

การออกแบบตัวควบคุมสัญญาณ ดีเอ็มเอ็กซ์ 512
โดยใช้เฟลป์ซีแอนด์ไฮเคอร์เนล
A DESIGN OF EMBEDDED DMX512 CONTROLLER
USING FPGA AND XILKERNEL

सरत योवत्तनर
SARAT YOOWATTANA

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการทอหาตามหลักฐตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์
คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2553

KMITL 2010-EN-M-040-020

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

การออกแบบตัวควบคุมสัญญาณ ดีเอ็มเอ็กซ์ 512

โดยใช้เอฟพีจีเอและไซเคอร์เนล

A DESIGN OF EMBEDDED DMX512 CONTROLLER
USING FPGA AND XILKERNEL



T110454

สารัตถ์ อยู่วัฒนะ

SARAT YOOWATTANA

เลขหมู่.....110454
เลขทะเบียน.....-2 ๒๒๐, 2553
วัน,เดือน,ปี.....

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2553

KMITL 2010-EN-M-040-020

**A DESIGN OF EMBEDDED DMX512 CONTROLLER
USING FPGA AND XILKERNEL**

SARAT YOOWATTANA

**A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENT FOR THE DEGREE OF
MASTER OF ENGINEERING IN ELECTRONICS ENGINEERING
FACULTY OF ENGINEERING
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
2010
KMITL 2010-EN-M-040-020**

COPYRIGHT 20010

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ใบรับรองวิทยานิพนธ์

หัวข้อวิทยานิพนธ์ การออกแบบตัวควบคุมสัญญาณ ดีเอ็มเอ็กซ์ 512 โดยใช้เอฟพีจีเอ และไซเคอร์เนล

Thesis Title A Design of Embedded DMX512 Controller Using FPGA and XILKernel

นักศึกษา นายสารัตต์ อยู่วัฒนะ

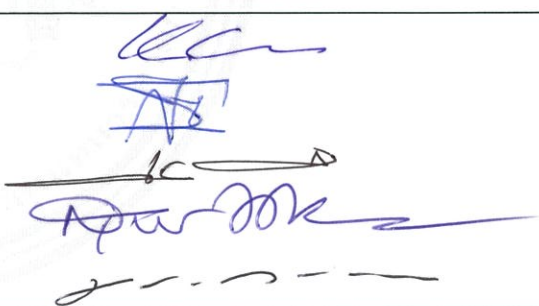
รหัสประจำตัว 50060402

ปริญญา วิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชา วิศวกรรมอิเล็กทรอนิกส์

อาจารย์ที่ปรึกษาวิทยานิพนธ์ รศ.ดร.มนัส สังวรศิลป์

หมายเลขวิทยานิพนธ์ KMITL-2010-EN-M-040-020

คณะกรรมการสอบวิทยานิพนธ์		ลายมือชื่อ
ผศ.ดร.กิตติพล	จิตสกุล	
รศ.ดร.สุรพันธ์	เอื้อไพฑูลย์	
รศ.ดร.สุวิภา	สมควรพาณิชย์	
รศ.ดร.สุรนนท์	น้อยมณี	
รศ.ดร.มนัส	สังวรศิลป์	

วัน / เดือน / ปี ที่สอบ วันจันทร์ที่ 8 มีนาคม พ.ศ. 2553 เวลา 13.30-15.30 น.

สถานที่สอบ ณ อาคาร A ชั้น 3 ห้องประชุม 2

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

KING MONKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

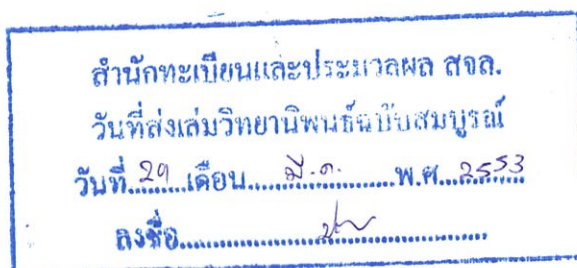
คณะวิศวกรรมศาสตร์ รับรองแล้ว



(รองศาสตราจารย์ ดร.กอบชัย เดชหาญ)

คณบดี คณะวิศวกรรมศาสตร์

วันที่ 8 มีนาคม พ.ศ. 2553



หัวข้อวิทยานิพนธ์	การออกแบบตัวควบคุมสัญญาณ ดีเอ็มเอ็กซ์ 512 โดยใช้เอฟพีจีเอ และไซเคอร์เนล
นักศึกษา	นายสารัตต์ อยู่วัฒนะ
รหัสนักศึกษา	50060402
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมอิเล็กทรอนิกส์
พ.ศ.	2553
อาจารย์ที่ปรึกษาวิทยานิพนธ์	รศ.ดร.มนัส สัจวรศิลป์

บทคัดย่อ

วิทยานิพนธ์ฉบับนี้เสนอวิธีการสร้างตัวกำเนิดสัญญาณดีเอ็มเอ็กซ์โดยการใช้เอฟพีจีเอ ร่วมกับไซเคอร์เนล ซึ่งสามารถสร้างตัวกำเนิดสัญญาณดีเอ็มเอ็กซ์หลายตัวไว้ในไอซีเพียงตัวเดียว และสามารถใช้ไมโครคอนโทรลเลอร์เพียงตัวเดียวในการควบคุมการทำงานให้สามารถทำงานได้ทันเวลากัน โดยได้ออกแบบวงจรรวมดิจิทัลในเอฟพีจีเอด้วยภาษาวีเอชดีแอล และใช้งานร่วมกับเอ็มเบดเด็ดโปรเซสเซอร์ ซึ่งเอ็มเบดเด็ดโปรเซสเซอร์นั้นได้ติดตั้งซอฟต์แวร์ปฏิบัติการไซเคอร์เนล เพื่อให้สามารถทำงานแบบหลายเทรคพร้อมกันได้ ซึ่งทำให้แก้ปัญหาในการเพิ่มช่องสัญญาณโดยใช้ไมโครคอนโทรลเลอร์หลาย ๆ ตัวมาต่อกัน และยังช่วยเพิ่มประสิทธิภาพของเฟรมเรท ที่สามารถรองรับการแสดงผลเป็นภาพเคลื่อนไหวได้

Thesis Title	A Design of Embedded DMX512 Controller Using FPGA and XILKernel
Student	Mr. Sarat Yoowattana
Student ID.	50060402
Degree	Master of Engineering
Program	Electronics Engineering
Year	2010
Thesis Advisor	Assoc. Prof. Dr. Manas Sangworasil

ABSTRACT

This thesis offers a way to create DMX512 signal generator using FPGA with XILkernel. This can create multiple DMX512 signal generator in a single IC and can be used only one Microcontroller to control that allows all channels to work together simultaneously. The digital integrated circuit designs in FPGA using VHDL language and used with embedded processors. The embedded processor is installed software kernel to allow multiple thread work together simultaneous. This technique can replace multiple microcontrollers connected technique and can improve the performance of frame rate and support a video display.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จได้อย่างดีด้วยคำแนะนำและคำปรึกษาจาก รศ.ดร.มนัส สังวรศิลป์ ซึ่งเป็นอาจารย์ผู้ควบคุมวิทยานิพนธ์และ อาจารย์ชินภัทร นันทจิวงกรชัย ข้าพเจ้ารู้สึกทราบซึ่งในความอนุเคราะห์จากท่านอาจารย์ทั้งสองท่าน และขอขอบพระคุณเป็นอย่างสูง

ขอกราบพระคุณคณาจารย์ภาควิชาวิศวกรรมอิเล็กทรอนิกส์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุก ๆ ท่านที่ได้ประสิทธิ์ประสาทวิชาให้กับข้าพเจ้า

ขอขอบคุณบัณฑิตศึกษาและบัณฑิตวิทยาลัย คณะวิศวกรรมศาสตร์ที่ให้ความช่วยเหลือในเรื่องต่างๆ

สุดท้ายนี้ข้าพเจ้าขอกราบขอบพระคุณ บิดา มารดา และครอบครัวของข้าพเจ้าที่เป็นกำลังใจ และให้การสนับสนุนในทุกเรื่องๆ ทำให้ข้าพเจ้าสามารถทำวิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงด้วยดี

คุณค่าและประโยชน์อันพึงมาจากวิทยานิพนธ์ฉบับนี้ ข้าพเจ้าขอบแต่ผู้มีพระคุณทุกท่าน

สารัตถ์ อยู่วัฒนะ

สารบัญ

หน้า

บทคัดย่อภาษาไทย.....	I
บทคัดย่อภาษาอังกฤษ.....	II
กิตติกรรมประกาศ.....	..III
สารบัญ.....	..IV
สารบัญตาราง.....	.VIII
สารบัญรูป.....	..IX
บทที่ 1 บทนำ	1
1.1ความเป็นมาและความสำคัญของปัญหา	1
1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา	1
1.3 สมมติฐานของการศึกษา.....	2
1.4 ทฤษฎีหรือแนวความคิดที่ใช้ในการวิจัย	2
1.5 ขอบเขตการวิจัย.....	2
1.6 ขั้นตอนของการศึกษา.....	3
บทที่ 2 มาตรฐานโปรโตคอล DMX512 และงานวิจัยที่เกี่ยวข้อง	4
2.1 มาตรฐานของโปรโตคอล DMX512.....	4
2.1.1 ประวัติความเป็นมา.....	4
2.1.2 ดีเอ็มเอ็กซ์ 512 (DMX512)	4
2.1.3 จังหวะสัญญาณ DMX512 (Timing diagram)	6
2.2 อัตราของการแสดงผลสูงสุดใน 1 วินาที (Frame Rate)	8
2.3 งานวิจัยที่เกี่ยวข้อง.....	8
2.3.1 การออกแบบแอลอีดีเมทริกซ์ (LED Matrix) ด้วยเอฟพีจีเอ โดยใช้โปรโตคอลดีเอ็มเอ็กซ์ 512.....	9
2.3.2 การวิจัยเรื่องระบบ LED Display ที่รวมโปรโตคอล CAN (Controller Area Network) Bus และ DMX512 ไว้ด้วยกัน [4]	9
2.3.3 การวิจัยระบบส่องสว่างของสถานที่โดยใช้โปรโตคอล DMX512 [5]	9
บทที่ 3 สถาปัตยกรรมและการออกแบบบน FPGA	11

สารบัญ (ต่อ)

หน้า

3.1 บทนำ.....	11
3.2 เอฟพีจีเอ (FPGA: Field Programmable Gate Array)	12
3.2.1 เปรียบเทียบกับไมโครคอนโทรลเลอร์ (Microcontroller).....	12
3.2.2 จุดเด่นของ FPGA	13
3.2.3 FPGA ในท้องตลาดและการเลือกใช้.....	13
3.3 รายละเอียดสถาปัตยกรรมของ FPGA ตระกูล VIRTEX II Pro	13
3.3.1 บล็อกอินพุต/เอาต์พุต (IOB).....	14
3.3.2 ลอจิกเซลล์ (Logic Cells)	15
3.3.3 บล็อกแรม (Block RAM).....	16
3.3.4 ดีเลย์ล๊อคลูป (Delay-Locked Loop).....	17
3.3.5 ข้อมูลเกี่ยวกับหมายเลข FPGA	17
3.4 การออกแบบวงจรดิจิทัลในปัจจุบัน	18
3.4.1 การออกแบบโดยใช้การวาดผังวงจร (Schematic design)	18
3.4.2 การออกแบบโดยใช้ภาษาระดับสูง (HDL).....	19
3.5 การเขียนภาษา HDL	20
3.5.1 ลักษณะเฉพาะในการออกแบบ (Design specifications)	21
3.5.2 การจำลองการทำงานฟังก์ชัน (Functional simulation)	22
3.5.3 การสังเคราะห์วงจร (Synthesis).....	22
3.5.4 การจำลองการทำงานในระดับพฤติกรรม Timing simulation.....	23
3.5.5 การวางและเชื่อมเส้นทาง (Place & Route)	23
3.6 การออกแบบจากบนลงล่างและการออกแบบจากล่างขึ้นบน	23
3.7 RTL (Register Transfer Level)	24
3.9 Kernel และ Xilkernel.....	25
3.9.1 XILKernel.....	25
3.9.2 การจัดการของ Xilkernel	25

สารบัญ (ต่อ)

หน้า

3.9.3 การสร้าง Xilkernel	26
3.9.4 Xilkernel API.....	26
3.10 บทสรุป.....	28
บทที่ 4 การออกแบบตัวควบคุมดีเอ็มเอ็กซ์ 512 โดยใช้เอฟพีจีเอ.....	29
4.1 การออกแบบตัวสร้างสัญญาณ DMX (DMX Creator)	30
4.1.1 การทำงานของ DMX Creator	32
4.1.2 เอ็มเบ็ดเด็ด เคเวลอปเมนต์ คิท	33
4.1.2 การออกแบบคอร์สำหรับชุดต่อพ่วง (Peripheral DMX Core)	34
4.2 การทำงานของ DMX Controller	35
4.3 โฟลว์ชาร์ตของโปรแกรม.....	37
4.3.1 การทำงานแบบทำงานด้วยตนเอง	37
4.3.2 การทำงานแบบรอรับคำสั่งจากคอมพิวเตอร์.....	38
4.4 รูปแบบสัญญาณเอาต์พุตมาตรฐาน	39
4.5 รูปแบบของตัวกำเนิดสัญญาณดีเอ็มเอ็กซ์.....	40
4.5.1 ตัวกำเนิดสัญญาณดีเอ็มเอ็กซ์โดยใช้ไมโครคอนโทรลเลอร์.....	40
4.5.2 ตัวกำเนิดสัญญาณดีเอ็มเอ็กซ์โดยใช้เอฟพีจีเอ	40
4.6 บล็อกไดอะแกรมของฮาร์ดแวร์.....	41
บทที่ 5 การทดลองและผลการทดลอง	43
5.1 รายละเอียดของเครื่องมือและอุปกรณ์ที่ใช้ในการทดลอง	43
5.1.1 บอร์ดทดลอง FPGA XUPV2P	44
5.1.2 อุปกรณ์รับสัญญาณ DMX512 รุ่น DEC1P-3S.....	44
5.1.3 เครื่องสร้างสัญญาณดีเอ็มเอ็กซ์ รุ่น DMX 192 II	45
5.2 โหมดการทำงาน.....	46
5.2.1 การทดลองในโหมดทำงานด้วยตัวเอง (Standalone).....	46
5.2.2 การทำงานในโหมดทางผ่าน (Bypass).....	46

สารบัญ (ต่อ)

หน้า

5.3 การทดลองโดยใช้เครื่องสร้างสัญญาณคี่เอ็มเอ็กซ์รุ่น DMX 192 II.....	47
5.4 การทดลองโดยใช้เอฟพีจีเอ สร้างวงจรดิจิทัลเป็นตัวกำเนิดสัญญาณคี่เอ็มเอ็กซ์ที่หลาย ช่องสัญญาณ	49
5.4.1 ทำการทดลองวัดสัญญาณเอาต์พุตของเพื่อตรวจสอบ Timing.....	49
5.4.2 ทำการวัดเพื่อเปรียบเทียบค่าหน่วงเวลาของช่องสัญญาณ 0 และ 3	50
5.4.2 เฟรมเรตสูงสุดที่รองรับ	52
บทที่ 6 สรุปผลการวิจัย และข้อเสนอแนะ.....	55
บรรณานุกรม.....	56

สารบัญตาราง

ตารางที่	หน้า
2.1 คำอธิบายคำสั่งต่าง ๆ ของโปรโตคอลดีเอ็มเอ็กซ์	7
2.2 ระยะเวลาที่ใช้ต่ำสุดในแต่ละคำสั่งของโปรโตคอลดีเอ็มเอ็กซ์	8
3.1 ตารางเปรียบเทียบคุณสมบัติ FPGA ตระกูล VIRTEX II Pro series	17
4.1 ทฤษฎีการและค่ากำลังงานที่ใช้ใน 1 วงจรของตัวสร้างสัญญาณดีเอ็มเอ็กซ์	32
4.2 หน้าที่ของรีจิสเตอร์แต่ละตัว.....	34
4.3 ลำดับข้อมูลที่จะถูกเขียนลงบัฟเฟอร์.....	35
4.4 ค่าระยะเวลามาตรฐานของโปรโตคอลดีเอ็มเอ็กซ์.....	39

สารบัญรูป

รูปที่	หน้า
รูปที่ 2.1 ตัวอย่างการเชื่อมต่ออุปกรณ์ดีเอ็มเอ็กซ์ บนบัส	5
รูปที่ 2.2 ตัวอย่างอุปกรณ์ที่ใช้ Address ของ DMX มากกว่า 1 Address ในตัวเดียว	6
รูปที่ 2.3 จังหวะมาตรฐานของสัญญาณดีเอ็มเอ็กซ์.....	6
รูปที่ 3.1 การแบ่งกลุ่มของวงจรรวม ASIC	11
รูปที่ 3.2 บล็อกไดอะแกรมของโครงสร้าง FPGA ตระกูล VIRTEX II Pro	14
รูปที่ 3.3 บล็อกอินพุต/เอาต์พุต (IOB) ของโครงสร้าง FPGA ตระกูล VIRTEX II Pro.....	15
รูปที่ 3.4 บล็อก CLB ของ FPGA ตระกูล VIRTEX II Pro (สมมาตรในแต่ละ CLB)	16
รูปที่ 3.5 บล็อกแรมพอร์ตคู่.....	16
รูปที่ 3.6 การออกแบบโดยการวาดผังวงจร	18
รูปที่ 3.7 การออกแบบโดยการเขียนเป็นภาษาวีเอชดีแอล	19
รูปที่ 3.8 ขั้นตอนการออกแบบวงจรด้วยภาษา HDL	21
รูปที่ 3.9 การจำลองการทำงานด้วยโปรแกรม ไอเอสอี ซิมูเลเตอร์.....	22
รูปที่ 3.10 กระบวนการออกแบบ (Design Methodology).....	23
รูปที่ 3.11 ตัวอย่าง Register Transfer Level บล็อกไดอะแกรม	24
รูปที่ 3.12 โมดูลของไซเคอร์เนล (XILKernel).....	26
รูปที่ 3.13 ขั้นตอนการสร้างไซเคอร์เนล (XILKernel)	26
รูปที่ 4.1 การเชื่อมต่อในการทำงานแบบทำงานด้วยตัวเอง	29
รูปที่ 4.2 การเชื่อมต่อในการทำงานแบบรับข้อมูลผ่านคอมพิวเตอร์.....	30
รูปที่ 4.3 บล็อกไดอะแกรมของตัวสร้างสัญญาณดีเอ็มเอ็กซ์	30
รูปที่ 4.4 บล็อกของตัวสร้างสัญญาณดีเอ็มเอ็กซ์ (DMX Creator)	31
รูปที่ 4.5 สเตปเมชชีนของตัวสร้างสัญญาณดีเอ็มเอ็กซ์ (DMX Creator).....	31
รูปที่ 4.6 ลักษณะโครงสร้างของโปรเซสเซอร์และอุปกรณ์ต่อพ่วงที่สร้างด้วยโปรแกรม EDK ..	33
รูปที่ 4.7 การตั้งค่ารีจิสเตอร์ในการใช้แม่แบบของ Xilinx.....	34
รูปที่ 4.8 บล็อกของวงจรดิจิทัลที่จะนำไปเชื่อมต่อกับพีแอลบีบัส (PLB BUS).....	35

สารบัญรูป (ต่อ)

รูปที่	หน้า
รูปที่ 4.9 ไบท์ข้อมูลที่ถูกเขียนลงหน่วยความจำ.....	36
รูปที่ 4.10 โฟลว์ชาร์ตการทำงานแบบทำงานด้วยตัวเอง	37
รูปที่ 4.11 รูปแบบการกำหนดค่าในไฟล์ของการทำงานแบบทำงานด้วยตัวเอง.....	37
รูปที่ 4.12 โฟลว์ชาร์ตการทำงานแบบรอรับข้อมูลจากคอมพิวเตอร์	38
รูปที่ 4.13 รูปแบบข้อมูลที่ส่งออกจากคอมพิวเตอร์	39
รูปที่ 4.14 รูปแบบมาตรฐานของสัญญาณดีเอ็มเอ็กซ์	39
รูปที่ 4.15 การเชื่อมต่อไมโครคอนโทรลเลอร์กับบัสดีเอ็มเอ็กซ์ 1 ช่องสัญญาณ	40
รูปที่ 4.16 การเชื่อมต่อไมโครคอนโทรลเลอร์หลายตัวกับบัสดีเอ็มเอ็กซ์แบบหลายช่องสัญญาณ.....	40
รูปที่ 4.17 การเชื่อมต่อไมโครคอนโทรลเลอร์ร่วมกับเอฟพีจีเอแบบหลายช่องสัญญาณ	41
รูปที่ 4.18 บล็อกไดอะแกรมของฮาร์ดแวร์ที่ใช้ในงานวิจัย	41
รูปที่ 5.1 บอร์ดเอฟพีจีเอที่ใช้ในการพัฒนา	43
รูปที่ 5.2 บล็อกไดอะแกรมของบอร์ดทดลองที่ใช้ในงานวิจัย.....	44
รูปที่ 5.3 อุปกรณ์รับสัญญาณดีเอ็มเอ็กซ์ รุ่น DEC1P-3S.....	45
รูปที่ 5.4 บล็อกไดอะแกรมของอุปกรณ์รับสัญญาณดีเอ็มเอ็กซ์ รุ่น DEC1P-3S.....	45
รูปที่ 5.5 เครื่องสร้างสัญญาณดีเอ็มเอ็กซ์ รุ่น DMX 192 II.....	45
รูปที่ 5.6 การเชื่อมต่อในการทำงานแบบทำงานด้วยตัวเอง	46
รูปที่ 5.7 การเชื่อมต่อในโหมดทางผ่าน.....	46
รูปที่ 5.8 โปรแกรมควบคุมดีเอ็มเอ็กซ์ที่ใช้ในการทดลอง.....	47
รูปที่ 5.9 การเชื่อมต่อระหว่าง DMX 192 II กับอุปกรณ์ดีเอ็มเอ็กซ์ 512.....	47
รูปที่ 5.10 สัญญาณดีเอ็มเอ็กซ์ที่ออกจาก ไมโครคอนโทรลเลอร์	48
รูปที่ 5.11 สัญญาณดีเอ็มเอ็กซ์ที่ออกจากไมโครคอนโทรลเลอร์ ใน 1 เฟรม.....	48
รูปที่ 5.12 สัญญาณ Timing ที่ออกจาก FPGA ผ่านการแปลงให้เป็นอาร์เอส 485.....	49
รูปที่ 5.13 สัญญาณเอาต์พุตของช่องสัญญาณ 0 และ 3	52
รูปที่ 5.14 ระยะเวลาที่ใช้ในการแสดงผลของการส่งข้อมูล 512 อุปกรณ์.....	54

บทที่ 1

บทนำ

1.1ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบัน การตกแต่งอาคารหรือสถานที่ด้วยแสงไฟต่างๆ เป็นที่นิยมกันอย่างแพร่หลาย โดยมีจุดประสงค์เพื่อความสวยงาม หรือเพื่อทางการค้า และการพัฒนาของแอลอีดี (LED: Light Emitting Diode) ก้าวหน้าอย่างรวดเร็ว ทำให้อุปกรณ์ส่องสว่างต่าง ๆ มีการพัฒนาตามไปด้วย ทั้งด้านของอายุการใช้งาน ความหลากหลายของสี หรือด้านพลังงาน อุปกรณ์ต่าง ๆ เหล่านี้จะมีโปรโตคอล (Protocol) ที่นิยมใช้อย่างแพร่หลายนั่นคือโปรโตคอลดีเอ็มเอ็กซ์ 512 (DMX512) โดยปรกติโปรโตคอล DMX512 จะสามารถรองรับอุปกรณ์ที่ต่อรวมได้สูงสุด 512 อุปกรณ์ ต่อ DMX512 1 ช่องสัญญาณ (Channel) ซึ่งหากต้องการต่ออุปกรณ์มากกว่า 512 อุปกรณ์ จะต้องทำการเพิ่มช่องสัญญาณเพื่อให้สามารถเพิ่มการควบคุมอุปกรณ์ ได้อีก 512 อุปกรณ์

โดยทั่วไปแล้วหากระบบมีขนาดเล็ก สามารถที่จะสร้างตัวควบคุม DMX512 ด้วยการใชไมโครคอนโทรลเลอร์ (Microcontroller) ได้ แต่ถ้าหากระบบมีขนาดใหญ่และมีความต้องการช่องสัญญาณ DMX512 หลายช่องสัญญาณ อาจนำไมโครคอนโทรลเลอร์หลายๆตัวมาต่อกัน หรือว่าอาจนำคอมพิวเตอร์ (Computer) เข้ามาใช้งานแทน

การนำคอมพิวเตอร์เข้ามาใช้เพื่อสร้างเป็น DMX512 Controller นั้นทำให้มีความสะดวกในการใช้งานหลายๆ ด้าน เช่นความเร็ว การควบคุม การสั่งงาน แต่จำเป็นที่ต้องมีอุปกรณ์ต่อเพิ่มเติมเพื่อรับสัญญาณต่างๆจากคอมพิวเตอร์เช่นยูเอสบี (USB) หรือจากพีซีไอ (PCI) แล้วมาแปลงให้อยู่ในรูปแบบโปรโตคอลแบบ DMX512 ซึ่งอุปกรณ์ที่จะรับข้อมูลจากคอมพิวเตอร์และส่งออกไปยังช่องสัญญาณของ DMX512 นั้นนิยมใช้ไมโครคอนโทรลเลอร์ ดังนั้นหากมีช่องสัญญาณ DMX512 หลายช่องสัญญาณก็จะต้องมีไมโครคอนโทรลเลอร์ หลาย ๆ ตัว

การแก้ปัญหานี้คือ ได้ทำการสร้าง DMX512 Controller ลงใน FPGA ซึ่งใน FPGA 1 ตัวสามารถสร้าง DMX Controller ได้หลายตัวขึ้นอยู่กับขนาดของ FPGA ที่ใช้แล้วตัว FPGA นี้สามารถรับข้อมูลต่าง ๆ จากไมโครคอนโทรลเลอร์หรือจากคอมพิวเตอร์ได้

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

วัตถุประสงค์ของการศึกษานี้เพื่อ สร้าง DMX512 Controller หลาย ๆ ตัวให้อยู่ใน FPGA เพียงตัวเดียวและสามารถควบคุม DMX512 Controller โดยการใช้ไมโครคอนโทรลเลอร์หรือคอมพิวเตอร์เพียงตัวเดียวได้ และ DMX512 Controller ทุกตัวสามารถทำงานพร้อมกันได้

1.3 สมมติฐานของการศึกษา

ข้อด้อยของโปรโตคอลดีเอ็มเอช 512 นั้นคือการรองรับการเชื่อมต่ออุปกรณ์ที่ 512 อุปกรณ์เท่านั้น ซึ่งในความเป็นจริงแล้วจุดมุ่งหมายของโปรโตคอลนี้ถูกออกแบบมาเพื่อใช้สำหรับควบคุมอุปกรณ์ต่าง ๆ บนเวทีคอนเสิร์ต แต่ในปัจจุบันมีการนิยมใช้โปรโตคอลนี้ในการควบคุมอุปกรณ์ตกแต่งหลาย ๆ ชนิด ทำให้มีความต้องการในการเพิ่มการควบคุมอุปกรณ์มากขึ้น แต่หากว่าเพิ่มอุปกรณ์ในบัสมากขึ้น จะทำให้เฟรมเวิร์กประสิทธิภาพแย่ง และมีการแสดงผลที่ไม่สวยงาม

การแก้ปัญหาข้างต้นนี้ ได้ใช้วิธีสร้างวงจรดิจิทัล ที่ทำหน้าที่สร้างสัญญาณตามโปรโตคอลดีเอ็มเอช 512 ด้วยเอฟพีจีเอ ซึ่งทำให้สามารถสร้างช่องสัญญาณได้หลายช่องสัญญาณและสามารถรองรับการทำงานที่ความเร็วสูงได้

