วิทยานิพนธ์

การออกแบบและสร้างการบีบอัดสัญญาณภาพวิดีโอบนอุปกรณ์ FPGA

DESIGN AND IMPLEMENTATION OF VIDEO COMPRESSION ON FPGA

นายเฉลิมพล สงพราหมณ์

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์

พ.ศ. 2550

ใบรับรองวิทยานิพนธ์

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์

วิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมไฟฟ้า)

ปริญญา

วิศวกรรมไฟฟ้า	วิศวกรรมไฟฟ้า
สาขา	**ภาควิชา**

เรื่อง การออกแบบและสร้างการบีบอัดสัญญาณภาพวิดีโอบนอุปกรณ์ FPGA

Design and Implementation of Video Compression on FPGA

นามผู้วิจัย นายเฉลิมพล สงพราหมณ์

ได้พิจารณาเห็นชอบโดย

ประธานกรรมการ .....

(ผู้ช่วยศาสตราจารย์ชูเกียรติ การะเกตุ, Ph.D.)

กรรมการ .....

(รองศาสตราจารย์ณัฏฐกา หอมทรัพย์, Ph.D.)

กรรมการ .....

(อาจารย์จันทนี รุ่งเรืองพิทยากุล, D.Sc.)

หัวหน้าภาควิชา .....

(รองศาสตราจารย์มงคล รักษาพัชรวงศ์ , Ph.D.)

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์รับรองแล้ว

.....

(รองศาสตราจารย์วินัย อาจคงหาญ, M.A.)

คณบดีบัณฑิตวิทยาลัย

วันที่ ..... **เดือน** มีนาคม **พ.ศ.** 2550

วิทยานิพนธ์

เรื่อง

การออกแบบและสร้างการบีบอัดสัญญาณภาพวิดีโอบนอุปกรณ์ FPGA

Design and Implementation of Video Compression on FPGA

โดย

นายเฉลิมพล สงพราหมณ์

เสนอ

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์
เพื่อความสมบูรณ์แห่งปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมไฟฟ้า)
พ.ศ. 2550

เฉลิมพล สงพราหมณ์ 2550: การออกแบบและสร้างการบีบอัดสัญญาณภาพวิดีโอบนอุปกรณ์ FPGA ปริญญาวิศวกรรมศาสตรมหาบัณฑิต สาขาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า ประธานกรรมการที่ปรึกษา: ผู้ช่วยศาสตราจารย์ชูเกียรติ การะเกตุ, Ph.D. 98 หน้า

วิทยานิพนธ์นี้นำเสนอการออกแบบและสร้างอุปกรณ์ลดขนาดสัญญาณภาพบนชิพ FPGA ทั้งนี้เนื่องมาจากการที่สัญญาณภาพ Video มีขนาดข้อมูลที่มากทำให้การเก็บข้อมูลนั้นต้องการพื้นที่ในการจัดเก็บมากด้วย จึงทำให้ไม่มีความคล่องตัว ในการที่จะนำข้อมูลนั้นไป Presentation หรือส่ง E-mail โดยเฉพาะอย่างยิ่ง ในกรณีที่เป็นการติดต่อสื่อสารด้วยสัญญาณภาพ Video ด้วยแล้วจำเป็นต้องใช้ช่องสัญญาณที่กว้างมาก ๆ การออกแบบการลดขนาดสัญญาณภาพจึงช่วยให้การติดต่อสื่อสารมีความต้องการใช้ช่องสัญญาณที่ลดลง สำหรับการติดต่อสื่อสารข้อมูลความเร็วต่ำ เช่น โทรภาพ (Video Telephony) โดยการออกแบบได้ใช้วิธี Haar Wavelet Transform ซึ่งเป็นทฤษฎีทางคณิตศาสตร์ที่มีเนื้อหาของการแปลงรูปภาพ และนำมาเข้ารหัส Huffman ที่จะช่วยในการลดขนาด Entropy ของสัญญาณภาพ

การออกแบบดังกล่าวได้ใช้ภาษา VHDL โปรแกรมลงในชิพ FPGA เนื่องจาก ชิพ FPGA เป็นชิพในลักษณะของ Programmable Device ที่สามารถโปรแกรมให้เป็นวงจรดิจิตอลใด ๆ ก็ได้ ซึ่งในปัจจุบันความจุเกตภายในตัวชิพ FPGA นั้นได้เพิ่มระดับจากไม่กี่พันเกตไปจนถึงล้านเกต ดังนั้นจึงรองรับวงจรทางดิจิตอลที่สลับซับซ้อนได้ รวมถึงมีความเร็วสูงในการทำงานเมื่อเทียบกับไมโครโปรเซสเซอร์ทั่วไป โดยผลจากการทดลองบีบอัดไฟล์ข้อมูลภาพขนาด 91 Kbytes สามารถลดขนาดข้อมูลลงได้ถึง 65% มีประสิทธิภาพในการบีบอัดไฟล์ข้อมูล (Compression Ratio) เท่ากับ 2.94 ใช้เวลาเพียง 0.172 วินาที โดยจะเร็วกว่าเมื่อเปรียบเทียบกับโปรแกรมใน PC ซึ่งใช้เวลา 0.6 วินาที และในปัจจุบันการออกแบบชิพ FPGA นั้นเป็นที่นิยมทำกันและมีแนวโน้มที่จะนำมาใช้งานกันมากขึ้นเรื่อย ๆ

_____ _____ ___ / ___ / ___

ลายมือชื่อนิสิต ลายมือชื่อประธานกรรมการ

Chaloemphol Songpharm 2007: Design and Implementation of Video Compression on FPGA. Master of Engineering (Electrical Engineering), Major Field: Electrical Engineering, Department of Electrical Engineering. Thesis Advisor: Assistant Professor Chugiat Garagate, Ph.D. 98 pages.

This thesis describe the design and implementation of Video Compression on FPGA. Due to the data size of video signal is high, which wide require large data storage. That is not convenient for data presentation or e-mail. Especially in Video signal communication, it was requires wide-bandwidth. So the design of video compression is able to decrease the demand of channel in narrow band data communication such as Video Telephony. Haar Wavelet Transform which is mathematical theory that is use for Huffman encoding will use for entropy of video signal compression.

In this thesis, VHDL is used for programming FPGA. FPGA is a programmable device that can be programed as any digital circuit. Nowadays the gate capacity of FPGA developed from thousands to millions available to support a complicated digital circuit. Also the speed which compare to a microprocessor, is also higher. From the experiment, in 91 Kbytes data compression, this method is 65% successful, the compression ratio is 2.94 and takes only 0.172 seconds that is quicker than PC Program which takes 0.6 seconds. Nowadays, the design of FPGA is populared and widely used for user in the future.

_____ _____ ___ / ___ / ___

Student's signature Thesis Advisor's signature

กิตติกรรมประกาศ

วิทยานิพนธ์ครั้งนี้ลุล่วงไปได้ด้วยดี ด้วยความช่วยเหลือและการสนับสนุนจากบุคคลหลาย ๆ ท่านด้วยกัน ซึ่งผู้เขียนขอขอบพระคุณทุก ๆ ท่านดังต่อไปนี้

ขอขอบพระคุณ ผศ.ดร.ชูเกียรติ การะเกตุ, รศ.ดร. ณัฏฐกา หอมทรัพย์ และ อ.ดร.จันทนี รุ่งเรืองพิทยากุล ผู้ซึ่งคอยให้คำปรึกษา ความรู้ ความเข้าใจ และ กำลังใจ ตลอดระยะเวลาในการศึกษา ผู้เขียนรู้สึกซาบซึ้งในความเมตตาของทุกท่านเป็นอย่างสูง

ขอขอบคุณ เพื่อน ๆ ที่เรียนปริญญาโทด้วยกันที่ช่วยให้คำแนะนำและให้กำลังใจด้วยความเป็นห่วงตลอดเวลา

สุดท้ายนี้ขอขอบคุณภรรยาที่แสนดี ที่อดทน และเป็นกำลังใจที่ดีสม่ำเสมอตลอดมา และลูกต้นน้ำที่เป็นกำลังใจให้คุณพ่อตั้งแต่อยู่ในครรภ์

เฉลิมพล สงพราหมณ์

มีนาคม 2550

สารบัญ

สารบัญตาราง

สารบัญภาพ

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

คำอธิบายสัญลักษณ์และคำย่อ

FPGA	=	Field Programmable Gate Array
VHDL	=	VHSIC Hardware Description Language (VHSIC: Very High Speed Integrated Circuit)
HDL	=	Hardware Description Language
DoD	=	Department of Defense
ASIC	=	Application-Specific Integrated Circuit
SSI	=	Small-Scale Integration
MSI	=	Medium-Scale Integration
LSI	=	Large-Scale Integration
VLSI	=	Very Large-Scale Integration
ULSI	=	Ultra large Scale Integration
PROM	=	Programmable Read Only Memory
PLA	=	Programmable Logic Array
PLD	=	Programmable Logic Device
EPLD	=	Erasable Programmable Logic Device
ITAR	=	United States International Traffic and Arms Regulations

การออกแบบและสร้างการบีบอัดสัญญาณภาพวิดีโอบนอุปกรณ์ FPGA

Design and Implementation of Video Compression on FPGA

คำนำ

หากเรากล่าวถึง Video หลายคนคงนึกถึงภาพยนตร์ที่อยู่ในตลับเทป Cassette หรือที่จัดเก็บในแผ่น CD ROM เท่านั้น แต่สัญญาณภาพ Video นั้นไม่ได้ถูกจำกัดเพียงแค่นั้น ในปัจจุบันสัญญาณภาพ Video ได้เข้ามีบทบาทต่อการทำงานของเรามากขึ้น ไม่ว่าจะเป็นการจัดทำ Presentation ในรูปแบบของ multimedia การส่ง E-mail ทั้งนี้เป็นเพราะการสื่อสัญญาณด้วยภาพ Video จะสามารถให้ความเข้าใจที่ชัดเจนแก่ผู้ที่ได้รับเป็นอย่างมาก นอกจากนี้แล้วยังมีการใช้สัญญาณภาพ Video เพื่อใช้ในการติดต่อสื่อสารถึงกันอีกด้วย ดังเช่น การประชุมทางไกล (Video Conference) การเรียนการสอนทางไกล (Tele-Education) และการแพทย์ทางไกล (Tele-Medicine) เป็นต้น

ซึ่งในการใช้งานเพื่อติดต่อสื่อสารกันนั้น จำเป็นต้องอาศัยระบบการส่งสัญญาณที่มีอยู่เป็นสื่อในการติดต่อที่สำคัญ ในปัจจุบันได้มีวิธีการเช่าวงจรจากผู้ให้บริการที่มีอยู่ซึ่งมีราคาค่อนข้างสูงมาก และยังไม่มีความคล่องตัวในการติดต่อไปยังที่ต่าง ๆ อีกด้วย ทั้งนี้ก็เนื่องมาจากการที่สัญญาณภาพ Video มีขนาดข้อมูลที่มากทำให้การเก็บข้อมูลนั้นต้องการพื้นที่ในการจัดเก็บมากด้วย จึงทำให้ไม่มีความคล่องตัว ในการที่จะนำข้อมูลนั้นไป Presentation หรือส่ง E-mail และโดยเฉพาะอย่างยิ่งในกรณีที่เป็นการติดต่อสื่อสารด้วยสัญญาณภาพ Video ด้วยแล้วจำเป็นต้องใช้ช่องสัญญาณที่มี Bandwidth ที่กว้างมาก ๆ ดังเช่น ในการประชุมทางไกลจะต้องใช้ขนาดของช่องสัญญาณอย่างน้อย 384 kbps เป็นต้น

การลดขนาดของสัญญาณภาพจะสามารถนำมาช่วยให้การติดต่อสื่อสารด้วยสัญญาณภาพมีความต้องการใช้ช่องสัญญาณที่ลดลง ซึ่งอาจนำไปสู่การติดต่อสื่อสารด้วยสัญญาณภาพผ่านระบบโทรศัพท์พื้นฐาน (Video Telephony) ได้

สำหรับการออกแบบและสร้างการบีบอัดสัญญาณภาพอุปกรณ์ FPGA นี้ FPGA เป็นชิพในลักษณะของ Programmable Device ที่สามารถโปรแกรมตัวมันให้สามารถเป็นวงจรดิจิตอลใด ๆ ก็

ได้ โครงสร้างข้างในประกอบด้วยอาเรย์ของลอจิกเกตต่าง ๆ มากมาย ซึ่งในปัจจุบันความจุเกตภายในตัว FPGA Chip นั้นได้เพิ่มจากระดับไม่กี่พันตัว เป็นถึงล้านตัว ดังนั้นจึงรองรับวงจรทางดิจิตอลที่สลับซับซ้อนได้ รวมถึงความเร็วในการทำงานที่ได้เปรียบอย่างมากมายเมื่อเทียบกับไมโครโปรเซสเซอร์ทั่วไป นอกจากนี้ ในด้านการออกแบบ พัฒนา และทดสอบ ก็ทำได้ง่าย เนื่องจากเราสามารถโปรแกรมได้เอง และหลาย ๆ ครั้ง จึงไม่มีความเสี่ยงใด ๆ ทั้งสิ้น และในปัจจุบันการออกแบบที่เป็น FPGA base IC Design นั้นเป็นที่นิยมทำกันและมีแนวโน้มที่จะนำมาใช้งานกันมากขึ้นเรื่อย ๆ

ซึ่งในการออกแบบและสร้างการบีบอัดสัญญาณภาพอุปกรณ์ FPGA นี้ก็จะได้มุ่งเน้นที่ความรวดเร็วของการบีบอัดสัญญาณ และคุณภาพของสัญญาณที่ได้จาการสร้างสัญญาณขึ้นใหม่ที่มีขนาด Bandwidth ที่คาดว่าจะใช้ในการส่งสัญญาณภาพ เป็นสำคัญ

วัตถุประสงค์

1. เพื่อลดขนาดข้อมูลของสัญญาณภาพที่เหมาะสมกับการใช้งานในรูปแบบของ Multimedia และการสื่อสารด้วยภาพผ่านระบบช่องสัญญาณระดับต่ำ
2. เพื่อศึกษา Algorithm ของ Wavelet Transform ในการลดขนาดข้อมูลของสัญญาณภาพ และการสร้างของสัญญาณภาพขึ้นมาใหม่
3. ออกแบบและสร้าง FPGA ในการบีบอัดสัญญาณภาพ

การตรวจเอกสาร

1. สัญญาณภาพ

1.1 สัญญาณภาพ หรือ Video

สัญญาณภาพ หรือ Video เกิดจากการนำเอาภาพที่มีความสัมพันธ์อย่างต่อเนื่องกันมาวางเรียงต่อเนื่องกันไปตามลำดับเวลา ซึ่งโดยทั่วไปแล้วสำหรับสัญญาณภาพที่สมบูรณ์นั้นภาพแต่ละภาพนำมาวางเรียงต่อกันนี้จะต้องเรียบร้อยภายในเวลา 1 ส่วน 30 วินาที หรือ อักนัยหนึ่งก็คือ ภายใน 1 วินาที จะต้องแสดงภาพให้ได้ถึง 30 ภาพ ซึ่งเราจะเรียกว่า 30 Frame per Second (Fps)

สำหรับภาพ (Image) ที่นำมาให้ ก็เป็นได้ทั้งภาพสี (RGB) ที่มีขนาดของข้อมูลเป็น 24 bpp (bit per pixel) หรือ ภาพขาวดำ (Gray) ที่มีขนาดของข้อมูลเป็น 8 bpp นอกจากนั้นก็จะมีภาพสีที่มีขนาดของข้อมูลเป็น 8 bpp และ 16 bpp โดยที่ทั้ง 2 รูปแบบหลังนั้นเป็นการใช้ตารางของแถบสี (Color Index) มาช่วยในการลดขนาดของข้อมูลภาพซึ่งจะมีรูปแบบการเก็บข้อมูลที่ไม่แน่นอน ส่วนในกรณีที่มีสัญญาณภาพสีขนาด 24 bpp นี้ จะมีขนาดของข้อมูลที่ต้องการสำหรับสัญญาณภาพขนาด 320×240 ในเวลา 1 วินาที จะประมาณ 6.5 Mbytes และเมื่อนำมาแปลงให้อยู่ในรูปของภาพสีขาวดำก็จะมีขนาดของข้อมูลประมาณ 2 Mbytes

โดยทั่วไปแล้วสัญญาณภาพที่พบอยู่ จะสามารถแบ่งออกได้เป็น 3 ประเภทใหญ่ ๆ ได้ดังนี้

1. ภาพธรรมชาติ (Spatial Redundancy)
2. ภาพวัตถุเคลื่อนไหว (Spectral Redundancy)
3. ภาพคนสนทนา (Temporal Redundancy)

1.2 การวัดคุณภาพของสัญญาณภาพ

ในการวัดคุณภาพของสัญญาณภาพนั้นไม่สามารถวัดได้โดยตรงจากสัญญาณภาพแต่จะสามารถได้จากค่าเฉลี่ยที่ได้จากการวัดคุณภาพของภาพที่แสดงขึ้นมาทั้งหมด ซึ่งเป็นการวัดค่าของ

สัญญาณที่สูงที่สุดเทียบการสัญญาณรบกวนที่เกิดขึ้น (Peak Signal-to-Noise Ratio) โดยมีหน่วยที่วัดได้เป็น dB ซึ่งสามารถคำนวณได้ตามสมการที่ 1

$$PSNR = 20\log_{10}\frac{2^{bpp}-1}{RMSE} \qquad (1)$$

เมื่อ RMES (Root Mean-Squared Error) ซึ่งหาได้จากสมการที่ 2 ดังนี้

$$RMSE = \sqrt{\frac{1}{NM}\sum_{i=1}^{N}\sum_{J=1}^{M}\left[f(i,j) - \hat{f}(i,j)\right]^2} \qquad (2)$$

โดยที่ N และ M เป็นขนาด pixel ของภาพ ส่วน f เป็นข้อมูลของภาพต้นแบบและ $\hat{f}$ เป็นข้อมูลของภาพได้จากการสร้างขึ้นใหม่ (Hilton และคณะ, 1994)

1.3 อัตราการบีบอัดข้อมูล (Compression Ratio)

เป็นค่าที่วัดจากอัตราส่วนระหว่างข้อมูลของภาพก่อนการบีบอัด กับขนาดข้อมูลที่ทำการบีบอัดแล้ว โดยทั่วไปขนาดข้อมูลก่อนการบีบอัด สามารถพิจารณาได้จากจำนวนของจุดสีภายในภาพ (ผลคูณของความกว้างของภาพกับความสูงของภาพ) กับขนาดข้อมูลที่ใช้แทนในแต่ละจุดสี (Bit Color) ภาพที่ใช้ในระบบสีจริง (True Color Image) ทั่วไปมีขนาดเท่ากับ 3 Bytes (24 Bits) (Fernandez, 1997) ซึ่งอัตราการบีบอัดข้อมูลคำนวณได้จากสมการที่ 3 เมื่อ C เป็นจำนวน Bits ที่ใช้แทนค่าสีในแต่ละจุดสี W_{Img} และ H_{Img} เป็นความกว้างและความสูงของภาพตามลำดับและ N_c เป็นขนาดแฟ้มข้อมูลที่ได้หลังจากการบีบอัด

$$CR = \frac{C \cdot Wing \cdot Himg}{Nc} \qquad (3)$$

1.4 RGB Color Model

RGB Color Model สามารถนำไปใช้ระบุสีได้มากมาย ขึ้นอยู่กับความสามารถของระบบที่ใช้อยู่ ส่วนมากในการนำไปใช้ 24 bit แต่ละช่องสีประกอบด้วย 8 bit นั้นหมายความว่าแต่ละช่องสี ทั้ง 3 ช่องสี ของ RGB จะประกอบไปด้วยระดับสี 256 ระดับ คือ 0 – 255 ระดับสี ฉะนั้นพื้นฐานของ RGB Color Model มีทั้งหมด 256×256×256=16.7 ล้านกว่าสี

เมื่อ ค่า RGB ใน 24 bit ต่อ 1 Pixel ถูกระบุโดยตัวเลข 0 – 255 จะได้สีดังตัวตัวอย่างเช่น

(0, 0, 0)	คือ สีดำ
(255, 255, 255)	คือ สีขาว
(255, 0, 0)	คือ สีแดง
(0, 255, 0)	คือ สีเขียว
(0, 0, 255)	คือ สีน้ำเงิน
(255, 255, 0)	คือ สีเหลือง
(0, 255, 255)	คือ สีฟ้า
(255, 0, 255)	คือ สีม่วง

2. การบีบอัดสัญญาณภาพ

2.1 Image Compression Using the Haar Wavelet Transform

วิธีการสร้างการบีบอัดภาพโดย Haar Wavelet transforms (Morton and Peterson, 1997) ในเริ่มต้นจะนำภาพต้นฉบับที่แบ่งเป็นสีแต่ละ Pixel แทนลงใน เมตริกซ์นี้ โดยให้ค่าที่น้อยที่สุดแทนสีดำ และค่าที่มากทีสุดแทนสีขาว ในขั้นตอนนี้จะเรียกว่า averaging และ differencing เพื่อจะทำให้เมตริกซ์ใหม่ เหมือนกับภาพต้นฉบับอย่างถูกต้อง ผลใกล้เคียงเหมือนส่งภาพต้นฉบับ

Averaging และ Differencing

ขั้นตอนนี้เราจะนำแถวแรกของเมตริกซ์ 8×8 แทนที่แสดงข้างล่าง โดยเมตริกซ์ 8×8 นี้จะมีขั้นตอนอยู่ 3 ขั้นตอน (2^3=8)

[3 5 4 8 13 7 5 3]

ขั้นตอนที่ 1

จะทำการเฉลี่ยค่าแต่ละคู่ในสตริงต้นฉบับแล้วทำผลลัพธ์มาวางใน 4 ตำแหน่งแรกของสตริงใหม่ ส่วน 4 ตำแหน่งที่เหลือจะเป็น Differences โดยจะได้จากค่าแรกของแต่ละคู่มาทำการลบกัน

(ตัวอย่าง 3-4 = -1, 4-6 = -2) ซึ่งจะเรียกว่า Coefficients

ผลลัพธ์ของขั้นตอนแรกที่ได้จะมี 4 ค่า averages และ 4 ค่า Coefficients (ตัวเอียง) ดังนี้

[4 6 10 4 ***-1*** ***-2*** ***3*** ***1***]

ขั้นตอนที่ 2

วิธีการคล้ายกัน จาก 4 ค่าแรก สตริงใหม่จะเหลือ 2 ค่า และค่า Coefficients โดยค่า Coefficients ที่เหลืออีก 4 ค่า ซึ่งจะนำมาจากขั้นตอนที่1 ซึ่งจะได้ผลลัพธ์สำหรับขั้นตอนที่ 2 ดังนี้

[5 7 ***-1*** ***3*** -1 -2 3 1]

ขั้นตอนที่ 3

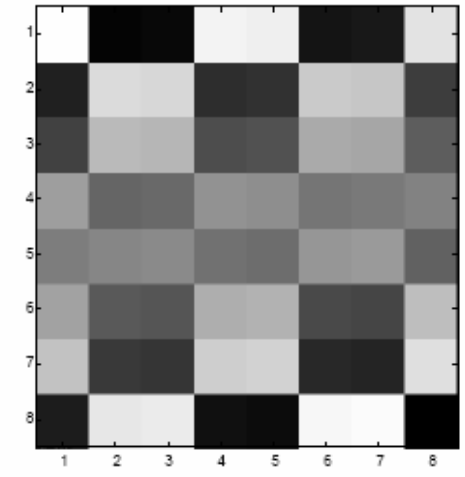
ยังคงกระทำเหมือนเดิม โดยจะได้ค่า Averages ตำแหน่งแรก และตามด้วย 7 Coefficients

[6 ***-1*** -1 3 -1 -2 3 1]

Image Representation

จากการที่ได้ใช้ Averaging และ Differencing ในเมตริกซ์แล้ว ต่อไปจะเป็นการใช้เมตริกซ์ (A) แทนภาพ ดังภาพที่ 1 โดยให้ค่าจำนวนมากแทนลำดับสีสว่าง

$$A = \begin{bmatrix} 64 & 2 & 3 & 61 & 60 & 6 & 7 & 57 \\ 9 & 55 & 54 & 12 & 13 & 51 & 50 & 16 \\ 17 & 47 & 46 & 20 & 21 & 43 & 42 & 24 \\ 40 & 26 & 27 & 37 & 36 & 30 & 31 & 33 \\ 32 & 34 & 35 & 29 & 28 & 38 & 39 & 25 \\ 41 & 23 & 22 & 44 & 45 & 19 & 18 & 48 \\ 49 & 15 & 14 & 52 & 53 & 11 & 10 & 56 \\ 8 & 58 & 59 & 5 & 4 & 62 & 63 & 1 \end{bmatrix}$$



<u>ภาพที่ 1</u> การใช้เมตริกซ์ (A) แทนภาพ โดยให้ค่าจำนวนมากแทนลำดับสีสว่าง

จะได้ค่า Averaging และ Differencing แต่ละแถวโดยผลลัพธ์ แถว Average จะอยู่ในคอลัมน์แรกและส่วนที่เหลือจะเป็น Coefficients ของแถว

$$\begin{bmatrix} 32.5 & 0 & .5 & .5 & 31 & -29 & 27 & -25 \\ 32.5 & 0 & -.5 & -.5 & -23 & 21 & -19 & 17 \\ 32.5 & 0 & -.5 & -,5 & -15 & 13 & -11 & 9 \\ 32.5 & 0 & .5 & .5 & 7 & -5 & 3 & -1 \\ 32.5 & 0 & .5 & ,5 & -1 & 3 & -5 & 7 \\ 32.5 & 0 & -.5 & -.5 & 9 & -11 & 13 & -15 \\ 32.5 & 0 & -.5 & -.5 & 17 & -19 & 21 & -23 \\ 32.5 & 0 & .5 & .5 & -25 & 27 & -29 & 31 \end{bmatrix}$$

ต่อไปจะทำการ Averaging และ Differencing ทางด้านคอลัมน์ โดยจะได้เมตริกซ์ดังนี้

$$\begin{bmatrix} 32.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & -4 & 4 & -4 \\ 0 & 0 & 0 & 0 & 4 & -4 & 4 & -4 \\ 0 & 0 & .5 & .5 & 27 & -25 & 23 & -21 \\ 0 & 0 & -.5 & -.5 & -11 & 9 & -7 & 5 \\ 0 & 0 & .5 & .5 & -5 & 7 & -9 & 11 \\ 0 & 0 & -.5 & -.5 & 21 & -23 & 25 & -27 \end{bmatrix}$$

หลักจากนั้นจะได้เมตริกซ์ที่แทนภาพ โดย Average รวมเพียง 1 ค่า จะอยู่มุมซ้ายบนของเมตริกซ์ ส่วนที่เหลือจะเป็น Coefficients ซึ่งแทนรายละเอียดของภาพและจากการที่ทราบว่าเราสามารถลดข้อมูลที่ไม่จำเป็นออกจากเมตริกซ์ได้ แต่ยังคงสามารถประมาณค่าของเมตริกซ์ต้นฉบับได้ดี

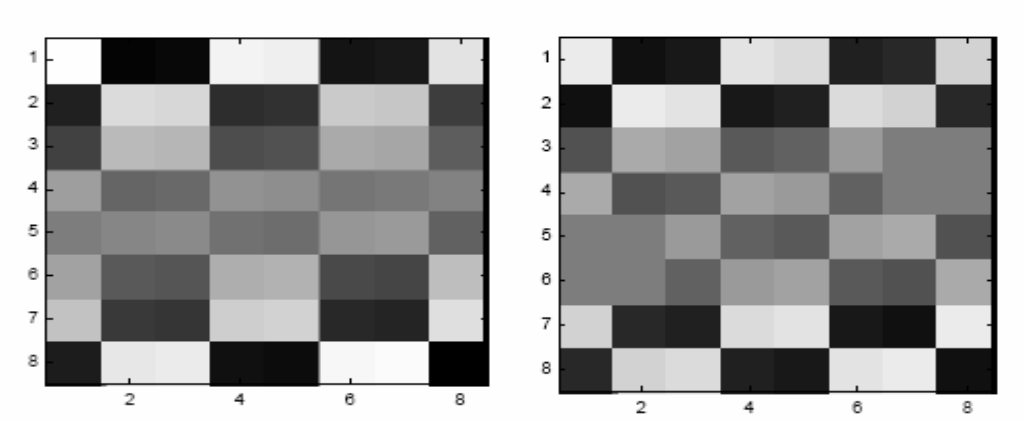
จึงได้ทำการเลือกจำนวน (δ) โดยจะกำหนดค่าให้เป็นศูนย์ ทุกค่าที่มีค่าน้อยกว่า δ โดยเลือก $\delta = 5$ จะได้ 18 รายละเอียด Coefficients (ตัวเอียง)

$$\begin{bmatrix} 32.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ 0 & 0 & 0 & 0 & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ 0 & 0 & \boldsymbol{0} & \boldsymbol{0} & 27 & -25 & 23 & -21 \\ 0 & 0 & \boldsymbol{0} & \boldsymbol{0} & -11 & 9 & -7 & \boldsymbol{0} \\ 0 & 0 & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & 7 & -9 & 11 \\ 0 & 0 & \boldsymbol{0} & \boldsymbol{0} & 21 & -23 & 25 & -27 \end{bmatrix}$$

ต่อไปจะได้ค่าประมาณของเมตริกซ์ต้นฉบับจากการ Inverse ของ Averaging และ Differencing

$$\begin{bmatrix} 59.5 & 5.5 & 7.5 & 57.5 & 55.5 & 9.5 & 11.5 & 53.5 \\ 5.5 & 59.5 & 57.5 & 7.5 & 9.5 & 55.5 & 53.5 & 11.5 \\ 21.5 & 43.5 & 41.5 & 23.5 & 25.5 & 39.5 & 32.5 & 32.5 \\ 43.5 & 21.5 & 23.5 & 41.5 & 39.5 & 25.5 & 32.5 & 32.5 \\ 32.5 & 32.5 & 39.5 & 25.5 & 23.5 & 41.5 & 21.5 & 21.5 \\ 32.5 & 32.5 & 25.5 & 39.5 & 41.5 & 23.5 & 43.5 & 43.5 \\ 53.5 & 11.5 & 9.5 & 55.5 & 57.5 & 7.5 & 5.5 & 59.5 \\ 11.5 & 53.5 & 55.5 & 9.5 & 7.5 & 57.5 & 59.5 & 5.5 \end{bmatrix}$$

จากภาพ 2b จะได้ภาพที่ใกล้เคียงต้นฉบับจากการใช้ขั้นตอนดังกล่าว โดยจะมีบางรายละเอียดที่หายไป แต่จะน้อยมาก และการสูญเสียก็ไม่ได้เด่นชัดมาก



(a) (b)

ภาพที่ 2 (a) Original Image

(b) Decompressed Image

3. การเข้ารหัสข้อมูลแบบ Huffman

(Pitas, 1995) การเข้ารหัสแบบ Huffman นั้นจะแบ่งการเข้ารหัสข้อมูลเป็นสองส่วนคือการเข้ารหัสข้อมูลของค่า DC และการเข้ารหัสข้อมูลของค่า AC

การเข้ารหัสค่า DC ของแถวข้อมูล

ในการเข้ารหัสค่า DC ของแถวข้อมูลนั้น (ข้อมูลตัวแรกในแถวซึ่งคิดค่าสเปกตรัมสัมประสิทธิ์กระแสตรงที่จุด (0,0) ของบล็อกพิกเซลมาก 8×8) จะถูกเข้ารหัสเฉพาะค่าความแตกต่างระหว่างค่า DC ของแถวข้อมูลปัจจุบันกับค่า DC ของแถวข้อมูลซึ่งเป็นสีชนิดเดียวกัน ที่ถูกเข้ารหัสแถวล่าสุดที่ผ่านไป เช่น เมื่อกำลังทำการเข้ารหัส ค่า DC ของแถวข้อมูล C_b อยู่ซึ่งมีค่าเท่ากับ 40 จะทำการหาค่าความแตกต่างกันค่า DC ของแถวข้อมูล C_b ที่ถูกเข้ารหัสผ่านไปแล้วแถวล่าสุด (สมมติว่ามีค่าเท่ากับ 15) ดังนั้นค่าที่จะถูกนำมาเข้ารหัสคิด 25 (40-15) ซึ่งการเข้ารหัสข้อมูลค่า DC จะมีการเข้ารหัสสองส่วนคือ

