ระบบบันทึกภาพเพื่อรักษาความปลอดภัย

Video Capturing for Security System

โดย

นางสาว ดวงเดือน ทิพย์พิมาน

นางสาว เสาวภาคย์ เตชะนิรัติศัย

นางสาว โสรยา ฉกษ์ใหญ่

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2550

ระบบบันทึกภาพเพื่อรักษาความปลอดภัย

Video Capturing for Security System

โดย

นางสาว ดวงเดือน ทิพย์พิมาน

นางสาว เสาวภาคย์ เตชะนิรัติศัย

นางสาว โสรยา ฤกษ์ใหญ่

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2550

ปริญญานิพนธ์ปีการศึกษา 2550

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง **ระบบบันทึกภาพเพื่อรักษาความปลอดภัย**

Video Capturing for Security System

ผู้จัดทำ

1. นางสาวดวงเดือน	ทิพย์พิมาน	47010253
2. นางสาวเสาวภาคย์	เตชะนิรัติศัย	47010911
3. นางสาวโสรยา	ฤกษ์ใหญ่	47010914

.....อาจารย์ที่ปรึกษา

(รศ.เกรียงไกร วงศ์โรจนภรณ์)

.....อาจารย์ที่ปรึกษา

(รศ.ดร.สุวิพล สิทธิชีวภาค)

โครงงานเลขที่ 502432

ระบบบันทึกภาพเพื่อรักษาความปลอดภัย

Video Capturing for Security System

โดย นางสาวดวงเดือน ทิพย์พิมาน 47010253
นางสาวเสาวภาคย์ เตชะนิรัติศัย 47010911
นางสาวโสรยา ฤกษ์ใหญ่ 47010914

อาจารย์ที่ปรึกษา รศ. เกรียงไกร วงศ์โรจนภรณ์
รศ.ดร. สุวิพล สิทธิชีวภาค

บทคัดย่อ

โครงงานนี้เป็นการนำเอาเทคโนโลยีในเรื่องของ FPGA เข้ามาประยุกต์ใช้กับระบบรักษาความปลอดภัย โดยจะใช้ภาษา VHDL กำหนดการทำงานของ FPGA ที่มีการรับภาพที่เป็นสัญญาณ VDO แล้วนำภาพที่ได้มาบันทึกเพื่อนำไปใช้ประโยชน์ต่อไป ซึ่งสามารถนำไปใช้ในระบบรักษาความปลอดภัยขององค์กรต่างๆได้

Abstract

This project integrates FPGA technology with security system. By using VHDL language control FPGA chip, that interface with video signal and capture image for applied in security system.

สารบัญ

หน้า

สารบัญรูปภาพ

หน้า

หน้า

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

ปัจจุบัน ระบบรักษาความปลอดภัยจัดได้ว่าเป็นระบบที่จำเป็นอย่างยิ่งสำหรับทุกองค์กร ซึ่งการบันทึกภาพก็เป็นส่วนหนึ่งของวิวัฒนาการที่เกิดขึ้น เพื่อการรักษาความปลอดภัยอย่างมีประสิทธิภาพ โครงงานนี้จึงเป็นการนำเอา อุปกรณ์ FPGA ซึ่งเป็นอุปกรณ์ที่สามารถโปรแกรมวงจรลงไปโดยไม่ต้องใช้อุปกรณ์แยกชิ้น ทำให้สะดวกในการออกแบบ และสามารถนำไปประยุกต์ได้หลากหลายแนวทาง

สัญญาณคอมโพสิทวิดีโอ และสัญญาณ VGA เป็นสัญญาณที่ได้จากจอภาพ ซึ่งประกอบไปด้วยสัญญาณหลักๆ 5 สัญญาณด้วยกันคือ สัญญาณ Horizontal Sync และ Vertical Sync เป็นระดับสัญญาณลอจิก TTL ส่วนอีก 3 สัญญาณเป็นสัญญาณอนาล็อก ดังนั้นเมื่อจะทำระบบบันทึกภาพ จึงจำเป็นต้องรู้ถึงลักษณะของสัญญาณ และรูปแบบที่จะใช้ในการจัดเก็บ ซึ่ง สัญญาณ Horizontal Sync และ Vertical Sync นั้น เป็นสัญญาณที่เป็นดิจิตอลอยู่แล้ว จึงไม่เป็นปัญหา แต่สัญญาณ RGB เป็นสัญญาณอนาล็อก จึงต้องทำการแปลงสัญญาณให้เป็น สัญญาณดิจิตอล เพื่อให้สะดวกในการจัดเก็บ เราจึงสามารถศึกษาได้ทั้งลักษณะของสัญญาณภาพ และ วงจรที่ใช้ในการทำงานได้มากมาย

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อศึกษาการทำงานรวมถึงความสามารถของ FPGA
2. เพื่อศึกษา หลักการทำงานของวงจรแปลงสัญญาณจากอนาล็อกเป็นดิจิตอล
3. เพื่อศึกษา การเก็บข้อมูลลงใน RAM ซึ่งต้องมีการกำหนดค่า Address ก่อนทำการเก็บข้อมูล
4. นำความรู้ที่ได้มาสร้างระบบแสดงผลภาพให้ได้มาตรฐานตามต้องการ

1.3 ขอบเขตของโครงงาน

ในส่วนของเทอมนี้ จะเน้นการศึกษา การส่งสัญญาณภาพเข้าไปเก็บใน Block RAM ภายในและทำการแสดงผลได้ และยังมีการใช้สัญญาณ ภาพส่งออกไปยัง RAM ที่ต่ออยู่ภายนอก และเรียกแสดงผลข้อมูลจาก RAM ที่ต่ออยู่ภายนอกนั้น ผ่านพอร์ทอนุกรมได้ ซึ่งยังต้องพัฒนาในส่วนของวงจรและโปรแกรมต่อไป

1.4 ประโยชน์ที่ได้รับ

1. สามารถประยุกต์ใช้อุปกรณ์ FPGA ได้อย่างกว้างขวาง
2. สามารถเขียนโปรแกรมเพื่อควบคุมการทำงานของ FPGA ได้
3. สามารถแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิตอลได้

บทที่ 2
ทฤษฎีและหลักการ

2.1 ทฤษฎีเบื้องต้นเกี่ยวกับการทำงานของจอ VGA

สัญญาณที่ใช้ในระบบภาพ VGA ประกอบด้วย 5 สัญญาณเพื่อที่จะนำมาควบคุมการแสดงผล โดยมี 2 สัญญาณที่เกี่ยวข้องกับระดับสัญญาณลอจิก TTL นั่นคือสัญญาณ HS (Horizontal Synchronization) และ VS (Vertical Synchronization) ซึ่งใช้สำหรับสังเคราะห์สัญญาณภาพ ส่วนอีก 3 สัญญาณนั้นจะเกี่ยวข้องกับสัญญาณอนาล็อก โดยมีค่า 0.7 - 1.0 V_{p-p} จะใช้ในการควบคุมสัญญาณสี ซึ่งสัญญาณสีประกอบด้วยสี Red(R) Green(G) และ Blue(B)

รูปที่ 2.1 VGA connector (female) และการนับขา

ในการแสดงผลของจอมอนิเตอร์นั้นจะเป็นมาตรฐานเดียวกันคือ มีการแสดงทีละเส้น line-by-line และจากบนลงล่าง top-to-bottom ในแต่ละเส้นนั้นจะมีการกวาดจากซ้ายไปขวา left-to-right ดังนั้นเราจึงสามารถกำหนดการแสดงในแต่ละช่วงของการกวาดให้เป็นลักษณะของ coordinate x-y โดยใช้สัญญาณ Horizontal และ Vertical Synchronization (HS และ VS) เป็นตัวกำหนด (reference) โดยที่สัญญาณ HS จะเป็นตัวบอกช่วงที่สัญญาณทำการกวาดในแนวนอน (x-coordinate) และ VS จะเป็นตัวบอกว่าตอนนี้ทำการกวาดในเส้นที่เท่าไร (y-coordinate)

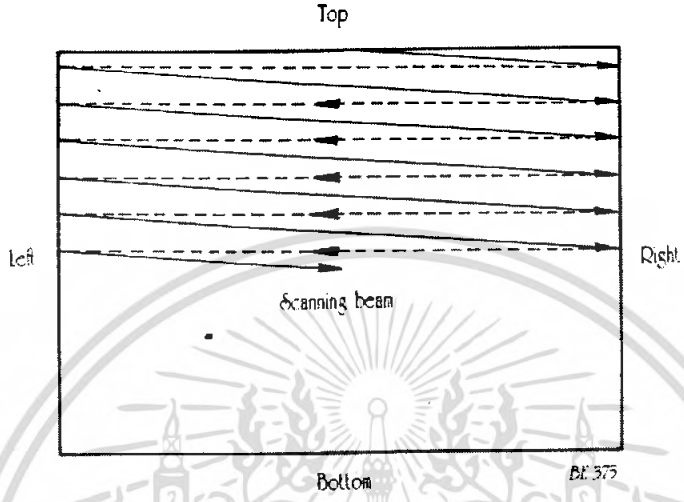
2.2 การกวาดตรวจภาพ (Scanning)

การที่จะนำภาพ 2 มิติ เคลื่อนย้ายจากที่หนึ่งไปยังอีกที่หนึ่ง ทำได้โดย ทำภาพนั้นเป็นสัญญาณไฟฟ้าซึ่งจะง่ายต่อการเคลื่อนย้าย แต่ปัญหาคือจะทำการเปลี่ยนภาพ 2 มิติไปเป็นโวลต์เตจที่เปลี่ยนตามเวลาปัญหานี้แก้ได้โดยทฤษฎีของการกวาดตรวจ

วิธีการสแกนของ CRT ที่นิยมใช้กันอย่างแพร่หลายในการออกแบบทางการค้าโทรทัศน์และจอภาพแสดงผล มีชื่อเรียกว่า การตรวจกวาดแบบราสเตอร์ (Raster scan) ซึ่งวิธีนี้จะต้องมีการให้กำเนิดซิงก์ทั้งในแนวแกนตั้งและแนวนอนเพื่อให้ลำอิเล็กตรอนสามารถเคลื่อนที่ได้

ในการสแกนแบบนี้จะเริ่มทำจากซ้ายไปขวา และ จากบนลงไปข้างของจอภาพ โดยที่ลำอิเล็กตรอน จะถูกเลี้ยวเบนให้ไปอยู่ทางมุมซ้ายบนและเมื่อกวาดไปทางขวาของฉาก ลำอิเล็กตรอนก็จะ

สะบัดกลับโดยตอนที่กลับนั้นก็จะมีการเลื่อนตำแหน่งลงไปด้วย การสะบัดกลับนี้เรียกว่า Retrace หรือ fly back ซึ่งในส่วนของการสะบัดกลับของลำอิเล็กตรอนนั้นความเข้มของลำอิเล็กตรอนจะลดลง เมื่อลดความเข้มลงอย่างเพียงพอแล้วฉากที่เคลือบด้วยฟลูเรซเซนท์ก็จะไม่ถูกกระตุ้นทำให้ไม่เกิดแสงสว่าง จึงมองไม่เห็นเส้นที่ลากกลับ



รูปที่ 2.2 วิธีการเบื้องต้นของการสแกน

2.3 การซิงโครไนซ์ (Synchronization)

สิ่งสำคัญที่ทำให้การสแกนภาพนิ่งได้ถูกต้องนั้นเป็นหน้าที่ของสัญญาณซิงก์ ซึ่งแบ่งได้เป็นสัญญาณ Horizontal และ Vertical sync ทำงานไปพร้อมๆกันได้เพราะสัญญาณนี้จะส่งข้อมูล timing แนบไปกับสัญญาณ video

การซิงโครไนซ์ในแนวแกนนอนและแกนตั้ง

- Horizontal Sync จะเริ่มทำงานที่การเริ่ม retrace หรือจบการ trace จะไม่ทำตอนเริ่ม trace

- Vertical Sync เริ่มทำงานที่การเริ่ม retrace ในแนวแกนตั้ง (ก็คือจะทำงานตอน beam อยู่ที่ล่างขวาของจอภาพ)

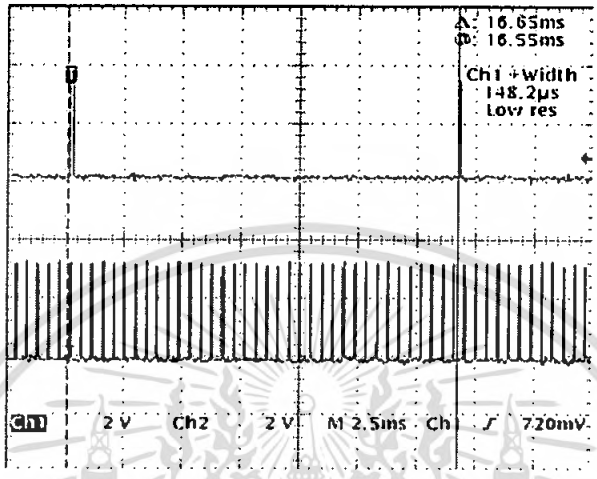
ถ้าหากไม่มีสัญญาณ Vertical sync จะทำให้ภาพที่ถูกสร้างทวนซ้ำขึ้นมาจะไม่สามารถคงตำแหน่งในแนวแกนตั้ง ภาพจะวิ่งขึ้น-ลง แต่ถ้าหากสัญญาณ Horizontal sync ไม่ซิงก์กันจะทำให้ภาพนั้นจะเลื่อนซ้าย-ขวา และบางส่วนจะเหมือนภาพถูกแยกออก

โดยทั่วไปแล้วจอภาพส่วนใหญ่จะใช้สัญญาณอนาล็อก RGB มาตรฐานที่ 0.7 V_{p-p} ส่วนสัญญาณซิงก์ก็จะขึ้นอยู่กับประเภทลักษณะที่ใช้ดังต่อไปนี้

1. RGB + HSYNC + VSYNC จะใช้สายในการเชื่อมต่อ 5 เส้น มาตรฐานของสัญญาณ RGB ที่ใช้คือ 0.7 V_{p-p} และสัญญาณซิงก์ใช้มาตรฐาน TTL หรือ 1V_{p-p} ขึ้นอยู่กับระบบ ลักษณะการใช้งานแบบนี้จะง่ายและเป็นที่นิยม

2. RGB + Composite sync จะใช้สายในการเชื่อมต่อ 4 เส้น เพราะจะใช้สัญญาณซิงก์ 1 เส้น (เรียก CSYNC) ใช้มากในพวกโปรเจกเตอร์ในสมัยก่อน

3. RGB + Sync on green จะใช้สายในการเชื่อมต่อ 3 เส้น สัญญาณ R และ B ยังคงใช้ 0.7 V_{p-p} แต่สัญญาณ G จะส่งรวมไปกับสัญญาณ CYNC ซึ่งจะใช้งานที่ -0.3 V_{p-p}



รูปที่ 2.3 สัญญาณ Vertical Sync เทียบกับ สัญญาณ Horizontal Sync

2.4 องค์ประกอบของภาพ

หากเราตัดภาพจากหนังสือพิมพ์สักภาพหนึ่ง แล้วขยายขึ้นด้วยกล้อง หรือแว่นขยายจะพบว่าภาพมีองค์ประกอบมาจากจุดสีขาวและจุดสีดำมากมาย มาเรียงกันประกอบขึ้นเป็นภาพจุดเหล่านี้เองที่เรียกว่าองค์ประกอบของภาพ (Picture Element) หรือ พิกเซล (Pixel)

ทำนองเดียวกัน ภาพที่ปรากฏทางจอภาพก็เอามาจากหลักการนี้ ภาพที่เกิดขึ้นบนจอภาพหรือโทรทัศน์ประกอบด้วยเส้นขวางเล็กๆในแนวนอนเป็นจำนวนมาก แต่ละเส้นนั้นมีทั้งส่วนที่ดำสนิท ส่วนที่จาง และส่วนที่สว่างรวมกันอยู่ เส้นเหล่านี้เราได้มาจากการกวาดลำแสงแสดงความแตกต่างกันบนเส้นกวาดลำแสงหรือเส้นสแกนเหล่านี้เองที่เราจัดว่าเป็นองค์ประกอบภาพ

ปัจจุบันส่วนที่เรียกว่าองค์ประกอบภาพได้ถูกนำไปใช้งานอย่างจริงจังมากขึ้น ในส่วนโทรทัศน์หรือเครื่องเล่นวีดีโอคาสเซ็ทเร็คคอร์ดเดอร์ ซึ่งจะมีการนำพิกเซลเหล่านี้เก็บไว้ในหน่วยความจำเพราะข้อมูลที่เป็นพิกเซลเท่านั้นที่ระบบดิจิตอลจะจัดการข้อมูลได้ เราจะพบวิธีการนี้ในโทรทัศน์ระบบดิจิตอล โทรทัศน์ระบบคอมพิวเตอร์ โทรทัศน์จอภาพแอลซีดี วีซีอาร์ ระบบภาพซ้อนภาพ เป็นต้น

ในปัจจุบันสำหรับโทรทัศน์ธรรมดาเราจะพบว่ามีการเพิ่มเส้นสแกนภาพให้มากขึ้น และแน่นอนว่าจำนวนพิกเซลย่อมมากขึ้นด้วย อย่างโทรทัศน์จอใหญ่หรือ โทรทัศน์ที่ต้องการรายละเอียดสูง HDTV อาจจะต้องใช้เส้นสแกนภาพมากกว่า 625 เส้น เช่นที่นิยมใช้ในปัจจุบันคือ 725 เส้น หรืออย่างเครื่องฉายวีดีโอโปรเจกเตอร์จะต้องใช้เส้นภาพ 2,200 เส้นภาพ และหากเป็นจอใหญ่หลายร้อยนิ้วจะต้องเพิ่มรายละเอียดมากขึ้นอีก นั่นคือการเพิ่มพิกเซลอีลีเมนต์นั่นเอง

2.5 คุณสมบัติของแสงและสี

แสงโดยทั่วไปแบ่งเป็น 2 ประเภท คือแสงที่ตามนุษย์มองเห็น กับแสงที่ตามนุษย์มองไม่เห็น

- แสงที่ตามนุษย์มองไม่เห็น ก็เช่นแสงจำพวกรังสีแกมมา (Gamma-ray) รังสีเอ็กซ์ (X-ray) แสงเหนือม่วง (Ultraviolet ray) และแสงใต้แดง (Infrared ray) แสงเหล่านี้เป็นส่วนหนึ่งของพลังแม่เหล็กไฟฟ้า แต่มีความยาวคลื่นแตกต่างกันออกไป

- แสงที่ตามนุษย์มองเห็นเป็นคลื่นแม่เหล็กไฟฟ้าที่มีช่วงความยาวคลื่นประมาณ 380 nm จนถึงประมาณ 780 nm ซึ่งสีที่มองเห็นนี้ ตาของมนุษย์จะรู้สึกได้ 2 ประการ โดยประการแรกคือความรู้สึกว่าแสงนี้เป็นสีอะไร (sensation of color) ส่วนประการที่ 2 คือแสงสีนี้มีความสว่างมากหรือน้อย (sensation of brightness)

2.6 ความสำคัญ 3 ประการของแสงที่ตามองเห็น

แสงที่ตามองเห็นทำให้รู้สึกที่สำคัญอยู่ 3 ประการคือ เกิดความรู้สึกในเรื่องสีของแสง (Hue) เกิดความรู้สึกในเรื่องการส่องสว่าง (Brightness) เกิดความรู้สึกในเรื่องราวสีอิ่มตัว (Saturation)

- ความรู้สึกในเรื่องสีของแสง (Hue) จะทำให้สายตาสามารถแยกแยะออกได้ว่าแสงที่เห็นเป็นสีแดง สีเขียว สีเหลือง เป็นต้น

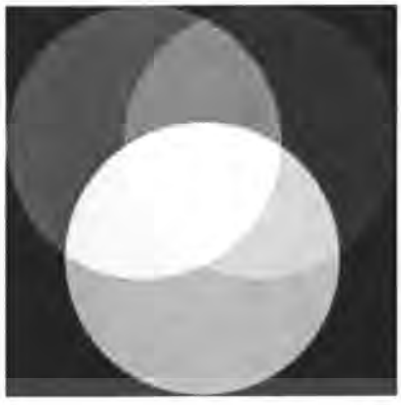
- ความรู้สึกในเรื่องการส่องสว่าง (Brightness or Value) จะทำให้รู้สึกว่าแสงนี้สว่างหรือมืด

- ความรู้สึกในเรื่องแสงสีอิ่มตัว (Saturation or Chrome) จะทำให้รู้สึกถึงความบริสุทธิ์ของแสงสีได้ว่าเข้มหรือจาง

2.7 การผสมและการแยกแสงสี

การผสมแสงสีจะมีลักษณะแตกต่างจากการผสมแม่สีที่ใช้ในภาพวาด การผสมสีโดยทั่วไปแล้วในสีที่ใช้วาดเขียนนั้นเมื่อผสมแล้วจะทำให้สีที่ได้นั้นมีความเข้มข้นมากกว่าเดิม ซึ่งเป็นลักษณะการผสมแบบ Subtractive mixture ส่วนการผสมแสงสีนี้จะเป็นการผสมแม่สี (primary color) เพื่อทำให้เกิดสีต่างๆ ขึ้น ซึ่งแม่สีนี้จะต้องเป็นสีอิสระคือ ไม่สามารถนำสีอื่นมาผสมเป็นสีนั้นได้ แม่สีของแสงนั้นมีอยู่ด้วยกัน 3 สีคือ แดง เขียวและสีน้ำเงิน นอกจากนั้นแสงสีอื่นที่มองเห็นนั้นได้มาจากการผสม ดังนี้

แสงสีเหลือง(Yellow)	ได้มาจากแสงสีแดง+แสงสีเขียว
แสงสีฟ้าอมเขียว(Cyan)	ได้มาจากแสงสีเขียว+แสงสีน้ำเงิน
แสงสีม่วง(Magenta)	ได้มาจากแสงสีน้ำเงิน+แสงสีแดง
แสงสีขาว(White)	ได้มาจากแสงสีแดง+แสงสีเขียว+แสงสีน้ำเงิน



รูปที่ 2.4 การผสมแสงสี RGB

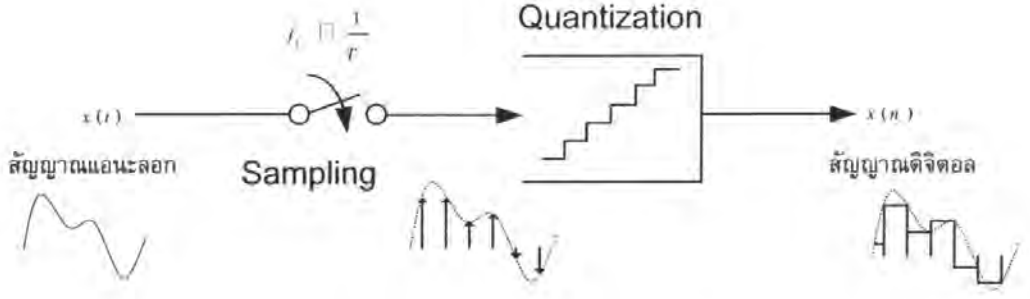
2.8 การสุ่ม (Sampling) และ การควอนไตซ์ (Quantization)

ความถี่ที่ใช้ในการสุ่มในระบบภาพนั้นจะเป็นตัวบ่งบอกถึงขนาดภาพ ภาพที่ผ่านการสุ่มด้วยความถี่สูงๆ ก็จะได้จุดภาพที่ละเอียดมากขึ้น สามารถเก็บรายละเอียดได้มากขึ้น ในกรณีที่จุดภาพมีค่าน้อยจะทำให้เกิดผลอย่างหนึ่งคือ การเกิดซ้ำกันของจุดภาพ (Pixel Replication) ทำให้เห็นภาพเป็นบล็อกๆ (Checker-Board Effect)

การควอนไตซ์เป็นการเข้ารหัสของระดับที่ผ่านการสุ่ม เพื่อจัดเข้าระดับที่เป็นมาตรฐานหรือเป็นไปตามที่ต้องการ แต่ในทางด้านการประมวลผลภาพการควอนไตซ์เป็นการจัดระดับของสัญญาณภาพที่ผ่านการสุ่มให้อยู่ในระดับเทา จำนวนระดับเทาที่ใช้นั้นเท่ากับสองยกกำลังตามจำนวนบิต แสดงตามสมการดังนี้

$$G = 2^m$$

เมื่อ G เท่ากับจำนวนระดับเทา และ m เป็นจำนวนบิตข้อมูลที่ใช้ ตัวอย่างเช่น ตัวแปลงสัญญาณอนาล็อกเป็นดิจิตอลที่ให้ข้อมูลจากการแปลงแล้ว 8 บิต ทำให้ได้ระดับเทาที่แตกต่างกัน 256 ระดับ ระดับการควอนไตซ์นั้นจะมีผลต่อภาพที่เก็บ ถ้าใช้ระดับการควอนไตซ์ที่มีจำนวนระดับความแตกต่างน้อยหรือกล่าวอีกนัยหนึ่งคือ จำนวนบิตข้อมูลดิจิตอลที่น้อยกว่าปกตินั้นจะทำให้เกิดความผิดพลาดของข้อมูลสูง สาเหตุที่เป็นเช่นนี้เพราะว่าความห่างของระดับนั้นมีมาก เวลาทำการควอนไตซ์จะเกิดการปรับค่าที่ได้จากการสุ่มให้เข้าสู่ระดับที่กำหนด ถ้าข้อมูลที่ได้จากการสุ่มห่างจากระดับที่กำหนดมากเท่าใด ก็จะทำให้เกิดความผิดพลาดมากขึ้นเท่านั้นหรือกล่าวอีกนัยหนึ่งคือ เราไม่มีระดับเทาที่แทนค่าของระดับความเข้มของภาพได้ทั้งหมด ส่วนจำนวนระดับเทาหรือบิตของข้อมูลภาพที่ใช้นั้นปกติไม่ควรต่ำกว่า 64 ระดับเทา หรือจำนวนบิตไม่ควรต่ำกว่า 6 บิต จึงเหมาะสมกับสายตาของคนเราที่จะไม่รู้สึกว่าเกิดความคลาดเคลื่อนขึ้นกับภาพ แต่ถ้าใช้จำนวนบิตที่ต่ำกว่านี้จะทำให้เกิดผลอย่างหนึ่งที่เรียกว่า ขอบเทียม (false contour) แม้มีการใช้บิตของจุดภาพที่น้อยลง แต่สายตาเราก็ยังไม่สามารถตรวจจับความแตกต่างของภาพได้ แต่ถ้ามีการลดจำนวนบิตของจุดภาพลงไปอีก จะทำให้เราสามารถจับความผิดเพี้ยนของภาพที่เกิดขึ้นได้

