การประยุกต์ใช้งาน FPGA กับอุปกรณ์ตรวจสอบอุณหภูมิ

TEMPERATURE MEASUREMENT APPLICATION BY FPGA

โดย

นายศุภชัย งามอเนกรัตน์

นายมารุต ใจซื่อ

รายงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

การประยุกต์ใช้งาน FPGA กับอุปกรณ์ตรวจสอบอุณหภูมิ

TEMPERATURE MEASUREMENT APPLICATION BY FPGA

โดย

นายศุภชัย งามอเนกรัตน์

นายมารุต ใจซื่อ

อาจารย์ที่ปรึกษา

รศ.ดร.มนัส สังวรศิลป์

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

การประยุกต์ใช้งาน FPGA กับอุปกรณ์ตรวจสอบอุณหภูมิ

TEMPERATURE MEASUREMENT APPLICATION BY FPGA

1. นายศุภชัย งามอเนกรัตน์ รหัส 42015241
2. นายมารุต ใจซื่อ รหัส 42515737

ปริญญานิพนธ์นี้ได้รับการตรวจสอบแล้ว พร้อมที่จะทำการสอบได้

.....

(รศ.ดร.มนัส สังวรศิลป์)

อาจารณ์ที่ปรึกษา

ปริญญานิพนธ์ ปีการศึกษา 2544

ภาควิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การประยุกต์ใช้งาน FPGA กับอุปกรณ์ตรวจสอบอุณหภูมิ

ผู้จัดทำ

1. นายศุภชัย งามอเนกรัตน์ รหัส 42015241
2. นายมารุต ใจซื่อ รหัส 42515737

.....อาจารย์ที่ปรึกษา

(รศ.ดร.มนัส สังวรศิลป์)

การประยุกต์ใช้งาน FPGA กับอุปกรณ์ตรวจสอบอุณหภูมิ

นายศุภชัย งามอเนกรัตน์
นายมารุต ใจซื่อ
รศ.ดร.มนัส สังวรศิลป์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2544

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้เป็นการนำเสนอการประยุกต์ใช้งาน FPGA กับอุปกรณ์ตรวจสอบอุณหภูมิโดยนำข้อมูลการวัดอุณหภูมิจากอุปกรณ์การวัดอุณหภูมิ IC DS1620 ส่งผ่านไปแสดงผลยังจอ 7-Segment และส่งข้อมูลเชื่อมต่อไปยังคอมพิวเตอร์แบบอนุกรมผ่านอุปกรณ์ RS232 จากนั้นนำข้อมูลที่ได้ไปประยุกต์ใช้งานกับซอร์ฟแวร์ที่เขียนขึ้นด้วยภาษา Microsoft Visual C++ การทำงานจะแบ่งเป็น 2 โหมดการทำงานคือ Real Times เป็นการวัดอุณภูมิเวลาต่อเวลาและ Record Times จะสามารถบันทึกข้อมูลที่ต้องการวัดเก็บไว้ใน RAM ซึ่งจะใช้ FPGA เป็นตัวประมวลผลและกำหนดการทำงานทั้งหมด

Temperature Measurement Application by FPGA

Mr. Supachai Ngamanekrat
Mr. Marut Jaisue
Dr. Manas Sangvarasin Advisor
2001

Abstract

This thesis is presented application of Field Programmable Gate Array (FPGA) for temperature measurement application. System to brought data of thermometer equipment (IC DS1620) sent the result to 7-Segment and connected to computer series type. After used the software to apply and show data by Microsoft Visual C++ Language. Modes of process were real times mode and record time's mode. Real times mode examined real temperature and showed in present time. Record time's mode saved temperature data in memory (RAM) and read temperature data from memory to show in next time. The total process used FPGA to control.

สารบัญ

สารบัญรูป

หน้า

หน้า

หน้า

บทที่ 1
บทนำ

ความเป็นมาของโครงการ

FPGA เป็นอุปกรณ์ที่สามารถโปรแกรมได้ โดยใช้สำหรับวงจรที่ได้ทำการออกแบบลงไปเพื่อให้อุปกรณ์ FPGA มีฟังชันท์การทำงานตามแบบที่ต้องการ และในที่นี้จะเป็นการออกแบบอุปกรณ์ FPGA ให้ทำงานควบคุมอุปกรณ์วัดอุณหภูมิและเชื่อมต่อกับคอมพิวเตอร์ให้สามารถวัดและแสดงผลเพื่อใช้เก็บข้อมูลอุณหภูมิในงานการทดลองวิจัยต่างๆ เช่น ในการทดลองทางวิทยาศาสตร์ที่จะนำข้อมูลอุณหภูมิไปวิเคราะห์ ซึ่งคงไม่สะดวกแน่เมื่อต้องนั่งบันทึกข้อมูลแล้วนำไปพล็อตค่าในระยะเวลาหลายๆ ชั่วโมงด้วยตัวเอง ดังนั้นโครงงานนี้จึงช่วยให้สามารถวัดและบันทึกค่าอุณหภูมิต่างๆ ได้ง่ายขึ้น

วัตถุประสงค์ของโครงการ

1. ศึกษาขั้นตอนการใช้งาน IC FLEX 10K
2. ศึกษาการเชื่อมต่อ IC FLEX 10K ร่วมกับอุปกรณ์อินพุทและเอาท์พุท
3. ศึกษาการเขียนโปรแกรมด้วยภาษา VHDL ควบคุมการทำงาน
4. ศึกษาการนำ FPGA ไปประยุกต์ใช้งานกับการวัดอุณหภูมิ
5. เพื่อนำผลที่วัดได้ไปประยุกต์ใช้งานและแสดงบนคอมพิวเตอร์ได้
6. เพื่อบันทึกค่าอุณหภูมิในช่วงเวลาต่างๆเก็บไว้ในหน่วยความจำได้

ขอบเขตของโครงการ

ออกแบบการทำงานด้วยภาษา VHDL เพื่อใช้โปรแกรมลงในอุปกรณ์ FPGA ให้ควบคุมทำงานอ่านค่าอุณหภูมิจากอุปกรณ์การวัดอุณหภูมิ (IC DS1620) แล้วนำค่าไปแสดงผลผ่าน 7-Segments และสามารถบันทึกค่าเก็บไว้ในหน่วยความจำ (RAM) ในเวลาที่ต้องการบันทึก จากนั้นนำข้อมูลต่างๆ ส่งไปพล็อตและแสดงค่าในคอมพิวเตอร์ผ่านพอร์ตอนุกรม RS232

ขีดความสามารถของโครงการ

1. วัดอุณหภูมิแบบ Real Times และแบบ Record Times
2. สามารถวัดและบันทึกค่าอุณหภูมิในช่วง 0 – 125 °C ในระยะเวลาสูงสุด 68 ชม.
3. เชื่อมต่อข้อมูลกับคอมพิวเตอร์ผ่านพอร์ตอนุกรม RS232
4. สามารถแสดงผลของการวัดและตรวจสอบอุณหภูมิผ่านทางคอมพิวเตอร์ได้
5. นำค่าที่วัดได้ไปพล็อตแสดงความสัมพันธ์ระหว่างอุณหภูมิกับเวลา

ประโยชน์ที่ได้รับ

1. ได้รู้ถึงวิธีใช้งานและเชื่อมต่อ FPGA กับอุปกรณ์ภายนอก
2. ศึกษาการออกแบบด้วยอุปกรณ์ FPGA กับตัววัดอุณหภูมิและสามารถนำไปใช้วัดอุณหภูมิได้จริงในงานการทดลองต่างๆ

บทที่ 2

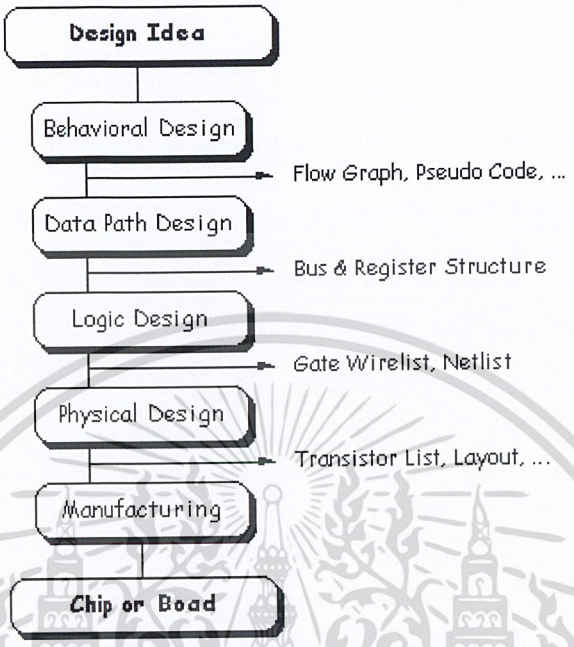
ภาษา วีเอชดีแอล

ภาษา วีเอชดีแอล

ความซับซ้อนและขนาดของระบบดิจิตอลในปัจจุบันได้เพิ่มมากขึ้นทุกขณะส่งผลให้มีการนำคอมพิวเตอร์เพื่อช่วยใน การออกแบบหรือ CAD มาใช้ในขบวนการออกแบบฮาร์ดแวร์เพิ่มขึ้นเช่นกัน อีกทั้งอุปกรณ์และวิธีการ ออกแบบใหม่ๆ ก็ถูกพัฒนาขึ้นมาเพื่อช่วยอำนวยความสะดวกให้กับนักออกแบบมากขึ้นด้วย สำหรับภาษาบรรยายอุปกรณ์ฮาร์ดแวร์ (HDL : Hardware Description Language) ก็เป็นเครื่องมืออย่างหนึ่งที่ได้รับการพัฒนามาอย่างต่อเนื่อง เพื่อช่วยให้การปรับปรุงขบวนการออกแบบระบบดิจิตอลเป็นไปอย่างมีประสิทธิภาพ

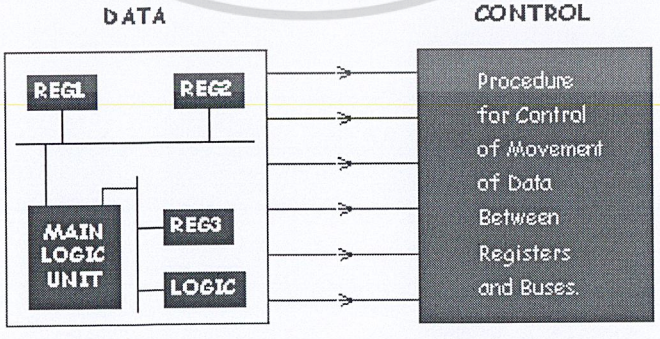
การออกแบบระบบดิจิตอล

ในการออกแบบระบบดิจิตอล เริ่มตั้งแต่การกำหนดแนวความคิดเบื้องต้นจนกระทั่งได้ออกมาเป็นอุปกรณ์ฮาร์ดแวร์ ที่ใช้งานได้จะต้องผ่านขั้นตอนต่างๆ มากมาย และในแต่ละขั้นตอนผู้ออกแบบจะต้องตรวจสอบผลลัพธ์ในแต่ละขั้น ก่อนเข้าสู่กระบวนการออกแบบในขั้นต่อไป รูปที่ 2.1 แสดงขั้นตอนปกติที่ใช้ในการออกแบบระบบดิจิตอลทั่วไป ขั้น แรกผู้ออกแบบจะกำหนดแนวความคิดในการออกแบบแล้วทำการพัฒนาให้สามารถนำมาใช้ได้อย่างสมบรูณ์ ซึ่งภาย ในขั้นตอนนี้ผู้ออกแบบจำเป็นต้องสร้างรูปแบบระบบในเชิงพฤติกรรมขึ้นมาตรวจสอบซึ่งอาจจะเป็นผังงานแสดงแบบหรือ รหัสคำสั่งเทียม (Pseudo code) ก็ได้



รูปที่ 2.1 แสดงขั้นตอนการออกแบบระบบดิจิตอล

ขั้นตอนต่อไปเป็นการออกแบบระบบเส้นทางของข้อมูล ผู้ออกแบบจะกำหนดส่วนประกอบของรีจิสเตอร์และวงจรลอจิก ที่จำเป็นทั้งหมดเพื่อนำมาประกอบเป็นระบบที่สมบูรณ์ โดยแต่ละองค์ประกอบสามารถเชื่อมต่อกันด้วยบัสหนึ่งหรือสอง ทิศทาง (Unidirectional or Bidirectional Bus) ส่วนกระบวนการในการควบคุมการเคลื่อนย้ายข้อมูลระหว่าง รีจิสเตอร์และวงจรลอจิกจะขึ้นอยู่กับพฤติกรรมของระบบที่กำหนดไว้ดังรูปที่ 2.2



รูปที่ 2.2 การออกแบบระบบเส้นทางของข้อมูล

ขั้นตอนถัดมาเป็นการออกแบบวงจรลอจิก ซึ่งจะเกี่ยวข้องกับการนำเกทดิจิตอลพื้นฐานและฟลิปฟลอป (flip-flop) มาประกอบเป็นอุปกรณ์ย่อยต่างๆ เช่น รีจิสเตอร์เก็บข้อมูล บัสวงจรลอจิก และส่วนควบคุมฮาร์ดแวร์ ซึ่งผลลัพธ์ ที่ได้ในขั้นตอนนี้จะเป็นเครือข่ายของการโยงใยระหว่างเกทและฟลิปฟลอปนั่นเองการออกแบบในขั้นตอนถัดไป เป็นการเปลี่ยนเครือข่ายการโยงใยที่ได้จากขั้นตอนที่แล้วให้เป็นลำดับของทรานซิสเตอร์ (Transistor List) และ Layout ซึ่งขั้นตอนนี้จะเกี่ยวข้องโดยตรงกับการจัดวางทรานซิสเตอร์หรือไลบรารีเซลล์เพื่อ แทนเกทและฟลิปฟลอปต่างๆและในขั้นตอนสุดท้ายจะเป็นการส่งระบบที่ออกแบบไว้ไปทำการเจือสารที่โรงงานเพื่อผลิตออกมาเป็น วงจรรวมในที่สุด

1.1 ประวัติความเป็นมาของภาษา VHDL

VHDL ย่อมาจากคำว่า VHSIC Hardware Description Language (VHSIC : Very High Speed Integrated Circuit) เป็นภาษาโปรแกรมระดับสูง (High Level Language) ที่ใช้สำหรับการออกแบบฮาร์ดแวร์ในระบบดิจิตอล ตัวของภาษาสามารถบรรยายพฤติกรรมการทำงานในรูปของลำดับชั้น (Hierarchy) และ สามารถเขียนได้หลายรูปแบบ ด้วยเหตุผลนี้จึงทำให้ภาษา VHDL เป็นเครื่องมือที่ใช้ออกแบบตั้งแต่ขั้นตอนบนสุด คือ แนวความคิดที่จะแก้ปัญหา ลงไปทีละขั้นจนถึงขั้นตอนของการสร้างวงจรจริง และตัวภาษาก็เปิดโอกาสให้วิศวกร ได้พัฒนาและจำลองการทำงานของรูปแบบฟังก์ชันการทำงานของวงจรอย่างสังเขป โดยยังไม่ต้องคำนึงถึงรายละเอียดเกี่ยวกับโครงสร้างวงจรจริง นอกจากนั้น VHDL ยังเป็นภาษาที่สนับสนุนลักษณะต่างๆ ของระบบดิจิตอลที่มี ความซับซ้อนได้ทั้งหมด ดังนั้น VHDL จึงเป็นภาษาที่น่าสนใจในการศึกษาและนำไปใช้งานเป็นอย่างยิ่ง วิวัฒนาการของภาษา VHDL เริ่มต้นประมาณปี ค.ศ. 1981 เมื่อกระทรวงกลาโหมสหรัฐอเมริกา หรือ DoD (Department of Defense) ได้พยายามปรับปรุงอุปกรณ์อิเล็กทรอนิกส์และคอมพิวเตอร์ที่ใช้ในกิจการทางทหาร ให้มีความทันสมัยมากขึ้น ประกอบกับเทคโนโลยีทางด้านไมโครอิเล็กทรอนิกส์มีการพัฒนาไปอย่างรวดเร็วดังจะเห็นได้จากการนำวงจรดิจิตอลหลายๆ วงจรมาทำการผลิตอยู่บนแผ่นซิลิกอนที่มีพื้นที่เพียง 1 - 2 ตารางเซนติเมตรเท่านั้น ซึ่ง เป็นผลให้ประสิทธิภาพในการทำงานของวงจรสูงขึ้นตลอดจนความน่าเชื่อถือ ในการทำงานและความคงทนต่อสภาพ แวดล้อมสูง แต่เนื่องจากในขณะนั้นขั้นตอนของการออกแบบ การผลิต และการตรวจสอบวงจรต้นแบบ เป็นขบวนการที่ ต้องใช้วิศวกร และเวลาในดำเนินการมาก ฉะนั้นทาง DoD จึงจัดตั้งโครงการขึ้นมาเพื่อศึกษาวิธีการที่ช่วยในการพัฒนา วงจรอิเล็กทรอนิกส์ โดยเฉพาะอย่างยิ่งวงจรระบบดิจิตอล ให้สามารถนำไปผลิตได้เร็วขึ้น ซึ่งโครงการดังกล่าวมีชื่อว่า "Very High

Speed Integrated Circuits" หรือ VHSIC โดยในระยะแรกนั้นโครงการนี้ถือเป็นความลับทาง ด้านความมั่งคงของประเทศ และอยู่ภายใต้ความควบคุมดูแลของ United States International Traffic and Arms Regulations (ITAR) สำหรับมาตรฐานของภาษาที่ใช้บรรยาย พฤติกรรมวงจรหรือฮาร์ดแวร์ของระบบสำหรับโครงการ VHSIC ที่ DoDได้ให้ไว้สามารถสรุปได้ดังนี้

- ต้องเป็นภาษาที่นำไปเขียนรูปแบบระบบดิจิตอล และมีคุณสมบัติที่สามารถเข้าใจได้ทั้งมนุษย์และเครื่องคอมพิวเตอร์ โดยไม่ ต้องมีการแปลหรือเปลี่ยนแปลงอีก
- สามารถนำไปใช้เป็นเอกสารประกอบโครงการได้
- ต้องเป็นภาษาที่เขียนขึ้นสำหรับใช้จำลองการทำงานของวงจร

ฉะนั้นภาษาดังกล่าวนี้จึงจัดเป็นภาษาโปรแกรมระดับสูง เช่นเดียวกับภาษาปาสคาล หรือภาษาซี ซึ่งในทางวิศวกรรม ภาษาที่ใช้ในการออกแบบฮาร์ดแวร์นี้เรียกว่า "Hardware Description Language" หรือ HDL

ในตอนเริ่มแรกนั้น DoD ได้มอบหมายให้บริษัทไอบีเอ็ม เท็กซัสอินสตูเมนท์ และอินเตอร์เมทริกซ์ เป็นผู้ศึกษาและพัฒนาโครงการ ซึ่งการดำเนินงานเป็นไปอย่างต่อเนื่อง จนกระทั่งในปี ค.ศ.1985 ทาง ITAR ได้ยกเลิกข้อจำกัดในการถ่ายทอด เทคโนโลยีทางทหารออกจากโครงการนี้ ดังนั้นภาษา VHDL จึงเริ่มเป็นที่รู้จักกันโดยทั่วไป และประมาณปี ค.ศ. 1987 IEEE ได้ทำการกำหนดมาตรฐานของภาษานี้เป็น IEEE 1076-1987 และมีชื่อเรียกว่า VHDL ซึ่งมาตรฐานนี้ได้รับ การปรับปรุงจนเป็นมาตรฐาน IEEE 1076-1993 หรือ VHDL 1993 เนื่องจากในขณะนั้น DoD เป็นลูกค้ารายใหญ่ ของอุตสาหกรรมอิเล็กทรอนิกส์และคอมพิวเตอร์ ดังนั้นจึงมีผู้รับโครงการต่างๆ จาก DoD ไปดำเนินการวิจัยและพัฒนา เป็นจำนวนมาก และเพื่อให้ทุกโครงการอยู่ในมาตรฐานเดียวกันหมด ดังนั้นทาง DoD จึงได้กำหนดว่า ทุกๆ โครงการต้อง เขียนอยู่ในรูปของภาษา VHDLเท่านั้น ซึ่งทำให้ DoD สามารถนำโครงการเหล่านี้ไปจำลองกับเครื่องคอมพิวเตอร์ได้ หลายๆระบบ

ข้อกำหนด

DoD ได้ตั้งข้อกำหนดสำหรับภาษา VHDL ในเดือนมกราคมปี ค.ศ.1983 ไว้ดังนี้

1.ลักษณะทั่วไป

DoD ได้กำหนดให้ VHDL เป็นภาษาสำหรับการออกแบบและบรรยายของฮาร์ดแวร์ ซึ่งหมายถึงความสามารถ ในการอธิบายและออกแบบในระดับสูง การจำลอง (Simulation) การสังเคราะห์ (Synthesis) และการทดสอบ (Testing) นอกจากนั้น VHDLยังถูกกำหนดไว้สำหรับการบรรยายฮาร์ดแวร์ตั้งแต่ระดับบน ซึ่งก็คือระบบจนถึง ระดับเกทอีกด้วย เนื่องจากในการทำงานของระบบดิจิตอลนั้น ทุกๆ องค์ประกอบภายในระบบไม่ว่าเล็กหรือใหญ่จะทำงานไปพร้อมๆ กัน ซึ่งในเรื่องของความพร้อมเพรียงในการทำงานนี้ก็ถือเป็นข้อกำหนดที่สำคัญอย่างหนึ่งของ VHDL ด้วยเช่นกัน (สำหรับในภาษาที่ใช้ในการบรรยายฮาร์ดแวร์นั้นความพร้อมเพรียงจะหมายถึงทุกๆ คำสั่ง องค์ประกอบ เกทหรือวงจรลอจิกจะถูกนำมาปฏิบัติทั้งหมด ดังนั้นในที่สุดแล้วก็จะดูเหมือนว่าได้มีการปฏิบัติไป พร้อมๆ กัน)

2. สนับสนุนการออกแบบแบบลำดับขั้น

การออกแบบแบบลำดับขั้น เป็นลักษณะที่สำคัญอย่างหนึ่ง สำหรับการออกแบบระบบที่มีหลายๆ ระดับ โดยในการออกแบบจะประกอบด้วย ส่วนการบรรยายการเชื่อมต่อ และส่วนการบรรยายหน้าที่การทำงาน ซึ่งหน้าที่การทำงานของระบบสามารถกำหนดได้ด้วยตัวเอง หรืออาจถูกกำหนดโดยโครงสร้างที่ประกอบด้วยองค์ประกอบย่อยๆ ลงไปได้เช่นกัน แต่ที่ระดับล่างสุดองค์ประกอบต้องถูกบรรยายหน้าที่การทำงานด้วยตัวมันเอง และ ไม่สามารถกำหนดการทำงานโดยลักษณะแบบโครงสร้างได้

3. ไลบรารี

VHDL ได้สนับสนุนการมีไลบรารี เพื่อระบบการจัดการที่ดี ผู้ออกแบบสามารถกำหนดลักษณะและการทำงานของอุปกรณ์พื้นฐานไว้ในระบบไลบรารี หรือจะใช้ไลบรารีที่ระบบได้จัดเตรียมไว้แล้วก็ได้ โมเดลและการบรรยายที่ถูกต้องควรจัดเก็บไว้ในไลบรารี หลังจากที่ได้ผ่านการคอมไพล์เรียบร้อยแล้ว เพื่อให้ผู้ออกแบบคนอื่นๆ สามารถนำไป ใช้ได้ด้วย

4. ลำดับคำสั่ง

แม้ว่าการปฏิบัติคำสั่งหรือกระบวนการ โดยพร้อมเพรียงกันจะเป็นคุณสมบัติที่สำคัญของ VHDL ก็ตาม ตัวภาษาเองก็ยังมีการจัดเตรียมลักษณะการควบคุมแบบลำดับคำสั่งไว้ให้ด้วย เมื่อผู้ออกแบบได้กำหนดหน้าที่และองค์ประกอบที่ทำงานพร้อมกันของระบบไว้เรียบร้อยแล้ว ผู้ออกแบบยังสามารถบรรยายหน้าที่การทำงาน ซึ่งเป็นรายละเอียดภายในของแต่ละองค์ประกอบได้ในลักษณะเดียวกับการเขียนโปรแกรมที่ประกอบด้วย โครงสร้างแบบ case, if - then - else และ loop

ทั่วๆ ไปได้ การบรรยายแบบลำดับคำสั่งทำให้การออกแบบหน้าที่การทำงานของอุปกรณ์กระทำได้สะดวก และง่ายขึ้นอย่างไรก็ตามโครงสร้างทั้งหมดของ VHDL ก็ยังคงเป็นการทำงานแบบพร้อมเพรียงกันเช่นเดิม

5. การกำหนดคุณสมบัติ

นอกจากการกำหนดอินพุท และ เอาท์พุทแล้ว เงื่อนไขอื่นๆ ก็มีผลต่อการปฏิบัติหน้าที่ของอุปกรณ์ฮาร์ดแวร์ด้วยเช่นกัน โดยสิ่งนี้รวมถึงสภาพแวดล้อม และ ลักษณะทางกายภาพของอุปกรณ์นั้นๆ ด้วยซึ่งภาษาสำหรับการออกแบบที่ดีควรให้ผู้ออกแบบกำหนดคุณสมบัติของอุปกรณ์ที่ใช้ได้ด้วย เช่น สามารถกำหนดขนาด ลักษณะทางกายภาพเวลา โหลด และเงื่อนไขทางสภาพแวดล้อมอื่นๆซึ่งความสามารถในการกำหนดคุณสมบัตินี้ก็เป็นส่วนหนึ่งที่มีอยู่ในภาษา VHDL ด้วยเช่นกัน

6. ชนิดของข้อมูล

VHDL สามารถกำหนดชนิดของข้อมูลไม่เพียงแต่ชนิด BIT และ BOOLEAN เท่านั้น แต่ยังสามารถกำหนดชนิดของข้อมูลเป็นจำนวนเต็ม จำนวนจริง จุดทศนิยม และชนิดลำดับการนับ (Enumerate Type) หรือแม้แต่ชนิดของ ข้อมูลที่ผู้ออกแบบกำหนดขึ้นมาเองก็ได้

