สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

การแสดงคลื่นสัญญาณหัวใจผ่านจอ VGA

ECG Monitor With VGA

T055448

โดย

นายกิตติพงษ์ ขาวลำลี

นางสาวชารุวรรณ ศิริรักษ์

นายจิรพงษ์ วีระอาชากุล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

เลขหมู่

เลขทะเบียน 55448

วัน,เดือน,ปี 9 พ.ค. 2548

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

b

i

การแสดงคลื่นสัญญาณหัวใจผ่านจอ VGA

ECG Monitor With VGA

โดย

นายกิตติพงษ์ ขาวสำลี 43010027

นางสาวจารุวรรณ ศิริรักษ์ 43010060

นายจิรพงษ์ วีระอาชากุล 43010063

อาจารย์ที่ปรึกษา

ผศ.ดร.ยุทธนา คิดใจเดียว

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปริญญานิพนธ์ปีการศึกษา 2546

ภาควิชาอิเลคทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การแสดงคลื่นสัญญาณหัวใจผ่านจอ VGA

ผู้จัดทำ

1. นายกิตติพงษ์ ขาวสำลี 43010027
2. นางสาวจารุวรรณ ศิริรักษ์ 43010060
3. นายจิรพงษ์ วีระอาชากุล 43010063

.....อาจารย์ที่ปรึกษา

(ผศ.ดร.ยุทธนา คิดใจเดียว)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การแสดงผลคลื่นไฟฟ้าหัวใจผ่านจอ VGA

นาย กิตติพงษ์ ขาวสำลี 43010027
นางสาว จารุวรรณ ศิริรักษ์ 43010060
นาย จิรพงษ์ วีระอาชากุล 43010063

ผศ.ดร.ยุทธนา คิดใจเดียว อาจารย์ที่ปรึกษา

ปีการศึกษา 2546

บทคัดย่อ

แนวคิดในการนำเสนอการใช้งานของ FPGA ให้สามารถรองรับความคิดริเริ่มต่างๆในปัจจุบันเริ่มเห็นได้ชัดเจนมากขึ้นการนำเอาFPGA และการเขียนคำสั่งในรูปแบบการบรรยายเชิงพฤติกรรมโดยใช้ภาษา VHDL มาใช้ออกแบบวงจรที่มีคุณสมบัติเป็นเครื่องวัดสัญญาณคลื่นไฟฟ้าหัวใจ เป็นการออกแบบโดยมุ่งไปที่การทดแทนเครื่องวัดสัญญาณคลื่นไฟฟ้าหัวใจที่ใช้ทั่วไปในโรงพยาบาล ข้อได้เปรียบของการออกแบบนี้คือการลดต้นทุนในการใช้เครื่องวัดที่มีอยู่แล้วใช้ชิพ FPGA มาทดแทน การแสดงผลผ่านจอมอนิเตอร์ของเครื่องคอมพิวเตอร์ซึ่งเป็นการหาได้ง่ายกว่า การจัดซื้ออุปกรณ์เครื่องวัดสัญญาณคลื่นหัวใจจากต่างประเทศที่มีราคาสูง เป็นประโยชน์ต่อการลดทุนขององค์กรได้มาก ดังนั้นข้อสำคัญของการออกแบบคือ การออกแบบให้เครื่องวัดสัญญาณคลื่นหัวใจมีประสิทธิภาพสามารถนำไปใช้งานได้จริง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ECG Monitor with VGA

Mr. Kittipong Khawsamlee 43010027

Miss Charuwan sirirak 43010060

Mr. Jirapong Wera-archakul 43010063

Abstract

Nowadays the circuit designed with FPGA is going to be useful and more popular in this project, the FPGAs and their behaviors described via VHDL are used to design an ECG monitor, this has a benefit in decreasing the import demand of ECG monitors by using a VGA monitor, the cost of product is very low. However, this project was designed to meet the operate performance.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

โดยที่ในการทำวิจัยและวิทยานิพนธ์ เรื่อง กลุ่มกระผมได้รับความช่วยเหลือเป็นอย่างมาก ทั้งด้านอุปกรณ์เครื่องใช้ คำแนะนำและข้อมูลต่างๆจากบุคคลต่อไปนี้

1. ผศ.ดร.ยุทธนา คิดใจเดียว อาจารย์ ประจำภาควิชาอิเล็กทรอนิกส์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
2. อ.ประภากร สุวรรณะ อาจารย์ ประจำภาควิชาอิเล็กทรอนิกส์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
3. รศ.ดร.มนัส สังวรศิลป์ อาจารย์ ประจำภาควิชาอิเล็กทรอนิกส์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
4. รศ.ดร.สุรพันธุ์ เอื้อไพบูลย์ อาจารย์ ประจำภาควิชาอิเล็กทรอนิกส์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ทั้งนี้ขอขอบคุณห้อง Biomedical Signal and Image Processing Laboratory ที่เอื้อเฟื้ออุปกรณ์ในการทดลองครั้งนี้ กลุ่มกระผมจึงขอแสดงความขอบพระคุณ เป็นอย่างสูงไว้ ณ.ที่นี้

นาย กิตติพงษ์ ขาวสำลี
นาย จารุวรรณ ศิริรักษ์
นาย จิรพงษ์ วีระอาชากุล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

หน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

หน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ(ต่อ)

หน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ(ต่อ)

หน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูปภาพ

หน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูปภาพ (ต่อ)

หน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูปภาพ (ต่อ)

หน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูปภาพ (ต่อ)

หน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง

หน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
บทนำ

1.1 ความเป็นมา

ทุกวันนี้เครื่องมือทางการแพทย์ที่ใช้ในประเทศไทย โดยรวมแล้วยังคงต้องจัดซื้อจากต่างประเทศ ซึ่งใช้งบประมาณในการจัดซื้อจำนวนมหาศาล เนื่องมาจากการผลิตอุปกรณ์ เครื่องมือแพทย์ที่ใช้ในการวัดในต่างประเทศที่มีคุณภาพ เพราะการตรวจดูแลคนไข้ของแพทย์จำเป็นต้องอาศัยเครื่องมือที่มีความถูกต้องและแม่นยำสูงมาก ซึ่งศักยภาพของผู้ผลิตเครื่องมือวัดในประเทศไทยยังไม่สามารถตอบสนองได้ความถูกต้องของผลการวัดที่ไม่น่าเชื่อถือ จึงเป็นสาเหตุสำคัญที่ต้องพึ่งอุปกรณ์เหล่านี้จากต่างประเทศ ซึ่งมีความจำเป็นที่ต้องใช้งบประมาณที่มากในการจัดซื้อ

ECG Monitor (Electrocardiograph: ECG) เป็นอุปกรณ์ที่มีความจำเป็นในการดูแลคนไข้และเป็นอุปกรณ์อยู่มากพอสมควร ซึ่งอุปกรณ์ชนิดนี้จะแสดงสัญญาณทางไฟฟ้าที่ร่างกายมนุษย์ผลิตออกมา เพื่อให้แพทย์ผู้รักษาสามารถวินิจฉัยการเปลี่ยนแปลงทางคลื่นไฟฟ้าต่างๆ เหล่านี้เพื่อโยงไปหา สาเหตุอาการป่วยของคนไข้ได้ แต่เนื่องจากเครื่องมือวัดชนิดนี้ยังมีราคาที่สูง การใช้งานของเครื่องมือที่มีประสิทธิภาพที่สูงชนิดนี้ จึงมักจะพบ ในโรงพยาบาลที่ใหญ่ และเป็นของเอกชน แต่บางแห่งยังคงใช้เครื่องที่มีการใช้งานมานาน เมื่อต้องการจะเปลี่ยนให้เป็นเครื่องใหม่นั้น จะต้องมีการพิจารณาในแง่ของความจำเป็น และงบประมาณที่สามารถจะซื้อได้ จึงเป็นสาเหตุที่โรงพยาบาลหลายแห่ง ยังขาดเครื่องวัดคลื่นไฟฟ้าหัวใจที่มีคุณภาพอยู่

แนวความคิดที่ต้องการผลิต ECG Monitor จึงมีออกมาโดยปริญญานิพนธ์นี้ ได้นำการออกแบบวงจรแบบการบรรยายพฤติกรรมเป็นภาษา VHDL ใช้งานร่วมกับชิพ FPGA และสามารถแสดงผลผ่านจอมอนเตอร์มาร่วมใช้งาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.2 วัตถุประสงค์

1.2.1 เพื่อศึกษาลักษณะของคลื่นหัวใจและความผิดปกติของสัญญาณคลื่นหัวใจ

1.2.2 เพื่อการออกแบบและการสร้างเครื่องแสดงสัญญาณคลื่นหัวใจด้วย FPGA และจอมอนิเตอร์

1.2.3 เพื่อสร้างเครื่องมือแพทย์ที่มีราคาถูกและสามารถใช้งานได้จริง

1.2.4 เพื่อศึกษาการออกแบบวงจรโดยใช้ชิพ FPGA ที่สามารถปรับปรุงคุณสมบัติการออกแบบโดยไม่ต้องเปลี่ยนแปลงวงจรทางกายภาพแต่สามารถเปลี่ยนแปลงโปรแกรมการออกแบบได้

1.3 เนื้อหาของโครงงาน

ในการสร้างโครงงานนี้มีเนื้อหาต่างๆ ที่สำคัญ ซึ่งแยกได้เป็นบทดังนี้

บทที่ 2 กล่าวถึงทฤษฎีการทำงานของคลื่นหัวใจ การเกิดคลื่นไฟฟ้า การศึกษารูปร่างของสัญญาณไฟฟ้าหัวใจ การวิเคราะห์คลื่นไฟฟ้าหัวใจ ความผิดปกติของจังหวะการเต้นของหัวใจ การตรวจวัดคลื่นไฟฟ้าหัวใจ

บทที่ 3 กล่าวถึงทฤษฎีของการสร้างภาพของจอมอนิเตอร์ การสร้างสีทางแสง หลักการเกิดภาพและสัญญาณต่างๆที่จำเป็นในการสร้างภาพ

บทที่ 4 กล่าวถึงความเป็นมาของชิพ FPGA และประเภทชิพ ASIC การแบ่งประเภทชิพที่ใช้ในปัจจุบัน

บทที่ 5 กล่าวถึงความเป็นมาของภาษา VHDL ลักษณะต่างๆของภาษาประเภทนี้ คุณสมบัติที่สำคัญ และรูปแบบการเขียนพร้อมตัวอย่าง

บทที่ 6 กล่าวถึงส่วนต่างๆของแบบ(Design Unit)

บทที่ 7 กล่าวถึงเทคโนโลยีของ FPGA

บทที่ 8 กล่าวถึงการทดลองใช้งานโปรแกรม MAX PLUS II

บทที่ 9 กล่าวถึงการออกแบบและการสร้าง โดยการเขียนภาษาแบบบรรยายพฤติกรรม โดยใช้โปรแกรม MAX PLUS II เริ่มจากการสร้างวงจรที่รับสัญญาณคลื่นหัวใจมาเก็บไว้ในอุปกรณ์ชิพ FPGA และประมวลสัญญาณเหล่านี้มาใช้ในการสร้างภาพออกจอมอนิเตอร์

บทที่ 10 เป็นการทดลองและวัดค่าต่างๆออกมา

บทที่ 11 เป็นการสรุปในตอนท้าย ระบุถึงปัญหาและอุปสรรคในการทำงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่2
สัญญาณไฟฟ้าหัวใจและหลักการแสดงคลื่นหัวใจ

2.1 บทนำ

เพื่อเป็นพื้นของความเข้าใจแนวความคิดและการทำงานของเครื่องวัดสัญญาณคลื่นไฟฟ้าหัวใจ ในบทนี้จะกล่าวถึงธรรมชาติของคลื่นไฟฟ้าหัวใจเสียก่อน

2.2 ธรรมชาติของคลื่นไฟฟ้าหัวใจ

คลื่นหัวใจมีแหล่งกำเนิดเป็นสัญญาณอิมพัลส์ที่บริเวณ SA node (Sinoartrial node) ในบริเวณผังหัวใจซีกบนขวา (right atrium) และการกระจายพร้อมกับรีโพลาไรซ์ไปทั่วร่างกาย เมื่อวัดโดยการต่อขั้วไฟฟ้าเข้าแบบดิฟเฟอร์เรนเชียล เข้ากับผิวหนังบริเวณ หน้าอก แขน ขา ได้ลูกคลื่น 1คาบเวลา คล้ายกับรูปที่แสดงในรูปที่ 2.3 ในคนปกติ คลื่นไฟฟ้าหัวใจจะประกอบด้วยคลื่น P, QRS, T และ U ซึ่งลักษณะการมีอยู่ของคลื่นองค์ประกอบเหล่านี้จะเป็นข้อมูลที่สำคัญในการวิเคราะห์การทำงานของหัวใจและความผิดปกติ

เมื่อเกิดอิมพัลส์และเริ่มกระจายไปบริเวณซีกบนของหัวใจ จะทำให้เกิดการบีบตัวของกล้ามเนื้อด้านบนของหัวใจโลหิตในหัวใจห้องบนด้านขวาหรือด้านซ้ายนั้น จะถูกบีบลงไปด้านล่างของหัวใจ อิมพัลส์นี้จะถูกแพร่กระจายผ่านเนื้อเยื่อตัวนำมายังผิวหนังจะทำให้เกิดสัญญาณที่เรียกว่า **P wave**

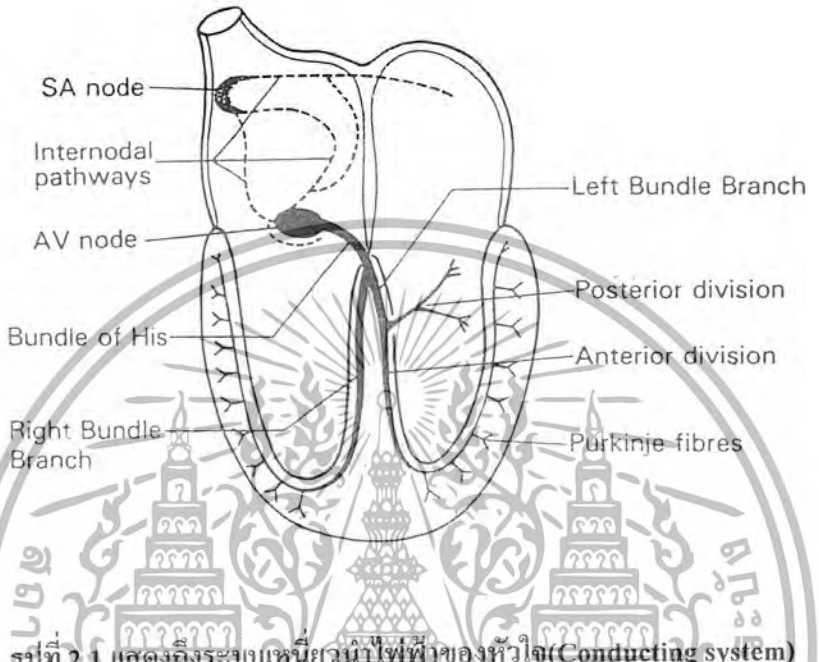
ในบริเวณ atrioventricular node จะมีการหน่วงเวลาของการกระตุ้นอิมพัลส์เพื่อให้เลือดสามารถถ่ายโอนจากด้านบนของหัวใจไปถึงด้านล่างของหัวใจสร็จสมบูรณ์ การหน่วงเวลานี้เป็นส่วนหลักของ P-R Interval ของรูปคลื่น ECG

อิมพัลส์ที่เกิดจะกระจายไปด้านล่างของหัวใจ ด้วยการคลายตัวของกล้ามเนื้อด้านล่างหัวใจ เมื่อขบวนการเสร็จสมบูรณ์จะมีคาบเวลาเกิดซ้ำขึ้นอีก และจะผลิตรูปคลื่นECGออกมาอีกครั้ง

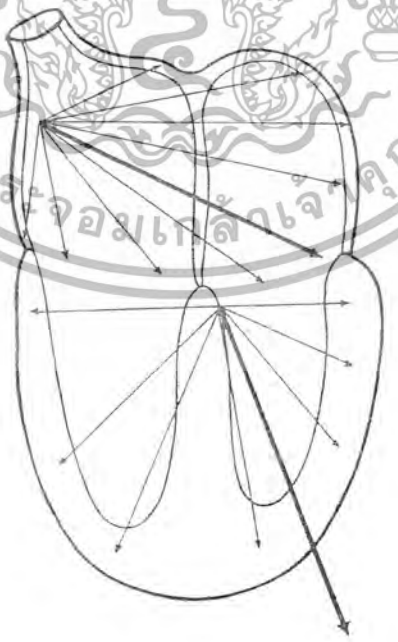
จะเห็นว่าส่วนผลิต รูปคลื่น ECG จะนำข้อมูลทางพยาธิสภาพของหัวใจ มาสู่แพทย์เพื่อวินิจฉัยอาการผิดปกติ ตัวอย่างเช่น R-R Interval สามารถบอกถึงอัตราการเต้นของหัวใจ (Cardiac Rythm) ภายใต้อิทธิพลของระบบประสาทอัตโนมัติ ความไม่เสถียรของ Cardiac Rythn สามารถบ่งถึงการเต้นที่ผิดปกติ ขนาดและช่วงเวลาของ P และ QRS บ่งชี้ถึงสภาพกล้ามเนื้อหัวใจ การลดทอนขนาดสัญญาณ อาจจะบ่งบอกของการทำลายกล้ามเนื้อหัวใจบริเวณที่เกี่ยวข้อง ขนาดเพิ่มอาจบอกความผิดปกติของหัวใจโต นอกจากนี้การหน่วงเวลานานเกินไปในจุด atrioventricular เป็นการบ่งบอกถึงการปิดกั้นของทั้งหมดหรือบางส่วนของอิมพัลส์ จากการขาดช่วงการซิงโรไนซ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ระหว่าง P-wave และ QRS Complex อาการผิดปกติสามารถรักษาได้ทางการใช้ยาและสังเกตผลการรักษาจากรูปคลื่น ECG

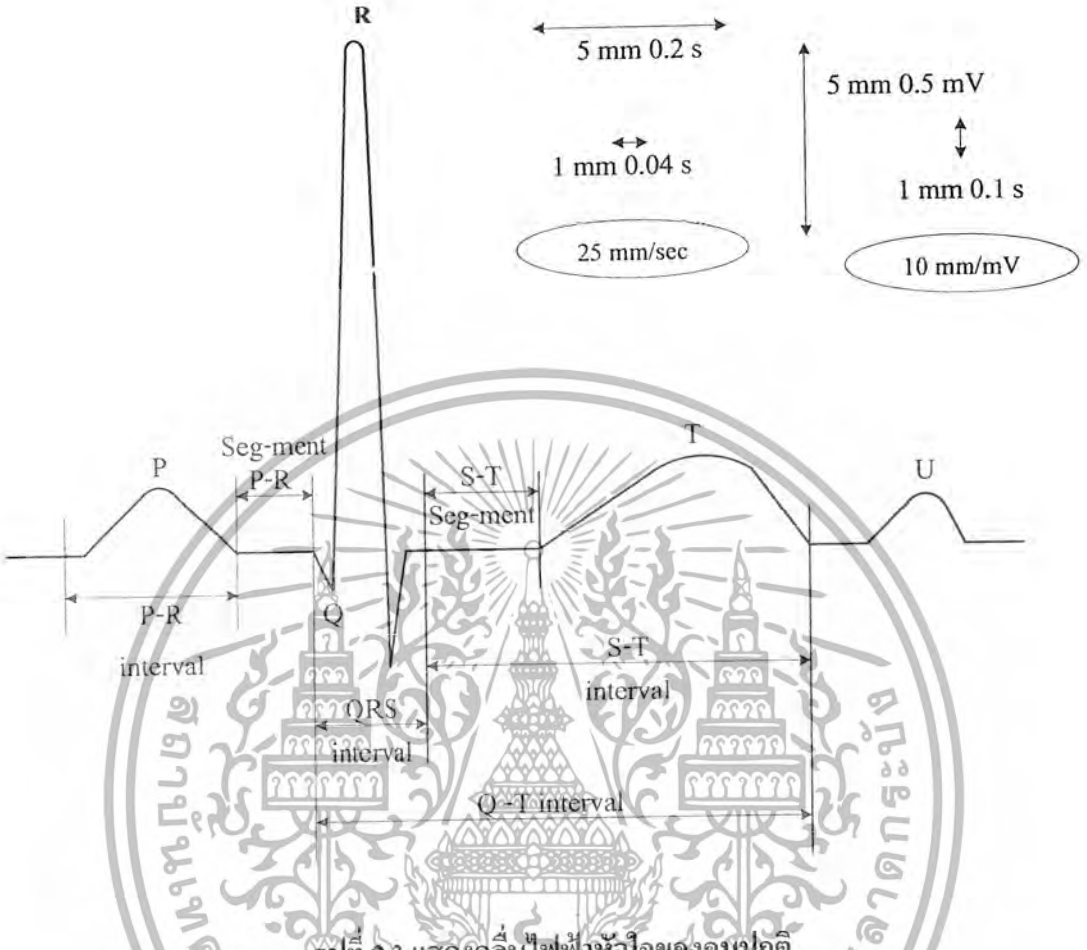


รูปที่ 2.1 แสดงถึงระบบเหนี่ยวนำไฟฟ้าของหัวใจ(Conducting system)



รูปที่ 2.2 การแผ่กระจายของ Electrical activation ในเอเทรียม และใน Ventical

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.3 แสดงคลื่นไฟฟ้าหัวใจของคนปกติ

บนรูปคลื่น ECG จะมีสัญญาณไฟฟ้าร่างกายชนิดอื่นๆ ปะปนมาด้วยเช่นสัญญาณ ECG จากสมอง สัญญาณ EMG จากกล้ามเนื้อ รวมทั้ง Motion Artifact อย่างไร ก็ดีทางเทคนิคการแยกสัญญาณ ECG ออกมาจากสัญญาณเหล่านี้ได้ไม่ยากเนื่องจากคลื่นไฟฟ้าหัวใจมีความถี่สเปกตรัมในช่วง 3-40 Hz อุปกรณ์แสดงรูปคลื่น ECG ทางการแพทย์จะมีแบนวิดท์ของการตอบสนองความถี่สำหรับการประยุกต์ใช้งานที่แตกต่างกันแบบที่ใช้งานในคลีนิค ที่ใช้สำหรับบันทึกมาตรฐาน 12 Lead EGC คือ 0.05-100Hz ในกรณีที่คนไข้มีอาการทรุดหนัก แบนวิดท์ของเครื่องวัดจะกำหนดไว้ที่ 0.5-50Hz สำหรับการวัดของอัตราการเต้นของหัวใจที่ใช้ทดสอบ QRS Complex จะตัดสัญญาณรบกวน P-wave และ T-wave ออกจาก ECG

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.3 การวัดคลื่นไฟฟ้าหัวใจ

การวัดคลื่นไฟฟ้าหัวใจสามารถทำได้ 2 รูปแบบคือ การวัดแบบเวกเตอร์คาร์ดิโอกราฟ (Vectorcardiograph) และการวัดแบบอิเล็กโตรคาร์ดิโอกราฟ (Electrocardiograph) ซึ่งสามารถอธิบายได้ดังนี้

การวัดแบบคาร์ดิโอกราฟ คือ การวัดการเปลี่ยนแปลงขนาดของเวกเตอร์ของความต่างศักย์เกิดขึ้น บนแกนหนึ่งเทียบอีกแกนหนึ่ง โดยพิจารณาจาก 3 แกน ที่ตั้งฉากกันมีอยู่ด้วยกัน 3 ระนาบ คือ ระนาบที่มองทางด้านหน้า ด้านซ้าย และ ด้านบน วิธีนี้ต้องใช้แพทย์ผู้เชี่ยวชาญในการวินิจฉัย

การวัดแบบอิเล็กโตรคาร์ดิโอกราฟ คือ การวัดการเปลี่ยนแปลงขนาดของเวกเตอร์ของความต่างศักย์ที่เกิดขึ้นในแนวแกนใดๆ เทียบกับเวลา สามารถวินิจฉัยได้ง่าย

การวัดคลื่นหัวใจแบบอิเล็กโตรคาร์ดิโอกราฟมีเพื่อการวินิจฉัยระบบการทำงานของหัวใจ สามารถแบ่งตามวัตถุประสงค์ของการวัดได้2 ประเภทคือ เพื่อการวินิจฉัยคนไข้ข้างเตียงแบบมาตรฐาน(Standard Clinical ECG) และการวัดเพื่อการมอนิเตอร์ (Monitoring ECG)

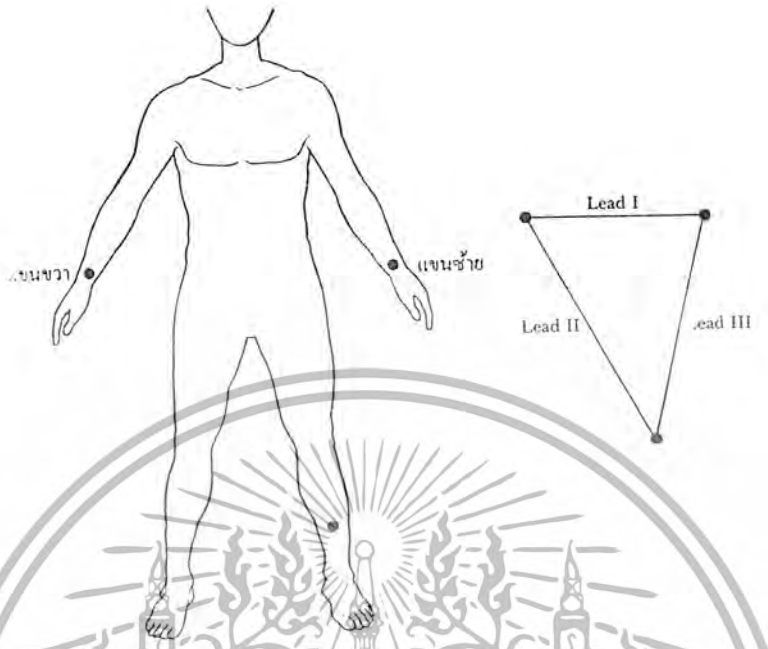
2.3.1 การวัดเพื่อวินิจฉัยคนไข้ข้างเตียงแบบมาตรฐาน

การวัดเพื่อวินิจฉัยคนไข้ข้างเตียงแบบมาตรฐานนั้น เป็นการวัดคลื่นไฟฟ้าหัวใจโดยตำแหน่งที่เป็นการวัดกำหนดไว้เป็นมาตรฐานแล้ว วิธีการวัดเพื่อวินิจฉัยคนไข้ข้างเตียงสามารถแบ่งออกได้เป็น 3 วิธี คือ วิธีการวัดแบบStandard Limb Lead วิธีการวัดแบบ Augment Limb Lead และวิธีการวัดแบบ Unipolar Chest Lead ซึ่งสามารถอธิบายได้ดังนี้

2.3.1.1 วิธีการวัดแบบStandard Lead หรือ Bipolar lead หรือ limb lead เกิดจากการวาง electrode ไว้บนแขนและขา คือ electrode อันหนึ่งอยู่บนแขนขวาอีกอันหนึ่งอยู่บนแขนซ้าย และอีกอันอยู่บนขาซ้าย หรือเท้าซ้าย แต่เรามักวางไว้บนขาซ้ายเพราะสะดวกกว่า และคลื่นไฟฟ้าที่ได้ก็จะมีรูปร่างเหมือนกันไม่ว่าเราจะวางไว้บนเท้าซ้ายหรือ ขาซ้ายคลื่นไฟฟ้าที่ได้นี้เป็นการวัดความต่างศักย์ระหว่างขั้วทั้งสองของ electrode ที่วางไว้ตามจุดต่างๆ จากการทำเช่นนี้ทำให้เราได้คลื่นไฟฟ้า 3 lead ด้วยกันคือ

1. **Lead I** ได้จากการวัดความต่างศักย์ระหว่างแขนขวาและแขนซ้าย
2. **Lead II** ได้จากการวัดความต่างศักย์ระหว่างแขนขวาและขาซ้าย
3. **Lead III** ได้จากการวัดความต่างศักย์ระหว่างแขนซ้ายและขาซ้าย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



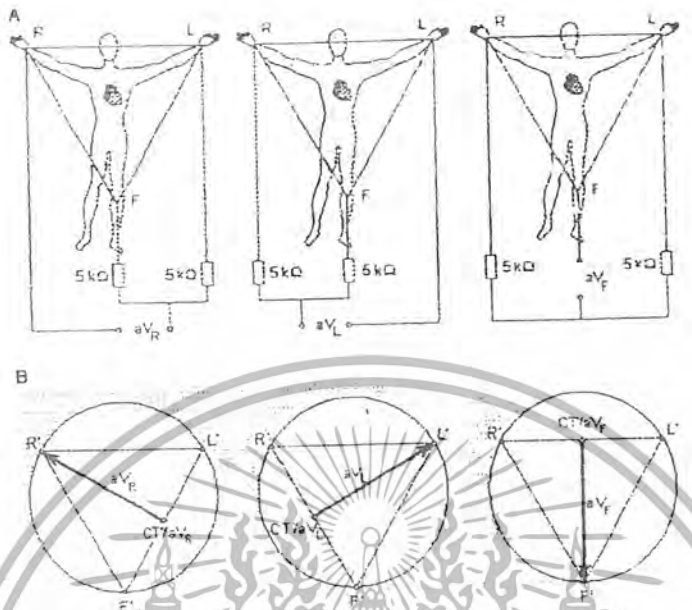
รูปที่ 2.4 การวาง electrode ไว้บนแขนและขาทำให้ได้Lead I, IIและIII

จะเห็นได้ว่าstandard lead นี้เป็นการวัดความต่างศักย์ระหว่างจุด 2 จุดด้วยกันและelectrode ที่จุดหนึ่งจะเป็นขั้วบวก electrode ที่อีกจุดหนึ่งจะเป็นขั้วลบ ดังนั้นเราจึงอาจเรียก lead ที่ได้จากการวาง electrode เช่นนี้ว่า bipolar lead แต่เนื่องจากelectrode ทั้ง 2 ขั้วนี้จะต้องวางอยู่บนแขน 2 ข้าง หรือแขนกับ ขา เราจึงอาจเรียก lead ที่ได้ว่าเป็น limb lead (รูปที่10) บางคนนิยมเรียกรวมไปเลยว่าเป็นbipolar limb lead

2.3.1.2 วิธีการวัดแบบ Augment Limb Lead

วิธีการวัดแบบ Augment Limb Lead เป็นวิธีการวัดคลื่นหัวใจที่ประกอบด้วย Lead aVR, Lead aVLและ Lead aVF ดังรูปที่2.5 สำหรับวัดสัญญาณไฟฟ้าหัวใจแบบ Augment Limb Lead จะมีตัวต้านทานค่าR_2 ต่อที่ขั้วบวกของวงจรขยายค่าความแตกต่าง มีไว้เพื่อสมดุลย์ของความต้านทานของอินพุทของวงจรขยายค่าความแตกต่าง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.5 วิธีการวัดแบบ Augment Limb Lead

2.3.1.3 วิธีการวัดแบบ **Unipolar Chest Lead** เป็นการวัดขนาดสัญญาณไฟฟ้าหัวใจ ระหว่างตำแหน่งใดๆบนหน้าอก (ขั้ววัดบวก) เทียบกับค่าเฉลี่ยของความต่างศักย์ของตำแหน่ง RA,LAและ LL วิธีการวัดในรูป 2.6 วิธีนี้ประกอบด้วย 6Lead คือ Lead V1 ถึงV6 คือการกำหนด ตำแหน่งของขั้วบวกอยู่ในบริเวณต่างๆ บริเวณหน้าอก 6 ตำแหน่งแสดงในรูปที่2.6 และ2.7

Lead V1 วางที่ช่องระหว่างกระดูกซี่โครงช่องที่4 ทางด้านขวาติดกับขอบกระดูกหน้าอก

Lead V2 วางที่ช่องระหว่างกระดูกซี่โครงช่องที่ 4ทางด้านซ้าย ติดกับขอบกระดูกหน้าอก

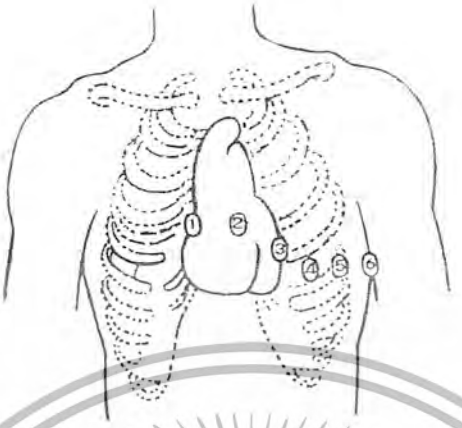
Lead V3 วางอยู่กึ่งกลางระหว่าง V2 และV4 พอดี

Lead V4 วางอยู่บนเส้นกึ่งกลางของกระดูกไหปลาร้า (mid-clavicular line) ในช่องระหว่างกระดูกซี่โครงช่องที่ 5

Lead V5 วางอยู่บนจุดซึ่งตัดกันระหว่างเส้น anterior axillary line กับเส้นขนาน (horizontal line) ที่ลากจากV4

Lead V6 วางอยู่บนจุดตัดกันระหว่างเส้น mid-axillary line กับเส้นขนานที่ลากจากV4ไป

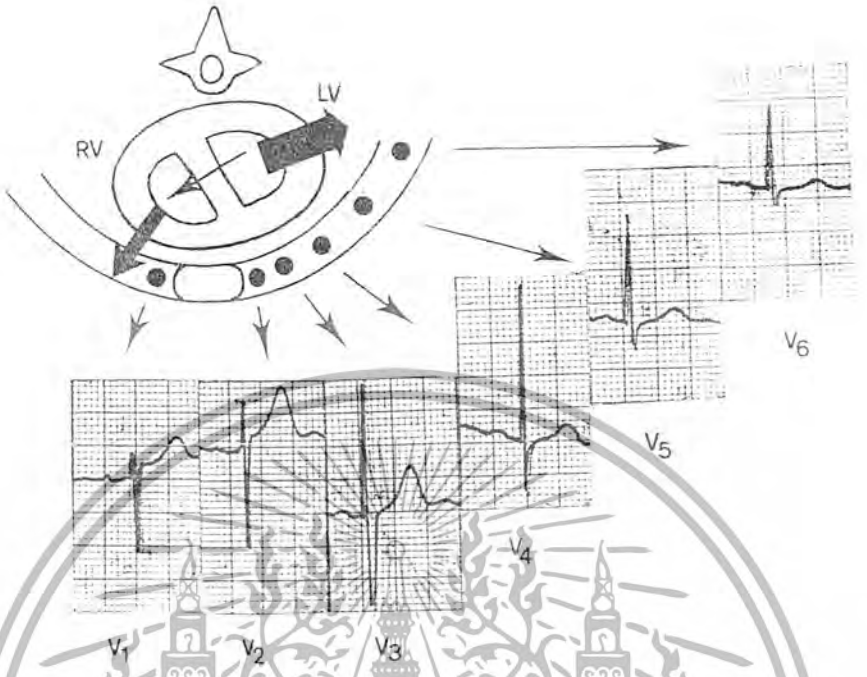
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.6 การวาง Electrode ตามตำแหน่งต่างๆบนหน้าอก ทำให้ได้ chest lead หรือ Precordial lead V_1-V_6

รูปที่ 2.7 ความสัมพันธ์ของ Electrode ที่วางบนหน้าอกกับตัวหัวใจ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.8 แสดงภาคตัดขวางของหัวใจส่วนเวนตริเคิลขวา(RV) และเวนตริเคิลซ้าย(LV)การเกิดQRS complex ที่ปกติจาก V_1-V_6

2.4 การวัดเพื่อการมอนิเตอร์

การวัดแบบมอนิเตอร์ใช้วัดสัญญาณไฟฟ้าหัวใจในขณะที่มีการเคลื่อนย้ายผู้ป่วย เพื่อตรวจสอบจังหวะและอัตราการเต้นของหัวใจ ตำแหน่งที่วัดควรเป็นตำแหน่งที่ให้สัญญาณ R แรงมากเพื่อให้อัตราการเต้นของหัวใจต่อสัญญาณรบกวน (Signal to noise Ratio: $S_{/N}$) มีค่าสูง ตำแหน่งของการวัดมอนิเตอร์จะทำโดยการติดขั้วบวกไว้ที่ตำแหน่งV1ของ Unipolar Chest Lead และติดขั้วลบไว้ใกล้ไหล่ซ้ายและติดขั้วัดใช้อ้างอิงที่ตำแหน่งใดๆบริเวณหน้าอกลักษณะของคลื่นหัวใจที่วัดได้จะใกล้เคียงกับV1ของUnipolar Chest Lead ซึ่งเป็นสัญญาณที่ใช้ในการคำนวณอัตราการเต้นของหัวใจ

2.4.1 ทฤษฎีการวัดสัญญาณไฟฟ้าจากร่างกายผู้ป่วย

ในการที่จะสร้างระบบการวัดขึ้นมานั้น จำเป็นที่จะต้องศึกษาทฤษฎีและวิธีการวัดเพื่อนำสัญญาณไฟฟ้าจากร่างกายผู้ป่วยมาทำการแสดงผลเป็นรูปคลื่นของสัญญาณ แสดงผลเป็นตัวเลข หรือวิธีอื่นๆแล้วแต่วิธีการที่จะแสดงผลออกมาซึ่งในระบบที่จัดทำขึ้นมานี้จะทำการศึกษาวิธีการวัดสัญญาณไฟฟ้าหัวใจกับการหายใจ ดังที่จะกล่าวดังต่อไปนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

