ปรับปรุงวงจร 2D-DCT เพื่อเพิ่มความเร็วในการบีบอัดรูปภาพ JPEG ด้วย FPGA

High speed 2D-DCT for JPEG Compression based on an FPGA

นเรศ ขวัญทอง

Naras Kwantong

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญา

วิศวกรรมศาสตรมหาบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์

มหาวิทยาลัยสงขลานครินทร์

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Computer Engineering

Prince of Songkla University

2553

ชื่อวิทยานิพนธ์ ปรับปรุงวงจร 2D-DCT เพื่อเพิ่มความเร็วในการบีบอัดรูปภาพ JPEG ด้วย FPGA

ผู้เขียน นายนเรศ ขวัญทอง

สาขาวิชา วิศวกรรมคอมพิวเตอร์

อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

……..…..……..…

(ผู้ช่วยศาสตราจารย์ ดร.วรรณรัช สันติอมรทัต)

อาจารย์ที่ปรึกษาร่วม

……..…..……..…

(ดร.อนันท์ ชกสุริวงค์)

คณะกรรมการสอบ

……...…ประธานกรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.ณัฏฐา จินดาเพ็ชร์)

……..……...…กรรมการ

(รองศาสตราจารย์ ดร.วัฒนพงศ์ เกิดทองมี)

……..……...…กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.วรรณรัช สันติอมรทัต)

……..……...…กรรมการ

(ดร.อนันท์ ชกสุริวงค์)

บัณฑิตวิทยาลัย มหาวิทยาลัยสงขลานครินทร์ อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษา ตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์

……..……...……..

(รองศาสตราจารย์ ดร.เกริกชัย ทองหนู)

คณบดีบัณฑิตวิทยาลัย

ชื่อวิทยานิพนธ์	ปรับปรุงวงจร 2D-DCT เพื่อเพิ่มความเร็วในการบีบอัดรูปภาพ JPEG ด้วย FPGA
ผู้เขียน	นายนเรศ ขวัญทอง
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ปีการศึกษา	2552

บทคัดย่อ

วิทยานิพนธ์เล่มนี้นำเสนอการออกแบบวงจรดิจิตอลเพื่อเพิ่มความเร็วการประมวลผลสมการ 2D-DCT ซึ่งเป็นขั้นตอนที่ซับซ้อนมากที่สุดของการบีบอัดรูปภาพแบบ JPEG สามารถที่จะนำไปพัฒนาในการเพิ่มความเร็วในการบีบอัดรูปภาพแบบ JPEG ให้เสร็จทันตามเหตุการณ์จริงที่เกิดขึ้น สำหรับการออกแบบใช้เทคโนโลยีการประมวลผลแบบขนานประมวลผลและทดสอบบนสถาปัตยกรรมของ FPGA จากการทดสอบประสิทธิภาพของวงจรการประมวลผลสมการ 2D-DCT บนอุปกรณ์ FPGA Xilinx ตระกูล Virtex4 ชิปเบอร์ XC4VFX12 จำนวนลอจิกที่ใช้ไป 36 เปอร์เซ็นต์ ปริมาณพลังงานที่ใช้ 0.257 วัตต์ จำนวนการไหลของข้อมูลภาพที่ประมวลผลได้ 369 เฟรม/วินาที เมื่อความละเอียดของภาพสีขนาด 1920 x 1080 พิคเซล

คำสำคัญ : เอฟพีจีเอ, 2D-DCT, JPEG, compression, parallel processing

Thesis Title	High speed 2D-DCT for JPEG Compression based on an FPGA
Author	Mr.Naras Kwantong
Major Program	Computer Engineering
Academic Year	2009

ABSTRACT

This paper presents the design and speed improvement of Two-Dimensional Discrete Cosine Transform (2D-DCT) for JPEG Compression using FPGA. The design is the most computationally intensive core and is the critical part in JPEG Compression. The parallelism has been employed to improve the speed aiming to support real-time application. The preliminary result shows that our design has slices utilization of 36%, power consumption of 0.257 W running at 287 MHz based on Xilinx FPGA XC4VFX12. The compression rate is 369 frame per second for color image.

Keywords : 2D-DCT, FPGA, JPEG, compression, parallel processing

กิตติกรรมประกาศ

ขอขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร.วรรณรัช สันติอมรทัต อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ได้ให้คำปรึกษาต่างๆ ให้ความรู้พร้อมทั้งชี้แนะแนวทางการทำงาน ตลอดจนช่วยตรวจสอบและแก้ไขวิทยานิพนธ์ฉบับนี้ให้เป็นไปอย่างสมบูรณ์

ขอขอบพระคุณ ดร.อนันท์ ชกสุริวงค์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม ที่ให้คำปรึกษาต่างๆ ให้ความรู้และช่วยตอบปัญหาที่เกิดขึ้น ตลอดจนช่วยตรวจสอบและแก้ไขวิทยานิพนธ์ฉบับนี้ให้เป็นไปอย่างสมบูรณ์

ขอขอบพระคุณ รองศาสตราจารย์ ดร.วัฒนพงศ์ เกิดทองมี และผู้ช่วยศาสตราจารย์ ดร.ณัฏฐา จินดาเพ็ชร์ ที่ให้เกียรติเป็นกรรมการสอบวิทยานิพนธ์ และให้คำแนะนำต่างๆ เพื่อนำไปแก้ไขวิทยานิพนธ์ให้เสร็จอย่างสมบูรณ์

ขอขอบคุณพระคุณ คณาจารย์และบุคลากรภาควิชาวิศวกรรมคอมพิวเตอร์ทุกท่านที่ให้การช่วยเหลือและเป็นที่ปรึกษาตลอดช่วงทำวิทยานิพนธ์

ขอขอบพระคุณ บัณฑิตวิทยาลัย มหาวิทยาลัยสงขลานครินทร์ ที่ได้มอบทุนการศึกษาและทุนสนับสนุนการทำวิจัย

ขอขอบคุณ เพื่อนๆ นักศึกษาปริญญาโท ที่เป็นกำลังใจและให้คำแนะนำตลอดช่วงการทำวิทยานิพนธ์

และท้ายสุดนี้ ขอน้อมรำลึกถึงพระคุณของ บิดา มารดา และครอบครัว ที่ส่งเสริมและให้การสนับสนุนจนกระทั่งข้าพเจ้าประสบความสำเร็จในการศึกษา

นเรศ ขวัญทอง

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

รายการตาราง

ตารางที่ **หน้า**

รายการภาพประกอบ

ภาพประกอบ **หน้า**

สัญลักษณ์คำย่อและตัวย่อ

2D-DCT	Two-Dimensional Discrete Cosine Transform
2D-IDCT	2-Dimention Invert Discrete Cosine Transform
DWT	Discrete Wavelet Transform
FPGA	Field Programmable Gate Array
HDL	Hardware Description Language
JPEG	Joint Photographic Experts Group
MSE	Mean Square Error
PSNR	Peak Signal to Noise Ratio
RMSE	Root Mean Square Error
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
ZTE	Zero Tree Encoding

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของวิทยานิพนธ์

รูปแบบของการบีบอัดภาพได้รับความนิยมเพื่อให้ไฟล์ภาพมีขนาดเล็กลงหรือใช้เนื้อที่ในการจัดเก็บน้อยลง ซึ่งการบีบอัดภาพมีความสำคัญมากในระบบการสื่อสารและการจัดเก็บ เช่น การรับส่งภาพผ่านระบบเครือข่ายมือถือ และระบบการจัดเก็บรูปภาพในอินเตอร์เน็ต เป็นต้น เมื่อกล่าวถึงพื้นที่จัดเก็บจึงหมายถึงพื้นที่ในหน่วยความจำนั่นเอง รูปแบบการบีบอัดภาพแบบ JPEG (Joint Photographic Experts Group) เป็นอัลกอลิทึมที่มีประสิทธิภาพในการบีบอัดสูงและได้รับความนิยมอย่างแพร่หลาย และยังเป็นอัลกอลิทึมพื้นฐานของการบีบอัดรูปภาพเคลื่อนไหวหรือที่เรียกกันว่า MJPEG

เทคโนโลยีสมองกลฝังตัวเริ่มเข้ามามีบทบาทสำคัญในปัจจุบัน ระบบสมองกลฝังตัวคือระบบประมวลผลด้วยคอมพิวเตอร์ที่มีขนาดเล็กพกพาง่าย ใช้พลังงานจากแบตเตอรี่ และมีขนาดหน่วยความจำที่จำกัด ดังนั้นหากมีการนำการบีบอัดรูปภาพมาใช้ในระบบสมองกลฝังตัวจะต้องคำนึงถึงข้อจำกัดในเรื่องของความสามารถในการประมวลผล หรือขนาดของหน่วยความจำเป็นสำคัญ การบีบอัดภาพแบบ JPEG เป็นที่นิยมในปัจจุบันขั้นตอนและวิธีการที่ซับซ้อน จึงส่งผลให้เกิดความล่าช้าในการประมวลผลตามไปด้วย งานบางชนิดต้องการข้อมูลการบีบอัดแบบทันทีทันใด (Real Time) และต้องการความละเอียดที่สูงเพื่อความถูกต้องของชิ้นงาน อาทิเช่น ระบบกล้องรักษาความปลอดภัยที่ต้องการความถูกต้อง รวดเร็ว และเพื่อเป็นการประหยัดอุปกรณ์เก็บข้อมูล ความเร็วในการบีบอัดรูปภาพจึงเข้ามามีบทบาทสำคัญมากในการพัฒนาอุปกรณ์

นอกจากรูปแบบการบีบอัดรูปภาพแบบ JPEG แล้ว ปัจจุบันยังมีรูปแบบการบีบอัดรูปภาพชนิดอื่นที่มีชื่อว่า JPEG2000 รูปแบบการบีบอัดภาพนี้ถูกพัฒนาขึ้นในปี ค.ศ. 2000 ซึ่งเป็นบริษัทเดียวกับการบีบอัดรูปภาพแบบ JPEG การบีบอัดรูปภาพแบบ JPEG2000 เป็นรูปแบบที่ใช้เทคนิคแตกต่างออกไปจากการบีบอัดรูปภาพแบบ JPEG กล่าวคือ การบีบอัดรูปภาพแบบ JPEG ใช้เทคนิคการจัดการรูปแบบข้อมูลโดยใช้วิธี 2D-DCT (Two-Dimensional Discrete Cosine Transform) ส่วนการบีบอัดรูปภาพแบบ JPEG2000 ใช้เทคนิคการจัดการรูปแบบข้อมูลโดยใช้วิธี DWT (Discrete Wavelet Transform) ซึ่งทั้ง 2 รูปแบบมีจุดเด่นที่แตกต่างกัน

Khurram [7] ได้ทำการทดสอบและเปรียบเทียบประสิทธิภาพเทคนิคการจัดการข้อมูลระหว่าง 2D-DCT กับ DWT ในด้านต่างๆของการบีบอัดข้อมูลภาพ ซึ่งมีรายละเอียดในการเปรียบเทียบประสิทธิภาพเทคนิคการจัดการข้อมูลแบ่งออกเป็น 3 ส่วนหลักๆ ดังต่อไปนี้

1. เปรียบเทียบประสิทธิภาพการบีบอัดข้อมูลชนิดรูปภาพด้วยค่า PSNR

PSNR (Peak Signal to Noise Ratio) เป็นสมการการคำนวณสามารถนำไปใช้เปรียบเทียบข้อมูลระหว่างข้อมูลต้นฉบับกับข้อมูลที่ผ่านกรรมวิธีแปลงข้อมูล(Transform) เพื่อทดสอบหาค่าความคลาดเคลื่อนของข้อมูล ซึ่งสำหรับในการเปรียบเทียบประสิทธิภาพการบีบอัดข้อมูลชนิดรูปภาพ การใช้เทคนิค DWT เมื่อทดสอบบีบอัดด้วยอัตราส่วนที่เท่ากันจะได้ผลของค่า PSNR ที่ดีกว่าการใช้เทคนิค 2D-DCT อีกทั้งการมองด้วยสายตามนุษย์ภาพที่ได้จากการบีบอัดด้วยเทคนิค DWT จะเห็นรูปภาพที่มีคุณภาพรายละเอียดที่ดีกว่าการบีบอัดด้วย 2D-DCT ผลการเปรียบเทียบค่า PSNR จากการบีบอัดรูปภาพระหว่าง JPEG กับ JPEG2000 ของ Khurram ซึ่งเป็นค่าเฉลี่ยจากการบีบอัดรูปภาพหลายๆ แบบ แสดงได้ดังตารางที่ 1-1

ตารางที่ 1-1 แสดงผลเปรียบเทียบค่า PSNR จากการบีบอัดรูปภาพที่ทดสอบโดย Khurram

รูปแบบการบีบอัดรูปภาพ	**PSNR (dB)**
JPEG	35.90
JPEG2000	37.50

2. เปรียบเทียบประสิทธิภาพการบีบอัดข้อมูลชนิดภาพเคลื่อนไหวด้วยค่า PSNR

การบีบอัดข้อมูลชนิดภาพเคลื่อนไหวได้มีการพัฒนานำเทคนิค DWT เข้ามาใช้การบีบอัดภาพเคลื่อนไหวโดนมีจุดประสงค์ที่ว่าจะนำเข้ามาแทนที่การบีบอัดด้วยเทคนิด 2D-DCT แต่จากการทดสอบประสิทธิภาพและนำมาเปรียบเทียบของ Khurram ปรากฏว่าผลจากค่า PSNR ที่บีบอัดด้วยบิตเรท (Bit rate) และเฟรมเรท (Frame rate) ที่เท่ากัน การใช้เทคนิค DWT คือการบีบอัดแบบ Zero Tree Encoding (ZTE) กับเทคนิค 2D-DCT คือ การบีบอัดแบบ MPEG-4 แทบจะไม่เห็นความแตกต่างกันของคุณภาพความคมชัดของภาพ เพราะค่า PSNR ที่ได้มีค่าเท่ากันมากดังตารางที่ 1-2 โดยมีการแบ่งข้อมูลเป็น 2 ส่วนตามมาตรฐานภาพสีบนระบบดิจิตอล คือ ค่าความสว่างของแสง (Y) กับค่าเฉลี่ยของความเข้มของสี (C)

ตารางที่ 1-2 แสดงผลเปรียบเทียบค่า PSNR การบีบอัดภาพเคลื่อนไหวทดสอบโดย Khurram

ภาพเคลื่อนไหวที่ใช้ทดสอบ	Bit rate	Y/C	MPEG-4 PSNR (dB)	ZTE PSNR (dB)
Akiyo 10 fames/s	24 kb/s	Y	37.46	36.64
		C	42.15	44.02
Hall Monitor 10 frames/s	24 kb/s	Y	34.46	34.11
		C	39.38	39.63
Coast Guard 7.5 frame/s	48 kb/s	Y	29.74	29.20
		C	40.78	40.88
News 7.5 frames/s	48 kb/s	Y	35.10	35.17
		C	39.11	40.46

นอกจากนั้นการใช้เทคนิค DWT ในการบีบอัดข้อมูลภาพเคลื่อนไหวทำให้เกิดปัญหาขึ้น 2 กรณี โดยเฉพาะสำหรับภาพเคลื่อนไหวที่ใช้บิตเรทที่ต่ำ มีรายละเอียดดังนี้

1. ปัญหาเรื่องความคมชัดของเส้นขอบภาพจะลดลงอันเนื่องมาจากการทำบล๊อคแมทชิ่ง (Block-matching) ทำให้ภาพที่ได้มีลักษณะเบลอ (Blur)
2. การเปลี่ยนรูปภาพของภาพเคลื่อนไหว (Picture-Refreshing) เนื่องจากที่ภาพมีลักษณะเบลอจึงทำให้การเปลี่ยนรูปภาพบางรูปส่งผลต่อภาพเคลื่อนไหวที่แสดงให้เห็นมีความแตกต่างกันน้อย ทำให้การเปลี่ยนแปลงเฟรมไม่สามารถเปลี่ยนได้ทุกความแตกต่างของภาพ

เหตุผลอันเนื่องมาจากปัญหาทั้ง 2 กรณีจึงส่งผลให้การบีบอัดรูปภาพด้วย DWT สามารถทำได้ยากกว่าการบีบอัดรูปภาพด้วย 2D-DCT จึงทำให้ความนิยมใช้เทคนิค DWT เข้ามาบีบอัดข้อมูลภาพเคลื่อนไหวมีน้อยกว่าเทคนิค 2D-DCT

3. เปรียบเทียบประสิทธิภาพการพัฒนาบนอุปกรณ์ฮาร์ดแวร์

การเปรียบเทียบประสิทธิภาพการพัฒนาบนอุปกรณ์ฮาร์ดแวร์ระหว่างเทคนิคการบีบอัดรูปภาพด้วย DWT กับ 2D-DCT สิ่งที่เป็นปัจจัยที่จะต้องคำนึงนำมาใช้ในการเปรียบเทียบ คือ ความเร็วในการประมวลผล และทรัพยากรที่สูญเสียในการออกแบบวงจร ซึ่งทั้ง 2 ปัจจัยสามารถสรุปได้ดังตารางที่ 1-3 ซึ่งเป็นเปรียบเทียบผลการออกแบบวงจรประมวลผล DWT และ 2D-DCT บน FPGA (Field Programmable Gate Array) รุ่น Xilinx Virtex-E และ Altera Apex20KE

ตารางที่ 1-3 เปรียบเทียบประสิทธิภาพด้านฮาร์แวร์ระหว่าง DWT และ 2D-DCT โดย Khurram

FPGA	Number of CLB/LE		Number of ESB/RAM		Processing rate (MSa/sec)	
	2D-DCT	DWT	2D-DCT	DWT	2D-DCT	DWT
Xilinx Virtex-E	1279	3784	1	24	80	55
Altera Apex20KE	2834	7381	1	24	80	55

จากตารางที่ 1-3 สังเกตได้ว่าความเร็วในการประมวลผลของ 2D-DCT จะมีความเร็วที่สูงกว่า DWT และจำนวนทรัพยากรที่ใช้เทคนิค DWT ใช้มากกว่า 2 เท่าของเทคนิค 2D-DCT จึงสามารถกล่าวได้ว่าประสิทธิภาพด้านฮาร์ดแวร์ของเทคนิค 2D-DCT ได้ผลประสิทธิภาพที่ดีกว่าเทคนิค DWT

จากผลการทดสอบและนำมาเปรียบเทียบประสิทธิภาพกันระหว่างเทคนิค DWT ใช้ในการบีบอัดรูปภาพแบบ JPEG2000 และเทคนิค 2D-DCT ซึ่งใช้ในการบีบอัดรูปภาพแบบ JPEG สามารถสรุปได้ว่า การบีบอัดข้อมูลภาพนิ่งของ DWT จะได้คุณภาพรายละเอียดที่ดีกว่า เมื่อบีบอัดในอัตราที่สูงทำให้ภาพไม่แตกเป็นบล๊อคเล็กๆ เหมือนกับแบบ 2D-DCT ส่วนการบีบอัดข้อมูลชนิดภาพเคลื่อนไหวจะได้คุณภาพที่ใกล้เคียงกันมากอันเนื่องมาจาก DWT จะมีปัญหาเกี่ยวกับความคมชัดของเส้นขอบภาพ และการเปลี่ยนเฟรมของภาพเคลื่อนไหวให้แสดงผลความแตกต่างของแต่ละเฟรมให้ทั่วทั้งภาพ และเมื่อนำมาทั้ง 2 แบบมาทดสอบในระดับฮาร์ดแวร์ปรากฏว่าเทคนิค DWT มีความซับซ้อนของวงจรมากกว่าทำให้ความเร็วในการบีบอัดช้ากว่า อีกทั้งยังใช้ทรัพยากรในการออกแบบวงจรมากกว่า 2 เท่าของการออกแบบวงจร 2D-DCT

ด้วยเทคนิค DWT สามารถพัฒนาอัลกอริทึมที่ใช้ในการบีบอัดรูปภาพที่มีชื่อว่า JPEG2000 ซึ่งมีประสิทธิภาพการบีบอัดสูง สามารถบีบอัดไฟล์ให้ได้ขนาดเล็กลงมากและยังคงรายละเอียดของภาพไม่แตกเป็นบล๊อคเล็กๆ เหมือนกับการบีบอัดรูปภาพแบบ JPEG แต่ด้วยปัญหาสิทธิบัตรของผู้ผลิต จึงทำให้ความนิยมที่จะนำมาใช้ยังมีน้อยมาก อาทิเช่น ในระบบอินเตอร์เน็ตข้อมูลประเภทรูปภาพเป็นข้อมูลที่ใช้กันอย่างแพร่หลาย และชนิดรูปภาพที่นิยมใช้มากที่สุด คือรูปภาพแบบ JPEG ซึ่งเป็นที่นิยมใช้มากกว่ารูปภาพแบบ JPEG2000 อีกทั้งอุปกรณ์แทบทุกชนิดที่ใช้ข้อมูลประเภทรูปภาพจะต้องมีการรองรับการบีบอัดรูปภาพแบบ JPEG ดังนั้นผู้จัดทำจึงคิดที่จะเพิ่มประสิทธิภาพด้านความเร็วการบีบอัดรูปภาพโดยเลือกที่จะเพิ่มความเร็วของการบีบอัดรูปภาพแบบ JPEG เพื่อเพิ่มขีดความสามารถให้กับงานบีบอัดรูปภาพที่ต้องการความเร็วสูง

Luciano [9] ได้ออกแบบการบีบอัดรูปภาพแบบ JPEG ใช้งานบน FPGA และกล่าวไว้ว่าขั้นตอนของการบีบอัดรูปภาพแบบ JPEG สูญเสียเวลาในการทำงานมากที่สุดอยู่ที่โมดูลการประมวลผลสมการ 2D-DCT ซึ่งเป็นขั้นตอนที่มีความยุ่งยากและซับซ้อนมากที่สุด นอกจากนั้นจำนวนของลอจิกของโมดูล 2D-DCT ที่ใช้มีมากกว่าครึ่งของวงจรการบีบอัดทั้งหมด สังเกตได้จากตารางที่ 1-4

ตารางที่ 1-4 แสดงผลการสังเคราะห์วงจรบีบอัดรูปภาพแบบ JPEG ที่ออกแบบโดย Luciano

Hardware Block	**Logic Cells**	**Frequency (MHz)**	**Clock Cycle (Cycles)**	**Latency (1/F x clock cycle)**
Color Space Converter	373	39.1	5	0.128 uS
2D-DCT	**3,962**	**36.2**	**163**	**4.503 uS**
Quantization	306	56.2	4	0.071 uS
Zigzag Buffer	86	128.2	66	0.515 uS
Entropy Coder	657	47.2	5	0.106 uS

จากตารางที่ 1-4 พบว่าขั้นตอนของการบีบอัดภาพในส่วนของการประมวลผลสมการ 2D-DCT เป็นขั้นตอนที่ใช้เวลาในการประมวลผลมากที่สุด ซึ่งหากต้องการเพิ่มความเร็วให้กับการบีบอัดรูปภาพแบบ JPEG จึงควรที่จะพัฒนาในส่วนของ 2D-DCT เพื่อเพิ่มประสิทธิภาพด้านความเร็วของการประมวลผลให้รองรับกับงานแบบ Real-time

งานวิจัยนี้ ได้เน้นการเพิ่มความเร็วให้กับการบีบอัดรูปภาพ JPEG ในส่วนของการพัฒนาและออกแบบวงจร 2D-DCT ซึ่งเป็นขั้นตอนที่มีความซับซ้อนและสิ้นเปลืองเวลาการประมวลมากที่สุดของขั้นตอนการบีบอัดรูปภาพ JPEG และเน้นการประมวลผลแบบขนาน (Parallel Processing) เพื่อออกแบบและพัฒนาวงจร 2D-DCT บนสถาปัตยกรรม FPGA ซึ่งรองรับโครงสร้างการทำงานแบบขนานได้เป็นอย่างดี ทั้งนี้เพื่อเสริมให้อุปกรณ์สมองกลฝังตัวที่มีข้อด้อยในเรื่องของความเร็วการประมวลผลของโปรเซสเซอร์ ให้สามารถทำงานมีประสิทธิภาพเทียบเท่ากับคอมพิวเตอร์ความเร็วสูง

1.2 วัตถุประสงค์ของวิทยานิพนธ์

เพื่อเพิ่มความเร็วการประมวลผลของวงจร 2D-DCT สำหรับใช้ในการบีบอัดรูปภาพด้วยอัลกอริทึม JPEG ด้วยวิธีการประมวลผลแบบขนาน

1.3 ขอบเขตของการทำวิทยานิพนธ์

1. ศึกษาและออกแบบวงจรการคำนวณสมการ 2D-DCT เพื่อเพิ่มความเร็วในการบีบอัดรูปภาพขนาด 640x480 และ 1920x1080 ให้มีความเร็วในการไหลของข้อมูลที่ไม่ต่ำกว่า 25 เฟรมต่อวินาที ซึ่งเป็นขนาดภาพและความเร็วที่ใช้ในกล้อง IP camera ในปัจจุบัน

2. ออกแบบวงจรบรรยายด้วยภาษา VHDL ซึ่งเป็นภาษาโปรแกรมระดับสูงสามารถใช้ออกแบบและพัฒนาวงจรดิจิตอลบน FPGA

3. ทดสอบและจำลองผลการทำงานบนเทคโนโลยี Xilinx FPGA ตระกูล Virtex 4 ชิปเบอร์ XC4VFX12

1.4 ขั้นตอนและระยะเวลาการดำเนินงาน

การดำเนินงานทำวิทยานิพนธ์ชุดนี้มีขั้นตอนการทำงานดังต่อไปนี้

1. ศึกษาค้นคว้าข้อมูลเอกสารที่เกี่ยวข้องโดยแบ่งออกเป็นหมวดหมู่ดังต่อไปนี้
   - ขั้นตอนการบีบอัดรูปภาพแบบ JPEG
   - เทคนิคการประยุกต์วงจร 2D-DCT ให้ง่ายต่อการออกแบบวงจรดิจิตอล
   - เทคนิคการเพิ่มความเร็วด้วยการประมวลผลแบบขนาน
   - งานวิจัยอื่นๆ ที่เกี่ยวข้อง
2. วิเคราะห์และออกแบบวงจร 2D-DCT
   - ศึกษาและวิเคราะห์สมการ 2D-DCT
   - ออกแบบวงจร 2D-DCT โดยใช้วิธีการประมวลผลแบบขนาน
   - ใช้ภาษา VHDL ในการออกแบบและบรรยายบนโปรแกรม Xilinx 8.1
   - ทดสอบวงจรด้วย Test Bench Waveform
3. ทดสอบวงจรและแก้ไขระบบการประมวลผล
   - วิเคราะห์หาข้อผิดพลาดและหาวิธีแก้ไขปรับปรุง
   - แก้ไขวงจรให้เหมาะสมกับทรัพยากรที่ใช้
   - ปรับปรุงวงจรให้ถูกต้องตามสมการ 2D-DCT
   - ทดสอบวงจรหลังจากทำการแก้ไขแล้ว
4. จัดทำเอกสารรายงานผลการทำวิทยานิพนธ์

ตารางที่ 1-5 แสดงระยะเวลาการดำเนินงานในแต่ละขั้นตอน

กรกฎาคม 2551 - มีนาคม 2553

ขั้นตอนที่	2551						2552												2553		
	7	8	9	10	11	12	1	2	3	4	5	6	7	8	9	10	11	12	1	2	3
1																					
2																					
3																					
4																					

1.5 สถานที่ในการทำวิทยานิพนธ์

ห้องปฏิบัติการ Embedded System ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยสงขลานครินทร์ วิทยาเขตหาดใหญ่

1.6 เครื่องมือที่ใช้ในการทำวิทยานิพนธ์

ด้านฮาร์ดแวร์

- เครื่องคอมพิวเตอร์ ความเร็ว 3.2 กิกะเฮิร์ต หน่วยความจำ 2 กิกะไบท์ ฮาร์ดดิส ความจุ 80 กิกะไบร์ จำนวน 1 ชุด สำหรับใช้พัฒนาและทดสอบวงจรด้วยโปรแกรมจำลองการทำงาน

ด้านซอฟต์แวร์

- ระบบปฏิบัติการ Microsoft Windows XP
- โปรแกรม Xilinx ISE 8.1 ใช้สำหรับพัฒนาออกแบบวงจร 2D-DCT
- โปรแกรม Matlab 7.8 ใช้สำหรับจำลองขั้นตอนการทำงานการบีบอัดรูปภาพแบบ JPEG

1.7 ประโยชน์ที่คาดว่าจะได้รับ

1. วงจรคำนวณ 2D-DCT ที่สามารถประมวลผลได้เร็ว
2. ระบบบีบอัดภาพแบบ JPEG ที่ใช้ 2D-DCT ที่นำเสนอ สามารถรองรับการไหลผ่านของข้อมูลที่ดีกว่าหรือเทียบเท่ากับภาพเคลื่อนไหวที่ทันกับเวลาเหตุการณ์จริงไม่ต่ำกว่า 25 เฟรมต่อวินาที

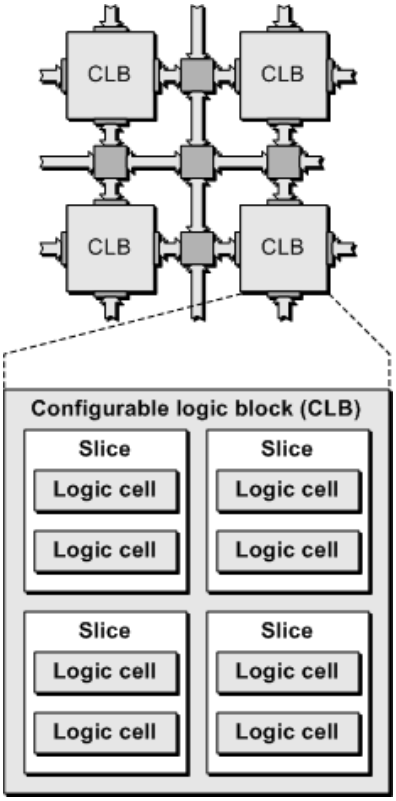
บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

ในบทนี้กล่าวถึงทฤษฎีและงานวิจัยที่เกี่ยวข้อง ประกอบด้วยสถาปัตยกรรม FPGA ภาษา VHDL ทฤษฎีการบีบอัดรูปภาพแบบ JPEG และกล่าวถึงวิธีการออกแบบวงจรดิจิตอลของสมการ 2D-DCT ที่นำเสนอมาในอดีตและจะนำแนวทางมาพัฒนาเพื่อเพิ่มความเร็วให้กับวงจรการประมวลผล ซึ่งเป็นองค์ความรู้ที่สำคัญในการดำเนินการทำวิทยานิพนธ์

2.1 สถาปัตยกรรม FPGA [3]

FPGA (Field Programmable Gate Array) คือ อุปกรณ์อิเล็กทรอนิคที่จัดอยู่ในรูปแบบของเซมิคอนดักเตอร์(Semi-Conductor) ชนิดหนึ่ง สถาปัตยกรรมของ FPGA ประกอบไปด้วยหน่วยย่อยการทำงานที่เรียกว่า ลอจิกบล็อค (Logic Block) ลอจิกบล็อคแต่ละตัวจะประกอบไปด้วยสไลซ์ (Slice) ที่มี ลอจิกเซลล์ (Logic Cells) ใช้สำหรับเชื่อมต่อกับลอจิกเซล์ตัวอื่นเพื่อออกแบบเป็นวงจรดิจิตอล โดยการเชื่อมต่อของลอจิกเซลล์จะทำการปิดและเปิดสวิทช์ทรานซิสเตอร์ (Transistor Switch) เพื่อเลือกใช้ลอจิกตามที่ต้องการ ซึ่งจำนวนของเกตทรานซิสเตอร์ที่มีให้ใช้อยู่ในปัจจุบันของ FPGA มีตั้งแต่หลักไม่กี่ร้อยตัวจนไปถึงหลักหลายล้านตัว ขึ้นอยู่กับชนิดและรุ่นของ FPGA

การปิด-เปิด ของการสวิทช์เพื่อเชื่อมต่อระหว่างลอจิกเซลล์นั้นกระทำโดยใช้โปรแกรมที่ผู้ผลิตได้สร้างขึ้นเพื่อออกแบบวงจรและทำการสร้างเน็ตลิสส์ (Netlist) เพื่อบรรจุข้อมูลการสวิทช์ข้อมูลลงไปในอุปกรณ์ FPGA โดยในการออกแบบสามารถทำได้ด้วยการวาดวงจร (schematic entry) หรือการบรรยายด้วยใช้ภาษาระดับสูง HDL (Hardware Description Language) ที่นิยมใช้ออกแบบอยู่ทั่วไปในปัจจุบัน คือ Verilog และ VHDL (VHSIC Hardware Description Language) ลอจิกบล็อคที่ใช้ในการออกแบบวงจรดิจิตอลมีลักษณะดังภาพประกอบ 2-1



ภาพประกอบ 2-1 แสดงลักษณะของลอจิกบล็อค [5]

ด้วยเทคโนโลยีการสวิทซ์เลือกลอจิกของทรานซิสเตอร์ประกอบกับมีเครื่องมืออุปกรณ์ช่วยให้สะดวกในการออกแบบและโปรแกรม การออกแบบวงจรดิจิตอลบน FPGA จึงสามารถที่จะออกแบบวงจรที่มีขนาดใหญ่ เหมาะสำหรับการสร้างวงจรแบบขนานและมีความซับซ้อนของวงจรให้อยู่ในชิปแค่ตัวเดียวได้ง่าย อีกทั้งยังสามารถเปลี่ยนแปลงหรือแก้ไขวงจรการออกแบบได้ตลอดเวลาซึ่งเป็นที่มาของคำว่า “ Programmable ”

2.2 ภาษา VHDL [1]

VHDL ย่อมาจาก VHSIC Hardware Description Language (VHSIC : Very High Speed Integrated Circuit) เป็นภาษาระดับสูงใช้สำหรับออกแบบวงจรฮาร์ดแวร์ในระบบดิจิตอล ซึ่งถูกพัฒนาโดยกระทรวงกลาโหมของสหรัฐอเมริกา (Department of Defense : DoD) ในช่วงปี ค.ศ. 1980 มีเป้าหมายของโครงการเพื่อพัฒนาขีดความสามารถในการออกแบบวงจรรวมให้สูงขึ้นและสามารถทำได้ง่ายโดยมีเป้าหมายหลัก 2 อย่างคือ

1. ต้องการภาษาที่สามารถรองรับการออกแบบวงจรที่มีความซับซ้อน
2. ต้องการภาษาที่เป็นมาตรฐานหรือเป็นเป็นภาษากลางที่ทำให้สามารถเผยแพร่ผลงานการออกแบบกันภายในกลุ่มนักออกแบบวงจรดิจิตอล

จนกระทั่งปี ค.ศ. 1986 ภาษา VHDL ได้ถูกปรับปรุงและสามารถกำหนดเป็นมาตรฐานของ IEEE (The Institute of Electronics and Electrical Engineers) โดยประกาศเป็นมาตรฐานในเดือนธันวาคมปี ค.ศ. 1987 อยู่ในหมวด IEEE 1076-1987 ภาษา VHDL มีขีดความสามารถในการออกแบบวงจรในลักษณะระบบดิจิตอลเท่านั้น ซึ่งในปัจจุบันได้มีการวิจัยและกำลังพัฒนาให้สามารถออกแบบวงจรอนาล็อค (Analog Circuit) โดยใช้ชื่อว่า " VHDL-AMS (VHDL-Analog Mixed Signal) "

ด้วยประสิทธิภาพของภาษา VHDL สามารถสรุปประโยชน์ของภาษาได้ 6 ประการดังต่อไปนี้

1. เป็นภาษามาตรฐานสากลรองรับโดยสถาบัน IEEE และมีอุปกรณ์เครื่องมือต่างๆ ที่รองรับภาษา VHDL มากมาย อีกทั้งมีบริษัทจำนวนมากที่ให้การสนับสนุนและร่วมพัฒนา

