การออกแบบและสร้างตัวกำจัดสัญญาณเสียงสะท้อนโดยใช้ FPGA

DESIGN AND IMPLEMENTATION OF AN FPGA-BASED FOR VOICE ECHO CANCELLER

วัชระ ภัคมาศร์
WATCHARA PAKKAMART

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้า
บัณฑิตวิทยาลัย
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2545

ISBN 974-648-748-5

การออกแบบและสร้างตัวกำจัดสัญญาณเสียงสะท้อนโดยใช้ FPGA

DESIGN AND IMPLEMENTATION OF AN FPGA – BASED FOR VOICE ECHO CANCELLER

วัชระ ภัคมาตร์

WATCHARA PAKKAMART

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้า
บัณฑิตวิทยาลัย
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2545

ISBN 974-648-748-5

DESIGN AND IMPLEMENTATION OF AN FPGA – BASED FOR VOICE ECHO CANCELLER

WATCHARA PAKKAMART

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING

SCHOOL OF GRADUATED STUDIES

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2002

ISBN 974-648-748-5

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองวิทยานิพนธ์

หัวข้อวิทยานิพนธ์ การออกแบบและสร้างตัวกำจัดสัญญาณ เสียงสะท้อนโดยใช้ FPGA

DESIGN AND IMPLEMENTATION OF AN FPGA-BASED FOR VOICE ECHO CANCELLER

ชื่อนักศึกษา นายวัชระ ภัคมาตร์

รหัสประจำตัว 38061233

ปริญญา วิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชา วิศวกรรมไฟฟ้า

อาจารย์ผู้ควบคุมวิทยานิพนธ์ รศ.ดร.กอบชัย เดชหาญ

คณะกรรมการสอบวิทยานิพนธ์		ลายมือชื่อ
รศ.ณรงค์	เหมกรณ์	
รศ.ดร.ยุทธพงษ์	รังสรรค์เสรี	
รศ.ชวลิต	เบญจางคประเสริฐ	
รศ.ดร.ฟูศักดิ์	ชีวสุวิทย์	
รศ.ดร.กอบชัย	เดชหาญ	

วัน/เดือนปี ที่สอบ 23 เมษายน 2545 **เวลา** 14.00-16.00 น.

สถานที่สอบ ณ อาคาร 12 ชั้น ชั้น 4 (ห้อง E12-404)

บัณฑิตวิทยาลัยรับรองแล้ว

(รศ.ดร.บุญวัฒน์ อัตชู)

คณบดีบัณฑิตวิทยาลัย

วันที่ 15 **เดือน** พฤษภาคม **พ.ศ.** 2545

หัวข้อวิทยานิพนธ์	การออกแบบและสร้างตัวกำจัดสัญญาณเสียงสะท้อน โดยใช้ FPGA
นักศึกษา	นายวัชระ ภัคมาตร์
รหัสประจำตัว	38061233
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมไฟฟ้า
พ.ศ.	2545
อาจารย์ผู้ควบคุมวิทยานิพนธ์	รศ.ดร. กอบชัย เดชหาญ

บทคัดย่อ

วิทยานิพนธ์ฉบับนี้นำเสนอการออกแบบและสร้างตัวกำจัดสัญญาณเสียงสะท้อนโดยใช้ FPGA ในการออกแบบได้ใช้อัลกอริทึมแบบ LMS adaptive transversal filter ขนาด 128 Tap ทำงานเป็นตัวกำจัดสัญญาณเสียงสะท้อน ระบบที่สร้างขึ้นจะทำงานอยู่ในย่านความถี่ 300- 3400 Hz มีค่าความถี่ของอัตราการสุ่มเป็น 40 kHz และขนาดของข้อมูลทางอินพุทเป็น 12 บิต การออกแบบโครงสร้างการทำงานของตัวกรองปรับตัวได้ (Adaptive Filter) ได้ออกแบบให้มีลักษณะเป็นหน่วยคำนวณทางคณิตศาสตร์หลายหน่วยทำงานขนานกัน โดยในแต่ละหน่วยของการคำนวณจะทำงานเพียงฟังก์ชันเดียว จึงทำให้เวลาในการประมวลผลข้อมูลสามารถที่จะกำหนดและควบคุมได้ทั้งระบบโดยอิสระจากค่าของอัตราความถี่สุ่มและได้ทำการทดสอบระบบด้วยการจำลองการทำงานในห้องปฏิบัติการ ผลการทดสอบพบว่าระบบสามารถกำจัดสัญญาณเสียงสะท้อนได้ดี

Thesis Title	Design and Implementation of an FPGA – Based for Voice Echo Canceller
Student	Watchara Pakkamart
Student ID.	38061233
Degree	Master Degree
Programme	Electrical Engineering
Year	2002
Thesis Advisor	Assoc.Prof. Dr. Kobchai Dejhan

ABSTRACT

This thesis presents a design and implementation of voice the echo canceller (VEC) by using FPGA as core processor with 128 tap based on LMS adaptive transversal filter algorithm, 12-data bit input design with 300-3400 Hz frequency range, 40 kHz sampling rate. The design structure of adaptive filter is pararell arithmetic units which work for only one function, so whole processing time can be assigned and controlled and it is independent of the sampling frequency. All results have been tested and it can completely reject the echo.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงได้อย่างดี ด้วยคำแนะนำและคำปรึกษาพร้อมทั้งกำลังใจจาก รศ.ดร.กอบชัย เดชหาญ ซึ่งเป็นอาจารย์ผู้ควบคุมวิทยานิพนธ์ ผู้วิจัยรู้สึกทราบซึ้งในความกรุณาของท่านและขอกราบขอบพระคุณเป็นอย่างสูง

ขอขอบพระคุณ นอ. ชาติชาย ดิษฐกุล ป้อมพระจุลจอมเกล้า ที่ได้มอบพื้นฐานความรู้ด้านการออกแบบขั้นสูงและภาษา VHDL ให้แก่ผู้วิจัยซึ่งเป็นส่วนสำคัญยิ่งในงานวิจัยนี้

ขอขอบพระคุณ ศ.ดร. สุระ เลขะกุล St. Clound State University , Minnesota USA ที่ได้แนะนำเทคนิคการออกแบบโครงสร้างวงจรดิจิตอลด้วยภาษา VHDL ให้แก่ผู้วิจัย

ขอขอบคุณ คุณชำนาญ ปัญญาใส หัวหน้าฝ่ายออกแบบวงจรรวม ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ (NECTEC) ที่ได้ให้ความอนุเคราะห์ด้านซอฟแวร์สำหรับใช้การวิจัย ซึ่งเป็นส่วนสำคัญที่ทำให้การเริ่มต้นงานวิจัยของผู้วิจัยเป็นไปได้อย่างดี

ขอขอบคุณ อ.พิสิษฐ์ อิทธิยาวุธ และบุคลากรภาควิชาวิศวกรรมไฟฟ้า สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ ที่ให้กำลังใจแก่ผู้วิจัยมาโดยตลอด

ขอขอบคุณ อ.นุชรินทร์ ทิพยวรรณากร ภาควิชาสถิติประยุกต์ คณะวิทยาศาสตร์ประยุกต์ สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ ที่ช่วยให้คำแนะนำด้านการประมวลผลสัญญาณสุ่มและเป็นกำลังใจที่สำคัญแก่ผู้วิจัยเป็นอย่างดีมาโดยตลอด

สุดท้ายขอขอบคุณน้อง ๆ ทุกคนที่ช่วยต่อวงจรทดสอบ จนงานวิจัยชิ้นนี้สำเร็จสมบูรณ์ได้

วัชระ ภัคมาตร์

สารบัญ

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

โดยทั่วไปแล้วการประยุกต์ใช้งานของ DSP (Digital signal processing) ในงานต่าง ๆ จะถูกประยุกต์ใช้บนชิพ DSP controller ซึ่งจำเป็นจะต้องมีการคำนึงถึงขีดความสามารถ ข้อจำกัด หรือข้อกำหนดของชิพแต่ละเบอร์ที่ถูกกำหนดโดยบริษัทผู้ผลิต นอกจากนั้นสิ่งที่ทำให้เกิดเป็นปัญหาต่อการนำชิพไปประยุกต์ใช้งานที่พบค่อนข้างมาก คือ ส่วนของซอฟแวร์สนับสนุน(Utility) ดังเช่น Compiler debugger และ Download tool ซึ่งโดยส่วนใหญ่แล้วผู้ผลิตมักจะออกแบบให้เหมาะกับระบบฮาร์ดแวร์ของตนเองเท่านั้น ดังนั้นผู้ใช้งานจึงมักจะถูกจำกัดขอบเขตของการประยุกต์ใช้ค่อนข้างมากทำให้อาจเป็นอุปสรรคในการศึกษาวิจัยและพัฒนาสู่ระบบอุตสาหกรรมได้ โดยเฉพาะการประยุกต์ใช้งานกับวงจรกรองที่ประสิทธิภาพของการทำงานจะขึ้นอยู่กับขนาดความแม่นยำของตัวสัมประสิทธิ์ (Coefficient precision) ดังนั้นโครงสร้างและสถาปัตยกรรมของชิพ DSP จึงเป็นตัวจำกัดสมรรถนะ ของวงจรกรองด้วย

ในวงจรกำจัดเสียงสะท้อน (Echo canceller) ซึ่งมีระบบการทำงานหลักขึ้นอยู่กับวงจรกรอง จึงยังมีข้อจำกัดอยู่กับเทคโนโลยีที่นำมาใช้ของชิพ DSP แต่ละเบอร์และการเพิ่มค่าความแม่นยำของตัวสัมประสิทธิ์จะช่วยให้ค่าความคลาดเคลื่อนของเสียงสะท้อน (Echo residual) มีค่าลดน้อยลง [1] ได้ แต่ในชิพ DSP การเพิ่มค่าความแม่นยำของสัมประสิทธิ์จำเป็นจะต้องมีการใช้เทคนิคเชิงเลขแบบอิงดรรชนี (Floating point) ซึ่งก็จะส่งผลกระทบต่อค่าพารามิเตอร์อื่น ๆ ในระบบด้วย เช่น จำนวนของชุดคำสั่ง (Instruction code) ที่ใช้ในการทำงาน ค่าความถี่ของสัญญาณนาฬิกา และจำนวนบิตความกว้างของข้อมูล เป็นต้น

ในปัจจุบันได้มีการคิดค้นเทคนิคการออกแบบขั้นสูง (High level design) ที่เปิดโอกาสให้ผู้ใช้งานสามารถที่จะออกแบบระบบงานทางด้านดิจิตอลลงบนอุปกรณ์แบบ PLD (Programmable logic Device) หรือ FPGA (Field programmable gate array) ได้ในระดับของการออกแบบที่อัลกอริทึม จึงทำให้การใช้ FPGA มีความยืดหยุ่นสูงมากในการออกแบบสร้างหน่วยประมวลผลสัญญาณทางดิจิตอลในแบบที่เป็นข้อมูลจริง (Real data bits)ได้ จึงน่าจะนำมาใช้งานในการสร้างวงจรกำจัดเสียงสะท้อนที่มีสมรรถนะตรงตามความต้องการของผู้ใช้งานได้

1.2 ความหมายและวัตถุประสงค์ของการศึกษาวิจัย

1. เพื่อศึกษาแนวทางการใช้เทคนิคของการออกแบบขั้นสูงสำหรับการพัฒนาและสร้างอุปกรณ์ DSP ที่ใช้งานในทางวิศวกรรมสื่อสาร

2. เพื่อสร้างต้นแบบตัวกำจัดสัญญาณเสียงสะท้อนโดยใช้อุปกรณ์ประเภท PLD หรือ FPGA

1.3 สมมุติฐานของการศึกษา

ปัจจุบันได้มีการนำอุปกรณ์ประเภท PLD และ FPGA มาใช้งานทางด้านการประมวลผลสัญญาณดิจิตอลจนเป็นผลสำเร็จในระดับของการออกแบบเป็นวงจรกรองได้เป็นอย่างดี ทั้งนี้เนื่องจากอุปกรณ์ดังกล่าวมีลักษณะโครงสร้างที่ยืดหยุ่นสำหรับการประยุกต์ใช้งาน จึงมีความเป็นไปได้ที่จะนำอุปกรณ์ PLD และ FPGA มาประยุกต์ใช้เพื่อสร้างเป็นหน่วยคำนวณคณิตศาสตร์ที่ซับซ้อนของวงจรกรองปรับตัวได้ (Adaptive filter) สำหรับเครื่องกำจัดสัญญาณเสียงสะท้อนในระบบโทรศัพท์ เพื่อให้ได้เครื่องต้นแบบที่มีโครงสร้างการคำนวณตรงตามความต้องการของผู้ใช้หรือผู้ออกแบบมากกว่าโครงสร้างที่เป็นอุปกรณ์แบบคอนโทรลเลอร์ทั่วไป

1.4 ทฤษฎีและแนวคิดที่ใช้ในการวิจัย

แนวคิดที่ใช้ในการวิจัยแบ่งเป็นสามส่วนคือ

1) ทฤษฎีของวงจรกรองปรับตัวได้ (Adaptive filter)

2) ทฤษฎีการออกแบบโดยใช้เทคนิคการออกแบบขั้นสูง VHDL และ DFL

3) ทฤษฎีการทำ Logical implementation บน FPGA และ PLD

1.5 ขอบเขตของงานวิจัย

1) ทำการออกแบบและสร้างต้นแบบของตัวกำจัดเสียงสะท้อนด้วยอัลกอริทึมของวงจรกรองปรับตัวได้โดยใช้เทคนิค LMS (Least mean square) ร่วมกับ FIR (FIR Filter) ขนาดไม่ต่ำกว่า 128 Tap ในย่านความถี่ใช้งานไม่เกิน 3400 Hz

2) สถาปัตยกรรม (Architecture) ของการประมวลผลสำหรับตัวกำจัดเสียงสะท้อนมีขนาดของข้อมูลไม่เกิน 16 บิต มีแรงดันทางด้านอินพุทและเอาท์พุทในส่วนที่เป็นสัญญาณแบบอนาลอกมีขนาดไม่เกิน 2 Vpp.

3) ในงานวิจัยจะทำการครอบคลุมในส่วนของการประมวลผลแบบดิจิตอลเท่านั้น

4) การทดสอบการทำงานของต้นแบบตัวกำจัดเสียงสะท้อนที่สร้างได้ โดยจำลองระบบการใช้งานจริงในห้องปฏิบัติการด้วยคอมพิวเตอร์ (Simulation)

1.6 ขั้นตอนของการวิจัย

1) ศึกษาอัลกอริทึมของตัวกำจัดเสียงสะท้อนที่มีส่วนของการประมาณค่าสัญญาณเสียงสะท้อน (Echo Prediction) ด้วยเทคนิค LMS โดยใช้วงจรกรองปรับตัวได้

2) จำลองระบบการทำงานของตัวกำจัดเสียงสะท้อนบนโปรแกรม MATLAB เพื่อตรวจสอบความถูกต้องสมบูรณ์ของการทำงานและลำดับขั้นตอนของอัลกอริทึม

3) นำอัลกอริทึมส่วนของการประมาณค่าสัญญาณเสียงสะท้อนที่ได้จากข้อ 1) เขียนเป็นโปรแกรมด้วยภาษาเชิงพฤติกรรม (Behavioral language) เช่น ภาษา VHDL และ DFL

4) สร้างส่วนของหน่วยควบคุม (Control unit) อินเตอร์เฟสพอร์ท (Interface port) และค่าประมาณของสัญญาณเสียงสะท้อนสำหรับตัวกำจัดเสียงสะท้อนที่ทำงานบนชิพ FPGA ของ Xilinx หรือบนชิพของ Altera

5) สร้างระบบการสร้างข้อมูล (Data acquisition) และประกอบวงจรให้ทำงานร่วมกับชิพที่ได้จากข้อ 4)

6) ทดสอบการทำงานของระบบและปรับปรุงแก้ไขระบบ

7) สรุปผลการวิจัย

1.7 วรรณกรรมและงานวิจัยที่เกี่ยวข้อง

งานวิจัยของ Zhahong Zhang และ Gunter Schmer [1] ได้รายงานการศึกษาวิเคราะห์ถึงผลกระทบของขนาดที่จำกัดในเรื่องของความแม่นยำของค่าสัมประสิทธิ์ในวงจรกรอง (Finite filter coefficient precision) ที่มีต่อสมรรถนะการทำงานของอัลกอริทึมแบบ LMS ในการกำจัดสัญญาณสะท้อน โดยงานวิจัยดังกล่าวได้สร้างอัลกอริทึมแบบ LMS จากนั้นทำการปรับขนาดจำนวนบิตของสัมประสิทธิ์ของวงจรกรองและขนาดของค่าผิดพลาดจากการควอนไทซ์ (Quantization error) ของสัญญาณข้อมูล ซึ่งก็ได้พบว่าผลของค่าผิดพลาดจากการควอนไทซ์ไม่กระทบต่ออัตราการลู่เข้าของอัลกอริทึมแบบ LMS แต่ผลของค่าผิดพลาดจากการควอนไทซ์ที่มีอยู่ในสัมประสิทธิ์ของวงจรกรองมีผลกระทบต่อระดับของ Final convergence level ของวงจรกรอง ทั้งนี้ขึ้นอยู่กับจำนวนบิตที่ใช้ในการคำนวณของสัมประสิทธิ์ของวงจรกรอง โดยทั่วไปการคำนวณจะถูกกำหนดด้วยจำนวนบิตที่คงที่ แต่ในทางปฏิบัติการคำนวณบางอย่างทำให้เกิดจำนวนบิตที่เพิ่มขึ้น เช่น การคูณ จึงทำให้

ข้อมูลบางส่วนถูกตัดปลายทิ้งไปในระดับที่มีผลกระทบต่อความถูกต้องของผลลัพธ์ของการคำนวณ ดังนั้นผู้วิจัยจึงเสนอทางเลือกที่สามารถแก้ไขปัญหาดังกล่าวได้ โดยการใช้อุปกรณ์ PLD และ FPGA เพื่อสร้างหน่วยประมวลผลทางคณิตศาสตร์ เพื่อให้ผู้ออกแบบสามารถกำหนดระดับความแม่นยำของข้อมูลได้ตามความเหมาะสม

บทที่ 2
การกำจัดสัญญาณเสียงสะท้อน

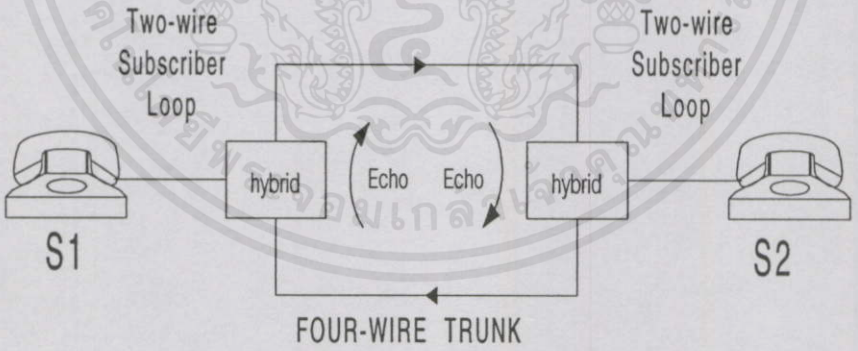
2.1 บทนำ

ในการติดต่อสื่อสารด้วยโทรศัพท์ทั่วไปจะมีการเกิดสัญญาณสะท้อนรบกวนในระบบ ทำให้เกิดความไม่สะดวกต่อทั้งผู้รับและผู้ส่งข้อมูลข่าวสารและอาจจะมีบางครั้งที่รุนแรงจนกระทั่งการติดต่อสื่อสารไม่สามารถดำเนินไปได้อย่างต่อเนื่อง ดังนั้นเพื่อให้การติดต่อสื่อสารด้วยโทรศัพท์มีประสิทธิภาพสูงขึ้นจึงมีความจำเป็นที่จะต้องกำจัดสัญญาณเสียงสะท้อนภายในระบบซึ่งในบทนี้จะได้ศึกษาถึงสาเหตุและแนวทางในการกำจัดสัญญาณเสียงสะท้อนและวิธีการที่จะนำมาใช้กำจัดสัญญาณเสียงสะท้อนตามลำดับ

2.2 การกำจัดสัญญาณเสียงสะท้อน

2.2.1 สัญญาณเสียงสะท้อนในระบบโทรศัพท์

ต้นกำเนิดของเสียงสะท้อนสามารถที่จะพิจารณาได้จากการเชื่อมต่อกันระหว่างตัว subscriber S_1 และ S_2 ที่แสดงไว้ดังรูปที่ 2.1



รูปที่ 2.1 สาเหตุการเกิดสัญญาณสะท้อนในระบบโทรศัพท์

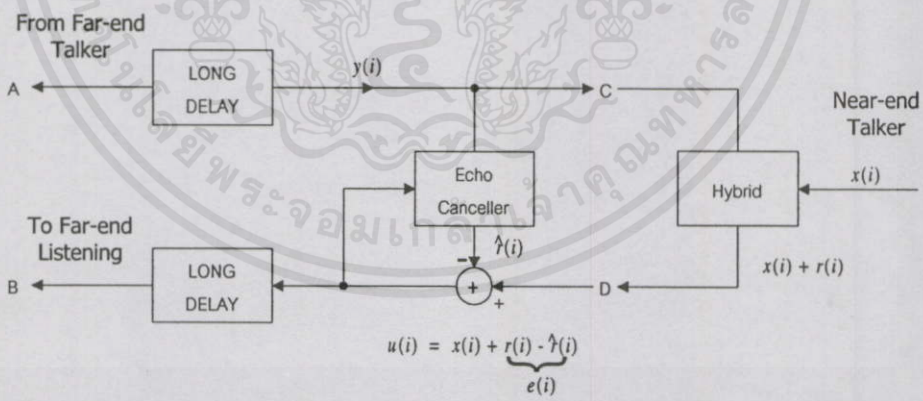
จากรูปที่ 2.1 การเชื่อมต่อจะประกอบไปด้วยส่วนของโครงข่ายแบบ 2 สาย (2-wire network) ที่อยู่ด้านปลายทั้งสองและมีการเชื่อมต่อแบบ 4 สาย (4-wire network) อยู่ตรงกลาง ส่วนอุปกรณ์ไฮบริด (Hybrid) ที่อยู่ปลายของระบบคอยทำการปรับเปลี่ยนจากการสื่อสารแบบ 2 สายไปเป็นการสื่อสารแบบ 4 สาย ในส่วนของการสื่อสารแบบ 2 สาย ก็จะประกอบไปด้วย Subscribers Loops และก็อาจจะมีบางส่วนที่จะเป็นโครงข่ายท้องถิ่น (Local network) ยิ่งกว่านั้นในส่วนนี้จะมีทิศทางของการส่ง

ผ่าน (Transmission) แบบสองทางโดยจะถูกทำการส่งผ่านด้วยสายคู่เดียวกัน สัญญาณจากผู้พูด S_1 และ S_2 จะถูกป้อนใส่เข้าไปในส่วนนี้ แต่วงจรส่วนที่เป็นการสื่อสารแบบ 4 สายนั้นการสื่อสารในแต่ละทางจะถูกแยกออกจากกัน สัญญาณเสียงจากผู้พูด S_1 ก็จะวิ่งไปตามเส้นทางด้านบนของรูปและมีทิศทางตามลูกศร ในขณะที่เสียงจากจุดกำเนิด S_2 ก็จะไปตามเส้นทางด้านล่างของรูป การแบ่งแยกสัญญาณทั้งสองออกจากกันนั้นเป็นสิ่งที่จำเป็นเนื่องจากต้องมีการนำไปรวมกับสัญญาณพาหะ (Carrier) ระบบขยายสัญญาณ (Amplifier) และระบบการสลับสายสัญญาณดิจิตอล (Digital switch)

อุปกรณ์ไฮบริดจะที่ทำหน้าที่เปลี่ยนการส่งผ่านจากระบบการสื่อสารแบบ 2 สายให้เป็นการสื่อสารแบบ 4 สาย หน้าที่ของไฮบริดที่อยู่ทางด้านขวามือก็จะทำการควบคุมให้พลังงานของสัญญาณจาก S_1 ไปยังส่วนที่เป็นการสื่อสารแบบ 2 สายของ S_2 โดยที่จะไม่ยินยอมให้มันย้อนกลับมายัง S_1 อีกด้วยเส้นทางของวงจรที่อยู่ด้านล่าง แต่เนื่องจากเกิดการไม่แมตในอุปกรณ์ไฮบริด จึงทำให้พลังงานบางส่วนเกิดการย้อนกลับไปยัง S_1 ทำให้ได้ยินเป็นเสียงที่มีการหน่วงเวลาไปจากปกติ ซึ่งการเกิดเหตุการณ์เช่นนี้จะเรียกว่า " เสียงสะท้อนถึงผู้พูด " (Talker Echo)

2.2.2 หลักการทำงานของตัวกำจัดสัญญาณเสียงสะท้อน

หลักการทำงานของตัวกำจัดสัญญาณสะท้อนที่ใช้สำหรับการสื่อสารในระบบทิศทางเดียวได้แสดงไว้ดังในรูปที่ 2.2



รูปที่ 2.2 หลักการทำงานของตัวกำจัดสัญญาณเสียงสะท้อน

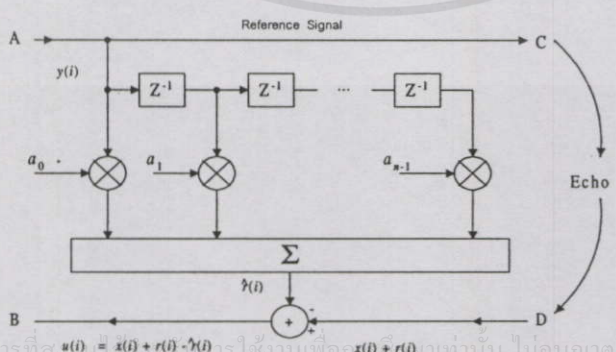
รูปที่ 2.2 แสดงถึงบางส่วนของการเชื่อมต่อสื่อสารแบบ 4 สายที่อยู่ใกล้กับส่วนของการสื่อสารแบบ 2 สาย ด้วยการส่งผ่านสัญญาณเสียงไปในทิศทางเดียวระหว่างพอร์ท A ไปยังพอร์ท C และระหว่างพอร์ท D ไปยังพอร์ท B สัญญาณที่แสดง far-end talker แสดงด้วยสัญลักษณ์ $y(i)$ สัญญาณสะท้อนแสดงด้วย $r(i)$ และสัญญาณ near-end talker แสดงด้วย $x(i)$ สัญญาณสะท้อนที่

ไม่ต้องการจะรวมอยู่กับสัญญาณ near-end talker บน part D สัญญาณที่ถูกรับได้จาก far-end talker $y(i)$ ก็จะถูกใช้ให้เป็นเหมือนกับสัญญาณอ้างอิงของตัวกำจัดสัญญาณสะท้อน โดย $y(i)$ จะถูกใช้เพื่อทำการสร้างสัญญาณลอกเลียนแบบซึ่งจะเรียกว่า $\hat{r}(i)$ จากนั้นสัญญาณลอกเลียนแบบที่ได้นี้จะถูกนำไปหักลบออกจาก near-end talker ที่ถูกรวมอยู่กับสัญญาณสะท้อนก็จะทำให้ได้เป็นการส่งผ่านเฉพาะ near-end signal $u(i)$ กลับไปยังด้าน far-end listener เท่านั้น เมื่อ $u(i) = x(i) + r(i) - \hat{r}(i)$ ในทางอุดมคติแล้วส่วนที่เหลือของสัญญาณสะท้อน (echo error) จะมีค่าเป็น $e(i) = r(i) - \hat{r}(i)$ ซึ่งจะมีค่าน้อยมากหลังจากได้ทำการกำจัดสัญญาณสะท้อนแล้ว

ตัวกำจัดสัญญาณสะท้อนจะทำการสร้างสัญญาณที่จำลองเลียนแบบสัญญาณสะท้อนโดยการป้อนสัญญาณอ้างอิงเข้าไปยังทรานส์เวอร์สซัลฟิลเตอร์ซึ่งแสดงไว้ดังรูปที่ 2.3 ถ้าค่าของฟังก์ชันถ่ายโอน (Transfer function) ของทรานส์เวอร์สซัลฟิลเตอร์มีความเหมือนกันทุกประการกับส่วนของสัญญาณที่เป็นสัญญาณสะท้อน $r(i)$ ตัวสัญญาณสะท้อนจำลอง $\hat{r}(i)$ ที่ได้จากการลอกเลียนแบบก็จะมีความเหมือนกันทุกประการกับสัญญาณจริง ดังนั้นจึงทำให้ได้ผลลัพธ์ออกมาเป็นการกำจัดโดยรวม เนื่องจากค่าของฟังก์ชันถ่ายโอนของส่วนที่เกิดการสะท้อนจาก port C ไปยัง D โดยปกติแล้วไม่สามารถทราบค่าได้ ตัวกำจัดสัญญาณสะท้อนก็จะใช้วิธีการปรับค่าสัมประสิทธิ์ของทรานส์เวอร์สซัลฟิลเตอร์ เพื่อที่จะลดค่าของ $e(i)$ ลง อัลกอริทึมที่ทำการปรับค่า (Adaptive Algorithm) ของสัมประสิทธิ์จะทำการอนุมานได้จากค่าของ $e(i)$ ที่มีอยู่

2.2.3 อัลกอริทึมการกำจัดสัญญาณสะท้อน (Echo cancellation algorithm)

สัญญาณสะท้อน $r(i)$ ที่เวลา i ในรูปที่ 2.3 สามารถที่จะเขียนให้เป็นการประสานกันระหว่างสัญญาณ far-end reference signal $y(i)$ กับ h_k ซึ่งเป็นผลตอบสนองต่ออิมพัลส์แบบไม่ต่อเนื่อง (Discrete impulse response) ของวงจรส่วนที่ทำให้เกิดการสะท้อน (Echo-path) ระหว่างพอร์ท C กับ D ได้เป็น



รูปที่ 2.3 การสร้างสัญญาณที่จำลองเลียนแบบสัญญาณสะท้อนด้วยทรานส์เวอร์สซัลฟิลเตอร์

$$r(i) = \sum_{k=0}^{N-1} h_k\, y(i-k) \qquad (2.1)$$

สมมติว่ามีความเป็นเชิงเส้นและค่าของ N มีช่วงที่จำกัด (Finite duration) ในการตอบสนองของวงจรส่วนที่ทำให้เกิดการสะท้อน และตัวกำจัดเสียงสะท้อนที่มี N tap จะปรับค่าของสัมประสิทธิ์ N ตัว คือค่า a_k ของทรานส์เวอร์สซัลฟิลเตอร์ เพื่อที่จะสร้างสัญญาณที่จำลองการสะท้อนของสัญญาณ $r(i)$ ซึ่งกำหนดได้เป็น

$$\hat{r}(i) = \sum_{k=0}^{N-1} a_k\, y(i-k) \qquad (2.2)$$

จะเห็นได้อย่างชัดเจนว่าถ้า $a_k = h_k$ เมื่อ $k = 0, \ldots, N\text{-}1$ ก็จะทำให้ $\hat{r}(i) = r(i)$ ที่ทุก ๆ ค่าของเวลา i ซึ่งหมายความว่าสัญญาณสะท้อนถูกกำจัดออกไปได้อย่างสมบูรณ์

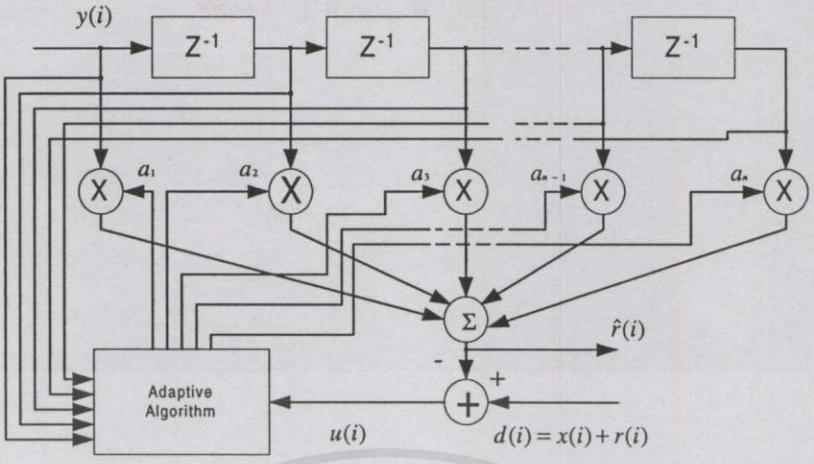
เนื่องจากโดยทั่วไปแล้วค่าของผลตอบสนองต่ออิมพัลส์ h_k ของวงจรส่วนที่ทำให้เกิดการสะท้อนจะไม่ทราบค่า และอาจจะมีการเปลี่ยนแปลงอย่างช้า ๆ ตามเวลา [2] ดังนั้นจึงมีความจำเป็นที่จะต้องให้อัลกอริทึมที่ทำการปรับค่าของสัมประสิทธิ์เป็นแบบวงปิด (close-loop) เพื่อที่จะทำให้เกิดค่าต่ำที่สุดของค่า MSE (Mean squared error) ที่ได้จากผลต่างระหว่างค่าของสัญญาณสะท้อนจริงกับค่าของสัญญาณสะท้อนจำลองของมัน จากรูปที่ 2.3 จะเห็นได้ว่าค่าของ near-end error signal $u(i)$ จะถูกรวมเอาไว้ด้วยส่วนของ $r(i) - \hat{r}(i)$ ซึ่งเป็น echo-path error กับส่วนของสัญญาณเสียงพูดด้าน near-end $x(i)$ ซึ่งจะไม่มีความสัมพันธ์ใด ๆ กับสัญญาณเสียงพูดด้าน far-end [3] $y(i)$ ทำให้ได้สมการเป็นดังนี้

$$E[u^2(i)] = E[x^2(i)] + E[e^2(i)] \qquad (2.3)$$

เมื่อ E ในที่นี้แสดงถึง expectation operator จากสมการค่าที่อยู่ในเทอมของสัญญาณสะท้อน $E[e^2(i)]$ จะต้องมีค่าต่ำที่สุดเมื่อค่าของเทอมซ้ายมือในสมการมีค่าต่ำที่สุด ถ้าในกรณีที่ไม่มีสัญญาณเสียงพูดด้าน near-end $(x(i) = 0)$ ค่าที่ต่ำที่สุดนี้ก็จะถูกค้นหาจนพบได้โดยการปรับค่าของสัมประสิทธิ์ a_k ไปตามทิศทางของ negative gradient ของ $E[e^2(i)]$ ในแต่ละ step ด้วยการปรับค่าในสมการ Update Equation

2.2.4 วงจรกรองปรับตัวได้ แบบ LMS อะแดปทีฟทรานสเวอร์สซัลฟิลเตอร์ (LMS adaptive transversal filter)

โครงสร้างการทำงานของวงจรกรองปรับตัวได้แบบอะแดปทีฟทรานสเวอร์สซัลฟิลเตอร์แสดงได้ดังรูปที่ 2.4



รูปที่ 2.4 บล็อกไดอะแกรมของ อะแดปทีฟทรานส์เวอร์สซัลฟิลเตอร์

จากรูปเมื่อกำหนดให้ค่าจริง (Real value) ของ input signal vector เป็น $y(i)$ และสัมประสิทธิ์ของฟิลเตอร์เป็น $a(i)$

$$y(i)=\begin{bmatrix} y_1(i) \\ y_2(i) \\ \vdots \\ y_n(i) \end{bmatrix} \quad ; \quad a(i)=\begin{bmatrix} a_1(i) \\ a_2(i) \\ \vdots \\ a_n(i) \end{bmatrix} \tag{2.4}$$

สัญญาณเอาท์พุตของฟิลเตอร์จะได้เป็น

$$\hat{r}(i)=\sum_{k=1}^{n} a_k(i)y_k(i) \tag{2.5}$$

และสามารถเขียนให้อยู่ในรูปเมตริกซ์ได้เป็น

$$\hat{r}(i)=a^T(i)y(i)=y^T(i)a(i) \tag{2.6}$$

เมื่อกำหนดให้สัญญาณที่ต้องการ เป็น $d(i)$ จะทำให้ได้

$$e(i)=d(i)-\hat{r}(i)=d(i)-a^T(i)y(i) \tag{2.7}$$

เมื่อหาค่า square error จะได้เป็น

$$e^2(i)=\left[d(i)-a^T(i)y(i)\right]^2$$

จากความสัมพันธ์ $a^T(i)y(i)=y^T(i)a(i)$ ดังนั้น

$$e^2(i) = d^2(i) - 2d(i)a^T(i)y(i) + a^T(i)y(i)y^T(i)a(i) \qquad (2.8)$$

จากสมการที่ (2.8) เมื่อหาค่า mean square error ก็จะได้เป็น $E[e^2(i)]$ นั่นคือ

$$E[e^2(i)] = E[d^2(i)] - 2\Phi^T(d,y)a(i) + a^T(i)\Phi^T(y,y)a(i) \qquad (2.9)$$

ซึ่งจะเห็นได้ว่าค่าของ a ไม่ได้เป็นค่าทางสถิติ [3] แต่จะเป็นค่าคงที่ในการทำ expectation ดังนั้นจะกำหนดให้ค่าของสหสัมพันธ์ข้ามสัญญาณ (cross-correlation) ระหว่างสัญญาณอินพุตกับสัญญาณตอบสนองที่ต้องการ เป็นดังนี้

$$\Phi(d,y) \underline{\Delta} E[d(i)y(i)] = E\begin{bmatrix} d(i)y_1(i) \\ d(i)y_2(i) \\ \vdots \\ d(i)y_n(i) \end{bmatrix} \qquad (2.10)$$

ซึ่งจะเป็นค่าสหสัมพันธ์ข้ามสัญญาณระหว่างสัญญาณอินพุตกับสัญญาณตอบสนองที่ต้องการในชั่วขณะ (stationary process) ทำให้ได้เป็น

$$E[d(i)y_1(i)] = \lim_{N\to\infty}\frac{1}{N}\sum_{k=1}^{N} d(k)y_1(k) \qquad (2.11)$$

สมการที่ (2.11) นี้จะถูกนำไปใช้กับสัญญาณอินพุตทุก ๆ ค่า และจากการที่ได้กำหนดให้ค่าเมตริกซ์สหสัมพันธ์ข้ามสัญญาณของตัวแปรทางอินพุตเป็น

$$\Phi(y,y) \underline{\Delta} E[y(i)y^T(i)]$$

$$\Phi(y,y) = E\begin{bmatrix} y_1(i)y_1(i) & y_1(i)y_2(i) & \cdots & y_1(i)y_n(i) \\ y_2(i)y_1(i) & y_2(i)y_2(i) & \cdots & y_2(i)y_n(i) \\ \vdots & \vdots & & \vdots \\ y_n(i)y_1(i) & y_n(i)y_2(i) & \cdots & y_n(i)y_n(i) \end{bmatrix} \qquad (2.12)$$

เนื่องจาก $E[e^2(i)]$ เป็นฟังก์ชันสมการกำลังสอง จึงจะทำการพิจารณาพื้นผิวดังกล่าวเป็นแบบรูปร่างก้นถ้วย (bowl shape) ต่อจากนั้นก็จะใช้กระบวนการปรับค่า (adaptive process) ที่ทำงานอย่างต่อเนื่องเพื่อหาจุดต่ำที่สุดของพื้นผิว และวิธีการที่นำมาใช้ก็เป็นแบบ steepest descent method โดยใช้เกรเดียนต์เวกเตอร์ (gradient vector) มากำหนดบริเวณของเส้นทางการเคลื่อนที่ ซึ่งค่าเกรเดียนต์เวกเตอร์ของสัญญาณผิดพลาดจะมีค่าเป็น

$$\nabla\left[e^{2}(i)\right]=\begin{bmatrix}\dfrac{\partial\left[e^{2}(i)\right]}{\partial a_{1}}\\ \dfrac{\partial\left[e^{2}(i)\right]}{\partial a_{2}}\\ \vdots\\ \dfrac{\partial\left[e^{2}(i)\right]}{\partial a_{n}}\end{bmatrix} \qquad (2.13)$$

เมื่อแทนค่าของสมการที่ (2.9) ลงในสมการที่ (2.13) ก็จะได้เป็น

$$\nabla\left[e^{2}(i)\right]=-2\Phi(d,y)a(i)+2\Phi^{T}(y,y)a(i) \qquad (2.14)$$

จากสมการ (2.9) ในเทอมแรกค่า $E[d^{2}(i)]$ นั้นจะเป็นค่าคงที่เมื่อเทียบกับ a ด้วยเหตุนี้ค่าของอนุพันธ์ที่ได้จึงกลายเป็นศูนย์ ส่วนเทอมที่สองจะได้จากการคูณแล้วทำการหาค่าอนุพันธ์เทียบกับ a ในแต่ละแถว ด้วยเหตุที่เป็นการหาอนุพันธ์บางส่วนก็จะทำให้ทุกเทอมยกเว้นเทอมที่มีค่าของ a อยู่ด้วยมีค่าเป็นศูนย์ ส่วนเทอมสุดท้ายก็สามารถหาได้โดยการคูณและหาค่าอนุพันธ์ ซึ่งในที่นี้จะสังเกตเห็นว่า $\Phi_{nK}(i)=\Phi_{Kn}(i)$ สำหรับการหาค่าของเกรเดียนต์เวกเตอร์ของสัมประสิทธิ์ a_{LMS} ก็จะกำหนดให้ค่าของเกรเดียนต์เป็นศูนย์ ดังนั้น

$$\Phi(d,y) = \Phi(y,y)a_{LMS} \qquad (2.15)$$

เมื่อหาค่าของ a_{LMS} จะได้เป็น

$$a_{LMS} = \Phi^{-1}(y,y)\Phi(d,y) \qquad (2.16)$$

ค่าต่ำที่สุดของ MSE จะหาได้จาก

$$E\left[e^{2}(i)\right]_{MIN}=E\left[d^{2}(i)\right]-a_{LMS}^{T}\Phi(d,y) \qquad (2.17)$$

