เครื่องแสดงผลระดับสัญญาณเสียง

Sound Level Monitoring

โดย

นายประภูศักดิ์ คุณวัลลี

นายวันชัย ราวีศรี

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

เครื่องแสดงผลระดับสัญญาณเสียง

Sound Level Monitoring

โดย

นายประภัศกดิ์ คุณวัลลี รหัส 44015204

นายวันชัย ราวีศรี รหัส 44015214

อาจารย์ที่ปรึกษา

รศ.ดร.มนัส สังวรศิลป์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

ปริญญานิพนธ์ ปีการศึกษา 2546

ภาควิชาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เครื่องแสดงผลระดับสัญญาณเสียง

(Sound Level Monitoring)

ผู้จัดทำ

นายประภูศักดิ์ คุณวัลลี 44015204

นายวันชัย ราวีศรี 44015214

.....

(รศ.ดร.มนัส สังวรศิลป์)

อาจารย์ที่ปรึกษา

เรื่อง เครื่องแสดงผลระดับสัญญาณเสียง

(Sound Level Monitoring)

ผู้จัดทำ

นายประภูศักดิ์ คุณวัลลี 44015204

นายวันชัย ราวีศรี 44015214

โครงงานนี้ได้รับการตรวจสอบแล้ว พร้อมที่จะทำการสอบได้

.....

(รศ. ดร. มนัส สังวรศิลป์)

อาจารย์ที่ปรึกษา

เครื่องแสดงผลระดับสัญญาณเสียง

นายประภูศักดิ์ คุณวัลลี 44015204
นายวันชัย ราวีศรี 44015214
รศ.ดร.มนัส สังวรศิลป์
อาจารย์ที่ปรึกษา
ปีการศึกษา 2546

บทคัดย่อ

ปัจจุบันปัญหาด้านเสียงเข้ามามีผลกระทบต่อสุขภาพชีวิตในการทำงานในโรงงาน กล่าวคือประสาทหูหากได้รับปริมาณเสียงที่มากและนานเกินไป จะมีผลต่อสุขภาพกายและใจซึ่งทำให้เกิดผลเสียขึ้นหลายด้านอาทิเช่น ความสามารถในการทำงานลดลง มีความเครียดอารมณ์แปรปรวน ด้วยเหตุนี้เครื่องวัดระดับสัญญาณเสียงจึงได้เกิดขึ้นเพื่อวัดระดับสัญญาณเสียงในช่วงเวลาต่างๆ โดยแสดงผลบนจอคอมพิวเตอร์ มีหลักการคือรับสัญญาณอินพุทเข้ามาใช้วงจรฟิลเตอร์เลือกเฉพาะความถี่ในย่านที่ต้องการแล้วนำมาแปลงเป็นสัญญาณดิจิตอลด้วยวงจร Analog to Digital จากนั้นนำ FPGA มาประยุกต์ใช้งานเพื่อประมวลผลหาระดับความสูง นำระดับความสูงที่ได้นั้นแสดงผลออกจอคอมพิวเตอร์ผ่านทางการสื่อสารแบบอนุกรม

SOUND LEVEL MONITORING

Mr.Praposak Kunvallee

Mr.Wanchai Rawesri

Advisor

Assoc. Prof. Dr. Manas Sangworasil

Academic year 2546

Abstract

In the present pollution of sound have effect to health at the factory. If the ear to receive sound quantity over abundant then it reduce ability of work and have serious. Therefore the display level of sound is design to display level of sound at the time that want to see on screen of computer. It receive signal to filter circuit for want to choose frequency then convert signal to digital by A/D circuit for FPGA to process signal to find maximum amplitude and display level of sound on computer through serial port. And then design circuit in MAX+PLUS II program. Use VHDL language for design and control FPGA

กิตติกรรมประกาศ

ในการจัดทำโครงงานในครั้งนี้ที่สำเร็จลุล่วงได้ด้วยดีนั้น ต้องขอขอบคุณ รศ.ดร. มนัส สังวรศิลป์ อาจารย์ที่ปรึกษา ที่คอยให้คำแนะนำแนวความคิดในการทำงานแต่ละขั้นตอน รวมทั้งตรวจแก้ไขข้อบกพร่อง และให้ข้อเสนอแนะที่เป็นประโยชน์ต่างๆ ทำให้โครงงานนี้สำเร็จลุล่วงมาได้ด้วยดี และขอขอบคุณ พี่ศุภชัย(แม็ก) ที่ได้แนะนำวิธีการศึกษาค้นคว้า และให้การช่วยเหลือเป็นอย่างดีมาตลอด รวมถึงเพื่อนๆและพี่ๆทุกคนที่ให้กำลังใจให้คำปรึกษาและคอยช่อยเหลือจนกระทั่งปริญญานิพนธ์นี้เสร็จสมบูรณ์ได้

ขอบพระคุณบิดามารดาเป็นที่ตั้ง ขอบพระคุณท่านอาจารย์ทุกท่านที่ได้ถ่ายทอดแนะแนวทางการศึกษา ขอบคุณพี่ ขอบใจน้อง ขอบใจเพื่อน ขอบใจตัวเอง

ตลอดทั้งทุกท่านที่ยังไม่ได้เอ่ยนามในที่นี้ที่มีส่วนช่วยให้ปริญญานิพนธ์นี้สำเร็จด้วยดี ทางคณะผู้จัดทำต้องขอขอบพระคุณไว้ ณ. โอกาสนี้

สารบัญ

เรื่อง **หน้า**

สารบัญ

บทที่ 1
บทนำ

1.1 ความเป็นมา

โลกเดินหน้าด้วยกงล้อของเทคโนโลยี และมิยอมรั้งรอว่าคนในประเทศโลกที่สามอย่างประเทศไทยจะเดินก้าวทันหรือไม่ ในช่วงหลายปีที่ผ่านมา แวดวงคนทำงานด้านวิทยาศาสตร์เทคโนโลยีและวิศวกรรม ต่างประจักษ์ถึงการเคลื่อนไหวเปลี่ยนแปลงที่เกิดขึ้นตลอดเวลา ในบางคราวนับเนื่องเพียงชั่วโมงเท่านั้นความเปลี่ยนแปลงก็เกิดขึ้น แน่นอน...เทคโนโลยีคือส่วนหนึ่งในการพัฒนาที่เราหลีกเลี่ยงไม่ได้ เพราะสิ่งนี้เป็นปัญญาประดิษฐ์ที่เข้ามาอำนวยความสะดวกให้กับชีวิตมนุษย์ แต่การพัฒนามิใช่มีเพียงด้าน และด้านที่นำมาใช้นั้นต้องสอดคล้องต่อชีวิต หรือให้ความหมายแก่ชีวิตมนุษย์มากขึ้น

ในระบบการบริหารงานคุณภาพ การประกันคุณภาพด้านสิ่งแวดล้อมหรือที่เรียกว่า ISO14000 นั้นประกอบด้วยองค์ประกอบหลายอย่าง ไม่ว่าจะเป็นการประกันคุณภาพด้านอากาศ การประกันคุณภาพด้านน้ำ การประกันคุณภาพด้านสุขอนามัย หรือการประกันคุณภาพด้านเสียงเป็นต้น

โดยเฉพาะอย่างยิ่งปัญหาด้านเสียงเข้ามามีผลกระทบต่อสุขภาพชีวิตในการทำงานในโรงงาน กล่าวคือประสาทหูหากได้รับปริมาณเสียงที่มากและนานเกินไป จะมีผลต่อสุขภาพกายและใจซึ่งทำให้เกิดผลเสียขึ้นหลายด้านอาทิเช่น ความสามารถในการทำงานลดลง มีความเครียดอารมณ์แปรปรวน ด้วยเหตุนี้เครื่องวัดระดับสัญญาณเสียงจึงได้เกิดขึ้นเพื่อวัดระดับสัญญาณเสียงในช่วงเวลาต่างๆ

การวัดปริมาณของเสียงนั้นจะนิยมวัดกันในหน่วยเดซิเบลเป็นมาตรฐาน ซึ่งการใช้งานจะเป็นลักษณะเครื่องพกพาเสียส่วนใหญ่ ดังนั้นโครงงานนี้จึงได้ออกแบบการวัดปริมาณของเสียงให้สามารถแสดงที่หน้าจอคอมพิวเตอร์ได้เพื่อง่ายแก่การเก็บข้อมูลนำไปเข้าระบบการประกันคุณภาพด้านเสียงในการประกันคุณภาพด้านสิ่งแวดล้อม ISO14000

1.2 วัตถุประสงค์ของโครงงาน

1. สามารถออกแบบเป็นขั้นตอนที่ปฏิบัติขึ้นเพื่อเป็นการทดลองในวิชาโครงงานวิศวกรรม
2. สามารถควบคุมการออกแบบ โดยใช้ภาษา VHDL เพื่อใช้งานกับเอฟพีจีเอได้
3. สามารถเรียนรู้และใช้งานโปรแกรม MAX+PLUS II ในการสร้างวงจรทางลอจิกด้วยภาษา VHDL และสามารถสังเคราะห์วงจรที่สร้างขึ้นมาได้
4. สามารถออกแบบวงจรอิเล็กทรอนิกส์พื้นฐาน เพื่อนำไปสู่การพัฒนาในขั้นสูงต่อไป
5. เพื่อให้สามารถใช้งานวงจรที่ออกแบบมากับบอร์ด MASTER FLEX – A01 ได้

1.3 ขอบเขตการดำเนินงาน

1. ศึกษาการใช้เอฟพีจีเอ (MASTER FLEX–A01) ในการรับส่งข้อมูลแบบอนุกรม
2. ใช้ VHDL ในการออกแบบการบรรยายเชิงพฤติกรรมเลียนแบบการทำงานของ วงจรอิเล็กทรอนิกส์
3. สัญญาณจากแหล่งกำเนิดจะผ่านวงจรฟิลเตอร์ เพื่อกรองเอาเฉพาะความถี่ที่อยู่ในย่านที่ต้องการมาใช้งาน
4. สัญญาณที่ใช้งานดังกล่าว จะถูกแปลงให้เป็นสัญญาณ Digitalขนาด 8 bit ด้วยวงจร Analog to Digital จากนั้นจะไปเข้า FPGA เพื่อประมวลผล
5. การใช้ FPGA ส่งข้อมูลไปยัง Computer เพื่อให้หน้าจอ Computer แสดงผลออกมาเป็นลำดับความแรงของสัญญาณ โดยพลอตกราฟด้วยโปรแกรม Visual Basic
7. หน้าจอ Computer แสดงเป็นระดับของสัญญาณออกมา

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. เข้าใจถึงทฤษฎีต่างๆ ในการออกแบบของวงจรอิเล็กทรอนิกส์โดยโปรแกรม MAX+PLUS II
2. เข้าใจในการเขียนภาษา VHDL บนโปรแกรมMAX+PLUS II กับบอร์ดเอฟพีจีเอ (MASTER FLEX–A01)
3. ได้เรียนรู้ถึงระบบสัญญาณของวงจรการทำงานอิเล็กทรอนิกส์ในส่วนต่างๆ
4. เข้าใจในการนำผลของสัญญาณไปใช้งาน โดยมีการอินเตอร์เฟสพอร์ตอนุกรมได้
5. มีความรู้ความเข้าใจเกี่ยวกับภาษา VHDL และ การใช้งาน FPGA
6. ช่วยให้เราเสริมทักษะในการปฏิบัติงานจริง
7. ได้รู้จักวางแผนงาน ในการปฏิบัติงานที่เป็นระบบมากยิ่งขึ้น
8. รู้จักแก้ไขปัญหาต่าง ได้ด้วยตัวเองหรือขอคำปรึกษาจากอาจารย์และพี่ๆได้ดี
9. ส่งเสริมความคิดที่เป็นระบบ ระเบียบมีแบบแผนมากยิ่งขึ้น
10. เป็นแนวทางเพื่อนำไปสู่การศึกษาค้นคว้าวิจัยในระดับสูงต่อไป

บทที่ 2
วงจรกรองความถี่ หน่วยความดังของเสียงและไมโครโฟน
(Filter Circuit Volume Unit of Sound and Microphone)

2.1 วงจรกรองความถี่แบบแอคทีฟเบื้องต้น

วงจรกรองความถี่(Filter)สามารถแบ่งออกเป็น 2 แบบใหญ่ ๆ คือ แบบพาสซีฟ(Passive filters) และแอคทีฟ(Active filters) วงจรกรองความถี่เป็นวงจรที่สามารถทำหน้าที่เลือกความถี่ที่ต้องการหรือตัดความถี่ที่ไม่ต้องการออกก็ได้ การใช้งานวงจรกรองความถี่สามารถใช้กรองสัญญาณรบกวน หรือกรองเอาสัญญาณข่าวสารออกมาจากคลื่นพาห์ในระบบวิทยุ ดังนั้นอุปกรณ์ที่นำมาใช้ในวงจรกรองความถี่ ถ้าเป็นแบบพาสซีฟจะใช้ตัวต้านทาน ตัวเก็บประจุ ร่วมกับอุปกรณ์ที่สามารถทำการขยายสัญญาณ เช่น ออปแอมป์ ทำให้วงจรกรองความถี่แบบแอคทีฟมีข้อดีกว่าแบบพาสซีฟ คือ

-ไม่มีการสูญเสียของสัญญาณเนื่องจากออปแอมป์สามารถทำการขยายสัญญาณเพื่อชดเชยการลดทอนของสัญญาณได้

-ราคาถูก โดยเฉลี่ยแล้ววงจรกรองความถี่แบบแอคทีฟจะมีราคาถูกกว่าแบบพาสซีฟเนื่องจากตัวเหนี่ยวนำที่ใช้ในแบบพาสซีฟมีราคาแพง และสร้างได้ยากกว่า

-การปรับค่า วงจรกรองความถี่แบบแอคทีฟ สามารถปรับค่าความถี่ที่ต้องการได้ง่ายภายใต้ย่านที่กว้างกว่าที่สามารถปรับได้ในแบบพาสซีฟ

-การแยกระหว่างอินพุตและเอาต์พุต เนื่องจากวงจรกรองความถี่ แบบแอคทีฟ มีการใช้ออปแอมป์ประกอบในวงจรจึงทำให้วงจรกรองความถี่แบบนี้มีอินพุตอิมพีแดนซ์สูงและเอาต์พุตอิมพีแดนซ์ต่ำ ทำให้ไม่มีผลการรบกวนกันระหว่างแหล่งจ่ายสัญญาณอินพุตและโหลด

แต่อย่างไรก็ตาม การกรองความถี่แบบแอคทีฟก็มีข้อเสียอยู่บางประการ เทื่อเทียบกับแบบพาสซีฟ คือ

-การตอบสนองความถี่ วงจรกรองความถี่แบบแอคทีฟมีความสามารถในการตอบสนองความถี่แคบพาสซีฟเนื่องจากขีดจำกัดของออปแอมป์

-แหล่งจ่ายไฟเลี้ยง เนื่องจากใช้ออปแอมป์ทำให้ต้องใช้แหล่งจ่ายไฟเลี้ยงในการทำงานในขณะที่แบบพาสซีฟไม่ต้องใช้แหล่งจ่ายไฟเลี้ยง

-วงจรกรองกรองความถี่แบบแอคทีฟสามารถแบ่งออกเป็นชนิด ตามหน้าที่การทำงานได้อีก 4 ชนิด ได้แก่

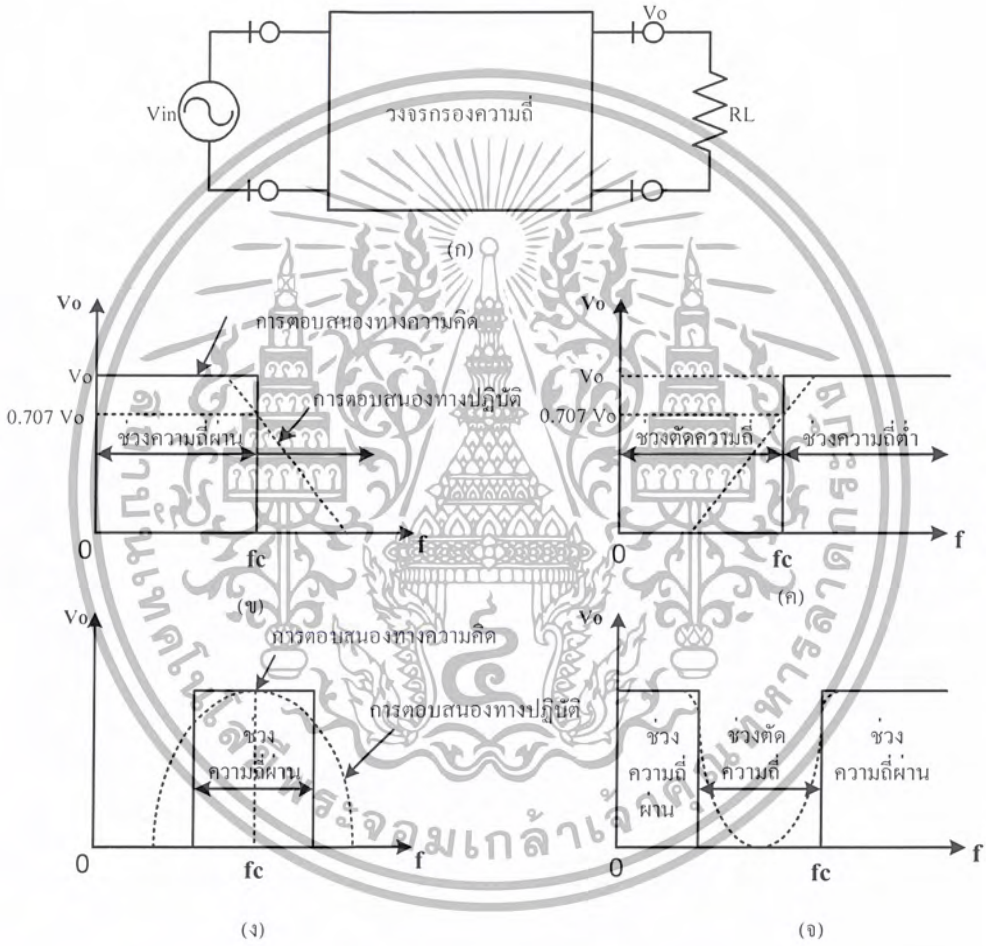
-วงจรกรองความถี่ต่ำย่าน (Low pass filter , LPF)

-วงจรกรองความถี่สูงย่าน (High pass filter , HPF)

-วงจรกรองแถบความถี่ย่าน (Band pass filter , BPF)

-วงจรตัดความถี่ต่ำย่าน (Band reject filter , BRF)

การทำงานของวงจรกรองความถี่แต่ละชนิด แสดงดังรูปที่ 2.1



รูปที่ 2.1 การทำงานของวงจรกรองความถี่แต่ละชนิด

(ก) วงจรทดสอบวงจรกรองความถี่

(ข) การทำงานของวงจร LPF

(ค) การทำงานของวงจร HPF

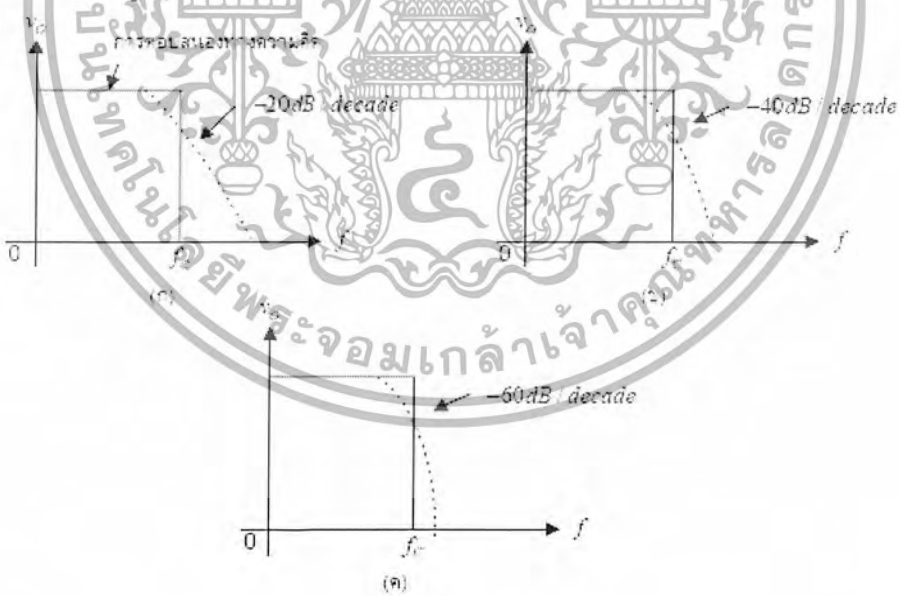
(ง) การทำงานของวงจร BPF

(จ) การทำงานของวงจร BRF

จากรูปที่ 2.1 (ข) ซึ่งเป็นผลการตอบสนองความถี่ของวงจร LPF จะพบว่าในทางความคิดหรืออุดมคติถ้าความถี่ของ อินพุตมีค่ามากกว่า ความถี่ที่ตองการ (Cut off frequency, fc) แล้ว วงจรจะตัดสัญญาณความถี่นั้น ไม่ให้ออกไปที่เอาต์พุตเลย แต่ในทางปฏิบัติวงจรไม่สามารถตอบสนองความถี่เช่นนี้ได้ วงจรจะค่อยๆ ลดการตอบสนองความถี่ของช่วงที่ไม่ต้องการลง ส่วนผลการตอบสนองความถี่ของวงจร HPF ดังรูปที่ 2.1 (ค) มีลักษณะตรงกันข้ามกับวงจร LPF ส่วนในรูปที่ 2.1 (ง) เป็นผลการตอบสนองความถี่ของวงจรของวงจร BPF คือ วงจรจะยอมให้ความถี่เฉพาะแถบหรือช่วงที่ต้องการเท่านั้นออกไปที่เอาต์พุต สำหรับผลของวงจร BRF ดังรูปที่ 2.1(จ) ก็จะมีลักษณะตรงข้ามกับวงจร BPF

2.1.1 ลำดับของวงจรกรองความถี่

จากในรูปที่ 2.1(ข)เราจะเห็นว่าในทางปฏิบัติแล้วเมื่อความถี่อินพุตเปลี่ยนแปลงถึงจุดตัดความถี่ (fc) ของวงจร วงจรจะค่อยๆ ลดการตอบสนองความถี่ลง โดยอัตราการเปลี่ยนแปลงนี้จะขึ้นอยู่กับลำดับ (Order)ของวงจรกรองความถี่ซึ่งโดยทั่วไป วงจรกรองความถี่มีตั้งแต่ลำดับที่1,2,3,4ไปเรื่อยๆ จนถึงลำดับที่ n สำหรับตัวอย่างลำดับของวงจรกรองความถี่ที่มีผลต่อการเปลี่ยนแปลงการตอบสนองความถี่ ของวงจรกรองความถี่ต่ำผ่าน (LPF) แสดงไว้ในรูปที่ 2.2



รูปที่ 2.2 การตอบสนองความถี่ของวงจรกรองความถี่ที่ลำดับที่ n

(ก) ลำดับที่ 1 (First order)

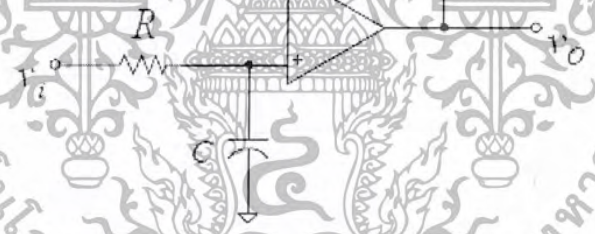
(ข) ลำดับที่ 2 (Second order)

(ค) ลำดับที่ 3 (Third order)

จากรูปที่ 2.2 จะเห็นว่าวงจรกรองความถี่ต่ำผ่านที่ตอบสนองความถี่ได้ใกล้เคียงอุดมคติมากที่สุด คือ ลำดับที่ 3 ถ้าลำดับที่สูงกว่านี้ การตอบสนองความถี่ก็จะยิ่งใกล้อุดมคติมากขึ้นไปอีก แต่ก็ทำให้วงจรกรองความถี่ที่มีลำดับสูงขึ้นตามไปด้วย ในทางใช้งานจริง มักจะนิยมใช้วงจรกรองความถี่ลำดับที่ 2 (2^{nd} order filter) มากกว่าเนื่องจากสามารถใช้ออปแอมป์เพียงตัวเดียวสร้างได้

วงจรกรองความถี่ลำดับที่ 2 จากรูปที่ 2.2 (ข) พบว่ามีอัตราการตกของความถี่เมื่อเทียบกับอัตราขยายเท่ากับ -40 db/decade โดยเครื่องหมายลบแสดงถึงอัตราให้เห็นว่าเป็นการตกและตัวเลขนี้หมายความถึง วงจรกรองความถี่ต่ำอยู่ลำดับที่ 2 ซึ่งจะมีอัตราการขาย 40 db/decade ต่อช่วง 10 เท่าของความถี่ ยกตัวอย่างเช่น ทุความถี่ 1 kHZ วงจรกรองความถี่มีอัตราขยาย 40 db (100 เท่า) เมื่อความถี่เพิ่มขึ้นไปถึง 10 kHZ วงจรกรองความถี่มีอัตราขยายลดลงเหลือ 0 db (1 เท่า) เนื่องจากความถี่ 1 kHZ ไปยังความถี่ 10 kHZ เราเรียกที่ 1 decade

2.1.2 วงจรกรองความถี่ต่ำผ่านอันดับที่1



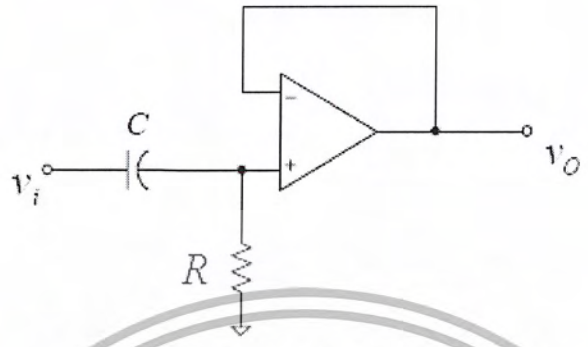
รูปที่ 2.3 วงจรกรองความถี่ต่ำผ่านอันดับที่ 1

วงจรกรองความถี่ต่ำผ่านอันดับที่1อยู่ในรูปที่ 2.3 ออปแอมป์ต่อเป็นวงจรตามแรงดันเพื่อทำให้เอาต์พุตอิมพีแดนซ์ของวงจรกรองความถี่ต่ำโดยค่าความถี่ตัด(fc) สามารถกำหนดได้จากค่าอุปกรณ์ภายนอก ดังสมการ 2.1

$$fc = \frac{0.1591}{RC} \qquad (2.1)$$

วงจรในรูปที่ 2.3 นี้จะมีอัตราขยายลูปปิด (Acl) เท่ากับ 1 ตามสมการของวงจรตามแรงดัน

2.1.3 วงจรกรองความถี่สูงผ่านอันดับที่ 1



รูปที่ 2.4 วงจรกรองความถี่สูงผ่านอันดับที่ 1

วงจรกรองความถี่ต่ำผ่านในรูปที่ 2.3 ถ้าหากการสลับตำแหน่ง R และ C ดังรูปที่ 2.4 ก็จะได้วงจรกรองความถี่สูงผ่านเท่ากับสมการ (2.1)