2. ได้รับการสนับสนุนในการออกแบบและพัฒนาจากรัฐบาลกระทรวงกลาโหมของสหรัฐอเมริกา

3. เป็นภาษาที่ใช้จริงในอุตสาหกรรมใช้ในการออกแบบวงจรดิจิตอล

4. เป็นภาษาที่ใช้ได้หลายระบบมีซอฟต์แวร์ (Software) หลายอย่างรองรับไม่ยึดติดกับซอฟแวร์แค่ตัวเดียวทำให้สะดวกและเพิ่มความหลายหลายในการออกแบบใช้งาน

5. สามารถง่ายต่อการนำกลับมาแก้ไขวงจรหรือนำกลับมาใช้ใหม่คล้ายคลึงกับการโปรแกรมซอฟแวร์ทั่วๆ ไป

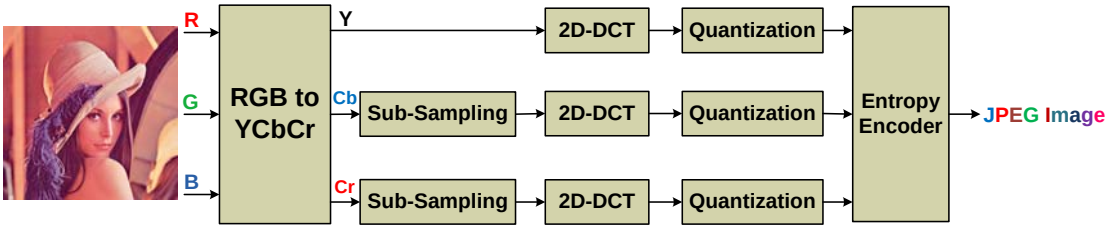
6. เป็นภาษาที่สามารถใช้เป็นเอกสารอ้างอิงประกอบได้ สามารถอธิบายพฤติกรรมของวงจรดิจิตอลในการออกแบบได้ทันที

2.3 การบีบอัดรูปภาพแบบ JPEG

JPEG (Joint Photographic Experts Group)[13] คือ อัลกอริทึมของการบีบอัดรูปภาพที่ได้รับการยอมรับและนิยมใช้กันมากที่สุดในปัจจุบัน เนื่องจากอัลกอริทึมนี้สามารถที่จะเก็บข้อมูลความละเอียดสูงได้โดยใช้พื้นที่ในการเก็บน้อยมาก สามารถเก็บข้อมูลได้ทั้งภาพสีและขาวดำและได้ลักษณะสีของรูปภาพที่ชัดเจนแม่นยำ การบีบอัดรูปภาพแบบ JPEG จะใช้วิธีการบีบอัดรูปภาพแบบคงข้อมูลหลัก (Lossy Compression) คือ การยอมสูญเสียข้อมูลบางส่วนทำให้ขนาดของข้อมูลลดลงมาก แต่คุณภาพของรูปภาพหากวัดด้วยการมองของสายตาของมนุษย์ จะพบว่าคุณภาพบางส่วนหายไปแต่ยังคงความใกล้เคียงกับต้นฉบับมาก

การบีบอัดภาพแบบ JPEG ออกแบบโดยใช้เทคนิคที่เรียกว่า DCT (Discrete Cosine Transform) เป็นการแปลงค่าความสว่างของรูปภาพให้อยู่ในรูปแบบเชิงความถี่(Frequency Domain)

ทำให้สามารถเลือกแทนค่าของสัมประสิทธิ์หรือในที่นี้คือแอมพลิจูดของค่าความถี่ต่างๆ ได้โดยอาศัยตัวแปรที่มีนัยสำคัญที่ต่างกันได้ การที่สามารถลดนัยสำคัญของค่าตัวเลขลงไปได้ทำให้สามารถลดขนาดของหน่วยความจำที่ใช้เก็บข้อมูลรูปภาพได้ มีลำดับขั้นตอนการประมวลผลภาพประกอบ 2-2



ภาพประกอบ 2-2 ขั้นตอนการบีบอัดรูปภาพแบบ JPEG [11]

2.3.1 Color Space Conversion

เป็นขั้นตอนการเปลี่ยนรูปแบบภาพ RGB ให้อยู่ในรูปแบบของ YCbCr โดยเป็นการเปลี่ยนตามลักษณะกระบวนการรับรู้ของประสาทตาของมนุษย์ใช้สมการ [11] ดังต่อไปนี้

$$Y = 0.299R + 0.587G + 0.114B$$

$$Cb = 128 - 0.168736R + 0.331264G + 0.5B$$

$$Cr = 128 + 0.5R - 0.418688G - 0.081312B$$

กำหนดให้ Y = ค่าความสว่างของแสงในระบบดิจิตอล

Cb = ค่าความเข้มของสีน้ำเงินในระบบดิจิตอล

Cr = ค่าความเข้มของสีแดงในระบบดิจิตอล

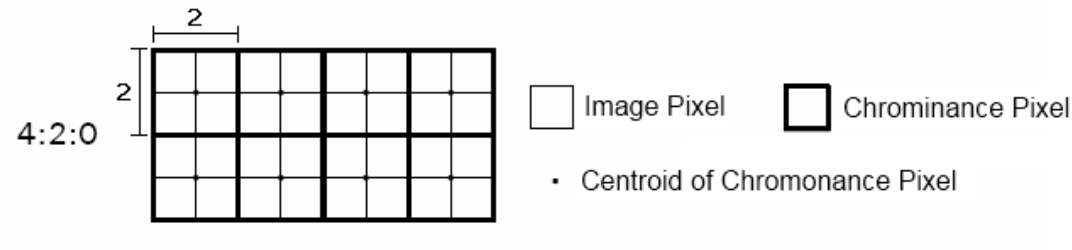
R = ค่าความเข้มของสีแดงในระบบอนาล็อก

G = ค่าความเข้มของสีเขียวในระบบอนาล็อก

B = ค่าความเข้มของสีน้ำเงินในระบบอนาล็อก

2.3.2 Sub-sampling [6]

ในระบบภาพสีจะมีองค์ประกอบของความเข้มแสง(Luminance) และองค์ประกอบของความเข้มสี (Chrominance) โดยปกติสายตามนุษย์มีสัมผัสในการแยกองค์ประกอบของความเข้มสีได้ไม่ดี จึงสามารถที่จะลดข้อมูลในส่วนของความเข้มสีได้ แต่สายตามนุษย์สามารถแยกแยะความเข้มของแสงได้ดีจึงไม่สามารถลดข้อมูลในส่วนของความเข้มแสงได้ ดังนั้นจึงสามารถที่จะลดข้อมูลในส่วนของ C_B, C_R ได้ดังภาพประกอบ 2-2



ภาพประกอบ 2-3 แสดงสักษณะการทำ Sub-Sampling

จากภาพประกอบ 2-3 การที่จะทำ Sub-Sampling นั้นจะทำการจัดการในส่วนของโครมิแนนท์ดังที่กล่าวไว้ข้างต้นโดยการพิจารณาค่าโครมิแนนท์ขนาด 2x2 พิกเซลแล้วหาค่าจุดกึ่งกลางของค่าโครมิแนนท์ทั้ง 4 (Centroid of Chrominance Pixel) จากนั้นก็แทนค่าของจุดกึ่งกลางนั้นเป็นข้อมูลกลับไปยัง 4 พิกเซลที่ทำการหาค่าจุดกึ่งกลาง

2.3.3 Two-Dimention Discrete Cosine Transform (2D-DCT) [11]

เป็นการแปลงข้อมูลจากโดเมนเวลา เป็นโดเมนความถี่โดยนำข้อมูลที่เข้ามาแบ่งย่อยๆเป็นส่วนเล็กๆ เป็นเลขยกกำลังของเลข 2 เช่น 4,8,16 เป็นต้น ยิ่งบล็อคมีขนาดใหญ่จะยิ่งบีบอัดข้อมูลได้มากขึ้นแต่คุณภาพของรูปภาพจะลดลงมาก ดังนั้นการที่จะเลือกว่าจะใช้ขนาดเท่าไรจะต้องคำนึงถึงคุณภาพของภาพที่บีบอัดและความเหมาะสมของข้อมูลหลังจากการบีบอัดข้อมูลโดยมาตรฐานของระบบการบีบอัดรูปภาพแบบ JPEG จะเลือกขนาดที่ 8x8 พิกเซล เพราะสายตามนุษย์สามารถสังเกตความแตกต่างได้ไม่มากจนเกินไปและขนาดข้อมูลก็สามารถบีบอัดได้อย่างเหมาะสม

สมการ 2D-DCT [8] แสดงได้ดังนี้

$$F(u,v) = \propto (u) \propto (v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cos\left[\frac{(2x+1)u\pi}{2N}\right] \cos\left[\frac{(2y+1)v\pi}{2N}\right]$$

กำหนดให้ X = พิกัดในแนวนอนของเมตริกข้อมูลที่ใช้สำหรับคำนวณ

Y = พิกัดในแนวตั้งของเมตริกข้อมูลที่ใช้สำหรับคำนวณ

u = พิกัดในแนวนอนของเมตริกผลลัพธ์

v = พิกัดในแนวตั้งของเมตริกผลลัพธ์

$f(x.y)$ = เมตริกข้อมูลที่ใช้สำหรับคำนวณ

$F(u,v)$ = เมตริกผลลัพธ์

N = ขนาดความกว้างและยาวของเมตริกข้อมูลที่ใช้สำหรับคำนวณ

$$u, v = 0,1,2,...., N-1$$

$$\propto (u) = \begin{cases} \sqrt{\frac{1}{N}} & \text{เมื่อ u = 0} \\ \sqrt{\frac{2}{N}} & \text{เมื่อ u = 1, 2, 3, …. , N-1} \end{cases}$$

2.3.4 Quantization [11]

เป็นขั้นตอนที่ยินยอมให้มีการสูญเสียข้อมูลได้ในกระบวนการบีบอัดซึ่งกระบวนการนี้จะทำการแปลงข้อมูลที่ได้จากการทำ 2D-DCT ที่เป็นทศนิยมให้เป็นจำนวนเต็มและเป็นการทำให้มีค่าที่ซ้ำกันเพิ่มมากขึ้นด้วย สายตามนุษย์จะมีความรู้สึกไวต่อการรับรู้ค่าสัมประสิทธิ์ความถี่ต่ำของรูปภาพ จากการทำ 2D-DCT จะทำการนอร์มอไลซ์ด้วยค่าที่ต่ำบริเวณสัมประสิทธิ์ที่มีความถี่ต่ำ และนอร์มอไลซ์ด้วยค่าที่สูงบริเวณสัมประสิทธิ์ที่มีความถี่สูง ซึ่งจะทำให้เกิดค่าที่เป็นศูนย์เพิ่มมากขึ้น การควอนไตซ์มีรูปแบบดังต่อไปนี้

$$QuantizationValue(i, j) = \frac{DCT(i, j)}{QuantizationMatrix(i, j)}$$

โดยการทำควอนไตซ์จะแบ่งออกเป็น 2 ส่วนคือ

1. Luminance ในส่วนนี้จะเป็นส่วนของความเข้มแสงซึ่งประสาทรับรู้ของคนจะรับรู้ได้ไวซึ่งสามารถที่จะปรับคุณภาพของรูปภาพได้ (Quality factor) โดยการปรับเปลี่ยนค่าลูมิแนนท์ควอนไตซ์ ถ้าต้องการคุณภาพสูงก็ปรับให้ค่าคุณภาพให้มากกว่า 50 แต่หากต้องการคุณภาพของรูปภาพที่ต่ำให้ปรับต่ำกว่า 50 ดังสมการ [11] ต่อไปนี้

เมตริกควอนไตซ์ของลูมิแนนซ์

$$Q_{luminance} = \begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix}$$

สมการการคำนวณหาค่าเมตริกควอนไตซ์ของลูมิแนนซ์ใช้เป็นตัวจัดการคุณภาพของรูปภาพ ซึ่งเป็นการนำเมตริกควอนไตซ์ของลูมิแนนซ์ที่กล่าวไว้ข้างต้นมีค่าคุณภาพกำหนดไว้เท่ากับ 50 หรืออาจจะเขียนว่า Q_{50} โดยมีสมการ [11] การคำนวณดังต่อไปนี้

$$Qx = \begin{cases} \frac{100-x}{50} \times Q_{50} & \text{เมื่อ} \quad Qx > 50, Qx < 100 \\ \frac{50}{x} \times Q_{50} & \text{เมื่อ} \quad Qx > 0, Qx < 50 \end{cases}$$

Qx คือ ลูมิแนนท์ควอนไตซ์แฟคแตอร์

x คือ ควอนไตซ์แฟคเตอร์

Q_{50} คือ เมตริกควอนไตซ์ของลูมิแนนซ์

2. Chrominance ในส่วนนี้จะทำการจัดการค่าของ C_B และ C_R โดยใช้สมการ [11] ดังต่อไปนี้

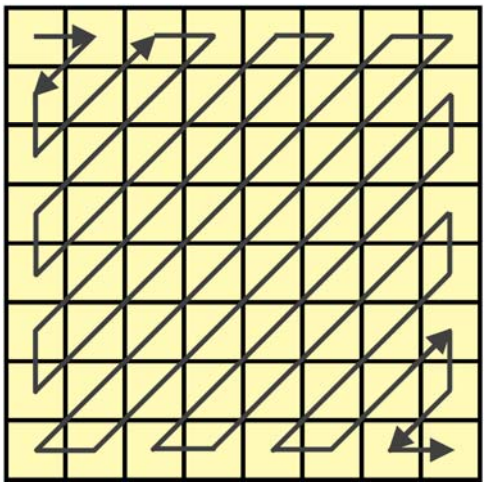
$$Q_{Chrominance} = \begin{pmatrix} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{pmatrix}$$

ในส่วนของการทำควอนไตซ์ของ Chrominance มีเมตริกที่ทำการควอนไตซ์เป็นมาตรฐานซึ่งเป็นเมตริกที่แปลงข้อมูลให้มีค่าเป็น 0 เพิ่มมากขึ้นอีก มีค่ามาตรฐานแค่เมตริกเดียว

เมื่อกำหนดค่าเมตริกค่าคงที่ได้ตามที่ได้กล่าวไว้ข้างต้นแล้วจึงนำไปประมวลผลดังสมการ Quantization Value ในลำดับต่อไป

2.3.5 Entropy Encoder [11]

เป็นวิธีการจัดการใช้การเรียงข้อมูลโดยใช้วิธีเรียงข้อมูลแบบ Zig-Zag ดังภาพประกอบ 2-4 เพื่อจัดการข้อมูลที่ผ่านการควอนไตซ์มาเรียงเป็นแถวแล้วจะทำให้ค่าที่เป็นศูนย์อยู่ติดกันเพื่อง่ายต่อการเข้ารหัสในลำดับต่อไป



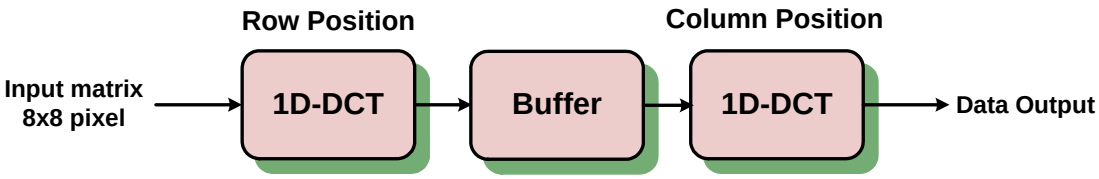
ภาพประกอบ 2-4 แสดงการเรียงข้อมูลแบบ Zig-Zag

หลักการเข้ารหัสจะใช้วิธีการเข้ารหัสฮัฟแมนและการเข้ารหัสแบบรันเรนส์เข้ามาช่วยในการเข้ารหัส โดยการเข้ารหัสจะจัดการในส่วนค่าที่ไม่เป็นศูนย์ตามมาตรฐานฮัพแมน ซึ่งจากการที่ทำควอนไตซ์และจัดเรียงข้อมูลแบบซิกแซกทำให้เกิดค่าที่มีเลขศูนย์ติดๆกันจะช่วยให้สามารถลดข้อมูลได้มากเมื่อผ่านการเข้ารหัส

2.4 การออกแบบวงจรการคำนวณสมการ 2D-DCT

2D-DCT (Discrete Cosine Transform) เป็นขั้นตอนหนึ่งของการบีบอัดรูปภาพด้วยอัลกอริทึม JPEG เป็นขั้นตอนที่มีความซับซ้อนมากที่สุด สังเกตได้จากสมการที่กล่าวไว้ในหัวข้อที่ 2.3.3 การคำนวณสมการ 2D-DCT ซึ่งเป็นสมการที่ทำการคำนวณข้อมูลในลักษณะ 2 มิติ โดยตามมาตรฐานระบบการบีบอัดรูปภาพแบบ JPEG จะแบ่งข้อมูลออกเป็นส่วนย่อยๆ มีลักษณะเป็นเมตริกขนาด 8x8 พิกเซล และจัดการข้อมูลในแนวนอนและในแนวตั้ง การคิดคำนวณจะคำนวณข้อมูลในเมตริกขนาด 8x8 ที่แบ่งย่อยๆ แสดงเป็นผลลัพธ์ของสมการเป็นเมตริกขนาด 8x8 เช่นเดียวกับเมตริกที่ป้อนเข้ามา

Latha [8] ได้ออกแบบวงจร 2D-DCT ที่สามารถใช้กับการบีบอัดภาพเคลื่อนไหว (VDO Compression) ซึ่งได้นำเสนอแนวคิดที่น่าสนใจนั่นคือ การใช้วงจรการประมวลผล 1D-DCT เข้ามาช่วยในการออกแบบวงจร 2D-DCT ซึ่งเรียกการออกแบบนี้ว่าการประมวลผลแบบเวคเตอร์ (Vector Processing) โดยมีแผนผังโครงสร้างการออกแบบดังภาพประกอบ 2-5



ภาพประกอบ 2-5 แสดงการออกแบบวงจร 2D-DCT โดยวิธีการประมวลผลแบบเวคเตอร์

จากภาพประกอบ 2-5 แสดงการใช้วงจร 1D-DCT ออกแบบเป็นวงจร 2D-DCT ประกอบไปด้วยวงจร 1D-DCT จำนวน 2 วงจร โดยวงจรในส่วนแรกจะทำการรับข้อมูลขนาด 8x8 พิกเซล เข้ามาคำนวณโดยส่งข้อมูลเข้าไปคำนวณในลักษณะมองข้อมูลในแนวนอน จากนั้นส่งไปเก็บใน บับเฟอร์ (Buffer) เพื่อรอให้ประมวลผลจนเสร็จ 1 แถวในแนวตั้ง จากนั้นจึงส่งต่อไปยังวงจร 1D-DCT ส่วนที่ 2 ซึ่งในส่วนนี้จะคำนวณโดยมองข้อมูลในแนวตั้ง

จากสมการการคำนวณ 2D-DCT ในหัวข้อที่ 2.3.3 สามารถที่จะเขียนแยกเป็นสมการ 1D-DCT ได้ 2 สมการ [8] ดังต่อไปนี้

$$C = K \bullet \cos\frac{(2 \cdot col + 1) \bullet row \bullet \pi}{2 \bullet M} \qquad \text{สมการที่ (1)}$$

$$K = \sqrt{\frac{1}{N}} \quad \text{เมื่อ} \quad \text{row} = 0$$

$$K = \sqrt{\frac{2}{N}} \quad \text{เมื่อ} \quad \text{row} \neq 0$$

$$C^T = K \bullet \cos\frac{(2 \cdot row + 1) \bullet col \bullet \pi}{2 \bullet N} \qquad \text{สมการที่ (2)}$$

$$K = \sqrt{\frac{1}{M}} \quad \text{เมื่อ} \quad \text{col} = 0$$

$$K = \sqrt{\frac{2}{M}} \quad \text{เมื่อ} \quad \text{col} \neq 0$$

กำหนดให้ M คือ จำนวนสมาชิกในแนวตั้ง

N คือ จำนวนสมาชิกในแนวนอน

row คือ ค่าประจำตำแหน่งสมาชิกของเมตริกในแนวนอน

col คือ ค่าประจำตำแหน่งสมาชิกของเมตริกในแนวตั้ง

C คือ เมตริกค่าคงที่ใช้สำหรับคำนวณข้อมูลในแนวตั้ง

C^T คือ เมตริกค่าคงที่ใช้สำหรับคำนวณข้อมูลในแนวตั้ง

จากสมการที่ 1 คือสมการ C เป็นสมการ 1D-DCT ซึ่งจะมีการแทนค่าตำแหน่งของเมตริกและจำนวนของสมาชิกในเมตริกนั้นเข้าไปเพื่อหาคำตอบของตำแหน่งนั้นส่วนสมการที่ 2 คือ C^T เป็นสมการเช่นเดียวกับสมการที่ 1 แต่ต่างกันที่เป็นการสลับการแทนค่าตำแหน่งในสมการ

ตามลักษณะรูปแบบของเมตริกค่าสมาชิกในตารางเมตริกในแต่ละตัว จะมีค่าประจำตำแหน่งของแต่ละตัว ซึ่งข้อมูลที่ใช้ในการคำนวณจะถูกแบ่งออกเป็นส่วนๆ ลักษณะแบบเมตริกมีขนาด 8x8 พิกเซล การกำหนดค่าตำแหน่งของสมาชิกในเมตริกตามรูปแบบของการบีบอัดรูปภาพแบบ JPEG กำหนดให้เริ่มจากซ้ายไปขวาในแนวนอน และจากบนลงมาข้างล่างในแนวตั้ง เช่น ตำแหน่งซ้ายบนสุดคือตำแหน่งเริ่มต้นของทั้งแนวตั้งและแนวนอนกำหนดให้เป็น (0,0) ถัดไปด้านซ้ายมือของตำแหน่ง (0,0) ก็จะค่าตำแหน่งเป็น (0,1) และที่ตำแหน่งขวาล่างสุดเป็นตำแหน่งสุดท้ายของทั้งแนวนอนและแนวตั้งจะกำหนดให้เป็น (7,7)

ดังนั้นจึงสรุปค่าประจำตำแหน่งของสมาชิกในตารางเมตริกตามรูปแบบของการบีบอัดรูปภาพแบบ JPEG ได้ดังภาพประกอบ 2-6

(0,0)	**(0,1)**	**(0,2)**	**(0,3)**	**(0,4)**	**(0,5)**	**(0,6)**	**(0,7)**
(1,0)	**(1,1)**	**(1,2)**	**(1,3)**	**(1,4)**	**(1,5)**	**(1,6)**	**(1,7)**
(2,0)	**(2,1)**	**(2,2)**	**(2,3)**	**(2,4)**	**(2,5)**	**(2,6)**	**(2,7)**
(3,0)	**(3,1)**	**(3,2)**	**(3,3)**	**(3,4)**	**(3,5)**	**(3,6)**	**(3,7)**
(4,0)	**(4,1)**	**(4,2)**	**(4,3)**	**(4,4)**	**(4,5)**	**(4,6)**	**(4,7)**
(5,0)	**(5,1)**	**(5,2)**	**(5,3)**	**(5,4)**	**(5,5)**	**(5,6)**	**(5,7)**
(6,0)	**(6,1)**	**(6,2)**	**(6,3)**	**(6,4)**	**(6,5)**	**(6,6)**	**(6,7)**
(7,0)	**(7,1)**	**(7,2)**	**(7,3)**	**(7,4)**	**(7,5)**	**(7,6)**	**(7,7)**

ภาพประกอบ 2-6 เมตริกแสดงค่าประจำตำแหน่งของสมาชิกภายในเมตริกขนาด 8x8

จากค่าประจำตำแหน่งของเมตริกดังภาพประกอบ 2-6 แทนค่าในแนวนอนและแนวตั้งอยู่ในรูปแบบของ (row, col) เมื่อ row คือ ค่าตำแหน่งในแนวนอน และ col คือ ค่าตำแหน่งในแนวตั้ง เมื่อแทนค่าประจำตำแหน่งเข้าไปในสมการ 1D-DCT ทั้ง 2 สมการ และแทนค่าจำนวนสมาชิกในแนวนอนและจำนวนสมาชิกในแนวตั้ง จากมาตรฐานของการบีบอัดรูปภาพแบบ JPEG ขนาดเมตริกข้อมูลจะมีขนาด 8x8 จึงแทนค่าจำนวนสมาชิกเท่ากับ 8 ทั้งแนวนอนและแนวตั้ง จะได้เป็นค่าคงที่ประจำตำแหน่งต่างๆ เป็นเลขทศนิยม ซึ่งสามารถสรุปเป็นเมตริก C และ C^T โดยแสดงเฉพาะส่วนของเลขทศนิยม ดังเมตริก [8] ต่อไปนี้

เมตริกค่าคงที่ของสมการ C

$$C = \begin{bmatrix} 23170 & 23170 & 23170 & 23170 & 23170 & 23170 & 23170 & 23170 \\ 32138 & 27246 & 18205 & 6393 & -6393 & -18205 & -27246 & -32138 \\ 30274 & 12540 & -12540 & -30274 & -30274 & -12540 & 12540 & 30274 \\ 27246 & -6393 & -32138 & -18205 & 18205 & 32138 & 6393 & -27246 \\ 23170 & -23170 & -23170 & 23170 & 23170 & -23170 & -23170 & 23170 \\ 18205 & -32138 & 6393 & 27246 & -27246 & -6393 & 32138 & -18205 \\ 12540 & -30274 & 30274 & -12540 & -12540 & 30274 & -30274 & 12540 \\ 6393 & -18205 & 27246 & -32138 & 32138 & -27246 & 18205 & -6393 \end{bmatrix}$$

เมตริกค่าคงที่ของสมการ C^T

$$C^T = \begin{bmatrix} 23170 & 32138 & 30274 & 27246 & 32170 & 18205 & 12540 & 6393 \\ 23170 & 27246 & 12540 & -6393 & -23170 & -32138 & -30274 & -18205 \\ 23170 & 18205 & -12540 & -32138 & -23170 & 6393 & 30274 & 27246 \\ 23170 & 6393 & -30274 & -18205 & 23170 & 27246 & -12540 & -32138 \\ 23170 & -6393 & -30274 & 18205 & 23170 & -27246 & -12540 & 32138 \\ 23170 & -18205 & -12540 & 32138 & -23170 & -6393 & 30274 & -27246 \\ 23170 & -27246 & 12540 & 6393 & -23170 & 32138 & -30274 & 18205 \\ 23170 & -32138 & 30274 & -27246 & 23170 & -18205 & 12540 & -6393 \end{bmatrix}$$

การออกแบบให้ประมวลผลแบบเวคเตอร์ซึ่งมีโครงสร้างดังภาพประกอบ 2.4 สามารถเขียนเป็นสมการการคำนวณ [8] ดังต่อไปนี้

สมการการคำนวณวงจร 2D-DCT

$$Y = C \bullet X \bullet C^T \qquad \text{สมการที่ (3)}$$

กำหนดให้ Y คือ เมตริกผลลัพธ์ของการคำนวณวงจร 2D-DCT

X คือ เมตริกข้อมูลที่ต้องการเข้ามาประมวลผล

C คือ เมตริกค่าคงที่สมการ 1D-DCT

C^T คือ เมตริกค่าคงที่สมการ 1D-DCT ที่สลับค่าระหว่างแนวตั้งกับแนวนอน

วงจร 2D-DCT จากที่กล่าวข้างต้น ประกอบไปด้วย 2 ขั้นตอน คือ

1. การประมวลผลสมการ 1D-DCT ป้อนข้อมูลในแนวนอน
2. การประมวลผลสมการ 1D-DCT ป้อนข้อมูลในแนวตั้ง

พิจารณาจากสมการที่ 3 สมการวงจร 2D-DCT เป็นสมการที่มีการคูณกันของเมตริกกับเมตริก สามารถที่จะแยกคำนวณเป็นสมการย่อยได้ 2 สมการตามลักษณะการประมวลผลของทั้ง 2 ขั้นตอน โดยค่าคงที่ทั้ง 2 เมตริกนั่นคือ C และ C^T จะเป็นตัวกำหนดลักษณะการคำนวณในขั้นตอนที่ 1 จะเป็นการนำค่าเมตริกข้อมูลคูณกับ C^T เพื่อให้เป็นการคูณแบบแนวนอนได้สมการ [8] ดังต่อไปนี้

$$Z = X \bullet C^T \qquad \text{สมการที่ (4)}$$

กำหนดให้ Z คือ ค่าผลลัพธ์ที่ได้จากขั้นตอนที่ 1
X คือ เมตริกข้อมูลที่ต้องการเข้ามาประมวลผล
C^T คือ เมตริกค่าคงที่สมการ 1D-DCT ที่สลับค่าระหว่างแนวตั้งกับแนวนอน

จากสมการที่ 4 เป็นสมการของการคำนวณของขั้นตอนที่ 1 ซึ่งได้ผลลัพธ์ คือ Z ส่งไปพักข้อมูลที่บับเฟอร์จนครบ 1 แถวในแนวตั้งจากนั้นจะส่งต่อไปประมวลผลต่อในขั้นตอนที่ 2 โดยขั้นตอนที่ 2 มี สมการ [8] การคำนวณดังต่อไปนี้

$$Y = C \bullet Z \qquad \text{สมการที่ (5)}$$

กำหนดให้ Z คือ ค่าผลลัพธ์ที่ได้จากขั้นตอนที่ 1
C คือ เมตริกค่าคงที่สมการ 1D-DCT
Y คือ เมตริกผลลัพธ์ของการคำนวณวงจร 2D-DCT

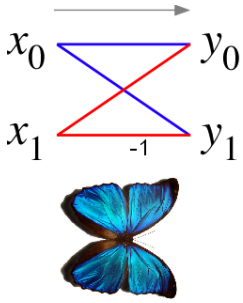
จากสมการที่ 5 เป็นสมการของการคำนวณของขั้นตอนที่ 2 ซึ่งได้ผลลัพธ์ คือ Y เป็นผลลัพธ์ของวงจรทั้งหมด นั่นคือ ผลลัพธ์ของการคำนวณสมการ 2D-DCT

2.5 เทคนิคการออกแบบการประมวลผลแบบขนาน

การประมวลผลแบบขนาน คือ รูปแบบการประมวลผลที่สามารถทำงานหลายงานได้ในช่วงเวลาเดียวกัน โดยแต่ละงานจะต้องไม่มีความเกี่ยวเนื่องกันของข้อมูล ซึ่งทำให้สามารถประมวลผลได้เร็วขึ้น เพราะมีการแบ่งระบบการทำงานออกเป็นส่วนๆที่แยกออกจากกัน จึงเป็นอีกวิธีหนึ่งที่จะทำให้ระบบการประมวลผลทำงานได้อย่างเต็มประสิทธิภาพ ซึ่งหากกล่าวถึงการออกแบบการประมวลผลแบบขนานในระดับฮาร์ดแวร์ สถาปัตยกรรม FPGA เป็นอุปกรณ์สมองกลชนิดหนึ่งที่สามารถทำการออกแบบวงจรดิจิตอลการคำนวณต่างๆ ให้มีการประมวลผลแบบขนานได้

โครงสร้างการประมวลผลที่สามารถนำมาออกแบบบนอุปกรณ์ FPGA ให้ประมวลผลแบบขนานได้ คือ การประมวลผลแบบกระจาย (Distributed Computation) [12] เป็นการออกแบบวงจรให้มีการแบ่งงานออกเป็นส่วนย่อยๆ โดยการออกแบบมีการแบ่งข้อมูลที่ใช้ในการประมวลผลออกเป็นกลุ่มต่างๆ ตามความเหมาะสมของลักษณะงาน จากนั้นกระจายข้อมูลไปยังงานในส่วนต่างๆ เพื่อประมวลผลพร้อมกัน

Reza [10] ได้กล่าวถึงเทคนิคการออกแบบวงจรคำนวณสมการ 2D-DCT โดยใช้โครงสร้างวงจรคำนวณทางคณิตศาสตร์การกระจายแบบผีเสื้อ(Distributed Arithmetic with Butterfly Computation) เป็นการคำนวณซึ่งใช้เทคนิคการคำนวณแบบขนาน มีรูปร่างโครงสร้างที่คล้ายกับปีกผีเสื้อดังภาพประกอบที่ 2-7



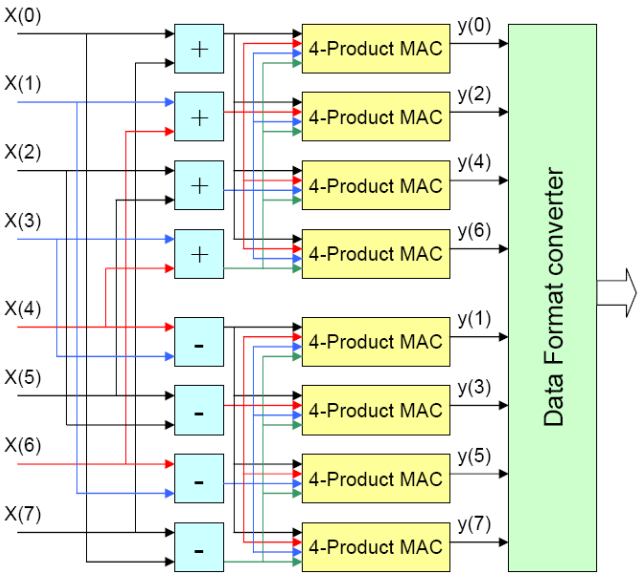
ภาพประกอบ 2-7 แสดงภาพโครงสร้างการคำนวณแบบผีเสื้อ [12]

จากภาพประกอบที่ 2-7 แสดงลักษณะการคำนวณวงจรทางคณิตศาสตร์แบบผีเสื้อเป็นการนำค่าตัวเลข 2 จำนวน มาจับคู่กันแล้วทำการบวกและลบกันจะได้ผลลัพธ์ 2 ผลลัพธ์ แสดงดังสมการที่ 6 และ สมการที่ 7

$$Y_0 = X_0 + X_1 \qquad \text{สมการที่ (6)}$$

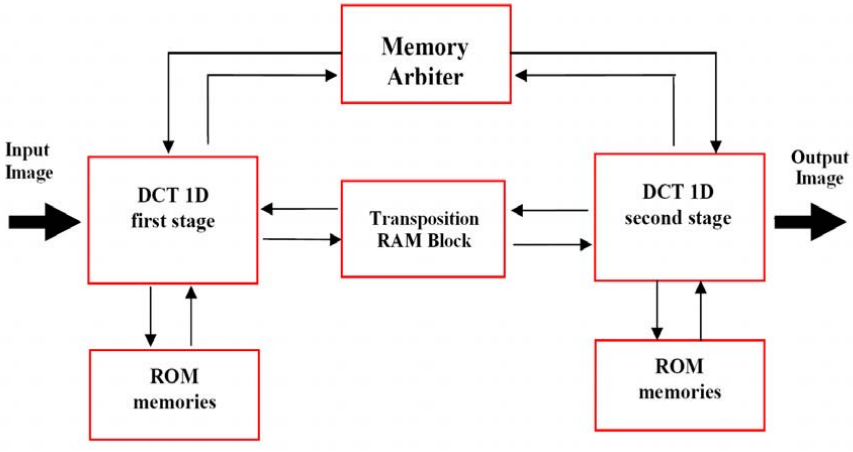
$$Y_1 = X_0 - X_1 \qquad \text{สมการที่ (7)}$$

ด้วยโครงสร้างของวงจรการออกแบบการคำนวณ 1D-DCT เป็นการนำข้อมูลในแต่ละแถวคูณกับค่าคงที่ของเมตริกค่าคงที่ DCT โดยทำการออกแบบนำเมตริกข้อมูลเข้ามาประมวลผล มีการจับคู่ของข้อมูลโดยใช้โครงสร้างแบบผีเสื้อเข้าสู่วงจรคูณแบบ 4-product MAC จากนั้นทำการแปลงข้อมูลนำมาเรียงกันก่อนส่งไปประมวลผลในขั้นตอนถัดไป แผนผังโครงสร้างการคำนวณแสดงดังภาพประกอบ 2-8



ภาพประกอบ 2-8 แสดงโครงสร้างวงจร 1D-DCT ออกแบบโดย Reza [10]