1.4 ทฤษฎีหรือแนวความคิดที่ใช้ในการวิจัย

โปรโตคอล DMX512 สามารถรองรับการควบคุมอุปกรณ์ได้สูงสุด 512 อุปกรณ์ หากนำไปใช้งานในระบบที่มีขนาดใหญ่ จำเป็นต้องมีอุปกรณ์ควบคุมเพิ่มตามขึ้นมา หรืออาจใช้คอมพิวเตอร์เข้ามาทำงานในส่วนนี้ แต่เอาต์พุตของคอมพิวเตอร์เองก็ต้องถูกแปลงสัญญาณให้อยู่ในรูปแบบโปรโตคอล DMX512 ด้วยวงจรภายนอกอีกครั้งหนึ่ง อีกทั้งคอมพิวเตอร์เองจะต้องประกอบไปด้วยอุปกรณ์อื่นๆอีกหลายๆส่วนเช่น Hard disk, RAM, Switching Power supply การนำ FPGA มาใช้สร้าง DMX512 Core นั้นจะช่วยแก้ปัญหานาฬิกาของฮาร์ดแวร์ต่าง ๆ ที่เพิ่มขึ้น ซึ่งการสร้าง DMX512 หลาย ๆ ตัวใน FPGA นั้นจะยืดหยุ่นต่อการควบคุมซึ่ง การควบคุมสามารถที่จะใช้ไมโครคอนโทรลเลอร์เพียงตัวเดียวควบคุมการทำงานของ DMX512 หลายๆ ช่องสัญญาณได้

1.5 ขอบเขตการวิจัย

วิทยานิพนธ์ฉบับนี้นำเสนอการสร้าง DMX512 Core โดยใช้ภาษา VHDL ซึ่งทำการออกแบบลงใน FPGA ของ Xilinx โดยให้มี DMX512 Core 4 ช่องสัญญาณ และ DMX512 Core จะถูกควบคุมด้วยไมโครคอนโทรลเลอร์เพียง 1 ตัว จากนั้นทดสอบการทำงานร่วมกับอุปกรณ์ DMX512 ชนิดที่เป็นอุปกรณ์ส่องสว่าง และจากนั้นหาค่าหน่วยเวลาของระหว่างช่องสัญญาณโดยทำการวัดสัญญาณเอาต์พุตด้วยออสซิลโลสโคป และหาอัตราของเฟรมภาพที่เล่นใน 1 วินาทีที่ทุกช่องสัญญาณแสดงผลโดยการวัดด้วยออสซิลโลสโคปเช่นเดียวกัน

1.6 ขั้นตอนของการศึกษา

วิทยานิพนธ์ฉบับนี้ ประกอบด้วยรายละเอียดและเนื้อหาต่างๆ แบ่งออกเป็น 6 บท ซึ่งในแต่ละบทมีหัวข้อและเนื้อหาดังต่อไปนี้

บทที่ 1 บทนำ ในบทนี้จะกล่าวถึง ความเป็นมาและความสำคัญของปัญหา ความมุ่งหมาย และวัตถุประสงค์ของการศึกษา ทฤษฎีหรือแนวคิดที่ใช้ในการวิจัย ขอบเขตของการวิจัย และขั้นตอนของการศึกษา

บทที่ 2 มาตรฐานของโปรโตคอล DMX512 และงานวิจัยที่เกี่ยวข้อง ซึ่งเป็นงานวิจัยที่เกี่ยวกับการออกแบบอุปกรณ์ควบคุมและอุปกรณ์แสดงผล

บทที่ 3 กล่าวถึง สถาปัตยกรรมและการออกแบบบน FPGA ซึ่งรวมถึงพื้นฐานของไซเคอร์เนล (XILKernel)

บทที่ 4 กล่าวถึงการออกแบบตัวควบคุมดีเอ็มเอ็กซ์ 512 โดยใช้ FPGA ซึ่งแบ่งการทำงานออกเป็น 2 โหมดการทำงาน คือ โหมดทำงานด้วยตัวเอง (Standalone) และโหมดรับข้อมูลจากคอมพิวเตอร์ (Bypass)

บทที่ 5 การทดลองและผลการทดลอง

บทที่ 6 บทสรุปและข้อเสนอแนะ

บทที่ 2

มาตรฐานโปรโตคอล DMX512 และงานวิจัยที่เกี่ยวข้อง

ในบทนี้จะกล่าวถึงหลักการพื้นฐานของมาตรฐานโปรโตคอล DMX512 ซึ่งจะอธิบายถึงประวัติความเป็นมาและงานวิจัยที่เกี่ยวข้องซึ่งมีผู้ได้วิจัยไว้ก่อนหน้านี้ เพื่อใช้เป็นแนวทางในการพัฒนางานวิจัยต่อไป

2.1 มาตรฐานของโปรโตคอล DMX512

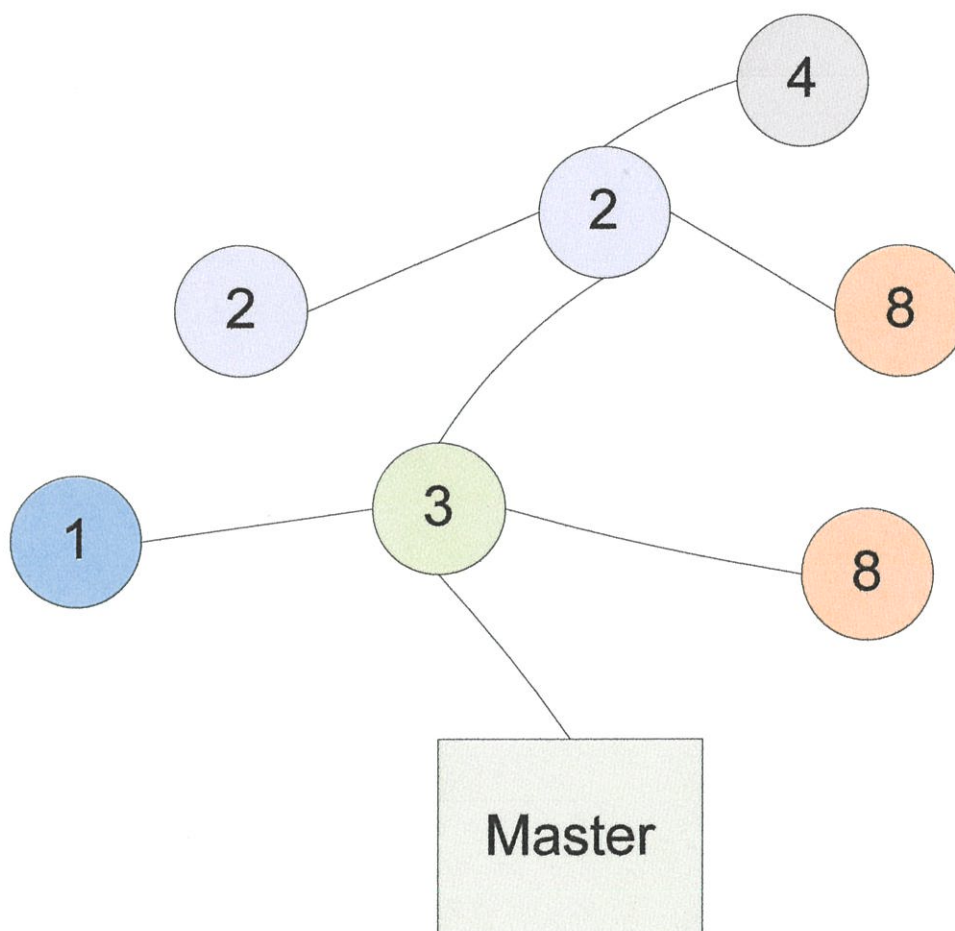
2.1.1 ประวัติความเป็นมา

DMX 512 ได้รับการพัฒนาเป็นครั้งแรกในปี 1986 โดยคณะกรรมการของสถาบันทางเทคโนโลยีของสหรัฐอเมริกา ที่มีชื่อว่า USITT (U.S. Institute of Theater Technology [1]) ซึ่งมีจุดประสงค์ที่จะใช้ให้เป็นมาตรฐานการส่งสัญญาณสำหรับการควบคุมสำหรับ อุปกรณ์หรี่ไฟ (Dimmer) และ อุปกรณ์ควบคุม (Controller) และUSITTได้มีการนำมาปรับปรุงมาตรฐานอีกครั้งในปี 1990 และได้ใช้ชื่อว่า DMX512/1990 ต่อมาในปี 2004 ANSI (American National Standards Institute) ได้อนุมัติและรับรองมาตรฐาน E1.11[2] ที่เป็นโปรโตคอล DMX512-A ซึ่งเป็นมาตรฐานที่ใช้ในการควบคุมอุปกรณ์แสงสว่างและอุปกรณ์ต่าง ๆ ด้วยสัญญาณอนุกรมดิจิทัลแบบ Asynchronous ในวันที่ 8 เดือนพฤษภาคม ปี ค.ศ 2004

2.1.2 ดีเอ็มเอ็กซ์ 512 (DMX512)

DMX512 คือ มาตรฐานโปรโตคอลชนิดหนึ่งที่ใช้สำหรับควบคุมอุปกรณ์ต่าง ๆ โดยอุปกรณ์แต่ละตัวจะมีที่อยู่ (Address) เป็นของตัวเอง มาตรฐานนี้เป็นที่นิยมอย่างมากในการใช้ควบคุมอุปกรณ์ส่องสว่างหรืออุปกรณ์อื่น ๆบนเวทีคอนเสิร์ต DMX512 เป็นตัวย่อมาจากคำภาษาอังกฤษว่า Digital Multiplex ส่วนตัวเลข 512 แสดงจำนวนตำแหน่งที่สามารถควบคุมอุปกรณ์ได้ การส่งสัญญาณควบคุมของมาตรฐาน DMX512 จะส่งสัญญาณแบบดิจิทัลความเร็ว 250 กิโลบิตต่อวินาที (Kbps) บนบัสแบบ RS-485 (TIA/EIA-485-A) ซึ่งสามารถส่งผ่านสัญญาณแบบดิจิทัลจากชุดควบคุมไปยังอุปกรณ์หลายๆตัวซึ่งต่อขนานอยู่บนบัส เป็นระยะทางถึง 4,000 ฟุต หรือประมาณ 1 กิโลเมตร โดยใช้สายสัญญาณเพียงสองเส้นเท่านั้น(ไม่รวมสายกราวด์และสาย power supply) โดยที่ชุดควบคุมจะส่งสัญญาณมาเป็นชุด (Packet) โดยแต่ละชุดจะส่งสัญญาณออกมาสูงสุด 512 ช่องสัญญาณหรือน้อยกว่านั้นขึ้นอยู่กับจำนวนอุปกรณ์ที่ต่ออยู่บนบัส ส่วนตัวอุปกรณ์ปลายทางแต่ละตัวที่ต่ออยู่ในระบบ จะทำหน้าที่รับสัญญาณเฉพาะที่ตรงตำแหน่งที่ตั้งไว้เท่านั้นเพื่อนำมาใช้งาน โดยระบบ DMX512 สามารถที่จะควบคุมการทำงานของอุปกรณ์

ต่างๆ ได้ถึง 512 ตำแหน่งหรือควบคุมอุปกรณ์ได้ 512 function อุปกรณ์ซึ่งใช้งานบนระบบอาจจะ เป็นอุปกรณ์หลายๆอย่างใช้งานร่วมกันเช่น Moving Head, Dimmer, Relay หรืออุปกรณ์อื่นๆที่ รองรับการใช้งานกับระบบ DMX ได้ การเชื่อมต่ออุปกรณ์ DMX สามารถที่จะเชื่อมต่อ อุปกรณ์แบบ star หรืออุปกรณ์หลายๆ ตัวสามารถที่จะมี Address เดียวกันได้



รูปที่ 2.1 ตัวอย่างการเชื่อมต่ออุปกรณ์ดีเอ็มเอ็กซ์ บนบัส

จากรูปที่ 2.1 เป็นตัวอย่างการเชื่อมต่ออุปกรณ์ต่างๆ บนบัส ซึ่งอุปกรณ์ที่ต่ออยู่จะรับ ข้อมูลเดียวกัน แต่อุปกรณ์จะเก็บเฉพาะข้อมูลโดยตรงกับตำแหน่งของตัวเองที่ได้ตั้งไว้เท่านั้น ดังนั้น เมื่อเราส่งข้อมูลชุดที่ 2 และ 8 จะมีอุปกรณ์ทำงานเหมือนกันทั้ง 2 อุปกรณ์

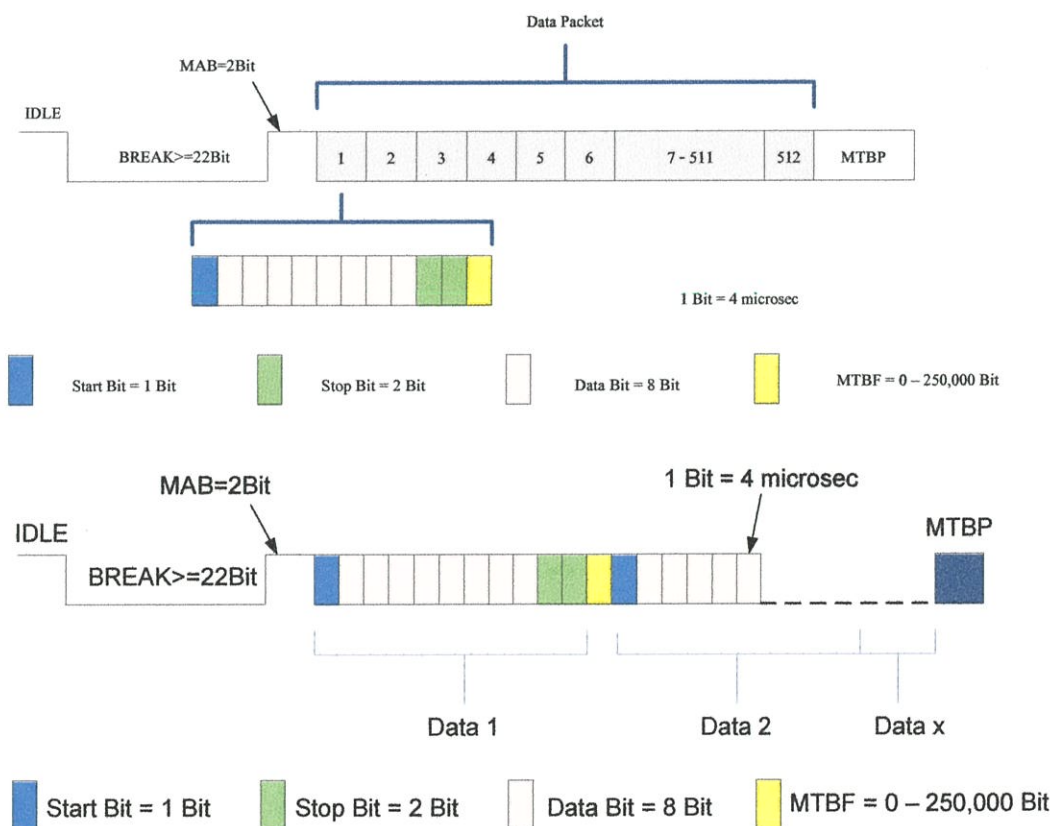
ในอุปกรณ์บางตัวสามารถที่จะมี Address มากกว่า 1 address (รูปที่ 2.2) ได้เพื่อที่จะทำให้ สามารถรองรับรูปแบบต่าง ๆ ของอุปกรณ์ ตัวอย่างเช่นอุปกรณ์ชนิดสปอตไลท์ซึ่งสามารถปรับ ความสว่างของแสงที่แสดงได้ 256 ระดับ ก็จะใช้ Address เพียง 1 address เท่านั้น ($2^8 = 256$) แต่ หากว่าอุปกรณ์นั้นมีความสามารถที่จะแสดงค่าระดับความสว่างของแสงได้มากกว่า 256 ระดับ อุปกรณ์ตัวนั้นก็จะสามารถมี Address 2 address ในตัวเองได้ โดยจะทำให้สามารถรับข้อมูลได้

เพิ่มจากเดิม 8 บิต เป็น 16 บิต และตัวอุปกรณ์เองก็สามารถที่จะแสดงผลได้สูงถึง 65,536 ระดับ ($2^{16} = 65,536$)



รูปที่ 2.2 ตัวอย่างอุปกรณ์ที่ใช้ Address ของ DMX มากกว่า 1 Address ในตัวเดียว

2.1.3 จังหวะสัญญาณ DMX512 (Timing diagram)



รูปที่ 2.3 จังหวะมาตรฐานของสัญญาณซีเอ็มเอ็มเอกซ์

รูปที่ 2.3 โพรโทคอลมาตรฐานของสัญญาณ DMX512 ประกอบไปด้วยคำสั่งต่าง ๆ ซึ่งในช่วงสภาวะบัสว่างหรือช่วงที่ไม่มีการส่งสัญญาณ จะมีค่าเป็น '1' ซึ่งเรียกว่า IDLE และหากเริ่มมีการส่งสัญญาณ ตัวอุปกรณ์มาสเตอร์จะส่งสัญญาณ Break ซึ่งมีค่าเป็น '0' อย่างน้อย 22 บิต หรือ 88 ไมโครเซก ซึ่งสัญญาณ Break นี้สามารถที่จะยาวนานกว่า 88 ไมโครเซกได้ แต่ต้องไม่เกิน 1 วินาที หลังจากที่ถูกอุปกรณ์มาสเตอร์ส่งสัญญาณ Break แล้วจะส่งสัญญาณ MAB หรือเรียกว่า Mark after break ซึ่งจะต้องมีค่าเท่ากับ 2 บิต หรือ 8 ไมโครวินาที จากนั้นอุปกรณ์มาสเตอร์จะส่งชุดข้อมูลออกมาเรียงลำดับตั้งแต่ชุดที่ 1 ถึงชุดที่ 512 ซึ่งในแต่ละชุดข้อมูลจะประกอบไปด้วย สตาร์ทบิต, คำคำบิต(8บิต), สตอปบิต(2บิต), และ MTBF หรือ Mark Time Between Frame ซึ่ง MTBF นี้จะมีระยะเวลาระหว่าง 0 – 1 วินาที เมื่ออุปกรณ์มาสเตอร์ ส่งข้อมูลทั้ง 512 ข้อมูลเรียบร้อยแล้ว ก็ จะทำการส่ง MTBP หรือ Mark Time Between Package ซึ่งเป็นเหมือนช่วงหน่วงเวลาระหว่างส่งข้อมูลแต่ละชุด แสดงในตารางที่ 2.1

IDLE	เป็นสัญญาณที่ส่งออกมาในขณะที่ไม่มีการส่ง Packet DMX ออกมา โดยสภาวะลอจิกจะเป็น '1'
BREAK	BREAK สัญญาณเริ่มต้นก่อนส่ง DMX Packet โดยจะกำหนดให้ output เป็นลอจิก '0' อย่างน้อย 88 uSec หรือเท่ากับ 22 บิต สัญญาณ BREAK นี้สามารถยาวนานกว่า 88 uSec ได้
MAB	MAB (Mark After Break) สัญญาณ MAB นี้จะส่งต่อจาก BREAK โดยจะส่งสภาวะลอจิก '1' อย่างน้อย 8 uSec หรือ 2 พัลส์
START Bit	START Bit สัญญาณ START Bit จะถูกส่งเป็นบิตแรกก่อนเริ่มส่งข้อมูลทุกครั้ง โดยมีความยาว 1 บิต
DATA Bit	DATA Bit สัญญาณ DATA Bit จะถูกส่งต่อจาก START Bit โดยจะมีความยาวเท่ากับ 8 บิต และจะเริ่มส่งจาก LSB ก่อน
STOP Bit	STOP Bit เมื่อส่ง DATA Bit เรียบร้อยแล้วสัญญาณ STOP Bit จะถูกส่งออกมา โดยมีความยาวเท่ากับ 2 บิต
MTBF	MTBF หรือ Mark Time Between Frame สัญญาณนี้จะถูกส่งออกหลังจากส่ง Data Frame ครบ 1 Frame โดยสถานะจะเป็น '1' ซึ่งจะสามารถใช้ช่วงเวลาตั้งแต่ 0 วินาที จนถึง 1 วินาทีได้ โดยส่วนมากอุปกรณ์ส่วนใหญ่จะไม่สามารถรองรับการทำงานที่ 0 วินาทีได้
MTBP	MTBP หรือ Mark Time Between Packets จะถูกส่งออกมาหลังจาก Data Frame สุดท้าย จะเป็นเหมือนกับ Delay ก่อนที่จะมีการส่ง Data ชุดใหม่อีกครั้ง

ตารางที่ 2.1 คำอธิบายคำสั่งต่าง ๆ ของโพรโทคอลดีเอ็มเอ็กซ์

2.2 อัตราของการแสดงผลสูงสุดใน 1 วินาที (Frame Rate)

อัตราการแสดงผลสูงสุดใน 1 วินาที จะเท่ากับจำนวนครั้งที่อุปกรณ์ Master สามารถส่งข้อมูล Packet ให้แก่อุปกรณ์ Slave ใน 1 วินาที โดยที่ระยะเวลาต่าง ๆ ของ สัญญาณแสดงในตารางที่ 2.2

ชื่อสัญญาณ	จำนวนบิตต่ำสุดที่ใช้	เวลาที่ใช้ (us)
BREAK	22	88
MAB	2	8
START	1	4
DATA	8	32
STOP	2	8
MTBF	0	0
MTBP	0	0

ตารางที่ 2.2 ระยะเวลาที่ใช้ต่ำสุดในแต่ละคำสั่งของโปรโตคอลดีเอ็มเอ็กซ์

ใน 1 Package จะประกอบไปด้วย BREAK + MAB + (512 Data frames) + (511 MTBF) + MTBP ดังนั้นระยะเวลาที่สั้นที่สุดที่ใช้ในการส่ง Package 1 Package เท่ากับ

$$\begin{aligned}
 &= 88 + 8 + (512 \times (4 + 32 + 8)) + (511 \times 0) + 0 \text{ us} \\
 &= 22,624 \text{ us} \\
 &= 22.624 \text{ ms}
 \end{aligned}$$

ดังนั้น ใน 1 วินาที สามารถแสดงผลได้สูงสุดเท่ากับ

$$\begin{aligned}
 F &= 1/T \\
 &= 1/22.624 \text{ ms} \\
 &= 44 \text{ Frame ต่อวินาที}
 \end{aligned}$$

2.3 งานวิจัยที่เกี่ยวข้อง

ในหัวข้อนี้จะกล่าวถึงวิธีการและผลงานต่าง ๆ ที่เกี่ยวข้องกับรวมถึงงานวิจัย เพื่อที่จะนำไปใช้เป็นแนวทางในการพัฒนางานวิจัย

2.3.1 การออกแบบแอลอีดีเมทริกซ์ (LED Matrix) ด้วยเอฟทีจีเอ โดยใช้โปรโตคอลดีเอ็มเอ็กซ์ 512

การสร้างอุปกรณ์แสดงผลทางแสงโดยใช้ LED เป็นที่นิยมอย่างแพร่หลาย รวมถึงการสร้างแอลอีดีเมทริกซ์ (LED Matrix) ที่ซึ่งสามารถนำไปแสดงผลเป็นภาพ หรือข้อความได้ แต่โปรโตคอลในการควบคุมนั้นก็แตกต่างกันไปตามแต่บริษัทผู้ผลิตได้ทำการออกแบบ และโปรโตคอลที่เป็นที่นิยมมากที่สุด ในการควบคุมอุปกรณ์ส่องสว่างคือดีเอ็มเอ็กซ์ 512 (DMX512) ซึ่ง Carlos Lopez และคณะ ได้ทำการออกแบบแอลอีดีเมทริกซ์ ที่สามารถรองรับการทำงานของโปรโตคอลดีเอ็มเอ็กซ์ 512 ได้ [3] ซึ่งแสดงให้เห็นว่าแอลอีดีเมทริกซ์ สามารถที่จะทำงานโดยใช้โปรโตคอลการควบคุมแบบ DMX512 ได้เป็นอย่างดี แต่ข้อเสียคือจำเป็นต้องใช้ตำแหน่งที่อยู่ (Address) ของโปรโตคอลดีเอ็มเอ็กซ์ จำนวนมากเพื่อที่จะทำให้สามารถสร้างจอแสดงผลขนาดใหญ่ได้

โดยในการออกแบบนั้นได้ออกแบบให้มีขนาด 10 x 7 จุด และใช้แอลอีดีจำนวน 70 ดวง แต่ละดวงจะมี 3 สีในตัวนั่นคือ แดง เขียว และน้ำเงิน ซึ่งแอลอีดีแต่ละสีใน 1 ดวงนั้นสามารถควบคุมได้อิสระจึงทำให้ต้องมี Address ประจำอยู่ที่แต่ละสีของแอลอีดีแต่ละดวง ดังนั้นจำนวน Address ที่ใช้สำหรับควบคุมอุปกรณ์ตัวนี้จะเท่ากับ 210 Address ตำแหน่ง Address แต่ละตำแหน่งนั้นจะมีขนาด 8 บิตซึ่งทำให้สามารถปรับความสว่างโดยวิธีดับเบิลยูเอ็ม (PWM) ได้ 256 ระดับ ทำให้แอลอีดี 1 ดวงนั้นสามารถที่จะแสดงผลได้มากกว่า 16 ล้านสี

2.3.2 การวิจัยเรื่องระบบ LED Display ที่รวมโปรโตคอล CAN (Controller Area Network) Bus และ DMX512 ไว้ด้วยกัน [4]

ตามพื้นฐานของโปรโตคอลแบบ DMX512 นั้นมีข้อด้อย เช่น มีความน่าเชื่อถือน้อย, มีความเร็วต่ำ, ระยะทางสั้น ซึ่งไม่สามารถที่จะรองรับความต้องการที่เพิ่มขึ้นของตลาดได้ ดังนั้น Feng Tian [4] และคณะ ได้ทำการพัฒนาโปรโตคอล DMX512 โดยใช้วิธีการรวม CAN Bus ไว้กับ DMX512 ซึ่งจะให้มีประสิทธิภาพเพิ่มขึ้นทั้ง ความน่าเชื่อถือ, ความเร็ว, และระยะทางที่ไกลขึ้น ผลงานของพวกเขาสามารถพัฒนาและนำไปใช้งานได้อย่างกว้างขวาง โดยสามารถนำไปควบคุมอุปกรณ์ส่องสว่างที่มีระยะทางระหว่างจุดห่างกันไกลได้ เช่น สนามบิน หรือสนามกีฬา

2.3.3 การวิจัยระบบส่องสว่างของสถานที่โดยใช้โปรโตคอล DMX512 [5]

โปรโตคอล DMX512 ได้มีจุดมุ่งหมายสำหรับควบคุมอุปกรณ์ต่างๆ สำหรับระบบเวที ดังนั้นมาตรฐานต่างๆ ที่ได้กำหนดมาแล้วนั้น มีความเหมาะสมเป็นอย่างมาก แต่ในปัจจุบันได้มีการนำโปรโตคอลชนิดนี้ไปใช้ร่วมกับอุปกรณ์อื่นๆ อีกหลายชนิด จึงมีความต้องการที่จำเป็นต้องใช้งานจำนวนอุปกรณ์มากกว่าที่จะรองรับได้ใน 1 ช่องสัญญาณ อีกทั้งความต้องการเรื่องความเร็วมีการ

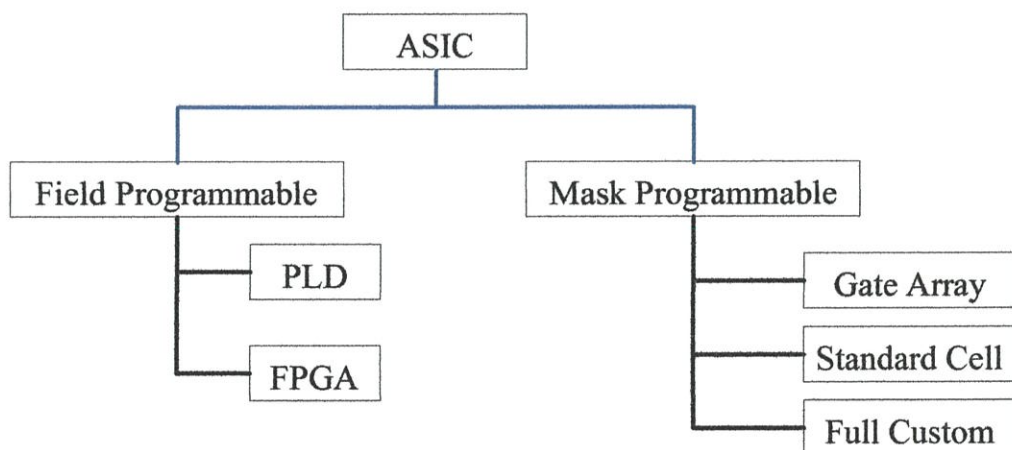
เพิ่มขึ้นเพื่อให้สามารถแสดงภาพเคลื่อนไหวได้ โดยที่ Huifen Wang [5] และคณะ ได้ทำการ ออกแบบตัวควบคุม DMX512 ที่ใช้ในการควบคุมระบบส่องสว่างของสถานที่โดยสามารถที่จะ ควบคุมการทำงาน และข้อมูลผ่านทางอินเทอร์เน็ต (Internet) ได้ โดยใช้ไมโครคอนโทรลเลอร์