สมการที่ (2.16) สามารถที่จะหาค่าโดยตรงหลังจากการคำนวณหาเมตริกซ์สหสัมพันธ์ข้ามสัญญาณ ซึ่งจะทำให้ค่าของสัมประสิทธิ์สามารถถูกกำหนดได้อย่างรวดเร็ว [4] ส่วนในเวลาที่เป็นลำดับเวลา (step time) ถัดไปก็จะทำการปรับค่าของสัมประสิทธิ์ โดยใช้อัลกอริทึมแบบทำซ้ำ (Iterative algorithm) การปรับค่าของสัมประสิทธิ์ในแต่ละลำดับเวลาจะใช้สมการที่ (2.18)

$$a(i+1)=a(i)+\mu\hat{\nabla}\left[e^{2}(i)\right] \qquad (2.18)$$

เมื่อกำหนดให้

$a(i)$: เวคเตอร์ของสัมประสิทธิ์ก่อนทำการปรับค่า

$a(i+1)$: เวคเตอร์ของสัมประสิทธิ์ หลังจากทำการปรับค่า

μ : ค่าคงที่ สำหรับใช้ควบคุมอัตราการลู่เข้า

$\hat{\nabla}[e^2(i)]$: ค่าโดยประมาณของเกรเดียนต์เวกเตอร์เมื่อเทียบกับ a

ค่าโดยประมาณของเกรเดียนต์เวกเตอร์จะถูกหาได้โดยการสมมุติว่าเกรเดียนต์ของค่าเฉลี่ย (expected value) มีค่าเท่ากับเกรเดียนต์เออร์เรอร์ของมัน ดังนั้น

$$\hat{\nabla}[e^2(i)] \approx \nabla[e^2(i)] \qquad (2.19)$$

เมื่อทำการหาอนุพันธ์ จะได้

$$\nabla[e^2(i)] = 2e(i)\nabla[e(i)] \qquad (2.20)$$

จากสมการ (2.7) ของค่าความผิดพลาดจะได้เป็น

$$e(i) = d(i) - a^T(i)y(i)$$

เมื่อหาค่าอนุพันธ์ ค่าของเกรเดียนต์ก็จะกลายเป็น

$$\nabla[e(i)] = -y(i) \qquad (2.21)$$

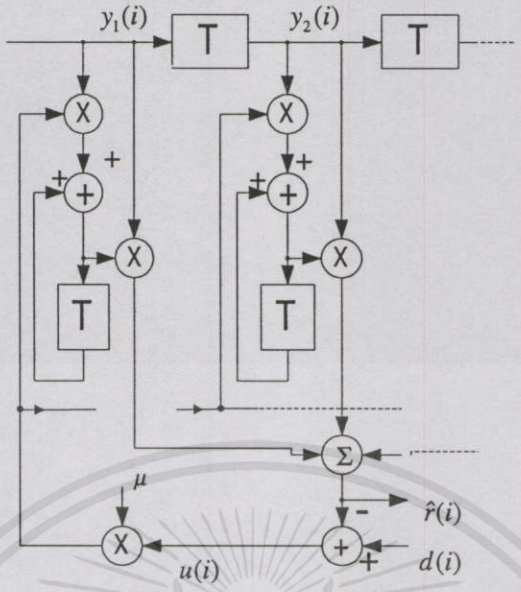
แต่เนื่องจากค่าของ $d(i)$ ไม่ได้เป็นฟังก์ชันของ $a(i)$ ดังนั้นเมื่อแทนค่าของสมการ (2.21) ลงในสมการ (2.20) ทำให้ได้เป็น

$$\nabla[e^2(i)] = -2e(i)y(i) \qquad (2.22)$$

และ

$$a(i+1) = a(i) + 2\mu e(i)y(i) \qquad (2.23)$$

ซึ่งสมการ (2.23) เป็นสมการสำหรับปรับค่า (Update Equation) ของฟิลเตอร์และรูปที่ 2.5 แสดงบล็อกไดอะแกรมการทำงานของอัลกอริทึม LMS อะแดปทีฟทรานส์เวอร์สซัลฟิลเตอร์ในแต่ละ tap



รูปที่ 2.5 บล็อกไดอะแกรมของ LMS อะแดปทีฟทรานส์เวอร์ซัลฟิลเตอร์

2.2.5 การตรวจจับสัญญาณทางด้านใกล้ (Near-End Speech Detection)

เมื่อผู้พูดทั้งด้าน near-end และ far-end พูดพร้อมกัน ซึ่งสภาวะเงื่อนไขนี้จะเรียกว่า "double talk" จะพบว่าค่าสัญญาณ $u(i)$ ของรูปที่ 2.2 ก็จะประกอบไปด้วยส่วนประกอบของ near-end talker $x(i)$ รวมอยู่กับส่วนที่เหลือตกค้างของการกำจัดสัญญาณสะท้อน $e(i)$ จึงจำเป็นจะต้องมีการหยุดค้างการกำจัดสัญญาณสะท้อนไว้ชั่วคราวในระหว่างที่เกิด double talk ทั้งนี้เพื่อหลีกเลี่ยงการเกิดไดเวอร์เจนซ์ (Divergence) สถานะการเกิด double talk สามารถที่จะทำการตรวจพบได้โดยใช้ตัวตรวจจับเสียงพูดด้านใกล้ ดำเนินการที่ near-end และ far-end signal $s(i)$ และ $y(i)$ ตามลำดับ

จากอัลกอริทึมการตรวจจับสัญญาณด้านใกล้ของ A.A.Geiget [5] ที่ประกอบไปด้วยส่วนของการแจ้งเตือนการมีสัญญาณเสียงพูดด้านใกล้ โดยเมื่อใดก็ตามที่

$$|s(i)| = |x(i)+r(i)| \geq \frac{1}{2}\max\{|y(i)|,|y(i-1)|,\ldots,|y(i-N)|\} \qquad (2.24)$$

เมื่อ N เป็นจำนวนของตัวอย่างที่มีอยู่ในหน่วยความจำของอะแดปทีฟทรานส์เวอร์สซัลฟิลเตอร์ จึงมีความจำเป็นในการที่จะต้องทำการเปรียบเทียบ $s(i)$ กับสัญญาณที่เพิ่งจะผ่านไปของสัญญาณด้าน far-end มากกว่าที่จะเปรียบเทียบกับ $y(i)$ ทั้งนี้เพราะว่าไม่ทราบค่าของเวลาหน่วงในส่วนของ echo-path สำหรับค่าของแฟคเตอร์ 0.5 ที่อยู่ในสมการที่ (2.24) มีพื้นฐานอยู่บนสมมติฐานที่ว่ามีความสูญเสียใน echo-path ขณะที่ผ่านตัวไฮบริดอย่างน้อยที่สุด 6 dB อัลกอริทึมจะดำเนินการส่งผลในทันทีที่พบว่ามีค่าของกำลังงานที่เปรียบเทียบได้มากกว่าค่าที่มีอยู่ในหน่วยความจำ

จากอัลกอริทึมในสมการที่ (2.24) สามารถปรับให้มีความเหมาะสมในการใช้งานได้มากขึ้นด้วยการประมาณกำลังงานชั่วขณะของ $y(i)$ และ $s(i)$ ตามลำดับ สำหรับการประมาณกำลังงานของสัญญาณที่เพิ่งจะผ่านไปของสัญญาณด้าน far-end $y(i)$ ที่ได้รับและสัญญาณไฮบริดด้าน near-end $s(i)$ การประมาณกำลังงานเหล่านี้ทำได้โดยการคำนวณแบบเวียนซ้ำ (recursive) ดังสมการที่ (2.26) และ (2.27)

$$\bar{s}(i+1) = (1-\alpha)\bar{s}(i) + \alpha|s(i)| \qquad (2.25)$$

$$\bar{y}(i+1) = (1-\alpha)\bar{y}(i) + \alpha|y(i)| \qquad (2.26)$$

เมื่อค่าของ α เป็นค่าของอัตราขยายของวงจรกรอง

$$|\bar{s}(i)| \geq \max\{\bar{y}(i), \bar{y}(i-1), \ldots, \bar{y}(i-N)\} \qquad (2.27)$$

เนื่องจากการทำงานของอัลกอริทึมการตรวจจับสัญญาณเสียงพูดด้านใกล้จะทำการตรวจจับในลักษณะการหากำลังงานสูงสุดชั่วขณะ (short-term power peak) มันจึงมีความต้องการในการแจ้งเตือนอย่างต่อเนื่องของสัญญาณเสียงพูดด้าน near-end ซึ่งก็จะทำให้ได้การตรวจจับที่ถูกต้องของสัญญาณ near-end ตามต้องการ

บทที่ 3
การประยุกต์ใช้งานของ FPGA

3.1 บทนำ

อุปกรณ์ FPGA เป็นอุปกรณ์ทางด้านอิเล็กทรอนิกส์ที่มีการออกแบบให้มีโครงสร้างการทำงานแบบดิจิตอลโดยให้ผู้ใช้งานสามารถกำหนดหรือจัดคุณสมบัติการทำงาน (Configuration) ได้ตามความต้องการ แต่อย่างไรก็ตามภายใต้เงื่อนไขของแบบแผนโครงสร้าง (Paradigm) ของ FPGA แต่ละชนิดก็จะพบว่าในแต่ละแบบแผนโครงสร้างก็ให้คุณลักษณะที่โดดเด่นแตกต่างกันไป จึงอาจจะมีความเหมาะสมต่องานที่แตกต่างกันตามไปด้วย ดังนั้นการนำอุปกรณ์ FPGA ไปใช้งานสิ่งที่ผู้ใช้จะต้องคำนึงถึงก็คือชนิดและคุณสมบัตของ FPGA เพื่อให้การนำไปใช้งานได้เกิดประโยชน์อย่างสูงสุด

3.2 แบบแผนโครงสร้างของ FPGA (FPGA paradigm)

เมื่อมีการเริ่มต้นสร้าง Digital electronics ก็ได้มีการพยายามค้นหาแนวทางในการออกแบบวงจรและสิ่งที่จำเป็นต่อการออกแบบที่สุดคือส่วนประกอบที่สำคัญ 3 ส่วนของวงจรที่จะต้องมีคือ

1) มีตัวดำเนินการข้อมูล (Data operators) โดยเกทจะถูกสร้างขึ้นเพื่อให้สามารถใช้งานตามวัตถุประสงค์ของการทำงานได้อย่างสมบูรณ์ในการส่งผ่าน (State transform) สถานะทางอินพุตไปเป็นสถานะทางเอาท์พุต

2) มีส่วนของอุปกรณ์สำหรับเก็บข้อมูล (Storage element) ดังเช่น แลทช์ (Latches) ฟลิปฟลอป (Flip-flop) ซึ่งจะทำหน้าที่เก็บผลลัพธ์แต่ละสถานะ (state operand)

3) มีการเชื่อมต่อของสาย (Wires) ระหว่างจุด มีระบบบัสเพื่อเชื่อมต่อสื่อสารส่งค่าระหว่างอุปกรณ์ที่เก็บข้อมูลกับส่วนของตัวดำเนินการข้อมูล

โดยทั่วไปสารกึ่งตัวนำจะแสดงให้เห็นถึงความสามารถในการเป็นอุปกรณ์ขั้นพื้นฐานที่มีอยู่อย่างหลากหลายทั้งในด้านแนวความคิดและรูปแบบ ในอดีตการดำเนินงานกับอุปกรณ์เหล่านี้จะต้องใช้อยู่บนระบบของแผงวงจรชั่วคราว (Breadboard) เป็นการเชื่อมต่อขาของเกทและฟลิปฟลอปโดยตรงเพื่อให้อุปกรณ์เหล่านี้ทำงานร่วมกันบนแผ่นวงจรพิมพ์ (Printed circuit board) แต่อย่างไรก็ตาม เมื่อไอซี (Integrated circuit) ได้มีวิวัฒนาการมากขึ้น ทำให้ไม่สามารถเข้าถึงการทำงานของไอซีได้โดยตรงแต่จะเป็นการใช้งานไอซีในลักษณะกลุ่มวงจรลอจิก เช่น กลุ่มของ TTL แต่เมื่อได้มีการนำไอซีรูปแบบเดิมที่คุ้นเคยมาใช้งานอีกโดยนำกลุ่มฟังก์ชันทางลอจิกมาทำการโปรแกรมเพื่อเชื่อมต่อสายไฟในระหว่างการผลิต ดังเช่น อุปกรณ์ประเภทที่มีฟังก์ชันการทำงานเฉพาะทาง (Application specific integrated-circuit , ASIC) ก็ทำให้สามารถเข้าถึงการทำงานของไอซีได้

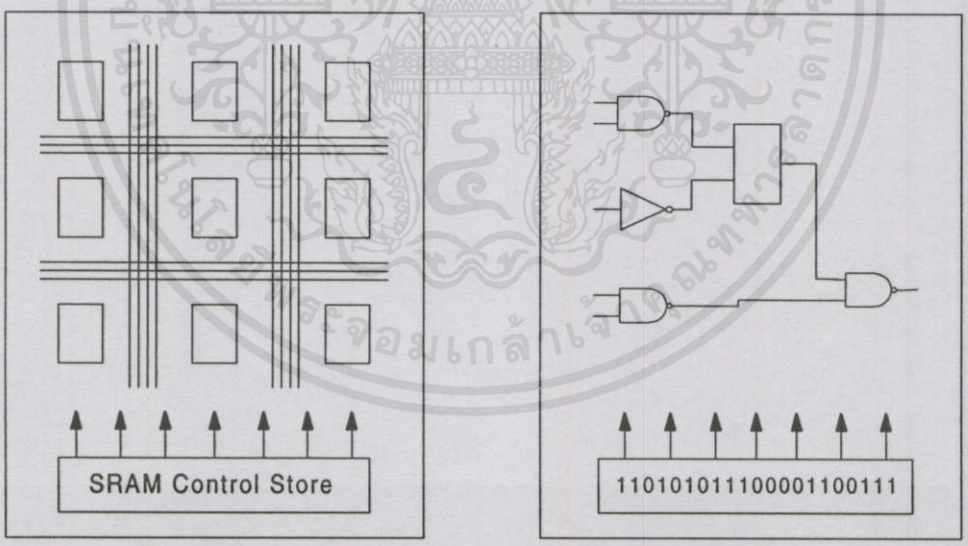
โดยตรง แต่วิธีการนี้ยังมีต้นทุนที่สูงมากและต้องใช้ระยะเวลาการดำเนินงานที่ยาวนานนับตั้งแต่การออกแบบไปจนถึงจบขั้นตอนกระบวนการผลิต

เมื่อมีการพัฒนา FPGA ขึ้นมาโดยเป็นอุปกรณ์ที่รวมเอาอุปกรณ์พื้นฐานสามชนิดเข้าไว้ด้วยกัน คือ อุปกรณ์เก็บข้อมูล อุปกรณ์ลอจิก และสายสัญญาณตัวนำ ที่สามารถทำการโปรแกรมเพื่อสร้างให้เป็นวงจรที่ทำงานตามกำหนดได้ด้วย ดังนั้น FPGA จึงเป็นทางออกของการแก้ปัญหาที่เป็นจุดอ่อนของอุปกรณ์ทั้งแบบชิ้นส่วนมาตรฐาน (Standard parts) และอุปกรณ์แบบ ASIC ที่สำคัญคือมันยอมให้มีการเข้าถึงได้ทุกส่วนของอุปกรณ์ที่อยู่ในตัวไอซี จึงก่อให้เกิดวิวัฒนาการแบบใหม่ของเทคโนโลยีพื้นฐานทางอิเล็กทรอนิกส์ส่งผลให้การออกแบบระบบสามารถมีทางเลือกได้ถึง 3 ทาง คือ

1) การใช้ชิ้นส่วนมาตรฐาน โดยใช้อุปกรณ์ที่มีฟังก์ชันตามมาตรฐานของผู้ผลิตที่มีในตลาด ซึ่งจะทำให้มีต้นทุนการผลิตต่ำ

2) การใช้ ASIC ผู้ใช้สามารถกำหนดฟังก์ชันการทำงานตามความต้องการของตนเองได้ แต่ก็มีต้นทุนการผลิตสูง

3) การใช้ FPGA ผู้ใช้กำหนดฟังก์ชันการทำงานตามความต้องการของตนเอง แต่ทำให้มีต้นทุนการผลิตต่ำได้ในจำนวนการผลิตจำนวนหนึ่ง



รูปที่ 3.1 แนวความคิดทางโครงสร้างของ FPGA

แนวคิดที่เป็นโครงสร้างง่าย ๆ ของ Reprogrammable FPGA นั้นดังแสดงในรูปที่ 3.1 โดยจะมีตัวควบคุมที่เป็น Static RAM มีหน่วยของ Function unit และสายตัวนำที่สามารถกำหนดได้โดยกำหนดใน Control store เพื่อระบุชิ้นส่วนของวงจรเกท และการเชื่อมต่อต่าง ๆ ส่วนประเภทหรือตระกูลของ FPGA จะถูกกำหนดโดยอาศัยความแตกต่างกันของโครงสร้างทางสถาปัตยกรรม

ภายในซึ่งเป็นหน่วยเล็ก ๆ ที่เรียกว่า Function unit กับข่ายโครงสร้างการเชื่อมต่อภายใน (Interchip wiring organization) เนื่องจากวงจรที่ถูกสร้างขึ้นโดย FPGA จะเป็นการกำหนดค่าภายในของ SRAM control store การนำไปประยุกต์ใช้งานของ FPGA จึงมีได้หลายแนวทางคือ

1) การสร้างชิ้นงานแบบทันที (Instantaneous implementation) จะมีลักษณะเหมือนกับการเชื่อมต่อกันบนหน่วยความจำโดยตรง (Memory-wiring) ทำให้ใช้เวลาสร้างสั้นมาก ชิ้นงานที่ออกแบบสามารถถูกนำไปใช้งานได้ทันทีซึ่งจะแตกต่างจากการออกแบบโดยใช้ VLSI ที่ต้องใช้เวลาหลายสัปดาห์ในการผลิตงานตามแบบที่ได้ออกแบบไว้จึงมีความได้เปรียบในการสร้างงานต้นแบบ

2) มีความยืดหยุ่นในการกำหนดการทำงานของโครงสร้าง (Dynamic reconfiguration) เนื่องจากสถาปัตยกรรมบางส่วนของ FPGA สามารถที่จะทำการโปรแกรมได้ใหม่ในขณะที่กำลังทำงาน (Run time) ซึ่งเป็นช่วงเวลาที่ FPGA กำลังทำงานตามคำสั่งของโปรแกรม ดังนั้นชิพจึงสามารถถูกนำไปใช้งานได้ในฟังก์ชันอื่นที่แตกต่างจากคำสั่งในตอนแรก แต่ก็จะขึ้นอยู่กับความสามารถของวงจรที่เป็นโครงสร้างหลักและความพอเพียงของทรัพยากรในการปรับเปลี่ยนฟังก์ชันการทำงานของมันเองด้วย

3) มีความปลอดภัยในการรักษาข้อมูลของแบบชิ้นงาน (Design security) เนื่องจากข้อมูลการทำคอนฟิกกูเรชันของ FPGA จะไม่ปรากฏให้เห็นถ้าไม่มีการจ่ายพลังงานให้กับชิพ เป็นการเพิ่มขอบเขตของการนำไปประยุกต์ใช้งานร่วมกับระบบรักษาความปลอดภัยของข้อมูลร่วมกับระบบอื่นที่ผู้ออกแบบได้สร้างขึ้น

4) มีความยืดหยุ่นในการพัฒนาระบบ (Field programmability) ระบบงานที่สร้างโดยใช้ FPGA สามารถที่จะทำการปรับปรุงและตรวจสอบได้ในภาคสนาม โดยการดำเนินการของผู้ใช้ระบบเอง (Operator) หรือจากการควบคุมระยะไกล (Telemeter) เพื่อแก้ไขข้อบกพร่อง ซ่อมแซมส่วนที่เสียหาย หรือเพิ่มเติมฟังก์ชันการทำงานใหม่ ๆ เข้าไป

3.3 การออกแบบและสร้างชิ้นงานโดยใช้ FPGA

(Design and Implementation using FPGAs)

การออกแบบเพื่อใช้งาน FPGA จะมีขั้นตอนของการออกแบบแยกออกจากขั้นตอนของการแมพ (Mapping) อย่างชัดเจน เพราะว่าชิ้นงานต้นแบบอาจจะต้องใช้ FPGA มากกว่าหนึ่งตัวซึ่งบ่อยครั้งแบบที่ได้จะเป็นการออกแบบระบบที่มีขนาดใหญ่เพียงระบบเดียว เครื่องมือที่เป็นตัวกลาง (Medium) ในการเชื่อมต่องานทั้งสองส่วนก็มักจะมีผลกระทบมากต่อการออกแบบโดยใช้ FPGA ด้วยเช่นกัน แต่อย่างไรก็ตามการออกแบบมักจะไม่สามารถทำให้มีรูปแบบที่ง่ายต่อการทำความเข้าใจและอาจจะต้องใช้ความพยายามอย่างมากเมื่อเป็นงานที่มีความคิดที่ค่อนข้างแปลกใหม่ดังนั้น

แนวคิดที่เกี่ยวกับการออกแบบจึงมักจะมุ่งไปในเรื่องของปัญหาปลีกย่อยต่าง ๆ ที่นำมาใช้ในการออกแบบ เช่น การลดรูปวงจรลอจิก (Logic minimization) การออกแบบด้วยเทคนิค State machine หรืออาจจะมุ่งไปที่การลดขนาดของแบบให้เป็นสมการทางคณิตศาสตร์เพื่อให้ได้วิธีการแก้ปัญหาที่สมบูรณ์ เพื่อเป็นการแก้ปัญหานี้จึงต้องมีการแยกขั้นตอนการออกแบบกับการสร้างชิ้นงานออกจากกันทำให้พิจารณาได้ว่า ตัวแบบในความหมายที่เป็นนามธรรม จะหมายถึงสิ่งที่แสดงการเชื่อมต่อกันของส่วนประกอบต่างๆ ที่มีการกำหนดคุณสมบัติไว้อย่างชัดเจนทุกชิ้นส่วน ชิ้นส่วนบางชิ้นอาจจะมีการเรียกใช้ซ้ำ ๆ กันและอาจจะถูกประกอบจากชิ้นส่วนปลีกย่อยอื่น ๆ โครงสร้างของการออกแบบในลักษณะนี้จะพบเห็นได้ในทุก ๆ แขนงงานทางวิศวกรรม เช่น ในงานอิเล็กทรอนิกส์โครงสร้างนี้พบในการเชื่อมต่อของสายไฟระหว่างเกทในงานทางเครื่องกลจะพบได้ในการจับยึดด้วยหมุดหรือขอเกี่ยวที่จับยึดระหว่างคานต่าง ๆ เป็นต้น

อย่างไรก็ตามแบบงานที่นำไปสร้างเป็นชิ้นงานได้จริงนั้น ชิ้นส่วนแบบนามธรรมซึ่งมีการกำหนดในเชิงโครงสร้างไว้แล้วจะต้องสามารถแมพลงบนชิ้นส่วนที่เป็นทางกายภาพที่มีอยู่ในเครื่องมือตัวกลางในการออกแบบนั้นได้ ถ้ากระบวนการนี้สามารถทำได้ก็จะทำให้เกิดชิ้นงานขึ้นมาได้จริง ด้วยเหตุนี้จึงสามารถกำหนดคุณลักษณะเชิงปริมาณของแบบเพื่อให้เกิดความมั่นใจในคุณภาพของแบบงานได้โดยจะพิจารณาจากลักษณะต่อไปนี้

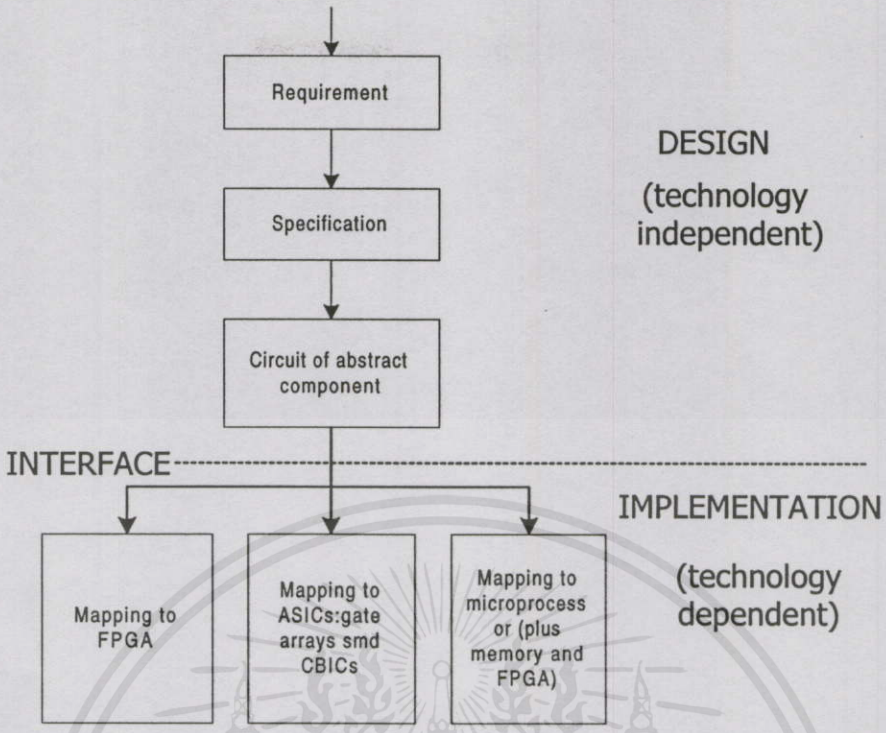
1) พื้นที่ใช้งาน (Space) แสดงด้วยจำนวนของอุปกรณ์ที่จะต้องใช้ ส่งผลถึงพื้นที่ใช้งานของบอร์ดซึ่งจะเป็นการช่วยให้สามารถคำนวณหาต้นทุนของชิ้นงานได้

2) ความเร็ว (Speed) แสดงด้วยความเร็วในการทำงานของงานต้นแบบที่สร้างขึ้นซึ่งก็จะเป็นการกำหนดสมรรถนะที่ต้องการของงานต้นแบบ

3) พลังงานและความต้องการกำลังงาน (Energy and power demand) การทำงานของชิ้นงานจะทำให้เกิดความสูญเสียในรูปของความร้อน จำเป็นต้องลดเพื่อให้เกิดความสูญเสียน้อยที่สุด แต่อาจจะมีผลกระทบโดยตรงต่อการจัดการในเรื่องของพื้นที่ใช้งานของบอร์ด

4) ระยะเวลาที่ใช้ในการทำงานที่เหมาะสม (Timeliness) เป็นระยะเวลาโดยรวมที่ต้องใช้ทั้งหมดก่อนชิ้นงานจะผลิตออกสู่ตลาด

ในการออกแบบและสร้างชิ้นงานจะมีการดำเนินงานดังแสดงในรูปที่ 3.2 จากโฟลว์ชาร์ทการดำเนินงานจะเห็นงานแบ่งแยกส่วนกันอย่างชัดเจนในเรื่องของการออกแบบที่มีแนวทางการทำงานของผู้ออกแบบไม่ต้องขึ้นอยู่กับเทคโนโลยีการผลิต กับอีกส่วนหนึ่งที่เป็นขั้นตอนของการสร้างชิ้นงานจริงที่แนวทางการทำงานจะขึ้นอยู่กับเทคโนโลยีที่เลือกใช้ โดยอาจจะแบ่งเป็นแนวทางสำหรับการออกแบบได้หลายรูปแบบตามลักษณะการสร้างชิ้นงาน เช่น ชิ้นงานสร้างบน FPGA ชิ้นงานสร้างบนเกทอาร์เรย์ (Gate arrays) และชิ้นงานที่จะสร้างบน cell-base ASIC เป็นต้น



รูปที่ 3.2 ความสัมพันธ์ระหว่างการออกแบบและการสร้างชิ้นงาน

รายละเอียดของกระบวนการออกแบบแสดงให้เห็นแนวโน้มของหลักเกณฑ์ของการออกแบบเพราะว่ามันจะต้องป้องกันไม่ให้มีการออกแบบในลักษณะที่ทำให้ไม่สามารถสร้างเป็นชิ้นงานจริงได้ แต่การจำแนกว่างานใดเป็นการออกแบบที่ดีนั้นยังไม่มีความชัดเจนมากนัก เพราะยังขึ้นอยู่กับเทคโนโลยีที่เลือกใช้ ซึ่งผู้ออกแบบต้องมีความกล้าที่จะกำหนดรูปแบบของเทคโนโลยีเพื่อจะหาจุดที่ดีที่สุดของความต้องการโดยคำนึงถึงความเป็นไปได้ในเชิงเศรษฐศาสตร์ ด้วยดังนั้นข้อมูลบางส่วนของขั้นตอนการสร้างต้องถูกนำไปพิจารณาในขณะที่เป็นขั้นตอนการออกแบบโดยสามารถอยู่ในรูปแบบของกฎการออกแบบ (Design rules) ข้อแนะนำในทางปฏิบัติ หรือรูปแบบอื่นๆ ที่ได้จากทักษะและประสบการณ์ของผู้ออกแบบ

3.4 รูปแบบที่ใช้ในการสร้างชิ้นงาน (Implementation styles)

ถึงแม้ว่าในการสร้างชิ้นงานของระบบอิเล็กทรอนิกส์จะสามารถขยายจำนวนระดับชั้นทางกายภาพ (Physical hierarchy) ที่สามารถผลิตได้จริงทั้งในส่วนของอุปกรณ์และแผ่นวงจรพิมพ์ แต่สิ่งสำคัญที่สุดคือการทำให้แผ่นวงจรพิมพ์นั้นเป็นที่แพร่หลายจนเป็นอุปกรณ์มาตรฐาน ความก้าวหน้าในเรื่องของอุปกรณ์และเทคโนโลยีของบอร์ดทำให้เกิดรูปแบบต่าง ๆ ในการสร้างชิ้นงาน ซึ่งก็พอที่จะจำแนกได้เป็นยุคของเทคโนโลยีด้านต่าง ๆ เช่นยุคของ SSI/MSI LSI และ VLSI ที่เติบโตอย่างรวดเร็วมากในช่วง 30 ปีที่ผ่านมา

3.4.1 ยุคกลุ่มตระกูลของลอจิกยุคแรกเริ่ม

ยุคนี้จะอยู่ในช่วงของปี 1960 จนถึง 1970 ซึ่งจะเป็นแบบฉบับของ Texas Instrument และ Transistor-Transistor Logic (TTL) และกลุ่มของ National Semiconductor (กลุ่มของ 4000 series) ที่มักนิยมเรียกกันว่า Small-scale integration (SSI) รวมไปถึงลอจิกที่อยู่ในรูปของเกท ฟลิปฟลอป อุปกรณ์ด้าน Register transfer counter และ Arithmetic-logic unit (ALU) ที่อยู่ในระดับชิพ ด้วยเมื่อนำมาใช้งานในระดับของบอร์ด อุปกรณ์เหล่านั้นจะมีจำนวนที่ไม่มากนักในแต่ละบอร์ดรวมถึงขนาดของเส้นทางของข้อมูล (Data-path size) ที่มีขนาดเล็ก เช่น ขนาด 8 บิต เป็นต้น

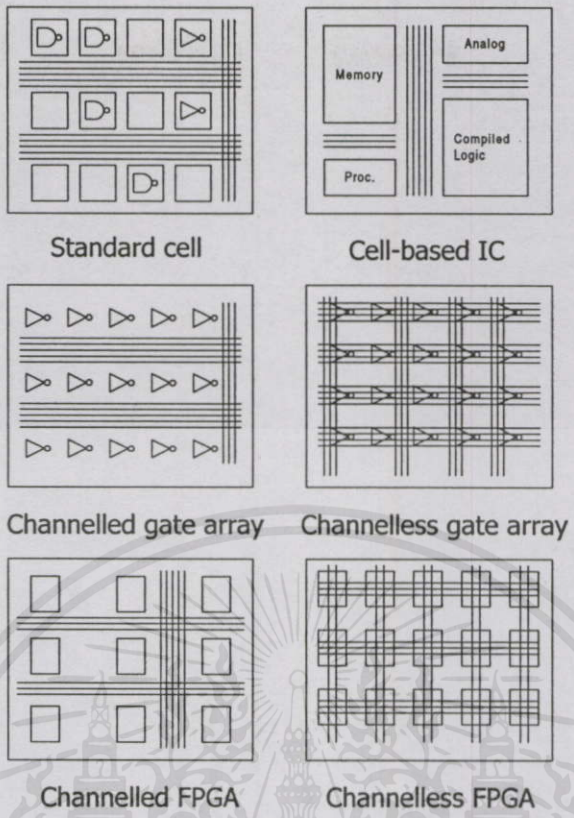
3.4.2 ยุคอุปกรณ์ LSI และ VLSI

ในปี 1970 ได้มีการคิดค้น Dynamic RAM ของบริษัท Intel และปิดยุคด้วยการที่ตัวชิพแบบ Microprocessor ขนาด 32 บิต ได้กลายเป็นอุปกรณ์มาตรฐาน ดังเช่น Motorola 68000 เป็นต้น ความก้าวหน้าของเทคโนโลยีได้ทำให้เกิดการเปลี่ยนแปลงที่สำคัญสองประการคือ ขนาดของตลาดและจำนวนที่มากมายของตลาด เช่น ตลาดของ SRAM Micro controller ในยุคนี้อุปกรณ์ทางลอจิกของยุคแรกยังถูกใช้งานอยู่อย่างต่อเนื่องและมีปริมาณที่มากขึ้นอยู่ภายในชิพเดียวกันขนาดของบอร์ดก็จะโตมากขึ้นและมักจัดให้ทุกอย่างอยู่บนบอร์ดเดียวกัน มีการพยายามลดต้นทุนการผลิตลง โดยลดจำนวนขา การเชื่อมต่อ และมีการกำหนดขาการเชื่อมต่อบนแผ่นวงจรพิมพ์ด้วยอุปกรณ์ Connector เพื่อการนำระบบย่อยเหล่านี้ไปใช้งาน

3.4.3 ยุคของอุปกรณ์ ASIC

เริ่มจากในปี 1980 เมื่อได้มีการพัฒนาอุปกรณ์ให้มีความหนาแน่นในชิพได้มากขึ้นควบคู่ไปกับการทำให้มีความซับซ้อนขึ้นในรูปแบบการทำงานจนต้องใช้เทคโนโลยีที่ใช้ในการผลิตมาสร้างอุปกรณ์ให้เป็น ASIC แทนการใช้อุปกรณ์มาตรฐาน เช่น อุปกรณ์ Mask-programmable gate arrays อุปกรณ์ Cell-based IC (CBIC) อุปกรณ์ Programmable array logic (PAL) อุปกรณ์ Field-programmable logic array (FPLA) รวมไปถึงอุปกรณ์ที่สามารถโปรแกรมได้รุ่นแรก ๆ ที่ปรากฏให้เห็นได้ในระหว่างยุคที่ใช้ ASIC เป็นเทคโนโลยีพื้นฐานเพื่อการผลิต

การจัดแบ่งกลุ่มประเภทของ ASIC โดยทั่วไปแล้วได้มีการแบ่งเป็น 3 กลุ่มคือเกทอาร์เรย์ CBIC และ PLD ส่วน FPGA นั้นจะถูกจัดอยู่ในกลุ่มของ PLD ในแต่ละกลุ่มของ ASIC จะมีความแตกต่างกันทั้งทางด้านรูปแบบของโครงสร้างและความต้องการในกระบวนการผลิต เช่น CBIC มีความต้องการทุกขั้นตอนในกระบวนการสร้างขึ้นเป็นตัวไอซี และยังต้องใช้เวลาที่ยาวนานกว่ามากเมื่อเปรียบเทียบกับเกทอาร์เรย์ที่ต้องการเพียงขั้นตอนของการทำ metalisezation ในกระบวนการผลิตเท่านั้น ในการนำ PLD ไปใช้งานสามารถที่จะดำเนินการเพียงบันทึกบิตข้อมูลลงบนชิพเพื่อทำการหลอม Antifuse เท่านั้น ในรูปที่ 3.3 แสดงให้เห็นถึงสัญลักษณ์ของสถาปัตยกรรมภายในของ ASIC



รูปที่ 3.3 สัญลักษณ์ของสถาปัตยกรรมภายในของ ASIC

อุปกรณ์เกทอาร์เรย์ โครงสร้างของเกทอาร์เรย์ในยุคแรกจะคล้ายกับสถาปัตยกรรมแบบมีช่องสายสัญญาณเชื่อมต่อ (Channeled architecture) ที่มีลักษณะเป็นแถว และมีบางส่วนอยู่ในแนวคอลัมน์ของกลุ่มเกทที่ถูกกั้นด้วยช่องสายสัญญาณเชื่อมต่อ (Wiring channel) ที่มีขนาดความจุคงที่ ภายในชิพจะมีระดับของความหนาแน่นประมาณพันเกท แต่ก็พอเพียงต่อการนำไปใช้เป็นส่วนของวงจรควบคุมระบบ ซึ่งขึ้นอยู่กับความต้องการในแต่ละงานที่ต้องการนำไปใช้งาน เช่น ในส่วนของข้อมูลและหน่วยความจำ เป็นต้น นอกจากนี้เกทอาร์เรย์ยังมีวิวัฒนาการต่อไปจนกระทั่งอยู่ในรูปแบบที่ไม่มีช่องสายสัญญาณเชื่อมต่อ (Channelless) หรืออยู่ในรูปแบบ "Sea-of-gate" และมีโครงสร้างของสารกึ่งตัวนำเป็นแบบ Multilevel metalization ด้วยโครงสร้างเช่นนี้จึงทำให้มีความเป็นอิสระสูงในการที่จะเชื่อมต่อวงจรกันของเกทด้วยตัวนำแบบที่มีสายเชื่อมต่อหลายชั้น (Multiple level wire) ที่อยู่ในชิพได้เป็นอย่างดี

อุปกรณ์ Cell-based ICs การออกแบบโดยใช้ Cell-based จะมีการเตรียมไลบรารีของ Proven cell ไว้ให้ผู้ใช้ โดยทั่วไปแล้วมักจะกำหนดให้มีความสูงคงที่แต่จะสามารถปรับความกว้างได้ขึ้นอยู่กับรูปแบบของฟังก์ชันที่จะใช้งาน โดยในช่วงปี 1980 ได้มีการพัฒนากลุ่มเซลล์มาตรฐาน (Standard cell families) ที่มีฟังก์ชันซับซ้อนมาก และเทคโนโลยีทางด้านซิลิคอนที่ดีขึ้นทำให้เพิ่มความสามารถของการออกแบบกลุ่มของฟังก์ชันได้ดีขึ้น แต่กระบวนการสร้างชิพ (Fabrication) ก็

จะต้องทำทุกขั้นตอนให้เสร็จอย่างสมบูรณ์ทั้งการทำ mask และ process อุปกรณ์ชนิดนี้จะเป็น ASIC ที่มีความหนาแน่นสูงมากและมีสมรรถนะสูงที่สุด

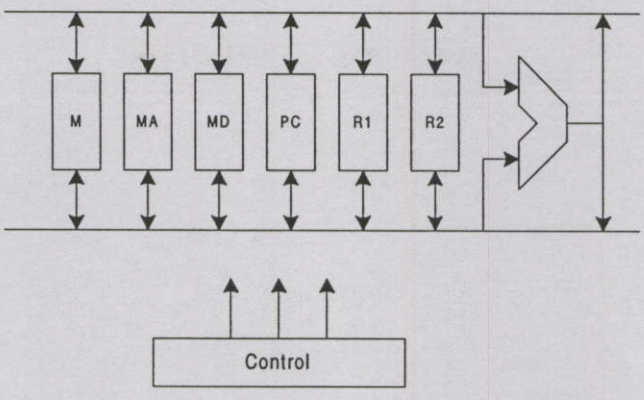
อุปกรณ์ Programmable logic device การถือกำเนิดขึ้นของ PLD สามารถถือได้ว่าเกิดจากอุปกรณ์ประเภท Monolithic memory และทำให้เกิดการสร้าง PAL ขึ้นเป็นครั้งแรกในปี 1978 ซึ่งมันมีลักษณะคล้ายกับ FPLA ในยุคแรกก็ยังไม่ประสบผลสำเร็จนักในรูปแบบของอุปกรณ์แบบ Two-level product จนกระทั่งถึงปี 1980 แนวคิดพื้นฐานของ PLD ได้ถูกพัฒนาขึ้นเป็น Complex PLD ที่อยู่ในรูปแบบของ FPGA และโครงสร้างแบบ PAL

