

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

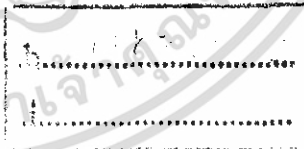
การออกแบบวงจรสังเคราะห์ความถี่แบบดิจิทัล โดยใช้การประมาณค่าจาก
สมการโพลีโนเมียล และปรับปรุงวงจรสะสมเฟส

**A DESIGN OF DIGITAL FREQUENCY SYNTHESIZER USING A POLYNOMIAL
APPOXIMATION AND PHASE ACCUMULATOR IMPROVEMENT**



โดย
นายจักรวรรดิ กาญจนะสิงห์
นายพลิชฐ์ ศรีเพรา

เลขหมู่.....
เลขทะเบียน 62822
เลข..... 23 ส.ค. 2549



ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมโทรคมนาคม
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2548

ผ่านการตรวจชิ้นงานแล้ว
(ลงชื่อ).....ผู้ตรวจ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์อย่างอื่น
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้
(ลงชื่อ).....ผู้ตรวจ

การออกแบบวงจรสังเคราะห์ความถี่แบบดิจิทัล โดยใช้การประมาณค่าจาก
สมการโพลีโนเมียล และปรับปรุงวงจรสะสมเฟส

**A DESIGN OF DIGITAL FREQUENCY SYNTHESIZER USING A POLYNOMIAL
APPROXIMATION AND PHASE ACCUMULATOR IMPROVEMENT**



นายจักรวรรดิ กาญจนะสิงห์ 46015046

นายพลิชฐ์ ศรีเหรา 46015062

อาจารย์ที่ปรึกษา

รศ.ดร.กอบชัย เดชหาญ

อาจารย์ศรวัฒน์ ชิวปรีชา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2548

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปริญญาโทปีการศึกษา 2548

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การออกแบบวงจรสังเคราะห์ความถี่แบบดิจิทัลโดยใช้การประมาณค่าจากสมการโพลีโนเมียล
และปรับปรุงวงจรสะสมเฟส

**A DESIGN OF DIGITAL FREQUENCY SYNTHESIZER USING A POLYNOMIAL
APPOXIMATION AND PHASE ACCUMULATOR IMPROVEMENT**


ผู้จัดทำ

1. นายจักรวรรดิ กาญจนะสิงห์ 46015046

2. นายพลิชฐ์ ศรีหระ 46015062


.....

(รศ.ดร. กอบชัย เดชหาญ)


.....

(อาจารย์ ศรวัดณ์ ชิวปรีชา)

อาจารย์ที่ปรึกษา

อาจารย์ที่ปรึกษา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การออกแบบวงจรสังเคราะห์ความถี่แบบดิจิทัล โดยใช้การประมาณ
ค่าจากสมการโพลีโนเมียล และปรับปรุงวงจรสะสมเฟส

**A DESIGN OF DIGITAL FREQUENCY SYNTHESIZER
USING A POLYNOMIAL APPROXIMATION AND PHASE
ACCUMULATOR IMPROVEMENT**

โดย นายจักรวรรดิ กาญจนะสิงห์ 46015046

นายพลิชฐ์ ศรีเหรา 46015062

อาจารย์ที่ปรึกษา รศ.ดร. กอบชัย เดชหาญ
อ. ศรวิวัฒน์ ชิวปรีชา

บทคัดย่อ

โครงงานนี้เป็นการออกแบบวงจรสังเคราะห์ความถี่แบบดิจิทัลบน FPGA และสามารถโปรแกรมผ่านคอมพิวเตอร์ โดยการใช้การประมาณค่าแบบโพลีโนเมียลในการประมาณค่าสัญญาณที่ต้องการสังเคราะห์ และพัฒนาวงจรสะสมเฟสให้ทำงานได้เร็วขึ้น โดยอธิบายพฤติกรรมการทำงานของวงจรโดยใช้ภาษา VHDL

Abstract

This thesis presents a design of digital frequency synthesizer on FPGA and it is able to program via computer. The technique to design is based on polynomial approximation to estimate the desired synthesizing signal. The thesis also proposes a method to develop the phase accumulator to have the high speed. All operating characteristics of this circuit are described by VHDL language.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

ปริญญาานิพนธ์นี้สำเร็จลุล่วงได้ดีด้วยความกรุณาของ รศ.ดร.กอบชัย เดชหาญและ
อาจารย์ ศรวีฉวี ชิวปริษา อาจารย์ที่ปรึกษา ผู้คอยเอาใจใส่ดูแลและเมตตาแก้ไขด้วยดีเสมอมา คณะผู้จัดทำ
รู้สึกซาบซึ้งในความกรุณาเป็นอย่างยิ่งและขอขอบคุณอย่างสูง

ขอขอบคุณห้องปฏิบัติการสำหรับสถานที่ เครื่องมือ และอุปกรณ์ในการทำวิทยานิพนธ์ฉบับนี้
ขอบคุณพระคุณอาจารย์ทุกท่านที่ประสิทธิ์ประสาทวิชาความรู้ต่างๆ ให้แก่ศิษย์ทั้งทางตรงและทางอ้อม
ตลอดจนกำลังใจ คำแนะนำเกี่ยวกับปัญหาต่างๆ และความช่วยเหลือทุกอย่างระหว่างทำปริญญาานิพนธ์นี้
ของเพื่อนๆทุกคน และรุ่นพี่ที่ให้คำปรึกษา

สุดท้ายนี้ขอกราบขอบคุณพระคุณบิดา มารดา ที่ให้ความสำคัญกับการศึกษาของลูกและให้
การสนับสนุนเอาใจใส่ดูแลด้วยดีเสมอมา รวมทั้งกำลังใจอันยิ่งใหญ่อย่างหาที่เปรียบมิได้

นายจักรวรรดิ กาญจนะสิงห์
นายพลิชฐ์ ศรีเระ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

	หน้า
บทที่ 1 บทนำ	1
บทที่ 2 ทฤษฎีและหลักการ	2
2.1 ทฤษฎีการสังเคราะห์สัญญาณดิจิทัลโดยตรง	2
2.2 การเขียนภาษา VHDL	3
2.2.1 Terminology และ Convention	3
2.2.2 การออกแบบจากบนลงล่าง	5
2.2.3 ภาษา VHDL และ ส่วนประกอบต่างๆของภาษา	7
2.2.3.1 หน่วยการออกแบบเอนทิตี	7
2.2.3.2 หน่วยการออกแบบสถาปัตยกรรม	8
2.2.3.3 หน่วยการออกแบบแพ็คเกจ	12
2.2.3.4 หน่วยการออกแบบโครงสร้าง	13
2.2.4 ชุดคำสั่งลำดับ (Sequential Statements)	13
2.2.4.1 Process Statement	14
2.2.4.2 Wait Statement	14
2.2.4.3 IF-THEN-ELSE statement	15
2.2.4.4 CASE Statement	16
2.3 การบีบอัดควอดแรนต์ (Quadrant compression)	16
2.3.1 ขั้นตอนวิธีซันเดอร์แลนด์ (Sunderland algorithm)	19
2.4 การสังเคราะห์ความถี่แบบดิจิทัลที่ไม่ใช้หน่วยความจำ	19
2.4.1 ขั้นตอนวิธีคอร์ดิก (CORDIC algorithm)	20
2.4.2 การคูณเชิงซ้อน (Complex multiplication)	21
2.4.3 การประมาณค่าโพลีโนเมียล (The Polynomial Approximation)	22
บทที่ 3 การออกแบบและการสร้าง	23
3.1 การออกแบบวงจรเชิงเลขด้วยอุปกรณ์ FPGA	23
3.2 การออกแบบโดยใช้ภาษาอธิบายการทำงานของฮาร์ดแวร์	24
3.2.1 การจำลองการทำงานของวงจร (Simulation)	24
3.2.2 การสังเคราะห์วงจร	24
3.2.3 การแบ่งวงจร (Partitioning)	25
3.2.4 การวางอุปกรณ์ (Placement)	25
3.2.5 การเชื่อมต่อสัญญาณ (Routing)	25
3.2.6 การโปรแกรมอุปกรณ์ FPGA (Configuration)	26

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

	หน้า
3.3 FPGA ตระกูล SPRATAN XC3S-xxx	26
3.4 Direct Digital Frequency Synthesizer (DDS)	28
3.4.1 การออกแบบและการสร้าง	29
3.4.2 การสร้างสัญญาณรูปคลื่นไซน์	30
3.5 รายละเอียดการออกแบบ	31
3.6 ส่วนดิจิทัล (Digital part)	31
3.6.1 วงจรสร้างเฟส 32 บิต (32-bit phase accumulator)	31
3.6.2 วงจรบวกแบบ Look-Ahead-Carry	32
3.6.3 วงจรเปลี่ยนเฟสเป็นแอมพลิจูด (Phase-to-amplitude converter)	35
3.6.3.1 รูปไซน์ (Sine)	36
3.6.3.2 รูปแรมป์ (Ramp)	37
3.6.4 การประมาณค่าโพลีโนเมียล (The Polynomial Approximation)	38
3.6.4.1 รูปไซน์ (Sine)	38
3.7 วงจรแปลงระดับแรงดัน	39
3.7.1 หลักการทำงาน	40
3.8 วงจรแปลงระดับสัญญาณดิจิทัลเป็นอนาล็อก	40
3.8.1 การออกแบบวงจรและสายวงจรส่วนของวงจรแปลงดิจิทัลเป็นอนาล็อก	41
3.8.2 การออกแบบวงจรและสายวงจรส่วนของวงจรส่งข้อมูลผ่านพอร์ตอนุกรม โดยใช้ MCS-51	42
บทที่ 4 การทดลองและผลการทำงาน	44
4.1 การควบคุมการใช้งานการเชื่อมต่อผ่านพอร์ตอนุกรม	45
4.2 การออกแบบวงจรส่วนต่างๆ โดยใช้ภาษา VHDL	46
4.2.1 ส่วนของวงจร DIV 2500	46
4.2.2 ส่วนของวงจร Dff	48
4.2.3 ส่วนของวงจร Dff-com	50
4.2.4 ส่วนของวงจร SWIT_PULSE	51
4.2.5 ส่วนของวงจร SERIAL_COMMUNICATION	52
4.2.6 ส่วนของวงจร SELECT_INPUT	54
4.2.7 ส่วนของวงจร Lookup table	55
4.2.8 ส่วนของวงจร POLYNOMIAL APPROXIMATION	56

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

	หน้า
4.2.9 ส่วนของวงจร RANDOM	58
4.2.10 ส่วนของวงจร MUX_OUTPUT	59
4.2.11 ส่วนของวงจร ACM_32BIT	61
4.2.12 ส่วนของวงจร (8 - bit Carry Lookahead Adder)	62
4.3 การนำไปใช้งานจริง	64
4.4 การจัดตำแหน่งขา FPGA	65
4.5 ผลการทดลองการทำงานของ DDS OSCILATOR โดยใช้วิธีตารางเปิดดู(Lookup-Table)	67
4.5.1 การทดสอบการกำเนิดสัญญาณสี่เหลี่ยมที่ความถี่ต่างๆ	67
4.5.2 การทดสอบการกำเนิดสัญญาณสามเหลี่ยมที่ความถี่ต่างๆ	72
4.5.3 การทดสอบการกำเนิดสัญญาณฟันเลื่อยที่ความถี่ต่างๆ	77
4.5.4 การทดสอบการกำเนิดสัญญาณซายน์ที่ได้จากตารางเปิดดูเปรียบเทียบกับ สัญญาณ ซายน์ที่ได้จากสมการ โพลีโนเมียลที่ความถี่ต่างๆ	80
บทที่ 5 บทวิจารณ์และสรุป ภาคผนวก กิตติกรรมประกาศ หนังสืออ้างอิง	98

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป

	หน้า
รูปที่ 2.1 ตัวอย่างการเก็บข้อมูลของสัญญาณไซน์ (Sine Wave) ของ 1 คาบเวลา	2
รูปที่ 2.2 แสดงขั้นตอนการออกแบบจากบนลงล่าง	6
รูปที่ 2.3 แสดงโครงสร้างโดยทั่วไปของหน่วยการออกแบบเอนทิตี	8
รูปที่ 2.4 แสดงรูปแบบของ RS_flipflop	8
รูปที่ 2.5 แสดงโครงสร้างโดยทั่วไปของหน่วยการออกแบบสถาปัตยกรรม	9
รูปที่ 2.6 แสดงหน่วยการออกแบบสถาปัตยกรรมของ RS_flipflop	10
รูปที่ 2.7 แสดงโครงสร้างภายในสถาปัตยกรรมของ RS_flipflop	10
รูปที่ 2.8 แสดงหน่วยการออกแบบสถาปัตยกรรมของ RS_flipflop ในลักษณะโครงสร้าง	10
รูปที่ 2.9 แสดงหน่วยการออกแบบสถาปัตยกรรมของ RS_flipflop ในลักษณะพฤติกรรม	11
รูปที่ 2.10 แสดงหน่วยการออกแบบสถาปัตยกรรมของ RS_flipflop ในลักษณะผสม	11
รูปที่ 2.11 แสดงโครงสร้างโดยทั่วไปของส่วนการประกาศเพิกเกิด	12
รูปที่ 2.12 แสดงโครงสร้างโดยทั่วไปของบอดีเพิกเกิด	13
รูปที่ 2.13 แสดงโครงสร้างโดยทั่วไปของหน่วยการออกแบบ โครงแบบ	13
รูปที่ 2.14 การสังเคราะห์สัญญาณชาน์เต็มรูปด้วยเทคนิคบิวัตควอแดรนต์	17
รูปที่ 2.15 ขั้นตอนวิธีผลต่างชาน์-เฟส	18
รูปที่ 2.16 ขั้นตอนวิธีซันเตอร์แลนด์	19
รูปที่ 2.17 การสร้างสัญญาณชาน์ด้วยขั้นตอนวิธีคอร์ดิก	20
รูปที่ 2.18 การหมุนเวกเตอร์ของการคูณเชิงซ้อน เมื่อ $r_\alpha = r_\beta$	21
รูปที่ 2.19 การสังเคราะห์สัญญาณชาน์ด้วยการคูณเชิงซ้อน	22
รูปที่ 2.20 สถาปัตยกรรมของวงจรสังเคราะห์ความถี่แบบดิจิทัลโดยใช้วิธีการประมาณค่าโพลีโนเมียล	22
รูปที่ 3.1 ลักษณะด้านบนของตัว FPGA	23
รูปที่ 3.2 แสดงความหมายเบอร์บนตัว FPGA	23
รูปที่ 3.3 แสดง การดาวน์โหลดโปรแกรมลงชิพ FPGA ที่หน้าจจะแสดงชิพ Flash PROM และ FPGA พร้อมกัน	26
รูปที่ 3.4 โครงสร้างภายในของ FPGA ตระกูล SPRATAN3 เบอร์ XC3S200	27
รูปที่ 3.5 แสดง Block Diagram โครงสร้างของวงจรสังเคราะห์ความถี่ที่ใช้หน่วยความจำเป็นค่าขนาดของสัญญาณเอาไว้	28
รูปที่ 3.6 วงจรสร้างสัญญาณชาน์แบบเชิงเลข	28
รูปที่ 3.7 ขั้นตอนการทำงานของ ตัวกำเนิดสัญญาณไซน์	29
รูปที่ 3.8 หน่วยความจำสำหรับเปลี่ยนเฟสเป็นแอมพลิจูดของสัญญาณชาน์	30
รูปที่ 3.9 ผลจากค่าในตารางมาวาดกราฟ	30

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป (ต่อ)

	หน้า
รูปที่ 3.10 สถาปัตยกรรมของชิปวงจรรวมเพื่อสังเคราะห์ความถี่แบบดิจิทัลที่ใช้หน่วยความจำ	31
รูปที่ 3.11 โครงสร้างของวงจรถ่ายเฟสขนาด 32 บิต	32
รูปที่ 3.12 วงจรบวกแบบ Look-Ahead-Carry	34
รูปที่ 3.13 วงจรบวกแบบ Look-Ahead-Carry ในรูปแบบที่เป็นโมดูล	36
รูปที่ 3.14 ขั้นตอนวิธีโมดิไฟซันเดอร์แลนด์	37
รูปที่ 3.15 การสังเคราะห์สัญญาณรูปฟันเลื่อย	37
รูปที่ 3.16 สถาปัตยกรรมของวงจรถ่ายเฟสความถี่แบบดิจิทัลโดยใช้วิธีการประมาณค่าโพลีโนเมียล	39
รูปที่ 3.17 โครงสร้างของวงจรถ่ายเฟสโพลีโนเมียล	39
รูปที่ 3.18 วงจรแปลงระดับแรงดัน	40
รูปที่ 3.19 วงจรแปลงดิจิทัลเป็นอนาล็อกและลายวงจรแปลงดิจิทัลเป็นอนาล็อก	41
รูปที่ 3.20 วงจรส่งข้อมูลผ่านพอร์ตอนุกรมและลายวงจรส่งข้อมูลจาก MCS-51	42
รูปที่ 3.21 ภาพถ่ายอุปกรณ์และวงจรถ่ายเฟสจริง	43
รูปที่ 4.1 แสดงเมนูหลักโปรแกรมการใช้งานดิจิทัลออสซิลเลเตอร์	45
รูปที่ 4.2 แสดงเมนูของค่าที่คำนวณได้	46
รูปที่ 4.3 แสดงเมนู Project Create	46
รูปที่ 4.4 สัญลักษณ์ของส่วนวงจร DIV 2500	47
รูปที่ 4.5 ผลการสังเคราะห์เป็นส่วนวงจร DIV 2500	47
รูปที่ 4.6 ผลการจำลองการทำงานของส่วนวงจร DIV 2500	48
รูปที่ 4.7 สัญลักษณ์ของส่วนวงจร Dff	48
รูปที่ 4.8 ผลการสังเคราะห์เป็นส่วนวงจร Dff	49
รูปที่ 4.9 ผลการจำลองการทำงานของส่วนวงจร Dff	49
รูปที่ 4.10 สัญลักษณ์ของส่วนวงจร Dff_com	50
รูปที่ 4.11 ผลการสังเคราะห์เป็นส่วนวงจร Dff_com	50
รูปที่ 4.12 ผลการจำลองการทำงานของส่วนวงจร Dff_com	51
รูปที่ 4.13 สัญลักษณ์ของส่วนวงจร SWIT_PULSE	51
รูปที่ 4.14 ผลการสังเคราะห์เป็นส่วนวงจร SWIT_PULSE	51
รูปที่ 4.15 ผลการจำลองการทำงานของส่วนวงจร SWIT_PULSE	52
รูปที่ 4.16 สัญลักษณ์ของส่วนวงจร SERIAL_COMMUNICATION	52
รูปที่ 4.17 ผลการสังเคราะห์เป็นส่วนวงจร SERIAL_COMMUNICATION	53
รูปที่ 4.18 ผลการจำลองการทำงานของส่วนวงจร SERIAL_COMMUNICATION	53
รูปที่ 4.19 สัญลักษณ์ของส่วนวงจร SELECT_INPUT	54
รูปที่ 4.20 ผลการสังเคราะห์เป็นส่วนวงจร SELECT_INPUT	54

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้กับโครงการแข่งขันการศึกษาค้นคว้า ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป (ต่อ)

	หน้า
รูปที่ 4.21 ผลการจำลองการทำงานของส่วนวงจร SELECT_INPUT	54
รูปที่ 4.22 สัญลักษณ์ของส่วนวงจร LUT	55
รูปที่ 4.23 ผลการสังเคราะห์เป็นวงจร LUT	55
รูปที่ 4.24 ผลการจำลองการทำงานของส่วนวงจร LUT	56
รูปที่ 4.25 สัญลักษณ์ของส่วนวงจร POLYNOMIAL APPROXIMATION	56
รูปที่ 4.26 ผลการสังเคราะห์เป็นวงจร POLYNOMIAL APPROXIMATION	57
รูปที่ 4.27 ผลการจำลองการทำงานของส่วนวงจร POLYNOMIAL APPROXIMATION	57
รูปที่ 4.28 สัญลักษณ์ของส่วนวงจร RANDOM	58
รูปที่ 4.29 ผลการสังเคราะห์เป็นวงจร RANDOM	58
รูปที่ 4.30 ผลการจำลองการทำงานของส่วนวงจร RANDOM	58
รูปที่ 4.31 สัญลักษณ์ของส่วนวงจร MUX_OUTPUT	59
รูปที่ 4.32 ผลการสังเคราะห์เป็นวงจร MUX_OUTPUT	60
รูปที่ 4.33 ผลการจำลองการทำงานของส่วนวงจร MUX_OUTPUT	60
รูปที่ 4.34 สัญลักษณ์ของส่วนวงจร ACM32	61
รูปที่ 4.35 ผลการสังเคราะห์เป็นวงจร ACM32	61
รูปที่ 4.36 ผลการจำลองการทำงานของส่วนวงจร ACM32	62
รูปที่ 4.37 สัญลักษณ์ของส่วนวงจร ADDER 8 BIT	62
รูปที่ 4.38 ผลการสังเคราะห์เป็นวงจร ADDER 8 BIT	63
รูปที่ 4.39 ผลการจำลองการทำงานของส่วนวงจร ADDER 8 BIT	63
รูปที่ 4.40 บอร์ดทดลอง FPGA รุ่น FPGA-3D-XC3S200	64
รูปที่ 4.41 รูปวงจรทดสอบการทำงานของวงจร	64
รูปที่ 4.42 ผลการจำลองการทำงานของส่วนวงจร DDFS	65
รูปที่ 4.43 แสดงการกำหนดค่าให้ FPGA	65
รูปที่ 4.44 แสดงการโปรแกรมวงจรลง FPGA และพร้อมใช้งาน	66
รูปที่ 4.45 การทดสอบการกำเนิดสัญญาณสี่เหลี่ยมที่ความถี่ต่างๆ	67
รูปที่ 4.46 การทดสอบการกำเนิดสัญญาณสามเหลี่ยมที่ความถี่ต่างๆ	72
รูปที่ 4.47 การทดสอบการกำเนิดสัญญาณฟันเลื่อยที่ความถี่ต่างๆ	77
รูปที่ 4.48 การทดสอบการกำเนิดสัญญาณไซน์เปรียบเทียบกับสัญญาณไซน์ที่ได้จาก วิธีการ โพลีโนเมียล ที่ความถี่ต่างๆ	80

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง

	หน้า
ตารางที่ 2.1 ตารางแสดงการเก็บข้อมูลของสัญญาณไซน์ (Sine Wave)	3
ตารางที่ 2.2 ความสัมพันธ์ของบิตเอ็มเอสบีกับควอดแรนต์และสัญญาณไซน์	17
ตารางที่ 3.1 แสดง Ram แบบ Single Port ที่สร้างจาก Block Ram แต่ละชุด	27
ตารางที่ 3.2 ค่าคงที่ที่ใช้ในสมการที่ (3.12)	38
ตารางที่ 4.1 รูปแบบการส่งข้อมูลผ่านทางพอร์ตอนุกรม	44



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.1 ความเป็นมาของโครงการ

ปัจจุบันมีการใช้เครื่องมือสื่อสารประเภทต่างๆ มีส่วนช่วยอำนวยความสะดวกในการดำรงชีวิตประจำวันของมนุษย์เป็นอย่างมากไม่ว่าจะเป็นการสื่อสารดาวเทียม (Satellite communication), โทรศัพท์มือถือ (Mobile phone), วิทยุติดตามตัว (Radio pager) รวมไปถึงอุปกรณ์เครื่องใช้ไฟฟ้าตามบ้าน (Home appliance) ส่วนประกอบหลักที่สำคัญที่สุดในอุปกรณ์สื่อสารเหล่านี้ก็คือวงจรเพื่อสังเคราะห์ความถี่ (frequency synthesizer) ที่ทำหน้าที่สังเคราะห์สัญญาณให้ได้ความถี่ตามต้องการ วงจรสังเคราะห์ความถี่ที่มีใช้กันอยู่ในปัจจุบันนั้นมีมากมายหลายประเภทด้วยกัน

เนื้อหาในบทนี้จะเกี่ยวกับทฤษฎีและหลักการการทำงานของวงจรสังเคราะห์ความถี่แบบดิจิทัล คือวิธีแบบใช้ตารางเปิดดู(Look-up table), และการประมาณค่าโพลิโนเมียล(The Polynomial Approximation)

1.2 ความมุ่งหมายและวัตถุประสงค์ของปริิญญาานิพนธ์

ในปริิญญาานิพนธ์กล่าวถึงวงจรกำเนิดสัญญาณดิจิทัลโดยใช้การอธิบายพฤติกรรมการทำงาน ของวงจรด้วยภาษา VHDL (Very High Speed Integrated Circuit Hardware Description Language) ซึ่งปริิญญาานิพนธ์นี้มีจุดประสงค์ดังนี้

1. เพื่อศึกษาทฤษฎีและหลักการการคำนวณของวงจรกำเนิดสัญญาณดิจิทัล (Direct Digital Frequency Synthesis)
2. เพื่อศึกษาการเขียนภาษาวีเอชดีแอลในการออกแบบระบบฮาร์ดแวร์ดิจิทัล ซึ่งเริ่มตั้งแต่การออกแบบแก้ไขตรวจสอบ จำลองการทำงาน จนถึงสังเคราะห์วงจร
3. เพื่อศึกษาโปรแกรมต่างๆที่ใช้ในการออกแบบฮาร์ดแวร์ดิจิทัล เช่น WebPack Xilinx.
4. เพื่อศึกษาการเชื่อมต่อคอมพิวเตอร์กับเอฟพีจีเอผ่านทางพอร์ตอนุกรม
5. เพื่อศึกษาการใช้งาน MCS-51 ที่ใช้เชื่อมต่อกับเอฟพีจีเอผ่านทางพอร์ตอนุกรม
6. สามารถสร้างวงจรกำเนิดสัญญาณชายน์

1.3 ขอบเขตของโครงการ

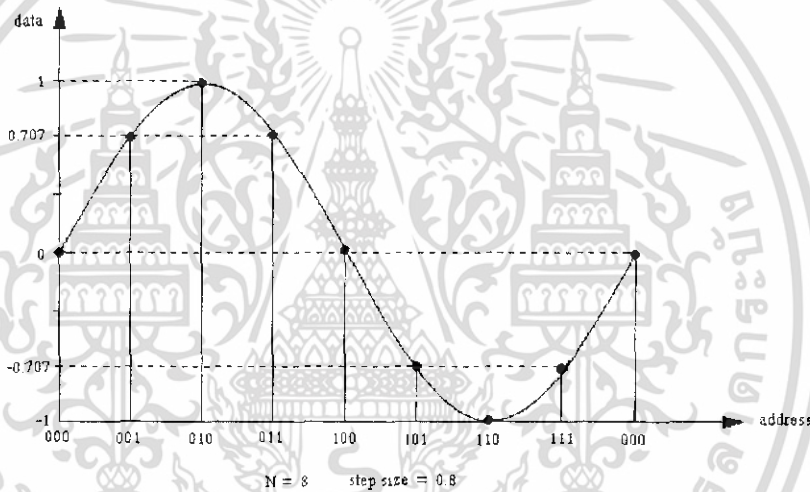
ขอบเขตของโครงการนี้จะออกแบบวงจรสังเคราะห์ความถี่แบบดิจิทัลโดยใช้การประมาณค่าโพลิโนเมียลแทนการใช้หน่วยความจำ โดยจะออกแบบวงจรสร้างเฟส (Phase accumulator) และวงจรเปลี่ยนเฟสเป็นแอมพลิจูด (Phase-to-amplitude converter) ด้วยวิธีการออกแบบจากบนลงล่าง (Top-down design) ซึ่งวงจรทั้งสองส่วนนี้จะถูกสังเคราะห์ (Synthesis) และจัดวาง-เชื่อมต่อ (Place-and-route) ลงบนชิปเอฟพีจีเอ (Field Programmable Gate Array) แล้วนำไปทดสอบร่วมกับชิปแปลงดิจิทัลเป็นอนาล็อก (DAC chip) เพื่อตรวจสอบการทำงานและประสิทธิภาพของวงจรต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2 ทฤษฎีและหลักการ

2.1 ทฤษฎีการสังเคราะห์สัญญาณดิจิทัลโดยตรง (Direct Digital Frequency Synthesizer : DDS)

วงจรสังเคราะห์ความถี่แบบดิจิทัล (Direct Digital Frequency Synthesizer : DDS) มีหลักคือในวงจรดิจิทัลสร้างสัญญาณเฟสขึ้นมาเพื่อนำมาอ้างอิงเป็นค่าแอมพลิจูดของรูปสัญญาณ (Waveform) ที่ต้องการสังเคราะห์ โดยใช้หน่วยความจำในการเก็บรูปสัญญาณของสัญญาณที่ต้องการสังเคราะห์จะเก็บข้อมูลของสัญญาณที่จะกำเนิดขึ้นมานั้นให้ครบคาบของสัญญาณซึ่งถ้าหากเรายิ่งเก็บข้อมูลของสัญญาณจำนวนมากเท่าไรก็จะทำให้สัญญาณที่ผลิตออกมาใกล้เคียงกับความเป็นจริงมากเท่านั้น แต่จะมีข้อเสียคือจะสิ้นเปลืองเนื้อที่ในหน่วยความจำมากตามไปด้วย มีข้อจำกัดในเรื่องของเวลาในการเข้าถึงหน่วยความจำ ขนาดของวงจร



รูปที่ 2.1 ตัวอย่างการเก็บข้อมูลของสัญญาณไซน์ (Sine Wave) ของ 1 คาบเวลา

ซึ่งการเก็บข้อมูลจะต้องใช้การสุ่ม (Sampling) ข้อมูลบนสัญญาณแต่ละจุดด้วยเวลาที่เท่ากันทุกจุด ดังนั้นจุดบนสัญญาณที่ต้องการเก็บคือ 360/จำนวนข้อมูล เช่นต้องการเก็บข้อมูล 1024 ค่า ดังนั้นจะต้องทำการเก็บข้อมูล สามารถทำเป็นตารางได้ดังนี้ การเก็บข้อมูลจำนวน N ค่าพิจารณาได้จากตารางที่ 2.1 หลังจากที่เราได้ข้อมูลมาแต่ละจุดแล้ว จะต้องมาทำการจัดค่า (Quantization) ให้มีค่าเป็นทางดิจิทัลเมื่อเราได้ข้อมูลที่มีค่าเป็นเลขฐานสองแล้วก็จะนำค่าข้อมูลเหล่านี้ไปเก็บในหน่วยความจำโดยการจัดเรียงกันไปคือค่าของสัญญาณจุดแรกบนสัญญาณจะถูกเก็บที่ตำแหน่ง (Address) แรกค่าของข้อมูลที่สองจะถูกเก็บในตำแหน่ง (Address) ถัดไปจนครบหมดทุกค่า

นอกจากสัญญาณไซน์แล้วเราสามารถสร้างสัญญาณชนิดอื่นได้อีกมากโดยใช้โปรแกรมคำนวณค่าของสัญญาณต่างๆ โดยการคำนวณค่าของสัญญาณแต่ละจุด โดยใช้หลักการเกี่ยวกับการกำเนิดคลื่นไซน์ที่ได้กล่าวมาข้างต้นนั่นเอง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 2.1 ตารางแสดงการเก็บข้อมูลของสัญญาณไซน์ (Sine Wave)

ข้อมูลที	เฟส	ข้อมูล
0	$0. \pi / N$	$f(0)=\sin(0. \pi / N)$
1	$1. \pi / N$	$f(1)=\sin(1. \pi / N)$
2	$2. \pi / N$	$f(2)=\sin(2. \pi / N)$
3	$3. \pi / N$	$f(3)=\sin(3. \pi / N)$
*	*	*
*	*	*
*	*	*
*	*	*
N - 3	$(N - 3). \pi / N$	$f(N - 3)=\sin((N - 3). \pi / N)$
N - 2	$(N - 2). \pi / N$	$f(N - 2)=\sin((N - 2). \pi / N)$
N - 1	$(N - 1). \pi / N$	$f(N - 1)=\sin((N - 1). \pi / N)$

2.2 การเขียนภาษา VHDL

2.2.1 Terminology และ Convention

การเขียนรูปแบบของระบบดิจิทัลด้วยภาษา VHDL นั้น จะมีศัพท์เทคนิคเฉพาะ ฉะนั้นในส่วนนี้จะเป็นการบรรยาย และอธิบายศัพท์บางคำที่จะต้องพบในรายงานชุดนี้

ลักษณะของรูปแบบ (modal styles) : ลักษณะการเขียนรูปแบบ (model) ด้วยภาษา VHDL สามารถแบ่งได้เป็น

- Behavioral Model : หรือที่เรียกอีกอย่างว่า algorithmic description เป็นรูปแบบที่บรรยายพฤติกรรมของระบบดิจิทัล ในส่วนที่บรรยายมีโครงสร้างคล้ายกับภาษาชั้นสูง (high level language) ทั่วไป เช่น PASCAL หรือ C เป็นต้น ในการจำลองการทำงาน (simulation) คำสั่งแต่ละคำสั่ง (statement) จะถูกประเมินผลเป็นไปตามลำดับ (sequential) จากบนลงล่าง ยกเว้นในกรณีของคำสั่ง LOOP หรือการใช้โปรแกรมย่อยรูปแบบลักษณะนี้จะไม่ให้รายละเอียดที่เกี่ยวกับการผลิต หรือโครงสร้างของ Hardware แต่ในทางตรงข้ามที่รายละเอียดเกี่ยวกับความสัมพันธ์ระหว่าง input กับ output ที่ดี

- Dataflow Model : เรียกอีกอย่างหนึ่งได้ว่า “ Register transfer level ” (RTL) เป็นรูปแบบที่ถูกเขียนขึ้น เพื่อจุดประสงค์ที่จะใช้เป็นเครื่องมือสำหรับสังเคราะห์วงจรอัตโนมัติ รูปแบบลักษณะนี้ส่วนใหญ่จะเป็น procedural constructs และ functional operators

- Structural Model : เป็นรูปแบบที่แสดงการเชื่อมต่อกันระหว่างอุปกรณ์ต่างๆ ที่ประกอบกันขึ้นเป็นวงจรหรือระบบดิจิทัล และสามารถเรียกอีกอย่างได้ว่า “ net list representation “ เป็นการเขียนที่แสดงให้เห็นโครงสร้างของ hardware

- Mixed - Level Model : จากคุณสมบัติที่อ่อนตัวของภาษา VHDL จึงสามารถที่จะเขียนรูปแบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โดยใช้ลักษณะต่างๆ บรรยายวงจรหรือระบบดิจิทัลเดียวกันได้ ฉะนั้นรูปแบบเช่นนี้จึงมีการเขียนแบบผสม

Concurrency ในภาษา VHDL นั้น ชุดคำสั่งจะทำงานในเวลาเดียวกันและอิสระต่อกัน ลักษณะเช่นนี้เป็นคุณสมบัติที่เป็นความจริงทางฟิสิกส์ของวงจรรีเลย์ทรอนิกส์ ชุดคำสั่งนี้เรียกว่า “concurrent statement” และจะทำงานก็ต่อเมื่อมีการเปลี่ยนแปลงค่าของสัญญาณ

Sequential : นอกจากความสามารถที่ชุดคำสั่งจะทำงานแบบ concurrent แล้วบางครั้งการเขียนรูปแบบในลักษณะที่บรรยายพฤติกรรมของวงจร มีความจำเป็นที่จะต้องให้ชุดคำสั่งทำงานเป็นลำดับขึ้นเรียงกันจากบนลงล่าง อย่างเช่นการเขียนแบบ behavioral model เป็นต้น ชุดคำสั่งที่เป็น sequential นี้จะใช้ในโปรแกรมย่อย (subprogram) และ process statement

Driver : สัญญาณต่างๆ (signal) ใน VHDL นั้นจะถูกควบคุมด้วยตัวขับหรือ “driver” สัญญาณเหล่านี้จะรับค่าใหม่ (ระดับของสัญญาณ) ได้ด้วยตัวขับนี้เอง

Transaction : การเกิด transaction กับ signal นั้นจะเกิดขึ้นเมื่อมีการกำหนดค่าหนึ่งให้กับ signal signal นั้น ค่าใหม่ที่ signal ได้รับอาจจะมีผลหรือไม่ผลทำให้เกิดการเปลี่ยนแปลงของระดับสัญญาณ (event) เช่นการเปลี่ยนจากค่า logic ‘0’ เป็นค่า logic ‘1’ เป็นต้น

Event : คือการเปลี่ยนระดับค่าของ SIGNAL จากระดับหนึ่งไประดับอื่น อย่างเช่น ในระบบดิจิทัลการเปลี่ยนจาก Logic ‘0’ เป็นค่า Logic ‘1’ หรือในทางตรงกันข้ามถือว่า SIGNAL นั้นเกิด “event” ฉะนั้นจะเห็นได้ว่า การที่จะเกิด event ได้นั้นจะต้องเกิด transaction ไม่จำเป็นต้องเกิด event ทุกครั้ง

Sensitivity List : คือรายชื่อของ signal ต่างๆ ที่มีผลทำให้เกิดการทำงานของ concurrent statement เมื่อเกิด event ขึ้นกับ signal ตัวใดตัวหนึ่งหรือหลายตัวพร้อมกันในรายชื่อนั้น

Object : ในภาษา นั้นคำว่า ใช้เขียนเพื่อบ่งบอกถึงองค์ประกอบส่วนหนึ่งของรูปแบบ ซึ่งเปรียบได้เหมือนกับภาษาซีที่มีไว้สำหรับบรรจุก่าต่างๆ สามารถแบ่งออกได้เป็นสามชั้น (class) ด้วยกันคือ

CONSTANT : ได้แก่ object ประเภทหนึ่งเมื่อกำหนดค่าเริ่มต้นไปแล้วจะคงค่านั้นไว้ตลอดไม่สามารถดัดแปลงหรือแก้ไขได้ สามารถประกาศใช้ได้ในส่วนประกาศต่างๆ ของรูปแบบ (model)

SIGNAL : หมายถึง object ประเภทหนึ่งที่สามารถกำหนดค่าที่สัมพันธ์กับเวลาให้ได้นั้น หมายความว่า SIGNAL สามารถรับค่าได้เพียงค่าเดียวเท่านั้นในขณะเวลาหนึ่ง SIGNAL จะรับค่าหนึ่งได้จากขับสัญญาณหรือ driver ซึ่งตัวขับนี้อาจจะเก็บค่าในอนาคตสำหรับ SIGNAL ไว้ด้วย SIGNAL สามารถประกาศใช้ได้ในส่วนที่เป็น sequential body เท่านั้น ดังนั้น SIGNAL จึงสามารถถูกนำไปใช้ตลอดโครงสร้างของรูปหรือที่เรียกว่า global object

VARIABLE : หรือตัวแปร ได้แก่ object ที่สามารถกำหนดค่าใดๆ ให้ได้และสามารถที่จะเปลี่ยนแปลงค่าได้ตลอดการจำลองการทำงาน แต่จะเก็บค่าเพียงค่าเดียวเท่านั้นในขณะเวลาหนึ่ง เนื่องจาก VARIABLE สามารถประกาศใช้ได้ในส่วนที่เป็น sequential body เท่านั้นอันได้แก่ส่วนประกอบของ PROCESS, FUNCTION หรือ PROCEDURE ดังนั้น VARIABLE จึงสามารถนำไปใช้ได้เฉพาะในขอบเขตที่ถูกประกาศใช้เท่านั้น (local object)

ประเภทของ object ที่กำหนดไว้แล้ว (predefined type) : ได้แก่ TYPE ที่กำหนดไว้ใน package

ชื่อ STANDARD และกำหนดโดย IEEE ว่าจะต้องมีในระบบที่ใช้พัฒนา VHDL ฉะนั้นจึงไม่จำเป็นต้องเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ประกาศใช้ในทุกรูปแบบที่เขียนขึ้น TYPE ประเภทนี้ได้แก่

- 1) BOOLEAN คือกลุ่มของค่า FALSE และ TRUE
- 2) BIT คือกลุ่มของค่า '0' และ '1'
- 3) INTEGER คือกลุ่มของค่า -2147483647 ถึง 2147483647
- 4) REAL คือกลุ่มของค่า -1.0E38 ถึง 1.5E38
- 5) CHARACTER คือกลุ่มของค่าพัญชนะ 'A' - 'Z', 'a' - 'z' อักษรหรือเครื่องหมายพิเศษ และตัวอักษรควบคุม
- 4) TIME ได้แก่หน่วยเวลาที่มีค่าพื้นฐานเป็นวินาที (second ย่อด้วย s หรือ S)
- 5) SEVERITY LEVEL คือกลุ่มของค่า NOTE, WARNING, ERROR, FAILURE ส่วนต่างในการเขียน VHDL

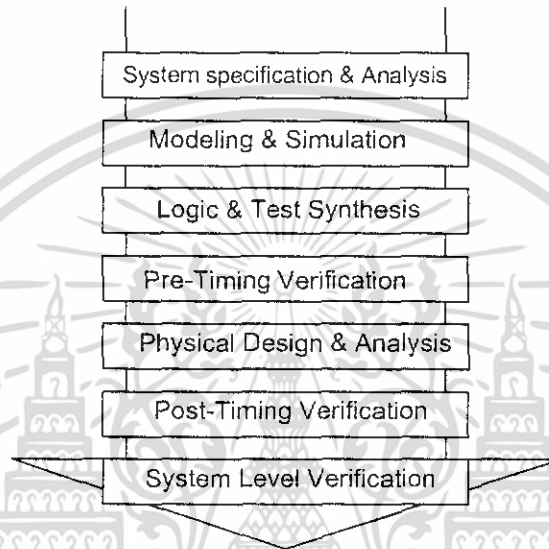
การเขียนรูปแบบหรือ modeling ด้วยภาษา VHDL มีความจำเป็นที่จะต้องแนะนำให้ผู้รู้จักกับส่วนต่าง ๆ ของแบบ (design units) ที่ใช้ภาษาเสียก่อน และนี่ก็เป็นขั้นตอนแรกที่สำคัญที่สุดของการศึกษาศึกษาเรียนรู้การใช้ภาษา VHDL เขียนรูปแบบบรรยายระบบดิจิทัลในมุมมองของการออกแบบลักษณะ Top-Down Design นอกจากนั้นการที่จะเข้าใจในเรื่องของ โครงสร้าง และส่วนต่าง ๆ ของรูปแบบ VHDL ให้ถูกต้องเสียก่อน

การออกแบบวงจรเชิงเลข (Digital Circuit) นั้น ในปัจจุบันก้าวหน้าไปอย่างมากโดยการใช้ภาษาบรรยายการทำงานของวงจร (Hardware Description Language : HDL) ซึ่งเป็นภาษาที่ใช้สำหรับออกแบบฮาร์ดแวร์ โดยภาษาที่เป็นมาตรฐานสากลเช่น Verilog หรือ VHDL (VHSIC Hardware Description Language (VHSIC : Very High Speed Integrated Circuit)) หรือภาษาที่ไม่เป็นมาตรฐานเช่น AHDL (Altera Hardware Description Language) หรือ PHDL (Philips Hardware Description Language) เป็นต้น มาบรรยายการทำงานของวงจรที่ได้ออกแบบไว้ ซึ่งในวิทยานิพนธ์นี้ได้ใช้ภาษา VHDL มาทำการออกแบบวงจร Digital Oscillator ทำให้ลดความยุ่งยากในการนำเอาอุปกรณ์มาเชื่อมต่อให้เป็นวงจรรวมทั้งลดเวลาที่ใช้ในการออกแบบและทดสอบการทำงาน ซึ่งมีความแตกต่างเป็นอย่างมากเมื่อเปรียบเทียบกับการออกแบบในอดีตที่ผ่านมา คือผู้ออกแบบจะต้องนำเอาอุปกรณ์แต่ละตัวที่ทำการออกแบบไว้ มาทำการต่อทดสอบในแผงวงจรจริง และทำการทดสอบวงจรเพื่อหาข้อผิดพลาด ซึ่งต้องใช้เวลานานกับการแก้ปัญหาแต่ละอย่างที่เกิดขึ้น แต่ในการออกแบบด้วยภาษา VHDL ผู้ออกแบบเพียงแค่เขียนซอร์สโค้ด (Source Code) บรรยายการทำงานของวงจร หลังจากนั้นก็ทำการคอมไพล์ (Compile) แล้วจำลองการทำงาน (Simulate) ดูว่าได้ฟังก์ชันการทำงานและไทม์มิ่ง (Timing) ตามที่ต้องการหรือไม่จากนั้นก็นำซอร์สโค้ดที่ได้ไปทำการสังเคราะห์ด้วยโปรแกรมสังเคราะห์ (Synthesis Tool) สุดท้ายนำวงจรที่ได้จากการสังเคราะห์ไปทำการแมป (Map) ลงไปยัง FPGA (Field Programmable Gate Array) เพื่อเป็นชิป (Chip) ต้นแบบสำหรับการนำไปทดสอบการทำงาน

2.2.2 การออกแบบจากบนลงล่าง

ในการพัฒนางจรรวมเชิงเลขขนาดใหญ่ที่มีความซับซ้อน ผู้ออกแบบมักจะมองการออกแบบให้อยู่ในรูปของบล็อกไดอะแกรมก่อน จากนั้นจึงวิเคราะห์ให้ลึกถึงรายละเอียดต่อไป ซึ่งภาษา VHDL นั้นอนุญาตให้อธิบายการทำงานของแต่ละบล็อก และวิเคราะห์การทำงาน แก้ไขและปรับปรุงการทำงานจากเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น เมื่ออนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ผลที่วิเคราะห์ เพื่อให้ได้การทำงานตามที่ต้องการ โดยการออกแบบในลักษณะนี้เรียกว่า หลักการออกแบบจากบนลงล่าง (Top-Down Design) ซึ่งถ้าเปรียบเทียบกับวิธีการออกแบบจากล่างขึ้นบน (Bottom-Up Design) จะเห็นได้ว่าการออกแบบจากล่างขึ้นบนจะใช้เวลาในการออกแบบมากกว่าเพราะเป็นการวาดวงจรด้วยอุปกรณ์ต่างๆ (Schematic Capture) ที่ประกอบกันเข้าเป็นวงจรที่ต้องการออกแบบ จำลองการทำงาน ตรวจสอบความถูกต้อง ซึ่งใช้เวลามาก และถ้าวงจรที่ต้องการออกแบบมีความซับซ้อนก็จะเป็นเรื่องที่ยากมากในการออกแบบลักษณะนี้ ดังนั้นการใช้ภาษา VHDL กับหลักการออกแบบจากบนลงล่างจึงเป็นวิธีการที่เหมาะสมสำหรับการออกแบบและพัฒนาวงจรที่มีความซับซ้อนมากขึ้น ทั้งยังช่วยลดเวลาและค่าใช้จ่ายในการออกแบบ