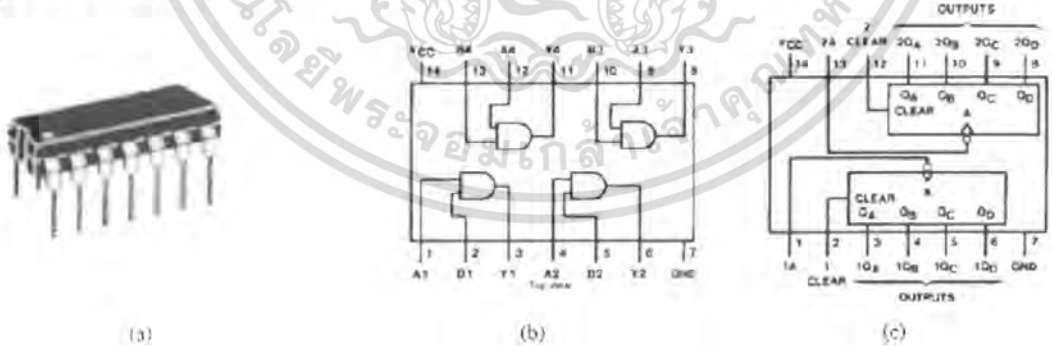


รูปที่ 2.5 กระบวนการสุ่มสัญญาณและการควอนไตซ์

2.9 การออกแบบวงจรดิจิตอลด้วย FPGA และ CPLD

2.9.1 ไอซีมาตรฐาน

การออกแบบวงจรดิจิตอลขนาดเล็กๆ โดยปกติจะนิยมใช้ชิพหรือไอซีมาตรฐานเช่น ไอซีCMOS ตระกูล4000 และ 7HC00 หรือใช้ไอซี TTL ตระกูล 74LS00 เป็นต้น โดยมีตัวอย่างลักษณะไอซีแสดงดังรูปที่ 2.6a เช่น CMOS เบอร์ 74LS08 (มีแอนด์เกตแบบ 2 อินพุตอยู่ภายใน 4 ตัว) แสดงดังรูปที่ 2.6b และเบอร์ 74HC393 (มีวงจรนับฐานสองหรือเลขไบนารี 4 บิตอยู่ภายใน 2 ตัว) แสดงดังรูปที่ 2.6c จะเห็นได้ว่าไอซีสำเร็จรูปเหล่านี้จะมีฟังก์ชันการทำงานทางลอจิกแบบตายตัวและเป็นวงจรขนาดเล็กๆอยู่ภายในเพียงไม่กี่ตัว จึงไม่เหมาะกับงานออกแบบวงจรขนาดใหญ่หรือวงจรความถี่สูง เนื่องจากเกิดเวลาล่าช้าขึ้นในตัวไอซีและสายสัญญาณ โดยที่สัญญาณต่างๆที่วิ่งอยู่ในสายเส้นทองแดงของ PCB หรือสายสัญญาณนั้นความเร็ว (ค่ากลางๆ ประมาณครึ่งหนึ่งของความเร็วแสงหรือ 15-18 เซนติเมตรต่อนาโนวินาที) การออกแบบวงจรที่ใช้ความถี่สูงหลายสิบเมกกะเฮิร์ตและแผงวงจรมีขนาดใหญ่โดยไอซีหลายๆตัวจึงมีความยุ่งยากมากและอาจทำไม่ได้



(a) (b) (c)

รูปที่ 2.6 ตัวอย่างไอซีตระกูล TTL หรือ CMOS

ต่อมาได้มีคิดค้นไอซี หรือ ชิพดิจิตอลเอนกประสงค์ที่สามารถโปรแกรมให้เป็นฟังก์ชันทำงานตามที่ต้องการได้ โดยที่ภายในชิพจะบรรจุวงจรลอจิกพื้นฐานที่มีฟังก์ชันทำงานแบบไม่ตายตัวไว้เป็นจำนวนมากเรียกว่า Programmable Logic Device = PLD เป็นผลสำเร็จประมาณปี 1970 ปัจจุบันได้มีการออกแบบวงจรขนาดใหญ่โดยใช้ชิพดิจิตอลเอนกประสงค์แทนการออกแบบด้วยไอซีมาตรฐานตระกูล

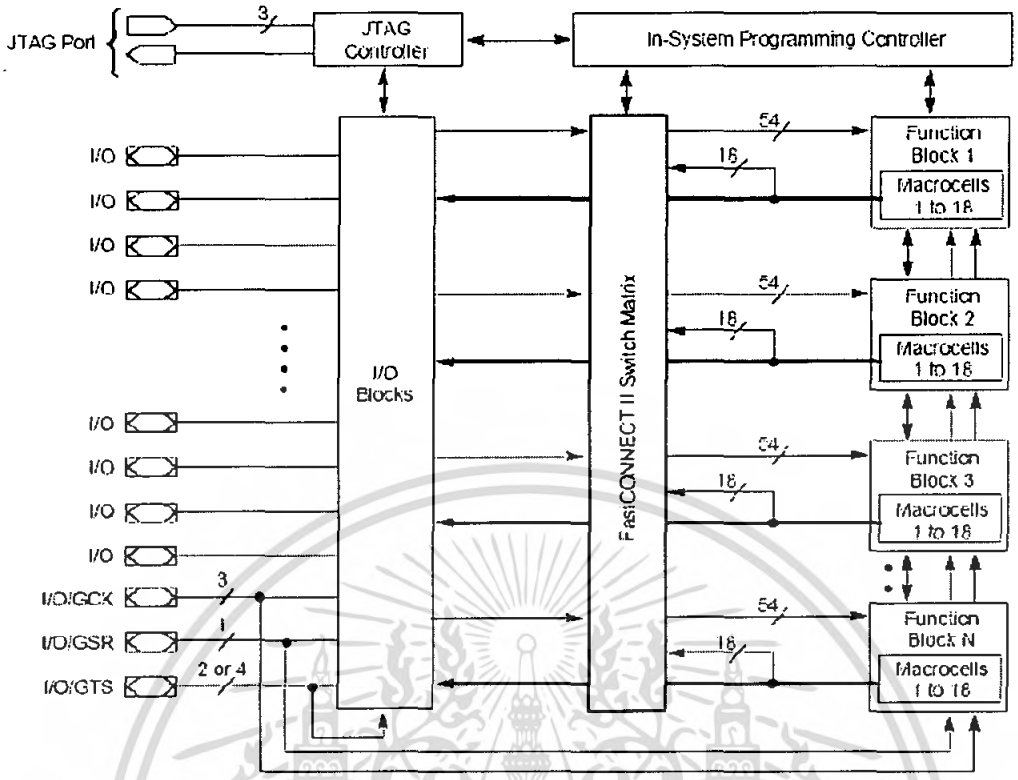
TTL หรือ CMOS กันมากขึ้น บ่อยครั้งเราจะพบชิพ FPGA (Field Programmable Gate Array) และ/หรือ CPLD (Complex Programmable Logic Device) เป็นส่วนประกอบที่ติดตั้งอยู่บนแผงวงจร เช่น ทางด้านสื่อสาร การแพทย์ การทหาร ระบบเครือข่าย หรือเครื่องมือวัดต่างๆ เป็นต้น

การที่เราออกแบบวงจรย่อยส่วนต่างๆ รวมไว้ใน FPGA และ/หรือ CPLD เพียงตัวเดียว หรือเพียงไม่กี่ตัวได้นั้นจะทำให้ แผงวงจรมีขนาดลดลงอย่างมาก ทำให้ได้วงจรจะทำงานได้เร็วขึ้นเพราะสายสัญญาณต่างๆสั้นลง และสายสัญญาณส่วนใหญ่จะอยู่ภายในชิพทำให้ได้ผลิตภัณฑ์ที่มีขนาดเล็กกะทัดรัด

2.9.2 CPLD

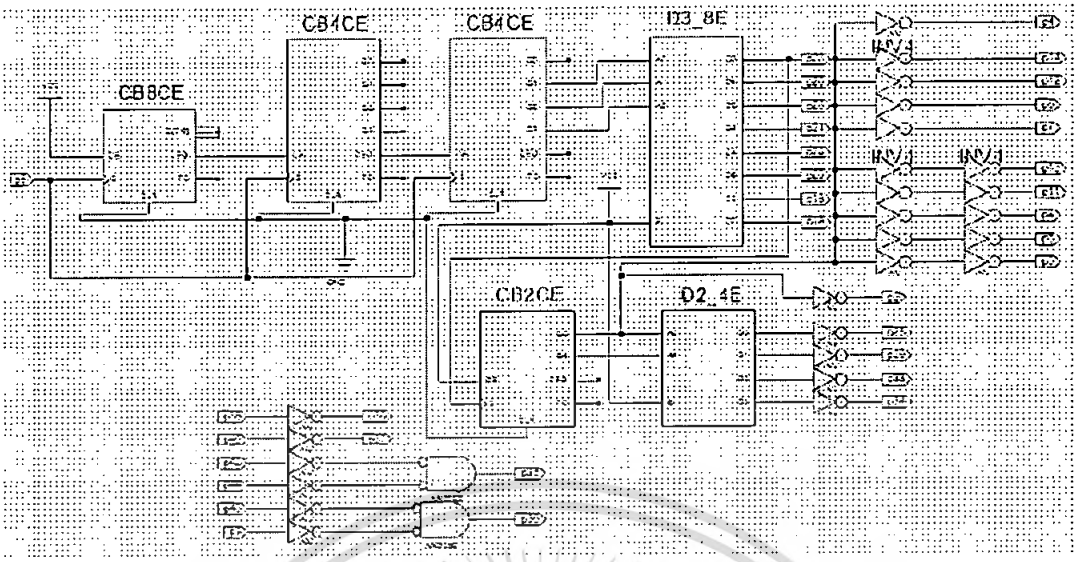
CPLD เป็นไอซีประเภท PLD ซึ่งเป็นชิพดิจิตอลเอนกประสงค์ชนิดหนึ่งที่สามารถโปรแกรมให้มีฟังก์ชันทำงานตามที่ต้องการได้ ตัวอย่าง CPLD เบอร์ XC9536XL ที่มีความจุวงจร 800 เกตและมี 34 อินพุตเอาต์พุต (I/O) แสดงดังรูปที่ 2.7 โครงสร้างภายในชิพ CPLD ตระกูล XC9500XL แสดงดังรูปที่ 2.8

รูปที่ 2.7 ตัวอย่าง CPLD เบอร์ XC9536XL ที่มีความจุวงจร 800 เกต



รูปที่ 2.8 โครงสร้างภายในของ CPLD ตระกูล XC9500XL

โครงสร้าง CPLD ประกอบส่วนหลักๆ คือ Function Block (FB) และ I/O Block (IOB) ที่สามารถเชื่อมต่อถึงกันด้วย Switch matrix ภายใน Function Block จะประกอบไปด้วยวงจรลอจิกพื้นฐานต่างๆที่สามารถโปรแกรมเป็นวงจรดิจิตอลได้ตามต้องการ ส่วน I/O Block จะทำหน้าที่เป็นบัฟเฟอร์ที่อินพุตหรือเอาต์พุตของตัวชิพ วงจร In-system programming ใช้สำหรับโปรแกรมชิพผ่านทางพอร์ต JTAG โดยมี JTAG Controller เป็นตัวควบคุม ตัวอย่างวงจรรูปที่ 2.9 นั้นอาจสร้างได้โดยใช้ไอซีมาตรฐานตระกูล 7400 ไม่น้อยกว่า 10 ตัว แต่เมื่อสร้างวงจรนี้ไว้ใน CPLD เบอร์ XC9572XL จะกินความจุวงจรสูงสุดเพียง 62% เท่านั้น (ยกเว้นขา I/O ที่มีการใช้ทุกขา) แสดงดังรูปที่ 2.10 แม้ว่า CPLD จะสามารถสร้างวงจรดิจิตอลทั่วๆไปได้มากมาย แต่ต้องถือว่า CPLD นั้นมีความจุวงจรต่ำมากเมื่อเทียบกับชิพ FPGA ซึ่งอธิบายในหัวข้อต่อไป โดยทั่วๆไป CPLD จะมีความจุวงจรไม่เกิน 10,000 เกต ข้อจำกัดของ CPLD คือใช้เวลาในการโปรแกรมค่อนข้างนานเนื่องจากมีโครงสร้างตัวเก็บข้อมูลภายในจำพวก EEPROM ซึ่งเขียนข้อมูลลงไปได้ช้าและราคาแพง (ต่อเกตแพงกว่า FPGA) แต่เมื่อโปรแกรมวงจรไว้ใน CPLD แล้วข้อมูลวงจรนั้นก็จะคงอยู่แม้ว่าจะไม่มีไฟเลี้ยงตัวชิพแล้วก็ตาม การโปรแกรมสามารถทำซ้ำได้หลายๆครั้ง



รูปที่ 2.9 ตัวอย่างวงจรดิจิตอล

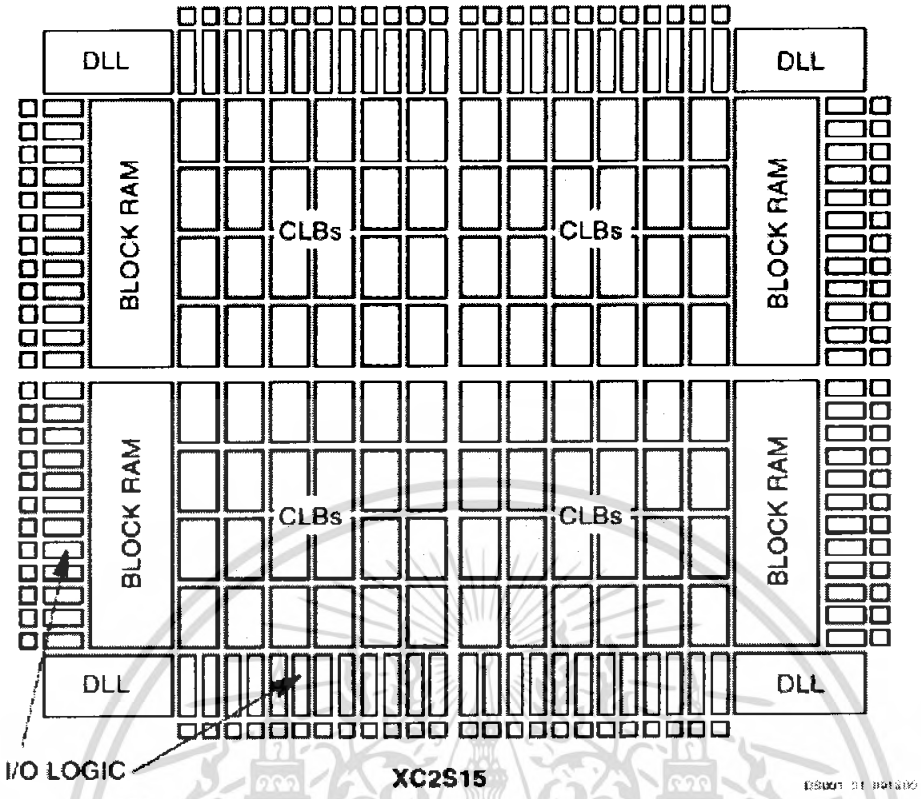
RESOURCES SUMMARY

Macrocells Used	Pterms Used	Registers Used	Pins Used	Function Block Inputs Used
44/72 (62%)	60/360 (17%)	28/72 (39%)	34/34 (100%)	46/216 (22%)

รูปที่ 2.10 แสดงรายการอุปกรณ์ที่ต้องใช้ใน CPLD

2.9.3 FPGA

FPGA เป็นชิพประเภท PLD เช่นเดียวกันจึงเป็นชิพเอนกประสงค์ที่สามารถโปรแกรมให้เป็นวงจรดิจิตอลได้ตามต้องการ แต่จะมีโครงสร้างภายในแตกต่างจาก CPLD อย่างสิ้นเชิง และซับซ้อนกว่ามาก ตัวอย่างโครงสร้างภายในของ FPGA ตระกูล Spartan-II เบอร์ XC2S15 แสดงดังรูปที่ 2.11



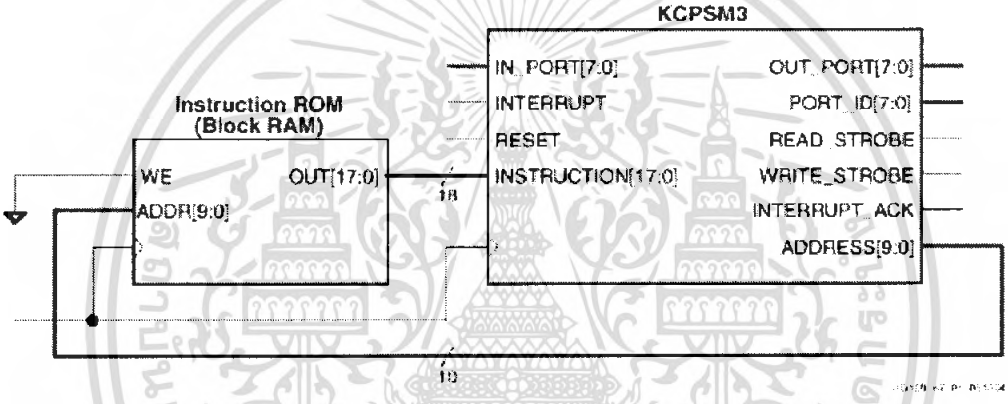
รูปที่ 2.11 โครงสร้างของ FPGA Spartan-II

ซึ่งโครงสร้างภายในของ FPGA ประกอบด้วย 2 ส่วนหลักๆ คือ Configurable Logic Block (CLB) ต่างๆและ I/O LOGIC เพื่อใช้โปรแกรมให้เป็นวงจรดิจิตอลตามต้องการ การเชื่อมต่อถึงกันภายในชิพมีหลากหลายลำดับชั้น นอกจากนี้ยังมีอุปกรณ์ภายในที่ทำหน้าที่เฉพาะงาน เช่น Delay-Locked Loop (DLL) และหน่วยความจำ (Block RAM) เพื่ออำนวยความสะดวกในหลักสิบล้านเกตขึ้นไปขึ้นอยู่กับเทคโนโลยีที่ในการผลิต การโปรแกรม FPGA สามารถทำได้โดยการโหลดข้อมูลวงจร (Configuration data) ลงไปเก็บที่เซลหน่วยความจำแบบ RAM (เป็นคนละส่วน Block RAM) ที่อยู่ภายใน FPGA ดังนั้น FPGA จึงไม่มีข้อจำกัดในการโปรแกรมซ้ำและสามารถโปรแกรมข้อมูลวงจรลงไปได้เร็ว แต่ RAM มีข้อเสียที่สำคัญคือข้อมูลวงจรจะสูญหายหากไม่มีไฟเลี้ยง จึงต้องใช้หน่วยความจำภายนอกชิพที่สามารถเก็บข้อมูลวงจรอยู่ได้แม้ว่าจะไม่มีไฟเลี้ยงในตัวชิพ เช่น Serial PROM เพื่อใช้เก็บข้อมูลและจะทำการโหลดข้อมูลจาก Serial PROM ลงชิพ FPGA อย่างอัตโนมัติทุกครั้งที่มีการจ่ายไฟเลี้ยง

2.9.4 ข้อเปรียบเทียบระหว่าง FPGA และ CPLD กับไมโครคอนโทรลเลอร์

นักออกแบบวงจรดิจิตอลส่วนใหญ่จะคุ้นเคยกับการไมโครคอนโทรลเลอร์กันดีอยู่แล้ว เช่น ตระกูล MCS-51 และ ตระกูล PIC เป็นต้น ซึ่งมีการออกแบบสถาปัตยกรรมและชุดคำสั่งไว้ภายในเรียบร้อยแล้ว จึงสามารถนำชุดคำสั่งต่างๆมาโปรแกรมให้ไมโครคอนโทรลเลอร์ทำงานได้ตามที่ต้องการ ซึ่งต่างจาก FPGA และ CPLD ที่จะนำไปใช้กับงานด้านออกแบบฮาร์ดแวร์ดิจิตอลเป็นหลัก โดยเฉพาะ

อย่างยิ่งงานที่เป็นวงจรขนาดใหญ่หรือความถี่สูงๆ ซึ่งไมโครคอนโทรลเลอร์เองก็สามารถสร้างโดยใช้ FPGA ได้เช่นกัน ปัจจุบันได้มีการออกแบบไมโครคอนโทรลเลอร์แบบฝังตัว (Embedded microcontroller) และ วงจรดิจิตอลส่วนต่างๆรวมไว้ใน FPGA ทำให้ได้วงจรที่ทำงานได้เร็วและมีความยืดหยุ่นสูง ตัวอย่างการออกแบบไมโครคอนโทรลเลอร์แบบฝังตัวไว้ใน FPGA ตระกูล Spartan-3 เบอร์ XC3S200 แสดงดังรูปที่ 2.12 โดยจะกินพื้นที่วงจรใน FPGA ประมาณ 4-8 % และสามารถ RUN ที่ความถี่สูงมากถึง 87 MHz ซึ่งไมโครคอนโทรลเลอร์ที่สร้างจาก FPGAจะมีความเร็วสูงกว่าไมโครคอนโทรลเลอร์ทั่วไปที่มีขายในท้องตลาด ประมาณ 2-20 เท่า และสร้างไว้ใน FPGA ได้หลายๆตัว การออกแบบวงจรดิจิตอลจำนวนมากไว้ใน FPGA ทำให้แผงวงจรมีขนาดเล็กลงอย่างมาก วงจรจะทำงานได้เร็วขึ้นและมีความน่าเชื่อถือได้สูง เพราะสายสัญญาณต่างๆ จะสั้นลงและส่วนใหญ่จะอยู่ภายในชิพ FPGA โดยไม่มีจุดเชื่อมวงจรต่ออยู่ภายนอก



รูปที่ 2.12 ตัวอย่างการออกแบบไมโครคอนโทรลเลอร์ตระกูลพิโคแบบฝังตัวใน FPGA Spartan-3

2.9.5 กระบวนการออกแบบวงจร

ในกระบวนการออกแบบวงจรโดยทั่วไปมักจะมีขั้นตอนหรือการทำงานเป็นขั้นตอนดังนี้

- System Requirement
- System Design
- System Implementation
- Testing and Debugging
- Documentation

กระบวนการทำ System Requirement คือ การหาความต้องการว่าเราต้องการสร้างวงจรอะไร วงจรทำงานอย่างไร และ มีอินพุตเอาต์พุตอะไรบ้าง กระบวนการทำ System Design คือการออกแบบ และหาวิธีแก้ไขปัญหาจากขั้นตอนแรก จากนั้นจึงเป็นการทำ System Implementation เพื่อสร้างวงจรตามที่ได้ออกแบบไว้ แล้วจึงทำวงจรเพื่อทดสอบ (Testing) และหากมีปัญหาก็ทำการแก้ไข (Debugging) จากนั้นจึงทำ Documentation เพื่อสร้างเอกสารอธิบายการทำงานของวงจรที่ออกแบบทั้งหมด

การออกแบบวงจรดิจิตอลโดยใช้ FPGA และ CPLD สามารถทำได้หลายวิธีได้แก่ วิธีวาดผังวงจร (Schematic) เขียนบรรยายพฤติกรรมการทำงานของวงจรด้วยภาษา HDL (Hardware Description Language) และเขียนแผนภูมิสถานะ (State diagram) การออกแบบด้วยวิธีวาดผังวงจรเหมาะสำหรับการออกแบบวงจรเล็กๆ การออกแบบวงจรดิจิตอลขนาดใหญ่นั้นไม่เหมาะกับการออกแบบด้วยวิธีวาดผังวงจรเนื่องจากใช้เวลานานและตรวจสอบข้อผิดพลาดได้ยาก การออกแบบด้วยภาษาHDL ซึ่งเป็นภาษาระดับสูง เช่น ภาษา VHDL และ Verilog เป็นต้น จึงเสมือนเป็นเครื่องทุ่นแรงชั้นเยี่ยมที่ใช้ในการออกแบบวงจรดิจิตอลขนาดใหญ่ เพราะเป็นการใช้ความสามารถของซอฟแวร์ทูลช่วยในการออกแบบโดยไม่จำเป็นต้องคิดรายละเอียดของวงจรในระดับเกต

2.9.6 VHDL

VHDL ย่อมาจาก VHSIC Hardware Description Language โดยที่ VHSIC = Very High Speed Integrated Circuit ซึ่งพัฒนาโดยกระทรวงกลาโหมสหรัฐอเมริกาและประกาศเป็นมาตรฐาน IEEE 1076-1987 ในปี ค.ศ.1987 (VHDL87) มีความสมบูรณ์ยิ่งขึ้น ต่อมา IEEE1076-1987 ได้ปรับปรุงเป็น IEEE 1076-1993 (VHDL93) และเนื่องจากการปรับปรุง IEEE 1076-1993 เป็นไปอย่างล่าช้าจึงไม่ได้รวบรวมเอาคุณสมบัติใหม่ๆ ที่สำคัญที่ใช้สำหรับการสังเคราะห์วงจรเข้าไปด้วย ด้วยเหตุนี้จึงได้มีการนิยาม Numeric standard หรือ Synthesis standard เพิ่มเติมเป็นมาตรฐาน IEEE 1076.3 ออกมาในปลายปี ค.ศ. 1995

VHDL เป็นภาษาระดับสูงที่ใช้ในการออกแบบระบบและวงจรดิจิตอลโดยมีรูปแบบคล้ายกับภาษาปาสคาล เป็นภาษาที่สามารถอ่านเข้าใจได้ง่าย สามารถออกแบบวงจรในระดับต่างๆได้กลายระดับโดยการออกแบบจะอยู่ในรูปแบบของฟังก์ชั่นเท่านั้น การเปลี่ยนแปลงแก้ไขวงจรหรือนำกลับมาใช้ใหม่จึงทำได้ง่าย ทั้งยังมีซอฟต์แวร์ทูลช่วยในการตรวจสอบความถูกต้องของวงจรที่ออกแบบ (Verification) โดยจำลองการทำงาน ซึ่งสามารถดูผลลัพธ์ของวงจรได้โดยจำเป็นต้องทดลองกับวงจรจริง