2.1.4 วงจรกรองความถี่ต่ำผ่านอันดับที่ 2

รูปที่ 2.5 วงจรกรองความถี่ต่ำผ่านอันดับที่ 2

วงจรของความถี่ต่ำผ่านลำดับที่ 2 สามารถสร้างได้ง่ายโดยใช้ออปแอมป์เพียงตัวเดียว ดังในรูปที่ 2.5 วงจรนี้อาจเรียกว่า วงจรกรองความถี่แบบ Sallen Key ซึ่งตั้งชื่อตามผู้ออกแบบวงจร สำหรับวงจรนี้ความถี่ตัด (Cut off frequency, fc) สามารถกำหนดได้จากค่าอุปกรณ์ที่ต่ออยู่ภายนอกตามสมการ (2.2)

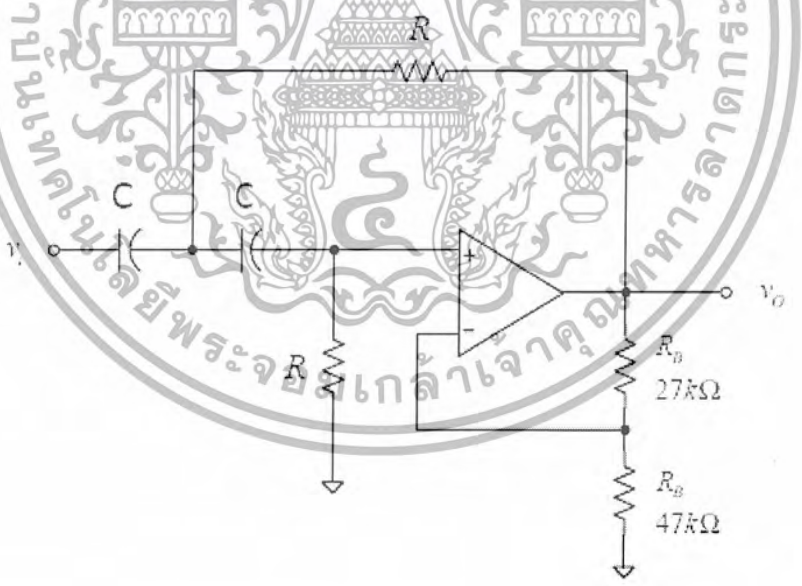
$$fc = \frac{1}{2\pi\sqrt{R1R2C1C2}} \qquad (2.2)$$

ถ้าหากเรากำหนดให้ R1=R2=R และ C1=C2=C สมการ (2.1) สามารถลดรูปได้เป็น

$$fc = \frac{1}{2\pi RC} \qquad (2.3)$$

ส่วนตัวต้านทาน Ra และ Rb มีไว้เพื่อเป็นตัวกำหนดอัตราการขยายวงจรลูปปิด(Acl) ได้ แต่โดยปกติแล้ว เราจะมีการกำหนดให้ Acl นี้มีค่าเท่ากับ 1.586 เพื่อให้วงจรสามารถทำงานได้อย่างมีประสิทธิภาพที่สุด นั่นแสดงว่า เราควรกำหนด ค่า RB มีค่าเป็น 0.586 เท่าของRa เนื่องจากออปแอมป์ต่อเป็นวงจรขยายแบบไม่กลับเฟสสำหรับค่า Ra และRbที่เหมาะสม คือRa =47kΩ และRb= 27kΩ

2.1.5 วงจรกรองความถี่สูงผ่านลำดับที่ 2



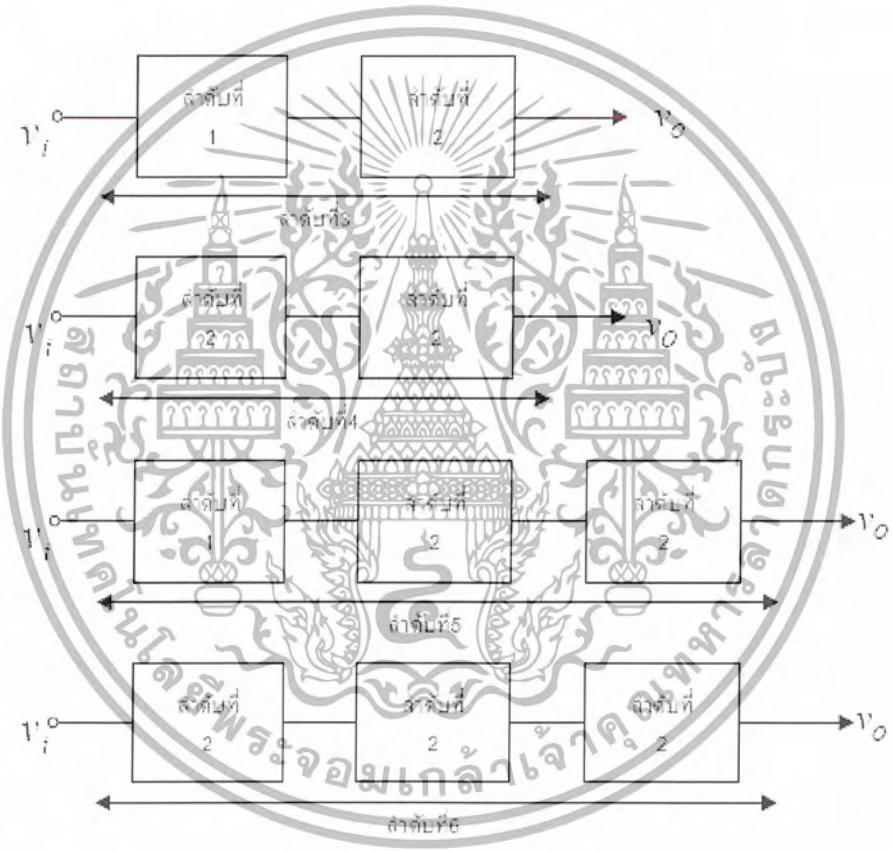
รูปที่ 2.6 วงจรกรองความถี่สูงผ่านลำดับที่ 2

วงจรกรองความถี่สูงผ่านลำดับที่ 2 มีลักษณะวงจรคล้ายกับวงจรกรองความถี่ต่ำผ่านในรูปที่ 2.3 เพียงแต่สลับตำแหน่งของ R และ C เท่านั้นดังแสดงในรูปที่ 2.5 ส่วนค่าความถี่ตัด (fc) สามารถหาได้จากสมการ (2.4)

$$fc = \frac{1}{2\pi RC} \qquad (2.4)$$

2.1.6 การสร้างวงจรของความถี่ต่ำและสูงผ่านในลำดับที่สูงขึ้น

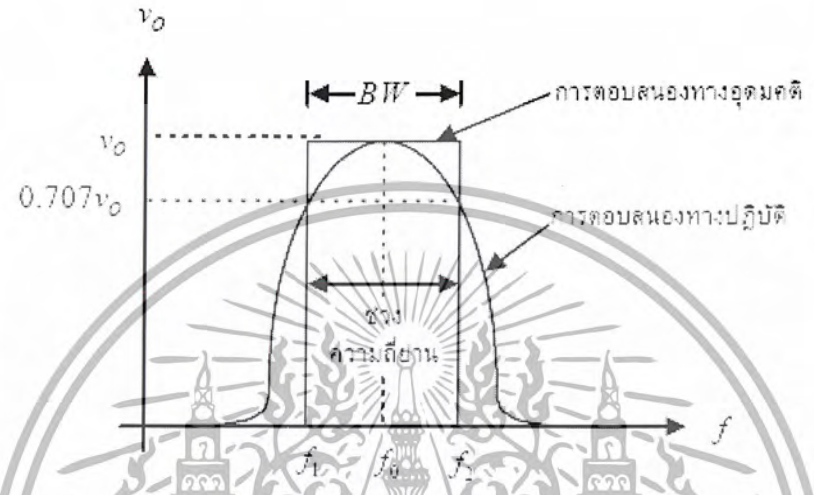
เราสามารถสร้างวงจรกรองความถี่ต่ำและสูงผ่านให้สูงกว่าลำดับที่สองได้ โดยการนำวงจรของความถี่ที่มีความถี่ตัดเท่ากันมาต่อเรียงกัน ดังรูปที่ 2.7 ซึ่งเราเรียกการต่อแบบนี้ว่า แคสเคด (Cascade) จะทำให้ได้ลำดับที่สูงขึ้นตามผลบวกของวงจรของความถี่ที่ต่อเรียงกัน



รูปที่ 2.7 การสร้างวงจรกรองความถี่ต่ำและสูงผ่านให้มีลำดับที่สูงขึ้น

2.1.7 วงจรกรองแถบความถี่

วงจรกรองแถบความถี่ (Band Pass Filter, BPF) หรือ BPF เป็นวงจรที่กรองเอาเฉพาะความถี่ช่วงที่ต้องการออกมาเท่านั้น ดังนั้นการตอบสนองความถี่ของวงจร BPF นี้จะเป็นดังรูปที่ 2.8



รูปที่ 2.8 การตอบสนองความถี่ของวงจร BPF

ความถี่ศูนย์กลาง (Center Frequency, fo) ของวงจร BPF สามารถกำหนดได้จากค่าอุปกรณ์ที่ต่อในวงจรเช่นเดียวกัน

ประสิทธิภาพ (Quality factor, Q) ของวงจร BPF นี้สามารถวัดได้จาก

$$Q = \frac{fo}{BW} \qquad (2.5)$$

เมื่อ BW คือแถบความถี่ที่ต้องการให้ผ่านมีหน่วยเป็น Hz หาได้จาก

$$BW = f2 - f1 \qquad (2.6)$$

และ fo ก็สามารถหาได้จาก

$$fo = \sqrt{f1 f2} \qquad (2.7)$$

ดังนั้น

$$f1 = \sqrt{\frac{BW^2}{4} + fo^2} - \left(\frac{BW}{2}\right) \qquad (2.8)$$

และ

$$f2 = f1 + BW \qquad (2.9)$$

2.2 หน่วยความดังของสียง(Volume Unit of Sound)

เดซิเบลหรือ dB เป็นหน่วยพื้นฐานในการวัดระดับสัญญาณเสียง กำเนิดขึ้นในปี ค.ศ. 1928 โดยบริษัทโทรศัพท์แห่งหนึ่งเพื่อใช้วัดการลดทอนที่เกิดขึ้นในสายโทรศัพท์ เดซิเบลมีค่าอ้างอิงแตกย่อยลงไปอีกมากมาย แต่ที่จะกล่าวนี้เป็นเดซิเบลเฉพาะในเรื่องของเครื่องเสียง

เดซิเบลใช้แพร่หลายในงานวัดย่านความถี่เสียง เพราะอัตราของตัวเลขในระบบลอการิทึมนั้นใกล้เคียงกับธรรมชาติการได้ยินของหูมนุษย์ เสียงที่หูมนุษย์สามารถได้ยินได้โดยไม่เป็นอันตรายจะอยู่ในช่วงตั้งแต่ 130dBลงมา และเสียงที่เปลี่ยนแปลงความดังอย่างน้อยที่สุด 1 dB หูถึงจะจับได้ และเสียงที่เปลี่ยนแปลงความดังไป 10 dB หูจะรู้สึกว่าเปลี่ยนไป 2 เท่า

หน่วยพื้นฐานของระบบนี้คือ เบล (bel) เบลคือ ลอการิทึมของอัตราส่วนของพลังงานสองค่า เดซิเบลเป็น 0.1 ของเบลหรือพูดกลับกันคือ เบลเป็น 10 เท่าของเดซิเบล

$$dB = 10 \log (P_1/P_2)$$

และเมื่อเราสามารถวัดโวลได้ง่ายกว่าเพาเวอร์ ดังนั้นเมื่อเพาเวอร์เท่ากับโวลเตจยกกำลังสอง (ความต้านทานคงที่)จะได้ว่า

$$dB = 10 \log (V_2^1/V_2^2)$$

แต่เนื่องจาก $\log A^2 = 2 \log A$

ดังนั้น $dB = 20 \log (V_1/V_2)$

เมื่อเดซิเบลเป็นอัตราส่วนระหว่างจำนวน 2 จำนวนการเปรียบเทียบจำนวนหนึ่งกับค่าอ้างอิงแล้วแยกชนิดออกไปย่อยๆ จะเป็นวิธีที่ง่ายขึ้น มีผู้จัดทำตารางเปรียบเทียบเดซิเบลย่อยๆ นี้ไว้อีกหลายแบบ เช่น

dB SPL คือ Sound Prcssure Level หรือระดับความดังของเสียงที่วัดเป็น dB อ้างอิงกบความดัน 0.00002 n/m^2 (เท่ากับ 0.0002 ไมโครบาร์) ซึ่งเป็นระดับเสียงค่อยที่สุดที่มนุษย์สามารถรับรู้ มีหน่วยสำหรบวัดความดังเสียงอยู่มากมาย แต่ละหน่วยก็สัมพันธ์ กันดังตัวอย่าง เช่นที่
1 ไมโครบาร์ เท่ากับ 74 dB SPL , 1 Pa เท่ากับ 94 dB SPL ค่า SPL ที่เป็นค่าเดซิเบลบวก ก็คือ เสียงที่ดังกว่าระดับดังกล่าวขึ้นมาเรื่อย ๆ และสามารถสรุปเป็นตารางในตารางที่1

ตารางที่ 1 ระดับความดังของเสียง

ระดับเดซิเบล	ต้นเสียง
0	จุดเริ่มได้ยินของหูมนุษย์
10	เสียงใบไม้กรอแกรบ
20	เสียงในห้องเก็บเสียงเงียบๆ
30	เสียงในสำนักงานเงียบๆ
40	เสียงนกร้องไกลๆ
50	เสียงในสำนักงานที่เปิดประตูทิ้งไว้
60	เสียงสนทนากันห่าง 3 ฟุต
70	เสียงสนทนากันห่าง 1 ฟุต
80	เสียงออร์เคสตร้าในระดับเฉลี่ย
90	เสียงพัดลมระบายอากาศขนาดใหญ่
100	เสียงเลื่อยยนต์
110	เสียงคอนเสิร์ตวงดนตรีแนวร็อค
120	เสียงเครื่องบินไอพ่นบินขึ้นในระยะ 1500 ฟุต
130-140	จุดที่เริ่มเจ็บปวดจากการได้ยิน
150-160	จุดที่ทำให้หูหนวกเฉียบพลัน

2.3 ไมโครโฟน (Microphone)

อุปกรณ์ทางอิเล็กทรอนิกส์ที่ทำหน้าที่เปลี่ยนคลื่นเสียง ให้เป็นคลื่นไฟฟ้าความถี่เสียง อุปกรณ์ที่เห็นได้ชัด ได้แก่ ไมโครโฟน

ไมโครโฟนเป็นอุปกรณ์ทางอิเล็กทรอนิกส์ ทำหน้าที่เปลี่ยนคลื่นเสียง (Sound wave) หรืออากาศจากแหล่งกำเนิดเสียง เช่น เสียงพูด เสียงเพลง เสียงเครื่องดนตรี เป็นต้น ให้เป็นสัญญาณไฟฟ้าความถี่เสียง ไหลไปตามสายไมโครโฟนสู่เครื่องขยายเสียง

ชนิดของไมโครโฟน ที่แบ่งตามวัสดุที่ใช้ในไมโครโฟน มี 6 ชนิด คือ

2.3.1) ไมโครโฟนชนิดคาร์บอน (Carbon Microphone) ไมโครโฟนชนิดนี้ให้เสียงที่มีคุณภาพไม่ค่อยดี ปัจจุบันใช้ในเครื่องโทรศัพท์เท่านั้น

2.3.2) ไมโครโฟนชนิดคริสตัล (Crystal Microphone) ไมโครโฟนที่มีราคาถูก น้ำหนักเบาแต่ไม่ทนต่อสภาพความร้อน หรือความชื้นสูง เพราะอาจทำให้คริสตัลเสื่อมได้ ไมโครโฟนแบบนี้ให้กำลังไฟฟ้าออกมาสูง และสามารถสภาพสัญญาณได้ดีจึงไม่ต้องอาศัยหม้อแปลง (Transformer) ในตัวของไมโครโฟนช่วยแต่อย่างใด สามารถส่งสัญญาณไปยังเครื่องขยายเสียงได้โดยตรง สามารถใช้สายไมโครโฟนต่อยาวออกไปได้ไม่เกิน 25 ฟุต เพราะถ้าพ่วงสายยาวกว่านี้จะทำให้มีสัญญาณอื่นมารบกวนได้และทำให้สัญญาณจากไมโครโฟนอ่อนลงมาก

2.3.3) ไมโครโฟนชนิดเซรามิค (Ceramic Microphone) มีลักษณะการออกแบบหรือหลักการทำงานคล้ายกับไมโครโฟนชนิดคริสตัล ต่างกันที่วัสดุเซรามิคมีคุณภาพดีกว่าคริสตัล เพราะทนทานต่อการเปลี่ยนแปลงของอุณหภูมิและความชื้นมากกว่า

2.3.4)ไมโครโฟนชนิดคอนเดนเซอร์ (Condenser Microphone) เป็นไมโครโฟนที่กำลังนิยมใช้อยู่ในปัจจุบัน สามารถรับเสียงได้ไวมาก มีราคาแพงและมักติดอยู่กับเครื่องบันทึกเสียงทั่ว ๆ ไป

2.3.5) ไมโครโฟนชนิดริบบอน (Ribbon or Velocity Microphone) เป็นไมโครโฟนที่บอบบาง เสียง่ายไม่มีไดอะแฟรม การทำงานอาศัยการสั่นสะเทือนของแผ่นริบบอน มีลักษณะบางเบา และขึงตึงอยู่ระหว่างแม่เหล็กถาวรกำลังสูงและจะทำงานทันทีเมื่อได้รับการสั่นสะเทือนเป็นไมโครโฟนที่มีคุณภาพสูงและควบคุมสัญญาณได้ดีที่สุด (Highest Fidelity) แต่ไม่ค่อยนิยมใช้กันมาก เพราะมีข้อเสียคือไม่เหมาะต่องานสถานที่ แม้แต่เสียงลมพัดก็จะรับเสียงเอาไว้หมดอาจแก้ได้โดยใช้วัสดุกันลม เป็นกระบอกฟองน้ำสวมครอบแต่ก็ไม่ได้ผลนัก นอกจากนี้ยังมีปัญหาอื่น ๆ อีก เช่น สัญญาณไฟฟ้าที่ได้ออกมาค่อนข้างต่ำ (Low Output) ต้องใช้เครื่องขยายเสียงที่มีกำลังแรง และ คุณภาพสูง ถ้าพูดใกล้มากเสียงลมหายใจจะกลบเสียงที่พูด ไมโครโฟนชนิดนี้ไม่นิยมใช้นอกสถานที่ มักพบในสถานีส่งวิทยุโทรทัศน์และบันทึกเสียง

2.3.6) ไมโครโฟนชนิดไดนามิค (Dynamic Microphone) เป็นแบบที่ได้รับนิยมมากเพราะให้คุณภาพเสียงดีเหมือนธรรมชาติ มีความทนทานเหมาะสมกับการกระจายเสียงหรือระบบเสียงหลายประเภท แต่ราคาค่อนข้างสูง

2.4 ข้อมูลของไมโครโฟนที่จะนำมาใช้งาน

เพื่อให้ไมโครโฟนสามารถทำงานร่วมกับอุปกรณ์ในระบบขยายเสียงได้อย่างมีประสิทธิภาพสูงสุด จึงควรต้องทราบข้อมูลของไมโครโฟนที่จะนำมาใช้ดังนี้ คือ

2.4.1) อิมพีแดนซ์ (Impedance) หมายถึงตัวเลขที่บอกค่าความต้านทานของไมโครโฟนที่เกิดขึ้นขณะที่มีสัญญาณไฟฟ้าความถี่เสียง หรือกระแสสลับไหลผ่านมีหน่วยเป็นโอห์ม แบ่งเป็น 2 พวก คือ

1.1 อิมพีแดนซ์สูง (High Impedance) จะมีค่าอยู่ในช่วง 5,10,50 หรืออาจถึง 100 กิโลโอห์ม (KQ) จะให้กำลังใจของสัญญาณออกมาต่ำ (Low Power Output) มีเสียงรบกวนได้ง่าย เช่นเสียงฮัม ยิ่งถ้าต่อสายยาว ๆ หรือเกินกว่า 25 ฟุต ก็ยิ่งทำให้สูญเสียกำลังของสัญญาณมากขึ้น คุณภาพของเสียงจะลดลงด้วย ใช้ต่อร่วมกับเครื่องขยายเสียงโดยต่อช่องที่ช่อง High Impedance

1.2 อิมพีแดนซ์ต่ำ(Low Impedance) มีค่าอิมพีแดนซ์อยู่ในช่วง 200 ถึง 600 โอห์มซึ่งมีคุณภาพดีให้กำลังของสัญญาณออกสูง (High Power Output) ไม่มีเสียงรบกวนสามารถใช้กับสายยาว ๆ ได้แต่จะมีความไวในการรับเสียงต่ำใช้ต่อร่วมกับเครื่องขยายเสียงที่ช่อง Low Impedance

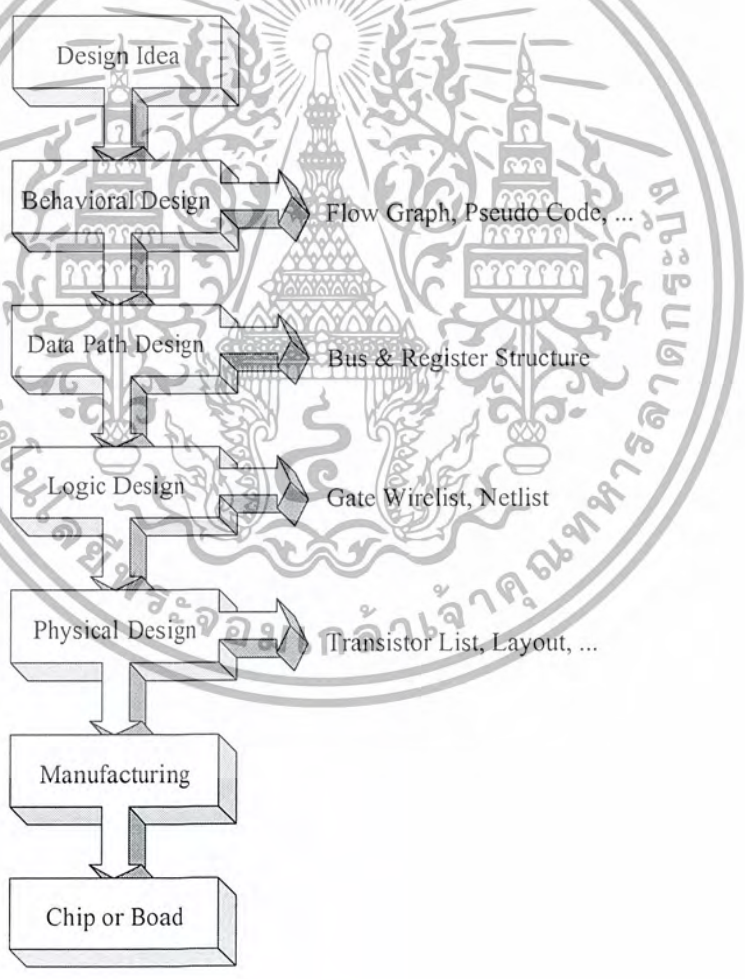
2.4.2) ผลในการตอบสนองความถี่ของเสียง (Frequency Response) คือความสามารถของไมโครโฟนในการรบความถี่ของคลื่นเสียงได้กว้างและมีความเรียบมากน้อย ซึ่งไมโครโฟนแต่ละชนิดก็จะออกแบบมาเพื่อใช้ในลักษณะงานต่าง ๆ กัน ฉะนั้น จึงมีความสามารถในการตอบสนองความถี่ต่าง ๆกัน มีหน่วยเป็น เฮิรตซ์ (Hertz: Hz) เช่น ไมโครโฟน สำหรับพูดในที่ชุมนุมชน ประกาศ สั่งงาน จะใช้ช่วงการตอบสนองความถี่ต่ำ ๆ และแคบ ๆ ก็พอ เช่น 300-5,000 เฮิรตซ์ แต่ถ้าต้องการคุณภาพของเสียงเรียบและแยกความถี่ได้กว้างขึ้น ควรอยู่ในช่วง 70-10,000 เฮิรตซ์ ถ้าต้องการคุณภาพของเสียงที่ดีเยี่ยมนอกจากเสียงพูดแล้ว ยังมีเสียงดนตรีด้วย ควรต้องใช้ไมโครโฟนที่ให้ผลตอบสนองความถี่ที่กว้างและเก็บความถี่ได้ละเอียดยิ่งขึ้น ควรอยู่ในช่วง 50-15,000 เฮิรตซ์ แต่ราคาก็จะค่อนข้างแพงตามคุณภาพ

2.4.3) ความไวในการรับเสียงของไมโครโฟน(Sensitivity) คือความสามารถในการรับความแรงของคลื่นเสียงที่มาจากแหล่งกำเนิดเสียงจากระยะทางใกล้ไกลต่าง ๆ กัน นั่นเองไมโครโฟนที่มีความไวสูงจะสามารถรับเสียงเบา ๆ และอยู่ไกลออกไปได้ไมโครโฟนความไวต่ำ ต้องป้อนคลื่นเสียงดัง ๆ และใกล้ ๆ มีหน่วยเป็น เดซิเบล (Decibel: dB) โดยวัดจากสัญญาณที่ได้ออกจากไมโครโฟนผ่านไปเข้าเครื่องขยายเสียง เช่น -90 dB -60dB -45dB เป็นต้น ค่าติดลบมาก จะมีความไวกว่า

บทที่ 3
องค์ประกอบพื้นฐานของภาษา VHDL

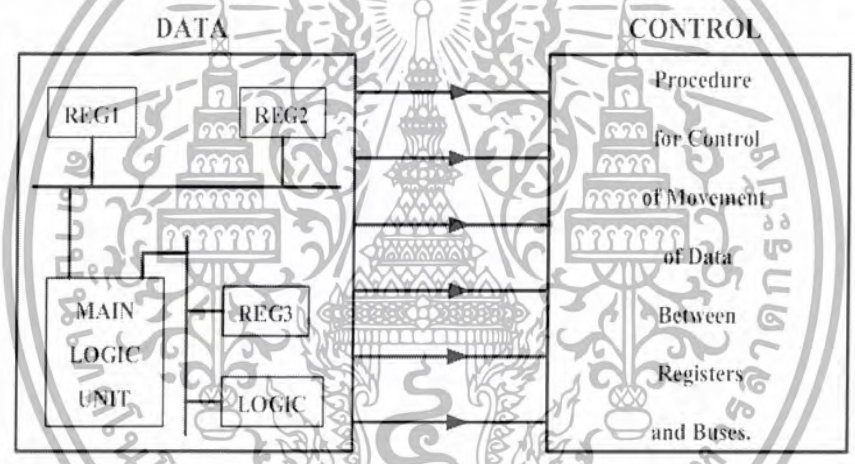
ความซับซ้อนและขนาดของระบบดิจิตอลในปัจจุบันได้เพิ่มมากขึ้นทุกขณะ ส่งผลให้มีการนำคอมพิวเตอร์เพื่อช่วยในการออกแบบหรือ CAD มาใช้ในขบวนการออกแบบฮาร์ดแวร์เพิ่มขึ้นเช่นกัน อีกทั้งอุปกรณ์และวิธีการออกแบบใหม่ๆ ก็ถูกพัฒนาขึ้นมาเพื่อช่วยอำนวยความสะดวกให้กับนักออกแบบมากขึ้นด้วย สำหรับภาษาบรรยายพฤติกรรมฮาร์ดแวร์ (HDL : Hardware Description Language) ก็เป็นเครื่องมืออย่างหนึ่งที่ได้รับการพัฒนามาอย่างต่อเนื่องเพื่อช่วยให้การปรับปรุงขบวนการออกแบบระบบดิจิตอลเป็นไปอย่างมีประสิทธิภาพ

