การกำจัดสัญญาณรบกวนในสัญญาณจำลองที่มีอัตราการชักสัญญาณต่ำด้วยวิธีการแมชชิ่งเพิชยูทบน FPGA

Denoising in Low Sampling Rate Simulated Signals based on the Matching Pursuit Method on an FPGA

ณัฐวีระ สงวนวงศ์

Natthaweera Sa-nguanwong

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญา
วิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า
มหาวิทยาลัยสงขลานครินทร์

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Engineering in Electrical Engineering
Prince of Songkla University
2553

ชื่อวิทยานิพนธ์ การกำจัดสัญญาณรบกวนในสัญญาณจำลองที่มีอัตราการชักสัญญาณต่ำด้วยวิธีการแมชชิงเพิชยูทบน FPGA

ผู้เขียน นายณัฐวีระ สงวนวงศ์

สาขาวิชา วิศวกรรมไฟฟ้า

อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

……...

(ผู้ช่วยศาสตราจารย์ ดร.ณัฏฐา จินดาเพ็ชร์)

คณะกรรมการสอบ

……...…ประธานกรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.พรชัย พฤกษ์ภัทรานนต์)

……..…กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.วรรณรัช สันติอมรทัต)

……..…กรรมการ

(รองศาสตราจารย์ ดร.วัฒนพงศ์ เกิดทองมี)

บัณฑิตวิทยาลัย มหาวิทยาลัยสงขลานครินทร์ อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษา ตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า

……...

(ศาสตราจารย์ ดร.อมรรัตน์ พงศ์ดารา)

คณบดีบัณฑิตวิทยาลัย

ชื่อวิทยานิพนธ์ การกำจัดสัญญาณรบกวนในสัญญาณจำลองที่มีอัตราการชักสัญญาณต่ำด้วยวิธีการแมชชิงเพิชยูทบน FPGA
ผู้เขียน นายณัฐวีระ สงวนวงศ์
สาขาวิชา วิศวกรรมไฟฟ้า
ปีการศึกษา 2553

บทคัดย่อ

วิทยานิพนธ์นี้นำเสนอสถาปัตยกรรมแบบขนานของกระบวนการแม็ชชิงเพิชยูท (Matching pursuit) สำหรับกำจัดสัญญาณรบกวนในสัญญาณจำลองที่มีอัตราการชักสัญญาณต่ำบน FPGA (Field Programmable gate array) กระบวนการแม็ชชิงเพิชยูททำการแยกสัญญาณจำลองออกเป็นอะตอมที่เหมาะสมโดยทำการโปรเจกชันสัญญาณจำลองไปยังอะตอมซึ่งอยู่ภายในดิกชันนารีแล้วได้สัญญาณองค์ประกอบที่มีพลังงานสูง ส่วนสัญญาณที่มีพลังงานต่ำถูกนำไปใช้เป็นสัญญาณในกระบวนการทำซ้ำรอบถัดไป กระบวนการนี้สามารถหยุดทำงานได้โดยกำหนดจำนวนรอบในการโปรเจกชัน การออกแบบกระบวนการแม็ชชิงเพิชยูทในรูปแบบขนานกันทำได้โดยการสร้างอะตอมจำนวน 1600 อะตอมและแบ่งอะตอมออกเป็นกลุ่มย่อยๆเรียกว่าดิกชันนารีจำนวน 40 กลุ่ม แล้วทำการโปรเจกชันสัญญาณจำลองไปยังอะตอมที่อยู่ดิกชันนารีทุกๆกลุ่มเพื่อหาค่าผลคูณภายใน จากนั้นผลคูณภายในแต่ละกลุ่มจะถูกนำมาเปรียบเทียบเพื่อหาผลคูณภายในสูงสุด ซึ่งกระบวนการแม็ชชิงเพิชยูทที่ทำงานแบบขนานกันนี้สามารถทำงานได้เร็วกว่าการทำงานในแบบทั่วไป เนื่องจากเป็นการลดเวลาการหาผลคูณภายในที่จากเดิมต้องทำทีละหนึ่งอะตอมมาเป็นการหาผลคูณภายในพร้อมๆกัน 40 อะตอม ในการทดสอบได้สร้างสัญญาณจำลองที่มีลักษณะคล้ายกับสัญญาณเสียงหัวใจรวมกับสัญญาณรบกวนด้วยโปรแกรมMATLABจากนั้นทำการแยกสัญญาณด้วยกระบวนการแม็ชชิงเพิชยูทที่ทำงานแบบขนานบน FPGA เบอร์ Xilinx Virtex5 XC5VSX95T พบว่าสามารถทำกระบวนการแยกสัญญาณรบกวนออกจากสัญญาณจำลองโดยการวนรอบจำนวน 10 รอบภายในเวลา 3.386 ms และมีความเร็วเป็น 338 เท่าเมื่อเทียบกับกระบวนการเดียวกันโดยใช้โปรแกรม MATLAB บนคอมพิวเตอร์ส่วนบุคคลที่มีความเร็วซีพียู 3.0 GHz และหน่วยความจำ 4 GB

คำสำคัญ แมชชิงเพิชยูท FPGA

Thesis Title	Denoising in Low Sampling Rate Simulated Signals based on the Matching Pursuit Method on an FPGA
Author	Mr.Natthaweera Sa-nguanwong
Major Program	Electrical Engineering
Academic Year	2010

ABSTRACT

This thesis presents a parallel architecture of the matching pursuit algorithm for denoising in low sampling rate simulated signals on an FPGA (Field Programmable Gate Array). The matching pursuit separates the signal into appropriate atoms by projecting it onto atoms in dictionaries to obtain the high energy signal. The low energy signal will be the main signal in the next round. The repeat process can be stopped by assigning the number of rounds. The parallel matching pursuit is performed by dividing 1600 atoms into 40 groups called dictionaries and finding inner product by projecting signal onto atoms in every group. After that, inner products from every group are compared to find the maximum inner product again. This parallel process is faster than sequential process because it reduces processing time by finding 40 inner products simultaneously. In simulations, the signals were generated to have the feature like heart signals mixed with a random noise. Then, the signals were separated by the designed parallel architecture on an FPGA , Virtex5 XC5VSX95T. The results show that the designed parallel architecture of matching pursuit can separate noise from simulated signal by repeating the process 10 rounds within 3.386 ms which is 338 time faster than the processing by MATLAB on a personal computer with 3.00 GHz and 4 GB RAM.

Keywords: Matching pursuit method , FPGA

สารบัญ

สารบัญ (ต่อ)

รายการตาราง

รายการภาพประกอบ

รายการภาพประกอบ (ต่อ)

บทที่ 1

บทนำ

1.1 ความสำคัญและที่มาของงานวิจัย

แม็ชชิงเพิชยูท เป็นวิธีการแยกสัญญาณให้อยู่ในรูปของ linear expansion โดยอาศัยการโปรเจคชัน(projection) สัญญาณที่ต้องการวิเคราะห์ลงบนอะตอม (atom) ที่ถูกสร้างขึ้นและเก็บไว้ในดิกชันนารี(dictionary) ซึ่งวิธีการวิเคราะห์สัญญาณโดยกระบวนการแม็ชชิงเพิชยูทมีการนำไปประยุกต์ใช้กับการบีบอัดข้อมูล (data compression) การจดจำรูปแบบ (pattern recognition) และการวิเคราะห์-สังเคราะห์สัญญาณเสียง แต่เนื่องจากวิธีการนี้เป็นวิธีการที่ใช้เวลาในการทำงานนาน เป็นเพราะต้องทำการหาอะตอมที่ให้ผลคูณภายใน(inner product) มีค่ามากที่สุด ดังนั้นวิธีการวิเคราะห์สัญญาณแบบนี้จึงมักจะใช้งานบนคอมพิวเตอร์ โดยในทางปฏิบัติเราสามารถเพิ่มประสิทธิภาพการวิเคราะห์สัญญาณโดยวิธีการแม็ชชิงเพิชยูทได้โดยการออกแบบดิกชันนารี , ออกแบบการค้นหาอะตอมที่มีผลคูณภายในมากที่สุด หรือออกแบบโครงสร้างการทำงานในรูปแบบขนานกัน ซึ่งรูปแบบการทำงานแบบขนานกันนั้น ผู้ดำเนินการวิจัยพบว่าเป็นจุดเด่นข้อหนึ่งที่พบได้ใน FPGA และภายใน FPGA นั้นมีคุณสมบัติคล้ายกับคอมพิวเตอร์ ดังนั้นในโครงงานวิจัยนี้จึงเป็นการออกแบบกระบวนการแม็ชชิงเพิชยูท โดยเป็นการทำงานแบบขนานกันภายใน FPGA

1.2 การทบทวนเอกสาร

1. รุ่งลาวัลย์ วิไลรัตน์. 2547. การวิเคราะห์กราฟเสียงเต้นหัวใจด้วยวิธีการแมชชิงเพิชยูท. วิทยานิพนธ์วิศวกรรมศาสตรมหาบัณฑิต มหาวิทยาลัยสงขลานครินทร์

นำเสนอการวิเคราะห์กราฟเสียงเต้นหัวใจ (Phonocadiogram) โดยใช้วิธีการแมชชิงเพิชยูท ซึ่งเป็นวิเคราะห์สัญญาณเพื่อให้ได้องค์ประกอบของสัญญาณเชิงความถี่ (frequency content) และตำแหน่งเวลาที่เกิด (time localization)

วิทยานิพนธ์นี้ได้พัฒนากระบวนการแมชชิงเพิชยูทโดยใช้โปรแกรม MATLAB และทำการวิเคราะห์กราฟเสียงเต้นหัวใจระหว่างคนปกติและผู้ป่วยโรคหัวใจที่เก็บไว้ในคอมพิวเตอร์จำนวน 12 ตัวอย่าง ซึ่งวิธีการแมชชิงเพิชยูทสามารถแสดงองค์ประกอบเชิงความถี่และตำแหน่งเวลาที่เกิดเสียงได้สอดคล้องกราฟเสียงเต้นหัวใจ

2.Xuan Zhang , Louis-Gilles Durand. "Analysis-synthesis of the phonocadiogram base on the matching pursuit method" , IEEE Transactions on biomedical engineering , vol 45 , no 8 , August 1998

ในบทความนี้นำเสนอการสังเคราะห์และวิเคราะห์เสียงหัวใจโดยใช้กระบวนการแม็ชชิงเพิชยูทซึ่งเป็นวิธีการที่มาจาก gabor wavelet หรือที่เรียกว่า time-frequency atom ซึ่งสร้างสัญญาณ sinusoid และ Gaussian window function

ในบทความนี้ได้ทดลองกับสัญญาณเสียงหัวใจที่ไม่มีสัญญาณรบกวนและสัญญาณหัวใจที่มีเสียงรบกวนขนาด 10% (energy) จำนวน 11 ตัวอย่าง พบว่าวิธีการนี้สามารถกำจัดสัญญาณรบกวนออกจากสัญญาณเสียงหัวใจได้ และให้ค่า NRMSE อยู่ที่ 2.2 %

3.StephaneG. Mallat ,Zhifeng Zhang, " Matching pursuit with time-frequency dictionaries", IEEE Trans. Signal processing. Vol.41 , pp.3397-3415 , 1993

บทความนี้นำเสนอกระบวนการแม็ชชิงเพิชยูท ซึ่งเป็นการแยกสัญญาณใดๆแบบ Linear expansion โดยวิธีการนี้จะต้องใช้ดิกชันนารี (Dictionary) ซึ่งประกอบไปด้วยกลุ่มของฟังก์ชันที่เรียกว่า Time-frequency atoms ซึ่งในการวิเคราะห์หรือการแยกสัญญาณนั้น ทำได้โดยการโปรเจกชัน (Projection) สัญญาณซ้ำๆ ไปยังอะตอม หรือ Time – frequency atoms ของฟังก์ชันดิกชันนารี (Function dictionary) ซึ่งอะตอมนั้นมีความสัมพันธ์กับสัญญาณในลักษณะที่ทำให้ขนาดของ Inner product ระหว่างอะตอมกับสัญญาณที่นำมาวิเคราะห์มีค่าสูงสุด ซึ่งในการสร้างดิกชันนารีจะทำได้โดยการสเกล(scaling) ทรานสเลท (translate) มอดูเลท (modulate) วินโดว์ฟังก์ชัน หรือเรียกว่า Gaussian window function

4. Sacha Krstulovi´c and R´emi Gribonval. "MPTK: MATCHING PURSUIT MADE TRACTABLE", Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on, 14-19 May 2006

นำเสนอวิธีการออกแบบซอฟแวร์ที่มีชื่อว่า MPTK (Matching pursuit tool kit)ซึ่งเป็นซอร์ฟแวร์ที่มีการทำกระบวนการแม็ชชิงเพิชยูทแบบขนาน ซึ่งทำให้ลดความซับซ้อนในการคำนวณ และทดลองประยุกต์ใช้กับสัญญาณเสียง ซึ่งสามารถลดความซับซ้อนในการทำงานจาก $O(N^2)$ จนเหลือเพียง $O(N \log N)$ ครั้ง และใช้เวลาทำงานเพียง หนึ่งในสี่เท่าของเวลาที่บันทึกจากสัญญาณเดิม

5. Gang Xu , Jing Meng, "Signal enhancement with matching pursuit", Vehicular Technology Conference 2004. VTC2004-Fall.2004 IEEE 60th Vol 3 26-29 Sept.2004. p1986-1990

ในบทความนี้นำเสนอการนำกระบวนการวิเคราะห์สัญญาณโดยกระบวนการแม็ชชิงเพิชยูทมาใช้ในการกำจัดสัญญาณรบกวน(noise reduction) ซึ่งวิธีการกำจัดสัญญาณรบกวนโดยใช้วงจรกรอง(filter)แบบต่างๆ จำเป็นที่จะต้องทราบถึงลักษณะของสัญญาณและสัญญาณรบกวนก่อนจึงจะสามารถเลือกใช้วงจรกรองเพื่อให้ผลที่ถูกต้องได้

แต่กระบวนการแม็ชชิงเพิชยูทนั้น จะทำงานคล้ายกับกระบวนการที่ปรับตัวได้(adaptive algorithm) โดยที่จะแยกสัญญาณให้อยู่ในรูปของ linear weighed expansion ของอะตอมที่สร้างขึ้น และหลักการในการแยกสัญญาณนั้นจะมองว่า สัญญาณหลักจะมีค่า coherence ที่สูง และสัญญาณรบกวนจะมีค่า coherence ต่ำ ซึ่งกระบวนการกำจัดสัญญาณรบกวนนี้จะดึงเอาเฉพาะกลุ่มสัญญาณที่มีค่า coherence สูงออกมา

ในบทความฉบับนี้ได้ทำการทดลองกำจัดสัญญาณรบกวนเปรียบเทียบกับวิธีการ wavelet soft threshold ซึ่งพบว่าวิธีการแม็ชชิงเพิชยูทสามารถกำจัดสัญญาณรบกวนได้ดีกว่า เมื่อค่า SNR ของสัญญาณมีค่าต่ำและค่าความเบี่ยงเบนเมื่อนำมาให้กำจัดสัญญาณรบกวนจากสัญญาณ Doppler (Doppler signal) น้อยกว่าวิธีการ wavelet soft threshold

6. Yan Meng , Andrew P.Brown , Ronald A.IItis. "MP Core : Algorithm and Design Techniques for Efficient Channel Estimation in Wireless Applications" Proceeding. 42nd.Design Automation Conference , 2005.

นำเสนอการออกแบบกระบวนการแม็ชชิงเพิชยูทโดยใช้ FPGA เพื่อใช้ในการประมวลผลช่องสัญญาณ (channel estimation) และ multiuser detection ในเครือข่ายไร้สาย ซึ่งพบว่าเมื่อนำกระบวนการนี้ไปใช้งานบน FPGA สามารถทำงานได้เร็วกว่าคอมพิวเตอร์

1.3 วัตถุประสงค์โครงการ

เพื่อศึกษาและพัฒนาโครงสร้างกระบวนการกำจัดสัญญาณรบกวนบนในสัญญาณที่มีอัตราการชักสัญญาณต่ำด้วยวิธีการแม็ชชิงเพิชยูท บน FPGA

1.4 ขอบเขตงานวิจัย

จำลองการทำงานการกำจัดสัญญาณรบกวนบนในสัญญาณที่มีอัตราการชักสัญญาณต่ำด้วยวิธีการแม็ชชิงเพิชยูทบน FPGA ได้

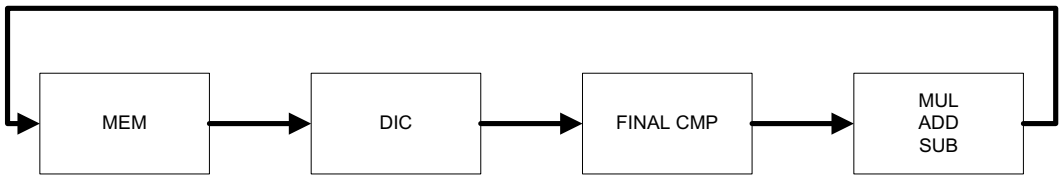
1.5 ระเบียบวิธีวิจัย

1.5.1 สัญญาณ

สัญญาณที่นำมาใช้ในการทดลองเป็นสัญญาณจำลองที่มีความยาว 40 ตัวอย่าง โดยออกแบบให้มีลักษณะคล้ายกับสัญญาณเสียงหัวใจ พร้อมทั้งเพิ่มสัญญาณรบกวนชนิด random noise ความยาว 40 ตัวอย่าง

1.5.2 การออกแบบ

การออกแบบกระบวนการแม็ชชิงเพิชยูทบน FPGA แบ่งออกเป็น 4 ส่วนหลักๆ ดังภาพประกอบที่ 1-1



ภาพประกอบที่ 1-1 แสดงการออกแบบโดยรวมของกระบวนการแม็ชชิงเพิชยูทบน FPGA

ในการออกแบบกระบวนการนี้บน FPGA สามารถแยกได้เป็น 4 ส่วน หลักๆ คือ

- โมดูล MEM เป็นโมดูลที่ทำหน้าที่เก็บสัญญาณตั้งต้นในรอบแรก และเก็บผลลัพธ์เพื่อใช้ในการคำนวณในรอบถัดไป
- โมดูล DIC เป็นโมดูลที่ทำหน้าที่เก็บอะตอมเพื่อใช้ในการคำนวณพร้อมทั้งยังประกอบไปด้วยโมดูลย่อยที่ใช้ในการหาค่าผลคูณภายในสูงสุดซึ่งใน FPGA นี้ จะประกอบไปด้วยโมดูลนี้ 40 ตัว ซึ่งทุกตัวจะทำงานขนานกัน
- โมดูล FINAL CMP เป็นโมดูลที่ทำหน้าที่หาค่าผลคูณภายในสูงสุดที่ได้จากการคำนวณของโมดูล DIC
- โมดูล MUL ADD SUB เป็นโมดูลที่ใช้หาสัญญาณตั้งต้นในรอบใหม่ และหาสัญญาณใหม่ซึ่งผลลัพธ์ที่ได้จะนำไปเก็บในโมดูล MEM เพื่อใช้ในรอบถัดไป

ผลลัพธ์ที่ได้จากกระบวนการแม็ชชิงเพิชยูทบน FPGA จะถูกนำมาเปรียบเทียบกับกระบวนการแม็ชชิงเพิชยูทที่ออกแบบบน MATLAB

1.5.3 การวิเคราะห์

ในการวิเคราะห์ผลที่ได้จากกระบวนการแม็ชชิงเพิชยูทบน FPGA และ MATLAB จะใช้การเปรียบเทียบสัญญาณที่สร้างขึ้นมาใหม่ที่ได้จากทั้งสองกระบวนการและเปรียบเทียบเวลาที่ใช้ไปจากทั้งสองกระบวนการ

บทที่ 2

ทฤษฎี

2.1กระบวนการแม็ชชิงเพิชยูท (Matching Pursuit Process)

การวิเคราะห์สัญญาณด้วยวิธีการแม็ชชิงเพิชยูท เป็นกระบวนการแยกสัญญาณใดๆแบบ Linear expansion โดยมองสัญญาณตั้งต้นเกิดมาจากสัญญาณย่อยๆหลายสัญญาณรวมกัน ซึ่งวิธีการนี้จะต้องใช้ดิกชันนารี (dictionary) ซึ่งประกอบไปด้วยกลุ่มของสัญญาณที่เรียกว่าอะตอม(atom) หรือในบางครั้งเรียกว่า Time-frequency atoms และในการวิเคราะห์หรือแยกสัญญาณนั้น ทำได้โดยการโปรเจกชัน (Projection) ซึ่งเป็นหาค่าผลคูณภายในระหว่างสัญญาณตั้งต้นกับอะตอม และโปรเจกชันสัญญาณซ้ำๆไปยังกลุ่มของอะตอมที่เรียกว่าดิกชันนารี ซึ่งอะตอมนั้นมีความสัมพันธ์กับสัญญาณในลักษณะที่ทำให้ขนาดของผลคูณภายในระหว่างอะตอมกับสัญญาณที่นำมาวิเคราะห์มีค่าสูงสุด และสำหรับการสร้างดิกชันนารี หรือเรียกว่า “รีดันแดนซ์ดิกชันนารี”(Redundant dictionary) ทำได้โดยการสเกล (scaling) ทรานสเลท (translating) และมอดูเลท (modulating) วินโดว์ฟังก์ชัน (Window function: $h(t)$) ซึ่งเป็นผลมาจาก wavelet envelope function $g(t)$ และ sinusoid function $u(t)$ [1]

เราสามารถแสดงสัญญาณต่างๆของวิธีการแม็ชชิงเพิชยูท ได้ดังนี้

$$x(t) = \sum_{i=0} a_i \; h_i(t) \qquad (2.1)$$

$$h_i(t) = {}_i \; g_i(t) \; u_i(t) \qquad (2.2)$$

$$g_i(t) = 2^{1/4} e^{\;(t \;\; p_i/s_i)^2} \qquad (2.3)$$

$$u_i(t) = \cos(2 \; f_i t \quad {}_i) \qquad (2.4)$$

โดยที่

$x(t)$ คือ สัญญาณใดๆที่ต้องการวิเคราะห์

$h(t)$ คือ Window function อะตอมใดๆที่ถูกสร้างขึ้นด้วยGaussian window function $g(t)$ และ Sinusoid function $u(t)$ และเก็บไว้ในดิกชันนารี

$g_i(t)$ คือ Gaussian window function

a_i คือ สัมประสิทธิ์การยืดขยาย โดยที่ค่ากำลังสองของ a_i จะบ่งบอกถึงพลังงานของอะตอม $h_i(t)$

s_i คือ Scale factor ซึ่งทำหน้าที่ควบคุมความยาวของกลุ่มอะตอม $h_i(t)$

p_i คือ ตัวเจาะจงตำแหน่ง ทำหน้าที่ควบคุมตำแหน่งของกลุ่มอะตอม $h_i(t)$

$_i$ คือ normalize factor ทำหน้าที่ทำให้ $\|h_i(t)\| = 1$

f_i คือ ความถี่

$_i$ คือ เฟส

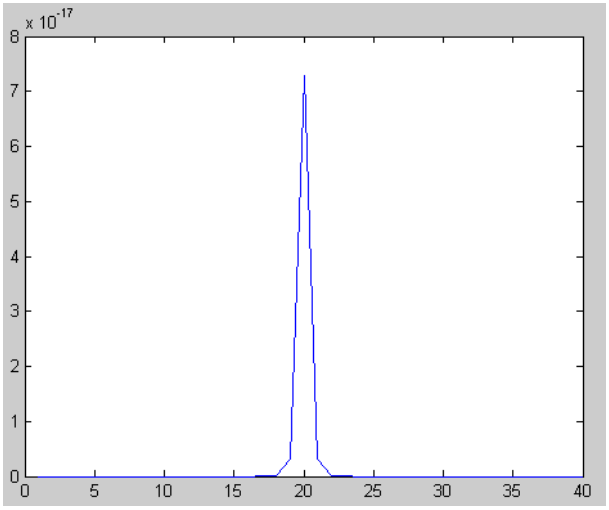
จากสมการที่ (2.1) ถึง (2.4) ค่าพารามิเตอร์ทั้งหมดจะถูกเก็บในรูปของ $_i$ โดยที่ซึ่งจะได้เป็น $_i = (\ _i, s_i, p_i, f_i, \ _i)$

จากสมการที่ (2.3) และ(2.4) เมื่อกำหนดให้ $f = \frac{2\quad k}{N}$ จะได้ the discrete window function ตามสมการที่ (2.5)

$$h(n) = \quad g(\frac{n\quad p}{s})\cos(\frac{2\quad k}{N}\ n\quad) \qquad (2.5)$$

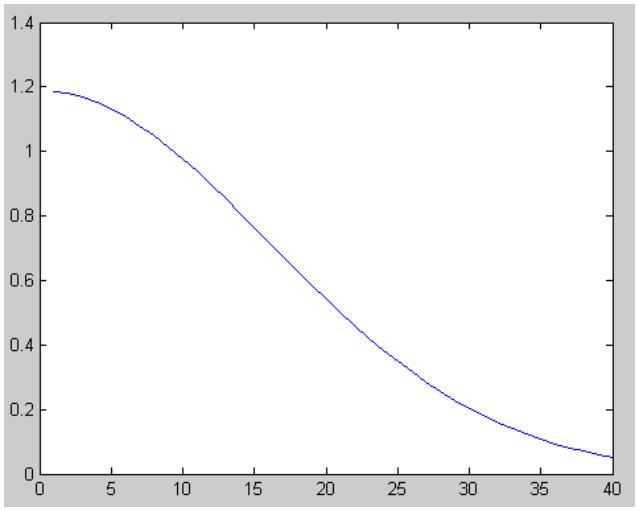
ในทางปฏิบัติกลุ่มของอะตอมที่เรียกว่าดิกชั่นนารี ซึ่งในบางครั้งจะเรียกว่า Gabor dictionary เป็นกลุ่มของtime-frequency atom ที่สร้างจากสมการ (2.5) จะถูกสร้างขึ้นไว้ล่วงหน้า โดยที่ดิกชันนารีเหล่านี้จะมีลักษณะเฉพาะตัวคือ

- เมื่อ $s = 1$และ $f = 0$ โดยกำหนดค่า p เป็นค่าใดๆ เราจะได้อะตอมที่มีลักษณะแบบ impulse ที่ตำแหน่ง p ดังภาพประกอบที่ 2-1



ภาพประกอบที่ 2-1 แสดงลักษณะ impulse เมื่อกำหนดให้ $s = 1, f = 0$ และ $p = 20$

- เมื่อ $s = N$ และ $p = 0$ เราจะได้อะตอมที่มีลักษณะแบบ sine wave [2] ดังภาพประกอบที่ 2-2



