บทที่ 4

ขั้นตอนการดำเนินงาน

ความนำ

ในบทนี้ กล่าวถึงวิธีการทำงานในการจำลองโปรแกรมบนคอมพิวเตอร์และการพัฒนาเป็นการทำงานตามเวลาจริงบน DSK โดยกล่าวถึงจุดประสงค์ของการทดลองและรายละเอียดของการทำงานที่สำคัญในแต่ละขั้นตอน

4.1 การจำลองโปรแกรมบนคอมพิวเตอร์ส่วนบุคคล

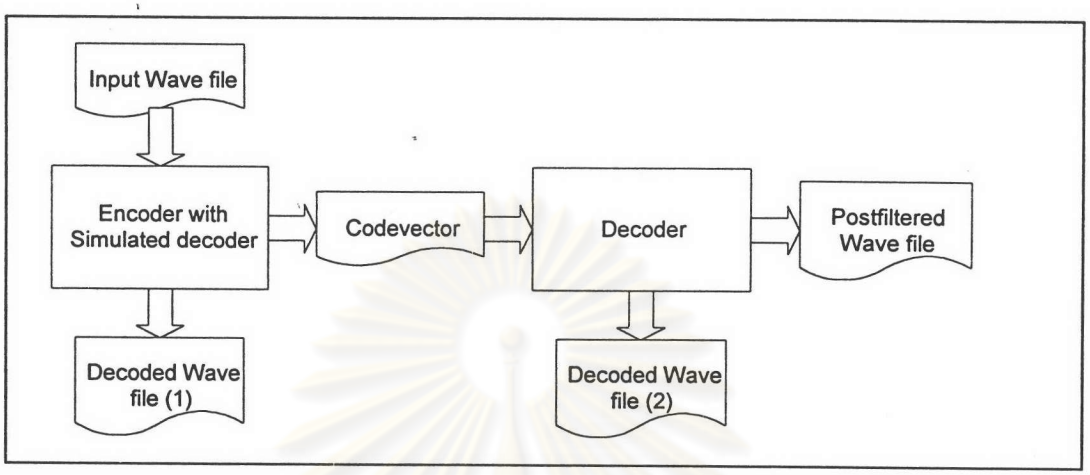
4.1.1 การจำลองโปรแกรมโดยใช้ MATLAB

วัตถุประสงค์ การเขียนโปรแกรมเพื่อทดสอบบนคอมพิวเตอร์ในขั้นนี้เป็นขั้นตอนแรกของการทำงานในการวิจัยครั้งนี้ มีจุดประสงค์เพื่อทำความเข้าใจกับหลักการทำงานของมาตรฐาน G.728 อย่างถูกต้องและแม่นยำเสียก่อน เพราะว่าการเขียนโปรแกรมโดยใช้ภาษาระดับสูงอย่าง MATLAB นั้นสามารถทำได้อย่างรวดเร็วและมีประสิทธิภาพมากกว่าการเขียนโปรแกรมโดยใช้ภาษาระดับล่างอย่างแอสเซมบลี และยังมีความถูกต้องของผลการคำนวณที่สูงกว่าเพราะมีความละเอียดของการเก็บข้อมูลที่สูงกว่า ผลที่ได้จะสามารถใช้เป็นเกณฑ์ในการเปรียบเทียบกับผลที่จะได้จากโปรแกรมที่เขียนบนแอสเซมบลีในภายหลัง เป็นการเปรียบเทียบความแตกต่างระหว่างการคำนวณแบบจุดลอยของ MATLAB และการคำนวณแบบจุดตรึงของแอสเซมบลีบนชิป TMS320C50

รายละเอียดของการทำงาน

ในขั้นนี้จะใช้โปรแกรมสำเร็จรูปที่สามารถทำการคำนวณทางคณิตศาสตร์ที่ซับซ้อนได้โดยง่าย คือ MATLAB รุ่น 4.2c.1 โดยทำการเขียนเป็นโปรแกรมย่อยหลายโปรแกรม ทำการทดสอบการทำงานของโปรแกรมย่อยทีละโปรแกรม แล้วจึงนำโปรแกรมย่อยเหล่านี้มารวมเป็นโปรแกรมใหญ่ที่สามารถทำการเข้ารหัสและถอดรหัสเสียงพูดได้ตามมาตรฐานG.728 เสียงพูดที่ใช้เป็นข้อมูลขาเข้าของโปรแกรมทำการอัดจากเสียงพูดหลาย ๆ ตัวอย่างโดยใช้ซอฟต์แวร์ที่มากับการ์ดเสียงทั่ว ๆ ไปเช่นเวฟสตูดิโอ (wavestudio) เสียงที่ใช้จะเป็นเสียงช่องสัญญาณเดี่ยว (mono) อัตราสุ่มข้อมูลอยู่ที่ 8 กิโลเฮิรตซ์ โดยใช้อัตราข้อมูล 16 บิตต่อหนึ่งสัญญาณ ข้อมูลขาออกของตัวเข้ารหัสจะได้ทั้งรหัสของเสียงที่ถูกเข้ารหัสแล้วและแฟ้มข้อมูลเสียงที่ถูกสังเคราะห์ขึ้นที่ตัวสังเคราะห์เสียงจำลองการถอดรหัสที่มีอยู่ภายในตัวเข้ารหัส รหัสเสียงที่ได้เมื่อนำไปผ่านตัวถอดรหัสจะได้แฟ้มข้อมูลเสียงที่ถูกถอดรหัสออกมาเหมือนกับแฟ้มข้อมูลเสียงที่ได้มาจากตัวเข้ารหัส แฟ้มเสียงที่ใช้ในการจำลองโปรแกรมนี้จะใช้รูปแบบการเก็บแบบแฟ้มนามสกุล .wav ซึ่งได้จากการอัดเสียงด้วยโปรแกรมเวฟสตูดิโอโดยมีอัตราของการสุ่มข้อมูลเท่ากับ 11.025 กิโลเฮิรตซ์ซึ่งสูงกว่าอัตราที่ต้องการในการวิจัยนี้ จึงต้องเขียนโปรแกรมเพื่อแปลงแฟ้มเสียงที่ได้ให้มีอัตราการสุ่มข้อมูลเป็น 8 กิโลเฮิรตซ์ โปรแกรมที่ใช้คือ w11to8.m ที่ทำงานบนแมทแลบ วิธีการทำงานของโปรแกรมคือแปลงข้อมูลให้มีอัตราการสุ่มสูงขึ้นไป 8 เท่าจะได้อัตราสุ่มเป็น 88.2 กิโลเฮิรตซ์แล้วจึงลดอัตราการสุ่มลงมา

11 เท่าก็จะได้อัตราการสุ่มข้อมูลของแฟ้มเสียงเป็น 8.018 กิโลเฮิรตซ์หรือประมาณ 8 กิโลเฮิรตซ์ตามที่ต้องการ รายละเอียดของโปรแกรมแสดงดังในภาคผนวก ง



รูป 4.1 บล็อกไดอะแกรมการทำงานของตัวเข้ารหัสและถอดรหัสบน MATLAB

การจำลองโปรแกรมบน MATLAB นี้ไม่ใช่การทำงานตามเวลาจริง ดังนั้นข้อมูลขาเข้าที่ใช้จึงต้องเป็นแฟ้มข้อมูลเสียงที่ได้จากการอัดเสียงเก็บไว้ล่วงหน้าแล้วจึงนำมาใช้กับโปรแกรมที่เขียนขึ้น เมื่อโปรแกรมทำงานเสร็จแล้วก็จะได้แฟ้มข้อมูลเสียงที่ผ่านการถอดรหัสออกมาจึงจะนำไปเปิดฟังโดยใช้การ์ดเสียงเพื่อฟังคุณภาพของแฟ้มข้อมูลเสียงขาออกเปรียบเทียบกับข้อมูลเสียงขาเข้าว่ามีความแตกต่างกันมากน้อยเพียงใด จากข้อแตกต่างหรือข้อผิดพลาดที่ได้จะนำไปเป็นแนวทางในการปรับปรุงโปรแกรมและเมื่อได้โปรแกรมที่ปรับปรุงเรียบร้อยแล้วจะทำการทดสอบกับแฟ้มข้อมูลเสียงเดิมเพื่อเปรียบเทียบผลอีกครั้ง ซึ่งขั้นตอนต่าง ๆ เหล่านี้ใช้เวลาค่อนข้างมากพอสมควรถึงแม้ว่าจะปรับปรุงโปรแกรมให้กระชับและทำงานได้รวดเร็วขึ้นมากแล้วเวลาที่ใช้ก็ยังคงมากเมื่อเปรียบเทียบกับคำว่าการทำงานตามเวลาจริง