1. SIZE หมายถึง จำนวนบิตที่จะต้องใช้ในการใส่ค่าข้อมูล (ผลต่าง) ซึ่งจะหาจากตาราง Huffman
2. AMPLITUDE หมายถึง ค่าขนาดของผลต่าง ดังนั้นรหัสที่เข้า คือ

(SIZE) (AMPLITUDE)

ในตารางHuffmanจะเก็บความยาวต่างๆ ที่จะใช้ในการแทนข้อมูลตามค่า Category ของข้อมูลนั้น ซึ่งการหาค่า Category ของข้อมูลสามารถหาได้จากตารางค่า JPEC Coefficient Coding Categories ซึ่งแสดงไว้ในตารางที่ 1 เมื่อได้ค่า Category แล้วก็สามารถหารหัสข้อมูลได้จากตารางค่า DC ของHuffman (ตารางที่ 2) โดยการเทียบหาตามค่า Category ที่ได้จากตารางที่ 1

ตารางที่ 1 แสดงค่า Category ของข้อมูล

Rage	DC Difference Category	AC Category
0	0	N/A
-1.1	1	1
-3,-2,3,3	2	2
-7,…,-4,4,…,7	3	3
-15,…,-8,8,…,15	4	4
-31,…,-16,16,…,31	5	5
-63,…,-32,32,…,63	6	6
-127,…,-64,64,…,127	7	7
-255,…,-128,128,…,255	8	8
-511,…,-256,256,…,511	9	9
-1023,…,-512,512,…,1023	A	A
-2047,…,-1042,1024,…,2047	B	B
-4095,…,-2048,2048,…,4095	C	C
-8191,…,-4096,4096,…,8191	D	D
-16382,…,-8198,8192,…,16383	E	E
-32767,…,-16384,16384,…,32676	F	F

ตารางที่ 2 แสดงค่า DC ของการเข้ารหัสแบบ Huffman

Category	Base Code	Length	Category	Base Code	Length
0	010	3	6	1110	10
1	011	4	7	11110	12
2	100	5	8	111110	14
3	001	5	9	1111110	16
4	101	7	A	11111110	18
5	110	8	B	111111110	20

จะเห็นว่ารหัสที่ได้จากตารางที่ 2 นั้นใช้เป็นตัวบอกช่วยของข้อมูลที่ถูกเข้ารหัสเท่านั้นซึ่งจะได้รหัสตัวที่หนึ่งของข้อมูลออกมา ส่วนการระบุค่าของข้อมูลนั้นจำเป็นจะใช้รหัสตัวที่สองเป็นตัวระบุ โดยรหัสตัวที่สองนี้จะหามาจากค่าของข้อมูลโดยตรงโดยจะมีจำนวนบิตเท่ากับค่า Category ของข้อมูลที่นำมาเข้ารหัสนั่นเอง และจะเอาจากค่า 2's Complement ถ้าข้อมูลมีค่าเป็นลบ

<u>การเข้ารหัสค่า AC ของแถวข้อมูล</u>

เนื่องจากการจัดเรียงข้อมูลแบบซิกแซกนั้น จะทำให้ค่าศูนย์ที่เกิดขึ้นจากการควอนไตซ์อยู่เรียงกันอย่างต่อเนื่องยาว ๆ ทำให้สามารถเข้ารหัสได้ง่าย และยิ่งมีศูนย์ที่ต่อเนื่องกันมากก็จะสามารถเข้ารหัสด้วยจำนวนที่น้อยลงได้มาก และโดยมากค่าศูนย์จะต่อเนื่องกันอยู่ในส่วนท้าย ๆ ของแถวข้อมูล

การเข้ารหัสค่า AC (ข้อมูลตั้งแต่ตัวที่ 2 ถึงตัวที่ 64 ในแถวข้อมูล) จะต่างจากในการเข้ารหัส DC คือจะทำการเข้ารหัสค่าของข้อมูลโดยตรง (ไม่เข้ารหัสเฉพาะค่าความแตกต่างเหมือนใน DC) และการเข้ารหัสค่า AC นั้นจะทำการเข้ารหัสเฉพาะค่า AC ที่มีค่าไม่เป็นศูนย์เท่านั้น ส่วนค่าAC ที่เป็นศูนย์นั้น JPEG จะอาศัยการเข้ารหัสแบบ run-length มาช่วยในการเข้ารหัสดังนี้คือในการเข้ารหัสค่า AC ที่ไม่เป็นศูนย์แต่ละตัวจะเน้นจำนวนของข้อมูล AC ที่เป็นศูนย์ซึ่งอยู่ติดกันด้านหน้าของข้อมูลที่จะเข้ารหัสนั้นโดยถือเป็นค่า run ของข้อมูลในการเข้ารหัส เช่นค่า AC ภายในแถวเป็น 5 8 0 0 9 7 0 0 0 0 4 _ _ _ จะทำการเข้ารหัสเฉพาะข้อมูลที่ไม่เป็นศูนย์และมีค่า run ของข้อมูลแต่ละตัวดังนี้คือ 5(Run-0), 8(Run-0), 7(Run-0), 4(Run-4), …

โดยมีการเข้ารหัสข้อมูลดังกล่าวคล้ายใน DC แต่จะใช้รหัส 3 ตัวแทนข้อมูลแต่ละตัว ดังนี้

1. RUNLENGTH คือ จำนวนค่า 0 ที่ต่อเนื่องกันก่อนหน้าข้อมูลตัวที่จะเข้ารหัส
2. SIZE คือ จำนวนบิตที่จะต้องใช้ในการใส่ค่าข้อมูลที่ไม่เท่ากับ 0 ซึ่งหาจากตารางHuffman
3. AMPLITUDE คือ ค่าแอมพลิจูดหรือค่าของข้อมูลที่ไม่เท่ากับ 0 นั่นเอง

ดังนั้นรหัสคือ (RUNLENGTH, SIZE) (AMPLITUDE)

หมายเหตุ มีเงื่อนไขพิเศษ คือ ถ้ามี 0 ต่อเนื่องกัน มากกว่า 16 ตัว สมมุติว่าเป็น 22 ตัว จะได้รหัสเป็น (15, 0) (6,4) (13)

(15, 0) ความหมายคือ มี 0 ต่อเนื่องกัน 16 ตัว (เป็นรูปแบบที่กำหนดตายตัว)

(6, 4) (13) ความหมายคือ มี 0 อีก 6 ตัว ใช้ 4 บิตในการแทนค่าข้อมูลที่มีค่าเท่ากับ 13

ซึ่งตารางที่ใช้ในการดูค่าจำนวนบิตที่ใช้ต่าง ๆ กันเรียกว่าเป็นแบบความยาวของรหัสไม่คงที่หรือ (Variable Length Code) เนื่องจากข้อมูลที่มาเข้ารหัสจะมีค่า Run เข้ามาเกี่ยวข้องดังนั้นในตารางค่า AC ของHuffman จำต่างจากในตารางค่า DC ของ Huffman คือจะมีการเปลี่ยนแปลงจาก Category ใน DC เป็น Run/Category นั่นคือการหารหัสข้อมูลจากตารางค่า AC ของ Huffman (ในภาคผนวก) จะต้องทราบค่า Run และ Category ของข้อมูลที่นำมาเข้ารหัส ซึ่งการหาค่า Category ของข้อมูล AC จะเทียบหาจากตาราง JPEG Coefficient Coding Categories ตาม ตารางที่ 3(Pitas, 1995) เช่นเดียวกับใน DC เมื่อทราบค่า Run และ Category ของข้อมูลแล้วก็จะสามารถหารหัสข้อมูลจากตารางค่า AC ของHuffmanได้โดยเทียบค่าตามค่า Run/Category

และภายในตารางค่า AC ของHuffmanจะมีรหัสพิเศษ 2 ตัว คือ รหัสเมื่อค่า Run/Category เท่ากับ 0/0 ให้ในกรณีเมื่อทำการเข้ารหัสข้อมูลภายในแถวจนกระทั่งเหลือแต่ข้อมูลที่เป็นศูนย์เพียงอย่างเดียวก็จะใช้รหัสนี้เป็นตัวบอกตัวถอดรหัสว่าข้อมูลที่เหลือทั้งหมดภายในแถวมีค่าเป็นศูนย์ และรหัสอีกตัวหนึ่งเมื่อ Run/Category เท่ากับ F/0 ใช้เมื่อมีข้อมูลที่มีค่าศูนย์อยู่ติดกัน 16 ตัวภายในแถวโดยยังมีข้อมูลที่ไม่เป็นศูนย์ที่ยังไม่ถูกเข้ารหัสเหลืออยู่ในแถวอีก ก็จะใช้รหัสตัวนี้แทนข้อมูลที่มีค่าศูนย์ 16 ตัวดังกล่าว

จะเห็นได้ว่ารหัสข้อมูลที่ได้จากตารางค่า AC ของ Huffman นั้นใช้ในการบอกจำนวนศูนย์ (คือค่า Run) ที่อยู่ด้านหน้าของข้อมูลนั้นและช่วงของค่าข้อมูล (คือค่า Category) ที่นำมาเข้ารหัส ดังนั้นจะต้องมีรหัสตัวที่สองสำหรับใช้ในการระบุค่าของข้อมูล ซึ่งการหารรหัสตัวที่สองนี้ใช้วิธีเดียวกันกับใน DC คือเอามาจาก LSB จำนวน Category บิตของข้อมูลที่นำมาเข้ารหัสและจะเอามาจากค่า 2’s Complement ถ้าข้อมูลมีค่าเป็นลบ

ตารางที่ 3 แสดง JPEG Default AC Code (Luminance)

Run/Category	Base Code	Length	Run/Category	Base Code	Length
0/0	**1010(=EOB)**	**4**			
0/1	00	3	8/1	11111010	9
0/2	01	4	8/2	1111111110000000	17
0/3	100	6	8/3	1111111110110111	19
0/4	1011	8	8/4	1111111110111000	20
0/5	11010	10	8/5	1111111110111001	21
0/6	111000	12	8/6	1111111110111010	22
0/7	1111000	14	8/7	1111111110111011	23
0/8	1111110110	18	8/8	1111111110111100	24
0/9	1111111110000010	25	8/9	1111111110111101	25
0/A	1111111110000011	26	8/A	1111111110001101	26
1/1	1100	5	9/1	111111000	10
1/2	111001	8	9/2	111111110111111	18
1/3	1111001	10	9/3	111111111000000	19
1/4	111110110	13	9/4	111111111000001	20
1/5	11111110110	16	9/5	111111111000010	21
1/6	1111111110000100	22	9/6	111111111000011	22
1/7	1111111110000101	23	9/7	111111111000100	23
1/8	1111111110000110	24	9/8	111111111000101	24
1/9	1111111110000111	25	9/9	111111111000110	25
1/A	1111111110001000	26	9/A	11111111ooo111	26
2/1	11011	6	A/1	111111001	10
2/2	11111000	10	A/2	1111111111001000	18
2/3	1111110111	13	A/3	1111111111001001	19
2/4	1111111110001001	20	A/4	1111111111001010	20
2/5	1111111110001010	21	A/5	1111111111001011	21

ตารางที่3 (ต่อ)

Run/Category	Base Code	Length	Run/Category	Base Code	Length
2/6	1111111110001011	22	A/6	1111111111001100	22
2/7	1111111110001100	23	A/7	1111111111001101	23
2/8	1111111110001101	24	A/8	1111111111001110	24
2/9	1111111110001110	25	A/9	1111111111001111	25
2/A	1111111110001111	26	A/A	1111111111010000	26
3/1	111010	7	B/1	111111010	10
3/2	111110111	11	B/2	1111111111010001	18
3/3	11111110111	14	B/3	1111111111010010	19
3/4	1111111110010000	20	B/4	1111111111010011	20
3/5	1111111110010001	21	B/5	1111111111010100	21
3/6	1111111110011010	22	B/6	1111111111010101	22
3/7	1111111110010011	23	B/7	1111111111010110	23
3/8	1111111110010100	24	B/8	1111111111010111	24
3/9	1111111110010101	25	B/9	1111111111011000	25
3/A	1111111110010110	26	B/A	1111111111011001	26
4/1	111011	7	C/1	1111111010	11
4/2	1111111000	12	C/2	1111111111011010	18
4/3	1111111110010111	19	C/3	1111111111011011	19
4/4	1111111110011000	20	C/4	1111111111011100	20
4/5	1111111110011001	21	C/5	1111111111011101	21
4/6	1111111110011010	22	C/6	1111111111011110	22
4/7	1111111110011011	23	C/7	1111111111011111	23
4/8	1111111110011100	24	C/8	1111111111100000	24
4/9	1111111110011101	25	C/9	1111111111100001	25
4/A	1111111110011110	26	C/A	1111111111100010	26
5/1	1111010	8	D/1	11111111010	12

ตารางที่3 (ต่อ)

Run/Category	Base Code	Length	Run/Category	Base Code	Length
5/2	1111111001	12	D/2	111111111110001	18
5/3	1111111110011111	19	D/3	1111111111100011	19
5/4	1111111110100000	20	D/4	1111111111100100	20
5/5	1111111110100001	21	D/5	1111111111100101	21
5/6	1111111110100010	22	D/6	1111111111100110	22
5/7	1111111110100011	23	D/7	1111111111101000	23
5/8	1111111110100100	24	D/8	1111111111101001	24
5/9	1111111110100101	25	D/9	1111111111101010	25
5/A	1111111110100110	26	D/A	1111111111101011	26
6/1	1111011	8	E/1	111111110110	13
6/2	11111111000	13	E/2	1111111111101100	18
6/3	1111111110100111	19	E/3	1111111111101101	19
6/4	1111111110101000	20	E/4	1111111111101110	20
6/5	1111111110101001	21	E/5	1111111111101111	21
6/6	1111111110101010	22	E/6	1111111111110000	22
6/7	1111111110101011	23	E/7	1111111111110001	23
6/8	1111111110101100	24	E/8	1111111111110010	24
6/9	1111111110101001	25	E/9	1111111111110011	25
6/A	1111111110101110	26	E/A	1111111111110100	26
7/1	11111001	9	F/0	111111110111	12
7/2	11111111001	13	F/1	1111111111110101	17
7/3	1111111110101111	19	F/2	1111111111110110	18
7/4	1111111110110000	20	F/3	1111111111110111	19
7/5	1111111110110001	21	F/4	1111111111111000	20
7/6	1111111110110010	22	F/5	1111111111111001	21
7/7	1111111110110011	23	F/6	1111111111111010	22

ตารางที่3 (ต่อ)

Run/Category	Base Code	Length	Run/Category	Base Code	Length
7/8	1111111110110101	24	F/7	1111111111111011	23
7/9	1111111110110101	25	F/8	1111111111111100	24
7/A	1111111110110110	26	F/9	1111111111111101	25
			F/A	1111111111111110	26

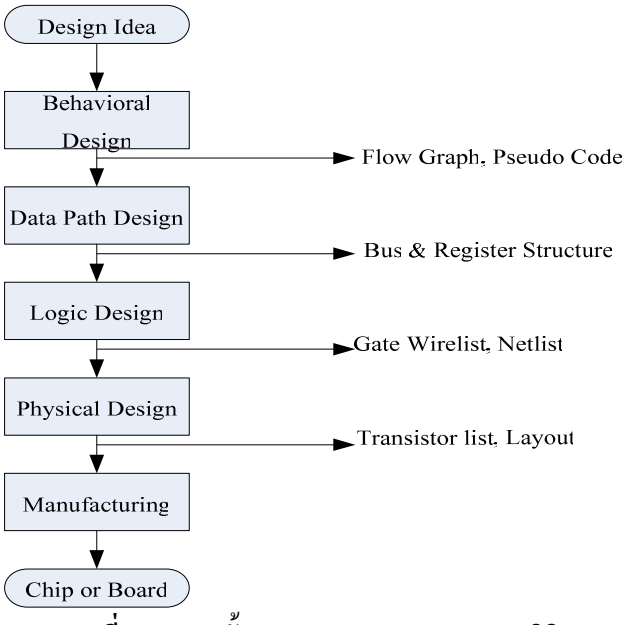
การเข้ารหัสข้อมูลแบบ Huffman เป็นการเข้ารหัสเฉพาะข้อมูลที่มีค่าไม่เป็นศูนย์ ดังนั้นจากการพยายามลดค่าของข้อมูลให้เป็นศูนย์มาก ๆ ในขั้นตอนของการควอนไตซ์เซชั่นและการอ่านข้อมูลแบบซิกแซกที่พยายามทำให้ค่าที่เป็นศูนย์มากอยู่เรียงกันนั้นจึงมีประโยชน์ต่อการเข้ารหัสข้อมูลแบบ Huffman มาก เพราะจะทำให้ข้อมูลที่ต้องทำการเข้ารหัสมีจำนวนน้อยลง

4. ภาษา VHDL

(Yalamanchili, 2001) ความซับซ้อนและขนาดของระบบดิจิตอลในปัจจุบันได้เพิ่มมากขึ้นทุกขณะ ส่งผลให้มีการนำคอมพิวเตอร์เพื่อช่วยใน การออกแบบหรือ CAD มาใช้ในขบวนการออกแบบฮาร์ดแวร์เพิ่มขึ้นเช่นกัน อีกทั้งอุปกรณ์และวิธีการออกแบบใหม่ๆ ก็ถูกพัฒนาขึ้นมาเพื่อช่วยอำนวยความสะดวกให้กับนักออกแบบมากขึ้นด้วย สำหรับภาษาบรรยายอุปกรณ์ฮาร์ดแวร์ (HDL: Hardware Description Language) ก็เป็นเครื่องมืออย่างหนึ่งที่ได้รับการพัฒนามาอย่างต่อเนื่อง เพื่อช่วยให้การปรับปรุงขบวนการออกแบบระบบดิจิตอลเป็นไปอย่างมีประสิทธิภาพ

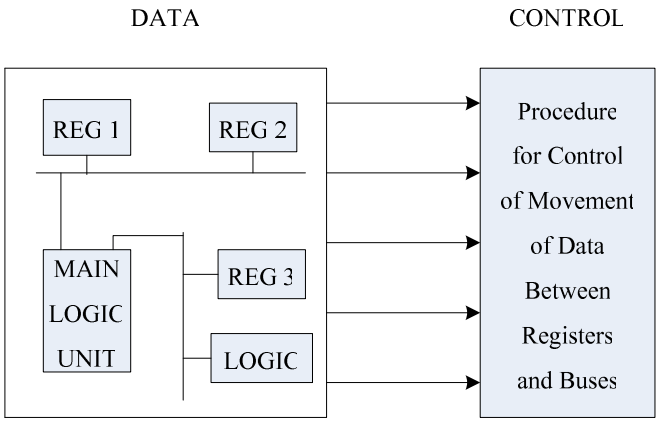
4.1 การออกแบบระบบดิจิตอล

ในการออกแบบระบบดิจิตอล เริ่มตั้งแต่การกำหนดแนวความคิดเบื้องต้นจนกระทั่งได้ออกมาเป็นอุปกรณ์ฮาร์ดแวร์ที่ใช้งานได้จะต้องผ่านขั้นตอนต่างๆ มากมาย และในแต่ละขั้นตอนผู้ออกแบบจะต้องตรวจสอบผลลัพธ์ในแต่ละขั้น ก่อนเข้าสู่กระบวนการออกแบบในขั้นต่อไป ภาพที่ 3 แสดงขั้นตอนปกติที่ใช้ในการออกแบบระบบดิจิตอลทั่วไป ขั้นแรกผู้ออกแบบจะกำหนดแนวความคิดในการออกแบบแล้วทำการพัฒนาให้สามารถนำมาใช้ได้อย่างสมบรูณ์ ซึ่งภายในขั้นตอนนี้ผู้ออกแบบจำเป็นต้องสร้างรูปแบบระบบในเชิงพฤติกรรมขึ้นมาตรวจสอบซึ่งอาจจะเป็นผังงานแสดงแบบหรือ รหัสคำสั่งเทียม (Pseudo code) ก็ได้



ภาพที่ 3 แสดงขั้นตอนการออกแบบระบบดิจิตอล

ขั้นตอนต่อไปเป็นการออกแบบระบบเส้นทางของข้อมูล ผู้ออกแบบจะกำหนดส่วนประกอบของรีจิสเตอร์และวงจรลอจิก ที่จำเป็นทั้งหมดเพื่อนำมาประกอบเป็นระบบที่สมบรูณ์ โดยแต่ละองค์ประกอบสามารถเชื่อมต่อกันด้วยบัสหนึ่งหรือสองทิศทาง (Unidirectional or Bidirectional Bus) ส่วนกระบวนการในการควบคุมการเคลื่อนย้ายข้อมูลระหว่าง รีจิสเตอร์และวงจรลอจิกจะขึ้นอยู่กับพฤติกรรมของระบบที่กำหนดไว้ดังภาพที่ 4



ภาพที่ 4 การออกแบบระบบเส้นทางของข้อมูล

ขั้นตอนถัดมาเป็นการออกแบบวงจรลอจิก ซึ่งจะเกี่ยวข้องกับการนำเกทดิจิตอลพื้นฐานและฟลิปฟลอป (flip-flop) มาประกอบเป็นอุปกรณ์ย่อยต่างๆ เช่น รีจิสเตอร์เก็บข้อมูลบัส วงจรลอจิก และส่วนควบคุมฮาร์ดแวร์ ซึ่งผลลัพธ์ ที่ได้ในขั้นตอนนี้จะเป็นเครือข่ายของการโยงใยระหว่างเกทและ ฟลิปฟลอปนั่นเองการออกแบบในขั้นตอนถัดไป เป็นการเปลี่ยนเครือข่ายการโยงใยที่ได้จากขั้นตอนที่แล้วให้เป็นลำดับของทรานซิสเตอร์ (Transistor List) และ Layout ซึ่งขั้นตอนนี้จะเกี่ยวข้องโดยตรงกับการจัดวางทรานซิสเตอร์หรือไลบรารีเซลล์เพื่อ แทนเกทและฟลิปฟลอปต่างๆและในขั้นตอนสุดท้ายจะเป็นการส่งระบบที่ออกแบบไว้ไปทำการเจือสารที่โรงงานเพื่อผลิตออกมาเป็นวงจรรวมในที่สุด

4.2 ข้อกำหนด

DoD (Department of Defense) ได้ตั้งข้อกำหนดสำหรับภาษา VHDL ในเดือนมกราคม ปี ค.ศ.1983 ไว้ดังนี้

1. ลักษณะทั่วไป

DoD ได้กำหนดให้ VHDL เป็นภาษาสำหรับการออกแบบและบรรยายของฮาร์ดแวร์ ซึ่งหมายถึงความสามารถ ในการอธิบายและออกแบบในระดับสูง การจำลอง (Simulation) การสังเคราะห์ (Synthesis) และการทดสอบ (Testing) นอกจากนั้น VHDL ยังถูกกำหนดไว้สำหรับการบรรยายฮาร์ดแวร์ตั้งแต่ระดับบนซึ่งก็คือระบบจนถึงระดับเกทอีกด้วยเนื่องจากในการทำงานของระบบดิจิตอลนั้น ทุกๆ องค์ประกอบภายในระบบไม่ว่าเล็กหรือใหญ่จะทำงานไปพร้อมๆ กัน ซึ่งในเรื่องของความพร้อมในการทำงานนี้ก็ถือเป็นข้อกำหนดที่สำคัญอย่างหนึ่งของ VHDL ด้วยเช่นกัน (สำหรับในภาษาที่ใช้ในการบรรยายฮาร์ดแวร์นั้นความพร้อมกันจะหมายถึงทุกๆ คำสั่ง องค์ประกอบเกทหรือวงจรลอจิกจะถูกนำมาปฏิบัติทั้งหมด ดังนั้นในที่สุดแล้วก็จะดูเหมือนว่าได้มีการปฏิบัติไป พร้อมๆ กัน)

2. สนับสนุนการออกแบบแบบลำดับขั้น

การออกแบบแบบลำดับขั้นเป็นลักษณะที่สำคัญอย่างหนึ่งสำหรับการออกแบบระบบที่มีหลายๆ ระดับ โดยในการออกแบบจะประกอบด้วยส่วนการบรรยายการเชื่อมต่อ และส่วนการบรรยายหน้าที่การทำงานซึ่งหน้าที่การทำงานของระบบสามารถกำหนดได้ด้วยตัวเองหรืออาจถูกกำหนดโดยโครงสร้างที่ประกอบด้วยองค์ประกอบย่อยๆ ลงไปได้เช่นกัน แต่ที่ระดับล่างสุดองค์ประกอบต้องถูกบรรยายหน้าที่การทำงานด้วยตัวมันเอง และไม่สามารถกำหนดการทำงานโดยลักษณะแบบโครงสร้างได้

3. ไลบรารี

(Sjoholm, 1997) VHDL ได้สนับสนุนการมีไลบรารีเพื่อระบบการจัดการที่ดี ผู้ออกแบบสามารถกำหนดลักษณะและการทำงานของอุปกรณ์พื้นฐานไว้ในระบบไลบรารี หรือจะใช้ไลบรารีที่ระบบได้จัดเตรียมไว้แล้วก็ได้โมเดลและการบรรยายที่ถูกต้องควรจัดเก็บไว้ในไลบรารีหลังจากที่ได้ผ่านการคอมไพล์เรียบร้อยแล้วเพื่อให้ผู้ออกแบบคนอื่นๆ สามารถนำไป ใช้ได้ด้วย

4. ลำดับคำสั่ง

แม้ว่าการปฏิบัติคำสั่งหรือกระบวนการโดยพร้อมกันจะเป็นคุณสมบัติที่สำคัญของ VHDL ก็ตามตัวภาษาเองก็ยังมีการจัดเตรียมลักษณะการควบคุมแบบลำดับคำสั่งไว้ให้ด้วย เมื่อผู้ออกแบบได้กำหนดหน้าที่และองค์ประกอบที่ทำงานพร้อมกันของระบบไว้เรียบร้อยแล้ว ผู้ออกแบบยังสามารถบรรยายหน้าที่การทำงานซึ่งเป็นรายละเอียดภายในของแต่ละองค์ประกอบได้ในลักษณะเดียวกับการเขียนโปรแกรมที่ประกอบด้วยโครงสร้างแบบ case, if - then - else และ loop ทั่วๆ ไปได้ การบรรยายแบบลำดับคำสั่งทำให้การออกแบบหน้าที่การทำงานของอุปกรณ์กระทำได้ สะดวกและง่ายขึ้น อย่างไรก็ตามโครงสร้างทั้งหมดของ VHDL ก็ยังคงเป็นการทำงานแบบพร้อมกันเช่นเดิม

5. การกำหนดคุณสมบัติ

นอกจากการกำหนดอินพุทและเอาท์พุทแล้ว เงื่อนไขอื่นๆ ก็มีผลต่อการปฏิบัติหน้าที่ของอุปกรณ์ฮาร์ดแวร์ด้วยเช่นกัน โดยสิ่งนี้รวมถึงสภาพแวดล้อมและลักษณะทางกายภาพของอุปกรณ์นั้นๆ ด้วย ซึ่งภาษาสำหรับการออกแบบที่ดีควรให้ผู้ออกแบบกำหนดคุณสมบัติของอุปกรณ์ที่ใช้ได้ด้วย เช่น สามารถกำหนดขนาด ลักษณะทางกายภาพเวลา โหลด และเงื่อนไขทางสภาพแวดล้อมอื่นๆ ซึ่งความสามารถในการกำหนดคุณสมบัตินี้ก็เป็นส่วนหนึ่งที่มีอยู่ในภาษา VHDL ด้วยเช่นกัน

6. ชนิดของข้อมูล

VHDL สามารถกำหนดชนิดของข้อมูลไม่เพียงแต่ชนิด BIT และ BOOLEAN เท่านั้น แต่ยังสามารถกำหนดชนิดของข้อมูลเป็นจำนวนเต็ม จำนวนจริง จุดทศนิยม และชนิดลำดับการนับ (Enumerate Type) หรือแม้แต่ชนิดของ ข้อมูลที่ผู้ออกแบบกำหนดขึ้นมาเองก็ได้

7. โปรแกรมย่อย

ความสามารถในการใช้ฟังก์ชันและโพรซีเจอร์ (Procedure) ก็เป็นข้อกำหนดอีกอย่างหนึ่งใน VHDL ซึ่งผู้ออกแบบสามารถนำโปรแกรมย่อยมาใช้ในการเปลี่ยนแปลงชนิดของข้อมูล การกำหนดหน่วยของลอจิก การกำหนดตัวกระทำต่างๆ หรือหน้าที่อื่นๆ ตามที่ต้องการได้ เช่นเดียวกับการเขียนโปรแกรมทั่วไป

8. การควบคุมเวลา

VHDL อนุญาตให้ผู้ออกแบบสามารถกำหนดเวลาในการส่งผ่านข้อมูลหรือสัญญาณได้ตามต้องการ การตรวจสอบ การออกแบบเกทหรือการหน่วงเวลาก็สามารถกระทำได้โดยการกำหนดช่วงเวลาที่แน่นอนหรือกำหนดให้มีการรอคอยเหตุการณ์ (Event) นอกจากนี้ก็ยังสามารถกำหนดรูปแบบของสัญญาณนาฬิกาได้อีกด้วย

9. การกำหนดแบบโครงสร้าง

การกำหนดโครงสร้างขององค์ประกอบต่างๆ สามารถกระทำได้ในทุกระดับของการออกแบบ โดยการกำหนดโครง สร้างขององค์ประกอบร่วมที่เกิดจากองค์ประกอบย่อยซึ่งแตกต่างกันหรือเหมือนกันก็เป็นข้อกำหนดอย่างหนึ่งของ VHDL เช่นกัน

4.3 องค์ประกอบพื้นฐานของ VHDL

(Sjoholm, 1997) รูปแบบพื้นฐานที่ใช้ในการบรรยายถึงองค์ประกอบของ VHDL จะประกอบไปด้วยส่วนกำหนดการเชื่อมต่อ (Interface) และส่วนกำหนดลักษณะเชิงสถาปัตยกรรม (Architecture) ดังแสดงในภาพที่ 5 โดยในการบรรยายการเชื่อมต่อจะขึ้น ต้นด้วยคำว่า ENTITY แล้วตามด้วยชื่อขององค์ประกอบจากนั้นตามด้วยคำว่า IS และถัดมาจะเป็นการบรรยายถึงพอร์ตการติดต่อ อินพุท - เอาท์พุท ขององค์ประกอบ ส่วนลักษณะภายนอกอื่น ๆ เช่น เวลาอุณหภูมิก็สามารถรวมเข้าไปในส่วนนี้ ได้เช่นกัน ในส่วนของการกำหนดลักษณะเชิงสถาปัตยกรรมจะขึ้นต้นด้วยคำว่า ARCHITECTURE ซึ่งเป็นส่วนที่ใช้ บรรยายหน้าที่การทำงานขององค์ประกอบ โดยหน้าที่การทำงานนี้จะขึ้นอยู่กับสัญญาณอินพุท เอาท์พุทและพารามิเตอร์ อื่นๆ ที่ได้กำหนดไว้ในส่วนของการเชื่อมต่อดังภาพที่ 5 และสำหรับการบรรยายหน้าที่ขององค์ประกอบจะเริ่มต้นหลังจากคำว่า BEGIN เป็นต้นไป

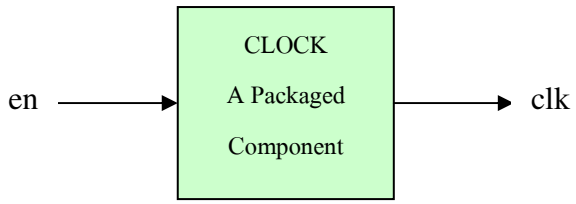
```
Entity component_name IS
    Input and output ports
    Physical and other parameters
END [Component_name] ;

ARCHITECTURE identifier OF component_name IS
    [declaration]
BEGIN
    Specification of the functionality of the component in
terms of its input lines as influenced by physical and
other parameters
END [identifier]
```