3.4.4 ยุคของ Programmable Logic

การเกิดขึ้นของอุปกรณ์ประเภท Complex programmable ดังเช่น FPGA ทำให้สามารถที่จะคาดการณ์ได้ว่าระบบที่ใช้ไมโครโปรเซสเซอร์สามารถที่จะถูกแทนได้ด้วย SRAM-programmable FPGA แม้กระทั่งเครื่องคำนวณแบบ Von Neumann engine ที่สนับสนุนการคำนวณแบบใช้งานได้เอนกประสงค์ก็สามารถที่จะดำเนินการได้เป็นอย่างดีบน FPGA

3.5 รูปแบบการออกแบบ (Design styles)

จากรูปที่ 3.2 เมื่อนำงานที่ออกแบบได้ไปสร้างเป็นชิ้นงานในทางปฏิบัติก็ยังมีสิ่งที่ยุ่งยาก คือการสร้างวงจรของอุปกรณ์ที่มีลักษณะเป็นแบบนามธรรม (Abstract components) ที่จะต้องสร้างขึ้นตามรายละเอียด (Specification) ที่กำหนด เนื่องจากเป็นการทำงานเกี่ยวกับระบบดิจิตอล ดังนั้นข้อมูลอาจอยู่ในรูปของอัลกอริทึมที่มีรูปแบบเป็นทางการหรือไม่เป็นทางการก็ได้ การแปลผล (Interpretation) ของอัลกอริทึมจึงมีผลกระทบต่อการใช้ประโยชน์บางอย่างจากข้อมูลของการออกแบบอย่างหลีกเลี่ยงไม่ได้ อัลกอริทึมหนึ่ง ๆ สามารถเป็นได้ทั้งกลุ่มของกฎ (Set of rules) หรือกลุ่มของขั้นตอนการทำงานเพื่อทำการแปลงข้อมูล (Transforming data) และการดำเนินการคำนวณ (Executing a computation) โดยปกติแล้วตัวอัลกอริทึมจะมีลักษณะเป็น Static description ที่จำเป็นต้องใช้เครื่องมือ (Machine) ในการแปลผลเพื่อแสดงการทำงาน วงจรของอุปกรณ์ที่มีลักษณะเป็นแบบนามธรรมก็เป็นเครื่องมือที่แสดงถึงการทำงานตามอัลกอริทึม ดังนั้นงานที่หนักของการออกแบบจึงอยู่ที่การเลือกกลุ่มอุปกรณ์ที่ต้องสัมพันธ์กันทั้งฮาร์ดแวร์และซอฟท์แวร์ในการเชื่อมต่ออุปกรณ์เหล่านี้เข้าด้วยกัน กระบวนการนี้จะเรียกว่า สถาปัตยกรรมระบบ (System architecture) ซึ่งจะมีรูปแบบที่เกิดขึ้น 2 แนวทาง คือ ตัวแบบแผนทางซอฟท์แวร์ (Software paradigm) ที่อาศัยตัวแบบการคำนวณของ Von Neumann และตัวแบบแผนทางฮาร์ดแวร์ (Hardware paradigm) สถาปัตยกรรมที่มีการใช้ FPGA จะมีลักษณะเป็นตัวแบบแผนประเภทใดประเภทหนึ่งหรือถูกสร้างจากอุปกรณ์ที่มีลักษณะเป็นตัวแบบแผนประเภทใดประเภทหนึ่งก็ได้ เมื่อสร้างโดยตัวแบบแผนทางซอฟท์แวร์ อัลกอริทึมจะถูกแปลโดยตัวประมวลผล (Processor) แล้วแมพเป็นรหัส



รูปที่ 3.4 เส้นทางข้อมูล (Datapath) ของ Von Neumann engine

ในรูปที่ 3.4 แสดงต้นแบบของเครื่องคำนวณแบบ Von Neumann engine ที่ประกอบไปด้วยหน่วยความจำ (Memory : M) รีจิสเตอร์ต่าง ๆ (Memory address , Memory data , Program counter) และ ALU เมื่อสร้างโดยตัวแบบแผนทางฮาร์ดแวร์อัลกอริทึมจะถูกแมพลงบน Storage unit และ Function unit ด้วยลักษณะการคำนวณที่เหมือนกัน นอกจากนี้ระดับที่สูงที่สุดของมโนภาพของแบบงาน (Design abstraction) อาจจะทำให้ส่งผลดีต่อความสามารถของการออกแบบหรือลดช่องว่างในการออกแบบกับงานที่ออกแบบได้จริง โดยเฉพาะงานที่ต้องการความรีบด่วนมากกว่าปกติ กระบวนการออกแบบที่ดีนี้ก็จะช่วยทำให้ผู้ใช้ต้องปรับปรุงตรวจสอบงานเพียงเล็กน้อยเท่านั้น เพื่อให้งานเป็นไปตามกฎการออกแบบ เช่น การเลือกสถาปัตยกรรมแบบ Pipelined จะทำให้ความสามารถของระบบมีสภาพที่ดีขึ้นถึง 100% แทนการเลือกเทคโนโลยีที่สร้างวงจร (Circuit technology) ซึ่งจะทำให้ระบบมีสภาพที่ดีขึ้นเพียงประมาณ 10% เท่านั้น

เพื่อแสดงแนวทางของสถาปัตยกรรมทั้งสองแบบ โดยจะพิจารณาการออกแบบอย่างง่ายทั้งสองแบบของอัลกอริทึมการหาผลคูณของเวคเตอร์ที่มีสมการดังนี้

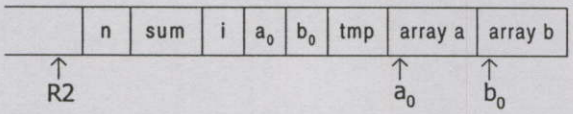
$$\sum_{i=0}^{n-1} a_i * b_i$$

3.5.1 ตัวแบบแผนทางซอฟท์แวร์ (Softeware Paradigm)

การออกแบบภายใต้ตัวแบบแผนทางซอฟท์แวร์ จะพบว่างานของการออกแบบคือการเขียนกลุ่มคำสั่ง (Program) สำหรับอัลกอริทึม ดังนี้

```
sum = 0
for i=0 to n-1 do
    sum=sum+a[i]*b[i]
```

จากนั้นคอมไพล์ด้วยเครื่องคำนวณแบบ Von Neumann และใช้อุปกรณ์ในรูปที่ 3.4 ด้วยการสมมุติว่าเป็นโครงสร้างอย่างง่ายแทนตำแหน่งลงบนแผนผังการทำงาน โดยรีจิสเตอร์ R1 ให้ทำหน้าที่เป็น accumulator และให้ R2 เป็นรีจิสเตอร์พื้นฐาน (base register) ดังนั้นการจัดหน่วยความจำจะแสดงได้เป็น



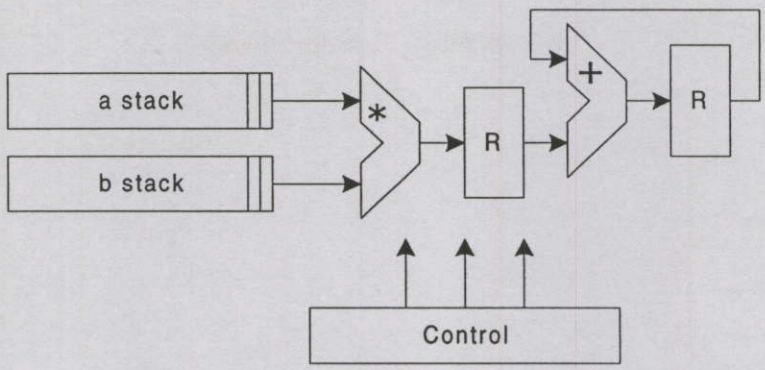
โปรแกรมที่จะทำการคำนวณฟังก์ชันแสดงดังรูปที่ 3.5

```
        CLR     R1
        STR     R1,R2,2      sum=0
        STR     R1,R2,3      i=0
loop    LOAD    R1,R2,4      R1=a0
        ADD     R1,R2,3      R1=a0+i
        LOAD    R1,R1,0      R1=a[i]
        STR     R1,R2,6      tmp=a[i]
        LOAD    R1,R2,5      R1=a0
        ADD     R1,R2,3      R1=b0+i
        LOAD    R1,R1,0      R1=b[i]
        MLT     R1,R2,6      R1=b[i]*tmp
        ADD     R1,R2,2      R1=sum+ditto
        STR     R1,R1,2
        LOAD    R1,R2,3      R1=i
        ADDIM   R1,1         R1=R1+1
        STR     R1,R2,3
        CMP     R1,R2,1      compare i:n
        BNE     loop         branch if i ≠ n
```

รูปที่ 3.5 ตัวอย่างโปรแกรมการคำนวณโดยใช้ Von Neumann Computer

3.5.2 ตัวแบบแผนทางฮาร์ดแวร์ (Hardware Paradigm)

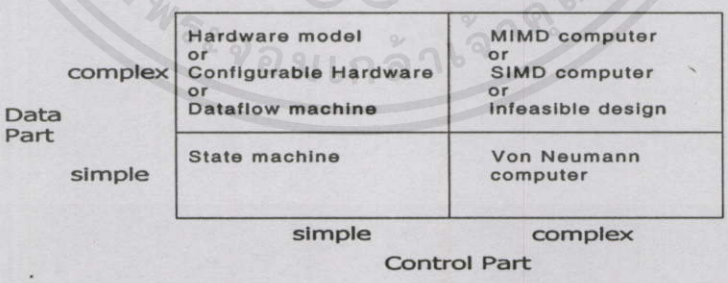
การออกแบบภายใต้ตัวแบบแผนทางฮาร์ดแวร์ ตัวแบบจะมีพื้นฐานอยู่บนการเชื่อมต่อภายในของฮาร์ดแวร์สแตคจำนวน 2 ชุด มี Unit function 2 หน่วย และรีจิสเตอร์ 2 ตัว ดังแสดงในรูปที่ 3.6 ข้อมูลจะแสดงในรูปของบิตอนุกรม (bit-serial) หรือบิตขนาน (bit-parallel) นอกจากนั้นฮาร์ดแวร์ สแตคจะมีขนาดที่จำกัดชัดเจน และสามารถถูกแบ่งแยกการทำงานออกจากกันได้อย่างอิสระ การออกแบบเช่นนี้สามารถสร้างได้จากชิ้นส่วนมาตรฐานที่ระดับ MSI



รูปที่ 3.6 หน่วยประมวลผลของ vector product

3.5.3 ตัวแบบทางฮาร์ดแวร์ที่สามารถคอนฟิกกูเรชันได้ (Configurable hardware paradigm)

จากตัวอย่างการออกแบบทั้งสองแนวทางจะเป็นลักษณะที่ง่ายของสถาปัตยกรรมแต่ก็สามารถทำให้เห็นถึงความแตกต่างกันได้มากเมื่อพิจารณาเส้นทางของข้อมูล (Data path) และโครงสร้างการควบคุมของแต่ละสถาปัตยกรรม สำหรับตัวแบบทางซอฟท์แวร์จะมีหน่วยคำนวณอย่างง่ายบนหน่วยความจำพื้นฐานของเครื่อง รีจิสเตอร์ไฟล์ และ ALU ที่ถูกแสดงความซับซ้อนไว้ด้วยรหัสส่วนของการควบคุม (Compiled code) ได้อย่างชัดเจน ส่วนตัวแบบแผนทางฮาร์ดแวร์จะมีเส้นทางข้อมูลซึ่งซับซ้อนกว่าเนื่องจากประกอบไปด้วยชิ้นส่วนทั้งหมด 6 ชุดเชื่อมต่อกันด้วยโครงสร้างในรูปแบบการควบคุมที่เป็นไปตามลำดับ สำหรับความซับซ้อนของข้อมูลและโครงสร้างการควบคุมสามารถเลือกได้โดยดูจากรูปที่ 3.7



รูปที่ 3.7 การเลือกความซับซ้อนของข้อมูลและโครงสร้างการควบคุม

การออกแบบด้วยตัวแบบทางฮาร์ดแวร์จะมีลักษณะเป็นเครื่องที่ใช้งานเฉพาะทางมีการทำงานตามอัลกอริทึมด้วยความเร็วสูงและมีค่าใช้จ่ายในการสร้างชิ้นงานต้นแบบอย่างน้อยหนึ่งชิ้นที่ค่าหนึ่ง ในทางกลับกันตัวแบบแผนทางซอฟท์แวร์จะให้ คำตอบที่สมรรถนะต่ำกว่าเนื่องจากต้องเสียเวลาในการแปลโปรแกรมแต่ คำตอบที่ได้จะมีค่าใช้จ่ายถูกกว่าเพราะเครื่องคอมพิวเตอร์มีการผลิตเป็น

จำนวนมากจึงมีราคาถูกและนำมาใช้ได้อีก เมื่อมี การสร้าง FPGA ที่ใช้งานได้อย่างกว้างขวาง สามารถโปรแกรมซ้ำได้และนำมาใช้ได้อีก สามารถสร้างได้เป็นจำนวนมาก ดังนั้นรูปแบบของฮาร์ดแวร์ที่ทำคอนฟิกกูเรชันให้มีความสามารถในการคำนวณจึงถูกสนับสนุนมากขึ้น เนื่องจากการมีข้อได้เปรียบในเรื่องของสมรรถนะที่อัลกอริทึมสามารถถูกแมพไปยังฮาร์ดแวร์ได้โดยตรง ดังนั้นเมื่อผลิตเป็นปริมาณมากก็จะทำให้มีต้นทุนที่ ถูกได้เช่นกัน ตารางที่ 3.1 แสดงภาพรวมของการออกแบบสถาปัตยกรรมของตัวแบบทางแผนทางฮาร์ดแวร์และซอฟท์แวร์แต่ในสภาพการณ์จริงสถาปัตยกรรมที่ดีที่สุดจะมีองค์ประกอบได้จากทั้งสองตัวแบบ

ตารางที่ 3.1 เปรียบเทียบแนวทางต่างๆ ของการออกแบบ [6]

Feature	Software	Hardware	Configurable Hardware
Instruction interpretation	Integral number of cycles	None	None
Gate delay	Low	Low	High
Spatial demand	PCB	IC	PCB
Cost (1 off)	Commodity	High NRE	Commodity
Architural efficiency	Fixed	Tailored architecture	Tailored architectures
Excution efficiency	Fixed units	Tailored excutation units	Tailored architectures
Bit efficiency	Fixed	Variable word size	Variable word size
Data reprecentation	Fixed	Variable word size	Variable word size
Reusablity	Yes	No	Yes

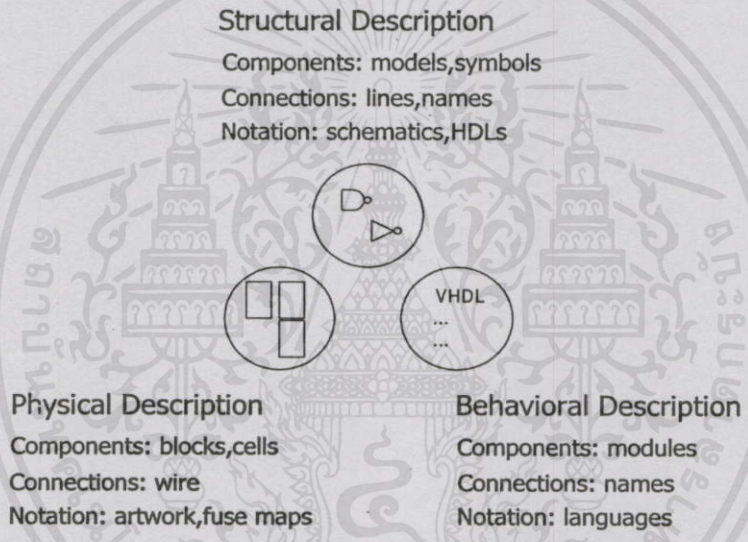
3.6 วิธีการออกแบบ (Design methodologies)

วิธีการออกแบบระบบด้วยการใช้อุปกรณ์ FPGA เพื่อให้ได้ชิ้นงานต้นแบบที่เหมาะสมจะต้องพิจารณาสิ่งสำคัญต่างๆที่มีอยู่หลายประการคือ

3.6.1 การอธิบายแบบ (Describing a design)

การอธิบายแบบมีอยู่หลายรูปแบบโดยเฉพาะเมื่อเทคโนโลยีมีความก้าวหน้ามากขึ้นและความซับซ้อนของแบบมีมากขึ้น แผ่นแบบของแผนภาพ (Diagramming templates) และบอร์ดภาพร่าง (Drawing boards) ได้ถูกแทนด้วยโปรแกรมการทำงานของระบบออกแบบที่เรียกว่า CAD ทำให้การอธิบายวงจรพื้นฐานที่เป็นอุปกรณ์ เช่น ทรานซิสเตอร์ รีซิสเตอร์ และคาปาซิเตอร์ อยู่ในรูปแบบของฐานข้อมูลที่ถูกรวบรวมไว้ในรูปของแผนภาพลำดับขั้น (Hierarchical schematics) นอกจากนี้ก็มีการใช้โปรแกรมภาษาที่อธิบายฮาร์ดแวร์ (Hardware description language program : HDL program) ใช้ตัวแบบจำลอง (Simulation model) การออกแบบแผนงานทางศิลป์ (Layout artwork) และแผนแบบทดสอบ (Test pattern) ต่างๆ ทำให้เห็นถึงความแตกต่างของการให้คำอธิบายแบบ (Design description) กับบันทึกหมายเหตุ (Notation) โดยความหมายของการให้คำอธิบายแบบที่ชัดเจนนั้นข้อมูลจะต้องถูกออกแบบหรือสร้างให้อยู่ใน 3 กลุ่มต่อไปนี้ คือ กลุ่มโครงสร้าง

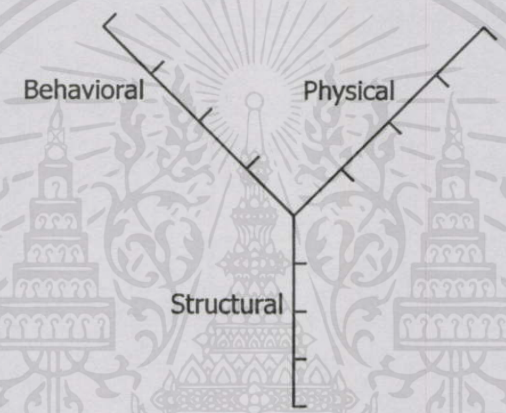
(Structural domain) กลุ่มพฤติกรรม (Behavioral domain) และกลุ่มกายภาพ (Physical domain) ดังรูปที่ 3.8 แสดงแผนภาพของเวนน์และลักษณะเฉพาะของส่วนประกอบภายในการอธิบายแบบแต่ละกลุ่ม การออกแบบที่อยู่ในระดับลอจิกการอธิบายโครงสร้างสามารถเขียนเป็นลอจิกไดอะแกรมได้ การอธิบายแบบในเชิงพฤติกรรม (Behavioral description) สามารถแสดงเป็นกลุ่มของสมการทางลอจิก (Logic equation) ได้ และการอธิบายแบบด้วยทางกายภาพ (Physical description) ก็จะเป็นกลุ่มของแผนภาพเซลล์มาตรฐาน (Standard cell layout) ของ ASIC แต่ละตัว (หรือ Fuse map ของ PAL) ส่วนบันทึกหมายเหตุนั้นมีจุดมุ่งหมายเพื่อใช้ในการบรรยายสิ่งเหล่านี้ โดยบันทึกหมายเหตุที่แตกต่างกันก็อาจจะใช้สร้างส่วนย่อย ๆ ที่แตกต่างกันในแต่ละส่วนของตัวแบบงาน



รูปที่ 3.8 กลุ่มต่าง ๆ ของการอธิบายแบบ

นอกจากนี้แบบงานที่มีความซับซ้อนจะถูกสร้างโดยการอธิบายแบบในรูปแบบต่าง ๆ จากหลายระดับ ในทางปฏิบัติแบบงานในเชิงวิศวกรรมมักเป็นการดำเนินงานแบบแบ่งส่วนงานให้แยกย่อยก่อนการเข้าไปแก้ปัญหา (Divide and conquer) ทำให้แบบแผนทางความคิด (Conceptual) การทำงานของระบบงานที่มีขนาดใหญ่ถูกแบ่งให้เป็นหน่วยที่เล็กลงจนกระทั่งเหมาะสมต่อการสร้างเป็นชิ้นงานจริงในระดับอุปกรณ์ทางกายภาพได้ ตัวอย่างของระดับแบบแผนความคิด (Conceptual levels) ของมโนภาพของแบบงานมักจะประกอบไปด้วย แผนผังวงจร อุปกรณ์ลอจิก อุปกรณ์ Register transfer และ Processor-Memory-Switch (PMS) ในระดับที่เป็นทางกายภาพก็จะประกอบไปด้วยอุปกรณ์สารกึ่งตัวนำ และแผ่นวงจรพิมพ์ เป็นต้น ดังนั้นการมีเพียงมิติเดียวของการอธิบายแบบดังรูปที่ 3.8 จึงมีลักษณะที่เข้าใจง่ายและสามารถถูกปรับเปลี่ยนหรือแก้ไขเพื่อให้มีการอธิบายแบบในระดับที่แตกต่างแยกย่อยไปได้เท่าที่ต้องการโดยไม่จำกัดจำนวนลำดับชั้นของ

การอธิบาย ด้วยเหตุนี้จึงเป็นวิธีที่ดีสำหรับการนึกภาพและดูการเลื่อนไหล (Flow) ของกระบวนการออกแบบ ในรูปที่ 3.9 แสดงถึงรูปกราฟความสัมพันธ์กันบน 3 แกน (Gajski-Kuhn diagram) โดยให้ Y แทนการอธิบายแบบของทั้งสามกลุ่ม แต่ละแกนจะแทนสเกลระดับของการออกแบบ ในแผนภาพนี้ขั้นตอนของกระบวนการออกแบบจะถูกแสดงในรูปของการผ่านจุด (transition) จากบนแกนหนึ่งไปยังจุดที่อยู่บนแกนอื่น เช่น การสร้างแผนภาพจากลอจิกไดอะแกรม ซึ่งมีลักษณะการอธิบายแบบในเชิงพฤติกรรม เป็นต้น ก็จะมีกระบวนการไหลของการออกแบบ (Design process flow) เป็นเหมือนกับการหมุนรอบแกนทั้งสาม (Spiral path) จากจุดเริ่มต้นของกราฟเป็นแบบเชิงพฤติกรรม แล้วพัฒนาต่อไปจนเป็นแบบโครงสร้างและเป็นแบบทางกายภาพในที่สุด นอกจากนี้การอธิบายแบบทั้ง 3 กลุ่มจะมีรูปแบบเฉพาะที่แตกต่างกัน คือ



รูปที่ 3.9 แผนภาพ Gajski-Kuhn diagram

Structural design เป็นแบบที่แสดงในเชิงโครงสร้างการเชื่อมต่อของชิ้นส่วนต่าง ๆ เข้าด้วยกันอย่างง่าย เช่น การใช้สายไฟเชื่อมต่ออุปกรณ์ที่อิเล็กทรอนิกส์เข้าไว้ด้วยกัน การใช้หมุดย้ำและขอเกี่ยวเพื่อยึดโครงของอุปกรณ์เข้าไว้ด้วยกัน เป็นต้น ซึ่งจะพบว่าตำแหน่งทางโครงสร้างถูกกำหนดโดยสถาปัตยกรรมที่ทำให้ระบบมีลักษณะเป็นไปตามที่ต้องการ เช่น สมรรถนะ ความสมบูรณ์ของโครงสร้าง (Structural integrity) และอื่น ๆ สำหรับแบบที่แสดงในเชิงโครงสร้างเพื่อใช้งานด้าน FPGA นั้นจะใช้โปรแกรมประเภท schematic drawing tool

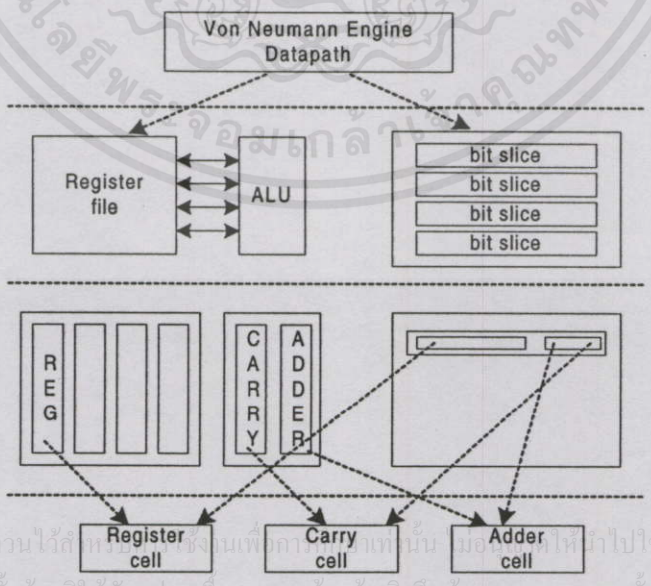
Physical design เป็นแบบที่แสดงในเชิงกายภาพที่จะต้องมีการรวมเอาการสร้างข้อมูลที่พอเพียงสำหรับการออกแบบเพื่อการผลิตรวมถึงการทดสอบไว้ด้วย แบบเชิงกายภาพสำหรับ FPGA จำเป็นจะต้องมีการกำหนดหรือนิยามฟังก์ชันสำหรับลอจิกบล็อกและส่วนของการเชื่อมต่อที่เชื่อมต่ออยู่ภายในด้วย เหมือนกับการสร้างแผนภาพไว้ในการทำ VLSI ส่วนข้อมูลสำหรับการผลิตชิ้นงานจะอยู่ในรูปของแฟ้มข้อมูลแบบไบนารีสำหรับการทำโปรแกรม เครื่องมือที่ใช้ออกแบบ

เชิงกายภาพจะต้องมีทั้ง Symbolic editors , Generator และซอฟท์แวร์สำหรับกำหนดเส้นทางและตำแหน่งแบบอัตโนมัติ (Automatic place-and-route software)

Behavioral design เป็นแบบที่แสดงในเชิงพฤติกรรม รายละเอียดของแบบชิ้นงานจะอยู่ในรูปของฟังก์ชันการทำงาน โดยอาจจะถูกรวมไว้ในตัวแบบจำลอง (Simulation model) ในโปรแกรม HDL ในกลุ่มของสมการลอจิกหรือในตารางความจริง (Truth table) ก็ได้ แม้กระทั่งการอธิบายด้วยลอจิกไดอะแกรมก็สามารถแสดงในแบบเชิงพฤติกรรมได้ด้วยการใช้แบบแผนการเขียนภาพวาด (Drawing) อย่างง่าย ๆ

3.6.2 การออกแบบที่มีรูปแบบเป็นลำดับขั้น (Hierarchical design)

การออกแบบที่มีรูปแบบเป็นลำดับขั้นหรือแบบเชิงโครงสร้างถูกค้นพบเมื่อความซับซ้อนในการออกแบบที่เป็น Full-custom มีมากขึ้นในด้านการผลิตส่งผลกระทบต่อวิศวกรสร้างซอฟท์แวร์ที่จะต้องปรับวิธีการเขียนโปรแกรมโดยใช้เทคนิคการแบ่งย่อยงานแล้วเข้าไปแก้ปัญหาเพื่อผลิตซอฟท์แวร์ให้มีความซับซ้อนในการสร้างมากขึ้นตามไปด้วย แนวความคิดเกี่ยวกับการแยกส่วน (Modularization) การซ่อนสาระสำคัญ (Information hiding) การตรวจสอบอย่างเป็นขั้นตอน (Stepwise refinement) จึงถูกนำมาใช้เพื่อการออกแบบ VLSI ทำให้มีการนำเทคนิคการเขียนโปรแกรมมาประยุกต์ใช้กับการทำเลย์เอาท์ของ VLSI เพราะว่าโครงสร้างการเขียนโปรแกรมยอมให้มีการแยกรายละเอียดต่าง ๆ ของอาร์คเวิร์กได้ โดยเฉพาะอย่างยิ่งเมื่อมีการทำซ้ำหรือการทำงานอย่างมีเงื่อนไข ในรูปที่ 3.10 จะเป็นการแสดงถึงการออกแบบที่มีลักษณะเป็นลำดับขั้น



รูปที่ 3.10 ลำดับขั้นการแบ่งส่วนของ datapath

จากรูปที่ 3.10 แสดงการแยกส่วนแบบมีลำดับขั้นของเส้นทางข้อมูลที่ถูกกำหนดขึ้นให้อยู่ในรูปของเครื่องคำนวณแบบ Von Neumann โดยแบบงานสามารถถูกแจกแจงให้เห็นรายละเอียดด้วยกลุ่มของแบบแผนวงจรที่ระดับต่ำสุดได้เป็นวงจรหรือลอจิกไดอะแกรมของเซลล์ต่าง ๆ ทั้งการแบ่งส่วนด้วยรายละเอียดจริงของแบบงานและความแตกต่างของแต่ละส่วนด้วยทางเลือกที่อยู่ในแผนผังการแบ่งส่วน ดังส่วนที่อยู่ในแนวแกนนอนก็จะเป็นการให้ความสำคัญกับการออกแบบของ Bit slice หรือในแนวแกนตั้งก็จะเป็นการให้ความสำคัญกับการออกแบบของ Functional unit จากรูปจะเป็นแบบแผนการแบ่งส่วนที่ทางเลือกจะเป็นการแสดงด้วยแผนผังการเชื่อมต่อของสายในตัวอย่างนี้เป็นบัสที่ใช้ในการสื่อสารระหว่าง Operands และส่งผลลัพธ์ที่ได้ระหว่างรีจิสเตอร์ไฟล์กับ ALU การออกแบบที่ถูกต้องของแบบที่เป็นลำดับขั้นจะต้องแบ่งชิ้นส่วนของกลุ่มสายเหล่านี้ไปยังเซลล์ที่อยู่ด้านใต้ของเส้นแบ่งชั้นเสมอ เพราะจะแสดงให้เห็นถึงระดับความซับซ้อนที่แตกต่างกันในการเชื่อมต่อข้อมูลเพื่อนำไปจำลองการทำงาน

3.6.3 การออกแบบที่มีความเป็นอิสระจากเทคโนโลยี (Technology-independent design)

การออกแบบที่มีความเป็นอิสระจากเทคโนโลยีจะใช้ได้เฉพาะกับแบบงานที่เป็นการออกแบบขั้นสูงเท่านั้นโดยใช้โปรแกรมภาษา HDL (Hardware Description Language) และจะแปลงรูปแบบของข้อมูลอย่างอัตโนมัติด้วยโปรแกรมสังเคราะห์เพื่อให้อยู่ในรูปของตัวกลาง (Intermediate form) ที่สามารถนำไปกำหนดอุปกรณ์เป้าหมายให้มีรูปแบบในการสร้าง (Implementation styles) ที่แตกต่างกันหรือผู้ผลิตที่แตกต่างกันได้ จึงมีลักษณะคล้ายกับเทคโนโลยีตัวแปลภาษาระดับสูง (High level language compiler technology) ที่เป็นตัวสร้างรหัสการทำงานจากข้อมูลในรูปแบบตัวกลางไปสร้างเป็นรหัสสำหรับเครื่องจักร (Machine) ที่ต้องการ สิ่งที่จะกำหนดว่าเป็นการออกแบบที่มีความเป็นอิสระจากเทคโนโลยีก็คือ มันสามารถรวมเอาสิ่งที่ผู้ใช้ได้ออกแบบขึ้นเองด้วยการทำงานในภาษาระดับสูงเข้าไปเป็นฐานข้อมูลในระบบได้ด้วย แต่ข้อด้อยของการออกแบบที่มีความเป็นอิสระจากเทคโนโลยีก็คือ มีการสูญเสียสูงในเรื่องของประสิทธิภาพการแมพจากแบบซึ่งเป็นการออกแบบขั้นสูงให้ไปเป็นชิ้นงานจริง และมีการสูญเสียจากการที่ผู้ออกแบบสามารถควบคุมค่าของเวลาที่ใช้ในการทำงานและขนาดของเกทในแบบงานได้โดยตรง ซึ่งเหตุผลทั้งคู่จะเป็นกุญแจที่นำไปสู่ความสำเร็จหรือไม่สำเร็จในการสร้างชิ้นงานจริงได้

3.6.4 การออกแบบด้วยวิธี Mead-Conway

การออกแบบด้วยวิธี Mead-Conway จะนิยมใช้ในการออกแบบวงจร VLSI มีลักษณะเป็นการออกแบบที่เป็นลำดับขั้นที่มีการมุ่งเน้นในเรื่องของการจัดการเกี่ยวกับความซับซ้อนในการสื่อสารภายในหรือการเชื่อมต่อของสายตัวนำในชิ้นงาน ซึ่งเป็นเรื่องที่สำคัญมากเพราะว่าต้นทุนของเวลาที่จะใช้ในการเคลื่อนย้ายข้อมูลไปรอบ ๆ วงจรของ VLSI อาจจะมีผลมากต่อการทำงาน

ทั้งหมดของมัน วัตุประสงค์ของเทคนิคการออกแบบคือเพื่อเพิ่มประสิทธิภาพของงานที่ออกแบบ ซึ่งจะประกอบไปด้วยหลักเกณฑ์ดังนี้

1) การเชื่อมต่อของสายจะเป็นกุญแจสำคัญในการออกแบบในการออกแบบขั้นสูง

2) ใช้กฎพื้นฐานในการออกแบบในการทำให้ส่วนที่เป็นนามธรรมหมดไปจากรายละเอียด

3) ใช้ Building block เป็นส่วนของทางเดินข้อมูลและ PLA (Two-level logic block) เนื่องจากมันมีโครงสร้างของการเชื่อมต่อของตัวนำที่ดี

วิธีการนี้จะจัดวางตำแหน่งของชิ้นส่วนต่าง ๆ ภายในวงจร VLSI โดยคำนึงถึงโครงสร้างและรูปแบบทางกายภาพเป็นหลัก การพยายามรักษารายละเอียดรูปแบบของโครงสร้างและรูปแบบทางกายภาพให้เหมือนเดิมที่สุดจะช่วยหลีกเลี่ยงการเกิดข้อบกพร่อง (bug) ที่อาจจะเป็นกับดักที่ซับซ้อนในการออกแบบหากจำเป็นต้องมีการแยกโครงสร้างกับลำดับขั้นทางกายภาพออกจากกัน

เทคนิคนี้สามารถนำมาใช้ได้กับ FPGA ได้เป็นอย่างดีภายใต้เงื่อนไขคือ FPGA จะต้องมีความเหมือนกันอย่างแท้จริงกับตัวสื่อกลางหรือโปรแกรมที่ใช้สำหรับการออกแบบและสร้างวงจร อย่างในกรณีนี้จะเห็นได้ว่าสถาปัตยกรรมของ FPGA แบบ Fine-grain จะมีความคล้ายกันกับลอจิกอาร์เรย์ที่โปรแกรมได้ เนื่องจากชิ้นส่วนในระดับเกทอาร์เรย์ของมันเป็นแบบเซลลูล่าร์อาเรย์มีความเหมาะสมกับอุปกรณ์แวดล้อมมากกว่าสถาปัตยกรรมแบบ Coarse-grain channeled ซึ่งผู้จัดรูปแบบนั้นสามารถที่จะคาดหวังได้จากการเตรียมทางเลือกต่างๆในการสร้างได้โดยตรง

3.6.5 การออกแบบเพื่อให้ได้ความเร็วของการทำงานที่เหมาะสม (Temporal design)

เนื่องจากการใช้ FPGA หรือสื่อกลางที่ช่วยในการสร้างชิ้นงานใด ๆ ก็ตามยังคงต้องมีการตัดสินใจในเรื่องเวลาการทำงานของตัวชิ้นงานเสมอ โดยเป็นการพิจารณาเลือกระหว่างขนาดของพื้นที่ใช้งานภายในชิพที่จะทำให้มีเวลาในการทำงานได้ดีที่สุดกับสมรรถนะของชิ้นงาน ดังนั้นการเตรียมหรือเลือกลักษณะเฉพาะของ Storage element ในส่วนที่เป็นเซลล์พื้นฐานของFPGA อาจจะเป็นสิ่งที่จำกัดขอบเขตทางเลือกต่าง ๆ ที่จะใช้ในการออกแบบซึ่งจะมีอยู่ 2 ลักษณะคือ

1) ระบบสัญญาณนาฬิกา (Clocked)

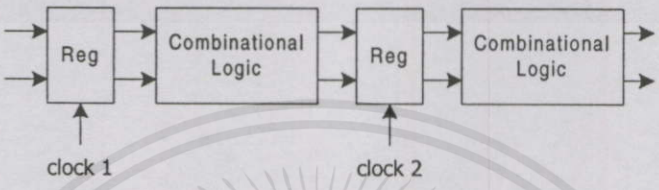
2) ระบบการกำหนดเวลา Self-timed

ดังมีรายละเอียดต่อไปนี้

3.6.5.1 ระบบสัญญาณนาฬิกา (Clocked)

สัญญาณนาฬิกาจะเป็นตัวกำหนดกฎเกณฑ์จังหวะการทำงานของทุก ๆ เซลล์ให้ทำงานแบบซิงโครนัสกับสัญญาณนาฬิกาของระบบ ตัวอย่างของวงจรในรูปที่ 3.11 มีการทำงานเป็นแบบลำดับของวงจรคำนวณที่เป็นแบบส่งผ่านตัวรีจิสเตอร์ ในวงจรจะพบว่าช่วงเวลาของสัญญาณนาฬิกาเป็นสิ่งที่กำหนดการไหลของข้อมูลที่ผ่านระบบ ดังนั้นผลลัพธ์จากการคำนวณที่เสร็จสิ้นใน

ช่วงหนึ่ง ๆ ด้วยวงจรที่เป็นคอมบิเนชั่นลอจิกจะถูกเก็บไว้ในรีจิสเตอร์ที่อยู่ตรงกลางก่อนที่จะส่งต่อไปให้ช่วงถัดไป ปัญหาสามารถที่จะเกิดขึ้นได้เมื่อมีการหน่วงเวลาบนสายสัญญาณที่อยู่ในระบบคือทำให้เกิดการเบี่ยงเบน (Skew) ของสัญญาณนาฬิกา ส่งผลให้เกิดการเลื่อนเวลาในการทำงานของรีจิสเตอร์ ดังนั้นใน FPGA บางกลุ่มจึงได้มีการเตรียมระบบที่เป็นโครงข่ายพิเศษที่ครอบคลุมทั้งชิพไว้โดยเฉพาะเพื่อให้เกิดค่าของเวลาหน่วงน้อยที่สุดส่งผลให้การทำงานของระบบเป็นแบบซิงโครนัสมากที่สุดเท่าที่จะเป็นไปได้



รูปที่ 3.11 รูปแบบของ Register transfer model

ถ้าในกรณีที่เซลล์ของ FPGA ถูกใช้งานเป็นแบบ Smart-memory ที่อยู่ภายในระบบคอมพิวเตอร์ก็จะทำให้แบบแผนผังของระบบสัญญาณนาฬิกา (Clocked scheme) อาจจะเป็นเทคนิคที่ถูกเลือกมาใช้ได้ เพราะว่ามันจะยอมให้ไมโครโปรเซสเซอร์ทำการอ่านหรือเขียนบนจุดใด ๆ ของวงจรที่สร้างขึ้นไว้ภายในได้โดยที่เซลล์ไม่ไปรบกวนการทำการคำนวณของระบบ เพราะเนื่องจากมันมีโครงสร้างเป็นแบบเซลล์อาร์เรย์จึงเป็นที่น่าสังเกตว่าการที่มีระบบสัญญาณนาฬิกาที่สามารถเลือกได้จะทำให้ผู้ใช้งานไม่ต้องกังวลเรื่องการกำจัดสัญญาณรบกวนอันเนื่องมาจากการมีเวลาเหลื่อมกันของสัญญาณ (Timing hazard) และแนวทางการใช้สัญญาณนาฬิกาเพื่อให้วงจรสามารถทำงานได้อย่างถูกต้องมี 2 แนวทางคือ

1) การใช้ Single-phase clocking คือการใช้อุปกรณ์ที่มีการกระตุ้นการทำงานแบบ Master-slave storage element และมีสัญญาณนาฬิกาแบบ Single-phase ดังในรูปที่ 3.11 ที่สัญญาณนาฬิกาทั้งสองตัวต้องเป็นสัญญาณเดียวกัน รูปแบบเช่นนี้ได้มีการกำหนดเป็นกฎเกณฑ์ที่ใช้ในระบบ TTL แต่อย่างไรก็ตามอุปกรณ์ FPGA ก็สนับสนุนรูปแบบนี้ด้วยเช่นกัน