ภาพประกอบที่ 2-2 แสดงลักษณะ impulse เมื่อกำหนดให้ $s = N$ และ $p = 0$

องค์ประกอบของสัญญาณที่แยกออกมาด้วยวิธีการแม็ชชิงเพิชยูทนั้น สามารถนำมาวิเคราะห์ได้ทั้งใน Time domain และ Frequency domain พร้อมๆกัน ซึ่งการกระทำอย่างนี้จะทำให้มองเห็นคุณสมบัติต่างๆของสัญญาณ $x(t)$ ได้ชัดเจนยิ่งขึ้น และองค์ประกอบที่แยกมาได้นี้สามารถนำกลับไปรวมกันใหม่ได้สัญญาณที่คล้ายคลึงกับสัญญาณเดิมมาก [1]

เนื่องจากสัญญาณที่นำมาวิเคราะห์ในงานวิจัยนี้เป็นสัญญาณเวลาเต็มหน่วย (Discrete-time signal) ดังนั้นเราจะแทนสัญญาณที่ต้องการวิเคราะห์ด้วยสัญลักษณ์ $x(t)$ หรือ $x[t]$

ในทางปฏิบัติ รีดันแดนซ์ดิกชันนารี (อาจเรียกว่า Gabor dictionary) ของ Time-frequency atoms จะต้องถูกสร้างขึ้นก่อนเป็นอันดับแรก และในการแยกสัญญาณ $x(t)$ ไปเป็นชุดของอะตอมที่มีโครงสร้างแบบ Time-frequency ของสัญญาณได้นั้นต้องทำการออโทกอนัลโปรเจกชัน (Orthogonal projection) สัญญาณ $x(t)$ ซ้ำๆลงไปบนดิกชันนารี

ให้ $h_0(t)$ คือ อะตอมหนึ่งใน Gabor dictionary ในการโปรเจกชันครั้งแรกจะแยกสัญญาณออกเป็นสองส่วนดังสมการ

$$x(t) = x(t), h_0(t) \quad h_0(t) \qquad r^1 x(t) \tag{2.6}$$

เมื่อ

$x(t), h_0(t)$ คือ Inner product ของ f และ g_0

$x(t), h_0(t) = x(t) \ \overline{h_0}(t) \ dt$

$\overline{h_0}(t)$ คือ Complex conjugate ของ $h_0(t)$

$r^1 x(t)$ คือ The residue vector ซึ่งได้จากการประมาณค่า x ในทิศทางของ $h_0(t)$ โดยที่ $R^1 x$ จะออโทกอนัลกับ $h_0(t)$

และเนื่องจาก $\|h(t)\| = 1$ ดังนั้นพลังงานของสัญญาณสามารถเขียนเป็นสมการได้ดังนี้

$$\|x(t)\|^2 = | \ x(t), h_0(t) \ |^2 \qquad \|R^1 x(t)\|^2 \tag{2.7}$$

จากสมการที่ (2.7) จะต้องหาอะตอม $h_0(t)$ ที่เหมาะสมที่สุดที่ทำให้ $\|R^1 x(t)\|$ มีค่าน้อยที่สุด หรือเมื่อนำ $h_0(t)$ มาหา Inner product กับสัญญาณ $x(t)$ แล้วทำให้ $| \ x(t), h_0(t) \ |$ มีค่าสูงสุด

หลังจากนั้นทำกระบวนการนี้ซ้ำโดยการนำ $R^1 x(t)$ แทนที่ $x(t)$ ถ้ากระบวนการนี้ถูกทำซ้ำๆไปจนกระทั่งสัญญาณถูกแยกไปเป็นจำนวน m องค์ประกอบแล้ว จะสามารถแสดงการแยกองค์ประกอบ (Decompose) ของสัญญาณ $x(t)$ ได้ดังนี้

$$x(t) = \sum_{i=0}^{m-1} R^i x(t), h_i(t) \quad h_i(t) \quad R^m x(t) \tag{2.8}$$

และในทำนองเดียวกัน สามารถหาค่าพลังงานของ f ในสมการที่ (2.9) ได้ดังนี้

$$\|x(t)\|^2 = \sum_{i=0}^{m-1} | \ R^i x(t), h_i(t) \ |^2 \quad \|R^m x(t)\|^2 \tag{2.9}$$

ในสมการ (2.10) จะแสดงให้เห็นว่าเมื่อ m มีค่าเข้าใกล้ สัญญาณจะถูกแทนด้วยอนุกรมกำลังสองของ Time-frequency atoms จากดิกชันนารี โดยไม่มีการผิดเพี้ยนใดๆดังนี้

$$x(t) = \sum_{n=0} R^i x(t), h_i(t) \quad h_i(t) \tag{2.10}$$

$$\lim_m R^m x(t) = 0$$

และค่าพลังงานของสัญญาณ คือ

$$\|x(t)\|^2 = \sum_{n=0} \left| R^i x(t), h_i(t) \quad h_i(t) \right|^2 \tag{2.11}$$

การแยกองค์ประกอบของสัญญาณด้วยวิธีการแม็ชชิงเพิชยูทนั้น องค์ประกอบของสัญญาณที่มีกำลังสูงกว่ามักจะถูกแยกออกมาก่อน ซึ่งองค์ประกอบเหล่านี้จะถูกพิจารณาว่าเป็นโคฮีเรนท์ (Coherent part) ของสัญญาณ เนื่องจากรูปคลื่นและสัญญาณมีลักษณะที่คล้ายๆกัน สำหรับการหยุดกระบวนการทำซ้ำ ทำได้โดยการกำหนดจำนวน m ของ Time-frequency atoms และ residual energy : ^{2} ไว้ล่วงหน้า โดยกำหนดให้ความสัมพันธ์ระหว่างค่า Signal residue และ energy threshold ดังสมการที่ (2.12)

$$\left| R^n x(t) \right|^2 \quad ^2 \tag{2.12}$$

สามารถคำนวณค่า residual energy ได้จากสมการ (2.13) ดังนี้

$$R_m = \log_{10} \frac{E \sum_{i=1}^{m} a_i^2}{E} \tag{2.13}$$

เมื่อ

$E = \sum_{n=0}^{N-1} x^2(t)$ คือ Total energy ของ $x(t)$

a_i คือ ขนาดของแต่ละ Time-frequency atoms

และการคำนวณหาค่าผิดพลาดระหว่างสัญญาณเดิม (Original signal) และสัญญาณที่นำมารวมกันใหม่ (Reconstructed signal) ได้จากสมการที่ (2.14) ดังนี้ [1 , 2 และ 3]

$$NRMSE = 100 \sqrt{\frac{\sum_{n=0}^{N-1} e^2(t)}{\sum_{n=0}^{N-1} x^2(t)}} \tag{2.14}$$

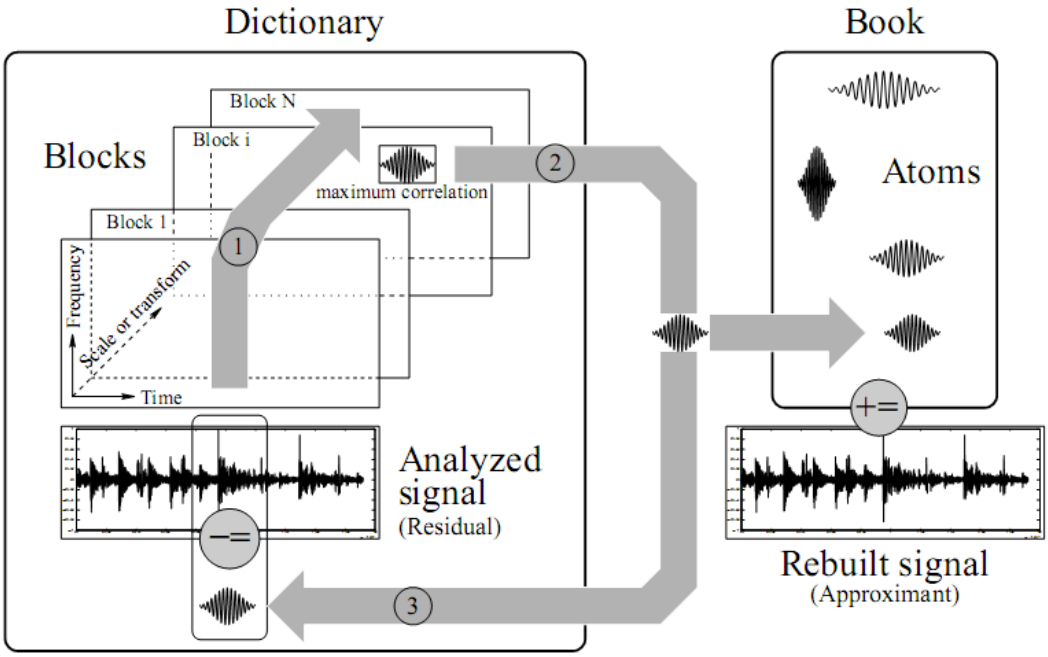
เมื่อ

$NRMSE$ คือ The normalized root-mean square error ระหว่างสัญญาณดั้งเดิมและสัญญาณที่นำกลับมารวมกันใหม่

$x(t)$ คือ สัญญาณดั้งเดิม

$e(t)$ คือ ผลต่างระหว่างสัญญาณดั้งเดิมและสัญญาณที่นำกลับมารวมกันใหม่

ซึ่งการหยุดกระบวนการทำซ้ำนั้น นอกจากทำโดยการกำหนดจำนวน m ของ Time-frequency atoms และ residual energy : ^{2} ไว้ล่วงหน้าแล้ว เราอาจทำได้โดยการกำหนดค่าต่ำสุดของ $NRMSE$ ไว้ในการเขียนโปรแกรมก็ได้ [2]



ภาพประกอบที่ 2-3 แสดงรูปแบบการทำงานของกระบวนการแม็ชชิงเพิชยูทอย่างง่าย [4]

จากภาพประกอบที่ 2-3 เราสามารถสรุปการทำงานของกระบวนการแม็ชชิงเพิชยูทได้เป็น 3 ขั้นตอนดังนี้

1.หาอะตอมที่มีค่า inner product สูงสุด ซึ่งเกิดจากการโปรเจคชันสัญญาณตั้งต้นไปยังอะตอมที่อยู่ในดิกชันนารี

2.นำอะตอมที่มีค่า inner product สูงสุด มาลบออกจากสัญญาณตั้งต้น

3.ผลลัพธ์ที่ได้จากการลบ นำไปเป็นสัญญาณตั้งต้นในรอบใหม่

2.2 การลดสัญญาณรบกวนโดยใช้กระบวนการแม็ชชิงเพิชยูท (noise reduction principle of the matching pursuit)

กระบวนการแม็ชชิงเพิชยูทสามารถเลือกอะตอม $h_i(t)$ ซึ่งเป็นอะตอมในรอบที่ i ที่สามารถให้ค่า Inner product กับสัญญาณดั้งเดิม $R^i x(t)$ ในทุกๆรอบการทำงานได้ ซึ่งอะตอม $h_i(t)$ ที่ได้ในแต่ละรอบนั้น จะสามารถประมาณถึงสัญญาณ $R^i x(t)$ ในแต่ละรอบได้เช่นกัน

และเมื่อกระบวนการแม็ชชิงเพิชยูทยังคงกระทำกระบวนการอยู่เรื่อยๆ ผลลัพธ์ที่ได้ในแต่ละรอบจะเป็นอะตอม $h_i(t)$ ที่เป็นองค์ประกอบของสัญญาณดั้งเดิมทั้งหมด

การลดสัญญาณรบกวนโดยใช้กระบวนการแม็ชชิงเพิชยูทจะอาศัยหลักการที่ว่าด้วยการมองสัญญาณที่เข้ามาในระบบเป็นผลรวมของสัญญาณสองชนิด

$$f = s \quad w \tag{2.15}$$

ชนิดแรกเป็นสัญญาณหลัก(s) เป็นสัญญาณที่มีค่า coherence ratio สูง ชนิดที่สองเป็นสัญญาณรบกวน(w) เป็นสัญญาณที่มีค่า coherence ratio ต่ำ ซึ่งค่า coherence ratio สามารถหาได้จากสมการ (2.16)

$$\lambda(f) = \sup \frac{\left| \quad f, h(t) \quad \right|}{\|f\|} \tag{2.16}$$

ค่า coherence ratio เมื่อกระบวนการดำเนินไป n รอบ

$$\tilde{\lambda}(R^n x(t)) = \frac{\left| \quad R^n x(t), h_n(t) \quad \right|}{\left\| R^n x(t) \right\|} \tag{2.17}$$

และ

$$\tilde{\lambda}(R^n x(t)) \quad \lambda(R^n x(t)) \quad \frac{1}{} \tilde{\lambda}(R^n x(t)) \tag{2.18}$$

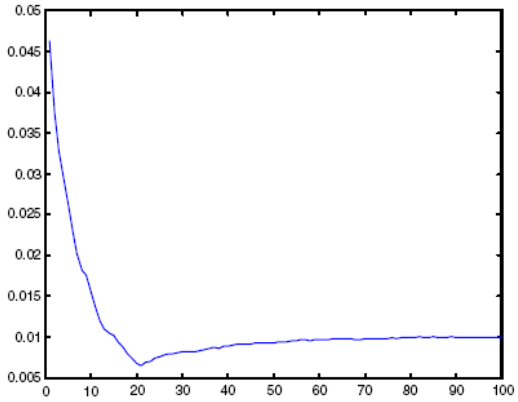
ส่วนประกอบของสัญญาณที่ปราศจากสัญญาณรบกวนจะมีค่า coherence ratio สูง และสัญญาณรบกวนนั้นจะมี coherence ratio ต่ำกระบวนการแม็ชชิงเพิชยูทจะดึงเอาสัญญาณที่มีค่า coherence ratio สูงสุด(s) ซึ่งมีพลังงานสูงสุดออกมาจากสัญญาณรบกวน(w) ซึ่งสัญญาณที่เหลือจะมีค่า $R^i x(t)$ coherence ratio ลดลงและเมื่อให้กระบวนการนี้ทำงานเรื่อยๆ กระบวนการนี้จะดึงเอาสัญญาณที่มีพลังงานสูงเรื่อยๆจนเหลือเพียงแค่สัญญาณรบกวนเท่านั้น

การหยุดกระบวนการแม็ชชิงเพิชยูทนี้สามารถทำได้โดยกำหนดค่า

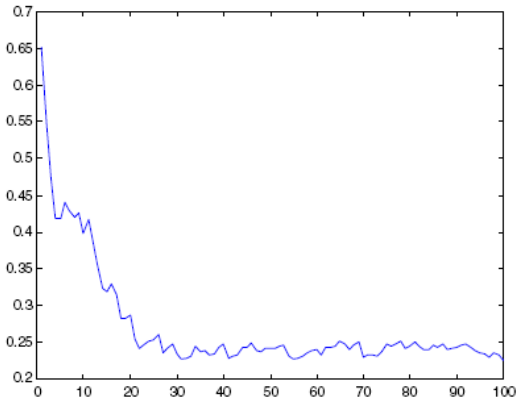
- P_K ซึ่งเป็นจำนวนรอบการทำงานของกระบวนแม็ชชิงเพิชยูท ซึ่งจะทำให้ค่า σ ของสัญญาณลดลง ดังภาพประกอบที่ 2-3 หรือ

- λ_C ซึ่งเป็นค่า coherence ratio ของสัญญาณหลัก ซึ่งถ้าค่าต่ำกว่าที่กำหนดนี้ จะเป็นค่าของสัญญาณรบกวนดังภาพประกอบที่ 2-4

เมื่อกระบวนการทำงานจนถึงค่าใดค่าหนึ่งก่อน จะทำให้กระบวนการหยุดทำงาน



ภาพประกอบที่ 2-4 กราฟแสดงความสัมพันธ์ระหว่าง P_K และ σ



ภาพประกอบที่ 2-5 กราฟแสดงความสัมพันธ์ระหว่างค่า λ_C กับจำนวนรอบการทำงาน

2.2.1 การลดสัญญาณรบกวนชนิด wide band noise

วิธีการทั่วไปในการลด wide band noise มีด้วยกัน 2 ประเภท[2] คือ

- **optimal estimation , optimal filtering** : วิธีการนี้ filter ที่ใช้ส่วนมาก คือ least square estimation , Wiener filtering , Kalman filtering ซึ่งวิธีการนี้เป็นการประมาณค่าสัญญาณโดยใช้ข้อมูลที่ได้จากสัญญาณในเวลานั้นๆ และค่าประมาณของสัญญาณในเวลาก่อนหน้านี้ แต่เนื่องจากค่าพารามิเตอร์ของวงจรกรองประเภทนี้เป็นค่าคงที่ จึงทำให้ไม่เหมาะสมเมื่อนำมาใช้งานกับสัญญาณที่มีการเปลี่ยนแปลงตลอดเวลา และในส่วนของ Kalman filter นั้น เป็นวงจรกรองที่สามารถใช้ได้กับสัญญาณที่เปลี่ยนแปลงตามเวลา แต่ในการใช้งานจริงนั้นจำเป็นต้องทราบถึงลักษณะของสัญญาณ และลักษณะของสัญญาณรบกวนด้วย จึงจะทำให้สามารถใช้วงจรกรองประเภทนี้ได้อย่างมีประสิทธิภาพ

- **threshold method** : เป็นวิธีการซึ่งอยู่บนพื้นฐานของ wavelet transform ซึ่งวิธีการนี้เริ่มแรกจะแปลงสัญญาณโดยใช้ wavelet จากนั้นจะกำหนดให้ค่าสัมประสิทธิ์(coefficient) ให้น้อยกว่าค่า amplitude ที่อยู่ใน transform domain โดยใช้วิธีการ soft threshold หรือ hard threshold จากนั้นใช้ inverse wavelet เพื่อให้ได้สัญญาณที่ไม่มีสัญญาณรบกวนออกมา ซึ่งวิธีการนี้เหมาะสำหรับการวิเคราะห์สัญญาณที่เปลี่ยนแปลงตลอดเวลา แต่ก็จำเป็นที่จะต้องทราบถึงลักษณะของสัญญาณและสัญญาณรบกวนเช่นเดียวกันจึงจะทำให้ใช้วงจรกรองประเภทนี้ได้อย่างมีประสิทธิภาพ

สำหรับวงจรกรองที่ใช้ในการกรองสัญญาณรบกวนอย่างง่าย เช่น FIR และ IIR นั้น จะกรองสัญญาณในรูปของความถี่ที่ไม่ต้องการออกไป ซึ่งในบางครั้งทำให้สัญญาณที่สำคัญนั้นถูกกำจัดออกไป

วิธีการกำจัดสัญญาณรบกวนข้างต้นนั้น สามารถใช้วิธีการแม็ชชิงเพิซยูทแทนได้ โดยวิธีการนี้จะมองในแง่ของพลังงาน ซึ่งเป็นความสัมพันธ์ระหว่างสัญญาณหลักกับสัญญาณรบกวน ซึ่งสัญญาณหลักจะมีพลังงานสูง และสัญญาณรบกวนจะมีพลังงานต่ำ ทำให้ผลคูณภายในระหว่างสัญญาณหลักกับอะตอมที่สร้างขึ้นมีค่าสูงกว่าผลคูณภายในระหว่างสัญญาณรบกวนกับอะตอม

เมื่อทำกระบวนการนี้เรื่อยๆ โดยดึงเอาสัญญาณที่มีพลังสูงออกมา จะทำให้สามารถแยกสัญญาณหลักออกจากสัญญาณรบกวนได้

จากการทดลองการลดสัญญาณรบกวนโดยใช้สัญญาณเสียงโดยเพิ่ม Gaussian random white noise เปรียบเทียบระหว่างการใช้วิธีการแม็ชชิงเพิซยูทและเวฟเลท พบว่า ค่า SNR เมื่อมีการเปลี่ยนแปลงสัญญาณหลักและสัญญาณรบกวนที่ได้จากกระบวนการแม็ชชิงเพิซยูท ดีกว่ากระบวนการเวฟเลทในหลายๆกรณีดังภาพประกอบที่ 2-5 [5]

TABLE I Contrast of optimality noise reduction and coherent ratio judging noise reduction

Noise Level σ_w	Original (SNR)	Optimality noise reduction			Coherent ratio judging noise reduction		
		K1	SNR	RMS	K2	SNR	RMS
0.0005	41.7311	75	42.7285	0.0071	52	41.8568	0.0079
0.0025	27.7517	35	29.4508	0.0328	32	29.2781	0.0335
0.005	21.7311	29	24.2214	0.0600	26	23.8896	0.0623
0.01	15.7105	20	19.2870	0.1060	20	19.2870	0.1060
0.02	9.6899	13	14.6079	0.1896	14	14.1960	0.1980
0.03	6.1681	10	10.8065	0.2900	9	10.1077	0.3171
0.04	3.6693	11	7.8131	0.4076	8	7.6118	0.4165
0.05	1.7311	6	6.4582	0.5112	7	6.1229	0.5047
0.06	0.1475	6	6.1085	0.5298	6	6.1085	0.5298
0.12	-5.8731	2	2.6815	0.7494	1	2.4027	0.7717
0.24	-11.8937	1	-2.7206	1.3520	0	/	/

TABLE II The experiment results of Chirp signal

σ_w	0.0005	0.005	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08
SNR_OR	42.013	22.013	15.992	9.972	6.450	3.951	2.013	0.429	-0.910	-2.070
SNR_WA	40.982	20.403	14.617	11.728	7.149	5.450	4.065	2.016	1.416	1.133
SNR_MP	41.466	24.199	19.719	12.697	9.708	6.642	5.968	5.265	3.875	3.245

TABLE III The experiment result of Doppler signal

σ_w	0.0005	0.005	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08
SNR_OR	41.789	21.731	15.711	9.690	6.168	3.669	1.731	0.148	-1.191	-2.351
SNR_WA	43.030	24.363	18.915	11.575	10.363	7.107	6.378	5.068	4.606	3.357
SNR_MP	41.755	23.890	19.287	14.196	10.108	7.612	6.123	6.109	5.159	4.946

TABLE IV The experiment results of Blocks signal

σ_w	0.0005	0.005	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08
SNR_OR	40.251	20.251	14.230	8.210	4.688	2.189	0.251	-1.333	-2.672	-3.832
SNR_WA	41.377	19.614	13.618	7.977	5.127	3.994	3.144	2.333	2.102	1.849
SNR_MP	37.079	20.416	14.110	9.593	6.865	4.816	4.118	2.938	1.726	1.347

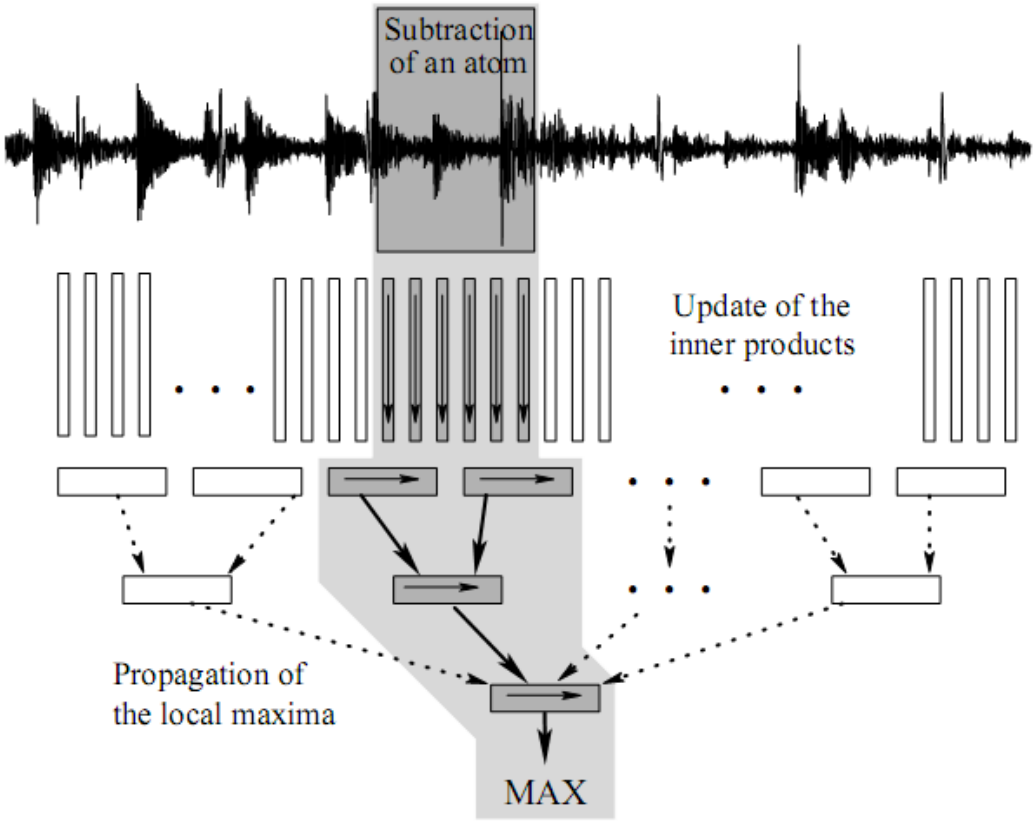
ตารางที่ 1 แสดงค่า SNR ระหว่างวิธีการแม็ชชิงเพิชยูทและเวฟเลท[5]

จากกระบวนการทำงานของกระบวนการแม็ชชิงเพิชยูทนี้จะขึ้นอยู่กับขนาดของอะตอมที่อยู่ในดิกชันนารี ซึ่งกระบวนการปกตินี้จะทำงานอยู่ที่ $O(N^2)$ ครั้ง ที่เป็นเช่นนี้เพราะกระบวนการนี้ต้องหาอะตอมที่มีผลคูณภายในสูงสุดในทุกๆดิกชันนารี โดยในขั้นตอนที่เป็นการคำนวณค่าผลคูณภายในนั้น ถ้าสัญญาณมีจำนวน N ตัว จะต้องคำนวณเป็นจำนวน N ครั้ง และในขั้นตอนที่หาค่า

ความสัมพันธ์สูงสุดนั้นก็จะต้องเปรียบเทียบค่าอีก N ครั้งซึ่งจะทำให้เกิดปัญหาคอขวดในการคำนวณ และถ้าสัญญาณมีขนาดใหญ่มากๆ จะทำให้ใช้เวลาในการคำนวณมากขึ้นเช่นกัน แต่ในบทความ นำเสนอการลดกระบวนการทำงานให้เหลือเพียง $O(N \log N)$ โดยเสนอวิธีการลดการคำนวณดังนี้ [4]

วิธีการเพิ่มความเร็วในการทำงานกระบวนการแม็ชชิงเพิชยูท มีด้วยกัน 2 วิธี คือ

- ออกแบบการทำงานแบบขนาน (Parallelism) ซึ่งเป็นการแบ่งดิกชันนารีออกเป็นกลุ่มๆและแยกกันหาผลคูณภายในในแต่ละกลุ่ม
- ออกแบบดิกชันนารีใหม่ โดยประยุกต์การใช้งาน FFT เข้ามาช่วย