การออกแบบวงจรการคำนวณสมการ 2D-DCT ออกแบบโดย Reza ได้ใช้วงจรการคำนวณสมการ 1D-DCT เข้ามาช่วยในการประมวลผล โดยใช้การประมวลผลแบบเวคเตอร์ ดังเช่น ได้กล่าวเอาไว้ในหัวข้อที่ 2.4 การส่งข้อมูลจากการประมวลผลสมการ 1D-DCT ขั้นตอนที่ 1 ไปยังวงจร 1D-DCT ขั้นตอนที่ 2 มีการพักข้อมูลในหน่วยความจำ(Memory) ที่สร้างขึ้นบนแผ่นลอจิกของ FPGA โดยถูกสร้างขึ้นมีลักษณะหน่วยความจำแบบคู่ (Dual Buffer) เพื่อที่จะสามารถสลับกันทำหน้าที่รับและส่งข้อมูลจากขั้นตอนที่ 1 ไปยังขั้นตอนที่ 2 เป็นการลดระยะเวลาในการประมวลผล และมีการเก็บค่าคงที่ 1D-DCT ที่ใช้ในการประมวลผลไว้ในหน่วยความจำที่อยู่ในอุปกรณ์ (ROM) เพื่อประหยัดทรัพยากรของแผ่นลอจิก โครงสร้างการออกแบบของ Reza ดังภาพประกอบที่ 2-9



ภาพประกอบ 2-9 แสดงแผนผังโครงสร้าง 1D-DCT ออกแบบโดย Reza

จากการวงจรการคำนวณสมการ 1D-DCT ออกแบบโดย Reza ดังภาพประกอบ 2-9 สามารถนำเทคนิคการประมวลผลขนานแบบผีเสื้อ มาประยุกษ์เพื่อเพิ่มประสิทธิภาพด้านความเร็วในการประมวลผลได้เพราะสามารถแบ่งแยกการประมวลผลให้ได้ผลลัพธ์หลายๆ จำนวนในช่วงเวลาเดียวกัน แต่จะมีความแตกต่างคือ โครงสร้างการออกแบบจะไม่มีการพักข้อมูลจากการประมวลผลสมการ 1D-DCT ขั้นตอนที่ 1 ส่งไปยังการจากการประมวลผลสมการ 1D-DCT ขั้นตอนที่ 2 เพื่อเป็นการลดระยะเวลาในการประมวลผลซึ่งรายละเอียดการออกแบบจะกล่าวในลำดับถัดไปในบทที่ 3

2.6 วงจรการคำนวณทางคณิตศาสตร์ออกแบบโดยผู้ผลิตอุปกรณ์

การออกแบบวงจรโดยใช้วงจรการคำนวณทางคณิตศาสตร์ซึ่งออกแบบโดยผู้ผลิต (Macro Circuit) เป็นการให้โปรแกรมทำการออกแบบวงจรการคำนวณทางคณิตศาสตร์ (Arithmetic Circuit) เป็นวงจรที่ได้ทำการออกแบบให้มีความเหมาะสมกับโครงสร้างของ FPGA ที่ใช้ในการพัฒนาทำให้ได้วงจรที่มีประสิทธิภาพสูงทั้งด้านการใช้ทรัพยากรและความเร็วในการประมวลผล และนอกจากนั้นใน FPGA บางชนิดมีอุปกรณ์สำหรับใช้ในงานคำนวณทางคณิตศาสตร์โดยเฉพาะ ซึ่งเป็นอุปกรณ์ประมวลผลคำนวณงานด้านสัญญาณดิจิตอล (Digital Signal Processing) โดยในการทำวิทยานิพนธ์ฉบับนี้ได้เลือกใช้อุปกรณ์ FPGA Xilinx Virtex4 XC4VFX12 เป็นอุปกรณ์ที่มีวงจรการคำนวณทางคณิตศาสตร์อยู่ภายในที่มีชื่อว่า “ DSP48 ”

DSP48 เป็นอุปกรณ์การประมวลผลวงจรสัญญาณดิจิตอลที่ใช้สำหรับคำนวณสมการทางคณิตศาสตร์ ได้ถูกออกแบบนำมาใช้ในสถาปัตยกรรม FPGA Xilinx ตระกูล Virtex4, Virtex5 และ Virtex6 จุดเด่นของ DSP48 คือ เป็นอุปกรณ์การคำนวณวงจรทางคณิตศาสตร์ที่สามารถประมวลผลคำนวณ วงจรคูณที่มีขนาด 18 บิต คูณกับ 18 บิต มีขนาดผลลัพธ์เท่ากับ 36 บิต รวมกับบิตแสดงสัญลักษณ์จำนวนเต็มบวกหรือจำนวนเต็มลบ (Sign Bit) ได้ผลลัพที่มีขนาดสูงสุด 48 บิต สามารถคำนวณวงจรบวก-ลบ 3 จำนวน ที่มีขนาดผลลัพธ์ไม่เกิน 48 บิตได้

การเรียกใช้ DSP48 นอกจากจะเป็นการเพิ่มประสิทธิภาพด้านความเร็วในการคำนวณวงจรการคำนวณทางคณิตศาสตร์แล้ว ยังทำให้ประหยัดแผ่นลอจิกที่ใช้ในการออกแบบวงจรเพราะ DSP48 เป็นเครื่องมือที่อยู่ในอุปกรณ์ FPGA ซึ่งไม่จำเป็นต้องสิ้นเปลืองแผ่นลอจิกเพื่อใช้ในการออกแบบวงจร ซึ่งการเรียกใช้ DSP48 สามารถทำได้โดยกำหนดค่าการสังเคราะห์ของโปรแกรม Xilinx 8.1 ให้มีการเรียกใช้ในการสังเคราะห์วงจร ซึ่งโปรแกรมจะเรียกใช้และวิเคราะห์กำหนดตำแหน่งที่จะทำการแทรกเครื่องมือ DSP48 เข้าไปให้มีความเหมาะสมมากที่สุดในการ

ออกแบบวงจร ซึ่งในงานวิทยานิพนธ์ได้มีการใช้ทั้งวงจรมาโคร และ DSP48 เข้ามาช่วยในการพัฒนาเพื่อเพิ่มประสิทธิภาพด้านความเร็วให้แก่วงจรการคำนวณสมการ 2D-DCT

2.7 บทสรุป

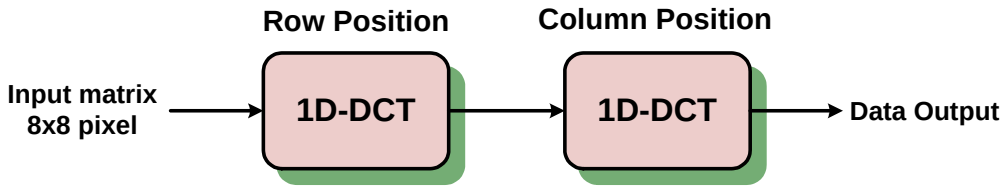
การเพิ่มความเร็วให้กับระบบการบีบอัดรูปภาพแบบ JPEG ขั้นตอนที่ควรปรับปรุงมากที่สุด คือ ขั้นตอนการคำนวณสมการ 2D-DCT เพราะเป็นวงจรการคำนวณที่สูญเสียเวลาในการประมวลผลมากที่สุดของขั้นตอนการบีบอัดรูปภาพทั้งหมด ดังนั้นวิทยานิพนธ์ชุดนี้จากทฤษฎีและงานวิจัยที่เกี่ยวข้องที่กล่าวมาข้างต้นเกี่ยวกับเทคนิคการประมวลผลวงจร 2D-DCT และเทคนิคการประมวลผลแบบขนาน จึงเสนอแนวทางการพัฒนาวงจรการคำนวณสมการ 2D-DCT เพื่อเพิ่มประสิทธิภาพด้านความเร็วให้กับวงจรการบีบอัดรูปภาพแบบ JPEG ใช้แนวทางจากตัวอย่างงานวิจัยที่ได้นำเสนอข้างต้นมาพัฒนาปรับปรุง โดยภายในวงจรจะนำเทคโนโลยีการประมวลผลแบบขนานการกระจายข้อมูลแบบผีเสื้อ เข้าไปผสมผสานกับเทคนิคการใช้สมการ 1D-DCT ออกแบบและปรับปรุงวงจรใหม่เพื่อลดเวลาในการประมวลผล ซึ่งในบทถัดไปจะนำเสนอการวิเคราะห์และวิธีการออกแบบพัฒนาวงจรการคำนวณ 2D-DCT

บทที่ 3

การออกแบบวงจร 2D-DCT ประมวลผลแบบขนาน

ในบทนี้จะกล่าวถึงวิธีการวิเคราะห์การทำงานของ 2D-DCT เพื่อออกแบบสถาปัตยกรรมวงจร 2D-DCT โดยใช้เทคนิคการประมวลผลแบบขนานเข้ามาช่วยเพิ่มความเร็วในการประมวลผล เพื่อนำไปใช้เป็นส่วนหนึ่งของการพัฒนาการบีบอัดรูปภาพแบบ JPEG บน ระบบสมองกลฝังตัวที่ใช้ FPGA ได้ โดยจะแบ่งเป็นหัวข้อย่อยๆ ของการออกแบบในแต่ละส่วนภายในวงจร การนำแต่ละขั้นตอนมารวมกัน และท้ายที่สุดจะกล่าวถึงบทสรุปที่ได้จากการออกแบบวงจร 2D-DCT

วิธีการออกแบบวงจรดิจิตอลการคำนวณ 2D-DCT ที่ได้นำเสนอในหัวข้อที่ 2.4 ประกอบไปด้วย วงจรการคำนวณสมการ 1D-DCT ที่ทำการประมวลผล 2 ครั้ง นั่นคือในแนวตั้งและแนวนอนของเมตริกข้อมูล โดยขั้นตอนที่อยู่ระหว่างการคำนวณสมการ 1D-DCT ทั้ง 2 วงจร จะใช้วงจรบัพเฟอร์ในการเก็บข้อมูลที่ได้จากวงจร 1D-DCT ขั้นตอนที่ 1 เพื่อให้ครบตามจำนวนที่วงจร 1D-DCT ขั้นตอนที่ 2 ต้องการใช้ในการประมวลผล ซึ่งในขั้นตอนนี้จะสิ้นเปลืองเวลาเพื่อรอให้วงจรในขั้นตอนที่ 1 ประมวลผลให้ครบตามจำนวนที่ต้องการก่อนที่จะส่งต่อไปในขั้นตอนถัดไป ดังนั้นได้นำปัญหาในจุดนี้มาแก้ไข โดยการออกแบบวงจรใหม่ให้วงจร 1D-DCT ขั้นตอนที่ 1 สามารถประมวลผลข้อมูลตามที่วงจร 1D-DCT ขั้นตอนที่ 2 ต้องการให้เสร็จใน 1 ช่วงเวลา (Clock Cycle) ซึ่งจะทำให้ไม่จำเป็นต้องสร้างวงจรบัพเฟอร์เพื่อเก็บข้อมูลและเป็นการลดเวลาที่สิ้นเปลืองในการประมวลผล วงจร 2D-DCT ใหม่ที่ไม่มีวงจรบัพเฟอร์แสดงดังภาพประกอบ 3-1



ภาพประกอบ 3-1 แสดงวงจร 2D-DCT โดยไม่มีการพักข้อมูลที่วงจรบัพเฟอร์

จากภาพประกอบ 3-1 วงจร 2D-DCT เมื่อไม่มีการพักข้อมูลที่วงจรบัพเฟอร์จึงสามารถที่จะส่งผ่านข้อมูลจากขั้นตอนที่ 1 ไปยังขั้นตอนที่ 2 ได้ทันที นั่นคือเป็นการลดเวลาที่ใช้ในการประมวลผล โดยมีรายละเอียดการออกแบบในแต่ละขั้นตอนดังต่อไปนี้

3.1 การออกแบบวงจร 2D-DCT ขั้นตอนที่ 1

การคำนวณสมการ 2D-DCT นั้นจะใช้สมการ 1D-DCT คำนวณ 2 ครั้งในแนวนอนและแนวตั้งดังที่ได้กล่าวไว้ในบทที่ 2 ดังนั้นในการออกแบบวงจร 2D-DCT จากภาพประกอบ 3-1 ขั้นตอนที่ 1 ในการประมวลผลจะเป็นการออกแบบวงจรสมการ 1D-DCT ประมวลผลข้อมูลในแนวนอน ดังสมการที่ 4 ในบทที่ 2 เป็นการคูณกันระหว่างเมตริก X กับเมตริก C^T ขนาด 8x8

ให้เมตริก X เป็นเมตริกข้อมูลที่จะนำไปประมวลผลมีลักษณะดังนี้

$$X = \begin{bmatrix} X_{00} & X_{01} & X_{02} & X_{03} & X_{04} & X_{05} & X_{06} & X_{07} \\ X_{10} & X_{11} & X_{12} & X_{13} & X_{14} & X_{15} & X_{16} & X_{17} \\ X_{20} & X_{21} & X_{22} & X_{23} & X_{24} & X_{25} & X_{26} & X_{27} \\ X_{30} & X_{31} & X_{32} & X_{33} & X_{34} & X_{35} & X_{36} & X_{37} \\ X_{40} & X_{41} & X_{42} & X_{43} & X_{44} & X_{45} & X_{46} & X_{47} \\ X_{50} & X_{51} & X_{52} & X_{53} & X_{54} & X_{55} & X_{56} & X_{57} \\ X_{60} & X_{61} & X_{62} & X_{63} & X_{64} & X_{65} & X_{66} & X_{67} \\ X_{70} & X_{71} & X_{72} & X_{73} & X_{74} & X_{75} & X_{76} & X_{77} \end{bmatrix}$$

นำมาคูณกับเมตริก C^T มีลักษณะดังนี้

$$C^T = \begin{bmatrix} 23170 & 32138 & 30274 & 27246 & 32170 & 18205 & 12540 & 6393 \\ 23170 & 27246 & 12540 & -6393 & -23170 & -32138 & -30274 & -18205 \\ 23170 & 18205 & -12540 & -32138 & -23170 & 6393 & 30274 & 27246 \\ 23170 & 6393 & -30274 & -18205 & 23170 & 27246 & -12540 & -32138 \\ 23170 & -6393 & -30274 & 18205 & 23170 & -27246 & -12540 & 32138 \\ 23170 & -18205 & -12540 & 32138 & -23170 & -6393 & 30274 & -27246 \\ 23170 & -27246 & 12540 & 6393 & -23170 & 32138 & -30274 & 18205 \\ 23170 & -32138 & 30274 & -27246 & 23170 & -18205 & 12540 & -6393 \end{bmatrix}$$

การคูณกันของเมตริกดังสมการที่ 4 คือ การที่นำตัวเลขในแนวนอนของเมตริก X 1 แถว 8 ตัวเลข มาคูณกับตัวเลขแนวตั้งของเมตริก C^T ตำแหน่งต่อตำแหน่ง จากนั้นทำการบวกค่าที่ได้จากการคูณทั้ง 8 ตัวเข้าด้วยกันได้คำตอบ 1 ตัวเลข ตำแหน่งตามลำดับของแถวแนวนอนของเมตริก X และแถวแนวตั้งของเมตริก C^T เช่น ตัวเลขของแถวหมายเลข 0 ตามแนวนอนของเมตริก X นั่นคือ X_{00} ถึง X_{07} มาคูณกับตัวเลขแนวตั้งหมายเลข 2 ของเมตริก C^T นั่นคือ 30274 , 12540 , ... , 30274 คูณตำแหน่งต่อตำแหน่งจากนั้นนำมาบวกกันทั้ง 8 ตัวเลข จะได้ผลลัพธ์ของเมตริก Z จำนวน 1 ตัวเลข คือ ตำแหน่ง (0 , 2) หากใช้ข้อมูลแถวหมายเลข 0 ของเมตริก X คูณกับแนวตั้ง

ทุกๆ แถวของเมตริก C^T สามารถสรุปเป็นสมการการคูณกันของเมตริกได้ผลลัพธ์ของเมตริก Z จำนวน 1 แถวในแนวนอน คือ Z_{00} ถึง Z_{07} ดังต่อไปนี้

$$Z_{(0,0)} = 23170(X_{00} + X_{01} + X_{02} + X_{03} + X_{04} + X_{05} + X_{06} + X_{07})$$

$$Z_{(0,1)} = 32138\, X_{00} + 27246\, X_{01} + 18205\, X_{02} + 6393\, X_{03} - 6393\, X_{04} - 18205\, X_{05} - 27246\, X_{06} - 32138\, X_{07}$$

$$= 32138(X_{00} - X_{07}) + 27246(X_{01} - X_{06}) + 18205(X_{02} - X_{05}) + 6393(X_{03} - X_{04})$$

$$Z_{(0,2)} = 30274(X_{00} + X_{07}) + 12540(X_{01} + X_{06}) - 12540(X_{02} + X_{05}) - 30274(X_{03} + X_{04})$$

$$Z_{(0,3)} = 27246(X_{00} - X_{07}) - 6393(X_{01} - X_{06}) - 32138(X_{02} - X_{05}) - 18205(X_{03} - X_{04})$$

$$Z_{(0,4)} = 23170(X_{00} + X_{07}) - 23170(X_{01} + X_{06}) - 23170(X_{02} + X_{05}) - 23170(X_{03} + X_{04})$$

$$Z_{(0,5)} = 18205(X_{00} - X_{07}) - 32138(X_{01} - X_{06}) + 6393(X_{02} - X_{05}) + 27246(X_{03} - X_{04})$$

$$Z_{(0,6)} = 12540(X_{00} + X_{07}) - 30274(X_{01} + X_{06}) + 30274(X_{02} + X_{05}) - 12540(X_{03} + X_{04})$$

$$Z_{(0,7)} = 6393(X_{00} - X_{07}) - 18205(X_{01} - X_{06}) + 27246(X_{02} - X_{05}) - 32138(X_{03} - X_{04})$$

จากสมการการคูณกันของเมตริก X กับเมตริก C^T หากพิจารณาเมตริกค่าคงที่ C^T จะสังเกตได้ว่าตัวเลขในทุกๆ แถวเดียวกันในแนวตั้งของเมตริก C^T จะมีตำแหน่งที่มีตัวเลขเหมือนกันที่สามารถนำมาจับคู่กันได้ 4 คู่ นั่นคือ ตำแหน่งในแนวนอนที่ 0 กับ 7 , 1 กับ 6 , 2 กับ 5 และ 3 กับ 4 แต่จะมีอย่างหนึ่งที่แตกต่างกันคือ ทุกๆ แถวในแนวตั้งที่เป็นเลขคู่นั่นคือแถวที่ 0 , 2 , 4 และ 6 เครื่องหมายแสดงค่าบวกหรือลบของค่าคงที่คู่นั้นจะเหมือนกัน แต่ในทุกๆ แถวในแนวตั้งที่เป็นเลขคี่นั่นคือแถวที่ 1 , 3 , 5 และ 7 เครื่องหมายแสดงค่าบวกหรือลบของค่าคงที่คู่นั้นจะต่างกัน

ดังนั้นเพื่อง่ายต่อการคำนวณและออกแบบวงจรดิจิตอลจะทำการจับคู่จำนวนที่เหมือนกัน และพิจารณากรณีคู่ที่เครื่องหมายแสดงค่าบวกหรือลบเหมือนกัน ตัวเลขของเมตริก X ที่นำมาคูณค่าคงที่ของทั้งคู่จะนำมาบวกกัน กรณีที่เครื่องหมายแสดงค่าบวกหรือลบต่างกัน ตัวเลขของเมตริก X ที่นำมาคูณค่าคงที่ของทั้งคู่จะนำมาลบกัน เหมือนกับเป็นการดึงค่าคงที่ร่วมออกมาเพื่อจัดการค่า X ก่อน หลังจากนั้นจึงจะนำไปคูณกับค่าคงที่นั้นต่อไป ลำดับสุดท้ายหลังจากการคูณกับค่าคงที่ทั้ง 4 กลุ่มแล้วจึงทำการบวกค่าทั้ง 4 กลุ่มเข้าด้วยกันได้ผลลัพธ์ของการคำนวณสมการ 1D-DCT จำนวน 1 ตำแหน่ง ตามหมายเลขในแนวนอนและแนวตั้งของเมตริกที่ใช้คูณกัน

หากใช้เมตริก X ข้อมูล 1 แถวในแนวนอน คูณกับเมตริก C^T ในแนวตั้งทุกแถวจะได้ผลลัพธ์ Z จำนวน 8 ตัวเลขในแนวนอนดังสมการต่อไปนี้

$$Z_{(k,0)} = 23170(X_{k0} + X_{k1} + X_{k2} + X_{k3} + X_{k4} + X_{k5} + X_{k6} + X_{k7})$$

$$Z_{(k,1)} = 32138(X_{k0} - X_{k7}) + 27246(X_{k1} - X_{k6}) + 18205(X_{k2} - X_{k5}) + 6393(X_{k3} - X_{k4})$$

$$Z_{(k,2)} = 30274(X_{k0} + X_{k7}) + 12540(X_{k1} + X_{k6}) - 12540(X_{k2} + X_{k5}) - 30274(X_{k3} + X_{k4})$$

$$Z_{(k,3)} = 27246(X_{k0} - X_{k7}) - 6393(X_{k1} - X_{k6}) - 32138(X_{k2} - X_{k5}) - 18205(X_{k3} - X_{k4})$$

$$Z_{(k,4)} = 23170(X_{k0} + X_{k7}) - 23170(X_{k1} + X_{k6}) - 23170(X_{k2} + X_{k5}) - 23170(X_{k3} + X_{k4})$$

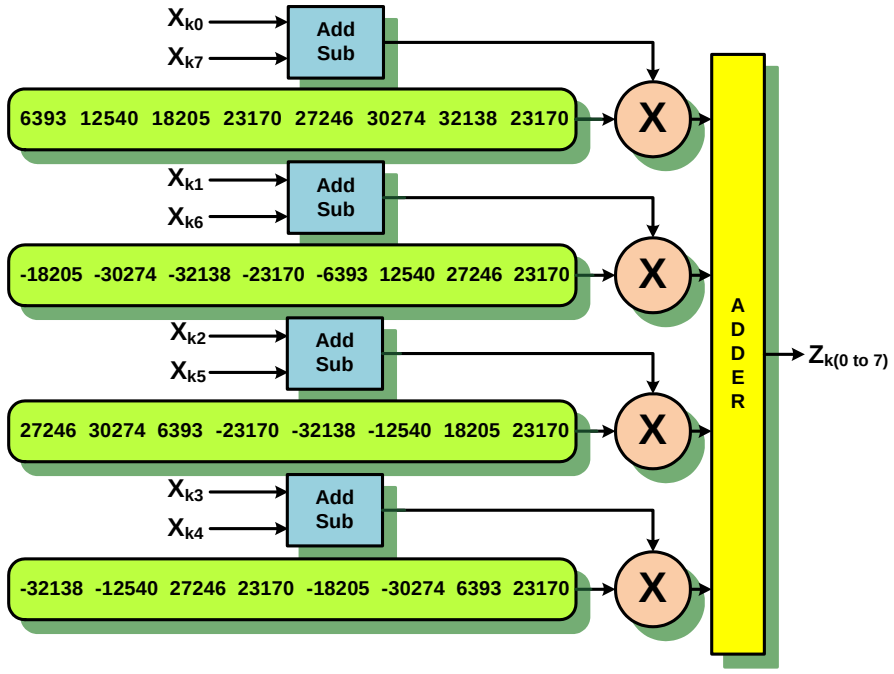
$$Z_{(k,5)} = 18205(X_{k0} - X_{k7}) - 32138(X_{k1} - X_{k6}) + 6393(X_{k2} - X_{k5}) + 27246(X_{k3} - X_{k4})$$

$$Z_{(k,6)} = 12540(X_{k0} + X_{k7}) - 30274(X_{k1} + X_{k6}) + 30274(X_{k2} + X_{k5}) - 12540(X_{k3} + X_{k4})$$

$$Z_{(k,3)} = 6393(X_{k0} - X_{k7}) - 18205(X_{k1} - X_{k6}) + 27246(X_{k2} - X_{k5}) - 32138(X_{k3} - X_{k4})$$

กำหนดให้ k คือ หมายเลขแสดงลำดับของแถว

จากสมการ Z แสดงการคูณของเมตริก X กับเมตริกค่าคงที่ C^T จากที่กล่าวมาข้างต้น สมการหาค่า Z เกิดจากผลรวมของกลุ่มที่จับคู่กัน 4 กลุ่มของตัวเลขค่าคงที่นำมาคูณกับผลบวกหรือลบของเมตริก X ซึ่งสามารถที่จะออกแบบวงจรดิจิตอลให้คำนวณแบบขนาน 4 แถว ดังภาพประกอบ 3.2 แสดงแผนภาพการออกแบบวงจรดิจิตอลของสมการ 1D-DCT ขั้นตอนที่ 1 ของวงจรคำนวณสมการ 2D-DCT



ภาพประกอบ 3-2 แผนภาพวงจรคำนวณสมการ 1D-DCT ส่วนที่ 1 จำนวน 1 ผลลัพธ์

จากภาพประกอบ 3-2 เป็นแผนภาพแสดงการออกแบบวงจรดิจิตอล 1D-DCT ซึ่งเป็นขั้นตอนส่วนที่ 1 ของวงจร 2D-DCT สำหรับในวงจรจะทำการออกแบบวงจรให้ประมวลผลได้เร็วขึ้น คือออกแบบวงจรให้ประมวลผลแบบขนานจำนวน 4 แถว ภายในวงจรจะประกอบไปด้วยวงจรบวกหรือลบซึ่งรับข้อมูลจากเมตริก X ซึ่งเป็นเมตริกข้อมูลที่ป้อนเข้ามาจำนวน 1 แถวในแนวระนาบ โดยจะจับคู่ของเมตริก X ในแถวเดียวกันเป็น 4 กลุ่ม ตามรูปแบบของการหาค่า Z_{k0} จนถึง Z_{k7} ตามที่กล่าวไว้ข้างต้น โดยค่าเมตริก X ที่จับคู่กันจะนำไปประมวลผลในวงจรบวกหรือลบขึ้นอยู่กับเครื่องหมายของกลุ่มค่าคงที่ของเมตริก C^T

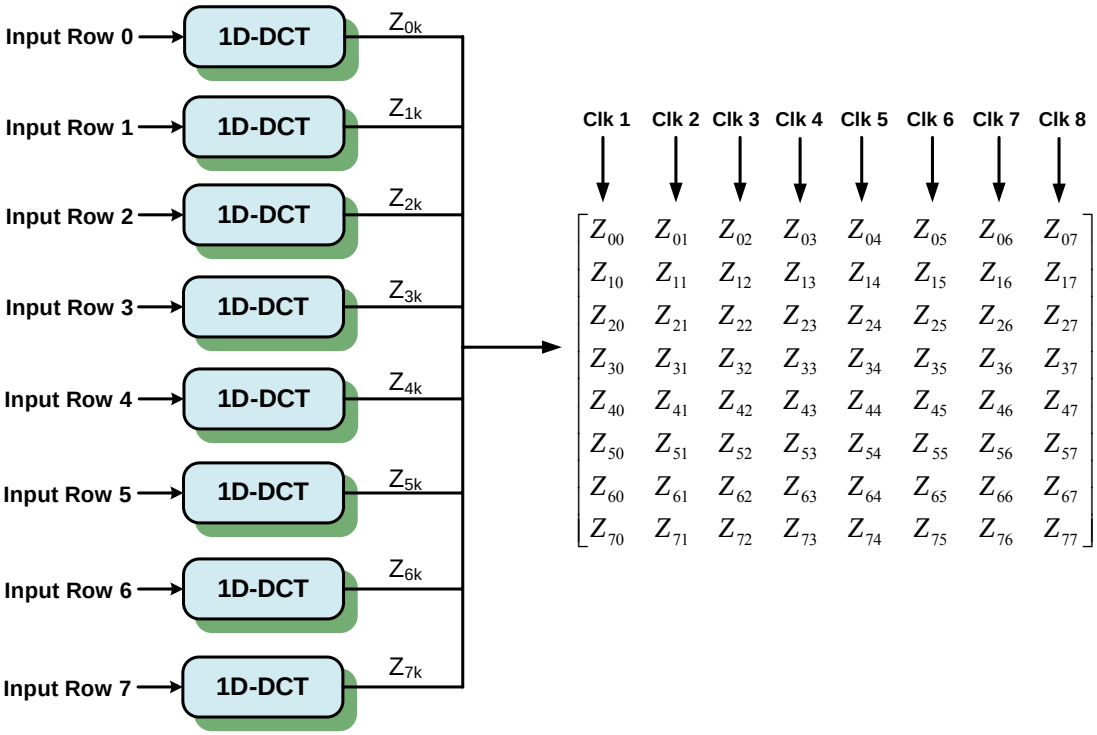
ส่วนถัดมาคือค่าคงที่จะถูกเก็บอยู่ใน RAM ให้เรียงลำดับการป้อนข้อมูลตามลำดับตั้งแต่ Z_{k0} ไปจนถึง Z_{k7} การป้อนข้อมูลของ RAM จะถูกควบคุมด้วยสัญญาณนาฬิกาโดยจะเปลี่ยนค่าของการป้อนข้อมูลทุกๆ 1 รอบลูกคลื่น ดังนั้นกลุ่มของค่า X ที่รับเข้ามาจะบวกหรือลบจะขึ้นอยู่กับการป้อนข้อมูลของ RAM ด้วย ในที่นี้ค่าเมตริก Z ในตำแหน่งคู่จะบวกค่าของ X ส่วนในตำแหน่งคี่จะลบค่าของ X สลับกันไปเรื่อยๆ ทุกๆ ลูกคลื่น

ส่วนสุดท้ายคือการรับค่าที่ได้จากการคูณแบบขนานทั้ง 4 แถวเข้ามาบวกรวมกันเป็นผลลัพธ์ของวงจร 1D-DCT จำนวน 1 ผลลัพธ์ พิกัดเดียวกับหมายเลขแนวระนาบของเมตริก X และหมายเลขในแนวตั้งของเมตริกค่าคงที่ C^T ในทุกๆ ส่วนของการออกแบบวงจร 1D-DCT ขั้นตอนที่ 1 การออกแบบด้วยภาษา VHDL จะประกาศให้ผลลัพธ์ที่ได้จากการประมวลผลในส่วนต่างๆ ภายในวงจร 1D-DCT อยู่ในรูปของเส้นสัญญาณดิจิตอล (Signal) ที่ทำการเชื่อมต่อทุกๆส่วนเข้าด้วยกันโดยไม่มีการพักข้อมูลหรือเก็บไว้ในรีจิสเตอร์ (Register) จึงทำให้ใน 1 ช่วงสัญญาณนาฬิกาวงจร 1D-DCT ที่ทำการออกแบบ สามารถที่จะประมวลผลเสร็จได้ผลลัพธ์ Z จำนวน 1 ผลลัพธ์

การที่จะนำข้อมูลที่ได้จากวงจร 1D-DCT ขั้นตอนส่วนที่ 1 ไปประมวลผลต่อในขั้นตอนส่วนที่ 2 เนื่องจากเป็นการใช้ค่าคงที่เมตริก C จำนวน 1 แถวในแนวนอน คูณกับเมตริก Z ในแนวตั้งดังสมการที่ 5 ในบทที่ 2 จำเป็นที่จะต้องใช้ข้อมูลจำนวน 1 แถวในแนวตั้งของเมตริก Z แต่เนื่องจากวงจร 1D-DCT ดังภาพประกอบ 3-2 สามารถประมวลผลได้จำนวน 1 ผลลัพธ์ต่อ 1 ลูกคลื่นสัญญาณนาฬิกา ดังนั้นเพื่อลดเวลาในการประมวลผลวิธีที่ต้องการให้ประมวลผลเมตริก Z ให้เสร็จ 1 แถวในแนวตั้ง ภายใน 1 ลูกคลื่นสัญญาณนาฬิกาและไม่จำเป็นต้องสร้างวงจรบัฟเฟอร์ขึ้นมาเพื่อรอเก็บค่า Z คือ การประมวลผลแบบขนาน

ลักษณะการออกแบบวงจรให้ประมวลผลแบบขนานให้ได้ผลลัพธ์เมตริก Z จำนวน 1 แถวในแนวตั้งคือสร้างวงจรดังภาพประกอบ 3-2 ขึ้นมา 8 วงจร รับค่าเมตริกข้อมูล X

เข้ามาวงจรละ 1 แถว นั่นคือเป็นการรับข้อมูลเมตริก X เข้ามาพร้อมกัน แยกประมวลผลแต่ละแถวด้วยวงจร 1D-DCT จะทำให้ได้ผลลัพธ์เมตริก Z จำนวน 8 ผลลัพธ์ในแนวตั้งดังภาพประกอบ 3-3



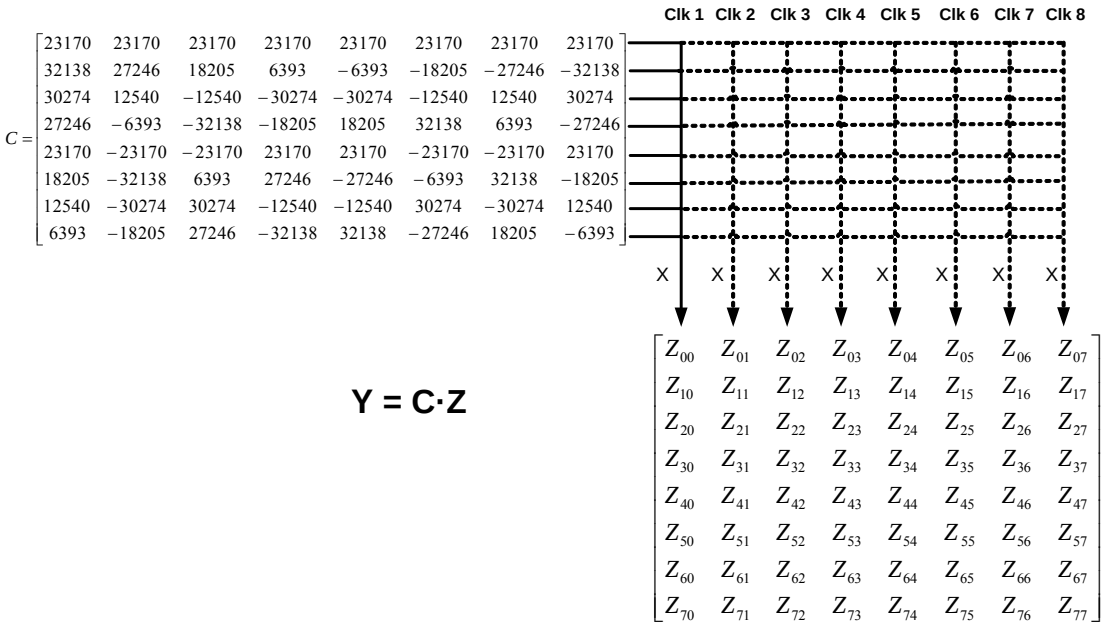
ภาพประกอบ 3-3 แผนภาพวงจรคำนวณสมการ 1D-DCT ส่วนที่ 1 จำนวน 8 ผลลัพธ์

จากภาพประกอบ 3-3 เป็นการประมวลผลแบบขนานของวงจร 1D-DCT ซึ่งรับข้อมูลเข้า มา 8 แถว นั่นคือ 1 เมตริกข้อมูล แยกประมวลผลแต่ละแถวจะได้ผลลัพธ์จำนวน 1 แถวในแนวตั้งตัวอย่างเช่น ในช่วงเวลาลูกคลื่นแรกของสัญญาณนาฬิกา ในแต่ละวงจรจะส่งค่าคงที่ที่ใช้คำนวนค่า Z_{k0} เมื่อ k คือ หมายเลขแถวของข้อมูลที่รับเข้ามา วงจรที่รับแถวหมายเลข 0 จะให้ผลลัพธ์คือ Z_{00} วงจรที่รับแถวหมายเลข 1 จะให้ผลลัพธ์คือ Z_{10} ไปจนถึงแถวหมายเลขที่ 7 จึงทำให้ในช่วงสัญญาณนาฬิกาครั้งแรก จะได้ค่าผลลัพธ์จำนวน 8 ผลลัพธ์ในแนวตั้ง Z_{00} , Z_{10} ,....., Z_{70} และในช่วงสัญญาณนาฬิกาถัดมาจะได้ผลลัพธ์อีก 8 ผลลัพธ์ คือ Z_{01} , Z_{11} ,....., Z_{71} ครบ 8 ลูกคลื่นสัญญาณนาฬิกาจะได้ผลลัพธ์ของสมการ 1D-DCT ขั้นตอนที่ 1 จำนวนผลลัพธ์ 1 เมตริก ในการออกแบบทุกๆ 1 ช่วงสัญญาณนาฬิกาจะได้ผลลัพธ์ 1 แถวในแนวตั้ง ซึ่งผลลัพธ์นี้ในทุกๆ 1 ลูกคลื่นสัญญาณนาฬิกา จะทำการส่งต่อไปยังการประมวลผลวงจร 1D-DCT ขั้นตอนที่ 2 ทันทีโดยไม่มีการพักข้อมูลเพราะข้อมูลผลลัพธ์ที่ได้จากขั้นตอนที่ 1 ในทุกๆ 1 ลูกคลื่นสัญญาณนาฬิกาเพียงพอที่จะใช้ในการประมวลผลในขั้นตอนถัดไป