2.9.7 องค์ประกอบพื้นฐานของ VHDL

รูปแบบพื้นฐานที่ใช้ในการบรรยายถึงองค์ประกอบของ VHDL จะประกอบไปด้วยส่วนกำหนดการเชื่อมต่อ (Interface) และส่วนกำหนดลักษณะเชิงสถาปัตยกรรม (Architecture) โดยในการบรรยายการเชื่อมต่อจะขึ้น ต้นด้วยคำว่า ENTITY แล้วตามด้วยชื่อขององค์ประกอบจากนั้นตามด้วยคำว่า IS และ ถัดมาจะเป็นการบรรยายถึงพอร์ตการติดต่ออินพุท-เอาท์พุท ขององค์ประกอบส่วนลักษณะภายนอกอื่นๆ เช่น เวลา อุณหภูมิก็สามารถรวมเข้าไปในส่วนนี้ ได้เช่นกัน ในส่วนของการกำหนดลักษณะเชิงสถาปัตยกรรมจะขึ้นต้นด้วยคำว่า ARCHITECTURE ซึ่งเป็นส่วนที่ใช้ บรรยายหน้าที่การทำงานขององค์ประกอบ โดยหน้าที่การทำงานนี้จะขึ้นอยู่กับสัญญาณอินพุท เอาท์พุทและพารามิเตอร์ อื่นๆ ที่ได้กำหนดไว้ในส่วนของการเชื่อมต่อดังรูปที่ 2.13 และสำหรับการบรรยายหน้าที่ขององค์ประกอบจะเริ่มต้นหลังจาก คำว่า BEGIN เป็นต้นไป

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

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

รูปที่ 2.13 การกำหนดการเชื่อมต่อและสถาปัตยกรรม

2.9.7.1 การกำหนดการเชื่อมต่อ

การกำหนดการเชื่อมต่อเป็นระดับบนสุดของการออกแบบ โดยในระดับนี้ต้องกำหนดพอร์ตสำหรับการติดต่อกับองค์ประกอบ ภายนอกอื่นๆ ดังตัวอย่างในรูปที่ 2.14 ซึ่งเป็นบล็อกไดอะแกรม และการบรรยายการเชื่อมต่อขององค์ประกอบสำหรับตัวจ่าย สัญญาณนาฬิกา ในบรรทัดแรกของการบรรยายการเชื่อมต่อเป็นการกำหนดชื่อขององค์ประกอบซึ่งกำหนดเป็น clock_ component ตามด้วยคำว่า PORT และชื่อของพอร์ตอยู่ภายในวงเล็บ ส่วน IN และ OUT เป็นการกำหนด โหนดของสัญญาณให้เป็นอินพุทหรือเอาท์พุท และ BIT เป็นการแสดงชนิดของข้อมูล


```
ENTITY clock_component IS
    PORT (en : IN BIT;ck : OUT BIT)
END clock_name;
```

รูปที่ 2.14 บล็อกไดอะแกรมและการบรรยายการเชื่อมต่อของ clock_ component

2.9.7.2 การกำหนดรูปแบบการบรรยาย

หน้าที่การทำงานขององค์ประกอบจะถูกบรรยายภายในส่วนนี้ ซึ่งในการบรรยายสามารถกำหนดค่าของสัญญาณ เอาท์พุทในเทอมของอินพุทหรือในรูปขององค์ประกอบอื่นๆ หรือทั้งสองอย่างรวมกันก็ได้ ดังตัวอย่างการบรรยายของ clock_ component ในรูปที่ 2.15 ซึ่งเป็นการบรรยายในเชิง

พฤติกรรมโดยมี en เป็นอินพุทและ ck เป็นเอาท์พุท PROCESS เป็นคำที่ใช้ในการเริ่มต้นสำหรับการบรรยายในเชิงพฤติกรรม และภายในโปรเซสกำหนดให้ periodic เป็นตัวแปรที่มีค่าเริ่มต้นเป็น "0" ถ้าสัญญาณ en มีค่าเป็น "1"' จะทำให้ตัวแปร periodic ถูกคอมพลีเมนท์ (complement) และส่งค่าให้กับ ck ซึ่งเป็นสัญญาณเอาท์พุท และสำหรับคำสั่ง WAIT จะเป็นการกำหนดให้สัญญาณมีคาบเวลาเท่ากับ 1 ไมโครวินาที

```
ARCHITECTURE behavioral OF clock_component IS

BEGIN
   PEOCESS
      VARIABLE periodic : BIT := '0';
   BEGIN
      IF en='1' THEN
       periodic :=  Not periodic;
      END IF;
       ck <= periodic;
       WAIT FOR 1 US;
  END PROCESS;
END behavioral;
```

รูปที่ 2.15 การบรรยายเชิงพฤติกรรมของ clock _ component

2.9.7.3 หน่วยการออกแบบแพ็กเก็ต

ข้อมูลต่างๆ ตลอดจนโปรแกรมย่อย ที่เป็นประโยชน์ต่อการเขียนรูปแบบการบรรยายระบบดิจิตอล สามารถเก็บไว้ใน ส่วนของแพ็กเก็ต ซึ่งหน่วยการออกแบบต่างๆ เช่น หน่วยการออกแบบ Entity หน่วยการออกแบบสถาปัตยกรรมหรือ หน่วยการออกแบบแพ็กเกจอื่นๆ สามารถเรียกข้อมูลเหล่านี้ไปใช้ได้ นอกจากนั้นสิ่งที่นิยมทำกันมากคือการนำรูปแบบ มาตรฐานต่างๆ เช่น อุปกรณ์มาตรฐาน (เช่น ไอซีตระกูล 74XX เป็นต้น) มาเก็บไว้ในรูปของแพ็กเกจ ที่ทุกคนสามารถ เข้าถึงได้ ตามปกติแล้วแพ็กเกจจะแบ่งออกเป็น 2 ส่วนคือ การประกาศแพ็กเกจ (Package declaration) และ ส่วนของบอดี้แพ็กเกจ (Package body) เนื่องจาก แพ็กเกจถูกสร้างขึ้นเป็นส่วนแยกต่างหากออกจากรูปแบบที่ กำลังเขียนอยู่ ฉะนั้นการที่นำแพ็กเกจไปใช้นั้นจะต้องมีการเชื่อมโยงหรืออ้างอิงเสียก่อน ซึ่งในภาษา VHDL สามารถ กระทำได้ด้วยชุดคำสั่ง USE

- PACKAGE DECLARATION

ส่วนที่มีความสำคัญที่สุดของแพ็กเกจ (ถ้ามองในแง่ของการนำไปใช้จากภายนอก) ได้แก่ส่วนการประกาศแพ็กเกจ เนื่องจากเป็นส่วนที่ใช้กำหนดชื่อของสิ่งที่ประกาศอยู่ภายในแพ็กเกจ สำหรับนำไปใช้ภายนอกตัวของแพ็กเกจเอง ถ้ามี การประกาศสิ่งใดๆ ในส่วนของส่วนบอดี้แพ็กเกจ แต่ไม่ถูกประกาศในส่วนการประกาศแพ็กเกจจะทำให้ค่าและพฤติ กรรมไม่สามารถนำไปใช้งานในส่วนนอกได้ซึ่งเปรียบเทียบได้กับสิ่งที่ประกาศไว้ในส่วนของการประกาศ Entity คือ จุดเชื่อมต่อ หรือ พอร์ต ที่มีหน้าที่

ติดต่อกับโลกภายนอก ฉะนั้นโดยทั่วไปแล้วแพ็กเกจสามารถสร้างขึ้นได้โดยไม่จำเป็น ต้องมีส่วนบอดี้ และยังสามารถนำไปใช้งานจากรูปแบบภายนอกได้เช่น ใช้สำหรับประกาศ ชนิด (Type) หรือสัญญาณ เช่นเดียวกับ ส่วนบอดี้แพ็กเกจที่ไม่จำเป็นต้องมี ส่วนของการประกาศแพ็กเกจ แต่แพ็กเกจนั้นจะไม่สามารถนำไปใช้จาก รูปแบบอื่นได้

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

รูปที่ 2.16 โครงสร้างทั่วไปของส่วนการประกาศแพ็กเกจ

- PACKAGE BODY

โครงสร้างซึ่งประกอบด้วยลำดับคำสั่งที่ใช้บรรยายฟังก์ชันการทำงานของโปรแกรมย่อยทั้งหลาย ซึ่งชื่อของโปรแกรมย่อยนั้นๆ ได้ถูกประกาศไปแล้วในส่วนของการประกาศแพ็กเกจ จะถูกเก็บไว้ในส่วนของบอดี้แพ็กเกจ ทั้งนี้รวมถึง การกำหนดค่าคงที่ต่างๆ อันได้แก่ค่าคงที่ที่ถูกประกาศชื่อไว้ก่อนในส่วนของการประกาศแพ็กเกจ และถูกกำหนดค่าใน ส่วนของบอดี้แพ็กเกจ ฉะนั้นในส่วนของบอดี้แพ็กเกจจึงไม่จำเป็นต้องมี ถ้าในส่วนของการประกาศแพ็กเกจไม่มีการ ประกาศชื่อที่เป็นโปรแกรมย่อย หรือค่าคงที่ การเขียนบอดี้แพ็กเกจนั้นจะเป็นไปตามกฎเกณฑ์ดังแสดงในรูปที่ 2.17

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

รูปที่ 2.17 โครงสร้างของบอดี้แพ็กเกจ

2.9.7.4 หน่วยการออกแบบ Configuration

ดังที่ทราบกันแล้วว่าระบบดิจิตอลรูปแบบหนึ่งไม่ว่าจะเป็นอะไรก็ตาม จะสามารถมีหน่วยการออกแบบ Entity ได้ เพียงหนึ่งเดียวเท่านั้น ซึ่งในหน่วยการออกแบบ Entity หนึ่งหน่วยนี้อาจจะมีสถาปัตยกรรมที่เป็นหน่วยรองได้หลาย หน่วย ดังนั้นจะต้องมีหน่วยการออกแบบ Configuration มาเพื่อกำหนดการใช้ Configuration ของการประกอบ Entity กับหน่วยการออกแบบสถาปัตยกรรมหน่วยใดๆ เข้าด้วยกัน

```
COMFIGURTRION identifier OF entiry_name IS
    Configuretion_declarative_part
END ;
```

รูปที่ 2.18 โครงสร้างโดยทั่วไปของหน่วยการออกแบบโครงแบบ

2.9.7.5 โปรแกรมย่อย

การใช้ฟังก์ชันและโพรซีเจอร์ใน VHDL เปรียบได้กับการใช้โปรแกรมย่อยในการเขียนโปรแกรมภาษาชั้นสูงทั่วๆ ไปค่าที่ถูกส่งกลับหรือถูกเปลี่ยนแปลงโดยโปรแกรมย่อยอาจจะมีหรือไม่มีผลต่อฮาร์ดแวร์โดยตรงก็ได้ เช่นถ้าใช้ฟังก์ชัน แทนการกระทำในสมการบูลีนก็จะมีผลต่อวงจรลอจิกจริงๆในขณะที่ถ้าใช้โปรแกรมย่อยในการเปลี่ยนชนิดของข้อมูล หรือในการคำนวณค่าการหน่วงเวลาแล้วก็จะไม่มีผลต่อโครงสร้างของฮาร์ดแวร์ รูปที่ 2.19 แสดงการใช้โพรซีเจอร์ เพื่อเปลี่ยนข้อมูลชนิด 8 บิตเป็นค่าจำนวนเต็ม และรูปที่ 2.20 แสดงการใช้ฟังก์ชันโดยกำหนดให้ X เป็นตัวแปรชนิด บิตแทนการกระทำในสมการบูลีน

```
TYPE byte IS ARRAY (7 DOWNTO 0) OF BIT;
  ...
PROCEDURE byte_to_integer (ib : IN byte; oi : OUT INTEGER) IS
        VARIABLE result: INTEGER := 0;
BEGIN
        FOR i IN 0 TO 7 LOOP
                IF ib(i) = '1' THEN
                result := result + 2**i;
                END IF;
        END LOOP;
        oi := result;
END byte_to_integer
```

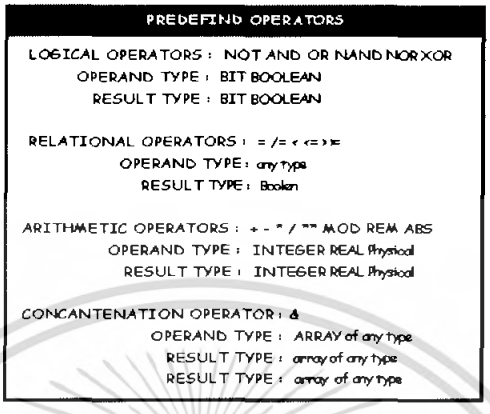
รูปที่ 2.19 การใช้โพรซีเจอร์

```
FUNCTION f (a, b, c:BIT) RETURN BIT  IS
        VARIABLE x:BIT;
BEGIN
        x := ((NOT a) AND (NOT b) AND c);
        RETURN x;
END f;
```

รูปที่ 2.20 การใช้ฟังก์ชัน

2.9.7.6 โอเปอร์เรเตอร์

การบรรยายเชิงพฤติกรรมในภาษา VHDL มีตัวดำเนินการหรือโอเปอร์เรเตอร์ทางลอจิก และคณิตศาสตร์เช่นเดียว กับภาษาซอฟต์แวร์ทั่วไปดังรูปที่ 2.21



รูปที่ 2.21 ตัวดำเนินการใน VHDL

2.9.7.7 เวลาและความพร้อมเพรียง

ในวงจรอิเล็กทรอนิกส์อุปกรณ์ทุกๆ ตัวจะอยู่ในสภาพเตรียมพร้อมเสมอ (Always Active) และจะมีเรื่องของเวลา เข้ามาเกี่ยวข้องในทุกๆเหตุการณ์ที่เกิดขึ้นเสมอ VHDL เป็นภาษาที่ได้รับการออกแบบมาเพื่อให้สามารถบรรยายรูป แบบและการพ้องกันของเวลาสำหรับการทำงานของอุปกรณ์ได้อย่างถูกต้อง การบรรยายการทำงานที่อยู่ภายในส่วน ของการบรรยายสถาปัตยกรรม จะมีการทำงานที่พร้อมเพรียงกันเสมอ หรือแม้แต่โปรเซสซึ่งมีการทำงานภายในเป็น แบบลำดับคำสั่งก็ตาม ซึ่งหากมีหลายๆ โปรเซสอยู่ภายในโครงสร้างเดียวกัน ทุกๆ โปรเซสก็จะทำงานไปพร้อมๆ กัน ด้วย

2.9.7.8 สัญญาณและตัวแปร

สัญญาณมีลักษณะเป็นเสมือนตัวกลางฮาร์ดแวร์ที่ใช้ในการส่งผ่านข้อมูลและมีเรื่องของเวลาเข้ามาเกี่ยวข้องด้วยการ กำหนดค่าให้กับสัญญาณจะใช้สัญลักษณ์ <= ในการส่งค่าและสามารถใช้คำสั่ง AFTER เพื่อกำหนดช่วงเวลาในการ ส่งผ่านค่าของสัญญาณ เช่น w <= a AFTER 12 NS หมายถึงการกำหนดค่าสัญญาณ a ให้กับ w หลังจากเวลา ผ่านไป 12 นาโนวินาที ในทางตรงข้ามตัวแปรมีลักษณะเป็นเสมือนตัวกลางที่ใช้ในการส่งผ่านข้อมูลและไม่มีเรื่องของ เวลาเข้ามาเกี่ยวข้องด้วย ซึ่งตัวแปรจะถูกใช้ในส่วนที่มีการทำงานเป็นแบบลำดับคำสั่งเช่นใน ฟังก์ชัน โพรซีเจอร์ และ โปรเซส สำหรับการกำหนดค่าให้กับตัวแปรจะใช้สัญลักษณ์ :=

2.9.8 การบรรยายเชิงพฤติกรรม

การบรรยายลักษณะการทำงานของอุปกรณ์ฮาร์ดแวร์ในเชิงพฤติกรรม เป็นการบรรยายลักษณะการเปลี่ยนแปลงของข้อมูลในรูปแบบของอัลกอริธึมสำหรับการคำนวณผลลัพธ์ที่เกิดขึ้นซึ่งสืบเนื่อง มาจากการเปลี่ยนแปลงสภาวะของข้อมูล ที่เข้ามาโดยไม่คำนึงถึงลักษณะโครงสร้างหรือความสัมพันธ์ของอุปกรณ์ที่อยู่ภายในว่าจะเป็นอย่างไร ในหัวข้อนี้จะ แสดงถึงการบรรยายเชิงพฤติกรรม แทนการใช้โมดูลฮาร์ดแวร์รวมถึงข้อกำหนดต่างๆ ที่ควรรู้

2.9.9 โปรเซส

โปรเซสเป็นรูปแบบพื้นฐานอย่างหนึ่งที่ใช้ในการกำหนดให้กับสัญญาณ โปรเซสจะอยู่ในสถานะที่เตรียมพร้อมอยู่เสมอ และจะปฏิบัติคำสั่งพร้อมๆ กันกับโปรเซสอื่นๆ ที่อยู่ในสถาปัตยกรรมบรรยายเดียวกัน โดยโปรเซสจะปฏิบัติงานตามคำ สั่งทันทีที่มีเหตุการณ์เกิดขึ้นกับสัญญาณที่อยู่ทางด้านขวามือของสัญลักษณ์กำหนดค่าให้กับสัญญาณ (<=) การบรรยาย โปรเซสจะเริ่มต้นด้วยคำสั่ง PROCESS และจบด้วยคำสั่ง END PROCESS ในรูปที่ 2.22 เป็นการแสดงส่วน ประกอบของการบรรยายแบบโปรเซส ซึ่งประกอบด้วยส่วนของการประกาศตัวแปรที่ต้องใช้และส่วนของการปฏิบัติ คำสั่งเพื่อให้ได้ผลลัพธ์ที่ต้องการ

รูปที่ 2.22 รูปแบบของการบรรยายแบบโปรเซส

2.9.10 การกำหนดตัวดำเนินการภายในโปรเซส

ตัวดำเนินการภายในโปรเซสมี 3 ชนิดคือ ตัวแปร (Variable) ไฟล์ (File) และตัวคงที่ (Constant) ซึ่งตัวดำเนิน การทั้งสามชนิดนี้หากมีการประกาศไว้ในโปรเซสใดก็จะใช้ได้เฉพาะภายในโปรเซสนั้นเท่านั้นสำหรับการติดต่อกับภาย นอกหรือระหว่างโปรเซสสามารถทำได้โดยใช้สัญญาณ (Signal) หรือตัวคงที่ที่ได้ประกาศไว้ในส่วนของ ARCHITECTURE ในรูปที่ 2.23 แสดงตัวอย่างการประกาศตัวกระทำภายในโปรเซส ซึ่งจะอยู่ระหว่างคำสั่ง PROCESS และ BEGIN และค่าเริ่มต้นที่ถูกกำหนดให้กับตัวดำเนินการภายในโปรเซสจะถูกนำมาใช้ในตอนเริ่มต้น ของการปฏิบัติเพียงครั้งเดียวเท่านั้น ต่างกับค่าเริ่มต้นที่อยู่ภายในโปรแกรมย่อยจะถูกนำมาใช้ทุกครั้งที่มีการเรียกใช้ โปรแกรมย่อยนั้น ๆ

```
PROCESS
   FILE flush : TEXT IS IN "filename.dat";
   VARIABLE var : BIT;
   CONSTANT n : INTEGER := 0;
BEGIN
  ...
END PROCESS;
```

รูปที่ 2.23 ตัวอย่างการประกาศตัวดำเนินการภายในโปรเซส

2.9.11 การกำหนดการกระทำภายในโปรเซส

การกระทำใดๆ ภายในโปรเซสจะเป็นการปฏิบัติแบบลำดับ (Sequential) เสมอ ซึ่งภายในโปรเซสสามารถใช้ประโยค เงื่อนไขหรือการทำซ้ำได้เช่น IF-THEN - ELSE,CASE - WHEN, FOR LOOP และ WHILE LOOP ดังตัวอย่างในรูปที่ 2.24 และ 2.25

```
ARCHITECTURE demo OF partial_process IS
  ...
BEGIN
  PROCESS
  ...
   BEGIN
        ....
        x <= '1';
        IF x = '1' THEN
          perform action_1
        ELSE
          perform action_2
        END IF;
        ...
  END PROCESS;
END demo;
```

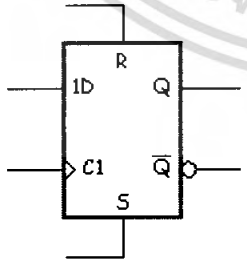
รูปที่ 2.24 เงื่อนไขการกระทำในโปรเซส

```
ARCHITECTURE demo OF partial_process IS
  ...
BEGIN
  PROCESS
    BEGIN
          ...
          x <= a AFTER 10 NS;
          y <= b AFTER 6 NS;
          ...
  END PROCESS;
END demo;
```

รูปที่ 2.25 แสดงการกระทำในโปรเซส

2.9.12 การกระตุ้นและขับขั้งการกระทำของโปรเซส

การกระทำภายในโปรเซสจะอยู่ในสภาพเตรียมพร้อม และมีการปฏิบัติงานอยู่ตลอดเวลาที่มีการเปลี่ยนแปลงของเหตุการณ์ เกิดขึ้น อย่างไรก็ดีเราสามารถกระตุ้นหรือขับขั้งการกระทำภายในโปรเซสได้โดยการกำหนดรายการของสัญญาณที่ต้อง การให้โปรเซสปฏิบัติงานเมื่อมีเหตุการณ์เกิดขึ้นกับสัญญาณที่กำหนดไว้เท่านั้น ส่วนเหตุการณ์ใดๆ ที่เกิดขึ้นกับสัญญาณ ที่ไม่ได้กำหนดไว้ในรายการก็จะไม่ส่งผลให้มีการกระทำภายในโปรเซส ซึ่งรายการของสัญญาณนี้เรียกว่า Sensitivity List และจะกำหนดไว้ภายในวงเล็บหลังคำสั่ง PROCESS รูปที่ 2.26 (a) แสดงตัวอย่างโมเดล และรูปที่ 2.26(b) เป็นตัวอย่างการบรรยายการเชื่อมต่อของ D-Flip Flop ส่วนรูปที่ 2.27 แสดงถึงการบรรยายเชิงพฤติกรรมของ D-Flip Flop โดยในรูปที่ 2.27 (a) เป็นการใช้ตัวกระทำภายนอกโปรเซส และรูปที่ 2.27 (b) เป็นการใช้ตัวกระทำภายในโปรเซส โดยมีรายการของสัญญาณ (rst, set, clk) เป็นตัวกระตุ้นการปฏิบัติงานภายในโปรเซส




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

(a) (b)

รูปที่ 2.26 (a) ตัวอย่างโมเดล D-Flip Flop(b) การบรรยายการเชื่อมต่อของ D-Flip Flop

```
ARCHITECTURE behavioral OF d_sr_flipflop IS
  SIGNAL state : BIT := '0';
BEGIN
   dff : PEOCESS (rst, set, clk)
          BEGIN
        IF set= '1' THEN
            state <= '1' AFTER sq_delay;
          ELSIF rst = '1' THEN
            state <= '0' AFTER rq_delay;
          ELSIF clk = '1' AND clk ' EVENT THEN
            state <= d AFTER cq_delay;
        END IF;
  END PROCESS dff;
  q <= state;
  qb <= NOT state;
END behavioral;
```

(a)

```
ARCHITECTURE average_dalay_ behavioral  OF  d_sr_flipflop IS
BEGIN
   dff : PEOCESS (rst, set, clk)
          VARIABLE state : BIT := '0';
        BEGIN
       IF set= '1' THEN
          state <= '1';
        ELSIF rst = '1' THEN
          state <= '0';
        ELSIF clk = '1' AND clk ' EVENT THEN
          state <= d;
       END IF;
        q <= state AFTER (sq_delay + rq_delay + cq_delay)/3;
        qb <= NOT state AFTER(sq_delay + rq_delay + cq_delay)/3;
  END PROCESS dff;
END behavioral;
```

(b)

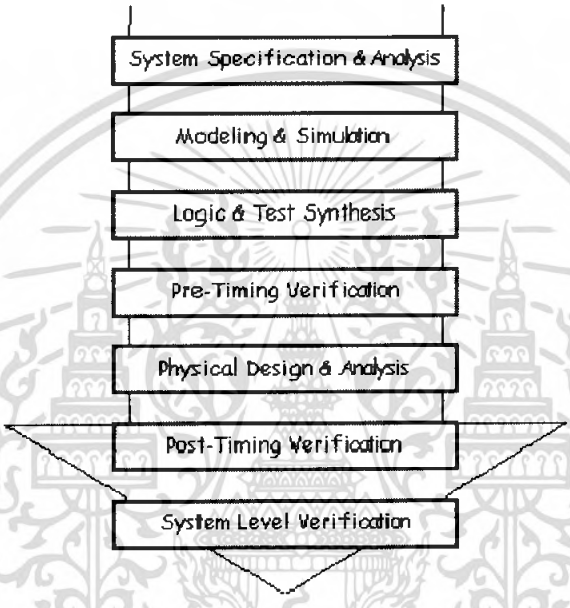
รูปที่ 2.27 การบรรยายเชิงพฤติกรรมของ D-FlipFlop

(a) การใช้ตัวกระทำภายนอกโปรเซส (b) การใช้ตัวกระทำภายในโปรเซส

2.9.13 การออกแบบจากบนลงล่าง

ในการพัฒนาวงจรรวมดิจิตอลขนาดใหญ่ที่มีความซับซ้อน วิศวกรหรือผู้ออกแบบมักจะมองการออกแบบให้อยู่ในรูปของ บล็อกไดอะแกรมก่อนที่ทำวิเคราะห์ให้ลึกถึงรายละเอียดต่อไป ซึ่งภาษา VHDL นั้นอนุญาตให้อธิบายและวิเคราะห์การ ทำงานของแต่ละบล็อก รวมถึงการปรับปรุงการทำงานจากผลที่วิเคราะห์เพื่อให้ได้การทำงานตามต้องการ นอกจากนี้ยัง สามารถเพิ่มเติมในรายละเอียดในแต่ละขั้นตอน

ได้ ซึ่งหลักการนี้สอดคล้องกับหลักการออกแบบจากบนลงล่าง (Top - Down Design) นั่นเอง ถ้าทดลองเปรียบเทียบกับการออกแบบจากล่างขึ้นบน (Bottom - Up Design) จะเห็นได้ว่า การออกแบบจากล่างขึ้นบนจะใช้เวลาการออกแบบมากกว่า 90% เนื่องจากเป็นการวาดวงจรด้วยอุปกรณ์ต่างๆ (Schematic capture) ที่ประกอบกันเข้าเป็นวงจรที่ต้องการออกแบบ ก่อนแล้วจึงทำการจำลองการทำงาน และตรวจ สอบความถูกต้อง ดังนั้นการใช้ภาษา VHDL กับหลักการออกแบบจากบนลงล่างจึงเป็นทางออกให้กับวิศวกรให้สามารถ ออกแบบและพัฒนาวงจรที่มีความซับซ้อนได้มากขึ้น ทั้งยังช่วยลดเวลาและค่าใช้จ่ายในการออกแบบด้วย



