วงจรกรองความถี่กลางผ่านแบบ IIR ปรับค่าตัวแปรอัตโนมัติโดยวิธีการนอร์มอลไลซ์เกรเดียนต์

Adaptive IIR Band-Pass Filter Using Normalized Gradient Algorithm

โดย

นายจิตติกร อาชีวะ

นายสันติ ถือทอง

นายอำนาจ บุญมี

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตรบัณฑิต

ภาควิชาเทคนิคอุตสาหกรรม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2542

หัวข้อปริญญานิพนธ์ วงจรกรองความถี่กลางผ่านแบบ IIR ปรับค่าตัวแปรอัตโนมัติโดยวิธีการนอร์มอลไลซ์เกรเดียนต์

Adaptive IIR Band-Pass Filter Using Normalized Gradient Algorithm

โดย	**1. นายจิตติกร**	**อาชีวะ**	**รหัส**	**40013325**	**ห้อง 3U/1**
	2. นายสันติ	**ถือทอง**	**รหัส**	**40013355**	**ห้อง 3U/1**
	3. นายอำนาจ	**บุญมี**	**รหัส**	**40013360**	**ห้อง 3U/1**

ภาควิชา เทคนิคอุตสาหกรรม

อาจารย์ที่ปรึกษา ผศ.ชวลิต เบญจางคประเสริฐ

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตรบัณฑิต

คณะกรรมการสอบปริญญานิพนธ์

_____ **อาจารย์ที่ปรึกษา**
(_____)

_____ **คณะกรรมการ**
(_____)

_____ **คณะกรรมการ**
(_____)

_____ **คณะกรรมการ**
(_____)

_____ **คณะกรรมการ**
(_____)

หัวข้อปริญญานิพนธ์	**วงจรกรองความถี่กลางผ่านแบบ IIR ปรับค่าตัวแปรอัตโนมัติโดยวิธีการนอร์มอลไลซ์เกรเดียนต์**				
โดย	**1. นายจิตติกร**	**อาชีวะ**	**รหัส**	**40013325**	**ห้อง 3U/1**
	2. นายสันติ	**ถือทอง**	**รหัส**	**40013355**	**ห้อง 3U/1**
	3. นายอำนาจ	**บุญมี**	**รหัส**	**40013360**	**ห้อง 3U/1**
ภาควิชา	**เทคนิคอุตสาหกรรม**				
อาจารย์ที่ปรึกษา	**ผศ.ชวลิต เบญจางคประเสริฐ**				

บทคัดย่อ

โครงงานนี้เสนอวงจรกรองความถี่กลางผ่านแบบ IIR ปรับค่าตัวแปรอัตโนมัติโดยวิธีนอร์มอลไลซ์เกรเดียนต์ ทำให้วงจรปรับสัมประสิทธิ์ของตัวแปร และสามารถติดตามสัญญาณทางอินพุตตามความถี่ที่ใช้งานได้เร็วกว่าอัลกอริทึมเดิม ซึ่งสามารถยืนยันได้ด้วยการจำลองการทำงานด้วยโปรแกรมคอมพิวเตอร์ และเขียนโปรแกรมบนบอร์ดประมวลผลสัญญาณดิจิตอล เบอร์ TMS320c50 เพื่อให้บอร์ดประมวลผลสัญญาณดิจิตอลทำงานตามอัลกอริทึมใหม่ จากผลการทดลองระหว่างการทำงานด้วยโปรแกรม กับการใช้งานจริงให้ผลดีสอดคล้องกัน

PROJECT	**Adaptive IIR Band-Pass Filter Using Normalized Gradient Algorithm**		
NAME	**1. Mr.Chittikorn**	**Archeewa**	**40013325**
	2. Mr.Santi	**Thuthong**	**40013355**
	3. Mr.Amnaj	**Boonmee**	**40013360**
DEPARTMENT OF	**Industrial Technology**		
ADVISOR	**Asst. Prof. Chawalit Benjangkaprasert**		

ABSTRACT

This project presents an adaptive IIR band-pass filter using normalized gradient algorithm. The algorithm can adjusted the coefficient and can track the input frequency faster than the previous algorithm. The simulation and the application on DSP- board TMS 320c50 are show good result.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ สำเร็จลุล่วงได้ด้วยดี ก็เนื่องจากได้รับคำแนะนำ และการช่วยเหลือส่วนข้อมูลต่างๆ อย่างดียิ่งจาก ผู้ช่วยศาสตราจารย์ ชวลิต เบญจางคประเสริฐ ที่เป็นอาจารย์ที่ปรึกษา และนอกจากนี้ยังได้รับความเอื้อเฟื้อเครื่องมือและห้องปฏิบัติการในการทำโครงงานครั้งนี้จาก อาจารย์ อรลาภ แสงอรุณ , อาจารย์ นภพินทุ์ อนันตรศิริชัย

ขอกราบขอบพระคุณบิดา มารดา ที่ช่วยเป็นกำลังใจในการทำงานและคอยให้ความช่วยเหลือ อุปการะในทุกด้าน

ขอขอบคุณเพื่อนๆ ทั้งหลายสำหรับคำแนะนำ และความช่วยเหลือทุกๆ อย่าง

คณะผู้จัดทำ

12 พฤศจิกายน 2542

คำอธิบายสัญญลักษณ์

k	:	จำนวนเวลาการ Sampling (จำนวนเต็ม)
x(k)	:	สัญญาณทางด้าน Input
y(k)	:	สัญญาณทางด้าน Output
n(k)	:	สัญญาณรบกวนแบบเกาส์
H(z)	:	Transfer Function ของวงจร IIR Band-Pass Filter
$\alpha_0(k)$	:	สัมประสิทธิ์คงที่สำหรับกำหนดค่า Q-Factor ของวงจรกรอง ความถี่ที่มีค่าระหว่าง $0 < \alpha_0 < 1$
$\alpha_1(k)$	:	สัมประสิทธิ์แปรค่าที่ใช้กำหนดค่าความถี่กลาง (ω_0)มีค่าระหว่าง $-1 < \alpha_1 < 1$
$\psi_{01}(k), \psi_{02}(k)$	:	สัญญาณควบคุม Adaptive ของ $\alpha_1(k)$
μ_0, μ_1	:	Step Size Parameter
$\frac{\partial \lvert H(\partial\alpha_1(k))\rvert^2}{\partial\alpha_1(k)}$	:	สัดส่วนโดยตรงกับค่า $y(k)$ และ $\psi(k)$ ซึ่งต่างก็มาจาก H(z) เนื่องจากว่า $\alpha_1(k)$ จะติดตามความถี่ได้เร็วหรือไม่นั้นก็ขึ้นอยู่กับค่าของ $y(k)*\psi(k)$
$\alpha_{0,opt}(k)$	:	ค่า $\alpha_1(k)$ มีค่าสูงสุดซึ่งหาได้จากสมการ $\frac{\partial H(\alpha_0(k))}{\partial\alpha_0(k)} = 0$
$\lvert H(\partial\alpha_1(k))\rvert^2$	:	เป็นฟังก์ชั่นของ α_0 ซึ่งเป็นสัดส่วนโดยตรงกับค่าเฉลี่ยกำลังสองของ $[y(k)]^2$
$\prod(\alpha_0)$	:	เป็นฟังก์ชั่นของ α_0 มีค่าเท่ากับ $\frac{\partial \lvert H(\alpha_1(k))\rvert}{\partial\alpha_1(k)}$
γ	:	เป็นค่า Forgetting Factor โดยมีค่าระหว่าง $0 << \gamma << 1$
$\beta(k)$	:	กำลังงานของ Update Function

สารบัญ

หน้า

บทที่ 1
ความรู้เบื้องต้นเกี่ยวกับ ADAPTIVE DIGITAL FILTER

1. บทนำ

การประมวลสัญญาณไฟฟ้าในอดีต จะเป็นการใช้อุปกรณ์ ทางอนาล็อก เช่น รีซีสเตอร์ , คาปาซิเตอร์ มาประกอบเป็นตัวฟิลเตอร์ เพื่อจัดรูปแบบของสัญญาณใหม่ (ประมวลผลสัญญาณ) ซึ่งคุณสมบัติของฟิลเตอร์ก็สามารถจะปรับปรุงเปลี่ยนแปลงไปได้ตามวงจรที่ประกอบและค่าของตัวอุปกรณ์เอง ในการประกอบวงจรทางอนาล็อกนั้น สิ่งหนึ่งที่เป็นข้อเสียที่หลีกเลี่ยงไม่ได้ก็คือ การที่ค่าของอุปกรณ์อาจเปลี่ยนแปลงไปได้ตามอุณหภูมิ ส่งผลทำให้การทำงานผิดพลาดไปด้วย โดยเฉพาะอย่างยิ่งหากทำงานที่ความถี่สูงๆ การประกอบวงจรจะเป็นไปด้วยความยากลำบาก ผลอันเนื่องจาก สัญญาณรบกวนและ การเดินทางของคลื่นไฟฟ้าในความถี่สูงๆ นั้นจะเดินทางไปเฉพาะที่ผิวของวัตถุเท่านั้น ทำให้การประกอบอุปกรณ์ลงปริ้นต้องออกแบบทำด้วยวัสดุพิเศษ

ปัจจุบันนี้วิทยาการทางด้านคอมพิวเตอร์ได้เจริญก้าวหน้ามาก และได้ก้าวไปอย่างรวดเร็ว สามารถนำเอาคอมพิวเตอร์มาประมวลผลได้ ทำให้การศึกษาเรื่องของการประมวลสัญญาณไฟฟ้าดิจิตอลนั้นสามารถกระทำได้ง่ายขึ้น ซึ่งการประมวลผลด้วยคอมพิวเตอร์สามารถที่จะปรับปรุงพัฒนาฟิลเตอร์ได้อย่างต่อเนื่องและรวดเร็วเพราะมีข้อมูลเดิมอยู่ในหน่วยความจำตลอดเวลา ตัดผลเสียเรื่องการของระบบประมวลผลอนาล็อกเรื่องการเปลี่ยนแปลงค่าของอุปกรณ์ตามอุณหภูมิไปได้ ทั้งนี้เพราะเราสามารถจะสร้างฟิลเตอร์ได้โดยการเขียนโปรแกรมคอมพิวเตอร์ ทั้งนี้ยังง่ายในการเปลี่ยนแปลงคุณสมบัติได้โดยการเปลี่ยนแปลงค่าของ พารามิเตอร์ในโปรแกรมเท่านั้นเอง สามารถจำลองดูผลการทำงาน บันทึกผลไว้ประมวลต่อไปได้ และหากเป็นการกระทำกับสัญญาณที่มีความถี่สูงๆ ปัจจุบันก็มีอุปกรณ์ต่อเชื่อมกับคอมพิวเตอร์ผลสัญญาณโดย ฮาร์ดแวร์ ซึ่งจะมีความเร็วสูงมาก และได้มีการพัฒนาความเร็วสูงไปเรื่อยๆ

ปริญญานิพนธ์ฉบับนี้ได้นำ บอร์ดประมวลผล (DSP Board) TMS320C50 ของบริษัท Texas Instruments มาใช้ในการประมวลผลสัญญาณดิจิตอล โดยออกแบบวงจรกรองความถี่ที่ใช้อะแดปทีฟอัลกอริทึม ชนิดนอร์มอลไซ์เกรเดียนต์ ประยุกต์เข้ากับวงจรผ่านแถบความถี่ (band-pass filter) แบบ ไอไออาร์ ผลของการออกแบบนี้ ได้เปรียบเทียบผล จากโปรแกรมที่เขียนขึ้นเพื่อการจำลองงานจริง ตามสมการต่างๆ กับ ผลจากการทดลองกับ บอร์ดประมวลผล (DSP Board) ที่ใช้กับงานจริง

2. INTRODUCTION TO DIGITAL FILTER

สัญญาณเต็มหน่วย (discrete-time signal) ที่ได้จากการวัดอาจจะมีสัญญาณอย่างอื่นที่ไม่ต้องการซึ่งเรียกว่า สัญญาณรบกวน (noise) ปะปนผสมเข้ามาด้วย จึงจำเป็นต้องนำสัญญาณนี้ผ่านเข้าไปในอุปกรณ์อย่างหนึ่งซึ่งเรียกว่า ตัวกรองสัญญาณ (filter) ซึ่งทำหน้าที่กรองเอาสัญญาณรบกวนออกที่ด้านออกของตัวกรองเราจะได้สัญญาณที่ต้องการ ในการกรองสัญญาณนั้นก่อนอื่นจะต้องรู้จักคุณลักษณะเฉพาะตัวของสัญญาณที่ต้องการจะวัดและของสัญญาณรบกวนที่เกิดขึ้น เพื่อให้สามารถสร้างตัวกรองสัญญาณที่มีคุณสมบัติยอมให้เฉพาะสัญญาณที่ต้องการผ่านออกมาทางด้านออกได้เท่านั้น กระบวนการกรองสัญญาณเต็มหน่วยนี้หากใช้วิธีทางดิจิตอล เช่นใช้คอมพิวเตอร์จะเรียกว่า ตัวกรองสัญญาณดิจิตอล (digital filter)

Digital Filter เป็นสิ่งที่สำคัญมากที่สุดอย่างหนึ่งในการนำไปใช้ในระบบ DSP (Digital Signal Processing) เป็นการนำ Algorithm ทางคณิตศาสตร์มาใช้ในระบบ Hardware หรือ Software กล่าวคือ สัญญาณ Digital Input จะทำการสร้างสัญญาณ Digital Output โดยใช้ Filtering Algorithm เพื่อให้ได้รับสัญญาณจากการ Filtering ตามความต้องการเพื่อนำไปใช้ใน Digital Filter สำหรับชนิดของ ตัวกรองสัญญาณ ที่สำคัญจะถูกกำหนดโดย

$$y(n) = \sum_{k=0}^{n-1} h(k).x(n-k) \qquad (1)$$

โดยที่ h(k) คือ สัมประสิทธิ์ ของ Filter

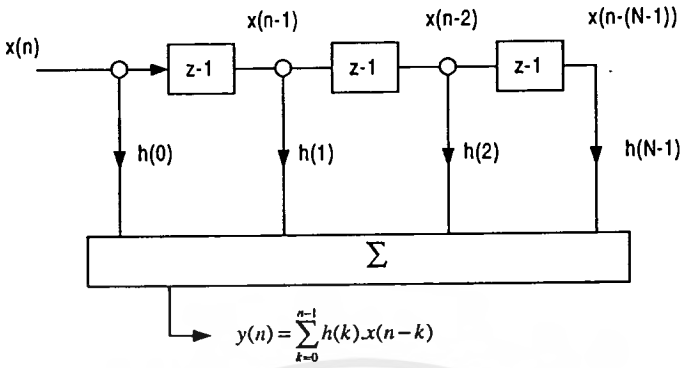
k คือ 0,1 ,2,...n-1

x(n) คือ Input Filter

y(n) คือ Output Filter

ซึ่งสำหรับ ตัวกรองสัญญาณ ที่ให้มา ค่าสัมประสิทธิ์จะกำหนด Characteristic ของ ตัวกรองสัญญาณ Filtering นี้จะอยู่ในการ Convolution ที่แท้จริงของสัญญาณ และการตอบสนองต่อ Impulse ของ ตัวกรองสัญญาณ ใน Time Domain ซึ่งก็คือ h(k)

รูปที่จะกล่าวถึงแสดงถึง Block Diagram ซึ่งจะแสดงถึงนิยามของ Digital Filter ที่กำหนดไว้ก่อนหน้านี้ โดยที่รูปแบบ ตัวกรองสัญญาณ ชนิดนี้เป็นที่รู้จักอย่างแพร่หลาย เรียกว่า Transversal Filter จากรูป Z^{-1} จะแสดงถึง Delay ของช่วงเวลาในการสุ่มตัวอย่างหนึ่งครั้ง



รูปที่ 1 Block Diagram ของ Transversal Filter โดยที่มี h(k),(k=0,1,2,…,n-1)

วัตถุประสงค์ของ ตัวกรองสัญญาณ โดยรวมๆ แล้วเป็นการกำจัดหรือลด Noise จากสัญญาณที่ต้องการดังตัวอย่างดังรูปโดยการดึงข้อมูลของกลุ่มที่สนใจในช่วงหนึ่งออกมาซึ่ง Noise ในระบบอื่นๆ อาจจะปะปนมาด้วยดังนั้นเราจึงทำการกรองสัญญาณที่แท้จริงออกมาเพื่อที่จะสามารถนำไปใช้ในการวิเคราะห์ได้ถูกต้อง

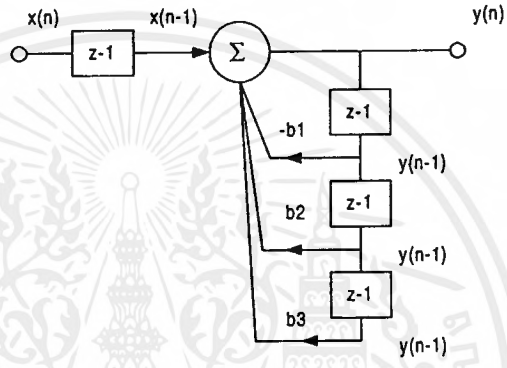
3. Z-TRANSFORM

Z-Transform เป็นการแปลงอีกชนิดหนึ่งซึ่งนักออกแบบตัวกรองสัญญาณทั้งหลายนิยมใช้อ้างถึง วิธีการนี้จะเป็นการแปลงลำดับซึ่งเป็นตัวเลขที่เขียนเรียงกันให้เป็นฟังก์ชันของตัวแปรเชิงซ้อน (Complex variable) หรือกล่าวให้ง่ายคือการเอาลำดับชุดหนึ่งไปสร้างฟังก์ชันตามกฎเกณฑ์ทางคณิตศาสตร์ที่กำหนดไว้ ดังนั้นคุณสมบัติต่างๆที่มีอยู่ในลำดับจะถูกส่งเข้าไปอยู่ในฟังก์ชันที่สร้างขึ้นด้วย และจะใช้อักษร Z ในภาษาอังกฤษเป็นสัญลักษณ์ของตัวแปรในฟังก์ชันที่สร้างขึ้น

Z-Transform เป็นสิ่งที่สำคัญที่สุดอย่างหนึ่งในระบบ DSP นั่นคือใช้ในการออกแบบเพื่อวิเคราะห์สัญญาณ Error ใน Digital Filter โดยเฉพาะในระบบ IIR Filter ซึ่งมักจะถูกใช้ในการคำนวณหาค่าสัมประสิทธิ์ของ Digital Filter และวิเคราะห์ผลของการ Quantization Errors คือใช้ในการแสดงโครงสร้างของ Digital Filter

4. REALIZATION STRUCTURES FOR DIGITAL FILTER

Discrete-Time Filter จะถูกนำมาแสดงบ่อย ๆ ในรูปแบบของ Block Diagram หรือ Signal Flow Graph ซึ่ง Block Diagram จะเป็นการสะดวกกว่าที่ใช้ในการแสดง Difference Equations หรือ Equivalent ของ Transfer Function ซึ่งสามารถพิจารณาดังตัวอย่าง Discrete Filter ด้วย Difference Equations ดังรูปต่อไปนี้



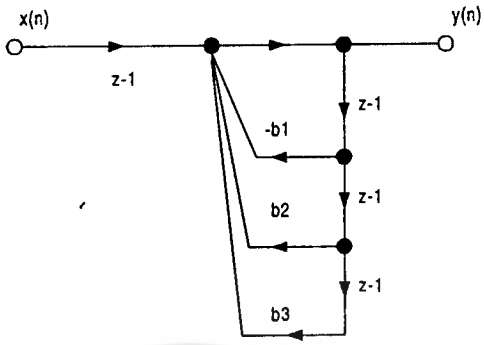
รูปที่ 3 การวิเคราะห์สมการ Difference Equation ในรูปแบบ Block Diagram

โดยสมมุติสมการดังนี้

$$y(n) = x(n-1) - b_1 y(n-1) + b_2 y(n-2) + b_3 y(n-3)$$

Block Diagram แสดงถึงสมการดังกล่าวจากรูปสัญลักษณ์ Z^{-1} จะแสดงถึง Delay ของช่วงเวลาหนึ่งหน่วย ซึ่งอาจจะพิจารณาใน Node ต่างๆ ลูกศรแสดงถึงการ Multipliers และค่าคงที่ตัวถัดมา ทำการคูณกับตัวประกอบ ความสัมพันธ์ระหว่าง Difference Equation และ Block Diagram จะค่อยๆ ปรากฏขึ้น และ Signal Flow Graph Diagram ก็ใช้แสดงกับ Difference Equation เช่นเดียวกันดังรูปที่ 4 โดยสมมุติสมการดังนี้

$$y(n) = x(n-1) - b_1 y(n-1) + b_2 y(n-2) + b_3 y(n-3)$$

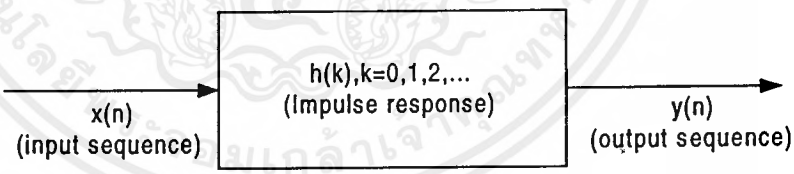


รูปที่ 4 การวิเคราะห์สมการ Difference Equation ในรูปแบบ Signal Flow Graph Diagram

5. TYPE OF DIGITAL FILTER

Digital Filter ที่ใช้อย่างกว้างขวางแบ่งออกเป็น 2 ชนิด

1. Finite Impulse Response (FIR)
2. Infinite Impulse Response (IIR)



รูปที่ 5 Block Diagram แสดงถึงหลักการของ Digital Filter

ตัวกรองสัญญาณแต่ละชนิด จะมีรูปแบบพื้นฐานแต่ละตัวซึ่งสามารถแสดงลำดับของการตอบสนองต่อ Impulse h(k) ; (k=0,1,2,...) ดังรูปที่ 5 สัญญาณ Input และ Output ที่ต่อตัวกรองสัญญาณ จะถูกนำมาเกี่ยวพันโดยผลรวมของการ Convolution ซึ่งสมการแต่ละชนิดแสดงได้ดังต่อไปนี้

$$y(n) = \sum_{k=0}^{\infty} h(k).x(n-k) \quad : \quad \text{IIR Digital Filter} \qquad (2)$$

$$y(n) = \sum_{k=0}^{n-1} h(k).x(n-k) \quad : \quad \text{FIR Digital Filter} \qquad (3)$$

จากสมการข้างต้น สำหรับ IIR Digital Filter นั้น ผลการตอบสนองต่อ Impulse จะอยู่ในช่วงที่ไม่จำกัด แต่สำหรับ FIR Digital Filter จะอยู่ในช่วงที่กำหนด โดยเริ่มที่ h(k) สำหรับ FIR จะมีเพียง N ค่า ส่วนในการคำนวณหาค่า Output ของ IIR ไม่สามารถเป็นไปได้ เนื่องจากสมการมีความยาวของผลการตอบสนองต่อ Impulse นั้นยาวมาก (∞ : Infinite) ดังนั้นเราจึงแทนสมการของ IIR Digital Filter ในรูปของ Recursive Form

$$y(n) = \sum_{k=0}^{\infty} h(k).x(n-k) = \sum_{k=0}^{\infty} a_k.x(n-k) - \sum_{k=1}^{\infty} b_k.x(n-k) \qquad (4)$$

โดยที่ a_k และ b_k เป็นสัมประสิทธิ์ของ ตัวกรองสัญญาณ ดังนั้นจากสมการข้างต้นคือสมการที่แตกต่างกันระหว่าง FIR และ IIR Digital Filter จากสมการเหล่านั้นจะมีค่า h(k) สำหรับ FIR และ a_k และ b_k สำหรับ IIR ซึ่งสมการทั้งสองเป็นสิ่งที่สำคัญมากที่ใช้ในการออกแบบ ตัวกรองสัญญาณ เป็นส่วนใหญ่ จากสมการ กระแสการสุ่มตัวอย่าง Output y(n) เป็น Function ของ Output ที่ผ่านมา ซึ่งค่าที่ถูกต้องจะเท่ากับค่าตัวอย่าง Input ที่สุ่มมา ทั้งในอดีตและปัจจุบัน ซึ่งนั้นคือ IIR จะมีการ Feedback ซึ่งจะทำการเปรียบเทียบกับสมการ FIR แล้ว กระแสการสุ่มตัวอย่าง Output y(n) จะเป็นเพียง Function ของค่า Input ของอดีตและปัจจุบัน เท่านั้นเอง แต่อย่างไรก็ตามเมื่อ b_k มีค่าเท่ากับ 0 สมการ IIR คือ สมการ FIR นั้นเอง

และจากสมการข้างล่าง นี้ก็คือ Transfer Function ของ FIR และ IIR Digital Filter ซึ่งค่า Transfer Function เหล่านี้จะมีประโยชน์มาก ในการคำนวณหาค่า ใน Frequency Response

$$\frac{H(z) = \sum_{k=0}^{n} a_k.z^{-k}}{1 + \sum_{k=1}^{m} b_k.z^{-k}} \quad : \quad \text{IIR Digital Filter} \qquad (5)$$

$$H(z) = \sum_{k=0}^{n-1} h(k).z^{-k} \quad : \quad \text{FIR Digital Filter} \qquad (6)$$

6. SUMMARY OF KEY CHARACTERISTIC FEATURES OF FIR FILTER

1. Characteristic ของ FIR Digital Filter จะแสดงดังสมการ

$$y(n) = \sum_{k=0}^{n-1} h(k).x(n-k) \qquad (7)$$

$$H(z) = \sum_{k=0}^{n-1} h(k).z^{-k} \qquad (8)$$

โดยที่ h(k) ; k = 0,1,2,...,n-1 เป็นสัมประสิทธิ์ในการตอบสนองต่อ Impulse ของ ตัวกรองสัญญาณ

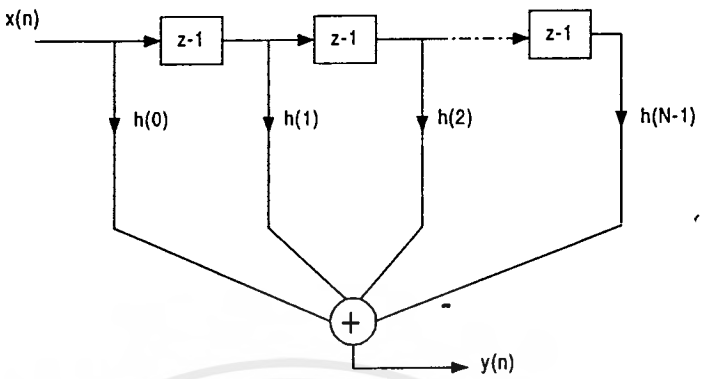
H(z) เป็น Transfer Function ของ ตัวกรองสัญญาณ

n เป็น Filter Length

นั่นก็คือ จำนวนของสัมประสิทธิ์ของ ตัวกรองสัญญาณ ของสมการแรกเป็น FIR Difference Equation ซึ่งมันจะเป็นสมการใน Time Domain และจะพิจารณา FIR Filter ในรูปของ Non - Recursive Form ซึ่งจะทำการสุ่มตัวอย่าง Output y(n) เป็น Function เพียงค่าของ Input ในอดีตและปัจจุบัน เท่านั้น x(n) เมื่อ ตัวกรองสัญญาณ ถูกนำไปใช้ในรูปแบบนี้นั่นคือ จะเป็นการหาค่าโดยตรง จะทำให้ได้ค่าคงที่ (Stable) และ Transfer Function ของ ตัวกรองสัญญาณ จะใช้ในการวิเคราะห์และหาค่าตัวกรองสัญญาณ ในทาง Frequency Response

2. FIR Filter สามารถมีความเป็น Linear Phase Response ที่แท้จริง

3. FIR Filter ง่ายที่จะนำไปใช้ระบบ DSP Processor ทั้งหมดมีความเหมาะสมที่จะใช้กับ FIR Filtering Non-Recursive FIR Filter จะมีข้อผิดพลาดน้อยจากผลของความยาวที่จำกัดกว่า IIR Filter



รูปที่ 6 โครงสร้างของ FIR Digital Filter

7. SUMMARY OF THE BASIC FEATURES OF IIR FILTER

Characteristic ของ IIR Digital Filter ถูกกำหนดตาม Recursive Equation ดังนี้

$$y(n)=\sum_{k=0}^{\infty}h(k).x(n-k)=\sum_{k=0}^{\infty}a_k.x(n-k)-\sum_{k=1}^{\infty}b_k.x(n-k) \qquad (9)$$

โดยที่ h(k) เป็นการตอบสนองต่อ Impulse ของ ตัวกรองสัญญาณ ซึ่งจะมีช่วงเวลามิ่ไม่จำกัด

a_k และ b_k เป็นสัมประสิทธิ์ของ Filter

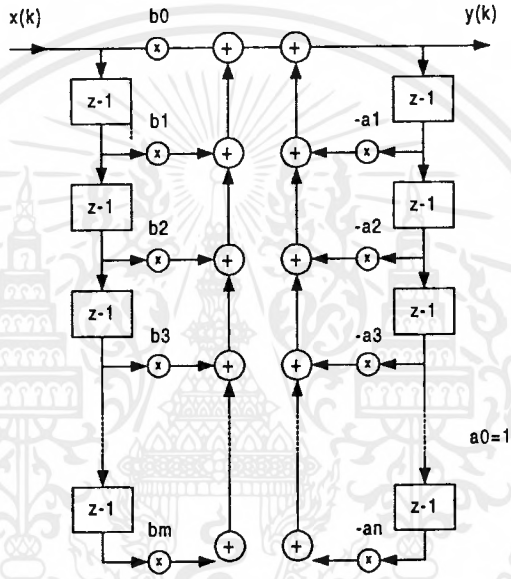
x(n) และ y(n) เป็น Input และ Output ของ Filter

Transfer Function สำหรับ IIR Filter ถูกกำหนดโดย

$$H(z)=\frac{a_0+a_1z^{-1}+a_2z^{-2}+...+a_nz^{-n}}{1+b_1z^{-1}+b_2z^{-2}+..b_mz^{-m}}=\frac{\sum_{k=0}^{n}a_k.z^{-k}}{1+\sum_{k=1}^{m}b_k.z^{-k}} \qquad (10)$$

ส่วนที่สำคัญในการออกแบบ IIR Filter คือการค้นหาค่าที่เหมาะสม สำหรับค่าสัมประสิทธิ์ และให้สอดคล้องกับลักษณะของ ตัวกรองสัญญาณ ใน Frequency Response ดังนั้นสมการดังกล่าวทั้งสอง จึงเป็น Characteristic Equation สำหรับ IIR Filter

จากสมการแรก กล่าวได้ว่ากระแสการสุ่มตัวอย่าง Output , y(n) เป็น Function ของ Output ที่ผ่านมา y(n) ซึ่งจะเท่ากับค่าการสุ่มตัวอย่าง Input ทั้งในปัจจุบันและอดีต นั่นคือ IIR Filter เป็นระบบ Feedback ซึ่งปกติแล้ว IIR Filter ต้องการเพียงสัมประสิทธิ์ 2-3 ตัวมากกว่า FIR Filter สำหรับข้อกำหนดเดียวกัน ซึ่ง IIR Filter จะถูกใช้เมื่อจุด Cut Off ที่แคบและมี Throughput ที่สูง เป็นความต้องการที่สำคัญ เหตุผลที่จะทำให้ IIR Filter ไม่คงที่ (Unstable).



รูปที่ 7 โครงสร้าง IIR Digital Filter

8. ADAPTIVE DIGITAL FILTER

Adaptive Digital filter เป็นสิ่งจำเป็นในการกรองความถี่แบบ Digital ที่มีลักษณะการปรับค่าต่างๆ ด้วยตัวเอง โดยทำการปรับค่าอย่างอัตโนมัติ เพื่อที่จะเป็นค่าของสัญญาณ Input ต่อไป

Adaptive Filter เราหมายถึงอุปกรณ์ซึ่งออกแบบตัวมันเองในความคิดที่ว่า มันจะบรรจุกลุ่มของ Parameters ที่ปรับค่าได้ และค่านั้นจะถูกกำหนดโดยอัตโนมัติ ขึ้นอยู่กับการประมาณค่าคุณสมบัติทางสถิติของสัญญาณที่เกี่ยวข้อง ฉะนั้นทฤษฎีของ Adaptive Filter จะมีความใกล้ชิดกับการออกแบบ ตัวกรองสัญญาณ ที่ดีที่สุด ในปัญหาการออกแบบต้องการที่จะหากลุ่มที่ดีที่สุดของ Filter Parameters จาก

ความรู้ของคุณสมบัติของสัญญาณที่เกี่ยวข้อง ให้เหมาะสมกับบรรทัดฐาน หรืออีกอย่างหนึ่ง ปัญหาของ Adaptive Filter คือ ความต้องการที่จะหา Algorithm (ขบวนการในการแก้ปัญหา) สำหรับปรับค่า Filter Parameter ในสถานการณ์ที่มีความรู้เกี่ยวกับคุณสมบัติของสัญญาณที่เกี่ยวข้องไม่สมบูรณ์ ดังนั้นการทำงานของ Adaptive Filter จะลู่เข้าสู่ลักษณะ ตัวกรองสัญญาณ ที่ดีที่สุด หลังจากผ่านขบวนการซ้ำๆ เกี่ยวกับจำนวนตัวเลขมากมายของ Algorithm

Adaptive Filter อาจจะมีลักษณะแตกต่างกันดังนี้

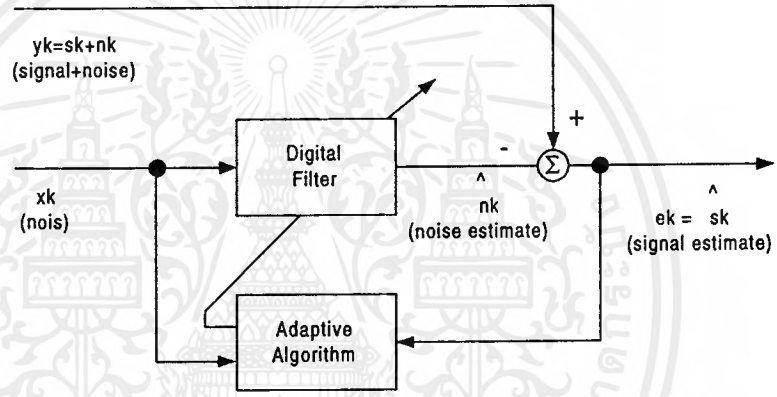
1. Adaptive Filter ลักษณะ Open Loop หรือไม่กระทำซ้ำ ประกอบด้วยขบวนการสองขั้นโดยที่ขั้นแรก คือ การเรียนรู้สถิติของสัญญาณที่เกี่ยวข้อง แล้วนำผลที่ได้ใส่เข้าไปใน Algorithm ที่ไม่มีการกระทำซ้ำ (Non-Recursive Algorithm) ลักษณะเช่นนี้จะมีข้อเสียที่ต้องการความละเอียดมากและ Hardware ราคาแพง
2. Adaptive Filter ลักษณะ Closed Loop หรือกระทำซ้ำ ที่เรื่องของสถิติไม่ได้ถูกประมาณอย่างเด่นชัด แต่ว่าการออกแบบของ Adaptive Filter จะทำได้ในขบวนการเดียวโดยการเฉลี่ยของ Algorithm ที่กระทำซ้ำ (Recursive Algorithm) ซึ่งจะเปลี่ยนค่า Filter Parameters โดยอัตโนมัติด้วยข้อมูลที่สุ่มเข้ามาใหม่ในแต่ละครั้ง ในแต่ละการกระทำซ้ำของ Algorithm Adaptive Filter จะค่อยๆ เรียนรู้เกี่ยวกับสถิติของสัญญาณที่เกี่ยวข้องและการปรับปรุงของค่าปัจจุบัน ของ Filter ที่เปลี่ยนค่า Parameters ได้ จะถูกคำนวณโดยใช้ข่าวสารใหม่นี้และ Adjustable Filter (Filter ที่เปลี่ยนค่า Parameters) ค่า Parameters ของมันจะเพิ่มขึ้นและการกระทำซ้ำครั้งต่อไปจะมีฐานจากการปฏิบัติการของ ตัวกรองสัญญาณ ด้วยกลุ่มของค่าที่ปรับปรุงแล้วในลักษณะของ Closed loop การเรียนรู้และการปฏิบัติการคำนวณจะรวมกันเป็นขบวนการเดียวกัน ซึ่งดูเหมือนว่าจะมีเครื่องมือที่ง่ายกว่าลักษณะ Open Loop

Adaptive Digital Filter มีคุณสมบัติใน Frequency Response จะทำการปรับหรือแก้ไขโดยอัตโนมัติ เพื่อที่จะปรับปรุงโครงสร้างให้สอดคล้องกับเกณฑ์บางส่วนในการที่จะให้ ตัวกรองสัญญาณทำการปรับและเปลี่ยนค่าลักษณะสัญญาณ Input เนื่องจากการปรับค่าตัวเอง และมีความง่ายในการสร้าง Adaptive Digital Filter จะพบว่าถูกนำมาใช้ในการหลายๆ อย่าง อาทิเช่น Telephone Echo

Canceling , Radar Signal Processing , Navigator System Equation of Communication channels และ Biomedical Signal Enhancement

9. CONCEPTS OF ADAPTIVE DIGITAL FILTER

Adaptive Filter ประกอบด้วยสองส่วนที่แตกต่างกัน Digital Filter กับ สัมประสิทธิ์ที่ใช้ในการปรับค่า และ Adaptive Algorithm ซึ่งจะใช้ในการปรับค่าหรือปรับปรุงค่าสัมประสิทธิ์ของ ตัวกรองสัญญาณ ดังรูป



รูปที่ 8 Adaptive Noise Canceler

สัญญาณ Input 2 สัญญาณ y_k และ x_k ถูกป้อนอย่างต่อเนื่องไปยัง Adaptive Filter สัญญาณ y_k ประกอบด้วยสัญญาณที่ต้องการ (s_k) และ Noise ปะปนอยู่ สมมุติว่าสัญญาณทั้งสองเกี่ยวพันซึ่งกันและกัน โดยที่ x_k เป็นสัญญาณที่ใช้ในการตรวจสอบสัญญาณที่ปะปนมา ซึ่งจะเกี่ยวพันกับ Noise (n_k) สัญญาณ x_k จะเป็นสัญญาณที่ผ่านกระบวนการ Digital Filter เพื่อที่จะประมาณค่าของ Noise ซึ่งให้เป็น n_k' ของ n_k ดังนั้น ค่าประมาณของสัญญาณที่ต้องการจะหาได้จากการหักล้างของ Output ของ Digital Filter จากสัญญาณ y_k

$$s_k' = y_k - n_k' = s_k + n_k - n_k'$$

วัตถุประสงค์ที่สำคัญใน Noise Canceling ก็เพื่อจะสร้างค่าประมาณที่ดีที่สุดของ Noise ใน s_k สัญญาณที่ปะปนกัน และค่าประมาณที่ดีที่สุดของสัญญาณซึ่งจะทำให้ได้โดยการใช้ s_k' ในระบบการ Feedback เพื่อทำการปรับค่าสัมประสิทธิ์ของ Digital Filter , Adaptive Algorithm ที่เหมาะสมเพื่อลด Noise ให้มากที่สุดใน s_k' สัญญาณ Output s_k' จะนำไปใช้สองส่วนคือ

1. ใช้ในการประมาณค่าสัญญาณที่ต้องการ
2. สัญญาณที่ผิดพลาดจะถูกใช้ในการปรับค่าสัมประสิทธิ์ของ ตัวกรองสัญญาณ

5. การเปลี่ยนแปลงค่าของ $\alpha_1(k)$ จะเร็วขึ้นได้ โดยการปรับค่า α_0 ให้เปลี่ยนแปลงไปด้วยวิธี การ Stochastic Gradient เนื่องจากว่า $\alpha_1(k)$ จะติดตามความถี่ได้เร็วหรือไม่นั้นขึ้นอยู่กับค่าของ $y(k)\cdot\psi(k)$ เรียกว่า Update Function และค่าของ $y(k)$ และ $\psi(k)$ ต่างก็มาจาก $H(z)$ ดังนั้น $[y(k)\cdot\psi(k)]^2$ จะเป็นสัดส่วนโดยตรงกับ $\frac{\partial[H(\partial\alpha_1(k))]^2}{\partial\alpha_1(k)}$ เพื่อใช้วิธีการ Stochastic Gradient จะต้องพิสูจน์เทอม $\frac{\partial[H(\partial\alpha_1(k))]^2}{\partial\alpha_1(k)}$ เมื่อ Plot กับ α_0 แล้วต้องมียอดสูงสุดของกราฟเพียงยอดเดียว
6. จากกราฟมียอดสูงสุดยอดเดียวจึงสามารถใช้วิธี Stochastic Gradient ได้ Adaptive Algorithm สำหรับ $\alpha_0(k)$ สามารถกำหนดได้ดังสมการ

$$\alpha_0(k+1)=\alpha_0(k)+\frac{\mu_0}{2}\frac{\partial[y(k)\cdot\psi(k)]^2}{\partial\alpha_0(k)}$$

$$\alpha_0(k+1)=\alpha_0(k)+\mu_0 y(k)\psi(k)[\psi(k)\psi_{01}(k)+\psi_{02}(k)y(k)]$$

โดยที่ $\psi_{01}(k)=\frac{\partial y(k)}{\partial\alpha_0(k)}$ และ $\psi_{02}(k)=\frac{\partial\psi(k)}{\partial\alpha_0(k)}$

ψ_{01},ψ_{02} : เป็นสัญญาณควบคุมแบบ Adaptive ของ $\alpha_0(k)$

μ_0 : เป็น Step Size Parameters

7. จากสมการ $\alpha_0(k+1)$ จะเห็นว่าต้องใช้สัญญาณควบคุมแบบอะแดปทีฟ 2 ตัวคือ $\psi_{01}(k),\psi_{02}(k)$ และจากสมการ $\psi_{01}(k)$ ต้องใช้สัญญาณอินพุท 2 ตัว คือ $x(k),y(k)$ เพื่อสร้างสัญญาณควบคุมแบบ Adaptive $\psi_{01}(k)$ จากสมการ $\psi_{02}(k)$ ต้องใช้สัญญาณอินพุท 3 ตัว คือ $y(k),\psi(k),\psi_{01}(k)$ เพื่อสร้างสัญญาณควบคุมแบบ Adaptive $\psi_{02}(k)$ ซึ่งวงจรที่ได้ยังซับซ้อนและยากในการออกแบบ แสดงได้ใน Adaptive Algorithm IIR Band-Pass Filter
8. สำหรับ $\alpha_0(k)$ ที่จะทำให้ $\alpha_1(k)$ มีค่าสูงสุด จะหาได้จาก $\alpha_{0,opt}=\frac{\partial H_1(\alpha_0(k))}{\partial\alpha_0(k)}=0$

$$\alpha_{0,opt}(k)=\frac{\sin(\omega)-\cos(\omega)+\alpha_1(k)}{\sin(\omega)+\cos(\omega)-\alpha_1(k)} \quad \text{เมื่อ } \alpha_1(k)<\cos(\omega)$$

$$\alpha_{0,opt}(k)=\frac{\sin(\omega)+\cos(\omega)-\alpha_1(k)}{\sin(\omega)-\cos(\omega)+\alpha_1(k)} \quad \text{เมื่อ } \alpha_1(k)>\cos(\omega)$$

9. ฟังก์ชันของ $\psi(k),\psi_{01}(k),\psi_{02}(k)$ จะขึ้นอยู่กับฟังก์ชันหลายตัว ไม่ขึ้นกับฟังก์ชันตัวใดตัวหนึ่ง ถ้าสามารถเปลี่ยนฟังกชันเหล่านี้ให้ขึ้นอยู่กับฟังก์ชันตัวใดตัวหนึ่งได้ ก็จะทำให้วงจรมีความง่าย ขึ้น ซึ่งทำได้โดยการหา Transfer Function ของฟังก์ชันเหล่านี้

TRANSFER FUNCTION ของ $\psi(k)$ เขียนสัญลักษณ์ด้วย $G(z)=\frac{\psi(z)}{y(z)}$

TRANSFER FUNCTION ของ $\psi_{01}(k)$ เขียนสัญลักษณ์ด้วย $G_{01}(z)=\frac{\psi_{01}(z)}{y(z)}$

TRANSFER FUNCTION ของ $\psi_{02}(k)$ เขียนสัญลักษณ์ด้วย $G_{02}(z)=\frac{\psi_{02}(z)}{\psi(z)}$

เมื่อได้ Transfer Functions ทั้ง 3 แล้ว ก็จะนำไปเขียนไดอะแกรมได้ เนื่องจากฟังก์ชันแต่ละตัวขึ้นอยู่กับฟังก์ชันใดฟังก์ชันหนึ่งเท่านั้น

10. คุณลักษณะของ $|H(\alpha_0)|$ เทียบกับ α_0 จากรูปที่ 2 จะเห็นได้ว่ากราฟไม่มียอดสูงสุด ดังนั้นค่าสูงสุดของค่าเฉลี่ยของ $y(k)*\psi(k)$ สามารถประมาณให้เท่ากับค่าเฉลี่ยของ $\psi(k)$ ได้ ดังนั้นค่าเฉลี่ยของ $\psi(k)$ เป็นสัดส่วนโดยตรงกับ $\frac{\partial|H(\alpha_1(k))|^2}{\partial\alpha_1(k)}$