วงจร 1D-DCT ในส่วนที่ 1 ได้ออกแบบให้รับข้อมูลเป็นเลขฐาน 2 ขนาดข้อมูลแต่ละตัวจะมีขนาด 16 บิต เข้าไปประมวลผลในวงจรขนาด 27 บิตและเนื่องจากค่าคงที่ที่ได้กำหนดไว้เป็นเลขทศนิยมขนาด 16 บิต ดังนั้นเมื่อส่งข้อมูลผ่านไปยังขั้นตอนต่อไปจะทำการส่งไปเฉพาะบิตที่ 17 จนถึงบิตที่ 27 ซึ่งมีขนาด 11 บิต การประมวลผลวงจรทั้งหมด จะประมวลผลใช้ลักษณะการคำนวนตัวเลขแบบจำนวนเต็ม (Fixed Point Number)

3.2 การออกแบบวงจร 2D-DCT ขั้นตอนที่ 2

วงจร 2D-DCT ขั้นตอนที่ 2 เป็นวงจรประมวลผลสมการ 1D-DCT ที่รับข้อมูลมาจากวงจรสมการ 1D-DCT ขั้นตอนที่ 1 ซึ่งการออกแบบจะออกแบบดังสมการที่ 5 ดังที่กล่าวไว้ในบทที่ 2 เป็นสมการที่ใช้เมตริกค่าคงที่ C คูณกับผลลัพธ์ที่ได้จากขั้นตอนที่ 1 เป็นข้อมูลขนาด 1 แถวในแนวตั้งของเมตริกขนาด 8x8 มีลักษณะการประมวลผลดังภาพประกอบ 3-4

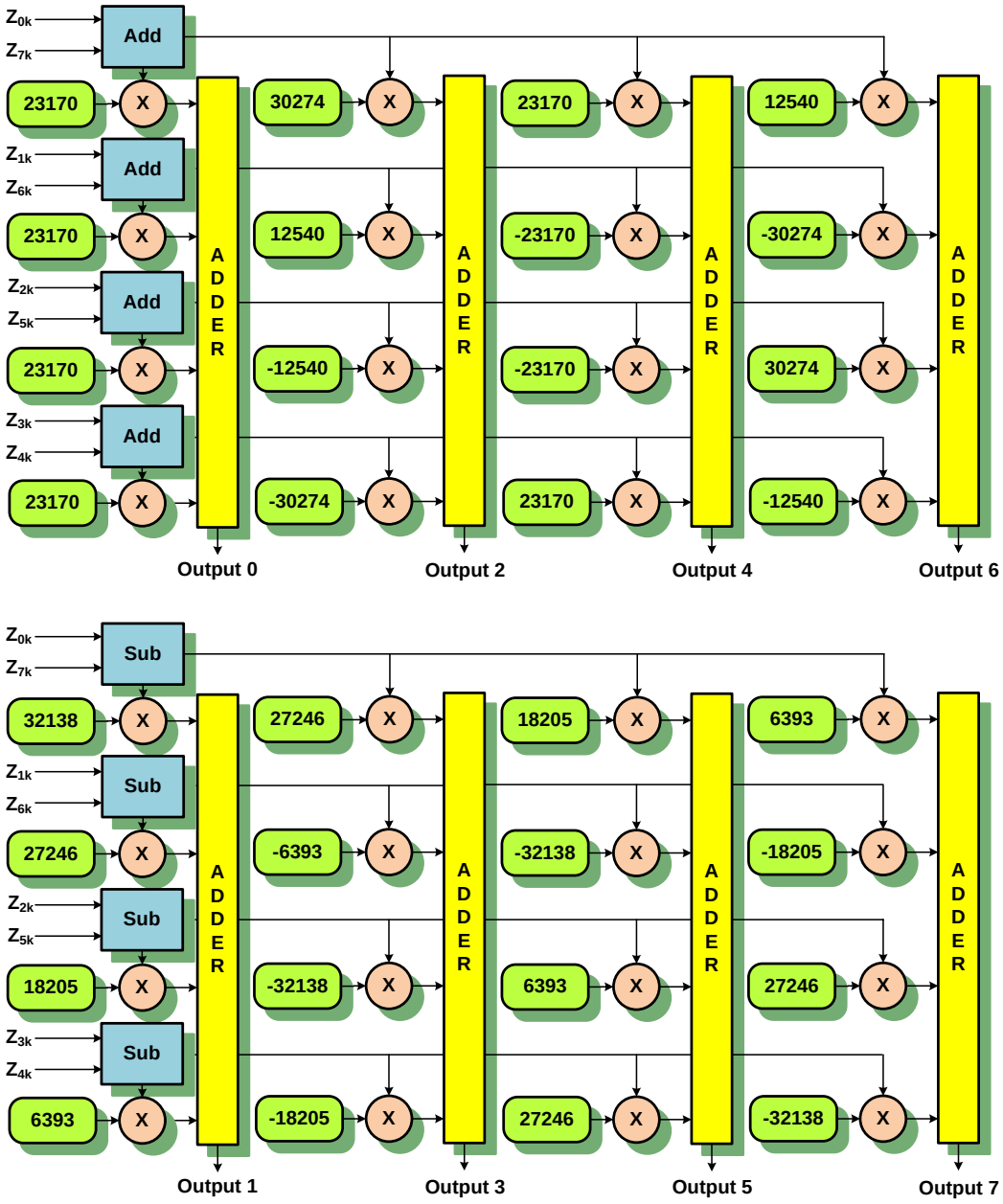


ภาพประกอบ 3-4 แสดงแผนผังลักษณะการประมวลผลของวงจร 1D-DCT ขั้นตอนที่ 2

จากภาพประกอบ 3-4 เป็นลักษณะการออกแบบการประมวลผลของวงจร 1D-DCT ขั้นตอนที่ 2 เป็นการคูณกันระหว่างเมตริกค่าคงที่ C กับแถวของเมตริก Z จำนวน 1 แถวในแนวตั้ง ซึ่งเป็นค่าผลลัพธ์ที่ได้จากการประมวลผลวงจร 1D-DCT ขั้นตอนที่ 1 วิธีการนี้จะเป็นการคำนวณสมการ 2D-DCT โดยพิจารณาข้อมูลในลักษณะแนวตั้งของเมตริกขนาด 8x8 ภายในวงจรจะมีลักษณะการออกแบบให้ส่งค่าคงที่ทุกๆ แถวภายในเมตริก C ในแนวนอน คูณกับ ค่าในเมตริก Z ในแนวตั้งตำแหน่งต่อตำแหน่ง จากนั้นรวมผลลัพธ์ที่ได้จากการคูณเข้าด้วยกัน

หากพิจารณาในแต่ละแถวในแนวนอนของค่าคงที่ C ที่ใช้คูณกับเมตริก Z ในแนวตั้งจะเหมือนกับค่าคงที่ในแนวตั้งของเมตริก C^T ในขั้นตอนที่ 1 เพราะเกิดมาจากการสลับประจำตำแหน่งในแนวนอนกับแนวตั้งหรือที่เรียกว่า การทรานสโพส (Transpose) ซึ่งสามารถที่จะใช้วิธีการจับคู่ของค่าคงที่แต่ละแถวเป็น 4 คู่ ดังเช่นวิธีการออกแบบวงจร 1D-DCT ขั้นตอนที่ 1 เพื่อที่จะนำไปคูณกับทุกๆ แถวในแนวตั้งของเมตริก Z

ดังนั้นวิธีการออกแบบจึงมีลักษณะการประมวลผลเหมือนกับขั้นตอนที่ 1 แต่มีข้อที่แตกต่างกันคือมุมมองลักษณะการพิจารณาข้อมูลในขั้นตอนที่ 1 จะพิจารณาข้อมูลรับเข้ามาในแนวนอนแต่ในขั้นตอนที่ 2 จะพิจารณาข้อมูลในแนวตั้งซึ่งเป็นการพิจารณาข้อมูลแบบ 2 มิติ ตามสมการ 2D-DCT จากภาพประกอบ 3-4 ในช่วงของสัญญาณนาฬิกาแรกของการประมวลผล เป็นการประมวลผลโดยรับค่าข้อมูลจากขั้นตอนที่ 1 จำนวน 1 แถวในที่นี้คือแถวหมายเลข 0 ในแนวตั้งของเมตริก Z เพื่อนำไปคูณกับทุกๆ แถวในแนวนอนของเมตริก C ซึ่งผลลัพธ์จากการคูณกันสำหรับแถวบนสุดต่ำแหน่ง หมายเลข 0 ในแนวนอนของเมตริกค่าคงที่ C จะได้ผลลัพธ์ในตำแหน่ง (0,0) และสำหรับแถวที่ตำแหน่ง หมายเลข 1 ในแนวนอนของเมตริกค่าคงที่ C คูณกับเมตริก Z ในแถวแนวตั้งที่รับเข้ามาจะได้ผลลัพธ์ในตำแหน่ง (1,0) ครบทั้ง 8 แถว จะได้ผลลัพธ์ของการประมวลผลสมการ 2D-DCT จำนวน 8 ผลลัพธ์ ในตำแหน่ง (0,0) , (1,0) , ….. , (7,0) โดยแผนภาพการประมวลผลในวงจร 1D-DCT ขั้นตอนที่ 2 แสดงดังภาพประกอบ 3-5



ภาพประกอบ 3-5 แสดงแผนผังวงจร 1D-DCT ขั้นตอนที่ 2 ของวงจร 2D-DCT

จากภาพประกอบ 3-5 แสดงแผนผังการออกแบบวงจร 1D-DCT ขั้นตอนที่ 2 ของการประมวลผล 2D-DCT วงจรออกแบบตามสมการการคูณกันของเมตริกค่าคงที่ C กับ แนวในแนวตั้งของเมตริก Z สมการประมวลผลคล้ายกับสมการวงจร 1D-DCT ขั้นตอนที่ 1 นั่นคือ มีการจับคู่กันของข้อมูลในที่นี้คือ แถวในแนวตั้งของเมตริก Z การจับคู่จะขึ้นอยู่กับค่าคงที่ในแนวนอนของเมตริก C ซึ่งสามารถแบ่งการจับคู่ได้ 4 กลุ่มที่มีค่าคงที่เหมือนกัน เช่นเดียวกับวงจร 1D-DCT ในขั้นตอนที่ 1 คือตำแหน่งที่ 0 กับ 7 , 1 กับ 6 , 2 กับ 5 และ 3 กับ 4

ภายในวงจรมีการแบ่งการทำงานออกเป็น 2 กลุ่ม ตามหลักทฤษฎีการประมวลผลขนานแบบผีเสื้อ นั่นคือ กลุ่มที่ 1 สำหรับแถวของเมตริกค่าคงที่ C ในแนวนอนที่เป็นแถวเลขประจำตำแหน่งเป็นเลขคู่จะให้กลุ่มของเมตริก Z แต่ละคู่บวกกันก่อนจะนำไปคูณกับค่าคงที่ประจำกลุ่ม และกลุ่มที่ 2 สำหรับในแถวของเมตริกค่าคงที่ C ในแนวนอนที่เป็นแถวเลขประจำตำแหน่งเป็นเลขคี่จะให้กลุ่มของเมตริก Z แต่ละคู่ลบกันก่อนจะนำไปคูณกับค่าคงที่ประจำกลุ่ม จากนั้นในขั้นตอนสุดท้ายจึงนำกลุ่มของการคูณทั้ง 4 มาบวกกันได้ผลลัพธ์จำนวน 8 ผลลัพธ์ โดยแถวในแนวนอนของเมตริกค่าคงที่ C ที่ไปคูณกับเมตริก Z ในแนวตั้งจะได้ผลลัพธ์ในตำแหน่งขึ้นอยู่กับค่าประจำตำแหน่งแถวของเมตริกค่าคงที่ C และเมตริก Z

สำหรับวงจร 1D-DCT ขั้นตอนที่ 2 รับข้อมูลผลลัพธ์ที่ส่งมาจากวงจร 1D-DCT ขั้นตอนที่ 1 จำนวน 8 ผลลัพธ์ในที่นี้คือเมตริก Z ในแนวตั้งจำนวน 1 แถว รับมาขนาด 11 บิต ส่งเข้าไปคูณกับค่าคงที่เมตริก C ที่มีข้อมูลแต่ละตัวขนาด 16 บิต ดังนั้นวงจรภายในจึงใช้การประมวลผล 27 บิต และผลลัพธ์ได้ส่งบิตที่ 17 ถึง 27 ซึ่งมีขนาด 11 บิต แปลงเป็นเลขขนาด 16 บิต ส่งเป็นข้อมูลผลลัพธ์การคำนวนวงจร 2D-DCT การออกแบบด้วยภาษา VHDL จะประกาศให้ผลลัพธ์ที่ได้จากการประมวลผลในส่วนต่างๆ ภายในวงจร 1D-DCT อยู่ในรูปของเส้นสัญญาณดิจิตอล ที่ทำการเชื่อมต่อทุกๆส่วนเข้าด้วยกันโดยไม่มีการพักข้อมูล ซึ่งปริมาณผลลัพธ์ที่ได้ใน 1 ช่วงสัญญาณจะเท่ากับ 8 ผลลัพธ์ ดังนั้นเมตริกขนาด 8x8 จำนวน 1 เมตริก จะสามารถประมวลผลได้เสร็จใช้สัญญาณนาฬิกา 8 ช่วงสัญญาณนาฬิกา

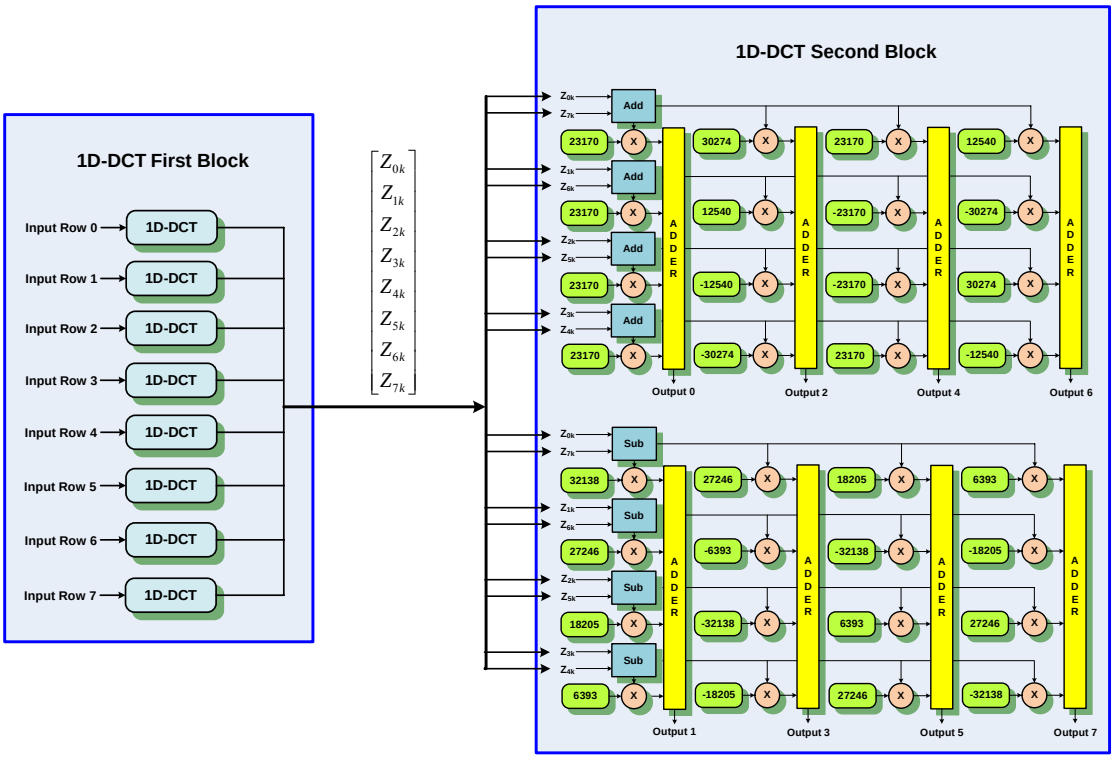
3.3 การรวมวงจร 1D-DCT ทั้ง 2 ขั้นตอนเป็นวงจร 2D-DCT

จากการออกแบบวงจร 1D-DCT ที่กล่าวไว้ข้างต้นทั้ง 2 ขั้นตอน สามารถสรุปได้ว่าทั้ง 2 ขั้นตอนมีการออกแบบที่คล้ายกันนั่นคือ ออกแบบให้ประมวลผลสมการ 1D-DCT ที่มีวิธีการเหมือนกันแต่ต่างกันที่การพิจารณาลักษณะข้อมูลที่แตกต่างกัน ซึ่งในขั้นตอนที่ 1 นั้นพิจารณาข้อมูลในแนวนอน แต่ในขั้นตอนที่ 2 พิจารณาข้อมูลในแนวตั้ง วงจรภายในของทั้ง 2 ขั้นตอน ออกแบบให้มีการประมวลผลข้อมูลขนาด 27 บิต เนื่องจากกรณีที่ทำให้การคูณกันของเมตริกจะทำให้มีค่าของตัวเลขสูงสุดคือ การคูณกันของแถวหมายเลข 0 ในแนวตั้งของเมตริก C^T กับ เมตริกข้อมูล 1 แถวในแนวนอน ภายในวงจร 1D-DCT ขั้นตอนที่ 1 และในแถวหมายเลข 0 ในแนวนอนของเมตริก C กับเมตริกข้อมูล 1 แถวในแนวตั้ง ภายในวงจร 1D-DCT ขั้นตอนที่ 2 ซึ่งทั้ง 2 กรณีที่กล่าวนั้นเป็นการใช้เลขค่าคงที่ 23170 คูณกับข้อมูล 1 แถว ซึ่งมี 8 ตำแหน่ง จากนั้นนำมารวมกัน โดยแต่ละพิกเซลของข้อมูลจะมีค่าสูงสุดคือ 255 นั่นหมายความว่าค่าสูงสุดที่ได้คือ (255 x 23170) x 8 = 47,266,800

เมื่อพิจารณาข้อมูลเป็นเลขฐาน 2 จะมีค่าเท่ากับ 10110100010011101111110000 ซึ่งมีขนาด 26 บิต และรวมกับบิตแสดงเครื่องหมายบวกหรือลบ (Bit Sign Number) รวมเป็น 27 บิต ข้อมูลที่ได้จะรวมกับขนาดของเลขทศนิยม ดังนั้นเพื่อง่ายต่อการคำนวณและประหยัดทรัพยากรในการออกแบบวงจรจึงลดข้อมูลให้อยู่ในรูปของจำนวนเต็มโดยการเลื่อนบิตไปทางขวา (Shift Right) จำนวน 16 บิต ทำให้เหลือข้อมูล 11 บิตเพื่อส่งไปคำนวณในขั้นตอนต่อไป ดังนั้นข้อมูลที่ส่งจากวงจร 1D-DCT ขั้นตอนที่ 1 ไปยังวงจร 1D-DCT ขั้นตอนที่ 2 จึงมีขนาด 11 บิต

การออกแบบวงจร 1D-DCT ขั้นตอนที่ 1 ใน 1 ช่วงเวลาได้ผลลัพธ์ 1 แถวในแนวตั้ง และได้ทำการเชื่อมต่อสัญญาณของผลลัพธ์ที่ได้ไปยังวงจร 1D-DCT ขั้นตอนที่ 2 เพื่อใช้เป็นข้อมูลนำไปประมวลผลโดยไม่ต้องพักข้อมูล เพราะจำนวนผลลัพธ์ที่ได้จากวงจร 1D-DCT ขั้นตอนที่ 1 เพียงพอที่จะให้วงจร 1D-DCT ขั้นตอนที่ 2 ใช้เพื่อประมวลผล ซึ่งในวงจร 1D-DCT ขั้นตอนที่ 2 รับข้อมูลจากขั้นตอนที่ 1 นำมาประมวลผลได้ผลลัพธ์ 1 แถวในแนวตั้งเสร็จใน 1 ช่วงเวลา ดังนั้นจึงสังเกตได้ว่าการคำนวณสมการ 2D-DCT เมื่อรับข้อมูลเมตริกที่ต้องการประมวลผลเข้าสู่วงจร 1D-DCT ขั้นตอนที่ 1 และผลลัพธ์ที่ได้มีการเชื่อมต่อไปยังวงจร 1D-DCT ขั้นตอนที่ 2 ทันที ซึ่งทั้ง 2 ขั้นตอนของวงจร 1D-DCT สามารถประมวลผลเสร็จใน 1 ช่วงสัญญาณนาฬิกา จึงส่งผลให้ วงจร 2D-DCT ที่ได้ทำการเชื่อมต่อวงจร 1D-DCT ทั้ง 2 ขั้นตอนสามารถประมวลผลได้ผลลัพธ์ 1 แถวในแนวตั้งของเมตริกผลลัพธ์ขนาด 8x8 และใช้สัญญาณนาฬิกา 8 ช่วงเวลาเพื่อประมวลผลข้อมูลเมตริกขนาด 8x8 ให้เสร็จ 1 เมตริก

จากการออกแบบวงจร 1D-DCT ทั้ง 2 ขั้นตอนทำให้ไม่จำเป็นต้องสร้างวงจรบัฟเฟอร์เพื่อรอเก็บข้อมูลที่ได้จากวงจรขั้นตอนที่ 1 ก่อนส่งต่อไปประมวลผลในขั้นตอนที่ 2 จึงเป็นการลดเวลาในการประมวลผลที่สูญเสียไปในช่วงของการพักข้อมูลที่วงจรบัฟเฟอร์ แผนผังการออกแบบรวมวงจร 1D-DCT ทั้ง 2 ขั้นตอน เข้าเป็นวงจรการคำนวณสมการ 2D-DCT แสดงดังภาพประกอบที่ 3-6



ภาพประกอบ 3-6 แผนผังการออกแบบวงจรการคำนวณสมการ 2D-DCT

3.4 บทสรุป

จากการออกแบบวงจร 2D-DCT ที่กล่าวมาข้างต้น เป็นการใช้เทคนิคการประมวลผลแบบขนานเข้ามาช่วยเพิ่มประสิทธิภาพในด้านของความเร็วในการประมวลผล การออกแบบโดยรับข้อมูลขนาด 16 บิต จำนวน 1 เมตริก ขนาด 8x8 เข้ามาประมวลผล ใน 1 ช่วงเวลาได้ผลลัพธ์ขนาด 1 แถวในแนวตั้ง ของเมตริกขนาด 8x8 ดังนั้นผลลัพธ์ของการคำนวนสมการ 2D-DCT นั่นคือข้อมูลที่เป็นเมตริกขนาด 8x8 ใช้ระยะเวลาการคำนวณ 8 ช่วงเวลา โดยออกแบบบนสถาปัตยกรรม FPGA ซึ่งบรรยายโดยใช้ภาษา VHDL

ในบทถัดไปจะกล่าวถึงการทดสอบการทำงานของวงจรในด้านของความถูกต้องในการประมวลผล ความเร็วสูงสุดที่สามารถใช้ จำนวนทรัพยากรที่ใช้บนสถาปัตยกรรม FPGA และรวมถึงพลังงานที่สูญเสียไปในการประมวลผล

บทที่ 4

ประสิทธิภาพของการออกแบบวงจร 2D-DCT ประมวลผลแบบขนาน

ในบทนี้จะเป็นการทดสอบประสิทธิภาพของวงจร 2D-DCT ซึ่งออกแบบให้ประมวลผลแบบขนานและลดจำนวนวงจรที่ใช้ ดังที่กล่าวไว้ในบทที่ 3 โดยออกแบบบนเทคโนโลยี FPGA ตระกูล Virtex 4 ชิปเบอร์ XC4VFX12 ด้วยภาษา VHDL การทดสอบแบ่งออกเป็น 5 ประเด็นหลักคือ การทดสอบหาความถูกต้องของการประมวลผลด้วยโปรแกรม Matlab 7.8 การทดสอบหาความถูกต้องของวงจรการประมวลผลบนสถาปัตยกรรม FPGA การทดสอบหาความสามารถด้านความเร็วสูงสุดของการประมวลผล การทดสอบหาจำนวนทรัพยากรที่ใช้บนสถาปัตยกรรม FPGA และท้ายที่สุดคือการทดสอบหาพลังงานที่สูญเสียในการประมวลผล

4.1 การทดสอบหาความถูกต้องของการประมวลผลวงจร 2D-DCT

การทดสอบหาความถูกต้องของการประมวลผลวงจร 2D-DCT เป็นการตรวจสอบความคลาดเคลื่อนของผลลัพธ์ว่าตรงตามสมการ 2D-DCT หรือไม่ เพราะในการคำนวณสมการ 2D-DCT จะมีเศษของการคำนวณอยู่ในเลขจำนวนทศนิยม แต่เพื่อให้ง่ายและเหมาะสมกับอุปกรณ์ FPGA การออกแบบวงจรใช้การประมวลผลแบบจำนวนเต็มดังที่กล่าวไว้ในบทที่ 3 ซึ่งทำให้เกิดความผิดพลาดของข้อมูลขึ้น ดังนั้นจึงจำเป็นต้องทดสอบประสิทธิภาพความแม่นยำในการประมวลผลของวงจร 2D-DCT เพื่อสามารถบอกถึงความเหมาะสมที่จะนำไปประยุกต์ใช้เป็นส่วนหนึ่งของการออกแบบวงจรการบีบอัดรูปภาพได้

ในการทดสอบประสิทธิภาพความถูกต้องของขั้นตอนวิธีการคำนวณสมการ 2D-DCT จะใช้โปรแกรม Matlab 7.8 ในการทดสอบขั้นตอนวิธีการประมวลผล เนื่องจากโปรแกรม Matlab 7.8 เป็นโปรแกรมที่มีฟังก์ชั่นสามารถคำนวณสมการทางคณิตศาสตร์ได้ง่าย ใช้เขียนโปรแกรมเพื่อออกแบบโปรแกรมประยุกต์(Application) และการจำลองการทำงาน(Simulation) ต่างๆได้ เหมาะกับใช้กับงานการประมวลผลภาพ(Image Processing) โดยขั้นตอนของการใช้โปรแกรม Matlab 7.8 ในการทดสอบได้แบ่งออกเป็น 2 ส่วนหลักๆ คือ ส่วนของการสร้างโปรแกรมจำลองการทำงานของการบีบอัดรูปภาพแบบ JPEG โดยการประมวลผล 2D-DCT ใช้ขั้นตอนการประมวลผลดังที่กล่าวในบทที่ 3 และอีกส่วนในการทดสอบเป็นการเปรียบเทียบภาพต้นแบบกับ

ภาพที่ผ่านการบีบอัดเพื่อหาค่าความคลาดเคลื่อน รายละเอียดในการทดสอบจะกล่าวเรียงตามลำดับดังนี้

4.1.1 การออกแบบโปรแกรมประยุกต์จำลองการทำงานของวงจร 2D-DCT

การออกแบบโปรแกรมจำลองการทำงานเพื่อใช้ในการทดสอบขั้นตอนวิธีการประมวลผลสมการ 2D-DCT จะทำการสร้างเมตริก C และ C^T ซึ่งเป็นเมตริกของค่าคงที่ที่จะใช้ในการประมวลผล 2D-DCT จากนั้นป้อนรูปภาพที่ต้องการทดสอบเข้าไปเก็บไว้ในรูปของเมตริกเพื่อใช้เป็นข้อมูลสำหรับทดสอบ ซึ่งชนิดของภาพที่ป้อนเข้าไปใช้ภาพแบบ Portable Gray Map(PGM) ซึ่งเป็นรูปภาพแบบโทนสีเทา(Grayscale Image Format) เป็นภาพที่มีค่าประจำพิกเซลอยู่ในช่วง 0 ถึง 255 ขนาดภาพที่ใช้ 512x512 พิกเซล ชนิดของภาพที่นำมาใช้ในการทดสอบเป็นรูปภาพที่ได้รับการยอมรับให้สามารถใช้เป็นมาตรฐานในการทดสอบการบีบอัดรูปภาพประกอบไปด้วย

- Baboon
- Barbara
- Boat
- Bridge
- Lena
- Peppers
- Splash
- Tiffany

การทดสอบจะออกแบบโปรแกรมทดสอบโดยวิธีการประมวลผลใช้ขั้นตอนและวิธีการเช่นเดียวกับการออกแบบวงจร 2D-DCT โดยเริ่มจากการแบ่งข้อมูลของภาพต้นแบบออกเป็นส่วนย่อยๆ ที่มีขนาด 8x8 พิกเซล ตามมาตรฐานการบีบอัดรูปภาพแบบ JPEG จากนั้นนำข้อมูลแต่ละส่วนที่ได้เข้าสู่การประมวลผลสมการ 2D-DCT ขั้นตอนที่ 1 นำเมตริกที่แบ่งออกเป็นส่วนๆ ใช้เป็นเมตริกตัวตั้งเพื่อนำไปคูณกับเมตริก C^T จากนั้นทำการชิปขวา 16 บิต นั้นคือการใช้ข้อมูลของสมาชิกแต่ละตัวที่ผ่านการคูณแล้วหารด้วย 65535 แล้วแปลงข้อมูลให้เป็นแบบจำนวนเต็ม เป็นการสิ้นสุดการประมวลผลสมการ 2D-DCT ขั้นตอนที่ 1 ในขั้นตอนถัดไปใช้ข้อมูลที่ทำการแปลงเป็นจำนวนเต็มแล้วเป็นเมตริกตัวคูณนำไปคูณกับค่าคงที่ C จากนั้นหารด้วย 65535 แล้วแปลงข้อมูลให้เป็นแบบจำนวนเต็ม เป็นการสิ้นสุดการประมวลผลสมการ 2D-DCT ขั้นตอนที่ 2 และได้ข้อมูลที่เป็นผลลัพธ์ของการคำนวณสมการ 2D-DCT หลังจากนั้นขั้นตอนถัดไปนำผลลัพธ์ของการประมวลผล 2D-DCT หารด้วยเมตริกค่าคงที่ควอนไตซ์ (Quantization Matrix) ในขั้นตอนนี้คือ

ขั้นตอนการทำควอนไตเซชั่นตามรูปแบบการบีบอัดรูปภาพแบบ JPEG ในขั้นตอนสุดท้ายของการบีบอัดรูปภาพเป็นการเข้ารหัสซึ่งข้อมูลที่เข้ารหัสแล้วนำมาถอดรหัสไม่มีผลต่อการสูญเสียข้อมูลจึงไม่ขอนำมาใช้ในการทดสอบความถูกต้องของการประมวลผล

หลังจากทำการใช้โปรแกรมจำลองการบีบอัดข้อมูลรูปภาพแบบ JPEG จัดการประมวลผลกับภาพต้นแบบแล้ว ในขั้นตอนการนำข้อมูลกลับมาใช้เพื่อให้เป็นรูปภาพตามที่ต้องสามารถทำได้โดยออกแบบโปรแกรมนำข้อมูลที่ผ่านกระบวนการบีบอัดมาประมวลผลย้อนกลับตามหลักการถอดรหัสข้อมูลรูปภาพแบบ JPEG นั่นคือ เริ่มจากการนำข้อมูลรูปภาพที่บีบอัดแล้วมาแบ่งออกเป็นส่วนย่อยๆ เป็นเมตริกขนาด 8x8 จากนั้นนำเมตริกแต่ละส่วนมาคูณกับเมตริกความไตซ์ขั้นตอนถัดไปทำการประมวลผลด้วยสมการย้อนกลับของสมการ 2D-DCT นั่นคือสมการ 2-Dimention Invert Discrete Cosine Transform (2D-IDCT) เพื่อให้กลับไปสู่ข้อมูลที่สามารถแสดงเป็นภาพตามที่ต้องการ

การคำนวณ 2D-IDCT โปรแกรม Matlab 7.8 มีฟังก์ชั่นที่สามารถนำมาใช้ได้โดยในการทำวิจัยได้ทดสอบการทำงานของฟังก์ชั่นดังกล่าวโดย ใช้ฟังก์ชั่นการคำนวณ 2D-DCT ที่มีอยู่ภายในโปแกรม Matlab 7.8 แปลงตัวอย่างข้อมูลในเมตริกขนาด 8x8 แล้วใช้ฟังก์ชั่น 2D-IDCT แปลงข้อมูลย้อนกลับ ผลการทดลอง คือ เมื่อนำข้อมูลต้นแบบกับข้อมูลที่ผ่านการคำนวณด้วยฟังก์ชั่น 2D-DCT และคำนวณย้อนกลับด้วยฟังก์ชั่น 2D-IDCT มาเปรียบเทียบกับต้นแบบ ปรากฏว่าได้ผลลัพธ์ที่เหมือนกันที่เมตริกต้นแบบและเมตริกที่ผ่านการประมวลผล จึงสรุปได้ว่าฟังก์ชั่น 2D-IDCT สามารถนำมาใช้ในการทดสอบได้ ฟังก์ชั่น 2D-IDCT ที่ใช้ในการทดสอบมีลักษณะดังต่อไปนี้

idct2(A)

กำหนดให้ A คือ เมตริกที่ต้องการประมวลผลด้วยสมการ 2D-IDCT

หลังจากทำการแปลงข้อมูลที่ผ่านการบีบอัดกลับมาเป็นข้อมูลภาพแล้ว ในขั้นตอนถัดไปของการทดสอบ คือการนำภาพที่แปลงกลับมาเปรียบเทียบกับภาพต้นแบบ เพื่อหาค่าความคลาดเคลื่อนของภาพทั้งสองและวิเคราะห์เพื่อสรุปว่าสามารถใช้เป็นส่วนหนึ่งของการบีบอัดรูปภาพแบบ JPEG ได้หรือไม่ โดยที่ไม่สูญเสียคุณภาพของรูปภาพมากเกินไป ซึ่งในหัวข้อถัดไปจะกล่าวถึงวิธีการของการเปรียบเทียบรูปภาพเพื่อหาความคลาดเคลื่อนของภาพสองภาพ

4.1.2 การเปรียบเทียบภาพต้นแบบกับภาพที่ผ่านการบีบอัดรูปภาพ

การเปรียบเทียบรูปภาพสองภาพเพื่อหาความแตกต่างที่สามารถบอกถึงความถูกต้องแม่นยำที่ยอมรับได้สามารถใช้สมการการทดสอบที่เรียกว่า Peak Signal to Noise Ratio (PSNR) ซึ่งเป็นค่ามาตรฐานที่บ่งบอกถึงคุณภาพที่เปลี่ยนไประหว่างรูปภาพสองภาพ ใช้ในการเปรียบเทียบที่สภาวะต่างๆกัน ซึ่งรายละเอียดของสมการที่ใช้ในการคำนวณหาค่า PSNR [2] มีดังนี้

$$PSNR = 20\log_{10}\left(\frac{b}{RMSE}\right) \qquad (1)$$

กำหนดให้ b คือ ค่าสูงสุดที่เป็นไปได้ของพิกเซลในภาพ

RMSE (Root Mean Square Error) คือ สแควรูสของค่าเฉลี่ยความผิดพลาด รายละเอียดของสมการการคำนวณดังนี้

$$RMSE = \sqrt{MSE} \qquad (2)$$

MSE คือ ค่าเฉลี่ยของความผิดพลาดในการประมวลผล (Mean Square Error) รายละเอียดของสมการการคำนวณดังนี้

$$MSE = \frac{1}{MN}\sum_{y=1}^{M}\sum_{x=1}^{N}\left[I(x,y) - I'(x,y)\right]^2 \qquad (3)$$