7. โปรแกรมย่อย

ความสามารถในการใช้ฟังก์ชันและโพรซีเจอร์ (Procedure) ก็เป็นข้อกำหนดอีกอย่างหนึ่งใน VHDL ซึ่งผู้ออกแบบสามารถนำโปรแกรมย่อยมาใช้ในการเปลี่ยนแปลงชนิดของข้อมูล การกำหนดหน่วยของลอจิก การกำหนดตัวกระทำต่างๆ หรือหน้าที่อื่นๆ ตามที่ต้องการได้เช่นเดียวกับการเขียนโปรแกรมทั่วไป

8. การควบคุมเวลา

VHDLอนุญาตให้ผู้ออกแบบสามารถกำหนดเวลาในการส่งผ่านข้อมูลหรือสัญญาณได้ตามต้องการ การตรวจสอบ การออกแบบเกทหรือการหน่วงเวลาก็สามารถกระทำได้โดยการกำหนดช่วงเวลาที่แน่นอน หรือกำหนดให้มีการรอคอย เหตุการณ์ (Event) นอกจากนี้ก็ยังสามารถกำหนดรูปแบบของสัญญาณนาฬิกาได้อีกด้วย

9. การกำหนดแบบโครงสร้าง

การกำหนดโครงสร้างขององค์ประกอบต่างๆ สามารถกระทำได้ในทุกระดับของการออกแบบ โดยการกำหนดโครง สร้างขององค์ประกอบร่วมที่เกิดจากองค์ประกอบย่อย ซึ่งแตกต่างกันหรือ เหมือนกันก็เป็นข้อกำหนดอย่างหนึ่งของ VHDL เช่นกัน

2.2 องค์ประกอบพื้นฐานของ VHDL

ในการเขียนรูปแบบบรรยายระบบดิจิตอลในมุมมองของการออกแบบลักษณะบนลงล่าง จะต้องทำความเข้าใจในเรื่องของโครงสร้างและส่วนประกอบต่างๆ ของรูปแบบภาษาวีเอชดีแอลเสียก่อน ซึ่งส่วนประกอบที่สำคัญและเป็นพื้นฐานของการเขียนมี 4 หน่วยคือ

- หน่วยการออกแบบเอนทิตี (Entity Design Unit)
- หน่วยการออกแบบสถาปัตยกรรม (Arcjotectire Design Unit)
- หน่วยการออกแบบแพ็กเก็ต (Paclage Design Unit)
- หน่วยการออกแบบโครงแบบ (Configuration Design Unit)

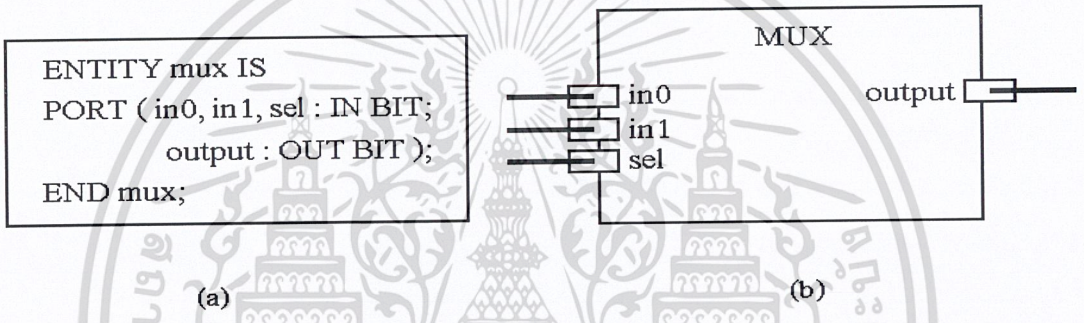
หน่วยการออกแบบเอนทิตี

หน่วยการออกแบบนี้เป็นส่วนที่ใช้สำหรับติดต่อระหว่างโลกภายนอกกับรูปแบบที่เขียนขึ้น ที่เรียกว่า หน่วยการออกแบบเอนทิตี ในส่วนนี้ใช้กำหนดจุดเชื่อมต่อ ของรูปแบบ กำหนดทิศทางการไหลของสัญญาณ และประเภทของค่าที่สามารถกำหนดให้กับสัญญาณตามจุดต่างๆ ของข้อมูลที่ไหล่ผ่านจุดต่อเหล่านั้น รูปที่ 2.3 แสดงให้เห็นโครงสร้างอย่างง่าย ๆ ของ หน่วยการออกแบบเอนทิตี

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

รูปที่ 2.3 แสดงโครงสร้างโดยทั่วไปของหน่วยการออกแบบเอนทิตี

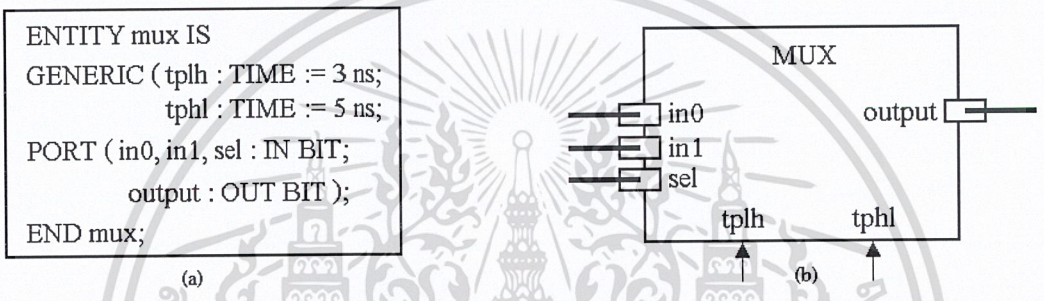
ส่วนนี้จะขึ้นต้นด้วยคำ ENTITY และ IS ระหว่างคำทั้งสองเป็นส่วนสำหรับชื่อของรูปแบบที่ต้องการจะเขียน (component_name) สำหรับการตั้งชื่อนั้นต้องเป็นไปตามกฎเกณฑ์ของภาษาหลังจากนั้นจะตามด้วยส่วนที่ใช้กำหนดช่องทางเข้าและออกของข้อมูล (input-output) รวมทั้งพารามิเตอร์อื่นๆ ส่วนนี้เรียกว่าส่วนหัว(entity header) และที่สำคัญคือ หน่วยการออกแบบเอนทิตีจะต้องปิดท้ายด้วยคำว่า END และเครื่องหมายอัฒภาคเสมอ (;)



รูปที่ 2.4 แสดงรูปแบบของมัลติเพลกซ์
(a) หน่วยการออกแบบเอนทิตีในรูปของวีเอชดีแอล
(b) มุมมองของตัวเชื่อมประสาน (Interfacing)

ในรูปที่ 2.4 เป็นหน่วยการออกแบบเอนทิตี ที่บรรยายอุปกรณ์ที่มีชื่อว่ามัลติเพลกซ์ หรือ MUX ในส่วนหัวของเอนทิตี มีการกำหนดจัดต่อ 4 จุดภายใต้ชุดคำสั่ง PORT โดยที่ 3 จุดแรกเป็นจุดให้ข้อมูลไหลผ่านเข้า ได้แก่ in0.in1.se1 ซึ่งกำหนดด้วยทิศทางการติดต่อกับภายนอกเป็นการไหลเข้าของข้อมูล (IN) ที่แสดงด้วยรูปสี่เหลี่ยมโปร่งในรูปที่ 2.4 ส่วนจุดเอาต์พุตเป็นจุดให้ข้อมูลไหลออก ซึ่งกำหนดด้วยทิศทางการติดต่อกับภายนอกเป็นการไหลออก (OUT) ที่แสดงด้วยรูปสี่เหลี่ยมทึบในรูปที่ 2.4 ส่วนประกอบประเภทของข้อมูลที่จะไหล เข้าและออก นั้นเป็นประเภท BIT ที่สามารถมีค่าได้เพียงสองค่าคือ “0” และ “1” เท่านั้น

นอกจากนั้นผู้ออกแบบยังสามารถกำหนดค่าพารามิเตอร์ทางฟิสิกส์ที่เป็นข้อมูลเพิ่มเติมอื่นๆ ลงในส่วนหัวของเอนทิตีได้อีก เช่น ข้อมูลเกี่ยวกับความเร็วในการทำงานของอุปกรณ์ อันได้แก่ ค่าเวลาหน่วงแพร่กระจาย (Propagation delay time) พารามิเตอร์เหล่านี้ เรียกว่า เจนเนริก (Generic) ที่กำหนดด้วยคำสั่ง GENERIC จากตัวอย่างในรูปที่ 2.5



รูปที่ 2.5 รูปแบบมัลติเพลกซ์ที่ประกอบด้วยข้อมูลค่าเวลาหน่วงแพร่กระจาย
(a) หน่วยการออกแบบเอนทิตีในรูปของวีเอชดีแอล
(b) มุมมองของตัวเชื่อมประสาน

ในบางกรณีสามารถที่จะใช้ภาษาวีเอชดีแอล สร้างรูปแบบที่ปราศจากช่องทางไหล เข้าและออกของข้อมูล ได้ ซึ่งส่วนใหญ่จะพบในการสร้างรูปแบบ สำหรับตรวจสอบการทำงานของอีกรูปแบบหนึ่ง คือ วีเอชดีแอลสำหรับการทดสอบเปรียบเทียบ (Test bench)

```
ENTITY test_bench IS
END test bench;
```

รูปที่ 2.6 หน่วยการออกแบบเอนทิตีที่ไม่มีการกำหนดช่องทางที่ต่อกับภายนอก

หน่วยการออกแบบสถาปัตยกรรม

คือส่วนที่ใช้เขียนบรรยายพฤติกรรมของรูปแบบ ในมุมมองของการจำลองการทำงานพฤติกรรมต่างๆ ที่บรรยายในส่วนนี้ขึ้นอยู่กับข้อมูลที่ผ่านเข้าและออก ตรงช่องทางตลอดจนพารามิเตอร์ต่างๆ ที่กำหนดใน หน่วยการออกแบบเอนทิตี รูปที่ 2.7 แสดงให้เห็นถึงโครงสร้างอย่างง่ายๆ ของหน่วยการออกแบบสถาปัตยกรรม

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

รูปที่ 2.7 หน่วยการออกแบบเอนทิตีที่ไม่มีการกำหนดช่องทางที่ต่อกับภายนอก

ส่วนของหน่วยการออกแบบสถาปัตยกรรมเริ่มต้นด้วยคำ ARCHITECTURE และตามด้วยชื่อ (identifier) สิ่งที่ต้องกำหนดลงไปได้แก่ สิ่งที่แสดงให้เห็นว่า ARCHITECTURE นั้นใช้ บรรยายหน่วยการออกแบบเอนทิตีใดๆ (OT<entity design unit> IS) ส่วนที่อยู่ระหว่าง ARCHITECTURE และ BEGIN เป็นพื้นที่ส่วนประกาศหน่วยของสถาปัตยกรรมกำหนด (architecture declarative area) ที่เป็นเพียงส่วนเผื่อเลือก (option) ในบริเวณนี้สามารถใช้เขียนประกาศกำหนดค่าต่างๆ ที่จะนำไปใช้ภายในสถาปัตยกรรมนั้นได้ อาทิเช่นประเภท (type) ต่างๆ ตัวอย่างเช่น bit, bit_vector), สัญญาณ (signal), ตัวคงที่ (constant) โปรแกรมย่อย ได้แก่ function และ procedure) และอุปกรณ์ (component) ส่วนที่ใช้บรรยายความสัมพันธ์ระหว่างข้อมูลที่ไหลเข้า และไหลออกของรูปแบบ (สัญญาณที่กำหนดในชุด

คำสั่ง PORT) นั้นจะถูกบรรยายในบริเวณเนื้อที่ระหว่างคำว่า BEGIN กับ END ของหน่วยการออกแบบสถาปัตยกรรม และนอกจากนั้นชุดคำสั่งทุกคำสั่งที่อยู่ภายในบริเวณนี้จะเป็นชุดคำสั่งแบบแข่งขนาน (concurrent statement) เท่านั้น หน่วยการออกแบบสถาปัตยกรรมจะต้องปิดท้ายด้วยคำสั่ง END และชื่อของสถาปัตยกรรมนั้นๆ เป็นส่วนเผื่อเลือกโดยทั่วไปการเขียนรูปแบบระบบดิจิตอลด้วยภาษาวีเอชดีแอลสามารถเขียนได้ในลักษณะต่างๆ ดังนี้

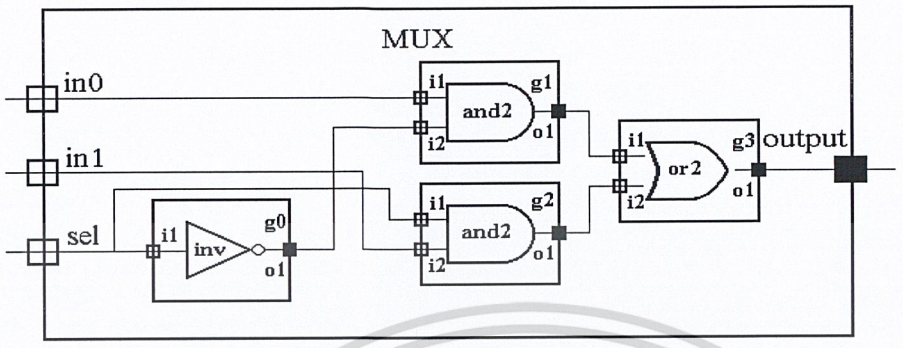
- ประเภทการไหลของข้อมูล (Dataflow description)
- ประเภทพฤติกรรม (Behavioral description)
- ประเภทโครงสร้าง (Structure description)
- ประเภทผสม (Mixed model description)

```
ARCHITECTURE data_flow OF mux IS
BEGIN
Output <= ((NOT se1) AND in0) OR (se1 AND in1);
END data_flow;
```

รูปที่ 2.8 แสดงหน่วยการออกแบบสถาปัตยกรรมของมัลติเพลกซ์ตามฟังก์ชันบูลีน

Output = (se1.in0) + (se1.in1)

จากรูปที่ 2.8 ส่วนที่บรรยายความสัมพันธ์ระหว่างข้อมูลที่ไหลเข้า (*in0, in1*) กับข้อมูลที่ไหลออก (output) ประกอบด้วยชุดคำสั่งแบบแข่งขันขนานเพียงชุดเดียว ซึ่งเขียนเป็นประเภทการไหลของข้อมูลของมัลติเพลกซ์ หรือ ระดับการถ่ายโอนข้อมูลระหว่างรีจิสเตอร์ (RTL:Register Transfer Level)



รูปที่ 2.9 แสดงโครงสร้างภายในสถาปัตยกรรมของมัลติเพลกซ์

จากรูปที่ 2.9 เป็นหน่วยการออกแบบสถาปัตยกรรมของมัลติเพลกซ์ประเภทโครงสร้าง โดยใช้ อินเวอร์เตอร์ (inv ที่ตำแหน่ง g0), แอนด์เกต 2 อินพุตจำนวน 2 ตัว (and2 ที่ตำแหน่ง g1 และ g2) และ ออร์เกต 2 อินพุต (or2 ที่ตำแหน่ง g3) มาสร้างตามฟังก์ชันบูลีนของรูปที่ 2.8

```
ARCHITECTURE struc OF mux IS
COMPONENT inv
PORT (i1 : IN  BIT ; o1 : OUT  BIT);
COMPONENT and2
PORT (i1, i2 : IN BIT ; o3 : OUT  BIT);
COMPONENT or2
PORT (i1, i2: IN BIT; o1: OUT BIT);
END COMPONENT;
SIGNAL int0, int1, int2: BIT:
BEGIN
g0 : inv        PORT MAP (i1 => sel, o1 => int0);
a1 : and2       PORT MAP (i1 => in0. i2 => int0. o1  =>
```

รูปที่ 2.10 หน่วยการออกแบบสถาปัตยกรรมของมัลติเพลกซ์ประเภทโครงสร้าง

```
ARCHITECTURE behav  OF mux  IS
BEGIN
PROCESS (in0, in1, sel)
BEGIN
        IF (se1 = '0") THEN  output <= in0:
        ELSE  output <= in1;
        END  IF;
END   PROCESS
END  behav;
```

รูปที่ 2.11 หน่วยการออกแบบสถาปัตยกรรมของมัลติเพลกซ์ประเภทพฤติกรรม

ไม่ว่าเขียนบรรยายส่วนของสถาปัตยกรรมของมัลติเพลกซ์ในลักษณะของ ประเภทพฤติกรรม ประเภทการไหลของข้อมูล ประเภทโครงสร้างหรือประเภทผสมที่นำเอาแต่ละประเภทมาเขียนไว้ในส่วนของสถาปัตยกรรม ก็ตามต่างก็มีพฤติกรรมเดียวกันและจะให้ผลลัพธ์จากการจำลองการทำงานที่เหมือนกัน ซึ่งนี้ก็เป็นข้อดีของภาษาวีเอชดีแอล

รูปแบบพื้นฐานที่ใช้ในการบรรยายถึงองค์ประกอบของ VHDL จะประกอบไปด้วยส่วนกำหนดการเชื่อมต่อ (Interface) และส่วนกำหนดลักษณะเชิงสถาปัตยกรรม (Architecture) ดังแสดงในรูปที่ 2.12 โดยในการบรรยายการเชื่อมต่อจะขึ้น ต้นด้วยคำว่า ENTITY แล้วตามด้วยชื่อขององค์ประกอบจากนั้นตามด้วยคำว่า IS และถัดมาจะเป็นการบรรยายถึงพอร์ต การติดต่อ อินพุท - เอาท์พุท ขององค์ประกอบ ส่วนลักษณะภายนอกอื่น ๆ เช่น เวลา อุณหภูมิก็สามารถรวมเข้าไปในส่วนนี้ ได้เช่นกัน ในส่วนของการกำหนดลักษณะเชิงสถาปัตยกรรมจะขึ้นต้นด้วยคำว่า ARCHITECTURE ซึ่งเป็นส่วนที่ใช้ บรรยายหน้าที่การทำงานขององค์ประกอบ โดยหน้าที่การทำงานนี้จะขึ้นอยู่กับสัญญาณอินพุท เอาท์พุทและพารามิเตอร์ อื่นๆ ที่ได้กำหนดไว้ในส่วนของการเชื่อมต่อดังรูปที่ 2.12 และสำหรับการบรรยายหน้าที่ขององค์ประกอบจะเริ่มต้นหลังจาก คำว่า BEGIN เป็นต้นไป

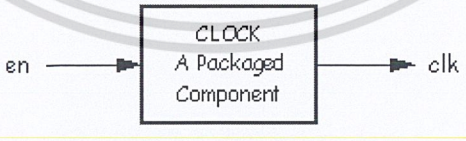
```
ENTITY component_name IS
    Input and output ports
    Physical and other parameters
END [component_name] ;

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

รูปที่ 2.12 การกำหนดการเชื่อมต่อและสถาปัตยกรรม

การกำหนดการเชื่อมต่อ

การกำหนดการเชื่อมต่อเป็นระดับบนสุดของการออกแบบโดยในระดับนี้ต้องกำหนดพอร์ตสำหรับการติดต่อกับองค์ประกอบภายนอกอื่นๆ ดังตัวอย่างในรูปที่ 2.13 ซึ่งเป็นบล็อคไดอะแกรมและการบรรยายการเชื่อมต่อขององค์ประกอบ สำหรับตัวจ่ายสัญญาณนาฬิกาในบรรทัดแรกของการบรรยายการเชื่อมต่อเป็นการกำหนดชื่อขององค์ประกอบ ซึ่งกำหนดเป็น clock_ component ตามด้วยคำว่า PORT และชื่อของพอร์ตอยู่ภายในวงเล็บ ส่วน IN และ OUT เป็นการกำหนดโหนดของสัญญาณให้เป็นอินพุทหรือเอาท์พุท และ BIT เป็นการแสดงชนิดของข้อมูล




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

รูปที่ 2.13 บล็อคไดอะแกรมและการบรรยายการเชื่อมต่อของ clock_ component

การกำหนดรูปแบบการบรรยาย

หน้าที่การทำงานขององค์ประกอบจะถูกบรรยายภายในส่วนนี้ ซึ่งในการบรรยายสามารถกำหนดค่าของสัญญาณเอาท์พุทในเทอมของอินพุทหรือในรูปขององค์ประกอบอื่นๆ หรือทั้งสองอย่างรวมกันก็ได้ดังตัวอย่างการบรรยายของ Clock_component ในรูปที่ 2.14 ซึ่งเป็นการบรรยายในเชิงพฤติกรรมโดยมี en เป็นอินพุทและ ck เป็นเอาท์พุท PROCESS เป็นคำที่ใช้ในการเริ่มต้นสำหรับการบรรยายในเชิงพฤติกรรม และภายในโปรเซสกำหนดให้ periodic เป็นตัวแปรที่มีคำเริ่มต้นเป็น "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;
ENDbehavioral;
```

รูปที่ 2.14 การบรรยายเชิงพฤติกรรมของ clock _ component

หน่วยการออกแบบแพ็กเก็ต

ข้อมูลต่างๆ ตลอดจนโปรแกรมย่อยที่เป็นประโยชน์ต่อการเขียนรูปแบบ การบรรยายระบบดิจิตอล สามารถเก็บไว้ใน ส่วนของแพ็กเก็ต ซึ่งหน่วยการออกแบบต่างๆ เช่น หน่วยการออกแบบ Entity หน่วยการออกแบบสถาปัตยกรรมหรือ หน่วยการออกแบบแพ็กเกจอื่นๆ สามารถเรียกข้อมูลเหล่านี้ไปใช้ได้ นอกจากนั้นสิ่งที่นิยมทำกันมากคือการนำรูปแบบ มาตรฐานต่างๆ เช่น อุปกรณ์มาตรฐาน (เช่น ไอซีตระกูล 74XX เป็นต้น)มาเก็บไว้ในรูปของแพ็กเกจ ที่ทุกคนสามารถ เข้าถึงได้ ตามปกติแล้วแพ็กเกจจะแบ่งออกเป็น 2 ส่วนคือ การประกาศแพ็กเกจ (Package declaration)และส่วนของบอดี้แพ็กเกจ (Package body) เนื่องจาก แพ็กเกจถูกสร้างขึ้นเป็นส่วนแยกต่างหากออกจากรูป

แบบที่กำลังเขียนอยู่ ฉะนั้นการที่นำแพ็กเกจไปใช้นั้นจะต้องมีการเชื่อมโยงหรืออ้างอิงเสียก่อน ซึ่งในภาษา VHDL สามารถ กระทำได้ด้วยชุดคำสั่ง USE

- PACKAGE DECLARATION

ส่วนที่มีความสำคัญที่สุดของแพ็กเกจ (ถ้ามองในแง่ของการนำไปใช้จากภายนอก) ได้แก่ ส่วนการประกาศแพ็กเกจ เนื่องจากเป็นส่วนที่ใช้กำหนดชื่อของสิ่งที่ประกาศอยู่ภายในแพ็กเกจสำหรับนำไปใช้ภายนอกตัวของแพ็กเกจเอง ถ้ามี การประกาศสิ่งใดๆ ในส่วนของส่วนบอดี้แพ็กเกจแต่ไม่ถูกประกาศในส่วนการประกาศแพ็กเกจจะทำให้ค่าและพฤติกรรมไม่สามารถนำไปใช้งานในส่วนนอกได้ ซึ่งเปรียบเทียบได้กับสิ่งที่ประกาศไว้ในส่วนของการประกาศ Entity คือ จุดเชื่อมต่อหรือ พอร์ต ที่มีหน้าที่ติดต่อกับโลกภายนอก ฉะนั้นโดยทั่วไปแล้วแพ็กเกจสามารถสร้างขึ้นได้โดยไม่จำเป็นต้องมีส่วนบอดี้ และยังสามารถนำไปใช้งานจากรูปแบบภายนอกได้เช่น ใช้สำหรับประกาศ ชนิด (Type) หรือสัญญาณ เช่นเดียวกับ ส่วนบอดี้แพ็กเกจที่ไม่จำเป็นต้องมี ส่วนของการประกาศแพ็กเกจแต่แพ็กเกจนั้นจะไม่สามารถนำไปใช้จาก รูปแบบอื่นได้

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

รูปที่ 2.15 โครงสร้างทั่วไปของส่วนการประกาศแพ็กเกจ

- PACKAGE BODY

โครงสร้างซึ่งประกอบด้วยลำดับคำสั่งที่ใช้บรรยายฟังก์ชันการทำงานของโปรแกรมย่อยทั้งหลาย ซึ่งชื่อของโปรแกรมย่อยนั้นๆ ได้ถูกประกาศไปแล้ว ในส่วนของการประกาศแพ็กเกจ จะถูกเก็บไว้ในส่วนของบอดี้แพ็กเกจ ทั้งนี้รวมถึง การกำหนดค่าคงที่ต่างๆ อันได้แก่ค่าคงที่ที่ถูกประกาศชื่อไว้ก่อน ในส่วนของการประกาศแพ็กเกจ และถูกกำหนดค่าใน ส่วนของบอดี้แพ็กเกจ ฉะนั้นในส่วนของบอดี้แพ็กเกจจึงไม่จำเป็นต้องมี ถ้าในส่วนของการประกาศแพ็กเกจไม่มีการ ประกาศชื่อที่เป็นโปรแกรมย่อย หรือค่าคงที่ การเขียนบอดี้แพ็กเกจนั้นจะเป็นไปตามกฎเกณฑ์ดังแสดงในรูปที่ 2.16