3.1 การออกแบบระบบดิจิตอล



รูปที่ 3.1 ขั้นตอนการออกแบบระบบดิจิตอล

ในการออกแบบระบบดิจิตอล เริ่มตั้งแต่การกำหนดแนวความคิดเบื้องต้นจนกระทั่งได้ออกมาเป็นอุปกรณ์ฮาร์ดแวร์ที่ใช้งานได้จะต้องผ่านขั้นตอนต่างๆ มากมาย และในแต่ละขั้นตอนผู้ออกแบบจะต้องตรวจสอบผลลัพธ์ในแต่ละขั้นก่อนเข้าสู่กระบวนการออกแบบในขั้นต่อไป รูปที่ 3.1 แสดงขั้นตอนปกติที่ใช้ในการออกแบบระบบดิจิตอลทั่วไป ขั้นแรกผู้ออกแบบจะกำหนดแนวความคิดในการออกแบบแล้วทำการพัฒนาให้สามารถนำมาใช้ได้อย่างสมบรูณ์ ซึ่งภายในขั้นตอนนี้ผู้ออกแบบจำเป็นต้องสร้างรูปแบบระบบในเชิงพฤติกรรมขึ้นมาตรวจสอบซึ่งอาจจะเป็นผังงานแสดงแบบ (Flow Graph) หรือรหัสคำสั่งเทียม (Pseudo Code) ก็ได้ ขั้นตอนต่อไปเป็นการออกแบบระบบเส้นทางของข้อมูล (Bus) ผู้ออกแบบจะกำหนดส่วนประกอบของรีจิสเตอร์และวงจรลอจิกที่จำเป็นทั้งหมดเพื่อนำมาประกอบเป็นระบบที่สมบรูณ์ โดยแต่ละองค์ประกอบสามารถเชื่อมต่อกันด้วยบัสหนึ่งหรือสองทิศทาง (Unidirectional or Bidirectional Bus) ส่วนกระบวนการในการควบคุมการเคลื่อนย้ายข้อมูลระหว่างรีจิสเตอร์และวงจรลอจิกจะขึ้นอยู่กับพฤติกรรมของระบบที่กำหนดไว้ดังรูปที่ 3.2



รูปที่ 3.2 การออกแบบระบบเส้นทางของข้อมูล

ขั้นตอนถัดมาเป็นการออกแบบวงจรลอจิก ซึ่งจะเกี่ยวข้องกับการนำเกทดิจิตอลพื้นฐานและฟลิปฟลอป (flip-flop) มาประกอบเป็นอุปกรณ์ย่อยต่างๆ เช่น รีจิสเตอร์เก็บข้อมูล บัสวงจรลอจิก และส่วนควบคุมฮาร์ดแวร์ ซึ่งผลลัพธ์ที่ได้ในขั้นตอนนี้จะเป็นเครือข่ายของการโยงใยระหว่างเกทและฟลิปฟลอปนั่นเอง การออกแบบในขั้นตอนถัดไปเป็นการเปลี่ยนเครือข่ายการโยงใยที่ได้จากขั้นตอนที่แล้วให้เป็นลำดับของทรานซิสเตอร์ (Transistor List) และ Layout ซึ่งขั้นตอนนี้จะเกี่ยวข้องโดยตรงกับการจัดวางทรานซิสเตอร์หรือไลบรารีเซลล์เพื่อแทนเกทและฟลิปฟลอปต่างๆและในขั้นตอนสุดท้ายจะเป็นการส่งระบบที่ออกแบบไว้ไปทำการเจือสารที่โรงงานเพื่อผลิตออกมาเป็นวงจรรวมในที่สุด

3.2 ประวัติความเป็นมาของภาษาวีเอชดีแอล

วีเอชดีแอล (VHDL) ย่อมาจากคำว่า VHSIC Hardware Description Language (VHSIC : Very High Speed Integrated Circuit) เป็นภาษาโปรแกรมระดับสูง (High Level Language) ที่ใช้สำหรับการออกแบบฮาร์ดแวร์ในระบบดิจิตอล ตัวของภาษาสามารถบรรยายพฤติกรรมการทำงานในรูปของลำดับชั้น (Hierarchy) และสามารถเขียนได้หลายรูปแบบ ด้วยเหตุผลนี้จึงทำให้ภาษาวีเอชดีแอลเป็นเครื่องมือที่ใช้ออกแบบตั้งแต่ขั้นตอนบนสุด คือ แนวความคิดที่จะแก้ปัญหา ลงไปทีละขั้นจนถึงขั้นตอนของการสร้างวงจรจริง และตัวภาษาก็เปิดโอกาสให้วิศวกรได้พัฒนาและจำลองการทำงานของรูปแบบฟังก์ชันการทำงานของวงจรอย่างสังเขป โดยยังไม่ต้องคำนึงถึงรายละเอียดเกี่ยวกับโครงสร้างวงจรจริง นอกจากนั้นวีเอชดีแอลยังเป็นภาษาที่สนับสนุนลักษณะต่างๆ ของระบบดิจิตอลที่มีความซับซ้อนได้ทั้งหมด ดังนั้นวีเอชดีแอลจึงเป็นภาษาที่น่าสนใจในการศึกษาและนำไปใช้งานเป็นอย่างยิ่ง

วิวัฒนาการของวีเอชดีแอลเริ่มต้นประมาณปี ค.ศ. 1981 เมื่อกระทรวงกลาโหมสหรัฐอเมริกา หรือ DoD (Department of Defense) ได้พยายามปรับปรุงอุปกรณ์อิเล็กทรอนิกส์และคอมพิวเตอร์ที่ใช้ในกิจการทางทหารให้มีความทันสมัยมากขึ้น ประกอบกับเทคโนโลยีทางด้านไมโครอิเล็กทรอนิกส์มีการพัฒนาไปอย่างรวดเร็วดังจะเห็นได้จากการนำวงจรดิจิตอลหลายๆ วงจรมาทำการผลิตอยู่บนแผ่นซิลิกอนที่มีพื้นที่เพียง 1 – 2 ตารางเซนติเมตรเท่านั้น ซึ่งเป็นผลให้ประสิทธิภาพในการทำงานของวงจรสูงขึ้น ตลอดจนความน่าเชื่อถือในการทำงานและความคงทนต่อสภาพแวดล้อมสูง แต่เนื่องจากในขณะนั้นขั้นตอนของการออกแบบ การผลิต และการตรวจสอบวงจรต้นแบบ เป็นขบวนการที่ต้องใช้วิศวกร และเวลาในดำเนินการมาก ฉะนั้นทาง DoD จึงจัดตั้งโครงการขึ้นมาเพื่อศึกษาวิธีการที่ช่วยในการพัฒนาวงจรอิเล็กทรอนิกส์ โดยเฉพาะอย่างยิ่งวงจรระบบดิจิตอล ให้สามารถนำไปผลิตได้เร็วขึ้น ซึ่งโครงการดังกล่าวมีชื่อว่า "Very High Speed Integrated Circuits" หรือ VHSIC โดยในระยะแรกนั้นโครงการนี้ถือเป็นความลับทางด้านความมั่งคงของประเทศ และอยู่ภายใต้ความควบคุมดูแลของ United States International Traffic and Arms Regulations (ITAR)

สำหรับมาตรฐานของภาษาที่ใช้บรรยายพฤติกรรมวงจรหรือฮาร์ดแวร์ของระบบสำหรับโครงการ VHSIC ที่ DoD ได้ให้ไว้สามารถสรุปได้ดังนี้

- ต้องเป็นภาษาที่นำไปเขียนรูปแบบระบบดิจิตอล และมีคุณสมบัติที่สามารถเข้าใจได้ทั้งมนุษย์และเครื่องคอมพิวเตอร์โดยไม่ต้องมีการแปลหรือเปลี่ยนแปลงอีก
- สามารถนำไปใช้เป็นเอกสารประกอบโครงการได้
- ต้องเป็นภาษาที่เขียนขึ้นสำหรับใช้จำลองการทำงานของวงจร

ฉะนั้นภาษาดังกล่าวนี้จึงจัดเป็นภาษาโปรแกรมระดับสูง เช่นเดียวกับภาษาปาสคาล หรือภาษาซี ซึ่งในทางวิศวกรรมภาษาที่ใช้ในการออกแบบฮาร์ดแวร์นี้เรียกว่า "Hardware Description Language"หรือ HDL

ในตอนเริ่มแรกนั้น DoD ได้มอบหมายให้บริษัทไอบีเอ็ม เท็กซัสอินสตูเมนท์ และอินเตอร์เมทริกซ์ เป็นผู้ศึกษาและพัฒนาโครงการ ซึ่งการดำเนินงานเป็นไปอย่างต่อเนื่อง จนกระทั่งในปี ค.ศ.1985 ทาง ITAR ได้ยกเลิกข้อจำกัดในการถ่ายทอดเทคโนโลยีทางทหารออกจากโครงการนี้ ดังนั้นภาษาวีเอชดีแอลจึงเริ่มเป็นที่รู้จักกันโดยทั่วไป และประมาณปี ค.ศ. 1987 IEEE ได้ทำการกำหนดมาตรฐานของภาษานี้เป็น IEEE 1076-1987 และมีชื่อเรียกว่าวีเอชดีแอลซึ่งมาตรฐานนี้ได้รับการปรับปรุงจนเป็นมาตรฐาน IEEE 1076-1993 หรือ VHDL 1993

เนื่องจากในขณะนั้น DoD เป็นลูกค้ารายใหญ่ของอุตสาหกรรมอิเล็กทรอนิกส์และคอมพิวเตอร์ ดังนั้นจึงมีผู้รับโครงการต่างๆ จาก DoD ไปดำเนินการวิจัยและพัฒนาเป็นจำนวนมาก และเพื่อให้ทุกโครงการอยู่ในมาตรฐานเดียวกันหมด ดังนั้นทาง DoD จึงได้กำหนดว่า ทุกๆ โครงการต้องเขียนอยู่ในรูปของภาษาวีเอชดีแอลเท่านั้น ซึ่งทำให้ DoD สามารถนำโครงการเหล่านี้ไปจำลองกับเครื่องคอมพิวเตอร์ได้หลายๆระบบ

3.3 ข้อกำหนดของภาษาวีเอชดีแอล

3.3.1 ลักษณะทั่วไป

DoD ได้กำหนดให้วีเอชดีแอลเป็นภาษาสำหรับการออกแบบและบรรยายของฮาร์ดแวร์ ซึ่งหมายถึงความสามารถในการอธิบายและออกแบบในระดับสูง การจำลอง (Simulation) การสังเคราะห์ (Synthesis) และการทดสอบ (Testing) นอกจากนั้นวีเอชดีแอลยังถูกกำหนดไว้สำหรับการบรรยายฮาร์ดแวร์ตั้งแต่ระดับบนซึ่งก็คือระบบจนถึงระดับเกทอีกด้วย

เนื่องจากในการทำงานของระบบดิจิตอลนั้น ทุกๆ องค์ประกอบภายในระบบไม่ว่าเล็กหรือใหญ่จะทำงานไปพร้อมๆ กัน ซึ่งในเรื่องของความพร้อมเพรียงในการทำงานนี้ก็ถือเป็นข้อกำหนดที่สำคัญอย่างหนึ่งของวีเอชดีแอลด้วยเช่นกัน (สำหรับในภาษาที่ใช้ในการบรรยายฮาร์ดแวร์นั้นความพร้อมเพรียงจะหมายถึงทุกๆ คำสั่ง องค์ประกอบ เกทหรือวงจรลอจิกจะถูกนำมาปฏิบัติทั้งหมด ดังนั้นในที่สุดแล้วก็จะดูเหมือนว่าได้มีการปฏิบัติไปพร้อมๆ กัน)

3.3.2 สนับสนุนการออกแบบแบบลำดับขั้น

การออกแบบแบบลำดับขั้นเป็นลักษณะที่สำคัญอย่างหนึ่งสำหรับการออกแบบระบบที่มีหลายๆ ระดับ โดยในการออกแบบจะประกอบด้วยส่วนการบรรยายการเชื่อมต่อ และส่วนการบรรยายหน้าที่การทำงาน ซึ่งหน้าที่การทำงานของระบบสามารถกำหนดได้ด้วยตัวเอง หรืออาจถูกกำหนดโดยโครงสร้างที่ประกอบด้วยองค์ประกอบย่อยๆ ลงไปได้เช่นกัน แต่ที่ระดับล่างสุด องค์ประกอบต้องถูกบรรยายหน้าที่การทำงานด้วยตัวมันเอง และไม่สามารถกำหนดการทำงานโดยลักษณะแบบโครงสร้างได้

3.3.3 ไลบรารี

วีเอชดีแอลได้สนับสนุนการมีไลบรารีเพื่อระบบการจัดการที่ดี ผู้ออกแบบสามารถกำหนดลักษณะและการทำงานของอุปกรณ์พื้นฐานไว้ในระบบไลบรารี หรือจะใช้ไลบรารีที่ระบบได้จัดเตรียมไว้แล้วก็ได้ โมเดลและการบรรยายที่ถูกต้องควรจัดเก็บไว้ในไลบรารีหลังจากที่ได้ผ่านการคอมไพล์เรียบร้อยแล้วเพื่อให้ผู้ออกแบบคนอื่นๆ สามารถนำไปใช้ได้ด้วย

3.3.4 ลำดับคำสั่ง

แม้ว่าการปฏิบัติคำสั่งหรือกระบวนการโดยพร้อมเพรียงกันจะเป็นคุณสมบัติที่สำคัญของ วีเอชดีแอลก็ตาม ตัวภาษาเองก็ยังมีการจัดเตรียมลักษณะการควบคุมแบบลำดับคำสั่งไว้ให้ด้วย เมื่อผู้ออกแบบได้กำหนดหน้าที่และองค์ประกอบที่ทำงานพร้อมกันของระบบไว้เรียบร้อยแล้ว ผู้ออกแบบยังสามารถบรรยายหน้าที่การทำงานซึ่งเป็นรายละเอียดภายในของแต่ละองค์ประกอบได้ในลักษณะเดียวกับการเขียนโปรแกรมที่ประกอบด้วยโครงสร้างแบบ case, if - then - else และ loop ทั่วๆ ไปได้

การบรรยายแบบลำดับคำสั่งทำให้การออกแบบหน้าที่การทำงานของอุปกรณ์กระทำได้สะดวกและง่ายขึ้น อย่างไรก็ตามโครงสร้างทั้งหมดของวีเอชดีแอลก็ยังคงเป็นการทำงานแบบพร้อมเพรียงกันเช่นเดิม

3.3.5 การกำหนดคุณสมบัติ

นอกจากการกำหนดอินพุทและเอาท์พุทแล้ว เงื่อนไขอื่นๆ ก็มีผลต่อการปฏิบัติหน้าที่ของอุปกรณ์ฮาร์ดแวร์ด้วยเช่นกัน โดยสิ่งนี้รวมถึงสภาพแวดล้อมและลักษณะทางกายภาพของอุปกรณ์นั้นๆ ด้วย ซึ่งภาษาสำหรับการออกแบบที่ดีควรให้ผู้ออกแบบกำหนดคุณสมบัติของอุปกรณ์ที่ใช้ได้ด้วย เช่น สามารถกำหนดขนาด ลักษณะทางกายภาพ เวลา โหลด และเงื่อนไขทางสภาพแวดล้อมอื่นๆ ซึ่งความสามารถในการกำหนดคุณสมบัตินี้ก็เป็นส่วนหนึ่งที่มีอยู่ในภาษาวีเอชดีแอลด้วยเช่นกัน

3.3.6 ชนิดของข้อมูล

วีเอชดีแอลสามารถกำหนดชนิดของข้อมูลไม่เพียงแต่ชนิด BIT และ BOOLEAN เท่านั้น แต่ยังสามารถกำหนดชนิดของข้อมูลเป็นจำนวนเต็ม จำนวนจริง จุดทศนิยม และชนิดลำดับการนับ (Enumerate Type) หรือแม้แต่ชนิดของข้อมูลที่ผู้ออกแบบกำหนดขึ้นมาเองก็ได้

3.3.7 โปรแกรมย่อย

ความสามารถในการใช้ฟังก์ชันและโพรซีเจอร์ (Procedure) ก็เป็นข้อกำหนดอีกอย่างหนึ่งในวีเอชดีแอลซึ่งผู้ออกแบบสามารถนำโปรแกรมย่อยมาใช้ในการเปลี่ยนแปลงชนิดของข้อมูล การกำหนดหน่วยของลอจิก การกำหนดตัวกระทำต่างๆ หรือหน้าที่อื่นๆ ตามที่ต้องการได้เช่นเดียวกับการเขียนโปรแกรมทั่วไป

3.3.8 การควบคุมเวลา

วีเอชดีแอลอนุญาตให้ผู้ออกแบบสามารถกำหนดเวลาในการส่งผ่านข้อมูลหรือสัญญาณได้ตามต้องการ การตรวจสอบ การออกแบบเกทหรือการหน่วงเวลาก็สามารถกระทำได้โดยการกำหนดช่วงเวลาที่แน่นอนหรือกำหนดให้มีการรอคอยเหตุการณ์ (Event) นอกจากนี้ก็ยังสามารถกำหนดรูปแบบของสัญญาณนาฬิกาได้อีกด้วย

3.3.9 การกำหนดแบบโครงสร้าง

การกำหนดโครงสร้างขององค์ประกอบต่างๆ สามารถกระทำได้ในทุกระดับของการออกแบบ โดยการกำหนดโครงสร้างขององค์ประกอบร่วมที่เกิดจากองค์ประกอบย่อยซึ่งแตกต่างกันหรือเหมือนกันก็เป็นข้อกำหนดอย่างหนึ่งของวีเอชดีแอลเช่นกัน

3.4 องค์ประกอบพื้นฐานของวีเอชดีแอล

รูปแบบพื้นฐานที่ใช้ในการบรรยายถึงองค์ประกอบของวีเอสดีแอลจะประกอบไปด้วยส่วนกำหนดการเชื่อมต่อ (Interface) และส่วนกำหนดลักษณะเชิงสถาปัตยกรรม (Architecture) ดังแสดงในรูปที่ 3.3 โดยในการบรรยายการเชื่อมต่อจะขึ้นต้นด้วยคำว่า ENTITY แล้วตามด้วยชื่อขององค์ประกอบจากนั้นตามด้วยคำว่า IS และถัดมาจะเป็นการบรรยายถึงพอร์ตการติดต่อ อินพุท – เอาท์พุท ขององค์ประกอบ ส่วนลักษณะภายนอกอื่น ๆ เช่น เวลา อุณหภูมิก็สามารถรวมเข้าไปในส่วนนี้ได้เช่นกัน

```
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 3.3 การกำหนดการเชื่อมต่อและสถาปัตยกรรม

ในส่วนของการกำหนดลักษณะเชิงสถาปัตยกรรมจะขึ้นต้นด้วยคำว่า ARCHITECTURE ซึ่งเป็นส่วนที่ใช้บรรยายหน้าที่การทำงานขององค์ประกอบ โดยหน้าที่การทำงานนี้จะขึ้นอยู่กับสัญญาณอินพุท – เอาท์พุทและพารามิเตอร์อื่นๆ ที่ได้กำหนดไว้ในส่วนของการเชื่อมต่อดังรูปที่ 3.3 และสำหรับการบรรยายหน้าที่ขององค์ประกอบจะเริ่มต้นหลังจากคำว่า BEGIN เป็นต้นไป

3.4.1 การกำหนดการเชื่อมต่อ

การกำหนดการเชื่อมต่อเป็นระดับบนสุดของการออกแบบ โดยในระดับนี้ต้องกำหนดพอร์ตสำหรับการติดต่อกับองค์ประกอบภายนอกอื่นๆ ดังตัวอย่างในรูปที่ 3.4 ซึ่งเป็นบล็อคไดอะแกรม และการบรรยายการเชื่อมต่อขององค์ประกอบสำหรับตัวจ่ายสัญญาณนาฬิกา

ในบรรทัดแรกของการบรรยายการเชื่อมต่อเป็นการกำหนดชื่อขององค์ประกอบซึ่งกำหนดเป็น 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 3.4 บล็อคไดอะแกรมและการบรรยายการเชื่อมต่อของ clock_ component

3.4.2 การกำหนดรูปแบบการบรรยาย

หน้าที่การทำงานขององค์ประกอบจะถูกบรรยายภายในส่วนนี้ ซึ่งในการบรรยายสามารถกำหนดค่าของสัญญาณเอาท์พุทในเทอมของอินพุทหรือในรูปขององค์ประกอบอื่นๆ หรือทั้งสองอย่างรวมกันก็ได้ ดังตัวอย่างการบรรยายของ clock_component ในรูปที่ 3.5 ซึ่งเป็นการบรรยายในเชิงพฤติกรรมโดยมี en เป็นอินพุทและ ck เป็นเอาท์พุท

PROCESS เป็นคำที่ใช้ในการเริ่มต้นสำหรับการบรรยายในเชิงพฤติกรรม และภายในโปรเซสกำหนดให้ periodic เป็นตัวแปรที่มีค่าเริ่มต้นเป็น “0” ถ้าสัญญาณ en มีค่าเป็น “1”’ จะทำให้ตัวแปร periodic ถูกคอมพลีเมนท์ (complement) และส่งค่าให้กับ ck ซึ่งเป็นสัญญาณเอาท์พุท และสำหรับคำสั่ง WAIT จะเป็นการกำหนดให้สัญญาณมีคาบเวลาเท่ากับ 1 ไมโครวินาที

```
ARCHITECTURE behavioral OF clock_componant IS
  BEGIN
    PROCESS
      VARIABLE periodc : BIT := ' 0 ' ;
    BEGIN
      IF en = '1' THEN
       periodc = Not periodc ;
      END IF ;
     Ck <= periodc 1 US ;
      WAIT FOR 1 US;
    END PROCESS;
  END behavioral;
```

รูปที่ 3.5 การบรรยายเชิงพฤติกรรมของ clock_component

3.4.3 หน่วยการออกแบบแพ็กเก็ต

ข้อมูลต่างๆ ตลอดจนโปรแกรมย่อย ที่เป็นประโยชน์ต่อการเขียนรูปแบบการบรรยายระบบดิจิตอล สามารถเก็บไว้ในส่วนของแพ็กเก็ต ซึ่งหน่วยการออกแบบต่างๆ เช่น หน่วยการออกแบบ Entity หน่วยการออกแบบสถาปัตยกรรม หรือหน่วยการออกแบบแพ็กเกจอื่นๆ สามารถเรียกข้อมูลเหล่านี้ไปใช้ได้ นอกจากนั้นสิ่งที่นิยมทำกันมากคือการนำรูปแบบมาตรฐานต่างๆ เช่นอุปกรณ์มาตรฐาน (เช่น ไอซีตระกูล 74XX เป็นต้น) มาเก็บไว้ในรูปของแพ็กเกจ ที่ทุกคนสามารถเข้าถึงได้

ตามปกติแล้วแพ็กเกจจะแบ่งออกเป็น 2 ส่วนคือ การประกาศแพ็กเกจ (Package declaration) และส่วนของบอดี้แพ็กเกจ (Package body) เนื่องจาก แพ็กเกจถูกสร้างขึ้นเป็นส่วนแยกต่างหากออกจากรูปแบบที่กำลังเขียนอยู่ ฉะนั้นการที่นำแพ็กเกจไปใช้นั้นจะต้องมีการเชื่อมโยงหรืออ้างอิงเสียก่อน ซึ่งในภาษาวีเอชดีแอลสามารถกระทำได้ด้วยชุดคำสั่ง USE

1. PACKAGE DECLARATION

ส่วนที่มีความสำคัญที่สุดของแพ็กเกจ (ถ้ามองในแง่ของการนำไปใช้จากภายนอก) ได้แก่ส่วนการประกาศแพ็กเกจ เนื่องจากเป็นส่วนที่ใช้กำหนดชื่อของสิ่งที่ประกาศอยู่ภายในแพ็กเกจ สำหรับนำไปใช้ภายนอกตัวของแพ็กเกจเอง ถ้ามีการประกาศสิ่งใดๆ ในส่วนของส่วนบอดี้แพ็กเกจ แต่ไม่ถูกประกาศในส่วนการประกาศแพ็กเกจ จะทำให้ค่าและพฤติกรรมไม่สามารถนำไปใช้งานในส่วนนอกได้ ซึ่งเปรียบเทียบได้กับสิ่งที่ประกาศไว้ในส่วนของการประกาศ Entity คือ จุดเชื่อมต่อ หรือ พอร์ต ที่มี

หน้าที่ติดต่อกับโลกภายนอก ฉะนั้นโดยทั่วไปแล้วแพ็กเกจสามารถสร้างขึ้นได้โดยไม่จำเป็นต้องมีส่วนบอดี้ และยังสามารถนำไปใช้งานจากรูปแบบภายนอกได้เช่น ใช้สำหรับประกาศ ชนิด (Type) หรือสัญญาณ เช่นเดียวกับ ส่วนบอดี้แพ็กเกจที่ไม่จำเป็นต้องมี ส่วนของการประกาศแพ็กเกจ แต่แพ็กเกจนั้นจะไม่สามารถนำไปใช้จากรูปแบบอื่นได้

```
PACKAGE pack_name IS
    Package_declarative_part
END pack_name;
```

รูปที่ 3.6 โครงสร้างทั่วไปของส่วนการประกาศแพ็กเกจ

2. PACKAGE BODY

โครงสร้างซึ่งประกอบด้วยลำดับคำสั่งที่ใช้บรรยายฟังก์ชันการทำงานของโปรแกรมย่อยทั้งหลาย ซึ่งชื่อของโปรแกรมย่อยนั้นๆ ได้ถูกประกาศไปแล้วในส่วนของการประกาศแพ็กเกจ จะถูกเก็บไว้ในส่วนของบอดี้แพ็กเกจ ทั้งนี้รวมถึงการกำหนดค่าคงที่ต่างๆ อันได้แก่ค่าคงที่ที่ถูกประกาศชื่อไว้ก่อนในส่วนของการประกาศแพ็กเกจ และถูกกำหนดค่าในส่วนของบอดี้แพ็กเกจ ฉะนั้นในส่วนของบอดี้แพ็กเกจจึงไม่จำเป็นต้องมี ถ้าในส่วนของการประกาศแพ็กเกจไม่มีการประกาศชื่อที่เป็นโปรแกรมย่อย หรือค่าคงที่ การเขียนบอดี้แพ็กเกจนั้นจะเป็นไปตามกฎเกณฑ์ดังแสดงในรูปที่ 3.7