ภาพประกอบที่ 2-6 วิธีการหาผลคูณภายในสูงสุดโดยใช้กระบวนการทำงานแบบแผนภาพต้นไม้

จากภาพประกอบที่ 2-5 เป็นการออกแบบกระบวนการแม็ชชิงเพิชยูทแบบขนาน โดยแบ่งดิกชันนารีออกเป็นกลุ่มย่อย โดยให้แต่ละกลุ่มหาค่า inner product ภายในแต่ละกลุ่ม ซึ่งกลุ่มที่มีอะตอมที่มี inner product ก็จะนำอะตอมนั้นไปใช้ในขั้นต่อไป ซึ่งวิธีการนี้จะลดการคำนวณจากแบบเดิมที่ต้องทำการหา inner product ทีละอะตอมจนหมด [4]

2.3 การออกแบบกระบวนการแม็ชชิงเพิชยูทบน FPGA

เนื่องจากในการออกแบบกระบวนการแม็ชชิงเพิชยูทนี้เป็นการออกแบบเพื่อใช้กับการประมวลผลสัญญาณในระบบเครือข่ายไร้สาย ซึ่งอยู่บนพื้นฐานของวิธีการ Maximum likelyhood (ML) ดังนั้นรูปแบบสัญญาณ รูปแบบสมการ และการทำงาน จะแตกต่างกับกระบวนการแม็ชชิงเพิชยูทที่ใช้สำหรับการประมวลผลสัญญาณเสียง

ในการประมวลผลสัญญาณของระบบเครือข่ายไร้สายนั้น ลักษณะของสัญญาณจะอยู่ในรูปของ

$$r = Sf \quad n \qquad C^{MN_S*1} \tag{2.19}$$

โดยที่ M คือ จำนวน bit ของข้อมูล, n คือ สัญญาณที่พิจารณาซึ่งรวมกับ noise ในรูปแบบของ Gaussian noise , f คือ สัมประสิทธิ์ของแต่ละช่องสัญญาณซึ่งอยู่ในรูปแบบของเวกเตอร์ และ S คือ ลักษณะสัญญาณเฉพาะตัว(characteristic signal) ซึ่งจะอยู่ในรูปของเมตริกซ์ ซึ่งในที่นี้จำเป็นต้องประมาณค่า f ($\hat{f}$)โดยที่เราทราบ S ซึ่งจะได้สมการเป็น

$$\hat{f} = \underset{f \quad A_{Nf}}{\arg\min} \{\|r \quad Sf\|\}^2 \tag{2.20}$$

แต่วิธีการนี้มีความซับซ้อนมาก โดยสามารถประมาณค่าได้เป็น $C_{N_f}^{N_s} = (N_S!)/(N_f!(N_S \quad N_f)!)$ ทำให้ไม่สามารถนำมาใช้งานในรูปแบบ real time ได้

Sparse ML	MP	Fast MP
$O(MN_S C_{N_f}^{N_s} Q^{2(N_S N_f)})$	$O(2N_f MN_S^2)$	$O(2MN_S^2)$

ตารางที่ 2 แสดงถึงจำนวนขั้นตอนในการคำนวณในแต่ละวิธี

ดังนั้น เพื่อต้องการประมวลผลสัญญาณในรูปแบบ real time จึงจำเป็นต้องออกแบบวิธีการแม็ชชิงเพิชยูทใหม่ โดยปรับเปลี่ยนเป็น

$$\|r \quad Sf\|^2 \qquad 2\,\mathrm{Re}\{(V^O)^H f\} \quad f^H Af \tag{2.21}$$

ซึ่ง V^O คือ match filter โดยจะมีค่าเป็น $V^O = S^T r \qquad C^{N_S*1}$ และในการคำนวณสามารถนำ V^O มาคำนวณในรูปแบบขนานและ $A = S^T S \qquad R^{N_S*N_S}$ โดยที่ทั้ง S และ A จะถูกคำนวณไว้ล่วงหน้าและเก็บไว้ในหน่วยความจำโดยขนาดของหน่วยความจำจะขึ้นอยู่กับจำนวนbitของข้อมูล(M) และ V^O สามารถนำมาคำนวณแบบขนานเพื่อเพิ่มความเร็วได้

ในการหยุดกระบวนการนี้สามารถทำได้โดย 1) ให้จำนวนรอบ (j) มีค่าเท่ากับจำนวนbit ของสัมประสิทธิ์แต่ละช่องสัญญาณ(N_f) หรือ 2) กำหนดให้พลังงานให้น้อยกว่า SNR ที่ตั้งไว้

2.4 ปัจจัยในการออกแบบกระบวนการแมชชิงเพิชยูทบน FPGA

ในการออกแบบกระบวนการแม็ชชิงเพิชยูทบน FPGA นั้น จำเป็นต้องคำนึงถึงพารามิเตอร์หลักๆ 3 อย่าง คือ

- **การกำหนดลักษณะตัวแปรที่จะใช้ในการคำนวณ**

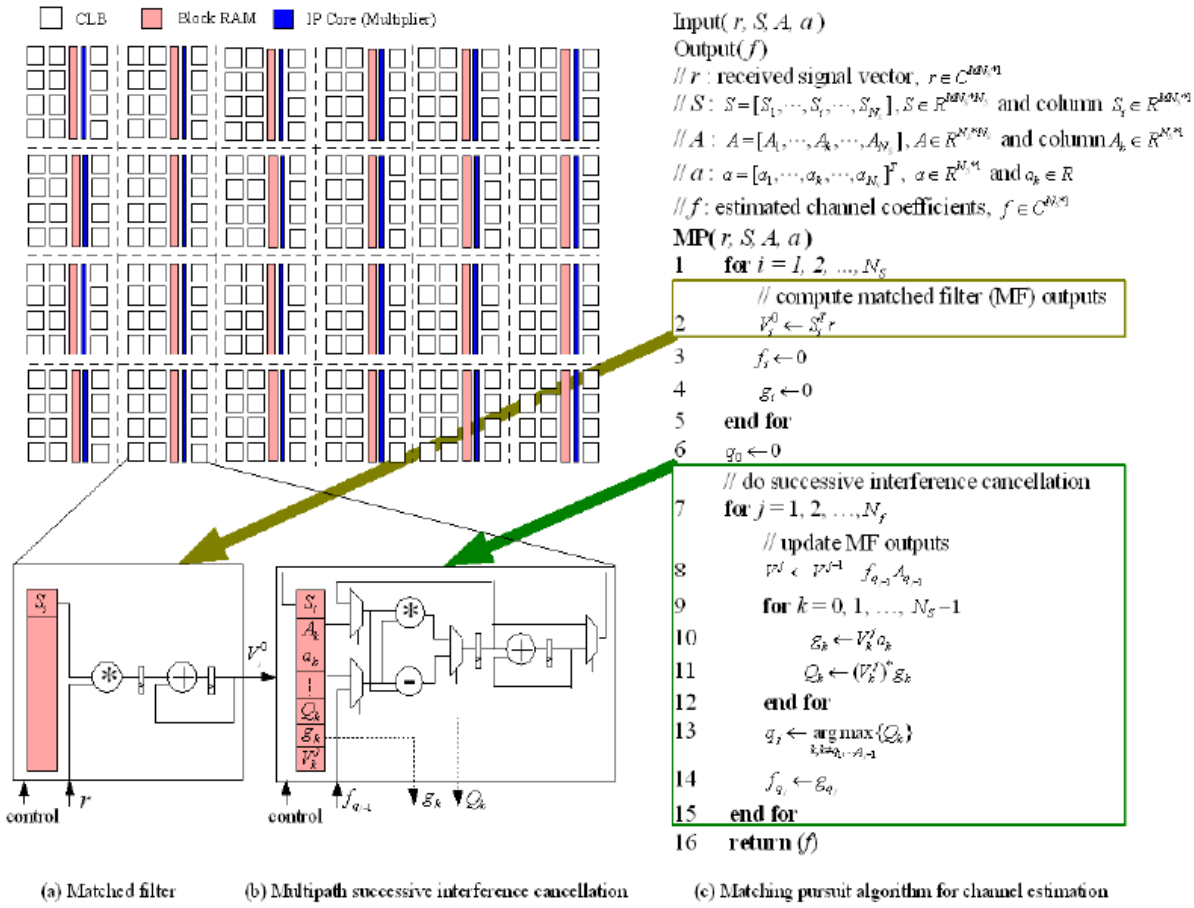
ในการประมวลผลสัญญาณไร้สายด้วยวิธีการแม็ชชิงเพิชยูทนี้ ตัวแปรแต่ละตัวจะอยู่ในรูปการคูณกันของ เมตริกซ์ – เวกเตอร์ ซึ่งสามารถแยกได้เป็นการคูณกันของ เวกเตอร์ – เวกเตอร์ ซึ่งสามารถนำมาคูณกันแบบขนานได้ แต่จะทำให้ทรัพยากรที่ใช้ มากขึ้นตามไปด้วย

- รูปแบบการนำเสนอข้อมูล

ในการประมวลผลนี้จะใช้รูปแบบข้อมูลแบบ fixed point ขนาด 8 bit เนื่องจากทำงานได้เร็ว , ใช้ทรัพยากรน้อย และใช้พลังงานต่ำกว่าการใช้รูปแบบข้อมูลแบบ floating point แต่ในรายงานฉบับนี้จะใช้ fixed point ขนาด 16 bit เนื่องมาจากข้อมูลขนาด 8 bit ไม่สามารถแทนข้อมูลแต่ละตัวอย่างของสัญญาณที่จำลองที่อยู่ในช่วงระหว่าง [-1 , 1] ได้ เมื่อเปรียบเทียบกับการใช้ข้อมูลขนาด 16 bit

- รูปแบบการกระจายตัวของข้อมูล

เนื่องจากในกระบวนการแม็ชชิงเพิชยูท จำเป็นต้องประมวลผลข้อมูลจำนวนมาก ดังนั้นจึงจำเป็นต้องจัดเรียงข้อมูลเพื่อให้สามารถทำงานได้เร็วยิ่งขึ้น ซึ่งในที่นี้จะทำการเรียงข้อมูลในรูปแบบขนาน (parallel) และรวมโมดูลต่างๆให้อยู่รวมกันเป็นกลุ่ม ซึ่งในแต่ละกลุ่มจะประกอบไปด้วย RAM , วงจรคูณ (multiplier) และ CLB [6]



ภาพประกอบที่ 2-8 แสดงการออกแบบกระบวนการแม็ชชิงเพิชยูทเพื่อใช้ในการประมวลผลสัญญาณไร้สาย[6]

จากภาพประกอบที่ 2-8 แสดงการออกแบบกระบวนการแมชชิงเพิชยูทบน FPGA เพื่อใช้ในการประมวลผลช่องสัญญาณในเครือข่ายไร้สาย ซึ่งในแต่ละกลุ่มจะประกอบไปด้วยโมดูลหลัก 2 ส่วน คือ

- Matched filter ซึ่งทำหน้าที่เปรียบเสมือนการหา inner product ในกระบวนการแม็ชชิงเพิชยูทแบบปกติ ภายในจะประกอบไปด้วย S_i ซึ่งเป็น Channel coefficient ซึ่งทำหน้าที่เป็นอะตอมและทำการสร้างและเก็บไว้ใน RAM ภายใน FPGA และในแต่ละโมดูลทำการหาค่า inner product กับสัญญาณ r ในรูปแบบขนานกัน
- Multipath successive interference cancellation เป็นโมดูลที่ทำหน้าที่ในการลบสัญญาณที่ได้จาก Match filter ออกจากสัญญาณที่สนใจ

บทที่ 3

การออกแบบกระบวนการแม็ชชิงเพิชยูท

กระบวนการแม็ชชิงเพิชยูทในรายงานวิจัยฉบับนี้ ถูกใช้งานในสองลักษณะ คือ ออกแบบบน MATLAB และออกแบบเพื่อให้ทำงานบน FPGA เพื่อเปรียบเทียบผลลัพธ์ที่ได้จาก FPGA

3.1การออกแบบกระบวนการแม็ชชิงเพิชยูทบน MATLAB

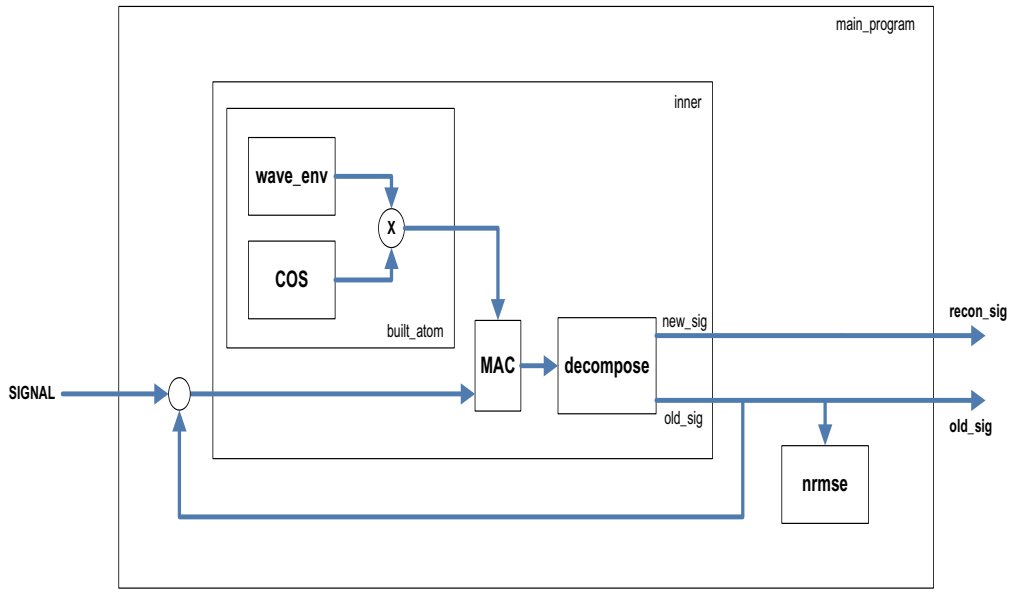
ในการออกแบบกระบวนการแม็ชชิงเพิชยูทบน MATLAB ผู้วิจัยได้ใช้โปรแกรม MATLAB เวอร์ชัน 7.8.0.347 (R2009a) บนคอมพิวเตอร์ส่วนบุคคลที่มีความเร็วซีพียู 3.0 GHz และหน่วยความจำ 4 GB ซึ่งลักษณะการทำงานของกระบวนการแม็ชชิงเพิชยูทบน MATLAB เป็นไปตามภาพประกอบที่ 3-1

จากภาพประกอบที่ 3-1 เมื่อนำสัญญาณที่ต้องการวิเคราะห์เข้ามาในระบบ จะต้องกำหนดจำนวนรอบในการทำงาน จากนั้นโปรแกรมจะทำการสร้างอะตอมขึ้นมา และทำการหาผลคูณภายในโดยทำการคูณกันระหว่างสัญญาณภายนอก(หรือสัญญาณตั้งต้นในรอบถัดไป) กับอะตอมที่สร้างขึ้นมา และทำการเปรียบเทียบกับผลคูณภายในก่อนหน้านี้ เพื่อหาค่าผลคูณภายในสูงสุด

เมื่อทำการหาค่าผลคูณภายในระหว่างระหว่างสัญญาณภายนอกกับอะตอมที่สร้างขึ้นทุกตัวจนได้ค่าผลคูณภายในสูงสุดแล้ว จะทำการคูณกันระหว่างผลคูณภายในสูงสุดกับอะตอมที่ให้ค่าผลคูณภายในสูงสุด ซึ่งจะได้เป็นสัญญาณใหม่ซึ่งเป็นสัญญาณที่มีพลังงานสูง แล้วนำสัญญาณใหม่ที่ได้ไปลบออกจากสัญญาณตั้งต้นซึ่งจะได้ผลลัพธ์เป็นสัญญาณตั้งต้นในรอบถัดไปซึ่งเป็นสัญญาณที่มีพลังงานต่ำลงมา

ขณะเดียวกันสัญญาณใหม่จะถูกแสดงให้ผู้ใช้เห็นโดยการนำมาบวกสะสมเรื่อยๆในแต่ละรอบซึ่งจะแสดงให้เห็นถึงการเปลี่ยนแปลงสัญญาณใหม่ที่สามารถหากลุ่มของสัญญาณตั้งต้นที่มีพลังงานสูงออกมาได้ พร้อมกันนี้ระบบจะทำการตรวจสอบจำนวนรอบ โดยที่ถ้าจำนวนรอบยังไม่ครบกับที่กำหนดไว้ตอนต้น ระบบจะทำการหาผลคูณภายในสูงสุดโดยใช้สัญญาณตั้งต้นในรอบถัดไป แต่ถ้าจำนวนรอบครบตามที่กำหนด ระบบจะหยุดการทำงานพร้อมทั้งแสดงสัญญาณที่ต้องการวิเคราะห์พร้อมทั้งสัญญาณใหม่ที่ได้จากการวิเคราะห์

ในการทำงานนี้จะอาศัยสมการที่ 2.1 – 2.5 มาใช้ในการออกแบบ ซึ่งจากภาพประกอบที่ 3-1 ผู้วิจัยสามารถแบ่งฟังก์ชัน main program สำหรับการทำงานใน MATLAB ได้ดังนี้



ภาพประกอบที่ 3-1 แสดงการทำงานของกระบวนการแม็ชชิงเพิชยูทบน MATLAB

3.1.1 ฟังก์ชัน inner : เป็นฟังก์ชันที่ใช้หา inner product ระหว่างสัญญาณตั้งต้น และอะตอมที่สร้างขึ้น และมีการหาค่า inner product ที่มากที่สุด ภายในฟังก์ชันนี้ประกอบไปด้วย 3 ส่วนหลัก คือ

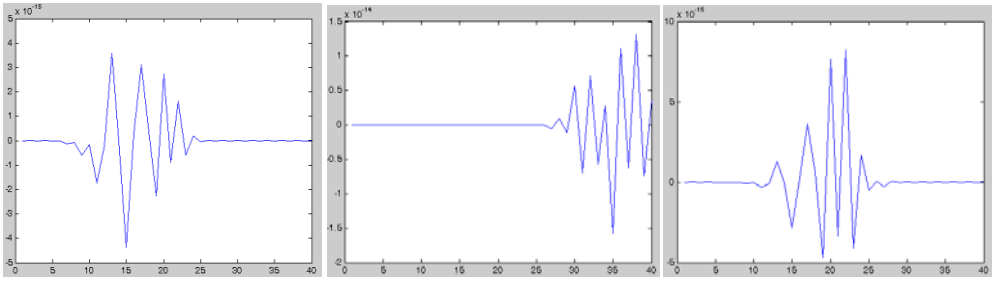
3.1.1.1 ฟังก์ชัน built atom : เป็นฟังก์ชันที่ใช้ในการสร้างอะตอม โดยจะใช้สมการที่2.3 – 2.5ในการสร้าง โดยค่าพารามิเตอร์ s, p และ k จะทำให้อะตอมที่สร้างขึ้นทั้งหมดมีลักษณะแตกต่างกัน และการสร้างนั้นจะอาศัยการวนรอบ ซึ่งใน 1 รอบการทำงานนั้นจะสร้างอะตอมได้ 1 ตัวอย่าง ซึ่งภายในฟังก์ชันนี้จะประกอบไปด้วย ฟังก์ชัน wave_env และฟังก์ชัน cos ซึ่งผลลัพธ์ทั้งสองฟังก์ชันนี้นำมาคูณกัน จะได้เป็นสัญญาณอะตอมตามที่ต้องการ อะตอมที่ถูกสร้างขึ้นโดยฟังก์ชันนี้จะเป็นดังภาพประกอบที่ 3-2

- ฟังก์ชัน wave_env ซึ่งทำหน้าที่สร้างกลุ่มสัญญาณอะตอมตามสมการที่ 2-5
- ฟังก์ชัน cos ซึ่งทำหน้าที่สร้างสัญญาณรูป cos
- mul ทำหน้าที่หาผลคูณของฟังก์ชัน cos และ wave_en

3.1.1.2 ฟังก์ชัน decompose : เป็นฟังก์ชันที่ใช้ในการแยกสัญญาณ โดยการนำอะตอมซึ่งถูกคูณด้วยค่า inner product ไปลบออกจากสัญญาณตั้งต้น ซึ่งผลลัพธ์ที่ได้จะเป็นสัญญาณตั้งต้นในรอบถัดไป

3.1.1.3 MAC ทำหน้าที่หาผลคูณภายในระหว่างอะตอมที่สร้างขึ้นกับสัญญาณตั้งต้นในแต่ละรอบ

3.1.2 ฟังก์ชัน : nrmse : เป็นฟังก์ชันที่ใช้แสดงค่า rms ของสัญญาณหลังจากที่ทำการแยกเอาอะตอมที่เป็นส่วนประกอบออกไปในแต่ละรอบ



ภาพประกอบที่ 3-2 ตัวอย่างอะตอมที่สร้างจากฟังก์ชัน built_atom โดยมีค่า s, p และ k ต่างกัน

ในรายงานวิจัยฉบับนี้ ได้ออกแบบกระบวนการแม็ชชิงเพิชยูทให้สามารถวิเคราะห์สัญญาณจำลองที่มีอัตราการชักสัญญาณต่ำซึ่งมีความยาว 40 ตัวอย่าง ได้ จำนวนอะตอมทั้งหมดที่กระบวนการนี้สร้างขึ้นนั้น ขึ้นอยู่กับความยาวของสัญญาณและค่าพารามิเตอร์ s , p , k

จากบทความ[2] พบว่า ค่า scale factor (s) จะอยู่ในช่วง [1,N] ส่วนพารามิเตอร์ p , k จะอยู่ในช่วง (0,N-1) และเนื่องจากการสร้างอะตอมขึ้นมาแต่ละครั้งจะต้องใช้พารามิเตอร์ทั้งสามตัว จึงทำให้เกิดความยุ่งยากในการสร้าง ดังนั้น เราสามารถลดความซับซ้อนในการสร้างได้โดยการกำหนดให้ scale factor อยู่ในรูปของ exponential ดังสมการ

$$s = \quad ^{j} \qquad (3.1)$$

เมื่อ j คือ octave scale ซึ่งมีค่าอยู่ระหว่าง$(0, \log N)$

จาก [3] พบว่า เมื่อ $= 2$ และจาก [2] พบว่าเมื่อ $j = 3$ จะให้ผลลัพธ์ในการสร้างอะตอมที่ดีที่สุด

3.2การออกแบบกระบวนการแม็ชชิงเพิชยูทบน FPGA

ในการออกแบบกระบวนการแม็ชชิงเพิชยูทบน FPGA ผู้วิจัยได้ใช้ซอฟท์แวร์ Xilinx ISE 12.1 และได้ทำการออกแบบกระบวนการนี้บน FPGA Xilinx Virtex5 XC5VSX95T ซึ่งก่อนหน้านี้ผู้วิจัยได้ออกแบบโดยใช้ซอฟท์แวร์ Xilinx ISE 10.1.3 และใช้ FPGA Xilinx Virtex2 Pro X2VP30 แต่ติดปัญหาบางประการซึ่งจะชี้แจงในบทที่ 5 โดยลักษณะโครงสร้างวงจรโดยรวมจะอ้างอิงกับ

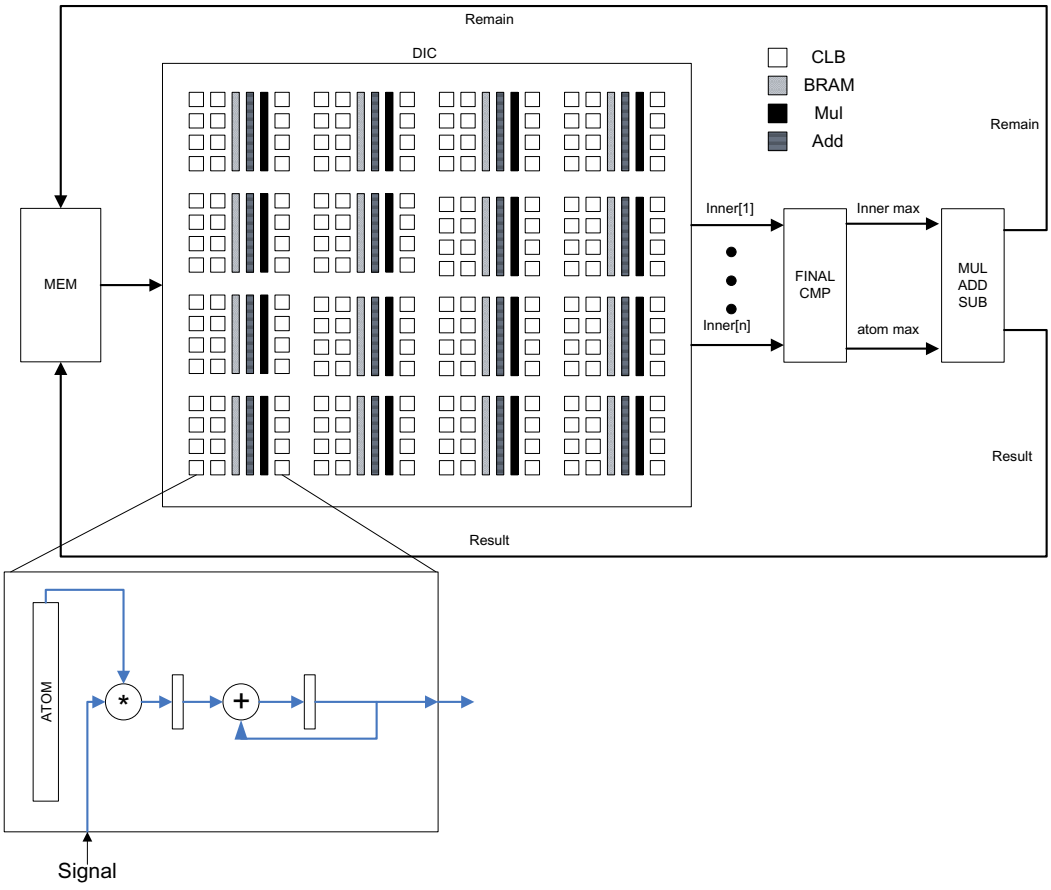
การออกแบบใน FPGA Xilinx Virtex2 Pro X2VP30 และหัวข้อที่ 1.5.2 เป็นไปดังภาพประกอบที่ 3-3 ซึ่งภายใน FPGA นั้นจะประกอบไปด้วยโมดูลหลักๆ 4 กลุ่ม คือ

-กลุ่มของโมดูล MEM ทำหน้าที่เก็บสัญญาณตั้งต้นในรอบแรก และเก็บผลลัพธ์เพื่อใช้ในการคำนวณในรอบถัดไป

-กลุ่มของโมดูล DIC ทำหน้าที่เก็บอะตอมเพื่อใช้ในการคำนวณพร้อมทั้งยังประกอบไปด้วยโมดูลย่อยที่ใช้ในการหาค่าผลคูณภายในสูงสุดซึ่งใน FPGA นี้ จะประกอบไปด้วยโมดูลนี้ 40 ตัว ซึ่งทุกตัวจะทำงานขนานกัน