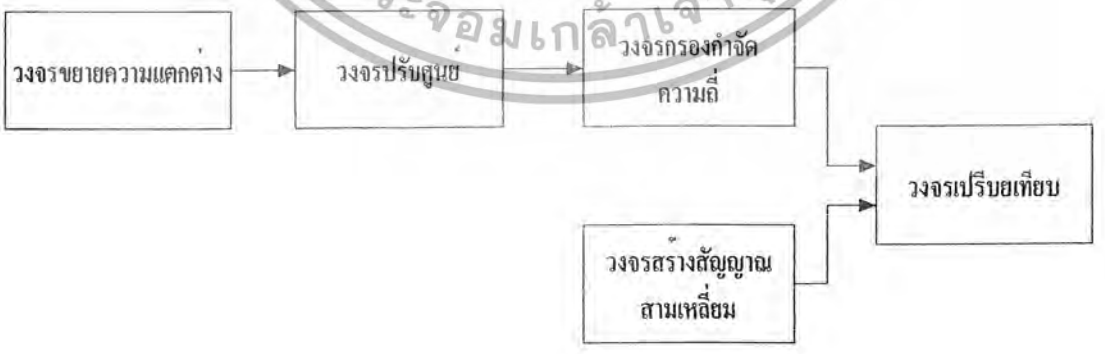
2.4.1.1 การวัดสัญญาณไฟฟ้าหัวใจ

หัวใจเปรียบเสมือนแหล่งกำเนิดไฟฟ้า ซึ่งศักดาไฟฟ้าที่เกิดขึ้นจะกระจายออกจากขั้วบวกและลบไปตามส่วนต่างๆของร่างกาย และเราสามารถวัดศักดาไฟฟ้าตกคร่อมระหว่างจุดใดๆบนผิวหนังของร่างกายได้โดยการติดอิเล็กโตรดบนผิวหนัง จากนั้นทำการขยายศักดาไฟฟ้า ลักษณะของคลื่นสัญญาณไฟฟ้าหัวใจที่วัดได้จะมีการเปลี่ยนแปลงของศักดาไฟฟ้าอยู่ในช่วง 0.5-5 mV ส่วนความถี่อยู่ในช่วง 0.5-200Hz แต่เราต้องการขนาดของสัญญาณไฟฟ้าที่ใช้ตามบ้านความถี่ 50 Hz นั้นมีโอกาสที่เหนี่ยวนำเข้ามารวมกับสัญญาณไฟฟ้าหัวใจได้ ในการออกแบบวงจรขยายสัญญาณนั้น--- ควรมีคุณสมบัติดังนี้

- ความต้านทานอินพุทมีค่าสูงกว่า 2M ohm
- กระแสไฟฟ้ารั่วไหลที่อินพุทมีค่าน้อยกว่า 1-10 A
- สัญญาณรบกวนต่ำกว่า 1 V
- ค่า CMRR (Common Mode Rejection Ratio) มีค่าสูงกว่า 60dB

2.4.1.2 ส่วนของวงจรลอย (Floating Circuit)

ในส่วนนี้จะมีการขยายสัญญาณคลื่นไฟฟ้าหัวใจ ซึ่งสัญญาณเอาท์พุทของวงจรนี้จะเป็นสัญญาณคลื่นไฟฟ้าหัวใจ เพื่อป้องกันอันตรายที่อาจเกิดจากกระแสไฟฟ้ารั่ว (Leakage Current)จึงใช้วิธีส่งสัญญาณเอาท์พุทที่ได้จากวงจรขยายส่วนแรกไปยังวงจรต่างๆที่อยู่ภาคหลัง โดยการส่งผ่านทางแสงด้วยอุปกรณ์ Opto-couple ทำให้วงจรขยายส่วนแรกและส่วนหลังแยกอิสระกันทางวงจร



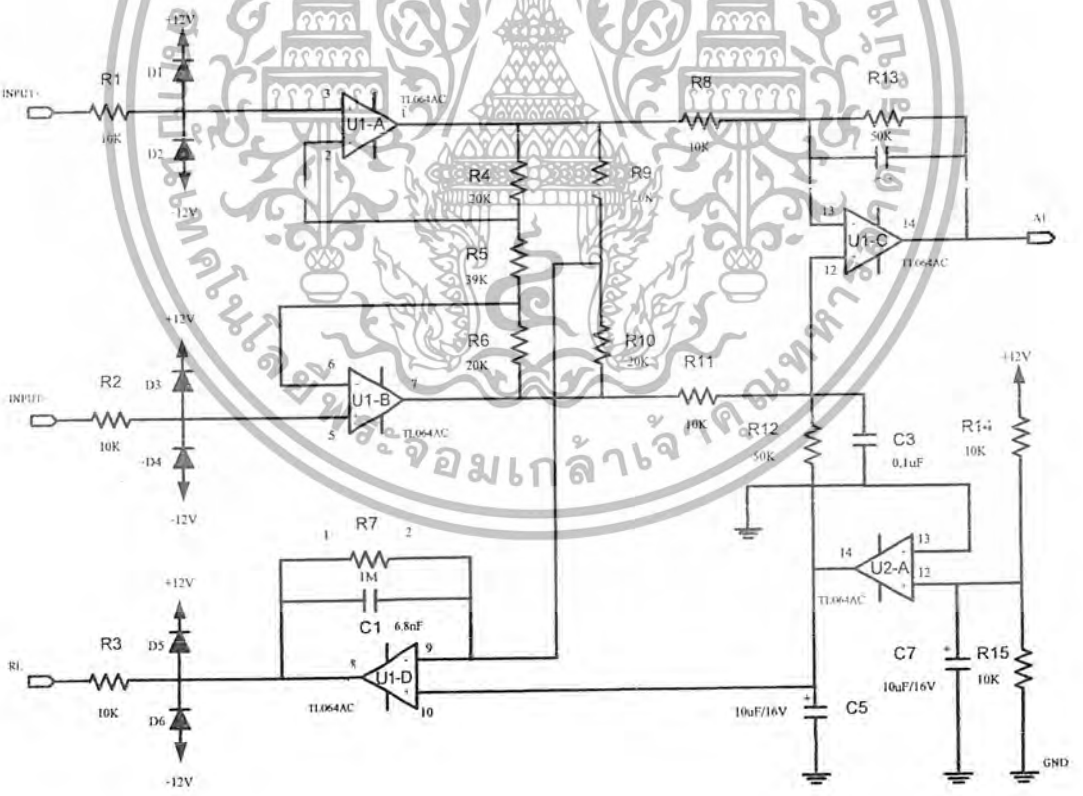
รูปที่ 2.9 แสดงบล็อกไดอะแกรมวงจรลอย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.5 วงจรวัดสัญญาณไฟฟ้าทางหัวใจ

วงจรขยายความแตกต่าง (Differential Amplifier) เป็นวงจรแรก ที่ขยายคลื่นไฟฟ้าหัวใจที่มีขนาดของสัญญาณน้อยมากประมาณ 1 mV มีการทำงานคือรับสัญญาณจากอิเลคโตรดที่ติดอยู่บนผิวหนังซึ่งมีค่าความต้านทานสูงและมีสัญญาณรบกวนจากไฟฟ้าบ้านกระแสสลับความถี่ 50Hz เหนี่ยวนำดังนั้นวงจรขยายที่จะออกแบบมาใช้นั้นควรมีคุณสมบัติดังนี้คือ

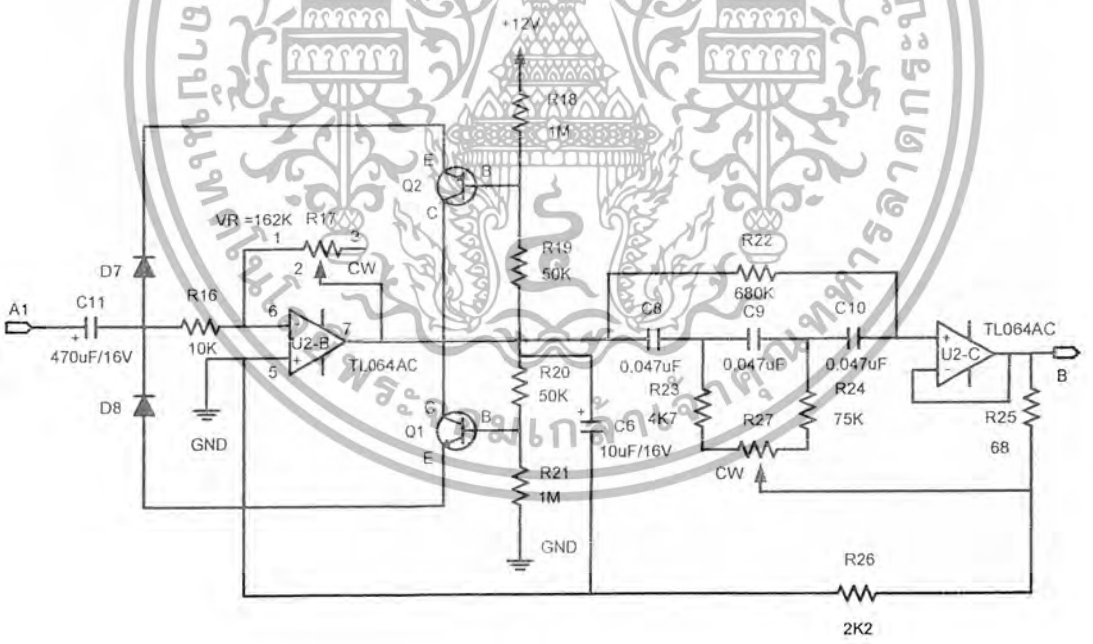
- มีอินพุทพีแดนซ์ (Input Impedances) สูงเมื่อเทียบกับความต้านทานของผิวหนัง เพื่อป้องกันการเสียสมดุลของวงจรจะมีผลเสียต่อการทำงานของวงจรขยาย
- มีค่า CMRR (Common Mode Rejection) สูงคือวงจรนั้นจะต้องมีอัตราการขยายดิฟเฟอเรนเชียล (Differential Gain) สูงและมีอัตราขยายคอมมอนโหมด (Common Mode Gain) ต่ำ ทำให้สามารถกำจัดสัญญาณรบกวน 50 Hz ที่เข้ามาในลักษณะสัญญาณคอมมอนโหมดออกไปได้



รูปที่ 2.10 วงจรขยายความแตกต่าง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

วงจรขยายความแตกต่างที่ใช้แสดงในรูปที่ 2.10 ซึ่งจะประกอบด้วยออปแอมป์ 3 ตัว คือ U1-A U1-B และU1-D สำหรับออปแอมป์ตัวที่ 4 คือ U1-D เป็นวงจรป้อนกลับแบบลบ (Negative Feedback) เพื่อใช้แทนกราวด์หรือเรียกว่า RL Driver (Right Leg Driver) จะทำหน้าที่ลดศักดาไฟฟ้าคอมมอนโหมดที่เกิดขึ้นได้ระหว่างร่างกายผู้ป่วยกับกราวด์ของวงจรลอย เมื่อสัญญาณไฟฟ้าหัวใจผ่านวงจรขยายความแตกต่างซึ่งก็จะมีศักดาไฟฟ้าออฟเซ็ทถูกขยายมาด้วย ปัญหาที่เกิดขึ้นก็คือเมื่อผู้ป่วยขยับตัวจะทำให้ความต้านทานตรงรอยสัมผัสของอิเลคโตรดกับผิวหนังเปลี่ยนแปลง และวงจรเสียสมดุลเกิดศักดาไฟฟ้าออฟเซ็ทที่เอาท์พุทของวงจรขยายความแตกต่าง คลื่นไฟฟ้าหัวใจจะลอยออกห่างจากระดับศูนย์ และกลับเข้าระดับสูงช้ามาก เนื่องจากค่าคาบเวลาคงที่ (Time Constant) ของตัวเก็บประจุกับความต้านทานอินพุทของวงจรถัดไปมีค่ามาก วงจรปรับศูนย์นั้นจะช่วยลดค่าคาบเวลาคงที่ให้น้อยลงเมื่อเกิดศักดาไฟฟ้าออฟเซ็ทถึงระดับที่กำหนดซึ่ง U2-B ในวงจรรูปที่ 2.11 จะทำหน้าที่ดังกล่าว



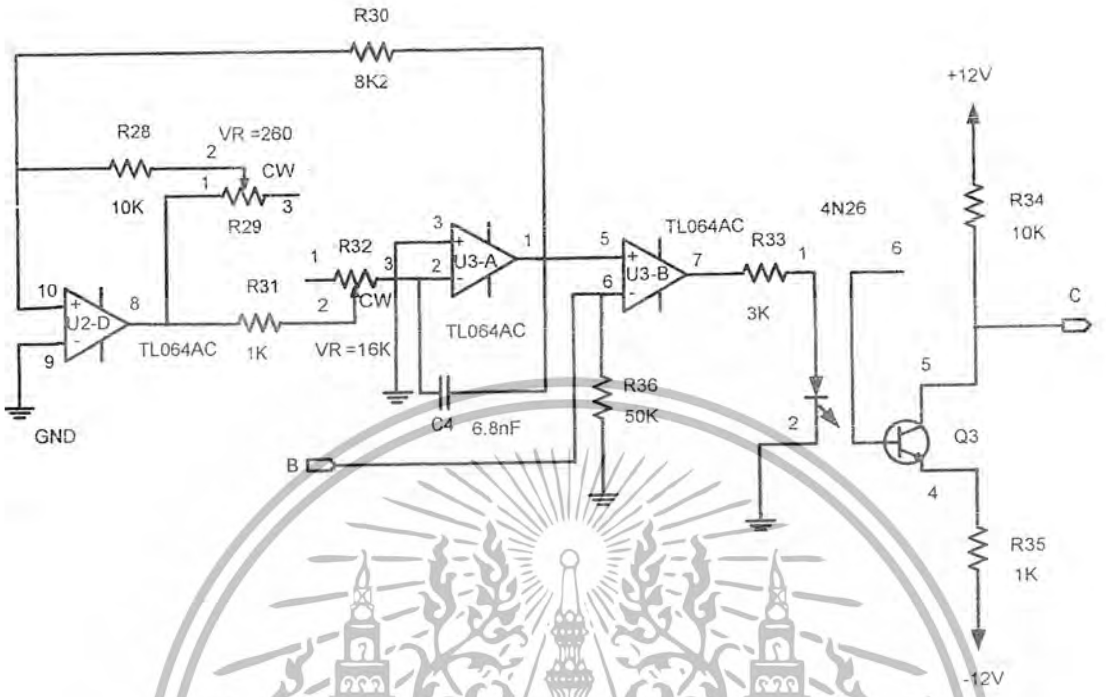
รูปที่ 2.11 วงจรปรับศูนย์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากนั้นสัญญาณจะเข้าไปในส่วนของวงจรกรองกำจัดความถี่ จากนั้นสัญญาณจะเข้าไปในส่วนของวงจรกรองกำจัดความถี่ เพื่อกำจัดสัญญาณรบกวนความถี่ 50Hz ที่ผ่านเข้ามาได้ แต่เนื่องจากคลื่นไฟฟ้าหัวใจมีความถี่ต่ำอยู่ในช่วง 0.5 ถึง 200Hz ถ้าเราใช้วงจรกรองความถี่ต่ำ (Low Pass Filter) แบบธรรมดาที่ความถี่ต่ำกว่า 50 Hz ผ่านไปได้ ก็จะทำให้สัญญาณไฟฟ้าหัวใจส่วนที่มีความถี่สูงกว่า 50 Hz ถูกกำจัดออกไป ดังนั้นเราจึงต้องใช้วงจรกรองความถี่แบบ Notch Filter ซึ่งจะยอมให้สัญญาณความถี่สูงและต่ำกว่า 50 Hzผ่านไปได้ส่วนสัญญาณรบกวน 50 Hz จะถูกกำจัดออกไปซึ่ง U2-C จะทำหน้าที่เป็น Notch Filter จากนั้นจะทำการส่งสัญญาณด้วยวงจรส่งผ่านสัญญาณแสง เพื่อทำการแยกกราวด์ระหว่างวงจรส่วนหน้าที่สัมผัสกับร่างกายกับวงจรถัดไป สำหรับป้องกันกระแสรั่วไหลจากเครื่องเข้าไปทำอันตรายต่อผู้ป่วยได้ จากข้อเสีย I-V Characteristic ของ LED ไม่เป็นเชิงเส้นต้องมีการปรับขนาดของสัญญาณ และไบอัสไฟตรงให้พอเหมาะเพื่อที่จะให้ความเป็นเชิงเส้นให้มากที่สุดจึงต้องมีการปรับแต่ง (ปรับค่า Vr) และเมื่อเปลี่ยนตัว Opto Isolater จะต้องมีการปรับแต่งใหม่ทุกครั้ง

จากข้อเสียของวงจรส่งผ่านสัญญาณแสงแบบเดิมที่กล่าวมานั้นได้มีการแก้ไขโดยการนำวงจร PWM (Pulse Width Modulator) มาใช้แสดงดังรูปที่ 2.12 ทำให้คุณสมบัติในการส่งผ่านสัญญาณไม่ขึ้นอยู่กับ I-V Characteristic ของ Opto Isolator และความไม่เป็นเชิงเส้นของ Opto Isolator จากรูปที่ 2.12 U2-D และ U3-A จะทำหน้าที่สร้างสัญญาณสามเหลี่ยม (Triangle Signal) ความถี่ 2.5KHz เพื่อเป็นสัญญาณเปรียบเทียบ (Reference Signal) เข้าที่ขาอินพุทไม่กลับเฟส (Non Inverting Input) ของ U3-B ซึ่งทำหน้าที่เปรียบเทียบสัญญาณ (Comparator) โดยสัญญาณที่ออกจากวงจรกรองความถี่ต่ำจะถูกนำมาเข้าที่ขาอินพุทกลับเฟส (Inverting Input) ของ U3-B สัญญาณที่เอาท์พุทของ U3-B จะมีลักษณะเป็นพัลส์ (Pulse Width) จะแปรเปลี่ยนไปตามระดับสัญญาณที่ขาอินพุทไม่กลับเฟสดังแสดงได้ดังรูป 2.12

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

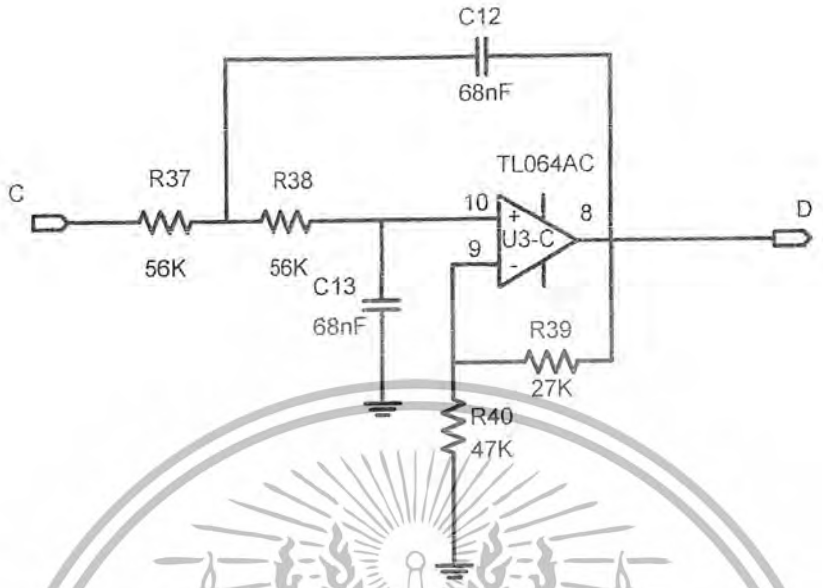


รูปที่ 2.12 แสดงการแยก Ground โดยใช้ วงจรส่งผ่านทางแสง

2.5.1 วงจรดีมอดดูเลเตอร์

หลังจากจากที่เราได้รับสัญญาณจากตัวรับสัญญาณแสง ซึ่งเป็นสัญญาณพัลส์แล้วจะต้องทำการแปลงสัญญาณคลื่นไฟฟ้าหัวใจตามเดิม โดยการใช้วงจรดีมอดดูเลเตอร์ ซึ่งวงจรที่ทำหน้าที่ดังกล่าวคือวงจรกรองความถี่ต่ำ ที่จะยอมให้ความถี่ในช่วงของสัญญาณคลื่นไฟฟ้าหัวใจผ่านออกไปได้ โดยมี U3-C ทำหน้าที่ดังกล่าว ดังแสดงในรูป2.13

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

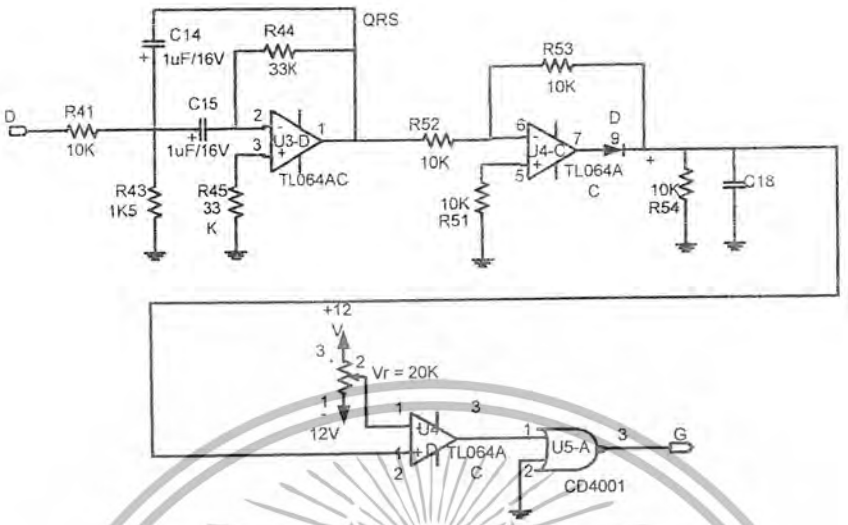


รูปที่ 2.13 วงจรดีมอดดูเลเตอร์

และเนื่องจากคลื่นไฟฟ้าหัวใจประกอบด้วยคลื่นต่างๆ คือ P,QRS และ T ในบางครั้งคลื่น T มีขนาดความสูงใกล้เคียงกับคลื่น QRS ในการหาอัตราการเต้นของหัวใจเมื่อหัวใจทำงาน 1 รอบ จะต้องมีพัลส์ 1 ลูกส่งเข้าไปในวงจรเฟสล็อคลูป และพัลส์นี้ได้จากวงจรตรวจจับยอดคลื่น (Peak Detector) ดังนั้นถ้าคลื่น QRS และคลื่น T มีขนาด ใกล้เคียงอาจทำให้ได้พัลส์ 2 ลูกใน 1 รอบการทำงานของหัวใจ เนื่องจากคลื่น QRS มีความถี่กว่าสูงกว่าคลื่น T หลายเท่า โดยมีความถี่ประมาณ 17Hz จึงสามารถใช้วงจรกรองความถี่แยกเอาคลื่น QRS ออกจากคลื่น T โดยมี U4-A ทำงานในส่วนแยกคลื่น QRS จากนั้นจะเข้าสู่วงจรสร้างสัญญาณนาฬิกาเพื่อนำไปนับจำนวนอัตราการเต้นของหัวใจ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

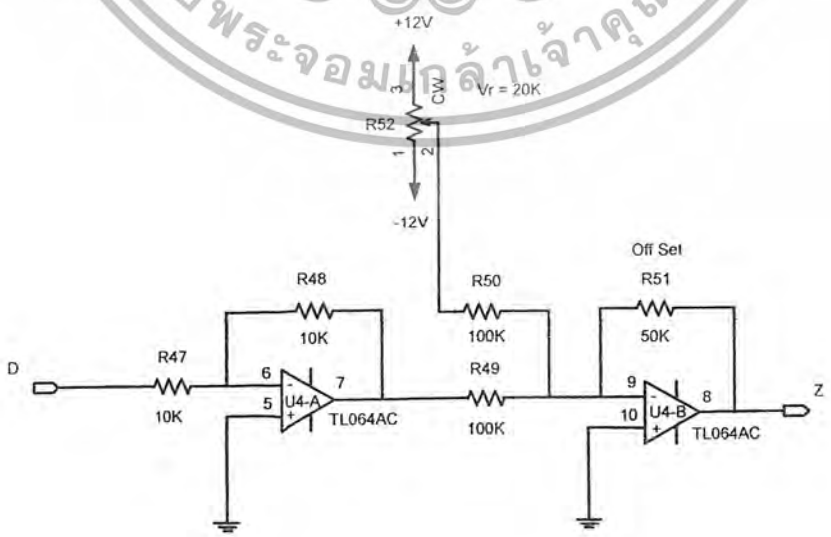
สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง



รูปที่ 2.14 วงจรสร้างสัญญาณนาฬิกา

2.5.2 วงจรยกระดับแรงดัน (Off Set)

จากคุณสมบัติของ ADC 0820 ที่รับระดับแรงดันที่ 0-5V นั้นเราจึงต้องสร้างวงจรยกระดับแรงดันขึ้นมาในเบื้องต้นนั้นเราได้สัญญาณมาจากขา U4-A (D) นำเข้าวงจร Inverting Amplifier ก่อนแล้ว U4-B วงจร Summing Amplifier กับสัญญาณไฟตรงโดยมี Variable Resistance เป็นตัวยกระดับสัญญาณไฟตรง



รูปที่ 2.15 วงจรยกระดับแรงดัน (Off Set)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

55448

บทที่ 3
หลักการเกิดภาพในจอมอนิเตอร์

3.1 ทฤษฎีพื้นฐานของแสงและสี

แสงสีโดยทั่วไปมี 2 ประเภท คือ แสงที่สายตามนุษย์มองเห็น(visible rays) และแสงที่สายตาของมนุษย์ที่ไม่สามารถมองเห็นได้ซึ่งได้แก่แสงจำพวก รังสีแกมมา(gramma-rays),รังสีเอกซ์(X-rays),รังสีอัลตร้าไวโอเลท(ultraviolet-rays) และรังสีอินฟาเรด(infrared-rays) แสงเหล่านี้ต่าง ก็เป็นพลังงานคลื่นแม่เหล็กไฟฟ้า แต่มีความถี่หรือความยาวคลื่นแตกต่างกันออกไปส่วนของคลื่นแม่เหล็กไฟฟ้าที่น่าสนใจของจอมอนิเตอร์ (monitor) ที่แสดงออกมานี้ก็คือส่วนของคลื่นแม่เหล็กไฟฟ้า ที่มนุษย์สามารถมองเห็นได้ซึ่งมีความยาวคลื่นประมาณ 380 นาโนเมตรจนถึง 780 นาโนเมตร แสงที่มองเห็นนี้จะทำให้ตาของมนุษย์ได้ความรู้สึก 2 ประการ คือ 1.ความรู้สึกว่าแสงมีความสว่างมากน้อยเพียงไร (sensation of brightness) 2. ความรู้สึกที่สามารถแยกแยะว่าเป็นสีอะไร (sensation of color)

3.2 ความสำคัญ 3 ประการของแสงที่มองเห็น

แสงที่มองเห็นจะทำให้เกิดความรู้สึกที่สำคัญ 3 ประการคือ

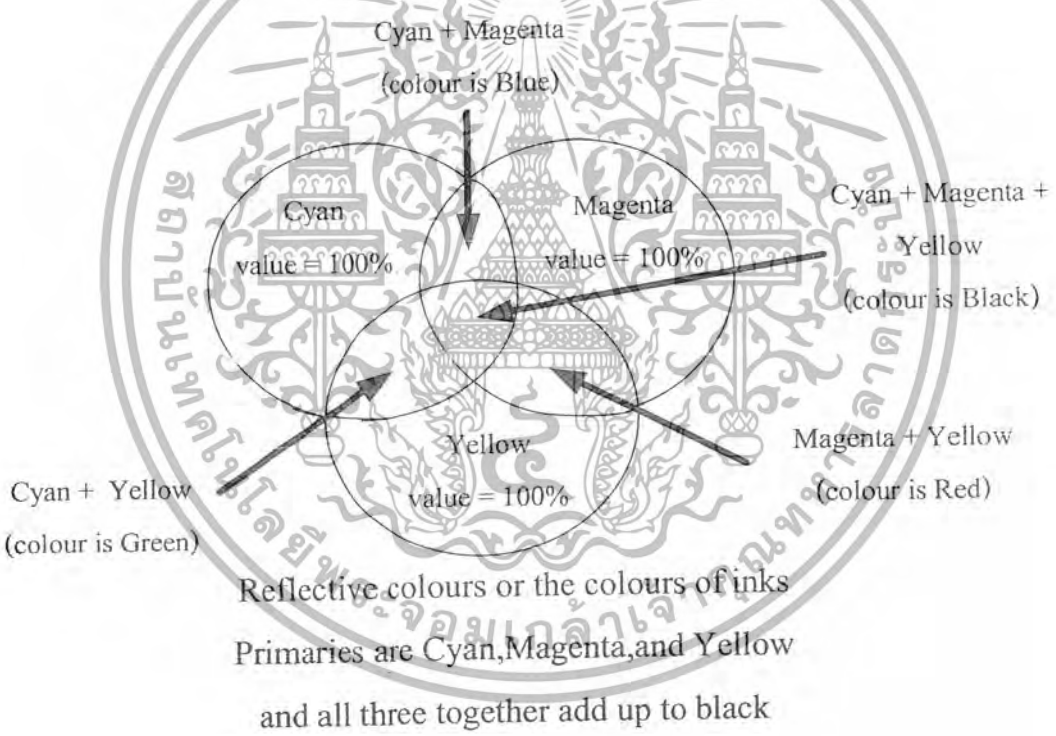
1. เกิดความรู้สึกในเรื่องของแสงสี(hue)
2. เกิดความรู้สึกในเรื่องการส่องสว่าง(brightness)
3. เกิดความรู้สึกในเรื่องแสงสีอิ่มตัว (saturation)

โดยความรู้สึกในด้านแสงสี(hue)จะทำให้ตาของมนุษย์สามารถแยกแยะออกได้ว่าแสงที่มองเห็นได้นั้นเป็นแสงสีอะไร เป็นสีแดง สีเขียวหรือสีน้ำเงิน เป็นต้น ส่วนความรู้สึกในเรื่องการส่องสว่าง(brightness)เป็นความรู้สึกที่บอกว่าแสงที่ตามองเห็นนี้มีความสว่างหรือมืด สำหรับความรู้สึกทางด้านแสงสีอิ่มตัว(saturation of chroma) จะทำให้สามารถรู้ความบริสุทธิ์ว่าแสงสีที่เห็นเป็นแสงสีที่มีชัดเจน หรือ จาง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

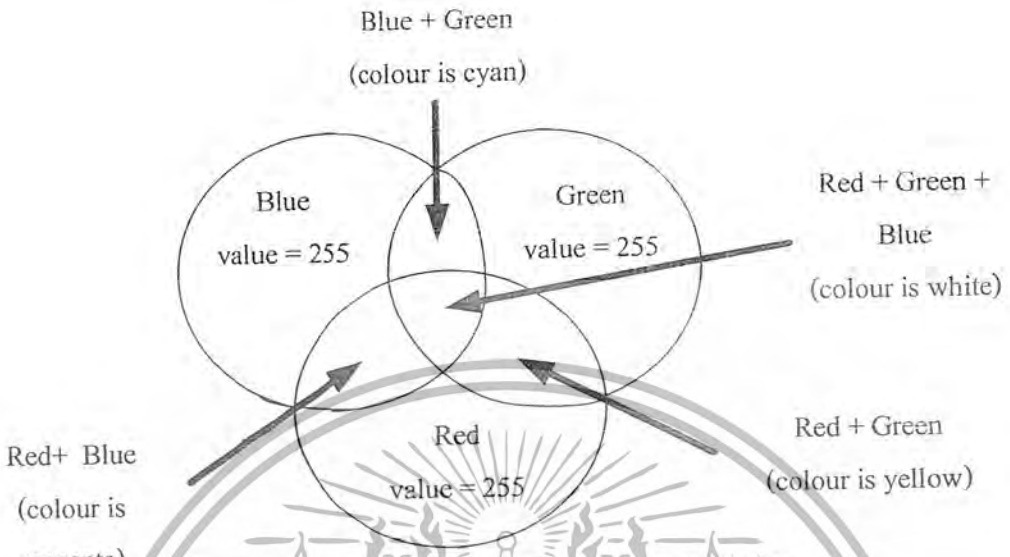
3.3 การผสมและการแยกสี

การผสมแสงสีมีลักษณะแตกต่างไปจากการผสมสีทางการวาดภาพ การผสมสีโดยทั่วไป เช่น ในการวาดเขียนหรือการพิมพ์ภาพสี จะทำให้ได้สีผสมที่มีความเข้มข้นกว่า ซึ่งเป็นลักษณะวิธีของวิธี Subtractive mixer ส่วนการผสมสีทางแสงนั้น จะทำให้แสงสีที่ผสมมีลักษณะเจือจางกว่าแม่สีเดิม อันเป็นลักษณะวิธีของ Additive mixer ดังรูปที่ 3.1 และการผสมแสงสีที่ใช้ในการสร้างภาพของจอมอนิเตอร์นั้นจะเป็นการผสมแม่สี (primary color) เพื่อทำให้เกิดแสงสีต่างๆขึ้น สีที่เป็นแม่สีนี้ต้องเป็นสีที่ไม่สามารถเกิดจาการผสมสี ซึ่งในระบบโทรทัศน์และจอมอนิเตอร์จะมีแม่สีอยู่ 3สีคือ 1.สีแดง 2.สีเขียว 3.สีน้ำเงิน



รูปที่ 3.1 การผสมสีแบบ Subtractive Mixer

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.2 การผสมสีแบบ Additive Mixer

แสงสีต่างๆที่เราสามารถมองเห็นได้นั้น เกิดจากการผสมสีทางแม่สีทั้งนั้น ซึ่งอาจเกิดจากการนำสีใดสีหนึ่งไปผสมกับสีใดสีหนึ่ง หรือ การนำสีใดสีหนึ่งไปหักจากสีใดสีหนึ่งก็ได้ ยกตัวอย่างเช่น

แสงสีเหลือง (Yellow) = แสงสีแดง (Red) + แสงสีเขียว (Green)

แสงสีเขียวน้ำเงิน (Cyan) = แสงสีเขียว (Green) + แสงสีน้ำเงิน (Blue)

แสงสีม่วง (Magenta) = แสงสีน้ำเงิน (Blue) + แสงสีแดง (Red)

แสงสีขาว (White) = แสงสีแดง (Red) + แสงสีเขียว (Green) + แสงสีน้ำเงิน (Blue)

แสงที่เกิดจากแม่สีทั้งสามสี ได้มีการทดลองหาความยาวคลื่นที่แน่นอนไว้แล้วจากคณะกรรมการระหว่างประเทศในเรื่องของการส่องสว่าง(Internationnal Committee of Illumination หรือ Commission Internation de I'Eclairage หรือเรียกสั้นๆว่า CIE) ได้กำหนดความยาวคลื่นของแสงจากแม่สีทั้งสามไว้ในตาราง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รายการ	ความยาวคลื่น(นาโนเมตร)	หมายเหตุ
สีน้ำเงิน	435.8	เป็นส่วนหนึ่งของสเปกตรัมเชิงเส้นของปรอท
สีเขียว	546.1	เป็นส่วนหนึ่งของสเปกตรัมเชิงเส้นของปรอท
สีแดง	700	การประมาณค่าออกเป็นตัวเลขหยาบๆ

3.1 ตารางแสดงความยาวคลื่นของแม่สีทั้งสามในระบบแสงสีแดง สีเขียวและแสงสีน้ำเงิน(RGB)

3.4 องค์ประกอบของภาพ

หากเราตัดภาพจากหนังสือพิมพ์ภาพหนึ่งแล้วขยายด้วยกล้องหรือแว่นขยาย จะพบว่าภาพที่ได้นั้นมีองค์ประกอบมาจากจุดสีขาวและจุดสีดำมากมาย มาเรียงประกอบกันขึ้นเป็นภาพ จุดเหล่านี้เองที่เรียกว่าองค์ประกอบของภาพ(Picture Element) หรือ PIXELนั่นเอง

