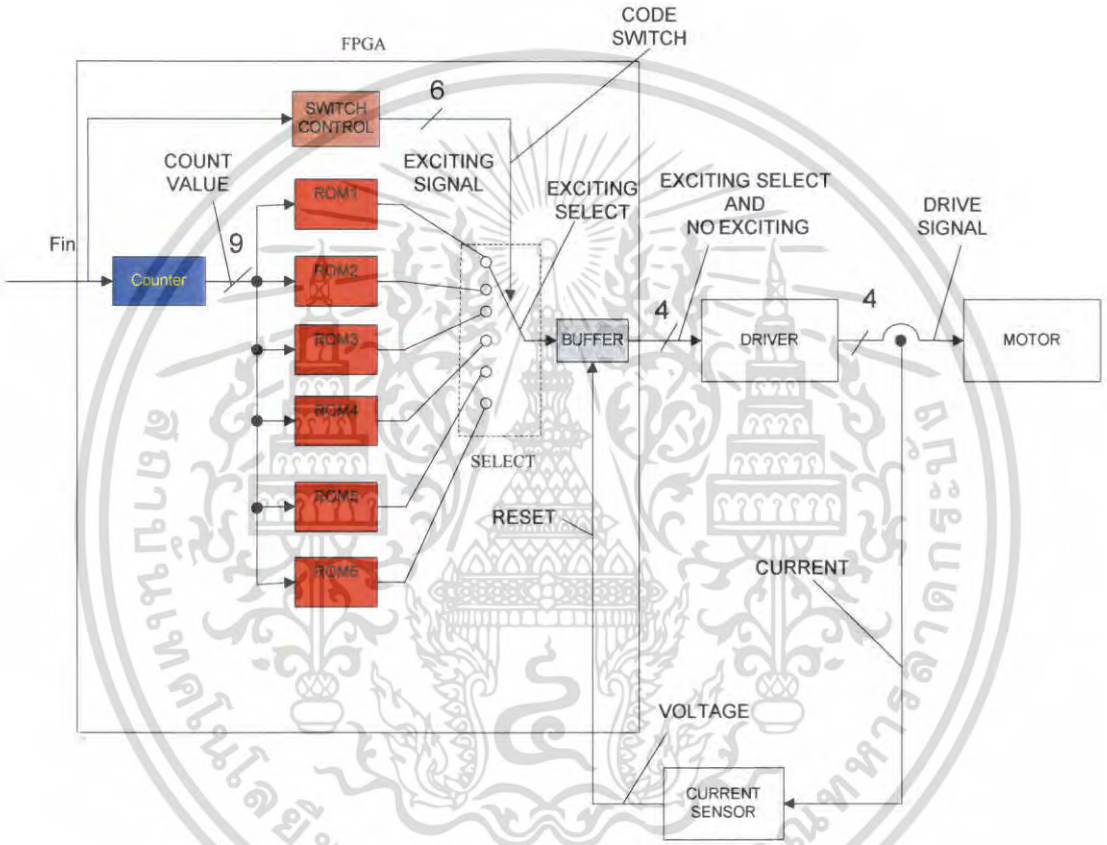
เมื่อเราทราบว่า สายเส้นใดเป็นสาย COMMON แล้วแต่เรายังไม่ทราบว่าสาย GROUND เส้นใดเป็นเฟสที่ 1 เฟสที่ 2 เฟสที่ 3 และเฟสที่ 4 ในการเรียงเฟสนั้นให้ใช้โอห์มมิเตอร์วัดโดยนำสาย V+ เข้าที่สาย COMMON วัดเทียบกับสาย GROUND เส้นใดก็ได้ 1 เส้นจะทำให้แกนโรเตอร์เคลื่อนไปข้างหน้า 1 STEPเมื่อเปลี่ยนสาย GROUND เส้นแรกเป็นเส้นที่ 2 ลาก MOTOR ไม่เคลื่อนที่ไปข้างหน้าแสดงว่าการเรียงเฟสไม่ถูกต้องก็ให้วัดเทียบกับสาย GROUND เส้นใหม่ต่อไป หาก MOTOR เคลื่อนที่ไปข้างหน้าตามกันวัดที่สาย GROUND เส้นต่อไปเรื่อย ๆ ก็จะทำให้ทราบว่าสายเส้นใดเป็นเฟสแรกสายเส้นใดเป็นเฟสที่ 2 เฟสที่ 3 และเฟสที่ 4 การเรียงเฟสของ Stepping Motor แบบ PM ทั้งชนิดที่เป็น COMMON ภายนอกและชนิดที่เป็น COMMON ภายในใช้หลักการเดียวกัน

## 2.3 Current Sensor

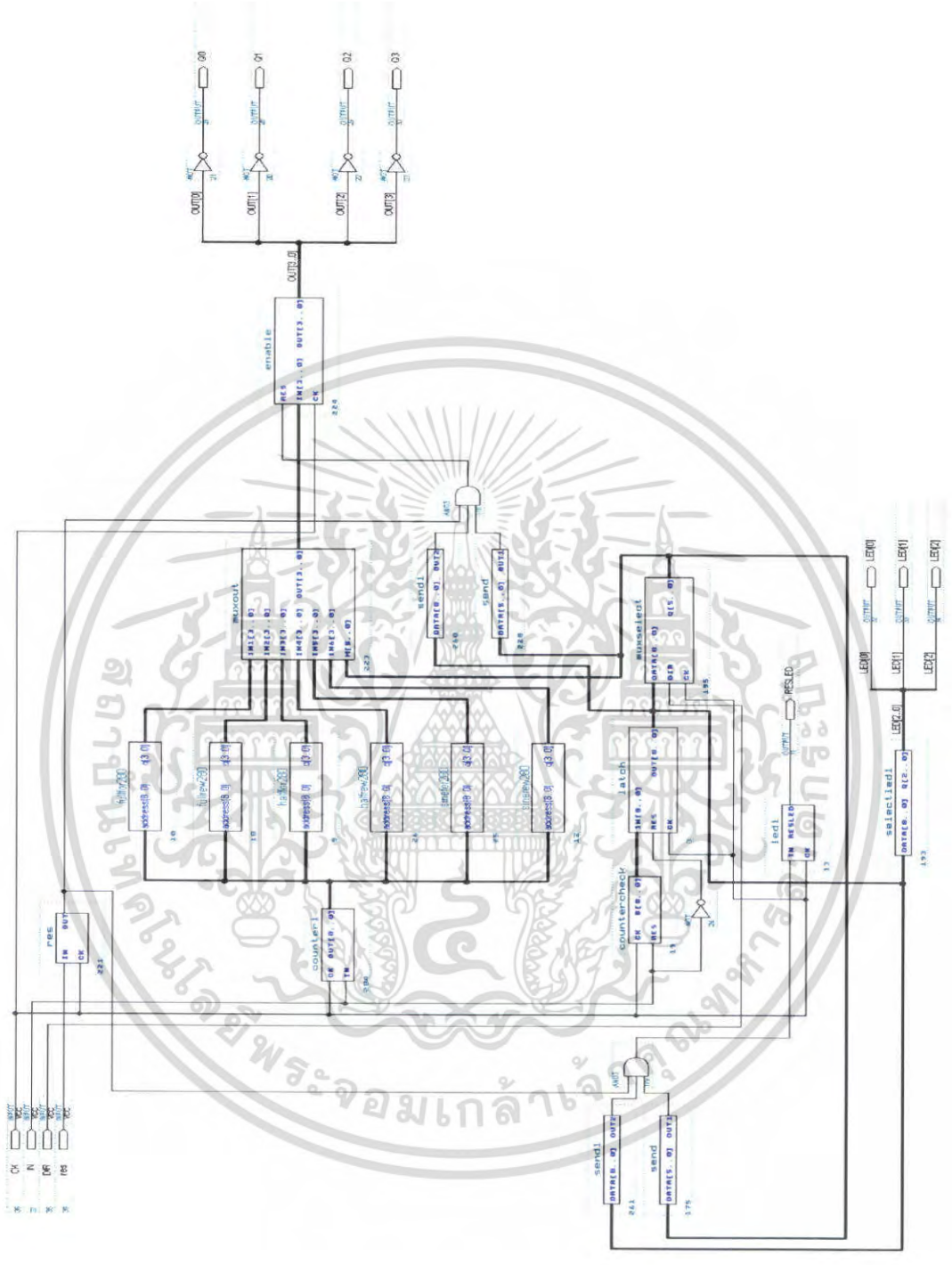
ในการขับสเตปปิงมอเตอร์เมื่อมอเตอร์เกิดการ Slip จะทำให้ไม่มีแรงดันย้อนกลับเกิดขึ้น ทำให้กระแสที่มาจากแหล่งจ่ายมีค่าสูงซึ่งเมื่อสเตปปิงมอเตอร์รับกระแสมาก ๆ จึงมีผลทำให้เกิดความร้อนและเมื่อมีความร้อนสะสมมาก ๆ จะทำให้มอเตอร์ไหม้เกิดความเสียหายได้ดังนั้นเราจึงมี Current Sensor ไว้ตรวจสอบกระแสไฟฟ้า ถ้ามีกระแสมากก็ให้ทำการป้อนกลับไปยัง FPGA เพื่อทำการ Reset สัญญาณที่จ่ายเข้ามาเป็นการป้องกันความเสียหายของมอเตอร์

# บทที่ 3
# หลักการและการออกแบบ

ในโครงงานจะใช้ FPGA เป็นตัวประมวลผลสัญญาณและสร้างวงจรดิจิตอลต่าง ๆ เพื่อเป็นสัญญาณป้อนต่อไปยังตัวขับเคลื่อน(Driver) บล็อกไดอะแกรมของระบบโดยรวมแสดงดังรูปที่ 3.1




**รูปที่ 3.1** บล็อคแสดงระบบโดยรวมของโครงงาน

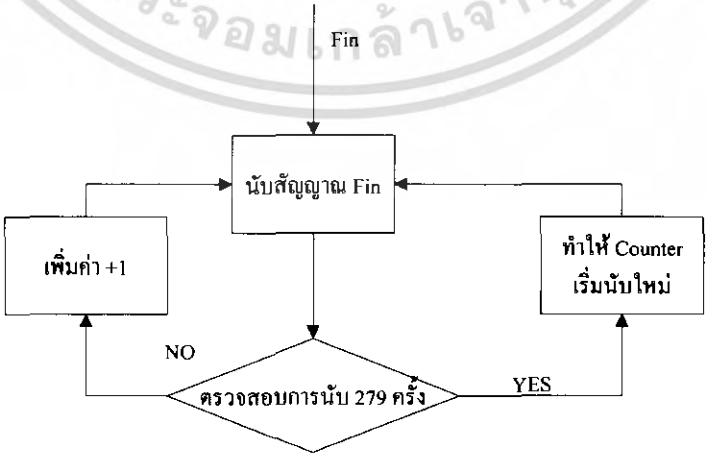


**รูปที่ 3.2** โปรแกรมที่ใช้สร้างสัญญาณขับสเตปปิงมอเตอร์

จากรูปที่ 3.1 เมื่อป้อนความถี่ค่า(Fin) Counter จะทำการนับสัญญาณความถี่ (Fin) และนำค่าที่นับได้ไปทำการเลือก Address ใน ROM ซึ่งภายใน ROM จะมีข้อมูลดิจิตอลที่ใช้สำหรับสร้างรูปแบบสัญญาณในการขับสเตปปิงมอเตอร์ 3 โหมด คือ Full step Half step และ Sinusoidal mini step ซึ่งจะให้ผลตอบสนองทางความเร็วที่แตกต่างกันในความถี่ที่เท่ากัน ดังนั้นจึงทำการจัดเรียงบิตภายใน ROM เพื่อทำให้ในขณะเปลี่ยนโหมดความเร็วในการหมุนของสเตปปิงมอเตอร์จะสามารถหมุนได้อย่างต่อเนื่องสัญญาณที่ได้จากROM (EXCITING SIGNAL) นี้จะถูกเลือกโหมดสัญญาณที่ใช้ในการขับสเตปปิงมอเตอร์ (EXCITING SELECT) โดยเลือกจากกำหนดย่านความถี่ (Fin) ในชุดของบล็อก SWITCH CONTROL ซึ่งแต่ละโหมดจะใช้ย่านความถี่ที่แตกต่างกันซึ่งจะได้สัญญาณที่เป็นบิต (CODE SWITCH) หลังจากนั้นบิตดังกล่าวจะผ่านบล็อก BUFFER เพื่อตรวจสอบว่ามีสัญญาณ RESET หรือไม่ถ้าไม่มีสัญญาณ RESET จะทำให้ EXCITING SIGNAL ที่เลือกไว้ออกที่ OUTPUT เมื่อมอเตอร์เกิดการสลิปจะใช้กระแสสูง (CURRENT) ที่สุดในโหมดนั้น CURRENT SENSOR จะให้ค่าแรงดัน (VOLTAGE) ที่ทำให้ FPGA เกิดการ ACTIVE และนำค่าแรงดันที่ได้มาตรวจสอบเงื่อนไขอื่น ๆ และส่งค่าไปยัง BUFFER เพื่อตัดสัญญาณขับสเตปปิงมอเตอร์ดังนั้นโปรแกรมที่ได้สร้างขึ้นจะได้ดังรูปที่ 3.2