รูปที่ 2.2 แสดงขั้นตอนการออกแบบจากบนลงล่าง

จากรูปที่ 2.2 แสดงให้เห็นถึงขั้นตอนการออกแบบจากบนลงล่างทั้งนี้ในทางปฏิบัติอาจจะมีข้อแตกต่างไปจากนี้บ้างเล็กน้อย โดยขั้นตอนของการออกแบบจากบนลงล่างมีรายละเอียด ดังนี้

1. ขั้นตอนการสร้างข้อกำหนดของความต้องการ และวิเคราะห์ระบบ เพื่อหาแนวความคิดและหลักการ (Idea and Concept) ในการแก้ปัญหา
2. ขั้นตอนการเขียนรูปแบบของระบบที่ต้องการออกแบบ โดยใช้ภาษา VHDL สำหรับ บรรยายพฤติกรรมการทำงาน พร้อมทั้งจำลองการทำงาน เพื่อเปรียบเทียบและตรวจสอบความถูกต้องกับข้อกำหนด
3. ขั้นตอนการสังเคราะห์ซึ่งจะต้องทำการกำหนดเทคโนโลยีที่จะมารองรับวงจรที่ออกแบบและระบบช่วยออกแบบจะทำการสังเคราะห์วงจรที่ได้จากรูปแบบที่เขียนขึ้นให้อยู่ในรูปของวงจรที่ประกอบด้วยอุปกรณ์อิเล็กทรอนิกส์ หรือวงจร ในระดับเกต (Gate Level) และ การเชื่อมต่อกันของอุปกรณ์เหล่านั้น หรือ ไม่ก็อยู่ในรูปของเน็ตลิสต์ (Net list) ที่สามารถนำไปผลิตลงบนอุปกรณ์อื่นได้
4. หลังจากการสังเคราะห์วงจรให้อยู่ในระดับเกตหรือเน็ตลิสต์แล้ว ข้อมูลที่ได้นอกจากจะเป็นข้อมูลสำหรับจำลองการทำงานในเรื่องของความถูกต้องของฟังก์ชันแล้วยังมีข้อมูลที่เกี่ยวข้องกับเวลาด้วย ซึ่งจากความจริงที่ว่า อุปกรณ์อิเล็กทรอนิกส์ทุกชิ้นจะมีเวลาหน่วงของการเคลื่อนผ่าน

(Propagation Delay time) เสมอ ถึงแม้ว่าจะเป็นเวลาที่น้อยมากในระดับ นาโนวินาที แต่ถ้าภายในเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาก็เท่านั้น เมื่อผู้ดูแลเห็นว่าเป็นประโยชน์ในการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

วงจรหนึ่งประกอบด้วยเกทของฟังก์ชันต่างๆจำนวน 10,000 เกท ขึ้นไป เวลา ดังกล่าวนี้จะสะสม กันมากขึ้นจนอาจจะทำให้การทำงานของวงจรทั้งหมดผิดไป หรือไม่สามารทำงานในย่านความถี่สัญญาณพาหะสูงๆได้

5. ขั้นตอนของการผลิตเป็นวงจรรจริง (Technology and Device Mapping) โดยนำข้อมูลที่ได้ จากการสังเคราะห์มาผลิต ซึ่งอาจจะอยู่ในรูปของอุปกรณ์ FPGA หรือวงจรรวม ASIC
6. หลังจากที่ได้อวงจรจริงมาแล้วยังต้องมีความจำเป็นที่จะต้องตรวจสอบการทำงานที่ค่านิ่งถึง เวลาด้วย เพื่อความถูกต้องของวงจรครั้งสุดท้ายก่อนที่จะนำไปรวมเข้ากับอุปกรณ์อื่นๆ ให้เป็นระบบ เพราะในขั้นตอนนี้วงจรที่ออกแบบจะประกอบด้วยอินพุทและเอาต์พุทแพด (Pad) ซึ่งเป็นจุดต่อสำหรับรับและส่งสัญญาณกับภายนอก
7. หลังจากที่นำวงจรที่ออกแบบรวมเข้ากับอุปกรณ์อื่นๆให้เป็นระบบแล้วนั้น จะต้องทดสอบการทำงานรวมทั้งระบบร่วมกับอุปกรณ์อื่นๆอีกครั้ง ซึ่งเป็นการทดสอบการทำงานจริงครั้งสุดท้าย

2.2.3 ภาษา VHDL และ ส่วนประกอบต่างๆของภาษา

วิวัฒนาการของภาษา VHDL นั้นเริ่มต้นประมาณปี ค.ศ. 1981 โดยที่กระทรวงกลาโหมสหรัฐอเมริกา หรือ DOD (Department of Defence) ได้ทำการพัฒนาโครงการที่มีชื่อว่า VHSIC ซึ่งเป็นการพัฒนาโปรแกรมซึ่งจัดเป็นภาษาระดับสูงเช่นเดียวกับภาษา C หรือ Pascal แต่สามารถบรรยายพฤติกรรมการทำงานของวงจรเชิงเลข หรือ โครงสร้างของวงจรได้ ทั้งนี้เพื่อให้สามารถออกแบบและสร้างวงจรรวมได้รวดเร็วขึ้น

ในการเขียนรูปแบบบรรยายระบบเชิงเลขในลักษณะของการออกแบบจากบนลงล่างจะต้องทำความเข้าใจในเรื่องของโครงสร้างและส่วนประกอบต่างๆของรูปแบบภาษา VHDL เสียก่อนซึ่งส่วนประกอบที่สำคัญและเป็นพื้นฐานของการเขียนมี 4 หน่วย คือ

- หน่วยการออกแบบเอนทิตี (Entity Design unit)
- หน่วยการออกแบบสถาปัตยกรรม (Architecture Design unit)
- หน่วยการออกแบบแพ็คเกจ (Package Design unit)
- หน่วยการออกแบบโครงแบบ (Configuration Design unit)

2.2.3.1 หน่วยการออกแบบเอนทิตี

หน่วยการออกแบบนี้เป็นส่วนที่ใช้สำหรับติดต่อกันระหว่างภายนอกกับรูปแบบที่เขียนขึ้นโดยเป็นการกำหนดจุดเชื่อมต่อของรูปแบบ กำหนดทิศทางการไหลของสัญญาณ และประเภทของค่าที่สามารถกำหนดให้กับสัญญาณตามจุดต่างๆของข้อมูลที่ไหลผ่านจุดต่อเหล่านั้น รูปที่ 2.3 แสดงให้เห็นถึงโครงสร้างของหน่วยการออกแบบเอนทิตี

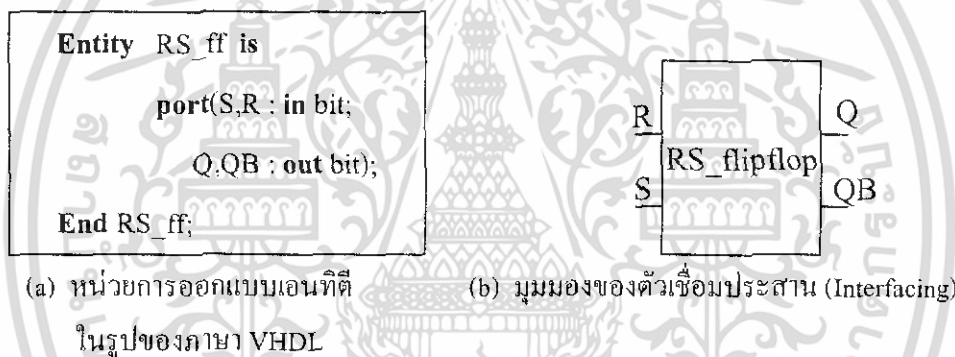
```

Entity component_name is
    Input and Output ports
    Physical and other parameters
End [component_name];

```

รูปที่ 2.3 แสดงโครงสร้างโดยทั่วไปของหน่วยการออกแบบเอนทิตี

ส่วนนี้จะขึ้นต้นด้วยคำว่า Entity และ is ระหว่างคำทั้งสองคำเป็นส่วนสำหรับชื่อของรูปแบบที่ต้องการจะเขียน(component name)หลังจากนั้นจะตามด้วยส่วนที่ใช้กำหนดช่องทางเข้าและออกของข้อมูล(input-output) รวมทั้งพารามิเตอร์อื่นๆและที่สำคัญคือหน่วยการออกแบบเอนทิตีจะต้องปิดท้ายด้วยคำว่าEndและเครื่องหมายอัฒภาคเสมอ (;)



รูปที่ 2.4 แสดงรูปแบบของ RS_flipflop

ในรูปที่ 2.4 เป็นหน่วยการออกแบบเอนทิตีที่บรรยายอุปกรณ์ชื่อ RS_flipflop ในส่วนหัวของเอนทิตีมีการกำหนดจุดต่อ 4 จุด ภายใต้ชุดคำสั่ง port โดยที่ 2 จุดแรกเป็นจุดให้ข้อมูลไหลผ่านเข้า ได้แก่ R, S ซึ่งกำหนดด้วยทิศทาง การติดต่อกับโลกภายนอกเป็นการไหลเข้าของข้อมูล (in) ส่วนจุดเอาต์พุตเป็นจุดให้ข้อมูลไหลออก ได้แก่ Q,QB ซึ่งกำหนดด้วยทิศทาง การติดต่อกับโลกภายนอกเป็นการไหลออก (out) ส่วนประเภทของข้อมูลที่จะไหลเข้าและออกนั้นเป็นประเภท bit ที่สามารถมีค่าได้เพียงสองค่าเท่านั้น คือ “0” และ “1” เท่านั้น

2.2.3.2 หน่วยการออกแบบสถาปัตยกรรม

หน่วยการออกแบบสถาปัตยกรรมคือส่วนที่ใช้เขียนบรรยายพฤติกรรมของรูปแบบในมุมมองของการจำลองการทำงาน พฤติกรรมต่างๆที่บรรยายในส่วนนี้ขึ้นอยู่กับข้อมูลที่ผ่านเข้าและออกตรงช่องทาง ตลอดจนพารามิเตอร์ต่างๆที่กำหนดในหน่วยการออกแบบเอนทิตี รูปที่ 2.5 แสดงให้เห็นถึง

โครงสร้างของหน่วยการออกแบบสถาปัตยกรรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

Architecture identifier of component_name is
    [declaration]

Begin
    Specification of the functionality
    of the component in terms of its
    input lines and as influenced by
    physical and other parameters

End [identifier];

```

รูปที่ 2.5 แสดงโครงสร้างโดยทั่วไปของหน่วยการออกแบบสถาปัตยกรรม

ส่วนของหน่วยการออกแบบสถาปัตยกรรมเริ่มต้นด้วยคำว่า Architecture และตามด้วยชื่อ (identifier) สิ่งที่ต้องกำหนดลงไปได้แก่ สิ่ง que แสดงให้เห็นว่า Architecture นั้นใช้บรรยายหน่วยการออกแบบเอนทิตีใดๆ (of <entity design unit> is) ส่วนที่อยู่ระหว่าง Architecture และ Begin เป็นพื้นที่ส่วนประกาศหน่วยของสถาปัตยกรรมกำหนด (Architecture declaration area) ที่เป็นส่วนเลือเลือก (Option) ในบริเวณนี้สามารถใช้เขียนประกาศกำหนดค่าต่างๆที่จะนำไปใช้ภายในสถาปัตยกรรมนั้นได้ อาทิเช่น ประเภท (Type) ต่างๆ (ตัวอย่างเช่น bit, bit vector), สัญญาณ (signal), ค่าคงที่ (constant), โปรแกรมย่อย (ได้แก่ function และ procedure) และอุปกรณ์ (component) ส่วนที่ใช้บรรยายความสัมพันธ์ระหว่างข้อมูลที่ไหลเข้าและไหลออกของรูปแบบ (สัญญาณที่กำหนดในชุดคำสั่ง port) นั้นจะถูกบรรยายในบริเวณเนื้อที่ระหว่างคำว่า Begin กับ End ของหน่วยการออกแบบสถาปัตยกรรม และนอกจากนั้นชุดคำสั่งทุกคำสั่งที่อยู่ภายในบริเวณนี้จะเป็นชุดคำสั่งแบบแข่งขนาน (Concurrent statement) เท่านั้น คือทุกๆ statement จะทำงานพร้อมกัน ลำดับก่อนหลังจะ ไม่มีผลต่อการทำงานของรูปแบบ หน่วยการออกแบบสถาปัตยกรรมจะต้องปิดท้ายด้วยคำสั่ง End และชื่อของสถาปัตยกรรมนั้นๆ โดยทั่วไปการเขียนรูปแบบระบบเชิงเลขด้วยภาษา VHDL สามารถเขียนได้ในลักษณะต่างๆ ดังนี้

- ลักษณะการไหลของข้อมูล (Dataflow style)
- ลักษณะพฤติกรรม (Behavioral style)
- ลักษณะ โครงสร้าง (Structural style)
- ลักษณะผสม (Mixed Model style)

รูปที่ 2.6 ส่วนที่บรรยายความสัมพันธ์ระหว่างข้อมูลที่ไหลเข้า (R, S) กับข้อมูลที่ไหลออก (Q, QB) ประกอบด้วยชุดคำสั่งแบบแข่งขนาน 2 ชุด ซึ่งเขียนเป็นประเภทการไหลของข้อมูล หรือเรียกว่า ระดับการถ่ายโอนข้อมูลระหว่างรีจิสเตอร์ (RTL : Register Transfer Level)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

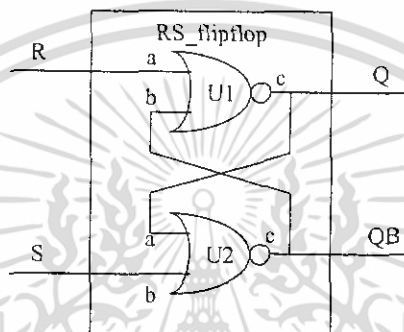
```

Architecture dataflow of RS_ff is
  Begin
    Q <= not(QB or R);
    QB <= not(Q or S);
  End dataflow;

```

รูปที่ 2.6 แสดงหน่วยการออกแบบสถาปัตยกรรมของ RS_flipflop

ตามฟังก์ชันบูลีน $Q = \overline{QB + R}$ และ $QB = \overline{Q + S}$



รูปที่ 2.7 แสดงโครงสร้างภายในสถาปัตยกรรมของ RS_flipflop

รูปที่ 2.7 เป็นหน่วยการออกแบบสถาปัตยกรรมของ RS_flipflop ในลักษณะโครงสร้างซึ่งเปรียบเสมือนการนำอุปกรณ์ที่มีอยู่ในไลบรารี (Library) มาต่อเป็นวงจรตามต้องการ โดยใช้พอร์ต 2 อินพุต (nor2) จำนวนสองตัวมาสร้างตามฟังก์ชันบูลีน

```

Architecture struc of RS_ff is
  component nor2
    port(a,b : in bit;
         c :out bit);
  end component;

  Begin
    U1 : nor2 port map(R,QB,Q);
    U2 : nor2 port map(S,Q,QB);
  End struc;

```

รูปที่ 2.8 แสดงหน่วยการออกแบบสถาปัตยกรรมของ RS_flipflop ในลักษณะโครงสร้าง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

Architecture behave of RS_ff is

  Begin
    process(R,S)
      begin
        if R='0' and S='1' then
          Q <= '1';
          QB <= '0';
        elsif R='1' and S='0' then
          Q <= '1';
          QB <= '0';
        end if;
      end process;
    End behave;

```

รูปที่ 2.9 แสดงหน่วยการออกแบบสถาปัตยกรรมของ RS_flipflop ในลักษณะพฤติกรรม

รูปที่ 2.9 เป็นการเขียนบรรยายการทำงานของรูปแบบในลักษณะพฤติกรรม ซึ่งจะเห็นได้ว่ามีลักษณะที่เหมือนกับการเขียนโปรแกรมทั่วไป โดยจะต้องมีการใช้งานส่วนที่เรียกว่า process และการทำงานของรูปแบบจะขึ้นอยู่กับเปลี่ยนแปลงของสิ่งที่อยู่ภายใน process (อินพุต R, S) ซึ่งเรียกว่า Sensitivity list การเขียนในลักษณะนี้ลำดับก่อนหลังของชุดคำสั่งจะมีผลต่อการทำงานของรูปแบบที่เขียนขึ้น

```

Architecture mixed of RS_ff is

  component nor2
    port(a,b : in bit;
         c : out bit);
  end component;

  Begin
    U1 : nor2 port map(R,QB,Q);
    QB <= not(Q or S);
  End mixed;

```

รูปที่ 2.10 แสดงหน่วยการออกแบบสถาปัตยกรรมของ RS_flipflop ในลักษณะผสม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ไม่ว่าจะเขียนบรรยายส่วนของสถาปัตยกรรมของ RS_flipflop ในลักษณะของพฤติกรรม การไหลของข้อมูล โครงสร้าง หรือผสมที่นำเอาแต่ละลักษณะมาเขียนไว้ในส่วนของสถาปัตยกรรมก็ตามต่างก็มีพฤติกรรมเดียวกัน และจะให้ผลลัพธ์จากการจำลองการทำงานที่เหมือนกัน ซึ่งถือว่าเป็นข้อดีของภาษา VHDL

2.2.3.3 หน่วยการออกแบบแพ็คเกจ

ข้อมูลต่างๆ ตลอดจนโปรแกรมย่อยที่เป็นประโยชน์ต่อการเขียนรูปแบบบรรยายระบบเชิงเลขสามารถเก็บไว้ในส่วนของแพ็คเกจได้ และข้อมูลเหล่านี้สามารถเรียกไปใช้ได้โดยหน่วยการออกแบบ เอนทิตี หน่วยการออกแบบสถาปัตยกรรมหรือจากหน่วยการออกแบบแพ็คเกจอื่นๆ โดยปกติแล้วแพ็คเกจจะแบ่งออกเป็น 2 ส่วน คือ การประกาศแพ็คเกจ (Package Declaration) และส่วนของบอดี้แพ็คเกจ (Package Body) เนื่องจากแพ็คเกจถูกสร้างขึ้นเป็นส่วนแยกต่างหาก ออกจากรูปแบบที่กำลังเขียนอยู่ ฉะนั้นการที่จะนำแพ็คเกจไปใช้นั้นจะต้องมีการเชื่อมโยงหรืออ้างอิงเสียก่อน ซึ่งในภาษา VHDL สามารถทำได้ด้วยชุดคำสั่ง USE

Package Declaration

ส่วนที่มีความสำคัญที่สุดของแพ็คเกจ(ถ้ามองในแง่ของการนำไปใช้จากภายนอก)ได้แก่ส่วนการประกาศแพ็คเกจ เพราะจะเป็นส่วนที่กำหนดชื่อของสิ่งที่ประกาศอยู่ในแพ็คเกจสำหรับนำไปใช้ภายนอกตัวของแพ็คเกจเองสิ่งใดๆที่ถูกระบุไว้ในส่วนของบอดี้แพ็คเกจแต่ไม่ได้ถูกระบุไว้ในส่วนการประกาศแพ็คเกจจะไม่สามารถถูกนำค่าและพฤติกรรมไปใช้ส่วนนอกได้ ซึ่งสามารถเปรียบเทียบได้กับสิ่งที่ประกาศไว้ในส่วนของการประกาศเอนทิตีคือจุดเชื่อมต่อหรือพอร์ตที่มีหน้าที่ติดต่อกับโลกภายนอก ฉะนั้นโดยทั่วไปแล้วแพ็คเกจสามารถสร้างขึ้นได้โดยไม่จำเป็นต้องมีส่วนบอดี้และยังสามารถถูกนำไปใช้จากรูปแบบภายนอกได้ เช่น ใช้สำหรับประกาศชนิด (Type) หรือสัญญาณเช่นเดียวกันกับส่วนบอดี้แพ็คเกจที่ไม่จำเป็นต้องมีส่วนของการประกาศแพ็คเกจ แต่แพ็คเกจนั้นจะไม่สามารถถูกนำไปใช้จากรูปแบบอื่นได้

```
Package package_name is
    Package_declaration_part
End package_name;
```

รูปที่ 2.11 แสดงโครงสร้างโดยทั่วไปของส่วนการประกาศแพ็คเกจ

Package body

โครงสร้างที่ประกอบด้วยคำสั่งต่างๆ ในรูปของคำสั่งลำดับ (Sequence) ที่ใช้บรรยายฟังก์ชันการทำงานของโปรแกรมย่อย (Subprogram)ทั้งหลายที่ชื่อของโปรแกรมย่อยนั้นๆ ที่ถูกระบุไปในส่วนของการประกาศแพ็คเกจแล้วจะถูกรวบรวมไว้ในส่วนบอดี้แพ็คเกจทั้งนี้รวมทั้งการกำหนดค่าคงที่ต่างๆอันได้แก่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ค่าคงที่ที่ถูกประกาศชื่อก่อนในส่วนของการประกาศแพ็คเกจแต่ถูกกำหนดค่าในส่วนของบอดีแพ็คเกจ ฉะนั้นส่วนบอดีแพ็คเกจจึงไม่จำเป็นต้องมีถ้าในส่วนของการประกาศแพ็คเกจไม่มีการประกาศชื่อที่เป็น โปรแกรมย่อยหรือค่าคงที่ การเขียนบอดีแพ็คเกจนั้นเป็นไปตามกฎเกณฑ์ที่แสดงในรูปที่ 2.12

```

Package body package_name is
    declarative part
End package_name;

```

รูปที่ 2.12 แสดงโครงสร้างโดยทั่วไปของบอดีแพ็คเกจ

2.2.3.4 หน่วยการออกแบบโครงแบบ

ดังที่ทราบกันแล้วว่ารูปแบบหนึ่งของระบบดิจิทัล ไม่ว่าจะเป็นอะไร จะมีหน่วยการออกแบบ เอนทิตีได้ เพียงหนึ่งหน่วยเท่านั้น แต่ในขณะที่ หน่วยการออกแบบเอนทิตี หนึ่งหน่วยนี้อาจจะมี สถาปัตยกรรม ที่เป็นหน่วยรอง ได้หลายหน่วย ดังนั้นจะต้องมีหน่วยการออกแบบโครงแบบมาเพื่อกำหนดการใช้โครงแบบ (Configuration) ประกอบเอนทิตีกับหน่วยการออกแบบสถาปัตยกรรมหน่วยไหนเข้าด้วยกัน

```

Configuration identifier of entity_name is
    Configuration_declarative_part
End;

```

รูปที่ 2.13 แสดงโครงสร้างโดยทั่วไปของหน่วยการออกแบบโครงแบบ

2.2.4 ชุดคำสั่งลำดับ (Sequential Statements)

ภาษา VHDL สามารถใช้เขียนรูปแบบ (modeling) บรรยายระบบดิจิทัลในลักษณะของ behavioral description ที่โครงสร้างภายในประกอบด้วย sequential statement การศึกษาในรายละเอียดของโครงสร้างดังกล่าว สำหรับ software engineering ที่มีความคุ้นเคยกับการเขียน โปรแกรมด้วยภาษาชั้นสูง อาทิเช่น C หรือ PASCAL อยู่ก่อนแล้ว จะสามารถเข้าใจโครงสร้างแบบ sequential ได้ง่าย เพียงแต่ต้องทำความเข้าใจเกี่ยวกับลักษณะการทำงานของ hardware เพิ่มเติม ในภาษา VHDL มีคำสั่งดังต่อไปนี้

- WAIT statement
- VARIABLE assignment
- Signal assignment
- IF-THEN-ELSE statement
- CASE statement
- Loops
- NEXT statement

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- EXIT statement
- RETURN statement
- NULL statement
- Procedure call
- ASSERTION statement

จะกล่าวเฉพาะ statement ที่สำคัญๆ เพื่อความเข้าใจในการทำงานแบบ sequential เท่านั้น ตามที่เคยกล่าวมาแล้วว่าภาษา VHDL เป็นภาษาที่มีคุณสมบัติเป็นแบบแข่งขันกันนั้น คือ ชุดคำสั่งภายในตัวโครงสร้างจะเป็นชุดคำสั่งแบบแข่งขันกัน เช่นเดียวกับภาษา ADA ชุดคำสั่งลำดับหรือ sequential statement ที่เรียกว่า อันได้แก่ process statement

2.2.4.1. Process Statement

หัวใจสำคัญของ Concurrent shell ที่ทำให้สามารถเขียน VHDL model เพื่อบรรยายพฤติกรรมของระบบดิจิทัลอิเล็กทรอนิกส์ในลักษณะ behavioral description ได้แก่คำสั่ง process ที่โครงสร้างภายในจะประกอบด้วยชุดคำสั่งแบบลำดับเท่านั้น ชุดคำสั่งเหล่านี้จะทำงานเป็นลำดับจากบนลงล่าง เมื่อ PROCESS ถูกกระตุ้นให้ทำงาน

คำ PROCESS ในบรรทัดแรกของโครงสร้าง แสดงถึงจุดเริ่มต้นของชุดคำสั่ง process ในบางกรณีใน architecture หนึ่งอาจจะมีโครงสร้างของชุดคำสั่ง process หลายชุดได้ คำ END PROCESS บอกถึงจุดสิ้นสุดของชุดคำสั่ง process เมื่อคำสั่ง END PROCESS ถูกปฏิบัติแล้วชุดคำสั่ง process จะหยุดการทำงานลงชั่วขณะ (แต่ยังคง active อยู่ตลอดเวลา) จนกว่าจะมีสัญญาณอย่างน้อยตัวหนึ่งใน sensitivity list เกิด event ขึ้นอีก ชุดคำสั่ง process เป็นชุดคำสั่งแบบแข่งขันกัน นั้นหมายความว่า โดยปกติแล้วชุดคำสั่ง process จะทำงานตลอดเวลา การที่ชุดคำสั่งทั้งหลายอยู่ภายในบล็อกของชุดคำสั่ง process เป็นชุดคำสั่งแบบลำดับ นั้นถ้าชุดคำสั่ง process ใดที่ไม่มี sensitivity list เป็นตัวควบคุมการทำงาน จะทำให้เกิดการทำงานที่เปรียบเสมือนว่าเป็นวงรอบ (loop) ที่ไม่รู้จบขึ้น วิธีการป้องกันการเกิดเหตุการณ์เช่นนี้ คือการเติม wait statement ลงในส่วนของชุดคำสั่ง process

2.2.4.2. Wait Statement

ชุดคำสั่ง process สามารถมี sensitivity list ได้เพียงอันเดียว หมายความว่าชุดคำสั่ง process จะถูกกระตุ้นได้จากการที่สัญญาณหนึ่งในรายชื่อที่เกิด event ขึ้นเท่านั้น หลังจากที่ถูกระตุ้นแล้วคำสั่งทั้งหลายที่อยู่ภายในจะทำงานแบบลำดับลงมาจนกระทั่งหมด และชุดคำสั่ง process จะหยุดการทำงานชั่วคราวจนกว่าจะมี event เกิดขึ้นอีกกับสัญญาณตัวใดตัวหนึ่งในรายชื่อนั้นอีก ถ้ากรณีที่อยู่ในรายชื่อประกอบด้วยสัญญาณหลายตัว และเกิด event ขึ้นในเวลาเดียวกัน จะมีสัญญาณเพียงตัวเดียวจากทั้งหมดเท่านั้น ที่กระตุ้นการทำงาน ของชุดคำสั่ง process ซึ่งไม่สามารถที่จะบอกได้ว่าเป็นตัวใด ดังนั้นการใช้ชุดคำสั่ง process ร่วมกับ sensitivity list จึงมีขีดจำกัดอยู่มากในภาษา VHDL มีวิธีการหลีกเลี่ยงปัญหา เช่นนี้โดยใช้ชุดคำสั่ง wait statement

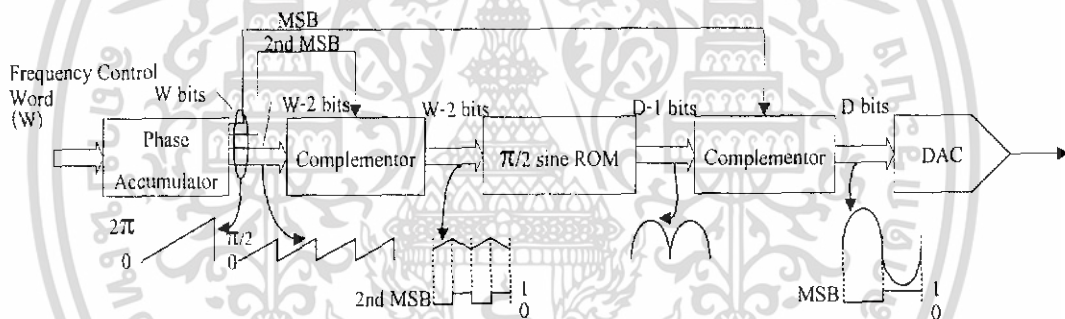
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

นัยสำคัญสูงสุด (MSB) หรือเรียกว่าบิตเอ็มเอสบี 2 บิตแรกมากำหนดควอดแรนต์ของสัญญาณขาขึ้นที่จะสังเคราะห์ ความสำคัญของบิตเอ็มเอสบีกับควอดแรนต์และค่าแอมพลิจูดสัญญาณขาขึ้นเราสามารถสรุปได้ดังตารางที่ 2.2 และ รูปที่ 2.14 ตามลำดับ

ตารางที่ 2.2 ความสัมพันธ์ของบิตเอ็มเอสบีกับควอดแรนต์และสัญญาณขาขึ้น

Quadrant	MSB	MSB-1	Sine
Quadrant I	0	0	$\frac{1}{2} + \frac{1}{2} \sin(I)$
Quadrant II	0	1	$\frac{1}{2} + \frac{1}{2} \sin(I \text{ complemented})$
Quadrant III	1	0	$\frac{1}{2} - \frac{1}{2} \sin(I \text{ complemented})$
Quadrant IV	1	1	$\frac{1}{2} - \frac{1}{2} \sin(I)$

* I เป็นดัชนี (Index) เริ่มจาก 0 ถึง $2^{W-2}-1$



รูปที่ 2.14 การสังเคราะห์สัญญาณขาขึ้นเต็มรูปด้วยเทคนิคบีบอัดควอดแรนต์

จากรูป 2.14 หน่วยความจำ (ROM) จะเก็บแอมพลิจูดของสัญญาณขาขึ้นในช่วง 0 ถึง $\pi/2$ บิตเอ็มเอสบี 2 บิต บนจะถูกใช้สำหรับกำหนดควอดแรนต์ของสัญญาณขาขึ้นและบิตที่เหลือ $W-2$ บิต จะถูกใช้สำหรับชี้ตำแหน่ง ของหน่วยความจำเพื่อดึงแอมพลิจูดสัญญาณขาขึ้นที่ต้องการออกมา บิตเอ็มเอสบีบนสุด (MSB) ทำหน้าที่กำหนดเครื่องหมาย (Sine) ของสัญญาณเอาต์พุตและบิตเอ็มเอสบีที่สอง (2^{th} MSB) กำหนดว่าแอมพลิจูดของสัญญาณเอาต์พุตเป็นขาขึ้นหรือขาลงกล่าวคือในควอดแรนต์ที่ 1 และ 3 สัญญาณเอาต์พุตของสัญญาณสร้างเฟส (Phase accumulator) จะถูกนำมาใช้โดยตรงส่วนควอดแรนต์ที่ 2 และ 4 นั้นสัญญาณเอาต์พุตของวงจรสร้างเฟสจะถูกคอมพลิเมนต์ (Complemented) สัญญาณแอมพลิจูดเอาต์พุตที่ออกจากหน่วยความจำจะมีลักษณะเป็นสัญญาณเต็มลูกคลื่น (Full wave) ซึ่งสามารถเปลี่ยนเป็นสัญญาณขาขึ้นโดยการคูณแอมพลิจูดของสัญญาณที่อยู่ในช่วง π ถึง 2π ด้วย -1

จะเห็นได้ว่าวิธีการบีบอัดควอดเรนท์นี้สามารถใช้ข้อมูลของแอมพลิจูดของสัญญาณขาเข้าทั้งสี่ควอดเรนท์หรือสัญญาณเต็มลูกคลื่นได้ นั่นหมายถึงวิธีการบีบอัดควอดเรนท์นี้สามารถลดขนาดของหน่วย ความจำได้ถึง 75 เปอร์เซ็นต์

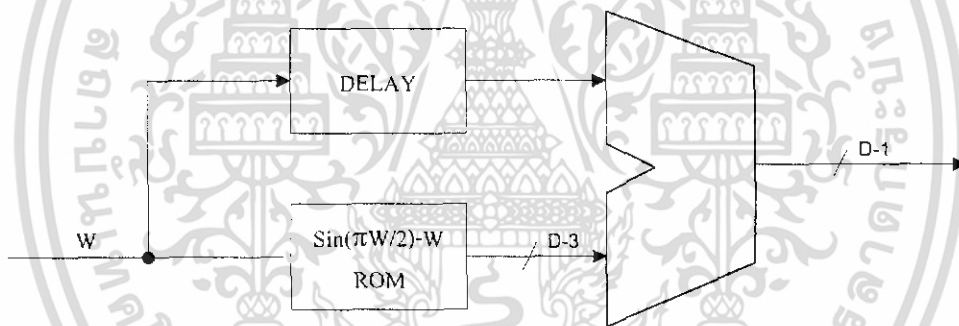
ด้วยเหตุนี้วิธีการบีบอัดควอดเรนท์จึงมักจะถูกนำมาใช้สำหรับออกแบบวงจรสังเคราะห์ความถี่แบบที่ใช้หน่วยความจำอยู่เสมอ ซึ่งในการออกแบบจริงๆแล้ว นอกจากการบีบอัดควอดเรนท์แล้วมักจะมีการบีบอัดข้อมูลเพิ่มเติมต่อไปอีก สรุปเป็นวิธีต่างๆได้ดังนี้

ขั้นตอนวิธีผลต่างไซน์-เฟส (Sine-phase difference algorithm)

ขั้นตอนวิธีผลต่างไซน์-เฟสนี้หน่วยความจำจะเก็บค่าของฟังก์ชัน $f(W)$ ซึ่งเป็นค่าผลต่างของแอม-พลิจูดไซน์กับค่าเฟสที่เฟสนั้นๆฟังก์ชัน $f(W)$ สามารถเขียนเป็นสมการได้ดังนี้

$$f(W) = \sin\left(\frac{\pi W}{2}\right) - W \quad (2.2)$$

สัญญาณขาเข้าที่ต้องการจะถูกสังเคราะห์โดยการนำค่าแอมพลิจูดจากหน่วยความจำมาบวกกับค่าของเฟสที่ต้องการสังเคราะห์ ดังแสดงในรูปที่ 2.15



รูปที่ 2.15 ขั้นตอนวิธีผลต่างไซน์-เฟส

วิธีการเก็บค่าของ $f(W)$ แทนการเก็บค่า $\sin(\pi W/2)$ สามารถลดจำนวนบิตของแอมพลิจูดได้ 2 บิต เพราะ

$$\max\left[\sin\left(\frac{\pi W}{2}\right) - W\right] \approx 0.21 \max\left[\sin\left(\frac{\pi W}{2}\right)\right] \quad (2.3)$$

จะเห็นว่าในขั้นตอนวิธีผลต่างไซน์-เฟสนี้จะมีฮาร์ดแวร์ของวงจร (Adder) สำหรับทำฟังก์ชัน

$$\left[\sin\left(\frac{\pi W}{2}\right) - W\right] + W \quad (2.4)$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากการที่ขั้นตอนวิธีผลต่างซายน์-เฟสสามารถลดจำนวนบิตของแอมพลิฟายเออร์ได้ นั่นคือขนาดของหน่วยความจำจะลดลง เป็นผลให้วงจรสังเคราะห์ความถี่จะสามารถทำงานได้เร็วขึ้น

2.3.1 ขั้นตอนวิธีซันเดอร์แลนด์ (Sunderland algorithm)

ขั้นตอนวิธีซันเดอร์แลนด์ ถูกพัฒนาต่อจากขั้นตอนวิธีฮัททิสัน (Hutchison algorithm) ขั้นตอนวิธีซันเดอร์แลนด์มีหลักการทำงานโดยแบ่งสัญญาณเฟสที่ได้รับจากรังสร้างเฟสออกเป็นสามส่วน (a, b, และ c) ดังนี้

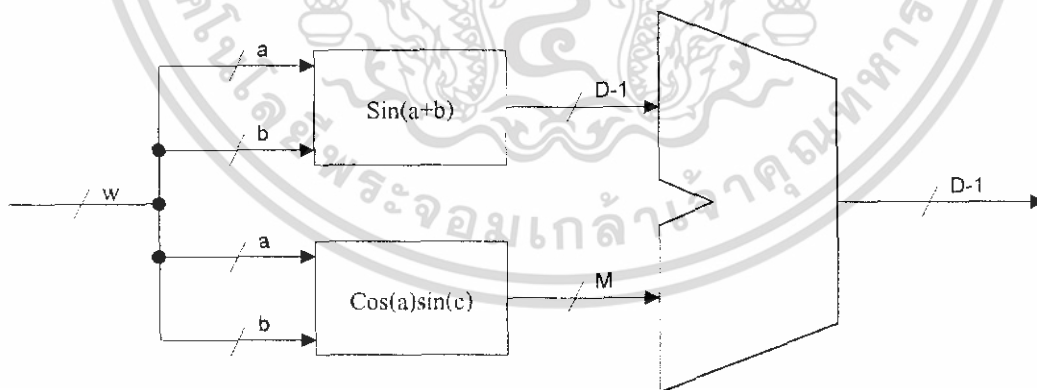
$$\sin(x) = \sin(a+b+c) \quad (2.5)$$

$$\sin(a+b+c) = \sin(a+b)\cos c + \cos a \cos b \sin c - \sin a \sin b \sin c \quad (2.6)$$

$$\sin(a+b+c) \approx \sin(a+b) + \cos a \sin b \quad (2.7)$$

และสัญญาณ $\sin(a+b+c)$ สามารถกระจายออกได้ดังสมการที่ (2.6) และประมาณค่าได้ดังสมการที่ (2.7) ตามลำดับ ซึ่งขั้นตอนวิธีซันเดอร์แลนด์นี้สามารถบีบอัดหน่วยความจำได้ประมาณ 11 เท่า

จากสมการที่ (2.7) เราสามารถนำขั้นตอนวิธีซันเดอร์แลนด์มาสร้างเป็นฮาร์ดแวร์ได้โดยแบ่งหน่วยความจำออกเป็นสองส่วนคือหน่วยความจำหยาบ (Coarse ROM) และหน่วยความจำละเอียด (Fine ROM) โดยหน่วยความจำหยาบจะเก็บค่าแอมพลิฟายเออร์ของ $\sin(a+b)$ และหน่วยความจำละเอียดจะเก็บค่าแอมพลิฟายเออร์ของ $\cos a \sin c$ ดังแสดงในรูปที่ 2.16



รูปที่ 2.16 ขั้นตอนวิธีซันเดอร์แลนด์

2.4 การสังเคราะห์ความถี่แบบดิจิทัลที่ไม่ใช้หน่วยความจำ

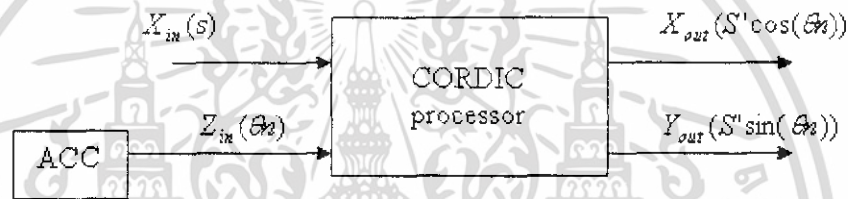
จากรายละเอียดการทำงานของ การสังเคราะห์ความถี่แบบดิจิทัลที่ใช้หน่วยความจำ ซึ่งมีข้อเสียสำคัญคือคุณภาพของสัญญาณจะขึ้นอยู่กับความละเอียดของแอมพลิฟายเออร์และหน่วยความจำกล่าวคือถ้าเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ต้องการสัญญาณที่มีคุณภาพสูงก็จำเป็นต้องใช้หน่วยความจำขนาดใหญ่เพื่อเก็บแอมพลิจูดของสัญญาณให้ได้มากและละเอียดที่สุด ซึ่งการใช้หน่วยความจำที่มีขนาดใหญ่จะมีผลให้ความเร็วในการเข้าถึงหน่วยความจำลดลง, วงจรที่ออกแบบมีขนาดใหญ่ขึ้น และกินกำลังงานของวงจรเพิ่มขึ้น

เพื่อเป็นการหลีกเลี่ยงข้อเสียของการสังเคราะห์ความถี่แบบดิจิทัลโดยใช้หน่วยความจำในหัวข้อนี้จึงจะอธิบายหลักการทำงานของวงจรสังเคราะห์ความถี่แบบดิจิทัลที่ไม่ต้องใช้หน่วยความจำ ซึ่งสามารถสรุปเป็นวิธีต่างๆ ได้ดังนี้

2.4.1 ขั้นตอนวิธีคอร์ดิก (CORDIC algorithm)

ขั้นตอนวิธีคอร์ดิก (Coordinate Rotation Digital Computer, CORDIC) เป็นขั้นตอนวิธีที่ใช้ในการคำนวณฟังก์ชันตรีโกณมิติ (Trigonometric function) และเหมาะต่อการนำไปออกแบบเป็นฮาร์ดแวร์ เนื่องจากการคำนวณที่ใช้ในขั้นตอนวิธีคอร์ดิกนั้นสามารถทำได้ด้วยวิธีการเลื่อนและบวก (Shift-and-add operation)



รูปที่ 2.17 การสร้างสัญญาณไซน์ด้วยขั้นตอนวิธีคอร์ดิก

รูปที่ 2.17 แสดงโครงสร้างของการสร้างสัญญาณไซน์โดยการใช้ขั้นตอนวิธีคอร์ดิก ซึ่งประกอบด้วยสัญญาณอินพุต X_{in}, Y_{in}, Z_{in} และสัญญาณเอาต์พุต X_{out}, Y_{out} โดยสัญญาณ Z_{in} เป็นสัญญาณเฟส เพื่อนำมาสังเคราะห์เป็นสัญญาณเอาต์พุตไซน์ (Sine) และโคไซน์ (Cosine) X_{out} และ Y_{out} และสัญญาณ X_{in} และ Y_{in} เป็นค่าข้อมูลแอมพลิจูด เราสามารถเขียนความสัมพันธ์ระหว่างสัญญาณเอาต์พุตและเอาต์พุตของขั้นตอนวิธีคอร์ดิกได้ดังนี้

$$X_{out} = k[X_{in} \cos(Z_{in}) - Y_{in} \sin(Z_{in})] \quad (2.18)$$

$$Y_{out} = k[Y_{in} \cos(Z_{in}) + X_{in} \sin(Z_{in})] \quad (2.19)$$

ใน Gielis และคณะได้ใช้คอร์ดิกโปรเซสเซอร์ (CORDIC processor) ทำหน้าที่แปลงค่าที่อยู่ในรูปโพลาร์ (Polar form) ให้เป็นค่าที่อยู่ในรูปคาร์ทีเซียน (Cartesian form) ถ้ากำหนดให้สัญญาณอินพุต Y_{in} เท่ากับ 0 จะสามารถเขียนสมการความสัมพันธ์ระหว่างอินพุตได้ดังนี้

$$X_{out} = k[X_{in} \cos(Z_{in})] \quad (2.20)$$

$$Y_{out} = k[X_{in} \sin(Z_{in})] \quad (2.21)$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขนาดของแอมพลิจูดและความถี่ของสัญญาณที่สังเคราะห์สามารถควบคุมได้โดยสัญญาณ X_m และ Y_m ตามลำดับ

2.4.2 การคูณเชิงซ้อน (Complex multiplication)

การสังเคราะห์สัญญาณชายน์หรือโคซายน์ด้วยการคูณเชิงซ้อนมีหลักการคือ กำหนดให้มีจำนวนเชิงซ้อน (Complex number) สองจำนวนคือ a และ b มีค่าดังสมการ

$$a = r_\alpha (\cos \alpha + j \sin \alpha) \quad (2.22)$$

$$b = r_\beta (\cos \beta + j \sin \beta) \quad (2.23)$$

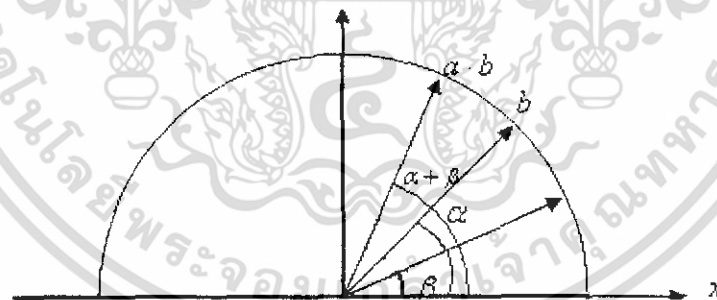
โดย r_α และ r_β เป็นความยาวของเวกเตอร์ α และ β เป็นมุมของเวกเตอร์จากสมการที่ (2.22) และ (2.23) เราสามารถเขียนผลคูณ (Product) ของ a และ b เป็นสมการได้ดังนี้

$$a \cdot b = r_\alpha r_\beta [(\cos \alpha \cos \beta - \sin \alpha \sin \beta) + j(\cos \alpha \sin \beta + \cos \beta \sin \alpha)] \quad (2.24)$$

