สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

การแสดงคลื่นสัญญาณหัวใจผ่านจอ VGA

ECG Monitor with VGA

จัดทำโดย

1.นายทศพล พงษ์เจริญ

2.นายศักดิ์ชัย กระแสอินทร์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2545

เลขหมู่.....
เลขทะเบียน 50326
วัน,เดือน,ปี 29 มิ.ย. 2547

.b.....
.i.....

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การแสดงคลื่นสัญญาณหัวใจผ่านจอ VGA

ECG Monitor with VGA

จัดทำโดย

1.นายทศพล พงษ์เจริญ 43015210

2.นายศักดิ์ชัย กระแสอินทร์ 43015232

อาจารย์ที่ปรึกษา

ร.ศ .ดร. สุรพันธุ์ เอื้อไพบูลย์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2545

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปริญญานิพนธ์ปีการศึกษา 2545

ภาควิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การแสดงคลื่นสัญญาณหัวใจผ่านจอ VGA

จัดทำโดย

1.นายทศพล พงษ์เจริญ 43015210

2.นายศักดิ์ชัย กระแสอินทร์ 43015232

โครงงานนี้ได้รับการตรวจสอบแล้วพร้อมที่จะทำการสอบได้

.....

(ร.ศ.ดร. สุรพันธุ์เอื้อไพบูลย์)

อาจารย์ที่ปรึกษา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การแสดงผลคลื่นไฟฟ้าหัวใจผ่านจอ VGA

1.นายทศพล พงษ์เจริญ
2.นายศักดิ์ชัย กระแสอินทร์
รศ.ดร.สุรพันธุ์ เอื้อไพบูลย์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2545

บทคัดย่อ

แนวคิดในการนำเสนอการใช้งานของ FPGA ให้สามารถรองรับความคิดริเริ่มต่าง ๆ ในปัจจุบันเริ่มเห็นได้ชัดเจนมากขึ้น การนำเอา FPGA และการเขียนคำสั่งในรูปแบบการบรรยายเชิงพฤติกรรมโดยใช้ภาษา VHDL มาใช้ออกแบบวงจรที่มีคุณสมบัติเป็นเครื่องวัดสัญญาณคลื่นไฟฟ้าหัวใจ เป็นการออกแบบโดยมุ่งไปที่การทดแทนเครื่องวัดสัญญาณคลื่นไฟฟ้าหัวใจที่ใช้ทั่วไปในโรงพยาบาล ข้อได้เปรียบของการออกแบบนี้คือ การลดต้นทุนในการใช้เครื่องวัดที่มีอยู่แล้วใช้ชิพ FPGA มาทดแทน การแสดงผลผ่านจอมอนิเตอร์ของเครื่องคอมพิวเตอร์ซึ่งเป็นการหาได้ง่ายกว่าการจัดซื้ออุปกรณ์เครื่องวัดสัญญาณคลื่นหัวใจจากต่างประเทศที่มีราคาสูง เป็นประโยชน์ต่อการลดทุนขององค์กรได้มาก ดังนั้นข้อสำคัญของการออกแบบคือ การออกแบบให้เครื่องวัดสัญญาณคลื่นหัวใจมีประสิทธิภาพสามารถนำไปใช้งานได้จริง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ECG Monitor with VGA

Mr.Thosapol Pongjareon

Mr.Sakchai Krasae-in

Assoc. Prof.Dr. Surapan Airphaiboon (Advisor)

Education Year 2002

Abstract

Nowadays, the process about designing the circuits with FPGA is going to be useful and more popular. So the FPGA and the behavior description language via VHDL are used to design an ECG Monitor in this project. This has a benefit in decreasing the demand of import ECG Monitor. By using the VGA Monitor the cost of product is very low. Howeve the designing of this project should have the performance for operating.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

หน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

หน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูปภาพ

หน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูปภาพ(ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูปภาพ(ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
บทนำ

1.1 ความเป็นมา

ทุกวันนี้เครื่องมือวัดทางการแพทย์ที่ใช้ในประเทศไทย โดยรวมแล้วยังคงต้องจัดซื้อจากต่างประเทศ ซึ่งใช้งบประมาณในการจัดซื้อจำนวนมหาศาล เนื่องมาจากการผลิตอุปกรณ์ เครื่องมือแพทย์ที่ใช้ในการวัดในต่างประเทศที่มีคุณภาพ เพราะการตรวจดูแลคนไข้ของแพทย์จำเป็นต้องอาศัยเครื่องมือที่มีความถูกต้อง และความแม่นยำสูงมาก ซึ่งศักยภาพของผู้ผลิตเครื่องมือวัดในประเทศไทยยังไม่สามารถตอบสนองได้ความถูกต้องของผลการวัดที่ไม่น่าเชื่อถือ จึงเป็นสาเหตุสำคัญที่ต้องพึ่งอุปกรณ์เหล่านี้จากต่างประเทศ ซึ่งมีความจำเป็นที่ต้องใช้งบประมาณที่มากในการจัดซื้อ

ECG Monitor (Electrocardiograph: ECG) เป็นอุปกรณ์ที่มีความจำเป็นในการดูแลคนไข้และเป็นอุปกรณ์ที่ต้องมีจำนวนอยู่มากพอสมควร ซึ่งอุปกรณ์ชนิดนี้จะแสดงสัญญาณทางไฟฟ้าที่ร่างกายมนุษย์ผลิตออกมา เพื่อบอกให้แพทย์ผู้รักษาสามารถวินิจฉัยการเปลี่ยนแปลงทางคลื่นไฟฟ้าต่างๆเหล่านี้เพื่อโยงไปหาสาเหตุอาการป่วยของคนไข้ได้แต่เนื่องจากเครื่องมือวัดชนิดนี้ยังมีราคาที่สูง การใช้งานของเครื่องมือที่มีประสิทธิภาพที่สูงชนิดนี้ จึงมักจะพบในโรงพยาบาลที่ใหญ่และของเอกชน แต่บางแห่งยังคงใช้เครื่องที่มีสภาพการใช้งานมานาน เมื่อต้องการจะเปลี่ยนให้เป็นเครื่องใหม่นั้นจะต้องมีการพิจารณาในแง่ของความจำเป็น และงบประมาณที่สามารถจะซื้อได้ จึงเป็นสาเหตุที่โรงพยาบาลหลายแห่งยังขาดเครื่องวัดคลื่นไฟฟ้าหัวใจที่มีคุณภาพอยู่

แนวความคิดที่ต้องการผลิต ECG Monitor จึงมีออกมาโดยปริญญานิพนธ์นี้ได้นำการออกแบบวงจรแบบการบรรยายพฤติกรรมเป็นภาษา VHDL ใช้งานร่วมกับ ชิพ FPGA และสามารถแสดงผลผ่านจอมอนิเตอร์มาร่วมใช้งาน

1.2 วัตถุประสงค์

1. เพื่อศึกษาลักษณะของคลื่นหัวใจและความผิดปรกติของสัญญาณคลื่นหัวใจ
2. เพื่อการออกแบบและการสร้างเครื่องแสดงสัญญาณคลื่นหัวใจด้วย FPGA และจอมอนิเตอร์
3. เพื่อสร้างเครื่องมือแพทย์ที่มีราคาถูกและสามารถใช้งานได้จริง
4. เพื่อศึกษาการออกแบบวงจรโดยใช้ชิพ FPGA ที่สามารถปรับปรุงคุณสมบัติ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การออกแบบโดยไม่ต้องเปลี่ยนแปลงวงจรทางกายภาพแต่สามารถเปลี่ยนแปลงโปรแกรมการออกแบบได้

1.3 เนื้อหาของโครงงาน

ในการสร้างโครงงานนี้มีเนื้อหาต่างๆที่สำคัญ ซึ่งแยกได้เป็นบทดังนี้

บทที่ 2 กล่าวถึงทฤษฎีการทำงานของคลื่นหัวใจ การเกิดคลื่นไฟฟ้า การศึกษารูปร่างของสัญญาณคลื่นไฟฟ้าหัวใจ การวิเคราะห์คลื่นไฟฟ้าหัวใจ ความผิดปรกติของจังหวะการเต้นของหัวใจ การตรวจวัดคลื่นไฟฟ้าของหัวใจ

บทที่ 3 กล่าวถึงทฤษฎีของการสร้างภาพของจอมอนิเตอร์ การสร้างสีทางแสง หลักการเกิดภาพและสัญญาณต่างๆที่จำเป็นในการสร้างภาพ

บทที่ 4 กล่าวถึงความเป็นมาของชิพ FPGA และชิพประเภท ASIC การแบ่งประเภทชิพที่ใช้ในปัจจุบัน

บทที่ 5 กล่าวถึงความเป็นมาของภาษา VHDL ลักษณะต่างๆของภาษาประเภทนี้ คุณสมบัติที่สำคัญ และรูปแบบการเขียนพร้อมตัวอย่าง

บทที่ 6 กล่าวถึงการออกแบบและการสร้างโดยการเขียนภาษาแบบบรรยายพฤติกรรม โดยใช้โปรแกรม MAX+plus® II เริ่มจากการสร้างวงจรที่รับสัญญาณคลื่นหัวใจมาเก็บไว้ในอุปกรณ์ชิพ FPGA แล้วประมวลสัญญาณเหล่านี้มาใช้ในการสร้างภาพออกจอมอนิเตอร์

บทที่ 7 เป็นการทดลองและการวัดค่าต่างๆออกมา

บทที่ 8 เป็นการสรุปในตอนท้าย ระบุถึงปัญหาและอุปสรรคในการทำงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2
สัญญาณไฟฟ้าหัวใจและหลักการแสดงคลื่นหัวใจ

2.1 บทนำ

เพื่อเป็นพื้นของความเข้าใจแนวความคิดและการทำงานของเครื่องวัดสัญญาณคลื่นไฟฟ้าหัวใจ ในบทนี้จะกล่าวถึงธรรมชาติของคลื่นไฟฟ้าหัวใจเสียก่อน

2.2 ธรรมชาติของคลื่นไฟฟ้าหัวใจ

คลื่นหัวใจมีแหล่งกำเนิดเป็นเป็นสัญญาณอิมพัลส์ที่บริเวณ SA node (Sinoartrial node) ในบริเวณผังหัวใจซีกบนขวา(right atrium) และการกระจายพร้อมกับรีโพลาไรซ์ไปทั่วร่างกาย เมื่อวัดโดยการต่อขั้วไฟฟ้าเข้าแบบดิฟเฟอร์เรนเชียลเข้ากับผิวหนังบริเวณหน้าอก แขน ขา ได้ลูกคลื่น 1 คาบเวลา คล้ายกับรูปที่แสดงในรูปที่ 2.1 ในคนปรกติ คลื่นไฟฟ้าหัวใจจะประกอบด้วยคลื่น P, QRS, T และ U ซึ่งลักษณะการมีอยู่ของคลื่นองค์ประกอบเหล่านี้ จะเป็นข้อมูลที่สำคัญในการวิเคราะห์การทำงานของหัวใจและความผิดปรกติ

เมื่อเกิดอิมพัลส์และเริ่มกระจายไปบริเวณซีกบนของหัวใจ จะทำให้เกิดการบีบตัวของกล้ามเนื้อด้านบนของหัวใจโลหิตในหัวใจห้องบนด้านขวาหรือด้านซ้ายนั้นจะถูกบีบลงไปด้างล่างของหัวใจ อิมพัลส์นี้จะถูกแพร่กระจายผ่านเนื้อเยื่อตัวนำมายังผิวหนังจะทำให้เกิดสัญญาณที่เรียกว่า P Wave

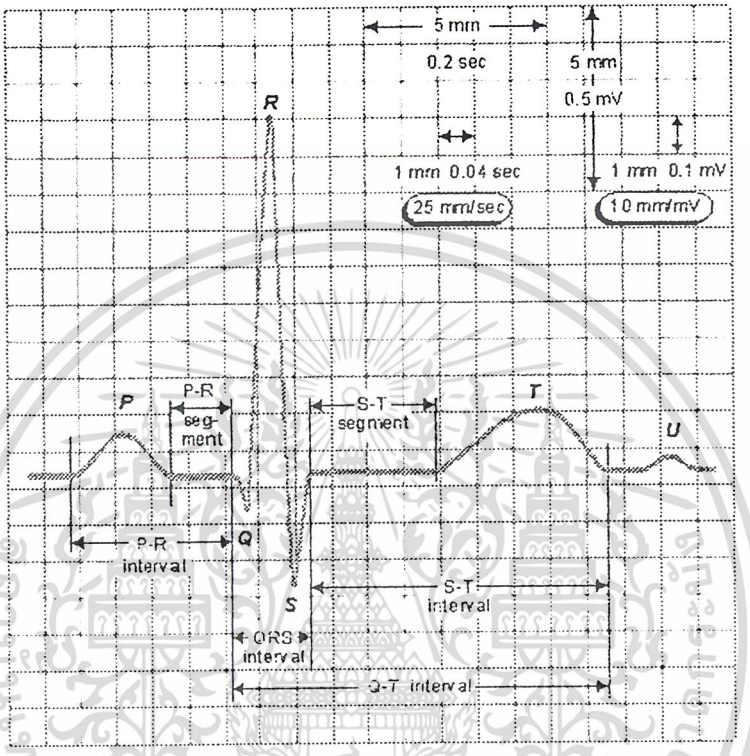
ในบริเวณ atrioventricular node จะมีการหน่วงเวลาของการกระตุ้นอิมพัลส์เพื่อให้เลือดสามารถถ่ายโอนจากด้านบนของหัวใจไปถึงด้านล่างของหัวใจเสร็จสมบูรณ์ การหน่งเวลานี้เป็นส่วนหลักของ P-R Interval ของรูปคลื่น ECG

อิมพัลส์ที่เกิดจะกระจายไปด้านล่างของหัวใจ ด้วยการคลายตัวของกล้ามเนื้อด้านล่างหัวใจ เมื่อขบวนการเสร็จสมบูรณ์จะมีคาบเวลาเกิดซ้ำขึ้นอีก และจะผลิตรูปคลื่น ECG ออกมาอีกครั้ง

จะเห็นว่าส่วนผลิตรูปคลื่น ECG จะนำข้อมูลทางพยาธิสภาพของหัวใจ มาสู่แพทย์เพื่อวินิจฉัยอาการผิดปรกติ ตัวอย่างเช่น R-R Interval สามารถบอกถึงอัตราการเต้นของหัวใจ(Cardiac Rhythm) ภายใต้อิทธิพลของระบบประสาทอัตโนมัติ ความไม่เสถียรของ Cardiac Rhythm สามารถบ่งถึงการเต้นที่ผิดปรกติ ขนาดและช่วงเวลาของ P และ QRS บ่งชี้ถึงสภาพกล้ามเนื้อหัวใจ การลดทอนขนาดสัญญาณ อาจจะบ่งบอกของการทำลายกล้ามเนื้อหัวใจบริเวณที่เกี่ยวข้อง ขนาดเพิ่มอาจบอกความผิดปรกติของหัวใจโต นอกจากนี้การหน่วงเวลานานเกินไปในจุด atrioventricular เป็น

เอกสารนี้มีลิขสิทธิ์ สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การบ่งบอกถึงการปิดกั้นของทั้งหมดหรือบางส่วนของอิมพัลส์ จากการขาดช่วงการซิงโครไนซ์ระหว่าง P-Wave และ QRS Complex อาการผิดปรกติสามารถรักษาได้ทางการใช้ยาและสังเกตผลการรักษาจากรูปคลื่น ECG



รูปที่ 2.1 แสดงคลื่นไฟฟ้าหัวใจของคนปรกติ

บนรูปคลื่น ECG จะมีสัญญาณไฟฟ้าร่างกายชนิดอื่นๆ ปะปนมาด้วยเช่นสัญญาณ EEG จากสมอง สัญญาณ EMG จากกล้ามเนื้อ รวมทั้ง Motion Artifact อย่างไรก็ดีทางเทคนิคการแยกสัญญาณ ECG ออกมาจากสัญญาณเหล่านี้ได้ไม่ยากเนื่องจากคลื่นฟฟ้าหัวใจมีความถี่สเปกตรัมในช่วง 3-40Hz อุปกรณ์แสดงรูปคลื่น ECG ทางการแพทย์จะมีแบนวิดท์ของการตอบสนองความถี่สำหรับการประยุกต์การใช้งานที่แตกต่างกันแบบที่ใช้งานในคลีนิคที่ใช้สำหรับบันทึกมาตรฐาน 12 Lead ECG คือ 0.05-100Hz ในกรณีที่คนไข้มีอาการทรุดหนัก แบนด์วิดท์ของเครื่องวัดจะกำหนดไว้ที่ 0.5-50Hz สำหรับการวัดของอัตราการเต้นของหัวใจที่ใช้ทดสอบ QRS Complex จะตัดสัญญาณรบกวน P-Wave และ T-Wave ออกจาก ECG

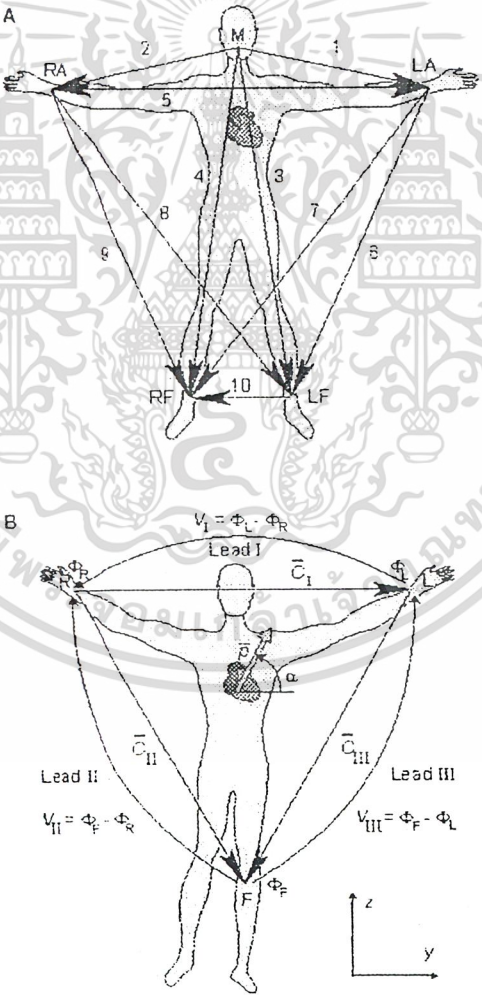
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.3 การวัดคลื่นไฟฟ้าหัวใจ

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

การแบบคาร์ดิโอกราฟ คือการวัดการเปลี่ยนแปลงขนาดของเวกเตอร์ ของความต่างศักดิ์ที่เกิดขึ้น บนแกนหนึ่งเทียบอีกแกนหนึ่ง โดยพิจารณาจาก 3 แกนที่ตั้งฉากกันมีอยู่ด้วยกัน 3 ระนาบ คือ ระนาบที่มองทางด้านหน้า ด้านซ้าย และด้านบน วิธีนี้ต้องใช้แพทย์ผู้เชี่ยวชาญในการวินิจฉัย

การวัดแบบอิเล็กโตรคาร์ดิโอกราฟ คือการวัดการเปลี่ยนแปลงขนาดของเวกเตอร์ของความต่างศักดิ์ที่เกิดขึ้นในแนวแกนใดๆเทียบกับเวลา สามารถวินิจฉัยได้ง่าย



รูปที่ 2.2 ตำแหน่งการวัดคลื่นหัวใจทั้ง 12 ลีดมาตราฐาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

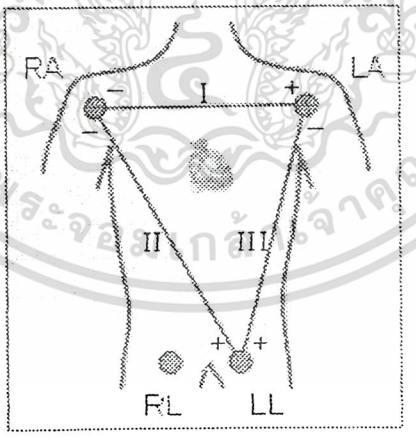
การวัดคลื่นหัวใจแบบอิเล็กโตรคาร์ดิโอกราฟมีเพื่อการวินิจฉัยระบบการทำงานของหัวใจสามารถแบ่งตามวัตถุประสงค์ของการวัดได้ 2 ประเภทคือ เพื่อการวินิจฉัยคนไข้ข้างเตียงแบบมาตราฐาน(Standard Clinical ECG) และการวัดเพื่อการมอนิเตอร์ (Monitoring ECG)

2.3.1 การวัดเพื่อวินิจฉัยคนไข้ข้างเตียงแบบมาตราฐาน

การวัดเพื่อวินิจฉัยคนไข้ข้างเตียงแบบมาตราฐานนั้น เป็นการวัดคลื่นไฟฟ้าหัวใจโดยตำแหน่งที่เป็นการวัดได้กำหนดไว้เป็นมาตราฐานแล้ว วิธีการวัดเพื่อวินิจฉัยคนไข้ข้างเตียงสามารถแบ่งออกได้เป็น 3 วิธีคือ วิธีการวัดแบบ Standard Limb Lead วิธีการวัดแบบ Augment Limb Lead และวิธีการวัดแบบ Unipolar Chest Lead ซึ่งสามารถอธิบายได้ดังนี้

2.3.1.1 วิธีการวัดแบบ Standard Limb Lead

วิธีการวัดแบบ Standard Limb Lead เป็นการวัดสัญญาณไฟฟ้าหัวประกอบไปด้วย Lead I, II และ III ดังรูปที่ 2.3 ซึ่งสามารถทำการวัดคลื่นไฟฟ้าหัวใจแบบ Standard Limb Lead ทั้ง Lead I,II และ III โดยการติดขั้ววัดของวงจรขยายค่าความแตกต่าง



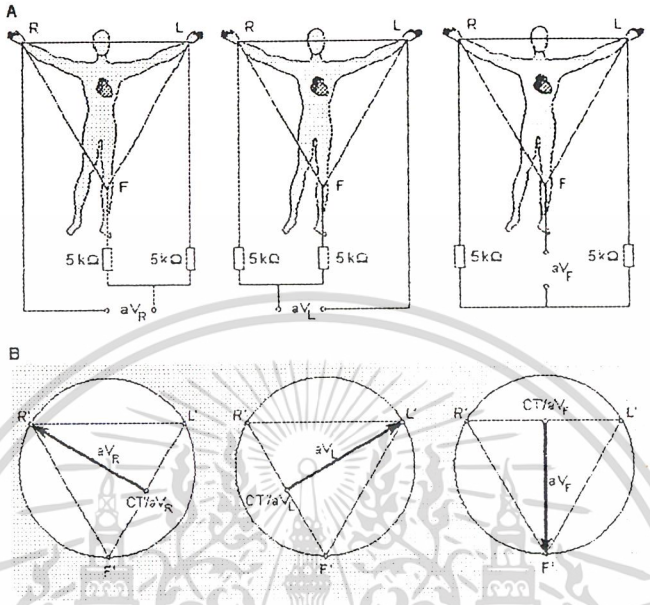
รูปที่ 2.3 วิธีการวัดแบบ Standard Limb Lead