11. จากสมการ $\frac{\partial|H(\alpha_1(k))|}{\partial\alpha_1(k)}$ ซึ่งเป็นฟังก์ชันของ α_0 สามารถเขียนใหม่ให้อยู่ในรูปที่ง่ายขึ้นได้ ดังสมการ

$$\Pi(\alpha_0)=\frac{\partial|H(\alpha_1(k))|}{\partial\alpha_1(k)}$$

คุณลักษณะของ $|\Pi(\alpha_0)|$ เทียบกับ α_0 จากรูปที่ 3 จะเห็นได้ว่ามียอดสูงสุดเพียงยอดเดียวดังนั้นสามารถดัดแปลงหรือปรับปรุงอัลกอริทึมสำหรับ Update ของ $\alpha_0(k)$ ได้ดังสมการ

$$\alpha_0(k+1)=\alpha_0(k)+\frac{\mu_0}{2}\frac{\partial\psi(k)^2}{\partial\alpha_{0(k)}}$$

$$\alpha_0(k+1)=\alpha_0(k)+\mu_0\psi(k)\psi_{02}(k)$$

โดยที่ $\psi_{02}(k)=\frac{\partial\psi(k)}{\partial\alpha_0(k)}$

μ_0 : เป็น Step Size Parameter

และ $\psi_{02}(k)$ เป็นสัญญาณควบคุมแบบ Adaptive ของ $\alpha_0(k)$ จะเห็นได้ว่าใช้เพียงแค่ตัวเดียว สามารถแสดงได้ใน Simplified Adaptive Algorithm IIR Band-Pass Filter

1. ADAPTIVE ALGORITHM โดยการใช้ฟิลเตอร์แบบค่า Q - FACTOR คงที่

กำหนดให้ช่วงเวลาการแซมปลิ้ง (Sampling) เป็น T วินาที ถูกนอร์มัลไลซ์ (Normalized) อยู่ที่ 1 ตัวแปร k (เลขจำนวนเต็ม) เป็นเวลาการแซมปลิ้ง และสัญญาณทางด้านอินพุทเขียนได้ดังสมการ

$$X(k)=A\cdot\sin(k\cdot\omega)+n(k) \qquad (1)$$

โดยที่ $X(k)$ คือสัญญาณทางด้านอินพุท ซึ่งประกอบด้วยคลื่นซายน์

$A\cdot\sin(k\cdot\omega)$: A เป็นค่าคงที่

ω (rad/sec) : ความเร็วเชิงมุม มีค่าในช่วง $0<\omega<\pi$

$n(k)$: สัญญาณรบกวนแบบเกาส์

สำหรับ Transfer Function ของวงจรกรองความถี่ IIR Digital Band-Pass Filter อันดับสองที่มีความถี่กลางเป็น ω_0 คือ

$$H(z)=\left[\frac{1-\alpha_0}{2}\right]\left[\frac{1-z^{-2}}{1-\alpha_1(k).(1-\alpha_0).z^{-1}+\alpha_0 z^{-2}}\right] \qquad (2)$$

โดยที่ $\alpha_1(k)$ มีค่าระหว่าง $-1<\alpha_1(k)<1$ เป็นสัมประสิทธิ์แปรค่าที่ใช้สำหรับกำหนด ω_0 ซึ่ง

$$\alpha_1(k) = \cos(\omega_0) \tag{3}$$

α_0 มีค่าอยู่ระหว่าง $0 < \alpha_0 < 1$ เป็นสัมประสิทธิ์คงที่สำหรับกำหนดค่า Q-Factor ของวงจรกรองความถี่

เพื่อนำ Adaptive Algorithm ไปใช้งานจึงต้องมีการศึกษาค่า Parameters แต่ละตัวของ $H(z)$ โดยแทน $z = e^{j\omega}$ เมื่อ ω มีค่าระหว่าง $0 < \omega < \pi$ แล้ว Plot กราฟของ $|H(e^{j\omega})|$ เทียบกับแกน ω พบว่าค่าของ α_0 มีค่าอยู่ระหว่าง $0 < \alpha_0 < 1$ จะมีผลต่อการเปลี่ยนแปลงของ Q - Factor โดยมีความสัมพันธ์ที่แปรผันต่อกันดังนี้

1. เมื่อค่า α_0 เข้าใกล้ศูนย์ ค่า Q - Factor มีค่าน้อยลง กราฟของ $H(e^{j\omega})$ จะมีความกว้างมาก
2. เมื่อค่า α_0 เข้าใกล้หนึ่ง ค่า Q - Factor มีค่าสูงขึ้น กราฟของ $H(e^{j\omega})$ จะมีความกว้างน้อยลง ค่าของ α_1 จะมีผลทำให้ความถี่กลาง ω_0 เบี่ยงไปโดยที่
   2.1 ค่าของ $\alpha_1 = 0$ จะทำให้ ω_0 อยู่ตรงกลางพอดี
   2.2 ค่าของ α_1 เป็นลบ ความถี่ ω_0 จะเบี่ยงเบนไปทางขวา
   2.3 ค่าของ α_1 เป็นบวก ความถี่ ω_0 จะเบี่ยงเบนไปทางซ้าย

เพื่อหาการเปลี่ยนแปลงของ $\alpha_1(k)$ เมื่อมีสัญญาณอินพุทป้อนเข้ามา จึงต้องหาเอาท์พุท $y(k)$ เมื่อ $Y(z)$ เป็น Z-Transform ของ $y(k)$ ความสัมพันธ์ของ Transfer Function จะเป็น $H(z) = \frac{Y(z)}{X(z)}$ หรือ $Y(z) = H(z) \cdot X(z)$ เมื่อ

$$Y_{(z)} = H_{(z)} . X_{(z)} \tag{4}$$

$$Y_{(z)} = \left(\frac{1-\alpha_0}{2} \right) \left(\frac{1-z^{-2}}{1-\alpha_1(k)(1+\alpha_0)z^{-1}+\alpha_0 z^{-2}} \right) X_{(z)}$$

$$Y_{(z)}\left(1-\alpha_1(k)(1+\alpha_0)z^{-1}+\alpha_0 z^{-2}\right) = \left(\frac{1-\alpha_0}{2} \right)\left(1-z^{-2}\right) X_{(z)}$$

$$\left(Y_{(z)}-\alpha_1(k)(1+\alpha_0)Y_{(z)}z^{-1}+\alpha_0 Y_{(z)}z^{-2}\right)=\left(\frac{1-\alpha_0}{2}\right)\left(X_{(z)}-X_{(z)}z^{-2}\right) \quad (5)$$

TAKE INVERSE Z-TRANFORM

$$\left(y(k)-\alpha_1(k)(1+\alpha_0)y(k-1)+\alpha_0 y(k-2)\right)=\left(\frac{1-\alpha_0}{2}\right)\left(x(k)-x(k-2)\right)$$

ดังนั้นสามารถเขียนเอาท์พุทของวงจรกรองความถี่ให้อยู่ในรูปทางเวลาจะได้

$$y(k)=\left(\frac{1-\alpha_0}{2}\right)\left(x(k)-x(k-2)\right)+\alpha_1(k)(1+\alpha_0)y(k-1)-\alpha_0 y(k-2) \quad (6)$$

ค่าเฉลี่ยกำลังสอง ของ $[y(k)]^2$ จะเป็นสัดส่วนโดยตรงกับ $\left|H\left(e^{j\omega}\right)\right|^2$ ที่ $z=e^{j\omega}$ เมื่อใช้วิธีการของ Stochastic Gradient ค่า $\alpha_1(k)$ ที่ทำให้ $[y(k)]^2$ มีค่าสูงสุด Adaptive Algorithm สามารถกำหนดได้ดังสมการ

$$\alpha_1(k+1)=\alpha_1(k)+\frac{\mu_1}{2}\frac{\partial y^2(k)}{\partial \alpha_{1(k)}} \quad (7)$$

$$=\alpha_1(k)+\mu_1 y(k)\cdot\psi(k) \quad (8)$$

เมื่อ $\psi(k)=\frac{\partial y(k)}{\partial \alpha_1(k)}$ เป็นสัญญาณควบคุม ADAPTIVE ของ $\alpha_1(k)$

μ_1 เป็น Step Size Parameters ถ้ามีค่าเล็กพอจะทำให้การเปลี่ยนแปลงของ $y(k)$ ในแต่ละครั้งมีน้อย จึงสามารถประมาณสัญญาณควบคุม Adaptive ได้ดังนี้

$$\psi(k-i)=\frac{\partial y(k-i)}{\partial \alpha_1(k-i)}\approx\frac{\partial y(k-i)}{\partial \alpha_1(k)} \quad (9)$$

เมื่อ i เป็นจำนวนเต็มบวกใด ๆ

ดังนั้นจึงเขียนสมการได้ดังนี้

$$\psi(k)=\alpha_1(k)(1+\alpha_0)\psi(k-1)-\alpha_0\psi(k-2)+(1+\alpha_0)y(k-1) \qquad (10)$$

พิจารณาความเร็วในการ Converge ของ $\alpha_1(k)$ โดยการ Plot กราฟ $\alpha_1(k)$ เทียบกับแกนเวลา k จะพบว่า

1. ความละเอียดของค่า μ_1 จะส่งผลต่อการเกิด Noise ถ้าค่านี้มีความละเอียดมาก Noise จะลดลง แต่การ Converge ก็จะช้าลงด้วย
2. ความเร็วเชิงมุม ω_0 ที่มีค่ามากจะทำให้การ Converge เร็วกว่าค่า ω_0 ที่มีค่าน้อย
3. ความสูงของสัญญาณอินพุทที่เข้ามาก็มีผลต่อการ Converge โดยที่ถ้าความแรงของสัญญาณอินพุทมีมากก็จะทำให้การ Converge เร็วขึ้นด้วย
4. α_0 ค่ามาก(เข้าใกล้หนึ่ง) จะทำให้มีการแกว่งของสัญญาณ แต่การ Converge จะเร็วเมื่อ เข้าใกล้สถานะคงที่ และเมื่อค่า α_0 ค่าน้อย(เข้าใกล้ศูนย์) การ Converge ในช่วงเริ่มต้นเร็วและแทบไม่มีการแกว่งของสัญญาณ

2. ADAPTIVE ALGORITHM โดยใช้ฟิลเตอร์ที่สามารถปรับค่า Q-FACTOR ได้

การเปลี่ยนแปลงค่าของ $\alpha_1(k)$ จะเร็วขึ้นได้โดยการปรับค่า α_0 ให้เปลี่ยนแปลงไปด้วยวิธีการStochastic Gradient เนื่องจากว่า $\alpha_1(k)$จะติดตามความถี่ได้เร็วหรือไม่นั้นขึ้นอยู่กับค่าของ $y(k)*\psi(k)$เรียกว่า Update Function และค่าของ $y(k)$ และ $\psi(k)$ ต่างก็มาจาก $H(z)$ ดังนั้น $[y(k)*\psi(k)]^2$ ก็จะเป็นสัดส่วนโดยตรงกับ $\frac{\partial[H(\partial\alpha_1(k))]^2}{\partial\alpha_1(k)}$

เพื่อใช้วิธีการ Stochastic Gradient จะต้องพิสูจน์เทอม $\frac{\partial[H(\partial\alpha_1(k))]^2}{\partial\alpha_1(k)}$ เมื่อ Plot กับ α_0แล้วต้องมียอดสูงสุดของกราฟเพียงยอดเดียวเท่านั้น จึงต้องหาค่าของสมการนี้ เพื่อนำมาเขียนกราฟจาก

$$H_{(z)}=\left(\frac{1-\alpha_0}{2}\right)\left(\frac{1-z^{-2}}{1-\alpha_1(k)(1+\alpha_0)z^{-1}+\alpha_0 z^{-2}}\right)$$

$$= \frac{1}{\sqrt{\left[\left\{\frac{\cos(\omega)-\alpha_1(k)}{\sin(\omega)}\right\}\left\langle\frac{1+\alpha_0}{1-\alpha_1}\right\rangle\right]^2+1}} \tag{15}$$

ยกกำลังสองทั้งสองข้าง

$$\left|H\left(e^{j\omega}\right)\right|^2 = \left[\left\{\frac{(\cos(\omega)-\alpha_1(k)(1+\alpha_0))}{\sin(\omega)(1-\alpha_0)}\right\}^2+1\right]^{-1} \tag{16}$$

จะได้

$$H(\alpha_0) = \frac{\partial\left|H\left(e^{j\omega}\right)\right|^2}{\partial\alpha_1(k)} = 2\frac{\left\{\frac{1+\alpha_0}{\sin(\omega)(1-\alpha_0)}\right\}^2\{\cos(\omega)-\alpha_1(k)\}}{\left[\left\{\frac{(\cos(\omega)-\alpha_1(k)(1+\alpha_0)}{\sin(\omega)(1-\alpha_0)}\right\}^2+1\right]^2} \tag{17}$$

เมื่อนำไปเขียนกราฟ α_0 เปลี่ยนจาก 0 ถึง 1 พบว่ากราฟที่ได้จะมียอดสูงที่สุดเพียงหนึ่งยอดเท่านั้น ดังนั้นสามารถใช้ Stochastic Gradient เพื่อหาสมการเปลี่ยนแปลง α_0 ได้

Adaptive Algorithm สำหรับ $\alpha_0(k)$ สามารถกำหนดได้ดังสมการ

$$\alpha_0(k+1) = \alpha_0(k) + \frac{\mu_0}{2}\frac{\partial(y(k)\psi(k))^2}{\partial\alpha_0(k)} \tag{18}$$

$$\alpha_0(k+1) = \alpha_0(k) + \mu_0 y(k)\psi(k)[\psi(k.)\psi_{01}(k) + \psi_{02}(k).y(k)] \tag{19}$$

โดยที่ $\psi_{01}(k) = \frac{\partial y(k)}{\partial\alpha_0(k)}$ และ $\psi_{02}(k) = \frac{\partial\psi(k)}{\partial\alpha_0(k)}$

ψ_{01}, ψ_{02} : เป็นสัญญาณควบคุมแบบ Adaptive ของ $\alpha_0(k)$

μ_0 : เป็น Step Size parameter

จาก

$$y(k)=\left(\frac{1-\alpha_0}{2}\right)(x(k)-x(k-2))+\alpha_1(k)(1+\alpha_0)y(k-1)-\alpha_0 y(k-2)$$

จะได้

$$\begin{aligned}\psi_{01}(k)&=\frac{\partial y(k)}{\partial \alpha_0(k)}\\&=\frac{1}{2}(x(k-2)-x(k))+\alpha_1(k)y(k-1)-y(k-2)\\&\quad+\alpha_1(k)(1+\alpha_0(k))\psi_{01}(k-1)-\alpha_0(k)\psi_{01}(k-2)\end{aligned} \tag{20}$$

และจาก

$$\psi(k)=\alpha_1(k)(1+\alpha_0)\psi(k-1)-\alpha_0\psi(k-2)+(1+\alpha_0)y(k-1)$$

จะได้

$$\begin{aligned}\psi_{02}(k)&=\frac{\partial \psi(k)}{\partial \alpha_0(k)}\\&=\alpha_1(k)[1+\alpha_0(k)]\psi_{02}(k-1)+\alpha_1(k)\psi(k-1)-\alpha_0(k)\psi_{02}(k-2)\\&\quad-\psi(k-2)+[1+\alpha_0(k)]\psi_{01}(k-1)+y(k-1)\end{aligned} \tag{21}$$

เมื่อนำทั้งสองฟังก์ชันไปแทนใน $\alpha_0(k)$ สำหรับค่าของ μ_0 ในฟังก์ชัน $\alpha_0(k)$ จะต้องมีค่ามากกว่าค่า μ_0 ของฟังก์ชัน $\alpha_1(k)$ เพื่อที่จะทำให้การลู่เข้า (Converge) ของฟังก์ชัน $\alpha_0(k)$ เร็วกว่า $\alpha_1(k)$ แต่ต้องคำนึงถึงเสถียรภาพ เพราะถ้าค่า μ ทั้งสองแตกต่างกันมากเกินไปจะทำให้มีริบเปิล (Ripple) เกิดขึ้น เมื่อฟังก์ชัน $\alpha_1(k)$ Converge และค่าของ $\alpha_0(k)$ ที่หาได้จากฟังก์ชันนี้นั้น จะมีค่าอยู่ระหว่าง $-1<\alpha_0(k)<1$ แต่ว่าค่า $\alpha_0(k)$ ที่ใช้กับ IIR Band-Pass Filter สำหรับโครงงานนี้จะมีค่าอยู่ระหว่าง $0<\alpha_0(k)<1$ เท่านั้น จึงต้องมีการควบคุมค่านี้

จากสมการ (19) จะเห็นว่าต้องใช้สัญญาณควบคุมแบบอะแดปทีฟ 2 ตัวคือ $\psi_{01}(k),\psi_{02}(k)$ และจากสมการที่ (20) ต้องใช้สัญญาณอินพุท 2 ตัว คือ $x(k), y(k)$ เพื่อสร้างสัญญาณควบคุมแบบ Adaptive $\psi_{01}(k)$ จากสมการที่ (21) ต้องใช้สัญญาณอินพุท 3 ตัว คือ $y(k),\psi\ (k),\psi_{01}(k)$ เพื่อสร้างสัญญาณควบคุมแบบ Adaptive $\psi_{02}(k)$ ซึ่งวงจรที่ได้ยังซับซ้อนและยากในการออกแบบ

สำหรับ $\alpha_0(k)$ ที่จะทำให้ $\alpha_1(k)$ มีค่าสูงสุด จะหาได้จาก $\alpha_{0,opt} = \dfrac{\partial H_1(\alpha_0(k))}{\partial \alpha_0(k)} = 0$

จาก

$$H(\alpha_0) = \frac{\partial \left| H\left(e^{j\varpi}\right)\right|^2}{\partial \alpha_1(k)} = 2 \cdot \frac{\left\{\dfrac{(1+\alpha_0)}{\sin(\omega)(1-\alpha_0)}\right\}^2 \{\cos(\omega)-\alpha_1(k)\}}{\left[\left\{\dfrac{(\cos(\omega)-\alpha_1(k))(1+\alpha_0)}{\sin(\omega)(1-\alpha_0)}\right\}^2 + 1\right]^2}$$

กำหนดให้ $h(\alpha_0)=\dfrac{1+\alpha_0}{1-\alpha_0}, A=\dfrac{1}{\sin(\omega)}$ และ $B=\{\cos(\omega)-\alpha_1(k)\}$

จะได้

$$H(\alpha_0) = \frac{2 \cdot \mathrm{A}^2 h^2(\varepsilon_0)\mathrm{B}}{\left\{\mathrm{A}^2\mathrm{B}^2 h^2(\alpha_0)+1\right\}^2} = 0$$

ดังนั้น

$$\frac{\partial}{\partial \alpha_0(k)}\left[\frac{\mathrm{A}^2 h^2(\alpha_0)B}{\left\{\mathrm{A}^2\mathrm{B}^2 h^2(\alpha_0)+1\right\}^2}\right] = 0$$

$$2.A^2B\frac{\cdot\left[\begin{array}{c}\left(A^2B^2h^2(\alpha_0)+1\right)^2(2.h(\alpha_0)-h'(\alpha_0))-h^2(\alpha_0)\\ \{-2.(A^2B^2h^2(\alpha_0)+1)(A^2B^2 2.h(\alpha_0)h'(\alpha_0))\}\end{array}\right]}{(A^2B^2h^2(\alpha_0)+1)^4}=0$$

$$2.A^2B\frac{\cdot\left[\begin{array}{c}2.\left(A^2B^2h^2(\alpha_0)+1\right)^2(h(\alpha_0)-h'(\alpha_0))\\ -4.(A^2B^2h^2(\alpha_0)+1)h^3(\alpha_0)A^2B^2h'(\alpha_0)\end{array}\right]}{(A^2B^2h^2(\alpha_0)+1)^4}=0$$

$$2.A^2B\frac{\left[\begin{array}{c}2.\left(A^2B^2h^2(\alpha_0)+1\right)^2(h(\alpha_0)-h'(\alpha_0))\\-4.(A^2B^2h^2(\alpha_0)+1)h^3(\alpha_0)A^2B^2h'(\alpha_0)\end{array}\right]}{(A^2B^2h^2(\alpha_0)+1)^4}=0$$

$$2.A^2B\frac{\left[\quad 2.h(\alpha_0)h'(\alpha_0)\{A^2B^2h^2(\alpha_0)+1-2.A^2B^2h^2(\alpha_0)\}\right]}{(A^2B^2h^2(\alpha_0)+1)^3}=0$$

$$\frac{\left[4.A^2B^2h(\alpha_0)h'(\alpha_0)(1-A^2B^2h^2(\alpha_0))\right]}{(A^2B^2h^2(\alpha_0)+1)^3}=0$$

จะมีอยู่ 2 กรณีคือ

$$4\cdot A^2\cdot B\cdot h(\alpha_0)h'(\alpha_0)=0$$

และ

$$1-A^2B^2h^2(\alpha_0)^2=0$$

จาก

$$4\cdot A^2\cdot B\cdot h(\alpha_0)h'(\alpha_0)=0$$

$$h(\alpha_0)h'(\alpha_0)=0$$

$$h'(\alpha_0)=\frac{\partial h(\alpha_0)}{\partial\alpha_0}=\frac{\partial}{\partial\alpha_0}\left(\frac{1+\alpha_0}{1-\alpha_0}\right)=\frac{2}{(1-\alpha_0)^2}$$

$$\therefore h'(\alpha_0)=\frac{2}{(1-\alpha_0)^2}=0$$

จะได้

$$\left[\frac{(1+\alpha_0)}{(1-\alpha_0)}\right]\frac{2}{(1-\alpha_0)^2}=0$$

$$\left[\frac{\left(1+\alpha_0{}^2\right)^2}{(1-\alpha_0)^2}\right]=0$$

$$1-\alpha_0{}^2=0$$

$$\therefore \alpha_0=\pm 1$$

และจาก

$$1-A^2B^2h^2(\alpha_0)=0$$
$$A^2B^2h^2(\alpha_0)=1$$
$$A^2B^2\left[\frac{1+\alpha_0}{1-\alpha_0}\right]^2=1$$
$$A^2B^2(1+\alpha_0)^2=(1-\alpha_0)^2$$
$$A\cdot B\cdot(1+\alpha_0)=(1-\alpha_0)$$
$$A\cdot B.(1+\alpha_0)=\pm(1-\alpha_0)$$

กรณีที่ 1

$$A\cdot B\cdot(1+\alpha_0)=(1-\alpha_0)$$
$$A\cdot B+A\cdot B\cdot\alpha_0=1-\alpha_0$$
$$\alpha_0=\frac{1-A\cdot B}{1+A\cdot B}$$

$$A\cdot B=\frac{\cos(\omega)-\alpha_1(k)}{\sin(\omega)}$$
$$\alpha_0=\frac{1-\dfrac{\cos(\omega)-\alpha_1(k)}{\sin(\omega)}}{1+\dfrac{\cos(\omega)-\alpha_1(k)}{\sin(\omega)}}$$
$$\therefore \quad \alpha_0=\frac{\sin(\omega)-\cos(\omega)+\alpha_1(k)}{\sin(\omega)+\cos(\omega)-\alpha_1(k)}$$

กรณีที่ 2

$$A \cdot B \cdot (1+\alpha_0) = (1-\alpha_0)$$

$$A \cdot B + A \cdot B \cdot \alpha_0 = 1 - \alpha_0$$

$$\alpha_0 = \frac{1 + A \cdot B}{1 - A \cdot B}$$

$$\alpha_0 = \frac{1 + \dfrac{\cos(\omega) - \alpha_1(k)}{\sin(\omega)}}{1 - \dfrac{\cos(\omega) - \alpha_1(k)}{\sin(\omega)}}$$

$$\therefore \quad \alpha_0 = \frac{\sin(\omega) + \cos(\omega) - \alpha_1(k)}{\sin(\omega) - \cos(\omega) + \alpha_1(k)}$$

ทดสอบโดยการกำหนดให้ $\alpha_1(k) > \cos(\omega)$ และ $\alpha_1(k) < \cos(\omega)$ พบว่า

$$\alpha_{0.opt}(k) = \frac{\sin(\omega) - \cos(\omega) + \alpha_1(k)}{\sin(\omega) + \cos(\omega) - \alpha_1(k)} \quad \text{เมื่อ} \quad \alpha_1(k) < \cos(\omega) \tag{22}$$

$$\alpha_{0.opt}(k) = \frac{\sin(\omega) + \cos(\omega) - \alpha_1(k)}{\sin(\omega) - \cos(\omega) + \alpha_1(k)} \quad \text{เมื่อ} \quad \alpha_1(k) > \cos(\omega) \tag{23}$$

ลักษณะฟังก์ชันของ $\psi(k), \psi_{01}(k), \psi_{02}(k)$ จะขึ้นอยู่กับฟังก์ชันหลายตัว ไม่ขึ้นกับฟังก์ชันตัวใดตัวหนึ่ง ถ้าสามารถเปลี่ยนฟังกชันเหล่านี้ให้ขึ้นอยู่กับฟังก์ชันตัวใดตัวหนึ่งได้ ก็จะทำให้วงจรมีความง่ายขึ้น ซึ่งทำได้โดยการหา Transfer Function ของฟังก์ชันเหล่านี้

Transfer Function $\psi(k)$ เขียนสัญลักษณ์ด้วย $G(z)$ ในรูป Z-Transform หาได้จาก

$$\psi(k) = \alpha_1(k) \cdot (1+\alpha_0(k))\psi(k-1) - \alpha_0(k)\psi(k-2) + (1+\alpha_0(k)) \cdot y(k-1)$$

TAKE Z-TRANFORM

$$\psi(z) = \alpha_1(k) \cdot (1+\alpha_0(k)) \cdot \psi(z) \cdot z^{-1} - \alpha_0(k)\psi(z) \cdot z^{-2} + (1+\alpha_0(k)) \cdot y(z) \cdot z^{-1}$$

$$\psi(z) = \left[1 - \alpha_1(k)(1+\alpha_0(k))z^{-1} + \alpha_0(k)z^{-2}\right] = (1+\alpha_0(k))z^{-1}y(z)$$

$$G(z)=\frac{\psi_{(z)}}{Y_{(z)}}=\frac{(1+\alpha_0(k))z^{-2}}{1-\alpha_1(k)\cdot(1+\alpha_0(k))\cdot z^{-1}+\alpha_0(k)z^{-2}} \tag{24}$$

Transfer Function ของ $\psi_{01}(k)$เขียนสัญลักษณ์ด้วย $G_{01}(z)$ หาได้จาก

$$\psi_{01}(k)=\frac{1}{2}[x(k-2)-x(k)]+\alpha_1(k)y(k-1)-y(k-2)+ \\ \alpha_1(k)[1+\alpha_0(k)]\psi_{01}(k-1)-\alpha_0(k)\cdot\psi_{01}(k-2) \tag{25}$$

เปลี่ยน $x(k)$ ให้อยู่ในรูปของ $y(k)$

$$y(k)=\left(\frac{1-\alpha_0}{2}\right)[x(k)-x(k-2)]+\alpha_1(k)(1+\alpha_0)y(k-1)-\alpha_0 y(k-2)$$

$$\frac{1}{2}[x(k-2)-x(k)]=\frac{-y(k)}{1-\alpha_0(k)}+\frac{1+\alpha_0(k)}{1-\alpha_0(k)}\alpha_1(k)y(k-1) \\ -\frac{\alpha_0(k)}{1-\alpha_0(k)}y(k-2) \tag{26}$$

หา $\frac{\psi_{01}(k)}{y(k)}=G_{01}(z)$

$$\psi_{01}(k)-\alpha_1(k)\cdot[1+\alpha_0(k)]\psi_{01}(k-1)+\alpha_0(k)\psi_{01}(k-2)=$$

$$\frac{-y(k)}{1-\alpha_0(k)}+\frac{1+\alpha_0(k)}{1-\alpha_0(k)}\alpha_1(k)y(k-1)+\frac{\alpha_0(k)}{1-\alpha_0(k)}y(k-2) \tag{27}$$

$$+\alpha_1(k)y(k-1)-y(k-2)$$

TAKE Z-TRANSFORM

$$\psi_{01}(z)\left[1-\alpha_1(k)(1+\alpha_0(k))z^{-1}+\alpha_0(k)z^{-2}\right]=$$

$$y(z)\cdot\left[\frac{1+\alpha_0(k)}{1-\alpha_0(k)}\alpha_1(k)z^{-1}-\frac{1+\alpha_0(k)\cdot z^{-2}}{1-\alpha_0(k)}+\frac{\alpha_1(k)\cdot(1-\alpha_0(k))}{1-\alpha_0(k)}z^{-1}-\frac{1+\alpha_0(k)}{1-\alpha_0(k)}z^{-2}\right]$$

$$G_{01}(z)=\frac{\psi_{01}(z)}{y(z)}=\frac{2\cdot\alpha_1(k)\cdot z^{-1}-z^{-2}-1}{(1-\alpha_0(k))\left[1-\alpha_1(k)\cdot(1+\alpha_0(k))z^{-1}+\alpha_0(k)z^{-2}\right]} \tag{28}$$

Transfer Function ของ $\psi_{02}(k)$เขียนสัญลักษณ์ด้วย $G_{02}(z)$หาได้จาก

$$\psi_{02}(k)=\alpha_1(k)[1+\alpha_0(k)]\psi_{02}(k-1)+\alpha_1(k)\psi(k-1)-\alpha_0(k)\psi_{02}(k-2)-\psi(k-2)$$
$$+[1+\alpha_0(k)]\psi_{01}(k-1)+y(k-1)] \tag{29}$$

เปลี่ยน $\psi_{01}(k), y(k)$ ให้อยู่ในเทอมของ $\psi(k)$
จาก

$$\psi(k)=\alpha_1(k)\cdot(1+\alpha_0(k))\cdot\psi(k-1)-\alpha_0(k)\psi(k-2)+(1+\alpha_0(k))\cdot y(k-1)$$

จะได้

$$y(k-1)=1+\frac{\psi(k)}{1+\alpha_0(k)}-\alpha_1(k)\cdot\psi(k-1)+\frac{\alpha_0(k)}{1+\alpha_0(k)}\psi(k-2) \tag{30}$$

นำไปแทนใน $\psi_{02}(k)$เป็น

$$\psi_{02}(k) = \alpha_1(k)\cdot[1+\alpha_0(k)]\psi_{02}(k-1)+\alpha_1(k)\cdot\psi(k-1)$$

$$-\alpha_0(k)\cdot\psi_{02}(k-2)-\psi(k-2)+[1+\alpha_0(k)]\psi_{01}(k-1)$$
$$+\frac{\psi(k)}{1+\alpha_0(k)}-\alpha_1(k)\cdot\psi(k-1)+\frac{\alpha_0(k)}{1+\alpha_0(k)}\psi(k-2) \quad (31)$$

TAKE Z-TRANSFORM

$$\psi_{02}(z)=\alpha_1(k)\cdot[1+\alpha_0(k)]\psi_{02}(z)z^{-1}+\alpha_1(k)\cdot\psi(z)z^{-1}$$
$$-\alpha_0(k)\cdot\psi_{02}(z)z^{-2}-\psi(z)z^{-2}+[1+\alpha_0(k)]\psi_{01}(z)z^{-1} \quad (32)$$
$$+\frac{\psi(z)}{1+\alpha_0(k)}-\alpha_1(k)\cdot\psi(z)z^{-1}+\frac{\alpha_0(k)}{1+\alpha_0(k)}\psi(z)z^{-2}$$

$$\psi_{02}(z)\left[1-\alpha_1(k)\cdot(1+\alpha_0(k))z^{-1}+\alpha_0(k)\cdot z^{-2}\right]$$
$$=\alpha_1(k)\cdot\psi(z)z^{-1}-\psi(z)z^{-2}+\frac{\psi(z)}{1+\alpha_0(k)}-\alpha_1(k)\cdot\psi(z)z^{-1} \quad (33)$$
$$+\frac{\alpha_0(k)}{1+\alpha_0(k)}\psi(z)z^{-2}+(1+\alpha_0(k))\psi_{01}(z)z^{-1}$$

$$\psi_{01}(z)=\frac{2\cdot\alpha_1(k)\cdot z^{-1}-z^{-2}-1}{(1-\alpha_0(k))\left[1-\alpha_1(k)\cdot(1+\alpha_0(k))z^{-1}+\alpha_0(k)z^{-2}\right]}\times y(z) \quad (34)$$

$$y(z)z^{-1}=\frac{\psi(z)}{1+\alpha_0(k)}-\alpha_1(k)\cdot\psi(z)z^{-1}+\frac{\alpha_0(k)}{1+\alpha_0(k)}\psi(z)z^{-2} \quad (35)$$

$$\psi_{01}(z)=\frac{\left[2\cdot\alpha_1(k)\cdot z^{-1}-z^{-2}-1\right]\cdot\left[1-\alpha_1(k)\cdot(1+\alpha_0(k))z^{-1}+\alpha_0(k)z^{-2}\right]\cdot\psi(z)}{\left(1-\alpha_0{}^2(k)\right)\left[1-\alpha_1(k)\cdot(1+\alpha_0(k))z^{-1}+\alpha_0(k)z^{-2}\right]\cdot z^{-1}} \quad (36)$$

$$\psi_{01}(z)=\frac{2\cdot\alpha_1(k)\cdot z^{-1}-z^{-2}-1}{\left(1-\alpha_0{}^2(k)\right)\cdot z^{-1}} \quad (37)$$

เอา ψ_{01} ที่ได้ไปแทนลงใน ψ_{02} จะได้

$$\psi_{02}(z)\left[1-\alpha_1(k)\cdot(1+\alpha_0(k))z^{-1}+\alpha_0(k)\cdot z^{-2}\right]=\alpha_1(k)\cdot\psi(z)z^{-1}-\psi(z)z^{-2}+\frac{\psi(z)}{1+\alpha_0(k)}$$

$$-\alpha_1(k)\cdot\psi(z)z^{-1}+\frac{\alpha_0(k)}{1+\alpha_0(k)}\psi(z)z^{-2}+\frac{2\cdot\alpha_1(k)\cdot z^{-1}-z^{-2}-1}{(1-\alpha_0(k))}\psi(z)$$

$$=\left[\alpha_1(k)\cdot z^{-1}-z^{-2}+\frac{1}{1+\alpha_0(k)}-\alpha_1(k)\cdot z^{-1}+\frac{\alpha_0(k)}{1+\alpha_0(k)}z^{-2}+\frac{2\cdot\alpha_1(k).\left(z^{-1}-z^{-2}-1\right)}{(1-\alpha_0(k))}\right]\cdot\psi(z)$$

$$\frac{\psi_{02}(z)}{\psi(z)}=\frac{\left[\begin{array}{l}-\left(1-\alpha_0{}^2(k)\right)z^{-2}+1-\alpha_0(k)+\alpha_0(k)(1-\alpha_0(k))z^{-2}+2\cdot\alpha_1(k)(1+\alpha_0(k))z^{-1}-\\ (1+\alpha_0(k))z^{-2}-1-\alpha_0(k)\end{array}\right]}{\left(1-\alpha_0{}^2(k)\right)\left[1-\alpha_1(k)\cdot(1+\alpha_0(k))z^{-1}+\alpha_0(k)\cdot z^{-2}\right]} \tag{38}$$

$$G_{02}(z)=\frac{\psi_{02}(z)}{\psi(z)}=2\cdot\frac{-\alpha_0(k)+\alpha_1(k)\cdot(1+\alpha_0(k))z^{-1}-z^{-2}}{\left(1-\alpha_0{}^2(k)\right)\left[1-\alpha_1(k)\cdot(1+\alpha_0(k))z^{-1}+\alpha_0(k)\cdot z^{-2}\right]} \tag{39}$$

เมื่อได้ Transfer Function ทั้ง 3 แล้ว ก็จะใช้นำไปเขียนวงจรได้เนื่องจากฟังก์ชั่นแต่ละตัวขึ้นอยู่กับฟังก์ชั่นใดฟังก์ชั่นหนึ่งเท่านั้น จึงสามารถเขียนวงจรได้

จากสมการ

$$y(k)=\frac{(1-\alpha_0)}{2}[x(k)-x(k-2)]+\alpha_1(k)(1+\alpha_0)y(k-1)-\alpha_0 y(k-2) \tag{6}$$

$$\alpha_1(k+1)=\alpha_1(k)+\mu_1 y(k)\cdot\psi(k) \tag{8}$$

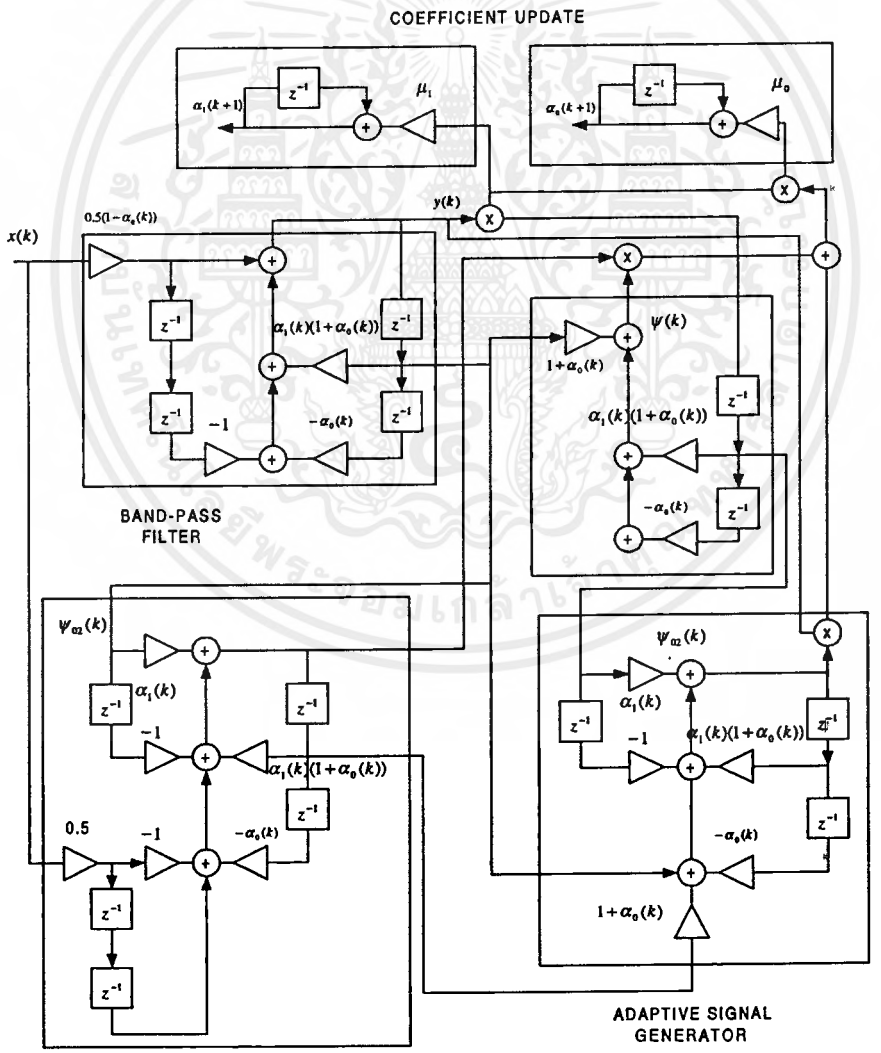
$$\psi(k)=\alpha_1(k)\cdot(1+\alpha_0)\cdot\psi(k-1)-\alpha_0\psi(k-2)+(1+\alpha_0)\cdot y(k-1) \tag{10}$$

$$\alpha_0(k+1)=\alpha_0(k)+\mu_0 y(k)\cdot\psi(k)\cdot[\psi(k)\cdot\psi_{01}(k)+\psi_{02}(k)\cdot y(k)] \tag{19}$$

$$\psi_{01}(k)=\frac{1}{2}[x(k-2)-x(k)]+\alpha_1(k)\cdot y(k-1)-y(k-2)+ \alpha_1(k)[1+\alpha_0(k)]\psi_{01}(k-1)-\alpha_0(k)\psi_{01}(k-2) \tag{20}$$

$$\psi_{02}(k)=\alpha_1(k)[1+\alpha_0(k)]\psi_{02}(k-1)+\alpha_1(k)\psi(k-1)-\alpha_0(k)\psi_{02}(k-2)-\psi(k-2) + [1+\alpha_0(k)]\psi_{01}(k-1)+y(k-1) \tag{21}$$

สามารถนำมาเขียนวงจร ADAPTIVE FILTER ได้ดังรูปที่ 1



รูปที่ 1 แผนภาพบล็อกของ ADAPTIVE ALGORITHM

แทนค่า $Z = e^{j\omega}$

$$H(e^{j\omega}) = \frac{1-\alpha_0}{2} \frac{1-(e^{j\omega})^{-2}}{1-\alpha_1(k)\cdot(1+\alpha_0)\cdot(e^{j\omega})^{-1}+\alpha_0(e^{j\omega})^{-2}}$$

คูณด้วย $e^{j\omega}$ ทั้งเศษและส่วน

$$H(e^{j\omega}) = \frac{1-\alpha_0}{2} \frac{e^{j\omega}-e^{-j\omega}}{e^{j\omega}-\alpha_1(k)\cdot(1+\alpha_0)+\alpha_0 e^{-j\omega}}$$

จากกฎของ EULER

$$e^{\pm j\theta} = \cos(\theta) \pm j\sin(\theta)$$

$$H(e^{j\omega}) = \left(\frac{1-\alpha_0}{2}\right)\left(\frac{\cos(\omega)+j\cdot\sin(\omega)-\cos(\omega)+j\cdot\sin(\omega)}{\cos(\omega)+j\cdot\sin(\omega)-\alpha_1(k)\cdot(1+\alpha_0)+\alpha_0\cos(\omega)-j\cdot\alpha_0\sin(\omega)}\right)$$

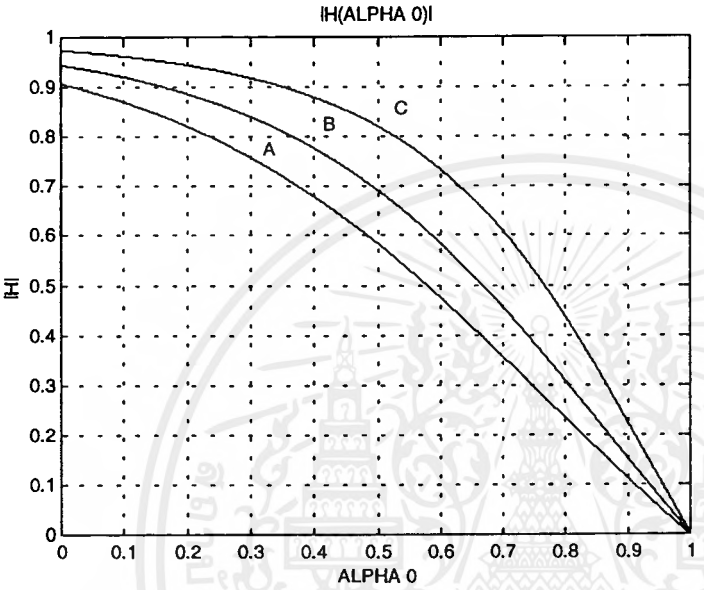
$$H(e^{j\omega}) = \left(\frac{1-\alpha_0}{2}\right)\left(\frac{2\cdot j\cdot\sin(\omega)}{\cos(\omega)(1+\alpha_0)-\alpha_1(k)\cdot(1+\alpha_0)+j\cdot\sin(\omega)(1-\alpha_0)}\right) \quad (42)$$

$$H(e^{j\omega}) = \left(\frac{j\cdot\sin(\omega)(1-\alpha_0)}{(1+\alpha_0)\{\cos(\omega)-\alpha_1(k)\}+j\cdot\sin(\omega)(1-\alpha_0)}\right) \quad (43)$$

$$|H(e^{j\omega})| = \sqrt{\left(\frac{[\sin(\omega)(1-\alpha_0)]^2}{[(1+\alpha_0)\{\cos(\omega)-\alpha_1(k)\}]^2+[\sin(\omega)(1-\alpha_0)]^2}\right)} \quad (44)$$

$$\therefore \quad H(\alpha_0) = \left(\frac{\sin(\omega)(1-\alpha_0)}{[(1+\alpha_0)^2\{\cos(\omega)-\alpha_1(k)\}^2+(1-\alpha_0)^2\sin^2(\omega)]^{\frac{1}{2}}}\right) \quad (45)$$

คุณลักษณะของ $|H(\alpha_0)|$ เทียบกับ α_0 แสดงดังรูปที่ 3 จากรูปจะเห็นได้ว่า $|H(\alpha_0)|$ ไม่มียอดเลย ดังนั้นค่าสุดของค่าเฉลี่ยของ $y(k)\psi(k)$ สามารถประมาณให้เท่ากับค่าเฉลี่ยของ $\psi(k)$ ได้ ดังนั้นค่าเฉลี่ยของ $\psi(k)$ เป็นสัดส่วนโดยตรงกับ $\frac{\partial|H(\alpha_1(k))|^2}{\partial\alpha_1(k)}$ ซึ่งจะเป็นไปตามสมการดังต่อไปนี้



$A:\alpha_1(k)=0.1,\omega=\pi/3$
$B:\alpha_1(k)=0.2.\omega=\pi/3$
$C:\alpha_1(k)=0.3,\omega=\pi/3$