2) การใช้ Two-phase clocking เป็นการใช้สัญญาณนาฬิกาแบบ Two-phase ที่ไม่มีการเหลื่อมทับกัน (Overlap) ให้กับตัวแลชท์ การทำเช่นนี้ก็จะทำให้นำไปประยุกต์ใช้ในวงจรในรูปที่ 3.11 ได้ และการที่ไม่มีความเหลื่อมกันของสัญญาณนาฬิกาทั้งสองจะช่วยป้องกันการเคลื่อนย้ายข้อมูลมากกว่าหนึ่งช่วงในแต่ละวงรอบสัญญาณนาฬิกา ดังนั้นถ้าใน FPGA มีการเตรียมแลชท์ที่สามารถควบคุมได้โดยตรงจากสัญญาณนาฬิกาของระบบก็จะทำให้สามารถหลีกเลี่ยงการใช้สายสัญญาณพิเศษเพิ่มขึ้นมาสำหรับใช้ในการป้อนสัญญาณนาฬิกาได้

3.6.5.2 ระบบการกำหนดเวลา Self-timed

ข้อกำหนดเรื่องเวลานี้หากในแต่ละเซลล์สร้างสัญญาณที่ชัดเจนของ “go” และ “done” ขึ้นมาโดยที่มีเส้นทางที่เป็นแบบขนานไปกับสัญญาณที่เป็นข้อมูลและมีการใช้การเปลี่ยนขอบ (Edge transition) ของตัวสัญญาณมาเป็นการทำ Handshake ซึ่งจะพบว่าเป็นแนวทางที่น่าสนใจมากเพราะว่ามันจะช่วยให้ลดปัญหาได้มากในเรื่องของเวลาในขณะที่มีความถูกต้องของการทำงานสูงมาก อย่างไรก็ตามการสร้าง self-time FPGA มีต้นทุนที่สูงมากในเรื่องของขนาดพื้นที่ของเซลล์ เนื่องจากความซับซ้อนของเซลล์จะมีมากกว่าการสร้างด้วยเทคโนโลยีแบบซิงโครนัสในส่วนที่เป็นสายสัญญาณ “go” และ “done” และต้องมีจำนวนของเกทที่มากพอใน การใช้งานหากจะต้องมีการแบ่งหรือแยกสายสัญญาณที่ตัวมัลติเพล็กเซอร์ ที่สำคัญคือฟังก์ชันการทำงานของมันจะต้องถูกคำนวณไว้ก่อนเพื่อเตรียมไว้สำหรับสิ่งที่จำเป็นต่อการควบคุมด้วย

3.7 โครงสร้างของ FPGA ตระกูล Virtex-E Family

3.7.1 สถาปัตยกรรมของ FPGA ในตระกูล Virtex-E

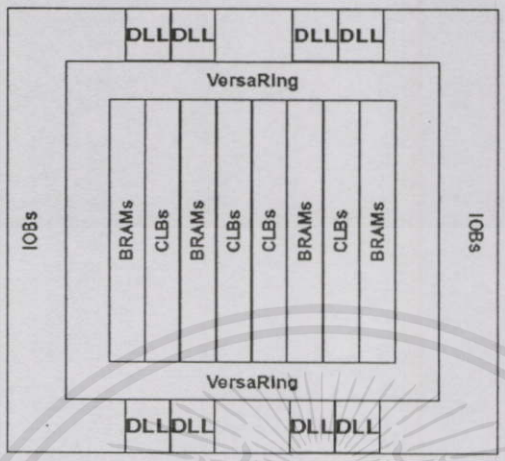
อุปกรณ์ FPGA ตระกูล Virtex-E ของบริษัท Xilinx ได้ถูกออกแบบให้มีความยืดหยุ่นสูง โดยการให้มีโครงสร้างพื้นฐานเป็นแบบ RAM-based ที่ประกอบไปด้วยกลุ่มของลอจิกที่สามารถทำการโปรแกรมได้ (Configuration Logic Blocks , CLB) ที่ถูกล้อมรอบด้วยกลุ่มของอินพุตและเอาท์พุต (Input/output Blocks , IOB) ที่มีการเชื่อมต่อภายในด้วยความเร็วที่สูงมากและสามารถใช้ประโยชน์ได้เอนกประสงค์ ส่วนในการใช้งานสามารถที่จะออกแบบให้มีการทำงานเป็นแบบซิงโครนัสที่มีค่าความถี่ของสัญญาณนาฬิกาสูงได้ถึง 240 MHz ในตารางที่ 3.2 จะเป็นการแสดงถึงสมรรถนะการทำงานของวงจรพื้นฐาน (Common Circuit Function) ที่มีอยู่ใน FPGA ตระกูลนี้ และค่าของเวลาที่แสดงในตารางจะเป็นค่าพารามิเตอร์ที่อยู่ในกรณีต่ำที่สุดที่จะยอมรับได้

3.7.1.1 โครงสร้างของ Virtex-E Array

โครงสร้างของ Virtex-E จะมีลักษณะเป็นเกท์อาร์เรย์ที่ผู้ใช้สามารถโปรแกรมได้ แสดงดังรูปที่ 3.12 ซึ่งจะประกอบไปด้วยส่วนที่สามารถกำหนดได้โดยผู้ใช้อยู่สองส่วนหลัก ๆ คือ CLB จะเป็นส่วนที่สำหรับใช้ในการทำโครงสร้างทางลอจิก และ IOB จะเป็นส่วนที่สำหรับใช้ในการทำการเชื่อมต่อระหว่างขาของชิพและ CLB ต่าง ๆ

ตัว CLB จะมีการเชื่อมต่อกันภายในผ่านโครงข่ายของการเชื่อมต่อทั่วไป (general routing matrix , GRM) ใน GRM จะประกอบไปด้วยอาร์เรย์ของสวิตช์เชื่อมต่อ (routing switch) ที่มีตำแหน่งอยู่ระหว่างการเชื่อมต่อภายในของช่องทางการเชื่อมต่อ (routing channel) ในทางแนวตั้ง

กับแนวนอน ในแต่ละ CLB จะวางซ้อนกันเป็นชั้น ๆ ที่เรียกว่า Versa Block ซึ่งก็จะเป็นการทำให้มีการเชื่อมต่อกันได้ระหว่าง CLB กับ GRM



รูปที่ 3.12 สถาปัตยกรรมของ FPGA รุ่น Virtex-E

ตารางที่ 3.2 สมรรถนะของวงจรพื้นฐานใน Virtex-E

Function	Bits	Virtex-E (-7)
Register-to-Register	-	-
Adder	16	4.3 ns
	64	6.3 ns
Pipelined Multiplier	8x8	4.4 ns
	16x16	5.1 ns
Address Decoder	16	3.8 ns
	64	5.5 ns
16:1 Multiplexer	-	6.1 ns
Parity Tree	9	3.5 ns
	18	4.3 ns
	36	5.9 ns
Chip-to-Chip	-	-

ในส่วนของ Versa Ring ที่เชื่อมต่อกับ I/O จะเป็นส่วนที่เพิ่มการติดต่อระหว่างภายในตัวชิพกับสิ่งแวดล้อมภายนอก ซึ่งการเชื่อมต่อนี้จะเพิ่มความยืดหยุ่นของการจัดตำแหน่งขาของอุปกรณ์ได้ นอกจากนั้นได้มีการเพิ่มวงจรดังต่อไปนี้เข้ากับส่วนของ GRM ด้วยคือ

1) บล็อกของหน่วยความจำที่มีขนาด 4096 บิต ในแต่ละบล็อก

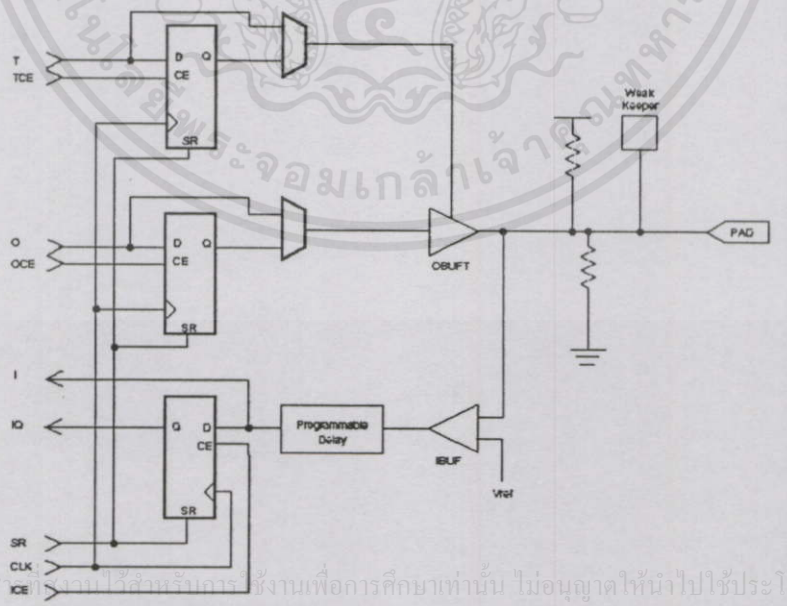
2) ตัว Clock DLL ซึ่งจะใช้สำหรับการจัดแบ่งกระจายการชดเชยค่าเวลาหน่วงของสัญญาณนาฬิกา

3) บัฟเฟอร์แบบ 3-state (BUFT) ซึ่งจะช่วยในการเป็นวงจรขับในแต่ละ CLB ที่มีการเชื่อมต่อกันอยู่ในแนวตั้งของภายในชิพ

ค่าของลอจิกที่ถูกเก็บไว้ในเซลล์หน่วยความจำจะเป็นตัวควบคุมสถานะการทำงานของส่วนประกอบทางลอจิก และการเชื่อมต่อภายในของส่วนประกอบภายในของตัวชิพ ซึ่งค่าเหล่านี้จะถูกโหลดเข้าสู่เซลล์หน่วยความจำในตอนจ่ายกำลังงานในตัวชิพ และสามารถที่จะทำการโหลดซ้ำได้ตามความจำเป็น เมื่อต้องการเปลี่ยนฟังก์ชันการทำงานของตัวอุปกรณ์

3.7.1.2 กลุ่มของขา IOB

ในรูปที่ 3.13 จะแสดงถึงกลุ่ม IOB ของชิพ Virtex-E การมีรูปแบบที่สามารถให้มีการเลือกเป็นอินพุตและเอาท์พุตได้ทำให้มันสามารถที่จะถูกใช้งานได้ในหลายมาตรฐานครอบคลุมแรงดันตั้งแต่ 0.7 V. (มาตรฐาน HSTL I) ไปจนถึง 3.3 V. (มาตรฐาน LVTTL) นอกจากนั้นจะพบว่ามีฟังก์ชันการทำงานอยู่ 3 รูปแบบที่ IOB สามารถกำหนดได้ในแต่ละรูปแบบการทำงานของแลทช์ในแบบที่ใช้ขอบสัญญาณกระตุ้น แบบ D-type ฟลิปฟลอป หรือแบบใช้ระดับสัญญาณกระตุ้นที่มีอิสระจากกัน รวมทั้งยังมีความอิสระจากกันในเรื่องของทิศทางของการไหลของข้อมูลอีกด้วย องค์ประกอบที่น่าสนใจอีกส่วนหนึ่งก็คือผู้ใช้สามารถการทำ pull-up และ pull-down หรือที่ขาภายในชิพได้โดยตรง จากรูปที่ 3.13 สามารถพิจารณาแยกส่วนของอินพุตและเอาท์พุต ได้คือ



รูปที่ 3.13 กลุ่มของ IOB ของ Virtex-E

3.7.1.3 ขาสัญญาณเมื่อเป็นอินพุต

ในส่วนอินพุตของ IOB การเชื่อมต่อของสัญญาณอินพุตทำได้ทั้งในแบบที่เชื่อมต่อได้โดยตรงและแบบที่ต้องผ่านฟลิปฟลอป และในส่วนขององค์ประกอบเวลาหน่วง (Delay element) ที่ขา D ของฟลิปฟลอปนี้จะคอยกำจัดค่าของเวลาหน่วงที่เกิดจากการเชื่อมต่อขาของอุปกรณ์ (Pad-to-pad hold time) ค่าของเวลาหน่วงนี้จะถูกทำให้เหมาะสมกับค่าของเวลาหน่วงภายในของสัญญาณนาฬิกาใน FPGA ถ้ามีการนำไปใช้งาน ซึ่งก็จะทำให้มั่นใจได้ว่าค่าของเวลาหน่วงนี้จะเป็นศูนย์ นอกจากนั้นในแต่ละขาของอินพุตบัฟเฟอร์สามารถที่จะถูกกำหนดให้มีระดับของแรงดันเป็นไปตามมาตรฐานได้ ซึ่งในบางมาตรฐานก็ยังเปิดโอกาสให้ผู้ใช้งานสามารถที่จะป้อนแรงดันระดับเริ่มเปลี่ยน (Threshold voltage) ให้แก่ชิพ เพื่อเป็นแรงดันอ้างอิง V_{ref} ในกรณีของ R-pull-up และ R-pull-down ของแต่ละอินพุตของชิพ เมื่อถูกกำหนดให้ใช้งานหลังจากการทำคอนฟิกกูเรชันแล้ว ค่าของมันจะอยู่ในช่วง 50-100 kΩ

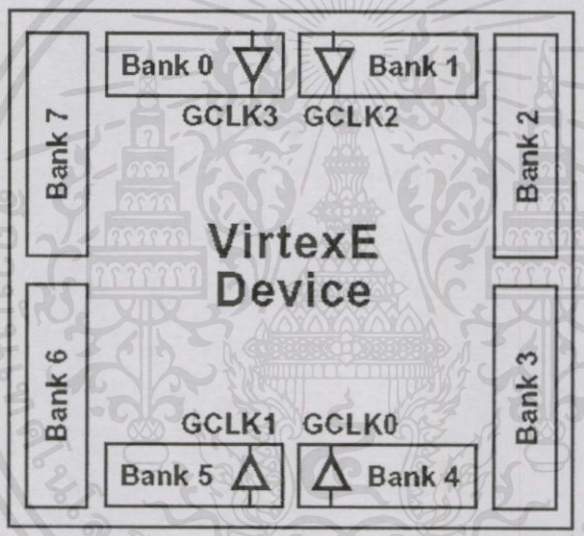
3.7.1.4 ขาสัญญาณเมื่อเป็นเอาท์พุต

วงจรในส่วนของเอาท์พุตจะประกอบไปด้วยบัฟเฟอร์แบบ 3-state ที่กำหนดให้ขับวงจรในแต่ละขาของชิพ ขาเอาท์พุตนี้สามารถที่จะกำหนดให้ทำงานได้ทั้งในแบบการต่อโดยตรงกับวงจรลอจิกภายในชิพ หรือจะต่อผ่าน IOB เอาท์พุตฟลิปฟลอปก็ได้ ส่วนขาควบคุม 3-state ของเอาท์พุตสามารถที่จะเชื่อมต่อได้โดยตรงจากลอจิกภายใน หรือต่อผ่านฟลิปฟลอปที่ได้เตรียมไว้ให้สามารถทำการซิงโครนัสหรือไม่ก็ได้ ส่วนในแต่ละเอาท์พุตไดรเวอร์สามารถที่จะถูกโปรแกรมได้อย่างอิสระในการที่จะกำหนดให้มีค่าของแรงดันของสัญญาณที่มีมาตรฐานแตกต่างกันได้ และสามารถที่จะขับกระแสได้สูงถึง 24 mA. รับ (Sink) กระแสได้สูงถึง 48 mA. ในการที่จะใช้ในระบบบัสต่าง ๆ เพื่อให้ค่าสัญญาณรบกวนที่เกิดในระบบมีค่าต่ำที่สุด โดยส่วนใหญ่แล้วมักจะให้ระดับมาตรฐานของสัญญาณเอาท์พุตด้าน high ขึ้นอยู่กับระดับแรงดันของ V_{CCO} จึงมีความจำเป็นในการที่จะต้องจ่ายแรงดันของ V_{CCO} นี้ให้มีค่าใกล้เคียงกับค่ามาตรฐานที่ได้กำหนดไว้ให้มากที่สุด ส่วนวงจร weak keeper ที่ถูกเชื่อมต่ออยู่ในแต่ละเอาท์พุตเมื่อถูกนำมาใช้งาน จะทำให้วงจรส่วนที่คอยตรวจวัดระดับแรงดันทำการตรวจวัดแรงดันและขับแรงดันที่ขาให้มีค่าสูงขึ้นหรือลดลงให้เหมาะสมกับระดับของสัญญาณทางด้านอินพุต ถ้าในกรณีที่ขาของมันถูกเชื่อมต่ออยู่กับสัญญาณที่มีลักษณะเป็นแหล่งจ่ายหลายชุด (multiple-source) จะทำให้วงจร weak-keeper ทำการคงค่าของสัญญาณในสถานะสุดท้ายไว้ถ้าทุกไดร์เวอร์ถูกยกเลิกการใช้งาน ดังนั้นการรักษาระดับแรงดันที่ถูกต้องของลอจิกในวิธีการแบบนี้จึงกำจัด bus chatter ไปได้เป็นอย่างดี เนื่องจากวงจร weak-keeper จะใช้ IOB อินพุตบัฟเฟอร์มาทำหน้าที่เป็นตัวตรวจวัดระดับของแรงดันอินพุต ดังนั้นการกำหนดและป้อนแรงดัน V_{ref} ที่เหมาะสม จึงเป็นสิ่งที่จำเป็นถ้าต้องการระดับแรงดันในมาตรฐานของ

สัญญาณในระดับนั้น ๆ แต่อย่างไรก็ตามการป้อนแรงดันจำเป็นจะต้องให้เป็นไปตามกฎของ I/O banking rule คงไว้ด้วย

3.7.1.5 กลุ่มของขาสัญญาณ I/O Banking

ในบางมาตรฐานของ I/O มีความจำเป็นต้องใช้แรงดัน V_{CCO} และหรือ V_{ref} ซึ่งระดับแรงดันเหล่านี้จะต้องป้อนจากภายนอกเข้ากับขาของชิพเพื่อให้กับกลุ่มของ IOB ที่เรียกว่า แบงก์ ดังนั้นการกำหนดระดับของแรงดันมาตรฐานของ I/O จึงสามารถที่ทำได้ภายในแบงก์ที่กำหนดเท่านั้น ภายในชิพจะมี 8 แบงก์ แยกกันอยู่ด้านละ 2 แบงก์ในแต่ละด้านของชิพ ดังรูปที่ 3.14 แต่ละแบงก์ก็จะมีขา V_{CCO} อยู่หลายขา และทุกขาจำเป็นจะต้องถูกต่อให้มีระดับแรงดันเดียวกัน ซึ่งระดับแรงดันนี้จะถูกกำหนดโดยค่าของแรงดันมาตรฐานของเอาท์พุตที่ถูกนำมาใช้

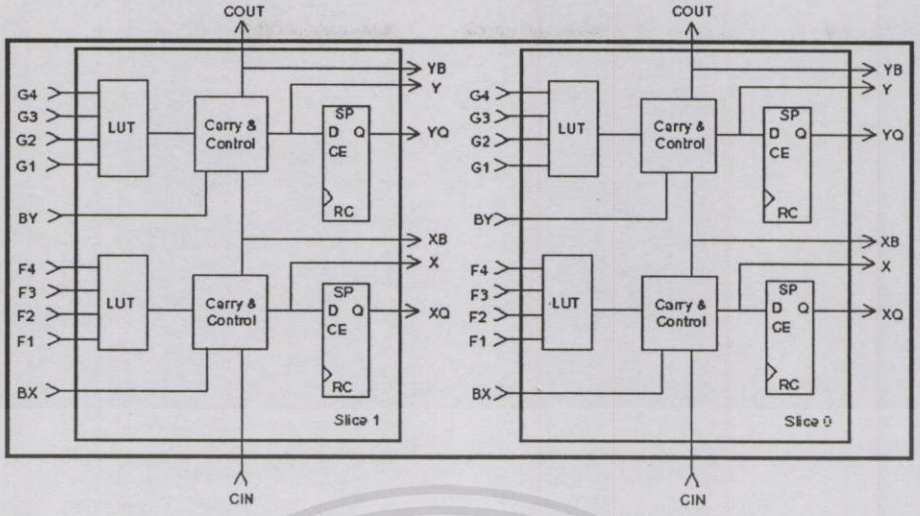


รูปที่ 3.14 กลุ่มของ I/O แบงก์ ภายใน Virtex-E

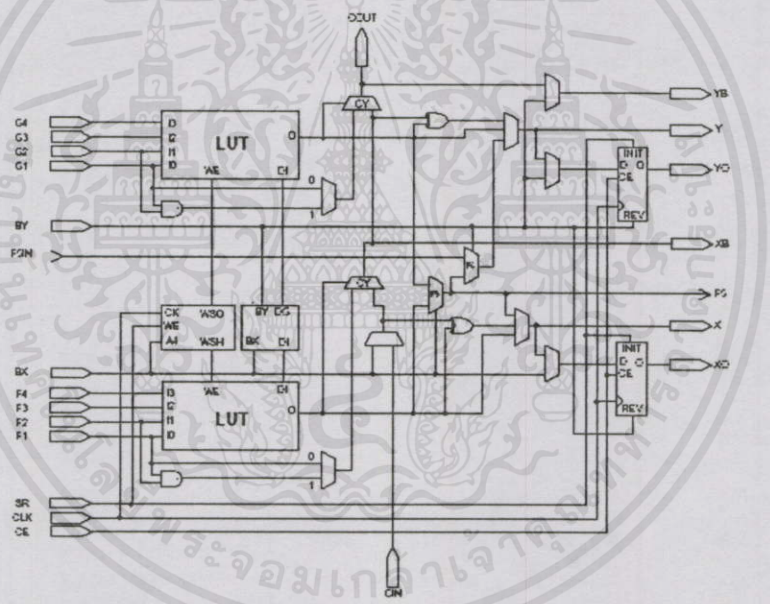
3.7.2 โครงสร้างภายในของชิพ Virtex-E

3.7.2.1 โครงสร้างของ Configurable Logic Block

พื้นฐานในการสร้างบล็อกของ VirtexE-CLB จะเป็นแบบลอจิกเซลล์ (Logic cell , LC) ภายใน LC จะประกอบไปด้วยฟังก์ชันเจนเนอเรเตอร์ขนาด 4 อินพุตอยู่ 1 ตัว carry logic และ ส่วนของการเก็บข้อมูลสัญญาณจากเอาท์พุตของฟังก์ชันเจนเนอเรเตอร์ในแต่ละ LC จะขับทั้งเอาท์พุตและขา D ของฟลิปฟลอป ในแต่ละ CLB จะประกอบไปด้วยโครงสร้างของ LC 4 ชุด อยู่ในโครงสร้างแบบ slice 2 ชุด ดังแสดงในรูปที่ 3.15 ส่วนรูปที่ 3.16 แสดงถึงโครงสร้างรายละเอียด



รูปที่ 3.15 โครงสร้างแบบ 2-Slice CLB ของ Virtex-E



รูปที่ 3.16 รายละเอียดภายใน Slice

ของภายใน slice นั่นก็คือจะพบว่ามี LC อยู่จำนวน 4 ชุด อยู่ภายในแต่ละ CLB ซึ่งจะประกอบรวมกันทั้งหมดแล้วจะทำให้ได้เป็นฟังก์ชันเจนเนอเรเตอร์ขนาด 5 ถึง 6 อินพุต ดังนั้นเมื่อทำการประมาณจำนวนของ system gate ที่อยู่ภายในอุปกรณ์ก็จะทำให้ได้ค่าเป็นว่าในแต่ละ CLB จะมีอยู่ประมาณ 4.5 LCs. จากรูปที่ 3.16 ตัวฟังก์ชันเจนเนอเรเตอร์ของ Virtex-E จะถูกสร้างจากตาราง look-up table (LUT) ขนาด 4 อินพุตซึ่งก็จะทำให้การทำงานเป็นฟังก์ชันที่ในแต่ละ LUT มีค่าเป็น 16x1 บิต ซิงโครนัส RAM หรืออาจจะนำในแต่ละ slice ที่มี LUT อยู่สองชุดมารวมกันเพื่อสร้าง 16x2 บิต ซิงโครนัส RAM ได้ ตัว LUT ยังสามารถที่จะทำให้เป็นชิฟท์รีจิสเตอร์ขนาด

16 บิต ได้อีกซึ่งเป็นแนวคิดสำหรับการทำ capturing high-speed หรือ burst mode data ซึ่งโหมดนี้สามารถที่จะถูกใช้งานเพื่อการเก็บข้อมูลในการประยุกต์ใช้งานทางด้าน DSP และส่วนของการเก็บข้อมูลก็สามารถที่จะถูกกำหนดให้เป็นแบบ D-type ฟลิปฟลอป หรือเป็นแบบแลทช์ก็ได้ โดยที่ในส่วนของ D อินพุตสามารถที่จะถูกขับได้จากทั้งฟังก์ชันเจนเนอเรเตอร์ที่อยู่ภายใน slice เอง หรือขับโดยตรงจากอินพุตของ slice โดยการบายพาสการทำงานของฟังก์ชันเจนเนอเรเตอร์ ส่วนสัญญาณนาฬิกาและสัญญาณเอ็นเนเบิลในแต่ละ slice จะเป็นแบบซิงโคนัสเซ็ต และรีเซ็ต (SR และ BY) ในทางกลับกันสัญญาณทั้งสองนี้สามารถที่จะถูกกำหนดให้ทำงานเป็นแบบอะซิงโครนัสก็ได้เช่นกัน นอกจากนี้ตัวมัลติเพล็กเซอร์ F_5 ในแต่ละ slice จะทำให้ฟังก์ชันการทำงานเพิ่มได้เป็นเสมือนมี 5 อินพุต จากตัวมัลติเพล็กเซอร์ขนาด 4 : 1 หรือทำให้สามารถเลือกฟังก์ชันได้สูงถึง 9 อินพุต ในทำนองเดียวกันตัวมัลติเพล็กเซอร์ F6 ก็จะรวมเอาท์พุตของฟังก์ชันเจนเนอเรเตอร์ทั้ง 4 ตัวใน CLB โดยการเลือกตัวใดตัวหนึ่งจากเอาท์พุตของ F5 ทำให้สามารถใช้มัลติเพล็กเซอร์แบบ 6 อินพุต มาทำงานเป็นแบบมัลติเพล็กเซอร์ขนาด 8 : 1 หรือทำให้สามารถเลือกฟังก์ชันได้สูงถึง 19 อินพุตในแต่ละ CLB นอกจากนั้นใน CLB ยังมี ส่วนที่ใช้สำหรับทำการป้อนผ่านโดยตรงเป็นจำนวน 2 ชุดต่อ slice ทำหน้าที่เป็นทางด่วนข้อมูลแบบพิเศษหรือเป็นส่วนเพิ่มเติมของการเชื่อมต่อที่ไม่ต้องใช้วงจรลอจิกภายใน CLB

ในส่วนการกระทำทางด้านคำนวณก็จะมีตัว dedicate carry logic ที่มีอยู่ใน CLB จะเป็นตัวที่ทำให้ carry ที่ได้จากการคำนวณทางคณิตศาสตร์มีความเร็วสูงมากขึ้นให้เหมาะกับการคำนวณทางคณิตศาสตร์ที่ต้องการความเร็วมาก ๆ ในแต่ละ CLB จะมีเส้นทางของ carry อยู่ 2 โครงข่าย (ในแต่ละ slice จะมี 1 โครงข่าย)

ส่วนของการทำประมวลผลทางคณิตศาสตร์จะประกอบไปด้วย XOR เกท ทำให้เกิดเป็น full adder ขนาด 2 บิตในแต่ละ slice นอกจากนั้นก็จะมี AND เกทแบบพิเศษที่จะใช้สำหรับการทำวงจรคูณความเร็วสูง นอกจากนี้ส่วนของ dedicated carry นั้นสามารถที่จะนำมาใช้สำหรับการต่อ cascade ให้กับฟังก์ชันเจนเนอเรเตอร์ซึ่งก็จะทำให้ได้ฟังก์ชันของลอจิกที่ใช้งานได้กว้างขวางมากขึ้น และในแต่ละ CLB จะมีวงจรขับ BUFT อยู่ 2 ตัว ซึ่งจะใช้ขับระบบบัสภายในชิพ และในแต่ละตัวของ BUFT ก็จะมีขาควบคุมและขาอินพุตที่เป็นอิสระจากส่วนอื่น ๆ ภายใน CLB

3.7.2.2 โครงสร้างของหน่วยความจำภายใน (Block select RAM)

ชิพรุ่น Virtex-E จะมี RAM ให้เลือกใช้ได้ตั้งแต่ขนาด 64 kbits จนถึง 832 kbits ขึ้นอยู่กับเบอร์ของ FPGA เกิดจากการรวมหน่วยความจำที่เรียกว่า block select RAM จำนวนมากไว้ด้วยกัน ซึ่งมันจะมีลักษณะของการกระจายของหน่วยความจำนี้ไปบนโครงสร้างพื้นฐานของแต่ละ CLB ในแต่ละเซลของ Block Select RAM จะเป็นแบบซิงโครนัสพอร์ทคู่ขนาด 4096 บิต มีสัญญาณควบคุมที่เป็นอิสระต่อกันในแต่ละพอร์ท ขนาดความกว้างของข้อมูลในแต่ละพอร์ท

สามารถที่จะถูกกำหนดได้อย่างเป็นอิสระต่อกันซึ่งก็รวมไปถึงการเปลี่ยนขนาดความกว้างของบัสข้อมูลได้ด้วย นอกจากนั้นกลุ่ม RAM เหล่านี้ยังถูกออกแบบให้มีลักษณะเป็นการเชื่อมต่อแบบพิเศษเพื่อที่จะทำให้สามารถนำไปใช้ทำการเชื่อมต่อกับ CLB และ Block RAM อื่น ๆ ได้อย่างพอเพียงกับความต้องการ

3.7.2.3 โครงข่ายการเชื่อมต่อของขาสัญญาณอินพุตทเอาท์พุต (I/O routing)

ตัวชิพ Virtex-E ได้มีการเพิ่มส่วนของการเชื่อมต่อของอุปกรณ์อื่น ๆ ที่อยู่รอบ ๆ ตัวมันและมีรูปแบบเป็นการเชื่อมต่อระหว่าง CLB อาร์เรย์กับ IOB การเพิ่มการเชื่อมต่อในส่วนนี้จะเรียกว่า Versa Ring ซึ่งจะมีความสะดวกมากในการปรับตำแหน่งขาของชิพ (pin-swapping , pin-locking) เช่น ในกรณีที่มีการออกแบบลอจิกใหม่ ก็จะทำให้สามารถปรับเปลี่ยนโครงสร้างภายในได้โดยยังคงขาของอุปกรณ์ไว้ในตำแหน่งเดิมได้บนแบบของแผ่นวงจรพิมพ์เดิม ทำให้ลดเวลาที่ใช้ออกสู่ตลาดทั้งนี้เพราะว่าการออกแบบแผ่นวงจรพิมพ์และระบบอื่น ๆ สามารถที่จะออกแบบและทำการผลิตได้แม้กระทั่งในระหว่างที่ส่วนของลอจิกยังอยู่ในระหว่างการออกแบบอยู่ก็ตาม

3.7.2.4 โครงข่ายการเชื่อมต่อภายในแบบพิเศษ (Dedicate routing)

ในโครงสร้างของ VirtexE มีการเตรียมชิ้นส่วนพิเศษที่ใช้ในการเชื่อมต่อเป็นสายสัญญาณโดยเฉพาะสำหรับกลุ่มของสายสัญญาณเชื่อมต่อที่เตรียมไว้ คือ กลุ่มของสายสัญญาณชนิดพิเศษที่ใช้ในการทำ multiple bus ระหว่าง CLB ซึ่งจะทำให้การทำงานของระบบมีความเร็วสูงขึ้นได้ และจะมีเครือข่ายในแต่ละ CLB เพื่อใช้สำหรับการรองรับการเกิดการแผ่กระจายของสัญญาณ carry ของ CLB ที่อยู่ติดกัน นอกจากนี้ยังมียังมีระบบสายสัญญาณนาฬิกาแบบพิเศษที่เรียกว่า Global Clock Distribution Network เพื่อป้องกันการเกิดการเบี่ยงเบนที่รุนแรงของสัญญาณนาฬิกาได้

3.7.2.5 โครงข่ายการเชื่อมต่อของสัญญาณนาฬิกา (Clock routing)

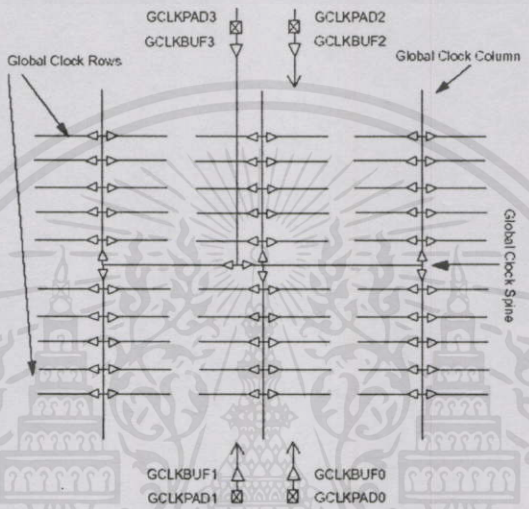
ในส่วนของการเชื่อมต่อของสัญญาณนาฬิกาจะมีการกระจายสัญญาณนาฬิาและสัญญาณอื่น ๆ ด้วยค่าของ fanout ที่สูงผ่านไปตลอดทั้งชิพซึ่งใน Virtex-E จะประกอบไปด้วยโครงข่ายการเชื่อมต่อสัญญาณนาฬิกาอยู่ 2 แถวคือ global routing และ local clock routing โดย

1) Global routing จะมีโครงข่ายอยู่ 4 ชุดที่มีลักษณะทางอินพุตเป็นแบบพิเศษที่ออกแบบมาให้มีการกระจายของ fanout ที่มีค่าสูงของสัญญาณนาฬิกาและมีค่าเบี่ยงเบนต่ำที่สุดในแต่ละโครงข่าย สามารถที่จะนำไปขับ CLB , IOB , Block RAM ได้พร้อมกันทั้งหมด

2) Local clock routing กลุ่มของโครงข่ายนี้มีความยืดหยุ่นมากกว่า global routing เพราะว่ามันจะไม่ถูกยึดติดกับการเชื่อมต่อเพียงกับขาสัญญาณนาฬิกาเพียงอย่างเดียวจึงสามารถที่จะประยุกต์ใช้งานกับสัญญาณอื่นในกรณีที่ต้องการให้เกิดค่าเบี่ยงเบนของสัญญาณต่ำได้อีก

3.7.2.6 การกระจายของสัญญาณนาฬิกาหลัก (Global clock distribution)

ในชิพ Virtex-E จะมีโครงสร้างของการกระจายสัญญาณนาฬิกาเป็นแบบมีความเร็วสูง ความเบี่ยงเบนต่ำผ่านโดยผ่าน global routing มีโครงข่ายของการกระจายแสดงไว้ดังรูปที่ 3.17 จากรูปจะมีบัฟเฟอร์หลัก (GCLKBUF) อยู่ทั้งหมด 4 ตัว โดย 2 ตัวอยู่ด้านบน และส่วนที่เหลืออยู่ด้านล่างตรงกลางของโครงข่าย ซึ่งมันจะทำหน้าที่คอยขับสัญญาณให้โครงข่ายหลักที่จะส่งต่อไปเพื่อขับสัญญาณนาฬิกาให้กับส่วนประกอบอื่น ๆ ต่อไป



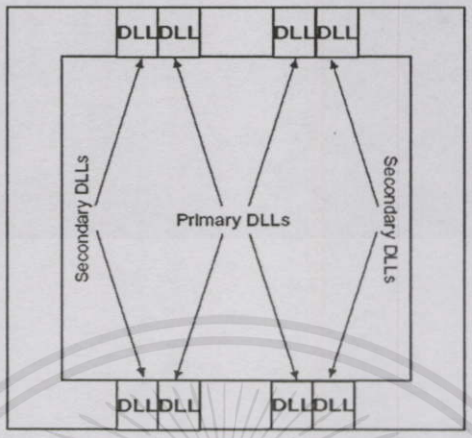
รูปที่ 3.17 โครงข่ายของการกระจายสัญญาณ Global Clock

3.7.2.7 Digital Delay-Lock Loop

ในชิพแต่ละตัวจะมีอุปกรณ์ Delay-Locked Loop (DLL) อยู่ 8 ตัวโดย 4 ตัวจะอยู่ด้านบนและอีก 4 ตัวอยู่ด้านล่างดังรูปที่ 3.18 ตัว DLL สามารถที่จะถูกนำมาใช้สำหรับการกำจัดความเบี่ยงเบนที่เกิดขึ้นระหว่างขาสัญญาณนาฬิกาของชิพกับขาสัญญาณนาฬิกาของเซลล์ต่าง ๆ ที่ต่ออยู่ภายในชิพ โดยแต่ละ DLL สามารถที่จะใช้ขับโครงข่ายหลักของสายสัญญาณได้ 2 วงจร โดยทั่วไปแล้ว DLL จะทำการปรับค่าเวลาหน่วงของสัญญาณนาฬิกาโดยอัตโนมัติ มันจึงถูกใช้เป็นตัวตรวจสอบระบบสัญญาณนาฬิกาของอินพุตและทำหน้าที่แจกจ่ายสัญญาณนาฬิกาให้กับระบบ

นอกจากการกำจัดค่าของเวลาหน่วงในการแจกจ่ายสัญญาณนาฬิกาแล้ว ตัว DLL ได้รวมเอาส่วนของการทำการควบคุมแบบพิเศษเข้ามาด้วย เพื่อให้สามารถสร้างสัญญาณนาฬิกาได้หลายคาบโดยที่ตัว DLL จะแบ่งเฟสของสัญญาณนาฬิกาออกเป็น 4 ควอเดรเจอร์ และสามารถที่ทำการทวีคูณความถี่ขึ้นหรือหารความถี่ลงได้ โดยปกติแล้วตัว DLL จะมีการทำงานอยู่ในรูปแบบของ clock mirror ดังนั้นด้วยการทำให้เกิดการขับสัญญาณจาก DLL ภายในชิพแล้วจากนั้นก็ป้อน

สัญญาณที่ได้กลับเข้าไปใน DLL อีกครั้ง จึงทำให้มันสามารถที่จะลดค่าเบี่ยงเบนของสัญญาณนาฬิกาบนบอร์ดในกรณีที่มีการใช้ชิพบนบอร์ดมากกว่า 1 ตัวได้



รูปที่ 3.18 การจัดวางตำแหน่งของ DLL

บทที่ 4

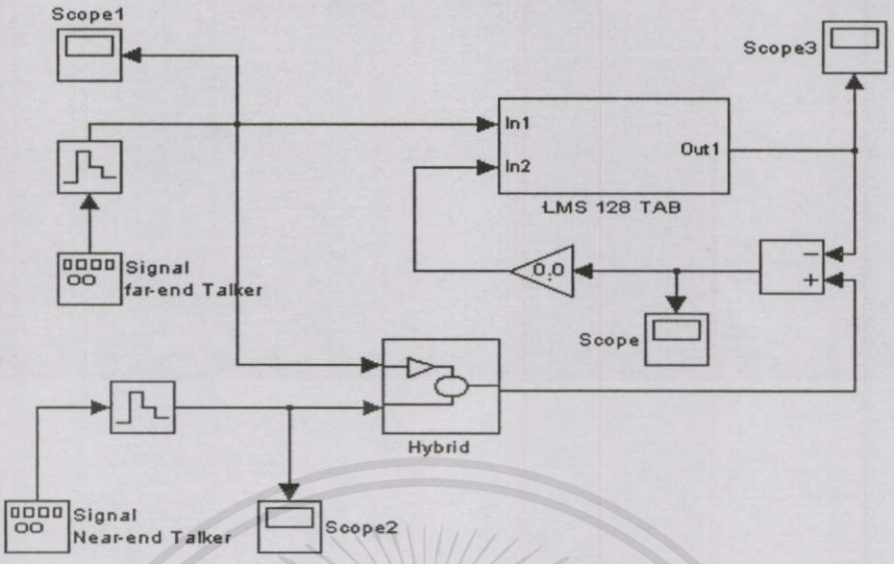
การออกแบบ สร้าง และจำลองการทำงานของตัวกำจัดสัญญาณเสียงสะท้อน