```
PACKAGE BODY package_name IS
    Declarative part
END pack_name
```

รูปที่ 3.7 โครงสร้างของบอดี้แพ็กเกจ

3.4.4 หน่วยการออกแบบ Configuration

ดังที่ทราบกันแล้วว่าระบบดิจิตอลรูปแบบหนึ่งไม่ว่าจะเป็นอะไรก็ตาม จะสามารถมีหน่วยการออกแบบ Entity ได้เพียงหนึ่งเดียวเท่านั้น ซึ่งในหน่วยการออกแบบ Entity หนึ่งหน่วยนี้อาจจะมีสถาปัตยกรรมที่เป็นหน่วยรองได้หลายหน่วย ดังนั้นจะต้องมีหน่วยการออกแบบ Configuration มาเพื่อกำหนดการใช้ Configuration ของการประกอบ Entity กับหน่วยการออกแบบสถาปัตยกรรมหน่วยใดๆ เข้าด้วยกัน

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

รูปที่ 3.8 โครงสร้างโดยทั่วไปของหน่วยการออกแบบโครงแบบ

3.4.5 โปรแกรมย่อย

การใช้ฟังก์ชันและโพรซีเจอร์ในภาษาวีเอชดีแอลเปรียบได้กับการใช้โปรแกรมย่อยในการเขียนโปรแกรมภาษาชั้นสูงทั่วๆ ไป ค่าที่ถูกส่งกลับหรือถูกเปลี่ยนแปลงโดยโปรแกรมย่อยอาจจะมีหรือไม่มีผลต่อฮาร์ดแวร์โดยตรงก็ได้ เช่นถ้าใช้ฟังก์ชันแทนการกระทำในสมการบูลีนก็จะมีผลต่อวงจรลอจิกจริงๆ ในขณะที่ถ้าใช้โปรแกรมย่อยในการเปลี่ยนชนิดของข้อมูลหรือในการคำนวณค่าการหน่วงเวลาแล้วก็จะไม่มีผลต่อโครงสร้างของฮาร์ดแวร์ รูปที่ 3.9 แสดงการใช้โพรซีเจอร์เพื่อเปลี่ยนข้อมูลชนิด 8 บิตเป็นค่าจำนวนเต็ม และรูปที่ 3.10 แสดงการใช้ฟังก์ชันโดยกำหนดให้ X เป็นตัวแปรชนิดบิตแทนการกระทำในสมการบูลีน

```
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 : IN 0 TO 7 LOOP
        IF ib (i) = '1' THEN
        Result := result + 2**i ;
        END IF;
    END LOOP ;
    oi := result ;
END byte_to_integer;
```

รูปที่3.9 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 3.10 การใช้ฟังก์ชัน

3.4.6 โอเปอร์เรเตอร์

การบรรยายเชิงพฤติกรรมในภาษาวีเอชดีแอลมีตัวดำเนินการหรือโอเปอร์เรเตอร์ทางลอจิกและคณิตศาสตร์เช่นเดียวกับภาษาซอฟท์แวร์ทั่วไปดังรูปที่ 3.11

PREDEFIND OPERATORS

LOGICAL OPERATORS : AND OR NAND NOR XOR

OPERAND TYPE : BIT BOOLEAN

RESULTTYPE : BIT BOOLEAN

RELATIONAL OPERATIOR : = / = < <= > >=

OPERAND TYPE : any type

RESULTTYPE : Boolean

ARITMETIC OPERATOR : + - * / ** **MOD REM AES**

OPERAND TYPE : INTTEGER REAL Physical

RESULTTYPE : INTEGER REAL Physical

CONCANTENATION OPERATOR : &

OPERAND TYPE : ARRAY of any type

RESULTTYPE : array of any type

RESULTTYPE : array of any type

รูปที่ 3.11 ตัวดำเนินการในวีเอชดีแอล

3.4.7 เวลาและความพร้อมเพรียง

ในวงจรอิเล็กทรอนิกส์อุปกรณ์ทุกๆ ตัวจะอยู่ในสภาพเตรียมพร้อมเสมอ (Always Active) และจะมีเรื่องของเวลาเข้ามาเกี่ยวข้องในทุกๆ เหตุการณ์ที่เกิดขึ้นเสมอ วีเอชดีแอลเป็นภาษาที่ได้รับการออกแบบมาเพื่อให้สามารถบรรยายรูปแบบและการพ้องกันของเวลาสำหรับการทำงานของอุปกรณ์ได้อย่างถูกต้อง การบรรยายการทำงานที่อยู่ภายในส่วนของการบรรยายสถาปัตยกรรม จะมีการทำงานที่พร้อมเพรียงกันเสมอ หรือแม้แต่โปรเซสซึ่งมีการทำงานภายในเป็นแบบลำดับคำสั่งก็ตาม ซึ่งหากมีหลายๆ โปรเซสอยู่ภายในโครงสร้างเดียวกัน ทุกๆ โปรเซสก็จะทำงานไปพร้อมๆ กันด้วย

3.4.8 สัญญาณและตัวแปร

สัญญาณมีลักษณะเป็นเสมือนตัวกลางฮาร์ดแวร์ที่ใช้ในการส่งผ่านข้อมูลและมีเรื่องของเวลาเข้ามาเกี่ยวข้องด้วย การกำหนดค่าให้กับสัญญาณจะใช้สัญลักษณ์ <= ในการส่งค่าและสามารถใช้คำสั่ง AFTER เพื่อกำหนดช่วงเวลาในการส่งผ่านค่าของสัญญาณเช่น w <= a AFTER 12 NS หมายถึงการกำหนดค่าสัญญาณ a ให้กับ w หลังจากเวลาผ่านไป 12 นาโนวินาที

ในทางตรงข้าม ตัวแปรมีลักษณะเป็นเสมือนตัวกลางที่ใช้ในการส่งผ่านข้อมูลและไม่มีเรื่องของเวลาเข้ามาเกี่ยวข้องด้วย ซึ่งตัวแปรจะถูกใช้ในส่วนที่มีการทำงานเป็นแบบลำดับคำสั่งเช่นใน ฟังก์ชัน โพรซีเจอร์ และโปรเซส สำหรับการกำหนดค่าให้กับตัวแปรจะใช้สัญลักษณ์ :=

3.5 การบรรยายเชิงพฤติกรรม

การบรรยายลักษณะการทำงานของอุปกรณ์ฮาร์ดแวร์ในเชิงพฤติกรรม เป็นการบรรยายลักษณะการเปลี่ยนแปลงของข้อมูลในรูปแบบของอัลกอริธึมสำหรับการคำนวณผลลัพธ์ที่เกิดขึ้นซึ่งสืบเนื่องมาจากการเปลี่ยนแปลงสภาวะของข้อมูลที่เข้ามาโดยไม่คำนึงถึงลักษณะโครงสร้างหรือความสัมพันธ์ของอุปกรณ์ที่อยู่ภายในว่าจะเป็นอย่างไร ในหัวข้อนี้จะแสดงถึงการบรรยายเชิงพฤติกรรมแทนการใช้โมดูลฮาร์ดแวร์รวมถึงข้อกำหนดต่างๆ ที่ควรรู้

3.6 โปรเซส

โปรเซสเป็นรูปแบบพื้นฐานอย่างหนึ่งที่ใช้ในการกำหนดให้กับสัญญาณ โปรเซสจะอยู่ในสถานะที่เตรียมพร้อมอยู่เสมอและจะปฏิบัติคำสั่งพร้อมๆ กันกับโปรเซสอื่นๆ ที่อยู่ในสถาปัตยกรรมบรรยายเดียวกัน โดยโปรเซสจะปฏิบัติงานตามคำสั่งทันทีที่มีเหตุการณ์เกิดขึ้นกับสัญญาณที่อยู่ทางด้านขวามือของสัญลักษณ์กำหนดค่าให้กับสัญญาณ (<=)

```
PROCESS
  declarative part
  ...
BEGIN
  statement part
  ...
END PROCESS;
```

รูปที่ 3.12 รูปแบบของการบรรยายแบบโปรเซส

การบรรยายโปรเซสจะเริ่มต้นด้วยคำสั่ง PROCESS และจบด้วยคำสั่ง END PROCESS ในรูปที่ 3.12 เป็นการแสดงส่วนประกอบของการบรรยายแบบโปรเซส ซึ่งประกอบด้วยส่วนของการประกาศตัวแปรที่ต้องใช้และส่วนของการปฏิบัติคำสั่งเพื่อให้ได้ผลลัพธ์ที่ต้องการ

3.7 การกำหนดตัวดำเนินการภายในโปรเซส

ตัวดำเนินการภายในโปรเซสมี 3 ชนิดคือ ตัวแปร (Variable) ไฟล์ (File) และตัวคงที่ (Constant) ซึ่งตัวดำเนินการทั้งสามชนิดนี้หากมีการประกาศไว้ในโปรเซสใดก็จะใช้ได้เฉพาะภายในโปรเซสนั้นเท่านั้น สำหรับการติดต่อกับภายนอกหรือระหว่างโปรเซสสามารถทำได้โดยใช้สัญญาณ (Signal) หรือตัวคงที่ที่ได้ประกาศไว้ในส่วนของ ARCHITECTURE ในรูปที่ 3.13 แสดงตัวอย่างการประกาศตัวกระทำภายในโปรเซส ซึ่งจะอยู่ระหว่างคำสั่ง PROCESS และ BEGIN และค่าเริ่มต้นที่ถูกกำหนดให้กับตัวดำเนินการภายในโปรเซสจะถูกนำมาใช้ในตอนเริ่มต้นของการปฏิบัติเพียงครั้งเดียวเท่านั้น ต่างกับค่าเริ่มต้นที่อยู่ภายในโปรแกรมย่อยจะถูกนำมาใช้ทุกครั้งที่มีการเรียกใช้โปรแกรมย่อยนั้น ๆ

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

รูปที่ 3.13 ตัวอย่างการประกาศตัวดำเนินการภายในโปรเซส

3.8 การกำหนดการกระทำภายในโปรเซส

การกระทำใดๆ ภายในโปรเซสจะเป็นการปฏิบัติแบบลำดับ (Sequential) เสมอ ซึ่งภายในโปรเซสสามารถใช้ประโยคเงื่อนไขหรือการทำซ้ำได้เช่น IF-THEN-ELSE , CASE-WHEN , FOR LOOP และ WHILE-LOOP ดังตัวอย่างในรูปที่ 3.14 และ 3.15

```
ARCHITECTURE demo OF paratial_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 3.14 เงื่อนไขการกระทำในโปรเซส

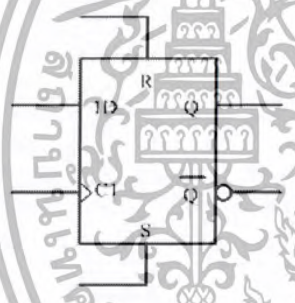
```
ARCHITECTURE demo OF paratial_process IS
  ...
BEGIN
  PROCESS
    BEGIN
      ...
      X <= a AFTER 10 NS;
      Y <= b AFTER 6 NS;
      ...
  END PROCESS;
END demo;
```

รูปที่ 3.15 การกระทำในโปรเซส

3.9 การกระตุ้นและยับยั้งการกระทำของโปรเซส

การกระทำภายในโปรเซสจะอยู่ในสภาพเตรียมพร้อม และมีการปฏิบัติงานอยู่ตลอดเวลาที่มีการเปลี่ยนแปลงของเหตุการณ์เกิดขึ้น อย่างไรก็ดีเราสามารถกระตุ้นหรือยับยั้งการกระทำภายในโปรเซสได้โดยการกำหนดรายการของสัญญาณที่ต้องการให้โปรเซสปฏิบัติงานเมื่อมีเหตุการณ์เกิดขึ้นกับสัญญาณที่กำหนดไว้เท่านั้น ส่วนเหตุการณ์ใดๆ ที่เกิดขึ้นกับสัญญาณที่ไม่ได้กำหนดไว้ในรายการก็จะไม่ส่งผลให้มีการกระทำภายในโปรเซส ซึ่งรายการของสัญญาณนี้เรียกว่า Sensitivity List และจะกำหนดไว้ภายในวงเล็บหลังคำสั่ง PROCESS

รูปที่ 3.16 (a) แสดงตัวอย่างโมเดล และรูปที่ 3.16 (b) เป็นตัวอย่างการบรรยายการเชื่อมต่อของ D-Flip Flop ส่วนรูปที่ 3.17 แสดงถึงการบรรยายเชิงพฤติกรรมของ D-Flip Flop โดยในรูปที่ 3.17 (a) เป็นการใช้ตัวกระทำภายนอกโปรเซส และรูปที่ 3.17 (b) เป็นการใช้ตัวกระทำภายในโปรเซส โดยมีรายการของสัญญาณ (rst, set, clk) เป็นตัวกระตุ้นการปฏิบัติงานภายในโปรเซส




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

(b)

รูปที่ 3.16 (a) ตัวอย่างโมเดล D-Flip Flop
(b) การบรรยายการเชื่อมต่อของ D-Flip Flop

```
ARCHITECTURE behavioral OF d_sr_flipflop IS
  SIGNAL state : BIT := '0' ;
EGIN
    Diff : PROCESS ( rst,set,clk )
    BEGIN
    IF set = '1' THEN
        State <= '0' 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 diff;
    q <= state ;
    qb <= NOT state ;
END behavioral;
```

(a)

```
ARCHITECTURE average_delay_behavioral OF d_sr_flipflop  IS
BEGIN
 Diff : PROCESS ( 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;
END behavioral;
```

(b)

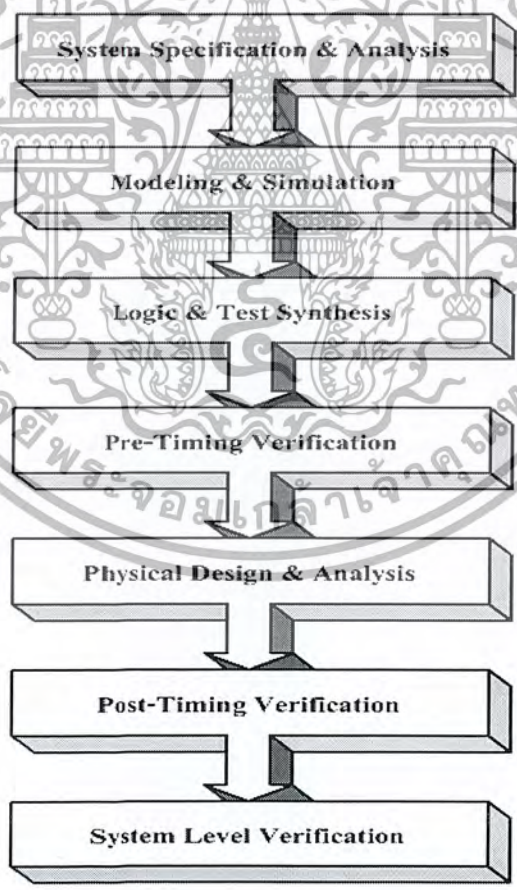
รูปที่ 3.17 การบรรยายเชิงพฤติกรรมของ D-FlipFlop

(a) การใช้ตัวกระทำภายนอกโปรเซส

(b) การใช้ตัวกระทำภายในโปรเซส

3.10 การออกแบบจากบนลงล่าง

ในการพัฒนาวงจรรวมดิจิตอลขนาดใหญ่ที่มีความซับซ้อน วิศวกรหรือผู้ออกแบบมักจะมองการออกแบบให้อยู่ในรูปของบล็อกไดอะแกรมก่อนที่ทำวิเคราะห์ให้ลึกถึงรายละเอียดต่อไป ซึ่งภาษาวีเอชดีแอลนั้นอนุญาตให้อธิบายและวิเคราะห์การทำงานของแต่ละบล็อก รวมถึงการปรับปรุงการทำงานจากผลที่วิเคราะห์เพื่อให้ได้การทำงานตามต้องการ นอกจากนี้ยังสามารถเพิ่มเติมในรายละเอียดในแต่ละขั้นตอนได้ ซึ่งหลักการนี้สอดคล้องกับหลักการออกแบบจากบนลงล่าง (Top - Down Design) นั่นเอง ถ้าทดลองเปรียบเทียบกับการออกแบบจากล่างขึ้นบน (Bottom - Up Design) จะเห็นได้ว่าการออกแบบจากล่างขึ้นบนจะใช้เวลาการออกแบบมากกว่า 90% เนื่องจากเป็นการวาดวงจรด้วยอุปกรณ์ต่างๆ (Schematic capture) ที่ประกอบกันเข้าเป็นวงจรที่ต้องการออกแบบ ก่อนแล้วจึงทำการจำลองการทำงาน และตรวจสอบความถูกต้อง ดังนั้นการใช้ภาษาวีเอชดีแอลกับหลักการออกแบบจากบนลงล่างจึงเป็นทางออกให้กับวิศวกรให้สามารถออกแบบและพัฒนาวงจรที่มีความซับซ้อนได้มากขึ้น ทั้งยังช่วยลดเวลาและค่าใช้จ่ายในการออกแบบด้วย



รูปที่ 3.18 ขั้นตอนการออกแบบจากบนลงล่าง

จากรูปที่ 3.18 แสดงถึงขั้นตอนของการออกแบบจากบนลงล่าง ทั้งนี้ในทางปฏิบัติอาจมีข้อแตกต่างไปจากนี้บ้างเล็กน้อย เนื่องจากขั้นตอนของการผลิต (Implementation) สามารถกระทำได้หลายเทคโนโลยี สำหรับรายละเอียดของขั้นตอนการออกแบบจากบนลงล่างในแต่ละขั้นตอนมีดังนี้

1.สร้างข้อกำหนดของความต้องการ และวิเคราะห์ระบบ เพื่อหาแนวความคิดและหลักการ (Idea and Concept) ในการแก้ปัญหา

2.เขียนรูปแบบของระบบที่ต้องการออกแบบโดยใช้ภาษาวีเอชดีแอลหรือภาษา HDL อื่น ๆ สำหรับบรรยายพฤติกรรมการทำงาน พร้อมทั้งจำลองการทำงาน เพื่อเปรียบเทียบและตรวจสอบความถูกต้องกับข้อกำหนด

3.หลังจากที่ได้หลักการขั้นต้นพร้อมแนวความคิดที่ผ่านการตรวจสอบแล้ว หลักการนี้จะถูกเพิ่มเติมในรายละเอียดลงมาเป็นลำดับขั้นที่สอง จนกระทั่งอยู่ในระดับที่จะนำไปผลิตวงจรจริง หรือสังเคราะห์ในขั้นตอนนี้เองเทคโนโลยีที่จะมารองรับวงจรออกแบบจะถูกกำหนดขึ้น และระบบช่วยการออกแบบจะสังเคราะห์วงจรที่ได้จากรูปแบบที่เขียนขึ้นให้อยู่ในรูปของวงจรที่ประกอบด้วยอุปกรณ์อิเล็กทรอนิกส์ หรือวงจรในระดับเกท และการเชื่อมต่อระหว่างกันของอุปกรณ์เหล่านั้นหรือไม่ก็อยู่ในรูปของ Netlist ที่สามารถนำไปผลิตในอุปกรณ์อื่นได้

4.หลังจากการสังเคราะห์วงจรให้อยู่ในระดับเกทหรือ Netlist แล้ว ข้อมูลนี้จะถูกใช้สำหรับจำลองการทำงานในเรื่องความถูกต้องของฟังก์ชัน พร้อมกับนำข้อมูลที่เกี่ยวข้องกับเวลาเข้ามาประกอบการพิจารณาด้วย ซึ่งตามปกติแล้วอุปกรณ์ทางอิเล็กทรอนิกส์ทุกชิ้นจะมีเวลาหน่วงของการแพร่กระจาย (Propagation Delay Time) เสมอ ถึงแม้ว่าจะเป็นเวลาที่น้อยมากในระดับนาโนวินาทีก็ตาม แต่ถ้าภายในวงจรหนึ่งประกอบด้วยเกทของฟังก์ชันต่างๆ จำนวน 10,000 เกท ขึ้นไป เวลาดังกล่าวนี้จะสะสมกันมากขึ้น จนอาจทำให้การทำงานของวงจรรวมทั้งหมดผิดพลาดไป หรือไม่สามารถทำงานในย่านความถี่สัญญาณนาฬิกาที่สูงได้

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

6.ทำการตรวจสอบการทำงานและตัวแปรทางด้านเวลาทั้งหมด เพื่อความถูกต้องของวงจรเป็นครั้งสุดท้ายก่อนนำไปรวมเข้ากับอุปกรณ์อื่นๆ ให้เป็นระบบดิจิตอล เนื่องจากในขั้นตอนนี้วงจรที่ออกแบบ จะประกอบด้วยจุดต่อทางอินพุทและเอาท์พุท ซึ่งเป็นจุดต่อสำหรับการรับและส่งสัญญาณกับภายนอก

7.นำวงจรที่ออกแบบไว้ประกอบเข้ากับอุปกรณ์อื่นๆ ให้เป็นระบบที่สมบูรณ์ แล้วทำการทดสอบการทำงานทั้งระบบร่วมกับอุปกรณ์อื่นๆ อีกครั้งเพื่อควบคุมคุณภาพของผลิตภัณฑ์

บทที่ 4
เอฟพีจีเอเทคโนโลยี (FPGA Technology)

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 4.1

รูปที่ 4.1 แสดงผังแสดงการแบ่งกลุ่มของวงจรรวม ASIC

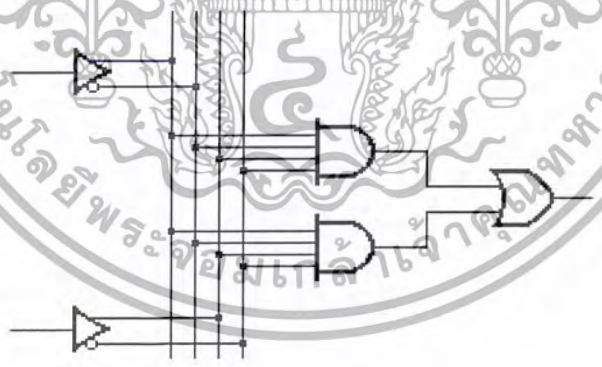
4.1 Field Programmable

อุปกรณ์วงจรรวมเฉพาะงาน ASIC แบบ field programmable มีอยู่มากมายหลายชนิด แต่มีลักษณะการสร้างหรือกำหนดการทำงานของวงจรที่เหมือนกัน กล่าวคือ ผู้ใช้งานสามารถออกแบบและสร้างวงจรที่ต้องการใช้ลงในตัวอุปกรณ์ได้เองโดยไม่ต้องไปโรงงานเพื่อผลิต โดยเฉพาะอย่างยิ่งในปัจจุบันนี้มีเครื่องมือที่ใช้ช่วยในการออกแบบ และสร้างวงจรร่วมกับไมโครคอมพิวเตอร์ที่มี

ความสามารถสูงในการพัฒนาตั้งแต่ขั้นการออกแบบ การจำลองการทำงาน จนถึงจัดสร้างวงจรลงในอุปกรณ์ รวมทั้งอุปกรณ์ Field Programmable เหล่านี้สามารถหาซื้อได้ง่ายทำให้การสร้างวงจรอิเล็กทรอนิกส์จนถึงระบบไมโครโพรเซสเซอร์หันมาใช้อุปกรณ์จำพวกนี้ เป็นอุปกรณ์ประกอบในวงจรแทนอุปกรณ์ย่อยๆ แยกชิ้น (Discrete component)

4.1.1 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

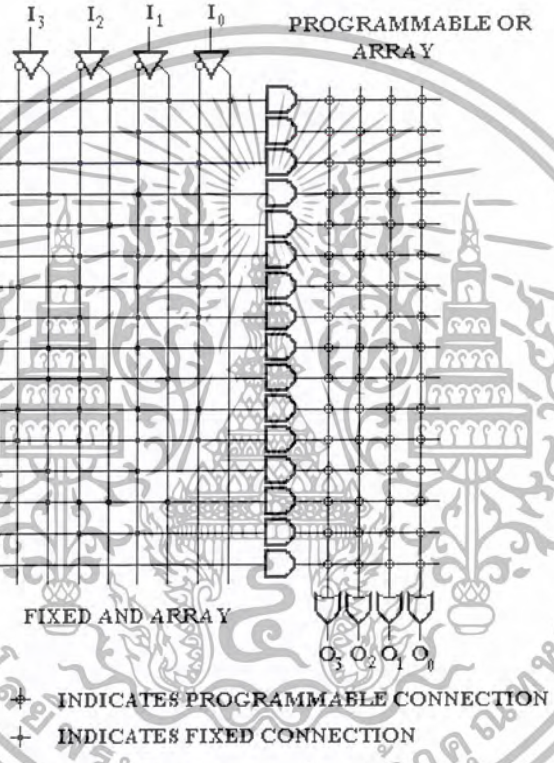


รูปที่ 4.2 แสดงวงจรพื้นฐานของอุปกรณ์พีแอลดีซึ่งอยู่ในรูปผลคูณร่วมบวก

4.1.2 พรอม (PROM: Programmable Read Only Memory)

พรอมคือหน่วยความจำรอม (ROM) ที่โปรแกรมได้ ซึ่งนับว่าเป็นอุปกรณ์พีแอลดี ชนิดหนึ่งซึ่งวงจรภายในของพรอมเสมือนกับประกอบไปด้วยแถวลำดับของแอนด์เกตและออเกต (Amd-Or Array) ผลเอาต์พุตที่ขาต้าเอาต์พุตสามารถแสดงในสมการของฟังก์ชั่นผลคูณร่วมบวก (Sum of product) ของสัญญาณอินพุตที่ขาแอดเดรส รูปที่ 4.3 แสดงถึงลักษณะการต่อเป็นแถวลำดับของแอนด์

เกตและออเกตของพรอมขนาด 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



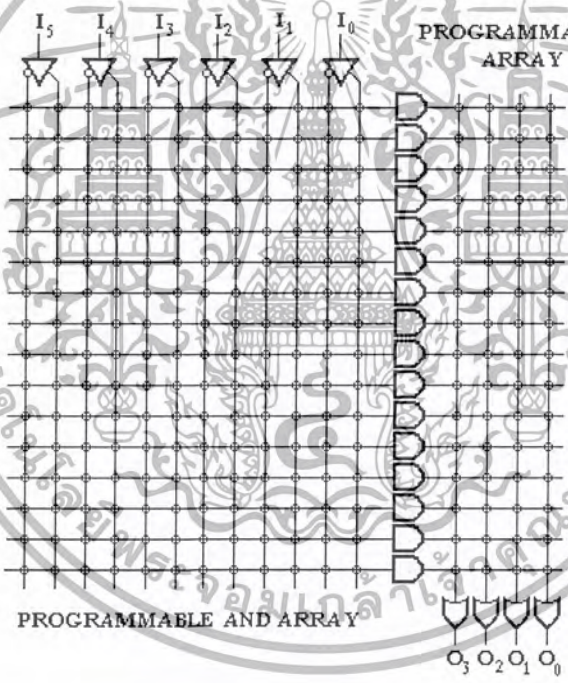
รูปที่ 4.3 แสดงลักษณะของพรอมเมื่อเปรียบเทียบเป็นวงจรในรูปผลคูณร่วมบวก

4.1.3พีเอแอล (PAL: Programmable Array Logic)

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

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



รูปที่ 4.4 แสดงวงจรพื้นฐานภายในของพีแอลเอ

4.1.5 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) สามารถโปรแกรมและ

ลบได้โดยใช้กระแสไฟฟ้า (Static RAM based) ภายในจัดเรียงเป็นเมทริกซ์ของลอจิกเซลล์ ล้อมรอบภายนอกด้วยอินพุต เอาต์พุตเซลล์ อุปกรณ์แอลซีเอตัวแรกของบริษัทไซริ้ง คือ แอลซีเอเบอร์ 2064 ประกอบด้วยเซลล์เรียงเป็นเมทริกซ์มีจำนวน 64 เซลล์ แต่ละเซลล์เรียกว่า ซีแอลบี (CLB : Configurable Logic Block) แต่ในปัจจุบัน ได้คพัฒนามาอยู่ในรูปของ เอฟพีจีเอ (FPGA : Field Programmable Gate Array) ซึ่งมีประสิทธิภาพ ความจุของเกตสูงมากขึ้น โดยสร้างออกมาเป็นอนุกรม (Series) ต่างๆ เช่น ตระกูล XC 3000 และ ตระกูล XC 4000

4.1.6 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 และลบได้โดยใช้แสงอุลตร้าไวโอเลทฉายผ่านช่องหน้าต่างกระจกเข้าไปกกระทบในตัวชิปของอุปกรณ์