ภาพที่ 5 การกำหนดการเชื่อมต่อและสถาปัตยกรรม

4.3.1 การกำหนดการเชื่อมต่อ

การกำหนดการเชื่อมต่อเป็นระดับบนสุดของการออกแบบ โดยในระดับนี้ต้องกำหนดพอร์ตสำหรับการติดต่อกับองค์ประกอบ ภายนอกอื่นๆ ดังตัวอย่างในภาพที่ 6 ซึ่งเป็นบล็อกไดอะแกรม และการบรรยายการเชื่อมต่อขององค์ประกอบสำหรับตัวจ่าย สัญญาณนาฬิกา ในบรรทัดแรกของการบรรยายการเชื่อมต่อเป็นการกำหนดชื่อขององค์ประกอบซึ่งกำหนดเป็น clock_component ตามด้วยคำว่า PORT และชื่อของพอร์ตอยู่ภายในวงเล็บ ส่วน IN และ OUT เป็นการกำหนดโหนดของสัญญาณให้เป็นอินพุทหรือเอาท์พุท และ BIT เป็นการแสดงชนิดของข้อมูล




```
Entity clock_component IS
    PORT (en :IN BIT;ck : OUT BIT)
END clock_name ;
```

ภาพที่ 6 บล็อกไดอะแกรมและการบรรยายการเชื่อมต่อของ clock_ component

4.3.2 การกำหนดรูปแบบการบรรยาย

หน้าที่การทำงานขององค์ประกอบจะถูกบรรยายภายในส่วนนี้ ซึ่งในการบรรยายสามารถกำหนดค่าของสัญญาณ เอาท์พุทในเทอมของอินพุทหรือในรูปขององค์ประกอบอื่นๆ หรือทั้งสองอย่างรวมกันก็ได้ ดังตัวอย่างการบรรยายของ clock component ในภาพที่ 7 ซึ่งเป็นการบรรยายในเชิงพฤติกรรมโดยมี en เป็นอินพุทและ ck เป็นเอาท์พุท PROCESS เป็นคำที่ใช้ในการเริ่มต้นสำหรับการบรรยายในเชิงพฤติกรรม และภายในโปรเซสกำหนดให้ periodic เป็นตัวแปรที่มีคำเริ่มต้นเป็น "0" ถ้าสัญญาณ en มีค่าเป็น "1"' จะทำให้ตัวแปร periodic ถูกคอมพลีเมนท์ (complement) และส่งค่าให้กับ ck ซึ่งเป็นสัญญาณเอาท์พุท และสำหรับคำสั่ง WAIT จะเป็นการกำหนดให้สัญญาณมีคาบเวลาเท่ากับ 1 ไมโครวินาที

```
ARCHITECTURE behavioral OF clock_component IS
BEGIN
   PROCESS
      VARIABLE periodic : BIT := '0';
   BEGIN
      IF en='1' THEN
        Periodic := Not periodic;
      END IF;
      Ck <= periodic;
      WAIT FOR 1 US;
   END PROCESS
END behavioral ;
```

ภาพที่ 7 การบรรยายเชิงพฤติกรรมของ clock _ component

4.3.3 หน่วยการออกแบบแพ็กเก็ต

ข้อมูลต่างๆ ตลอดจนโปรแกรมย่อย ที่เป็นประโยชน์ต่อการเขียนรูปแบบการบรรยายระบบดิจิตอล สามารถเก็บไว้ในส่วนของแพ็กเก็ต ซึ่งหน่วยการออกแบบต่างๆ เช่น หน่วยการออกแบบ Entity หน่วยการออกแบบสถาปัตยกรรมหรือ หน่วยการออกแบบแพ็กเกจอื่นๆ สามารถเรียกข้อมูลเหล่านี้ไปใช้ได้ นอกจากนั้นสิ่งที่นิยมทำกันมากคือการนำรูปแบบ มาตรฐานต่างๆ เช่น อุปกรณ์มาตรฐาน (เช่น ไอซีตระกูล 74XX เป็นต้น) มาเก็บไว้ในรูปของแพ็กเกจ ที่ทุกคนสามารถเข้าถึงได้ ตามปกติแล้วแพ็กเกจจะแบ่งออกเป็น 2 ส่วนคือ การประกาศแพ็กเกจ (Package declaration)และ ส่วนของบอดี้แพ็กเกจ (Package body) เนื่องจากแพ็กเกจถูกสร้างขึ้นเป็นส่วนแยกต่างหากออกจากรูปแบบที่กำลังเขียนอยู่ ฉะนั้นการที่นำแพ็กเกจไปใช้นั้นจะต้องมีการเชื่อมโยงหรืออ้างอิงเสียก่อน ซึ่งในภาษา VHDL สามารถกระทำได้ด้วยชุดคำสั่ง USE

4.4.3.1 PACKAGE DECLARATION

ส่วนที่มีความสำคัญที่สุดของแพ็กเกจ (ถ้ามองในแง่ของการนำไปใช้จากภายนอก) ได้แก่ส่วนการประกาศแพ็กเกจ เนื่องจากเป็นส่วนที่ใช้กำหนดชื่อของสิ่งที่ประกาศอยู่ภายในแพ็กเกจ สำหรับนำไปใช้ภายนอกตัวของแพ็กเกจเอง ถ้ามีการประกาศสิ่งใดๆ ในส่วนของส่วนบอดี้แพ็กเกจ แต่ไม่ถูกประกาศในส่วนการประกาศแพ็กเกจจะทำให้ค่าและพฤติกรรมไม่สามารถนำไปใช้งานในส่วนนอกได้ซึ่งเปรียบเทียบได้กับสิ่งที่ประกาศไว้ในส่วนของการประกาศ Entity คือ จุดเชื่อมต่อ หรือ พอร์ต ที่มีหน้าที่ติดต่อกับโลกภายนอก ฉะนั้นโดยทั่วไปแล้วแพ็กเกจสามารถสร้างขึ้นได้โดยไม่จำเป็น ต้องมีส่วนบอดี้ และยังสามารถนำไปใช้งานจากรูปแบบภายนอกได้เช่นใช้สำหรับประกาศ ชนิด (Type) หรือสัญญาณ เช่นเดียวกับ ส่วนบอดี้แพ็กเกจที่ไม่จำเป็นต้องมีส่วนของการประกาศแพ็กเกจ แต่แพ็กเกจนั้นจะไม่สามารถนำไปใช้จากรูปแบบอื่นได้

```
PACKAGE package_name IS
    Package_declarative_part
END package_name ;
```

ภาพที่ 8 โครงสร้างทั่วไปของส่วนการประกาศแพ็กเกจ

4.3.3.2 PACKAGE BODY

โครงสร้างซึ่งประกอบด้วยลำดับคำสั่งที่ใช้บรรยายฟังก์ชันการทำงานของโปรแกรมย่อยทั้งหลาย ซึ่งชื่อของโปรแกรมย่อยนั้นๆ ได้ถูกประกาศไปแล้วในส่วนของการประกาศแพ็กเกจ จะถูกเก็บไว้ในส่วนของบอดี้แพ็กเกจ ทั้งนี้รวมถึง การกำหนดค่าคงที่ต่างๆ อันได้แก่ค่าคงที่ที่ถูกประกาศชื่อไว้ก่อนในส่วนของการประกาศแพ็กเกจ และถูกกำหนดค่าใน ส่วนของบอดี้แพ็กเกจ ฉะนั้นในส่วนของบอดี้แพ็กเกจจึงไม่จำเป็นต้องมี ถ้าในส่วนของการประกาศแพ็กเกจไม่มีการ ประกาศชื่อที่เป็นโปรแกรมย่อย หรือค่าคงที่การเขียนบอดี้แพ็กเกจนั้นจะเป็นไปตามกฎเกณฑ์ดังแสดงในภาพที่ 9

```
PACKAGE BODY package_name IS
    Declarative part
END package_name ;
```

ภาพที่ 9 โครงสร้างของบอดี้แพ็กเกจ

4.3.4 หน่วยการออกแบบ Configuration

ดังที่ทราบกันแล้วว่าระบบดิจิตอลรูปแบบหนึ่งไม่ว่าจะเป็นอะไรก็ตาม จะสามารถมีหน่วยการออกแบบ Entity ได้ เพียงหนึ่งเดียวเท่านั้น ซึ่งในหน่วยการออกแบบ Entity หนึ่งหน่วยนี้อาจจะมีสถาปัตยกรรมที่เป็นหน่วยรองได้หลาย หน่วย ดังนั้นจะต้องมีหน่วยการออกแบบ Configuration มาเพื่อกำหนดการใช้ Configuration ของการประกอบ Entity กับหน่วยการออกแบบสถาปัตยกรรมหน่วยใดๆ เข้าด้วยกัน

CONFIGARATION identifier **OF** entiry_name **IS**

Configuration_declarative_part

END ;

<u>ภาพที่ 10</u> โครงสร้างโดยทั่วไปของหน่วยการออกแบบโครงแบบ

4.3.5 โปรแกรมย่อย

การใช้ฟังก์ชันและโพรซีเจอร์ใน VHDL เปรียบได้กับการใช้โปรแกรมย่อยในการเขียนโปรแกรมภาษาชั้นสูงทั่วๆ ไปค่าที่ถูกส่งกลับหรือถูกเปลี่ยนแปลงโดยโปรแกรมย่อยอาจจะมีหรือไม่มีผลต่อฮาร์ดแวร์โดยตรงก็ได้ เช่นถ้าใช้ฟังก์ชัน แทนการกระทำในสมการบูลีนก็จะมีผลต่อวงจรลอจิกจริงๆในขณะที่ถ้าใช้โปรแกรมย่อยในการเปลี่ยนชนิดของข้อมูล หรือในการคำนวณค่าการหน่วงเวลาแล้วก็จะไม่มีผลต่อโครงสร้างของฮาร์ดแวร์ ภาพที่ 11 แสดงการใช้โพรซีเจอร์ เพื่อเปลี่ยนข้อมูลชนิด 8 บิตเป็นค่าจำนวนเต็ม และภาพที่ 12 แสดงการใช้ฟังก์ชันโดยกำหนดให้ X เป็นตัวแปรชนิดบิตแทนการกระทำในสมการบูลีน

```
TYPE byte IS ARRAY (7 DOWN TO 0) OF BIT;
   …
PROCEDURE byte_to_integer (ib : IN byte; oi : OUT INTERGER) IS
        VARIABLE result: INTERGER :=0;
BEGIN
        FOR i IN 0 TO 7 LOOP
              IF ib(i) = '1' THEN
              Result := result + 2**i;
               END IF;
        END LOOP;
        oi := result
END byte_to_integer
```

ภาพที่ 11 การใช้โพรซีเจอร์

```
FUNCTION f (a,b,c:BIT) RETURN BIT IS
          VARIABLE x:BIT;
BEGIN
          x := ((NOT a) AND (NOT b) AND c);
          RETURN x;
END f ;
```

ภาพที่ 12 การใช้ฟังก์ชัน

4.3.6 โอเปอร์เรเตอร์

การบรรยายเชิงพฤติกรรมในภาษา VHDL มีตัวดำเนินการหรือโอเปอร์เรเตอร์ทางลอจิกและคณิตศาสตร์เช่นเดียว กับภาษาซอฟต์แวร์ทั่วไปดังภาพที่ 13

PREDEFIND OPERATORS
LOGICAL OPERAORS : NOT AND OR NAND NOR XOR OPERAND TYPE : BIT BOOLEAN RESULT TYPE : BIT BOOLEAN
RELATIONAL OPERATORS : = /= < <= > >= OPERAND TYPE : any type RESULT TYPE : Boolean
ARITHMETIC OPERATORS : + - * / ** MOD REM ABS OPERAND TYPE : INTEGER REAL Physical RESULT TYPE : INTEGER REAL Physical
CONCANTENATION OPERATOR : & OPERAND TYPE : ARRAY of any type RESULT TYPE : array of any type RESULT TYPE : array of any type

ภาพที่ 13 ตัวดำเนินการใน VHDL

4.3.7 เวลา

ในวงจรอิเล็กทรอนิกส์อุปกรณ์ทุกๆ ตัวจะอยู่ในสภาพเตรียมพร้อมเสมอ (Always Active) และจะมีเรื่องของเวลาเข้ามาเกี่ยวข้องในทุกๆเหตุการณ์ที่เกิดขึ้นเสมอ VHDL เป็นภาษาที่ได้รับการออกแบบมาเพื่อให้สามารถบรรยายรูปแบบและการพ้องกันของเวลาสำหรับการทำงานของอุปกรณ์ได้อย่างถูกต้อง การบรรยายการทำงานที่อยู่ภายในส่วน ของการบรรยายสถาปัตยกรรมจะมีการทำงานที่พร้อมกันเสมอ หรือแม้แต่โปรเซสซึ่งมีการทำงานภายในเป็นแบบลำดับคำสั่งก็ตาม ซึ่งหากมีหลายๆ โปรเซสอยู่ภายในโครงสร้างเดียวกัน ทุกๆ โปรเซสก็จะทำงานไปพร้อมๆ กันด้วย

4.3.8 สัญญาณและตัวแปร

สัญญาณมีลักษณะเป็นเสมือนตัวกลางฮาร์ดแวร์ที่ใช้ในการส่งผ่านข้อมูลและเรื่องของเวลาเข้ามาเกี่ยวข้องด้วยการ กำหนดค่าให้กับสัญญาณจะใช้สัญลักษณ์ <= ในการส่งค่าและสามารถใช้คำสั่ง AFTER เพื่อกำหนดช่วงเวลาในการ ส่งผ่านค่าของสัญญาณ เช่น w <= a AFTER 12 NS หมายถึงการกำหนดค่าสัญญาณ a ให้กับ w หลังจากเวลา ผ่านไป 12 นาโนวินาที ในทางตรงข้ามตัวแปรมีลักษณะเป็นเสมือนตัวกลางที่ใช้ในการส่งผ่านข้อมูลและไม่มีเรื่องของ เวลาเข้ามาเกี่ยวข้องด้วย ซึ่งตัวแปรจะถูกใช้ในส่วนที่มีการทำงานเป็นแบบลำดับคำสั่งเช่นใน ฟังก์ชันโพรซีเจอร์ และ โปรเซส สำหรับการกำหนดค่าให้กับตัวแปรจะใช้สัญลักษณ์: =

4.4 การบรรยายเชิงพฤติกรรม

การบรรยายลักษณะการทำงานของอุปกรณ์ฮาร์ดแวร์ในเชิงพฤติกรรม เป็นการบรรยายลักษณะการเปลี่ยนแปลงของข้อมูลในรูปแบบของอัลกอริธึมสำหรับการคำนวณผลลัพธ์ที่เกิดขึ้น ซึ่งสืบเนื่องมาจากการเปลี่ยนแปลงสภาวะของข้อมูล ที่เข้ามาโดยไม่คำนึงถึงลักษณะโครงสร้างหรือความสัมพันธ์ของอุปกรณ์ที่อยู่ภายในว่าจะเป็นอย่างไรในหัวข้อนี้จะแสดงถึงการบรรยายเชิงพฤติกรรม แทนการใช้โมดูลฮาร์ดแวร์รวมถึงข้อกำหนดต่างๆ ที่ควรรู้

4.5 โปรเซส

โปรเซสเป็นรูปแบบพื้นฐานอย่างหนึ่งที่ใช้ในการกำหนดให้กับสัญญาณ โปรเซสจะอยู่ในสถานะที่เตรียมพร้อมอยู่เสมอ และจะปฏิบัติคำสั่งพร้อมๆ กันกับโปรเซสอื่นๆ ที่อยู่ในสถาปัตยกรรมบรรยายเดียวกัน โดยโปรเซสจะปฏิบัติงานตามคำ สั่งทันทีที่มีเหตุการณ์เกิดขึ้นกับสัญญาณที่อยู่ทางด้านขวามือของสัญลักษณ์กำหนดค่าให้กับสัญญาณ (<=) การบรรยาย โปรเซสจะเริ่มต้นด้วยคำสั่ง PROCESS และจบด้วยคำสั่ง END PROCESS ในภาพที่ 14 เป็นการแสดงส่วนประกอบของการบรรยายแบบโปรเซส ซึ่งประกอบด้วยส่วนของการประกาศตัวแปรที่ต้องใช้และส่วนของการปฏิบัติคำสั่งเพื่อให้ได้ผลลัพธ์ที่ต้องการ

```
PROCESS
    Declarative part
    …
BEGIN
    Statement part
    …
END PROCESS ;
```

ภาพที่ 14 รูปแบบของการบรรยายแบบโปรเซส

4.6 การกำหนดตัวดำเนินการภายในโปรเซส

ตัวดำเนินการภายในโปรเซสมี 3 ชนิดคือ ตัวแปร (Variable) ไฟล์ (File) และตัวคงที่ (Constant) ซึ่งตัวดำเนิน การทั้งสามชนิดนี้หากมีการประกาศไว้ในโปรเซสใดก็จะใช้ได้เฉพาะภายในโปรเซสนั้นเท่านั้นสำหรับการติดต่อกับภายนอกหรือระหว่างโปรเซสสามารถทำได้โดยใช้สัญญาณ (Signal) หรือตัวคงที่ที่ได้ประกาศไว้ในส่วนของ ARCHITECTURE ในภาพที่ 15 แสดงตัวอย่างการประกาศตัวกระทำภายในโปรเซส ซึ่งจะอยู่ระหว่างคำสั่ง PROCESS และ BEGIN และค่าเริ่มต้นที่ถูกกำหนดให้กับตัวดำเนินการภายในโปรเซสจะถูกนำมาใช้ในตอนเริ่มต้น ของการปฏิบัติเพียงครั้งเดียวเท่านั้น ต่างกับค่าเริ่มต้นที่อยู่ภายในโปรแกรมย่อยจะถูกนำมาใช้ทุกครั้งที่มีการเรียกใช้ โปรแกรมย่อยนั้น ๆ

```
PROCESS
    FILE flush : TEXT IS IN “filename.dat”;
    VARIABLE var : BIT;
    CONSTANT n : INTEGER :=0;
BEGIN
    …
END PROCESS ;
```

ภาพที่ 15 ตัวอย่างการประกาศตัวดำเนินการภายในโปรเซส

4.7 การกำหนดการกระทำภายในโปรเซส

การกระทำใดๆ ภายในโปรเซสจะเป็นการปฏิบัติแบบลำดับ (Sequential) เสมอ ซึ่งภายในโปรเซสสามารถใช้ประโยค เงื่อนไขหรือการทำซ้ำได้เช่น IF-THEN - ELSE,CASE - WHEN, FOR LOOP และ WHILE LOOP ดังตัวอย่างในภาพที่ 16 และ 17

```
ARCHITECTURE demo OF partial_process IS
…
BEGIN
    PROCESS
    …
    BEGIN
            …
            x <= '1';
            IF x = '1' THEN
                Perform action_1
            ELSE
                Perform action_2
            END IF;
            …
      END PROCESS;
END demo;
```

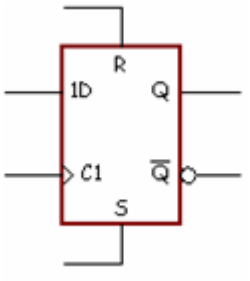
ภาพที่ 16 เงื่อนไขการกระทำในโปรเซส

```
ARCHITECTURE demo OF partial_process IS
   …
BEGIN
   PROCESS
   BEGIN
         …
         x <=  a AFTER 10 NS;
         y <= b AFTER 6 NS;
         …
    END PROCESS;
END demo;
```

ภาพที่ 17 แสดงการกระทำในโปรเซส

4.8 การกระตุ้นและยับยั้งการกระทำของโปรเซส

การกระทำภายในโปรเซสจะอยู่ในสภาพเตรียมพร้อม และมีการปฏิบัติงานอยู่ตลอดเวลาที่มีการเปลี่ยนแปลงของเหตุการณ์ เกิดขึ้น อย่างไรก็ดีเราสามารถกระตุ้นหรือยับยั้งการกระทำภายในโปรเซสได้โดยการกำหนดรายการของสัญญาณที่ต้อง การให้โปรเซสปฏิบัติงานเมื่อมีเหตุการณ์เกิดขึ้นกับสัญญาณที่กำหนดไว้เท่านั้น ส่วนเหตุการณ์ใดๆ ที่เกิดขึ้นกับสัญญาณ ที่ไม่ได้กำหนดไว้ในรายการก็จะไม่ส่งผลให้มีการกระทำภายในโปรเซส ซึ่งรายการของสัญญาณนี้เรียกว่า Sensitivity List และจะกำหนดไว้ภายในวงเล็บหลังคำสั่ง PROCESS ภาพที่ 18 (a) แสดงตัวอย่างโมเดล และภาพที่ 18 (b) เป็นตัวอย่างการบรรยายการเชื่อมต่อของ D-Flip Flop ส่วนภาพที่ 19 แสดงถึงการบรรยายเชิงพฤติกรรมของ D-Flip Flop โดยในภาพที่ 19 (a) เป็นการใช้ตัวกระทำภายนอกโปรเซส และภาพที่ 19 (b) เป็นการใช้ตัวกระทำภายในโปรเซส โดยมีรายการของสัญญาณ (rst, set, clk) เป็นตัวกระตุ้นการปฏิบัติงานภายในโปรเซส



(a)

```
ENTITY component_name IS
    Input and output ports
    Physical and other parameters
END component_name;
```

(b)

ภาพที่ 18 (a) ตัวอย่างโมเดล D-Flip Flop

(b) การบรรยายการเชื่อมต่อของ D-Flip Flop

```
ARCHITECTURE behavioral OF d_sr_flipflop IS
    SIGNAL state : BIT :='0';
BEGIN
    dff : PROCESS (rst, set, clk)
          BEGIN
          IF set = '1' THEN
               state <= '1' AFTER sq_delay
          ELSEIF rst = '1' THEN
               state <= '0' AFTER rq_delay
          ELSEIF clk ='1' AND clk 'EVENT THEN
               state <=  d AFTER cq_delay;
          END IF;
     END PROCESS dff;
END behavioral;
```

(a)

```
ARCHITECTURE average_delay_behavioral OF d_sr_flipflop IS
BEGIN
    dff : PROCESS (rst, set, clk)
          VARIABLE state : BIT :='0';
        BEGIN
       IF set = '1' THEN
            state <= '1';
          ELSEIF rst = '1' THEN
            state <= '0';
          ELSEIF clk = '1' AND clk 'EVENT THEN
            state <= d;
       END IF;
          q <= state AFTER (sq_delay + rq_delay + cq_delay)/3;
          qb <= NOT state AFTER (sq_delay + rq_delay + cq_delay)/3;
    END PROCESS dff;
END behavioral;
```

(b)

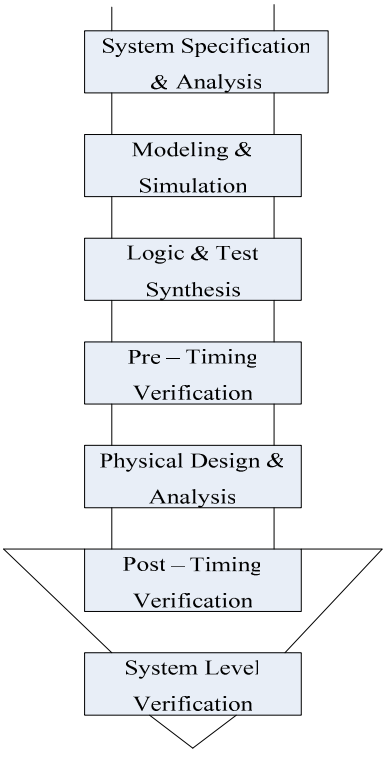
ภาพที่ 19 การบรรยายเชิงพฤติกรรมของ D-Flip Flop

(a) การใช้ตัวกระทำภายนอกโปรเซส

(b) การใช้ตัวกระทำภายในโปรเซส

4.9 การออกแบบจากบนลงล่าง

ในการพัฒนาวงจรรวมดิจิตอลขนาดใหญ่ที่มีความซับซ้อน วิศวกรหรือผู้ออกแบบมักจะมองการออกแบบให้อยู่ในรูปของ บล็อกไดอะแกรมก่อนที่ทำวิเคราะห์ให้ลึกถึงรายละเอียดต่อไป ซึ่งภาษา VHDL นั้นอนุญาตให้อธิบายและวิเคราะห์การ ทำงานของแต่ละบล็อก รวมถึงการปรับปรุงการทำงานจากผลที่วิเคราะห์เพื่อให้ได้การทำงานตามต้องการ นอกจากนี้ยัง สามารถเพิ่มเติมในรายละเอียดในแต่ละขั้นตอนได้ ซึ่งหลักการนี้สอดคล้องกับหลักการออกแบบจากบนลงล่าง (Top - Down Design) นั่นเอง ถ้าทดลองเปรียบเทียบกับการออกแบบจากล่างขึ้นบน (Bottom - Up Design) จะเห็นได้ว่า การออกแบบจากล่างขึ้นบนจะใช้เวลาการออกแบบมากกว่า 90% เนื่องจากเป็นการวาดวงจรด้วยอุปกรณ์ต่างๆ (Schematic capture) ที่ประกอบกันเข้าเป็นวงจรที่ต้องการออกแบบ ก่อนแล้วจึงทำการจำลองการทำงาน และตรวจ สอบความถูกต้อง ดังนั้นการใช้ภาษา VHDL กับหลักการออกแบบจากบนลงล่างจึงเป็นทางออกให้กับวิศวกรให้สามารถออกแบบและพัฒนาวงจรที่มีความซับซ้อนได้มากขึ้น ทั้งยังช่วยลดเวลาและค่าใช้จ่ายในการออกแบบด้วย



ภาพที่ 20 ขั้นตอนการออกแบบจากบนลงล่าง

จากภาพที่ 20 แสดงถึงขั้นตอนของการออกแบบจากบนลงล่าง ทั้งนี้ในทางปฏิบัติอาจมีข้อแตกต่างไปจากนี้บ้างเล็กน้อยเนื่องจากขั้นตอนของการผลิต (Implementation) สามารถกระทำได้หลายเทคโนโลยี สำหรับรายละเอียดของขั้นตอน การออกแบบจากบนลงล่างในแต่ละขั้นตอนมีดังนี้

1. สร้างข้อกำหนดของความต้องการ และวิเคราะห์ระบบเพื่อหาแนวความคิดและหลักการ (Idea and Concept) ในการแก้ปัญหา

2. เขียนรูปแบบของระบบที่ต้องการออกแบบโดยใช้ภาษา VHDL หรือ ภาษา HDL อื่น ๆ สำหรับบรรยายพฤติกรรมการทำงานพร้อมทั้งจำลองการทำงานเพื่อเปรียบเทียบและตรวจสอบความถูกต้องกับข้อกำหนด

3. หลังจากที่ได้หลักการขั้นต้นพร้อมแนวความคิดที่ผ่านการตรวจสอบแล้วหลักการนี้จะถูกเพิ่มเติมในรายละเอียดลงมา เป็นลำดับขั้นที่สองจนกระทั้งอยู่ในระดับที่จะนำไปผลิตวงจรจริง หรือสังเคราะห์ในขั้นตอนนี้เองเทคโนโลยีที่จะมารองรับ วงจรออกแบบจะถูกกำหนดขึ้นและระบบช่วยการออกแบบจะสังเคราะห์วงจรที่ได้จากรูปแบบที่เขียนขึ้นให้อยู่ในรูปของวงจรที่ประกอบด้วยอุปกรณ์อิเล็กทรอนิกส์หรือวงจรในระดับเกท และการเชื่อมต่อระหว่างกันของอุปกรณ์เหล่านั้น หรือไม่ก็อยู่ในรูปของ Net list ที่สามารถนำไปผลิตในอุปกรณ์อื่นได้

4. หลังจากการสังเคราะห์วงจรให้อยู่ในระดับเกทหรือ Net list แล้ว ข้อมูลนี้จะถูกใช้สำหรับจำลองการทำงานในเรื่อง ความถูกต้องของฟังก์ชัน พร้อมกับนำข้อมูลที่เกี่ยวข้องกับเวลาเข้ามาประกอบการพิจารณาด้วย ซึ่งตามปกติแล้วอุปกรณ์ทางอิเล็กทรอนิกส์ทุกชิ้นจะมีเวลาหน่วงของการแพร่กระจาย (Propagation Delay Time) เสมอ ถึงแม้ว่าจะเป็นเวลาที่น้อยมากในระดับนาโนวินาทีก็ตาม แต่ถ้าภายในวงจรหนึ่งประกอบด้วยเกทของฟังก์ชันต่างๆ จำนวน 10,000 เกท ขึ้นไป เวลาดังกล่าวนี้จะสะสมกันมากขึ้น จนอาจทำให้การทำงานของวงจรรวมทั้งหมดผิดพลาดไป หรือไม่สามารถทำงานในย่านความถี่สัญญาณนาฬิกาที่สูงได้

5. ผลิตเป็นวงจรจริง (Technology and device mapping) โดยนำข้อมูลที่ได้จากการสังเคราะห์มาผลิต ซึ่งอาจ จะอยู่ในรูปของแผงวงจรไฟฟ้า ที่ประกอบด้วยอุปกรณ์หลายๆ ชิ้นหรืออยู่ในรูปของวงจรรวม ASIC

6. ทำการตรวจสอบการทำงานและตัวแปรทางด้านเวลาทั้งหมด เพื่อความถูกต้องของวงจรเป็นครั้งสุดท้ายก่อนนำไปรวมเข้ากับอุปกรณ์อื่นๆ ให้เป็นระบบดิจิตอล เนื่องจากในขั้นตอนนี้วงจรที่ออกแบบจะประกอบด้วยจุดต่อทางอินพุทและเอาท์พุท ซึ่งเป็นจุดต่อสำหรับการรับและส่งสัญญาณกับภายนอก

7. นำวงจรที่ออกแบบไว้ประกอบเข้ากับอุปกรณ์อื่นๆ ให้เป็นระบบที่สมบูรณ์ แล้วทำการทดสอบการทำงานทั้งระบบร่วมกับอุปกรณ์อื่นๆ อีกครั้งเพื่อควบคุมคุณภาพของผลิตภัณฑ์

จะเห็นได้ว่าการใช้ VHDL เป็นการเขียนโปรแกรมในภาษาชั้นสูง เพื่อแสดงถึงการทำงานของ Hardware ซึ่งสามารถใช้ VHDL ออกแบบวงจรในทุกระดับตั้งแต่ Gate Level จนถึง System Level ×โดยที่ VHDL จะเป็นภาษามาตรฐานในการออกแบบ เพื่อให้ผู้ออกแบบสามารถแลกเปลี่ยนข้อมูลกันได้โดยไม่มีอุปสรรคในด้านเครื่องมือออกแบบอีกต่อไป

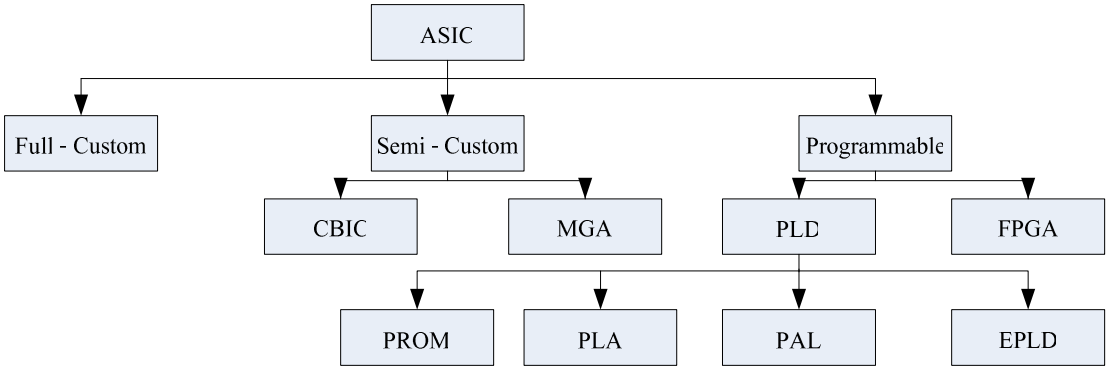
5. FPGA

(Coffman , 2000)ในช่วงก่อนทศวรรษ 1970 อุตสาหกรรมเซมิคอนดัคเตอร์ได้ถูกปลุกให้ตื่นตัวขึ้นหลังจากที่มีการประดิษฐ์วงจรรวมหรือไอซีตัวแรกสำเร็จในยุคแรกนั้นไอซีขนาดเล็กหรือ SSI (Small-Scale Integration) ประกอบไปด้วยเกทดิจิตอลจำนวนไม่มากนัก(ประมาณ 1 ถึง 10 ตัว) ต่อมาได้มีการเพิ่มปริมาณของเกทดิจิตอลและฟังก์ชันทางลอจิกให้มากขึ้นจนเป็น MSI (Medium-Scale Integration) การพัฒนาไอซีเป็นไปอย่างต่อเนื่องจนมาถึงยุคของ LSI (Large-Scale Integration) ซึ่งเป็นยุคที่มีการสร้างไมโครโปรเซสเซอร์ตัวแรกขึ้นและในปัจจุบันเป็นยุคของ VLSI (Very Large-Scale Integration) ซึ่งเทคโนโลยีในการสร้างไอซีรุดหน้าจนสามารถสร้างไมโครโปรเซสเซอร์ขนาด 64 บิต ที่มีหน่วยความจำแคชกับหน่วยคำนวณทางคณิตศาสตร์ของโฟล์ทติงพอยน์ (Floating-Point Arithmetic Units) รวมอยู่ภายในตัวมันและเนื่องจากการปรับปรุงเทคโนโลยีของกระบวนการสร้างซีมอส ที่มีมาอย่างต่อเนื่องทำให้ขนาดของทรานซิสเตอร์ที่บรรจุอยู่ภายในไอซีมีขนาดเล็กลงเรื่อยๆ จนบางคนโดยเฉพาะ ในญี่ปุ่นใช้คำว่า ULSI (Ultra large Scale Integration) เพื่อใช้เรียกระดับของไอซีในปัจจุบันแต่คนส่วนมากยังมักนิยมเรียกเพียงแค่ VLSI