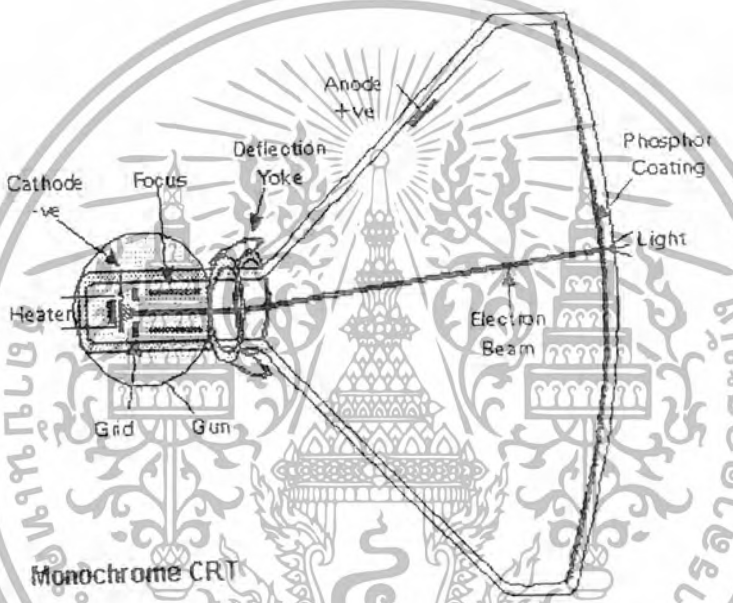
ในทำนองเดียวกันภาพที่ปรากฏทางด้านจอมอนิเตอร์ก็นำหลักการข้างต้นนี้ ภาพที่เกิดมาจากเส้นขวางเล็กๆวางเรียงกันในแนวนอนจำนวนมาก บนจอภาพแต่ละเส้นนั้นมีส่วนที่ดำสนิทส่วนที่จางและส่วนที่สว่างวางเรียงกันอยู่ เส้นเหล่านี้ได้มาจากการกวาดลำเส้นของแสง(scan)ความแตกต่างกันบนเส้นกวาดลำแสง หรือ เส้นสแกนเหล่านี้เองเราจัดว่าเป็นองค์ประกอบของภาพ

จอภาพทำงานโดยการแสดงภาพ ซึ่งอาจเป็นภาพกราฟิกหรือตัวอักษร ซึ่งเกิดจากการประมวลผลของการ์ดวีจีเอจอภาพแบ่งเป็น2ประเภท คือ จอภาพสีเดียวหรือจอภาพโมโนโครม (Monochrome) และจอสี (Color Monitor) ปัจจุบันจอภาพสีเดียวนั้นไม่เป็นที่นิยมใช้กับคอมพิวเตอร์ หากจะมีใช้ก็เฉพาะงานเฉพาะอย่างเท่านั้น ส่วนที่นิยมใช้ก็คือจอสี โดยแบ่งได้อีกเป็น 3 ประเภท คือจอสีวีจีเอ (VGA = Video Graphics Array) และจอสีSuper VGA (SVGA=Super Video Graphics Array) และจอสีLCD (Liquid Crystal Display)ซึ่งประเภทหลังนี้มีราคาแพงมาก จอภาพที่ได้รับความนิยมในปัจจุบันคือ จอSVGA เนื่องจากมีราคาไม่แพงมากนัก และเหมาะกับ Application ที่ออกแบบให้มีความสามารถแสดงภาพกราฟิก นอกจากนี้ Application ประเภทมัลติมีเดียหรือเกมต่างๆต่างก็ต้องการจอภาพที่มีความละเอียดสูง(High Resolution)สามารถแสดงสีได้หลายๆสี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.5 การทำงานของจอภาพ

จอภาพมีหลักการทำงานแบบเดียวกับจอโทรทัศน์โดยจะมีกระแสไฟฟ้าแรงสูง (High Voltage) คอยกระตุ้นให้อิเล็กตรอนภายในหลอดภาพแตกตัว อิเล็กตรอนดังกล่าวจะทำให้เกิดลำแสงอิเล็กตรอนจะเกิดการเรื่องแสงและปรากฏเป็นจุดสีต่างๆ (RGB Color) ซึ่งรวมเป็นภาพบนจอภาพนั่นเอง



รูปที่ 3.3 โครงสร้างของหลอด Cathode Ray Tube

เพื่อความเข้าใจในการกำเนิดสัญญาณภาพ อันดับแรกเราต้องเข้าใจกระบวนการสร้างภาพบนผิวหลอดภาพก่อน สัญญาณภาพ VGA(VGA video signal)มีส่วนประกอบของสัญญาณที่สำคัญ 5สัญญาณ จะมี2 สัญญาณที่เป็นระดับสัญญาณแบบ TTL Logic มีสัญญาณ Horizontal sync(Hor sync) และสัญญาณVertical sync (Ver sync)สัญญาณทั้ง2 จะถูกใช้ในการเข้าจังหวะของภาพส่วนสัญญาณอีก 3 สัญญาณที่เหลือจะเป็นสัญญาณสีของแม่สีที่ลักษณะของสัญญาณจะเป็นระดับสัญญาณทางอนาล็อกมีขนาดตั้งแต่ 0.7-1 V_{p-p}หรือเป็นสัญญาณควบคุมการเกิดสีเรียกว่าสัญญาณ RGB ประกอบด้วยสัญญาณ สีแดง(red) สัญญาณสีเขียว(green)และสัญญาณสีน้ำเงิน(blue) ซึ่งการเปลี่ยนระดับอนาล็อกของสัญญาณทั้ง3จะเป็นตัวกำหนดการเกิดสีบนผิวจอภาพ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

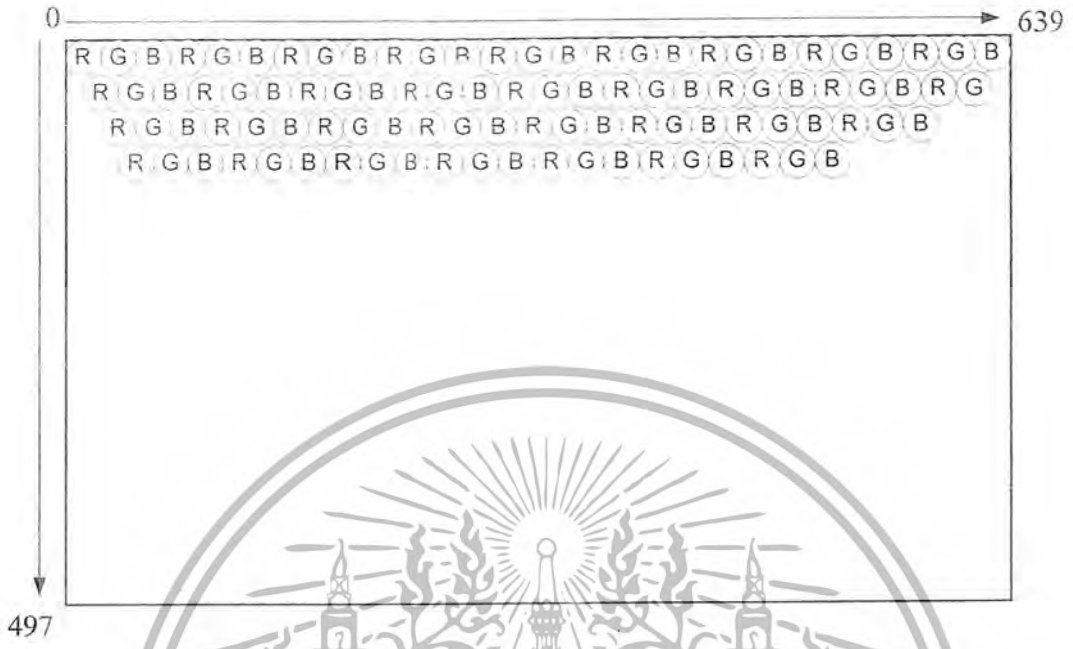
3.6 เทคโนโลยีการสร้างภาพ

อุปกรณ์หลักของการสร้างภาพ คือ หลอดภาพ Cathode Ray Tube (CRT)การทำงานเริ่มจากการยิงลำแสงอิเล็กตรอนจากก้นของหลอดภาพไปสู่ผิวจอภาพเป็นการกวาด(scan) ภาพตามแนวนอน(horizontal) โดยเริ่มจากส่วนบนของหลอดภาพเริ่ม horizontal line เส้นที่1 โดยการหักเหของลำอิเล็กตรอนนั้นจะเกิดจาก Deflection Yoke เป็นลักษณะขดลวดที่อยู่ทางก้นหลอด อาศัยสนามแม่เหล็กหรือสนามไฟฟ้าสถิตควบคุมการเบี่ยงเบนลำอิเล็กตรอนเรียกว่า การกวาดภาพ(scan) ส่วนการเกิดสีต่างๆนั้น จะเป็นการควบคุมขนาดของสัญญาณRGB ทั้ง3 สัญญาณเพื่อไปควบคุมความเข้มของลำอิเล็กตรอนที่วิ่งไปกระทบกับผิวของจอภาพซึ่งเคลือบด้วยสารฟอสเฟอร์อยู่ก่อนแล้ว สารฟอสเฟอร์นั้นเมื่อได้รับพลังงานจากการชนของอิเล็กตรอนจะทำให้เกิดการเรืองแสง ซึ่งการเรืองแสงของจอและลักษณะของจอจะแสดงดังรูปที่3.4

รูปที่ 3.4 ลักษณะการยิงอิเล็กตรอนและโครงสร้างของผิวมอนิเตอร์

รูปแบบโดยทั่วไปของจอ VGA จะกำหนดให้ขนาดมาตรฐานของจำนวน Picture Element หรือ Pixel มีขนาดเป็น 480 x 640 Pixel โดยได้มาจากจำนวนแถวของแนวนอน(Horizontal Row) คูณกับจำนวนหลักของแนวตั้ง(Vertical Column) ซึ่งสามารถอธิบายดังรูปที่ 3.5

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.5 แสดงจำนวนของ Pixel ขนาด 480*640 ของจอมอนิเตอร์

ในการกวาดสัญญาณแต่ละ Pixel นั้น สัญญาณ RGB จะเปลี่ยนค่าทำให้เกิดสีในแต่ละ Pixel มีค่าต่างกันเมื่อมองดูแล้วจะเกิดเป็นภาพออกมา

การกวาดสัญญาณภาพตามเส้น Horizontal ตั้งแต่หลักที่ 0 จนถึง หลักที่ 640 จะทำให้เกิดสัญญาณ Hor Sync จำนวน 1 เส้นเพื่อเป็นการส่งสัญญาณให้จอมอนิเตอร์ทำการสะบัดกลับของลำอิเล็กตรอนไปเริ่มที่ตำแหน่ง 0ใหม่ แต่เป็นการเริ่มต้นแถวใหม่ทางด้าน Vertical เป็นแถวที่ 1 โดยการสะบัดกลับมาเริ่มต้นการกวาดภาพใหม่จะทำจนกระทั่งตำแหน่ง (Horizontal, Vertical)เท่ากับ (480,640) แล้วจอมอนิเตอร์จะต้องได้รับสัญญาณ Ver Sync 1 ครั้ง แล้วพบว่าการสะบัดจะเริ่มที่ตำแหน่ง(0,0)ใหม่ในระหว่างที่มีการสะบัดจะมีสัญญาณ Blanking เพื่อให้จอภาพมืดไม่ให้เห็นเส้นสะบัดและเป็นเช่นนี้ไปจนเกิดเป็นภาพ ซึ่งแต่ละ 1 วินาที จอมอนิเตอร์ VGA จะเกิดภาพ 60ครั้งต่อวินาที(Refresh rate) และไม่ควรต่ำกว่า 30ครั้งต่อวินาทีเนื่องจากต้องการทำให้ตาของคนมองไม่เห็นการกระพริบใน แต่ละภาพ (frame) ซึ่งคาบเวลาที่ใช้ในการเกิดแต่ละ Pixel คือ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากการทำงานใน 1 Frame จะได้จำนวน Pixel = 480 x 640

= 307200 Pixel

ใน 1วินาทีจะเกิดFrame ทั้งหมด =307200 x 60

=1843200 Pixel

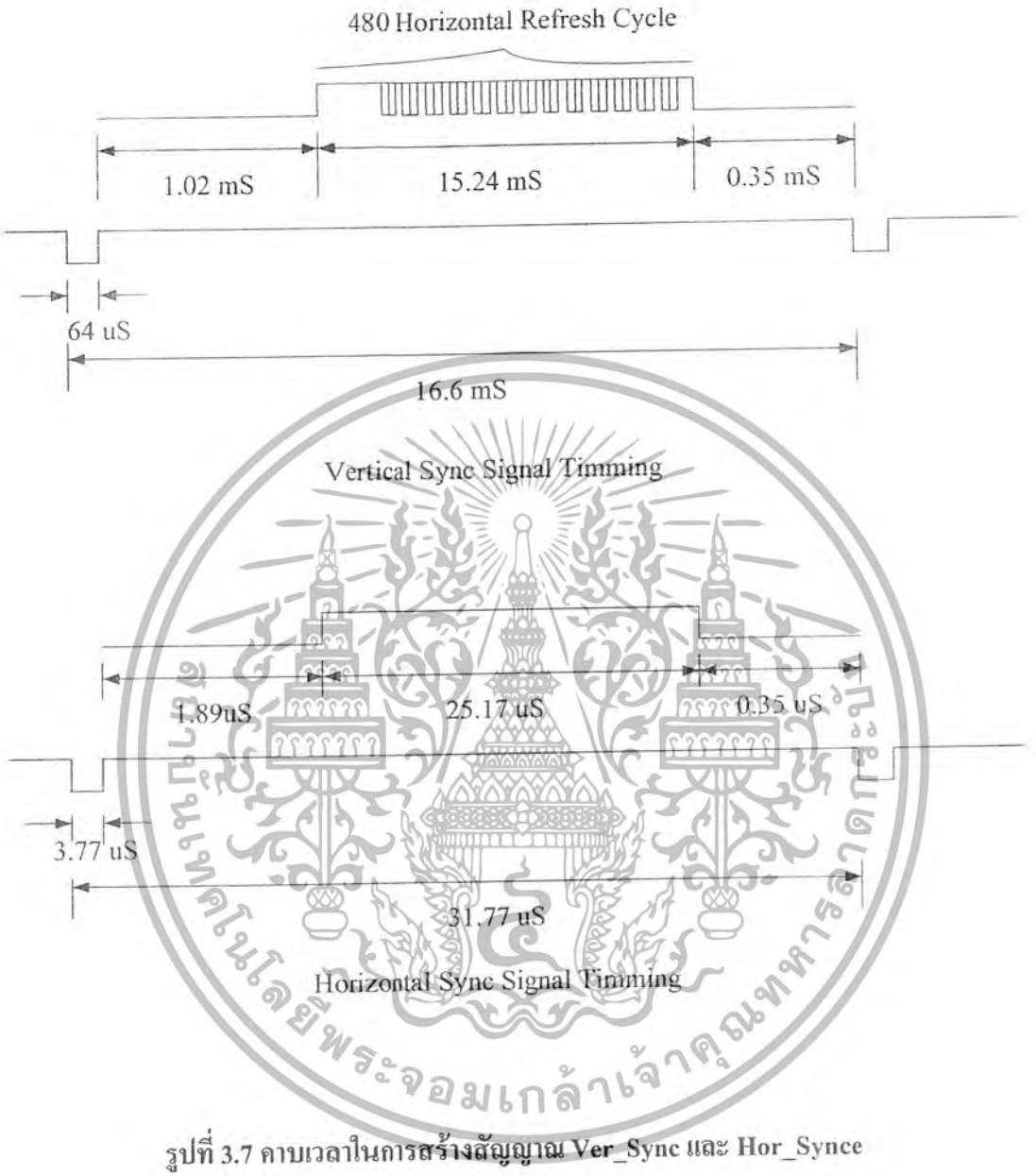
ดังนั้นใน 1 Pixel จะใช้เวลาในการเกิด =1/1843200

=54.253 nS

แต่เราได้ประมาณให้มีคาบเวลาเป็น 40 nS ซึ่งจะได้ความถี่ที่ใช้ประมวลผลเป็น 25MHz

รูปที่ 3.6 รูปแบบการกวาดภาพ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.7 คาบเวลาในการสร้างสัญญาณ Ver_Sync และ Hor_Synce

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4

อุปกรณ์ FPGA และการเขียนโปรแกรม

ความก้าวหน้าของเทคโนโลยีอิเล็กทรอนิกส์ในปัจจุบัน ทำให้เกิดการพัฒนาความสามารถของอุปกรณ์อิเล็กทรอนิกส์ต่างๆมากมาย ซึ่งทำให้อุปกรณ์อิเล็กทรอนิกส์ มีขนาดที่เล็กลง ใช้งานง่ายขึ้น และราคาถูกลง ทำให้เกิดการลดค่าใช้จ่ายและใช้พลังงานน้อยลง ในขณะเดียวกันก็ได้มีการเพิ่มประสิทธิภาพและระดับความเชื่อถือได้ของวงจรรวมที่สูงขึ้น เห็นได้ชัดจากเทคโนโลยีไมโครเซสเซอร์และหน่วยความจำในปัจจุบัน ในการพัฒนาวงจรอิเล็กทรอนิกส์เราจะพิจารณาออกเป็น 2 ส่วน

ส่วนที่ 1 เป็นการพัฒนาทางด้านวงจรอนาลอก ซึ่งการพัฒนาในส่วนนี้เมื่อเราออกแบบวงจรออกมาได้แล้ว จำเป็นจะต้องนำไป Fabrication ซึ่งจะต้องทราบเทคโนโลยีที่จะใช้ในการสร้าง มีค่าใช้จ่ายสูงและใช้ระยะเวลานาน

ส่วนที่ 2 เป็นการพัฒนาทางด้านวงจรดิจิตอล ในปัจจุบันนี้เทคโนโลยีการผลิตอุปกรณ์อิเล็กทรอนิกส์ ได้มีความก้าวหน้ามากขึ้นซึ่งเมื่อเราออกแบบวงจรทางดิจิตอลแล้วเราสามารถที่จะ Implement ลงบนชิพไอซีได้เลยซึ่งจะกล่าวต่อไป

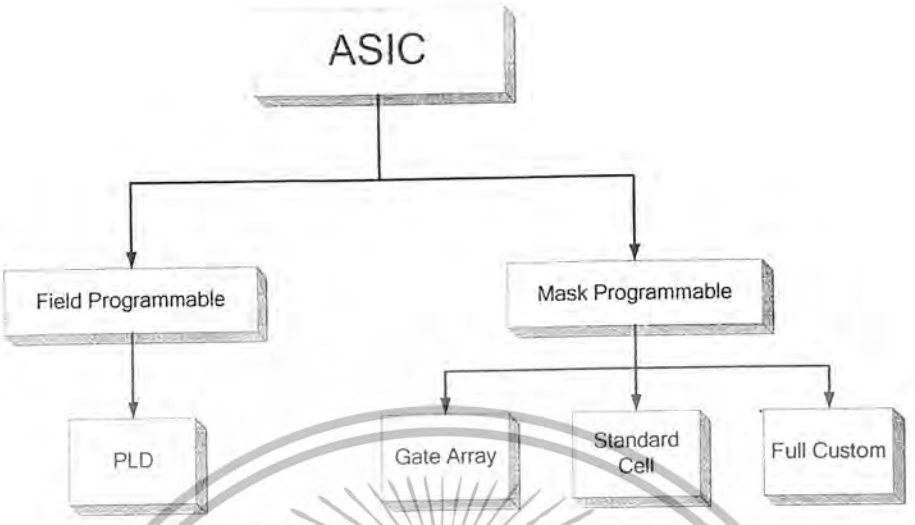
ในการพัฒนาวงจรดิจิตอลแบบเก่า ซึ่งใช้ดิจิตอลไอซีมาต่อวงจร จะเห็นว่ามีความยุ่งยากมากทั้งขนาดและการทดสอบความถูกต้องในการทำงาน ทำให้เกิดช่องว่างระหว่างไอซีมาตรฐานและวงจรรวมมากยิ่งขึ้น ในการพัฒนาวงจรรวมทางดิจิตอลได้เพิ่มความหนาแน่นและจำนวนฟังก์ชั่นลอจิก (Function Logic) ที่เหมาะสม นักออกแบบอุปกรณ์ทางด้านดิจิตอลได้พิจารณาถึงการผลิตให้มีปริมาณมากๆ และในการผลิตวงจรรวมเฉพาะงาน (ASIC: Application Specific Integrated Circuit) จะแบ่งตามการสร้างออกเป็น 2 กลุ่ม คือ

4.1. Mask Programmable

4.2. Field Programmable

ดังแสดงในรูปที่ 4.1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.1 แสดงแผนผังการแบ่งกลุ่มของวงจรรวม ASIC

4.1 Mask Programmable

การใช้งานวงจรรวมเฉพาะงาน ASIC ในเชิงพาณิชย์ จำเป็นต้องใช้วงจรรวมเฉพาะงาน ASIC แบบ Mask programmable เนื่องจากต้นทุนต่อหนึ่งตัวจะต่ำกว่าวงจรแบบ Field programmable ASIC. ในกรณีที่ปริมาณการผลิตสูงนับพันนับหมื่นตัวขึ้นไป ตัวอย่างเช่น วงจร CPLD ตัวหนึ่งอาจสูงถึงหนึ่งพันบาท ในขณะที่ถ้าผลิตวงจรรวมที่มีคุณสมบัติเหมือนกันทุกประการโดยใช้ Mask programmable แล้ว ราคาตัวหนึ่งจะลดลงเหลือเพียงไม่ถึงหนึ่งร้อยบาท การใช้วงจรรวมแบบ Mask programmable จึงมีบทบาทสำคัญในการผลิตสินค้าอิเล็กทรอนิกส์ในเชิงพาณิชย์ในปัจจุบัน

วงจรรวมเฉพาะงานประเภทนี้ หลังจากผู้ใช้ออกแบบวงจรและตรวจสอบการทำงานจนเป็นที่น่าพอใจแล้ว ต้องส่งให้ผู้ผลิตทำการเจือสาร ไม่สามารถโปรแกรมได้ด้วยตนเองเหมือนกับวงจรรวมเฉพาะงานแบบ Field programmable ช่วงเวลาการผลิตออกใช้งานจึงใช้เวลานับเดือนและมีค่าใช้จ่ายเบื้องต้นในการเจือสารสูง วงจรรวมเฉพาะงานแบบ Mask Programmable ASIC ในปัจจุบันได้แก่ เกตอาเรย์ , เซลล์มาตรฐานและฟูลัสตัม (full custom)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

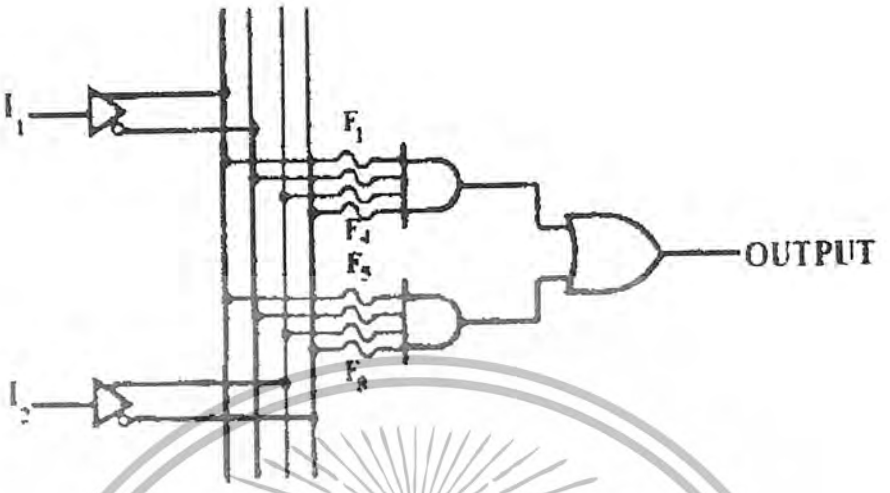
4.2 Field Programmable

อุปกรณ์วงจรรวมเฉพาะงาน ASIC แบบ field programmable มีอยู่มากมายหลายชนิด แต่มีลักษณะการสร้างหรือกำหนดการทำงานของวงจรที่เหมือนกัน กล่าวคือ ผู้ใช้งานสามารถออกแบบและสร้างวงจรที่ต้องการใช้ลงในตัวอุปกรณ์ได้เองโดยไม่ต้องไปโรงงานเพื่อผลิต โดยเฉพาะอย่างยิ่งในปัจจุบันนี้มีเครื่องมือที่ใช้ช่วยในการออกแบบ และสร้างวงจรร่วมกับไมโครคอมพิวเตอร์ที่มีความสามารถสูงในการพัฒนาตั้งแต่ขั้นการออกแบบ การจำลองการทำงาน จนถึงจัดสร้างวงจรลงในอุปกรณ์ รวมทั้งอุปกรณ์ Field Programmable เหล่านี้สามารถหาซื้อได้ง่ายทำให้การสร้างวงจรอิเล็กทรอนิกส์จนถึงระบบไมโครโพรเซสเซอร์ทันมาใช้อุปกรณ์จำพวกนี้ เป็นอุปกรณ์ประกอบในวงจรแทนอุปกรณ์ย่อยๆ แยกชิ้น (Discrete component)

4.2.1 พีแอลดี (PLD: Programmable Logic Device)

ภายในอุปกรณ์พีแอลดีถูกเตรียมเป็นวงจรพื้นฐาน ทางด้านลอจิกต่อกันอยู่เป็นกลุ่มมีทั้งวงจรคอมไบเนชั่น (Combination) และซีเควนเชียล (Sequential) ซึ่งมีส่วนประกอบเป็นวงจรภายในเทคโนโลยีของวงจรที่ใช้สร้างพีแอลดีมีทั้ง ทีทีแอล (TTL) อีซีแอล (ECL) และ ซีมอส (CMOS) ตามความเหมาะสมของแต่ละระบบ อุปกรณ์พีแอลดีทุกชนิดมีหลักการพื้นฐานของวงจรภายในที่เหมือนกันโดยมีวงจรคอมไบเนชั่นที่ให้ผลเป็นผลคูณร่วมบวก (Sum of product) ประกอบไปด้วยชุดของแอนด์เกตที่ต่อร่วมกับออเกต การโปรแกรมคือการเลือกว่าจะให้มีการต่ออินพุตภายในของแอนด์เกตกับสัญญาณอินพุตใดบ้าง ซึ่งมีทั้งจากภายนอกและสัญญาณป้อนกลับจากเอาต์พุตภายในเอง การติดต่ออินพุตของออเกตกับเอาต์พุตของ แอนด์เกต ตัวต่างๆ วิธีการเลือกหรือการโปรแกรมทางกายภาพ อินพุตต่างๆของอุปกรณ์ทุกตัวจะถูกต่อผ่านฟิวส์เข้ากับแหล่งสัญญาณ ซึ่งถ้าไม่ต้องการใช้สัญญาณใดจะตัดฟิวส์ทำให้สามารถโปรแกรมได้ครั้งเดียว อุปกรณ์พีแอลดีบางชนิดใช้มอสทรานซิสเตอร์แทนฟิวส์ทำให้สามารถโปรแกรมโดยใช้กระแสไฟฟ้า และสามารถลบและโปรแกรมใหม่เข้าไปได้อีก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

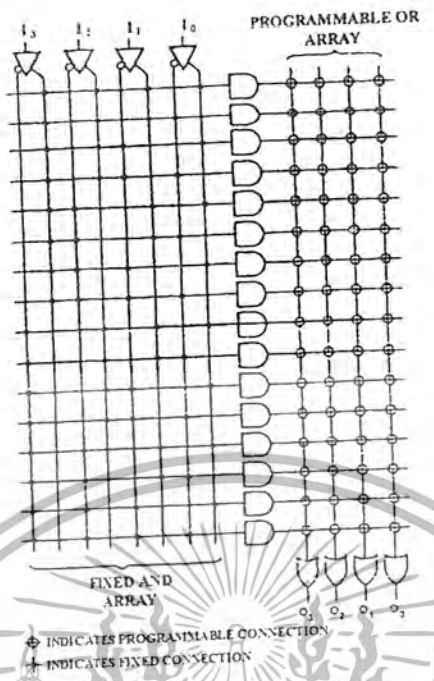


รูปที่ 4.2 แสดงวงจรพื้นฐานของอุปกรณ์พีแอลดีซึ่งอยู่ในรูปผลคูณร่วมบวก

4.2.2 พรอม (**PROM: Programmable Read Only Memory**)

พรอมคือหน่วยความจำรอม (ROM) โปรแกรมได้ ซึ่งนับว่าเป็นอุปกรณ์พีแอลดีชนิดหนึ่งซึ่งวงจรภายในของพรอมเสมือนกับประกอบไปด้วย แถวลำดับของแอนด์และออเกต (And/Or Array) ผลเอาต์พุตที่ขาดค้างเอาต์พุตสามารถแสดงในสมการของฟังก์ชั่นผลคูณร่วมบวก (Sum of product) ของสัญญาณอินพุตที่ขาแอดเดรส รูปที่ 3 แสดงถึงลักษณะการต่อเป็นแถวลำดับของแอนด์เกตและออเกตของพรอมขนาด 16 x 4 บิต วงจรทางด้านซ้ายบนสุดเป็น แอนด์เกตที่ให้ผลเป็นผลคูณ (Product) ของกรณีอินพุตเป็น 0000 แอนด์เกตที่อยู่ถัดลงมาเป็นผลคูณของกรณีที่อินพุตเป็น 0001, 0010, ... จนถึงตัวล่างสุดคือผลคูณในกรณีที่อินพุตเป็น 1111 ที่เอาต์พุตแต่ละบิตของหน่วยความจำ สามารถเลือกได้ว่าจะให้เป็น 1 ในกรณีที่อินพุตจากแอดเดรส เป็นอย่างไรบ้าง เหมือนกันเป็นจากนำเอาต์พุตจากผลคูณที่ต้องการให้เอาต์พุตแต่ละบิตเป็น 1 ไปออกันจึงเปรียบเหมือนกับว่าในพรอมมีจำนวนแอนด์เกตเท่ากับจำนวนตำแหน่งความจำ และมีออเกตจำนวนเท่ากับจำนวนบิตของสัญญาณข้อมูลออก (Data output) อินพุตออเกตทุกตัวสามารถต่อเข้ากับแอนด์เกตตัวใดก็ได้ทุกตัว ซึ่งอาจเรียกได้ว่าเป็นพีแอลดีแบบ fixed AND / programmable OR

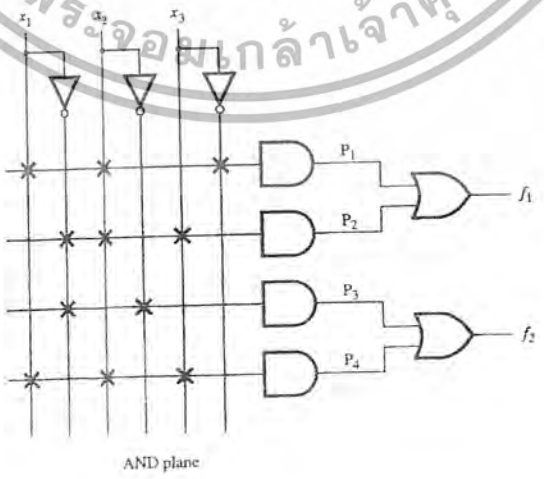
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.3 แสดงลักษณะของพรอมเมื่อเปรียบเทียบเป็นวงจรในรูปผลคูณร่วมบวก

4.2.3 พีแอลแอล (PAL: Programmable Array Logic)

ในช่วงกลางปี ค.ศ. 1970 บริษัท เอ็ม เอ็มไอ (MMI : Monolithic Memory) ในประเทศสหรัฐอเมริกา ได้พัฒนาอุปกรณ์พีแอลแอล เป็นพีแอลชนิดใหม่โดยใช้เทคโนโลยีแบบแอลเอสไอ (LSI : Large Scale Integration) สามารถโปรแกรมเลือกวงจรภายใน โดยใช้ฟิวส์ที่เชื่อมต่ออยู่ระหว่างสัญญาณอินพุตภายนอก และการป้อนกลับจากภายในกับแอนด์เกตที่ต่อเป็นฟังก์ชั่นผลคูณ (Product) อยู่ในตัววงจร

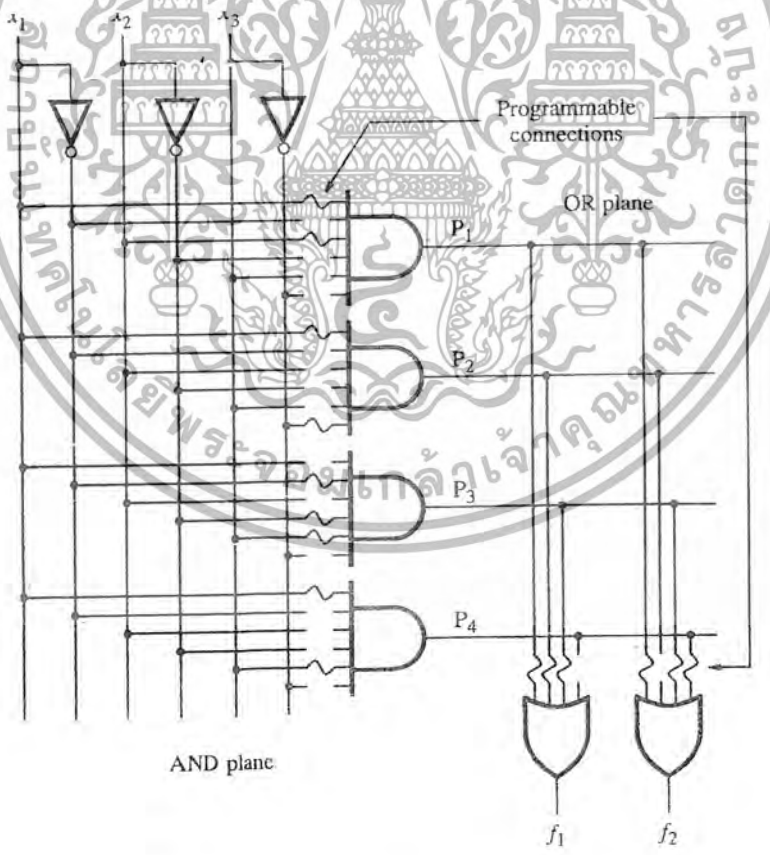


รูปที่ 4.4 แสดงโครงสร้างภายในของพีแอลแอล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2.4 พีแอลเอ (PLA: Programmable Logic Array)

อุปกรณ์ที่สามารถโปรแกรมได้แบบพีแอลเอเกิดขึ้นเมื่อปี ค.ศ. 1975 โดยบริษัทซิกเนทริกส์ (Signetics) สหรัฐอเมริกา ซึ่งเป็นบริษัทผู้ผลิตวงจรรวมรายใหญ่รายหนึ่ง ผลิตและนำเสนออุปกรณ์โดยใช้ชื่อว่า เอฟพีแอลเอ (FPLA : Field Programmable Logic Array) สามารถโปรแกรมการต่อลอจิกทั้งทางด้านแอนด์เกตและออเกตได้ และยังเลือกเอาต์พุตเป็น active high หรือ active low โดยต่อผ่านเอ็กคูลสีบออเกต ให้ทำหน้าที่เป็นนอนอินเวอร์เตอร์หรือเป็นอินเวอร์เตอร์แล้วแต่ภายในของพีแอลเอ ต่อมาปี ค.ศ. 1979 บริษัทซิกเนทริกส์ ได้สร้างเอฟพีแอลเอใหม่ที่มีเรจิสเตอร์ต่ออยู่ภายในวงจร เพิ่มขึ้น รวมทั้งสามารถเลือกสัญญาณอินพุตที่มาจากการป้อนกลับจากเรจิสเตอร์ได้ด้วย ทำให้สามารถใช้อุปกรณ์พีแอลเอใหม่นี้สร้างวงจร State machine ได้ อุปกรณ์ใหม่ที่มีเรจิสเตอร์ อยู่ด้วยนี้ถูกตั้งชื่อใหม่เป็น เอฟพีแอลเอส (FPLS: Field Programmable Logic Sepucncer) มีทั้งที่เป็นทีทีแอลและซีมอส



รูปที่ 4.5 แสดงวงจรพื้นฐานภายในของพีแอลเอ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2.5 FPGA (Field Programmable Gate Array)

FPGA เป็นชิพในลักษณะของ Programmable Device ที่เราสามารถโปรแกรมตัวมันให้สามารถเป็นวงจรดิจิตอลใดๆก็ได้ โครงสร้างข้างในประกอบด้วย อาร์เรย์ของลอจิกเกทต่างๆมากมาย ซึ่งในปัจจุบันความจุเกทภายในตัวชิพ FPGA ได้เพิ่มขึ้นในระดับไม่กี่พันตัว จนถึงล้านตัว ดังนั้นจึงรองรับวงจรดิจิตอลที่มีความสลับซับซ้อนเป็นอย่างมาก และรวมถึงความเร็วในการทำงานที่ได้เปรียบอย่างมากมายเมื่อเปรียบเทียบกับไมโครโปรเซสเซอร์ นอกจากนี้ในด้านการออกแบบพัฒนาและทดสอบก็ทำได้ง่าย เนื่องจากเราสามารถโปรแกรมได้เองและสามารถโปรแกรมหลายๆครั้ง จึงไม่มีความเสี่ยงใดๆทั้งสิ้น และในปัจจุบันการออกแบบที่เป็น FPGA Base IC Design นั้นก็เป็นที่นิยมทำกันและมีแนวโน้มที่จะนำมาใช้งานมากขึ้นเรื่อยๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5
ประวัติความเป็นมาของภาษาVHDL