และเมื่อเปรียบเทียบความสัมพันธ์ของฟังก์ชันทางตรีโกณในสมการที่ (2.25) และ (2.26) กับสมการที่ (2.24) จะเห็นได้ว่าการคูณจำนวนเชิงซ้อนสองจำนวนเข้าด้วยกันจะทำให้เกิดจำนวนเชิงซ้อนใหม่ที่มีค่าของมุม (Angle) เท่ากับผลรวมของมุมของตัวคูณทั้งสอง และรัศมี (Radius) เท่ากับผลคูณของ r_α และ r_β ดังรูปที่ 2.18

$$\sin(\omega + \nu) = \cos \omega \sin \nu + \cos \nu \sin \omega \quad (2.25)$$

$$\cos(\omega + \nu) = \cos \omega \cos \nu - \sin \omega \sin \nu \quad (2.26)$$



รูปที่ 2.18 การหมุนเวกเตอร์ของการคูณเชิงซ้อน เมื่อ $r_\alpha = r_\beta$

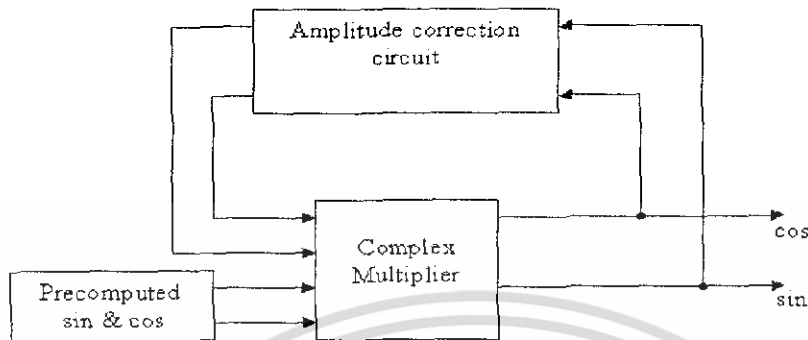
ถ้ากำหนดให้ตัวคูณตัวตั้ง (Multiplier) ในสมการที่ (2.24) คงที่และให้ตัวคูณตัวที่สอง (Multiplicand) เป็นค่าของผลลัพธ์ของการคูณครั้งก่อนหน้า จะได้ว่าเวกเตอร์ที่เกิดจากการคูณจะหมุนรอบวงกลมหนึ่งหน่วยด้วยความเร็วมุมคงที่ นั่นก็คือการสร้างสัญญาณชายน์และโคซายน์นั่นเอง ค่าความถี่ของสัญญาณเอาต์พุตจะถูกกำหนดโดยการเพิ่มของมุมดังสมการ

$$f_{out} = \frac{\beta}{2\pi} \cdot f_{clk} \quad (2.27)$$

โดย β คือมุมของเวกเตอร์ตัวคูณ และ f_{clk} คือความถี่ของสัญญาณนาฬิกาของระบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากการที่ระบบเป็นระบบแบบย้อนกลับ (Feedback) จะทำให้เกิดการสะสมค่าความผิดพลาดในกรณีที่มีแอมพลิจูดของสัญญาณขาเข้าหรือโศขานน์ที่เกิดจากผลคูณแวกเตอร์มีการตัดทอน ซึ่งสะสมเพิ่มขึ้นเรื่อยๆ จึงจำเป็นต้องมีการเพิ่มวงจรชดเชยแอมพลิจูดสำหรับแก้ไขแอมพลิจูดให้ถูกต้องก่อนที่จะย้อนกลับไปที่คุณใหม่ ดังแสดงในรูปที่ 2.19



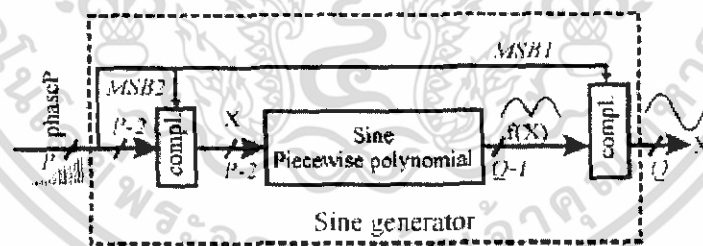
รูปที่ 2.19 การสังเคราะห์สัญญาณขาเข้าด้วยการคูณเชิงซ้อน

2.4.3 การประมาณค่าโพลีโนเมียล (The Polynomial Approximation)

วงจรสังเคราะห์ความถี่แบบดิจิทัลที่ออกแบบจะสังเคราะห์สัญญาณรูปขาเข้าด้วยสมการอนุกรมเทย์เลอร์อันดับห้า

$$\sin(\phi) = \phi(1 + a_2\phi^2 + a_4\phi^4) + \varepsilon(\phi), \quad 0 \leq \phi \leq \frac{\pi}{2} \quad (2.8)$$

สถาปัตยกรรมของวงจรสังเคราะห์ความถี่แบบดิจิทัลโดยใช้วิธีการประมาณค่าโพลีโนเมียลแสดงได้ดังรูปที่ 2.20



รูปที่ 2.20 สถาปัตยกรรมของวงจรสังเคราะห์ความถี่แบบดิจิทัลโดยใช้วิธีการประมาณค่าโพลีโนเมียล

สถาปัตยกรรมของวงจรสังเคราะห์ความถี่แบบดิจิทัลโดยใช้วิธีการประมาณค่าโพลีโนเมียลแสดงได้ดังรูปที่ 2.20 ประกอบด้วยวงจรสร้างเฟส, วงจรสร้างโพลีโนเมียล โดยวงจรสร้างเฟสจะทำหน้าที่สร้างสัญญาณเฟสสำหรับป้อนให้กับวงจรสร้างโพลีโนเมียล (Polynomial generator) และเนื่องจากค่าสัญญาณขาเข้าที่ถูกประมาณค่าด้วยสมการที่ (2.8) นั้นมีค่าใกล้เคียงกับสัญญาณขาเข้าจริงในช่วง 0 ถึง $\pi/2$ ดังนั้นเทคนิคการฮีบฮัดควอดแรนธ์ จึงถูกนำมาใช้ โดยที่บิตเอ็มเอสบีบนสุด (MSB) ถูกใช้สำหรับกำหนดแอมพลิจูดของสัญญาณขาเข้าว่าเป็นบวกหรือลบ และบิตเอ็มเอสบีถัดมา (2^{nd} MSB) ถูกใช้สำหรับกำหนดว่าค่าของเฟสจะถูกคอมพลิเมนต์หรือไม่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

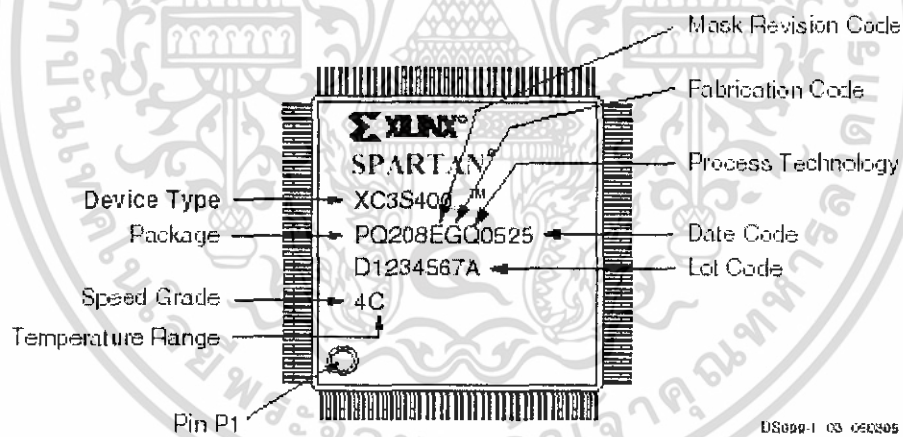
บทที่ 3

การคำนวณและการสร้าง

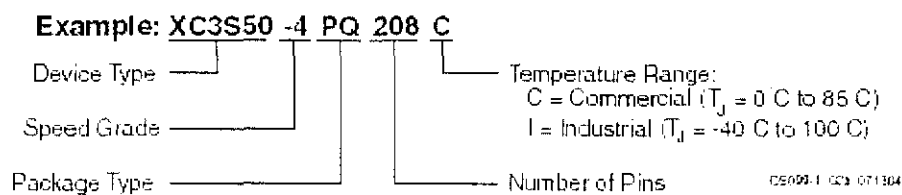
3.1 การออกแบบวงจรเชิงเลขด้วยอุปกรณ์ FPGA

อุปกรณ์ FPGA (Field Programmable Gate Array) เป็นอุปกรณ์ที่ใช้ในการโปรแกรมวงจรที่ได้ ออกแบบลงไปเพื่อให้อุปกรณ์ FPGA มีฟังก์ชันการทำงานตามที่ออกแบบไว้ ในการทำ FPGA ซึ่งเป็นวิธีการออกแบบ IC (Integrated Circuit) แบบ Semicustom อีกวิธีหนึ่ง เมื่อเทียบกับการทำ ASICs (Application Specific Integrated Circuits) แล้วนั้นก็ยังมีทั้งข้อดีและข้อเสีย คือ การทำ FPGA จะมีข้อจำกัดในด้านขนาดของวงจรเพราะภายในอุปกรณ์ FPGA จะมีจำนวนเกต (Gate) ให้ใช้จำนวนจำกัดและการทำ FPGA ก็เหมาะสำหรับการทำผลิตภัณฑ์ต้นแบบหรือเพื่อผลิตในปริมาณต่ำ ส่วนข้อดีของการทำ FPGA ก็คือระยะเวลาที่ใช้ในการทำตั้งแต่เขียนรหัส (Code) อธิบายฮาร์ดแวร์จนกระทั่งดาวน์โหลด (Download) นั้นน้อยกว่าการทำ ASIC มากและการตรวจสอบหรือแก้ไขการออกแบบก็ทำได้สะดวก

การทำ FPGA ในปัจจุบันมีประสิทธิภาพและความสะดวกมากขึ้น ทั้งนี้ก็เนื่องจากทางบริษัทผู้ผลิตอุปกรณ์ FPGA ได้เพิ่มความสามารถของอุปกรณ์ FPGA โดยเพิ่มจำนวนองค์ประกอบภายในหรือปรับปรุงโครงสร้างสถาปัตยกรรมภายในและยังได้เพิ่มประสิทธิภาพของซอฟต์แวร์ที่ใช้ทำ PPR (Partitioning Placement and Routing) สำหรับอุปกรณ์นั้นๆ ด้วย ลักษณะของตัว FPGA และการนำไปใช้งานแสดงดังใน รูปที่ 3.1



รูปที่ 3.1 ลักษณะด้านบนของตัว FPGA



รูปที่ 3.2 แสดงความหมายเบอร์ของตัว FPGA

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำหรับตัวอุปกรณ์ FPGA นั้นก็มีโครงสร้างพื้นฐานเทคโนโลยีที่ใช้สร้างตลอดจนเทคนิควิธีการโปรแกรมที่แตกต่างกันสำหรับผู้ผลิตแต่ละราย นอกจากนั้นอุปกรณ์ FPGA ของแต่ละผู้ผลิตก็มีโครงสร้างและความสามารถที่แตกต่างกันบางส่วน ในการใช้งานนั้นอุปกรณ์ FPGA สามารถนำไปประยุกต์ใช้งานได้ เช่น การประมวลผลสัญญาณเชิงเลข (DSP : Digital Signal Processing) การออกแบบไมโครคอนโทรลเลอร์ เป็นต้น โดยมีขั้นตอนในการออกแบบ

3.2 การออกแบบโดยใช้ภาษาอธิบายการทำงานของฮาร์ดแวร์

ในการออกแบบวงจรเชิงเลขนั้นทำได้โดยการวาดวงจรหรือใช้ภาษาอธิบายฮาร์ดแวร์ในขั้นตอนนี้เป็นขั้นตอนที่ไม่แตกต่างกันระหว่างการออกแบบด้วย FPGA และ ASIC ในกรณีที่ใช้ภาษาอธิบายฮาร์ดแวร์ แต่ในกรณีที่ออกแบบโดยวิธีการวาดวงจรจะแตกต่างกันโดยที่การทำวิธีนี้จะต้องคำนึงถึงเทคโนโลยีที่จะใช้ซึ่งแต่ละเทคโนโลยีก็มีความแตกต่างกันไป จะเห็นได้ว่าการออกแบบโดยใช้ภาษาอธิบายฮาร์ดแวร์ ทำให้สะดวกกว่าเพราะการทำด้วยวิธีนี้ไม่ต้องคำนึงถึงเทคโนโลยีที่จะใช้ (Technology independence) และที่สำคัญการออกแบบด้วยวิธีนี้สามารถที่จะแก้ไข โมเดล (Model) หรือเปลี่ยนแปลงเทคโนโลยีได้สะดวกกว่าเพราะไม่ต้องวาดวงจรใหม่ นั่นคือการออกแบบโดยใช้ภาษาอธิบายฮาร์ดแวร์จะทำให้โมเดลที่ได้ไม่ขึ้นกับเทคโนโลยี

ในการเขียนโค้ดสิ่งที่ต้องคำนึงถึงคือเขียนอย่างไรจึงสามารถสังเคราะห์เป็นวงจรได้และให้คุณสมบัติของวงจรตามที่กำหนด เพราะลักษณะการเขียนโค้ดจะมีผลโดยตรงกับวงจรที่ได้ เนื่องจากในการสังเคราะห์วงจรนั้นซอฟต์แวร์สังเคราะห์วงจร (Synthesis Tools) จะทำการสังเคราะห์ตามโค้ดที่เขียนถ้าอธิบายการทำงานของวงจรเดียวกันแต่เขียนโค้ดในลักษณะที่ต่างกันเมื่อสังเคราะห์แล้วจะได้วงจรที่ต่างกัน และจากวงจรที่ต่างกัน เมื่อนำไปทำต้นแบบด้วย FPGA หรือการทำ ASIC แล้วจะได้ไอซีที่มีคุณสมบัติต่างกันทั้งในด้านของขนาดหรือความเร็ว (Area and Time) ส่วนการเขียนโค้ดลักษณะใดเพื่อให้ได้ผลลัพธ์ที่ดีที่สุดนั้นก็ขึ้นอยู่กับประสบการณ์ในการออกแบบ

3.2.1 การจำลองการทำงานของวงจร (Simulation)

ขั้นตอนนี้เป็นขั้นตอนที่สำคัญเพราะเป็นขั้นตอนที่ใช้ตรวจสอบฟังก์ชันการทำงานของวงจรว่าถูกต้องหรือไม่ มีข้อผิดพลาดตรงไหน เพื่อที่จะได้ทำการแก้ไขให้ถูกต้อง ในขั้นตอนนี้จะใช้ซอฟต์แวร์สำหรับทำการจำลองการทำงานของวงจร เช่น V-System และ Model Sim ของบริษัท Model Technology

3.2.2 การสังเคราะห์วงจร

ในขั้นตอนนี้จะใช้ซอฟต์แวร์สังเคราะห์วงจร (Synthesis tools) ทำการสังเคราะห์โค้ดเพื่อให้ได้เป็นวงจรขึ้นมา แต่ต้องตรวจสอบด้วยว่าซอฟต์แวร์นั้นๆสนับสนุนเทคโนโลยี FPGA (FPGA Library) ที่ต้องการใช้หรือไม่โดย FPGA ที่นิยมใช้งานเช่นของบริษัท Xilinx ตระกูล XC4000 และบริษัท Altera ตระกูล FLEX 10 K ซอฟต์แวร์สังเคราะห์วงจรที่นิยมใช้เช่น โปรแกรม Leonardo Spectrum ของบริษัท Exemplar Logic ซึ่งในขั้นตอนนี้ซอฟต์แวร์สังเคราะห์วงจรจะแปลงโค้ดและทำการ optimize (Optimization) เพื่อให้ได้วงจรตามเทคโนโลยีที่เลือกใช้ นอกจากนี้ยังสามารถกำหนดข้อบังคับสำหรับวงจรได้เช่น ข้อบังคับในเรื่องของเวลา (Time Constraints) หรือข้อบังคับในเรื่องของพื้นที่ ซึ่งข้อบังคับเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษานาน นอนอยู่ภายใต้เห็นไปใช้ประโยชน์ในการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เหล่านี้จะถูกรวบรวมไปใช้ในขั้นตอนออปติไมซ์เพื่อให้งจรที่ได้เป็นไปตามที่กำหนด ส่วนสำคัญในการออปติไมซ์คือการเทียบ (Mapping) วงจรให้เข้ากับเทคโนโลยีที่ใช้เพื่อให้ได้วงจรที่เหมาะสมกับโครงสร้างสถาปัตยกรรมภายในอุปกรณ์ FPGA ในกรณีของ Xilinx ตระกูล XC4000 และ Altera ตระกูล FLEX 10 K จะเทียบโดยใช้วิธี LUT (Look Up Table) เมื่อทำการสังเคราะห์วงจรเสร็จแล้วซอฟต์แวร์สังเคราะห์วงจรก็จะมีกรายงานผลว่าวงจรที่ออกแบบไปนั้นเป็นอย่างไร เช่น มีความหน่วง (Delay) เท่าไรใช้ทรัพยากรต่างๆใน FPGA อะไรบ้าง เป็นต้น

3.2.3 การแบ่งวงจร (Partitioning)

ขั้นตอนนี้เป็นการแบ่งวงจรที่ได้จากการสังเคราะห์ให้เป็นส่วนย่อยๆ สำหรับลงใน CLB, IOBs หรือองค์ประกอบอื่นๆ ภายในอุปกรณ์ FPGA สำหรับเกณฑ์ที่ใช้ในการแบ่งคือให้แต่ละส่วนที่จะแยกออกจากกันมีจำนวนสัญญาณที่เชื่อมต่อระหว่างกันน้อยที่สุดเท่าที่จะทำได้เพื่อช่วยลดความหนาแน่นในขั้นตอนการเชื่อมต่อสัญญาณ (Routing) ในขั้นตอนนี้จะใช้ซอฟต์แวร์ทำโดยซอฟต์แวร์จะเทียบส่วนประกอบของวงจรเช่น เกท (Gate), ฟลิปฟลอป (Flipflop) ลงในทรัพยากรต่างๆ ที่มีอยู่ในอุปกรณ์ FPGA (CLBs, IOBs, BUFT) และ (Edge Decoder) หลังจากทำขั้นตอนนี้เสร็จแล้วสามารถที่จะทราบว่าวงจรใช้จำนวนทรัพยากรภายในอุปกรณ์ FPGA ไปเท่าไร ส่วนซอฟต์แวร์ที่ใช้ในขั้นตอนนี้ขึ้นอยู่กับตัว FPGA ที่ใช้งานเช่น FPGA ของบริษัท Xilinx จะใช้ ISE WebPACK 7.0 ซอฟต์แวร์ย่อยอื่นๆอีก เพื่อให้การทำ PPR (Partitioning, Placement and Routing) เป็นไปอย่างต่อเนื่อง

3.2.4 การวางอุปกรณ์ (Placement)

ขั้นตอนนี้เป็นการเลือกทำเลที่ตั้งของแต่ละส่วนของวงจรที่ผ่านการแบ่งวงจร (Partitioning) มาแล้วว่าจะอยู่ในตำแหน่งใดในอุปกรณ์ FPGA เพื่อให้ได้ผลลัพธ์ที่ดีที่สุด เช่น วงจรส่วนไหนควรอยู่ใกล้กันเพื่อจะได้ค้นหาเส้นทาง (Route) ได้ง่ายหรือช่วยลดความหน่วง

จะเห็นได้ว่าตำแหน่งภายในอุปกรณ์ FPGA นั้นมีความสำคัญเพราะถ้าจัดวางวงจรลงในตำแหน่งที่ไม่เหมาะสมแล้วจะทำให้ความหน่วงเพิ่มขึ้นหรือตัว Router ทำการค้นหาเส้นทางสัญญาณได้ไม่หมด

3.2.5 การเชื่อมต่อสัญญาณ (Routing)

ในขั้นตอนนี้เป็นการเชื่อมต่อสัญญาณระหว่างองค์ประกอบต่างๆ ภายในอุปกรณ์ FPGA เช่น ระหว่าง CLBs หรือระหว่าง CLBs กับ IOBs ขั้นตอนนี้จะทำต่อเนื่องจากการวางอุปกรณ์ ในกรณีที่ทำการวางอุปกรณ์ไว้ไม่ดีซอฟต์แวร์ก็จะทำการเชื่อมต่อสัญญาณได้ไม่หมดหรือเกิดความหน่วงเกินค่าที่กำหนดในข้อบังคับโดยสามารถทำขั้นตอนนี้ได้โดยใช้ซอฟต์แวร์เช่นกันหรือทำการเชื่อมต่อสัญญาณด้วยตัวเอง (Manual Layout) ก็ได้แต่ทางที่ดีควรใช้ซอฟต์แวร์ทำดีกว่าโดยให้ทำการค้นหาเส้นทางหลายๆครั้งเพื่อหาครั้งที่ดีที่สุด นอกจากนั้นการกำหนดข้อบังคับทางเวลา (Time Constraints) จะช่วยให้ผลที่ได้จากการทำการเชื่อมต่อสัญญาณดีขึ้นได้

3.2.6 การโปรแกรมอุปกรณ์ FPGA (Configuration)

หลังจากที่วงจรผ่านขั้นตอนต่างๆจนกระทั่งผ่านการทำ PPR (Partitioning, Placement and Routing) แล้วนั้น ถึงตอนนี้ก็สามารถที่จะดาวน์โหลดลงในอุปกรณ์ FPGA ได้แล้วในการดาวน์โหลดนี้ ก่อนอื่นต้องแปลงแบบวงจรรวมที่ได้ให้เป็นข้อมูลวงจร (Configuration data) ซึ่งอยู่ในรูปของบิตสตรีม (Bit-Stream) ก่อนแล้วจึงดาวน์โหลดไปเพื่อให้อุปกรณ์ FPGA มีฟังก์ชันการทำงานตามวงจรที่ออกแบบไว้

จากที่อธิบายมาทั้งหมดจะเห็นได้ว่าการออกแบบเพื่อทำ FPGA นั้น ทำได้สะดวกกว่าการทำ ASIC มากเพราะใช้เวลาน้อยกว่ามาก ส่วนสำคัญที่ใช้ในการทำ FPGA คือซอฟต์แวร์ที่ใช้ตั้งแต่การเขียนโค้ดอธิบายฮาร์ดแวร์จนกระทั่งดาวน์โหลดลงในอุปกรณ์ FPGA ซึ่งซอฟต์แวร์ที่ใช้ต้องเป็นซอฟต์แวร์ที่ใช้ทำงานต่อเนื่องกัน ซึ่งในการดาวน์โหลดลงในอุปกรณ์ FPGA จะทำการสร้างไฟล์ .bin ที่สามารถนำไปโปรแกรมลงชิพ FPGA ได้ โดยใช้โปรแกรม iMPACT programmer ผ่านทางสายดาวน์โหลด JTAG



รูปที่ 3.3 แสดง การดาวน์โหลดโปรแกรมลงชิพ FPGA ที่หน้าจอก็จะแสดงชิพ Flash PROM และ FPGA พร้อมกัน

3.3 FPGA ตระกูล SPRATAN XC3S-xxx

FPGA ตระกูล SPRATAN ของบริษัท XILINX เป็น FPGA ที่มีโครงสร้างภายในเป็นแบบ SRAM Based FPGA ใช้เทคโนโลยีในการโปรแกรมเหมือนกับหน่วยความจำแบบ SRAM (Static Ram) ทำให้การโปรแกรมสามารถทำซ้ำได้โดยไม่จำกัดจำนวนครั้ง และใช้เวลาโปรแกรมน้อยมาก (ระดับ nsec) การโปรแกรมทำได้ง่ายเช่นเดียวกับการเขียน SRAM ทั่วไป และสามารถเก็บโค้ดคำสั่งต่างๆจากการเขียนด้วยภาษา VHDL ซึ่งเป็น Block Ram ที่ฝังตัวอยู่ในชิพ FPGA แต่มีข้อเสียคือ มันต้องการไฟเลี้ยงในการเก็บวงจรที่ออกแบบไว้

วงจรสังเคราะห์ที่ความถี่ดิจิทัลแบบโดยตรง มีโครงสร้างหลัก ๆ ตามรูป 3.5 จะอาศัยหน่วยความจำเพื่อเก็บค่าขนาดของสัญญาณที่เฟสต่าง ๆ เอาไว้ ซึ่งรูปสัญญาณที่เก็บเอาไว้ อาจจะเป็นรูปสัญญาณไซน์ (Sine) สัญญาณรูปฟันเลื่อย (sawtooth) หรือรูปสัญญาณอื่น ๆ ที่ผู้ใช้งานได้ทำการออกแบบไว้ เมื่อต้องการรูปสัญญาณออกมาที่เอาท์พุท ก็จะทำการเรียกข้อมูลที่เฟสต่าง ๆ ที่เก็บเอาไว้ออกมา เพื่อป้อนให้กับวงจรแปลงสัญญาณดิจิทัลเป็นอนาล็อก ดังรูปที่ 3.5 หากต้องการจะเปลี่ยนความถี่ของสัญญาณก็ทำได้โดย การเปลี่ยนอัตราการเรียกข้อมูลออกจากหน่วยความจำ (เป็นความเร็วของสัญญาณนาฬิกา) หรือเปลี่ยนลำดับการเรียกข้อมูลที่เฟสต่าง ๆ เพื่อให้จำนวนของ ข้อมูลในหนึ่งคาบของสัญญาณ

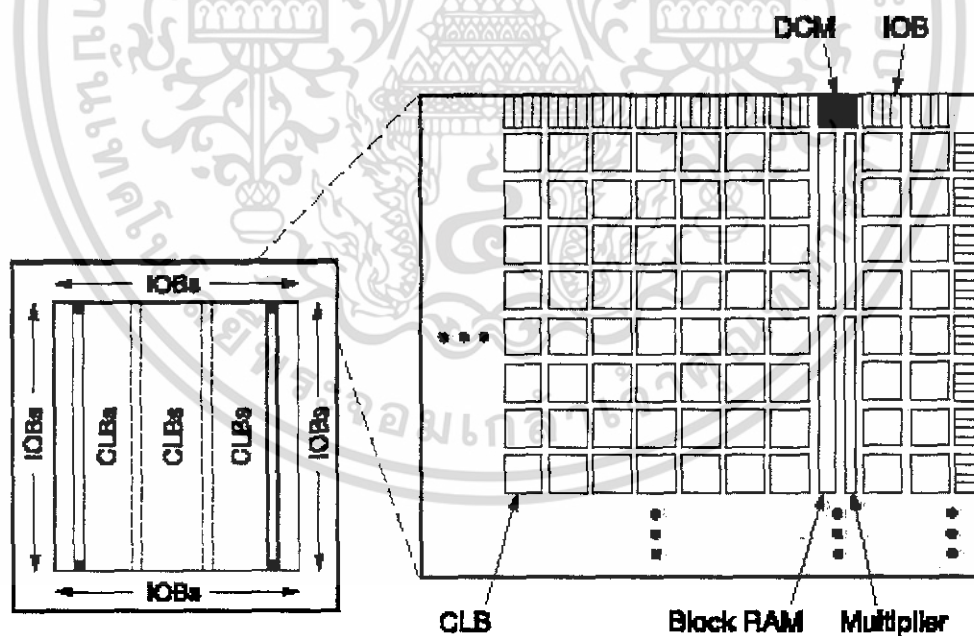
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เปลี่ยนไป ดังเช่นในรูปที่ 3.6 จะเห็นว่ามีรูปของสัญญาณสองความถี่ รูปสัญญาณที่มีความถี่สูงกว่า จะมีจำนวนข้อมูลที่ถูกรับออกมาน้อยกว่ารูปสัญญาณที่มีความถี่ต่ำกว่า

มีข้อสังเกตคือ ระยะ ระหว่างข้อมูลที่ถูกรับออกมามีค่าของสัญญาณทั้งสองความถี่มีค่าเท่ากัน ดังนั้นความละเอียดในการเปลี่ยนความถี่จะขึ้นอยู่กับขนาดของหน่วยความจำที่ใช้เก็บข้อมูล

ตารางที่ 3.1 แสดง Ram แบบ Single Port ที่สร้างจาก Block Ram แต่ละชุด

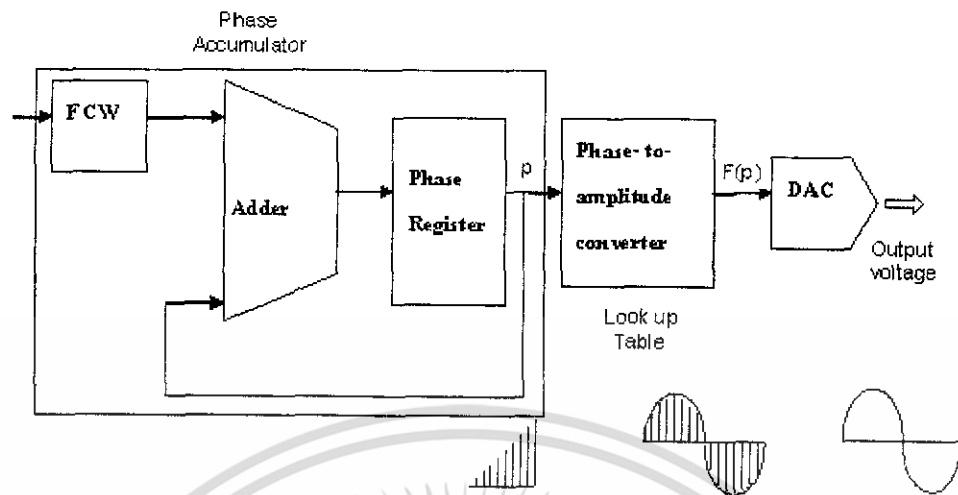
Organization	Memory Depth	Data Width	Parity Width
512x36	512	32	4
1kx18	1024	16	2
2kx9	2048	8	1
4kx4	4069	4	-
8kx2	8192	2	-
16kx1	16384	1	-



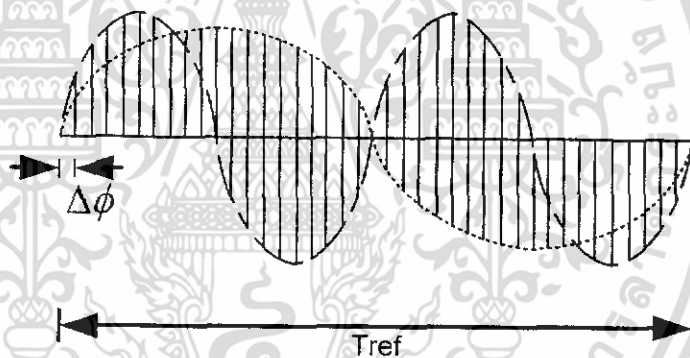
รูปที่ 3.4 โครงสร้างภายในของ FPGA ตระกูล SPRATAN3 เบอร์ XC3S200

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.4 Direct Digital Frequency Synthesizer (DDS)



รูปที่ 3.5 แสดง Block Diagram โครงสร้างของวงจรสังเคราะห์ความถี่ที่ใช้หน่วยความจำเป็นค่าขนาดของสัญญาณเอาไว



รูปที่ 3.6 วงจรสร้างสัญญาณไซน์แบบเชิงเลข

มีข้อสังเกตคือ ระยะ ระหว่างข้อมูลที่เรียกออกมาของสัญญาณทั้งสองความถี่มีค่าเท่ากัน ดังนั้นความละเอียดในการเปลี่ยนความถี่จะขึ้นอยู่กับขนาดของหน่วยความจำที่ใช้เก็บข้อมูล

หลักการของวงจรสังเคราะห์ความถี่ดิจิทัลแบบโดยตรง (Direct Digital Frequency Synthesizer) นั้นจะมีสัญญาณนาฬิกา (Fclk) นำมาป้อนให้กับส่วนของ Phase Accumulator ซึ่งวงจรนี้จะทำการบวกวนซ้ำด้วยค่าที่รับเข้ามาจากส่วน เพิ่มเฟส ผลของการบวกจะ ได้ค่าออกมาเป็น เฟสสะสม P ค่าสะสมเฟส P นี้ จะมีลักษณะเพิ่มขึ้นด้วยจำนวนเท่ากันทุกครั้งที่ทำกรบวก ตามจังหวะสัญญาณนาฬิกา อ้างอิง (Fclk) ค่าเฟสสะสม P จะนำไปใช้ในตารางเปิดดู (Look up table) ซึ่งได้จัดเก็บค่าขนาดของรูปสัญญาณไว้แล้ว ซึ่งจะ ได้สัญญาณ F(P) ออกมา ซึ่งเป็นค่าแบบดิจิทัล มาทำการเปลี่ยนเป็นค่า อนาลอก ด้วยวงจร Digital - to - Analog Converter (DA) ก็จะได้ Output voltage เป็นสัญญาณแบบที่ต้องการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากโครงสร้างนี้ การเปลี่ยนแปลงของ จะมีผลต่อค่าของเฟสที่เกิดขึ้นในแต่ละรอบของ สัญญาณนาฬิกา และจำนวนรอบของสัญญาณนาฬิกา

ความถี่ที่สามารถกำเนิดได้จากโครงสร้างนี้ คือ

$$F_{out} = (F_{cw} * F_{clk}) / 2^N \tag{3.1}$$

เมื่อ

F_{cw} = Frequency Control Word

F_{out} = Frequency Output

F_{clk} = Frequency Clock

N = จำนวน bit ของ Phase Accumulator

ถ้าค่า N มีค่ามาก Step size ของความถี่จะมีความละเอียดมากขึ้นด้วย

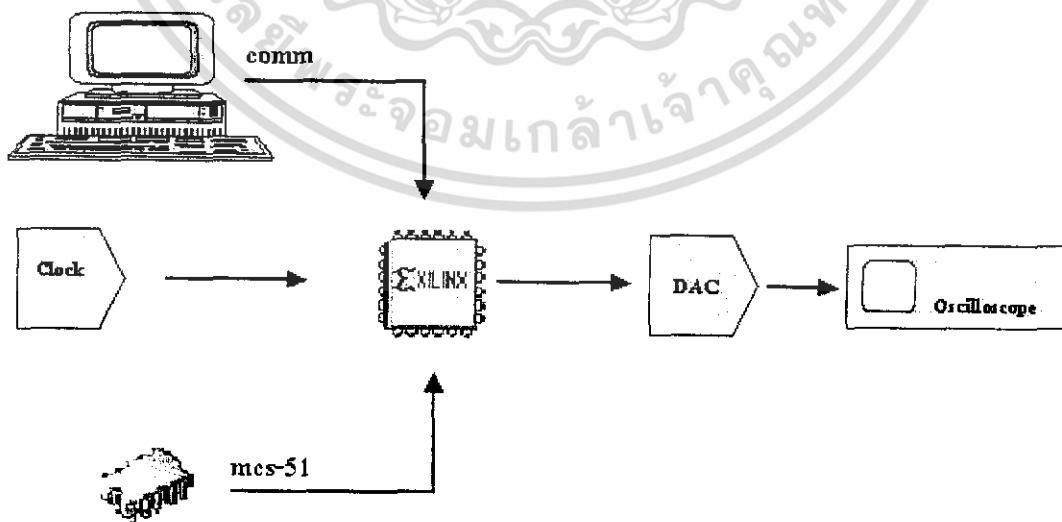
เราสามารถหาความละเอียดในการปรับความถี่ (Frequency resolution) ของวงจรสังเคราะห์ความถี่แบบดิจิทัลได้โดยกำหนดให้ F_{cw} มีค่าเท่ากับ 1 ซึ่งสามารถเขียนเป็นสมการได้ดังนี้

$$F_{res} = F_{clk} / 2^N \tag{3.2}$$

โดย F_{res} คือค่าความละเอียดในการปรับความถี่ ตัวอย่างเช่นในกรณีที่ใช้รีจิสเตอร์ความถี่ขนาด 32 บิต และวงจรทำงานที่สัญญาณนาฬิกาความถี่ 100 เมกะเฮิร์ตซ์จะมีความละเอียดในการปรับความถี่เท่ากับ 0.0233 เฮิร์ตซ์

3.4.1 การออกแบบและการสร้าง

การออกแบบตัวกำเนิดสัญญาณไซน์แบบดิจิทัลด้วย FPGA เป็นการออกแบบโดยการนำเอา Personal Computer (PC) และ MCS-51 กับ FPGA ต่อทำงานร่วมกัน ดัง รูปที่ 3.7

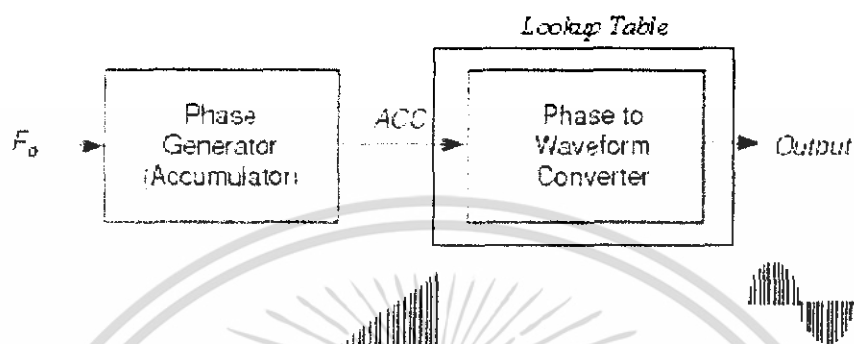


รูปที่ 3.7 ขั้นตอนการทำงานของ ตัวกำเนิดสัญญาณไซน์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

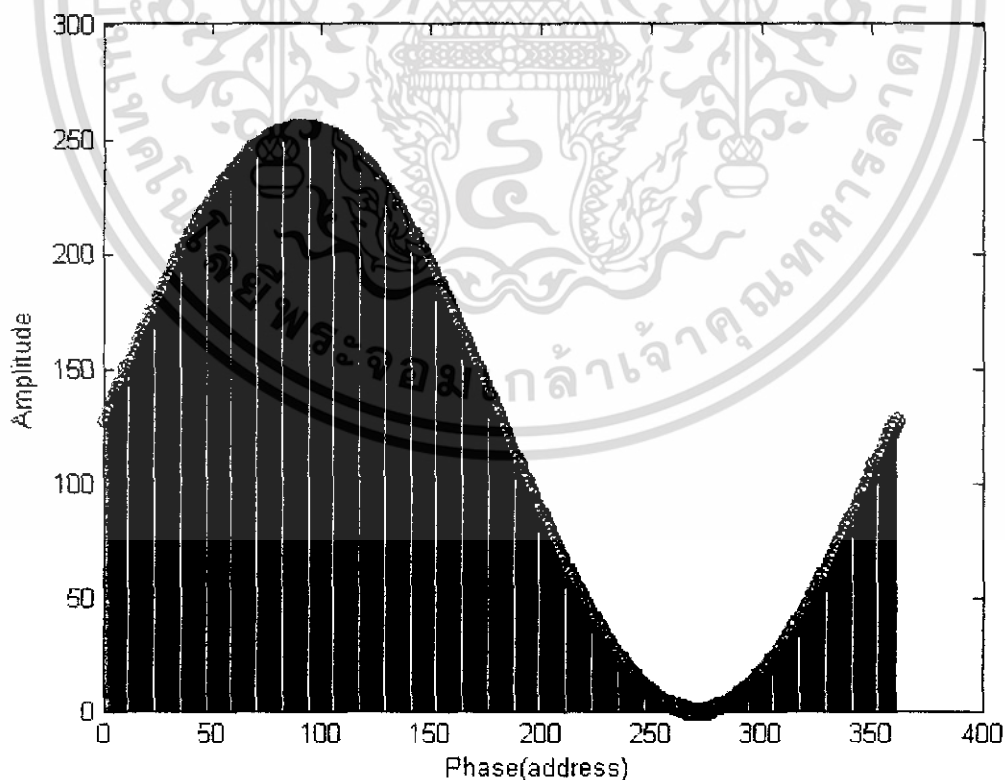
3.4.2 การสร้างสัญญาณรูปคลื่นไซน์

การสร้างสัญญาณรูปคลื่นไซน์ จะใช้โปรแกรม MATLAB ในการสร้างแอมพลิจูดของสัญญาณที่ต้องการสังเคราะห์ เก็บไว้ในหน่วยความจำ ค่าของแอมพลิจูดที่ถูกดึงออกจากหน่วยความจำจะสัมพันธ์กับค่าของเฟสที่ได้รับจากวงจรสร้างเฟส กล่าวคือค่าของเฟสจะถูกใช้เป็นตำแหน่ง (Address) สำหรับอ้างอิงหน่วยความจำในการดึงค่าแอมพลิจูดออกมา



รูปที่ 3.8 หน่วยความจำสำหรับเปลี่ยนเฟสเป็นแอมพลิจูดของสัญญาณไซน์

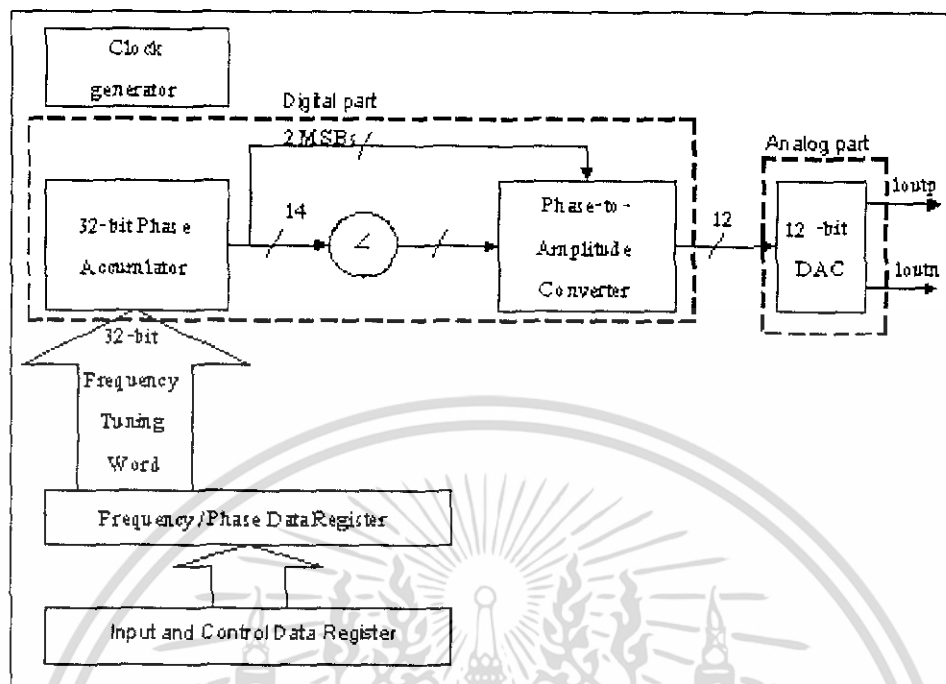
โปรแกรม MATLAB ในการสร้างแอมพลิจูดของสัญญาณไซน์ ขนาด 8 บิต จำนวน 360 ค่า



รูปที่ 3.9 ผลจากค่าในตารางมาวาดกราฟ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่นิยมนำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.5 รายละเอียดการออกแบบ



รูปที่ 3.10 สถาปัตยกรรมของชิปวงจรรวมเพื่อสังเคราะห์ความถี่แบบดิจิทัลที่ใช้หน่วยความจำ

วงจรรวมเพื่อสังเคราะห์ความถี่แบบดิจิทัลที่ใช้หน่วยความจำมีสถาปัตยกรรมตามรูปที่ 3.10 โดยจะแบ่งออกเป็นส่วนหลักๆคือส่วนดิจิทัล (Digital part) และส่วนอนาล็อก (Analog part) โดยส่วนดิจิทัลจะประกอบด้วยวงจรรสร้างเฟสขนาด 32 บิต และวงจรเปลี่ยนเฟสเป็นแอมพลิจูด และส่วนประกอบหลักของส่วนอนาล็อกคือวงจรแปลงดิจิทัลเป็นวงจรขนาด 12 บิต เราสามารถสรุปรายละเอียดการออกแบบวงจรแยกตามส่วนประกอบหลักๆได้ดังนี้