จากการปรากฏตัวของ VLSI ในช่วงทศวรรษ 1980 ทำให้วิศวกรเริ่มมีการออกแบบไอซีตามความต้องการของลูกค้าซึ่งใช้ใน ระบบที่เจาะจงนอกเหนือจากการใช้ไอซีมาตรฐานเพียงอย่างเดียว โดยไอซีเหล่านี้มีชื่อเรียกว่า ASIC: Application-Specific Integrated Circuit (ออกเสียงว่า เอ-ซิก) ซึ่งตัวอย่างของ ASIC ได้แก่ชิพไอซีที่ใช้ สำหรับตุ๊กตาของเล่นพูดได้ ดาวเทียม และ ชิพที่ภายในบรรจุด้วยไมโครโปรเซสเซอร์กับอุปกรณ์ทางลอจิกอื่นๆ

5.1 ประเภทของ ASIC

ASIC แบ่งเป็น 3 ประเภทใหญ่ๆ คือ Full-custom, Semi-custom และ Programmable ดังภาพที่ 21



ภาพที่ 21 ประเภทของ ASIC

1. Full-custom

ASIC ประเภทนี้ลูกค้าจะเป็นผู้ออกแบบเซลล์ลอจิก (เช่น แอนด์เกท ออร์เกท มัลติเพล็กเซอร์ และฟลิบฟล็อบ) และลักษณะการจัดวางอุปกรณ์บนตัวไอซีรวมถึงหน้ากากสำหรับควบคุมการเจือและสร้างชั้นสาร (Mask) ต่างๆ ที่ ใช้ในการทำไอซีเอง ดังนั้นค่าใช้จ่ายในการออกแบบและการผลิตจะสูงมาก

2. Semi-custom

ASIC ประเภทนี้เซลล์ลอจิกจะถูกออกแบบเอาไว้ก่อนแล้วในรูปแบบของไลบรารีและลูกค้าจะเป็นผู้ออกแบบ Mask ต่างๆเอง ตัวอย่างของไอซีประเภทนี้ได้แก่ Standard-Cell-Based ASIC และ Masked Gate-Array-Based ASIC

2.1 Standard-Cell-Based ASIC

ไอซีประเภทนี้จะมีพื้นที่สำหรับจัดวางเซลล์ลอจิกมาตรฐานซึ่งถูกออกแบบเอาไว้แล้ว ในบางครั้งเซลล์มาตรฐานเหล่านี้จะถูกนำมาประกอบกันเป็นเซลที่มีขนาดใหญ่ขึ้นเรียกว่า Mega cell สำหรับการออกแบบนั้นผู้ออกแบบจะทำเพียงแค่กำหนดตำแหน่งของเซลล์มาตรฐานและการเชื่อมต่อภายในของแต่ละเซลล์เท่านั้นแต่อย่างไรก็ดีเซลล์ต่างๆ เหล่านี้สามารถวางที่ตำแหน่งใดๆ ก็ได้บนแผ่นเวเฟอร์ซิลิกอนนั่นก็หมายความว่าชั้น Mask จะถูกจัดวางตามความต้องการของผู้ออกแบบ

2.2 Masked Gate-Array-Based ASIC

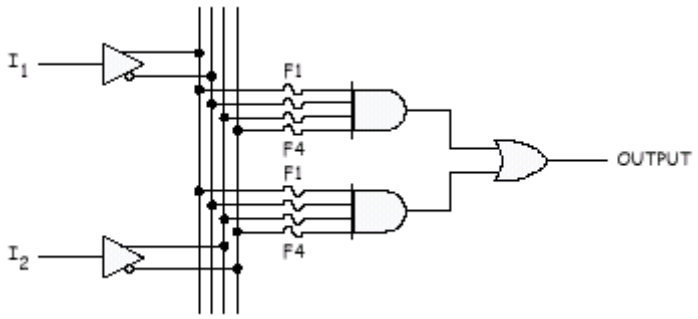
ไอซีชนิดนี้จะมีทรานซิสเตอร์หรือเกทถูกสร้างมาในลักษณะของอะเรย์สองมิติบนแผ่นเวเฟอร์ซิลิกอน และผู้ออกแบบจะทำการออกแบบ Mask เพื่อใช้สำหรับกำหนดการต่อเชื่อมของทรานซิสเตอร์แต่ละตัว

3. Programmable

ASIC ประเภทนี้เซลล์ลอจิกจะถูกออกแบบไว้ก่อนเช่นเดียวกับ Semi-Custom แต่ชั้นของ Mask จะไม่สามารถเปลี่ยนแปลงได้ตามความต้องการของผู้ออกแบบ ไอซีประเภทนี้ยังแบ่งออกเป็น 2 ชนิดคือ Programmable Logic Device (PLD) และ Field Programmable Gate Array (FPGA)

3.1 Programmable Logic Device (PLD)

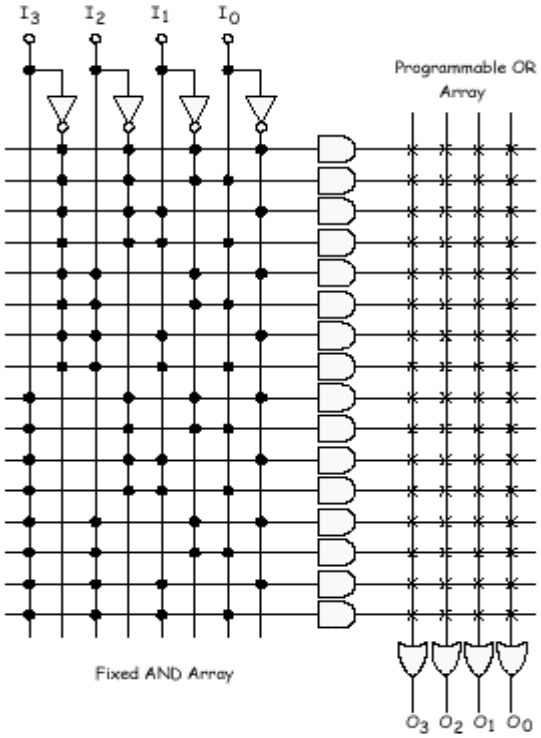
มีโครงสร้างภายในเป็นวงจรพื้นฐานทางด้านลอจิกต่อกันอยู่เป็นกลุ่มซึ่งมีทั้งวงจรคอมบิเนชัน(Combination) และซีเควนเชียล (Sequential) สำหรับเทคโนโลยีของวงจรที่ใช้สร้าง PLD จะมีทั้ง TTL, ECL และ CMOS ตามความเหมาะสมของแต่ละระบบ ไอซี PLD ทุกชนิดมีหลักการพื้นฐานของวงจรภายในที่เหมือนกันโดยมีวงจรคอมบิเนชันที่เป็นผลคูณร่วม บวก (Sum of product)ประกอบไปด้วยชุดของแอนด์เกทต่อร่วมกับออร์เกทและในการโปรแกรมจะเป็นการเลือกว่าอินพุทภายในของแอนด์เกทกับสัญญาณอินพุทใดบ้างที่จะต้องต่อถึงกันซึ่งมีทั้งจากภายนอกและสัญญาณป้อนกลับจากเอาท์พุทภายในเอง เช่น การติดต่ออินพุทของออร์เกทกับเอาท์พุทของแอนด์เกทตัวต่างๆ สำหรับการโปรแกรมทางกายภาพนั้นอินพุทต่างๆ ของอุปกรณ์ทุกตัวจะถูกต่อผ่านฟิวส์เข้ากับแหล่งสัญญาณ ซึ่งถ้าไม่ต้องการใช้สัญญาณใดก็จะตัดฟิวส์ตัวนั้นทิ้งทำให้สามารถโปรแกรมได้เพียงครั้งเดียว ไอซี PLD บางชนิดใช้ มอสทรานซิสเตอร์แทนฟิวส์ทำให้สามารถโปรแกรมโดยใช้กระแสไฟฟ้าและสามารถลบแล้วโปรแกรมเข้าไปใหม่ได้อีก สำหรับไอซีในตระกูล PLD ได้แก่ PROM, PAL, PLA และ EPLD



ภาพที่ 22 วงจรพื้นฐานของ PLD ซึ่งอยู่ในรูปผลคูณร่วมบวก

3.1.1 PROM (Programmable Read Only Memory)

PROM คือหน่วยความจำประเภท ROM ซึ่งนับว่าเป็นไอซี PLD ชนิดหนึ่งซึ่งวงจรภายในของ PROM ประกอบไปด้วยอะเรย์ของแอนด์และออร์เกท (And - Or Array) ผลลัพธ์ที่ขาดาต้าเอาท์พุทสามารถแสดงได้ในสมการของฟังก์ชันผลคูณร่วมบวก (Sum of product) ของสัญญาณอินพุทที่ขาแอดเดรส

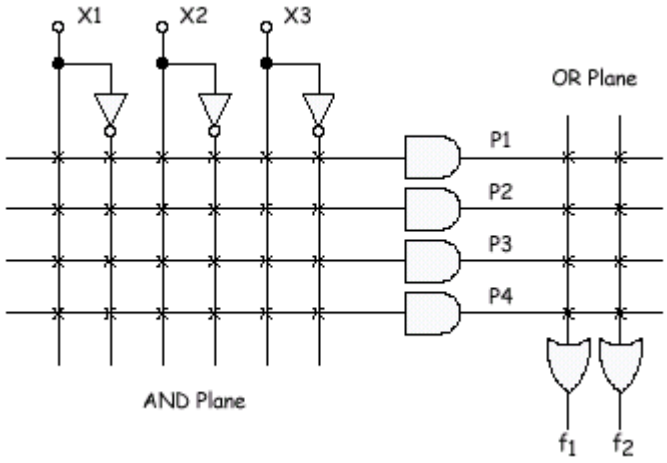


ภาพที่ 23 ลักษณะของ PROM เมื่อเปรียบเทียบเป็นวงจรในรูปผลคูณร่วมบวก

ภาพที่ 23 แสดงถึงลักษณะการเชื่อมต่อแอนด์เกทและออร์เกทของ PROM ขนาด 16×4 บิต วงจรทางด้านซ้ายบนสุดเป็นแอนด์เกทจะให้ผลคูณ (Product) ของกรณีที่อินพุทเป็น 0000 แอนด์เกทที่อยู่ถัดลงมาเป็นผลคูณของกรณีที่อินพุทเป็น 0001, 0010, ...จนถึงตัวล่างสุดคือ ผลคูณในกรณีที่อินพุทเป็น 1111 ซึ่งสำหรับ PROM ที่มีจำนวนอินพุท n ตัวจะมีค่าอินพุทที่ เป็นไปได้ทั้งหมดเท่ากับ 2n และค่าอินพุทเหล่านี้จะถูกจัดวางอยู่ในส่วนอะเรย์ของ AND ซึ่งไม่สามารถแก้ไขได้ แต่ในส่วนของ OR จะเป็นส่วนที่อนุญาตให้ทำการโปรแกรมได้ และเนื่องจากการที่ด้าน AND ของ PROM มีการคอมบิเนชันของอินพุทที่เป็นไปได้ทั้งหมด ดังนั้นผู้ออกแบบจึงไม่จำเป็นต้องทำการลดรูปของฟังก์ชันลอจิกที่ออกแบบไว้เลยแต่อย่าง ไรก็ดีการกระทำเช่นนี้อาจทำให้เกิดจำนวนวงจรที่ไม่มีประสิทธิภาพจำนวนมากบนตัวชิพได้

3.1.2 PLA (Programmable Logic Array)

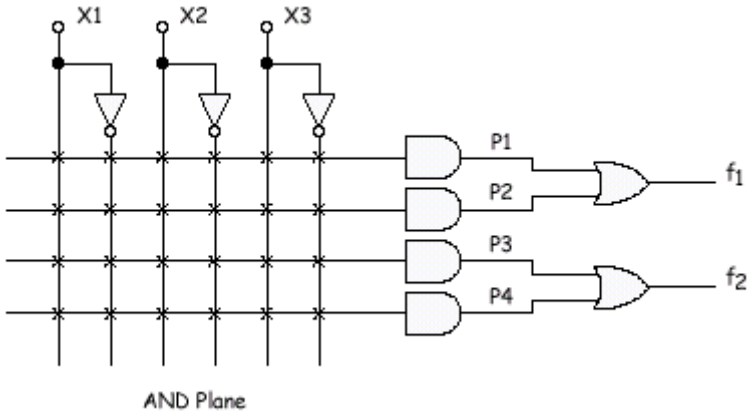
ลักษณะเด่นของ PLA คือสามารถโปรแกรมการเชื่อมต่อได้ทั้งทางด้าน AND และด้าน OR ทำให้มีความยืดหยุ่นในการใช้งานมาก แต่อย่างไรก็ดีข้อเสียที่เห็นได้อย่างชัดเจน ของ PLA คือความยุ่งยากในการสร้างและคุณสมบัติทางด้านความเร็วที่ลดลงเนื่องจากสัญญาณจะต้องวิ่งผ่าน อะเรย์ของ AND และ OR



<u>ภาพที่ 24</u> วงจรพื้นฐานภายในของ PLA

3.1.3 PAL (Programmable Array Logic)

PAL มีลักษณะโครงสร้างที่ใกล้เคียงกับ PROM และ PLA มาก แต่การโปรแกรม PAL จะสามารถทำได้เพียงด้าน AND เท่านั้น



ภาพที่ 25 วงจรพื้นฐานภายในของ PAL

3.1.4 EPLD (Erasable Programmable Logic Device)

EPLD เป็นอุปกรณ์ที่สามารถทำการโปรแกรมได้หลายครั้งซึ่งเหมาะสำหรับการทำวงจรต้นแบบ สำหรับเทคโนโลยีที่ใช้ในการสร้างจะเหมือนกับ CMOS EPROM คือ ใช้มอสทรานซิสเตอร์เชื่อมต่อระหว่างสัญญาณอินพุทกับจุดที่ต้องการแทนการใช้ฟิวส์แบบเดิมทำให้สามารถโปรแกรมการต่อวงจรภายในอุปกรณ์ด้วยการจ่ายไฟฟ้าตามขนาดที่กำหนดไว้และลบได้โดยใช้แสงอัลตร้าไวโอเลทฉายผ่านช่องหน้าต่างกระจกของตัวชิพ

3.2 Field-Programmable Gate Array (FPGA)

เป็นอุปกรณ์ที่มีความซับซ้อนมากกว่า PLD ไปอีกระดับหนึ่ง ซึ่งในความเป็นจริงแล้ว PLD และ FPGA แตกต่างกันน้อยมาก สำหรับ FPGA แล้วนับว่าเป็นอุปกรณ์ตัวใหม่ในตระกูลของ ASIC ซึ่งมีการเจริญเติบโตอย่างรวดเร็วและมีบทบาทที่สำคัญในการเข้ามาแทนที่ระบบอิเล็กทรอนิกส์ที่ใช้ TTL โครงสร้างภายในของ FPGA ประกอบไปด้วยอะเรย์ของลอจิกเกทต่างๆ มากมาย ซึ่งในปัจจุบันความจุเกทภายใน ตัวชิพ FPGA ได้เพิ่มขึ้นจากระดับไม่กี่พันตัวจนถึงระดับ

ล้านตัวซึ่งสามารถรองรับวงจรดิจิตอลที่มีความสลับซับซ้อนได้เป็นอย่างดี นอกจากนี้ในด้านการออกแบบพัฒนาและทดสอบก็ทำได้ง่ายซึ่งในปัจจุบัน การออกแบบวงจรโดยใช้ FPGA กำลังเป็นที่นิยมและมีแนวโน้มที่จะนำมาใช้งานมากขึ้นเรื่อย

5.2 ทำความรู้จักกับ FPGA

ในปัจจุบันมี FPGA อยู่ 4 ชนิดที่วางขายอยู่ในท้องตลาดได้แก่ Symmetrical Array, Row-Based, Hierarchical PLD และ Sea-of-Gates ซึ่งแต่ละชนิดก็มีลักษณะการเชื่อมต่อภายในและการโปรแกรมที่แตกต่างกันไป นอกจากนี้ในการแบ่งประเภทของ FPGA อาจแบ่งได้ตามเทคโนโลยีที่ใช้ในการโปรแกรม ซึ่งมีอยู่ 2 แบบคือ การโปรแกรมโดยการทำให้เกิดการเปลี่ยนแปลงทางกายภาพของตัวชิพ และการโปรแกรมโดยการใช้หน่วยความจำ

1. การโปรแกรมโดยการทำให้เกิดการเปลี่ยนแปลงทางกายภาพ

1.1 Fuse เป็นวิธีการโปรแกรมที่สามารถทำได้เพียงครั้งเดียว ซึ่งหลังจากที่โปรแกรมแล้วจุดเชื่อมต่อจะขาดจากกัน

1.2 Anti Fuse เป็นวิธีการโปรแกรมที่คล้ายกับแบบ Fuse แต่ต่างกันที่หลังจากทำการโปรแกรมแล้วจุดเชื่อมต่อจะเชื่อมถึงกัน

2. การโปรแกรมโดยใช้หน่วยความจำ

2.1 EEPROM Based FPGA

FPGA ที่ใช้การโปรแกรมแบบนี้มักเรียกว่า CPLD ซึ่งเทคโนโลยีที่ใช้จะเหมือนกับ EEPROM ทำให้มีความจุของเกทต่ำ โดยทั่วไปจะน้อยกว่า 20,000 เกท แต่ข้อดีของ EEPROM Based FPGA คือสามารถเก็บข้อมูลที่โปรแกรมลงไปได้โดยม่จำเป็นต้องมีไฟเลี้ยง และในการโปรแกรมจะใช้ทรานซิสเตอร์ 1 ตัวต่อ 1 บิต ซึ่งการโปรแกรมสามารถทำได้ประมาณ 10,000 ครั้ง

2.2 SRAM Based FPGA

FPGA แบบนี้จะใช้เทคโนโลยีในการโปรแกรมเหมือนกับ SRAM (Static RAM) ทำให้สามารถโปรแกรมซ้ำได้โดยไม่จำกัดจำนวนครั้ง นอกจากนี้ยังมีความจุของเกทในระดับปานกลางถึงสูงมาก (ประมาณ 10,000 - 1,000,000 เกท) ซึ่งข้อดีของ SRAM Based FPGA คือใช้เวลาในการโปรแกรมน้อย (ระดับ nsec) การโปรแกรมทำได้ง่ายเทียบได้กับการเขียน SRAM ทั่วไป และ

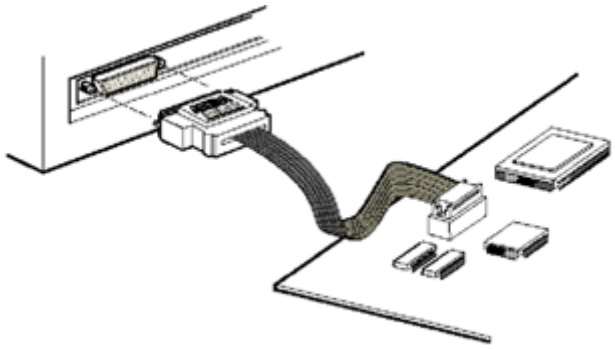
เหมาะสำหรับการออกแบบวงจรที่มีความสลับซับซ้อน ส่วนข้อเสียคือไม่สามารถเก็บโปรแกรมในภาวะที่ไม่มีไฟเลี้ยงได้ ดังนั้น FPGA ชนิดนี้จึงมักใช้ควบคู่กับ ROM เพื่อเก็บโปรแกรมและทำการโหลดโปรแกรมลงในตัวชิพในขณะที่เริ่มต้นใช้งาน

5.3 ปัจจัยที่ทำให้การออกแบบ FPGA ทำได้ง่ายและสะดวกรวดเร็ว

1. ผู้ออกแบบไม่จำเป็นต้องทราบถึงโครงสร้างภายในของตัวชิพ เพียงแต่มีความรู้เกี่ยวกับขั้นตอนการออกแบบลอจิกก็เพียงพอแล้ว ต่างกับการใช้ไมโครโปรเซสเซอร์ซึ่งจำเป็นต้องศึกษาโครงสร้างภายในรวมถึง ภาษา Assembly ของไมโครโปรเซสเซอร์ตัวนั้นด้วย

2. มีการออกแบบโดยใช้ภาษาในการอธิบายการทำงานของวงจร หรือ HDL (Hardware Description Language) เป็นเครื่องมือในการออกแบบ ซึ่งเป็นวิธีการที่มีความยืดหยุ่นสูง ทำได้รวดเร็ว และไม่จำเป็น ต้องทราบถึงลักษณะของวงจรที่ต้องการว่าจะเชื่อมต่อกันอย่างไร เพียงแต่กำหนดลักษณะการทำงานให้มันจากนั้นตัวซอฟต์แวร์จะทำ Synthesis and Optimize ให้ทั้งหมด นอกจากนี้ภาษาที่ใช้ยังเป็นมาตรฐานเดียวกันสามารถใช้ได้กับชิพทุกตัวและทุกบริษัท

3. การโปรแกรมสามารถทำได้เองและใช้เวลาไม่นาน เพียงแค่ส่งข้อมูลผ่านสายดาวน์โหลดทางพอร์ตของคอมพิวเตอร์ก็สามารถโปรแกรมตัวชิพขณะที่อยู่ในระบบได้ โดยไม่จำเป็นต้องถอดมาโปรแกรมข้างนอก ดังภาพที่ 26 และที่สำคัญสามารถโปรแกรมได้หลายครั้ง จึงทำให้ง่ายในการแก้ไขและพัฒนาโดยไม่ต้องเสีย ค่าใช้จ่ายเพิ่มแต่อย่างใด



ภาพที่ 26 การโปรแกรมลงในชิพ

5.4 การออกแบบโดยใช้ภาษาอธิบายพฤติกรรมของฮาร์ดแวร์

ในการออกแบบวงจรดิจิตอลนั้นสามารถทำได้โดยการวาดวงจร (Schematic) หรือใช้ภาษาอธิบายพฤติกรรม (Hardware Description Language) ของฮาร์ดแวร์ ในกรณีของการออก แบบวงจรด้วย ASIC ชนิด Full Custom ผู้ออกแบบจะต้องเขียนวงจรด้วย Schematic จากนั้นจะนำวงจรที่ออกแบบไว้ไปทำการจำลองการทำงาน (Simulate) ซึ่งหากผลออกมาเป็นที่พอใจก็จะต้อง Layout เป็นชั้นสาร และในการออกแบบ ASIC ชนิดนี้ผู้ออกแบบจำเป็นจะต้องทราบถึงเทคโนโลยีที่ใช้ในการสร้างด้วย หลังจากได้ Layout ที่สมบูรณ์แล้วจึงจะส่งไปเข้ากระบวนการสร้างไอซีหรือ Fabrication เพื่อสร้างเป็นชิพไอซีออกมา แต่ในการออกแบบวงจรด้วย FPGA โดยการใช้ Schematic หรือใช้ภาษาอธิบายการทำงานของวงจรจะทำได้สะดวกกว่า เนื่องจากวิธีการนี้ผู้ออกแบบไม่จำเป็นต้องคำนึงถึงเทคโนโลยีที่จะใช้สร้างไอซีและที่สำคัญการออกแบบโดยวิธีนี้สามารถแก้ไขโมเดล (Model) หรือเปลี่ยนแปลงเทคโนโลยีได้สะดวกกว่า เพราะไม่ต้องวาดวงจรใหม่ นั่นคือการออกแบบโดยใช้ภาษาอธิบายฮาร์ดแวร์ จะทำให้โมเดลที่ได้ไม่ขึ้นกับเทคโนโลยี สำหรับภาษาที่ใช้ สำหรับอธิบายพฤติกรรมของฮาร์ดแวร์ที่ใช้กันก็มี VHDL, AHDL และ Verilog เป็นต้น ส่วนรายละเอียด ของขั้นตอนในการออกแบบสามารถอธิบายได้ดังนี้

1.การสังเคราะห์วงจร (Logic Synthesis)

ในขั้นตอนนี้จะใช้ซอฟต์แวร์ในการสังเคราะห์วงจร (Synthesis Tools) ทำการสังเคราะห์พฤติกรรมของวงจรที่ได้จากการออกแบบด้วย Schematic หรือ VHDL ซึ่งต้องทำการตรวจสอบด้วยว่าซอฟต์แวร์ นั้นสนับสนุนเทคโนโลยี FPGA (FPGA Library) ที่ต้องการหรือไม่ ตัวอย่างเช่น FPGA ของบริษัท XILINX และบริษัท ALTERA จะมีซอฟต์แวร์หลายตัวที่สามารถใช้ได้ เช่น Max Plus II ในขั้นตอนนี้ ซอฟต์แวร์สังเคราะห์วงจรจะทำการแปลงโค้ด VHDL และทำการ Optimize เพื่อให้ได้วงจรตามเทคโนโลยีที่เลือกใช้ในการสังเคราะห์วงจรนั้นวงจรระดับเกท (Gate Level) จะไม่เหมาะสมกับโครง สร้างที่มีอยู่ในอุปกรณ์ FPGA ดังนั้นในการ Optimize ซอฟต์แวร์สังเคราะห์วงจร จะต้องทำการ Optimize ให้ได้เป็นวงจรที่ประกอบ ด้วยกลุ่มของลอจิกที่เหมาะสมกับอุปกรณ์ FPGA นั้นๆจึงทำให้ผลที่ได้มีประสิทธิภาพและในขั้นตอนการสังเคราะห์วงจรนี้ผู้ออกแบบสามารถกำหนดข้อบังคับสำหรับโมเดล แต่ละตัวได้ เช่น ข้อบังคับในเรื่องเวลา (Timing Constraints) หรือข้อบังคับในเรื่องของพื้นที่ (Area) หรือกำหนดชนิดและตำแหน่งของ I/O ซึ่งข้อบังคับเหล่านี้จะถูกนำไปใช้ในขั้นตอน Optimize เพื่อให้วงจรที่ได้เป็นไปตามที่กำหนด ส่วนสำคัญในการ Optimize คือการเทียบ (Mapping) โมเดลให้เข้ากับ เทคโนโลยีที่ใช้เพื่อให้ได้วงจรที่

เหมาะสมกับโครงสร้างและสถาปัตยกรรมภายในอุปกรณ์ FPGA เมื่อทำ การสังเคราะห์วงจรเสร็จแล้ว ซอฟแวร์การสังเคราะห์วงจรก็จะมีการรายงานผลว่าโมเดลที่ออกแบบไปนั้น เป็นอย่างไร เช่น มีค่าความหน่วง (Delay) เท่าใด ใช้ทรัพยากรต่างๆใน FPGA อะไรบ้าง เมื่อมาถึงขั้นตอนนี้ ผู้ออกแบบก็จะทราบว่าโมเดลเป็นไปตามข้อบังคับหรือไม่ ถ้าไม่ก็สังเคราะห์ใหม่จนกว่าจะเป็นไปตามที่กำหนด

2.การแบ่งวงจร (Partitioning)

ขั้นตอนนี้เป็นการแบ่งวงจรที่ได้จากการสังเคราะห์ เป็นส่วนย่อยๆ สำหรับลงใน CLBs, IOBs หรือองค์ประกอบอื่นๆ ภายในอุปกรณ์ FPGA สำหรับเกณฑ์ที่ใช้ในการแบ่งคือให้แต่ละส่วนที่จะแยกออกจากกันมีจำนวนสัญญาณที่เชื่อมต่อระหว่างกันน้อยที่สุดเท่าที่จะทำได้ เพื่อลดความหนาแน่นในตอนทำการเชื่อม ต่อสัญญาณ (routing) ในขั้นตอนนี้จะใช้ซอฟต์แวร์ทำโดยซอฟต์แวร์จะเทียบส่วนประกอบของวงจรเช่น เกท (gate), ฟลิป-ฟลอบ (flip-flop) ลงในทรัพยากรต่างๆ ที่มีอยู่ในอุปกรณ์ FPGA หลังจากทำขั้นตอนนี้เสร็จแล้วผู้ออกแบบสามารถที่จะทราบว่าวงจรใช้จำนวนทรัพยากรภายในอุปกรณ์ FPGA ไปเท่าไร ส่วนข้อมูลทางเวลานั้นผู้ออกแบบจะทราบเฉพาะความหน่วงภายในแต่ละส่วนเท่านั้น หรือที่เรียกว่าความหน่วงลอจิก(logic delay) ส่วนซอฟต์แวร์จะรวมเอาซอฟต์แวร์ย่อยอื่นๆ อีก เพื่อให้การทำ PPR (Partitioning Placement & Routing) เป็นไปอย่างต่อเนื่อง

3. การวางอุปกรณ์ (Placement)

ขั้นตอนนี้เป็นการเลือกทำเลที่ตั้งของแต่ละส่วนของวงจรที่ผ่านการแบ่งวงจร (Partitioning) มาแล้วว่าควรจะอยู่ ณ ตำแหน่งไหนในอุปกรณ์ FPGA เพื่อให้ได้ผลลัพธ์ดีที่สุด เช่น วงจรส่วนไหนควรอยู่ใกล้กันเพื่อจะได้ค้นหาเส้นทางได้ (route) ง่ายหรือช่วยลดความหน่วง จะเห็นได้ว่าตำแหน่งภายในอุปกรณ์ FPGA นั้นมีความสำคัญเพราะถ้าจัดวางวงจรลงในตำแหน่งที่ไม่เหมาะสมแล้วจะทำให้ความหน่วงเพิ่มขึ้นหรือ Router ทำการค้นหาเส้นทางสัญญาณได้ไม่หมด การวางอุปกรณ์ที่ดีควรวางส่วนต่างๆให้อยู่ใกล้กันโดยเฉพาะส่วน ที่มีการเชื่อมต่อสัญญาณด้วยกัน นอกจากนั้นการกำหนดตำแหน่งขา I/O (I/O pin) ตามตำแหน่งขา I/O ของ FPGA บนแผ่น PCB ก็จะมีผลโดยตรงเลยคือซอฟต์แวร์จะวาง I/O ลงในตำแหน่งที่ผู้ออกแบบกำหนด ซึ่งบางครั้งตำแหน่งที่กำหนดไปไม่เหมาะสม ดังนั้นการกำหนดขา I/O ควรกำหนดตำแหน่งให้เหมาะสม หรือไม่ก็ให้ซอฟต์แวร์จัดการเอง

4. การเชื่อมต่อสัญญาณ (Routing)

ในขั้นตอนนี้เป็นการเชื่อมต่อสัญญาณระหว่างองค์ประกอบต่างๆ ภายในอุปกรณ์ FPGA ขั้นตอนนี้จะทำต่อเนื่องจากการวางอุปกรณ์ในกรณีที่ทำการวางอุปกรณ์ไว้ไม่ดีซอฟต์แวร์ก็จะทำการเชื่อมต่อสัญญาณ ได้ไม่หมด (เนื่องจากจำนวนทรัพยากรสำหรับเชื่อมต่อสัญญาณนั้นมีอยู่จำกัด) หรือเกิดความหน่วงเกินค่าที่กำหนดในข้อบังคับผู้ออกแบบสามารถทำขั้นตอนนี้ได้โดยใช้ซอฟท์แวร์หรือผู้ออกแบบจะทำการเชื่อมต่อสัญญาณด้วยตนเองก็ได้ แต่ทางที่ดีควรใช้ซอฟต์แวร์ทำดีกว่า นอกจากนั้นการกำหนดข้อบังคับทางเวลาจะช่วยให้ผลที่ได้จากการเชื่อมต่อสัญญาณดีขึ้นได้