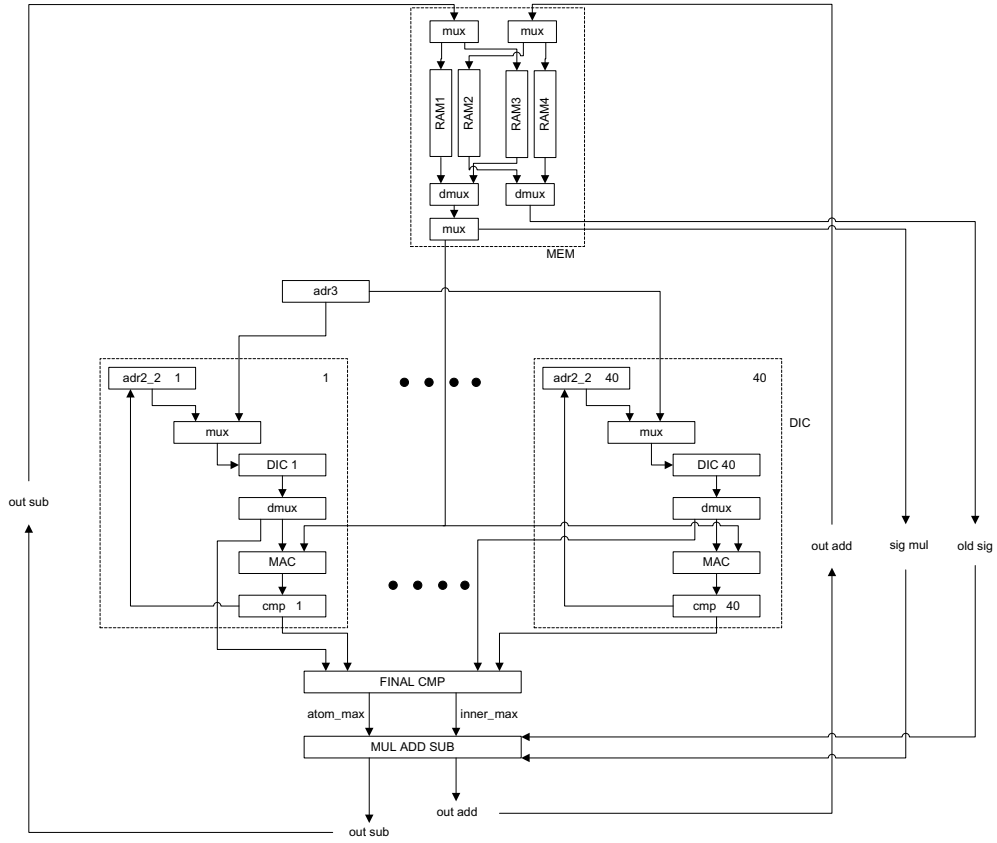
-โมดูล FINAL CMP ทำหน้าที่หาค่าผลคูณภายในสูงสุดที่ได้จากการคำนวณของโมดูล DIC

-โมดูล MUL ADD SUB ใช้หาสัญญาณตั้งต้นในรอบใหม่ และหาสัญญาณใหม่ซึ่งผลลัพธ์ที่ได้จะนำไปเก็บในโมดูล MEM เพื่อใช้ในรอบถัดไป



ภาพประกอบที่ 3-3 แสดงวงจรรวมของกระบวนการแม็ชชิงเพิชยูทบน FPGA

โดยลักษณะการทำงานของกระบวนการแม็ชชิงเพิชยูทบนFPGAจะเป็นไปดังภาพประกอบที่ 3-4



ภาพประกอบที่ 3-4 แสดงรายละเอียดของโมดูลของกระบวนการแม็ชชิงเพิชยูท

3.2.1โมดูลที่เกี่ยวข้องกับกระบวนการแม็ชชิงเพิชยูทบน FPGA

1. โมดูล DIC[n] : เป็นโมดูลที่ใช้ในการเก็บค่าดิกชันนารีที่ได้จากการสร้างของฟังก์ชัน built_atom ซึ่งประกอบไปด้วยสมการ 2.2 – 2.4 โดยใช้ MATLAB ซึ่งโมดูลนี้ถูกสังเคราะห์ให้เป็น Single port ROM ภายใน ROM แต่ละตัวจะเก็บข้อมูลของดิกชันนารีขนาด 16 1600 bit

โมดูล DIC[n] จะถูกสร้างขึ้นจำนวน 40 ตัว โดยแต่ละตัวจะเก็บค่าดิกชันนารีซึ่งอะตอมในแต่ละโมดูลนั้นจะไม่ซ้ำกันโมดูลละ 40 อะตอม ซึ่งจะมีอะตอมรวมทั้งหมดเท่ากับ 1600 อะตอม และมีชื่อที่แตกต่างกัน โดยจะเรียงลำดับตามอะตอมที่สร้างขึ้นจากฟังก์ชัน built_atom

ในการออกแบบดิกชั่นนารีนี้ ผู้วิจัยได้ออกแบบให้ดิกชั่นนารีเก็บอะตอมเป็นกลุ่มละเท่าๆกัน ซึ่งขนาดของสัญญาณจะมีผลต่อจำนวนอะตอมและความยาวของอะตอมตามสมการที่ 2.3 และ 2.5 จะเห็นได้ว่าสัญญาณที่มีความยาวจะสัมพันธ์กับจำนวนอะตอมในรูปแบบ exponential

และเนื่องมาจาก RAM ใน FPGA Xilinx Virtex2 Pro X2VP30 มีขนาด 2248 KB [7]ทำให้สามารถสร้างอะตอมเพื่อบรรจุใน RAM ได้มากที่สุดเพียง 1600 อะตอม และใช้พื้นที่ 1154 KB ซึ่งเพียงพอที่จะใช้สำหรับวิเคราะห์สัญญาณที่มีความยาว 40 ตัวอย่าง

2. โมดูล MAC : เป็นโมดูลที่ใช้ในการหาค่า inner product ระหว่างสัญญาณและอะตอมซึ่งภายในเป็นวงจร MAC

ในการหาค่า inner product ทำได้โดยการนำสัญญาณจำลองมาคูณกับอะตอมทีละ ตัวอย่าง และบวกสะสมไว้ ซึ่งสอดคล้องกับวงจร MAC ซึ่งภายในประกอบด้วยวงจรคูณและบวก

3. โมดูล cmp[n] : เป็นโมดูลที่ใช้ในการเปรียบเทียบค่า inner product ที่ได้จากโมดูล MAC ซึ่งค่าที่มากที่สุดในแต่ละกลุ่มจะถูกส่งไปยังโมดูลถัดไปเพื่อเปรียบเทียบหาค่ามากที่สุดจากค่าทั้งหมด

4. โมดูล FINAL CMP : เป็นโมดูลที่รับค่า inner product สูงสุดในแต่ละกลุ่มมา เพื่อหาค่า inner product สูงสุดจากค่าทั้งหมด และเมื่อได้ผลลัพธ์แล้ว จะทำการส่งค่า inner product สูงสุด พร้อมทั้งอะตอมที่ให้ค่า inner product สูงสุด ไปยังโมดูลถัดไป

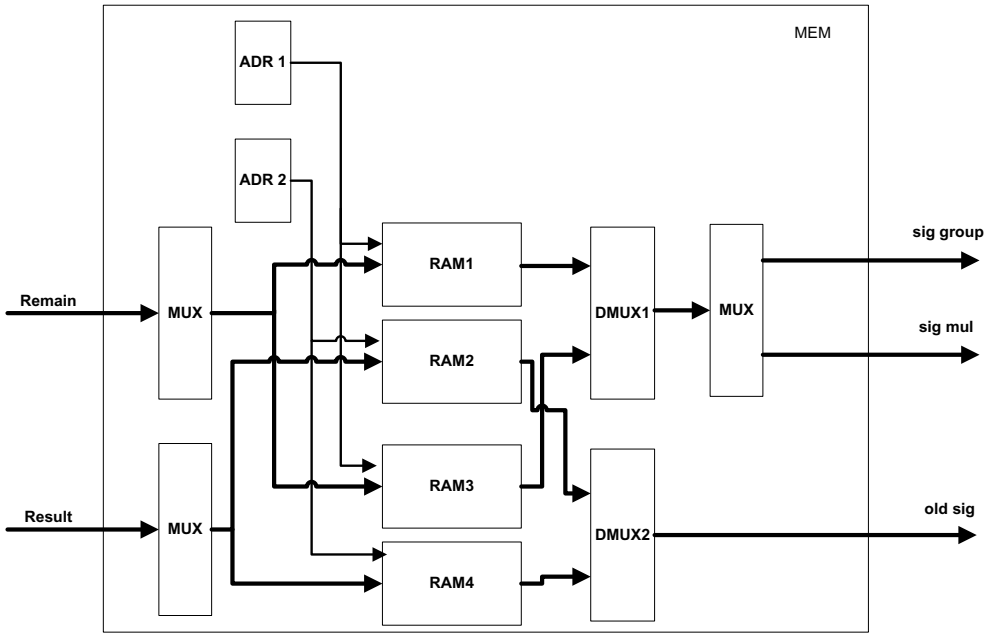
5. โมดูล MUL ADD SUB : เป็นโมดูลที่รับค่า inner product สูงสุด , อะตอมที่ให้ค่า inner product สูงสุด และสัญญาณดั้งเดิม โดยจะนำค่า inner product สูงสุดคูณกับ อะตอมที่ให้ค่า inner product สูงสุด แล้วนำไปลบออกจากสัญญาณดั้งเดิม ผลลัพธ์ที่ได้จะกลายเป็นสัญญาณตั้งต้นในรอบถัดไป และผลคูณระหว่าง inner product และอะตอม จะถูกนำไปบวกสะสมไว้ กลายเป็นสัญญาณที่แยกออกมาจากสัญญาณดั้งเดิม

6. โมดูล MEM : เป็นโมดูลที่ทำหน้าที่เก็บสัญญาณที่นำมาใช้ทดลองในรอบแรก และเก็บผลลัพธ์ที่ได้จากรอบแรกเพื่อใช้เป็นสัญญาณตั้งต้นในรอบถัดไป โดยภายในจะประกอบไปด้วย RAM ขนาด 16 40 bit จำนวน 4 ตัว ซึ่งเป็นไปตามภาพประกอบที่ 3-5

สัญญาณที่ใช้ทดลองในรอบแรกและรอบถัดไปจะถูกเก็บไว้ใน RAM1 และ RAM3 โดยจะใช้โมดูล mux สำหรับเลือก RAM ที่ใช้เก็บสัญญาณทดลองในรอบถัดไปและใช้โมดูล dmux ในการส่งสัญญาณไปยังกลุ่มของโมดูลดิกชั่นนารี หรือโมดูล MUL ADD SUB และRAM2 และ RAM4 จะทำหน้าที่เก็บสัญญาณที่ได้จากผลคูณระหว่าง inner product สูงสุด กับอะตอมที่ให้ค่า inner product สูงสุด โดยใช้โมดูล mux สำหรับเลือก RAM ที่ใช้เก็บสัญญาณที่สร้างขึ้นมาใหม่นี้เช่นเดียวกัน

สัญญาณตั้งต้นและสัญญาณที่สร้างขึ้นมาใหม่จะถูกเก็บไว้ใน RAM1 และ RAM2 และเมื่อคำนวณเสร็จในรอบแรก สัญญาณตั้งต้นในรอบถัดไป และสัญญาณที่สร้างขึ้นมาใหม่ในรอบแรกจะ

ถูกเก็บใน RAM3 และ RAM4 ตามลำดับ ซึ่งจะทำงานสลับกันแบบนี้จนกว่าจะครบตามจำนวนรอบที่ตั้งไว้



ภาพประกอบที่ 3-5 แสดงองค์ประกอบภายในโมดูล MEM

ลักษณะการทำงานของกระบวนการแม็ชชิงเพิชยูทบน FPGA เป็นไปดังภาพประกอบที่ 3-4 โดยภายในจะจัดกลุ่มของโมดูลต่างๆไว้เป็นกลุ่มๆซึ่งแต่ละกลุ่มจะประกอบไปด้วย

- กลุ่มของโมดูล MEM
- กลุ่มของโมดูล DIC
- โมดูล FINAL CMP

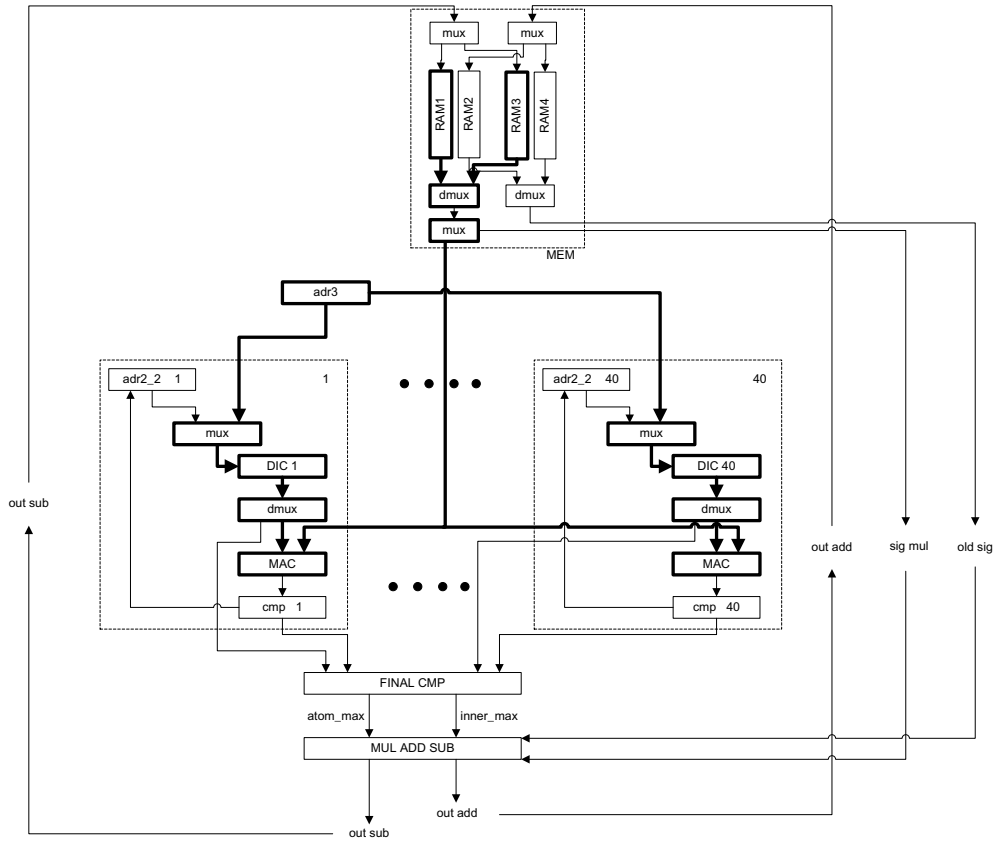
-โมดูล MUL ADD SUB

-กลุ่มของโมดูล MEM จะประกอบไปด้วย โมดูล RAM1 , RAM2 , RAM3 และ RAM4 ทำหน้าที่เก็บผลลัพธ์เพื่อใช้ในการคำนวณในรอบถัดไป โมดูล MUX เพื่อใช้เลือกRAM สำหรับเก็บข้อมูล , โมดูล ADR 1 , ADR 2 และโมดูล dmux เพื่อใช้ในการกระจายสัญญาณตั้งต้นไปยังกลุ่มของโมดูลดิกชั่นนารี

- กลุ่มของโมดูล DIC ในแต่ละกลุ่มจะประกอบไปด้วย โมดูล DIC[n] ทำหน้าที่เก็บอะตอม , โมดูล adr2_2[n] , โมดูล mux , โมดูล dmux ,โมดูล cmp[n]ทำหน้าที่หาค่าผลคูณภายในสูงสุด และโมดูล MAC[n]

กลุ่มของโมดูลดิกชันนารี แต่ละกลุ่มจะทำงานเป็นอิสระต่อกัน และโมดูลทั้งหมดจะถูกควบคุมด้วยโมดูล control ซึ่งจะทำหน้าที่ควบคุมสัญญาณ enable ของโมดูลต่างๆ

3.2.2 กระบวนการทำงานของกระบวนการแม็ชชิงเพิชยูทบน FPGA



ภาพประกอบที่ 3-6 แสดงการทำงานของกระบวนการแม็ชชิงเพิชยูท ส่วนที่ 1

1.โมดูล ADR 1 ซึ่งอยู่ภายในโมดูล MEM ทำหน้าที่สร้าง address ส่งไปยังโมดูล RAM1 และ RAM3 เพื่อเรียกสัญญาณตั้งต้นในตำแหน่งนั้นๆออกมา ซึ่งสัญญาณตั้งต้นเหล่านี้จะผ่านโมดูล dmux เพื่อกระจายสัญญาณตั้งตั้งเหล่านี้ไปยังกลุ่มของโมดูลดิกชันนารีต่างๆทุกกลุ่ม

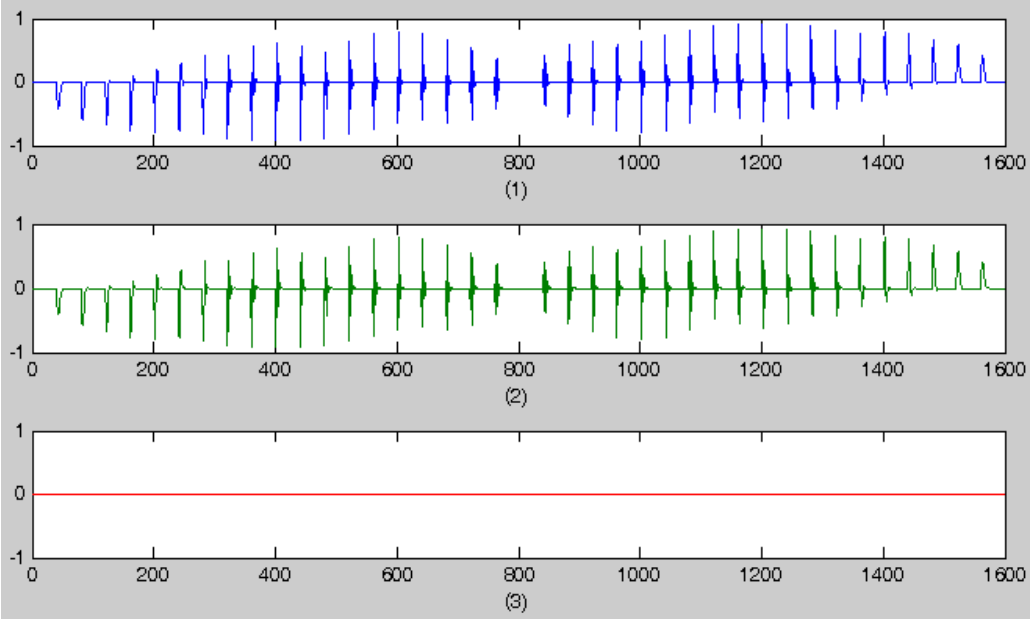
ขณะเดียวกันโมดูล ADR 3 ทำหน้าที่สร้าง address ส่งไปยังโมดูล DIC[n] เพื่อเรียกอะตอมในตำแหน่งนั้นออกมาและส่งผ่านไปยังโมดูล dmux และโมดูล MAC เพื่อทำการหาค่า inner product ระหว่างอะตอมและสัญญาณดั้งเดิมที่เรียกออกมาก่อนหน้านี้ ดังภาพประกอบที่ 3-5

เนื่องจากสัญญาณตั้งต้นและอะตอมที่สร้างจากฟังก์ชัน built_atom ในMATLAB นั้น มีค่าอยู่ในช่วง (-1,1) ทำให้เมื่อต้องนำสัญญาณเหล่านี้มาใช้บน FPGA จะต้องมีการแปลงค่าให้อยู่ในรูป

ของ binary ดังนั้น ผู้วิจัยจึงเลือกใช้ Fixed point แบบ Q1.14 โดยที่ bit MSB จะแสดงค่าเครื่องหมาย ถัดมาเป็นบิตที่แสดงค่าของ Integer ซึ่งมี 1 bit และ 14 bit สุดท้ายแสดงค่าของ Fraction

ในการสร้างอะตอมเพื่อใช้สำหรับกระบวนการแม็ชชิงเพิชยูทบน FPGA นั้น ผู้วิจัยได้ใช้ฟังก์ชัน built_atom ในการสร้างอะตอม แล้วใช้ฟังก์ชัน quantizer ใน MATLAB เพื่อกำหนดลักษณะการแปลงข้อมูลจาก double precision ให้เป็น fixed point และใช้ฟังก์ชัน num2bin ในการแปลงข้อมูลให้อยู่ในรูปแบบ fixed point ซึ่งตัวอย่างของอะตอมที่อยู่ในรูปแบบ fixed point แสดงในภาพประกอบที่ 3-6 โดยในการแปลงข้อมูลแต่ละครั้งจะทำครั้งละ 40 อะตอม จากนั้นจึงบันทึกข้อมูลที่ได้โดยใช้นามสกุลของไฟล์คือ .coe เพื่อนำไปเป็นข้อมูลของ ROM และ RAM ภายใน FPGA

โดยในภาพประกอบที่ 3-6(1) เป็นภาพประกอบที่สร้างขึ้นโดยใช้ฟังก์ชัน built_atom ภาพประกอบที่ 3-6(2) เป็นภาพประกอบที่ได้จากการอ่านค่าไฟล์ .coe ในตำแหน่งเดียวกันจาก FPGA และภาพประกอบที่ 3-6(3) เป็นผลต่างระหว่างค่าของอะตอมระหว่าง FPGA และ MATLAB



ภาพประกอบที่ 3-7 แสดงตัวอย่างอะตอมที่ใช้ใน MATLAB และอะตอมที่ใช้ในFPGA

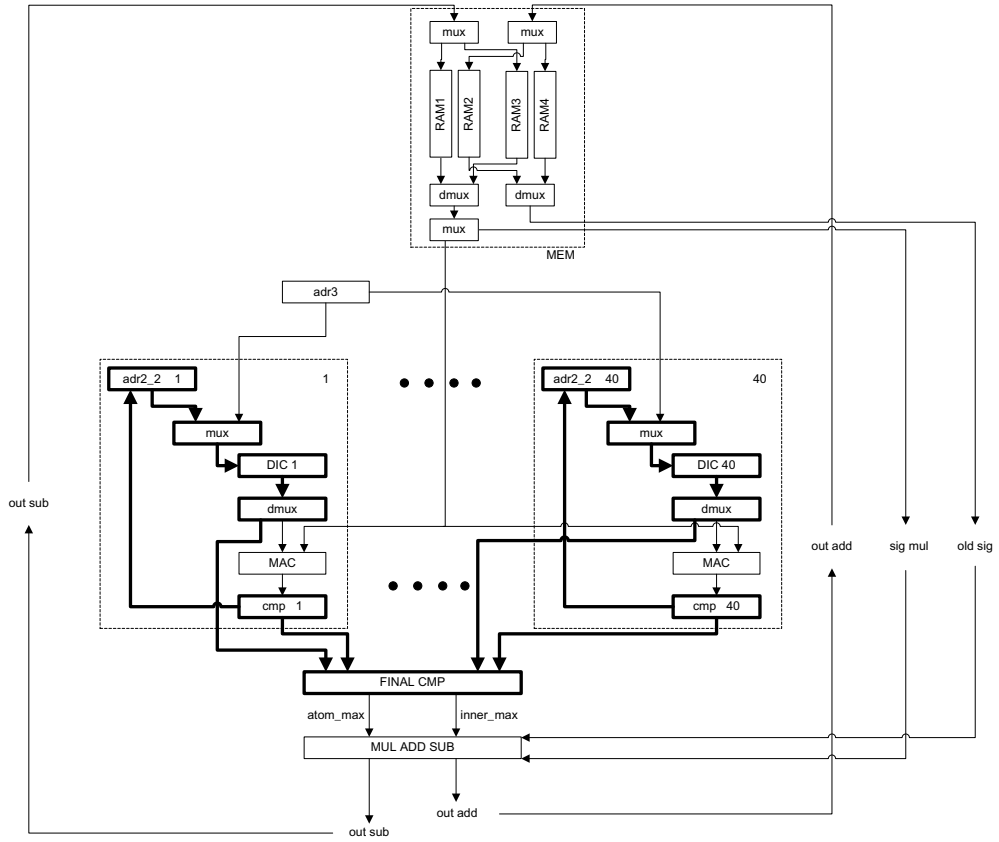
และผลลัพธ์ที่ได้จากการคูณและบวกกันภายในโมดูล MAC นั้นมีจำนวน 32 bit แต่เนื่องจากผลลัพธ์นั้นอยู่ในช่วง (-3,3) แต่เนื่องจากสัญญาณตั้งต้น และอะตอมต่างมีจำนวน 16 bit และเพื่อสะดวกในการคำนวณ ผู้วิจัยจึงนำช่วงข้อมูลในช่วง [30:15] ซึ่งมีค่าเท่ากับ fixed point ใน

รูปแบบ Q2.13 มาใช้ในการคำนวณและส่งไปยังโมดูลถัดไป ซึ่งผลลัพธ์ของการนำช่วงข้อมูลนี้มาใช้เมื่อเทียบกับค่าใน MATLAB พบว่ามีความผิดเพี้ยนน้อย

31	30	29	28		27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
1	1	1	1		1	1	0	1	0	1	1	1	0	1	1	1	1	1	1	0	1	0	0	0	0	0	0	1	1	1	0	1	1
0	0	0	0		0	0	1	0	1	0	0	0	1	0	0	0	0	0	0	1	0	1	1	1	1	1	1	0	0	0	1	1	2
0	0	0	0		0	0	1	0	1	0	0	0	1	0	0	0	0	0	0	1	0	1	1	1	1	1	1	0	0	0	1	1	3