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

รูปที่ 2.16 โครงสร้างของบอดี้แพ็กเกจ

หน่วยการออกแบบ Configuration

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

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

รูปที่ 2.17 โครงสร้างโดยทั่วไปของหน่วยการออกแบบโครงแบบ

โปรแกรมย่อย

การใช้ฟังก์ชันและโพรซีเจอร์ใน VHDL เปรียบได้กับการใช้โปรแกรมย่อยในการเขียนโปรแกรมภาษาชั้นสูงทั่วๆ ไปค่าที่ถูกส่งกลับ หรือถูกเปลี่ยนแปลงโดยโปรแกรมย่อยอาจจะมีหรือไม่มีผลต่อฮาร์ดแวร์โดยตรงก็ได้ เช่นถ้าใช้ฟังก์ชัน แทนการกระทำในสมการบูลีนก็จะมีผลต่อวงจรลอจิกจริงๆ ในขณะที่ถ้าใช้โปรแกรมย่อยในการเปลี่ยนชนิดของข้อมูล หรือในการคำนวณค่าการหน่วงเวลาแล้วก็จะไม่มีผลต่อโครงสร้างของฮาร์ดแวร์ รูปที่ 2.18 แสดงการใช้โพรซีเจอร์ เพื่อเปลี่ยนข้อมูลชนิด 8 บิตเป็นค่าจำนวนเต็ม และรูปที่ 2.19 แสดงการใช้ฟังก์ชันโดยกำหนดให้ X เป็นตัวแปรชนิด บิตแทนการกระทำในสมการบูลีน

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

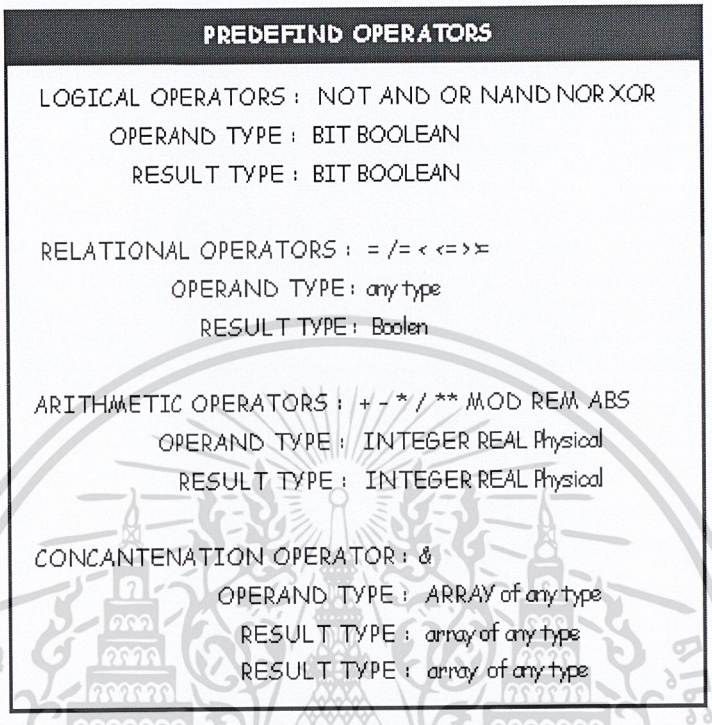
รูปที่ 2.18 การใช้โพรซีเจอร์

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

รูปที่ 2.19 การใช้ฟังก์ชัน

โอเปอร์เรเตอร์

การบรรยายเชิงพฤติกรรมในภาษาVHDLมีตัวดำเนินการหรือโอเปอร์เรเตอร์ทางลอจิกและคณิตศาสตร์เช่นเดียว กับภาษาซอฟต์แวร์ทั่วไปดังรูปที่ 2.20



รูปที่ 2.20 ตัวดำเนินการใน VHDL

เวลาและความพร้อมเพรียง

ในวงจรอิเล็กทรอนิกส์อุปกรณ์ทุกๆ ตัวจะอยู่ในสภาพเตรียมพร้อมเสมอ (Always Active) และจะมีเรื่องของเวลาเข้ามาเกี่ยวข้องในทุกๆเหตุการณ์ที่เกิดขึ้นเสมอ VHDL เป็นภาษาที่ได้รับการออกแบบมาเพื่อให้สามารถบรรยายรูปแบบและการพ้องกันของเวลาสำหรับการทำงานของอุปกรณ์ได้อย่างถูกต้อง การบรรยายการทำงานที่อยู่ภายในส่วน ของการบรรยายสถาปัตยกรรม จะมีการทำงานที่พร้อมเพรียงกันเสมอ หรือแม้แต่โปรเซสซึ่งมีการทำงานภายในเป็น แบบลำดับคำสั่งก็ตาม ซึ่งหากมีหลายๆ โปรเซสอยู่ภายในโครงสร้างเดียวกัน ทุกๆ โปรเซสก็จะทำงานไปพร้อมๆ กัน ด้วย

สัญญาณและตัวแปร

สัญญาณมีลักษณะเป็นเสมือนตัวกลางฮาร์ดแวร์ที่ใช้ในการส่งผ่านข้อมูล และ มีเรื่องของเวลาเข้ามาเกี่ยวข้องด้วยการ กำหนดค่าให้กับสัญญาณจะใช้สัญลักษณ์ <= ในการส่งค่าและสามารถใช้คำสั่ง AFTER เพื่อกำหนดช่วงเวลาในการ ส่งผ่านค่าของสัญญาณ เช่น w <= a AFTER 12 NS หมาย

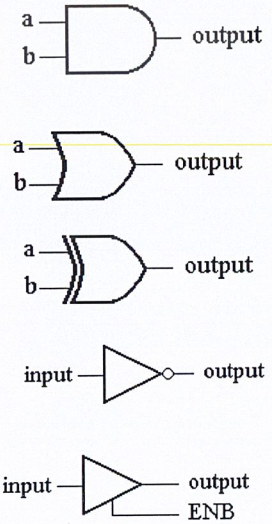
ถึงการกำหนดค่าสัญญาณ a ให้กับ w หลังจากเวลา ผ่านไป 12 นาโนวินาที ในทางตรงข้ามตัวแปรมีลักษณะเป็นเสมือนตัวกลางที่ใช้ในการส่งผ่านข้อมูลและไม่มีเรื่องของ เวลาเข้ามาเกี่ยวข้องด้วย ซึ่งตัวแปรจะถูกใช้ในส่วนที่มีการทำงานเป็นแบบลำดับคำสั่งเช่นใน ฟังก์ชัน โพรซีเจอร์ และ โปรเซส สำหรับการกำหนดค่าให้กับตัวแปรจะใช้สัญลักษณ์: =

2.3 ภาษาวีเอชดีแอลเพื่อการสังเคราะห์

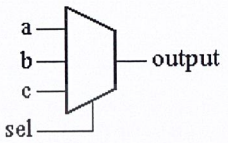
ภาษาวีเอชดีแอล เป็นภาษาที่เขียนเพื่อจำลองการทำงานของวงจร ซึ่งในบางรูปแบบการเขียนไม่สามารถที่จะนำไปสังเคราะห์ได้ทั้งหมด ดังนั้นถ้าต้องการเขียนเพื่อนำไปสังเคราะห์ ควรหลีกเลี่ยงรูปแบบต่างๆ ที่ไม่สามารถนำไปสังเคราะห์ได้ ในที่นี้ขึ้นอยู่กับความสามารถของโปรแกรมที่ใช้สังเคราะห์แต่ละโปรแกรม ดังนั้นในหัวข้อนี้จะแสดงตัวอย่างของการเขียนโมเดลในรูปแบบต่างๆที่สามารถนำไปสังเคราะห์ ซึ่งยึดหลักการเขียนตาม ViewSynthesis User's Guide ของโปรแกรมซึ่งเป็นโปรแกรม Viewlogic ที่ใช้ในสังเคราะห์วงจรทั้งหมดในการออกแบบไมโครคอนโทรลเลอร์โดยแบ่งออกเป็น 2 กลุ่ม คือ เกตและฟลิปฟลอปประเภทต่างๆ

2.3.1 ตัวอย่างรูปแบบการเขียนเกตพื้นฐาน

```
SIGNAL a, b, c, d, input : vlbit_1d(3 DOWNTO 0);
SIGNAL se1 : vlbit_1d(1 DOWNTO 0);
SIGNAL enb : vlbit;
……..
        ouput <= a AND b;
……..
        ouput <= a OR b;
……..
        ouput <= a XOR b;
……..
        ouput <= a NOT b;
….
```



```
ouput  <= a input WHEN enb='1'  ELSE "ZZ";
....
WITH se1  SELECT
        output <=  a   WHEN  "00" ELSE
                   b  WHEN  "01" ELSE
                   c;
```



2.3.2 ตัวอย่างรูปแบบการเขียนฟลิปฟลอปพื้นฐาน

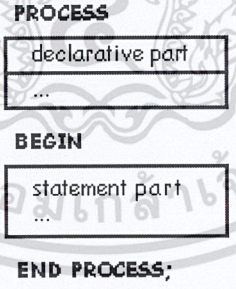
```
SIGNAL  input,  output  :  vlbit_ld (3  DOWNTO 0);
SIGNAL  clock,  enable,  reset  :  vlbit ;
        ....
      PROCESS  BEGIN
            WAIT  UNTIL  PRISING (clock);
                  Output <= input;
      END PROCESS;
      ....
      PROCESS  BEGIN
      WAIT  UNTIL  PRISING (clock)  OR  (reset='1');
        IF (reset='1')  THEN  output <='00';
        ELSIF  (enble = '1' ) THEN  ouput <= input ;
        END  IF ;
      END PROCESS;
```

2.5 การบรรยายเชิงพฤติกรรม

การบรรยายลักษณะการทำงานของอุปกรณ์ฮาร์ดแวร์ในเชิงพฤติกรรม เป็นการบรรยายลักษณะการเปลี่ยนแปลงของ ข้อมูลในรูปแบบของอัลกอริธึม สำหรับการคำนวณผลลัพธ์ที่เกิดขึ้น ซึ่งสืบเนื่องมาจากการเปลี่ยนแปลงสภาวะของข้อมูล ที่เข้ามาโดยไม่คำนึงถึงลักษณะโครงสร้างหรือความสัมพันธ์ของอุปกรณ์ที่อยู่ภายในว่าจะเป็นอย่างไรในหัวข้อนี้จะ แสดงถึงการบรรยายเชิงพฤติกรรม แทนการใช้โมดูลฮาร์ดแวร์รวมถึงข้อกำหนดต่างๆ ที่ควรรู้

โปรเซส

โปรเซสเป็นรูปแบบพื้นฐานอย่างหนึ่งที่ใช้ในการกำหนดให้กับสัญญาณโปรเซสจะอยู่ในสถานะที่เตรียมพร้อมอยู่เสมอ และจะปฏิบัติคำสั่งพร้อมๆกันกับโปรเซสอื่นๆ ที่อยู่ในสถาปัตตกรรมบรรยายเดียวกัน โดยโปรเซสจะปฏิบัติงานตามคำ สั่งทันทีที่มีเหตุการณ์เกิดขึ้นกับญญาณที่อยู่ทางด้านขวามือของสัญลักษณ์กำหนดค่าให้กับสัญญาณ (<=) การบรรยาย โปรเซสจะเริ่มต้นด้วยคำสั่ง PROCESS และจบด้วยคำสั่ง END PROCESS ในรูปที่ 2.21 เป็นการแสดงส่วน ประกอบของการบรรยายแบบโปรเซส ซึ่งประกอบด้วยส่วนของการประกาศตัวแปรที่ต้องใช้และส่วนของการปฏิบัติ คำสั่งเพื่อให้ได้ผลลัพธ์ที่ต้องการ



รูปที่ 2.21รูปแบบของการบรรยายแบบโปรเซส

การกำหนดตัวดำเนินการภายในโปรเซส

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

(Signal) หรือตัวคงที่ที่ได้ประกาศไว้ในส่วนของ ARCHITECTURE ในรูปที่ 5.13 แสดงตัวอย่างการประกาศตัวกระทำภายในโปรเซส ซึ่งจะอยู่ระหว่างคำสั่ง PROCESS และ BEGIN และค่าเริ่มต้นที่ถูกกำหนดให้กับตัวดำเนินการภายในโปรเซสจะถูกนำมาใช้ในตอนเริ่มต้น ของการปฏิบัติเพียงครั้งเดียวเท่านั้น ต่างกับค่าเริ่มต้นที่อยู่ภายในโปรแกรมย่อยจะถูกนำมาใช้ทุกครั้งที่มีการเรียกใช้ โปรแกรมย่อยนั้น ๆ

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

รูปที่ 2.22 ตัวอย่างการประกาศตัวดำเนินการภายในโปรเซส

การกำหนดการกระทำภายในโปรเซส

การกระทำใดๆ ภายในโปรเซสจะเป็นการปฏิบัติแบบลำดับ (Sequential) เสมอ ซึ่งภายในโปรเซสสามารถใช้ประโยค เงื่อนไขหรือการทำซ้ำได้เช่น IF-THEN - ELSE, CASE - WHEN, FOR LOOP และ WHILE LOOP ดังตัวอย่างในรูปที่ 2.23 และ 2.24

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

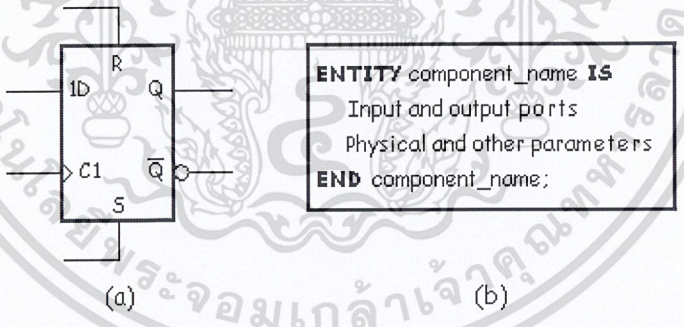
รูปที่ 2.23 เงื่อนไขการกระทำในโปรเซส

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

รูปที่ 2.24 แสดงการกระทำในโปรเซส

การกระตุ้นและยับยั้งการกระทำของโปรเซส

การกระทำภายในโปรเซสจะอยู่ในสภาพเตรียมพร้อม และมีการปฏิบัติงานอยู่ตลอดเวลาที่มีการเปลี่ยนแปลงของเหตุการณ์ เกิดขึ้น อย่างไรก็ดีเราสามารถกระตุ้นหรือยับยั้งการกระทำภายในโปรเซสได้โดยการกำหนดรายการของสัญญาณที่ต้อง การให้โปรเซสปฏิบัติงานเมื่อมีเหตุการณ์เกิดขึ้นกับสัญญาณที่กำหนดไว้เท่านั้น ส่วนเหตุการณ์ใดๆ ที่เกิดขึ้นกับสัญญาณ ที่ไม่ได้กำหนดไว้ในรายการก็จะไม่ส่งผลให้มีการกระทำภายในโปรเซส ซึ่งรายการของสัญญาณนี้เรียกว่า Sensitivity List และจะกำหนดไว้ภายในวงเล็บหลังคำสั่ง PROCESS รูปที่ 2.25 (a) แสดงตัวอย่างโมเดล และรูปที่ 2.25 (b) เป็นตัวอย่างการบรรยายการเชื่อมต่อของ D-Flip Flop ส่วนรูปที่ 2.26 แสดงถึงการบรรยายเชิงพฤติกรรมของ D-Flip Flop โดยในรูปที่ 2.26 (a) เป็นการใช้ตัวกระทำภายนอกโปรเซส และรูปที่ 2.26 (b) เป็นการใช้ตัวกระทำภายในโปรเซส โดยมีรายการของสัญญาณ (rst, set, clk) เป็นตัวกระตุ้นการปฏิบัติงานภายในโปรเซส



รูปที่ 2.25 (a) ตัวอย่างโมเดล D-Flip Flop

(b) การบรรยายการเชื่อมต่อของ D-Flip Flop

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

(a)

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

(b)

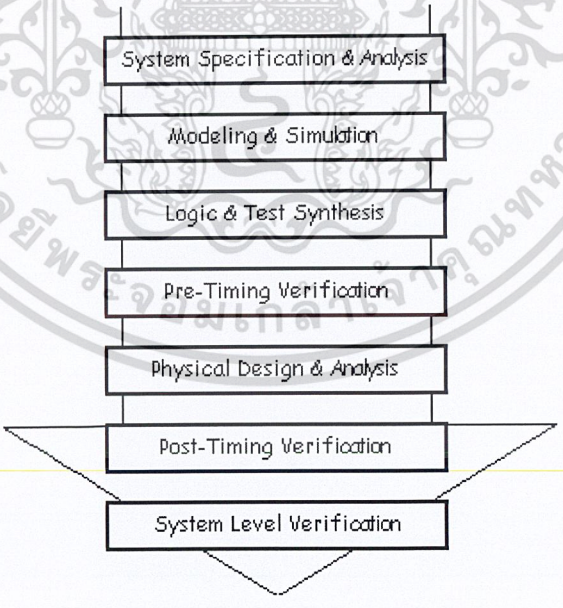
รูปที่ 2.26 การบรรยายเชิงพฤติกรรมของ D-FlipFlop

(a) การใช้ตัวกระทำภายนอกโปรเซส

(b) การใช้ตัวกระทำภายในโปรเซส

2.6 การออกแบบจากบนลงล่าง

ในการพัฒนาวงจรรวมดิจิตอลขนาดใหญ่ที่มีความซับซ้อน วิศวกรหรือผู้ออกแบบมักจะมองการออกแบบให้อยู่ในรูปของ บล็อกไดอะแกรมก่อนที่ทำวิเคราะห์ให้ลึกถึงรายละเอียดต่อไป ซึ่งภาษา VHDL นั้นอนุญาตให้อธิบาย และวิเคราะห์การทำงานของแต่ละบล็อก รวมถึงการปรับปรุงการทำงานจากผลที่วิเคราะห์เพื่อให้ได้การทำงานตามต้องการ นอกจากนี้ยังสามารถเพิ่มเติมในรายละเอียดในแต่ละขั้นตอนได้ ซึ่งหลักการนี้สอดคล้องกับหลักการออกแบบจากบนลงล่าง (Top - Down Design) นั่นเอง ถ้าทดลองเปรียบเทียบกับการออกแบบจากล่างขึ้นบน (Bottom - Up Design) จะเห็นได้ว่าการออกแบบจากล่างขึ้นบนจะใช้เวลาการออกแบบมากกว่า 90% เนื่องจากเป็นการวาดวงจรด้วยอุปกรณ์ต่างๆ (Schematic capture) ที่ประกอบกันเข้าเป็นวงจรที่ต้องการออกแบบก่อนแล้วจึงทำการจำลองการทำงาน และตรวจ สอบความถูกต้อง ดังนั้นการใช้ภาษา VHDL กับหลักการออกแบบจากบนลงล่างจึงเป็นทางออกให้กับวิศวกรให้สามารถออกแบบและพัฒนาวงจรที่มีความซับซ้อนได้มากขึ้นทั้งยังช่วยลดเวลาและค่าใช้จ่ายในการออกแบบด้วย



รูปที่ 2.27ขั้นตอนการออกแบบจากบนลงล่าง

จากรูปที่ 2.27 แสดงถึงขั้นตอนของการออกแบบจากบนลงล่าง ทั้งนี้ในทางปฏิบัติอาจมีข้อแตกต่างไปจากนี้บ้าง เล็กน้อยเนื่องจากขั้นตอนของการผลิต (Implementation) สามารถกระทำได้หลายเทคโนโลยี สำหรับรายละเอียดของขั้นตอน การออกแบบจากบนลงล่างในแต่ละขั้นตอนมีดังนี้

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

2.เขียนรูปแบบของระบบที่ต้องการออกแบบโดยใช้ภาษา VHDL หรือ ภาษา HDL อื่นๆ สำหรับบรรยายพฤติกรรมการ ทำงาน พร้อมทั้งจำลองการทำงาน เพื่อเปรียบเทียบและตรวจสอบความถูกต้องกับข้อกำหนด

3. หลังจากที่ได้หลักการขั้นต้นพร้อมแนวความคิดที่ผ่านการตรวจสอบแล้ว หลักการนี้จะถูกเพิ่มเติมในรายละเอียดลงมา เป็นลำดับขั้นที่สอง จนกระทั้งอยู่ในระดับที่จะนำไปผลิตวงจริง หรือสังเคราะห์ในขั้นตอนนี้เองเทคโนโลยีที่จะมารองรับ วงจรออกแบบจะถูกกำหนดขึ้น และระบบช่วยการออกแบบจะสังเคราะห์วงจรที่ได้จากรูปแบบที่เขียนขึ้นให้อยู่ในรูปของวงจรที่ประกอบด้วยอุปกรณ์อิเล็กทรอนิกส์ หรือวงจรในระดับเกท และการเชื่อมต่อระหว่างกันของอุปกรณ์เหล่านั้นหรือ ไม่ก็อยู่ในรูปของ Netlist ที่สามารถนำไปผลิตในอุปกรณ์อื่นได้

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

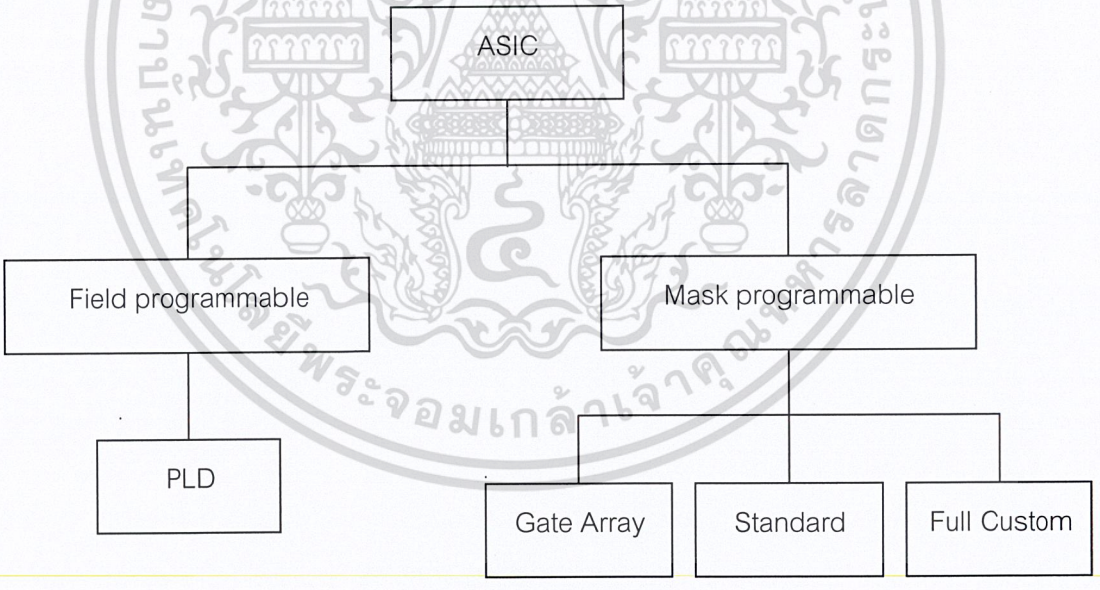
5. ผลิตเป็นวงจรจริง (Technology and device mapping) โดยนำข้อมูลที่ได้จากการสังเคราะห์มาผลิต ซึ่งอาจ จะอยู่ในรูปของแผงวงจรไฟฟ้า ที่ประกอบด้วยอุปกรณ์หลายๆ ชิ้น หรืออยู่ในรูปของวงจรรวม ASIC

6. ทำการตรวจสอบการทำงานและตัวแปรทางด้านเวลาทั้งหมด เพื่อความถูกต้องของวงจรเป็นครั้งสุดท้ายก่อนนำไปรวมเข้ากับอุปกรณ์อื่นๆ ให้เป็นระบบดิจิตอล เนื่องจากในขั้นตอนนี้ วงจรที่ออกแบบ จะประกอบด้วยจุดต่อทางอินพุทและเอาท์พุท ซึ่งเป็นจุดต่อสำหรับการรับและส่งสัญญาณกับภายนอก

7. นำวงจรที่ออกแบบไว้ประกอบเข้ากับอุปกรณ์อื่นๆ ให้เป็นระบบที่สมบูรณ์ แล้วทำการทดสอบการทำงานทั้งระบบร่วมกับอุปกรณ์อื่นๆ อีกครั้งเพื่อควบคุมคุณภาพของผลิตภัณฑ์

บทที่ 3
เอฟพีจีเอ

ความก้าวหน้าของอุตสาหกรรมอิเล็กทรอนิกส์ปัจจุบันทำให้เกิดการพัฒนาความสามารถของอุปกรณ์ต่างๆ มากมายซึ่งทำให้เกิดการลดค่าใช้จ่าย การสิ้นเปลืองพลังงานและขนาด ในขณะเดียวกันก็มีการเพิ่มประสิทธิภาพและระดับความเชื่อถือได้ของวงจรรวมที่สูงขึ้นเห็นได้ชัดจากเทคโนโลยีไมโครโพรเซสเซอร์และหน่วยความจำปัจจุบัน ทุกๆ ครั้งที่มีการพัฒนาขึ้นทำให้เกิดช่องว่างวงจรรวมและไอซีมาตรฐานมากขึ้น ในการพัฒนาเพิ่มความหนาแน่นและจำนวน ฟังก์ชั่นลอจิกที่เหมาะสม นักออกแบบอุปกรณ์ทางด้านดิจิตอลได้พิจารณาถึงการผลิตให้ขนาดมากๆ และการผลิตวงจรรวม (ASIC: Application Specific Integrated Circuit) ซึ่งวงจรรวมจะแบ่งตามการสร้างออกเป็น 2 กลุ่ม คือ Field programmable และ Mask programmable ดังแสดงในรูปที่ 3.1



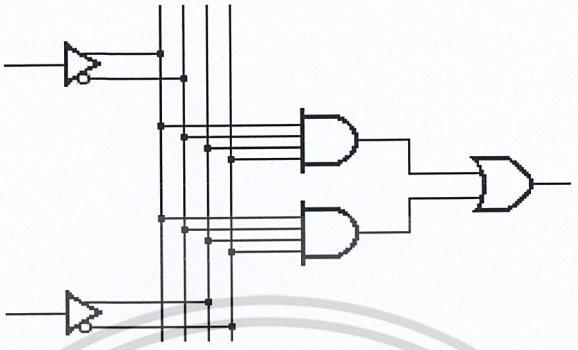
รูปที่ 3.1 ผังแสดงการแบ่งกลุ่มของวงจรรวม ASIC

3.1 Field Programmable

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

3.1.1 พีแอลดี (PLD: Programmable Logic Device)