รูปที่ 2.28 ขั้นตอนการออกแบบจากบนลงล่าง

จากรูปที่ 2.28 แสดงถึงขั้นตอนของการออกแบบจากบนลงล่าง ทั้งนี้ในทางปฏิบัติอาจมีข้อแตกต่างไปจากนี้บ้าง เล็กน้อยเนื่องจากขั้นตอนของการผลิต (Implementation) สามารถกระทำได้หลายเทคโนโลยี สำหรับรายละเอียดของขั้นตอน การออกแบบจากบนลงล่างในแต่ละขั้นตอนมีดังนี้

1.สร้างข้อกำหนดของความต้องการ และวิเคราะห์ระบบ เพื่อหาแนวความคิดและหลักการ (Idea and Concept) ใน การแก้ปัญหา

2.เขียนรูปแบบของระบบที่ต้องการออกแบบโดยใช้ภาษา VHDL หรือ ภาษา HDL อื่น ๆ สำหรับบรรยายพฤติกรรมการ ทำงาน พร้อมทั้งจำลองการทำงาน เพื่อเปรียบเทียบและตรวจสอบความถูกต้องกับข้อกำหนด

3.หลังจากที่ได้หลักการขั้นต้นพร้อมแนวความคิดที่ผ่านการตรวจสอบแล้วหลักการนี้จะถูกเพิ่มเติมในรายละเอียดลงมา เป็นลำดับขั้นที่สอง จนกระทั่งอยู่ในระดับที่จะนำไปผลิตวงจริง หรือสังเคราะห์ในขั้นตอนนี้เองเทคโนโลยีที่จะมารองรับ วงจรออกแบบจะถูกกำหนดขึ้น และระบบช่วยการ

ออกแบบจะสังเคราะห์วงจรที่ได้จากรูปแบบที่เขียนขึ้นให้อยู่ในรูปของ วงจรที่ประกอบด้วยอุปกรณ์อิเล็กทรอนิกส์ หรือวงจรในระดับเกท และการเชื่อมต่อระหว่างกันของอุปกรณ์เหล่านั้นหรือ ไม่ก็อยู่ในรูปของ Netlist ที่สามารถนำไปผลิตในอุปกรณ์อื่นได้

4.หลังจากการสังเคราะห์วงจรให้อยู่ในระดับเกทหรือ Netlist แล้ว ข้อมูลนี้จะถูกใช้สำหรับจำลองการทำงานในเรื่อง ความถูกต้องของฟังก์ชัน พร้อมกับนำข้อมูลที่เกี่ยวข้องกับเวลาเข้ามาประกอบการพิจารณาด้วย ซึ่งตามปกติแล้วอุปกรณ์ ทางอิเล็กทรอนิกส์ทุกชิ้นจะมีเวลาหน่วงของการแพร่กระจาย (Propagation Delay Time) เสมอ ถึงแม้ว่าจะเป็น เวลาที่น้อยมากในระดับนาโนวินาทีก็ตาม แต่ถ้าภายในวงจรหนึ่งประกอบด้วยเกทของฟังก์ชันต่างๆ จำนวน 10,000 เกท ขึ้นไป เวลาดังกล่าวนี้จะสะสมกันมากขึ้น จนอาจทำให้การทำงานของวงจรรวมทั้งหมดผิดพลาดไป หรือไม่สามารถทำ งานในย่านความถี่สัญญาณนาฬิกาที่สูงได้

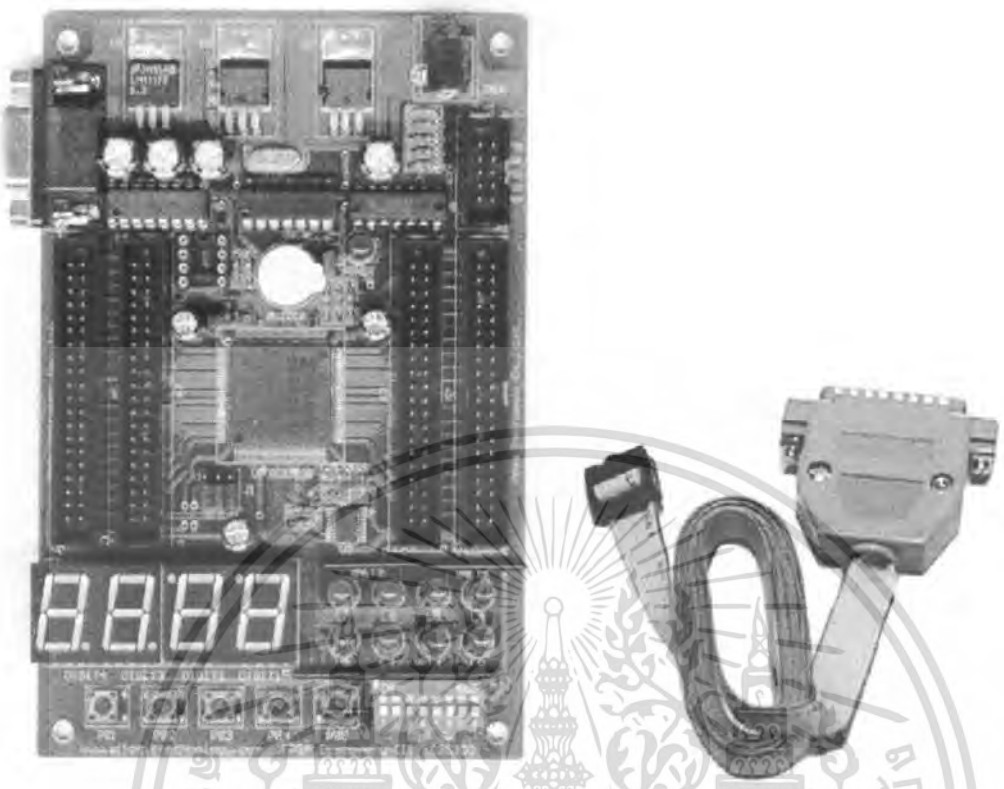
5.ผลิตเป็นวงจรจริง (Technology and device mapping) โดยนำข้อมูลที่ได้จากการสังเคราะห์มาผลิต ซึ่งอาจ จะอยู่ในรูปของแผงวงจรไฟฟ้า ที่ประกอบด้วยอุปกรณ์หลายๆ ชิ้นหรืออยู่ในรูปของวงจรรวม ASIC

6.ทำการตรวจสอบการทำงานและตัวแปรทางด้านเวลาทั้งหมด เพื่อความถูกต้องของวงจรเป็นครั้งสุดท้ายก่อนนำไปรวมเข้ากับอุปกรณ์อื่นๆ ให้เป็นระบบดิจิตอล เนื่องจากในขั้นตอนนี้ วงจรที่ออกแบบ จะประกอบด้วยจุดต่อทางอินพุทและเอาท์พุท ซึ่งเป็นจุดต่อสำหรับการรับและส่งสัญญาณกับภายนอก

7.นำวงจรที่ออกแบบไว้ประกอบเข้ากับอุปกรณ์อื่นๆ ให้เป็นระบบที่สมบูรณ์ แล้วทำการทดสอบการทำงานทั้งระบบร่วมกับอุปกรณ์อื่นๆ อีกครั้งเพื่อควบคุมคุณภาพของผลิตภัณฑ์

2.9.14 FPGA Discovery-III XC3S400

บอร์ดทดลองอเนกประสงค์รุ่น Discovery-III XC3S400 มีรายละเอียดดังแสดงในรูป โดยที่บอร์ดนี้จะเป็นบอร์ดพัฒนา FPGA ที่มีความจุวงจร 400,000 เกต และใช้ Platform Flash PROM สำหรับเก็บข้อมูลวงจร ซึ่งสามารถโปรแกรมวงจรลง Platform Flash PROM ผ่านทางสายดาวน์โหลดแบบ JTAG ได้โดยตรง บนบอร์ดมีอุปกรณ์อำนวยความสะดวกที่เพียบพร้อมด้วยอุปกรณ์อินพุทเอาต์พุตอย่างครบครัน จึงเหมาะสำหรับ LAB ออกแบบวงจรดิจิตอลและออกแบบไอซีขั้นสูง งานพัฒนาออกแบบวงจรขนาดใหญ่



รูปที่ 2.29 บอร์ดทดลองเอนกประสงค์ FPGA Discovery-III XC3S400

คุณสมบัติทั่วไป

บอร์ดทดลองเอนกประสงค์ FPGA Discovery-III XC3S200 มี 2 รุ่นด้วยกันคือ

- FPGA Discovery-III XC3S200F (รุ่น 200,000 เกต)
- FPGA Discovery-III XC3S200F4 (รุ่น 400,000 เกต)

บอร์ด FPGA Discovery-III XC3S200F และ FPGA Discovery-III XC3S200F4 จะมีลักษณะเหมือนกันทุกประการ แต่ที่แตกต่างกันคือ FPGA Discovery-III XC3S200F จะใช้ FPGA ตระกูล Spartan-3 ของ Xilinx เบอร์ XC3S200-4TQ144C ซึ่งเป็น FPGA ขนาดความจุวงจร 200,000 เกต , Package แบบ TQ144 , Speed Grade:4 และ Platform Flash PROM เบอร์ XCF01SV020C ในขณะที่ FPGA Discovery-III XC3S200F4 จะใช้ชิพเบอร์ XC3S400-4TQ144C ซึ่งเป็น FPGA ขนาด 400,000 เกต Package แบบ TQ144, Speed Grade:4 และ Platform Flash PROM เบอร์ XCF02SVO020C ที่สามารถโปรแกรมข้อมูลวงจรซ้ำได้ถึง 20,000 ครั้ง คุณสมบัติอื่นๆเป็นดังนี้

- 7-segment จำนวน 4 หลัก (ใช้ร่วมกับ Expansion ports และสามารถถอดออกได้)
- LED จำนวน 8 ดวง (ใช้ร่วมกับ Expansion ports สามารถแยกออกจาก I/O ได้โดยหักเอา RNET3 และ RNET4 ออก)
- Buzzer จำนวน 1 ตัว(ใช้ร่วมกับ Expansion ports)
- DIP Switch 8 บิต

- Push Botton Switch จำนวน 5 ตัว
- Expansion ports(80 Bits 3.3V. I/O)
- RS-232 Port 1 Port (ใช้ร่วมกับ Expansion ports)
- I^2C Socket สำหรับ EEPROM (ใช้ร่วมกับ Expansion ports)
- 25 MHz Oscillator (สามารถโปรแกรมเป็นความถี่อื่นๆได้โดยใช้ Digital Frequency Synthesizer มีที่อยู่ใน FPGA)

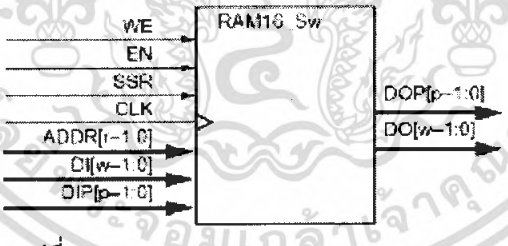
คุณสมบัติสำคัญของชิพ FPGA ตระกูล Spartan-3 เบอร์ XC3S400

- ความจุวงจร 400,000 เกต
- 18Kb block RAMs จำนวน 16 ชุด (รวม 288 Kb)
- 18x18 hardware multiplier จำนวน 16 ชุด
- Digital Clock Manager (DCM) จำนวน 4 ชุด

โดยอุปกรณ์ที่อยู่ภายในชิพมีคุณสมบัติดังนี้

หน่วยความจำ 18 Kb block RAM

18 Kb block RAM เป็นหน่วยความจำที่มีความเร็วสูงมาก (โดยประมาณ) 200 MHz ที่มีอยู่ในชิพ สามารถฟอร์มให้เป็น RAM หรือ ROM ที่มีขนาดต่างๆกันได้รวมทั้ง FIFO ด้วย โดยรูปด้านล่างนี้แสดงขนาด RAM แบบ Single Port และรูปถัดไปแสดงตัวอย่าง RAM แบบ Single Port ขนาดต่างๆ ที่สร้างจาก Block RAM แต่ละชุด



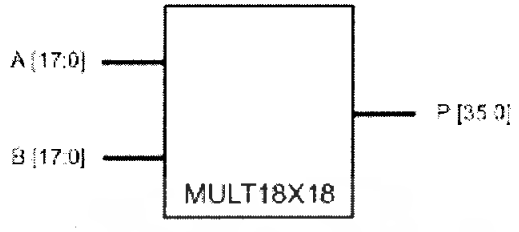
รูปที่ 2.30 แสดงขนาด RAM แบบ Single Port

Organization	Memory Depth	Data Width	Parity Width
512x36	512	32	4
1Kx18	1024	16	2
2Kx9	2048	8	1
4Kx4	4096	4	-
8Kx2	8192	2	-
16Kx1	16384	1	-

รูปที่ 2.31 RAM แบบ Single Port ขนาดต่างๆ ที่สร้างจาก Block RAM แต่ละชุด

18x18 Hardware multiplier

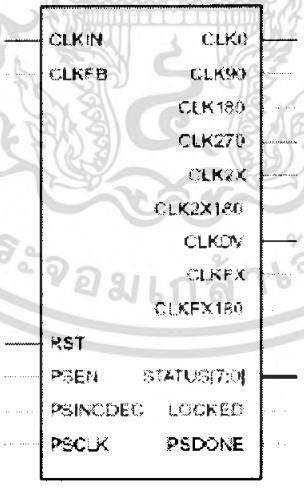
18 x 18 hardware multiplier เป็นฮาร์ดแวร์ของวงจรคูณสำเร็จรูปขนาด 18x18 บิตที่มีความเร็วในการทำงานสูง ซึ่งมีสัญลักษณ์ดังรูป สำหรับคนที่ต้องการตัวคูณมากๆ อาจจะต้องใช้วิธีมัลติเพล็กซ์เข้าช่วย



รูปที่ 2.32 สัญลักษณ์ของ 18x18 hardware multiplier

Digital Clock Manager

Digital Clock Manager(DCM) เป็นวงจรที่มีความสำคัญมากที่ช่วยจัดการเกี่ยวกับสัญญาณนาฬิกาและเลื่อนเฟสของสัญญาณนาฬิกา มีอยู่ในชิพจำนวน 4 ชุด จึงทำให้การออกแบบวงจรง่ายขึ้น เนื่องจากสามารถสร้างความถี่ต่างๆได้จาก ออสซิลเลเตอร์จากภายนอกเพียงชุดเดียว ไม่เพียงเท่านั้น สัญญาณนาฬิกาดังกล่าวยังซิงก์โครไนซ์กับสัญญาณของออสซิลเลเตอร์เดิมอีกด้วย DCM มีสัญลักษณ์และจะทำงานในหน้าที่ดังต่อไปนี้



รูปที่ 2.33 สัญลักษณ์ของวงจร DCM

- หารความถี่ (Clock Divider) เป็นวงจรหารซึ่งจะให้ความถี่เอาต์พุตเท่ากับความถี่อินพุตหารด้วยตัวเลขดังต่อไปนี้ คือ 1.5 , 2 , 2.5 , 3 , 3.5 , 4 , 4.5 , 5 , 5.5 , 6 , 6.5 , 7 , 7.5 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 หรือ 16 ตามลำดับ

- สร้างความถี่สองเท่า (Clock Doubler) เป็นวงจรซึ่งจะให้ความถี่ที่เอาต์พุตจะเป็น 2 เท่าของความถี่อินพุต

- Digital Frequency Synthesize (DFS) เป็นวงจรซึ่งสามารถกำหนดให้ความถี่เอาต์พุตเท่ากับผลคูณของความถี่อินพุตกับอัตราส่วนของ M/D โดยที่ M = 2 ถึง 32 และ D = 1 ถึง 32 วงจรนี้นำไปใช้งาน เช่น สร้างวงจรเปลี่ยนจากการส่งข้อมูลแบบขนานเป็นอนุกรม ซึ่งต้องสร้างสัญญาณนาฬิกาสูงกว่าของเดิม เช่น 10-11 เท่า เป็นต้น หรืองานอื่นๆ ที่ต้องใช้วงจรฟรีเควนซิซินธิไซเซอร์ เช่น ต้องการความถี่ 66.66666 MHz แต่ออสซิลเลเตอร์บนบอร์ดนี้ คือ 25 MHz เป็นสัญญาณอินพุตให้กับ DFS ดังนั้นต้องกำหนดที่ DFS ให้ M=8 และ D=3 แล้วจะได้เอาต์พุตต้องการคือ 66.66666 MHz

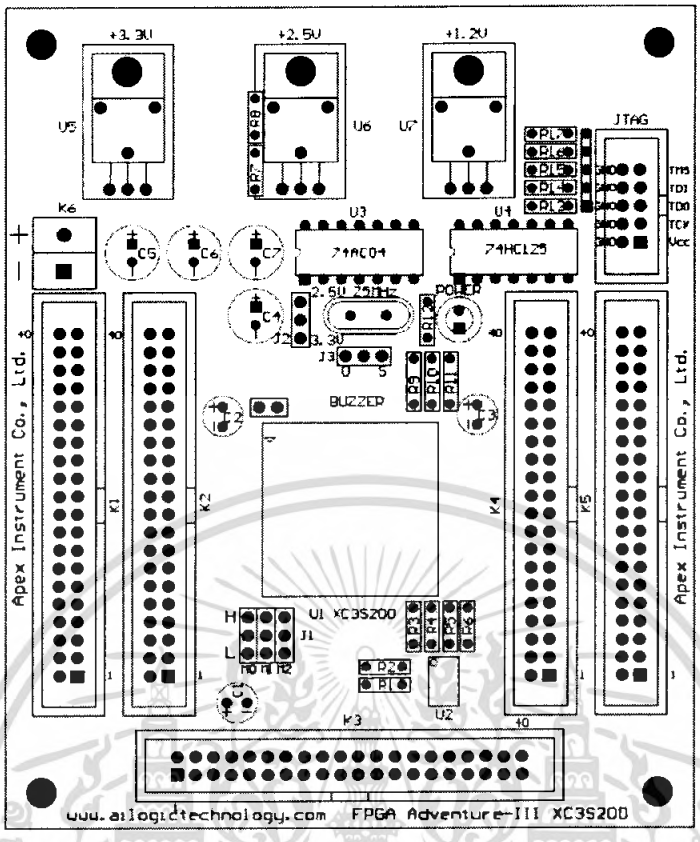
- Delay-Locked Loop (DLL) เป็นวงจรใช้แก้ปัญหาการเลื่อนเฟสในวงจรให้กลับมาตรงตามเฟสที่ต้องการ

- Quadrant Phase Shift เป็นวงจรเลื่อนเฟส 90 , 180 และ 270 องศา ตามลำดับ

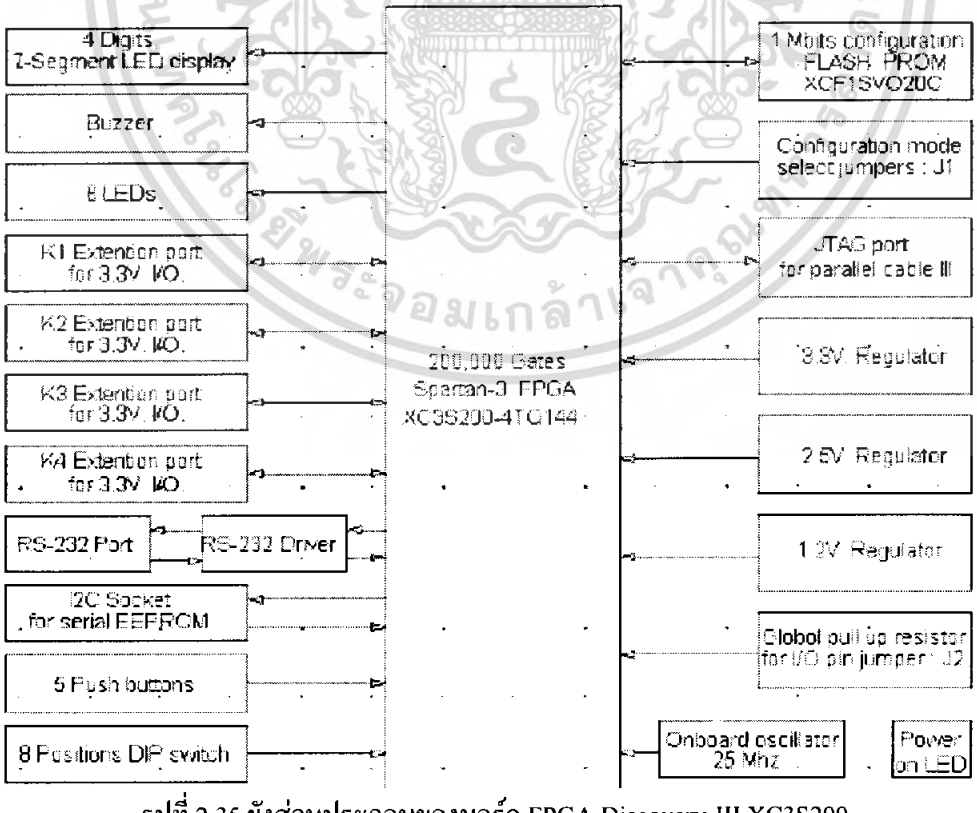
- Fine Phase Shift เป็นวงจรที่ใช้ในการเลื่อนเฟสอย่างละเอียด มีความละเอียดอยู่ที่ 1/255 เท่าของคาบความถี่ วงจรนี้มีความสำคัญมากเช่นกัน ที่ใช้ในการชดเชยการเลื่อนเฟสที่เกิดขึ้นในวงจร ทำให้การออกแบบง่ายขึ้นมาก

หลักการทำงานของบอร์ดเอนกประสงค์

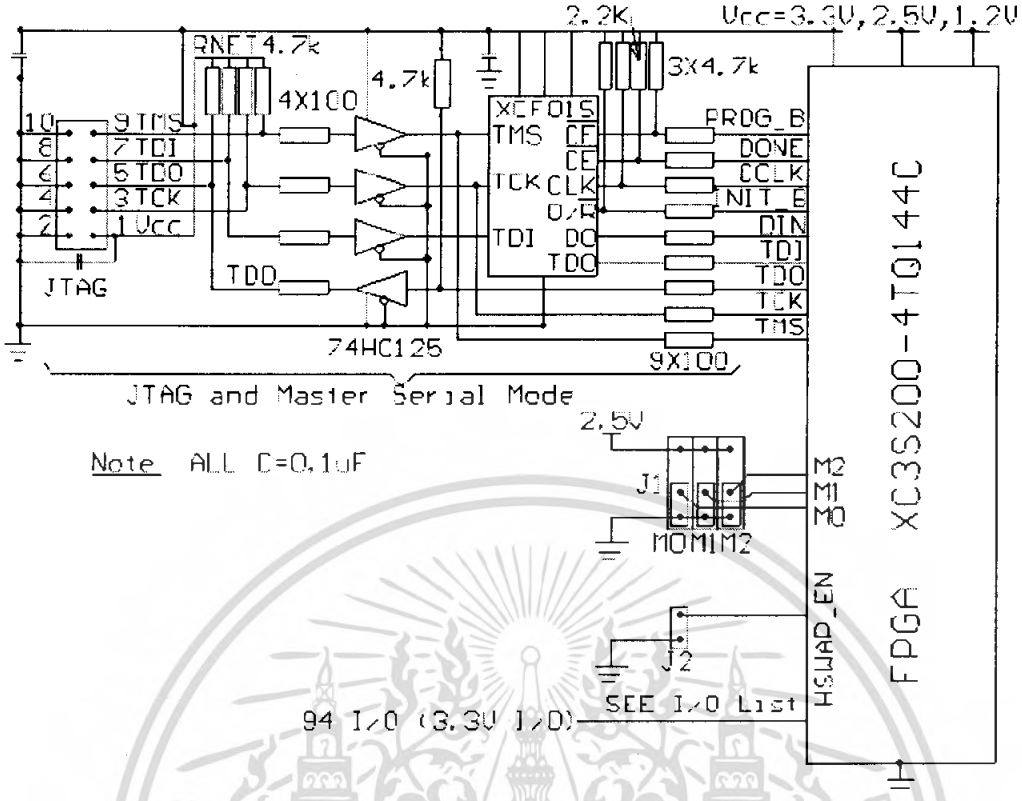
บอร์ดทดลองอเนกประสงค์รุ่น FPGA Discovery-III XC3S200 มีการจัดวางอุปกรณ์ดังรูปที่ 2.34 มีบล็อกไดอะแกรมดังรูปที่ 2.35 รายละเอียดการจัดวาง I/O (โดยย่อ) แสดงดังรูปที่ 2.36 และอุปกรณ์ที่ต่ออยู่กับขา FPGA (I/O List) ตามตารางที่ 2.1



รูปที่ 2.34 การจัดวางตำแหน่งการวางอุปกรณ์ด้านบน



รูปที่ 2.35 ผังส่วนประกอบของบอร์ด FPGA Discovery-III XC3S200



รูปที่ 2.36 การจัดวาง I/O ต่างๆของ FPGA Discovery-III XC3S200 (โดยย่อ)

ตารางที่ 2.1 รายละเอียดของอุปกรณ์ที่ต่ออยู่กับขา FPGA (I/O List) (K1- K5)

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

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

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

RS-232	FPGA Pinout	Descriptions
TX	p131	ICL3232CP
RX	p132	ICL3232CP

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

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

Oscillator	FPGA Pinout	Descriptions
OSC	p127	25MHz , GCLK6

BUZZER	FPGA Pinout	Descriptions
BUZZER	p125	BUZZER

ตารางที่ 2.1 (ต่อ)

K1 CONNECTOR					
Descriptions	FPGA Pinout	k1 Pinout	k1 Pinout	FPGA Pinout	Descriptions
GND	-	40	39	p128	I/O , I2C-SCL
GND	-	38	37	p130	I/O
GND	-	36	35	p132	I/O , RS-232(RX)
GND	-	34	33	p137	I/O
GND	-	32	31	p141	I/O
GND	-	30	29	p2	I/O
GND	-	28	27	p5	I/O
GND	-	26	25	p7	I/O
GND	-	24	23	p10	I/O
GND	-	22	21	p12	I/O
GND	-	20	19	p14	I/O
GND	-	18	17	p17	I/O
GND	-	16	15	p20	I/O , dp-7 Segment
GND	-	14	13	p23	I/O , g-7 Segment
GND	-	12	11	p25	I/O , f-7 Segment
GND	-	10	9	p27	I/O , e-7 Segment
GND	-	8	7	p30	I/O , d-7 Segment
GND	-	6	5	p32	I/O , c-7 Segment
GND	-	4	3	p35	I/O , b-7 Segment
+3.3V,Vcc	-	2	1	p40	I/O , a-7 Segment