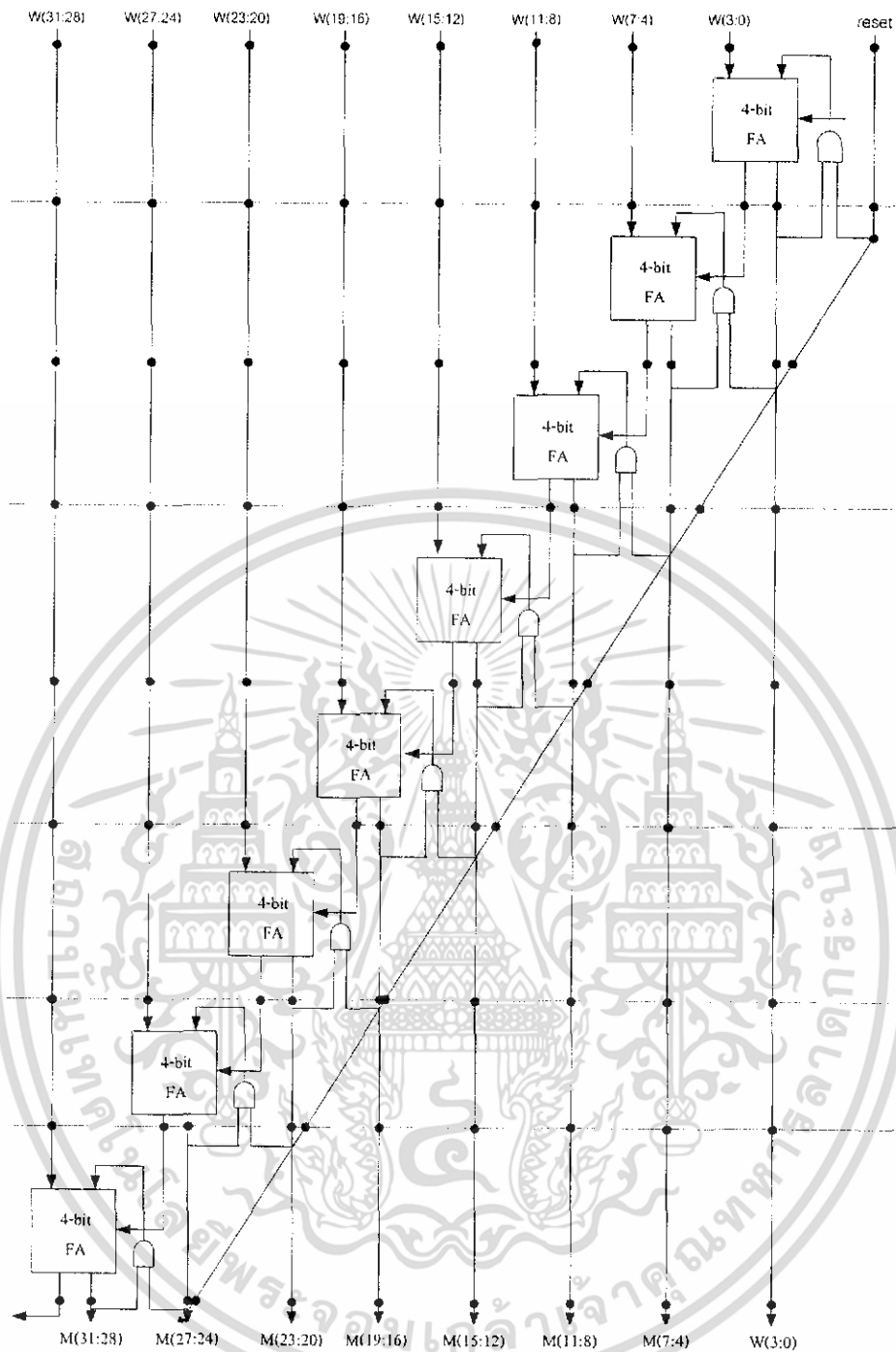
3.6 ส่วนดิจิทัล (Digital part)

วงจรในส่วนดิจิทัลได้ถูกออกแบบด้วยวิธีการออกแบบจากบนลงล่าง (Top-down design) ซึ่งมีขั้นตอนคือเริ่มจากการกำหนดข้อกำหนด (Specification) ของวงจร และเมื่อได้ข้อกำหนดของวงจรแล้วก็จะทำการเขียนโค้ดคำสั่งบรรยายพฤติกรรมการทำงานของวงจรด้วยภาษาวีเอสดีแอล สร้างแล้วนำไปสังเคราะห์

3.6.1 วงจรสร้างเฟส 32 บิต (32-bit Phase Accumulator)

วงจรสร้างเฟส 32 บิตมีโครงสร้างดังรูปที่ 3.11 ประกอบด้วยวงจรวกขนาด 4 บิต (4-bit full adder) จำนวน 8 ชุดมาต่อกันในลักษณะสายท่อ (Pipeline) เพื่อเพิ่มความเร็วในการทำงานของวงจร นอกจากนี้ยังมีการนำสัญญาณรีเซ็ตแบบสายท่อมาใช้เพื่อเคลียร์ค่าของรีจิสเตอร์เฟส โดยที่จุดค่าๆในรูปที่ 3.10 เป็นรีจิสเตอร์สำหรับปรับเวลา (Pre-skewing and De-skewing registers)

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น เมื่อผู้ดูแลให้เข้าไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้คัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.11 โครงสร้างของวงจรสร้างเฟสขนาด 32 บิต

3.6.2 วงจรบวกแบบ Look-Ahead-Carry

เมื่อได้ศึกษาวงจรบวกแบบขนาน n บิต ทำให้ทราบว่า ความเหมาะสมของวงจรบวกแบบขนาน n บิตนี้ จำนวนบิตสูงสุดที่สามารถทำการบวกได้ดี คือ 16 บิต และระดับความเร็วในการทำการบวกอยู่ในระดับปานกลางถึงเร็วมาก (High Speed) ถ้าหากมีการนำวงจรบวกแบบขนาน n บิต มาทำการบวกกับค่าที่มีจำนวนมากขึ้น และระดับความเร็ว (Very High Speed) ที่สูงขึ้นด้วยแล้ว ยังทำให้ไม่เกิดการทดขึ้น หรือไม่มีการรวมค่าตัวทดออก (Co) เข้ากับบิตถัดไป เนื่องจากเวลาที่ค่าตัวทดออก (Co) ที่ส่งออกจากวงจรบวกนี้เป็นเอกสารที่ส่งวนไปสำหรับการใช้งานเพื่อการศึกษาเท่านั้น เมื่อนุญตให้เข้าเบเซประเขยนดานการคาไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บวกแบบคิดค่าตัวทศตใดๆ ไปสู่อีกวงจรบวกแบบคิดค่าตัวทศตอื่น ไม่มีการจัดการเรื่องเวลาที่
เหมาะสมกับค่าของการบวกของตัวตั้ง (A) และค่าของตัวบวก (B)

ฉะนั้นในวงจรบวกแบบ Look-Ahead-Carry จึงเป็นวงจรบวกที่มีการแก้ไข และจัดการในเรื่อง
ของช่วงเวลาของค่าตัวทศตออก (Co) โดยลดผลของเวลาที่ใช้ในการทดค่าออก ระหว่างวงจรบวกแบบ
คิดค่าตัวทศ

การเริ่มต้นออกแบบวงจรบวกแบบ Look-Ahead-Carry จะพิจารณากันที่สมการของวงจรบวก
แบบคิดค่าตัวทศ

$$S_n = A_n \oplus B_n \oplus C_n$$

= ผลบวกของการบวกกันจำนวน n บิต

(3.3)

$$C_{n+1} = (A_n \oplus B_n)C_n + A_n B_n$$

ในสภาวะที่สามารถทำให้ $C_{n+1} = 1$ ได้นั้น ค่าของ $(A_n \oplus B_n)$ และ C_n จะต้องมีค่าเท่ากับ "1"
หรือ $A_n B_n = 1$

1	1	0	1	C_n
0	1	1	1	A_n
$\oplus 1$	$\oplus 0$	$\oplus 1$	$\oplus 1$	$+ B_n$
10	10	10	11	$C_{n+1} S_n$
$(A_n \oplus B_n)C_n = 1$		$A_n B_n = 1$		

พิจารณาสมการ (3.3) เมื่อเราทำการบวกในหลักแรกหรือบิต 0 (LSB) จะมีเพียงค่าตัวตั้ง (A) และ
ค่าตัวบวก (B) เพียงสองจำนวนเท่านั้น ซึ่งในการบวกกันระหว่างค่าตัวตั้ง (A) และค่าตัวบวก (B) ของหลัก
แรก (บิต 0) ก่อให้เกิดค่าตัวทศตออก (Co) ไปสู่การบวกระหว่างค่าตัวตั้ง (A) และค่าตัวบวก (B) ของหลัก
ที่ 2 (บิต 1) และจะเป็นเช่นนี้เรื่อยไป จนกว่าจะถึงบิตสูงสุด (MSB) ของการบวก ทำให้สามารถสรุปผล
การกระทำการบวกได้ดังสมการที่ (3.4)

$$P_n = A_n \oplus B_n$$

$$G_n = A_n B_n$$
(3.4)

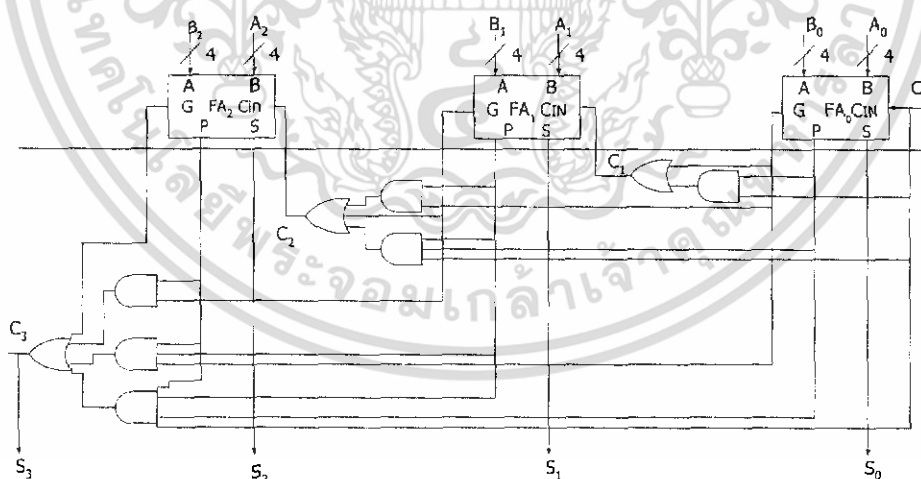
ให้พิจารณาวงจรบวกแบบคิดค่าตัวทศ (Full Adder) เราจะได้เห็นโนด P และ G อยู่ในวงจร ซึ่ง
หมายถึงค่าที่แสดงอยู่ในสมการ (3.4) นั่นเอง จากผลของสมการ (3.3) และ (3.4) สามารถสร้างสมการของ
การบวกของจำนวน 2 จำนวน ซึ่งมีขนาดบิต ได้ดังสมการ (3.5)

หลักที่ 1 (บิต 0)	S_0	=	$P_0 \oplus C_0$
	C_1	=	$P_0 C_0 \oplus G_0$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$\begin{aligned}
 \text{หลักที่ 2 (บิต 1)} \quad S_1 &= P_1 \oplus C_1 \\
 C_2 &= P_1 C_1 \oplus G_1 \\
 &= P_1(P_0 C_0 \oplus G_0) + G_1 \\
 &= P_1 P_0 C_0 + P_1 G_0 + G_1 \\
 \text{หลักที่ 3 (บิต 2)} \quad S_2 &= P_2 \oplus C_2 \tag{3.5} \\
 C_3 &= P_2(P_1 P_0 C_0 + P_1 G_0 + G_1) + G_2 \\
 &= P_2 P_1 P_0 C_0 + P_2 P_1 G_0 + P_2 G_1 + G_2 \\
 &\vdots \\
 &\vdots \\
 &\vdots \\
 \text{หลักที่ } n+1 \text{ (บิต } n) \quad S_n &= P_n \oplus C_n \\
 C_{n+1} &= P_n C_n \oplus G_n \\
 &= P_n P_{n-1} P_{n-2} \dots P_0 C_0 + P_n P_{n-1} \dots P_1 G_0 + \dots + G_n
 \end{aligned}$$

วงจรวกแบบ Look-Ahead-Carry จะอาศัยโครงสร้างของวงจรวกแบบคิดค่าตัวทด (Full Adder) โดยจะดึงจาก โหนด P และ G มาใช้งาน ซึ่ง S จะหมายถึงค่าผลบวกของการบวกส่วนค่า G_{n+1} นั้น จะต้องผ่านชุดวงจร Look-Ahead-Carry Generator วงจร Look-Ahead-Carry Generator จะเป็นส่วนที่อยู่ด้านล่างของเส้นประในรูปที่ 3.12 ซึ่งถ้าหากเราพิจารณากันอย่างผิวเผินจะเห็นค่าของตัวทศออก (C_0) เพราะว่าการบวกของวงจรวกแบบคิดค่าตัวทด (Full Adder) เราจะใช้เพียงค่า P_n, G_n และ S_n เท่านั้น



รูปที่ 3.12 วงจรวกแบบ Look-Ahead-Carry

ในกรณีที่เรากำลังต้องการออกแบบวงจรวกแบบ Look-Ahead-Carry ให้อยู่ในรูปแบบโมดูล โดยในหนึ่งโมดูลจะมีจำนวน m บิต เราสามารถขยายผลของสมการ (3.4) ได้ดังสมการ (3.6)

จากสมการ (3.7) ค่าตัวแปร r จะหมายถึงหลักที่กระทำการบวก เราสามารถทราบจำนวนบิต (n) หรือจำนวนวงจรวกแบบคิดค่าตัวทด (Full Adder) ของวงจรวกแบบ Look-Ahead-Carry ในรูปแบบ

โมดูลได้จากผลคูณของจำนวนบิต (m) กับจำนวนหลักที่กระทำการบวก (r) แล้วบวกเข้าไปอีกหนึ่งค่า เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปเผยแพร่โดยไม่ได้รับอนุญาต ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สมการ (3.6)

$$n = mr + 1 \quad (3.6)$$

หลักที่ 1, m บิต

$$S_{m0} = P_{m0} + C_{m0}$$

$$C_{m0} = P_{m0}C_{m0} + G_{m0}$$

หลักที่ 2, m บิต

$$S_{m1} = P_{m1} + C_{m1}$$

$$C_{m2} = P_{m1}C_{m1} + G_{m1}$$

$$= P_{m1}(P_{m0}C_{m0} + G_{m0}) + G_{m1}$$

$$= P_{m1}P_{m0}C_{m0} + P_{m1}G_{m0} + G_{m1}$$

หลักที่ 3, m บิต

$$S_{m2} = P_{m2} + C_{m2} \quad (3.7)$$

$$C_{m3} = P_{m2}C_{m2} + G_{m2}$$

$$= P_{m2}(P_{m1}P_{m0}C_{m0} + P_{m1}G_{m0} + G_{m1}) + G_{m2}$$

$$= P_{m2}P_{m1}P_{m0}C_{m0} + P_{m2}P_{m1}G_{m0} + P_{m2}G_{m1} + G_{m2}$$

หลักที่ r, m บิต

$$S_{nr} = P_{nr} + C_{nr}$$

$$C_{nr+1} = P_{nr}C_{nr} + G_{nr}$$

$$= S_{nr-1}$$

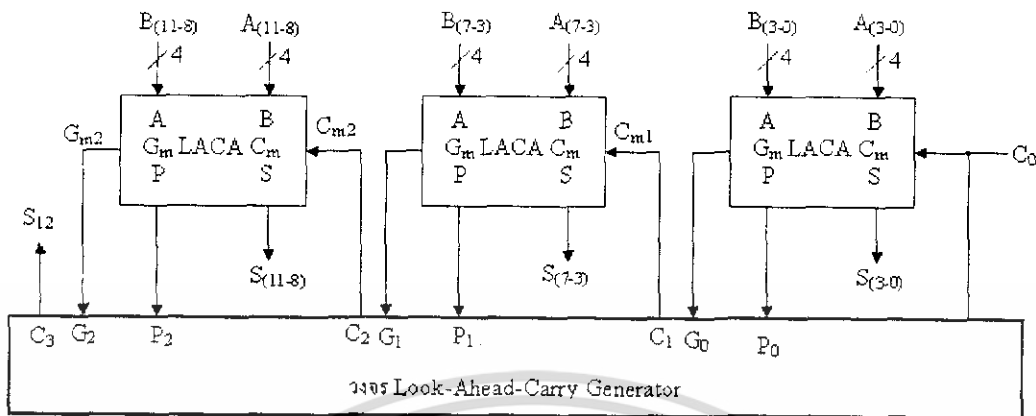
ในรูปที่ 3.13 เป็นตัวอย่างการนำสมการ (3.7) มาใช้ ซึ่งเป็นวงจรบวกแบบ Look-Ahead-Carry ในรูปแบบที่เป็นโมดูลจำนวน 12 บิต โดยหนึ่งโมดูล เราจะกำหนดให้มีความสามารถกระทำการบวกได้ 4 บิต (m) เพราะฉะนั้นเราจะต้องใช้วงจรบวกแบบคิดค่าตัวทด (Full Adder) จำนวน 3 ชุดด้วยกันเสมือนว่า วงจรดังรูปที่ 3.13 มีการกระทำการบวกกันแค่ 3 หลัก (r) และแทนค่าลงในสมการ (3.6) เราก็จะได้ว่า ผลลัพธ์ที่จะได้จากการบวกของวงจร Look-Ahead-Carry ในรูปที่ 3.13 คือ $n = mr + 1 = (4)(3) + 1 = 13$ บิต คือ S12-S0 นั่นเอง

คุณสมบัติเด่นของวงจรบวกแบบ Look-Ahead-Carry เมื่อเปรียบเทียบกับวงจรบวกแบบขนาน n บิต คือวงจรบวกแบบ Look-Ahead-Carry จะช่วยในการปรับแต่ง โดยการหน่วงค่าเวลาของตัวทอดออก (Co) แต่เมื่อมาพิจารณาการออกแบบจำนวนบิตมากๆ ดิกว่าวงจรบวกแบบขนาน n บิต ณ ความเร็วในการบวกที่เท่ากัน

3.6.3 วงจรเปลี่ยนเฟสเป็นแอมป์ริจูด (Phase-to-amplitude converter)

วงจรเปลี่ยนเฟสเป็นแอมป์ริจูดสามารถสังเคราะห์รูปสัญญาณได้ทั้งหมด 4 รูปสัญญาณ ได้แก่ รูปไซน์ รูปแรมป์ รูปฟันเลื่อย และสัญญาณแบบสุ่ม วิธีบีบอัดควอดเรนท์ ได้ถูกนำมาใช้ในการ เอกสารฉบับเอกสารทสมวันเวสท์หวั่งการเงันเพื่อกการศึกษาเท้านน เมื่อผู้อยู่ใต้เห็นาไปเอาระเอษณทานการค้ ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สังเคราะห์สัญญาณทุกสัญญาณ นอกจากวิธีการบีบอัดควอดแรนท์แล้วเราสามารถอธิบายขั้นตอนการสังเคราะห์สัญญาณแต่ละรูปสัญญาณ ได้ดังนี้



รูปที่ 3.13 วงจรบวกแบบ Look-Ahead-Carry ในรูปแบบที่เป็นโมดูล

3.6.3.1 รูปไซน์ (Sine)

ใช้หลักการของการเก็บรูปสัญญาณไว้ในหน่วยความจำแบบอ่านอย่างเดียว (ROM) โดยบีบอัดข้อมูลเพื่อลดขนาดของหน่วยความจำด้วยขั้นตอนวิธีโมดิฟายซันเดอร์แลนด์ (Modified Sunderland algorithm)

ในขั้นตอนวิธีซันเดอร์แลนด์ค่าของเฟสขนาด 12 บิต (ϕ) ที่ถูกสร้างจากวงจรสร้างเฟสจะถูกแบ่งออกเป็น 3 ส่วน เท่ากัน ส่วนละ 4 บิต (a, b, c) ดังสมการที่ (3. 8)

$$\phi = a + b + c \tag{3.8}$$

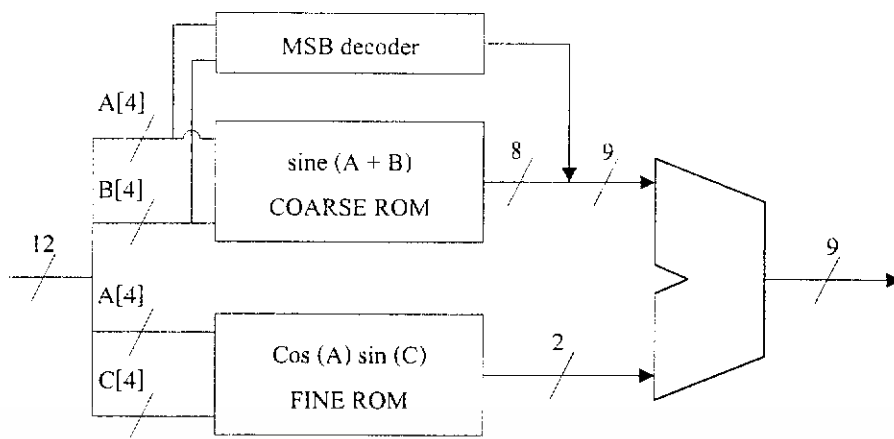
และจากความสัมพันธ์ของฟังก์ชัน $\sin(\phi) = \sin(a + b + c)$ เราสามารถประมาณค่าของฟังก์ชัน $\sin(\phi)$ ได้ดังสมการที่ (3.9) และ (3.10) ตามลำดับ

$$\sin(a + b + c) = \sin(a + b)\cos c + \cos a \cos b \sin c - \sin a \sin b \sin c \tag{3.9}$$

$$\sin(a + b + c) \approx \sin(a + b) + \cos a \sin b \tag{3.10}$$

สมการที่ (3.10) จะถูกใช้ในการออกแบบเป็นวงจรจริง โดย $\sin(a + b)$ จะถูกเก็บในหน่วยความจำหยาบ (Coarse ROM) และ $\cos a \sin c$ จะถูกเก็บในหน่วยความจำละเอียด (Fine ROM) สัญญาณเอาต์พุตของทั้งหน่วยความจำหยาบและหน่วยความจำละเอียดจะถูกนำมารวมกันเพื่อสร้างเป็นสัญญาณไซน์ดังแสดงในรูปที่ 3.12 โดยที่สัญญาณเอาต์พุตของวงจรเปลี่ยนเฟสเป็นแอมพลิจูดด้วยขั้นตอนวิธีซันเดอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



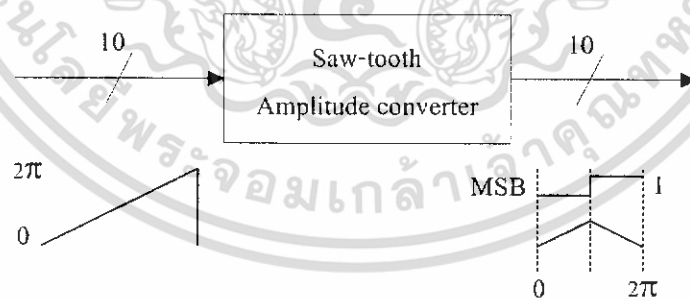
รูปที่ 3.14 ขั้นตอนวิธี โมดิไฟซันเดอร์แลนค์

แลนค์จะมีขนาด 11 บิต ซึ่งจะถูกนำไปดีโคดควอแดรนต์และสร้างเป็นสัญญาณแอมพลิจูดเอาต์พุตของสัญญาณขาเข้าขนาด 12 บิต ต่อไป

3.6.3.2 รูปแรมป์ (Ramp)

สัญญาณรูปแรมป์นั้นสามารถสังเคราะห์ได้โดยตรงจากสัญญาณเอาต์พุตของวงจรสร้างเฟส โดยจะดึงเฉพาะส่วนบิตเอ็มเอสบี 12 บิต แรกมาเป็นสัญญาณเอาต์พุตของวงจร เปลี่ยนเฟสเป็นแอมพลิจูดเพื่อป้อนให้กับวงจรแปลงเป็นดิจิทัลเป็นอนาลอกต่อไป

รูปฟันเลื่อย (Saw-tooth)



รูปที่ 3.15 การสังเคราะห์สัญญาณรูปฟันเลื่อย

สัญญาณรูปฟันเลื่อยสามารถสังเคราะห์ได้จากบิตเอ็มเอสบี 10 บิตแรกที่ได้จากเอาต์พุตของวงจรสร้างเฟส โดยถ้าบิตเอ็มเอสบีบนสุดมีค่าเท่ากับ 0 สัญญาณ 10 บิต จากวงจรสร้างเฟสจะถูกส่งต่อไปยังวงจรแปลงเป็นอนาลอก แต่ถ้าบิตเอ็มเอสบีบนสุดมีค่าเท่ากับ 1 สัญญาณ 10 บิต จากวงจรสร้างเฟสจะถูกคอมพลิเมนต์ก่อนที่จะส่งต่อไปยังวงจรแปลงดิจิทัลเป็นอนาลอก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.6.4 การประมาณค่าโพลิโนเมียล (The Polynomial Approximation)

3.6.4.1 รูปไซน์ (Sine)

วงจรสังเคราะห์ความถี่แบบดิจิทัลที่ออกแบบจะสังเคราะห์สัญญาณรูปไซน์ด้วยสมการอนุกรมเทย์เลอร์อันดับห้า

$$\sin(\phi) = \phi(1 + a_2\phi^2 + a_4\phi^4) + \varepsilon(\phi) \quad , 0 \leq \phi \leq \frac{\pi}{2} \quad (3.11)$$

เราสามารถเขียนความสัมพันธ์ของเอาต์พุตของวงจรสร้างเฟสขนาด N บิต (m) กับค่าเฟส (ϕ) ของวงจรสังเคราะห์ความถี่แบบดิจิทัลที่มีสถาปัตยกรรมตามรูปที่ ได้ดังนี้

$$m = \left(\frac{\phi}{2\pi}\right) 2^N \quad (3.12)$$

เราสามารถเขียนสมการที่ (3.11) ให้อยู่ในรูปของเอาต์พุตของวงจรสร้างเฟสได้ดังนี้

$$y(m) = m(k_0 - k_2m^2 + k_4m^4) \quad (3.13)$$

เพื่อให้สะดวกต่อการออกแบบสมการที่ (3.13) จะถูกคูณด้วย 10^9 โดยที่ค่า k_0, k_2, k_4 มีค่าดังตารางที่ 3.2

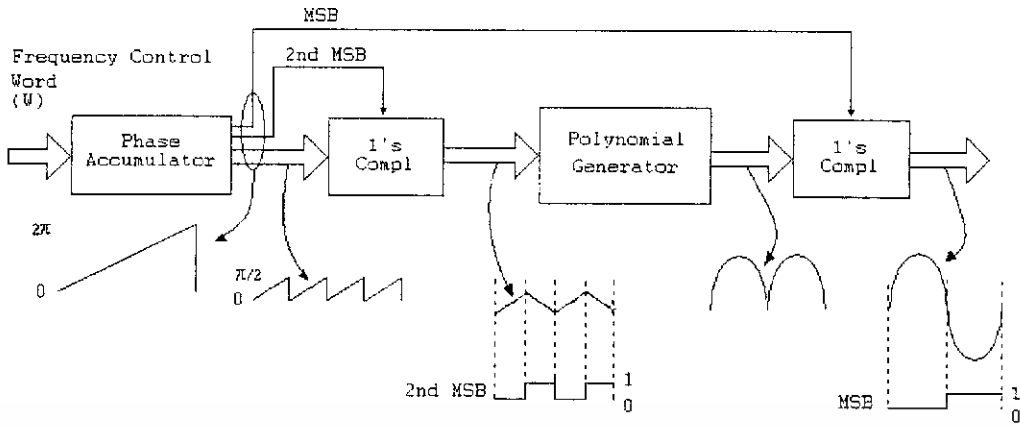
ตารางที่ 3.2 ค่าคงที่ที่ใช้ในสมการที่ (3.12)

k_0	$k_0 * 10^9$
k_2	$k_2 * 10^9$
k_4	$k_4 * 10^9$

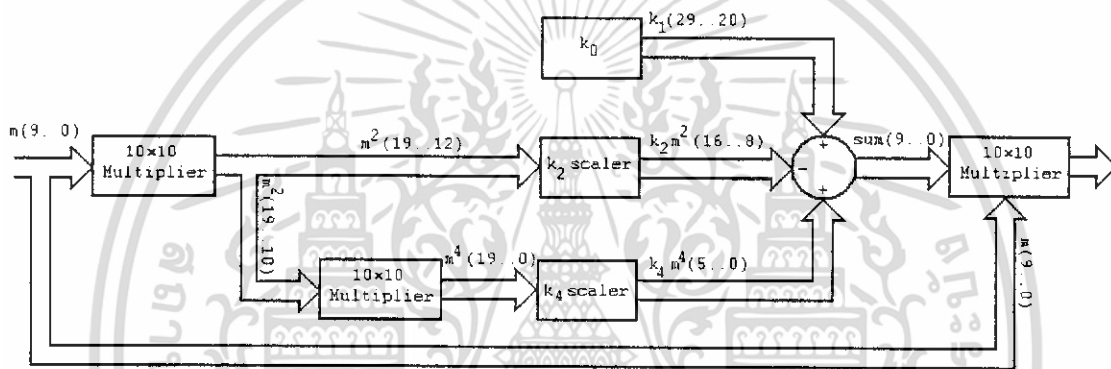
จะเห็นว่าถ้านำสมการที่ (3.12) ไปออกแบบเป็นวงจรจะต้องมีการคูณกันถึงห้าครั้ง ซึ่งไม่เหมาะต่อการนำไปออกแบบเป็นวงจรทางฮาร์ดแวร์ ดังนั้นการคูณของ k_2m^2 และ k_4m^4 จึงถูกออกแบบด้วยการเลื่อน-บวก (Shift-and-add) ซึ่งจะได้ค่าของ k_2 และ k_4

สถาปัตยกรรมของวงจรสังเคราะห์ความถี่แบบดิจิทัลโดยใช้วิธีการประมาณค่าโพลิโนเมียลแสดงได้ดังรูปที่ 3.16 ประกอบด้วยวงจรสร้างเฟส, วงจรสร้างโพลิโนเมียล โดยวงจรสร้างเฟสจะทำหน้าที่สร้างสัญญาณเฟสสำหรับป้อนให้กับวงจรสร้างโพลิโนเมียล (Polynomial generator) และเนื่องจากค่าสัญญาณไซน์ที่ถูกประมาณค่าด้วยสมการที่ (3.12) นั้นมีค่าใกล้เคียงกับสัญญาณไซน์จริงในช่วง 0 ถึง $\pi/2$ ดังนั้นเทคนิคการบีบอัดควอแดรนต์ จึงถูกนำมาใช้ โดยที่บิตเอ็มเอสบีบนสุด (MSB) ถูกใช้สำหรับกำหนดแอมพลิจูดของสัญญาณไซน์ว่าเป็นบวกหรือลบ และบิตเอ็มเอสบีถัดมา (2^{nd} MSB) ถูกใช้สำหรับกำหนดว่าค่าของเฟสจะถูกคอมพลิเมนต์หรือไม่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.16 สถาปัตยกรรมของวงจรสังเคราะห์ความถี่แบบดิจิทัลโดยใช้วิธีการประมาณค่าโพลีโนเมียล



รูปที่ 3.17 โครงสร้างของวงจรสังเคราะห์โพลีโนเมียล

รูปที่ 3.17 แสดงโครงสร้างของวงจรสังเคราะห์โพลีโนเมียลที่ใช้สำหรับการประมาณค่าแอมพลิจูดของสัญญาณขาเข้าในช่วง 0 ถึง $\pi/2$ ซึ่งประกอบด้วยวงจรถูคูณขนาด 10×10 บิต จำนวน 3 วงจร, วงจรเลื่อน-บวก (Shift-and-add) สำหรับคำนวณค่า $k_2 m^2$ และ $k_4 m^4$ และวงจรถูกบวกและวงจรถอยอย่างละหนึ่งตัว

สัญญาณเฟสขนาด 32 บิต จากวงจรสังเคราะห์เฟสจะถูกตัดให้เหลือเฉพาะบิตเอ็มเอสบีบนสุด 12 บิต $m(11:0)$ โดยที่บิตเอ็มเอสบีสองบิตแรกถูกใช้สำหรับการกำหนดควอดแดรนต์ และอีก 10 บิตที่เหลือ $m(9:0)$ จะถูกนำไปใช้ในการคำนวณเป็นสัญญาณขาเข้า

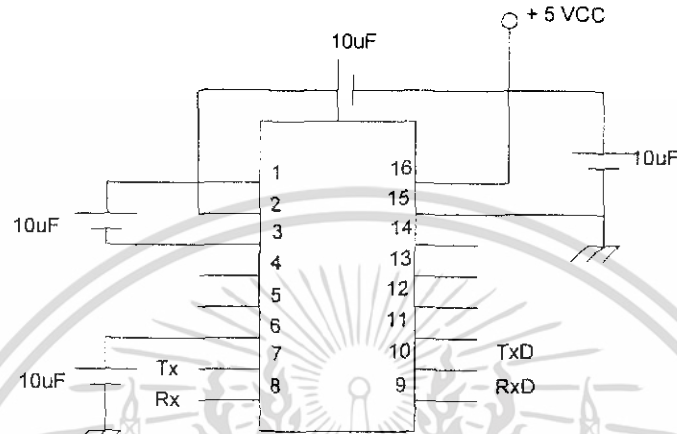
3.7 วงจรแปลงระดับแรงดัน

ในการเชื่อมต่อวงจรที่ทำงานระดับแรงดันแบบพีทีแอล เข้ากับพอร์ต RS-232 ของเครื่องคอมพิวเตอร์ที่มีระดับแรงดัน -15 โวลต์ ถึง +15 โวลต์ นั้นจะต้องมีวงจรพิเศษเพื่อทำการเปลี่ยนแปลงระดับแรงดันให้เหมาะสมซึ่งในที่นี้เราได้เลือกใช้ MAX 232 ซึ่งใช้อุปกรณ์ประกอบจากภายนอกน้อย คือ ใช้ C เพียง 5 ตัวเท่านั้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.7.1 หลักการทำงาน

จากวงจรจะใช้ C ที่ต่อระหว่างขา 1 กับ 3, ระหว่างขา 4 กับ 5, ขา 2 กับ 6 เป็นตัวกำหนดระดับแรงดันที่ใช้ในการเชื่อมต่อโดยขา R11 และ R10 จะเป็นขาที่รับระดับแรงดัน -15 โวลต์ ถึง +15 โวลต์ และแปลงออกเป็นแรงดัน 0 โวลต์ และ +5 โวลต์ ตามลำดับ ออกที่ขา R10 และ R20 ส่วนขา T10 และ T11 จะรับแรงดันที่เป็น 0 โวลต์ และ +5 โวลต์ ตามลำดับ แปลงเป็นระดับแรงดัน -10 โวลต์ +10 โวลต์ ออกที่ขา 7 และ 10

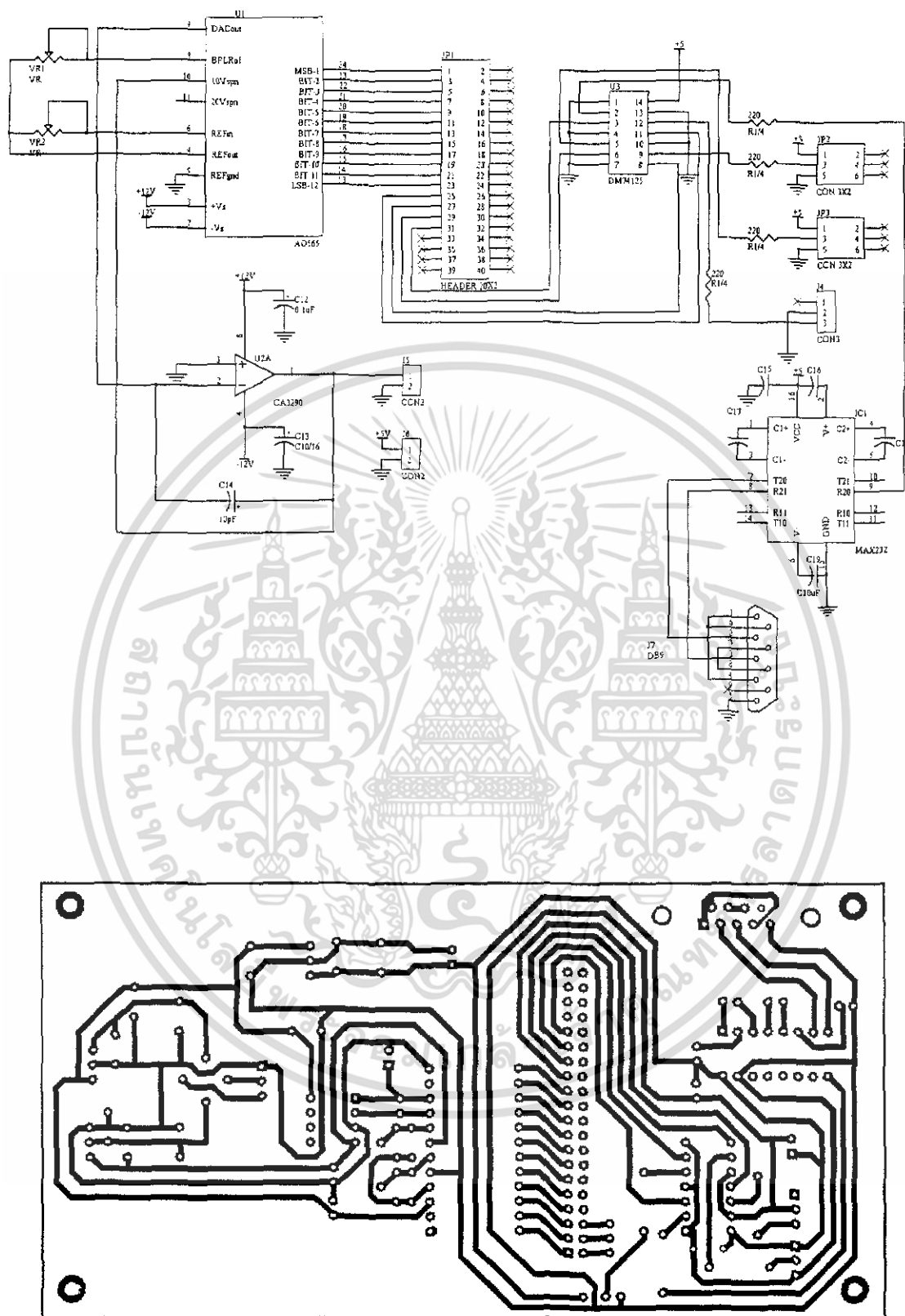


รูปที่ 3.18 วงจรแปลงระดับแรงดัน

3.8 วงจรแปลงระดับสัญญาณดิจิทัลเป็นอนาล็อก

เป็นส่วนที่นำสัญญาณ Output จาก FPGA ที่เป็นสัญญาณแบบดิจิทัล ขนาด 12 bit มาแปลงให้สัญญาณ Analog โดยใช้ IC DAC AD 565 วงจรใช้งานแสดงดังรูป

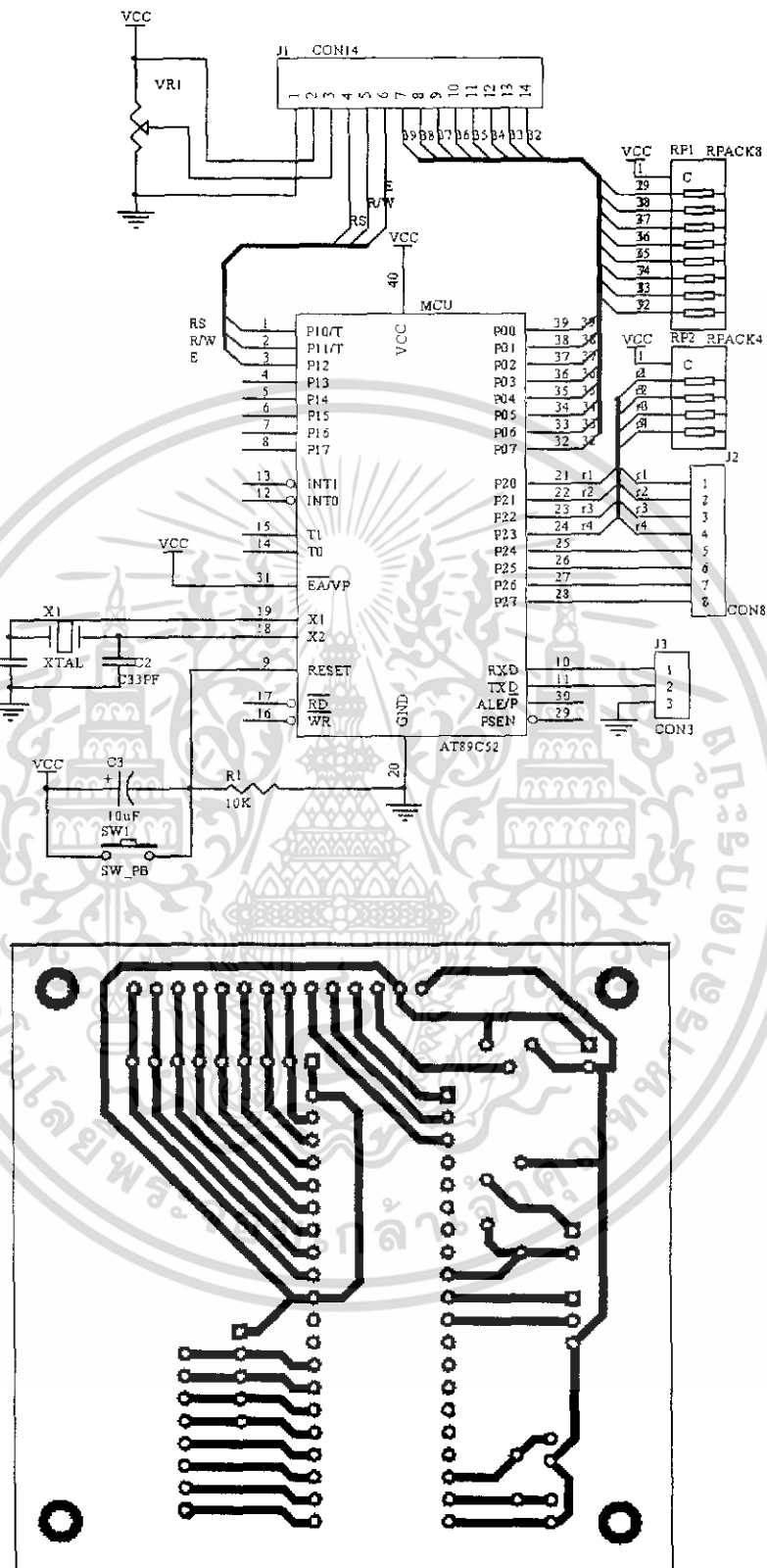
3.8.1 การออกแบบวงจรและลายวงจรส่วนของวงจรแปลงดิจิตอลเป็นอนาล็อก



รูปที่ 3.19 วงจรแปลงดิจิตอลเป็นอนาล็อกและลายวงจรแปลงดิจิตอลเป็นอนาล็อก

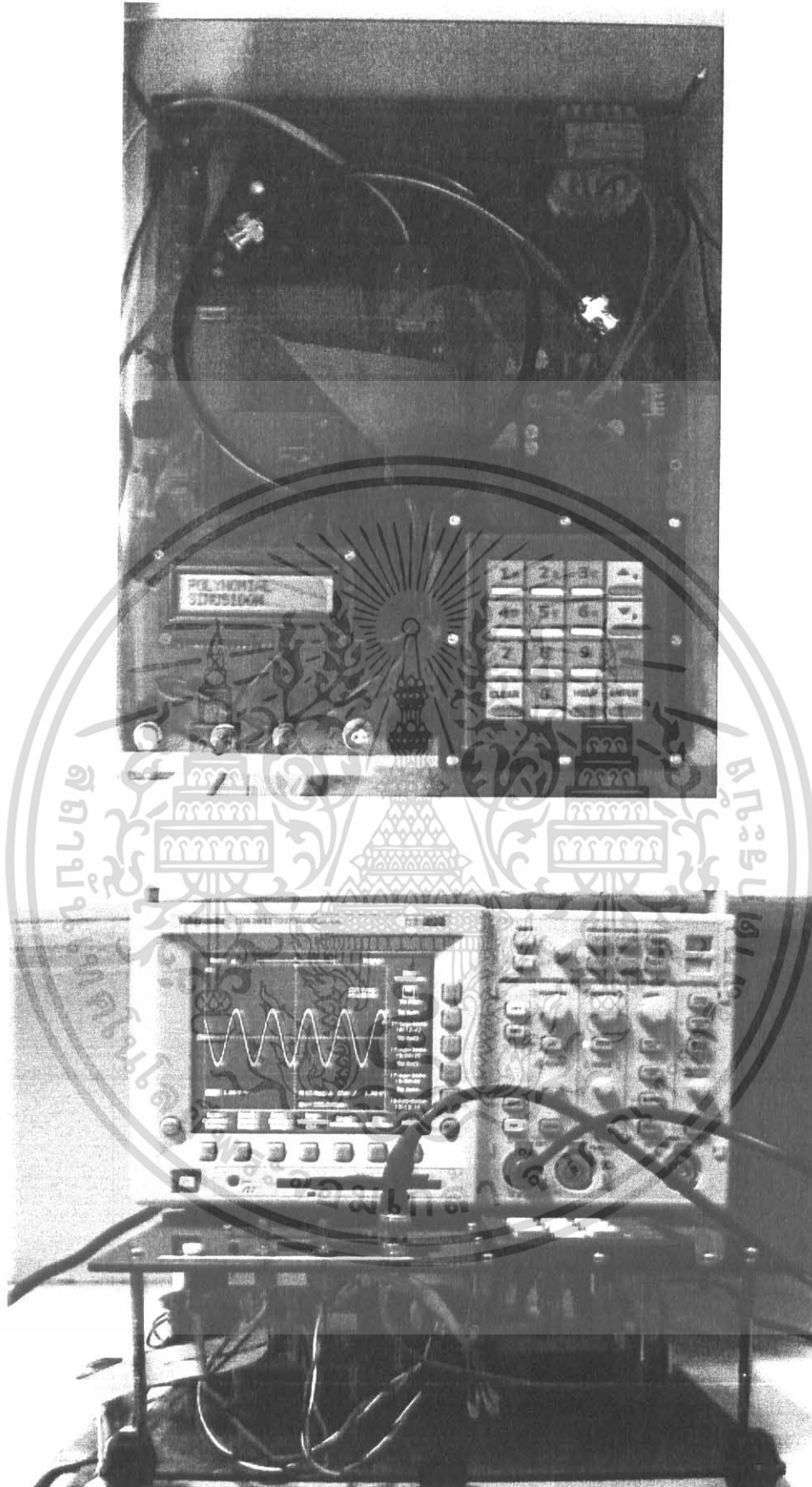
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.8.2 การออกแบบวงจรและลายวงจรส่วนของวงจรส่งข้อมูลผ่านพอร์ตอนุกรมโดยใช้ MCS-51