2.3.1.2 วิธีการวัดแบบ Augment Limb Lead

วิธีการวัดแบบ Augment Limb Lead เป็นวิธการวัดคลื่นหัวใจที่ประกอบด้วย Lead aVR, Lead aVL และLead aVF ดังรูปที่ 2.4 สำหรับวัดสัญญาณไฟฟ้าหัวใจ แบบ Augment Limb Lead จะ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

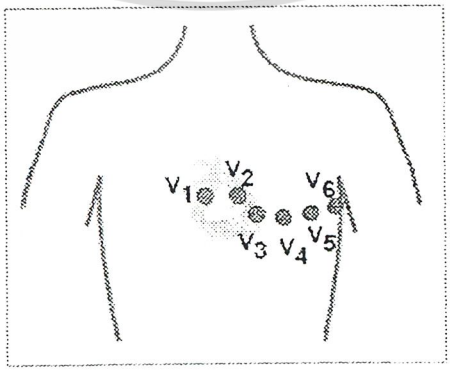
มีตัวต้านทานค่า R/2 ต่อที่ขั้วบวกของวงจรขยายค่าความแตกต่าง มีไว้เพื่อสมดุลย์ของความต้านทานของอินพุทของวงจรขยายค่าความแตกต่าง



รูปที่ 2.4 วิธีการวัดแบบ Augment Limb Lead

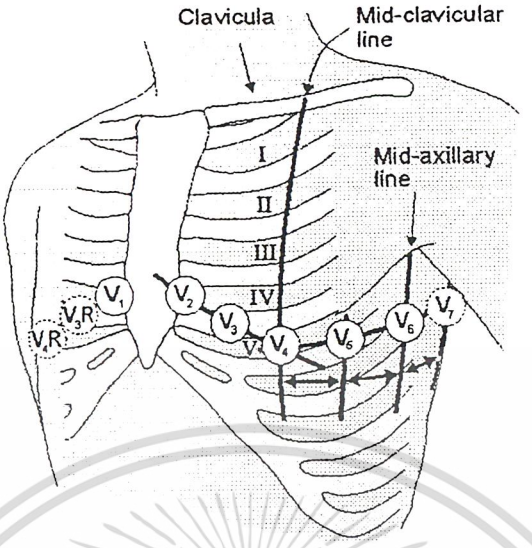
2.3.1.3 วิธีการวัดแบบ Unipolar Chest Lead

วิธีการวัดแบบ Unipolar Chest Lead เป็นการวัดขนาดสัญญาณไฟฟ้าหัวใจระหว่างตำแหน่งใดๆบนหน้าอก(ขั้ววัดบวก)เทียบกับค่าเฉลี่ยของความต่างศักดิ์ของตำแหน่ง RA, LA และ LL วิธีการวัดในรูปที่ 2.5 วิธีนี้ประกอบด้วย 6ลีด คือ Lead V1 ถึง V6 คือการกำหนดตำแหน่งของขั้วบวกอยู่ในบริเวณต่างๆบริเวณหน้าอก 6 ตำแหน่ง แสดงในรูปที่ 2.6และ 2.7

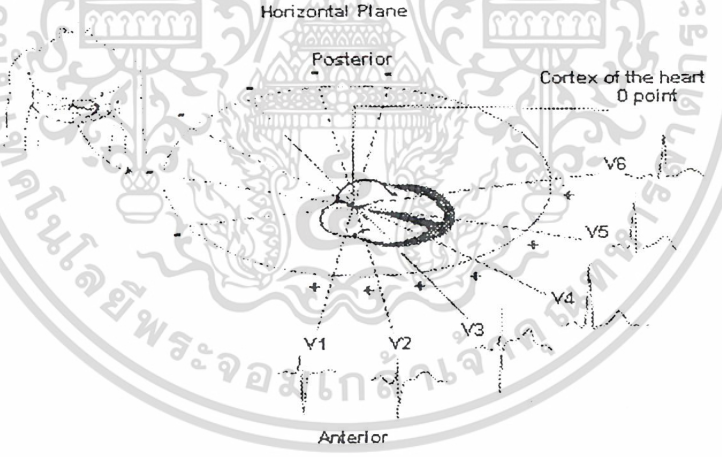


รูปที่ 2.5 วิธีการวัดคลื่นหัวใจไฟฟ้าแบบ Unipolar Chest Lead

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่2.6 ตำแหน่งการติดตั้งขั้ววัดบวกวิธีการวัดแบบ Unipolar Chest Lead



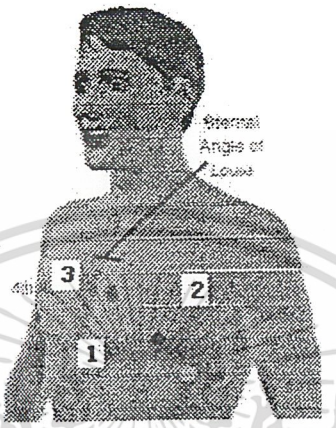
รูปที่ 2.7 ตัวอย่างคลื่นไฟฟ้าหัวใจแบบ Unipolar Chest Lead V1 ถึง V6

2.4 การวัดเพื่อการมอนิเตอร์

การวัดแบบมอนิเตอร์ใช้วัดสัญญาณไฟฟ้าหัวใจในขณะที่มีการเคลื่อนย้ายผู้ป่วยเพื่อตรวจสอบจังหวะและอัตราการเต้นของหัวใจ ตำแหน่งที่วัดควรเป็นตำแหน่งที่ให้สัญญาณ R แรงมากเพื่อให้อัตราการเต้นของหัวใจต่อสัญญาณรบกวน(Signal to noise Ratio: S/N) มีค่าสูง ตำแหน่งของการวัดมอนิเตอร์แสดงได้ดังรูปที่ 2.7 โดยการติดขั้วบวกไว้ที่ตำแหน่ง V1 ของUnipolar Chest Lead

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

(ตำแหน่งหมายเลข 1) และติดขั้วลบไว้ใกล้ไหล่ซ้าย(ตำแหน่งหมายเลข 2) และติดขั้ววัดใช้อ้างอิงที่ตำแหน่งใดๆบริเวณหน้าอก(ตำแหน่งหมายเลข 3) ลักษณะของคลื่นหัวใจที่วัดได้จะใกล้เคียงกับ V1 ของ Unipolar Chest Lead ซึ่งเป็นสัญญาณที่ใช้ในการคำนวณอัตราการเต้นของหัวใจ



รูปที่ 2.8 ตำแหน่งการติดขั้ววัดหัวใจของวิธีการวัดคลื่นไฟฟ้าเพื่อการมอนิเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3
หลักการเกิดภาพในจอมอนิเตอร์

3.1 ทฤษฎีพื้นฐานของแสงและสี

แสงโดยทั่วไปมี 2 ประเภท คือแสงที่สายตามนุษย์มองเห็น(visible rays) และแสงที่สายตาของมนุษย์ที่ไม่สามารถมองเห็นได้ซึ่งได้แก่แสงจำพวก รังสีแกมมา(gramma-rays), รังสีเอกซ์ (X-rays), รังสีอัลตร้าไวโอเลท(ultraviolet rays) และรังสีอินฟราเรด(infrared rays) แสงเหล่านี้ต่างก็เป็นพลังงานทางคลื่นแม่เหล็กไฟฟ้า แต่มีความถี่หรือความยาวคลื่นแตกต่างกันออกไป ส่วนของคลื่นแม่เหล็กไฟฟ้าที่น่าสนใจของจอมอนิเตอร์(monitor)ที่แสดงออกมานี้ ก็คือส่วนของคลื่นแม่เหล็กไฟฟ้าที่มนุษย์สามารถมองเห็นได้ซึ่งมีความยาวคลื่นประมาณ 380 นาโนเมตรจนถึง 780 นาโนเมตร แสงที่มองเห็นนี้จะทำให้ตาของมนุษย์ได้ความรู้สึก 2 ประการคือ 1. ความรู้สึกว่าแสงมีความสว่างมากน้อยเพียงไร (sensation of brightness)2. ความรู้สึกที่สามารถแยกแยะว่าเป็นสีอะไร (sensation of color)

3.2 ความสำคัญ 3 ประการของแสงที่มองเห็น

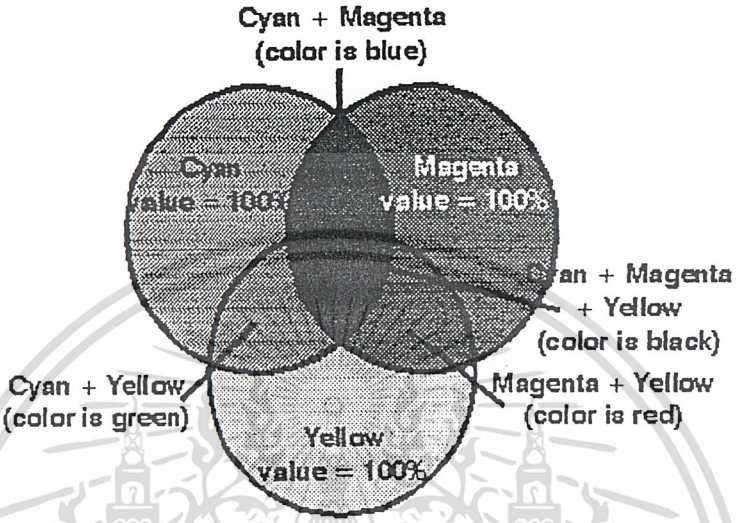
แสงที่มองเห็นจะทำให้เกิดความรู้สึกที่สำคัญ 3 ประการคือ 1.เกิดความรู้สึกในเรื่องของแสงสี(hue) 2. เกิดความรู้สึกในเรื่องการส่องสว่าง (brightness) และ 3. เกิดความรู้สึกในเรื่องแสงสีอิ่มตัว(saturation) โดยความรู้สึกในด้านแสงสี(hue) จะทำให้ตาของมนุษย์สามารถแยกแยะออกได้ว่าแสงที่มองเห็นได้นั้นเป็นแสงสีอะไรเป็นสีแดง สีเขียวหรือสีน้ำเงิน เป็นต้น ส่วนความรู้สึกในเรื่องการส่องสว่าง(brightness) เป็นความรู้สึกที่บอกว่าแสงที่ตามองเห็นนี้มีความสว่างหรือมืด สำหรับความรู้สึกทางด้านแสงสีอิ่มตัว(saturation of chroma) จะทำให้สามารถรู้ความบริสุทธิ์ว่าแสงสีที่เห็นเป็นแสงสีที่มีชัดเจน หรือจาง

3.3 การผสมและการแยกสี

การผสมแสงสีมีลักษณะแตกต่างไปจากการผสมสีทางการวาดภาพ การผสมสีโดยทั่วไปเช่นในการวาดเขียนหรือการพิมพ์ภาพสี จะทำให้ได้สีผสมที่มีความเข้มข้นกว่า ซึ่งเป็นลักษณะวิธีของวิธี Subtractive mixer ส่วนการผสมสีทางแสงนั้น จะทำให้แสงสีที่ผสมมีลักษณะเจือจางกว่าแม่สีเดิม อันเป็นลักษณะวิธีของ Additive mixer ดังรูปที่ 3.1 และการผสมแสงสีที่ใช้ในการสร้างภาพ

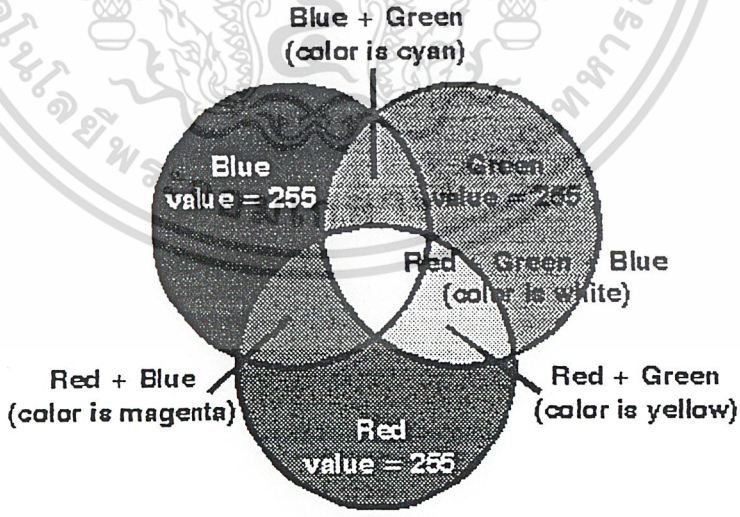
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ของจอมอนิเตอร์นั้น จะเป็นการผสมแม่สี(primary color) เพื่อทำให้เกิดแสงสีต่างๆขึ้น สีที่เป็นแม่สีนี้ต้องเป็นสีที่ไม่สามารถเกิดจากการผสมสี ซึ่งในระบบโทรทัศน์และมอนิเตอร์จะมีแม่สีอยู่ 3 สีคือ 1. สีแดง 2.สีเขียวและ3.สีน้ำเงิน



Reflective colors or the colors of inks.
Primaries are Cyan, Magenta, and Yellow and all three together add up to black.

รูปที่ 3.1 การผสมสีแบบ Subtractive Mixer



Transmission colors or the colors of light.
Primaries are Red, Blue, and Green and all three together add up to white.

รูปที่ 3.2 การผสมสีแบบ Additive Mixer

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แสงสีต่างๆที่เราสามารถมองเห็นได้นั้น เกิดจากการผสมสีทางแม่สีทั้งนั้น ซึ่งอาจเกิดจากการนำสีใดสีหนึ่งไปผสมกับสีใดสีหนึ่ง หรือการนำสีใดสีหนึ่งไปหักจากสีใดสีหนึ่งก็ได้ ยกตัวอย่างเช่น

แสงสีเหลือง(yellow) = แสงสีแดง(red) + แสงสีเขียว(green)

แสงสีเขียวน้ำเงิน(cyan) = แสงสีเขียว(green) + แสงสีน้ำเงิน(blue)

แสงสีม่วง(magenta) = แสงสีน้ำเงิน(blue) + แสงสีแดง(red)

แสวสีขาว(white) = แสงสีแดง(red)+แสงสีเขียว(green)+แสงสีน้ำเงิน+(blue)

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

ตารางแสดงความยาวคลื่นของแม่สีทั้งสามในระบบแสงสีแดงสีเขียวและแสงสีน้ำเงิน(RGB)

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

3.4 องค์ประกอบของภาพ

หากเราตัดภาพจากหนังสือพิมพ์ภาพหนึ่งแล้วขยายด้วยกล้องหรือแว่นขยาย จะพบว่าภาพที่ได้นั้นมีองค์ประกอบมาจากจุดสีขาวและจุดสีดำมากมาย มาเรียงประกอบกันขึ้นเป็นภาพ จุดเหล่านี้เองที่เรียกว่าองค์ประกอบของภาพ(Picture Element) หรือ PIXEL นั่นเอง

ในทำนองเดียวกันภาพที่ปรากฏทางด้านจอมอนิเตอร์ก็นำหลักการข้างต้นนี้ ภาพที่เกิดมาจากเส้นขวางเล็กๆวางเรียงกันในแนวนอนจำนวนมาก บนจอภาพแต่ละเส้นนั้นมีส่วนที่ดำสนิท ส่วนที่จางและส่วนที่สว่างวางเรียงกันอยู่ เส้นเหล่านี้ได้มาจากการกวาดลำเส้นของแสง(scan) ความแตกต่างกันบนเส้นกวาดลำแสงหรือเส้นสแกนเหล่านี้เองเราจัดว่าเป็นองค์ประกอบของภาพ

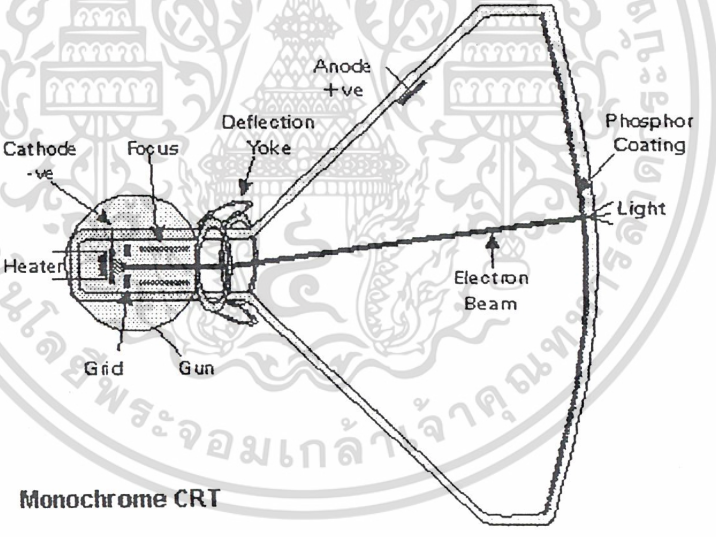
จอภาพทำงานโดยการแสดงภาพ ซึ่งอาจเป็นภาพกราฟิกหรือตัวอักษร ซึ่งเกิดจากการประมวลผลของการ์ดวีจีเอจอภาพแบ่งเป็น 2 ประเภท คือจอภาพสีเดียวหรือจอภาพโมโนโครม (Monochrome) และจอสี (Color Monitor) ปัจจุบันจอภาพสีเดียวนั้นไม่เป็นที่นิยมใช้กับคอมพิวเตอร์ หากจะมีใช้ก็เฉพาะงานเฉพาะอย่างเท่านั้น ส่วนที่นิยมใช้ก็คือจอสี โดยแบ่งได้อีกเป็น 3 ประเภท คือจอสีวีจีเอ (VGA = Video Graphics Array) และจอสี Super VGA (SVGA = Super

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Video Graphics Array) และจอ LCD (Liquid Crystal Display) ซึ่งประเภทหลังนี้มีราคาแพงมาก จอภาพที่ได้รับความนิยมในปัจจุบันคือจอ SVGA เนื่องจากมีราคาไม่แพงมากนัก และเหมาะกับ Application ที่ออกแบบให้มีความสามารถแสดงภาพกราฟิก นอกจากนี้ Application ประเภทมัลติมีเดียหรือเกมต่างๆ ต่างก็ต้องการจอภาพที่มีความละเอียดสูง (High Resolution) สามารถแสดงสีได้หลายๆสี

3.5 การทำงานของจอภาพ

จอภาพมีหลักการทำงานแบบเดียวกับจอโทรทัศน์โดยจะมีกระแสไฟฟ้าแรงสูง (High Voltage) คอยกระตุ้นให้อิเล็กตรอนภายในหลอดภาพแตกตัว อิเล็กตรอนดังกล่าวจะทำให้เกิดลำแสงอิเล็กตรอนไปกระตุ้นผลึกฟอสฟอรัสที่ฉาบอยู่บนหลอดภาพ เมื่อฟอสฟอรัสถูกกระตุ้นจากอิเล็กตรอนจะเกิดการเรืองแสงและปรากฏเป็นจุดสีต่างๆ (RGB Color) ซึ่งรวมเป็นภาพบนจอภาพนั่นเอง



รูปที่3.3 โครงสร้างของหลอด Cathode Ray Tube

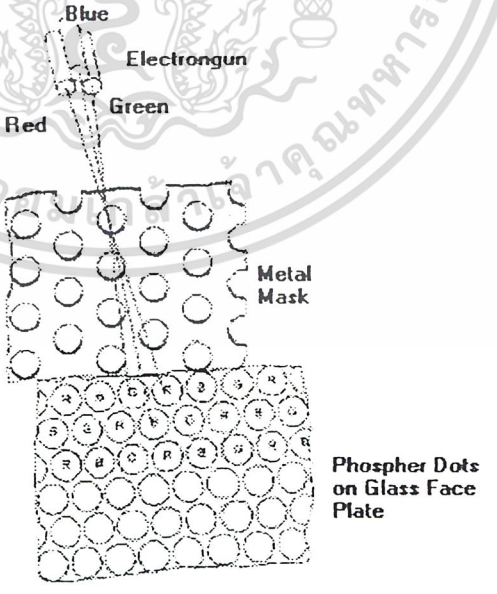
เพื่อความเข้าใจในการกำเนิดสัญญาณภาพ อันดับแรกเราต้องเข้าใจกระบวนการสร้างภาพบนผิวหลอดภาพก่อน สัญญาณภาพVGA (VGA video signal) มีส่วนประกอบของสัญญาณที่สำคัญ 5 สัญญาณ จะมี 2 สัญญาณที่เป็นระดับสัญญาณแบบ TTL Logic มีสัญญาณ Horizontal sync (Hor sync) และสัญญาณ Vertical sync (Ver sync) สัญญาณทั้ง 2 จะถูกใช้ในการเข้าจังหวะของภาพ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ส่วนสัญญาณอีก 3 สัญญาณที่เหลือจะเป็นสัญญาณสีของแม่สีที่ลักษณะของสัญญาณจะเป็นระดับสัญญาณทางอนาล็อกมีขนาดตั้งแต่ 0.7-1 โวล์ท พีค-พีค(peak to peak) หรือเป็นสัญญาณควบคุมการเกิดสีเรียกว่าสัญญาณ RGB ประกอบด้วยสัญญาณ สีแดง(Red) สัญญาณสีเขียว(Green) และสัญญาณสีน้ำเงิน(Blue) ซึ่งการเปลี่ยนระดับอนาล็อกของสัญญาณทั้ง 3 จะเป็นตัวกำหนดการเกิดสีบนผิวจอภาพ

3.6 เทคโนโลยีการสร้างภาพ

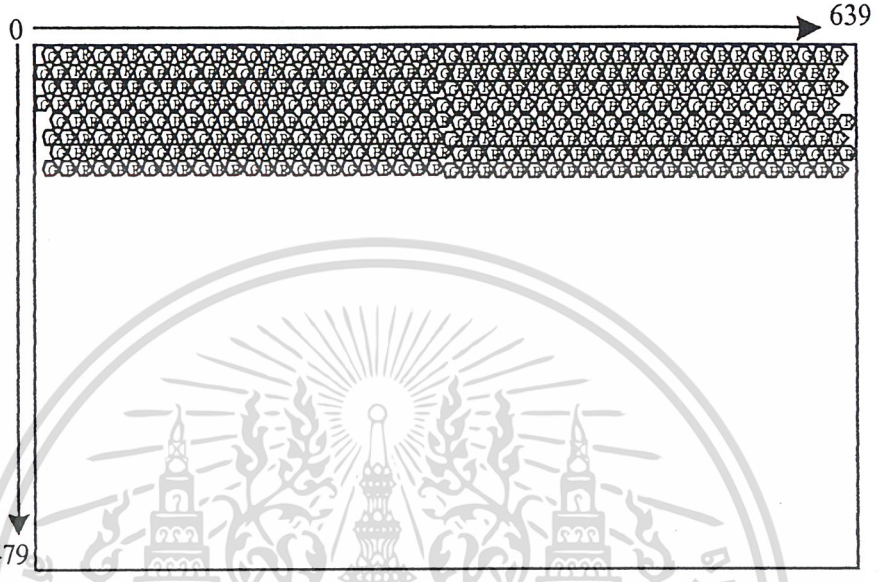
อุปกรณ์หลักของการสร้างภาพคือ หลอดภาพ Cathode Ray Tube (CRT) การทำงานเริ่มจากการยิงลำอิเล็กตรอนจากก้นของหลอดภาพไปสู่ผิวจอภาพเป็นการกวาด(scan) ภาพตามแนวนอน(horizontal) โดยเริ่มจากส่วนบนของหลอดภาพเริ่ม horizontal line เส้นที่ 1 โดยการหักเหของลำอิเล็กตรอนนั้นจะเกิดจาก Deflection Yoke เป็นลักษณะขดลวดที่อยู่ทางก้นหลอด อาศัยสนามแม่เหล็กหรือสนามไฟฟ้าสถิตควบคุมการเบี่ยงเบนลำอิเล็กตรอนเรียกว่า การกวาดภาพ(scan) ส่วนการเกิดสีต่างๆนั้น จะเป็นการควบคุมขนาดของสัญญาณ RGB ทั้ง 3 สัญญาณเพื่อไปควบคุมความเข้มของลำอิเล็กตรอนที่วิ่งไปกระทบกับผิวของจอภาพซึ่งเคลือบด้วยสารฟอสเฟอร์อยู่ก่อนแล้ว สารฟอสเฟอร์นั้นเมื่อได้รับพลังงานจากการชนของอิเล็กตรอนจะทำให้เกิดการเรืองแสง ซึ่งการเรืองแสงของจอ และลักษณะของจอจะแสดงดังรูปที่ 3.4