5.1 ประวัติความเป็นมาของภาษาVHDL

วิวัฒนาการของภาษาVHDL เริ่มประมาณปี ค.ศ.1981 โดยที่กระทรวงกลาโหมสหรัฐอเมริกา หรือ Department of Defense (DOD) มองเห็นว่าอุปกรณ์อิเล็กทรอนิกส์ และคอมพิวเตอร์ที่ใช้ในกิจการทหาร ได้รับ การพัฒนาเป็นไปอย่างล่าช้า ซึ่งไม่อาจยอมรับได้ในปัจจุบัน เพราะเทคโนโลยีทางด้านไมโครอิเล็กทรอนิกส์ ได้รับการพัฒนาเป็นไปอย่างรวดเร็ว ดังนั้นทาง DOD จึงตั้งโครงการขึ้นเพื่อศึกษา วิธีการที่จะช่วยพัฒนาวงจรอิเล็กทรอนิกส์ โดยเฉพาะอย่างยิ่งระบบวงจรดิจิตอล ให้สามารถนำไปผลิตได้เร็วขึ้น และโครงการดังกล่าวที่มีชื่อว่า Very High Speed Integrated Circuits หรือ VHSIC ในระยะแรกนั้นโครงการเป็นความลับทางด้านความมั่นคงของประเทศ และอยู่ในความควบคุมดูแลของ United States International Traffic and Arms Regulation (ITAR) ในปี ค.ศ. 1983 ตามคำแนะนำของคณะทำงาน ("Woods Hole" workshop) ทางDOD ได้ออกความต้องการมาตรฐานของภาษาที่ใช้สำหรับการบรรยายพฤติกรรมของวงจร หรือ Hardware ของระบบสำหรับโครงการ VHSIC ซึ่งมีสาระสำคัญพอสรุปได้ดังนี้

- ต้องเป็นภาษาที่นำไปเขียนรูปแบบระบบดิจิตอล และมีคุณสมบัติที่สามารถจะเข้าใจได้ทั้งคนและเครื่อง โดยไม่มีการแปล หรือเปลี่ยนแปลงอีก
- สามารถนำไปใช้เป็นเอกสารประกอบโครงการได้ (Project Documentation)
- ต้องเป็นภาษาที่เขียนขึ้นสำหรับใช้ในการจำลองการทำงานของวงจร (Simulation Language)

ฉะนั้น ภาษานี้จึงเป็นภาษาโปรแกรมระดับสูง (High Level Language) เช่นเดียวกับภาษา PASCAL, FORTRAN, ADA ซึ่งใช้ในวิศวกรรมการออกแบบHardware ที่เรียกว่า Hardware Description Language หรือ HDL ดังนั้นภาษามาตรฐานนี้จึงมีชื่อว่า VHSIC-HDL หรื อ VHDL นั่นเอง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การเขียนภาษา VHDL

การนำภาษาVHDL มาบรรยายฟังก์ชันการทำงานของ hardware ในระบบดิจิตอลนั้น คือความอ่อนตัวของภาษาที่สามารถจำลองการทำงานจากหลักการของรูปแบบ (Simulate Conceptual Design) แต่ในขณะเดียวกันนั้นก็สามารถจำลองการทำงานของhardware ที่ให้รายละเอียดเกี่ยวกับเวลาอย่างถูกต้อง (Timing Based Simulation) และจากโครงสร้างของภาษา ยังสามารถจำลองการทำงานในรูปแบบของลำดับขั้น (Hierarchy of Simulation Levels) ความสามารถดังกล่าวนี้ ช่วยให้นักวิศวกรออกแบบ สามารถที่จะเขียนรูปแบบบรรยาย จากระดับสูงสุดของวงจรที่อยู่ในรูปสังเขป (High Level of Abstraction) ลงสู่รายละเอียดในระดับล่างของวงจรได้เช่น gate level เป็นต้น

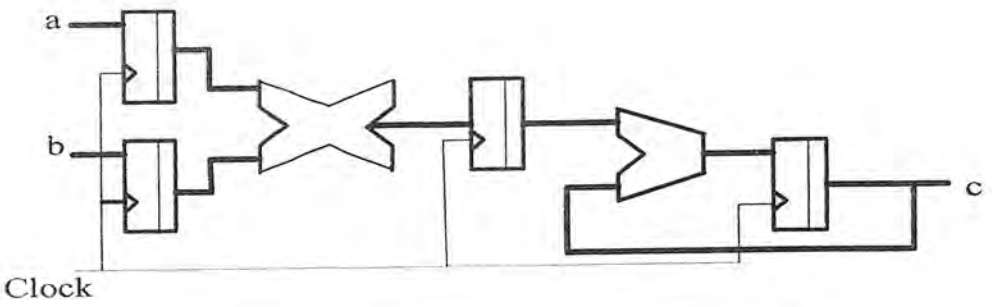
ภาษา VHDL เป็นภาษาที่สนับสนุนการเขียนรูปแบบในทุกๆลักษณะ และวิธีการ ดังเช่นตัวอย่างที่แสดงในรูป 1.1 คือรูปแบบ (Model) ของลำดับขั้นตอนของการคูณ และการหาผลรวม (Multiply Accumulate Algorithm) ของตัวแปรสองตัว a และ b ส่วนผลลัพธ์คือ c ในลักษณะของเลขฐานสอง (Binary Number) ในรูปแบบนี้แสดงในระดับที่สังเขปที่สุดของแนวความคิดที่จะแก้ปัญหา เพื่อหาผลลัพธ์ โดยไม่ได้คำนึงถึงโครงสร้างของวงจรอย่างที่เคยชิน เช่น อุปกรณ์วงจรรวม Arithmetic and Logic Unit (ALU)

```
FOR i IN 1 TO 1024 LOOP
        Result:=result + a(i)*b(i);
END LOOP;
C<=result;
```

รูปที่ 5.1 VHDL Statement สำหรับ Multiply Accumulate Algorithm

ภาษาVHDL สามารถใช้บรรยายลำดับขั้นตอนของการคูณ และหาผลรวม (Multiply Accumulate Algorithm) โดยแสดงในรายละเอียดของการสร้างวงจรดังกล่าวจริงๆ ตามรูปที่ 1.2

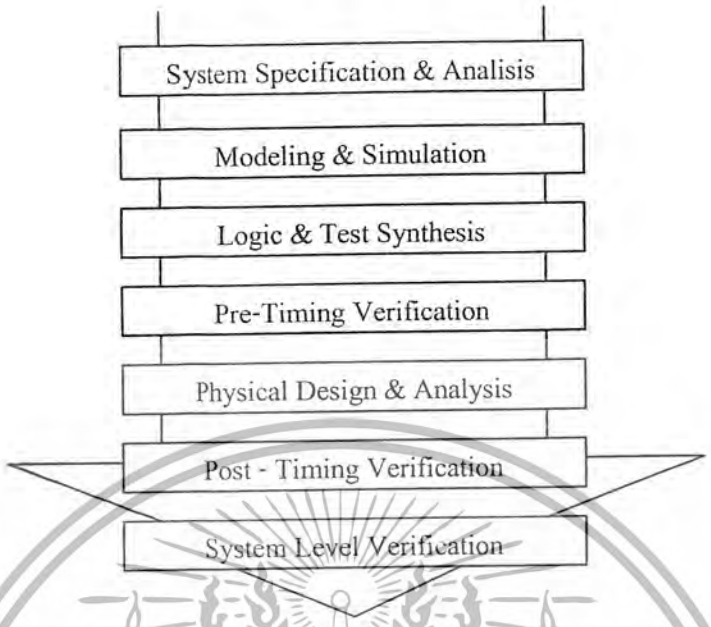
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.2: โครงสร้างของ Multiply Accumulate Unit

นอกจากนั้นVHDL ยังเป็นภาษาที่สนับสนุนลักษณะต่างๆของระบบดิจิตอล (Digital System) ที่มีความซับซ้อนได้ทั้งหมด การเริ่มต้นขบวนการของTop-Down Design เริ่มต้นด้วยการเขียนรูปแบบ (Modeling) ในระดับบน (Top Level) ของแนวความคิดอย่างสังเขป วิศวกรสามารถที่จะพัฒนาสภาพแวดล้อมต่างๆเพื่อใช้ตรวจสอบความถูกต้องของแนวคิดกับสิ่งที่ต้องการจริง หรือ Specification ของงาน จากรูปที่ 1.3 แสดงถึงขั้นตอนการออกแบบในลักษณะของ Top Down Design ทั้งนี้ในทางปฏิบัติอาจมีข้อแตกต่างไปบ้างเล็กน้อย เนื่องมาจากขั้นตอนของการผลิต (Implementation) สามารถทำได้ในหลายๆเทคโนโลยี ได้แก่ Programmable Logic Devices ได้แก่ PLA, FPGA หรือ CPLD เป็นต้น นอกจากนี้ยังมี Semi-Custom IC (Gate Array, Standard Cell) และ Full Custom IC

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.3 ขั้นตอนของ Top Down Design

ขบวนการออกแบบ โดยใช้วิธี Top Down Design มีรายละเอียดดังนี้

- **System Specification and Analysis** : ขั้นตอนในการสร้างข้อกำหนดของความต้องการ (Specification) และระบบวิเคราะห์ระบบ เพื่อหาแนวความคิดและหลักการ (Idea and Concept) ในการแก้ปัญหา
- **Modeling and Simulation** : การเขียนรูปแบบของระบบที่ต้องการโดยใช้ภาษา VHDL หรือภาษา HDL อื่นๆ และเพื่อตรวจสอบความถูกต้องและข้อกำหนด(Specification)
- **Logic and Test Synthesis** : หลังจากที่ได้หลักการขั้นต้นพร้อมแนวความคิดที่ผ่านการตรวจสอบแล้ว มาเป็นลำดับขั้นตอนจนถึงอยู่ในระดับที่จะนำไปผลิตวงจรหรือสังเคราะห์ ในขั้นตอนนี้เทคโนโลยีที่จะมารองรับการออกแบบจะถูกกำหนดขึ้นให้อยู่ในรูปของวงจรที่ประกอบด้วยอุปกรณ์อิเล็กทรอนิกส์ (gate-level) และการเชื่อมต่อระหว่างกันของอุปกรณ์เหล่านั้น หรือไม่ก็อยู่ในรูป Netlist ที่สามารถนำไปผลิตลงบนอุปกรณ์อื่นได้
- **Pre – Timing Verification** : หลังจากการสังเคราะห์วงจรให้อยู่ในรูป gate – level หรือ netlist แล้ว ข้อมูลที่ได้จากผู้ผลิตวงจรนั้น นอกจากจะเป็นข้อมูลสำหรับจำลองการทำงานในเรื่องความถูกต้องของฟังก์ชัน (functional simulation) แล้ว ยังมีข้อมูลที่เกี่ยวกับเวลาด้วย ซึ่งเป็นความจริงที่ว่า อุปกรณ์ทางอิเล็กทรอนิกส์ทุกชิ้นจะมี propagation delay เสมอ 5 ถึงแม้ว่าจะเป็นเวลาที่น้อยมากในระดับ nanosecond แต่ภายในวงจรหนึ่งประกอบด้วยเกท

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ของฟังก์ชั่นต่างๆ จำนวน 1000 gate ขึ้นไป เวลาดังกล่าวนี้สะสมกันมากขึ้น จนอาจจะทำให้การทำงานของวงจรรวมทั้งหมดผิดไป หรือไม่สามารถทำงานในย่านความถี่สัญญาณนาฬิกาที่สูง

- **Physical Design and Analysis** : คือขั้นตอนของการผลิตเป็นวงจรจริง (technology and device mapping) โดยนำข้อมูลที่ได้จากการสังเคราะห์มาผลิต ซึ่งอาจจะอยู่ในรูปของแผงวงจรไฟฟ้า (Printed Board: PCB) ที่ประกอบด้วยอุปกรณ์หลายๆชิ้น หรืออยู่ในรูปของวงจรรวมเฉพาะงาน (ASIC)
- **Post – Timing Verification** : หลังจากที่ได้วงจรจริงมาแล้ว ยังต้องมีความจำเป็นที่ต้องตรวจสอบการทำทำงานที่คำนึกถึงเวลาด้วย เพื่อความถูกต้องของวงจรครั้งสุดท้ายก่อนที่จะนำไปรวมเข้ากับอุปกรณ์อื่นๆ ให้เป็นระบบดิจิตอล เพราะในขั้นตอนนี้วงจรที่ออกแบบ จะประกอบด้วย input และ output pad ซึ่งเป็นจุดต่อที่สำหรับรับและส่งสัญญาณกับภายนอก
- **System Level Verification** : หลังจากที่นำวงจรที่ออกแบบรวมเข้ากับอุปกรณ์อื่นๆ ให้เป็นระบบดิจิตอลแล้วนั้น เป็นการควบคุมคุณภาพของผลิตภัณฑ์

5.2 Terminology and Conventions

การเขียนรูปแบบของระบบดิจิตอลด้วยภาษา VHDL นั้น จะมีศัพท์เทคนิคเฉพาะ ฉะนั้นในบทนี้จะเป็นการบรรยาย และอธิบายศัพท์บางคำที่จะต้องใช้

ลักษณะของรูปแบบ (Model styles): ลักษณะของการเขียนรูปแบบ (model) ด้วยภาษา VHDL สามารถแบ่งได้เป็น

- **Behavioral Model** : หรือเรียกอีกอย่างได้ว่า algorithmic description เป็นรูปแบบที่บรรยายพฤติกรรมของระบบดิจิตอล ในส่วนที่บรรยายมีส่วนคล้ายกับภาษาชั้นสูง (high level language) ทั่วๆ ไปเช่น Pascal หรือ C เป็นต้น
- **Dataflow Model** : เรียกอย่างหนึ่งได้ว่า " Register Transfer Level " (RTL) เป็น รูปแบบที่ถูกเขียนขึ้น เพื่อจุดประสงค์ที่จะใช้เครื่องมือสำหรับสังเคราะห์วงจรอัตโนมัติ รูปแบบนี้ลักษณะส่วนใหญ่จะเป็น procedural constructs และ functional operators
- **Structural Model** : เป็นรูปแบบที่แสดงการเชื่อมต่อกันระหว่างอุปกรณ์ต่างๆ ที่ประกันขึ้นเป็นวงจรหรือระบบดิจิตอล และสามารถเรียกอีกอย่างได้ว่า "netlist representation" เป็นการเขียนที่แสดงให้เห็นถึงโครงสร้างของ Hardware

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- **Mixed – Level Model** : จากคุณสมบัติที่อ่อนตัวของภาษา VHDL จึงสามารถที่จะเขียนรูปแบบ โดยใช้ลักษณะต่างๆที่กล่าวมาแล้วข้างต้น บรรยายวงจรหรือระบบดิจิตอลเดียวกัน ฉะนั้นรูปแบบเช่นนี้จึงมีการเขียนแบบผสม
- **Concurrency** : ในภาษา VHDL นั้น ชุดคำสั่ง (statements) แต่ละชุดจะทำงานในเวลาเดียวกันอิสระต่อกัน ลักษณะเช่นนี้เป็นคุณสมบัติที่เป็นความจริงทางฟิฟิกส์ของวงจรอิเล็กทรอนิกส์ ชุดคำสั่งนี้เรียกว่า " **concurrent statement** " และจะทำงานเมื่อมีการเปลี่ยนแปลงค่าของสัญญาณ
- **Sequential** : นอกจากความสามารถที่ชุดคำสั่งจะทำงานแบบ concurrent แล้ว บางครั้งการเขียนรูปแบบในลักษณะที่บรรยายพฤติกรรมของวงจร มีความจำเป็นที่จะต้องให้ชุดคำสั่งทำงานเป็นลำดับขั้นเรียงกันจากบนลงล่าง
- **Driver :** สัญญาณต่างๆ (signal) ใน VHDL นั้นจะถูกควบคุมด้วยตัวขับหรือ "driver" สัญญาณเหล่านี้จะรับค่าใหม่ (ระดับของสัญญาณ) ได้ด้วยตัวขับนี้เอง
- **Transaction :** การเกิด Transaction กับ signal นั้นจะเกิดขึ้นเมื่อมีการกำหนดค่าๆหนึ่งให้กับ signal นั้น ค่าใหม่ที่ signal ได้รับอาจมีผลหรือ ไม่มีผลทำให้เกิดการเปลี่ยนแปลงของระดับสัญญาณ (event)
- **Event :** คือการเปลี่ยนระดับค่าของ Signal จากระดับหนึ่งไปสู่ระดับอื่น เช่น จาก 0 เป็น 1
- **Sensitivity List :** คือรายชื่อของ signal ต่างๆ ที่มีผลให้เกิดการทำงานของ concurrent statement เมื่อเกิด event ขึ้นกับ signal ตัวใดตัวหนึ่งหรือหลายตัวพร้อมกันในรายชื่อนั้น
- **Object :** ในภาษา VHDL นั้นคำว่า object ใช้เขียนเพื่อบ่งบอกถึงองค์ประกอบส่วนหนึ่งของรูปแบบ ซึ่งเปรียบได้เหมือนกับภาชนะที่มีไว้สำหรับบรรจุค่าต่างๆ สามารถแบ่งออกได้เป็นสามชั้น (class) ด้วยกันคือ
- **CONSTANT** : ได้แก่ object ประเภทหนึ่งที่เมื่อกำหนดค่าเริ่มต้นให้แล้ว จะคงค่านั้นไว้ตลอด ไม่สามารถดัดแปลง หรือแก้ไขได้ สามารถประกาศใช้ได้ในส่วนที่เป็นส่วนประกาศต่างๆของรูปแบบ(model)
- **SIGNAL** : หมายถึง object ประเภทหนึ่งที่สามารถกำหนดค่าที่สัมพันธ์กับเวลาให้ได้ นั้นหมายความว่า SIGNAL สามารถ รับค่าได้เพียงค่าเดียวเท่านั้นในขณะเวลาหนึ่ง SIGNAL จะรับค่าๆหนึ่งได้จากตัวขับสัญญาณหรือ driver ซึ่งตัวขับนี้อาจจะเก็บค่าในอนาคตสำหรับ SIGNAL ไว้ด้วย SIGNALสามารถประกาศใช้ได้ในส่วนที่เป็นเนื้อที่ของ concurrent body เท่านั้น ดังนั้น SIGNAL จึงสามารถถูกนำไปใช้ได้ตลอดโครงสร้างของรูปแบบ(model) หรือที่เรียกว่า global object

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- **VARIABLE :** หรือตัวแปรได้แก่ object ที่สามารถกำหนดค่าใดๆให้ได้และ สามารถที่จะเปลี่ยนแปลงค่าได้ตลอดการจำลองการทำงาน แต่จะเก็บค่าเพียงค่าเดียวเท่านั้นในขณะเวลาหนึ่ง เนื่องจาก VARIABLEสามารถประกาศใช้ได้ในส่วนที่เป็นsequential body เท่านั้นอันได้แก่ส่วนประกาศของ PROCESS,FUNCTIONหรือPROCEDUREดังนั้น VARIABLE จึงสามารถนำไปใช้ได้เฉพาะในขอบเขตที่ถูกประกาศใช้เท่านั้น(local object)
- **การประกาศใช้ object (object declaration) :** การที่จะใช้ object ชั้นต่างๆตามที่กล่าวมาแล้วในการเขียนรูปแบบด้วยภาษา VHDL นั้นจะต้องมีการประกาศใช้ก่อน การประกาศใช้ object สามารถใช้ชุดคำสั่งตามโครงสร้างดังนี้

```
object_class identifier:TYPE[:=initial_value];
```

ซึ่ง object_class ได้แก่ CONSTANT,SIGNALหรือ VARIABLE การตั้งชื่อ(identifier)เป็นไปตามกฎของภาษาVHDL ซึ่งจะกล่าวถึงในส่วนต่อไป TYPEคือการกำหนดประเภทของ object ที่ประกาศนั้นๆ นอกจากนั้นยังสามารถกำหนดค่าเริ่มต้นของ object ได้ (initial_value) ซึ่งส่วนนี้เป็นเพียง option และการประกาศจะต้องอยู่ในพื้นที่ที่กำหนดให้ของแต่ละส่วนของรูปแบบ (declarative area)

- **การตั้งชื่อ object :** การตั้งชื่อจะต้องเป็นไปตามกฎต่อไปนี้
  1) ชื่อ (identifier) ประกอบด้วยตัวหนังสือ (พยัญชนะและตัวเลข)ในภาษาอังกฤษได้แก่
     - พยัญชนะA-Z, a-z
     - ตัวเลข 0,1,2,3,4,5,6,7,8,9
     - เครื่องหมายขีดเส้นใต้ (underscore) "_"
  2) ชื่อจะต้องขึ้นต้นด้วยพยัญชนะเสมอ
  3) ชื่อสามารถประกอบด้วยพยัญชนะ ตัวเลข และเครื่องหมายขีดเส้นใต้จำนวนไม่จำกัด
  4) การใช้เครื่องหมายขีดเส้นใต้ (_) ทุกครั้ง จะต้องนำหน้าด้วยพยัญชนะหรือตัวเลขและตาม ด้วยตัวพยัญชนะหรือตัวเลขเช่นกัน
  5) พยัญชนะตัวใหญ่หรือตัวเล็กไม่มีความแตกต่างกัน (case insensitive)
  6) ห้ามใช้คำสงวน (reserved word) ของภาษา VHDL (ผนวก ง.)
- **ค่าเริ่มต้น (initial value) :** การประกาศใช้ object ทุกครั้งควรจะต้องกำหนดค่าเริ่มต้นให้ด้วย เพราะค่าที่กำหนดนี้จะถูกนำไปใช้ เมื่อเริ่มต้นจำลองการทำงาน(simulation) ของรูปแบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
SIGNAL example_signal : BIT := '1';
```

ในกรณีที่ไม่มีการกำหนดค่าเริ่มต้น ระบบจำลองการทำงานจะนำค่าที่น้อยที่สุด (อยู่ทางซ้ายมือสุด ของกลุ่มค่าของแต่ละประเภท (TYPE) มาเป็นค่าเริ่มต้นแทน

- **ประเภทข้อมูล(data type) :** ได้แก่ TYPE ของ object ที่จะเป็นตัวกำหนดค่าใดบ้างในกลุ่มของค่า (set of value) ของแต่ละ TYPE สามารถที่จะกำหนดให้กับobject ได้ นอกจากนั้น TYPE ยังเป็นตัวกำหนดการทำงานในลักษณะต่างๆ(operation) ของobject นั้นๆ TYPE แบ่งออกเป็น 4ประเภทคือ
  1) **Scalar** ได้แก่ตัวเลข (Numeric) ซึ่งในภาษาVHDL มีตัวเลขจำนวนเต็มบวก(INTEGER) เช่น 1,30,100 เป็นต้น และเลขจำนวนจริง(REAL)เช่น1.0,30.,1E2เป็นต้น
  2) **Enumeration** กลุ่มของค่าประเภทนี้ได้แก่ ตัวหนังสือ หรือชื่อต่างๆ
  3) **Physical** ได้แก่หน่วยทางฟิสิกส์ใน[illegible]
  4) **File** เป็น TYPE ภายนอกสามารถมีค่าได้หลายๆอย่าง
- **ประเภทย่อยของข้อมูล(subtype) :** สำหรับ TYPE ที่กำหนดไว้แล้ว(predefined type และ user-defined type)สามารถที่จะแบ่งออกเป็นกลุ่มย่อยลงไปได้อีก โดยที่องค์ประกอบของ TYPEใหม่นี้จะเป็นส่วนหนึ่งของTYPE เดิม หรือที่เรียกว่า **subtype**

```
TYPE qite IS( '0', '1', 'Z', 'X' );
SUBTYPE tit IS qit RANGE '0' TO 'Z';
```

- **ประเภทของ object ที่กำหนดไว้แล้ว(predefined type) :** ได้แก่ TYPEที่กำหนดไว้ใน package ชื่อ STANDARD และกำหนดโดย IEEEว่าจะต้องมีในระบบที่ใช้พัฒนา VHDL ฉะนั้นจึงไม่จำเป็นต้องประกาศใช้ในทุกๆรูปแบบที่เขียนขึ้น TYPE ประเภทนี้ได้แก่
  1. **BOOLEAN** คือกลุ่มของค่า FALSE และ TRUE
  2. **BIT** คือกลุ่มของค่า '0' และ '1'
  3. **INTEGER** คือกลุ่มของค่า –214748347 ถึง 214748347
  4. **REAL** คือ กลุ่มของค่า –1.0E38 ถึง 1.05E38
  5. **CHARACTER** คือกลุ่มของค่า พยัญชนะ 'A' – 'Z' , 'a'-'z', อักษรหรือเครื่องหมายพิเศษและตัวอักษรควบคุม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6. **TIME** คือได้แก่หน่วยเวลาที่มีค่าพื้นฐานเป็นวินาที(second ย่อ ด้วยs หรือ S)
7. **SEVERITY LEVEL** คือกลุ่มของค่า NOTE, WARNING,ERROR,FAILURE

- **Operator :** เนื่องจากTYPE ของobject จะเป็นตัวบ่งบอกถึง operator ที่ object นั้นๆ สามารถกระทำได้ ตลอดจน TYPE ของผลลัพธ์ที่ได้จากการทำงานในภาษาVHDL แบ่ง operator ออกได้เป็นประเภทต่างๆตามที่แสดงในรูปที่1.6
- **Delay :** หมายถึงช่วงระยะเวลาระหว่างที่เริ่มต้นของสาเหตุ จนกระทั่งเป็นผลออกมาให้เห็นของปรากฏการณ์หนึ่งๆในความเป็นจริงทางธรรมชาติอุปกรณ์ hardware ทุกอย่าง อาทิ เช่น gateต่างๆในระบบดิจิตอลจะมีdelay แฝงอยู่เสมอ (ที่เรียกว่า propagation delay time) การที่ภาษาVHDL เป็นภาษาที่ใช้บรรยาย hardware จึงสามารถบรรยายพฤติกรรมของ delay ได้ ซึ่งจำแนก delay ออกเป็น

Predefined Operators	
Logical Operator:	NOT,AND,OR,NAND,NOR,XOR
Operand TYPE:	BIT,BOOLEAN
Result TYPE:	BIT,BOOLEAN
Relational Operator:	=,/=,<,<=,>,>=
Operand TYPE:	TYPE ใดๆก็ได้
Result TYPE:	BOOLEAN
Arithmetic Operator:	+,-,*,/,**,MOD,REM,ABS
Operand TYPE:	INTEGER,REAL,Physical
Result TYPE:	INTEGER,REAL,Physical
Concatenation Operator:	&
Operand TYPE:	arrayของ TYPE ทุกประเภท
Result TYPE:	arrayของ TYPE ทุกประเภท

ตารางที่ 5.1 Operator ที่กำหนดไว้ในภาษา VHDL

1) **Delay Selection** คือ การกำหนดdelayที่จะมีผลต่อSIGNALในรูปแบบลักษณะใด Inertial Delay ได้แก่ ปฏิกิริยาต่อต้านการเปลี่ยนแปลง ระบบที่ประกอบด้วยinertial delay จะแสดงผลของการหน่วงเวลาต่อเมื่อ สัญญาณที่มาบังคับให้เกิดการเปลี่ยนแปลง(จากdriver) มีช่วงระยะเวลานานกว่า inertial delayของระบบนั้นๆ Transport Delay จากความหมาย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ของคำว่า transport คือ การขนส่งจากจุดหนึ่งไปยังอีกจุดหนึ่งนั้น ฉะนั้น transport delay จึงแสดงผลของการหน่วงเวลาทุกครั้งเสมอไม่ว่าการเปลี่ยนแปลงของตัวขับ (driver)นั้นจะมีช่วงระยะเวลานานเท่าไร

2) **Internal Delay** ได้แก่ delay ภายในระบบของVHDL ทั้งนี้เนื่องจากภาษา VHDL เป็นภาษาที่ชุดคำสั่งทั้งหลายทำงานแบบ concurrent ต่อกัน ซึ่งแตกต่างไปจากภาษาโปรแกรมชั้นสูงทั่วๆไป แต่การทำงานของระบบจำลองการทำงานด้วยเครื่องคอมพิวเตอร์(simulator) เครื่องไม่สามารถที่จะทำงานสองคำสั่ง(หรือมากกว่า)ได้กลไกที่จะทำให้เป็นไปตามหลักการของconcurrent statement ได้นั้นเรียกว่า delta delay (δ)ที่หมายถึง internal delay ของระบบพัฒนาVHDL

- **LRM :** คือคู่มืออ้างอิง (Language Reference Manual) ของมาตรฐาน IEEE 1076(ปัจจุบัน IEEE1076-1993)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 6

ส่วนต่างๆของแบบ (Design Units)

6.1 กล่าวนำ

ก่อนที่จะศึกษาต่อไปถึงชุดคำสั่งอื่นๆ ที่ใช้ในการเขียนรูปแบบ:หรือ modeling ด้วยภาษา VHDL มีความจำเป็นที่จะต้องแนะนำให้รู้จักกับส่วนต่างๆของแบบ (design units)ที่ใช้ในภาษาเสียก่อน และนี่ก็เป็นขั้นตอนแรกที่สำคัญที่สุดของการศึกษาเรียนรู้การใช้ภาษา VHDL เขียนรูปแบบบรรยายระบบดิจิตอลในมุมมองของการออกแบบลักษณะ Top-Down Design นอกจากนั้น การที่จะเข้าใจในกฎเกณฑ์ของภาษาได้นั้น จะต้องทำความเข้าใจในเรื่องของโครงสร้าง และส่วนต่างๆของรูปแบบ VHDL ให้ถูกต้องเสียก่อน

ภาษา VHDL นั้นประกอบด้วยส่วนต่างๆ ที่สำคัญและเป็นพื้นฐานของการเขียนรูปแบบระบบดิจิตอลที่สำคัญ 4 หน่วย คือ

1. Entity Design Unit
2. Architecture Design Unit
3. Package Design Unit
4. Configuration Design Unit

6.2 Entity Design Unit

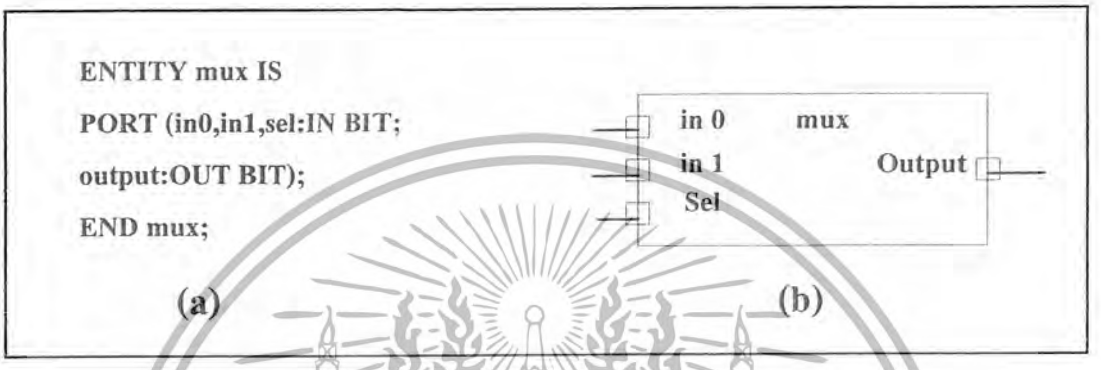
หน่วยของแบบ (Design Unit) ส่วนที่ใช้สำหรับติดต่อระหว่างโลกภายนอกกับรูปแบบ (Model) ที่จะเขียนขึ้น ส่วนนี้เรียกว่า " Entity design unit" ในส่วนนี้ใช้กำหนดจุดต่อ(Connection point) ของรูปแบบ กำหนดทิศทางการไหลของสัญญาณ(mode) และประเภทของค่า(type of value) ที่สามารถกำหนดให้สัญญาณตามจุดต่างๆ (PORT) ของข้อมูลที่ไหลผ่านจุดต่อเหล่านั้นรูปที่6.1 แสดงให้เห็นโครงสร้างอย่างง่ายๆของ entity design unit

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

รูปที่ 6.1โครงสร้างอย่างง่ายๆของ entity design unit

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ส่วนนี้จะขึ้นต้นด้วยคำ ENTITY และ IS ระหว่างคำทั้งสองเป็นไปตามกฎเกณฑ์ของภาษา หลังจากนั้นจะตามด้วยส่วนที่ใช้กำหนดช่องทาง เข้า-ออก ของข้อมูล (input-output) รวมทั้งพารามิเตอร์อื่นๆส่วนนี้เรียกว่าส่วนหัว (entity header) และที่สำคัญคือ entity design unit จะต้องปิดท้ายด้วยคำว่า END และเครื่องหมายอัฒภาค หรือ semicolon(;)



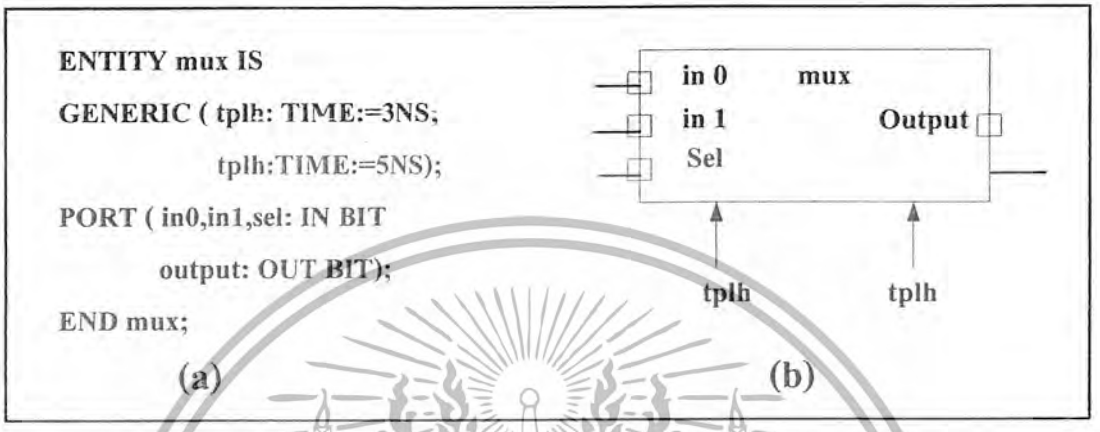
รูปที่6.2 รูปแบบของ2:1multiplexer,(a)VHDL entity design,(b)มุมมองของinterface

ในรูปที่6.2เป็น entity design unit ที่บรรยายอุปกรณ์ที่มีชื่อว่า mux ในส่วนหัวของ entity(header)มีการกำหนดจุดต่อ 4 จุดภายใต้ชุดคำสั่ง PORTโดยที่3 จุดแรกเป็นจุดให้ข้อมูลไหลผ่าน(input)ได้แก่ in0,in1และsel ซึ่งกำหนดด้วยทิศทางการติดต่อกับภายนอก(mode)เป็นการไหลเข้า (IN) ที่แสดงด้วยรูปสี่เหลี่ยมโปร่งในรูปที่6.1 ส่วนจุด output เป็นจุดให้ข้อมูลไหลออก(output) ซึ่งกำหนดด้วยทิศทางการติดต่อกับภายนอกเป็นการไหลออก(OUT) ที่แสดงด้วยรูปสี่เหลี่ยมทึบในรูปที่6.2 ส่วนประเภท(type)ของข้อมูลที่จะไหล เข้า-ออกนั้น เป็นประเภทBIT ที่สามารถมีค่าได้เพียงสองค่าคือ '0' และ '1' เท่านั้น

นอกจากนั้นผู้ออกแบบยังสามารถกำหนดค่าพารามิเตอร์ทางฟิสิกส์ที่เป็นข้อมูลเพิ่มเติมอื่นๆ ลงในส่วนหัวของ entity ได้อีก อาทิ เช่น ข้อมูลเกี่ยวกับความเร็วในการทำงานของอุปกรณ์อันได้แก่ propagation delay time พารามิเตอร์เหล่านี้ เรียกว่า genertic ที่กำหนดด้วยคำสั่ง GENERIC จากตัวอย่างในรูปที่6.2 สามารถที่จะเพิ่มข้อมูลเกี่ยวกับความเร็วในการทำงานได้ตามที่แสดงในรูปที่6.3เช่นเดียวกับตัวอย่างแรกหน่วยของแบบนี้มีช่องทางติดต่อกับภายนอก4จุด แต่ได้เพิ่มข้อมูลของ propagation delay time สำหรับนำไปใช้ในการบรรยายพฤติกรรมของอุปกรณ์ ตัวพารามิเตอร์หรือที่เรียกว่า generic นี้มีชื่อว่า tplh และ tphl มีประเภทของข้อมูลเป็นเวลา (TIME)และจะเป็นค่าตายตัว (default value) สำหรับรูปแบบนี้เสมอ ซึ่งในที่นี้จะมีค่า 3 nanosecond และ 5 nanosecond ตามลำดับ ค่าตายตัวนี้สามารถที่จะเปลี่ยนแปลงให้มีค่าอื่นได้แล้วแต่กรณี ทั้งนี้ขึ้นอยู่กับอุปกรณ์ที่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จะทำการเขียนรูปแบบ ซึ่งจะเห็นได้ภายหลังว่าgeneric นั้นเป็นส่วนที่มีประโยชน์มาก เพราะสามารถสร้างรูปแบบที่มีความอ่อนตัวสูง สามารถนำไปใช้บรรยายอุปกรณ์ประเภทเดียวกันแต่มีความแตกต่างกันทางเทคโนโลยีได้



รูปที่ 6.3 : multiplexer ที่ประกอบด้วยข้อมูลเกี่ยวกับเวลา

(a)VHDL entitydesign , (b) มุมมองของinterface

ในบางกรณีสามารถที่จะใช้ภาษาVHDL สร้างรูปแบบที่ปราศจากช่องทาง ไหล เข้า-ออกของข้อมูล(input-output) ได้ซึ่งส่วนใหญ่จะพบในการสร้างรูปแบบ สำหรับตรวจสอบการทำงานของอีกรูปแบบหนึ่ง(VHDL test bench)