5. ความหน่วงด้านเวลา (Delay)

ในการทำ FPGA นั้นความหน่วงที่เกิดขึ้นเป็นความหน่วงที่เกิดจากการวางตำแหน่ง (layout) ของอุปกรณ์ ซึ่งผู้ออกแบบไม่สามารถเข้าไปแก้ไขได้ แต่สามารถทำให้มีความหน่วงน้อยที่สุดได้ สำหรับความหน่วงที่เกิดขึ้นนั้นแยกได้เป็นสองประเภทคือ

- ความหน่วงลอจิก (Logic delay) เป็นความหน่วงภายในองค์ประกอบของอุปกรณ์ FPGA เอง
- ความหน่วงที่เกิดจากการเชื่อมต่อสัญญาณ (Routing delay)

เป็นความหน่วงที่เกิดจากการเชื่อมต่อสัญญาณระหว่างองค์ประกอบภายในอุปกรณ์ FPGA โดยปกติแล้วค่าความหน่วงลอจิกไม่ควรเกิน 50% ของค่าความหน่วงที่ยอมรับได้เพราะความหน่วงที่เกิดจากการเชื่อมต่อสัญญาณมักจะมีค่ามากกว่าค่าความหน่วงลอจิก ดังนั้นในการวางอุปกรณ์ และเชื่อมต่อสัญญาณ ผู้ออกแบบควรกำหนดข้อบังคับทางเวลาเพื่อให้ซอฟต์แวร์ได้ทำงานอย่างมีประสิทธิภาพเพิ่มขึ้น และเพื่อให้ได้ผลลัพธ์ที่ดีขึ้นค่าความหน่วงที่ได้หลังจากการวางอุปกรณ์ และเชื่อมต่อสัญญาณแล้วจะมีค่าความ หน่วงที่ค่อนข้างแน่นอน ซึ่งผู้ออกแบบสามารถทราบได้ว่าโมเดลที่ออกแบบนั้นเป็นไปตามข้อกำหนดหรือไม่

6.การจำลองการทำงานของวงจร (Simulation)

ในขั้นตอนนี้เป็นขั้นตอนที่สำคัญอีกขั้นตอนหนึ่ง เพราะเป็นขั้นตอนที่ผู้ออกแบบตรวจสอบฟังก์ชันการทำงานของโมเดลว่าถูกต้องหรือไม่มีข้อผิดพลาดตรงไหนเพื่อจะได้ทำการแก้ไขให้ถูกต้องในขั้นตอนนี้จะมีซอฟต์แวร์ที่ใช้สำหรับทำการจำลองการทำงานของวงจรที่ใช้อยู่

เช่น Model Sim ของบริษัท Model Technology หรือ Max Plus II ของบริษัท Altera ในการจำลองการทำงานของวงจรควรทำทุกครั้งหลังจากที่มีการทำแต่ละขั้นตอนหลักเสร็จแล้ว เพื่อจะได้ทราบว่าข้อผิดพลาดของโมเดล เกิดขึ้นตอนไหนจะได้แก้ไขข้อผิดพลาดตรงขั้นตอนนั้นๆ ได้เลย ไม่ต้องมาคอยตรวจหาขั้นตอนที่ทำให้เกิดข้อผิดพลาด นั่นคือการทำการจำลองการทำงานของวงจร ต้องทำทั้งหลังการเขียนโค้ด, การสังเคราะห์วงจร และการทำ PPR การจำลองการทำงานของวงจรหลังจากที่เขียนโค้ดเสร็จแล้วนั้น ผู้ออกแบบสามารถทราบได้แค่โมเดลทำงานถูกต้องหรือไม่เท่านั้น (functional test) ยังไม่สามารถตรวจสอบการทำงานในเชิงเวลาได้ถูกต้อง ในการจำลองการทำงานของวงจรหลังจากที่สังเคราะห์เป็นวงจรแล้ว เพื่อตรวจสอบว่าฟังก์ชันการทำงานยังคงถูกต้องหรือไม่ และค่าความหน่วงที่เกิดขึ้นเป็นไปตามข้อบังคับหรือไม่มีข้อผิดพลาดเกิดขึ้นหรือไม่ถ้ามีจะแก้ไขให้ถูกต้อง

ในการจำลองการทำงานของวงจรหลังจากที่ทำการวางอุปกรณ์การเชื่อมต่อสัญญาณ (Post layout simulation) แล้วก็มีความสำคัญเช่นกันเพราะผลที่ได้จากการจำลองการทำงานของวงจรในตอนนี้จะเป็นผลลัพธ์ของโมเดลเลย ซึ่งผู้ออกแบบนอกจากจะตรวจสอบฟังก์ชันการทำงานแล้วยังต้องตรวจสอบคุณสมบัติอื่นๆ เช่น ความหน่วงที่ได้จากการทำ PPR ในรูปแบบค่าความหน่วงมาตรฐาน (Standard Delay Format: SDF) ว่าตรงตามที่กำหนดหรือไม่ หรือตรวจสอบว่าวงจรรวมสามารถใช้งานที่ความถี่สูงสุดเท่าไรนั่นเอง ในการจำลองการทำงานของวงจรควรใช้ซอฟต์แวร์ตัวเดียวกันตลอดเพื่อจะได้เปรียบเทียบผลที่ได้จากขั้นตอนต่างๆ

7. การโปรแกรมอุปกรณ์ FPGA (Configuration)

หลังจากที่โมเดลผ่านขั้นตอนต่างๆ จนกระทั่งผ่านการทำ PPR (Partitioning, Placement & Routing) แล้วนั้นถึงตอนนี้ก็สามารถที่จะดาวน์โหลด (download) ลงในอุปกรณ์ FPGA ได้แล้ว ในการดาวน์โหลดนี้ก่อนอื่นต้องแปลงแบบวงจรรวมที่ได้เป็นข้อมูลวงจร (configuration data) ซึ่งอยู่ในรูปของบิตสตรีม (bit stream) ก่อนแล้วจึงดาวน์โหลดลงไปเพื่อให้อุปกรณ์ FPGA มีฟังก์ชันการทำงานตามโมเดลที่ผู้ออกแบบต้องการ ซึ่งในขั้นตอนนี้จะใช้วิธีที่แตกต่างกันออกไปสำหรับอุปกรณ์ FPGA ของแต่ละบริษัทผู้ผลิตคือ ในกรณีที่เป็นอุปกรณ์ FPGA ชนิดที่ต้องโปรแกรมโดยวิธี SRAM นั้น ในการใช้งานผู้ออกแบบจะต้องเก็บข้อมูลวงจรไว้ในหน่วยความจำประเภท EPROM หรือ serial PROM ด้วยเพื่อจะใช้งานสะดวกขึ้น คือในการใช้งานโมเดลครั้งต่อไปไม่ต้องดาวน์โหลดข้อมูลวงจรจากเครื่องคอมพิวเตอร์อีก เพราะมีข้อมูลวงจรเก็บอยู่ในหน่วยความจำอยู่แล้วแต่กรณีที่อุปกรณ์ FPGA เป็นชนิดที่โปรแกรมโดยใช้วิธี EPROM หรือ Anti fuse ก็ไม่

จำเป็นต้องมีหน่วยความจำสำหรับเก็บข้อมูลวงจร เพราะว่าอุปกรณ์ FPGA ชนิดนี้เมื่อดาวน์โหลดข้อมูลวงจรลงไป ข้อมูลที่ดาวน์โหลดลงไปก็ยังคงอยู่ในอุปกรณ์ FPGA และครั้งต่อไปก็ใช้งานโมเดลที่ออกแบบไว้ได้เลย

5.5 เครื่องมือสำหรับการออกแบบ FPGA

จะเห็นได้ว่าการออกแบบเพื่อทำ FPGA นั้นทำได้สะดวกกว่า ASIC มากเพราะใช้เวลาน้อยกว่ามากด้วย ส่วนสำคัญที่ใช้ในการทำ FPGA คือซอฟต์แวร์ที่ใช้ตั้งแต่เขียนโค้ดอธิบายฮาร์ดแวร์จนกระทั่งดาวน์โหลดลงในอุปกรณ์ FPGA ซึ่งซอฟต์แวร์ที่ใช้ต้องเป็นซอฟต์แวร์ที่ทำงานต่อเนื่องกันได้ สำหรับซอฟต์แวร์ที่ใช้ทำการจำลองการทำงานของวงจรนั้น ต้องสามารถใช้งานต่อเนื่องกับซอฟต์แวร์ที่ใช้ทั้งระบบ เพราะโมเดลที่ได้จากการทำขั้นตอนต่างๆ (ด้วยซอฟต์แวร์ ต่างๆ ต้องเอามาจำลองการทำงานได้ และในการจำลองการทำงานของวงจรควรใช้ซอฟต์แวร์ตัวเดียวกันตลอดทั้งระบบ เพื่อจะได้เปรียบเทียบผลได้ง่าย ในอดีตซอฟต์แวร์ส่วนใหญ่จะใช้งานอยู่บนคอมพิวเตอร์สมรรถนะสูงอย่างเวิร์คสเตชั่น (Workstation) ในปัจจุบันมีการพัฒนาซอฟต์แวร์ที่ใช้บนพีซี (PC) มากขึ้นซึ่งสามารถลดค่าใช้จ่ายในด้านอุปกรณ์คอมพิวเตอร์

FPGA เป็นไอซีหรือชิพอเนกประสงค์ที่สามารถโปรแกรมให้เป็นวงจรดิจิตอลอะไรก็ได้โดยวิธีการโปรแกรมแบบง่ายๆ และสามารถแก้ไขวงจรได้อย่างสะดวกด้วยการโปรแกรมซ้ำ FPGA จะเหมาะสำหรับการออกแบบวงจรดิจิตอลขนาดกลางจนถึงวงจรขนาดใหญ่มากๆ เช่น ไมโครโพรเซสเซอร์ ดิจิตอลฟิลเตอร์ ส่วนประกอบหลักของแผงวงจรดิจิตอลของอุปกรณ์ทางด้านการแพทย์ เครื่องมือวัดต่างๆ อุปกรณ์สื่อสาร หรือ อุปกรณ์เครือข่าย เป็นต้น นักออกแบบดิจิตอลที่เคยออกแบบวงจรขนาดใหญ่และวงจรดิจิตอลที่ซับซ้อนมากๆ ก็จะทราบว่าอาจมีความจำเป็นต้องใช้ FPGA ขนาดใหญ่ที่มีความจุวงจรมากตั้งแต่ขนาดหลายหมื่นเกตจนไปถึงระดับหลักหลายล้านเกต

อุปกรณ์และวิธีการ

อุปกรณ์

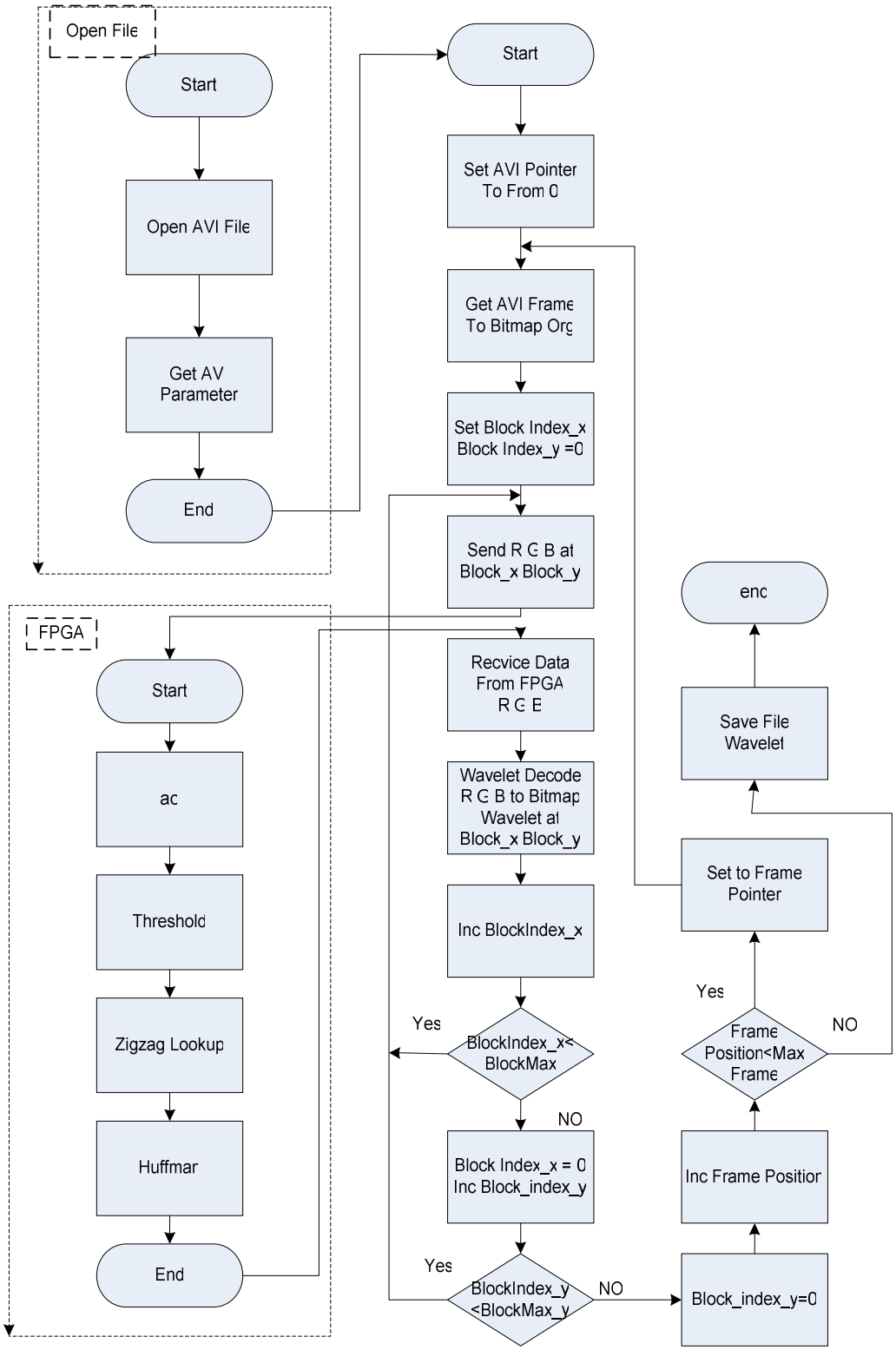
1. เครื่องคอมพิวเตอร์ส่วนบุคคล (Personal Computer)
2. โปรแกรม Xilinx
3. โปรแกรม Delphi
4. FPGA Chip รุ่น Discovery-III XC32S200F

วิธีการ

วิธีการจะมีอยู่ด้วยกัน 2 ส่วนทั้งในส่วนของ Software และ Hardware

1. Software ทดสอบการบีบอัดภาพ

สำหรับการออกแบบและสร้างการบีบอัดสัญญาณภาพวีดีโอบนอุปกรณ์ FPGA นี้ได้ทำการพัฒนาเพื่อนำมาใช้ลดขนาดของสัญญาณภาพ โดยมีขั้นตอนการทำงานของ Software ดังภาพที่ 27



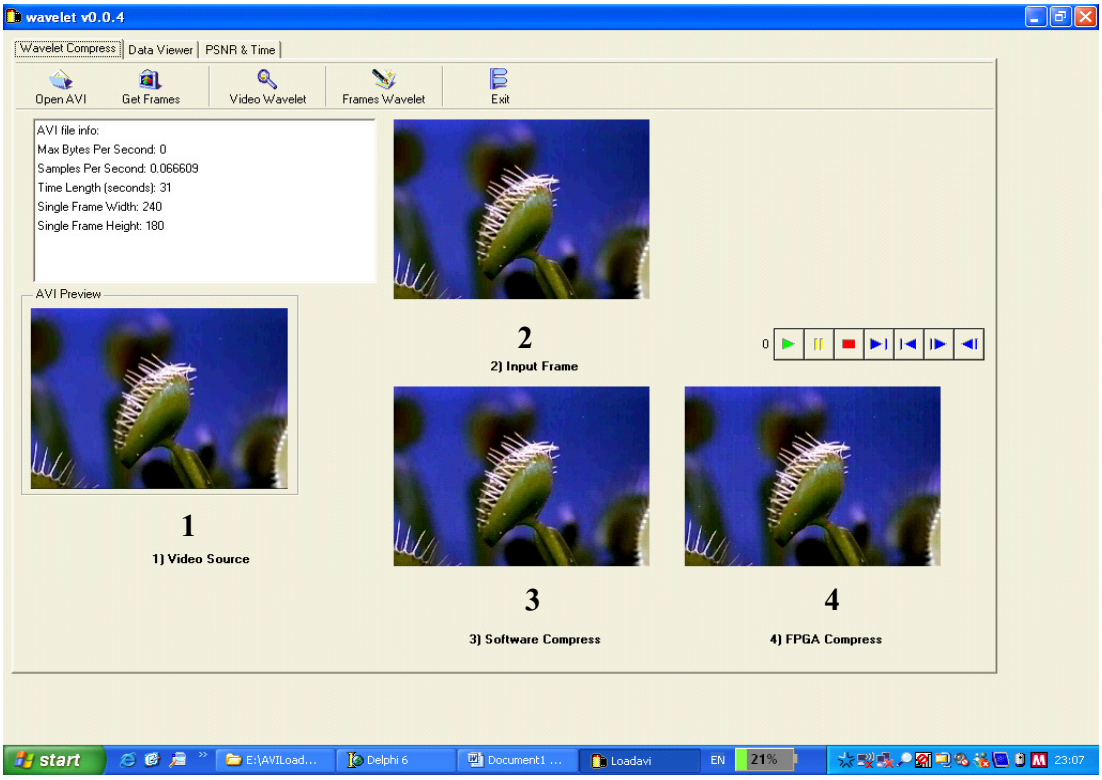
ภาพที่ 27 ขั้นตอนการทำงานของ Software บีบอัดสัญญาณภาพ

ในการเขียนโปรแกรมจะมีหลักๆอยู่ด้วยกัน 2 ส่วน คือในส่วนของ โปรแกรม Delphi และ FPGA โดยมีขั้นตอนดังต่อไปนี้

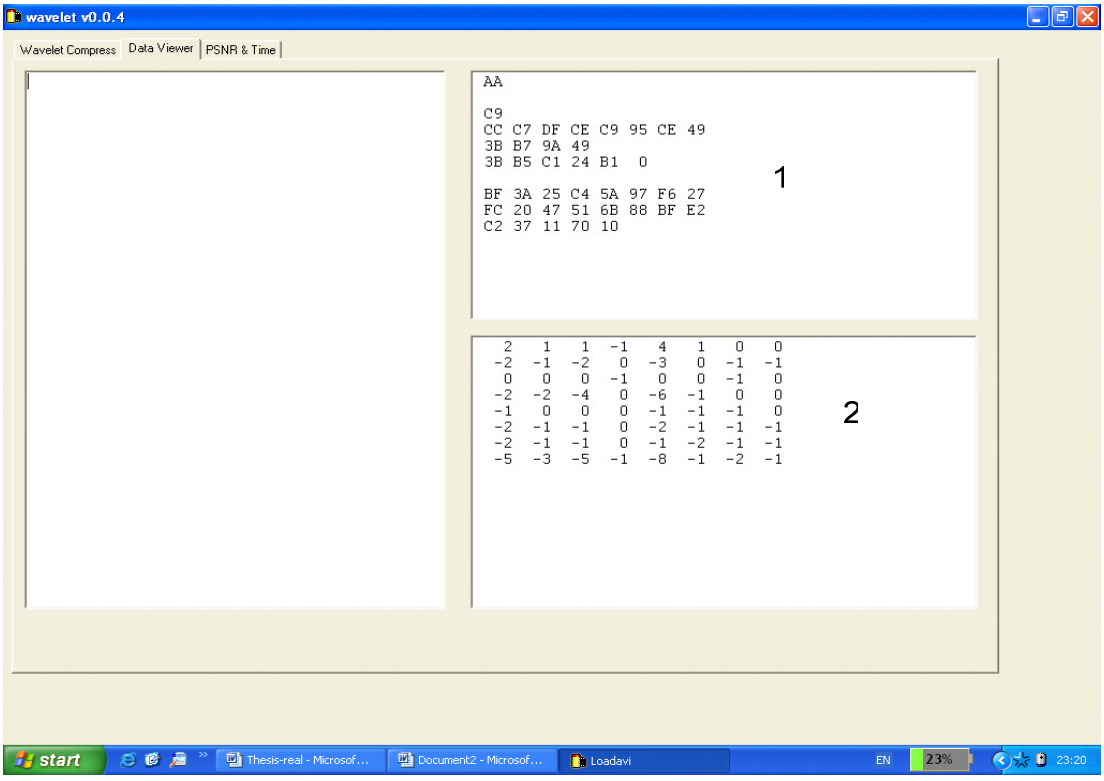
1. ทำการเปิด AVI File ที่ต้องการทำการบีบอัด โดยโปรแกรมจะทำการ Set AVI Pointer ให้มีค่าเริ่มต้นเป็น 0 จากนั้น นำ AVI Frame ที่ 1 มาทำการบีบอัด
2. ทำการ Set ค่า Block x_y ให้มีค่าเป็น 0 เพื่อที่จะทำการบีบอัดที่ Block เริ่มต้น
3. ส่ง R, G, B ของ Block ที่ 1 ที่จะทำการบีบอัด ไปยัง FPGA
4. FPGA จะทำการคำนวณ Haar Wavelet และจัดรูปให้อยู่ในแบบของ Zigzag เพื่อที่จะทำให้ง่ายต่อการเข้ารหัส
5. จากนั้นทำการตั้งค่า Threshold เพื่อที่จะตัดค่าตัวเลขที่ต่ำกว่า Threshold ให้มีค่าเป็นศูนย์
6. นำไปเข้ารหัส Huffman แล้วส่งกลับไป Decode ที่โปรแกรม Delphi
7. ทำการ Decode แล้วตรวจสอบ ว่าถึง Block สุดท้ายที่ทำการบีบอัดแล้วหรือยังถ้ายังให้ทำการบีบอัดต่อไป
8. เมื่อทำการบีบอัดจนเสร็จเรียบร้อยใน Block สุดท้ายแล้ว ก็จะทำการบีบอัดใน Frame ต่อไป
9. หลังจากบีบอัดจนครบ AVI File แล้ว ก็จะทำการ save เพื่อดูขนาดของข้อมูล เป็นอันสิ้นสุดการทำงานของโปรแกรม

การเขียน Form โปรแกรมบีบอัดสัญญาณภาพ

ในการเขียนโปรแกรมนั้นจะต้องมีการออกแบบรูปร่างของโปรแกรมก่อน ซึ่งในการออกแบบเพื่อนำมาใช้ในการทดสอบและบันทึกค่าต่าง ๆ จากการทดลอง ที่มีความสำคัญต่อการทำงานของการลดขนาดของสัญญาณภาพนอกจากนี้ยังมีรายละเอียดอื่น ๆ อีกเช่น ปุ่มกดเริ่มและหยุด เมนู เป็นต้นแสดงดังตัวอย่างของโปรแกรม ในภาพที่ 28

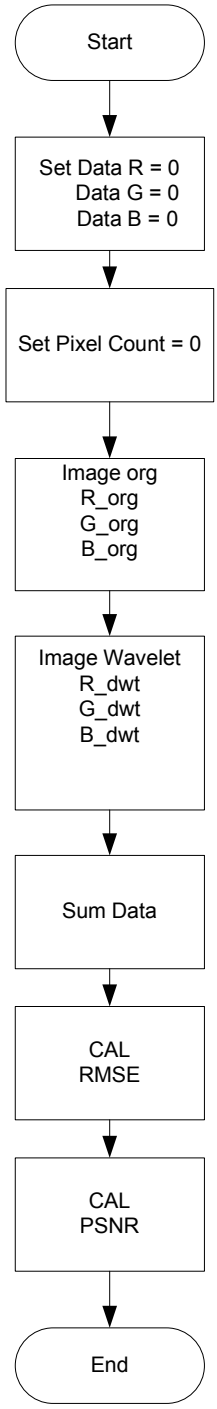


ภาพที่ 28 แสดงตัวอย่าง โปรแกรมทดสอบการบีบอัดภาพที่ได้พัฒนาและใช้ในการทดลอง



ภาพที่ 29 Data Viewer

จากภาพที่ 28 ได้แสดงถึงลักษณะและการใช้งานของโปรแกรมทดสอบการบีบอัดภาพที่พัฒนาขึ้นเพื่อทำการทดสอบในเรื่องการลดขนาดของสัญญาณภาพ โดย 1) Video Source เป็นสัญญาณภาพที่ ที่ต้องการบีบอัด 2) Input Frame แสดง Input Frame ที่ทำการบีบอัด 3) Software Compress ผลของการบีบอัดโดยใช้ Software ที่พัฒนาบน Delphi 4) FPGA Compress เป็นผลการบีบอัดข้อมูลที่ได้จากอุปกรณ์ FPGA ซึ่งจะมี function ที่จำเป็นต้องใช้ในการทดลอง ได้แก่ การกำหนดลักษณะการทดสอบโดยจะเป็นแบบ Frame หรือ แบบ Video ภาพที่ 29 Data viewer 1) เป็นการแสดงผลของข้อมูล Huffman จาก FPGA 2) เป็นการ Inverse ข้อมูลกลับมาเป็นเมตริกซ์ภาพ นอกจากนี้ยังมี function ที่ใช้ในการเก็บข้อมูลเพื่อนำมาวิเคราะห์หาข้อสรุปด้วย ได้แก่ คุณภาพของสัญญาณ (PSNR) ของแต่ละภาพที่แสดงออกมา โดยจากสูตร PSNR สามารถนำมาเขียน Software บน Delphi เพื่อให้คำนวณค่า PSNR ได้ดังมีขั้นตอนการทำงานของ Software แสดงดังภาพ 30 และบันทึกผลคุณภาพของสัญญาณ (PSNR) ที่ได้ แสดงดังภาพที่ 31



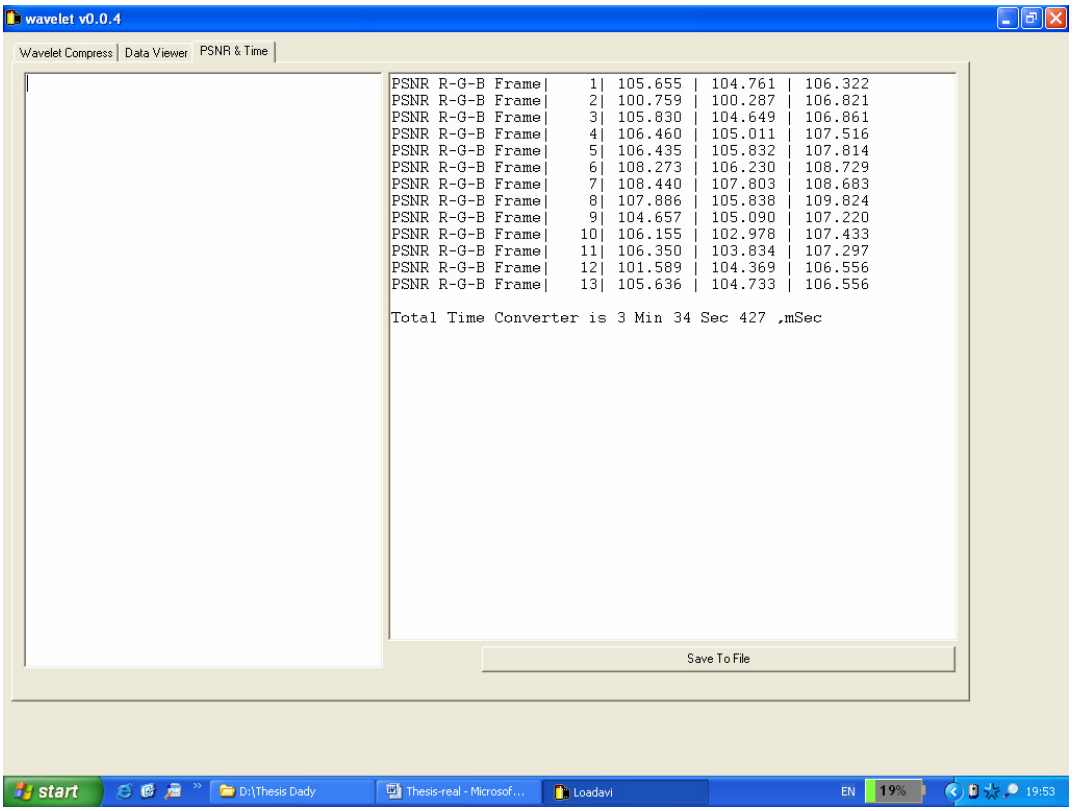
ภาพที่ 30 ขั้นตอนการทำงานของ Software การวัดคุณภาพสัญญาณ (PSNR)

จากภาพที่ 30 มีขั้นตอนการทำงานดังต่อไปนี้

1. ทำการ Set ค่าข้อมูล R, G, B และ Pixel Count ให้มีค่าเป็น 0 เพื่อจะได้ทำการคำนวณค่า PSNR เริ่มต้นที่ Frame แรก
2. หาค่าข้อมูล R, G, B ของภาพต้นฉบับ และ ภาพที่ได้จากการบีบอัด
3. ทำการรวมผลของข้อมูลที่ได้จากภาพต้นฉบับ และภาพที่ได้จากการบีบอัด

$$\sum_{i=1}^{N}\sum_{J=1}^{M}\left[f(i,j) - \hat{f}(i,j) \right]^2$$

4. ทำการคำนวณค่า RMSE $RMSE = \sqrt{\frac{1}{NM}\sum_{i=1}^{N}\sum_{J=1}^{M}\left[f(i,j) - \hat{f}(i,j) \right]^2}$
5. ทำการคำนวณค่า PSNR $PSNR = 20\log_{10}\frac{2^{bpp}-1}{RMSE}$

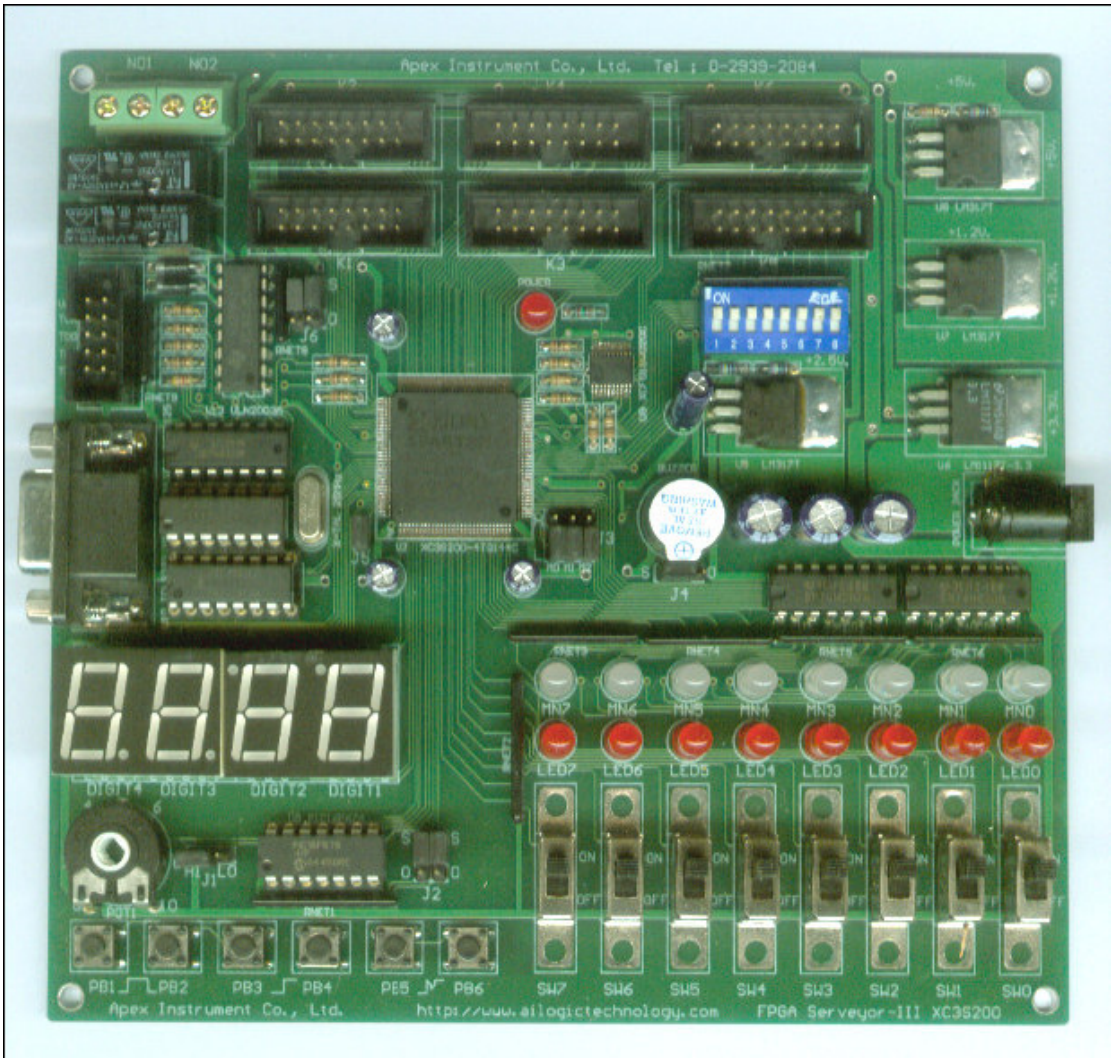


ภาพที่ 31 แสดงการบันทึกผลการทดสอบสัญญาณภาพ (PSNR)