K2 CONNECTOR					
Descriptions	FPGA Pinout	K2 Pinout	K2 Pinout	FPGA Pinout	Descriptions
GND	-	40	39	p129	I/O , I2C-SDA
GND	-	38	37	p131	I/O , RS-232(TX)
GND	-	36	35	p135	I/O
GND	-	34	33	p140	I/O
GND	-	32	31	p1	I/O
GND	-	30	29	p4	I/O
GND	-	28	27	p6	I/O
GND	-	26	25	p8	I/O
GND	-	24	23	p11	I/O
GND	-	22	21	p13	I/O
GND	-	20	19	p15	I/O
GND	-	18	17	p18	I/O
GND	-	16	15	p21	I/O
GND	-	14	13	p24	I/O
GND	-	12	11	p26	I/O
GND	-	10	9	p28	I/O
GND	-	8	7	p31	I/O , DIGIT1
GND	-	6	5	p33	I/O , DIGIT2
GND	-	4	3	p36	I/O , DIGIT3
+3.3V, Vcc	-	2	1	p41	I/O , DIGIT4

ตารางที่ 2.1 (ต่อ)

K3 CONNECTOR					
Descriptions	FPGA Pinout	K3 Pinout	K3 Pinout	FPGA Pinout	Descriptions
I O	p124	40	39	-	GND
I O	p122	38	37	-	GND
I O	p118	36	35	-	GND
I O	p113	34	33	-	GND
I O	p108	32	31	-	GND
I O	p105	30	29	-	GND
I O	p103	28	27	-	GND
I O	p100	26	25	-	GND
I O	p98	24	23	-	GND
I O	P96	22	21	-	GND
I O	p93	20	19	-	GND
I O	p90	18	17	-	GND
I O	p87	16	15	-	GND
I O	p85	14	13	-	GND
I O	p83	12	11	-	GND
I O	p80	10	9	-	GND
I O , L7	p78	8	7	-	GND
I O , L3	p76	6	5	-	GND
I O , L6	p73	4	3	-	GND
I O , L2	p69	2	1	-	+3.3 V, Vcc

ตารางที่ 2.1 (ต่อ)

K4 CONNECTOR					
Descriptions	FPGA Pinout	K4 Pinout	K4 Pinout	FPGA Pinout	Descriptions
I/O,BUZZER	p125	40	39	-	GND
I/O	p123	38	37	-	GND
I/O	p119	36	35	-	GND
I/O	p116	34	33	-	GND
I/O	p112	32	31	-	GND
I/O	p107	30	29	-	GND
I/O	p104	28	27	-	GND
I/O	p102	26	25	-	GND
I/O	p99	24	23	-	GND
I/O	p97	22	21	-	GND
I/O	p95	20	19	-	GND
I/O	p92	18	17	-	GND
I/O	p89	16	15	-	GND
I/O	p86	14	13	-	GND
I/O	p84	12	11	-	GND
I/O	p82	10	9	-	GND
I/O,L5	p79	8	7	-	GND
I/O,L1	p77	6	5	-	GND
I/O,L4	p74	4	3	-	GND
I/O,L0	p70	2	1	-	+3.3V, Vcc

ด้านเอาต์พุตบอร์ดทดลองนี้จะมี LED จำนวน 8 ดวงคือ LED0-LED7 โดยต่อขาคาโถด (Cathode) ลงกราวด์และต่อขา แอโนดเข้ากับ I/O ของชิพ FPGA มีตัวต้านทานแบบเนตเวิร์ค R=470 โอห์ม คือ RNET3 และ RNET4 ต่ออนุกรมอยู่เพื่อกำจัดกระแส ดังนั้นถ้า FPGA ส่งลอจิก '1' มาที่ขาใดจะทำให้ LED ติดสว่าง เนื่องจาก I/O ที่ใช้ขับ LED และบาง I/O ที่คอนเนคเตอร์ K3 และ K4 ต่อพาวงกันอยู่ ดังนั้นหากต้องการใช้ I/O ของ K3 และ K4 ตรงส่วนนี้หรือไม่ต้องการใช้ LED ก็สามารถหักตัวต้านทานแบบเนตเวิร์คออกได้ง่าย

2.10 การสื่อสารข้อมูลแบบอนุกรม

การสื่อสารข้อมูลแบบอนุกรมเป็นการรับ หรือ ส่งข้อมูลในลักษณะกลุ่มของบิตครั้งละ 1 บิต เรียงลำดับเรื่อยไปจนสิ้นสุดแต่ในบางกรณีก็สามารถรับส่งข้อมูลครั้งละหลายๆบิตได้หากแต่จะต้องมีการตกลงกันระหว่างตัวส่งกับตัวรับว่า จะรับส่งข้อมูลคราวละกี่บิต ตัวรับจะต้องรอข้อมูลมาให้ครบทุกบิตเสียก่อนจึงทำการประมวลผล ส่งผลให้การสื่อสารข้อมูลแบบอนุกรมอาจจะมีความเร็วในการรับส่งข้อมูลต่ำกว่าแบบขนาน ในด้านจำนวนสายสัญญาณการรับส่งข้อมูลแบบอนุกรมจะใช้จำนวนสายที่น้อยกว่ามาก

เองจากการสื่อสารข้อมูลแบบขนานมีการโอนย้ายมาพร้อมกันจึงมีความจำเป็นต้องใช้จำนวนสายสัญญาณมากขึ้นตามจำนวนบิตของข้อมูลด้วยในขณะที่การสื่อสารแบบอนุกรมนั้นต้องการสายสัญญาณเพียงสอง หรือ สามเส้นเท่านั้น แต่อัตราในการรับส่งข้อมูลอาจต่ำกว่าแบบขนาน ทำให้ระยะทางในการสื่อสารข้อมูลแบบอนุกรมสามารถทำได้มากกว่าการสื่อสารแบบขนาน ดังนั้นการสื่อสารแบบขนานจึงไม่เหมาะในการสื่อสารกับอุปกรณ์ภายนอกเป็นระยะทางไกล เพราะ จะทำให้สิ้นเปลืองค่าใช้จ่ายมาก

การสื่อสารข้อมูลแบบอนุกรมนั้นแบ่งออกเป็น 2 แบบคือ การสื่อสารข้อมูลแบบอนุกรมแบบซิงโครนัส และการสื่อสารข้อมูลแบบอะซิงโครนัส โดยการสื่อสารข้อมูลอนุกรมแบบซิงโครนัสนั้นจะมีสัญญาณนาฬิการวมอยู่กับการรับ และ ส่งด้วย ตัวอย่างการส่งข้อมูลอนุกรมแบบซิงโครนัสได้แก่ คีย์บอร์ดของคอมพิวเตอร์ ซึ่งสายเส้นหนึ่งจะเป็นของสัญญาณนาฬิกา ส่วนสายอีกเส้นหนึ่งจะเป็นของข้อมูล ดังนั้นการติดต่อกันแบบซิงโครนัสนี้จะต้องสายที่เชื่อมต่อกันอย่างน้อยที่สุด 3 เส้น คือ สัญญาณนาฬิกา ข้อมูล และ กราวด์ ส่วนการสื่อสารข้อมูลอนุกรมแบบซิงโครนัสนั้นสามารถรับ และ ส่งข้อมูลไป ในสายโดยไม่จำเป็นต้องมีสัญญาณนาฬิการวมอยู่ด้วยเหมือนกับการรับส่งข้อมูลแบบซิงโครนัส แต่จะใช้การกำหนดค่าสัญญาณนาฬิกาทั้งภาครับ และภาคส่งให้มีค่าเท่ากัน ซึ่งเรียกสัญญาณนาฬิกาที่ใช้ในการกำหนดค่าให้ภาครับ และ ภาคส่งนี้ว่า อัตราการถ่ายทอดข้อมูล หรือ บิตเรต (Baudrate) มีหน่วยเป็นบิตต่อวินาที (bit per second : bps)

2.10.1 ความเร็วของการสื่อสารข้อมูลแบบอนุกรม

เนื่องจากการสื่อสารข้อมูลแบบอนุกรมเป็นการรับส่งข้อมูลในลักษณะกลุ่ม หรือ บิตข้อมูล (Bit Stream) ดังนั้น จึงต้องให้ความสนใจในการพิจารณาเรื่องอัตราเร็ว ในการรับส่งบิตเหล่านี้เป็นอันดับแรก โดยทั่วไปมักจะระบุกันในหน่วยของจำนวนบิตข้อมูลภายในเวลาหนึ่งวินาที เรียกว่า อัตราบอด ตามค่ามาตรฐานเหล่านี้ ได้แก่ 110, 150, 300, 1200, 2400, 4800, 9600, 19200 บอด ข้อมูลทั้ง 8 บิตนี้หากว่าถูกส่งออกมาด้วยอัตรา 9600 บอด จะใช้เวลาในการส่งข้อมูลหนึ่งบิตมีค่าเท่ากับ 1/9600 หรือ 104 us และเวลาในการส่งข้อมูลทั้งแปดบิตมีค่าเท่ากับ 8*104 หรือ 832 us

2.10.2 รูปแบบของการส่งข้อมูลแบบอนุกรม

การสื่อสารข้อมูลอนุกรมแบบอะซิงโครนัสจะใช้การแปลงข้อมูลขนานให้เป็นอนุกรมแล้วเพิ่มเติมบิตบางอย่างร่วมไปกับการส่งข้อมูลจริงซึ่งรูปแบบของข้อมูลที่ใช้ในการรับส่งข้อมูลอนุกรมแบบอะซิงโครนัสประกอบด้วย 4 ส่วน ได้แก่

1. บิตเริ่มต้น (Start Bit) ซึ่งจะมีขนาด 1 บิต บิตเริ่มต้นมีหน้าที่สำหรับการบ่งบอกให้ทราบถึงตำแหน่งเริ่มต้นของบิตข้อมูล ตามปกติแล้วค่าของบิตเริ่มต้นจะเป็นระดับลอจิกต่ำ

2. บิตข้อมูลแบบอนุกรมจะมีขนาด 5, 6, 7 หรือ 8 บิต

3. บิตแสดงสภาวะความเป็นเลขคู่ หรือ เลขคี่ของข้อมูล บิตที่เป็น 1 (Parity Bit) จะมีขนาด 1 บิต หรือไม่มี บิตนี้มีหน้าที่เพื่อตรวจสอบความถูกต้องของข้อมูลโดยทั่วไปมักเรียกว่า บิตพาริตี้ และจะนำไปต่อท้ายบิตข้อมูล ค่าของบิตนี้ขึ้นอยู่กับจำนวนค่าของบิตที่เป็น 1 ซึ่งจะเป็นได้ 2 ลักษณะ คือ พาริตี้คู่ (Even

Parity) หรือ พาริตี้คี่ (Odd Parity) ตัวอย่างเช่น ระบบที่ติดต่อกันโดยระบุว่าจะใช้พาริตี้คู่ ทางด้านส่งจะนำค่าข้อมูลที่จะส่งมาพิจารณาหาจำนวนของบิตเป็น 1 หากเป็นเลขจำนวนคู่อยู่แล้ว ค่าของพาริตี้จะมีค่าเป็น 0 แต่ถ้าจำนวนของบิตที่เป็น 1 เป็นเลขจำนวนคี่ ค่าของพาริตี้ก็จะมีค่าเป็นหนึ่ง ส่วนทางด้านรับก็จะทำการตรวจสอบจำนวนบิตที่เป็น 1 ของข้อมูลที่ได้รับมาทั้งหมดรวมทั้งบิตพาริตี้ ถ้ามีค่าเป็นเลขจำนวนคู่ แสดงว่าข้อมูลที่ได้รับเข้ามานี้ถูกต้อง แต่หากไม่เป็นเลขจำนวนคู่แสดงว่าเกิดการผิดพลาดของข้อมูลขึ้น เป็นต้น

4. บิตสุดท้าย (Stop Bit) จะมีขนาด 1, 1.5 หรือ 2 บิต บิตสุดท้ายเป็นบิตที่เพิ่มเข้ามาเพื่อระบุของขอบเขตการสิ้นสุดของกลุ่มบิตข้อมูล บิตสุท้ายสามารถโปรแกรมได้คือ 1, 1.5 หรือ 2 บิต ดังนั้นกรณีของการส่งข้อมูล 8 บิต หากข้อมูลถูกส่งออกไปด้วยอัตราเร็ว 9600 บอด เวลาโดยรวมในการส่งข้อมูล 1 ไบต์ จะมีค่าเป็น 12*104 หรือ 1.25 ms

2.10.3 มาตรฐานพอร์ตอนุกรมแบบ RS-232

มาตรฐานการเชื่อมต่อแบบอนุกรม RS-232 เป็นมาตรฐานอุตสาหกรรมที่ออกแบบมาเพื่อใช้กับการรับส่งข้อมูลอนุกรมแบบอะซิงโครนัส 2 ทิศทาง โดยมาตรฐาน ในอดีตนั้นถูกออกแบบมาเพื่อการส่งผ่านข้อมูลจากคอมพิวเตอร์ไปยังโมเด็มเพียงอย่างเดียวเท่านั้นเพื่อที่จะนำข้อมูลจากโมเด็มนี้สื่อสารผ่านสายโทรศัพท์ไปยังคอมพิวเตอร์อีกชุดหนึ่งที่อยู่ห่างไกลกัน โดยคณะกรรมวางมาตรฐานที่มีชื่อว่า EIA RS-232 มาตรฐานนี้ในช่วงแรกจะใช้คอนเน็กเตอร์ (Connector) เป็นแบบ DB-25 โดยกำหนดความยาวสูงสุดของสายสัญญาณไว้ที่ 50 ฟุต มีระดับสัญญาณตั้งแต่ -3 V ถึง -12 V แสดงว่ามีข้อมูล (MASK) และ +3 V ถึง +12 V แสดงเป็นช่องว่าง (Space)

มาตรฐาน RS-232 ได้กำหนดรูปแบบของอุปกรณ์ที่เชื่อมต่อข้อมูล (Data Terminal Equipment : DTE) กับวงจรข้อมูลปลายทาง (Data Circuit Terminating : DCE) ไว้ว่า อุปกรณ์ DTE จะต้องเป็นอุปกรณ์ที่มีการประมวลผลในตัวเช่น ไมโครคอนโทรลเลอร์หรือ ไมโครคอมพิวเตอร์ ซึ่งมีความสามารถในการสร้างข้อมูลแบบอนุกรมได้ ส่วนอุปกรณ์ DCE จะทำหน้าที่เป็นเพียงตัวรับข้อมูลที่ส่งมาจาก DTE เท่านั้น โดยการรับส่งข้อมูลระหว่างอุปกรณ์ทั้งสองจะกระทำผ่านมาตรฐาน RS-232

ข้อแตกต่างระหว่างอุปกรณ์ DTE และ DCE อย่างหนึ่งที่เห็นได้ชัดคือ คอนเน็กเตอร์ของ DTE เป็นตัวผู้ ส่วนคอนเน็กเตอร์ DCE เป็นตัวเมีย ซึ่งพอร์ตอนุกรมของคอมพิวเตอร์ที่ใช้กันอยู่ทั่วไปจะเป็นคอนเน็กเตอร์ตัวผู้ ส่วนคอนเน็กเตอร์ที่อยู่ในโมเด็มจะเป็นตัวเมีย

- คอนเน็กเตอร์สำหรับพอร์ต RS-232 และการเชื่อมต่อ

มาตรฐานการเชื่อมต่อแบบอนุกรม RS-232 จะใช้คอนเน็กเตอร์แบบ DB-25 ตัวผู้หรือแบบ DB-9 ตัวผู้ ซึ่งคอนเน็กเตอร์แบบ DB-25 จะมีขาต่อใช้งานเพียง 9 เส้นเช่นเดียวกับคอนเน็กเตอร์แบบ DB-9 เนื่องจากขาอื่นที่เคยใช้งานในอดีต ในปัจจุบันมีการใช้งานไม่มากนัก จึงถูกยกเลิกไป

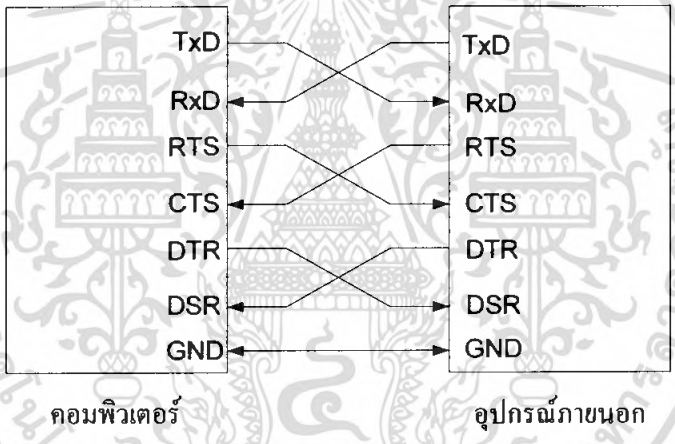
ตารางที่ 2.2 การจัดขาของคอนเน็กเตอร์พอร์ตอนุกรมตามมาตรฐาน RS-232 ทั้งแบบ DB-25 และ DB-9

คอนเน็กเตอร์ DB-9	คอนเน็กเตอร์ DB-25	ชื่อของสายสัญญาณ	ชนิดของสายสัญญาณ
1	8	Data Carrier Detect :DCD	อินพุต
2	3	Receive Data :RD	อินพุต
3	2	Transmitted Data : TD	เอาท์พุต
4	20	Data terminal Ready: DTR	เอาท์พุต
5	7	Signal Ground : GND	-
6	6	Data Set Ready :DSR	อินพุต
7	4	Request to send :RTS	เอาท์พุต
8	5	Clear to send :CTS	อินพุต
9	22	Ring Indicator : RI	อินพุต

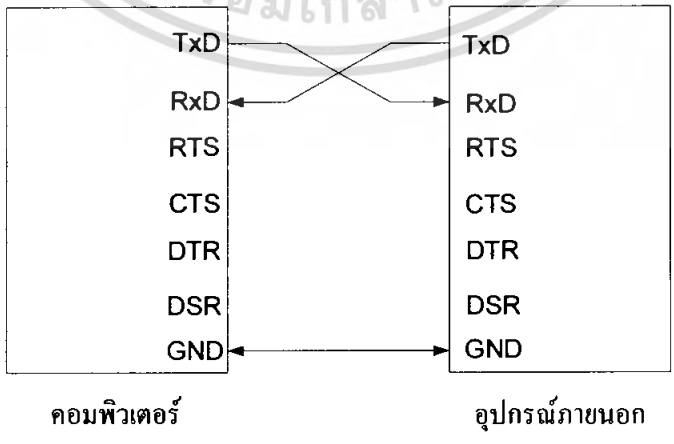
สำหรับการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอกนั้นแสดงดังในรูป 2.37 ลูกศรในรูปแสดงถึงทิศทางของข้อมูล ในรูปเป็นการเชื่อมต่อแบบ Null modem หรือการเชื่อมต่อโดยตรงโดยไม่ต้องผ่านโมเด็ม โดยมีการตรวจสอบหรือแฮนด์เช็กเต็มรูปแบบ ส่วนในรูปที่ 2.38 เป็นการเชื่อมต่อในลักษณะที่ใช้สายสัญญาณเพียง 3 เส้น โดยเส้นหนึ่งสำหรับส่งข้อมูล อีกเส้นหนึ่งสำหรับรับข้อมูลและเส้นสุดท้ายเป็นกราวด์ สำหรับหน้าที่ในการทำงานแต่ละขาของพอร์ตอนุกรม RS-232 มีดังนี้

- Data Carrier Detect : DCD หรืออาจเรียกว่า "Carrier Detect : CD" ขานี้จะแอกทีฟเมื่อมีการส่งสัญญาณพาหะจากอุปกรณ์สื่อสารข้อมูลเช่น โมเด็ม สำหรับการใช้งานปรกติ ขานี้จะไม่ถูกใช้งานมากนัก
- Receive Data : RD ขานี้ใช้เพื่อรับข้อมูลอนุกรมที่เข้ามายังคอมพิวเตอร์ โดยนำข้อมูลที่อ่านได้เก็บไว้ในรีจิสเตอร์บัฟเฟอร์
- Transmitted Data หรือ TD ใช้ส่งข้อมูลออกจากคอมพิวเตอร์ โดยนำข้อมูลที่เก็บอยู่ในบัฟเฟอร์สำหรับส่งข้อมูล (Transmitted Buffer) ส่งออกไป
- Data terminal Ready : DTR เป็นขาที่ส่งสัญญาณออกจากคอมพิวเตอร์ให้อุปกรณ์ปลายทางรับรู้ว่าต้องการติดต่อด้วยโดยขา DTR นี้ต้องเชื่อมกับ DSR ของอุปกรณ์ปลายทางและขา DTR ของอุปกรณ์ปลายทางต้องเชื่อมกับขา DSR ของคอมพิวเตอร์ถ้าใช้การเชื่อมต่อเป็นแบบ Null Modem ซึ่งใช้สายในการเชื่อมต่อเพียงสามเส้นจะต้องต่อขา DTR และ DSR ของตัวมันเองเข้าด้วยกัน
- Signal Ground : GND กราวด์ของระบบ

- Data Set Ready : DSR ขานี้จะใช้คู่กับขา DTR เพื่อตรวจสอบการเชื่อมต่อกันระหว่างคอมพิวเตอร์กับอุปกรณ์ปลายทางซึ่งขา DSR นี้จะเป็นขาสำหรับรับข้อมูลจากภายนอกซึ่งถูกส่งมาจากขา DTR
- Request to send : RTS เป็นขาสำหรับส่งสัญญาณร้องขอให้ทางอุปกรณ์ปลายทางส่งข้อมูลกลับมายังคอมพิวเตอร์ โดยขาที่รับสัญญาณ RTS ก็คือขา CTS ในกรณีที่ใช้การเชื่อมต่อแบบ Null Modem จะต้องเชื่อมต่อขา RTS และ CTS ของตัวมันเองเข้าด้วยกัน เพื่อที่จะให้การรับและส่งข้อมูลเกิดขึ้นได้ตลอดเวลา
- Clear to send : CTS ขานี้จะคอยรับสัญญาณจากขา RTS เมื่อรับสัญญาณได้ ข้อมูลที่ขา TD จะถูกส่งออกไป ดังนั้นขานี้จึงถูกใช้เพื่อตรวจสอบการเชื่อมต่อว่าพร้อมที่จะรับข้อมูลหรือไม่
- Ring Indicator : RI ใช้ในการแสดงสถานะสัญญาณเรียกจากสัญญาณโทรศัพท์ ปกติในการสื่อสารโดยทั่วไปสายนี้จะไม่ถูกใช้งาน จะใช้งานก็ต่อเมื่อมีการเชื่อมต่อกับโมเด็มและโปรแกรมมีการตรวจสอบสัญญาณนี้เท่านั้น



รูปที่ 2.37 การต่ออุปกรณ์ภายนอกเข้ากับคอมพิวเตอร์แบบ Null Modem



รูปที่ 2.38 การต่ออุปกรณ์ภายนอกเข้ากับคอมพิวเตอร์แบบ RS-232 โดยใช้สัญญาณเพียง3 เส้น

- UART (Universal Asynchronous Receiver Transmitter)

UART ย่อมาจากคำว่า "Universal Asynchronous Receiver Transmitter" หมายถึงอุปกรณ์ที่ทำหน้าที่รับและส่งข้อมูลอนุกรมแบบอะซิงโครนัสนั้นเอง ถือว่าเป็นหัวใจสำคัญของการสื่อสารแบบอนุกรม

หน้าที่หลักของ UART คือทำหน้าที่แปลงข้อมูลที่อยู่ในรูปแบบขนานจากคอมพิวเตอร์ให้อยู่ในรูปแบบอนุกรมซิงโครนัสแล้วส่งออกไป และทำหน้าที่แปลงข้อมูลอนุกรมแบบอะซิงโครนัสที่ป้อนเข้ามายัง UART ให้เป็นแบบขนานก่อนที่จะส่งเข้าสู่คอมพิวเตอร์ ซึ่งนอกจาก UART จะส่งข้อมูลไปยังคอมพิวเตอร์แล้ว ยังแจ้งข้อมูลอื่นๆ ให้คอมพิวเตอร์รับทราบด้วยเช่น อัตราเร็วในการรับส่งข้อมูล, รูปแบบการส่งข้อมูล, ความผิดพลาดที่เกิดขึ้นระหว่างการถ่ายทอดข้อมูล เป็นต้น ภายในจะมีส่วนของวงจรสร้างบอดเรตแบบโปรแกรมได้ (programmable buadrate generator) โดยการกำหนดค่าตัวหารให้กับสัญญาณนาฬิกาของ UART โดยตัวหารนี้มีขนาด 16 บิต ดังนั้นจึงสามารถกำหนดตัวหารอยู่ในช่วง 1 ถึง 65,535 สามารถรับส่งข้อมูลได้ทั้งแบบฮาล์ฟดูเพล็กซ์ (Half duplex) และฟูลดูเพล็กซ์ โดยการส่งแบบฮาล์ฟดูเพล็กซ์เป็นการส่งแบบทิศทางเดียวส่วนการส่งแบบฟูลดูเพล็กซ์นั้นสามารถรับและส่งข้อมูลได้ในคราวเดียวกัน

- ระดับแรงดันที่ใช้งานสำหรับพอร์ตอนุกรม RS-232

มาตรฐานการสื่อสารข้อมูลของพอร์ตอนุกรม ได้ระบุช่วงระดับแรงดันสำหรับการทำงานของพอร์ตอนุกรมไว้ว่า ที่ลอจิก " 0 " จะมีระดับสัญญาณ +3 V ถึง +15 V ส่วนลอจิก " 1 " จะมีระดับสัญญาณ -3 V ถึง -15 V ด้วยเหตุนี้จึงทำให้ไม่สามารถที่จะนำขาเอาท์พุตใดๆต่อเข้ากับลอจิกเกตเพื่อใช้งานได้โดยตรง จะต้องผ่านวงจรเพื่อที่จะเปลี่ยนระดับแรงดันเสียก่อน โดยปกติจะใช้ไอซีพวก RS-232 transceiver ที่นิยมมากคือ MAX 232 หรือ ICL232 ไอซีกลุ่มนี้จะทำหน้าที่แปลงระดับแรงดันของ RS-232 ให้อยู่ในระดับทีทีแอล (TTL) โดยลอจิก " 0 " ซึ่งเดิมมีระดับสัญญาณ +3 V ถึง +15 V จะถูกแปลงเป็น 0 V ส่วนลอจิก " 1 " ซึ่งเดิมมีระดับสัญญาณ -3 V ถึง -15 V จะถูกแปลงเป็น 5 V ทั้งนี้เพื่อให้สามารถเชื่อมต่อกับอุปกรณ์ดิจิตอลอื่นๆที่ใช้ระดับแรงดันทีทีแอลได้