```
ENTITY test_bench IS
END test_bench;
```

รูปที่ 6.4 : Entity design unit ที่ไม่มีการกำหนดช่องติดต่อกับภายนอก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.3 Architecture Design Unit

คือส่วนที่ใช้เขียนบรรยายกำหนดพฤติกรรมของรูปแบบ ในมุมมองของการจำลองการทำงาน(simulation)พฤติกรรมต่างๆที่บรรยายในส่วนนี้ขึ้นอยู่กับข้อมูลที่ผ่าน เข้า-ออก ตรงช่องทางตลอดจนพารามิเตอร์ต่างๆ (ports and generics) ที่กำหนดในentity design unit รูปที่6.5 แสดงให้เห็นโครงสร้างอย่างง่ายๆของarchitecture design unit

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

รูปที่ 6.5 : โครงสร้างอย่างง่ายๆของ architecture design unit

ส่วนของ architecture design unit นั้นเริ่มต้นด้วยคำARCHITECTURE และตามด้วยชื่อ (identifier) สิ่งที่ต้องกำหนดลงไปได้แก่ สิ่งที่แสดงให้เห็นว่า architecture นั้นใช้บรรยายentity design unit ใด (OF <entity design unit> IS) ส่วนที่อยู่ระหว่าง ARCHITECTURE และ BEGIN เป็นส่วนประกาศกำหนด(architecture declarative area)ที่เป็นเพียงส่วนเผื่อเลือก (option) ในบริเวณนี้สามารถใช้เขียนประกาศกำหนดค่าต่างๆที่จะนำไปใช้ภายใน architecture นั้นได้ อาทิ เช่น ประเภท (type) ต่างๆ (ตัวอย่างเช่น BIT,BIT_VECTOR),สัญญาณ(SIGNAL),ดังคงที่ (CONSTANT), โปรแกรมย่อย(ได้แก่ FUCTION และ PROCEDURE) และอุปกรณ์ (COMPONENT)ส่วนที่ใช้บรรยายความสัมพันธ์ระหว่างข้อมูลที่ไหลเข้าและไหลออกของรูปแบบ (สัญญาณที่กำหนดในชุดคำสั่ง PORT)นั้นจะถูกบรรยายในบริเวณเนื้อที่ระหว่างคำว่า BEGINและ ENDของarchitecture design unit และนอกจากนั้นชุดคำสั่งทุกคำสั่งที่อยู่ภายในบริเวณนี้จะเป็นชุดคำสั่งแบบแข่งขนาน(current statement) เท่านั้น architecture design unit จะต้องปิดท้ายด้วยคำสั่ง END และชื่อของarchitecture (identifier) นั้นๆที่เป็นส่วนเผื่อเลือก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โดยทั่วไปการเขียนรูปแบบระบบดิจิตอลด้วยภาษาVHDL สามารถเขียนได้ในลักษณะต่างๆตามที่กล่าวมาแล้ว

- Dataflow description
- Behavioral description
- Structure description
- Mixed model description

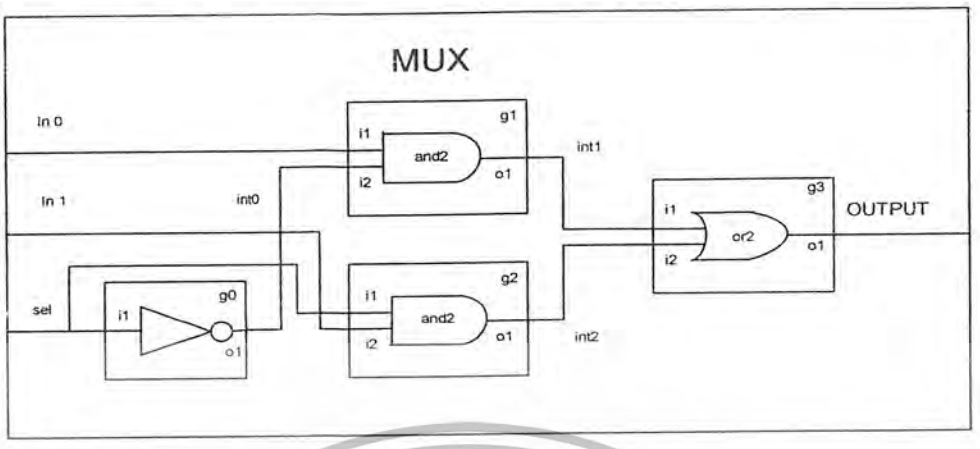
```
ARCHITECTURE data_flow OF mux IS
BEGIN
   Output<=((NOT sel)AND in0)OR(sel AND
in1);
END data_flow;
```

รูปที่ 6.6 : Architecture design unit ของ mux ตาม boolean expression

$$Output=(\overline{sel}*in0)+(sel*in1)$$

รูปที่ 6.6 ส่วนที่บรรยายความสัมพันธ์ระหว่างข้อมูลที่ไหลเข้า(in0,in1)กับข้อมูลที่ไหลออก(output)ประกอบด้วยชุดคำสั่งแบบแข่งขันกันเพียงชุดเดียว ซึ่งเป็นการบรรยายพฤติกรรมของ2:1 mux การบรรยายลักษณะนี้เรียกว่า dataflow description หรือ register transfer level (RTL)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 6.7(a):Architecture description ของ2:1multiplexer (structural description)

รูปที่6.7 เป็นarchitecture ของการบรรยาย 2:1mux ในลักษณะของ structural description โดยใช้ inverter (inv ที่อุปกรณ์ g0),AND-gate 2inputs จำนวน 2gates(and2 ที่อุปกรณ์ g1 และg2) และOR-gate 2 inputs (or2 ที่อุปกรณ์ g3) มาสร้างตามboolean expressionของรูปที่6.6

```
ARCHITECTURE struct OF mux IS
        COMPONENT inv
        PORT(i1:IN BIT;o1:OUT BIT);
        END COMPONENT;
        COMPONENT and2
        PORT(i1,i2:IN BIT;o1:OUT BIT);
        END COMPONENT;
        COMPONENT or2
        PORT(i1,i2:IN BIT;o1:OUT BIT);
END COMPONENT;
        SIGNAL int0,int1,int2:BIT;
BEGIN
        g0:inv
        PORT MAP(i1=>sel,01=>int0);
        g1:and2
        PORT MAP(i1=>in0,i2=>int0,o1=>int1);
        g2:and2
        PORT MAP(i1=>sel,i2=>int1,o1=>int2);
        g3:or2
        PORT MAP(i1=>int1,i2=>int2,o1=>output);
END struct;
```

รูปที่ 6.7 : (b):Architecture description ของ2:1mux(structural description)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่6.7(a)เป็นโครงสร้าง(structure)ของการต่อวงจรภายในโดยที่มี สัญญาณ int0,int1และ int2เป็นสัญญาณภายใน รูปที่6.7(b)เป็น architecture ของVHDL model ที่เขียนกำหนดการเชื่อมต่อภายในด้วยVHDL netlist

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

รูปที่ 6.8 : architecture description ของ2:1 mux(behavioral description)

การบรรยาย 2:1 mux ในลักษณะของ behavioral description ได้แสดงให้เห็นอีกครั้งในรูปที่6.8 ซึ่งจะเห็นได้ว่าส่วนที่เป็น architecture design unit ทั้งหมด(รูปที่6.6,6.7และ6.8)ต่างบรรยายพฤติกรรมเดียวกันและจะให้ผลลัพธ์จากการจำลองการทำงานที่เหมือนกัน

6.4 Package Design Unit

ข้อมูลต่างๆตลอดจนโปรแกรมย่อย(subprogram)ที่เป็นประโยชน์ต่อการเขียนรูปแบบบรรยายระบบดิจิตอล สามารถ เก็บไว้ในส่วนที่เรียกว่า package ได้ และข้อมูลเหล่านี้สามารถนำไปใช้ได้โดย entity design unit ,architecture design unit หรือจาก package design unit อื่นๆ ด้วยชุดคำสั่ง USE statement นอกจากนั้นสิ่งที่นิยมทำกันมากคือรูปแบบ(model)มาตรฐานต่างๆอาทิ เช่น standard component (model ของ IC ตระกูล 74xx)จะถูกเก็บไว้ใน package ที่ทุกคนสามารถเข้าถึง และนำไปใช้ได้ สิ่งที่สามารถประกาศ หรือ บรรจุได้ในpackage ได้แก่

- Subprogram
- Types
- Constants

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- Sibnals
- Aliases
- Attributes
- Component
- Disconnection Specification

โดยปกติแล้ว package จะแบ่งเป็นสองส่วนคือ

1) Package declaration

2) Package body

เนื่องจาก Package ถูกสร้างเป็นส่วนแยกต่างหากออกจากรูปแบบ (model) ที่กำลังเขียนอยู่ ฉะนั้นการที่จะนำ Package ไปใช้นั้น จะต้องมีการเชื่อมโยง หรือ อ้างอิงเสียก่อน ซึ่งในภาษาVHDL สามารถกระทำได้ด้วยชุดคำสั่ง USE statement รูปที่6.9 เป็นตัวอย่างของการเขียน Package declaration ที่มีการประกาศ TYPE,CONSTANT,COMPONENTและSIGNAL

```
PACKAGE example IS
        TYPE cd IS ('C','D');
        CONSTANT pi:REAL:=3.14159;
        COMPONENT ttl_74163 IS
                PORT (a,b:IN BIT;
                        c:OUT BIT );
        END COMPONENT;
        SIGNAL global_clock:BIT;
END example;
```

รูปที่ 6.9 : ตัวอย่างของ Package declaration

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.4.1 Package body

โครงสร้างประกอบด้วยคำสั่งต่างๆ ในรูปของคำสั่งลำดับ (sequential statement) ที่ใช้บรรยายฟังก์ชันการทำงานของโปรแกรมย่อย (subprogram)ทั้งหลายที่ชื่อของโปรแกรมย่อยนั้นๆที่ถูกประกาศไปในส่วนของ Package declaration แล้ว จะถูกเก็บไว้ใน Package body ทั้งนี้รวมทั้ง deferred constant (อันได้แก่ตัวคงที่ที่ถูกประกาศชื่อก่อนในส่วนของ declaration แต่ถูกกำหนดค่าในส่วนของbody ของ Package) ฉะนั้น ส่วน Package body จึงไม่จำเป็นต้องมี ถ้าในส่วน Package declaration ไม่มีการประกาศชื่อ (identifier) ที่เป็นโปรแกรมย่อย (subprogram) หรือ deferred constant การเขียน package body นั้นเป็นไปตามกฎเกณฑ์ที่แสดงในรูปที่6.10

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

รูปที่ 6.10 : โครงสร้างของ Package body

6.4.2 Package declaration

ส่วนที่มีความสำคัญที่สุดของ Package (มองในแง่ของการนำไปใช้จากภายนอก) ได้แก่ Package declaration เพราะจะเป็นส่วนที่กำหนดชื่อ (identifier) ของสิ่งที่ประกาศอยู่ภายใน Package สำหรับนำไปใช้ภายนอกตัวของ Package เอง ถ้าสิ่งใดๆถูกประกาศในส่วนของ Package Body แต่ไม่ถูกประกาศใน Package declaration จะไม่สามารถถูกนำค่า และพฤติกรรมไปใช้จากส่วนนอกได้ ซึ่งสามารถเปรียบเทียบได้กับสิ่งที่ประกาศไว้ในส่วนของ entity declaration คือ interface ที่มีหน้าที่ติดต่อกับโลกภายนอก ฉะนั้นโดยทั่วไปแล้ว Package สามารถ สร้างขึ้นได้โดยไม่จำเป็นต้องมีส่วน body และยังสามารถถูกนำไปใช้จากรูปแบบ(model)ภายนอกได้เช่นใช้สำหรับประกาศ TYPE หรือ signal(global)เช่นเดียวกันกับ Package body ที่ไม่จำเป็นต้องมี Package declaration แต่ Package นั้นจะไม่สามารถถูกนำไปใช้จากรูปแบบ (Model) อื่นได้ การเขียน Package declaration มีกฎเกณฑ์ตามที่แสดงในรูปที่6.11

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
PACKAGE  package_name IS
        package_declaration_ part
END package_name;
```

รูปที่ 6.11:โครงสร้างของ Package declaration

คำว่าPACKAGE และ END PACKAGE กำหนดขอบเขตของ Package declarationระหว่างนั้น จะเป็นส่วนที่ใช้ประกาศต่างๆสิ่งที่สามารถประกาศในส่วนนี้ได้แก่

- ส่วนประกาศกำหนดโปรแกรมย่อย (Subprogram declarations)
- Type declaration
- Subtype declarations
- Object declarations (signals constants)
- Alias declarations
- Attribute specifications
- Component specifications
- Disconnection specification

ชื่อของ Package_name ที่ใช้ใน package body จะต้องเป็นชื่อเดียวกับชื่อที่กำหนดไว้ใน package Declaration ในรูปที่6.12 แสดง ตัวอย่างของการเขียน package (declaration และ body ที่สัมพันธ์กัน) โดยนำการกำหนดโปรแกรมย่อย (subprogram)ประเภท FUNTION

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
-- Package declaration
PACKAGE pack_funct IS
FUNCTION mean (a,b,c:REAL)RETURN REAL;
END pack_funct;
-- Package body
PACKAGE BODY pack_funct IS
        FUNCTION mean (a,b,c:REAL)RETURN
REAL IS
        BEGIN
        RETURN(a+b+c)/3.0;
        END mean;
END pack_funct;
```

รูปที่ 6.12 : ตัวอย่างการเขียน package

ในส่วนของ package declaration จะบรรจุส่วนที่เรียกว่า function declaration ในที่นี้เป็นการประกาศชื่อของโปรแกรมย่อย (FUNCTION) mean และส่วนที่บรรยายการทำงานของโปรแกรมย่อย mean (เรียกว่า function body) จะถูกเก็บไว้ในส่วน package body แต่สิ่งที่สำคัญอย่างหนึ่งของการเขียน package คือ ก่อนที่จะนำ package ไปใช้ (โดยการอ้างจากภายนอก) package นั้นจะต้องถูกวิเคราะห์เสียก่อนว่าถูกต้อง หรือ พูดง่ายๆได้ว่าจะต้องผ่านการ compile ก่อนนั้นเอง

6.5 Configuration Design Unit

ดังที่ทราบกันแล้วว่ารูปแบบ หนึ่งของระบบดิจิตอลไม่ว่าจะเป็นอะไร จะมี entity design unit ได้เพียงหน่วยเดียวเท่านั้น แต่ในขณะที่ entity design unit หนึ่งหน่วยนี้อาจจะมี architecture ที่เป็นหน่วยรองได้หลายหน่วย

ดังนั้นจึงเกิดคำถามขึ้นว่า ในการจำลองการทำงานของรูปแบบ (model)นั้น simulator จะนำ architecture อันไหนไปจำลอง? คำตอบของคำถามนี้คือ ต้องบอกให้ simulator ทราบ และในรูปแบบ VHDL นั้นการบอกหรือกำหนดคือ การใช้ CONFIGURATION ประกอบ entity กับ architecture design unit ที่ต้องการเข้าด้วยกัน รูปที่ 6.13 แสดงกฎเกณฑ์การเขียนconfiguration

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

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

รูปที่ 6.13 : โครงสร้างของconfiguration

ในรูปที่ 6.14 เป็นรูปแบบ(model)ง่ายๆของ AND-gate 2 input ที่มีส่วนรองอันได้แก่ architecture

สองแบบคือ dataflow description และ behavioral description

```
ENTITY and2 IS
        GENERIC (ttl_delay: TIME := 3NS);
        PORT(in1,in2:IN BIT;
                Output: OUT BIT);
END and2;
ARCHITECTURE dataflow OF          and2 IS
BEGIN
        Output<=in1 AND in2  AFTER ttl_delay;
END dataflow;
ARCHITECTURE behave OF and2 IS
BEGIN
        PROCESS(in1,in2)
        BEGIN
                IF(in1= '1' AND in2= '1') THEN
                        Output<= '1' AFTER ttl_delay;
                ELSE
                output<= '0';
        END PROCESS;
END behave;
```

รูปที่ 6.14: VHDL model ของAND-gate 2 inputs

ก่อนที่จะนำรูปแบบ (model)ไปจำลองการทำงานจะต้องมีการประกอบ architecture ที่ต้องการเข้ากับ entity design unit เสียก่อน (configuration) ซึ่งระบบVHDL ส่วนใหญ่ถ้าไม่กำหนดการประกอบ architecture เครื่อง simulator จะนำ architecture หน่วยสุดท้ายที่ผ่านการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

วิเคราะห์ไปใช้จำลองการทำงาน ในรูปที่6.15 แสดงการประกอบ architecture ชื่อ dataflow เข้ากับ entity design unit