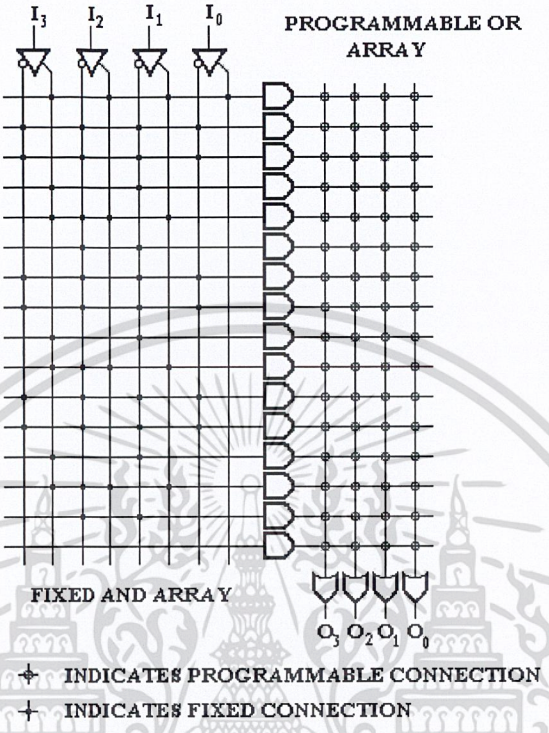
ภายในอุปกรณ์พีแอลดีถูกเตรียมเป็นวงจรพื้นฐานทางด้านลอจิกต่อกันอยู่เป็นกลุ่มมีทั้งวงจรคอมบิเนชั่น (Combination) และซีเควลเชียล (Sequential) ซึ่งมีส่วนประกอบเป็นวงจรภายในเทคโนโลยีของวงจรที่ใช้สร้างพีแอลดีมีทั้ง ทีทีแอล (TTL) อีซีแอล (ECL) และ ซีมอส (CMOS) ตามความเหมาะสมของแต่ละระบบ อุปกรณ์พีแอลดีทุกชนิดมีหลักการพื้นฐานของวงจรภายในที่เหมือนกันโดยมีวงจรหลักเป็นวงจรคอมบิเนชั่นที่ให้ผลเป็นผลคูณร่วมบวก (Sum of product) ประกอบไปด้วยชุดของแอนด์เกตที่ต่อร่วมกับออเกตการโปรแกรมคือ การเลือกว่าจะจะให้มีการต่ออินพุตภายในของแอนด์เกตกับสัญญาณอินพุตใดบ้างซึ่งมีทั้งจากภายนอกและสัญญาณป้อนกลับจากเอาต์พุตภายในเอง การติดต่ออินพุตของออเกตกับเอาต์พุตของ แอนด์เกต ตัวต่างๆ วิธีการเลือกหรือการโปรแกรมทางกายภาพ อินพุตต่างๆ ของอุปกรณ์ทุกตัวจะถูกต่อผ่านฟิวส์เข้ากับแหล่งสัญญาณ ซึ่งถ้าไม่ต้องการใช้สัญญาณใดจะตัดฟิวส์ทำให้สามารถโปรแกรมได้ครั้งเดียว อุปกรณ์พีแอลดีบางชนิดใช้มอสทรานซิสเตอร์แทนฟิวส์ทำให้สามารถโปรแกรมโดยใช้กระแสไฟฟ้า และสามารถลบและโปรแกรมใหม่เข้าไปได้อีก



รูปที่ 3.2 แสดงวงจรพื้นฐานของอุปกรณ์พีแอลดีซึ่งอยู่ในรูปผลคูณร่วมบวก

3.1.2 พรอม (PROM: Programmable Read Only Memory)

พรอมคือหน่วยความจำรอม (ROM) ที่โปรแกรมได้ ซึ่งนับว่าเป็นอุปกรณ์พีแอลดี ชนิดหนึ่ง ซึ่งวงจรภายในของพรอมเสมือนกับประกอบไปด้วยแถวลำดับของแอนด์เกตและออเกต (Amd-Or Array) ผลเอาต์พุตที่ขาต้าเอาต์พุตสามารถแสดงในสมการของฟังก์ชั่นผลคูณร่วมบวก (Sum of product) ของสัญญาณอินพุตที่ขาแอดเดรส รูปที่ 3.3 แสดงถึงลักษณะการต่อเป็นแถวลำดับของแอนด์เกตและออเกตของพรอมขนาด 16x4 บิต วงจรทางด้านซ้ายบนสุดเป็น แอนด์เกตที่ให้ผลเป็นผลคูณ (Product) ของกรณีอินพุตเป็น 0000 แอนด์เกตที่อยู่ถัดลงมาเป็นผลคูณของกรณีที่อินพุต เป็น 0001, 0010, ... จนถึงตัวล่างสุดคือผลคูณในกรณีที่อินพุตเป็น 1111 ที่อินพุตแต่ละบิตของหน่วยความจำสามารถเลือกได้ว่าจะให้เป็น 1 ในกรณีที่อินพุตจากแอดเดรส เป็นอย่างไรบ้างเหมือนกันเป็นการนำเอาต์พุตจากผลคูณที่ต้องการให้เอาต์พุตแต่ละบิตเป็น 1 ไปออกันจึงเปรียบเหมือนกับว่าในพรอมมีจำนวนแอนด์เกตเท่ากับจำนวนตำแหน่งความจำและมีออเกตจำนวนเท่ากับจำนวนบิตของสัญญาณข้อมูลออก (Data output) อินพุต่ของออเกตทุกตัวสามารถต่อเข้ากับแอนด์เกตตัวใดก็ได้ทุกตัว ซึ่งอาจเรียกได้ว่าเป็นพีแอลดีแบบ fixed AND/programmable OR



รูปที่ 3.3 แสดงลักษณะของพรอมเมื่อเปรียบเทียบเป็นวงจรในรูปผลคูณร่วมบวก

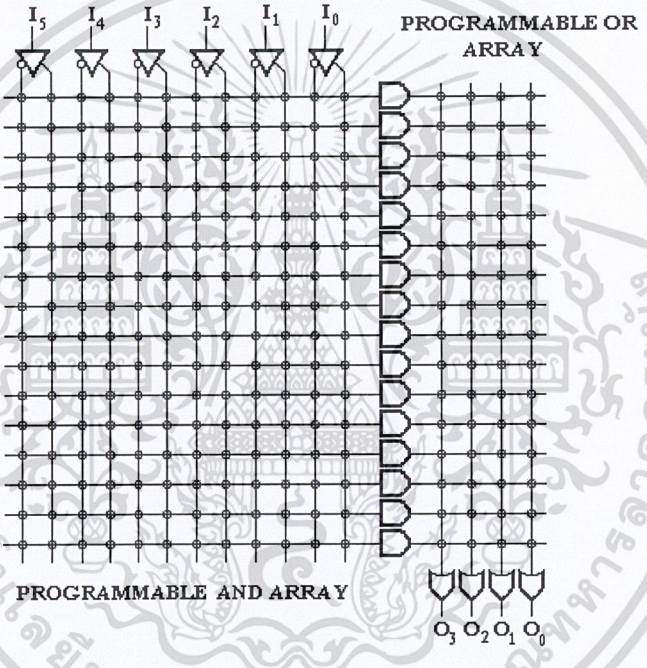
3.1.3พีเอแอล (PAL: Programmable Array Logic)

ในช่วงกลางปี ค. ศ. 1970 บริษัทเอ็มเอ็มไอ (MMI: Monolithic Memory) ในประเทศสหรัฐเมริกา ได้พัฒนาอุปกรณ์พีเอแอล เป็นพีแอลดีชนิดใหม่โดยใช้เทคโนโลยีแบบแอลเอสไอ (LSI: Large Scale Integration) สามารถโปรแกรมเลือกวงจรภายใน โดยใช้ฟิวส์ที่เชื่อมต่ออยู่ระหว่างสัญญาณอินพุตภายนอกและการป้อนกลับจากภายในกับแอนด์เกตที่ต่อเป็นฟังก์ชั่นผลคูณ (Product) อยู่ในตัววงจรรวม

พีแอลเอ (PLA: Programmable Logic Array)

อุปกรณ์ที่สามารถโปรแกรมได้แบบพีแอลเอเกิดขึ้นเมื่อปี ค.ศ. 1975โดยบริษัทซิกเนทริกส์ (Signetics)สหรัฐเอมริกา ซึ่งเป็นบริษัทผู้ผลิตวงจรรวมรายใหญ่รายหนึ่ง ผลิตและนำเสนออุปกรณ์โดยใช้ชื่อว่า เอฟพีแอลเอ (FPLA : Field Programmable Logic Array) สามารถโปรแกรมการต่อลอจิกทั้งทางด้านแอนด์เกตและออเกตได้ และยังเลือกเอาต์พุตเป็น active high หรือ active low โดยต่อ

ผ่านเอ็กคูลสีบออเกต ให้ทำหน้าที่เป็นนอนอินเวอร์เตอร์หรือเป็น อินเวอร์เตอร์แล้วแต่ภายในของพีแอลเอต่อมาปี ค.ศ. 1979 บริษัทซิกเนทริกส์ ได้สร้างเอฟพีแอลเอใหม่ที่มีรีจิสเตอร์ต่ออยู่ภายในวงจรเพิ่มขึ้นรวมทั้งสามารถเลือกสัญญาณอินพุตที่มาจากการป้อนกลับจากรีจิสเตอร์ได้ด้วย ทำให้สามารถใช้อุปกรณ์พีแอลเอใหม่นี้สร้างวงจร State machine ได้ อุปกรณ์ใหม่ที่มีรีจิกเตอร์ อยู่ด้วยนี้ถูกตั้งชื่อใหม่เป็น เอฟพีแอลเอส (FPLS: Field Programmable Logic Sequencer) มีทั้งที่เป็นทีทีแอลและซีมอส



รูปที่ 3.4 แสดงวงจรพื้นฐานภายในของพีแอลเอ

แอลซีเอ (LCA: Logic Cell Array)

อุปกรณ์ชนิดนี้ถูกสร้างขึ้นเมื่อประมาณปี ค.ศ. 1986 โดยบริษัทไซริ้ง (XILINX Inc.) ซึ่งเป็นบริษัทที่ร่วมทำการค้นคว้ากับบริษัทเอ็มเอ็มไอ (MMI) สร้างเป็นอาเรย์ที่ประกอบด้วยเกตจำนวน 1,200-1,800 เกต มีลักษณะของสถาปัตยกรรมที่ใกล้เคียงกับเกตอาเรย์ (Gate array) โดยโปรเซสแบบซีมอส 1.6 ไมครอนขึ้นโลหะคู่ (CMOS 1.6 microns double – layer metal) สามารถโปรแกรมและลบได้โดยใช้กระแสไฟฟ้า (Static RAM based) ภายในจัดเรียงเป็นเมทริกซ์ของลอจิกเซลล์ ล้อมรอบ

ภายนอกด้วยอินพุต เอาต์พุตเซลล์ อุปกรณ์แอลซีเอตัวแรกของบริษัทไซริ้ง คือ แอลซีเอเบอร์ 2064 ประกอบด้วยเซลล์เรียงเป็นเมทริกซ์มีจำนวน 64 เซลล์ แต่ละเซลล์เรียกว่า ซีแอลบี (CLB : Configurable Logic Block) แต่ในปัจจุบัน ได้ดพัฒนามาอยู่ในรูปของ เอฟพีจีเอ (FPGA : Field Programmable Gate Array) ซึ่งมีประสิทธิภาพ ความจุของเกตสูงมากขึ้น โดยสร้างออกมาเป็นอนุกรม (Series) ต่างๆ เช่น ตระกูล XC 3000 และ ตระกูล XC 4000 เป็นต้น

อีพีแอลดี (EPLD: Erasable P5rogrammable Logic Device)

อีพีแอลดีเป็นอุปกรณ์สามารถโปรแกรมได้ที่สามารถลบและกระทำการโปรแกรมใหม่ได้ เพื่อใช้ทำวงจรต้นแบบ ตัวอย่างได้แก่ พีแอลดี ในอนุกรมอีพี (EP series) ของบริษัทอัลเทอร่า (Altera Ine.) ประเทศสหรัฐเอมริกาซึ่งเป็นบริษัทที่ผลิตอีพีแอลดี เป็นรายแรกโดยเริ่มเมื่อปี ค.ศ. 1984 เป็น พีแอลดี ที่ใช้โปรเซสเหมือนกับซีมอสอี พรอม (CMOS EPROM) คือ ใช้มอสทรานซิสเตอร์เชื่อมต่อระหว่างสัญญาณอินพุตกับจุดที่ต้องการแทนการใช้ฟิวส์ดังเดิม ทำให้สามารถโปรแกรมการต่อวงจรภายในอุปกรณ์ด้วยการจ่ายไฟฟ้าตามขนาดที่กำหนดเข้าไปยังตัวอุปกรณ์ และลบได้โดยใช้แสงอุลตร้าไวโอเลทฉายผ่านช่องหน้าต่างกระจกเข้าไปตกกระทบในตัวชิปของอุปกรณ์

3.2 Mask programmable

การใช้งานวงจรรวม ASIC ในเชิงพาณิชย์ จำเป็นต้องใช้วงจรรวม ASIC แบบ Mask programmable เนื่องจากต้นทุนต่อหนึ่งตัวต่ำกว่าวงจรแบบ Field programmable ASIC ในกรณีที่ปริมาณการผลิตสูงนับพันพันนับหมื่นตัวขึ้นไป ตัวอย่าง เช่น วงจรอีพีแอลดี ตัวหนึ่งอาจสูงถึงหนึ่งพันบาท ในขณะที่ถ้าผลิตวงจรรวมที่มีคุณสมบัติเหมือนกันทุกประการโดยใช้ Mask programmable แล้ว ราคาตัวหนึ่งจะลดลงเหลือเพียงไม่ถึงหนึ่งร้อยบาท การใช้วงจรรวมแบบ Mask programmable จึงมีบทบาทสำคัญในการผลิตสินค้าอิเล็กทรอนิกส์ในเชิงพาณิชย์ในปัจจุบัน

วงจรรวมประเภทนี้ หลังจากผู้ใช้ออกแบบวงจรและตรวจสอบการทำงานจนเป็นที่น่าพอใจแล้ว ต้องส่งให้ผู้ผลิตทำการเจือสาร ไม่สามารถโปรแกรมได้ด้วยตนเองเหมือนกับวงจรรวมแบบ Field programmable ช่วงเวลาการผลิตออกใช้งานจึงใช้เวลานับเดือนและมีค่าใช้จ่ายเบื้องต้นในการเจือสารสูง วงจรรวมแบบ Mask programmable ASIC ในปัจจุบันได้แก่ เกตอาเรย์, เซลล์มาตรฐาน และฟู่ลัสตัม (full custom)

เกตอาเรย์ (Gate Array)

วงจรรวมนี้ประกอบด้วยแถวลำดับของวงจรเกต ซึ่งอาจจะเป็นวงจรเกตประเภทเดียวกันหรือต่างชนิดกันก็ได้ กับขั้วต่อสายไฟ (Pad) สำหรับต่อกับวงจรภายนอก ผู้ใช้มีหน้าที่ออกแบบการเชื่อมโยงทางไฟฟ้าระหว่างวงจรเกตแต่ละตัวและขั้วต่อสายไฟเพื่อให้ทำหน้าที่ตามต้องการ แล้วจึงส่งผลการออกแบบนี้ไปยังโรงงานผู้ผลิตวงจรรวมเกตอาเรย์ นั้นไปทำการเจือสารต่อไป โดยทั่วไปไปแล้วปรากฏในการใช้วงจรรวมเกตอาเรย์สร้างวงจรที่ต้องการใช้งานทั่วไปจะมีเพียง 25-30% ของพื้นที่ซิลิคอนที่ใช้สำหรับวงจรเกต ส่วนพื้นที่ที่เหลือจะใช้กับการเชื่อมโยงวงจรเกตเข้าด้วยกันเองและเข้ากับขั้วต่อสายไฟ พื้นที่ซิลิคอนจะใช้ประโยชน์สูงถึง 75% สำหรับวงจรเชิงเลขที่มีลักษณะสม่ำเสมอ เช่น หน่วยความจำเป็นต้น

เซลล์มาตรฐาน (Standard Cell)

ปัญหาการใช้พื้นที่ซิลิคอนอย่างมากมายสำหรับการเชื่อมโยงวงจรของวงจรรวม เกตอาเรย์ทำให้เกิดขีดจำกัดในความซับซ้อนของวงจรที่ใช้ ประกอบกับผู้ใช้จำนวนมากต้องการรวบรวมวงจรรวมมาตรฐาน อาทิ วงจรตระกูล 7400 วงจรตระกูล 4000 จำนวนหลายตัวเข้าเป็นวงจรรวม ASIC เพียงตัวเดียว ทำให้เกิดวงจรรวม ASIC แบบเซลล์มาตรฐานขึ้น วงจรรวม เซลล์มาตรฐานนี้ ผู้ใช้เป็นผู้เลือกกลุ่มวงจรทำหน้าที่ต่างๆ เช่น เกต ฟลิบฟลอป ตัวนับ ตัวเลื่อน หน่วยความจำ หรือกระทั่งไมโครโปรเซสเซอร์จากแฟ้มข้อมูล (Library) ของผู้ผลิต ซึ่งอาจจะเป็นแฟ้มข้อมูลคอมพิวเตอร์ เหมาะสมสำหรับวงจรเชิงเลขที่มีความซับซ้อน การผลิตวงจรรวม เซลล์มาตรฐานจะมีต้นสูงกว่าวงจรรวมเกตอาเรย์และใช้เวลาในการออกแบบและเจือสารยาวกว่าสองถึงสามเท่าตัว วงจรรวมเซลล์มาตรฐานจึงเหมาะสมกับการใช้งานเชิงพาณิชย์ที่มีปริมาณการผลิตนับหมื่นตัวขึ้นไปบทบาทวงจรรวมเซลล์มาตรฐานจะมีเพิ่มมากขึ้นในอนาคต เมื่อตัวทุนการเจือสารลดลง

ฟูลัสตัม (Full Custom)

วงจรฟูลัสตัม นี้ผู้ใช้เป็นผู้ออกแบบเองทั้งหมด ตั้งแต่ระดับวงจรเชิงเลขจนกระทั่งจึงระดับกายภาพ แล้วจึงส่งข้อมูลการออกแบบไปให้ผู้ผลิตเจือสารในรูปแฟ้มข้อมูลมาตรฐาน เช่น GDS II, CIF การใช้ฟูลัสตัม ASIC นี้เท่าที่แพร่หลายอยู่ในปัจจุบัน จะเป็นไปเพื่อการศึกษาและการวิจัยและเพื่อการผลิตจำนวนน้อย ส่วนใหญ่จะเป็นการเจือสารในลักษณะที่ใช้ค่าใช้จ่ายร่วมกันกล่าวคือ รวบรวมการออกแบบกลายวงจรลงบนแผ่นซิลิคอนเดียวกันเพื่อประหยัดค่าเจือสาร

รูปแบบการเจือสารวงจรรวมฟูดัสตัมที่ประกอบด้วยการออกแบบหลายวงจรรวมอยู่บนแผ่นซิลิคอนเดียวกัน มีกระทำอยู่ 3 รูปแบบดังนี้

1. Multi-project wafer (MPW) การเจือสารวงจรรวมฟูดัสตัมแบบนี้ แผ่นเวเฟอร์ (Wafer) จะแบ่งเป็นส่วนๆ ที่เรียกว่าได (Die) และไดแต่ละชุดจะเจือสารวงจรต่างกัน ผู้ผลิตที่ให้บริการแบบนี้ได้แก่ โมซิส (MOSIS) และ ออบิต (ORBIT) ในสหรัฐอเมริกา

2. Multi-project chip (MPC) เป็นการเจือสารออกแบบหลายวงจรลงบนไดชุดเดียวกันโดยที่ไดทุกชุดบนเวเฟอร์เดียวกันจะมีลักษณะเหมือนกัน การจัดวางวงจรที่ออกแบบลงบนได เน้นการใช้พื้นที่ซิลิคอนให้เป็นประโยชน์สูงสุดเป็นสำคัญ ตัวอย่างผู้ผลิตที่ให้บริการแบบนี้ได้แก่ อาวา (AWA) ในออสเตรเลีย และ อีเอสทู (ES II) ในสหราชอาณาจักร

3. Multi-project reticle (MPR) เป็นการเจือสารในลักษณะผสมผสานระหว่างฟูดัสตัมกัลเซลล์มาตรฐานกล่าวคือ ได้จะแบ่งออกเป็นส่วนๆ อย่างสม่ำเสมอแต่ละส่วนบรรจุการออกแบบแต่ละวงจร โดยไดชุดหนึ่งจะบรรจุวงจรที่ออกแบบได้ประมาณ 8-วงจรและไดทุกชุดในแผ่นเวเฟอร์จะมีลักษณะเหมือนกัน-ในปัจจุบันมีเพียงอาวาที่ใช้บริการการเจือสารลักษณะนี้ โดยจำกัดเฉพาะประเภทเซลล์มาตรฐานเท่านั้น

3.3 เอฟพีจีเอ (FPGA: Field Programmable Gate Array)

เป็นอุปกรณ์ที่ถูกพัฒนาต่อจากอุปกรณ์แอลซีเอของบริษัทไซรินซ์ (XILINX Inc.) โดยมีประสิทธิภาพการทำงานและมีปริมาณความหนาแน่นของเกตสูง สามารถจะกำหนดฟังก์ชั่นการทำงานได้ความต้องการของผู้ใช้โดยผ่านการโปรแกรมเอฟพีจีเอได้รวบรวมข้อดีทั้งหมดของการทำดัสตัมวีแอลเอสไอ (Custom VLSI) มารวมไว้ทั้งหมดได้แก่ การออกแบบการผลิต, ลดเวลาที่จะส่งตัวผลิตภัณฑ์ออกตลาด ซึ่งเป็นประโยชน์ต่อการผลิตวงจรเป็นอย่างมาก นักออกแบบเพียงกำหนดฟังก์ชั่นการทำงานของวงจร ดังนั้นการออกแบบวงจรโดยใช้เอฟพีจีเอ สามารถออกแบบและทดสอบภายในเวลาเพียง 2-3 วันเท่านั้น ตรงกันข้ามกับการออกแบบโดยใช้เกตอาเรย์ ซึ่งใช้เวลาหลายอาทิตย์การเปลี่ยนแปลงแก้ไขแบบก็เช่นเดียวกัน จากประโยชน์ของเอฟพีจีเอ ดังกล่าวมา ทำให้เกิดการประหยัดค่าใช้จ่ายเป็นอย่างมาก เพราะได้ความเสี่ยงในการที่จะต้องแก้ไขตัววงจร การเลื่อนเวลาการออกผลิตภัณฑ์ ลดค่าเอ็นอาร์อี (NRE : Nonrecurring Engineering Cost) ลงไปด้วย

เทคโนโลยีของ FPGA

เนื่องจากเป็นลักษณะของชิพที่สามารถโปรแกรมได้นั่นก็คือ สามารถกำหนดจุดเชื่อมต่อต่างๆ ภายในได้ เพื่อประกอบเป็นลักษณะของวงจรตามที่เราต้องการได้ ซึ่งเราสามารถแบ่งลักษณะของจุดเชื่อมต่อต่างๆได้ดังนี้

1. Physical Changing

1.1 Fused สามารถโปรแกรมได้เพียงครั้งเดียว หลังจากโปรแกรมจุดเชื่อมต่อขาดจากกัน

1.2 Anty Fuse สามารถโปรแกรมได้เพียงครั้งเดียว หลังจากโปรแกรม จุดเชื่อมต่อจะเชื่อมถึงกัน

2. Memory Base

2.1 EEPROM – Base FPGA

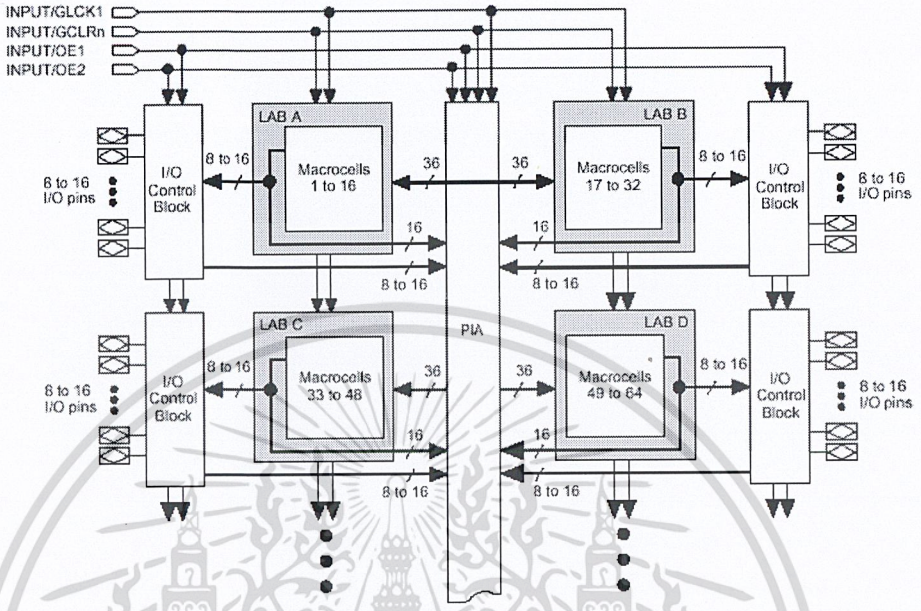
มักเรียก FPGA ประเภทนี้ว่า CPLD จะใช้เทคโนโลยีเหมือนกับ EEPROM ในการโปรแกรม ซึ่งจะทำให้มีความจุของเกตต่ำ โดยทั่วๆไปจะน้อยกว่า 20,000 เกต แต่ข้อดีของ EEPROM-Base FPGA คือสามารถเก็บข้อมูลที่โปรแกรมลงไปได้โดยไม่จำเป็นต้องมีไฟเลี้ยง และในการโปรแกรม จะใช้ทรานซิสเตอร์ 1 ตัวต่อ 1 บิท สามารถโปรแกรมได้ประมาณ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