2.11 หน่วยความจำ

จากการที่ได้ศึกษามาทั้งหมดจะพบว่าในระบบคอมพิวเตอร์จะมีการประมวลผลเป็นเลขฐานสองและการเก็บค่าเลขฐานสองไว้ในระบบอาจทำได้หลายวิธี เช่น ใช้ฟลิปฟลอปในการเก็บข้อมูลแต่ละบิต หรือเก็บข้อมูลหลายๆบิต โดยนำฟลิปฟลอปมาต่อเป็นรีจิสเตอร์ ถ้าหากนำฟลิปฟลอปมาต่อกัน 8 ตัวจะกลายเป็นอุปกรณ์เก็บข้อมูลขนาด 1 ไบต์ได้ สำหรับอุปกรณ์เก็บข้อมูลในระบบดิจิตอลหรือระบบคอมพิวเตอร์จะเรียกว่าหน่วยความจำ (memory) ซึ่งสามารถเก็บได้หลายตำแหน่งหรือหลายไบต์ และสามารถทำการอ่าน/เขียนข้อมูลได้ การที่มันเก็บข้อมูลได้จำนวนมากนั้นจะต้องมีการระบุว่าจะอ่านหรือเขียนข้อมูลที่ตำแหน่งใด โดยค่าตำแหน่งในหน่วยความจำนี้เรียกว่า แอดเดรส (address)

ระบบคอมพิวเตอร์จะแบ่งหน่วยความจำออกได้เป็นสองประเภทคือ หน่วยความจำปฐมภูมิ (primary memory) หรือ หน่วยความจำหลัก ซึ่งเป็นหน่วยความจำที่หน่วยประมวลผลหรือ CPU สามารถ

อ่าน/เขียนข้อมูลได้โดยตรง ซึ่งจะสร้างจากอุปกรณ์ประเภทสารกึ่งตัวนำ มีทั้งแบบที่อ่านข้อมูลได้อย่างเดียว (read only) และแบบอ่าน/เขียนข้อมูลได้ หน่วยความจำอีกประเภทหนึ่งจะใช้เก็บสำรองข้อมูล เรียกว่า หน่วยความจำสำรอง หรือหน่วยความจำทุติยภูมิ (secondary memory) ได้แก่ แผ่นดิสก์ ฮาร์ดดิสก์ ซีดีรอม หน่วยความจำประเภทนี้ CPU ไม่สามารถอ่านข้อมูลได้โดยตรง จะต้องมีวงจรอิเล็กทรอนิกส์ภายนอกช่วย

2.11.1 พื้นฐานการเก็บข้อมูล

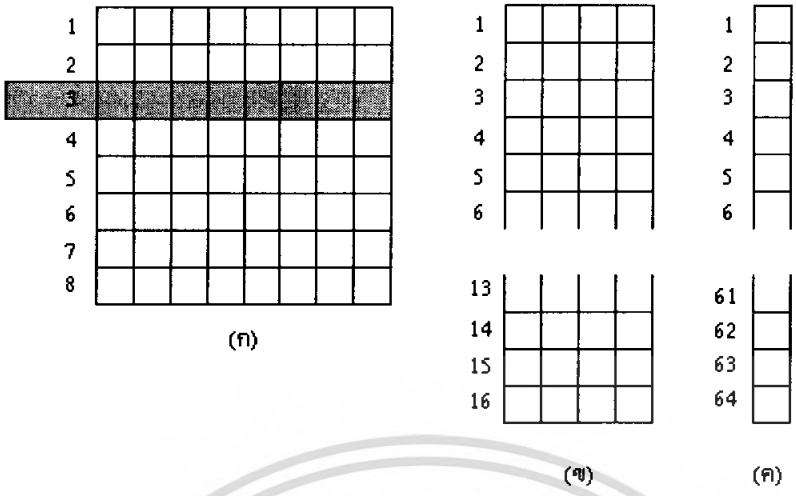
ถ้าหากมีหน่วยความจำที่เก็บข้อมูลได้ 16 ตำแหน่ง และแต่ละตำแหน่งเก็บข้อมูลเลขฐานสองได้ 8 บิตหรือ 8 เซลล์ การอ้างตำแหน่งหน่วยความจำหรือการอ้างแอดเดรสสามารถใช้เลขฐานสอง 4 บิตแทนได้ ดังนั้นตำแหน่งหน่วยความจำ (memory address) จะมีค่าตั้งแต่ 0000 ถึง 1111 ส่วนกลุ่มของข้อมูล 8 บิตที่เก็บในหน่วยความจำจะเรียกว่าไบต์ ดังนั้นหน่วยความจำประเภทนี้จะมีขนาด 16 ไบต์ ดังแสดงในรูปที่ 2.39

แอดเดรส 4 บิต	ข้อมูล 8 บิต
0000	xxxxxxxx
0001	xxxxxxxx
0010	xxxxxxxx
0011	xxxxxxxx
.....	.....
.....	.....
1110	xxxxxxxx
1111	xxxxxxxx

ข้อมูล 16 ไบต์

รูปที่ 2.39 หน่วยความจำขนาด 8 บิต 16 ตำแหน่ง

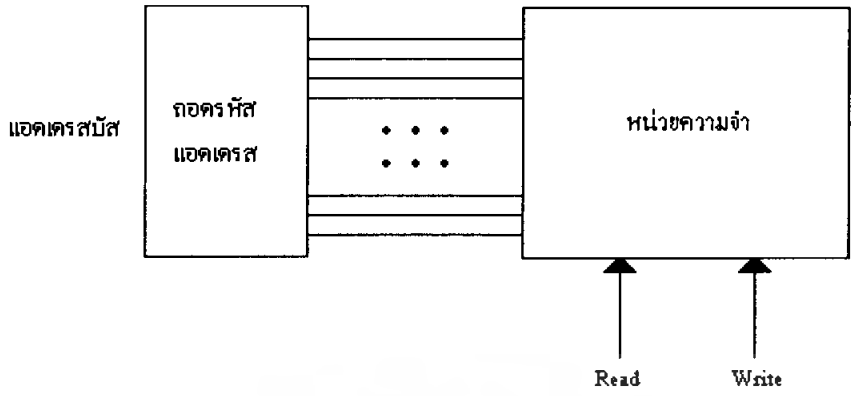
ในรูปที่ 2.40 แสดงโครงสร้างของหน่วยความจำที่เก็บข้อมูลได้ 64 เซลล์ โดยรูปที่ 2.40 (ก) จะเป็นโครงสร้างแบบอาร์เรย์ 8x8 โดยแต่ละตำแหน่งจะเก็บข้อมูลได้ 8 บิต ทำให้เก็บข้อมูลทั้งหมด 64 บิต หรือ 8 ไบต์ สำหรับรูปที่2.40(ข) จะเป็นโครงสร้างแบบอาร์เรย์ 16x4 แต่ละตำแหน่งจะเก็บข้อมูลได้ 1 นิบเบิลหรือ 4 บิต บางครั้งจะเรียกว่าหน่วยความจำขนาด 16 นิบเบิลส่วนรูปที่2.40(ค) เป็นโครงสร้างแบบอาร์เรย์ 64x1 ซึ่งแต่ละตำแหน่งเก็บข้อมูลได้ 1 บิต ในการบอกขนาดของหน่วยความจำมักจะบอกเป็นเวิร์ดข้อมูล เช่น หน่วยความจำแบบ 16kx4 จะเรียกว่าเก็บข้อมูล 16,384 เวิร์ดโดยแต่ละเวิร์ดมีขนาด 4 บิตเป็นต้น



รูปที่ 2.40 โครงสร้างหน่วยความจำขนาด 64 เซลล์แบบต่างๆ

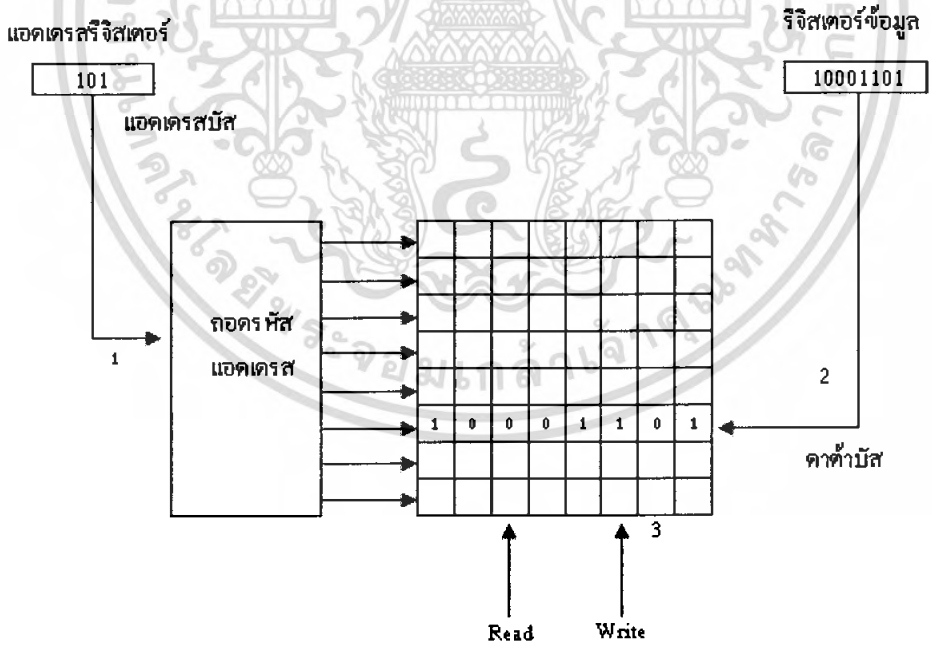
ในการอ่าน/เขียนข้อมูลกับหน่วยความจำ จะต้องมีสัญญาณระบุด้วยว่าต้องการอ่านหรือเขียนข้อมูล สัญญาณนี้เรียกว่าสัญญาณควบคุม ดังนั้นในระบบหน่วยความจำจะมีกลุ่มสัญญาณแอดเดรสซึ่งใช้ในการอ้างตำแหน่งหน่วยความจำ กลุ่มสัญญาณข้อมูลใช้สำหรับเป็นที่เข้า-ออกของข้อมูลกับหน่วยความจำ และกลุ่มสัญญาณควบคุม ใช้เป็นสัญญาณในการอ่าน/เขียนหน่วยความจำ จากหน่วยความจำตัวอย่างในรูปที่ 2.39 จะเห็นว่ากลุ่มสัญญาณแอดเดรสจะประกอบด้วยสายสัญญาณ 4 เส้น ในระบบคอมพิวเตอร์จะเรียกกลุ่มสัญญาณหลายๆเส้นว่า บัส (bus) ดังนั้นเราจะพบว่าในหน่วยความจำจะประกอบด้วยบัสข้อมูลหรือดาต้าบัส (data bus) แอดเดรสบัส (address bus) และบัสควบคุม (control bus) สำหรับในรูปที่2.40 ถ้าเป็นการอ่านข้อมูลตำแหน่งที่ 3 จะส่งค่าแอดเดรสเป็น 3 และอ่านข้อมูลหนึ่งไบต์ออกมา

ไดอะแกรมโครงสร้างของหน่วยความจำสามารถแสดงได้ดังรูปที่ 2.41 ซึ่งจะประกอบด้วยแอดเดรสบัส ดาต้าบัส และบัสควบคุม โดยแอดเดรสบัสจะเป็นบัสแบบทิศทางเดียว (directional bus) ซึ่งจะเข้าสู่หน่วยความจำทิศทางเดียว โดยบัสนี้จะต่อกับวงจรถอดรหัสแอดเดรสเพื่อใช้ในการอ้างตำแหน่งของหน่วยความจำว่าต้องการติดต่อกับหน่วยความจำตำแหน่งใด ส่วนดาต้าบัสจะเป็นบัสแบบสองทิศทาง (bidirectional bus) เนื่องจากข้อมูลสามารถเข้า-ออกหน่วยความจำได้ ส่วนบัสควบคุมจะประกอบด้วยสัญญาณ Read และ Write สำหรับอ่านและเขียนข้อมูลตามลำดับ



รูปที่ 2.41 ไดอะแกรมโครงสร้างหน่วยความจำ

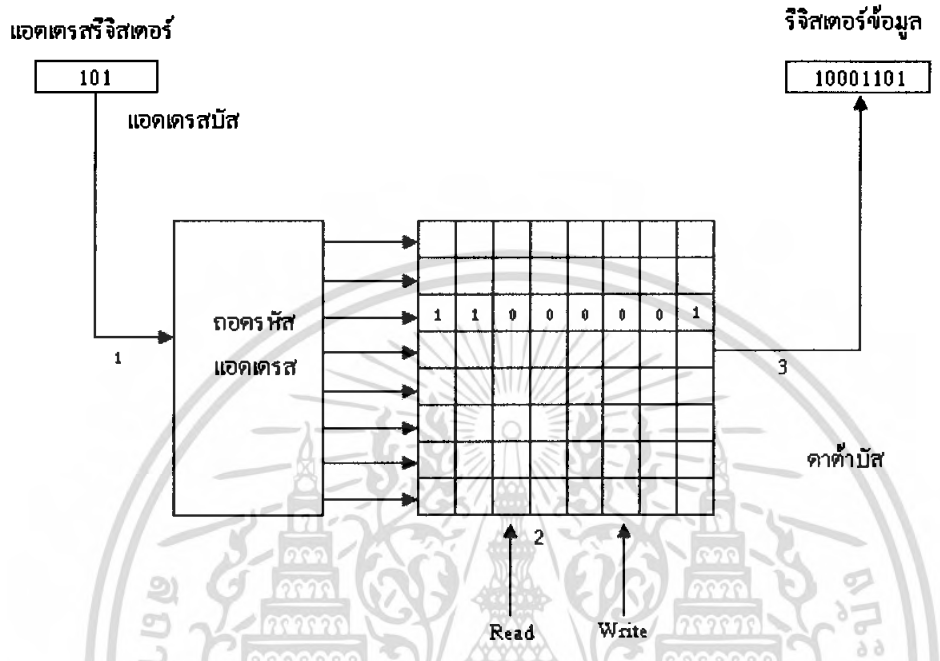
ขั้นตอนการทำงานของการเขียนข้อมูลลงหน่วยความจำแสดงได้ดังรูปที่ 2.42 ซึ่งเป็นการเขียนข้อมูลขนาด 1 ไบต์ลงหน่วยความจำ เริ่มแรกจะมีการส่งค่าแอดเดรสเข้าไปก่อนเพื่อบอกว่าต้องการเขียนข้อมูลลงหน่วยความจำตำแหน่งใด ตามรูปเป็นการอ้างแอดเดรสตำแหน่งที่ 5 (101) ต่อมาจะส่งข้อมูลขนาด 1 ไบต์เข้าทางดาต้าบัส แสดงส่งสัญญาณควบคุม Write เพื่อเขียนข้อมูลก็จะทำให้ข้อมูลถูกเก็บลงหน่วยความจำทันที



รูปที่ 2.42 ขั้นตอนการแสดงการเขียนข้อมูลลงหน่วยความจำโดยหมายเลขแสดงลำดับการทำงาน

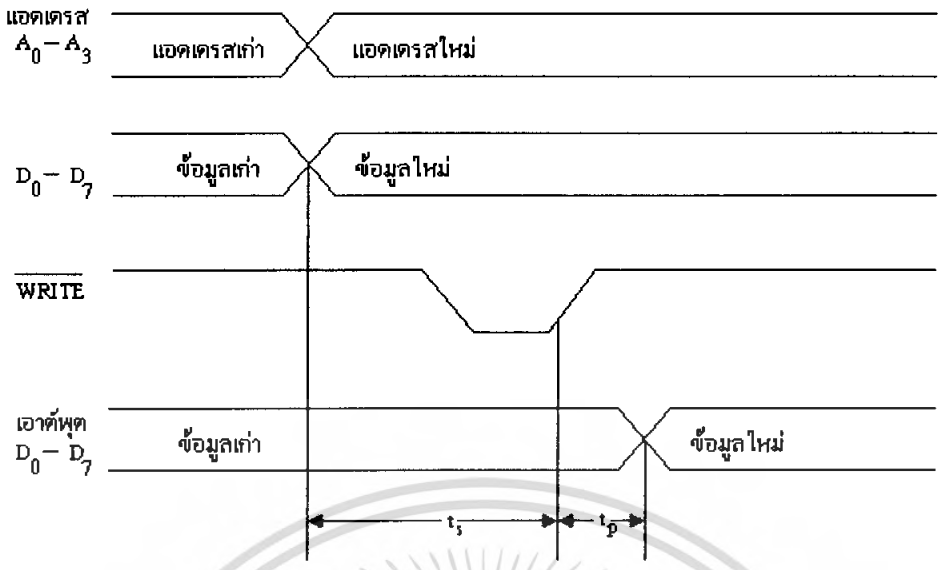
สำหรับกระบวนการอ่านข้อมูลสามารถแสดงได้ดังรูปที่ 2.43 โดยเริ่มแรกจะมีการส่งหมายเลขตำแหน่งที่ต้องการอ่านข้อมูลจากรีจิสเตอร์แอดเดรสเข้าไปทางแอดเดรสบัส โดยในรูปจะเป็นการอ่าน

ตำแหน่งที่ 3 (011) จากนั้นจะส่งสัญญาณไปบอกหน่วยความจำว่าต้องการอ่านข้อมูลจากนั้นข้อมูลจะถูกอ่านออกมาเก็บในรีจิสเตอร์เก็บข้อมูลทางคาต้าบัส โดยการอ่านข้อมูลนี้ ข้อมูลเดิมจะยังคงอยู่ในหน่วยความจำด้วย ไม่มีการเปลี่ยนแปลง



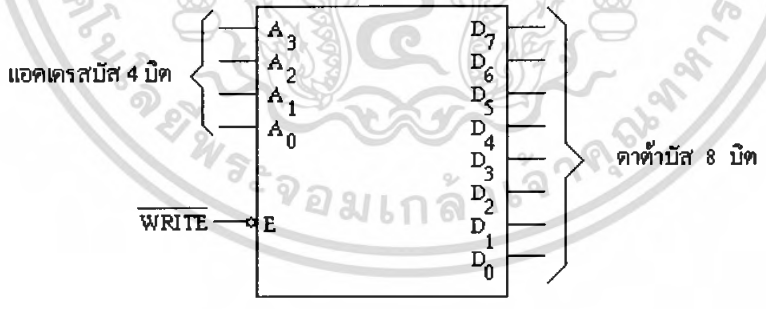
รูปที่ 2.43 การแสดงการอ่านข้อมูลลงหน่วยความจำโดยหมายเลขแสดงลำดับการทำงาน

จะเห็นว่าการอ่าน/เขียนหน่วยความจำจะมีลำดับการทำงานที่แน่นอน สำหรับอุปกรณ์ที่เป็นหน่วยความจำจะบอกลำดับการอ่าน/เขียนข้อมูลนี้เป็นไดอะแกรมเวลา ดังแสดงในรูปที่ 2.44 โดยเส้นที่ไขว้กันจะเป็นการบอกว่าเริ่มต้นการเปลี่ยนแปลง จากรูปจะเป็นไดอะแกรมเวลาของการเขียนข้อมูลเริ่มต้นจะส่งค่าแอดเดรสออกมาก่อน และส่งข้อมูลในเวลาต่อมา จากนั้นส่งสัญญาณ Write สำหรับเขียนข้อมูลซึ่งจะแอกทีฟ LOW ทำให้มีข้อมูลใหม่ในหน่วยความจำในเวลาต่อมา นอกจากนี้เราสามารถสร้างระบบหน่วยความจำจากไอซีต่างๆ ได้ ดังแสดงในรูปที่ 2.45



รูปที่ 2.44 ไดอะแกรมเวลาการอ่านข้อมูล

ระบบหน่วยความจำที่ได้ออกแบบขึ้นในรูปที่ 2.45 นี้จะเป็นหน่วยความจำขนาด 16 ไบต์(16X8) จะเห็นว่าโครงสร้างของหน่วยความจำจะซับซ้อน แต่ได้มีการออกแบบเป็นไอซีหน่วยความจำขึ้นทำให้สามารถนำมาใช้งานได้โดยง่าย ระบบหน่วยความจำที่ผ่านมานี้สามารถเขียนเป็นไดอะแกรมได้ ดังรูปที่ 2.46



รูปที่ 2.45 ไดอะแกรมของหน่วยความจำขนาด 16 ไบต์

2.11.2 หน่วยความจำแบบอ่านได้เขียนได้ (RAM)

หน่วยความจำประเภท RAM (Random Access Memory) เป็นหน่วยความจำที่สามารถอ่านและเขียนข้อมูลกี่ครั้งก็ได้ แต่ต้องมีแหล่งจ่ายไฟต่ออยู่ตลอดเวลา ถ้าไม่มีแหล่งจ่ายไฟ ข้อมูลที่เก็บเอาไว้จะหายไปทันที คำว่า Random Access หรือการเข้าถึงข้อมูลแบบสุ่ม หมายถึงว่าการอ่านข้อมูลที่ตำแหน่งใดๆ นั้นจะใช้เวลาในการอ่านเท่ากัน หน่วยความจำประเภทนี้ถูกสร้างเป็นไอซีสารกึ่งตัวนำ มีทั้งที่ใช้

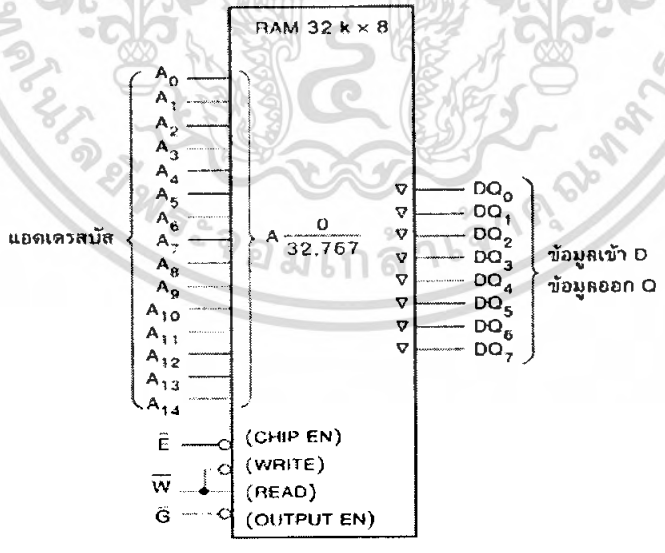
เทคโนโลยีแบบไบโพลาร์ เทคโนโลยีแบบ MOS และแบบที่ใช้ทั้งสองประเภทรวมกันเรียกว่า BiMOS หน่วยความจำ RAM แบบไบโพลาร์จะเป็นหน่วยความจำแบบสแตติก RAM(SRAM) ซึ่งเป็นหน่วยความจำที่สามารถเก็บข้อมูลได้ตลอดไปตราบที่มีไฟเลี้ยงต่ออยู่สำหรับแบบ MOS จะแบ่งออกเป็นหน่วยความจำแบบสแตติก RAM และแบบไดนามิก RAM(DRAM) ซึ่งเป็นหน่วยความจำที่ต้องมีสัญญาณกระตุ้นข้อมูลอยู่ตลอดเวลาเพื่อให้ข้อมูลยังคงอยู่ในหน่วยความจำ

2.11.3 หน่วยความจำแบบสแตติก RAM (SRAM)

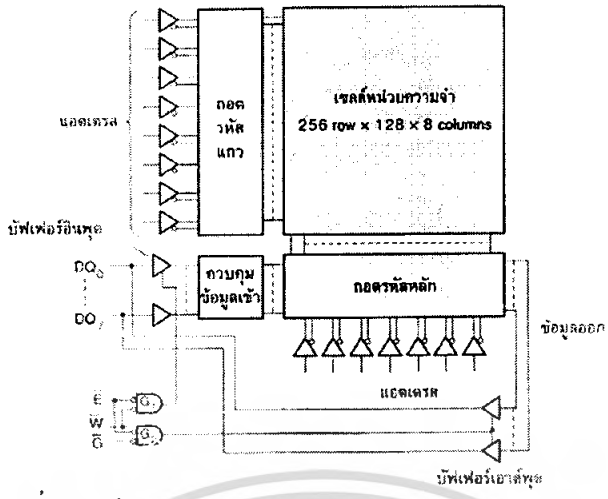
หน่วยความจำแบบนี้ โครงสร้างภายในประกอบด้วยเซลล์ข้อมูลขนาด 1 บิตต่อเรียงกัน และข้อมูลที่เก็บไว้จะคงอยู่ตลอดถ้ามีไฟเลี้ยงต่ออยู่ การเก็บข้อมูลของเซลล์แต่ละบิตจะถูกสร้างด้วยฟลิปฟลอปที่ทำด้วยทรานซิสเตอร์แบบไบโพลาร์ หรือ MOSFET หรือ CMOS แล้วแต่เทคโนโลยีที่ใช้ผลิต การเก็บข้อมูลโดยใช้ฟลิปฟลอปนี้จะต้องใช้พลังงานไฟฟ้าเพื่อให้ข้อมูลคงอยู่ ซึ่งจะเห็นว่าต่างจากการเก็บข้อมูลของ ROM ที่โครงสร้างภายในถูกออกแบบให้เป็นลอกจิก “0” หรือ “1” เลย

ในรูปที่ 2.46 แสดงสัญลักษณ์ทางลอจิกของหน่วยความจำ SRAM ขนาด 32k x 8 จะเห็นว่าประกอบด้วยแอดเดรสบัสจำนวน 15 เส้น ดาต้าบัสจำนวน 8 เส้น และกลุ่มสัญญาณควบคุม

ในหน่วยความจำ SRAM แต่ละตำแหน่งอาจเก็บข้อมูล 1 บิต, นิบเบิล(4 บิต) , ไบต์ (8 บิต) หรือหลายไบต์ก็ได้ ในปัจจุบันไอซีหน่วยความจำ SRAM หนึ่งตัว ถ้าหากนำข้อมูลทั้งหมดมารวมกันสามารถจุได้เป็นเมกะไบต์ ดังที่เห็นในคอมพิวเตอร์ทั่วไป โครงสร้างภายในของ RAM จะนำหน่วยเก็บข้อมูลแต่ละเซลล์มาต่อกันเป็นอาร์เรย์เช่นเดียวกับโครงสร้างของ ROM แสดงในรูปที่ 2.47