4.2 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 และฟลูคัสตัม (full custom)

4.2.1 เกตอาเรย์ (Gate Array)

วงจรรวมนี้ประกอบด้วยแถวลำดับของวงจรเกต ซึ่งอาจจะเป็นวงจรเกตประเภทเดียวกันหรือต่างชนิดกันก็ได้ กับขั้วต่อสายไฟ (Pad) สำหรับต่อกับวงจรภายนอก ผู้ใช้มีหน้าที่ออกแบบการเชื่อมโยงทางไฟฟ้าระหว่างวงจรเกตแต่ละตัวและขั้วต่อสายไฟเพื่อให้ทำหน้าที่ตามต้องการ แล้วจึงส่งผลการออกแบบนี้ไปยังโรงงานผู้ผลิตวงจรรวมเกตอาเรย์ นั้นไปทำการเจือสารต่อไป โดยทั่วไปไป

แล้ว ปรากฏในการใช้วงจรรวมเกตอาเรย์สร้างวงจรที่ต้องการใช้งานทั่วไปจะมีเพียง 25-30% ของพื้นที่ซิลิคอนที่ใช้สำหรับวงจรเกต ส่วนพื้นที่ที่เหลือจะใช้กับการเชื่อมโยงวงจรเกตเข้าด้วยกันเองและเข้ากับขั้วต่อสายไฟ พื้นที่ซิลิคอนจะใช้ประโยชน์สูงถึง 75% สำหรับวงจรเชิงเลขที่มีลักษณะสม่ำเสมอ เช่น หน่วยความจำเป็นต้น

4.2.2 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 ซึ่งอาจจะเป็นแฟ้มข้อมูลคอมพิวเตอร์เหมาะสมสำหรับวงจรเชิงเลขที่มีความซับซ้อน การผลิตวงจรรวม เซลล์มาตรฐานจะมีต้นสูงกว่าวงจรรวมเกตอาร์เรย์และใช้เวลาในการออกแบบและเจือสารยาวกว่าสองถึงสามเท่าตัว วงจรรวมเซลล์มาตรฐานจึงเหมาะสมกับการใช้งานเชิงพาณิชย์ที่มีปริมาณการผลิตนับหมื่นตัวขึ้นไปบทบาทวงจรรวมเซลล์มาตรฐานจะมีเพิ่มมากขึ้นในอนาคต เมื่อตัวทุนการเจือสารลดลง

4.2.3ฟลูคัสตัม (Full Custom)

วงจรฟลูคัสตัม นี้ผู้ใช้เป็นผู้ออกแบบเองทั้งหมด ตั้งแต่ระดับวงจรเชิงเลขจนกระทั่งจึงระดับกายภาพ แล้วจึงส่งข้อมูลการออกแบบไปให้ผู้ผลิตเจือสารในรูปแฟ้มข้อมูลมาตรฐาน เช่น GDS II, CIF การใช้ฟลูคัสตัม ASIC นี้เท่าที่แพร่หลายอยู่ในปัจจุบัน จะเป็นไปเพื่อการศึกษาและการวิจัยและเพื่อการผลิตจำนวนน้อย ส่วนใหญ่จะเป็นการเจือสารในลักษณะที่ใช้ค่าใช้จ่ายร่วมกันกล่าวคือ รวบรวมการออกแบบกลายวงจรลงบนแผ่นซิลิคอนเดียวกันเพื่อประหยัดค่าเจือสาร

รูปแบบการเจือสารวงจรรวมฟลูคัสตัมที่ประกอบด้วยการออกแบบหลายวงจรรวมอยู่บนแผ่นซิลิคอนเดียวกัน มีกระทำอยู่ 3 รูปแบบดังนี้

1. Multi-project wafer (MPW) การเจือสารวงจรรวมฟูคัสตัมแบบนี้ 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) เป็นการเจือสารในลักษณะผสมผสานระหว่างฟูคัสตัมกัลเซลล์มาตรฐานกล่าวคือ ได้จะแบ่งออกเป็นส่วนๆ อย่างสม่ำเสมอแต่ละส่วนบรรจุการออกแบบแต่ละวงจร

โดยไดชุดหนึ่งจะบรรจุวงจรที่ออกแบบได้ประมาณ 8-วงจรและไดทุกชุดในแผ่นเวเฟอร์จะมีลักษณะเหมือนกัน-ในปัจจุบันมีเพียงอาวาที่ใช้บริการการเจือสารลักษณะนี้ โดยจำกัดเฉพาะประเภทเซลล์มาตรฐานเท่านั้น

4.3 เอฟพีจีเอ (FPGA: Field Programmable Gate Array)

เป็นอุปกรณ์ที่ถูกพัฒนาต่อจากอุปกรณ์แอลซีเอของบริษัทไซริงซ์ (XILINX Inc.) โดยมีประสิทธิภาพการทำงานและมีปริมาณความหนาแน่นของเกตสูง สามารถจะกำหนดฟังก์ชั่นการทำงานได้ความต้องการของผู้ใช้โดยผ่านการโปรแกรมเอฟพีจีเอได้รวบรวมข้อดีทั้งหมดของการทำคัสตัมวีแอลเอสไอ (Custom VLSI) มารวมไว้ทั้งหมดได้แก่ การออกแบบการผลิต, ลดเวลาที่จะส่งตัวผลิตภัณฑ์ออกตลาด ซึ่งเป็นประโยชน์ต่อการผลิตวงจรเป็นอย่างมาก นักออกแบบเพียงกำหนดฟังก์ชั่นการทำงานของวงจร ดังนั้นการออกแบบวงจรโดยใช้เอฟพีจีเอ สามารถออกแบบและทดสอบภายในเวลาเพียง 2-3 วัน เท่านั้น ตรงกันข้ามกับการออกแบบโดยใช้เกตอาร์เรย์ ซึ่งใช้เวลาหลายอาทิตย์การเปลี่ยนแปลงแก้ไขแบบก็เช่นเดียวกัน จากประโยชน์ของเอฟพีจีเอ ดังกล่าวมา ทำให้เกิดการประหยัดค่าใช้จ่ายเป็นอย่างมาก เพราะได้ความเสี่ยงในการที่จะต้องแก้ไขตัววงจร การเลื่อนเวลาการออกผลิตภัณฑ์ ลดค่าเอ็นอาร์อี (NRE : Nonrecurring Engineering Cost) ลงไปด้วย

4.3.1 เทคโนโลยีของ FPGA

เนื่องจากเป็นลักษณะของชิพที่สามารถโปรแกรมได้นั่นก็คือ สามารถกำหนดจุดเชื่อมต่อต่างๆ ภายในได้ เพื่อประกอบเป็นลักษณะของวงจรตามที่เราต้องการได้ ซึ่งเราสามารถแบ่งลักษณะของจุดเชื่อมต่อต่างๆ ได้ดังนี้

1. Physical Changing

1.1 Fused สามารถโปรแกรมได้เพียงครั้งเดียว หลังจากโปรแกรมจุดเชื่อมต่อขาดจากกัน

1.2 Anty Fuse สามารถโปรแกรมได้เพียงครั้งเดียวหลังจากโปรแกรม จุดเชื่อมต่อจะเชื่อมถึงกัน

2. Memory Base

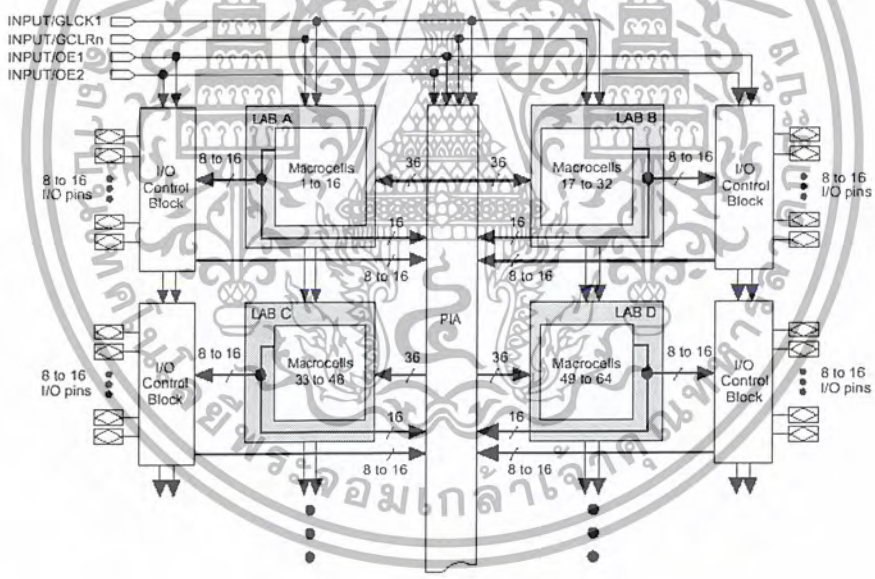
2.1 EEPROM – Base FPGA

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

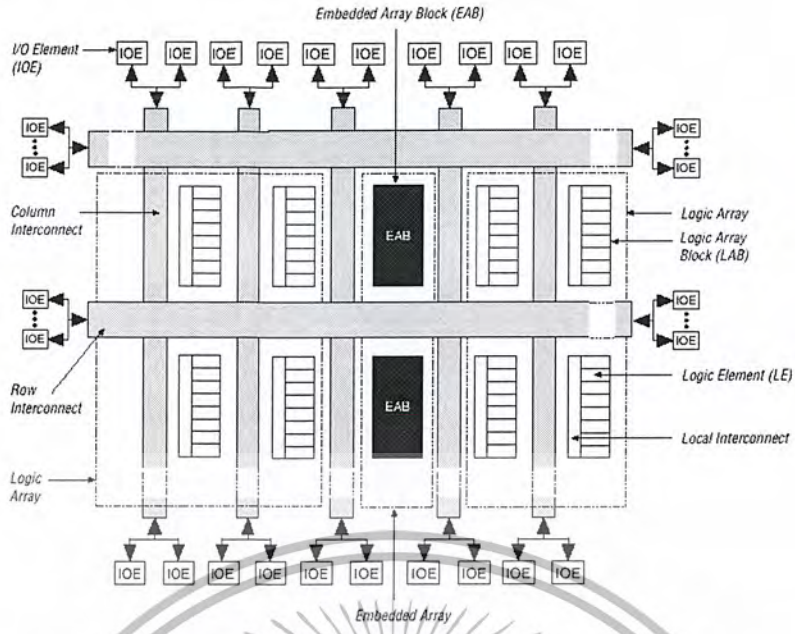
2.2 SRAM - Base FPGA

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

ลักษณะโครงสร้างภายในเป็นอาร์เรย์ลอจิกบ๊อกซ์ที่สามารถ ทำการโปรแกรม ได้ดังแสดงในรูปที่ 4.5



รูปที่ 4.5 โครงสร้างภายในของ FPGA ตระกูล MAX7000S



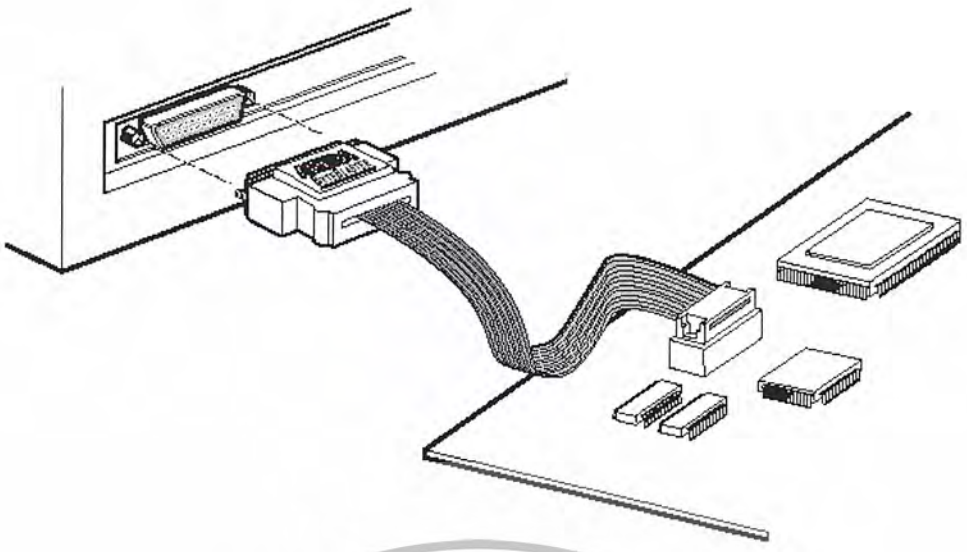
รูปที่ 4.6 โครงสร้างภายในของ FPGA ตระกูล FLEK10K

4.3.2 ทำไมการออกแบบถึงทำได้ง่ายและสะดวกรวดเร็ว

1.ในการออกแบบเราไม่จำเป็นต้องรู้ถึง โครงสร้างภายในของตัวชิพ 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 4.7 การโปรแกรมลงในชิพ

4.3.3 การออกแบบโดยใช้ภาษาอธิบายพฤติกรรมของฮาร์ดแวร์ (HDL)

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

4.3.4 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** ใน

ขั้นตอนนี้ซอร์พแวร์สังเคราะห์วงจรจะทำการแปลงโค้ด 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

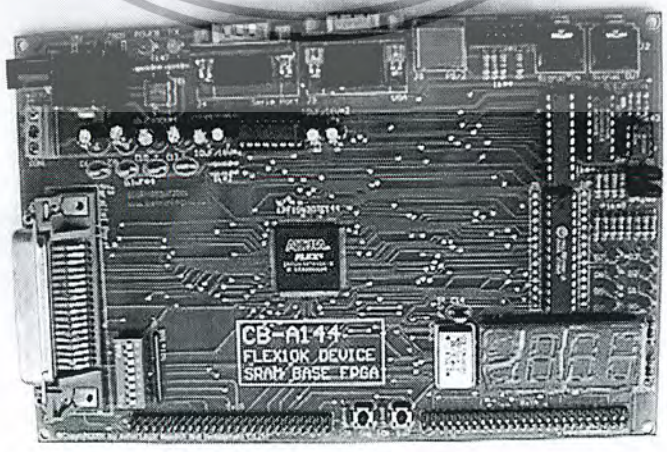
บทที่ 5
การออกแบบเครื่องแสดงผลระดับสัญญาณเสียง

5.1 บอร์ดทดลอง MASTER FLEX – A01

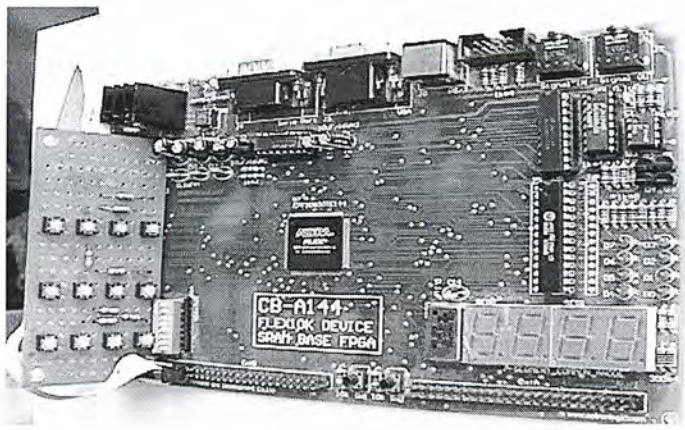
บอร์ดทดลอง MASTER FLEX – A01 นี้เป็นบอร์ดทดลอง FPGA ที่ออกแบบมาสำหรับใช้งานกับชิฟเอฟพีจีเอในตระกูล FLEX10K เบอร์ EPF10K20TC144 เป็นเอฟพีจีเอที่มีความจุของเกทประมาณ 20,000 เกท ด้วยความจุของเกทที่มีมากถึง 20,000 เกททำให้สามารถออกแบบวงจรหรือระบบดิจิตอลที่มีขนาดใหญ่หรือซับซ้อนได้ดี อีกทั้งภายในบอร์ด MASTER FLEX – A01 ยังประกอบไปด้วยชุดอินเตอร์เฟสที่สำคัญอีกหลายตัวทำให้การออกแบบและใช้งานชิฟเอฟพีจีเอกับอุปกรณ์ภายนอกสามารถทำได้สะดวกมากยิ่งขึ้น

รูปที่ 5.1 ชิฟเอฟพีจีเอในตระกูล FLEX10K เบอร์ EPF10K20TC144

สามารถแสดงโครงสร้างของบอร์ด MASTER FLEX – A01 ในลักษณะของมุมมอง Perspective View และ Top View ได้ดังรูปที่ 5.2 และรูปที่ 5.3



รูปที่ 5.2 บอร์ด MASTER FLEX – A01 ในมุมมอง Perspective

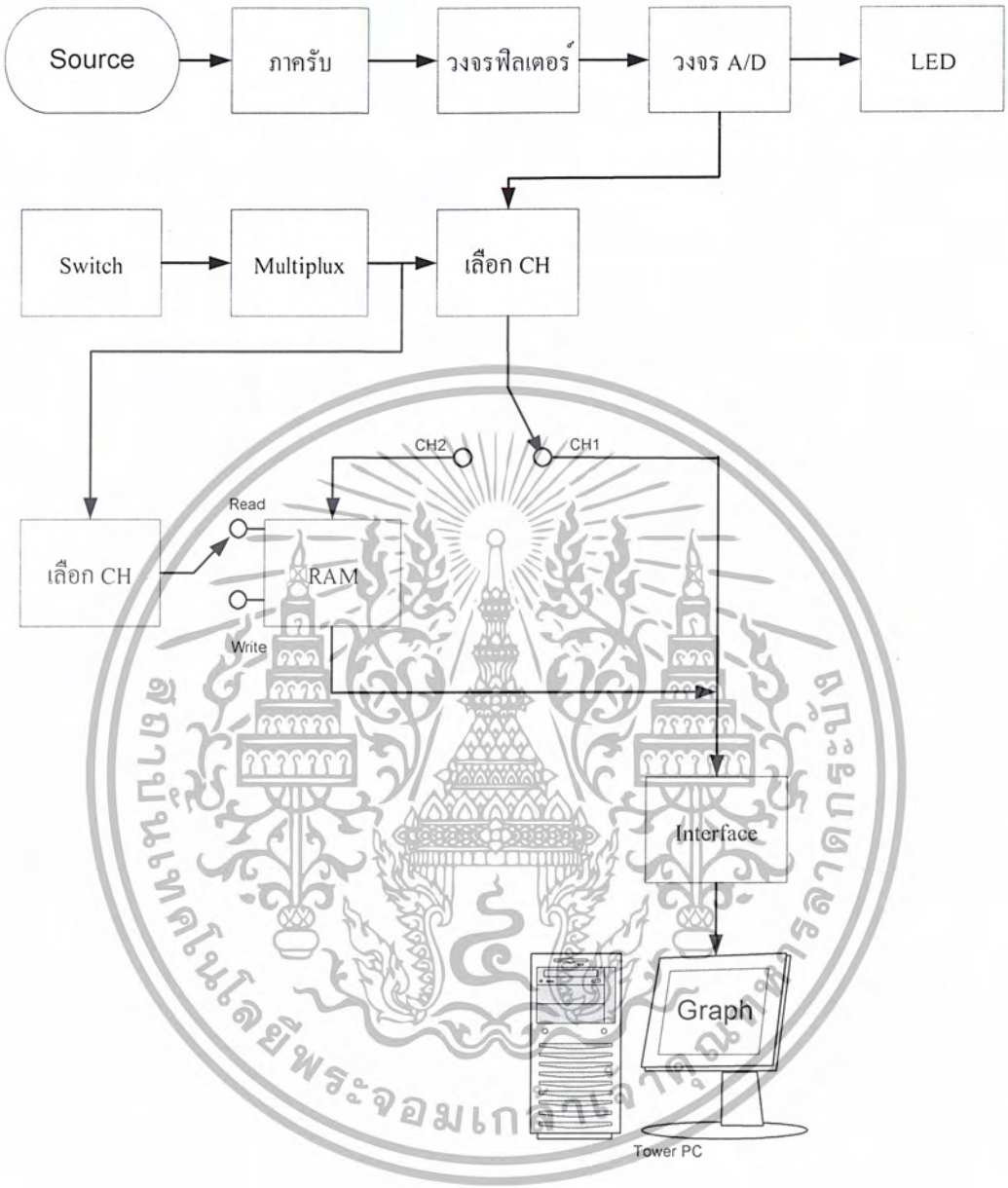


รูปที่ 5.3 บอร์ด MASTER FLEX – A01 ในมุมมองของ Top View

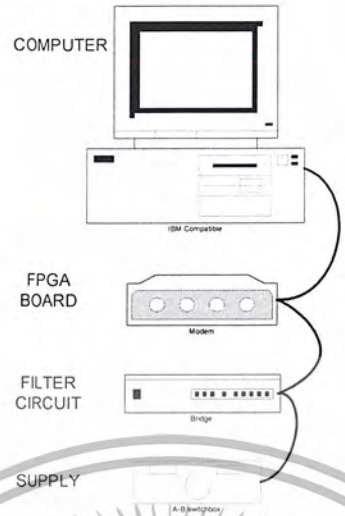
ภายในบอร์ด MASTER FLEX – A01 จะประกอบด้วย

- วงจรรวมตระกูล FLEX10K ในอนุกรม EPF10K20TC144
- JTAG CONNECTOR
- พอร์ตขยายช่องสัญญาณขนาด 26x2 2ชุด
- คอนเน็กเตอร์สำหรับส่งข้อมูลอนุกรม แบบ DB9 ตัวผู้ 1 ชุด พร้อมไอซีตัวแปลงระดับสัญญาณดิจิตอลเป็นระดับของ RS-232
- คอนเน็กเตอร์สำหรับเชื่อมต่อกับจอVGA
- คอนเน็กเตอร์แบบ PS/2
- คอนเน็กเตอร์แบบ Centronics Port สำหรับส่งข้อมูลแบบขนาน
- Analog to Digital Converter ขนาด 8บิต
- Digital to Analog Converter ขนาด 8บิต
- หน่วยความจำแบบ SRAM 64 KByte
- สวิตช์แบบกดติด – ปล่อยดับจำนวน 2 ตัว
- ดิพสวิตช์ 8 บิต 1 ชุด
- ไดโอดเปล่งแสง 8 ดวง
- ชุดแสดงผล 7-Segment 4 หลักขนิด Common Anode แบบ Multiplex
- โมดูลออสซิลเลเตอร์ความถี่ 25.175 MHz 1 ชุด
- คอนเน็กเตอร์แหล่งจ่ายไฟให้กับบอร์ด สำหรับไฟกระแสตรงหรือกระแสสลับแรงดัน 7 ถึง 12 โวลท์
- คอนเน็กเตอร์แหล่งจ่ายไฟให้กับบอร์ด สำหรับไฟกระแสตรง +-15โวลท์

5.2 ขั้นตอนการออกแบบ



รูปที่ 5.4 บล็อกไดอะแกรมของการออกแบบเครื่องแสดงผลระดับสัญญาณเสียง



รูปที่ 5.5 ส่วนประกอบของเครื่อง

ขั้นตอนการออกแบบสามารถแบ่งได้ 3 ส่วนคือ

1) ขั้นตอนการออกแบบส่วนของ Hard ware

ออกแบบวงจรฟิลเตอร์

2) ขั้นตอนการออกแบบส่วนของ FPGA

2.1) เขียนโปรแกรมภาษา VHDL รับข้อมูลจาก A/D

2.2) เขียนโปรแกรมภาษา VHDL รับค่าจากสวิทช์เข้าวงจร Multiplex

2.3) เขียนโปรแกรมภาษา VHDL เพื่อเก็บข้อมูลลงใน RAM

2.4) เขียนโปรแกรมภาษา VHDL เพื่อ Interface ไปยัง Computer

2.5) เขียนโปรแกรมภาษา VHDL สร้าง Clock 19.2KHz เพื่อสื่อสารแบบอนุกรม

2.6) เขียนโปรแกรมภาษา VHDL สร้าง Clock 200KHz เพื่อ Sampling ข้อมูลใน A/D และเพื่อเป็น Clock ให้แก่ RAM

2.7) วาดวงจรใน Graphic Editor file โดยนำ Graphic จากข้อ 2.1) ถึง 2.6) มาต่อกัน

3) ขั้นตอนการออกแบบส่วนของ Visual Basic

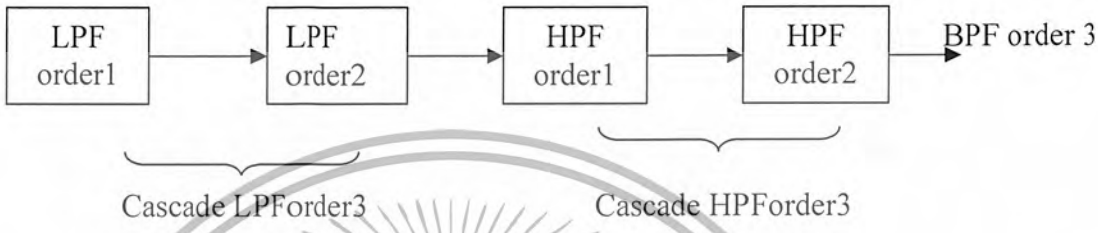
3.1) เขียนโปรแกรม Visual Basic เพื่อรับข้อมูลจาก FPGA

3.2) เขียนโปรแกรม Visual Basic เพื่อนำข้อมูลไปพล็อตกราฟแสดงระดับของสัญญาณ

5.3 ขั้นตอนการออกแบบส่วนของ Hard ware

5.3.1 การออกแบบของวงจรแบนพาสฟิลเตอร์

ใช้วงจร LPF order 3 cascade กับวงจร HPF order 3 จะได้เป็นวงจรแบนพาสฟิลเตอร์ order 3 เนื่องจากต้องการช่วงของความถี่ ตั้งแต่ fp =300 -3 KHZ ผ่านออกไปใช้งานได้ จะผลของ Qp ที่ ต่ำมากๆ



รูปที่ 5.6 วงจร LPF order 3 cascade กับวงจร HPF order 3 เป็นวงจร BPF order 3

ในส่วนของการออกแบบวงจร LPF ส่งผ่านความถี่ที่ต่ำกว่า 3 kHZ โดยมี Loss ไม่เกิน 3 dB และมี Pass band Gain เป็น 2 เท่า กำจัดความถี่มากกว่า 8 kHZ อย่างน้อย 18 dB ใช้ Filter order 3 Normalized Butterworth LPF function

ตารางที่ 2.1 Normalized Butterworth LPF Loss order 1-5

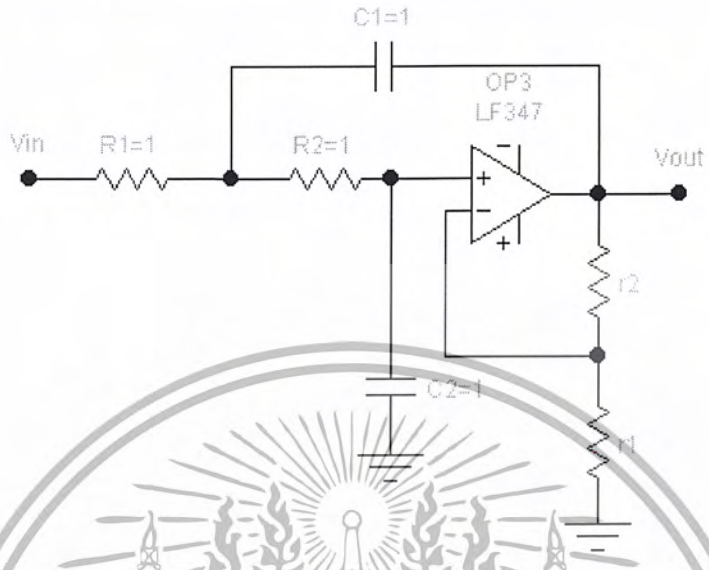
N	Normalized Butterworth LPF function	Q of section ortder 2
1	$(s+1)$	-
2	$(s^2 + 1.4142s + 1)$	0.707
3	$(s+1)(s^2 + s + 1)$	1
4	$(s^2 + 0.7654s + 1)(s^2 + 1.8478s + 1)$	1.307,0.5412
5	$(s + 1)(s^2 + 0.6180s + 1)(s^2 + 1.6180s + 1)$	1.618,0.618

$$H(S) = T_{LPF} \text{ Gain} = \frac{1}{(S+1)(S^2+S+1)}.$$

เราสามารถสร้าง Filter order 3 โดยการ cascade กันของ $\frac{1}{S+1} \cdot \frac{1}{S^2+S+1}$