```
CONFIGURATION dataflow_ and OF and2
IS
        FOR dataflow
        END FOR;
END dataflow_and;
```

รูปที่ 6.15 : Configuration ของรูปแบบ (model) and 2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 7

เทคโนโลยีของ FPGA

เนื่องจากเป็นลักษณะของชิพที่สามารถโปรแกรมได้นั่นก็คือ สามารถกำหนดจุดเชื่อมต่อต่างๆภายในได้ เพื่อประกอบเป็นลักษณะของวงจรตามที่เราต้องการได้ ซึ่งเราสามารถแบ่งลักษณะของจุดเชื่อมต่อต่างๆได้ดังนี้

7.1 Physical Changing

7.1.1 Fused สามารถโปรแกรมได้เพียงครั้งเดียว หลังจากโปรแกรมจุดเชื่อมต่อขาดจากกัน

7.1.2 Anti Fuse สามารถโปรแกรมได้เพียงครั้งเดียว หลังจากโปรแกรม จุดเชื่อมต่อจะเชื่อมถึงกัน

7.2 Memory Base

7.2.1 EEPROM – Base FPGA

มักเรียก FPGA ประเภทนี้ว่า CPLD จะใช้เทคโนโลยีเหมือนกับ EEPROM ในการโปรแกรม ซึ่งจะทำให้มีความจุของเกตต่ำ โดยทั่วๆไปจะน้อยกว่า 20,000 เกต แต่ข้อดีของ EEPROM-Base FPGA คือสามารถเก็บข้อมูลที่โปรแกรมลงไปได้โดยไม่จำเป็นต้องมีไฟเลี้ยง และในการโปรแกรม จะใช้ทรานซิสเตอร์ 1 ตัวต่อ 1 บิท สามารถโปรแกรมได้ประมาณ 10,000 ครั้ง มักจะมีการจัดสถาปัตยกรรมในรูปแบบอาร์เรย์ ใช้ AND- OR Plane ในการทำลอจิกฟังก์ชัน

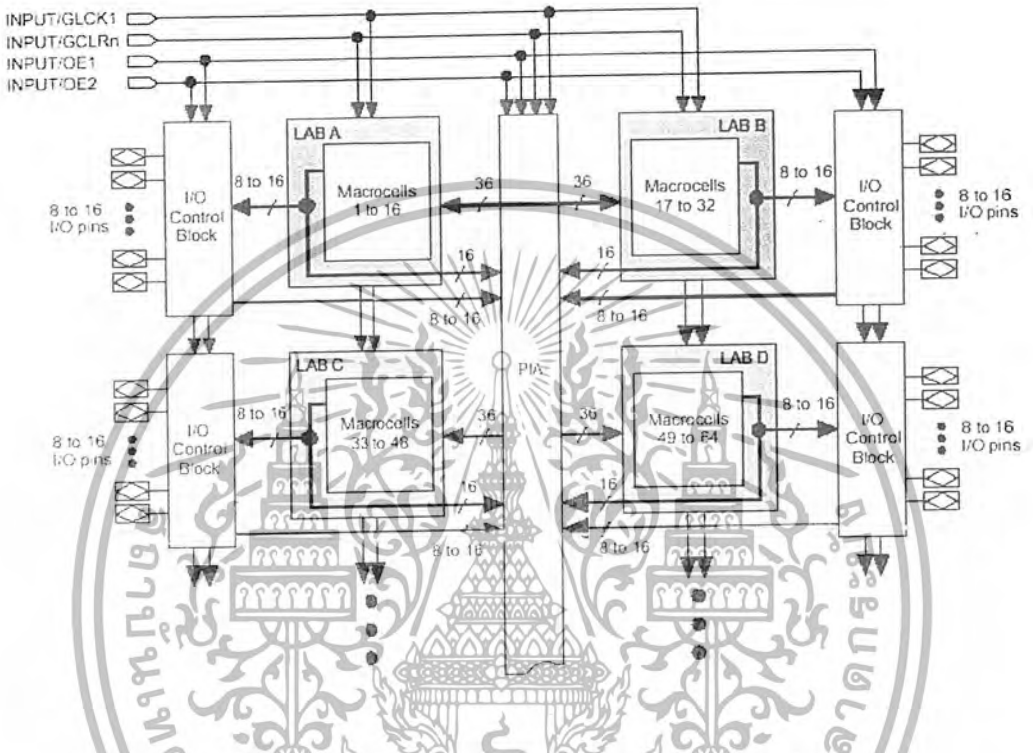
7.2.2 SRAM – Base FPGA

จะใช้เทคโนโลยีเหมือน SRAM ในการโปรแกรม ซึ่งจะสามารถทำให้โปรแกรมซ้ำได้ไม่จำกัดจำนวนครั้ง มีความจุของเกตปานกลางถึงสูงมาก (ประมาณ 10,000 – 1,000,000 เกต) จะใช้ Look-Up Table ในการทำลอจิกฟังก์ชัน (Logic Function) และจะมีการจัดทรัพยากรภายในโครงสร้างแบบอาร์เรย์ ข้อดีของ SRAM – Base FPGA คือจะใช้เวลาในการโปรแกรมน้อย (ในระดับ mS) การโปรแกรมจะทำได้ง่ายเทียบเท่ากับการเขียน SRAM ทั่วไป และไม่จำกัดจำนวนครั้งในกระบวนการผลิตจะทำได้ง่ายและเหมาะสมสำหรับการออกแบบวงจรที่มีความสลับซับซ้อน ข้อเสียก็คือไม่สามารถเก็บโปรแกรมในภาวะที่ไม่มีไฟเลี้ยงได้ มักจะใช้ FPGA ชนิดนี้ควบคู่กับ ROM เพื่อเก็บโปรแกรมและจะโหลดโปรแกรมเข้าในตัวชิพเมื่อเริ่มต้นใช้งาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

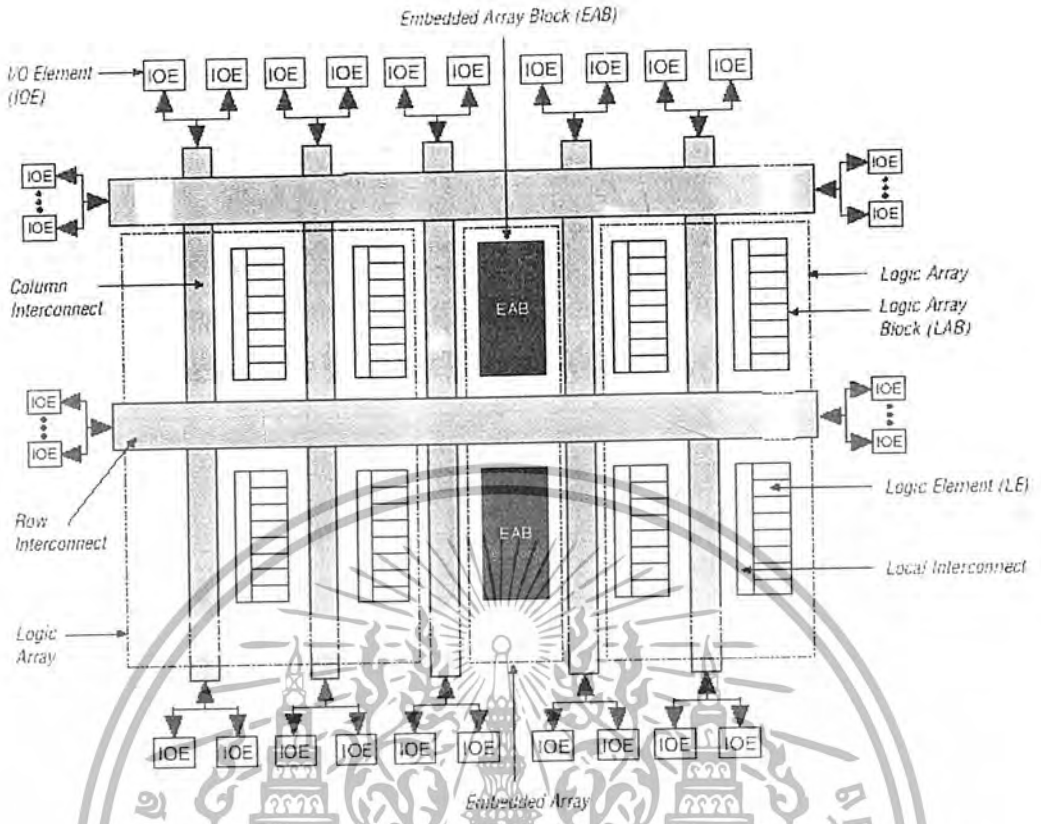
7.3 โครงสร้างภายใน

ลักษณะโครงสร้างภายใน เป็นอาร์เรย์ลอจิกบ็อกซ์ที่สามารถทำการโปรแกรมได้ดังแสดงในรูปที่ 7.1



รูปที่ 7.1 โครงสร้างภายในของ FPGA ตระกูล MAX7000S

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 7.2 โครงสร้างภายในของ FPGA ตระกูล FLEX10K

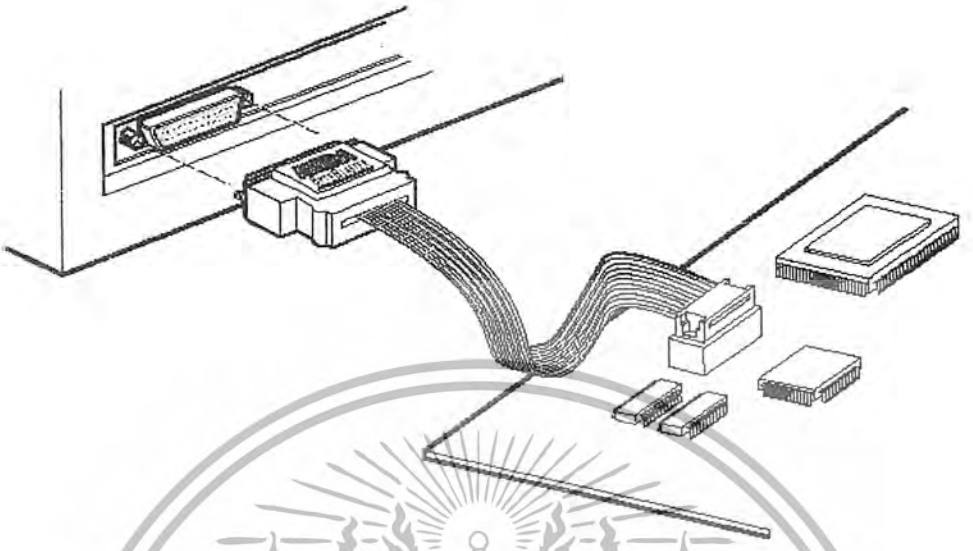
7.4 ทำไมการออกแบบถึงทำได้ง่ายและสะดวกรวดเร็ว

7.4.1 ในการออกแบบเราไม่จำเป็นต้องรู้ถึงโครงสร้างภายในของตัวชิพ เพียงแต่รู้ขั้นตอนการออกแบบลอจิกก็พอ ไม่เหมือนไมโครโปรเซสเซอร์ที่เราจำเป็นต้องรู้โครงสร้างภายในรวมถึงการศึกษาการเขียนภาษา Assembly ซึ่งแต่ละตัวก็ไม่เหมือนกันด้วย

7.4.2 การใช้ภาษาในการอธิบายการทำงานของวงจร ที่เรียกว่า **HDL (Hardware Description Language)** จะช่วยได้มากสำหรับการออกแบบ เนื่องจากเป็นวิธีการที่มีความยืดหยุ่นสูง ทำได้เร็ว และไม่จำเป็นต้องรู้ลักษณะของวงจรที่จะออกแบบว่าต่อกันอย่างไร เพียงแต่กำหนดลักษณะการทำงานให้มัน และตัวซอฟท์แวร์จะทำ Synthesis and Optimize ให้เราเอง นอกจากนี้ภาษาที่ใช้ ยังเป็นมาตรฐานเดียวกัน สามารถใช้ได้กับชิพทุกตัวและทุกบริษัท

7.4.3 การโปรแกรมสามารถทำได้เองและใช้เวลาไม่นาน โดยเพียงแค่ส่งข้อมูลผ่านสายดาวโหลดทางพอร์ตของคอมพิวเตอร์ก็สามารถโปรแกรมตัวชิพได้ขณะที่เอยู่ในระบบโดยไม่จำเป็นต้องถอดมาโปรแกรมข้างนอก ดังรูปที่ 7.3 และ ที่สำคัญสามารถโปรแกรมได้หลายครั้ง จึงทำให้ง่ายในการแก้ไขและพัฒนาโดยที่ไม่ต้องเสียค่าใช้จ่ายเพิ่มแต่อย่างใด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 7.3 การโปรแกรมลงในชิพ

7.5 การออกแบบโดยใช้ภาษาอธิบายพฤติกรรมของฮาร์ดแวร์ (HDL)

ในการออกแบบวงจรดิจิตอลนั้นสามารถทำได้โดยการวาดวงจร (Schematic) หรือใช้ภาษาอธิบายพฤติกรรม (Hardware Description Language) ของฮาร์ดแวร์ ในกรณีของการออกแบบวงจรด้วย ASIC เราจะต้องเขียนวงจรด้วย Schematic แล้วนำวงจรนั้นไป Simulate หากผลออกมาเป็นที่พอใจก็จะต้อง Layout เป็นชั้นสาร และในการออกแบบ ASIC จำเป็นจะต้องรู้ว่าจะใช้เทคโนโลยีอะไรเช่นที่ NECTEC ใช้อยู่ จะใช้เทคโนโลยีของ ALCATEL 0.5 um. เมื่อได้ชั้น Layout เสร็จสมบูรณ์ จึงจะส่งไป Fabrication เป็นชิพไอซี แล้วจึงจะนำมาใช้งานได้ แต่ในการออกแบบวงจรด้วย FPGA โดยการใช้ Schematic หรือใช้ภาษาอธิบายการทำงานของวงจรจะทำได้สะดวกกว่า เพราะการทำวิธีนี้ ผู้ออกแบบไม่ต้องคำนึงถึงเทคโนโลยีที่จะใช้และที่สำคัญ การออกแบบโดยวิธีนี้สามารถแก้ไขโมเดล (Model) หรือเปลี่ยนแปลงเทคโนโลยีได้สะดวกกว่า เพราะไม่ต้องวาดวงจรใหม่ นั่นคือการออกแบบโดยใช้ภาษาอธิบายฮาร์ดแวร์ จะทำให้โมเดลที่ได้ไม่ขึ้นกับเทคโนโลยี สำหรับภาษาที่ใช้สำหรับอธิบายพฤติกรรมของฮาร์ดแวร์ที่ใช้กันก็มี VHDL, AHDL, Verilog สำหรับในการทดลองนี้จะให้นักศึกษาใช้การวาด **Schematic** ส่วน VHDL นั้นจะมีใน Lab ต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

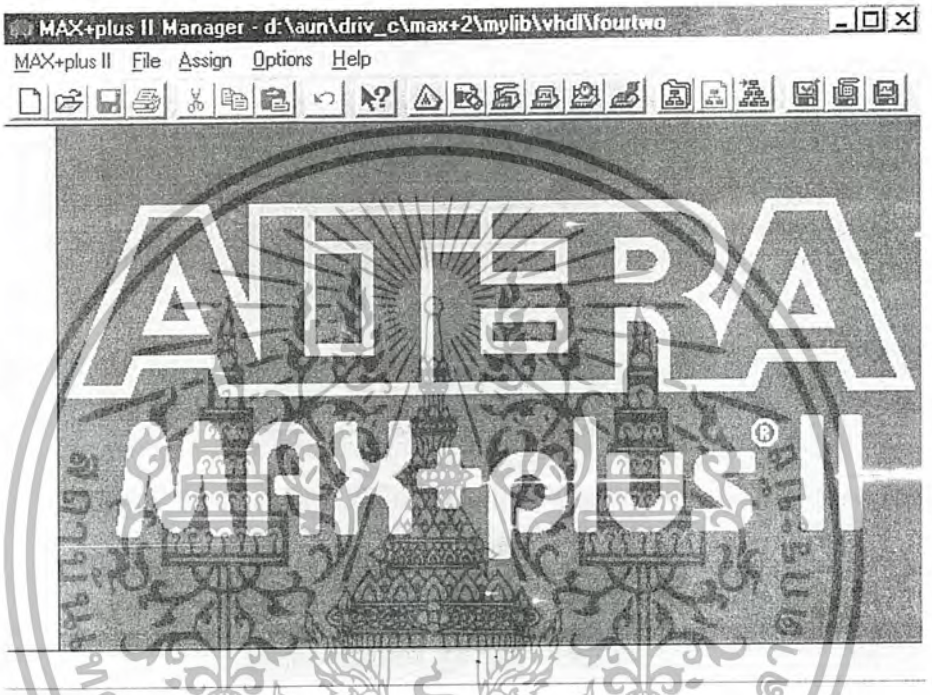
7.6 การสังเคราะห์วงจร (Logic Synthesis)

ในขั้นตอนนี้จะใช้ซอร์พแวร์ในการสังเคราะห์วงจร (Synthesis Tools) ทำการสังเคราะห์พฤติกรรมของวงจรที่ได้จากการออกแบบด้วย Schematic หรือ VHDL ซึ่งต้องทำการตรวจสอบด้วยว่าซอร์พแวร์นั้นสนับสนุนเทคโนโลยี FPGA (FPGA Library) ที่ต้องการหรือไม่ ตัวอย่างเช่น FPGA ของบริษัท XILINX และบริษัท ALTERA จะมีซอร์พแวร์หลายตัวที่สามารถใช้ได้ เช่น **Max+Plus II** ในขั้นตอนนี้ซอร์พแวร์สังเคราะห์วงจรจะทำการแปลงโค้ด VHDL และทำการ Optimize เพื่อให้ได้วงจรตามเทคโนโลยีที่เลือกใช้ ในการสังเคราะห์วงจรนั้นวงจรระดับเกต (Gate Level) จะไม่เหมาะสมกับโครงสร้างที่มีอยู่ในอุปกรณ์ FPGA ดังนั้นในการ Optimize ซอร์พแวร์สังเคราะห์วงจร จะต้องทำการ Optimize ให้ได้เป็นวงจรที่ประกอบด้วยกลุ่มของลอจิกที่เหมาะสมกับอุปกรณ์ FPGA นั้นๆ จึงทำให้ผลที่ได้มีประสิทธิภาพและในขั้นตอนการสังเคราะห์วงจรนี้ผู้ออกแบบสามารถกำหนดข้อบังคับสำหรับโมเดลแต่ละตัวได้ เช่น ข้อบังคับในเรื่องเวลา (Timing Constraints) หรือข้อบังคับในเรื่องของพื้นที่ (Area) หรือกำหนดชนิดและตำแหน่งของ I/O ซึ่งข้อบังคับเหล่านี้จะถูกนำไปใช้ในขั้นตอน Optimize เพื่อให้วงจรที่ได้เป็นไปตามที่กำหนด ส่วนสำคัญในการ Optimize คือการเทียบ (Mapping) โมเดลให้เข้ากับเทคโนโลยีที่ใช้เพื่อให้ได้วงจรที่เหมาะสมกับโครงสร้างและสถาปัตยกรรมภายในอุปกรณ์ FPGA เมื่อทำการสังเคราะห์วงจรเสร็จแล้ว ซอร์พแวร์ การสังเคราะห์วงจรก็จะมีการรายงานผลว่าโมเดลที่ออกแบบไปนั้นเป็นอย่างไร เช่นมีค่าความหน่วง (Delay) เท่าไหร่ ใช้ทรัพยากรต่างๆใน FPGA อะไรบ้าง เมื่อมาถึงขั้นตอนนี้ผู้ออกแบบก็จะทราบว่าโมเดลเป็นไปตามข้อบังคับหรือไม่ ถ้าไม่ก็สังเคราะห์ใหม่จนกว่าจะเป็นไปตามที่กำหนด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 8
การทดลองที่ 1 การใช้งานโปรแกรม Max Plus II เบื้องต้น

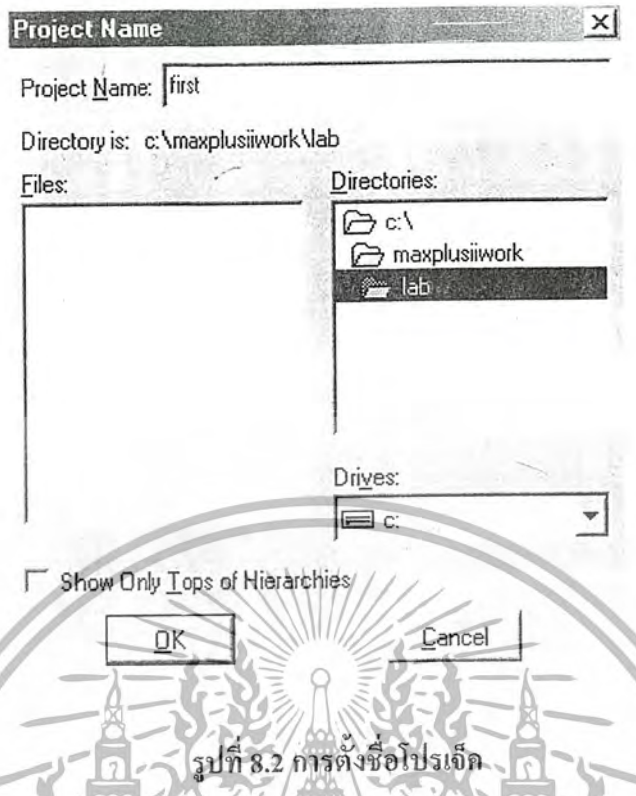
8.1 Run โปรแกรม Max+Plus II 9.5 BASELINE ขึ้นมา



รูปที่ 8.1 Run โปรแกรม Max+Plus II 9.5 BASELINE

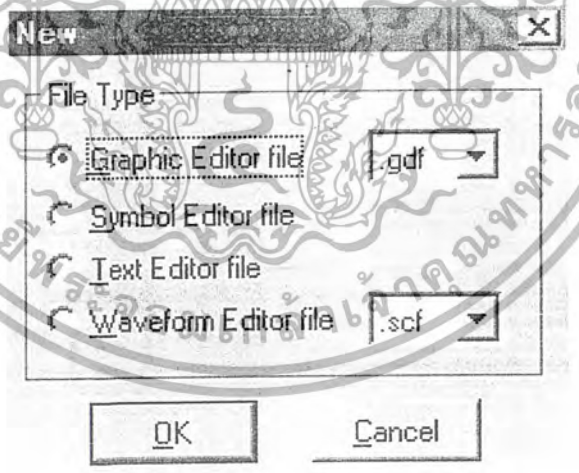
จากนั้นทำการตั้งชื่อของโปรเจ็ค ที่เราจะสร้างขึ้นมา ชื่อของไฟล์ทุกไฟล์ที่อยู่ในโปรเจ็คเดียวกันจะมีชื่อที่เหมือนกันแตกต่างกันที่นามสกุล การตั้งชื่อโปรเจ็คเริ่มจาก **File / Project / Name** เลือก Folder ที่จะเก็บโปรเจ็คของเราและใส่ชื่อของโปรเจ็คเป็น **First**

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 8.2 การตั้งชื่อโปรเจ็ค

8.2 เลือกประเภทของไฟล์ที่จะสร้าง File / New / Graphic Editor File



รูปที่ 8.3 เลือกประเภทของไฟล์ที่จะสร้าง File / New / Graphic Editor File

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โดยที่

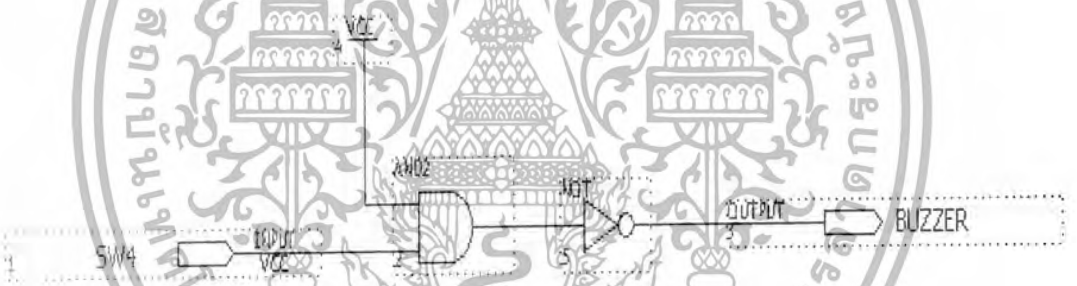
Graphic Editor File: เป็นไฟล์กราฟฟิกส์ที่เราสามารถนำอุปกรณ์ต่างๆใน Library มาวางต่อกันได้เลย

Symbol Editor File: เป็นไฟล์ที่ใช้เก็บสัญลักษณ์เพื่อสื่อให้รู้ว่าโมเดลที่เราได้สร้างขึ้นมีอินพุท, เอาท์พุทเป็นอย่างไร

Text Editor File: เป็น Text file ใช้สำหรับเขียน Source code เพื่ออธิบายพฤติกรรมของวงจรหรือโมเดลต่างๆที่เราจะสร้างขึ้นหรือเพื่อไว้สำหรับเก็บข้อความทั่วๆไป

Waveform Editor File: เป็นไฟล์ไว้สำหรับการกำหนดรูปแบบของสัญญาณอินพุทเพื่อใช้ในการ Simulate และไว้สำหรับให้ Max Plus II แสดงผลของเอาท์พุทที่ได้จากการ Simulate

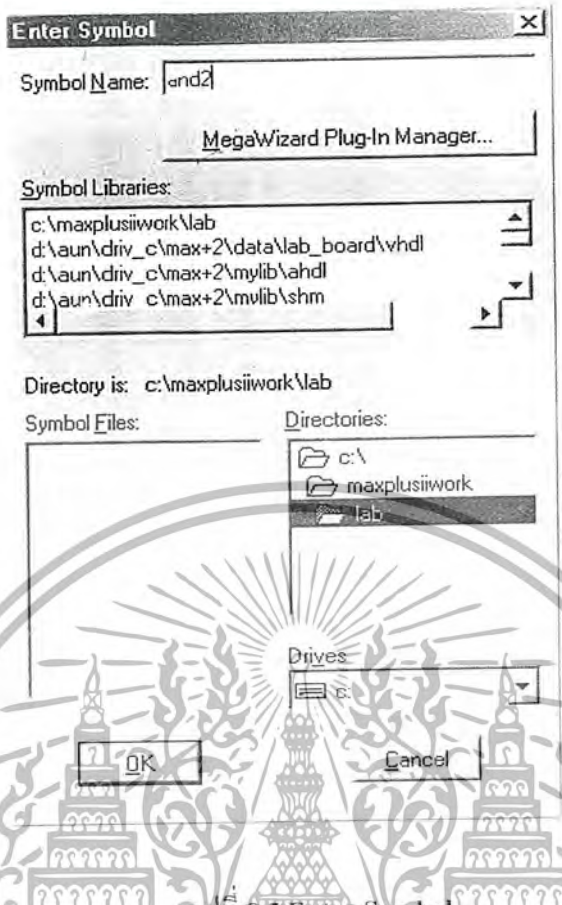
3.ในการอธิบายการใช้งานโปรแกรม Max Plus II นี้เราจะใช้วงจรดังรูปที่11 ประกอบคำอธิบาย



รูปที่ 8.4 ตัวอย่างการวาดวงจร

จากรูปที่ 8.4 AND GATE, NOT GATE, INPUT, OUTPUT, VCC จะเป็นอุปกรณ์ที่มาพร้อมกับโปรแกรม Max Plus II ในการที่จะเอาอุปกรณ์ต่างๆ ใน Library ของ Max Plus II มาใช้ให้ทำการ Double Click ที่ Graphic Form จะมี Dialog Enter Symbol ปรากฏขึ้นมา ให้เลือกอุปกรณ์ที่เราต้องการจะใช้งานขึ้นมาวางไว้ที่ Graphic Form

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



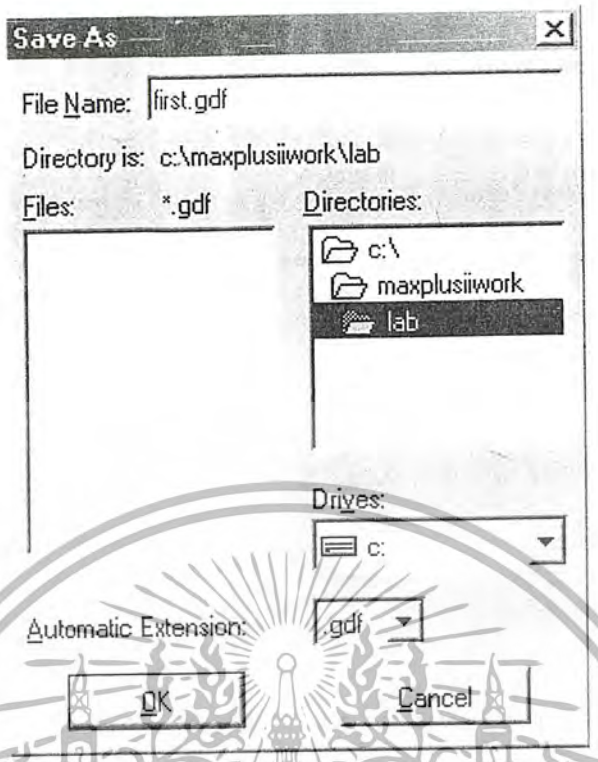
รูปที่ 8.5 Enter Symbol

โดยที่

- AND 2 input จะใช้ชื่อ Symbol Name ว่า **and2**
- NOT 1 input จะใช้ชื่อ Symbol Name ว่า **not**
- Input จะใช้ชื่อ Symbol Name ว่า **input**
- Output จะชื่อ Symbol Name ว่า **output**
- VCC จะใช้ชื่อ Symbol Name ว่า **vcc**

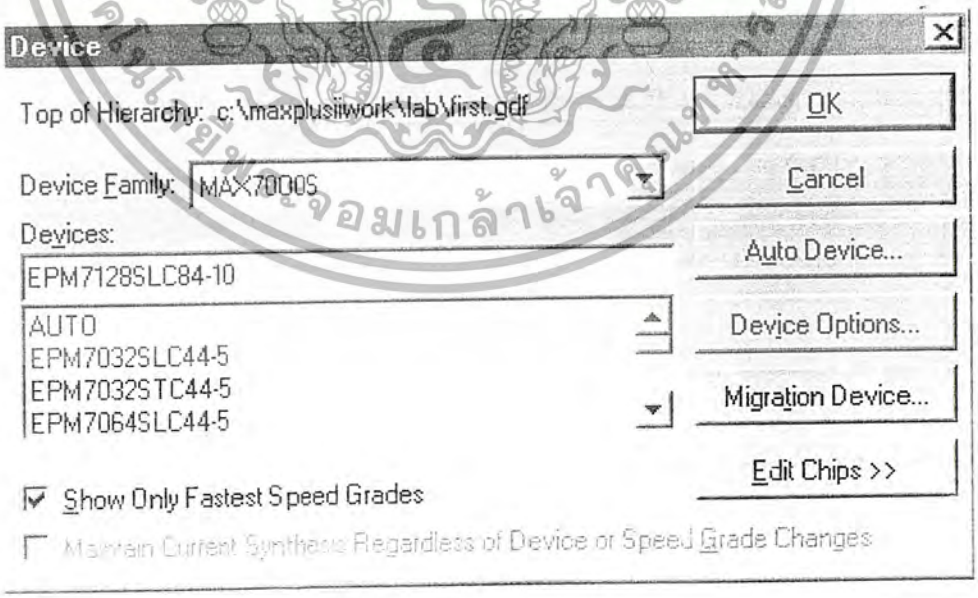
อุปกรณ์ทั้งหมดจะอยู่ใน Symbol Library ชื่อ **...\maxplusii\max2lib\primg** เมื่อทำการวางอุปกรณ์จนครบทุกตัวแล้ว ให้ทำการเชื่อมต่ออุปกรณ์ต่างๆเข้าด้วยกันโดยเลื่อน pointer ของ mouse ไปที่ขาต่างๆของอุปกรณ์ pointer ของ mouse จะเปลี่ยนเป็นเครื่องหมาย + ให้ทำการ Click ขวาค้างไว้แล้วลากเส้นไปต่อกับ input หรือ output ของอุปกรณ์ตัวอื่น เมื่อเชื่อมต่ออุปกรณ์ต่างๆจนครบทั้งหมดแล้วให้ทำการเปลี่ยน PIN_NAME ของอุปกรณ์ input กับ output โดยการไป **Double Click** ที่ PIN_NAME ของ input แล้วเปลี่ยนชื่อเป็น **SW1** สำหรับ PIN_NAME ของ output เปลี่ยนเป็น **BUZZER** เมื่อวาดรูปวงจรเสร็จเรียบร้อยแล้วให้ทำการบันทึกวงจรที่เราสร้างขึ้นมา **File / Save as** ให้ใส่ชื่อไฟล์เป็น **first.gdf** แล้วก็ OK

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 8.6 การ Save

8.3 ทำการระบุเบอร์ของชิพที่เราจะใช้ Assign / Device จะมีไดอะล็อก Device ปรากฏขึ้นมา ให้ทำการเลือก Device Family เป็น MAX700S และเลือก Device เป็น EPM7128SLC84-10

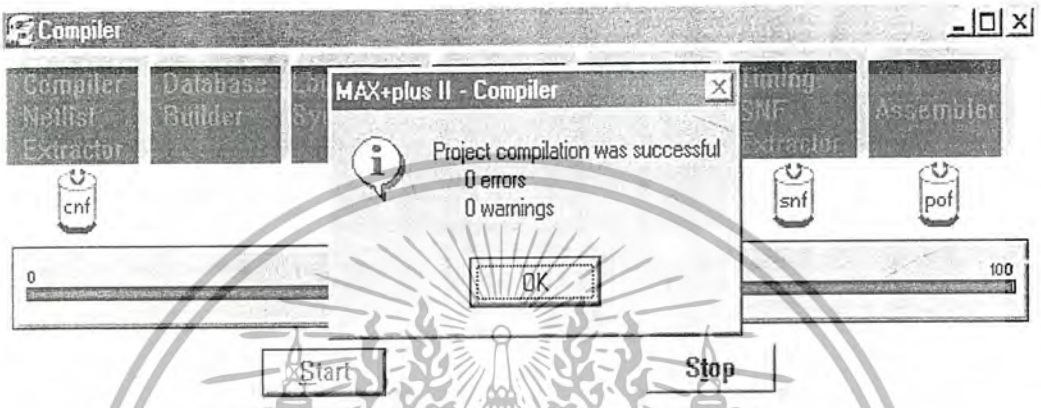


รูปที่ 8.7การระบุเบอร์ของชิพที่เราจะใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

8.4 ทำการคอมไพล์วงจรที่เราได้สร้างขึ้นมา Max+Plus II / Compiler / Start

มื่อคอมไพล์เสร็จจะมีหน้าต่างรายงานผลการคอมไพล์ error กับ warring หากมีความผิดพลาดเกิดขึ้นจะมีข้อความสีแดงบอกว่า error และจะบอกด้วยว่า error เพราะอะไร

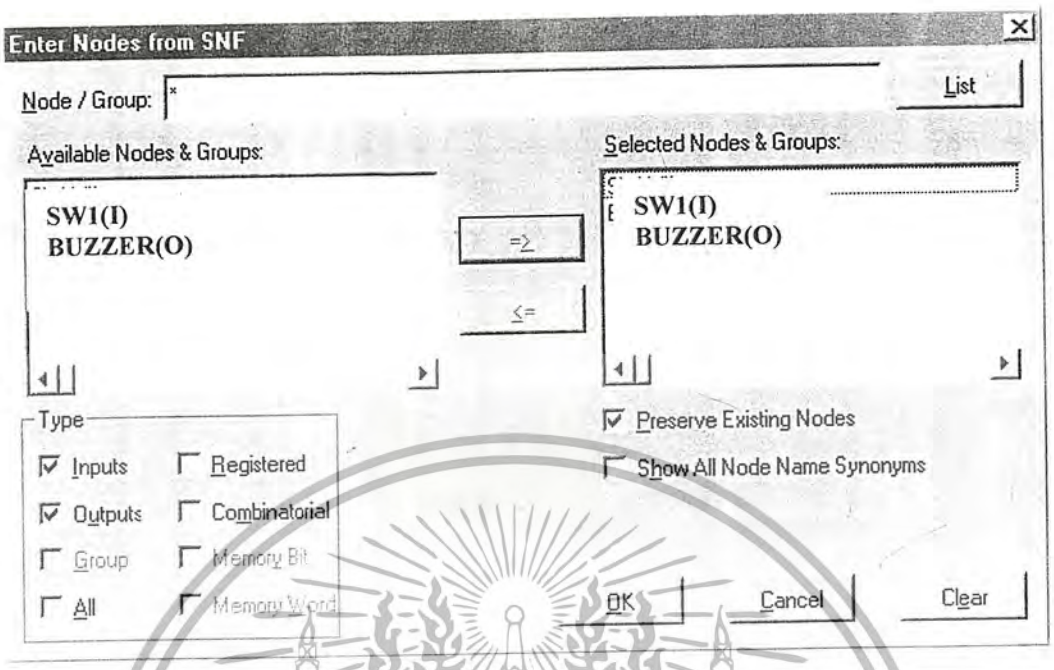


รูปที่ 8.8การคอมไพล์วงจรที่เราได้สร้างขึ้นมา

8.5 ทำการ Simulate

โดยจะต้องสร้างไฟล์ Waveform ขึ้นมาก่อน **File / New / Waveform Editer file** (จะได้หน้าต่างดังรูปที่ 8.9 ก่อนนี่จะทำการ Simulate จะต้องทำการกำหนดอินพุทให้แก่วงจรก่อน โดยโหลด Noad ต่างๆ เข้ามา **Noad / Enter Noad from SNF** จะมีไดอะล็อก Enter Nodes from SNF ปรากฏขึ้นมาให้คลิกที่ **List** จะมี Noad ต่างๆ ที่อยู่ในวงจรปรากฏขึ้นมาให้เราเลือก Noad ที่เป็น input กับ output ดังรูปที่8.8 โดยการกดปุ่มที่มี เครื่องหมายลูกศรชี้ไปทางขวา

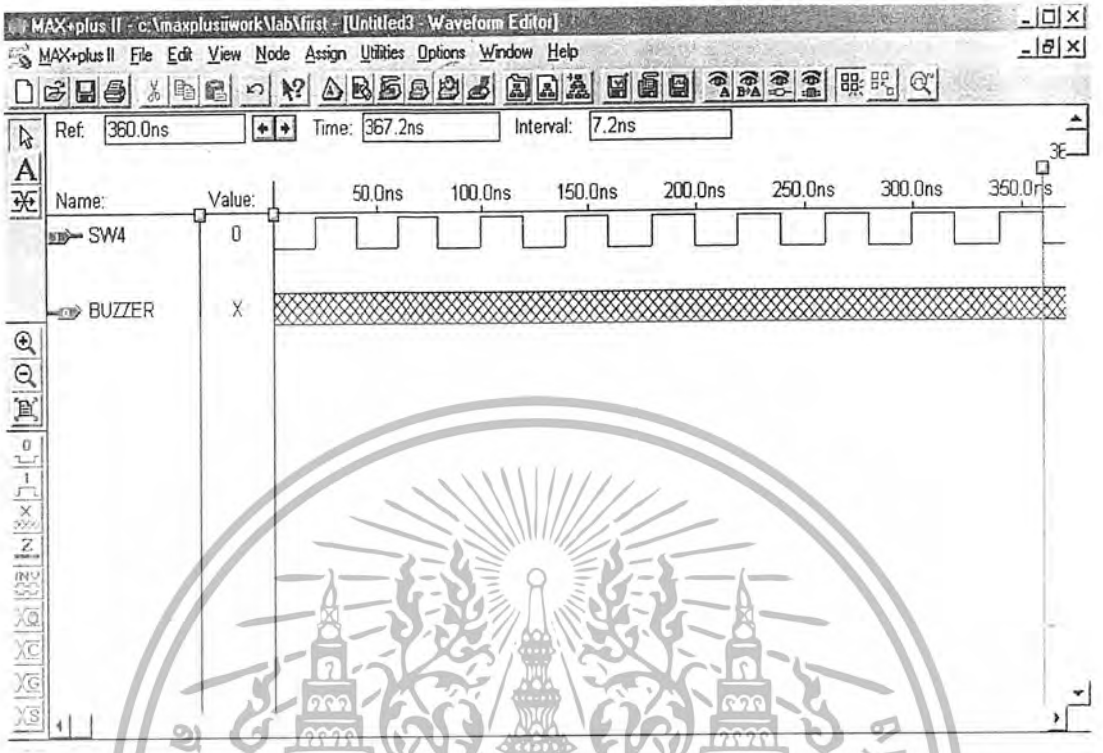
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 8.9 การ Simulate

- กำหนดเวลาสิ้นสุดการ Simulate (End Time) ให้กับตัวโปรแกรม **File / End Time** ให้ค่า End Time เท่ากับ **1.0 us** ซึ่งจะเป็นการบอกว่าให้โปรแกรมทำการ Simulate ตั้งแต่ 0.0 us - 1.0 us
- กำหนดขนาดของกริด **Option / Gride Size** กำหนดให้กริดมีขนาดเท่ากับ **10.00ns**
- ทำการกำหนดรูปแบบสัญญาณให้กับ Node Input โดยที่ คลิกที่ **SW1** แถบดำปรากฏขึ้นมาหลังจากนั้นทำการกำหนดรูปแบบสัญญาณให้มีลักษณะเป็นพัลซ์ **Edit / Over write / Clock** ในช่อง **Multiply By** ให้ใส่ **2** แล้วคลิก OK ซึ่งจะเป็นการกำหนดให้สัญญาณในช่วง Logic 1 และ Logic 0 มีค่าเวลาเป็น 2 เท่าของกริด

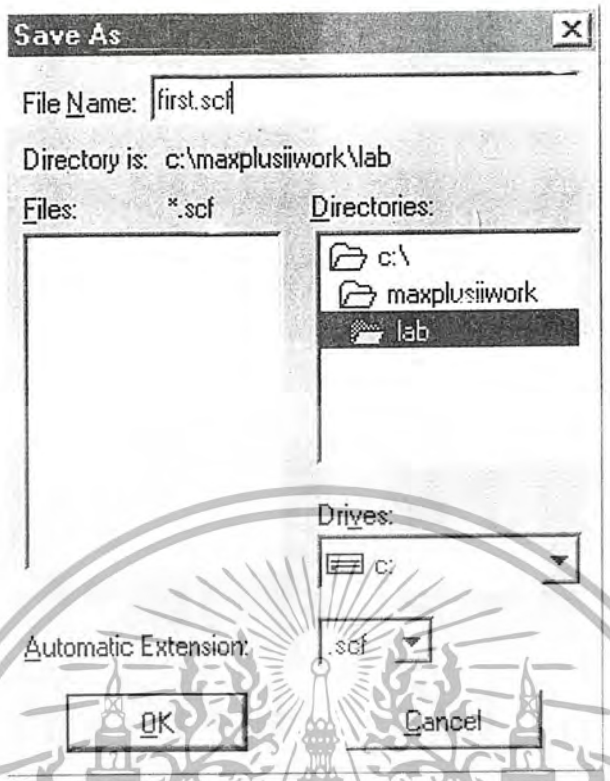
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 8.10 สัญญาณที่ได้จากการ Simulate

- บันทึกไฟล์ Waveform ที่ได้สร้างขึ้น **File / Save as** กำหนดให้ชื่อไฟล์ที่จะบันทึกเป็น **First.scf**

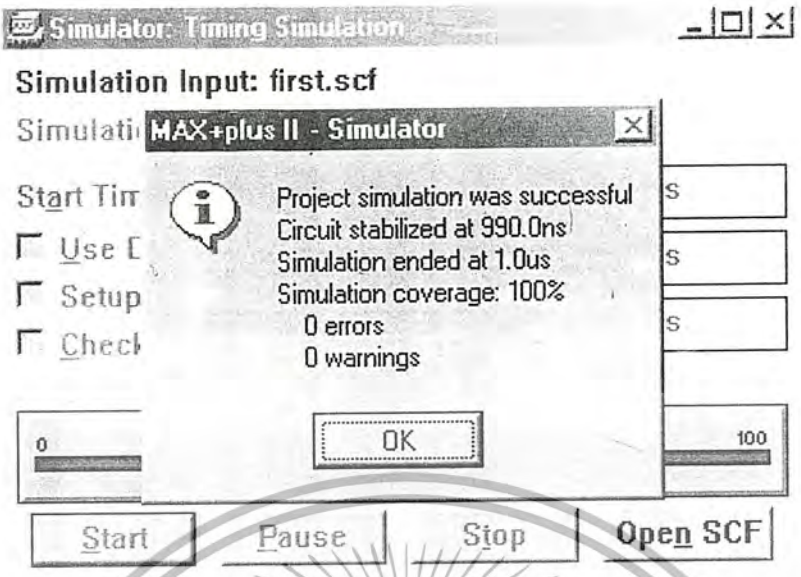
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 8.11 การ Save แบบเปลี่ยนชื่อ File

- ทำการจำลองการทำงานของวงจร Max+Plus II / **Simulation** เมื่อ Simulate เสร็จจะมีไดอะล็อกขึ้นมารายงานผลการ Simulate ว่ามี error หรือ warring หรือไม่ และเราสามารถดูผลการ Simulate ได้ที่ Form ของ Waveform Editor พร้อมทั้งวาด **Waveform** ต่างๆไว้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 8.12 หน้าต่าง Simulate

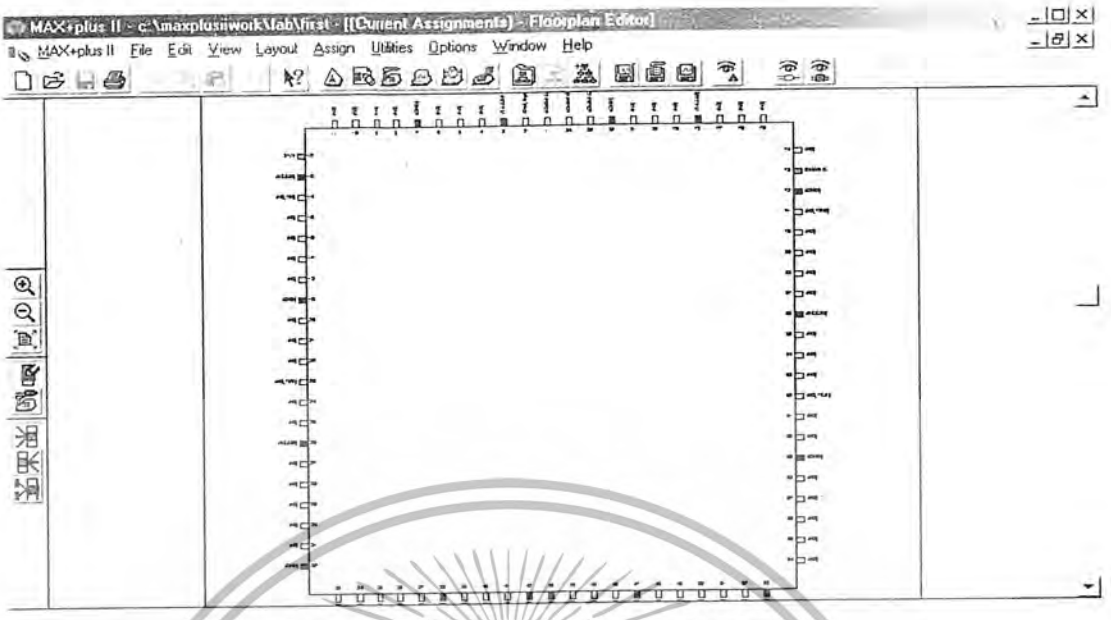
8.6 วิเคราะห์ Timing Analyzer

จะใช้สำหรับหาค่า Delay Time ระหว่าง Noad ต่างๆ Max+Plus II / Timing Analysis จะมี ไดอะล็อก Timing Analyzer ปรากฏขึ้นมาให้เลือกชนิดของการวิเคราะห์ ให้ทำการเลือกวิเคราะห์

ค่าเวลาหน่วง Analysis / Delay Time

รูปที่ 8.13 วิเคราะห์ Timing Analyzer

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 8.14 เมื่อทำการเปลี่ยนแปลงตำแหน่งขาของชิพเรียบร้อยแล้วให้ save และ คอมไพล์ใหม่อีกครั้งProject / Save Compile & Simulate

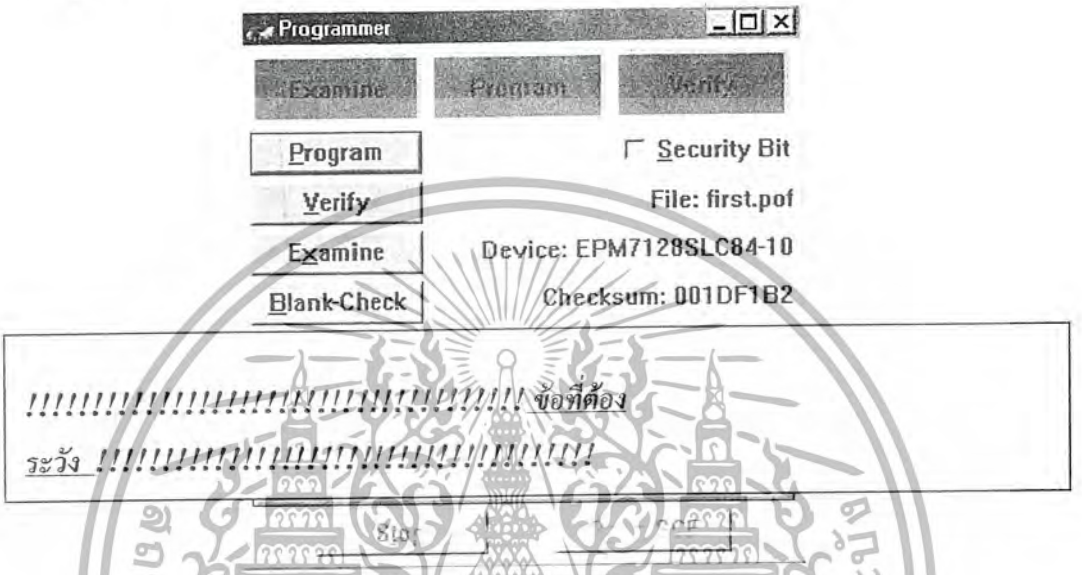
8.7 การเปลี่ยนแปลงตำแหน่งขาของชิพ

เราสามารถเปลี่ยนแปลงตำแหน่งขาของชิพโดยคลิกที่ **Max+Plus / Floorplan Editor** จะปรากฏฟอร์มของ Floorplan Editor ขึ้นมาเลือกที่เมนู **Assign / Back-Annotate Project** คลิกที่ **Chip, Pin & Device / OK** สังเกตว่าเมื่อผ่านขั้นตอนนี้เราจะเห็นตำแหน่งขาที่โปรแกรม Max+Plus II ได้เลือกให้เรา ที่นี้เราสามารถที่จะเปลี่ยนแปลงตำแหน่งขาต่างๆเหล่านี้ได้เพื่อให้เหมาะสมกับบอร์ดทดลองของเรา โดยคลิกที่ **Layout / Current Assignment Foorplan** ให้ทำการเปลี่ยนตำแหน่งขาของ **sw1 เป็นขาที่ 49** และตำแหน่งของขาของ **Buzzer เป็นขาที่ 10** โดยการคลิกขวาที่ขา sw1 ค้างไว้แล้วลากมาที่ขา 49 การเปลี่ยนแปลงตำแหน่งของขา Buzzer ก็ให้ทำเช่นเดียวกัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

8.8 การโปรแกรมวงจรลงในชิพ Max+Plus II / Programmer / Program

ไฟล์ข้อมูลที่จะโปรแกรมลงในชิพหากเลือก Device ของ FPGA เป็นอุปกรณ์ประเภท EEPROM – Base FPGA หรือที่เรียกว่า CPLD (เบอร์ IC จะขึ้นต้นด้วย EPM) ไฟล์ที่จะโปรแกรมจะมีนามสกุล *.pof



รูปที่ 8.15 การโปรแกรมวงจรลงในชิพ Max+Plus II / Programmer / Program

8.9 ทดสอบการทำงานจริงของวงจรที่เราได้โปรแกรมลงในชิพ แล้วอธิบายการทำงาน

สรุปขั้นตอนการใช้ MAX+PLUS II

วาดวงจร

1. Run โปรแกรม Max+Plus II 9.5 BASELINE ขึ้นมา
2. ตั้งชื่อโปรเจ็คเริ่มจาก **File / Project / Name** เลือก Folder ที่จะเก็บโปรเจ็คของเราและใส่ชื่อของโปรเจ็คตามต้องการ
3. สร้างไฟล์วงจร **File / New / Graphic Editor File** แล้ววาดวงจรที่ต้องการ แล้ว **save**

Compile

4. ระบุเบอร์ของชิพ **Assign / Device** ให้ทำการเลือก Device Family เป็น **MAX700S** และเลือก Device เป็น **EPM7128SLC84-10**
5. คอมไพล์วงจรที่เราได้สร้างขึ้นมา **Max+Plus II / Compiler / Start**

Simulate

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6. Simulate โดยจะต้องสร้างไฟล์ Waveform ขึ้นมาก่อน **File / New / Waveform Editor file**
7. โหลด Noad ต่างๆ เข้ามา **Noad / Enter Noad from SNF**
8. กำหนด **File / End Time** เท่ากับ **1.0 us**
9. กำหนด **Option / Gride Size** มีขนาดเท่ากับ **10.00ns**
10. กำหนดรูปแบบสัญญาณให้กับ **Node Input**
11. บันทึกไฟล์ Waveform ที่ได้สร้างขึ้น **File / Save as**
12. การจำลองการทำงานของวงจร Max+**Plus II / Simulation**

กำหนดขา

13. เปลี่ยนแปลงตำแหน่งขาของชิพ **Max+Plus / Floorplan Editor**
14. เลือกที่เมนู **Assign / Back-Annotate Project** คลิกที่ **Chip,Pin & Device / OK**
15. คลิกที่ **Layout / Current Assignment Foorplan**
16. ทำการ **Drag and Drop** ขาต่างๆ ให้ตรงกับความต้องการ

โปรแกรม

17. โปรแกรมวงจรลงในชิพ Max+**Plus II / Programmer / Program**
18. ทดสอบการทำงานจริงของวงจร

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 9

การออกแบบ และ การสร้าง

ในการออกแบบและการสร้าง ECG MONITOR นั้น เราได้ทำการออกแบบ โดยแบ่งการออกแบบ เป็นสามภาคด้วยกันคือ

9.1 ส่วนของภาคการแสดงผลคลื่นสัญญาณหัวใจ

9.2 ส่วนของการแสดงผลตัวอักษรและกราฟิก

9.3 การออกแบบในส่วนของวงจร (Hardware)

ซึ่งในส่วนการทำงานของทั้งสามภาคนั้นได้ทำงานแยกอิสระต่อกันแต่จะมีบางส่วนเท่านั้นที่ทั้งสามภาคจะต้องทำงานร่วมกัน โดยการทำงานทั้งสามภาคนั้นสามารถอธิบายได้ดังนี้

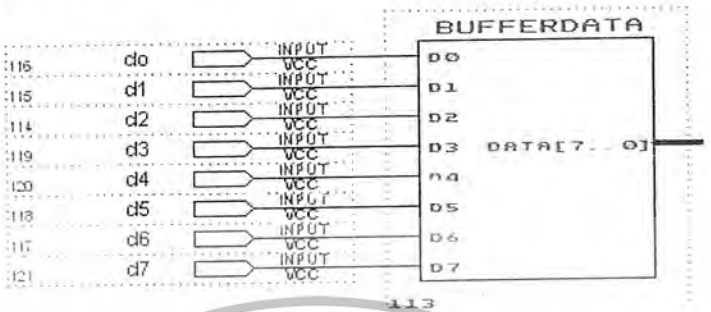
9.1 ส่วนของภาคการแสดงผลคลื่นสัญญาณหัวใจ

วงจรรูปที่ 9.1 เป็นวงจรที่นำเอาข้อมูลจาก A/D ที่ได้จากการ Sampling สัญญาณคลื่นหัวใจที่ถูกขยายความแรงของสัญญาณแล้ว โดยข้อมูลที่ได้จะมีความละเอียด 0 – 256 (8 บิต) ระดับ ซึ่งหมายความว่า สัญญาณคลื่นหัวใจที่เข้ามาทุกๆ 1/500 วินาที (ความถี่ในการ Sampling) จะถูกแปลงเป็นสัญญาณดิจิตอล สัญญาณที่ได้จะถูกเก็บไปที่ SRAM ขนาด 512 Byte โดยใช้เทคนิคในการอ้างอิงแอดเดรส เช่น ทุกครั้งที่มีข้อมูลใหม่เข้ามาแอดเดรสของการเขียนจะเพิ่มขึ้น ซึ่งในการออกแบบกำหนดไว้ว่ามีการเขียนข้อมูลใน SRAM แอดเดรสของการอ่านจะมีการเปลี่ยนแปลงด้วย เป็นการออกแบบให้ภาพที่ปรากฏมีการเคลื่อนที่ ลักษณะที่เลื่อนภาพจากทางด้านขอบจอทางขวามือไปทางซ้ายมือ แล้วนำข้อมูลที่ได้จากการอ่าน SRAM ไปสร้างสัญญาณ RGB ส่งออกไปพร้อมกับ สัญญาณ H_Sync และ V_Sync ของ Monitor

จากรูปจะอธิบายได้คร่าวๆ ดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

9.1.1 บล็อก BUFFERDATA



รูปที่ 9.1 บล็อก BUFFERDATA

เป็นวงจรรับค่าข้อมูลที่ได้จาก A/D ข้อมูลที่ได้จะผ่านเข้าสู่ SRAM โดยตรงเป็นการออกแบบให้ข้อมูลมีรูปแบบเหมือนกับข้อมูลทางINPUTของSRAM

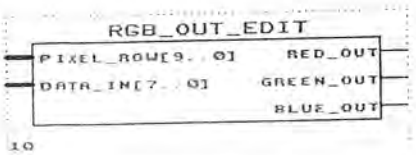
9.1.2 บล็อก LPM_RAM_DQ

รูปที่ 9.2 บล็อก LRM_RAM_DQ

เป็นSRAMที่สร้างจากคำสั่ง ARRAY ถูกออกแบบโดย บริษัท ATERA ได้ออกแบบให้มีการใช้งานที่คล่องตัวและมีความยืดหยุ่นในการใช้สูง ซึ่งผู้ออกแบบสามารถกำหนดขนาดของแอดเดรสความกว้างของข้อมูลในการออกแบบนี้ผู้ออกแบบได้กำหนด SRAMให้มีขนาด512Byte จากการอ่านและการเขียนจะขึ้นอยู่กับสัญญาณ CLOCK_INPUT (ขาWE)ว่ามีสัญญาณClockลอจิกใดจะเป็นการอ่านหรือการเขียนหน่วยความจำ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

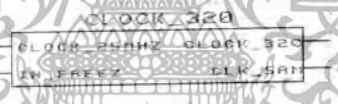
9.1.3 บล็อก RGB_OUT_EDIT



รูปที่ 9.3 บล็อก RGB_OUT_EDIT

การออกแบบได้กำหนดให้ INPUT เป็น PIXEL_ROW ขนาด 10บิต ค่าที่ได้นี้เป็นค่าROW ที่ใช้ในการสแกนภาพจะนับตั้งแต่ 0-480 และอีกขาหนึ่งเป็นขา DATA_IN ขานี้จะรับข้อมูลจากการอ่านSRAM การออกแบบบล็อกนี้กำหนดไว้ว่าถ้าต้องการให้แถว (ROW)ไหนแสดง ผลของข้อมูลก็กำหนดให้ ค่าของPIXEL_ROW นั้นมีค่าเท่ากับ DATAIแล้วขับสัญญาณที่ใช้สร้างสีผ่านRGB_OUT

9.1.4 บล็อกCLOCK_320

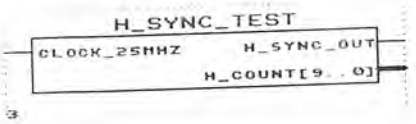


รูปที่ 9.4 บล็อกCLOCK_320

บล็อกนี้จะสร้างความถี่ในการ SAMPLING ให้กับ INPUT ที่ ADC กับ ขาWE ของSRAM การทำงานจะนำCLK ของระบบมาหารเหลือ500 HZ แต่ OUTPUTที่ออกมาเป็น2สัญญาณออกมา 2 สัญญาณนั้นมีความหมายว่าอีกสัญญาณหนึ่งจะมีความหมาย ว่าการออกแบบให้ ADCทำงานแบบมีการ DELAY ทำการเปลี่ยนค่าไปก่อนแล้วค่อยส่ง CLOCK เก็บค่าDATAที่ออกรอไว้ เพื่อการทำงานที่เข้าจังหวะกัน และที่ INPUTจะมีขา IN FREEZE ขานี้ได้ออกแบบให้ผู้ใช้หยุดการ SAMPLING ของข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

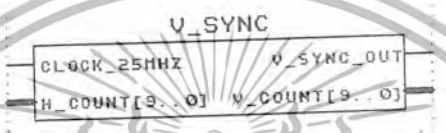
9.1.5 บล็อก H_SYNC



รูปที่ 9.5 บล็อก H_SYNC

เป็นบล็อกที่ใช้เป็นตัวกำเนิดสัญญาณH_SYNC โดยที่INPUTจะเป็นCLOCK_25MHz

9.1.6 บล็อก V_SYNC



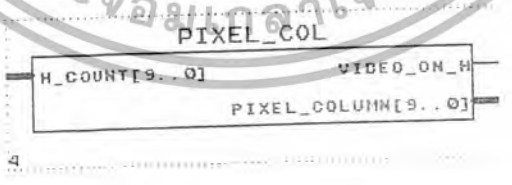
รูปที่ 9.6 บล็อก V_SYNC

เป็นบล็อกที่ใช้เป็นตัวกำเนิดสัญญาณV_SYNC โดยที่INPUTจะเป็นCLOCK_25MHz

9.1.7 บล็อก PIXEL_ROW

รูปที่ 9.7 บล็อก PIXEL_ROW

9.1.8 PIXEL_COL

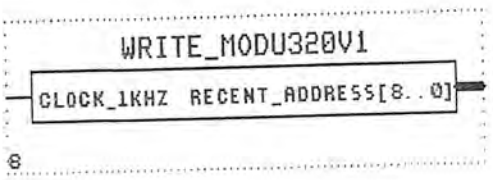


รูปที่ 9.8 บล็อก PIXEL_COL

โดยบล็อกนี้จะสร้างสัญญาณออกมา 2 สัญญาณคือ VIDEO_ON_H และ PIXEL_COL ซึ่งทั้ง 2 สัญญาณนี้จะใช้ในการสร้างสัญญาณภาพ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

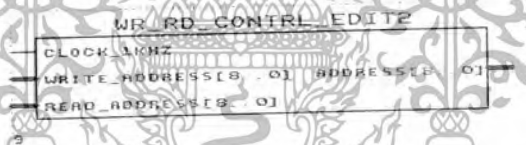
9.1.9 บล็อก WRITE_MODU320V1



รูปที่ 9.9 บล็อก WRITE_MODU320V1

เป็นบล็อกที่ใช้ในการกำหนดค่า ADRESS ที่ใช้ในการเขียน (WRITE)เข้าสู่ SRAM โดย ตำแหน่งแอดเดรสที่จะใช้ INPUT เป็น CLOCK_500 Hz เป็นการรับค่าCLOCK มานับแล้ว ค่าที่นับได้จะเป็นแอดเดรสของการเขียน

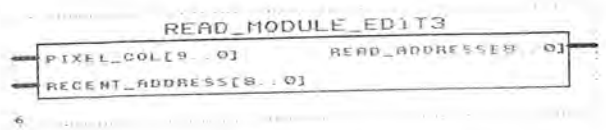
9.1.10 บล็อก WR_RD_CONTROL



รูปที่ 9.10 บล็อก WR_RD_CONTROL

จะใช้ในการควบคุมการเขียนและการอ่านให้เป็นไปตามสัญญาณ CLOCK ถ้ามีสัญญาณ CLOCK เข้ามาจะเป็นการเขียน ถ้าไม่มีสัญญาณCLKเข้ามาแอดเดรสที่ปล่อยออกไปเป็นแอดเดรสที่ใช้ในการอ่าน

9.1.11 บล็อก READ_MOD_EDIT3

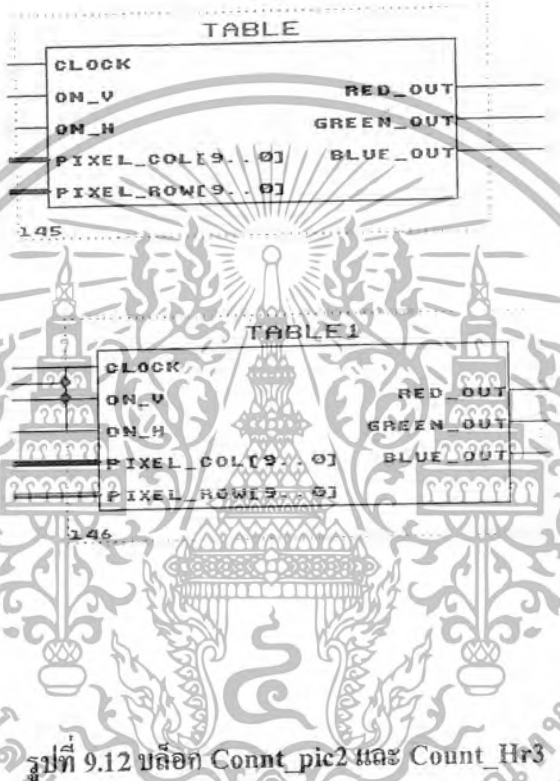


รูปที่ 9.11 บล็อก READ_MOD_EDIT3

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บล็อกนี้เป็นบล็อกสำคัญที่ใช้ในการเลื่อนตำแหน่งแอดเดรสในการอ่านข้อมูลโดยที่INPUT ประกอบไปด้วย PIXEL_COL_กับ RECENT_ADDRESS (เป็นค่าที่บอกว่าเป็นแอดเดรสล่าสุดที่ใช้ในการเขียน) ทั้ง2 แอดเดรสนี่จะเป็นตัวหลักในการเลื่อนภาพ

9.1.12 TABLE



รูปที่ 9.12 บล็อก Connt_pic2 และ Count_Hr3

เป็นการแสดงตารางเพื่อให้พิจารณาด้าน Column เพื่อใช้พิจารณาคาบของสัญญาณ โดยใช้หลักการscan

Table 1

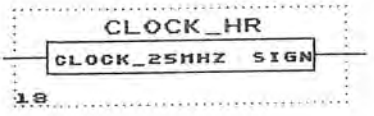
การแสดงตารางด้านRow เพื่อพิจารณาVoltage ของSignal

Flow-RGB

โดยการแสดงผลทางหน้าจอทั้งหมดจะถูกแสดงออกทางของ RGB ทางBlock Flow-RGB ซึ่งทั้งสัญญาณหัวใจ ตัวเลขแสดงอัตราเฉลี่ยและตัวอักษรจะถูก Show ออกทางจอโดยที่ทุกจุดจะถูกระบุให้มีค่าRGB #000

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

9.1.13 Clock-HR



รูปที่ 9.13 บล็อก Clock-HR

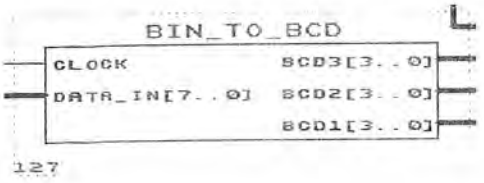
เป็นตัวที่ทำการส่งสัญญาณทุกๆ 6 วินาที ออกไปให้กับ Block อื่นๆ เพื่อการคำนวนค่า

9.1.14 HR

รูปที่ 9.14 บล็อก HR

เป็นส่วนทำการคำนวณค่า โดยการนับค่าภายใน 6 วินาทีว่ามี Pulse ของหัวใจเข้ามากี่ลูกเพื่อนำไปหาค่าเฉลี่ย โดยรับค่าสัญญาณหัวใจที่ถูกแปลงเป็น Clock แล้วนำมานับ

9.1.15 BIN-TO-BCD

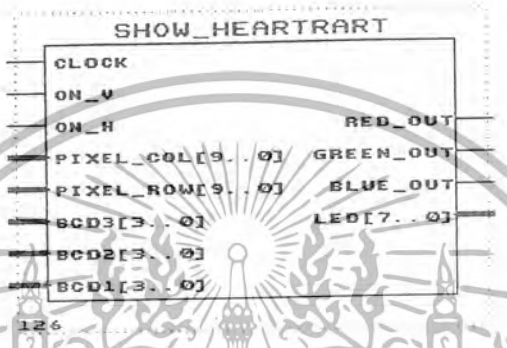


รูปที่ 9.15 บล็อก BIN-TO-BCD

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เป็นส่วนการแสดงค่าจากเลข 8 bit แยกเป็นค่า 4 bit 3 หลักเพื่อนำเข้าสู่ส่วนการแสดงผลการนับอัตราเฉลี่ยของหัวใจเพื่อแปลงค่าเป็น BCD และเป็นส่วนแปลงค่าจากเลข 8 bit เป็น BCD SHOW HEARTRATE

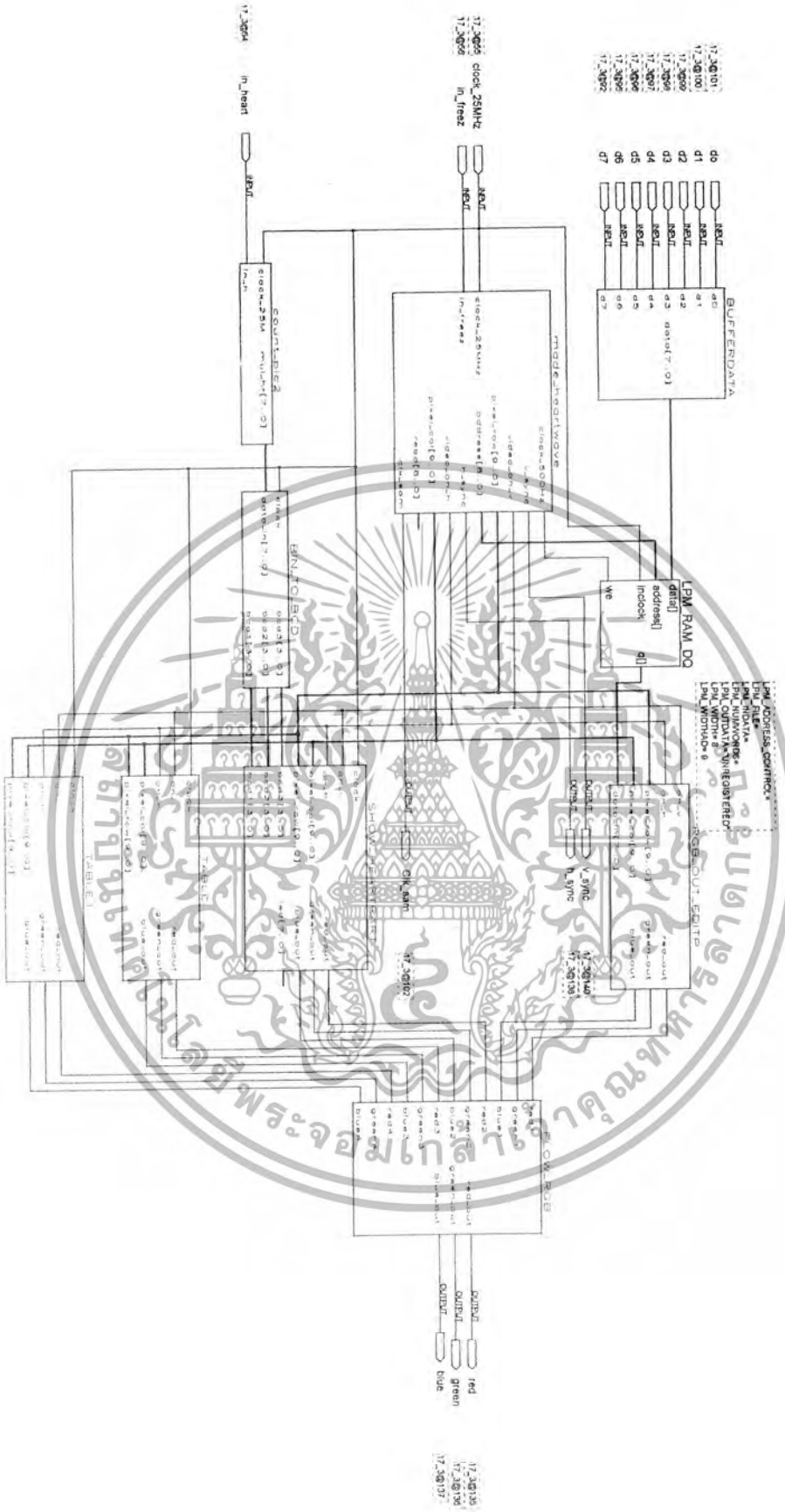
9.1.16 SHOW HEARTRATE



รูปที่ 9.16 บล็อก SHOW HEARTRATE

ใช้แสดงตัวอักษรและเป็นส่วนค่าเลข BCD ออกทางจอเป็นเลข 3 หลัก โดย Update ทุกๆ 6 วินาที และแสดงค่าตัวอักษรทั้งหมดออกมาโดยการใช้ File จาก RAM ที่มาทำการเรียกใช้ขึ้นมา 1 ตัวเพื่อเก็บค่าตัวอักษรและใช้ Address ในการระบุเรียกใช้โดยจะมีการขยายตัวอักษรออกโดยการใช้ขยายทิศละ 4 เท่า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



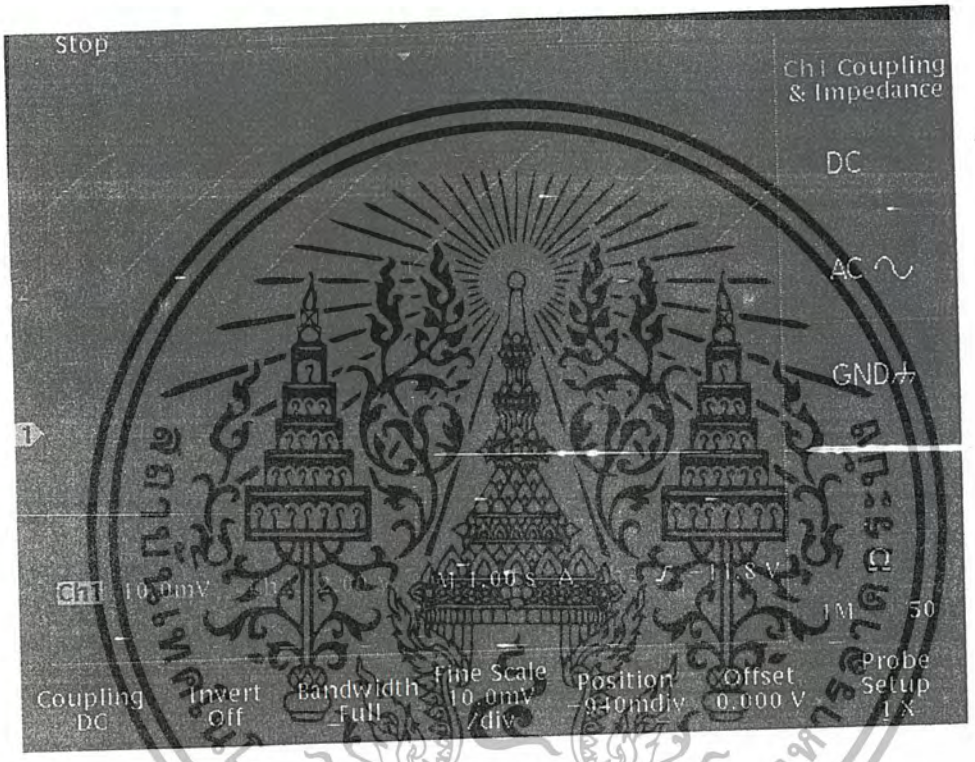
รูปที่ 9.17 แสดง Block Diagram ส่วนการแสดงการเขียนโปรแกรม Max Plus II

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 10

การทดลองและผลการทดลอง

ผลการทดสอบส่วนของการวัดสัญญาณทาง Function Generator

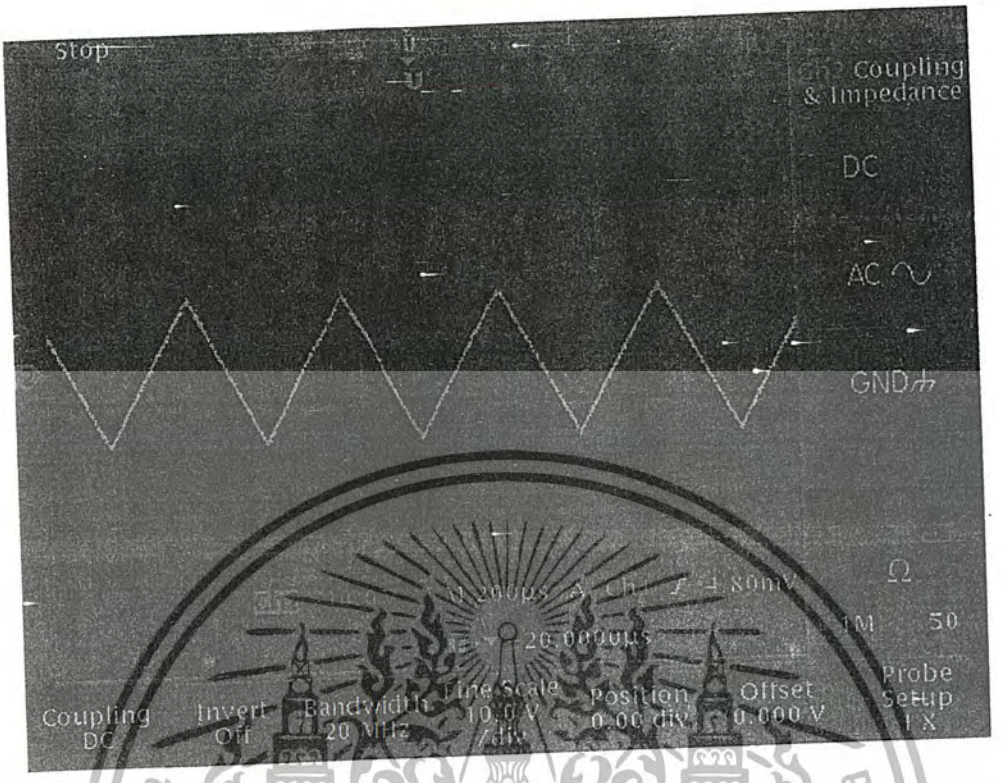


รูปที่ 10.1 แสดงสัญญาณ Input เทียบกับสัญญาณ Output

1. **Channel Input**
2. **Channel Output**

แสดงรูปสัญญาณที่วัดเทียบระหว่างสัญญาณ Input (Channel 1) เทียบกับสัญญาณ Output (Channel 2) ที่ความถี่ 0.5 Hz

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



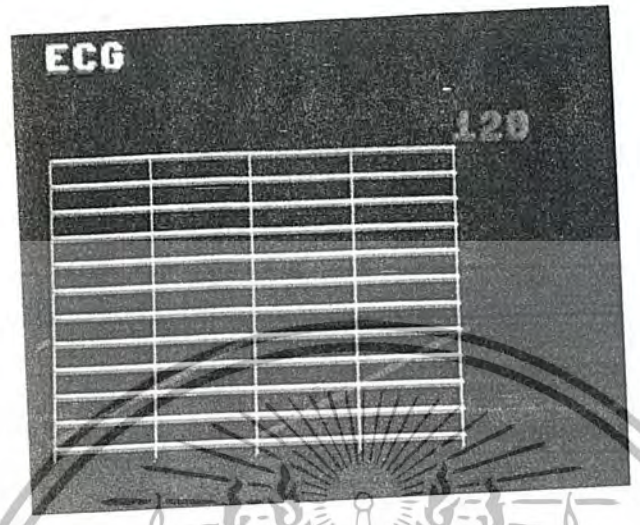
รูปที่ 10.2 แสดงสัญญาณที่นำมา Compare กับสัญญาณ ECG

แสดงสัญญาณที่นำมา Compare กับสัญญาณ Input เพื่อทำการ Modulate

ป้องกันด้านคุณสมบัติด้าน I-V Characteristics ของ Opto Isolator

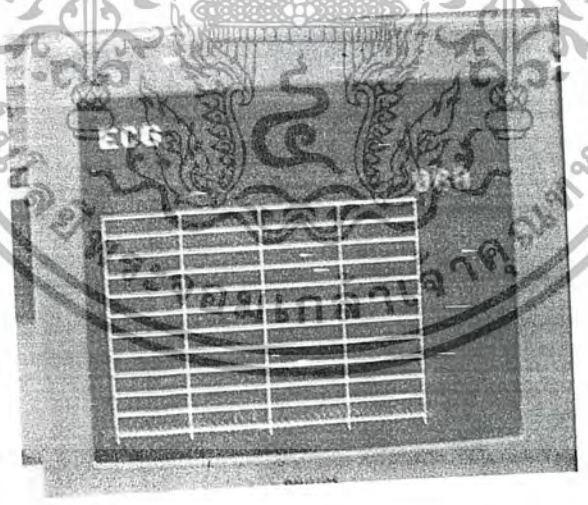
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ผลการทดลองเมื่อป้อนสัญญาณ Input เข้าบอร์ด FPGA



รูปที่ 10.5 แสดงสัญญาณที่วัดได้เมื่อป้อนสัญญาณจาก Generator

-เมื่อป้อนสัญญาณจำลอง ECG จากเครื่อง Generator ที่ความถี่ 2Hz แรงดัน 2Vp-p



รูปที่ 10.7 แสดงสัญญาณที่วัดได้เมื่อป้อนสัญญาณจาก

ECG Generator

-เมื่อป้อนสัญญาณจำลองจากเครื่อง ECG Generator

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 11
สรุปผลการทดลอง

11.1 สรุปผลการทดลอง

-จากผลการทดลองเห็นว่าสามารถแสดงการวาดคลื่นสัญญาณบนหน้าจอโดยที่กราฟมีความราบเรียบโดยพัฒนาเพิ่มจากของเดิม และสามารถแสดงผลได้ตาม Input ออกทางจอ VGA
-อุปกรณ์ทางด้าน Instrument Amplifier ที่ออกแบบใช้งานได้สมบูรณ์ยิ่งขึ้น
-สามารถวัดอัตราการเต้นของหัวใจได้

11.2 ปัญหาที่พบ

-สัญญาณที่วัดได้ยังคงมี ปัญหาเนื่องจาก Noise ที่เกิดจากไฟบ้านมารบกวนสัญญาณ ECG ซึ่งทำให้สัญญาณที่วัดได้ไม่ค่อยมีความเรียบของสัญญาณ
-สัญญาณที่วัดได้ยังไม่มีความต่อเนื่องของสัญญาณ
-ไม่สามารถปรับ Time/Division ได้ ซึ่งทำให้ไม่สามารถเห็นสัญญาณที่ต่อเนื่องกันได้
-ขนาดสัญญาณที่วัดได้ยังไม่มากพอที่จะนำไปใช้จริงได้
-การวัดอัตราการเต้นของหัวใจยังไม่มีความละเอียดพอ

11.3 แนวทางการแก้ไขปัญหา

จากผลการทดลองที่ได้ และปัญหาที่พบนั้นเนื่องจากการเขียนโปรแกรมยังคงมีปัญหาในบางส่วนซึ่งต้องการการพัฒนาต่อไปเพื่อความสมบูรณ์ ในส่วนของ Hardware ยังคงต้องการการแก้ไขในส่วนของการ Filter Noise ซึ่งยังไม่สามารถกำจัดNoise ได้ทั้งหมด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บรรณานุกรม

1. นอ.ชาติชาย ดิษฐกุล รน. ” การใช้โปรแกรมภาษา VHDL” ภาควิชาวิศวกรรมคอมพิวเตอร์
2. นายทศพล พงษ์เจริญ, นายศักดิ์ชัย กระแสอินทร์”การแสดงคลื่นสัญญาณหัวใจผ่านจอ VGA” ปริญญานิพนธ์หลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต สาขาวิชาอิเล็กทรอนิกส์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ปีการศึกษา 2545
3. Stephen Brown and Zvonko Vranesic “Fundamentals of Digital Logic with VHDL Design” International Editions 2000
4. รศ. ดร. มนัส สังวรศิลป์, ดร. กิตติพล ชิตสกุล, อ.เกษมสุข เสพติริสุข, บุญอนันต์ เกียงเอีย “เปิดโลก FPGA กับ WIZARD PLD-A01” สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
5. ยงยุทธ สหัสกุล,ECG ทางคลินิก,การศึกษาแพทย์ศาสตร์ คณะแพทย์ศาสตร์ศิริราช พยาบาลมหาวิทยาลัยมหิดล,2546
6. ชมพูนุช อ่องจริต,CLINICAL ECG,คณะแพทย์ศาสตร์จุฬาลงกรณ์มหาวิทยาลัย,2525

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก

Source Code

บล็อก bin_to_bcd