รูปที่ 2 คุณลักษณะของ $|H(\alpha_0)|$

จาก

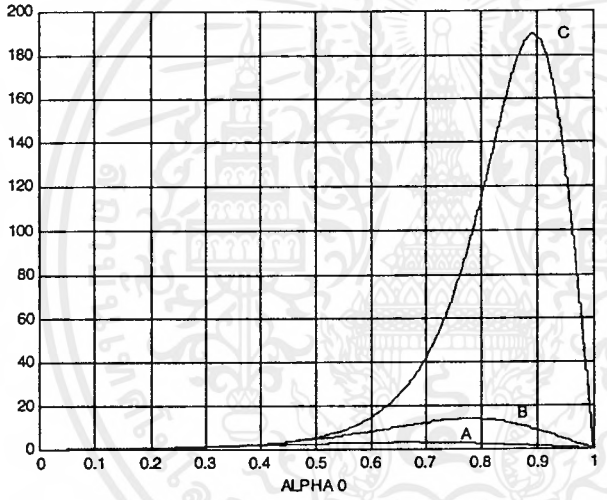
$$\left|\frac{\partial|H(\alpha_1(k))|}{\partial\alpha_1(k)}\right|^2 = \frac{\left\{\frac{1+\alpha_0}{\sin(\omega)(1-\alpha_0)}\right\}^2\{\cos(\omega)-\alpha_1(k)\}^2}{\left[\left\{\frac{(\cos(\omega)-\alpha_1(k))(1+\alpha_0)}{\sin(\omega)(1-\alpha_0)}\right\}^2+1\right]^3}$$

$$= \frac{\left[(1+\alpha_0)^2(1-\alpha_0)\sin(\omega)\{\cos(\omega-\alpha_1(k))\}\right]^2}{\left[(1+\alpha_0)^2\{\cos(\omega)-\alpha_1(k)\}^2+(1-\alpha_0)^2\sin^2(\omega)\right]^3} \qquad (46)$$

$$\therefore \quad \frac{\partial|\mathrm{H}(\alpha_1(k))|}{\partial\alpha_1(k)} = \frac{(1+\alpha_0)^2(1-\alpha_0)\sin(\omega)\{\cos(\omega)-\alpha_1(k)\}}{\left[(1+\alpha_0)^2\{\cos(\omega)-\alpha_1(k)\}^2+(1-\alpha_0)^2\sin^2(\omega)\right]^{3/2}} \tag{47}$$

จากสมการที่ 47 ซึ่งเป็นฟังก์ชั่นของ α_0 สามารถเขียนใหม่ให้อยู่ในรูปที่ง่ายขึ้นได้ดังสมการ

$$\Pi(\alpha_0) = \frac{\partial|\mathrm{H}(\alpha_1(k))|}{\partial\alpha_1(k)} \tag{48}$$



$A: \alpha_1(k) = 0.1, \omega = \pi/3$
$B: \alpha_1(k) = 0.2. \omega = \pi/3$
$C: \alpha_1(k) = 0.3, \omega = \pi/3$

รูปที่ 3 คุณลักษณะของ $|\Pi(\alpha_0)|$

คุณลักษณะของ $|\Pi(\alpha_0)|$ เทียบกับ α_0 แสดงดังรูปที่ 4 จากรูปจะเห็นว่า $|\Pi(\alpha_0)|$ มียอดเดียว เช่นกับวงจรกรองความถี่กลางผ่าน ดังนั้นสามารถดัดแปลงหรือปรับปรุงอัลกอริทึมสำหรับการ UPDATE ของ $\alpha_0(k)$ ขึ้นใหม่ได้ดังสมการ

$$\alpha_0(k+1) = \alpha_0(k) + \frac{\mu_0}{2}\frac{\partial\psi^2(k)}{\partial\alpha_0(k)} \tag{49}$$

$$= \alpha_0(k) + \mu_0\psi(k)\cdot\psi_{02}(k) \tag{50}$$

โดยที่ $\psi_{02}(k)=\frac{\partial\psi(k)}{\partial\alpha_0(k)}$: μ_0 เป็น Step Size Parameter

และ $\psi_{02}(k)$ เป็นสัญลักษณ์ควบคุมแบบ Adaptive ของ $\alpha_0(k)$จะเห็นว่าใช้เพียงแค่ตัวเดียวเท่านั้น

จากความสัมพันธ์ของสมการ (6) , (10) , (20) , (21) สามารถเขียน $\psi_{01}(k),\psi_{02}(k)$ ใหม่ได้ดังสมการ

จาก

$$y(k)=\frac{(1-\alpha_0)}{2}(x(k)-x(k-2))+\alpha_1(k)\cdot(1+\alpha_0)\cdot y(k-1)-\alpha_0 y(k-2)$$

$$\frac{1}{2}[x(k-2)-x(k)]=\frac{-y(k)}{1-\alpha_0(k)}+\frac{1+\alpha_0(k)}{1-\alpha_0(k)}\alpha_1(k)y(k-1)-\frac{\alpha_0(k)}{1-\alpha_0(k)}y(k-2)$$

นำไปแทนค่า $\psi_{01}(k)=\frac{\partial y(k)}{\partial\alpha_0(k)}$ จะได้

$$\psi_{01}(k)=\alpha_1(k)\cdot[1+\alpha_0(k)]\psi_{01}(k-1)-\alpha_0(k)\psi_{01}(k-2)+\alpha_1(k)y(k-1)-y(k-2)$$

$$\frac{-y(k)}{1-\alpha_0(k)}+\frac{\alpha_1(k)(1+\alpha_0(k))y(k-1)}{1-\alpha_0(k)}-\frac{\alpha_0(k)y(k-2)}{1-\alpha_0(k)}$$

$$=\alpha_1(k)(1+\alpha_0(k))\psi_{01}(k-1)-\alpha_0(k)\psi_{01}(k-2)+\frac{1}{1-\alpha_0(k)}$$

$$\left[\begin{array}{l}-y(k)+\alpha_1(k)(1+\alpha_0(k))y(k-1)-\alpha_0(k)y(k-2)+(1-\alpha_0(k))\alpha_1(k)y(k-1)\\ -(1-\alpha_0(k))y(k-2)\end{array}\right]$$

จะได้

$$\psi_{01}(k)=\alpha_1(k)(1\alpha_0(k))\psi_{01}(k-1)-\alpha_0(k)\psi_{01}(k-2)+\frac{1}{1-\alpha_0(k)}$$

$$[-y(k)+2\cdot\alpha_1(k)y(k-1)-y(k-2)] \tag{51}$$

จาก

$$\psi(k)=\alpha_1(k)\cdot(1+\alpha_0)\cdot\psi(k-1)-\alpha_0\psi(k-2)+(1+\alpha_0)\cdot y(k-1)$$

$$y(k-1)=\frac{\psi(k)}{1+\alpha_0(k)}-\alpha_1(k)\cdot\psi(k-1)+\frac{\alpha_0(k)}{1+\alpha_0(k)}\psi|k-2| \tag{52}$$

นำไปแทนใน $\psi_{02}(k)$

$$\psi_{02}(k)=\alpha_1(k)\cdot[1+\alpha_0(k)]\psi_{02}(k-1)-\alpha_0(k)\cdot\psi_{02}(k-2)-\psi(k-2)+$$

$$(1+\alpha_0(k))\psi_{01}(k-1)+\frac{\psi(k)}{1+\alpha_0(k)}+\frac{\alpha_0(k)}{1+\alpha_0(k)}\psi(k-2)$$

จาก(52) Take Z-Transform ได้

$$y(z)\cdot z^{-1}=\frac{\psi(z)}{1+\alpha_0(k)}-\alpha_1(k)\psi(z)z^{-1}+\frac{\alpha_0(k)}{1+\alpha_0(k)}\psi(z)z^{-2} \tag{53}$$

และจาก (34)

$$\psi_{01}(z)=\frac{2\cdot\alpha_1(k)\cdot z^{-1}-z^{-2}-1}{(1-\alpha_0(k))[1-\alpha_1(k)\cdot(1+\alpha_0(k))z^{-1}+\alpha_0(k)z^{-2}]}\times y(z)$$

แทนค่าจะได้

$$\psi_{01}(z)\cdot z^{-1}=\frac{2\cdot\alpha_1(k)\cdot z^{-1}-z^{-2}-1}{(1-\alpha_0{}^2(k))}\times\psi(z) \tag{54}$$

กำหนดให้ $h(\alpha_0)=\frac{1+\alpha_0}{1-\alpha_1}$, $A=\frac{1}{\sin(\omega)}$, $B=\{\cos(\omega)-\alpha_1(k)\}$

จะได้

$$|\Pi(\alpha_0)|=\frac{A^2h^2(\alpha_0)B^2}{\{A^2B^2h^2(\alpha_0)+1\}^3}$$

ดังนั้น

$$\frac{\partial}{\partial\alpha_0(k)}=\left[\frac{A^2h^2(\alpha_0)B^2}{\{A^2B^2h^2(\alpha_0)+1\}^3}\right]=0$$

$$\frac{\left[\begin{array}{l}\left(A^2B^2h^2(\alpha_0)+1\right)^3\left(A^2B^2 2\cdot h(\alpha_0)h'(\alpha_0)\right)-\left(A^2B^2h^2(\alpha_0)\right)\\ \left\{3\cdot\left(A^2B^2h^2(\alpha_0)+1\right)^2\left(A^2B^2 2\cdot h(\alpha_0)h'(\alpha_0)\right)\right\}\end{array}\right]}{\left(A^2B^2h^2(\alpha_0)+1\right)^6}=0$$

$$A^2B^2\cdot\frac{\left[\begin{array}{l}2\cdot\left(A^2B^2h^2(\alpha_0)+1\right)^3\left(h(\alpha_0)h'(\alpha_0)\right)\\ -6\cdot\left(A^2B^2h^2(\alpha_0)+1\right)^2h^3(\alpha_0)A^2B^2h'(\alpha_0)\end{array}\right]}{\left(A^2B^2h^2(\alpha_0)+1\right)^6}=0$$

$$A^2B^2\cdot\frac{\left[2\cdot h(\alpha_0)h'(\alpha_0)\{A^2B^2h^2(\alpha_0)+1-3\cdot A^2B^2h^2(\alpha_0)\}\right]}{\left(A^2B^2h^2(\alpha_0)+1\right)^4}=0$$

$$\frac{2\cdot A^2B^2\cdot h(\alpha_0)h'(\alpha_0)\left(1-A^2B^2h^2(\alpha_0)\right)}{\left(A^2B^2h^2(\alpha_0)+1\right)^4}$$

จะมีอยู่ 2 กรณี คือ $2A^2h(\alpha_0)h'(\alpha_0)=0$ และ $1-A^2B^2h^2(\alpha_0)=0$

จาก

$$2 \cdot A^2 B^2 \cdot h(\alpha_0) h'(\alpha_0) = 0$$

$$h(\alpha_0) h'(\alpha_0) = 0$$

$$h'(\alpha_0) = \frac{\partial h(\alpha_0)}{\partial \alpha_0} = \frac{\partial (1+\alpha_0)}{\partial \alpha_0 (1-\alpha_0)}$$

$$\therefore h'(\alpha_0) = \frac{2}{(1-\alpha_0)^2}$$

ได้

$$\left[\frac{1+\alpha_0}{1-\alpha_0}\right]\frac{2}{(1-\alpha_0)^2} = 0$$

$$\left[\frac{(1-\alpha_0{}^2)^2}{(1-\alpha_0)^2}\right] = 0$$

$$1-\alpha_0^2 = 0$$

$$\therefore \quad \alpha_0 = \pm 1$$

และจาก

$$1 - A^2 B^2 h^2(\alpha_0) = 0$$

$$A^2 B^2 h(\alpha_0) = 1$$

$$A^2 B^2 \left[\frac{1+\alpha_0}{1-\alpha_0}\right]^2 = 1$$

$$A^2 B^2 (1+\alpha_0)^2 = (1-\alpha_0)^2$$

$$A \cdot B \cdot (1+\alpha_0) = \pm(1-\alpha_0)$$

กรณีที่ 1

$$\psi(k)=\alpha_1(k)\cdot(1+\alpha_0)\cdot\psi(k-1)-\alpha_0\psi(k-2)+(1+\alpha_0)\cdot y(k-1)$$

TAKE Z TRANSFORM

$$y(z)z^{-1}=\frac{\psi(z)}{1+\alpha_0(k)}-\alpha_1(k)\cdot\psi(z)z^{-1}+\frac{\alpha_0(k)}{1+\alpha_0(k)}\psi(z)z^{-2}$$

$$\psi(z)\left[1-\alpha_1(k)(1+\alpha_0(k))z^{-1}-\alpha_0(k)z^{-2}\right]=(1+\alpha_0(k))z^{-1}y(z) \tag{61}$$

$$\therefore \quad G(z)=\frac{\psi(z)}{y(z)}=\frac{(1+\alpha_0(k))z^{-1}}{1-\alpha_1(k)\cdot(1+\alpha_0(k))\cdot z^{-1}+\alpha_0(k)z^{-2}} \tag{62}$$

และจาก

$$\psi_{02}(k)=\alpha_1(k)\cdot[1+\alpha_0(k)]\psi_{02}(k-1)-\alpha_0(k)\cdot\psi_{02}(k-2)+\frac{2}{1-\alpha_0{}^2(k)}$$

$$\left[-\alpha_0(k)\psi(k)-\psi(k-2)+\alpha_1(k)(1+\alpha_0(k))\psi(k-1)\right]$$

TAKE Z-TRANZFORM

$$\psi_{02}(z)=\alpha_1(k)\cdot[1+\alpha_0(k)]\psi_{02}(z)z^{-1}-\alpha_0(k)\cdot\psi_{02}(z)z^{-2}+\frac{2}{1-\alpha_0{}^2(k)} \tag{63}$$

$$\left[-\alpha_0(k)\psi(z)-\psi(z)z^{-2}+\alpha_1(k)(1+\alpha_0(k))\psi(z)z^{-1}\right]$$

$$\psi_{02}(z)=\left[1-\alpha_1(k)\cdot[1+\alpha_0(k)]z^{-1}\alpha_0(k)\cdot z^{-2}\right]=\frac{2}{1-\alpha_0{}^2(k)}$$

$$\left[-\alpha_0(k)-z^{-2}+\alpha_1(k)(1+\alpha_0(k))z^{-1}\right]\psi(z)$$

$$\therefore \qquad G_{02}(z)=\frac{\psi_{02}(z)}{\psi(z)}=\frac{2}{1-\alpha_0{}^2(k)}\cdot\frac{\left[-\alpha_0(k)+\alpha_1(k)(1+\alpha_0(k))z^{-1}-z^{-2}\right]}{\left[1-\alpha_1(k)(1+\alpha_0(k))z^{-1}+\alpha_0(k)z^{-2}\right]} \qquad (64)$$

ดังนั้นจากสมการ

$$y(k)=\left(\frac{1-\alpha_0}{2}\right)[x(k)-x(k-2)]+\alpha_1(k)(1+\alpha_0)y(k-1)-\alpha_0 y(k-2) \qquad (6)$$

$$\alpha_1(k+1)=\alpha_1(k)+\mu_1 y(k)\cdot\psi(k) \qquad (8)$$

$$\psi(k)=\alpha_1(k)\cdot(1+\alpha_0)\cdot\psi(k-1)-\alpha_0\psi(k-2)+(1+\alpha_0)\cdot y(k-1) \qquad (10)$$

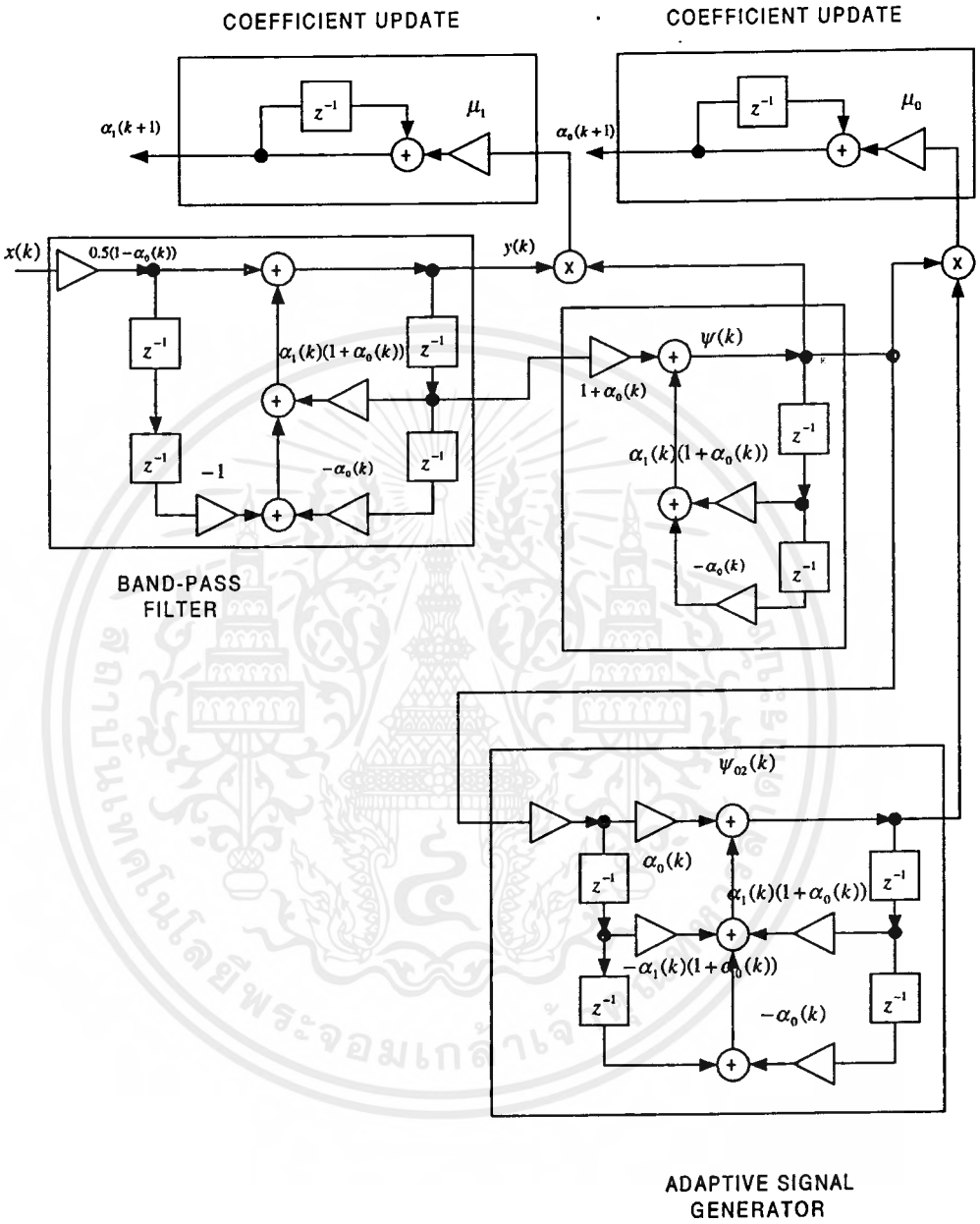
$$\alpha_0(k+1)=\alpha_0(k)+\mu_0\psi(k)\cdot\psi_{02}(k) \qquad (50)$$

$$\psi_{02}(k)=\alpha_1(k)\cdot[1+\alpha_0(k)]\psi_{02}(k-1)-\alpha_0(k)\cdot\psi_{02}(k-2)+\frac{2}{1-\alpha_0{}^2(k)} \qquad (57)$$

$$[-\alpha_0(k)\psi(k)-\psi(k-2)+\alpha_1(k)(1+\alpha_0(k))\psi(k-1)]$$

จากสมการดังกล่าวสามารถนำมาออกแบบวงจรเป็น วงจร Adaptive Algorithm ที่ใช้สำหรับการดีเทกสัญญาณซายน์คลื่นเดียวโดยใช้ Simplified Adaptive Algorithm และสามารถปรับค่า Q-Factor ได้ดังรูปที่ 4

จากวงจรจะเห็นว่าใช้สัญญาณควบคุมแบบ Adaptive เพียงสองตัวเท่านั้นคือ $\psi(k)$, $\psi_{02}(k)$ และมีอินพุทเดียว ซึ่งจะทำให้สามารถนำไปใช้งานได้ง่ายขึ้น



รูปที่ 4 แผนภาพบล็อก SIMPLIFILED ADAPTIVE ALGORITHM

4. ADAPTIVE ALGOLITHM โดยวิธีการนอร์มอลไลซ์เกรเดียนซ์

จากอะแดปทีปอัลกอริทึมโดยการใช้ฟิลเตอร์แบบค่า Q-Factor คงที่และอะแดปทีปอัลกอริทึมโดยการ Adaptive Algorithm โดยใช้วิธีการนอร์มอลไลซ์เกรเดียนต์ใช้ฟิลเตอร์ที่มีค่า Q-Factor คงที่ และจะเห็นได้ว่าเมื่อกำหนดให้ค่า Q-Factor ของวงจรมีค่าสูงและความถี่ อินพุตห่างจากความถี่กลางทำให้การอัปเดตค่าสัมประสิทธิ์ $\alpha_1(k)$ ทำได้ช้าและในทางกลับกันเมื่อความถี่อินพุตใกล้กับความถี่กลางจะทำให้สัมประสิทธิ์ที่อัปเดตมีค่าความแปรปรวนสูง ซึ่งสามารถแก้ปัญหานี้ได้โดยการ นอร์มอลไลซ์เกรเดียนต์ ซึ่งจะได้สมการ ของวิธีการนอร์มอลไลซ์เกรเดียนซ์ ซึ่งสามารถนำไปใช้ดังต่อไปนี้

สมการ

$$y(k)=\left(\frac{1-\alpha_0}{2}\right)[x(k)-x(k-2)]+\alpha_1(k)(1+\alpha_0)y(k-1)-\alpha_0 y(k-2) \tag{65}$$

$$\alpha_1(k+1)=\alpha_1(k)+\frac{\mu}{\beta(k)}y(k)\cdot\psi(k) \tag{66}$$

$$\psi(k)=\alpha_1(k)\cdot(1+\alpha_0)\cdot\psi(k-1)-\alpha_0\psi(k-2)+(1+\alpha_0)\cdot y(k-1) \tag{67}$$

$$\beta(k)=\gamma.\beta(k-1)+(1-\gamma)[y(k).\psi(k)]^2 \tag{68}$$

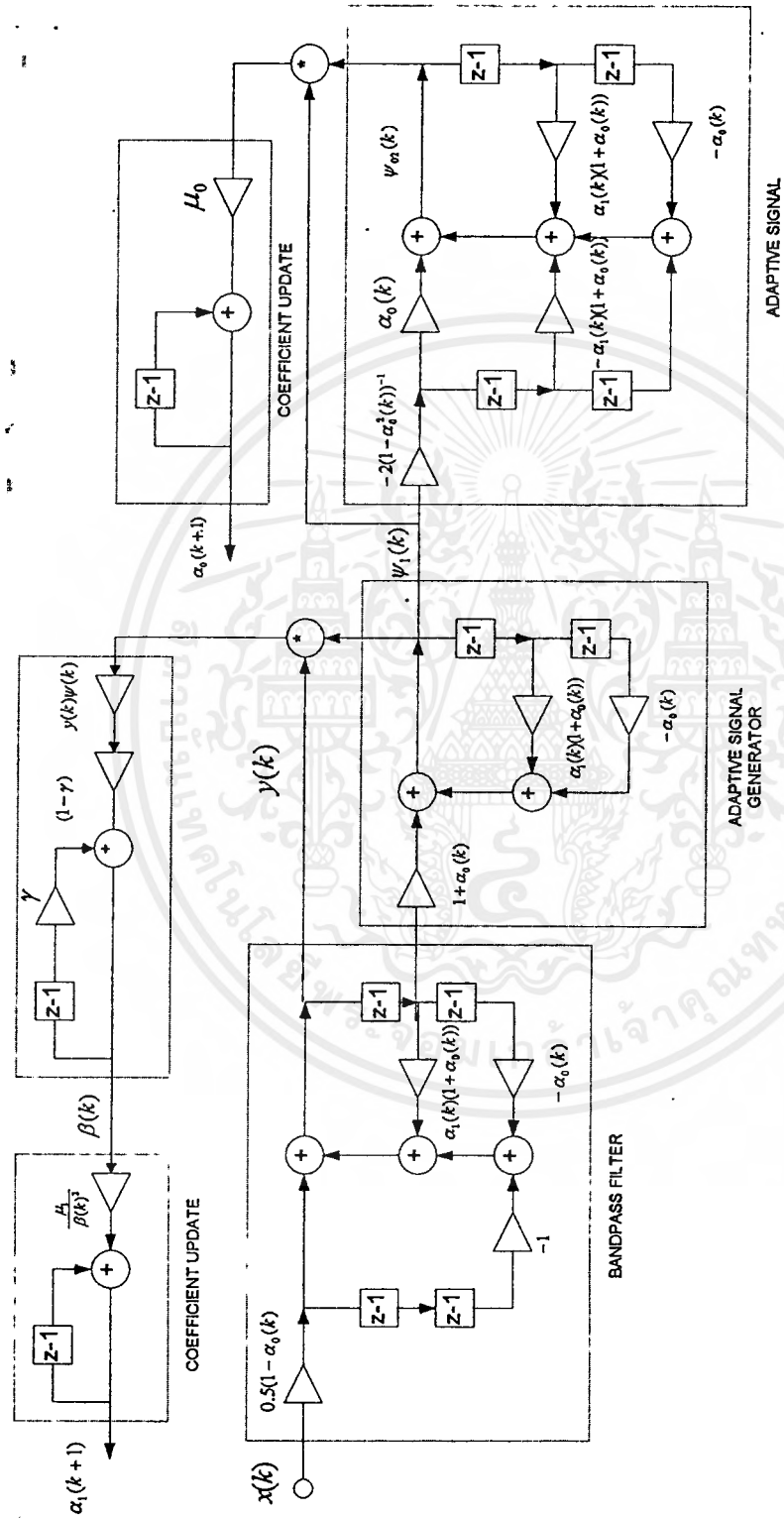
$$\alpha_0(k+1)=\gamma.\beta(k)+\mu_0\psi(k).\psi_{02}(k) \tag{69}$$

เมื่อ $\psi(k)$: เป็นสัญญาณควบคุมอะเดปทีปของ $\alpha_1(k)$

γ : เป็นค่า Forgetting Factor โดยมีค่า $0 << \gamma << 1$

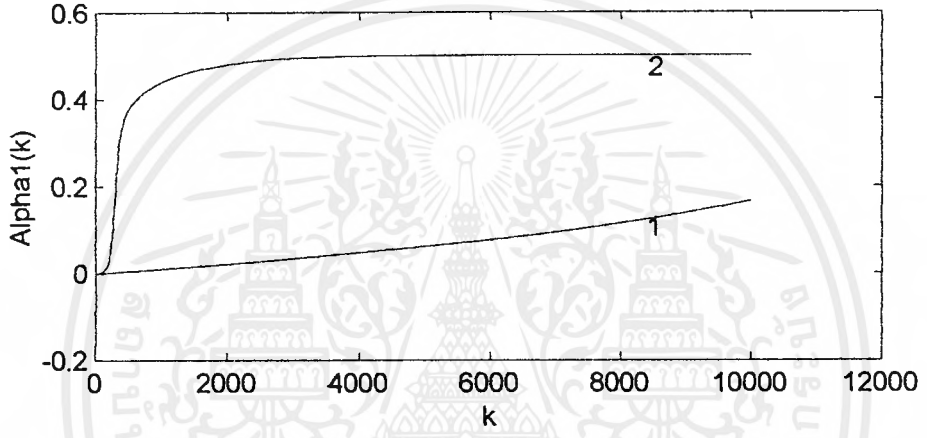
μ : Step Size Parameter

$\beta(k)$: กำลังงานของอัปเดตฟังก์ชัน



รูปที่ 5 แผนภาพบล็อก NORMALIZED GRADIENT ALGORITHM

จากรูปเป็นการทดลองโดยทำการเปลี่ยนแปลงค่า α_0 ที่เป็นพารามิเตอร์ที่ใช้กำหนดค่า Q – Factor ของวงจร จะเห็นได้ว่า แม้ว่าวงจรจะมีค่า Q – Factor ต่ำ แต่ก็ยังมีสามารถอัปเดตค่าสัมประสิทธิ์ $\alpha_1(k)$ ได้รวดเร็วและวงจรจะทำงานได้เร็วขึ้น เมื่อค่า Q – Factor สูงขึ้น โดยพารามิเตอร์ที่ใช้ทดลอง คือ A = 1 , SNR = 10 dB , $\omega = \pi/3$, $\gamma = 0.98$ และ $\mu = 0.0005$ และทำการเปลี่ยนแปลง α_0 ให้มีค่าเท่ากับ 0.2 , 0.5 , 0.7 ตามลำดับ



รูปที่ 7 เปรียบเทียบความเร็วในการอัปเดตค่าสัมประสิทธิ์ $\alpha_1(k)$
เมื่อกำหนดให้ $\alpha_0 = 0.8$
1 : SIMPLIFIED ADAPTIVE ALGORITHM , 2 : NORMALIZED GRADIENT ALGROITHM

สรุปผลการทดลองจากกราฟ

กราฟที่ 1 เป็นการแสดงคุณสมบัติของสมการทรานเฟอร์ฟังก์ชันแบนพาสด์เทียบกับมุม ω พบว่าค่าของ α_0 มีผลทำให้ค่า Q – Factor หรือความกว้างของการตอบสนองของทรานเฟอร์ฟังก์ชันเปลี่ยนแปลงไปโดยค่า Q – Factor จะแปรผันตรงกับค่าของ α_0 และค่าของ α_1 เป็นค่าที่กำหนดจุดการตอบสนองสูงสุดของทรานเฟอร์ฟังก์ชัน โดยจุดตอบสนองสูงสุดจะเบี่ยงเบนไปทางด้านซ้ายถ้า α_1 มีค่ามากกว่าศูนย์

กราฟที่ 2 เป็นการแสดงคุณสมบัติของทรานเฟอร์ฟังก์ชันเทียบกับ α_1 พบว่าจุดตอบสนองสูงสุดของทรานเฟอร์ฟังก์ชันจะอยู่ที่ $\alpha_1 = \cos(\omega)$ เสมอ และค่า α_0 ก็ยังคงทำหน้าที่กำหนด Q - Factor

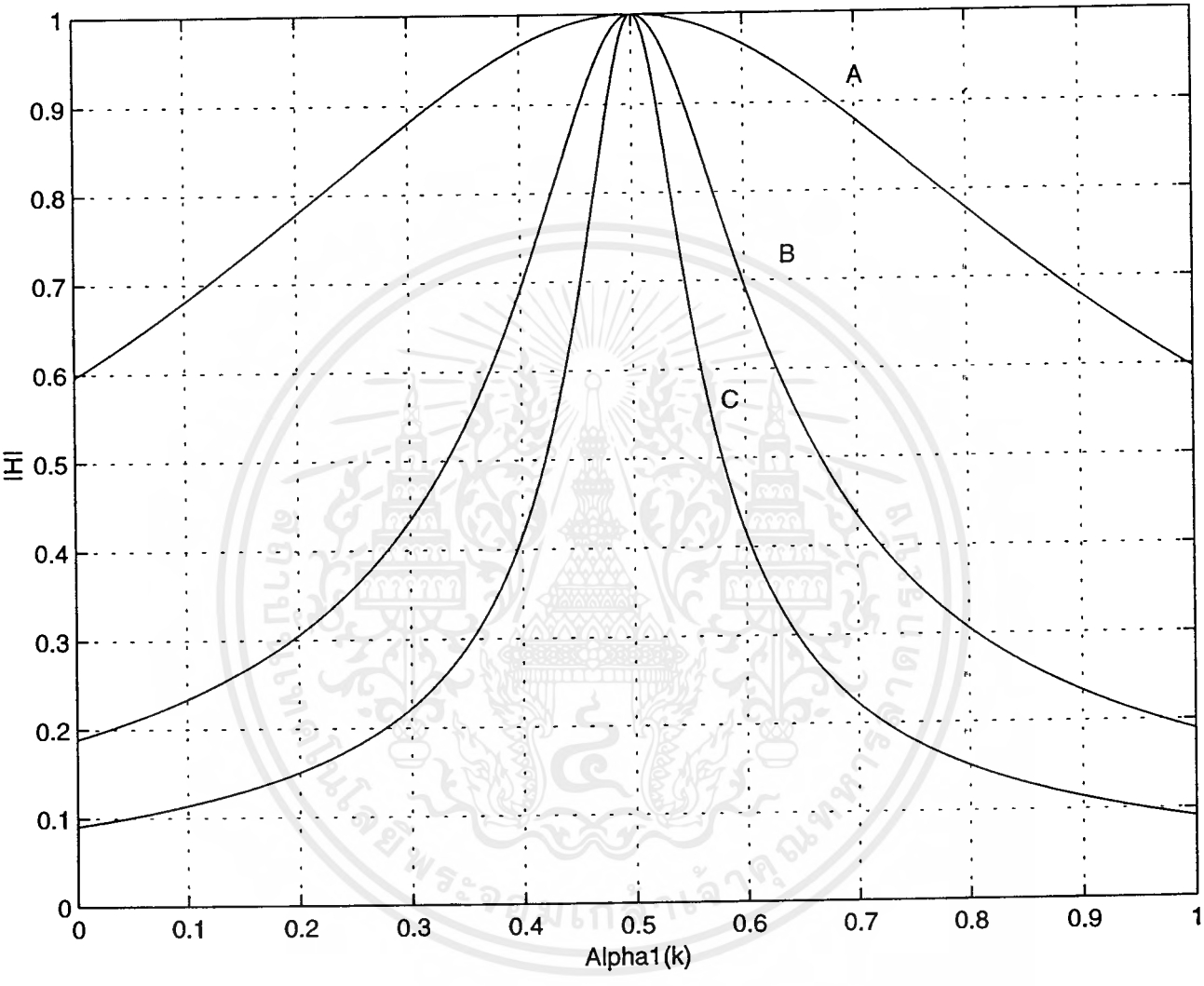
กราฟที่ 3 เป็นกราฟแสดงคุณสมบัติของ $\alpha_1(k)$ เทียบกับแกนเวลา k ของทั้งสองแบบคือ ซิมพลิไพย์อะแดปทีฟอัลกอริทึมแบบใหม่ กับ อะแดปทีฟอัลกอริทึมแบบเดิม เมื่อ α_0 คงที่พบว่าค่า μ จะมีผลทำให้การลู่เข้าของกราฟเร็วขึ้น โดยที่ถ้าค่า μ ก็จะลู่เข้าเร็ว ถ้าค่าน้อย (เข้าใกล้ศูนย์) ก็จะลู่เข้าช้า ค่า μ เป็นความละเอียดของการลู่เข้าถ้ามีค่ามากจะทำให้ลู่เร็วและถ้ามากเกินไปจะทำให้ลู่เข้าอย่างขาดเสถียรภาพ ค่า ω เป็นมุมฟังก์ชัน $\alpha_1 = \cos(\omega)$ และค่าของ α_0 จะทำให้การลู่เข้าช้าถ้าค่าของ α_0 น้อย (เข้าใกล้ศูนย์) และจะเร็วถ้ามีค่ามาก(เข้าใกล้หนึ่ง)

กราฟที่ 4 เป็นกราฟแสดงการเปรียบเทียบการ Converge ของ $\alpha_1(k)$ เมื่อเทียบกับเวลา k พบว่าสัญญาณของซิมพลิไฟย์อะแดปทีฟอัลกอริทึมแบบใหม่ที่ใช้วงจรกรองความถี่ผ่านแบบ IIR อันดับ 2 และสามารถปรับค่า Q – Factor ได้นั้น สามารถ Converge สัญญาณได้เร็วกว่า อะแดปทีฟอัลกอริทึมแบบเดิมที่ใช้วงจรกรองความถี่ผ่านแบบ IIR ที่สามารถปรับค่า Q – Factor ได้เช่นเดียวกัน โดยกำหนดสัมประสิทธิ์ของตัวแปรต่างๆ เหมือนกันทุกประการ

กราฟที่ 5 เป็นกราฟแสดงการเปรียบเทียบการ Converge ของ $\alpha_1(k)$ เมื่อเทียบกับแกนเวลา k ของผลการทดลองทั้ง 3 แบบคือ ซิมพลิไฟย์อะแดปทีฟอัลกอริทึมแบบใหม่ ที่ใช้วงจรกรองความถี่ผ่านแบบ IIR อันดับ 2 และสามารถปรับค่า Q – Factor กับ อะแดปทีฟอับกอริทึมแบบเดิมที่ใช้วงจรกรองความถี่ผ่านแบบ IIR ที่สามารถปรับค่า Q – Factor และ อะแดปทีฟอับกอริทึมโดยใช้วิธีนอร์มอลไลซ์เกรเดียนต์ ที่ใช้วงจรกรองความถี่ผ่านแบบ IIR ที่มีค่า Q – Factor คงที่ได้ พบว่าวิธีนี้สามารถติดตามสัญญาณอินพุตตามความถี่ใช้งานได้เร็วกว่าทั้งสองแบบโดยได้กำหนดให้สัมประสิทธิ์ของตัวแปรต่างๆ เหมือนกันทุกประการ

กราฟแสดงคุณสมบัติของ $|H(\omega)|$ เทียบกับ ω

IIR Band - Pass Filter



กราฟที่ 2

กำหนด

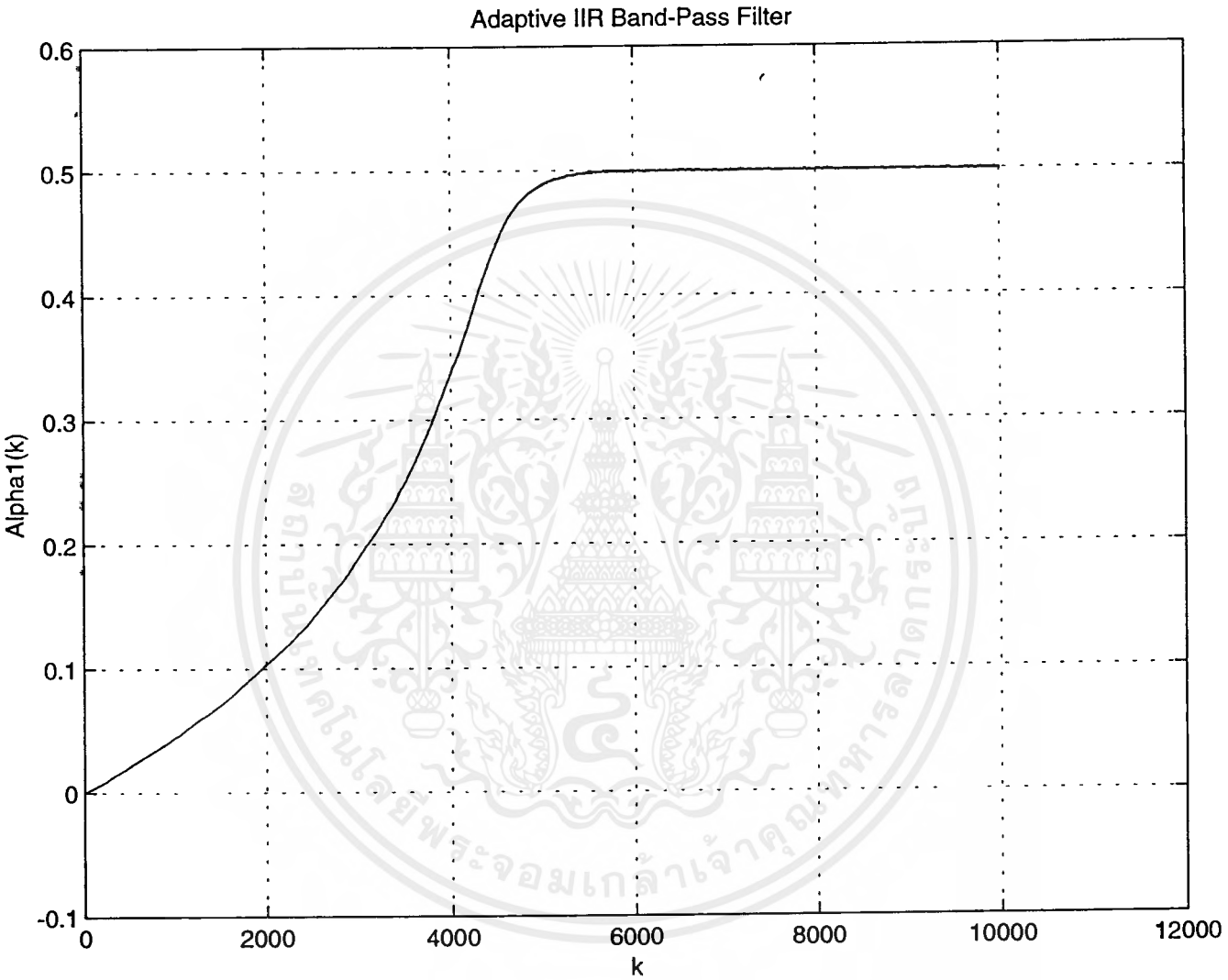
(A) $\omega = \pi/3$, $\alpha_0 = 0.4$ **(B)** $\omega = \pi/3$, $\alpha_0 = 0.8$

(D) $\omega = \pi/3$, $\alpha_0 = 0.9$

จากสมการ Transfer Function $H(z) = \frac{1-\alpha_0}{2} \frac{1-z^{-2}}{1-\alpha_1(k).(1+\alpha_0).z^{-1}+\alpha_0 z^{-2}}$

กราฟแสดงการ Converge ของ $\alpha_1(k)$ เทียบกับ แกนเวลา k ของ

Adaptive IIR Band - Pass Filter



กราฟที่ 3

กำหนด A = 1 , SNR = 10 dB , $\omega = \pi/3$, $\mu_0 = 0.0006$, $\mu_1 = 0.0003$

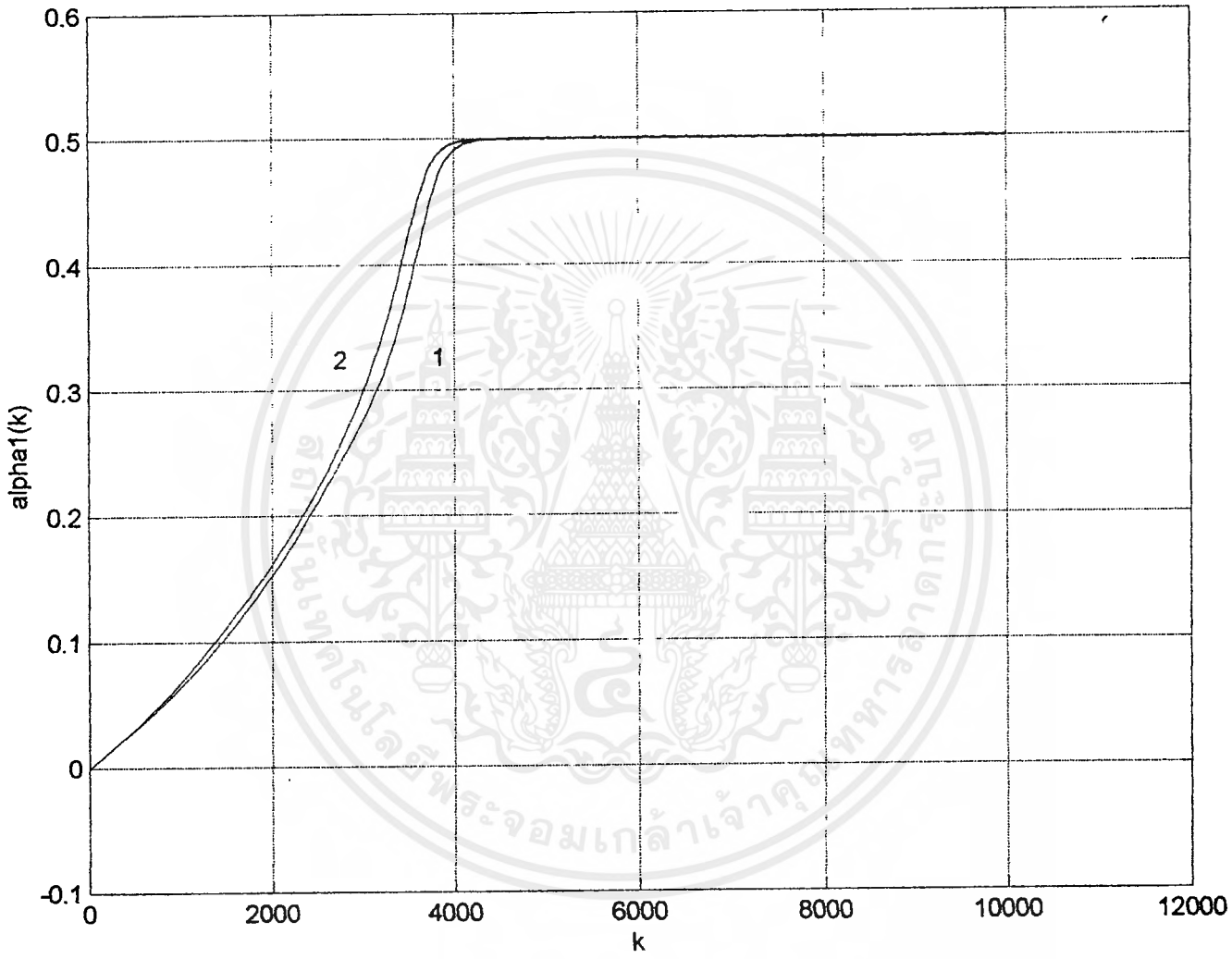
$\alpha_{0\,min} =$

$\alpha_{0\,max} =$

กราฟแสดงการเปรียบเทียบการ Converge ของ $\alpha_1(k)$ เทียบกับ แกนเวลา k ของ

Adaptive IIR Band - Pass Filter และ

Simplified Adaptive IIR Band - Pass Filter



กราฟที่ 4

กำหนด A = 1 , SNR = 10 dB , $\omega = \pi/3$, $\mu_0 = 0.0006$, $\mu_1 = 0.0003$

1 : ADAPTIVE FILTER , 2 : SIMPLIFILED ADAPTIVE FILTER

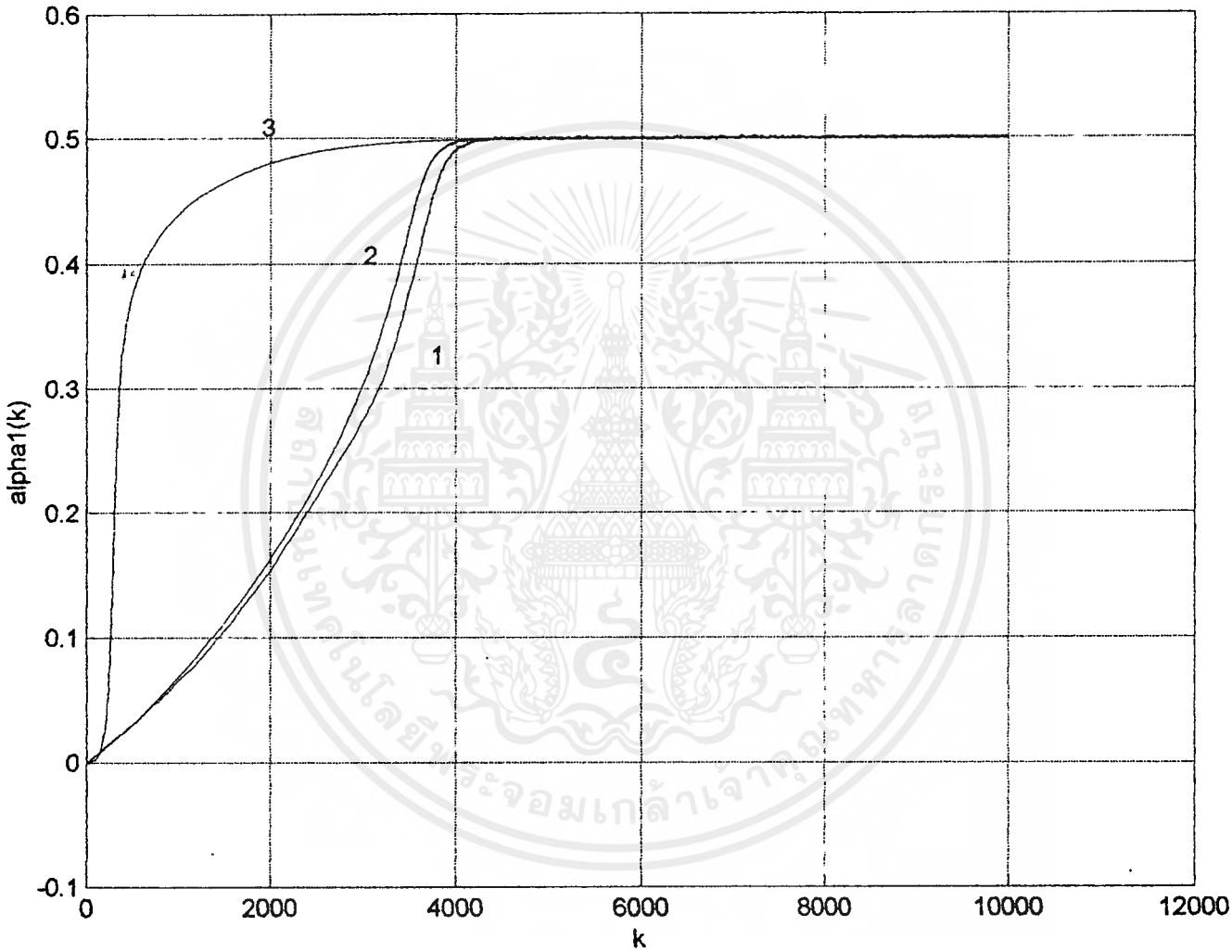
$\alpha_{0\,min} = 0$, $\alpha_{0\,max} = 0.7$