รูปที่ 3.20 วงจรส่งข้อมูลผ่านพอร์ตอนุกรมและลายวงจรส่งข้อมูลมาจาก MCS-51

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.21 ภาพถ่ายอุปกรณ์และวงจรใช้งานจริง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4

การทดลองและผลการทำงาน

การออกแบบวงจรกำเนิดสัญญาณไซน์แบบดิจิทัล ประกอบด้วย 3 ส่วนคือ ส่วนของโครงสร้างวงจรกำเนิดสัญญาณแบบดิจิทัลโดยวิธีการใช้ตารางเปิดดู (Look-up table) ส่วนของโครงสร้างวงจรกำเนิดสัญญาณแบบดิจิทัลโดย การประมาณค่าโดยโพลิโนเมียล (Polynomial Approximation) โดยได้ทำการพัฒนางจรสะสมเฟส (Accumulator) ซึ่งประกอบไปด้วยวงจรวกแบบคิดตัวทดส่วนหน้าขนาด 8 บิต (4-bit Carry Look-Ahead Adder) จำนวน 4 ชุด มาต่อกันในลักษณะสายท่อ (Pipeline) เพื่อเพิ่มความเร็วในการทำงานของวงจร จะใช้การอธิบายการพฤติกรรมการทำงานของวงจร ด้วยภาษา VHDL และการจำลองการทำงานในการควบคุมการทำงานในแต่ละวิธีมีการรูปแบบการทำงานดังต่อไปนี้

ตารางที่ 4.1 รูปแบบการส่งข้อมูลผ่านทางพอร์ตอนุกรม

Address(hex)	Register Name	Description
01	RegINC0	Bit7-Bit0 of frequency control word(W) $f_{out} = (f_{clk} * W) / 2^{32}$ Where f _{out} is output frequency F _{clk} is MCLK frequency W is 32-frequency control word
02	RegINC1	Bit15-Bit8 of frequency control word(W)
03	RegINC2	Bit23-Bit16 of frequency control word(W)
04	RegINC3	Bit31-Bit24 of frequency control word(W)
05	RegCTRL	DDS control register used for controlling DDS operation 0000000 Sinusindom(Look up table) 0000001 Rectangular(Look up table) 0000010 Tringular(Look up table) 0000011 Ramp(Look up table) 0000100 Sinusindom(Polynomial) 0000101 Ramp(Polynomial) 0000110 Random

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.1 การควบคุมการใช้งานการเชื่อมต่อผ่านพอร์ตอนุกรม

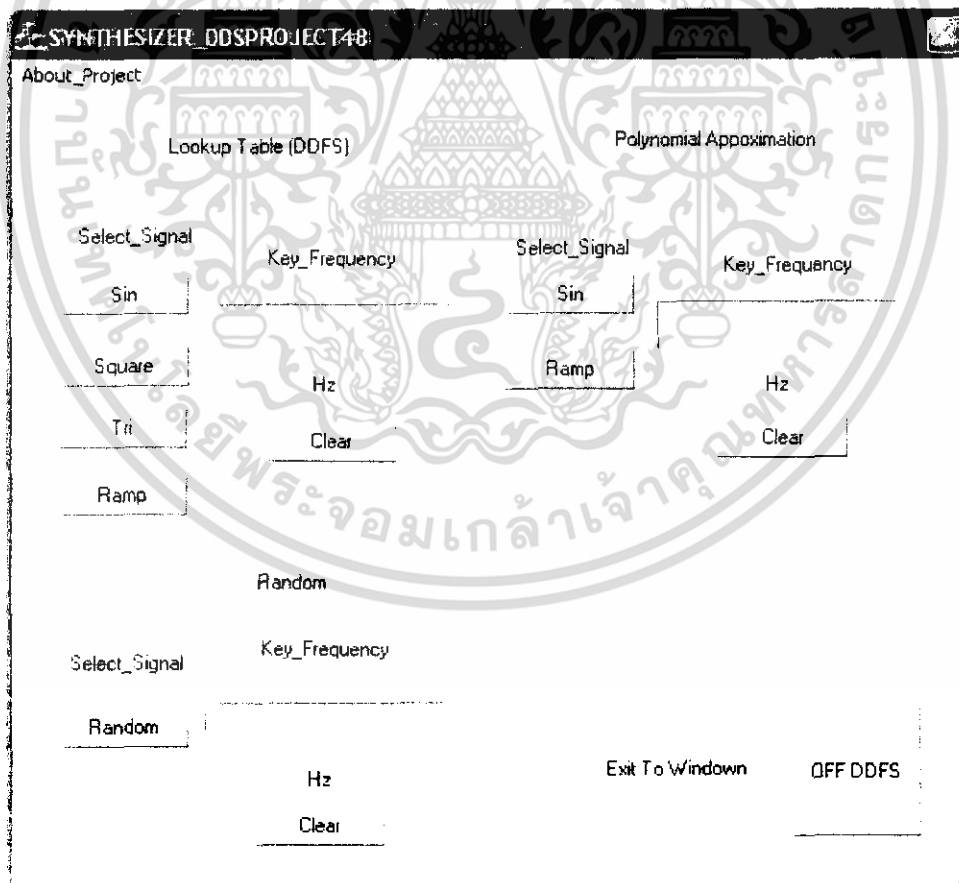
การควบคุมการทำงานของบอร์ดสามารถทำได้สองวิธี (1) ใช้โปรแกรมคอมพิวเตอร์ควบคุมการทำงานผ่านพอร์ตอนุกรม (RS-232) และ (2) ใช้ MCS-51 ควบคุมการทำงาน ซึ่งจะกล่าวถึงรายละเอียดของการทำงานในแต่ละแบบในหัวข้อถัดไป

โปรแกรมการใช้งานการเชื่อมต่อผ่านพอร์ตอนุกรมโดยใช้ วิชาวลซี

การทำงานของโปรแกรมที่เขียนด้วยวิชาวลซี มีความสะดวกในการเชื่อมต่อผ่านพอร์ตอนุกรมเราจึงนำมาประยุกต์ใช้งานร่วมกับ Oscillator จากรูปที่ 4.1 จะแสดงวิธีการเรียกใช้ออสซิลเลเตอร์โดยจะแบ่งเป็นสามส่วนหลักดังนี้

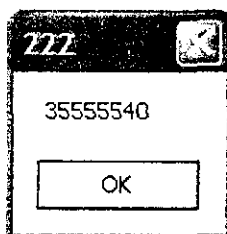
1. ส่วนของ Lookup Table (DDFS) ใช้สำหรับเรียกใช้การสร้างสัญญาณแบบตารางเปิดดู
2. ส่วนของ Polynomial Approximation ใช้สำหรับเรียกใช้โดยวิธีการสร้างแบบ โพลีโนเมียล
3. ส่วนของ Random ใช้สำหรับเรียกใช้โดยการสร้างสัญญาณสุ่ม

ในการใช้งานสามารถทำได้โดยการกำหนดความถี่ที่ต้องการลงใน EDIT BOX โปรแกรม จะคำนวณค่า FCW ส่งค่าไปให้ FPGA และจะแสดงเมนูของค่าที่คำนวณได้ ดังนี้



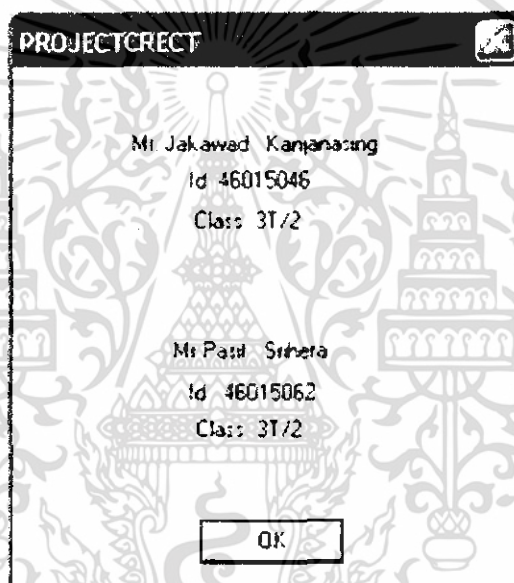
รูปที่ 4.1 แสดงเมนูหลักของโปรแกรมการใช้งานคิจดอลออสซิลเลเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.2 แสดงเมนู ของค่าที่คำนวณได้

ในการเชื่อมต่อเมนูย่อยจะเลือกที่ Select จะแสดงเมนูของ Advisor และ Project Create ขึ้นมาดังนี้



รูปที่ 4.3 แสดงเมนู Project Create

จากรูปที่ 4.3 แสดงรายละเอียดของผู้จัดทำของ โปรเจ็กต์

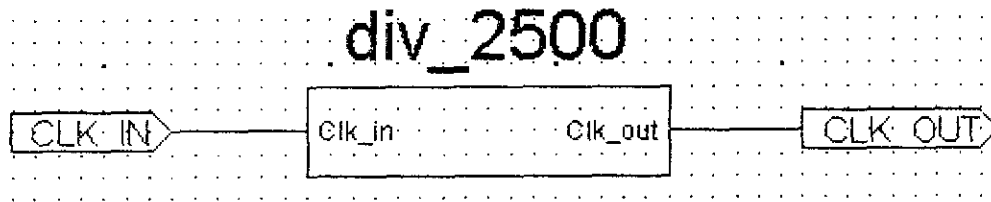
4.2 การออกแบบวงจรส่วนต่างๆ โดยใช้ภาษา VHDL

4.2.1 ส่วนของวงจร DIV_2500

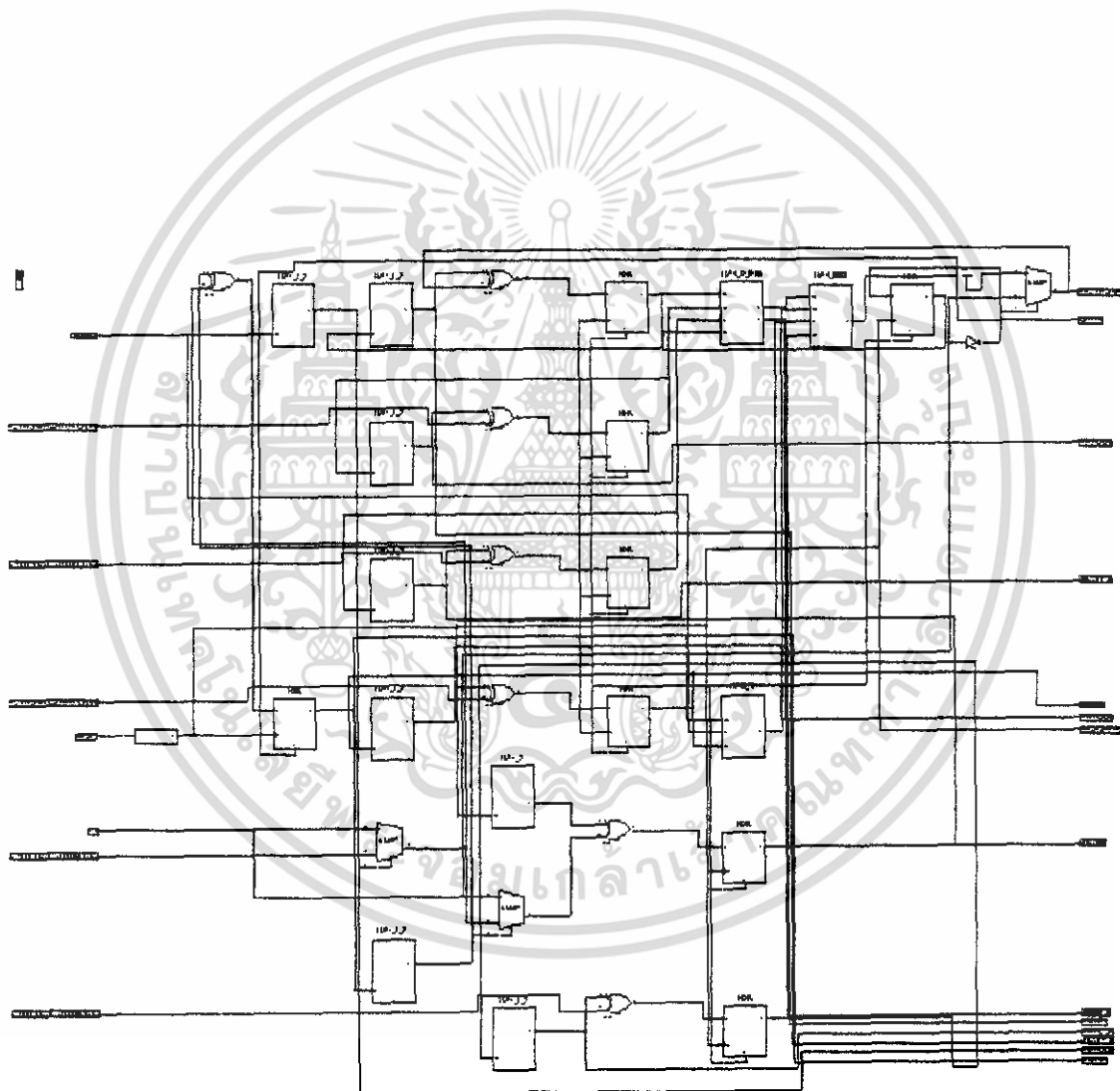
ทำหน้าที่เป็น โมดูลของวงจรหารความถี่ เนื่องจากภายในบอร์ด FPGA-3D-XC3S200 ใช้แหล่งกำเนิดแบบ Module Oscillator ความถี่ 24 MHz แต่ความถี่ที่ได้จะใช้เป็น Baud Rate ที่เราต้องการในการรับ มีค่าเท่ากับ 9600MHz เพราะฉะนั้นเราจะต้องทำการหารความถี่ที่ได้จาก Module Oscillator ลงให้เหลือเพียง 9600 MHz ซึ่งค่าของตัวหารจะเท่ากับ $24 \text{ MHz} / 9600 \text{ Hz}$ เท่ากับ 2500

มีลักษณะดังรูป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



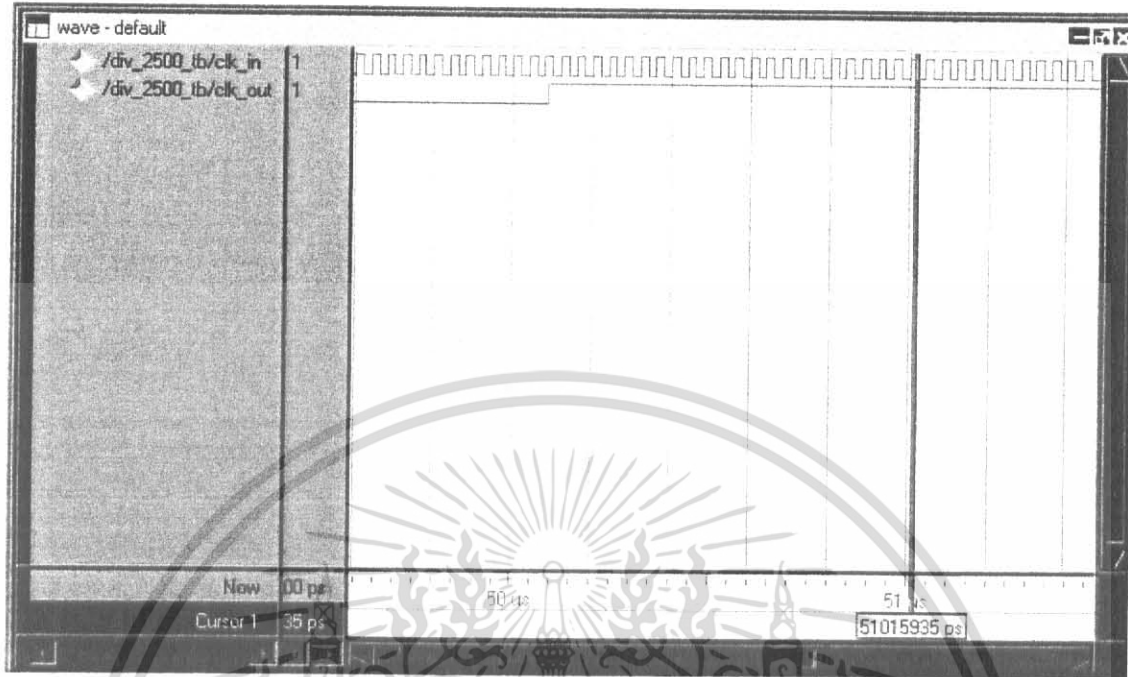
รูปที่ 4.4 สัญลักษณ์ของส่วนวงจร DIV_2500



รูปที่ 4.5 ผลการสังเคราะห์ที่เป็นวงจร DIV_2500

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

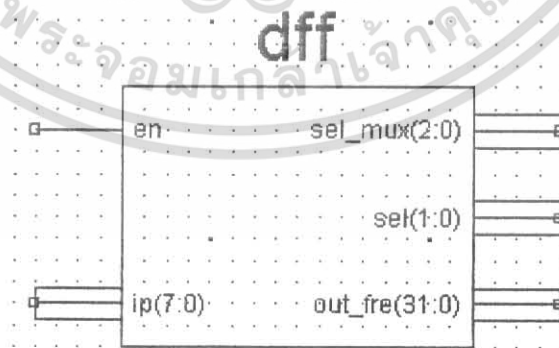
จากโปรแกรมที่เขียนขึ้นสามารถจำลองการทำงาน (simulation) ได้ดังนี้



รูปที่ 4.6 ผลการจำลองการทำงานของส่วนวงจร DIV_2500

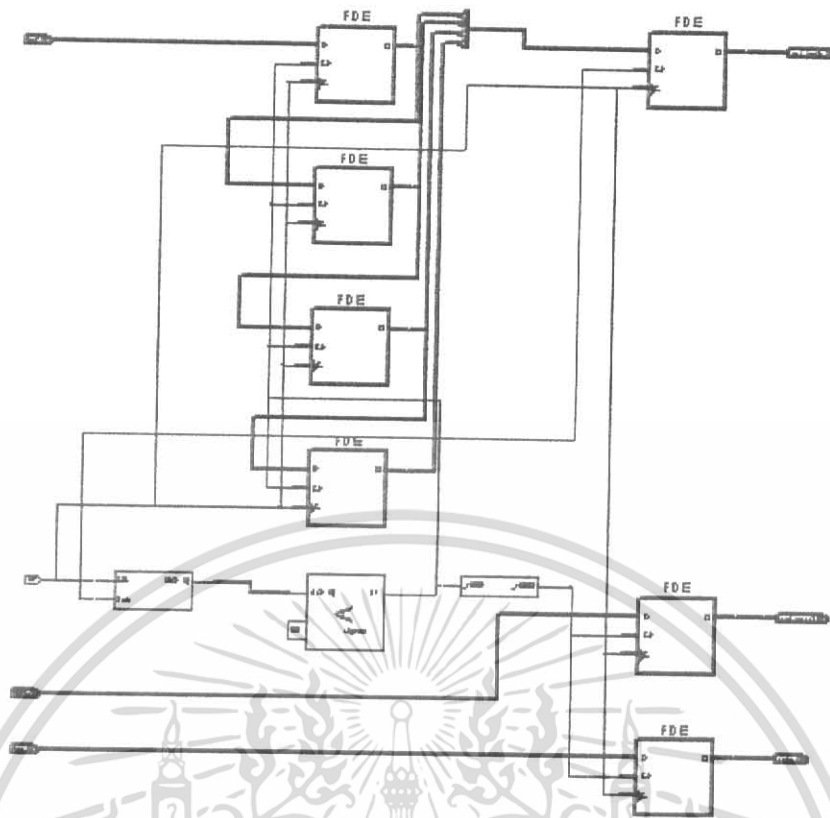
4.2.2 ส่วนของวงจร Dff

เป็น โมดูลของวงจร D Flip Flop ทำหน้าที่ Latch ค่าข้อมูลที่รับมาได้ และส่งค่าที่ได้รับมาให้กับส่วนต่างๆ ที่จะนำไปใช้งาน มีลักษณะดังรูป



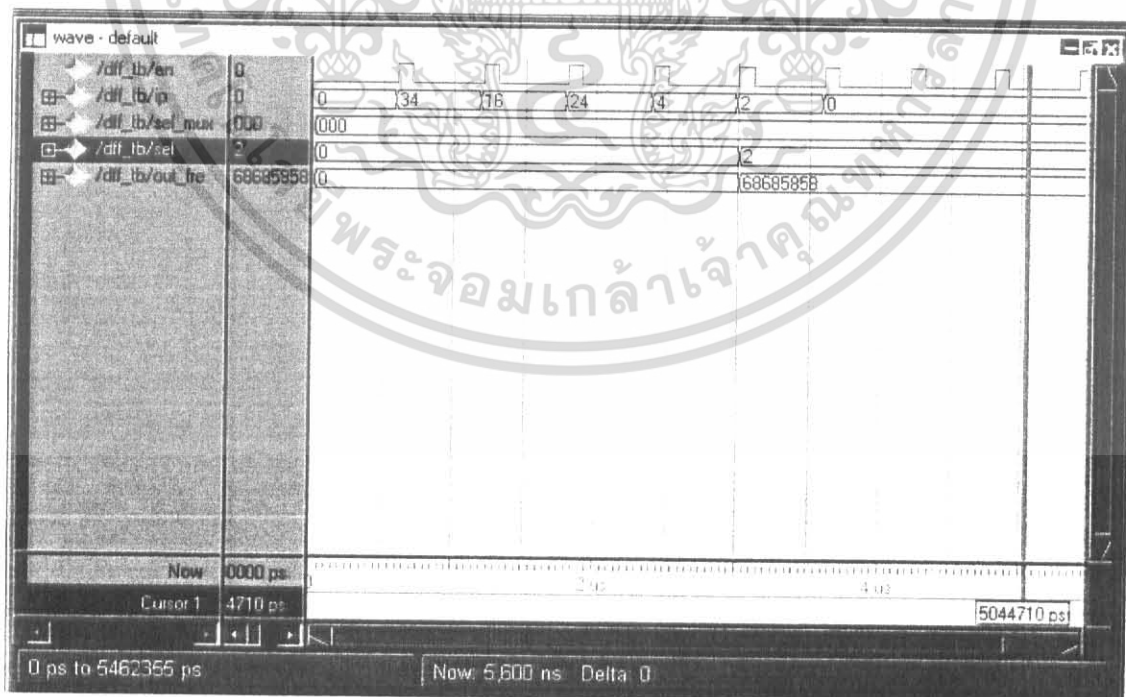
รูปที่ 4.7 สัญลักษณ์ของส่วนวงจร Dff

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.8 ผลการตั้งเครื่องเป็นวงจร Dff

จากโปรแกรมที่เขียนขึ้นสามารถจำลองการทำงาน (simulation) ได้ดังนี้

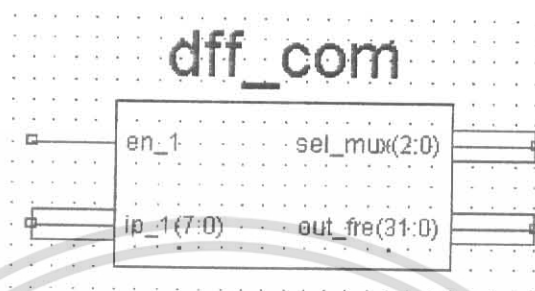


รูปที่ 4.9 ผลการจำลองการทำงานของส่วนวงจร Dff

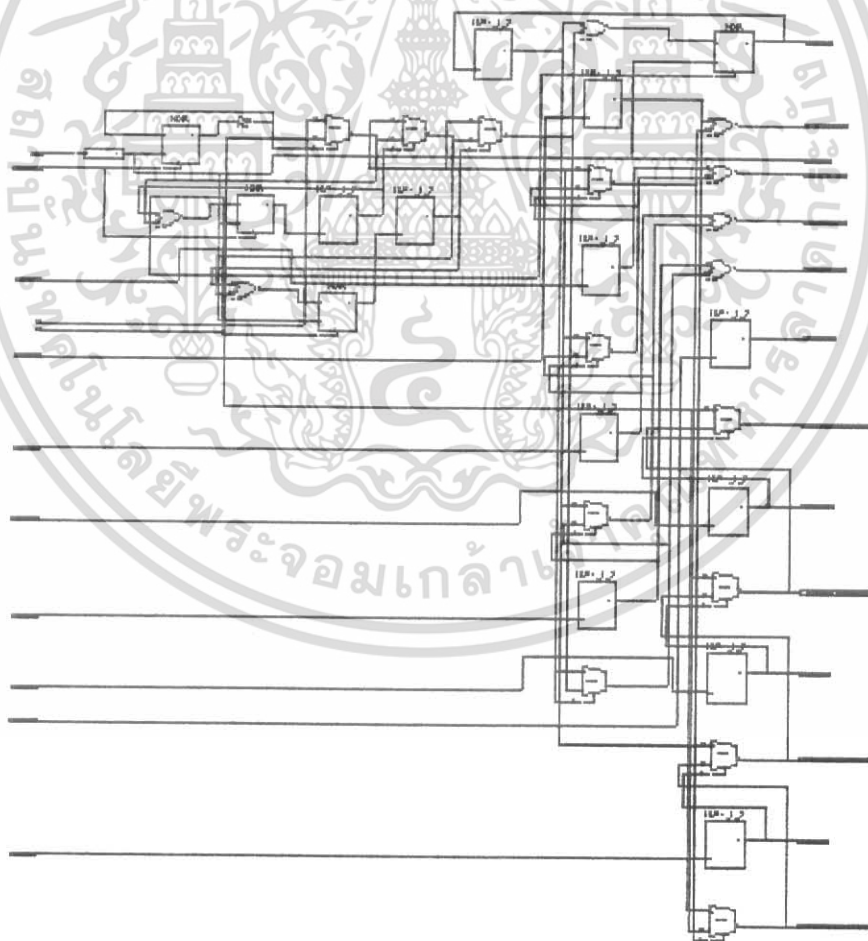
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2.3 ส่วนของวงจร Dff_Com

เป็นโมดูลของวงจร D Flip Flop ทำหน้าที่ Latch ค่าข้อมูลที่รับมาได้จากคอมพิวเตอร์ และทำการแปลงจากรหัสแอสกีที่ส่งค่ามาจากคอมพิวเตอร์ให้เป็นเลขไบนารี เพื่อส่งค่าให้กับส่วนต่างๆ ที่จะนำไปใช้งาน มีลักษณะดังรูป



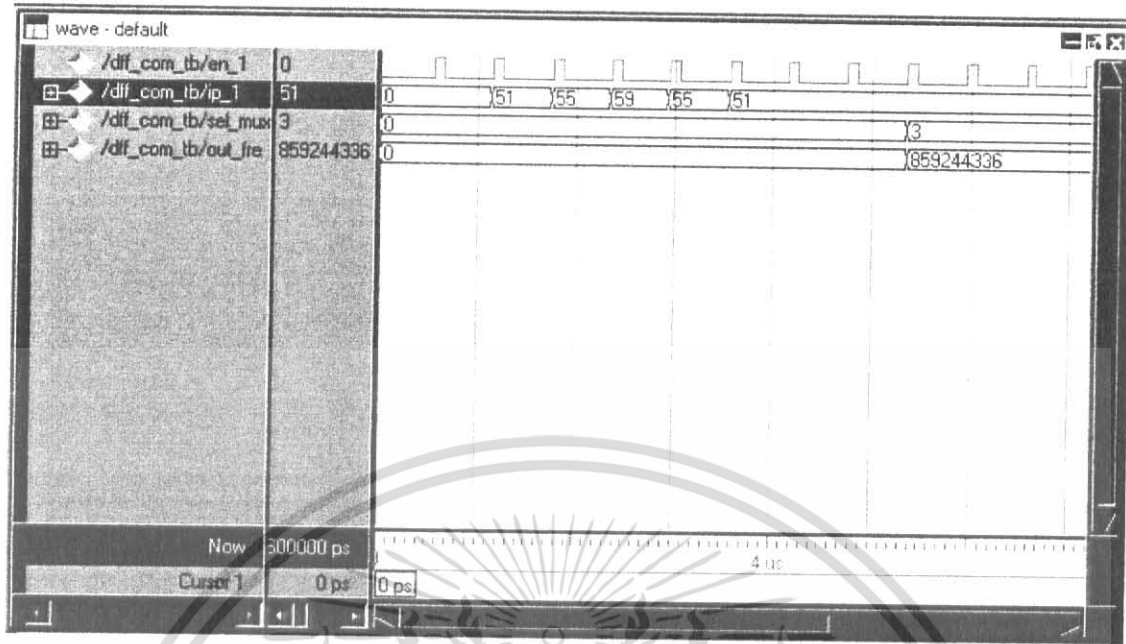
รูปที่ 4.10 สัญลักษณ์ของส่วนวงจร Dff_Com



รูปที่ 4.11 ผลการสังเคราะห์ที่เป็นวงจร Dff_Com

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

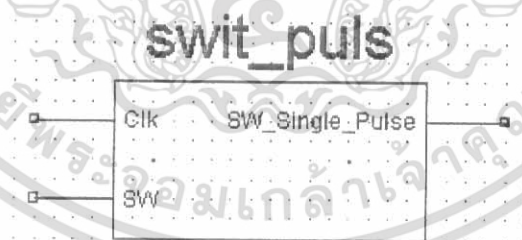
จากโปรแกรมที่เขียนขึ้นสามารถจำลองการทำงาน (simulation) ได้ดังนี้



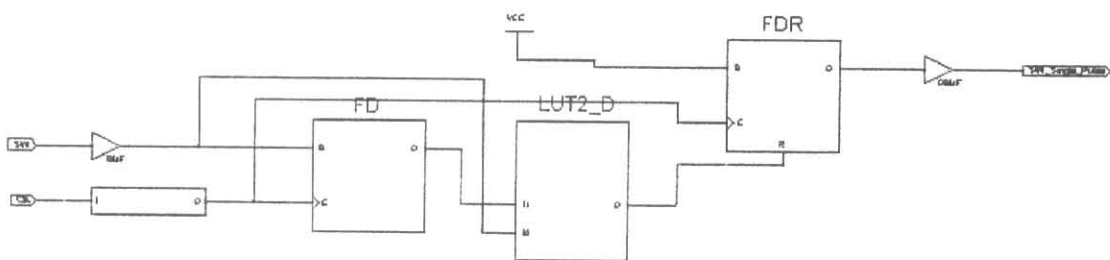
รูปที่ 4.12 ผลการจำลองการทำงานของส่วนวงจร Dff_Com

4.2.4 ส่วนของวงจร SWIT_PULSE

เป็น โมดูลรับค่าจากส่วนของ RX_EN ของส่วน SERIAL_COMMUNICATION โมดูลนี้จะทำการ Debounce สัญญาณที่รับมาและให้ Pulse ออกมา 1 ลูกเท่านั้น ขนาดของ Pulse จะเท่ากับเวลาที่ต่ออยู่กับ อินพุต CLK ซึ่งมีลักษณะดังรูป



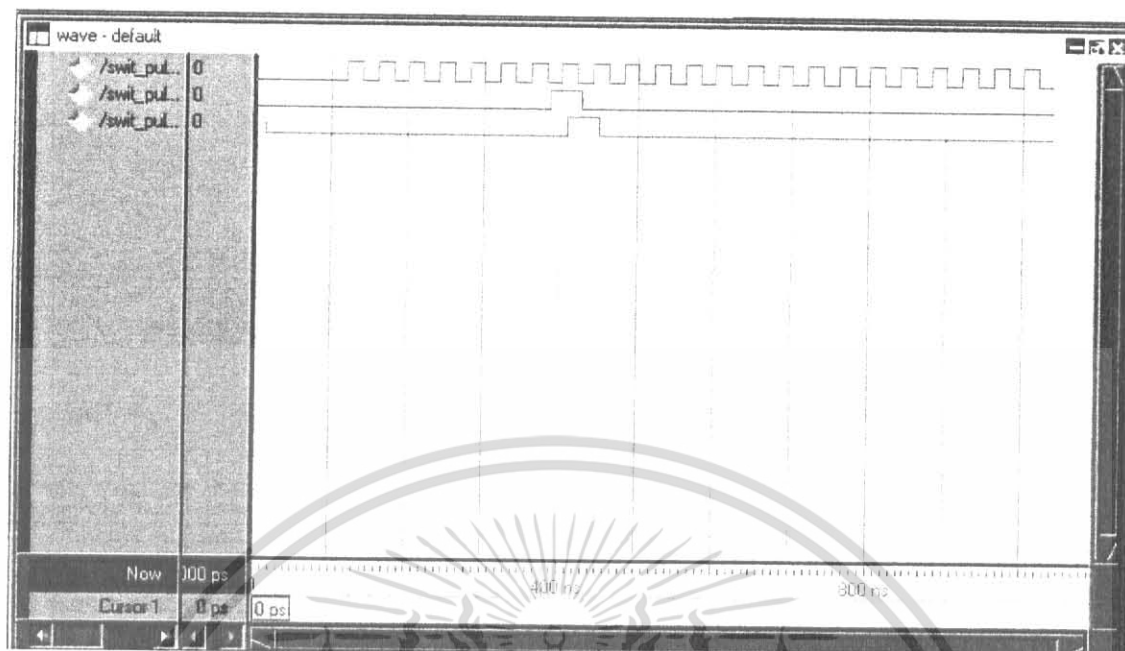
รูปที่ 4.13 สัญลักษณ์ของส่วนวงจร SWIT_PULSE



รูปที่ 4.14 ผลการสังเคราะห์ที่เป็นวงจร SWIT_PULSE

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

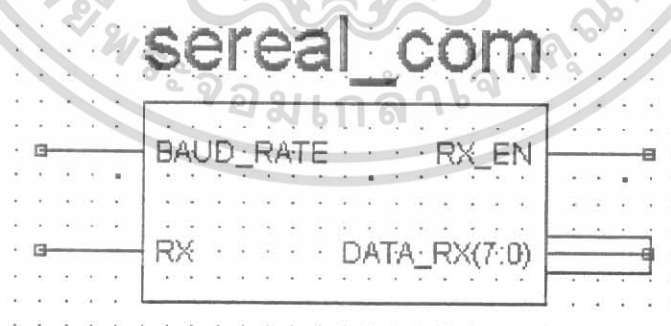
จากโปรแกรมที่เขียนขึ้นสามารถจำลองการทำงาน (simulation) ได้ดังนี้



รูปที่ 4.15 ผลการจำลองการทำงานของส่วนวงจร SWIT_PULSE

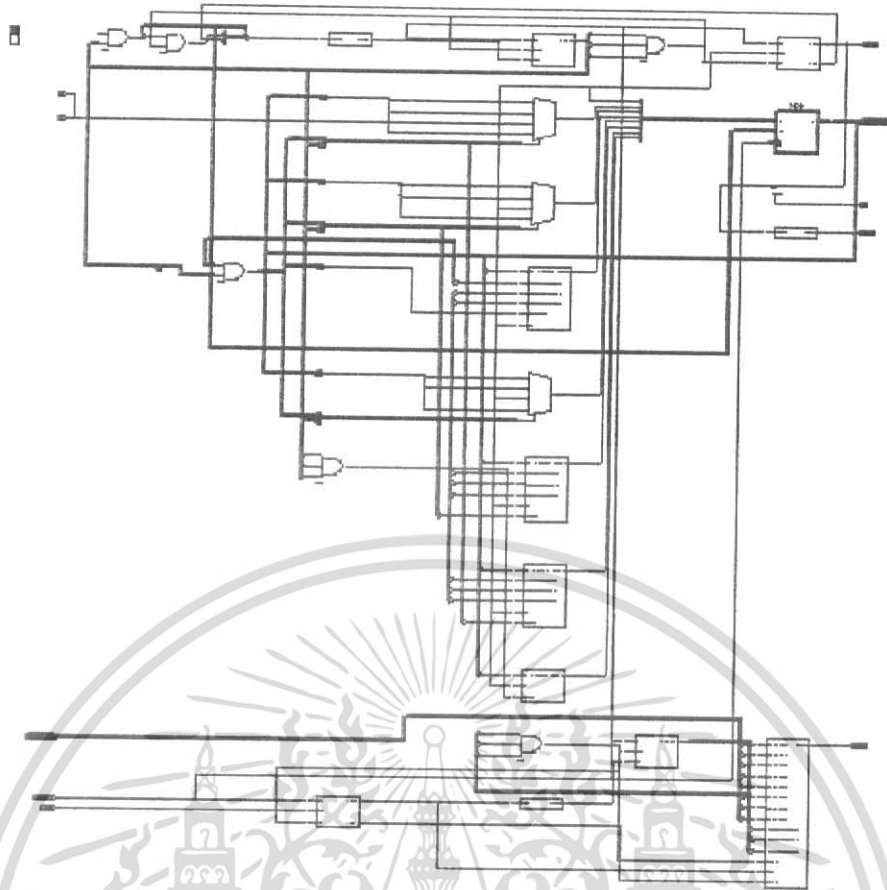
4.2.5 ส่วนของวงจร SERIAL_COMMUNICATION

เป็น โมดูลของวงจรรับ-ส่งข้อมูลแบบอนุกรม ที่รับคำมาจากพอร์ตอนุกรมของคอมพิวเตอร์ และผ่านวงจรปรับระดับแรงดันก่อนที่จะส่งเข้ามาที่เอพพีจีเอ มีลักษณะดังรูป

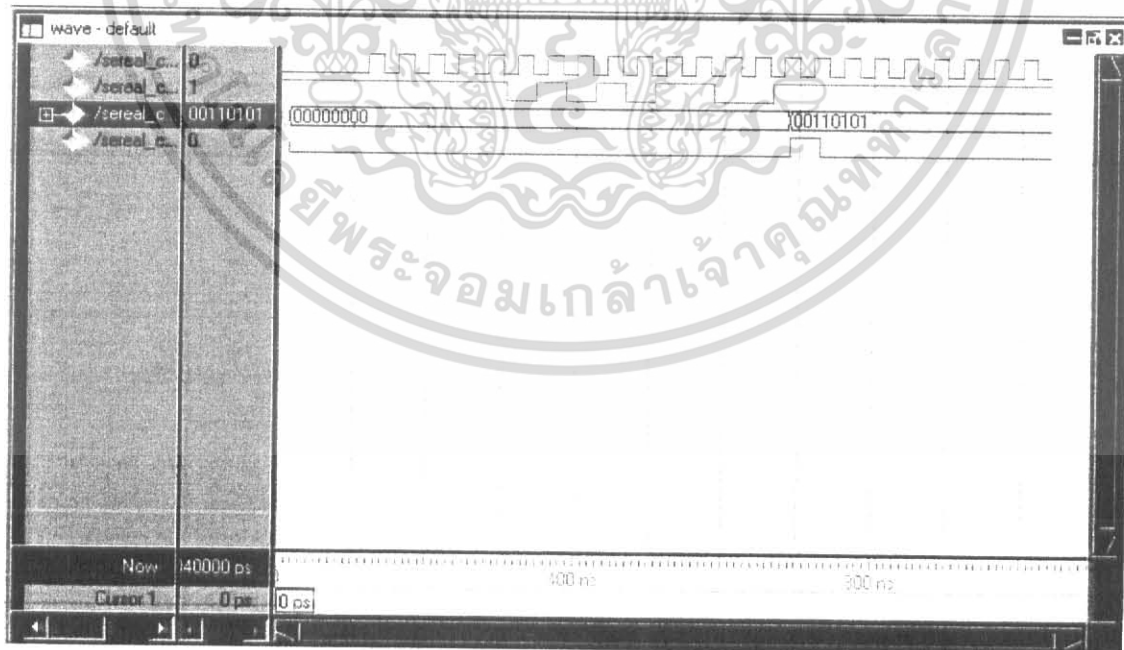


รูปที่ 4.16 สัญลักษณ์ของส่วนวงจร SERIAL_COMMUNICATION

เอกสารนี้เป็นเอกสารที่สวอนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.17 ผลการสังเคราะห์ที่เป็นวงจร SERIAL COMMUNICATION จากโปรแกรมที่เขียนขึ้นสามารถจำลองการทำงาน (simulation) ได้ดังนี้

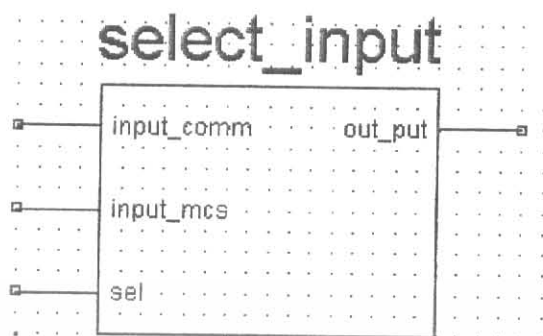


รูปที่ 4.18 ผลการจำลองการทำงานของส่วนวงจร SERIAL_COMMUNICATION

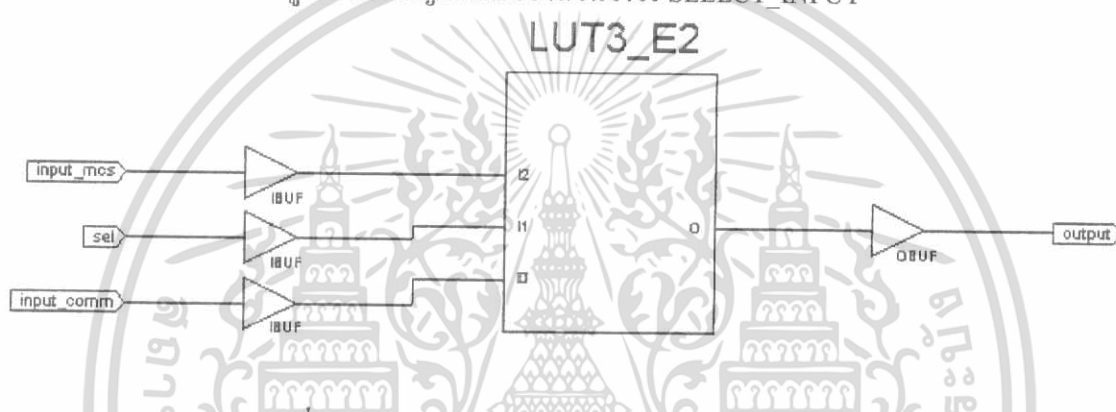
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2.6 ส่วนของวงจร SELECT_INPUT

เป็น โมดูลที่ทำหน้าที่เลือกอินพุตที่รับเข้ามาในส่วนของพอร์ตอนุกรมว่ามาคอมพิวเตอร์หรือมาจาก MCS-51 มีลักษณะดังรูป



รูปที่ 4.19 สัญลักษณ์ของส่วนวงจร SELECT_INPUT



รูปที่ 4.20 ผลการสังเคราะห์เป็นวงจร SELECT_INPUT

จาก โปรแกรมที่เขียนขึ้นสามารถจำลองการทำงาน (simulation) ได้ดังนี้

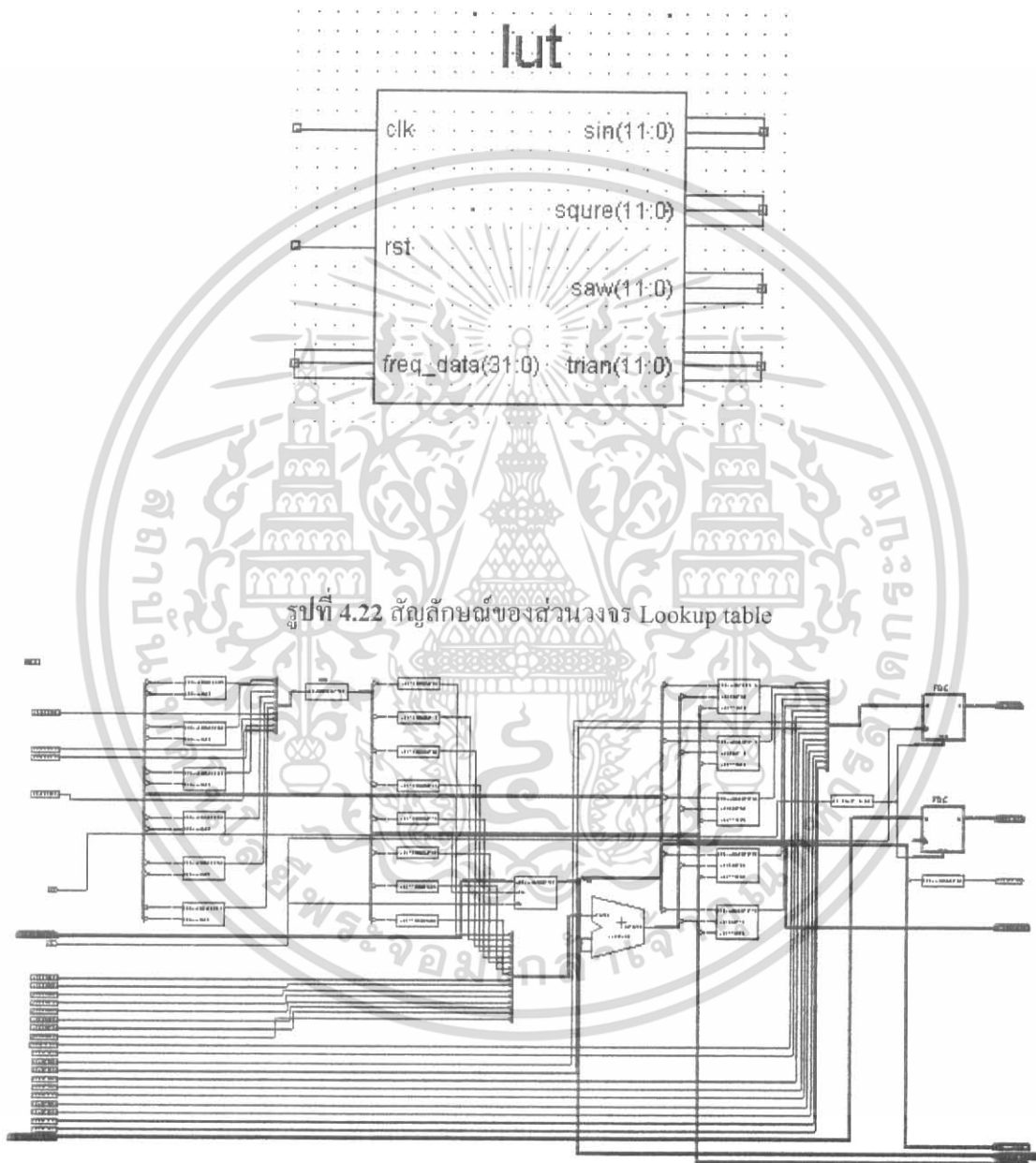


รูปที่ 4.21 ผลการจำลองการทำงานของส่วนวงจร SELECT_INPUT

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2.7 ส่วนของวงจร Lookup table