รูปที่ 2.46 สัญลักษณ์ของ 32k x 8 SRAM

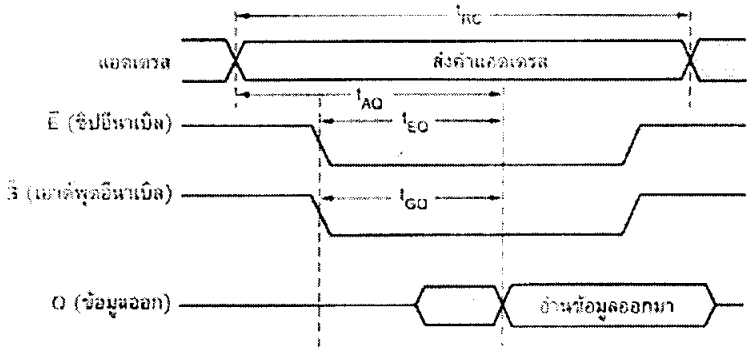


รูปที่ 2.47 โครงสร้างของ RAM แบบ 32k x 8

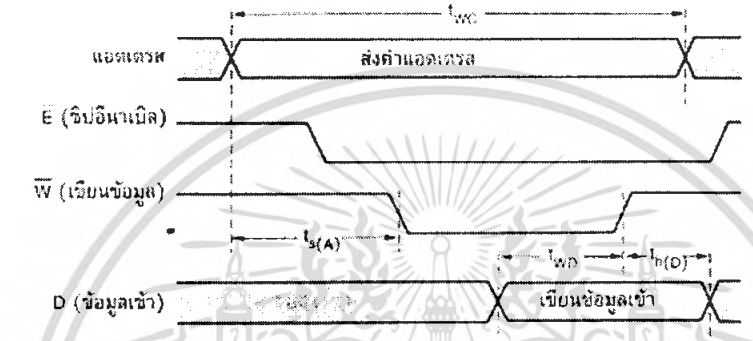
จากรูปที่ 2.47 แสดงโครงสร้างของ SRAM ขนาด 32k x 8 โดยประกอบด้วยเซลล์หน่วยความจำ 256 แถว แต่ละแถวมี 128 x 8 หลัก (128 หลัก แต่ละหลักมี 8 บิต) ทำให้เก็บข้อมูลได้ทั้งหมด 2^{15} หรือ 32,768 ตำแหน่งแต่ละตำแหน่งเก็บข้อมูล 8 บิต เนื่องจากเก็บข้อมูลได้ 32,768 ไบต์ จึงเรียก SRAM ขนาดนี้ว่า 32kB หรือ 262,144 บิต ขาชิปอีนาเบิล จะแอกทีฟ "0" ใช้สำหรับเลือกชิปให้ทำงาน ขา Write จะแอกทีฟ "0" ใช้สำหรับเขียนข้อมูลลงหน่วยความจำ ขาเอาต์พุตอีนาเบิล จะแอกทีฟ "0" ใช้สำหรับส่งข้อมูลออกทางเอาต์พุตของ SRAM และจะสังเกตเห็นว่ามีบัฟเฟอร์ต่ออยู่ทางคาต้าบัสของ SRAM เพื่อควบคุมการส่งข้อมูลเข้าออก

ในการใช้งาน SRAM ในรูปที่ 2.47 เริ่มแรกจะต้องให้ขาชิปอีนาเบิล เป็นลอจิก "0" เพื่อให้หน่วยความจำทำงาน จากนั้นจะส่งค่าแอดเดรสที่ต้องการติดต่อเข้าไป โดยส่ง 8 บิตแรก เข้าไปถอดรหัสทางด้านแถวเพื่อเลือกแถวใดแถวหนึ่งจาก 256 แถว และส่ง 8 บิตต่อมาเข้าไปถอดรหัสทางด้านหลักเพื่อเลือกหลักใดหลักหนึ่ง ถ้าต้องการอ่านข้อมูล จะทำให้ขา Write เป็น "1" และขาเอาต์พุตอีนาเบิล เป็น "0" จะทำให้เกต G_1 ได้เอาต์พุตเป็น "0" ส่วนเกต G_2 ได้เอาต์พุตเป็น "1" ทำให้บัฟเฟอร์ที่ต่ออยู่ทำงาน ทำให้ข้อมูลที่เก็บอยู่ถูกส่งออกมาทางคาต้าบัสได้

สำหรับการเขียนข้อมูล จะส่งค่าแอดเดรสที่ต้องการเขียนเข้าไปใน SRAM ตาขั้นตอนที่ได้กล่าวมา จากนั้นให้ขา Write เป็น "0" และขาเอาต์พุตอีนาเบิล เป็น "1" จะทำให้บัฟเฟอร์ที่ต่ออยู่กับ G_1 ทำงาน เมื่อมีการส่งข้อมูลเข้าไปทางคาต้าบัสจะทำให้ข้อมูลถูกเขียนเข้าไปในตัว SRAM ไดอะแกรมเวลาสำหรับการอ่าน/เขียนข้อมูลกับตัว SRAM แสดงได้ดังรูปที่ 2.48



(ก) ไซเกิลการอ่านข้อมูล

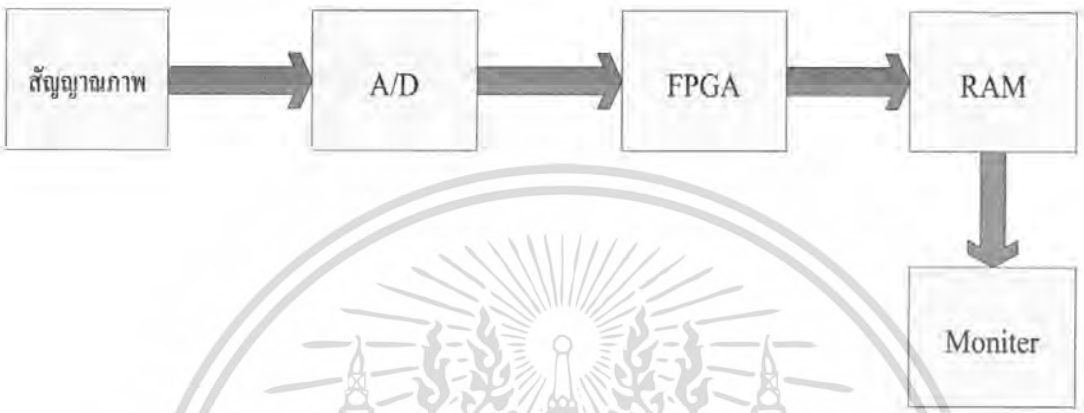


(ข) ไซเกิลการเขียนข้อมูล

รูปที่ 2.48 ไดอะแกรมเวลาการอ่าน/เขียนข้อมูลกับ SRAM

บทที่ 3
การคำนวณและการสร้าง

3.1 ภาพรวมการเชื่อมต่อของระบบ



รูปที่ 3.1 โครงสร้างของระบบโดยรวม

ส่วนประกอบทางด้านฮาร์ดแวร์

ส่วนประกอบทางด้านฮาร์ดแวร์ของระบบ ได้แก่

- วงจร A/D เป็นส่วนที่ทำหน้าที่แปลงสัญญาณภาพ
- RAM มีหน้าที่ในการเก็บข้อมูลไว้ ซึ่ง RAM จะใช้เป็นหน่วยในการพักข้อมูล ซึ่งขนาดความจุขึ้นอยู่กับความเหมาะสมของข้อมูลที่ต้องการจัดเก็บ
- วงจร MAX232 ซึ่งจะช่วยแปลงแรงดัน เพื่อให้ FPGA สามารถ ส่งข้อมูลไปยังพอร์ทอนุกรมของคอมพิวเตอร์ได้

ส่วนประกอบทางด้านซอฟท์แวร์

ส่วนประกอบทางด้านซอฟท์แวร์ของระบบ ได้แก่

- การเขียนโปรแกรม เพื่อกำหนด การ sampling เพื่อให้ได้ขนาดของภาพตามต้องการ
- การเขียนโปรแกรมเพื่อทำการรับค่า ข้อมูลที่ได้จาก วงจร A/D
- การเขียนโปรแกรม เพื่อสร้างแอดเดรส แล้วนำข้อมูลที่แปลงแล้วไปเก็บใน Block RAM ภายในหรือ Static RAM ซึ่งต่ออยู่ภายนอก FPGA

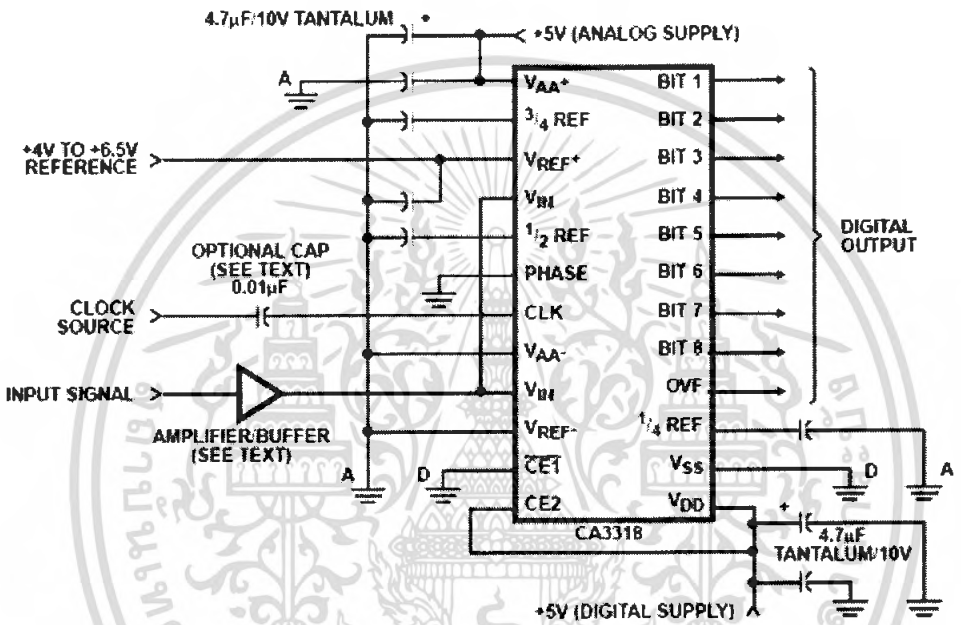
โดย ส่วนนี้ จะเขียนขึ้นด้วยภาษา VHDL แล้วโปรแกรมลงในส่วนของ FPGA เพื่อให้ FPGA ทำการควบคุมการทำงานของระบบต่อไป

การเขียนโปรแกรมเพื่อแสดงผลบนหน้าจอ ซึ่งการอ่านข้อมูลจะอ่านจากพอร์ทอนุกรม แล้วนำไปแสดงผลต่อไป

3.2 การแปลงสัญญาณอนาล็อกเป็นดิจิตอล

ในส่วนของการรับสัญญาณ จะรับสัญญาณ VGA จากคอมพิวเตอร์ ซึ่งสัญญาณ VGA นี้เป็นสัญญาณอนาล็อก จึงต้องทำการแปลงสัญญาณให้เป็นสัญญาณดิจิตอลเสียก่อน เพื่อที่จะให้สามารถประมวลผลใน FPGA ได้

การนำสัญญาณ VGA เข้าสู่วงจรแปลงสัญญาณอนาล็อกเป็นดิจิตอลนั้น เราจะใช้สายเชื่อมต่อพอร์ต VGA เข้าสู่วงจรแปลงสัญญาณโดยตรง สัญญาณที่เรานำมาใช้ในวงจรแปลงสัญญาณอนาล็อกเป็นดิจิตอลนั้น เราจะใช้สัญญาณสี RGB (Red Green Blue)



รูปที่ 3.2 ลักษณะการต่อวงจรแปลงสัญญาณอนาล็อกเป็นดิจิตอลโดยใช้ไอซีเบอร์ CA3318

ในส่วนของการแปลงสัญญาณอนาล็อกเป็นดิจิตอลนั้นจะใช้ไอซีเบอร์ CA3318 ซึ่งเป็น CMOS parallel (FLASH) analog-to-digital converter ในการแปลงสัญญาณ VGA ให้เป็นสัญญาณดิจิตอลที่มีขนาด 3 บิต

ไอซีเบอร์ CA3318 เป็นไอซีที่สามารถทำงานได้ในช่วงโวลเตจอินพุตกว้าง คือตั้งแต่ 4V – 7.5V ด้วยกำลังสูงสุด บนความถี่สัญญาณนาฬิกาที่ใช้ เมื่อทำการป้อนสัญญาณ 5V ความถี่สัญญาณนาฬิกาเป็น 15MHz จะได้กำลังเป็น 150mW

ด้วยอัตราการแปลงที่มีค่าสูง ทำให้ CA3318 เหมาะสำหรับแปลงสัญญาณดิจิตอลความเร็วสูง มีอัตราการแซมปลิ้งอยู่ที่ 15MHz – 30MHz ในโครงงานนี้เราใช้สัญญาณนาฬิกาความถี่ 15MHz

3.3 วงจรภายใน FPGA

วงจรภายใน FPGA ที่สร้างขึ้น ประกอบด้วย 4 ส่วนดังนี้

1. ส่วนบล็อกแรมภายใน FPGA
2. ส่วนที่ใช้ในการเก็บข้อมูลลงในบล็อกแรม
3. ส่วนที่ใช้ในการควบคุมขนาดของภาพที่จะเก็บลงในบล็อกแรม
4. ส่วนที่ทำการส่งต่อข้อมูลออกจากบล็อกแรม

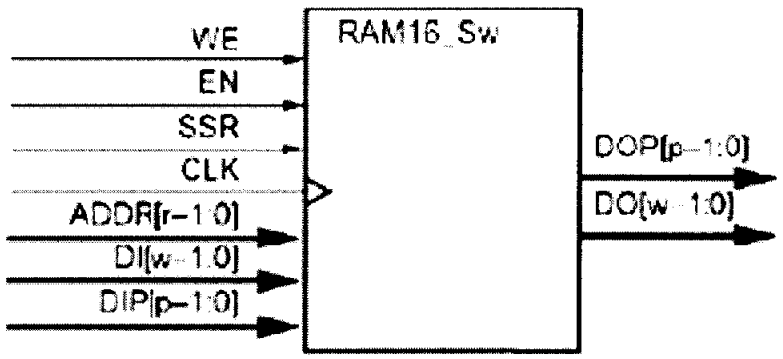
ในแต่ละส่วนมีรายละเอียดดังนี้

- ส่วนบล็อกแรมภายใน FPGA

ภายใน FPGA จะมีบล็อกแรมอยู่แล้วซึ่งในส่วนของบล็อกแรมนี้จะไม่รวมอยู่กับส่วนของเกตภายใน FPGA เราสามารถทำการเรียกใช้โดยไม่เสียจำนวนเกตไปโดยเปล่าประโยชน์ ความจุของบล็อกแรมภายใน FPGA รุ่นที่นำมาใช้คือ 288k ตารางที่ 3.1 จะแสดงขนาดของบล็อกแรมภายในของ FPGA ในแต่ละรุ่น ซึ่งลักษณะของบล็อกแรมภายใน FPGA จะมีลักษณะตามรูปที่ 3.3 ส่วนในการศึกษาการใช้ภาษา VHDL เรียกใช้บล็อกแรมนี้ เราจะทำการทดลองเรียกใช้บล็อกแรมขนาด 2k เพื่อทดสอบหลักการในการใช้ภาษา VHDL และเพื่อศึกษาโครงสร้างต่างๆของ FPGA

ตารางที่ 3.1 ขนาดของบล็อกแรมภายใน FPGA แต่ละรุ่น

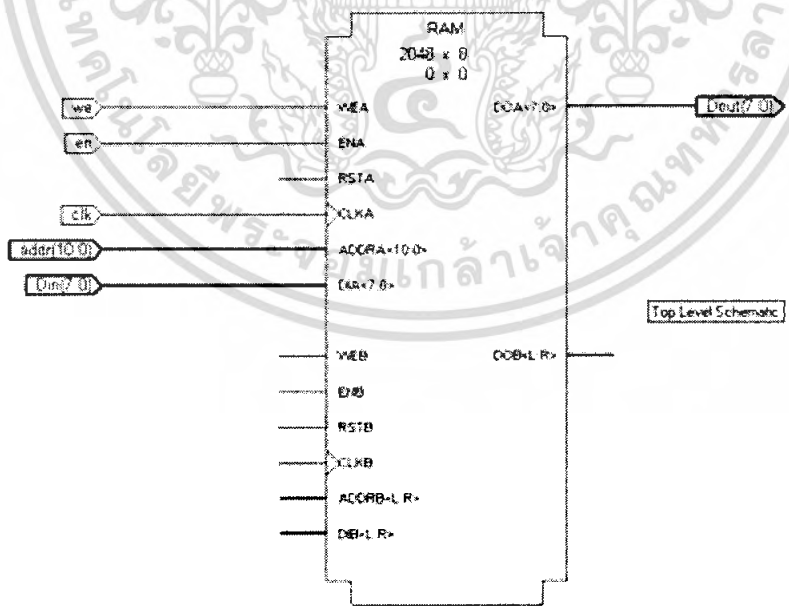
Device	RAM Columns	RAM Blocks Per Column	Total RAM Blocks	Total RAM Bits	Total RAM Kbits
XC3S50	1	4	4	73,728	72K
XC3S200	2	6	12	221,184	216K
XC3S400	2	8	16	294,912	288K
XC3S1000/L	2	12	24	442,368	432K
XC3S1500/L	2	16	32	589,824	576K
XC3S2000	2	20	40	737,280	720K
XC3S4000/L	4	24	96	1,769,472	1,728K
XC3S5000	4	26	104	1,916,928	1,872K



รูปที่ 3.3 ตัวอย่างบล็อกแรมภายใน FPGA

- ส่วนที่ใช้ในการเก็บข้อมูลลงในบล็อกแรม

ในการที่เราจะสามารถเก็บข้อมูลลงไปในแรมได้นั้นเราจะต้องทำการกำหนดแอดเดรสเพื่อที่จะให้ข้อมูลที่ต้องการเก็บถูกเขียนลงไปเก็บในแรมในตำแหน่งที่ถูกต้อง ดังนั้นจึงต้องมีการเขียนโปรแกรมเพื่อสร้างแอดเดรสป้อนให้กับแรมที่ใช้ด้วย นอกจากอินพุต และ แอดเดรสที่ทำการป้อนให้แรมแล้ว ยังต้องมีสัญญาณนาฬิกาอีกด้วย เพื่อเป็นการให้จังหวะในการเก็บข้อมูล ในทางกลับกันการอ่านข้อมูลที่อยู่ภายในแรมนั้นก็จะต้องมีจังหวะในการอ่านข้อมูลเช่นกัน โดยแรมจะทำการอ่านหรือเขียนข้อมูลดูจากสัญญาณเอนนาเบิล (en)



รูปที่ 3.4 การเขียนข้อมูลลงในแรมและการอ่านข้อมูลจากแรม

- ส่วนที่ใช้ในการควบคุมขนาดของภาพที่จะเก็บลงในบล็อกแรม

เนื่องจากบล็อกแรมที่อยู่ภายใน FPGA มีขนาดที่จำกัด เรื่องของการกำหนดขนาดภาพที่จะเก็บลงไปในบล็อกแรมจึงเป็นเรื่องที่สำคัญ โดยขนาดภาพที่ใช้ในโครงงานนี้คือ 128x128 ซึ่งแต่ละจุดจะมีความสว่างด้วยกัน 8 ระดับ ในส่วนนี้จะทำการความคุมให้สัญญาณก่อนเข้าไปเก็บลงในบล็อกแรมมีขนาดตามที่เราต้องการได้

- ส่วนที่ทำการส่งต่อข้อมูลออกจากบล็อกแรม

เป็นส่วนที่ทำหน้าที่ต้องกันข้ามกับส่วนการนำข้อมูลเก็บลงในบล็อกแรม โดยดูจากสัญญาณเอนนาเบิล (en) เพื่อเป็นการกำหนดว่าในช่วงเวลานั้นเป็นการเขียนหรืออ่านข้อมูล

บทที่ 4
การทดลองและผลการทดลอง

4.1 ทดลองต่อวงจรแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิตอล

เนื่องจากสัญญาณภาพที่ออกมาจากพอร์ท VGA หรือสัญญาณ VDO จะเป็นสัญญาณอนาล็อก จึงต้องมีการแปลงสัญญาณให้เป็นสัญญาณดิจิตอลก่อน โดยใช้ ไอซีเบอร์ CA3318 ซึ่งเป็นไอซีที่มีคุณสมบัติในการแปลงสัญญาณอนาล็อกเป็นดิจิตอล เพื่อใช้ในการประมวลผลในส่วนอื่นต่อไป

อุปกรณ์ที่ใช้ในการทดลอง

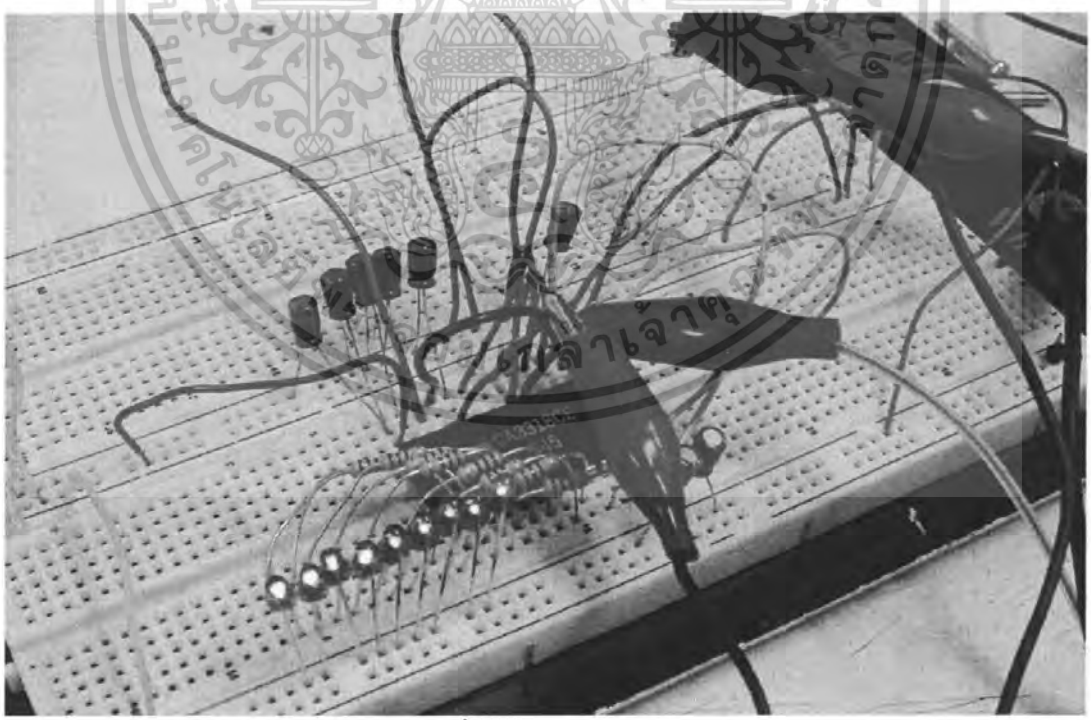
1. วงจร A/D โดยไอซีเบอร์ CA3318
2. ฟังก์ชันเจนเนอร์เรเตอร์
3. ออสซิลโลสโคป

จุดประสงค์การทดลอง

1. ทดสองวงจร A/D ที่ทำการต่อขึ้น
2. สังเกตลักษณะของสัญญาณเอาต์พุต

ขั้นตอนการทดลอง

1. ทำการต่อวงจร A/D โดยใช้ไอซีเบอร์ CA3318

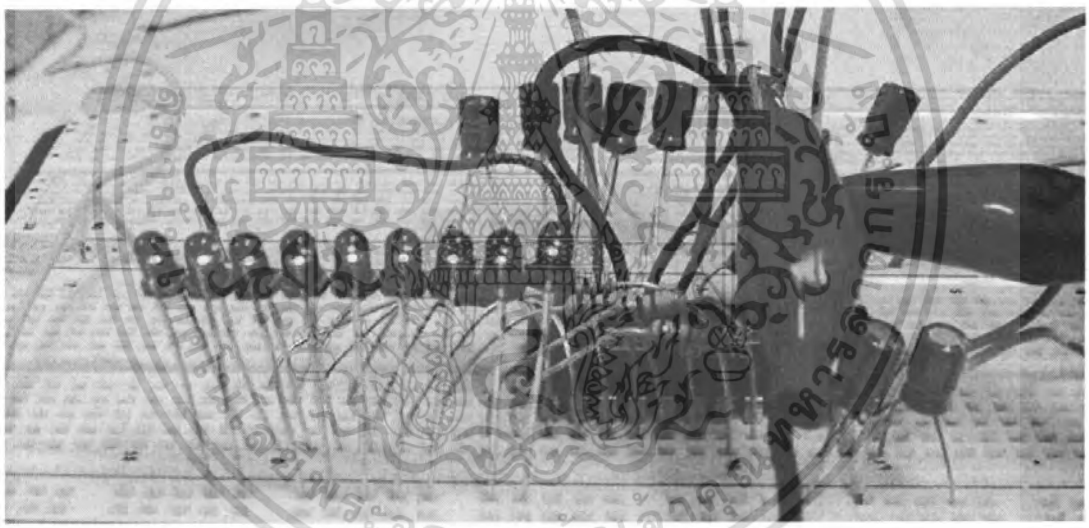


รูปที่ 4.1 การต่อวงจร A/D

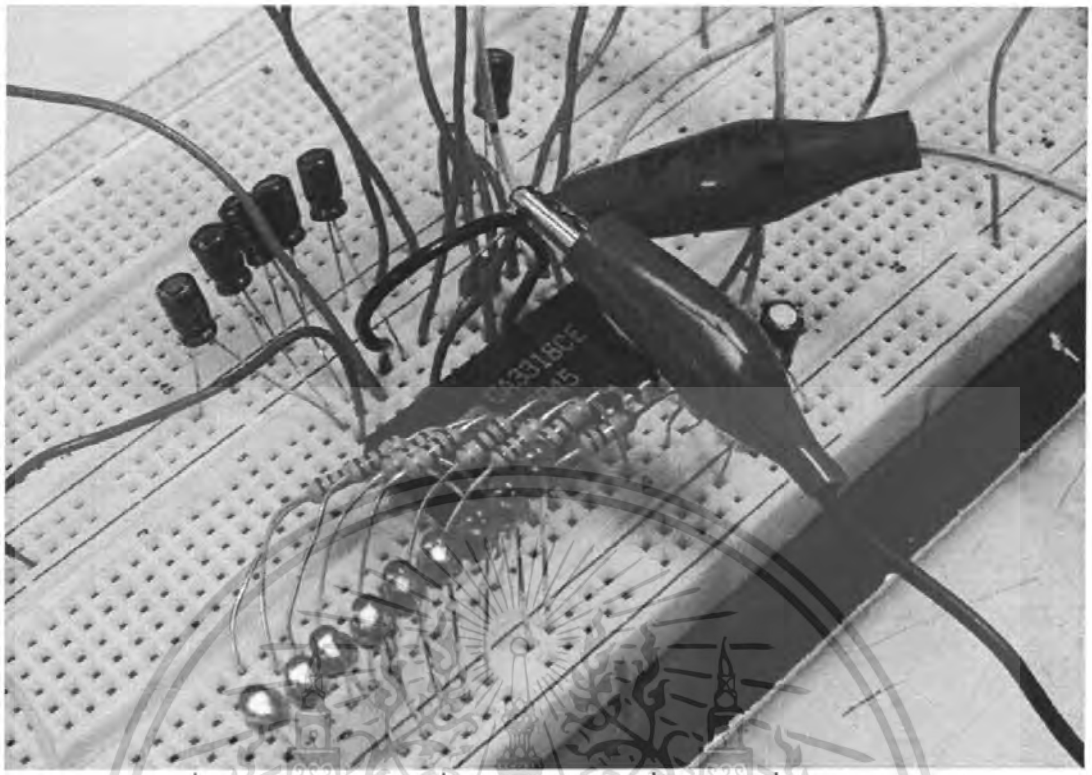
2. ทำการป้อนสัญญาณ อินพุทจากพอร์ท VGA ซึ่งทำการตั้งค่า การแสดงผลที่หน้าจอในขณะนั้นให้ไปเป็นสีแดง ในระดับความสว่างที่สว่างที่สุดคือ 256 ระดับ แล้วไล่ระดับความสว่างลงไปตามจำนวนบิต เช่น $2^8 = 256$ ระดับ , $2^7 = 128$ ระดับ