กราฟแสดงการเปรียบเทียบการ Converge ของ $\alpha_1(k)$ เทียบกับ แกนเวลา k ของ

Adaptive IIR Band - Pass Filter

Simplified Adaptive IIR Band - Pass Filter และ

Normalized Gradient Adaptive IIR Band - Pass Filter



กราฟที่ 5

กำหนด A = 1 , SNR = 10 dB , $\omega = \pi/3$, $\mu = 0.0003$, $\gamma = 0.98$, Alpha0 = 0.7

1 : ADAPTIVE FILTER

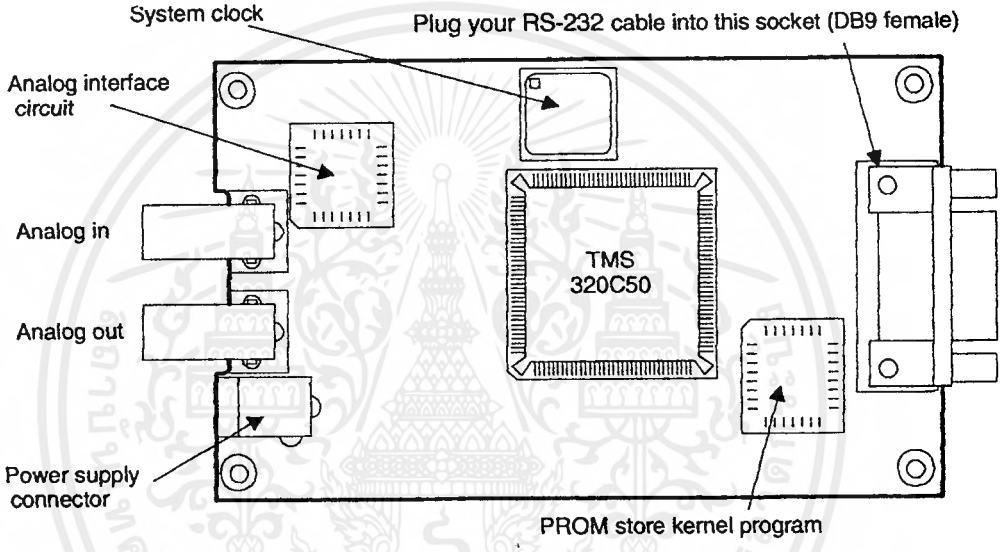
2 : SIMPLIFILED ADAPTIVE FILTER

3 : NORMALIZED GRADIENT

2. ระบบที่ต้องการ

ระบบที่เครื่องคอมพิวเตอร์ที่จะใช้กับชุด DSK นี้จะต้องมีคุณสมบัติดังนี้

- เป็นเครื่อง IBM PC/AT หรือเทียบเท่า
- หน่วยความจำ RAM อย่างน้อย 640 กิโลไบต์
- มีฟล็อปปี้ดิส และ ฮาร์ดดิสก์



รูปที่ 2 อุปกรณ์โดยทั่วไปของ TMS320C50 DSK BOARD

3. วงจรสมบูรณ์

บอร์ด DSK มีวงจรที่สมบูรณ์ในรูปที่ 3 , 4 , 5 และ 6 วงจรของการเชื่อมต่อระหว่างชิพ DSK กับอุปกรณ์ต่อร่วมอื่นๆ ในส่วนล่างเป็นภาคจ่ายไฟของบอร์ด DSK หัวใจสำคัญของบอร์ดอยู่ที่ U4 ชิพ DSP เบอร์ TMS320C50 เป็นไมโครโปรเซสเซอร์ตัวหนึ่งที่ใช้งานด้านการประมวลผลสัญญาณดิจิตอล โดยเฉพาะ ทำหน้าที่ประมวลผลข้อมูลทั้งหมดของบอร์ด DSK นี้ โดยต้องทำงานร่วมกับซอฟต์แวร์

U4 ทำงานได้โดยอาศัยสัญญาณนาฬิกาจากโมดูลกำเนิดสัญญาณนาฬิกา 40 เมกกะเฮิรตซ์ U5 SQ3300โดยมี X1 เซรามิกเรโซเนเตอร์ช่วยควบคุมความถี่ของสัญญาณนาฬิกาให้มีความเที่ยงตรงอีกชั้นหนึ่ง

U3 ชิพ ADC/DAC เบอร์ TLC32040CFN ของเท็กซัสอินสตรูเมนท์ ทำหน้าที่รับสัญญาณอนาลอกทางด้านอินพุทที่จ่ายเข้ามาทาง J1 แล้วแปลงเป็นข้อมูลทางดิจิตอลส่งไปประมวลผลต่อที่ U4 และเมื่อต้องการส่งสัญญาณออก U4 จะส่งข้อมูลดิจิตอลมายัง U3 เพื่อทำการแปลงเป็นสัญญาณอนาลอกจ่ายออกไปทาง J2 ความละเอียดของการแปลงสัญญาณดิจิตอลจะมีขนาด 14 บิต สามารถกำหนดอัตราแซมปลิ้งได้สูงสุด 19,200 แซมปลิ้งใน 1 วินาที

สำหรับการติดต่อกับคอมพิวเตอร์ผ่านทางพอร์ตอนุกรม RS-232C นั้นจะเป็นหน้าที่ของ U_{6A} เบอร์ 27PC256FM และ U_{7A} , U_{7C} , U_{7D} เบอร์ MC1489D โดย U6 ทำหน้าที่ส่งข้อมูลจากบอร์ด DSK ออกไปทาง P1 คอนเน็กเตอร์ชนิด D 9ขา ที่ต่อเข้ากับพอร์ตอนุกรม RS-232C ส่วน U7 ทำหน้าที่เป็นบัฟเฟอร์กำหนดข้อมูลจากคอมพิวเตอร์

ส่วนภาคจ่ายไฟบอร์ด DSK ต้องการไฟสลับ 9 โวลท์ จากภายนอกจ่ายเข้ามาทาง J3 และมีไดโอด D_1 , D_7 กับ คาปาซิเตอร์ C_1 , C_2 = 25 โวลท์ 1000 ไมโครฟารัด ทำหน้าที่เป็นวงจรทวีคูณแรงดันแล้วแบ่งแรงดันเป็นแรงดันบวก และลบจ่ายให้แก่ IC เรกกูเรเตอร์ U_1 เบอร์ T7805CT และ U_2 เบอร์ LM7905CT เป็นแรงดัน ± 5 โวลท์ไปเลี้ยงบอร์ด DSK

4. เกี่ยวกับชิพ TMS 320C50

TMS320C50 เป็นหัวใจหลักของบอร์ด DSK เป็นผลงานของ เท็กซัสอินสตรูเมนท์ เป็นไมโครโปรเซสเซอร์ตัวหนึ่งที่ถูกออกแบบมาให้ใช้งานด้าน DSP โดยเฉพาะ มีคาบเวลาของไซเกิลของคำสั่งเป็น 100 นาโนวินาที ในการทำงานคำสั่งแต่ละตัวจะใช้เวลาหนึ่งไซเกิล ทำให้สามารถประมวลผลได้ถึง 10 ล้านคำสั่งต่อวินาที จึงส่งผลให้ชิพนี้สามารถทำการประมวลผลแบบเวลาจริง (real time) ได้ทัน

ในตัว TMS320C50 มีหน่วยความจำความเร็วสูงขนาด 1,568 เวิร์ด โดยที่แต่ละเวิร์ดจะมีขนาด 16 บิต และมีรอมภายในขนาด 256 เวิร์ด สามารถอ้างอิงหน่วยความจำภายนอกได้ถึง 128 กิโลเวิร์ด แบ่งเป็นหน่วยความจำข้อมูลและหน่วยความจำโปรแกรมอย่างละ 64 กิโลเวิร์ด ALU (Arithmetic Logic Unit) และ แอกคิวมูเลเตอร์ ขนาด 32 บิตที่สามารถคูณเลขขนาด 32 บิตได้ภายใน 1 แมชชีนไซเกิลทั้งนี้เนื่องจากตัว ALU มีวงจรคูณแบบ 16×16 บิตฮาร์ดไวร์ นอกจากนี้ยังมีตัวตั้งเวลาและส่วน

กำเนิดสัญญาณนาฬิกาอยู่ภายในชิพ ส่วนของการอินเตอร์เฟสเป็นแบบขนาน 16 บิต แบ่งเป็นอินพุท 16 แชนเนล เอาท์พุท 16 แชนเนล วงจรติดต่อหน่วยความจำร่วม (Global Data Memory Interface)

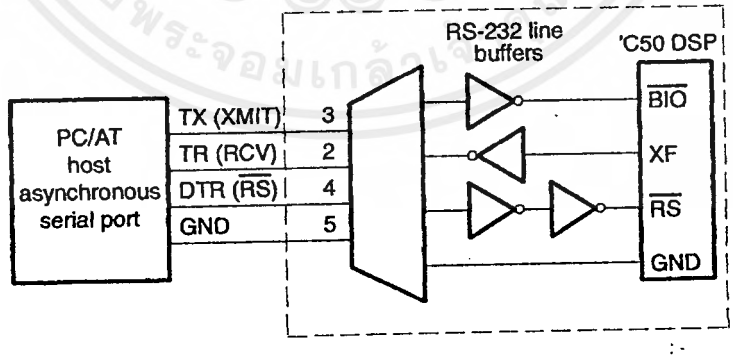
ตัว TMS320C50 สามารถตอใช้งานเพียงตัวเดียวหรือจะต่อใช้งานขนานกันหลายตัวที่เรียกว่าเป็นระบบ multi-processor system ก็ได้

ในการใช้งานกับหน่วยความจำภายนอกไม่ว่าจะเป็นอีพรอมหรือรอมตัวชิพจะทำงานได้ช้ามาก จึงได้เน้นให้ใช้หน่วยความจำแบบภายในชิพเป็นที่เก็บโปรแกรมแทน ในการใช้งานให้ดาวน์โหลดโปรแกรมจากหน่วยความจำภายนอกมาเก็บในชิพแล้วจึงรันโปรแกรมทำงานโดย TMS320C50 ได้เพิ่มคำสั่ง 'Block Transfer' ที่ใช้ในการโหลดข้อมูลจากหน่วยความจำภายนอกมาทีละบล็อค แล้วมาเก็บไว้ในแรมของชิพเอง ด้วยวิธีการนี้จะช่วยให้การทำงานเร็วมากขึ้น

การอ้างแอดเดรสของ TMS320C50 มีด้วยกัน 3 โหมด คือ โดยตรง (direct), ทางอ้อม (indirect) และทันทีทันใด (immediate) นอกจากนี้ยังมีรีจิสเตอร์เสริมอีก 8 ตัว

5. การติดต่อใช้งานบอร์ด DSK

การใช้งานบอร์ด DSK ทำได้ไม่ยาก อุปกรณ์ที่ต้องหาเพิ่มเติมคือ สายสัญญาณที่ใช้ต่อระหว่างบอร์ดกับคอมพิวเตอร์อาจใช้เป็นสายโมเด็มมาตรฐานก็ได้ โดยมีเงื่อนไขว่า ปลายด้านที่ต่อกับบอร์ด DSK ต้องเป็นคอนเน็กเตอร์ชนิด D 9 ขา ตัวผู้



รูปที่ 3 DSK to RS-232 Connections

6. ขั้นตอนการพัฒนาโปรแกรม

จุดประสงค์ของการพัฒนา คือ ต้องการสร้างซอฟต์แวร์ที่สามารถใช้งานกับบอร์ด DSK เป้าหมายได้โดยตรง ในชุด DSK จะมาพร้อมกับโปรแกรมแอสเซมเบลอร์และดีบักเกอร์ในการเขียนแก้ไขทดสอบและปรับปรุงซอฟต์แวร์ให้ได้ผลตามที่ต้องการ

7. ตัวแอสแซมเบลอร์

แอสแซมเบลอร์มีหน้าที่ในการแปลไฟล์ข้อมูลภาษาแอสเซมบลีให้เป็นไฟล์ภาษาเครื่องหรือที่มีชื่ออีกอย่างว่า Object code ที่ทำให้ชิพสามารถนำข้อมูลรหัสไปปฏิบัติงานได้โดยตรง ตัวแอสเซมเบลอร์ของ DSK เป็นแบบที่สามารถกำหนด 'directive' ก่อนที่จะทำการแอสเซมบลี้ได้ ทำให้สามารถกำหนดตำแหน่งของแอสเดรสของโปรแกรมเป็นแบบสัมพัทธ์ (คือเป็นแอดเดรสลอยๆที่สามารถกำหนดตำแหน่งที่แน่นอนได้ในภายหลัง) ทำให้ไม่จำเป็นต้องมีขั้นตอนการ linker

8. ดีบักเกอร์

ดีบักเกอร์มีหน้าที่ใช้ช่วยในการหาตำแหน่งและแก้ไขข้อผิดพลาด (error หรือ bug) ต่างๆที่อาจเกิดในดังโปรแกรม การทำงานของโปรแกรมจะแบ่งการทำงานในส่วนต่าง ๆแยกเป็นคนละหน้าต่างที่ง่ายต่อการใช้ ถึงแม้จะเป็นมือใหม่ก็ตาม ตัวดีบักเกอร์จะมีหน้าที่พื้นฐานที่สำคัญ ได้แก่ dinglestep execution, breakpoint setting และ runtime execution halt เป็นต้น

9. ติดตั้งซอฟต์แวร์

ก่อนใช้งานต้องทำการติดตั้งซอฟต์แวร์เสียก่อน โดยในชุด DSK นี้จะมีแผ่นดิสต์ 2 ชุด ชุดแรกจะเป็นแผ่น USER SOFTWARE อันเป็นแผ่นที่ใช้งานจริง ส่วนอีกชุดหนึ่งเป็น TMS320 Development Flow and Programmer's Interface ซึ่งบรรจุแนะนำวิธีการใช้งาน และพัฒนา TMS320C50

การติดตั้งจะติดตั้งทั้ง 2 แผ่นลงในฮาร์ดิสก์ก็ได้ หรือจะติดตั้งเฉพาะแผ่นใช้งานก็ได้ วิธีการก็ง่ายๆ เริ่มจากสร้างไดเร็กตอรีของ DSK เสียก่อนให้ชื่อว่า DSKTOOL จากนั้นก็อปปี้ทุกไฟล์ทุกไดเร็กตอรีย่อยในแผ่น USE SOFTWARE ลงในฮาร์ดดิสก์ เพียงเท่านี้ก็ใช้งานได้แล้ว

ส่วนแผ่นแนะนำนั้น จะมีลักษณะเหมือนโปรแกรมพรีเซนเตชั่น คือเป็นกราฟฟิกสี่สีสวยงามแนะนำขั้นตอนต่างๆ อย่างละเอียด ถ้าหากมีที่ว่างในฮาร์ดดิสก์ก็น่าจะติดตั้งลงไปด้วย

10. เริ่มต้นการใช้งาน

หลังจากติดตั้งโปรแกรมเรียบร้อยแล้ว ต่อสายระหว่างบอร์ด DSK กับคอมพิวเตอร์ จ่ายแรงดันไฟสลับ 9 โวลต์ให้แก่บอร์ด DSK ลองเอามือจับที่ไอซีเรกูเลเตอร์ บอร์ด 7805 ถ้าอุ่นแสดงว่าบอร์ด DSK ทำงานแล้วจากนั้นทำการทดลองรันโปรแกรม จากขั้นตอนการพัฒนาโปรแกรม ต้องเริ่มจากการเขียนโปรแกรมภาษาแอสเซมบลี้ แล้วทำการคอมไพล์โดยแอสเซมเบลอร์ จากนั้นจึงนำโปรแกรมมารัน

รายละเอียดของซอฟต์แวร์ทั้งหมด สามารถพิมพ์ออกมาอ่านรายละเอียดได้โดยพิมพ์ไฟล์ที่มีนามสกุล .DOC ทั้งหมดหรือถ้าต้องการศึกษาโปรมแกรมภาษาแอสเซมบลีให้พิมพ์ไฟล์ นามสกุล .ASM

ชิพ TMS320c50 ของบริษัท Texas Instruments จะทำการประมวลผลสัญญาญแบบจำนวนเต็ม (Fixed-point) ซึ่งจะต้องทำการแปลงจำนวนทศนิยมมาเป็นจำนวนเต็ม ก่อนจะนำไปประมวลผล ซึ่งชิพรุ่นใหม่นั้นสามารถประมวลผลสัญญาณแบบทศนิยม(Float-point) ได้เลยโดยไม่ต้องทำการแปลงเป็นจำนวนเต็ม

ชิพ DSP ตระกูล Fixed-Point ของบริษัท Texas Instruments มีการพัฒนามาหลายรุ่นซึ่งทั้งหมดจะมีชุดคำสั่งซึ่งสามารถใช้ร่วมกันได้โดยคำสั่งของรุ่นเก่าจะยังคงใช้ได้กับ ชิพ ตัวใหม่ ซึ่งสามารถทำให้ผู้ใช้ศึกษาเพิ่มเติม ไม่มากก็จะทำให้สามารถใช้ ชิพ รุ่นใหม่ ๆ ได้ไม่ยาก และหนึ่งในนั้นคือ ชิพ ในตระกูล C5x จะมีหลายรุ่นซึ่งจะแตกต่างกันในส่วนของ Memory ที่บรรจุมาภายในเพื่อให้ผู้ใช้เลือกนำไปใช้งานในลักษณะต่างๆ กันแต่ในส่วนหลักๆ ของการทำงานภายในจะเหมือนกันโดยในโครงงานนี้จะนำ ชิพ รุ่น C50 มาใช้งานแต่ในส่วนของทฤษฎีในบทนี้ จะเรียกรวมๆ ว่า C5x

11. การใช้งานทั่วไป

Automotive	Consumer	Control
Adaptive ride control	Digital radios/TVs	Disk drive control
Antiskid brakes	Educational toys	Engine control
Cellular telephones	Music synthesizers	Laser printer control
Digital radios	Power tools	Motor control
Engine control	Radar detectors	Robotics control
Global positioning	Solid-state answering machines	Servo control
Navigation		
Vibration analysis		
Voice commands		

General-Purpose	Graphics/Imaging	Industrial
Adaptive filtering	3-D rotation	Numeric control
Convolution	Animation/digital map	Power-line monitoring
Correlation	Homomorphic processing	Robotics
Digital filtering	Pattern recognition	Security access
Fast Fourier transforms	Image enhancement	
Hilbert transforms	Image compression/transmission	
Waveform generation	Robot vision	
Windowing	Workstations	
Instrumentation	**Medical**	**Military**
Digital filtering	Diagnostic equipment	Image processing
Function generation	Fetal monitoring	Missile guidance
Pattern matching	Hearing aids	Navigation
Phase-locked loops	Patient monitoring	Radar processing
Seismic processing	Prosthetics	Radio frequency modems
Spectrum analysis	Ultrasound equipment	Secure communications
Transient analysis		Sonar processing
Telecommunications		**Voice/Speech**
1200- to 19200-bps modems	DTMF encoding/decoding	Speech enhancement
Adaptive equalizers	Echo cancellation	Speech recognition
ADPCM transcoders	Fax	Speech synthesis
Cellular telephones	Line repeaters	Speaker verification
Channel multiplexing	Speaker phones	Speech vocoding
Data encryption	Spread spectrum communications	Voice mail
Digital PBXs	Video conferencing	Text-to-speech
Digital speech interpolation (DSI)	X.25 Packet Switching	
Personal digital assistants (PDA)	Personal communications systems (PCS)	

ตารางที่ 1 แสดงรายการของการใช้งาน C5X ในด้านต่างๆ

ด้วยลักษณะการทำงานแบบ Real Time และความสามารถในการใช้งานที่หลากหลายทำให้ ชิพ รุ่น C5x ใช้งานได้อย่างกว้างขวางในการแก้ปัญหาต่างๆ ที่มีการทำงานแบบ Signal-Processing เช่น การถอดรหัส หรือการกรองสัญญาณไม่เพียงเท่านั้นในรุ่น C5x ยังออกแบบมาเพื่อรองรับการทำงานที่มีการทำงานหลายอย่างในเวลาเดียวกันด้วย

ในรุ่น C5x ยังประกอบไปด้วย C50 , C51 และ C53 ซึ่งทั้งสามตัวเป็นสิ่งประดิษฐ์ประเภท Static CMOS ที่มีสถาปัตยกรรมภายในอยู่บนพื้นฐานของรุ่น C25 ด้วยการผสมผสานกันของสถาปัตยกรรมชั้นสูงที่มีการแยกระหว่าง Program - Bus และ Memory – Bus ด้วยการเพิ่มอุปกรณ์

สนับสนุนลงบน ชิพ การสร้าง On – chip Memory ที่มีขนาดใหญ่ขึ้นและคำสั่งพิเศษที่มีความสามารถสูงขึ้น ซึ่งเป็นพื้นฐานให้การทำงานที่มีความยืดหยุ่นและความเร็วสูงขึ้น

TMS320C51 ถูกออกแบบให้มีความเร็วในการทำงานมากกว่า 28 MIPS ในอนาคต คาดว่าจะมีการพัฒนาขึ้นอีก และสามารถตอบสนองความต้องการของตลาดได้มากขึ้น ในรุ่น C5x ออกแบบให้มีประโยชน์ใช้สอยได้ดังนี้

- ใช้หลักในการออกแบบที่มุ่งความเร็วสูงสุดและใช้งานได้ง่าย
- ใช้เทคโนโลยีขั้นสูงเพื่อเพิ่มความเร็วในการทำงาน
- มีความ Compatible (เข้ากันได้) กับรุ่นก่อนๆ ไม่ว่าจะเป็น C1x ,C2x ทำให้สามารถทำการ Upgrade ได้ง่าย
- มีการปรับปรุงชุดคำสั่งเพื่อความเร็วในการทำงานแบบ Algorithm (ขั้นตอนวิธี) ที่มีประสิทธิภาพมากขึ้นและเพิ่มความสามารถในการทำงานกับภาษาชั้นสูง
- ใช้เทคนิคแบบใหม่ในการออกแบบช่วยให้ประหยัดกำลังงานในการทำงานและทำให้การกระจายของคลื่นวิทยุไปสู่ภายนอกน้อยลง
- ใช้สถาปัตยกรรมชั้นสูงเพื่อเพิ่มความเร็วในการทำงานและสามารถทำงานได้หลายๆ อย่างในเวลาเดียวกัน

ตารางที่ 2 แสดงภาพรวมของความสามารถในรุ่น C5x โดยแสดงถึงความจุของ On - chip RAM และROM จำนวนของ Serial และ Parallel I/O port , Execution Time ของแต่ละ Machine Cycle และชนิดของ Package รวมทั้งจำนวนขาทั้งหมดของ ชิพ

TMS320 Device	on-chip memory			I/O Port		Cycle Time (ns)	Package type
	RAM		ROM	Serial	Parallel		
	Data	Data+Prog	Prog				
TMS320c50	1k	9k	2k	2	64k	50/35	132-pin plastic
TMS320c51	1k	1k	8k	2	64k	50/35	132-pin plastic
TMS320c53	1k	3k	16k	2	64k	50/35	132-pin plastic

ตารางที่ 2 แสดงภาพรวมของความสามารถในรุ่น C5x

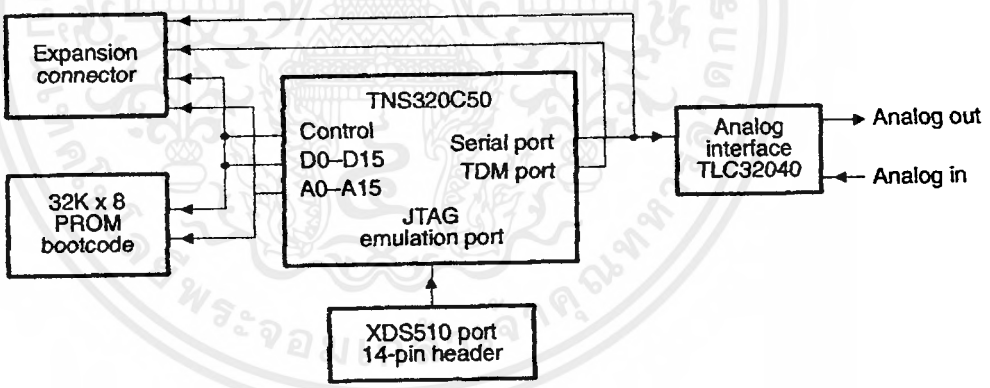
รายละเอียดของรุ่น C5x แสดงไว้ข้างล่างนี้โดยแสดงลักษณะที่สำคัญของแต่ละเบอร์ โดยชื่อของ ชิพ จะแสดงอยู่ในวงเล็บ

- ใช้เวลาทำงานตามคำสั่งแบบ Fixed-Point ด้วยความเร็ว 35-50 ns
- มีคำสั่งที่ Compatible กับรุ่น C1x , C2x
- RAM-Based Memory Operation (C50)
- 9k×16 bit on-chip program/data RAM (C50)
- 2k×16 bit on-chip boot ROM (C50)
- 1056×16 bit dual-access on-chip data RAM
- 32 bit ALU ,32bit ACC(accumulator) และ 32 bit ACCB (Accumulator Buffer)
- 16 bit PLU (Parallel Logic Unit)
- 16×16 bit parallel multiplier พร้อมทั้ง 32 bit product capability
- มีความสามารถในการคูณด้วยรอบคำสั่งเดียว
- 8 Auxiliary Register
- 11 Context – switch register (shadow register) เพื่อใช้ในการ Interrupt
- 0 -16 bit shifter และ 64 bit incremental
- 2 circular buffer สำหรับการอ้าง address แบบ circular มีคำสั่ง single – instruction repeat and block repeat
- มีคำสั่ง move แบบ block เพื่อการจัดการข้อมูลที่ดีขึ้น
- Full-Duplex Synchronous serial port เพื่อการสื่อสารระหว่าง C5x กับอุปกรณ์ภายนอก
- มี TMD (Time Division Multiple Access) Serial port
- 64k parallel i/o port และมี 16 port ที่สามารถ mapped ไปยัง memory
- 16 wait state generator ควบคุมการทำงานด้วย software
- เชื่อมต่อกับอุปกรณ์ภายนอกแบบ DMA ได้สะดวก
- ภายในประกอบด้วย 4 – deep pipeline สำหรับ delayed branch
- index – addressing mode
- Bit – Reverse index – addressing mode สำหรับ radix – 2 FFT
- มีความสามารถหารได้ภายใน clock ลูกเดียว

- มี clock generator ภายใน ชิพ
- JTAG boundary scans logic (IEEE standard, 1149.1)
- 5 volt static CMOS พร้อมด้วย power-down mode 2 mode

11. ส่วนสำคัญของ CPU

การพัฒนาขึ้นมาเป็นรุ่น C5x ยังใช้ Source Code ที่ Compatible กับรุ่น C1x และ C2x เมื่อมีการพัฒนา Program จึงทำได้ง่าย ในการพัฒนารุ่นต่างๆ ของ C5x นี้ยังมีการสร้าง 32 bit accumulator buffer (ACCB) เพิ่มเติมขึ้นมาด้วย เพื่อสนับสนุนการทำงานของ ACCใน Control function แบบใหม่มี PLU ซึ่งเป็นอิสระในการทำงานดังนั้นจึงสามารถทำงานในแบบสมการ Boolean ได้ดีขึ้น และมีการบริการการ interrupt ด้วย IRS จึงทำงานได้เร็วขึ้น การจัดการข้อมูลได้มีการปรับปรุงโดยมีการใช้คำสั่ง block move และมีคำสั่ง memory – mapped ในรุ่น C5x มี register สำหรับทำการ mapped memory ทั้งหมด 28 ตัว และมี register สำหรับทำการ mapped I/O ทั้งหมด 16 ตัว



รูปที่ 8 DSK Block Diagram

11.1 On – chip ROM

C50 มี 2K × 16 bit maskable programmable ROM หน่วยความจำนี้ใช้สำหรับการ Boot เครื่องในตอนเริ่มต้น ในกรณีที่ ROM ภายนอกมีความเร็วต่ำ โดยการเลือก ROM สามารถเลือกได้ในขณะ

Reset ด้วยขา MP / $\overline{MC}$ ในครั้งแรกที่ทำการ Boot เครื่อง โดย Boot ROM สามารถที่จะทำการยกเลิกการใช้งานด้วยการควบคุม bit ใน PMST Register

11.2 On – chip data RAM

อุปกรณ์ทั้งหมดในรุ่น C5x จะมี RAM ขนาด 1056 × 16 bit โดย RAM ในส่วนนี้จะมีความสามารถในการทำงานได้ถึง 2 ครั้งใน 1 machine cycle (dual – access RAM) โดยใช้สำหรับเก็บข้อมูลโดยเฉพาะแต่ก็สามารถเก็บProgram ในพื้นที่ส่วนนี้ได้ถ้าต้องการ

11.3 On – chip Program / Data RAM

C50 มี 9 K × 16 bit On – chip RAM ซึ่งพื้นที่หน่วยความจำส่วนนี้สามารถจัดโครงสร้างด้วย software ให้เป็นพื้นที่ Program หรือ data หรือ เป็นทั้งพื้นที่ program และ data ก็ได้

11.4 On – chip Memory Security

C5x มีระบบป้องกันข้อมูลที่บรรจุอยู่ภายใน On – chip Memory เมื่อมีการ Set ค่าของ Register ต่างๆ ที่เกี่ยวกับการแบ่งหน่วยความจำแล้วจะไม่มีคำสั่งจากภายนอกเข้าไปใช้งานในพื้นที่หน่วยความจำได้

11.5 Address – Mapped Software Wait - State Generator

การกำหนด Wait – State เมื่อติดต่อกับหน่วยความจำหรืออุปกรณ์ I/O สามารถกระทำได้โดยการ Set ค่าRegister ที่เกี่ยวข้องด้วย Software โดยไม่จำเป็นที่ต้องต่อ Hardware เพิ่มเติม ในส่วนนี้ประกอบไปด้วยวงจร Wait – State Generators จำนวน 16 ชุดด้วยกัน เพื่อทำการสร้าง Wait – State 0,1,2,3 หรือ 7 ในการติดต่อกับหน่วยความจำภายนอกและอุปกรณ์ I/O จะสามารถกำหนดขอบเขต เพื่อที่จะทำการ Map กับ Wait – State ได้ด้วยขนาด 16 K – word ต่อ 1 วงจร

11.6 Parallel I/O Ports

ชิพ ในรุ่น C5x จะมี I/O port ทั้งหมด 64 K-port โดยที่ 16 port ใดๆใน 64 K-port สามารถ Map ลงไปยัง Data Memory ได้ port เหล่านี้สามารถกำหนด Address ได้ด้วยคำสั่ง IN และ OUT และ I/O

port ที่ถูก Map ไปยัง Data Memory สามารถทำงานได้ด้วยคำสั่งอ่านและเขียนหน่วยความจำแบบปกติได้

11.7 I/O Ports

ชิพ ในรุ่น C5x ประกอบด้วย Serial port ความเร็วสูงจำนวน 2 port ซึ่งสามารถทำงานได้ด้วยความเร็วถึง ¼ ของ Machine Cycle (CLKOUT1) โดยที่ Serial port ตัวแรกทำงานแบบ Synchronous Full – Duplex และมี Buffer ภายในสำหรับการรับและการส่งข้อมูล ส่วน Serial port อีกตัวหนึ่งเป็นแบบ Full – Duplex ที่สามารถกำหนดการทำงานให้เป็นแบบ Synchronous หรือ TDM (Time Division Multiple Access) ก็ได้ซึ่ง TDM จะใช้งานทั่วไปในระบบ Multi – Processor

11.8 Hardware Timer

ชิพ ในรุ่น C5x จะมีวงจร Timing ขนาด 16 bit พร้อมด้วย 4 bit Pre-scalar ความถี่ Clock จะมีอัตราตั้งแต่ $1/2$ ถึง $1/32$ ของ Machine Cycle ขึ้นอยู่กับการ Program Divide-Down Ratio วงจร Timer สามารถควบคุมให้หยุด, เริ่มต้นใหม่, Reset หรือ Disable ด้วย Status bit พิเศษที่เป็นวงจร Timer

Registers	Description
SPC	serial port control register
DXR	data transmit register
DRR	data transmit register
XSR	transmit shift register
RSR	receive shift register

ตารางที่ 3 Serial Port Register

11.9 User-Maskable Interrupts

ชิพ ในรุ่น C5x มีสาย External – Interrupt จำนวน 4 เส้นดังนั้นอุปกรณ์ภายนอกแต่ละตัวสามารถควบคุม TMS320 ได้ และอุปกรณ์ภายในจะสามารถควบคุมการ Interrupt ได้ 5 ช่อง จาก Timer 1 ช่อง และ Serial อีก 4 ช่องด้วย

11.10 JTAG Scanning Logic

ชิพ ในรุ่น C5x มีความสามารถในการตรวจสอบการทำงานอย่างต่อเนื่องเกี่ยวกับทำงานของอุปกรณ์ภายนอกที่เชื่อมต่ออยู่และการทำงานภายใน CPU เองได้อย่างดี โดยที่ JTAG จะเชื่อมต่ออยู่กับวงจร Scanning Logic อื่นๆ ภายใน CPU ชิพ ในตระกูล C5x จึงสามารถทำงานเป็นวงจรตรวจสอบโดยอ่านค่าจากขาสัญญาณ JTAG Serial Scan และขาสัญญาณ Emulation – Dedicated

11.11 Packages

ชิพ ในรุ่น C5x มีลักษณะ Packages เป็น QFP (Quad Flat Pack Package) 132 pin โดยมีจุดประสงค์ให้กินพื้นที่ในบอร์ดให้น้อยที่สุด โดยใช้พื้นฐานเดียวกันกับการออกแบบ ชิพ C25

12. สถาปัตยกรรมภายใน

โครงสร้างสถาปัตยกรรมภายในของ TMS320 DSP ประกอบไปด้วยส่วนพื้นฐาน 3 ส่วนด้วยกัน

- Central Processing Unit (CPU)
- Memory
- Peripheral Interfacing Circuit

ซึ่งในส่วนนี้กล่าวถึง สถาปัตยกรรมและการทำงานหลักของ CPU C51 ที่มีความสามารถในการทำงานทางคณิตศาสตร์ด้วยความเร็วสูง ด้วยสถาปัตยกรรมการทำงานแบบขนานของมัน

<u>โครงสร้างสถาปัตยกรรมโดยรวม</u>

ชิพในรุ่น C5x เป็นชิพที่มีความสามารถสูงในการทำงานด้าน Digital Signal Processing โดยออกแบบให้มีความคล้ายคลึงกับ ชิพ ในรุ่น C25 ใช้สถาปัตยกรรมการทำงานแบบแยก Memory Bus ระหว่าง Program Bus และ Data Bus ให้มีการทำงานแยกกันต่างหาก เพื่อความเร็วสูงสุดในการทำงาน โดยมีคำสั่ง Transfer รองรับการทำงานทั้งสองส่วน

C5x สามารถใช้งานในการทำ 2 complement โดยการใช้ 32 bit ALU และ Accumulator ALU มีลักษณะเป็น General Purpose ที่ใช้ขนาด 16bit word จาก Data memory หรืออ้างแบบ Immediate จากคำสั่งหรือเก็บผลลัพธ์จากการคูณในขนาด 32 bit ด้วยการเพิ่มเติมการทำงาน ALU ทำให้ CPU สามารถทำคำสั่ง Boolean ได้ด้วย

Accumulator จะมีหน้าที่เป็น เอาท์พุท ของ ALU และในขณะเดียวกันมันก็เป็นทั้งอินพุทของ ALU ด้วย Accumulator มีขนาด 32 bit โดยแบ่งเป็น 16 bit บน (bit 31 ถึง bit 16) และ 16 bit ล่าง (bit 15 ถึง bit 0) ในชุดคำสั่งของ C5x ก็ยังมีที่เก็บข้อมูลชั่วคราวสำหรับ Accumulator ด้วยซึ่งมีขนาด 32 bit เท่ากัน มีชื่อเรียกว่า ACCB ในการเพิ่มความสามารถของ ALU มีการเพิ่มหน่วย PLU ขึ้นมาซึ่งสามารถทำคำสั่งเกี่ยวกับ Logic ได้ โดยการทำงานเกี่ยวกับ bit จะไม่ส่งผลกระทบไปยัง Accumulator ALU สามารถใช้ในการโอนย้าย bit ทำให้สามารถควบคุม bit และแก้ไข bit ได้อย่างรวดเร็ว เพื่อใช้ในการ set bit , clear bit , และ test bit สำหรับ status register ต่าง ๆ

การคูณเลข 16 bit 2 จำนวนแบบ 2 - Complement โดยให้ผลลัพธ์ออกมาเป็นเลข 32 bit สามารถกระทำได้ภายใน 1 รอบคำสั่ง ส่วนของการคูณประกอบไปด้วย 3 ส่วนสำคัญคือ Multiplier Array, PREG (Product Register) และ TREGO (Temporary Register) โดยที่ TREGO มีขนาด 16 bit และ PREG มีเลข 32 bit ในการคูณนั้นค่าของตัวคูณอาจจะนำมาจาก DATA Memory , Program Memory เมื่อใช้คำสั่ง MAC/MACD/MADS/MADD หรืออาจเป็นค่า Immediate ภายในคำสั่ง MPY# ก็ได้ ด้วยการคูณที่มีความสูงนี้ ทำให้ DSP มีการทำงานที่มีประสิทธิภาพสูงในการทำงาน เช่นการ Filer หรือ Correlation

ส่วนของ Scaling Shifter ภายใน C5x มีลักษณะที่เชื่อมต่อระหว่าง Data Bus และ ALU โดยมีอินพุท ขนาด 16 bit ทางด้าน Data Bus และ เอาท์พุท ขนาด 32 bit ทางด้าน ALU Scaling Shifter มีความสามารถในการเลื่อน bit ไปทางซ้ายได้ตั้งแต่ 0-16 bit จาก Input Data ที่เข้ามา ตามค่าที่กำหนดภายใน TREG1 โดยค่าทางขวาหลังจากการทำการเลื่อน bit แล้วจะถูกแทนที่ด้วยค่าศูนย์ ส่วนค่า MSB สามารถให้เป็น 0 หรือ Sign ก็ได้ ขึ้นอยู่กับสถานะของ Sign-Extension Mode bit (SXM) ของ status Register ST1 การเพิ่มความสามารถในการ Shift นี้ทำให้ Processor มีความสามารถเพิ่มขึ้นในด้าน Numerical-Scaling , Bit-Extraction , Extended-Arithmetic และ Overflow-Prevention

C5x มี Hardware Stack 8 ระดับ เพื่อป้องกันข้อมูลขณะ Interrupt และขณะทำคำสั่ง Call โดยขณะทำการ Interrupt Register หลักอันได้แก่ ACC, ACCB, ARCR, INDX, PMSTPREG, STO และ ST1 TREGS จะถูก Push ลงบน Stack ระดับแรกสุด และจะถูก POP ขึ้นมาเมื่อมีการ Interrupt Return หรือกลับจากการ Call

13. Memory

หน่วยความจำทั้งหมดของ ชิพ ในรุ่น C5x มีขนาดสูงสุด 224 K 16 – bit word พื้นที่ในหน่วยความจำแบ่งเป็น 4 ส่วนดังนี้คือ พื้นที่ Program พื้นที่ Program ขนาด 64 K-words , Local Data ขนาด 64 K-words , Global Data ขนาด 32 K-words และ I/O Port อีก 64 K ในสถาปัตยกรรมแบบขนาน จะทำให้อุปกรณ์แยกการทำงานได้ 3 ส่วน ได้แก่ Fetch , Read และ Write ซึ่งโครงสร้างและการทำงานอธิบายได้ดังนี้

1. Memory Space Chip

ใน C5x มีสถาปัตยกรรมหลายแบบ Memory Space ซึ่งสามารถทำงานใน Bus แบบขนาน ทั้งหมด 3 Bus สิ่งนี้เองที่ทำให้ CPU สามารถทำงานเกี่ยวกับ Program และ Data ได้ในเวลาเดียวกัน Parallel Bus ทั้งสามได้แก่ PAB (Program Read/Write Bus) , DAB1 (Data Read Bus) และ DAB2 (Data Write Bus) แต่ละ Bus ทำงานในพื้นที่หน่วยความจำที่แตกต่างกัน หน่วยความจำใน C5x แบ่งออกเป็นสี่ส่วนที่แยกเป็นอิสระจากกันคือ ส่วน Program , Local Data , Global Data และ Input/output Port ซึ่งสามารถ Mapped ให้อยู่ภายใน ชิพ หรืออยู่ข้างนอก ชิพ ก็ได้

Program Space จะเก็บค่าของคำสั่งที่จะทำการ Executed , Local Data Space เก็บค่าของ Data ที่ถูกใช้โดยคำสั่ง Global Data Space มีความสามารถในการ Share ข้อมูลระหว่าง Processor หรือสามารถจองไว้ใช้เพื่อเป็น Data Space ก็ได้ ส่วน I/O Space จะเชื่อมต่อไปยังอุปกรณ์ภายนอกในรูปแบบของ Memory Map ภายนอกและยังสามารถจองพื้นที่นี้ไว้ใช้เป็นพื้นที่เก็บ Data แบบ Extra ได้ด้วย