รูปที่ 3.4 ลักษณะการยิงลำอิเล็กตรอนและโครงสร้างของผิวมอนิเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

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



รูปที่ 3.5 แสดงจำนวนของ Pixel ขนาด 480x640 ของจอมอนิเตอร์

ในการกวาดสัญญาณแต่ละ Pixel นั้น สัญญาณ RGB จะเปลี่ยนค่าทำให้เกิดสีในแต่ละ Pixel มีค่าต่างกันเมื่อมองดูแล้วจะเกิดเป็นภาพออกมา

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

จากการทำงานใน 1 Frame จะได้จำนวน Pixel = 480x640

= 307200 Pixel

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

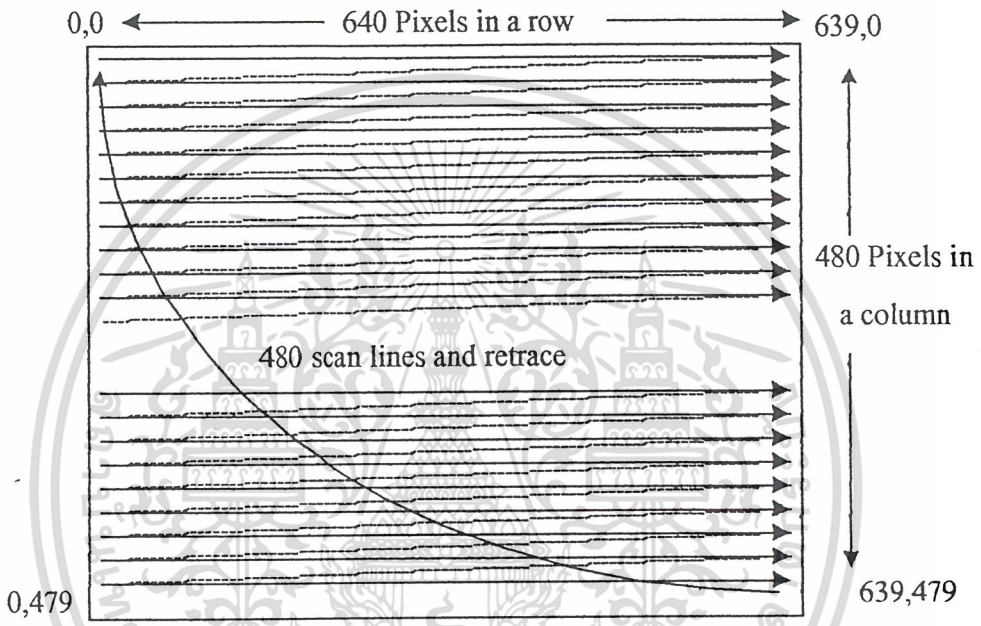
ใน 1 วินาทีจะเกิดFrame ทั้งหมด = 307200x60

= 1843200 Pixel

ดังนั้นใน 1 Pixel จะใช้เวลาในการเกิด = 1/(1843200)

= 54.253 nS

แต่เราได้ประมาณให้มีคาบเวลาเป็น 40 nS ซึ่งจะได้ความถี่ที่ใช้ประมวลผลเป็น 25 MHz

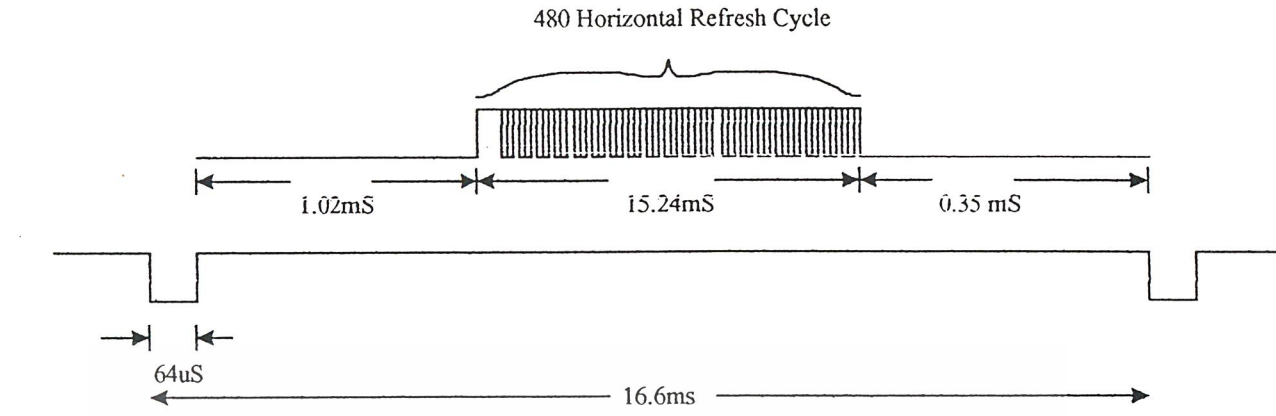


รูปที่ 3.6 รูปแบบการกวาดภาพ

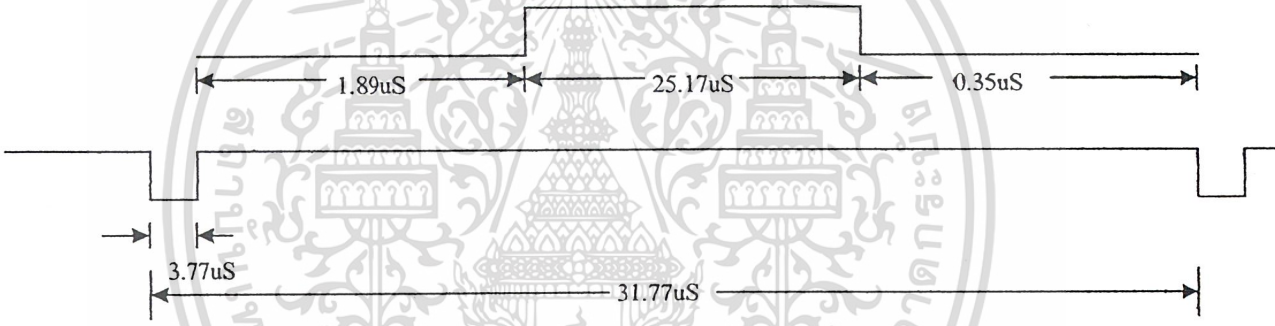
มาตราฐานของเวลาในการสร้างสัญญาณทั้งสัญญาณ Ver sync และสัญญาณ Hor syncจะปรากฏ ดังรูปที่ 3.8

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง



Vertical Sync Signal Timming



Horizontal Sync Signal Timming

รูปที่ 3.7 คาบเวลาในการสร้างสัญญาณ Ver Sync และ Hor Sync

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

50326

บทที่ 4
อุปกรณ์ FPGA

4.1 เทคโนโลยีASIC

ในช่วงก่อนทศวรรษ 1970 อุตสาหกรรมเซมิคอนดัคเตอร์ได้ถูกปลุกให้ตื่นตัวขึ้นหลังจากที่มีการประดิษฐ์วงจรรวมหรือไอซีตัวแรกสำเร็จ ในยุคแรกนั้นไอซีขนาดเล็กหรือ SSI (Small-Scale Integration) ประกอบไปด้วยเกทดิจิตอลจำนวนไม่มากนัก (ประมาณ 1 ถึง 10 ตัว) ต่อมาได้มีการเพิ่มปริมาณของเกทดิจิตอลและฟังก์ชันทางลอจิกให้มากขึ้นจนเป็น MSI (Medium-Scale Integration) การพัฒนาไอซีเป็นไปอย่างต่อเนื่องจนมาถึงยุคของ LSI (Large-Scale Integration) ซึ่งเป็นยุคที่มีการสร้างไมโครโปรเซสเซอร์ตัวแรกขึ้น และในปัจจุบันเป็นยุคของ VLSI (Very Large-Scale Integration) ซึ่งเทคโนโลยีในการสร้างไอซีรุดหน้าจนสามารถสร้างไมโครโปรเซสเซอร์ขนาด 64 บิต ที่มีหน่วยความจำแคชกับหน่วยคำนวณทางคณิตศาสตร์ของโฟล์ทติงพอยน์ (Floating-Point Arithmetic Units) รวมอยู่ภายในตัวมัน และเนื่องจากการปรับปรุงเทคโนโลยีของกระบวนการสร้างซีมอสที่มีมาอย่างต่อเนื่องทำให้ขนาดของทรานซิสเตอร์ที่บรรจุอยู่ภายในไอซีมีขนาดเล็กลงเรื่อยๆ จนบางคนโดยเฉพาะในญี่ปุ่นใช้คำว่า ULSI (Ultralarge Scale Integration) เพื่อใช้เรียกระดับของไอซีในปัจจุบัน แต่คนส่วนมากยังมักนิยมเรียกเพียงแค่ VLSI

จากการปรากฏตัวของ VLSI ในช่วงทศวรรษ 1980 ทำให้วิศวกรเริ่มมีการออกแบบไอซีตามความต้องการของลูกค้าซึ่งใช้ในระบบที่เจาะจงนอกเหนือจากการใช้ไอซีมาตรฐานเพียงอย่างเดียว โดยไอซีเหล่านี้มีชื่อเรียกว่า ASIC : Application-Specific Integrated Circuit ซึ่งตัวอย่างของ ASIC ได้แก่ชิพไอซีที่ใช้สำหรับตุ๊กตาของเล่นพูดได้ ดาวเทียม และชิพที่ภายในบรรจุด้วยไมโครโปรเซสเซอร์กับอุปกรณ์ทางลอจิกอื่นๆ

4.2 ประเภทของ ASIC

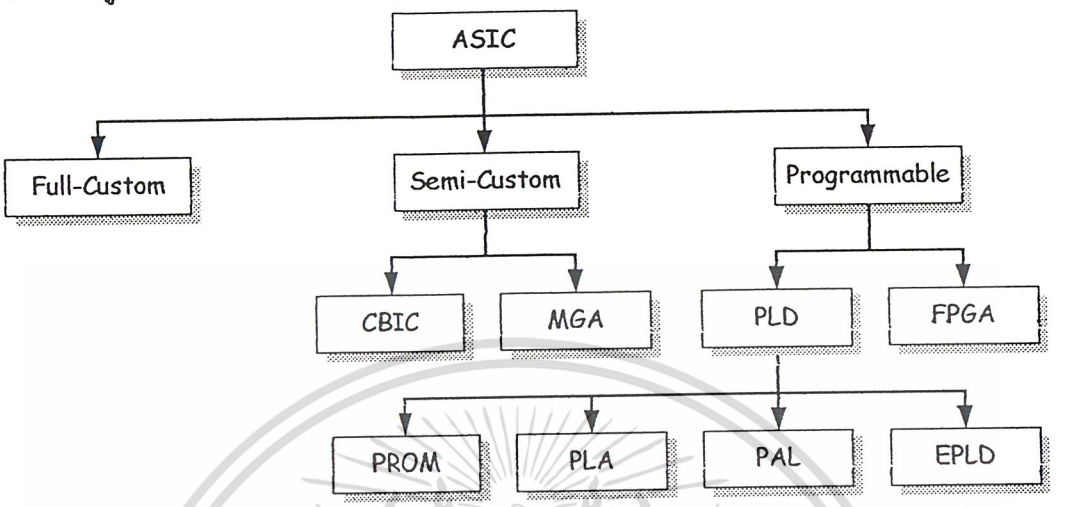
ASIC แบ่งเป็น 3 ประเภทใหญ่ๆ คือ Full-custom, Semi-custom และ Programmable ดังรูปที่ 4.1

4.2.1. Full-custom

ASIC ประเภทนี้ลูกค้าจะเป็นผู้ออกแบบเซลล์ลอจิก (เช่น แอนด์เกท ออร์เกท มัลติเพล็กเซอร์ และฟลิบฟล็อบ) และลักษณะการจัดวางอุปกรณ์บนตัวไอซีรวมถึงหน้ากากสำหรับควบคุม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การเจือและสร้างชั้นสาร (Mask) ต่างๆ ที่ใช้ในการทำไอซีเอง ดังนั้นค่าใช้จ่ายในการออกแบบและการผลิตจะสูงมาก



รูปที่ 4.1 ประเภทของ ASIC

4.2.2. Semi-custom

ASIC ประเภทนี้เซลล์ลอจิกจะถูกออกแบบเอาไว้ก่อนแล้วในรูปแบบของไลบรารีและลูกค้าจะเป็นผู้ออกแบบ Mask ต่างๆ เอง ตัวอย่างของไอซีประเภทนี้ได้แก่ Standard-Cell–Based ASIC และ Masked Gate-Array–Based ASIC

4.2.2.1 Standard-Cell–Based ASIC

ไอซีประเภทนี้จะมีพื้นที่สำหรับจัดวางเซลล์ลอจิกมาตรฐานซึ่งถูกออกแบบเอาไว้แล้ว ในบางครั้งเซลล์มาตรฐานเหล่านี้จะถูกนำมาประกอบกันเป็นเซลที่มีขนาดใหญ่ขึ้นเรียกว่า Megacell สำหรับการออกแบบนั้นผู้ออกแบบจะทำเพียงแค่กำหนดตำแหน่งของเซลล์มาตรฐานและการเชื่อมต่อภายในของแต่ละเซลล์เท่านั้น แต่อย่างไรก็ดีเซลล์ต่างๆ เหล่านี้สามารถวางที่ตำแหน่งใดๆ ก็ได้บนแผ่นเวเฟอร์ซิลิกอน นั่นก็หมายความว่าชั้น Mask จะถูกจัดวางตามความต้องการของผู้ออกแบบ

4.2.2.2 Masked Gate-Array-Based ASIC

ไอซีชนิดนี้จะมีทรานซิสเตอร์หรือเกทถูกสร้างมาในลักษณะของอะเรย์สองมิติบนแผ่นเวเฟอร์ซิลิกอน และผู้ออกแบบจะทำการออกแบบ Mask เพื่อใช้สำหรับกำหนดการต่อเชื่อมของทรานซิสเตอร์แต่ละตัว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2.3. Programmable

ASIC ประเภทนี้เซลล์ลอจิกจะถูกออกแบบไว้ก่อนเช่นเดียวกับ Semi-Custom แต่ชั้นของ Mask จะไม่สามารถเปลี่ยนแปลงได้ตามความต้องการของผู้ออกแบบ ไอซีประเภทนี้ยังแบ่งออกเป็น 2 ชนิดคือ Programmable Logic Device (PLD) และ Field-Programmable Gate Array (FPGA)

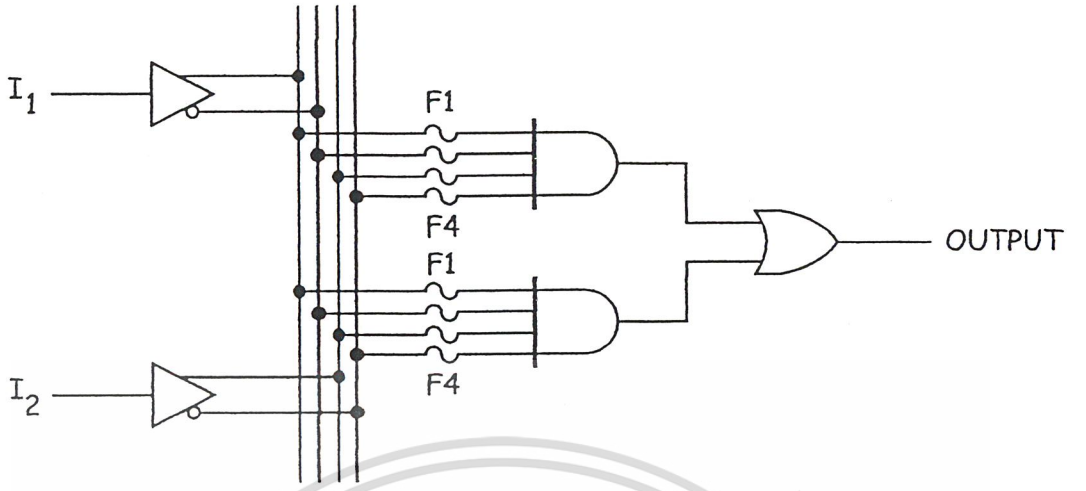
4.2.3.1 Programmable Logic Device (PLD)

มีโครงสร้างภายในเป็นวงจรพื้นฐานทางด้านลอจิกต่อกันอยู่เป็นกลุ่มซึ่งมีทั้งวงจรคอมบิเนชัน (Combination) และซีเควนเชียล (Sequential) สำหรับเทคโนโลยีของวงจรที่ใช้สร้าง PLD จะมีทั้ง TTL, ECL และ CMOS ตามความเหมาะสมของแต่ละระบบ ไอซี PLD ทุกชนิดมีหลักการพื้นฐานของวงจรภายในที่เหมือนกัน โดยมีวงจรคอมบิเนชันที่เป็นผลคูณร่วมบวก (Sum of product) ประกอบไปด้วยชุดของแอนด์เกทต่อร่วมกับออร์เกท และในการโปรแกรมจะเป็นการเลือกว่าอินพุทภายในของแอนด์เกทกับสัญญาณอินพุทใดบ้างที่จะต้องต่อถึงกัน ซึ่งมีทั้งจากภายนอกและสัญญาณป้อนกลับจากเอาท์พุทภายในเอง เช่น การติดต่ออินพุทของออร์เกทกับเอาท์พุทของแอนด์เกทตัวต่างๆ สำหรับการโปรแกรมทางกายภาพนั้นอินพุทต่างๆ ของอุปกรณ์ ทุกตัวจะถูกต่อผ่านฟิวส์เข้ากับแหล่งสัญญาณ ซึ่งถ้าไม่ต้องการใช้สัญญาณใดก็จะตัดฟิวส์ตัวนั้นทิ้ง ทำให้สามารถโปรแกรมได้เพียงครั้งเดียว ไอซี PLD บางชนิดใช้มอสทรานซิสเตอร์แทนฟิวส์ทำให้สามารถโปรแกรมโดยใช้กระแสไฟฟ้า และสามารถลบแล้วโปรแกรมเข้าไปใหม่ได้อีก สำหรับไอซีในตระกูล PLD ได้แก่ PROM, PAL, PLA และ EPLD

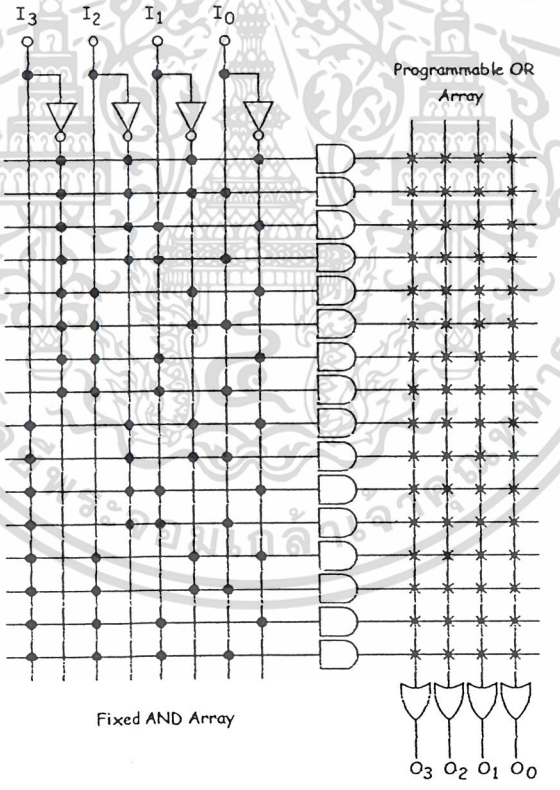
4.2.3.1.1 PROM (Programmable Read Only Memory)

PROM คือหน่วยความจำประเภท ROM ซึ่งนับว่าเป็นไอซี PLD ชนิดหนึ่งซึ่งวงจรภายในของ PROM ประกอบไปด้วยอะเรย์ของแอนด์และออร์เกท (And - Or Array) ผลลัพธ์ที่ขาดาต้าเอาท์พุทสามารถแสดงได้ในสมการของฟังก์ชันผลคูณร่วมบวก (Sum of product) ของสัญญาณอินพุทที่ขาแอดเดรส

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.2 วงจรพื้นฐานของ PLD ซึ่งอยู่ในรูปผลคูณร่วมบวก



รูปที่ 4.3 ลักษณะของ PROM เมื่อเปรียบเทียบเป็นวงจรในรูป*ผลคูณร่วมบวก*

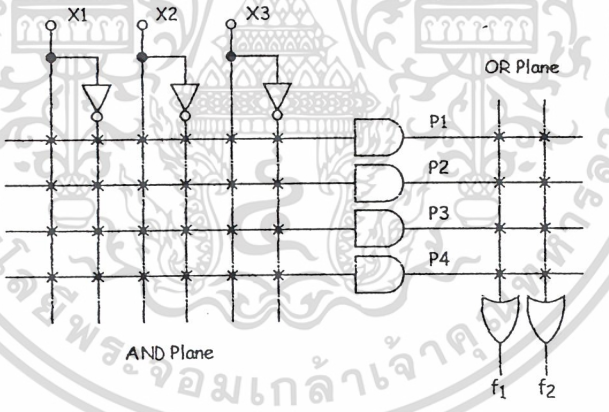
รูปที่ 4.3 แสดงถึงลักษณะการเชื่อมต่อแอนด์เกทและออร์เกทของ PROM ขนาด 16x4 บิต วงจรทางด้านซ้ายบนสุดเป็นแอนด์เกทจะให้ผลคูณ (Product) ของกรณีที่อินพุทเป็น 0000 แอนด์เกทที่อยู่ถัด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ลงมาเป็นผลคูณของกรณีที่อินพุทเป็น 0001, 0010, ... จนถึงตัวล่างสุดคือผลคูณในกรณีที่อินพุทเป็น 1111 ซึ่งสำหรับ PROM ที่มีจำนวนอินพุท n ตัวจะมีค่าอินพุทที่เป็นไปได้ทั้งหมดเท่ากับ 2^n และค่าอินพุทเหล่านี้จะถูกจัดวางอยู่ในส่วนอะเรย์ของ AND ซึ่งไม่สามารถแก้ไขได้ แต่ในส่วนของ OR จะเป็นส่วนที่อนุญาตให้ทำการโปรแกรมได้ และเนื่องจากการที่ด้าน AND ของ PROM มีการคอมบิเนชันของอินพุทที่เป็นไปได้ทั้งหมด ดังนั้นผู้ออกแบบจึงไม่จำเป็นต้องทำการลดรูปของฟังก์ชันลอจิกที่ออกแบบไว้เลย แต่อย่างไรก็ดีการกระทำเช่นนี้อาจทำให้เกิดจำนวนวงจรที่ไม่มีประสิทธิภาพจำนวนมากบนตัวชิพได้