บทที่ 3

สถาปัตยกรรมและการออกแบบบน FPGA

3.1 บทนำ

ปัจจุบันอุตสาหกรรมทางด้านอิเล็กทรอนิกส์มีความรวดเร็วไปอย่างรวดเร็วมาก และอีกทั้งประสิทธิภาพที่เพิ่มขึ้นของอุปกรณ์อิเล็กทรอนิกส์ เพื่อให้รองรับกับความต้องการของการนำมาใช้กับงานที่มีความซับซ้อนมากขึ้นเรื่อยๆ นั้น สิ่งสำคัญที่สุดย่อมขึ้นอยู่กับประสิทธิภาพของชิ้นส่วนแต่ละชิ้นที่อยู่ภายในอุปกรณ์อิเล็กทรอนิกส์โดยจะเป็นส่วนประกอบที่สำคัญอย่างมากซึ่งก็คือ วงจรรวม หรือที่เราเรียกกันสั้นๆ ว่า ไอซี (IC, integrated circuit) หรือ ไมโครชิป สำหรับตัวอย่างของการพัฒนาเทคโนโลยีไอซีที่พบเห็นได้ชัดเจนนั้น เช่น เทคโนโลยีไมโครโพรเซสเซอร์(MPU), ไมโครคอนโทรลเลอร์(MCU) และหน่วยความจำ(memory) ซึ่งในท้องตลาดปัจจุบันนี้มีไอซีหลายชนิดให้เลือกนำมาใช้งานกัน ตามความเหมาะสมทั้งด้านราคาและประสิทธิภาพ โดยเฉพาะอย่างยิ่งไอซีที่มีประสิทธิภาพสูงๆ ที่ใช้ในงานเฉพาะด้านหรือที่เรียกว่า ASIC (Application Specific Integrated Circuit, ASIC) ซึ่งจะสามารถจัดแบ่งตามโครงสร้างออกได้เป็น 2 กลุ่มใหญ่ๆ คือ Field programmable และ Mask programmable ดังแสดงในรูปที่ 3.1



รูปที่ 3.1 การแบ่งกลุ่มของวงจรรวม ASIC

ถ้าจะพิจารณาถึงข้อดี ในเชิงการออกแบบและการผลิต ไอซีตระกูล mask programmable นั้น อาจจะทำให้ได้ชิป ASIC ที่มีประสิทธิภาพสูงกว่า แต่ต้องมีการส่งวงจรไปทำหน้ากาก (mask set) และเอกสารยังต่างประเทศ จึงทำให้มีค่าใช้จ่ายเริ่มต้น (fixed cost) ที่สูงมาก ซึ่งเหมาะแก่การใช้งานที่ต้องการปริมาณ (volume) สูงเพื่อเฉลี่ยค่าใช้จ่ายเริ่มต้นเท่านั้น สำหรับการพัฒนาชิป ASIC ในปริมาณน้อยถึงปานกลางนั้น ควรใช้ชิป ASIC ชนิดโปรแกรมได้ (field programmable) ชนิดใดชนิดหนึ่งซึ่งสามารถหาซื้อได้ในราคาข้อมเยากกว่า โดย

ผู้ใช้งานสามารถออกแบบและสร้างวงจรที่ต้องการใช้ลงในตัวอุปกรณ์ได้เอง โดยไม่ต้องไปผลิตที่โรงงาน จึงเป็นทางเลือกที่น่าสนใจ โดยเฉพาะอย่างยิ่ง เทคโนโลยีที่ก้าวหน้าไปมาก ทำให้ประสิทธิภาพของไอซีแบบโปรแกรมได้สามารถตอบสนองต่อความต้องการของการใช้งานที่มีความซับซ้อนสูงได้

3.2 เอฟพีจีเอ (FPGA: Field Programmable Gate Array)

FPGA เป็นอุปกรณ์ที่ถูกพัฒนาต่อจากอุปกรณ์แอลซีเอ ของบริษัทไซลิงซ์ (XILINX Inc.) โดยมีจุดเด่นที่ประสิทธิภาพการทำงานและปริมาณความหนาแน่นของจำนวนเกตสูง การใช้งานที่สะดวก สามารถกำหนดฟังก์ชันการทำงานได้ตามความต้องการของผู้ใช้งาน โดยผ่าน Software เพื่อจะทำการโปรแกรมข้อมูลลงในชิป FPGA ให้ทำงานตามที่ออกแบบไว้โดยอัตโนมัติ ดังนั้นการออกแบบวงจรโดยใช้เอฟพีจีเอ สามารถออกแบบและทดสอบได้ภายในเวลาไม่นานเพียง 2-3 วัน เท่านั้น ซึ่งตรงกับข้ามกับการออกแบบโดยใช้เกตอาร์เรย์ ซึ่งจะใช้เวลาานหลายสัปดาห์ การเปลี่ยนแปลงแก้ไขต้นแบบก็เป็นเช่นเดียวกัน จากประโยชน์ของเอฟพีจีเอ ดังที่กล่าวมานั้น ทำให้เกิดการประหยัดค่าใช้จ่ายเป็นอย่างมาก เพราะได้ลดความเสี่ยงในการที่จะต้องแก้ไขตัววงจรต้นแบบ การเลื่อนเวลาการออกผลิตภัณฑ์ และลดค่าเอ็นอาร์อี (NRE: Nonrecurring Engineering Cost) ลงไปด้วย

3.2.1 เปรียบเทียบกับไมโครคอนโทรลเลอร์ (Microcontroller)

ไมโครคอนโทรลเลอร์ เป็นชิปประเภทใช้ประโยชน์ทั่วไป (general purpose devices) หมายความว่าสถาปัตยกรรม รวมถึงชุดคำสั่งทั้งหมดของชิปนั้นๆ ได้ถูกออกแบบและกำหนดมาเรียบร้อยแล้ว ผู้ใช้มีหน้าที่เพียงเรียบเรียงชุดคำสั่ง เพื่อให้ชิปปฏิบัติหน้าที่ตามที่ต้องการ ขอบจำกัดทางด้านสถาปัตยกรรมและชุดคำสั่งทำให้ ไมโครคอนโทรลเลอร์ไม่ใช่คำตอบของงานหลายงาน ตัวอย่างเช่น ไมโครคอนโทรลเลอร์ที่ทำงานที่ความเร็ว 20MHz ย่อมไม่สามารถใช้กับงานที่ต้องการความเร็วมากกว่านั้น ซึ่งอาจเป็นคุณสมบัติจำเป็นสำหรับงานในหลายๆด้าน เช่น โทรคมนาคม หรือ งานด้านการประมวลผลภาพ (Image processing) ทั้งหมดนี้ต่างกับการใช้ FPGA ที่ผู้ใช้เป็นคนออกแบบสถาปัตยกรรมทั้งหมดเอง โดยที่พื้นฐานเทคโนโลยี FPGA ในปัจจุบันสามารถทำงานที่สัญญาณนาฬิกาสูงกว่า 100MHz เพราะฉะนั้นประสิทธิภาพจริงๆของชิป จะขึ้นกับสถาปัตยกรรมที่ผู้ออกแบบคิดค้นและใช้เป็นหลัก นอกจากนั้นผู้ใช้อย่างยังสามารถออกแบบชุดคำสั่งใหม่ๆ ขึ้นเอง เพื่อให้เหมาะกับการใช้งานนั้นๆ ได้ รวมถึงความสามารถในการรวบรวมวงจรดิจิทัลอื่นๆ นอกชิปเข้ามาอยู่รวมกันใน FPGA เดียวได้ FPGA จึงเป็นมากกว่าทางเลือกหนึ่งของ ไมโครคอนโทรลเลอร์ แต่เป็นคำตอบที่ดีกว่าในงานออกแบบดิจิทัล

3.2.2 จุดเด่นของ FPGA

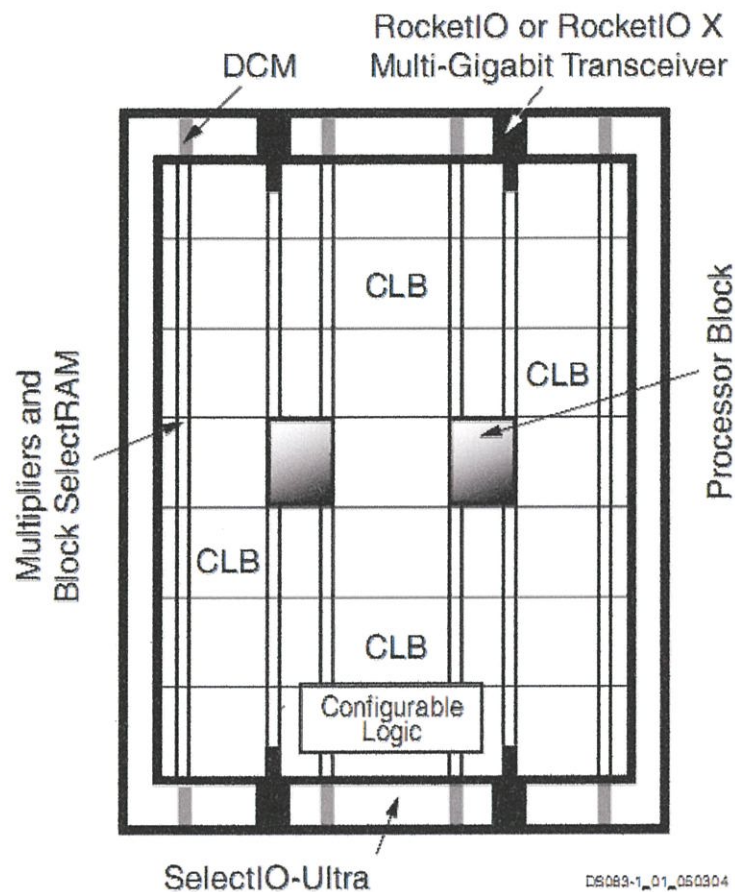
นอกจากระยะเวลาที่ใช้ในการออกแบบและพัฒนาจะเร็วใกล้เคียงกับการใช้ ชิปประเภทใช้ประโยชน์ทั่วไป ขณะที่ประสิทธิภาพใกล้เคียงกับชิป ASIC แบบ full-custom ซึ่งหมายถึงชิปที่ถูกออกแบบและพัฒนาใหม่ทั้งหมดเฉพาะงานนั้นๆ ซึ่งมีราคาแพงกว่ามากแล้ว ผู้ใช้ชิป FPGA ยังสามารถนำวงจรที่พร้อมใช้งานที่มีผู้ออกแบบไว้แล้ว ในรูปของทรัพย์สินทางปัญญา หรือ (IP-cores: intellectual property cores) มาใช้รวมกันวงจรที่เราออกแบบเองได้ทันที ซึ่งเป็นการเพิ่มประสิทธิภาพ และย่นเวลาการพัฒนาชิป FPGA ของเราได้มาก

3.2.3 FPGA ในท้องตลาดและการเลือกใช้

ปัจจุบันมีผู้ผลิตชิป FPGA อยู่หลายบริษัท เช่น Xilinx Corp., Altera หรือ Actel โดยบริษัทที่มีขนาดใหญ่ที่สุด และมีส่วนแบ่งทางการตลาดสูงที่สุดคือ Xilinx (www.xilinx.com) ปกติ FPGA ของแต่ละบริษัท จะมีหลายตระกูลขึ้นอยู่กับราคาและขนาดความจุของชิป FPGA ซึ่งมักถูกวัดด้วยค่าจำนวนเกตเทียบเท่า (equivalent gate) โดยปัจจุบัน FPGA จะมีขนาดตั้งแต่ น้อยกว่า 10,000 เกต จนถึงมากกว่า 5 ล้านเกตขึ้นอยู่กับตระกูล เช่น Xilinx ตระกูล 9,500 ซึ่งใช้เทคโนโลยี EEPROM มีขนาดความจุสูงสุดถึงประมาณ 7,000 เกต ขณะที่ Xilinx ตระกูล Spartan2 มีขนาดความจุสูงสุดถึงประมาณ 2 แสนเกต และสำหรับปัจจุบัน Xilinx ตระกูล Virtex ซึ่งใช้เทคโนโลยี SPAM จะมีราคาแพงกว่า แต่มีขนาดความจุเพิ่มขึ้นถึงกว่า 3 ล้านเกต เพื่อให้เป็นภาพเปรียบเทียบไมโครคอนโทรลเลอร์ ตระกูล 8051 มีขนาดเกตประมาณ 15,000 เกต ขนาดความจุของเกต ที่มากกว่าหมายถึงศักยภาพในการใช้งาน FPGA ตัวนั้น สำหรับงานที่มีความซับซ้อนมากกว่า

3.3 รายละเอียดสถาปัตยกรรมของ FPGA ตระกูล VIRTEX II Pro

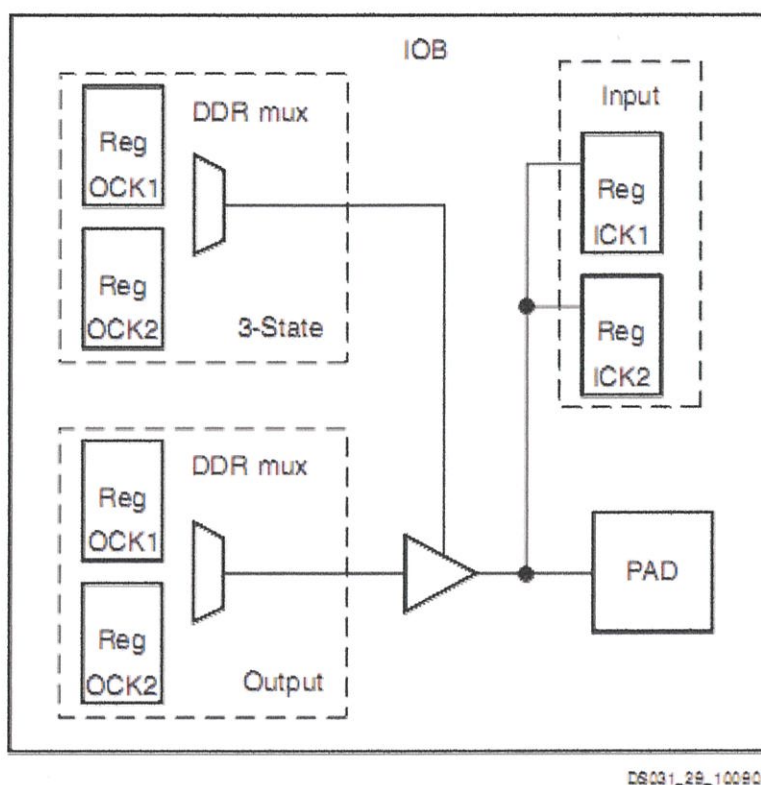
รูปที่ 3.2 แสดงบล็อกไดอะแกรมของโครงสร้างชิป FPGA ตระกูล VIRTEX II Pro ประกอบด้วย Configurable Logic Block (CLBs) ซึ่งเป็นบล็อกลอจิกวางโครงแบบได้ ที่มีลักษณะสม่ำเสมอยืดหยุ่นได้และโปรแกรมได้ บล็อก CLB ทั้งหมดจะถูกล้อมรอบด้วย I/O Blocks (IOBs) ที่โปรแกรมได้ บล็อกต่างๆ เหล่านี้เชื่อมต่อระหว่างกันด้วยแหล่งการเชื่อมโยง (Routing Resources) อันทรงประสิทธิภาพ สถาปัตยกรรมนี้มีความสามารถรองรับฟังก์ชันขั้นสูง อย่างบล็อก RAM และ บล็อกสัญญาณนาฬิกาควบคุม(clock control blocks)



รูปที่ 3.2 บล็อกไดอะแกรมของโครงสร้าง FPGA ตระกูล VIRTEX II Pro

3.3.1 บล็อกอินพุต/เอาต์พุต (IOB)

รูปที่ 3.3 แสดงบล็อกอินพุต/เอาต์พุต (IOB) ของโครงสร้าง FPGA ตระกูล VIRTEX II Pro ซึ่งรองรับมาตรฐานการให้สัญญาณ I/O ความเร็วสูงได้ถึง 16 มาตรฐาน เช่น LVCMOS, HTSL, SSTL, และ GTL I/O ความเร็วสูงนี้สามารถรองรับหน่วยความจำ และการเชื่อมต่อของบัส (bus interface) ที่ทันสมัยได้หลากหลายแบบ รีจิสเตอร์ IOB ทั้งสามตัว ทำหน้าที่เป็น edge-triggered D flip-flop (ฟลิปฟล็อปแบบประวิงชนิดที่ใช้ขอบสัญญาณนาฬิกา จุดชนวนการทำงาน) หรือไม่ก็เป็น (แอสตซ์แบบไวต่อระดับสัญญาณนาฬิกา)

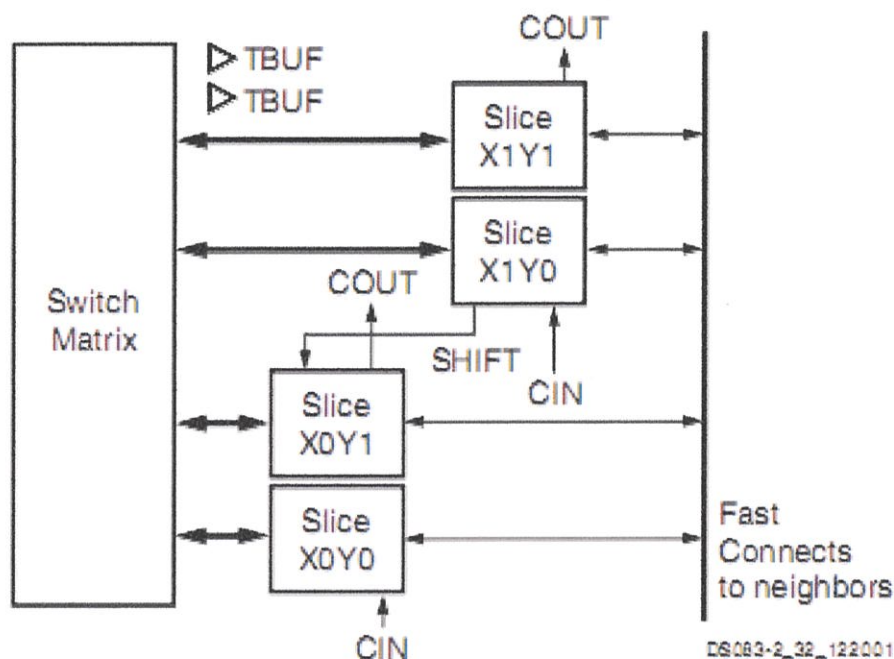


รูปที่ 3.3 บล็อกอินพุต/เอาต์พุต (IOB) ของโครงสร้าง FPGA ตระกูล VIRTEX II Pro

3.3.2 ลอจิกเซลล์ (Logic Cells)

ลอจิกเซลล์ (LC) เป็นส่วนประกอบพื้นฐานสำหรับ CLB ใน VIRTEX II Pro ดังแสดงในรูปที่ 3.4 ในเซลล์ลอจิกเซลล์หนึ่งมีตัวกำเนิดฟังก์ชันแบบ 4 input จำนวน 1 ตัว พร้อมลอจิกทอค (carry logic) และส่วนสำรองข้อมูล (storage element) เอาต์พุตจากตัวกำเนิดฟังก์ชันในแต่ละ LC เป็นสัญญาณขับทั้ง CLB output และ D input ของฟลิปฟล็อป ใน CLB แต่ละบล็อกของ VIRTEX II Pro จะประกอบด้วย LC จำนวน 4 เซลล์ จัดเรียงใน slice 4 slice นอกจากนี้พื้นฐานทั้งสี่แล้ว VIRTEX II Pro CLB ยังมีลอจิกที่รวมเอาตัวกำเนิดฟังก์ชันที่ให้ฟังก์ชัน 5-6 input ไว้ด้วย ยังผลให้ CLB มีจำนวน LC ที่นับได้จริงเท่ากับ 4.5 เซลล์

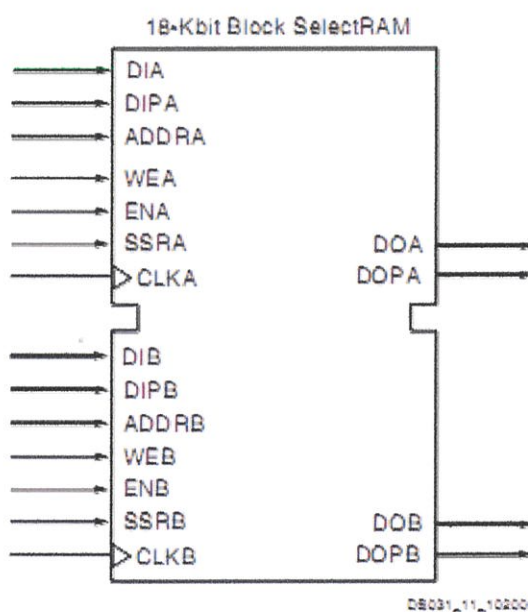
ตัวกำเนิดฟังก์ชันของ VIRTEX II Pro มีลักษณะเป็นตารางค้นหา (Look-Up Table หรือ LUTs) แบบ 4 input นอกจากเป็นตัวกำเนิดฟังก์ชันแล้ว LUT เหล่านี้สามารถทำหน้าที่เป็น 16 x 1-bit synchronous RAM หรือเป็น 16-bit shift register ที่เหมาะสมสำหรับบันทึกข้อมูลความเร็วสูงหรือข้อมูลที่ถูกส่งเป็นชุดอย่างรวดเร็ว (burst-mode data) ซึ่งพบบ่อยในงานจำพวก Digital Signal Processing (DSP) ส่วนสำรองข้อมูลใน VIRTEX II Pro slice อาจวางให้เป็นแบบ edge-triggered D flip-flops หรือเป็น level-sensitive latches ก็ได้



รูปที่ 3.4 บล็อก CLB ของ FPGA ตระกูล VIRTEX II Pro (สมมาตรในแต่ละ CLB)

3.3.3 บล็อกแรม (Block RAM)

บล็อกแรมพอร์ตคู่ของ VIRTEX II Pro FPGA แสดงดังรูปที่ 3.5 จะรวมเอาบล็อก RAM ที่เลือกกับหน่วยความจำขนาด 18Kbits สิ่งเหล่านี้ช่วยเสริมเติมเต็มการกระจาย RAM ที่ถูกเลือก กับ แหล่งที่ทำให้ทำให้โครงสร้าง RAM บางลง ดังในรูป CLB แต่ละบล็อกสามารถคอนฟิกูเรชันที่อัตราส่วนระหว่าง 16Kx1 และ 512x36



รูปที่ 3.5 บล็อกแรมพอร์ตคู่

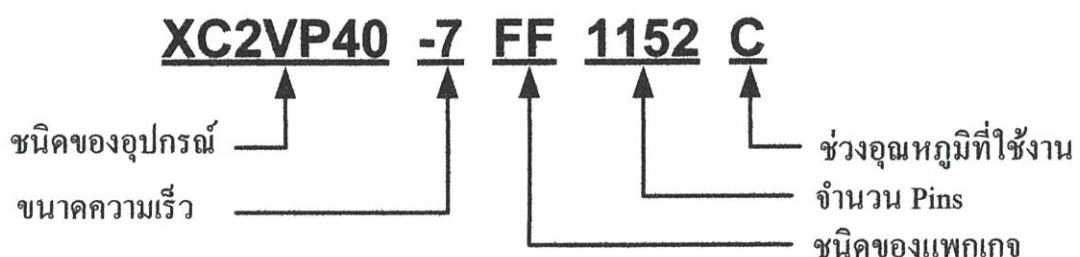
3.3.4 ดีเลย์ล็อกคูลูป (Delay-Locked Loop)

ดีเลย์ล็อกคูลูป (DLL) จะทำงานร่วมกับทุกสัญญาณนาฬิกาครอบคลุมถึงบัฟเฟอร์ (buffer) ในการกำจัดการหมุนวน (skew) ระหว่างขาสัญญาณนาฬิกาเข้า (clock input pad) และขาสัญญาณนาฬิกาเข้าภายใน (internal clock input pins) ทั้งอุปกรณ์ DLL แต่ละตัวสามารถจับสัญญาณนาฬิกาได้ครอบคลุมทั้งเครือข่าย (global clock networks) ได้ถึง 2 เครือข่าย DLL จะเฝ้าสังเกตสัญญาณนาฬิกาเข้า และสัญญาณนาฬิกาที่ถูกกระจาย แล้วทำการปรับอุปกรณ์หน่วงสัญญาณนาฬิกา (clock delay element) โดยอัตโนมัติ เมื่อ สัญญาณนาฬิกาเดินทางถึงอินพุตแล้ว DLL จะทำให้เกิดการหน่วงเพิ่มเติมเพื่อให้ขอบสัญญาณนาฬิกา (clock edge) เดินทางถึงฟลิปฟล็อปภายในวงจร ในเวลา 1 คาบ ทำให้มั่นใจว่าขอบสัญญาณภายในฟลิปฟล็อป สามารถประสานเวลากับขอบสัญญาณที่อินพุตได้เป็นอย่างดี

ตารางที่ 3.1 ตารางเปรียบเทียบคุณสมบัติ FPGA ตระกูล VIRTEX II Pro series

Device ⁽¹⁾	RocketIO Transceiver Blocks	PowerPC Processor Blocks	Logic Cells ⁽²⁾	CLB (1 = 4 slices = max 128 bits)		18 X 18 Bit Multiplier Blocks	Block SelectRAM+		DCMs	Maximum User I/O Pads
				Slices	Max Distr RAM (Kb)		18 Kb Blocks	Max Block RAM (Kb)		
XC2VP2	4	0	3,168	1,408	44	12	12	216	4	204
XC2VP4	4	1	6,768	3,008	94	28	28	504	4	348
XC2VP7	8	1	11,088	4,928	154	44	44	792	4	396
XC2VP20	8	2	20,880	9,280	290	88	88	1,584	8	564
XC2VPX20	8 ⁽⁴⁾	1	22,032	9,792	306	88	88	1,584	8	552
XC2VP30	8	2	30,816	13,696	428	136	136	2,448	8	644
XC2VP40	0 ⁽³⁾ , 8, or 12	2	43,632	19,392	606	192	192	3,456	8	804
XC2VP50	0 ⁽³⁾ or 16	2	53,136	23,816	738	232	232	4,176	8	852
XC2VP70	16 or 20	2	74,448	33,088	1,034	328	328	5,904	8	996
XC2VPX70	20 ⁽⁴⁾	2	74,448	33,088	1,034	308	308	5,544	8	992
XC2VP100	0 ⁽³⁾ or 20	2	89,216	44,096	1,378	444	444	7,992	12	1,164