```
LIBRARY altera;
USE altera.maxplus2.ALL;
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity bin_to_bcd is
port(
        clock           : in std_logic;
    data_in    : in std_logic_vector(7 downto 0);
    bcd3            : out std_logic_vector(3 Downto 0);
        bcd2            : out std_logic_vector(3 Downto 0);
        bcd1            : out std_logic_vector(3 Downto 0)
    );
end bin_to_bcd;
architecture arc_bintobcd of bin_to_bcd is
--      signal temp : std_logic_vector(19 Downto 0);
begin
        process(clock,data_in)
                variable temp : std_logic_vector(19 Downto 0);
        begin
                if( (clock'event) and (clock = '1') )then
                        ---------========== assign bcd1 ==========---------
                        temp(19 Downto 8) := "000000000000";
                        temp(7 Downto 0) := data_in;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        for i in 1 to 8 loop
            if temp(11 Downto 8)>= 5 then
                temp := temp +"00000000001100000000";
            end if;
            if temp(15 Downto 12) >= 5 then
                temp := temp +"00000011000000000000";
            end if;
            if temp(19 Downto 16) >= 5 then
                temp := temp+"00110000000000000000";
            end if;
            temp := temp(18 Downto 0) & '0';
        end loop;

        bcd3<=temp(19 Downto 16);
        bcd2<=temp(15 Downto 12);
        bcd1<=temp(11 Downto 8);

--      if data_in = "11111111" then
--          bcd3 <="1111";
--          bcd2 <="1111";
--          bcd1 <="1111";
--      end if;
        end if;

    end process;
end arc_bintobcd ;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บล็อก bufferdata

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
entity bufferdata is
port( d0:in std_logic;
        d1:in std_logic;
    d2:in std_logic;
    d3:in std_logic;
    d4:in std_logic;
        d5:in std_logic;
    d6:in std_logic;
        d7:in std_logic;
    data:out std_logic_vector(7 downto 0));
end bufferdata;
architecture buffer_behav of bufferdata is
begin
 data(0) <= d0;
 data(1) <= d1;
 data(2) <= d2;
 data(3) <= d3;
 data(4) <= d4;
 data(5) <= d5;
 data(6) <= d6;
 data(7) <= d7;
end buffer_behav;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บล็อก clock_500

```
library ieee;
use ieee.std_logic_arith.all;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity clock_500 is
port (clock_25MHz :in  std_logic;
    clock_320  :out std_logic;
    in_freez   :in std_logic;
    clk_sam   :out std_logic);
end clock_500 ;
architecture clock of clock_500 is
begin
 process(clock_25MHz)
        variable count : integer range 0 to 51000 ;
  begin
   wait until clock_25MHz'event and clock_25MHz='1';
   if in_freez ='1' then
        if count =  50000 then
       clock_320 <='1';
                        count :=0;
                  else
       clock_320 <='0';
       count :=count+1;
    if count > 49900  and count < 50000 then
      clk_sam <= '0';
    else
      clk_sam <='1' ;
    end if;
    --clk_sam <= '0';
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        end if;
    end if;
   end process;
end clock;
```

บล็อก clock_HR

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity clock_HR is
port( clock_25MHz       :in     std_logic;
        sign                    :out    std_logic);
end clock_HR ;
architecture count of clock_HR is
begin
process(clock_25MHz)
variable count_ck : integer range 0 to 150000000 ;
        begin
        wait until (clock_25MHz'event) and (clock_25MHz='1');
                if  count_ck    >=150000000  then
                        sign            <='1';
                        count_ck := 0;
                else sign       <='0';
                        count_ck  := count_ck+1;
            end if;
 end process;
 end count;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บล็อก count_HR3

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity count_HR3 is
port  (clock_hr  :in      std_logic;
                count_h         :in     std_logic_vector(7 downto 0);
                mul_hr          :out    std_logic_vector(7 downto 0));
end count_HR3 ;
architecture count of count_HR3 is
signal   a,p : std_logic_vector(7 downto 0);
signal  temp1,temp2 : integer range 0 to 255;
begin
process(clock_hr,count_h)
begin
        wait until  clock_hr'event and clock_hr = '1';
        --if clock_hr = '1' then
            --a     <= count_h ;
            --p     <= a+a+a+a+a+a+a+a+a+a;
        case count_h is
                    when"00000000"=>                --0
                    a<="00000000";
                when"00000001"=>                --1
                    a<="00001010" ;
                when"00000010"=>                --2
                    a<="00010100";
                when"00000011"=>                --3
                    a<="00011110";
                when"00000100"=>                --4
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
            a<="00101000";
when"00000101"=>                --5
            a<="00110010";
when"00000110"=>                --6
            a<="00111100";
when"00000111"=>                --7
            a<="01000110";
when"00001000"=>                --8
            a<="01010000";
when"00001001"=>                --9
            a<="01011010";
when"00001010"=>                --10
            a<="01100100";
when"00001011"=>                --11
            a<="01101110";
when"00001100"=>                --12
            a<="01111000";
when"00001101"=>                --13
            a<="10000010";
when"00001110"=>                --14
            a<="10001100";
when"00001111"=>                --15
            a<="10010110";
when"00010000"=>                --16
            a<="10100000";
when"00010001"=>                --17
            a<="10101010";
when"00010010"=>                --18
            a<="10110100";
when"00010011"=>                --19
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                a<="10111110";
            when"00010100"=>        --20
                a<="11001000";
            when"00010101"=>        --21
                a<="11010010";
            when"00010110"=>        --22
                a<="11011100";
            when"00010111"=>        --23
                a<="11100110";
            when"00011000"=>        --24
                a<="11110000";
            when"00011001"=>        --25
                a<="11111010";
            when others =>
                a<="11111111"; --other
        end case;
        if      a <= "00000000" then p <= "00000000";
        else p <= "00001010";
        end if;
end process;
        mul_hr <= a + p;
end count;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บล็อก hor_syn

```
-----hor_sync--------/
library ieee;
Library Work;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity h_sync is
 port(clock_25MHz : in std_logic;
          h_sync_out  : out std_logic;
          h_count     : out std_logic_vector(9 downto 0));
 end h_sync ;
architecture hor of h_sync is
 signal hor_count : std_logic_vector(9 downto 0);
 begin
  process
   begin
    wait until clock_25MHz'event and clock_25MHz='1';
     if hor_count=799 then
                hor_count<= "0000000000";
          else
                hor_count<=hor_count+1;
     end if;

          if hor_count<756 and hor_count>658 then
          h_sync_out<='0';
    else
          h_sync_out<='1';
    end if;
    h_count<=hor_count;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
 end process;
end hor;
```

บล็อก Ver_syn

```
--------Ver_syn----------
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity v_sync is
port(clock_25MHz : in std_logic;
   h_count      : in std_logic_vector(9 downto 0);
         v_sync_out : out std_logic ;
         v_count      :out std_logic_vector(9 downto 0));
end v_sync;
architecture ver of v_sync is
 signal ver_count : std_logic_vector(9 downto 0);
 begin
  process
  begin
         wait until clock_25MHz'event and clock_25MHz='1';
         if ver_count>523 and h_count>698 then
          ver_count<="0000000000";
    elsif(h_count=699) then
     ver_count<=ver_count+1;
    end if;

    if ver_count<495 and ver_count>492 then
     v_sync_out<='0';
          else
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        v_sync_out<='1';
    end if;
    v_count<=ver_count;
 end process;
end ver;
```

บล็อก HR

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity HR is
port  (          clock_hr      :in      std_logic;
                 in_h          :in      std_logic;
                 count_h       :out     std_logic_vector(7 downto 0));
end HR ;
architecture count of HR is
signal count,a :std_logic_vector(7 downto 0) ;
begin
count_h <= a;
process(clock_hr,in_h)
begin   if clock_hr = '1'  then
                    --a              <= "00000000";
                    count  <= "00000000";
               elsif    in_h'event and in_h = '1'then
                        count <= count + 1;
                        a <= count;
               end if;
end process;
end count;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บล็อก pixel_col

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity pixel_col is
port(h_count : in std_logic_vector(9 downto 0);
    video_on_h : out std_logic;
          pixel_column: out std_logic_vector(9 downto 0));
end pixel_col;
architecture p_col of pixel_col is
 begin
 process
  begin
   if(h_count<640) then
    video_on_h<='1';
    pixel_column<=h_count;
   else
    video_on_h<='0';
   end if;
 end process;
end p_col;
```

บล็อก pixel_row

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity pixel_row is
 port(v_count              :in std_logic_vector(9 downto 0);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    video_on_v  :out std_logic;
          pixel_row                :out std_logic_vector(9 downto 0));
end pixel_row;

architecture p_row of pixel_row is
 begin
  process
   begin
    if(v_count<480) then
            video_on_v<='1';
      pixel_row<=v_count;
     else
      video_on_v<='0';
     end if;
  end process;
end p_row;
```

บล็อก read_module_edit

```
library IEEE;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_1164.all;
entity read_module_edit is
port( pixel_col    :in  std_logic_vector(9 downto 0);
     recent_address:in  std_logic_vector(8 downto 0);
     read_address  :out std_logic_vector(8 downto 0));
end read_module_edit;
architecture read_modu_behav of read_module_edit is
 begin
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
--- move graph ----------
process
 variable read_add :std_logic_vector( 9 downto 0) ;
   begin
                    read_add := pixel_col + recent_address + 1;
                    if read_add < 512 then
                      read_address <= read_add ;
                    else
                      read_address <= read_add  - 512;
            end if;
 end process;
-------- end move graph ----------------------
----  don't move graph ----------
                    ----i  ...ldress<= pixel_col(9 downto 0);
---- end edit -------
end read_modu_behav;
```