ภาพประกอบที่ 3-8 แสดงการเลือกช่วงของข้อมูล

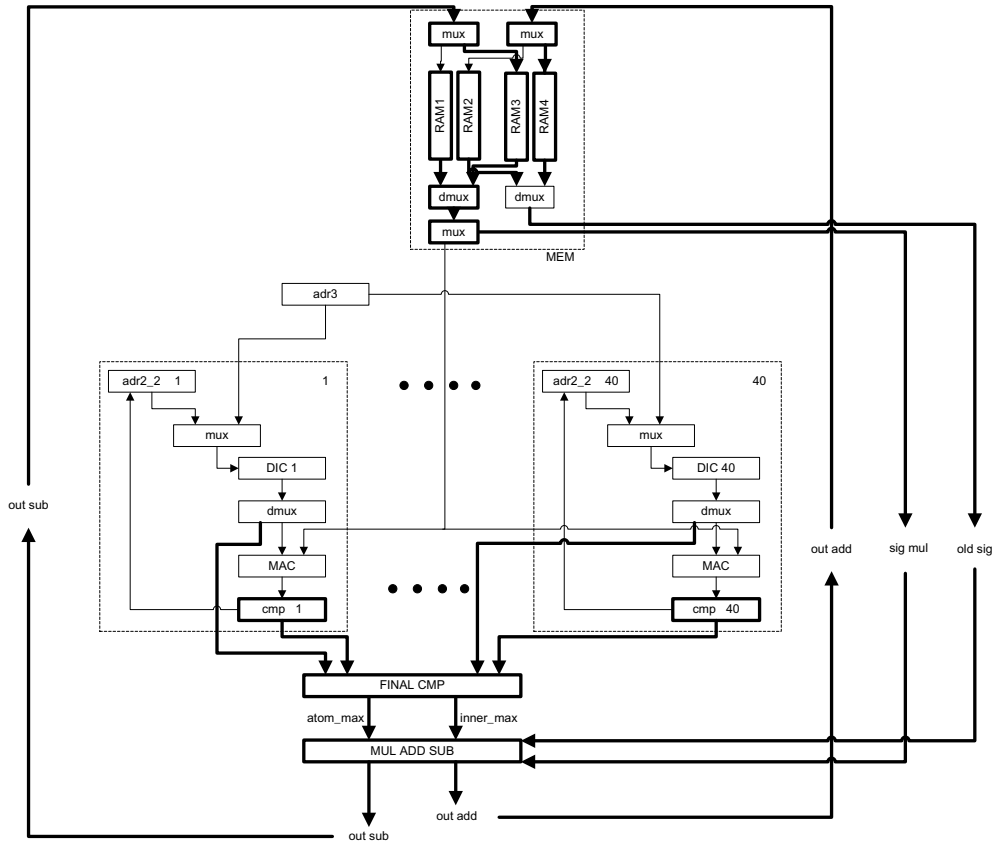
จากภาพประกอบที่ 3-7 ในบรรทัดที่ 1 เมื่อแทนด้วยจำนวนฐาน 16 คือ $FD77E81D_{16}$ ซึ่งมีค่าเท่ากับ -0.1583 ในบรรทัดที่ 2 เป็นค่า 2's complement ของบรรทัดที่ 1 ซึ่งมีค่า 0.15822 และในบรรทัดที่ 3 ส่วนที่เป็นสีส้ม เป็นช่วงของข้อมูลที่ผู้วิจัยนำมาใช้ ซึ่งมีค่า 0510_2 หรือ 0.1582



ภาพประกอบที่ 3-9 แสดงการทำงานของกระบวนการแม็ชชิงเพิชยูท ส่วนที่ 2

2.ในส่วนที่ 2 ตามภาพประกอบที่ 3-8 เมื่อโมดูล MAC ในแต่ละกลุ่มได้ค่า inner product มาแล้ว จะทำการส่งค่าต่อไปยังโมดูล cmp[n] เพื่อทำการหาค่า inner product สูงสุดในแต่ละกลุ่ม ซึ่งจะทำการเปรียบเทียบทุกๆค่า และทำการเก็บตำแหน่งของอะตอมที่ให้ค่า inner product ของแต่ละกลุ่มเอาไว้

เมื่อเปรียบเทียบครบทุกๆค่าในแต่ละกลุ่ม โมดูล cmp[n] จะทำการส่งค่า inner product ไปยังโมดูล FINAL CMP และส่งค่าตำแหน่งของอะตอมที่ให้ค่า inner product สูงสุดไปยังโมดูล adr2_2[n] เพื่อทำการเรียกค่าของอะตอม ณ ตำแหน่งนั้นๆออกมา อะตอมที่เรียกออกมานี้จะถูกส่งไปยังโมดูล final_cmp เช่นเดียวกัน



ภาพประกอบที่ 3-10 แสดงการทำงานของกระบวนการแม็ชชิงเพิชยูท ส่วนที่ 3

3.ในส่วนที่ 3 ตามภาพประกอบที่ 3-9 เมื่อโมดูล cmp[n] ในแต่ละกลุ่มได้ค่า inner product และตำแหน่งของอะตอมที่ให้ค่า inner product ของแต่ละกลุ่ม จะทำการส่งค่าดังกล่าวไปยังโมดูล FINAL CMP เพื่อทำการหาค่า inner product สูงสุดเมื่อเทียบกับทุกกลุ่ม เมื่อหาค่า inner product สูงสุดได้แล้วนั้น โมดูล FINAL CMP จะทำการส่งค่า inner product สูงสุด ไปยังโมดูล MUL ADD

SUB และส่งสัญญาณ enable กลับไปยังกลุ่มของดิกชันนารีที่มีอะตอมที่ทำให้ได้ค่า inner product ทำงาน เพื่อเรียกข้อมูลของอะตอมช่วงนั้นๆไปยังโมดูล MUL ADD SUB

โมดูล MUL ADD SUB ทำหน้าที่ในการหาผลคูณระหว่าง inner product สูงสุด และอะตอมที่ให้ค่า inner product สูงสุด และนำผลลัพธ์ที่ได้จากการคูณนั้นไปลบออกจากสัญญาณดั้งเดิมได้เป็นสัญญาณตั้งต้นในรอบใหม่เก็บไว้ใน RAM3 พร้อมทั้งนำผลลัพธ์นี้ไปบวกเก็บไว้ใน RAM4 ซึ่งadr 2 ในโมดูล mem จะทำหน้าที่สร้าง address ไว้สำหรับเก็บผลลัพธ์ทั้งสอง และเมื่อเสร็จกระบวนการก็จะวนกลับไปยังขั้นตอนที่ 1 อีกครั้ง ซึ่งในรอบถัดไปสัญญาณตั้งต้นจะอยู่ใน RAM3 แทน

แต่เนื่องจากผลคูณของ inner product สูงสุด และอะตอมที่ให้ค่า inner product สูงสุดนั้นอยู่ในช่วง (-3,3) ซึ่งอยู่ในรูปแบบ fixed point แบบ Q2.13 เมื่อนำไปลบกับสัญญาณดั้งเดิม ซึ่งอยู่ในรูปแบบ fixed point แบบ Q1.14 นั้น จะให้ผลลัพธ์ที่คลาดเคลื่อน

ดังนั้นผู้วิจัยจึงใช้วิธีการเพิ่ม bit ข้อมูลและเลื่อนตำแหน่งของทศนิยมให้ตรงกัน ดังภาพประกอบที่ 3-10

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	0	0	0	1	0	0	1	1	1	1	0	0
0	0	0	1	1	1	1	0	0	0	0	1	1	0	0	0	1
0	0	1	1	0	0	1	0	1	0	0	0	0	1	0	1	1

ภาพประกอบที่ 3-11 แสดงการเพิ่ม bit ข้อมูลและการเลื่อนทศนิยมให้ตรงกัน

ในบรรทัดที่สอง ผู้วิจัยแทนด้วยค่า 2.519278 ซึ่งเท่ากับ $0101\_0000\_1001\_1110_2$ และในบรรทัดที่สามแทนด้วย 0.9405 ซึ่งเท่ากับ $0011\_1100\_0011\_0001_2$ เมื่อนำค่าทั้งสองมาลบกันเลยนั้น จะทำให้ผลลัพธ์ที่ได้มีความคลาดเคลื่อน ดังนั้นในบรรทัดที่สองซึ่งเป็นตัวตั้ง ผู้วิจัยจึงต้องเพิ่มจำนวน bit จากเดิม 16 bit เป็น 17 bit และให้ค่า LSB มีค่าเท่ากับ 0 และในบรรทัดที่สามซึ่งเป็นตัวลบ ต้องเพิ่มเป็น 17 bit เช่นเดียวกัน และกำหนดให้ค่า MSB ในข้อมูลใหม่นี้ มีค่าเดียวกันกับ bit MSB เมื่อมีจำนวนข้อมูลเป็น 16 bit ซึ่งในที่นี้มีค่าเป็น 0 เช่นเดียวกัน

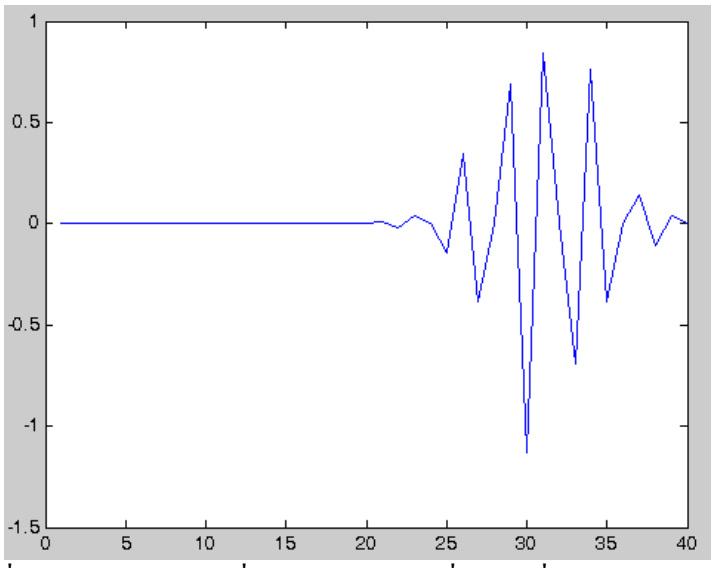
เมื่อนำค่าที่ได้ในบรรทัดที่สองลบด้วยบรรทัดที่สาม ผลลัพธ์ที่ได้ในบรรทัดที่สี่ ซึ่งมีจำนวน bit ข้อมูล 17 bit แต่สามารถตัด MSB ออกได้ ซึ่งมีค่าเท่ากับ 1.578796

บทที่ 4

ผลการทดลอง

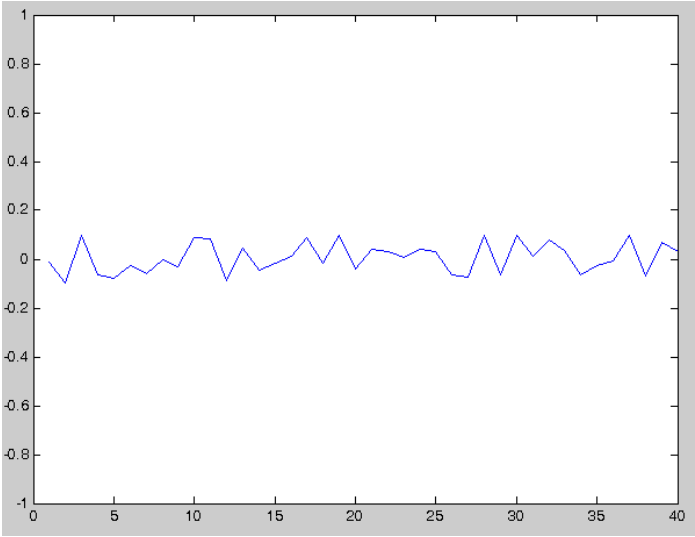
4.1การทดลองกระบวนการแม็ชชิงเพิชยูทโดยใช้ MATLAB

สัญญาณที่ใช้ทดลองสำหรับกระบวนการแม็ชชิงเพิชยูทโดยใช้ MATLAB สร้างขึ้นจากฟังก์ชัน built_atom โดยกำหนดค่า $s = 3$, $p = 31$ และ $k = 15$ ซึ่งจะได้สัญญาณที่มีลักษณะดังภาพประกอบที่ 4-1

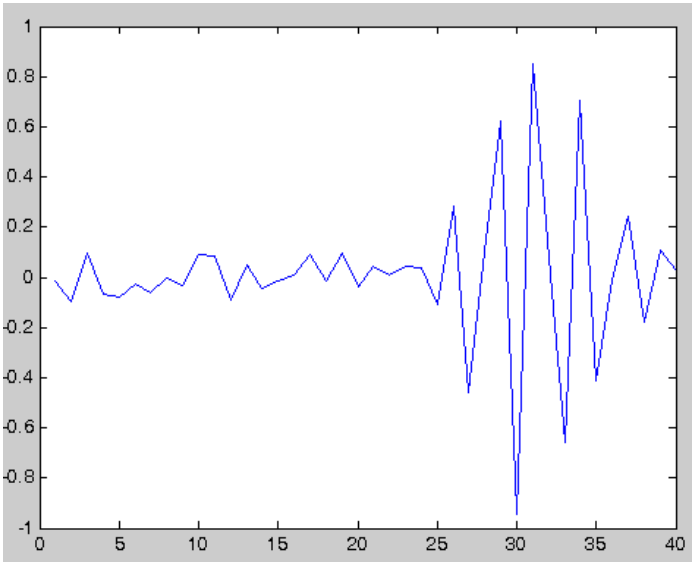


ภาพประกอบที่ 4-1 แสดงสัญญาณที่ใช้ในการทดลองซึ่งสร้างที่สร้างจากฟังก์ชัน built_atom

จากนั้นทำการเพิ่มสัญญาณรบกวน โดยใช้ฟังก์ชัน unifrnd ซึ่งเป็นฟังก์ชันที่สุ่มค่าขึ้นมาโดยสร้างให้เป็นเวกเตอร์ขนาด [1,40] แล้วนำไปคูณด้วยค่าคงที่ 0.1 เพื่อลดขนาดแอมปลิจูด ของข้อมูล ซึ่งจะได้สัญญาณรบกวนดังภาพประกอบที่ 4-2 จากนั้นนำสัณญาณที่ได้จากฟังก์ชัน built_atom และสัญญาณรบกวนมาบวกกัน จะได้เป็นสัญญาณตั้งต้นในการทดลองดังภาพประกอบที่ 4-3



ภาพประกอบที่ 4-2 แสดงสัญญาณรบกวนที่ใช้ในการทดลอง



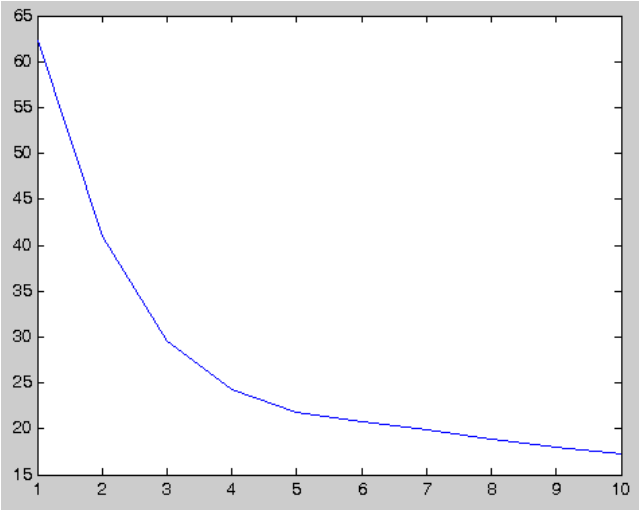
ภาพประกอบที่ 4-3 แสดงสัญญาณที่ใช้ในการทดลอง

จากภาพประกอบที่ 3-1 ในบทที่ 3เมื่อเริ่มกระบวนการแม็ชชิงเพิชยูทใน MATLAB สามารถทำได้โดยการเรียกใช้ฟังก์ชัน main_program พร้อมทั้งกำหนดตัวแปรให้เป็นตัวแปรที่ใช้เก็บค่าของสัญญาณดั้งเดิมที่แสดงในภาพประกอบ 4-3

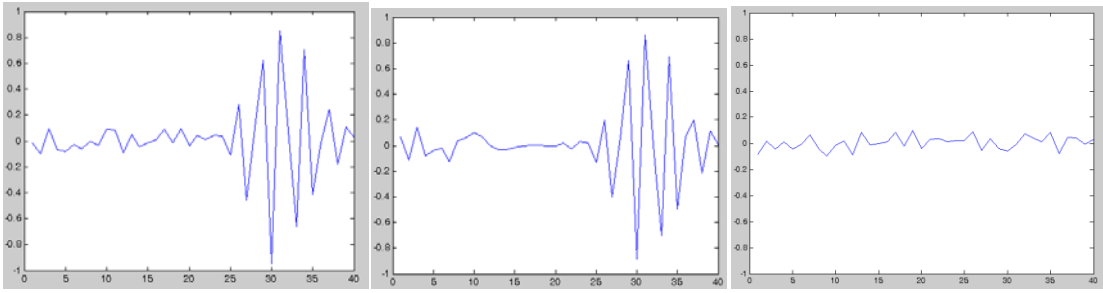
เนื่องจากกระบวนการแม็ชชิงเพิชยูทเป็นกระบวนการทำงานแบบวนรอบ ดังนั้นในแต่ละรอบก็จะมีอะตอมที่ให้ค่า inner product ไม่เหมือนกัน และอะตอมที่หาได้ในแต่ละรอบนั้นเมื่อนำมาลบออกจากสัญญาณดั้งเดิม จะทำให้สัญญาณดั้งเดิมนั้นมีพลังงานและค่า NRMS ดังภาพประกอบที่ 4-4 ลดลง ซึ่ง amplitude ของสัญญาณดั้งเดิมในแต่ละรอบมีขนาดลดลง เมื่อ

กระบวนการนี้วนรอบทำงานเรื่อยๆจนถึงจุดๆหนึ่งก็จะสามารถดึงเอาสัญญาณหลักออกมาจากสัญญาณรบกวนได้

จากภาพประกอบที่ 4-5 ในงานวิจัยนี้ ผู้วิจัยได้กำหนดจำนวนรอบการทำงานไว้ 10 รอบ เมื่อครบรอบการทำงาน เราจะได้สัญญาณใหม่ที่ไม่มีสัญญาณรบกวนดังภาพประกอบที่ 4-5(2) เมื่อเปรียบเทียบกับสัญญาณดั้งเดิมในภาพประกอบที่ 4-5(1) และสัญญาณรบกวนที่แยกได้โดยกระบวนการแม็ชชิงเพิชยูทดังภาพประกอบที่ 4-5(3)



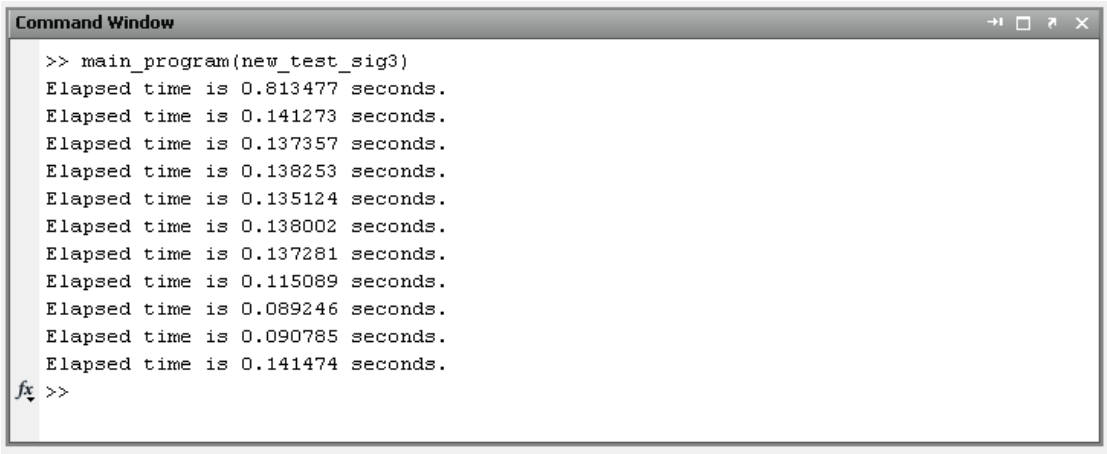
ภาพประกอบที่ 4-4 แสดงค่า NRMSสัญญาณที่ลดลงขณะทำการทดลอง



(1) (2) (3)

ภาพประกอบที่ 4-5 แสดงสัญญาณดั้งเดิม(1),สัญญาณที่ได้เมื่อเสร็จกระบวนการ(2), สัญญาณรบกวน(3)

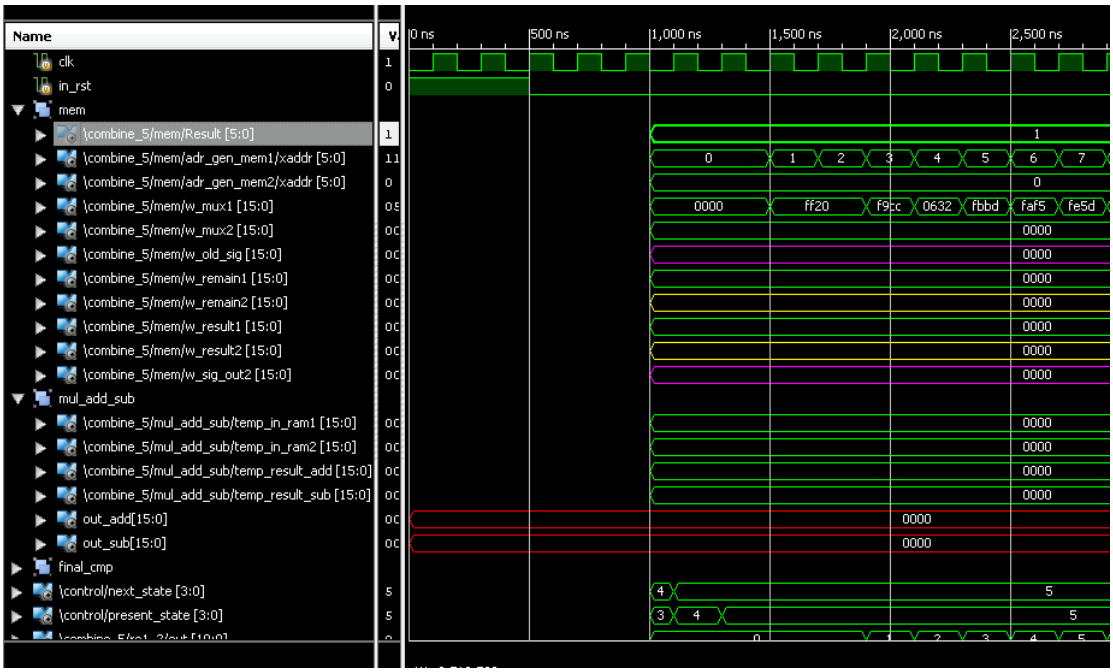
เนื่องจากกระบวนการแม็ชชิงเพิชยูทโดยใช้ MATLAB นั้นทำการหา inner product ทีละ 1 อะตอม ซึ่งจะทำให้ใช้เวลาในการทำงานมาก โดยเมื่อสัญญาณดั้งเดิมมีความยาวมาก ก็จะใช้เวลาในการทำงานมาก ในที่นี้สัญญาณดั้งเดิมมีความยาว 40 sample เวลาในการทำงานแต่ละรอบจนจบกระบวนการเป็นไปตามดังภาพประกอบที่ 4-6



ภาพประกอบที่ 4-6 แสดงเวลาที่ใช้ในการทำงานแต่ละรอบ

4.2 การทดลองกระบวนการแม็ชชิงเพิชยูทโดยใช้ FPGA

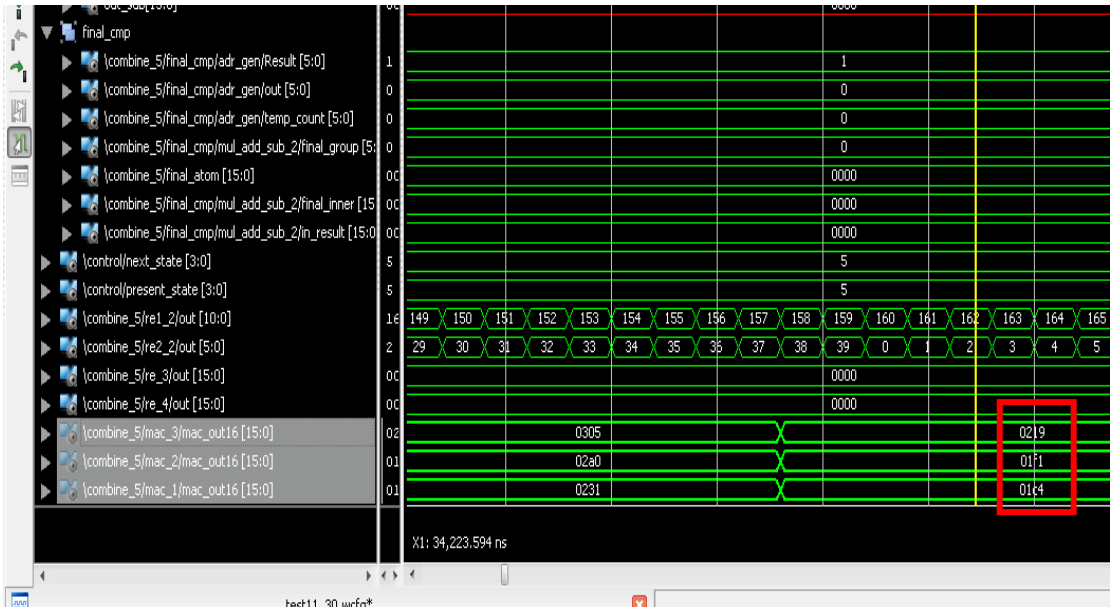
ในการทดลองการทำงานกระบวนการแม็ชชิงเพิชยูทบน FPGA ผู้ทดลองใช้ชิปของ Xilinx ตระกูล Virtex5 XC5VSX95T เริ่มต้นการทำงานโดยโมดูล control ส่งสัญญาณ reset มายังส่วน data เพื่อให้ทุกๆ โมดูลรีเซ็ทตัวเอง 500 ns



ภาพประกอบที่ 4-7 แสดงการทำงานของกระบวนการแม็ชชิงเพิชยูทโดยใช้ FPGA ในส่วนที่ 1

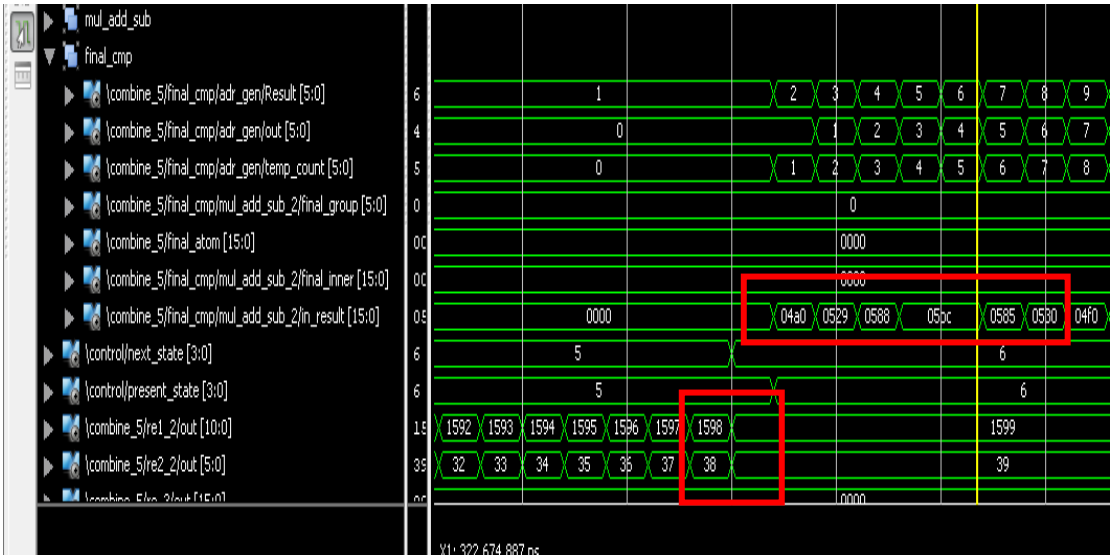
จากภาพประกอบที่ 4-7 เมื่อ reset ทุกโมดูลเป็นระยะเวลา 500 ns แล้ว โมดูล control ทำการ disable สัญญาณรีเซ็ทพร้อมทั้ง enable ให้โมดูล dic[n] ซึ่งเก็บข้อมูลดิกชันนารีและโมดูล mem