4.1.2 การจำลองโปรแกรมโดยใช้ตัวจำลองโปรแกรม(simulator)

วัตถุประสงค์ ในส่วนนี้เป็นการเขียนโปรแกรมด้วยภาษาแอสเซมบลีของ C5X และทำการแปลภาษาด้วยชุดของโปรแกรม DSP รุ่น TI640 ที่ใช้กับชิป TMS320Cxx ซึ่งประกอบไปด้วยโปรแกรม dspa.exe เป็นตัวแอสเซมเบลอที่จะแปลงไฟล์จากไฟล์แอสเซมบลี(ASM file)มาเป็นไฟล์เป้าหมาย(object file หรือ OBJ file) ไฟล์เป้าหมายที่ได้เมื่อนำไปผ่านโปรแกรม dsplnk.exe ซึ่งเป็นตัวเชื่อมโปรแกรมเป้าหมาย ก็จะได้เป็นโปรแกรมที่สามารถทำงานได้จริง(executable COFF file หรือ OUT file)ซึ่งสามารถโหลดลงไปบนชิป C5X ได้ หรือสามารถจำลองการทำงานของโปรแกรมบนคอมพิวเตอร์ส่วนบุคคลได้โดยใช้โปรแกรม sim5x.exe สำหรับการทำงานบน เอ็มเอสดอส หรือ sim5xw.exe สำหรับการทำงานบนวินโดว์

ขั้นตอนของการทำงาน

1) เขียนเป็นโปรแกรมย่อยก่อนเพื่อดูการทำงานว่าสามารถทำงานได้ถูกต้องหรือไม่

2) จัดตำแหน่งของหน่วยความจำชนิดต่าง ๆ ได้แก่หน่วยความจำสำหรับโปรแกรม หน่วยความจำสำหรับข้อมูลให้เหมาะสม เพราะหน่วยจำแบบแรมที่มีอยู่นั้นมีขนาดจำกัดจำนวนเพียง 9 กิโลเวอร์ดสำหรับโปรแกรมและ 1 กิโลเวอร์ดสำหรับข้อมูล

3) รวมโปรแกรมย่อยเป็นโปรแกรมใหญ่ ทำการเขียนการเชื่อมต่อระหว่างโปรแกรมต่าง ๆ ให้เหมาะสม

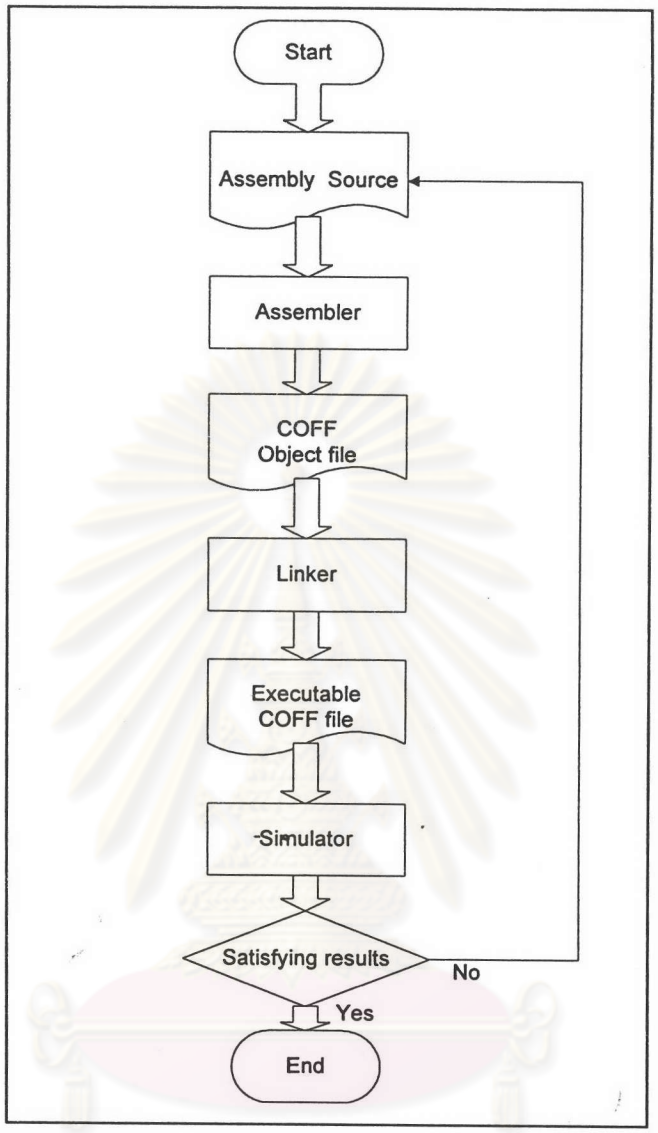
4) ทำการทดสอบโปรแกรมบนคอมพิวเตอร์ส่วนบุคคล เพื่อดูว่าสามารถทำงานได้ถูกต้องหรือไม่ รวมไปถึงคุณภาพของเสียงที่ได้ออกมาด้วย ในตอนแรกจะใช้สัญญาณขาเข้าเป็นคลื่นรูปซายน์ แล้วจึงเปลี่ยนไปใช้แฟ้มข้อมูลเสียงแบบเดียวกับที่ใช้ในการทดสอบโปรแกรมบน MATLAB

5) เก็บผลที่ได้จากการทดลองเข้ารหัสเสียงพูดและถอดรหัสออกมาและทำการประเมินผล

รายละเอียดของการทำงาน

วิธีการทำงานขั้นแรกเมื่อเขียนโปรแกรมเป็นไฟล์ภาษาแอสเซมบลีเสร็จแล้วก็จะทำการแอสเซมเบลอให้เป็นไฟล์เป้าหมาย และทำการเชื่อมต่อให้เป็นไฟล์ COFF แล้วจึงทำการจำลองการทำงานบนคอมพิวเตอร์ส่วนบุคคล เมื่อโปรแกรมทำงานเสร็จสิ้นแล้วจะมีการทดสอบผลที่ได้กับผลที่ต้องการ ถ้าผลที่ได้ยังไม่ดีพอก็จะย้อนกลับไปแก้ไขโปรแกรมใหม่จนกว่าจะได้ผลที่ตรงหรือใกล้เคียงกับที่ต้องการซึ่งใช้ผลจากการจำลองโปรแกรมบน MATLAB เป็นบรรทัดฐาน

การเขียนจะเริ่มต้นจากการเขียนทีละบล็อกย่อยก่อน บล็อกย่อยที่ประกอบกันเป็นตัวเข้ารหัสของงานวิจัยนี้มีทั้งหมด 23 บล็อก การตรวจสอบการทำงานของบล็อกทำได้โดยการตั้งเงื่อนไขเริ่มต้นของแต่ละบล็อกให้เหมือนกับที่ใช้ในโปรแกรมแต่ละบล็อกของ MATLAB จากนั้นจำลองการทำงานของแต่ละบล็อก เมื่อโปรแกรมทำงานเสร็จสิ้นแล้วก็ตรวจสอบผลที่ได้กับผลจาก MATLAB ถ้าผลไม่ถูกต้องก็ทำการแก้ไขโปรแกรมแล้วทดสอบใหม่ การทดสอบอาจจะสั่งให้ทำงานอย่างต่อเนื่องจนเสร็จสิ้นในครั้งเดียว หรือสั่งให้ทำงานทีละคำสั่ง(single step)แล้วตรวจสอบดูที่หน่วยประมวลผลที่รีจิสเตอร์ต่าง ๆ และในหน่วยความจำที่ใช้งานว่ามีการทำงานได้อย่างถูกต้องตามที่ต้องการหรือไม่ การมีโปรแกรมจำลองการทำงานทำให้การแก้ไขข้อผิดพลาดในโปรแกรมเป็นไปได้โดยง่าย เพราะทำให้เห็นการทำงานโดยละเอียดทุกขั้นตอน แต่โปรแกรมจำลองการทำงานมีข้อเสียคือทำงานได้ช้าถึงแม้จะใช้เครื่องคอมพิวเตอร์ที่มีความเร็วสูงพอสมควรเท่าที่สามารถหาได้ในปัจจุบันก็ยังใช้เวลาในการจำลองโปรแกรมนานกว่าการทำงานตามเวลาจริงเป็นอย่างมาก