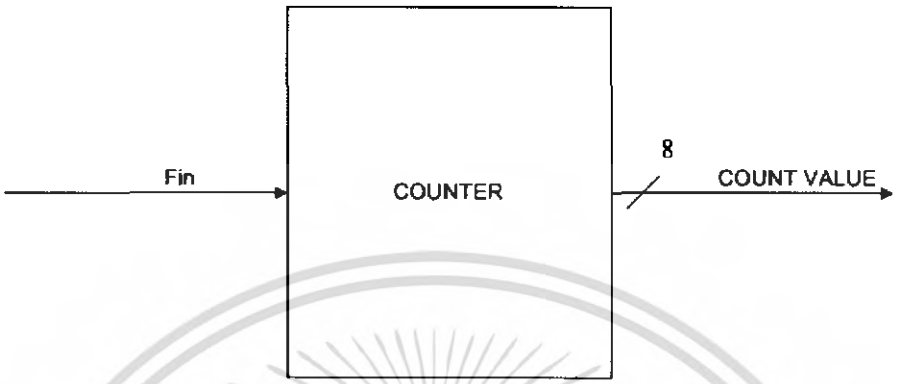
## 3.1 FPGA

ในส่วนของ FPGA ก็จะประกอบด้วยส่วนของวงจรดิจิตอลที่เขียนอยู่ในรูปของภาษา Verilog HDL เพื่อทำหน้าที่เป็นอุปกรณ์ที่ใช้ในการสร้างสัญญาณขับสเตปปิงมอเตอร์ซึ่งประกอบด้วยวงจรนับ (Counter) วงจรนับที่ออกแบบไว้เป็นการนับสัญญาณความถี่ซึ่งจะทำการนับตามเงื่อนไขถ้าจำนวนบิตที่ได้ยังไม่ถึง 279 ในฐานสิบให้ทำการนับจนถึงค่าที่กำหนดตามเงื่อนไขเมื่อได้ค่าตามเงื่อนไขให้ทำการเริ่มนับใหม่อีกครั้ง สามารถแสดงระบบโดยรวมของ Counter ได้ดังรูปที่ 3.3




**รูปที่ 3.3** แสดงระบบโดยรวมของ Counter

เมื่อทำการสร้างบล็อกไดอะแกรมระบบโดยรวมของ Counter เรียบร้อยแล้วให้นำบล็อคไดอะแกรมนี้มาออกแบบรูปร่างของ Counter โดยมี Fin และ COUNT VALUE เป็น INPUT และ OUTPUT ตามลำดับจะได้ดังรูปที่ 3.4



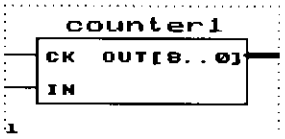

**รูปที่ 3.4** Counter

หลังจากนั้นจึงนำมาเขียนโปรแกรมด้วยภาษา Verilog HDL ตามที่ได้ออกแบบโครงสร้างของ Counter จะได้ดังรูปที่ 3.5

```
module counter1 (ck,in,out);
input ck,in;
output [8:0]out;
reg [8:0]out;
reg oin,ooin;
always @ (posedge ck)
begin
oin<=in;
ooin<=oin;
if(oin==1&&ooin==0)
begin
if(out<=278)
out=out+1;
else
out=0;
end
end
endmodule
```

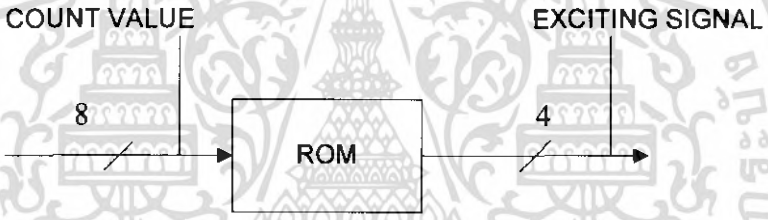
**รูปที่ 3.5** โปรแกรม Counter ด้วยภาษา Verilog HDL

เมื่อนำโปรแกรมมาสร้างบล็อกจะได้บล็อกดังรูปที่ 3.6



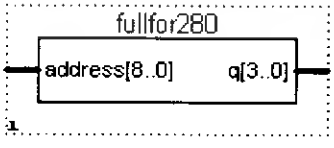

**รูปที่ 3.6** Counter ที่สร้างขึ้น

อุปกรณ์เก็บข้อมูล (ROM) ทำหน้าที่ในการเก็บค่าของบิตที่จัดเรียงไว้ในแต่ละสัญญาณโดยค่าของบิตเหล่านั้นถูกจัดเก็บไว้ภายใน Address ของ ROM แต่ละตัวเมื่อค่าที่ COUNTER นับจำนวน 8 บิต เข้ามาที่ ROM จะทำการเลือก Address ที่อยู่ภายใน ROM ซึ่งบิตที่ถูกเก็บไว้เหล่านั้นจะถูกส่งออกมาจำนวน 4 บิต ไปยัง OUTPUT ดังนั้นสามารถจึงทำการออกแบบ ROM ที่มี INPUT และ OUTPUT ตามต้องการได้ดังรูป 3.7

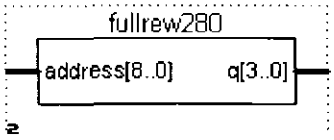



**รูปที่ 3.7** ROM

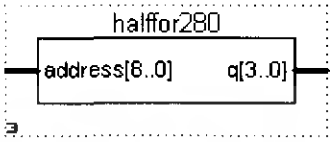
เมื่อทำการออกแบบ ROM ให้ได้จำนวน INPUT และ OUTPUT ตามต้องการแล้วหลังจากนั้นจึงทำการเขียนโปรแกรมเพื่อเก็บบิตของแต่ละโหมดภายใน ROM ซึ่งในโปรแกรมนี้กำหนดให้ ROM มี Address 9 บิต และมี OUTPUT 4 บิตสัญญาณที่ใช้ในการขับสเตปปิงมอเตอร์มีทั้งหมด 6 สัญญาณดังรูป ซึ่งโปรแกรมอ้างอิงได้จากภาคผนวก



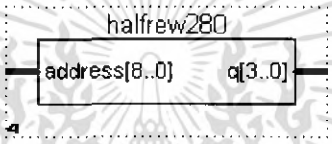

**รูปที่ 3.8** ROM ที่เก็บบิต Full step แบบ Forward ในการขับสเตปปิงมอเตอร์



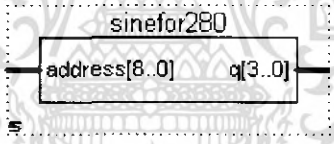

**รูปที่ 3.9** ROM ที่เก็บบิต Full step แบบ Reverse ในการขับสเตปปิงมอเตอร์



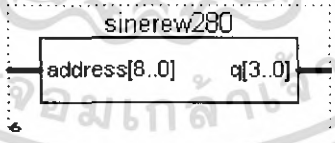

**รูปที่ 3.10** ROM ที่เก็บบิต Half step แบบ Forward ในการขับสเตปปิงมอเตอร์




**รูปที่ 3.11** ROM ที่เก็บบิต Half step แบบ Reverse ในการขับสเตปปิงมอเตอร์



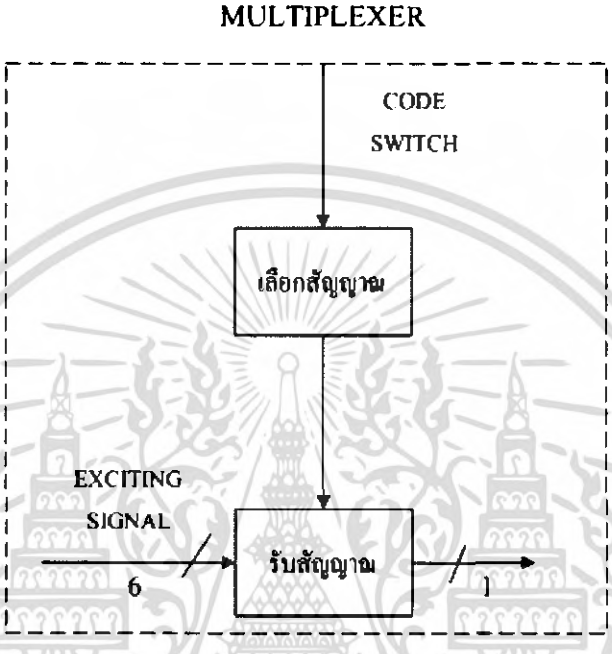

**รูปที่ 3.12** ROM ที่เก็บบิต Sinusoidal mini step แบบ Forward ในการขับสเตปปิงมอเตอร์




**รูปที่ 3.13** ROM ที่เก็บบิต Sinusoidal mini step แบบ Reverse ในการขับสเตปปิงมอเตอร์

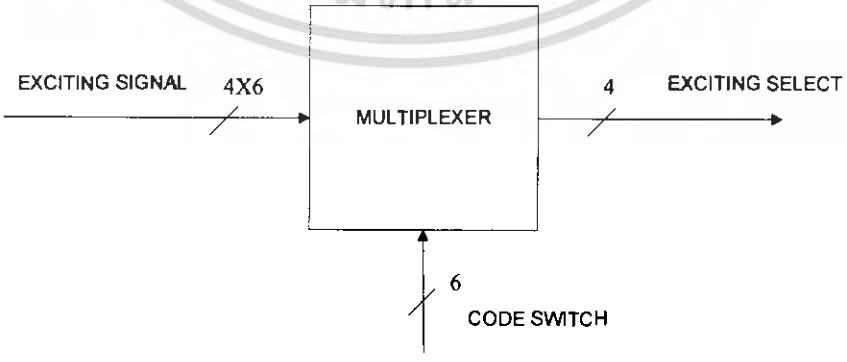
สัญญาณที่ทำการสร้างประกอบด้วย Full step แบบ Forward / Reverse, Half step แบบ Forward / Reverse และ Sinusoidal mini step แบบ Forward / Reverse ซึ่งโปรแกรมที่ใช้ในการสร้างสัญญาณนี้สามารถอ้างอิงได้จากภาคผนวก

บล็อก SELECT จากรูปที่ 3.1 จะทำหน้าที่เป็น MULTIPLEXER เพื่อเลือกรูปแบบสัญญาณที่ใช้ในการขับสเตปปิงมอเตอร์ซึ่งการทำงานจะรับบิตจำนวน 6 บิตจากชุดเลือกช่วงความถี่ของสัญญาณ (SWITCH CONTROL) ดังนั้นเมื่อได้รับบิตดังกล่าว MULTIPLEXER จะทำการเลือกสัญญาณที่ใช้ในการขับสเตปปิงมอเตอร์สามารถแสดงการทำงานของ MULTIPLEXER ได้ดังรูปที่ 3.9




**รูปที่ 3.14** ระบบของ MULTIPLEXER

เมื่อได้ทำการออกแบบระบบของ MULTIPLEXER เรียบร้อยแล้วหลังจากนั้นจะทำการออกแบบรูปร่างของ MULTIPLEXER ให้มี INPUT และ OUTPUT รวมทั้งจำนวนบิตของ CODE SWITCH ตามต้องการจะได้ดังรูปที่ 3.10



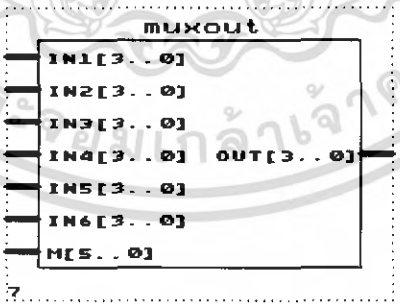

**รูปที่ 3.15** MULTIPLEXER

เมื่อทำการออกแบบรูปร่างของ MULTIPLEXER แล้วหลังจากนั้นให้ทำการเขียนโปรแกรมด้วยภาษา Verilog HDL จะได้ดังรูปที่ 3.11

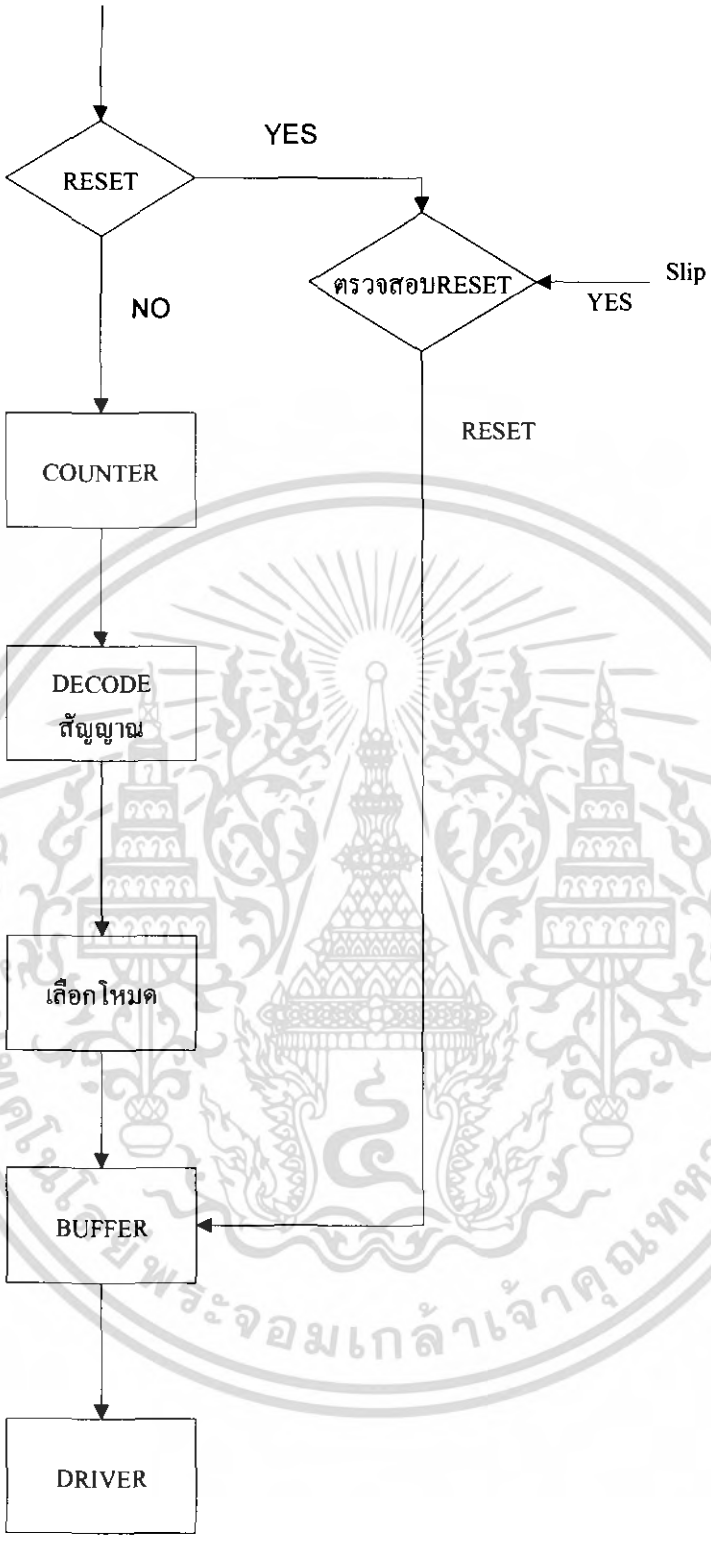
```
module muxout (out,in1,in2,in3,in4,in5,in6,m);
output [3:0]out;
input [5:0]m;
input [3:0]in1;
input [3:0]in2;
input [3:0]in3;
input [3:0]in4;
input [3:0]in5;
input [3:0]in6;
reg [3:0]out;
always
begin
case ({m})
6'b000001 :  out=in1;
6'b000010 :  out=in2;
6'b000100 :  out=in3;
6'b001000 :  out=in4;
6'b010000 :  out=in5;
6'b100000 :  out=in6;
default : out=in1;
endcase
end
endmodule
```