4.2.3.1.2 PLA (Programmable Logic Array)

ลักษณะเด่นของ PLA คือสามารถโปรแกรมการเชื่อมต่อได้ทั้งทางด้าน AND และด้าน OR ทำให้มีความยืดหยุ่นในการใช้งานมาก แต่อย่างไรก็ดีข้อเสียที่เห็นได้อย่างชัดเจนของ PLA คือความยุ่งยากในการสร้าง และคุณสมบัติทางด้านความเร็วที่ลดลงเนื่องจากสัญญาณจะต้องวิ่งผ่านอะเรย์ของ AND และ OR

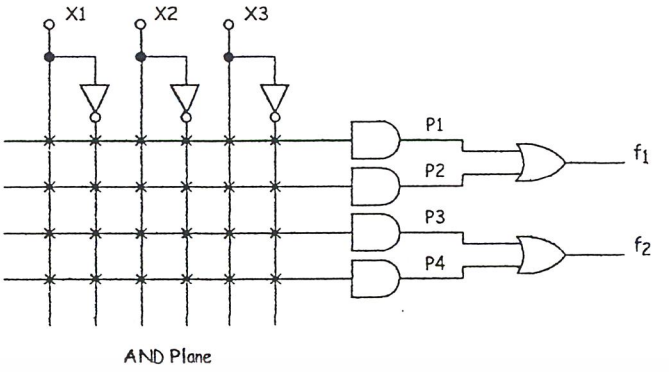


รูปที่ 4.4 วงจรพื้นฐานภายในของ PLA

4.2.3.1.3 PAL (Programmable Array Logic)

PAL มีลักษณะโครงสร้างที่ใกล้เคียงกับ PROM และ PLA มาก แต่การโปรแกรม PAL จะสามารถทำได้เพียงด้าน AND เท่านั้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.5 วงจรพื้นฐานภายในของ PAL

4.2.3.1.4 EPLD (Erasable Programmable Logic Device)

EPLD เป็นอุปกรณ์ที่สามารถทำการโปรแกรมได้หลายครั้งซึ่งเหมาะสำหรับการทำวงจรต้นแบบ สำหรับเทคโนโลยีที่ใช้ในการสร้างจะเหมือนกับ CMOS EPROM คือใช้มอสทรานซิสเตอร์เชื่อมต่อระหว่างสัญญาณอินพุทกับจุดที่ต้องการแทนการใช้ฟิวส์แบบเดิม ทำให้สามารถโปรแกรมการต่อวงจรภายในอุปกรณ์ด้วยการจ่ายไฟฟ้าตามขนาดที่กำหนดไว้ และลบได้โดยใช้แสงอัลตร้าไวโอเลทฉายผ่านช่องหน้าต่างกระจกของตัวชิพ

4.2.3.2 Field-Programmable Gate Array (FPGA)

เป็นอุปกรณ์ที่มีความซับซ้อนมากกว่า PLD ไปอีกระดับหนึ่ง ซึ่งในความเป็นจริงแล้ว PLD และ FPGA แตกต่างกันน้อยมาก สำหรับ FPGA แล้วนับว่าเป็นอุปกรณ์ตัวใหม่ในตระกูลของ ASIC ซึ่งมีการเจริญเติบโตอย่างรวดเร็วและมีบทบาทที่สำคัญในการเข้ามาแทนที่ระบบอิเล็กทรอนิกส์ที่ใช้ TTL

โครงสร้างภายในของ FPGA ประกอบไปด้วยอะเรย์ของลอจิกเกทต่างๆมากมาย ซึ่งในปัจจุบันความจุเกทภายในตัวชิพ FPGA ได้เพิ่มขึ้นจากระดับไม่กี่พันตัวจนถึงระดับล้านตัว ซึ่งสามารถรองรับวงจรดิจิตอลที่มีความสลับซับซ้อนได้เป็นอย่างดี นอกจากนี้ในด้านการออกแบบพัฒนาและทดสอบก็ทำได้ง่าย ซึ่งในปัจจุบันการออกแบบวงจรโดยใช้ FPGA กำลังเป็นที่นิยมและมีแนวโน้มจะนำมาใช้งานมากขึ้นเรื่อยๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5

ภาษา VHDL

VHDL ย่อมาจากคำว่า VHSIC Hardware Description Language (VHSIC : Very High Speed Integrated Circuit) เป็นภาษาโปรแกรมระดับสูง (High Level Language) ที่ใช้สำหรับการออกแบบฮาร์ดแวร์ในระบบดิจิตอล ตัวของภาษาสามารถบรรยายพฤติกรรมการทำงานในรูปของลำดับชั้น (Hierarchy) และสามารถเขียนได้หลายรูปแบบ ด้วยเหตุผลนี้จึงทำให้ภาษา VHDL เป็นเครื่องมือที่ใช้ออกแบบตั้งแต่ขั้นตอนบนสุด คือ แนวความคิดที่จะแก้ปัญหา ลงไปทีละขั้นจนถึงขั้นตอนของการสร้างวงจรจริง และตัวภาษาก็เปิดโอกาสให้วิศวกรได้พัฒนาและจำลองการทำงานของรูปแบบฟังก์ชันการทำงานของวงจรอย่างสังเขป โดยยังไม่ต้องคำนึงถึงรายละเอียดเกี่ยวกับโครงสร้างวงจรจริง นอกจากนั้น VHDL ยังเป็นภาษาที่สนับสนุนลักษณะต่างๆ ของระบบดิจิตอล

5.1 องค์ประกอบพื้นฐานของ VHDL

รูปแบบพื้นฐานที่ใช้ในการบรรยายถึงองค์ประกอบของ VHDL จะประกอบไปด้วยส่วนกำหนดการเชื่อมต่อ (Interface) และส่วนกำหนดลักษณะเชิงสถาปัตยกรรม (Architecture) ดังแสดงในรูปที่ 5.1 โดยในการบรรยายการเชื่อมต่อจะขึ้นต้นด้วยคำว่า ENTITY แล้วตามด้วยชื่อขององค์ประกอบจากนั้นตามด้วยคำว่า IS และถัดมาจะเป็นการบรรยายถึงพอร์ต การติดต่อ

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

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

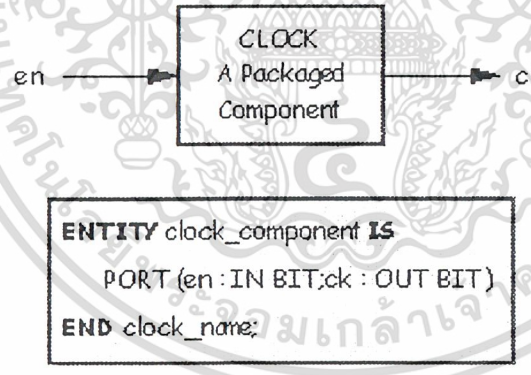
รูปที่ 5.1 การกำหนดการเชื่อมต่อและสถาปัตยกรรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

อินพุท – เอาท์พุท ขององค์ประกอบ ส่วนลักษณะภายนอกอื่น ๆ เช่น เวลา อุณหภูมิก็สามารถรวมเข้าไปในส่วนนี้ ได้เช่นกัน ในส่วนของการกำหนดลักษณะเชิงสถาปัตยกรรมจะขึ้นต้นด้วยคำว่า ARCHITECTURE ซึ่งเป็นส่วนที่ใช้ บรรยายหน้าที่การทำงานขององค์ประกอบ โดยหน้าที่การทำงานนี้จะขึ้นอยู่กับสัญญาณอินพุท เอาท์พุทและพารามิเตอร์ อื่นๆ ที่ได้กำหนดไว้ในส่วนของการเชื่อมต่อดังรูปที่ 5.1 และสำหรับการบรรยายหน้าที่ขององค์ประกอบจะเริ่มต้นหลังจาก คำว่า BEGIN เป็นต้นไป

5.1.1 การกำหนดการเชื่อมต่อ

การกำหนดการเชื่อมต่อเป็นระดับบนสุดของการออกแบบ โดยในระดับนี้ต้องกำหนดพอร์ตสำหรับการติดต่อกับองค์ประกอบ ภายนอกอื่นๆ ดังตัวอย่างในรูปที่ 5.2 ซึ่งเป็นบล็อคไดอะแกรม และการบรรยายการเชื่อมต่อขององค์ประกอบสำหรับตัวจ่าย สัญญาณนาฬิกา ในบรรทัดแรกของการบรรยายการเชื่อมต่อเป็นการกำหนดชื่อขององค์ประกอบซึ่งกำหนดเป็น clock_ component ตามด้วยคำว่า PORT และชื่อของพอร์ตอยู่ภายในวงเล็บ ส่วน IN และ OUT เป็นการกำหนด โหนดของสัญญาณให้เป็นอินพุทหรือเอาท์พุท และ BIT เป็นการแสดงชนิดของข้อมูล