2. Program memory

ชิพ ในรุ่น C5x สามารถต่อหน่วยความจำเพิ่มเติมได้ถึง 64 K 16 bit word เมื่อต่อหน่วยความจำดังกล่าวเพิ่มขึ้นแล้ว จะทำให้ C5x มีหน่วยความจำทั้งหมดคือ On-Chip ROM , Single – Access Program / Data Ram และ Dual – Access RAM ด้วยการใช้ Software จะทำให้สามารถจัดดครงสร้างของหน่วยความจำแต่ละส่วนให้อยู่ภายในหรือภายนอก Address Map ก็ได้เมื่อ Map ไปยัง Program Space ชิพ จะประมวลผลข้อมูลจำกัดอยู่ภายในขอบเขตของมันโดยอัตโนมัติเมื่อ CPU สร้าง Address นอกเหนือไปจากขอบเขต ชิพ จะเปลี่ยนการทำงานไปประมวลผลข้อมูลจากอุปกรณ์ภายนอกโดยอัตโนมัติ

2.1 การจัดโครงสร้างของ Program space

หลังจากที่ทำการ Reset การจัดโครงสร้างของข้อมูล จะถูกกำหนดด้วยขา MC/MP ถ้าแรงดันที่ขานี้มีค่า High C51 จะจัดโครงสร้างข้อมูลเป็น Microprocessor และ On-Chip ROM จะไม่มีการนำมาใช้งานถ้าขานี้มีแรงดันเป็น Low C51 จะจัดโครงสร้างข้อมูลเป็น Microcomputer คือ On-Chip ROM จะถูกนำมาใช้งาน ดังนั้นถ้าต้องการนำ C51 ไปใช้งานเป็น Microcomputer มันจะเริ่มทำงานด้วยคำสั่งบน On-Chip ROM ไม่เช่นนั้นมันจะเริ่มการทำงานที่หน่วยความจำภายนอกในขณะที่ Program เริ่มทำงานที่หน่วยความจำภายนอกในขณะที่ Program เริ่มทำงานเราสามารถทำการเปลี่ยนแปลงค่าของ MP/MC ได้ใน PMST Register โดยการทำตามคำสั่งดังเช่นแสดงไว้ในบรรทัดข้างล่างนี้

```
OPL#8,PMST                    ;Remove boot ROM from program space
```

2.2 Program Memory Address Mapped

ค่าของ Reset , Interrupt และ Trap Vector จะถูกเก็บไว้ในพื้นที่ Program ซึ่งพื้นที่ในส่วนนี้จะอยู่ในตอนต้นของเนื้อที่ในหน่วยความจำ ในขณะที่ Reset ข้อมูลใน Vector Mapped จะมีค่าชี้ไปยัง Address oh

2.3 Local Data Memory

ขนาดของ Local Data Memory สามารถขยายได้ถึง 64 K 16 bit-word โดยใน ชิพ C51 สร้างเนื้อที่ขนาด 1 K ไว้บนชิพ c50C50 มี 2K16 bit maskable programmable ROM หน่วยความจำนี้ใช้สำหรับการ Boot เครื่องในตอนแรก ROM สามารถเลือกได้ในขณะ reset ด้วยขา mp/mc เราสามารถ set การทำงานได้ด้วยการควบคุม bit ใน PMST

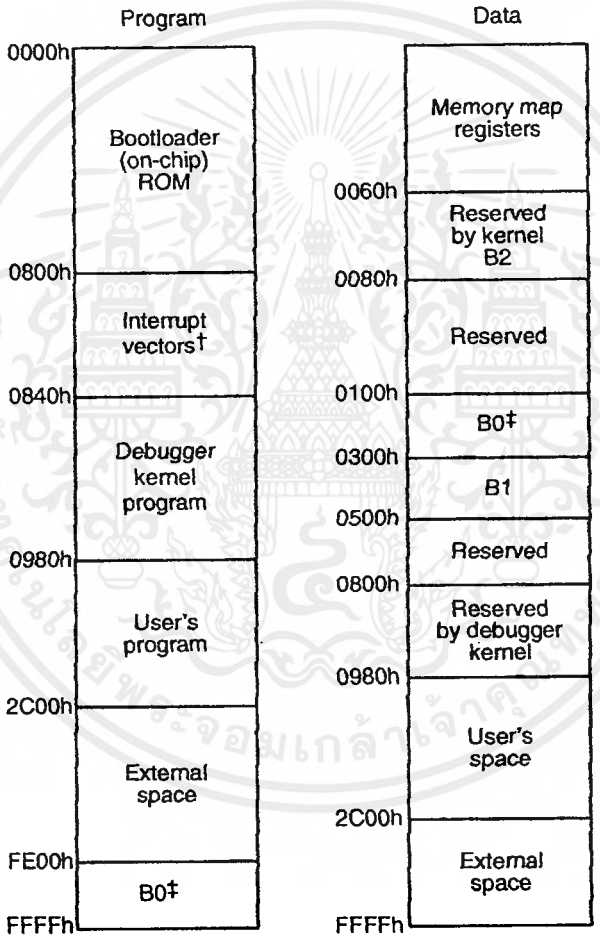
RAM ขนาด 1056×16 bit ซึ่งส่วนนี้จะสามารถทำงานแบบ dual-access สำหรับเก็บข้อมูลและสามารถใช้เก็บ program ในส่วนนี้ได้ด้วย

RAM 9k×16-bit on-ชิพ หน่วยความจำนี้สามารถใช้เป็นที่เก็บ program and data ก็ได้ Wait-state Generator จำนวน 16ชุด เพื่อทำการสร้าง wait-state ในการติดต่อกับหน่วยความจำภายนอกและอุปกรณ์ I/O จะกำหนดขอบเขตที่จะทำการ map กับ wait-state ได้

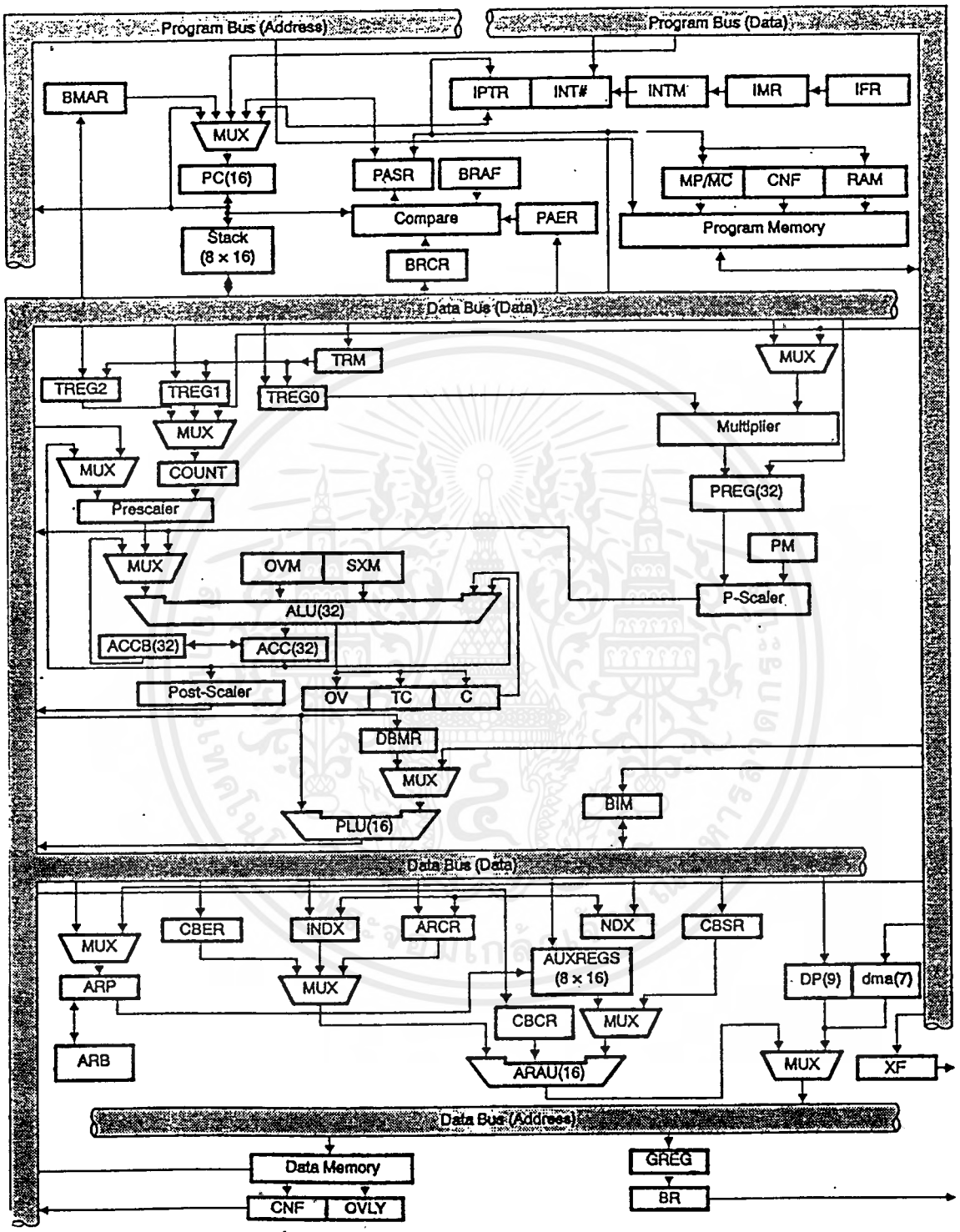
3. Local Data Memory

ขนาดของ Local Data Memory สามารถขยายได้ถึง 64 K 16 bit-word โดยใน ชิพ C51 จะสร้างเนื้อที่ขนาด 1 K ไว้บน ชิพ เป็นแบบ SARAM ชิพ ทั้งหมดในรุ่น C5x มีขนาดของหน่วยความจำ Dual-Access Space RAM (DARAM) ขนาด 1056 word โดยสามารถนำมาจัดโครงสร้างใหม่

ด้วย Software ให้อยู่ภายในหรือภายนอก Local Data Address Map ก็ได้ เมื่อหน่วยความจำส่วนนี้ถูก Map ลงใน Data Space ชิพ จะทำการประมวลผลข้อมูลโดยอยู่ภายในขอบเขตที่กำหนด ถ้ามีการกำหนดพื้นที่ใช้งานเกินขอบเขต Processor จะเปลี่ยนไปทำงานในพื้นที่ Address ภายนอกโดยอัตโนมัติ



รูปที่ 9 Memory Map of the DSK



รูปที่ 10 Block Diagram แสดงโครงสร้างภายในทั้งหมดของ C5x

4. Global Memory

สำหรับการทำงานแบบ Multi-Processing ชิพ ในรุ่น C5x มีความสามารถในการกำหนด Memory Space เพื่อใช้เป็นทางผ่านสำหรับสื่อสารข้อมูลระหว่าง Processor โดยอาศัยขาสัญญาณ BR และ Ready Global Memory เป็น Memory ที่จัดให้มีการใช้งานร่วมกันระหว่าง Processor มากกว่าหนึ่งตัว เพื่อที่ว่าการใช้งานร่วมกันจะทำได้โดยสะดวก Global Memory Address Space ถูกแบ่งออกเป็นส่วนๆ คือส่วน Local และ Global ในส่วน Local จะถูกใช้งานในหน้าที่ที่ต้องแยกอิสระจากกัน และในส่วน Global จะใช้เพื่อติดต่อสื่อสารกันกับ Processor ตัวอื่นๆ

5. อินพุท/เอาท์พุท Space

C5x มีพื้นที่รองรับอุปกรณ์ I/O ขนาด 64 K 16 bit เช่น Digital-to-Analog Converter (D/A)

Memory-Mapped Core Processor Registers			
Name	**Address**		**Description**
	Dec	**Hex**	
----------	0-3	0-3	Reserved
IMR	4	4	Interrupt Mask Register
GREG	5	5	Global Memory Allocation Register
IFR	6	6	Interrupt Flag Register
PMST	7	7	Processor Mode Status Register
RPTC	8	8	Repeat Counter Register
BRCR	9	9	Block Repeat Counter Register
PASR	10	A	Block Repeat Program Address Start Register
PAER	11	B	Block Repeat Program Address End Register
TREG0	12	C	Temporary Register Use for Multiplicand

TREG1	13	D	Temporary Register Use for Dynamic Shift Count (5 bits only)
TREG2	14	E	Temporary Register Use as Bit pointer in Dynamic Bit Test(4 bits only)
DBMR	15	F	Dynamic Bit Manipulation Register
AR0	16	10	Auxiliary Register Zero
AR1	17	11	Auxiliary Register One
AR2	18	12	Auxiliary Register Two
AR3	19	13	Auxiliary Register Three
AR4	20	14	Auxiliary Register Four
AR5	21	15	Auxiliary Register Five
AR6	22	16	Auxiliary Register Six
AR7	23	17	Auxiliary Register Seven
INDX	24	18	Index Register
ARCR	25	19	Auxiliary Register Compare Register
CBSR1	26	1A	Circular Buffer 1 Start Register
CBER1	27	1B	Circular Buffer 1 End Register
CBSR2	28	1C	Circular Buffer 2 Start Register
CBER2	29	1D	Circular Buffer 2 End Register
CBCR	30	1E	Circular Buffer Control Register
BMAR	31	1F	Block Move Address Register
Memory-Mapped Peripheral Registers			
DRR	32	20	Data Receive Register
DXR	33	21	Data Transmit Register
SPC	34	22	Serial Port Control Register
----------	35	23	Reserved
TIM	36	24	Timer Register

PRD	37	25	Period Register
TCR	38	26	Timer Control Register
----------	39	27	Reserved
PDWSR	40	28	Program/Data S/W Wait-State Register
LOWSR	41	29	I/O S/W Wait-State Register
CWSR	42	2A	S/W Wait-State Control Register
----------	43-47	2B-2F	Reserved
TRCV	48	30	TDM Data Receive Register
TDXR	49	31	TDM Transmit Data Register
TSPC	50	32	TDM Serial Port Control Register
TCSR	51	33	TDM Channel Select Register
TRTA	52	34	TDM Receive/Transmit Address Register
TRAD	53	35	TDM Received Address Register
Name	Address		Description
	Dec	Hex	
----------	54—79	36-4F	Reserved
Memory-Mapped I/O Ports			
PA0	80	50	I/O Port 50h
PA1	81	51	I/O Port 51h
PA2	82	52	I/O Port 52h
PA3	83	53	I/O Port 53h
PA4	84	54	I/O Port 54h
PA5	85	55	I/O Port 55h
PA6	86	56	I/O Port 56h
PA7	87	57	I/O Port 57h

PA8	88	58	I/O Port 58h
PA9	89	59	I/O Port 59h
PA10	90	5A	I/O Port 5Ah
PA11	91	5B	I/O Port 5Bh
PA12	92	5C	I/O Port 5Ch
PA13	93	5D	I/O Port 5Dh
PA14	94	5E	I/O Port 5Eh
PA15	95	5F	I/O Port 5Fh

ตารางที่ 4 Memory-Mapped Register And I/O Ports

การทำงานติดต่อกับ Parallel / Data Memory แต่พื้นที่ใช้งานของ I/O จะถูกแยกออกจาก Program / Data Memory ด้วยสัญญาณ IS Port ทั้งหมดจำนวน 65536 Ports สามารถทำงานได้ด้วยคำสั่ง IN และ OUT

16 Address จาก 64 K I/O Port จะถูก Mapped ลงบนพื้นที่ของ Data Memory ทำให้ I/O Port สามารถเขียนหรืออ่านได้ด้วยคำสั่ง IN และ OUT เช่นเดียวกับการประมวลผลคำสั่งอื่นที่อยู่ภายใน Data Space ดังตัวอย่างต่อไปนี้ ซึ่งอธิบายการอ้างตำแหน่งแบบ Direct Address เพื่อทำงานเกี่ยวกับอุปกรณ์ I/O ในตำแหน่ง 51 h

6. Software Application

C5x ยึดถือรูปแบบ Source Code .ให้ Compatible กับ C1x และ C2x และมีโครงสร้างที่พัฒนาขึ้นเพื่อปรับปรุงประสิทธิภาพในการทำงานและการใช้งานได้กว้างมากขึ้น ชุดคำสั่งใน C5x จะมีเพิ่มขึ้นเพื่อรองรับ Hardware รวมทั้งเพิ่มการทำ Data Movement และ Memory Mapped Register รวมไปถึงรองรับ Parallel Logic Unit (PLU) เพื่อการทำงานเกี่ยวกับคณิตศาสตร์ Boolean ,32 bit Accumulator Buffer และ register ที่ใช้สำหรับบริการการ Interrupt เนื่องจาก On-chip DARAM

1. Processor Initialization

ก่อนทำการประมวลผล Algorithm มีความจำเป็นที่จะต้องกำหนดค่าเริ่มต้นต่างๆให้แก่ Processor โดยทั่วไปการ Initialization จะกระทำทุกครั้งหลังจากที่มีการ Reset โดยหลังทำการ

Reset แล้ว IPTR bit ของ PMST จะถูก Clear ดังนั้น Vector จากการ Mapping จะชี้ไปที่ Page 0 ของพื้นที่ Program Memory ซึ่งพื้นที่ดังกล่าวโดยปกติแล้วบรรจุคำสั่ง Branch เพื่อชี้ไปยังตำแหน่งของ Initialization Routine ส่วนในHardware จะมีการ Disable Interrupt โดยอัตโนมัติ โครงสร้างของ Processor หลังจาก Reset ที่จะต้องทำการ Initialization ใหม่ได้แก่ส่วนต่างๆ ดังต่อไปนี้

- Memory-Mapped และ Peripheral Control Register
- Interrupt Structure (INTM)
- Mode Control (OVM,SXM,PM,AVIX,NDX,TRM)
- Memory Control (RAM,OVLY,CNF)
- Auxiliary Register และ Auxiliary Pointer (ARP)
- Data Memory Page Pointer (DP)

โดยที่ OVM (Overflow Mode), Tc (Test / Control Flax), IMR (Interrupt Mask Register), Auxiliary Register Pointer (ARP) , Auxiliary Register Pointer Buffer (ARB) และ Data Memory Page Pointer (DP) ไม่ต้องทำการ Initial ใหม่หลังทำการ Reset

2. Interrupt

C5x ประกอบไปด้วย External Maskable Interrupt จำนวน 4 ช่อง และ Non-Maskable Interrupt อีกช่องหนึ่งไว้ให้กับอุปกรณ์ภายนอก และยังมี Interrupt ภายใน ซึ่งได้แก่ Interrupt ที่เกิดจาก Serial Port , Timer และ Interrupt ที่เกิดจาก Software ในคำสั่ง Interrupt อันได้แก่ INTR , TRAP และ NMI

C5x มีความสามารถในการสร้าง Software Interrupt โดยใช้คำสั่ง INTR ซึ่งจะทำให้บางส่วนจาก 32 Interrupt Service Routine ทำการบริการการ Interrupt โดยที่ 20 ISR ถูกจองไว้เพื่อบริการแก่ External Interrupt , Peripheral Interrupt และสงวนไว้ใช้ในอนาคต ส่วน Interrupt Vector ที่เหลืออีก 12 Location สามารถนำไปใช้ได้โดยผู้เขียน Program ในคำสั่ง INTR สามารถอ้าง Interrupt ใดๆ ก็ได้ภายใน 32 Location

Context Saving และ Restoring Function จะทำงานเมื่อ Interrupt Trap ถูก Execute และ 8 Deep Hardware Stack จะถูกใช้เพื่อเก็บค่า Address ที่จะต้องกลับมาทำงานต่อหลังจากการบริการการ Interrupt เสร็จสิ้นเรียบร้อยแล้ว ดังนั้นจะมี One Deep Stack (หรือ Shadow Register) เพื่อเก็บค่าของแต่ละ Register ดังต่อไปนี้

ACC Accumulator

ACCB	Accumulator Buffer
PREG	Product Register
STO	Status Register 0 (INTM not restored)
ST1	Status Register 1 (XF not restored)
PMST	Processor Mode Status Register
TREG0	Temporary Register for Multiplier
TREG1	Temporary Register for Shift count
TREG2	Temporary Register for Bit test
INDX	Indirect Address Index Register
ARCR	Auxiliary Register compare Register

เมื่อได้รับสัญญาณ Interrupt Trap Register ทั้งหมดจะถูก Push ลงบน One-Deep Stack และ Shadow Register เหล่านี้จะถูก Pop เมื่อมีการกลับจาก Interrupt ด้วยคำสั่ง RET1 หรือ RETE

3. Software Stack

C5x มี Hardware Stack ภายในจำนวน 8 Deep ด้วยกันเพื่อใช้ในการเก็บค่า Address ในขณะทำการ Subroutine และ Interrupt

คำสั่ง PUSH และ POP มีการเพิ่มความสามาถของคำสั่งขึ้นมาเป็นคำสั่งใหม่อีก 2 คำสั่งได้แก่ PSHD และ POPD ทำให้ Stack สามารถ Store และ Recover ได้จาก Data Memory เมื่อมี Stack ถูกใช้ไปทั้งหมด 7 ค่าและมีการเขียนค่า Stack ทับลงไปอีกมากกว่า 2 ค่า โดยที่ค่าเก่ายังไม่ถูก POP ขึ้นมาจำเป็นที่จะต้องมีการขยาย Stack

เนื่องจากคำสั่ง Call ใช้ Stack ในการเก็บค่า Program Counter คำสั่ง BACC จะมีประโยชน์ในการเข้าสู่ Main Program โดยไม่จำเป็นต้องเก็บค่าของ Program Counter ไว้ใน Memory

4. Logical and Arithmetic Operations

PLU จะช่วยทำให้การส่งผ่านข้อมูลข้ามไปยัง Data Memory ทำได้โดยไม่ส่งผลกระทบต่อค่าภายในAccumulator หรือ Product Register ทำให้สามารถเคลื่อนย้ายค่าไปยังตำแหน่งใดๆ ภายใน Data Memory ได้โดยตรงกับ Source Operand

C5x ประกอบไปด้วยเงื่อนไขต่างๆ เพื่อทำการตรวจสอบก่อนที่จะกระโดดไปทำโปรแกรมอื่นๆบางส่วนจาก 13 เงื่อนไข โดยสามารถทดสอบแต่ละเงื่อนไขหรือทดสอบเงื่อนไขร่วมกันได้โดยใช้คำสั่ง CC , RETC , XC และ BCND

Search Algorithm ด้วยคำสั่ง CRGT ด้วยการใช้คำสั่ง CRGT และ RPTB เราสามารถค้นหาค่าสูงสุดภายใน Data Block และตำแหน่งของมันได้ การหาค่าต่ำสุดสามารถใช้คำสั่ง CRLT ค้นหาได้ในลักษณะเดียวกัน

5. Matrix Multiplication Using Nested Loops

C5x มีคำสั่ง 3 คำสั่ง คำสั่งแรกคือ RPT (Single-Instruction Repeat) สามารถทำคำสั่งซ้ำได้ n คำสั่ง คำสั่ง RPTB (Repeat Block) ทำงานด้วยจำนวน Loop ที่ตั้งไว้ภายใน BRCR Register ส่วนคำสั่ง BANZ (Branch if AR Not Zero) เป็นอีกคำสั่งที่สามารถกำหนดจำนวน loop ได้โดยใช้ Auxiliary Register

6. Circular Buffers

Circular Addressing เป็นหัวข้อสำคัญในชุดคำสั่งของ C5x การทำงานแบบ Convolution , Correlation และ FIR Filter สามารถทำงานได้โดยการใช้ circular Buffer ภายใน Memory C5x มีการรองรับการทำงานแบบต่อเนื่องกันระหว่าง 2 Buffer โดยใช้ Auxiliary Register โดยมี CBSR1 , CBSR2 , CBER1 ,CBER2 และ CBCR ทำหน้าที่เป็น Memory – Mapped Register ที่ควบคุมการทำงานของ Circular buffer

โดยก่อนที่จะเริ่มทำงานจะต้องมีการ load ค่า Address เริ่มต้นและสุดท้ายลงใน Buffer Register และ Auxiliary Register ทำงานเป็น Pointer จะต้องทำการ Initial ค่าด้วย

7. Extended-Precision Arithmetic

การวิเคราะห์ตัวเลข , การคำนวณ Floating – Point หรือการทำงานอื่นๆ อาจมีความจำเป็นที่จะต้องใช้ตัวเลขที่มีค่ามากกว่า 32 bit เนื่องจาก C5x เป็น 16/32 bit Fixed – Point Processor ดังนั้นในการคำนวณจึงต้องอาศัยการขยาย bit ด้วย Software เข้าช่วยในการทำงาน

7.1 การคูณ

มีคำสั่งในการใช้งาน เช่น MAC MACD MPY ซึ่งจะมีหน้าที่ในการคูณเลขแบบ 16 bit และผลลัพธ์ที่ได้จากการคำนวณ จะมีขนาด 32 bit ไว้ภายใน PREG (Product Register)

7.2 การหาร

มีคำสั่งในการใช้งาน เช่น SUB SUBC เป็นคำสั่งในการลบหลายๆ ครั้งโดยกำหนดให้ตัวตั้งและตัวหารมีค่าเป็นบวก 16 bit จะทำการลบทั้งหมด 16 ครั้ง แล้วจึงเก็บค่าผลหารที่มีขนาด 16 bit ไว้ในครึ่งล่างของ ACC(accumulator) และเก็บอีก 16 bit ไว้ในส่วนครึ่งบนของ ACC การใช้คำสั่ง SUBC ทำได้ด้วยการลบหลายๆครั้ง ตัวตั้งจะถูกชิฟจนกระทั่งตัวหารถูกลบจนหมด แล้ว

เปลี่ยนเป็นค่าลบ ในการที่ไม่ทำให้คำตอบมีค่าเป็นลบค่าหนึ่งจะถูกเลื่อนเข้าไปแทนที่ใน ครึ่งล่างของ ACC การหารที่กล่าวมานั้นตัวตั้งและตัวหารจะต้องมีค่าเป็นบวก โดยใช้คำสั่ง Absolute

14. REGISTER ที่สำคัญกำหนดสถานะที่สำคัญใน C5x

การทำงานของหลายๆส่วนจะถูกกำหนดโดย Register ทั้งหมด 4 ตัว คือ ST0 , ST1 , CBCR และ PMST การเปลี่ยนแปลงค่า การเขียน และการเก็บของเดิมไว้เมื่อมีการเรียกใช้ subroutine หรือเมื่อเกิดการ interrupt โดย Register PMST และ CBCR จะอยู่ใน Data Memory ดังนั้นจึงถูกกระทำโดย ALUและ PLU

ส่วนค่าใน ST0 และ ST1 จะถูกเขียนและอ่านโดยคำสั่ง LST ตามลำดับไม่สามารถเขียนและอ่านเหมือน PMST กับ CBCR ได้แต่คำสั่ง LST ไม่สามารถกำหนด Bit INTM ได้แต่ไม่ทั้งหมดสามารถใช้คำสั่ง SETC , CLRC เพื่อทำให้เป็น 1 และ 0 ได้ตามลำดับ

ST0(status register 0) มีขนาด 16 บิต

15	14	13	12	11	10	9	8	0
APR			OV	OVM	1	INTM	DP	

ARP Auxiliary Register Pointer 3 bit เป็นตัวชี้ Register AR เพื่อใช้ในการอ้าง address แบบ indirect เมื่อทำการเก็บค่าของ ARP ค่าของ ARP เดิมจะถูกเก็บใน ARB โดยค่าใน ARP สามารถเปลี่ยนได้หลายทางโดย การเปลี่ยนค่า เกี่ยวกับการอ้าง memory ของหลายคำสั่งหรือ คำสั่ง MAR

OV Overflow Flag bit จะเป็น 1 เมื่อเกิดการ Overflow ขึ้นที่ ALU สามารถเซ็ตบิตนี้จะเป็น 1 ด้วยคำสั่ง BCND หรือ LST

OVM Overflow Mode bit ถ้าเป็น 1 จะมีการ check การเกิด Overflow จาก Time และ counter ด้วยสามารถใช้คำสั่ง SETC CLRC และ LST ทำการเปลี่ยนแปลงค่า bit นี้ได้

INTM Interrupt Bit Mode เมื่อ bit นี้เป็น 0 จะมีเฉพาะ Interrupt แบบ Non-Maskable เท่านั้นที่สามารถทำงานได้ แต่เมื่อเป็น 1 ทุก ๆ Interrupt จะสามารถทำงานได้ โดย

เมื่อทำการ Reset bit นี้จะเป็น 1 สามารถทำการเปลี่ยนแปลงค่าด้วยคำสั่ง SETC, CLRC เหมือน bit อื่น ๆ แต่ไม่สามารถใช้คำสั่ง LST กับ bit นี้ได้โดยการเกิด Interrupt Maskable จะทำให้ bit นี้เป็น 0 และเมื่อมีการกับจาก Interrupt Service Routine ด้วยคำสั่ง RETE จะทำให้ bit กลายเป็น 1

DP Data Memory Page Pointer ค่า 9 bit ในนี้จะเป็นค่าที่ 7 bit ในคำสั่งในการอ้าง Memory แบบ Direct โดยโดย9 bit นี้จะเป็น MSB ในการอ้างMemory ซึ่งมี 16 bit ค่านี้สามารถเปลี่ยนแปลงโดยคำสั่ง LDP หรือ อ่านโดยคำสั่ง LST

ST1(Status Register 1) มีขนาด 16 บิต

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
ARB			CNF	TC	SXM	C	1	1	HM	1	XF	1	1	PM

ARB Auxiliary Register Pointer Buffer เป็นตัวกำหนดค่าสำรองของ ARP

CNF On-chip RAM Configuration Control Bit ถ้าเป็น 0 แสดงว่า RAM สามารถเป็นได้ทั้ง Program และ Data ถ้าเป็น 1 แสดงว่า RAM ใช้เป็นส่วนของ Program เพียงอย่างเดียว สามารถกระทำด้วยคำสั่ง SETC,CLRC,LST#1 เมื่อ RESET จะเป็น0

TC Test Control/flag Bit โดย bit สามารถเก็บผลของ ALU และ PLU ในการ Test bit การทำงานของชิพจะมีผลต่อคำสั่งBITT,CMPR,CPL,OPL,NORM และ XPL TC=0 เมื่อใช้คำสั่ง CLRC TC TC=1 เมื่อใช้คำสั่ง SETC TC

SXM Sign Extension Mode จะมีผลต่อเครื่องหมายการกระทำทางคณิตศาสตร์ SXM= 1 จะมีการเช็คเครื่องหมายทางคณิตศาสตร์โดยคำสั่ง CLRC SXM SXM= 0 จะไม่คำนึงถึงเครื่องหมายทางคณิตศาสตร์โดยคำสั่ง SETC SXM

C Carry Bit เกิดจากการกระทำทางคณิตศาสตร์ที่มีการยืม การทดในการคำนวณ C=0 เมื่อผลของการลบเกิดการยืมหรือผลจากการบวกที่ไม่ทำให้เกิด carry bit โดยใช้คำสั่ง CLRC C C=1 เมื่อผลของการบวกเกิดการทด หรือผลของการลบที่ไม่เกิดการยืมโดยใช้คำสั่ง SETC C

HM Hold Mode BitHM=0 เมื่อใช้คำสั่ง CLRC HM จะกระทำใน Program Memory ก่อนจนหมดแล้วทำการเซ็ตขาที่ทำการติดต่อกับสัญญาณภายนอกให้เป็น high impedance HM=1 เมื่อใช้คำสั่ง SETC HM จะหยุดกระทำทุกอย่างแล้วทำการactive ขา hold

XF XF Pin Status Bit เป็นขาแสดงสถานะของขา XF สามารถควบคุมอุปกรณ์ภายนอก XF=0 เมื่อใช้คำสั่ง CLRC XF XF=1 เมื่อใช้คำสั่ง SETC XFในการกระทำinterrupt จะไม่มีการเก็บค่าบิตนี้ไว้ ถ้าใน interrupt Routine มีการแก้ไขค่าบิตนี้จะไม่มีการคืนค่าเดิมหลังการ Return หลังการ interrupt

PM Product Shift Mode จะกำหนดเงื่อนไขการเก็บผลการคูณซึ่งจะมีการ shift หรือไม่ถ้าเป็น 00 ผลคูณจะไม่มีการ shift ถ้าเป็น 01 ผลคูณจะมีการ shift left ไป 1 บิต ถ้าเป็น 11 ผลคูณจะมีการ shift left ไป 4 บิตแล้วทำการแทน LSB ด้วย 0 ถ้าเป็น 11 ผลคูณจะมีการ shift left ไป 6 บิต จะมีผลกรณีที่มีการเก็บค่าไปยัง Data Memory ด้วย สามารถ เขียนด้วยคำสั่ง SPMและ LST#1

CBCR (Circular Buffer Control Register)

15 8	7 6	5 4	3 2	1 0
XXXX	CENB2	CAR2	CENB1	CAR1

CBSR1 เป็นรีจิสเตอร์ขนาด 16 บิตใช้เก็บค่าตำแหน่งแอสเดรสเริ่มต้นของ Circular Buffer#1

CBER1 เป็นรีจิสเตอร์ขนาด 16 บิตใช้เก็บค่าตำแหน่งแอสเดรสสุดท้ายของ Circular Buffer#1

CBSR2 เป็นรีจิสเตอร์ขนาด 16 บิตใช้เก็บค่าตำแหน่งแอสเดรสเริ่มต้นของ Circular Buffer#2

CBER2 เป็นรีจิสเตอร์ขนาด 16 บิตใช้เก็บค่าตำแหน่งแอสเดรสสุดท้ายของ Circular Buffer#2

CENB1 Circular Buffer 1 ทำงานที่ 1 และไม่ทำงานที่ 0

CENB2 Circular Buffer 2 ทำงานที่ 1 และไม่ทำงานที่ 0

CAR1 รีจิสเตอร์ช่วยชี้ตำแหน่งแอดเดรสใน Circular Buffer 1(AR0-AR7)

CAR2 รีจิสเตอร์ช่วยชี้ตำแหน่งแอดเดรสใน Circular Buffer 2(AR0-AR7)

การใช้งาน Circular Buffer ใน TMS320c50 สามารถทำได้โดยการกำหนดแอด เดรส เริ่มต้นและกำหนดแอดเดรสสุดท้าย ซึ่งจะต้องกำหนดรีจิสเตอร์ช่วยชี้ตำแหน่งแอดเดรส (AR0 ถึง AR7 ตัวใดตัวหนึ่ง)

PMST (Processor Mode Status Register)

	10	9	8	7	6	5	4	3	2	1	0
IPTR	0	0	0	AVIS	0	OVLY	RAM	MP/MC	NDX	TRM	BRAF

IPTR Interrupt vector pointer pointer bits เป็นบิตที่ใช้สำหรับเคลียร์ Vector ที่อยู่ภายในแอสเดรสใน program memory

AVIS เป็นบิตที่กำหนดให้ enable/disable program address ภายในมองเห็น address pins AVIS=1จะไม่มีการเปลี่ยนแปลงของขาแอสเดรส เมื่อมีการอ้างแอสเดรสภายใน AVIS=0 การเปลี่ยนแปลงภานใน จะแสดงที่ขาแอสเดรส

OVLY Ram Over Bit นี้จะทำการกำหนด RAM ส่วนที่สำหรับเก็บโปรแกรมสามารถกำหนดได้ด้วยตาราง

CNF	OVLY	DARAM B0	DARAM B1	DARAM B2	SARAM	OFF-chip
0	0	100h-2FFh	300h-4FFh	60h-7Fh		800h-FFFFh
0	1	100h-2FFh	300h-4FFh	60h-7Fh	800h-2BFFh	2C00h-FFFFh
1	0	-	300h-4FFh	60h-7Fh		800h-FFFFh
1	1	-	300h-4FFh	60h-7Fh	800h-2BFFh	2C00h-FFFFh

ตารางที่ 5 Local Data Memory Configure Control

RAM เมื่อบิตนี้เป็น 1 จะกำหนดให้ RAM ภายในเป็นส่วนของการเก็บโปรแกรม ถ้าเป็น 0 จะไม่ใช้ สามารถแสดงการใช้ RAM ดังตาราง

OVLY	RAM	ON-chip SARAM CONFIGURATION
0	0	Disabled
0	1	Mapped into program space
1	0	Mapped into data space
1	1	Mapped into both program and data space

ตารางที่ 6 On-chip Single-Access RAM Configuration Control

MP/MC Microprocessor/Microcomputer Bit เมื่อเป็น 0 จะมีการใช้ ROM ภายในเมื่อเป็น 1 จะไม่มีการใช้ ROM ภายในโดยจะอ่านสถานะจากขานี้ในการ Reset ครั้งแรกหลัดจากนั้นการ Reset ครั้งต่อไปจะไม่มีผล

NDX Enable Extra Index Register ถ้าเป็น 0 จะทำงานในโหมดการอ้าง Address แบบ indirect ที่เหมือนกับตระกูล C2x ซึ่งก็คือใช้ในการอ้าง Address แบบ Indirect แบบ Indexing หรือการเปรียบเทียบค่า AR ได้เฉพาะ ARO เท่านั้น แต่ในการทำงานของ C5x จะสามารถทำได้หมดเมื่อกำหนดให้ bit นี้เท่ากับ 1 โดยจะเป็น 0 เมื่อทำการ Reset

TRM Enable Multiple TREGs เป็น bit กำหนดการทำงานของ TREG TRGE TREG ให้ทำงานในโหมดต่าง ๆ คือถ้าเป็น 0 จะทำงานโหมดที่เหมือนกับ C2x ซึ่งทั้ง 3 Register จะมีผลเหมือนกันกับคำสั่งของ C2x แต่ถ้าเป็น 1 จะสามารถแยกใช้ T-Register แต่ละตัวเป็น Pre-Scaling หรือใช้กับคำสั่ง BITT ได้โดย bit นี้จะเป็น 0 เมื่อทำการ reset

BRAF Block Repeat Active Flag เป็น bit แสดงการทำงานของ Block repeat ซึ่งจะเป็น 1 เมื่อมีการทำงานโดยเมื่อทำการ Reset bit นี้จะเป็น 0

15. Register อื่น ๆ ที่สำคัญ

Software wait-state Register ในการใช้งาน C5x ต่อกับอุปกรณ์ภายนอกถ้าอุปกรณ์ภายนอกมีการทำงานช้ากว่าการอ้างถึงของ C5x มันจะอนุญาตรอให้มีการทำงานรอการทำงานของอุปกรณ์ภายนอกซึ่งสามารถกำหนดได้ด้วย Software โดยมี Register ต่างๆ มากำหนดค่าพวกนี้ซึ่งจะมีทั้งหมด 3 ตัว คือ PDWSR (16 BIT) และ CWSR (5 bit) ซึ่ง PDWSR จะเป็นตัวกำหนดการแบ่ง Memory ออกเป็น 4 Block อย่างละ 16k ทั้งของ Program Memory และ Data Memory , ส่วน LOWSR จะควบคุมทั้ง 16 ส่วนของ1/0 โดยแบ่ง 1/0 ออกเป็น 16 ส่วนคือสามารถมี 1/0 ที่มี Wait State เท่ากันได้ 16 ชุด ส่วน CWSR จะเป็นตัวกำหนด Range ของ Wait State โดย bit Big ใน CWSR จะเป็นตัวกำหนดโดยถ้าเป็น 0 จะกำหนด Wait State แยกเป็น port เลย แต่ถ้าเป็น 1 จะกำหนดเป็น block อย่างละ 8k

โดยที่ Range ของ Wait State สามารถกำหนดได้ถึง 7 Machine Cycle เพื่อให้การติดต่อกับอุปกรณ์ภายนอกสามารถควบคุมได้โดยผ่านทางขา Ready

การแบ่ง bit ใน Register ทั้งสองออกเป็นคู่ ๆ เพื่อกำหนด Wait State ให้ทั้งส่วนของ Program และ Data (ใน PDWSR) ซึ่งสามารถมีได้ 4 ค่าของ Wait State โดยจะเป็นค่าอะไรต้องดูจากค่าใน CRSR ประกอบเช่นใน bit 0 กับ 1 จะเป็นตัวกำหนด Wait State ของ Program MEMORY จากช่วง Address 000h ถึง 3fffh และส่วนของ 1/0 สามารถกำหนดได้ 2 แบบ ขึ้นอยู่กับ bit Big ใน CWSR โดยที่ถ้า bit Big เป็น 0 จะเป็นตัวกำหนดให้ 1/0 port ที่ Map เป็น Register เป็นคู่ ๆ แต่ถ้า bit Big เป็น 1 จะเป็นตัวกำหนด Wait State ให้ กับ 1/0 ที่แบ่งเป็นช่วง ๆ เหมือน memory โดยค่าของ Wait State แต่ละค่าจะดูจากใน CWSR อีกครั้ง การกำหนดค่าของ Wait State นั้นจะเป็นการทำกับอุปกรณ์ภายนอกเท่านั้นจะไม่มีผลกับ Memory ภายในแต่อย่างใด

โดยที่การกำหนดค่าใน CWSR จะทำการกำหนดค่าของ Wait State ให้กับส่วนต่าง ๆ ดังตารางที่ 7

โดยที่ค่าของ bit ต่างๆ จะมีผลต่อค่าของ Wait State ดังตารางที่ 8 (ต้องดูค่าที่กำหนดใน PDWSR และ IOWSR ประกอบด้วย)

โดยหลังการ Reset Wait State ทุกๆ ค่าจะกลายเป็น 7 Machine Cycle โดยค่าใน Bit Big จะเป็น 0

N (Bit Position in CWSR)	SPACE
0	Program
1	Data
2	I/o (lower-haif:Port 0-Port 7 if BIG=0,0000h-7FFFh if BIG=1)
3	I/o (lower-haif:Port 8-Port F if BIG=0,8000h-0FFFFh if BIG=1)
4	BIG mode bit

ตารางที่ 7 การใช้งาน Bit ต่างๆ ของ CWSR

Wait-State Field of PDWSR or LOWSR (Binary Value)	No of Wait States (CWSR Bit n=0)	No of Wait states(CWSR Bit n=1)
00	0	0
01	1	1
10	2	3
11	3	7

ตารางที่ 8 ความสัมพันธ์ของค่า Wait State กับค่าใน PDWSR และ IOWSR

16. Register เกี่ยวกับ Interrupt

ในการใช้งานอุปกรณ์ประเภท Microprocessor ลักษณะการทำงานที่สำคัญอย่างหนึ่งคือการ Interrupt เราต้องทราบการทำงานของการ Interrupt เพราะมีความจำเป็นที่ต้องการตั้งแต่ขั้นตอนการออกแบบ Hardware และการเขียนโปรแกรม โดย C5x มี Register ที่ทำการเกี่ยวกับการ Interrupt อยู่หลายตัวซึ่งอยู่ใน Register PMST คือ IPTR ซึ่งได้กล่าวถึงมาแล้วที่เหลืออีก 2 ตัวคือ IFR และ IMR ซึ่งมีรายละเอียดแต่ละตัวดังนี้ (ดูจากรายละเอียดเพิ่มเติมในส่วนของการจัดการ Interrupt)

IFR Register (Interrupt Flag Register) ขนาด 9 bit ใน Data Memory ถือเป็น Memory Map Register ใช้ในการแสดงสถานะของ Maskable Interrupt ทั้ง 9 ตัว โดยจะเป็น 1 เมื่อ

Interrupt นั้นทำงานโดยสามารถสั่ง Clear Interrupt จะทำให้มันเป็น 1 อัตโนมัติและจะ Clear อัตโนมัติเมื่อมีการตอบรับนั้นโดยจะทำการ Clear พร้อม ๆ กับ CPU สร้างสัญญาณ Interrupt Acknowledge โดยทั้งหมดจะเป็น 0 เมื่อทำการ Reset ค่า bit ต่าง ๆ ที่ทำการควบคุมการ Interrupt ใน register นี้จะสามารถแสดงดังนี้

8	7	6	5	4	3	2	1	0
INT4	TXINT	TRINT	XINT	RINT	TINT	INT3	INT2	INT1

IMR (Interrupt Mark Register) ขนาด 9 ใน bit ใน Data Memory ถือเป็น Memory Map Register เหมือนกับ IFR เป็น register ที่จะเป็นตัวกำหนดการใช้งาน Interrupt นั้น โดยสามารถเขียนและอ่านได้เหมือน Data Memory หรือคำสั่งเกี่ยวกับ Memory Map Register เช่น SAMM

17. Register ควยคุมการทำงานของอุปกรณ์ร่วม (Peripheral Interfacing Circuit)

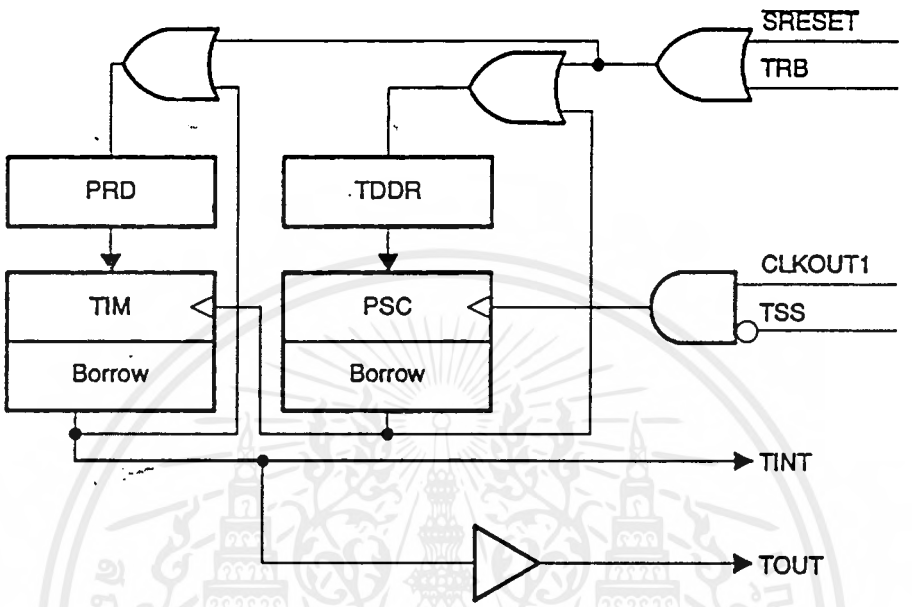
SPC Register (Serial Port Control Register) ขนาด 16 bit ที่ใช้ในการควบคุมการทำงานของ Serial port ที่จะทำการติดต่อกับอุปกรณ์ภายนอกแบบอนุกรมโดย อยู่ในส่วนของ Memory Map Register โดยมีการกำหนดการทำงานต่าง ๆ เหมือนกับ Data Memory และ memory map Register อื่น ๆ