**รูปที่ 3.16** โปรแกรม MULTIPLEXER ด้วยภาษา Verilog HDL

เมื่อนำโปรแกรมมาสร้างบล็อกจะได้บล็อกดังรูปที่ 3.12




**รูปที่ 3.17** MULTIPLEXER ที่สร้างขึ้น




**รูปที่ 3.18 แสดงการขั้นตอนทำงานของโปรแกรม**

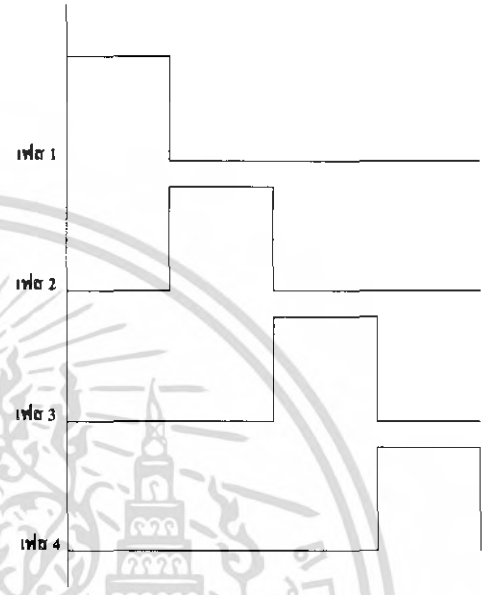
### 3.1.1 การสร้างสัญญาณ

โหมดที่ใช้ในการขับสเตปปิงมอเตอร์ในโครงงานนี้จะใช้สัญญาณขับทั้งหมดสามโหมดคือ Full step, Half step และ Sinusoidal mini step ซึ่งการขับปกติจะเป็นดังนี้

การขับแบบ Full step

**ตารางที่ 3.1** แสดงการขับแบบ Full step

| Step | เฟส (1) | เฟส (2) | เฟส (3) | เฟส (4) |
|---|---|---|---|---|
| Step1 | 1 | 0 | 0 | 0 |
| Step2 | 0 | 1 | 0 | 0 |
| Step3 | 0 | 0 | 1 | 0 |
| Step4 | 0 | 0 | 0 | 1 |

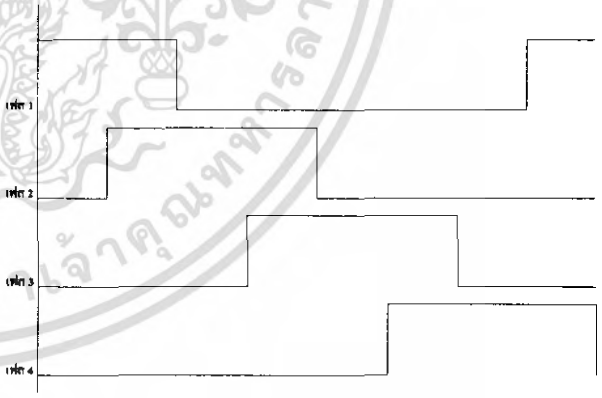



**รูปที่ 3.19** สัญญาณขับแบบ Full step

การขับแบบ Half step

**ตารางที่ 3.2** แสดงการขับแบบ Half step

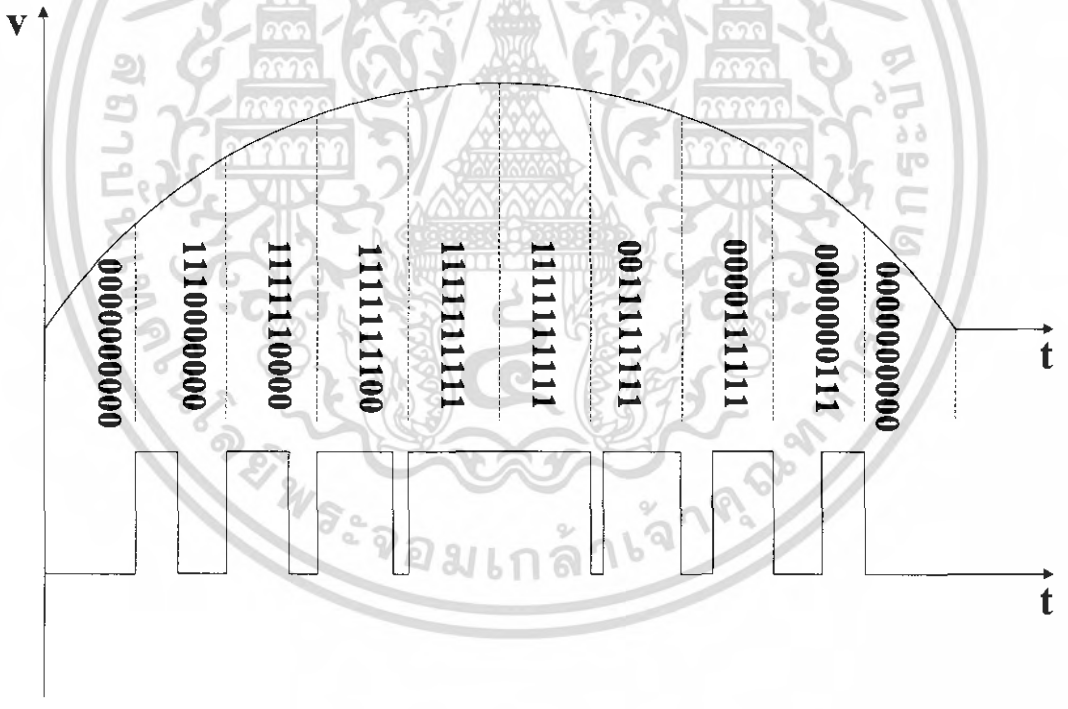
| Step | เฟส (1) | เฟส (2) | เฟส (3) | เฟส (4) |
|---|---|---|---|---|
| Step1 | 1 | 0 | 0 | 0 |
| Step2 | 1 | 1 | 0 | 0 |
| Step3 | 0 | 1 | 0 | 0 |
| Step4 | 0 | 1 | 1 | 0 |
| Step5 | 0 | 0 | 1 | 0 |
| Step6 | 0 | 0 | 1 | 1 |
| Step7 | 0 | 0 | 0 | 1 |
| Step8 | 1 | 0 | 0 | 1 |



**รูปที่ 3.20** สัญญาณขับแบบ Half step

ในรูปแบบของ Sinusoidal mini step จะแบ่ง ออกเป็นช่วงๆตามค่าองศาต่าง ๆ โดยเราจะใช้คลื่น Sine ช่วง 0-180 องศาเท่านั้นและขึ้นอยู่กับผู้ออกแบบว่าจะทำการแบ่งแต่ละช่วงออกเป็นกี่องศาตามความละเอียดที่เราต้องการ ซึ่งความละเอียดขององศาที่เราแบ่งจะทำให้เราทราบว่าควรที่ใช้จำนวนบิตที่แต่ละช่วงกี่บิต ในที่นี้เราแบ่งช่วงที่ใช้เป็นช่วงละ 20 องศา ดังนั้นทำให้เราต้องใช้บิตในแต่ละช่วงตั้งแต่ 8 บิตขึ้นไปเพราะถ้าใช้จำนวนบิตน้อยกว่านี้การเปลี่ยนแปลงแต่ละช่วงจะไม่เปลี่ยนแปลงทำให้สัญญาณที่จะนำไปขับไม่เป็นไปตามการขับแบบ Sinusoidal mini step และบิตที่ใช้ในครงงานนี้คือใช้ 10 บิตต่อการเปลี่ยนแปลงแต่ละช่วงองศาซึ่งหาได้ดังนี้

1. กำหนดความละเอียดในการแบ่งส่วนของสัญญาณ Sine คำนวณค่ามุมในแต่ละส่วน
2. นำค่า Sine ที่ได้ในแต่ละส่วนไปคิดเป็นเปอร์เซ็นและกำหนด Resolution ของสัญญาณ
3. กำหนดอัตรา logic “0” และ “1” ตามค่าเปอร์เซ็น
4. นำบิตข้อมูลที่ได้ไปเขียนโปรแกรมแล้วเก็บไว้ใน ROM เพื่อที่จะสามารถดึงข้อมูลออกมาใช้งานได้



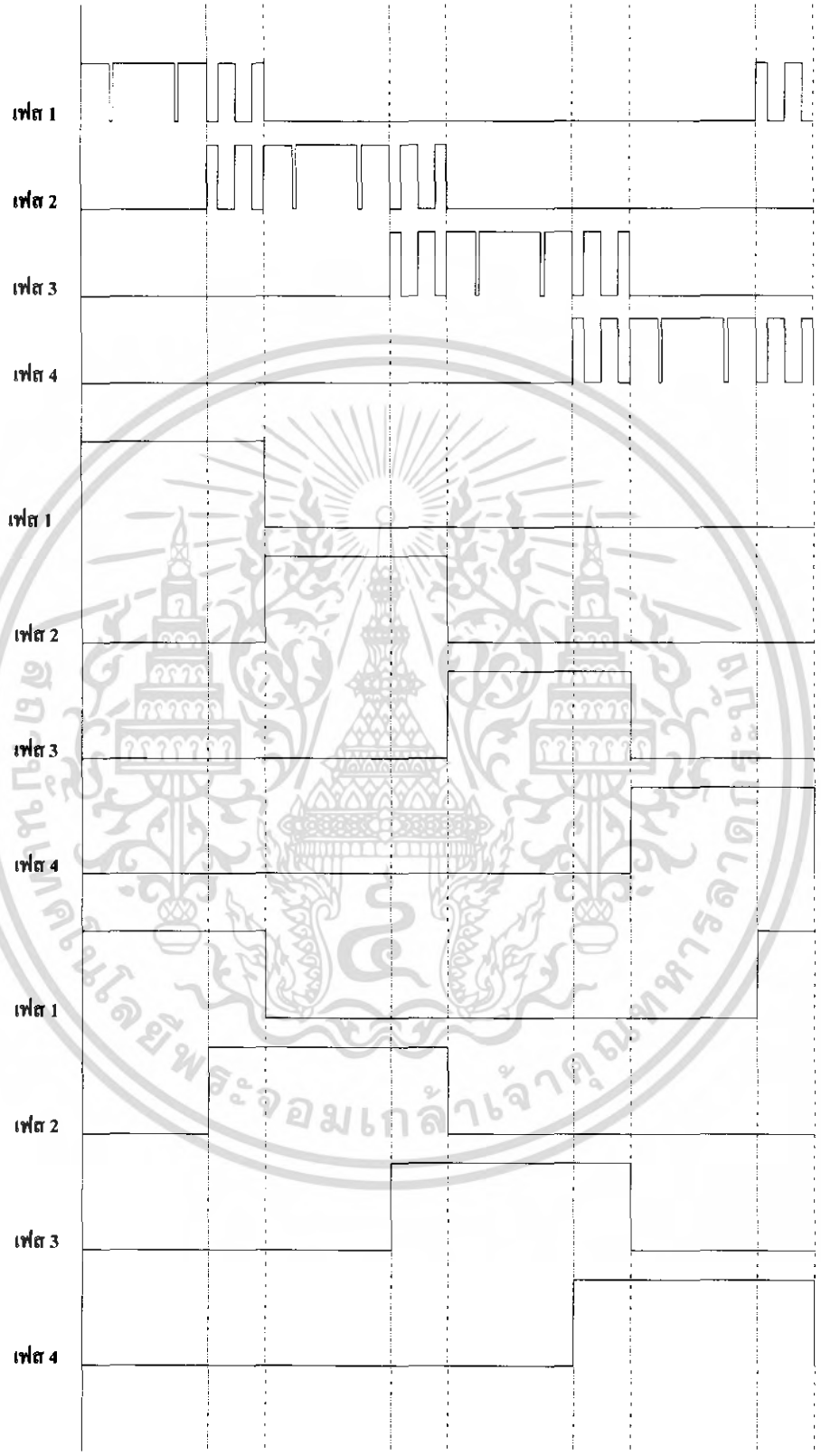

**รูปที่ 3.21** รูปแสดงการสร้างสัญญาณ Sinusoidal mini step

| ขั้นตอนที่ 1 | ขั้นตอนที่ 3 |
|---|---|
| Sine 0 = 0 | 10*0/100 = 0 บิต |
| Sine 20 = 0.3420 | 10*34.20/100 = 3.420 ≈ 3 บิต |
| Sine 40 = 0.6427 | 10*64.27/100 = 6.427 ≈ 6 บิต |
| Sine 60 = 0.8660 | 10*86.60/100 = 8.660 ≈ 8 บิต |
| Sine 80 = 0.9848 | 10*98.48/100 = 9.848 ≈ 10 บิต |
| Sine 100 = 0.9848 | 10*98.48/100 = 9.848 ≈ 10 บิต |
| Sine 120 = 0.8660 | 10*86.60/100 = 8.660 ≈ 8 บิต |
| Sine 140 = 0.6427 | 10*64.27/100 = 6.427 ≈ 6 บิต |
| Sine 160 = 0.3420 | 10*34.20/100 = 3.420 ≈ 3 บิต |
| Sine 180 = 0 | 10*0/100 = 0 บิต |

| ขั้นตอนที่ 2 | ได้บิตข้อมูลเป็นดังนี้ |
|---|---|
| 0 x 100 = 0 % | 0000000000 |
| 0.3420 x 100 = 34.20 % | 0000000111 |
| 0.6427 x 100 = 64.27 % | 0000111111 |
| 0.8660 x 100 = 86.60 % | 0011111111 |
| 0.9848 x 100 = 98.48 ≈ 100 % | 1111111111 |
| 0.9848 x 100 = 98.48 ≈ 100 % | 1111111111 |
| 0.8660 x 100 = 86.60 % | 0011111111 |
| 0.6427 x 100 = 64.27 % | 0000111111 |
| 0.3420 x 100 = 34.20 % | 0000000111 |
| 0 x 100 = 0 % | 0000000000 |