รูปที่ 5.2 บล็อคไดอะแกรมและการบรรยายการเชื่อมต่อของ clock_ component

5.1.2 การกำหนดรูปแบบการบรรยาย

หน้าที่การทำงานขององค์ประกอบจะถูกบรรยายภายในส่วนนี้ ซึ่งในการบรรยายสามารถกำหนดค่าของสัญญาณ เอาท์พุทในเทอมของอินพุทหรือในรูปขององค์ประกอบอื่นๆ หรือทั้งสองอย่างรวมกันก็ได้ ดังตัวอย่างการบรรยายของ clock_component ในรูปที่ 5.3 ซึ่งเป็นการบรรยายในเชิงพฤติกรรมโดยมี en เป็นอินพุทและ ck เป็นเอาท์พุท PROCESS เป็นคำที่ใช้ในการเริ่มต้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำหรับการบรรยายในเชิงพฤติกรรม และภายในโปรเซสกำหนดให้ periodic เป็นตัวแปรที่มีค่าเริ่มต้นเป็น "0" ถ้าสัญญาณ en มีค่าเป็น "1"' จะทำให้ตัวแปร periodic ถูกคอมพลีเมนท์ (complement) และส่งค่าให้กับ ck ซึ่งเป็นสัญญาณเอาท์พุท และสำหรับคำสั่ง WAIT จะเป็นการกำหนดให้สัญญาณมีคาบเวลาเท่ากับ 1 ไมโครวินาที

```
ARCHITECTURE behavioral OF clock_component IS

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

รูปที่ 5.3 การบรรยายเชิงพฤติกรรมของ clock _ component

5.1.3 หน่วยการออกแบบแพ็กเกจ

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

5.1.3.1 PACKAGE DECLARATION

ส่วนที่มีความสำคัญที่สุดของแพ็กเกจ (ถ้ามองในแง่ของการนำไปใช้จากภายนอก) ได้แก่ ส่วนการประกาศแพ็กเกจ เนื่องจากเป็นส่วนที่ใช้กำหนดชื่อของสิ่งที่ประกาศอยู่ภายในแพ็กเกจ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำหรับนำไปใช้ภายนอกตัวของแพ็กเกจเอง ถ้ามี การประกาศสิ่งใดๆ ในส่วนของส่วนบอดี้แพ็กเกจ แต่ไม่ถูกประกาศในส่วนการประกาศแพ็กเกจจะทำให้ค่าและพฤติ กรรมไม่สามารถนำไปใช้งานในส่วนนอกได้ซึ่งเปรียบเทียบได้กับสิ่งที่ประกาศไว้ในส่วนของการประกาศ Entity คือ จุดเชื่อมต่อ หรือ พอร์ต ที่มีหน้าที่ติดต่อกับโลกภายนอก ฉะนั้นโดยทั่วไปแล้วแพ็กเกจสามารถสร้างขึ้นได้โดยไม่จำเป็น ต้องมีส่วนบอดี้ และยังสามารถนำไปใช้งานจากรูปแบบภายนอกได้เช่น ใช้สำหรับประกาศ ชนิด (Type) หรือสัญญาณ เช่นเดียวกับ ส่วนบอดี้แพ็กเกจที่ไม่จำเป็นต้องมี ส่วนของการประกาศแพ็กเกจ แต่แพ็กเกจนั้นจะไม่สามารถนำไปใช้จาก รูปแบบอื่นได้

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

รูปที่ 5.4 โครงสร้างทั่วไปของส่วนการประกาศแพ็กเกจ

5.1.3.2 PACKAGE BODY

โครงสร้างซึ่งประกอบด้วยลำดับคำสั่งที่ใช้บรรยายฟังก์ชันการทำงานของโปรแกรมย่อยทั้งหลาย ซึ่งชื่อของโปรแกรมย่อยนั้นๆ ได้ถูกประกาศไปแล้วในส่วนของการประกาศแพ็กเกจ จะถูกเก็บไว้ในส่วนของบอดี้แพ็กเกจ ทั้งนี้รวมถึง การกำหนดค่าคงที่ต่างๆ อันได้แก่ค่าคงที่ที่ถูกประกาศชื่อไว้ก่อนในส่วนของการประกาศแพ็กเกจ และถูกกำหนดค่าใน ส่วนของบอดี้แพ็กเกจ ฉะนั้นในส่วนของบอดี้แพ็กเกจจึงไม่จำเป็นต้องมี ถ้าในส่วนของการประกาศแพ็กเกจไม่มีการประกาศชื่อที่เป็นโปรแกรมย่อย หรือค่าคงที่ การเขียนบอดี้แพ็กเกจนั้นจะเป็นไปตามกฎเกณฑ์ดังแสดงในรูปที่ 5.5

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

รูปที่ 5.5 โครงสร้างของบอดี้แพ็กเกจ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.1.4 หน่วยการออกแบบ Configuration

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

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

รูปที่ 5.6 โครงสร้างโดยทั่วไปของหน่วยการออกแบบโครงแบบ

5.1.5 โปรแกรมย่อย

การใช้ฟังก์ชันและโพรซีเจอร์ใน VHDL เปรียบได้กับการใช้โปรแกรมย่อยในการเขียนโปรแกรมภาษาชั้นสูงทั่วๆ ไปค่าที่ถูกส่งกลับหรือถูกเปลี่ยนแปลงโดยโปรแกรมย่อยอาจจะมีหรือไม่มีผลต่อฮาร์ดแวร์โดยตรงก็ได้ เช่นถ้าใช้ฟังก์ชัน แทนการกระทำในสมการบูลีนก็จะมีผลต่อวงจรลอจิกจริงๆในขณะที่ถ้าใช้โปรแกรมย่อยในการเปลี่ยนชนิดของข้อมูล หรือในการคำนวณค่าการหน่วงเวลาแล้วก็จะไม่มีผลต่อโครงสร้างของฮาร์ดแวร์ รูปที่ 5.7 แสดงการใช้โพรซีเจอร์ เพื่อเปลี่ยนข้อมูลชนิด 8 บิตเป็นค่าจำนวนเต็ม และรูปที่ 5.8 แสดงการใช้ฟังก์ชันโดยกำหนดให้ X เป็นตัวแปรชนิด บิตแทนการกระทำในสมการบูลีน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

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

รูปที่ 5.7 การใช้โพรซีเจอร์

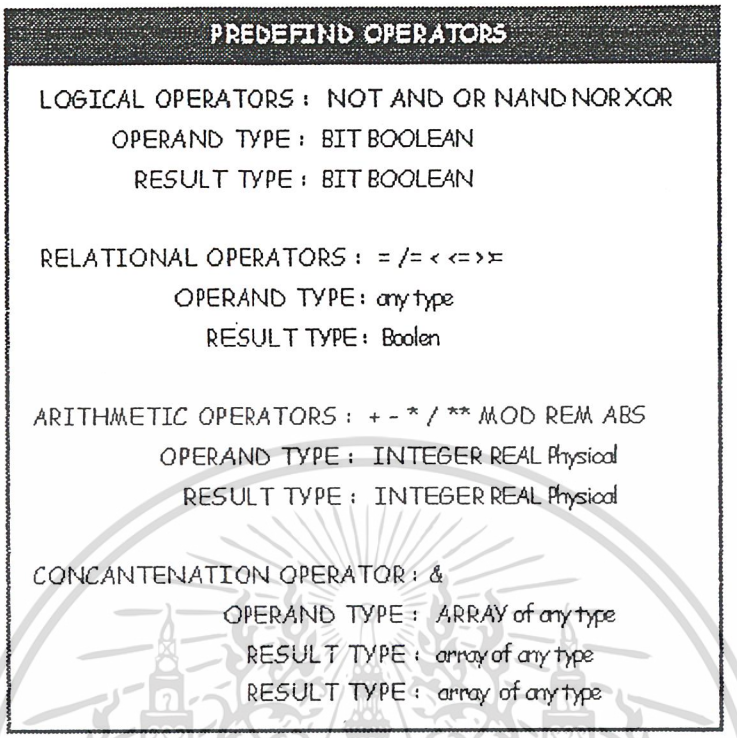
```
FUNCTION f (a, b, c: BIT) RETURN BIT  IS
        VARIABLE x: BIT;
BEGIN
        x := ((NOT a) AND (NOT b) AND c);
        RETURN x;
END f;
```

รูปที่ 5.8 การใช้ฟังก์ชัน

5.1.6 โอเปอร์เรเตอร์

การบรรยายเชิงพฤติกรรมในภาษา VHDL มีตัวดำเนินการหรือโอเปอร์เรเตอร์ทางลอจิกและคณิตศาสตร์เช่นเดียว กับภาษาซอฟต์แวร์ทั่วไปดังรูปที่ 5.9

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.9 ตัวดำเนินการใน VHDL

5.1.7 เวลาและความพร้อมเพรียง

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

5.1.8 สัญญาณและตัวแปร

สัญญาณมีลักษณะเป็นเสมือนตัวกลางฮาร์ดแวร์ที่ใช้ในการส่งผ่านข้อมูลและมีเรื่องของเวลาเข้ามาเกี่ยวข้องด้วยการ กำหนดค่าให้กับสัญญาณจะใช้สัญลักษณ์ <= ในการส่งค่าและสามารถใช้คำสั่ง AFTER เพื่อกำหนดช่วงเวลาในการ ส่งผ่านค่าของสัญญาณ เช่น w <= a AFTER 12 NS หมายถึงการกำหนดค่าสัญญาณ a ให้กับ w หลังจากเวลา ผ่านไป 12 นาโนวินาที ในทางตรงข้ามตัวแปรมีลักษณะเป็นเสมือนตัวกลางที่ใช้ในการส่งผ่านข้อมูลและไม่มีเรื่องของ เวลาเข้ามาเกี่ยวข้อง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

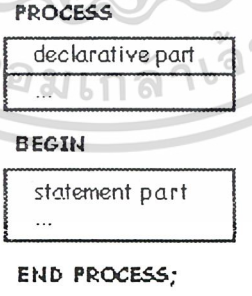
ด้วย ซึ่งตัวแปรจะถูกใช้ในส่วนที่มีการทำงานเป็นแบบลำดับคำสั่งเช่นใน ฟังก์ชัน โพรซีเจอร์ และโปรเซส สำหรับการกำหนดค่าให้กับตัวแปรจะใช้สัญลักษณ์ :=

5.2 การบรรยายเชิงพฤติกรรม

การบรรยายลักษณะการทำงานของอุปกรณ์ฮาร์ดแวร์ในเชิงพฤติกรรม เป็นการบรรยายลักษณะการเปลี่ยนแปลงของ ข้อมูลในรูปแบบของอัลกอริธึมสำหรับการคำนวณผลลัพธ์ที่เกิดขึ้น ซึ่งสืบเนื่องมาจากการเปลี่ยนแปลงสภาวะของข้อมูล ที่เข้ามาโดยไม่คำนึงถึงลักษณะโครงสร้างหรือความสัมพันธ์ของอุปกรณ์ที่อยู่ภายในว่าจะเป็นอย่างไร ในหัวข้อนี้จะ แสดงถึงการบรรยายเชิงพฤติกรรม แทนการใช้โมดูลฮาร์ดแวร์รวมถึงข้อกำหนดต่างๆ ที่ควรรู้

โปรเซส

โปรเซสเป็นรูปแบบพื้นฐานอย่างหนึ่งที่ใช้ในการกำหนดให้กับสัญญาณ โปรเซสจะอยู่ในสถานะที่เตรียมพร้อมอยู่เสมอ และจะปฏิบัติคำสั่งพร้อมๆ กันกับโปรเซสอื่นๆ ที่อยู่ในสถาปัตตกรรมบรรยายเดียวกัน โดยโปรเซสจะปฏิบัติงานตามคำ สั่งทันทีที่มีเหตุการณ์เกิดขึ้นกับญญาณที่อยู่ทางด้านขวามือของสัญลักษณ์กำหนดค่าให้กับสัญญาณ (<=) การบรรยาย โปรเซสจะเริ่มต้นด้วยคำสั่ง PROCESS และจบด้วยคำสั่ง END PROCESS ในรูปที่ 5.10 เป็นการแสดงส่วน ประกอบของการบรรยายแบบโปรเซส ซึ่งประกอบด้วยส่วนของการประกาศตัวแปรที่ต้องใช้และส่วนของการปฏิบัติ คำสั่งเพื่อให้ได้ผลลัพธ์ที่ต้องการ



รูปที่ 5.10 รูปแบบของการบรรยายแบบโปรเซส

การกำหนดตัวดำเนินการภายในโปรเซส

ตัวดำเนินการภายในโปรเซสมี 3 ชนิดคือ ตัวแปร (Variable) ไฟล์ (File) และตัวคงที่ (Constant) ซึ่งตัวดำเนิน การทั้งสามชนิดนี้หากมีการประกาศไว้ในโปรเซสใดก็จะใช้ได้เฉพาะภาย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในโปรเซสนั้นเท่านั้นสำหรับการติดต่อกับภาย นอกหรือระหว่างโปรเซสสามารถทำได้โดยใช้สัญญาณ (Signal) หรือตัวคงที่ที่ได้ประกาศไว้ในส่วนของ ARCHITECTURE ในรูปที่ 5.11 แสดงตัวอย่างการประกาศตัวกระทำภายในโปรเซส ซึ่งจะอยู่ระหว่างคำสั่ง PROCESS และ BEGIN และค่าเริ่มต้นที่ถูกกำหนดให้กับตัวดำเนินการภายในโปรเซสจะถูกนำมาใช้ในตอนเริ่มต้น ของการปฏิบัติเพียงครั้งเดียวเท่านั้น ต่างกับค่าเริ่มต้นที่อยู่ภายในโปรแกรมย่อยจะถูกนำมาใช้ทุกครั้งที่มีการเรียกใช้ โปรแกรมย่อยนั้น ๆ

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

รูปที่ 5.11 ตัวอย่างการประกาศตัวดำเนินการภายในโปรเซส

การกำหนดการกระทำภายในโปรเซส

การกระทำใดๆ ภายในโปรเซสจะเป็นการปฏิบัติแบบลำดับ (Sequential) เสมอ ซึ่งภายในโปรเซสสามารถใช้ประโยค เงื่อนไขหรือการทำซ้ำได้เช่น IF-THEN - ELSE,CASE - WHEN, FOR LOOP และ WHILE LOOP ดังตัวอย่างในรูปที่ 5.12 และ 5.13

การกระตุ้นและยับยั้งการกระทำของโปรเซส

การกระทำภายในโปรเซสจะอยู่ในสภาพเตรียมพร้อม และมีการปฏิบัติงานอยู่ตลอดเวลาที่มีการเปลี่ยนแปลงของเหตุการณ์ เกิดขึ้น อย่างไรก็ดีเราสามารถกระตุ้นหรือยับยั้งการกระทำภายในโปรเซสได้โดยการกำหนดรายการของสัญญาณที่ต้อง การให้โปรเซสปฏิบัติงานเมื่อมีเหตุการณ์เกิดขึ้นกับสัญญาณที่กำหนดไว้เท่านั้น ส่วนเหตุการณ์ใดๆ ที่เกิดขึ้นกับสัญญาณ ที่ไม่ได้กำหนดไว้ในรายการก็จะ ไม่ส่งผลให้มีการกระทำภายในโปรเซส ซึ่งรายการของสัญญาณนี้เรียกว่า Sensitivity List

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

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

รูปที่ 5.12 เงื่อนไขการกระทำในโปรเซส

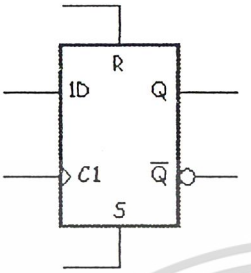
```
ARCHITECTURE demo OF partial_process IS
  ...
BEGIN
   PROCESS
    BEGIN
          ...
          x <= a AFTER 10 NS;
          y <= b AFTER 6 NS;
          ...
   END PROCESS;
END demo;
```

รูปที่ 5.13 แสดงการกระทำในโปรเซส

และจะกำหนดไว้ภายในวงเล็บหลังคำสั่ง PROCESS รูปที่ 5.14 (a) แสดงตัวอย่างโมเดล และรูปที่ 5.14 (b) เป็นตัวอย่างการบรรยายการเชื่อมต่อของ D-Flip Flop ส่วนรูปที่ 5.15 แสดงถึงการบรรยายเชิงพฤติกรรมของ D-Flip Flop โดยในรูปที่ 5.15 (a) เป็นการใช้ตัวกระทำภายนอกโปรเซส และรูป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ที่ 5.15 (b) เป็นการใช้ตัวกระทำภายในโปรเซส โดยมีรายการของสัญญาณ (rst, set, clk) เป็นตัวกระตุ้นการปฏิบัติงานภายในโปรเซส



(a)

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

(b)

รูปที่ 5.14 (a) ตัวอย่างโมเดล D-Flip Flop
(b) การบรรยายการเชื่อมต่อของ D-Flip Flop

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

(a)

รูปที่ 5.15 การบรรยายเชิงพฤติกรรมของ D-FlipFlop
(a) การใช้ตัวกระทำภายนอกโปรเซส

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

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

(b)

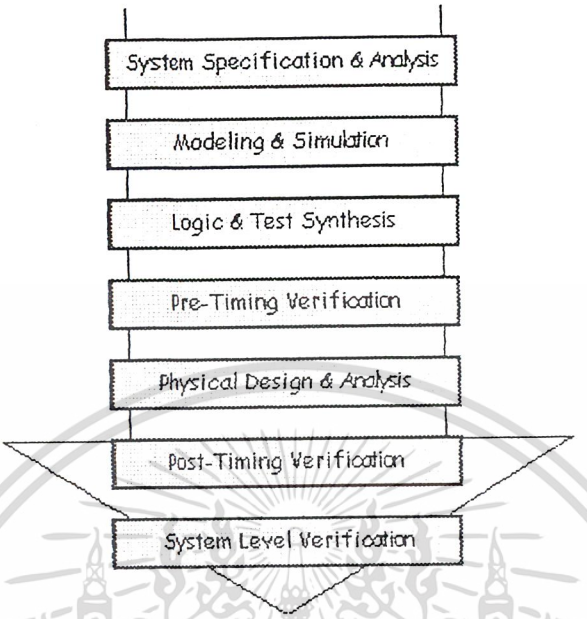
รูปที่ 5.15 การบรรยายเชิงพฤติกรรมของ D-FlipFlop

(b) การใช้ตัวกระทำภายในโปรเซส

การออกแบบจากบนลงล่าง

ในการพัฒนาวงจรรวมดิจิตอลขนาดใหญ่ที่มีความซับซ้อน วิศวกรหรือผู้ออกแบบมักจะมองการออกแบบให้อยู่ในรูปของ บล็อกไดอะแกรมก่อนที่ทำวิเคราะห์ให้ลึกถึงรายละเอียดต่อไป ซึ่งภาษา VHDL นั้นอนุญาตให้อธิบายและวิเคราะห์การ ทำงานของแต่ละบล็อก รวมถึงการปรับปรุงการทำงานจากผลที่วิเคราะห์เพื่อให้ได้การทำงานตามต้องการ นอกจากนี้ยัง สามารถเพิ่มเติมในรายละเอียดในแต่ละขั้นตอนได้ ซึ่งหลักการนี้สอดคล้องกับหลักการออกแบบจากบนลงล่าง (Top - Down Design) นั่นเอง ถ้าทดลองเปรียบเทียบกับการออกแบบจากล่างขึ้นบน (Bottom - Up Design) จะเห็นได้ว่า การออกแบบจากล่างขึ้นบนจะใช้เวลาการออกแบบมากกว่า 90% เนื่องจากเป็นการวาดวงจรด้วยอุปกรณ์ต่างๆ (Schematic capture) ที่ประกอบกันเข้าเป็นวงจรที่ต้องการออกแบบ ก่อนแล้วจึงทำการจำลองการทำงาน และตรวจ สอบความถูกต้อง ดังนั้นการใช้ภาษา VHDL กับหลักการออกแบบจากบนลงล่างจึงเป็นทางออกให้กับวิศวกรให้สามารถ ออกแบบและพัฒนาวงจรที่มีความซับซ้อนได้มากขึ้น ทั้งยังช่วยลดเวลาและค่าใช้จ่ายในการออกแบบด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.16 ขั้นตอนการออกแบบจากบนลงล่าง

จากรูปที่ 5.16 แสดงถึงขั้นตอนของการออกแบบจากบนลงล่าง ทั้งนี้ในทางปฏิบัติอาจมีข้อแตกต่างไปจากนี้บ้าง เล็กน้อยเนื่องจากขั้นตอนของการผลิต (Implementation) สามารถกระทำได้หลายเทคโนโลยี สำหรับรายละเอียดของขั้นตอน การออกแบบจากบนลงล่างในแต่ละขั้นตอนมีดังนี้

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

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

3.หลังจากที่ได้หลักการขั้นต้นพร้อมแนวความคิดที่ผ่านการตรวจสอบแล้วหลักการนี้จะถูกเพิ่มเติมในรายละเอียดลงมา เป็นลำดับขั้นที่สอง จนกระทั้งอยู่ในระดับที่จะนำไปผลิตวงจริง หรือสังเคราะห์ในขั้นตอนนี้เองเทคโนโลยีที่จะมารองรับ วงจรออกแบบจะถูกกำหนดขึ้น และระบบช่วยการออกแบบจะสังเคราะห์วงจรที่ได้จากรูปแบบที่เขียนขึ้นให้อยู่ในรูปของ วงจรที่ประกอบด้วยอุปกรณ์อิเล็กทรอนิกส์ หรือวงจรในระดับเกท และการเชื่อมต่อระหว่างกันของอุปกรณ์เหล่านั้น หรือ ไม่ก็อยู่ในรูปของ Netlist ที่สามารถนำไปผลิตในอุปกรณ์อื่นได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

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

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

6.ทำการตรวจสอบการทำงานและตัวแปรทางด้านเวลาทั้งหมด เพื่อความถูกต้องของวงจรเป็นครั้งสุดท้ายก่อนนำไปรวมเข้ากับอุปกรณ์อื่นๆ ให้เป็นระบบดิจิตอล เนื่องจากในขั้นตอนนี้ วงจรที่ออกแบบ จะประกอบด้วยจุดต่อทางอินพุทและเอาท์พุท ซึ่งเป็นจุดต่อสำหรับการรับและส่งสัญญาณกับภายนอก

7.นำวงจรที่ออกแบบไว้ประกอบเข้ากับอุปกรณ์อื่นๆ ให้เป็นระบบที่สมบูรณ์ แล้วทำการทดสอบการทำงานทั้งระบบร่วมกับอุปกรณ์อื่นๆ อีกครั้งเพื่อควบคุมคุณภาพของผลิตภัณฑ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 6
การออกแบบ และการสร้าง

ในการออกแบบและการสร้าง ECG MONITOR นั้น เราได้ทำการออกแบบ โดยแบ่งการออกแบบเป็นสองภาคด้วยกันคือ

6.1 ส่วนของภาคการแสดงผลคลื่นสัญญาณหัวใจ

6.2 ส่วนของภาคการแสดงผลตัวอักษรและกราฟฟิก

ซึ่งในส่วนการทำงานของทั้งสองภาคนั้นได้ทำงานแยกอิสระต่อกันแต่จะมีบางส่วนเท่านั้นที่ทั้งสองภาคจะต้องทำงานร่วมกัน โดยการทำงานทั้งสองภาคนั้นสามารถอธิบายได้ดังนี้

6.1 ส่วนของภาคการแสดงผลคลื่นสัญญาณหัวใจ