เราจะทำการพิจารณาส่วนของ Second order Gain = $\frac{1}{(S+1)(S^2+S+1)}$

จะได้ $\omega_p = 1$ R/Sec ดังนั้นจะได้ Normalized LPF circuit เป็น เป็นวงจรรูปที่ 5.7



รูปที่ 5.7 Normalized LPF circuit order 2

Normalized LPF ที่มีค่า Qp = 1 และ Gain มีค่าเป็น 1 เท่า แต่จากที่เราต้องการ ในส่วนของ Second order มี Gain = 2 ดังนั้น ในส่วนของ First order จะมี Gain = 1 เท่า

จะได้ Normalized LPF ที่ต้องการ

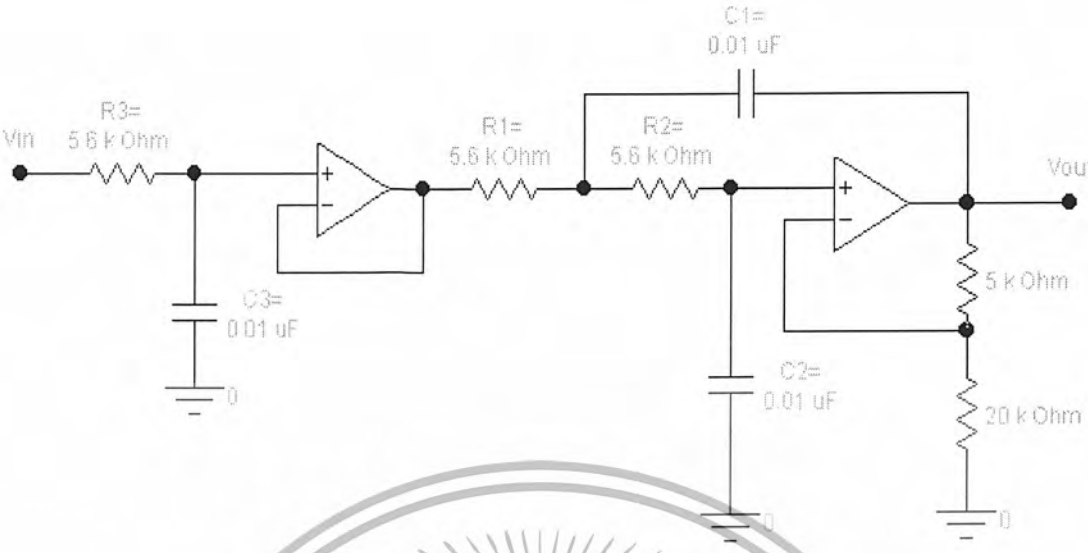
Frequency scaling โดยลดค่า C ลง $2\pi 3000$ เท่าได้

R1 =R2 =R3 = $1/\omega_p$ = $1/(2\pi 3000)$ = 53.0516 *E-06 Ω

C1 = C2 = C3 = 1F

Impedance scaling ลดค่า C1,C2,C3 เป็น 0.01 uF ลดค่าC ลง 1/0.01 uF เท่า

ดังนั้นเพิ่มค่า R1,R2,R3 เป็น 53.0516*E-06 × (1/0.01*E-06)= 5.3 KΩ $\approx$ 5.6 KΩ



รูปที่ 5.8 วงจร LPF Filter order 3

การออกแบบวงจร HPF เราสามารถเปลี่ยน LPF Function เป็น HPF Function โดยการแทนค่า S ใน LPF Function ด้วย ω_p/S ดังนั้นการออกแบบ Normalized LPF จะได้เป็น R_{LPF} แทนด้วย C ค่า 1/R , C_{LPF} แทนด้วย R ค่า 1/C จะได้วงจร Normalized HPF ซึ่งมีหัวข้อสรุปดังนี้

1) จากคุณสมบัติของระบบจะได้คุณสมบัติของ HPF ที่ต้องการ คือ

   ω_P , ω_S , A_{MAX} , A_{MIN}

2) จากคุณสมบัติของ HPF ในข้อ 1 หา Equivalent Normalized LPF ซึ่ง Equivalent Normalized LPF จะมีคุณสมบัติดังนี้

   ตารางที่ 2.2 Equivalent Normalized LPF มีคุณสมบัติดังนี้

HPF	Equivalent Normalized LPF
A_{MAX}	A_{MAX}
A_{MIN}	A_{MIN}
Ω_P	1
Ω_S	ω_P/ω_S

3) จาก Equivalent Normalized LPF หา LPF Function ที่สอดคล้องกับ Equivalent Normalized LPF นั้น (order ,type)

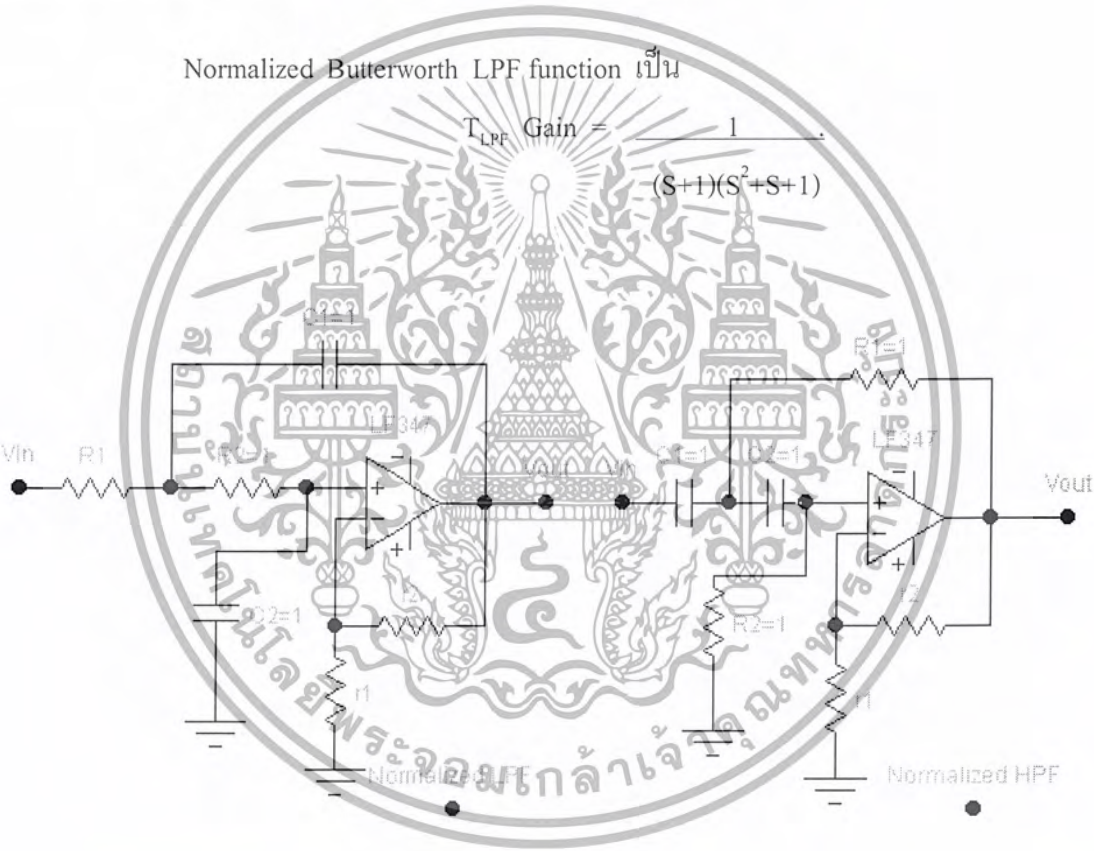
4) จาก Equivalent Normalized LPF Function จะได้วงจรของ Equivalent Normalized LPF

5) Transform วงจร Equivalent Normalized LPF เป็น Normalized HPF โดย R_{LPF} แทนด้วย $C_{HPF} = 1/R_{LPF}$ และ $R_{HPF} = 1/C_{LPF}$

6) Frequency/Impudence scaling จะได้วงจร HPF ที่ต้องการ

ในส่วนของวงจร High pass filter ก็จะยังคงอาศัยหลักการของการต่อแบบ คาสเคส(cacade) เช่นเดียวกับของ วงจร Low pass filter ซึ่งเรายังจะต่อเป็น order 3 เช่นเดียวกัน

การออกแบบวงจร HPF ส่งผ่านความถี่ที่สูงกว่า 300HZ โดยมี Loss ไม่เกิน 3 dB กำจัดความถี่มากกว่า 50 HZ อย่างน้อย 18 dB ใช้ Filter order 3



รูปที่ 5.9 Normalized LPF to HPF

เราสามารถสร้าง Filter order 3 โดยการ cascade กันของ $\frac{1}{S+1} \cdot \frac{1}{S^2+S+1}$

เราจะทำการพิจารณาส่วนของ Second order Gain = $\frac{1}{(S+1)(S^2+S+1)}$

$\Omega p = 1$, $\Omega_S = 2$, A_{max} ไม่เกิน 3 dB , Amin มากกว่า 18 dB fp =300 HZ , fs =50 HZ , $\epsilon = 1$

$$\text{Norm. } LPF_{BUTT} = \frac{1}{S+1} \cdot \frac{1}{S^2+S+1}$$

แทน $S = (\omega p/s) = 1{,}884.956\ /s$

$$\text{DENorm. } HPF_{BUTT} = \frac{s}{s+1{,}884} \cdot \frac{s^2}{s^2+1{,}884s+1884}$$

Qp = 1 , fp = 300 HZ

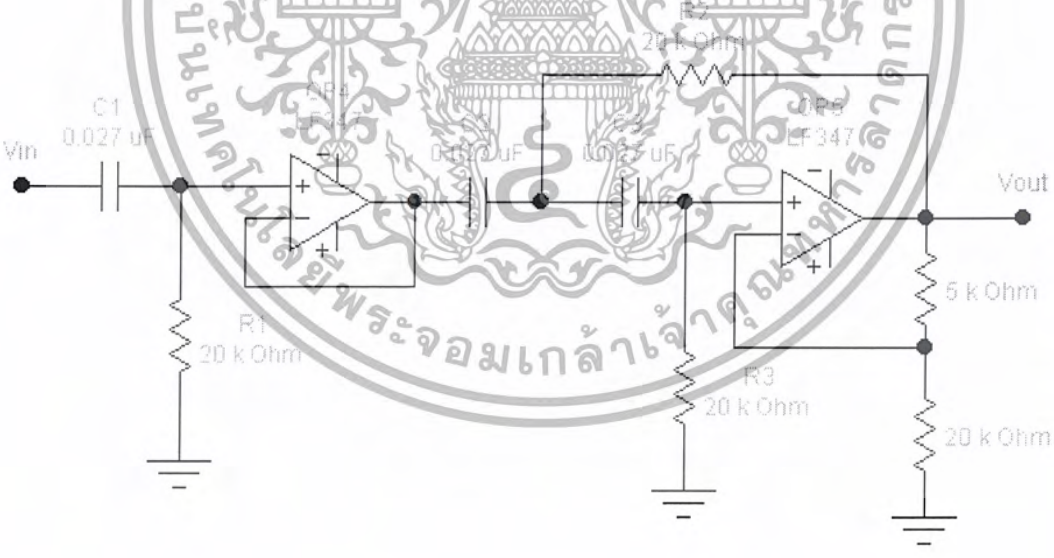
Frequency scaling โดยลดค่า C ลง $2\pi 300$ เท่าได้

$$R1 = R2 = R3 = 1/\omega p\ \Omega$$

$$C1 = C2 = C3 = 1F$$

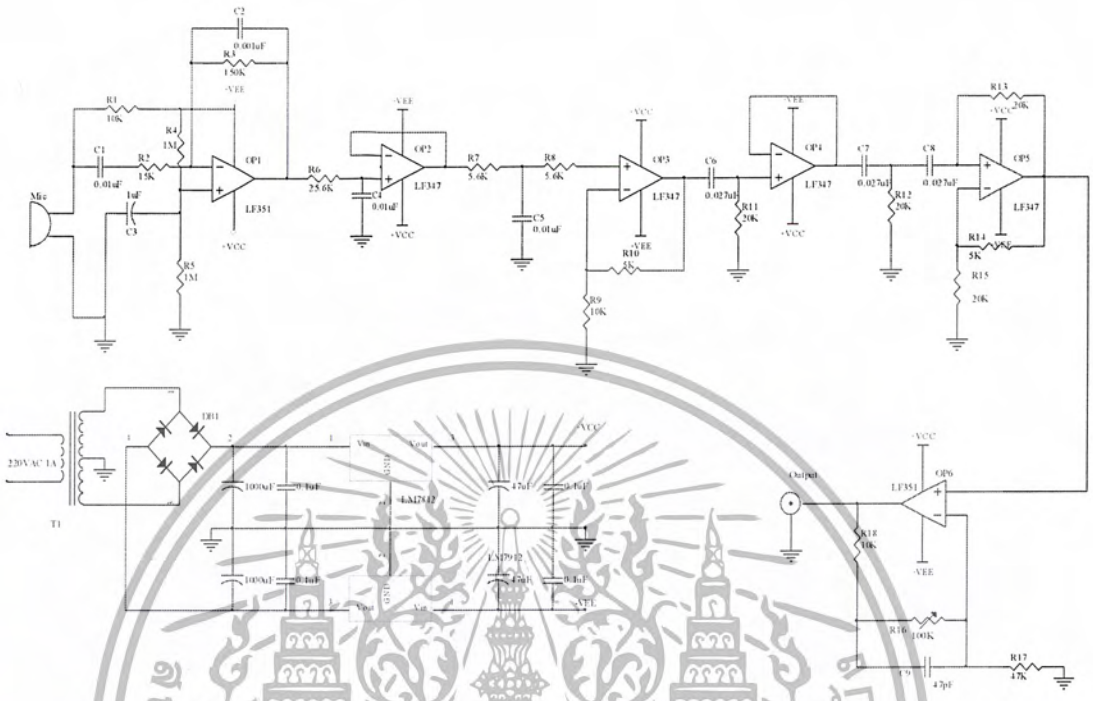
Impedance scaling ลดค่า C1,C2,C3 เป็น 0.027 uF ลดค่า C ลง 1/0.027 uF เท่า

ดังนั้นเพิ่มค่า R1,R2,R3 เป็น $1{,}884 \times 37{,}037{,}037 = 19.6\ K\Omega \approx 20\ K\Omega$



รูปที่ 5.10 วงจร HPF order 3

วงจร LPF order 3 cascade กับวงจร HPF order 3 จะได้เป็นวงจรแบนพาสฟิลเตอร์ order 3

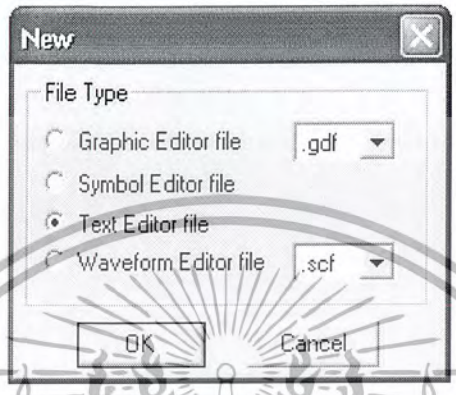


รูปที่ 5.11 วงจร BPF order 3 โดยมีวงจรขยายสัญญาณไมโครโฟน

5.4 ขั้นตอนการออกแบบส่วนของ FPGA

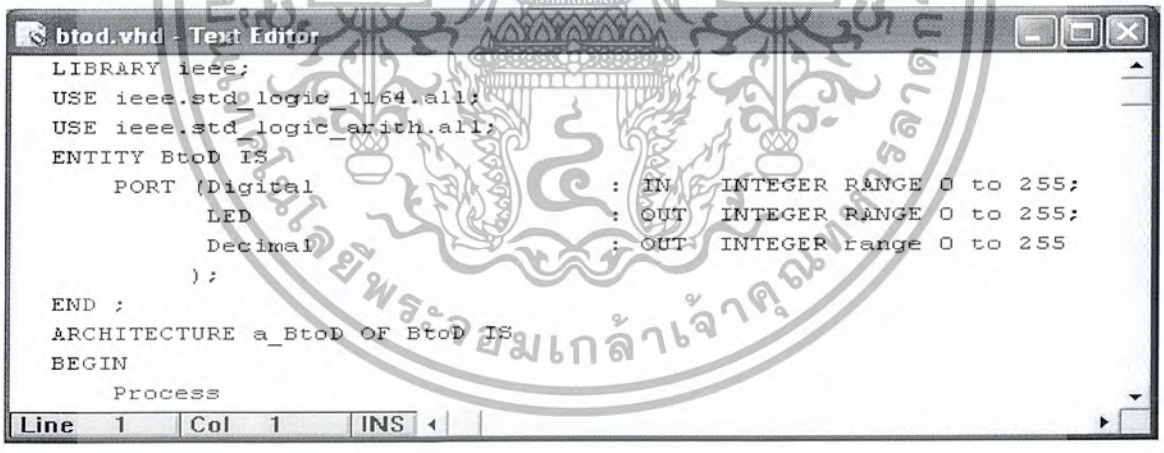
5.4.1 เขียนโปรแกรมภาษา VHDL รับข้อมูลจาก A/D

1) เปิดโปรแกรม MAX+PLUS II ขึ้นมาตั้งชื่อเป็นโปรเจ็คเป็นชื่อ BtoD เลือกชนิดของไฟล์ที่จะสร้างเป็น Text Editor Fileดังรูปที่ 5.12



รูปที่ 5.12 เลือกชนิดของFile ที่จะสร้าง

2) พิมพ์โค้ดภาษา VHDL ที่ทำหน้าที่เป็น วงจร BtoD ดังโปรแกรมด้านล่าง หลังจากนั้นบันทึกเป็น File ชื่อ BtoD.vhd

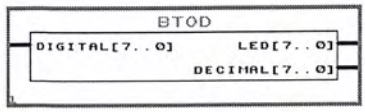


รูปที่ 5.13 โค้ดภาษา VHDL ที่ทำหน้าที่เป็นวงจร BtoD

3)เลือก Device เป็น FLEX10K เบอร์ชิฟ FPGA เป็น EPF10K20TC144-4 หลังจากนั้นทำการคอมไพล์วงจรที่สร้างเสร็จเรียบร้อยแล้ว

4) สร้างไฟล์ Waveform ขึ้นมาแล้วบันทึกในชื่อ BtoD.scf สำหรับจำลองการทำงาน

5) จะได้ Graphic Editor file ดังนี้

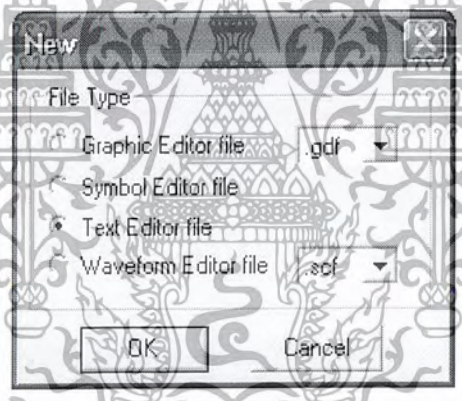


รูปที่ 5.14 Graphic Editor

6) จัดตำแหน่งขาของชิฟ FPGA ให้ตรงกับอินพุทและเอาท์พุทของพอร์ทที่ต้องการ แล้ว Save และ Complie ใหม่อีกครั้ง

5.4.2 เขียนโปรแกรมภาษา VHDL รับค่าจากสวิทช์เข้าวงจร Multiplex

1) เปิดโปรแกรม MAX+PLUS II ขึ้นมาตั้งชื่อเป็นโปรเจ็คเป็นชื่อ Multiplex เลือกชนิดของไฟล์ที่จะสร้างเป็น Text Editor Fileดังรูปที่ 5.15



รูปที่ 5.15 เลือกชนิดของFile ที่จะสร้าง

2) พิมพ์โค้ดภาษา VHDL ที่ทำหน้าที่เป็น วงจร Multiplex ดังโปรแกรมด้านล่าง หลังจากนั้นบันทึกเป็น File ชื่อ Multiplex.vhd

```
Library ieee;
Use ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
Entity Multiplex is
Port  (          sw      :      in bit_vector(0 to 6);
                                data  : out integer range 0 to 6
```

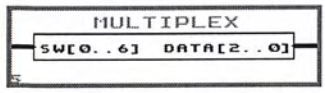
```
        );
End ;
Architecture a_Multiplex   of Multiplex is
Begin
Process
begin
if sw(0) = '0' then
        data <= 0 ;
elsif sw(1) = '0' then
        data <= 1 ;
elsif sw(2) = '0' then
        data <= 2 ;
elsif sw(3) = '0' then
        data <= 3 ;
elsif sw(4) = '0' then
        data <= 4 ;
elsif sw(5) = '0' then
        data <= 5 ;
elsif sw(6) = '0' then
        data <= 6 ;
end if;
end process;
end;
```

รูปที่ 5.16 โค้ดภาษา VHDL ที่ทำหน้าที่เป็นวงจร Multiplex

3) เลือก Device เป็น FLEX10K เบอร์ชิฟ FPGA เป็น EPF10K20TC144-4 หลังจากนั้นทำการคอมไพล์วงจรที่สร้างเสร็จเรียบร้อยแล้ว

4) สร้างไฟล์ Waveform ขึ้นมาแล้วบันทึกในชื่อ Multiplex.scf สำหรับจำลองการทำงาน

5) จะได้ Graphic Editor file ดังนี้



รูปที่ 5.17 Graphic Editor

6) จัดตำแหน่งขาของชิฟ FPGA ให้ตรงกับอินพุทและเอาท์พุทของพอร์ทที่ต้องการ แล้ว Save และ Complie ใหม่อีกครั้ง

5.4.3 เขียนโปรแกรมภาษา VHDL เพื่อเก็บข้อมูลลงใน RAM

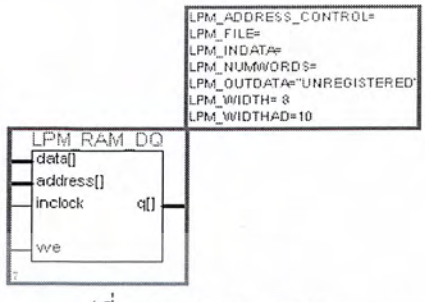
1) เปิดโปรแกรม MAX+PLUS II ขึ้นมาตั้งชื่อเป็นโปรเจ็คเป็นชื่อ RAM เลือกชนิดของไฟล์ที่จะสร้างเป็น Graphic Editor Fileดังรูปที่ 5.18

รูปที่ 5.18 เลือกชนิดของFile ที่จะสร้าง

2) เลือก Device เป็น FLEX10K เบอร์ชิฟ FPGA เป็น EPF10K20TC144-4 หลังจากนั้นทำการคอมไพล์วงจรที่สร้างเสร็จเรียบร้อยแล้ว

3) สร้างไฟล์ Waveform ขึ้นมาแล้วบันทึกในชื่อ RAM.scf สำหรับจำลองการทำงาน

4) จะได้ Graphic Editor file ดังนี้

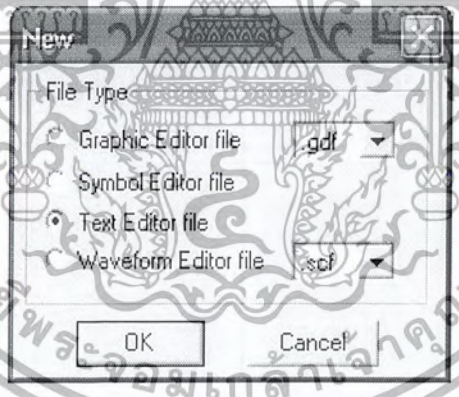


รูปที่ 5.19 Graphic Editor

5) นำRAM ไปใช้งานเป็นอุปกรณ์เก็บข้อมูลได้

5.4.4 เขียนโปรแกรมภาษา VHDL เพื่อ Interface ไปยัง Computer

1) เปิดโปรแกรม MAX+PLUS II ขึ้นมาตั้งชื่อเป็นโปรเจ็คเป็นชื่อ COMM_SERIAL เลือกชนิดของไฟล์ที่จะสร้างเป็น Text Editor Fileดังรูปที่ 5.20



รูปที่ 5.20 เลือกชนิดของFile ที่จะสร้าง

2) พิมพ์โค้ดภาษา VHDL ที่ทำหน้าที่เป็น วงจร COMM_SERIAL ดังโปรแกรมด้านล่าง หลังจากนั้นบันทึกเป็น File ชื่อ COMM_SERIAL.vhd

```
library ieee;
use ieee.std_logic_1164.all;

entity COMM_SERIAL is
```

```
port(
        BAUD_RATE       : in std_logic;
        RX              : in std_logic;
        DATA_TX     : in std_logic_vector(7 downto 0);
        TX              : out std_logic
        );
end COMM_SERIAL;

architecture rtl of COMM_SERIAL is
type    State_Type_RX is (Idel,ReceiveData,Stop);
type    State_Type_TX is (Idel,Start,TransData,Stop);
signal  State_RX : State_Type_RX := Idel;
signal  State_TX : State_Type_TX;
begin
        process(BAUD_RATE,DATA_TX)--,TX_EN
        variable TX_Data_Count : integer range 0 to 7 := 0;
        variable count : integer range 0 to 7;
        begin
            if BAUD_RATE'Event and BAUD_RATE = '1' then
                case State_TX is
                    when Idel =>
                        TX <= '1';
                        State_TX <= Start;
                    when Start =>
                        TX <= '0';
                        TX_Data_Count := 0;
                        State_TX <= TransData;
                    when TransData =>
                        if TX_Data_Count = 7 then
                            TX <= DATA_TX(TX_Data_Count);
```

```
                    State_TX <= Stop;
                else
                    TX <= DATA_TX(TX_Data_Count);
                TX_Data_Count := TX_Data_Count + 1;
                    State_TX <= TransData;
                end if;
            when others =>
                TX_Data_Count := 0;
                TX <= '1';
                if count=7 then
                    count:=0;
                    State_TX <= Idel;
                else
                    state_TX <=stop;
                    count:=count+1;
                end if;
            end case;
        end if;
    end process;

    process(BAUD_RATE,RX)
    variable RX_Data_Count : integer range 0 to 7 := 0;
    variable Buffer_RX : std_logic_vector(7 downto 0);
    begin
        if BAUD_RATE'Event and BAUD_RATE = '1' then
            case State_RX is
                when Idel =>
                    if RX = '0' then
                        RX_Data_Count := 0;
```