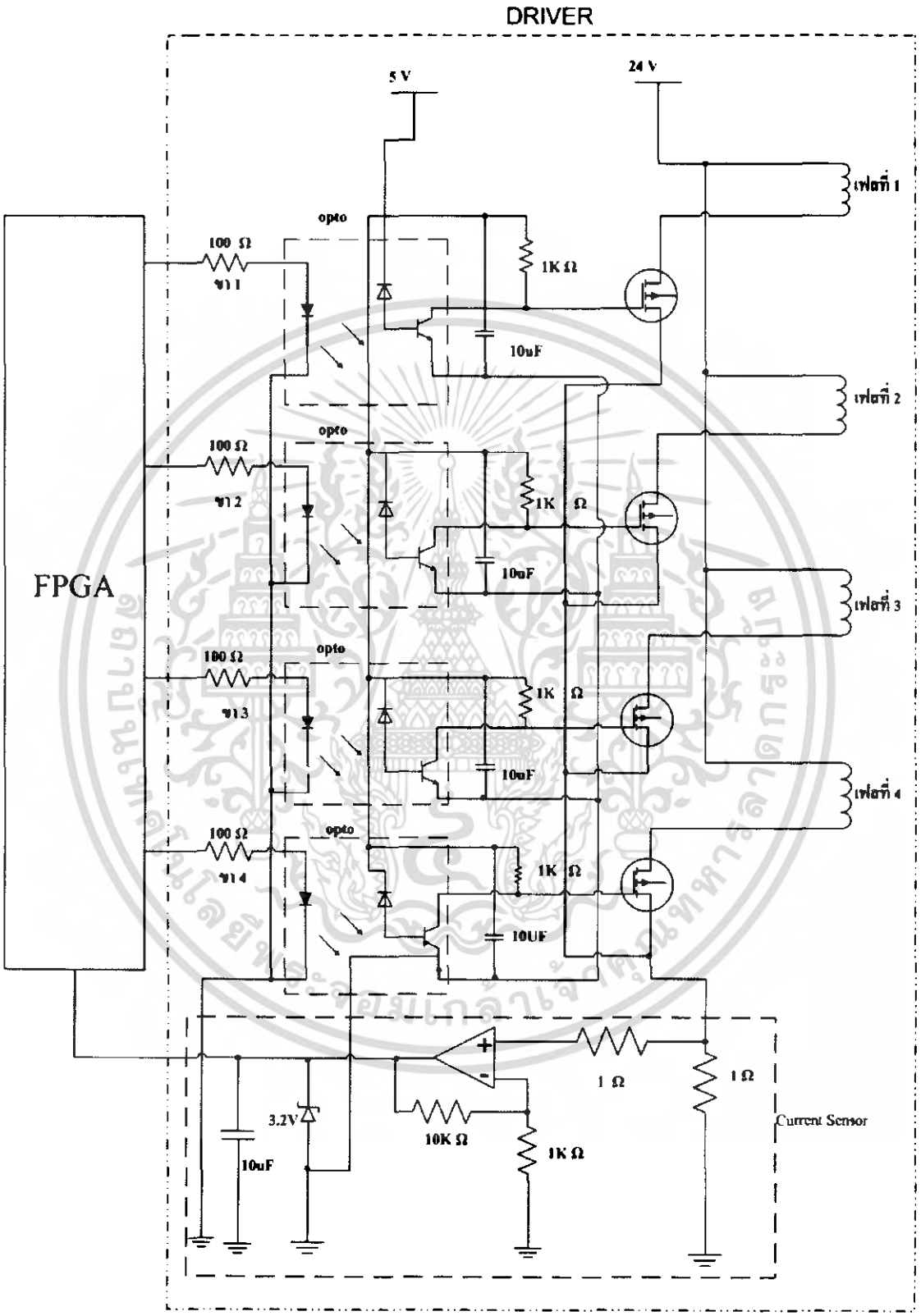
การอ่านบิตข้อมูลนั้นจะอ่านจากซ้ายไปขวาเสร็จแล้วอ่านลงมาเรื่อย ๆ จนครบข้อมูลทั้งหมดก็จะเหมือนกับการอ่านข้อมูลไปหนึ่ง step ของ Half step ของการขับหนึ่งเฟสแต่ที่เราต้องการคือให้ความเร็วรอบของมอเตอร์เท่ากันในทุก ๆ โหมดของการขับเราจึงต้องขยายจำนวนบิตของ Full step และ Half step ให้มีจำนวนบิตเท่ากับการขับแบบ Sinusoidal mini step คือจาก 16 บิตของ Full step และ 32 บิตของ Half step ให้มีขนาดเท่ากับ 100 บิตเท่ากับจำนวนบิตของ Sine เพราะ Sinusoidal mini step มีขนาดจำนวนบิตเท่ากับ 100 บิตแต่การกระตุ้นของโหมด Sine จะกระตุ้นที่ 60 องศาในตำแหน่งเริ่มต้นจึงใช้ข้อมูลในการขับแต่ละเฟสเท่ากับ 70 บิตแล้วนำข้อมูลมาเรียงจากซ้าย ๆ ไปขวาแล้วจึงนำค่าของเลขฐานสิบหกด้านขวาสุดมาเขียนโปรแกรมซึ่งลักษณะของสัญญาณจะแสดงได้ดังรูปที่ 3.22

ลักษณะสัญญาณที่ใช้งานเมื่อทำการจัดบิตแล้ว




**รูปที่ 3.22** ลักษณะสัญญาณเมื่อทำการจัดบิตแล้ว

### 3.1.2 บอร์ด DRIVER




**รูปที่ 3.23** วงจรขับสเตปปิงมอเตอร์

บอร์ด Driver จะรับเอาต์พุตจาก FPGA เข้ามาโดยผ่านอุปกรณ์ที่ทำหน้าที่เป็นสวิทช์ตัวแรก คือ Opto จะทำการแยก FPGA ซึ่งเป็นส่วนของการสร้างสัญญาณออกจากบอร์ด Driver ในส่วนของชุดขับกำลังหลังจากผ่าน Opto แล้วจะมีแรงดันที่ใช้สำหรับเปิดสวิทช์ตัวที่สอง ซึ่งจะเป็นสวิตช์ที่ใช้ในการขับสเตปปิงมอเตอร์โดยจะใช้มอสเฟตเป็นสวิทช์ในการขับ เมื่อสเตปปิงมอเตอร์เกิดการสลิปจะได้กระแสสูงสุดเมื่อผ่านชุด CURRENT SENSOR และให้ค่าแรงดันแก่ FPGA ซึ่งสามารถทำให้เกิดการตอบสนองได้มายัง FPGA วงจร Drive นั้นประกอบด้วยอุปกรณ์ดังนี้

R 100 Ω 4 ตัว

R 1K Ω 4 ตัว

C 0.1 $\mu F$ 4 ตัว

IRF 520N 4 ตัว

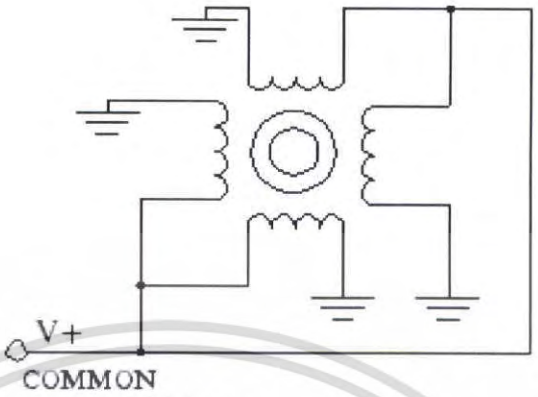
6N136 4 ตัว

### 3.1.3 สเตปปิงมอเตอร์

สเตปปิงมอเตอร์ที่ใช้ในการทดลองจะมีลักษณะดังรูปที่ 3.24 ซึ่งการทดลองของโครงงานนี้จะนำสเตปปิงมอเตอร์ชนิดที่มีสายอยู่จำนวน 5 เส้นคือ 1 เส้นเป็นสาย COMMON และอีก 4 เส้นเป็นสาย GROUND ดังรูปที่ 3.25 ซึ่งมีค่า Spec ของสเตปปิงมอเตอร์ตาม Name plate ที่ติดมาพร้อมกับสเตปปิงมอเตอร์จะเป็นดังนี้ คือ แรงดัน 24 V, กระแส 0.18 A และ Resolution 1.8 Degree/step ดังรูปที่ 3.26 ในการใช้งานเราจะทำการป้อนแรงดันเข้าที่สาย COMMON และใช้สายที่เหลือนำมาต่อผ่านสวิทช์เพื่อลง GROUND และทำการกระตุ้นตัวสวิทช์ตามเฟสที่เราต้องการ

**รูปที่ 3.24** สเตปปิงมอเตอร์ที่ใช้ทดลอง

โครงสร้างภายในของสเตปปิงมอเตอร์



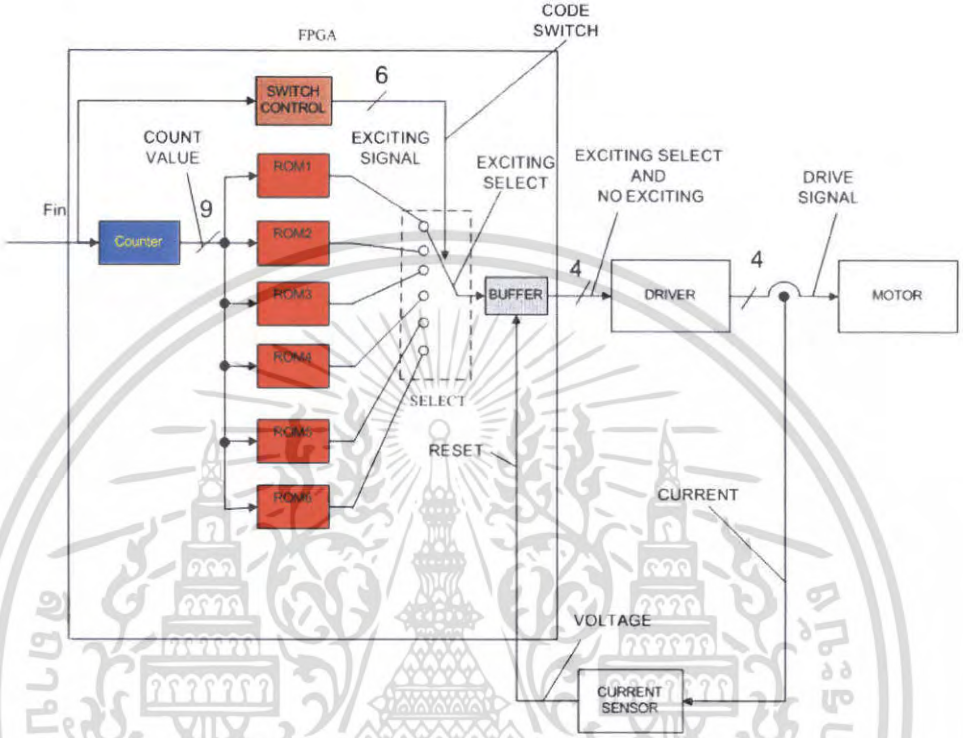

**รูปที่ 3.25** โครงสร้างของสเตปปิงมอเตอร์

Name plate ของสเตปปิงมอเตอร์


**รูปที่ 3.26** Name plate ของสเตปปิงมอเตอร์

# บทที่ 4

# ผลการทดลอง

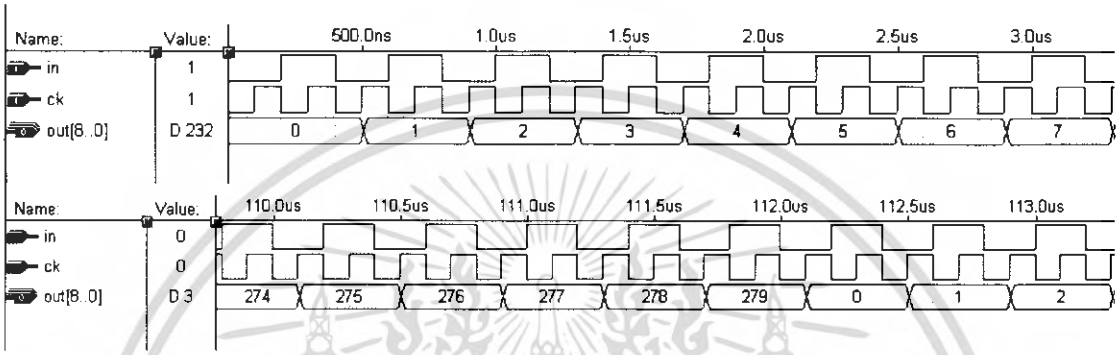



**รูปที่ 4.1** โครงสร้างของการสร้างสัญญาณขับสเตปปิงมอเตอร์

จากการทดลองเป็นการนำโครงสร้างของการสร้างสัญญาณสเตปปิงมอเตอร์มาทำการสร้างสัญญาณเพื่อขับสเตปปิงมอเตอร์ในลักษณะหลายโหมด