4.1 บทนำ

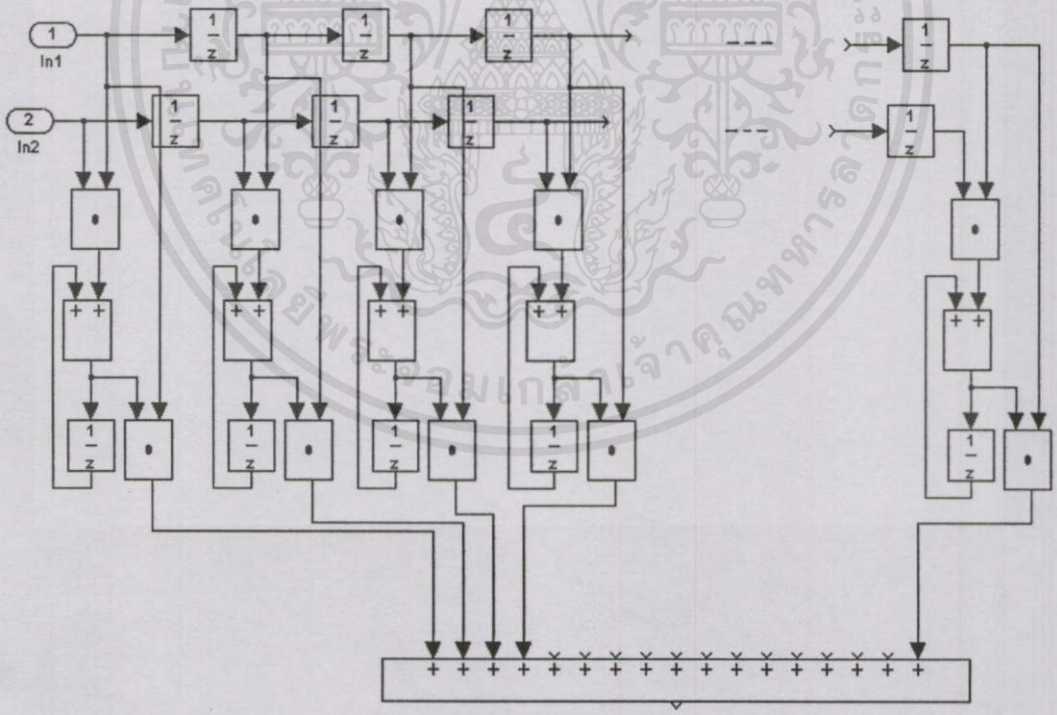
ในบทนี้จะแสดงถึงแนวทางที่ใช้ในการออกแบบและสร้างตัวกำจัดเสียงสะท้อนโดยในการออกแบบสร้างได้แบ่งการดำเนินการออกเป็นสองส่วนคือ ส่วนแรกจะเป็นการออกแบบวงจรกำจัดเสียงสะท้อนโดยการจำลองแบบเป็นโมเดลทางคณิตศาสตร์เพื่อหาค่าพารามิเตอร์ควบคุมอัตราการลู่เข้าที่ทำให้ระบบสามารถทำงานอยู่ในสภาวะเสถียรได้โดยใช้ซอฟแวร์โปรแกรม MATLAB จากนั้นก็จะเป็นการออกแบบในส่วนที่สองคือจะนำเอาโมเดลทางคณิตศาสตร์ที่ได้มาออกแบบสร้างเครื่องต้นแบบขึ้นโดยใช้อุปกรณ์ฮาร์ดแวร์ สำหรับการออกแบบสร้างจะมีลักษณะเป็นการออกแบบจากบนลงล่าง (Top-down design) ซึ่งจะเริ่มต้นจากการวิเคราะห์ถึงฟังก์ชันการทำงานของระบบโดยรวมก่อนเป็นอันดับแรก จากนั้นจึงทำการออกแบบฟังก์ชันการทำงานของระบบย่อย ๆ ที่จะมาประกอบรวมกันขึ้นเป็นระบบหลัก ในการออกแบบระบบย่อยจะใช้การกำหนดฟังก์ชันโดยภาษา VHDL ร่วมกับการออกแบบโดยการใช้แบบแผนวงจร (Schematic diagram) เพื่อสร้างแบบแผนวงจรของระบบ เมื่อได้แบบแผนวงจรที่มีความสมบูรณ์แล้ว จากนั้นก็จะใช้ซอฟแวร์โปรแกรม Xilinx Foundation series V3.1 มาทำการแปลงข้อมูลของแบบแผนวงจรให้กลายเป็นลักษณะข้อมูลสำหรับการทำคอนฟิกกูเรชันสำหรับนำไปดาวน์โหลดลงในชิพ FPGA เบอร์ XCV600EPQ240-7 ที่อยู่ในตระกูล Virtex-E ของบริษัท Xilinx และขั้นตอนสุดท้ายก็จะเป็นการนำเอา ชิพ FPGA ที่ผ่านการคอนฟิกกูเรชันแล้วไปทำงานร่วมกับอุปกรณ์ประกอบอื่น ๆ ที่จำเป็น เพื่อสร้างเป็นเครื่องต้นแบบของตัวกำจัดเสียงสะท้อน

4.2 แบบจำลองทางคณิตศาสตร์ของวงจรกำจัดเสียงสะท้อน

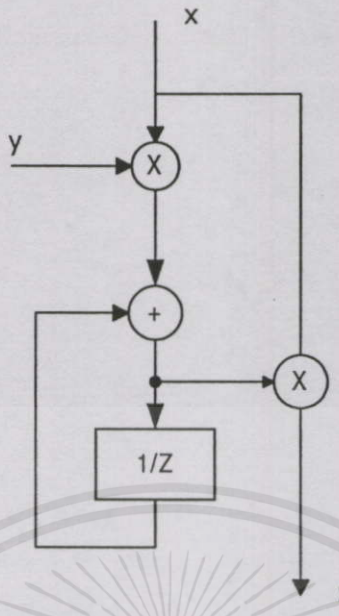
โครงสร้างของวงจรกำจัดเสียงสะท้อนเมื่อนำมาเขียนแบบโมเดลทางคณิตศาสตร์โดยใช้โปรแกรม Simulink และโปรแกรม MATLAB ได้ดังในรูปที่ 4.1 4.2 และ 4.3 และเมื่อทำการทดสอบการทำงานของโมเดลโดยการกำหนดเงื่อนไขต่าง ๆ เพื่อจำลองสภาวะการทำงานของระบบและทำการลองเปลี่ยนค่าพารามิเตอร์ที่ควบคุมอัตราการลู่เข้าของระบบจนได้ค่าที่เหมาะสมที่จะนำไปทำการออกแบบเป็นวงจรต้นแบบ โดยผลของการจำลองเมื่อกำหนดให้ค่าพารามิเตอร์ควบคุมอัตราการลู่เข้าของวงจรกรองมีค่าเป็น 0.00125 และสัญญาณทดสอบเป็นรูปสี่เหลี่ยมความถี่ 1 kHz แสดงได้ดังรูปที่ 4.4



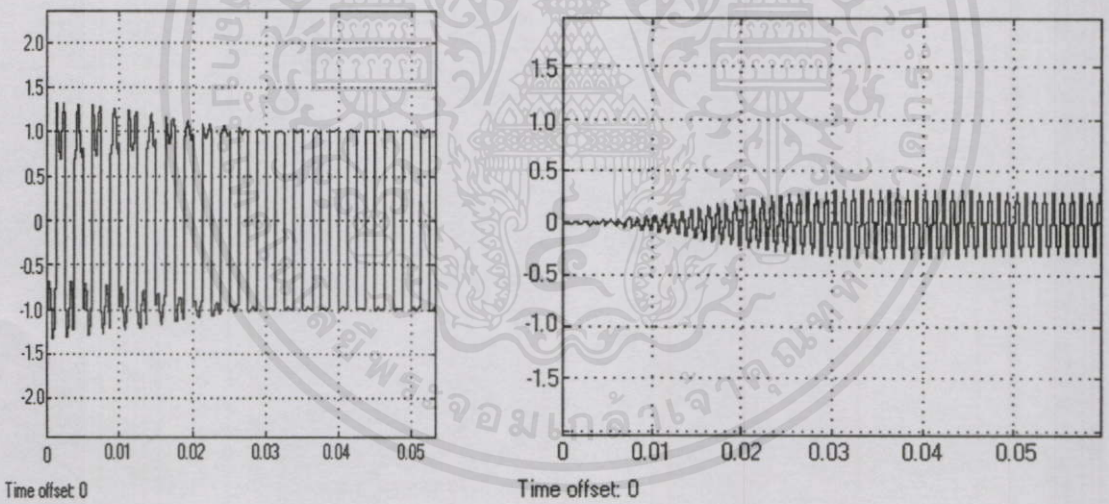
รูปที่ 4.1 โมเดลการทำงานทางคณิตศาสตร์ของตัวกำจัดสัญญาณเสียงสะท้อน



รูปที่ 4.2 โมเดลทางคณิตศาสตร์ในส่วนของ LMS อะแดปทีฟฟิลเตอร์ 128 Tap



รูปที่ 4.3 โมเดลทางคณิตศาสตร์ในส่วนของ Coefficient update module



ก) สัญญาณเอาท์พุตของตัวกำจัดเสียงสะท้อนที่ได้จาก Scope

ข) สัญญาณเอาท์พุตของอะแดปทีฟฟิลเตอร์ที่ได้จาก Scope 3

รูปที่ 4.4 ผลการทดสอบการทำงานของโมเดล ตามรูปที่ 4.1

4.3 การออกแบบและโครงสร้างของตัวกำจัดสัญญาณเสียงสะท้อน (Voice Echo Canceller)

การออกแบบโครงสร้างของตัวกำจัดสัญญาณเสียงสะท้อนจะเป็นการนำเอาโมเดลทางคณิตศาสตร์ที่ได้จากการจำลองโดยโปรแกรม MATLAB มาสร้างบนอุปกรณ์ฮาร์ดแวร์ โดยจะทำการออกแบบในแต่ละส่วนแยกจากกัน จากนั้นก็ทำการทดสอบการทำงานในระดับฟังก์ชันจนมีความถูกต้องตรงตามต้องการก่อนแล้วจึงนำแต่ละส่วนเข้ามาเชื่อมต่อกันเป็นระบบที่สมบูรณ์ ดังนั้นระบบที่ออกแบบจึงมีลักษณะเป็นระบบที่ทำการซิงโครนัสกันโดยใช้สัญญาณนาฬิกาจากภายนอก และเพื่อให้ วงจรของตัวกำจัดสัญญาณเสียงสะท้อนมีคุณสมบัติที่ดีจึงได้มีการกำหนดคุณลักษณะที่ต้องการในส่วนของวงจรกรองแบบดิจิตอลไว้ดังต่อไปนี้

1) ขนาดของค่าความถี่อัตราการสุ่ม (Sampling rate frequency) เป็น 40 kHz

2) ค่าของความแม่นยำ (Resolution) ของสัญญาณอินพุต 12 บิต (ขึ้นอยู่กับอุปกรณ์แปลงสัญญาณ) ซึ่งจะส่งผลให้มีค่าผิดพลาดจากการควอนไทซ์ต่ำกว่า -70 dB จนไม่ส่งผลต่อการประมวลผลของสัญญาณข้อมูล

3) การประมวลผลข้อมูลและการตัดปลายของข้อมูลภายในระบบ มีความผิดพลาดไม่เกิน 0.001% ของข้อมูลทางออกของระบบ

4) จำนวนของ order ของวงจรกรองเป็น 128 tap หรือมากกว่านั้น เพื่อให้ครอบคลุมค่าของเวลาหน่วงที่มากที่สุดที่อาจจะมีได้ในระบบของโทรศัพท์ตามมาตราฐาน ITU-T G.165

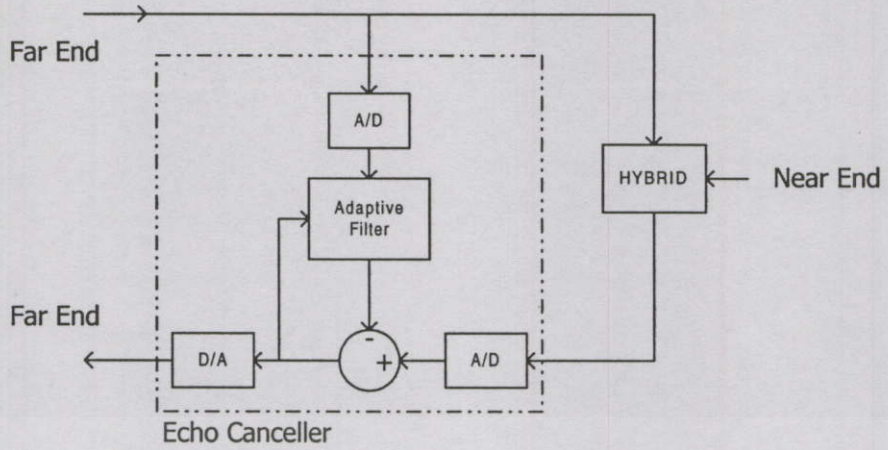
4.3.1 โครงสร้างและการเชื่อมต่อของวงจรกำจัดสัญญาณเสียงสะท้อน

โครงสร้างของตัวกำจัดสัญญาณเสียงสะท้อนที่ทำการออกแบบและสร้างขึ้นนี้จะมีการทำงานร่วมอยู่กับวงจรในส่วนที่เป็นโครงข่ายของ 4-wire network แสดงดังรูปที่ 4.5 โดยสัญญาณที่เข้าสู่วงจรกำจัดสัญญาณเสียงสะท้อนจะเป็นสัญญาณเสียงที่มีย่านความถี่ในช่วง 300 - 3400 Hz และถูกแปลงให้เป็นสัญญาณดิจิตอลโดยวงจร A/D converter เพื่อป้อนให้กับวงจรกรองแบบปรับตัวได้ทำการคำนวณหาค่าของสัญญาณเสียงสะท้อนและส่งไปหักล้างออกจากสัญญาณเสียงในระบบโทรศัพท์โดยวงจร sum ต่อไป

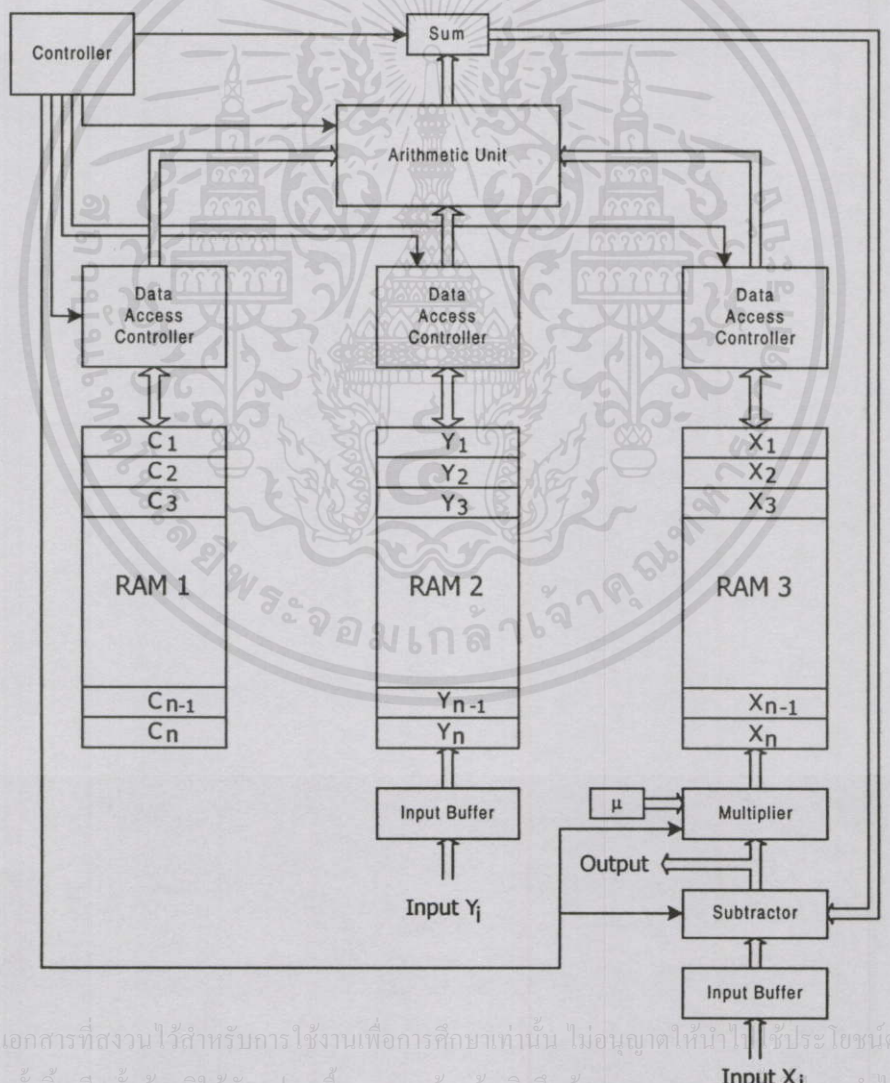
จากรูปที่ 4.1 – 4.3 จะพบว่าโครงสร้างของวงจรกำจัดสัญญาณเสียงสะท้อนจะประกอบไปด้วยส่วนประกอบหลัก 2 ส่วนคือ

1) วงจรที่แปลงสัญญาณอนาลอกของสัญญาณเสียง ให้เป็นสัญญาณแบบดิจิตอลและแปลงกลับจากสัญญาณแบบดิจิตอลให้เป็นสัญญาณอนาลอก

2) วงจรกรองปรับตัวได้แบบ FIR อะแดปทีฟฟิลเตอร์ ซึ่งจะเป็นหัวใจหลักของระบบ และวงจร sum ที่จะทำการกำจัดสัญญาณสะท้อน



รูปที่ 4.5 การเชื่อมต่อของวงจรกำจัดสัญญาณเสียงสะท้อนกับโครงข่ายแบบ 4-wire network



รูปที่ 4.6 ระบบงานย่อยภายในส่วนของ FIR อะแดปทีฟฟิลเตอร์ และวงจร sum

เนื่องจากการทำงานของโครงสร้างทั้งสองส่วนจะมีการทำงานร่วมกัน และข้อมูลที่เข้าสู่ระบบมีลักษณะเป็นอนุกรมเวลา (time series) ดังนั้นในการออกแบบสร้างระบบงานย่อย ๆ ภายใน FIR อะแดปทีฟฟิลเตอร์ จึงจำเป็นจะต้องมีการแบ่งโครงสร้างของแต่ละส่วนให้ทำงานอย่างสัมพันธ์กันดังแสดงในรูปที่ 4.6

จากรูปที่ 4.6 จะแสดงให้เห็นถึงโครงสร้างของระบบย่อย ๆ ที่ประกอบรวมกันขึ้นเป็น FIR อะแดปทีฟฟิลเตอร์ และส่วนของวงจร sum ของระบบซึ่งจะประกอบไปด้วยส่วนต่าง ๆ ดังต่อไปนี้

1) วงจรควบคุม (Controller) จะเป็นส่วนที่คอยทำหน้าที่สร้างสัญญาณต่าง ๆ เช่น สัญญาณควบคุมอัตราการสุ่มข้อมูล สัญญาณนาฬิกาของระบบและสัญญาณควบคุมเพื่อควบคุมการทำงานของระบบย่อย ๆ ทั้งหมดให้ทำหน้าที่อย่างสัมพันธ์กันกับการเลื่อนไหล (Shift) ของข้อมูล

2) วงจรคำนวณทางคณิตศาสตร์ (Arithmetic) จะเป็นส่วนของการคำนวณทางคณิตศาสตร์ในการประมวลหาค่าสัมประสิทธิ์ของวงจรกรอง และนำไปหาองค์ประกอบทางความถี่ในแต่ละ tap ส่งให้กับวงจร sum วงจรในส่วนของหน่วยประมวลผลทางคณิตศาสตร์นี้จะประกอบไปด้วยวงจรคูณสองชุด วงจรบวกหนึ่งชุด เชื่อมต่อกันดังในรูปที่ 4.3

3) วงจรควบคุมการทำงานในหน่วยความจำ (Data Access controller) และ RAM จะเป็นส่วนที่ทำการอ่านและเขียนข้อมูลลงในหน่วยความจำ โดยวงจรในส่วนนี้จะประกอบไปด้วย RAM ขนาด16 บิต 2 ชุด สำหรับเก็บข้อมูลของ tap delay และขนาด 32 บิตอีก 1 ชุด สำหรับเก็บค่าสัมประสิทธิ์ของในแต่ละ tap ของ วงจรกรองและวงจรที่ทำการควบคุมการอ่านและเขียนหน่วยความจำ 3 ชุด โดยการทำงานของการอ่านและเขียนหน่วยความจำทั้งสามส่วนจะทำงานขนานกันและอิสระต่อกัน

4) วงจรคูณทางคณิตศาสตร์ (Multiplier) เป็นวงจรทำหน้าที่คูณข้อมูลทางอินพุตกับค่าคงที่เพื่อควบคุมอัตราการลู่เข้าของระบบ

5) วงจรลบทางคณิตศาสตร์ (Subtracter) เป็นวงจรที่จะทำการนำค่าของสัญญาณสะท้อนที่คำนวณได้มาหักออกจากสัญญาณทางอินพุตก่อนที่จะส่งออกเป็นสัญญาณ เอาท์พุตต่อไป

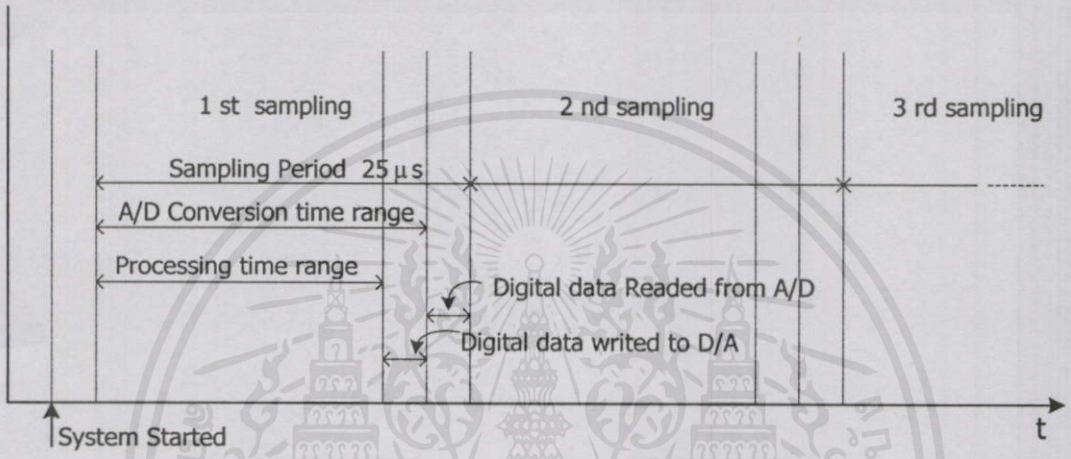
6) วงจรกำหนดค่าของอัตราการลู่เข้า (System Loop Gain , μ) จะเป็นวงจรที่ทำการกำหนดให้เป็นค่าคงที่ทางคณิตศาสตร์โดยมีรูปแบบเป็นเลขไบนารีแบบ 2's complement เพื่อนำไปเป็นตัวคูณกับค่าของข้อมูลทางอินพุต

7) วงจร sum จะเป็นวงจรสำหรับทำการบวกข้อมูลเข้าด้วยกันทุก ๆ ค่าของผลลัพธ์ที่ได้จากการประมวลผล

8) วงจรอินพุตบัฟเฟอร์ เป็นส่วนที่สำหรับเชื่อมต่อกับวงจรภายนอก ประกอบไปด้วยส่วนที่เป็นวงจรบัฟเฟอร์ วงจรแลทช์ข้อมูล และวงจรที่ทำการแปลงเลขไบนารีแบบไม่มีเครื่องหมายให้เป็นเลขแบบ 2's complement เพื่อให้ง่ายต่อการนำไปประมวลผลทางคณิตศาสตร์

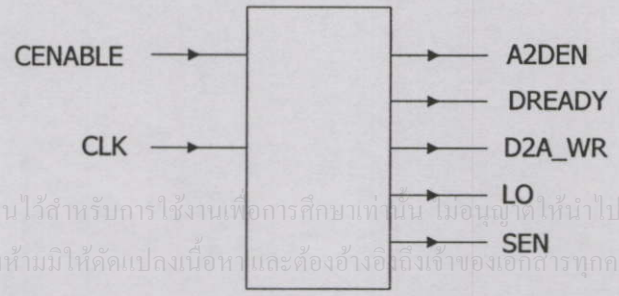
4.3.2 การออกแบบและโครงสร้างการทำงานของชุด Controller

วงจร controller เป็นหัวใจหลักในการควบคุมการทำงานของระบบ ทำหน้าที่สร้างสัญญาณการสุ่ม (Sampling signal) มีความถี่เป็น 40kHz สัญญาณควบคุมการอ่านและเขียนข้อมูลวงจร A/D Converter วงจร D/A Converter และหน่วยประมวลผลทั้งหมดในระบบตามลำดับ สำหรับการออกแบบลำดับขั้นการทำงานของวงจรจะมีแนวทางการทำงานตามไดอะแกรมทางเวลาแสดงได้ดังรูปที่ 4.7

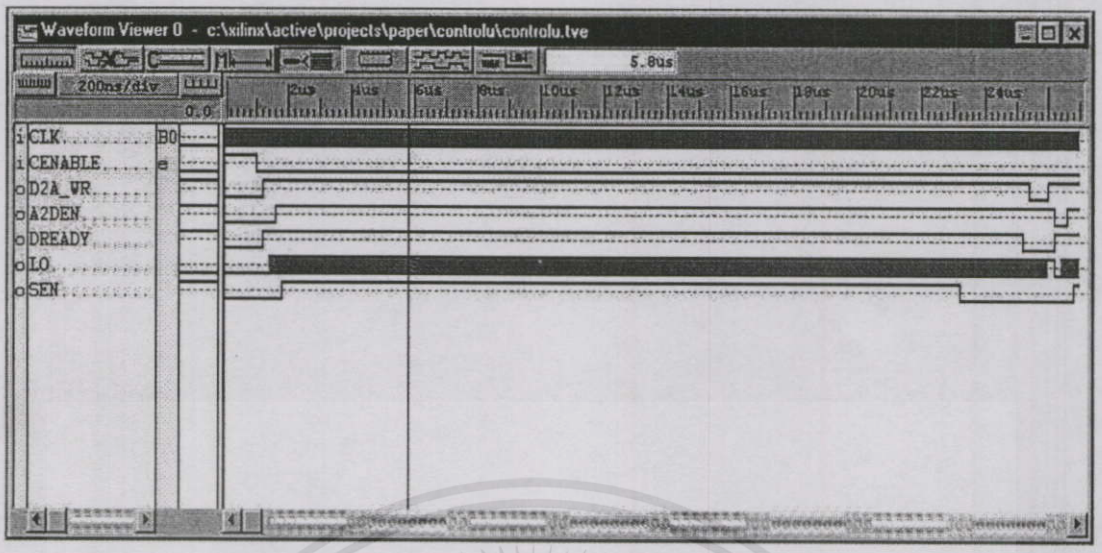


รูปที่ 4.7 ลำดับขั้นการควบคุมการอ่านเขียนและการประมวลผลข้อมูลของวงจรควบคุม

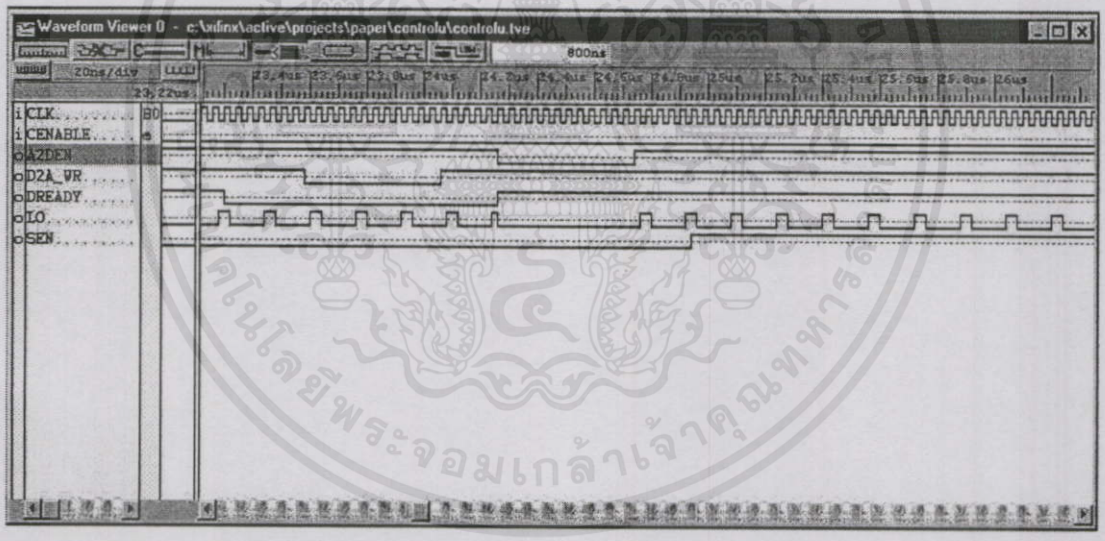
จากรูปที่ 4.7 ค่าของคาบเวลาสุ่ม (Sampling period) จะถูกกำหนดโดยค่าของอัตราความถี่สุมของข้อมูลทางด้านอินพุตของระบบซึ่งในที่นี้จะมีค่าเป็น 25 μs ดังนั้นค่าของเวลาที่จะใช้ในการอ่านข้อมูลจากวงจร A/D converter รวมกับเวลาการประมวลผล และเวลาของการเขียนข้อมูลจะต้องมีค่าไม่เกิน 25 μs ด้วย เมื่อนำไดอะแกรมทางเวลาจากรูปที่ 4.7 ไปสร้างเป็นวงจรการทำงานจริงโดยวิธีการออกแบบทางอัลกอริทึมด้วยภาษา VHDL และทดสอบการทำงานของวงจรด้วยโปรแกรม Xilinx Foundation Series V.3.1 จะได้ดังรูปที่ 4.8 - 4.10



รูปที่ 4.8 บล็อกไดอะแกรมของวงจร controller



รูปที่ 4.9 timing diagram แสดงคาบเวลาสุ่ม (ช่วงขอบขาขึ้นของสัญญาณ A2DEN signal) เป็น 25 μs



รูปที่ 4.10 timing diagram แสดงสัญญาณควบคุมต่าง ๆ ที่เข้าและออกจากวงจรควบคุม

จากรูปที่ 4.10 แสดงให้เห็นถึงสัญญาณต่าง ๆ ที่เกี่ยวข้องกับวงจรควบคุม คือ สัญญาณทางอินพุต จะประกอบไปด้วยสัญญาณนาฬิกาจากภายนอก (CLK) ซึ่งมีความถี่เป็น 24 MHz และสัญญาณ Chip Enable (CENABLE) ที่จะทำหน้าที่เป็นสัญญาณ Enable ของระบบ ส่วนสัญญาณทางเอาท์พุต ที่ได้คือ สัญญาณควบคุมของวงจร A/D Converter (A2DEN) ซึ่งจะทำงานในช่วงของขอบขาขึ้น และมี time period เป็น 25 μs และสัญญาณนี้จะถูกส่งไปควบคุมหน่วยต่าง ๆ ดังต่อไปนี้

1) หน่วยประมวลผล โดยเมื่อสัญญาณ A2DEN อยู่ในสถานะ High จะอนุญาตให้หน่วยประมวลผลทำการประมวลผลข้อมูลได้ แต่ถ้าอยู่ในสถานะ Low จะไม่อนุญาตให้หน่วยประมวลผลทำงาน

2) วงจรแปลง A/D converter จะเริ่มแปลงสัญญาณเมื่อขอบของสัญญาณ A2DEN เปลี่ยนสถานะจาก Low ไปเป็น High

3) วงจร Data input buffer จะแลทช์ข้อมูลที่ได้จากวงจร A/D Converter เมื่อสัญญาณ A2DEN เปลี่ยนสถานะจาก High ไปเป็น Low ก่อนที่จะส่งต่อให้กับหน่วยประมวลผลต่าง ๆ

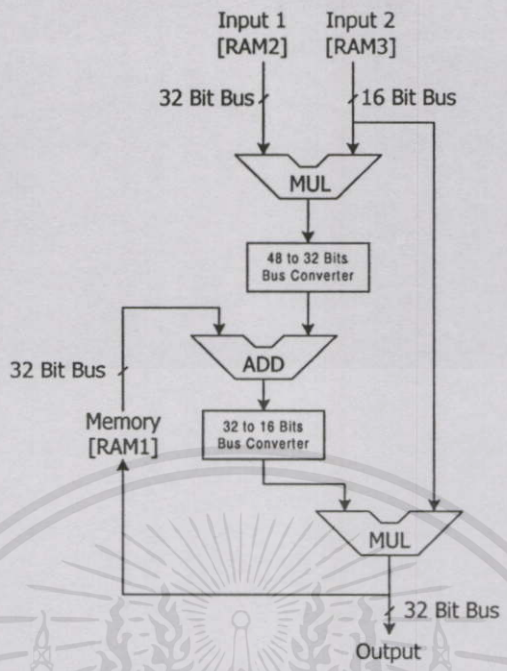
4) สัญญาณควบคุมของวงจร D/A Converter จะมีสองสัญญาณคือ Data Ready (DREADY) จะมีสถานะเป็น Active low เมื่อหน่วยประมวลผลทำงานเสร็จสิ้น และส่งข้อมูลที่ได้ไปยังเอาท์พุต และสัญญาณแลชท์ข้อมูลของ D/A Converter (D2A_WR) โดยจะทำงานเมื่อสัญญาณเปลี่ยนสถานะจาก Low ไปเป็น High

5) สัญญาณควบคุมการ sum ของวงจรกรองจะมีอยู่สองสัญญาณ คือ สัญญาณ Enable (SEN) โดยวงจร sum จะอยู่ในสถานะที่พร้อมทำงานเมื่อมีสถานะเป็น High และสัญญาณควบคุมการส่งผ่านข้อมูล (LO) จะทำการแลชท์ข้อมูลต่าง ๆ ที่มีเวลาหน่วง (delay) ต่างกัน ให้มีการหยุดรอกันก่อนที่จะเข้าสู่หน่วยคำนวณ

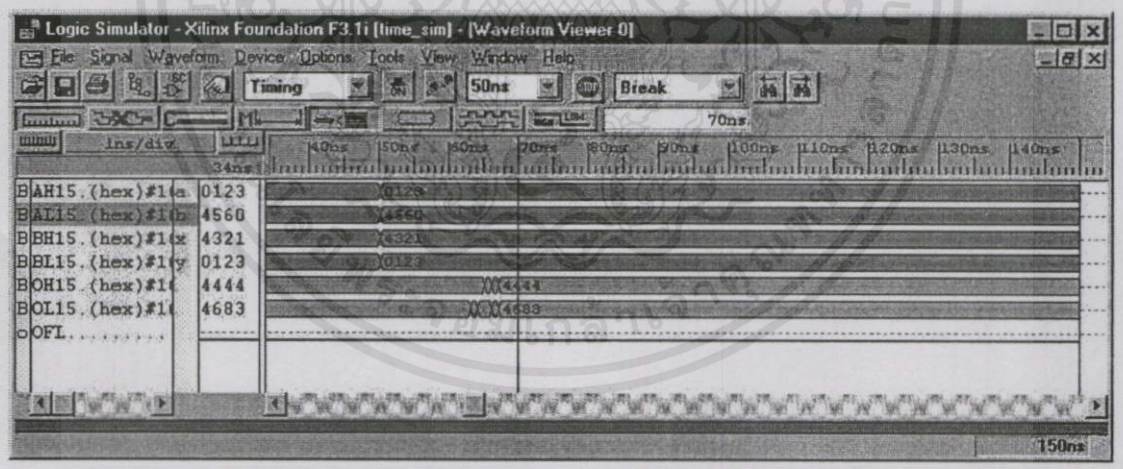
4.3.3 การออกแบบการทำงานของหน่วยประมวลผลทางคณิตศาสตร์ (Arithmetic Unit)

หน่วยประมวลผลทางคณิตศาสตร์ (Arithmetic Unit) เป็นโครงสร้างที่ประกอบไปด้วยวงจรคูณสองชุดและวงจรบวกอีกหนึ่งชุดเพื่อทำหน้าที่เป็น Coefficient update module ของวงจรกรองแบบปรับตัวได้ ดังแสดงในรูปที่ 4.11

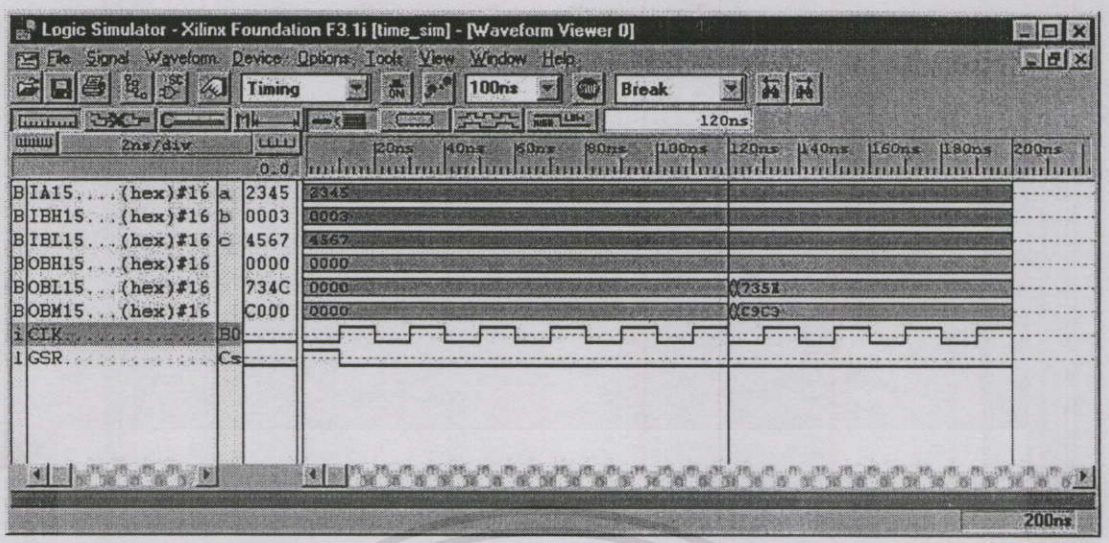
จากรูปที่ 4.11 การทำงานของ adder และ multiplier จะเป็นแบบ 2's complement และมีขนาดของบิตข้อมูลทางด้านอินพุตที่แตกต่างกันในแต่ละชุดทั้งนี้เพื่อให้มีความเหมาะสมกับลักษณะของข้อมูลและค่านัยสำคัญของข้อมูลที่สามารถยอมรับได้เมื่อมีความจำเป็นที่จะต้องลดจำนวนบิตของข้อมูลลงในบางขั้นตอนของกระบวนการประมวลผล ส่วนในรูปที่ 4.12 และ 4.13 จะเป็นการแสดงถึงความเร็วในการประมวลผลของวงจร adder และ multiplier เมื่อทำการทดสอบโดยการป้อนสัญญาณนาฬิกาขนาดความถี่ 48 MHz ให้กับระบบ ซึ่งพบว่าสามารถที่จะทำงานได้โดยใช้เวลาในการประมวลผลน้อยกว่า 20 ns สำหรับวงจร adder ขนาด 32 บิต และ 125 ns หรือ 6 คาบสัญญาณนาฬิกาสำหรับวงจร multiplier ขนาด 32 บิต



รูปที่ 4.11 โครงสร้างของหน่วยประมวลผลทางคณิตศาสตร์



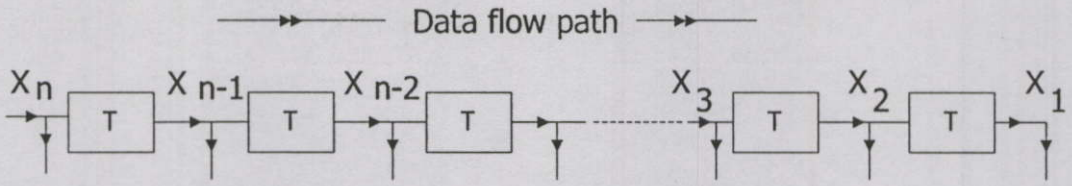
รูปที่ 4.12 ความเร็วในการคำนวณวงจร Adder ซึ่งใช้เวลาในการทำงานน้อยกว่า 20 ns



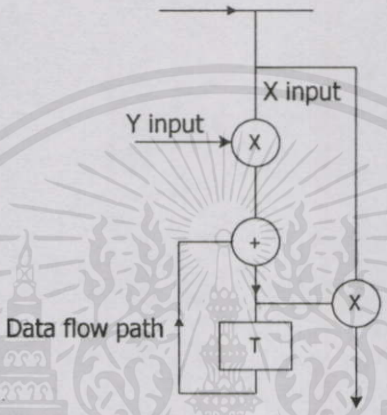
รูปที่ 4.13 ความเร็วในการคำนวณของวงจร Multiplier ซึ่งใช้เวลา 6 คาบของสัญญาณนาฬิกา