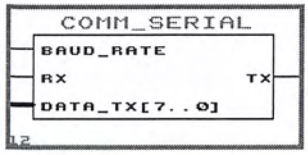
```
                    State_RX <= ReceiveData;
                else
                    RX_Data_Count := 0;
                    State_RX <= Idel;
                end if;
            when ReceiveData =>
                if RX_Data_Count = 7 then
                    Buffer_RX(RX_Data_Count) := RX;
                    State_RX <= Stop;
                else
                    Buffer_RX(RX_Data_Count) := RX;
                    RX_Data_Count := RX_Data_Count + 1;
                    State_RX <= ReceiveData;
                end if;
            when Stop =>
                RX_Data_Count := 0;
                State_RX <= Idel;
            when others =>
                RX_Data_Count := 0;
                State_RX <= Idel;
        end case;
    end if;
end process;
end rtl;
```

รูปที่ 5.21 โค้ดภาษา VHDL ที่ทำหน้าที่เป็นวงจร COMM_SERIAL

3) เลือก Device เป็น FLEX10K เบอร์ชิฟ FPGA เป็น EPF10K20TC144-4 หลังจากนั้นทำการคอมไพล์วงจรที่สร้างเสร็จเรียบร้อยแล้ว

4) สร้างไฟล์ Waveform ขึ้นมาแล้วบันทึกในชื่อ COMM_SERIAL.scf สำหรับจำลองการทำงาน

5) จะได้ Graphic Editor file ดังนี้

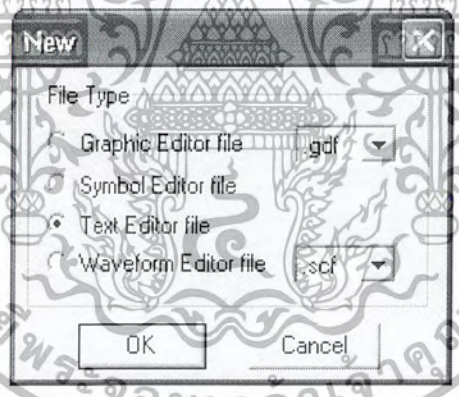


รูปที่ 5.22 Graphic Editor

6) จัดตำแหน่งขาของชิฟ FPGA ให้ตรงกับอินพุทและเอาท์พุทของพอร์ทที่ต้องการ แล้ว Save และ Complie ใหม่อีกครั้ง

5.4.5 เขียนโปรแกรมภาษา VHDL สร้าง Clock 19.2KHz เพื่อสื่อสารแบบอนุกรม

1) เปิดโปรแกรม MAX+PLUS II ขึ้นมาตั้งชื่อเป็นโปรเจ็คเป็นชื่อ Div500 เลือกชนิดของไฟล์ที่จะสร้างเป็น Text Editor File ดังรูปที่ 5.23



รูปที่ 5.23 เลือกชนิดของFile ที่จะสร้าง

2) พิมพ์โค้ดภาษา VHDL ที่ทำหน้าที่เป็น วงจร Div500 ดังโปรแกรมด้านล่าง หลังจากนั้นบันทึกเป็น File ชื่อ Div500.vhd

```
library ieee;
use ieee.std_logic_1164.all;
entity Div500 is
port(          Clk_in : in std_logic;
                 Clk_out : out std_logic );
```

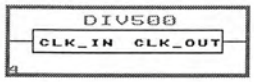
```
end Div500;

architecture rtl of Div500 is
begin
process(Clk_in)

variable Clk_temp : std_logic := '0';
variable count : integer range 0 to 499 := 0;

begin
if Clk_in'Event and Clk_in = '1' then
      if count < 249 then
            count := count + 1;
            Clk_temp := Clk_temp;
      else
            count := 0;
            Clk_temp := not(Clk_temp);
      end if;
      Clk_out <= Clk_temp;
end if;
end process;
end rtl;
```

รูปที่ 5.24 โค้ดภาษา VHDL ที่ทำหน้าที่เป็นวงจร Div500

3) เลือก Device เป็น FLEX10K เบอร์ชิฟ FPGA เป็น EPF10K20TC144-4 หลังจากนั้นทำการคอมไพล์วงจรที่สร้างเสร็จเรียบร้อยแล้ว

4) สร้างไฟล์ Waveform ขึ้นมาแล้วบันทึกในชื่อ Div500.scf สำหรับจำลองการทำงาน

5) จะได้ Graphic Editor file ดังนี้



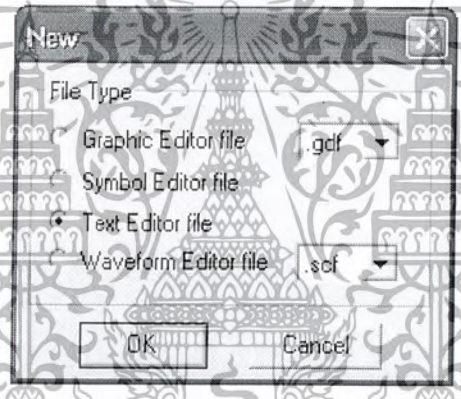
รูปที่ 5.25 Graphic Editor

6) จัดตำแหน่งขาของชิฟ FPGA ให้ตรงกับอินพุทและเอาท์พุทของพอร์ทที่ต้องการ แล้ว Save และ Complie ใหม่อีกครั้ง

5.4.6 เขียนโปรแกรมภาษา VHDL สร้าง Clock 200KHz

เพื่อ Sampling ข้อมูลใน A/D และเพื่อเป็น Clock ให้แก่ RAM

1) เปิดโปรแกรม MAX+PLUS II ขึ้นมาตั้งชื่อเป็นโปรเจ็คเป็นชื่อ Clock200k เลือกชนิดของไฟล์ที่จะสร้างเป็น Text Editor Fileดังรูปที่ 5.26



รูปที่ 5.26 เลือกชนิดของFile ที่จะสร้าง

2) พิมพ์โค้ดภาษา VHDL ที่ทำหน้าที่เป็น วงจร Clock200k ดังโปรแกรมด้านล่าง หลังจากนั้นบันทึกเป็น File ชื่อ Clock200k.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY Clock200k IS
    PORT (clk                   : IN  std_logic;
          Clk200k               : OUT std_logic
         );
END Clock200k;
```

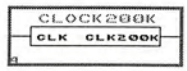
```
ARCHITECTURE rtl OF Clock200k IS
BEGIN
Process(clk)

variable Clk_temp : std_logic := '0';
Variable cnt : integer range 0 to 23;

Begin

if clk'event and clk='1' then
    if  cnt<=23 then
            cnt := cnt + 1;
            Clk_temp := Clk_temp;
    else
            cnt := 0;
            Clk_temp := not(Clk_temp); -- The Idia is here.
    end if;
    Clk200k <= Clk_temp;
end if;

End Process;
END ;
```

รูปที่ 5.27 โค้ดภาษา VHDL ที่ทำหน้าที่เป็นวงจร Clock200k

3) เลือก Device เป็น FLEX10K เบอร์ชิฟ FPGA เป็น EPF10K20TC144-4 หลังจากนั้นทำการคอมไพล์วงจรที่สร้างเสร็จเรียบร้อยแล้ว

4) สร้างไฟล์ Waveform ขึ้นมาแล้วบันทึกในชื่อ Clock200k.scf สำหรับจำลองการทำงาน

5) จะได้ Graphic Editor file ดังนี้

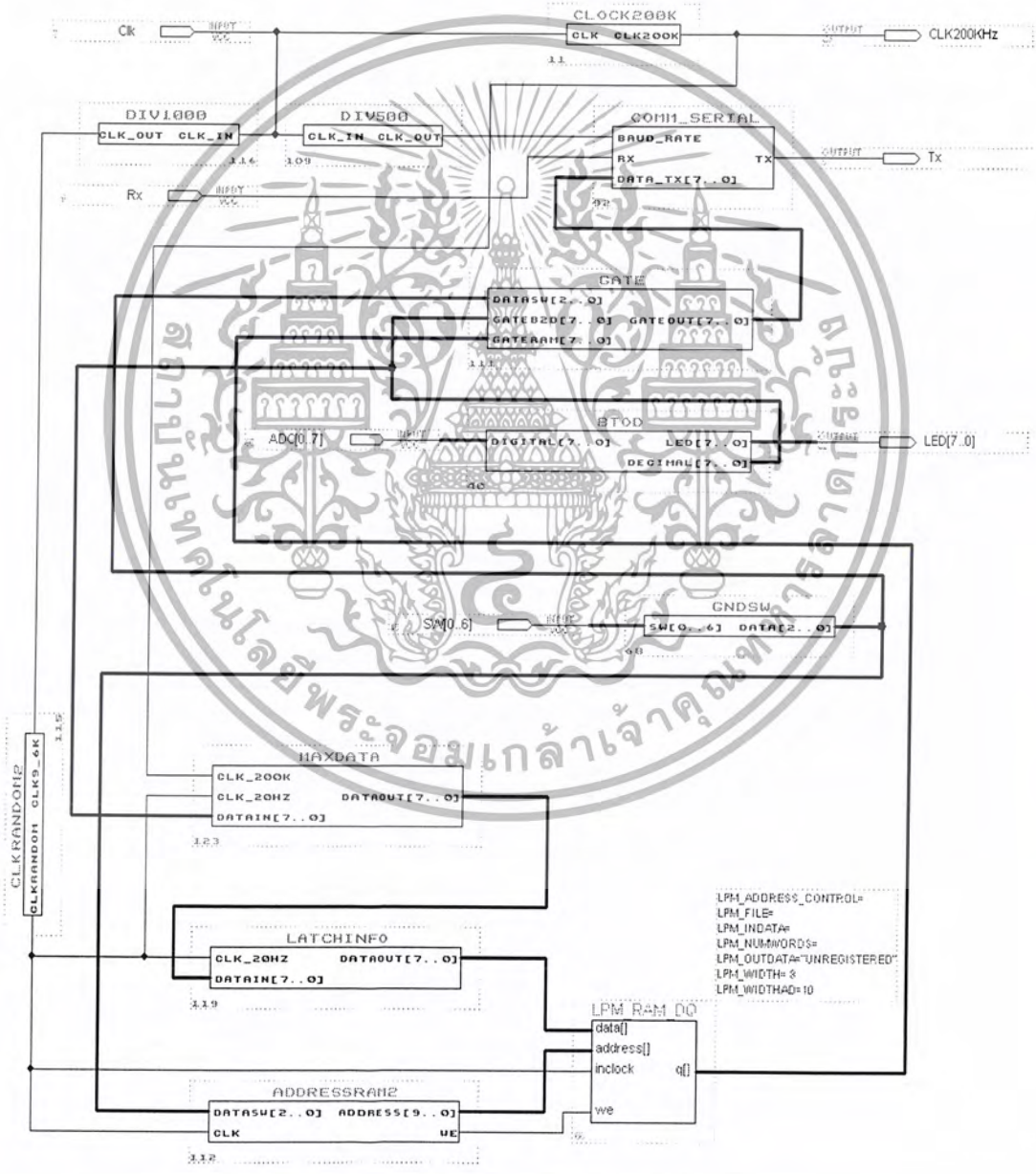


รูปที่ 5.28 Graphic Editor

6) จัดตำแหน่งขาของชิฟ FPGA ให้ตรงกับอินพุทและเอาท์พุทของพอร์ทที่ต้องการ แล้ว Save และ Complie ใหม่อีกครั้ง

5.4.7 วาดวงจรใน Graphic Editor file

วาดวงจรใน Graphic Editor file โดยนำ Graphic จากข้อ 2.1) ถึง 2.6) มาต่อกัน



รูปที่ 5.29 การออกแบบวงจรใน Graphic Editor file

อธิบายวงจรใน Graphic Editor file

Clock 200KHz จะกระตุ้นการทำงานของวงจรอนาลอกเป็นดิจิตอล และกระตุ้นการทำงานของ RAL (LMP_RAM_DQ) และ ADDRESSRAM ส่วน DIV500 จะเป็น Clock ที่ใช้ในการสื่อสารแบบอนุกรมซึ่งเราสาสมารถเปลี่ยนโหมดการสื่อสารได้จากตรงนี้ เมื่อเรากดสวิทช์เลือกสภาวะการทำงานของวงจรเช่น กดปุ่ม 0 จะเป็นการส่งข้อมูลที่ประมวลผลได้ขึ้นไปแสดงผลโดยทันทีโดยไม่ผ่าน RAM กล่าวคือข้อมูลจะผ่าน GATE ไปเข้าสู่ COMM_SERIAL เพื่อทำการส่งข้อมูลออกไปยังคอมพิวเตอร์เพื่อพล็อตกราฟ และหากกดปุ่ม 1 จะเป็นการเก็บข้อมูลเขียนลงใน RAM ซึ่งจังหวะนี้จะไม่เห็นข้อมูลใดออกกราฟ แต่หากเมื่อใดที่เรากดปุ่ม 2 ข้อมูลภายใน RAM จะถูกอ่านออกมา ผ่าน GATE ไปเข้าสู่ COMM_SERIAL เพื่อทำการส่งข้อมูลออกไปยังคอมพิวเตอร์เพื่อพล็อตกราฟ

RAM ที่ใช้นั้นเป็น RAM ที่อยู่ภายใน FPGA ขนาดของ RAM นั้นขึ้นกับเบอร์ของ FPGA ซึ่งแต่ละเบอร์จะมีขนาดแตกต่างกันออกไป โดยการใช้งานนั้นเราจะต้องกำหนด Address ให้กับ RAM เพื่อกำหนดที่อยู่ของข้อมูล ซึ่งวงจรที่ทำหน้าที่นี้ก็คือ วงจร ADDRESSRAM โดยจะคอยตรวจสอบดูว่า Clock ขาลงมาแล้วหรือยัง เพราะหากเป็น Clock ขาลงมาเมื่อไร Address ก็จะเพิ่มขึ้นทีละ 1 ไปจนถึง 1023 Address เหตุผลที่เลือกแค่นี้เพราะว่า RAM ตัวนี้มีความสามารถในการเก็บข้อมูลที่จำกัด จากวงจรพบว่าออกแบบสำหรับเก็บข้อมูลขนาด 8 บิต เพื่อเก็บเข้า Address ขนาด 1023 Address โดยเวลาที่ใช้ในการเก็บนั้นจะเท่ากับเวลาที่ใช้ในการอ่านซึ่งระยะเวลาคิดได้จากค่าส่วนกลับของความถี่คูณด้วยจำนวนของ Address นั่นเอง

วงจร SW จะทำหน้าที่รับข้อมูลที่กดเข้ามาจากฮาร์ดแวร์ โดยรับเข้ามาเป็นรหัสแบบดิจิตอล จากนั้น วงจร SW จะทำการ Multiplex ข้อมูลให้เป็นเลขฐานสิบ เพื่อให้ง่ายแก่การออกแบบใช้งานและเพื่อนำไปใช้บังคับให้วงจรที่เกี่ยวข้องสามารถทำงานตามที่สั่งการเข้ามาจากภายนอก

5.5 ขั้นตอนการออกแบบส่วนของ Visual Basic

ส่วนนี้จะทำหน้าที่รับข้อมูล FPGA เข้ามาเพื่อที่จะแสดงผลออกที่กราฟโดยมีขั้นตอนดังนี้

1.เรียกใช้งาน Visual Basic

2.จากไดอะล็อก New Project เลือกสร้างชนิดโปรเจ็กต์เป็น Standard EXE

3.ออกแบบหน้าต่างแอพพลิเคชั่นดังนี้

รูปที่ 5.30 หน้าต่างแอพพลิเคชั่น

4. กำหนด Properties ให้กับ ActiveX Control ต่างๆดังนี้

Object1	Properties	Value	Object2	Properties2	Value2
From	Name	From1	TextBox	Name	Txt3
	Caption	Graph		Text	-
ComboBox	Name	Combo1	Frame	Name	Frame1
	Caption	COM1		Caption	Received
Botton	Name	Command8	Botton	Name	Command1
	Caption	Connect		Caption	START
TextBox	Name	Text1	Botton	Name	Command2
	Text	19200,n,8,1		Caption	STOP
Botton	Name	Command3	OptionBotton	Name	Opt1000
	Caption	Disconnect		Caption	Option1
MSComm	Name	MSComm1	OptionBotton	Name	Opt500
	Settings	19200,n,8,1		Caption	Option2
Timer	Name	Timer1	OptionBotton	Name	Opt300
	Interval	0		Caption	Option3
Label	Name	Label1	OptionBotton	Name	Opt200
	Caption	Comport		Caption	Option4
Label	Name	Label2	OptionBotton	Name	Opt100
	Caption	Setting		Caption	Option5
Frame	Name	Frame2	OptionBotton	Name	Opt50
	Caption	Communication		Caption	Option6
TextBox	Name	Txt2	Tchart	Name	TChart1
	Text	-		TimerInterval	1000

5.เขียนโค้ดเพื่อควบคุมการทำงานของโปรแกรมดังนี้

```
Public buffer  As Variant
Public amplitude As Variant
Public INTTEST As Integer
Public buffer2  As String
Dim a(256)  As Integer
Public cnt As Integer
Public amplitude2  As String
Private Sub Command1_Click()
        Timer1.Interval = 300
        Timer1.Enabled = True          'START
End Sub

Private Sub Command2_Click()
        Timer1.Enabled = False         'STOP
End Sub
Private Sub Command3_Click()
        If MSComm1.PortOpen = True Then
            MSComm1.PortOpen = False   'Disconnect
        amplitude = 0
        End If
End Sub
Private Sub Opt100_Click()
        Timer1.Interval = 100
End Sub
Private Sub Opt1000_Click()
        Timer1.Interval = 1000
End Sub
Private Sub Opt200_Click()
        Timer1.Interval = 200
```

```
End Sub
Private Sub Opt300_Click()
        Timer1.Interval = 300
End Sub
Private Sub Opt50_Click()
        Timer1.Interval = 1
End Sub
Private Sub Opt500_Click()
        Timer1.Interval = 500
End Sub
Private Sub Command8_Click()
        cnt = 0
        On Error GoTo Errlabel
```

ตั้งค่า Setting คือ Baud Rate(อัตราการรับส่งข้อมูล)

```
        MSComm1.Settings = Text1.Text
        MSComm1.CommPort = 1
        MSComm1.RThreshold = 1
        MSComm1.PortOpen = True
        MSComm1.InputLen = 0
    Exit Sub
```

สำหรับตรวจสอบข้อผิดพลาดของโปรแกรม

```
Errlabel:   If Err.Number = 8002 Then MsgBox "Select com Port", vbInformation,
"8051 Control I/O"
End Sub
```

ชุดรับค่าจาก FPGA โดยจะอยู่ใน MSComm1_OnComm()

```
Private Sub MSComm1_OnComm()
Select Case MSComm1.CommEvent
        Case comEvReceive
```