2.2 SRAM - Base FPGA

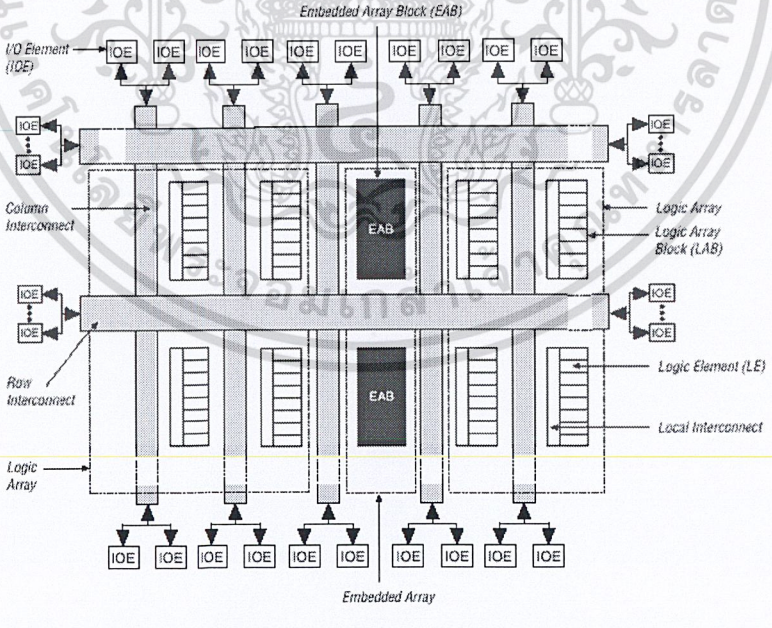
จะใช้เทคโนโลยีเหมือน SRAM ในการโปรแกรม ซึ่งจะสามารถทำให้โปรแกรมซ้ำได้ไม่จำกัดจำนวนครั้ง มีความจุของเกตปานกลางถึงสูงมาก (ประมาณ 10,000 – 1,000,000 เกต) จะใช้ Look-Up Table ในการทำลอจิกฟังก์ชัน (Logic Function) และจะมีการจัดทรัพยากรภายในโครงสร้างแบบอาร์เรย์ ข้อดีของ SRAM - Base FPGA คือจะใช้เวลาในการโปรแกรมน้อย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

โครงสร้างภายใน

ลักษณะโครงสร้างภายใน เป็นอาร์เรย์ลอจิกบ๊อกซ์ที่สามารถทำการโปรแกรมได้ดังแสดงในรูปที่ 3.5



รูปที่ 3.5 โครงสร้างภายในของ FPGA ตระกูล MAX7000S



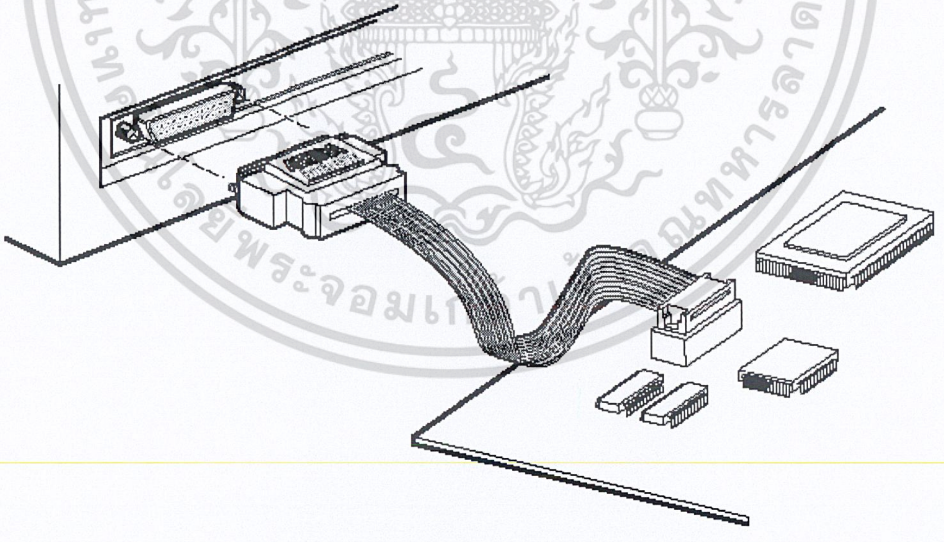
รูปที่ 3.6 โครงสร้างภายในของ FPGA ตระกูล FLEK10K

ทำไมการออกแบบถึงทำได้ง่ายและสะดวกรวดเร็ว

1. ในการออกแบบเราไม่จำเป็นต้องรู้ถึงโครงสร้างภายในของตัวชิพ เพียงแต่รู้ขั้นตอนการออกแบบลอจิกก็พอ ไม่เหมือนไมโครโปรเซสเซอร์ที่เราจำเป็นต้องรู้โครงสร้างภายในรวมถึงการศึกษาการเขียนภาษา Assembly ซึ่งแต่ละตัวก็ไม่เหมือนกันด้วย

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

3. การโปรแกรมสามารถทำได้เองและใช้เวลาไม่นาน โดยเพียงแค่ส่งข้อมูลผ่านสายดาวโหลดทางพอร์ตของคอมพิวเตอร์ก็สามารถโปรแกรมตัวชิพได้ขณะที่เอยู่ในระบบ โดยไม่จำเป็นต้องถอดมาโปรแกรมข้างนอก ดังรูปที่ 3.7 และที่สำคัญสามารถโปรแกรมได้หลายครั้ง จึงทำให้ง่ายในการแก้ไขและพัฒนาโดยที่ไม่ต้องเสียค่าใช้จ่ายเพิ่มแต่อย่างใด



รูปที่ 3.7 การโปรแกรมลงในชิพ

การออกแบบโดยใช้ภาษาอธิบายพฤติกรรมของฮาร์ดแวร์ (HDL)

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

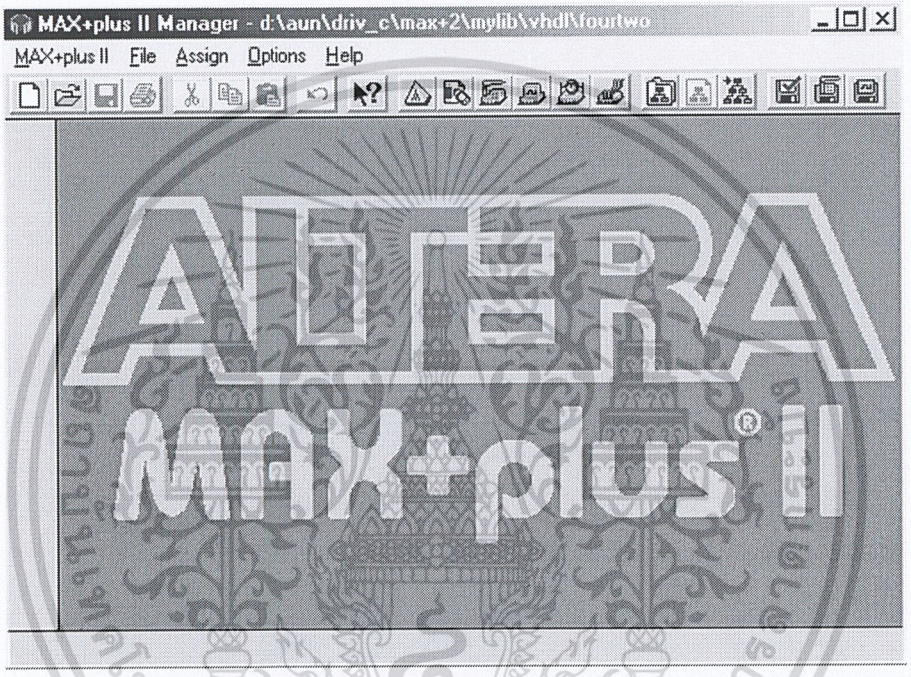
การสังเคราะห์วงจร (Logic Synthesis)

ในขั้นตอนนี้จะใช้ซอร์พแวร์ในการสังเคราะห์วงจร (Synthesis Tools) ทำการสังเคราะห์พฤติกรรมของวงจรที่ได้จากการออกแบบด้วย Schematic หรือ VHDL ซึ่งต้องทำการตรวจสอบด้วยว่าซอร์พแวร์นั้นสนับสนุนเทคโนโลยี FPGA (FPGA Library) ที่ต้องการหรือไม่ ตัวอย่างเช่น FPGA ของบริษัท XILINX และบริษัท ALTERA จะมีซอร์พแวร์หลายตัวที่สามารถใช้ได้ เช่น **Max+Plus II** ในขั้นตอนนี้ซอร์พแวร์สังเคราะห์วงจรจะทำการแปลงโค้ด VHDL และทำการ Optimize เพื่อให้ได้วงจรตามเทคโนโลยีที่เลือกใช้ ในการสังเคราะห์วงจรนั้นวงจรระดับเกต (Gate Level) จะไม่เหมาะสมกับโครงสร้างทีมีอยู่ในอุปกรณ์ FPGA ดังนั้นในการ Optimize ซอร์พแวร์สังเคราะห์วงจร จะต้องทำการ Optimize ให้ได้เป็นวงจรที่ประกอบด้วยกลุ่มของลอจิกที่เหมาะสมกับอุปกรณ์ FPGA นั้นๆ จึงทำให้ผลที่ได้มีประสิทธิภาพและในขั้นตอนการสังเคราะห์วงจรนี้ ผู้ออกแบบสามารถกำหนดข้อบังคับสำหรับโมเดลแต่ละตัวได้ เช่น ข้อบังคับในเรื่องเวลา (Timing Constraints) หรือข้อบังคับในเรื่องของพื้นที่ (Area) หรือกำหนดชนิดและตำแหน่งของ I/O ซึ่งข้อบังคับเหล่านี้จะถูกนำไปใช้ในขั้นตอน Optimize เพื่อให้วงจรที่ได้เป็นไปตามที่กำหนด ส่วนสำคัญในการ Optimize คือการเทียบ (Mapping) โมเดลให้เข้ากับเทคโนโลยีที่ใช้เพื่อให้ได้วงจรที่เหมาะสมกับโครงสร้างและสถาปัตยกรรมภายในอุปกรณ์

FPGA เมื่อทำการสังเคราะห์วงจรเสร็จแล้ว ซอร์พแวร์ การสังเคราะห์วงจรก็จะมีการรายงานผลว่าโมเดลที่ออกแบบไปนั้นเป็นอย่างไร เช่นมีค่าความหน่วง (Delay) เท่าไหร่ ใช้ทรัพยากรต่างๆใน FPGA อะไรบ้าง เมื่อมาถึงขั้นตอนนี้ ผู้ออกแบบก็จะทราบว่าโมเดลเป็นไปตามข้อบังคับหรือไม่ ถ้าไม่ก็สังเคราะห์ใหม่จนกว่าจะเป็นไปตามที่กำหนด

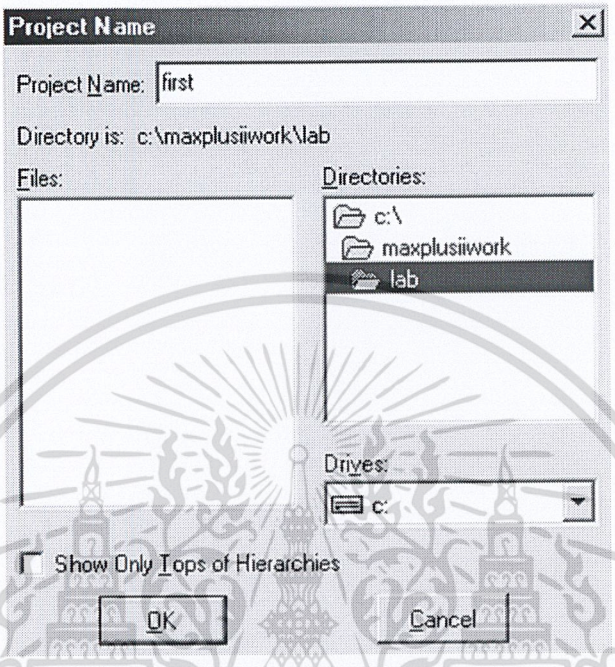
บทที่ 4
การใช้งานโปรแกรม Max Plus II เบื้องต้น

1. Run โปรแกรม  Max Plus II 9.5 BASELINE ขึ้นมา



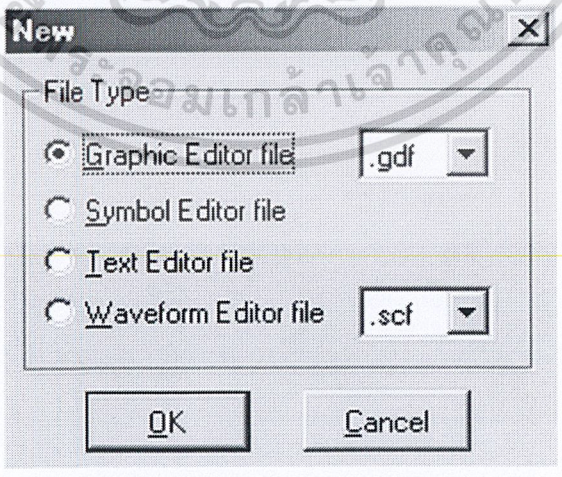
รูปที่ 4.1 โปรแกรม Max Plus II 9.5 BASELINE

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



รูปที่ 4.2 การตั้งชื่อโปรเจค

2. เลือกประเภทของไฟล์ที่จะสร้าง **File / New / Graphic Editor File**จะใช้งานขึ้นมาวางไว้ที่ Graphic Form



รูปที่ 4.3 การเลือกประเภทไฟล์

โดยที่

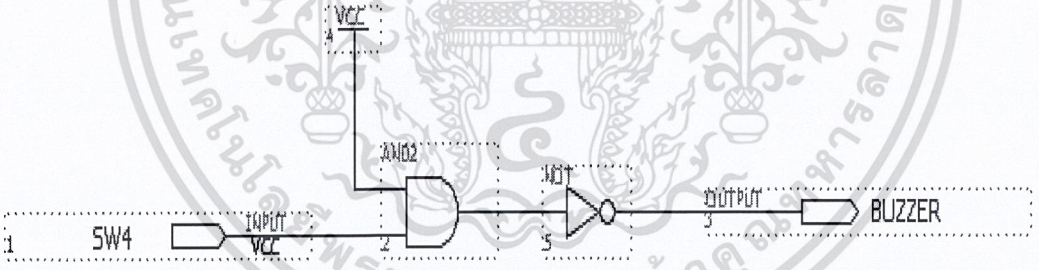
Graphic Editor File: เป็นไฟล์กราฟฟิกส์ที่เราสามารถนำอุปกรณ์ต่างๆใน Library มาวางต่อกันได้เลย

Symbol Editor File: เป็นไฟล์ที่ใช้เก็บสัญลักษณ์เพื่อสื่อให้รู้ว่าโมเดลที่เราได้สร้างขึ้นมีอินพุท , เอาท์พุทเป็นอย่างไร

Text Editor File: เป็น Text file ใช้สำหรับเขียน Source code เพื่ออธิบายพฤติกรรมของวงจรหรือโมเดลต่างๆที่เราจะสร้างขึ้นหรือเพื่อไว้สำหรับเก็บข้อความทั่วๆไป

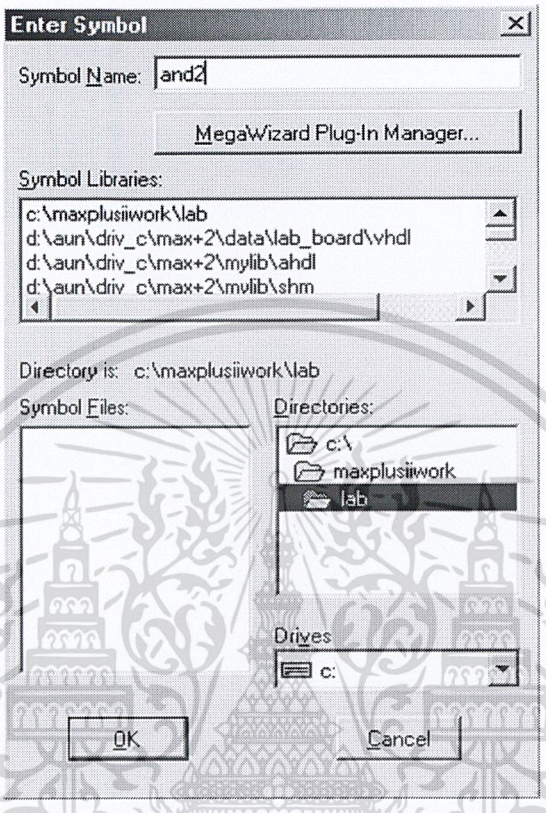
Waveform Editor File: เป็นไฟล์ไว้สำหรับการกำหนดรูปแบบของสัญญาณอินพุทเพื่อใช้ในการ Simulate และไว้สำหรับให้ Max Plus II แสดงผลของเอาท์พุทที่ได้จากการ Simulate

3.ในการอธิบายการใช้งานโปรแกรม Max Plus II นี้เราจะใช้วงจรดังรูปที่ 4.4 ประกอบคำอธิบาย



รูปที่ 4.4 วงจรตัวอย่าง

จากรูปที่ 4.4 AND GATE, NOT GATE, INPUT, OUTPUT, VCC จะเป็นอุปกรณ์ที่มาพร้อมกับโปรแกรม Max Plus II ในการที่จะเอาอุปกรณ์ต่างๆ ใน Library ของ Max Plus II มาใช้ให้ทำการ Double Click ที่ Graphic Form จะมี Dialog Enter Symbol ปรากฏขึ้นมา ให้เลือกอุปกรณ์ที่เราต้องการจะใช้งานขึ้นมาวางไว้ที่ Graphic Form



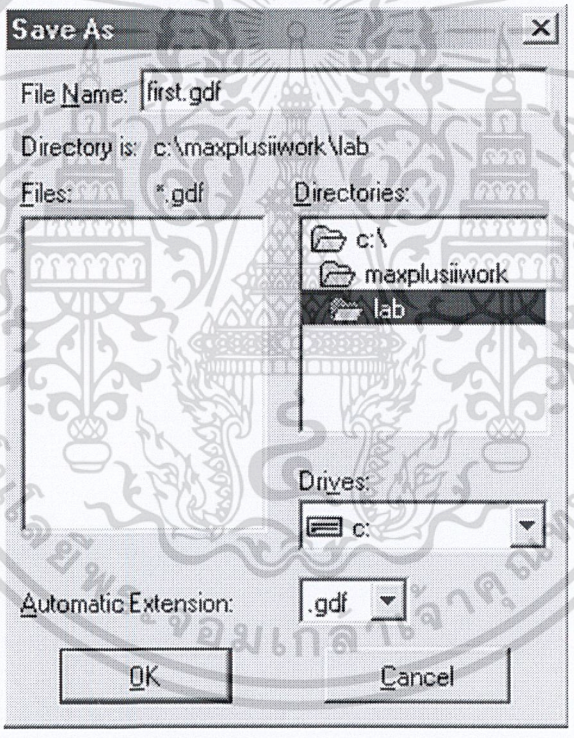
รูปที่ 4.5 การกำหนดสัญลักษณ์

โดยที่

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

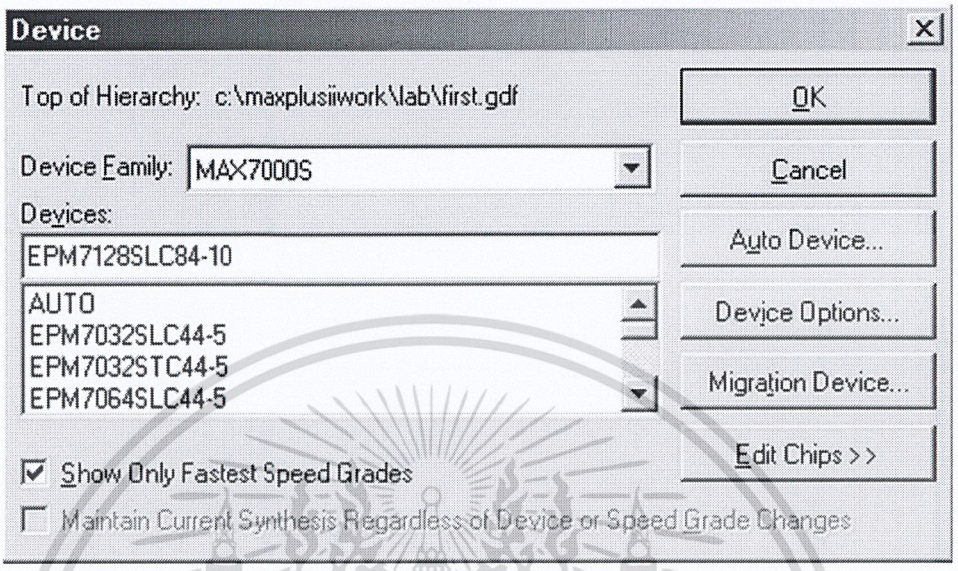
อุปกรณ์ทั้งหมดจะอยู่ใน Symbol Library ชื่อ **...\maxplusii\max2lib\primg** เมื่อทำการวางอุปกรณ์จนครบทุกตัวแล้ว ให้ทำการเชื่อมต่ออุปกรณ์ต่างๆเข้าด้วยกันโดยเลื่อน pointer ของ mouse ไป

ที่ขาต่างๆของอุปกรณ์ pointer ของ mouse จะเปลียนเป็นเครื่องหมาย + ให้ทำการ Click ขวาค้างไว้แล้วลากเส้นไปต่อกับ input หรือ output ของอุปกรณ์ตัวอื่น เมื่อเชื่อมต่ออุปกรณ์ต่างๆจนครบทั้งหมดแล้วให้ทำการเปลียน PIN_NAME ของอุปกรณ์ input กับ output โดยการไป **Double Click** ที่ PIN_NAME ของ input แล้วเปลี่ยนชื่อเป็น **SW1** สำหรับ PIN_NAME ของ output เปลี่ยนเป็น **BUZZER** เมื่อวาดรูปวงจรเสร็จเรียบร้อยแล้วให้ทำการบันทึกวงจรที่เราสร้างขึ้นมา **File / Save as** ให้ใส่ชื่อไฟล์ป็น **first.gdf** แล้วก็ OK



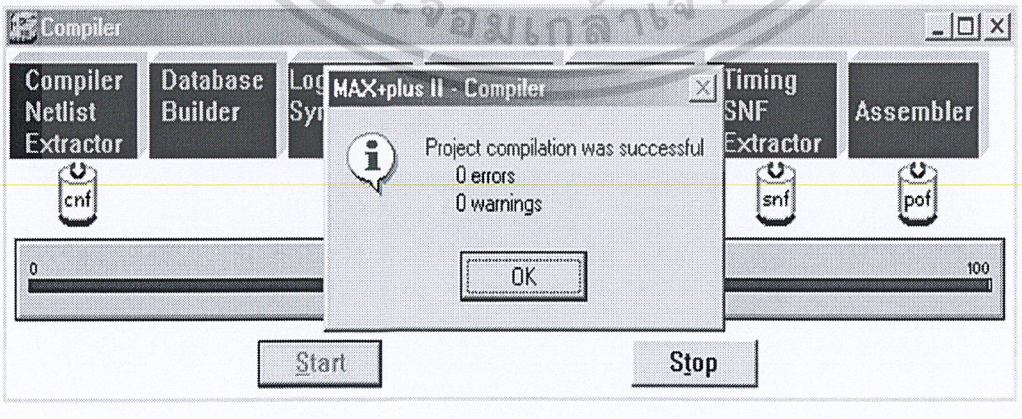
รูปที่ 4.6 การบันทึกข้อมูล

4. ทำการระบุเบอร์ของชิพที่เราจะใช้ **Assign / Device** จะมีไดอะล็อก Device ปรากฏขึ้นมา ให้ทำการเลือก Device Family เป็น **MAX700S** และเลือก Device เป็น **EPM7128SLC84-10**



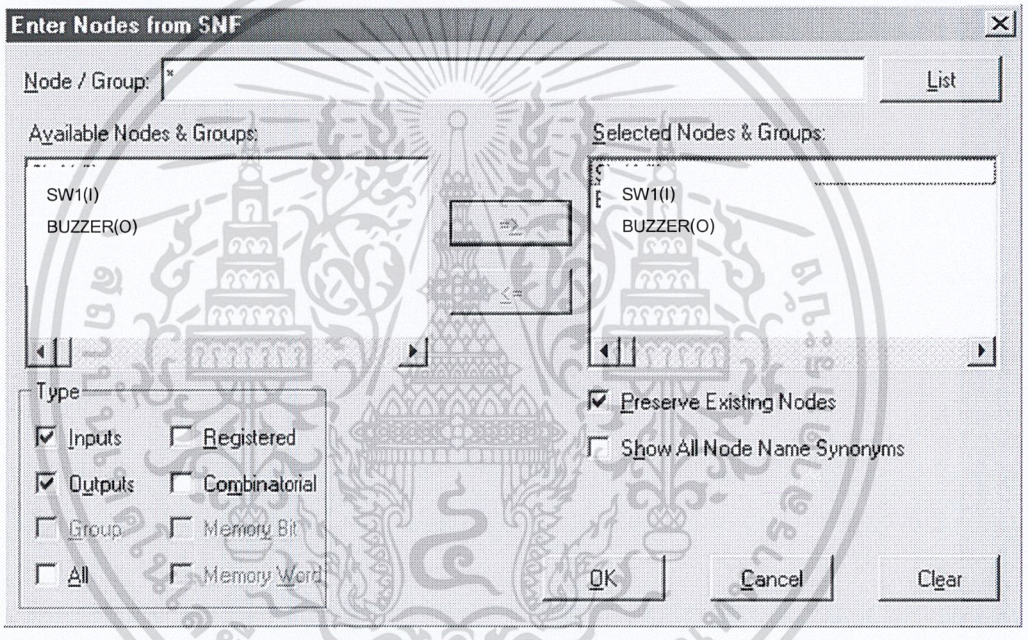
รูปที่ 4.7 การกำหนดอุปกรณ์

5. ทำการคอมไพล์วงจรที่เราได้สร้างขึ้นมา **Max+Plus II / Compiler / Start เมื่อคอมไพล์เสร็จจะมีหน้าต่างรายงานผลการคอมไพล์** error กับ warring หากมีความผิดพลาดเกิดขึ้นจะมีข้อความสีแดงบอกว่า error และจะบอกด้วยว่า error เพราะอะไร