กำหนดให้ I คือ ค่าประจำพิกเซลของรูปภาพต้นฉบับ

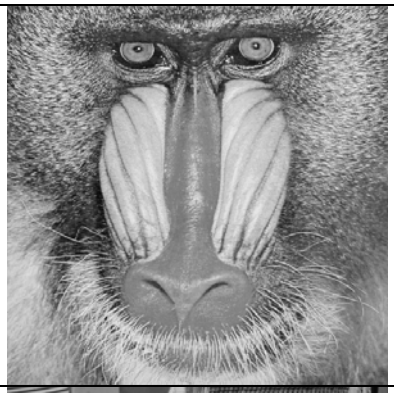
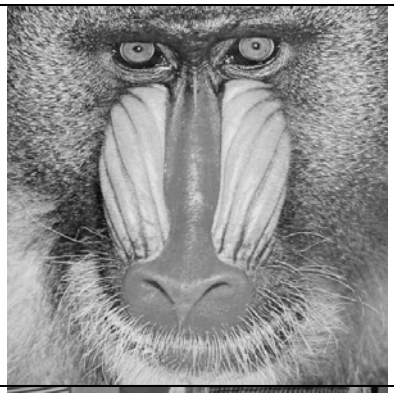
I' คือ ค่าประจำพิกเซลของรูปภาพที่ผ่านกระบวนการ 2D-DCT

M คือ จำนวนพิกเซลด้านความสูงของรูปภาพ

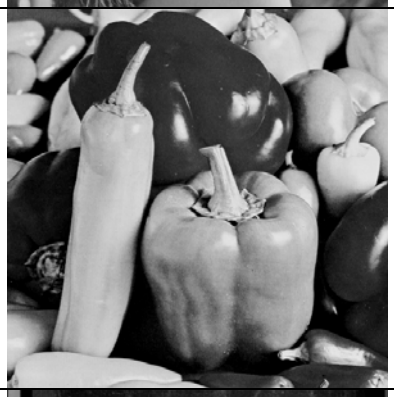
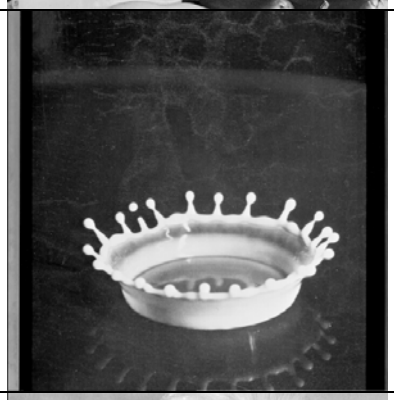
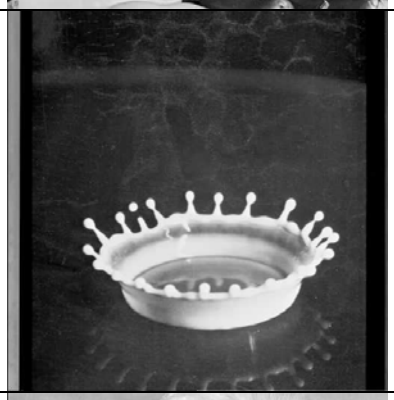
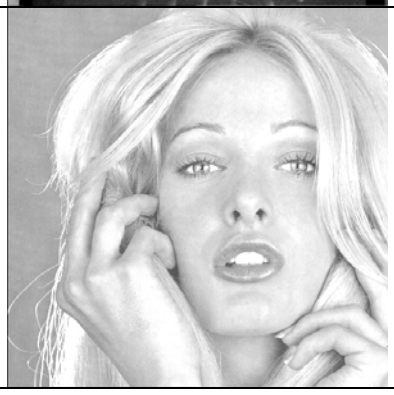
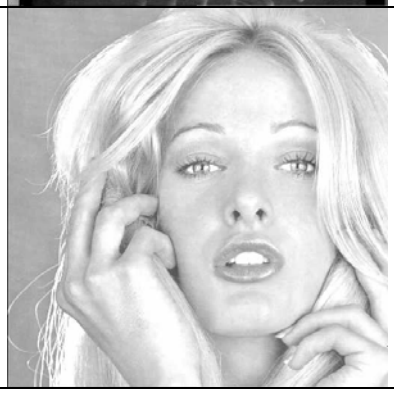
N คือ จำนวนพิกเซลด้านความกว้างของรูปภาพ

จากผลลัพธ์การคำนวณหาค่า PSNR สามารถวิเคราะห์เพื่อหาค่าความแตกต่างได้ โดยกรณีที่ค่า PSNR สูงกว่า 30 dB แสดงว่ารูปภาพที่นำมาเปรียบเทียบกับต้นแบบมีความแตกต่างที่สามารถยอมรับได้ซึ่งหากยิ่งมีค่าสูงภาพทั้งสองภาพจะมีค่าความใกล้เคียงมาก ในทางกลับกันถ้าหากค่า PSNR มีค่าต่ำกว่า 30 dB หรือเข้าใกล้ศูนย์มาก ความแตกต่างของภาพที่นำมาเปรียบเทียบก็จะมีค่าแตกต่างกับต้นแบบมากไม่สามารถยอมรับได้

ตารางที่ 4-1 แสดงตารางผลการบีบอัดรูปภาพและการหาค่าความคลาดเคลื่อน

ชื่อภาพ	ภาพต้นแบบ	ภาพที่ผ่านการบีบอัด	PSNR (dB)
Baboon			32.2765
Barbara			36.2114
Boat			37.6534
Bridge			33.4879

ตารางที่ 4-2 แสดงตารางผลการบีบอัดรูปภาพและการหาค่าความคลาดเคลื่อน (ต่อ)

ชื่อภาพ	ภาพต้นแบบ	ภาพที่ผ่านการบีบอัด	PSNR (dB)
Lena			38.5893
Peppers			36.6812
Splash			38.9172
Tiffany			36.4874

จากตารางที่ 4-1 และ 4-2 แสดงผลการทดลองการบีบอัดรูปภาพและการหาค่าความคลาดเลื่อนของภาพที่ทำการบีบอัดเปรียบเทียบกับภาพต้นแบบ สามารถสรุปได้ว่าหากสังเกตด้วยตาของมนุษย์ระหว่างภาพต้นแบบกับภาพที่ผ่านการบีบอัด สังเกตได้ว่าแทบจะไม่สามารถบอกได้ว่าภาพไหนเป็นภาพต้นแบบหรือภาพที่ผ่านการบีบอัดและมีความคลาดเคลื่อนของข้อมูล เมื่อพิจารณาค่า PSNR จะสังเกตว่าจะได้ผลลัพธ์อยู่ระหว่าง 30 – 40 dB นั่นหมายความว่า ภาพที่ผ่านการบีบอัดในการทดสอบเป็นการบีบอัดที่ยอมสูญเสียข้อมูลบางส่วนแต่คุณภาพของภาพแทบจะไม่ผิดเพี้ยนไปจากภาพต้นแบบมาก และหากพิจารณาจากค่า PSNR ของการบีบอัดรูปภาพเคลื่อนไหวที่กำหนดให้ค่ามาตรฐานอยู่ในช่วง 30 – 50 dB จึงสรุปได้ว่า หากพิจารณาด้านประสิทธิภาพการคำนวณของวงจรที่ออกแบบ มีความถูกต้องตามสมการ 2D-DCT และสามารถที่จะใช้เป็นส่วนหนึ่งของการพัฒนาวงจรการบีบอัดรูปภาพแบบ JPEG ทั้งภาพนิ่งและภาพเคลื่อนไหวได้

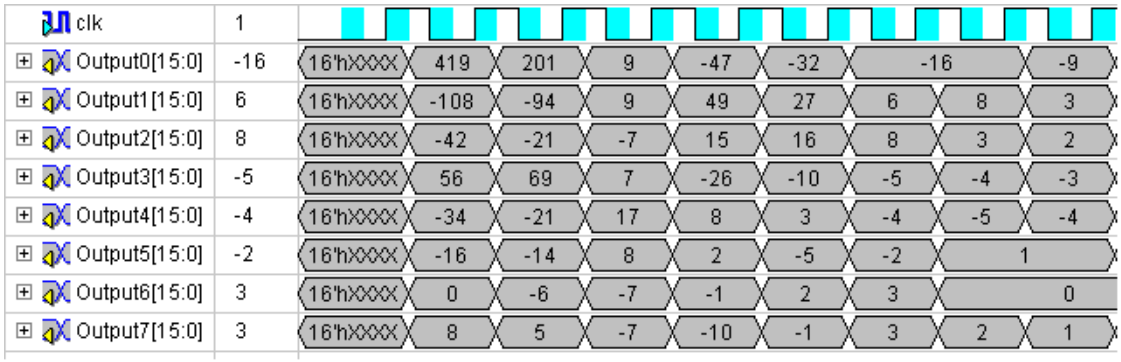
4.2 การทดสอบความถูกต้องของวงจรการประมวลผลบนสถาปัตยกรรม FPGA

เมื่อทำการทดสอบขั้นตอนการทำงานของวงจร 2D-DCT ด้วยโปรแกรมMatlab 7.8 สามารถพิสูจน์ให้เห็นว่าสามารถนำวิธีการนี้มาใช้ในการออกแบบเป็นวงจรดิจิตอลการประมวลผลสมการ 2D-DCT ได้ ดังนั้นจึงได้ทำการออกแบบวงจรบนสถาปัตยกรรม FPGA ด้วยโปรแกรม Xilinx 8.1 บรรยายด้วยภาษา VHDL

สำหรับ Xilinx 8.1 ภายในโปรแกรมนี้มีโปรแกรมจำลองการทำงานของวงจรที่ออกแบบคือ Test Bench Waveform เป็นโปรแกรมที่ใช้ตรวจสอบการทำงานของวงจรแสดงเส้นสัญญาณต่างๆ ที่อยู่ภายในวงจรที่มีการประมวลผล สามารถกำหนดการป้อนข้อมูลเข้าสู่วงจรและดูผลลัพธ์ของการประมวลผลได้ วงจร 2D-DCT ที่ออกแบบจะรับข้อมูลขนาด 16 บิต จำนวน 64 ตัว นั่นคือเมตริกขนาด 8x8 เป็นข้อมูลที่ป้อนเข้าสู่วงจร ดังนั้นจึงได้ทดลองป้อนข้อมูลเป็นเมตริกขนาด 8x8 ใช้เป็นข้อมูลที่จะนำไปประมวลผลดังเมตริกต่อไปนี้

$$\begin{bmatrix} 58 & 45 & 29 & 27 & 24 & 19 & 17 & 20 \\ 62 & 52 & 42 & 41 & 38 & 30 & 22 & 18 \\ 48 & 47 & 49 & 44 & 40 & 36 & 31 & 25 \\ 59 & 78 & 49 & 32 & 28 & 31 & 31 & 31 \\ 98 & 138 & 116 & 78 & 39 & 24 & 25 & 27 \\ 115 & 160 & 143 & 97 & 48 & 27 & 24 & 21 \\ 99 & 137 & 127 & 84 & 42 & 25 & 24 & 20 \\ 74 & 95 & 82 & 67 & 40 & 25 & 25 & 19 \end{bmatrix}$$

เมื่อทำการรันโปรแกรมการจำลองการทำงานโปรแกรมจะแสดงผลลัพธ์การทำงานเป็นข้อมูลที่ส่งออกมาจำนวน 8 ผลลัพธ์ต่อ 1 รอบสัญญาณนาฬิกา ดังภาพประกอบ 4-1



ภาพประกอบ 4-1 ผลการประมวลผลของวงจร 2D-DCT ด้วยโปรแกรมจำลองการทำงาน

จากภาพประกอบ 4-1 แสดงผลการประมวลผลของวงจร 2D-DCT ที่ได้ออกแบบโดยทุกๆ 1 รอบสัญญาณนาฬิกาจะได้ค่าผลลัพธ์จำนวน 8 ผลลัพธ์ ซึ่งเป็นผลลัพธ์ของวงจร 2D-DCT ที่แสดงอยู่ในรูปของเมตริกขนาด 8x8 จำนวน 1 แถวในแนวตั้ง ลักษณะข้อมูลที่ใช้ประมวลผลตามมาตรฐานของ JPEG จะถูกแบ่งออกเป็นเมตริกขนาด 8x8 ดังนั้นเมื่อสัญญาณนาฬิกาครบ 8 รอบ จะได้เมตริกผลลัพธ์ของสมการ 2D-DCT ซึ่งจากภาพประกอบ 4-1 สามารถสรุปเป็นเมตริกผลลัพธ์ของสมการได้ดังต่อไปนี้

$$\begin{bmatrix} 419 & 201 & 9 & -47 & -32 & -16 & -16 & -9 \\ -108 & -94 & 9 & 49 & 27 & 6 & 8 & 3 \\ -42 & -21 & -7 & 15 & 16 & 8 & 3 & 2 \\ 56 & 69 & 7 & -26 & -10 & -5 & -4 & -3 \\ -34 & -21 & 17 & 8 & 3 & -4 & -5 & -4 \\ -16 & -14 & 8 & 2 & -5 & -2 & 1 & 1 \\ 0 & -6 & -7 & -1 & 2 & 3 & 0 & 0 \\ 8 & 5 & -7 & -10 & -1 & 3 & 2 & 1 \end{bmatrix}$$

จากเมตริกผลลัพธ์ของการประมวลผลวงจร 2D-DCT ปรากฏว่าได้ผลลัพธ์ของการประมวลผลที่ถูกต้องตามสมการ 2D-DCT ดังนั้นจึงสามารถที่จะนำไปใช้ประยุกต์ออกแบบวงจรการบีบอัดรูปภาพแบบ JPEG ได้

4.3 การทดสอบหาความสามารถด้านความเร็วสูงสุดของการประมวลผล

การทดสอบความสามารถในด้านความเร็วสูงสุดของการประมวลผล ใช้โปรแกรม Xilinx 8.1 ซึ่งเป็นโปรแกรมที่ใช้พัฒนาและออกแบบวงจรดิจิตอลให้กับ FPGA มีเครื่องมือที่สามารถทำการสังเคราะห์วงจร (Synthesis) และทำการสร้างวงจรย้อนกลับ (Backannotate) ซึ่งมีข้อมูลในส่วนของค่า delay จากการ wiring แล้ว เพื่อดูเวลาที่สูญเสียไป (Delay Time) ในการประมวลผล โปรแกรมจะคำนวณค่าสัญญาณนาฬิกาสูงสุด (Maximum Frequency) ที่วงจรสามารถใช้ได้ในการประมวลผล จากนั้นนำความถี่ที่ได้มาคิดเป็นจำนวนเท่าของงานที่ได้ผลลัพธ์ต่อหนึ่งช่วงความถี่ สำหรับวงจร 2D-DCT ได้ผลลัพธ์ 8 ผลลัพธ์ในแนวตั้งใน 1 ช่วงเวลา ผลลัพธ์ของสมการ 2D-DCT มีลักษณะเป็นเมตริกขนาด 8x8 มีสมาชิกจำนวน 64 ผลลัพธ์ ดังนั้นจึงใช้ 8 รอบสัญญาณนาฬิกาจึงจะได้ผลลัพธ์ 1 เมตริก ผลการสังเคราะห์ความเร็วของวงจร 2D-DCT มีผลลัพธ์ดังตารางที่ 4-3

ตารางที่ 4-3 แสดงผลการสังเคราะห์ความเร็วของวงจร 2D-DCT ด้วยโปรแกรม Xilinx 8.1

Maximum Frequency	287.708 MHz
Maximum Throughput	(287 x 10^6) / 8 = 35,875,000 Matrix/Sec
Latency	8 / (287 x 10^6) = 0.028 uS

จากตารางที่ 4-3 ค่าความถี่สูงสุดที่สังเคราะห์และสามารถใช้ได้กับวงจรที่ออกแบบประมาณ 287 MHz นั่นคือจะสามารถใช้สัญญาณนาฬิกาได้ 287 ล้านรอบต่อวินาที เมตริกการคำนวณสมการ 2D-DCT จำเป็นต้องใช้สัญญาณนาฬิกา 8 รอบ จึงจะได้ผลลัพธ์ เมตริก 2D-DCT ขนาด 8x8 ดังนั้นจึงสรุปได้ว่า สามารถที่จะคำนวนได้ปริมาณ 38.5 ล้านเมตริกต่อวินาที คำนวณเป็นระยะเวลาเวลาที่ใช้ในการประมวลผลมีค่าเท่ากับ 0.025 uS

หากพิจารณาอัตราการไหลของข้อมูลเพียงแค่วงจรการคำนวณ 2D-DCT เพียงขั้นตอนเดียว เปรียบเทียบกับความละเอียดต่างๆ ซึ่งเป็นภาพที่ใช้ใน IP Camera ได้ดังตารางที่ 4-4 โดยแบ่งข้อมูลที่ใช้ในการเปรียบเทียบเป็น 2 รูปแบบคือ ข้อมูลรูปภาพแบบโทนสีเทา และข้อมูลรูปภาพสี เนื่องจากข้อมูลรูปภาพสีจะมีข้อมูลมากกว่าข้อมูลรูปภาพแบบโทนสีเทา 3 เท่า ตามหลักของข้อมูลรูปภาพดิจิตอล กล่าวคือ ใน 1 พิคเซล ของข้อมูลรูปภาพสีจะประกอบไปด้วยข้อมูล 3 ชนิด คือ Y, Cb และ Cr

ตารางที่ 4-4 แสดงอัตราการไหลของข้อมูลของวงจร 2D-DCT ในช่วงเวลา 1 วินาที

ค่าความละเอียดของรูปภาพ (Resolution)	อัตราการไหลของข้อมูล (ภาพต่อวินาที)	
	ข้อมูลภาพโทนสีเทา	ข้อมูลภาพสี
800 x 600	4783	1594
1024 x 768	2919	973
1280 x 1024	1747	582
1920 x 1080	1107	369

หากพิจารณานำค่าระยะเวลาที่ใช้ในการไหลของข้อมูลแทนค่าเข้าไปในวงจรของงานวิจัยที่เกี่ยวข้องที่ได้นำเสนอในบทที่ 1 ดังตารางที่ 1-4 เสมือนเป็นการทดสอบว่าหากมีการแทนวงจร 2D-DCT ที่ได้ทำการออกแบบเข้าไปในวงจรการบีบอัดรูปภาพเพื่อหาค่าระยะเวลาที่ใช้ในการบีบอัดรูปภาพ แต่เนื่องจากเป็นข้อมูลภาพสีจึงใช้ระยะเวลาในการประมวลผลวงจร 2D-DCT เพิ่มขึ้น 3 เท่า เพราะปริมาณข้อมูลภาพสีใน 1 พิคเซล ประกอบไปด้วยตัวเลข 3 จำนวน ระยะเวลาที่ใช้จึงเท่ากับ (0.027 uS) x 3 = 0.081 uS

เมื่อทำการแทนค่าระยะเวลาการประมวลผลสมการ 2D-DCT เข้าไปในวงจรการบีบอัดรูปภาพแบบ JPEG ของ Luciano ในตารางที่ 1-4 สามารถเปรียบเทียบระยะเวลาที่ใช้ในการบีบอัดรูปภาพได้ดังตารางที่ 4-5

ตารางที่ 4-5 แสดงเปรียบเทียบระยะเวลาในการบีบอัดรูปภาพแบบ JPEG

Hardware Block	**Latency (1/F x clock cycle)**	
	Naras*	Luciano[9]
Color Space Converter	0.128 uS	0.128 uS
2D-DCT	**0.081 uS**	**4.503 uS**
Quantization	0.071 uS	0.071 uS
Zigzag Buffer	0.515 uS	0.515 uS
Entropy Coder	0.106 uS	0.106 uS
JPEG Compression	**0.901 uS**	**5.323 uS**

4.4 การทดสอบหาจำนวนทรัพยากรที่ใช้บนสถาปัตยกรรม FPGA

การทดสอบจำนวนทรัพยากรที่ใช้บนสถาปัตยกรรม FPGA สามารถใช้โปรแกรม Xilinx 8.1 สังเคราะห์วงจรเพื่อหาค่าทรัพยากรของ FPGA ที่สูญเสียไปในการออกแบบและพัฒนาวงจร 2D-DCT ซึ่งทรัพยากรที่กล่าวถึงอาธิเช่น จำนวนของแผ่นลอจิก(Slice) จำนวนฟลิบฟลอบ (Flip Flop) หน่วยความจำ(RAM) เป็นต้น ซึ่งล้วนแต่เป็นอุปกรณ์ส่วนต่างๆที่อยู่บนสถาปัตยกรรม FPGA การจะที่สังเคราะห์ทรัพยากรที่ใช้จำเป็นต้องกำหนดชนิดของอุปกรณ์ต้นแบบที่จะใช้สำหรับทดสอบ ซึ่งในการทำงานวิจัยได้กำหนดอุปกรณ์ที่ใช้ คือ FPGA ตระกูล Virtex 4 ชิปเบอร์ XC4VFX12 ซึ่งผลการสังเคราะห์เพื่อหาจำนวนทรัพยากรที่ใช้ไปสรุปได้ดังตารางที่ 4-5

ตารางที่ 4-5 ผลการสังเคราะห์จำนวนทรัพยากรที่ใช้ของวงจร 2D-DCT บนสถาปัตยกรรม FPGA

Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	1998	5472	36%
Number of Slice Flip Flops	43	10944	1%
Number of 4 input LUTs	3751	10944	34%
Number of FIFO16/RAMB16s	4	36	11%
Number of GCLKs	1	32	3%
Number of DSP48s	32	32	100%

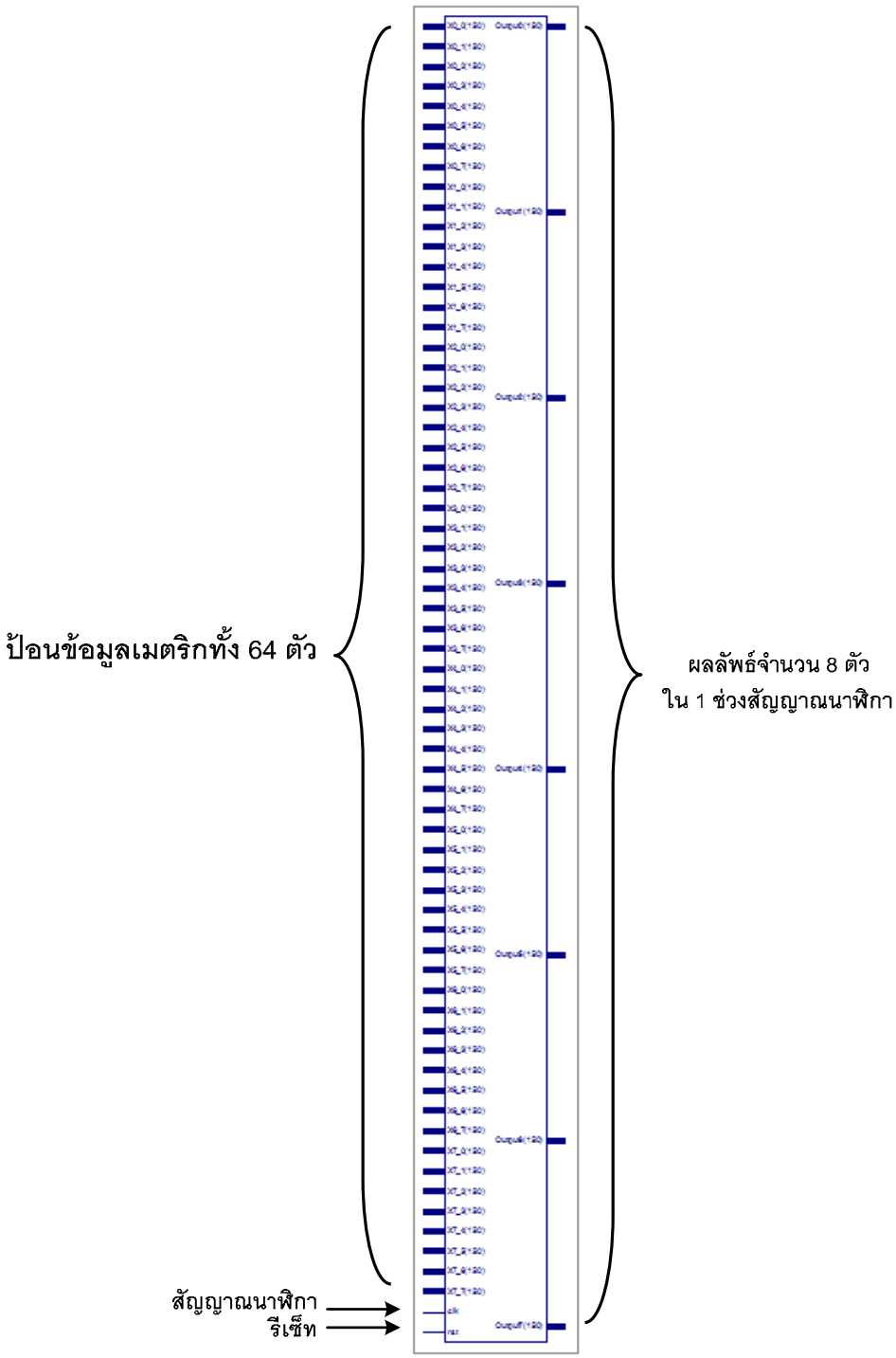
จากตารางที่ 4-5 แสดงผลการสังเคราะห์เพื่อหาจำนวนทรัพยากรที่ใช้ไปของอุปกรณ์ FPGA ซึ่งจำนวนทรัพยากรได้สรุปออกเป็นอัตราส่วนของปริมาณที่ใช้เป็นประโยชน์ต่อจำนวนทรัพยากรทั้งหมดที่มี สามารถสรุปได้ว่าอุปกรณ์ FPGA ที่ใช้ในการทดสอบมีทรัพยากรที่เพียงพอที่จะใช้ในการประมวลผล และยังมีทรัพยากรในบางส่วนที่ไม่ได้ใช้ไปในการออกแบบและพัฒนาซึ่งสามารถที่จะนำไปพัฒนาร่วมกับวงจรในขั้นตอนอื่นๆ ของการบีบอัดรูปภาพแบบ JPEG

ในการออกแบบวงจร 2D-DCT เพื่อให้ได้ความเร็วสูงนอกจากใช้ DSP48 ตามที่ได้กล่าวไว้ในหัวข้อที่ 2.6 วงจรการคำนวณทางคณิตศาสตร์ต่างๆ ได้ออกแบบโดยใช้วงจรมาโคร ซึ่งเป็นวงจรการคำนวณทางคณิตศาสตร์ที่ถูกออกแบบขึ้นโดยบริษัทผู้ผลิต นอกจากจะทำให้สามารถคำนวณได้ความเร็วสูงยังทำให้ช่วยลดจำนวนแผ่นลอจิกทรัพยากรของ FPGA อีกด้วย โดยมีรายละเอียดจำนวนการใช้วงจรมาโครในการออกแบบดัง ตารางที่ 4-6

ตารางที่ 4-6 ตารางจำนวนการใช้วงจรมาโครในการออกแบบวงจร 2D-DCT

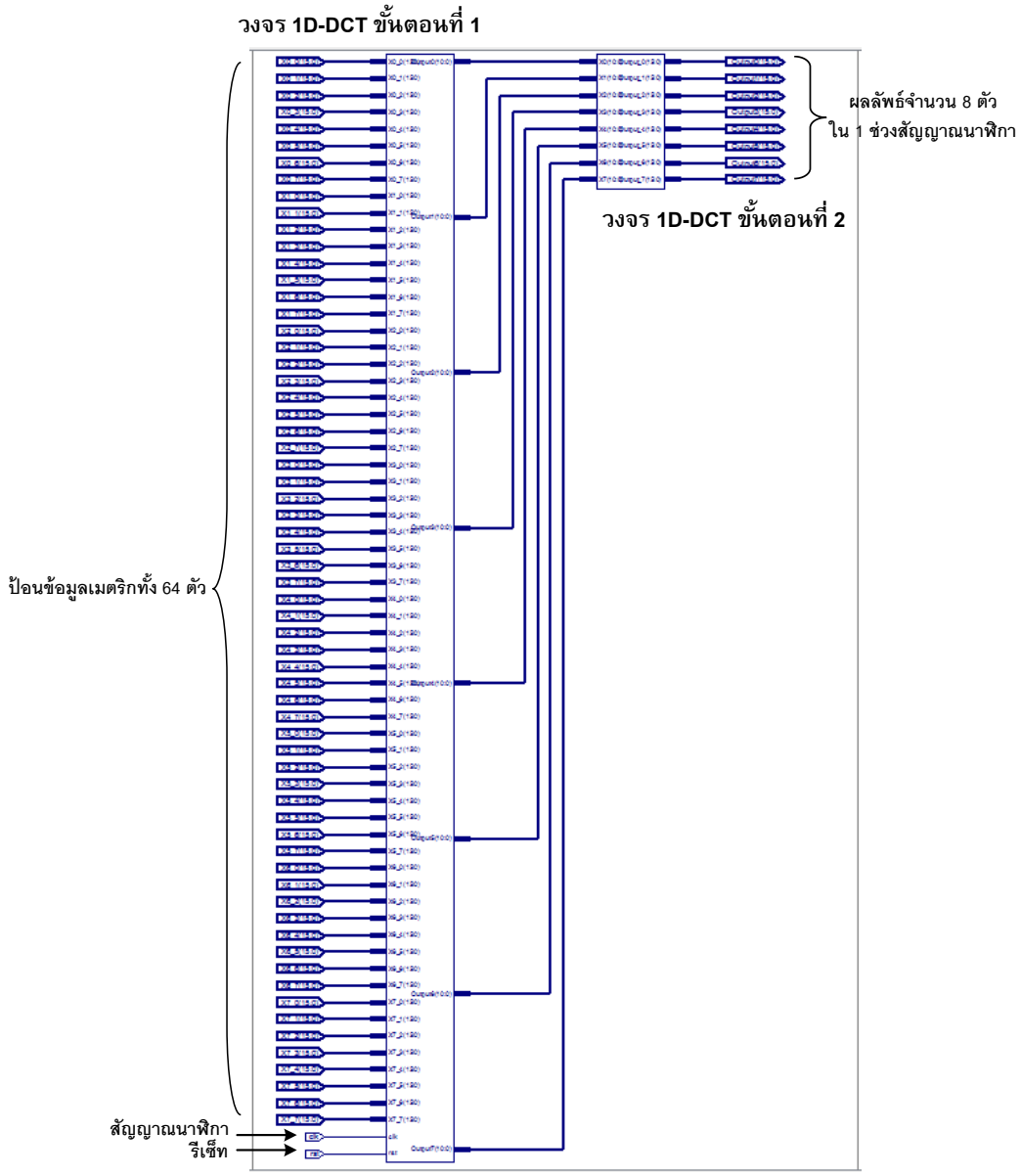
Macro Circuit	**Used**
Multipliers	
- 11 x16-bit multiplier	30
- 16 x11-bit multiplier	32
Adders/Subtractors	
- 11-bit adder	4
- 11-bit addsub	32
- 11-bit subtractor	4
- 27-bit adder	48
Counters	
- 3-bit up counter	1
Registers	
- 1-bit register	8
- 11-bit register	32
Comparators	
- 3-bit comparator less	1

จากการสังเคราะห์วงจร 2D-DCT ด้วยโปรแกรม Xilinx 8.1 สามารถที่จะแสดงแผนผังการออกแบบของวงจรเพื่อตรวจสอบความถูกต้องของวงจร และแสดงถึงการใช้ทรัพยากรในส่วนต่างๆ ในการออกแบบวงจร 2D-DCT รายละเอียดแผนผังวงจรในการออกแบบวงจรแบ่งออกเป็นส่วนย่อยๆ ดังนี้



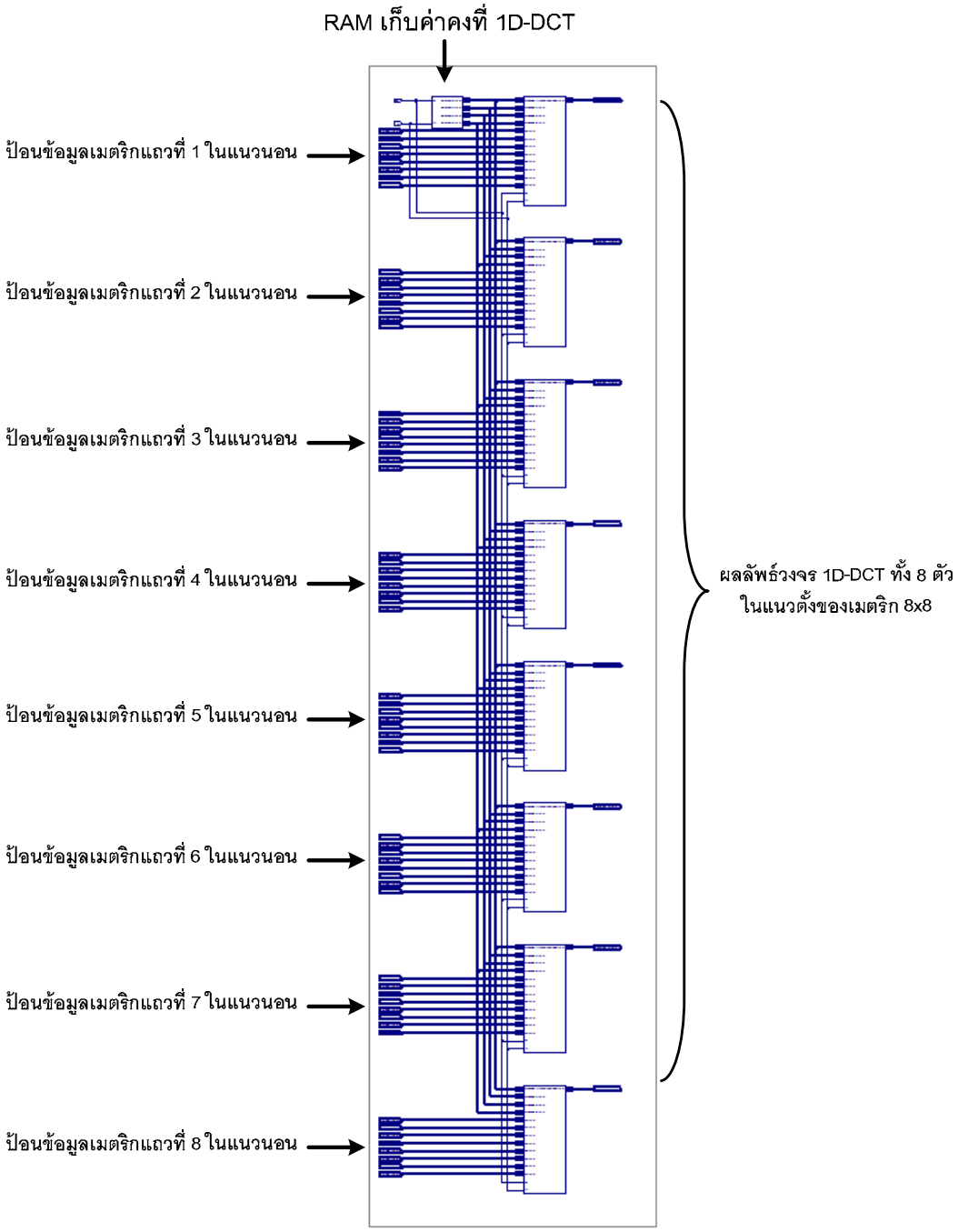
ภาพประกอบ 4-2 แสดงมุมมองด้านนอกของวงจร 2D-DCT

จากภาพประกอบ 4-2 แสดงมุมมองด้านนอกของวงจร 2D-DCT โดยแสดงถึงเส้นสัญญาณที่ป้อนเข้าสู่วงจรซึ่งรับข้อมูลเมตริกขนาด 8x8 มีจำนวนข้อมูล 64 จำนวน และสัญญาณผลลัพธ์ที่ออกจากวงจรมี 8 จำนวน นั่นคือ ผลลัพธ์ในแนวตั้งของเมตริกขนาด 8x8