เป็นโมดูลที่ทำหน้าที่ผลิตสัญญาณซายน์, สามเหลี่ยม, สี่เหลี่ยมและฟันเลื่อย โดยใช้วิธีตารางเปิดดู (Lookup table)

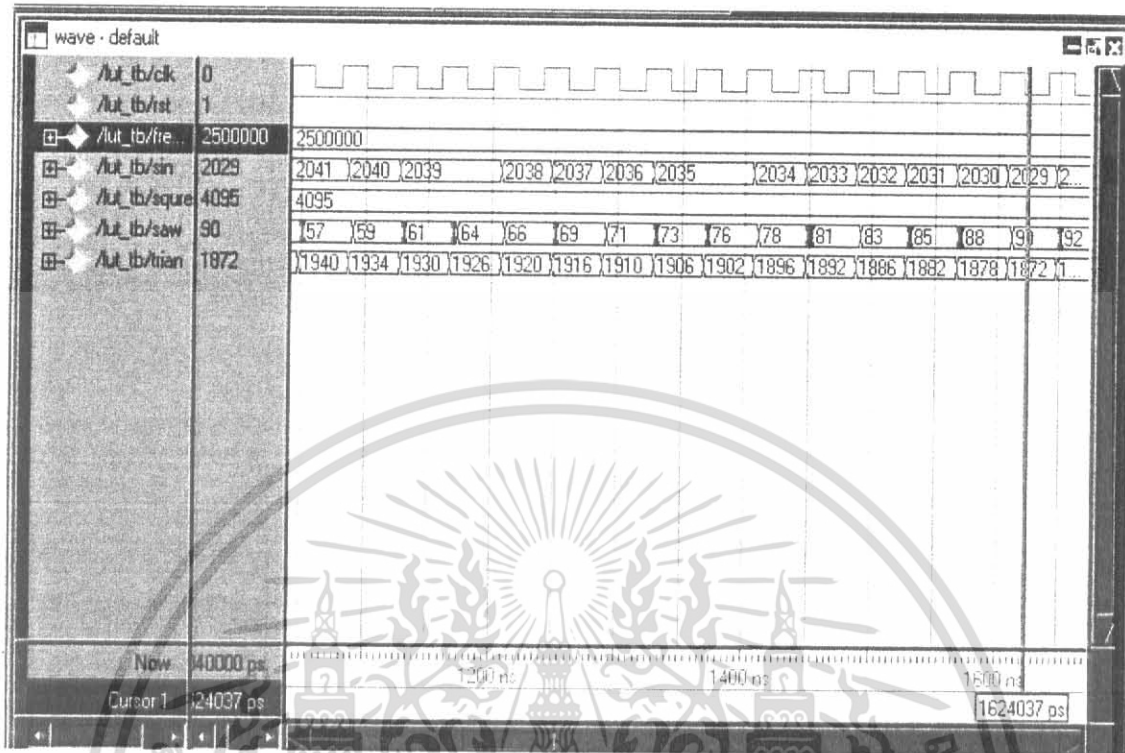


รูปที่ 4.22 สัญลักษณ์ของส่วนวงจร Lookup table

รูปที่ 4.23 ผลการสังเคราะห์ที่เป็นวงจร Lookup table

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

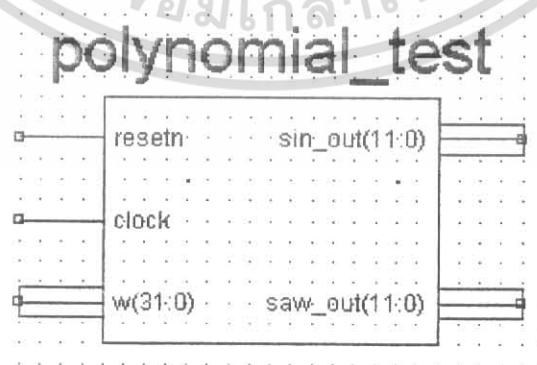
จากโปรแกรมที่เขียนขึ้นสามารถจำลองการทำงาน (simulation) ได้ดังนี้



รูปที่ 4.24 ผลการจำลองการทำงานของส่วนวงจร Lookup table

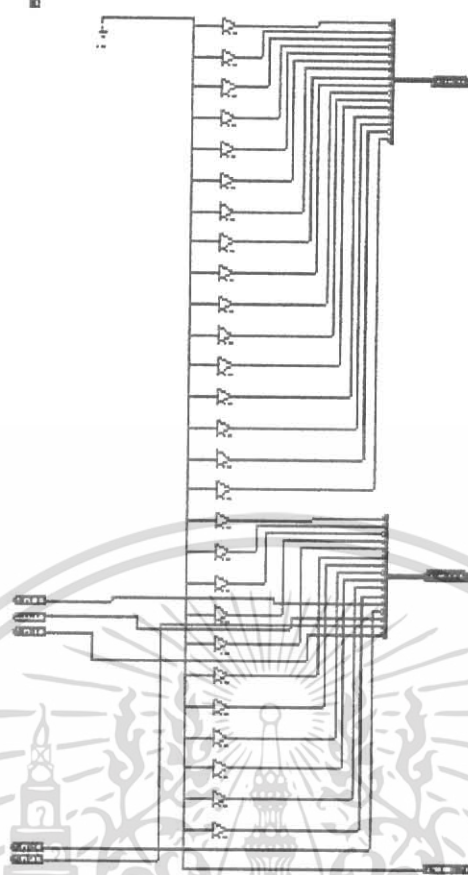
4.2.8 ส่วนของวงจร POLYNOMIAL APPROXIMATION

เป็น โมดูลที่ทำหน้าที่ผลิตสัญญาณไซน์, โดยใช้วิธีของ POLYNOMIAL APPROXIMATION

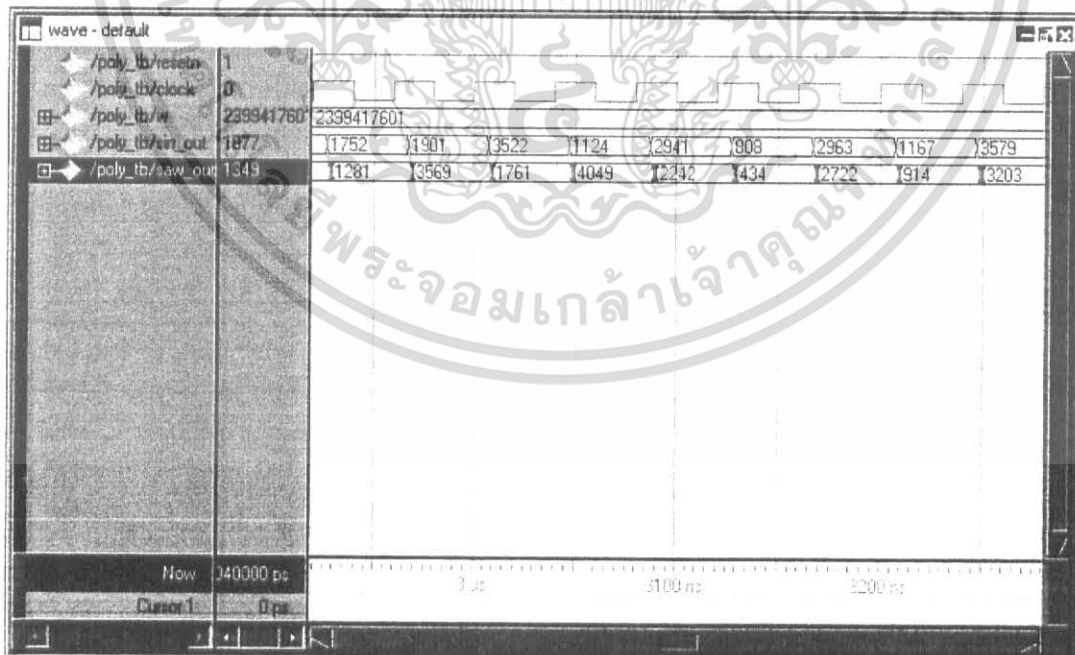


รูปที่ 4.25 สัญลักษณ์ของส่วนวงจร POLYNOMIAL APPROXIMATION

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



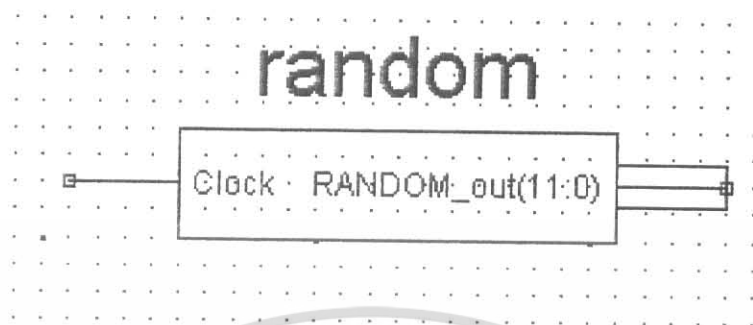
รูปที่ 4.26 ผลการสังเคราะห์ที่เป็นวงจร POLYNOMIAL APPROXIMATION จากโปรแกรมที่เขียนขึ้นสามารถจำลองการทำงาน (simulation) ได้ดังนี้



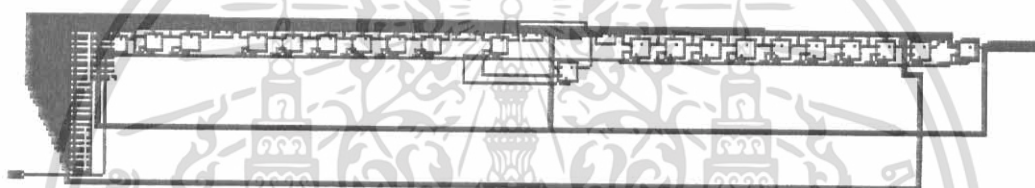
รูปที่ 4.27 ผลการจำลองการทำงานของส่วนวงจร วงจร POLYNOMIAL APPROXIMATION เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2.9 ส่วนของวงจร RANDOM

เป็น โมดูลของ RANDOM ทำหน้าที่ผลิตสัญญาณ RANDOM โดยใช้โครงสร้างของ Linear Feedback Shift Register (LFSR) มีลักษณะดังรูป

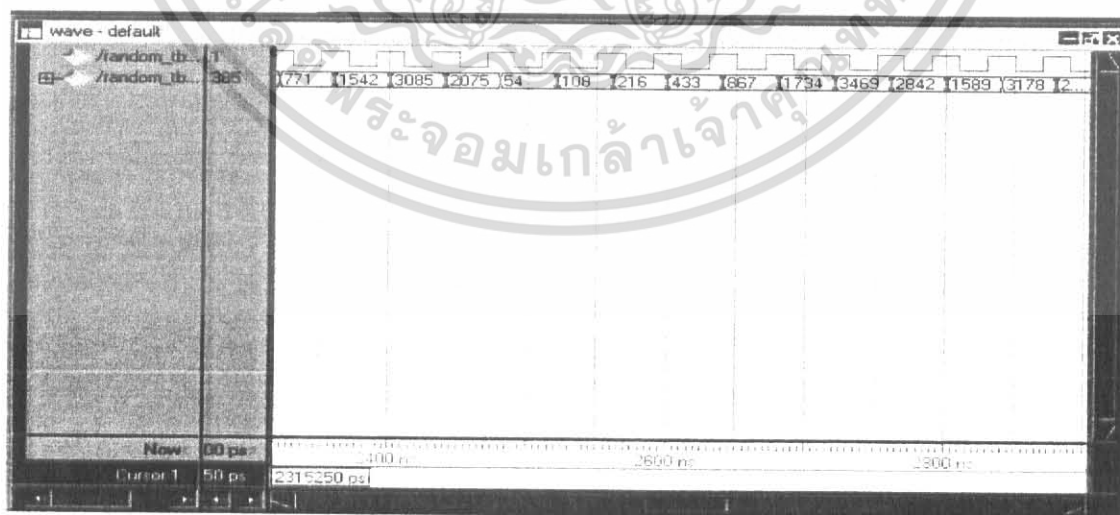


รูปที่ 4.28 สัญลักษณ์ของส่วนวงจร RANDOM



รูปที่ 4.29 ผลการจำลองการทำงานของส่วนวงจร RANDOM

จาก โปรแกรมที่เขียนขึ้นสามารถจำลองการทำงาน (simulation) ได้ดังนี้

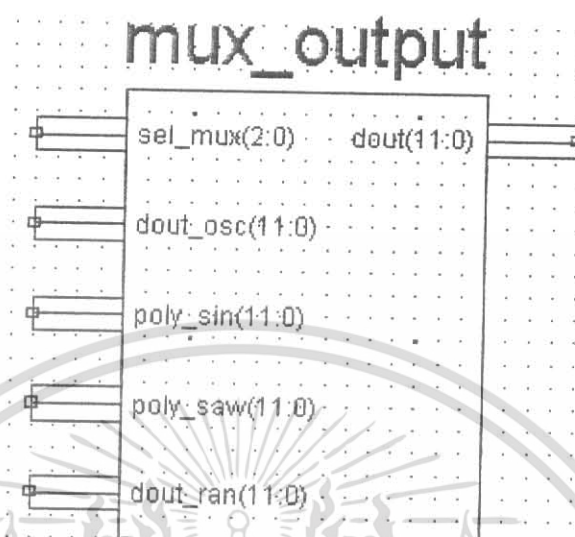


รูปที่ 4.30 ผลการจำลองการทำงานของส่วนวงจร RANDOM

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

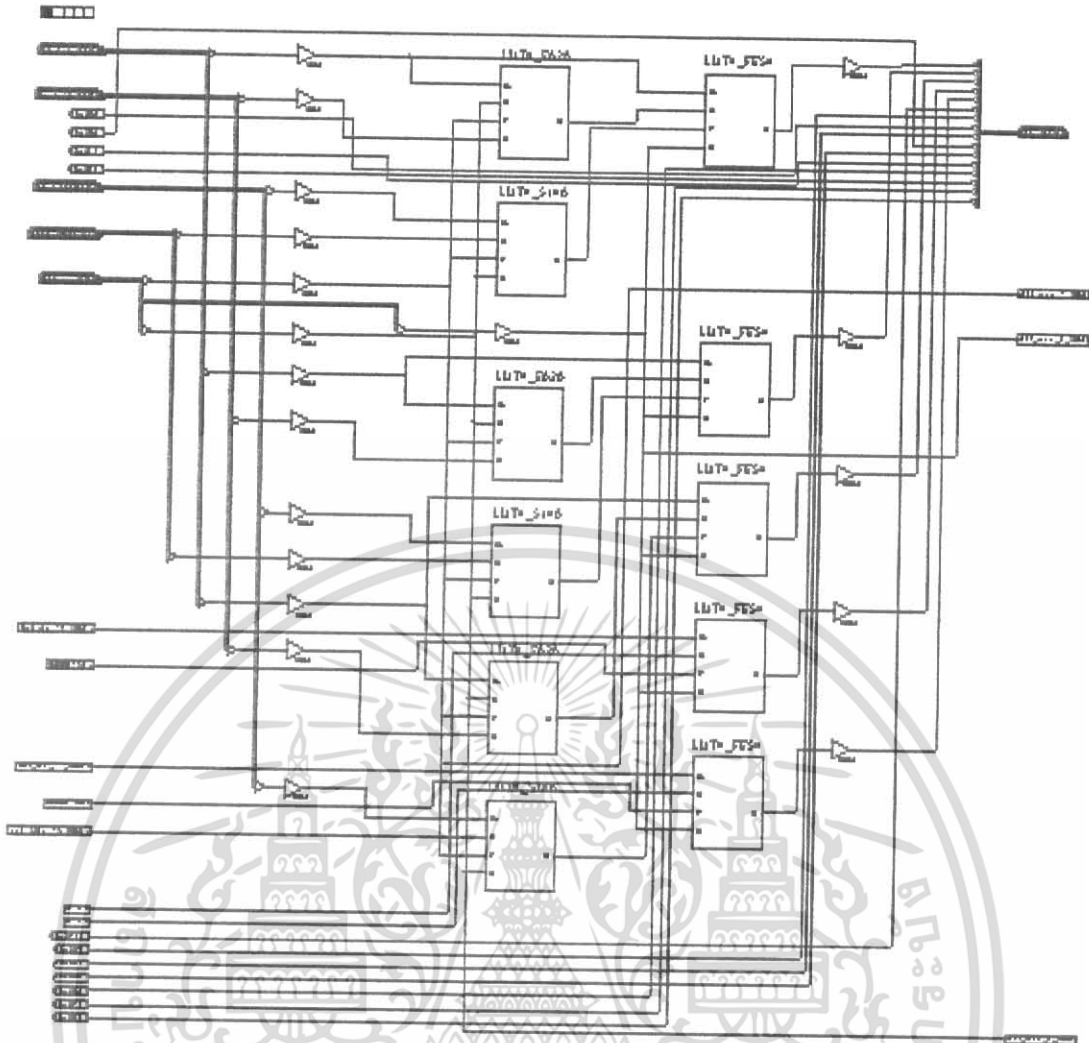
4.2.10 ส่วนของวงจร MUX_OUTPUT

เป็น โมดูลที่ทำหน้าที่ทำการเลือกว่าจะให้เอาต์พุตระหว่าง วิธีใช้ตารางเปิดดู, วิธี polinomial หรือของ LFSR ออกที่เอาต์พุตตามที่ต้องการ

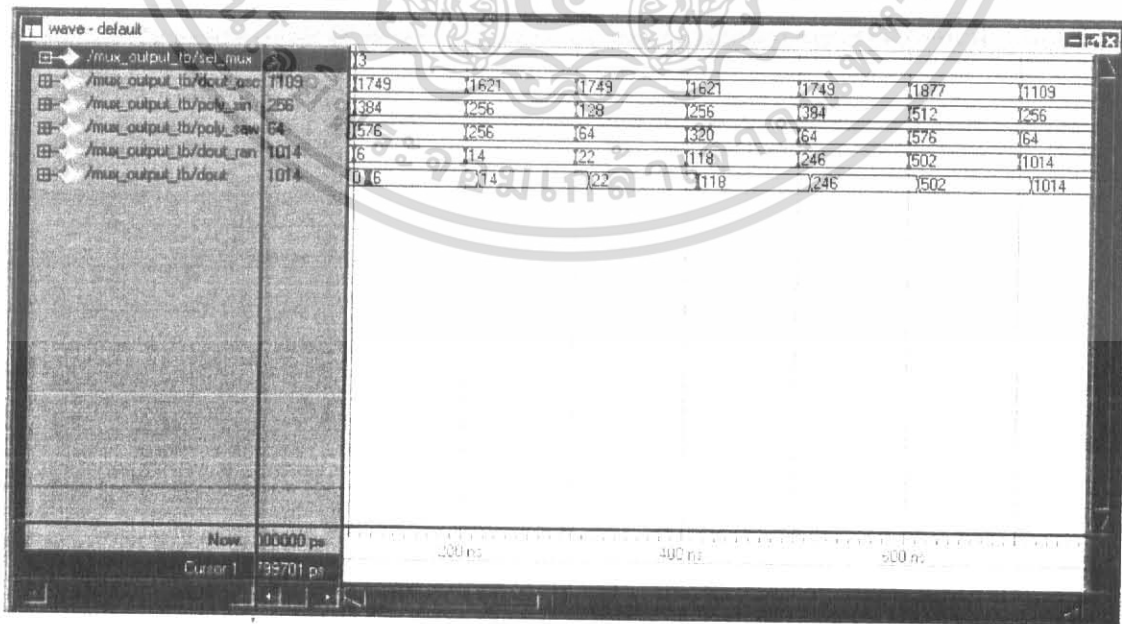


รูปที่ 4.31 สัญลักษณ์ของส่วนวงจร MUX_OUTPUT

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.32 ผลการสังเคราะห์เป็นวงจร MUX OUTPUT จากโปรแกรมที่เขียนขึ้นสามารถจำลองการทำงาน (Simulation) ได้ดังนี้

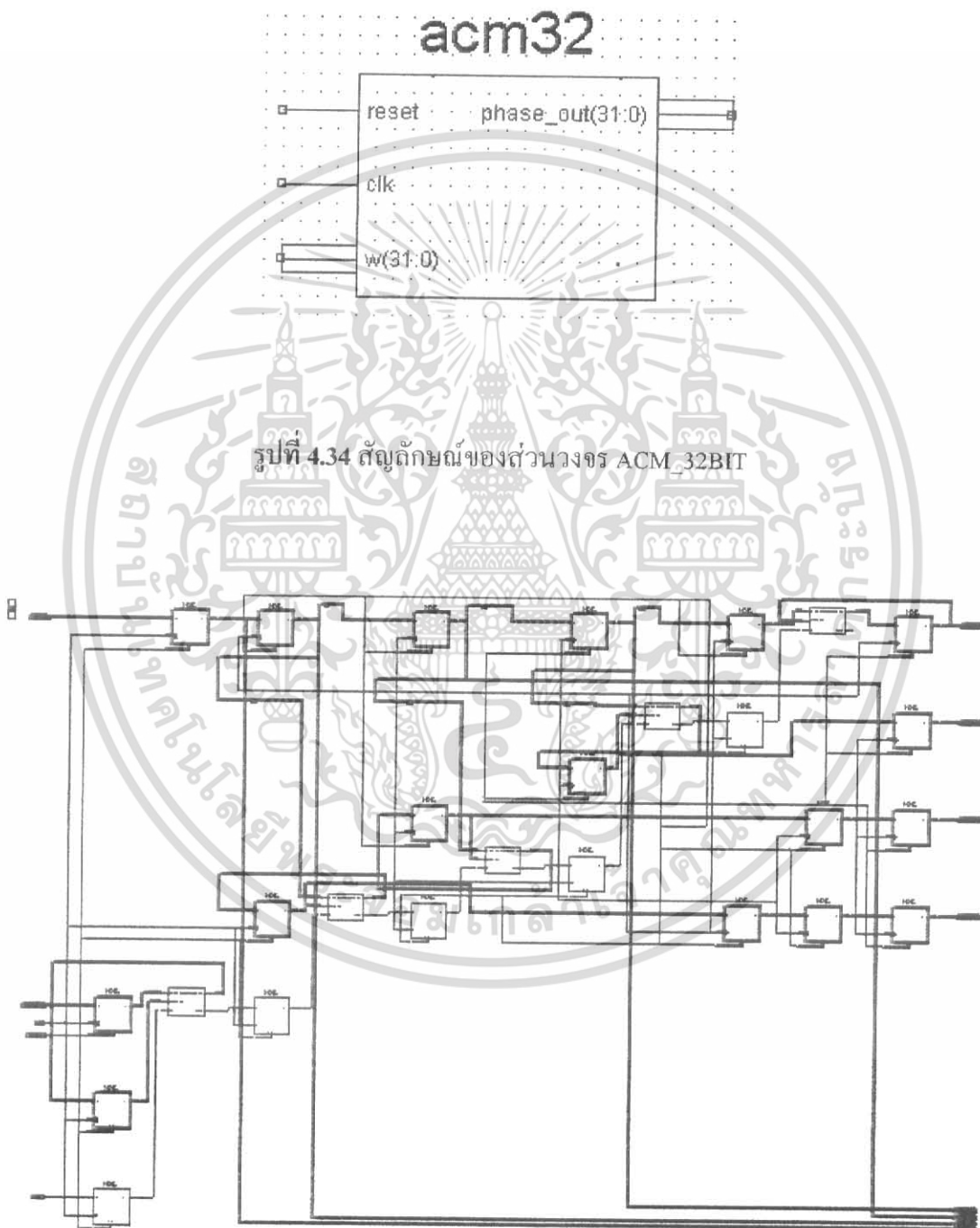


รูปที่ 4.33 ผลการจำลองการทำงานของส่วนวงจร MUX_OUTPUT

เอกสารนี้เป็นเอกสารที่สวอนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2.11 ส่วนของวงจร ACM_32BIT

วงจรสะสมเฟส (Accumulator) ซึ่งประกอบไปด้วยวงจรวกแบบคิดตัวทดล่วงหน้าขนาด 4 บิต (4-bit Carry Lookahead Adder) จำนวน 8 ชุด มาต่อกันในลักษณะสายท่อ (Pipeline) โดยวงจรสะสมเฟสจะทำการบวกค่าควบคุมความถี่ (Frequency control word, W) ที่ถูกเก็บไว้ในรีจิสเตอร์ความถี่เข้ากับค่าผลบวกก่อนหน้าที่ถูกเก็บไว้ในรีจิสเตอร์เฟส

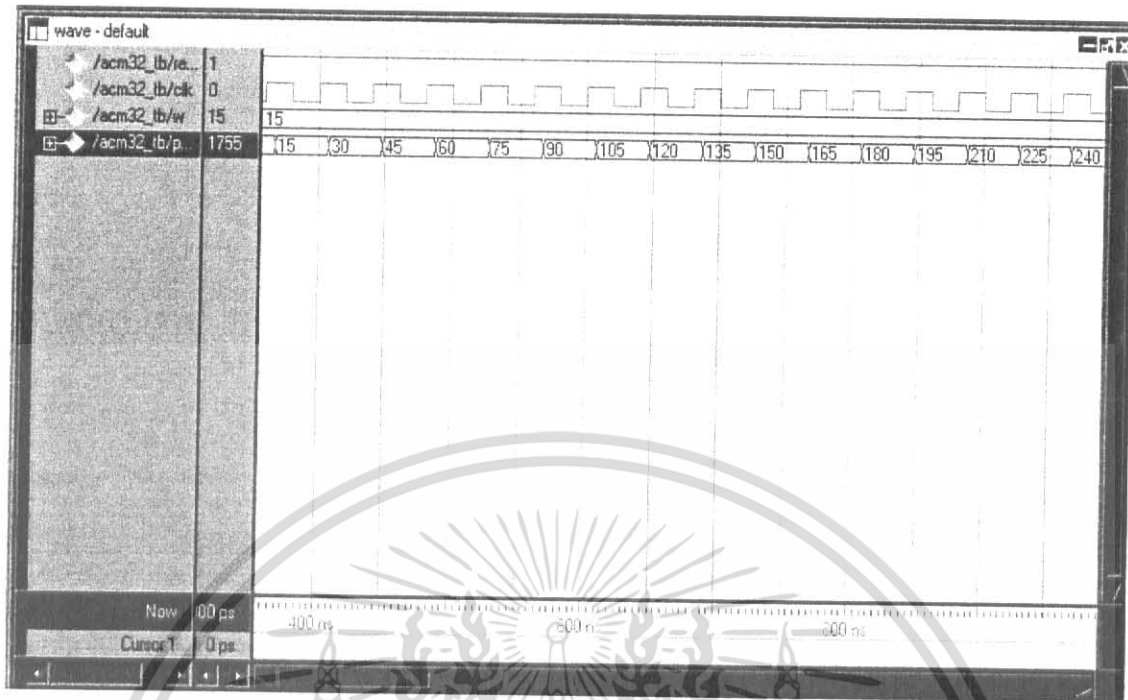


รูปที่ 4.34 สัญลักษณ์ของส่วนวงจร ACM_32BIT

รูปที่ 4.35 ผลการสังเคราะห์เป็นวงจร ACM_32BIT

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

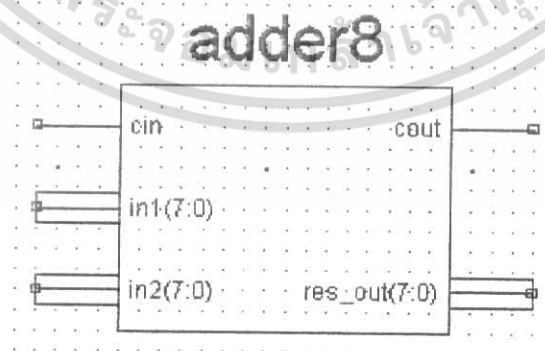
จากโปรแกรมที่เขียนขึ้นสามารถจำลองการทำงาน (simulation) ได้ดังนี้



รูปที่ 4.36 ผลการจำลองการทำงานของส่วนวงจร ACM_32BIT

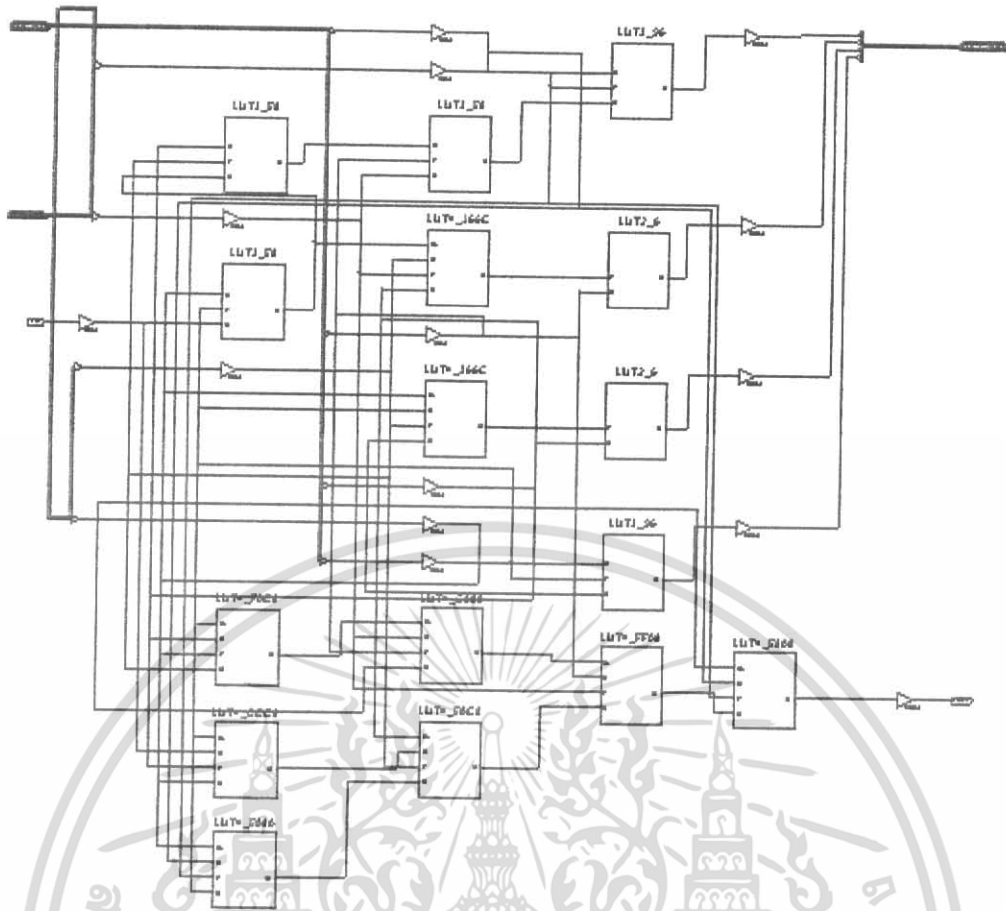
4.2.12 ส่วนของวงจร (8-bit Carry Lookahead Adder)

วงจรบวกแบบคิดตัวทดล่วงหน้าขนาด 8 บิต (8-bit Carry Lookahead Adder) จำนวน 4 ชุด มาต่อกันในลักษณะสายท่อ (Pipeline) เพื่อเพิ่มความเร็วในการทำงานของวงจร

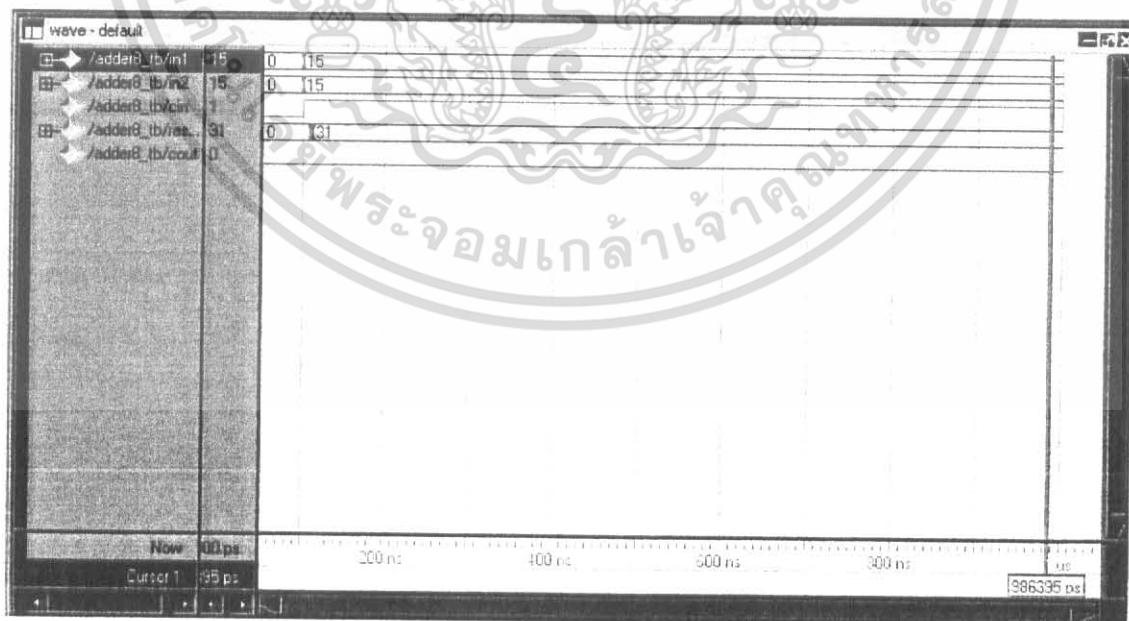


รูปที่ 4.37 สัญลักษณ์ของส่วนวงจร ADDER 8 BIT

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



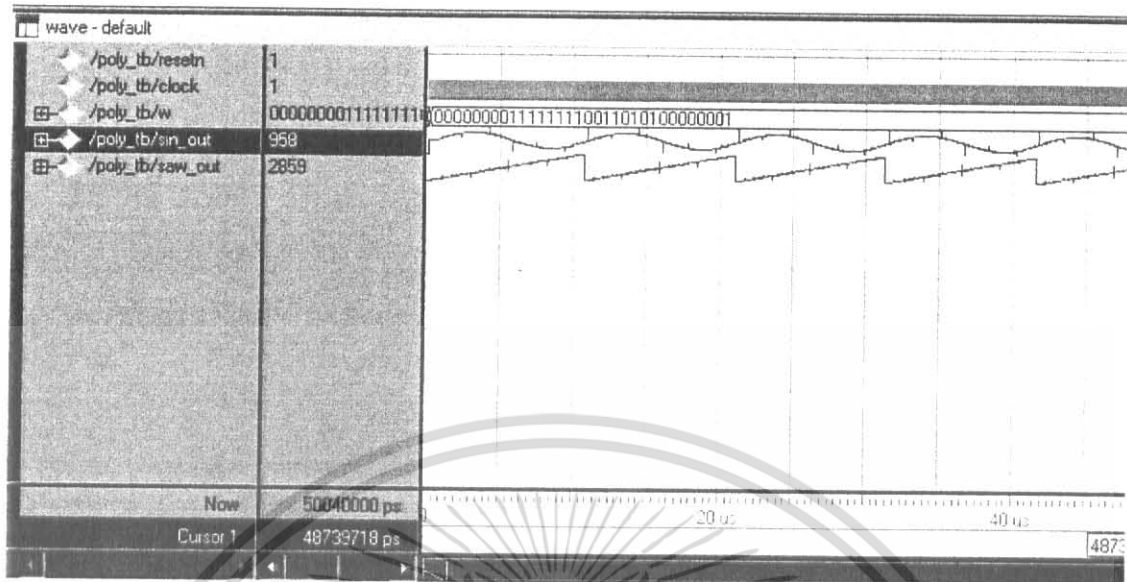
รูปที่ 4.38 ผลการสังเคราะห์ที่เป็นวงจร ADDER 8 BIT
จากโปรแกรมที่เขียนขึ้นสามารถจำลองการทำงาน (simulation) ได้ดังนี้



รูปที่ 4.39 ผลการจำลองการทำงานของส่วนวงจร ADDER 8 BIT

เอกสารนี้เป็นเอกสารที่สวอนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

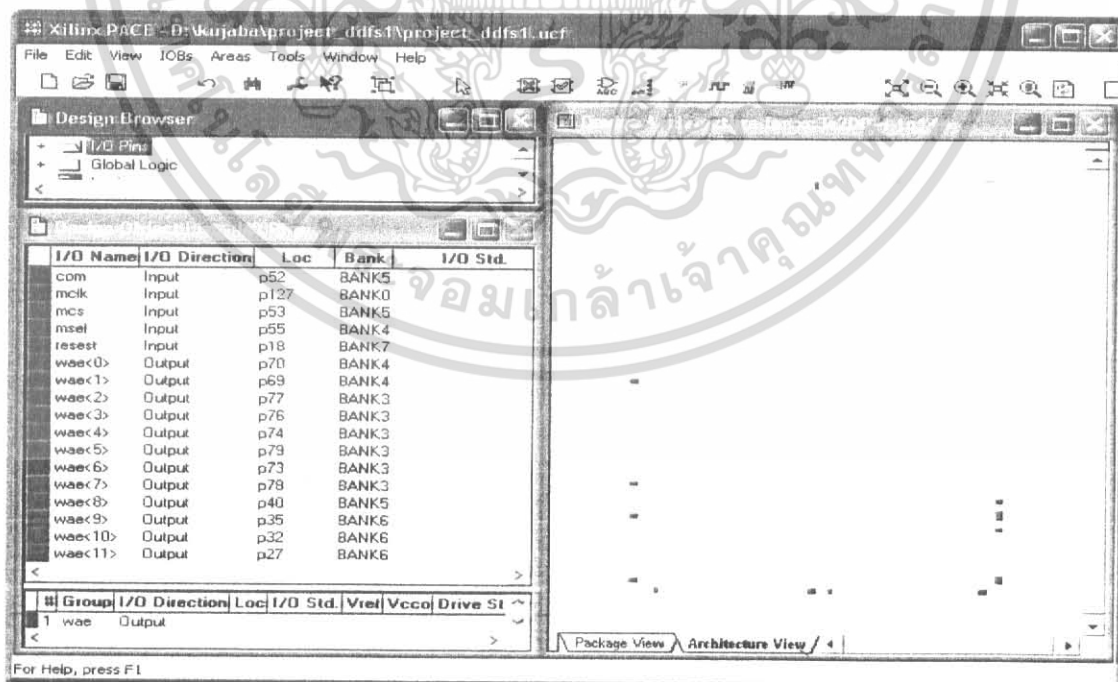
จากวงจรที่ออกแบบสามารถจำลองการทำงาน (simulation) ได้ดังนี้



รูปที่ 4.42 ผลการจำลองการทำงานของวงจร DDFS

4.4 การจัดตำแหน่งขา FPGA

การกำหนดขาให้ FPGA เมื่อจัดตำแหน่งขาเป็นที่เรียบร้อยแล้ว ให้ทำการบันทึกและ Compile อีกครั้ง เมื่อ Compile เสร็จแล้วให้ทำการโปรแกรมวงจรทดสอบที่ได้ออกแบบมาลงในบอร์ดทดลอง



รูปที่ 4.43 แสดงการกำหนดขาให้ FPGA

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

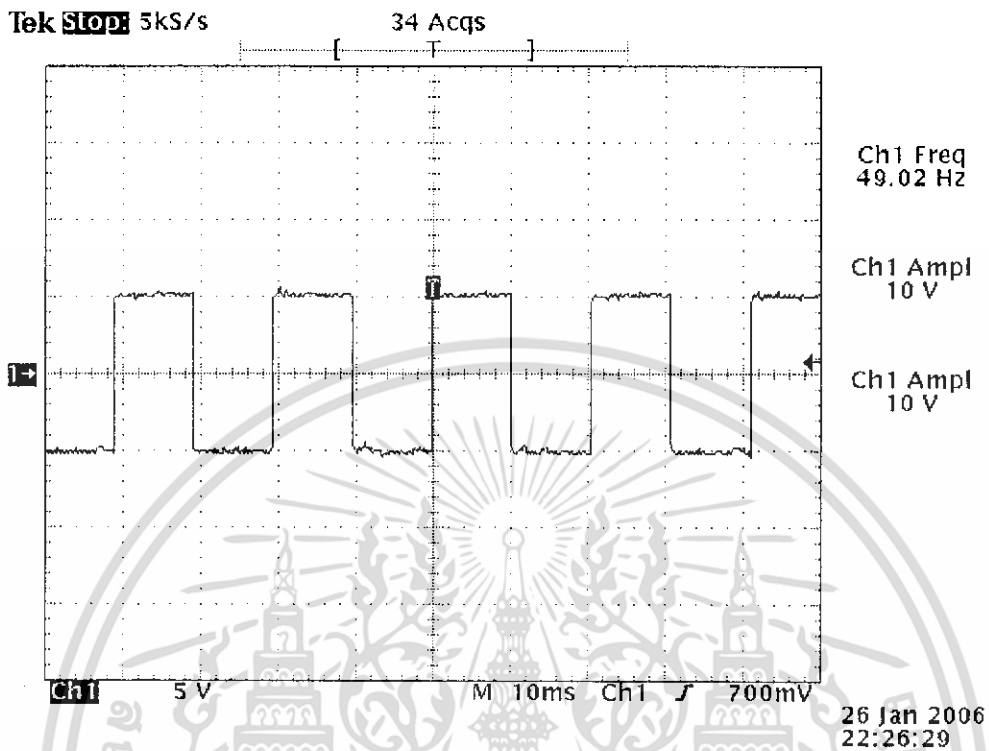


รูปที่ 4.44 แสดงการโปรแกรมวงจรลง FPGA และพร้อมใช้งาน

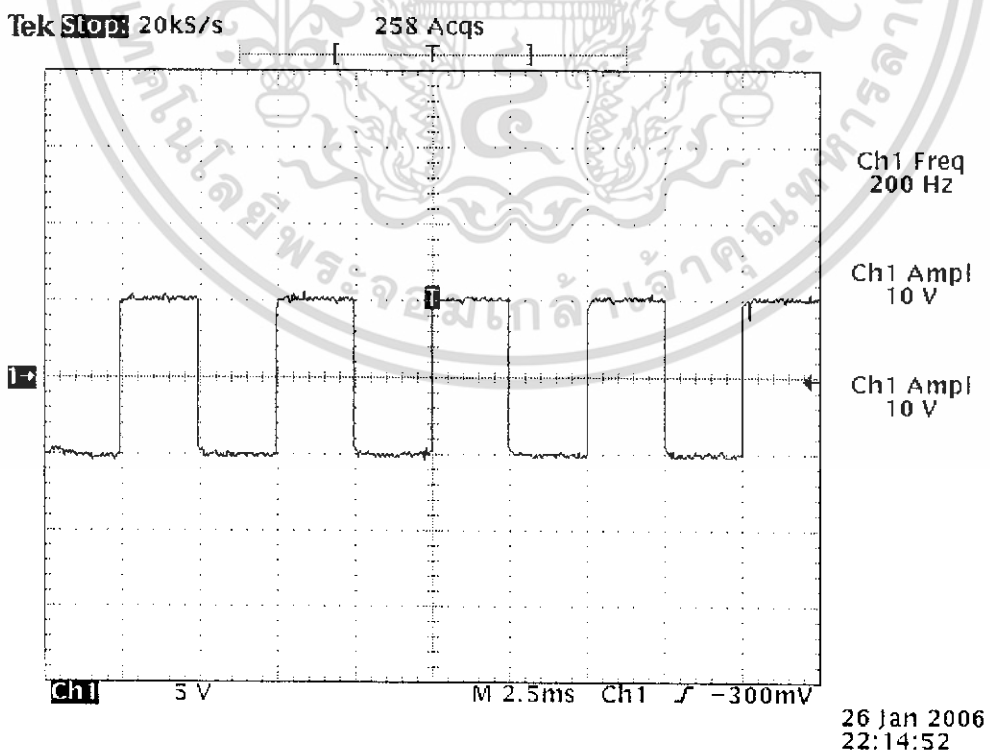
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.5 ผลการทดลองการทำงานของ DDS OSCILATOR โดยใช้วิธีการตารางเปิดดู (Lookup-Table)