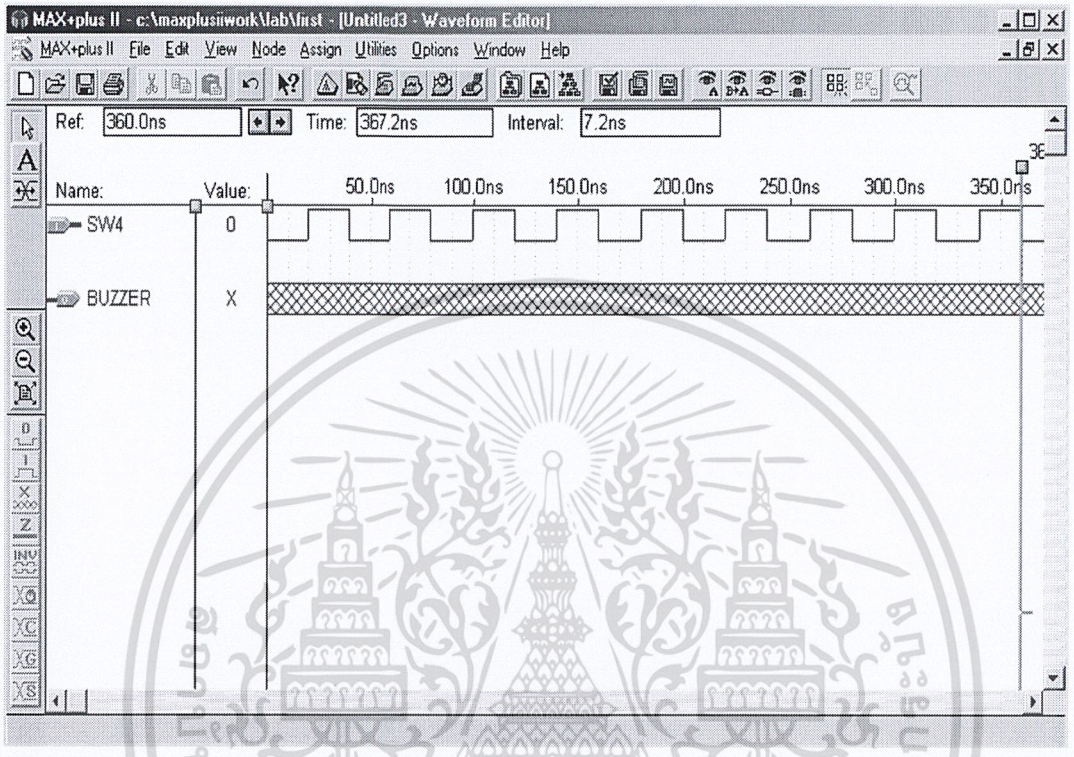
รูปที่ 4.8 รูปการคอมไพล์

6. ทำการ Simulate โดยจะต้องสร้างไฟล์ Waveform ขึ้นมาก่อน **File / New / Waveform Editer file** (จะได้หน้าต่างดังรูปที่ 4.17) ก่อนที่จะทำการ Simulate จะต้องทำการกำหนดอินพุทให้แก่วงจรก่อน โดยโหลด Noad ต่างๆ เข้ามา **Noad / Enter Noad from SNF จะมีไดอะล๊อก** Enter Nodes from SNF ปรากฏขึ้นมาให้คลิกที่ **List** จะมี Noad ต่างๆ ที่อยู่ในวงจรปรากฏขึ้นมาให้เราเลือก Noad ที่เป็น input กับ output ดังรูปที่ 4.16 โดยการกดปุ่มที่มี **เครื่องหมายลูกศรชี้ไปทางขวา**



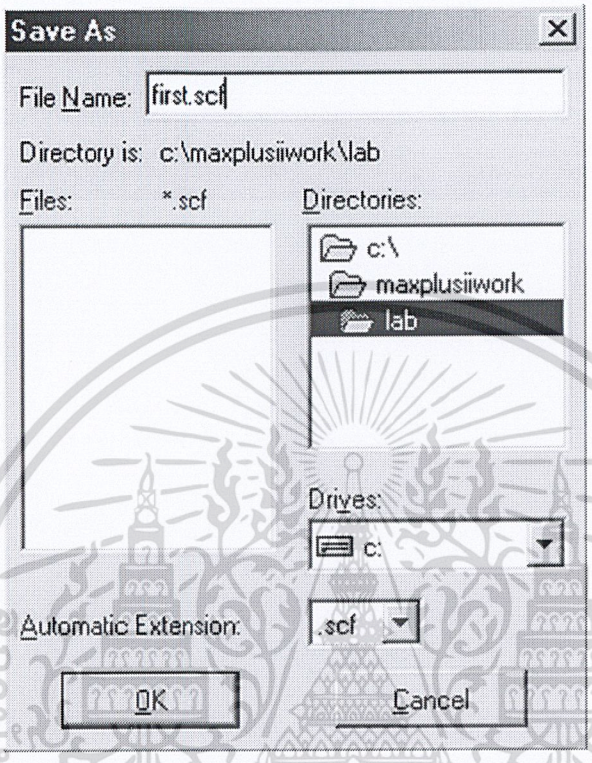
รูปที่ 4.9 การกำหนดโหนด

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



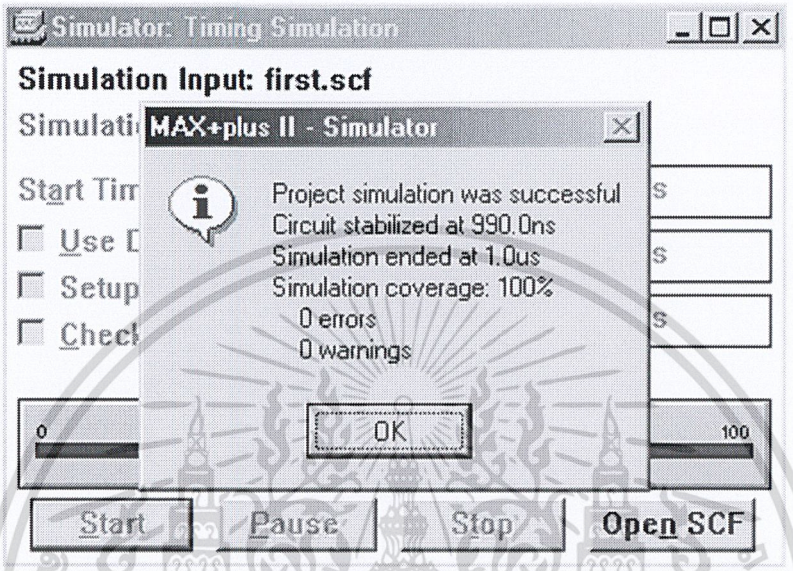
รูปที่ 4.10 การจำลองการทำงาน

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



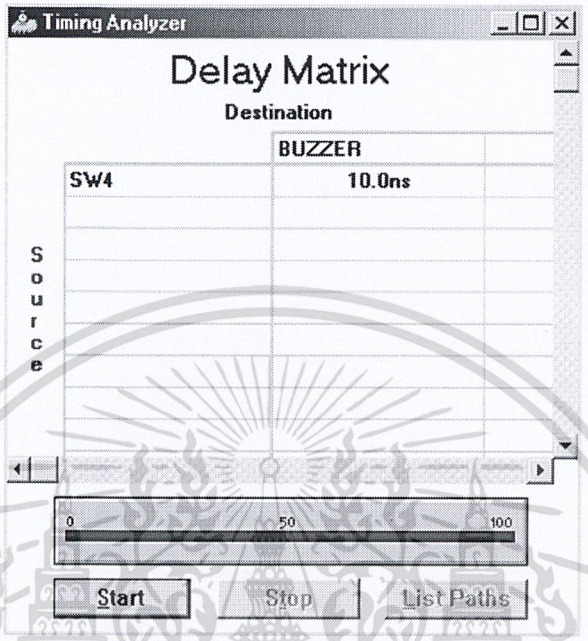
รูปที่ 4.11 การบันทึกรูปกราฟ

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



รูปที่ 4.12 การจำลองการทำงานอีกครั้ง

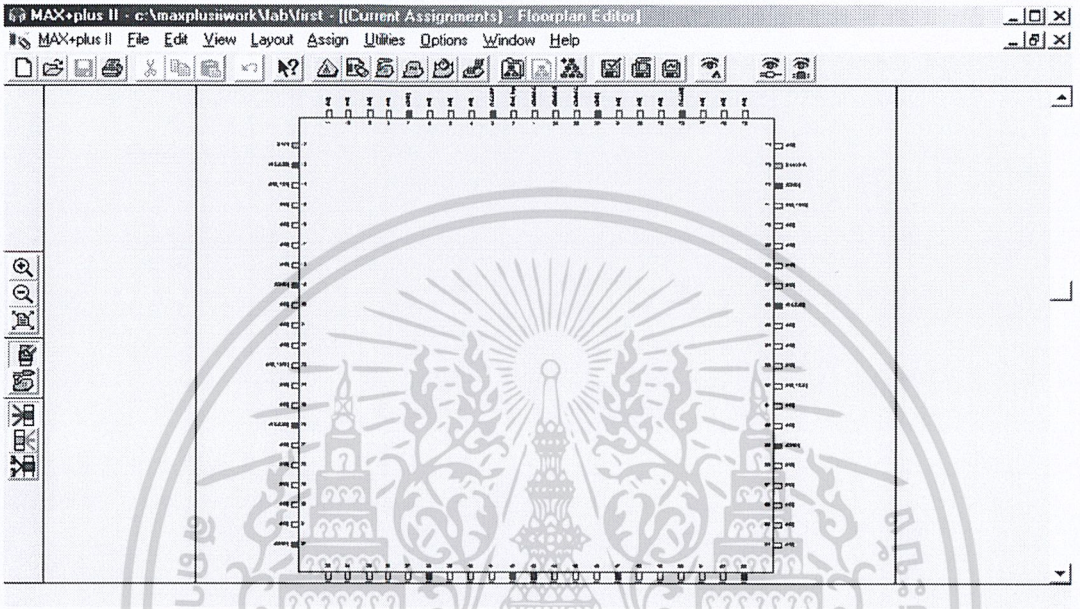
7. วิเคราะห์ Timing Analyzer จะใช้สำหรับหาค่า Delay Time ระหว่าง Noad ต่างๆ **Max+Plus II / Timing Analysis** จะมี ไดอะล็อก Timing Analyzer ปรากฏขึ้นมาให้เลือกชนิดของการวิเคราะห์ ให้ทำการเลือกวิเคราะห์ค่าเวลาหน่วง **Analysis / Delay Time**



รูปที่ 4.13 การจำลองการหน่วงเวลา

8. การเปลี่ยนแปลงตำแหน่งขาของชิพ

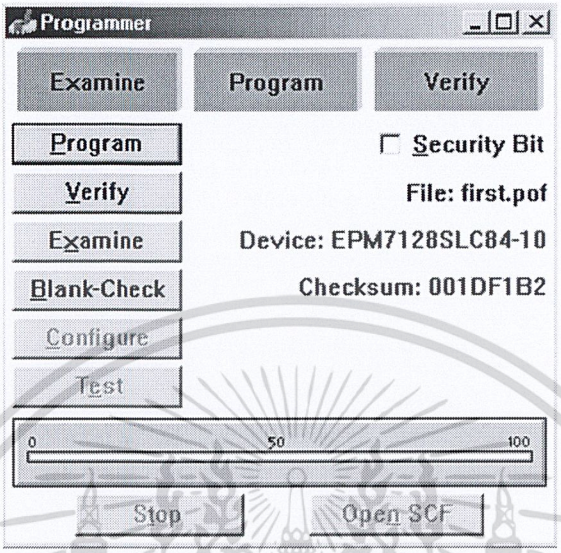
เราสามารถเปลี่ยนแปลงตำแหน่งขาของชิพโดยคลิกที่ **Max+Plus / Floorplan Editor** จะปรากฏฟอร์มของ Floorplan Editor ขึ้นมาเลือกที่เมนู **Assign / Back-Annotate Project** คลิกที่ **Chip,Pin & Device / OK** สังเกตุว่าเมื่อผ่านขั้นตอนนี้เราจะเห็นตำแหน่งขาที่โปรแกรม Max+Plus II ได้เลือกให้เรา ที่นี้เราสามารถที่จะเปลี่ยนแปลงตำแหน่งขาต่างๆเหล่านี้ได้เพื่อให้เหมาะสมกับบอร์ด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



รูปที่ 4.14 การกำหนดขาใช้งาน

เมื่อทำการเปลี่ยนแปลงตำแหน่งขาของชิพเรียบร้อยแล้วให้ save และ คอมไพล์ใหม่อีกครั้ง Project / Save Compile & Simulate

9. การโปรแกรมวงจรลงในชิพ Max+**Plus II / Programmer / Program** ไฟล์ข้อมูลที่จะโปรแกรมลงในชิพหากเลือก Device ของ FPGA เป็นอุปกรณ์ประเภท EEPROM – Base FPGA หรือที่เรียกว่า CPLD (เบอร์ IC จะขึ้นต้นด้วย EPM) ไฟล์ที่จะโปรแกรมจะมีนามสกุล *.pof



รูปที่ 4.15 การโปรแกรม

10. ทดสอบการทำงานจริงของวงจรที่เราได้โปรแกรมลงในชิพ แล้วอธิบายการทำงาน

สรุปขั้นตอนการใช้ MAX PLUS IIวาดวงจร

1. Run โปรแกรม **Max Plus II 9.5 BASELINE** ขึ้นมา
2. ตั้งชื่อโปรเจ็คเริ่มจาก **File / Project / Name** เลือก Folder ที่จะเก็บโปรเจ็คของเราและใส่ชื่อของโปรเจ็คตามต้องการ
3. สร้างไฟล์วงจร **File / New / Graphic Editor File** แล้ววาดวงจรที่ต้องการ แล้ว **save Compile**
4. ระบุเบอร์ของชิพ **Assign / Device** ให้ทำการเลือก Device Family เป็น **MAX700S** และเลือก Device เป็น **EPM7128SLC84-10**
5. คอมไพล์วงจรที่เราได้สร้างขึ้นมา **Max+Plus II / Compiler / Start Simulate**
6. Simulate โดยจะต้องสร้างไฟล์ Waveform ขึ้นมาก่อน **File / New / Waveform Editer file** โหลด Noad ต่างๆ เข้ามา Noad / Enter Noad from SNF

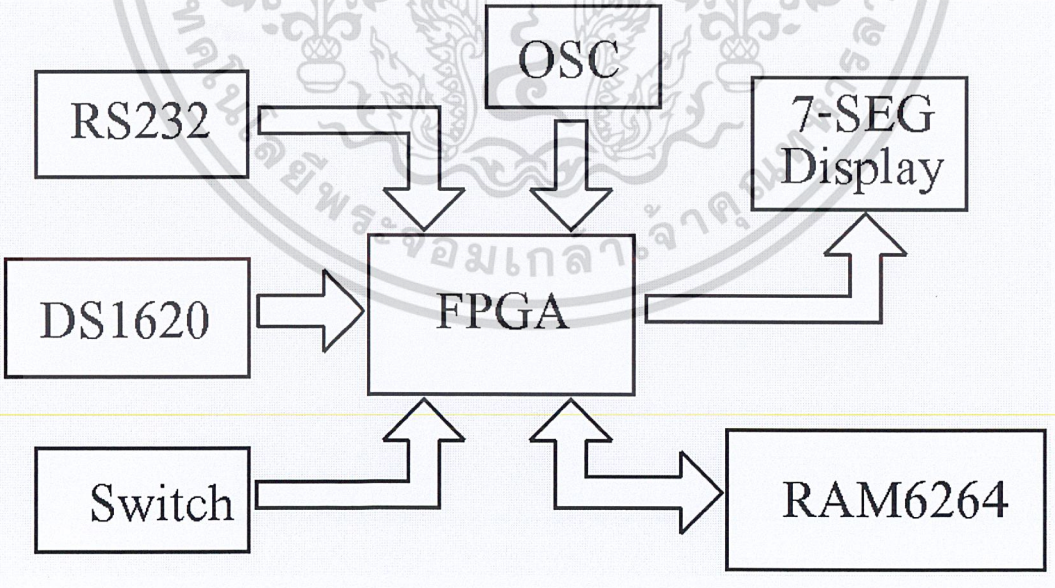
บทที่ 5
การออกแบบ

การออกแบบนั้นแบ่งออกได้เป็น 2 ส่วน คือ

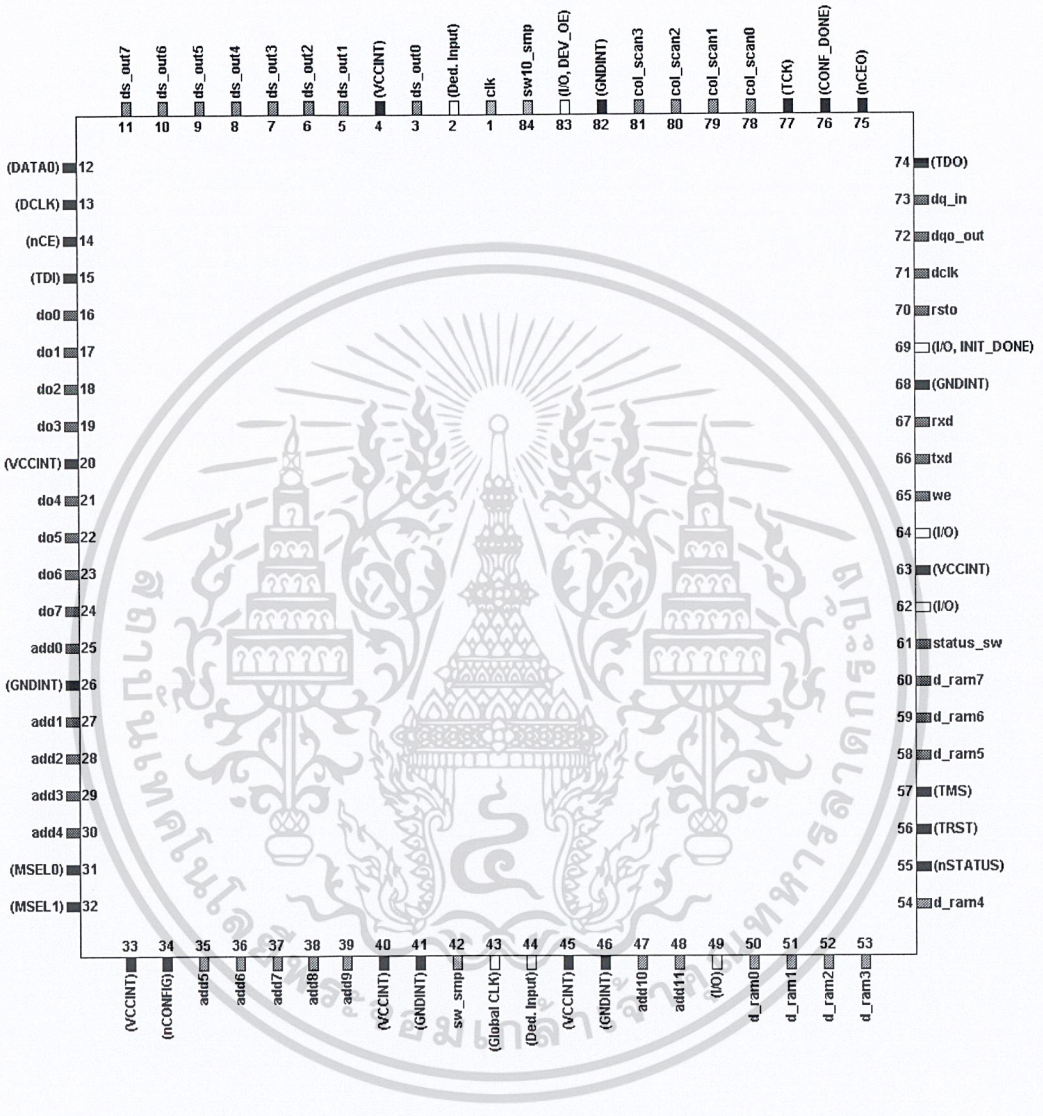
- ส่วนของฮาร์ดแวร์
- ส่วนของโปรแกรมควบคุม

5.1 ส่วนของฮาร์ดแวร์

การออกแบบพิจารณาได้จาก Block Diagram การทำงานของระบบ ดังรูปที่ 5.1 IC FPGA จะเป็นตัวควบคุมการเชื่อมต่อและควบคุมการทำงานของอุปกรณ์ต่างๆ จากรูปจะใช้อุปกรณ์การวัดอุณหภูมิ IC DS1620 ตัวถังแบบ DIP-8 สามารถวัดอุณหภูมิได้ในช่วง -55 °C ถึง 125 °C มีรูปแบบการส่งข้อมูลแบบอนุกรมมีโหมดการทำงานของตัวชิป 2 โหมด คือ Temperature Conversion และ Thermostat ในที่นี้จะใช้เพียงโหมด Temperature Conversion ซึ่งจะใช้จำนวนขาเท่ากับ 3 เส้น(ไม่รวม Vcc กับ Gnd) สำหรับการกำหนดขาต่างๆ ให้กับ IC FPGA จะถูกกำหนดได้ในการโปรแกรมดังรูปที่ 5.2



รูปที่ 5.1 Block Diagram แสดงการทำงานในส่วนของฮาร์ดแวร์



รูปที่ 5.2 แสดงตำแหน่งขาที่ใช้งานต่างๆ ที่ถูกกำหนดในโปรแกรม

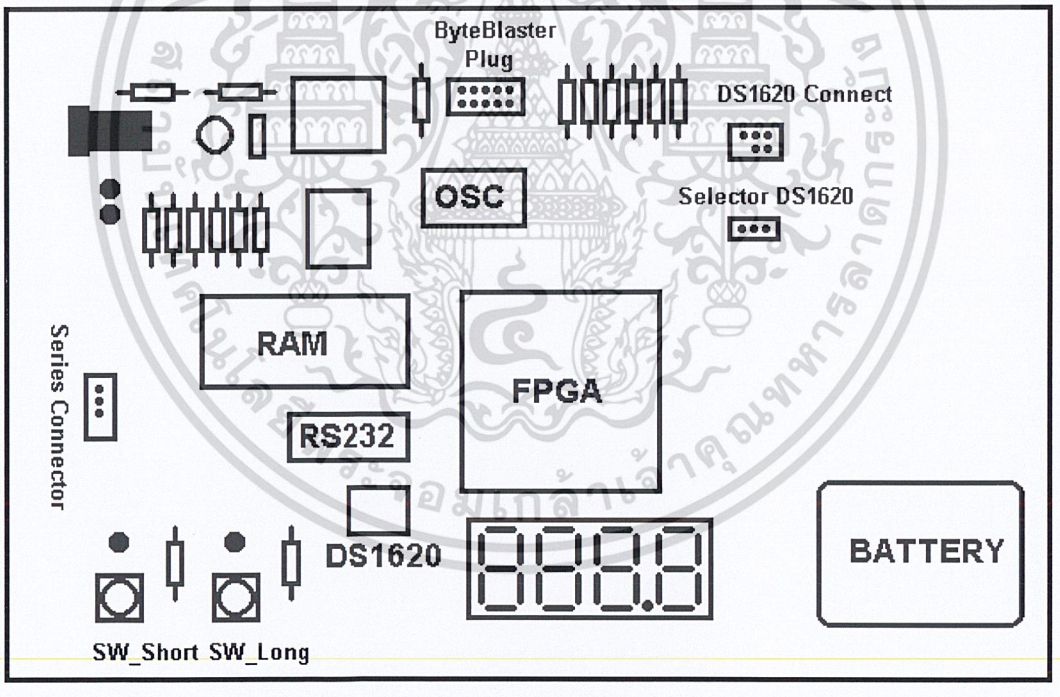
หลักการทำงาน

โครงสร้างของบอร์ด จะเป็นดังรูปที่ 5.3 ตัวประมวลผลกลาง (IC FPGA) จะทำหน้าที่ควบคุมและประมวลผลการทำงานของอุปกรณ์ต่างๆ ได้แก่หน่วยความจำข้อมูล, ตัวตรวจวัดอุณหภูมิ, ตัวติดต่อพอร์ตอนุกรม เมื่อเริ่มทำงานตัวควบคุมประมวลผลกลางจะทำการอ่านค่าอุณหภูมิจากตัวตรวจวัด

อุณหภูมิแล้วแปลงข้อมูลให้อยู่ในรูปแบบที่เหมาะสมแล้วส่งไปแสดงผลยัง 7-Segment และพอร์ตอนุกรม โดยมีฐานเวลาการทำงานจาก Oscillator เพื่อให้ได้เวลาที่แน่นอนและตรงกับความเป็นจริง

การติดต่อกับอุปกรณ์ต่างๆ ที่กล่าวมานั้นจะใช้รูปแบบการติดต่อแบบบัสข้อมูลอนุกรมเพื่อลดจำนวนของสายสัญญาณในระบบ เนื่องจากต้องการให้บอร์ดไม่ให้บอร์ดมีขนาดใหญ่เพื่อความสะดวกต่อการใช้งาน อีกทั้งอุปกรณ์ที่ใช้การรับส่งข้อมูลแบบอนุกรมเหล่านี้ นอกจากจะมีข้อดีในเรื่องของขนาดอุปกรณ์ที่เล็กแล้ว ในปัจจุบันยังได้รับการพัฒนาให้มีความสามารถในด้านการประหยัดพลังงาน คือ ใช้กระแสไฟฟ้าต่ำอีกด้วย

การใช้งานสามารถเลือกตัววัดอุณหภูมิได้ 2 แบบเพื่อให้สะดวกต่อการใช้งาน คือ 1. ตัววัดอุณหภูมิติดตั้งอยู่บนบอร์ด และ 2. ตัววัดอุณหภูมิแบบสายวัด โดยการเซต Selector DS1620 ที่ตัวบอร์ด