**รูปที่ 4.2** ภาพขณะทำการทดลอง

ทำการทดลองโดยการปรับความถี่ซึ่งโปรแกรมจะทำงานโดยเริ่มจาก Counter ทำการนับ PULSE ที่เข้ามาเพื่อจะนำค่าที่นับได้มาที่ROMซึ่งค่าที่นับได้จาก Counter จะต้องมีค่าไม่เกิน 279 เมื่อ Counter นับถึงค่าที่กำหนดคือ 279 จะทำการเริ่มต้นนับใหม่หลังจากนั้นค่าที่นับได้จาก Counter จะถูกนำมาชี้ตำแหน่งที่Addressใน ROM ซึ่งมีบิตสัญญาณที่ใช้ในการขับสเตปปิงมอเตอร์และสามารถแสดง WAVE FORM สัญญาณได้จากการ Simulation ในโปรแกรมเมื่อ PULSE เข้าไปใน Counter จะสามารถแสดงการนับได้ดังรูปที่ 4.3

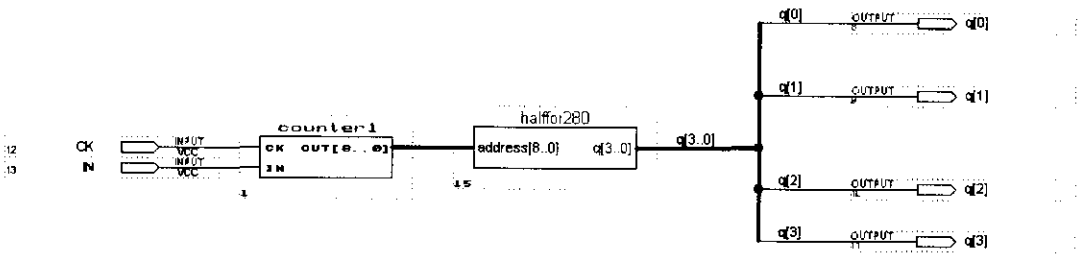



**รูปที่ 4.3** ภาพจำลองการทำงานของCOUNTER

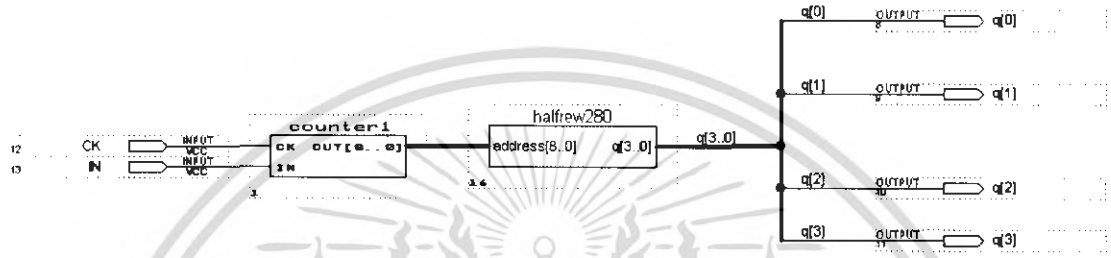
เมื่อทำการทดสอบสัญญาณภายใน ROM ของแต่ละโหมดจะสามารถทำการทดสอบได้ดังรูป


**รูปที่ 4.4** ทดสอบสัญญาณ Full step แบบ Forward

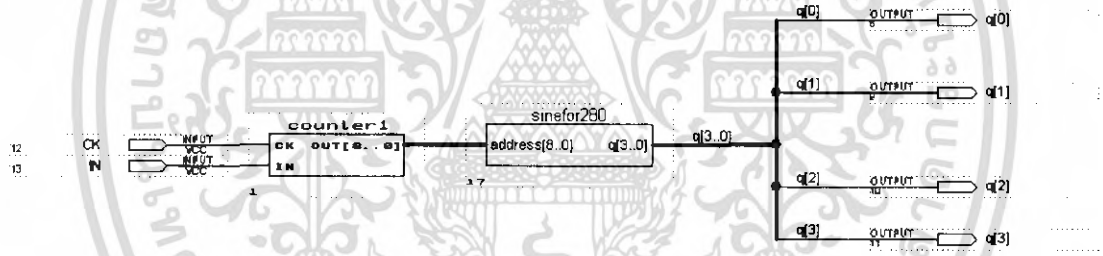

**รูปที่ 4.5** ทดสอบสัญญาณ Full step แบบ Reverse



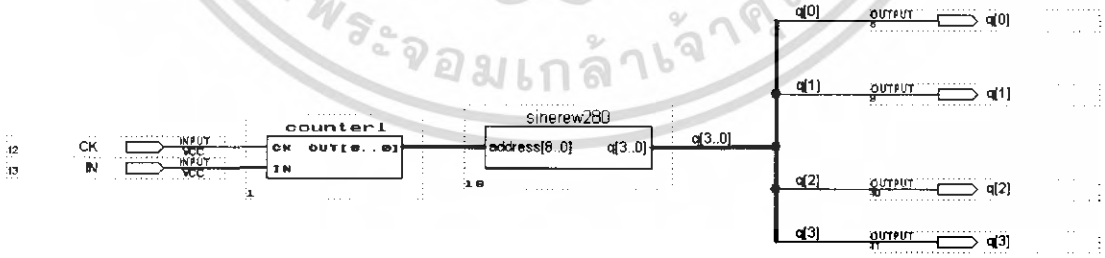

**รูปที่ 4.6** ทดสอบสัญญาณ Half step แบบ Forward




**รูปที่ 4.7** ทดสอบสัญญาณ Half step แบบ Reverse

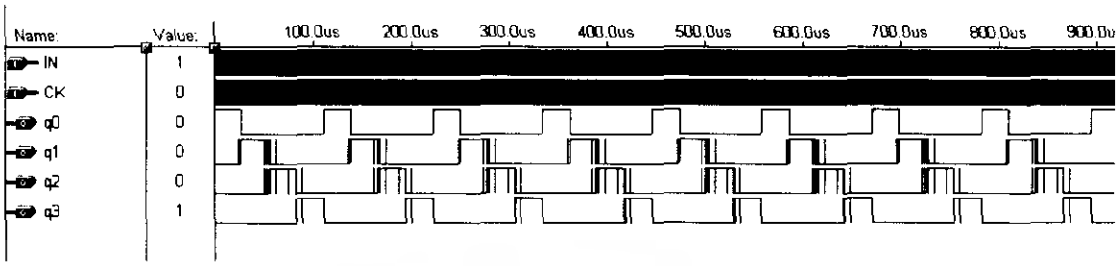



**รูปที่ 4.8** ทดสอบสัญญาณ Sinusoidal mini step แบบ Forward

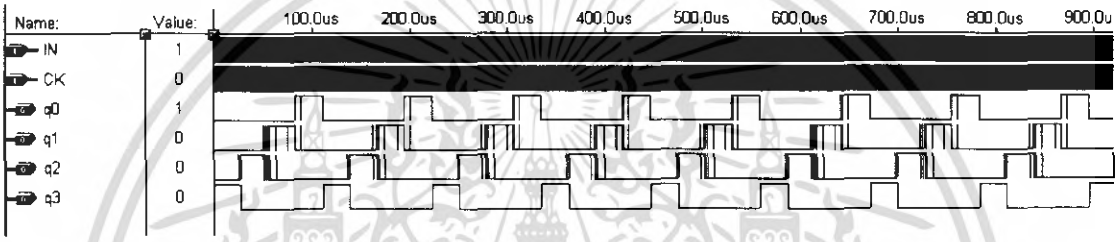



**รูปที่ 4.9** ทดสอบสัญญาณ Sinusoidal mini step แบบ Reverse

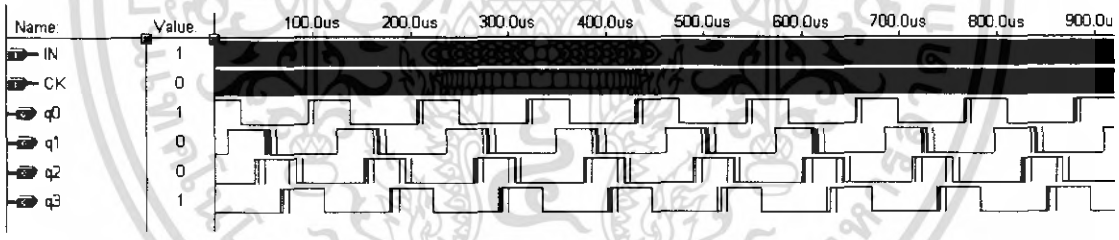
หลังจากนั้นทำการ Simulation สัญญาณในแต่ล่ะโหมดจะได้ WAVE FORM ของสัญญาณหลายรูปแบบที่ใช้ในการขับสเตปปิงมอเตอร์ดังรูป



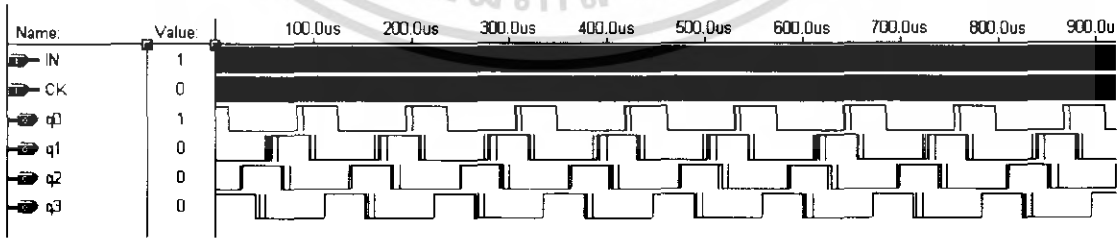

**รูปที่ 4.10** สัญญาณ Full step แบบ Forward



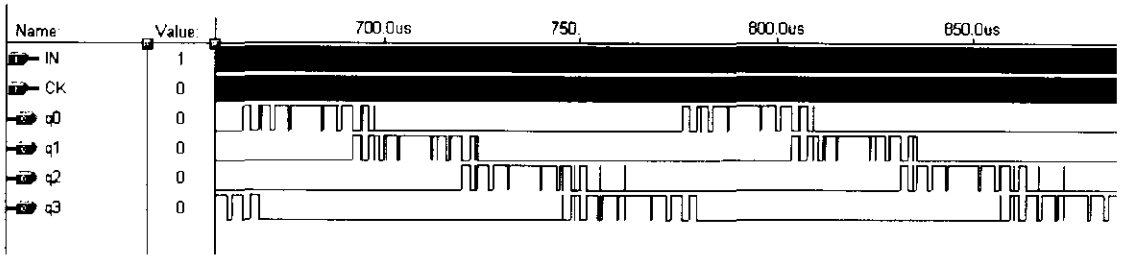

**รูปที่ 4.11** สัญญาณ Full step แบบ Reverse



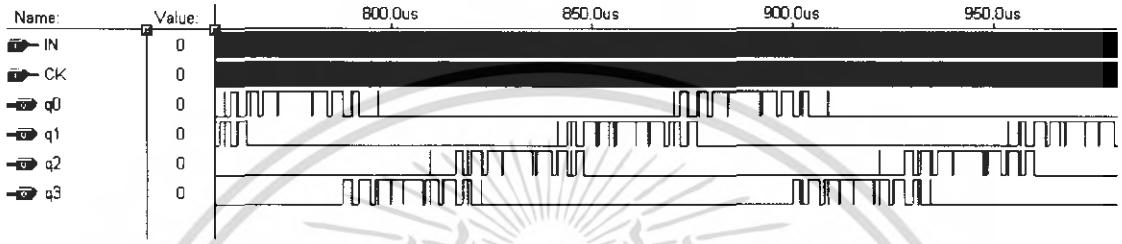

**รูปที่ 4.12** สัญญาณ Half step แบบ Forward




**รูปที่ 4.13** สัญญาณ Half step แบบ Reverse

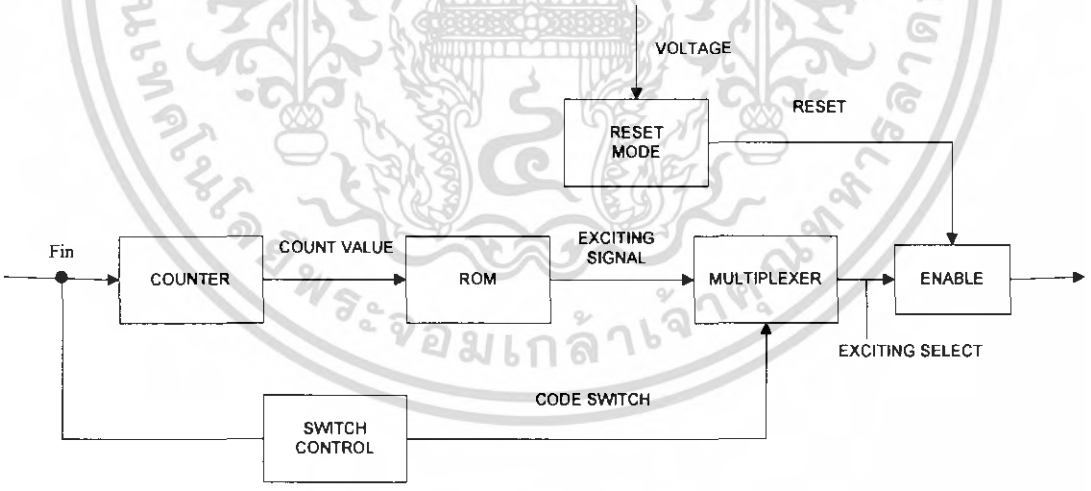



**รูปที่ 4.14** สัญญาณ Sinusoidal mini step แบบ Forword




**รูปที่ 4.15** สัญญาณ Sinusoidal mini step แบบ Reverse

สัญญาณที่ออกมาแต่ละโหมดจะถูกเลือกให้ออกไปทำการขับสเตปปิงมอเตอร์ซึ่งสามารถแสดงโครงสร้างบล็อกของระบบแสดงได้ดังนี้