4.5.1 การทดสอบการกำเนิดสัญญาณสี่เหลี่ยมที่มีความถี่ต่างๆ

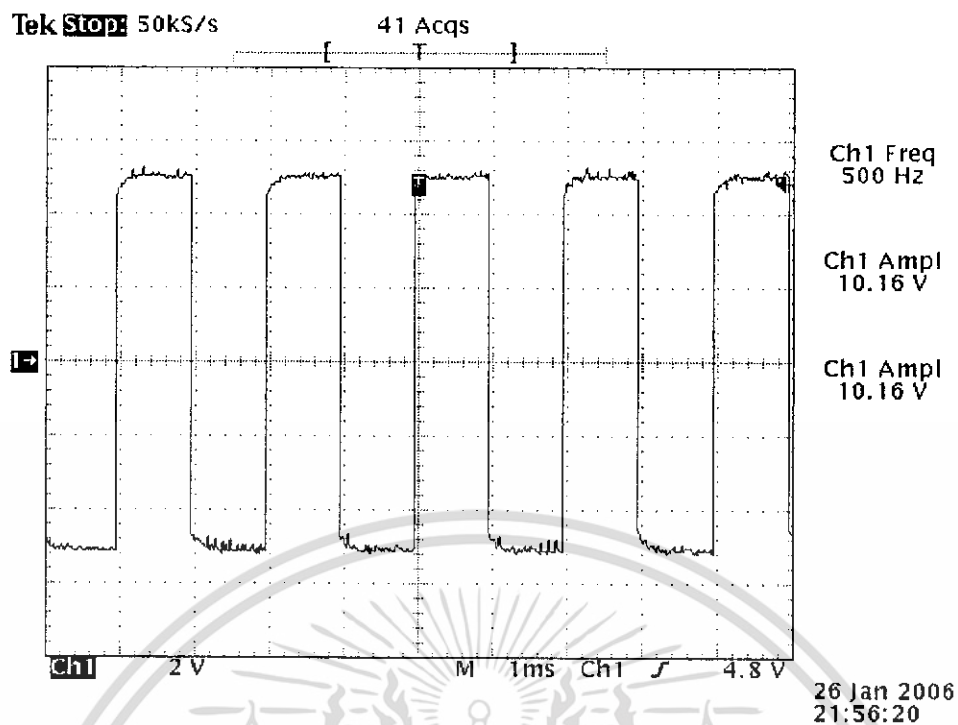


ก) รูปแสดงสัญญาณสี่เหลี่ยมที่มีความถี่ 50 Hz

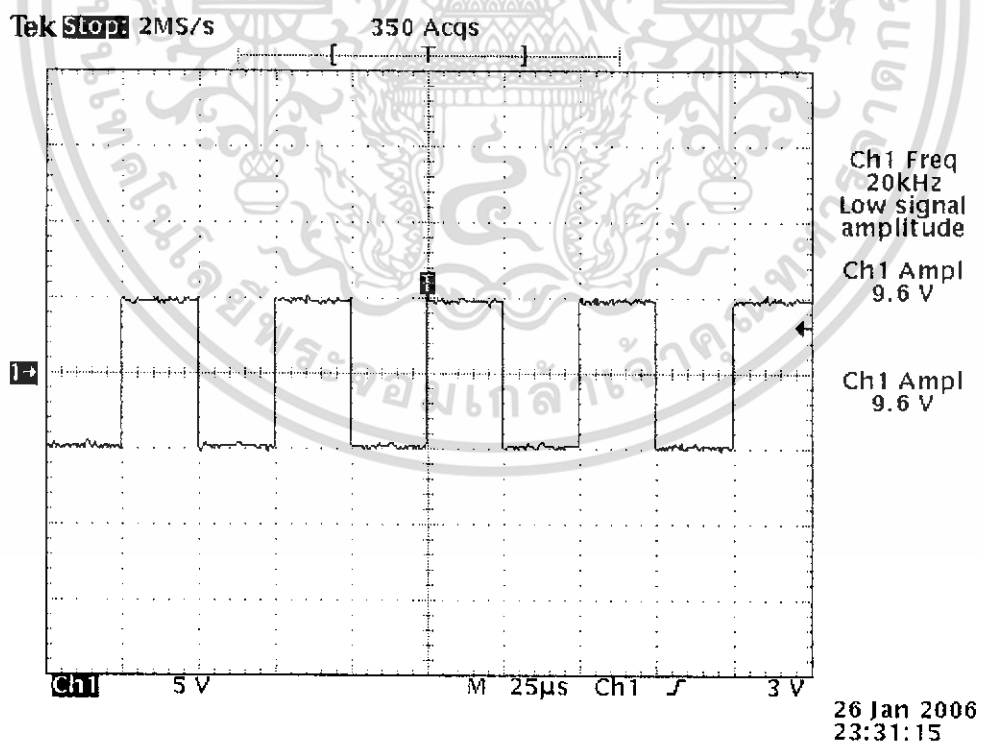


ข) รูปแสดงสัญญาณสี่เหลี่ยมที่มีความถี่ 200 Hz

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

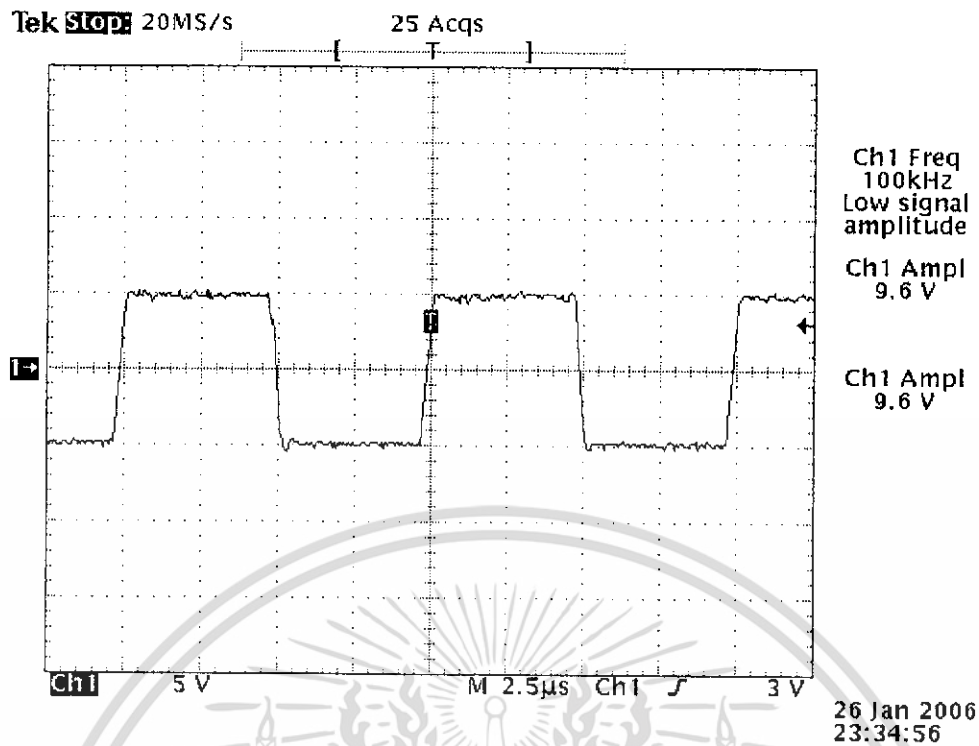


ค) รูปแสดงสัญญาณสี่เหลี่ยมที่ความถี่ 500 Hz

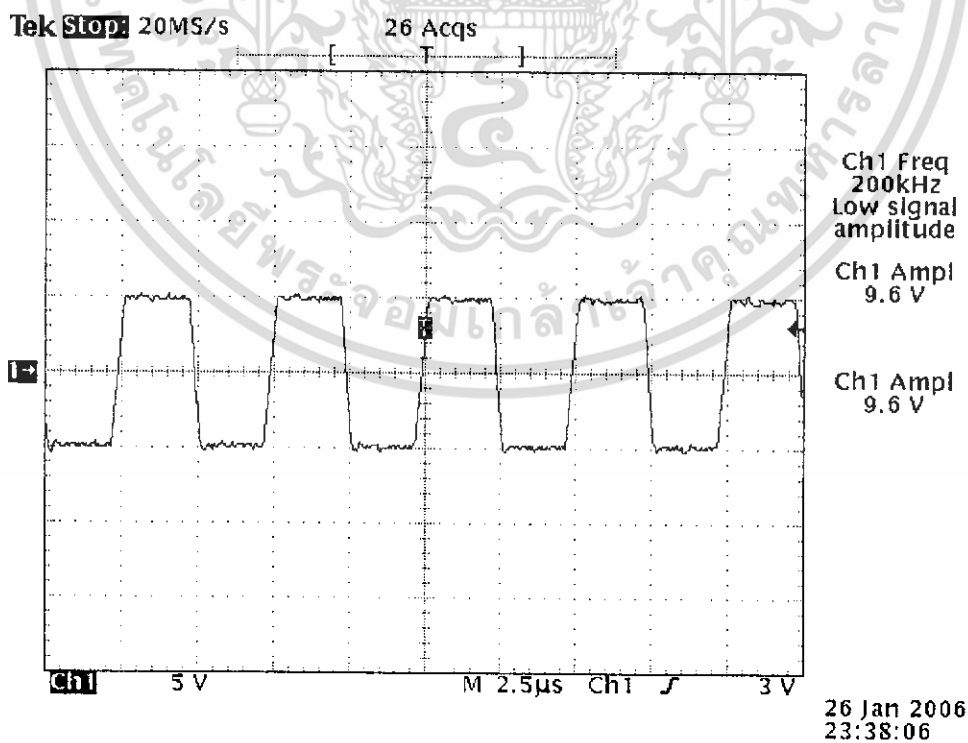


ง) รูปแสดงสัญญาณสี่เหลี่ยมที่ความถี่ 1 kHz

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

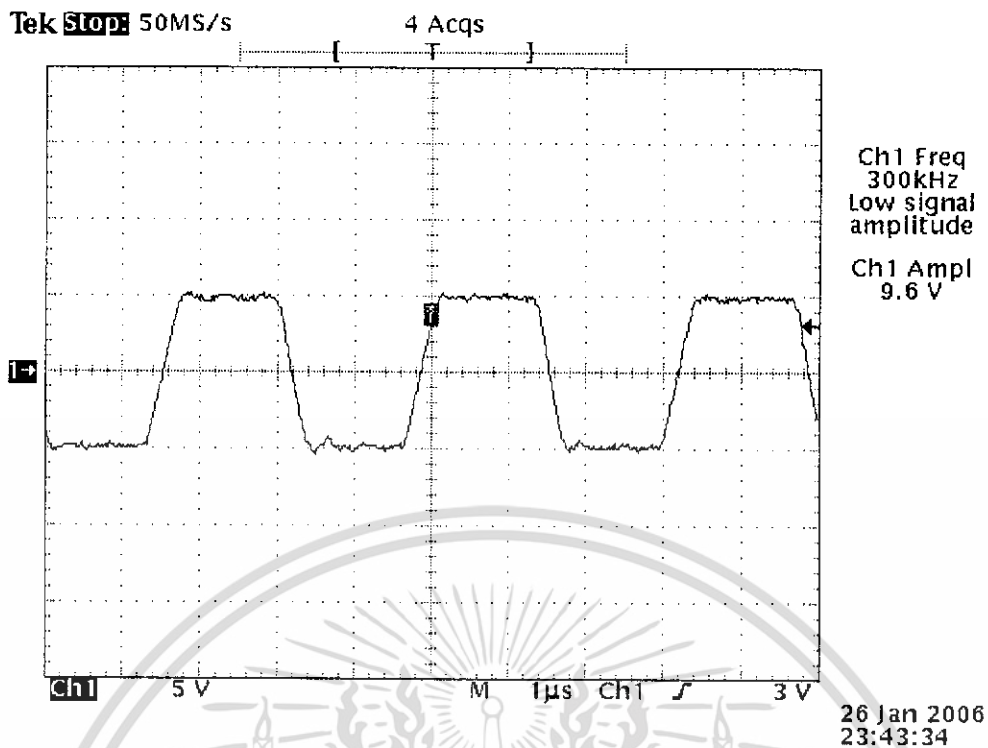


จ) รูปแสดงสัญญาณสี่เหลี่ยมที่ความถี่ 100 kHz

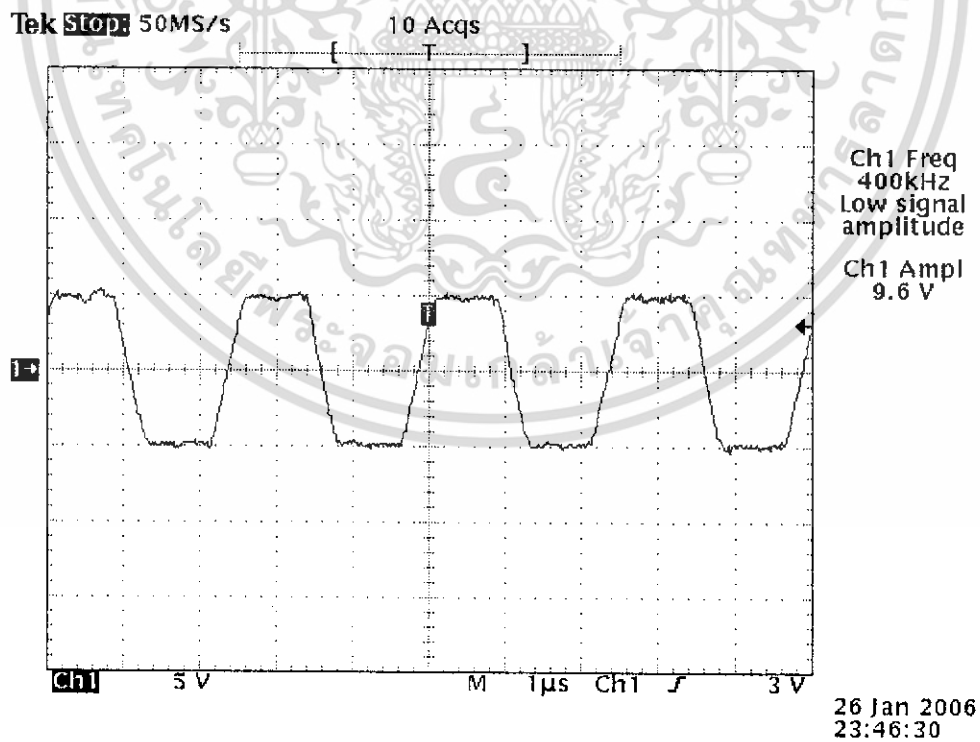


ค) รูปแสดงสัญญาณสี่เหลี่ยมที่ความถี่ 200 kHz

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

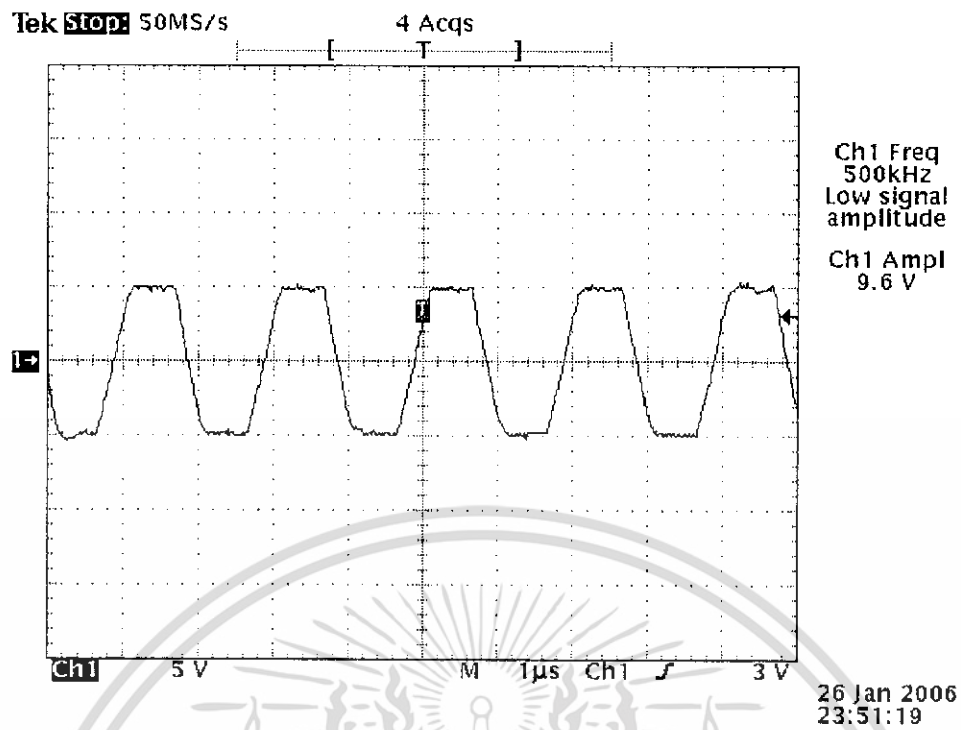


ข) รูปแสดงสัญญาณสี่เหลี่ยมที่ความถี่ 300 kHz

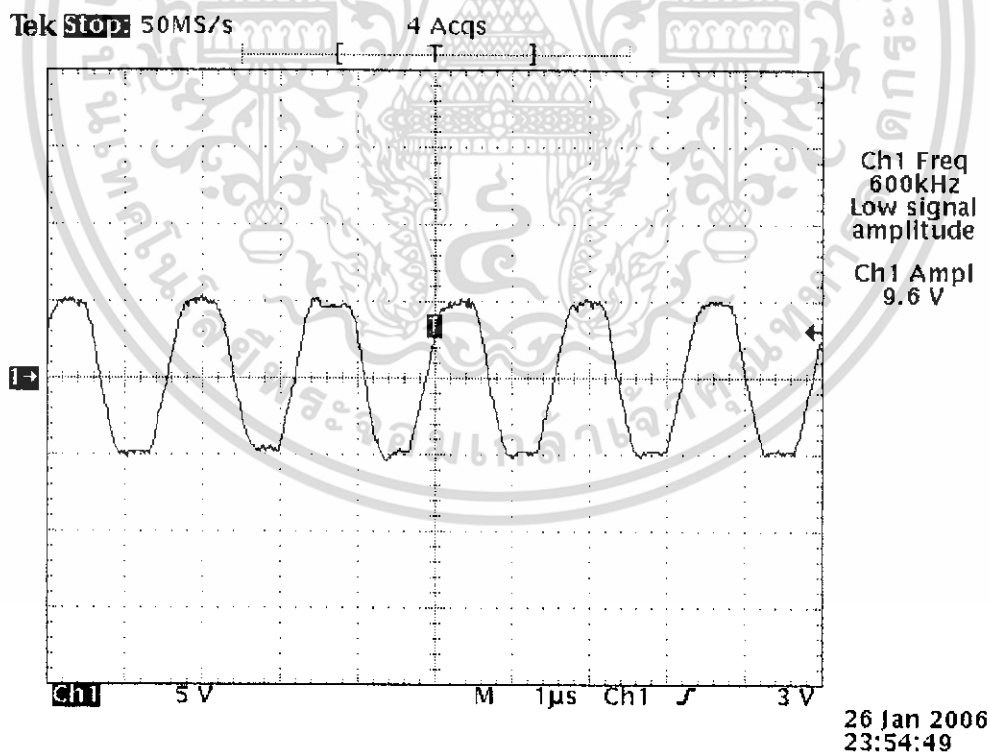


ข) รูปแสดงสัญญาณสี่เหลี่ยมที่ความถี่ 400 kHz

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาค้นคว้าเท่านั้น เมื่ออนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ณ) รูปแสดงสัญญาณสี่เหลี่ยมที่ความถี่ 500 kHz

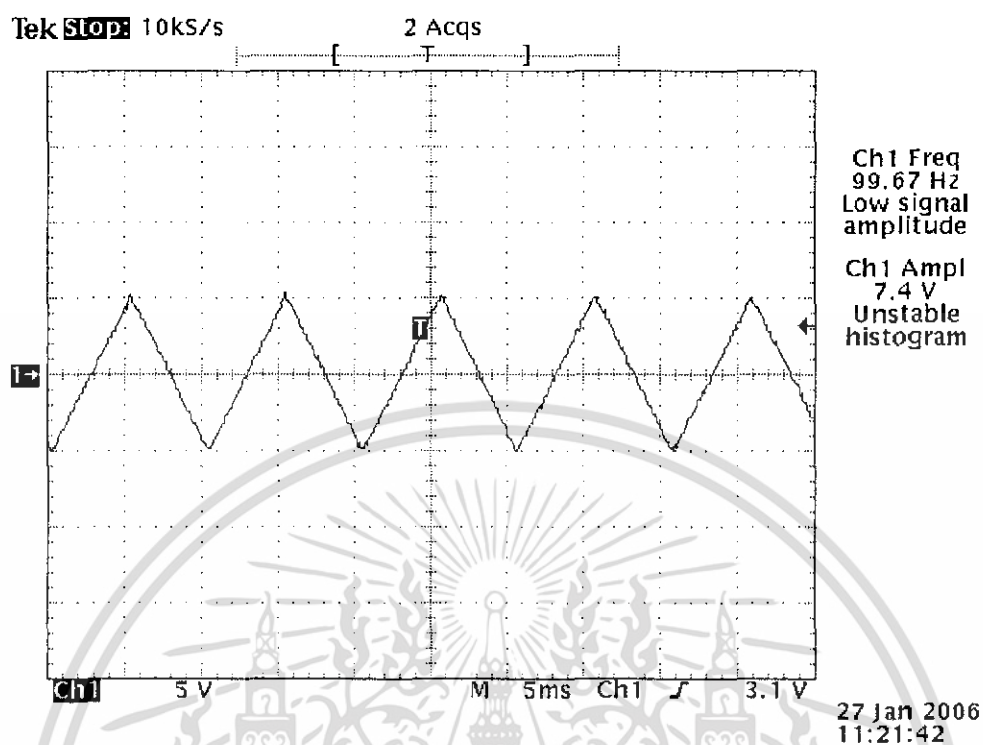


ญ) รูปแสดงสัญญาณสี่เหลี่ยมที่ความถี่ 600 kHz

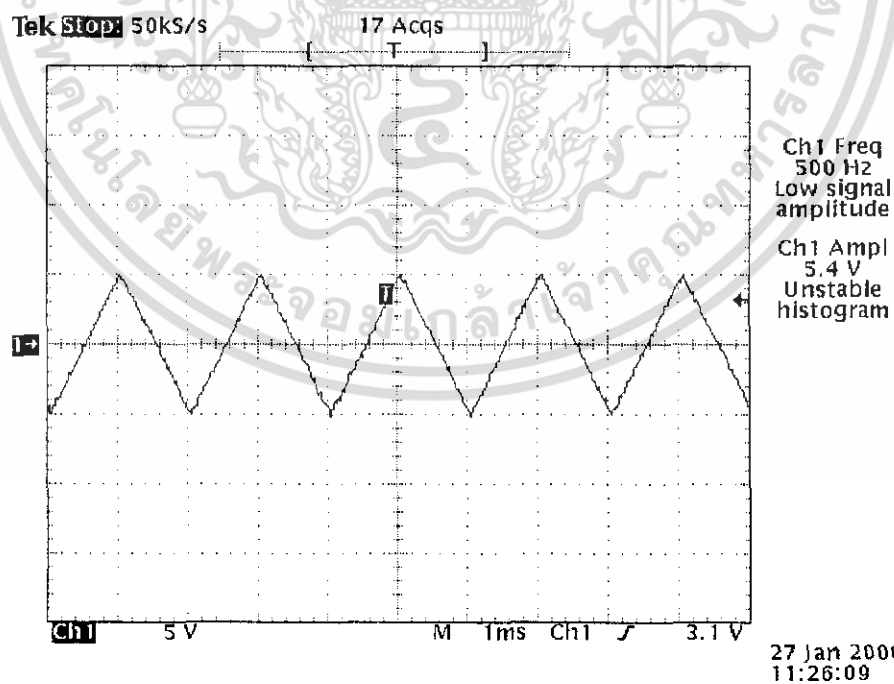
รูปที่ 4.45 การทดสอบการกำเนิดสัญญาณสี่เหลี่ยมที่ความถี่ต่างๆ

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์เพื่อการเรียนการสอนเท่านั้น เมื่อผู้ดูแลเห็นประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.5.2 การทดสอบการกำเนิดสัญญาณสามเหลี่ยมที่ความถี่ต่างๆ

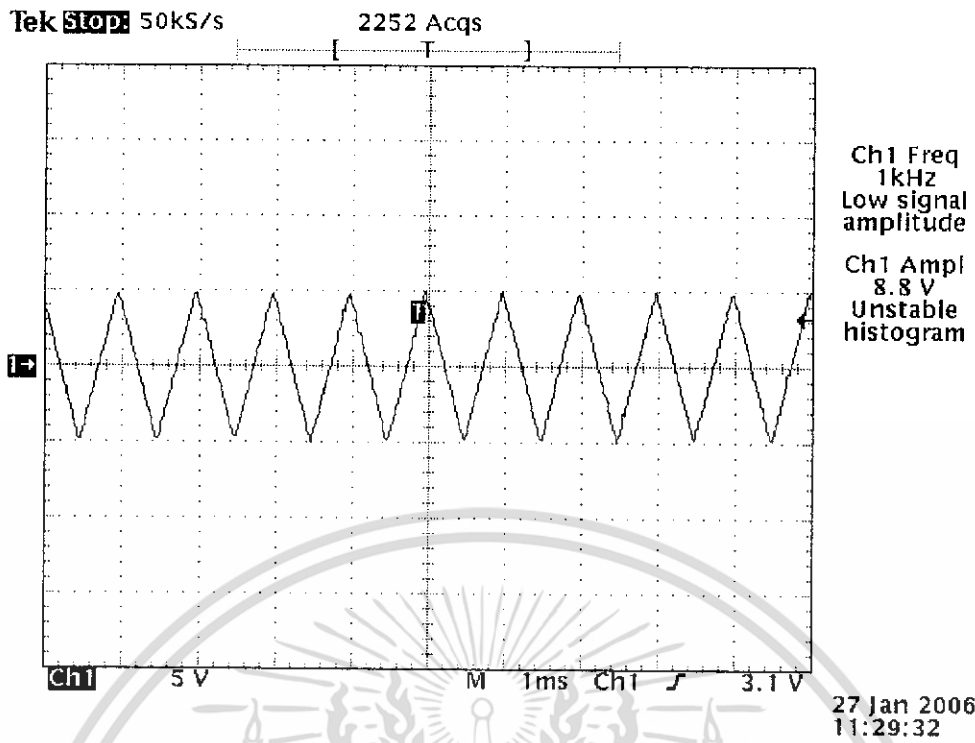


ก) รูปแสดงสัญญาณสามเหลี่ยมที่ความถี่ 100 Hz

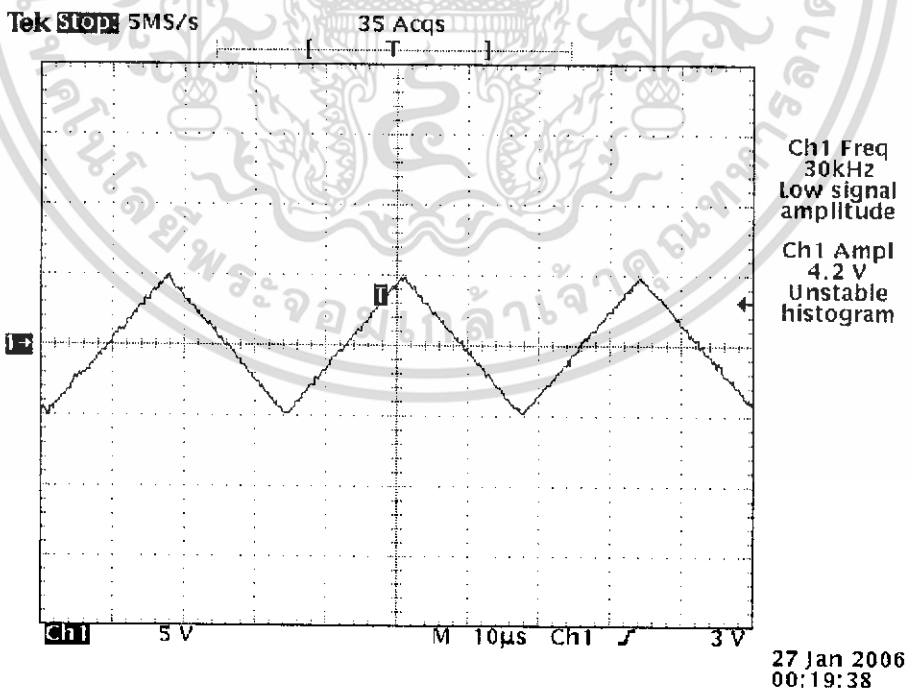


ข) รูปแสดงสัญญาณสามเหลี่ยมที่ความถี่ 500 Hz

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานภายในเท่านั้น เมื่อผู้ดูแลระบบเห็นว่าไม่เหมาะสมใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

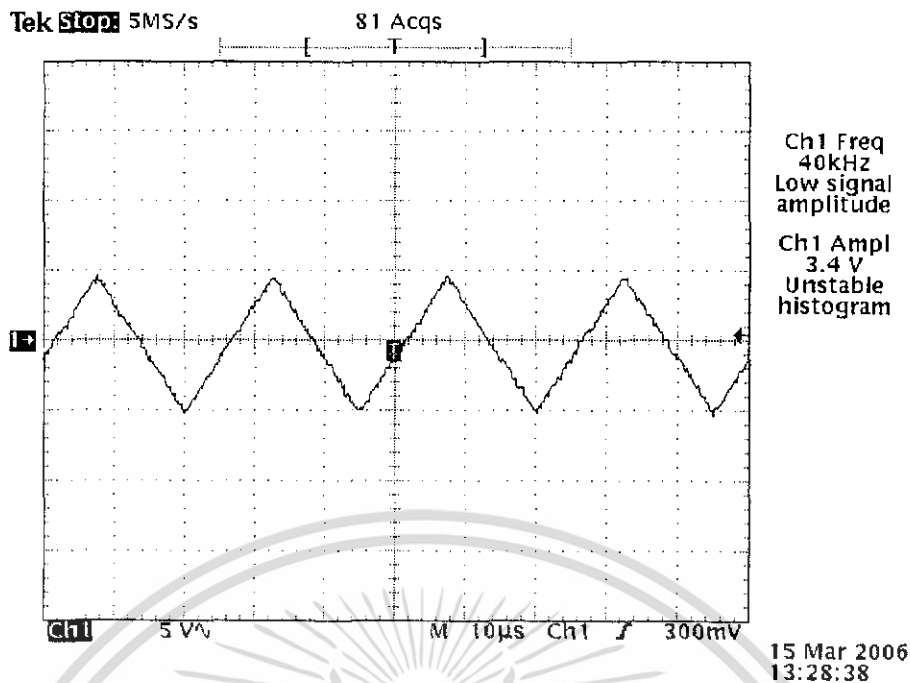


ก) รูปแสดงสัญญาณสามเหลี่ยมที่ความถี่ 1 kHz

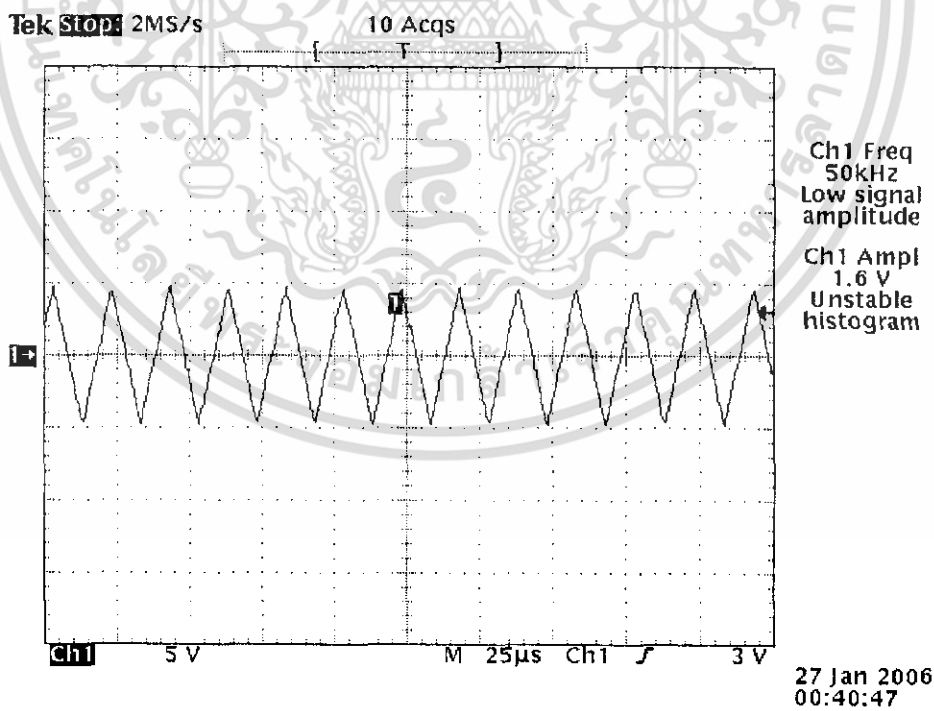


จ) รูปแสดงสัญญาณสามเหลี่ยมที่ความถี่ 30 kHz

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานภายในเท่านั้นไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

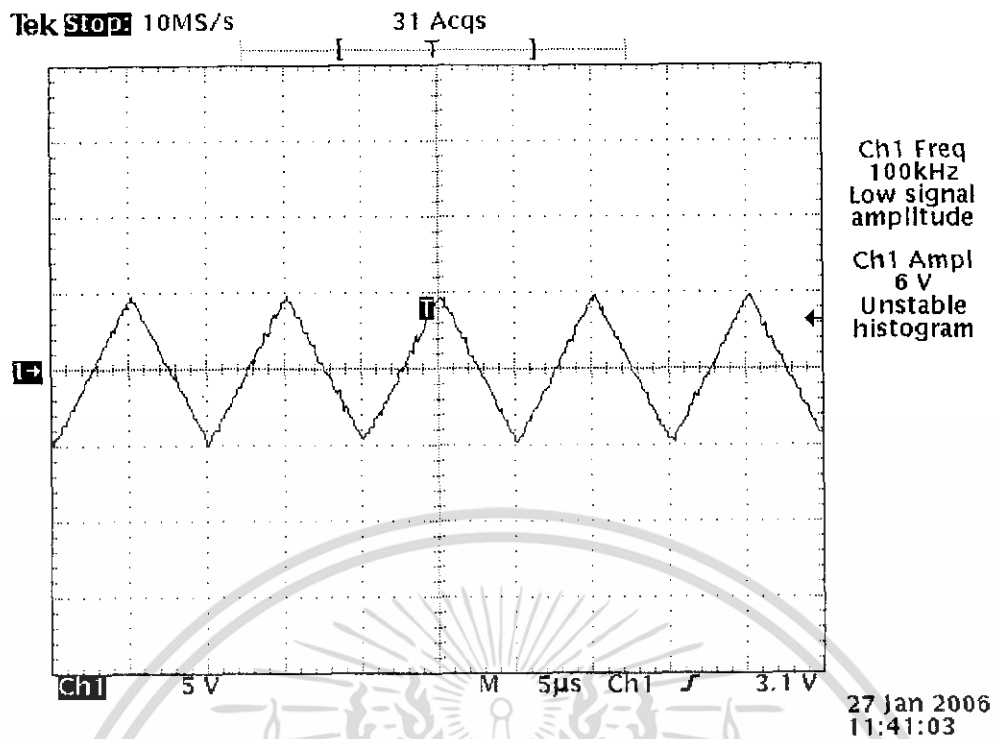


จ) รูปแสดงสัญญาณตามเหลี่ยมที่ความถี่ 40 kHz

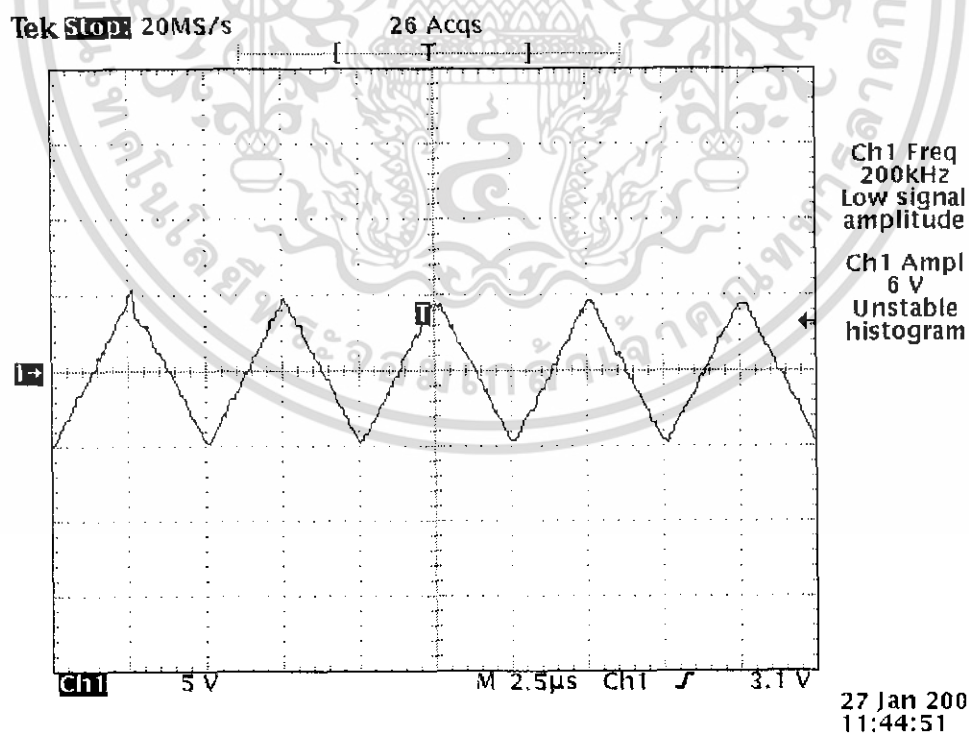


ค) รูปแสดงสัญญาณตามเหลี่ยมที่ความถี่ 50 kHz

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับใช้ภายในเท่านั้น เมื่อผู้ดูแลระบบเห็นหน้าไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

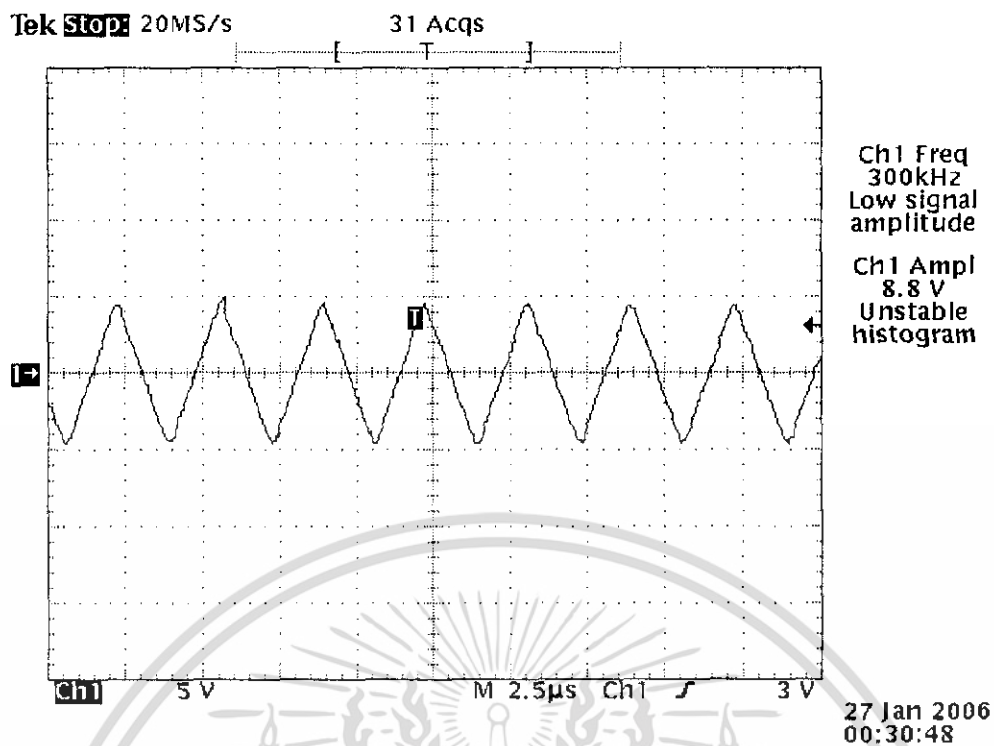


ข) รูปแสดงสัญญาณสามเหลี่ยมที่ความถี่ 100 kHz

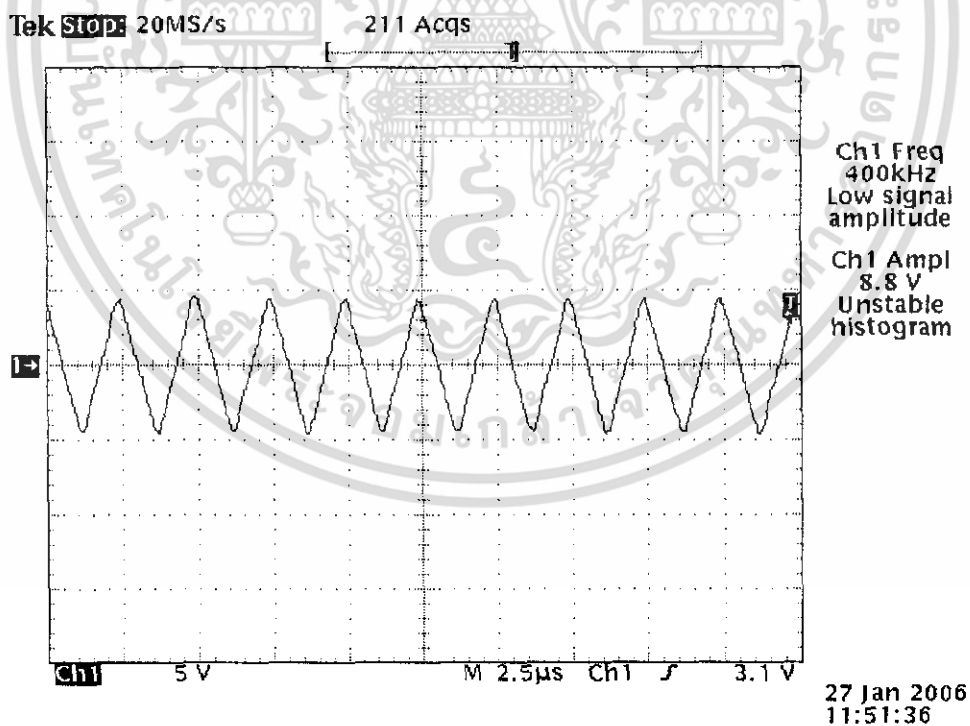


ข) รูปแสดงสัญญาณสามเหลี่ยมที่ความถี่ 200 kHz

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับใช้ภายในเท่านั้น ไม่ควรเผยแพร่ไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



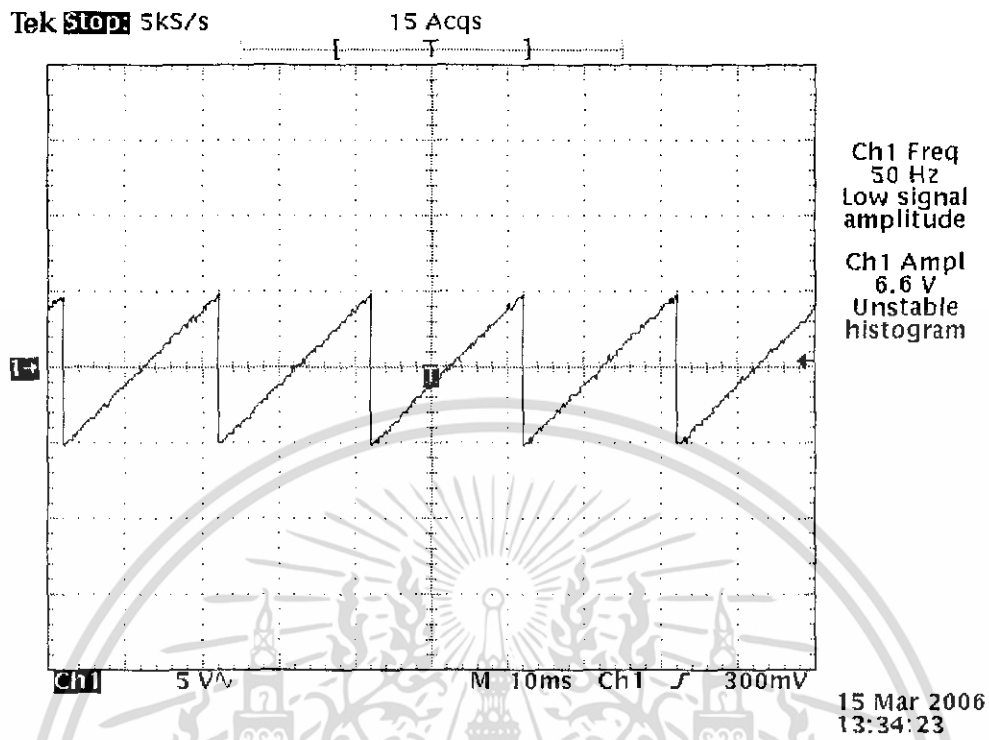
ณ) รูปแสดงสัญญาณสามเหลี่ยมที่ความถี่ 300 kHz



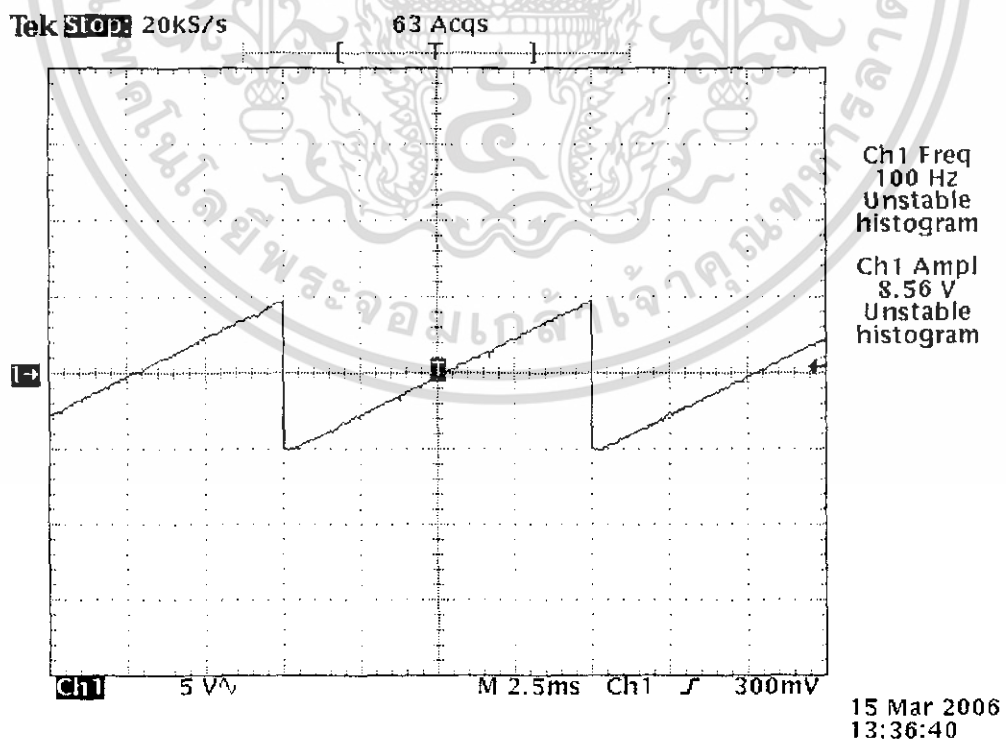
ญ) รูปแสดงสัญญาณสามเหลี่ยมที่ความถี่ 400 kHz