2 Hardware ของ FPGA ที่ใช้ในการบีบอัดสัญญาณภาพ

FPGA เป็นไอซีหรือชิพอเนกประสงค์ที่สามารถโปรแกรมให้เป็นวงจรดิจิตอลอะไรก็ได้ โดยวิธีการโปรแกรมแบบง่ายๆ และสามารถแก้ไขวงจรได้อย่างสะดวกด้วยการโปรแกรมซ้ำ FPGA จะเหมาะสำหรับการออกแบบวงจรดิจิตอลขนาดกลางจนถึงวงจรขนาดใหญ่มากๆ เช่น ไมโครโพรเซสเซอร์ ดิจิตอลฟิลเตอร์ส่วนประกอบหลักของแผงวงจรดิจิตอลของอุปกรณ์ทางด้านการแพทย์ เครื่องมือวัดต่างๆ อุปกรณ์สื่อสารหรืออุปกรณ์เครือข่ายเป็นต้นนักออกแบบดิจิตอลที่เคยออกแบบวงจรขนาดใหญ่และวงจรดิจิตอลที่ซับซ้อนมากๆ ก็จะทราบว่าอาจมีความจำเป็นต้องใช้ FPGA ขนาดใหญ่ที่มีความจุวงจรมากตั้งแต่ขนาดหลายหมื่นเกตจนไปถึงระดับหลักหลายล้านเกต

บอร์ดทดลองอเนกประสงค์รุ่น FPGA Discovery-III XC3S200 มีรายละเอียดแสดงดังรูปที่ 33 โดยที่บอร์ดนี้จะเป็นได้ทั้งบอร์ดทดลองและบอร์ดพัฒนา FPGA ในบอร์ดเดียวกันที่มีความจุวงจรมากถึง 200,000 เกต และใช้ Platform Flash PROM สำหรับเก็บข้อมูลวงจร ซึ่งสามารถโปรแกรมวงจรลง Platform Flash PROM ผ่านทางสายดาวน์โหลดแบบ JTAGได้โดยตรงและสามารถโปรแกรมซ้ำได้ถึง 20,000 ครั้ง บอร์ดอเนกประสงค์นี้มีอุปกรณ์อำนวยความสะดวกที่เพียบพร้อมด้วยอุปกรณ์อินพุตเอาต์พุตอย่างครบครันเพื่อให้ผู้ทดลองได้เรียนรู้การออกแบบวงจรรวมดิจิตอลตั้งแต่วงจรขั้นพื้นฐานจนถึงขั้นนำไปพัฒนาออกแบบสร้างวงจรขนาดใหญ่ได้ด้วยตัวเอง



ภาพที่ 32 แสดงBoard FPGA ที่ใช้ในการบีบอัดสัญญาณภาพ

คุณสมบัติทั่วไปของบอร์ดอเนกประสงค์

• FPGA ตระกูล Spatan-3 ของ Xilinx เบอร์ XC3S200 ขนาด 200,000 เกต Package แบบ TQ144 , Speed Grade:4

• Platform Flash PROM เบอร์ XCF01S ที่โปรแกรมข้อมูลวงจรซ้ำได้ ถึง 20,000 ครั้ง

• 7-Segment จำนวน 4 หลัก (ถอดออกได้)

• LED จำนวน 8 ดวง (สามารถแยกออกจาก I/O ได้โดยการถอดหรือหักเอา RNET3 และ RNET4 ออก)

• Buzzer จำนวน 1 ตัว DIP Switch 8 บิต

• Push Button Switch จำนวน 5 ตัว

• Expansion ports (80 Bits 3.3V. I/O)

• RS-232C Port 1 Port

• I2C Socket สำหรับ EEPROM

• 25 MHz Oscillator (เปลี่ยนเป็นความถี่อื่นๆ ได้โดยใช้ Digital Frequency Synthesizer มีที่อยู่ใน FPGA)

<u>คุณสมบัติที่สำคัญของชิพ FPGA ตระกูล Spartan-3 เบอร์ XC3S200 มีดังนี้</u>

• ความจุวงจร 200,000 เกต

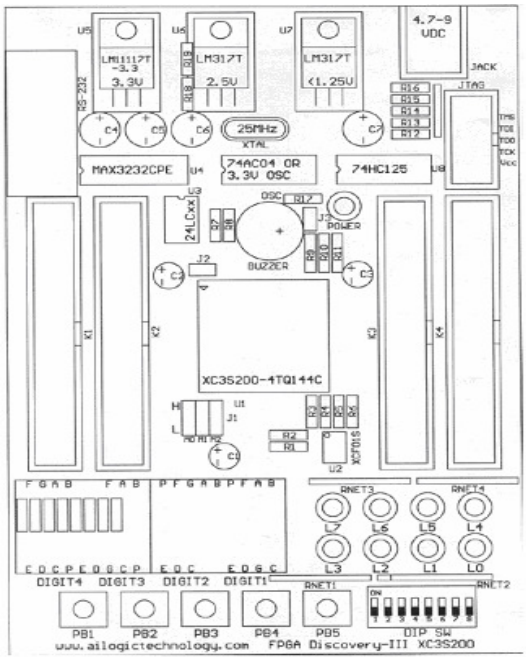
• 18Kb block Rams จำนวน 12 ชุด (รวม 216Kb)

• 18x18 hardware multiplier จำนวน 12 ชุด

• Digital Clock Manager (DCM) จำนวน 4 ชุด

• Digitally Controlled Impedance (DCI)

บอร์ดทดลองอเนกประสงค์รุ่น FPGA Discovery-III XC3S200 มีการจัดวางตำแหน่งการวางอุปกรณ์ด้านบนแสดงดังภาพที่ 33



<u>ภาพที่ 33</u> การจัดวางตำแหน่งการวางอุปกรณ์ด้านบน

อุปกรณ์ในตัว Board FPGA ที่เกี่ยวข้องทางด้าน Input ที่ใช้ในการปรับค่าต่างๆมีดังนี้

1. DIP Switch (DIP SW)

เป็นชุดของสวิทซ์เลื่อนขนาดเล็กที่ใช้ป้อนข้อมูลเข้าสู่ FPGA โดยถ้าเลื่อนลง (Off) จะเป็น “1” ถ้าเลื่อนขึ้น (On) จะเป็น “0” DIP SW ทุกตัวจึงทำงานแบบ Active Low โดย DIP Switchจะใช้ในการปรับค่า Threshold ตั้งแต่ค่า 0-15

2. Push button switch (PB1 – PB5) เป็นสวิทซ์กดติดปล่อยดับที่ให้สัญญาณเอาท์พุทเป็นระดับลอจิก “0” เมื่อกดสวิทซ์ และเป็นระดับลอจิก “1” เมื่อปล่อยสวิทซ์ Push button switch ทุกตัวจึงทำงานแบบ Active Low โดย Push button จะใช้ในการ Reset ในการจับเวลาในการทำงานของ FPGA และในการปรับค่า Threshold จะใช้ DIP Switch 1-4 โดยมีเงื่อนไขตามตารางที่ 4

ตารางที่ 4 แสดง DIP Switch ที่ใช้ในการปรับค่า Threshold

DIP Switch				ค่า Threshold			
1	2	3	4				
On	On	On	On	0	0	0	0
On	On	On	Off	0	0	0	1
On	On	Off	On	0	0	1	0
On	On	Off	Off	0	0	1	1
On	Off	On	On	0	1	0	0
On	Off	On	Off	0	1	0	1
On	Off	Off	On	0	1	1	0
On	Off	Off	Off	0	1	1	1
Off	On	On	On	1	0	0	0
Off	On	On	Off	1	0	0	1
Off	On	Off	Off	1	0	1	1
Off	Off	On	On	1	1	0	0
Off	Off	On	Off	1	1	0	1
Off	Off	Off	On	1	1	1	0
Off	Off	Off	Off	1	1	1	1

ในการวัดค่าเวลาของการทำงานของ FPGA จะใช้ DIP Switch 7 และ 8 เพื่อดูเวลาที่แสดงผลแต่ละหลักโดยมีเงื่อนไขตามตารางที่ 5

ตารางที่ 5 แสดง DIP Switch เพื่อใช้จับเวลาการแสดงผลแต่ละหลัก

DIP Switch		หลักที่
7	**8**	
On	On	1
Off	On	2
On	Off	3
Off	Off	4

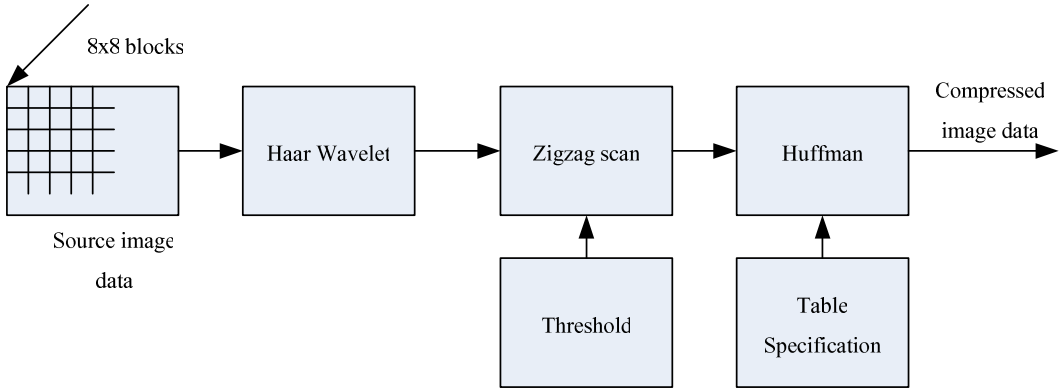
โดยค่าที่ได้จากการจับเวลาการทำงานของ FPGA จะเป็นเลขฐาน 16 และเมื่อต้องการวัดค่าใหม่ต้องทำการกด Reset ที่ PB1 (Push button switch) ของ Board FPGA

และสัญญาณภาพ Output ที่ได้จะเก็บไว้ใน File ที่ได้กำหนดไว้ซึ่งในการวิจัยครั้งนี้ได้กำหนดเป็น File สกุล .dwl เพื่อดูขนาดของFileที่ทำการบีบอัด

สำหรับในการพัฒนา Tools ที่นำมาใช้เป็นอุปกรณ์ในการทดสอบนี้ ได้พัฒนาโดยใช้ ภาษา Delphi และ VHDL โดยในส่วนของ VHDL ได้โปรแกรมลงบนชิพ PFGA รุ่น Discovery-III XC32S200F และใช้ภาษา Delphi ทำหน้าที่ อ่านข้อมูลFrame ของสัญญาณภาพ Decode สัญญาณภาพและแสดงผล ส่วนชิพ FPGA จะทำหน้าที่ในส่วนของการบีบอัดข้อมูลสัญญาณภาพ

โดยการยึดหลักในการออกแบบให้เป็นไปในแนวทางการศึกษาเพื่อศึกษาหาแนวทางในการลดขนาดของสัญญาณภาพ Video ด้วย Wavelet นี้ จึงได้พิจารณาจากหลักการทำงานที่เป็นอิสระต่อกันมากที่สุดเป็นอันดับแรกดัง Block Diagram ที่แสดงในภาพที่ 33 ที่การทำงานในแต่ละขั้นตอนจะเป็นอิสระต่อกันมากที่สุด

หลักการทำงานของโปรแกรมบีบอัดที่พัฒนาด้วย VHDL



ภาพที่ 34 แสดง Block Diagram การทำงานของ FPGA ที่ใช้ในการทดลอง

จาก Block Diagram ที่แสดงในภาพที่ 34 นี้จะเป็นได้ว่าเป็นกระบวนการบีบอัดสัญญาณที่มีขั้นตอนที่ง่าย ไม่มีความซับซ้อนต่อการทำงาน คือจะใช้หลักของการบีบอัดสัญญาณภาพในแต่ละ Frame ของสัญญาณภาพ แล้วนำไปแสดงภาพของสัญญาณทีละภาพ โดยที่ในแต่ละขั้นตอนก็จะมีความเป็นอิสระในการทำงานต่อกันซึ่งก็มีข้อดีก็คือเวลาที่ใช้ในการบีบอัดสัญญาณจะมีความรวดเร็ว เพราะว่ามีขั้นตอนที่ต้องทำงานน้อย ดังจะได้อธิบายหลักการทำงานของต่อละขั้นตอนดังนี้

- Source Image Data

ข้อมูล Frame ของภาพจากสัญญาณภาพสี (RGB) ที่มีขนาดของสัญญาณข้อมูลเป็น 24 bpp (bit per pixel) โดยทำการส่งข้อมูลทีละ 8×8 block ขนาด 64 bytes

- Haar Wavelet

ทำการแปลงเมตริกซ์ของภาพโดยใช้ Haar wavelet transform ของรูป Matrix ที่มีขนาด $n \times n$ ในที่นี้จะยกตัวอย่าง 8×8

โดยคำนวณหาค่า Average และ Difference จากสูตร

$$Average = (l + r) / 2$$

$$Difference = Average - l = r - Average$$

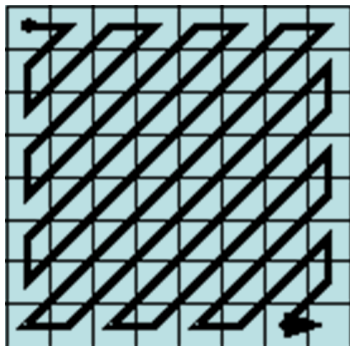
- l คือ ตัวซ้ายมือของ Matrix

- r คือ ตัวถัดจากตัว l

จะมีการคำนวณทั้งหมด 3 ครั้ง เพราะ $2^3 = 8$ ถ้า string มีความยาว 2^k แสดงว่าต้องมีการหาค่า Average และ Difference ทั้งหมด k ครั้ง ค่า average ที่หาได้จะถูกเก็บไว้ที่ตำแหน่งแรก(และถัดไป)ของส่วนที่ 1 ค่าที่ได้เรียกว่า approximation coefficient ของ n/2 เช่น
n = 8 ก็จะได้ว่า 8/2 = 2 ส่วน แต่ละส่วนมี 4 ตำแหน่ง และ difference ก็จะเก็บที่ตำแหน่งแรก (และถัดไป) ของส่วนที่ 2 ค่าที่ได้เรียกว่า detail coefficients เช่น แถวแรกของ Matrix 8 × 8

- Zigzag

การอ่านข้อมูลแบบ Zigzag เป็นการทำให้ค่าที่เป็นศูนย์อยู่เรียงกัน เพื่อที่จะทำให้ข้อมูลที่ต้องการเข้ารหัสมีจำนวนน้อยลง โดยจะมีรูปแบบการเรียงลำดับแสดงดังภาพที่ 35



ภาพที่ 35 การเรียงลำดับของ Zigzag

- Threshold

เป็นการกำหนดให้ค่าตัวเลขที่มีค่าน้อยกว่า Threshold มีค่าเป็นศูนย์เป็นการปรับลดค่าสัมประสิทธิ์เพื่อลดจำนวนบิตของข้อมูลให้ลดลง แต่จะทำให้เกิดการสูญเสียข้อมูลบางส่วนไปโดยจะทำการปรับค่า Threshold ที่ตัว DIP SW 1-4

- Huffman

การเข้ารหัสข้อมูลแบบ Huffman นั้นจะแบ่งการเข้ารหัสข้อมูลเป็น 2 ส่วนดังการเข้ารหัสข้อมูลของค่า DC และการเข้ารหัสข้อมูลของค่า AC

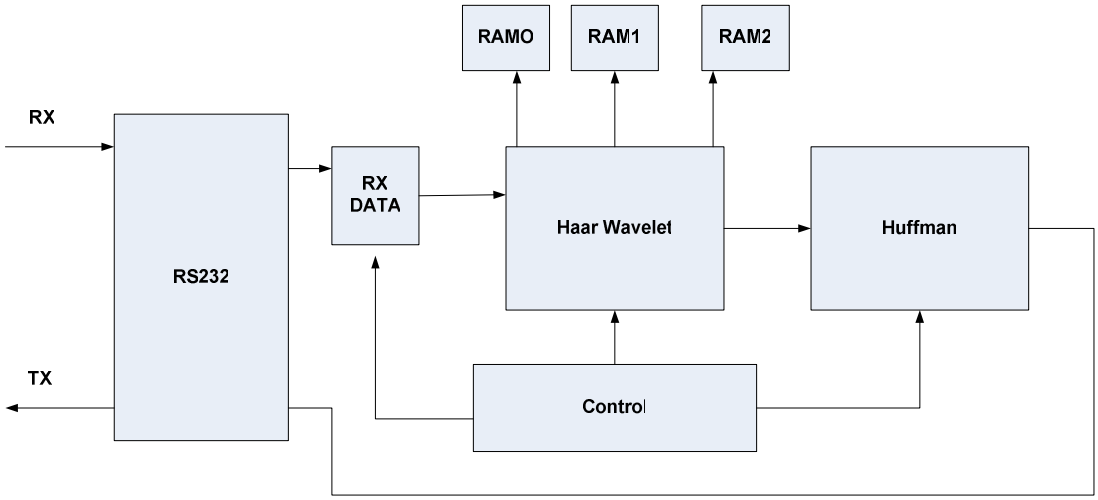
ในการเข้ารหัสค่า DC ของแถวข้อมูลนั้นได้จากข้อมูลตัวแรกที่จุด (0, 0) ของบล็อกพิกเซล โดยค่าที่ได้ใช้ในการแทนข้อมูลตามค่า Category ของข้อมูล เมื่อได้ค่า Category แล้วก็สามารถหารหัสข้อมูลได้จากตารางค่า DC ของ Huffman

การเข้ารหัสค่า AC (ข้อมูลตัวที่ 2 ถึงตัวที่ 64 ในแถวข้อมูล) การเข้ารหัส AC นั้นจะทำการเข้ารหัสเฉพาะค่า AC ที่มีค่าไม่เป็นศูนย์เท่านั้น ส่วนค่า AC ที่เป็นศูนย์จะอาศัยการเข้ารหัสแบบ Run length ซึ่งการหาค่า Category ของข้อมูล AC จะเทียบหาเช่นเดียวกับค่า DC เนื่องจากข้อมูลที่นำมาเข้ารหัสจะมีค่า Run length เข้ามาเกี่ยวข้องซึ่งสามารถหารหัสข้อมูลจากตารางค่า AC ของ Huffman ได้โดยเทียบตามค่า Run/Category

หลังจากนั้นศึกษารายละเอียดระบบการทำงานของวงจรทั้งหมด จากนั้นสร้างเป็นโมเดลของวงจรโดยใช้ภาษา VHDL ในการบรรยายพฤติกรรมของวงจร (Behavioral) ในแต่ละส่วนของวงจรที่ได้แบ่งไว้และทำการสังเคราะห์โมเดลของวงจรต่างๆโดยใช้โปรแกรมช่วยในการสังเคราะห์ (Synthesis Tools)

Module FPGA

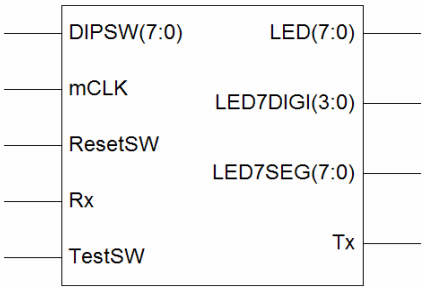
จากการ Synthesis โปรแกรม VHDL ที่เขียนจากหลักการทำงานที่ผ่านมาทำให้เกิด Module ต่าง ๆ ตาม Block diagram ดังนี้



ภาพที่ 36 แสดง Block Diagram ของ Module FPGA

- Module รวมของ FPGA ในการบีบอัดภาพ

เป็น Module โดยรวมทั้งหมดของการบีบอัดภาพ ซึ่งจะประกอบด้วยส่วนของ RS232, RX DATA, Haar Wavelet, Huffman, Control ดังแสดงในภาพที่ 37



ภาพที่ 37 แสดง Moduleรวมของ FPGA ในการบีบอัดภาพ

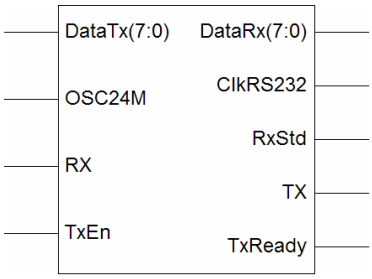
โดยมีการทำงานของขาสัญญาณต่างๆดังนี้

- DIPSW (7:0) Bit 0 – Bit 3 สำหรับกำหนดค่า Threshold, Bit 6 – Bit 7 เลือกหลักตัวจับเวลา
- mCLK Main Clock
- Reset SW Reset

- Rx รับสัญญาณจาก PC
- Test SW สำหรับ Test
- LED (7:0) LED Status หลอดที่ (4) Rx En, หลอดที่ (5) Haar En, หลอดที่ (6) Test หลอดที่ (7) Tx EN (Huffman)
- LED7DIGI (3:0) Seven Segment Common
- LED7SEG(7:0) Seven segment Data
- Tx ส่งออกไปยัง PC

- Module RS232

เป็น Module ที่ใช้ติดต่อส่งข้อมูลระหว่าง PC กับ FPGA แสดงดังภาพที่ 38



ภาพที่ 38 Module ของ RS232

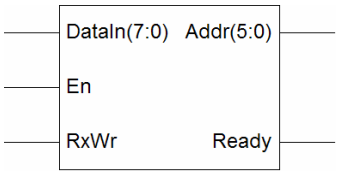
โดยมีการทำงานของขาสัญญาณต่างๆดังนี้

- DataTx(7:0) ข้อมูลที่จะส่ง
- OSC24M ความถี่ Clock ที่ใช้ 24 MHz
- Rx รับสัญญาณจาก PC
- TxEn Set เมื่อต้องการส่งข้อมูล
- DataRx(7:0) ข้อมูลที่จะรับ
- ClkRs232 Clock ของ Board rate 128,000 Hz UART
- RxStd Active “Low” เมื่อรับข้อมูลได้
- TX ส่งออกไปยัง PC

- TxReady ส่งข้อมูลเรียบร้อย (Hi) หรือ กำลังส่งข้อมูลอยู่ (Low)

- Module RX DATA

เป็น Module ที่รับข้อมูลจาก PC เพื่อทำการบีบอัดข้อมูล โดย Module นี้จะทำหน้าที่ตรวจสอบข้อมูลที่ส่งมาจาก PC ก่อนที่จะ Enable ให้ Module Haar Wavelet ทำงานต่อไป ภาพที่ 39 แสดง Module RX TX



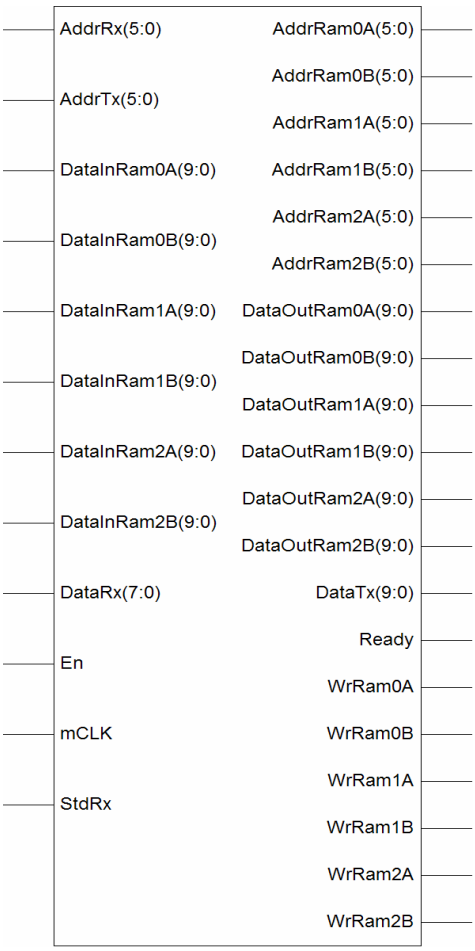
ภาพที่ 39 Module ของ RX DATA

โดยมีการทำงานของขาสัญญาณต่างๆดังนี้

- DataIn(7:0) ข้อมูลที่ได้รับจาก Module RS232
- En Active Module High
- RxWr เชื่อมต่อกับ RxStd ของ RS232 เพื่อจะนับจำนวนข้อมูล
- Addr(5:0) นับ Address Rx
- Ready Active “Hi” เมื่อรับข้อมูลได้ครบ 64 Bytes (8*8) หรือพร้อมรับข้อมูลชุดใหม่

- Module Haar wavelet

เป็น Module ที่ทำการแปลงเมตริกซ์ของภาพโดยใช้ Haar Wavelet transform เพื่อหาค่า Averaging และ Differencing เพื่อจะทำให้เมตริกซ์ภาพ เหมือนกับภาพต้นฉบับอย่างถูกต้อง



ภาพที่ 40 แสดง Module ของ Haar wavelet

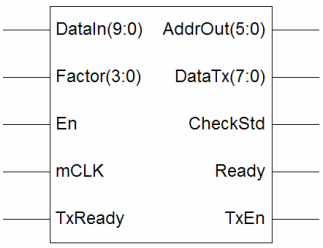
การทำงานของขาสัญญาณต่างๆดังนี้

- AddrRx(5:0) เชื่อมต่อกับ Module Rx Data สำหรับรับข้อมูล
- AddrTx(5:0) เชื่อมต่อกับ Module Huffman เพื่อนำข้อมูลออก
- DataInRam0A(9:0) Data Input RAM 0 Port A
- DataInRam0B(9:0) Data Input RAM 0 Port B
- DataInRam1A(9:0) Data Input RAM 1 Port A
- DataInRam1A(9:0) Data Input RAM 1 Port B
- DataInRam2A(9:0) Data Input RAM 2 Port A
- DataInRam2A(9:0) Data Input RAM 2 Port A
- DataRx(7:0) ข้อมูลที่ได้รับจาก RS 232

- En Enable เพื่อกำหนดให้ทำงาน Enable ทำ Haar Wavelet Disable ทำการรับส่งข้อมูล
- mCLK ความถี่ทำงาน 24 MHz
- StdRx เชื่อมต่อกับ RxStd ของ Module RS232
- AddrRam0A(5:0) Address RAM 0 Port A
- AddrRam0B(5:0) Address RAM 0 Port B
- AddrRam1A(5:0) Address RAM 1 Port A
- AddrRam1B(5:0) Address RAM 1 Port B
- AddrRam2A(5:0) Address RAM 2 Port A
- AddrRam2B(5:0) Address RAM 2 Port B
- DataOutRam0A(9:0) Dataout RAM 0 Port A
- DataOutRam0B(9:0) Dataout RAM 0 Port B
- DataOutRam1A(9:0) Dataout RAM 1 Port A
- DataOutRam1B(9:0) Dataout RAM 1 Port B
- DataOutRam2A(9:0) Dataout RAM 2 Port A
- DataOutRam2B(9:0) Dataout RAM 2 Port B
- DataTx(9:0) ข้อมูลส่งออกไปยัง Module Huffman
- Ready Active เมื่อทำ Haar Wavelet สำเร็จ
- WrRamoA สัญญาณเขียนข้อมูลลง RAM 0 Port A
- WrRamoB สัญญาณเขียนข้อมูลลง RAM 0 Port B
- WrRam1A สัญญาณเขียนข้อมูลลง RAM 1 Port A
- WrRam1A สัญญาณเขียนข้อมูลลง RAM 1 Port B
- WrRam2A สัญญาณเขียนข้อมูลลง RAM 2 Port A
- WrRam2A สัญญาณเขียนข้อมูลลง RAM 2 Port B

- Module Huffman

เป็นModule ที่ทำการเข้ารหัสข้อมูลโดย Module นี้จะทำการ Zigzag ข้อมูล และทำการเข้ารหัสตามตาราง Huffman แสดงดังภาพที่ 41



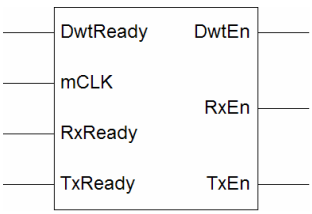
ภาพที่41 Module Huffman

การทำงานของขาสัญญาณต่างๆดังนี้

- DataIn(9:0) Data In จาก Haar Wavelet
- Factor จาก DIPSW ในการปรับ Threshold
- En กำหนดให้ Huffman
- mCLK Main Clock
- TxReady ตรวจสอบว่าพร้อมส่งข้อมูลหรือไม่จาก RS232
- AddrOut(5:0) ตำแหน่งของหน่วยความจำที่ต้องการอ่านตามตาราง Zigzag
- DataTx(7:0) ข้อมูลที่ทำ Huffman แล้ว เพื่อส่งออกไปยัง RS232
- CheckStd สำหรับทำการนับให้ตัวจับเวลา
- Ready เมื่อทำ Huffman และส่งข้อมูลเสร็จ
- Tx En เชื่อมต่อ RS232 เพื่อกำหนดจังหวะการส่ง

- Module Control

ทำหน้าที่ควบคุมจังหวะการทำงานในส่วนต่างๆ เพื่อให้การทำงานของระบบถูกต้อง ดังแสดงในภาพที่ 42



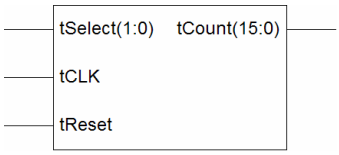
ภาพที่ 42 แสดง Module ของ Control

การทำงานของขาสัญญาณต่างๆดังนี้

- DwtReady ตรวจสอบว่าทำ Wavelet เรียบร้อย
- mCLK Main Clock
- RxReady ตรวจสอบว่ารับข้อมูลเรียบร้อย
- Txready ตรวจสอบว่าส่งข้อมูลเรียบร้อย
- DwtEn กำหนดให้ Haar Wavelet ทำงาน
- RxEn กำหนดให้รับข้อมูล
- TxEn กำหนดให้ Huffman ทำงาน

- Module Time count

เป็น Module ที่ใช้ในการจับเวลาการทำงานของ FPGA ในการบีบอัดภาพดังแสดงในภาพที่ 43



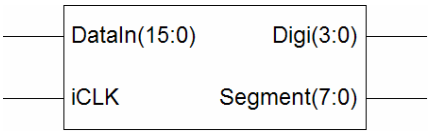
ภาพที่ 43 Module ของ Time Count

การทำงานของขาสัญญาณต่างๆดังนี้

- tSelect(1:0) เลือกหลักในการแสดงข้อมูลจาก DIP SW
- tCLK Clock จากการทำงานในส่วนของ Haar Wavelet และ Huffman เมื่อนับ
- tReset Reset ตัวนับ
- tCount ข้อมูลของหลักที่เลือกตาม tSelect