รูป 4.2 แผนภาพแสดงขั้นตอนการทำงานในการพัฒนาโปรแกรมบนตัวจำลองโปรแกรม

การเขียนโปรแกรมแบบฟิกซ์พอยต์แอสเซมบลีมีสิ่งที่ต้องระวังเป็นสำคัญคือการจัดเก็บค่าลงหน่วยความจำจากหน่วยประมวลผล เพราะหน่วยประมวลผลมีความละเอียดถึง 32 บิตหรือ 2 เวอร์ดแต่หน่วยความจำมีค่าความละเอียดเพียง 16 บิตหรือ 1 เวอร์ดถ้าต้องการเก็บค่าเพียง 16 บิตออกจากค่าที่มีอยู่ทั้งหมด 32 บิตจะต้องรู้ขอบเขตของข้อมูลที่เก็บอยู่ใน 32 บิตนั้นจึงจะเลือกตำแหน่งที่เหมาะสมในการจัดเก็บออกมาได้ การจัดเก็บที่ไม่เหมาะสมอาจจะทำให้เกิดผลที่ไม่คาดคิดต่อการทำงานของโปรแกรมโดยรวม การจัดเก็บที่ไม่เหมาะสมอาจจะเป็นเรื่องความละเอียดของข้อมูลน้อยกว่า 16 บิตมาก ๆ เพราะมีการเผื่อขอบเขตบนของข้อมูลไว้มากเกินไป ในกรณีนี้โปรแกรมยังสามารถทำงานได้เป็นปรกติแต่คุณภาพของการทำงานจะต่ำลงหรืออาจจะเกิดปัญหาข้อมูลมีขนาดเล็กมากจนกลายเป็น 0 ได้ ปัญหาเช่นนี้จะไม่เกิดขึ้นเลยถ้าเป็นการทำงานบนหน่วยประมวลผลแบบโฟลทติ้งพอยต์

เมื่อแก้ปัญหาที่กล่าวมาข้างต้นได้แล้วก็เป็นการรวมเอาบล็อกย่อย ๆ ที่เขียนเอาไว้ทั้งหมดมารวมกันเป็นส่วนเข้ารหัส โดยมีการกำหนดลำดับการทำงานและช่วงเวลาที่ต้องทำงานของแต่ละบล็อกให้ถูกต้องตามมาตรฐาน รวมถึงการส่ง

ผ่านข้อมูลในหน่วยความจำระหว่างบล็อกต่างๆ ให้ถูกต้อง บางบล็อกอาจจะส่งผ่านข้อมูลกันทางรีจิสเตอร์ซึ่งจะช่วยลดเวลาในการอ่านเขียนหน่วยความจำลงไปได้

ผลที่ได้จากการจำลองโปรแกรมแบบแอสเซมบลีบนเครื่องคอมพิวเตอร์นี้จะแตกต่างจากผลบน MATLAB เล็กน้อย สาเหตุที่สำคัญคือเรื่องความละเอียดของข้อมูล บน MATLAB รูปแบบการเก็บข้อมูลจะเป็นแบบจุดลอยขนาด 64 บิตต่อหนึ่งข้อมูล แต่ในแอสเซมบลีการเก็บข้อมูลเก็บได้เพียง 16 บิตหรือหนึ่งเวอร์ดต่อหนึ่งข้อมูล การใช้จำนวนบิตมากกว่า 16 บิตในการเก็บข้อมูลบนแอสเซมบลีนั้นสามารถทำได้ แต่นอกจากจะเพิ่มจำนวนหน่วยความจำที่ต้องใช้แล้วยังเพิ่มจำนวนคำสั่งที่ต้องใช้ในการอ่านเขียนและทำการคำนวณกับตัวเลขที่เพิ่มขึ้นมา ทำให้โปรแกรมมีความซับซ้อนมากขึ้นโดยที่ได้ความถูกต้องของการคำนวณเพิ่มขึ้นเพียงเล็กน้อย

การกำหนดพื้นที่ของหน่วยความจำบน TMS320C50 สำหรับตัวเข้ารหัสของ LD-CELP ในงานวิจัยนี้จะมีการกำหนดแยกกันโดยเด็ดขาดว่าจะให้ส่วนใดเป็นส่วนโปรแกรม ส่วนข้อมูลหรือส่วนตาราง ที่อยู่บนหน่วยความจำของ C5x การกำหนดว่าจะให้ส่วนใดอยู่ที่ไหนนั้นจะต้องเขียนเป็นคอมมานด์ไฟล์(CMD file)ที่ระบุตำแหน่งของแต่ละส่วนเอาไว้ชัดเจน

คอมมานด์ไฟล์จะถูกเรียกใช้ในขณะที่ทำการเชื่อมต่อไฟล์เป้าหมายให้เป็นไฟล์ COFF โดยตัวเชื่อมต่อจะทำตามคำสั่งที่เขียนไว้ในคอมมานด์ไฟล์ทั้งหมด คอมมานด์ไฟล์อาจจะระบุชื่อของไฟล์เป้าหมายที่ต้องใช้ (มีได้มากกว่า 1 ไฟล์) ชื่อของไฟล์ COFF ที่ต้องการ จุดเริ่มต้นของโปรแกรม(entry point)ส่วนประกอบต่าง ๆ ในหน่วยความจำและตำแหน่งของส่วนต่าง ๆ ของโปรแกรมที่จะต้องเขียนลงบนหน่วยความจำ ซึ่งคอมมานด์ไฟล์ที่ใช้ในการทดลองนี้คือ encode.cmd ทำการแบ่งเนื้อที่หน่วยความจำออกเป็น 2 ส่วนใหญ่ ๆ ส่วนที่อยู่ในภาคโปรแกรมประกอบด้วยตัวโปรแกรม เวกเตอร์สำหรับเส้นทางบริการอินเตอร์รัปต์ และตารางของค่าคงที่ ส่วนที่อยู่ในภาคข้อมูลประกอบด้วยข้อมูลใน DARAM และ SARAM ดังนี้ -เวกเตอร์สำหรับเส้นทางบริการอินเตอร์รัปต์ เริ่มที่ตำแหน่ง 0800h ถึง 083fh

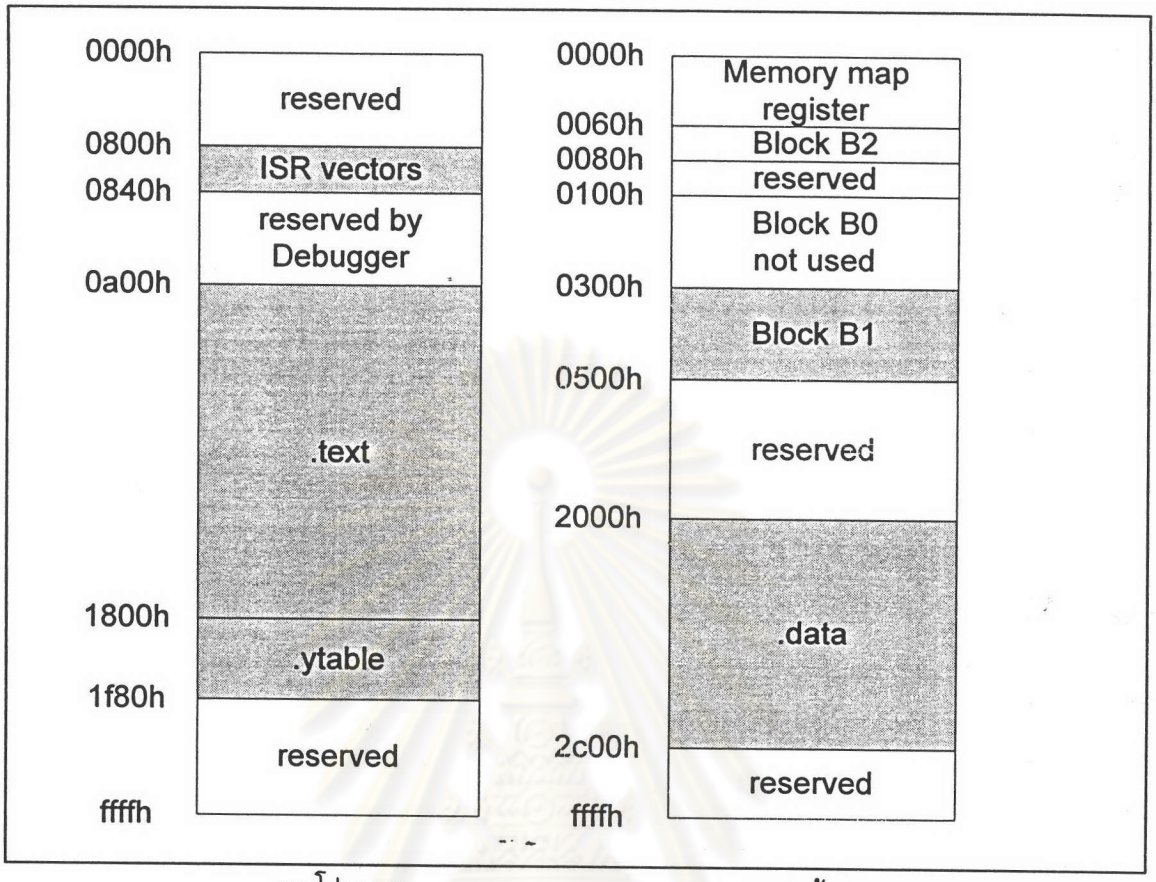
-โปรแกรม เริ่มที่ตำแหน่ง 0a00h ถึง 17ffh