บล็อก rgb_out_editp

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity rgb_out_editp is
port(
          on_v,on_h        :in std_logic ;
    pixel_row  :in std_logic_vector(9 downto 0);
    pixel_col  :in std_logic_vector(9 downto 0);
    data_in    :in std_logic_vector(7 downto 0);
    red_out,green_out,blue_out    :out std_logic
      );
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    end rgb_out_editp;
architecture color of rgb_out_editp is
begin
 process
  variable sum     : std_logic_vector(9 downto 0);
  begin
    sum    := pixel_row + data_in;
    if  sum = 460 and pixel_col <= 512 then
         red_out    <= '0' and on_v and on_h;
         green_out <= '1' and on_v and on_h;
         blue_out  <= '0' and on_v and on_h;
       else
         red_out   <= '0' and on_v and on_h;
         green_out <= '0' and on_v and on_h;
         blue_out  <= '0' and on_v and on_h;
       end if;
 end process ;
end color;
```

บล็อก show_heartrart

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
LIBRARY lpm;
USE lpm.lpm_components.ALL;
entity show_heartrart is
port(
          clock                    : in std_logic; --25Mhz
          on_v,on_h                : in std_logic;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    pixel_col              :in std_logic_vector(9 downto 0);
    pixel_row              :in std_logic_vector(9 downto 0);
    bcd3,bcd2,bcd1         :in std_logic_vector(3 downto 0);
    red_out,green_out,blue_out   :out std_logic;
          led : out std_logic_vector(7 Downto 0)
      );
end show_heartrart;
 architecture arc of show_heartrart is
          signal rom_address_fpga : std_logic_vector(8 Downto 0);
          signal rom_data_fpga : std_logic_vector(7 Downto 0);
          signal templed : std_logic_vector(7 Downto 0);
          signal flag : std_logic:='0';
          -- countstate
          signal expand_pixel_col : std_logic_vector( 1 Downto 0):="00";
          signal expand_pixel_row : std_logic_vector( 1 Downto 0):="00";
          signal point : Integer range 7 to 0 ;
          signal expand_row_point : std_logic_vector ( 2 Downto 0):="000";
          -- constant --
          constant start_row :std_logic_vector ( 9 Downto 0):="0001101111"; --111
          constant end_row :std_logic_vector ( 9 Downto 0):="0010001110";  --142
          constant base_address : std_logic_vector (8 Downto 0):="110000000";
          constant start_row_ecg : std_logic_vector ( 9 Downto 0):="0000010101"; --21
          constant end_row_ecg   : std_logic_vector (9 Downto 0) :="0000110100";--52
--signal xxx : INTEGER range 7 to 0;
-- PROGRAM BEGIN
begin
-- Small 8 by 8 Character Genrator ROM for Video Display
tiny_char_gen_rom:lpm_rom
    GENERIC MAP ( lpm_widthad => 9,
     lpm_numwords => 512,
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    lpm_outdata => "UNREGISTERED",
    lpm_address_control => "UNREGISTERED",
-- Reads in mif file for character generator data
    lpm_file => "tcgrom.mif",
    lpm_width => 8)
  PORT MAP ( address => rom_address_fpga, q => rom_data_fpga);
--rom_address_fpga<=;
led<=rom_data_fpga;
process(clock)
 variable sum     : std_logic_vector(9 downto 0);
begin
if (clock='1' and clock 'event) then
      -- clear address to start all round
      if(( (pixel_row = start_row) and (pixel_col = 501)) or ((pixel_row = start_row_ecg) and
(pixel_col = 20)))then
            rom_address_fpga<=base_address;
            expand_pixel_col<="00";
            expand_pixel_row<="00";
            expand_row_point<="000";
            point <= 7;
      end if;
--============================================ ECG Range
=============================================----
if ( ( pixel_row >= start_row_ecg) and ( pixel_row <= end_row_ecg) and ( pixel_col >= 21 ) and
( pixel_col <= 52 ) ) then
                  rom_address_fpga <= "000101000" + expand_row_point ;
                  if ( rom_data_fpga(point)= '1' ) then
                        red_out <= '1' and on_v and on_h ;
            green_out <= '0' and on_v and on_h ;
            blue_out  <= '1' and on_v and on_h ;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                else
                        red_out <= '0' and on_v and on_h ;
        green_out <= '0' and on_v and on_h ;
        blue_out  <= '0' and on_v and on_h ;
                end if;
                expand_pixel_col <= expand_pixel_col + '1';
                if ( expand_pixel_col ="00" ) then
                        point <= point - 1 ;
                end if;
        if ( pixel_col = 52 ) then
                        point <= 7;
                        expand_pixel_col<="00";
        end if;
    elsif ( ( pixel_row >= start_row_ecg) and ( pixel_row <= end_row_ecg) and ( pixel_col
>= 54 ) and ( pixel_col <= 85 ) ) then
                rom_address_fpga <= "000011000" + expand_row_point ;
                if ( rom_data_fpga(point)= '1' ) then
                        red_out <= '1' and on_v and on_h ;
        green_out <= '0' and on_v and on_h ;
        blue_out  <= '1' and on_v and on_h ;
                else
                        red_out <= '0' and on_v and on_h ;
        green_out <= '0' and on_v and on_h ;
        blue_out  <= '0' and on_v and on_h ;
                end if;
                expand_pixel_col <= expand_pixel_col + '1';
                if ( expand_pixel_col ="00" ) then
                        point <= point - 1 ;
                end if;
        if ( pixel_col = 85) then
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                point <= 7;
                expand_pixel_col<="00";
        end if;
    elsif ( ( pixel_row >= start_row_ecg) and ( pixel_row <= end_row_ecg) and ( pixel_col
>= 87 ) and ( pixel_col <= 118 ) ) then
                rom_address_fpga <= "000111000" + expand_row_point ;
                if ( rom_data_fpga(point)= '1' ) then
                        red_out <= '1' and on_v and on_h ;
            green_out <= '0' and on_v and on_h ;
            blue_out  <= '1' and on_v and on_h ;
                else
                        red_out <= '0' and on_v and on_h ;
            green_out <= '0' and on_v and on_h ;
            blue_out  <= '0' and on_v and on_h ;
                end if;
                expand_pixel_col <= expand_pixel_col + '1';
                if ( expand_pixel_col ="00" ) then
                        point <= point - 1 ;
                end if;
            if ( pixel_col = 118) then
                        point <= 7;
                        expand_pixel_col<="00";
                        expand_pixel_row <= expand_pixel_row + '1';
            end if;
        else
            flag <= '1';
        end if;
        if ( (pixel_row >= start_row_ecg) and (pixel_row <= end_row_ecg ) and (pixel_col =
119 ) and (expand_pixel_row = "00")) then
                expand_row_point <= expand_row_point + 1;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        end if;
-------------------- HR Text --------------------
---------- Heart Rate Range
--------------------------------------------------

        if ( ( pixel_row >= start_row) and ( pixel_row <= end_row) and ( pixel_col >= 501 ) and
( pixel_col <= 532 ) ) then
                        rom_address_fpga <= base_address + ( CONV_INTEGER(bcd3) * 8) +
expand_row_point ;
                        if ( rom_data_fpga(point)= '1' ) then
                                red_out <= '1' and on_v and on_h ;
                        green_out <= '0' and on_v and on_h ;
                        blue_out  <= '0' and on_v and on_h ;
                        else
                                red_out <= '0' and on_v and on_h ;
                        green_out <= '0' and on_v and on_h ;
                        blue_out  <= '0' and on_v and on_h ;
                        end if;
                        expand_pixel_col <= expand_pixel_col + '1';
                        if ( expand_pixel_col ="00" ) then
                                point <= point - 1 ;
                        end if;
                if ( pixel_col = 532 ) then
                                point <= 7;
                                expand_pixel_col<="00";
                end if;
        elsif ( ( pixel_row >= start_row) and ( pixel_row <= end_row) and ( pixel_col >= 533 )
and ( pixel_col <= 564 ) ) then
                        rom_address_fpga <= base_address + ( CONV_INTEGER(bcd2) * 8) +
expand_row_point ;
                        if ( rom_data_fpga(point)= '1' ) then
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                red_out <= '1' and on_v and on_h ;
        green_out <= '0' and on_v and on_h ;
        blue_out  <= '0' and on_v and on_h ;
            else
                red_out <= '0' and on_v and on_h ;
        green_out <= '0' and on_v and on_h ;
        blue_out  <= '0' and on_v and on_h ;
            end if;
            expand_pixel_col <= expand_pixel_col + '1';
            if ( expand_pixel_col ="00" ) then
                point <= point - 1 ;
            end if;
        if ( pixel_col = 564 ) then
            --expand_pixel_row <= expand_pixel_row + '1';
                point <= 7;
                expand_pixel_col<="00";
            end if;
    elsif ( ( pixel_row >= start_row) and ( pixel_row <= end_row) and ( pixel_col >= 565 )
and ( pixel_col <= 596 ) ) then
            rom_address_fpga <= base_address + ( CONV_INTEGER(bcd1) * 8) +
expand_row_point ;
            if ( rom_data_fpga(point)= '1' ) then
                red_out <= '1' and on_v and on_h ;
        green_out <= '0' and on_v and on_h ;
        blue_out  <= '0' and on_v and on_h ;
            else
                red_out <= '0' and on_v and on_h ;
        green_out <= '0' and on_v and on_h ;
        blue_out  <= '0' and on_v and on_h ;
            end if;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                expand_pixel_col <= expand_pixel_col + '1';
                if ( expand_pixel_col ="00" ) then
                        point <= point - 1 ;
                end if;
        if ( pixel_col = 596 ) then
                        point <= 7;
                        expand_pixel_col<="00";
                        expand_pixel_row <= expand_pixel_row + '1';
                end if;
        else
                flag <= '1';
        end if;
        if ( (pixel_row >= start_row_row) and (pixel_row <= end_row_row) and (pixel_col = 599 ) and
(expand_pixel_row = "00")) then
                expand_row_point <= expand_row_point + 1;
        end if;
 end if;
        if (flag='0') then
                red_out <= '0' and on_v and on_h ;
     green_out <= '0' and on_v and on_h ;
     blue_out  <= '0' and on_v and on_h ;
   else
        flag <= '0';
   end if;
 end process ;
end arc;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บล็อก table

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
LIBRARY lpm;
USE lpm.lpm_components.ALL;
entity table is
port(
        clock                       :in std_logic; --25Mhz
        on_v,on_h                   :in std_logic;
        pixel_col                   :in std_logic_vector(9 downto 0);
        pixel_row                   :in std_logic_vector(9 downto 0);
        red_out,green_out,blue_out  :out std_logic
    );
end table ;
architecture are of table is
        signal flag : std_logic:='0';
        signal temp_row:std_logic_vector(9 Downto 0);
        signal temp_col :std_logic_vector(9 Downto 0);
        signal red,green,blue    :std_logic;
        -- constant --
        constant start_row       :std_logic_vector (9 Downto 0):="0010010110"; --150
        constant end_row         :std_logic_vector (9 Downto 0):="0111010110"; --470
        constant start_col       :std_logic_vector (9 Downto 0):="0000010100"; --20
        constant end_col:std_logic_vector (9 Downto 0):="0111110100"; --500
        constant gap     : std_logic_vector(6 Downto 0):="1111000";--120
    --PROGRAM BEGIN
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บล็อก table1

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
LIBRARY lpm;
USE lpm.lpm_components.ALL;
entity table1 is
port(
        clock                   : in std_logic; --25Mhz
        on_v,on_h               : in std_logic;
        pixel_col               :in std_logic_vector(9 downto 0);
        pixel_row               :in std_logic_vector(9 downto 0);
        red_out,green_out,blue_out      :: std_logic
    );
end table1 ;
architecture arc of table1 is
        signal temp_row:std_logic_vector(9 Downto 0);
        signal red,green,blue   :std_logic;
        -- constant --
        constant start_row      :std_logic_vector (9 Downto 0):="0010010110"; --150
        constant end_row        :std_logic_vector (9 Downto 0):="0111010110"; --470
        constant start_col      :std_logic_vector (9 Downto 0):="0000010100"; --20
        constant end_col:std_logic_vector (9 Downto 0):="0111110100"; --500
        constant gap    : std_logic_vector(4 Downto 0):="11100";
-- PROGRAM BEGIN
begin
red_out<=red;
green_out<=green;
blue_out<=blue;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
process(clock)
begin
 if (clock='1' and clock 'event) then
        if (pixel_row = start_row ) and(pixel_col < end_col)then
                temp_row <= start_row;
        end if;
--      if ((pixel_row = temp_row ) and (pixel_col <= end_col) and (pixel_row >=start_row)
and (pixel_col >= start_col) and (pixel_row <= end_row) )then
        if ((pixel_row = temp_row ) and (pixel_col <= end_col) and (pixel_row
>=start_row)and(pixel_col >= start_col)and (pixel_row <= end_row)) then
                red <='1'and on_v and on_h;
                green<='1'and on_v and on_h;
                blue <='0'and on_v and on_h;
                  if(pixel_col = end_col) then
                    temp_row <= temp_row + gap;
                end if;
        else
                red <='0'and on_v and on_h;
                green<='0'and on_v and on_h;
                blue <='0'and on_v and on_h;
        end if;
 end if;
end process ;
end arc;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บล็อก v_sync

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity v_sync is
port(clock_25MHz : in std_logic;
   h_count       : in std_logic_vector(9 downto 0);
   v_sync_out    : out std_logic ;
   v_count       :out std_logic_vector(9 downto 0));
end v_sync;
architecture ver of v_sync is
 signal ver_count : std_logic_vector(9 downto 0);
 begin
  process
  begin
        wait until clock_25MHz'event and clock_25MHz='1';
        if ver_count>523 and h_count>698 then
         ver_count<="0000000000";
    elsif(h_count=699) then
     ver_count<=ver_count+1;
    end if;
     if ver_count<495 and ver_count>492 then
     v_sync_out<='0';
          else
           v_sync_out<='1';
    end if;
    v_count<=ver_count;
 end process;
end ver;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บล็อก wr_rd_contrl_edit

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;
library work;
entity wr_rd_contrl_edit is
port( clock_320Hz  :in std_logic ;
    write_address:in std_logic_vector(8 downto 0);
    read_address :in std_logic_vector(8 downto 0);
    address     :out std_logic_vector(8 downto 0));
end wr_rd_contrl_edit;
architecture wr_rd_contrl_behav of wr_rd_contrl_edit is
 begin
  process
   begin
if clock_320Hz = '1' then
  address <= write_address;
else
  address <= read_address;
end if;
end process;
 end wr_rd_contrl_behav;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

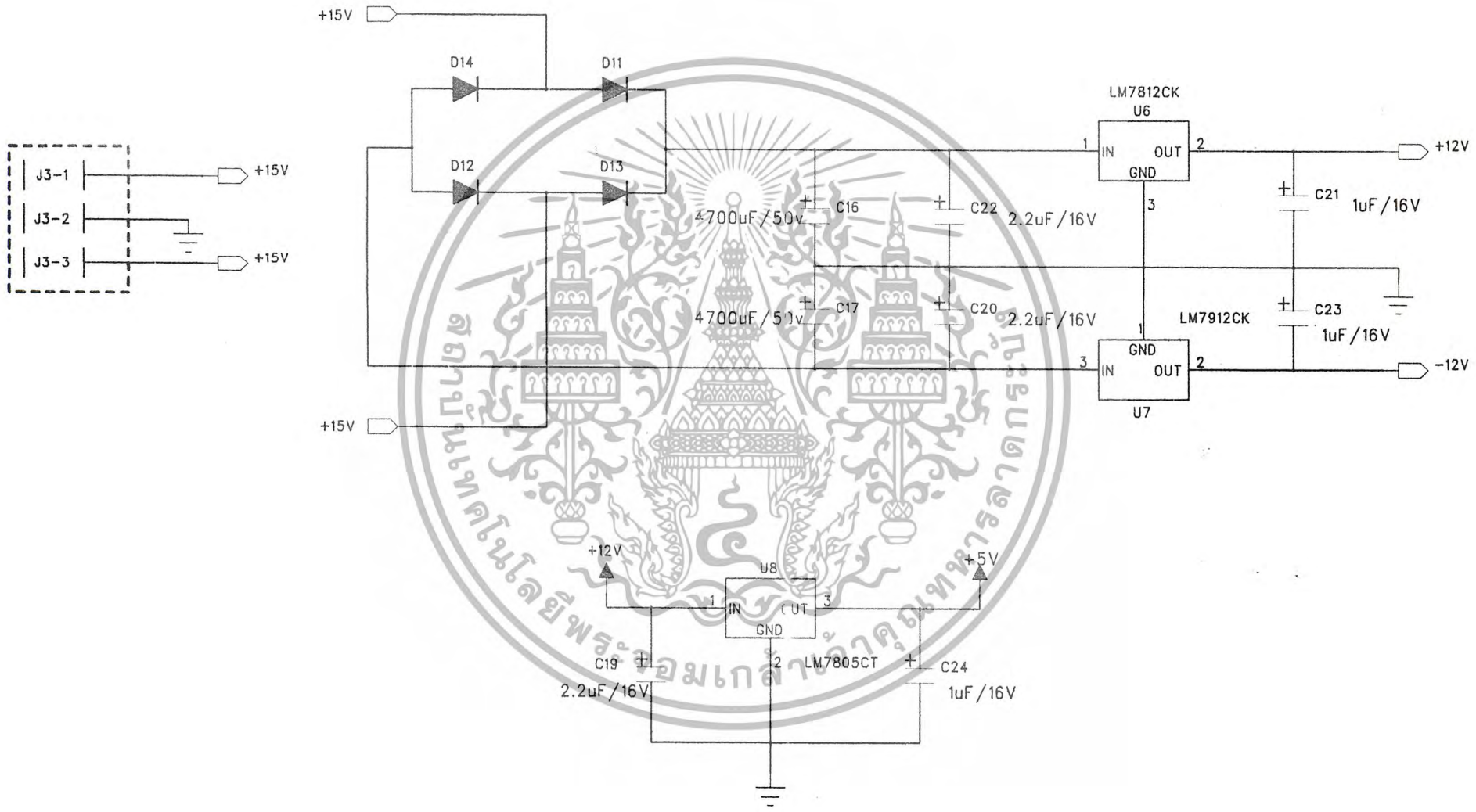
บล็อก write_modul320

```
library IEEE;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
library WORK;
entity write_modul320 is
port( clock_320Hz                    :in std_logic;
    recent_address          :out std_logic_vector(8 downto 0));
end write_modul320;
architecture write_modu_behav of write_modul320 is
begin
 process(clock_320Hz)
variable clock_count :   std_logic_vector(8 downto 0);
begin
        if (clock_320Hz='1') then
        clock_count :=clock_count+1;
        recent_address<=clock_count;
        end if;
end process;
end write_modu_behav;
```

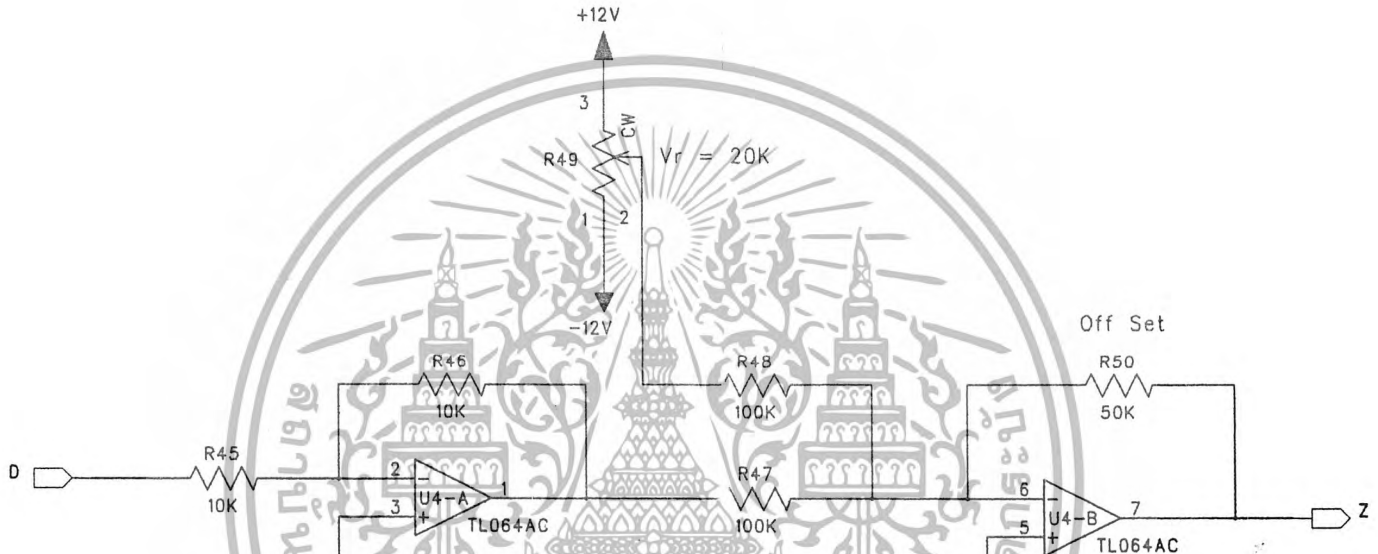
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

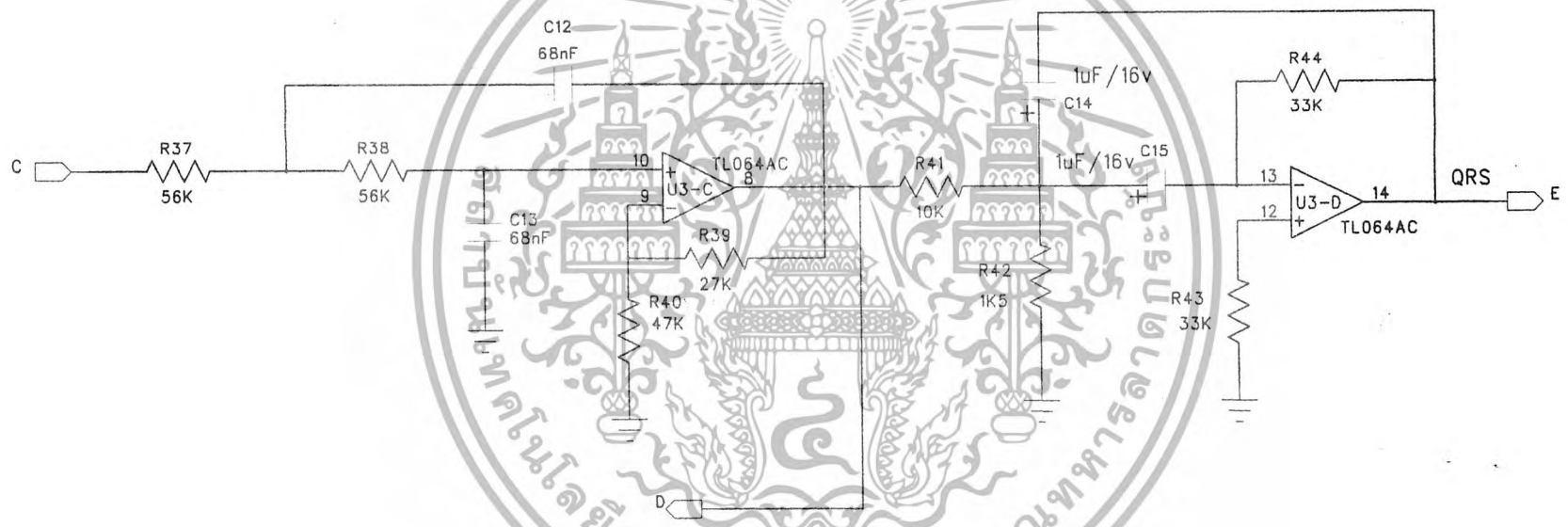
แสดงส่วนของการต่ออุปกรณ์ภาค Hardware

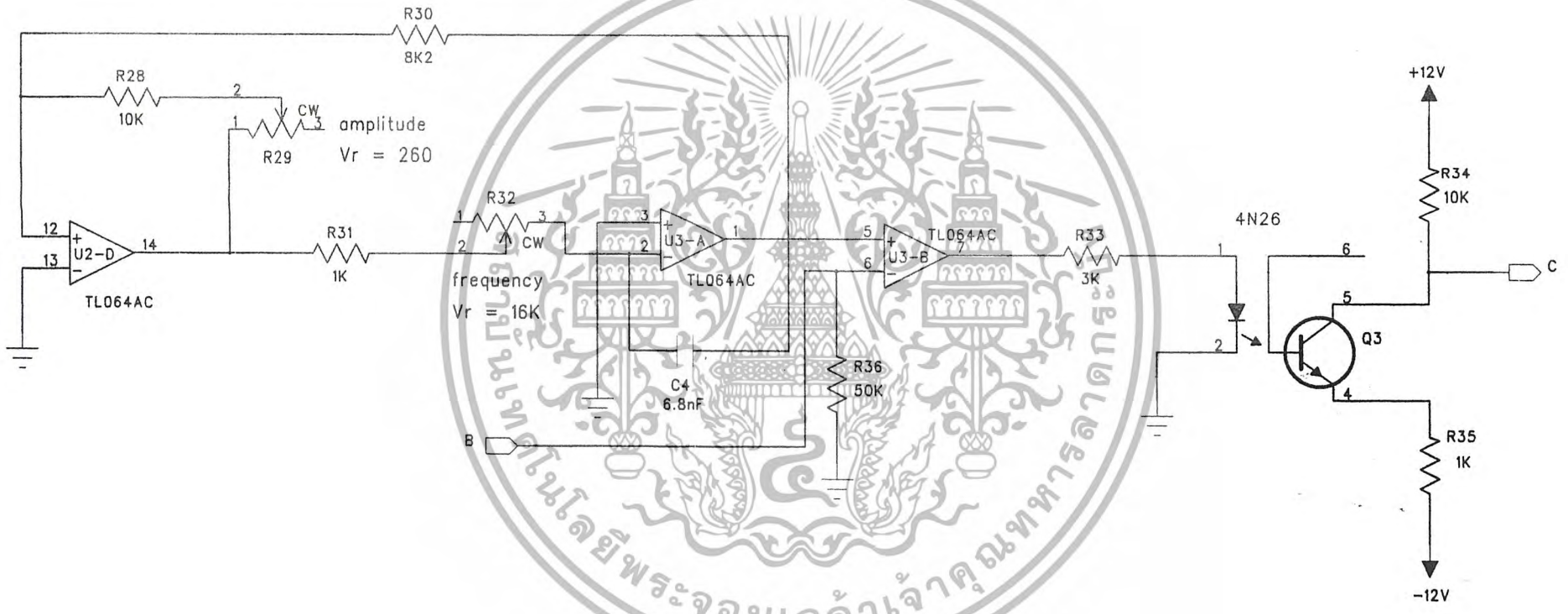
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

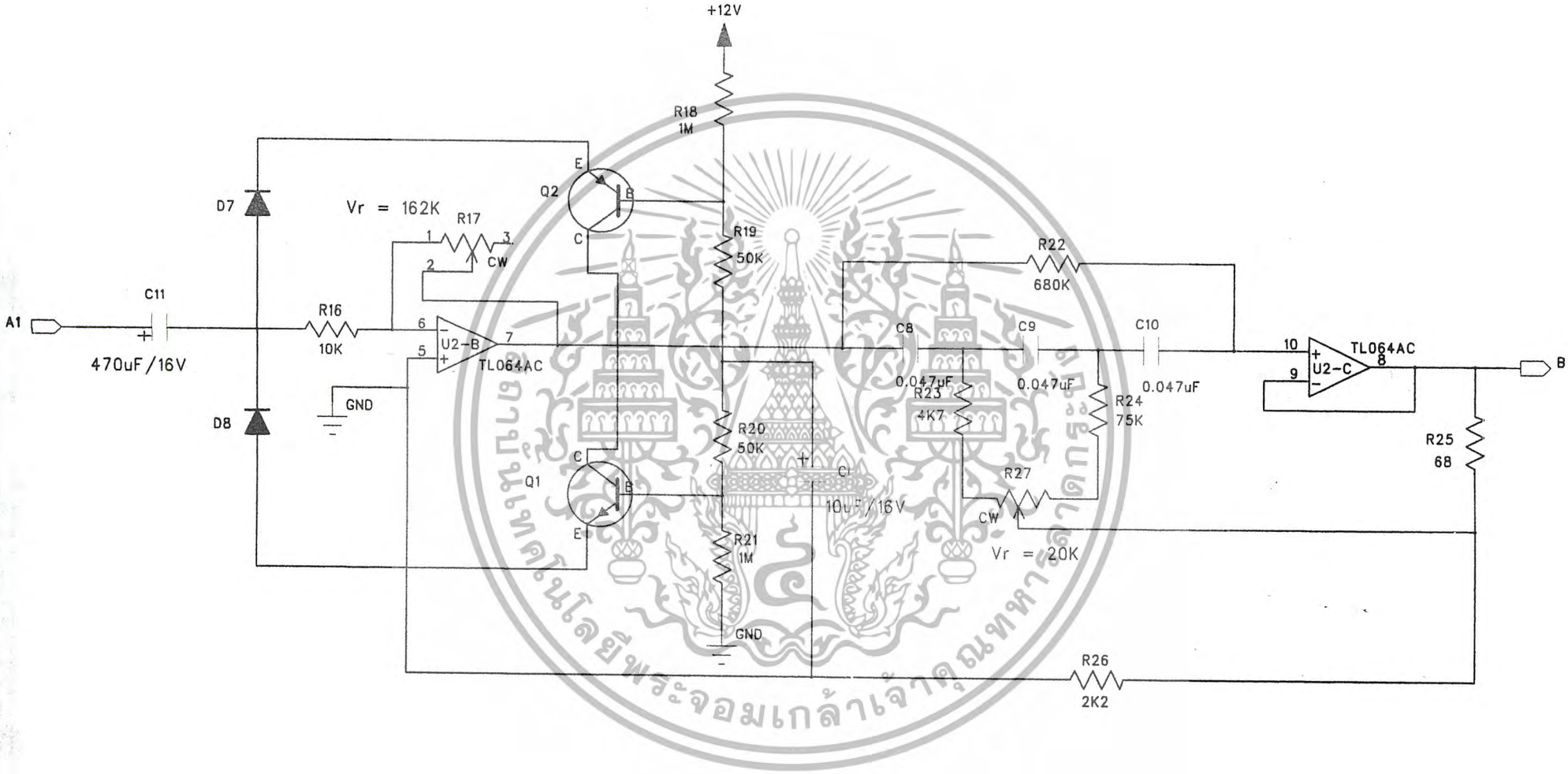

J3-1
J3-2
J3-3
+15V
+15V
+15V
D14
D11
D12
D13
+15V
C16
4700uF/50V
C17
4700uF/50V
C22
2.2uF/16V
C20
2.2uF/16V
LM7812CK
U6
IN
OUT
GND
+12V
C21
1uF/16V
LM7912CK
U7
GND
IN
OUT
-12V
C23
1uF/16V
+12V
+5V
U8
IN
OUT
GND
LM7805CT
C19
2.2uF/16V
C24
1uF/16V

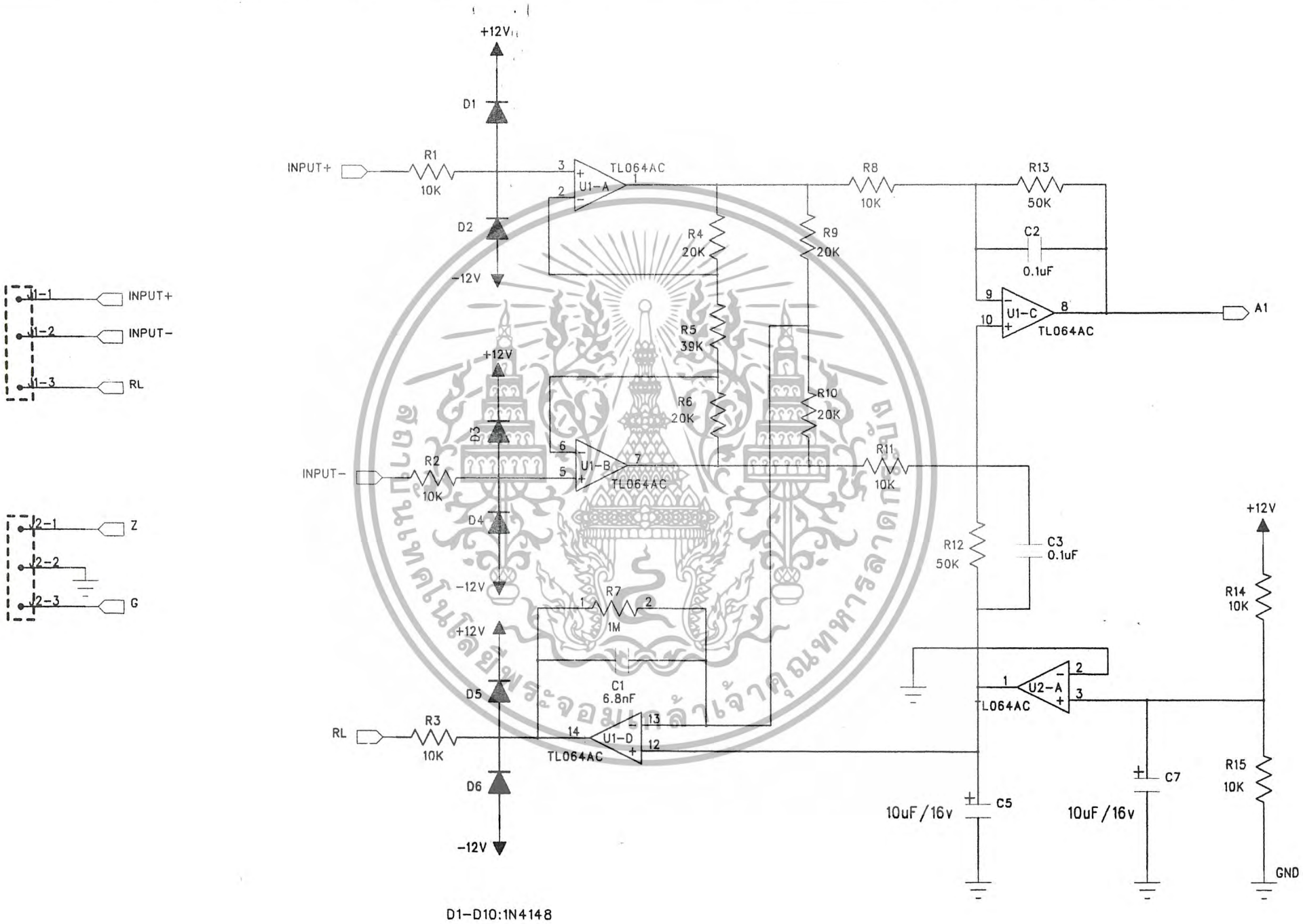

E
R52
10K
R51
10K
R53
10K
U4-C
TL064AC
D9
+12V
3
CW
R55
1
2
-12V
R54
10K
C18
1uF/16V
U4-D
TL064AC
U5-A
CD4001
G


+12V
3
CW
R49
Vr = 20K
1
2
-12V
Off Set
R46
10K
R48
100K
R50
50K
D
R45
10K
2
3
U4-A
TL064AC
1
R47
100K
6
5
U4-B
TL064AC
7
Z


C
R37
56K
R38
56K
C12
68nF
C13
68nF
10
9
U3-C
TL064AC
8
R39
27K
R40
47K
D
R41
10K
1uF / 16v
C14
1uF / 16v
C15
R42
1K5
R43
33K
R44
33K
13
12
U3-D
14
TL064AC
QRS
E


R30
8K2
R28
10K
2
CW
1
3
amplitude
R29
Vr = 260
12
13
U2-D
14
TL064AC
R31
1K
R32
1
3
2
CW
frequency
Vr = 16K
U3-A
TL064AC
C4
6.8nF
B
R36
50K
5
6
U3-B
7
TL064AC
R33
3K
4N26
1
2
6
5
4
Q3
+12V
R34
10K
C
R35
1K
-12V


+12V
R18
1M
Q2
E
B
C
D7
Vr = 162K
R17
1
2
3
CW
R19
50K
R22
680K
C11
A1
470uF/16V
R16
10K
6
5
U2-B
7
TL064AC
C8
C9
C10
0.047uF
0.047uF
0.047uF
R23
4K7
R24
75K
10
9
U2-C
8
TL064AC
B
R25
68
D8
GND
R20
50K
C
B
Q1
E
C1
10uF/16V
R27
CW
Vr = 20K
R21
1M
GND
R26
2K2


J1-1
INPUT+
J1-2
INPUT-
J1-3
RL
J2-1
Z
J2-2
J2-3
G
+12V
D1
INPUT+
R1
10K
3
2
U1-A
TL064AC
1
D2
-12V
R4
20K
R9
20K
R8
10K
R13
50K
C2
0.1uF
9
10
U1-C
8
TL064AC
A1
R5
39K
R6
20K
R10
20K
+12V
D3
INPUT-
R2
10K
6
5
U1-B
7
TL064AC
R11
10K
D4
-12V
R12
50K
C3
0.1uF
+12V
R14
10K
R7
1M
1
2
+12V
D5
C1
6.8nF
U2-A
TL064AC
RL
R3
10K
14
13
12
U1-D
TL064AC
D6
-12V
C5
10uF/16v
C7
10uF/16v
R15
10K
GND
D1-D10:1N4148