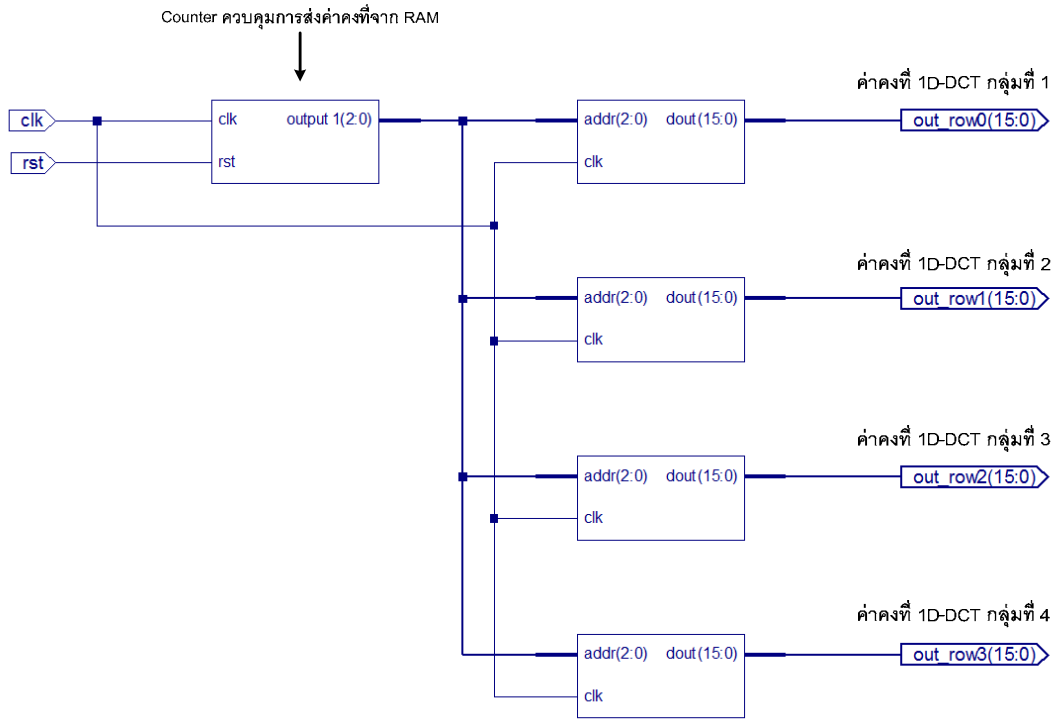
ภาพประกอบ 4-3 แสดงการเชื่อมต่อระหว่างวงจร 1D-DCT ทั้ง 2 ขั้นตอน

จากภาพประกอบ 4-3 แสดงการเชื่อมต่อเส้นสัญญาณข้อมูลจากวงจร 1D-DCT ขั้นตอนที่ 1 เชื่อมต่อเส้นสัญญาณไปยังวงจร 1D-DCT ขั้นตอนที่ 2 ซึ่งข้อมูลผลลัพธ์ทั้ง 8 จำนวน ของวงจร 1D-DCT ขั้นตอนที่ 1 เพียงพอกับความต้องการของวงจร 1D-DCT ขั้นตอนที่ 2 ดังนั้นการออกแบบจึงสามารถที่จะเชื่อมต่อเส้นสัญญาณระหว่างวงจร 1D-DCT ทั้ง 2 วงจรได้โดยไม่จำเป็นต้องสร้างวงจรการพักข้อมูล

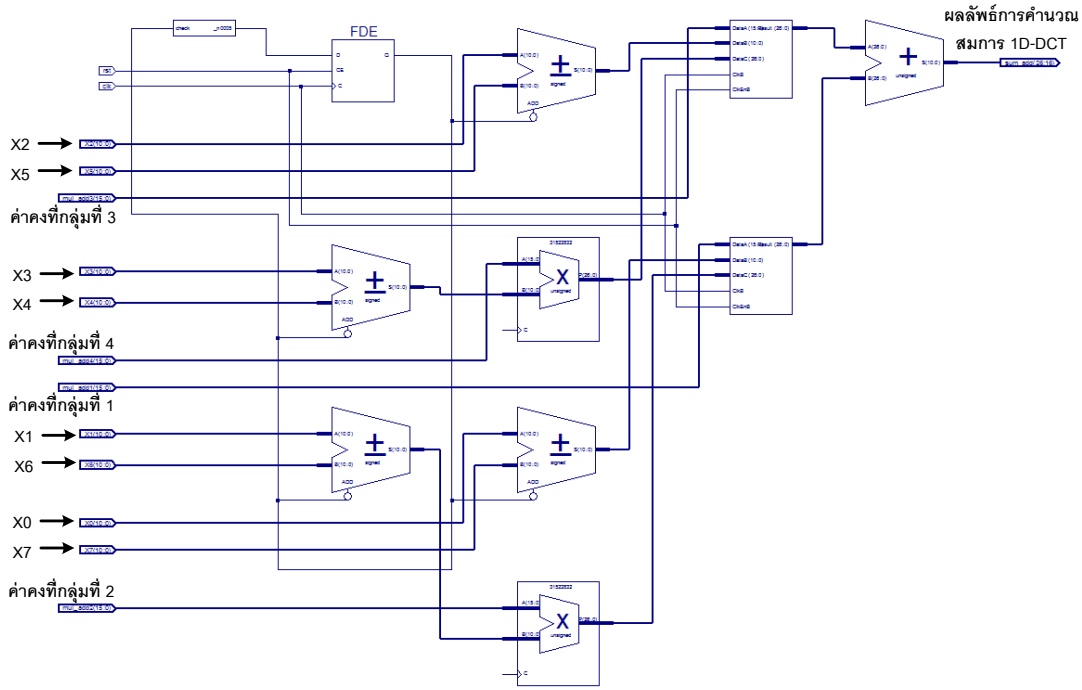


ภาพประกอบ 4-4 แสดงการประมวลผลแบบขนานของวงจร 1D-DCT ขั้นตอนที่ 1

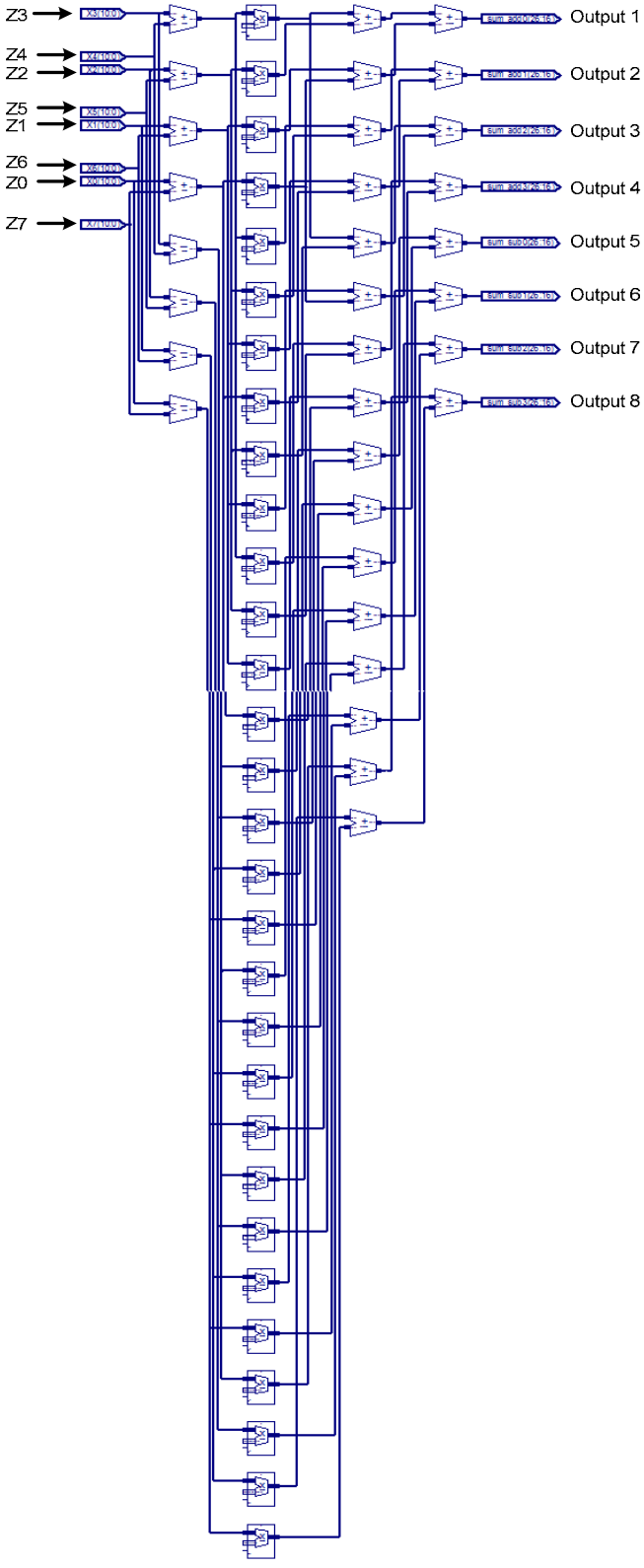
จากภาพประกอบ 4-4 แสดงการออกแบบวงจร 1D-DCT ขั้นตอนที่ 1 ที่ทำการออกแบบให้มีการประมวลผลแบบขนาน โดยแต่ละวงจรของวงจร 1D-DCT ทั้ง 8 วงจรมีการรับข้อมูลจากเมตริกขนาด 8x8 วงจรละ 1 แถวในแนวนอน เข้าทำการประมวลผลได้ผลลัพธ์ใน 1 ช่วงสัญญาณนาฬิกา รวมได้ผลลัพธ์ทั้งหมด 8 จำนวน ในแนวตั้งของเมตริก 8x8



ภาพประกอบ 4-5 แสดงการออกแบบวงจร RAM



ภาพประกอบ 4-6 แสดงวงจรการคำนวณสมการ 1D-DCT ขั้นตอนที่ 1 จำนวน 1 วงจร



ภาพประกอบ 4-7 แสดงวงจร 1D-DCT ขั้นตอนที่ 2

จากภาพประกอบ 4-5 แสดงการออกแบบ RAM สำหรับเก็บค่าคงที่ DCT ซึ่งการออกแบบจะมีวงจรนับ (Counter) เป็นตัวคอยควบคุมการป้อนที่อยู่ข้อมูลใน RAM เพื่อส่งข้อมูลออกไปยังวงจร 1D-DCT การออกแบบจะแบ่ง RAM ออกเป็น 4 แถว เพื่อส่งค่าคงที่ให้วงจรการคำนวณสมการ 1D-DCT เพื่อรองรับการประมวลผลแบบขนาน 4 กลุ่ม ตามที่ได้กล่าวไว้ในบทที่ 3

จากภาพประกอบ 4-6 แสดงวงจรคำนวณสมการ 1D-DCT ขั้นตอนที่ 1 จำนวน 1 วงจร ซึ่งภายในวงจรประกอบไปด้วยวงจรบวก-ลบ วงจรบวก และวงจรคูณ การออกแบบในภาพประกอบ 4-6 เป็นไปตามแผนผังที่ได้ทำการออกแบบดังภาพประกอบ 3-2 มีการรับข้อมูลเมตริกขนาด 8x8 ในแนวนอนเข้ามา 1 แถว ทำการจับคู่ข้อมูลเข้าด้วยกันโดยวิธีการจับคู่ได้กล่าวไว้ในบทที่ 3 จากนั้นส่งไปเข้าสู่วงจรบวก-ลบ นำไปคูณกับค่าคงที่ที่ส่งมาจาก RAM และรวมผลคูณเข้าด้วยกันส่งเป็นผลลัพธ์ของวงจรเพื่อส่งต่อไปยังวงจร 1D-DCT ขั้นตอนที่ 2

จากภาพประกอบ 4-7 แสดงแผนผังวงจรการคำนวณสมการ 1D-DCT ขั้นตอนที่ 2 โดยในวงจรจะถูกออกแบบให้รับข้อมูลจำนวน 8 ข้อมูล นั่นคือ ข้อมูลที่ทำการส่งมาจากวงจรสมการ 1D-DCT ขั้นตอนที่ 1 ใช้เทคนิคการออกแบบการประมวลผลขนานแบบผีเสื้อโดยมีการจับคู่และนำไปแบ่งเป็น 2 กลุ่มใหญ่ๆ คือ กลุ่มของการบวก และกลุ่มของการลบ จากนั้นทำการคูณกับค่าคงที่ 1D-DCT โดยในขั้นตอนนี้ค่าคงที่จะถูกเก็บไว้ในลอจิกของ FPGA เพราะค่าคงที่จำเป็นต้องใช้ตลอดเวลาไม่มีการเปลี่ยนแปลงค่าคงที่ของการคูณ

4.5 การทดสอบพลังงานที่สูญเสียในการประมวลผล

การทดสอบพลังงานที่สูญเสียในการประมวลผลสามารถใช้โปรแกรมเสริมของโปรแกรม Xilinx 8.1 นั่นคือ โปรแกรม Xpower ภายในโปรแกรมจะแบ่งการชนิดของพลังงานที่สูญเสียไปเป็น 2 ส่วน [14] คือ

1. Quiescent Power หรือ Static Power คือ พลังงานที่สูญเสียให้กับ FPGA เมื่อมีการปล่อยกระแสไฟฟ้าเข้าสู่วงจรแต่ยังไม่มีการสวิตช์ของทรานซิสเตอร์ เป็นพลังงานที่สูญเสียภายในทรานซิสเตอร์หรือบริเวณจุดรั่วไหลของกระแส เช่น ข้อต่อระหว่างทรานซิสเตอร์ กระแสที่ไหลจากส่วนของ Source ไปยัง Drain ของทรานซิสเตอร์ และกระแสในส่วนของ Gate ของทรานซิสเตอร์ เป็นต้น

2. Dynamic Power คือ พลังงานที่สูญเสียเมื่อลอจิกภายใน FPGA มีการสวิตซ์โดยพลังงานในส่วนนี้จะขึ้นอยู่กับตัวเก็บประจุ ความถี่ที่มีการสวิตซ์ และแรงดันไฟฟ้าที่ป้อนเข้าสู่วงจร ดังสมการต่อไปนี้

สมการคำนวณหาค่า Dynamic Power [4]

$$\text{Dynamic Power} = CV^2 f$$

เมื่อ C = ขนาดของตัวเก็บประจุของแต่ละโหนดที่มีการสวิตซ์

V = แรงดันไฟฟ้าที่ป้อนเข้าสู่วงจร

f = ค่าความถี่ของการสวิตซ์ในโหนด

ตารางที่ 4-7 แสดงพลังงานที่สูญเสียไปในการประมวลผลของวงจร 2D-DCT

Quiescent Power	0.254 W
Dynamic Power	0.003 W
Total Power	0.257 W

จากตารางที่ 4-7 เป็นการแสดงค่าพลังงานที่ใช้ในวงจร 2D-DCT สังเคราะห์ด้วยโปรแกรม Xpower ซึ่งเป็นโปรแกรมใช้ทดสอบพลังงานที่สูญเสียในวงจร โดยในส่วนของ Quiescent Power ซึ่งเป็นค่าพลังงานที่สูญเสียภายในทรานซิสเตอร์หรือภายในลอจิกของ FPGA มีค่าเท่ากับ 0.254 W ในส่วนของ Dynamic Power พลังงานที่สูญเสียไปในส่วนนี้คือ สัญญาณนาฬิกาที่ป้อนเข้าสู่วงจร สูญเสียพลังงานเท่ากับ 0.003 W รวมพลังงานที่สูญเสียไปทั้งหมด 0.257 W

4.6 การเปรียบเทียบประสิทธิภาพและข้อจำกัดของการประมวลผลวงจร 2D-DCT

จากผลการทดสอบประสิทธิภาพของวงจร 2D-DCT ในด้านต่างๆ ดังที่กล่าวไว้ข้างต้นโดยเฉพาะการทดสอบประสิทธิภาพด้านความเร็ว สามารถที่จะเพิ่มความเร็วในการประมวลผลให้มีความเร็วที่สูงกว่าความเร็วของภาพเคลื่อนไหวที่มีใช้ใน IP Camera และเมื่อนำประสิทธิภาพที่ได้จากการทดสอบไปเปรียบเทียบกับงานวิจัยที่เกี่ยวข้องที่ได้นำเสนอในตารางที่ 1-4 และหัวข้อที่ 2.4 และ 2.5 แสดงดังตารางที่ 4-8

ตารางที่ 4-8 เปรียบเทียบประสิทธิภาพวงจร 2D-DCT กับงานวิจัยที่เกี่ยวข้อง

Design	Technology	Logic Cells	Frequency (MHz)	Clock Cycle (Cycles)	Latency (1/F x clock cycle)
Naras	**Virtex4**	**3,720**	**287.7**	**8**	**0.081 uS**
Luciano[9]	FLEX 10KE	3,962	36.2	163	4.503 uS
Latha[8]	Virtex2	558	182.7	64	0.350 uS
Reza[10]	Virtex4	376	118.2	64	0.541 uS

สำหรับงานวิจัยที่นำมาเปรียบเทียบเป็นงานวิจัยที่นำเสนอทฤษฎีและวิธีการออกแบบวงจร 2D-DCT บนสถาปัตยกรรม FPGA การทดสอบที่จะนำมาเปรียบเทียบ คือ จำนวนสัญญาณนาฬิกาที่ใช้ในการประมวลผล(Clock Cycle) เมตริกขนาด 8x8 จำนวน 1 เมตริก ซึ่งจำนวนสัญญาณนาฬิกาที่ใช้ต่อหนึ่งเมตริกจะบอกประสิทธิภาพด้านความเร็วในการประมวลผล กล่าวคือ หากใช้ความถี่ในการประมวลผลที่เท่ากัน วงจรที่ใช้จำนวนรอบสัญญาณนาฬิกาที่น้อยกว่าจะประมวลผลได้เร็วกว่าวงจรที่ใช้รอบสัญญาณนาฬิกามาก จากตารางที่ 4-8 จะพบว่าจำนวนสัญญาณนาฬิกาที่ทดสอบจากวงจร 2D-DCT ที่ผู้ทำวิจัยได้ออกแบบใช้เพียง 8 รอบสัญญาณนาฬิกาต่อหนึ่งเมตริก ทดสอบบนสถาปัตยกรรม FPGA ชนิด Virtex4 สามารถใช้ความถี่สูงสุด 287.7 เมกกะเฮิร์ต ดังนั้นระยะเวลาที่ใช้ในการคำนวณต่อหนึ่งเมตริกขนาด 8x8 คือ 0.081 ไมโครวินาที (uS)

เทคโนโลยีการออกแบบให้วงจร 2D-DCT ประมวลผลแบบขนานตามที่ได้นำเสนอมีข้อจำกัดแบ่งเป็น 2 อย่าง คือ

1. จำนวนทรัพยากรที่ใช้เพราะการประมวลผลแบบขนานมีการสร้างวงจรให้สามารถประมวลผลได้หลายๆ งานในช่วงเวลาเดียวกัน ซึ่งจำเป็นต้องใช้พื้นที่จำนวนทรัพยากรในการออกแบบที่มากกว่าเพื่อที่จะรองรับการประมวลผล ซึ่งในการออกแบบวงจรดิจิตอล อุปกรณ์ที่ใช้ในการประมวลผลจะมีจำนวนทรัพยากรที่แตกต่างกันออกไป ดังนั้นการที่จะนำวิธีการออกแบบวงจรไปใช้จำเป็นจะต้องคำนึงถึงปัจจัยต่างๆ เช่น ลักษณะงานที่จะนำไปใช้ อุปกรณ์ที่นำมาประมวลผล เป็นต้น

2. โครงสร้างการไหลของข้อมูลของวงจร 2D-DCT ที่ได้ทำการออกแบบมีการออกแบบให้รับข้อมูลแบบขนาน กล่าวคือ ในวงจรจำเป็นที่จะต้องป้อนข้อมูลเมตริกการคำนวณขนาด 8x8 เข้าสู่วงจรพร้อมกันทั้งหมด 64 จำนวน ดังนั้น หากต้องการที่จะนำวงจรที่ได้ทำการ

ออกแบบไปใช้เป็นส่วนหนึ่งของการประมวลผลของวงจรบีบอัดรูปภาพ จึงจำเป็นต้องมีการออกแบบให้วงจรทุกส่วนประมวลผลแบบขนาน หรือมีการจัดการข้อมูลเมตริกขนาด 8x8 ให้สามารถป้อนเข้าสู่วงจร 2D-DCT พร้อมกันทั้งหมดในช่วงเวลาเดียวกัน

4.7 บทสรุป

จากการทดสอบประสิทธิภาพการออกแบบวงจร 2D-DCT ประมวลผลแบบขนานในด้านต่างๆ ดังที่กล่าวมาข้างต้น สรุปได้ว่า วงจรที่ได้ออกแบบสามารถประมวลผลคำนวณสมการ 2D-DCT ได้ถูกต้องและสามารถนำไปใช้ประยุกษ์ในการบีบอัดรูปภาพแบบ JPEG ได้

การทดสอบด้านความเร็วของการประมวลผลสามารถใช้สัญญาณความถี่ของสัญญาณนาฬิกาที่ป้อนเข้าสู่วงจรได้กว่า 287 MHz ซึ่งเป็นการบีบอัดรูปภาพได้ความเร็วแบบทันทีทันใด จึงสามารถที่จะนำไปประยุกษ์เป็นส่วนหนึ่งของการบีบอัดรูปภาพเคลื่อนไหวที่ต้องการความเร็วสูงทันกับเหตุการณ์ที่เกิดขึ้นได้

การทดสอบทรัพยากรและพลังงานที่ใช้ในการประมวลผลวงจรคำนวณสมการ 2D-DCT เมื่อใช้อ้างอิงกับอุปกรณ์ FPGA ตระกูล Virtex 4 ชิปเบอร์ XC4VFX12 ทรัพยากรแผ่นลอจิกของ FPGA ที่ใช้ไปประมาณ 36% นั่นหมายความว่ายังมีพื้นที่ของทรัพยากรเหลือที่จะใช้ในการประยุกษ์และพัฒนาวงจรการบีบอัดรูปภาพได้ นอกจากนั้นได้ใช้วงจรมาโครเข้ามาช่วยในการออกแบบของการคำนวณทางคณิตศาสตร์ ส่วนพลังงานที่ใช้ในการประมวลผลนอกจากพลังงานต่ำสุดที่สูญเสียในทรานซิสเตอร์หรือลอจิกต่างๆ เมื่อมีการป้อนกระแสไฟฟ้าเข้าสู่วงจรแล้ว ส่วนที่สูญเสียอันเนื่องมาจากการออกแบบวงจรและการประมวลผล คือส่วนของสัญญาณนาฬิกาเพียงส่วนเดียวเท่านั้น ซึ่งต้องใช้พลังงานเพิ่มขึ้นเพียง 3 mW จากค่าพลังงานต่ำสุดที่ต้องสูญเสีย

บทที่ 5
บทสรุป ปัญหาและข้อเสนอแนะ

ในบทนี้จะกล่าวถึงบทสรุปและข้อเสนอแนะของการดำเนินงานทำวิจัย ปัญหาและอุปสรรคต่างๆ ที่เกิดขึ้นในขณะที่ทำวิทยานิพนธ์ และท้ายที่สุดจะกล่าวถึงรายละเอียดให้ข้อเสนอแนะแก่ผู้ที่สนใจที่จะนำวิทยานิพนธ์ชุดนี้ไปพัฒนาต่อไป

5.1 บทสรุปของการทำวิทยานิพนธ์

วิทยานิพนธ์ชุดนี้เป็นการนำเสนอวิธีการออกแบบวงจรการคำนวณสมการ 2D-DCT ซึ่งเป็นขั้นตอนหนึ่งที่มีความซับซ้อนและใช้ระยะเวลาในการประมวลผลมากที่สุดของการบีบอัดรูปภาพแบบ JPEG ดังนั้นหากต้องการที่จะเพิ่มประสิทธิภาพด้านความเร็วของการบีบอัดรูปภาพแบบ JPEG จึงได้เสนอแนวคิดและวิธีการออกแบบวงจรการประมวลผลสมการ 2D-DCT เพื่อเพิ่มความเร็วในการประมวลผลเพื่อให้มีประสิทธิภาพความเร็วที่เทียบเท่าหรือสูงกว่าการบีบอัดรูปภาพแบบทันทีทันใด

การออกแบบวงจรดิจิตอลเพื่อใช้คำนวณสมการ 2D-DCT วิธีการที่ง่ายและได้ผลลัพธ์ที่ถูกต้อง คือ การใช้วงจรการคำนวณสมการ 1D-DCT เข้ามาช่วยในการออกแบบโดยการประมวลผล 2 ครั้ง ซึ่งข้อมูลที่ใช้ในการประมวลผลจะมีลักษณะเป็นเมตริกขนาด 8x8 ในขั้นตอนการประมวลผลวงจร 1D-DCT ครั้งแรก จะคำนวณโดยพิจารณาข้อมูลในแนวนอนของเมตริกข้อมูลก่อน หลังจากนั้นจึงส่งข้อมูลไปประมวลผลวงจร 1D-DCT อีกครั้งแต่พิจารณาข้อมูลในแนวตั้ง ซึ่งหากวิเคราะห์ขั้นตอนการคำนวณวงจรในขั้นตอนที่ 2 จะสามารถเริ่มประมวลผลจะต้องรอให้ขั้นตอนที่ 1 ประมวลผลให้ได้ข้อมูลตามที่ต้องการก่อน ดังนั้นหากสามารถลดช่วงเวลาที่ต้องรอระหว่างวงจร 1D-DCT ทั้งสองขั้นตอนลง ก็จะสามารถเพิ่มความเร็วในการประมวลผลได้

จึงได้เสนอแนวความคิดและทฤษฎีที่นำมาใช้ในการออกแบบวงจรคำนวณสมการ 2D-DCT เพื่อเพิ่มความเร็วในการประมวลผลประกอบไปด้วย 2 ส่วนหลักๆ ได้แก่

1. การประมวลผลแบบขนาน

หมายถึง การออกแบบวงจรการประมวลผลให้ในช่วงเวลาเดียวกันสามารถที่จะประมวลผลหลายๆ งานพร้อมกันได้ เป็นการลดระยะเวลาในการประมวลผลทำให้สามารถประมวลผลข้อมูลได้อย่างต่อเนื่อง โดยไม่ต้องสร้างวงจรบัพเฟอร์เพื่อรองานที่มีการประมวลผล

ก่อนหน้า อีกทั้งยังเป็นการใช้ประโยชน์ของหน่วยประมวลผลหลัก (Central Processing Unit) ได้อย่างเต็มประสิทธิภาพ

2. การใช้วงจรมาโคร(Macro) ในการออกแบบ

หมายถึง การนำวงจรการคำนวณทางคณิตศาสตร์ที่มีอยู่ในสถาปัตยกรรม FPGA เข้ามาใช้ในการประมวลผล เป็นวงจรที่ถูกพัฒนาขึ้นโดยบริษัทผู้ผลิตของอุปกรณ์ FPGA ซึ่งได้รับการทดสอบแล้วว่าเป็นวงจรที่มีความเร็วสูงสุดในการประมวลผลวงจรทางคณิตศาสตร์ ในการทำงานวิจัยได้ใช้อุปกรณ์ FPGA ตระกูล Virtex 4 ชิปเบอร์ XC4VFX12 ซึ่งในสถาปัตยกรรม FPGA ตระกูล Virtex 4 มีวงจรมาโครที่ชื่อว่า DSP48 เป็นวงจรมาโครชนิดหนึ่งที่สามารถประมวลผลได้สูงสุดถึง 48 บิต ช่วยให้ประมวลผลได้อย่างรวดเร็วอีกทั้งยังเป็นการลดการสิ้นเปลืองทรัพยากรจำนวนแผ่นลอจิกของอุปกรณ์ FPGA ที่ใช้ในการออกแบบวงจร

5.2 ผลที่ได้จากการทำวิทยานิพนธ์

ผู้ทำวิทยานิพนธ์ได้ศึกษาและวิเคราะห์การออกแบบวงจรดิจิตอลการคำนวณสมการ 2D-DCT ซึ่งมีจุดประสงค์เพื่อเป็นส่วนหนึ่งในการเพิ่มความเร็วในการประมวลผลของการบีบอัดรูปภาพแบบ JPEG เพื่อที่จะนำแนวคิดการใช้วงจรการประมวลผลสมการ 1D-DCT เข้ามาช่วยในการออกแบบวงจรสมการ 2D-DCT (รายละเอียดในบทที่ 2 หัวข้อ 2.5) โดยพัฒนาและออกแบบบนสถาปัตยกรรม FPGA บรรยายด้วยภาษา VHDL โดยใช้เทคนิคและแนวคิดการประมวลผลแบบขนาน เข้ามาช่วยลดระยะเวลาที่ใช้ในการประมวลผล (รายละเอียดในบทที่ 3) ท้ายที่สุดได้นำวงจรที่ผ่านการออกแบบมาทดสอบประสิทธิภาพในด้านความถูกต้องแม่นยำของการประมวลผล ความเร็วสูงสุดที่สามารถใช้ได้ ทรัพยากรและพลังงานที่สูญเสีย ซึ่งสรุปได้ว่าสามารถเพิ่มความเร็วในการประมวลผลได้เป็นที่น่าพอใจ ประสิทธิภาพด้านความถูกต้องในการประมวลผลเมื่อทดสอบกับการบีบอัดรูปอยู่ในระดับที่ยอมรับได้ รวมไปถึงทรัพยากรและพลังงานที่ใช้เหมาะสมกับอุปกรณ์ที่ใช้เป็นต้นแบบในการทำงานวิจัย (รายละเอียดในบทที่ 4)

ผลที่ได้รับจากการทำวิทยานิพนธ์ชุดนี้ คือ ได้แนวคิดและวิธีการออกแบบวงจรการประมวลผลสมการ 2D-DCT โดยพัฒนาและออกแบบบนสถาปัตยกรรม FPGA เพื่อใช้เป็นขั้นตอนหนึ่งของการบีบอัดรูปภาพแบบ JPEG สามารถเพิ่มความเร็วในการประมวลผลให้มีประสิทธิภาพที่ดีกว่าหรือเทียบเท่ากับการบีบอัดรูปภาพแบบทันทีทันใด

5.3 ปัญหาและอุปสรรค์ของการทำวิทยานิพนธ์

ปัญหาและอุปสรรค์ในการทำวิทยานิพนธ์ชุดนี้ แบ่งออกเป็น 3 ส่วนหลักๆ คือ ปัญหาที่เกิดขึ้นจากการทำวิทยานิพนธ์ ปัญหาที่เกิดขึ้นจากตัวเครื่องมือและอุปกรณ์ ปัญหาด้านผู้ทำวิทยานิพนธ์ ซึ่งจะกล่าวรายละเอียดต่างๆ ตามลำดับดังต่อไปนี้

1. ปัญหาที่เกิดขึ้นจากการทำวิทยานิพนธ์ คือในส่วนของทรัพยากรที่ใช้ในการออกแบบวงจรไม่เพียงพอ กล่าวคือ ด้วยพื้นฐานของการออกแบบวงจรดิจิตอล หากไม่สนใจขนาดของข้อมูลที่ใช้ในการประมวลผลผู้ทำการออกแบบจะออกแบบให้มีขนาดจำนวนบิตการคำนวณที่มากเพื่อรองรับกับการประมวลผลโดยทั่วไปจะออกแบบให้มีขนาด เช่น 8, 16, 32, 64 เป็นต้น แต่ในบางครั้งหากพิจารณาขนาดข้อมูลอย่างละเอียดแล้วจำนวนบิตที่ออกแบบให้มากเกินความต้องการจะทำให้สิ้นเปลืองทรัพยากรที่ใช้ในการประมวลผลโดยเปล่าประโยชน์ ทำให้ทรัพยากรที่มีไม่เพียงพอกับความต้องการ อีกทั้งการเทคนิคการเก็บค่าคงที่ไว้ใน RAM เป็นทางหนึ่งที่สามารถใช้ลดขนาดทรัพยากรที่ใช้ได้ ซึ่งผู้ทำวิทยานิพนธ์ได้แก้ปัญหาในตรงจุดนี้และได้ผลสรุปขนาดทรัพยากรที่ใช้เป็นที่ยอมรับได้ดังที่กล่าวไว้ในหัวข้อที่ 4.4

2. ปัญหาที่เกิดขึ้นจากตัวเครื่องมือและอุปกรณ์ คือ ในส่วนของการทดสอบความถูกต้องของขั้นตอนการบีบอัดรูปภาพ ในส่วนนี้จะต้องใช้โปรแกรมจำลองการทำงานการบีบอัดรูปภาพแบบ JPEG ซึ่งการที่จะเอามาใช้ในการทดสอบจะต้องเป็นโปรแกรมที่ได้ผ่านการทดสอบแล้วว่าสามารถบีบอัดรูปภาพได้จริงและถูกต้องตามหลักการบีบอัดรูปภาพแบบ JPEG โปรแกรมจำลองที่ใช้ทดสอบหาได้ยาก ซึ่งตัวผู้ทำวิทยานิพนธ์เองได้ใช้เวลาในการค้นหาวิธีการออกแบบโปรแกรมจำลองที่จะใช้ในการทดสอบค่อนข้างนาน ทั้งในเรื่องเครื่องมือที่จะใช้ในการสร้างโปรแกรมจำลอง โครงสร้างในการออกแบบและรวมไปถึงวิธีการวิเคราะห์ค่าที่ได้จากการทดสอบจากโปรแกรมจำลอง

3. ปัญหาด้านผู้ทำวิทยานิพนธ์ คือ เมื่อเริ่มต้นทำวิทยานิพนธ์ตัวของผู้ทำวิทยานิพนธ์มีพื้นฐานทักษะด้านการออกแบบวงจรดิจิตอล แต่ยังมีพื้นด้านการบีบอัดรูปภาพแบบ JPEG น้อยมาก ดังนั้นจึงใช้ระยะเวลาในการศึกษาเกี่ยวกับการบีบอัดรูปภาพแบบ JPEG ค่อนข้างนาน เพื่อให้ทราบถึงขั้นตอนและปัญหาเพื่อนำมาวิเคราะห์แก้ไข และรวมไปถึงพื้นฐานทักษะการเขียนรายงานเชิงวิชาการยังไม่ดีเท่าที่ควรจึงใช้ระยะเวลาค่อนข้างนานในการเขียนสรุปรายงานวิจัย

5.4 ข้อเสนอแนะ

งานวิจัยในวิทยานิพนธ์ชุดนี้ เป็นการศึกษาการบีบอัดรูปภาพแบบ JPEG และเน้นที่พัฒนาและออกแบบวงจรการบีบอัดรูปภาพเพียงแค่ในส่วนของการประมวลผลสมการ 2D-DCT ซึ่งผู้ที่สนใจควรจะศึกษาวิธีการการบีบอัดรูปภาพแบบ JPEG ในขั้นตอนต่างๆ เพื่อที่จะนำวงจร 2D-DCT ที่ได้นำเสนอในวิทยานิพนธ์ชุดนี้ ใช้เป็นส่วนหนึ่งของการออกแบบวงจรการบีบอัดรูปภาพแบบ JPEG ทั้งชนิดภาพนิ่งและภาพเคลื่อนไหว

เอกสารอ้างอิง

[1] ชำนาญ ปัญญาใส และ วัชรากร หนูทอง, “ ภาษา VHDL สำหรับการออกแบบวงจรดิจิตอล ,” *สำนักพิมพ์ซีเอ็ด*, กรุงเทพ, 2004, หน้า 11-13.

[2] บัณฑิต ทิพากร และ ชำรงรัตน์ อมรรักษา. (20 มีนาคม 2553). การทำภาพพิมพ์ลายน้ำดิจิตอลด้วยเทคนิคการกระจายแถบความถี่[Online].http://cpe.kmutt.ac.th/lab/mcl/DevTech.htm

[3] Altium. (2010, April 15). *การออกแบบ FPGA และ Embedded ด้วย NB2* [Online]. Available: http://www.altiumthai.com/index.php?option=com_content&view=article&id=46&Itemid=28

[4] A. Telikepalli, (2006, May 19). Power vs. Performance: The 90 nm Inflection Point [Online]. Available: http://www.xilinx.com/support/documentation/white_papers/wp223.pdf

[5] C. Maxfield. (2010, April 15). *FPGA Architectures*[Online]. Available:http://i.cmpnet.com/pldesignline/2006/08/fpga-101-09.gif

[6] Douglas A. Kerr. (2009, December 3). Chrominance Subsampling in Digital Images [Online]. Available: http://dougkerr.net/Pumpkin/articles/Subsampling.pdf .