รูปที่ 5.3 แสดงบอร์ดวัดอุณหภูมิและตำแหน่งต่างๆ

โหมดการทำงานมี 2 โหมด

1. Real Times Mode จะเป็นการวัดอุณหภูมิแบบเวลาต่อเวลา (Real Time) ซึ่งโหมดการทำงานนี้จะเกิดขึ้นตลอดเวลาโดยแสดงผลออกมาทาง 7-Segment

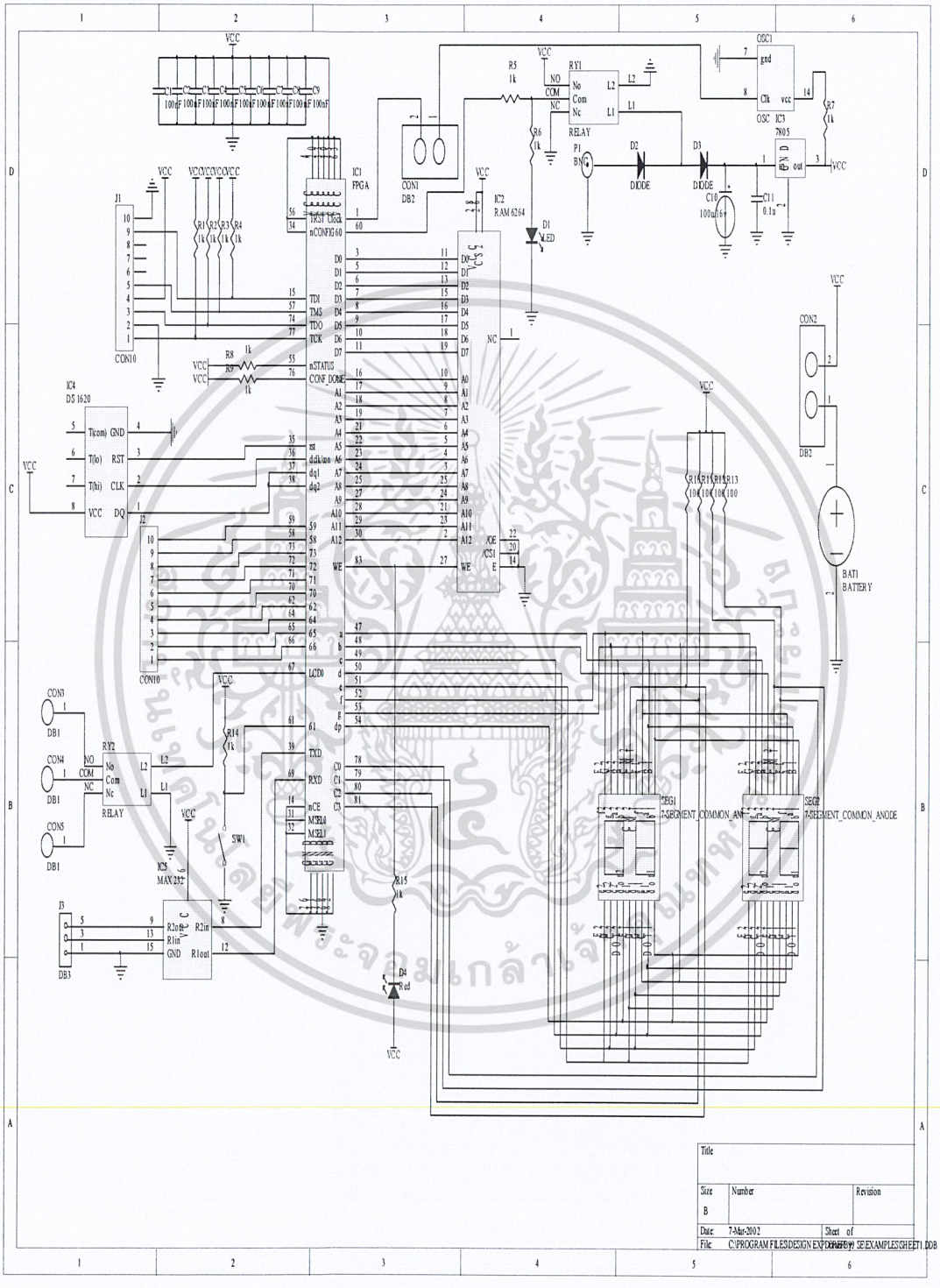
2. Record Times Mode จะเกิดขึ้นเมื่อ SW_Short หรือ SW_Long ถูกกด ซึ่งโหมดนี้จะเป็นการสั่งให้ข้อมูลอุณภูมิถูกบันทึกลงในหน่วยความจำ โดยการบันทึกข้อมูลนี้จะเป็นการบันทึกแบบซุ่มข้อมูล (Sampling Data) การซุ่มบันทึกข้อมูลนี้จะถูกแบ่งช่วงเวลาออกเป็น 2 แบบคือ แบบ Short Time จะใช้ระยะเวลาการซุ่มข้อมูล 1 ชุดต่อวินาที จากความจุของหน่วยความจำ (RAM) เท่ากับ 8 K-Byte ทำให้ระยะเวลาการบันทึกสูงสุดเท่ากับ 8192 วินาที หรือ ประมาณ 2 ชั่วโมง ส่วนแบบ Long Time จะใช้ระยะเวลาการซุ่มข้อมูล 2 ชุดต่อนาที ดังนั้นระยะเวลาที่จะใช้บันทึกข้อมูลแบบนี้เท่ากับ 68 ชั่วโมง

รายละเอียดของวงจร

วงจรสมบูรณ์ของบอร์ดวัดอุณหภูมิแสดงดังรูปที่ 5.4 ตัวควบคุมและประมวลผลส่วนกลางคือ IC FPGA เบอร์ EPF10K10LC84-3 ตระกูล Flex ของบริษัท Altera มีความจุเกทเท่ากับ 10,000 เกท 84 ขา มีขาให้ใช้งาน 59 ขาควบคุมอุปกรณ์ต่างๆ คือ

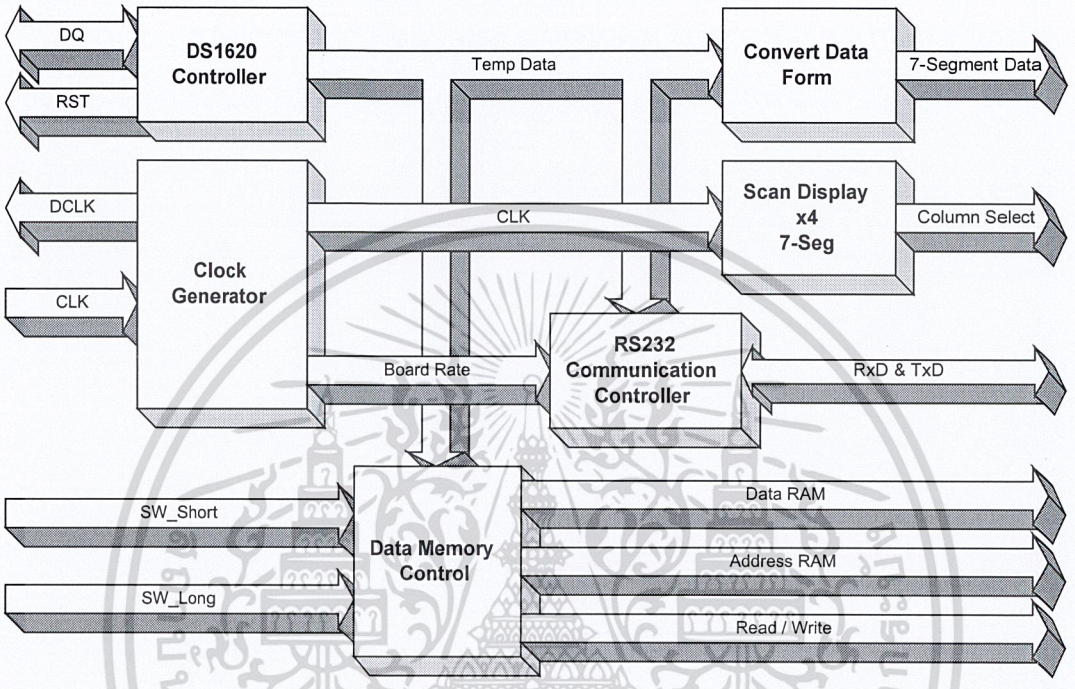
- หน่วยความจำ (RAM) ขนาด 8 K-Byte เบอร์ UT6264PC-70LL
- ตัววัดอุณหภูมิ เบอร์ DS1620 สามารถวัดอุณหภูมิได้ตั้งแต่ -55 ถึง +125 องศาเซลเซียส โดยจะแปลงข้อมูลที่วัดได้เป็นข้อมูลดิจิตอลขนาด 9 บิต สื่อสารข้อมูลผ่านการติดต่อแบบอนุกรมโดยใช้สาย 3 เส้น
- 7-Segment ใช้จำนวน 4 หลักด้วยกันเป็นแบบ Common Anode ใช้วิธีการสแกนหลักในการแสดงผล
- พอร์ตอนุกรม RS232 ใช้สำหรับติดต่อกับคอมพิวเตอร์เพื่อส่งข้อมูลไปพล็อตกราฟทางหน้าจอโดยใช้โปรแกรม Microsoft Visual C++
- สวิตช์และ LED ใช้เพื่อเลือกการซุ่มข้อมูลที่จะบันทึก และแสดงสถานะการทำงาน
- ไฟเลี้ยงวงจรใช้ไฟ DC 8-12 โวลต์ภายในบอร์ดมีไดโอดเรกติไฟร์ช่วยป้องกันการต่อไฟเลี้ยงผิดขั้วและมีเรกกูเลเตอร์เบอร์ 78L05 เพื่อสร้างแรงดันไฟเลี้ยง 5 โวลต์จ่ายให้กับวงจร
- ไฟสำรองจ่ายให้วงจร เนื่อง IC FPGA ในเบอร์จะใช้หน่วยความจำเป็นแบบ SDRAM ทำให้ต้องอาศัยไฟเลี้ยงตลอดเวลา จึงจำเป็นต้องมีไฟเลี้ยงสำรองจ่ายให้กับวงจรในขณะที่ไม่ได้จ่าย

ไฟเลี้ยงตามปกติ (ไม่ได้ใช้งาน) โดยแบตเตอร์รี่ที่ใช้กับ IC FPGA เบอร์ EPF10K10LC84-3 ตระกูล Flex ของบริษัท Altera มีขนาดแรงดัน 3.5 – 5.5 โวลต์



รูปที่ 5.4 แสดงวงจรรวมทั้งหมด

5.2 ส่วนของซอร์ฟแวร์จำลองการทำงาน

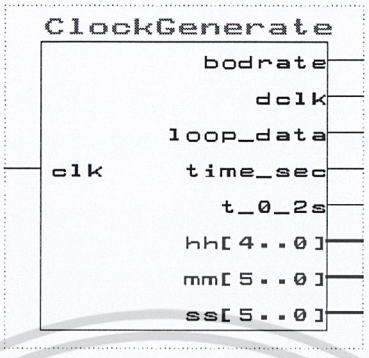


รูปที่ 5.5 Block Diagram การควบคุมการทำงาน

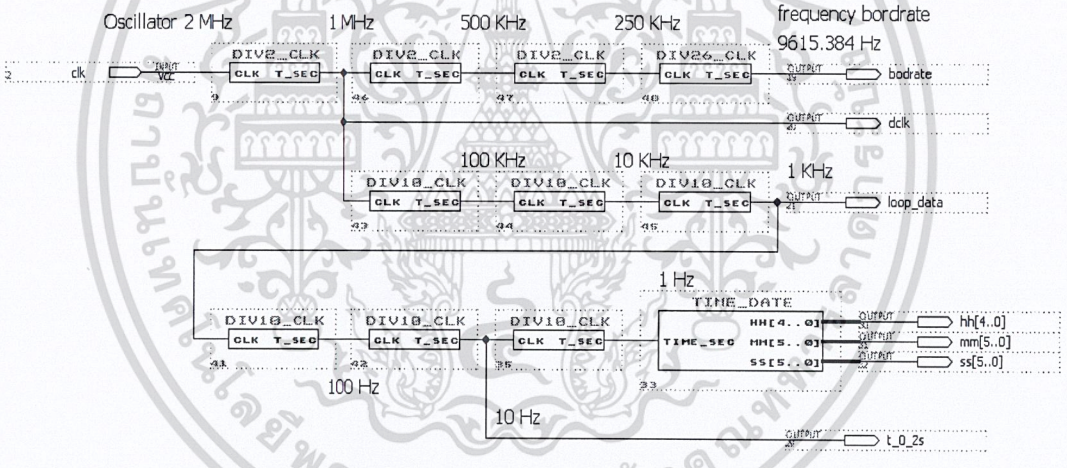
5.2.1 Clock Generator Block Diagram

ส่วนนี้จะเป็นส่วนที่สร้างสัญญาณความถี่ต่างๆ เพื่อใช้เป็นฐานเวลาในการอ้างอิงในส่วนของโปรแกรมควบคุมอื่นๆ โดยใช้หลักการหารความถี่จากสัญญาณนาฬิกาอ้างอิงขนาด 2 MHz โดยความถี่ที่สร้างมีดังนี้คือ

- Board rate ใช้สำหรับการติดต่อกับคอมพิวเตอร์แบบอนุกรมเท่ากับ 9600 Hz
- DCLK ใช้เป็นฐานเวลาในการติดต่อสื่อสารกับตัววัดอุณหภูมิแบบอนุกรม ซึ่งในที่นี้มีความถี่เท่ากับ 1 MHz
- Loop Data เป็นความถี่ที่ใช้กำหนดเวลาการสื่อสารข้อมูลต่อ 1 ชุดข้อมูล
- Time Sec เป็นความถี่ที่ใช้ซุ่มบันทึกข้อมูลทุก 1 วินาที
- t_0_2s เป็นความถี่ที่ใช้เป็นฐานเวลาในการบันทึกข้อมูล 1 ชุด



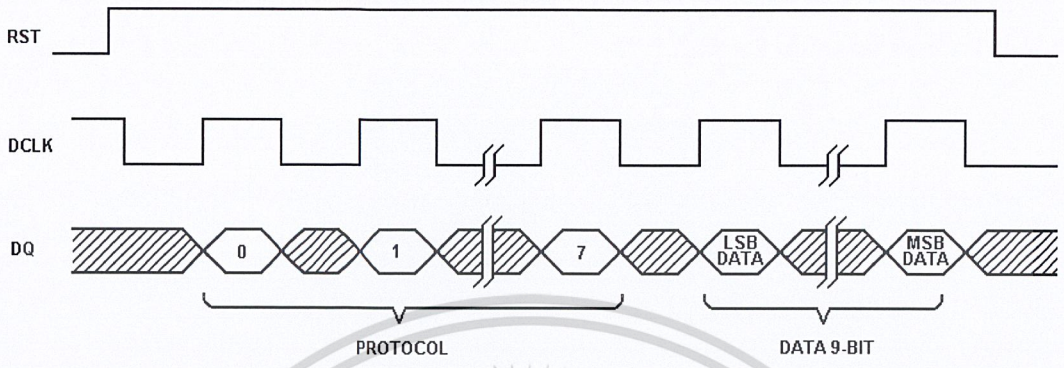
รูปที่ 5.6 Clock Generate Block Diagram



รูปที่ 5.7 โครงสร้างภายในของ Clock Generate Block Diagram

5.2.2 DS1620_Controller Block Diagram

เนื่องจากตัววัดอุณหภูมิที่ใช้เป็นเบอร์ DS1620 มีฟังก์ชันการทำงานส่งข้อมูลแบบอนุกรมใช้การควบคุมการทำงานให้วัดอุณหภูมิโดยสัญญาณ 3 ส่วน คือ DQ, DCLK, RST การสื่อสารข้อมูลมีรูปแบบดังรูปที่ 5.8 แบ่งเป็นส่วนคำสั่ง (Protocol) และส่วนของข้อมูล (Temperature Data) ขนาด 9 Bit คือ LSB DATA 8-BIT และ MSB DATA 1-BIT



รูปที่ 5.8 แสดงรูปแบบของการสื่อสารกับตัววัดอุณหภูมิ DS1620

การเซตโหมดให้ DS1620 ทำงานในโหมด Temperature Conversion ซึ่งเป็นโหมดสำหรับการวัดอุณหภูมิ สำหรับการเซตทำได้ดังนี้

Write Config [0Ch]

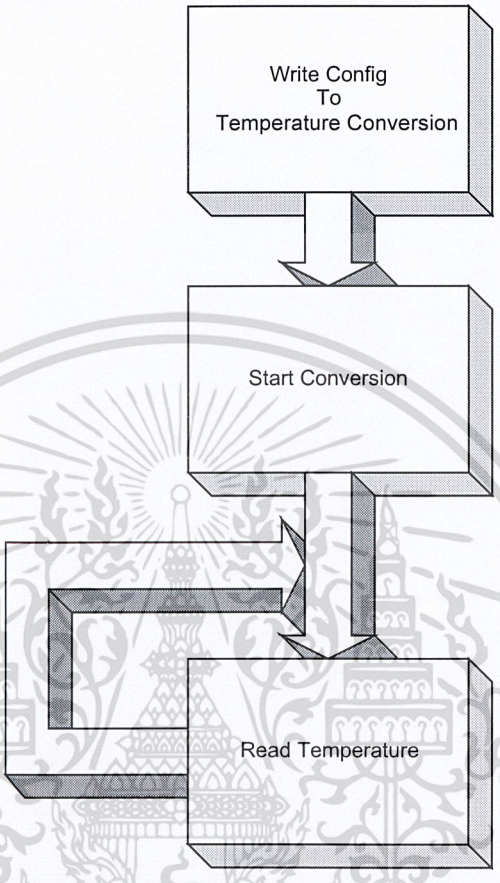
- Protocol = 0Ch "Write Config Command"
- Data = 00h "Set to Temperature Conversion Mod"

Start Convert T [EEh]

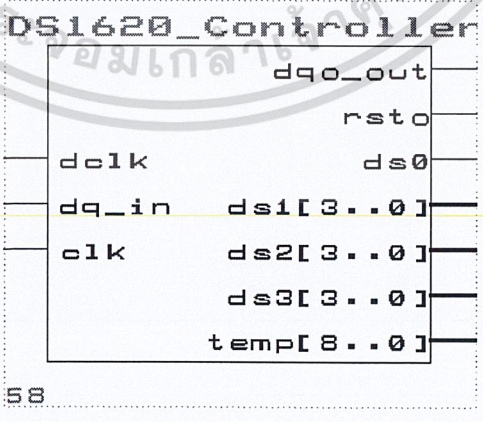
- Protocol = EEh "Start Conversion to Measuring Temperature"

Read Temperature [AAh]

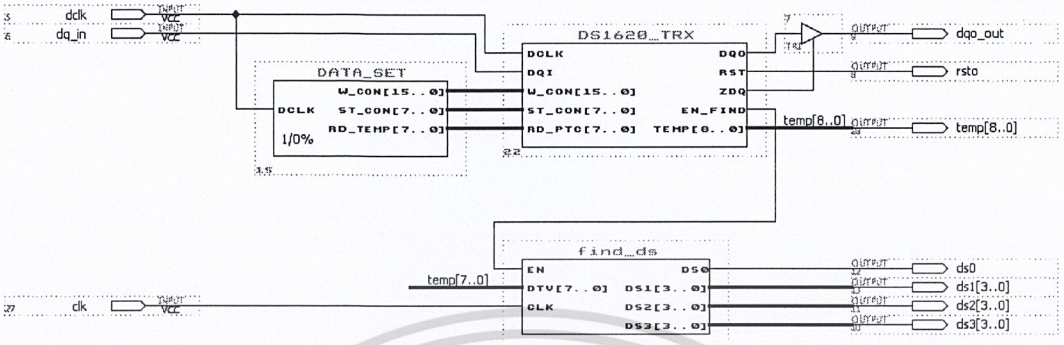
- Protocol = AAh "Read Temperature from DS1620"
- Data = Temperature Data



รูปที่ 5.9 แสดงขั้นตอนการเซตการทำงานของ DS1620



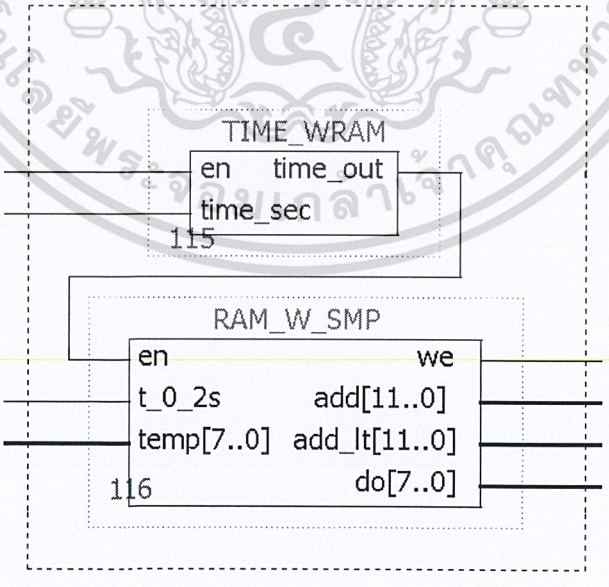
รูปที่ 5.10 DS1620_Controller Block Diagram



รูปที่ 5.11 โครงสร้างภายในของ DS1620_Controller Block Diagram

5.2.3 Data Memory Control Block Diagram

จะเป็นส่วนที่ใช้งานในโหมดการทำงานของโหมด Record Times Mode คือใช้เป็นส่วนการบันทึกข้อมูลและโหลดข้อมูลที่ได้ถูกซุ่มบันทึกไว้ การซุ่มบันทึกข้อมูลนั้นได้ถูกแบ่งการซุ่มข้อมูลออกเป็น 2 แบบคือ แบบ Short Time ใช้การซุ่มบันทึกข้อมูลในอัตรา 1 ครั้งต่อวินาทีและแบบ Long Time ใช้การซุ่มบันทึกข้อมูลในอัตรา 1 ครั้งต่อ 30 วินาที



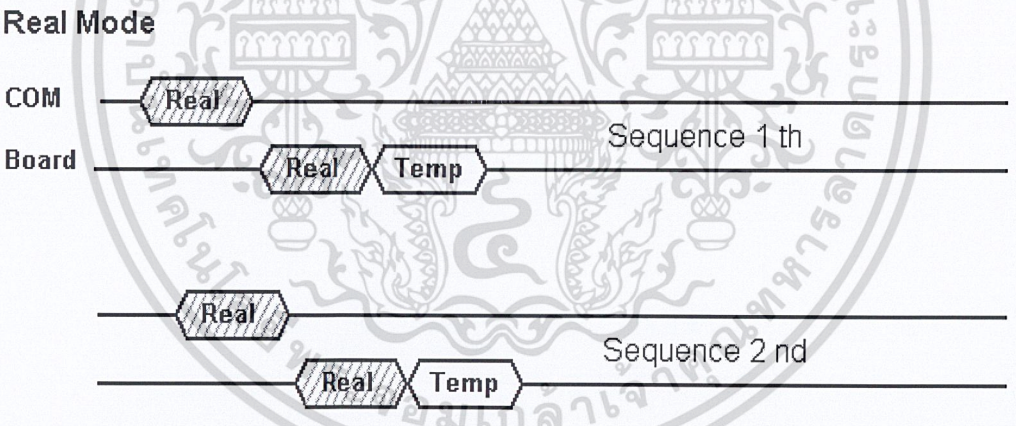
รูปที่ 5.12 แสดงโปรแกรม Data Memory Control

5.2.4 Communication Controller Block Diagram

ส่วนนี้เป็นโปรแกรมที่ใช้ควบคุมการทำงานการติดต่อกับคอมพิวเตอร์ผ่านพอร์ตอนุกรม (Series Port RS232) ซึ่งมีรูปแบบการสื่อสารดังนี้

- Baud rate = 9600 Hz
- Data = 8 Bit
- One Stop Bit
- No Parity Bit

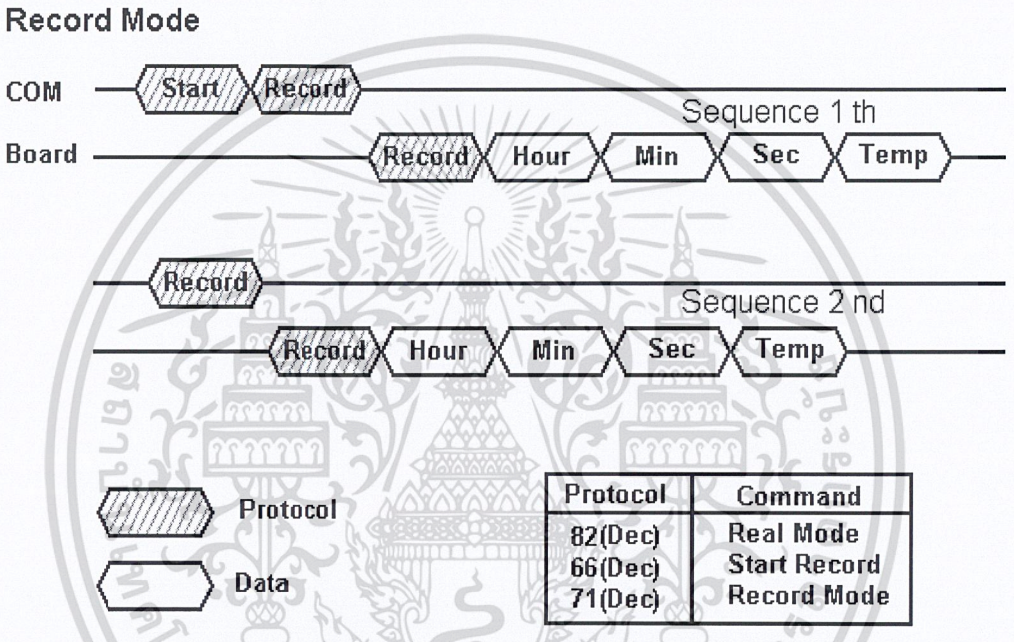
สำหรับวิธีการติดต่อสื่อสารกับคอมพิวเตอร์แบบReal Time Mode จะใช้รูปแบบดังรูปที่ 5.14 ซึ่ง Loop การทำงานจะเหมือนกันในทุกๆ Loop