-ตารางของค่าคงที่ เริ่มที่ตำแหน่ง 1800h ถึง 1f7fh

-ข้อมูลใน DARAM เริ่มที่ตำแหน่ง 0300h ถึง 04ffh

-ข้อมูลใน SARAM เริ่มที่ตำแหน่ง 2000h ถึง 2bffh

ขอบเขตที่กำหนดไว้นี้เป็นการแบ่งเนื้อที่เอาไว้ล่วงหน้า การเขียนโปรแกรมจริง ๆอาจจะไม่ได้ใช้เนื้อที่ทั้งหมดตามที่กำหนดไว้ ส่วนที่เหลืออาจจะได้ใช้สำหรับการปรับปรุงโปรแกรมในอนาคต



รูป 4.3 ตำแหน่งของส่วนต่าง ๆของโปรแกรมในหน่วยความจำของ 'C50 ที่ใช้ในตัวเข้ารหัส

การรับส่งข้อมูลเข้าออกระหว่างการจำลองโปรแกรมบนตัวจำลองโปรแกรม

ระหว่างการจำลองโปรแกรมบนตัวจำลองโปรแกรม โปรแกรมสามารถอ่านข้อมูลขึ้นมาหรือเขียนข้อมูลลงไปบนแฟ้มข้อมูลผ่านทางพอร์ทที่กำหนดไว้ได้ โดยจะต้องกำหนดชื่อไฟล์และพอร์ทที่จะใช้ติดต่อกับไฟล์นั้นไว้ในคอมมานด์ไฟล์ที่ตัวจำลองโปรแกรมจะเรียกใช้เมื่อเข้าสู่การจำลองโปรแกรม คอมมานด์ไฟล์นั้นชื่อว่า INIT.CMD เป็นไฟล์ที่กำหนดหน่วยความจำที่ใช้ได้ขณะที่มีการจำลองการทำงานของโปรแกรมและตำแหน่งของพอร์ทต่าง ๆ ที่ใช้ในการรับส่งข้อมูลและชื่อของไฟล์ที่จะติดต่อกับพอร์ทนั้น ในการทดลองนี้จะใช้พอร์ท 3 พอร์ท เป็นพอร์ทขาเข้า 1 พอร์ทและพอร์ทขาออก 2 พอร์ท โดยต่อเชื่อมกับไฟล์ต่าง ๆ ดังนี้

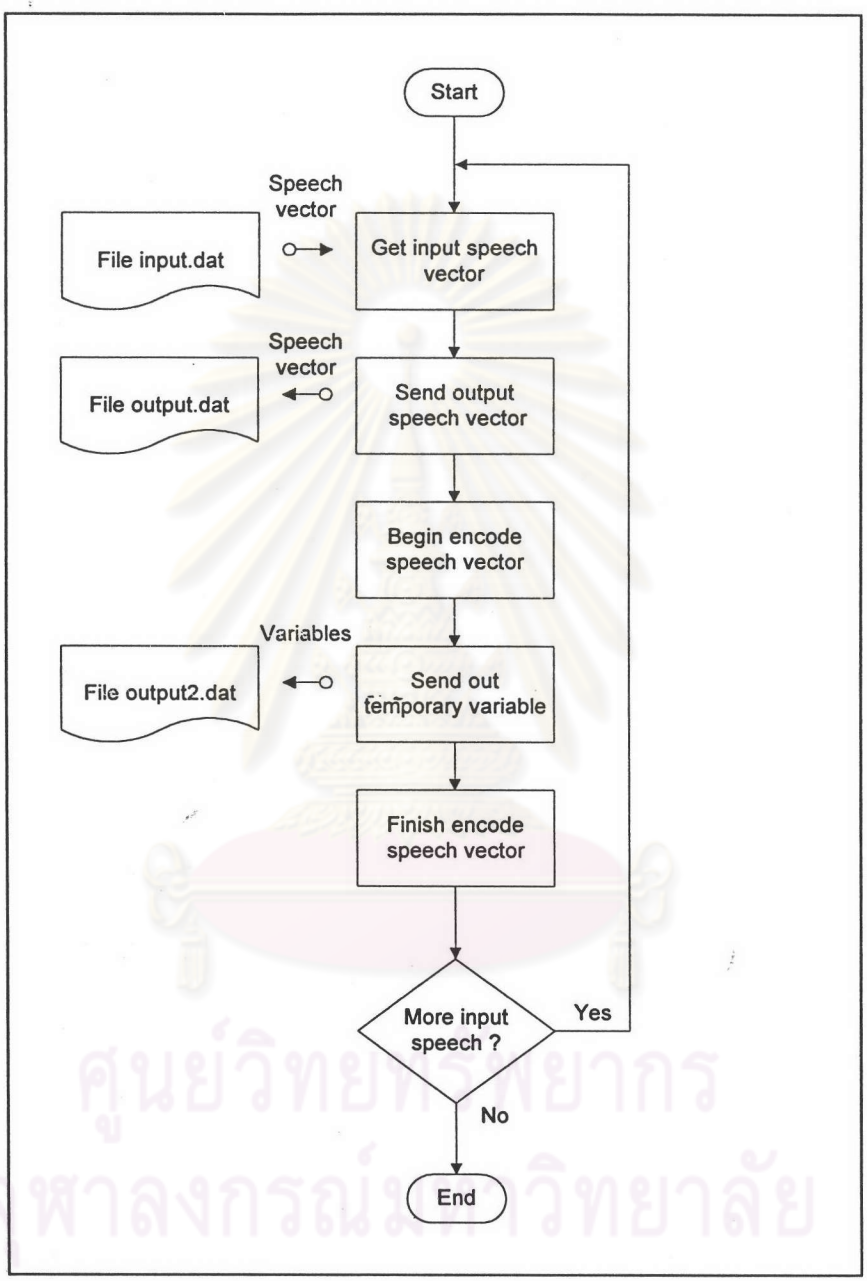
พอร์ท DRR ติดต่อกับไฟล์ in.dat

พอร์ท DXR ติดต่อกับไฟล์ out.dat

พอร์ท TDXR ติดต่อกับไฟล์ out2.dat

การจำลองโปรแกรมของตัวเข้ารหัส มีการอ่านข้อมูลจาก in.dat ผ่านทางพอร์ท DRR มาเก็บไว้ในบัฟเฟอร์ของสัญญาณขาเข้า ซึ่งเป็นเวกเตอร์ที่มีขนาด 5 ตัวอย่างสุ่ม เมื่อได้ข้อมูลเต็มบัฟเฟอร์แล้วก็จะเริ่มทำการเข้ารหัสเสียง ภายหลังการเข้ารหัสเสร็จสิ้นแล้วก็จะได้เวกเตอร์ข้อมูลเสียงที่ถูกสังเคราะห์ขึ้น(synthesized speech vector)ขนาด 5 ตัวอย่างสุ่มเช่นเดียวกันเก็บอยู่ในบัฟเฟอร์สัญญาณขาออก จากนั้นจะเป็นการอ่านข้อมูลเสียงขาออกเหล่านี้ผ่านทางพอร์ท DXR เพื่อ

เขียนลงในไฟล์ out.dat ส่วนพอร์ท TDXR นั้นใช้ในการรับส่งข้อมูลชนิดอื่น ๆ ที่เกิดขึ้นระหว่างการเข้ารหัสเพื่อนำมาใช้ในการแก้ไขความผิดพลาดของโปรแกรม