TCR Register(Time Control Register)

ใน c5x มี Hardware Timer ขนาด 16 bit โดยมี 4 บิต pre-scalar สามารถกำหนดการ control ผ่านทาง TCR Register ซึ่งถือเป็น Memory Map Register สามารถทำการเขียนการอ่านและเขียนเหมือนกับ Memory ทั่วไป

โดยที่อัตราการนับจะสามารถกำหนดได้ใน PDR และ PSC ซึ่ง Timer จะทำการนับโดยนับลงทุกๆ Machine Cycle โดยเมื่อนับถึง 0 แล้วจะทำการสร้างสัญญาณ Interrupt ของ Timer ขึ้นมาพร้อมกันกับการเปลี่ยนค่าที่ขา Tout โดยอัตราการหารหรือการ Interrupt จะเป็น

$$Tout = \frac{1}{t_c \times (TDDR+1) \times (PRD+1)}$$



รูปที่ 11 แสดง Block Diagram การทำงานของ Timer ใน C5x

โดยค่า PRD จะสามารถเขียนใน Register นี้โดยตรงแต่ค่าของ TDDR จะทำการเขียนผ่าน Register TCR : ซึ่งรายละเอียดของ Register TCR จะสามารถแสดงได้ดังนี้

รายละเอียดของ Register TCR

15-12	11	10	9-6	5	4	3-0
RESERVED	SOFT	FREE	PSC	TRB	TSS	TDDR

TDDR (Timer Division Down Ratio) เป็นตัวกำหนดอัตราการหาร โดยจะโหลดค่าจาก TDDR ไปเก็บใน PSC แล้วทำการนับลงโดยขณะนับ จะไม่เกี่ยวกับค่าใน TDDR

TSS (Time Slope Status bit) เป็นตัวกำหนดการ start stop ของ timer TSS=1 จะเป็นการ stop timer TSS=0 จะเป็นการ start timer

TRB (Time reset) จะเป็นบิตที่ทำการสั่ง Reset time ซึ่งจะทำการโหลดค่าจาก TDDR ไปเก็บใน PSC แล้วจะทำการนับใหม่ โดยขณะ อ่าน bit จะเป็น 0 ตลอด

PSC จะทำการนับลงมาเรื่อย ๆ ตามการทำงานของ CPU โดยไม่สามารถอ่านได้ต้องทำให้มันหยุดการทำงานก่อนโดยการให้ TSS=1

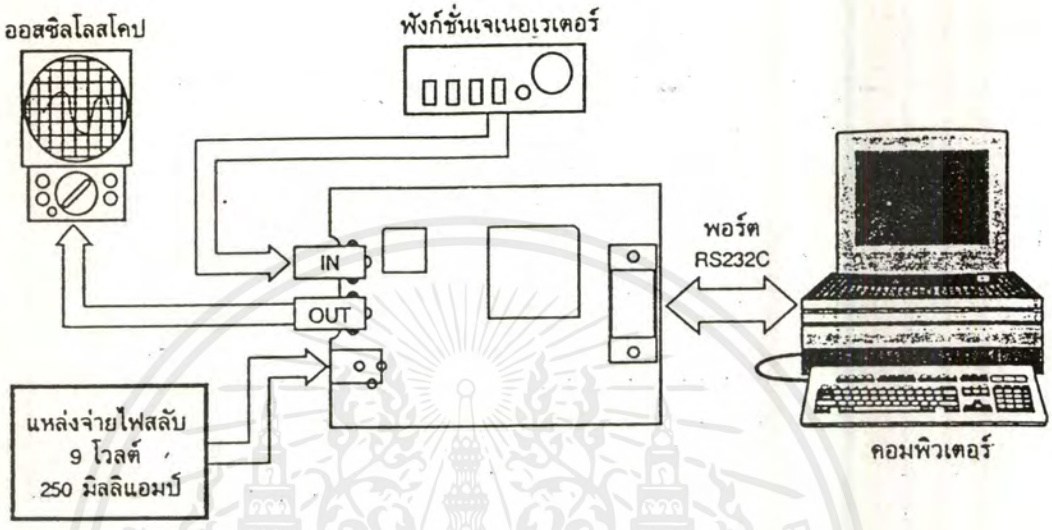
18. TMS320C5x DSK Starter Kit

ส่วนสำคัญในโครงงานนี้ เป็นชุดพัฒนาโปรแกรม สำหรับ ชิพ DSP ตระกูล TMS320C5x วึ่งเป็น ชิพ Digital Processing ตระกูล Fixed-Point ซึ่งนำมารวมกันกับ ชิพ Analog Interface เป็นบอร์ดที่ใช้ในการทดลองเกี่ยวกับ DSP ในชุดทดลองนี้ประกอบด้วย

ส่วนของ Debugger ที่ใช้พัฒนาโปรแกรม โดยทำการแก้ไขผ่านทาง Computer PC (โดยติดต่อกันผ่าน Serial Port ของ PC) ซึ่งสามารถ แก้ไข ดูค่า Register, Run program ที่ละ Step เพื่อดูผลการแปลงของค่าต่าง ๆ

ส่วนที่เป็นส่วนรับส่งสัญญาณจะผ่าน RCA Jack โดยสัญญาณ Analog ทั้งสองขั้วจะถูกจัดการโดย ชิพ Analog Interface เบอร์ TLC 32040 ซึ่งภายในประกอบด้วย Analog to Digital และ Digital to Analog ขนาด 14 ซึ่ง ชิพ สามารถโปรแกรม การทำงานของมันผ่านทาง Register ภายในโดยที่มันติดต่อกับ TMS320C5x แบบ Serial ทั้งรับและส่งข้อมูล ทั้งการโปรแกรมค่าต่างๆ ใน Register

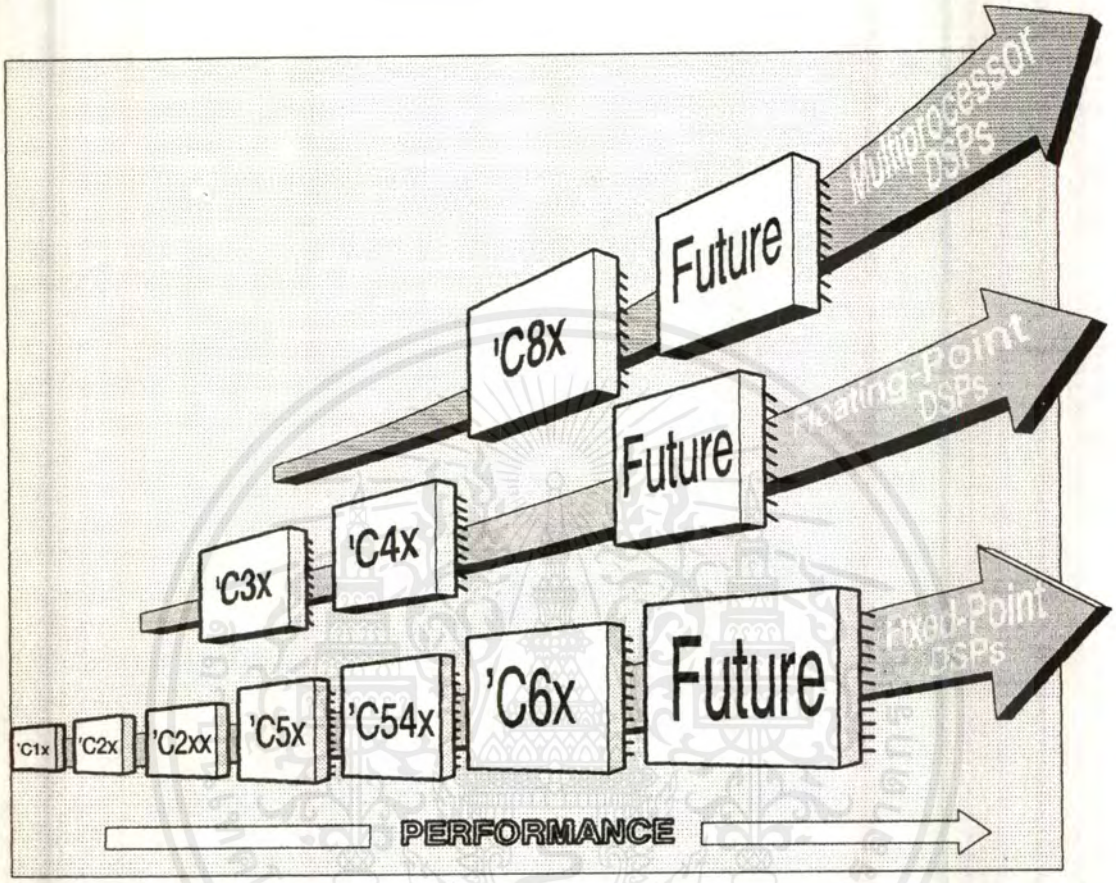
ส่วนที่สำคัญที่สุดคือ ชิพ DSP TM320C50 เป็นส่วนที่นำมาทำการรับค่าเข้ามาเพื่อกระทำการคำนวณแบบต่าง ๆ ตามโปรแกรมโดยภายในจะมีส่วนของ ROM และ RAM ในการเก็บ Program และ Data เพื่อทำการเข้าถึงที่รวดเร็วเพราะในการประมวลผลสัญญาณ แบบ Real Time ต้องใช้ความเร็วของอุปกรณ์สูง ส่วนของการเก็บ Program และ Data (10K ram) ก็มีมากพอสำหรับ Application ขนาดใหญ่ทางด้าน Signal Processing



รูปที่ 12 การต่อใช้งาน Board DSK

ใน Memory Map เป็นส่วนที่สำคัญมากอีกส่วนหนึ่งที่เราจะต้องทราบเกี่ยวกับพื้นที่ที่จะทำการเขียนโปรแกรม,พื้นที่ในการใช้ interrupt และพื้นที่ในการเก็บหน่วยความจำกับส่วนอื่นๆที่จำเป็นในการเขียนโปรแกรม

นอกจากนี้ยังมีความสะดวกสบายในการพัฒนาโปรแกรม เพราะเป็นชุดทดลองสำหรับการเริ่มต้นการเรียนรู้การทำงานของ DSP บริษัทผลิต ชิพ มาสำหรับผู้เริ่มต้นการศึกษาและยังมี Program ตัวอย่าง ซึ่งเป็น Program เกี่ยวกับการนำไปใช้งานด้านต่าง ๆ (สามารถ Download ได้เพิ่มเติมที่ WWW.TI.COM) รวมทั้งมีส่วนของการรับรองการขยาย Hardware ซึ่งสามารถทำได้สะดวก



รูปที่ 13 Evolution of the TMS320 Family

หมายเหตุ ในบทนี้ไม่ได้กล่าวถึงรายละเอียดของวงจรต่าง ๆ ของบอร์ด และการทำงานของ ชิพ Analog Interface โดยสามารถศึกษาข้อมูลเพิ่มเติมได้จาก User's guide ของ TMS320C5x Starter kit ส่วนรายละเอียดเพิ่มเติมที่ไม่ได้กล่าวถึงของ C50 สามารถศึกษาได้จาก TMS320C5x User's Guide

บทที่ 4
การเขียนโปรแกรม บน Chip DSP

จากบทที่ผ่านมาเราได้ทำการทดลอง Simulation ดูผลการทดลองโดยใช้โปรแกรม Matlab ส่วนหนึ่งในบทนี้เราจะทำการเขียนโปรแกรมลงบน Chip DSP เพื่อดูผลการตอบสนองต่อสัญญาณจริงๆ เทียบกับ ผลการทดลองที่ใช้ Computer ในการ Simulation ว่าเป็นอย่างไร

1. Frequency Response of Analog Interface Chip

ก่อนที่จะทำการเขียนโปรแกรม Chip เพื่อให้ทำงานเป็น Filter และทำการวิเคราะห์การตอบสนองในรูปแบบต่างๆ เราต้องทราบว่าที่ภาคสุดท้ายในส่วนของ Digital to Analog ใน IC Analog Interface (TLC 32040CFN) จะมี Low-Pass Filter (Reconstruction Filter) อยู่ทำให้การวิเคราะห์การตอบสนองความถี่ของ Filter ที่เราจะโปรแกรมลงบน Chip DSP เกิดการผิดพลาดได้ฉะนั้นก่อนอื่นเราจะต้องทำการวิเคราะห์การตอบสนองความถี่ของส่วนนี้เสียก่อน โดยทำการเขียนโปรแกรมรับค่าเข้ามาแล้วทำการส่งออกไปเลยโดยไม่ต้องทำการคำนวณใดๆ เลยโดยโปรแกรมสามารถแสดงได้ดังโปรแกรมที่ 1

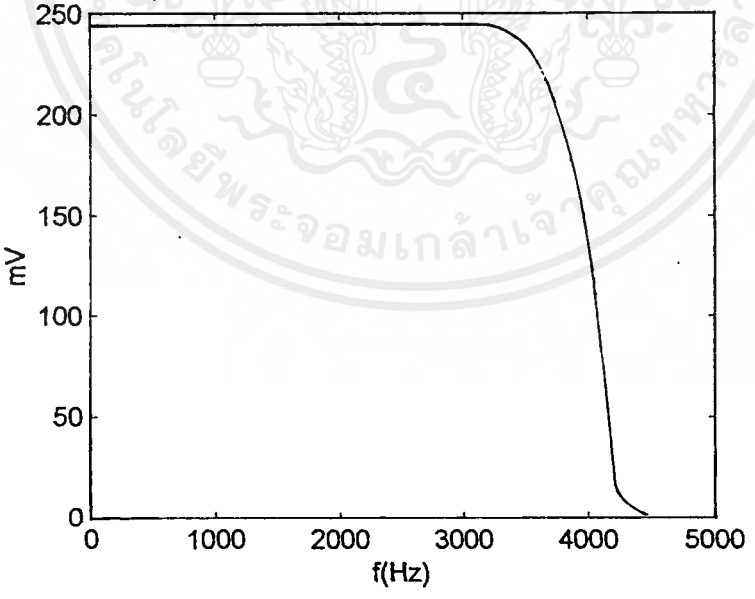
โดยความถี่ Sampling ที่ระบุและเปลี่ยนแปลงในการทดลองนี้จะไม่มีผลกับ Low-Pass Filter เพราะเราจะทำการเปลี่ยนแปลงเฉพาะค่าใน Register TB,RB ซึ่งจะไม่มีผลกับการตอบสนองความถี่ของ Low-Pass Filter (Switch Capacitor Filter) เพราะ Clock ที่ใช้ในส่วนนี้จะมาจากการหาร Master Clock กับค่าใน Register TA,RA (ดูรายละเอียดการทำงานของ Chip Analog Interface จาก DSK Starter Kit User Guide) จากนั้นทำการรันโปรแกรมข้างต้นสามารถเก็บค่าการตอบสนองที่ความถี่ต่างๆ ได้ดังนี้ โดย Amplitude ของ Input เท่ากับ 1.00 โวลท์

Frequency	Volt Output
500 Hz	245 mV
1.00 Hz	245 mV
1.50 Hz	245 mV
2.00 Hz	245 mV

2.50 Hz	245 mV
3.00 Hz	245 mV
3.40 Hz	233 mV
3.60 Hz	330 mV
3.82 Hz	170 mV
3.90 Hz	96.3 mV
4.00 Hz	92.3 mV
4.20 Hz	11.7 mV
4.50 Hz	2.44 mV

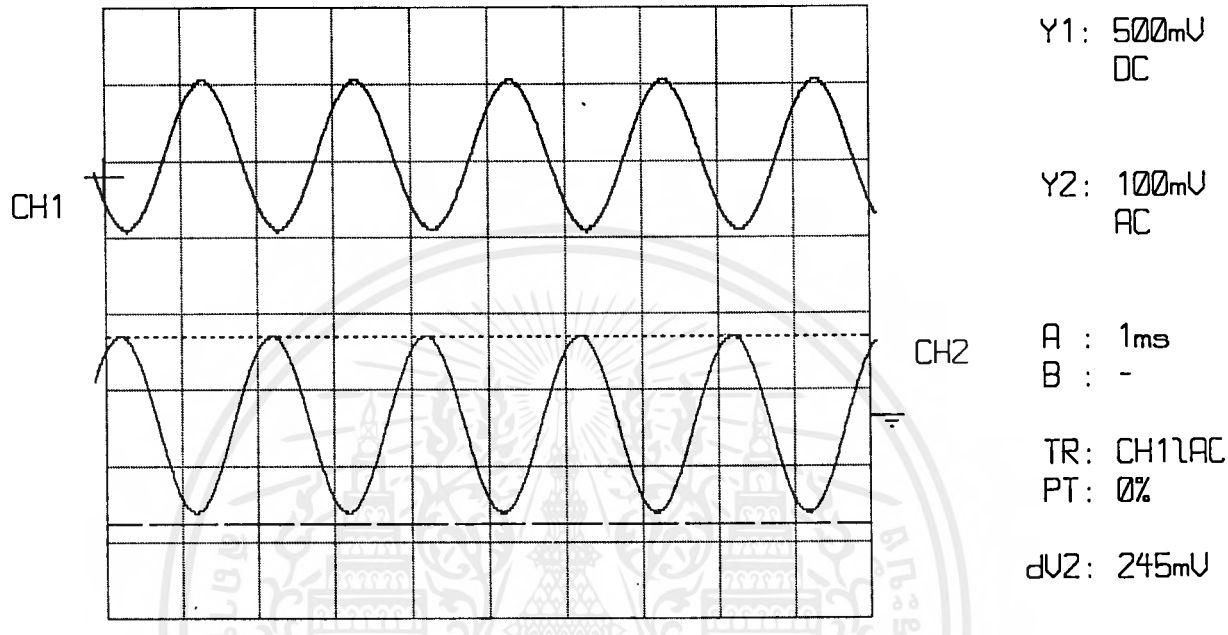
ตารางที่ 1 การตอบสนองความถี่

ซึ่งสามารถ Plot Graph ได้ดังนี้

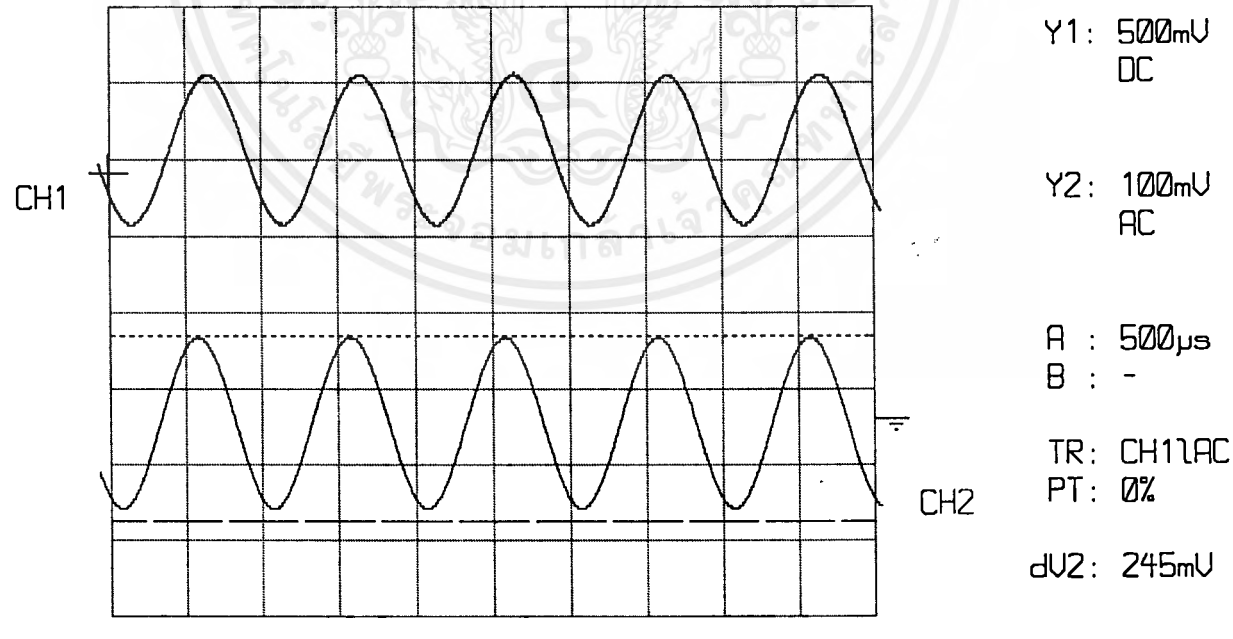


รูปที่ 1 กราฟแสดงการตอบสนองความถี่

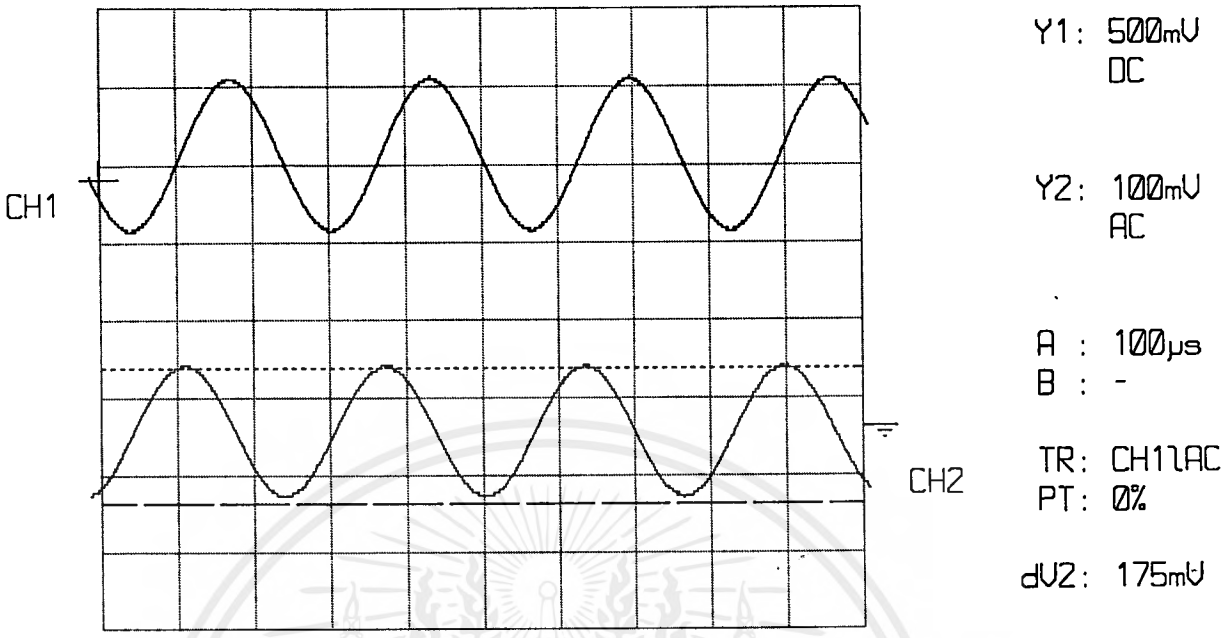
จากกราฟจะเห็นว่าความถี่ Cutoff ของ Low-Pass Filter จะอยู่ที่ 3.82 KHz ซึ่งเวลาทดลองจะต้องอยู่ในช่วงไม่เกินความถี่นี้ อีกอย่างคือการตอบสนองเชิง Phase ของมัน ซึ่งสามารถแสดงโดยรูปของการตอบสนองที่ความถี่ต่างๆ



รูปที่ 2 แสดงสัญญาณ Input และ Output ที่ความถี่ 500 Hz



รูปที่ 3 แสดงสัญญาณ Input และ Output ที่ความถี่ 1 KHz



รูปที่ 4 แสดงสัญญาณ Input และ Output ที่ความถี่ 3.82 KHz ที่เป็นจุด Cutoff

จากรูปแสดงถึงการตอบสนองทาง Amplitude และ Phase จะเห็นว่ามีการ Shift Phase ไปของแรงดัน Output โดยจากการทดลองเปลี่ยนค่าความถี่พบว่าจะมีการ Shift Phase ไปประมาณ 90 องศาที่ความถี่ประมาณทุกๆ 600 Hz

จากข้อมูลข้างต้นจะทำให้สามารถวิเคราะห์ผลการตอบสนองแบบต่างๆ ของ Filter ที่เราจะทำการทดลองได้ถูกต้องขึ้นทั้งทาง Amplitude และ Phase

2. IIR Band-Pass Filter

ในขั้นตอนนี้จะนำเอาสมการของ Band-Pass IIR Filter มาทำการเขียนโปรแกรมเพื่อดูผลการตอบสนองของ Transfer Function ที่เราทำการ Simulate ตอนแรก โดยดูผลการตอบสนองว่าจะเหมือนที่ทำการ Simulate หรือไม่

จากสมการที่ (6) ในบทที่ 2 Time Domain ของ IIR Band Pass Filter

$$y(k) = \left(\frac{1-\alpha_0}{2}\right)[x(k) - x(k-2)] + \alpha_1(1+\alpha_0).y(k-1) - \alpha_0 y(k-2)$$

โดยเราจะกำหนดความถี่การ Sampling เท่ากับ 15.625 KHz (ซึ่งเรากำหนดได้จาก การกำหนดจาก ค่า TA ,RA ,TB , RB ในโปรแกรม) โดยกำหนดความถี่กลางของ Filter = $\pi / 20$ ทำให้ได้ความถี่กลางจริงๆ ใน Time Domin ได้จากสมการ

ซึ่งสามารถหาความถี่ Sampling ได้จากสูตร

$$f_s = \frac{f_{MCLK}}{2 \times TA \times TB} \qquad (1)$$

โดย fs คือ ความถี่ในการ Sampling

f_{MCLK} คือ AIC Master Clock มีค่า 10.368 MHz AIC Shift Clock มีค่า 2.592 MHz หรือ Instruction Cycle 7.72 MHz Master Clock มีค่า 20 MHz หรือ Instruction Cycles 20 MHz

TA คือ จำนวนเต็มที่มีค่าระหว่าง $4 < TA < 31$

TB คือ จำนวนเต็มที่มีค่าระหว่าง $2 < TB < 63$

เพราะฉะนั้นเราทำการแทนค่าจากสูตร

$$f_s = \frac{f_{MCLK}}{2 \times TA \times TB}$$

$$f_s = \frac{10MHz}{2 \times 16 \times 20}$$

$$f_s = 15.625\ KHz$$

และทำให้เราได้ ความถี่กลางใน Time Domain ซึ่งสามารถหาได้จากสมการ

$$T = \frac{\omega}{\Omega} \qquad (2)$$

เมื่อ

$$\omega = \frac{2\pi}{N} \quad \text{radian/sample} \quad \text{(ความถี่ในทาง Digital)} \qquad (3)$$

และ

$$\Omega_0 = 2\pi f_0 = \frac{2\pi}{T_0} \quad \text{radian/sec} \quad \text{(ความถี่ในทาง Analog)} \qquad (4)$$

โดยที่ T คือ ช่วงเวลาของการ Sampling แต่ละครั้ง

N คือ จำนวน Sample

ω คือ จำนวน Sample ต่อรอบ

Ω คือ จำนวนรอบต่อหน่วยเวลา

แทนค่าตัวแปรในสมการจะได้

$$T = \frac{\omega}{\Omega}$$

$$= \frac{\omega}{\Omega_0}$$

$$T = \frac{2\pi / N}{2\pi / T_0}$$

$$\frac{1}{T_0} = \frac{2\pi / N}{2\pi / T}$$

$$\therefore \quad f_0 = \frac{f_s}{N}$$

กำหนดให้ N=40 แทนค่าในสมการจะได้

$$f_0 = \frac{f_s}{N}$$

$$= \frac{15.625KHz}{40}$$

เพราะฉะนั้น $f_0 = 390.625\ Hz$

ดังนั้นความถี่กลางใน Time Domain มีค่าเท่ากับ 390.625 Hz เพราะฉะนั้นเราจะได้ความถี่กลาง (ในทาง Digital) เท่ากับ $\frac{\pi}{20}$ และเราก็สามารถกำหนดให้

$$\alpha_1 = \cos(\omega_0)$$
$$= \cos(\frac{\pi}{20})$$
$$= 0.9876$$
$$\alpha_0 = 0.92$$

จากสมการของ Time Domain ของ IIR Band-Pass Filter เราทำการแทนค่าต่างๆได้ดังนี้

$$y(k) = 0.04x(k) - 0.04x(k-2) + 1.896y(k-1) - 0.92y(k-2) \qquad (5)$$

1. การแปลงค่าจำนวนทศนิยมมาทำการคำนวณในรูปจำนวนเต็ม

ค่าคงที่ต่างๆในแต่ละเทอมต้องทำการแปลงก่อนที่จะทำการเขียนลง Chip เพราะค่านี้จะเป็นค่าที่อยู่ในรูปของจำนวนทศนิยม (Floating-Point) ส่วนใน Chip ตระกูล C50 จะสามารถคำนวณในรูปจำนวนเต็ม (Fix-Point) ได้เท่านั้นโดยในการคำนวณนั้นจะต้องทำการนำค่าเหล่านี้มาทำการคูณกับค่า A/D แปลงมาซึ่งจะอยู่ในรูปของจำนวนเต็มเช่นกันแต่ระดับแรงดันใน Domain ของ Analog นั้นจะเป็นค่าทศนิยมเช่นกัน ซึ่งในการคูณกันนั้นผลออกมาจะไม่ถูกต้องตามความจริง เพราะการคูณค่าทศนิยมกับจำนวนใดๆ นั้นจะทำให้คำตอบที่ได้มีค่าน้อยลง เพราะฉะนั้นการทำการเก็บค่าคำตอบของการคูณนั้นต้องทำการ Scale ค่าให้อยู่ในค่าที่ถูกต้องเสียก่อน ซึ่งการ Scale ค่าของการคำนวณในรูป Binary คือ การทำการ Shift นั่นเองโดยการ Shift ขวาไปหนึ่ง Bit หมายถึงการหารสองในฐาน 10 ส่วนการ Shift ซ้ายไปหนึ่ง Bit คือการคูณด้วยสองในฐาน 10 โดยในการ Shift ซ้ายหรือการหารนั้นจะเป็นค่าที่ Bit LSB ไม่นำมาคิดเพราะการ Shift ซ้ายนั้นจะตัด LSB ทิ้ง หรือถ้าจะมองในอีกแง่หนึ่งคือการหารด้วย Shift นั้นจะได้ค่าที่ถูกต้องก็ต่อเมื่อ จำนวน Bit ทางด้าน LSB นั้นจะต้องเป็น 0 เท่ากับจำนวน Bit ที่ทำการ Shift ซึ่งใน Shift DSP โดยทั่วๆ ไปนั้นจะมี Shifter ที่ทำหน้าที่ในส่วนนี้อยู่แล้ว ซึ่งส่วนมากจะทำในขั้นตอนของการเก็บค่าและโหลดค่า

โดยในการแปลงค่าสัมประสิทธิ์ของ Filter ในที่นี้จะทำการแปลงโดยอ้างอิงกับการแปลงแรงดันของ A/D มาอยู่ในรูป Binary ที่จะทำการคำนวณเพราะจะเป็นการง่ายในการ Scale ค่าหลัง

การคำนวณโดยจะอ้างอิงกับแรงดันอ้างอิงในการแปลงค่าของ A/D ซึ่งก็คือ 2 Volt แต่ในการแปลงนั้นจะมีทั้งส่วนของบวกและลบโดยการแปลงส่วนของแต่ละค่าจะเป็น 14 Bit ซึ่งจะทำให้ได้รายละเอียดเท่ากับ $2^{14} = 16384$ ระดับ ซึ่งค่านี้จะเท่ากับ 1 ในโดเมนของแรงดัน

ซึ่งทำให้สามารถคำนวณค่าคงที่ต่างๆ ได้ดังนี้

$$0.04 \times 16384 = 655.36$$

$$1.896 \times 16384 = 31064$$

$$0.92 \times 16384 = 15073$$

สามารถเขียนสมการที่ (5) กับค่าคงที่ในหน่วย ของค่าที่ทำการ Scale แล้วได้ดังนี้

$$y(k) = 655x(k) - 655x(k-2) + 31064y(k-1) - 15073y(k-2) \qquad (6)$$

โดยสามารถเขียนโปรแกรมเพื่อให้ Chip DSP ทำการคำนวณหาแรงดัน Output แต่ละขณะเวลาเพื่อส่งออกไปที่ Digital to Analog เพื่อทำการแปลงกลับเป็น Analog สามารถเขียนโปรแกรมได้ดังโปรแกรมที่ 2

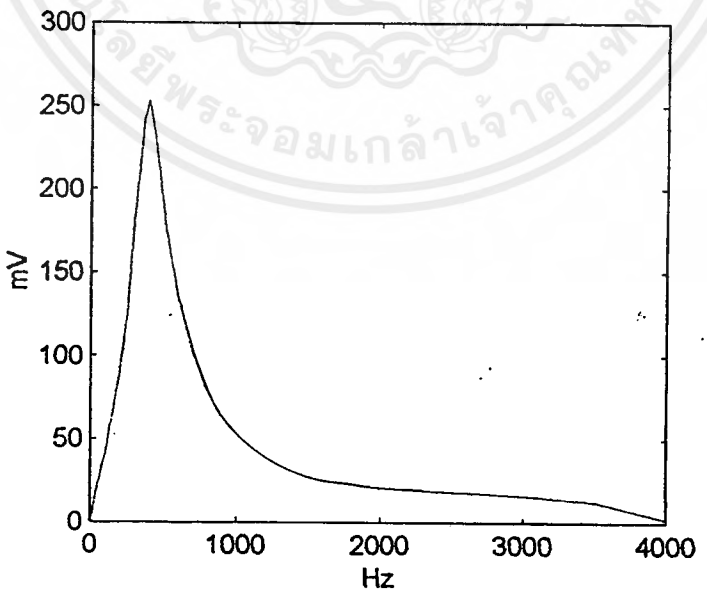
Frequency	Volt Output
50 Hz	22.4 mV
100 Hz	40.1 mV
200 Hz	91.1 mV
250 Hz	125 mV
300 Hz	183 mV
330 Hz	213 mV
360 Hz	241 mV
390 Hz	253 mV
420 Hz	241 mV
450 Hz	222 mV
515 Hz	176 mV

600 Hz	134 mV
650 Hz	119 mV
700 Hz	96.6 mV
800 Hz	81.4 mV
1 KHz	61.0 mV
2 KHz	26.2 mV
3 KHz	16.2 mV
3.5 KHz	12.5 mV
4 KHz	2.35 mV

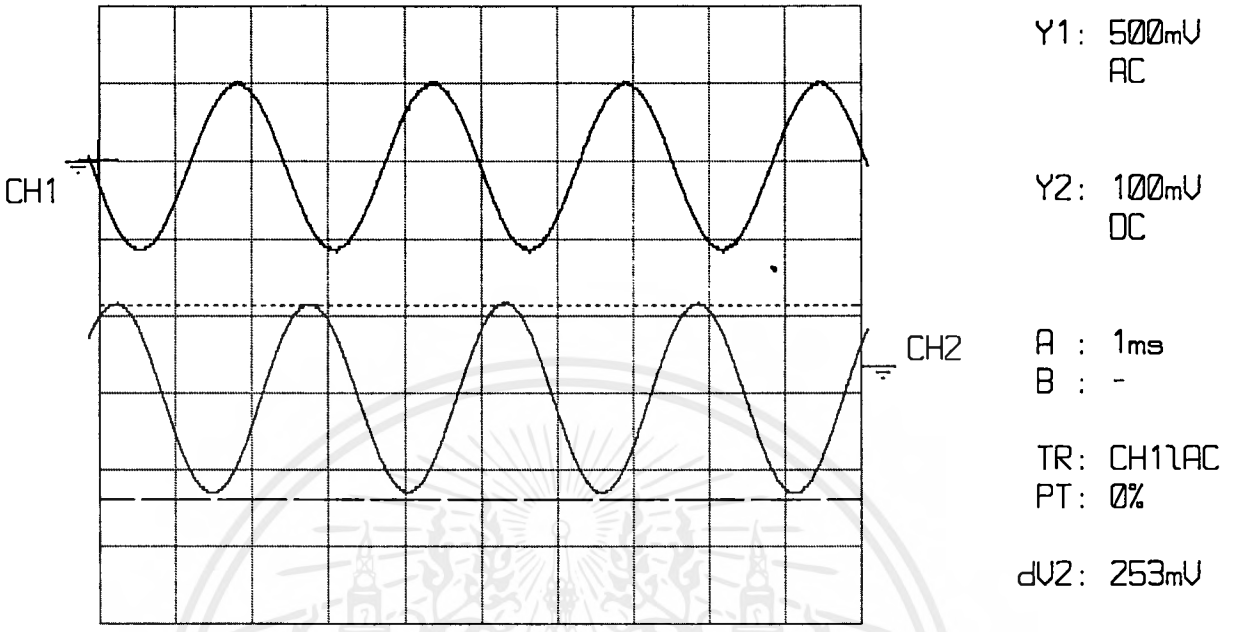
ตารางที่ 2 แสดงการตอบสนองความถี่ของ IIR Band-Pass Filter

จากการรันโปรแกรมข้างต้นแล้วทำการป้อนสัญญาณ Input ความถี่ต่างๆ แล้วทำการวัดค่าการตอบสนองทาง Amplitude ที่ความถี่ต่างๆ ดังตาราง

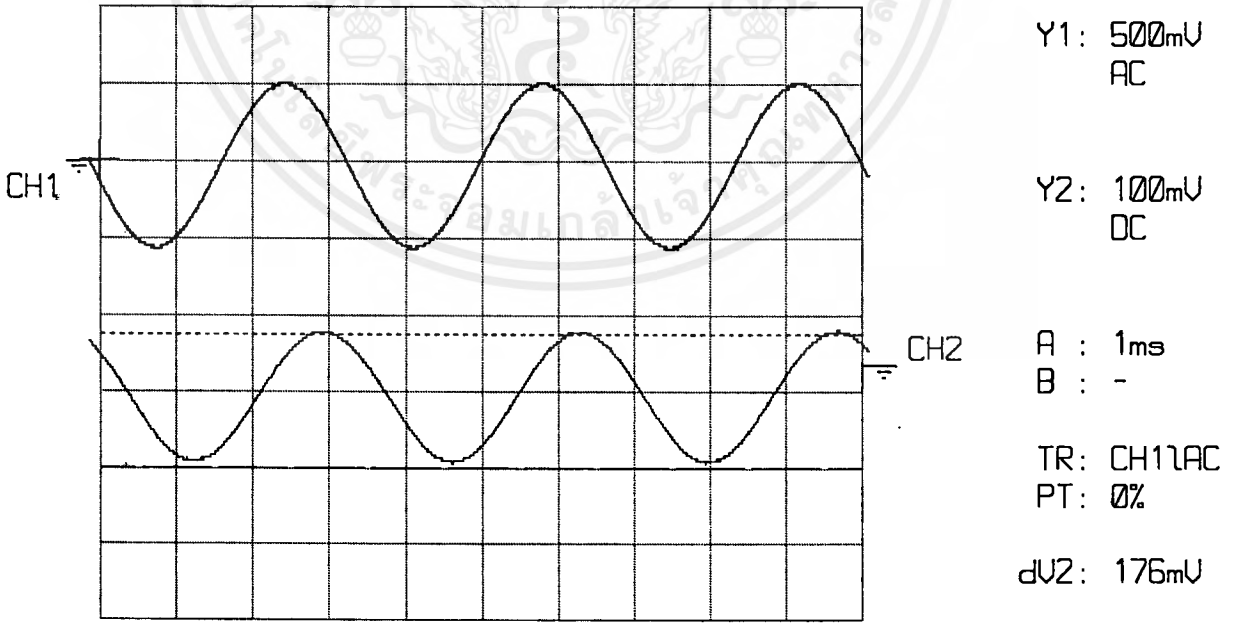
ซึ่งสามารถนำมา Plot Graph ได้ และจากกราฟจะเห็นว่าการตอบสนองความถี่เป็นไปตามที่คำนวณไว้คือค่า Amplitude ที่ความถี่ 390 Hz มีค่ามากที่สุด โดยรูปคลื่นที่ความถี่กลางและความถี่ Cutoff ทั้งสองด้านซึ่งสามารถนำมา Plot Graph ได้ดังนี้



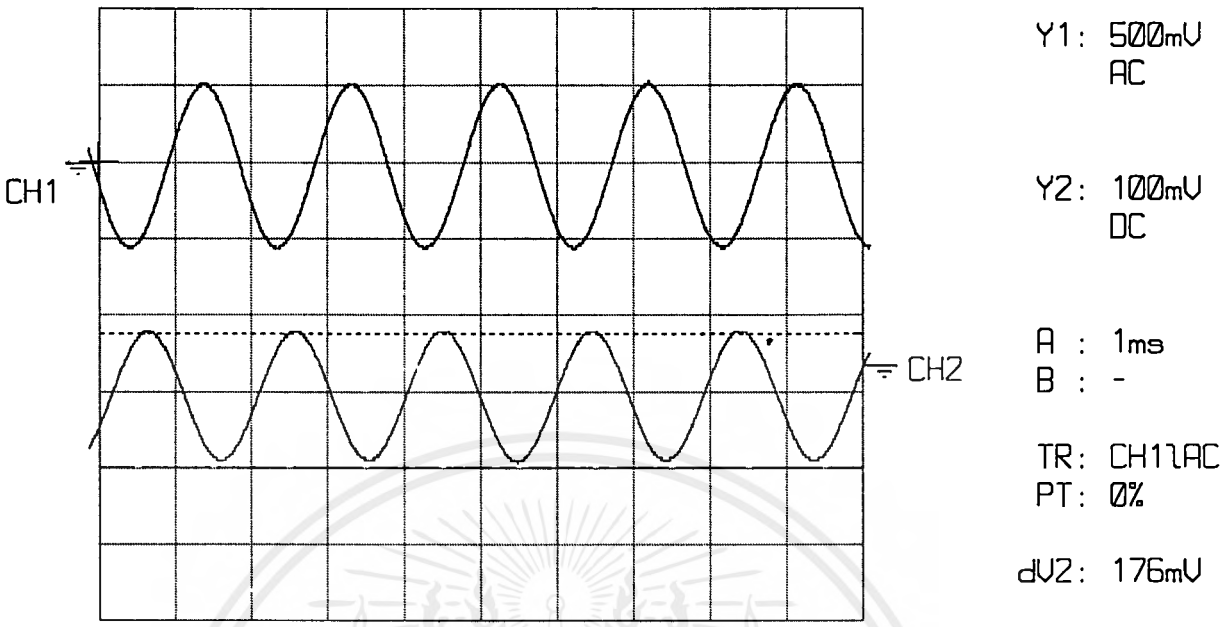
รูปที่ 5 กราฟแสดงการตอบสนองความถี่ของ IIR Band-Pass Filter



รูปที่ 6 การตอบสนองที่ความถี่กลางของ Band-Pass IIR Filterที่ความถี่ 390 Hz



รูปที่ 7 การตอบสนองที่ความถี่ Cutoff ด้านต่ำ ที่ความถี่ 296.08 Hz



รูปที่ 8 การตอบสนองที่ความถี่ Cutoff ด้านสูง ที่ความถี่ 515.48 Hz

3. Adaptive Digital IIR Filter โดยวิธีนอร์มอลไลซ์เกรเดียนซ์

ส่วนของ Adaptive Digital IIR Filter โดยวิธีนอร์มอลไลซ์เกรเดียนซ์จะทำการเขียนโปรแกรมเพื่อดูผลการตอบสนองต่อการเปลี่ยนแปลงความถี่ของสัญญาณ Input (การเปลี่ยนค่า $\alpha_1(k)$) ว่าจะตามสัญญาณ Input อย่างไรรวมทั้งการแยกสัญญาณ Sine wave ที่เป็นความถี่หลักออกจากสัญญาณรบกวน

1. ส่วนการโปรแกรม

ส่วนของการโปรแกรมจะเหมือนกับของ IIR Filter ในส่วนแรก แต่จะมีส่วนของการUpdate ค่าความถี่กลางของ Filter ในกรณีที่สัญญาณ Input มีการเปลี่ยนแปลงความถี่เพิ่มขึ้นมา คือการคำนวณในส่วนของ และ $\psi(k)$ และ $\beta(k)$

โดยจากสมการทาง Time Domain ของ Digital IIR Filter และสมการการ Update ค่าความถี่กลางที่ได้ทำการ Simulate ดูการตอบสนองค่าต่างๆ กับ Matlab มาแล้วในตอนแรกคือ

$$y(k)=\left(\frac{1-\alpha_0}{2}\right)[x(k)-x(k-2)]+\alpha_1(1+\alpha_0).y(k-1)-\alpha_0 y(k-2)$$