3.3.5 ข้อมูลเกี่ยวกับหมายเลข FPGA

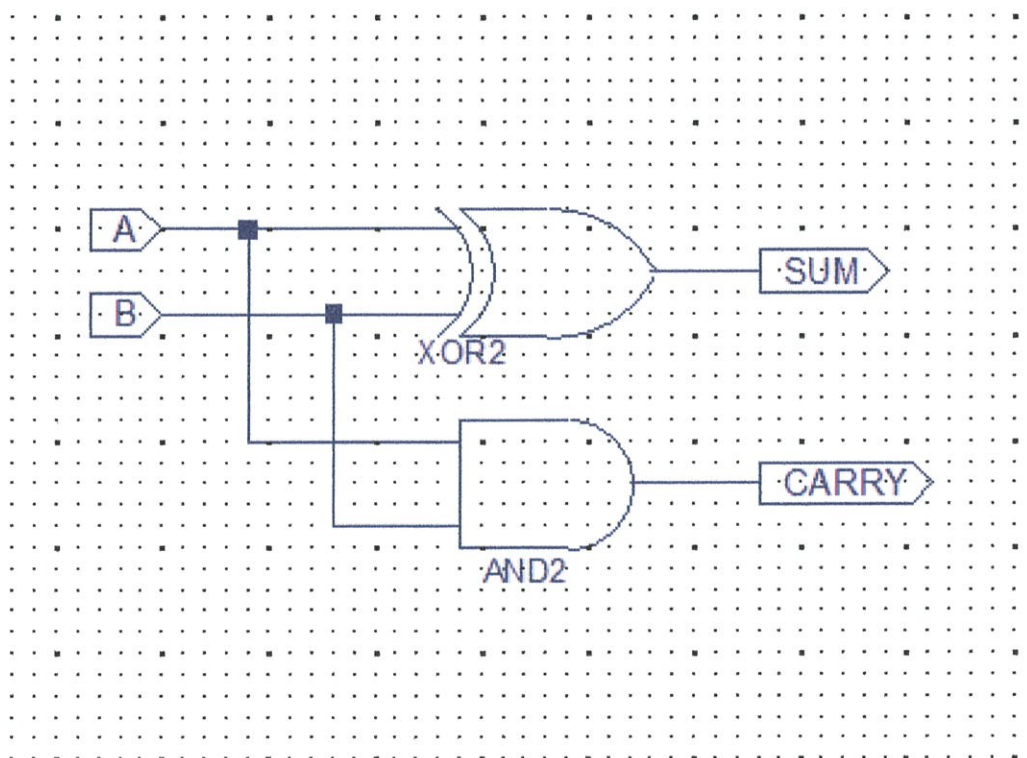


ชนิดและหมายเลข FPGA ตระกูล VIRTEX II Pro series

3.4 การออกแบบวงจรดิจิทัลในปัจจุบัน

รูปแบบการออกแบบวงจรดิจิทัลสามารถแบ่งออกได้เป็นหลายรูปแบบดังแสดงต่อไปนี้

3.4.1 การออกแบบโดยใช้การวาดผังวงจร (Schematic design)



รูปที่ 3.6 การออกแบบโดยการวาดผังวงจร

ในอดีตการออกแบบวงจรดิจิทัลที่เรารู้จักกัน จะเป็นการออกแบบวงจรในระดับลอจิกเกต (Logic Level) โดยอาศัยโปรแกรมช่วยในการวาดผังวงจรดังแสดงในรูปที่ 3.6 ซึ่งการออกแบบวงจรลักษณะนี้ค่อนข้างใช้เวลามากในการออกแบบ เนื่องจากการออกแบบทุกอย่างผู้ออกแบบจะต้องออกแบบเองทั้งหมด ซึ่งถ้าวงจรที่ต้องการออกแบบมีความซับซ้อนสูง ก็จะทำให้การออกแบบทำได้ยาก ซึ่งเป็นข้อจำกัดให้ผู้ออกแบบไม่สามารถออกแบบวงจรที่มีความซับซ้อนสูงได้ เช่น การออกแบบซีพียู และไมโครคอนโทรลเลอร์ เป็นต้น

3.4.2 การออกแบบโดยใช้ภาษาระดับสูง (HDL)

```

signal g_ccp_dat0,g_ccp_dat1,g_ccp_dat2 : std_logic_vector(7 downto 0);
signal g_ccp_dat3,g_ccp_dat4,g_ccp_dat5 : std_logic_vector(7 downto 0);
signal b_ccp_dat0,b_ccp_dat1,b_ccp_dat2 : std_logic_vector(7 downto 0);
signal b_ccp_dat3,b_ccp_dat4,b_ccp_dat5 : std_logic_vector(7 downto 0);
--***= Write Enable Ram ==*-
signal we0,we1,we2,we3,we4,we5 : std_logic_vector(0 downto 0);

--***** Component *****-
COMPONENT read_ram_ctrl_core_0
PORT(
    clk : IN std_logic;
    rst : IN std_logic;
    spi_done : IN std_logic;
    swap : OUT std_logic;
    blank : out STD_LOGIC;
    shift_data : OUT std_logic;
    line_driver : out STD_LOGIC_VECTOR (2 downto 0);
    addr_read : OUT std_logic_vector(6 downto 0);
    counter_out : OUT std_logic_vector(7 downto 0)
);
END COMPONENT;

COMPONENT spi_ctrl_core_0
PORT(
    clk : IN std_logic;
    clk_spi : in STD_LOGIC;
    rst : IN std_logic;
    shift : IN std_logic;
    change_line : out STD_LOGIC;
    swap : IN std_logic;
    ccp_r : IN std_logic_vector(5 downto 0);
    ccp_g : IN std_logic_vector(5 downto 0);
    ccp_b : IN std_logic_vector(5 downto 0);
    spi_done : OUT std_logic;
    blank : in std_logic;
    xlat : OUT std_logic;

```

รูปที่ 3.7 การออกแบบโดยการเขียนเป็นภาษาวีเอชดีแอล

ในปัจจุบันเทคโนโลยีการออกแบบวงจรดิจิทัล ได้พัฒนาสูงขึ้นและได้มีการพัฒนาภาษาที่ใช้สำหรับออกแบบวงจรดิจิทัล พร้อมกระบวนการออกแบบแนวใหม่ ที่จะทำให้ผู้ออกแบบไม่ถูกจำกัดในการออกแบบวงจรอีกต่อไป ซึ่งการออกแบบวงจรดิจิทัลแนวใหม่นี้ จะใช้การเขียนภาษาบรรยายพฤติกรรมฮาร์ดแวร์ (HDL: Hardware Description Language) มาใช้ในการออกแบบวงจร โดยภาษาจะบรรยายพฤติกรรมฮาร์ดแวร์ที่เป็นมาตรฐานซึ่งมีอยู่ 2 ภาษาที่นิยมใช้ด้วยกันคือ ภาษา Verilog และ VHDL โดยถ้านำมาเปรียบเทียบกับในโลกของการออกแบบซอฟต์แวร์ ก็เปรียบได้กับภาษา C และภาษา Pascal นั่นเอง โดยที่ภาษา Verilog จะมีโครงสร้างคล้ายกับภาษา C ส่วนภาษา VHDL จะมีโครงสร้างคล้ายภาษา Pascal (แสดงในรูปที่ 3.7)

การออกแบบวงจรโดยใช้ภาษา HDL จะเป็นกระบวนการออกแบบที่เราเรียกกันว่า Top-Down design โดยกระบวนการดังกล่าว จะอาศัยซอฟต์แวร์ช่วยในการออกแบบ ซึ่งเราเรียกซอฟต์แวร์จำพวกนี้ว่า EDA: Electronics Design Automation ที่จะทำหน้าที่ช่วยในการจำลองการทำงานจากโค้ดที่เขียนขึ้น (HDL Simulation) และช่วยในการนำโค้ดที่ผ่านการจำลองการทำงานไปสังเคราะห์ (Synthesis) ให้เป็นผังวงจรระดับลอจิกเกต (Logic level) ในรูปแบบของเทคโนโลยี ที่ผู้ออกแบบต้องการนำไปใช้งาน ซึ่งในปัจจุบัน มีอุปกรณ์หรือชิปที่สามารถรองรับการออกแบบวงจรแบบใช้ภาษา HDL ในการออกแบบ เช่น ชิป CPLD (Complex Programmable Logic Devices) หรือชิป FPGA (Field Programmable Gate Arrays) ที่มีความจุของเกตภายในสูง สามารถออกแบบวงจรที่มีความซับซ้อนสูงได้มากกว่า 1 ล้านเกต หรือจะนำไปออกแบบในระดับของเอสิค (ASIC: Application Specific Integrated Circuits)

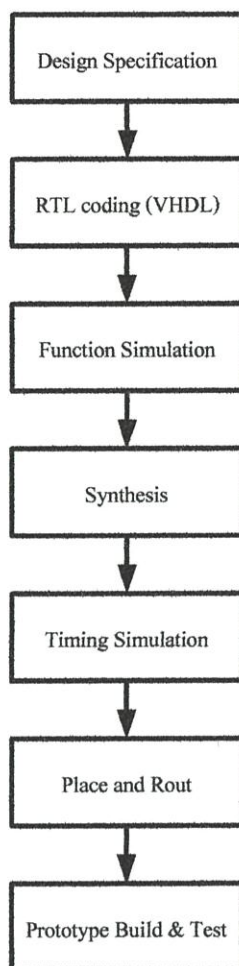
จะเห็นได้ว่าการออกแบบโดยใช้ภาษา HDL จะทำให้ผู้ออกแบบสามารถออกแบบวงจรที่ซับซ้อนได้อย่างสะดวกและรวดเร็ว และที่สำคัญการออกแบบจะไม่ยึดติดกับเทคโนโลยีใดๆ สามารถนำกลับมาใช้ใหม่ หรือกล่าวได้ว่า ถ้าผู้ออกแบบต้องการเปลี่ยนเทคโนโลยีของวงจรที่ออกแบบ ก็เพียงสังเคราะห์วงจรใหม่ ก็จะได้วงจรในเทคโนโลยีใหม่ทันที โดยไม่ต้องเสียเวลาในการวาดวงจรใหม่ เหมือนกับการออกแบบในลักษณะเดิมอีกต่อไป

3.5 การเขียนภาษา HDL

ดังที่ได้กล่าวไปแล้ว ในปัจจุบันแนวโน้มการออกแบบวงจรดิจิทัลขนาดใหญ่ จะหันไปสู่การใช้ภาษาอธิบายฮาร์ดแวร์ ซึ่งมีบทบาทมากขึ้นตามลำดับ ด้วยเหตุผลหลายประการ อาทิเช่น

- ความสามารถในการอธิบายการทำงานของวงจร ที่มีขนาดใหญ่่มาก
- ความสามารถในการนำกลับไปใช้งานได้อีกของการออกแบบที่ถูกสร้างขึ้น
- ความสามารถในการนำไปใช้งานได้กับหลายๆ เทคโนโลยี

โดยโค้ดที่ถูกเขียนจะไม่ขึ้นอยู่กับเทคโนโลยีเหล่านั้น หรือเป็นเพียงส่วนน้อยเท่านั้น นอกจากนั้นยังมีซอฟต์แวร์ที่มีประสิทธิภาพอีกมากมายสำหรับการสังเคราะห์วงจร ในปัจจุบันภาษาที่ใช้อธิบายฮาร์ดแวร์จะเป็นที่นิยมใช้ และเป็นมาตรฐานในทางอุตสาหกรรมด้วย คือ ภาษา VHDL และ Verilog HDL โดยที่ภาษาทั้งสองก็มีจุดเด่นจุดด้อย และประสิทธิภาพในการใช้งานรวมถึงซอฟต์แวร์ที่ใช้สนับสนุนที่ไม่แตกต่างกัน โดยจะขอแนะนำวิธีการเขียนภาษา HDL ทั้ง VHDL และ Verilog แต่เพียงคร่าวๆ เพื่อเป็นพื้นฐานในการออกแบบ

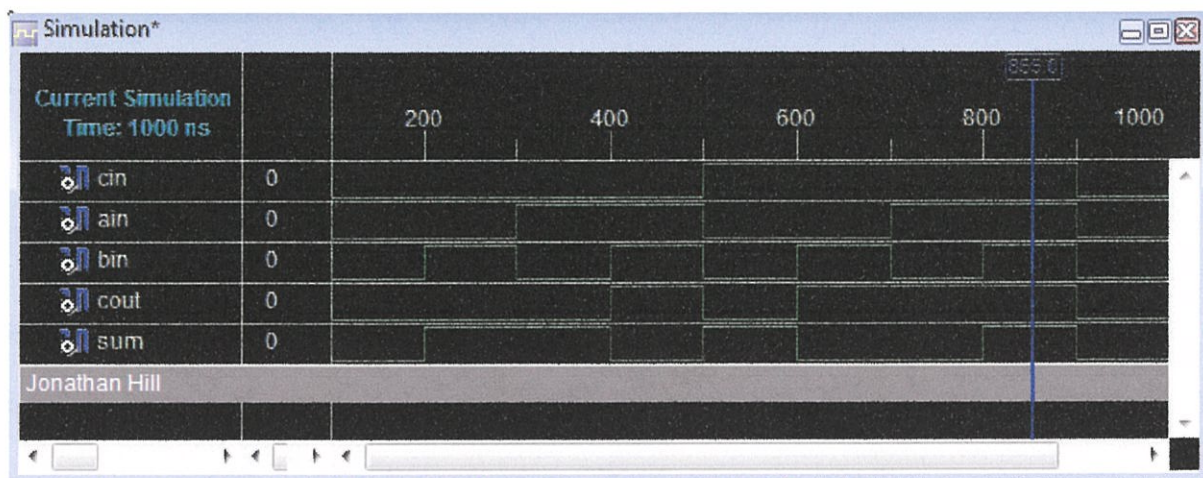


รูปที่ 3.8 ขั้นตอนการออกแบบวงจรด้วยภาษา HDL

3.5.1 ลักษณะเฉพาะในการออกแบบ (Design specifications)

รูปที่ 3.8 เป็นการแสดงการออกแบบวงจรโดยใช้ภาษา HDL จะเริ่มต้นที่การเขียนโค้ด หลังจากที่ได้มีการออกแบบสถาปัตยกรรมของวงจรอย่างดีแล้ว เช่น อาจจะเริ่มจากการแบ่งวงจรรวมออกเป็นวงจรย่อย เพื่อให้ง่ายต่อการออกแบบตามหลักการ Top-down Design Methodology ถ้าวางจรหรือบล็อกย่อยไม่ซับซ้อนจนเกินไป และง่ายต่อการสังเคราะห์วงจรในภายหลัง ก็สามารถเขียนอธิบายพฤติกรรมการทำงานของวงจรเหล่านั้น (Behavioral Description) เป็นโมเดลในภาษา HDL และเมื่อวงจรย่อยเหล่านั้นเสร็จแล้ว จึงนำมาประกอบเข้าด้วยกัน โดยการเรียกใช้ส่วนประกอบ และมีสายสัญญาณเชื่อมต่ออย่างเหมาะสม ดังนั้นการอธิบายวงจรในระดับนี้ จึงเป็นการอธิบายในเชิงโครงสร้าง (Structural Description) การเขียนโค้ดภาษา HDL เช่นเดียวกับภาษาสูงอื่นๆ สามารถใช้โปรแกรมจำพวก Text Editor ทั่วๆ ไป ได้ แต่อย่างไรก็ตาม ยังมีซอฟต์แวร์หลายตัวที่สนับสนุนการเขียนโค้ดภาษา HDL ซึ่งทำให้ง่ายขึ้นเวลาเขียนโค้ดในภาษา HDL รวมทั้งการคอมไพล์โค้ดเพื่อตรวจสอบเช็คความถูกต้องตามหลักไวยากรณ์ด้วย ตัวอย่างของคอมไพเลอร์ภาษา HDL สำหรับเครื่องพีซี ก็เช่น ซอฟต์แวร์ชื่อ ModelSim ของ Model Technology, Inc. เป็นต้น

3.5.2 การจำลองการทำงานฟังก์ชัน (Functional simulation)



รูปที่ 3.9 การจำลองการทำงานด้วยโปรแกรม ไอเอสอี ซิมูเลเตอร์

ขั้นตอนอีกขั้นตอนหนึ่งที่สำคัญในการออกแบบวงจร คือ การตรวจสอบว่าแบบจำลอง (Model) ในภาษา HDL ที่เราได้สร้างขึ้นนั้นสามารถทำหน้าที่ตามที่ได้กำหนดไว้หรือไม่ ซึ่งสามารถทำได้โดย การทำการจำลองการทำงาน เพื่อดูพฤติกรรมการทำงานของต้นแบบเหล่านั้น (Behavioral Simulation) และด้วยจุดประสงค์นี้เอง จะต้องมีการเขียนต้นแบบพิเศษสำหรับการทำการจำลองการทำงานขึ้นอีก ที่เรียกว่า HDL Testbench ซึ่งภายในจะมีการเรียกใช้หน่วยย่อย หรือส่วนประกอบที่ต้องการจะตรวจสอบพฤติกรรม การทำงาน และมีการกำหนดรูปแบบของ สัญญาณขาเข้า (Input Signal) อันเป็นการกระตุ้น (Stimulus) และเมื่อทำการจำลองการทำงาน สัญญาณขาออก (Output Signal) ของหน่วยย่อยนั้นก็จะเป็นปรากฏให้เห็นบนจอภาพ และสามารถนำไปเปรียบเทียบกับสัญญาณขาเข้าเพื่อตรวจสอบว่า การทำงานของหน่วยย่อย หรือวงจรที่ออกแบบแล้วถูกตรวจสอบ (Design Under Test หรือ DUT) เป็นไปตามที่คาดหวังไว้หรือไม่

3.5.3 การสังเคราะห์วงจร (Synthesis)

เมื่อได้ทำการตรวจสอบเป็นที่แน่ใจแล้วว่า ต้นแบบต่างๆ ที่ได้สร้างขึ้นสามารถทำงานได้อย่างถูกต้อง ขั้นตอนต่อไปก็คือ การผ่านโค้ดของต้นแบบเหล่านั้นไปทำการสังเคราะห์วงจร เพื่อให้ได้เนตลิสต์ในระดับเกตออกมา เครื่องมือที่ใช้ในการสังเคราะห์โค้ดภาษา HDL (HDL Synthesis Tool) ตัวอย่างเช่น ซอฟต์แวร์ของบริษัทเมนเดอร์กราฟฟิกส์ (Mentor Graphics) ที่มีชื่อว่า “ลีโอนาร์โด สเปกตรัม” (Leonardo Spectrum) ซึ่งสามารถใช้ในการสังเคราะห์วงจรให้เป็นเอสิก หรือเอฟพีจีเอ ก็ได้เพราะมีไลบรารี (library) ให้เลือกใช้ได้จากหลายบริษัทผู้ผลิต เนื่องจากเราจะสังเคราะห์วงจรสำหรับใช้งานกับเอฟพีจีเอ ของ Xilinx ดังนั้นไลบรารีที่ใช้ก็จะเป็นของ Xilinx เช่น ไลบรารีของตระกูล Virtex II Pro เป็นต้น

3.5.4 การจำลองการทำงานในระดับพฤติกรรม Timing simulation

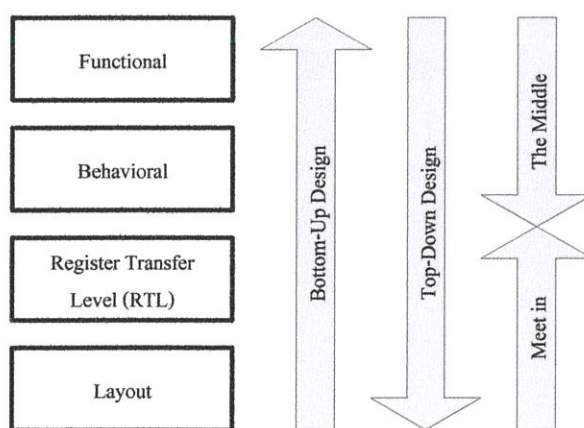
การทำงานจำลองการทำงานในระดับพฤติกรรม หรือหลังจากการสังเคราะห์วงจร แต่ยังมีได้ผ่านขั้นตอนการวางและเชื่อมเส้นทางนั้น จะใช้ในการตรวจสอบความถูกต้องทางลอจิกของวงจรเท่านั้น แต่ยังไม่มียละเอียดที่แน่นอนเกี่ยวกับเรื่องของเวลา (Timing) มากนัก เช่น ความล่าช้าของสัญญาณไฟฟ้าตามเส้นทางต่างๆ ภายในวงจร ระยะเวลาของสัญญาณเวลาน้อยที่สุดเท่าที่จะสามารถใช้กับวงจรได้ เป็นต้น เพราะจะต้องให้ผ่านขั้นตอนการวางและเชื่อมเส้นทางก่อน ดังนั้น การทำการจำลองการทำงานเพื่อตรวจสอบการทำงานของวงจร ในอุปกรณ์เอพฟิเจเอโดยคำนึงถึงเรื่องไทมมิ่ง (Timing) จะกระทำเป็นขั้นตอนสุดท้าย

3.5.5 การวางและเชื่อมเส้นทาง (Place & Route)

ผลที่ได้จากการสังเคราะห์นั้นจะเป็นเน็ตลิสต์ในระดับเกต โดยสามารถนำไปผ่านขั้นตอนการแปลงให้เป็นลอจิกบล็อก และรวมถึงการวางและเชื่อมเส้นทาง (Place & Route) ภายในอุปกรณ์เอพฟิเจเอ โดยใช้ซอฟต์แวร์ Xilinx ISE ของ Xilinx ก็ได้

3.6 การออกแบบจากบนลงล่างและการออกแบบจากล่างขึ้นบน

ในการออกแบบวงจรด้วยเทคโนโลยี ASIC หรือ FPGA มีวิธีการและกระบวนการออกแบบอยู่ 2 วิธีคือ การออกแบบจากบนลงล่าง (Top-Down Design) และการออกแบบจากล่างขึ้นบน (Bottom-Up Design) ดังรูปที่ 3.10 ซึ่งทั้งสองกระบวนการมีข้อดีและข้อเสียแตกต่างกัน



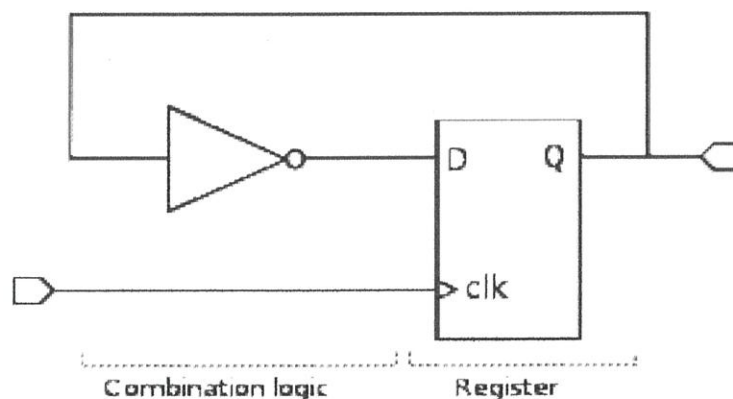
รูปที่ 3.10 กระบวนการออกแบบ (Design Methodology)

การออกแบบจากบนลงล่าง (Top-Down Design) จะมองการออกแบบจากระดับบนสุดของวงจร หรือระบบที่ต้องการออกแบบ นั่นคือ ระดับของฟังก์ชัน (Functional level) การทำงานของวงจรที่ต้องการออกแบบ และลงลึกในรายละเอียดระดับล่างของฟังก์ชัน ในระดับ

พฤติกรรม (Behavioral level) การทำงานของแต่ละฟังก์ชัน และออกแบบรายละเอียดของแต่ละฟังก์ชัน ในระดับรีจิสเตอร์ (RTL) ของแต่ละฟังก์ชัน และค่อยวิเคราะห์หาลึกลงไป ในรายละเอียดระดับลอจิก (Logic level) และระดับสุดท้าย ซึ่งเป็นระดับล่างที่สุดของการออกแบบ คือระดับผังภูมิวงจร (Layout) ซึ่งผู้ออกแบบจะไม่ทราบรายละเอียดของวงจรที่ออกแบบ จนกว่าจะถึงระดับสุดท้ายของการออกแบบ ซึ่งจะแตกต่างจากการออกแบบจากล่างขึ้นบน (Bottom-Up Design) ที่จะเริ่มออกแบบจากระดับล่างสุด และนำแต่ละส่วนประกอบกัน ขึ้นเป็นบล็อกในระดับบน เพื่อให้วงจรสามารถทำหน้าที่ตามฟังก์ชันนั้นๆ ได้ กระบวนการออกแบบลักษณะที่ผู้ออกแบบจะทราบรายละเอียดของวงจรทั้งหมด แต่ข้อเสียของกระบวนการนี้ คือผู้ออกแบบจะต้องเลือกเทคโนโลยีใดๆ ก่อนนำไปออกแบบวงจร ซึ่งถ้ามีการเปลี่ยนเทคโนโลยี จะต้องทำการออกแบบวงจรใหม่ แต่ในทางกลับกันผู้ออกแบบที่ใช้กระบวนการออกแบบจากบนลงล่าง จะไม่ยึดติดกับเทคโนโลยี ในการออกแบบวงจร เพราะการออกแบบลักษณะนี้วงจรจะอยู่ในรูปแบบของโมเดลภาษา HDL ซึ่งถ้าต้องการใช้เทคโนโลยีอะไร ก็เพียงนำไลบรารีของเทคโนโลยีมาสังเคราะห์ เป็นโมเดลของวงจรในระดับล่างต่อไปเท่านั้น นี่คือการแตกต่างระหว่างกระบวนการออกแบบ จากบนลงล่างกับกระบวนการออกแบบจากล่างขึ้นบน

3.7 RTL (Register Transfer Level)

RTL คือการเขียนบรรยายพฤติกรรมของวงจรในระดับของรีจิสเตอร์และวงจรคอมบิเนชัน (Register and Combination) [13] การออกแบบในระดับนี้ นักออกแบบต้องนึกถึงรูปแบบการประมวลผล, ไทม์มิ่งและการเชื่อมต่อของสัญญาณระหว่าง กลุ่มวงจรคอมบิเนชันและรีจิสเตอร์ โดยที่ไม่จำเป็นต้องพิจารณาพฤติกรรมวงจรระดับลอจิก หรือระดับทรานซิสเตอร์ รูปที่ 3.11 แสดงบล็อกไดอะแกรมของโครงสร้างของ RTL



รูปที่ 3.11 ตัวอย่าง Register Transfer Level บล็อกไดอะแกรม

ในการออกแบบวงจรดิจิทัลด้วย HDL ส่วนใหญ่ จะทำการเขียนบรรยายพฤติกรรมวงจรในระดับ RTL เนื่องจากมีความสะดวกในการออกแบบ และให้วงจรที่มีประสิทธิภาพมากกว่าการเขียนบรรยายในระดับลอจิก ซึ่งใช้เวลาออกแบบนาน และหาข้อผิดพลาดยาก หรือในระดับพฤติกรรม ออกแบบง่ายแต่วงจรไม่ให้ผลดีที่สุด ดังนั้นการออกแบบในระดับ RTL จะมีความสะดวกในขั้นตอนของการสังเคราะห์วงจร, การวิเคราะห์เวลา (Timing Analysis) ซึ่งเป็นการวิเคราะห์แก้ปัญหาของค่าเวลาภายในวงจร รวมทั้งการแบ่งวงจรแยกออกจากกันเป็นบล็อกย่อย (Partitioning) สำหรับการสังเคราะห์ (synthesis) เพื่อให้ผลลัพธ์ที่ดีที่สุด และยังสามารถทำความเข้าใจได้ไม่ยากนักสำหรับนักออกแบบ

3.9 Kernel และ Xilkernel

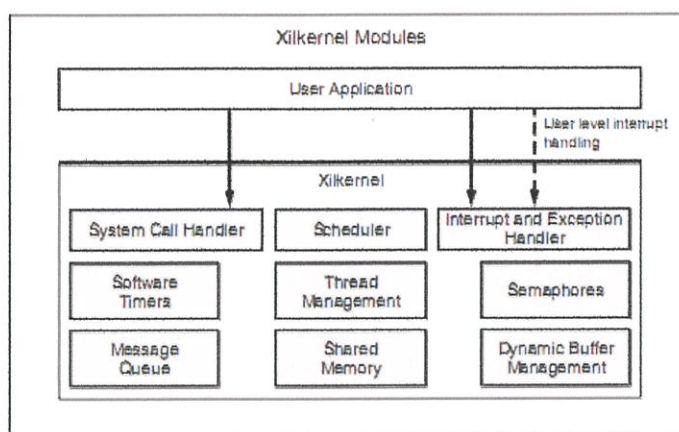
เคอร์เนล หมายถึง ส่วนประกอบหลักของระบบปฏิบัติการ ซึ่งคอยดูแลบริหารทรัพยากรของระบบ และติดต่อกับฮาร์ดแวร์และ ซอฟต์แวร์ เนื่องจากเป็นส่วนประกอบพื้นฐานของระบบปฏิบัติการ เคอร์เนล นั้นเป็นฐานล่างสุดในการติดต่อกับทรัพยากรต่างๆ เช่น หน่วยความจำ หน่วยประมวลผลกลาง และ อุปกรณ์อินพุตและเอาต์พุต