วงจรรูปที่ 6.1 จะเป็นวงจรที่นำเอาข้อมูลจาก A/D ที่ได้จากการ SAMPLING สัญญาณคลื่นหัวใจที่ถูกขยายความแรงของสัญญาณแล้ว โดยข้อมูลที่ได้จะมีความละเอียด 0-256 (8 บิต) ระดับ ซึ่งหมายความว่า สัญญาณคลื่นหัวใจที่เข้ามาทุกๆ 1/320 วินาที (ความถี่ในการ SAMPLING) จะถูกแปลง เป็นสัญญาณดิจิตอล สัญญาณที่ได้ทั้งหมดจะถูกเก็บไปที่ SRAM ขนาด 512Byte โดยใช้เทคนิคในการอ้างอิงแอดเดรส เช่น ทุกครั้งที่มีข้อมูลใหม่เข้ามาแอดเดรสของการเขียนจะเพิ่มขึ้น ซึ่งในกาออกแบบกำหนไว้ว่าถ้ามีการเขียนข้อมูลใน SRAM แอดเดรสของการอ่านจะมีการเปลี่ยนแปลงด้วยเป็นการออกแบบให้ภาพที่ปรากฏมีการเคลื่อนที่ ลักษณะที่เลื่อนภาพจากทางด้านขอบจอทางขวามือไปทางซ้ายมือ แล้วนำข้อมูลที่ได้จากการอ่าน SRAM ไปสร้างสัญญาณ RGB ส่งออกไปพร้อมกับ สัญญาณ H_SYNC และ V_SYNC ของ MONITOR

จากรูปจะอธิบายแบบคร่าว ๆ ดังนี้

6.1.1. บล็อก BUFFERDATA

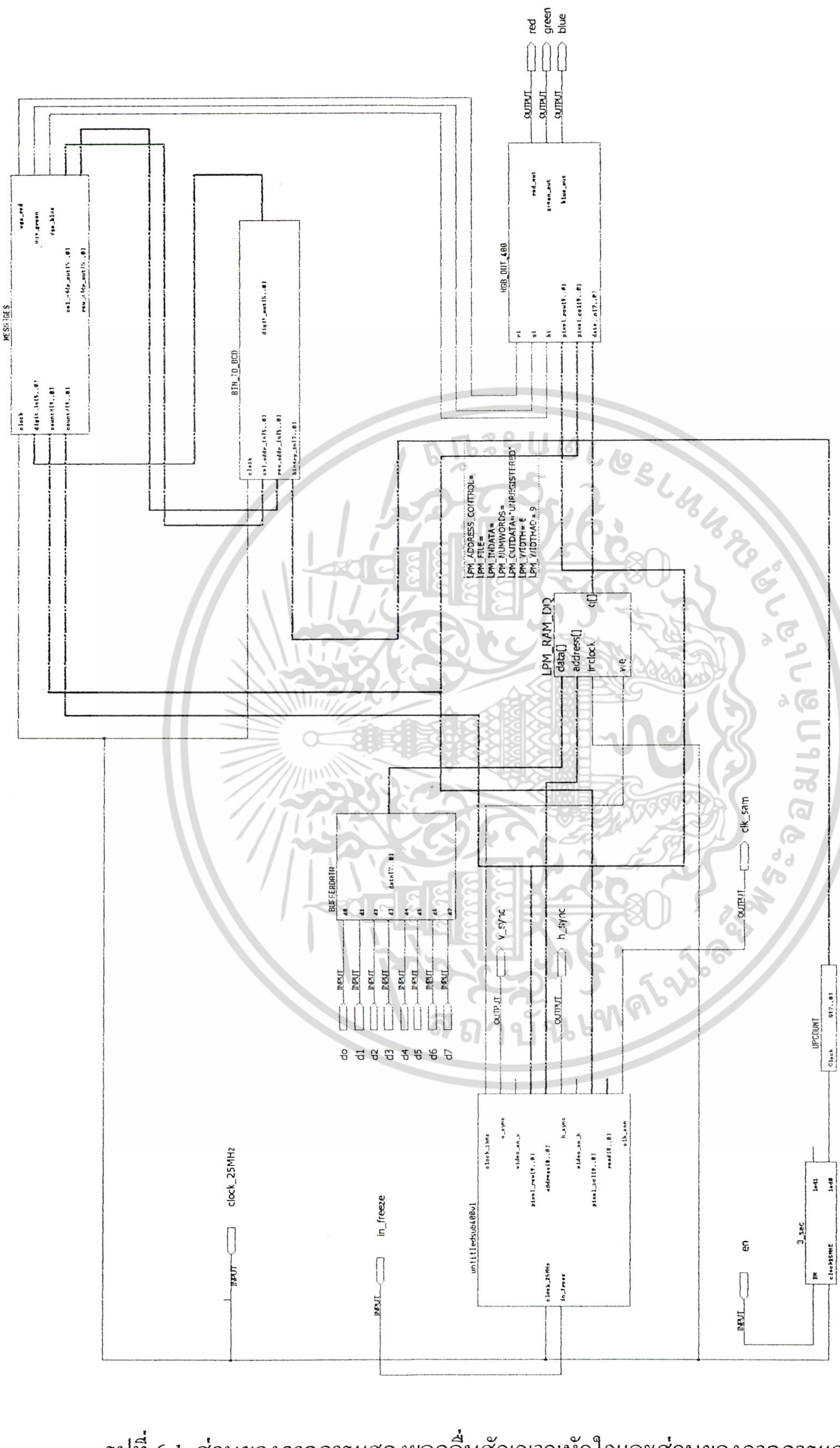
6.1.2. บล็อก LPM_RAM_DQ

6.1.3. บล็อก RGB_OUT_EDIT

6.1.4. บล็อก HEART_RATE

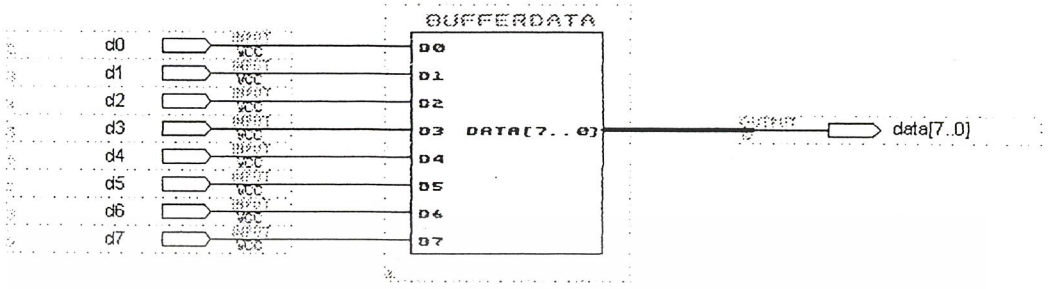
6.1.5. บล็อก UNTITLEDSUB320

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 6.1 ส่วนของภาคการแสดงผลคลื่นสัญญาณหัวใจและส่วนของภาคการแสดงผลตัวอักษร

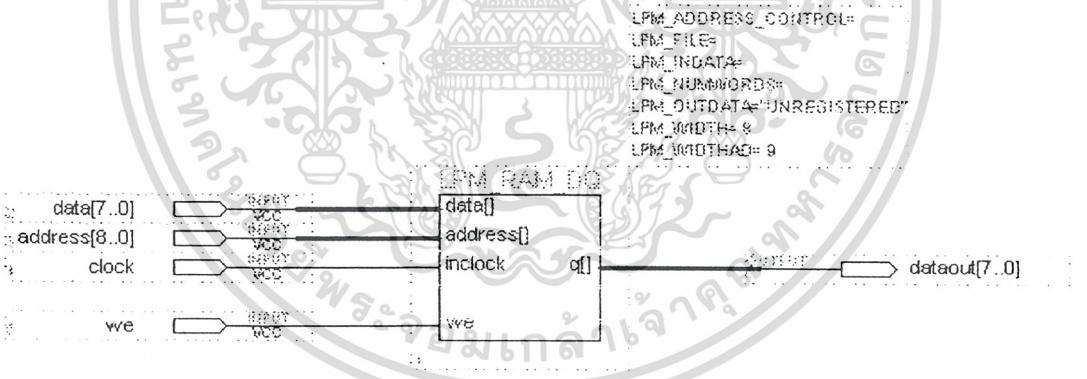
6.1.1. บล็อก BUFFERDATA



รูปที่ 6.2 บล็อก BUFFERDATA

เป็นวงจรรับค่าข้อมูลที่ได้จาก A/D ข้อมูลที่ได้จะผ่านเข้าสู่ SRAM โดยตรงเป็นการออกแบบให้ข้อมูลมีรูปแบบเหมือนกับข้อมูลทาง INPUTของ SRAM

6.1.2. บล็อก LPM_RAM_DQ

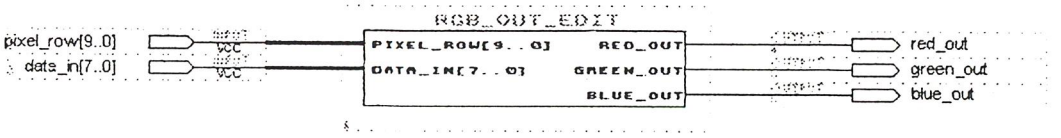


รูปที่ 6.3 บล็อก LPM_RAM_DQ

เป็น SRAM ที่สร้างจากคำสั่ง ARRAY ถูกออกแบบโดย บริษัท ATERAได้ออกแบบให้มีการใช้งานที่คล่องตัวและมีความยืดหยุ่นในการใช้สูง ซึ่งผู้ออกแบบสามารถกำหนดขนาดของแอดเดรสความกว้างของข้อมูลในการออกแบบนี้ผู้ออกแบบได้กำหนด SRAM ให้มีขนาด 512Byte จากรูปบล็อกข้างล่าง ขา WE เป็นขาที่ใช้ในการกำหนดการเขียนและการอ่านข้อมูลในการเลือกระหว่างการอ่านและการเขียนจะขึ้นอยู่กับสัญญาณ CLOCK_INPUT (ขาWE) ว่ามีสัญญาณ Clock ลอจิกใดจะเป็นการอ่านหรือการเขียนหน่วยความจำ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

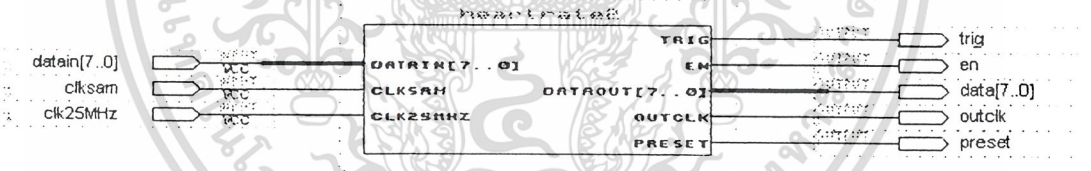
6.1.3. บล็อก RGB_OUT_EDIT



รูปที่ 6.4 บล็อก RGB_OUT_EDIT

การออกแบบได้กำหนดให้ INPUT เป็น PIXEL_ROW ขนาด 10 บิต ค่าที่ได้นี้เป็นค่า ROW ที่ใช้ในการสแกนภาพจะนับตั้งแต่ 0-480 และอีกขาหนึ่งเป็นขา DATA_IN ขานี้จะรับข้อมูลจากการอ่าน SRAM การออกแบบบล็อกนี้กำหนดไว้ว่าถ้าต้องการให้แถว(ROW) ไหนแสดง ผลของข้อมูลก็กำหนดให้ ค่าของ PIXEL_ROW นั้นมีค่าเท่ากับ DATA แล้วขับสัญญาณที่ใช้สร้างสีผ่าน RGB_OUT

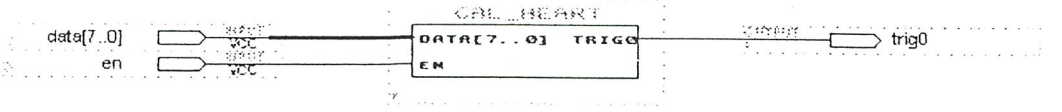
6.1.4. บล็อก HEART_RATE2



รูปที่ 6.5 บล็อก HEART_RATE2

จุดประสงค์ในการสร้างบล็อกนี้คือเพื่อหาค่า อัตราการเต้นของหัวใจต่อนาที(HEART RATE) ซึ่งในบล็อกนี้ประกอบด้วยบล็อกต่าง ๆ ดังนี้

6.1.4.1 บล็อก CAL_HEART



รูปที่ 6.6 บล็อก CAL_HEART

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการ[illegible]อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

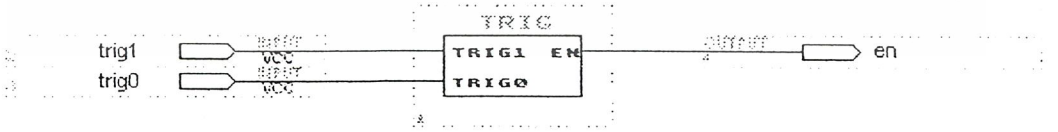
บล็อกนี้จะทำหน้าที่เหมือนกับวงจร COMPARATOR ของ OP-AMP ต่างกับเพียง INPUT เป็นดิจิตอล โดยระดับดิจิตอลจะเปรียบค่า 2 ค่าคือค่า THRESHOLD ที่ถูกกำหนดจากการตั้งค่าที่กำหนดกับค่า DATA ถ้าระดับ INPUT DATA มีค่ามากกว่า TRESHOLD บล็อกนี้ก็จะส่งค่า TRIG0 ออกไป แสดงว่ามีการตรวจจับเกิดขึ้นแล้ว

6.1.4.2 บล็อก SET

รูปที่ 6.7 บล็อก SET

บล็อกนี้มีความสำคัญอย่างยิ่ง ซึ่งในส่วนการของบล็อกนี้ จะมี INPUT เป็น INCLOCK กับ SETCLK ที่ INCLOCK จะรับการตรวจจับสัญญาณจาก BLOCK CAL_HEART มานับและเก็บค่าไว้ SETCLK มีไว้เพื่อ RESET COUNTS มีอยู่ให้นับใหม่เป็น 0 ขา SETCLK จะมีสัญญาณก็ต่อเมื่อมีการนับครบ 3 วินาที วิธีการนับจะอธิบายในบล็อกต่อไป บล็อกนี้จะมี OUTPUT เป็นสัญญาณ EN สัญญาณป้อนกลับสัญญาณให้กับบล็อกก่อนหน้านี้ เป็นการกำหนดให้เริ่มมีการนับ 3วินาทีใหม่ ในการเข้ามาของสัญญาณ INCLOCK ในแต่ละครั้งจะมีการนับ และคำนวณออกเป็นค่า HEARTRATE

6.1.4.3 บล็อก TRIG

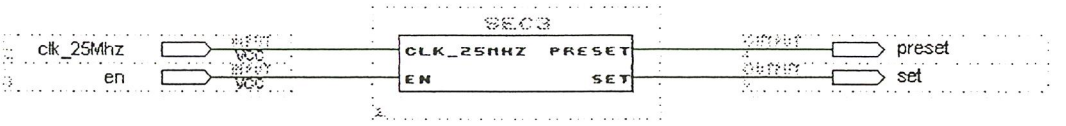


รูปที่ 6.8 บล็อก TRIG

บล็อกนี้เป็นบล็อกที่เพิ่มเข้ามา มีหน้าที่เป็นตัวเพิ่มความสามารถ โดยความจำเป็นของบล็อกนี้คือลดความผิดพลาดของระบบดิจิตอลของบล็อก SEC3 โดยใช้สัญญาณ TRIG0 กับสัญญาณ EN มาช่วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

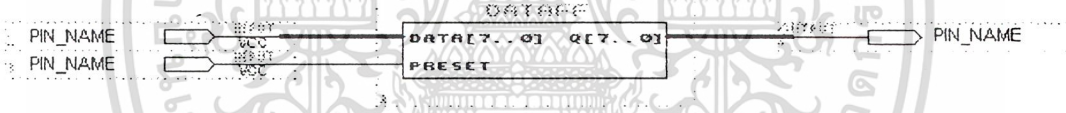
6.1.4.4 บล็อก SEC3



รูปที่ 6.9 บล็อก SEC3

บล็อกนี้จะทำการนับคาบเวลา 3 วินาที โดยใช้ CLK ของระบบ การนับจะถูกตั้งค่าจากสัญญาณ EN และสัญญาณ OUTPUT คือสัญญาณ SET แจ้งว่าการนับครบ 3 วินาทีแล้ว และอีกขาคือขา PRESET ขานี้จะส่งสัญญาณเช่นเดียวกับ SET แต่จะทำการส่งสัญญาณล่วงหน้าออกไปก่อน

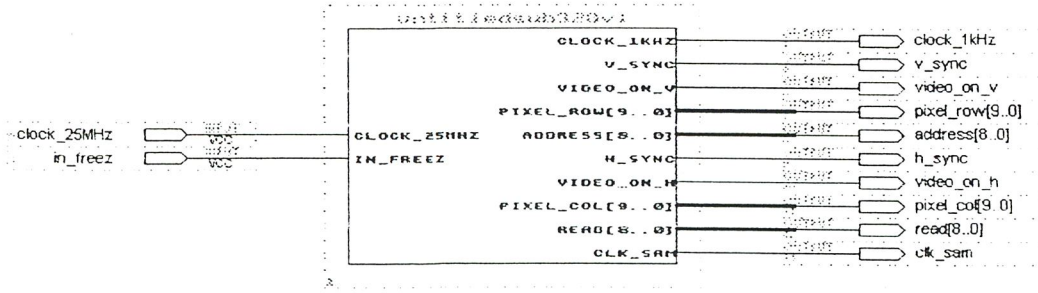
6.1.4.5 บล็อก DATAFF



รูปที่ 6.10 บล็อก DATAFF

บล็อกนี้ออกแบบมาเพื่อ LATCH ค่าที่ได้จากการคำนวณ HEARTRATE

6.1.5. บล็อก UNTITLEDSUB320

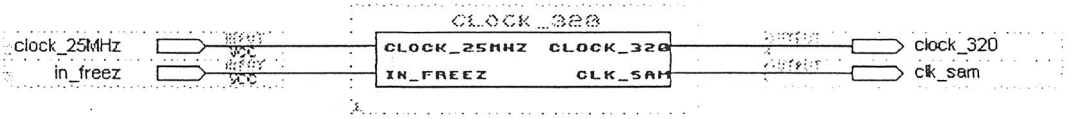


เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 6.11 บล็อก UNTITLEDSUB320

บล็อกนี้จะประกอบด้วยส่วนย่อยต่าง ๆ อีกอธิบายได้ดังนี้

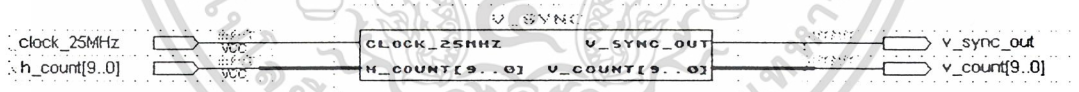
6.1.5.1 บล็อก CLOK_320



รูปที่ 6.12 บล็อก CLOK_320

บล็อกนี้จะสร้างความถี่ในการ SAMPLING ให้กับ INPUT ที่ ADC กับ ขา WE ของ SRAM การทำงานจะนำ CLK ของระบบมาหารเหลือ320 Hz แต่ OUTPUT ที่ออกมาออกเป็น 2 สัญญาณออกมา 2 สัญญาณนั้นมีความหมายว่าอีกสัญญาณหนึ่งจะมีความหมาย ว่าการออกแบบให้ ADC ทำงานแบบมีการ DELAY ทำการเปลี่ยนค่าไปก่อนแล้วค่อยส่ง CLOCK เก็บค่า DATA ที่ออกรอไว้ เพื่อการทำงานที่เข้าจังหวะกัน และที่ INPUT จะมีขา IN FREEZE ขานี้ได้ออกแบบให้ผู้ใช้หยุดการ SAMPLING ของข้อมูล

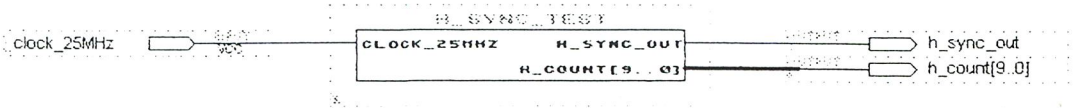
6.1.5.2 บล็อก V_SYNC



รูปที่ 6.13 บล็อก V_SYNC

เป็นบล็อกที่ใช้เป็นตัวกำเนิดสัญญาณ V_SYNC โดยที่ INPUT จะเป็น CLOCK_25 MHz

6.1.5.3 บล็อก H_SYNC

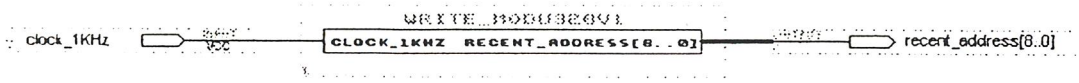


รูปที่ 6.14 บล็อก H_SYNC

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เป็นบล็อกที่ใช้เป็นตัวกำเนิดสัญญาณ H_SYNC โดยที่ INPUT จะเป็น CLOCK_25 MHz

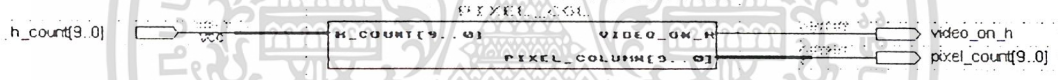
6.1.5.4 บล็อก WRITE_MODU320V1



รูป 6.15 บล็อก WRITE_MODU320V1

เป็นบล็อกที่ใช้ในการกำหนดค่า ADRESS ที่ใช้ในการเขียน(WRITE) เข้าสู่ SRAMโดยตำแหน่งแอดเดรสที่จะใช้ INPUT เป็น CLOCK_320Hz เป็นการรับค่า CLOCK มานับแล้วค่าที่นับได้จะเป็นแอดเดรสของการเขียน

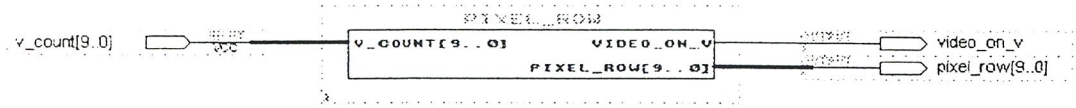
6.1.5.5 บล็อก PIXEL_COL



รูปที่ 6.16 บล็อก PIXEL_COL

โดยบล็อกนี้จะสร้างสัญญาณออกมา 2 สัญญาณคือ VIDEO_ON_H และ PIXEL_COLซึ่งทั้ง 2 สัญญาณนี้จะใช้ในการสร้างสัญญาณภาพ

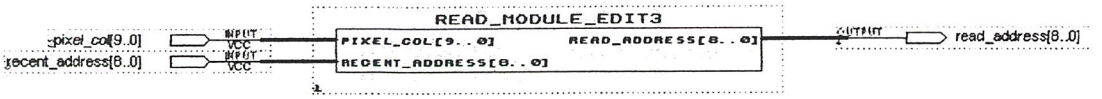
6.1.5.6 บล็อก PIXEL_ROW



รูปที่ 6.17 บล็อก PIXEL_ROW

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

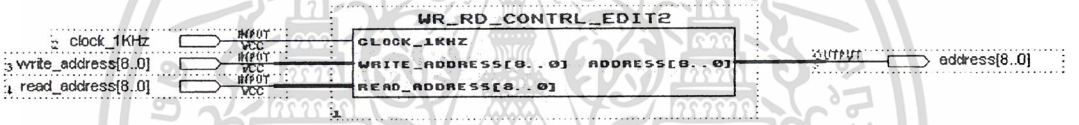
6.1.5.7 บล็อก READ_MOD_EDIT3



รูปที่ 6.18 บล็อก READ_MOD_EDIT3

บล็อกนี้เป็นบล็อกสำคัญที่ใช้ในการเลื่อนตำแหน่งแอดเดรสในการอ่านข้อมูล โดยที่ INPUT ประกอบไปด้วย PIXEL_COL กับ RECENT_ADDRESS (เป็นค่าที่บอกว่าเป็นแอดเดรสล่าสุดที่ใช้ในการเขียน)ทั้ง 2 แอดเดรสนี้จะเป็นตัวหลักในการเลื่อนภาพ

6.1.5.8 บล็อก WR_RD_CONTROL



รูปที่ 6.19 บล็อก WR_RD_CONTROL

จะใช้ในการควบคุมการเขียนและการอ่านให้เป็นไปตามสัญญาณ CLOCK ถ้ามีสัญญาณ CLOCK เข้ามาจะเป็นการเขียน ถ้าไม่มีสัญญาณ CLK เข้ามาแอดเดรสที่ปล่อยออกไปเป็นแอดเดรสที่ใช้ในการอ่าน

6.2 ส่วนของภาคการแสดงผลตัวอักษรและกราฟฟิก

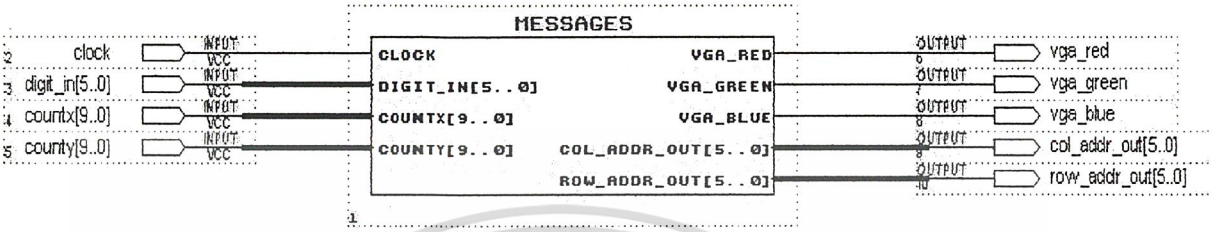
จากรูปที่ 6.1 จะแสดง บล็อก GRAPHIC DISPLAY ในส่วนบน ซึ่งในบล็อกนี้จะประกอบด้วยบล็อกต่าง ๆ ดังนี้

6.2.1 BLOCK MESSAGES

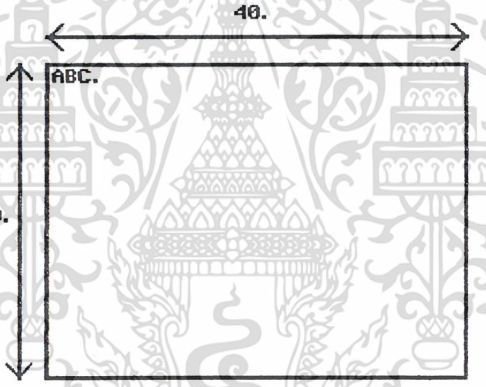
เป็น BLOCK DIAGRAM ที่ทำหน้าที่ในส่วนของการแสดงผลตัวอักษร,ข้อความและกราฟฟิกทั้งหมด โดยในส่วนของการทำงานนั้น BLOCK MESSAGES จะทำการรับสัญญาณ COUNTX

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

และ COUNTY มาทำการประมวลผล เพื่อกำหนดตำแหน่งแถวและหลักที่จำเป็นในการแสดงผล จำนวนอักษรในการแสดงผลสูงสุดตามแนวนอนได้ 40 ตัวอักษร และแนวตั้งได้ 30ตัวอักษร ซึ่งได้แสดงดังรูปที่ 1 และรูปที่ 2



รูปที่ 6.20 แสดง BLOCK MESSAGES



รูปที่ 6.21 แสดงจำนวนอักษรสูงสุดในการแสดงผล

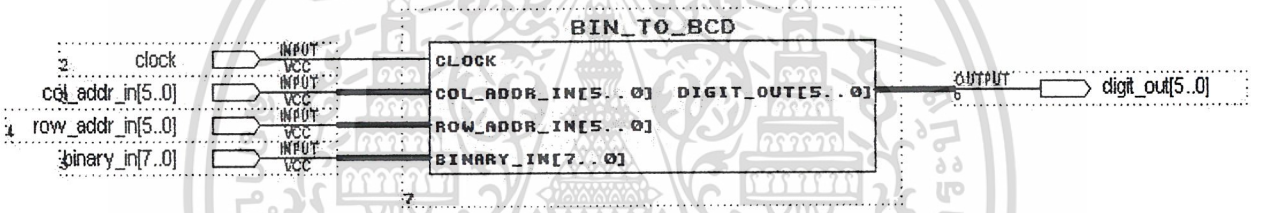
และ BLOCK นี้จะประกอบไปด้วยจะประกอบไปด้วย ROM ทั้งหมด 2 ชุด โดย ROM ชุดแรกนั้นจะทำหน้าที่เป็นตัวกำหนดข้อความและกราฟฟิกที่จะต้องการแสดงผลบนหน้าจอภาพว่าจะต้องการให้แสดงข้อความและกราฟฟิกออกมาอย่างไร และ ROM ชุดที่สอง นั้น จะทำหน้าที่เป็นตัวกำหนดลักษณะตัวอักษรที่จะต้องการแสดงผลในแต่ละตัว ว่าจะต้องการให้แสดงผลออกมาอย่างไร โดยตัวตัวอักษร,ข้อความและกราฟฟิกใน ROM ทั้งสองชุดนั้นจะถูกกำหนดไว้ในลักษณะ ASCII Text File ซึ่งจะเป็น Memory Initialization File ของ ROM ทั้งสองชุดในส่วนการแสดงผล HEART RATE นั้น จะทำการรับสัญญาณ HEART RATE(DIGIT_IN) ที่ได้จาก BLOCK BIN_TO_BCD นำมาการประมวลผลแล้วทำการแสดงผลออกจอภาพ โดยแสดงผลเป็นตัวเลข ซึ่งในการแสดงผล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

นั้น BLOCK MESSAGES ได้ทำการรวมส่วนของการแสดงผลไว้ด้วยโดยหลักการทำงานนั้นเหมือนกับส่วนแรกที่ใช้ในการแสดงผลคลื่นหัวใจ

6.2.2 BLOCK BIN_TO_BCD

BLOCK BIN_TO_BCD ทำหน้าที่รับสัญญาณที่ได้จากBLOCK ..... มาทำการเข้ารหัสเพื่อเปลี่ยนสัญญาณที่ได้เป็นสัญญาณตัวเลขแล้วทำการส่งสัญญาณที่ได้นั้น ส่งไปยัง BLOCK MASSAGES เพื่อใช้ในการแสดงผลต่อไป ซึ่ง BLOCK BIN_TO_BCD ได้แสดงดังรูปที่ 6.22



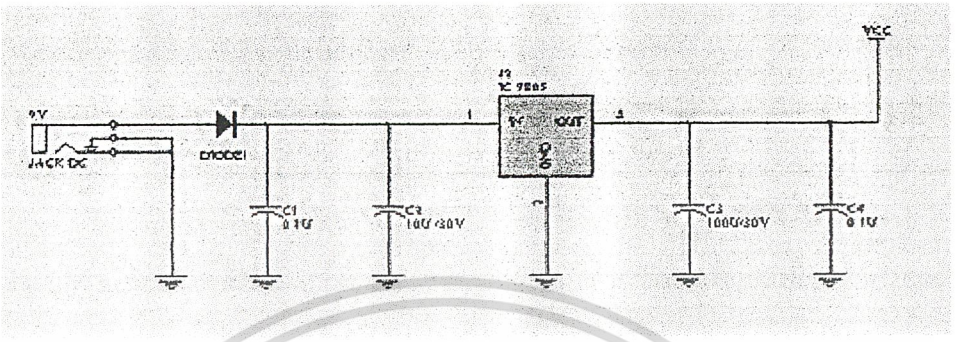
รูปที่ 6.22 แสดง BLOCK BIN_TO_BCD

โดยสัญญาณที่ใช้ในการควบคุมในการเข้ารหัสและส่งสัญญาณนั้นถูกควบคุมโดยสัญญาณ COL_ADDR_IN และ ROW_ADDR_IN เพื่อที่จะทำการกำหนดข้อมูลและลำดับข้อมูลที่ได้เข้ารหัสแล้ว ก่อนส่งไปยัง BLOCK MESSAGES

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.3 การออกแบบในส่วนของวงจร (Hardware)

6.3.1 การออกแบบภาคจ่ายไฟ (Regulator)

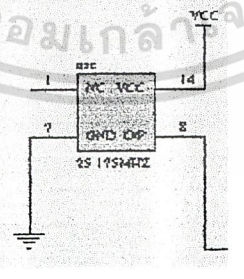


รูปที่ 6.23 วงจรภาคจ่ายไฟ

วงจรภาคจ่ายไฟจะทำหน้าที่เป็นตัวจ่ายแรงดันให้แก่วงจรอื่น ๆ รวมทั้ง FPGA ซึ่งวงจรภาคจ่ายไฟนี้จะใช้ IC เบอร์ 7805 ซึ่งเป็น IC REGULATE รักษาระดับแรงดันคงที่คือ 5 VOLT DC โดยมีคาปาซิเตอร์ เป็นตัวกรองระดับแรงดันให้เรียบ

6.2.2 OSCILLATOR

ในวงจรนี้จะใช้ OSCILLATOR แบบโมดูลออสซิลเลเตอร์ โดยที่โมดูลออสซิลเลเตอร์จะผลิตความถี่ที่ 25.175 MHz ป้อนให้ขา 55 (Global CLK) ของ FLEX เบอร์ EPF10K20TC144-4 ซึ่งเป็นขา CLOCK ดังรูปที่ 6.12

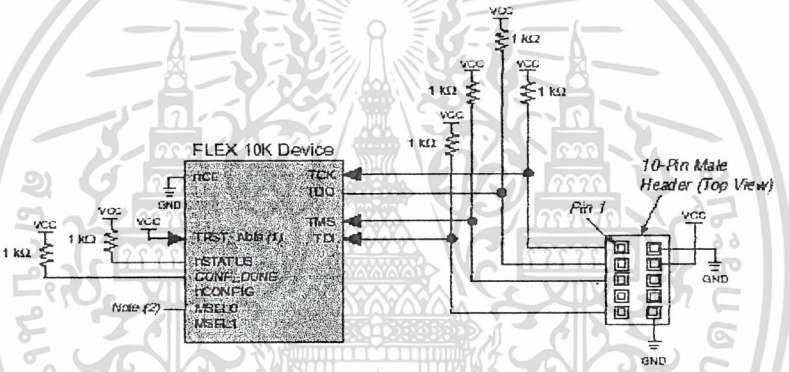


รูปที่ 6.24 โมดูลออสซิลเลเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.2.3 วงจรรวมตระกูล FLEX 10k ในอนุกรม EPF10K20TC144-4

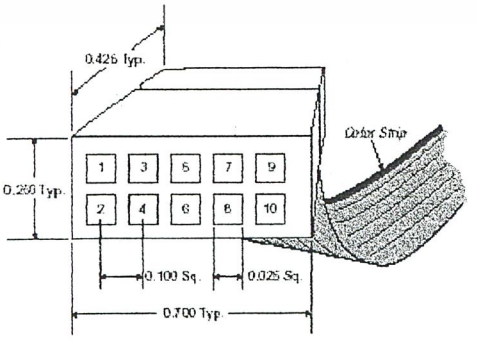
วงจรประเภทนี้จะใช้เทคโนโลยีเหมือน SRAM (Static RAM) ในการโปรแกรมทำให้สามารถ โปรแกรมซ้ำได้โดยไม่จำกัดจำนวนครั้ง นอกจากนี้ยังมีความจุเกท ในระดับปานกลางถึงสูงมาก (ประมาณ 10,000 – 1,000,000 เกท) ซึ่งข้อดีของ SRAM Based FPGA คือใช้เวลาในการโปรแกรมน้อย (ระดับ nSEC) การโปรแกรมทำได้ง่ายเทียบได้กับการเขียน SRAM ทั่วไป และ เหมาะสำหรับการออกแบบวงจรที่มีความสลับซับซ้อน ส่วนข้อเสีย คือไม่สามารถเก็บโปรแกรมในสภาวะที่ไม่มีไฟเลี้ยงได้ ดังนั้น FPGA ชนิดนี้จึงมักใช้คู่กับ Rom เพื่อเก็บโปรแกรมและทำการโหลดโปรแกรมลงในตัวชิพ ในขณะที่เริ่มต้นการใช้งาน ซึ่งการโหลดโปรแกรมนั้นทำได้ โดยการต่อ HEADER (ซึ่งใช้ต่อกับสาย Byblaster เพื่อดาวน์โหลด) กับวงจร FLEX ดังรูปที่ 6.13



รูปที่ 6.25 การต่อ FLEX กับอุปกรณ์เพื่อการดาวน์โหลด

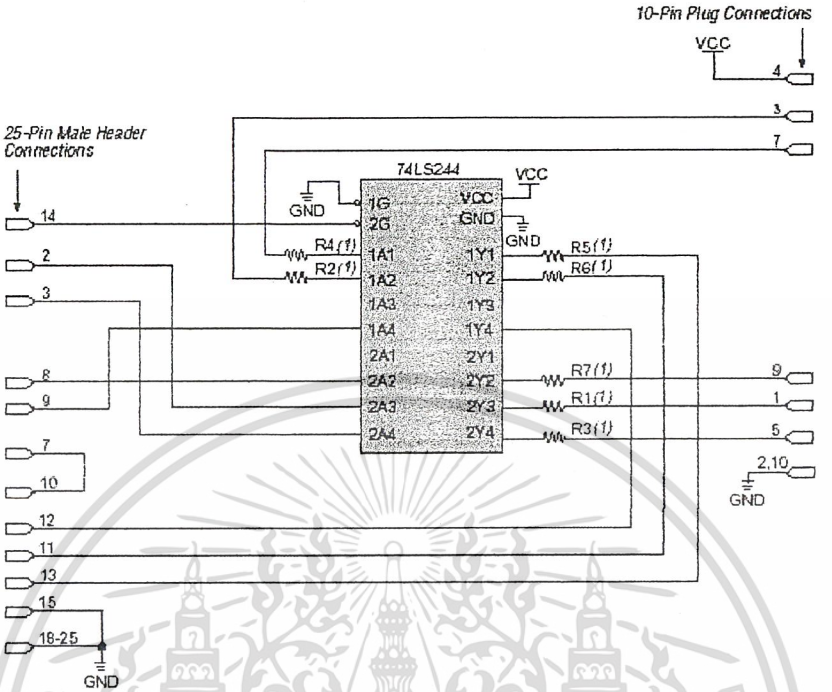
6.2.4 JTAG CONNECTOR

สำหรับการต่อสาย ByeBlaster ไว้สำหรับ Download ข้อมูลของลอจิกจากคอมพิวเตอร์ลงบนชิพไอซี ซึ่งจะมีตำแหน่งขา และการต่อกับสาย ByeBlaster ดังรูปที่ 6.14



รูปที่ 6.26 การต่อ JTAG กับ ByeBlaster

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 6.27 วงจรภายในของสาย ByeBlaster

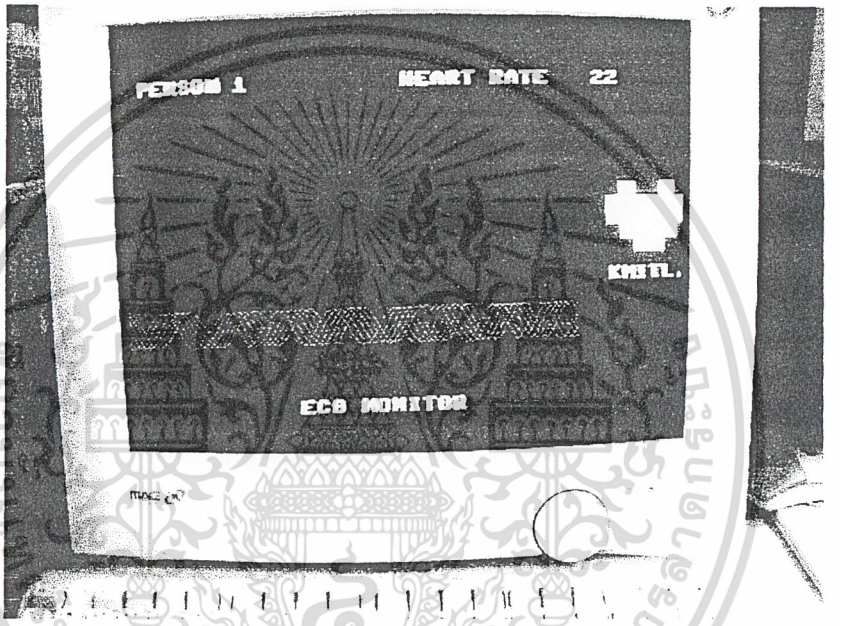
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 7

การทดลองและผลการทดลอง

ผลการทดลองโดยการ SIMULATE

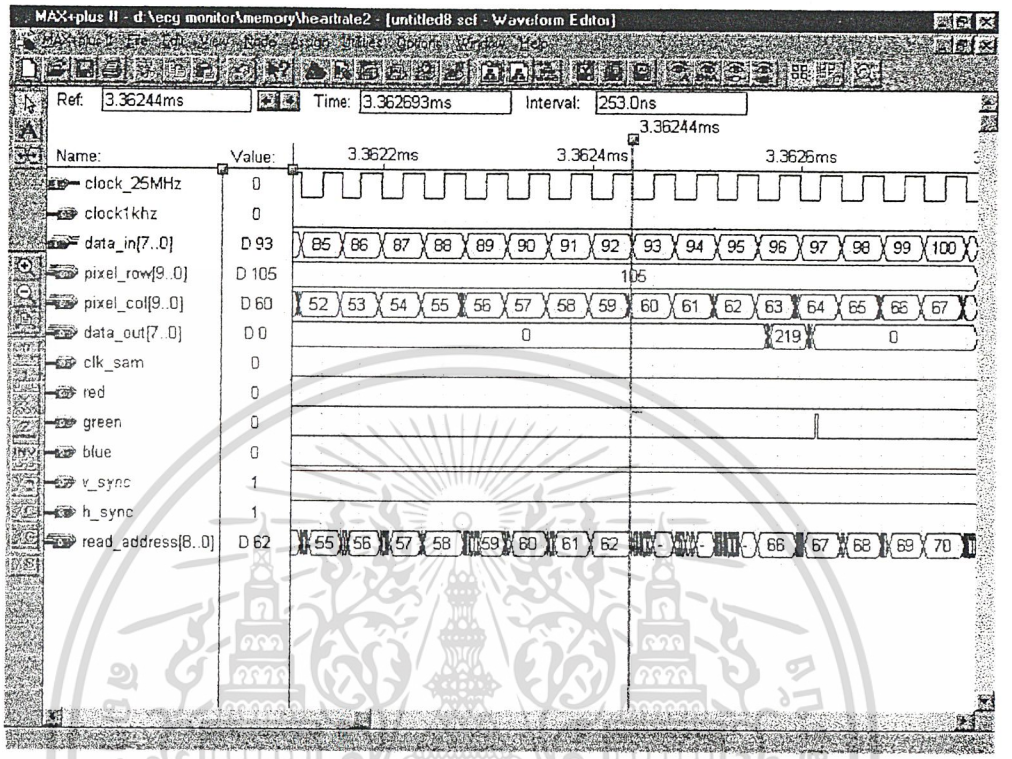
7.1 ผลการทดลองการรับสัญญาณ SINE มาทำการแสดงผล



รูปที่ 7.1 ผลการทดลองการรับสัญญาณ SINE มาแสดงผล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

7.2 ผลการทดลองส่วนภาคการแสดงผลคลื่นสัญญาณหัวใจ



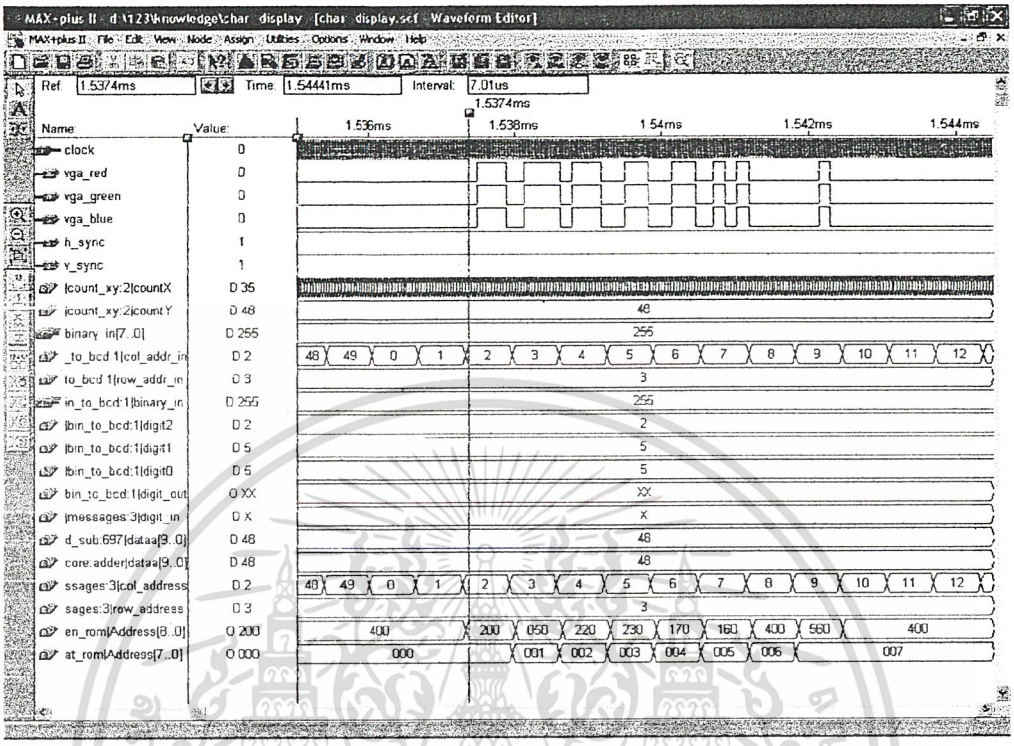
รูปที่ 7.2 แสดงผลการทดลองโดยการ SIMULATE ภาคการแสดงกราฟฟิก

จากผลการ Simulate จะได้ว่าในการสร้างภาพนั้นเราต้องเก็บข้อมูลที่ได้จากการ Sampling โดยการ Sampling แต่ละครั้งเราจะสร้างสัญญาณ ไปให้แก่ขา Clock ของ ADC และในการ Sampling แต่ละครั้งแอดเดรสที่ใช้ในการอ่านจะเลื่อนไปทีละ 2 ค่า ซึ่งเป็นเทคนิคในการสร้างภาพให้เลื่อนไปทางขวา ในการเขียนข้อมูลแอดเดรสที่เข้าขาของ SRAM จะเป็นแอดเดรสที่ใช้ในการเขียน และในกรณีที่ไม่มีการ Sampling แอดเดรสที่เข้าขาแอดเดรสของ SRAM จะเป็นแอดเดรสของการอ่าน ข้อมูลที่ได้จะเทียบกับค่าน้ำหนักของ Counter ของการสแกนทางแนวแถว ถ้าเท่ากันก็จะมีการสร้างจุดภาพจากผลการทดลอง จะเห็นว่าข้อมูลที่ใช้ในการแสดงภาพเป็นข้อมูลล่าสุดของการเก็บจะอยู่ที่แอดเดรสที่ 512 เสมอ

7.3 ผลการทดลองส่วนของภาคการแสดงผลตัวอักษรและกราฟฟิก

การทดลองนี้เราได้ทำการจำลองการทำงานในภาคแสดงผลทางด้าน กราฟฟิกเท่านั้น ไม่ได้รวมภาควัดสัญญาณคลื่นหัวใจนำไว้ด้วยด้วยกัน ซึ่งได้แสดงไว้ดังรูปที่ 7.1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 7.3 แสดงผลการทดลองโดยการ SIMULATE ภาคการแสดงผลตัวอักษรและกราฟฟิก