รูปที่ 4.2 การป้อนสัญญาณ ภาพสีในระดับต่างๆกัน

3. สังเกตเอาต์พุตที่ออกจากขา 1,2,3,4,5,6,7,8 ของไอซี ซึ่งต่อไว้กับ LED เพื่อดูว่า ผลที่ออกมา ทำให้ความสว่างของ LED ลดลงตามจำนวนของระดับที่ลดลงหรือไม่



(a)



รูปที่ 4.3(a),4.3(b) เอาต์พุตที่ได้เปรียบเทียบกันเมื่อทำการเปลี่ยนระดับสี

ผลการทดลอง

จากการทดลอง เมื่อทำการป้อนสัญญาณ R ซึ่งเป็นสัญญาณสีแดงเป็น อินพุทของวงจร ซึ่งสว่างในระดับที่ 256 จะทำให้หลอด LED ติดทุกหลอด และเมื่อทำการลดระดับสีลงเหลือ 128 , 64 ,32 ... ไปเรื่อยๆ จะพบว่า จำนวนหลอด LED ที่ติดลดลงเรื่อยๆ ตามระดับความสว่างของภาพที่ลดลง ซึ่งพบว่าเป็นไปตามทฤษฎี

4.2 การทดลองเขียนโปรแกรมการรับข้อมูลของ FPGA

หลังจากที่ทำการแปลงสัญญาณที่มาจาก VGA เป็นสัญญาณดิจิตอลแล้ว จะต้องนำค่าสัญญาณดิจิตอลที่ได้มาเก็บค่าต่อไป ซึ่งจะใช้ FPGA มาเป็นส่วนในการรับค่า ซึ่งทดลองโดยใช้การให้ FPGA ทำหน้าที่เป็น RAM แล้วลอง Write และ Read ข้อมูลว่าตรงกันหรือไม่

อุปกรณ์ที่ใช้ในการทดลอง

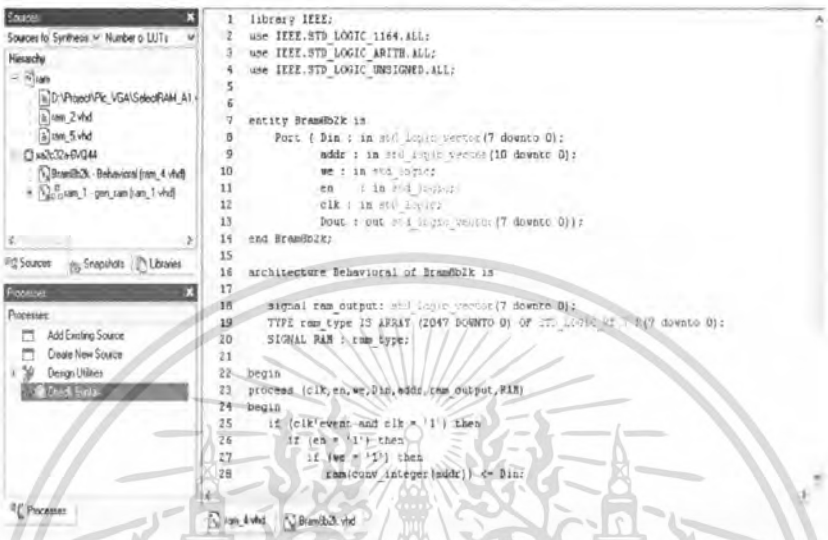
1. โปรแกรม Xilinx ISE 8.2i
2. โปรแกรม ModelSim XE III 6.2g

จุดประสงค์การทดลอง

1. เพื่อศึกษาการเขียนโปรแกรมด้วยภาษา VHDL
2. เพื่อศึกษาการทำงานของ Ram

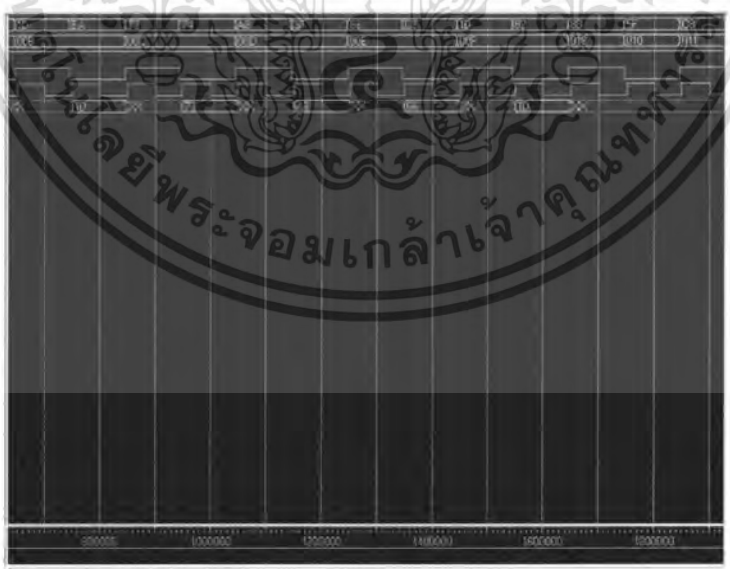
ขั้นตอนการทดลอง

1. ทดลองเขียนโปรแกรมการรับข้อมูลด้วยโปรแกรม Xilinx ISE 8.2i โดยทำการระบุ address ที่จะใช้เก็บข้อมูล และสุ่มค่าข้อมูลที่ต้องการจะเก็บลงไป



รูปที่ 4.4 การเขียนโปรแกรม เพื่อทำการระบุ address ในการเก็บค่าข้อมูล

2. Simulate ผลวงจรที่ได้ด้วยโปรแกรม ModelSim XE III 6.2g



รูปที่ 4.5 ผลจากการ simulate ด้วยโปรแกรม model sim

3. สังเกตผลที่ได้จากการ READ ข้อมูลว่าตรงกับที่ WRITE เข้าไปหรือไม่

ผลการทดลอง

จากการทดลองเมื่อทำการ เขียนโค้ดให้ FPGA โดยทำการสุ่มค่าข้อมูลที่จะนำเข้าไปเก็บ แล้วทำการ READ ข้อมูลที่เก็บเข้าไปออกมา พบว่า สามารถเก็บข้อมูลได้ถูกต้องตรงกับค่าที่ WRITE เข้าไป ซึ่งการ WRITE จะทำเมื่อขา WE (Write Enable) เป็นขอบขาขึ้น ซึ่งจะทำการเก็บค่า ตาม address ไปเรื่อยๆ ตามลักษณะการทำงานของ RAM

4.3 การทดลองเรียกใช้และเก็บค่าข้อมูล ไว้ใน Block RAM ภายใน FPGA

หลังจากนำเอาสัญญาณ VGA มาผ่านวงจรแปลงสัญญาณจากอนาล็อกเป็นดิจิตอลแล้ว ก็จะต้องใช้ FPGA มาช่วยในการเก็บค่าข้อมูลที่ส่งมา แล้วนำไปเก็บไว้ ยัง Block RAM ที่อยู่ภายใน FPGA ต่อไป ซึ่งจะทำการทดลอง เขียนโปรแกรม รับค่าข้อมูล แล้วส่งออกไป จากนั้นใช้ โปรแกรมที่สามารถอ่านค่าข้อมูลจากพอร์ทอนุกรม เพื่อตรวจสอบว่า ข้อมูลที่ได้รับมานั้น ตรงกับข้อมูลที่ส่งมาหรือไม่

อุปกรณ์ที่ใช้ในการทดลอง

1. โปรแกรม Xilinx ISE 8.2i
2. วงจร A/D โดยไอซีเบอร์ CA3318
3. โปรแกรมอ่านข้อมูลจากพอร์ทอนุกรม

จุดประสงค์การทดลอง

1. เพื่อศึกษาการเขียนโปรแกรมด้วยภาษา VHDL
2. เพื่อตรวจสอบว่า Block RAM ภายในสามารถรับข้อมูลที่ส่งมาได้ถูกต้องหรือไม่

ขั้นตอนการทดลอง

1. ทดลองเขียนโปรแกรมการรับข้อมูลด้วยโปรแกรม Xilinx ISE 8.2i โดยทำการระบุ address ที่จะใช้เก็บข้อมูลไว้ใน Block RAM และส่งข้อมูลที่ต้องการจะเก็บลงไป
2. ส่งสัญญาณภาพซึ่งเป็นภาพสีแดง ที่ระดับความสว่าง 128 ระดับ ขนาด 128 x 128

รูปที่ 4.6 ภาพสีแดงความสว่าง 128 ระดับ

3. ใช้โปรแกรมอ่านค่าข้อมูลที่รับเข้ามาทางพอร์ท อนุกรม จาก Block RAM ใน FPGA

```
            09:35 26 07 [RX] - 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F
7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F
7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F
7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F
7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F
7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F
7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F
7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F
7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F
7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F
7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F
7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F
7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F
7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F
7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F
7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F
7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F
7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F
7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F
7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F
7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F
7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F
7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F
7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F
7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F
```

รูปที่ 4.7 ข้อมูลที่อ่านได้จากการรับข้อมูลทางพอร์ทอนุกรม

4. ส่งสัญญาณภาพซึ่งเป็นภาพสีแดงและสีดำอย่างละครึ่ง ที่ระดับความสว่างสูงที่สุด ขนาด 128 x 128

รูปที่ 4.8 ภาพสีแดงดำ

5. ใช้โปรแกรมอ่านค่าข้อมูลที่รับเข้ามาทางพอร์ท อนุกรม จาก Block RAM ใน FPGA

Communication

ASCII | HEX | Decimal | Binary |

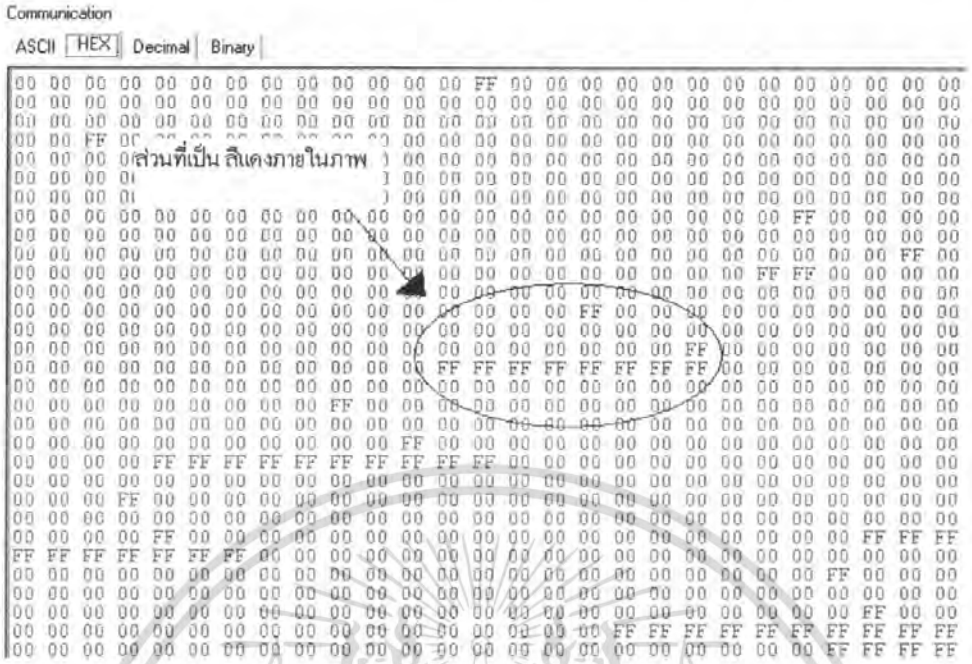
```
          09:42:17.50 [RX] - FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

รูปที่ 4.9 ข้อมูลที่อ่านได้จากการรับข้อมูลทางพอร์ทอนุกรม

6. ส่งสัญญาณภาพซึ่งเป็น ที่ระดับความสว่างสูงที่สุด

รูปที่ 4.10 ภาพทั่วไป

7. ใช้โปรแกรมอ่านค่าข้อมูลที่รับเข้ามาทางพอร์ท อนุกรม จาก Block RAM ใน FPGA



รูปที่ 4.11 ข้อมูลที่อ่านได้จากการรับข้อมูลทางพอร์ทอนุกรม

ผลการทดลอง

จากการที่ทำการ เขียนข้อมูลเพื่อให้ FPGA เป็นตัวรับข้อมูล และส่งต่อไปยัง Block RAM ภายใน แล้วส่งข้อมูลจาก ใน Block RAM นี้ไปยังพอร์ทอนุกรมของคอมพิวเตอร์ จากนั้นทำการตรวจสอบค่า ที่ได้รับจากพอร์ทอนุกรมของคอมพิวเตอร์ ในลักษณะ ที่เป็นเลขฐาน 16 แล้ว จะเห็นได้ว่า คอมพิวเตอร์สามารถอ่านข้อมูลที่ได้รับเข้ามาทางพอร์ทอนุกรมได้อย่างถูกต้อง โดยแต่ละค่าภายในภาพ ถือเป็น 1 จุด แต่ละจุดจะมี 8 บิต เนื่องจาก หน่วยความจำของ Block RAM ภายในมีอยู่จำกัด จึงสามารถส่งภาพได้เพียงขนาดไม่ใหญ่มากนัก ซึ่งจะเห็นได้ว่า ข้อมูลที่ส่งมากับ ที่รับได้มีค่าตรงกัน

4.4 การทดลอง เรียกข้อมูลที่เก็บใน Block RAM แล้วนำไปแสดงผลที่หน้าจอ

จากการทดลองก่อนหน้า เมื่อทำการอ่านค่าข้อมูลที่เก็บได้ถูกต้องแล้ว นำเอาข้อมูลที่ได้นั้นมาแสดงผลบนหน้าจอ โดยผ่านการเขียนโปรแกรมเพื่อนำข้อมูลที่ผ่านเข้ามาทางพอร์ทอนุกรมมาจัดเรียงแล้วแสดงผลตามขนาดภาพ

อุปกรณ์ที่ใช้ในการทดลอง

1. โปรแกรมแสดงผล
2. วงจร A/D โดยไอซีเบอร์ CA3318
3. FPGA
4. วงจรแปลงแรงดัน

จุดประสงค์การทดลอง

เพื่อทำการตรวจสอบภาพที่เก็บในบล็อกแรม ว่าเมื่อแสดงผลแล้วตรงกับภาพที่ส่งมาหรือไม่ เป็นการตรวจสอบผลและ ดูคุณภาพของภาพที่สามารถแสดงได้

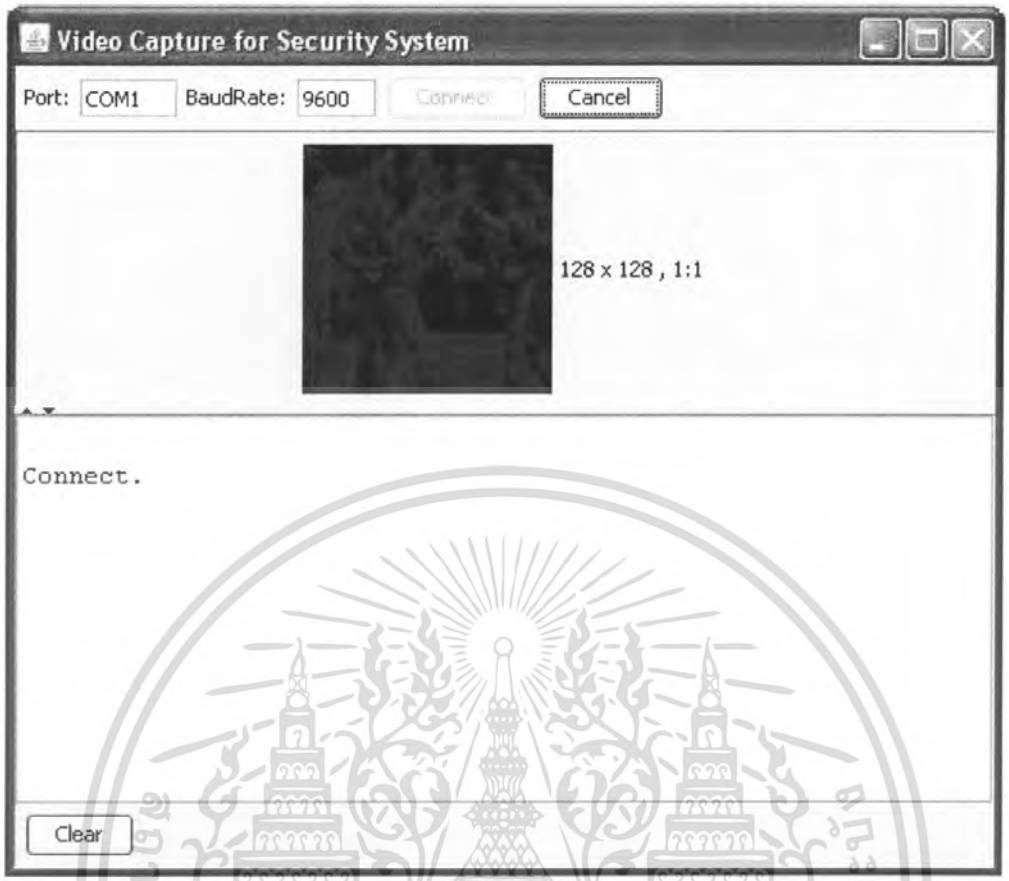
ขั้นตอนการทดลอง

1. ทดลองเขียนโปรแกรมการรับข้อมูลด้วยโปรแกรม Xilinx ISE 8.2i โดยทำการระบุ address ที่จะใช้เก็บข้อมูลไว้ใน Block RAM และส่งข้อมูลที่ต้องการจะเก็บลงไป

2. ส่งสัญภาพที่เป็นภาพทั่วไปจากหน้าจอ VGA ซึ่งเป็นหน้าจอที่ความละเอียด 800x600

รูปที่ 4.12 ภาพทั่วไป ระดับสีปกติ

3.ใช้โปรแกรมที่เขียนขึ้นเพื่อแสดงผลข้อมูลภาพที่รับเข้ามาทางพอร์ทอนุกรม



รูปที่ 4.13 ผลจากโปรแกรมที่ใช้แสดงผลบนหน้าจอ

ผลการทดลอง

จากการทดลองพบว่า เมื่อทำการส่งสัญญาณภาพซึ่งเป็นสัญญาณ R ผ่านเข้าวงจร A/D แล้วนำไปประมวลผลตามกระบวนการต่างๆ ใน FPGA แล้ว จะเห็นได้ว่า ภาพที่ส่งไปสามารถแสดงผลได้ในขนาด 128x128 ซึ่งขนาดของภาพมีขนาดเล็กลงก็เพราะ ขนาดความจุของ Block RAM มีปริมาณที่จำกัด

บทที่ 5
บทวิจารณ์และบทสรุป

5.1 สรุปผลการทดลอง

โครงงานนี้เป็นโครงงานที่นำเสนอการแสดงข้อมูลภาพ โดยใช้อุปกรณ์ FPGA (Field Programmable Gate Arrays) เป็นตัวควบคุมการทำงานทุกส่วนที่เกี่ยวกับสัญญาณดิจิตอล นี้ ทางผู้จัดทำโครงงานได้ทำการศึกษาสัญญาณภาพเป็นสัญญาณอินพุต ทำให้ได้รับความรู้ในเรื่องของการสแกนภาพ การส่งสัญญาณภาพผ่านพอร์ทอนุกรมของคอมพิวเตอร์ แล้วสามารถส่งต่อไปยัง BlockRAM ที่อยู่ภานใน FPGA และ Static RAM ที่ต่ออยู่ภายนอก แล้วนำภาพที่ส่งไปยัง RAM นั้นมาแสดงผลบนหน้าจอ ซึ่งช่วยให้ได้เรียนรู้วิธีการใช้งาน ภาษา VHDL มากขึ้นอีกด้วย โดยทางผู้ทดลองยังต้องการจะใช้การแสดงผลนี้กับ อินพุทที่เป็นภาพเคลื่อนไหว ซึ่งจะต้องมีการใช้ A/D เข้ามาแปลงสัญญาณก่อนจะส่งออกไป แต่ขณะนี้ยังไม่สามารถแก้ปัญหาเรื่องการหน่วงเวลาในการส่งออกไปเป็นภาพเคลื่อนไหวได้

5.2 วิจารณ์ผลการทดลอง

โครงงาน นี้ใช้สัญญาณภาพจากคอมพิวเตอร์เพื่อส่งออกไปยัง FPGA ให้ FPGA เป็นตัวบอกแอดเดรสที่ข้อมูลจะไปเก็บใน Block RAM ภายใน และ Static RAM ภายนอก โดยใช้สัญญาณ Hor Sync และ Ver Sync เป็นตัวช่วยกำหนดการทำงาน ซึ่งข้อมูลที่นำไปเก็บใน Block RAM ภายใน จะสามารถเรียกดูได้จาการเขียน โปรแกรมเพื่ออ่านข้อมูลจากพอร์ทอนุกรม แต่เนื่องจาก BlockRAM ภายในมีงานเล็ก ดังนั้นจึงไม่สามารถแสดงภาพที่มีความละเอียดสูงและมีขนาดใหญ่ได้ ส่วนในข้อมูลที่ถูกส่งไปที่ Static RAM ภายนอกซึ่งเป็นข้อมูลแบบขนาน แล้วใช้ FPGA ช่วยแปลงข้อมูลจากแบบขนาน ให้เป็นอนุกรม เพื่อส่งผ่านข้อมูลไปยังพอร์ทอนุกรมทางคอมพิวเตอร์ เพื่อแสดงผลข้อมูลบนหน้าจอ แต่เนื่องจากการใช้ RAM ภายนอก จะต้องมีการป้อนข้อมูลกลับเข้ามาใน FPGA เพื่อแปลงข้อมูลให้สามารถส่งต่อไปยังพอร์ทอนุกรมได้ จึงต้องมี บัฟเฟอร์ 2 ทิศทางเพื่อให้สามารถป้อนกลับข้อมูลได้ ยังคงอยู่ในการออกแบบวงจรเพื่อพัฒนาต่อไป

5.3 ปัญหาและอุปสรรคที่พบในการทำโครงงาน

1. Block RAM ที่อยู่ภายใน FPGA มีขนาดเล็ก จึงไม่สามารถเก็บภาพที่มีขนาดใหญ่ หรือละเอียดมากได้

2. ในการนำสัญญาณที่ส่งไปที่ RAM แล้วจะนำมาแสดงผล ข้อมูลเป็นข้อมูลแบบขนาน จำเป็นจะต้องมีวงจรเพิ่มเพื่อช่วยในการแปลงข้อมูลจากแบบขนานมาเป็นแบบ อนุกรม แล้วต้องนำข้อมูลอนุกรมมาแสดงผลนั้น ค่อนข้างเป็นปัญหาเนื่องจาก วงจร RAM ที่ใช้ในตอนแรก ไม่สามารถป้อนกลับข้อมูลเข้ามาใน FPGA เพื่อทำการแปลงได้ อีกทั้งการจัดเรียงข้อมูลที่ส่งมาให้สามารถแสดงผลได้นั้นทำได้ค่อนข้างยาก จึงยังคงอยู่ในส่วนของการปรับปรุงและพัฒนาชิ้นงานให้มีความสมบูรณ์ต่อไป

3. การต่อใช้งาน RAM ภายนอก เป็นวงจรที่ยุ่งยาก เพราะว่า RAM มีขา แอดเดรสที่จะต้องเชื่อมถึงกัน เพื่อให้การทำงานเป็นไปอย่างสอดคล้องกัน จึงทำให้วงจรที่ได้มีขนาดใหญ่ และซับซ้อน ต้องใช้

เวลาในการศึกษา ลักษณะของวงจรอยู่นาน อีกทั้งยังต้องคำนึงถึงส่วนที่จะสามารถป้อนกลับข้อมูลเข้าไปในวงจรเพื่อให้ สามารถแปลงข้อมูลจากแบบขนานให้กลายเป็นแบบอนุกรมอีกด้วย

4. การใช้ภาษา VHDL ซึ่งเป็นภาษาที่ใช้กับ FPGA เป็นเรื่องที่ต้องใช้เวลาในการศึกษา และต้องทำความเข้าใจ จึงทำให้ ต้องแก้ไข code ที่เขียนอยู่หลายครั้งจึงจะได้ผล

5.4 ประโยชน์ที่ได้รับจากโครงงานนี้

1. เข้าใจเรื่องการสแกนของสัญญาณภาพ

2. เข้าใจการทำงานของวงจรแปลงสัญญาณอนาล็อกเป็นดิจิตอลจนสามารถ ออกแบบและใช้งานวงจรได้มีประสิทธิภาพระดับหนึ่ง

3.ได้ความรู้เรื่องการใช้ภาษา VHDL ในการออกแบบวงจรดิจิตอลใน FPGA เพื่อใช้เป็นส่วนประมวลผลต่างๆ

4. เข้าใจขั้นตอนการทำงานของการเขียนข้อมูลลงในแรม

เอกสารอ้างอิง

1. ชำนาญ ปัญญาใส, วัชรากร หนูทอง,ภาษา VHDL สำหรับการออกแบบวงจรดิจิตอล,กรุงเทพฯ : ซีเอ็ดยูเคชั่น, 2547
2. ธีรวัฒน์ ประกอบผล , ดิจิตอลลอจิก , กรุงเทพฯ : ซีเอ็ดยูเคชั่น, 2546
3. เจน สงสมพันธุ์ , นิคมอนันต์ทิพย์ , เทคโนโลยี โทรทัศน์ , กรุงเทพฯ : สถาบันอิเล็กทรอนิกส์กรุงเทพรังสิต , 2549
4. http://elec.chandra.ac.th/learn/tipntrick/register/default.htm
5. http://www.alldatasheet.com/datasheet-pdf/pdf/66360/INTERSIL/CA3318.html