3.9.1 XILKernel

Xilkernel เป็น kernel ขนาดเล็กที่มีประสิทธิภาพและ แยกเป็นส่วนๆ ได้ถูกรวมอยู่ใน Platform Studio Framework และเป็นฟรีแวร์ที่รวมอยู่ใน Xilinx Embedded Development Kit (EDK). โดยที่มีความสามารถในการกำหนดค่าต่าง ๆ ขั้นสูง และให้ผู้ใช้สามารถที่จะเลือกการทำงานให้เหมาะสมได้ทั้ง ขนาดและฟังก์ชันการทำงาน และสนับสนุนคุณสมบัติต่าง ๆ ที่ Lightweight ต้องการ และ POSIX (Portable Operating System Interface [for Unix]) API Xilkernel สามารถทำงานได้ทั้ง MicroBlaze™ และ PowerPC 405

3.9.2 การจัดการของ Xilkernel

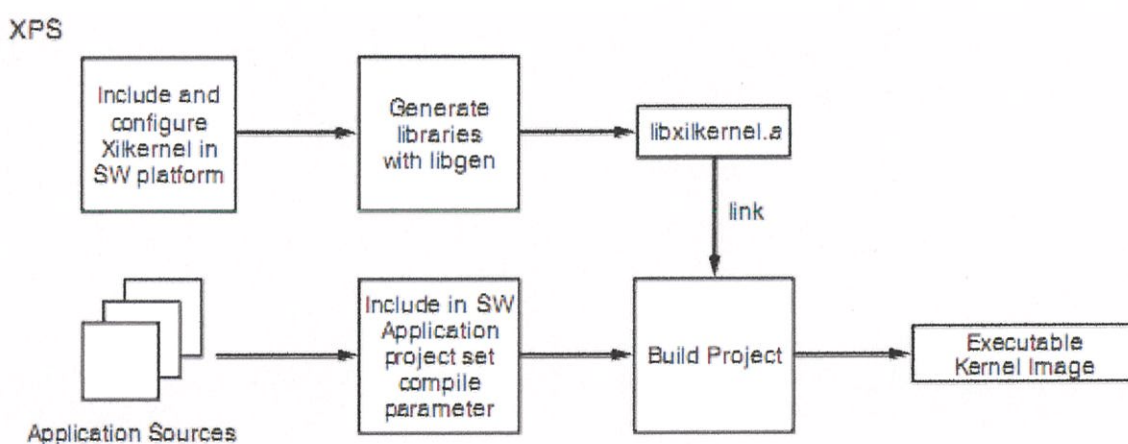
ผู้ใช้สามารถ เลือกหรือกำหนด Module ต่างๆ ใน Xilkernel ได้ตามต้องการ รูปที่ 3.12 แสดง module ต่างๆ ของ Xilkernel ที่ผู้ใช้สามารถเลือก หรือกำหนดค่าต่าง ๆ ได้โดยใช้โปรแกรม EDK



รูปที่ 3.12 โมดูลของไคเคอร์เนล (XILKernel)

3.9.3 การสร้าง Xilkernel

Xilkernel ถูกจัดรวบรวมให้อยู่ในรูปแบบ library ของ ฟังก์ชัน kernel ดังนั้นจึงนำไปสู่การใช้งานที่ง่าย ในการสร้าง Xilkernel ผู้ใช้ต้องเรียกการใช้งาน (include) Xilkernel ในระบบ software จากนั้นจึงต้องกำหนดค่าการใช้งานให้เหมาะสม และส่งงาน Libgen เพื่อสร้าง library ของ Xilkernel ซอสโค้ดของโปรแกรมประยุกต์สามารถแก้ไขและพัฒนาแยกออกจากกัน และเมื่อได้ทำการพัฒนาโปรแกรมประยุกต์เรียบร้อยแล้ว ขั้นตอนต่อไปเป็นการสร้าง Kernel image ผู้ใช้จะต้องเชื่อมโยงกับ library ของ Xilkernel ขั้นตอนการสร้างไคเคอร์เนลแสดงในรูปที่ 3.13



X10225

รูปที่ 3.13 ขั้นตอนการสร้างไคเคอร์เนล (XILKernel)

3.9.4 Xilkernel API

การจัดการ Thread Xilkernel สนับสนุนคำสั่งพื้นฐาน POSIX การสร้าง thread และการจัดการสามารถใช้งานแบบเดียวกันกับ POSIX ได้

Function ที่สำคัญ

`int pthread_create(pthread_t thread, pthread_attr_t* attr, void (*start_func) (void*), void* param)`

Parameters	thread คำตำแหน่งที่เก็บตัวบ่งชี้ thread ที่ได้สร้าง attr คือตัวชี้ของ โครงสร้างคุณลักษณะในการสร้าง thread start_func ชื่อ function ที่ให้ thread นี้ทำงาน param คือตัวชี้ argument ของ function thread
Returns	คืนค่า 0 ถ้าหากไม่มีข้อผิดพลาด คืนค่า -1 ถ้า thread อ้างอิงผิดตำแหน่ง คืนค่า EINVAL ถ้า attr อ้างอิง attributes ผิด คืนค่า EAGAIN ถ้า ไม่สามารถสร้าง thread ได้
Description	pthread_create() ใช้สำหรับสร้าง thread ใหม่ ซึ่งสามารถควบคุมค่าต่างๆ ในช่วงการสร้าง thread ได้โดยส่งผ่าน ค่าต่าง ๆ มายัง function
Includes	pthread.h

`int pthread_attr_init(pthread_attr_t* attr)`

Parameters	attr คือตัวชี้โครงสร้างคุณสมบัติ ที่จะกำหนดค่าเริ่มต้น
Returns	คืนค่า 0 ถ้าไม่มีข้อผิดพลาด คืนค่า 1 ถ้ามีข้อผิดพลาด คืนค่า EINVAL ถ้ามีข้อผิดพลาดเกี่ยวกับ attr
Description	Function นี้ใช้สำหรับ กำหนดค่าเริ่มต้น ของ thread
Includes	pthread.h

`sys_thread_new(xemacif_input_thread, netif, DEFAULT_THREAD_PRIO);`

Parameters	xemacif_input_thread ชื่อเทรตที่จะสร้าง netif ชื่อฮาร์ดแวร์ที่ใช้งานในเทรต DEFAULT_THREAD_PRIO ระดับความสำคัญของเทรต
Returns	-
Description	Function นี้ใช้สำหรับสร้าง thread ใหม่ของ เน็ตเวอร์เทรต
Includes	lwip.h

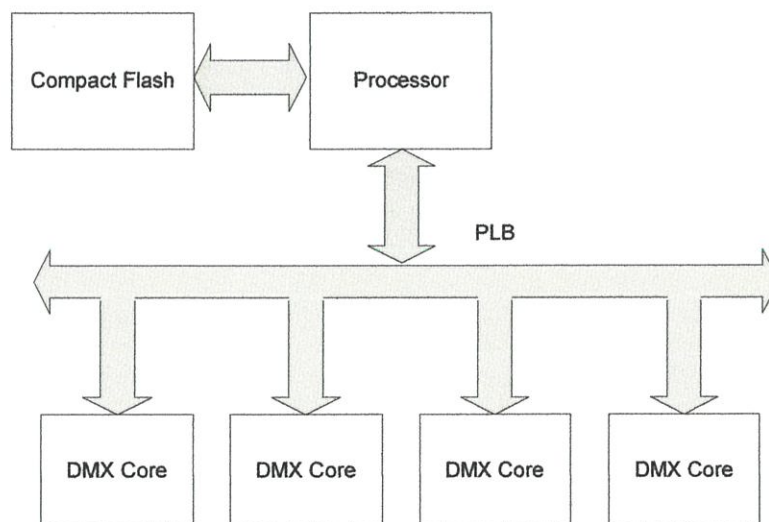
3.10 บทสรุป

ในบทนี้ได้บรรยายถึงประวัติความเป็นมา, สถาปัตยกรรม, โครงสร้างและองค์ประกอบต่างๆ ของ FPGA และรูปแบบชนิดต่างๆ ที่ใช้ในการออกแบบ, ลำดับขั้นตอนในการออกแบบ, ภาษามอดูลการออกแบบทางฮาร์ดแวร์ โดยที่กล่าวมาทั้งหมดนั้นจะพื้นฐานความเข้าใจในการประยุกต์ใช้งาน FPGA ให้ได้ประสิทธิภาพสูงสุด แต่ที่สำคัญคือประสบการณ์ในการแก้ปัญหาทางเทคนิคต่างๆ

บทที่ 4

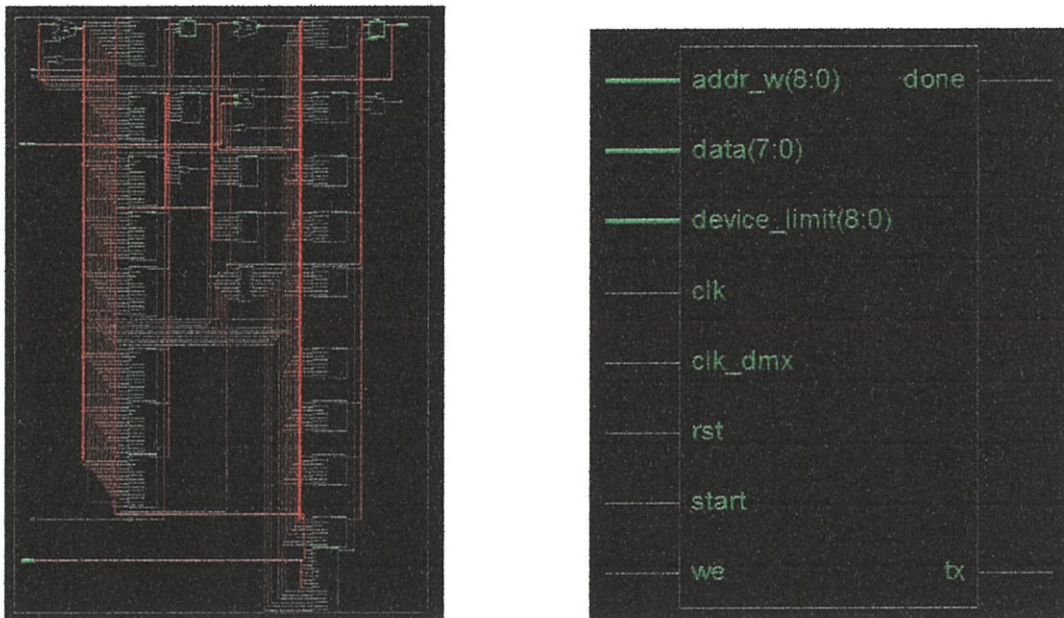
การออกแบบตัวควบคุมดีเอ็มเอ็กซ์ 512 โดยใช้เอฟพีจีเอ

วิทยานิพนธ์ฉบับนี้ นำเสนอวิธีการสร้างตัวควบคุม DMX512 แบบหลายช่องสัญญาณ โดยใช้ FPGA และ Xilinx ซึ่ง เป็นวิธีที่ทำให้สามารถควบคุมอุปกรณ์ที่รองรับโปรโตคอล DMX512 ได้หลายช่องสัญญาณและมีค่าหน่วยเวลาระหว่างช่องสัญญาณน้อย เพื่อรองรับการพัฒนาที่เพิ่มขึ้นอย่างรวดเร็วของอุปกรณ์ส่องสว่างหรือตกแต่งต่าง ๆ ที่มีความต้องการใช้ช่องสัญญาณที่เพิ่มมากขึ้นโดยไม่ใช้ Computer การออกแบบได้ออกแบบให้มามีการทำงาน 2 แบบ คือแบบทำงานด้วยตัวเองและแบบรับข้อมูลจากคอมพิวเตอร์ซึ่งแบบ ทำงานด้วยตัวเองจะใช้วิธีการเก็บค่าข้อมูลที่แสดงไว้ที่หน่วยความจำแบบแฟรช (Flash) และตัวโปรเซสเซอร์จะอ่านข้อมูลแล้วนำไปเขียนที่หน่วยความจำ (Memory) ของวงจรสร้างสัญญาณดีเอ็มเอ็กซ์ (DMX512 Creator Core) แต่ละตัว (แสดงดังรูปที่ 4.1) เพื่อให้ส่งออกไปยังอุปกรณ์ต่าง ๆ ส่วนในแบบรับข้อมูลจากคอมพิวเตอร์นั้น อุปกรณ์จะถูกควบคุมผ่านทาง TCP/IP โดยข้อมูลที่แสดงจะถูกส่งผ่านจากเครื่องคอมพิวเตอร์ และถูกนำออกไปแสดงผล โดยโปรแกรมที่ใช้ในการทดสอบนี้ถูกเขียนขึ้นด้วยภาษา C# (แสดงดังรูปที่ 4.2)

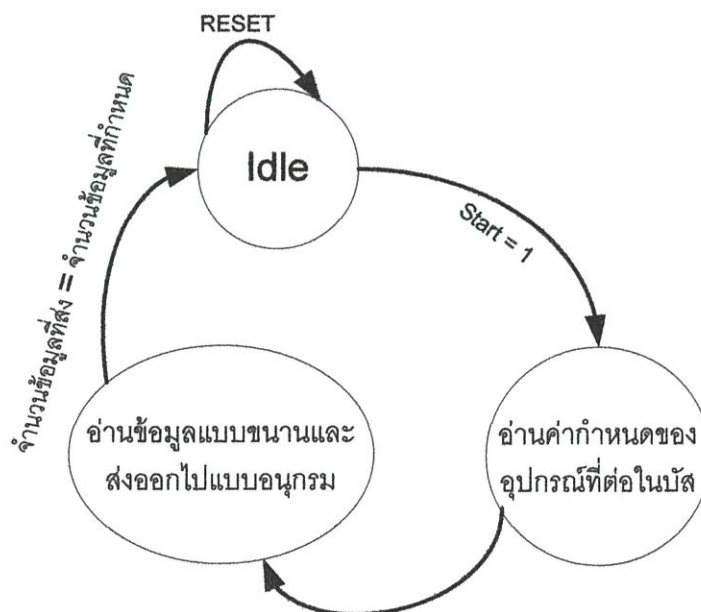


รูปที่ 4.1 การเชื่อมต่อในการทำงานแบบทำงานด้วยตัวเอง

เมื่อได้ทำการออกแบบบล็อกไดอะแกรมการทำงานของตัวสร้างสัญญาณดีเอ็มเอ็กซ์แล้ว จึงทำการออกแบบวงจรด้วยภาษาวีเอชดีแอล (VHDL) และทำการจำลองผลการทำงานด้วย โปรแกรมโมเดลซิม เพื่อทำการตรวจสอบการทำงาน จากนั้นจึงทำการสังเคราะห์ด้วยโปรแกรม ไอเอสอี (ISE) เพื่อให้ได้วงจรดิจิทัลสำหรับสร้างสัญญาณดีเอ็มเอ็กซ์ซึ่งแสดงในรูปที่ 4.4 ซึ่งการทำงานต่าง ๆ ของบล็อกตัวสร้างสัญญาณดีเอ็มเอ็กซ์จะเป็นได้ตามสเตตแมชชีนในรูปที่ 4.5



รูปที่ 4.4 บล็อกของตัวสร้างสัญญาณดีเอ็มเอ็กซ์ (DMX Creator)



รูปที่ 4.5 สเตตแมชชีนของตัวสร้างสัญญาณดีเอ็มเอ็กซ์ (DMX Creator)

ผลจากการสังเคราะห์วงจรแสดงในตารางที่ 4.1 ซึ่งทำการตั้งค่าออปติไมท์ (Optimize) แบบพื้นที่และแบบความเร็ว แสดงให้เห็นว่า การ Optimize แบบพื้นที่นั้นจะใช้ทรัพยากรของ FPGA น้อยและสามารถรองรับการทำงานของวงจรตัวสร้างสัญญาณ DMX ที่เราสร้างขึ้นได้

อย่างไรก็ตามเนื่องด้วยการ Optimize แบบพื้นที่นั้น จะมีข้อเสียคือกินพลังงานมากกว่า การ Optimize แบบความเร็ว การเลือกวิธี Optimize นั้นจึงขึ้นอยู่กับสถานะแวดล้อมการใช้งาน ต่างๆ

Resource Utilities		
	Area Optimize	Speed Optimize
Slice	192	246
FF	223	309
4 input LUT	310	341
Max.Speed	169 MHz	235 MHz

Core Performance (mW)		
	Area opt	Speed opt.
5 MHz	29.17	15.12
10 MHz	29.4	15.28
20 MHz	29.86	15.61
40 MHz	30.77	16.26
80 MHz	32.6	17.56

ตารางที่ 4.1 ทรัพยากรและค่ากำลังงานที่ใช้ใน 1 วงจรของตัวสร้างสัญญาณดิจิตอล

4.1.1 การทำงานของ DMX Creator

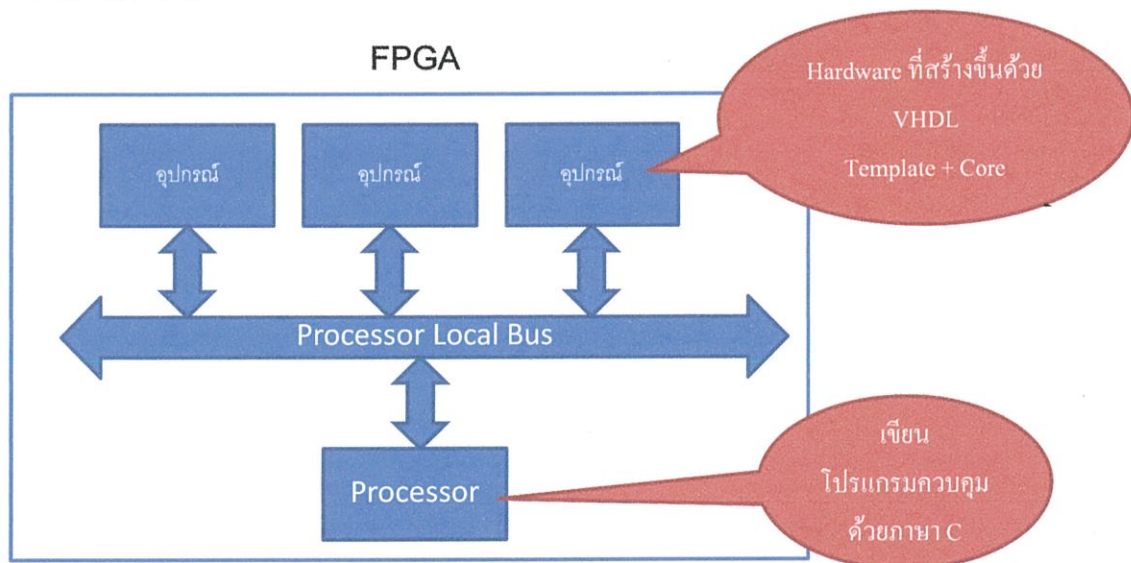
ในช่วงเริ่มต้นก่อนเริ่มการทำงาน จะต้องกำหนดอุปกรณ์ที่ต่ออยู่ในบัส โดยกำหนดที่ขา device_limit จากนั้นจึงเขียนข้อมูลลงสู่หน่วยความจำซึ่งตัวสลับหน่วยความจำ (RAM Multiplex) จะทำหน้าที่สลับสัญญาณที่จะเขียนลงสู่ชุดของหน่วยความจำว่าจะเขียนลงสู่ชุดไหน ซึ่งชุดของหน่วยความจำที่ถูกเขียนนี้จะสลับกับชุดที่ถูกอ่าน เมื่อเขียนข้อมูลลงสู่หน่วยความจำ 1 ชุดเรียบร้อยแล้ว จะทำการส่งสัญญาณ Start เพื่อสั่งให้เริ่มการทำงาน วงจรแกนกลางดิจิตอล (DMX Core) นี้จะเริ่มส่งสัญญาณที่อยู่ออกจากขา addr เพื่ออ่านข้อมูลจากหน่วยความจำที่ต่ออยู่ โดยชุดของหน่วยความจำที่ถูกอ่านนั้น จะถูกกำหนดโดย RAM Multiplex ข้อมูลที่อ่านจาก

หน่วยความจำที่ต่ออยู่จะเข้ามาทางขา data_in และจะถูกส่งออกไปยังขา TX แบบอนุกรม เมื่อข้อมูลแรกถูกส่งออกไปแล้ว DMX Core จะอ่านข้อมูลใหม่จากหน่วยความจำและส่งออกไปอีกเช่นเดียวกัน และในช่วงจังหวะนี้ซึ่งตัว DMX Core กำลังวนการทำงานอยู่ ข้อมูลชุดใหม่จะถูกเขียนเข้ามายังหน่วยความจำอีกชุดหนึ่ง และเมื่อ DMX Core ส่งข้อมูลเสร็จสิ้นแล้วจะส่งสัญญาณเสร็จสิ้นการทำงาน (Done) ออกมา เพื่อให้รู้ว่าข้อมูลถูกส่งออกไปหมดเรียบร้อยแล้ว จากนั้นจึงส่งสัญญาณ Start เข้ามาใหม่ และเริ่มการส่งอีกครั้ง ซึ่งการทำงานจะวนอยู่อย่างนี้เรื่อย ๆ

4.1.2 เอ็มเบ็ดเด็ด เดเวลอปเม้นท์ คิท

เอ็มเบ็ดเด็ด เดเวลอปเม้นท์ คิท (Embedded Development Kit) หรือ อีดีเค (EDK) เป็นซอฟต์แวร์ของไซลิงค์ ที่ใช้สำหรับสร้างโปรเซสเซอร์ภายในเอฟพีจีเอ ซึ่งโปรเซสเซอร์ที่ได้สร้างขึ้นจะพัฒนาโปรแกรมด้วยภาษาซี โดยตัว EDK เองจะมีคอมไพเลอร์สำหรับคอมไพล์ไว้ให้ด้วย

จุดเด่นของโปรแกรม EDK คือ ผู้ใช้สามารถสร้างวงจรดิจิทัลตามความต้องการซึ่งเป็นการออกแบบทางฮาร์ดแวร์ด้วยภาษา VHDL หรือ Verilog จากนั้นสามารถที่จะนำมาเชื่อมต่อกับโปรเซสเซอร์ที่ได้สร้างขึ้นด้วยโปรแกรม EDK แล้วควบคุมการทำงานต่าง ๆ ด้วยการเขียนโปรแกรมเป็นภาษาซี



รูปที่ 4.6 ลักษณะโครงสร้างของโปรเซสเซอร์และอุปกรณ์ต่อพ่วงที่สร้างด้วยโปรแกรม EDK

จากรูปที่ 4.6 แสดงให้เห็นว่าโปรเซสเซอร์และอุปกรณ์ต่อพ่วงเชื่อมต่อกันผ่านทาง Processor Local Bus หรือ PLB ดังนั้นหากเราจะนำวงจรดิจิทัลที่ได้ทำการออกแบบมาเชื่อมต่อจึงต้องนำมาใส่ในเทมเพลตเพื่อให้สามารถเชื่อมต่อกับ PLB ได้

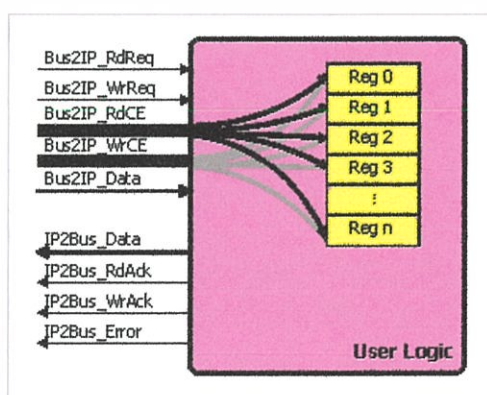
4.1.2 การออกแบบคอร์สำหรับชุดต่อพ่วง (Peripheral DMX Core)

การออกแบบ Peripheral DMX Core นั้นจะทำการออกแบบโดยใช้แม่แบบ (Template) ของ Xilinx ช่วยในการออกแบบ เพื่อที่จะสามารถนำไปเชื่อมต่อกับบัสของ PLB ได้ เพื่อที่จะทำให้สามารถรับคำสั่งและทำงานร่วมกับ Embedded Processor โดยโปรแกรม EDK จะสร้างแม่แบบที่เรียกว่า PLB Interface ขึ้นมาให้ และตัว Template นี้จะนำไปเชื่อมต่อกับ PLB บัส เพื่อที่จะได้สื่อสารกับ Processor อีกทีหนึ่ง โดยที่เราไม่จำเป็นต้องรู้ Timing ต่างๆ ของ PLB ก็ได้ซึ่งแม่แบบนี้จะจัดการในส่วน Timing ที่ติดต่อกับ PLB ให้เรียบร้อย

การใช้งานแม่แบบของ Xilinx นั้นสามารถที่จะกำหนดจำนวนรีจิสเตอร์ (register) ได้ว่าผู้ใช้งานนั้นมีความต้องการที่จะใช้รีจิสเตอร์กี่ตัว ซึ่งรีจิสเตอร์เหล่านี้จะเป็นตัวที่แม่แบบจะจัดการนำข้อมูลมาใส่ หรืออ่านออกไป ซึ่งในวิทยานิพนธ์ฉบับนี้ได้ทำการกำหนดให้มีการใช้งานรีจิสเตอร์ 4 ตัวซึ่งแต่ละตัวมีหน้าที่ดังแสดงในตารางที่ 4.2

Register ที่	Address	รายละเอียด
1	BaseAddress	ตั้งค่าอุปกรณ์ที่ต่ออยู่
2	BaseAddress + 4	ค่าสถานะ และ ตัวนับ Address
3	BaseAddress + 8	สั่งทำงานและรีเซ็ต
4	BaseAddress + C	เขียนข้อมูล

ตารางที่ 4.2 หน้าที่ของรีจิสเตอร์แต่ละตัว

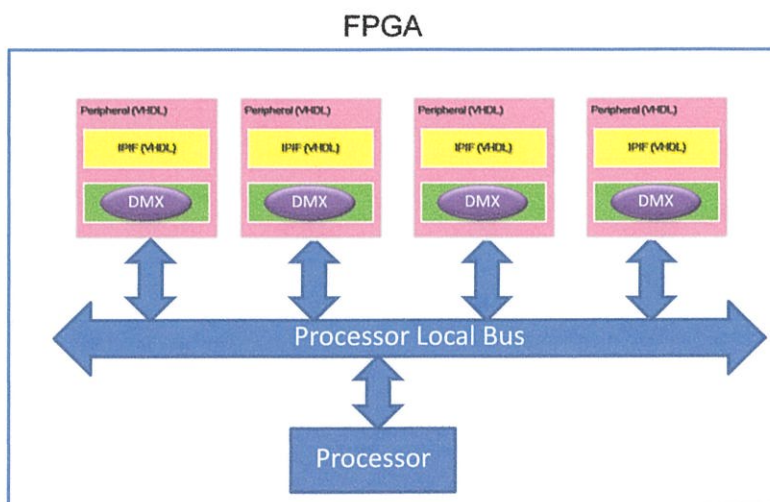


User logic software registers may take full advantage of the slave IPIF address-decoding service to generate CE decodes for all of the individual register of interest. The diagram on the left shows the simplest set of IPIC slave signals to read/write the registers.