ในส่วนของ RAM1 ซึ่งเก็บข้อมูลของสัญญาณดั้งเดิม โดยเห็นได้จากข้อมูลที่ออกจาก w_mux_1 โดยมีค่า address เป็นตัวบอกตำแหน่ง เห็นได้จากข้อมูลที่ออกจาก xaddr ของโมดูล adr_gen_mem1



ภาพประกอบที่ 4-8 แสดงการทำงานของกระบวนการแม็ชชิงเพิชยูทโดยใช้ FPGA ในส่วนที่ 2

จากภาพประกอบที่ 4-8 ระหว่างที่มีการเรียกข้อมูลอะตอมจาก dic[n] และ mem นั้น ภายกลุ่มของดิกชันนารีแต่ละกลุ่มจะมีการหาค่า inner product โดยใช้โมดูล mul2_v2 และมีการหาค่า inner product สูงสุดโดยใช้โมดูล cmp[n] ไปพร้อมๆกัน แต่ค่า inner product สูงสุดนั้น จะแสดงให้เห็นเมื่อมีการหาค่า inner product ระหว่างสัญญาณดั้งเดิม และอะตอมจนครบแล้ว ซึ่งในภาพประกอบในส่วนของกรอบสีแดง แสดงให้เห็นถึงค่า inner product ที่หาได้ระหว่างการทำงาน



ภาพประกอบที่ 4-9 แสดงการทำงานของกระบวนการแม็ชชิงเพิชยูทโดยใช้ FPGA ในส่วนที่ 3

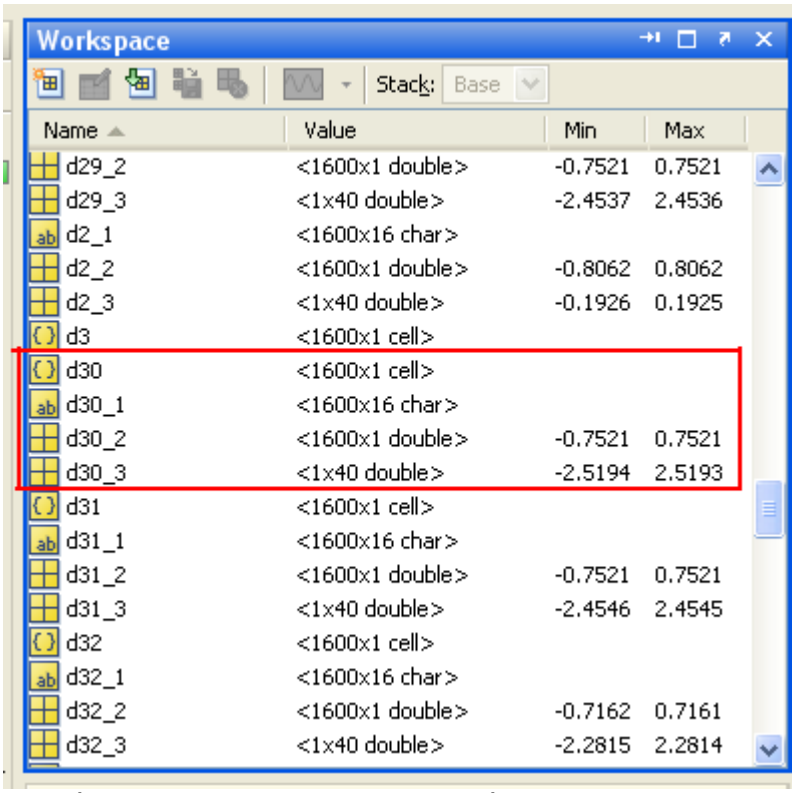
จากภาพประกอบที่ 4-9 เมื่อเรียกข้อมูลจาก dic[n] และ mem เพื่อทำการหา inner product จนครบทุกๆอะตอม ซึ่งเห็นได้จากสัญญาณ out จากโมดูล re1_2 ซึ่งเรียกข้อมูลจาก dic[n] ที่มีค่าถึง 1599 และสัญญาณ out จากโมดูล re2_2 ซึ่งเรียกข้อมูลจาก mem ที่มีค่าถึง 39 โมดูล control จะทำการ enable โมดูล final_cmp เพื่อทำการหาค่า inner product สูงสุดเมื่อเปรียบเทียบกับทุกๆกลุ่ม ซึ่งจะเห็นได้จากสัญญาณ in_result ที่เรียกค่า inner product จากทุกๆกลุ่มดิกชันนารีเข้ามาสู่โมดูล final_cmp

กลุ่ม	ค่า inner ฐาน 16	ค่าจาก FPGA	ค่าจาก MATLAB	กลุ่ม	ค่า inner ฐาน 16	ค่าจาก FPGA	ค่าจาก MATLAB
0	04DC	0.1519	0.1520	20	060A	0.1887	0.1888
1	059A	0.1750	0.1753	21	0A18	0.3154	0.3155
2	0627	0.1923	0.1925	22	0FE6	0.4968	0.4969
3	06CB	0.2123	0.2124	23	17C7	0.7430	0.7432
4	075C	0.2300	0.2302	24	21BF	1.0546	1.0546
5	07B9	0.2413	0.2415	25	2CEC	1.4038	1.4039
6	07E0	0.2461	0.2463	26	37C4	1.7427	1.7428
7	07D3	0.2445	0.2447	27	4139	2.0382	2.0384
8	079A	0.2375	0.2377	28	48FC	2.2808	2.2808
9	073E	0.2263	0.2265	29	4E83	2.4535	2.4536
10	06D0	0.2129	0.2130	30	509E	2.5193	2.5193
11	065D	0.1989	0.1990	31	4E8C	2.4546	2.4545
12	05EF	0.1854	0.1857	32	4903	2.2816	2.2814
13	058A	0.1731	0.1733	33	4137	2.0380	2.0377
14	0527	0.1610	0.1610	34	37C1	1.7423	1.7415
15	04B9	0.1479	0.1477	35	2D03	1.4066	1.4053

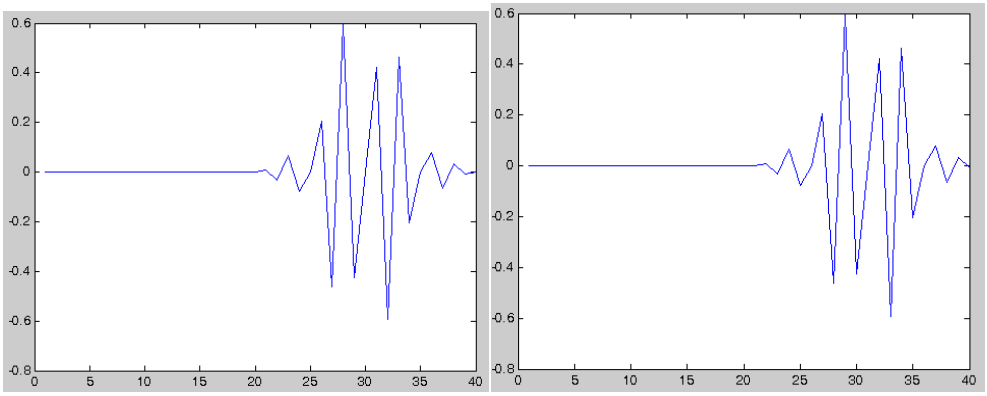
กลุ่ม	ค่า inner ฐาน 16	ค่าจาก FPGA	ค่าจาก MATLAB	กลุ่ม	ค่า inner ฐาน 16	ค่าจาก FPGA	ค่าจาก MATLAB
16	042A	0.1301	0.1302	36	221F	1.0663	1.0642
17	035D	0.1051	0.1053	37	189E	0.7694	0.7667
18	0232	0.0686	0.0687	38	189F	0.7694	0.7667
19	0363	0.1058	0.1059	39	0C16	0.3777	0.3744

ตารางที่ 3 แสดงค่า inner product ในรอบแรกเปรียบเทียบระหว่าง MATLAB และ FPGA

จากตารางที่ 1 แสดงให้เห็นค่าของ inner product ในแต่ละกลุ่มที่ได้จากกระบวนการแม็ชชิงเพิชยูทโดยใช้ FPGA ในรอบแรกซึ่งค่าที่ได้จะเป็นค่าที่อยู่ในฐาน 16 จากนั้นใช้ฟังก์ชัน bin2num ใน MATLAB ทำการแปลงค่าให้อยู่ในฐาน 10 โดยจะเห็นได้ว่าดิกชันนารีกลุ่มที่ 30 (กลุ่มที่ 31 เมื่อนับจาก 1) ให้ค่า inner product สูงสุด และตำแหน่งของกลุ่มนี้จะตรงกับตำแหน่งของกลุ่มดิกชันนารีที่คำนวณด้วย MATLAB ซึ่งจะได้กลุ่มที่ 30 (หรือกลุ่มที่ 31 เมื่อนับจาก 1) ดังภาพประกอบที่ 4-10 ซึ่งอะตอมที่ให้ค่า inner product สูงสุดที่ได้จากการคำนวณของ FPGA และ MATLAB มีค่าตรงกัน ดังภาพประกอบที่ 4-11



ภาพประกอบที่ 4-10 แสดงค่า inner product สูงสุดที่ได้จากการคำนวณของ MATLAB

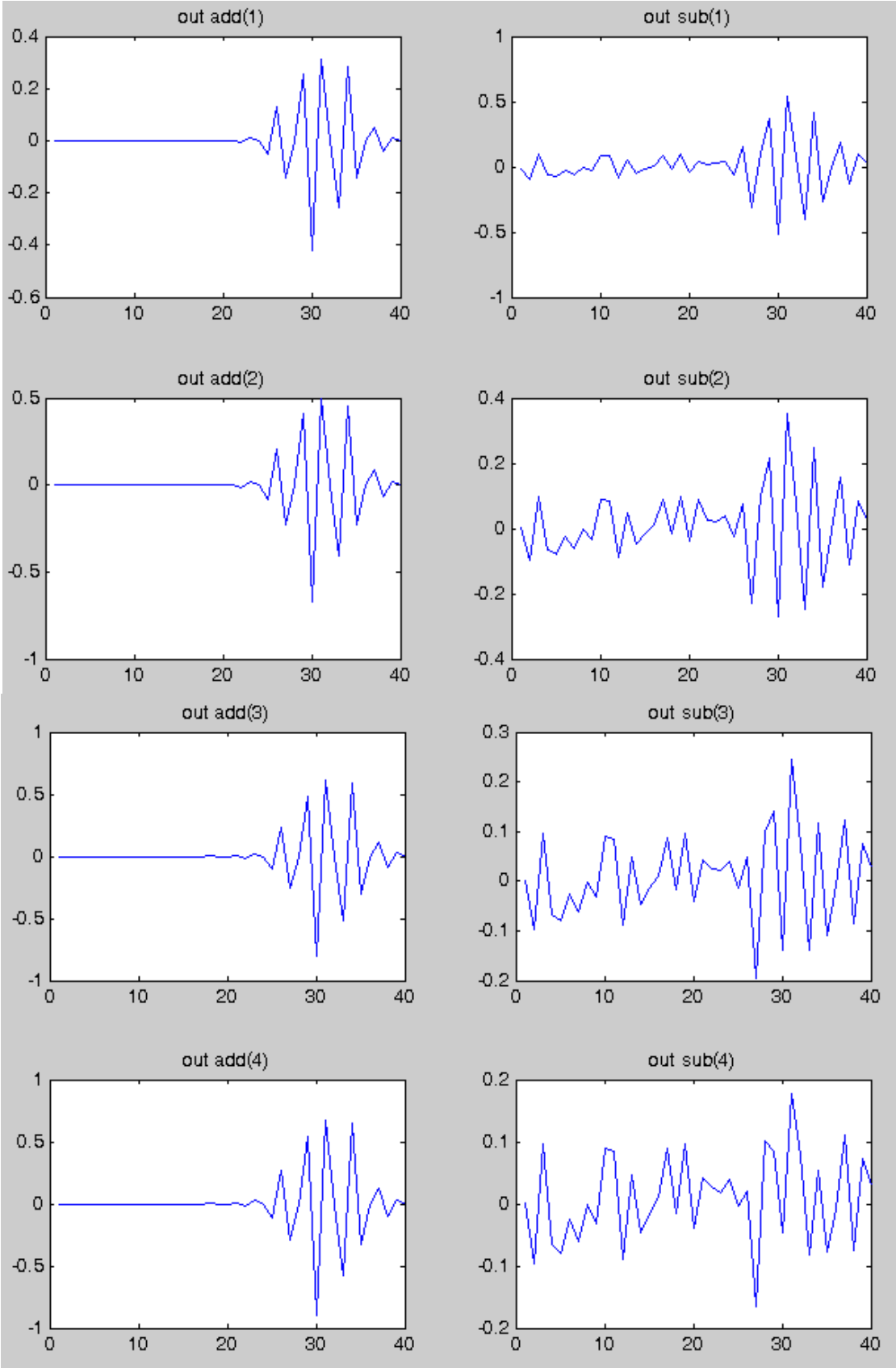


(1) (2)

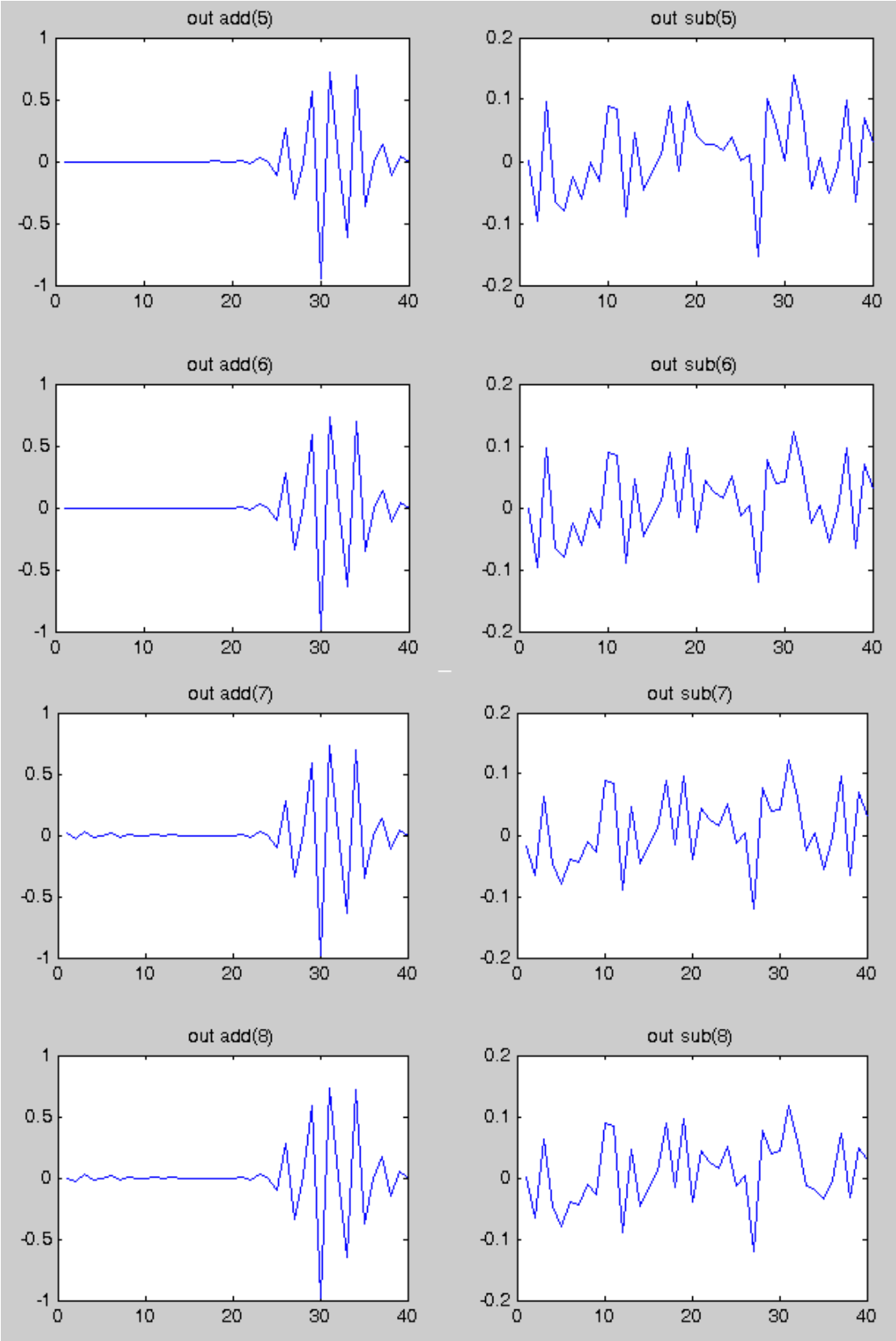
ภาพประกอบที่ 4-11 แสดงค่าอะตอมที่ให้ inner product สูงสุดในรอบแรก ,(1) อะตอมที่ได้จาก MATLAB , (2) อะตอมที่ได้จาก FPGA

เมื่อได้อะตอมที่ให้ค่า inner product สูงสุดในรอบแรกแล้วนั้นโมดูล control จะทำการ enable โมดูล mul_add_sub และโมดูล mem โดยที่โมดูล mul_add_sub จะรับค่า inner product สูงสุดและอะตอมที่ให้ค่าสูงสุดจากโมดูล final cmp เพื่อมาทำการคูณกัน และนำไปลบออกจากสัญญาณตั้งต้นจากโมดูล mem ซึ่งเก็บไว้ใน RAM1 แล้วเก็บผลลัพธ์จากการลบไว้ใน RAM3 เพื่อเป็นสัญญาณตั้งต้นในรอบถัดไป พร้อมกันนี้ค่าสัญญาณใหม่ที่เกิดจากการคูณกัน inner product สูงสุดและอะตอมที่ให้ค่าสูงสุดจะถูกในไปบวกสะสมกับสัญญาณที่เก็บไว้ใน RAM2 (ซึ่งรอบแรกมีค่าเป็นศูนย์) และเก็บไว้ใน RAM 4

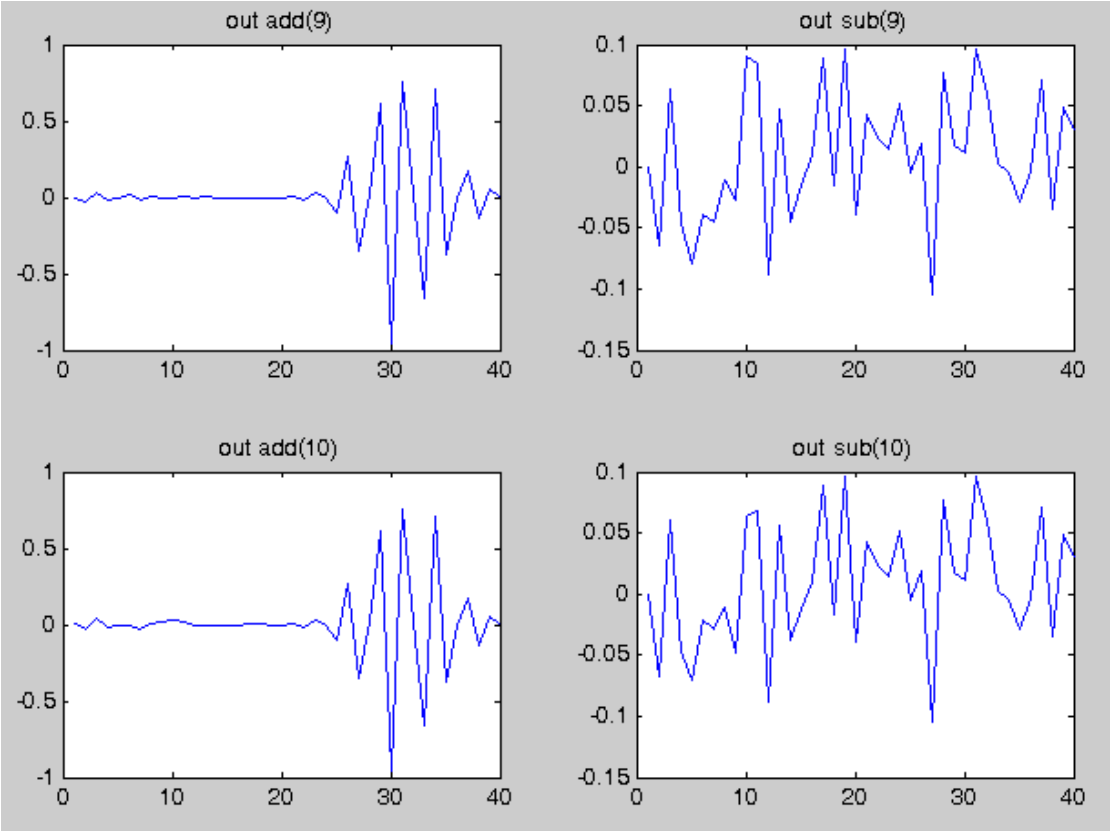
เมื่อเสร็จสิ้นกระบวนการในรอบแรก สัญญาณตั้งต้นในรอบที่สองจะถูกเก็บไว้ใน RAM3 และสัญญาณใหม่ที่ได้จากการคูณกันของ inner product สูงสุดและอะตอมที่ให้ค่าสูงสุดในรอบแรกจะถูกเก็บไว้ใน RAM4 ซึ่งในรอบที่สองนี้ กระบวนการการทำงานจะเหมือนกับที่กล่าวข้างต้นทุกประการ เพียงแต่จะสลับตำแหน่ง RAM ที่ใช้ในการเก็บข้อมูลทั้งสอง กล่าวคือ ในรอบแรกจะเรียกข้อมูลจาก RAM1 และ RAM2 และเก็บผลลัพธ์ที่ได้ใน RAM3 และ RAM4 ในรอบสองจะเรียกข้อมูลจาก RAM3 และ RAM4 และเก็บผลลัพธ์ที่ได้ใน RAM1 และ RAM2 ซึ่งจะเป็นเช่นนี้จนกว่าจะครบจำนวนรอบการทำงานที่ตั้งไว้ ซึ่งในรายงานนี้ผู้วิจัยได้กำหนดจำนวนรอบการทำงานไว้ 10 รอบ ซึ่งผลลัพธ์ของสัญญาณใหม่ และสัญญาณตั้งต้นในรอบถัดไปที่ได้จาก FPGA เป็นไปตามภาพประกอบที่ 4-12 และสัญญาณใหม่กับสัญญาณตั้งต้นในรอบถัดไปที่ได้จาก MATLAB เป็นไปตามภาพประกอบที่ 4-13



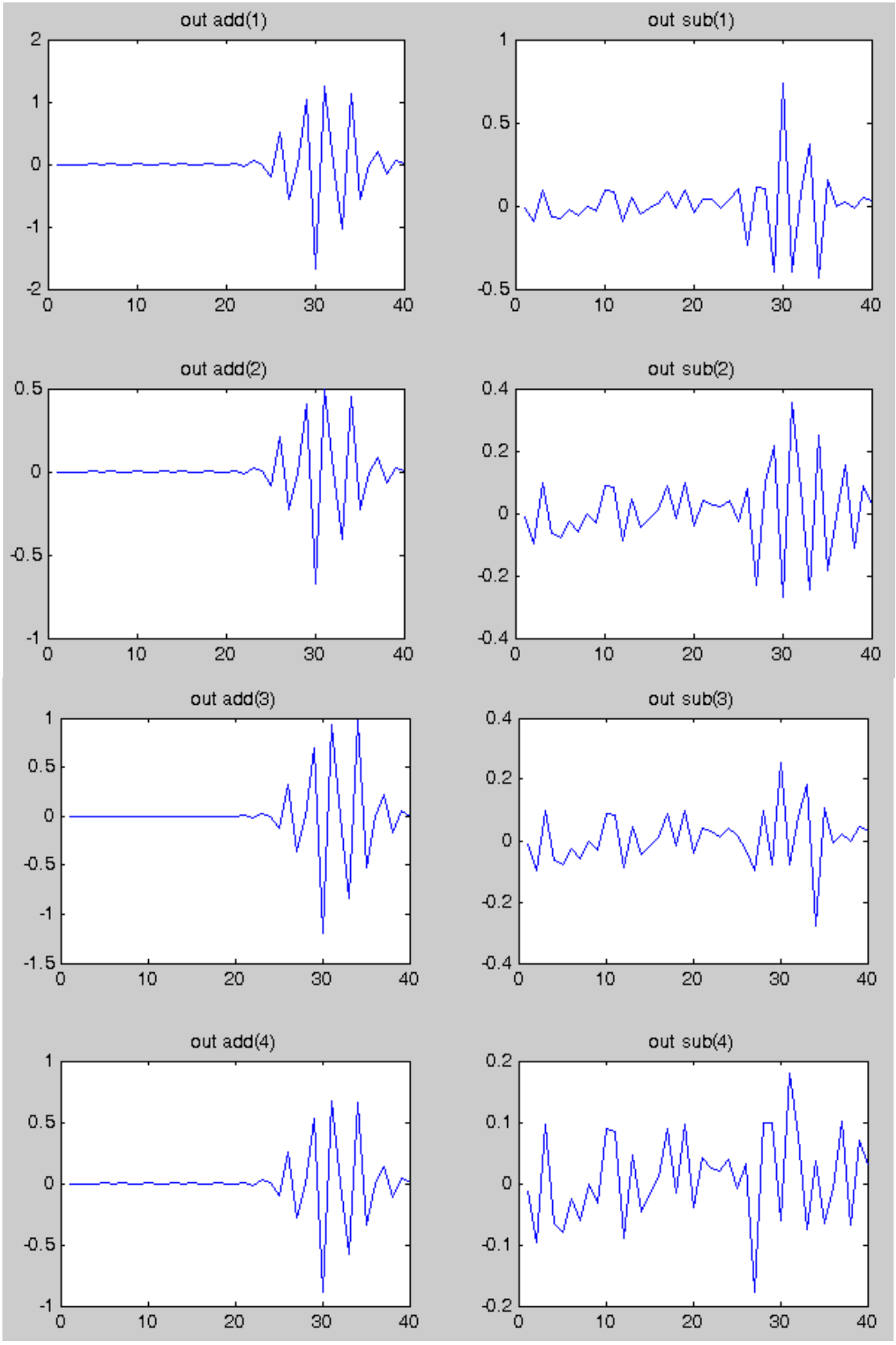
ภาพประกอบที่ 4-12 แสดงสัญญาณใหม่และสัญญาณตั้งต้นจาก FPGA