4.3.4 การออกแบบและโครงสร้างการทำงานของชุดวงจร Data Access Controller และ RAM

จากโครงสร้างของวงจรกรองในรูปที่ 4.1 – 4.3 มีจำนวน tap ของวงจรกรองเป็น 128 tap ดังนั้นหากจะออกแบบวงจรกรองตามโครงสร้างตามโมเดลทางคณิตศาสตร์แล้ว จะพบว่าจำเป็นจะต้องใช้โครงสร้างที่มีวงจรชิฟท์รีจิสเตอร์ 127 ชุด มาทำหน้าที่เป็น Unit delay (Z^{-1}) ในแต่ละเส้นทางการไหลของสัญญาณ (X และ Y) และในส่วนของการปรับค่าสัมประสิทธิ์ (coefficient) อีก 128 ชุด นอกจากนั้นจะต้องใช้วงจรคูณ 256 ชุด ทำงานร่วมกับวงจรบวก 128 ชุด จึงจะทำให้การทำงานของวงจรกรองสำเร็จลุล่วงได้ แต่ถ้าพิจารณาให้ดีจะพบว่าทั้งวงจรชิฟท์รีจิสเตอร์ วงจรบวก และวงจรคูณแต่ละชุดจะทำงานเพียงครั้งเดียวในแต่ละคาบเวลาของข้อมูลทางอินพุต หากมีการใช้วงจรหน่วยความจำ (RAM) มาทำงานแทนวงจรชิฟท์รีจิสเตอร์ และจัดโครงสร้างการส่งผ่านข้อมูลของวงจรบวกและวงจรคูณในวงจรกรองตามรูปที่ 4.3 ก็จะทำให้ต้องใช้วงจรบวกเพียง 1 ชุด ทำงานร่วมกับวงจรคูณ 2 ชุด ทำงาน 128 ครั้งในแต่ละคาบเวลา ก็จะทำให้การทำงานของวงจรกรองสำเร็จได้ผลดีเช่นเดียวกันโดยที่ใช้ทรัพยากรน้อยกว่าหลายเท่า ดังนั้นจึงได้มีการออกแบบวงจรกรองโดยใช้วงจรหน่วยความจำเป็นหลักและการเคลื่อนย้ายข้อมูลบนหน่วยความจำจึงจำเป็นจะต้องออกแบบเป็นสองรูปแบบตามลักษณะการเลื่อนไหลของข้อมูลคือ แบบแรกเป็นการเลื่อนข้อมูลของสัญญาณที่ได้จากอินพุตซึ่งจะมีลักษณะของการเลื่อนไหลของข้อมูลเป็นลำดับจากตัวที่ 1 ถึง n และแบบที่สองจะเป็นส่วนของการปรับค่าของสัมประสิทธิ์ซึ่ง ข้อมูลจะไม่มีการเลื่อนไหลแต่จะเป็นการกระทำซ้ำในตำแหน่งเดิม แสดงดังรูปที่ 4.14 และ 4.15 ตามลำดับ



รูปที่ 4.14 การเลื่อนไหลของข้อมูล (signal shifting) ของสัญญาณทางอินพุต X และ Y



รูปที่ 4.15 การเลื่อนไหลของข้อมูล (Signal shifting) ของสัญญาณ coefficient update

4.3.4.1 โครงสร้างการทำงานของวงจร RAM Access ในภาคของการ shift สัญญาณอินพุต X และ Y

จากรูปแบบการไหลของข้อมูลในรูปที่ 4.15 ทำให้สามารถออกแบบระบบควบคุมการอ่านและเขียนข้อมูลลงในหน่วยความจำได้ดังในรูปที่ 4.16 โดยระบบจะประกอบไปด้วยวงจรย่อย ๆ ที่มีฟังก์ชันการทำงานเป็นวงจรบัฟเฟอร์ของข้อมูล 2 ชุด วงจรนับ (Counter) 2 ชุด วงจรมัลติเพล็กซ์ข้อมูล (Multiplexer) 2 ชุด วงจรสร้างสัญญาณอ่านและเขียน (Read / Write signal generator) หน่วยความจำ และวงจรตรวจจับค่าแอดเดรส 0 เมื่อค่าบนแอดเดรสบัสมีค่าเป็น 0 โดยมีเงื่อนไขในการออกแบบก็คือ

1) วงจรนับทั้งสองจะนับค่าตั้งแต่ 128 , 127 , ... , 2 , 1 , 0

2) ค่าเริ่มต้นภายใน RAM จะมีค่าเป็นศูนย์

3) การปรับลดค่าของวงจรนับชุดที่ 2 (counter 2 ในรูปที่ 4.16) จะตามหลังค่าของวงจรนับชุดที่ 1 (counter 1) อยู่จำนวน 1 หน่วยเสมอ ยกเว้นค่าที่เริ่มต้นนับ (128) และค่าสุดท้ายที่นับ (0) สำหรับการทำงานของวงจรในการอ่านและเขียนข้อมูลบนหน่วยความจำในแต่ละวงรอบ (Cycle) จะมีอยู่ 5 ขั้นตอนคือ

ขั้นตอนที่ 1 กำหนดค่าของแอดเดรสด้วยค่าของ counter 1

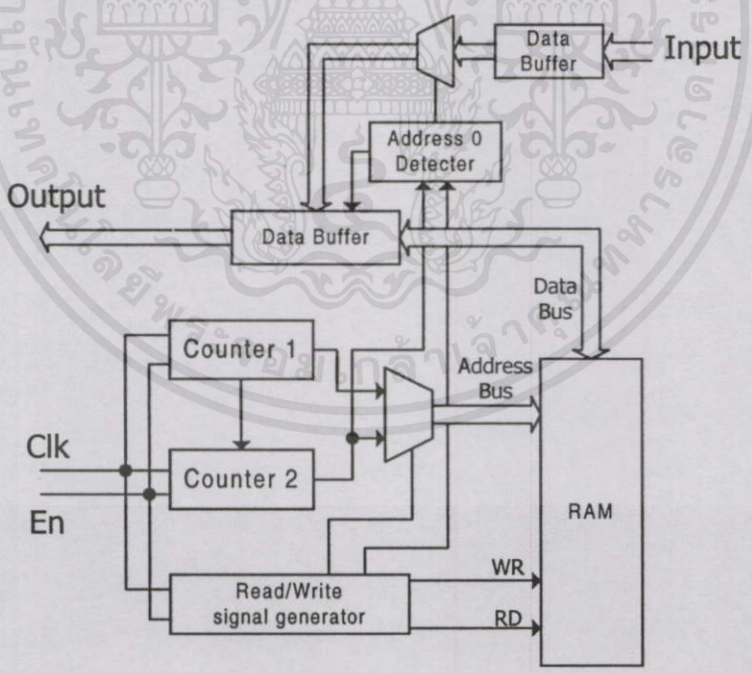
ขั้นตอนที่ 2 อ่านค่าจากหน่วยความจำ

ขั้นตอนที่ 3 กำหนดค่าของแอดเดรสด้วยค่าของ counter 2

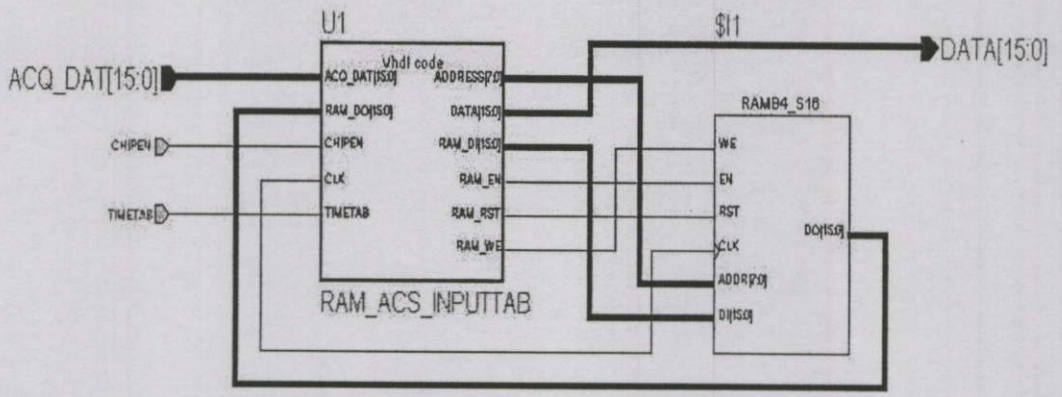
ขั้นตอนที่ 4 เขียนข้อมูลลงบนหน่วยความจำ

ขั้นตอนที่ 5 ปรับเปลี่ยนค่าของวงจรนับ (ตามเงื่อนไขในข้อที่ 3)

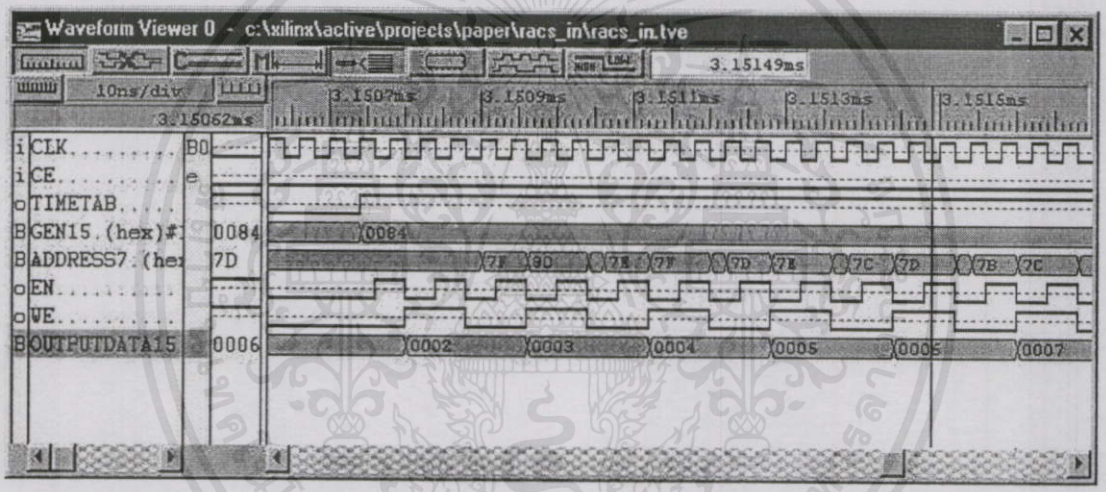
นอกจากนั้นการทำงานของวงจรจะมีกรณีเฉพาะอยู่สองกรณีคือ ในกรณีของการทำงานในวงรอบแรก (ค่าของวงจรนับทั้ง 2 ชุดมีค่าเป็น 128) เมื่อวงจรทำงานเสร็จสิ้นในวงรอบแล้ว วงจรนับชุดที่ 1 จะลดค่าลงในขณะที่วงจรนับชุดที่ 2 จะยังมีค่าคงเดิม และจากนั้นวงจรนับทั้งสองชุดจะลดค่าลงพร้อมกันเมื่อเสร็จสิ้นวงรอบที่สองเป็นต้นไป ส่วนอีกกรณีหนึ่งคือในวงรอบสุดท้าย (ค่าของวงจรนับทั้ง 2 ชุดมีค่าเป็น 0) การอ่านข้อมูลจะเปลี่ยนไปอ่านที่วงจรอินพุตบัฟเฟอร์แทนหน่วยความจำ แต่ในขั้นตอนของการเขียนข้อมูลจะยังคงขั้นตอนเดิม จากโครงสร้างการทำงานตามรูปที่ 4.16 สามารถที่จะนำมาออกแบบวงจรโดยใช้ภาษา VHDL และในรูปที่ 4.17 แสดงถึงบล็อกไดอะแกรมของวงจรและได้ทำการทดสอบการทำงานของวงจรเป็น timing diagram แสดงในรูปที่ 4.18



รูปที่ 4.16 โครงสร้างการทำงานของวงจร RAM Access ในภาคของการ shift สัญญาณอินพุต



รูปที่ 4.17 บล็อกไดอะแกรมการทำงานของวงจร RAM Access ในภาคของการ shift สัญญาณอินพุต X ,Y



รูปที่ 4.18 timing diagram การทำงานของวงจร RAM Access ในการ shift สัญญาณอินพุต X , Y

4.3.4.2 โครงสร้างการทำงานของวงจร RAM Access ในส่วนของการทำ coefficient update

จากรูปแบบการไหลของข้อมูลในรูปที่ 4.3 ทำให้สามารถออกแบบระบบควบคุมการอ่านและเขียนข้อมูลลงในหน่วยความจำได้ดังในรูปที่ 4.19 โดยระบบจะประกอบไปด้วยวงจรย่อย ๆ ที่มีฟังก์ชันการทำงานเป็นวงจรบัฟเฟอร์ของข้อมูล (Data buffer) 2 ชุด วงจรนับ 1 ชุด วงจรสร้างสัญญาณอ่านและเขียน (Read / Write signal generator) หน่วยความจำ และมีเงื่อนไขในการออกแบบคือ

1) วงจรนับ จะนับค่าตั้งแต่ 128 , 127 , ... , 2 , 1 , 0
2) ค่าข้อมูลตั้งต้นภายใน RAM จะมีค่าเป็นศูนย์

สำหรับการทำงานของวงจรในการอ่านและเขียนข้อมูลบนหน่วยความจำในแต่ละวงรอบ (cycle) จะมีอยู่ 4 ขั้นตอนคือ

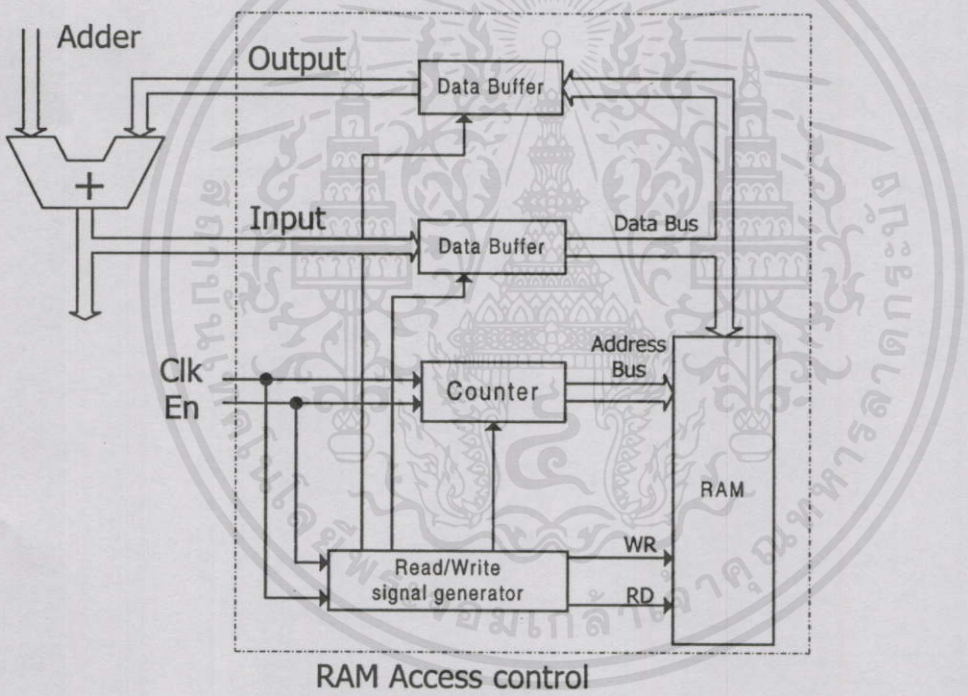
ขั้นตอนที่ 1 กำหนดค่าของแอดเดรสด้วยค่าของที่ได้จากวงจรนับ

ขั้นตอนที่ 2 อ่านค่าจากหน่วยความจำไปเก็บไว้ในวงจรบัฟเฟอร์ทางด้านเอาท์พุต

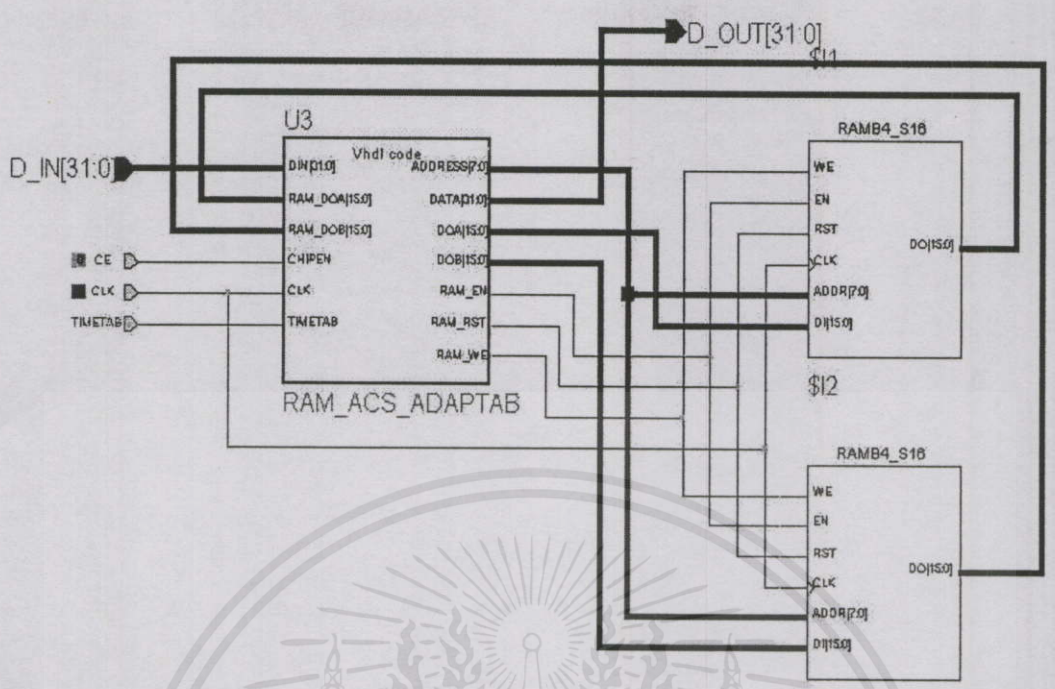
ขั้นตอนที่ 3 รับข้อมูลที่เป็นผลลัพธ์ของวงจรบวกจากวงจรบัฟเฟอร์ทางด้านอินพุต

ขั้นตอนที่ 4 เขียนข้อมูลลงบนหน่วยความจำ

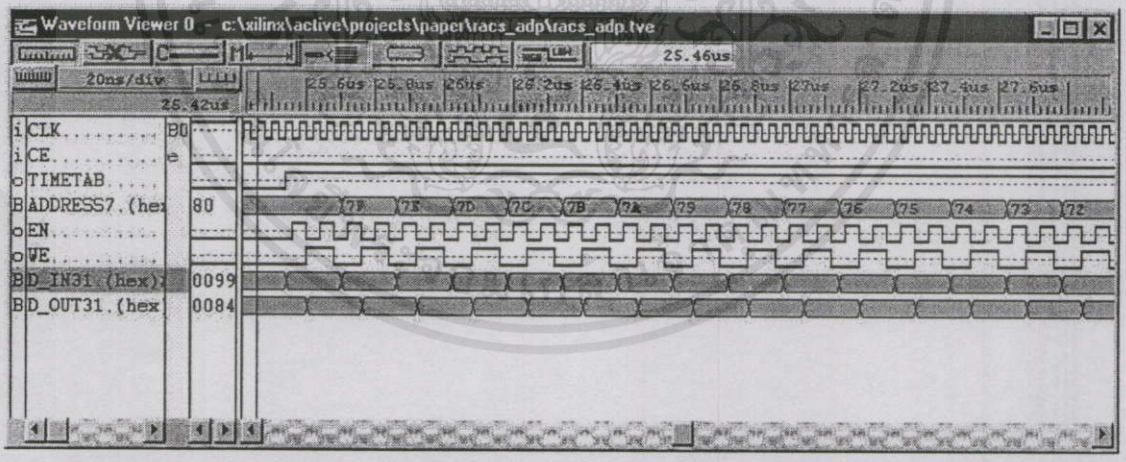
จากโครงสร้างการทำงานตามรูปที่ 4.19 สามารถที่จะนำไปออกแบบวงจรโดยใช้ภาษา VHDL ส่วนรูปที่ 4.20 แสดงถึงบล็อกไดอะแกรมของวงจร และได้ทำการทดสอบการทำงานของวงจรโดยแสดงเป็น timing diagram ดังแสดงในรูปที่ 4.21



รูปที่ 4.19 โครงสร้างการทำงานของวงจร RAM Access ในภาคของการทำ coefficient update



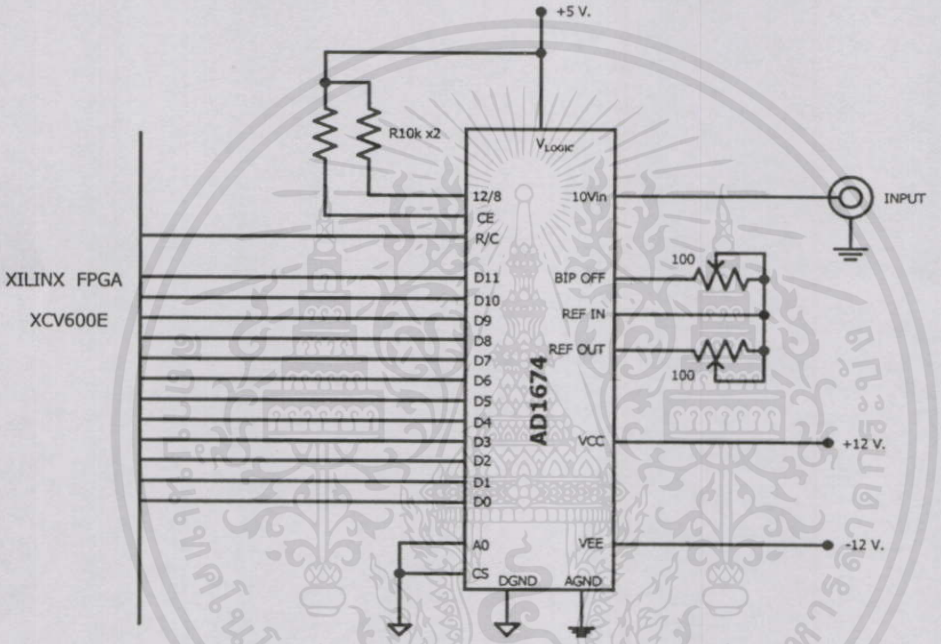
รูปที่ 4.20 บล็อกไดอะแกรมการทำงานของวงจร RAM Access ในภาคของการทำ coefficient update



รูปที่ 4.21 timing diagram การทำงานของวงจร RAM Access ในภาคของการทำ coefficient update

4.3.5 การออกแบบและโครงสร้างการทำงานของชุดวงจร Analog to Digital Converter

เนื่องจากวงจรกำจัดเสียงสะท้อนจะต้องรับข้อมูลที่เป็นสัญญาณเสียงที่มีความถี่อยู่ในย่าน 0.3 - 3.4 kHz จึงจำเป็นจะต้องมีวงจรที่ทำการแปลงสัญญาณเสียงแบบอนาลอกให้เป็นแบบดิจิตอล ในการออกแบบได้เลือกใช้ไอซี A/D ของบริษัท Analog Device เบอร์ AD1674J ซึ่งมีความแม่นยำ (accuracy) ของการแปลงสัญญาณเป็น 12 บิต มีค่าผิดพลาด Full-scale error ไม่เกิน 0.125% สามารถใช้งานได้ที่อัตราความถี่สุ่มสูงถึง 100 kHz และวงจรมีการทำงานที่ง่ายไม่ซับซ้อน ดังแสดงในรูปที่ 4.22



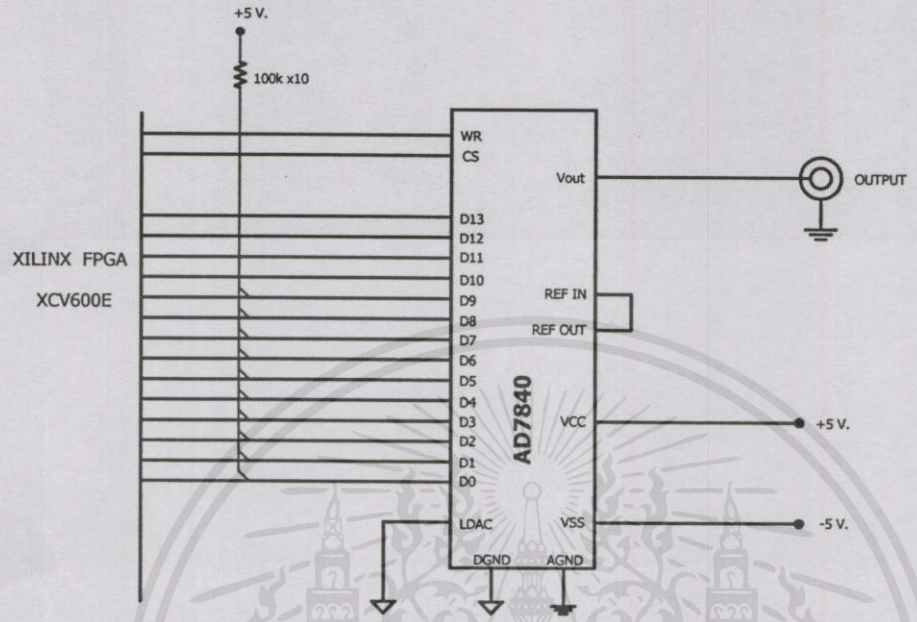
รูปที่ 4.22 วงจร A/D Converter โดยใช้ไอซีเบอร์ AD1674

ในวงจรจะใช้เพียงขาสัญญาณ R/C เท่านั้นในการควบคุมการทำงานจาก FPGA โดยขั้นตอนของการทำการแปลงสัญญาณจะเริ่มทำงานเมื่อสัญญาณมีการเปลี่ยนจาก high เป็น low (Falling edge) และสัญญาณดิจิตอลเอาท์พุตที่ได้จะถูกส่งไปยังวงจรอินพุตบัฟเฟอร์ของ FPGA โดยตรงเมื่อเสร็จสิ้นการทำงาน

4.3.6 การออกแบบและโครงสร้างการทำงานของชุดวงจร Digital to Analog Converter

วงจรการแปลงสัญญาณดิจิตอลเป็นสัญญาณอนาลอกจะทำการแปลงสัญญาณผลลัพธ์ที่ได้จากวงจรกำจัดเสียงสะท้อนที่อยู่ในรูปของสัญญาณดิจิตอลให้เป็นสัญญาณอนาลอกเพื่อป้อนเข้าสู่ช่องสัญญาณสื่อสารต่อไป ในการออกแบบได้ใช้ไอซีแปลงสัญญาณ D/A Converter เบอร์ AD7840 ของบริษัท Analog device ซึ่งมีคุณสมบัติที่สำคัญคือ มีความเร็วสูง (ใช้เวลาในการเขียนข้อมูล

เพียง 45 ns) มีความแม่นยำ (accuracy) สูงถึง 14 บิต มีค่าของลอจิกทางด้านอินพุตเป็นแบบ 2's complement และมีวงจรต่อใช้งานที่ง่ายไม่ซับซ้อนดังแสดงในรูปที่ 4.23



รูปที่ 4.23 วงจร D/A Converter โดยใช้ไอซีเบอร์ AD7840

4.3.7 การออกแบบและโครงสร้างการทำงานของชุดวงจร Input buffer และจัดระบบข้อมูล

เนื่องจากในการออกแบบระบบการประมวลผลทางคณิตศาสตร์ภายในวงจรกำจัดเสียงสะท้อนได้ใช้ระบบเลขไบนารีแบบ 2's complement เป็นหลัก ดังนั้นการจัดรูปแบบของข้อมูลทางด้านอินพุตที่ได้จากวงจรแปลงสัญญาณอนาลอกให้เป็นสัญญาณดิจิตอลโดยใช้ไอซีเบอร์ AD1674J ที่ให้สัญญาณดิจิตอลทางเอาท์พุตมีรูปแบบระบบเลขเป็น unsigned binary system จึงต้องมีการดำเนินการแก้ไขข้อมูลก่อนที่จะส่งเข้าหน่วยคำนวณ ในการออกแบบภาคอินพุตบัฟเฟอร์และวงจรแปลงระบบตัวเลขจะมีสิ่งที่ต้องพิจารณาคือคุณสมบัติการแปลงสัญญาณของไอซีเบอร์ AD1674J ดังตารางที่ 4.1 ซึ่งแสดงความสัมพันธ์ระหว่างสัญญาณอนาลอกทางด้านอินพุตแบบไบโพลาร์กับสัญญาณดิจิตอลขนาด 12 บิต เมื่อเปลี่ยนข้อมูลที่ได้ให้เป็นแบบ 2's complement แล้วจึงจะส่งข้อมูลให้กับระบบอื่นต่อไป สำหรับการแปลงรหัสของระบบเลขได้แสดงไว้ในตารางที่ 4.2 ซึ่งจะแสดงความสัมพันธ์ของข้อมูลที่ได้จากไอซีเบอร์ AD1674J กับระบบเลขที่ทำการแปลงรหัสแบบ 2's complement

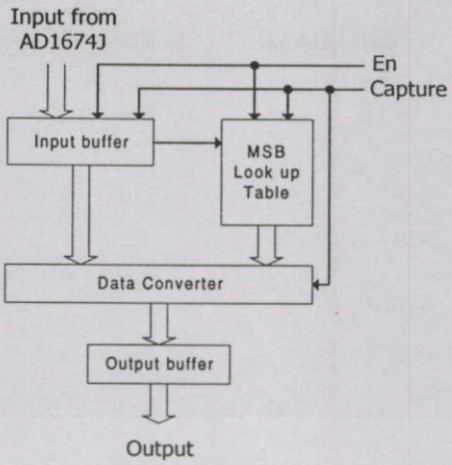
ตารางที่ 4.1 Ideal Input/Output Signal and Code table of AD1674

Analog Input, V_{in} (Bipolar mode $\pm$ 5V.)	ADC Latch Contents MSB LSB
+4.99756 V.	111111111111
+4.99512 V.	111111111110
+0.00244 V.	100000000001
0 V.	100000000000
-0.00244 V.	011111111111
-4.99756 V.	000000000001
-5.0 V.	000000000000

ตารางที่ 4.2 แสดงความสัมพันธ์ของ Unsigned binary **จาก** AD1674J **กับระบบเลข** 2's complement

Analog Input, V_{in}	AD1674J ADC Latch Contents MSB LSB	2's complement system MSB LSB
+4.99756 V.	111111111111	011111111111
+4.99512 V.	111111111110	011111111110
+0.00244 V.	100000000001	000000000001
0 V.	100000000000	000000000000
-0.00244 V.	011111111111	111111111111
-4.99756 V.	000000000001	100000000001
-5.0 V.	000000000000	100000000000

สำหรับโครงสร้างของวงจรอินพุตบัฟเฟอร์และจัดระบบข้อมูล จะประกอบไปด้วย วงจรบัฟเฟอร์ 2 ชุด ตาราง Look up table สำหรับอ้างอิงของบิต MSB และวงจรแปลงรหัสอีก 1 ชุด ซึ่งแสดงโครงสร้างไว้ในรูปที่ 4.24



รูปที่ 4.24 โครงสร้างการทำงานของภาคอินพุตบัฟเฟอร์และจัดระบบข้อมูล

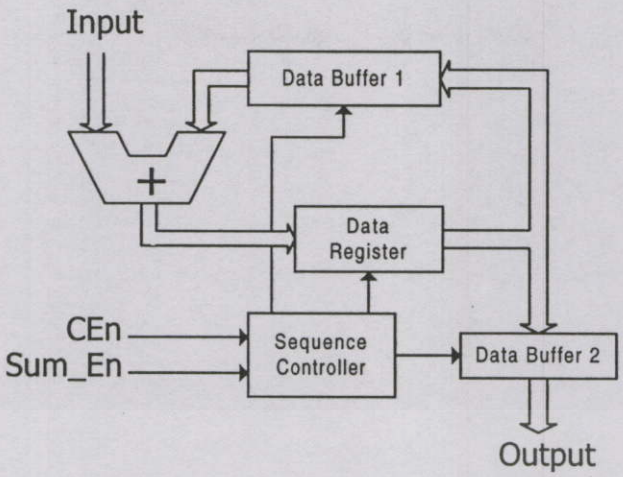
4.3.8 การออกแบบและโครงสร้างการทำงานของชุดวงจร Sum

วงจร sum จะเป็นวงจรที่ทำหน้าที่รวมข้อมูลที่ได้จากทุก ๆ องค์ประกอบทางความถี่เข้าด้วยกันเพื่อให้ได้ผลลัพธ์สุดท้ายที่เป็นการตอบสนองทางความถี่ของวงจรกรอง ดังนั้นโครงสร้างหลักของวงจร sum จึงประกอบไปด้วย วงจรบัฟเฟอร์ และวงจรบวกที่จะทำการบวกทุกครั้งเมื่อมีการส่งข้อมูลทางอินพุตเข้ามาและจะเก็บค่าของผลลัพธ์เมื่อได้ทำการรวมข้อมูลแต่ละครั้งไว้ในรีจิสเตอร์ข้อมูลเมื่อครบทั้ง 128 tap แล้วจึงให้ค่าออกไปยังเอาท์พุตโดยวงจร sequence controller จะควบคุมการทำงานให้เป็นลำดับขั้นตอนวนรอบ 3 ขั้นตอนคือ

ขั้นตอนที่ 1 เมื่อได้รับสัญญาณ CEn (Clear and Enable) วงจรจะเคลียร์ค่าในรีจิสเตอร์ให้เป็นศูนย์ จากนั้นก็จะส่งข้อมูลไปให้วงจรบัฟเฟอร์ชุดที่ 1

ขั้นตอนที่ 2 จะเป็นการตอบสนองต่อสัญญาณ Sum_En (sum enable) ซึ่งจะส่งมาทุกครั้งที่มีการส่งข้อมูลมาให้กับวงจรบวก เมื่อได้รับสัญญาณนี้แล้ววงจรรีจิสเตอร์ข้อมูลจะเก็บข้อมูลและส่งต่อให้กับบัฟเฟอร์ชุดที่ 1 วงจรจะทำงานซ้ำ ๆ ไปจนข้อมูลครบ 128 ค่า

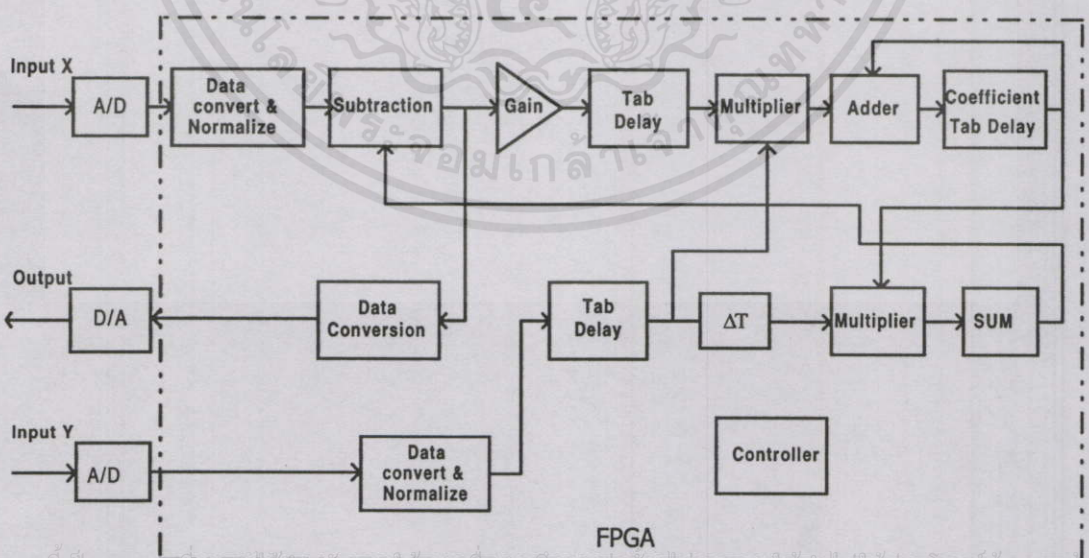
ขั้นตอนที่ 3 เป็นการส่งข้อมูลผลลัพธ์ที่ได้ให้กับวงจรบัฟเฟอร์ชุดที่ 2 เพื่อส่งออกเอาท์พุต สำหรับโครงสร้างการทำงานของวงจร sum ได้แสดงไว้ในรูปที่ 4.25



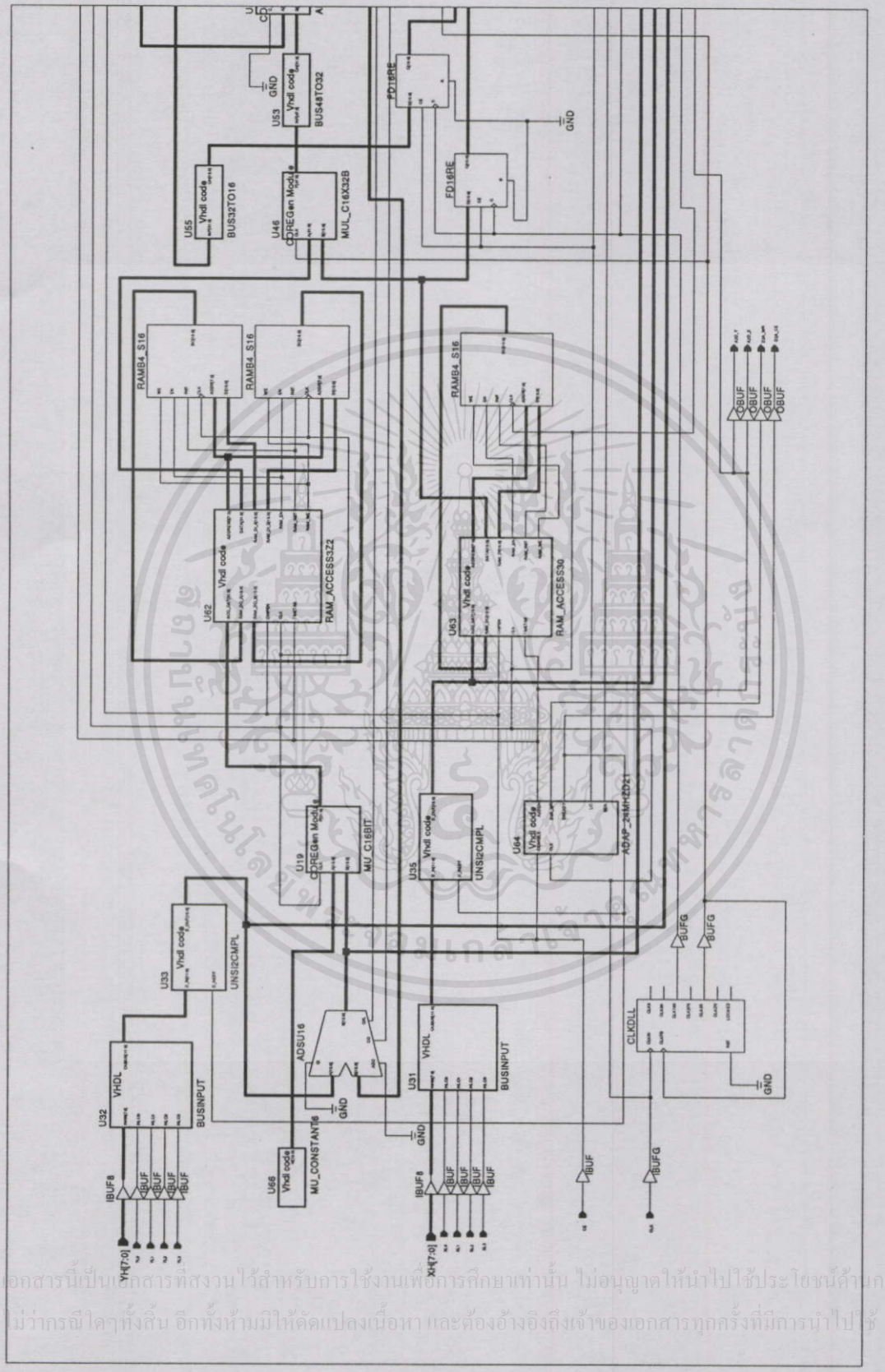
รูปที่ 4.25 โครงสร้างการทำงานของวงจร SUM