Number of software accessible registers: 1 (1 to 4096)

รูปที่ 4.7 การตั้งค่านีจิสเตอร์ในการใช้แม่แบบของ Xilinx

รูปที่ 4.8 แสดงบล็อกไดอะแกรมของ DMX Controller ที่ได้ทำการรวม DMX Sender และ PLB interface ไว้แล้ว ทำให้ Core นี้สามารถนำไปเชื่อมต่อกับ PLB บัสได้



รูปที่ 4.8 บล็อกของวงจรดิจิทัลที่จะนำไปเชื่อมต่อกับพีแอลบีบัส (PLB BUS)

4.2 การทำงานของ DMX Controller

เมื่อเริ่มต้นการทำงาน ตัว Processor จะทำการเขียนค่า Device Limit มาที่รีจิสเตอร์ที่ 1 เพื่อตั้งค่าอุปกรณ์ที่ต่ออยู่ จากนั้นจะส่งรีเซ็ต โดยเขียนมาที่รีจิสเตอร์ที่ 3 เพื่อรีเซ็ตค่า Address Counter แล้วจึงเขียนค่าที่จะส่งออกไปให้อุปกรณ์ ที่รีจิสเตอร์ที่ 4 โดยทุกครั้งที่มีการเขียน Address counter จะเพิ่มขึ้นเรื่อย ๆ และจะเก็บค่าไว้ที่รีจิสเตอร์ที่ 2 ค่าที่ถูกเขียนลงมาที่รีจิสเตอร์นี้ จะเก็บไว้ใน memory ภายใน ซึ่งเป็น buffer ขนาด 8x512 รีจิสเตอร์ตัวที่ 4 นี้จะมีขนาด 32 บิต และข้อมูลที่จะส่งให้กับอุปกรณ์แต่ละตัวนั้นจะมีขนาด 8 บิต ดังนั้น รีจิสเตอร์ตัวที่ 4 จะเก็บข้อมูลของ อุปกรณ์ 4 ตัว โดยแบ่งเป็นรายละเอียดดังนี้

บิตที่	เก็บข้อมูลของ
0 – 7	ข้อมูลอันดับแรก
8 – 15	ข้อมูลอันดับที่สอง
16 – 23	ข้อมูลอันดับที่สาม
24 – 31	ข้อมูลอันดับที่สี่

ตารางที่ 4.3 ลำดับข้อมูลที่จะถูกเขียนลงบัฟเฟอร์

ตัวอย่างเช่น(ดูรูปที่ 4.9 ประกอบ) เมื่อเริ่มทำการเขียนข้อมูลชุดแรก Address Counter จะมีค่าเท่ากับ 0 ดังนั้น

ข้อมูลในบิตที่ 0 – 7 จะถูกเขียนลง Address ที่ 0

ข้อมูลในบิตที่ 8 – 15 จะถูกเขียนลง Address ที่ 1

ข้อมูลในบิตที่ 16 – 23 จะถูกเขียนลง Address ที่ 2

ข้อมูลในบิตที่ 24 – 31 จะถูกเขียนลง Address ที่ 3

และจากนั้น Address Counter จะถูกเพิ่มค่าเป็น 4 และถ้าหากมีการเขียนข้อมูลครั้งต่อไปมาที่ รีจิสเตอร์ที่ 4 นี้ ข้อมูลต่างๆ จะถูกเขียนไปที่ memory ตาม address ต่อไปนี้

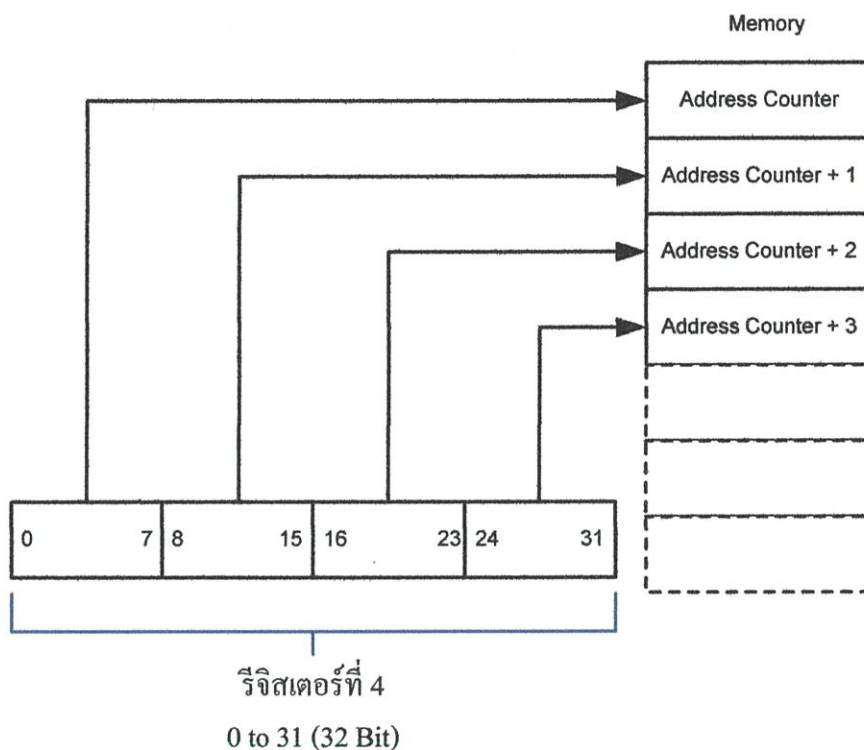
ข้อมูลในบิตที่ 0 – 7 จะถูกเขียนลง Address ที่ 4

ข้อมูลในบิตที่ 8 – 15 จะถูกเขียนลง Address ที่ 5

ข้อมูลในบิตที่ 16 – 23 จะถูกเขียนลง Address ที่ 6

ข้อมูลในบิตที่ 24 – 31 จะถูกเขียนลง Address ที่ 7

จากนั้น Address Counter จะถูกเพิ่มค่าเป็น 8 และวนการทำงานอย่างนี้จนกว่าจะได้รับคำสั่ง Start

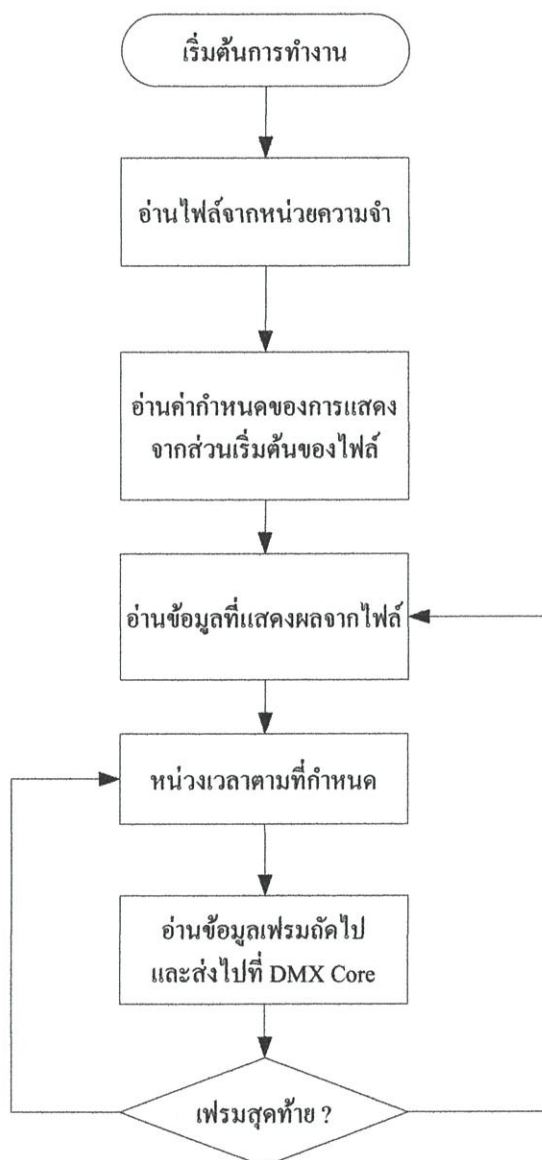


รูปที่ 4.9 ไบท์ข้อมูลที่ถูกเขียนลงหน่วยความจำ

เมื่อ Processor เขียนข้อมูลครบแล้ว จะสั่งให้เริ่มทำการส่งข้อมูลออกไปโดยเขียนคำสั่ง start มาที่รีจิสเตอร์ที่ 3 จากนั้นตัว DMX Creator ก็จะเริ่มสร้างสัญญาณเริ่มต้นก่อนการส่งและอ่านข้อมูลจากหน่วยความจำที่ต่ออยู่และทำการส่งออกไปยังอุปกรณ์ ในจังหวะเดียวกันนี้เมื่อ Processor ส่งสัญญาณ Start เพื่อให้ DMX Creator เริ่มทำงาน Processor ก็จะทำการเขียนข้อมูลใหม่ลงสู่หน่วยความจำอีกชุดหนึ่ง และรอสัญญาณเสร็จสิ้นการทำงานของ DMX Creator จึงเริ่มทำการส่งสัญญาณ start ใหม่

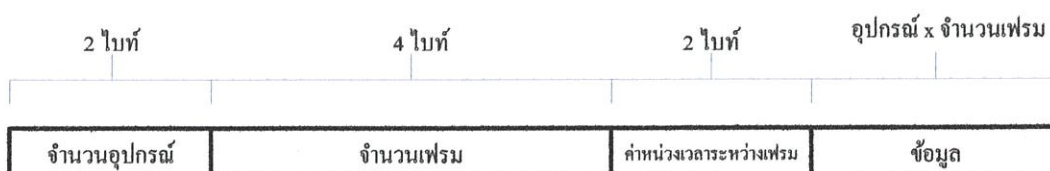
4.3 โฟลว์ชาร์ตของโปรแกรม

4.3.1 การทำงานแบบทำงานด้วยตนเอง



รูปที่ 4.10 โฟลว์ชาร์ตการทำงานแบบทำงานด้วยตัวเอง

การทำงานของโปรแกรมในโหมด Standalone นั้นจะทำงานในลักษณะอ่านข้อมูลจาก memory แล้วแสดงแบบวนลูบซึ่งแสดงในรูปที่ 4.10 โดยการทำงานจะเริ่มจากอ่านไฟล์จากหน่วยความจำ จากนั้นจึงอ่านค่ากำหนด (Header แสดงในรูปที่ 4.11) ในส่วนเริ่มต้นของไฟล์ ซึ่งค่ากำหนดต่าง ๆ นั้นจะบอกถึงจำนวนเฟรม, ค่าหน่วงเวลาระหว่างเฟรม

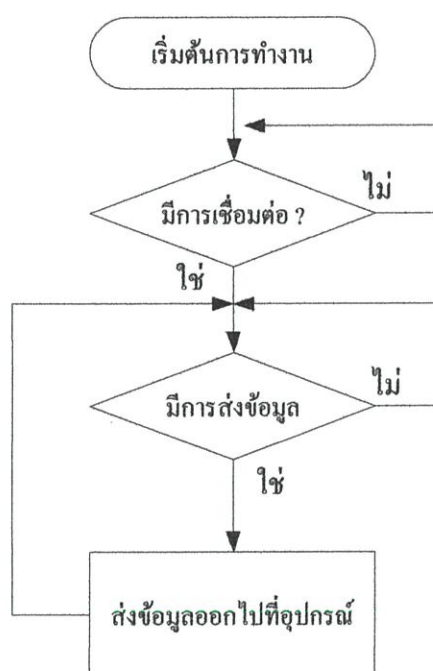


รูปที่ 4.11 รูปแบบการกำหนดค่าในไฟล์ของการทำงานแบบทำงานด้วยตัวเอง

จากรูปที่ 4.11 แสดงค่ากำหนดที่อยู่ส่วนต้นของไฟล์ โดย 2 ไบท์แรกจะเป็นจำนวนอุปกรณ์ที่ต่ออยู่ในบัส สีไบท์ถัดไปจะเป็นจำนวนเฟรมที่เล่น และถัดไปอีก 2 ไบท์ จะเป็นค่าหน่วยเวลาระหว่างเฟรม หลังจากนั้นจะเป็น ข้อมูลที่จะแสดง โดยมีขนาดเท่ากับ จำนวนอุปกรณ์คูณกับจำนวนเฟรม

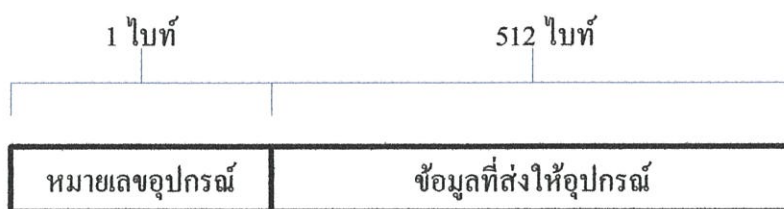
จากนั้น โปรแกรมจะอ่านข้อมูลที่จะนำไปแสดงผลเฟรมแรก และส่งออกไปยังอุปกรณ์ และจะหน่วงเวลาตามระยะเวลาที่กำหนด จากนั้นก็จะอ่านข้อมูลของเฟรมถัดไปและส่งออกไปยังอุปกรณ์อีกครั้งหนึ่ง การทำงานจะวนอยู่อย่างนี้จนกระทั่งถึงเฟรมสุดท้ายก็จะทำการย้อนกลับไปอ่านเฟรมแรกใหม่

4.3.2 การทำงานแบบรอรับคำสั่งจากคอมพิวเตอร์



รูปที่ 4.12 โฟลว์ชาร์ตการทำงานแบบรอรับข้อมูลจากคอมพิวเตอร์

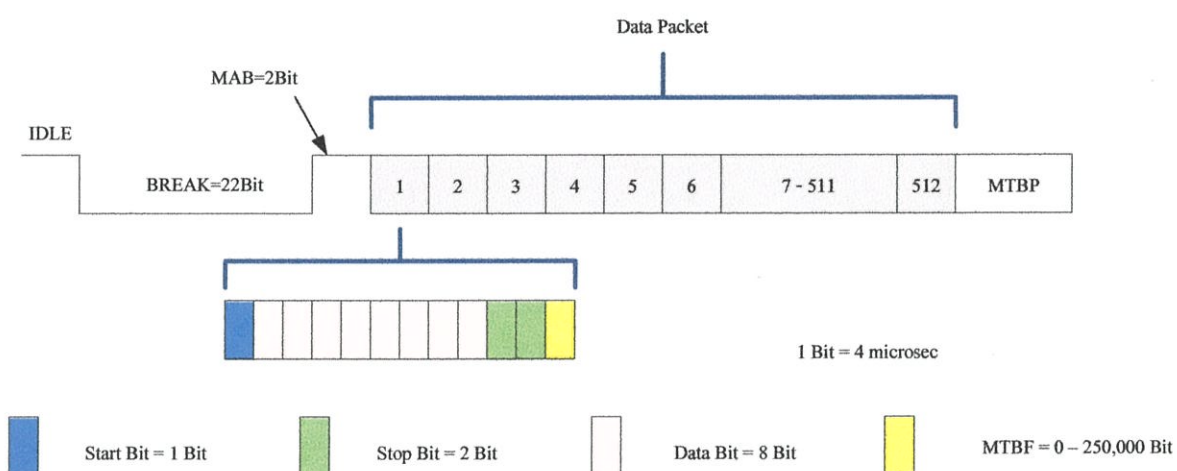
การทำงานในโหมดบายพาสแสดงในรูปที่ 4.12 การทำงานจะรอรับข้อมูลจาก Computer เพียงอย่างเดียว โดยเมื่อเริ่มการทำงาน จะรอรับการเชื่อมต่อจากคอมพิวเตอร์ทาง TCP/IP โดยเมื่อมีการเชื่อมต่อจากคอมพิวเตอร์อุปกรณ์ควบคุม DMX512 แบบฝั่งตัวจะรอรับข้อมูลที่ส่งมาจากคอมพิวเตอร์ โดยข้อมูลที่ส่งมาจะระบุถึงช่องสัญญาณที่จะทำการส่งข้อมูลให้แก่อุปกรณ์ DMX อีกด้วยซึ่งแสดงรูปแบบข้อมูลในภาพที่ 4.13 และเมื่ออุปกรณ์ควบคุม DMX512 แบบฝั่งตัวได้รับข้อมูลที่ส่งมาแล้วนั้นจะทำการเขียนข้อมูลลงไปยังช่องสัญญาณที่ระบุ



รูปที่ 4.13 รูปแบบข้อมูลที่ส่งออกจากคอมพิวเตอร์

4.4 รูปแบบสัญญาณเอาต์พุตมาตรฐาน

รูปที่ 4.14 แสดงรูปแบบของสัญญาณเคีเอ็มเอ็กซ์ 512 ซึ่งทำการส่งออกไปให้อุปกรณ์จำนวน 512ตัว โดยที่ระยะเวลาของแต่ละสัญญาณแสดงในตารางที่ 4.12



รูปที่ 4.14 รูปแบบมาตรฐานของสัญญาณเคีเอ็มเอ็กซ์

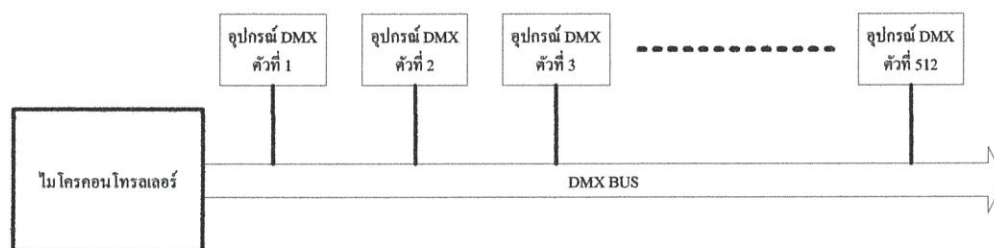
Description	ค่าต่ำสุด	ค่าปกติ	ค่าสูงสุด	หน่วย
BREAK	88	88	1,000,000	ไมโครวินาที
MAB		8		ไมโครวินาที
FRAME WIDTH		44		ไมโครวินาที
START/DATA/STOP BITS		4		ไมโครวินาที
MTBF	0	ไม่กำหนด	1,000,000	ไมโครวินาที
MTBP	0	ไม่กำหนด	1,000,000	ไมโครวินาที

ตารางที่ 4.4 ค่าระยะเวลามาตรฐานของโปรโตคอลเคีเอ็มเอ็กซ์

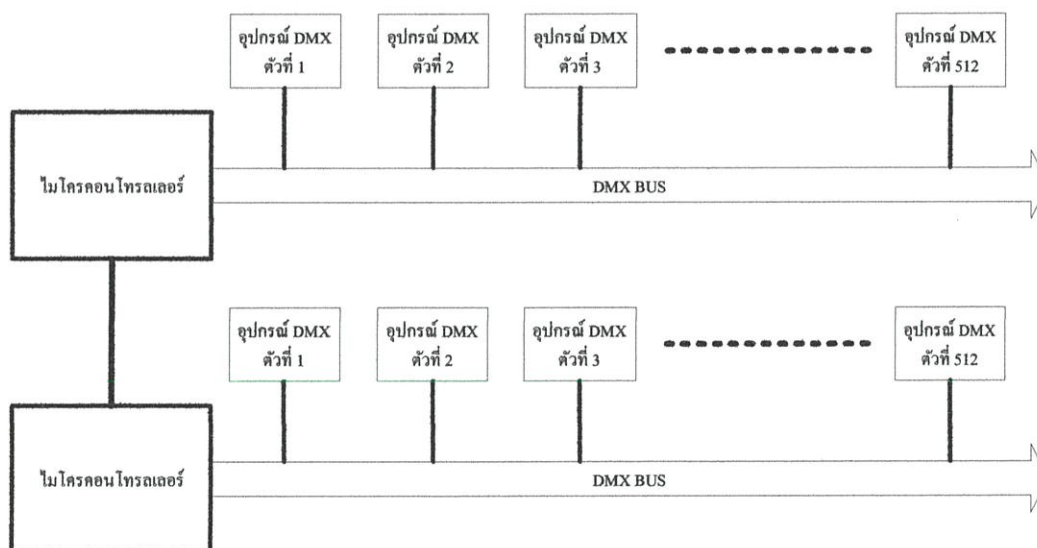
4.5 รูปแบบของตัวกำเนิดสัญญาณดีเอ็มเอ็กซ์

4.5.1 ตัวกำเนิดสัญญาณดีเอ็มเอ็กซ์โดยใช้ไมโครคอนโทรลเลอร์

จากรูป 4.15 อุปกรณ์กำเนิดสัญญาณ DMX โดยใช้ไมโครคอนโทรลเลอร์นั้นจะใช้พอร์ตอนุกรมในการสร้างสัญญาณ ซึ่งโดยทั่วไปพอร์ตอนุกรมของไมโครคอนโทรลเลอร์นั้นจะมี 1 หรือ 2 ช่องสัญญาณเท่านั้น ซึ่งในแต่ละช่องสัญญาณจะสามารถควบคุมอุปกรณ์ได้สูงสุด 512 อุปกรณ์ ดังนั้นในการเพิ่มช่องสัญญาณดีเอ็มเอ็กซ์โดยใช้ไมโครคอนโทรลเลอร์นั้นจึงจะต้องเพิ่มจำนวนไมโครคอนโทรลเลอร์ขึ้นมาและทำการเชื่อมต่อกันเพื่อให้ทำงานสัมพันธ์กัน



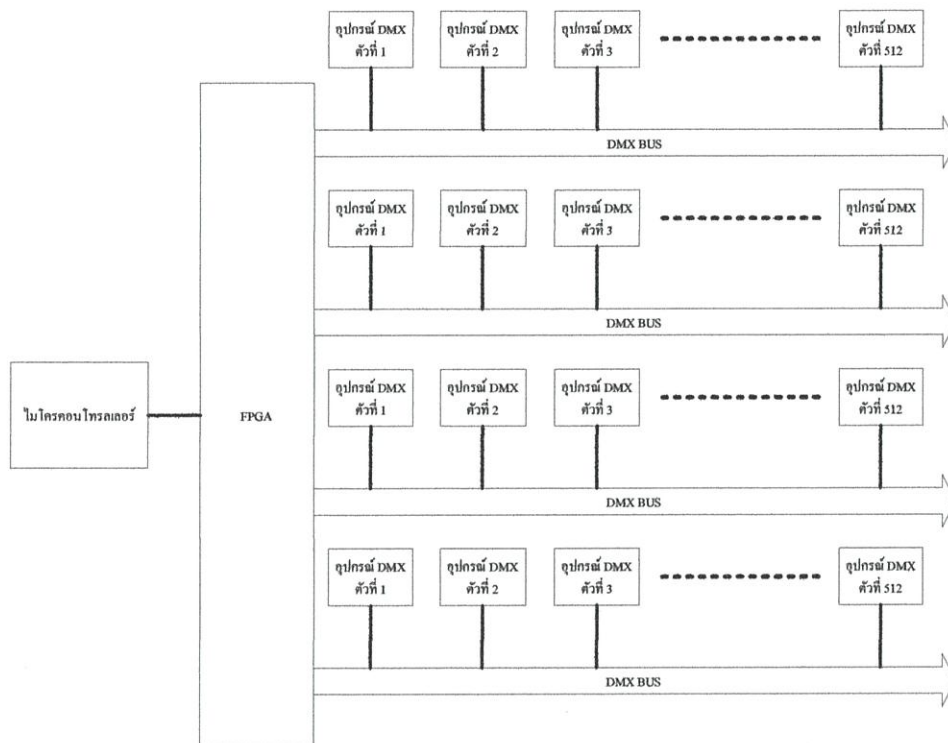
รูปที่ 4.15 การเชื่อมต่อไมโครคอนโทรลเลอร์กับบัสดีเอ็มเอ็กซ์ 1 ช่องสัญญาณ



รูปที่ 4.16 การเชื่อมต่อไมโครคอนโทรลเลอร์หลายตัวกับบัสดีเอ็มเอ็กซ์แบบหลายช่องสัญญาณ

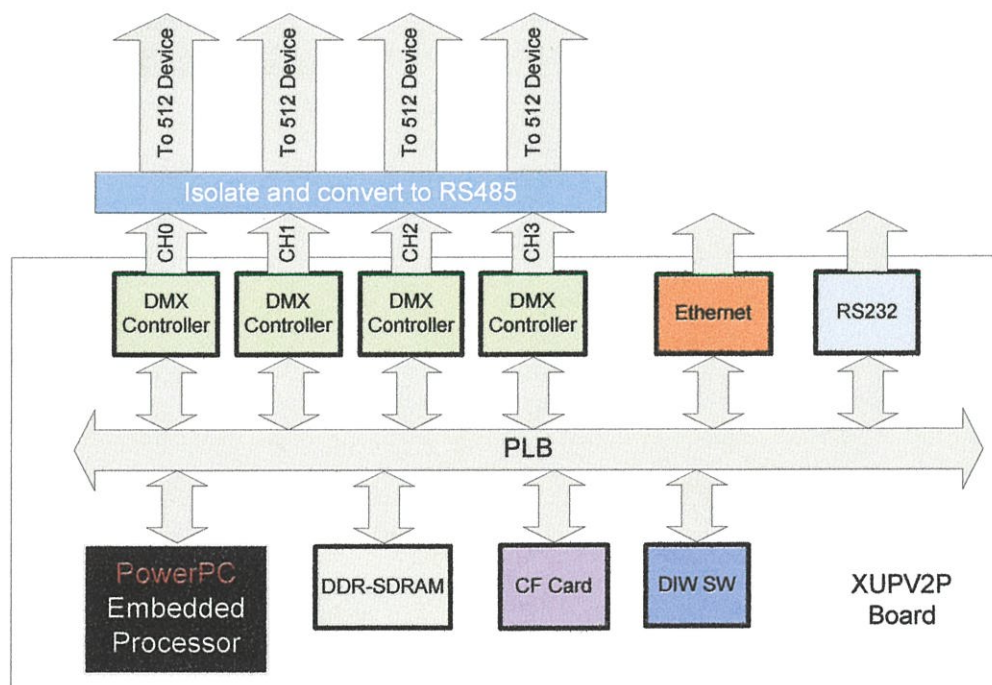
4.5.2 ตัวกำเนิดสัญญาณดีเอ็มเอ็กซ์โดยใช้เอฟพีจีเอ

ตัวกำเนิดสัญญาณดีเอ็มเอ็กซ์โดยใช้เอฟพีจีเอดังแสดงในรูปที่ 4.17 นั้น สามารถที่จะสร้างตัวกำเนิดสัญญาณหลาย ๆ ช่องสัญญาณภายในเอฟพีจีเอตัวเดียวได้ และสามารถให้ไมโครคอนโทรลเลอร์ในการควบคุมเพียงตัวเดียวซึ่งทำให้สามารถควบคุมช่องสัญญาณดีเอ็มเอ็กซ์ได้หลายช่องสัญญาณและมีประสิทธิภาพโดยใช้ไมโครคอนโทรลเลอร์เพียงตัวเดียว



รูปที่ 4.17 การเชื่อมต่อไมโครคอนโทรลเลอร์ร่วมกับเอฟพีจีเอแบบหลายช่องสัญญาณ

4.6 บล็อกไดอะแกรมของฮาร์ดแวร์



รูปที่ 4.18 บล็อกไดอะแกรมของฮาร์ดแวร์ที่ใช้ในงานวิจัย

เอ็มเบ็คเค็ดโปรเซสเซอร์ (Embedded Processor) ที่ใช้ในงานวิจัยนี้ เราได้ใช้เพาเวอร์พีซี (PowerPC) 405 ซึ่งอยู่ในตัวเอฟพีจีเอของไซลิงค์โดยใช้เอ็มเบ็คเค็ดเคเวลอปเม้นท์กิท (Embedded

Development Kit: EDK) ซึ่งเป็นซอฟต์แวร์ที่ใช้ในการพัฒนาระบบเอ็มเบดเด็ดซิสเต็มที่ใช้ในเอพพิจีเอของไซลิงค์ โดยที่เอ็มเบดเด็ดโปรเซสเซอร์ จะเชื่อมต่ออุปกรณ์ฮาร์ดแวร์ ต่าง ๆ ผ่านทางโปรเซสเซอร์โลกอลบัส (PLB) ซึ่งอุปกรณ์แต่ละชนิดจะถูกกำหนดเบสแอดเดรส (BaseAddress) ไว้ในขั้นตอนการออกแบบทางฮาร์ดแวร์ เพื่อให้สามารถใช้โปรแกรมประยุกต์ ในการควบคุมฮาร์ดแวร์ต่าง ๆ ที่ต่ออยู่ใน PLB ได้