[7] K. Z. Bukhari, “ Visual Data Transforms Comparison,” M.S. thesis, Dept. Elect. Eng., Delft Univ., Netherland, 2002.

[8] L. Pillai. “ Video Compression Using DCT ,” Appl.Note : Xapp610, Xilinx, 2002.

[9] L. V. Agostini and S. Bampi, “ High Throughput Architecture of JPEG Compressor for Color Images Targeting FPGAs,” *IEEE Trans. Electron.Syst.*, Vol.ICECS-13, pp180-183, Dec. 2006.

[10] R. E. Atani, M Baboli and S. Mirzakuchaki, “ Design and implementation of a 118 MHz 2D DCT processor ,” IEEE Trans. Indus. Electron. Vol. ISIE. pp 1076-1081, July. 2008.

[11] S. Timakul, “ Modification of JPEG algorithm for FPGA implementation ” M.S. thesis, Elect. Eng., KMITL Univ., Thailand, 2005.

[12] Wikipedia. (2010, April 5). Butterfly diagram[Online]. Available: http://en.wikipedia.org/wiki/Butterfly_diagram

[13] Wikipedia. (2010, April 5). JPEG [Online]. Available: http://th.wikipedia.org/wiki/JPEG

[14] Xilinx. (2009, June 24). Xilinx Power Estimator User Guide [Online]. Available: http://www.xilinx.com/support/documentation/user_guides/ug440.pdf

ภาคผนวก

ภาคผนวก ก

โปรแกรมจำลองทดสอบความถูกต้องขั้นตอนการบีบอัดรูปภาพแบบ JPEG

โปรแกรมจำลองการบีบอัดรูปภาพแบบ JPEG เป็นโปรแกรมที่ใช้ในการทดสอบความถูกต้องของขั้นตอนการบีบอัดรูปภาพ ซึ่งใช้ขั้นตอนวิธีการคำนวณสมการ 2D-DCT ดังที่ได้กล่าวไว้ในหัวข้อที่ 2.4 โปรแกรมจำลองการบีบอัดรูปภาพแบบ JPEG ออกแบบโดยใช้ Matlab 7.8 สำหรับซอสโค๊ดที่ใช้ในการทดสอบจะแบ่งออกเป็น 2 ส่วนย่อยๆ คือ โปรแกรมการบีบอัดรูปภาพ และโปรแกรมการคำนวณหาค่า PSNR ซึ่งรายละเอียดจะกล่าวเรียงตามลำดับดังนี้

โปรแกรมการบีบอัดรูปภาพ

```
function [ img ] = comp( input_f,output_f )

%    วิธีการใช้คำสั่งการทำงาน
%    comp('ชื่อไฟล์ที่ต้องการบีบอัด','ชื่อไฟล์ที่บันทึกหลังจากการบีบอัด')

aa = imread(input_f);        % อ่านข้อมูลรูปภาพที่ต้องการบีบอัด

figure('Name','Input image');
imshow(aa);                  % แสดงรูปภาพก่อนการบีบอัด

quant_multiple = 1;          % กำหนดค่าตัวคูณของค่าคงที่ Quantization

% เมตริกค่าคงที่สำหรับการคำนวณสมการ 1D-DCT ครั้งที่ 1
CT = [23170,32138,30274,27246,23170,18205,12540,6393;
      23170,27246,12540,-6393,-23170,-32138,-30274,-18205;
      23170,18205,-12540,-32138,-23170,6393,30274,27246;
      23170,6393,-30274,-18205,23170,27246,-12540,-32138;
      23170,-6393,-30274,18205,23170,-27246,-12540,32138;
      23170,-18205,-12540,32138,-23170,-6393,30274,-27246;
      23170,-27246,12540,6393,-23170,32138,-30274,18205;
      23170,-32138,30274,-27246,23170,-18205,12540,-6393;];

% เมตริกค่าคงที่สำหรับการคำนวณสมการ 1D-DCT ครั้งที่ 2
C = [23170,23170,23170,23170,23170,23170,23170,23170;
    32138,27246,18205,6393,-6393,-18205,-27246,-32138;
    30274,12540,-12540,-30274,-30274,-12540,12540,30274;
    27246,-6393,-32138,-18205,18205,32138,6393,-27246;
    23170,-23170,-23170,23170,23170,-23170,-23170,23170;
    18205,-32138,6393,27246,-27246,-6393,32138,-18205;
    12540,-30274,30274,-12540,-12540,30274,-30274,12540;
    6393,-18205,27246,-32138,32138,-27246,18205,-6393;];
```

```
% เมตริกค่าคงที่ Quantization
DCT_quantizer = ...
    [ 16  11  10  16  24  40  51  61; ...
      12  12  14  19  26  58  60  55; ...
      14  13  16  24  40  57  69  56; ...
      14  17  22  29  51  87  80  62; ...
      18  22  37  56  68 109 103  77; ...
      24  35  55  64  81 104 113  92; ...
      49  64  78  87 103 121 120 101; ...
      72  92  95  98 112 100 103  99 ];

% ***** การบีบอัดรูปภาพ *****
for b = 0:8:504
for a = 0:8:504
    for y = 1:8                          % แบ่งข้อมูลรูปภาพออกเป็นเมตรอกขนาด 8x8
        for x = 1:8

         input(x,y) = aa(a+x,y+b);

        end
    end

    Input = double(input);               % แปลงข้อมูลจากเลขฐาน 2 ให้เป็นฐาน 10

    buffer = Input*CT;                   % 2D-DCT ขั้นตอนที่ 1
    shift = fix(buffer/65535);

    buffer = C*shift;                    % 2D-DCT ขั้นตอนที่ 2
    Big_DCT = fix(buffer/65535);

                                         % การทำ Quantization
    DCT_matrix = floor (Big_DCT ...
          ./ (DCT_quantizer(1:8, 1:8) * quant_multiple) + 0.5);



                                           % รวมข้อมูลการบีบอัด
    for y1 = 1:8
        for x1 = 1:8

         Output_Big(a+x1,y1+b) = DCT_matrix(x1,y1);

        end
    end
end
end
```

```
% ***** การแสดงภาพที่ผ่านการบีบอัด *****
for b = 0:8:504
for a = 0:8:504

    for y = 1:8                          % แบ่งข้อมูลออกเป็นเมตรอกขนาด 8x8
        for x = 1:8

         input_dct_big(x,y) = Output_Big(a+x,y+b);

        end
    end

                                         % การทำ Quantization ย้อนกลับ
 IDCT_matrix = input_dct_big ...
               .* (DCT_quantizer(1:8, 1:8) * quant_multiple);


        idct_big = idct2(IDCT_matrix);  % การทำ 2D-IDCT

    for y1 = 1:8                        % รวมข้อมูลเป็นข้อมูลรูปภาพ
        for x1 = 1:8

         Image_Big(a+x1,y1+b) = idct_big(x1,y1);

        end
    end
end
end

    Image_Big = uint8(Image_Big);      % แปลงข้อมูลเป็นเลขฐาน 2 ขนาด 8 บิต

    imwrite(Image_Big, output_f);      % บันทึกรูปภาพที่ผ่านการบีบอัด
    imwrite(aa, 'input.bmp');

    figure('Name','Output image_Big');
    imshow(Image_Big);                 % แสดงรูปภาพที่ผ่านการบีบอัด
```

โปรแกรมการคำนวณหาค่า PSNR

```
double sum_diff;
double matrix_sum_diff;

sum_diff = 0;
matrix_sum_diff = 0;

for y = 1:512
    for x = 1:512

        matrix_sum_diff(x,y) = (aa(x,y) - Image_Big(x,y))^2;

        sum_diff = sum_diff + matrix_sum_diff(x,y);

    end
end


MSE = sum_diff/(512*512)

RMSE = sqrt(MSE)

PSNR = 20*(log10(255/RMSE))

end
```

ภาคผนวก ข

การออกแบบหน่วยความจำของ FPGA เพื่อเก็บค่าคงที่ด้วย IP Core

ในการออกแบบวงจร 2D-DCT ที่กล่าวไว้ในบทที่ 3 การเก็บค่าคงที่ของสมการ 1D-DCT จะเก็บไว้ในหน่อยความจำของอุปกรณ์ FPGA หรือที่เรียกว่า “ RAM ” ซึ่งในส่วนนี้ได้ใช้ไอพีคอร์(IP Core) ของโปรแกรม Xilinx 8.1 ในการออกแบบจะต้องสร้างไฟล์ใช้สำหรับโหลดค่าคงที่ลงไปในโปรแกรมที่ใช้ในการสร้างไอพีคอร์ การเขียนซอสโค๊ดสำหรับการโหลดลงไปในโปรแกรมและขั้นตอนการกำหนดค่าต่างๆ ของหน่วยความจำในไอพีคอร์ ซึ่งรายละเอียดจะกล่าวเรียงตามลำดับดังต่อไปนี้

ค่าคงที่ใช้ในการโหลดเข้าไปในไอพีคอร์

ค่าคงที่ที่ใช้เก็บไว้ในไอพีคอร์ เป็นค่าคงที่ที่ใช้ในการประมวลผลวงจร 1D-DCT ขั้นตอนที่ 1 ซึ่งการออกแบบในหัวข้อ 3.2 มีการใช้กลุ่มของค่าคงที่จำนวน 4 กลุ่มป้อนเข้ามายังวงจรการคำนวณกลุ่มละ 1 ตัว ดังนั้นจึงได้ทำการออกแบบโดยเก็บค่าคงที่ในหน่วยความจำ 4 ชุด ตามกลุ่มของค่าคงที่ ไฟล์ที่เขียนซอสโค๊ดจะจัดเก็บเป็นชนิดไฟล์แบบ .coe ซึ่งซอสโค้ดสำหรับโหลดค่าคงที่เข้าไอพีคอร์ ทั้ง 4 ชุด มีดังรายละเอียดต่อไปนี้

ค่าคงที่ชุดที่ 1 ไฟล์ชื่อ row0.coe

```
memory_initialization_radix = 2;        // จัดการข้อมูลอยู่ในรูปของเลขฐาน 2
memory_initialization_vector =          // ข้อมูลค่าคงที่
0101101010000010                        // 23170
0111110110001010                        // 32138
0111011001000010                        // 30274
0110101001101110                        // 27246
0101101010000010                        // 23170
0100011100011101                        // 18205
0011000011111100                        // 12540
0001100011111001;                       // 6393
```

ค่าคงที่ชุดที่ 2 ไฟล์ชื่อ row1.coe

```
memory_initialization_radix = 2;        // จัดการข้อมูลอยู่ในรูปของเลขฐาน 2
memory_initialization_vector =          // ข้อมูลค่าคงที่
0101101010000010                        // 23170
0110101001101110                        // 27246
0011000011111100                        // 12540
1110011100000111                        // -6393
1010010101111110                        // -23170
1000001001110110                        // -32138
1000100110111110                        // -30274
1011100011100011;                       // -18205
```

ค่าคงที่ชุดที่ 3 ไฟล์ชื่อ row2.coe

```
memory_initialization_radix = 2;        // จัดการข้อมูลอยู่ในรูปของเลขฐาน 2
memory_initialization_vector =          // ข้อมูลค่าคงที่
0101101010000010                        // 23170
0100011100011101                        // 18205
1100111100000100                        // -12540
1000001001110110                        // -32138
1010010101111110                        // -23170
0001100011111001                        // 6393
0111011001000010                        // 30274
0110101001101110;                       // 27246
```

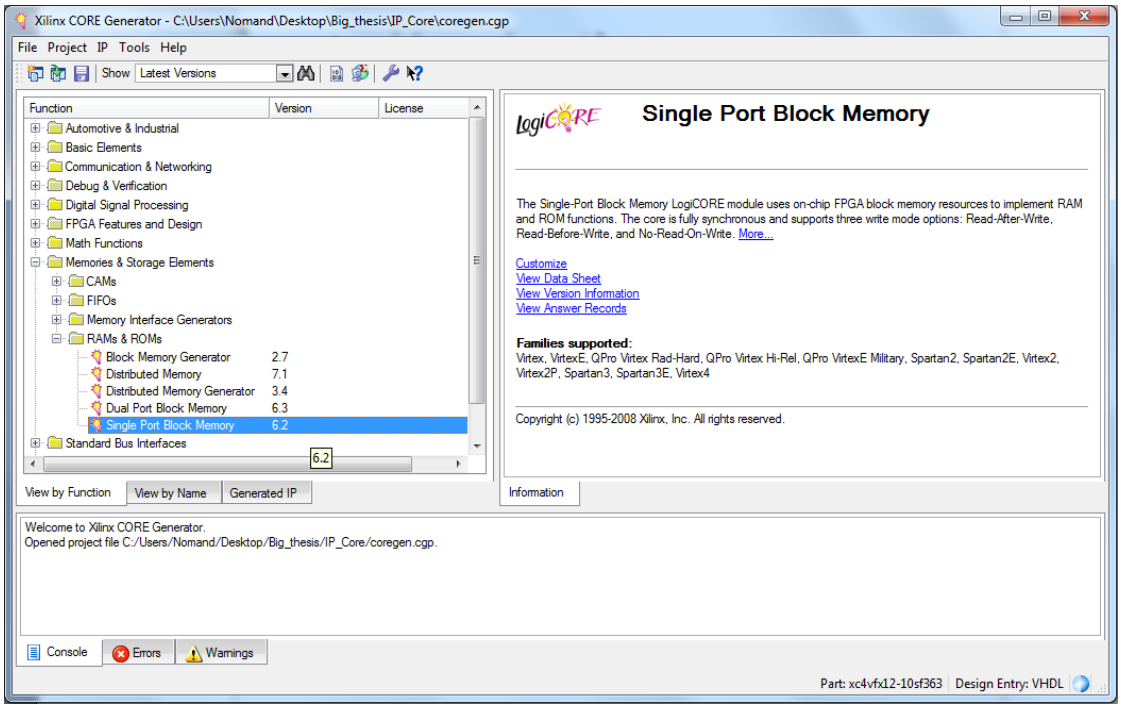
ค่าคงที่ชุดที่ 4 ไฟล์ชื่อ row3.coe

```
memory_initialization_radix = 2;          // จัดการข้อมูลอยู่ในรูปของเลขฐาน 2
memory_initialization_vector =            // ข้อมูลค่าคงที่
0101101010000010                          // 23170
0001100011111001                          // 6393
1000100110111110                          // -30274
1011100011100011                          // -18205
0101101010000010                          // 23170
0110101001101110                          // 27246
1100111100000100                          // -12540
1000001001110110;                         // -32138
```

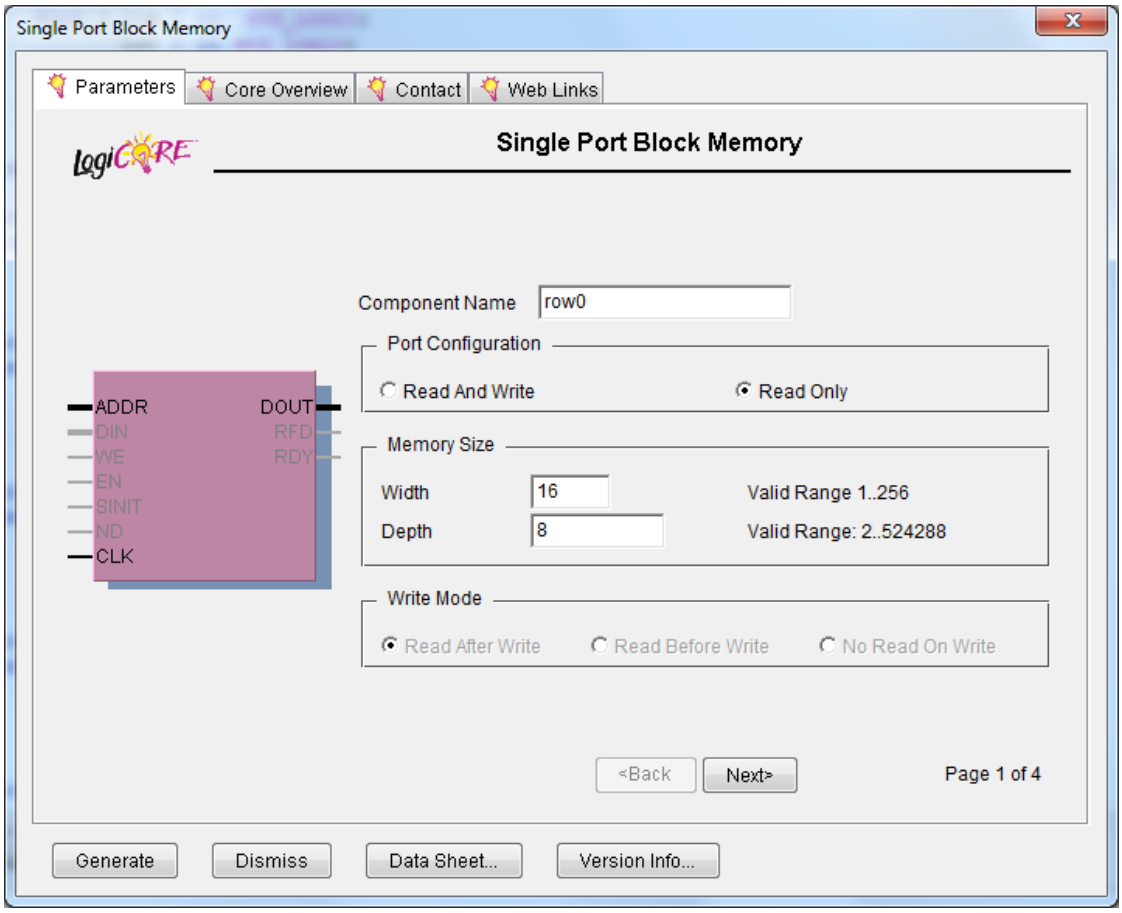
ขั้นตอนการกำหนดค่าต่างๆ ของหน่วยความจำในไอพีคอร์

หลังจากสร้างไฟล์สำหรับโหลดค่าคงที่เข้าสู่ไอพีคอร์แล้ว โปรแกรมที่ใช้ในการออกแบบคือ Xilinx CORE Generator โดยมีลำดับการกำหนดค่าของไอพีคอร์ ดังต่อไปนี้

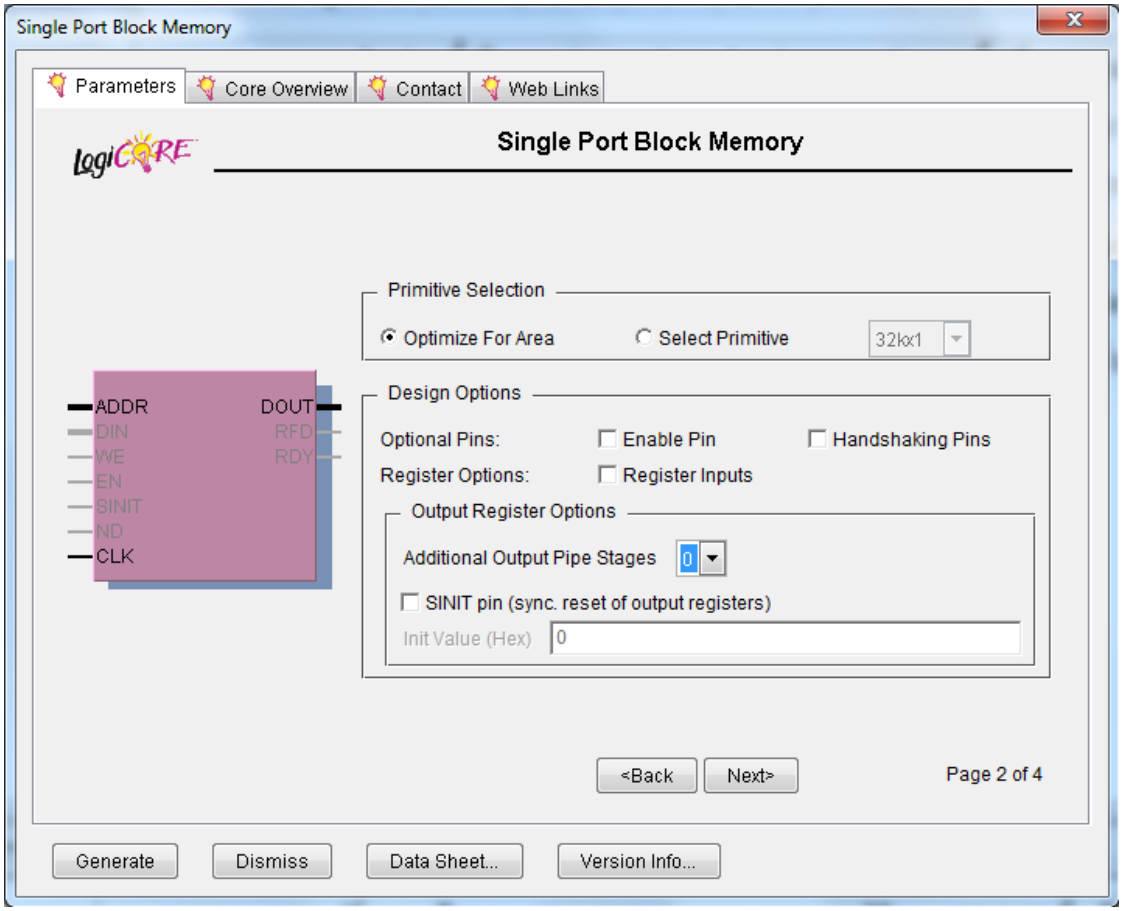
1. หลังจากสร้างกลุ่มงานแล้วเลือกไอพีคอร์ ที่มีชื่อว่า Single Port Block Memory



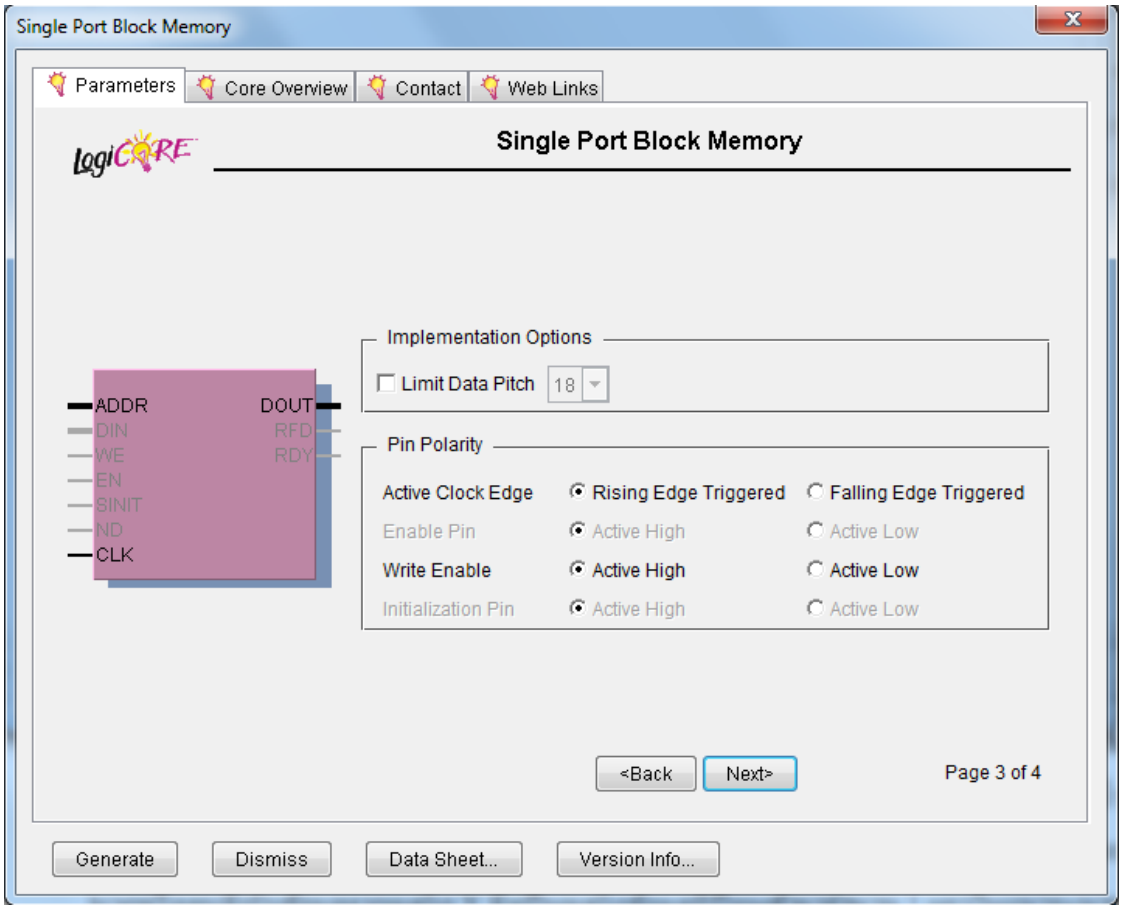
2. เมื่อเลือกไอพีคอร์ ที่มีชื่อว่า Single Port Block Memory แล้วจะพบหน้าต่างที่ให้กำหนดชื่อของคอมโพเน้น(Component) ซึ่งเป็นการตั้งชื่อกลุ่มวัตถุเครื่องมือสำหรับเรียกใช้งาน จากรูปเป็นการสร้างของกลุ่มค่าคงที่ชุดที่ 1 ดังนั้นจึงกำหนดให้เป็น row0 ในส่วนถัดมาเป็นการตั้งค่าเกี่ยวกับรูปแบบของพอร์ตที่ใช้ของหน่วยความจำ(Port Configuration) กำหนดให้เป็นอ่านค่าจากหน่วยความจำอย่างเดียว(Read Only) เพราะไม่มีการเขียนข้อมูลลงในหน่อยความจำแต่จะมีเฉพาะส่วนที่อ่านค่าเพื่อนำมาใช้ประมวลผล ในส่วนถัดมาเป็นการกำหนดขนาดของหน่วยความจำ (Memory Size) กำหนดให้ความกว้าง(Width) คือ จำนวนบิตของค่าคงที่แต่ละตัวซึ่งมีขนาดเท่ากับ 16 กำหนดความลึก(Depth) นั้นคือจำนวนที่อยู่ที่ใช้ภายในหน่วยความจำ(Address) นั่นคือจำนวนค่าคงที่ใน 1 แถว ซึ่งมีค่าเท่ากับ 8



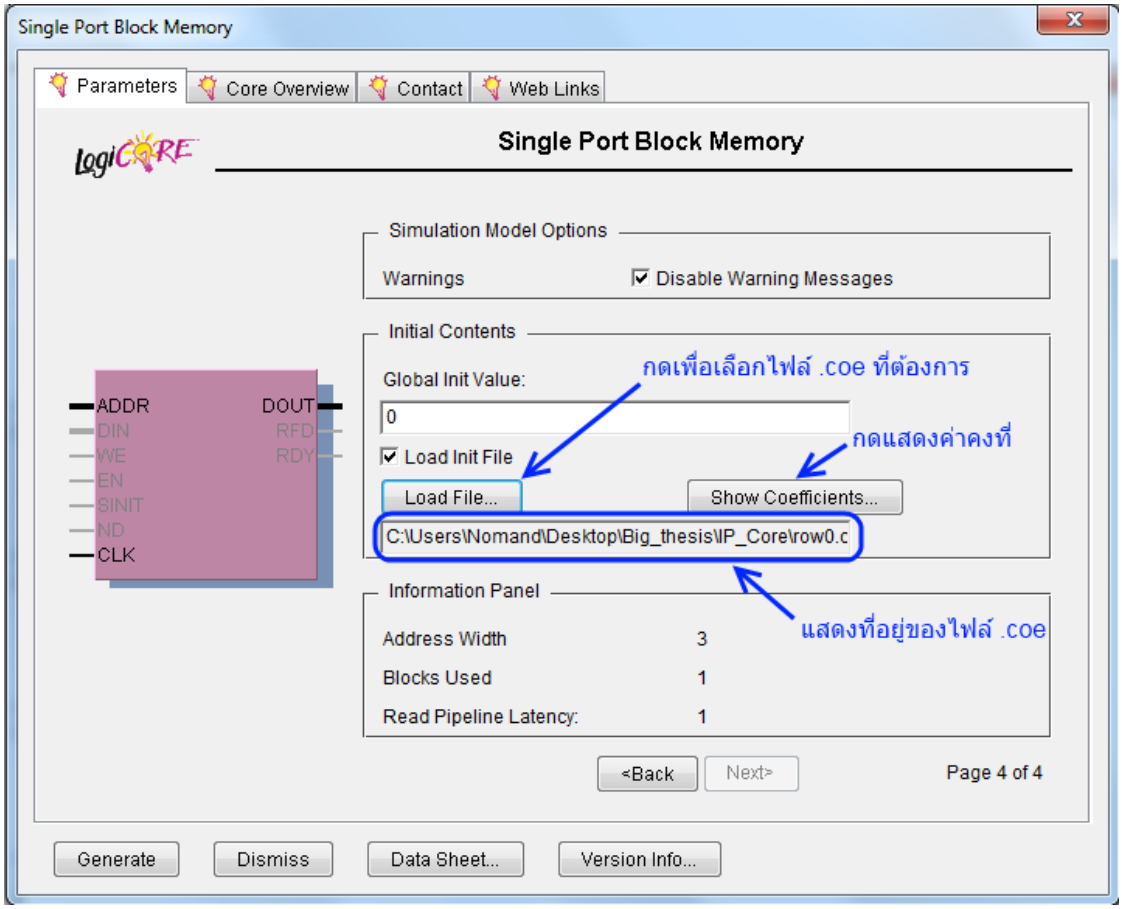
3. ในส่วนนี้เป็นการกำหนดตำแหน่งรูปแบบการจัดการพื้นที่ของหน่วยความจำและกำหนดรูปแบบการออกแบบไอพีคอร์ การกำหนดตำแหน่งรูปแบบการจัดการพื้นที่ของหน่วยความจำกำหนดให้จัดการตามขนาดของข้อมูลที่มีอยู่โดยไม่ต้องเลือกรูปแบบการจัดการแต่ให้โปรแกรมเป็นตัวตัดสินใจตามความเหมาะสมของพื้นที่ของหน่วยความจำ (Optimize For Area) ส่วนรูปแบบของไอพีคอร์ เป็นการกำหนดขาและโครงสร้างต่างๆ ของไอพีคอร์ ซึ่งในการทำวิจัยไม่จำเป็นต้องกำหนดขาเพิ่มเพราะในการใช้งานจะทำการป้อนที่อยู่ของข้อมูลภายในหน่วยความจำแล้วรับข้อมูลนั้นมาประมวลผล

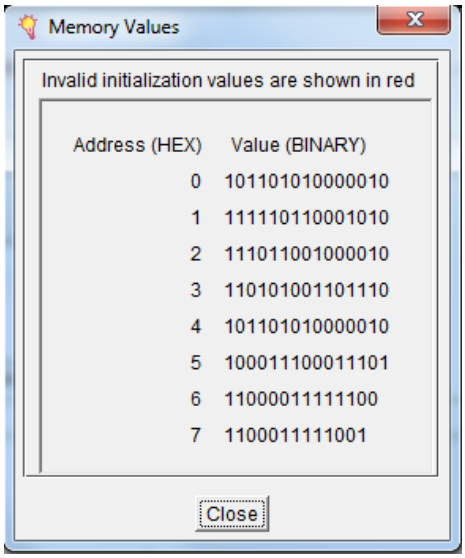


4. ในขั้นตอนถัดมาเป็นการเป็นการกำหนดช่วงสัญญาณนาฬิกาในการตรวจสอบข้อมูลที่อยู่ของข้อมูล โดยกำหนดให้มีการตรวจสอบทุกๆที่มีการป้อนสัญญาณนาฬิกาในช่วงของขอบขาขึ้น(Rising Edge Triggered)

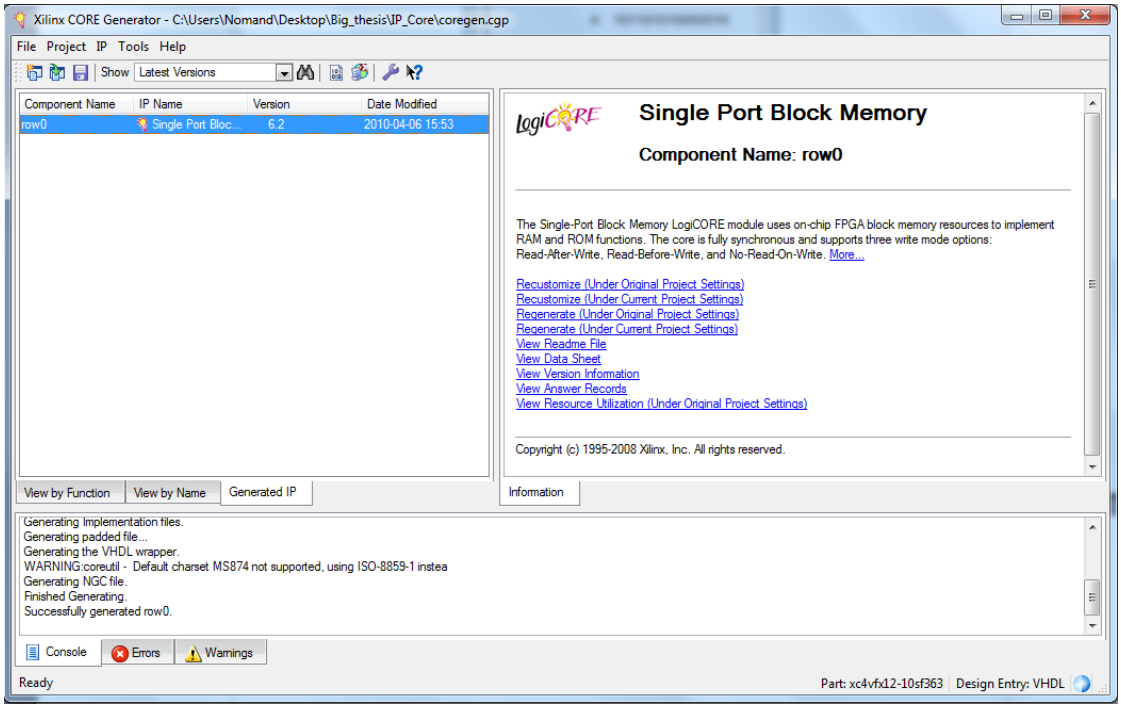


5. ในขั้นตอนถัดมาเป็นการโหลดค่าคงที่เข้าสู่ไอพีคอร์โดยการเลือก Load Init File และทำการเลือกที่อยู่ของไฟล์ .coe ซึ่งซอสโค้ดที่สร้างขึ้นตามรูปแบบของไอพีคอร์ เมื่อทำการโหลดไฟล์ .coe แล้วโปรแกรมจะทำการกำหนดค่าที่อยู่ในหน่วยความจำประจำค่าคงที่แต่ละค่า ซึ่งสามารถที่จะกดปุ่ม Show Coefficients เพื่อดูค่าที่อยู่และค่าคงที่ที่อยู่ในหน่วยความจำ


Single Port Block Memory
Parameters
Core Overview
Contact
Web Links
Single Port Block Memory
Simulation Model Options
Warnings
Disable Warning Messages
Initial Contents
Global Init Value:
0
Load Init File
Load File...
Show Coefficients...
C:\Users\Nomand\Desktop\Big_thesis\IP_Core\row0.c
กดเพื่อเลือกไฟล์ .coe ที่ต้องการ
กดแสดงค่าคงที่
แสดงที่อยู่ของไฟล์ .coe
ADDR
DIN
WE
EN
SINIT
ND
CLK
DOUT
RFD
RDY
Information Panel
Address Width 3
Blocks Used 1
Read Pipeline Latency: 1
<Back
Next>
Page 4 of 4
Generate
Dismiss
Data Sheet...
Version Info...