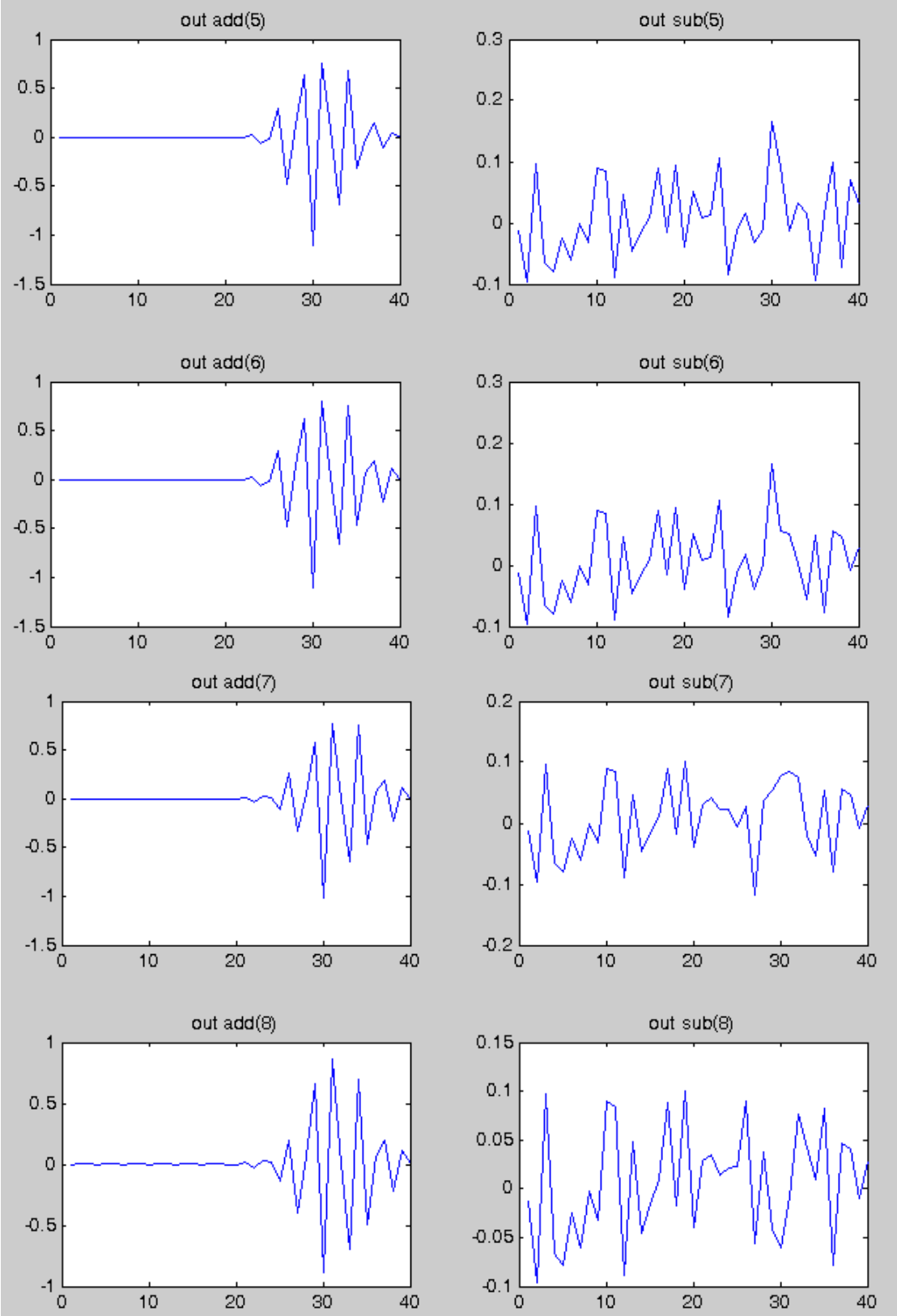
ภาพประกอบที่ 4-12 แสดงสัญญาณใหม่และสัญญาณตั้งต้นจาก FPGA (ต่อ)



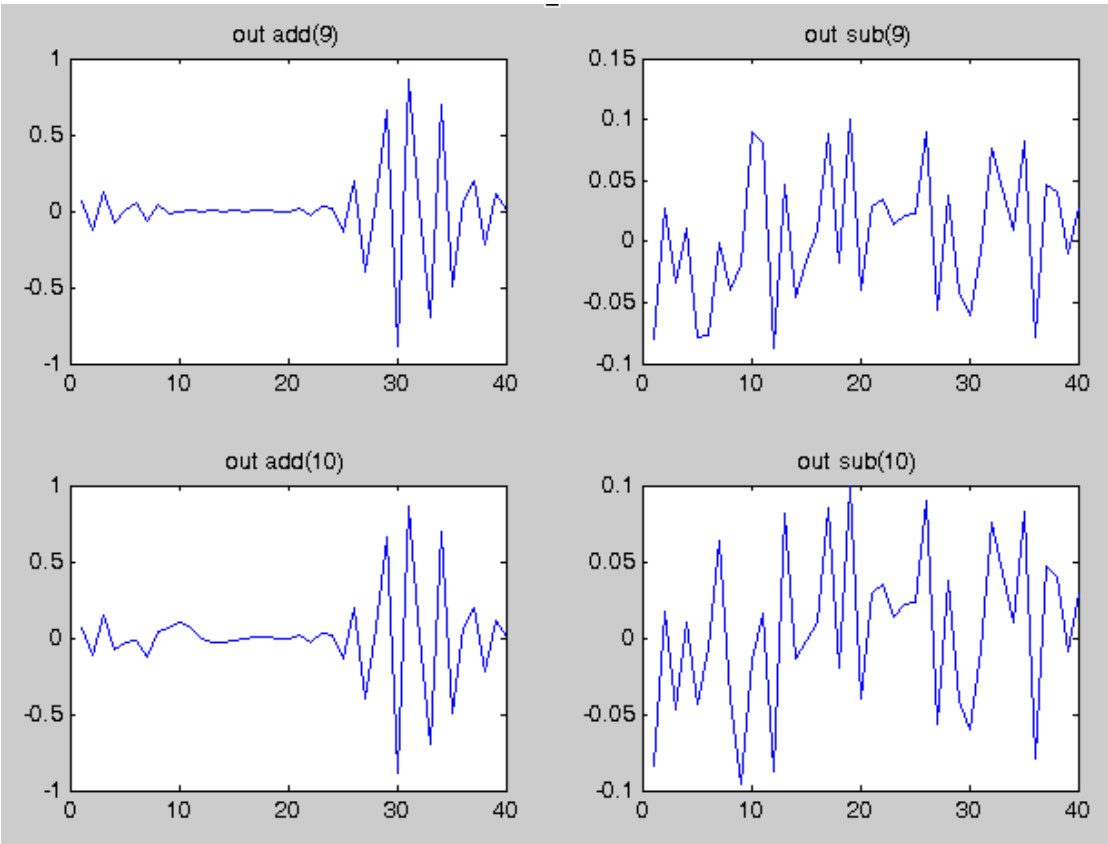
ภาพประกอบที่ 4-12 แสดงสัญญาณใหม่และสัญญาณตั้งต้นจาก FPGA (ต่อ)



ภาพประกอบที่ 4-13 แสดงสัญญาณใหม่และสัญญาณตั้งต้นจาก MATLAB



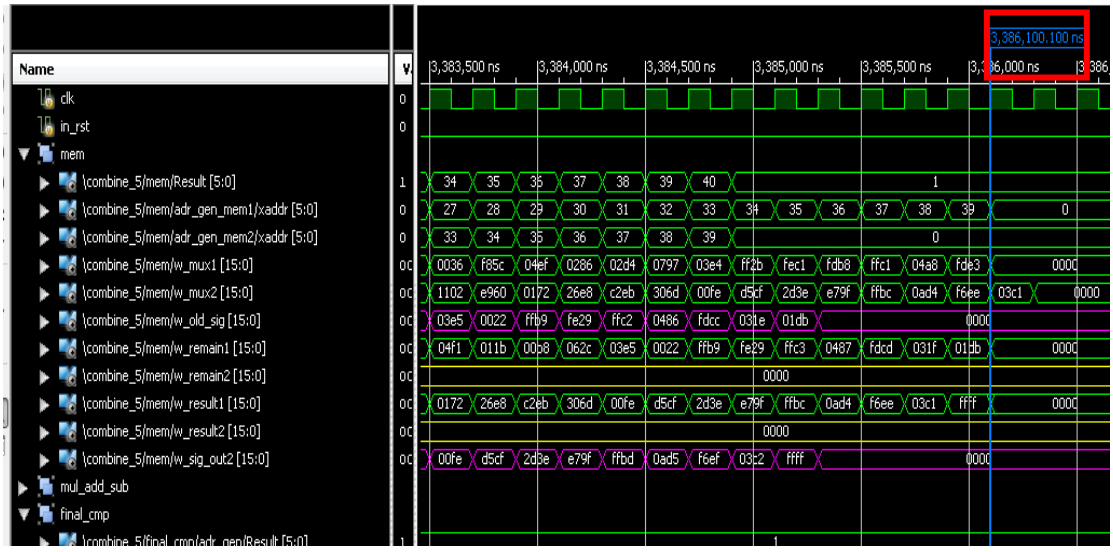
ภาพประกอบที่ 4-13 แสดงสัญญาณใหม่และสัญญาณตั้งต้นจาก MATLAB (ต่อ)



ภาพประกอบที่ 4-13 แสดงสัญญาณใหม่และสัญญาณตั้งต้นจาก MATLAB (ต่อ)

จากภาพประกอบที่ 4-12 และภาพประกอบที่ 4-13 สัญญาณ out add เป็นสัญญาณใหม่ที่ถูกสร้างขึ้น และสัญญาณ out sub เป็นสัญญาณตั้งต้นในรอบนั้นซึ่งตัวเลขในวงเล็บแสดงถึงรอบการทำงาน จะเห็นได้ว่า สัญญาณ out add ที่ได้จาก FPGA และ MATLAB เมื่อทำงานครบ 10 รอบ กระบวนการแม็ชชิงเพิชยูทสามารถดึงสัญญาณที่มีพลังงานสูง(แอมปลิจูดสูง)ออกมาได้ใกล้เคียงกัน และจะแตกต่างกันในส่วนของสัญญาณที่มีพลังงานต่ำ(แอมปลิจูดต่ำ) ซึ่งอยู่ในช่วงต้นของสัญญาณ

สำหรับสัญญาณ out sub ซึ่งเป็นสัญญาณตั้งต้นในรอบนั้นๆ พบว่าพลังงานของสัญญาณที่มีในแต่ละรอบมีค่าลดลง เห็นได้จากแอมปลิจูดของสัญญาณที่ได้จาก MATLAB และ FPGA มีค่าลดลงเมื่อรอบการทำงานเพิ่มมากขึ้น แต่ลักษณะสัญญาณที่ได้จาก MATLAB และ FPGA นี้มีลักษณะต่าง อันเนื่องมาจากการออกแบบการคำนวณใน FPGA ซึ่งมีการนำบิต[30:15] ที่ได้จากผลลัพธ์ 32 bit ซึ่งได้กล่าวไว้ในบทที่ 3 แล้วนั้นมาใช้งาน จึงทำให้ไม่สามารถแสดงผลเมื่อจำนวนมีค่าน้อยๆ ได้



ภาพประกอบที่ 4-14 แสดงเวลาที่ใช้เมื่อทำกระบวนการจำนวน 10 รอบเมื่อใช้ FPGA

จากภาพประกอบที่ 4-14 แสดงให้เห็นถึงเวลาที่ใช้เมื่อทำกระบวนการแม็ชชิงเพิชยูทครบ 10 รอบ ซึ่งใช้เวลาทั้งหมด 3.386 ms เมื่อเปรียบเทียบกับภาพประกอบที่ 4- 15 จะเห็นได้ว่ากระบวนการแม็ชชิงเพิชยูทบน MATLAB ใช้เวลา 1146.2 ms ในการทำงาน 10 รอบ ซึ่งกระบวนการแม็ชชิงเพิชยูทบน FPGA เร็วกว่ากระบวนการแม็ชชิงเพิชยูทบน MATLAB 338.15 เท่า

```
>> main_program(new_test_sig3)
Elapsed time is 0.137839 seconds.
Elapsed time is 0.137966 seconds.
Elapsed time is 0.142023 seconds.
Elapsed time is 0.138978 seconds.
Elapsed time is 0.133141 seconds.
Elapsed time is 0.090793 seconds.
Elapsed time is 0.092254 seconds.
Elapsed time is 0.090750 seconds.
Elapsed time is 0.090671 seconds.
Elapsed time is 0.091211 seconds.

ans =

    1.1462
```

ภาพประกอบที่ 4-15 แสดงเวลาที่ใช้เมื่อทำกระบวนการจำนวน 10 รอบเมื่อใช้ MATLAB

ในภาพประกอบที่ 4-16 แสดงถึงทรัพยากรที่ใช้เพื่อสังเคราะห์วงจรแม็ชชิงเพิชยูทบน FPGA

```
Device utilization summary:
---------------------------

Selected Device : 5vsx95tff1136-2


Slice Logic Utilization:
 Number of Slice Registers:            5929  out of  58880    10%
 Number of Slice LUTs:                 8627  out of  58880    14%
    Number used as Logic:              8627  out of  58880    14%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:   9936
   Number with an unused Flip Flop:    4007  out of   9936    40%
   Number with an unused LUT:          1309  out of   9936    13%
   Number of fully used LUT-FF pairs:  4620  out of   9936    46%
   Number of unique control sets:       153

IO Utilization:
 Number of IOs:                         34
 Number of bonded IOBs:                 34  out of    640     5%

Specific Feature Utilization:
 Number of Block RAM/FIFO:              42  out of    244    17%
    Number using Block RAM only:        42
 Number of BUFG/BUFGCTRLs:               1  out of     32     3%
 Number of DSP48Es:                     41  out of    640     6%
```

ภาพประกอบที่ 4-16 แสดงทรัพยากรของระบบที่ใช้ไป

และเวลาของสัญญาณนาฬิกาที่ได้จากการสังเคราะห์วงจรได้แสดงไว้ดังภาพประกอบที่ 4-17 ซึ่งจะเห็นได้ว่า สัญญาณนาฬิกาที่สั้นที่สุดในการใช้งานคือ 9.030 ns หรือที่ความถี่ 110.747 MHz

```
Timing Summary:
---------------
Speed Grade: -2

   Minimum period: 9.030ns (Maximum Frequency: 110.747MHz)
   Minimum input arrival time before clock: No path found
   Maximum output required time after clock: 2.835ns
   Maximum combinational path delay: No path found
```

ภาพประกอบที่ 4-17 แสดงสัญญาณนาฬิกาที่ได้จากการสังเคราะห์วงจร

บทที่ 5

การวิเคราะห์และสรุปผล

5.1 บทสรุป

เนื่องจากในงานวิจัยนี้ได้นำเสนอการกำจัดสัญญาณรบกวนด้วยวิธีการแม็ชชิงเพิชยูทในสัญญาณเสียง โดยที่สัญญาณจำลองนี้ได้ออกแบบให้มีลักษณะคล้ายสัญญาณเสียงหัวใจและมีอัตราการชักสัญญาณต่ำ อันเนื่องมาจากลักษณะของกาบอร์อะตอม(Gabor atom) ที่สร้างขึ้นนั้นมีลักษณะคล้ายกับสัญญาณเสียงหัวใจ และได้การออกแบบกระบวนการแม็ชชิงเพิชยูทในรูปแบบขนานกันทำได้โดยการสร้างอะตอมจำนวน 1600 อะตอมเพื่อให้เพียงพอกับการวิเคราะห์สัญญาณจำลองที่มีความยาว 40 ตัวอย่างและแบ่งอะตอมออกเป็นกลุ่มย่อยๆเรียกว่าดิกชันนารีจำนวน 40 กลุ่ม ซึ่งการทำงานแบบขนานกันนี้สามารถลดเวลาการหาผลคูณภายในที่จากเดิมต้องทำทีละหนึ่งอะตอม

จากการวิจัยพบว่ากระบวนการแม็ชชิงเพิชยูทใน FPAG ซึ่งทำงานแบบขนานกันนั้นใช้เวลาในการกำจัดสัญญาณรบกวนในสัญญาณจำลองจำนวน 10 รอบใช้เวลาเพียง 3.386 ms ซึ่งเมื่อเทียบวิธีการเดียวกันแต่ทำงานบน MATLAB นั้น พบว่าใช้เวลา 1146.2 ms จะเห็นได้ว่ากระบวนการแม็ชชิงเพิชยูทบน FPGA ทำงานเร็วกว่ากระบวนการแม็ชชิงเพิชยูทบน MATLAB

ในการทดลองนี้ผู้วิจัยได้ใช้สัญญาณนาฬิกาที่มีคาบของสัญญาณเท่ากับ 200 ns หรือ 5 MHz ซึ่งจากการทดลองนั้นเห็นได้ว่า คาบสัญญาณนาฬิกาต่ำสุดที่สามารถใช้ได้คือ 9.030 ns หรือประมาณ 110 MHz ดังนั้นกระบวนการนี้สามารถเพิ่มความเร็วในการทำงานได้อีก

5.2 ปัญหา

ผู้วิจัยได้ออกแบบกระบวนการแม็ชชิงเพิชยูทบน FPGA ให้สามารถประมวลผลสัญญาณที่มีความยาวเพิ่มจากเดิม 40 ตัวอย่าง เป็น 70 ตัวอย่าง ซึ่งไม่สามารถสังเคราะห์วงจรบน FPGA Xilinx Virtex2 Pro X2VP30 อันเนื่องมาจาก RAM ไม่เพียงพอ จึงทำการแก้ปัญหาโดยการใช้ FPGA Xilinx Virtex5 XC5VSX95T แทน ซึ่งสามารถสังเคราะห์วงจรได้ แต่เมื่อนำมาจำลองการทำงานพบว่าไม่สามารถทำงานได้ อันเนื่องมาจากหน่วยความจำในคอมพิวเตอร์ไม่เพียงพอ ซึ่งเดิมคอมพิวเตอร์ที่ใช้มีหน่วยความจำ 4 GB ภายหลังได้เพิ่มเป็น 8 GB ก็ยังไม่สามารถจำลองการทำงานได้ จึงจำเป็นต้องกลับมาใช้วงจรที่ออกแบบเพื่อประมวลผลสัญญาณ 40 ตัวอย่าง แทน ซึ่งสามารถทำงานได้

5.3 ข้อเสนอแนะ

ผู้วิจัยได้ออกแบบกระบวนการแม็ชชิงเพิชยูทบน FPGA โดยใช้ซอฟท์แวร์ Xilinx 10.1.3 ซึ่งสามารถสังเคราะห์วงจรบน FPGA Xilinx Virtex2 Pro X2VP30 ได้ แต่เมื่อนำมาจำลองการทำงานพบว่าใช้เวลานานประมาณ 1 ชั่วโมง แต่เมื่อนำวงจรเดิมที่ได้ออกแบบไว้มาใช้กับซอฟท์แวร์ Xilinx 12.1 พบว่า ไม่สามารถใช้ FPGA Xilinx Virtex2 Pro X2VP30 ได้ จึงได้เปลี่ยนมาใช้ FPGA Xilinx Virtex5 XC5VSX95T ซึ่งเมื่อนำมาจำลองการทำงาน พบว่าใช้เวลาประมาณ 15 นาที ซึ่งสามารถลดเวลาในการออกแบบวงจรลงได้มาก

บรรณานุกรม

[1] รุ่งลาวัลย์ วิไลรัตน์. 2547. การวิเคราะห์กราฟเสียงเต้นหัวใจด้วยวิธีการแมชชิงเพิชยูท. วิทยานิพนธ์วิศวกรรมศาสตรมหาบัณฑิต มหาวิทยาลัยสงขลานครินทร์

[2] Zhang, X. and Durand, L.G. 1998. Analysis-synthesis of the phonocadiogram base on the matching pursuit method. IEEE Transactions on biomedical engineering, vol 45, no 8, (August)

[3] Mallat, S. G. and Zhang, Z. 1993. Matching pursuit with time-frequency dictionaries. IEEE Trans. Signal processing. Vol.41: pp.3397-3415.

[4] Krstulovic, S. and Gribonval, R. 2006. MPTK: MATCHING PURSUIT MADE TRACTABLE. Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on, 14-19 May 2006

[5] Xu, G. and Meng, J. 2004. Signal enhancement with matching pursuit. Vehicular Technology Conference 2004. VTC2004-Fall.2004 IEEE 60th Vol 3 26-29 Sept.2004: p1986-1990.

[6] Meng, Y., Brown, A.P. and IItis, R.A. 2005. MP Core : Algorithm and Design Techniques for Efficient Channel Estimation in Wireless Applications. Proceeding. 42nd.Design Automation Conference , 2005.

[7] Virtex-IIPro and Virtex-IIProX Platform FPGAs: Complete Data Sheet. 2007. Xilinx. http://www.xilinx.com/support/documentation/data_sheets/ds083.pdf. (สืบค้นเมื่อ 3 กรกฎาคม 2553).

ภาคผนวก

บทความที่เผยแพร่ในงานประชุมวิชาการระดับชาติ

การพัฒนากระบวนการแม็ชชิงเพิชยูทด้วย FPGA

สำหรับแยกสัญญาณเสียงหัวใจ

การพัฒนากระบวนการแม็ชชิงเพิชยูทด้วย FPGA สำหรับแยกสัญญาณเสียงหัวใจ

ณัฐวีระ สงวนวงศ์
ภาควิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์
มหาวิทยาลัยสงขลานครินทร์
natthaweera@hotmail.com

ณัฏฐา จินดาเพ็ชร์
ภาควิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์
มหาวิทยาลัยสงขลานครินทร์
nattha.s@psu.ac.th

กิตติพัฒน์ ตันตระรุ่งโรจน์
ภาควิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์
มหาวิทยาลัยสงขลานครินทร์
kittipat.t@psu.ac.th

บทคัดย่อ

กระบวนการแม็ชชิงเพิชยูทเป็นกระบวนแยกสัญญาณให้อยู่ในรูปของ linear expansion ซึ่งเป็นกระบวนการที่ใช้เวลามากเพราะจะต้องหาอะตอมที่มีผลคูณภายในโดยการโปรเจคชันทีละตัว งานวิจัยนี้นำเสนอสถาปัตยกรรมการทำงานแบบขนานบน FPGA โดยทำการหาผลคูณภายในพร้อมกันทุกๆ กลุ่ม และสุดท้ายจะได้อะตอมที่มีผลคูณภายในสูงสุด จากการทำงานทดสอบบน Xilinx Virtex 2 Pro XC2VP30 พบว่าสามารถหาอะตอมที่มีลักษณะใกล้เคียงกับสัญญาณที่นำมาทดลองได้ภายในเวลา 0.329 ms

Abstract

Matching pursuit method is the process that separate signal in to linear expansion. The process use too much time because it must find atoms that has maximum inner product. In this paper, we propose a paralleled architecture for this process on an FPGA. The parallelization finds the atom in every groups and leads to the maximum inner product atom. From the simulation on Xilinx Virtex 2 Pro XC2VP30, this process can find atom that similar the test signal within 0.329 ms.