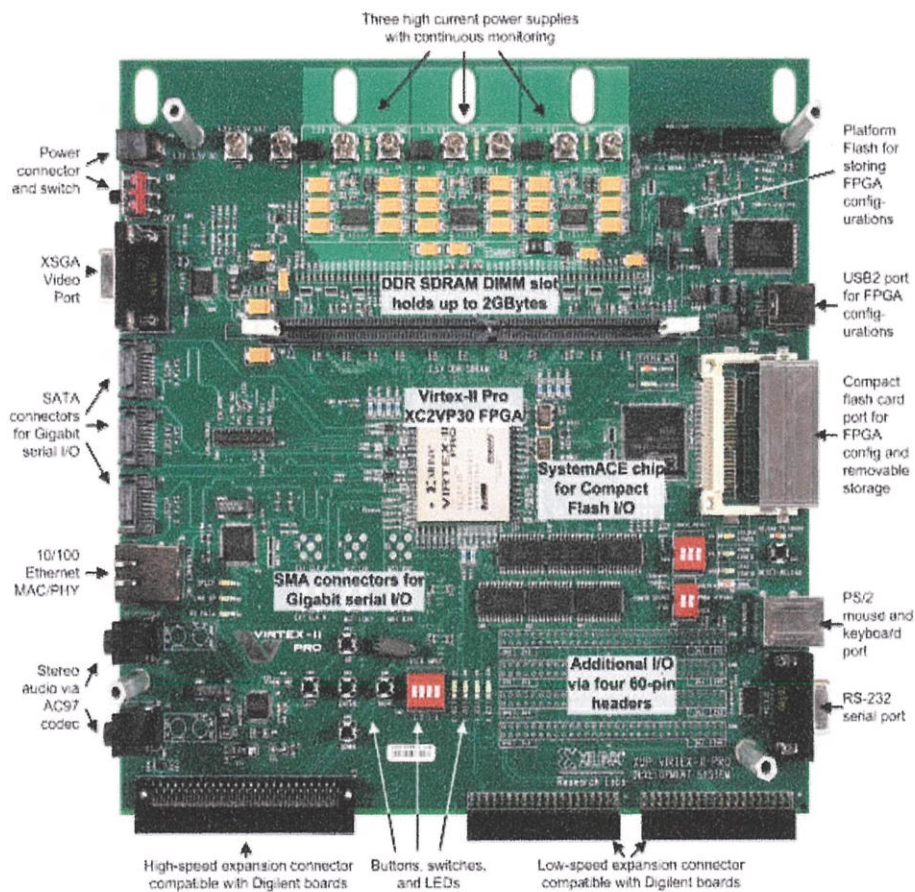
จากรูปที่ 4.18 ในขณะที่ เอ็มเบดเด็ดโปรเซสเซอร์เริ่มทำงาน Boot-loader ในตัว เอ็มเบดเด็ดโปรเซสเซอร์จะทำการโหลด Image ของ Software Application ซึ่งจะถูกเก็บอยู่ใน Compact Flash Memory มาไว้ที่ DDR-SDRAM ผ่านทาง PLB จากนั้น Embedded Processor จะรับคำสั่งจาก DDR-SDRAM โดย จะเริ่มทำการ Initial ค่าเริ่มต้นต่าง ๆ จากนั้นจะตรวจสอบโหมดการทำงานจาก DIP SW และเริ่มทำงานตามโหมดที่กำหนด โดยจะมี Port RS232 ไว้สำหรับการตรวจสอบการทำงานและข้อผิดพลาดต่าง ๆ

บทที่ 5

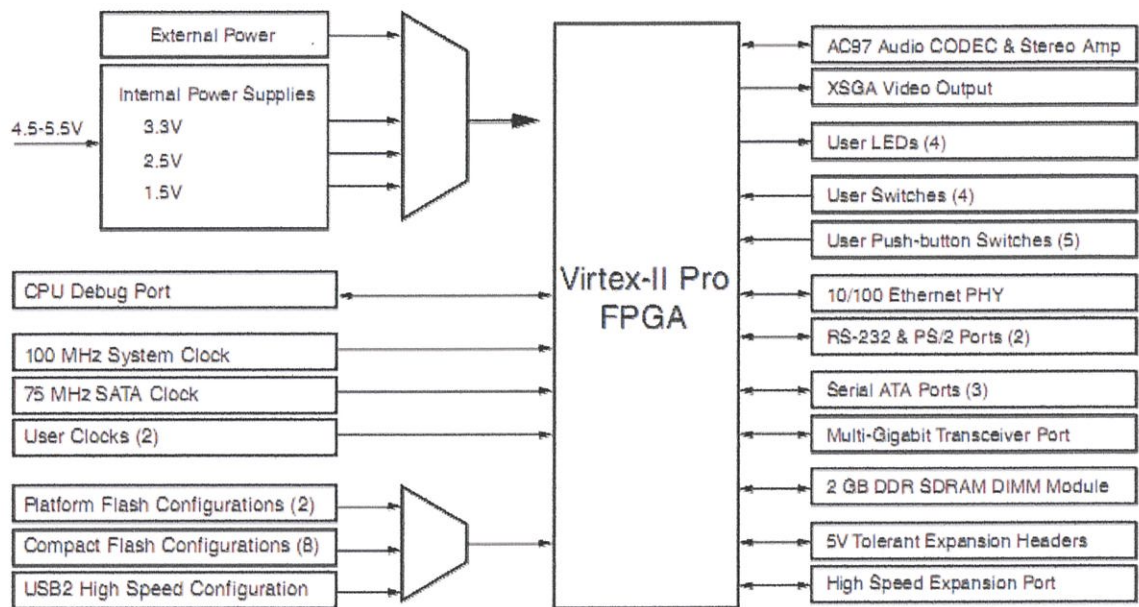
การทดลองและผลการทดลอง

ในบทนี้จะกล่าวถึงผลการทดลอง ตัวควบคุม DMX 512 แบบหลายช่องสัญญาณ ด้วย FPGA ซึ่งจะนำไปทดลองกับอุปกรณ์ DMX 512 ที่มีขายทั่วไป โดยจะทำการเปรียบเทียบสัญญาณของแต่ละช่องสัญญาณ และวัดผลการหน่วงเวลาของแต่ละช่องสัญญาณ

5.1 รายละเอียดของเครื่องมือและอุปกรณ์ที่ใช้ในการทดลอง



รูปที่ 5.1 บอร์ดเอฟพีจีเอที่ใช้ในการพัฒนา



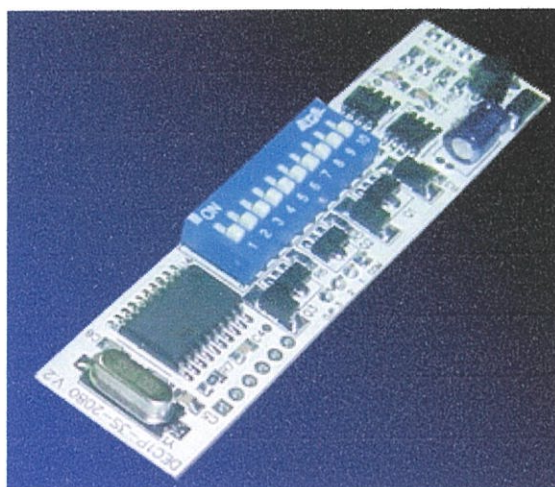
รูปที่ 5.2 บล็อกไดอะแกรมของบอร์ดทดลองที่ใช้ในงานวิจัย

5.1.1 บอร์ดทดลอง FPGA XUPV2P

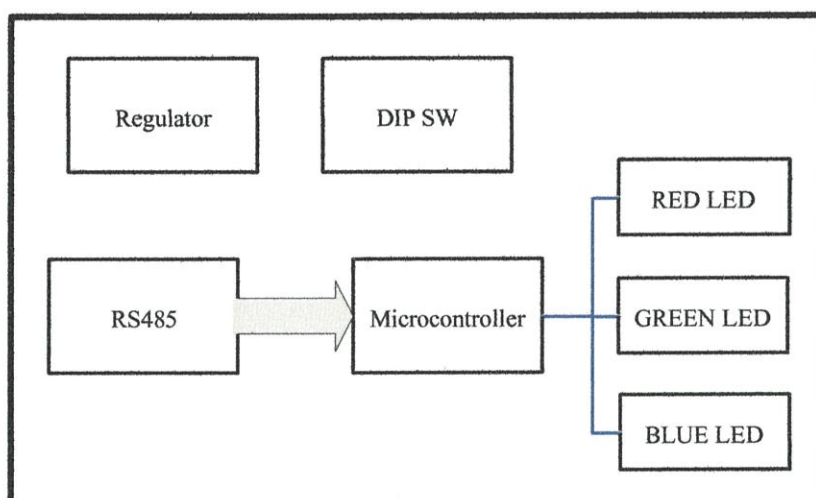
รูปที่ 5.1 เป็นบอร์ดทดลองซึ่งใช้ FPGA เบอร์ XC2VP30 ของบริษัท Xilinx โดยในตัว FPGA จะมี Embedded Processor อยู่ในตัว และมีอุปกรณ์ต่างๆ บนบอร์ดที่สามารถนำมาใช้กับการทดลองได้สะดวก เช่น คอมแพ็คแฟรชเพื่อใช้เก็บข้อมูลที่ทำการแสดงและโปรแกรมเมมโมรี, ดิจิตอลเอสดีแรม, อีเธอร์เน็ตพอร์ต, และอาร์เอส 232 ซึ่งบล็อกไดอะแกรมของบอร์ดแสดงในรูปที่ 5.2

5.1.2 อุปกรณ์รับสัญญาณ DMX512 รุ่น DEC1P-3S

อุปกรณ์ที่ใช้ทดสอบการรับสัญญาณ DMX นี้จะใช้อุปกรณ์ของบริษัท ซีทีเม็กซ์ เทคโนโลยี รุ่น DEC1P-3S ดังแสดงในรูปที่ 5.3 และบล็อกไดอะแกรมของอุปกรณ์แสดงในรูปที่ 5.4 ซึ่งเป็นอุปกรณ์ที่สามารถรับสัญญาณ DMX512 และนำข้อมูลที่ได้รับไปแสดงผล โดยในอุปกรณ์ 1 ตัว จะมีตำแหน่ง (Address) อยู่ 3 ตำแหน่ง ซึ่งเอาต์พุตทั้ง 3 ของอุปกรณ์ชนิดนี้จะมีลักษณะเป็นดิจิตอลเอาต์พุต ค่าที่ออกมาจากเอาต์พุตจะเป็นลักษณะ พัลส์ วิตซ์ โมดูเลชัน (PWM) ซึ่งจะมีความกว้างของพัลส์แตกต่างกันตามค่าที่ได้รับแต่ละตำแหน่ง (Address)



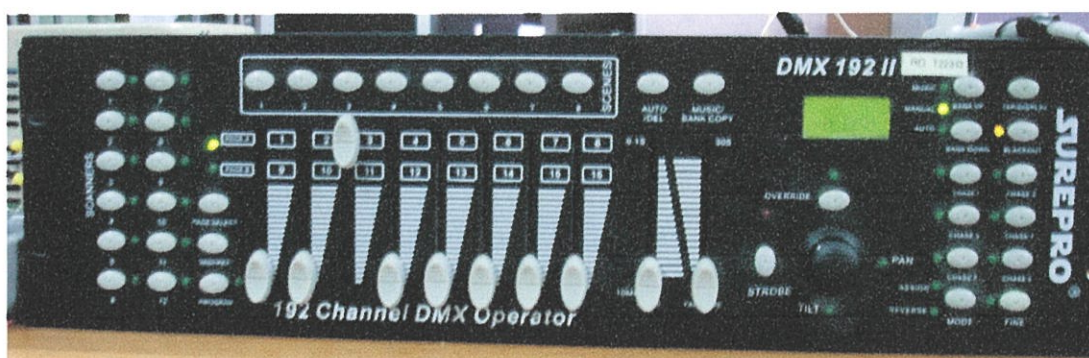
รูปที่ 5.3 อุปกรณ์รับสัญญาณดีเอ็มเอ็กซ์ รุ่น DEC1P-3S



รูปที่ 5.4 บล็อกไดอะแกรมของอุปกรณ์รับสัญญาณดีเอ็มเอ็กซ์ รุ่น DEC1P-3S

5.1.3 เครื่องสร้างสัญญาณดีเอ็มเอ็กซ์ รุ่น DMX 192 II

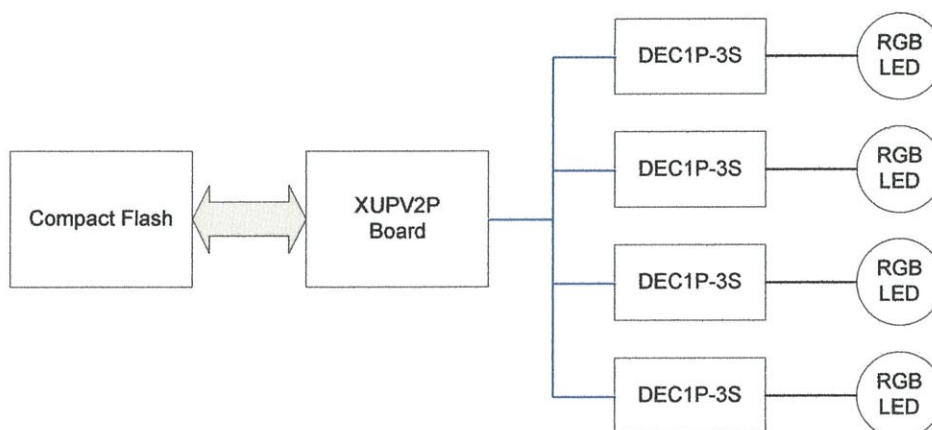
รูปที่ 5.5 แสดงเครื่องส่งสัญญาณ DMX ที่ทำมาใช้ในการทดลองโดยตัวเครื่องสามารถส่งสัญญาณควบคุมอุปกรณ์ได้สูงสุด 192 อุปกรณ์ และสามารถแสดงผลภาพได้สูงสุด 22 ภาพต่อวินาที



รูปที่ 5.5 เครื่องสร้างสัญญาณดีเอ็มเอ็กซ์ รุ่น DMX 192 II

5.2 โหมดการทำงาน

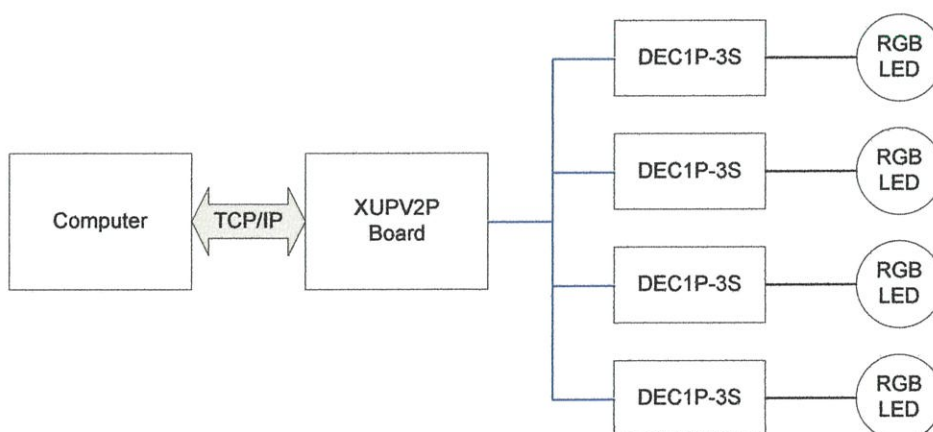
5.2.1 การทดลองในโหมดทำงานด้วยตัวเอง (Standalone)



รูปที่ 5.6 การเชื่อมต่อในการทำงานแบบทำงานด้วยตัวเอง

การทดลองแบบโหมดทำงานด้วยตัวเอง จะทำการสร้างไฟล์ขึ้นมา 4 ไฟล์ ซึ่งแต่ละไฟล์จะตั้งเป็นชื่อของช่องสัญญาณดีเอ็มเอ็กซ์ แล้วทำการบันทึกลงคอมแพ็คแฟรชเมมโมรี่ โดยรายละเอียดในไฟล์จะระบุถึงข้อมูลที่จะทำการแสดงของแต่ละช่องสัญญาณ จากนั้นเมื่อเริ่มการทำงาน ไฟล์เหล่านี้จะถูกอ่านออกมาจากคอมแพ็คแฟรช และส่งข้อมูลให้กับ DEC1P-3S ในโปรโตคอลแบบดีเอ็มเอ็กซ์ 512

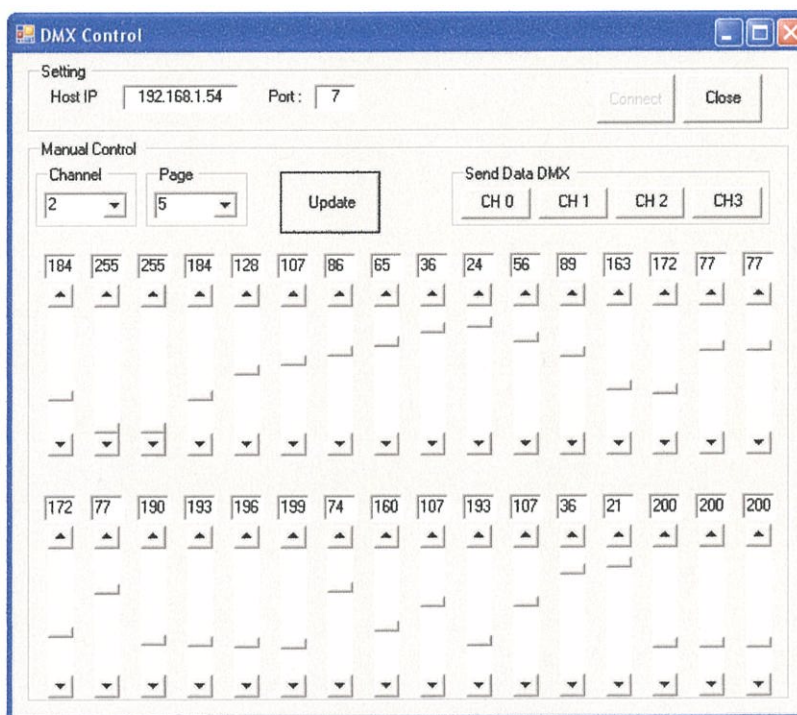
5.2.2 การทำงานในโหมดทางผ่าน (Bypass)



รูปที่ 5.7 การเชื่อมต่อในโหมดทางผ่าน

การทดลองในโหมดทางผ่านจะใช้การเชื่อมต่อกับคอมพิวเตอร์ผ่านทาง TCP/IP โปรโตคอลเพื่อนำข้อมูลจากโปรแกรมประยุกต์บนคอมพิวเตอร์ส่งต่อไปยังอุปกรณ์ดีเอ็มเอ็กซ์

โดยโปรแกรมประยุกต์ที่ใช้งานถูกเขียนด้วยโปรแกรมวิซวลซีชาร์ป (Visual C#) ดังแสดงในรูปที่ 5.8

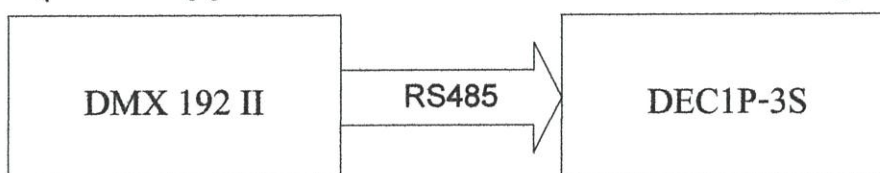


รูปที่ 5.8 โปรแกรมควบคุมดีเอ็มเอ็กซ์ที่ใช้ในการทดลอง

โปรแกรมที่ใช้นี้ ผู้ใช้งานจะทำการใส่ไอพีแอดเดรส (IP Address) ของอุปกรณ์ จากนั้นจึงทำการปรับค่าต่าง ๆ ที่จะส่งออกไปยังอุปกรณ์ด้วยสกรอร์บาร์ และกด Update

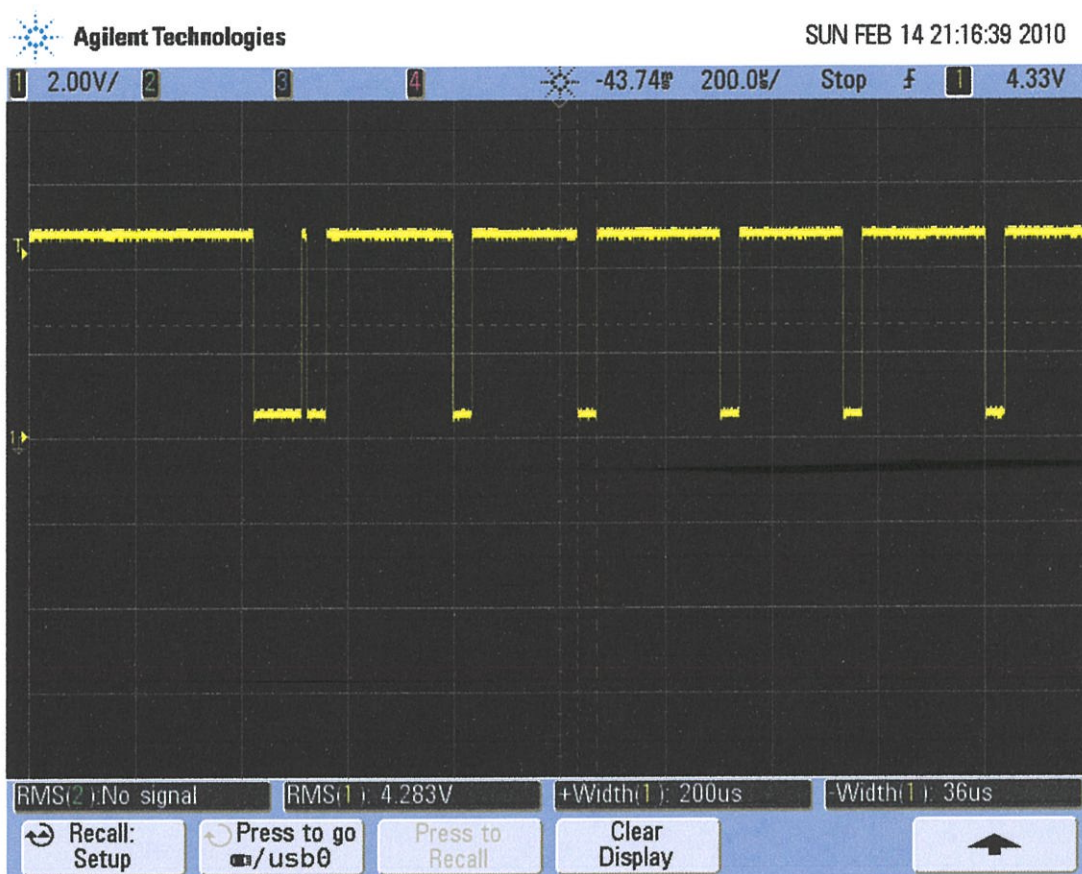
5.3 การทดลองโดยใช้เครื่องสร้างสัญญาณดีเอ็มเอ็กซ์รุ่น DMX 192 II

ทำการเชื่อมต่ออุปกรณ์เพื่อทำการวัดค่าสัญญาณเอาต์พุตที่ออกจาก DMX 192 II ซึ่งจะเชื่อมต่อกับอุปกรณ์รับสัญญาณดีเอ็มเอ็กซ์ 512 ผ่านทางอาร์เอส 485 บัส โดยแสดงในรูปที่ 5.9

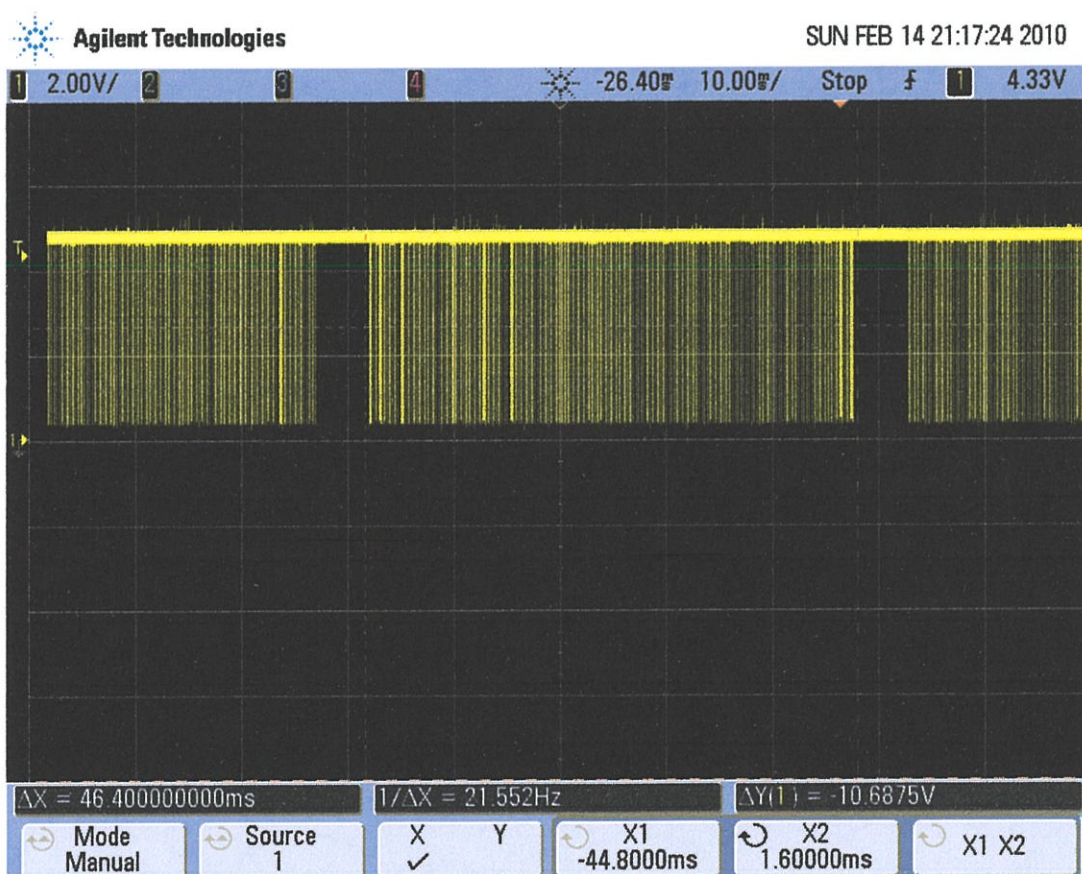


รูปที่ 5.9 การเชื่อมต่อระหว่าง DMX 192 II กับอุปกรณ์ดีเอ็มเอ็กซ์ 512

ทำการวัดสัญญาณเอาต์พุตด้วยออสซิลอสโคป แสดงในรูปที่ 5.10 และ 5.11



รูปที่ 5.10 สัญญาณดิเอ็มแอลที่ออกจาก ไมโครคอนโทรลเลอร์



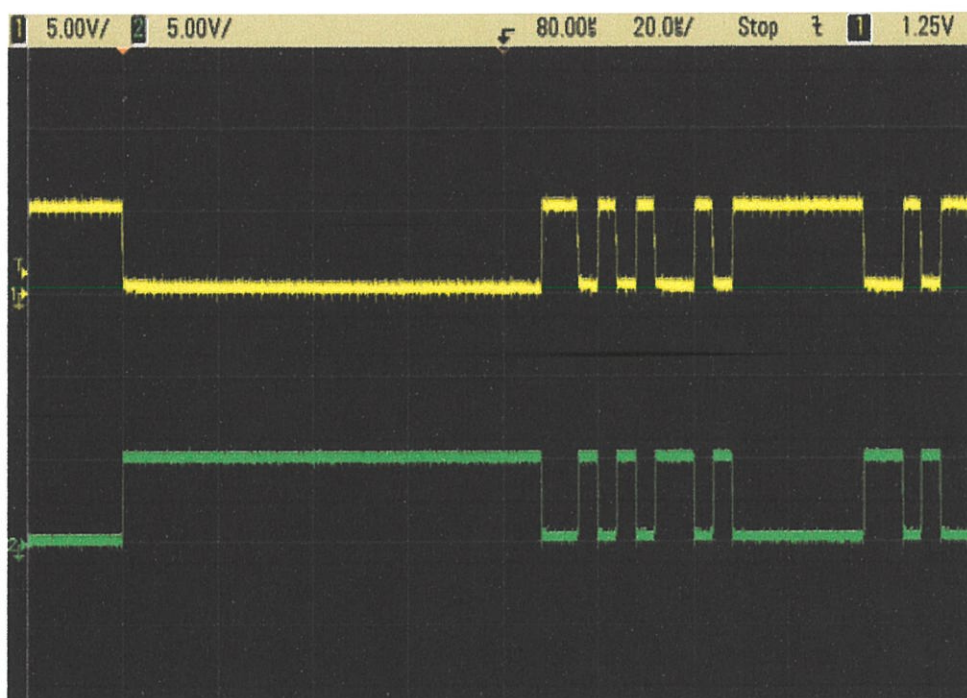
รูปที่ 5.11 สัญญาณดิเอ็มแอลที่ออกจากไมโครคอนโทรลเลอร์ ใน 1เฟรม