โดยจากสมการทาง Time Domain ของ Digital IIR Filter และสมการการ Update ค่าความถี่กลางที่ได้ทำการ Simulate ดูการตอบสนองค่าต่างๆ กับ Matlab มาแล้วในตอนแรกคือ

$$y(k)=\left(\frac{1-\alpha_0}{2}\right)[x(k)-x(k-2)]+\alpha_1(1+\alpha_0).y(k-1)-\alpha_0 y(k-2)$$

$$\alpha_1(k+1)=\alpha_1(k)+\frac{\mu}{\beta(k)}y(k)\psi(k)$$

$$\beta(k)=\gamma\beta(k-1)+(1-\gamma)[y(k).\psi(k)]^2$$

$$\psi(k)=(1+\alpha_0).\alpha_1(k).\psi(k-1)+(1+\alpha_0).y(k-1)-\alpha_0\psi(k-2)$$

ซึ่งสมการด้านบนในขั้นตอนนี้เราจะนำเอาสมการไปทำการเขียนโปรแกรมลงบน Chip DSP เพื่อทำการแยกสัญญาณ Sin wave ที่มี Noise โดยเราสามารถหาอัลกอริทึมของสัญญาณต่อสัญญาณรบกวนของ Noise แบบ Gaussian ในเทอมของกำลังงานของสัญญาณสามารถกำหนดได้จาก

$$SNR=10\log\left[\frac{v^2}{2\sigma^2}\right]$$

เมื่อ V = Amplitude ของสัญญาณ Input

σ = Standard deviation ของ Gaussian Noise

ต้องการขนาดของสัญญาณรบกวน กำหนดให้ SNR = 10 dB , กำหนดให้ Amplitude ของสัญญาณ Input = 1 V

$$\sigma=\frac{10^{\left[\log v-\frac{SNR}{20}\right]}}{\sqrt{2}}$$

$$\sigma=\frac{10^{\left[\log 1-\frac{10}{20}\right]}}{\sqrt{2}}$$

$$= 0.2236$$

ซึ่งเราสามารถนำไปกำหนดสัมประสิทธิ์ของสัญญาณรบกวนใน การจ่าย Input เข้าไป ซึ่งค่าที่เรากำหนดคือ SNR = 10 dB เหมือนกับที่ทำการ Simulate มารวมทั้งดูการเปลี่ยนแปลงความถี่กลางตามความถี่ Input ว่ามีการติดตามสัญญาณที่ความถี่ต่างๆ อย่างไรบ้างพร้อมทั้งเปรียบเทียบผลจากการทดลองของ Adaptive Digital IIR Filter กับ Adaptive Digital IIR Filter โดยวิธีการนอร์มอลไลซ์เกรเดียนซ์

การกำหนดค่าต่างๆ

ก่อนอื่นเรามาดูความสัมพันธ์ระหว่าง ความถี่ใน Time Domain และ Frequency Domain (Z-Domain) ก่อนซึ่งทำให้พิจารณาผลตอบสนองเชิงความถี่ได้อย่างถูกต้องโดยใน Transfer Function ที่นำมาทำการ Simulate นี้จะมีความสัมพันธ์ทางความถี่เหมือนกับการแปลงแบบ Impulse Invarious ซึ่งขึ้นอยู่กับคาบเวลาการ Sampling ตามสมการต่อไปนี้

$$T = \frac{\omega}{\Omega}$$

โดย ω = ความถี่ใน Frequency Domain (Z-Domain)

Ω = ความถี่ใน Time Domain ($2\pi f$)

T = คาบเวลาการ Sampling

โดยที่ความสัมพันธ์ดังกล่าวจะใช้ได้ดีในขอบเขตที่ความถี่ใน Time Domain ที่น้อยกว่าครึ่งหนึ่งของความถี่ Sampling มิฉะนั้นจะทำให้การตอบสนองความถี่ไม่เป็นไปตามเงื่อนไขที่ออกแบบไว้ คือจะเกิดการซ้อนทับของความถี่ (Aliasing)

ในขั้นตอนนี้เราจะกำหนดค่าความถี่ของสัญญาณ sampling เท่ากับ 10.08 KHz ค่า Q-Factor (α_0) เท่ากับ 0.7 (เหตุที่กำหนดมากไม่ได้เพราะจะทำให้การติดตามเปลี่ยนแปลงความถี่ของสัญญาณ Input ทำได้ช้าแต่ถ้ากำหนดค่า Q น้อยเกินไปก็จะทำให้ความถี่ที่เราไม่ต้องการออกมามาก) ส่วนค่าของความถี่กลาง $\alpha_1(k)$ จะเริ่มต้นที่ค่า 0 คือค่าความถี่ที่ครึ่งหนึ่งของ Nyquist Rate (ทำให้ได้ความถี่ใน Time Domain ประมาณเท่ากับ 2.5 KHz) ส่วนค่า Step Size Parameter (μ) กำหนดไว้ที่ 0.0003 และค่า Forgetting Factor (γ) กำหนดไว้ที่ 0.98 ดังนั้นสามารถแทนสมการได้เป็น

$$y(k) = 0.15[x(k) - x(k-2)] + 0.y(k-1) - 0.7.y(k-2)$$

$$\alpha_1(k+1) = \alpha_1(k) + \frac{0.003}{\beta(k)} y(k)\psi(k)$$

$$\beta(k) = 0.98\beta(k-1) + 0.02.[y(k).\psi(k)]^2$$

$$\psi(k) = 0.\psi(k-1) + 1.7.y(k-1) - 0.7.\psi(k-2)$$

จะเห็นว่ามีเทอมที่เป็นศูนย์คือเทอมที่มีการคูณกับ α_1 แต่พอมันทำการ Update ค่า α_1 เพื่อติดตามสัญญาณ Input มันก็จะมีค่าขึ้นมาเอง

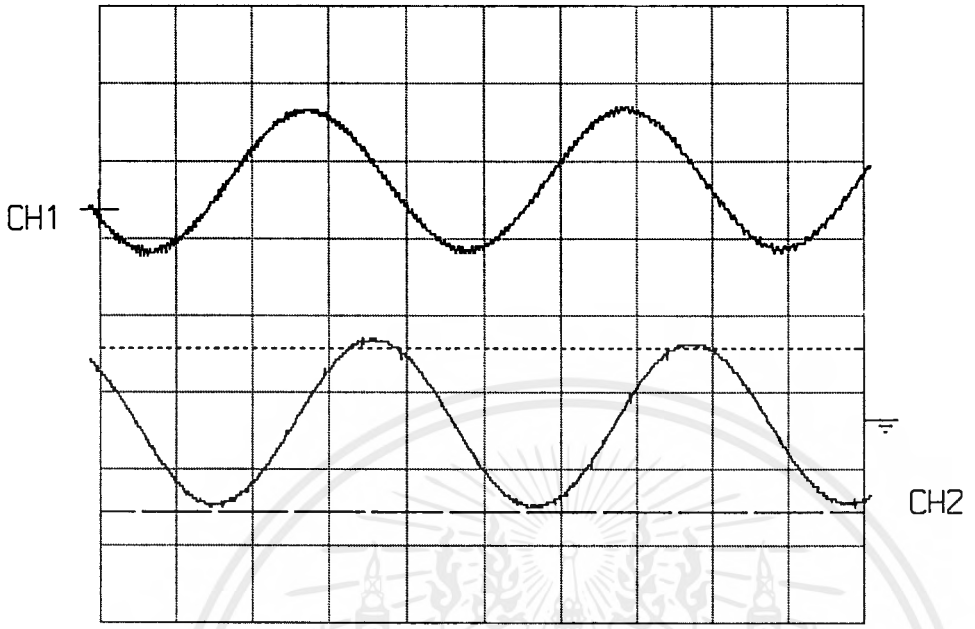
ซึ่งสามารถคำนวณค่าคงที่ต่างๆ โดยอ้างอิง A/D และ D/A Converter ขนาด 14 Bit หลังจากนั้นเราจะทำการเขียนโปรแกรมเพื่อดูผลการทดลองดังโปรแกรมที่ 3

2. ผลการตอบสนองสัญญาณในลักษณะต่างๆ ของAdaptive Digital IIR Filter โดยวิธีนอร์มอลไลซ์เกรเดียนซ์

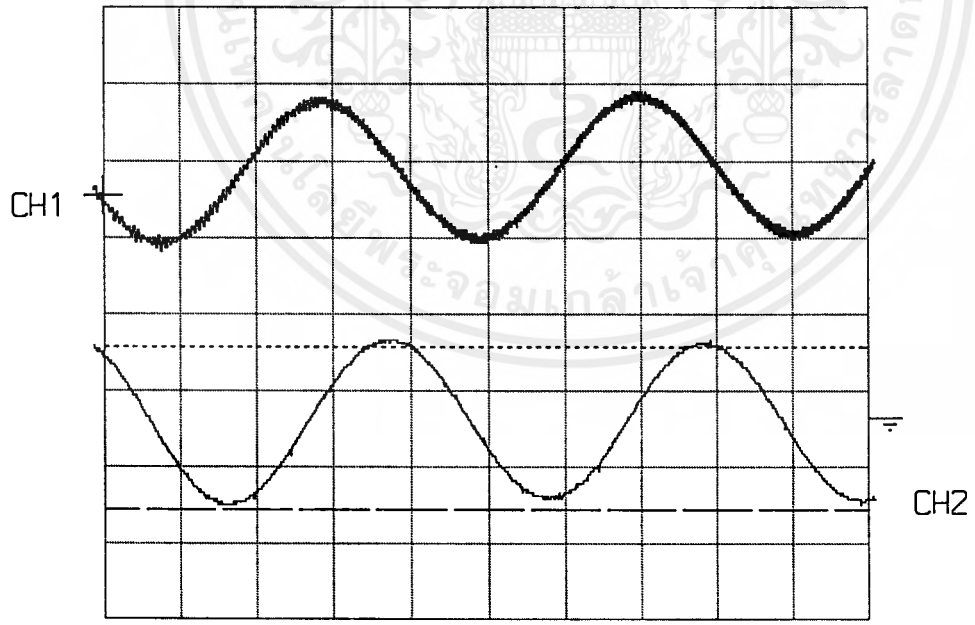
ในขั้นนี้เราจะทดลองโดยการนำเอาสัญญาณรบกวนที่เป็นแบบ Gaussian Noise มารวมกับสัญญาณ Sine Wave (ความถี่หลัก)

ส่วนการพิจารณาทาง Output จะดูลักษณะการขจัดสัญญาณรบกวนแบบต่างๆ และที่สำคัญคือการดูการติดตามการเปลี่ยนแปลงความถี่ของสัญญาณ Input การรักษาเสถียรภาพของการเปลี่ยนแปลงการติดตามความถี่กลาง

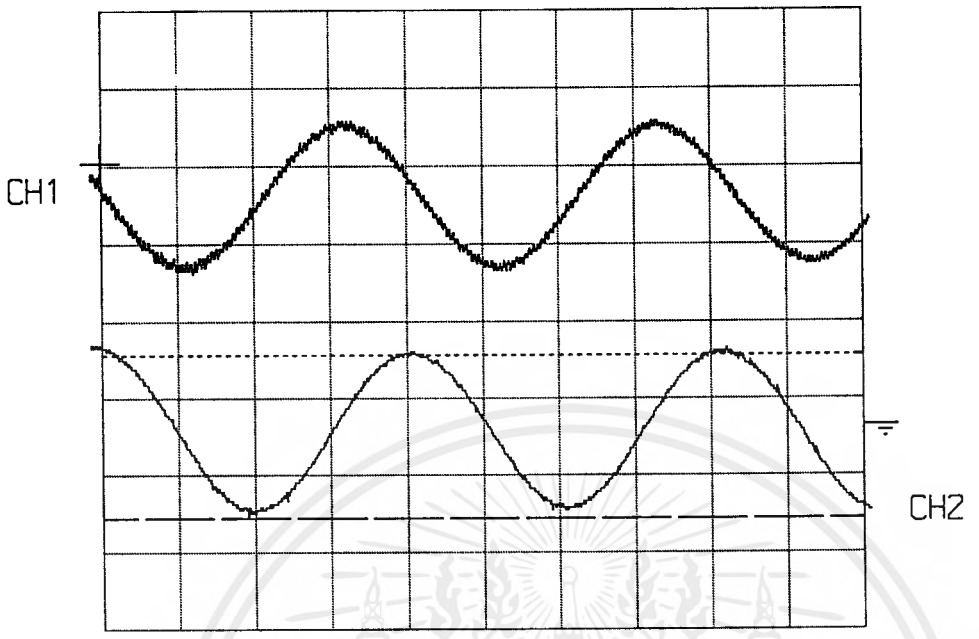
สามารถบันทึกผลการทดลองได้ดังนี้



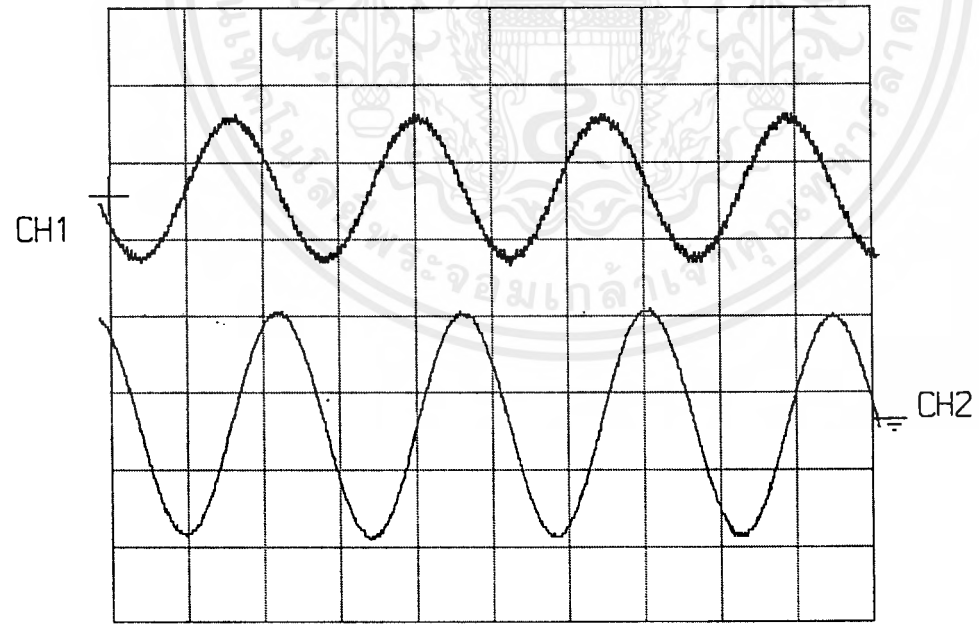
รูปที่ 9 เมื่อสัญญาณ Input เป็น sin 1.21K + 0.07sin 50K , SNR = 20 dB



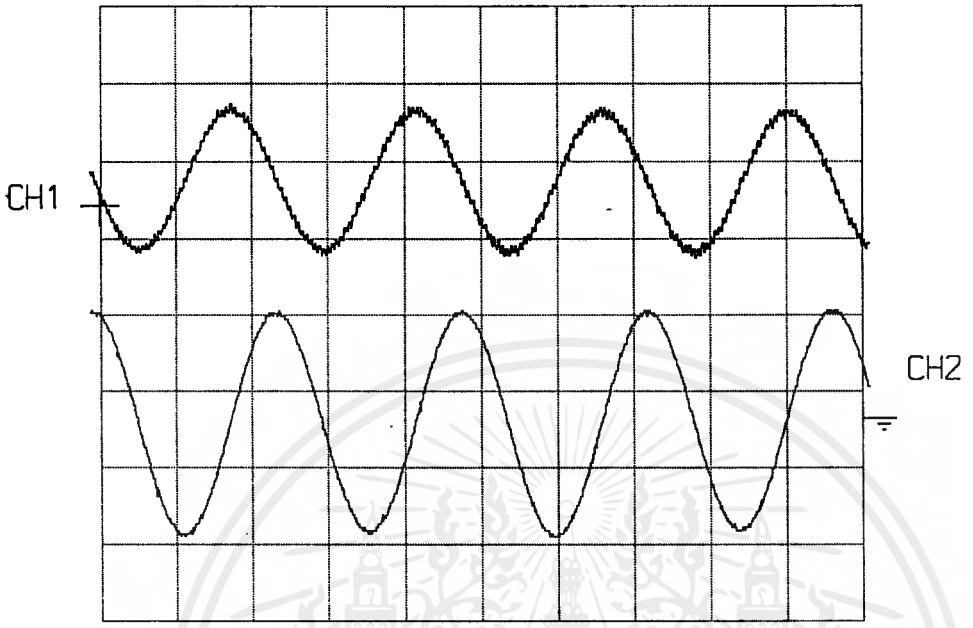
รูปที่ 10 เมื่อสัญญาณ Input เป็น sin 1.21K + 0.22sin 50K , SNR = 10 dB



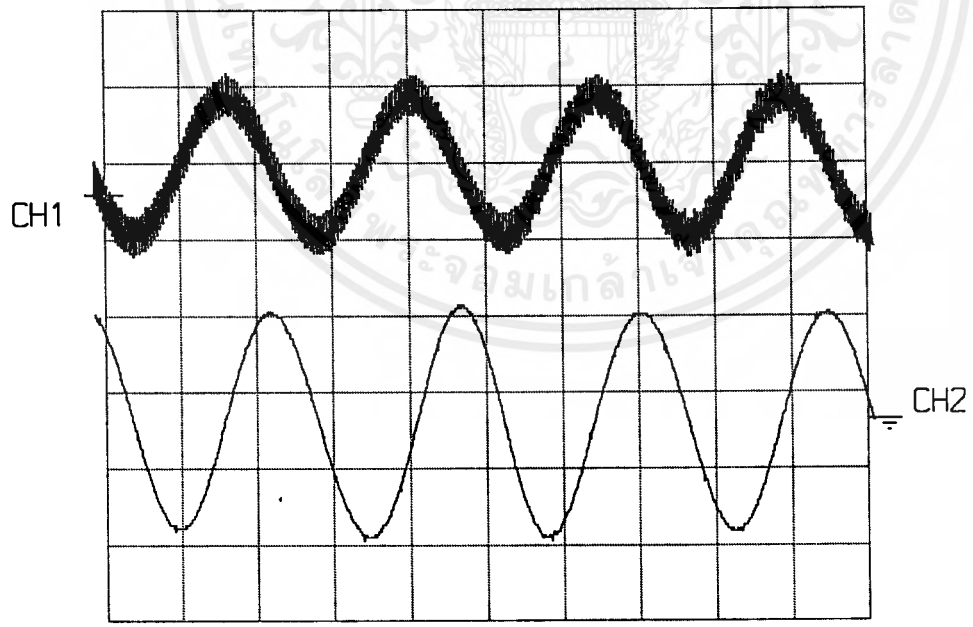
รูปที่ 11 เมื่อสัญญาณ Input เป็น sin 1.21K + 0.707sin 50K , SNR = 0 dB



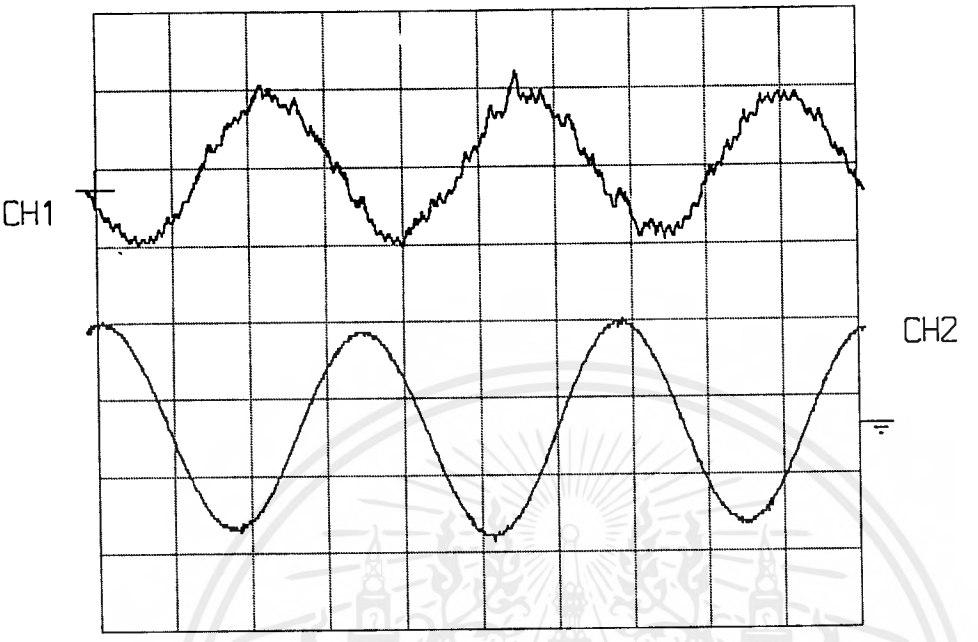
รูปที่ 12 เมื่อสัญญาณ Input เป็น sin 2.05K + 0.7sin 10K , SNR = 20 dB



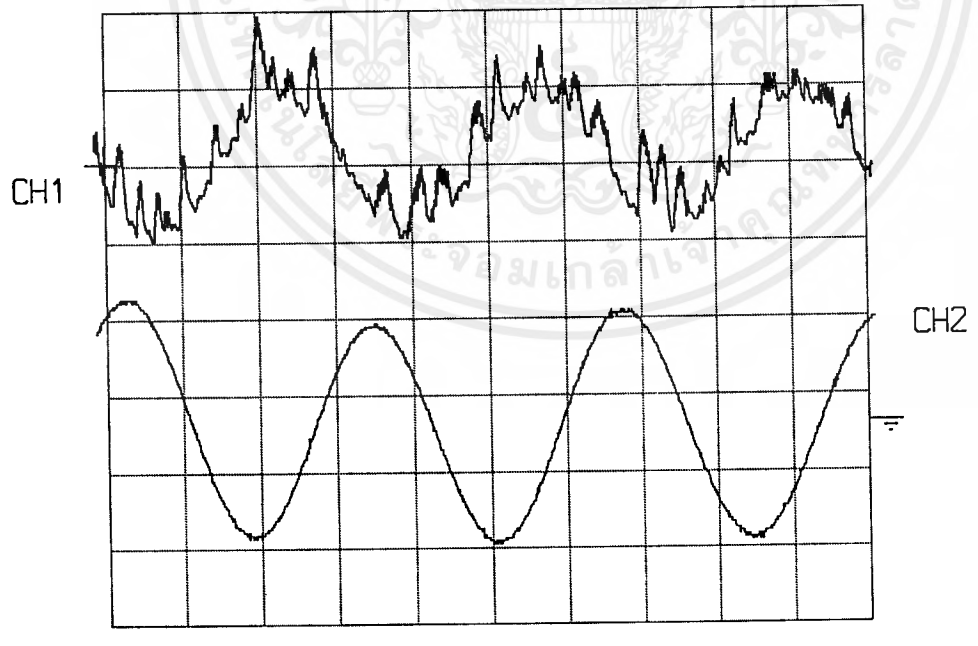
รูปที่ 13 เมื่อสัญญาณ Input เป็น sin 2.05K + 0.22sin 10K , SNR = 10 dB



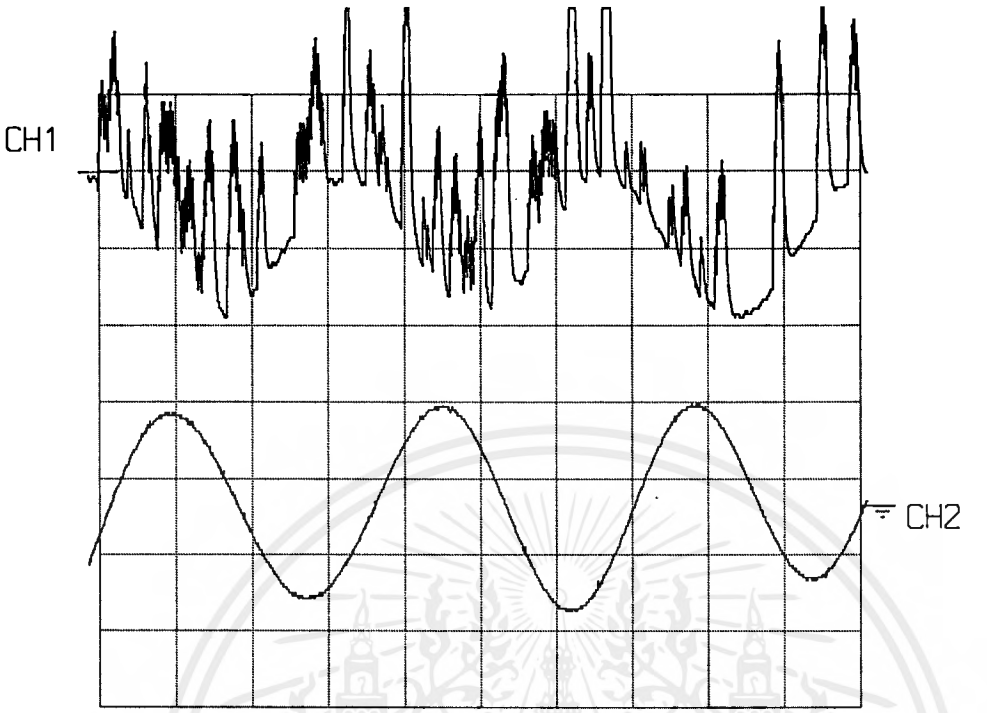
รูปที่ 14 เมื่อสัญญาณ Input เป็น sin 2.05K + 0.707sin 10K , SNR = 0 dB



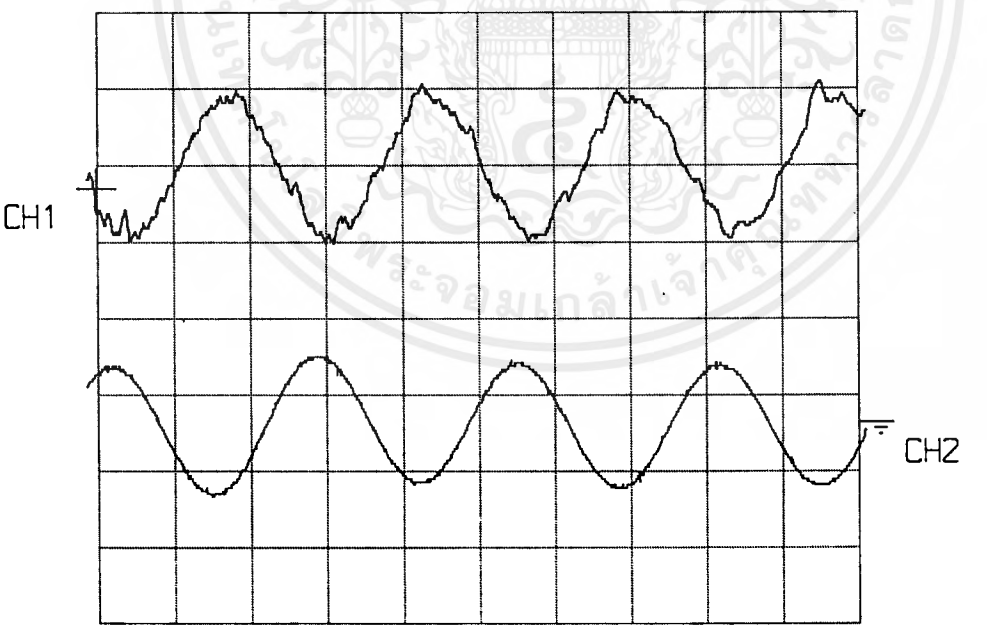
รูปที่ 15 เมื่อสัญญาณ Input เป็น sin 2.95K + 0.07sin 50K , SNR = 20 dB



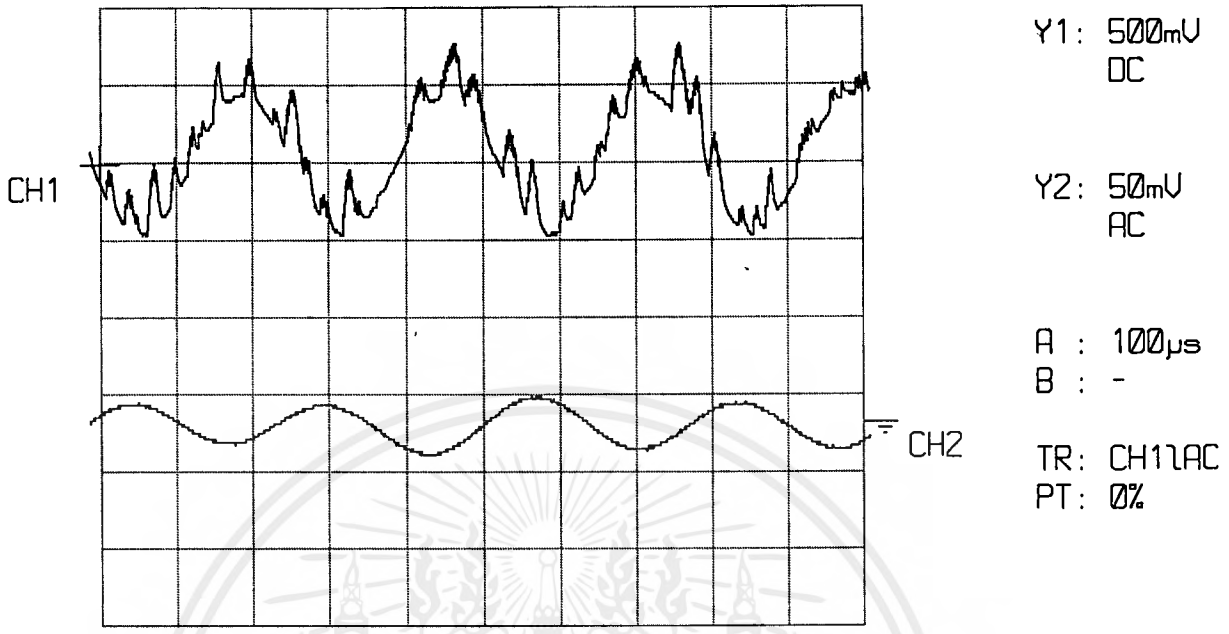
รูปที่ 16 เมื่อสัญญาณ Input เป็น sin 2.95K + 0.22sin 50K , SN



รูปที่ 17 เมื่อสัญญาณ Input เป็น sin 2.95K + 0.707sin 50K , SNR = 0 dB



รูปที่ 18 เมื่อสัญญาณ Input เป็น sin 3.75K + 0.07sin 50K , SNR = 20 dB

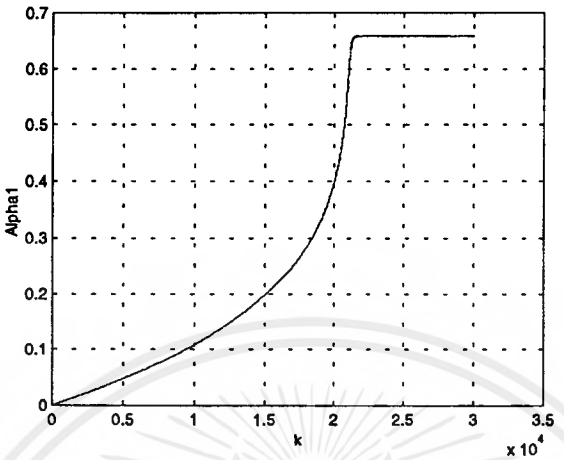


รูปที่ 19 เมื่อสัญญาณ Input เป็น sin 3.75K + 0.22sin 50K , SNR = 10 dB

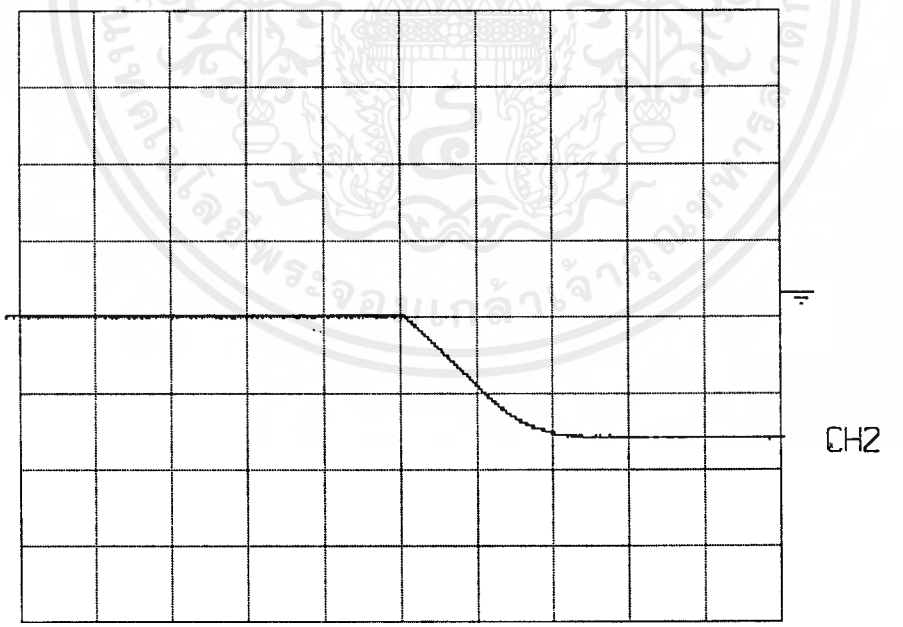
3. การเปลี่ยนแปลงค่า $\alpha_1(k)$ ตามสัญญาณ Input ลักษณะต่างๆ

ในขั้นตอนจะดูการเปลี่ยนแปลงค่า $\alpha_1(k)$ (ค่าความถี่กลางของ Filter) ตามสัญญาณ Input โดยในส่วนของโปรแกรมแทนที่จะส่งสัญญาณรูปคลื่น Output ออกมาแสดงที่ Scope

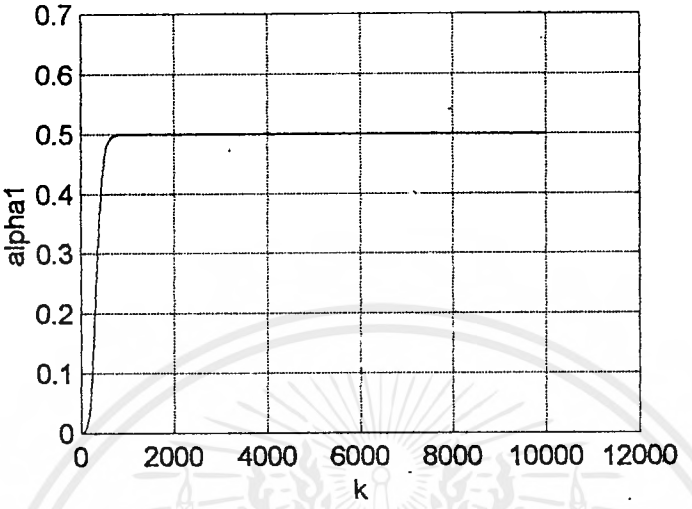
เราจะส่งสัญญาณ $\alpha_1(k)$ ออกมาทาง Output เพื่อดูการตอบสนองที่ Scope โดยการปรับ Time ที่ Scope แบบช้า (500 mS) โดยในการทดลองเราทำการเลื่อนความถี่แบบ manual ด้วยมือที่ Function Generator เพื่อดูการเปลี่ยนแปลง $\alpha_1(k)$ โดยสามารถแสดงได้ดังรูป



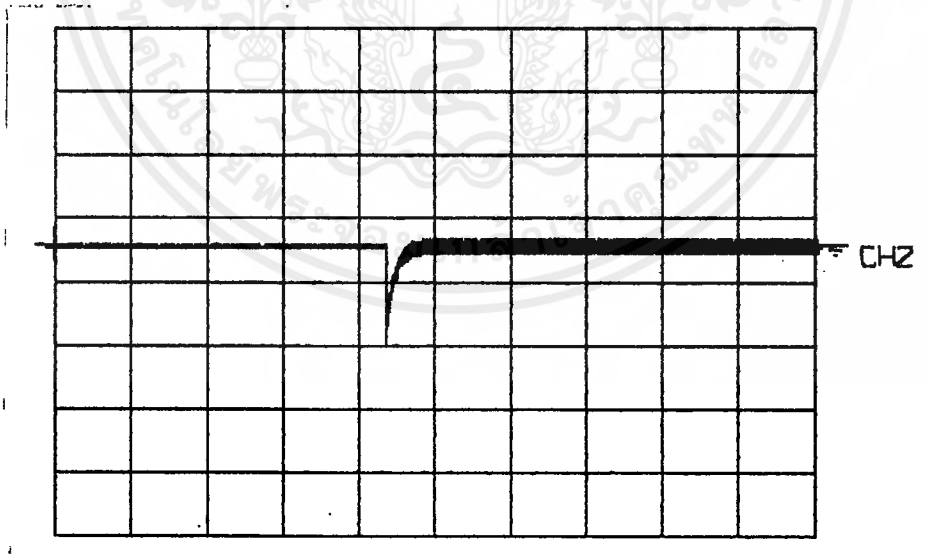
รูปที่ 20 การเปลี่ยนแปลงค่า $\alpha_1(k)$ ติดตามค่าความถี่สัญญาณ Input แบบ Adaptive IIR Band-Pass Filter (Simulate) , $\alpha_0 = 0.271$



รูปที่ 21 การเปลี่ยนแปลงค่า $\alpha_1(k)$ ติดตามค่าความถี่สัญญาณ Input แบบ Adaptive IIR Band-Pass Filter



รูปที่ 22 การเปลี่ยนแปลงค่า $\alpha_1(k)$ ติดตามค่าความถี่สัญญาณ Input แบบ Adaptive IIR Band-Pass Filter Normalized Gradient Algorithm (Simulate)



รูปที่ 23 การเปลี่ยนแปลงค่า $\alpha_1(k)$ ติดตามค่าความถี่สัญญาณ Input แบบ Adaptive IIR Band-Pass Filter Normalized Gradient Algorithm

การคำนวณเวลาในการปรับค่า $\alpha_1(k)$ จากรูปที่ 18 และ 20 ในขั้นตอนนี้จะทำการคำนวณค่า $\alpha_1(k)$ จะนำมาคำนวณหาค่าเวลาการติดตามความถี่สัญญาณ Input แล้วนำมาเปรียบเทียบกันโดยค่าความถี่ในการ Sampling 10 KHz จะสามารถนำมาทำการคำนวณได้โดยหาค่าเวลาที่ $\alpha_1(k)$ ติดตามสัญญาณ Input ตามรูปจะสามารถคำนวณค่าเวลาจริงๆ จากการเปรียบเทียบบัญญัติไตรยางค์

จาก

เวลา Sampling $\frac{1}{10KHz}$ คำนวณ 1 ครั้ง

เวลาที่อ่านได้ของ (Adaptive IIR Band-Pass Filter) คำนวณ $\frac{1\times 2}{\left[\frac{1}{10KHz}\right]}$

= จำนวนครั้งที่ $\alpha_1(k)$ ติดตามสัญญาณ Input

= 20,000 ครั้ง

โดยเปรียบเทียบกับผลของการ Simulation ที่เงื่อนไขเดียวกันจะเห็นได้ว่ามีค่าใกล้เคียงกันกับผลจากการทดลองจาก Chip DSP

และ

เวลา Sampling $\frac{1}{10KHz}$ คำนวณ 1 ครั้ง

เวลาที่อ่านได้ของ (Adaptive IIR Band-Pass Filter Normalized Gradient Algorithm)

คำนวณ $\frac{1\times 0.15}{\left[\frac{1}{10KHz}\right]}$

= จำนวนครั้งที่ $\alpha_1(k)$ ติดตามสัญญาณ Input

= 1,500 ครั้ง

โดยเปรียบเทียบกับผลของการ Simulation ที่เงื่อนไขเดียวกันจะเห็นได้ว่ามีค่าใกล้เคียงกันกับผลจากการทดลองจาก Chip DSP

เพราะฉะนั้นจะเห็นได้ว่าการติดตามความถี่สัญญาณ Input ผลการทดลองจาก Chip DSP ของ $\alpha_1(k)$ โดยวิธี Adaptive IIR Band-Pass Filter Normalized Gradient Algorithm จะติดตามได้ดีกว่าตามผลจากการลองโดยวิธี Adaptive IIR Band-Pass Filter เป็นจริงตาม การ Simulate

บทที่ 5
สรุปผลการทดลอง

1. สรุปผลการทดลอง

จากการศึกษาวิเคราะห์และออกแบบวงจรกรองความถี่กลางผ่านแบบ IIR ปรับค่าตัวแปรอัตโนมัติโดยวิธีการนอร์มอลไลซ์เกรเดียนซ์ เปรียบเทียบกับ อะแดปทีป IIR จากผลการทดลองจะเห็นว่าค่า α_1 ที่ติดตามสัญญาณ Input โดยวิธีการนอร์มอลไลซ์เกรเดียนซ์จะติดตามสัญญาณ Input ได้เร็วกว่าแบบอะแดปทีป IIR ดังรูปที่ 23 กับ 21 ซึ่งจากการทดลอง ได้ทดลองใส่ Input รวมกับสัญญาณรบกวนแบบเกาส์เซียนส์ ที่ระดับ 20 dB , 10 dB และ 0 dB ตามลำดับ และเมื่อเปรียบเทียบกับผลการ Simulation ก็เห็นได้ว่าเป็นไปตามทฤษฎีทุกอย่าง แต่มีข้อจำกัดของในการติดตามสัญญาณ Input ของ α_1 ในการทดลองกับบอร์ด DSP โดย $\alpha_0 = 0.7$ จะสามารถติดตามความถี่สัญญาณ Input ต่ำสุดประมาณ 1.7 KHz และทางด้านความถี่สูงประมาณ 2.8 KHz รอบๆความถี่กลาง (2.5 KHz) ซึ่งถ้า α_0 มากกว่านี้จะมีขอบเขตแคบกว่านี้อีก โดยการ Simulation จะสามารถติดตามความถี่ได้กว้างกว่าแต่จะใช้เวลาในการคำนวณนานมากขึ้น

ส่วนในการกำจัดสัญญาณรบกวนที่แยกเอาสัญญาณ Sine ออกจากสัญญาณรบกวนนั้น จะเห็นได้ว่าคลื่นสัญญาณ Output ออกมาเป็นรูปคลื่นสัญญาณ Sine ในระดับที่ดีมาก ซึ่งจากรูปของผลการทดลองจะเห็นได้ว่าสัญญาณรบกวนที่ 0 dB ซึ่งเป็นสัญญาณรบกวนที่ใกล้เคียงกับความจริงมากที่สุดก็สามารถที่จะกำจัดสัญญาณรบกวนออกมาได้ดีมาก

2. แนวทางในการพัฒนาและประยุกต์ใช้งาน

1. ทำการทดลองโดยการทำการสร้าง การ์ด Interface กับ คอมพิวเตอร์ PC ที่มีรูปแบบการคำนวณในรูปของ 32 bits และใช้การเขียนโปรแกรมด้วยภาษาชั้นสูงในคอมพิวเตอร์ โดยเลือกใช้ D/A และ A/D ที่มี Sampling Rate สูงๆ จะสามารถทำการทดลองในเงื่อนไขต่างๆ ได้กว้างขึ้น เช่นสามารถทดลองในช่วงความถี่กว้างขึ้น มีการคำนวณในรูปแบบที่ทำการประมาณค่าน้อยลง แต่จะสามารถที่จะนำไปใช้งานจริงได้ยาก

2. ทำการทดลองเขียนโปรแกรมที่สามารถเก็บคาของการ Update สัมประสิทธิ์ต่างๆ ในรูปแบบ 32 bits โดยโปรแกรมจะมีความซับซ้อนขึ้นพอสมควรให้สามารถทำการคำนวณได้ทัน

3. ทำการทดลองโดยสามารถปรับค่า Q-Factor ได้ด้วยวิธีการนี้ โดยการเขียนโปรแกรมเพิ่มเติมจากผลการทดลองนี้