รูป 4.4 แผนผังแสดงการทำงานของตัวเข้ารหัสบนตัวจำลองโปรแกรม

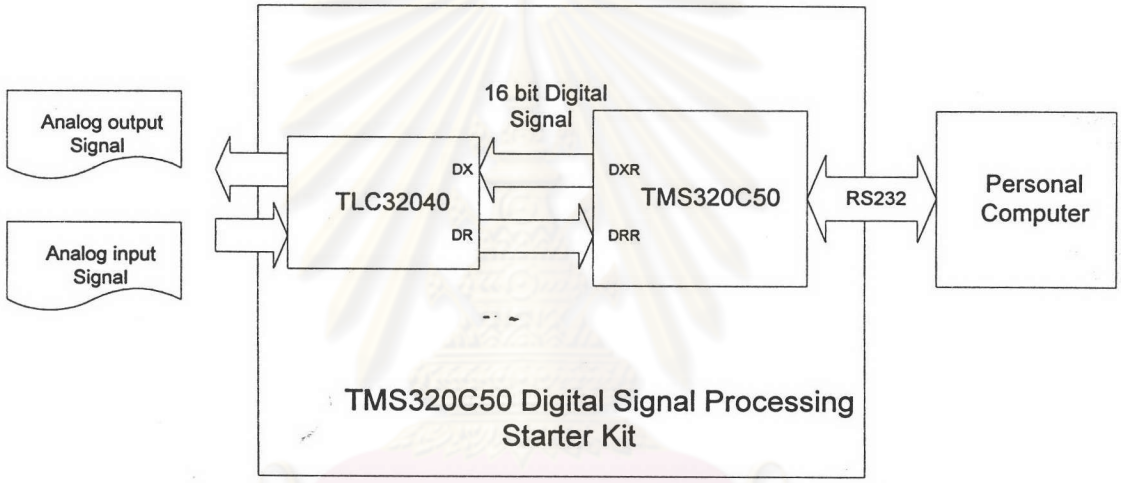
ข้อมูลจาก DAT ไฟล์จะอยู่ในรูปของตัวเลขฐาน 16 ซึ่งสามารถเขียนลงบนหน่วยความจำในตัวประมวลผลได้โดยตรง ได้มีการเขียนโปรแกรมบน MATLAB ชื่อ H2D.M ที่สามารถแปลงข้อมูลดังกล่าวให้เป็นขบวนของเลขฐาน 10 เพื่อให้ดูผลที่ได้ง่ายขึ้นจากการเขียนกราฟบน MATLAB และยังมีโปรแกรมชื่อ D2H.M ที่แปลงข้อมูลจากขบวนเลขฐาน 10 บน

MATLAB ให้เป็น DAT ไฟล์ได้ รายละเอียดการทำงานของโปรแกรมทั้งสองและคอมมานด์ไฟล์ที่ใช้ในการวิจัยนี้ดูได้ในภาคผนวก ง

4.2 การทำงานตามเวลาจริงบน TMS320C50 DSK

วัตถุประสงค์

การจำลองโปรแกรมบนคอมพิวเตอร์มีข้อด้อยในเรื่องเวลาในการทำงานที่นาน การเข้ารหัสเสียงเพียงไม่กี่นาทีอาจจะต้องใช้เวลาในการทำงานเป็นเวลาหลายชั่วโมง ทำให้การทดสอบการเข้ารหัสเสียงใช้เวลามาก การทำงานตามเวลาจริงจะช่วยให้การทดสอบการเข้ารหัสเสียงแบบต่าง ๆทำได้โดยรวดเร็ว มีรูปแบบการรับส่งข้อมูลดังในรูป 4.5



รูป 4.5 รูปแบบการรับส่งข้อมูลในการทำงานเข้ารหัสเสียงตามเวลาจริง

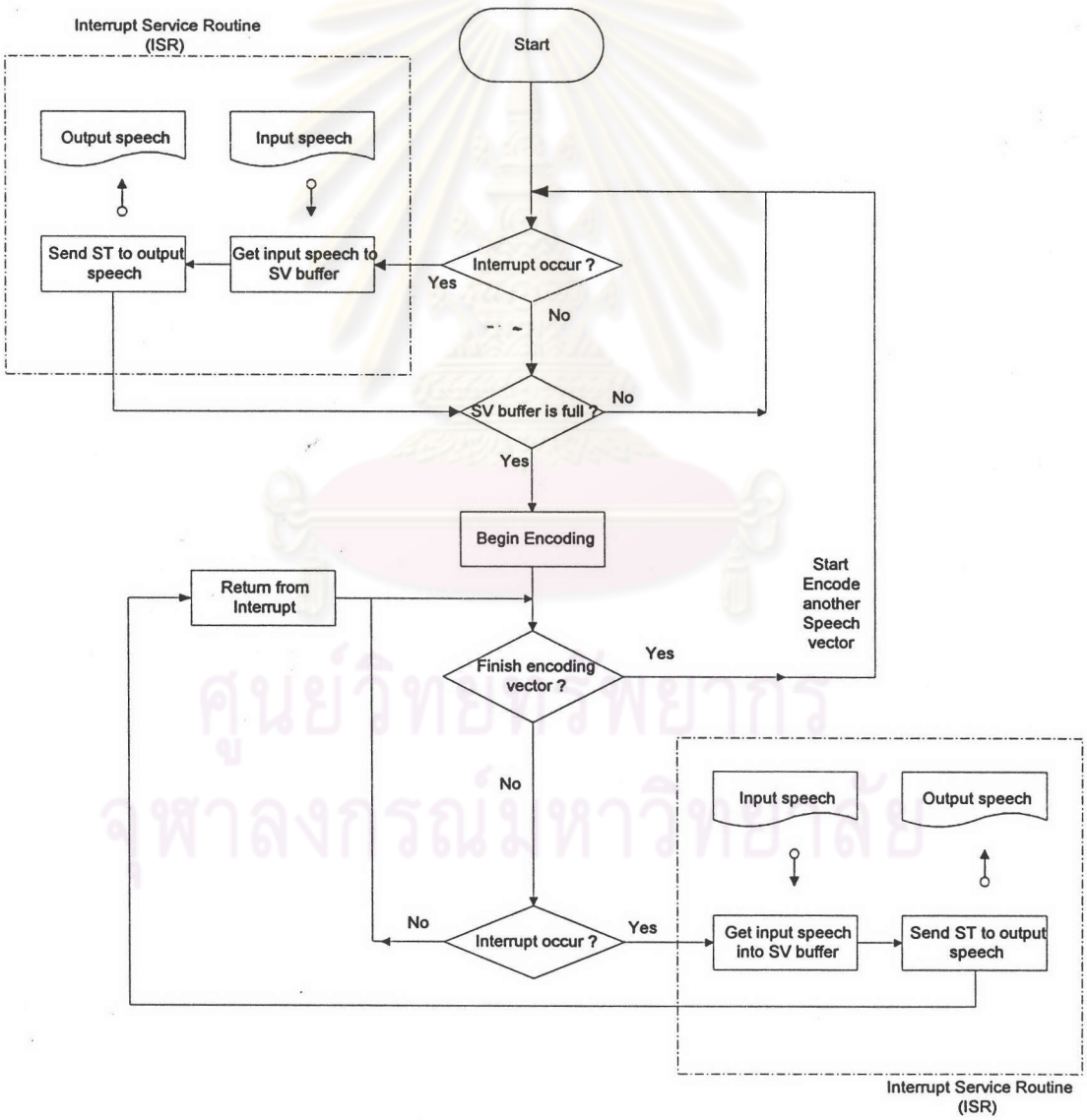
รายละเอียดของการทำงาน

เมื่อการจำลองการทำงานของโปรแกรมบนตัวจำลองโปรแกรมได้ผลตามที่ต้องการแล้ว โปรแกรมที่ได้เมื่อได้รับการดัดแปลงโปรแกรมเพียงเล็กน้อยก็เพียงพอที่จะให้โปรแกรมสามารถทำงานบน DSK ได้ เพราะว่าไฟล์ dsk5l.exe ของ DSK สามารถโหลด executable COFF ไฟล์(.out)ในการทำงานได้นอกเหนือไปจากไฟล์ .dsk ที่ออกแบบมาสำหรับ DSK โดยเฉพาะ เนื่องจากตัวแอสเซมเบลอของ DSK (dsk5a.exe) นั้นมีข้อจำกัดในการใช้งานหลายอย่าง ทำให้ไม่สามารถเขียนโปรแกรมขนาดใหญ่แบบที่ใช้ในงานวิจัยนี้ได้ และการใช้ตัวแอสเซมเบลอของ COFF ก็เป็นชนิดที่ใช้ในการจำลองโปรแกรมที่ผ่านมาแล้วด้วย ซึ่งสิ่งที่ต้องดัดแปลงมีดังนี้