```
                buffer = MSComm1.Input
                Label3.Caption = "ReturnHeximal = "
                Label3.Refresh
                Label4.Caption = "ReturnDecimal = "
                Label4.Refresh
                amplitude = (Asc(buffer))
            TChart1.Series(0).Add amplitude, Time, vbRed
            If (cnt > 256) Then
                cnt = 0
            End If
            Text2.Text = Text2.Text + CStr(amplitude) + " "
            a(cnt) = amplitude
            cnt = cnt + 1
            Txt2.Text = Hex(Asc(buffer))
            Txt3.Text = (Asc(buffer))          ' Decimal
        End Select
        End Sub
```

อธิบายส่วนของ Visual Basic

ส่วนของ Visual Basic จะทำหน้าที่ รับข้อมูลจากพอร์ตคอมหนึ่งโดยผ่านการสื่อสารแบบอนุกรมด้วยมาตรฐาน RS-232 เพื่อแสดงผลออกที่จอคอมพิวเตอร์ โดยสามารถเลือกรูปแบบการสื่อสารได้ว่าจะมีความเร็วในการสื่อสารมากน้อยเท่าใดโดยมีให้เลือกดังนี้ 110, 150, 300, 600, 1200, 2400, 4800, 9600และ 19200 บิตต่อวินาที ผ่านพอร์ตคอมไหน โดยคอนโทรลที่สำคัญที่ทำให้ Visual Basic สามารถสื่อสารผ่านพอร์ตอนุกรมได้คือ คอนโทรล MSComm ซึ่งไม่ใช่คอนโทรลมาตรฐาน ดังนั้นถ้าเราต้องการใช้งาน MSComm เราจะต้องทำการเพิ่มคอนโทรลนี้เข้าไปใน ToolBox ซึ่งสามารถกระทำได้โดยคลิกขวาที่ Toolbox แล้วเลือกเมนู Component ชื่อ Microsoft Comm Control 6.0

บทที่ 6

การทดลองและผลการทดลอง

เครื่องแสดงผลระดับสัญญาณเสียง

6.1 ผลการทดลองในวงจรกรองความถี่

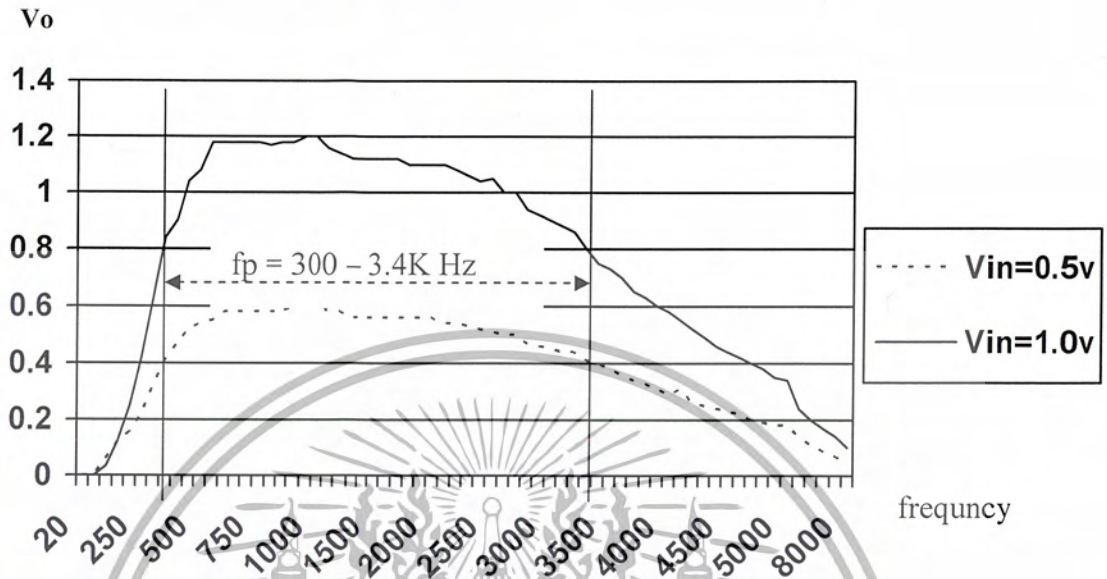
6.1.1 ผลการทดลองของวงจร Band pass filter

ในช่วง 300-3 kHZ Vin = 0.5Vpp และ1.0 Vpp

ความถี่ (HZ)	O/P Filter Narmal		ความถี่ (HZ)	O/P Filter narmal	
	0.5Vpp	1.0Vpp		0.5Vpp	1.0Vpp
20	0	0	2300	0.53	1.06
50	0	0	2400	0.52	1.04
100	0.09	0.035	2500	0.51	1.05
150	0.53	0.12	2600	0.5	1.0
200	0.13	0.25	2700	0.5	1.0
250	0.22	0.42	2800	0.47	0.94
300	0.32	0.63	2900	0.46	0.92
350	0.41	0.84	3000	0.45	0.9
400	0.47	0.904	3100	0.44	0.88
450	0.52	1.04	3200	0.44	0.86
500	0.54	1.08	3300	0.41	0.8
550	0.55	1.18	3400	0.4	0.75
600	0.58	1.18	3500	0.38	0.73
650	0.58	1.18	3600	0.36	0.7
700	0.58	1.18	3700	0.34	0.65
750	0.58	1.18	3800	0.33	0.63
800	0.58	1.17	3900	0.31	0.6
850	0.58	1.18	4000	0.29	0.58
900	0.6	1.18	4100	0.28	0.55

ความถี่ (HZ)	O/P Filter narmal		ความถี่ (HZ)	O/P Filter narmal	
	0.5Vpp	1.0Vpp		0.5Vpp	1.0Vpp
950	0.6	1.2	4200	0.26	0.52
1000	0.63	1.2	4300	0.25	0.49
1100	0.58	1.16	4400	0.24	0.46
1200	0.58	1.14	4500	0.23	0.44
1300	0.56	1.12	4600	0.22	0.42
1400	0.56	1.12	4700	0.2	0.4
1500	0.56	1.12	4800	0.19	0.38
1600	0.56	1.12	4900	0.18	0.35
1700	0.56	1.12	5000	0.18	0.34
1800	0.56	1.1	5500	0.14	0.24
1900	0.56	1.1	6000	0.11	0.2
2000	0.56	1.1	6500	0.09	0.17
2100	0.54	1.1	7000	0.07	0.14
2200	0.54	1.08	8000	0.05	0.1

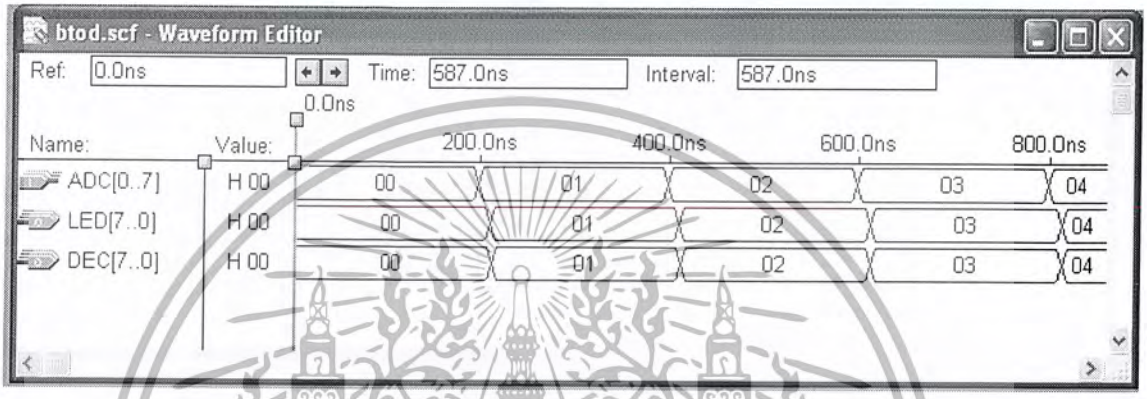
6.1.2 กราฟแสดงความสัมพันธ์ O/P Filter normal



รูปที่ 6.1 ความสัมพันธ์ O/P Filter normal

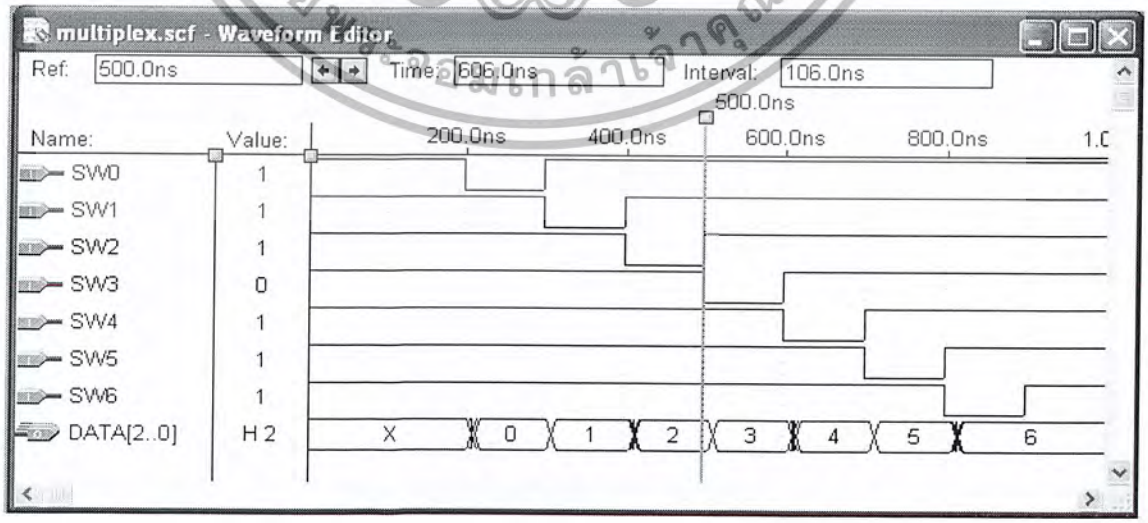
6.2 ผลการทดลองในส่วนของเอฟพีจีเอ

6.2.1) จากการออกแบบสร้างไฟล์ Waveform ขึ้นมาแล้วบันทึกในชื่อ BtoD.scf สำหรับจำลองการทำงานและให้โปรแกรม MAX+PLUS II บันทึกผลการ Simulation โดยกำหนดค่า End Time เท่ากับ 1.0uS ค่า Grid Size เท่ากับ 100nS สร้างสัญญาณอินพุทตามรูปที่ 6.2 หลังจากนั้นให้ทำการ Simulation เพื่อดูผลของเอาท์พุท



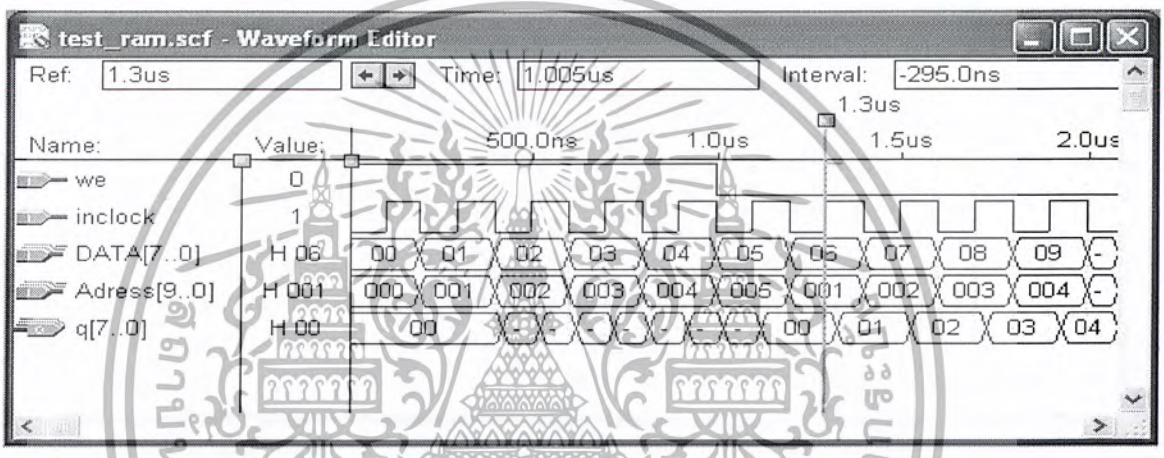
รูปที่ 6.2 ผลของการ Simulate Waveform ของ BtoD.scf

6.2.2) จากการออกแบบสร้างไฟล์ Waveform ขึ้นมาแล้วบันทึกในชื่อ Multiplex.scf สำหรับจำลองการทำงานและให้โปรแกรม MAX+PLUS II บันทึกผลการ Simulation โดยกำหนดค่า End Time เท่ากับ 1.0uS ค่า Grid Size เท่ากับ 100nS สร้างสัญญาณอินพุทตามรูปที่ 6.3 หลังจากนั้นให้ทำการ Simulation เพื่อดูผลของเอาท์พุท



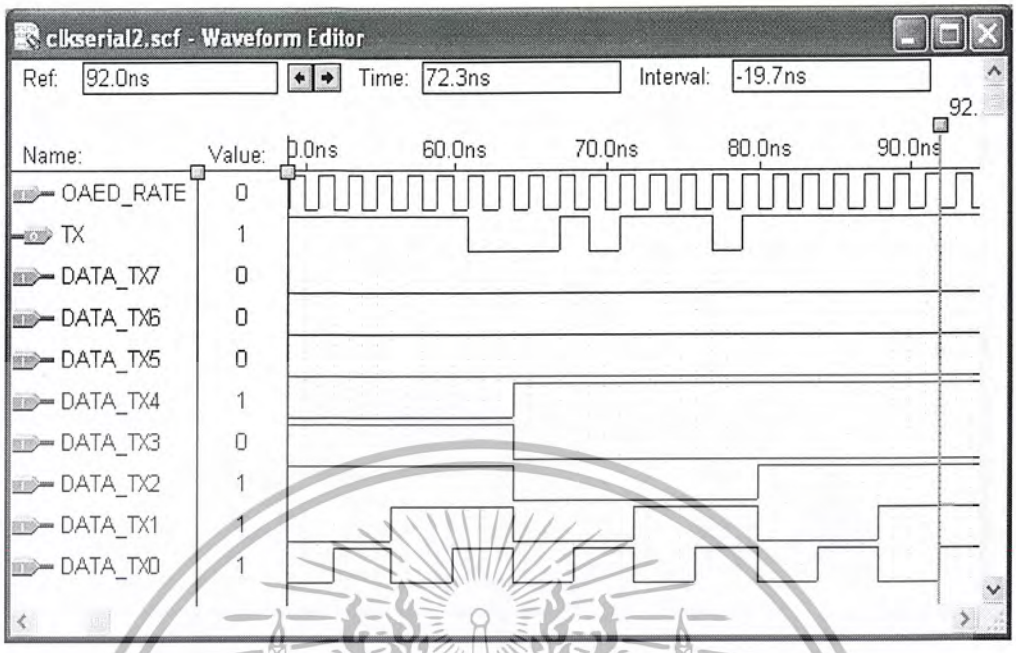
รูปที่ 6.3 ผลของการ Simulate Waveform ของ Multiplex.scf

6.2.3) จากการออกแบบสร้างไฟล์ Waveform ขึ้นมาแล้วบันทึกในชื่อ RAM.scf สำหรับจำลองการทำงานและให้โปรแกรม MAX+PLUS II บันทึกผลการ Simulation โดยกำหนดค่า End Time เท่ากับ 10.0uS ค่า Grid Size เท่ากับ 100nS สร้างสัญญาณอินพุทตามรูปที่ 6.4 หลังจากนั้นให้ทำการ Simulation เพื่อดูผลของเอาท์พุท



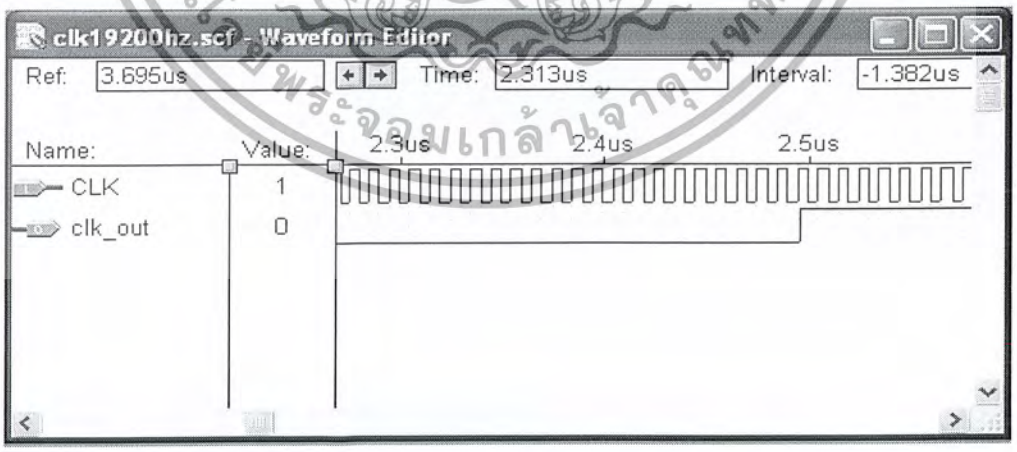
รูปที่ 6.4 ผลของการ Simulate Waveform ของ RAM.scf

6.2.4) จากการออกแบบสร้างไฟล์ Waveform ขึ้นมาแล้วบันทึกในชื่อ COMM_SERIAL.scf สำหรับจำลองการทำงานและให้โปรแกรม MAX+PLUS II บันทึกผลการ Simulation โดยกำหนดค่า End Time เท่ากับ 1.0uS ค่า Grid Size เท่ากับ 100nS สร้างสัญญาณอินพุทตามรูปที่ 6.5 หลังจากนั้นให้ทำการ Simulation เพื่อดูผลของเอาท์พุท



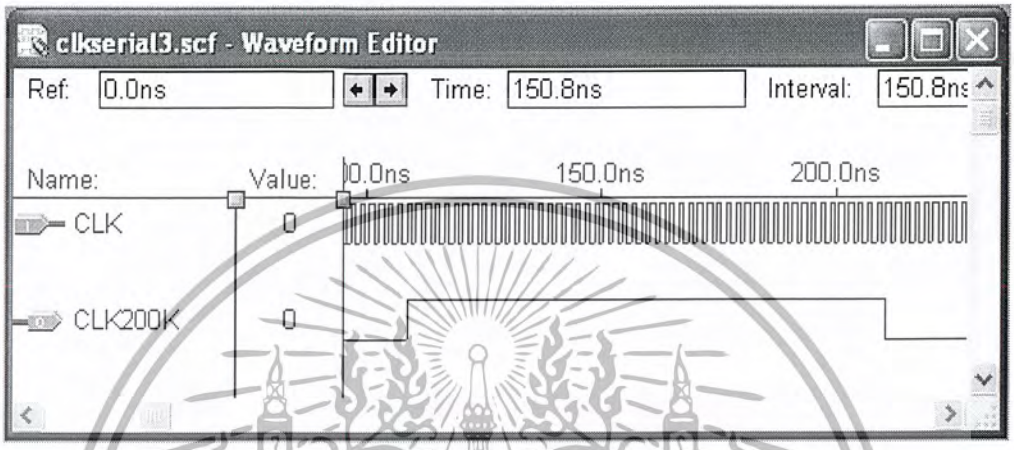
รูปที่ 6.5 ผลของการ Simulate COMM_SERIAL.scf

6.2.5) จากการออกแบบสร้างไฟล์ Waveform ขึ้นมาแล้วบันทึกในชื่อ Div500.scf สำหรับจำลองการทำงานและให้โปรแกรม MAX+PLUS II บันทึกผลการ Simulation โดยกำหนดค่า End Time เท่ากับ 10uS ค่า Grid Size เท่ากับ 5nS สร้างสัญญาณอินพุทตามรูปที่6.6 หลังจากนั้นให้ทำการ Simulation เพื่อดูผลของเอาท์พุท



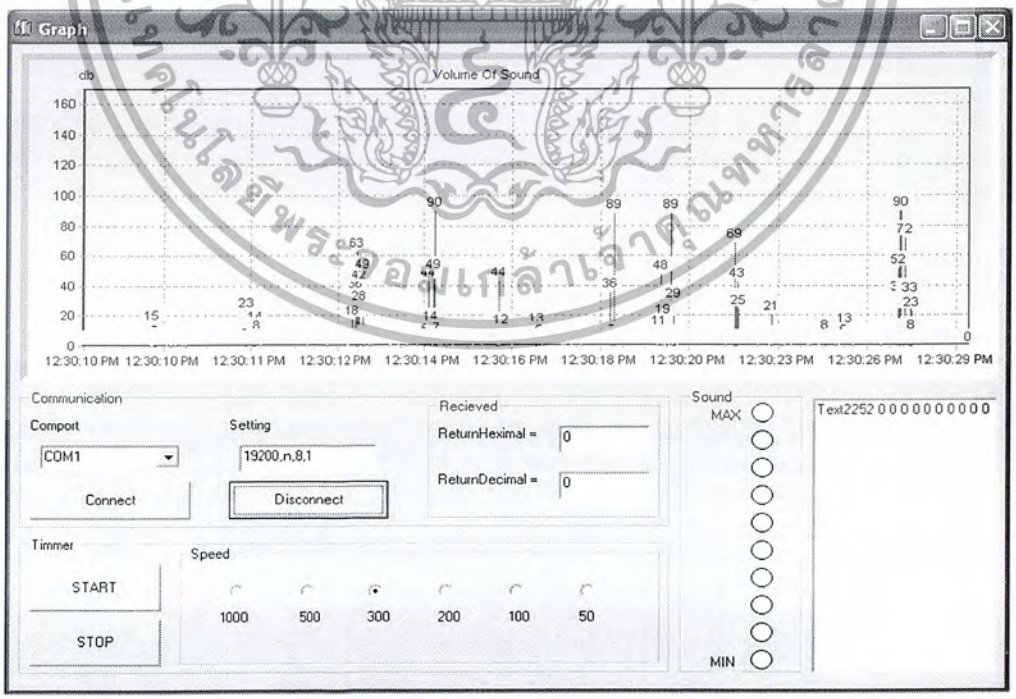
รูปที่ 6.6 ผลของการ Simulate Div500.scf

6.2.6) จากการออกแบบสร้างไฟล์ Waveform ขึ้นมาแล้วบันทึกในชื่อ Clock200k.scf สำหรับจำลองการทำงานและให้โปรแกรม MAX+PLUS II บันทึกผลการ Simulation โดยกำหนดค่า End Time เท่ากับ 1.0uS ค่า Grid Size เท่ากับ 100nS สร้างสัญญาณอินพุทตามรูปที่ 6.7 หลังจากนั้นให้ทำการ Simulation เพื่อดูผลของเอาท์พุท



รูปที่ 6.7 ผลของการ Simulate Clock200k.scf

6.3 ผลการทดลองในส่วนของ Visual Basic



รูปที่ 6.8 แสดงการทดลองป้อนสัญญาณเสียง

บทที่ 7
บทสรุป วิจารณ์ผลการทอลอง

7.1 สรุปผลการทดลอง

รายงานนี้ได้นำเสนอความรู้เกี่ยวกับ ทฤษฎี และการเขียน ภาษา VHDL , ความเป็นมาและเทคโนโลยีของ FPGA ว่าเป็นมาอย่างไร และการใช้งานต่างๆของ โปรแกรม MAXPLUS II รวมไปถึงหลักการทำงานและขั้นตอนการใช้งานโปรแกรมวิสชวลเบสิค

เมื่อเราเขียนโปรแกรมภาษา VHDL โดยไม่คำนึงถึงจำนวนเกจที่ใช้ใน FPGA ตัวนั้นๆแล้ว เมื่อใช้งานไปจนถึงระดับหนึ่งจะพบว่าการคอมไพล์โปรแกรมนั้นจะช้าลงเพราะจำนวนเกจที่ถูกใช้งานมีจำนวนมากขึ้นดังนั้นจึงจำเป็นต้องใช้เทคนิคการเขียนเพื่อใช้ทรัพยากรให้น้อยที่สุด

7.2 ปัญหาและอุปสรรค

1. วงจรบางส่วนที่มีความซับซ้อนมาก จะเป็นการยากในการออกแบบ และทดลอง
2. ในการออกแบบนั้นหาอุปกรณ์ที่ใช้ในการทำโครงงานได้ยาก จึงต้องใช้เวลานานในการทำโครงงาน
3. ผลที่ได้นั้นไม่เป็นไปตามทฤษฎี หรือที่ออกแบบไว้โดยตรง เพราะอาจเกิดจากตัวแปรหลายอย่าง เช่น สัญญาณรบกวน การเชื่อมต่ออุปกรณ์ เครื่องมือ หรือความผิดพลาดของผู้ทดลองเอง

7.3 บทวิจารณ์

ในการดำเนินงาน ซึ่งเป็นการใช้ภาษาอธิบายลักษณะการทำงาน ของแต่ละโมดูลที่กำหนดขึ้นซึ่ง ส่วนใหญ่จะใช้คอมพิวเตอร์ ช่วยในการออกแบบตั้งแต่ การเขียน ภาษาวีเอชดีแอล จนถึงการทดสอบและนำข้อมูลที่ได้ลงเก็บในเอฟพีจีเอ ซึ่งในการนำข้อมูลลงเอฟพีจีเอ จะต่อจากพอร์ตขนานของคอมพิวเตอร์ ซึ่งจากการทดลองพบว่า ช่วงที่ทำการนำข้อมูลลงเก็บในเอฟพีจีเอ แหล่งจ่ายจะถูกดึงกระแสมาก เพราะดูจากไฟที่แอลอีดี จะอ่อนลงขณะทำงานและสายที่ใช้ในการนำข้อมูลต้องยาวไม่มาก(ไม่ควรเกิน 1 เมตร) เพราะไม่สามารถทำการโหลดข้อมูลลงเอพีจีเอได้ และในการสังเคราะห์วงจร ถ้าวงจรที่ออกแบบมีความซับซ้อนเครื่องคอมพิวเตอร์ จะต้องใช้หน่วยความจำมาก ดังนั้นขณะทำการสังเคราะห์วงจรอยู่ไม่ควร เปิดโปรแกรมอื่น ใช้งานพร้อมกันเพราะจะทำให้เครื่องหยุดทำงานไปเฉยๆ รวมทั้งความเร็วของซีพียูก็มีผลต่อความเร็วในการคอมไพล์ด้วย

ทางด้านของภาษา VHDL และการใช้งาน FPGA ค่อนข้างมีลายละเอียดในการใช้งานทำให้ยุ่งยากบ้างในการศึกษาเบื้องต้น แต่หลังจากเริ่มเข้าใจระบบการออกแบบและโครงสร้างต่างของภาษาทำให้การออกแบบทำได้ง่ายขึ้น และหลังจากการสังเคราะห์โดยนำมาโปรแกรมลงบนชิพ FPGA แล้วบางครั้งอาจจะมีการทำงานที่ผิดพลาดบ้าง เราจึงต้องทำการแก้ไขโปรแกรมภาษา VHDL ใหม่อีก

ภาคผนวก

ตารางที่ 1 ความสัมพันธ์ระหว่างพอร์ตขยายช่องสัญญาณ Ext A และ Ext B กับ ตำแหน่งขา ของ EPF10K20TC144

Ext A PIN NUMBER	PIN EPF10K20	Ext B PIN NUMBER	PIN EPF10K20
1	144(I/O,NCS)	1	72(I/O)
2	143(I/O,OS)	2	70(I/O)
3	142(I/O,NWS)	3	69(I/O)
4	141(I/O,NRS)	4	68(I/O)
5	140(I/O)	5	67(I/O)
6	138(I/O)	6	65(I/O)
7	137(I/O)	7	64(I/O)
8	136(I/O)	8	63(I/O)
9	135(I/O)	9	62(I/O)
10	133(I/O)	10	60(I/O)
11	132(I/O)	11	59(I/O)
12	131(I/O)	12	56(DED.INPUT)
13	130(I/O)	13	55(GLOBALCLK)
14	128(I/O,DEV_OE)	14	54(DED.INPUT)
15	126(Ded.Input)	15	51(I/O)
16	125(Global CLK)	16	49(I/O)
17	124(Ded.input)	17	48(I/O)
18	122(I/O.DEVCLR)	18	47(I/O)
19	121(I/O)	19	46(I/O)
20	120(I/O)	20	44(I/O)
21	119(I/O)	21	43(I/O)
22	118(I/O)	22	42(I/O)

Ext A PIN NUMBER	PIN EPF10K20	Ext A PIN NUMBER	PIN EPF10K20
23	117(I/O)	23	41(I/O)
24	116(I/O,DATA7)	24	39(I/O)
25	114(I/O,DATA6)	25	38(I/O)
26	113(I/O,DATA5)	26	37(I/O)
27	112(I/O,DATA4)	27	36(I/O)
28	111(I/O,DATA3)	28	33(I/O)
29	110I/O,DATA2)	29	32(I/O)
30	109(I/O,DATA1)	30	31(I/O)
31	102(I/O)	31	30(I/O)
32	101(I/O)	32	29(I/O)
33	100(I/O)	33	28(I/O)
34	99(I/O)	34	27(I/O)
35	98(I/O)	35	26(I/O)
36	97(I/O)	36	23(I/O)
37	96(I/O)	37	22(I/O)
38	95(I/O)	38	21(I/O)
39	92(I/O)	39	20(I/O)
40	91(I/O)	40	19(I/O)
41	90(I/O)	41	18(I/O)
42	89(I/O)	42	17(I/O)
43	88(I/O)	43	14(I/O,INTDONE)
44	87(I/O)	44	13(I/O)
45	86(I/O)	45	12(I/O)
46	83(I/O)	46	11(I/O,RDBUSY)
47	82(I/O)	47	10(I/O)
48	81(I/O)	48	9(I/O)
49	80(I/O)	49	8(I/O)
50	79(I/O)	50	7(I/O,CLKUSR)

51	78(I/O)	51	GND
52	73(I/O)	52	VCC

ตารางที่ 2 ความสัมพันธ์ระหว่างวงจรสื่อสารข้อมูลแบบอนุกรมกับ
ตำแหน่งขาของ EPF10K20

Seerial Port Pin Nam	EPF10K20	Seerial Port Pin Nam	EPF10K20
RX	144(I/O.NCS)	TX	143(I/O,CS)

ตารางที่ 3 ความสัมพันธ์ระหว่างคอนเน็กเตอร์สำหรับเชื่อมต่อกับจอ VGA กับ
ตำแหน่งขาของ EPF10K20TC144

VGA Port Pin Name	EPF10K20 Pin	VGA Port Pin Name	EPF10K20 Pin
R	235(I/O)	V-SYNC	140(I/O)
G	136(I/O)	H-SYNC	138((I/O)
B	137(I/O)	-	-

ตารางที่ 4 ความสัมพันธ์ระหว่างคอนเน็กเตอร์แบบ PS/2 กับ
ตำแหน่งขาของ EPF10K20TC144

PS/2 PORT PIN	EPF10K20 PIN	PS/2 PORT PIN	EPF10K20 PIN
DATA	142(I/O,NWS)	CLK_PS2	141(I/O,NRS)

ตารางที่ 5 ความสัมพันธ์ระหว่างคอนเน็กเตอร์แบบ Centronics Port กับ
ตำแหน่งขาของ EPF10K20TC144

Centronics Port	EPF10K20 Pin	Centronics Port	EPF10K20 Pin
DB0	7(I/O,CLKUSR)	/LE/CR	18(I/O)

DB1	8(I/O)	/INITIALIZE	19(I/O)
DB2	9(I/O)	/SLIN	20(I/O)
DB3	10(I/O)	/ERROR	21(I/O)
DB4	11(I/O,RDBUSY)	SLCT	22(I/O)
DB5	12(I/O)	PE	23(I/O)
DB6	13(I/O)	/ACK	27(I/O)
DB7	14(I/O),INTDONE)	BUSY	27(I/O)
/STROBE	17(I/O)	-	-

ตารางที่ 6 ความสัมพันธ์ระหว่างวงจรแปลงสัญญาณดิจิตอลเป็นอนาล็อกกับ ตำแหน่งขาของ **EPF10K20TC144**

Circuit Pin Name	EPF10K20 Pin	Circuit Pin Name	EPF10K20 Pin
DAC0	91(I/O)	DAC4	87(I/O)
DAC1	90(I/O)	DAC5	86(I/O)
DAC2	89(I/O)	DAC6	83(I/O)
DAC3	88(I/O)	DAC7	82(I/O)

ตารางที่ 7 ความสัมพันธ์ระหว่างวงจรหน่วยความจำกับ ตำแหน่งขาของ **EPF10K20TC144**

Circuit Pin Name	EPF10K20 Pin	Circuit Pin Name	EPF10K20 Pin
/OE	41(I/O)	AD11	64(I/O)
/WE	42(I/O)	AD10	65(I/O)
DM7	43(I/O)	AD9	67(I/O)
DM6	44(I/O)	AD8	68(I/O)
DM4	47(I/O)	AD6	70(I/O)
DM3	48(I/O)	AD5	72(I/O)
DM2	49(I/O)	AD4	73(I/O)
DM1	51(I/O)	AD3	78(I/O)
DM0	59(I/O)	AD2	79(I/O)

AD14	60(I/O)	AD1	80(I/O)
AD13	62(I/O)	AD0	81(I/O)
AD12	63(I/O)	-	-

ตารางที่ 8 ความสัมพันธ์ระหว่างสวิตซ์กดติด- ปล่อยดับ กับ
ตำแหน่งขาของ EPF10K20TC144

Circuit Pin Name	EPF10K20 Pin	Circuit Pin Name	EPF10K20
SW1	56(DED.INPUT)	SW2	54(DED.INPUT)

ตารางที่ 9 ความสัมพันธ์ระหว่างดิฟสวิตซ์ 8 บิต กับ
ตำแหน่งขาของ EPF10K20TC144

Circuit Pin Name	EPF10K20 Pin	Circuit Pin Name	EPF10K20 Pin
DSW0	28(I/O)	DSW4	32(I/O)
DSW1	29(I/O)	DSW5	33(I/O)
DSW2	30(I/O)	DSW6	36(I/O)
DSW3	31(I/O)	DSW7	37(I/O)

ตารางที่ 10 ความสัมพันธ์ระหว่างไดโอดเปล่งแสงกับ
ตำแหน่งขาของ EPF10K20TC144

Circuit Pin Name	EPF10K20 Pin	Circuit Pin Name	EPF10K20 Pin
LED0	120(I/O)	LED4	130(I/O)
LED1	121(I/O)	LED5	131(I/O)
LED2	122(I/O,DEVCLR)	LED6	132(I/O)
LED3	128(I/O,DEVOE)	LED7	133(I/O)

ตารางที่ 11 ความสัมพันธ์ระหว่างวงจรแสดงผล 7 – Segment 4 หลักกับ ตำแหน่งขาของ EPF10K20TC144

Circuit Pin Name	EPF10K20 Pin	Circuit Pin Name	EPF10K20 Pin
SEG.A	109(I/O,DATA1)	SEG.G	116(I/O,DATA7)
SEG.B	110(I/O,DATA2)	SEG.DOT	117(I/O)
SEG.C	111(I/O,DATA3)	CM1	118(I/O)
SEG.D	112(I/O,DATA4)	CM2	119(I/O)
SEG.E	113(I/O,DATA5)	CM3	39(I/O)
SEG.F	114(I/O,DATA60	CM4	38(I/O)
OSC	55(GLOBAL CLK)		

ตารางที่ 12 ความสัมพันธ์ระหว่างวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอลกับ ตำแหน่งขาของ EPF10K20tc144

CIRCUIT PIN	EPF10K20 PIN	CIRCUIT PIN	EPF10K20 PIN
Clock200k	102(I/O)	ADC4	97(I/O)
ADC0	101(I/O)	ADC5	96(I/O)
ADC1	100(I/O)	ADC6	95(I/O)
ADC2	99(I/O)	ADC7	92(I/O)
ADC3	98(I/O)	-	-

บรรณานุกรม

รศ.ดร.มานัส สังวรศิลป์ ดร.กิตติพล ชิตสกุล อ.เกษม เสพศิริสุข และนายบุญอนันต์ เกียงเอีย, เปิดโลก FPGA WIZARD PLDA01, บริษัทแอสทรอนลอจิกรีเสริช์ทอนดิวิล็อปเมนต์

นอ.ชาติชาย ดิษฐกุล , เอกสารประกอบการเรียน ภาษา VHDL, ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

อภิชาติ ภู่พลับ, เริ่มต้นเขียนโปรแกรมติดต่อและควบคุมฮาร์ดแวร์ด้วย Visual Basic. นนทบุรี, อินโฟเพลส , 2546