รูปที่ 5.13 แสดงวิธีการติดต่อสื่อสารกับคอมพิวเตอร์แบบ Real Time Mode

สำหรับวิธีการติดต่อสื่อสารกับคอมพิวเตอร์แบบ Record Time Mode จะใช้รูปแบบดังรูปที่ 5.15 ซึ่ง Loop เริ่มต้นจะต้องส่งคำสั่งการเริ่มต้นการโหลดข้อมูลไปด้วยเพื่อที่จะเซต Address เริ่มต้นให้กับหน่วยความจำ (RAM) แล้วจากนั้นใน Loop ต่อไปจะเป็นเพียงคำสั่งการโหลดข้อมูลเท่านั้นเพราะบอร์ดวัดอุณหภูมินั้นจะเริ่มต้นการนับตำแหน่ง Address เองโดยอัตโนมัติ และในโหมดของ Record Times Mode นอกจากบอร์ดวัดอุณหภูมิจะส่งข้อมูลของอุณหภูมิที่ถูกบันทึกไว้แล้วนั้นบอร์ดวัด

อุณหภูมิจะทำการส่งข้อมูลของเวลาที่ได้ถูกบันทึกในครั้งแรกด้วยเพราะการโหลดข้อมูลมานั้นเป็นช่วงเวลาที่จะเกิดขึ้นเมื่อใดก็ได้ โดยเวลาที่ถูกบันทึกนี้จะได้มาจากโปรแกรม Clock Generator ซึ่งอยู่บนบอร์ดวัดอุณหภูมิ



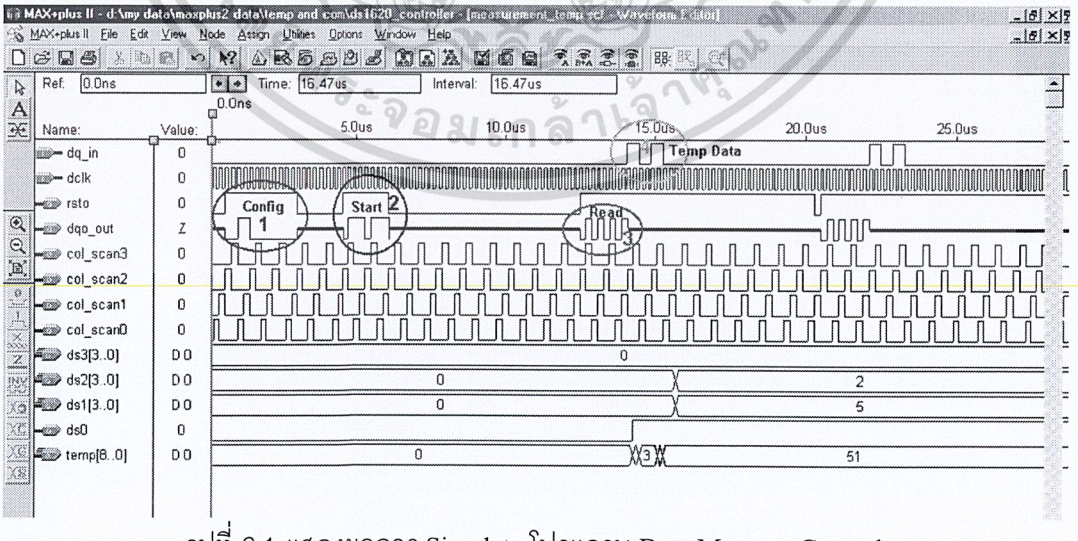
รูปที่ 5.14 แสดงวิธีการติดต่อสื่อสารกับคอมพิวเตอร์แบบ Real Time Mode

บทที่ 6
การทดลอง

จากการที่ได้ศึกษาในบทที่ผ่านมา สามารถทำการทดลองการทำงานของบอร์ดวัดอุณหภูมิแต่ละส่วน จากนั้นจะนำไปทดสอบการใช้งานวัดอุณหภูมิกับห้องและอุปกรณ์ต่างๆ แล้วแสดงผลออกที่จอคอมพิวเตอร์โดยผ่านพอร์ตอนุกรมซึ่งเชื่อมต่อกับบอร์ดวัดอุณหภูมิ

6.1 ผลการทดลองโปรแกรมควบคุมอุปกรณ์วัดตัวอุณหภูมิ

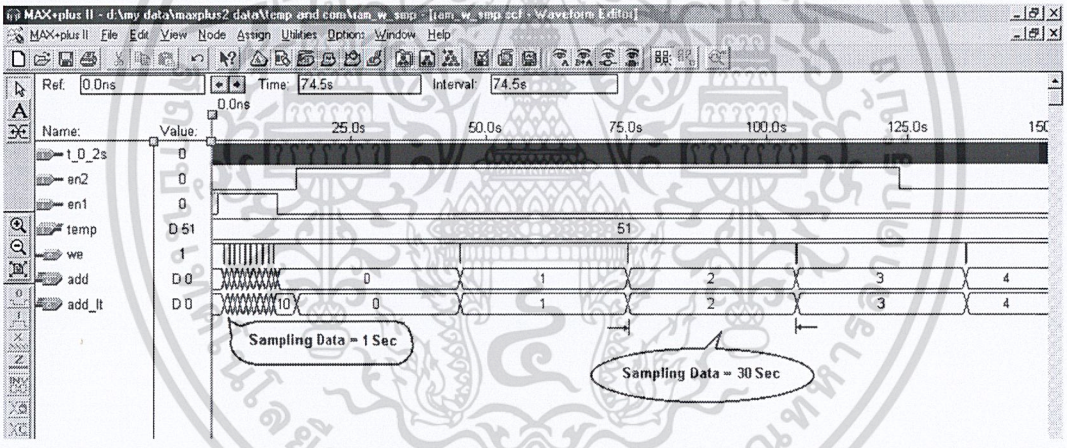
จากกราฟเป็นผลการทดลองการ Simulate โปรแกรม DS1620_Controller โดยเริ่มต้นนั้นจะต้องทำการเซตค่าและตั้งโหมดให้กับอุปกรณ์วัดอุณหภูมิ DS1620 ดังที่ได้กล่าวมาแล้วในบทที่แล้วเพื่อกำหนดการทำงานให้ทำการอ่านอุณหภูมิออกมา ซึ่งผลการ Simulate ได้ดังรูปที่ 6.1 ลำดับแรก (วงกลมที่ 1) เป็นการเซตค่ากำหนดให้ตัววัดอุณหภูมิทำงานในโหมดของ Temperature Conversion คือส่งค่าคำสั่ง Protocol = “0Ch” และส่งข้อมูลเซตโหมดการทำงานเท่ากับ = “00h” อันดับที่สอง (วงกลมที่ 2) เป็นส่วนคำสั่งให้ตัวอุปกรณ์วัดอุณหภูมิเริ่มต้นการวัดอุณหภูมิโดยส่งค่า Protocol = “EEh” และส่วนที่สาม (วงกลมที่ 3 และวงกลมที่ 4) จะเป็นชุดคำสั่งการอ่านอุณหภูมิที่ตัววัดอุณหภูมิทำการวัดออกมาได้ โดยส่งค่า Protocol = “AAh”



รูปที่ 6.1 แสดงผลการ Simulate โปรแกรม Data Memory Control

6.2 ผลการทดลองโปรแกรมควบคุมการอ่านและบันทึกหน่วยความจำ RAM

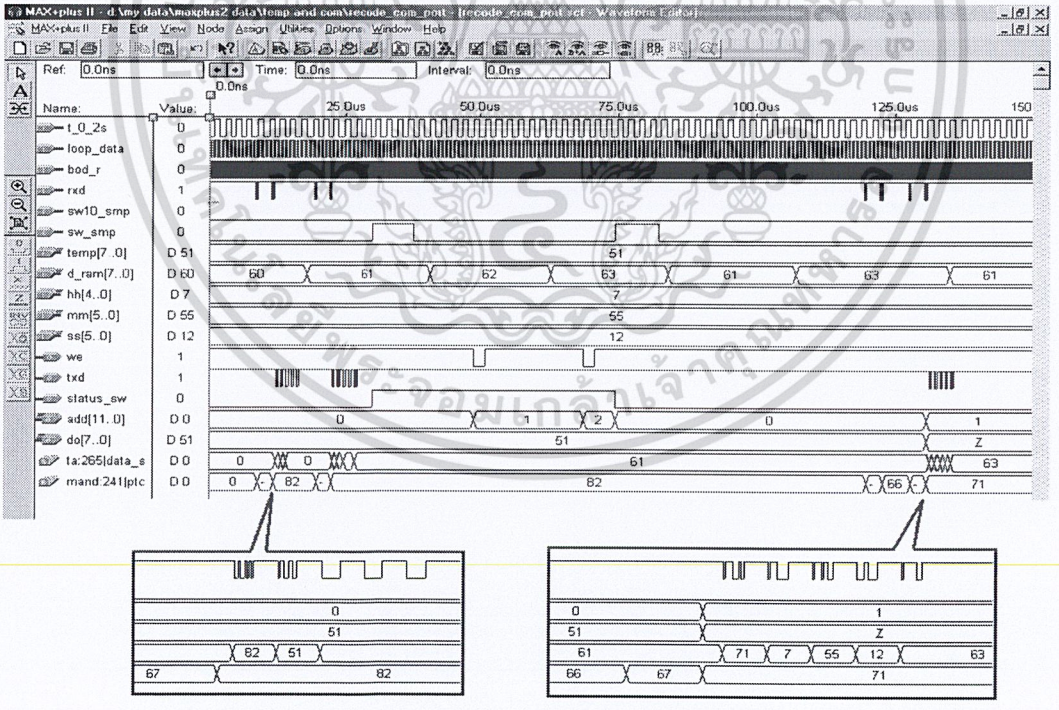
การอ่านและบันทึกหน่วยความจำ RAM นี้สามารถทำได้โดยการส่งสัญญาณไปควบคุมที่ขา R/W และการควบคุมการส่งข้อมูล (D0-D7) และตำแหน่ง (A0 – A12) จะเกิดขึ้นในโหมดการทำงานของ Record Times Mode โดยแบ่งออกเป็นแบบ Short Time และแบบ Long Time ดังรูปที่ 6.2 เป็นผลการทดลอง Simulate ทั้ง 2 แบบ คือ ช่วงแรกเป็นแบบ Short Time ที่ใช้เวลาในการซุ่มบันทึกข้อมูลเท่ากับ 1 ครั้งต่อวินาที (เมื่อสัญญาณ en1 มีสภาวะเป็น High) และช่วงที่สองเป็นแบบ Long Time ที่ใช้เวลาในการซุ่มบันทึกข้อมูลเท่ากับ 1 ครั้งต่อ 30 วินาที (เมื่อสัญญาณ en2 มีสภาวะเป็น High)



รูปที่ 6.2 แสดงผลการ Simulate โปรแกรม Memory Control

6.3 ผลการทดลองโปรแกรมควบคุมการสื่อสารข้อมูลกับคอมพิวเตอร์แบบอนุกรม

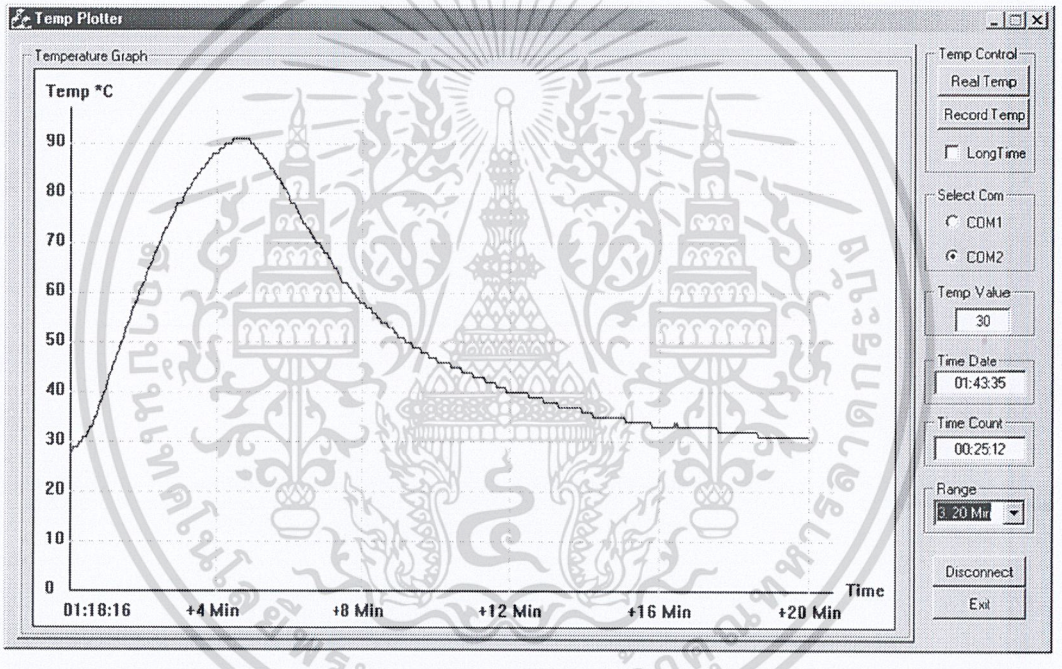
การสื่อสารกับคอมพิวเตอร์เพื่อส่งข้อมูลนี้จะทำงานอยู่ด้วยกัน 2 โหมดการทำงาน คือ การทำงานในโหมดของ Real Times Mode ซึ่งคอมพิวเตอร์จะต้องส่งสัญญาณมาบอกกับบอร์ดก่อนโดยใช้คำสั่งของ Real Times Mode หรือ Protocol = “82(Dec)” จากนั้นบอร์ดจะทำการแปลชุดคำสั่งแล้วจะส่งข้อมูลที่อ่านได้จากตัววัดอุณหภูมิส่งผ่านพอร์ตอนุกรมไปให้คอมพิวเตอร์ และจะวนทำงานอย่างนี้ไปเรื่อยจนกระทั่งถูกสั่งให้ยกเลิกการอ่าน สำหรับโหมดการทำงานในโหมดของ Record Times Mode คอมพิวเตอร์ก็จะส่งสัญญาณมาถามก่อนเช่นกันโดยใช้คำสั่ง Protocol = “71(Dec)” แล้วบอร์ดวัดอุณหภูมิก็จะส่งข้อมูลที่อ่านได้จะหน่วยความจำ RAM ส่งผ่านออกไปให้คอมพิวเตอร์ โดยลำดับแรกนั้นคอมพิวเตอร์จะต้องส่งชุดคำสั่งการเริ่มต้นการอ่านข้อมูลจากหน่วยความจำมาด้วยใช้ Protocol = “66(Dec)” ถ้าไม่มีการส่งข้อมูลเริ่มต้นการอ่าน บอร์ดก็จะส่งข้อมูลชุดคำสั่งการทำงานผิดพลาดออกมา Protocol = “79(Dec)” คอมพิวเตอร์ก็จะไม่ยอมรับข้อมูล



รูปที่ 6.3 แสดงผลการ Simulate โปรแกรม Communication Controller

6.4 ผลการทดลองการทำงานในโหมดของ Real Times Mode กับคอมพิวเตอร์

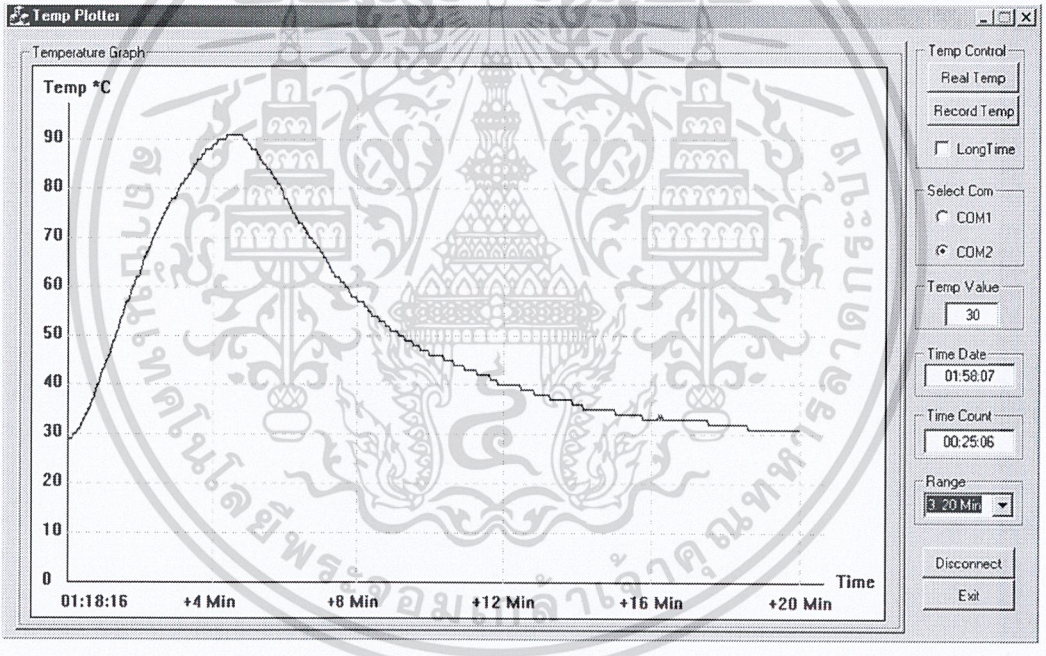
จากผลการทดลองเป็นการทดลองใช้บอร์ดวัดอุณหภูมิร่วมกับคอมพิวเตอร์ในการวัดอุณหภูมิของหัวแร้งตั้งแต่เริ่มเสียบปลั๊กจนความร้อนถึงประมาณ 90 °C แล้วจะถอดปลั๊ก ซึ่งผลการวัดนี้แสดงออกมาได้โดยการใช้โปรแกรมประยุกต์ที่เขียนขึ้นด้วย Microsoft Visual C++ ดังรูปที่ 6.4 ใช้เวลาไป 20 นาที



รูปที่ 6.4 แสดงผลการวัดอุณหภูมิของหัวแร้งแบบ Real Times

6.5 ผลการทดลองการทำงานในโหมดของ Real Times Mode แบบ Short Time กับคอมพิวเตอร์

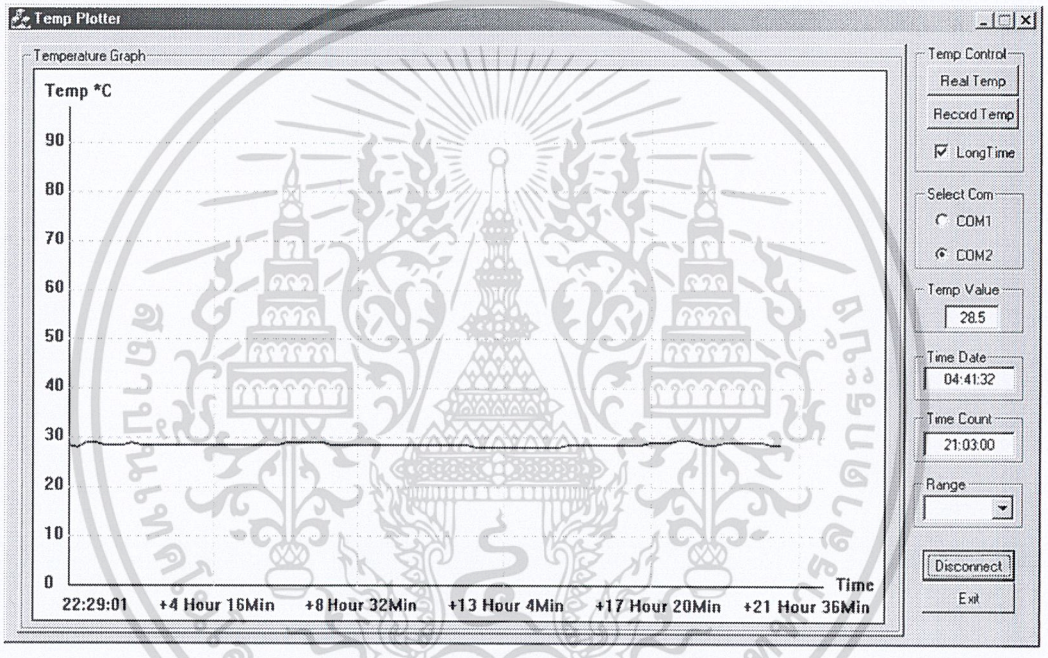
จากผลการทดลองเป็นการทดลองใช้บอร์ดวัดอุณหภูมิร่วมกับคอมพิวเตอร์ในการโหลดข้อมูลจากหน่วยความจำที่ได้ถูกบันทึกในเวลาเดียวกันกับตอนวัดอุณหภูมิในแบบ Record Times Mode แบบ Short Time วัดอุณหภูมิของหัวแร้งตั้งแต่เริ่มเสียบปลั๊กจนความร้อนถึงประมาณ 90 °C แล้วจะถอดปลั๊ก ดังรูปที่ 6.5 ใช้เวลาไป 20 นาที



รูปที่ 6.5 แสดงผลการวัดอุณหภูมิแบบ Real Times Mode แบบ Short Time

6.6 ผลการทดลองการทำงานในโหมดของ Real Times Mode แบบ Long Time กับคอมพิวเตอร์

เป็นการวัดอุณหภูมิในแบบ Real Times Mode แบบ Long Time โดยได้ทำการวัดอุณหภูมิในห้องพักตั้งแต่เวลา 22:29:31 จนกระทั่งถึงเวลาประมาณ 19 นาฬิกา ซึ่งผลการวัดแสดงได้ดังรูปที่ 6.6



รูปที่ 6.5 แสดงผลการวัดอุณหภูมิแบบ Real Times Mode แบบ Long Time

บทที่ 7
สรุปผลการทดลอง

สรุปผลการทดลอง

การทำงานแบบ FPGA ซึ่งใช้ภาษา VHDL เป็นส่วนบรรยายพฤติกรรมการทำงาน และจากการทดลองพบว่าส่วนต่างๆ ที่เขียนด้วยภาษานี้มีความยืดหยุ่นสูงในการออกแบบและสามารถที่จะเข้าใจการทำงานได้ง่าย อีกทั้งในการออกแบบมีความคล่องตัวในการแก้ไขระบบการทำงาน เพราะเราสามารถที่จะทำการทดสอบระบบการทำงานได้โดยการ Simulate ด้วยโปรแกรมเพื่อหาสาเหตุความผิดพลาดต่างๆ ที่เราจะทำการแก้ไขระบบ

ปริญญานิพนธ์ฉบับนี้เสนอผลงานเกี่ยวกับการประยุกต์ใช้งาน FPGA กับอุปกรณ์ตรวจสอบอุณหภูมิ โดยสามารถออกแบบ FPGA เพื่อควบคุมการทำงานของอุปกรณ์ตรวจสอบอุณภูมิ ซึ่งบอร์ดวัดอุณหภูมิที่ถูกออกแบบมานี้ สามารถนำไปใช้งานเป็นเครื่องมือวัดอุณหภูมิได้จริง คือ เมื่อทำการวัดอุณหภูมิจากตัววัดอุณหภูมิ (IC DS1620) แล้วนำผลที่ได้จากการวัดไปแสดงผลที่ 7-Segment และแสดงผลที่หน้าจอคอมพิวเตอร์ โดยผ่านพอร์ตอนุกรม RS-232 ซึ่งผลที่วัดได้ยังสามารถเก็บบันทึกไว้ในหน่วยความจำ RAM (สามารถบันทึกผลการวัดได้สูงสุด 68 ชั่วโมง) แล้วทำการพล็อตอุณหภูมิที่วัดได้เทียบกับเวลา โดยโปรแกรมประยุกต์ที่เขียนขึ้นมาด้วยภาษา Microsoft Visual VC++

ปัญหาและแนวทางแก้ไข

ในส่วนของฮาร์ดแวร์ เนื่องจากการ Wire lab บนบอร์ดวัดอุณหภูมิยังถูกออกแบบไม่ดีเท่าที่ควร ดังนั้นควรวางอุปกรณ์ให้เป็นระเบียบและชิดกันมากที่สุด

ในส่วนของซอร์ฟแวร์พยายามค้นคว้าหาตำราหรือเอกสารที่เกี่ยวข้องกับเทคนิคการเขียนโปรแกรม

การพัฒนาโครงการ

โครงการที่ทำขึ้นมานี้สามารถทำงานได้ตามขีดความสามารถที่กำหนดไว้ แต่ยังสามารถที่จะพัฒนาความสามารถการทำงานได้อีก

1. เพิ่มระยะเวลาการบันทึกให้ยาวนานขึ้น โดยการขยายหน่วยความจำเพิ่มขึ้น
2. เพิ่มความสามารถในการแสดงผลที่ต่อเนื่องทั้งฮาร์ดแวร์และซอร์ฟแวร์ให้สัมพันธ์กัน

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี โดยได้รับการสนับสนุนและความช่วยเหลือการแนะนำการให้คำปรึกษาจากคณาจารย์ประจำภาควิชาอิเล็กทรอนิกส์โดยเฉพาะอย่างยิ่งจากอาจารย์ รศ.ดร.มนัส สังวรศิลป์ ซึ่งเป็นอาจารย์ที่ปรึกษาปริญญานิพนธ์ ขอขอบพระคุณอาจารย์ทุกท่านที่ให้ความช่วยเหลือและคำปรึกษาทำให้ปริญญานิพนธ์ฉบับนี้เสร็จสมบูรณ์และสำคัญที่สุดขอกราบขอบพระคุณคุณพ่อคุณแม่ผู้บังเกิดเกล้าผู้เป็นแรงสนับสนุนอันยิ่งใหญ่ทั้งกำลังใจ กำลังทรัพย์และเป็นผู้ให้ตลอดมา

หนังสืออ้างอิง

Scarpino, Frank A. VHDL and AHDL digital system implementation Upper Saddle River, NJ : Prentice Hall PTR, c1998

Skahill, Kevin VHDL for programmable Imprint Reading, MA : Addison-Wesley, c1996

Rushton, Andrew VHDL for logic synthesis Edition 2nd Chichester : John Wiley, c1998