1) การรับและการส่งค่าสัญญาณเสียงในการทำงานตามเวลาจริงสามารถทำได้เพียงครั้งละ 1 ตัวอย่างสุ่มเท่านั้น โปรแกรมในส่วนของการรับส่งค่าจะอยู่ในเส้นทางบริการอินเตอร์รัปต์(Interrupt Service Routine หรือ ISR) ซึ่งเป็นส่วนที่จะทำงานเมื่อมีสัญญาณจากวงจรเชื่อมต่อสัญญาณอนาลอก(Analog Interface Circuit หรือ AIC)เข้ามาบอกว่ามีอินเตอร์รัปต์เข้ามาโปรแกรมหลักของการเข้ารหัสที่ทำงานอยู่ในขณะนั้นจะหยุดทำงานชั่วขณะแล้วมาทำงานใน ISR แทน โดย

จะมีการอ่านข้อมูลที่ถูกส่งมาจากส่วนแปลงข้อมูลอนาลอกเป็นข้อมูลดิจิตอล (A/D) เข้ามาเก็บไว้ในบัฟเฟอร์สัญญาณเสียงขาเข้าในขณะเดียวกันก็ส่งข้อมูลที่อยู่ในบัฟเฟอร์สัญญาณเสียงขาออกออกไปสู่ส่วนแปลงข้อมูลดิจิตอลเป็นข้อมูล อนาลอก (D/A) เมื่อเสร็จแล้วก็กลับไปทำงานในโปรแกรมหลักที่หยุดเอาไว้ก่อนหน้านี้ต่อไป จนเมื่อโปรแกรมหลักทำงานเสร็จสิ้นแล้วจะมีการตรวจสอบว่าบัฟเฟอร์ของสัญญาณเสียงขาเข้าได้รับข้อมูลเข้ามาเต็มแล้วหรือยัง ถ้ายังไม่เต็มก็จะมีการรออินเตอร์รัปต์จนกว่าจะได้ข้อมูลเต็มบัฟเฟอร์แล้วจึงเริ่มการทำงานในโปรแกรมหลักรอบถัดไปต่อไป ดังแสดงไว้ในรูป 4.6

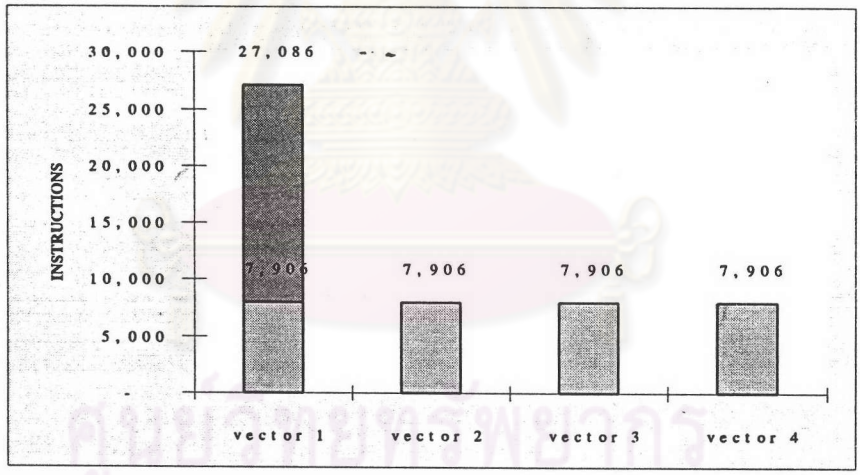
2) การติดต่อกับ AIC จะต้องมีการกำหนดการทำงานเริ่มต้นของ AIC (AIC initialization) ให้เหมาะสม เป็นการกำหนดอัตราการสุ่มข้อมูลและการเลือกใช้คุณสมบัติพิเศษอื่น ๆ ที่มีอยู่ในตัว AIC การไม่กำหนดการทำงานเริ่มต้นของ AIC ก่อนการทำงานจะทำให้ DSK ทำงานผิดพลาดไปจากที่ต้องการอย่างที่ไม่คาดคิด ซึ่งวิธีการกำหนดค่าเริ่มต้นให้แก่ AIC ดูได้ในภาคผนวก ค



รูป 4.6 แผนผังแสดงการทำงานของตัวเข้ารหัสสำหรับการทำงานตามเวลาจริง

3) การกระจายภาระ (Load Distribution) ระยะเวลาในการปรับค่าของพารามิเตอร์ใน G.728 มีทั้งแบบปรับทุกเวกเตอร์และแบบปรับทุกเฟรม การปรับค่าแบบปรับทุกเฟรมนั้นเพื่อให้ได้การทำงานที่ถูกต้องที่สุดจะต้องมีการปรับค่าใหม่ที่ก่อนการทำงานของเวกเตอร์แรกของเฟรม แต่ในการทำงานตามเวลาจริงไม่สามารถทำเช่นนั้นได้เพราะปริมาณของการคำนวณค่าพารามิเตอร์ใหม่ที่ต้องการนั้นมีปริมาณมาก เมื่อนำไปรวมกับการทำงานของเวกเตอร์ที่หนึ่งจะทำให้ปริมาณการคำนวณในเวกเตอร์ที่หนึ่งนั้นมีจำนวนสูงกว่าในเวกเตอร์ที่เหลือและระบบจะไม่สามารถทำงานตามเวลาจริงได้ ถ้าใช้อัตราการสุ่มข้อมูลที่ 8 กิโลเฮิรตซ์ปริมาณการคำนวณที่มากที่สุดสำหรับหนึ่งเวกเตอร์ในการทดลองนี้คือ 12,500 รอบการทำงาน

การคำนวณสำหรับส่วนปรับเปลี่ยนค่าพารามิเตอร์ทั้งหมดมีค่าประมาณ 19,000 คำสั่ง การคำนวณปกติในแต่ละเวกเตอร์มีค่าประมาณ 8,000 คำสั่ง เมื่อนำการคำนวณในส่วนปรับเปลี่ยนพารามิเตอร์ทั้งหมดมารวมกับเวกเตอร์ที่หนึ่งพบว่าการคำนวณมีค่าสูงมากเฉพาะในเวกเตอร์ที่หนึ่ง แต่ในเวกเตอร์อื่น ๆการคำนวณจะน้อยมากแสดงในรูปที่ 4.7 ตามมาตรฐาน G.728 ได้ออกแบบให้การปรับเปลี่ยนค่าของพารามิเตอร์สามารถกระจายการคำนวณไปยังเวกเตอร์ต่าง ๆได้ ทำให้ค่าของพารามิเตอร์ใหม่ที่จะใช้ในแต่ละเฟรมเกิดขึ้นที่เวกเตอร์ต่าง ๆกัน ค่าสัมประสิทธิ์ของวงจรกรองสังเคราะห์ 50 อันดับจะได้ค่าใหม่เมื่อเริ่มต้นเวกเตอร์ที่ 3 เช่นเดียวกับค่าสัมประสิทธิ์ของวงจรกรองเพิ่มน้ำหนักการรับฟังและค่าตารางพลังงานของชุดรหัสที่ใช้ในการค้นหาชุดรหัสที่ต้องการ ส่วนค่าสัมประสิทธิ์ของตัวทำนายอัตราขยายเชิงลอการิทึมจะได้ค่าใหม่เมื่อเริ่มต้นเวกเตอร์ที่ 2 จากมาตรฐานระบุว่าแม้ค่าใหม่ที่ได้จะไม่ได้เกิดขึ้นในเวกเตอร์ที่หนึ่งแต่คุณภาพของเสียงที่จะได้จากการเข้ารหัสก็ลดลงไปเพียงเล็กน้อยเท่านั้น