**รูปที่ 4.16** โครงสร้างบล็อกของระบบ

ในการทดลองจะมีการขับอยู่ 3 โหมด คือ

Full Step

Half Step

Sinusoidal mini step

ผลการทดลองทางความเร็วของแต่ละโหมดแสดงได้ดังตารางที่ 4.1

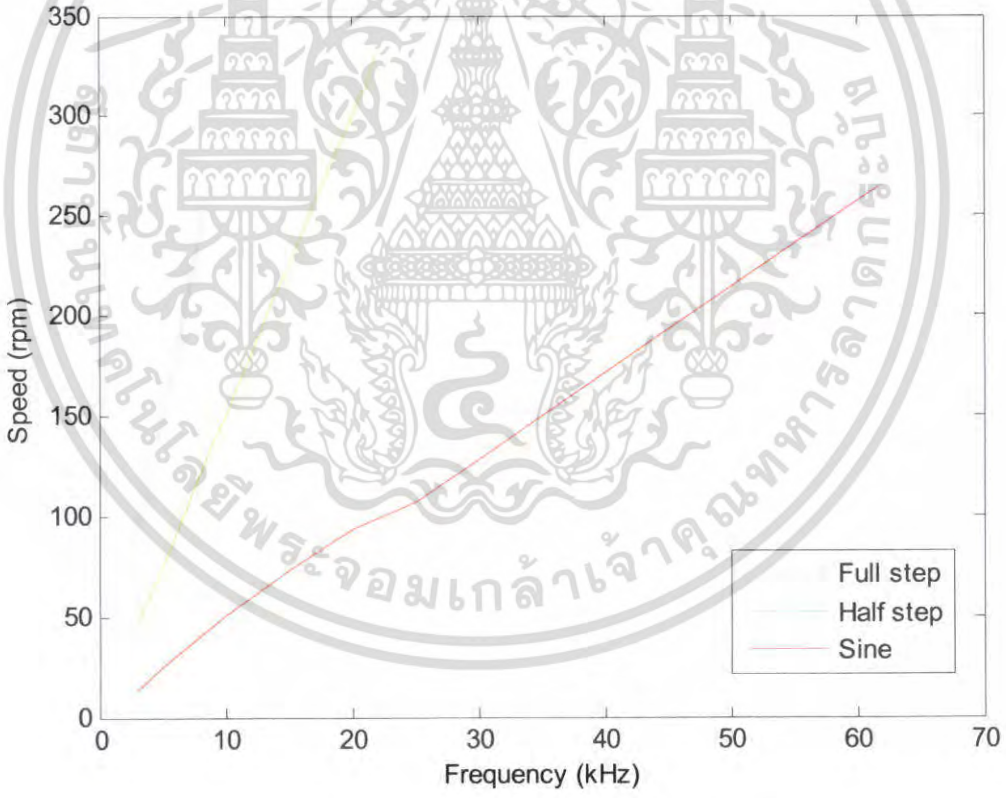
**ตารางที่ 4.1** แสดงความเร็วรอบของแต่ละโหมด

| ความถี่ Fin (Hz) | ความเร็วรอบ Full step (rpm) | ความเร็วรอบ Half step (rpm) | ความเร็วรอบ Sine (rpm) |
|---|---|---|---|
| 3000 | 89.5 | 45.1 | 12.7 |
| 4000 | 120.2 | 60.3 | 17.1 |
| 5000 | 150.2 | 74.9 | 25.5 |
| 6000 | 179.4 | 90 | 29.8 |
| 7000 | 208.5 | 105.3 | 34.5 |
| 8000 | 240 | 120.1 | 45.1 |
| 9000 | 269.4 | 135.4 | 47.3 |
| 10000 | | 150.1 | 51.7 |
| 11000 | | 165.1 | 55.5 |
| 12000 | | 179.7 | 60 |
| 13000 | | 194.7 | 64.1 |
| 14000 | | 209.8 | 68.6 |
| 15000 | | 225.7 | 73.4 |
| 16000 | | 239.8 | 77 |
| 17000 | | 255.8 | 81.2 |
| 18000 | | 269.3 | 85.7 |
| 19000 | | 284.5 | 89.9 |
| 20000 | | 299.8 | 94 |
| 21000 | | 313.8 | 98.7 |
| 22000 | | 329.6 | 102.7 |
| 23000 | | | 98.7 |
| 24000 | | | 102.7 |

| | | | |
|---|---|---|---|
| 25000 | | | 107 |
| 26000 | | | 111.7 |
| 27000 | | | 115.4 |
| 28000 | | | 120 |
| 29000 | | | 124.2 |
| 30000 | | | 128.6 |
| 31000 | | | 133.4 |
| 32000 | | | 136.6 |
| 33000 | | | 141.0 |
| 34000 | | | 145.4 |
| 35000 | | | 150.1 |
| 36000 | | | 154.9 |
| 37000 | | | 158.9 |
| 38000 | | | 163.1 |
| 39000 | | | 166.7 |
| 40000 | | | 171.7 |
| 41000 | | | 175.4 |
| 42000 | | | 179.5 |
| 43000 | | | 184.3 |
| 44000 | | | 188.9 |
| 45000 | | | 192.5 |
| 46000 | | | 197.3 |
| 47000 | | | 201.2 |
| 48000 | | | 205.2 |
| 49000 | | | 210.1 |
| 50000 | | | 214.5 |
| 51000 | | | 218.7 |
| 52000 | | | 222.5 |
| 53000 | | | 227.5 |
| 54000 | | | 231.3 |
| 55000 | | | 236.1 |

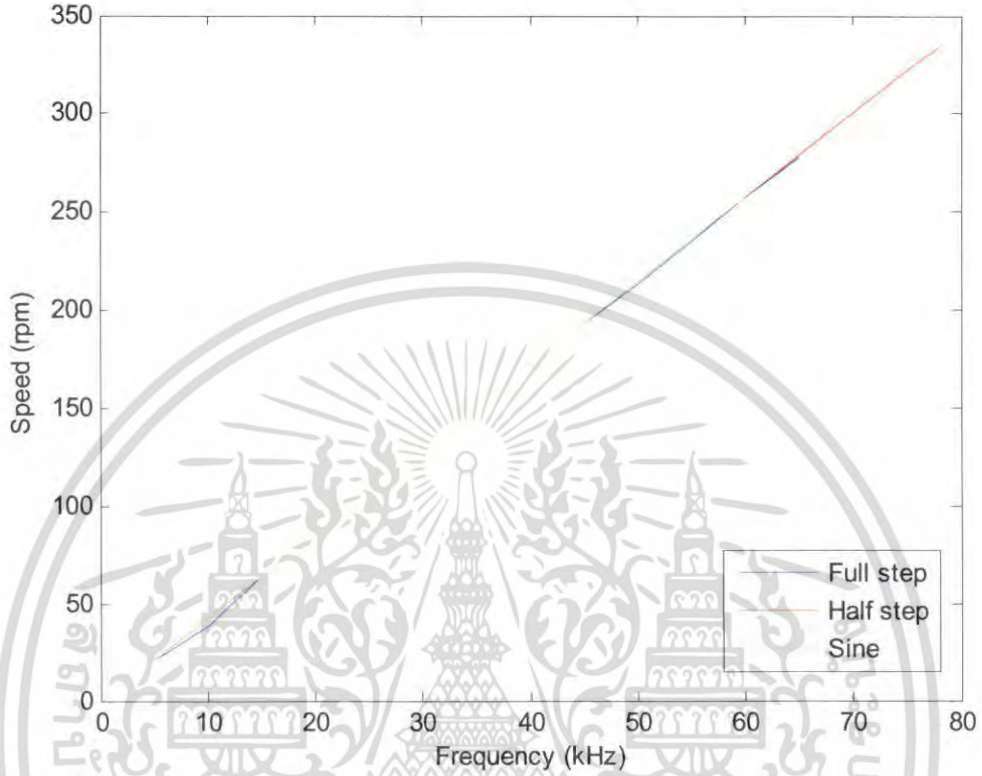
| | | | |
|---|---|---|---|
| 56000 | | | 239.9 |
| 57000 | | | 244.1 |
| 58000 | | | 248.4 |
| 59000 | | | 253.7 |
| 60000 | | | 257.4 |
| 61000 | | | 261.1 |
| 62000 | | | 265.1 |
| 62500 | | | 268.2 |

จากผลการทดลองความเร็วที่ได้เมื่อนำมาพล็อตกราฟจะเห็นว่าเมื่อป้อนความถี่สูงขึ้น ความเร็วรอบของมอเตอร์ที่ได้ก็จะมีค่าสูงขึ้นแต่จะไม่สอดคล้องกันในแต่ละโหมด




**รูปที่ 4.17** กราฟความเร็วของโหมดต่างๆในการขับสเตปปิงมอเตอร์ขณะไม่มีการจัดบิต

แต่เมื่อเรานำบิตของ Full , Half มาขยายจำนวนบิตให้เท่ากับ Sine จะทำให้ความเร็วรอบของมอเตอร์เท่ากันในทุก ๆ โหมด

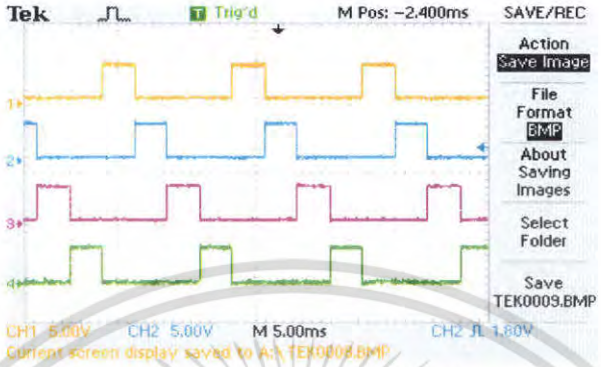



**รูปที่ 4.18** กราฟความเร็วของโหมดต่าง ๆ ในการขับสเตปปิงมอเตอร์ขณะมีการจัดบิต

เมื่อทำให้ความเร็วรอบเท่ากันก็จะได้ย่านของการใช้งานที่กว้างมากขึ้นและมีประสิทธิภาพมากขึ้นด้วยโหมดที่เราเลือกใช้คือโหมด Full Step ,Sine และ Half Step เพราะว่า Full step เหมาะสำหรับการใช้งานย่านความถี่ต่ำ ๆ ดังนั้นเราจึงเลือกให้เป็นโหมดที่ 1 ส่วน Half step นั้นจะใช้ย่านความถี่ได้มากที่สุด เราจึงให้เป็นโหมดสุดท้ายและเมื่อความถี่สูงเกินกว่า Half step มอเตอร์จะเกิดการสลิปเราจึงมี CURRENT SENSOR ไว้เป็นตัว Reset สัญญาณอินพุทซึ่งลักษณะสัญญาณต่าง ๆ เป็นดังนี้

Full stepเป็นสัญญาณที่ใช้ในการขับสเตปปิงมอเตอร์โดยกระตุ้นทีละเฟส

ลักษณะสัญญาณของ Full step

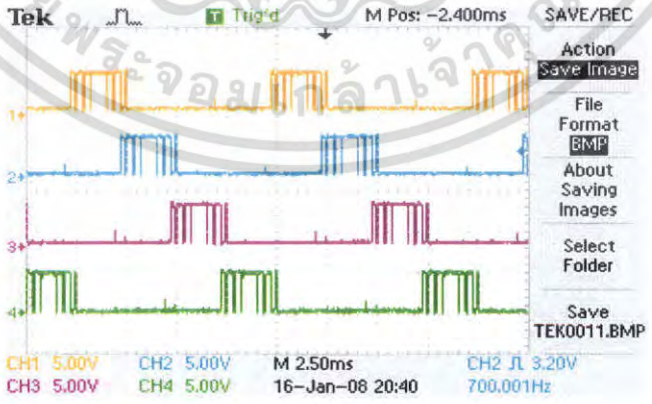



**รูปที่ 4.19** สัญญาณ Full step

เมื่อเราป้อนความถี่ (Fin) เข้าไปที่ FPGA ความถี่ (Fin) นั้นจะถูกนำไปนับและค่าที่นับได้จะนำไปชี้ตำแหน่ง Address ต่าง ๆ ที่เราเก็บไว้ภายใน ROM และให้รูปแบบสัญญาณออกมาซึ่งจะเป็นดังรูปที่ 4.19 ซึ่งจะอยู่ใน โหมด Full step มีความถี่ตั้งแต่เริ่มต้นจนถึง 20 kHz ตามโปรแกรมที่เราได้ออกแบบไว้