- Module Segment

เป็น Module ที่ใช้ในการแสดงเวลาการทำงานของ FPGA ดังแสดงในภาพที่ 43



ภาพที่ 44 Module Segment

การทำงานของขาสัญญาณต่างๆดังนี้

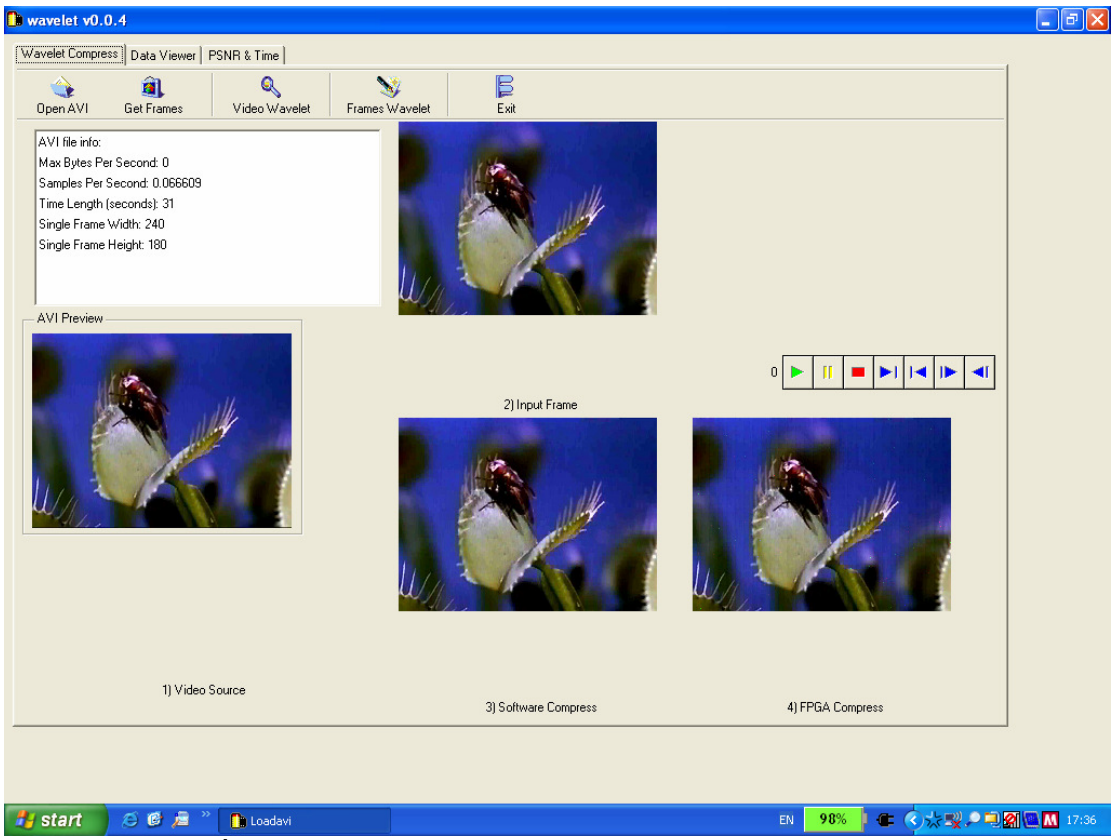
- DataIn(15:0) ข้อมูลตามหลักที่เลือก
- iCLK ความถี่จาก RS232 128,000 Hz
- Digi(3:0) Seven Segment Common
- Segment(7:0) Seven Segment Data

ผลและวิจารณ์

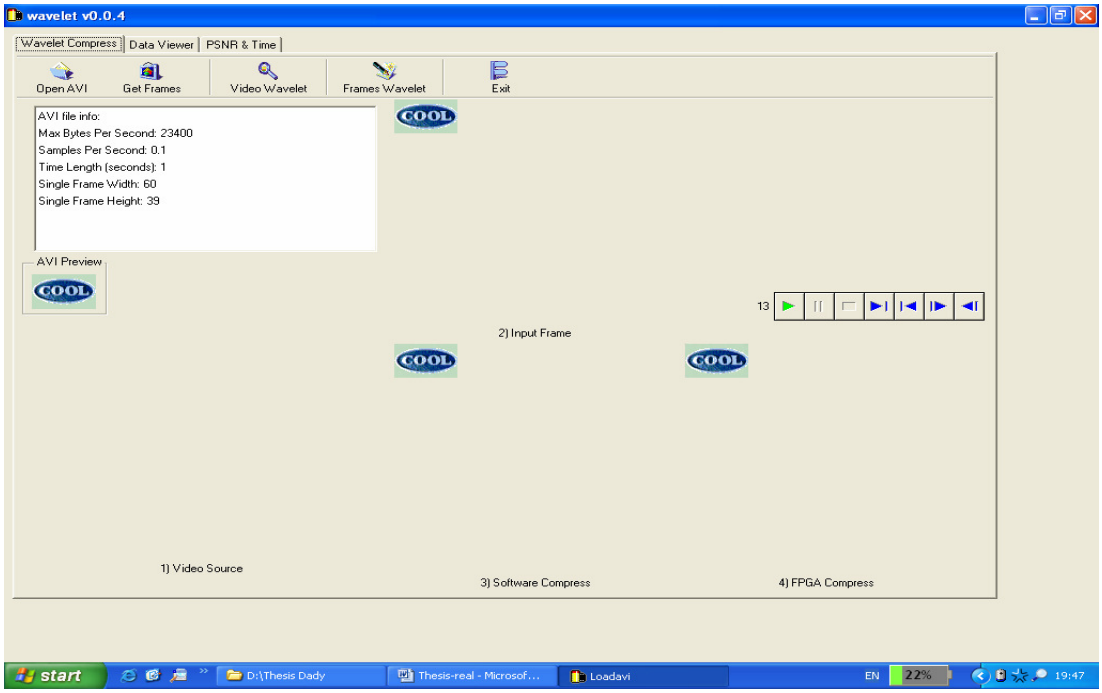
ในการทดลองเพื่อศึกษาถึงการลดของสัญญาณภาพด้วย Wavelet Technique นี้จะทำการทดสอบถึงคุณภาพของสัญญาณภาพที่ได้ (PSNR) และประสิทธิภาพของการบีบอัด (Compress Ratio) โดยมีวิธีการทดลองดังนี้

1. การวัดคุณภาพของสัญญาณ (PSNR:Peak Signal-to-Noise Ratio)
   ทำการส่งไฟล์วิดีโอ Cool.AVI ขนาด 60×39 Pixel จำนวน 13 เฟรม ไฟล์วิดีโอ Counter.AVI ขนาด 128 ×128 Pixel จำนวน 10 เฟรม และ Findcom.AVI ขนาด 48×45 Pixel จำนวน 8 เฟรมโดยวัดค่า PSNR แยกแต่ละสี คือ สีแดง สีเขียว และสีน้ำเงิน รวมทั้งค่าเฉลี่ยของทุกสี
2. การหาค่าประสิทธิภาพในการบีบอัดสัญญาณภาพว่าสามารถลดขนาดของข้อมูลได้จำนวนมากน้อยเท่าไหร่โดยประสิทธิภาพในการลดข้อมูลได้จะคิดในอัตราส่วนที่เป็นเปอร์เซ็นต์
3. ทำการตรวจสอบเวลาที่ใช้ในการบีบอัดสัญญาณภาพของชิพ FPGA และเวลาที่ใช้ในการทำงานทั้งหมด เพื่อที่จะวัดผลว่า FPGA มีการทำงานที่รวดเร็วเพียงใด
4. ทำการ Plot กราฟของข้อมูลระหว่างค่า PSNR ของแต่ละเฟรม เพื่อทำการเปรียบเทียบคุณภาพของสัญญาณแต่ละ Frame ว่ามีคุณภาพแตกต่างกันมากน้อยเพียงใด
5. ตรวจสอบใช้ทรัพยากรภายในชิพ FPGA ว่ามีการใช้งานไปมากน้อยเพียงใดเหมาะสมกับลักษณะงานหรือไม่

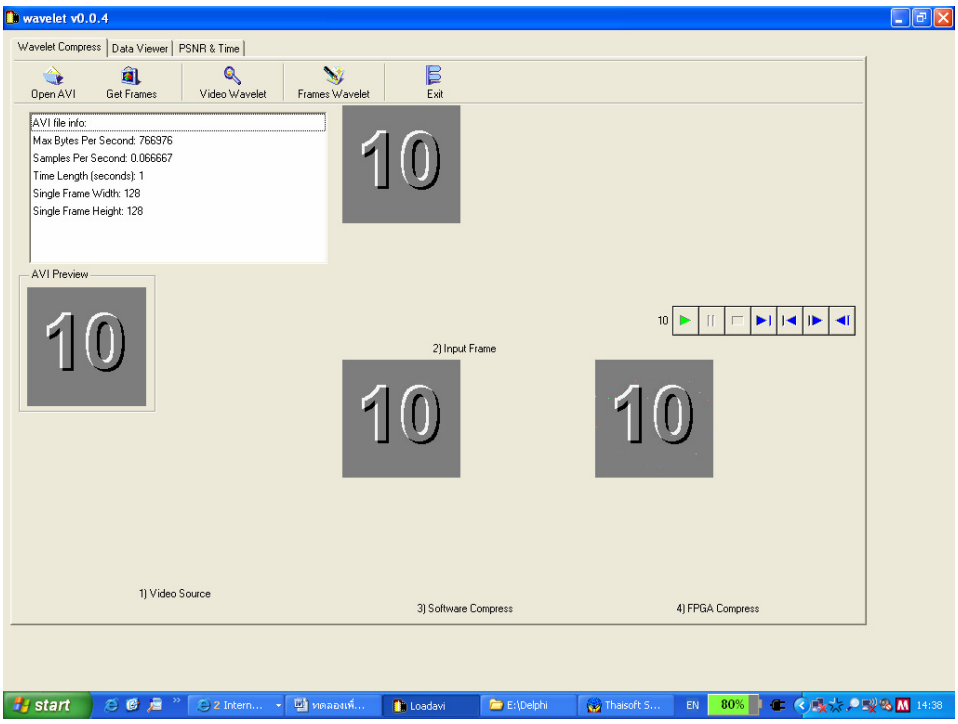
การทดสอบการลดขนาดของสัญญาณภาพโดยนำวงจรที่ทำการออกแบบมาเชื่อมต่อเครื่องคอมพิวเตอร์ทางพอร์ตอนุกรม และส่งไฟล์ข้อมูลที่ต้องการทดสอบจากเครื่องคอมพิวเตอร์ไปทำการบีบอัดข้อมูลในวงจรที่ออกแบบบน FPGA โดยนำมาเก็บไว้ในรูปแบบของไฟล์บนคอมพิวเตอร์อีกครั้ง ซึ่งแสดงสัญญาณภาพที่ใช้ทดลองดังภาพที่ 45 - 48



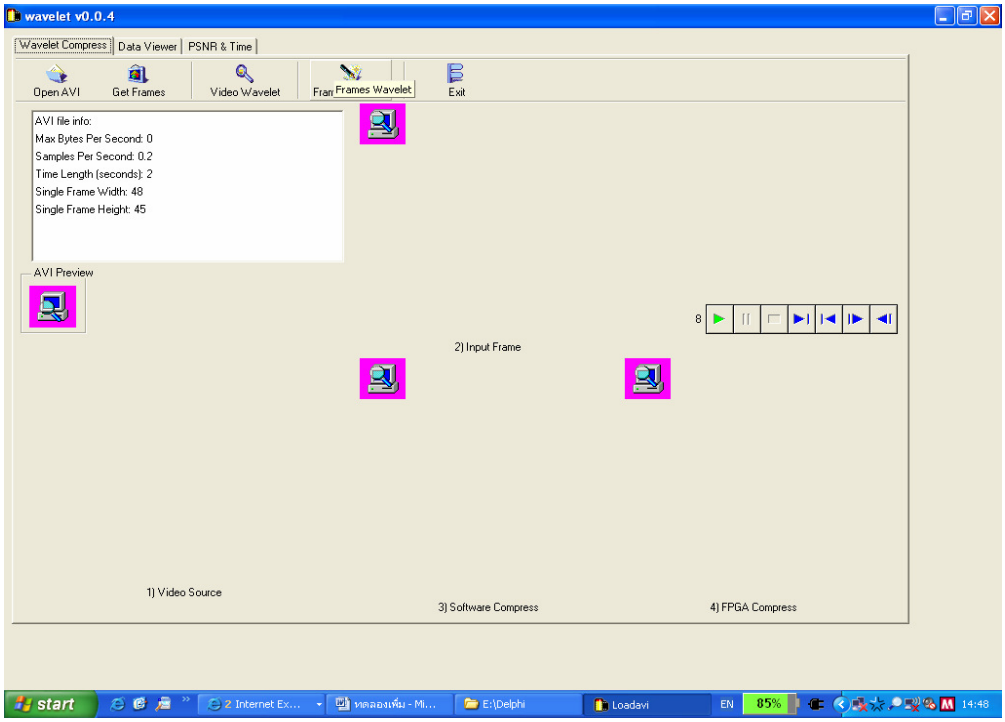
ภาพที่ 45 แสดงภาพที่ได้จากการบีบอัดสัญญาณภาพ ลักษณะเป็นเฟรม



ภาพที่ 46 แสดงภาพที่ได้จากการบีบอัดสัญญาณภาพ Cool.AVI



ภาพที่ 47 แสดงภาพที่ได้จากการบีบอัดสัญญาณภาพ Counter.AVI



ภาพที่ 48 แสดงภาพที่ได้จากการบีบอัดสัญญาณภาพ Findcom.AVI

ในการทดสอบและวัดผล แบ่งออกเป็น 3 ส่วน ดังนี้

1.การวัดคุณภาพของสัญญาณ (PSNR:Peak Signal-to-Noise Ratio)

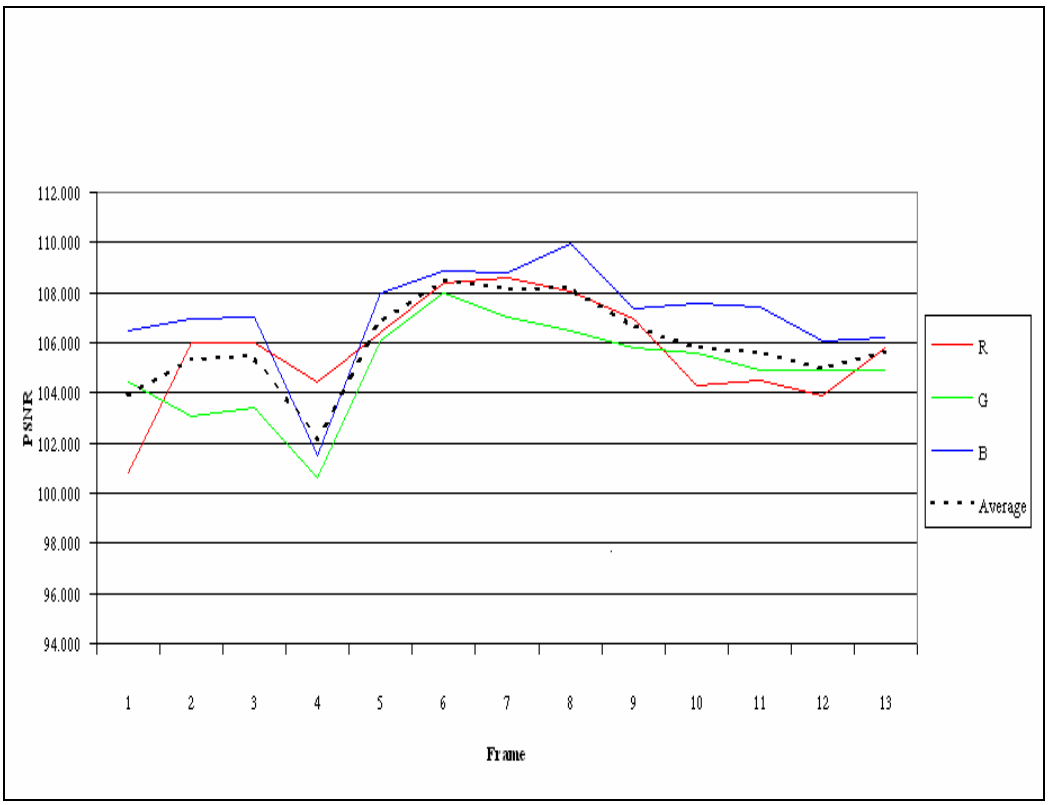
1.1ทำการส่งไฟล์วิดีโอ Cool.AVI ขนาด 60x39 pixel จำนวน 13 เฟรม โดยวัดค่า PSNR แยกแต่ละสี คือ สีแดง สีเขียว และสีน้ำเงิน รวมทั้งค่าเฉลี่ยของทุกสี ซึ่งได้ผลการทดลองตามแนวทางที่ศึกษาข้างต้น แสดงข้อมูลของค่า PSNR ของแต่ละเฟรม ดังตารางที่ 6-8 และแสดงกราฟของค่า PSNR ดังภาพที่ 49-51 ซึ่งแสดงค่า PSNR ตาม Threshold เท่ากับ 0, 8 และ 15 ที่ใช้ในการทดลองครั้งนี้

ตารางที่ 6 แสดงข้อมูลของค่า PSNR ไฟล์ Cool ของแต่ละ Frame (Threshold = 0)

Frame	ค่า PSNR (dB)			
	R	G	B	Avg.
1	100.827	104.399	106.460	103.895
2	105.974	103.097	106.959	105.343
3	106.002	103.396	106.999	105.466
4	104.449	100.627	101.513	102.196
5	106.412	106.093	107.953	106.819
6	108.411	107.991	108.867	108.423
7	108.578	107.037	108.822	108.146
8	108.025	106.509	109.963	108.166
9	106.987	105.795	107.358	106.713
10	104.312	105.595	107.571	105.826
11	104.487	104.915	107.435	105.612
12	103.899	104.937	106.087	104.974
13	105.774	104.937	106.221	105.644

จากตารางที่ 6 แสดงคุณภาพของสัญญาณที่การปรับ Threshold = 0 ในการวัดคุณภาพสัญญาณจะแยกค่าแต่ละสีจากค่าที่ได้จะเห็นว่าแต่ละสีจะมีคุณภาพของสัญญาณที่ใกล้เคียงกัน และแต่ละ Frame คุณภาพของสัญญาณก็จะไม่เท่ากันขึ้นอยู่กับภาพที่ทำการบีบอัด ซึ่งในการวัดคุณภาพ

ของสัญญาณที่ Threshold = 0 นี้จะไม่ทำการลดทอนข้อมูลค่าเนื่องจากตั้งค่า Threshold = 0 ทำให้ข้อมูลในเมตริกซ์ภาพไม่มีค่าใดโดนตัดไป



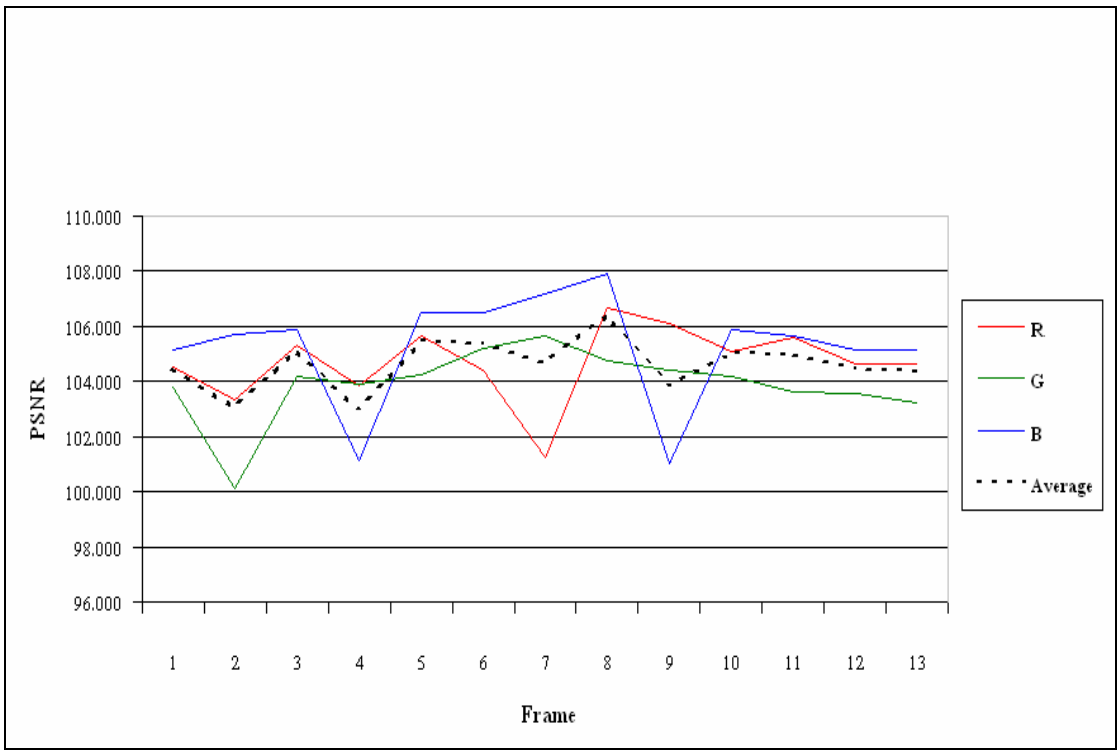
ภาพที่ 49 แสดงกราฟของค่า PSNR ไฟล์ Cool ของแต่ละ Frame (Threshold = 0)

กราฟในภาพที่ 49 เป็นการนำข้อมูลจากตารางที่ 6 มาทำการเขียนกราฟ จะเห็นได้ว่าในการปรับ Threshold ที่ 0 ค่าคุณภาพของสัญญาณ ของสีน้ำเงิน, แดง, และเขียว จะมีค่าคุณภาพของสัญญาณที่มากน้อยลดหลั่นกันไปในแต่ละ Frame โดย คุณภาพของสัญญาณสีน้ำเงินจะดีกว่า สีแดง และ สีเขียว ซึ่งในการปรับ Threshold ที่ 0 จะไม่มีการลดทอนข้อมูลแต่คุณภาพของสัญญาณที่แตกต่างกันเนื่องจากลักษณะของภาพที่มาทำการบีบอัด

ตารางที่ 7 แสดงข้อมูลของค่า PSNR ไฟล์ Cool ของแต่ละ Frame (Threshold = 8)

Frame	ค่า PSNR (dB)			
	R	G	B	Avg.
1	104.523	103.817	105.144	104.495
2	103.338	100.093	105.707	103.046
3	105.301	104.186	105.901	105.129
4	103.824	103.904	101.143	102.957
5	105.658	104.257	106.524	105.480
6	104.421	105.176	106.528	105.375
7	101.247	105.665	107.187	104.700
8	106.652	104.770	107.893	106.438
9	106.107	104.405	101.006	103.839
10	105.071	104.171	105.896	105.046
11	105.587	103.602	105.677	104.955
12	104.628	103.545	105.154	104.442
13	104.628	103.228	105.154	104.337

ในตารางที่ 7 เป็นการแสดงคุณภาพของสัญญาณที่ค่า Threshold = 8 โดยคุณภาพของสัญญาณยังใกล้เคียงกับ Threshold = 0



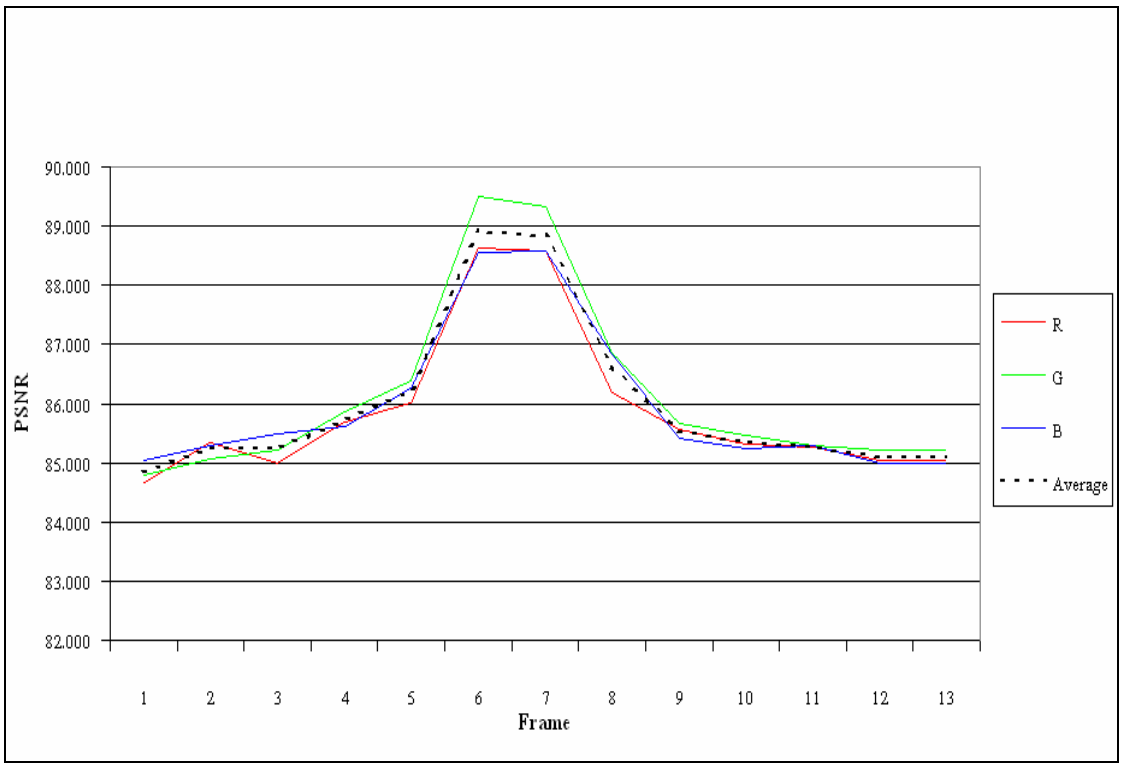
ภาพที่ 50 แสดงกราฟของค่า PSNR ไฟล์ Cool ของแต่ละ Frame (Threshold = 8)

จากกราฟในภาพที่ 50 ปรับค่า Threshold = 8 คุณภาพของสัญญาณสีน้ำเงินยังคงมากกว่า สีแดง และสีเขียว แสดงว่าข้อมูลสีแดงและสีเขียวได้ถูกลดทอนไปมากกว่า โดย Threshold = 8 จะตัดข้อมูลที่มีค่าตัวเลขน้อยกว่า 8 ให้มีค่าเป็นศูนย์

ตารางที่ 8 แสดงข้อมูลของค่า PSNR ไฟล์ Cool ของแต่ละ Frame (Threshold = 15)

Frame	ค่า PSNR (dB)			
	R	G	B	Avg.
1	84.679	84.779	85.043	84.834
2	85.346	85.073	85.281	85.233
3	84.993	85.219	85.493	85.235
4	85.686	85.854	85.626	85.722
5	86.021	86.377	86.252	86.217
6	88.628	89.496	88.557	88.894
7	88.570	89.338	88.568	88.825
8	86.198	86.856	86.839	86.631
9	85.570	85.663	85.416	85.550
10	85.311	85.469	85.251	85.344
11	85.255	85.286	85.281	85.274
12	85.029	85.208	84.999	85.079
13	85.033	85.208	84.999	85.080

ตารางที่ 8 เป็นการปรับ Threshold = 15 จะเห็นได้ชัดว่าคุณภาพของสัญญาณได้ลดลงมา เนื่องจากการปรับ Threshold ที่ 15 จะเป็นการลดทอนข้อมูลค่อนข้างมาก แต่การบีบอัดจะบีบอัดข้อมูลได้เยอะซึ่งจะกล่าวในประสิทธิภาพการบีบอัดต่อไป



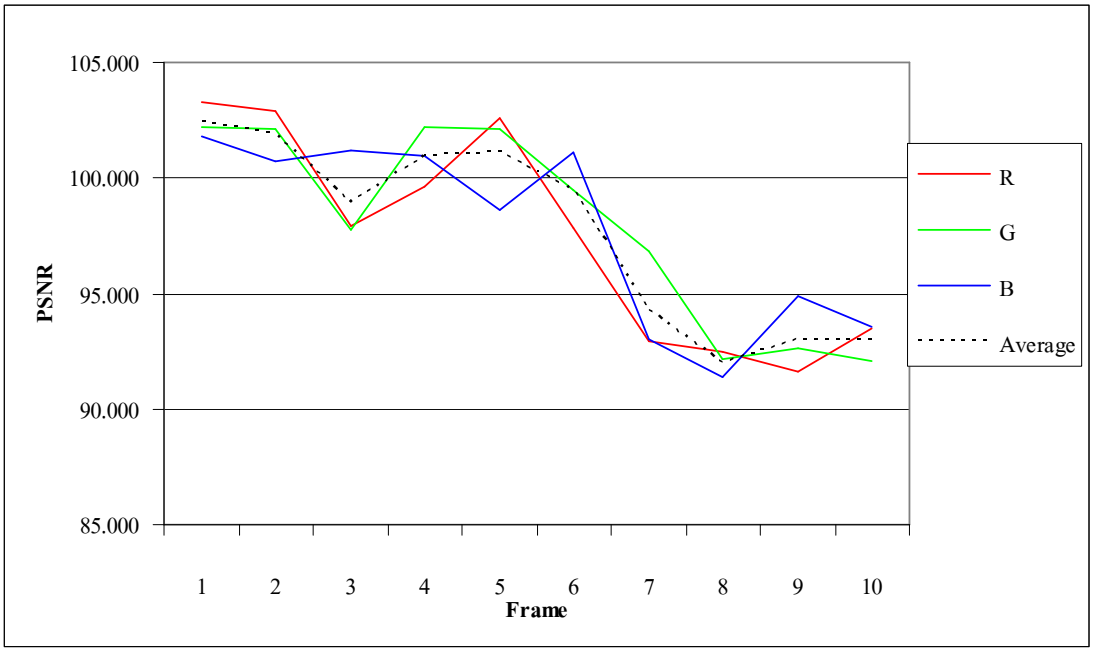
ภาพที่ 51 แสดงกราฟของค่า PSNR ไฟล์ Cool ของแต่ละ Frame (Threshold = 15)

ภาพที่ 51 จะเห็นว่าคุณภาพของสัญญาณสีเขียวจะมีมากกว่า สีน้ำเงินและสีแดง แสดงว่าในการปรับ Threshold = 15 สีเขียวถูกลดทอนข้อมูลน้อยกว่า

1.2 ทำการส่งไฟล์วีดีโอ Counter.AVI ขนาด 128×128 Pixel จำนวน 10 เฟรม โดยวัดค่า PSNR แยกแต่ละสี คือ สีแดง สีเขียว และสีน้ำเงิน รวมทั้งค่าเฉลี่ยของทุกสี โดยแสดงข้อมูลของค่า PSNR ของแต่ละเฟรม ตามตารางที่ 9-11 และแสดงกราฟของค่า PSNR ดังภาพที่ 52-54 ซึ่งแสดงค่า PSNR ตาม Threshold เท่ากับ 0, 8 และ 15

ตารางที่ 9 แสดงข้อมูลของค่า PSNR ไฟล์ Counter ของแต่ละ Frame (Threshold = 0)

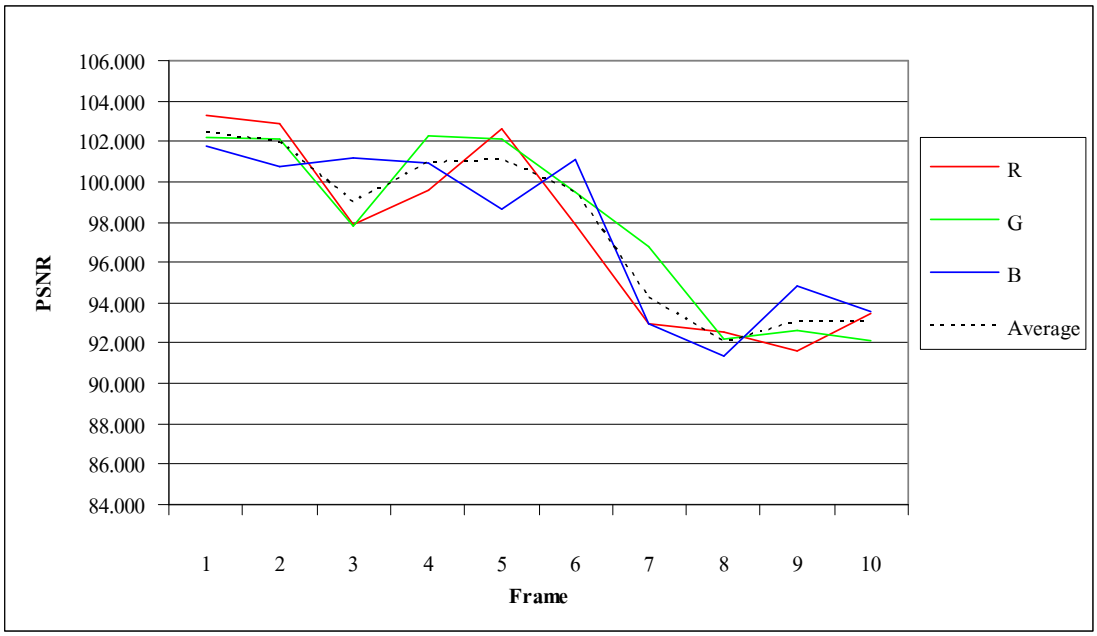
Frame	ค่า PSNR (dB)			
	R	G	B	Avg.
1	103.309	102.197	101.771	102.426
2	102.887	102.106	100.712	101.902
3	97.919	97.773	101.157	98.950
4	99.595	102.235	100.915	100.915
5	102.582	102.107	98.623	101.104
6	97.853	99.492	101.110	99.485
7	92.954	96.819	93.004	94.259
8	92.504	92.176	91.381	92.020
9	91.642	92.594	94.850	93.029
10	93.464	92.099	93.546	93.036