สัญญาณที่แสดงในรูปที่ 5.11 เป็นการแสดงผล 1 เฟรม โดยใช้ DMX 192 II ส่งข้อมูล 192 ข้อมูลให้กับอุปกรณ์เวลาที่ใช้ใน 1 เฟรมเท่ากับ 46.4 มิลลิวินาที ดังนั้นใน 1 วินาทีสามารถที่จะแสดงผลได้เท่ากับประมาณ 22 ภาพ

5.4 การทดลองโดยใช้เอฟพีจีเอ สร้างวงจรดิจิทัลเป็นตัวกำเนิดสัญญาณดีเอ็มเอ็กซ์ที่หลายช่องสัญญาณ

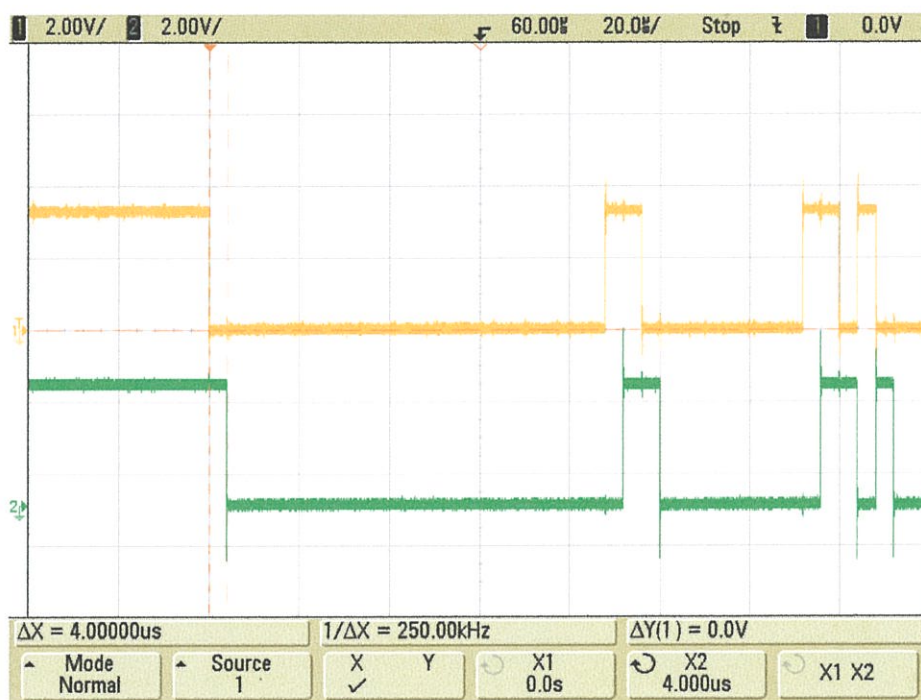
การสร้างอุปกรณ์กำเนิดสัญญาณดีเอ็มเอ็กซ์แบบหลายช่องสัญญาณนั้น สามารถใช้ไมโครคอนโทรลเลอร์หลาย ๆ ตัวมาต่อกันได้ แต่อาจมีความยุ่งยากในเรื่องของ ความสัมพันธ์ในการแสดงของไมโครคอนโทรลเลอร์แต่ละตัว หรือ การอ่านและส่งข้อมูลจากหน่วยความจำ ดังนั้นวิธีที่จะแก้ปัญหาจึงได้นำเสนอการสร้างวงจรดิจิทัลเป็นตัวกำเนิดสัญญาณดีเอ็มเอ็กซ์ลงในเอฟพีจีเอแล้วใช้ ไมโครคอนโทรลเลอร์ เพียงตัวเดียวในการควบคุม จะทำให้สามารถขยายช่องสัญญาณของดีเอ็มเอ็กซ์ได้ และยังง่ายแก่การพัฒนาโปรแกรมประยุกต์ที่ใช้ในไมโครคอนโทรลเลอร์ด้วย

5.4.1 ทำการทดลองวัดสัญญาณเอาต์พุตของเพื่อตรวจสอบ Timing

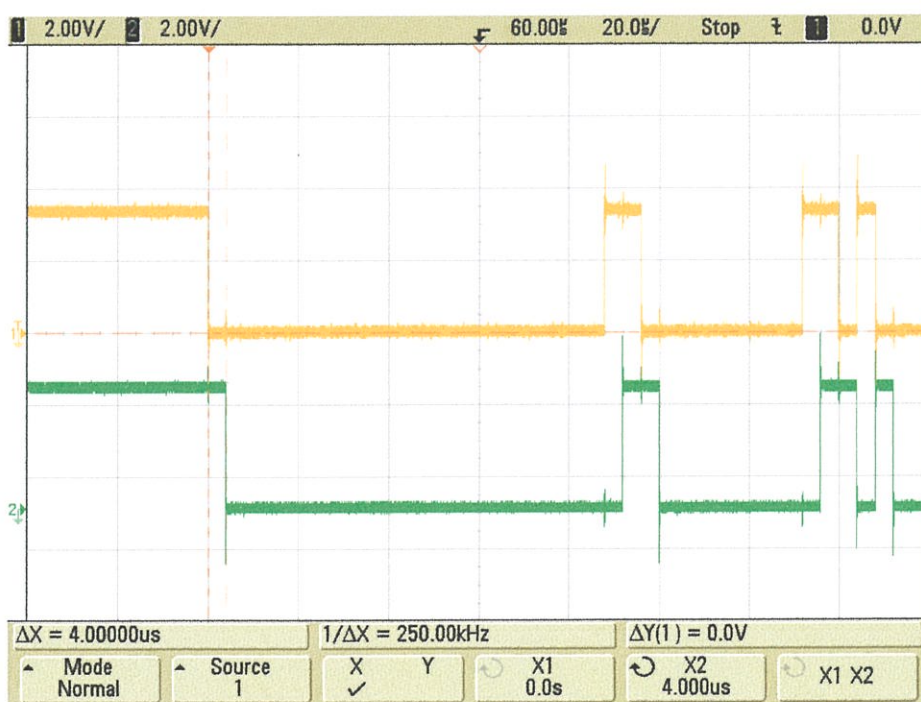


รูปที่ 5.12 สัญญาณ Timing ที่ออกจาก FPGA ผ่านการแปลงให้เป็นอาร์เอส 485

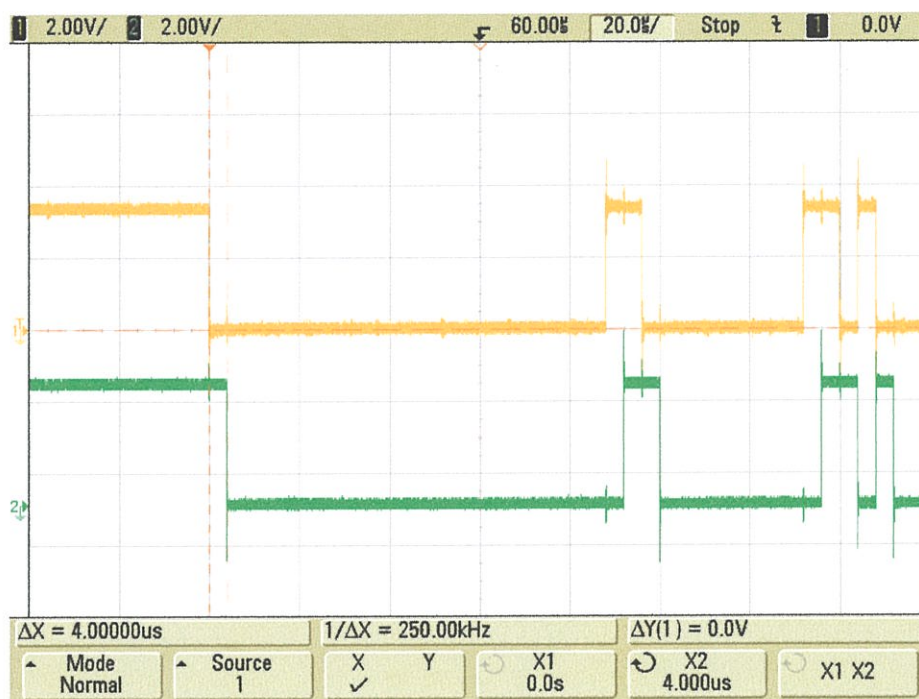
5.4.2 ทำการวัดเพื่อเปรียบเทียบค่าหน่วยเวลาของช่องสัญญาณ



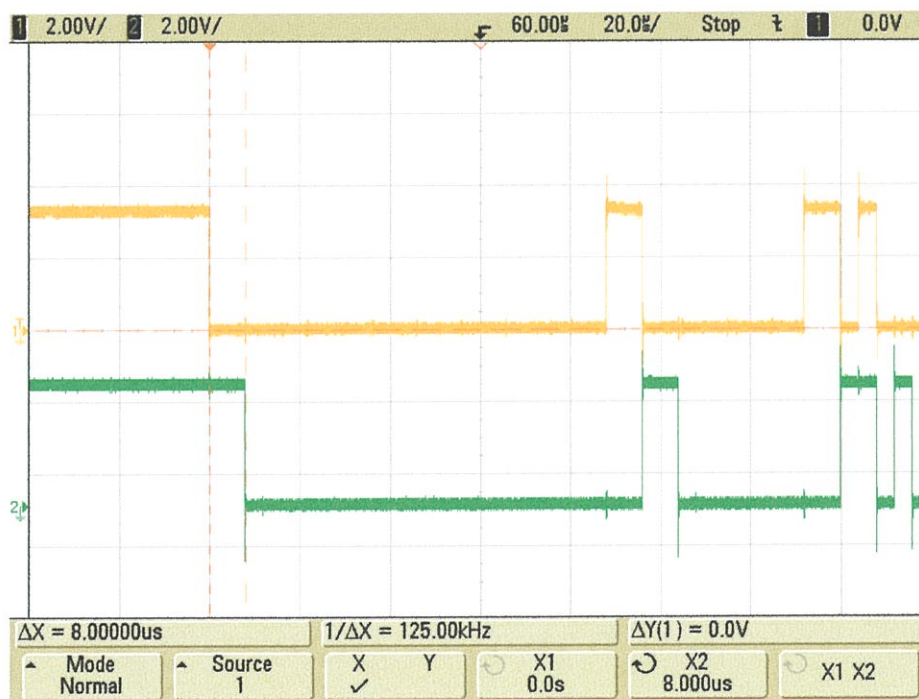
(ก)



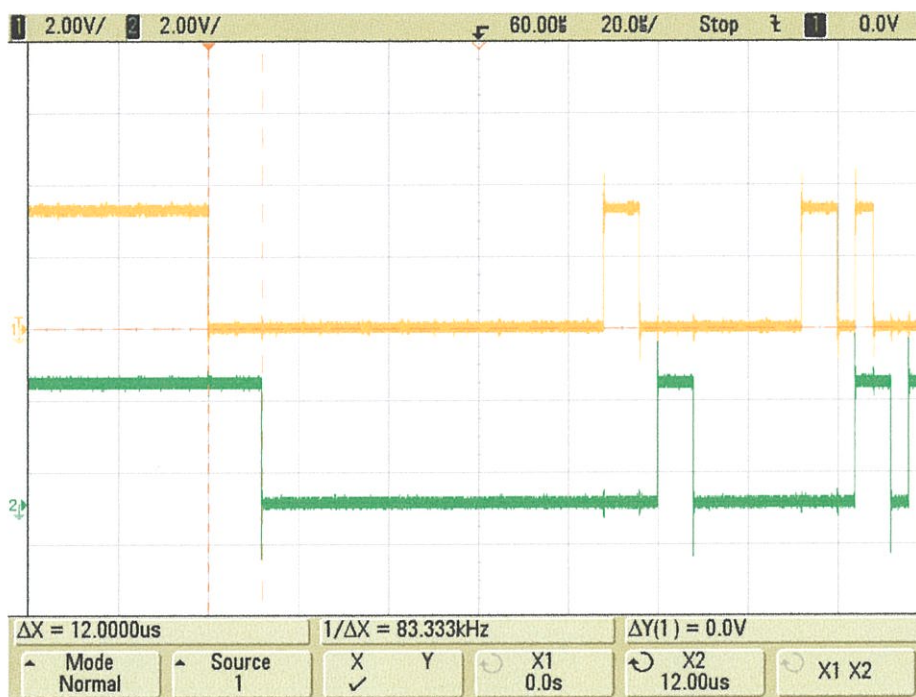
(ข)



(a)



(b)

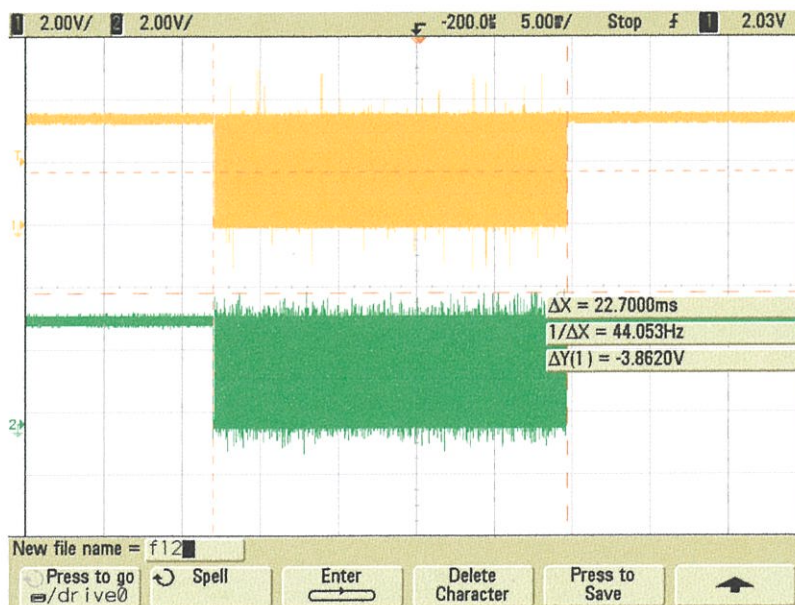


(จ)

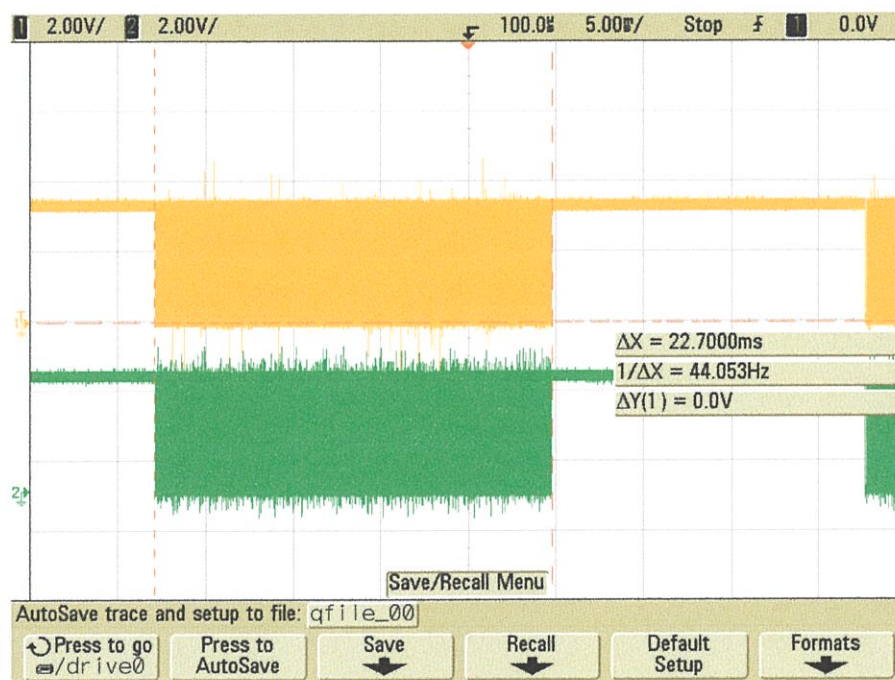
รูปที่ 5.13 (ก) ช่องสัญญาณที่ 0 และ 1, (ข) ช่องสัญญาณที่ 1 และ 2, (ค) ช่องสัญญาณที่ 2 และ 3, (ง) ช่องสัญญาณที่ 0 และ 2, (จ) ช่องสัญญาณที่ 0 และ 3

จากรูปที่ 5.13 (ก), (ข), (ค), (ง) เป็นสัญญาณเอาต์พุตของช่องสัญญาณต่าง ๆ เพื่อหาค่าหน่วยเวลาระหว่างช่องสัญญาณ พบว่าค่าหน่วยเวลาระหว่าง ช่องสัญญาณที่ 0 และ 1 มีค่าเท่ากับ 4 ไมโครวินาที และค่าหน่วยเวลาระหว่างช่องสัญญาณ 0 และ 3 มีค่าเท่ากับ 12 ไมโครเซก

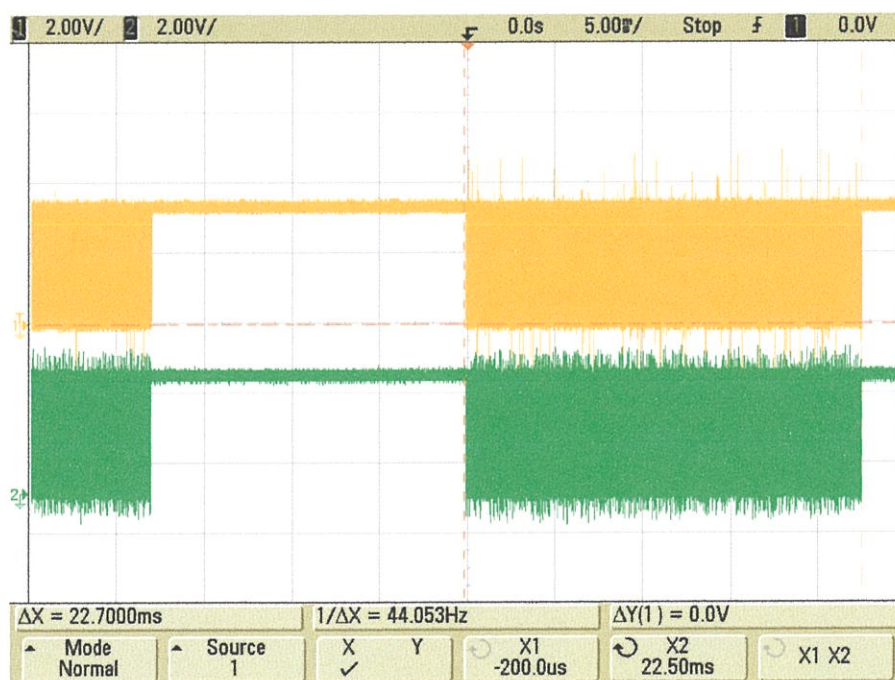
5.4.2 เฟรมเรทสูงสุดที่รองรับ



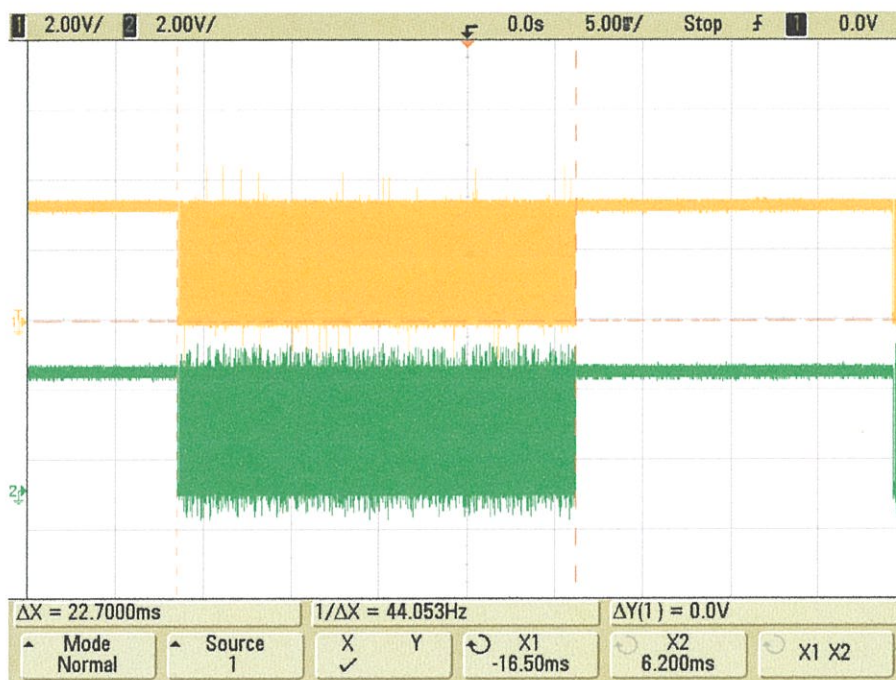
(ก)



(a)



(b)



(ง)

รูปที่ 5.14 เวลาที่ใช้ในการแสดงผลของการส่งข้อมูล 512 บิต (ก) ช่องสัญญาณ 0 และ 1, (ข) ช่องสัญญาณ 1 และ 2, (ค) ช่องสัญญาณ 2 และ 3, (ง) ช่องสัญญาณ 3 และ 1,

เฟรมเรทสูงสุดที่รองรับวัดโดยใช้ฮอสซิลโลสโคป วัดช่วงเวลาที่ส่งข้อมูลทั้ง 512 ข้อมูล ซึ่งในการส่งข้อมูลแต่ละเฟรมได้เขียนโปรแกรมเพื่อให้เกิดช่วงหน่วงเวลาเพื่อให้สะดวกแก่การวัดสัญญาณ โดยจากรูปที่ 5.14 จะเห็นระยะเวลาทั้งหมดที่ใช้ในการส่งภาพ 1 เฟรมคือ 22.7 มิลลิวินาที ดังนั้นอัตราสูงสุดที่สามารถเล่นภาพได้เท่ากับ 44 ภาพต่อวินาที

บทที่ 6

สรุปผลการวิจัย และข้อเสนอแนะ

การสร้างแสงและสีต่าง ๆ ในปัจจุบันได้มีการพัฒนาขึ้นอย่างรวดเร็ว อันเนื่องมาจากการพัฒนาของเทคโนโลยี LED ซึ่งทำให้สามารถสร้างสีได้มากมายจากการผสมสีของ LED และอายุการใช้งานของ LED ยาวนานกว่าอุปกรณ์ให้แสงสว่างแบบเก่า ทำให้มีอาคารหรือสถานที่ต่าง ๆ ใช้แสงสว่างในการตกแต่งเพิ่มมากขึ้น ซึ่งลักษณะการแสงจะมีทั้งแบบ stripe light, floodlight, flash, Text-Display ซึ่งการแสงเหล่านี้จะเป็นการแสงแบบสุ่ม (random) หรือแบบวนรอบ (loop) วิธีการแสงในลักษณะสุ่ม หรือวนรอบนี้ การนำคอมพิวเตอร์เข้าใช้ในการส่งข้อมูลให้กับอุปกรณ์ ทำให้เกิดความจำเป็น และสิ้นเปลืองพลังงานมากกว่าการใช้ระบบฝังตัว หรือหากนำไมโครคอนโทรลเลอร์มาใช้จะทำให้ไม่สามารถทำงานได้อย่างมีประสิทธิภาพเนื่องจากไมโครคอนโทรลเลอร์จะทำงานแบบลำดับขั้น ดังนั้นหากเราใช้ไมโครคอนโทรลเลอร์ออกแบบให้มี DMX Controller หลาย ๆ ช่องสัญญาณ (Channel) จะทำให้การทำงานในแต่ละช่องสัญญาณนั้นเกิดการหน่วงเวลา (การ Delay มากน้อยขึ้นอยู่กับความเร็วและจำนวนจำนวนช่องสัญญาณ) ทำให้การแสงและสีต่าง ๆ ผิดเพี้ยนไปจากความต้องการจริง

Embedded DMX512 Controller สามารถเชื่อมต่ออุปกรณ์ได้จำนวนมากเพื่อให้รองรับกับอุปกรณ์ในปัจจุบัน โดยไม่ต้องใช้ PC ซึ่งกินพลังงานมากกว่าและมีราคาสูงกว่า โดยที่ Embedded DMX512 Controller ทำงานแบบ Real-Time จึงทำให้ทุกช่องสัญญาณสามารถทำงานพร้อมกันได้ โดยการควบคุมต่าง ๆ หรือการ Update ข้อมูลจะทำงานผ่านทาง TCP/IP จึงทำให้สามารถเปลี่ยนแปลงข้อมูลที่จะแสงได้ ผ่านทางอินเทอร์เน็ตซึ่งจำนวนช่องสัญญาณ (Channel) นั้นสามารถเพิ่มได้ตามขนาดของ FPGA ที่ใช้

วิธีการที่นำเสนอในวิทยานิพนธ์เป็นเทคนิคหนึ่งเท่านั้นที่ช่วยในการปรับปรุงสมรรถนะของระบบการส่งข้อมูลในโปรโตคอลดีเอ็มเอช 512 แต่ก็ยังมีเทคนิควิธีการอื่นที่น่าสนใจ และสามารถทำให้ระบบมีสมรรถนะที่ดีขึ้นไม่ว่าจะเทคนิคของการรวมกันระหว่างดีเอ็มเอช และ แคนบัส เป็นต้น ซึ่งเมื่อนำวิธีการต่างๆ มาใช้ร่วมกันก็จะทำให้ระบบมีสมรรถนะที่ดีมากยิ่งขึ้น

บรรณานุกรม

- [1] USITT, “USITT DMX512,DMX512/1990,DMX512-A”
- [2] American National Standard E1.11-2004, Entertainment Technology – USITT DMX512-A – Asynchronous Serial Digital Data Transmission Stand for Controlling Lighting Equipment and Accessories, Entertainment Services and Technology Association, USA, 2004.
- [3] C. Lopez, L. Costas-Perez, J. Doval-Gandoy, **FPGA-Based Improved High Brightness RGB LED Matrix Automation.** IEEE International Symposium on Industrial Electronics, ISIE’08, Vigo pp 1902-1907, 2008.
- [4] F. Tian, X. Wu, J. Liu, “**Research and Realization of LED Display System Based on Combining CAN Bus with DMX512 Standard,**” IEEE International Conference on Audio, Language and Image Processing, ICALIP’08, pp. 813-818, July 2008.
- [5] H. Wang, Y. Cai, M. Wu, “**Research on landscape lighting system based on DMX512,**” IEEE International Conference on Industrial Informatics, INDIN’09, pp. 113-118, June 2009.
- [6] Xilinx University Program VIRTEX II PRO Reference Guide, Xilinx Inc., USA 2005.
- [7] Xilinx, “XILKernel”, 12.2006, pp. 1-3
- [8] Xilinx, “CoreConnect Architecture – Processor Local Bus (PLB)”
- [9] X. Yansheng, “**Based on TCP/IP & DMX512 Protocols of LED Decoration Lighting System,**” China Illuminating Engineering Journal, Volume 4, December, 2005
- [10] D. J Smith, “**HDL Chip Design,**” Sixth Edition, Doone Publications, Madison, AL, USA 1999.
- [11] S. Kilts, **Advanced FPGA Design: Architecture, Implementation, and Optimization.** Wiley-IEEE Press, June 2007.
- [12] J. Bergeron, **Writing Testbenches: Functional Verification of HDL Models,** Springer, 2nd edition, February 2003.
- [13] S. Brown, **Fundamental of Digital logic with VHDL design,** Mcgraw-Hill Second edition, July 2004.
- [14] BOSCH, “**CAN Specification**”, Version 2 September 1991