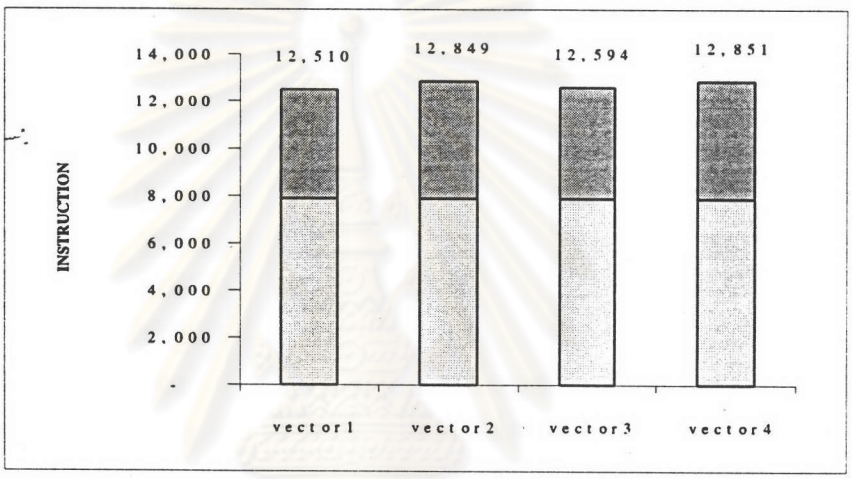


รูป 4.7 จำนวนคำสั่งที่ต้องคำนวณในแต่ละเวกเตอร์ของตัวเข้ารหัสก่อนการกระจายภาระ

เมื่อมีการกระจายบล็อกต่าง ๆของส่วนปรับเปลี่ยนพารามิเตอร์นี้ไปตามเวกเตอร์ต่าง ๆแล้ว ปริมาณการคำนวณในแต่ละเวกเตอร์ก็จะใกล้เคียงกันมีค่าสูงสุดอยู่ประมาณ 13,000 คำสั่งต่อเวกเตอร์เมื่อรวมส่วนปลีกย่อยอีกเล็กน้อยของโปรแกรมเข้าไปแล้ว แสดงในรูปที่ 4.8

แต่ปริมาณการคำนวณที่ได้เป็นการคำนวณสูงสุดที่เกิดขึ้นได้ในหนึ่งเวกเตอร์ปริมาณของการคำนวณปกติที่เกิดขึ้นจะน้อยกว่าค่านี้เพราะการทำงานของบล็อกค้นหาชุดรหัสที่ใช้ในการวิจัยนี้มีปริมาณการคำนวณที่ไม่คงที่ รูปแสดงการทำงานของบล็อกค้นหานี้แสดงในรูป 4.9 การค้นหาที่ใช้เป็นแบบผสมระหว่างการค้นหาแบบทั่วถึงหรือการค้นหาทั้งหมดกับการค้นหาแบบทรี(tree) ชุดรหัสขนาด 1,024 ตัวที่มีเก็บไว้ในตัวเข้ารหัสประกอบด้วยชุดรหัสรูปร่างจำนวน 128 ตัวโดยที่ชุด

รหัสรูปร่างแต่ละตัวจะจับคู่กับชุดรหัสอัตราขยายได้อีก 8 ตัว ในการค้นหาชุดรหัสจะเริ่มจากการเปรียบเทียบค่าที่มีกับชุดรหัสรูปร่างตัวแรกก่อนแล้วจึงเปรียบเทียบกับตัวต่อ ๆไปจนครบ 128 ตัว ระหว่างการเปรียบเทียบค่ากับชุดรหัสรูปร่างนั้นจะมีการพิจารณาว่าอัตราขยายที่ต้องใช้เป็นค่าบวกหรือลบซึ่งแบ่งออกเป็นค่าบวก 4 ค่าและค่าลบอีก 4 ค่าเมื่อทราบว่าเป็นค่าบวกหรือลบแล้วก็จะทำการค้นหาเฉพาะใน 4 ค่านั้นโดยการค้นหาอาจจะพบว่าอัตราขยายค่าแรกคือค่าที่ต้องการหรืออาจจะต้องค้นหาไปจนถึงค่าที่ 4 จึงจะได้ค่าที่ต้องการ ทำให้ปริมาณการคำนวณของบล็อกค้นหาชุดรหัสนี้ไม่คงที่ขึ้นกับว่าพบอัตราขยายที่ต้องการในแต่ละชุดรหัสรูปร่างเร็วหรือช้าเพียงใด ปริมาณการคำนวณสูงสุดของบล็อกนี้เกิดขึ้นเมื่อการค้นหาอัตราขยายต้องดำเนินไปถึงอัตราขยายตัวที่ 4 เสมอในทุก ๆชุดรหัสรูปร่างทั้งหมด 128 ตัวซึ่งโอกาสที่จะเกิดเหตุการณ์เช่นนี้ขึ้นมีค่อนข้างน้อยหรือคงไม่เกิดเหตุการณ์นี้ติดต่อกันเป็นเวลาหลาย ๆเวกเตอร์



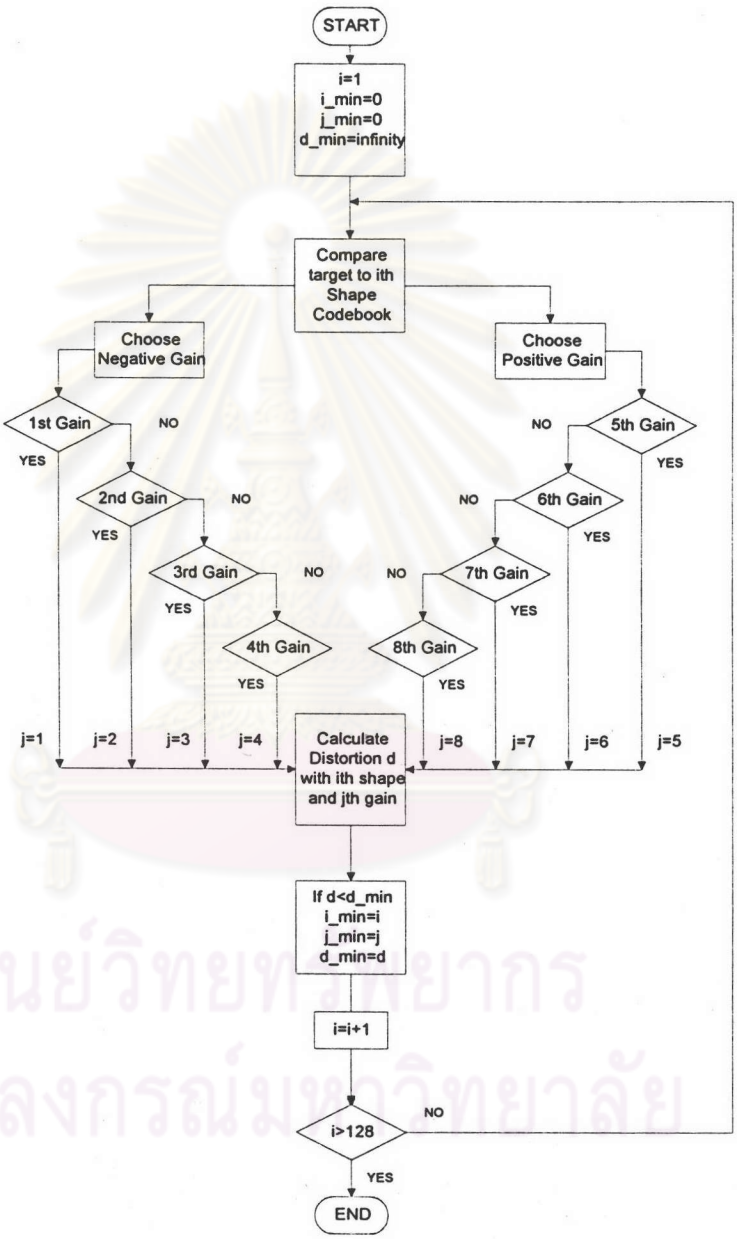
รูป 4.8 จำนวนคำสั่งที่ต้องคำนวณในแต่ละเวกเตอร์ของตัวเข้ารหัสหลังการกระจายภาระ

ตัวเข้ารหัสที่ได้จึงควรจะสามารถทำงานได้ทันเวลาจริงที่อัตราการสุ่มข้อมูลใกล้เคียงกับ 8 กิโลเฮิรตซ์โดยมีช่วงจังหวะเวลาที่ทำงานไม่ทันเวลาน้อยมาก และถ้าใช้อัตราการสุ่มข้อมูลประมาณ 7.2 กิโลเฮิรตซ์ตัวเข้ารหัสจะทำงานได้ทันเวลาจริงตลอดเวลาเพราะที่อัตราการสุ่มข้อมูลขนาดนี้ปริมาณการคำนวณที่มากที่สุดสำหรับหนึ่งเวกเตอร์จะได้เป็น 13,750 รอบการทำงานซึ่งมากเพียงพอแก่ความต้องการ