ภาพที่ 52 แสดงกราฟของค่า PSNR ไฟล์ Counter ของแต่ละ Frame (Threshold = 0)

ตารางที่ 10 แสดงข้อมูลของค่า PSNR ไฟล์ Counter ของแต่ละ Frame (Threshold = 8)

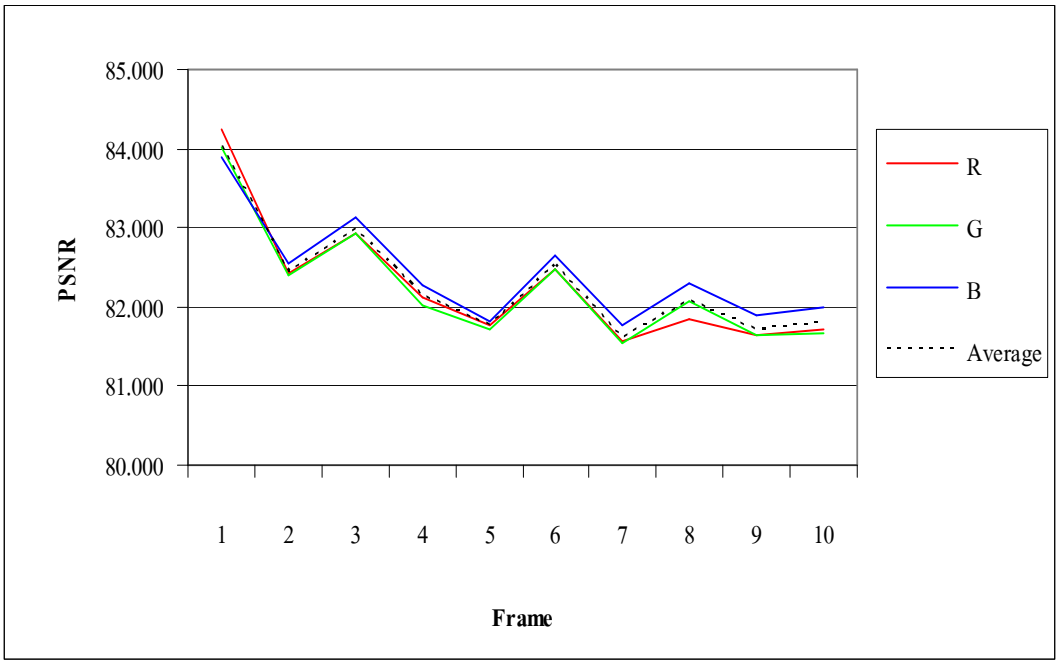
Frame	ค่า PSNR (dB)			
	R	G	B	Avg.
1	103.309	102.197	101.771	102.426
2	102.887	102.106	100.712	101.902
3	97.919	97.773	101.157	98.950
4	99.595	102.235	100.915	100.915
5	102.582	102.107	98.623	101.104
6	97.853	99.492	101.110	99.485
7	92.954	96.819	93.004	94.259
8	92.504	92.176	91.381	92.020
9	91.642	92.594	94.850	93.029
10	93.464	92.099	93.543	93.035



ภาพที่ 53 แสดงกราฟของค่า PSNR ไฟล์ Counter ของแต่ละ Frame (Threshold = 8)

ตารางที่ 11 แสดงข้อมูลของค่า PSNR ไฟล์ Counterของแต่ละ Frame (Threshold = 15)

Frame	ค่า PSNR (dB)			
	R	G	B	Avg.
1	84.242	84.016	83.900	84.053
2	82.413	82.397	82.552	82.454
3	82.932	82.918	83.120	82.990
4	82.120	82.021	82.284	82.142
5	81.762	81.718	81.820	81.767
6	82.478	82.480	82.639	82.532
7	81.568	81.530	81.759	81.619
8	81.834	82.072	82.286	82.064
9	81.633	81.634	81.892	81.720
10	81.729	81.667	81.996	81.797

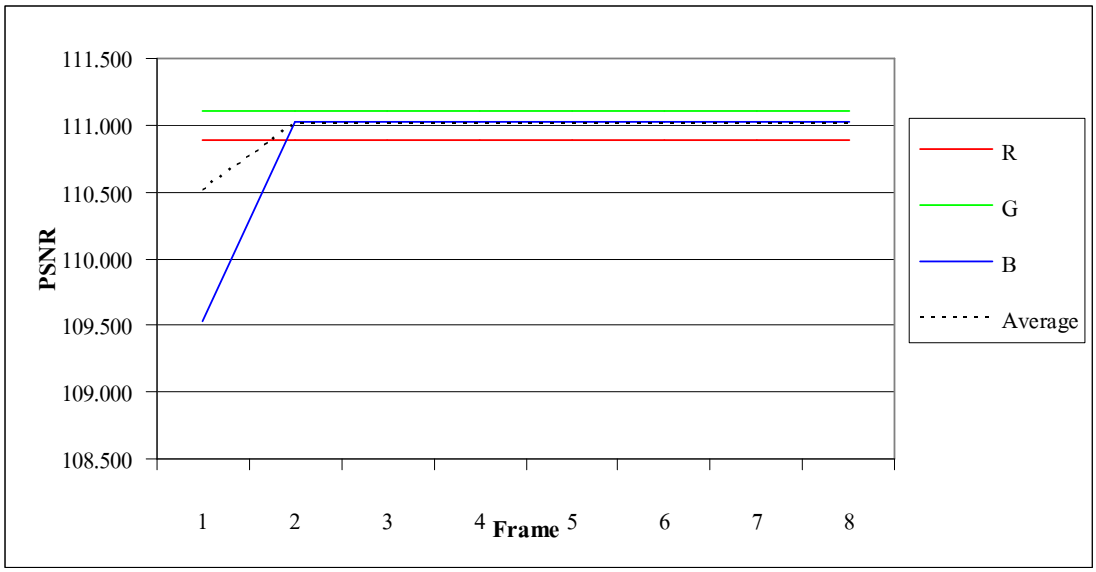


ภาพที่ 54 แสดงกราฟของค่า PSNR ไฟล์ Counter ของแต่ละ Frame (Threshold = 15)

1.3 ทำการส่งไฟล์วิดีโอ Findcom.AVI ขนาด 48×45 Pixel จำนวน 8 เฟรม โดยวัดค่า PSNR แยกแต่ละสี คือ สีแดง สีเขียว และสีน้ำเงิน รวมทั้งค่าเฉลี่ยของทุกสี โดยแสดงข้อมูลของค่า PSNR ของแต่ละเฟรม ตามตารางที่ 12-14 และแสดงกราฟของค่า PSNR ดังภาพที่ 55-57 ซึ่งแสดงค่า PSNR ตาม Threshold เท่ากับ 0, 8 และ 15

ตารางที่ 12 แสดงข้อมูลของค่า PSNR ไฟล์ Findcom ของแต่ละ Frame (Threshold = 0)

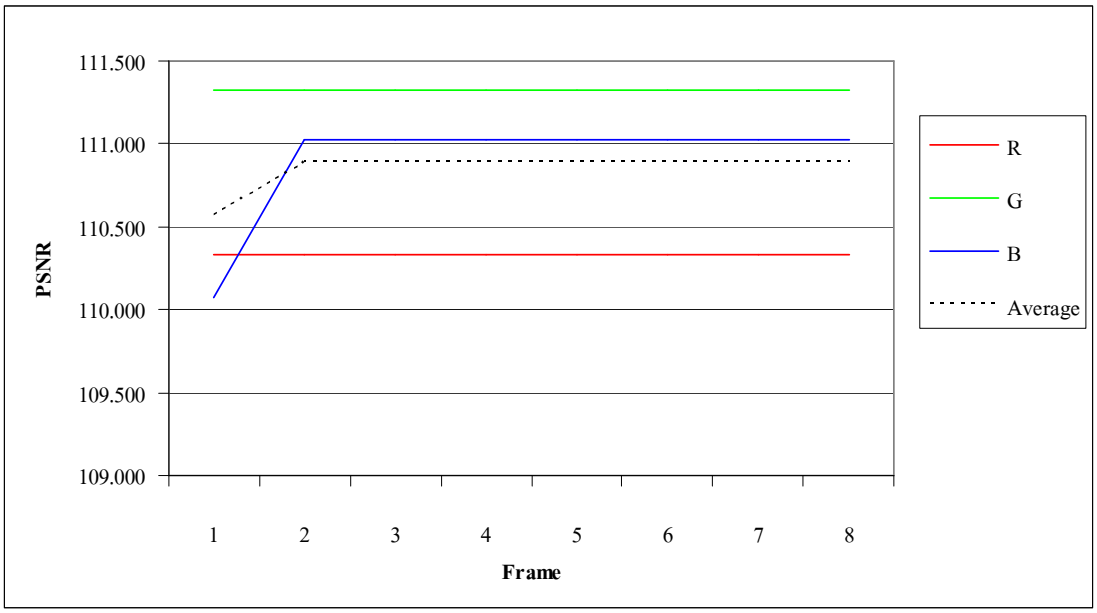
Frame	ค่า PSNR (dB)			
	R	G	B	Avg.
1	110.885	111.113	109.537	110.512
2	110.885	111.113	111.021	111.006
3	110.885	111.113	111.021	111.006
4	110.885	111.113	111.021	111.006
5	110.885	111.113	111.021	111.006
6	110.885	111.113	111.021	111.006
7	110.885	111.113	111.021	111.006
8	110.885	111.113	111.021	111.006



ภาพที่ 55 แสดงกราฟของค่า PSNR ไฟล์ Findcomของแต่ละ Frame (Threshold = 0)

ตารางที่ 13 แสดงข้อมูลของค่า PSNR ไฟล์ Findcom ของแต่ละ Frame (Threshold = 8)

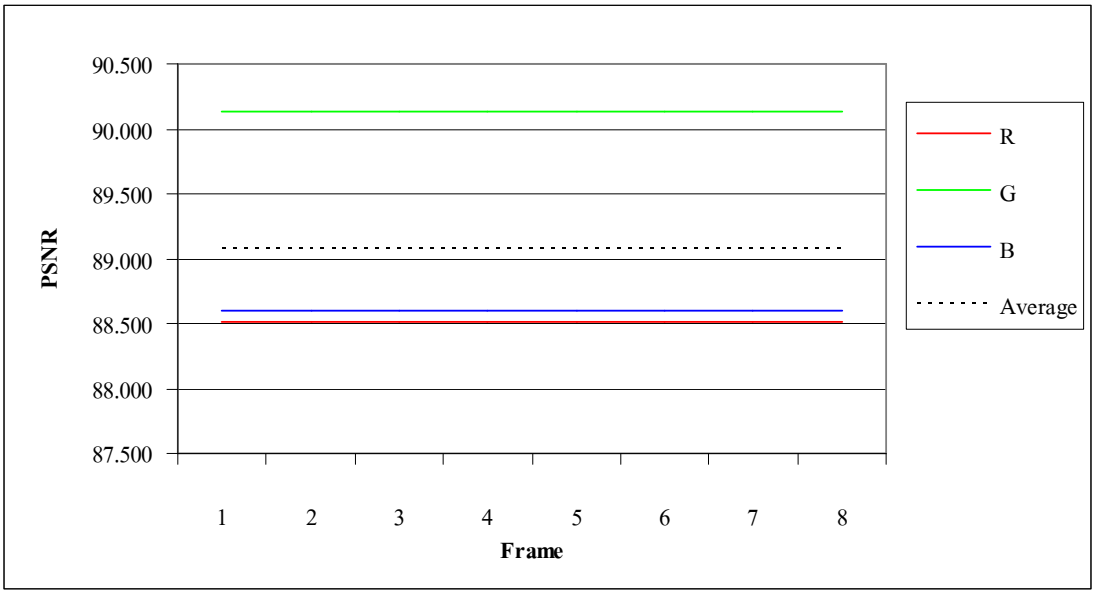
Frame	ค่า PSNR (dB)			
	R	G	B	Avg.
1	110.330	111.320	110.072	110.574
2	110.330	111.320	111.029	110.893
3	110.330	111.320	111.029	110.893
4	110.330	111.320	111.029	110.893
5	110.330	111.320	111.029	110.893
6	110.330	111.320	111.029	110.893
7	110.330	111.320	111.029	110.893
8	110.330	111.320	111.029	110.893



ภาพที่ 56 แสดงกราฟของค่า PSNR ไฟล์ Findcom ของแต่ละ Frame (Threshold = 8)

ตารางที่ 14 แสดงข้อมูลของค่า PSNR ไฟล์ Findcom ของแต่ละ Frame (Threshold = 15)

Frame	ค่า PSNR (dB)			
	R	G	B	Avg.
1	88.508	90.137	88.593	89.079
2	88.508	90.137	88.593	89.079
3	88.508	90.137	88.593	89.079
4	88.508	90.137	88.593	89.079
5	88.508	90.137	88.593	89.079
6	88.508	90.137	88.593	89.079
7	88.508	90.137	88.593	89.079
8	88.508	90.137	88.593	89.079



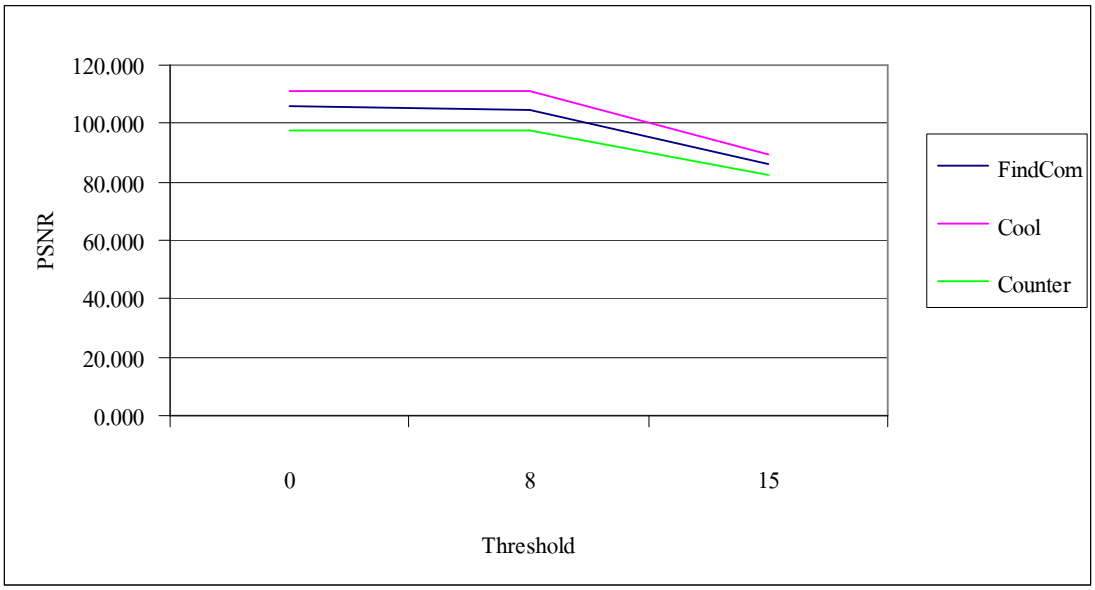
ภาพที่ 57 แสดงกราฟของค่า PSNR ไฟล์ Findcom ของแต่ละ Frame (Threshold = 15)

จากการทดลองในการวัดค่า PSNR ของทั้ง 3 ไฟล์ตัวอย่าง คือ Cool, Counter และFindcom

จะเห็นได้ว่าในการปรับค่าThresholdที่ต่ำคุณภาพของสัญญาณที่ได้ก็จะสูงขึ้น เนื่องจากสัมประสิทธิ์ของข้อมูลลดลงน้อย และเมื่อปรับค่า Threshold เพิ่มขึ้น คุณภาพของสัญญาณก็จะลดลงไปตามการปรับค่า Threshold ที่เพิ่มขึ้น จากการทดลองค่าคุณภาพของสัญญาณของทั้ง 3 ไฟล์ตัวอย่างในแต่ละ Threshold สามารถนำมาเปรียบเทียบได้ดังแสดงในตารางที่ 15 และภาพที่ 58

ตารางที่ 15 เปรียบเทียบค่า PSNR ของ ไฟล์วีดีโอที่ Threshold ค่าต่างๆ

	ค่า PSNR (dB)		
File	**0**	**8**	**15**
Cool	105.940	104.634	85.994
FindCom	110.945	110.853	89.079
Counter	97.713	97.712	82.314



ภาพที่ 58 เปรียบเทียบค่า PSNR ของ ไฟล์วีดีโอที่ Threshold ค่าต่างๆ

จากตารางที่ 15 และ กราฟที่ 58 ได้ทำการเปรียบเทียบผลการปรับ Threshold ของทั้ง 3 ไฟล์วีดีโอที่ทำการทดลองนั้น คุณภาพของสัญญาณที่ได้จะมีค่าลดลงตามการปรับ Threshold ที่เพิ่มขึ้น ซึ่งแสดงว่าคุณภาพของสัญญาณจะขึ้นอยู่กับการปรับ Threshold เนื่องจากการปรับ

Threshold ทำให้สัมประสิทธิ์ของข้อมูลลดทอนลงไป ซึ่งทำให้สามารถบีบอัดสัญญาณภาพได้ แต่ก็ทำให้คุณภาพของสัญญาณลดลงไปด้วย

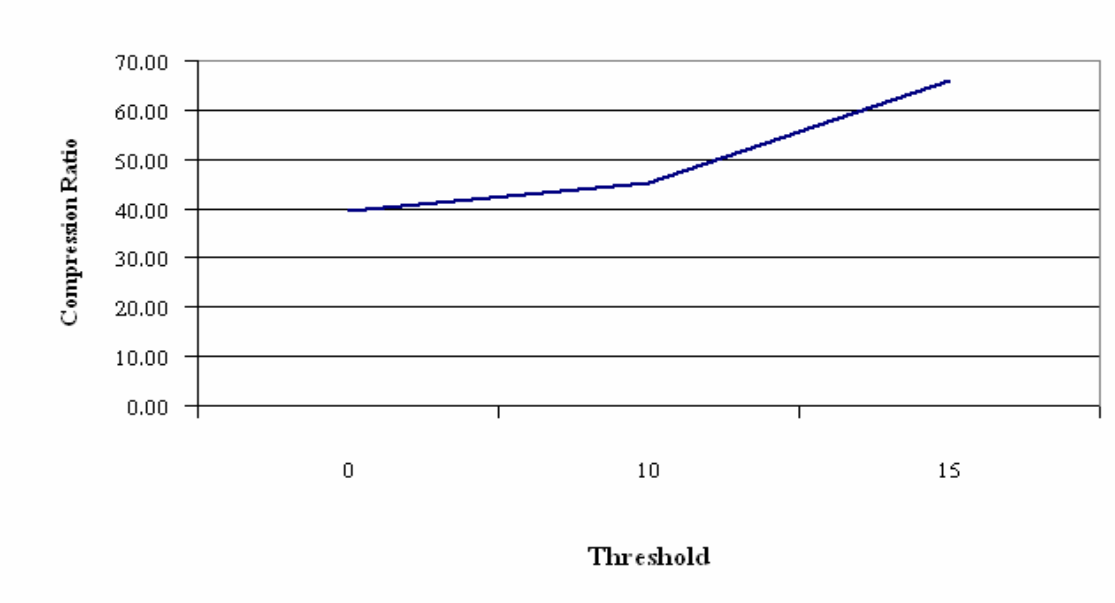
2. ประสิทธิภาพในการบีบอัดข้อมูล (Compression Ratio)

ทำการหาค่า Compression Ratio ตาม Threshold 0, 10, 15 โดยจะได้ค่า Compression Ratio ดังตารางที่ 16

ตารางที่ 16 แสดงค่า Compression Ratio ของการทดลอง

Threshold	สัญญาณภาพจาก Cool.AVI			
	Input (Kbytes)	Output (Kbytes)	Compression Ratio	Compress (%)
0	91	55	1.65	39.56
10	91	50	1.82	45.05
15	91	31	2.94	65.93

ตารางที่ 16 แสดงประสิทธิภาพของการบีบอัด และเปอร์เซ็นต์ของการบีบอัดที่ลดลง ซึ่งลดหลั่นกันไปตามการปรับค่า Threshold โดยจากตารางยิ่งปรับค่า Threshold มากขึ้นข้อมูลก็จะถูกบีบอัดได้มากขึ้น



ภาพที่ 59 แสดงข้อมูลและกราฟของค่า Compression Ratio

ภาพที่ 59 แสดงกราฟเปอร์เซ็นต์การลดลงของการบีบอัดซึ่งจะแปรผันตามการปรับค่า Threshold

3. เวลาที่ใช้ในการบีบอัดเปรียบเทียบในส่วนของ FPGA และ PC

โดยทำการหาค่าเวลาที่ใช้ในการบีบอัดโดยนำ File .AVI จำนวน 5 File มาจับเวลาเปรียบเทียบเวลาที่ใช้ในการทำงานบีบอัดสัญญาณภาพดังแสดงในตารางที่ 17

ตารางที่ 17 เวลาที่ใช้ในการบีบอัดสัญญาณภาพ

ชื่อFile	ขนาดของข้อมูล (Kbytes)	เวลาที่ใช้ในการบีบอัดของ FPGA	เวลาที่ใช้ในการบีบอัดของ PC
FindFile	6.4	0.365 msec	119 mSec
FindCom	51.84	92 mSec	340 mSec
Cool	91	172 mSec	600 mSec
Search	348	658 mSec	871 msec
Counter	491.52	916 mSec	2 Sec 983 mSec

จากผลการทดลองในตารางที่ 17 ในการบีบอัดสัญญาณภาพ FPGA จะทำงานได้เร็วกว่า PC

และการวิจัยครั้งนี้สามารถบีบอัดสัญญาณภาพ โดยใช้ทรัพยากรภายในชิพ FPGA ดังแสดงในตารางที่ 18

ตารางที่ 18 ใช้ทรัพยากรภายในชิพ FPGA

Logic Utilization	Used	Available	Utilization
Number of Slices	780	1920	40%
Number of Slice Flip Flops	419	3840	10%
Number of 4 input LUTs	1447	3840	37%
Number of bonded IOBs	31	97	31%
Number of BRAMs	3	12	25%
Number of GCLKs	5	8	62%

จากผลที่ได้ทำให้การปรับ Threshold เป็นการปรับลดค่าสัมประสิทธิ์เพื่อลดจำนวนบิตของข้อมูลลดลง ทำให้เกิดการสูญเสียข้อมูลบางส่วนไป โดยจากการทดลองได้ทดสอบการตั้งค่า Threshold ที่ค่าต่าง ๆ โดยพบว่าค่า Threshold ที่ต่ำ ค่าสัมประสิทธิ์ที่ลดลงจะน้อย โดยเฉพาะค่า Threshold เท่ากับศูนย์ จะไม่ลดค่าสัมประสิทธิ์เลย ข้อมูลก็จะไม่สูญเสีย และเมื่อปรับค่า Threshold ขึ้นไปจะทำให้สัมประสิทธิ์ข้อมูลโดนตัดทอนให้เป็นศูนย์เพิ่มขึ้น ข้อมูลบางส่วนจะสูญเสียไป

ผลที่ได้จากค่า PSNR คุณภาพของสัญญาณอยู่ในระดับดี มีค่าเฉลี่ยที่ 105 ซึ่งค่าของ PSNR นั้นยิ่งมากยิ่งดี จากเปรียบเทียบ PSNR ในแต่ละเฟรมมีค่าใกล้เคียงกัน และในการมองภาพต้นฉบับกับภาพที่ทำการบีบอัดปรากฏมีลักษณะใกล้เคียงกันมาก ส่วน Compression Ratio ก็สามารถบีบอัดข้อมูลได้ในระดับที่ดีโดยบีบอัดได้ในอัตราส่วนถึง 65 %

จากการตรวจสอบเวลาที่ใช้ในการทำงานของชิพ FPGA ในการทดลองครั้งนี้ได้ใช้เวลาในการบีบอัดสัญญาณภาพเท่ากับ 172 ms ซึ่งเป็นการทำงานที่รวดเร็วมีประสิทธิภาพ แต่จะมีการล่าช้าของการส่งข้อมูลระหว่าง PC กับ FPGA ซึ่งเป็นแบบอนุกรม

สรุป

งานวิจัยนี้นำเสนอแนวทางของการออกแบบการลดขนาดของสัญญาณภาพเพื่อให้ขนาดของข้อมูลน้อยลงโดยใช้ Algorithm Haar Wavelet และการเข้ารหัสตาราง Huffman ที่ใช้ในการเข้ารหัสภาพวีดีโอได้ถูกกำหนดไว้แล้วทำให้ไม่จำเป็นต้องใช้ Algorithm ที่ซับซ้อนในการเข้ารหัสเพียงแต่กำหนดค่าตาราง Huffman ทั้งหมดไว้ในหน่วยความจำของคอมพิวเตอร์เพื่อความรวดเร็วในการประมวลผล

โดย FPGA จะทำในส่วนของการ Compression ซึ่งการออกแบบจะใช้ภาษา VHDL ในการอธิบายลักษณะพฤติกรรมของวงจรที่ต้องการ จากนั้นจึงทำการดาวน์โหลดข้อมูลทางลอจิกลงในชิพ FPGA แล้วนำไปทดสอบการทำงาน ชิพ FPGA ที่ได้ทำการออกแบบมา โดยสามารถทำงานได้อย่างถูกต้องและมีประสิทธิภาพจะเห็นว่าชิพ FPGA เหมาะสำหรับการออกแบบระบบต้นแบบ เพราะFPGAสามารถกำหนดฟังค์ชันการทำงานตามความต้องการของผู้ออกแบบได้ อีกทั้งยังทำได้ง่ายสะดวกและรวดเร็ว ทำให้การออกแบบระบบต้นแบบของวงจรทางดิจิตอลทำได้รวดเร็วยิ่งขึ้น

ส่วนในด้าน Input Output จะใช้โปรแกรม Delphi เพื่อติดต่อระหว่าง PC กับ FPGA รวมทั้งผู้วิจัยได้พัฒนาโปรแกรม Delphi เพื่อใช้ทดสอบในการทำวิทยานิพนธ์ในครั้งนี้ด้วย

และจากการทำงานวิจัยในครั้งนี้สามารถบีบอัดสัญญาณภาพได้ระดับหนึ่งโดยผลที่ได้จากการทดลองซึ่งได้นำ File VDO ขนาด 91 Kbytes มาทำการทดสอบที่ค่า Threshold 15 นั้น ปรากฏว่าสามารถลดขนาดสัญญาณภาพลงเหลือขนาด 31 Kbyte คิดเป็น Compression Ratio ได้ 65.93 % และวัดเวลาในการทำงานของ FPGA ใช้เวลาเพียง 172 ms ซึ่งเป็นเวลาที่น้อยเมื่อเปรียบเทียบกับโปรแกรมใน PC ซึ่งใช้เวลา 0.6 วินาที

ผลจากการวิจัย ผู้ใช้สามารถปรับเปลี่ยนอัตราส่วนของการบีบอัดข้อมูลได้ สามารถนำไปใช้งานกับภาพดิจิตอลต่าง ๆ ได้ และมีขั้นตอนการคำนวณที่ไม่ยุ่งยากซับซ้อนเพื่อนำไปใช้งานด้านต่าง ๆ ได้อย่างกว้างขวาง

ข้อเสนอแนะ

1.เนื่องจากการวิจัยครั้งนี้เป็นการส่งข้อมูลแบบอนุกรมผ่านเครื่องคอมพิวเตอร์ส่วนบุคคล (PC) ซึ่งเกิดความล่าช้าในการปรับปรุงอาจเปลี่ยนการส่งข้อมูลเป็นแบบขนาน เพื่อที่จะทำให้การส่งข้อมูลทำได้รวดเร็วยิ่งขึ้น

2.พัฒนาให้เป็นระบบที่สามารถทำงานได้ตามเวลาจริงเพื่อนำไปใช้ในงานต่างๆ เช่นใน IP Camera ในการส่งข้อมูล Real Time ผ่าน อินเตอร์เน็ต

3.ทำการออกแบบให้สามารถใช้กับ File VDO ได้หลายชนิด เนื่องจากปัจจุบันมี File VDO หลากหลายจึงควรทำให้โปรแกรมสามารถใช้งานกับFile VDO อื่นๆ ด้วย

4.พัฒนาในส่วน PC ให้รวมอยู่ในตัว FPGA เพื่อความสะดวกในการนำไปใช้งาน ซึ่งถ้ารวมทุกส่วนไว้ใน FPGA ก็สามาถนำไปใช้งานต่อเข้ากับ Input ของตัว Source VDO ได้เลย อีกทั้งสามารถใช้งาน FPGA อย่างเต็มประสิทธิภาพเนื่องจาก FPGA มีความรวดเร็วในการทำงานสามารถทำงานในลักษณะ Real Time ได้

เอกสารและสิ่งอ้างอิง

Peggy,Morton.and Arne.Peterson. 1997. **Image Compression Using the Haar Wavelet Transform**

Sàenz,M.P. Sallama,K. Shen and E J.Delp. 1999. **An evaluation of color embedded wavelet image compression techniques**. Proceedings of the SPIE/IS&T Conference on Visual Communications and Image Processing San Jose California. 25p

Raymond Westwater and Borko Furht. 1997. **Real-time video compression: Techniques and Algorithms**. Kluwer Academic Publishers

John F. Wakerly. 2000. DIGITAL DESIGN: Principles and Practices, Prentice-Hall,Inc.

John G. Proakis. 2001. **Digital Communications**. McGraw-Hill

Ken Coftman President, Bytech Services. 2000. **Real World FPGA Design with Verily**. Prentice Hall PTR.

Loonis Pitas Avistotle University to Thessaloniki. 1995. **DIGITAL IMAGE PROCESSING ALGORITHMS**. Pretice Hall. Europe.

L.Prasad & S.SIyengar. 1997. **WEVELET ANALYSIS with Applications to IMAGE PROCESSING**. CRC Press LLC.

Rafacl C. Gonzalez, Richard E.Woods. 2002**. Digital Image Processing 2nd**. Prentice Hall, Inc.

Howward E.Burdick. 1997. **Digital Imagining Theory and Applications**. McGraw-Hill.

Stefan Sjoholm and Lennart Lindh. 1997. **VHDL for Designers**. Prentice Hall. Europe.

Sudhakar Yalamomchili. 2001. **Introductiory VHDL: From Simulation to Synthesis**. Georgin Institue of Technology Prentice Hall.

Amdrew Rishtor trans EDA Limited. **VHDL for logic Synthesis 2nd**. John Wiley F Sons Ltd. England.

Phillip A. Laplante and Alexander D. Stoyenko. **REAL-TIME IMAGING**. United States of America

ชัยมงคล ปราชญ์ชลชาญ. **กรณีศึกษาเรื่องการลดขนาดสัญญาณภาพ Video ด้วย Wavelet**. วิทยานิพนธ์ปริญญาโท. มหาวิทยาลัยเกษตรศาสตร์ กรุงเทพฯ. ปีการศึกษา 2544.

บุญอนันต์ เกียงเอีย. **การออกแบบเครื่องจับสัญญาณภาพทางการแพทย์**. วิทยานิพนธ์ปริญญาโท. สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง. กรุงเทพฯ. ปีการศึกษา 2545.

ประวัติการศึกษา และการทำงาน

ชื่อ –นามสกุล	นายเฉลิมพล สงพราหมณ์
วัน เดือน ปี ที่เกิด	วันที่ 19 เมษายน 2513
สถานที่เกิด	นครศรีธรรมราช
ประวัติการศึกษา	วิศวกรรมศาสตรบัณฑิต สาขา วิศวกรรมโทรคมนาคม มหาวิทยาลัยเทคโนโลยีมหานคร
ตำแหน่งหน้าที่การงานปัจจุบัน	วิศวกร
สถานที่ทำงานปัจจุบัน	บมจ.กสท.โทรคมนาคม