คำสำคัญ

Matching pursuit, FPGA

1. บทนำ

แม็ชชิงเพิชยูท เป็นวิธีการแยกสัญญาณให้อยู่ในรูปของ linear expansion โดยอาศัยการโปรเจคชัน (projection) สัญญาณที่ต้องการวิเคราะห์ลงบนอะตอม (atom) ที่ถูกสร้างขึ้นและเก็บไว้ในดิกชันนารี (dictionary) วิธีการวิเคราะห์สัญญาณโดยกระบวนการแม็ชชิงเพิชยูทมีการนำไปประยุกต์ใช้กับการบีบอัดข้อมูล (data compression), การจดจำรูปแบบ (pattern recognition) และการวิเคราะห์สังเคราะห์สัญญาณเสียง

ในการวิเคราะห์สัญญาณเสียงนั้น กระบวนการแม็ชชิงเพิชยูทได้นำมาใช้ในการลดสัญญาณรบกวนแบบช่วงกว้าง (wide band noise) และเมื่อเปรียบเทียบกับกระบวนการอื่น เช่น Wiener filter, Kalman filter และ Wavelet transform พบว่ากระบวนการข้างต้นมีข้อบกพร่องเพราะจำเป็นต้องทราบลักษณะของสัญญาณรบกวน [1]

เนื่องจากวิธีการนี้เป็นวิธีการที่ใช้เวลาในการทำงานนานเป็นเพราะต้องทำการหาอะตอมที่ให้ผลคูณภายใน (inner product) มีค่ามากที่สุด ดังนั้นวิธีการวิเคราะห์สัญญาณแบบนี้จึงมักจะถูกใช้งานบนคอมพิวเตอร์ [2, 3]

ในทางปฏิบัติเราสามารถเพิ่มประสิทธิภาพการวิเคราะห์สัญญาณโดยวิธีการแม็ชชิงเพิชยูทได้โดยการออกแบบดิกชันนารี ออกแบบการค้นหาอะตอมที่มีผลคูณภายในมากที่สุด หรือออกแบบโครงสร้างการทำงานในรูปแบบขนานกันซึ่งรูปแบบการทำงานแบบขนานกันนั้น [2] สามารถพัฒนาโดยใช้ FPGA (Field Programmable Gate Array) ซึ่งรองรับการทำงานแบบขนาน จากการศึกษาพบว่ากระบวนการแม็ชชิง-เพิชยูทที่พัฒนาบน FPGA นั้น มีการนำไปใช้ในกระบวนการประมวลผลสัญญาณ CDMA และเมื่อทำงานแบบขนานนั้นพบว่ากระบวนการนี้สามารถทำงานได้เร็วกว่าคอมพิวเตอร์ [4]

บทความฉบับนี้นำเสนอการออกแบบกระบวนการแม็ชชิง-เพิชยูทในรูปแบบขนานบน FPGA โดยเบื้องต้นซึ่งเป็นส่วนหนึ่งของการนำกระบวนการแม็ชชิงเพิชยูทที่ทำงานในรูปแบบขนานบน FPGA มาใช้ในการลดสัญญาณรบกวนในเสียงหัวใจ

2. ทฤษฎีที่เกี่ยวข้อง

2.1 กระบวนการแม็ชชิงเพิซยูท

กระบวนการแม็ชชิงเพิซยูท คือ ส่วนหนึ่งของกระบวนการวิเคราะห์สัญญาณ (signal analysis algorithm) ที่มีความแม่นยำสูง ที่เรียกว่า atomic decomposition [2] กระบวนการแยกสัญญาณให้อยู่ในรูปของผลรวมเชิงเส้น (linear combination) ซึ่งเป็นผลรวมของสัญญาณชิ้นเล็กๆ ที่เรียกว่าอะตอม (atom) โดยที่อะตอมเหล่านี้จะถูกนำมารวมกันกลายเป็นดิกชันนารี (Dictionary) [3]

กระบวนการแม็ชชิงเพิซยูทเป็นกระบวนการทำงานแบบวนรอบ (Iteration) ซึ่งสามารถแสดงสัญญาณต่างๆ ได้ดังนี้

$$f(t) = \sum_{n=0}^{+\infty} a_n g_{\gamma_n}(t) \qquad (1)$$

โดย $f(t)$ คือสัญญาณใดๆ ที่ต้องการวิเคราะห์

กระบวนการนี้จะเลือกอะตอมที่เหมาะสมกับสัญญาณโดยจะดูจากผลคูณภายในระหว่างสัญญาณกับอะตอมทุกๆ ตัวในดิกชันนารี ซึ่งอะตอมที่ค่าผลคูณภายในสูงสุดจะถูกเลือกมาใช้ในสมการ

$$f = < f, g_\gamma > g_\gamma + Rf \qquad (2)$$

โดย g_γ คือ อะตอมภายในดิกชันนารีที่มีค่าผลคูณภายในสูงสุด

เมื่อการทำงานรอบแรกผ่านไปก็จะนำค่า Rf มาแทนที่ f และจะทำการหาอะตอมที่มีผลคูณภายในสูงสุดในดิกชันนารีอันใหม่โดยใช้กระบวนการข้างต้นต่อไปเรื่อยๆ ซึ่งสามารถเขียนเป็นสมการดังนี้

$$R^n f = \sum_{i=0}^{n} < R^n f, g_{\gamma i} > g_{\gamma i} + R^{n+1} f \qquad (3)$$

กระบวนการนี้จะทำการแยกสัญญาณไปเรื่อยๆ ซึ่งเงื่อนไขในการหยุดนั้นจะมี 2 กรณีคือ การกำหนดจำนวนรอบการทำงาน และกำหนดค่าพลังงานต่ำสุดของสัญญาณที่เหลืออยู่ [5]

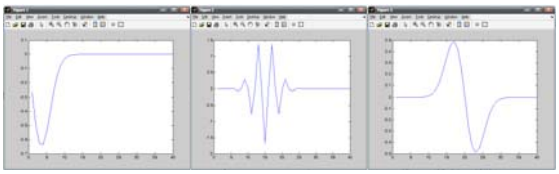
$$\left| R^i x(n) \right|^2 < \varepsilon^2 \qquad (4)$$

องค์ประกอบที่สำคัญอย่างหนึ่งของกระบวนการแม็ชชิง-เพิซยูท คือ การสร้างอะตอม (g) ซึ่งสมการมีหลายรูปแบบที่ใช้สร้างอะตอม โดยปกติอะตอมใดๆ ที่อยู่ในดิกชันนารีจะมีสมการ

$$g_{\gamma_n}(t) = \frac{1}{\sqrt{s_n}} g\left(\frac{t - u_n}{s_n} \right) e^{i\xi_n t} \qquad (5)$$

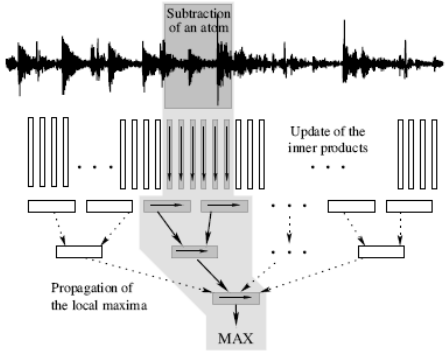
$$\gamma = (s_n, u_n, \xi_n) \qquad (6)$$

โดยที่ $g_{\gamma_n}(t)$ คืออะตอมใดๆ ที่เป็น Gaussian window function ที่อยู่ในดิกชันนารี, a_n คือสัมประสิทธิ์การยืดขยาย, s_n คือ scale factor ที่ทำให้ $|g_s| = 1$, p_n คือ ตัวเจาะจงตำแหน่ง, ξ_n คือ Frequency modulation และ u_n คือ Translation [3] ซึ่งตัวอย่างของอะตอมเป็นไปตามรูปที่ 1



รูปที่ 1 แสดงตัวอย่างอะตอมที่สร้างขึ้นโดยโปรแกรม MATLAB

วิธีการเพิ่มความเร็วในการทำงานกระบวนการแม็ชชิงเพิซ-ยูทนั้นมีด้วยกัน 2 วิธี คือ การออกแบบดิกชันนารีใหม่ [2] และการคำนวณโดยใช้รูปแบบแผนภาพต้นไม้ (Tree structure) ซึ่งวิธีการอันท้ายสุดเป็นวิธีการที่สามารถเพิ่มความเร็วในการคำนวณได้มากที่สุดดังรูปที่ 2



รูปที่ 2 วิธีการหาผลคูณภายในสูงสุดโดยใช้กระบวนการทำงานแบบแผนภาพต้นไม้ [2]

2.2 การลดสัญญาณรบกวนโดยใช้กระบวนการแม็ชชิงเพิซยูท

การลดสัญญาณรบกวนโดยใช้กระบวนการแม็ชชิงเพิชยูท คือ หลักการที่ว่าด้วยการมองสัญญาณที่เข้ามาในระบบเป็นผลรวมของสัญญาณสองชนิด

$$f = s + w \qquad (7)$$

ชนิดแรกเป็นสัญญาณหลัก (s) เป็นสัญญาณที่มีค่า coherence ratio สูง ชนิดที่สองเป็นสัญญาณรบกวน (w) เป็นสัญญาณที่มีค่า coherence ratio ต่ำ ซึ่งค่า coherence ratio สามารถหาได้จากสมการ

$$\lambda(f) = \sup_{\gamma \in \Gamma} \frac{|< f, g_\gamma >|}{\|f\|} \qquad (8)$$

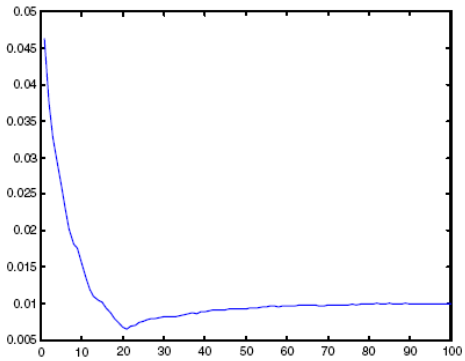
กระบวนการแม็ชชิงเพิชยูทจะดึงเอาสัญญาณที่มีค่า coherence ratio สูงสุด (s) ซึ่งมีพลังงานสูงสุดออกมาจากสัญญาณรบกวน (w) ซึ่งสัญญาณที่เหลือจะมีค่า coherence ratio ลดลงและเมื่อให้กระบวนการนี้ทำงานเรื่อยๆ กระบวนการนี้จะดึงเอาสัญญาณที่มีพลังงานสูงเรื่อยๆ จนเหลือเพียงแค่สัญญาณรบกวนเท่านั้น

การหยุดกระบวนการแม็ชชิงเพิชยูทนี้สามารถทำได้โดยกำหนดค่า

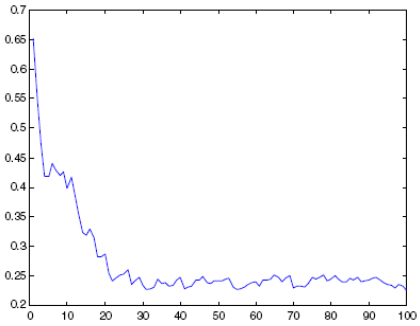
- P_K ซึ่งเป็นจำนวนรอบการทำงานของกระบวนแม็ชชิงเพิช-ยูท ซึ่งจะทำให้ค่า σ ของสัญญาณลดลง ดังรูปที่ 3

- λ_C ซึ่งเป็นค่า coherence ratio ของสัญญาณหลัก ซึ่งถ้าค่าต่ำกว่าที่กำหนดนี้ จะเป็นค่าของสัญญาณรบกวนดังรูปที่ 4

เมื่อกระบวนการทำงานจนถึงค่าใดค่าหนึ่งก่อน จะทำให้กระบวนการหยุดทำงาน [1]



รูปที่ 3 กราฟแสดงความสัมพันธ์ระหว่าง σ กับจำนวนรอบการทำงาน [1]



รูปที่ 4 กราฟแสดงความสัมพันธ์ระหว่างค่า λ_C กับจำนวนรอบการทำงาน [1]

2.3 ปัจจัยในการออกแบบกระบวนการแมชชิง-เพิชยูทบน FPGA

ในการออกแบบกระบวนการแม็ชชิงเพิชยูทบน FPGA นั้น จำเป็นต้องคำนึงถึงพารามิเตอร์หลักๆ 3 อย่าง คือ [4]

2.3.1 การกำหนดลักษณะตัวแปรที่จะใช้ในการคำนวณ

ในการประมวลผลด้วยวิธีการแม็ชชิงเพิชยูทนี้ ตัวแปรแต่ละตัวจะอยู่ในรูปการคูณกันของเมตริกซ์ – เวกเตอร์ ซึ่งสามารถแยกได้เป็นการคูณกันของเวกเตอร์ – เวกเตอร์ ซึ่งสามารถนำมาคูณกันแบบขนานได้ แต่จะทำให้ทรัพยากรที่ใช้ มากขึ้นตามไปด้วย

2.3.2 รูปแบบการนำเสนอข้อมูล

ในการประมวลผลนี้จะใช้รูปแบบข้อมูลแบบ fixed point ขนาด 16 bit ซึ่งแตกต่างจากบทความอ้างอิงที่ใช้แบบ 8 bit เนื่องจากในการสร้างอะตอมจากโปรแกรม MATLAB แล้วเก็บไว้ในหน่วยความจำของ FPAG นั้น จะต้องมีการแปลงค่าจาก floating point ให้กลายเป็น fixed point ในรูปแบบ [16 14] โดยเป็น vector ขนาด 16 bit และเป็น fraction 14 bit โดยการแปลงข้อมูลในรูปแบบนี้สามารถแสดงลักษณะอะตอมได้ใกล้เคียงกับอะตอมที่อยู่ในรูปแบบ floating point และการนำเสนอข้อมูลในรูปแบบ fixed point นั้นมีข้อดีคือ ทำงานได้เร็ว ใช้ทรัพยากรน้อย และใช้พลังงานต่ำกว่าการใช้รูปแบบข้อมูลแบบ floating point

2.3.3 รูปแบบการกระจายข้อมูล

เนื่องจากในกระบวนการแม็ชชิงเพิชยูทจำเป็นต้องประมวลผลข้อมูลจำนวนมาก ดังนั้นจึงจำเป็นต้องจัดเรียงข้อมูลเพื่อให้สามารถทำงานได้เร็วยิ่งขึ้น ซึ่งในที่นี้จะทำการเรียงข้อมูลใน

รูปแบบขนาน (parallel) และรวมโมดูลต่างๆ ให้อยู่รวมกันเป็นกลุ่มซึ่งในแต่ละกลุ่มจะประกอบไปด้วย RAM วงจรคูณ (multiplier) และ CLB

3. รายละเอียดการพัฒนา

การออกแบบกระบวนการแม็ชชิงเพิชยูทบน FPGA นี้ ได้ใช้โปรแกรม Xilinx ISE 10.1 และ Xilinx Vertex2 Pro XC2VP30 ในการออกแบบ และใช้โปรแกรม MATLAB ในการเปรียบเทียบผลการทำงาน ในการพัฒนาเริ่มต้นจากการสร้างอะตอมโดยใช้สมการ [5]

$$h(n) = \beta \cdot g\left(\frac{n-p}{s}\right)\cos\left(\frac{2\pi \cdot k}{N} \cdot n + \phi\right), \quad 0 \le n < N \qquad (9)$$

โดยแต่ละอะตอมจะมีขนาดความยาว 40 sample และมีจำนวนทั้งหมด 1600 อะตอม ซึ่งทุกอะตอมที่สร้างจาก (9) จะมี amplitude อยู่ในช่วง (-1, 1) และจะถูกแปลงให้เป็น fixed point ในรูปแบบ [16 14] อะตอมทั้งหมดนี้จะถูกแบ่งเป็นกลุ่มๆ ละ 40 อะตอม ซึ่งแต่ละกลุ่มเรียกว่าดิกชันนารี (dictionary) และเก็บไว้ในหน่วย ความจำของ FPGA

เนื่องจากอะตอมและสัญญาณที่ใช้ในการทดลองจะมีค่า amplitude อยู่ในช่วง (-1, 1) เมื่อพิจารณาในรูปแบบ floating point นั้น พบว่าการคูณกันระหว่างอะตอมและสัญญาณที่นำมาพิจารณานั้น ผลลัพธ์ที่ได้จะมีค่าอยู่ในช่วง (-1, 1) และเมื่อพิจารณาในรูปแบบ fixed point ที่เก็บในรูปแบบ [16 14] ผลคูณที่ได้จะมีจำนวน bit ผลลัพธ์ 32 bit ในรูปแบบ [32 28] ซึ่งเป็น vector ขนาด 32 bit โดยเป็น fraction จำนวน 18 bit

จากผลลัพธ์ในรูปแบบ fixed point ที่ได้นั้น มีจำนวนมากกว่าที่ได้ออกแบบไว้ ในการคำนวณผู้ทดลองจึงทำการใช้ในช่วงbit ที่ 29-14 เนื่องจากผลลัพธ์ที่ได้จะต้องอยู่ในช่วง (-1, 1) ดังนั้นการใช้ bit ผลลัพธ์ในช่วงที่ 29-14 มาใช้จึงเสมือนการนำผลลัพธ์ในรูปแบบ fixed point ในรูปแบบ [16 14] มาใช้งาน ซึ่งตัวอย่างการคำนวณแสดงในรูปที่ 5

จากรูปที่ 5 ในบรรทัดที่ 1 ผลลัพธ์ที่ได้จากการคำนวณหาค่าผลคูณภายในสูงสุดจากโมดูล MAC มีค่า -0.158226 ซึ่งเมื่อผ่านการทำ 2' compliment ในบรรทัดที่ 3 จะได้ค่าเท่ากับ 0.158226 ซึ่งเป็นผลลัพธ์ในรูปแบบ [32 28] แต่เมื่อพิจารณาในช่วงbit ที่ 29-14 (ส่วนที่เป็นสีส้ม) ซึ่งเป็น fixed point รูปแบบ [16 14] จะพบว่ามีค่า 0.158203 ซึ่งมีค่าใกล้เคียงกับ fixed point รูปแบบ [32 28]

3.1ภาพรวมของระบบ

การออกแบบกระบวนการแม็ชชิงเพิชยูทบน FPGA แบ่งขั้นตอนออกได้เป็น 3 ขั้นตอน คือ

- การหาผลคูณภายในระหว่างอะตอมกับสัญญาณที่สนใจ โดยจะทำการเรียกอะตอมที่เก็บไว้เป็นกลุ่มๆ ออกมากลุ่มละ 1 อะตอม จากนั้นหาผลคูณภายใน แล้วเปรียบเทียบภายในกลุ่ม ผลลัพธ์ที่ได้จากขั้นตอนนี้จะเป็นผลคูณภายในสูงสุดของแต่ละกลุ่ม

- การหาอะตอมที่ให้ค่าผลคูณภายในสูงสุดจากอะตอมทั้งหมด ซึ่งผลลัพธ์ที่ได้จะเป็นผลคูณภายในสูงสุด และอะตอมที่ให้ค่าผลคูณภายในสูงสุด

- การนำอะตอมที่มีค่าผลคูณภายในสูงสุดมาลบออกจากสัญญาณที่นำมาวิเคราะห์ ซึ่งผลลัพธ์ที่ได้จะเป็นสัญญาณใหม่ที่มีค่า coherence ratio สูง และสัญญาณที่คงเหลือนี้จะนำไปเป็นสัญญาณตั้งต้นในรอบถัดไป

3.2 การออกแบบและพัฒนาระบบ

รูปแบบของโครงสร้างภายในเป็นดังรูปที่ 6 ซึ่งมีลักษณะคล้ายกับ [4] และใช้โปรแกรม MATLAB ในการสร้างอะตอมโดยใช้สมการที่ (9) แล้วนำไปเก็บไว้ใน Block RAM ใน FPGA และทำการออกแบบโมดูลซึ่งทำงานตามรูปแบบหัวข้อ 3.1 ดังนี้

-กลุ่มของโมดูลดิกชันนารี มีทั้งหมด 40 กลุ่ม ทำหน้าที่หาผลคูณภายในสูงสุดของแต่ละกลุ่มระหว่างสัญญาณและอะตอมโดยการทำงานจะเป็นการทำงานแบบขนาน [2, 4] ซึ่งภายในจะประกอบด้วย Block RAM ซึ่งเก็บค่าของอะตอม , โมดูล compare , โมดูล MUX , โมดูล MAC

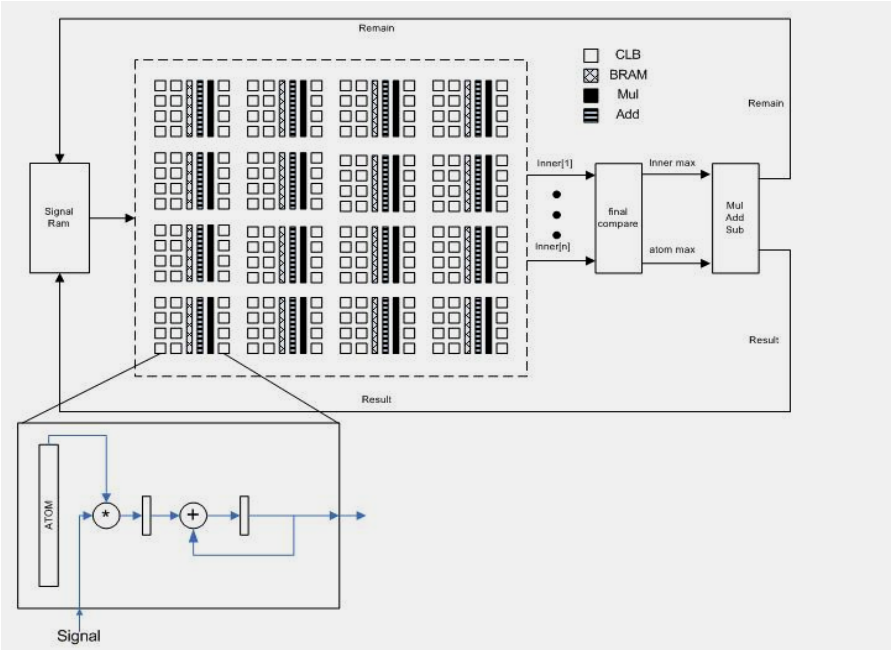
-โมดูล final compare ทำหน้าที่ในการเปรียบเทียบผลคูณภายในของทุกๆ กลุ่ม

-โมดูล mul add sub ทำหน้าที่นำค่าอะตอมที่มีค่าผลคูณภายในสูงสุดมาลบออกจากสัญญาณที่ต้องการวิเคราะห์

ผลลัพธ์ที่ได้จากการทำงานของโมดูล mul add sub คือ สัญญาณที่มีค่า coherence ratio สูงสุด ณ รอบการทำงานนั้นๆ ซึ่งจะนำมาเปรียบเทียบกับผลลัพธ์ที่ได้จากโปรแกรม MATLAB

31	30	29	28		27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
1	1	1	1		1	1	0	1	0	1	1	1	0	1	1	1	1	1	1	0	1	0	0	0	0	0	0	1	1	1	0	1	1
0	0	0	0		0	0	1	0	1	0	0	0	1	0	0	0	0	0	0	1	0	1	1	1	1	1	1	0	0	0	1	1	2
0	0	0	0		0	0	1	0	1	0	0	0	1	0	0	0	0	0	0	1	0	1	1	1	1	1	1	0	0	0	1	1	3

รูปที่ 5 แสดงตัวอย่างการคำนวณ Fixed point จากรูปแบบ [32 28] เป็น [16 14]



รูปที่ 6 แสดงโครงสร้างภายในของกระบวนการแม็ชชิงเพิชยูทใน FPGA

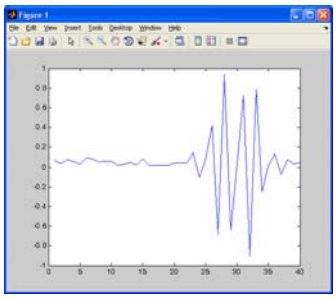
3.3 ข้อจำกัดของระบบ

เนื่องจากข้อจำกัดของ Block RAM ภายใน FPGA Xilinx XC2VP30 [6] ทำให้จำเป็นต้องออกแบบให้แต่ละอะตอมมีความยาวเพียงแค่ 40 sample ซึ่งถ้าใช้หน่วยความจำภายนอก FPGA นั้น สามารถเก็บข้อมูลของอะตอมได้มากกว่านี้ แต่จะทำให้การทำงานของกระบวนการนี้ช้าลง

4. การทดสอบการใช้งาน

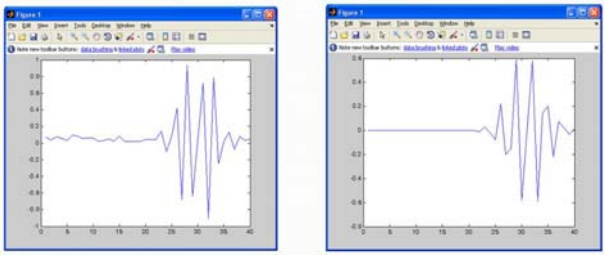
กระบวนการแม็ชชิงเพิชยูทถูกออกแบบโดยใช้โปรแกรม MATLAB ซึ่งจะแตกต่างกับการออกแบบกระบวนการนี้บน FPGA โดยที่อะตอมที่ใช้ใน MATLAB นั้น จะถูกสร้างและใช้งานทันทีและทำงานบนคอมพิวเตอร์ Core 2 Duo E8400 , 4 GB RAM แต่ใน FPGA นั้น อะตอมจะถูกสร้างขึ้นล่วงหน้าและเก็บไว้ใน Block RAM

สัญญาณที่ใช้ในการทดลองสร้างจากฟังก์ชัน random จากโปรแกรม MATLAB มีขนาดความยาว 40 sample มี amplitude อยู่ระหว่าง (-1, 1) และแปลงให้เป็น fixed point แล้วเก็บไว้ใน block RAM ภายใน FPGA ซึ่งลักษณะของสัญญาณเป็นดังรูปที่ 7



รูปที่ 7 กราฟแสดงสัญญาณที่ใช้ในการทดลอง

ในการทดลองทำกระบวนการแม็ชชิงเพิซยูทบน FPGA พบว่าในการทำงานรอบแรก กระบวนการนี้สามารถดึงเอาอะตอมที่มีลักษณะคล้ายกับสัญญาณที่นำมาวิเคราะห์ดังรูปที่ 8



รูปที่ 8 ภาพการเปรียบเทียบระหว่างสัญญาณที่ใช้ในการทดลองและอะตอมที่ได้จากกระบวนการแม็ชชิงเพิซยูท

ตารางที่ 1 แสดงเวลาที่ใช้ในการหาอะตอมในรอบแรก

	MATLAB	FPGA
Time(ms)	285	0.329

5. บทสรุป

บทความนี้ได้นำเสนอสถาปัตยกรรมแบบขนานสำหรับกระบวนการแม็ชชิงเพิซยูทบน FPGA จากการทดสอบในรอบแรกนั้นสามารถดึงเอาอะตอมซึ่งสัญญาณที่มีลักษณะคล้ายกับสัญญาณที่นำมาทดลองได้ โดยที่สัญญาณที่ได้นี้เป็นสัญญาณที่มีสัญญาณรบกวนน้อย โดยเมื่อเปรียบเทียบเวลาที่ใช้กับโปรแกรม MATLAB พบว่ากระบวนการนี้ใช้เวลาเพียง 0.329 ms ดังแสดงในตารางที่ 1

อะตอมที่ได้จากกระบวนการนี้ในรอบแรกเมื่อนำไปลบกับสัญญาณที่นำมาทดลองจะได้เป็นสัญญาณตั้งต้นในรอบถัดไป ซึ่งถ้ากระบวนการนี้ทำงานในรอบถัดไป ผลรวมของอะตอมที่ได้ในแต่ละรอบ คือสัญญาณที่นำมาทดลองที่ปราศจากสัญญาณรบกวน

6. กิตติกรรมประกาศ

งานวิจัยนี้ได้รับการสนับสนุนจากทีมวิจัย High Performance Embedded Systems and Applications ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ มหาวิทยาลัยสงขลานครินทร์

7. เอกสารอ้างอิง

[1] Xu,G., and Meng, J. 2004. Signal Enhancement with Matching Pursuit. Department of Information Engineering, North China Electric Power University.

[2] Krstulovic, S., and Gribonval, R. 2006. MPTK : Matching Pursuit Made Tractable. Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on. Vol 5 (May): III-496-499.

[3] รุ่งลาวัลย์ วิไรรัตน์. 2547. การวิเคราะห์กราฟเสียงเต้นหัวใจด้วยวิธีแม็ชชิงเพิซยูท. วิทยานิพนธ์วิศวกรรมศาตร-มหาบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า มหาวิทยาลัยสงขลานครินทร์.

[4] Meng, Y., Brown, A.P., Iltis, R.A., Sherwood T., Lee, H., and Kastner, R. 2005. MP Core : Algorithm and Design Techniques for Efficient Channel Estimation in Wireless Applications, Design Automation Conference.

[5] Zhang, X.,Durand, L.G., Senhadji, L., Lee, H.C., and Coatrieux, J.L. 1998. Analysis-Synthesis of the Phonocardiogram Based on the Matching Pursuit Method. IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING, Vol 45, no.8(AUGUST): 962-971.

[6] Virtex-II Pro and Virtex-II Pro X Platform FPGAs. เข้าถึงได้จาก: http://www.xilinx.com/support/documentation/data_sheets/ds083.pdf. สืบค้น 16 กุมภาพันธ์ พ.ศ. 2553

ประวัติผู้เขียน

ชื่อ สกุล นายณัฐวีระ สงวนวงศ์

รหัสประจำตัวนักศึกษา 5010120020

วุฒิการศึกษา

วุฒิ	ชื่อสถาบัน	ปีที่สำเร็จการศึกษา
วิทยาศาสตรบัณฑิต (ฟิสิกส์)	มหาวิทยาลัยสงขลานครินทร์	2549

ทุนการศึกษา (ที่ได้รับในระหว่างการศึกษา)

ทุนค่าเล่าเรียน คณะวิศวกรรมศาสตร์ มหาวิทยาลัยสงขลานครินทร์

การตีพิมพ์เผยแพร่ผลงาน

ณัฐวีระ สงวนวงศ์, ณัฏฐา จินดาเพ็ชร์, กิตติพัฒน์ ตันตระรุ่งโรจน์. 2553. การพัฒนากระบวนการแม็ชชิงเพิชยูทด้วย FPGA สำหรับแยกสัญญาณเสียงหัวใจ. ECTI – CARD 2010. ชลบุรี. 10-12 พฤษภาคม 2553, หน้า 31-36.