4.4 แบบแผนวงจรรวมของวงจรกำจัดเสียงสะท้อน

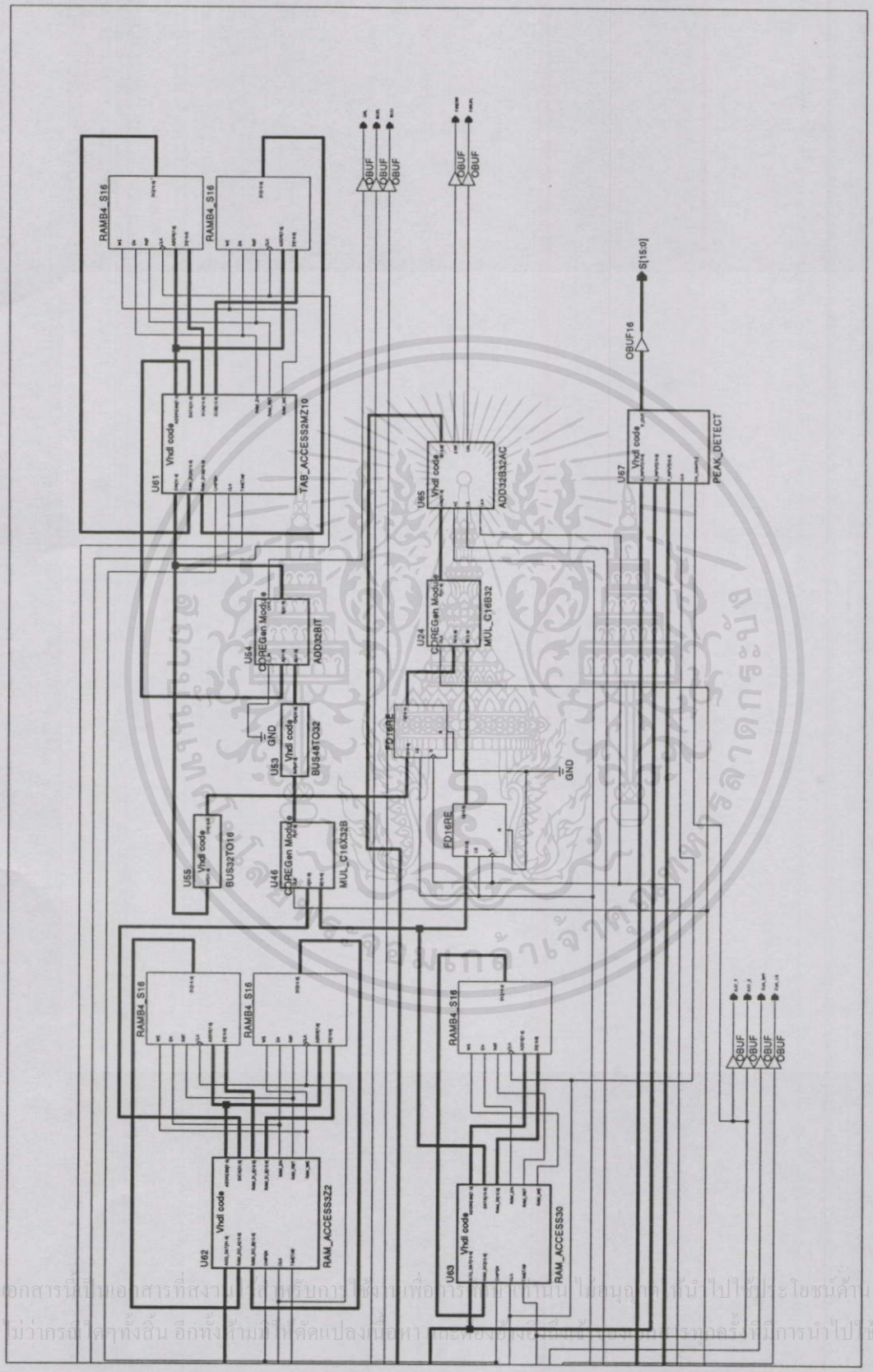
แบบแผนวงจรรวมของวงจรกำจัดเสียงสะท้อนจะได้จากการนำเอาแต่ละระบบย่อย ๆ มาเชื่อมต่อกันให้ทำงานอย่างเป็นระบบ และได้ทดลองตรวจสอบฟังก์ชันการทำงานด้วยการจำลองบนโปรแกรม Xilinx Foundation Series V3.1 เพื่อดูลำดับขั้นตอนการทำงานของระบบก่อนที่จะทำการสร้างบนฮาร์ดแวร์จริง ดังนั้นแบบแผนวงจรรวมของระบบการทำงานที่จะถูกนำไปทำคอนฟิกกูเรชันลงบนชิพ FPGA จึงแสดงได้ดังรูปที่ 4.26 และในรูปที่ 4.27 ซึ่งแสดง schematic circuit diagram ของระบบที่จะนำไปสร้างเป็นชิ้นงานจริง (implement) บน FPGA



รูปที่ 4.26 แบบแผนวงจรรวมของระบบการทำงานที่จะถูกนำไปทำคอนฟิกกูเรชันลงบนชิพ FPGA



รูปที่ 4.27 Schematic diagram ของระบบงานที่จะถูกนำไปทำคอนฟิกกูเรชันลงบนชิพ FPGA



รูปที่ 4.27 (ต่อ)

4.5 การทำ Implementation บน FPGA เบอร์ XCV600EPQ240-7

จาก shematic diagram ในรูปที่ 4.27 เมื่อนำไปทำการ Implement ลงบน FPGA เบอร์ XCV600EPQ240 ซึ่งเป็นชิพขนาด 240 ขา แต่มีขาที่เป็น I/O ให้ใช้ได้ทั้งสิ้น 158 ขา โดยจำนวนขององค์ประกอบ (element) กลุ่มต่าง ๆ ทั้ง 6 กลุ่มที่มีอยู่ในชิพและจำนวนที่ถูกใช้งานไปแสดงดังตารางที่ 4.3 ส่วนตารางที่ 4.4 แสดงถึงค่าของเวลาหน่วงที่จะเกิดขึ้นในกรณีต่าง ๆ ที่จะต้องคำนึงถึงเมื่อจะนำแบบงานนี้ไปใช้งาน

ตารางที่ 4.3 จำนวนขององค์ประกอบภายใน XCV600EPQ240 ที่ถูกใช้ในการทำ Implementation

ลำดับ	องค์ประกอบของชิพ (Element)	จำนวนที่มี	จำนวนที่ใช้	เปอร์เซ็นต์
1	External GCLKIOB	4	1	25%
2	External IOB	158	50	31%
3	BLOCKRAM	72	5	6%
4	SLICE	6912	778	11%
5	DLL	8	1	12%
6	GCLK	4	2	50%

ตารางที่ 4.4 ค่าของเวลาหน่วงที่เกิดขึ้นภายใน FPGA

Delay Summary Report	Delay
Average Connection Delay	1.716 ns
Maximum Pin Delay	6.389 ns
Average Connection Delay on the 10 Worst Nets	5.577 ns

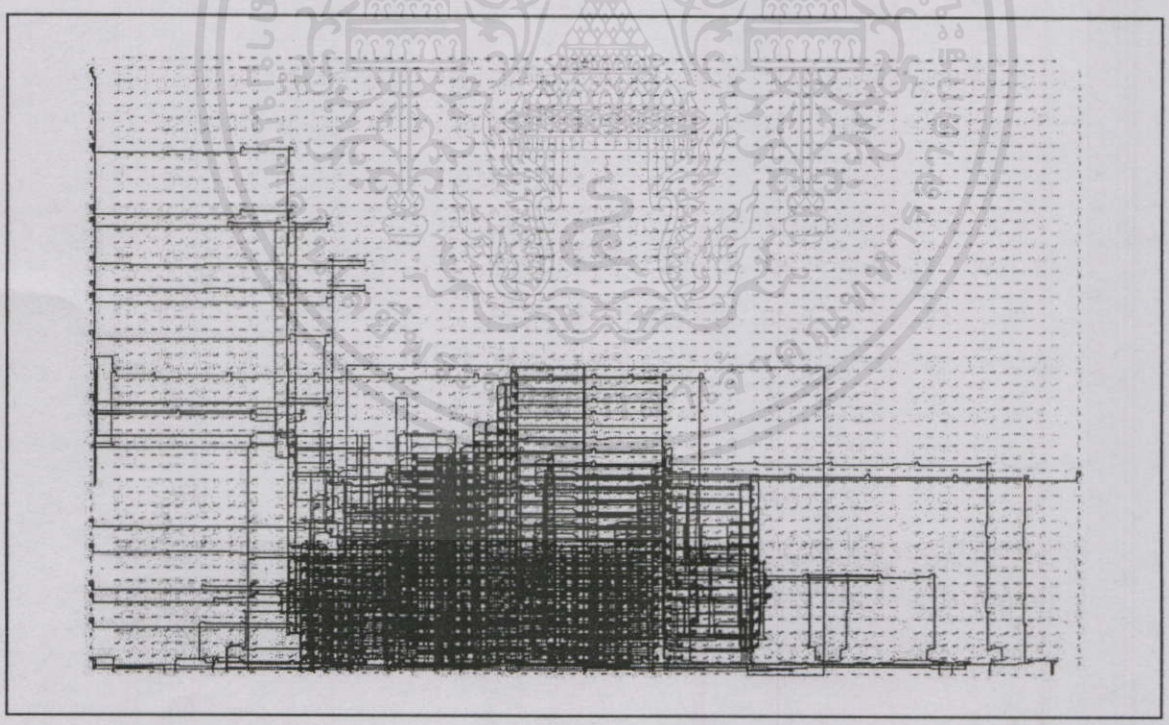
จากตารางที่ 4.3 จะพบว่ามีการใช้งานของ CLB ไป 389 บล็อกหรือ 778 slice คิดเป็น 11% ของปริมาณทั้งหมดที่มีอยู่ภายในชิพเบอร์ XCV600EPQ240 ดังนั้นถ้าหากจะเลือกใช้ชิพที่มีขนาดเล็ก เช่น เบอร์ XCV100E ซึ่งมี CLB 600 บล็อกหรือ 1200 slice ก็จะเป็นการให้ประสิทธิภาพในการใช้พื้นที่เพิ่มขึ้นเป็น 64 % ทำให้ต้นทุนในการสร้างถูกลงได้มากกว่าหลายเท่าด้วย

จากตารางที่ 4.4 เป็นข้อมูลที่เกี่ยวกับค่าของเวลาหน่วงที่ได้จากการทำ Implement บนชิพที่ได้กำหนดในตอนต้น ซึ่งพบว่าจะมีค่าของเวลาหน่วงของเส้นทางการเชื่อมต่อข้อมูล (Net) ในเส้นทางที่ยาวที่สุดภายในชิพ (เมื่อพิจารณาโดยเฉลี่ย) ประมาณ 5.577 ns และมีค่าเวลาหน่วงของขา I/O ประมาณ 6.389 ns และมีค่าเฉลี่ยของเวลาหน่วงที่ใช้การเชื่อมต่อระหว่าง slice ประมาณ 1.716 ns

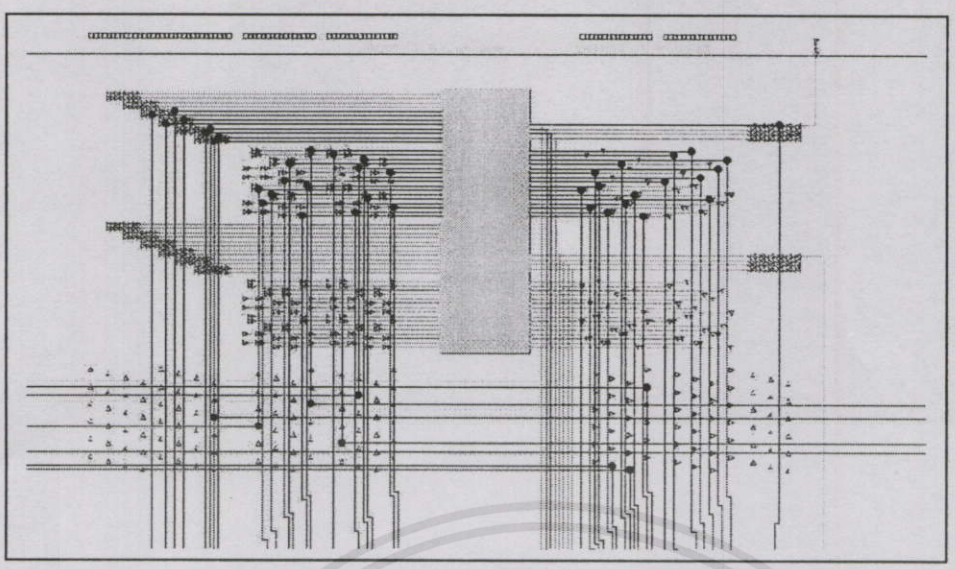
ส่วนตารางที่ 4.5 แสดงค่าของเวลาหน่วงสูงสุดของระบบที่เกิดขึ้นเป็น 21.081 ns ซึ่งเป็นการตอบสนองเพื่อให้เกิดการส่งผ่านของข้อมูลภายในชิพไปเป็นสัญญาณเอาท์พุตที่จะต้องใช้เวลาที่มากที่สุดภายหลังจากการเปลี่ยนแปลงของสัญญาณนาฬิกาของระบบ (รวมค่าของเวลาหน่วงที่ขาของชิพด้วย) แต่อย่างไรก็ตามจากค่าของคาบเวลาที่ใช้ในการทำงานน้อยที่สุดเป็น 19.470 ns ทำให้ระบบสามารถทำงานที่ความถี่ได้สูงถึง 51.361 MHz และในรูปที่ 4.28 แสดงให้เห็นถึงโครงสร้างการเชื่อมต่อภายในระหว่าง CLB ต่าง ๆ ส่วนรูปที่ 4.29 เป็นการเชื่อมต่อของสายเชื่อมต่อของ RAM Block และรูปที่ 4.30 แสดงเครื่องต้นแบบของตัวกำจัดสัญญาณเสียงสะท้อน

ตารางที่ 4.5 ค่าของเวลาหน่วงและความเร็วสูงสุดที่ระบบสามารถทำงานได้

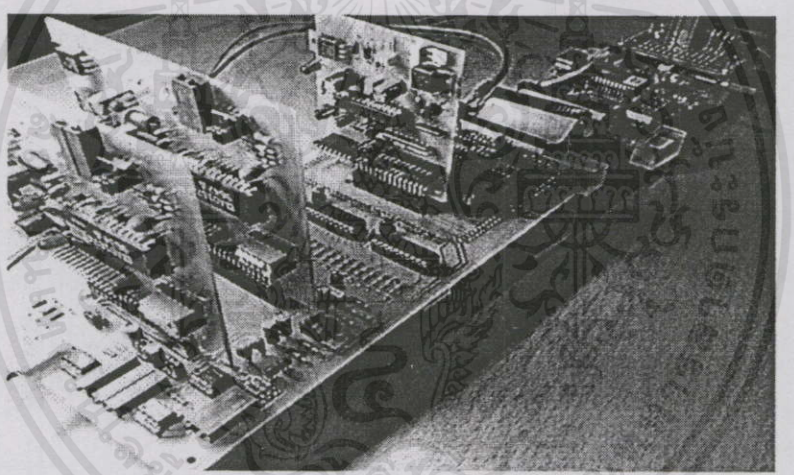
Timing constraint:	
Minimum period:	19.470ns (Maximum frequency: 51.361MHz)
Maximum combinational path delay:	21.081ns
Maximum net delay:	6.389ns



รูปที่ 4.28 โครงสร้างการเชื่อมต่อภายในระหว่าง CLB



รูปที่ 4.29 การเชื่อมต่อของสายเชื่อมต่อ (Net) กับหน่วยความจำ RAM Block



รูปที่ 4.30 ต้นแบบของตัวกำจัดสัญญาณเสียงสะท้อนที่ใช้ FPGA

4.6 สรุป

การออกแบบและสร้างเครื่องกำจัดเสียงสะท้อนโดยใช้ FPGA เป็นอุปกรณ์หลักนั้นได้ออกแบบโดยแบ่งระบบงานที่มีขนาดใหญ่ให้กลายเป็นระบบงานขนาดเล็กเชื่อมต่อการทำงานร่วมกัน ขนาดของระบบงานจะถูกแยกย่อยลงจนมีขนาดเพียงพอที่ผู้ออกแบบสามารถวิเคราะห์ได้โดยง่าย ก่อนที่จะทำการออกแบบและทดสอบฟังก์ชันนั้นโดยใช้เครื่องมือต่าง ๆ เช่น ภาษา VHDL และ CAD (schematic diagram) เป็นต้น เมื่อได้ระบบงานที่ตรงตามต้องการแล้วจึงนำไฟล์ข้อมูลภาษา VHDL และ schematic diagram นั้นไปแปลโดยวิธีการคอมไพล์ให้เป็นไฟล์รหัสข้อมูลสำหรับการทำคอนฟิกกูเรชันที่เหมาะสมกับชิพ FPGA แต่ละเบอร์ จึงทำให้สามารถที่จะออกแบบระบบงานที่ซับซ้อนและสามารถที่จะเลือกเทคโนโลยีที่จะทำการสร้างเป็นชิ้นงานจริงได้อย่างเหมาะสม

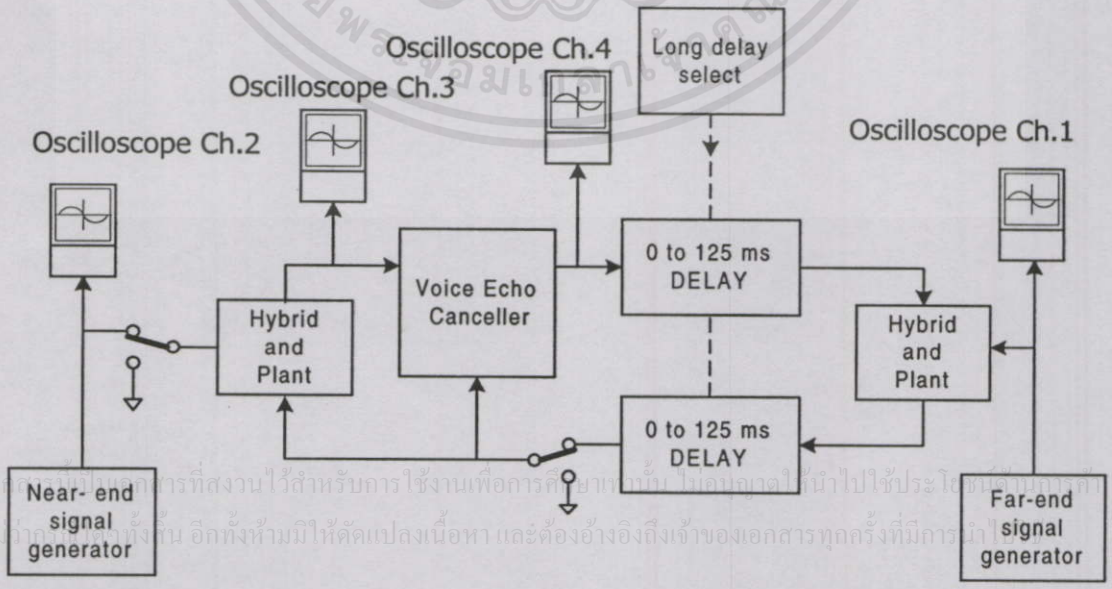
บทที่ 5
การทดสอบการทำงานของตัวกำจัดสัญญาณเสียงสะท้อน

5.1 บทนำ

การทดสอบการทำงานเครื่องต้นแบบของตัวกำจัดสัญญาณเสียงสะท้อนได้ดำเนินการในห้องปฏิบัติการโดยการกำหนดค่าพารามิเตอร์และทำการตรวจวัดผลลัพธ์ที่ได้จากนั้นก็นำไปคำนวณหาขนาดของการกำจัดสัญญาณเสียงสะท้อนของเครื่องต้นแบบ

5.2 วงจรทดสอบ

วงจรที่ใช้ในการทดสอบแสดงไว้ในรูปที่ 5.1 โดยในวงจรได้ใช้วงจรเอ็คโค่เจนเนอเรเตอร์ (echo generator) มาทำหน้าที่เป็นตัวกำหนดค่าของ long delay ในระบบเป็น 125 ms และใช้ออสซิลโลสโคปวัดรูปร่างและขนาดของสัญญาณที่จุดต่าง ๆ โดยออสซิลโลสโคปช่องสัญญาณที่ 1 วัดสัญญาณทางด้าน far-end ออสซิลโลสโคปช่องสัญญาณที่ 2 วัดสัญญาณทางด้าน near-end ออสซิลโลสโคปช่องสัญญาณที่ 3 วัดสัญญาณทางด้านอินพุตของตัวกำจัดสัญญาณเสียงสะท้อน และออสซิลโลสโคปช่องสัญญาณที่ 4 วัดสัญญาณเอาท์พุตของตัวกำจัดสัญญาณเสียงสะท้อน ส่วนวงจรไฮบริดของระบบได้ใช้วงจรซัม (sum) ซึ่งมีค่าของเวลาหน่วงของผู้พูดด้านใกล้ (Near-end delay) เป็น 0 ms แทนตัวไฮบริด และมีค่าของการรั่วไหลของสัญญาณเอ็คโค่ลีค (echo leak) น้อยกว่า –6 dB (ตามมาตราฐาน ITU-T G.165)



รูปที่ 5.1 วงจรทดสอบของตัวกำจัดสัญญาณเสียงสะท้อน

5.3 ขั้นตอนการทดสอบ

ในการทดสอบการทำงานของเครื่องกำจัดสัญญาณเสียงสะท้อนได้แบ่งขั้นตอนการทดสอบออกเป็นขั้นตอนต่างๆ คือ

1) ต่อวงจรทดสอบตามรูปที่ 5.1

2) ป้อนสัญญาณทดสอบให้กับระบบทั้งด้านที่เป็นผู้พูดด้านใกล้ และผู้พูดด้านไกล โดยมีเงื่อนไขการทดสอบคือ

- เมื่อมีสัญญาณของผู้พูดด้านไกลเพียงด้านเดียว
- เมื่อมีสัญญาณของผู้พูดด้านใกล้เพียงด้านเดียว
- เมื่อมีสัญญาณของผู้พูดด้านใกล้ และด้านไกลพร้อมกัน

3) วัดค่าของสัญญาณที่จุดต่าง ๆ ด้วยออสซิลโลสโคป

4) บันทึกและวิเคราะห์ผลการทดสอบ

5) สรุปผลการทดสอบ

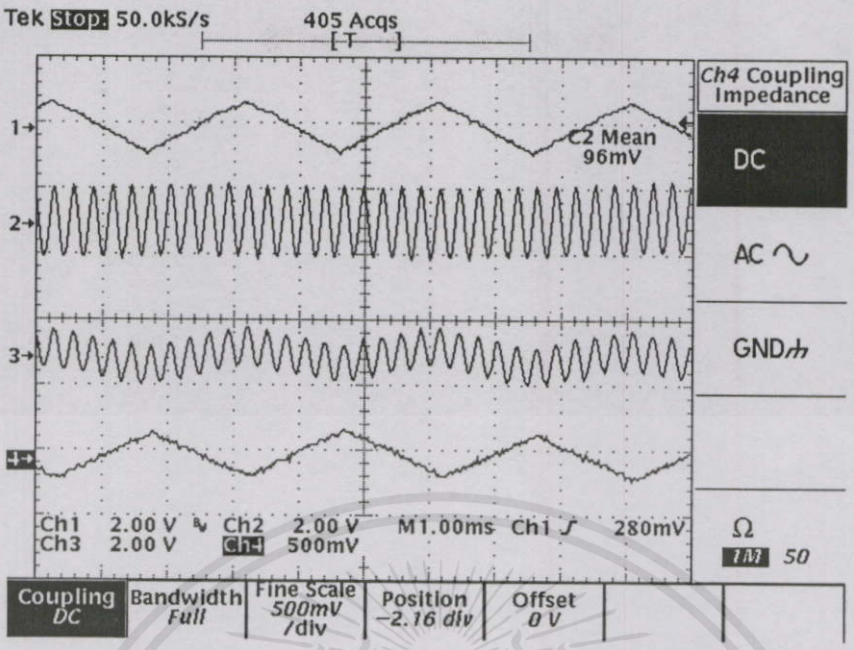
5.4 ผลการทดสอบ

การทดสอบการทำงานของตัวประมาณค่าสัญญาณสะท้อนได้ทดสอบโดยการป้อนสัญญาณทดสอบเป็นรูปสามเหลี่ยมให้กับทางด้านอินพุตของตัวกำจัดสัญญาณเสียงสะท้อนและวัดสัญญาณที่เอาท์พุตที่ได้จากตัวกรองปรับตัวได้ก่อนที่จะป้อนเข้าสู่วงจร sum ผลการวัดรูปร่างสัญญาณแสดงได้ดังรูปที่ 5.2

การทดสอบเมื่อมีสัญญาณของผู้พูดทางด้านไกลเพียงด้านเดียว ในการทดสอบได้ทำการป้อนสัญญาณความถี่เสียงรูปไซน์ สามเหลี่ยม และสี่เหลี่ยมแทนสัญญาณของผู้พูดทางด้านไกล และใช้เครื่องออสซิลโลสโคปวัดสัญญาณตามจุดต่าง ๆ ตามรูปที่ 5.1 ได้ผลการวัดดังแสดงไว้ในรูปที่ 5.3 ถึง 5.7

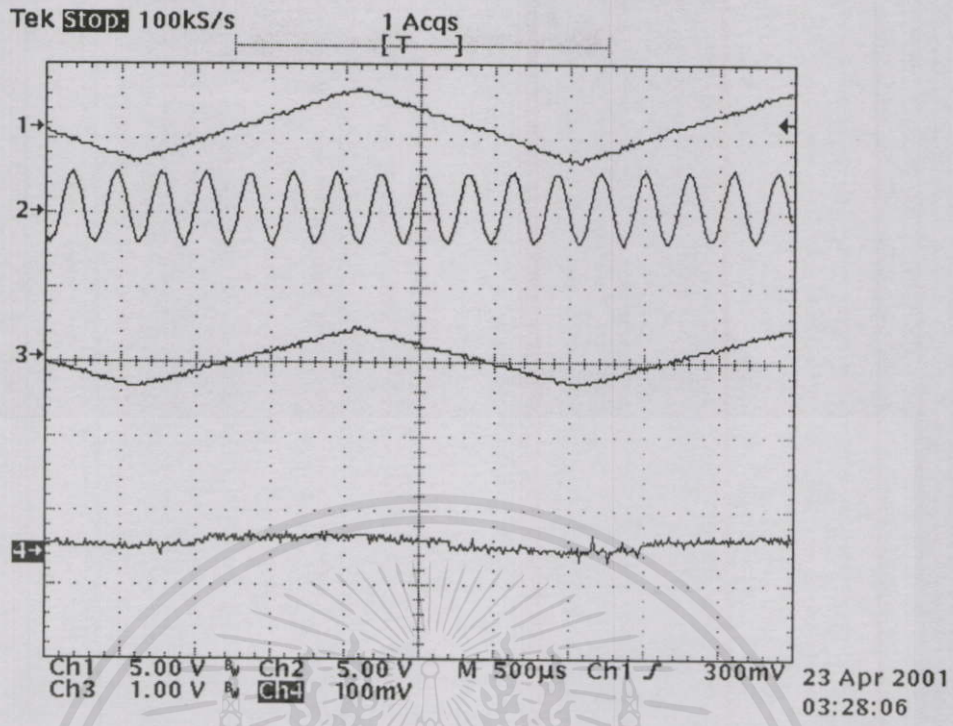
การทดสอบเมื่อมีสัญญาณของผู้พูดทางด้านใกล้เพียงด้านดียว ในการทดสอบได้ทำการป้อนสัญญาณความถี่เสียงแทนสัญญาณของผู้พูดทางด้านใกล้ และใช้เครื่องออสซิลโลสโคปวัดสัญญาณตามจุดต่าง ๆ ตามรูปที่ 5.1 ได้ผลการวัดดังแสดงไว้ในรูปที่ 5.8

การทดสอบเมื่อมีสัญญาณของผู้พูดทางด้านไกลเกิดขึ้นพร้อมกันกับการมีสัญญาณของผู้พูดทางด้านใกล้ ในการทดสอบได้ทำการป้อนสัญญาณความถี่เสียงรูปไซน์แทนสัญญาณของผู้พูดทางด้านใกล้และสัญญาณรูปสามเหลี่ยมแทนสัญญาณของผู้พูดทางด้านไกล ใช้เครื่องออสซิลโลสโคปวัดสัญญาณตามจุดต่าง ๆ โดยช่องสัญญาณที่ 1 วัดสัญญาณทางด้านไกล ผลการทดสอบแสดงไว้ในรูปที่ 5.9

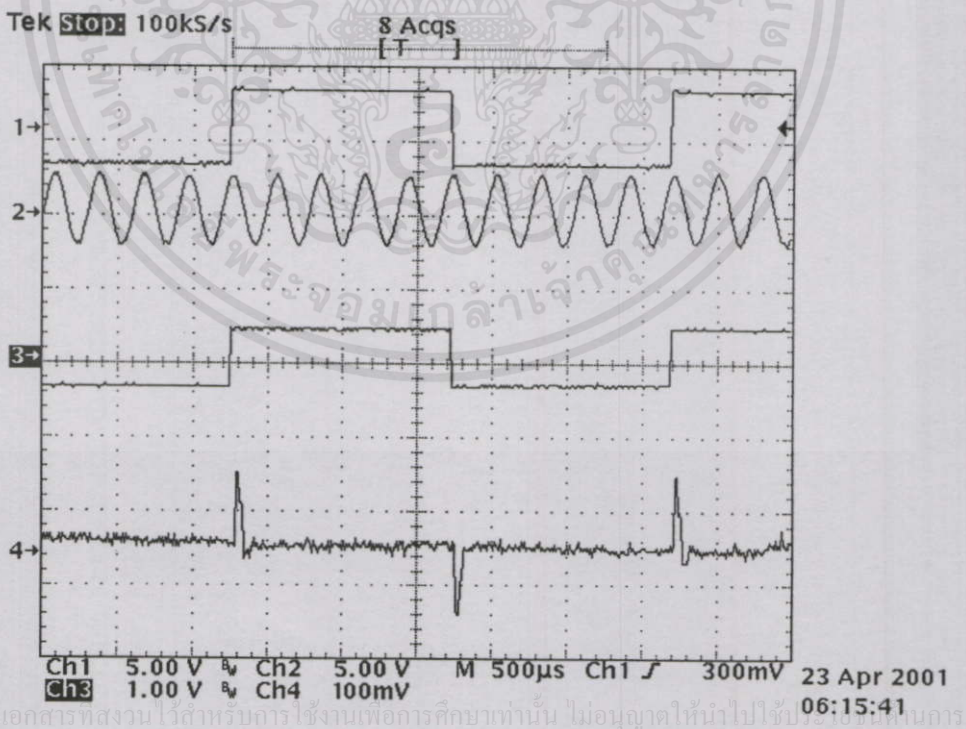


รูปที่ 5.2 รูปคลื่นที่ได้จากการ synthesis ของตัวกรองปรับตัวได้ (ch.4) เมื่อสัญญาณอินพุตเป็นรูปสามเหลี่ยม (ch.2)

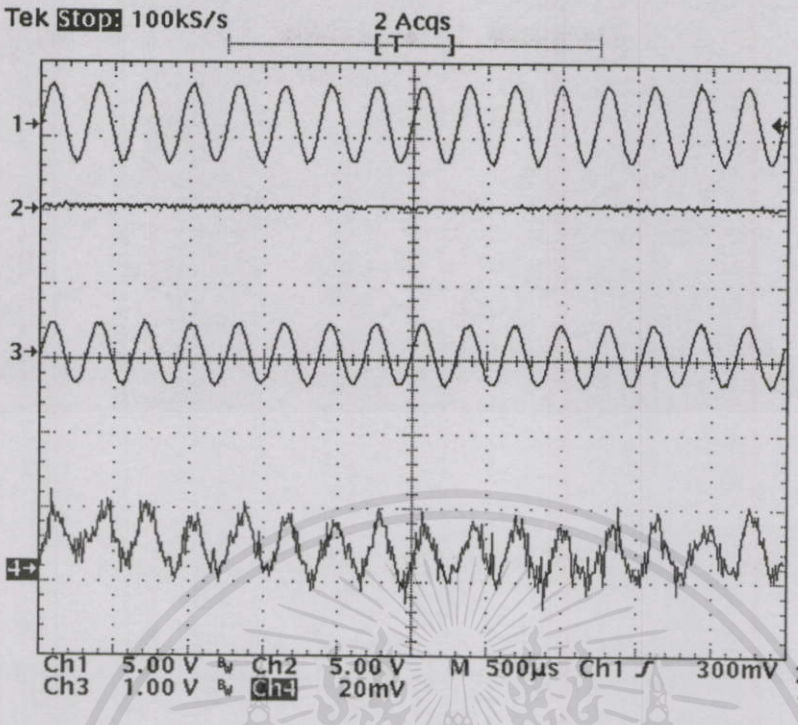
จากรูปที่ 5.2 เมื่อพิจารณารูปคลื่นในช่องสัญญาณที่ 4 ของออสซิลโลสโคปซึ่งเป็นสัญญาณที่ได้จากการสังเคราะห์ขึ้นมาของตัวกรองปรับตัวได้เพื่อเลียนแบบสัญญาณอ้างอิง (ช่องสัญญาณที่ 1 ของออสซิลโลสโคป) ที่ป้อนทางด้านอินพุต จะพบว่าสัญญาณที่ได้จากการสังเคราะห์มีรูปร่างของสัญญาณที่เหมือนกับสัญญาณอ้างอิงแต่จะมีการเลื่อน (shift) ของเฟส (phase) ไปประมาณ 175 องศาจากสัญญาณอ้างอิง ซึ่งเมื่อนำสัญญาณทั้งสองไปป้อนให้กับวงจร sum ก็จะเกิดการหักล้างกัน แต่ด้วยการที่มีการเลื่อนของเฟสไปไม่ถึง 180 องศา จึงได้ทำให้เกิดค่าของเศษที่เหลือตกค้าง (residual) อยู่บางส่วน แต่ก็ไม่ได้มีผลกระทบใด ๆ ในช่วงของความถี่ที่ใช้งานของเครื่องต้นแบบ



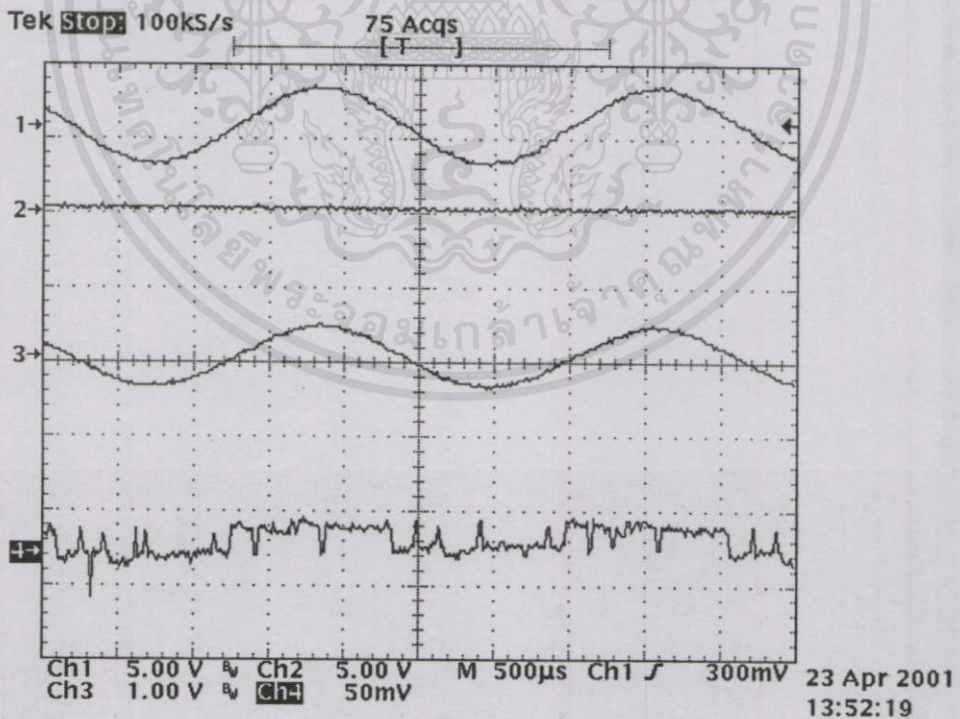
รูปที่ 5.3 ผลการทดสอบเมื่อสัญญาณของผู้พูดทางด้านไกลเป็นรูปสามเหลี่ยมความถี่ 340 Hz



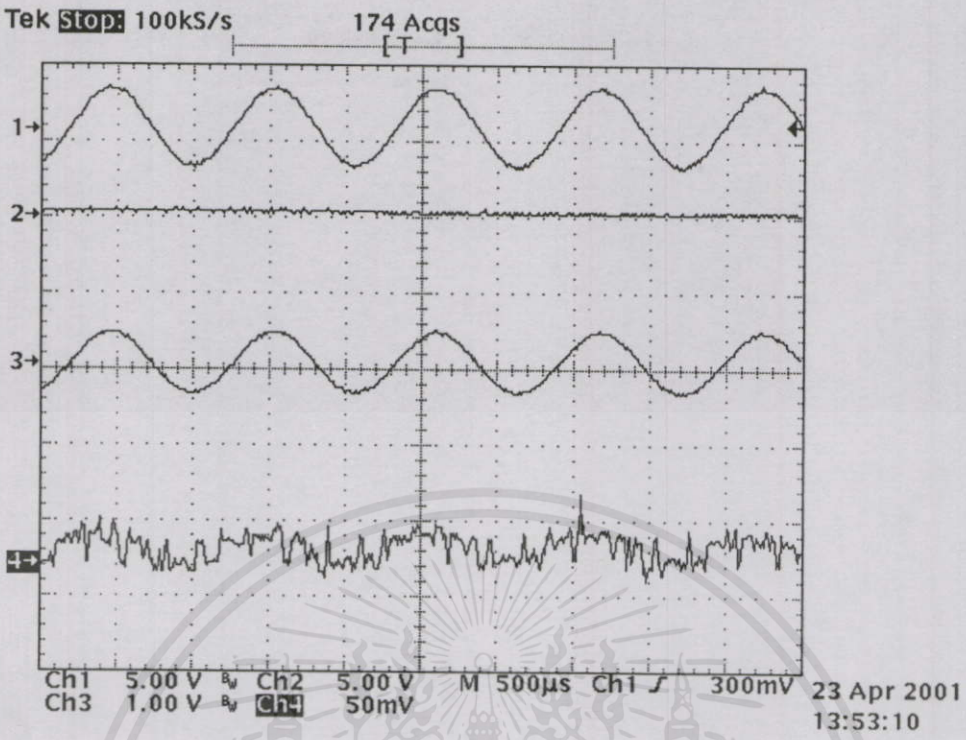
รูปที่ 5.4 ผลการทดสอบเมื่อสัญญาณของผู้พูดทางด้านไกลเป็นรูปสี่เหลี่ยมความถี่ 340 Hz



รูปที่ 5.5 ผลการทดสอบเมื่อสัญญาณของผู้พูดทางด้านไกลเป็นรูปไซน์ความถี่ 3.4 kHz



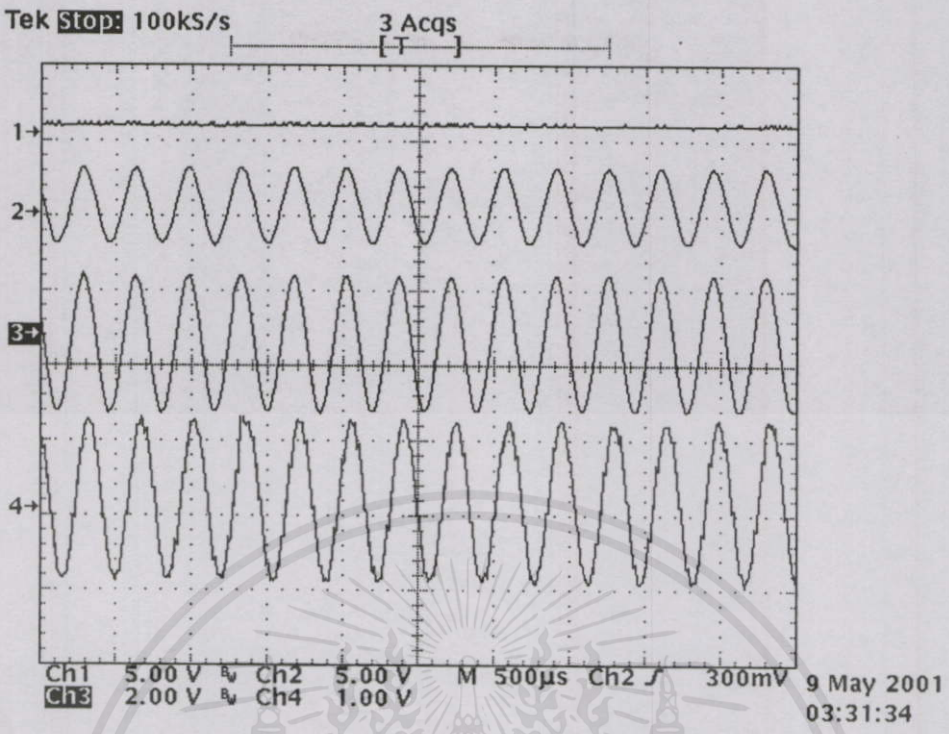
รูปที่ 5.6 ผลการทดสอบเมื่อสัญญาณของผู้พูดทางด้านไกลเป็นรูปไซน์ความถี่ 440 Hz



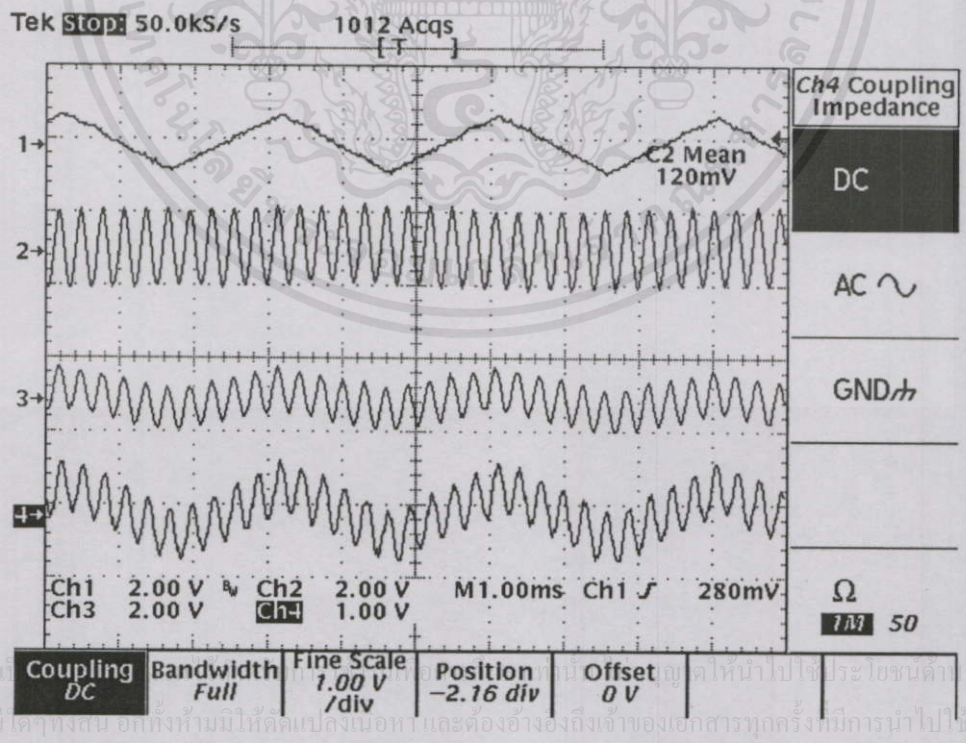
รูปที่ 5.7 ผลการทดสอบเมื่อสัญญาณของผู้พูดทางด้านไกลเป็นรูปไซน์ความถี่ 1.3 kHz