จากผลการทดลองเราจะได้ผลการทดลองดังรูปข้างบน ซึ่งในการออกแบบนั้นเราได้กำหนดให้ตัวอักษรที่จะแสดงนั้นอยู่บนแถวที่ 3 และเริ่มต้นแสดงผลหลักที่ 2 (ดังตำแหน่งที่ได้ขีดไว้)เริ่มแสดงผล โดย ROM ที่ใช้ในการเก็บข้อมูลรูปแบบการแสดงผลจะทำการชี้ตำแหน่งข้อมูลที่ได้ออกแบบไว้(ตำแหน่งล่างสุด ADDRESS [7..0]) จากนั้นข้อมูลที่ถูกชี้จะส่งไปยัง ROM อีกตัวเพื่อกำหนดว่าตัวอักษรที่ต้องการแสดงผลให้แสดงผลอย่างไร(ตำแหน่งถัดจากล่างสุดADDRESS [8..0]) และสุดท้ายจะนำข้อมูลที่ได้ออกแสดงผลออกจอภาพ(VGA_RED, GREEN, BLUE)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 8
สรุปผลการทดลอง

8.1 สรุปผลการทดลอง

จากผลการทดลองแสดงให้เห็นว่า สามารถออกแบบเครื่องวัดคลื่นสัญญาณหัวใจ ECG MONITOR ที่มีการทำงานแบบแบบเดียวกับเครื่องที่ใช้งานอยู่ทั่วไปในโรงพยาบาล ทำให้การออกแบบเครื่องมือทางการแพทย์ต้องการออกแบบมีราคาถูก และมีประสิทธิภาพใกล้เคียงกับที่ใช้ตามโรงพยาบาลทั่วไป และเทคนิควิธีการออกแบบที่ใช้ในโครงงานนี้ ได้ใช้เทคนิคการเขียนโปรแกรมภาษา VHDL โดยใช้ซอฟท์แวร์ของบริษัท ATERA ใช้โปรแกรม MAX+PLUSII ที่เวลาที่ใช้งานต้องโปรแกรมลงชิพ FPGA เบอร์ FLEX10k20CT144-4 ที่มีความจุเกต 20000 เกต เป็นชิพชนิด SRAM ใช้งานที่ความถี่สัญญาณนาฬิกาที่ 25 MHz แสดงผลการทำงานผ่านการแสดงผลที่จอ VGA MONITOR ลักษณะการแสดงผลเป็นแบบข้อความ(text) และการแสดงผลแบบกราฟฟิก (Graphic) ซึ่งการออกแบบจะได้ว่า

1. การออกแบบโดยใช้ภาษา VHDL เป็นภาษาที่มีความยืดหยุ่นในการใช้งานสูงมาก โดยผู้ออกแบบสามารถกำหนดลักษณะการทำงานให้มีการทำงานแบบใดก็ได้โดยไม่ต้องคำนึงถึงโครงสร้างภายในตัวไอซี

2. การออกแบบที่ต้องการให้เนื้อที่การต่อวงจรที่ประหยัดเนื้อที่มากที่สุดควรจะนำวิธีการเขียนโปรแกรมภาษา VHDL ไปใช้เพราะสามารถสร้างไอซีได้ตามคุณสมบัติที่ต้องการ

3. การออกแบบโดยใช้ภาษา VHDL เป็นภาษาที่มีความสามารถสูง สามารถตอบสนองการออกแบบที่ซับซ้อนได้

4. ต้นทุนและเวลาที่ใช้ออกแบบจะประหยัดได้มาก

5. ออกแบบเครื่องวัดคลื่นสัญญาณหัวใจ ECG MONITOR เมื่อนำไปใช้งานจริงสามารถออกแบบให้ใช้งานแทนเครื่องที่ใช้อยู่ได้

6. เครื่องวัดคลื่นสัญญาณหัวใจ ECG MONITOR ที่ออกแบบได้สามารถพกพาไปได้เนื่องจากมีน้ำหนักและขนาดเล็ก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

8.2 ปัญหาและแนวทางการแก้ปัญหา

1. ภาษา VHDL ที่ออกแบบมีความยากอยู่พอสมควรซึ่งผู้ออกแบบต้องออกแบบใช้เวลาการศึกษาการเขียน

2. โปรแกรมที่ใช้ออกแบบมีข้อจำกัดมากทำให้การออกแบบมีความยุ่งยากมากขึ้น และตัวโปรแกรมเองก็ไม่รองรับการคำสั่งการทำงานบางคำสั่งได้

3. ลักษณะการทำงานของเครื่องวัดสัญญาณคลื่นหัวใจ ที่ได้ยังมีข้อผิดพลาดอยู่บ้าง เช่นมีการแสดงสัญญาณรบกวน(NOISE) ออกมาปนกับการแสดงสัญญาณคลื่นหัวใจ ข้อความที่แสดงออกหน้าจอยังมีข้อผิดพลาดอยู่บ้าง

4. การออกแบบทีมีความผิดพลาดอยู่ทำให้ตัว ชิพ FPGA มีการสูญเสียทางด้านกำลังออกมามาก

5. ชิพ FPGA เบอร์ FLEX10k20CT144-4 เป็นชิพ ชนิด SRAM ทำให้เวลาเก็บข้อมูลต้องมีไฟเลี้ยงเสมอ

8.3 แนวทางการแก้ไขปัญหา

1. ผู้ออกแบบต้องมีความเข้าใจในตัวโครงสร้างภาษา VHDL มาก จึงจะอำนวยการออกได้ดี

2. การออกแบบที่มีประสิทธิภาพต้องออกแบบให้ประหยัดเนื้อที่เกตของ FPGA มากที่สุด

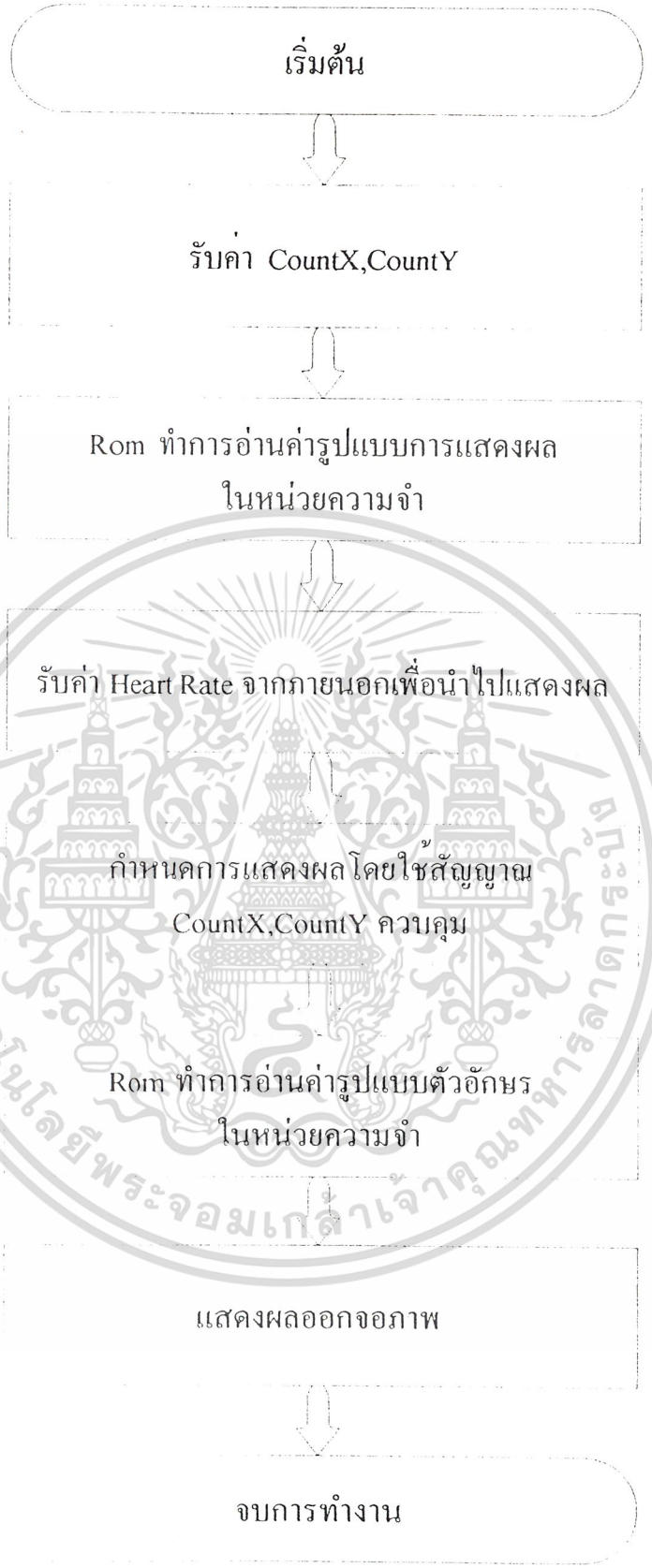
3. ผู้ออกแบบควรหาชิพ FPGA ที่มีราคาที่ถูกมาออกแบบเพราะปัจจุบันมีหลายบริษัทได้ผลิตชิพประเภทนี้ออกมาจำหน่ายมากมาย

4. ควรเลือกใช้โปรแกรมเกี่ยวกับการเขียนภาษา VHDL หลายๆโปรแกรมเพื่อเปรียบเทียบความสามารถในการออกแบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

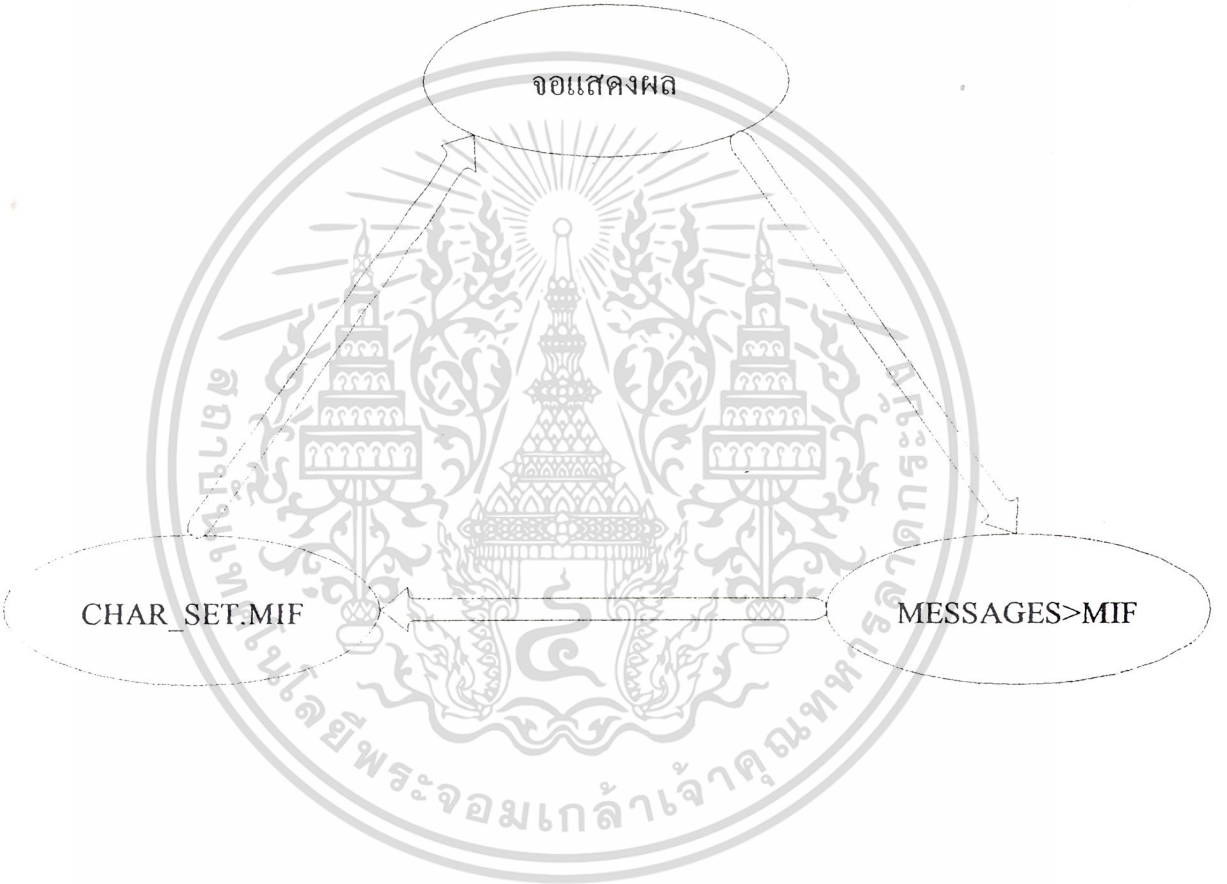
ภาคผนวก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



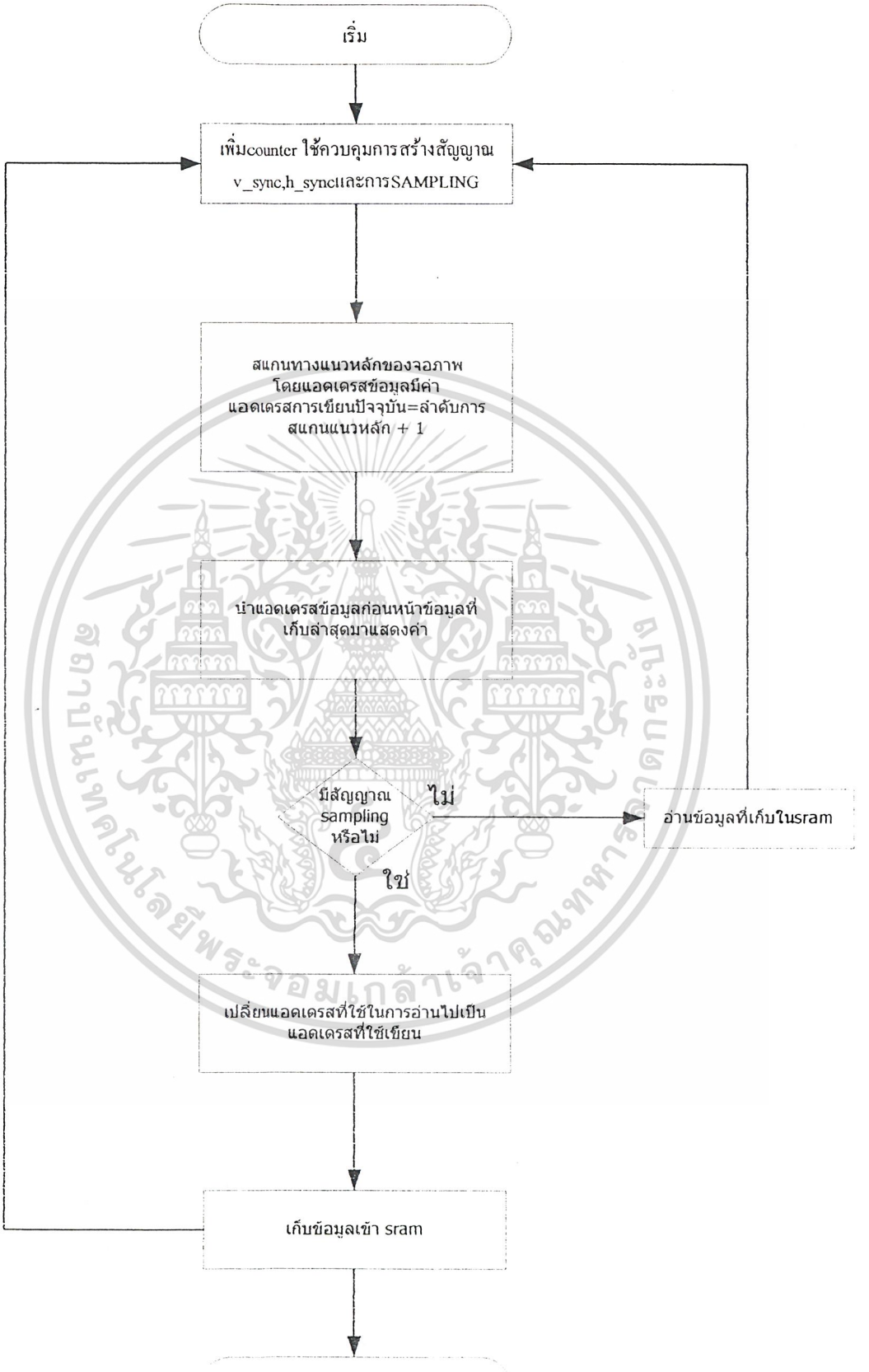
รูปที่ 1 ลำดับในการแสดงข้อความอักษรและกราฟฟิก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2 ลำดับในการแสดงข้อความอักษรและกราฟฟิก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
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;


library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity rgb_out_400 is
port(ri,gi ,bi  :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);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    red_out,green_out,blue_out    :out std_logic
      );
    end rgb_out_400;

architecture color of rgb_out_400 is
begin
 process
  variable sum      : std_logic_vector(9 downto 0);
 begin
              sum  := pixel_row + data_in;
       if  sum = 367  and pixel_col<=512 then
          red_out   <= '0';
          green_out <= '1';
          blue_out  <= '0';

      else
          red_out   <= ri ;
          green_out <= gi ;
          blue_out  <= bi ;
         end if;
 end process ;
end color;

library ieee;
use ieee.std_logic_arith.all;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity clock_400 is
port (clock_25MHz :in  std_logic;
     clock_400  :out std_logic;
     in_freez    :in std_logic;
     clk_sam   :out std_logic);
end clock_400 ;

architecture clock of clock_400 is
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
begin
 process(clock_25MHz)
  variable count : integer range 0 to 85000 ;
  begin
    wait until clock_25MHz'event and clock_25MHz='1';
     if in_freez ='1' then
      if count =  62500  then
           clock_400 <='1';
             count := 0 ;
          else
           clock_400 <='0';
           count :=count+1;
      if count > 62400  and count < 62500 then
           clk_sam <= '0';
       else
           clk_sam <='1' ;
        end if;
       end if;
      end if;
     end process;
 end clock;


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);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
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;

-----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_test 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_test ;

architecture hor of h_sync_test is
 signal hor_count : std_logic_vector(9 downto 0);
 begin
  process
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  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;
  end process;
 end hor;

library IEEE;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
library WORK;
entity write_modu320v1 is
port( clock_1kHz :in std_logic;
    recent_address:out std_logic_vector(8 downto 0));
end write_modu320v1;
architecture write_modu_behav of write_modu320v1 is
begin
 process(clock_1kHz)
 variable clock_count :std_logic_vector(8 downto 0);
 begin
if (clock_1kHz ='1') then
   clock_count :=clock_count+1;
   recent_address<=clock_count;
 end if;
 end process;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
end write_modu_behav;

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);
     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;

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
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
 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;

library IEEE;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_1164.all;

entity read_module_edit3 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_edit3;

architecture read_modu_behav of read_module_edit3 is
begin
 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 read_modu_behav;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
ibrary 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_edit2 is
port( clock_1kHz   :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_edit2;
architecture wr_rd_contrl_behav of wr_rd_contrl_edit2 is
 begin
  process
   begin
if clock_1KHz = '1' then
  address <= write_address;
else
  address <= read_address ;
end if;
end process;
 end wr_rd_contrl_behav;

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
library work;
entity cal_heart is
port( data     :in std_logic_vector(7 downto 0);
     en       :in std_logic;
     trig0    :out std_logic);

end cal_heart ;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
architecture cal_heart_behav of cal_heart is
begin
 process(en)
  variable buff :std_logic;
   begin
     if en'event and en ='0' then
       if data <200 then
            buff := '0';
        else
           buff := '1';
      end if;
     end if;
     trig0 <= buff;
 end process;
end cal_heart_behav ;

library IEEE;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity set is
port( incount :in std_logic;
      setclk     :in std_logic;
      en         :out std_logic;
      data_cal :out std_logic_vector(7 downto 0));
end set;

architecture set_be of set is
begin
 process(incount,setclk)
  variable count :integer;
  variable enbuf    :std_logic;
   begin
  if  setclk = '1' then
      count := 0;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
 elsif incount'event and incount ='1' then
       count  := count+1;
  end if;
 if count = 1 then
   enbuf := '1';
 else
   enbuf := '0';
 end if;
 en      <= enbuf;
  data_cal <= conv_std_logic_vector((count-1)*20,8);
end process;
end set_be;

library IEEE;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity trig is
port( trig1 :in std_logic;
       trig0 :in std_logic;
       en    :out std_logic);
end trig;

architecture trig_be of trig is
begin
 process(trig1)
  variable count :std_logic;
 begin
  if trig1='1' and trig0='1' then
    en <= '1';
    else
    en <= '0';
     end if ;
  end process;
 end trig_be;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity sec3 is
port (clk_25Mhz :in std_logic;
     en       :in std_logic;
     preset   :out std_logic;
     set      :out std_logic);
end sec3;

architecture sec3_be of sec3 is
begin
 process(clk_25Mhz,en)
  variable c :integer;
 begin
if (clk_25Mhz'event and clk_25Mhz ='1') then
  c:=c+1;
end if;
if en ='1' then
  c:= 0;
 end if;
if en = '0' then
 if c = 74995 then
  preset <= '1';
else
  preset <= '0';
end if;

if c = 75000   then
  set <= '1';
else
  set <= '0' ;
end if;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
end if;
end process;
end sec3_be;

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
entity dataff is
port( data:in std_logic_vector(7 downto 0);
      preset :in std_logic;
      q:out std_logic_vector(7 downto 0));
end dataff;
architecture dataff_behav of dataff is
begin
 process (preset)
    begin
      IF (preset= '1') THEN
       if data < 200 then
        q <= data;
       end if;
       end if;
   END PROCESS ;
 end dataff_behav;

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.all;
USE IEEE.STD_LOGIC_ARITH.all;
USE IEEE.STD_LOGIC_UNSIGNED.all;

ENTITY count_xy IS
        PORT(
                Clock                   : IN std_logic;
                countX,countY           : OUT STD_LOGIC_VECTOR(9 DOWNTO 0);
                H_Sync_out,V_Sync_out   : OUT STD_LOGIC);
    END count_xy;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
ARCHITECTURE behavior OF count_xy IS
        SIGNAL Hor_count,Ver_count          : STD_LOGIC_VECTOR(9 DOWNTO 0);
BEGIN
        countX   <= Hor_count;
        countY   <= Ver_count;
PROCESS
        BEGIN
                WAIT UNTIL(Clock'Event) AND (Clock='1');
                -- H_count counts pixels (640 + extra time for sync signals)
                --
                --  <-Clock out RGB Pixel Row Data ->  <-H Sync->
                --  ------------------------------------_____------
                --  0                   640  659      755   799
                --
                        IF (Hor_count >= 799) THEN
                                Hor_count <= "0000000000";
                        ELSE
                                Hor_count <= Hor_count + '1';
                        END IF;

                --Generate Horizontal Sync Signal
                        IF (Hor_count <= 755) AND (Hor_count >= 659) THEN
                                H_Sync_out <= '0';
                        ELSE
                                H_Sync_out <= '1';
                        END IF;

                --V_count counts rows of pixels (480 + extra time for sync signals)
                --
                --  <---- 480 Horizontal Syncs (pixel rows) -->  ->V Sync<-
                --  ---------------------------------------------_________---------
                --  0                             480   493-494     524
                --