Sinusoidal mini step คือ สัญญาณ PULSE ภายในจะถูกแบ่งให้เป็น PULSE ย่อย ๆ ใน 1 ลูก

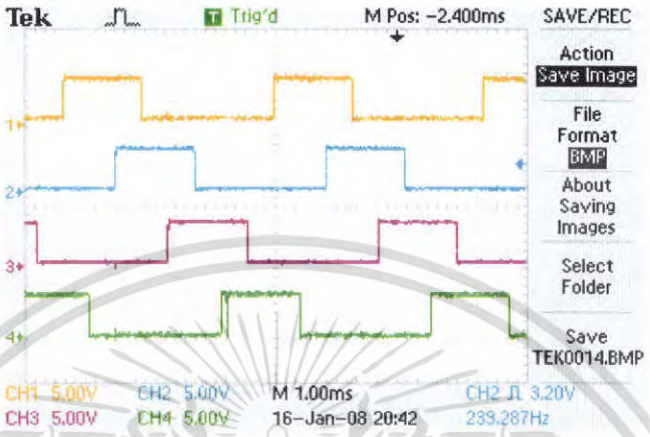
ลักษณะสัญญาณของ Sinusoidal mini step




**รูปที่ 4.20** สัญญาณ Sinusoidal ministep

จากรูปที่ 4.20 เป็นลักษณะสัญญาณการขับในโหมด Sinusoidal mini step ซึ่งความถี่ที่ใช้ในโหมดนี้จะอยู่ที่ 21-45 kHz โหมดนี้จะทำงานหลังต่อจากโหมด Full step

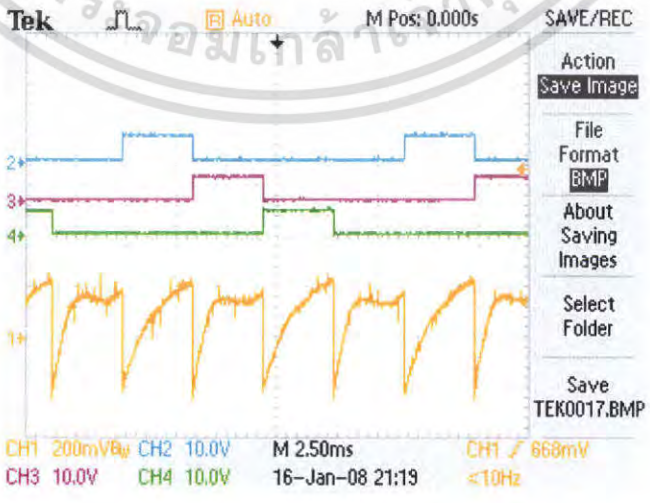
ลักษณะสัญญาณของ Half step




**รูปที่ 4.21** สัญญาณ Half step

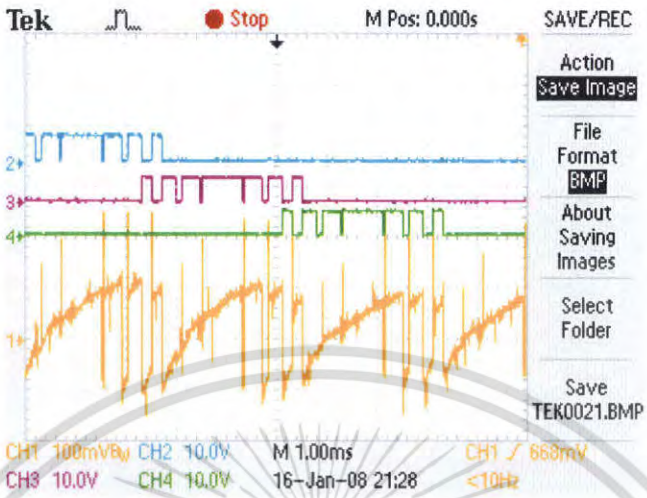
รูปที่ 4.21 เป็นลักษณะสัญญาณการขับในรูปแบบของโหมด Half step ความถี่ F(in) ที่ใช้สำหรับโหมดนี้จะต่อจากโหมด Sinusoidal mini step ความถี่ที่ใช้จะอยู่ในช่วง 46 - 65 kHz เป็นสัญญาณที่ใช้ในการขับสเตปปิงมอเตอร์โดยกระตุ้นทีละ 2 เฟสพร้อมกันเมื่อทำการจับกระแสในแต่ละโหมดของสัญญาณที่ใช้ขับสเตปปิงมอเตอร์ สัญญาณกระแสที่ได้จะเห็นว่ามีการใช้กระแสที่ไม่เท่ากัน โดยขณะก่อนสเตปปิงมอเตอร์เกิดการสลิปกระแสที่ได้จะมีค่าน้อยเมื่อเกิดการสลิปกระแสจะมีค่าสูงที่สุดในโหมดที่ใช้ขับขณะนั้น

ลักษณะสัญญาณของกระแสที่โหมด Full step



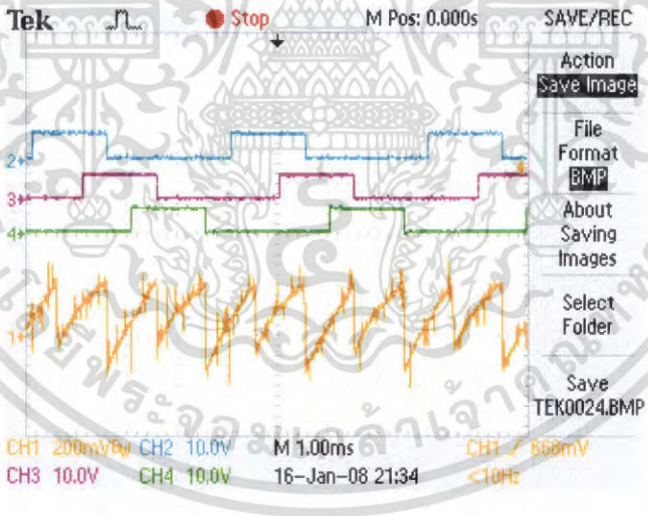

**รูปที่ 4.22** สัญญาณกระแส Full step

ลักษณะสัญญาณของกระแสที่โหมด Sinusoidal mini step




**รูปที่ 4.23** สัญญาณกระแส Sinusoidal mini step

ลักษณะสัญญาณของกระแสที่โหมด Half step




**รูปที่ 4.24** สัญญาณกระแส Half step

ลักษณะสัญญาณของกระแสเมื่อมอเตอร์สลิป


**รูปที่ 4.25** สัญญาณกระแสเมื่อสลิปที่โหมด Half step

จากรูปที่ 4.25 เป็นลักษณะสัญญาณของกระแสเมื่อมอเตอร์เกิดการสลิป การที่มอเตอร์เกิดการสลิปนั้นเกิดจากการที่มอเตอร์ไม่สามารถตอบสนองทันต่อความถี่ (Fin) ที่ป้อนเข้ามา การป้อนความถี่ (Fin) แต่ครั้งจะมีกระแสไหลเข้าไปในขดลวดจึงทำให้เกิดสนามแม่เหล็กขึ้นซึ่งจะทำให้เกิดการเหนี่ยวนำตัวโรเตอร์ทำให้มอเตอร์หมุน แต่การที่ป้อนกระแสในลักษณะที่เร็วเกินไปจะมีผลให้กระแสยังคงค้างอยู่ทำให้กระแสไม่สามารถเข้าไปได้จึงทำให้เกิดการสลิปของตัวมอเตอร์

ผลการทดลองเมื่อทำการจัดบิตภายในจะได้ดังตารางที่ 4.2

**ตารางที่ 4.2** แสดงความเร็วรอบของแต่ละโหมดเมื่อเรียงบิตแล้ว

| ความถี่ Fin (kHz) | ความเร็วรอบ Full step (rpm) | ความเร็วรอบ Half step (rpm) | ความเร็วรอบ Sine (rpm) |
|---|---|---|---|
| 5 | 21.3 | 21.4 | 21.4 |
| 10 | 38.4 | 42.8 | 42.8 |
| 15 | 64.2 | 64.4 | 64.2 |
| 20 | 85.7 | 85.8 | 85.8 |
| 25 | 107.3 | 107.2 | 107.1 |
| 30 | 128.7 | 128.5 | 128.6 |
| 35 | 150.1 | 150.1 | 150.0 |
| 40 | 171.5 | 171.5 | 171.4 |
| 45 | 193.2 | 193.3 | 193.0 |
| 50 | 213.6 | 214.0 | 214.3 |
| 55 | 235.3 | 235.6 | 235.7 |
| 60 | 257.1 | 257.2 | 257.5 |
| 65 | 278.3 | 278.7 | |
| 70 | | 300.0 | |
| 75 | | 321.4 | |
| 78 | | 334.4 | |

# บทที่ 5
# ปัญหา และ สรุปผล

ในการออกแบบบอร์ด DRIVER และโปรแกรมที่ได้ทำการออกแบบนั้นสามารถใช้งานได้กับสเตปปิงมอเตอร์ทุก RESOLUTION และสเตปปิงมอเตอร์ทุกแบบที่มีสาย COMMOM แต่กรณีที่มีสาย COMMON มากกว่า 2 เส้นขึ้นไปจะสามารถประยุกต์ใช้งานได้โดยการนำสาย COMMON ทั้งหมดมาต่อรวมกัน

จากการทดลองเมื่อทำการปรับความถี่จะทำให้ FPGA เปลี่ยนโหมดเป็น Full step เมื่อความถี่ที่ 1-22 kHz Sinusoidal mini step ความถี่ที่ 22-46 kHz และ Half step ความถี่ที่ 46-68 kHz ถ้าทำการให้ความถี่ที่มากกว่า 68 kHz จะทำให้มอเตอร์เกิดการสลิป CURRENT SENSOR จะส่งสัญญาณให้ FPGA ทำการตัดสัญญาณในการขับสเตปปิงมอเตอร์เพื่อป้องกันการเสียหายของสเตปปิงมอเตอร์เนื่องจากถ้ามอเตอร์มีการหยุดหมุนค่าแรงดันต้านกลับภายในสเตปปิงมอเตอร์จะเป็นศูนย์ทำให้กระแสที่ไหลเข้าสเตปปิงมอเตอร์มีค่าสูงขึ้นและเมื่อทำการกด Switch ยังสามารถกลับทางหมุนของสเตปปิงมอเตอร์ได้

จากการทดลองปัญหาที่เกิดขึ้น คือ เมื่อมีการสลับโหมดขณะที่อยู่ระหว่างการเปลี่ยนโหมดสัญญาณที่ได้จาก FPGA จะให้สัญญาณ 2 แบบพร้อมกัน เช่น เมื่อต้องการเปลี่ยนโหมดจากโหมด Full step ไป Sinusoidal mini step สัญญาณที่ได้จะได้สองสัญญาณพร้อมกัน คือ Full step และ Sinusoidal mini step หลังจากผ่านช่วงที่ทำการเปลี่ยนก็จะให้สัญญาณที่ได้ออกมาเพียงสัญญาณเดียวคือ Sinusoidal mini step

# บรรณานุกรม


1. ธีรยศ เวียงทอง ภาควิชาอิเล็กทรอนิกส์ มหาวิทยาลัยเทคโนโลยีมหานคร
   " เรียนรู้การออกแบบระบบดิจิตอล ด้วยภาษา VERILOG เบื้องต้น "
2. http://www.astronlogic.com
   " บทความเกี่ยวกับFPGA "
3. บริษัท ASTRON LOGIC
   " คู่มือ MAX+PLUS II "
4. บริษัท ASTRON LOGIC
   " คู่มือ FPGA รุ่น ACE1K50TC144-3 "
5. http://www.elecnet.chandra.ac.th/learn/tipntrick/stepping/default.htm
   " Stepping Motor "
6. http://www.wara.com/modules.php?name=News&file=article&sid=259
   " เขียนโปรแกรมควบคุม Stepping Motor "

# ภาคผนวก

FAIRCHILD
SEMICONDUCTOR®

**IRF520**

**Data Sheet** **January 2002**

## 9.2A, 100V, 0.270 Ohm, N-Channel Power MOSFET

This N-Channel enhancement mode silicon gate power field effect transistor is an advanced power MOSFET designed, tested, and guaranteed to withstand a specified level of energy in the breakdown avalanche mode of operation. All of these power MOSFETs are designed for applications such as switching regulators, switching convertors, motor drivers, relay drivers, and drivers for high power bipolar switching transistors requiring high speed and low gate drive power. These types can be operated directly from integrated circuits.

Formerly developmental type TA09594.

## Features

- 9.2A, 100V
- $r_{DS(ON)} = 0.270\Omega$
- SOA is Power Dissipation Limited
- Single Pulse Avalanche Energy Rated
- Nanosecond Switching Speeds
- Linear Transfer Characteristics
- High Input Impedance
- Related Literature
  - TB334 "Guidelines for Soldering Surface Mount Components to PC Boards"

## Ordering Information

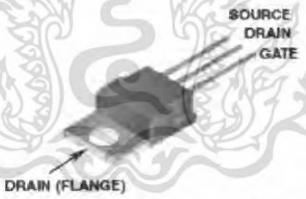
| PART NUMBER | PACKAGE | BRAND |
|---|---|---|
| IRF520 | TO-220AB | IRF520 |

NOTE: When ordering, use the entire part number.