จากรูปที่ 5.3 ถึงรูปที่ 5.7 ซึ่งเป็นการทดสอบการกำจัดสัญญาณเสียงสะท้อนในกรณีที่มีเพียงสัญญาณของผู้พูดด้านไกลเท่านั้น เมื่อทดสอบด้วยการป้อนสัญญาณรูปสี่เหลี่ยมจะพบว่าผลของการกำจัดสัญญาณสะท้อนก่อให้เกิดเศษที่เหลืออยู่ของสัญญาณในขณะที่มีการเปลี่ยนขอบของสัญญาณอินพุตเท่านั้น แต่เมื่อทดสอบด้วยการป้อนสัญญาณอินพุตด้วยรูปคลื่นที่มีการเปลี่ยนแปลงตลอดเวลาดังเช่นรูปไซน์และรูปสามเหลี่ยม จะพบว่ามีค่าของเศษที่เหลืออยู่เป็นสัญญาณเดียวกันกับทางด้านอินพุตแต่มีขนาดที่เล็กกว่าประมาณ 33 –34 dB เมื่อเทียบกับทางด้านของอินพุต

จากผลการทดสอบในรูปที่ 5.8 จะพบว่าเมื่อมีการป้อนสัญญาณเพียงด้านผู้พูดทางใกล้เพียงด้านเดียว ตัวกำจัดสัญญาณเสียงสะท้อนจะยอมให้มีการส่งผ่านสัญญาณทางอินพุตทั้งหมดไปยังด้านเอาท์พุตโดยไม่มีการดำเนินการใด ๆ รวมทั้งในกรณีที่เกิด double talk ดังในรูปที่ 5.9 ซึ่งมีสัญญาณพร้อมกันทั้งสองด้าน ตัวกำจัดสัญญาณเสียงสะท้อนก็จะทำการส่งผ่านสัญญาณทั้งหมดไปยังด้านเอาท์พุตและไม่ทำให้สัญญาณผิดเพี้ยนไปจากเดิมมากนักเช่นกัน



รูปที่ 5.8 ผลการทดสอบเมื่อสัญญาณของผู้พูดทางด้านใกล้เพียงด้านเดียว เมื่อสัญญาณมีความถี่ 2.8 KHz



รูปที่ 5.9 ผลการทดสอบเมื่อสัญญาณของผู้พูดทางด้านไกลเป็นรูปสามเหลี่ยมความถี่ 340 Hz และสัญญาณของผู้พูดทางด้านใกล้เป็นรูปไซน์ความถี่ 3.4 KHz

จากการทดสอบตามวงจรในรูปที่ 5.1 ได้ทำการวัดค่าของการกำจัดสัญญาณเสียงสะท้อนด้วยการป้อนสัญญาณทดสอบรูปไซน์แทนสัญญาณของผู้พูดทางด้านไกลเพียงด้านเดียวให้กับเครื่องกำจัดสัญญาณเสียงสะท้อนและทำการเปลี่ยนค่าของระดับสัญญาณตั้งแต่ 0.5 Vp. ไปจนถึง 2.0 Vp. และเปลี่ยนค่าความถี่ตั้งแต่ 340 Hz ไปจนถึง 3400 Hz จากนั้นทำการวัดค่าของเศษที่เหลืออยู่ของสัญญาณเสียงสะท้อนที่ทางด้านเอาท์พุตจำนวน 100 ตัวอย่างทดสอบ ดังแสดงในตารางที่ 5.1 และนำมาคำนวณหาอัตราการกำจัดสัญญาณสะท้อน (echo cancellation) ด้วยสมการ (5.1) ส่วนในตารางที่ 5.2 จะแสดงถึงจำนวนของอัตราการกำจัดสัญญาณสะท้อนที่ได้จากการทดสอบ และรูปที่ 5.10 จะแสดงกราฟของอัตราการกำจัดสัญญาณเสียงสะท้อนที่ได้จากตารางที่ 5.2

ตารางที่ 5.1 แสดงผลการทดสอบการกำจัดสัญญาณเสียงสะท้อน

Vin	Vout	ความถี่ (Hz)									
(Vp)	(mVp)	340	450	900	1200	1500	1800	2500	2800	3000	3400
0.5	Vout(mVp)	10	10	10	11	10	11	11	11	10	10
	Cancellation(dB)	-33.98	-33.98	-33.98	-33.15	-33.98	-33.15	-33.15	-33.15	-33.98	-33.98
0.6	Vout(mVp)	12	11	12	12	14	12	13	12	12	13
	Cancellation(dB)	-33.98	-34.74	-33.98	-33.98	-32.64	-33.98	-33.28	-33.98	-33.98	-33.28
0.7	Vout(mVp)	13	13	14	15	15	15	14	14	16	16
	Cancellation(dB)	-34.62	-34.62	-33.98	-33.38	-33.38	-33.38	-33.98	-33.98	-32.82	-32.82
0.8	Vout(mVp)	16	18	15	16	17	18	18	15	16	15
	Cancellation(dB)	-33.98	-32.96	-34.54	-33.98	-33.45	-32.96	-32.96	-34.54	-33.98	-34.54
0.9	Vout(mVp)	18	20	20	19	21	19	17	20	20	20
	Cancellation(dB)	-33.98	-33.06	-33.06	-33.51	-32.64	-33.51	-34.48	-33.06	-33.06	-33.06
1.0	Vout(mVp)	19	22	20	21	22	20	19	21	22	20
	Cancellation(dB)	-34.42	-33.15	-33.98	-33.56	-33.15	-33.98	-34.42	-33.56	-33.15	-33.98
1.3	Vout(mVp)	29	27	26	26	30	26	25	26	30	29
	Cancellation(dB)	-33.03	-33.65	-33.98	-33.98	-32.74	-33.98	-34.32	-33.98	-32.74	-33.03
1.5	Vout(mVp)	32	30	32	29	29	34	29	32	33	32
	Cancellation(dB)	-33.42	-33.98	-33.42	-34.27	-34.27	-32.89	-34.27	-33.42	-33.15	-33.42
1.8	Vout(mVp)	35	37	38	41	40	36	34	36	35	38
	Cancellation(dB)	-34.22	-33.74	-33.51	-32.85	-33.06	-33.98	-34.48	-33.98	-34.22	-33.51
2.0	Vout(mVp)	41	40	38	39	42	45	44	42	40	43
	Cancellation(dB)	-33.76	-33.98	-34.42	-34.20	-33.56	-32.96	-33.15	-33.56	-33.98	-33.35

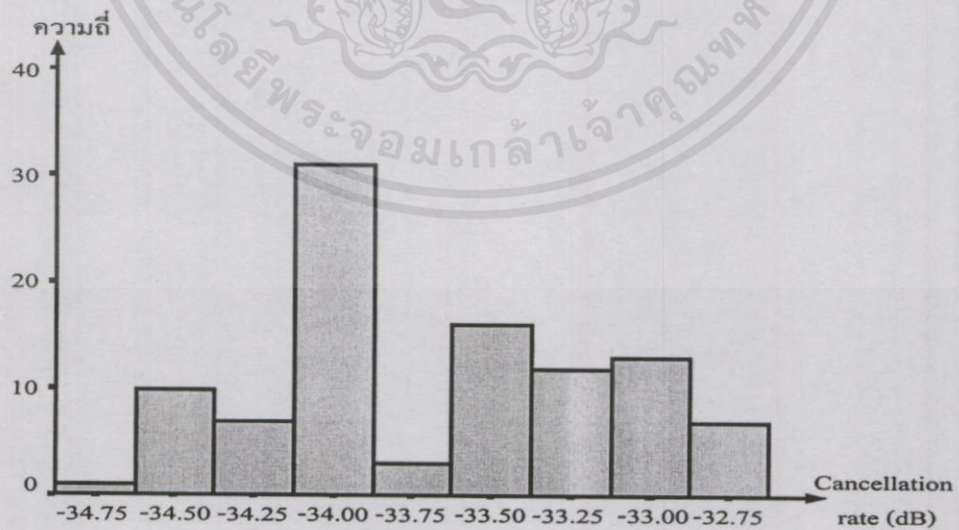
$$cancellation\ (dB) = 20 \log \frac{Vout}{Vin} \qquad (5.1)$$

เมื่อกำหนดให้

cancellation	:	ความสามารถในการกำจัดสัญญาณสะท้อน (dB)
Vout	:	สัญญาณทางด้านเอ้าพุทของตัวกำจัดสัญญาณเสียงสะท้อน (Vp.)
Vin	:	สัญญาณทางด้านอินพุทของตัวกำจัดสัญญาณเสียงสะท้อน (Vp.)

ตารางที่ 5.2 แสดงช่วงผลการทดสอบการกำจัดสัญญาณเสียงสะท้อน

Cancellation rate (dB)	ความถี่ที่พบ
น้อยกว่า -34.62	1
[-34.62 , -34.37)	10
[-34.37 , -34.12)	7
[-34.12 , -33.87)	31
[-33.87 , -33.62)	3
[-33.62 , -33.37)	16
[-33.37 , -33.12)	12
[-33.12 , -32.87)	13
มากกว่า -32.87	7
รวม	100



รูปที่ 5.10 กราฟแสดงอัตราการกำจัดสัญญาณเสียงสะท้อน

5.5 สรุปผลการทดสอบ

จากการทดสอบการทำงานของเครื่องต้นแบบของเครื่องกำจัดสัญญาณเสียงสะท้อนโดยใช้ FPGA ได้แสดงให้เห็นถึงความสามารถในการกำจัดสัญญาณเสียงสะท้อนได้ประมาณ 34 dB และสามารถทำงานได้ตลอดทั้งย่านความถี่ 340 – 3400 Hz และนอกจากนั้นการใช้ความถี่ของสัญญาณสุ่มที่สูงถึง 40 kHz ทำให้ได้ให้สัญญาณทางเอาท์พุตที่มีความผิดเพี้ยนน้อยเมื่อเทียบกับสัญญาณทางด้านอินพุตในกรณีเมื่อเกิด double talk

บทที่ 6
สรุปผลการวิจัยและข้อเสนอแนะ

6.1 สรุปผลการวิจัย

ในการวิจัยนี้ได้ทำการออกแบบสร้างเครื่องกำจัดสัญญาณเสียงสะท้อนโดยใช้ FPGA ซึ่งออกแบบสร้างให้ระบบทำงานได้ในอัตราความถี่สุ่มของสัญญาณทางด้านอินพุตและเอาท์พุตมีค่าที่สูงถึง 40 kHz และอัตราความถี่สุ่มเป็นอิสระจากสัญญาณนาฬิกาของหน่วยประมวลผลข้อมูล ในส่วนของการประมวลผลข้อมูลที่เป็นอะแดปทีฟฟิลเตอร์ได้ออกแบบให้มีหน่วยประมวลผลทางคณิตศาสตร์หลายชุดทำงานขนานกันและเป็นอิสระจากกันแทนการใช้หน่วยประมวลผลกลางเพียงชุดเดียว จึงทำให้สามารถกำหนดและควบคุมเวลาที่ใช้ในการประมวลผลของทั้งระบบได้ แม้ว่าการประมวลผลจะมีความซับซ้อนหลายขั้นตอนก็ตาม สำหรับหน่วยของการประมวลผลทางคณิตศาสตร์ที่ใช้ในเครื่องต้นแบบจะมีขนาดของจำนวนบิทข้อมูลที่แตกต่างกันขึ้นอยู่กับความจำเป็นของการประมวลผลแต่ละขั้นตอน และมีการตัดปลายในขั้นสุดท้ายของการประมวลผลก่อนส่งออกเอาท์พุต ทั้งนี้เพื่อให้ข้อมูลที่ได้มีความถูกต้องมากที่สุด

จากการทดสอบการทำงานของเครื่องต้นแบบพบว่าเครื่องกำจัดสัญญาณเสียงสะท้อนที่สร้างขึ้นสามารถกำจัดสัญญาณเสียงสะท้อนได้อย่างสมบูรณ์ มีอัตราการกำจัดสัญญาณสะท้อนได้สูงถึง 34.5 dB ซึ่งมีค่าใกล้เคียงกับค่าที่กำหนดในมาตรฐาน ITU-T G.165 คือ 27 dB และมีค่าเศษที่เหลือของสัญญาณสะท้อนในระดับต่ำ โดยใช้งานได้ในช่วงความถี่ 340 – 3400 Hz

จากผลการวิจัยได้พบว่าการทำงานของระบบที่สร้างด้วยอุปกรณ์ FPGA มีข้อได้เปรียบที่เหนือกว่าการสร้างโดยใช้ชิพคอนโทรลเลอร์ทั่วไปอยู่หลายประการคือ

1) ผู้ออกแบบสามารถกำหนดขนาดความกว้างของบิตข้อมูล (Data bits width) ได้เอง เพื่อให้มีขนาดเป็นไปตามความต้องการในขั้นตอนการออกแบบ

2) ผู้ออกแบบสามารถออกแบบและกำหนดลักษณะของข้อมูล เพื่อให้ผลลัพธ์ที่ได้มีค่าความแม่นยำตรงตามความต้องการในแต่ละขั้นตอนโดยการตัดปลายของข้อมูลในขั้นตอนนั้น ๆ

3) ผู้ออกแบบสามารถออกแบบ กำหนด และควบคุมค่าของเวลาที่ใช้ในการประมวลผลที่มีความซับซ้อนได้เป็นอย่างดี โดยใช้เทคนิคของการออกแบบเพื่อแบ่งแยกให้มีหน่วยประมวลผลหรือหน่วยคำนวณทางคณิตศาสตร์หลายหน่วยซึ่งทำงานแยกจากกันได้ตามความจำเป็น

6.2 ข้อเสนอแนะ

1) งานวิจัยนี้ได้ทดลองสร้างเครื่องต้นแบบให้มีลักษณะการทำงานที่เชื่อมต่อกับระบบภายนอกด้วยสัญญาณอินพุตและเอาท์พุตเป็นแบบอนาลอกเท่านั้น แต่หากจะนำไปใช้กับระบบที่มีลักษณะของข้อมูลเป็นแบบดิจิตอลก็สามารถทำได้เช่นกัน โดยต้องปรับโครงสร้างการทำงานของระบบในส่วนของอัตราสัญญาณสุ่มและรูปแบบลักษณะของข้อมูลที่ใช้ร่วมกัน (เช่น ขนาดความกว้างของบิตข้อมูลและรูปแบบรหัสตัวเลขไบนารี เป็นต้น) เพื่อให้มีความสอดคล้องกับระบบที่จะใช้งานด้วย

2) งานวิจัยนี้ใช้ชิพ FPGA เพื่อสร้างเป็นต้นแบบเท่านั้น ผู้วิจัยจึงเลือกใช้ชิพซึ่งมีขนาดที่ใหญ่ เกินความจำเป็นอยู่มาก หากจะนำไปใช้งานจริงควรปรับเปลี่ยนขนาดหรือตระกูลของชิพให้มีความเหมาะสมกับงานมากขึ้น ซึ่งจะสามารถลดต้นทุนการผลิตลงได้ประมาณ 40% – 80% ทั้งนี้ขึ้นอยู่กับเบอร์และตระกูลของชิพที่เลือกใช้ เช่น ชิพในตระกูล Spartan II ซึ่งเป็นชิพในตระกูลที่ใกล้เคียงกันกับที่ใช้ในเครื่องต้นแบบ แต่มีราคาที่ถูกกว่ามากเมื่อเปรียบเทียบจากขนาดของชิพที่เท่ากันและมีองค์ประกอบภายในชิพที่เกือบจะทดแทนกันได้ทั้งหมด

3) การนำ FPGA มาประยุกต์ใช้งานในด้านการประมวลผลสัญญาณดิจิตอลเป็นเทคโนโลยีที่มีความโดดเด่นมากเนื่องจากสามารถนำไปใช้งานร่วมกับระบบการแปลงสัญญาณอนาลอกให้เป็นสัญญาณดิจิตอลที่มีอัตราความเร็วสูงได้ และมีความยืดหยุ่นมากเมื่อต้องการใช้ในการประมวลผลของสัญญาณที่มีความซับซ้อนสูง แต่การประยุกต์ใช้งานด้วย FPGA ให้มีประสิทธิภาพสูงสุดได้นั้น ผู้ออกแบบต้องมีความรู้และเชี่ยวชาญในหลาย ๆ ด้าน ได้แก่ เทคนิคการออกแบบระดับสูง (High level design) เทคนิคการออกแบบด้วยภาษาที่อธิบายลักษณะทางฮาร์ดแวร์ (Hardware description language) กฎของการออกแบบ (Design rule) ความเชี่ยวชาญเฉพาะทางที่ต้องการนำไปประยุกต์ใช้ ความเข้าใจในเรื่องฮาร์ดแวร์ของวงจรดิจิตอล ความรู้เกี่ยวกับรูปแบบและคุณสมบัติของ FPGA แบบต่าง ๆ เป็นต้น

เอกสารอ้างอิง

[1] Z. Zhoang and G. Schmer, "Analysis of Filter Coefficient Precision on LMS Algorithm Performance for G.165/G.186 Echo Cancellation", Texas Instruments Application Report SPRA561, February 2000.

[2] B. Widrow and S. D. Stearns, "Adaptive signal processing", Prentice- Hall, Inc. 1985 pp. 339-347.

[3] T.A.C.M. Claasen and W.F.G. Mecklenbrauker, "Adaptive Techniques for Signal Processing in Communications", IEEE Commu. Magazine. Vol. 23, No.11, Nov. 1985. pp. 8 - 19.

[4] M.E. Frerking, "Digital Signal Processing in Communication System", Van Nostrand Reinhold New York. 1994. pp. 475 - 482.

[5] D.L. Duttweiler, "A Twelve-Channel Digital Voice Echo Canceller", IEEE Trans. Communication COM-26. No.5, May 1978. pp. 647 – 653

[6] J.V. Oldfield and R.C. Drof , "Field-Programmable Gate Arrays" New York : John Wiley & Sons Inc. 1995 pp. 1-25

ภาคผนวก

ไฟล์ข้อมูลภาษา VHDL ของตัวกำจัดสัญญาณเสียงสะท้อน

```
------------------------------------------------------
--SubProgram for make A to D interfacing--
-- Port IN   CLK :use clock 10 MHz (for make 100 ns. active LOW pulse width)
-- Port IN   CENABLE  :use for set  starting point of the data CONVERSION  ---- pulse
--
-- Port OUT              A2DEN :use for send the >400 ns active LOW pules -- to A/D
-- Port OUT              DREADY        :use for set the starting point of data loading -- to
ALU
-- Port OUT              D2A_WR        :use for control the parallel bit to D/A while --- the
--              other signals pin of D/A as:
--              'CS'  is held to LOW
--              'LDAC' is held to LOW

-- New version for Xilinx 24 MHz clock speed
-- File name is ADAP_24 MHz.vhd
-- Mar 22,2001----

library IEEE;
use IEEE.Std_Logic_1164.all;
use IEEE.Std_Logic_arith.all;
use IEEE.Std_Logic_unsigned.all;

-- Entity-----------------
entity A2D2AP_CONT_LT is

    port(       CLK             : in      std_logic;
       CENABLE                  : in      std_logic;
              A2DEN : out     std_logic;
              DREADY           : out     std_logic;
              HiCE             : out    std_logic;
              LO               : out    std_logic;
              SEN              : out    std_logic;
              D2A_WR           : out     std_logic
               );
    end A2D2AP_CONT_LT;

--Architecture-------------

architecture RTL of A2D2AP_CONT_LT is
        signal TMP_DATA         : std_logic ;
        signal DRIVE            : std_logic ;
        signal TEMP_D2A         : std_logic ;
        signal LDTMP    : std_logic ;
        signal COUNT    : std_logic_vector(9 downto 0);
        signal LOCOUNT          : std_logic_vector(1 downto 0);
        signal SECOUNT          : std_logic_vector(8 downto 0);
begin
COUNTER:
   process(CENABLE,CLK)
   begin
        if (CENABLE = '1') then
                count <= "0000000000";
        elsif (CLK'event and CLK ='1') then
                if (count < 600 ) then                  -- 600 * 41.6667 ns = 0.025ms
                        count <=  count +1;
                if (count >= 0) and (count <= 11) then
                                                        -- 12 * 41.6667 ns = 500 ns
                        DRIVE  <= '0';
                        else
                        DRIVE  <= '1';
```

```
                end if;
                        ----------
                if (count >= 577) and (count <= 600) then
                                        -- 24 * 41.6667 ns = 1000 ns --- at
                                        0.024 ms position
                  TMP_DATA <= '0';
                else
                  TMP_DATA <= '1';
                end if;
                        ----------
                if (count >= 584) and (count <= 595) then
                                        -- 12 * 41.6667 ns = 500 ns at --
                                        0.02433 ms position
                        TEMP_D2A    <= '0';
                else
                        TEMP_D2A    <= '1';
                end if;
                        ----------
        else
                        count <= "0000000000";
        end if;
    end if;
end process;

process(CLK,DRIVE,LOCOUNT)
begin
    if (DRIVE ='0') then
        LOCOUNT <= "00";
    elsif (CLK'event and CLK = '0') then
        LOCOUNT <= LOCOUNT +1;
    end if;

    if (LOCOUNT = 1) then
        LDTMP       <= '1';
    else
        LDTMP       <= '0';
    end if;
end process;

process(LDTMP,SECOUNT,DRIVE)
begin
    if (DRIVE ='0') then
        SECOUNT <= "000000000";
    elsif (LDTMP'event and LDTMP = '0') then
        SECOUNT <= SECOUNT +1;
    end if;

    if (SECOUNT > 1 AND SECOUNT < 130) then
        SEN     <= '1';
    else
        SEN     <= '0';
    end if;
end process;
-----------------------------------------
    LO      <= LDTMP;
    A2DEN       <= DRIVE;
    DREADY      <= TMP_DATA;
    D2A_WR      <= TEMP_D2A;
    HiCe    <= '1';
-----------------------------------------
```

```
end RTL;

--   CE  ===> Chip Enable :Must be synchronous with A/D control signal
--   (i.e.TIMETAB)
--   ,it's reset every time of new sampling begins.
--  AE  ===> Add Enable  :Enable Adder for storage data 's incomes 128
-- times
--  File name Add32b32at.vhd
---------  Mar 18,2001-------------------
-- For new version where work as summation register

library IEEE;
use IEEE.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_signed.all;


entity Add28bit is
   port (
        CE      :  in  STD_LOGIC;
        AE      :  in  STD_LOGIC;
        Ain     :  in  STD_LOGIC_VECTOR (31 downto 0);
        S       :  out STD_LOGIC_VECTOR (15 downto 0);
        Err     :  out STD_LOGIC;
        OFL     :  out STD_LOGIC
      );
end Add28bit;

architecture Add28bit_arch of Add28bit is
        signal TB       :       std_logic;
        signal TA       :       std_logic;
        signal TS       :       std_logic;
        signal SUM      :      STD_LOGIC_VECTOR (33 downto 0);
        signal TMP_SUM      :      STD_LOGIC_VECTOR (33 downto 0);
        signal TMP      :      STD_LOGIC_VECTOR (33 downto 0);
        signal ExTenAin         :      STD_LOGIC_VECTOR (1 downto 0);
begin

Capture_Data:
        process(AE,CE,TMP_SUM)
        begin
                if (CE = '0') then
                        SUM     <= "000000000000000000000000000000000000";
                elsif (AE'event and  AE = '1') then
                        SUM     <= TMP_SUM;
                end if;
        end process;

Correction_Data:
        process(Ain,ExTenAin)
        begin
                if ( Ain(31) = '0') then
                        ExTenAin  <= "00";
                else
                        ExTenAin  <= "11";
                end if;
                TMP     <=  ExTenAin & Ain;
        end process;

        TA      <=  TMP(33);                --
```

```
        TB      <= SUM(33);
        TS      <= TMP_SUM(33);
        S       <= TMP_SUM(31 downto 16);
        Err     <= TMP_SUM(32) or TMP_SUM(33);

Result:
        process(TMP,SUM)
        begin
                TMP_SUM <= TMP + SUM;
        end process;
Overflow:
        process(TA,TB,TS)
        begin
        OFL  <= ((not TA)and(not TB)and TS) or ((not TS)and TA and TB);
        end process;
end Add28bit_arch;
-- File Note:
-- Design for access memory range 128 byte via 24 MHz clock speed
-- but a remain memory bank  high (128 to 255) not used
-- Design for FND PC
-- Currently name : RAM_Access3.vhd
-- Mar,22,2001

library IEEE;
use IEEE.Std_Logic_1164.all;
use IEEE.Std_Logic_arith.all;
use IEEE.Std_Logic_unsigned.all;

-- Entity-----------------
entity data_access is
    port(       CLK             : in    std_logic;
                TIMETAB         : in    std_logic;
                CHIPEN          : in    std_logic;
                RAM_DO          : in    std_logic_vector(15 downto 0);
                ACQ_DAT         : in    std_logic_vector(15 downto 0);
                RAM_EN          : out   std_logic;
                RAM_RST         : out   std_logic;
                RAM_WE          : out   std_logic;
                RAM_DI          : out   std_logic_vector(15 downto 0);
                DATA            : out   std_logic_vector(15 downto 0);
                ADDRESS         : out   std_logic_vector(7 downto 0)
        );
    end data_access;

--Architecture-------------
architecture RTL of data_access is
        signal RAM_TOGLE    : std_logic ;
        signal SELEC        : std_logic ;
        signal SEQ03        : std_logic ;
        signal EN_C2        : std_logic ;
        signal STROB        : std_logic ;
        signal DATA_RDY             : std_logic ;
        signal COUNT01              : std_logic_vector(7 downto 0);
        signal COUNT02              : std_logic_vector(7 downto 0);
        signal COUNT03              : std_logic_vector(1 downto 0);
        signal TMP_DATA             : std_logic_vector(15 downto 0);
        signal DATA_LOAD            : std_logic_vector(15 downto 0);
begin
----<---------Generate sequence for reference  ---------->--------------
SEQUENT_COUNT:
```

```
        process(CHIPEN,CLK,TIMETAB)
        begin
                if (CHIPEN = '1' or TIMETAB = '0') then
                        COUNT03 <= "00";
                elsif (CLK'event and CLK = '0') then
                        COUNT03 <= COUNT03 +1;
                end if;
        end process;
----<---------Generate signal RAM_WE,RAM_EN ---------->-----------------
LOGIC_ENABLE:
        process(CHIPEN,STROB,COUNT03)
        begin
        if(CHIPEN ='0' and  STROB  = '1') then
                RAM_WE  <= COUNT03(1);
                RAM_EN  <= COUNT03(0);
         else
                RAM_WE  <= '0';
                RAM_EN  <= '0' ;
         end if;
        end process;

----<---------Generate Sequence for counter-Control ---------->----------
SEQUENT_GEN:
        process(COUNT03)
        begin
                if (COUNT03 = 3) then
                        SEQ03 <=  '1';
                else
                        SEQ03 <=  '0';
                end if ;
                ----------
        end process;
----<---------Generate Address to point RAM using counter---------->-----
COUNTER:
   process(TIMETAB,SEQ03,COUNT01,COUNT02,EN_C2)
   begin
        if (TIMETAB = '0') then
                        COUNT02 <= "01111111";
                        COUNT01 <= "01111111";
                        EN_C2 <= '0';
        elsif (SEQ03'event and SEQ03 ='0') then
                if (EN_C2 = '1' and COUNT02 < 255)  then
                        COUNT02        <= COUNT02 - 1;
                end if;
                if (COUNT01 < 255 ) then
                        COUNT01 <=    COUNT01 - 1;
                        EN_C2 <= '1';
                end if;
         end if;
   end process;
----<---------Selection of Address to point RAM using counter------->-----
   RAM_TOGLE <= COUNT03(0);                          -----<<<<<<<<<<<
   DATA_RDY  <= COUNT03(1);                          --rise edge = finished read
                                                  --fall edge = finished write
   process(RAM_TOGLE,TIMETAB)
   begin
        if (TIMETAB = '0') then
                SELEC <= '0';
        elsif (RAM_TOGLE'event and RAM_TOGLE = '0') then
                SELEC <= not SELEC;
```

```
        end if;
    end process;
                ----------
    process(SELEC,COUNT01,COUNT02)
    begin
        if ( SELEC = '0') then
                ADDRESS       <= COUNT01;
        else
                ADDRESS       <= COUNT02;
        end if;
    end process;
----<---------Generated of STROB signal ------->-----
    process(COUNT02)
    begin
        if ( COUNT02 = 255 ) then
                STROB <= '0';
        else
        STROB <= '1';
      end if;
    end process;
-----------------------------------------
DATA_SECTION:
        process(DATA_RDY,RAM_DO)
        begin
                if( DATA_RDY 'event and DATA_RDY ='1' ) then
                        if (COUNT01 < 255) then
                                TMP_DATA <= RAM_DO;    -- capture from RAM
                        else
                                TMP_DATA <= DATA_LOAD; --  capture from A/D
                        end if;
                end if;
        end process;
        ---------------------------
        process(TIMETAB,ACQ_DAT)
        begin
                if(TIMETAB'event and TIMETAB ='0') then
                        DATA_LOAD <=          ACQ_DAT;
                end if;
        end process;
        ------------------------------------------
        DATA   <= TMP_DATA;
        RAM_DI  <= TMP_DATA;
        RAM_RST       <= '0';


end RTL;
-- File Note:
-- Design for access memory range 128 byte via 25 MHz clock speed
-- but a remain memory bank  high (128 to 255) not used
-- Design for FND PC
-- Currently name : RAM_Access3Z1.vhd
-- Mar,22,2001

library IEEE;
use IEEE.Std_Logic_1164.all;
use IEEE.Std_Logic_arith.all;
use IEEE.Std_Logic_unsigned.all;

-- Entity-----------------
entity data_access is
```

```
    port(       CLK                 : in    std_logic;
                TIMETAB             : in    std_logic;
                CHIPEN              : in    std_logic;
                ACQ_DAT             : in    std_logic_vector(31 downto 0);
                RAM_DO_A            : in    std_logic_vector(15 downto 0);
                RAM_DO_B            : in    std_logic_vector(15 downto 0);
                RAM_EN              : out   std_logic;
                RAM_RST             : out   std_logic;
                RAM_WE              : out   std_logic;
                RAM_DI_A            : out   std_logic_vector(15 downto 0);
                RAM_DI_B            : out   std_logic_vector(15 downto 0);
                DATA                : out   std_logic_vector(31 downto 0);
                ADDRESS             : out   std_logic_vector(7 downto 0)
            );
    end data_access;

--Architecture----------
architecture RTL of data_access is
        signal RAM_TOGLE    : std_logic ;
        signal SELEC        : std_logic ;
        signal SEQ03        : std_logic ;
        signal EN_C2        : std_logic ;
        signal STROB        : std_logic ;
        signal DATA_RDY             : std_logic ;
        signal COUNT01              : std_logic_vector(7 downto 0);
        signal COUNT02              : std_logic_vector(7 downto 0);
        signal COUNT03              : std_logic_vector(1 downto 0);
        signal TMP_DATA_A   : std_logic_vector(15 downto 0);
        signal DATA_LOAD_A  : std_logic_vector(15 downto 0);
        signal TMP_DATA_B   : std_logic_vector(15 downto 0);
        signal DATA_LOAD_B  : std_logic_vector(15 downto 0);
begin
----<---------Generate sequence for reference  ---------->----------
SEQUENT_COUNT:
        process(CHIPEN,CLK,TIMETAB)
        begin
                if (CHIPEN = '1' or TIMETAB = '0') then
                        COUNT03 <= "00";
                elsif (CLK'event and CLK = '0') then
                        COUNT03 <= COUNT03 +1;
                end if;
        end process;
----<---------Generate signal RAM_WE,RAM_EN ---------->----------
LOGIC_ENABLE:
        process(CHIPEN,STROB,COUNT03)
        begin
        if(CHIPEN ='0' and  STROB  = '1') then
                RAM_WE  <= COUNT03(1);
                RAM_EN  <= COUNT03(0);
         else
                RAM_WE  <= '0';
                RAM_EN  <= '0' ;
         end if;
        end process;
----<---------Generate Sequence for counter-Control ---------->----------
SEQUENT_GEN:
        process(COUNT03)
        begin
                if (COUNT03 = 3) then
```

```
                    SEQ03 <=  '1';
                else
                    SEQ03 <=  '0';
                end if ;
                -----------
        end process;
----<---------Generate Address to point RAM using counter---------->-----
COUNTER:
   process(TIMETAB,SEQ03,COUNT01,COUNT02,EN_C2)
   begin
        if (TIMETAB = '0') then
                    COUNT02 <= "01111111";
                    COUNT01 <= "01111111";
                    EN_C2 <= '0';
        elsif (SEQ03'event and SEQ03 ='0') then
                if (EN_C2 = '1' and COUNT02 < 255)  then
                COUNT02       <= COUNT02 - 1;
                end if;
                if (COUNT01 < 255 ) then
                    COUNT01 <=   COUNT01 - 1;
                    EN_C2 <= '1';
                end if;
         end if;
   end process;
----<---------Selection of Address to point RAM using counter------->-----
   RAM_TOGLE <= COUNT03(0);                              -----<<<<<<<<<<<
--------------------------------
   process(COUNT03)
   begin
        if(COUNT03 ="10") then
                DATA_RDY  <= '1';
        else
                DATA_RDY  <= '0';
        end if;
   end process;

--------------------------------
   process(RAM_TOGLE,TIMETAB)
   begin
        if (TIMETAB = '0') then
                SELEC <= '0';
        elsif (RAM_TOGLE'event and RAM_TOGLE = '0') then
                SELEC <= not SELEC;
        end if;
    end process;
                --------------
   process(SELEC,COUNT01,COUNT02)
   begin
        if ( SELEC = '0') then
                ADDRESS       <= COUNT01;
        else
                ADDRESS       <= COUNT02;
        end if;
   end process;
----<---------Generated of STROB signal ------->-----
   process(COUNT02)
   begin
        if ( COUNT02 = 255 ) then
                STROB <= '0';
        else
```

```
                STROB <= '1';
        end if;
    end process;
-----------------------------------------
DATA_SECTION:
        process(DATA_RDY,RAM_DO_A,DATA_LOAD_A,RAM_DO_B,DATA_LOAD_B,CLK
)
        begin
                if( CLK'event and CLK ='1') then
                   if( DATA_RDY  ='1' ) then
                        if (COUNT01 < 255) then
                                TMP_DATA_A <= RAM_DO_A;
                                -- capture from RAM
                                TMP_DATA_B <= RAM_DO_B;
                        else
                                TMP_DATA_A <= DATA_LOAD_A;
                                -- capture from A/D
                                TMP_DATA_B <= DATA_LOAD_B;
                        end if;
                   end if;
                end if;
        end process;
        ----------------------------
        process(TIMETAB,ACQ_DAT)
        begin
                if(TIMETAB'event and TIMETAB ='0') then
                        DATA_LOAD_A <=      ACQ_DAT(31 downto 16);
                        DATA_LOAD_B <=      ACQ_DAT(15 downto 0);
                end if;
        end process;
        ----------------------------------------
        DATA(31 downto 16)  <= TMP_DATA_A;
        DATA(15 downto 0)   <= TMP_DATA_B;
        RAM_DI_A            <= TMP_DATA_A;
        RAM_DI_B            <= TMP_DATA_B;
        RAM_RST                 <= '0';
        ----------------------------
end RTL;

-- File Note:
-- Design for access memory range 128 byte via 24 MHz clock speed
-- but a remain memory bank  high (128 to 255) not used
-- Design for FND PC
-- Old name : Data_access.vhd
-- Currently name : TAB_Access2M.vhd
-- Mar,22,2001

library IEEE;
use IEEE.Std_Logic_1164.all;
use IEEE.Std_Logic_arith.all;
use IEEE.Std_Logic_unsigned.all;

-- Entity------------------
entity data_access is
    port(       CLK         : in    std_logic;
                TIMETAB     : in    std_logic;
                CHIPEN      : in    std_logic;
                RAM_DO      : in  std_logic_vector(15 downto 0);
                RAM_EN      : out   std_logic;
                RAM_RST     : out   std_logic;
```

```
                RAM_WE          : out     std_logic;
                DAT_RDY         : out     std_logic;
                DATA            : out   std_logic_vector(15 downto 0);
                ADDRESS         : out     std_logic_vector(7 downto 0)
        );
    end data_access;

--Architecture-------------
architecture RTL of data_access is
        signal SEQ03            : std_logic ;
        signal STROB            : std_logic ;
        signal DATA_RDY                 : std_logic ;
        signal OUT_RDY                  : std_logic ;
        signal COUNT01                  : std_logic_vector(7 downto 0);
        signal COUNT03                  : std_logic_vector(1 downto 0);
        signal TMP_DATA                 : std_logic_vector(15 downto 0);
begin


SEQUENT_COUNT:
        process(CHIPEN,CLK,TIMETAB)
        begin
                if (CHIPEN = '1' or TIMETAB = '0') then
                        COUNT03 <= "00";
                elsif (CLK'event and CLK = '0') then
                        COUNT03 <= COUNT03 +1;
                end if;
        end process;

--RAM_CONTROL-----------------
                -- RAM use  : Enabl   == RAM_EN  <== Use COUNT03(0)
                --          : Address == ADDRESS <== ADRS (COUNT01)
                --          : Reset   == RST     <== Short at GND mode
                --          : WriteEn == WE      <== Use COUNT03(1)
                --          : Data Ready == DAT_RDY <== used for latch
                --          data after read cycle

DATA_READY_GENERATOR:
        -------------
        process(CLK,COUNT03,TIMETAB,CHIPEN)
        begin
                if (COUNT03 = 2 and TIMETAB ='1' and CHIPEN ='0') then
                        OUT_RDY     <=  CLK;
                else
                        OUT_RDY     <= '0';
                end if;
        end process;
-----------------------------------------------------
SEQUENT_GEN:
        process(COUNT03)
        begin
                if (COUNT03 = 3) then
                        SEQ03 <=  '1';
                else
                        SEQ03 <=  '0';
                end if ;
                -----------
        end process;

COUNTER:
```

```
    process(TIMETAB,SEQ03,COUNT01,STROB)
    begin
        if (TIMETAB = '0') then
                    COUNT01 <= "01111111";
        elsif (SEQ03'event and SEQ03 ='0') then
            if (COUNT01 < 255 ) then
                    COUNT01 <= COUNT01 -1;
            end if;
        end if;
    end process;
-----------------------------------------------------

LOGIC_ENABLE:
        process(COUNT01)
        begin
            if (COUNT01 = 255) then
                STROB <= '0';
            else
              STROB <= '1';
            end if;
        end process;

        process(CHIPEN,STROB,OUT_RDY,COUNT03)
        begin
        if(CHIPEN ='0' and  STROB  = '1') then
                RAM_WE  <= COUNT03(1);
                RAM_EN  <= COUNT03(0);
                DAT_RDY <= OUT_RDY ;
         else
                RAM_WE  <= '0';
                RAM_EN  <= '0' ;
                DAT_RDY <= '0' ;
         end if;
        end process;
-----------------------------------------------
DATA_SECTION:
        -- use falling edge of DAT_RDY to latch keep of data
        process(CLK,RAM_DO,COUNT03)                    --DATA_RDY
        begin
--              if( DATA_RDY 'event and DATA_RDY ='0' ) then
--                      TMP_DATA <= RAM_DO;
--              end if;
                if( COUNT03 = 1) then
                  if(CLK'event and CLK ='0') then
                        TMP_DATA <= RAM_DO;
                  end if;
                end if;
        end process;

-----------------------------------------------
        ADDRESS <= COUNT01;
        RAM_RST       <= '0';
        DATA   <= TMP_DATA;           -- data from tab to sum
-----------------------------------------------------------
end RTL;
```

ประวัติผู้เขียน

นายวัชระ ภัคมาตร์ เกิดเมื่อวันที่ 11 พฤศจิกายน 2508 ที่จังหวัดขอนแก่น สำเร็จการศึกษาวิศวกรรมศาสตรบัณฑิต (วิศวกรรมอิเลกทรอนิคส์) จากสถาบันเทคโนโลยีราชมงคล วิทยาเขตเทเวศร์ ปีการศึกษา 2536 และประกาศนียบัตรวิชาชีพชั้นสูง (ไฟฟ้าอุตสาหกรรม) จากสถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