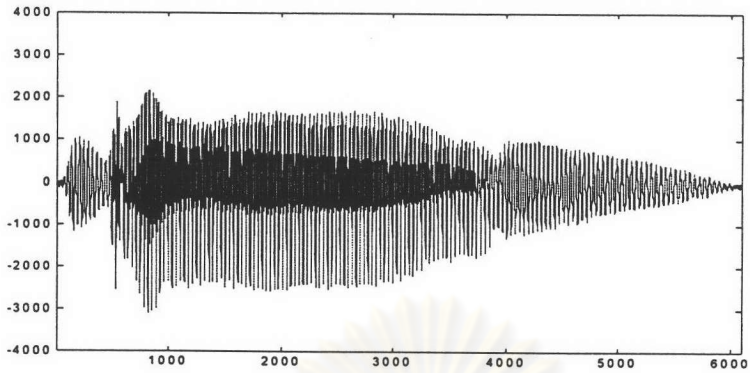
	ตัวประมวลผล	ตัวเข้ารหัส (Encoder)ของ G.728 LD-CELP		
		หน่วยความจำสำหรับโปรแกรม	หน่ายความจำสำหรับข้อมูล	MIPS
DSPSE	TMS320C5x	6.2 kwords	1.9 kwords	21.9
SASL	TMS320C5x	4 kwords	1 kwords	23.0
งานวิจัยนี้	TMS320C50	4.5 kwords	2 kwords	21

ตาราง 4.1 การเปรียบเทียบการคำนวณและหน่วยความจำที่ต้องการ

ตารางเปรียบเทียบปริมาณการคำนวณและหน่วยความจำที่ต้องการในงานวิจัยนี้กับงานของผู้วิจัยอื่นในการเขียนโปรแกรมตัวเข้ารหัสของ G.728 สำหรับการทำงานตามเวลาจริงด้วย TMS320C5x แสดงอยู่ในตาราง 4.1 และมีรายละเอียดอยู่ในภาคผนวก จ รายละเอียดของจำนวนคำสั่งที่ต้องใช้ในแต่ละบล็อกย่อยของตัวเข้ารหัสและเวกเตอร์ที่ทำงานรวมทั้งปริมาณการคำนวณใน 1 วินาทีของงานวิจัยนี้แสดงอยู่ในตารางที่ 4.2



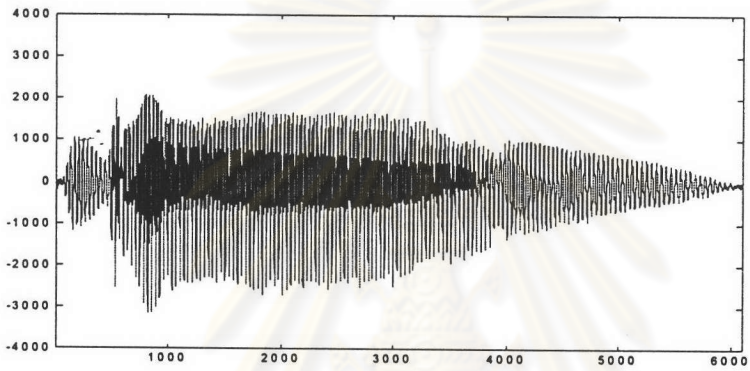
รูป 4.9 แผนภาพการทำงานของบล็อกค้นหาชุดรหัส



(ก) สัญญาณเสียงต้นฉบับ

แกนตั้ง = ขนาด

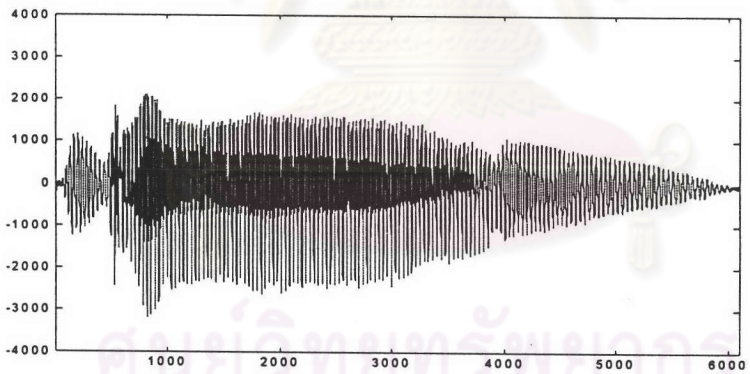
แกนนอน = ตัวอย่างสุ่ม



(ข) สัญญาณเสียงจากการเข้ารหัสบน MATLAB

แกนตั้ง = ขนาด

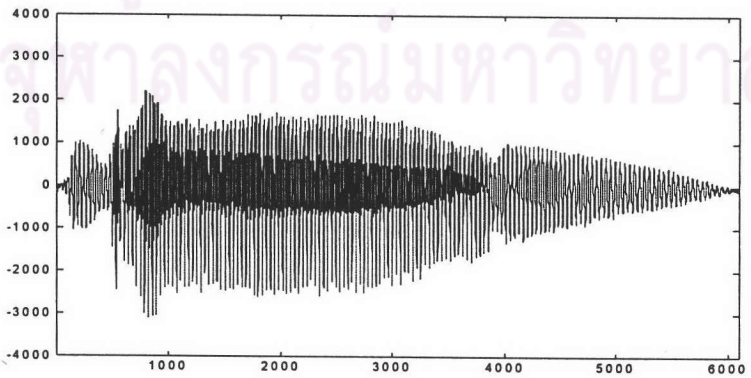
แกนนอน = ตัวอย่างสุ่ม



(ค) สัญญาณเสียงจากโพสต์ฟิลเตอร์บน MATLAB

แกนตั้ง = ขนาด

แกนนอน = ตัวอย่างสุ่ม



(ง) สัญญาณเสียงจากการเข้ารหัสบนแอสเซมบลี

แกนตั้ง = ขนาด

แกนนอน = ตัวอย่างสุ่ม

รูป ก.1 สัญญาณเสียงทั้งหมดของเสียงพูดคำว่า "เดิน" (ต่อ)

Block	Number of instruction	Times per frame	instructions per frame	v1	v2	v3	v4	vector 1	vector 2	vector 3	vector 4	MIPS
36	999	1	999		1			-	999	-	-	0.40
37	777	1	777			1		-	-	777	-	0.31
38	64	1	64			1		-	-	64	-	0.03
49	4,125	1	4,125				1	-	-	-	4,125	1.65
50_0	820	1	820				1	-	-	-	820	0.33
50_1	4,604	1	4,604	1				4,604	-	-	-	1.84
50_2	2,507	1	2,507		1			-	2,507	-	-	1.00
50_3	565	1	1,102			1		-	-	565	-	0.23
51	160	1	160			1		-	-	160	-	0.06
43	620	1	620		1			-	620	-	-	0.25
44	777	1	777		1			-	777	-	-	0.31
45	40	1	40		1			-	40	-	-	0.02
67	72	4	288	1	1	1	1	72	72	72	72	0.12
46	22	4	88	1	1	1	1	22	22	22	22	0.04
4748	33	4	132	1	1	1	1	33	33	33	33	0.05
4	231	4	924	1	1	1	1	231	231	231	231	0.37
9	308	4	1,232	1	1	1	1	308	308	308	308	0.49
10	223	4	892	1	1	1	1	223	223	223	223	0.36
11	25	4	100	1	1	1	1	25	25	25	25	0.04
12	236	1	236			1		-	-	236	-	0.09
1415	2,886	1	2,886			1		-	-	2,886	-	1.15
16	56	4	224	1	1	1	1	56	56	56	56	0.09
13	58	4	232	1	1	1	1	58	58	58	58	0.09
1718	6,553	4	26,212	1	1	1	1	6,553	6,553	6,553	6,553	10.48
1921	46	4	184	1	1	1	1	46	46	46	46	0.07
910	279	4	1,116	1	1	1	1	279	279	279	279	0.45
total of same instructions			31,624					7,906	7,906	7,906	7,906	12.65
total of different instructions			19,180					4,604	4,943	4,688	4,945	7.67
total			50,804					12,510	12,849	12,594	12,851	20.32

ตาราง 4.2 จำนวนคำสั่งที่ใช้ในการคำนวณของตัวเข้ารหัส