                        IF (Ver_count >= 524) AND (Hor_count >= 699) THEN
                                Ver_count <= "0000000000";
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                ELSE IF (Hor_count = 699) THEN
                        Ver_count <= Ver_count + '1';
                END IF;

        -- Generate Vertical Sync Signal
                IF (Ver_count <= 494) AND (Ver_count >= 493) THEN
                        V_Sync_out <= '0';
                ELSE
                        V_Sync_out <= '1';
                END IF;
        END IF;
    END PROCESS;
END behavior;


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 messages IS
    PORT
    (
        clock                                   : IN STD_LOGIC;
        digit_in                                : IN STD_LOGIC_VECTOR(5
DOWNTO 0);
        countX,countY                           : IN STD_LOGIC_VECTOR(9
DOWNTO 0);
        vga_red, vga_green, vga_blue            : OUT BOOLEAN;
        col_addr_out,row_addr_out               : OUT STD_LOGIC_VECTOR(5 DOWNTO 0)
    );
END messages;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
ARCHITECTURE behavior OF messages IS
        SIGNAL red_data,green_data, blue_data                                   :
boolean;
        SIGNAL in_screen, in_screen_vertical, in_screen_horizontal      : boolean;
        SIGNAL rom_mux_output
                : std_logic;
        SIGNAL rom_address
                        : std_logic_vector(8 DOWNTO 0);
        SIGNAL rom_data
                        : std_logic_vector(7 DOWNTO 0);
        SIGNAL col_address, row_address, con_row, con_col                       : std_logic_vector(5
DOWNTO 0);
        SIGNAL pixel_col_count, pixel_row_count                                 :
std_logic_vector(5 DOWNTO 0);
        SIGNAL format_address
                : std_logic_vector(7 DOWNTO 0);
        SIGNAL format_data
                        : std_logic_vector(5 DOWNTO 0);
BEGIN
char_gen_rom: lpm_rom
    GENERIC MAP (
                    lpm_widthad             => 9,
     lpm_numwords           => 512,
     lpm_outdata            => "UNREGISTERED",
     lpm_address_control => "UNREGISTERED",
     lpm_file               => "CHAR_SET.MIF",
     lpm_width              => 8 )
    PORT MAP ( address => rom_address, q => rom_data);

format_rom: lpm_rom
    GENERIC MAP (
                lpm_widthad             => 8,
     lpm_numwords           => 256,
     lpm_outdata            => "UNREGISTERED",
     lpm_address_control => "UNREGISTERED",
     lpm_file               => "MESSAGES.MIF",
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    lpm_width                => 6)
    PORT MAP ( address => format_address, q => format_data);


output : PROCESS(clock)
BEGIN
        IF clock'EVENT AND clock= '1' THEN
                red_data                <= (rom_mux_output = '1');
                Green_data              <= (rom_mux_output = '1');
                Blue_data               <= (rom_mux_output = '1');
                vga_Red                 <= red_data   AND in_screen;
                vga_Green               <= green_data AND in_screen;
                vga_Blue                <= blue_data  AND in_screen;
                in_screen               <= in_screen_horizontal AND
in_screen_vertical;
                rom_address(2 DOWNTO 0) <= pixel_row_count(3 DOWNTO 1);
                rom_mux_output          <= rom_data ( (CONV_INTEGER(NOT
pixel_col_count(3 DOWNTO 1))));
        END IF;
END PROCESS output;

video_display: PROCESS(clock)
BEGIN
        IF clock'event AND clock ='1' THEN
                col_address             <= countX(9 DOWNTO 4);
                row_address             <= countY(9 DOWNTO 4);
                col_addr_out            <= col_address;
                row_addr_out            <= row_address;
                pixel_col_count         <= countX(3 DOWNTO 0);
                pixel_row_count         <= countY(3 DOWNTO 0);
                in_screen_vertical      <= row_address >= "000000" AND row_address <= "011101";
                in_screen_horizontal    <= col_address >= "000000" AND col_address <= "100111";
        END IF;

END PROCESS video_display;

                format_address(3 DOWNTO 0) <= con_col(3 DOWNTO 0);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        format_address(7 DOWNTO 4) <= con_row(3 DOWNTO 0);

VIDEO_DISPLAY_DATA: PROCESS(clock)
BEGIN
        IF clock'EVENT AND clock = '1' THEN
        -- displaying "PERSON 1"
                IF col_address >= "000010" AND col_address <= "001001" AND row_address = "000011"
THEN
                                con_col <= col_address - "000010";
                                con_row <= "000000";
                                rom_address(8 DOWNTO 3) <= format_data;

        -- displaying "HEART RATE"
                ELSIF col_address >= "010100" AND col_address <= "011101" AND row_address =
"000011" THEN
                                con_col <= col_address - "010100";
                                con_row <= "000001";
                                rom_address(8 DOWNTO 3) <= format_data;

        -- displaying "ECG MONITOR"
                ELSIF col_address >= "001101" AND col_address <= "010111" AND row_address =
"011010" THEN
                                con_col <= col_address - "001101";
                                con_row <= "000010";
                                rom_address(8 DOWNTO 3) <= format_data;

        -- displaying "RATE(NUM)"
                ELSIF col_address >= "100000" AND col_address <= "100010" AND row_address =
"000011" THEN
                                rom_address(8 DOWNTO 3) <= digit_in;

        -- displaying "HEART"

                ELSIF (col_address = "100011" OR col_address = "100110") AND row_address =
"001011" THEN
                                rom_address(8 DOWNTO 3) <= "111111";
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                ELSIF col_address >= "100010" AND col_address <= "100111" AND row_address =
"001100" THEN
                        rom_address(8 DOWNTO 3) <= "111111";

                ELSIF col_address >= "100010" AND col_address <= "100111" AND row_address =
"001101" THEN
                        rom_address(8 DOWNTO 3) <= "111111";

                ELSIF col_address >= "100011" AND col_address <= "100110" AND row_address =
"001110" THEN
                        rom_address(8 DOWNTO 3) <= "111111";

                ELSIF (col_address = "100100" OR col_address = "100101") AND row_address =
"001111" THEN
                        rom_address(8 DOWNTO 3) <= "111111";

        -- displaying "KMITL."
                ELSIF col_address >= "100010" AND col_address <= "100111" AND row_address =
"010001" THEN
                        con_col <= col_address - "100010";
                        con_row <= "000011";
                        rom_address(8 DOWNTO 3) <= format_data;

                ELSE    rom_address(8 DOWNTO 3) <= "100000";

                END IF;
        END IF;
END PROCESS VIDEO_DISPLAY_DATA;
END behavior;


LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.all;
USE IEEE.STD_LOGIC_ARITH.all;
USE IEEE.STD_LOGIC_UNSIGNED.all;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
ENTITY bin_to_bcd IS
        PORT
        (
                clock                                                   : IN STD_LOGIC;
                col_addr_in,row_addr_in                 : IN STD_LOGIC_VECTOR(5 DOWNTO
0);
                binary_in                                               : IN STD_LOGIC_VECTOR(7
DOWNTO 0);
                digit_out                                               : OUT STD_LOGIC_VECTOR(5
DOWNTO 0)
        );
        END bin_to_bcd;

ARCHITECTURE behavior OF bin_to_bcd IS
        SIGNAL digit2                                           : STD_LOGIC_VECTOR(1
DOWNTO 0);
        SIGNAL digit1,digit0                            : STD_LOGIC_VECTOR(3 DOWNTO 0);
        SIGNAL          digit_2,digit_1,digit_0         : STD_LOGIC_VECTOR(5 DOWNTO 0);

BEGIN
bin_to_bcd: PROCESS(clock)
        VARIABLE digit                                          : STD_LOGIC_VECTOR(17
DOWNTO 0);
BEGIN
        IF clock'event AND clock ='1' THEN
                        digit(17 DOWNTO 8) :="0000000000";
                        digit(7 DOWNTO 0) := binary_in;

                FOR I IN 1 TO 8 LOOP
                        IF  digit(11 DOWNTO 8) >= 5 THEN
                                digit(11 DOWNTO 8) := digit(11 DOWNTO 8) + "0011";
                        END IF;

                        IF  digit(15 DOWNTO 12) >= 5 THEN
                                digit(15 DOWNTO 12) := digit(15 DOWNTO 12) + "0011";
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                END IF;

                digit := digit + digit;
            END LOOP;
            digit2 <= digit(17 DOWNTO 16);
            digit1 <= digit(15 DOWNTO 12);
            digit0 <= digit(11 DOWNTO 8);

            END IF;
END PROCESS bin_to_bcd;

encoder: PROCESS(clock)
BEGIN
    IF clock'event AND clock ='1' THEN
        IF      digit2 = "00" THEN digit_2 <= "100000";
        ELSIF digit2 = "01" THEN digit_2 <= "101110";
        ELSIF digit2 = "10" THEN digit_2 <= "101111";
        END IF;

        IF      digit1 = "0000" AND digit2 /= "00" THEN digit_1 <= "101101";
        ELSIF digit1 = "0000" AND digit2  = "00" THEN digit_1 <= "100000";
        ELSIF digit1 = "0001" THEN digit_1 <= "101110";
        ELSIF digit1 = "0010" THEN digit_1 <= "101111";
        ELSIF digit1 = "0011" THEN digit_1 <= "110000";
        ELSIF digit1 = "0100" THEN digit_1 <= "110001";
        ELSIF digit1 = "0101" THEN digit_1 <= "110010";
        ELSIF digit1 = "0110" THEN digit_1 <= "110011";
        ELSIF digit1 = "0111" THEN digit_1 <= "110100";
        ELSIF digit1 = "1000" THEN digit_1 <= "110101";
        ELSIF digit1 = "1001" THEN digit_1 <= "110110";
        END IF;

        IF      digit0 = "0000" THEN digit_0 <= "101101";
        ELSIF digit0 = "0001" THEN digit_0 <= "101110";
        ELSIF digit0 = "0010" THEN digit_0 <= "101111";
        ELSIF digit0 = "0011" THEN digit_0 <= "110000";
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                ELSIF digit0 = "0100" THEN digit_0 <= "110001";
                ELSIF digit0 = "0101" THEN digit_0 <= "110010";
                ELSIF digit0 = "0110" THEN digit_0 <= "110011";
                ELSIF digit0 = "0111" THEN digit_0 <= "110100";
                ELSIF digit0 = "1000" THEN digit_0 <= "110101";
                ELSIF digit0 = "1001" THEN digit_0 <= "110110";
                END IF;
        END IF;
END PROCESS encoder;

switch: PROCESS
BEGIN
                IF   col_addr_in = "100000" AND row_addr_in = "000011" THEN digit_out <= digit_2;
                ELSIF col_addr_in = "100001" AND row_addr_in = "000011" THEN digit_out <= digit_1;
                ELSIF col_addr_in = "100010" AND row_addr_in = "000011" THEN digit_out <= digit_0;
                END IF;
END PROCESS switch;
END behavior;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารอ้างอิง

ก. เอกสารการอ้างอิงที่มีเป็นวารสารภาษาอังกฤษ จัดเรียงตามลำดับ

1. Stefan Sjoholm,Lennart Lindh; **VHDL DESIGNERS**,ABB Industrial s ystem,Sweden,1997

2. Gerard J. Tortara and Sandra Reynolds Grabowski, **Principles of Anatomy and Physiology**, HarperCollinsCollegePublishers, New York, 1996

3. Alexander P. Spence, **Basic Human Anatomy**, The Benjamin / Cummings Publishing Company Inc., California

ข. เอกสารการอ้างอิงที่มีเป็นวารสารภาษาไทย จัดเรียงตามลำดับ

1. การทดลองทางอิเล็กทรอนิกส์ 3, **การประยุกต์ใช้ FPGA**, ภาควิชาอิเล็กทรอนิกส์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

2. สมศรี คาวฉาย, **อุปกรณ์การแพทย์สำหรับหอผู้ป่วยหนัก**, ภาควิชาสรีรวิทยา คณะแพทย์ศาสตร์ศิริราช พยาบาลม กรุงเทพมหานคร 2539

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้