## Symbol


## Packaging

JEDEC TO-220AB



## Absolute Maximum Ratings $T_C = 25^oC$, Unless Otherwise Specified

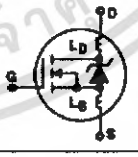
| | | IRF520 | UNITS |
|---|---|---|---|
| Drain to Source Breakdown Voltage (Note 1) | $V_{DS}$ | 100 | V |
| Drain to Gate Voltage ($R_{GS}$ = 20kΩ) (Note 1) | $V_{DGR}$ | 100 | V |
| Continuous Drain Current | $I_D$ | 9.2 | A |
| $T_C$ = 100°C | $I_D$ | 6.5 | A |
| Pulsed Drain Current (Note 3) | $I_{DM}$ | 37 | A |
| Gate to Source Voltage | $V_{GS}$ | ±20 | V |
| Maximum Power Dissipation | $P_D$ | 60 | W |
| Dissipation Derating Factor | | 0.4 | W/°C |
| Single Pulse Avalanche Energy Rating (Note 4) | $E_{AS}$ | 36 | mJ |
| Operating and Storage Temperature | $T_J$, $T_{STG}$ | -55 to 175 | °C |
| Maximum Temperature for Soldering | | | |
| Leads at 0.063in (1.6mm) from Case for 10s | $T_L$ | 300 | °C |
| Package Body for 10s, See Techbrief 334 | $T_{pkg}$ | 260 | °C |

*CAUTION: Stresses above those listed in "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress only rating and operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.*

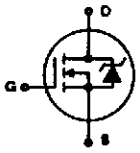
NOTE:

1. $T_J$ = 25°C to 150°C.

## Electrical Specifications $T_C = 25^oC$, Unless Otherwise Specified

| PARAMETER | SYMBOL | TEST CONDITIONS | | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|---|---|
| Drain to Source Breakdown Voltage | $BV_{DSS}$ | $I_D$ = 250µA, $V_{GS}$ = 0V (Figure 10) | | 100 | - | - | V |
| Gate to Threshold Voltage | $V_{GS(TH)}$ | $V_{GS}$ = $V_{DS}$, $I_D$ = 250µA | | 2.0 | - | 4.0 | V |
| Zero Gate Voltage Drain Current | $I_{DSS}$ | $V_{DS}$ = 95V, $V_{GS}$ = 0V | | - | - | 250 | µA |
| | | $V_{DS}$ = 0.8 x Rated $BV_{DSS}$, $V_{GS}$ = 0V, $T_J$ = 150°C | | - | - | 1000 | µA |
| On-State Drain Current (Note 2) | $I_{D(ON)}$ | $V_{DS}$ > $I_{D(ON)}$ x $r_{DS(ON)MAX}$, $V_{GS}$ = 10V (Figure 7) | | 9.2 | - | - | A |
| Gate to Source Leakage Current | $I_{GSS}$ | $V_{GS}$ = ±20V | | - | - | ±100 | nA |
| Drain to Source On Resistance (Note 2) | $r_{DS(ON)}$ | $I_D$ = 5.6A, $V_{GS}$ = 10V (Figure 8, 9) | | - | 0.25 | 0.27 | Ω |
| Forward Transconductance (Note 2) | $g_{fs}$ | $V_{DS}$ ≥ 50V, $I_D$ = 5.6A (Figure 12) | | 2.7 | 4.1 | - | S |
| Turn-On Delay Time | $t_{d(ON)}$ | $V_{DD}$ = 50V, $I_D$ = 9.2A, $R_G$ = 18Ω, $R_L$ = 5.5Ω MOSFET Switching Times are Essentially Independent of Operating Temperature | | - | 9 | 13 | ns |
| Rise Time | $t_r$ | | | - | 30 | 63 | ns |
| Turn-Off Delay Time | $t_{d(OFF)}$ | | | - | 18 | 70 | ns |
| Fall Time | $t_f$ | | | - | 20 | 59 | ns |
| Total Gate Charge (Gate to Source + Gate to Drain) | $Q_{g(TOT)}$ | $V_{GS}$ = 10V, $I_D$ = 9.2A, $V_{DS}$ = 0.8 x Rated $BV_{DSS}$, $I_{g(REF)}$ = 1.5mA (Figure 14) Gate Charge is Essentially Independent of Operating Temperature | | - | 10 | 30 | nC |
| Gate to Source Charge | $Q_{gs}$ | | | - | 2.5 | - | nC |
| Gate to Drain "Miller" Charge | $Q_{gd}$ | | | - | 2.5 | - | nC |
| Input Capacitance | $C_{ISS}$ | $V_{DS}$ = 25V, $V_{GS}$ = 0V, f = 1MHz (Figure 11) | | - | 350 | - | pF |
| Output Capacitance | $C_{OSS}$ | | | - | 130 | - | pF |
| Reverse Transfer Capacitance | $C_{RSS}$ | | | - | 25 | - | pF |
| Internal Drain Inductance | $L_D$ | Measured From the Contact Screw On Tab To Center of Die | Modified MOSFET Symbol Showing the Internal Devices Inductances | - | 3.5 | - | nH |
| | | Measured From the Drain Lead, 6mm (0.25in) From Package to Center of Die |  | - | 4.5 | - | nH |
| Internal Source Inductance | $L_S$ | Measured From the Source Lead, 6mm (0.25in) From Header to Source Bonding Pad | | - | 7.5 | - | nH |
| Thermal Resistance Junction to Case | $R_{\theta JC}$ | | | - | - | 2.5 | °C/W |
| Thermal Resistance Junction to Ambient | $R_{\theta JA}$ | Free Air Operation | | - | - | 80 | °C/W |

**Source to Drain Diode Specifications**

| PARAMETER | SYMBOL | TEST CONDITIONS | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|---|
| Continuous Source to Drain Current | $I_{SD}$ | Modified MOSFET Symbol Showing the Integral Reverse P-N Junction Diode  | - | - | 9.2 | A |
| Pulse Source to Drain Current (Note 3) | $I_{SDM}$ | | - | - | 37 | A |
| Source to Drain Diode Voltage (Note 2) | $V_{SD}$ | $T_J = 25^oC$, $I_{SD} = 9.2A$, $V_{GS} = 0V$ (Figure 13) | - | - | 2.5 | V |
| Reverse Recovery Time | $t_{rr}$ | $T_J = 25^oC$, $I_{SD} = 9.2A$, $dI_{SD}/dt = 100A/\mu s$ | 5.5 | 100 | 240 | ns |
| Reverse Recovered Charge | $Q_{RR}$ | $T_J = 25^oC$, $I_{SD} = 9.2A$, $dI_{SD}/dt = 100A/\mu s$ | 0.17 | 0.5 | 1.1 | µC |

NOTES:

2. Pulse test: pulse width ≤ 300µs, duty cycle ≤ 2%.
3. Repetitive rating: pulse width limited by Max junction temperature. See Transient Thermal Impedance curve (Figure 3).
4. $V_{DD} = 25V$, starting $T_J = 25^oC$, L = 640mH, $R_G = 25\Omega$, peak $I_{AS} = 9.2A$.

Agilent Technologies

# Single Channel, High Speed Optocouplers

## Technical Data

**6N135/6**
**HCNW135/6**
**HCNW4502/3**
**HCPL-2502**
**HCPL-0452/3**
**HCPL-0500/1**
**HCPL-4502/3**

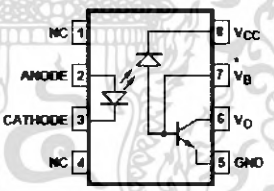
### Features

- **15 kV/µs Minimum Common Mode Transient Immunity at $V_{CM}$ = 1500 V (4503/0453)**
- **High Speed: 1 Mb/s**
- **TTL Compatible**
- **Available in 8-Pin DIP, SO-8, Widebody Packages**
- **Open Collector Output**
- **Guaranteed Performance from Temperature: 0°C to 70°C**
- **Safety Approval**
  UL Recognized – 3750 V rms for 1 minute (5000 V rms for 1 minute for HCNW and Option 020 devices) per UL1577
  CSA Approved
  IEC/EN/DIN EN 60747-6-2 Approved
  – $V_{IORM}$ = 630 V peak for HCPL-4503#060
  – $V_{IORM}$ = 1414 V peak for HCNW devices
- **Dual Channel Version Available (253X/4534/053X/0534)**
- **MIL-PRF-38534 Hermetic Version Available (55XX/65XX/4N55)**

### Applications

- **High Voltage Insulation**
- **Video Signal Isolation**
- **Power Transistor Isolation in Motor Drives**
- **Line Receivers**
- **Feedback Element in Switched Mode Power Supplies**
- **High Speed Logic Ground Isolation – TTL/TTL, TTL/CMOS, TTL/LSTTL**
- **Replaces Pulse Transformers**
- **Replaces Slow Phototransistor Isolators**
- **Analog Signal Ground Isolation**

### Functional Diagram




TRUTH TABLE (POSITIVE LOGIC)

| LED | $V_O$ |
|---|---|
| ON | LOW |
| OFF | HIGH |

* NOTE: FOR 4502/3, 0452/3, PIN 7 IS NOT CONNECTED.

A 0.1 µF bypass capacitor must be connected between pins 5 and 8.

### Description

These diode-transistor optocouplers use an insulating layer between a LED and an integrated photodetector to provide electrical insulation between input and output. Separate connections for the photodiode bias and output-transistor collector increase the speed up to a hundred times that of a conventional phototransistor coupler by reducing the base-collector capacitance.

*CAUTION: It is advised that normal static precautions be taken in handling and assembly of this component to prevent damage and/or degradation which may be induced by ESD.*

## Switching Specifications (AC)

Over recommended temperature ($T_A$ = 0°C to 70°C), $V_{CC}$ = 5 V, $I_F$ = 16 mA unless otherwise specified.

| Parameter | Sym. | Device | Min. | Typ.** | Max. | Units | Test Conditions | Fig. | Note |
|---|---|---|---|---|---|---|---|---|---|
| Propagation Delay Time to Logic Low at Output | $t_{PHL}$* | 6N135 HCPL-0500 HCNW135 | | 0.2 | 1.5 | µs | $T_A$ = 25°C, $R_L$ = 4.1 kΩ | 5, 6, 11 | 8, 9 |
| | | | | | 2.0 | | | | |
| | | 6N136 HCPL-2502 HCPL-4502/3 HCPL-0501 HCPL-0452/3 HCNW136 HCNW4502/3 | | 0.2 | 0.8 | | $T_A$ = 25°C, $R_L$ = 1.9 kΩ | | |
| | | | | | 1.0 | | | | |
| Propagation Delay Time to Logic High at Output | $t_{PLH}$* | 6N135 HCPL-0500 HCNW135 | | 1.3 | 1.5 | µs | $T_A$ = 25°C, $R_L$ = 4.1 kΩ | 5, 6, 11 | 8, 9 |
| | | | | | 2.0 | | | | |
| | | 6N136 HCPL-2502 HCPL-4502/3 HCPL-0501 HCPL-0452/3 HCNW136 HCNW4502/3 | | 0.6 | 0.8 | | $T_A$ = 25°C, $R_L$ = 1.9 kΩ | | |
| | | | | | 1.0 | | | | |
| Common Mode Transient Immunity at Logic High Level Output | $\|CM_H\|$ | 6N135 HCPL-0500 HCNW135 | | 1 | | kV/µs | $R_L$ = 4.1 kΩ, $I_F$ = 0 mA, $T_A$ = 25°C, $V_{CM}$ = 10 $V_{p-p}$, $C_L$ = 15 pF | 12 | 7, 8, 9 |
| | | 6N136 HCPL-2502 HCPL-4502 HCPL-0501 HCPL-0452 HCNW4502 | | 1 | | | $R_L$ = 1.9 kΩ | | |
| | | HCPL-4503 HCPL-0453 HCNW4503 | 15 | 30 | | | $R_L$ = 1.9 kΩ, $I_F$ = 0 mA, $T_A$ = 25°C, $V_{CM}$ = 1500 $V_{p-p}$, $C_L$ = 15 pF | | |
| Common Mode Transient Immunity at Logic Low Level Output | $\|CM_L\|$ | 6N135 HCPL-0500 HCNW135 | | 1 | | kV/µs | $R_L$ = 4.1 kΩ, $I_F$ = 16 mA, $T_A$ = 25°C, $V_{CM}$ = 10 $V_{p-p}$, $C_L$ = 15 pF | 12 | 7, 8, 9 |
| | | 6N136 HCPL-2502 HCPL-4502 HCPL-0501 HCPL-0452 HCNW4502 | | 1 | | | $R_L$ = 1.9 kΩ | | |
| | | HCPL-4503 HCPL-0453 HCNW4503 | 15 | 30 | | | $R_L$ = 1.9 kΩ, $I_F$ = 16 mA, $T_A$ = 25°C, $V_{CM}$ = 1500 $V_{p-p}$, $C_L$ = 15 pF | | |
| Bandwidth | BW | 6N135/6 HCPL-2502 HCPL-0500/1 | | 9 | | MHz | See Test Circuit | 8, 10 | 10 |
| | | HCNW135/6 | | 11 | | | | | |

*For JEDEC registered parts.
**All typicals at $T_A$ = 25°C.

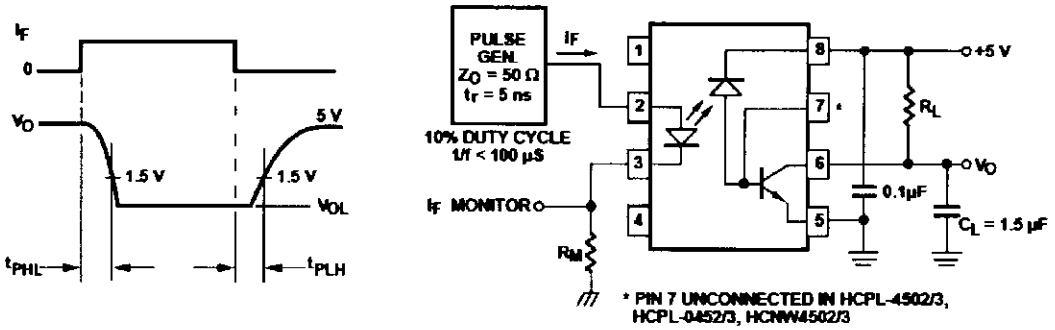



**Figure 11. Switching Test Circuit.**