4. อาจทำการเขียนโปรแกรมใน Chip DSP ที่มีการคำนวณในรูปทศนิยมได้

3. ปัญหาที่เกิดและทางแก้ไข

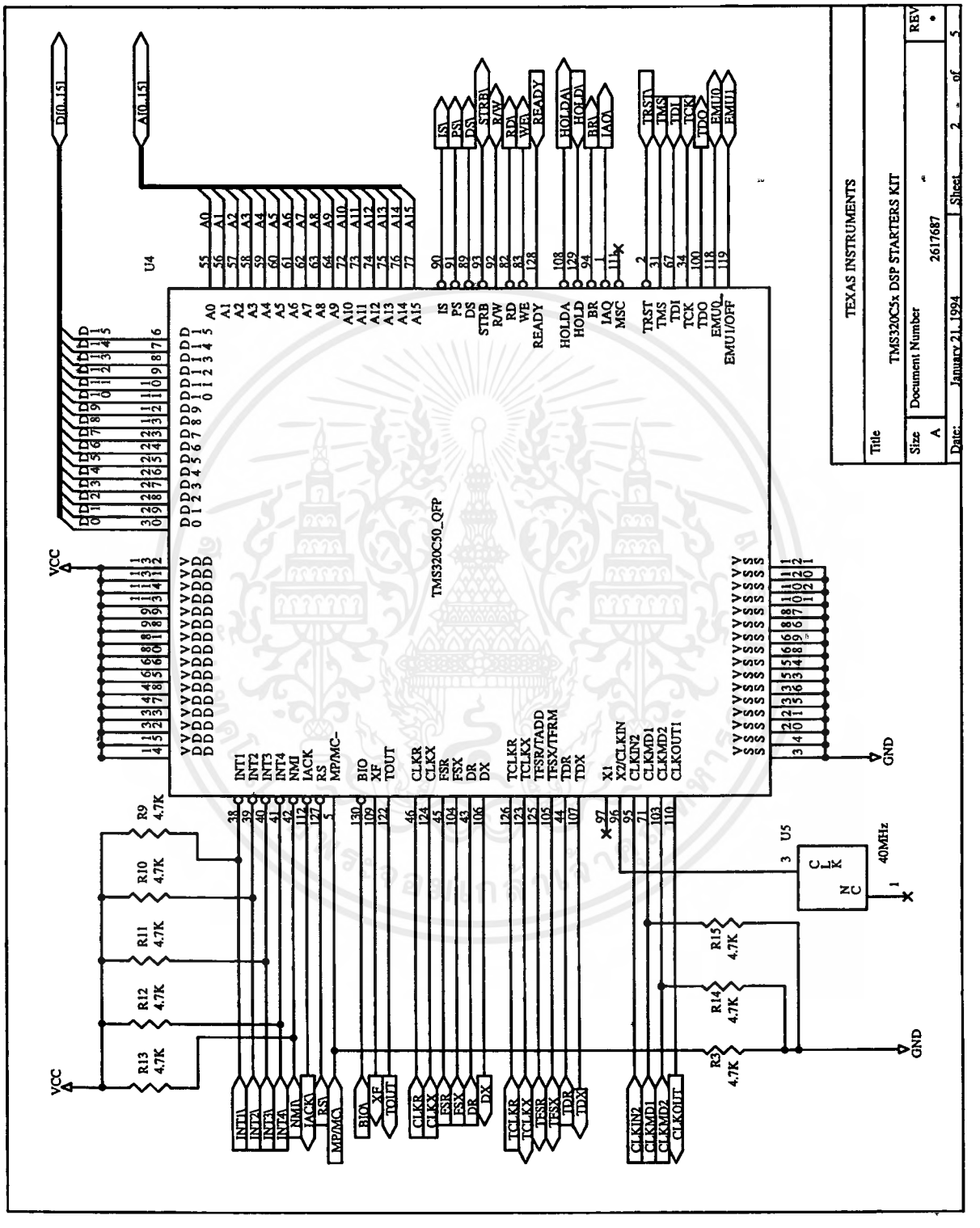
ในการทดลองปัญหาที่เกิดขึ้นประการหนึ่งจะขึ้นอยู่กับอุปกรณ์ประกอบการทดลอง ซึ่งต้องใช้ความละเอียดมากในการปรับ โดยค่าที่เปลี่ยนไปในการปรับแต่ละครั้งจะมีผลกับ Output ที่ได้เสมอ อีกประการหนึ่ง การเขียนโปรแกรมที่ติดต่อกับบอร์ด DSP ต้องศึกษาในโครงสร้างของบอร์ดให้เข้าใจ อาทิ เช่น หน่วยความจำ , คำสั่งต่างๆ ก่อนที่จะนำไปใช้งาน ในโครงงานนี้ใช้ภาษาแอสแซมบลี้ ซึ่งยากต่อความเข้าใจ แต่จะมีข้อดีที่สามารถทำงานได้เร็ว แต่ส่วนนี้สามารถใช้ภาษาซี ซึ่งใช้กันแพร่หลาย และมีผู้เชี่ยวชาญมากกว่าทดแทนได้ โดยผู้ใช้จะสามารถ download C-Compiler พร้อมทั้ง คู่มือ User's Guide ได้จาก Web site ของบริษัท Texas Instuments

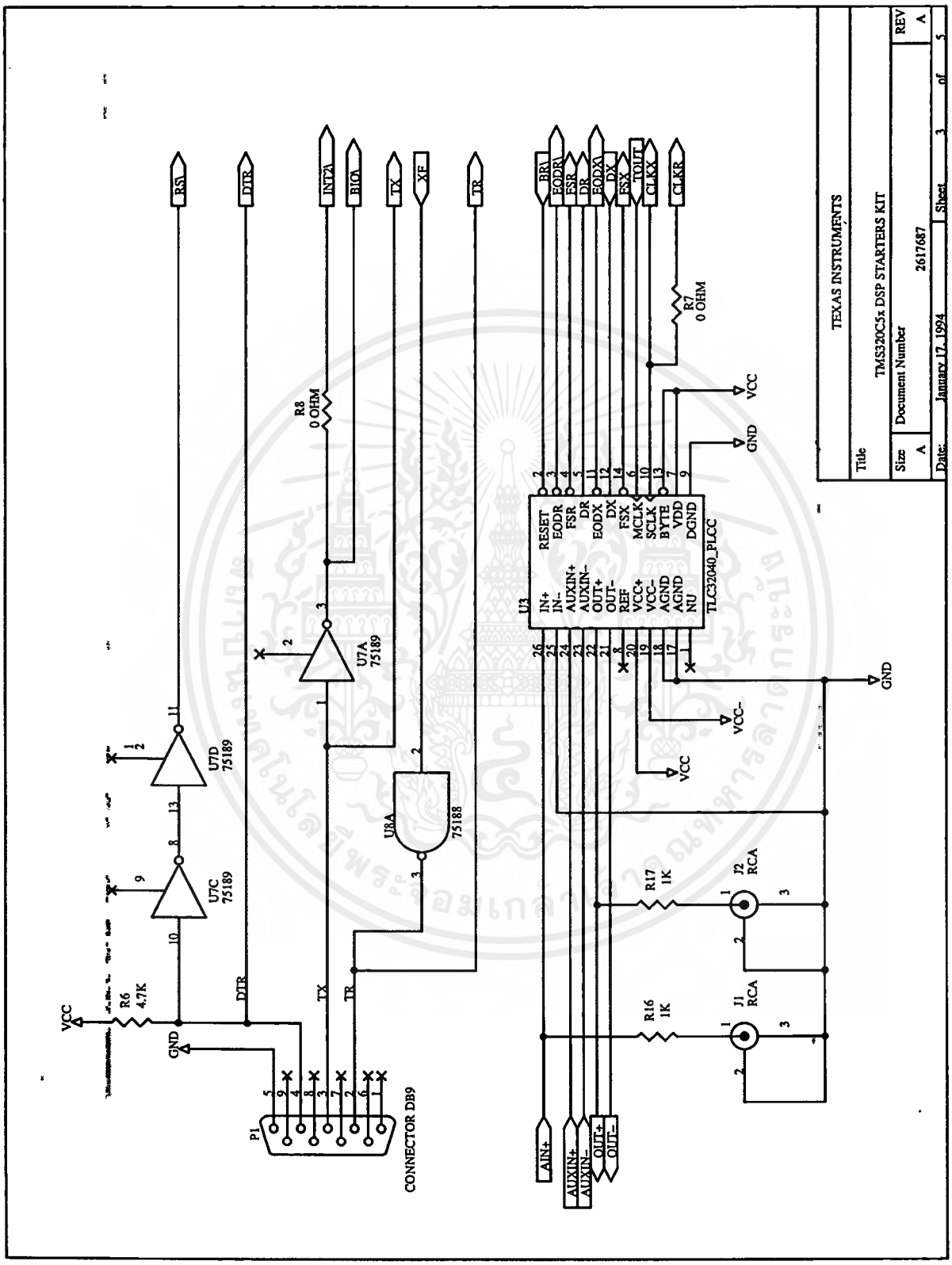
ภาคผนวก ก.

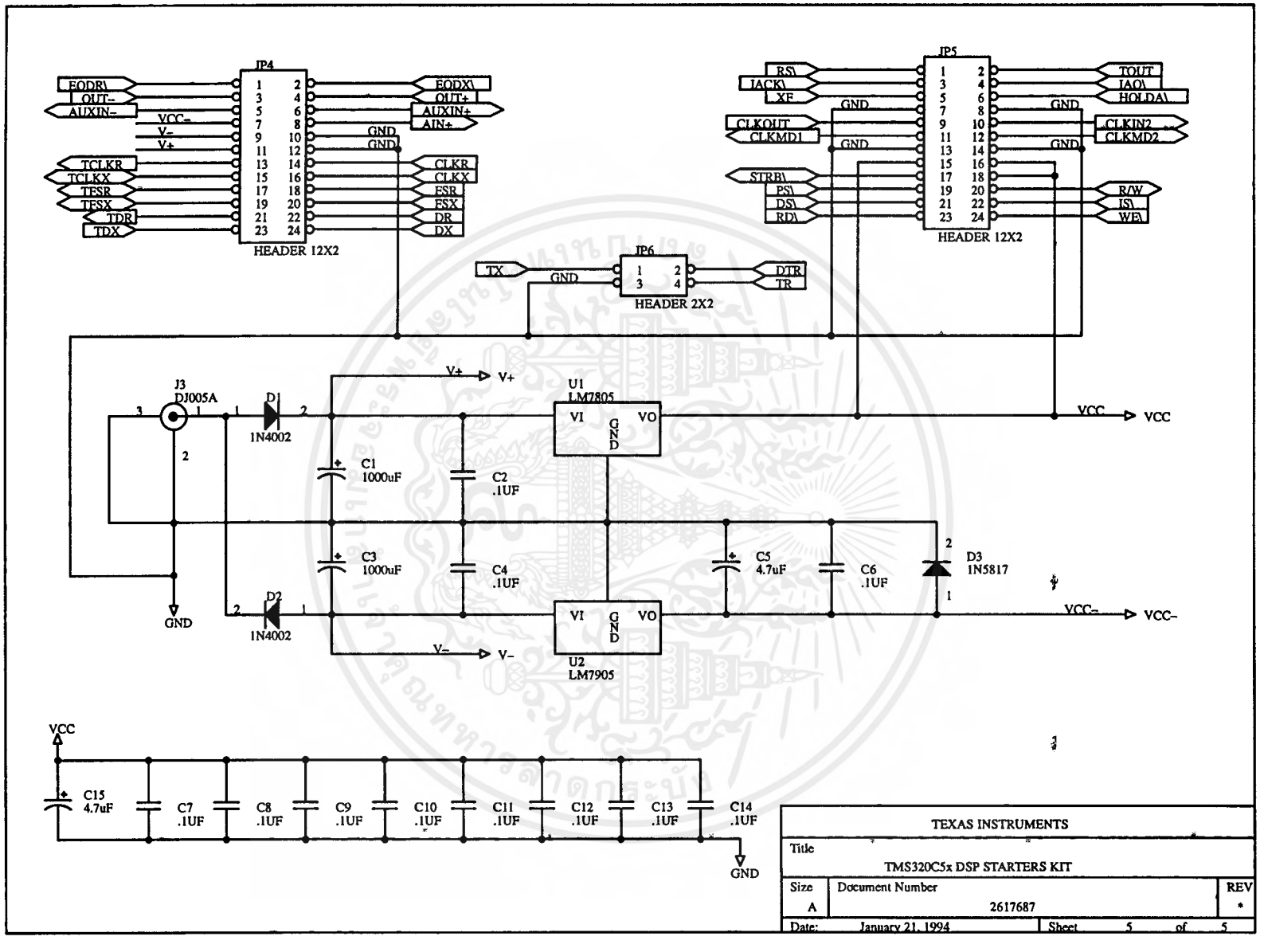
DSK Starter Kit (DSK) Circuti Board Dimensions

and

Schematic Diagrams






JP4
EODR\
OUT-
AUXIN-
VCC-
V-
V+
TCLKR
TCLKX
TESR
TESX
TDR
TDX
EODX\
OUT+
AUXIN+
AIN+
GND
GND
CLKR
CLKX
ESR
ESX
DR
DX
HEADER 12X2
JP5
RS\
IACK\
XF
GND
CLKOUT
CLKMD1
GND
STRB\
PS\
DS\
RD\
TOUT
IAO\
HOLDA\
GND
GND
CLKIN2
CLKMD2
R/W
IS\
WE\
HEADER 12X2
JP6
TX
GND
DTR
TR
HEADER 2X2
J3
DJ005A
D1
1N4002
D2
1N4002
GND
V+
V+
U1
LM7805
VI
GND
VO
VCC
VCC
C1
1000uF
C2
.1UF
C3
1000uF
C4
.1UF
C5
4.7uF
C6
.1UF
D3
1N5817
VCC-
VCC-
V-
V-
U2
LM7905
VI
GND
VO
VCC
C15
4.7uF
C7
.1UF
C8
.1UF
C9
.1UF
C10
.1UF
C11
.1UF
C12
.1UF
C13
.1UF
C14
.1UF
GND
TEXAS INSTRUMENTS
Title
TMS320C5x DSP STARTERS KIT
Size
A
Document Number
2617687
REV
*
Date:
January 21, 1994
Sheet
5
of
5

ภาคผนวก ข.

โปรแกรมการทำงาน

โปรแกรมที่ 1 โปรแกรมแสดงการติดต่อกับบอร์ด ทดสอบ Low-Pass Filter ของ Board

```
* FUNCTION:    Sampling an analog input signal                    *
*             and sending it to analog output                    *
*             with an overall gain of 1                          *
*                                                                *
* DESCRIPTION: This program initializes the TMS320C50 and        *
*             TLC32040 on the C5x DSK Board.                     *
*                                                                *
*             BAND-PASS: ON                                      *
*             Low cut-off frequency: 3.906 kHz                   *
*             High cut-off frequency: 326 Hz                     *
*             Sampling frequency: 15.625 kHz                     *
*                                                                *
* OPERATING CONDITIONS:                                          *
*                                                                *
*       ************************************************         *
*       * CAUTION: MAXIMUM INPUT VOLTAGE: 3 V  *                 *
*       ************************************************         *
*                                                                *
*             ANALOG INPUT <- Function Generator                 *
*                                                                *
*             ANALOG OUTPUT -> Oscilloscope                      *
*                                                                *
*                                                                *
* ASSEMBLING AND DIRECT EXECUTION:                               *
*                                                                *
* You must assemble and link this program with the               *
* Assembler/linker provided while purchasing the DSK.            *
*                                                                *
*             DSK5A AICDSK50.ASM ->generates AICDSK50.DSK        *
```

```
*          DSK5L AICDSK50.DSK                                    *
*                                                                *
**********************************************************************
*
* ASSEMBLING AND EXECUTION THROUGH DEBUGGER:                     *
*                                                                *
* You must assemble and link this program with the               *
* Assembler/linker provided while purchasing the DSK             *
*                                                                *
*          DSK5A AICDSK50.ASM ->generates AICDSK50.DSK           *
*          DSK5D                                                 *
*                                                                *
* Then, reset the DSK board by typing R                          *
* Then, load the DSK program by typing LD then AIC               *
* Then, execute the DSK program by typing XR                     *
**********************************************************************

        .mmregs         ; declaration of memory mapped registers*


**********************************************************************
* In this section, TA, RA, TB, RB, AIC_CTR registers are declared in data *
* memory and initialized.                                        *
**********************************************************************
        .ds   0f00h                 ; User's space data memory
TA      .word  16                   ; Fcut = 3.906 KHz
RA      .word  16

TB      .word  20                   ; Fs = 15.625 kHz
RB      .word  20
```

```
AIC_CTR .word   19h                 ; Overall gain=1, sync, loopback disabled
                                    ; band-pass filter enabled

;-------------------------------------------------------------;
; Bit definition of the AIC_CTR control register              ;
;-------------------------------------------------------------;
;     +----------+----------------+
;     | G1  G0 | SY  AX  LB  BP |            G1  G0  Gain
;     +----------+----------------+           ---------------
;       |  |    |  |  |  +-- Band-Pass Filter   0   0   1/2
;       |  |    |  |  +------ Loopback          1   1   1/2
;       |  |    |  +---------- Aux              0   1   1
; Gain ---+---+    +------------- Synch          1   0   2

*************************************************************************
* Set up the interrupt vector table in program memory and make the link *
* between the interrupt vector and its associated interrupt subroutine. *
* The interrupt vector table starts at 0800h, but location of serial    *
* port receive interrupt RINT is at 080Ah since it is the 5th in the    *
* interrupt table.                                                      *
*************************************************************************
        .ps    080ah                ; Interrupt Vectors memory
rint:   B      RECEIVE              ; 080Ah: Serial port receive interrupt
xint:   B      TRANSMIT             ; 080Ch: Serial port transmit interrupt

*************************
* START OF MAIN PROGRAM *
*************************
        .ps 0a00h                   ; User's Program memory
        .entry                      ; entry point of the program
```

```
*********************************
* TMS320C50 DSP Initialization *
*********************************
START:  SETC   INTM             ; Disable interrupts
        LDP    #0               ; Set data page pointer
        OPL    #0834h,PMST      ; Interrupt table starts at 0800h
                                ; Program and data ON-CHIP SARAM
                                ; (RAM=1 and OVLY=1)
                                ; microcomputer mode MP/MC=0
                                ; NDX=1, TRM=0 => C5X not enhanced mode
        LACC   #0
        SAMM   CWSR             ; zero wait-state access to memory
        SAMM   PDWSR

* Reset AIC by writing to PA2 (address >52) to DSK
        SPLK   #022h,IMR        ; using XINT for synch (and INT2 for debugger)
        CALL   AICINIT          ; call to subroutine of initialization
                                ; of the AIC

*************************************************************************
* After the next 4 instructions, the serial port receive interrupt is   *
* enabled. When RINT is triggered, the receive interrupt subroutine     *
* (RECEIVE: ) is executed: it stores the data contained in the DRR register *
* to Accumulator  then stores it to the DXR register.                   *
*************************************************************************
        CLRC   OVM              ; OVM = 0 - Overflow mode
        SPM    0                ; PM = 0 no shift after product reg.
        SPLK   #012h,IMR        ; RINT only (and INT2 for debugger)
        CLRC   INTM             ; enable interrupt
```

```
*************************************************
* Infinite loop - Waiting for reception of data *
*                                               *
* NB: the RINT interrupt is enabled             *
*************************************************
WAIT:                                   ; wait until interrupt
        B     WAIT                      ; wait for input data

***********************
* END OF MAIN PROGRAM *
***********************

***********************
* INTERRUPT ROUTINES *
***********************

*************************************
* Receive Interrupt Service Routine *
*************************************
RECEIVE:
        LAMM   DRR                      ; read data from DRR
        AND    #0FFFCh                  ; 14-bit resolution of the AIC
        SAMM   DXR                      ; send data to DXR (primary communic.)
        RETE
*************************************
* Transmit Interrupt Service Routine *
*************************************
TRANSMIT:
        RETE
```

```
*************
* FUNCTIONS *
*************


*****************************************************************************
* The AICINIT routine initializes the Serial Port and the TLC320C40 AIC, *
* by the following steps:                                                   *
* Step 1: Set the Timer frequency (10.000 MHz for AIC MCLK).                *
* Step 2: Configure frame synchronization signals.                          *
* Step 3: Initialisation of the Serial port                                 *
* Step 4: Initialisation of the AIC                                         *
*                                                                           *
* NB: AIC Reset line is connected to the /BR (Bus Request) pin of the       *
*     'C50. This pin is driven low when external global memory is           *
*     accessed. Therefore to reset the AIC, global memory must be           *
*     defined and accessed to drive the /BR pin low for at least 800ns.     *
*****************************************************************************
AICINIT: SPLK   #20h,TCR        ; To generate 10.000 MHz from Tout
         SPLK   #01h,PRD        ; for AIC master clock
         MAR    *,AR0
         LACC   #0008h          ; Non continuous mode, FSM=1, FSX as input
         SACL   SPC             ; reset the serial port
         LACC   #00c8h          ; 16 bit words
         SACL   SPC             ; bring the serial port out of reset
         LACC   #080h           ; init 8000h-FFFFh as global memory
         SACH   DXR             ; Dummy word sent to clear SPC register
         SACL   GREG            ; Store to Global Memory Alloc Reg.
         LAR    AR0,#0FFFFh     ; Use AR0 to point to location FFFFh
         RPT    #10000          ; Access global memory 10000 times
         LACC   *,0,AR0         ; to drive pin low for duration
```

```
        SACH    GREG                ; Restore GREG to 0000

*********************************************************************************
* The 3 blocks of instructions below send the parameters TA, TB and AIC_CTR *
* to the AIC for a 16 kHz sample rate with a gain setting of 1, BP enabled. *
* The contents of TA and TB ands AIC_CTR registers are each shifted 9 bits  *
* left then added to 2 control bits to configure a complete 16 bit AIC DX   *
* DATA word before its transmission to the AIC.                             *
*                                                                           *
* NB: Sending such a parameter requires a "secondary communication", hence  *
*    the call to the AIC_2ND subroutine.                                    *
*********************************************************************************
        ;----------------------
        LDP     #TA             ; Initialization of the data-page pointer
        SETC    SXM             ; Set sign extension mode
        LACC    TA,9            ; TA is shifted left 9 bits
        ADD     RA,2            ; RA is shifted left 2 bits
        CALL    AIC_2ND         ; call to AIC_2ND subroutine
        ;----------------------
        LDP     #TB             ; Initialization of the data-page pointer
        LACC    TB,9            ; TB is shifted left 9 bits
        ADD     RB,2            ; RB is shifted left 2 bits
        ADD     #02h
        CALL    AIC_2ND
        ;----------------------
        LDP     #AIC_CTR        ; Initialization of the data-page pointer
        LACC    AIC_CTR,2       ; AIC_CTR is shifted left 2 bits.
        ADD     #03h
        CALL    AIC_2ND
        RET
```

```
*************************************************************************
* The AIC_2ND routine initializes a secondary transmission, by the  *
* following steps:                                                  *
* Step 1: enable serial port transmit interrupt                     *
* Step 2: send 03h to initiate a secondary transmission             *
* Step 3: wait until ready for new data (idle)                      *
* Step 4: send AIC parameter                                        *
* Step 5: disable serial port transmit interrupt                    *
*                                                                   *
* NB:  The lower part of the accumulator contains the contents of   *
*      the AIC DX DATA configuration word. The higher part contains *
*      a control word which initiates secondary transmission mode   *
*************************************************************************
AIC_2ND:
        LDP   #0
        SACH  DXR               ; dummy word sent to clear SPC register
        CLRC  INTM              ; enable interrupt
        IDLE                    ; wait for transmit interrrupt
        ADD   #6h,15            ; 0000 0000 0000 0011 | AIC DX DATA WORD
        SACH  DXR               ; Send 0003h to initiate 2nd transmission
        IDLE
        SACL  DXR               ; Send AIC DX data word
        IDLE
        LACL  #0
        SACL  DXR               ; make sure the word got sent
        IDLE
        SETC  INTM              ; disable interrupt
        RET
        .end
```

โปรแกรมที่ 2 แสดงความถี่กลางที่กำหนดได้ในโปรแกรม

```
                .MMREGS
                .ds     0f00h
TA              .word   16              ;
RA              .word   16              ; This set up of AIC registers give
                                        ; a sampling freq of 15.625 Hz

TB              .word   20              ;
RB              .word   20              ;
AIC_CTR         .word   19h             ;
OUTPUT          .word   0               ;

;Filter Coefficient Generator  bandpass
a2      .word   -655    ; -0.04
a1      .word   0       ; 0
a0      .word   655     ; 0.04
b2      .word   -15073  ; -0.92
b1      .word   31067   ; 1.896

;Storage deley line
YN      .word   0,0
XN      .word   0,0
XNL     .word   0

        .ps     0080ah
rint:   B       RECEIVE
xint:   B       TRANSMIT

        .ps     0a00h
        .entry          ;
```

```
;-----------------------------------------------------------

        SETC    INTM
        LDP     #0
        OPL     #0834h,PMST
        LACC    #0
        SAMM    CWSR
        SAMM    PDWSR
        SETC    SXM             ; SXM MUST BE SET
        SPLK    #022h,IMR       ; This turns on receive interrupt only

        CALL    AICINIT
        SPLK    #12h,IMR
        CLRC    OVM
        SPM     0
        CLRC    INTM

WAIT:                           ; a main program would go here
        B       WAIT

RECEIVE:
        LDP     #XN
        CLRC    INTM            ; ENABLE INTERRUPTS

        LAMM    DRR             ; load accumulator with word received from AIC
                                ; Next line should be commented out if the
        SACL    XN              ; store the value of received word to a variable
        LAR     AR0,#XNL        ; load AR0 with address of last delay element
        ZAP                     ; ZERO ACC AND PRODUCT REGISTERS
        MAR     *,AR0           ; AR0 is the current AR register.
```

```
        RPT     #4              ; Repeat the next instruction 80 times through
        MACD    #a2,*-          ; the complete coeff table.
        APAC                    ; Accumulate last product

        SACH    OUTPUT,2        ; ACC -> OUTPUT.  Get rid of xtra sign bit.
        LACC    OUTPUT          ; left shift
        SACL    YN
        AND     #0fffch         ; Two LSB's must be zero for AIC
        SAMM    DXR             ; write output word to transmit register
        RETE


TRANSMIT:
        RETE

;*********************************************************************
; DESCRIPTION: This routine initializes the 'C50 serial port   *
;         and the TLC320C40's (AIC) TA,RA,TB,RB and      *
;         control registers                              *
;*********************************************************************
AICINIT: SPLK   #20h,TCR        ; To generate 10 MHz from Tout
         SPLK   #01h,PRD        ; for AIC master clock
         MAR    *,AR0
         LACC   #0008h          ; Non continuous mode
         SACL   SPC             ; FSX as input
         LACC   #00c8h          ; 16 bit words
         SACL   SPC
         LACC   #080h           ; Pulse AIC reset by setting it low
         SACH   DXR
```

```
        SACL   GREG
        LAR    AR0,#0FFFFh
        RPT    #10000              ; and taking it high after 10000 cycles
        LACC   *,0,AR0             ; (.5ms at 50ns)
        SACH   GREG
      ;-----------------------
        LDP    #TA                 ;
        SETC   SXM                 ;
        LACC   TA,9                ; Initialize TA and RA register
        ADD    RA,2                ;
        CALL   AIC_2ND             ;
      ;-----------------------
        LDP    #TB
        LACC   TB,9                ; Initialize TB and RB register
        ADD    RB,2                ;
        ADD    #02h                ;
        CALL   AIC_2ND             ;
      ;-----------------------
        LDP    #AIC_CTR
        LACC   AIC_CTR,2           ; Initialize control register
        ADD    #03h                ;
        CALL   AIC_2ND             ;
        RET                        ;

AIC_2ND:
        LDP    #0                  ; Data page point is 0  (MM regs)
        SACH   DXR                 ; send ACChi 00
        CLRC   INTM                ; enable interrupts
        IDLE                       ;wait for interrupt
```

```
        ADD    #6h,15              ; 0000 0000 0000 0011 XXXX XXXX
XXXX XXXX b
        SACH   DXR                 ; send ACChi to initiate secondary protocol
        IDLE                       ; wait for interrupt
        SACL   DXR                 ; send the T register data
        IDLE                       ; wait for interrupt
        LACL   #0                  ; clear ACCUMULATOR LOW
        SACL   DXR                 ; send another to make sure 1st word got
sent
        IDLE                       ; wait for interrupt
        SETC   INTM
        RET
      .END
```

โปรแกรมที่ 3 แสดง Output ของ Adaptive IIR Band-Pass Filter Normailized Gradient Algorithm

```
;SAMPLING FREQUENCY=10.08 KHz    FREQUENCY CUTOFF=3.9 KHz
        .MMREGS
        .ds    0f00h                ; declaration of memory mapped registers
TA          .word   16
RA          .word   16
TB          .word   31
RB          .word   31
AIC_CTR     .word   19h             ;Overall gain=1, sync, loopback disabled,
                                    ;band-pass filter enabled
OUTPUT      .word   0
temp1       .word   0
temp2       .word   0
temp3       .word   0
temp4       .word   0
************************************************************
;Filter Coefficient Generator  bandpass and constant*
************************************************************
mue         .word    5          ; 0 .0003
alp1k       .word    0          ; (start only)
a2          .word   -2458       ;  -0.15  -(1-alp0)/2
a1          .word    0          ;  0
a0          .word    2458       ;  0.15   (1-alp0)/2
b2          .word   -11469      ; -.7
b1          .word    0          ;  0     (start only)
d           .word    27853      ;  1.7    (1+alp0)
gram        .word    16056      ;  0.98
gram1       .word    328        ;  0.02
************************************
```

```
;Storage deley line and any volue*
************************************
quot            .word   0
rem             .word   0
YN1             .word   0
YN2             .word   0
XN              .word   0,0
XNL             .word   0
BUF             .word   0
YNx             .word   0
PHI1            .word   0
PHI2            .word   0
ban             .word   0
bata1           .word   0
*******************************************************************************
; Set up the interrupt vector table in program memory and make the link *
*******************************************************************************
        .ps     0080ah                  ;  Interrupt Vectors memory
rint:   B       RECEIVE                 ;   Serial port receive interrupt
xint:   B       TRANSMIT                ;   Serial port transmit interrupt

*********************************
; START OF MAIN PROGRAM *
*********************************
        .ps     0a00h                   ; User's Program memory
        .entry                          ; entry point of the program

        SETC    INTM                    ; Disable interrupts
        LDP     #0                      ; Set data page pointer
        OPL     #0834h,PMST             ; Interrupt table starts at 0800h
```

```
                                ; Program and data ON-CHIP SARAM
                                ; (RAM=1 and OVLY=1)
                                ; microcomputer mode MP/MC=0
                                ; NDX=1, TRM=0 => C5X not enh
    LACC    #0
    SAMM    CWSR                ; zero wait-state access to memory
    SAMM    PDWSR
    SETC    SXM                 ;  SXM MUST BE SET
    SPLK    #022h,IMR           ; using XINT for synch
    CALL    AICINIT             ; call to subroutine of initialization of the AIC
    SPLK    #12h,IMR            ; This turns on receive interrupt only
    CLRC    OVM                 ; OVM = 0 - Overflow mode
    SPM     0                   ; PM = 0 no shift after product reg.
    CLRC    INTM                ; enable interrupt

    LAR     AR5,#alp1k
    LAR     AR3,#mue            ;  provide  AR....
    LAR     AR4,#temp4

WAIT:                           ; wait until interrupt
    B       WAIT                ; wait for input data
*******************************
; END OF MAIN PROGRAM *
*******************************
*********************************************************************************
*****************************************
; Receive Interrupt Service Routine  *
*****************************************
RECEIVE:
    LDP     #XN
```

```
    CLRC    INTM            ; ENABLE INTERRUPTS FOR DEBUGGING DSK
PURPOSES
    LAMM    DRR             ; load accumulator with word received from AIC
    SACL    XN              ; store the value of received word to a variable
    LAR     AR0,#XNL        ; load AR0 with address of last delay element
    ZAP                     ; ZERO ACC AND PRODUCT REGISTERS
    MAR     *,AR0           ; AR0 is the current AR register.

    RPT     #4              ; Repeat the next instruction 4 times through
    MACD    #a2,*-          ; the complete coeff table.
    APAC                    ; Accumulate last product

    SACH    OUTPUT,2        ; ACC -> OUTPUT.  Get rid of xtra sign bit.
    LACC    OUTPUT
; SACL    YN1
; MAR     *,AR5             ; display  OUTPUT
; LACC    *
    AND     #0fffch         ; Two LSB's must be zero for AIC
    SAMM    DXR             ; write output word to transmit register

    LACC    YN2             ; Adaptation section
    SACL    YNx
    LAR     AR1,#PHI2       ; AR1 point to add off PHI2
    MAR     *,AR1           ; AR1 is curent auxilary register
    ZAP                     ; Zero Acc&Product register
    RPT     #2              ; Repeat next instruction 2 time
    MACD    #b2,*-          ; calulate PHI1
    APAC
    SACH    PHI1,2          ; store PHI1
    LACC    PHI1
```

```
AND    #0f000h              ; Zero 4 bit low off PHI1
SACL   PHI1
LAR    AR0,#YN1             ; calculate alp1(k+1)
MAR    *,AR0                ; AR0 is curent AR register
ZAP
MAC    #PHI1,*              ; yn*PHI1
APAC
SACH   temp1,2
ZAP
LAR    AR0,#temp1
MAR    *,AR0
SQRA   *,AR2
APAC
SACH   temp2,2
ZAP
LACC   temp2
SACL   ban
LAR    AR0,bata1
MAR    *,AR0
ZAP
RPT    #1
MACD   #gram,*-
APAC
SACH   bata1,2
LACC   bata1
AND    #0f000h
SACL   bata1
ZAP

LDP    #0
```

```
LT      mue
MPY     #bata1
SPH     temp3,2
LACC    bata1
ABS
SACH    bata1
LACL    mue
ABS
RPT     #15
SUBC    bata1
BIT     temp3,0
RETCD   NTC
 SACL   quot
 SACH   rem
LACL    #0
RETD
 SUB    quot
 SACL   quot
ZAP

LAR     AR0,#quot
MAR     *,AR0
MAC     #temp1,*
APAC
SACH    temp4,2
LACC    alp1k
MAR     *,AR4
ADD     *,0
SACL    alp1k
```

```
        MAR     *,AR5
        ZAP
        MAC     #d,*                    ;alp1(1+alp0)
        APAC
        SACH    b1,2                    ;UPDATE  COEFFICIENT
        RETE

TRANSMIT:
        RETE
*******************************************************************************
; The AICINIT routine initializes the Serial Port and the TLC320C40 AIC, *
*******************************************************************************
AICINIT:
        SPLK    #20h,TCR                ; To generate 10 MHz from Tout
        SPLK    #01h,PRD                 for AIC master clock
        MAR     *,AR0
        LACC    #0008h                  ; Non continuous mode
        SACL    SPC                     ; FSX as input
        LACC    #00c8h                  ; 16 bit words
        SACL    SPC
        LACC    #080h                   ; Pulse AIC reset by setting it low
        SACH    DXR
        SACL    GREG
        LAR     AR0,#0FFFFh
        RPT     #10000                  ; and taking it high after 10000 cycles
        LACC    *,0,AR0                 ; (.5ms at 50ns)
        SACH    GREG
        ;-----------------------
        LDP     #TA
        SETC    SXM
```

```
        LACC   TA,9             ; Initialize TA and RA register
        ADD    RA,2
        CALL   AIC_2ND
        ;-----------------------
        LDP    #TB
        LACC   TB,9             ; Initialize TB and RB register
        ADD    RB,2
        ADD    #02h
        CALL   AIC_2ND
        ;-----------------------
        LDP    #AIC_CTR
        LACC   AIC_CTR,2        ; Initialize control register
        ADD    #03h
        CALL   AIC_2ND
        RET

AIC_2ND:
        LDP    #0               ; Data page point is 0  (MM regs)
        SACH   DXR              ; send ACChi 00
        CLRC   INTM             ; enable interrupts
        IDLE                    ; wait for interrupt
        ADD    #6h,15           ; 0000 0000 0000 0011 XXXX XXXX XXXX
                                  XXXX b
        SACH   DXR              ; send ACChi to initiate secondary protocol
        IDLE                    ; wait for interrupt
        SACL   DXR              ; send the T register data
        IDLE                    ; wait for interrupt
        LACL   #0               ; clear ACC=0
        SACL   DXR              ; send another to make sure 1st word got sent
        IDLE                    ; wait for interrupt
```

```
SETC   INTM
RET
.END
```

โปรแกรมที่ 4 แสดงการติดตามสัญญาณ Input ของ Alpha 1

```
;SAMPLING FREQUENCY=10.08 KHz    FREQUENCY CUTOFF=3.9 KHz
        .MMREGS
        .ds    0f00h                ; declaration of memory mapped registers
TA              .word   16
RA              .word   16
TB              .word   31
RB              .word   31
AIC_CTR         .word   19h          ;Overall gain=1, sync, loopback disabled,
                                     ;band-pass filter enabled
OUTPUT          .word   0
temp1           .word   0
temp2           .word   0
temp3           .word   0
temp4           .word   0
*************************************************************
;Filter Coefficient Generator  bandpass and constant*
*************************************************************
mue             .word    5           ; 0 .0003
alp1k           .word    0           ; (start only)
a2              .word   -2458        ;  -0.15  -(1-alp0)/2
a1              .word    0           ;  0
a0              .word    2458        ;  0.15   (1-alp0)/2
b2              .word   -11469       ; -.7
b1              .word    0           ;  0     (start only)
d               .word    27853       ;  1.7    (1+alp0)
gram            .word    16056       ;  0.98
gram1           .word    328         ;  0.02
************************************
;Storage deley line and any volue*
```

```
*************************************
quot            .word   0
rem             .word   0
YN1             .word   0
YN2             .word   0
XN              .word   0,0
XNL             .word   0
BUF             .word   0
YNx             .word   0
PHI1            .word   0
PHI2            .word   0
ban             .word   0
bata1           .word   0
*****************************************************************************
; Set up the interrupt vector table in program memory and make the link *
*****************************************************************************
      .ps   0080ah                  ;  Interrupt Vectors memory
rint:  B    RECEIVE                 ;  Serial port receive interrupt
xint:  B    TRANSMIT                ;  Serial port transmit interrupt

*********************************
; START OF MAIN PROGRAM *
*********************************
      .ps   0a00h                   ; User's Program memory
      .entry                        ; entry point of the program

      SETC  INTM                    ; Disable interrupts
      LDP   #0                      ; Set data page pointer
      OPL   #0834h,PMST             ; Interrupt table starts at 0800h
                                    ; Program and data ON-CHIP SARAM
```

```
                                ; (RAM=1 and OVLY=1)
                                ; microcomputer mode MP/MC=0
                                ; NDX=1, TRM=0 => C5X not enh
    LACC    #0
    SAMM    CWSR                ; zero wait-state access to memory
    SAMM    PDWSR
    SETC    SXM                 ;  SXM MUST BE SET
    SPLK    #022h,IMR           ; using XINT for synch
    CALL    AICINIT             ; call to subroutine of initialization of the AIC
    SPLK    #12h,IMR            ; This turns on receive interrupt only
    CLRC    OVM                 ; OVM = 0 - Overflow mode
    SPM     0                   ; PM = 0 no shift after product reg.
    CLRC    INTM                ; enable interrupt

    LAR     AR5,#alp1k
    LAR     AR3,#mue            ;  provide  AR....
    LAR     AR4,#temp4

WAIT:                           ; wait until interrupt
    B       WAIT                ; wait for input data
*******************************
; END OF MAIN PROGRAM *
*******************************
*****************************************************************************
*****************************************
; Receive Interrupt Service Routine  *
*****************************************
RECEIVE:
    LDP     #XN
```

```
    CLRC    INTM            ; ENABLE INTERRUPTS FOR DEBUGGING DSK
PURPOSES
    LAMM    DRR             ; load accumulator with word received from AIC
    SACL    XN              ; store the value of received word to a variable
    LAR     AR0,#XNL        ; load AR0 with address of last delay element
    ZAP                     ; ZERO ACC AND PRODUCT REGISTERS
    MAR     *,AR0           ; AR0 is the current AR register.

    RPT     #4              ; Repeat the next instruction 4 times through
    MACD    #a2,*-          ; the complete coeff table.
    APAC                    ; Accumulate last product

    SACH    OUTPUT,2        ; ACC -> OUTPUT.  Get rid of xtra sign bit.
    LACC    OUTPUT
    SACL    YN1
    MAR     *,AR5           ; display  OUTPUT
    LACC    *
    AND     #0fffch         ; Two LSB's must be zero for AIC
    SAMM    DXR             ; write output word to transmit register

    LACC    YN2             ; Adaptation section
    SACL    YNx
    LAR     AR1,#PHI2       ; AR1 point to add off PHI2
    MAR     *,AR1           ; AR1 is curent auxilary register
    ZAP                     ; Zero Acc&Product register
    RPT     #2              ; Repeat next instruction 2 time
    MACD    #b2,*-          ; calulate PHI1
    APAC
    SACH    PHI1,2          ; store PHI1
    LACC    PHI1
```

```
AND    #0f000h              ; Zero 4 bit low off PHI1
SACL   PHI1
LAR    AR0,#YN1             ; calculate alp1(k+1)
MAR    *,AR0                ; AR0 is curent AR register
ZAP
MAC    #PHI1,*              ; yn*PHI1
APAC
SACH   temp1,2
ZAP
LAR    AR0,#temp1
MAR    *,AR0
SQRA   *,AR2
APAC
SACH   temp2,2
ZAP
LACC   temp2
SACL   ban
LAR    AR0,bata1
MAR    *,AR0
ZAP
RPT    #1
MACD   #gram,*-
APAC
SACH   bata1,2
LACC   bata1
AND    #0f000h
SACL   bata1
ZAP

LDP    #0
```

```
LT      mue
MPY     #bata1
SPH     temp3,2
LACC    bata1
ABS
SACH    bata1
LACL    mue
ABS
RPT     #15
SUBC    bata1
BIT     temp3,0
RETCD   NTC
SACL    quot
SACH    rem
LACL    #0
RETD
SUB     quot
SACL    quot
ZAP

LAR     AR0,#quot
MAR     *,AR0
MAC     #temp1,*
APAC
SACH    temp4,2
LACC    alp1k
MAR     *,AR4
ADD     *,0
SACL    alp1k
```

```
        MAR     *,AR5
        ZAP
        MAC     #d,*                    ;alp1(1+alp0)
        APAC
        SACH    b1,2                    ;UPDATE  COEFFICIENT
        RETE

TRANSMIT:
        RETE
*******************************************************************************
; The AICINIT routine initializes the Serial Port and the TLC320C40 AIC, *
*******************************************************************************
AICINIT:
        SPLK    #20h,TCR                ; To generate 10 MHz from Tout
        SPLK    #01h,PRD                 for AIC master clock
        MAR     *,AR0
        LACC    #0008h                  ; Non continuous mode
        SACL    SPC                     ; FSX as input
        LACC    #00c8h                  ; 16 bit words
        SACL    SPC
        LACC    #080h                   ; Pulse AIC reset by setting it low
        SACH    DXR
        SACL    GREG
        LAR     AR0,#0FFFFh
        RPT     #10000                  ; and taking it high after 10000 cycles
        LACC    *,0,AR0                 ; (.5ms at 50ns)
        SACH    GREG
        ;-----------------------
        LDP     #TA
        SETC    SXM
```

```
        LACC   TA,9                ; Initialize TA and RA register
        ADD    RA,2
        CALL   AIC_2ND
        ;-----------------------
        LDP    #TB
        LACC   TB,9                ; Initialize TB and RB register
        ADD    RB,2
        ADD    #02h
        CALL   AIC_2ND
        ;-----------------------
        LDP    #AIC_CTR
        LACC   AIC_CTR,2           ; Initialize control register
        ADD    #03h
        CALL   AIC_2ND
        RET

AIC_2ND:
        LDP    #0                  ; Data page point is 0  (MM regs)
        SACH   DXR                 ; send ACChi 00
        CLRC   INTM                ; enable interrupts
        IDLE                       ; wait for interrupt
        ADD    #6h,15              ; 0000 0000 0000 0011 XXXX XXXX XXXX
                                     XXXX b
□
        SACH   DXR                 ; send ACChi to initiate secondary protocol
□
        IDLE                       ; wait for interrupt
        SACL   DXR                 ; send the T register data
        IDLE                       ; wait for interrupt
        LACL   #0                  ; clear ACC=0
```

```
SACL   DXR                    ; send another to make sure 1st word got sent
IDLE                          ; wait for interrupt
SETC   INTM
RET
.END
```

เอกสารอ้างอิง

(1) กิตติพัฒน์ ต้นตระรุ่งโรจน์ , " สัญญาณและระบบเวลาเต็มหน่วย " , พิมพ์ดี , กรุงเทพฯ , พฤษภาคม , 2540

(2) ชวลิต เบญจางคประเสริฐ , อุทัย ศรีธีระวิโรจน์ และ กนก เจนจิระพงศ์เวช , " อะแดปทีฟอัลกอริทึมสำหรับการดีเทคสัญญาณซายน์คลื่นเดียวโดยใช้ฟิลเตอร์แบบปรับค่าคิวแฟกเตอร์ " , การประชุมวิชาการทางวิศวกรรมไฟฟ้าครั้งที่ 19 ภาควิชาวิศวกรรมไฟฟ้าคณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น, วันที่ 7-8 พฤศจิกายน 2539

(3) ชวลิต เบญจางคประเสริฐ , อุทัย ศรีธีระวิโรจน์ , กนก เจนจิระพงศ์เวช และ อรลาภ แสงอรุณ , " ซิมพลิไฟย์อะแดปทีฟอัลกอริทึมสำหรับการดีเทคสัญญาณซายน์คลื่นเดียวโดยใช้ฟิลเตอร์แบบปรับค่าคิวแฟกเตอร์ " , การประชุมวิชาการทางวิศวกรรมไฟฟ้าครั้งที่ 20 ภาควิชาวิศวกรรมไฟฟ้า จุฬาลงกรณ์มหาวิทยาลัย , พ.ศ.2540

(4) ชวลิต เบญจางคประเสริฐ , ราชู พันธ์ฉลาด , นภพินทุ์ อนันตรศิริชัย และ ประวิทย์ ชุมชู , " อะแดปทีฟ IIR แบนด์พาสฟิลเตอร์สำหรับการตรวจวัดสัญญาณซายน์ ด้วยวิธีการนอร์มอลไลซ์เกรเดียนต์ " , การประชุมวิชาการทางวิศวกรรมไฟฟ้าครั้งที่ 21 , พ.ศ. 2541

(5) สุธรรม ศรีเกษม , " MATLAB เพื่อการแก้ปัญหาทางวิศวกรรม " , ศรีอนันต์การพิมพ์ , กรุงเทพฯ

(6) อรรถสิทธิ์ หล่าสกุล , " Digital Signal Processing (DSP) Fundamentals "

(7) Widrow B.,Stearn S. , " Adaptive Signal Processing " , Macmillan Pentice-Hall , Inc.Englewood Cliffs , 1985

(8) Simon Haykin , " Introduction To Adaptive Filters " , Macmillan Publishing company , Newyork.