Memory Values
Invalid initialization values are shown in red
Address (HEX) Value (BINARY)
0 101101010000010
1 111110110001010
2 111011001000010
3 110101001101110
4 101101010000010
5 100011100011101
6 11000011111100
7 1100011111001
Close

6. ขั้นตอนสุดท้ายกดปุ่ม Generate เพื่อให้โปรแกรมทำการสร้างไอพีคอร์เพื่อนำไปใช้ในการออกแบบวงจร 2D-DCT ต่อไป



ในการสร้างไอพีคอร์หน่วยความจำของค่าคงที่ในชุดอื่นๆ สามารถสร้างได้โดยเริ่มทำจากขั้นตอนที่ 1 ถึงขั้นตอนที่ 6 แต่จะเปลี่ยนในส่วนของชื่อคอมโพเน้นและไฟล์ .coe ที่ใช้ในการโหลดเข้าสู่โปรแกรมสร้างไปพีคอร์ การเรียกใช้ไอพีคอร์ในการเขียนโปรแกรมออกแบบด้วยภาษา VHDL จะทำการนำไฟล์ทุกไฟล์ที่เกิดจากการสร้างด้วยโปรแกรมไปรวมกับไฟล์ที่ใช้ในการออกแบบวงจร 2D-DCT ในโฟลเดอร์(Folders) เดียวกันเพราะจะมีไฟล์ที่จำเป็นต่อการสังเคราะห์วงจรที่ออกแบบ จากนั้นจึงเรียกใช้คอมโพเน้นของหน่วยความจำตามวิธีการเรียกใช้วัตถุเครื่องมือของการเขียนภาษา VHDL

ภาคผนวก ค

การออกแบบวงจร 2D-DCT บรรยายด้วยภาษา VHDL

ในการออกแบบวงจร 2D-DCT ได้ออกแบบบนสถาปัตยกรรม FPGA โดยบรรยายด้วยภาษา VHDL ดังที่กล่าวไว้ในบทที่ 3 โดยในบทนี้จะกล่าวถึงซอสโค๊ดที่ใช้การออกแบบซึ่งในการเขียนโปรแกรมการออกแบบมีจำนวนมากจึงตัดนำมาเสนอเฉพาะส่วนที่สำคัญที่ใช้ในการออกแบบ รายละเอียดซอสโค๊ดแบ่งออกเป็นส่วนต่างๆ ดังต่อไปนี้

วงจร 1D-DCT ขั้นตอนที่ 1

กำหนดให้

X0,X1,X2,X3,X4,X5,X6,X7 คือ เป็นค่าของแถวเมตริกในแนวนอนของเมตริกข้อมูลที่ป้อนเข้ามา

clk คือ สัญญาณนาฬิกาที่ป้อนเข้าสู่วงจร

rst คือ สัญญาณสำหรับรีเซท(Reset) วงจร

mul_add1, mul_add2, mul_add3, mul_add4 คือ ช่องสัญญาณใช้รับค่าคงที่จากหน่วยความจำ

Output_add คือ สัญญาณผลลัพธ์ของวงจร 1D-DCT

ซอสโค๊ดวงจร 1D-DCT ขั้นตอนที่ 1

```
architecture Behavioral of DCT_Block is

signal ans_mul_add1,ans_mul_add2,ans_mul_add3,ans_mul_add4
       : signed(26 downto 0);
signal sum_add:signed(26 downto 0);
signal mul_s0,mul_s1,mul_s2,mul_s3:signed(10 downto 0);

begin

process(X0,X1,X2,X3,X4,X5,X6,X7,clk,rst)

       variable check:std_logic:='0';
       begin

          if(clk ='1' and clk'event)then
             if(rst = '1')then
             if check = '0' then
       mul_s0 <= signed(X0(10 downto 0)) + signed(X7(10 downto 0));
       mul_s1 <= signed(X1(10 downto 0)) + signed(X6(10 downto 0));
       mul_s2 <= signed(X2(10 downto 0)) + signed(X5(10 downto 0));
       mul_s3 <= signed(X3(10 downto 0)) + signed(X4(10 downto 0));

                   check := '1';
```

```
      else

mul_s0 <= signed(X0(10 downto 0)) - signed(X7(10 downto 0));
mul_s1 <= signed(X1(10 downto 0)) - signed(X6(10 downto 0));
mul_s2 <= signed(X2(10 downto 0)) - signed(X5(10 downto 0));
mul_s3 <= signed(X3(10 downto 0)) - signed(X4(10 downto 0));

            check := '0';

      end if;
      end if;
end if;

end process;
      ans_mul_add1 <= mul_s0*mul_add1;
      ans_mul_add2 <= mul_s1*mul_add2;
      ans_mul_add3 <= mul_s2*mul_add3;
      ans_mul_add4 <= mul_s3*mul_add4;

sum_add <= ans_mul_add1 + ans_mul_add2 + ans_mul_add3 + ans_mul_add4;

Output_add(10 downto 0) <= sum_add(26 downto 16);
```

จากซอสโค๊ดวงจร 1D-DCT ขั้นตอนที่ 1 ที่กล่าวไว้ข้างต้นเป็นเฉพาะส่วนการประมวลผลหาผลลัพธ์แค่ผลลัพธ์เดียว ดังนั้นหากต้องการผลลัพธ์ที่ได้จำนวน 8 ผลลัพธ์ในแนวตั้งเพื่อส่งไปประมวลผลในการคำนวณสมการ 1D-DCT ขั้นตอนที่ 2 จึงต้องสร้างบล๊อคการคำนวณวงจรสมการ 1D-DCT ขั้นตอนที่ 1 จำนวน 8 บล๊อค

วงจร 1D-DCT ขั้นตอนที่ 2

กำหนดให้

X0,X1,X2,X3,X4,X5,X6,X7 คือ เป็นค่าของผลลัพธ์ที่ส่งมาจากวงจร 1D-DCT ขั้นตอนที่ 1

clk คือ สัญญาณนาฬิกาที่ป้อนเข้าสู่วงจร

rst คือ สัญญาณสำหรับรีเซท(Reset) วงจร

Output_0,Output_1,Output_2,Output_3,

Output_4,Output_5,Output_6,Output_7 คือ สัญญาณผลลัพธ์ของวงจร 1D-DCT ขั้นตอนที่ 2

ซอสโค๊ดวงจร 1D-DCT ขั้นตอนที่ 2

```
architecture Behavioral of DCT_B2_2 is
type multiply is array (0 to 3) of signed(26 downto 0);
```

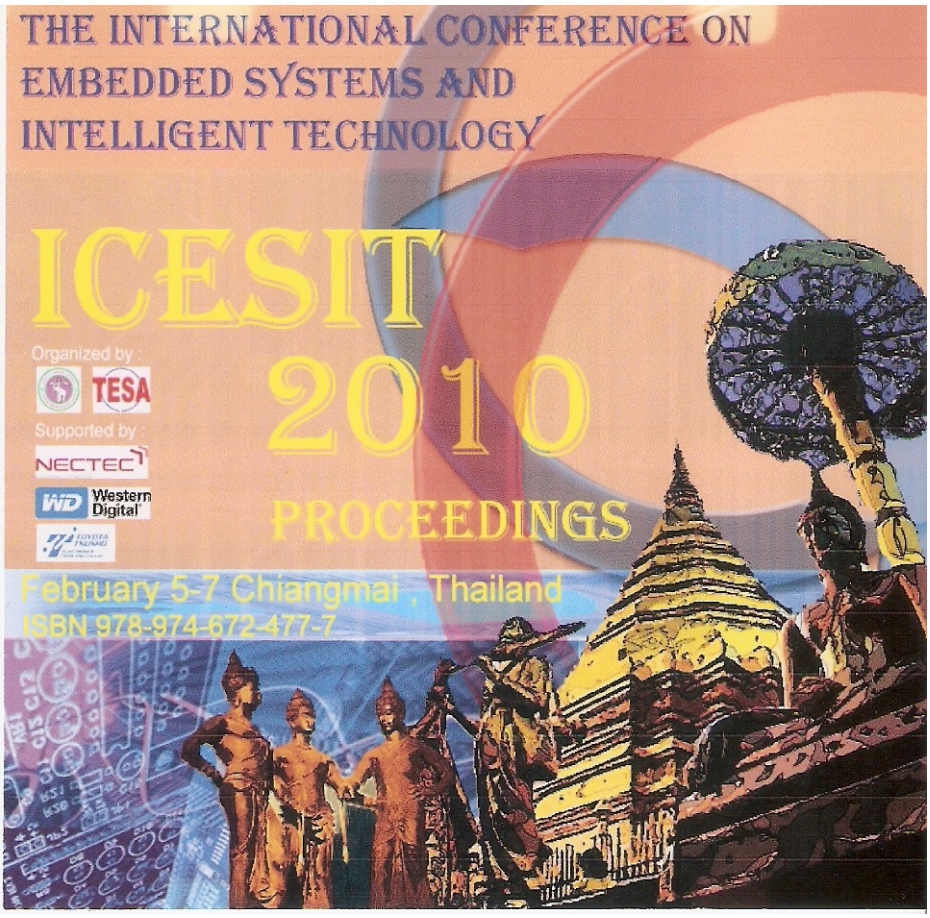
```
signal ans_mul_add0,ans_mul_add1,ans_mul_add2,ans_mul_add3,
       ans_mul_sub0,ans_mul_sub1,ans_mul_sub2,ans_mul_sub3:multiply;

type cons_dct is array (0 to 15) of signed(15 downto 0);

constant adder_part : cons_dct :=  (
       "0101101010000010","0111011001000010",
       "0101101010000010","0011000011111100",
       "0101101010000010","0011000011111100",
       "1010010101111110","1000100110111110",
       "0101101010000010","1100111100000100",
       "1010010101111110","0111011001000010",
       "0101101010000010","1000100110111110",
       "0101101010000010","1100111100000100" );

constant sub_part : cons_dct :=  (
       "0111110110001010","0110101001101110",
       "0100011100011101","0001100011111001",
       "0110101001101110","1110011100000111",
       "1000001001110110","1011100011100011",
       "0100011100011101","1000001001110110",
       "0001100011111001","0110101001101110",
       "0001100011111001","1011100011100011",
       "0110101001101110","1000001001110110");

signal add_s0,add_s1,add_s2,add_s3,
       sub_s0,sub_s1,sub_s2,sub_s3:signed(10 downto 0);
signal sum_add0,sum_add1,sum_add2,sum_add3,
       sum_sub0,sum_sub1,sum_sub2,sum_sub3:signed(26 downto 0);
signal out_add0,out_add1,out_add2,out_add3,
       out_sub0,out_sub1,out_sub2,out_sub3
                                 :std_logic_vector(26 downto 0);
begin

                  add_s0 <= X0 + X7;
                  add_s1 <= X1 + X6;
                  add_s2 <= X2 + X5;
                  add_s3 <= X3 + X4;

                  sub_s0 <= X0 - X7;
                  sub_s1 <= X1 - X6;
                  sub_s2 <= X2 - X5;
                  sub_s3 <= X3 - X4;


                  ans_mul_add0(0) <= add_s0*adder_part(0);
                  ans_mul_add0(1) <= add_s0*adder_part(1);
                  ans_mul_add0(2) <= add_s0*adder_part(2);
                  ans_mul_add0(3) <= add_s0*adder_part(3);

                  ans_mul_add1(0) <= add_s1*adder_part(4);
                  ans_mul_add1(1) <= add_s1*adder_part(5);
                  ans_mul_add1(2) <= add_s1*adder_part(6);
                  ans_mul_add1(3) <= add_s1*adder_part(7);

                  ans_mul_add2(0) <= add_s2*adder_part(8);
                  ans_mul_add2(1) <= add_s2*adder_part(9);
```

```
        ans_mul_add2(2) <= add_s2*adder_part(10);
        ans_mul_add2(3) <= add_s2*adder_part(11);

        ans_mul_add3(0) <= add_s3*adder_part(12);
        ans_mul_add3(1) <= add_s3*adder_part(13);
        ans_mul_add3(2) <= add_s3*adder_part(14);
        ans_mul_add3(3) <= add_s3*adder_part(15);

        ans_mul_sub0(0) <= sub_s0*sub_part(0);
        ans_mul_sub0(1) <= sub_s0*sub_part(1);
        ans_mul_sub0(2) <= sub_s0*sub_part(2);
        ans_mul_sub0(3) <= sub_s0*sub_part(3);

        ans_mul_sub1(0) <= sub_s1*sub_part(4);
        ans_mul_sub1(1) <= sub_s1*sub_part(5);
        ans_mul_sub1(2) <= sub_s1*sub_part(6);
        ans_mul_sub1(3) <= sub_s1*sub_part(7);

        ans_mul_sub2(0) <= sub_s2*sub_part(8);
        ans_mul_sub2(1) <= sub_s2*sub_part(9);
        ans_mul_sub2(2) <= sub_s2*sub_part(10);
        ans_mul_sub2(3) <= sub_s2*sub_part(11);

        ans_mul_sub3(0) <= sub_s3*sub_part(12);
        ans_mul_sub3(1) <= sub_s3*sub_part(13);
        ans_mul_sub3(2) <= sub_s3*sub_part(14);
        ans_mul_sub3(3) <= sub_s3*sub_part(15);



sum_add0 <= ans_mul_add0(0) + ans_mul_add1(0) +
            ans_mul_add2(0) + ans_mul_add3(0);
sum_add1 <= ans_mul_add0(1) + ans_mul_add1(1) +
            ans_mul_add2(1) + ans_mul_add3(1);
sum_add2 <= ans_mul_add0(2) + ans_mul_add1(2) +
            ans_mul_add2(2) + ans_mul_add3(2);
sum_add3 <= ans_mul_add0(3) + ans_mul_add1(3) +
            ans_mul_add2(3) + ans_mul_add3(3);

sum_sub0 <= ans_mul_sub0(0) + ans_mul_sub1(0) +
            ans_mul_sub2(0) + ans_mul_sub3(0);
sum_sub1 <= ans_mul_sub0(1) + ans_mul_sub1(1) +
            ans_mul_sub2(1) + ans_mul_sub3(1);
sum_sub2 <= ans_mul_sub0(2) + ans_mul_sub1(2) +
            ans_mul_sub2(2) + ans_mul_sub3(2);
sum_sub3 <= ans_mul_sub0(3) + ans_mul_sub1(3) +
            ans_mul_sub2(3) + ans_mul_sub3(3);
out_add0 <= std_logic_vector(sum_add0);
out_add1 <= std_logic_vector(sum_add1);
out_add2 <= std_logic_vector(sum_add2);
out_add3 <= std_logic_vector(sum_add3);
out_sub0 <= std_logic_vector(sum_sub0);
out_sub1 <= std_logic_vector(sum_sub1);
out_sub2 <= std_logic_vector(sum_sub2);
out_sub3 <= std_logic_vector(sum_sub3);
```

```
                Output_0(10 downto 0) <= out_add0(26 downto 16);
                Output_0(11) <= out_add0(26);
                Output_0(12) <= out_add0(26);
                Output_0(13) <= out_add0(26);
                Output_0(14) <= out_add0(26);
                Output_0(15) <= out_add0(26);

                Output_2(10 downto 0) <= out_add1(26 downto 16);
                Output_2(11) <= out_add1(26);
                Output_2(12) <= out_add1(26);
                Output_2(13) <= out_add1(26);
                Output_2(14) <= out_add1(26);
                Output_2(15) <= out_add1(26);

                Output_4(10 downto 0) <= out_add2(26 downto 16);
                Output_4(11) <= out_add2(26);
                Output_4(12) <= out_add2(26);
                Output_4(13) <= out_add2(26);
                Output_4(14) <= out_add2(26);
                Output_4(15) <= out_add2(26);

                Output_6(10 downto 0) <= out_add3(26 downto 16);
                Output_6(11) <= out_add3(26);
                Output_6(12) <= out_add3(26);
                Output_6(13) <= out_add3(26);
                Output_6(14) <= out_add3(26);
                Output_6(15) <= out_add3(26);

                Output_1(10 downto 0) <= out_sub0(26 downto 16);
                Output_1(11) <= out_sub0(26);
                Output_1(12) <= out_sub0(26);
                Output_1(13) <= out_sub0(26);
                Output_1(14) <= out_sub0(26);
                Output_1(15) <= out_sub0(26);

                Output_3(10 downto 0) <= out_sub1(26 downto 16);
                Output_3(11) <= out_sub1(26);
                Output_3(12) <= out_sub1(26);
                Output_3(13) <= out_sub1(26);
                Output_3(14) <= out_sub1(26);
                Output_3(15) <= out_sub1(26);


                Output_5(10 downto 0) <= out_sub2(26 downto 16);
                Output_5(11) <= out_sub2(26);
                Output_5(12) <= out_sub2(26);
                Output_5(13) <= out_sub2(26);
                Output_5(14) <= out_sub2(26);
                Output_5(15) <= out_sub2(26);

                Output_7(10 downto 0) <= out_sub3(26 downto 16);
                Output_7(11) <= out_sub3(26);
                Output_7(12) <= out_sub3(26);
                Output_7(13) <= out_sub3(26);
                Output_7(14) <= out_sub3(26);
                Output_7(15) <= out_sub3(26);

end Behavioral;
```

ภาคผนวก ง

ผลงานตีพิมพ์เผยแพร่จากวิทยานิพนธ์


THE INTERNATIONAL CONFERENCE ON
EMBEDDED SYSTEMS AND
INTELLIGENT TECHNOLOGY
ICESIT
2010
PROCEEDINGS
Organized by :
TESA
Supported by :
NECTEC
Western Digital
February 5-7 Chiangmai , Thailand
ISBN 978-974-672-477-7

High speed 2D-DCT for JPEG compression base on FPGAs

Naras Kwantong[1] , Wannarat Suntiamorntut[2] , Anant Choksuriwong[3]

Department of Computer Engineering , Prince of Songkla University
15 Kanjanavanit Road Hadyai District Songkha province 90110 , Thailand
(narasbig@hotmail.com,wannarat@coe.psu.ac.th,ant@coe.psu.ac.th)

***Abstract*— This paper presents the design and speed improvement of Two-Dimensional Discrete Cosine Transform (2D-DCT) for JPEG compression using FPGA. The design is the most intensive computation core and is the critical path in JPEG compression. The parallelism has been employed to improve the speed aiming to support real-time application. The preliminary result shows that our design has slides utilization 47%, power consumption 0.257 W running at 290 MHz based on Xilinx FPGA XC4VFX12.**

***Keywords*— 2D-DCT, FPGA, JPEG, compression, parallel processing**

I. INTRODUCTION

FPGA has been used to implement DSP algorithms. Since FPGA can be reconfigured in hardware, the critical path in some DSP algorithms will be mapped onto FPGA. Therefore, FPGA can be used as a co-processing to accelerate the computational rates.

JPEG (Joint Photographic Experts Group)[1,2] algorithm was proposed to compress the image data to save the storage in the system. Discrete Cosine Transform (DCT) is used in the JPEG which is the most intensive computation unit. Normally, JPEG compression has five steps, colour space conversion, downsampling, 2D-DCT, quantization and entropy coding.

In this paper presents how to design the JPEG image compression circuit based on FPGA architecture. We expect to get a better speed rather than running the JPEG algorithm on PC. We use both pipelining and parallel design technique to increase the performance.

This paper describes JPEG algorithm in section 2. The implementation of 2D-DCT used 1D-DCT is discussed in section 3. The result and conclusion present in section 4 and 5, respectively.

II. JPEG ALGORITHM

JPEG (Joint Photographic Experts Group) is the image compression algorithm that lossless resolution. It is the type of image for storage and the most popular use in internet system because high resolution, small size and use with various of bit depth level in colour image. JPEG image use discrete cosine transform(DCT) technique change data from time domain to frequency domain. It represent amplitude of frequency by coefficient that can reduce significance for decrease size of image and resolution. Figure 1 shown 5 step JPEG algorithm [3,4].

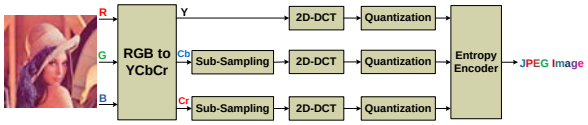


Figure 1 : JPEG algorithm

- Colour Space Conversion

This is the first step in JPEG algorithm. It change RGB to YCbCr that change same as acknowledgement of human optical nerve This step change RGB to YCbCr

$$Y = 0.299R + 0.587G + 0.114B$$
$$Cb = 128 - 0.168736R + 0.331264G + 0.5B$$
$$Cr = 128 + 0.5R - 0.418688G - 0.081312B$$

where y is luma, cb is blue chroma, cr is red chroma, R is red colour pixel, G is green colour pixel, B is blue colour pixel

- Sub-sampling

In colour image system has 2 component. There are luminance and chrominance. Normally, human eyesight cannot split all of chrominance. So, we can reduce data in chrominance but human eyesight can split luminance then we cannot reduce data in luminance [5,6].

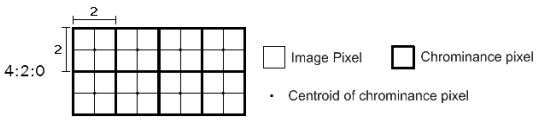


Figure 2 : Sub-sampling (4:2:0)

In figure 2 is sub-sampling 4:2:0 processing that split data in 2x2 pixel and find the centroid of chrominance pixel for represent in that 4 pixel

- 2D Discrete Cosine Transform (2D-DCT)

The 2D-DCT is technique use in image compression that change data from time domain to frequency domain. It split data same as power number 2,4,8,16. If spite in large block that is high performance for compress data but low resolution of image. So, the standard of JPEG split in 8x8 pixel because the human eyesight has limit of split the difference image and high performance for compress data [7].

2D-DCT equation :

$$F(u,v) = \propto(u) \propto(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cos\left[\frac{(2x+1)u\pi}{2N}\right] \cos\left[\frac{(2y+1)v\pi}{2N}\right]$$

$$u, v = 0,1,2,\ldots,N-1$$

$$\propto(u) = \begin{cases} \sqrt{\frac{1}{N}} & \text{for u = 0} \\ \sqrt{\frac{2}{N}} & \text{for u = 1 to N-1} \end{cases}$$

Where (x,y) is the position of data input matrix and (u,v) is the position of output matrix

- Quantization

This step allow lose some data in process because it increase the number of same data in 8x8 pixel. The human eyesight good detect low frequency coefficient of image. Data that transfer from 2D-DCT pixel low position normalize by low frequency coefficient and high position pixel normalize by high frequency coefficient. The result around high position pixel increase number of zero. So, it easy for encoder in next step.

Quantization equation :

$$QuantizationValue(i,j) = \frac{DCT(i,j)}{QuantizationMatrix(i,j)}$$

From quantization equation the DCT value divide by quantization matrix that divide value in the same matrix position

- Entropy Encoder

In this step order the data that transfer from quantization in zig-zag pattern the data arrange in row and the zero number in high position send to back of row that is easy for encoder. Huffman code and runlength encoder is the technique use for entropy encoder. The group of zero number in high position we can reject and represent by code. For the other position encode by Huffman standard.

III. Implement 2D-DCT use 1D-DCT

For design 2D-DCT equation process in digital circuit we process use 1D-DCT circuit 2 time. The first send data process in row position and the second send to 1D-DCT process in column position [8]. It is shown in figure 3.

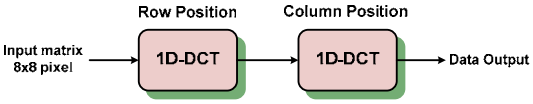


Figure 3 : 2D-DCT processing use 1D-DCT

1D-DCT equation :

$$C = K \bullet \cos\frac{(2 \cdot col + 1) \bullet row \bullet \pi}{2 \bullet M}$$

$$K = \sqrt{\frac{1}{N}} \quad \text{for} \quad \text{row} = 0$$

$$K = \sqrt{\frac{2}{N}} \quad \text{for} \quad \text{row} \neq 0$$

$$C^T = K \bullet \cos\frac{(2 \cdot row + 1) \bullet col \bullet \pi}{2 \bullet N}$$

$$K = \sqrt{\frac{1}{M}} \quad \text{for} \quad \text{col} = 0$$

$$K = \sqrt{\frac{2}{M}} \quad \text{for} \quad \text{col} \neq 0$$

Where M is number of column.
N is number of row.

From 1D-DCT equation we assign the value of matrix position and we can get the result the 2 constant matrix. There are C constant matrix and C^t constant transpose matrix.

Constant 1D-DCT matrix

$$C = \begin{bmatrix} 23170 & 23170 & 23170 & 23170 & 23170 & 23170 & 23170 & 23170 \\ 32138 & 27246 & 18205 & 6393 & -6393 & -18205 & -27246 & -32138 \\ 30274 & 12540 & -12540 & -30274 & -30274 & -12540 & 12540 & 30274 \\ 27246 & -6393 & -32138 & -18205 & 18205 & 32138 & 6393 & -27246 \\ 23170 & -23170 & -23170 & 23170 & 23170 & -23170 & -23170 & 23170 \\ 18205 & -32138 & 6393 & 27246 & -27246 & -6393 & 32138 & -18205 \\ 12540 & -30274 & 30274 & -12540 & -12540 & 30274 & -30274 & 12540 \\ 6393 & -18205 & 27246 & -32138 & 32138 & -27246 & 18205 & -6393 \end{bmatrix}$$

$$C^T = \begin{bmatrix} 23170 & 32138 & 30274 & 27246 & 32170 & 18205 & 12540 & 6393 \\ 23170 & 27246 & 12540 & -6393 & -23170 & -32138 & -30274 & -18205 \\ 23170 & 18205 & -12540 & -32138 & -23170 & 6393 & 30274 & 27246 \\ 23170 & 6393 & -30274 & -18205 & 23170 & 27246 & -12540 & -32138 \\ 23170 & -6393 & -30274 & 18205 & 23170 & -27246 & -12540 & 32138 \\ 23170 & -18205 & -12540 & 32138 & -23170 & -6393 & 30274 & -27246 \\ 23170 & -27246 & 12540 & 6393 & -23170 & 32138 & -30274 & 18205 \\ 23170 & -32138 & 30274 & -27246 & 23170 & -18205 & 12540 & -6393 \end{bmatrix}$$

We design the 2D-DCT circuit use 1D-DCT as follow equation

$$Y = C \bullet X \bullet C^T \qquad (1)$$

Where Y is the output of 2D-DCT. X is the input data.

- 1D-DCT first block look data in row position

Design this block for 2 matrix 8x8 multiply and send the result to next block. This block flow data matrix input in row position and multiply with 1D-DCT constant transpose matrix (C^t) that as follow equation

From equation 1

$$Y = C \bullet Z \qquad (2)$$

Where

$$Z = X \bullet C^T \qquad (3)$$

From equation 2 we design circuit input data (X) multiply with 1D-DCT constant transpose matrix

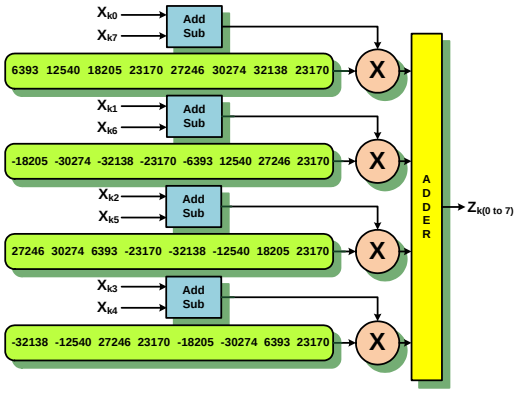


Figure 4 : 1D-DCT first block output 1 value

In figure 4 is 1D-DCT first block for one output value that design get the input 8 value in one row position from 8x8 matrix data input send to add or sub block because in 1D-DCT constant transpose matrix each column we can group one from 2 position in same column there are (0,7),(1,6),(2,5),(3,4). There are same number but even column number group in positive sign and odd column number group in negative sign then we split the constant value for multiply in next step and use sign for input value for add or sub in first step.

From the circuit design in figure 4 get one output value but in 1D-DCT second block process use 8 value input that is the output from 1D-DCT first block in column position. So, we design 1D-DCT first block circuit same as figure 4 use parallel processing in 8 circuit. Each circuit get 8 input value in row position from 8x8 matrix data input and the output we can get 8 value in column position for transfer to 1D-DCT second block follow as figure 5.

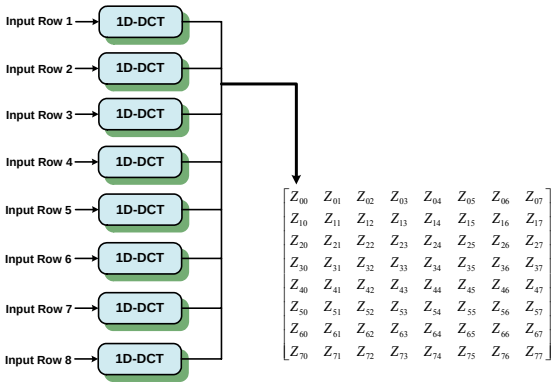


Figure 5 : 1D-DCT first block output 8 value

- 1D-DCT second block look data in column position

Matrix input value pass 1D-DCT first block one clock get the result one column then in 1D-DCT second block we design get input one column from first block multiply with all of column of 1D-DCT constant matrix in one clock follow as equation 2 and in figure 6 shown the circuit design.

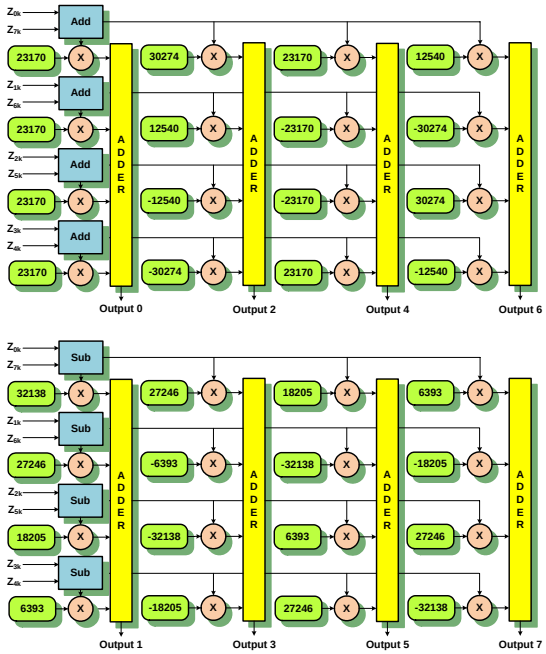


Figure 6 : 1D-DCT second block

IV. Test and Result

Each 8x8 pixel block has brought into this 2D-DCT at a time. One matrix data input is computed to get the result for each column in every clock. Therefore, we use eight clocks in total get the output one matrix.

The design is implemented by VHDL and tested under the ISE Simulator (Xilinx 10.1). We use Virtex4 family and device number XC4VFX12 to test our 2D-DCT. The matrix 8x8 pixel block is shown below. All number is represented by sign number [9].

For matrix 8x8 that is input value follow as

$$\begin{bmatrix} 58 & 45 & 29 & 27 & 24 & 19 & 17 & 20 \\ 62 & 52 & 42 & 41 & 38 & 30 & 22 & 18 \\ 48 & 47 & 49 & 44 & 40 & 36 & 31 & 25 \\ 59 & 78 & 49 & 32 & 28 & 31 & 31 & 31 \\ 98 & 138 & 116 & 78 & 39 & 24 & 25 & 27 \\ 115 & 160 & 143 & 97 & 48 & 27 & 24 & 21 \\ 99 & 137 & 127 & 84 & 42 & 25 & 24 & 20 \\ 74 & 95 & 82 & 67 & 40 & 25 & 25 & 19 \end{bmatrix}$$

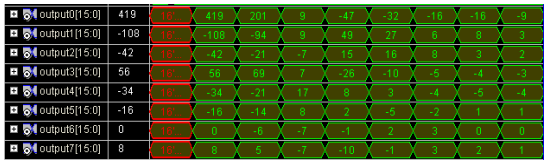


Figure 7 : shows the output of 2D-DCT process. The output is generated as followed

In Figure 7 shown the output of 2D-DCT process. The output from simulation follow in matrix as

$$\begin{bmatrix} 419 & 201 & 9 & -47 & -32 & -16 & -16 & -9 \\ -108 & -94 & 9 & 49 & 27 & 6 & 8 & 3 \\ -42 & -21 & -7 & 15 & 16 & 8 & 3 & 2 \\ 56 & 69 & 7 & -26 & -10 & -5 & -4 & -3 \\ -34 & -21 & 17 & 8 & 3 & -4 & -5 & -4 \\ -16 & -14 & 8 & 2 & -5 & -2 & 1 & 1 \\ 0 & -6 & -7 & -1 & 2 & 3 & 0 & 0 \\ 8 & 5 & -7 & -10 & -1 & 3 & 2 & 1 \end{bmatrix}$$

Table 1 : Device Utilization Summary

Logic Utilization	Used	Available	Utilization
Number of Slices	2618	5472	47%
Number of Slice Flip Flops	43	10944	1%
Number of 4 input LUTs	5035	10944	46%
Number of FIFO16/RAMB16s	4	36	11%
Total Power	0.257 W		

Table 1 shows the resource usage of this design. The circuit uses 47% of slices and the power consumes 0.257W.

The critical path in circuit is 3.334ns the maximum frequency is 299.985MHz. The arithmetic such as ROMs, Multipliers, Adders/Subtractors, Comparators use the macro circuit design by FPGA architecture.

Table 2 : Compare with literature review

Author	Device	Speed Grade	Package	Clock rate (MHz)
K. Naras	XC4VFX12	-10	SF363	299.98
L. Pillai	XC2V250	-6	FG456	182.75

The comparison is presented in table 2 our design gives a good performance having twice speed better than the previous design while the area is only 30% larger than the reference design.

V. Conclusion and Future Work

This paper presents the design of 2D-DCT Circuit that implement for improve speed JPEG Compression. The design base on FPGA architecture using parallel processing technique.

For 2D-DCT circuit the output from test bench simulation in figure 7 we can get the

output 1 matrix DCT 64 value finish in 8 clock and correct follow as 2D-DCT equation.

Colour Space Conversion, Subsampling, Quantization and Entropy Encoder are the future works.

VI. Reference

[1] The International Telegraph and Telephone Consultative Committee (CCITT). "information Technology – Digital Compression and Coding of Continuous-Tone Still Images Requirements and Guidelines ". Rec. T.81, 1992.

[2] W. Pennebaker, J. Mitchell. *JPEG Still Imag e Data Compression Standard,* Van Nostrand Reinhold, USA, 1992.

[3] Sekson Timakul , *Modification of JPEG Algorithm for FPGA Implementation* , M.eng Thesis ,King Mongkut Institute of Technology Ladkrabang , 2004.

[4] David Bell , "*JPEG Tutor"*, The University of Cambridge, June 8, 2007. Available: http://www.jpegtutor.co.uk. [Accessed September 29, 2008]

[5] Poynton, Charles."Chrominance Subsampling in Digital Images " July 2008. [Online]. Available: www.poynton.com/PDFs/ Chroma_subsampling_notation.pdf. [Accessed: September 20, 2008]

[6] Kerr, Douglas A."*Chrominance Subsampling in Digital Images*" June 2008.[Online]. Available:http://doug.kerr. home.att.net/pumpkin/Subsampling.pdf. [Accessed: September 20, 2008]

[7] Lienhart, G., Männer, R., Noffz, K. H., and Lay. *R. 2001. An FPGA-based video compressor for H.263compatible bit streams*, In Proceedings of the 2001 ACM/SIGDA Ninth international Symposium on Field Programmable Gate Arrays ,pp. 207-212 ,2001.

[8] Latha Pillai , *Video Compression Using DCT* , Appl. Note : Xapp610 , Xilinx , 2005.

[9] John Miano, *Compressed Image File Formats*, Addison Wesley Longman Inc , 1999, pp.87.

ประวัติผู้เขียน

ชื่อ สกุล นายนเรศ ขวัญทอง

รหัสประจำตัวนักศึกษา 5110120078

วุฒิการศึกษา

วุฒิ	ชื่อสถาบัน	ปีที่สำเร็จการศึกษา
วิศวกรรมศาสตรบัณฑิต (วิศวกรรมคอมพิวเตอร์)	มหาวิทยาลัยสงขลานครินทร์	2550

ทุนการศึกษา (ที่ได้รับในระหว่างการศึกษา)

ทุนผู้ช่วยวิจัยคณะวิศวกรรมศาสตร์ มหาวิทยาลัยสงขลานครินทร์

การตีพิมพ์เผยแพร่ผลงาน

N.Kwantong , W.Suntiamorntut and A.Choksuriwong , " **High speed 2D-DCT for JPEG Compression base on FPGAs** ", In *Proceedings of The 2010 International Conference on Embedded Systems and Intelligent Technology*, 5^{th}-7^{th} February 2010.