รูปที่ 4.46 การทดสอบการกำเนิดสัญญาณสามเหลี่ยมที่ความถี่ต่างๆ เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาดูเท่านั้น เมื่อผู้ญาติเห็นไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.5.3 การทดสอบการกำเนิดสัญญาณฟันเลื่อยที่ความถี่ต่างๆ

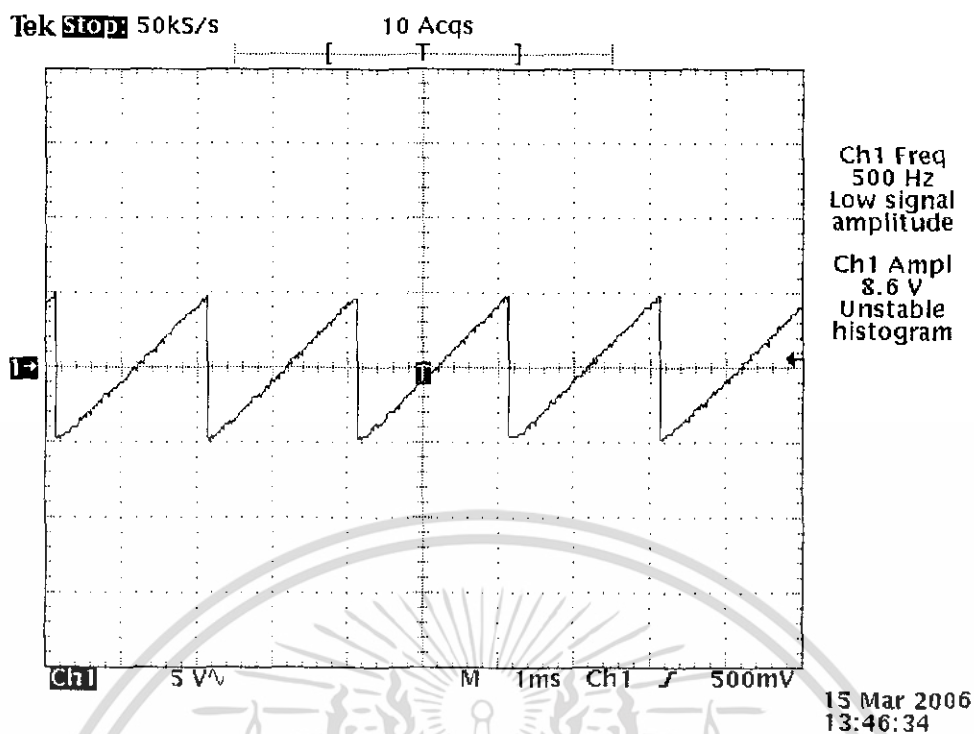


ก) รูปแสดงสัญญาณฟันเลื่อยที่ความถี่ 50 Hz

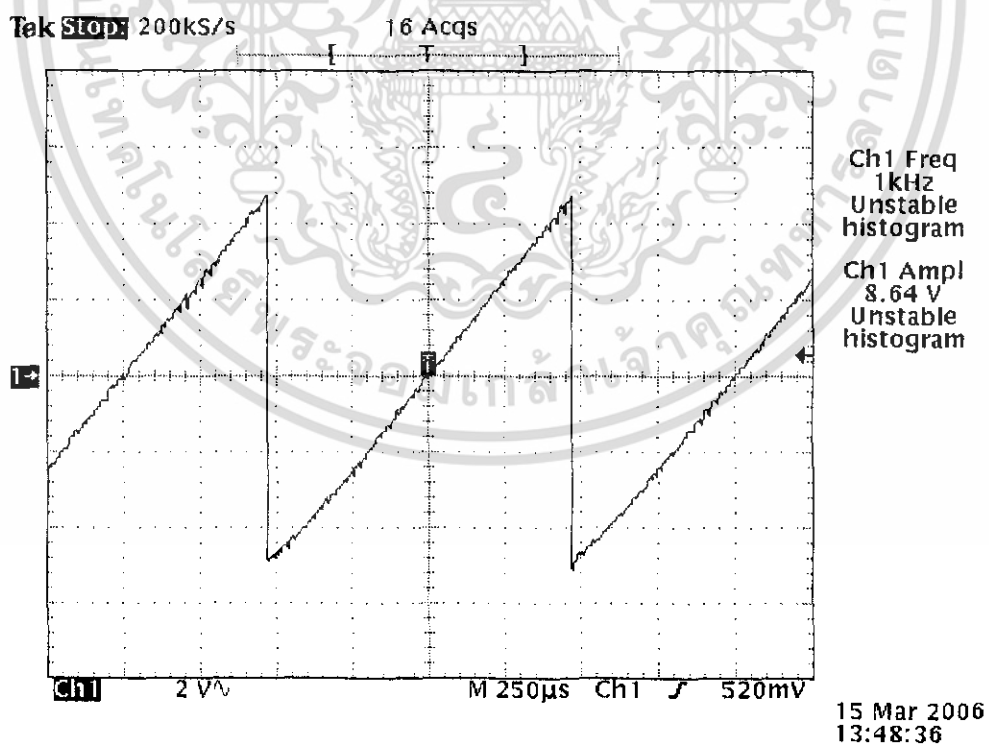


ข) รูปแสดงสัญญาณฟันเลื่อยที่ความถี่ 100 Hz

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับใช้เฉพาะในวิชาการศึกษาดูแลเท่านั้น เมื่อผู้ดูแลให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

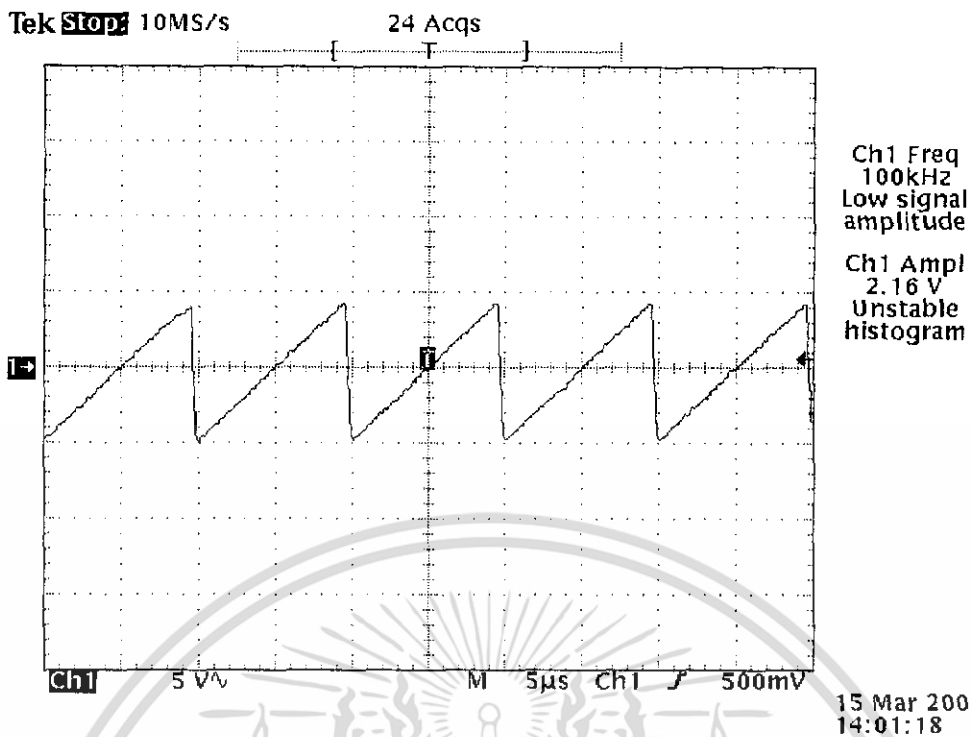


ก) รูปแสดงสัญญาณฟันเลื่อยที่ความถี่ 500 Hz

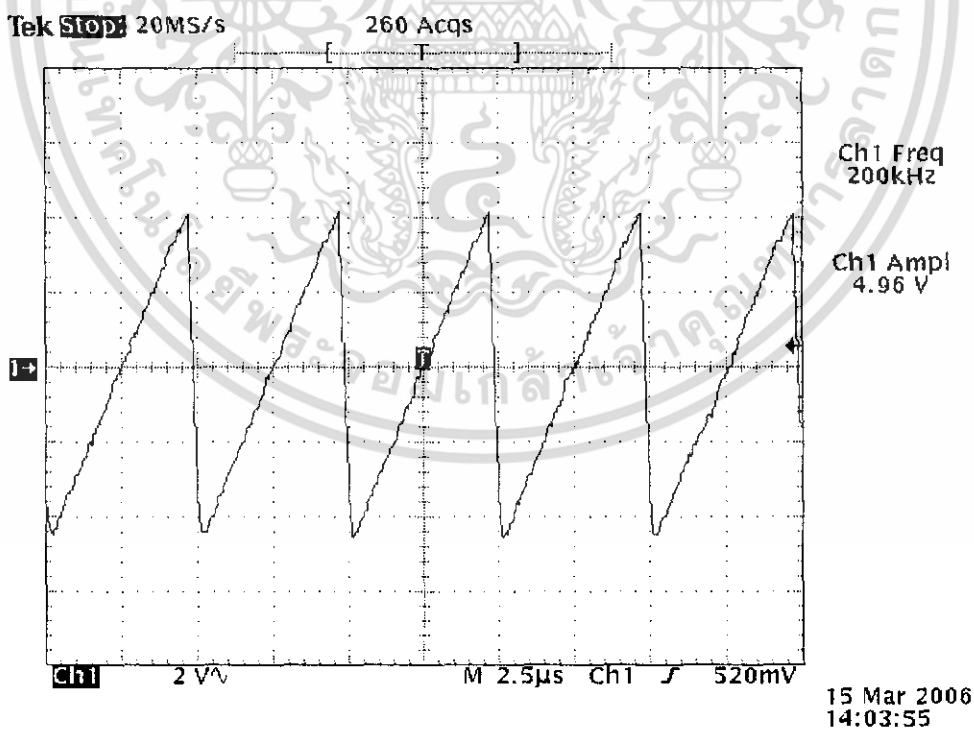


ข) รูปแสดงสัญญาณฟันเลื่อยที่ความถี่ 1 kHz

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับใช้เฉพาะในโครงการวิจัยเท่านั้น เมื่อผู้ผู้ใดให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



จ) รูปแสดงสัญญาณฟันเลื่อยที่ความถี่ 100 kHz

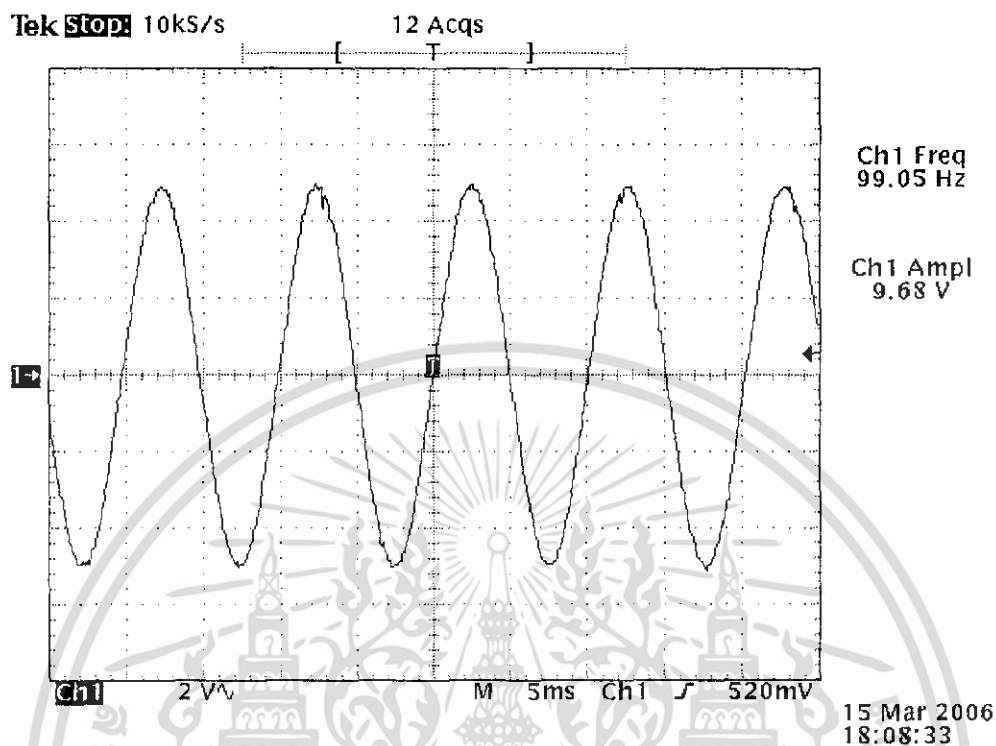


ข) รูปแสดงสัญญาณฟันเลื่อยที่ความถี่ 200 kHz

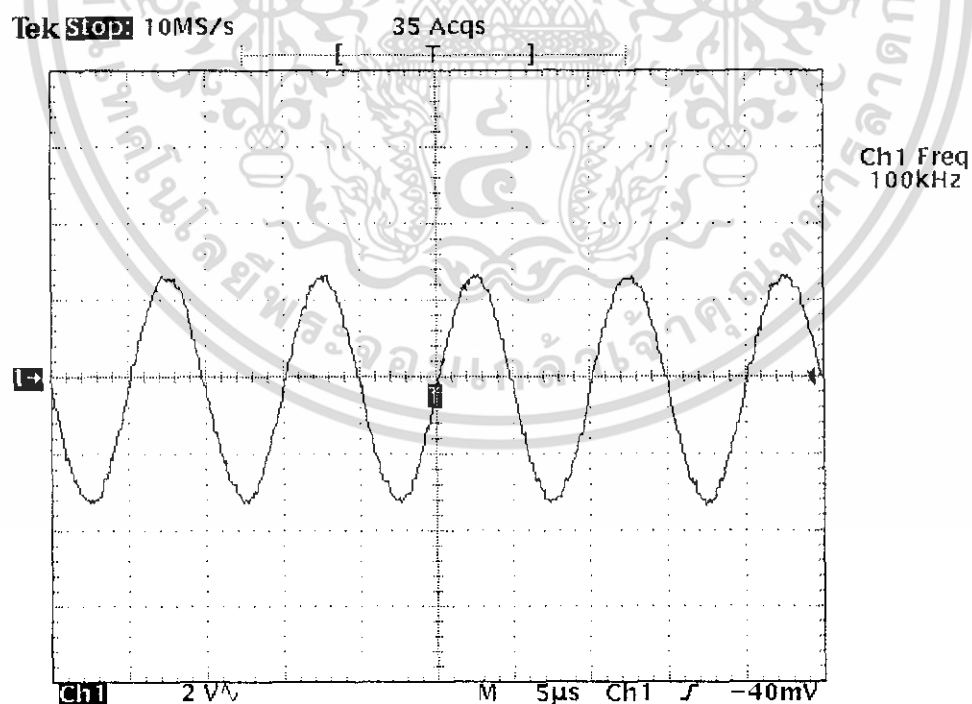
รูปที่ 4.47 การทดสอบการกำเนิดสัญญาณฟันเลื่อยที่ความถี่ต่างๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.5.4 การทดสอบการกำเนิดสัญญาณชานันต์ที่ได้จากตารางเปิดดูเปรียบเทียบกับสัญญาณชานันต์ที่ได้จากสมการโพลีโนเมียลที่ความถี่ต่างๆ

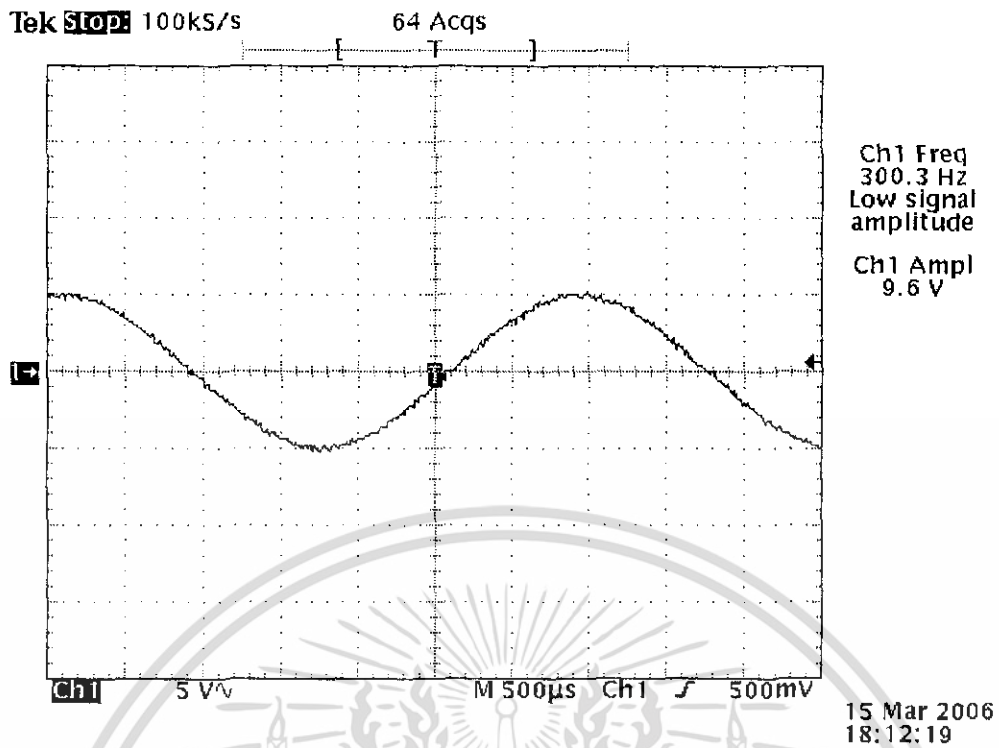


ก) รูปแสดงสัญญาณชานันต์ที่ความถี่ 100 Hz จากตารางเปิดดู

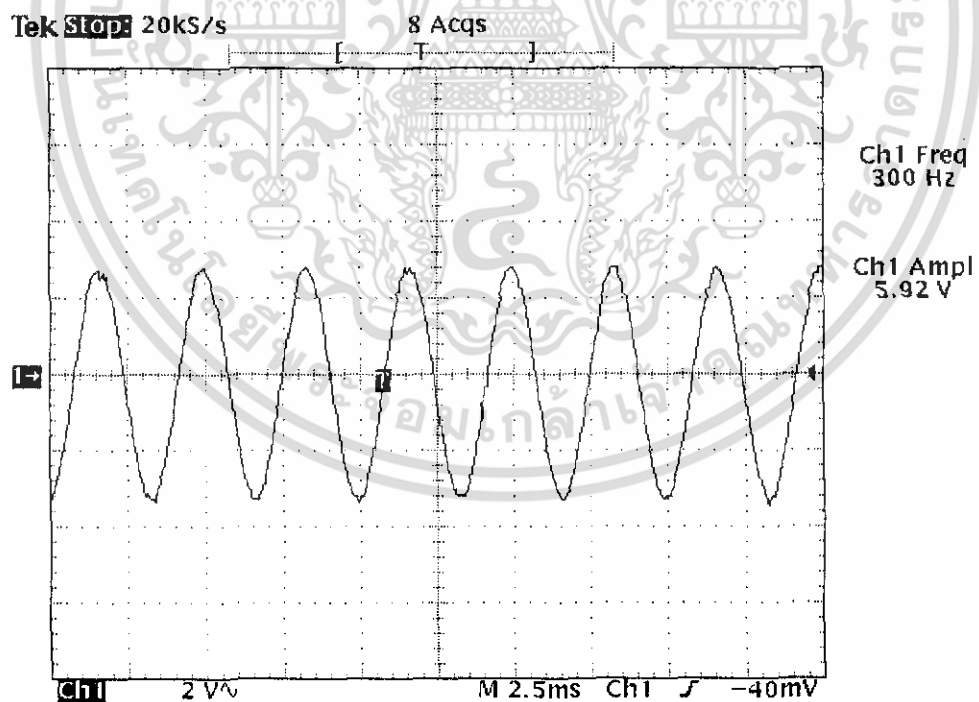


ข) รูปแสดงสัญญาณชานันต์ที่ความถี่ 100 Hz จากโพลีโนเมียล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

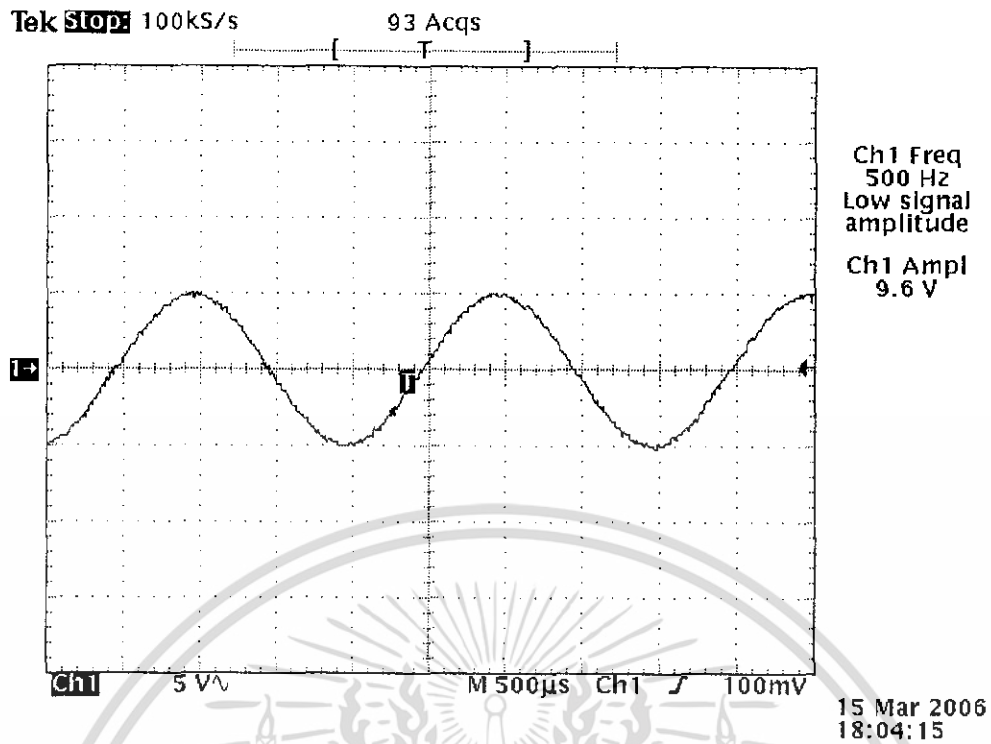


ก) รูปแสดงสัญญาณชานันท์ความถี่ 300 Hz จากตารางเปิดดู

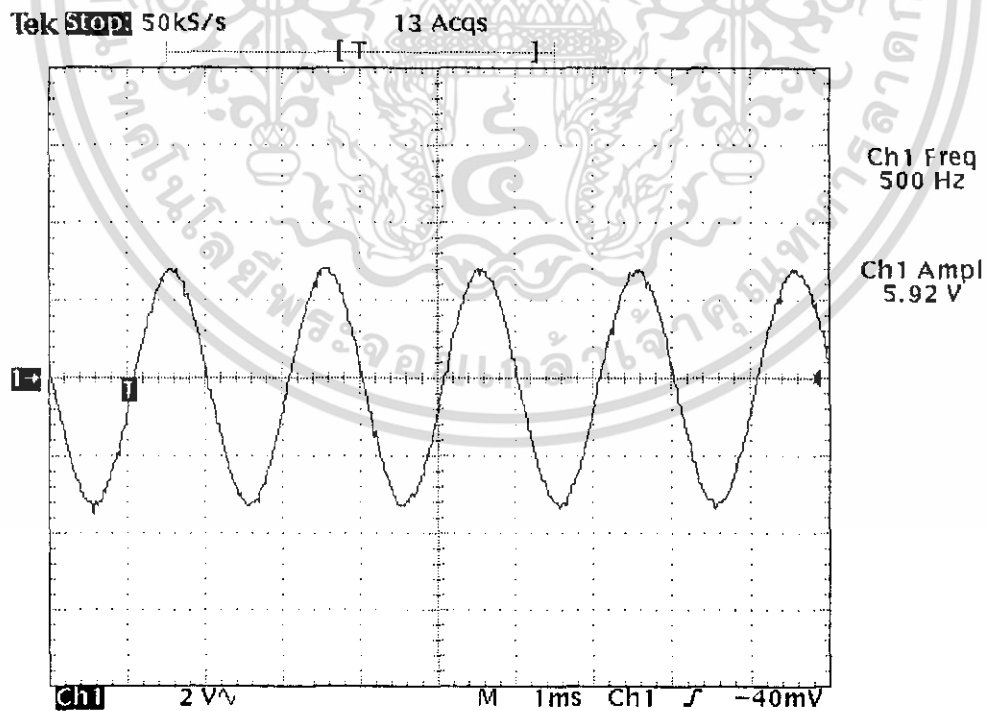


ง) รูปแสดงสัญญาณชานันท์ความถี่ 300 Hz จากโพลีโนเมียล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

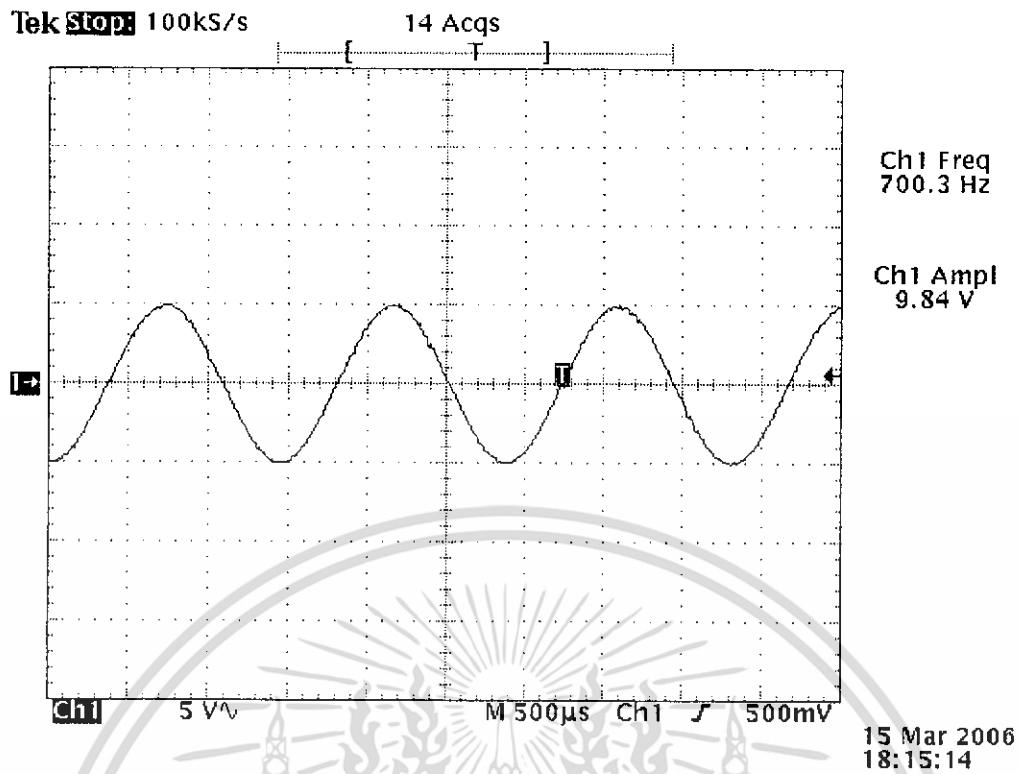


จ) รูปแสดงสัญญาณขาอินพุตที่ความถี่ 500 Hz จากตารางเปิดดู

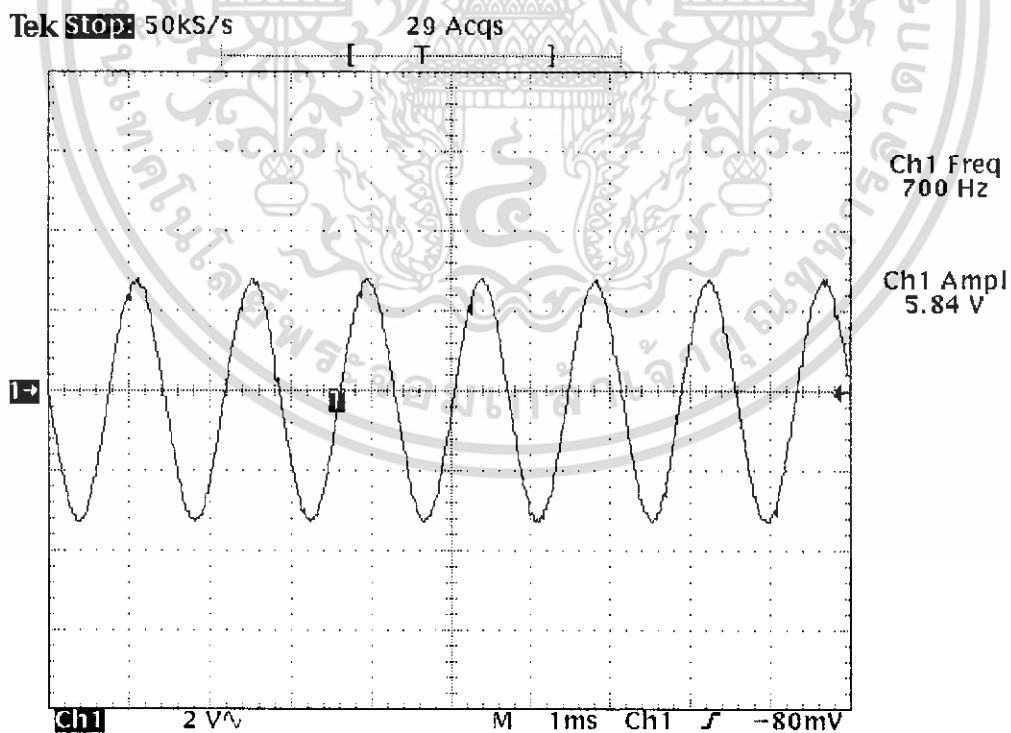


ค) รูปแสดงสัญญาณขาอินพุตที่ความถี่ 500 Hz จากโฟลิดิโนเมียด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

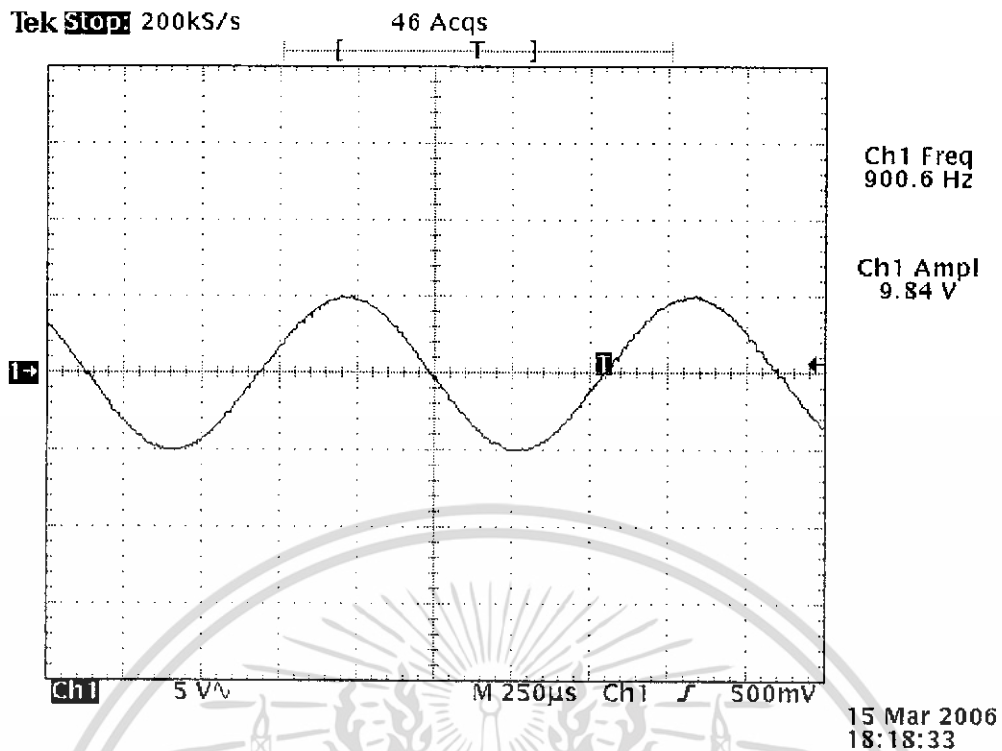


ข) รูปแสดงสัญญาณขาอินพุตที่ความถี่ 700 Hz จากตารางเปิดดู

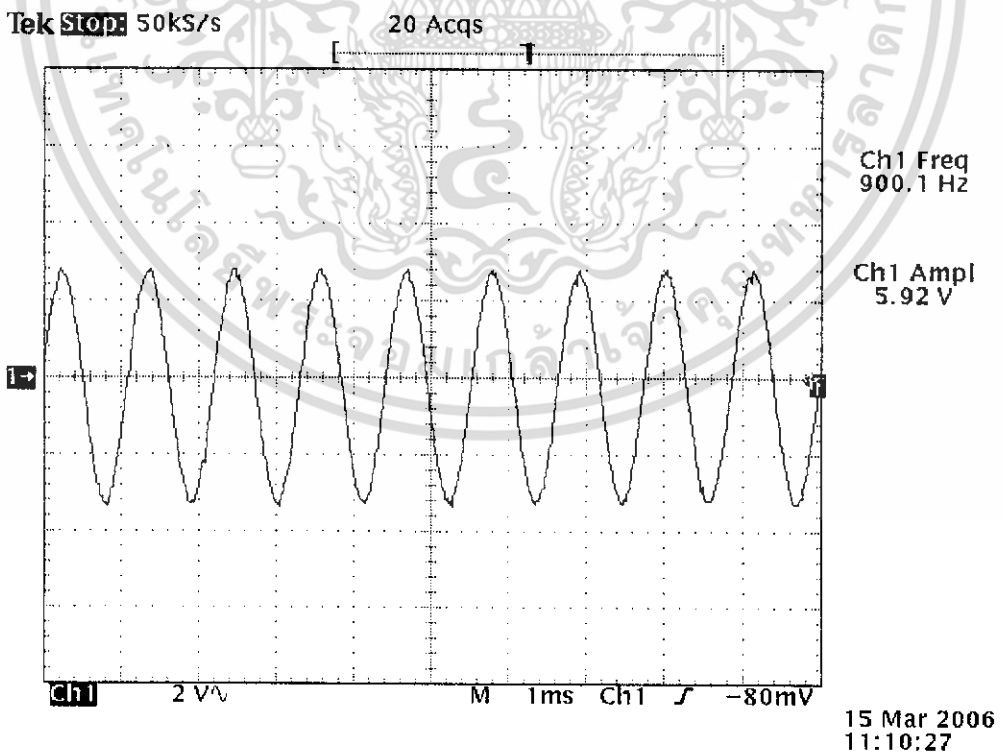


ข) รูปแสดงสัญญาณขาอินพุตที่ความถี่ 700 Hz จากโพลีโนเมียล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

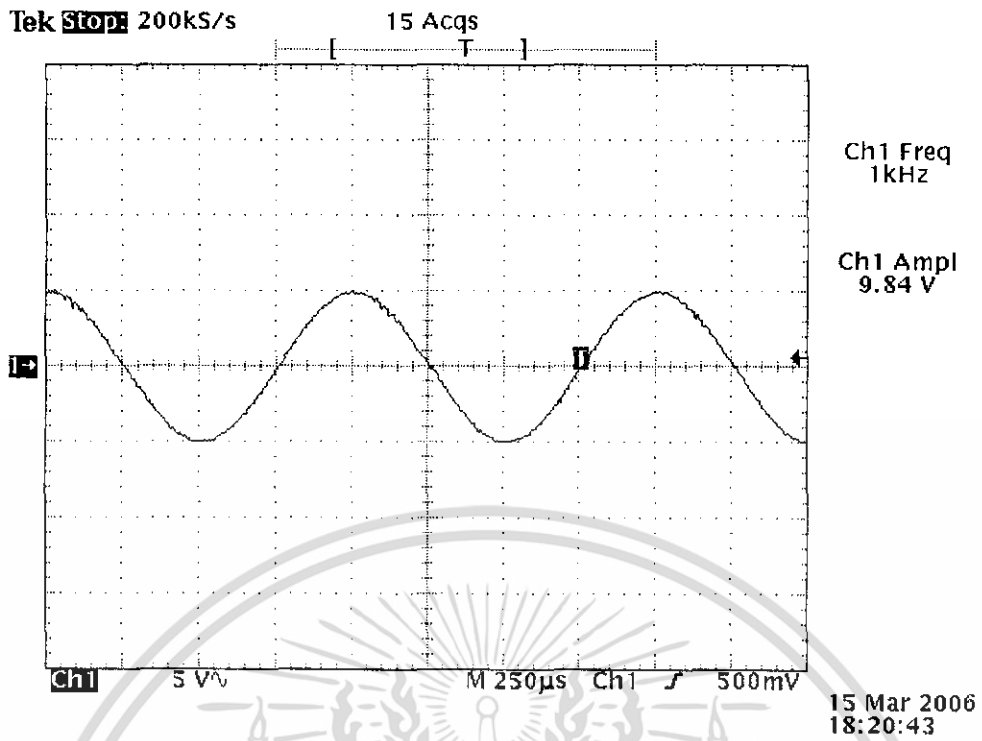


ณ) รูปแสดงสัญญาณชายนที่ความถี่ 900 Hz จากตารางเปิดดู

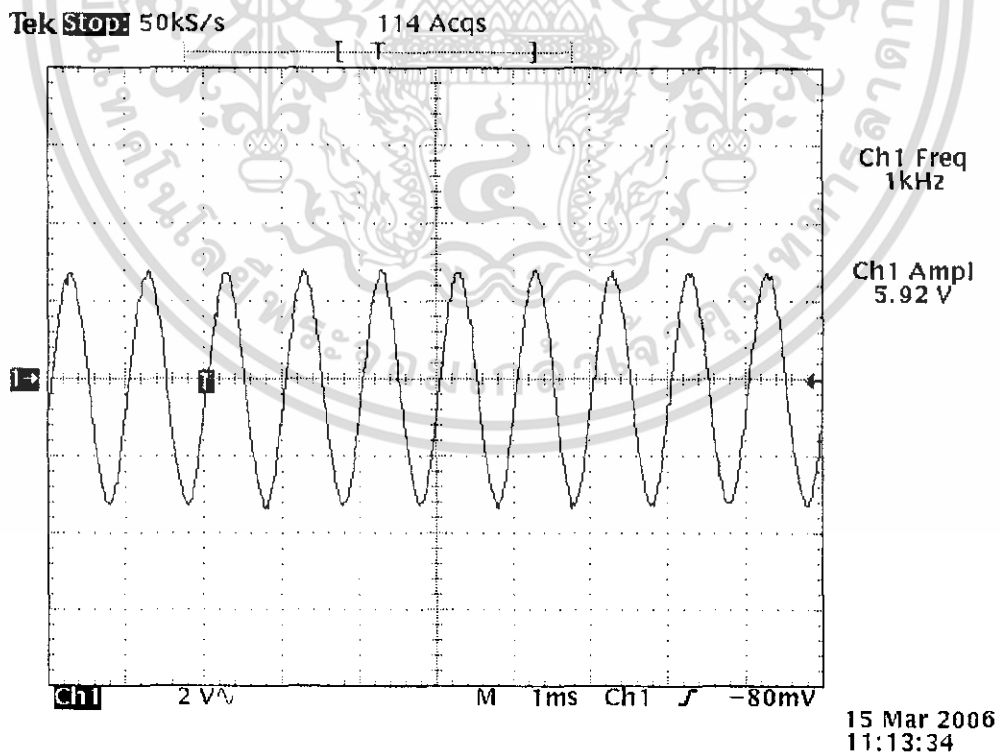


ญ) รูปแสดงสัญญาณชายนที่ความถี่ 900 Hz จากโพลีโนเมียล

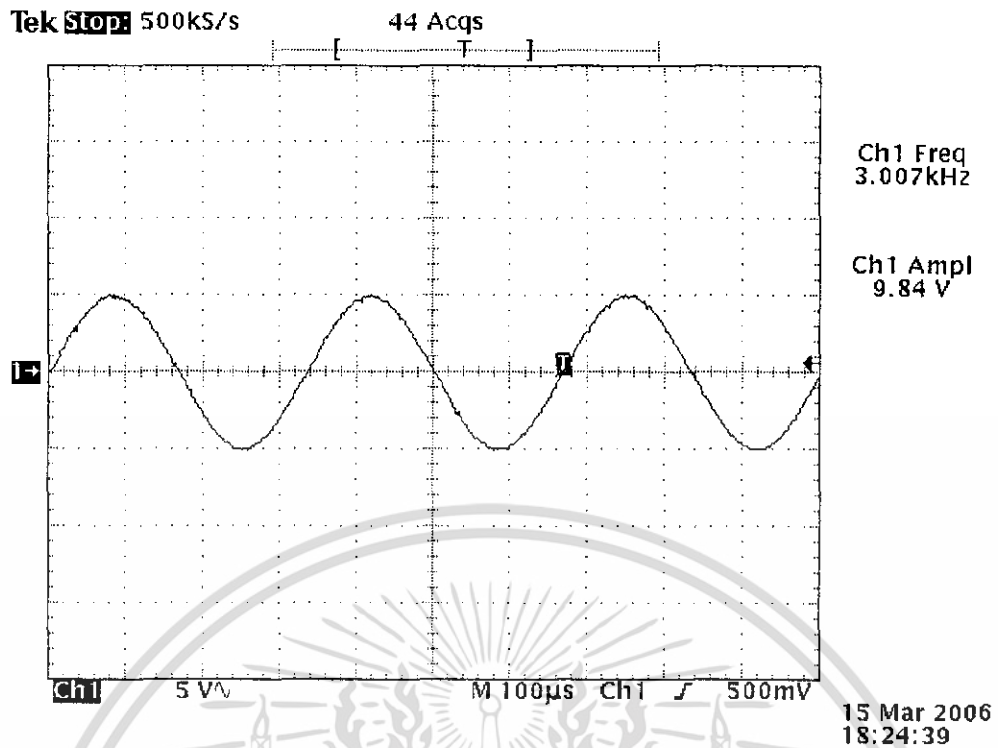
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



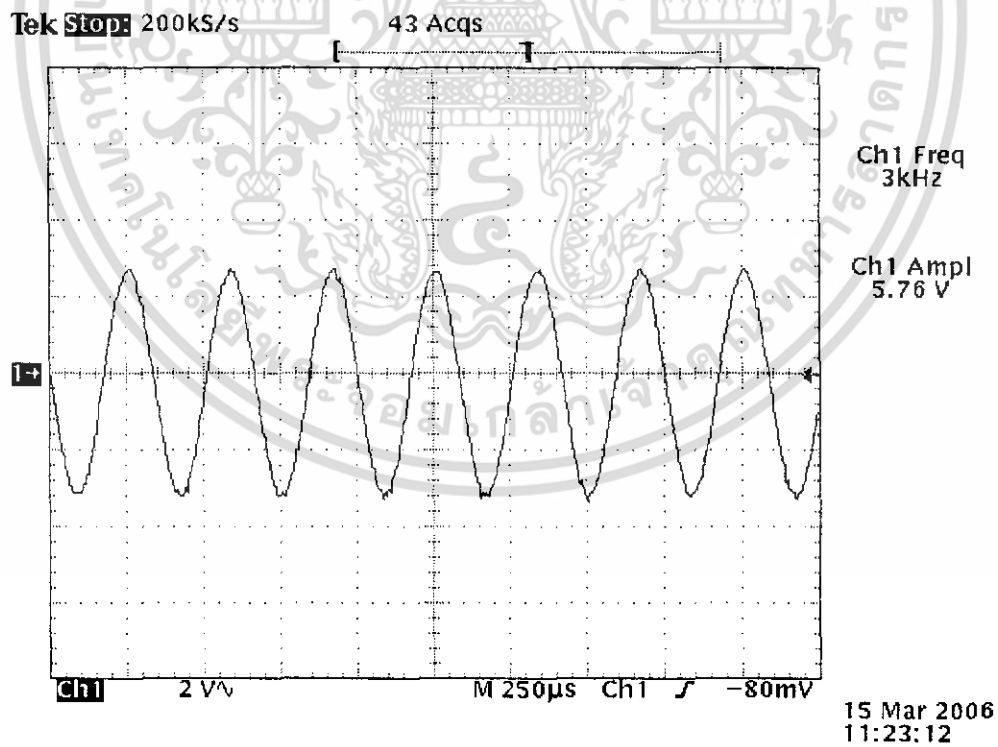
ฎ) รูปแสดงสัญญาณขาอินพุตที่ความถี่ 1 kHz จากตารางเปิดดู



เอกสารนี้เป็นเอกสารที่สงวนไว้รูปแสดงสัญญาณขาอินพุตที่ความถี่ 1 kHz จากโฟลเดอร์ที่ใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

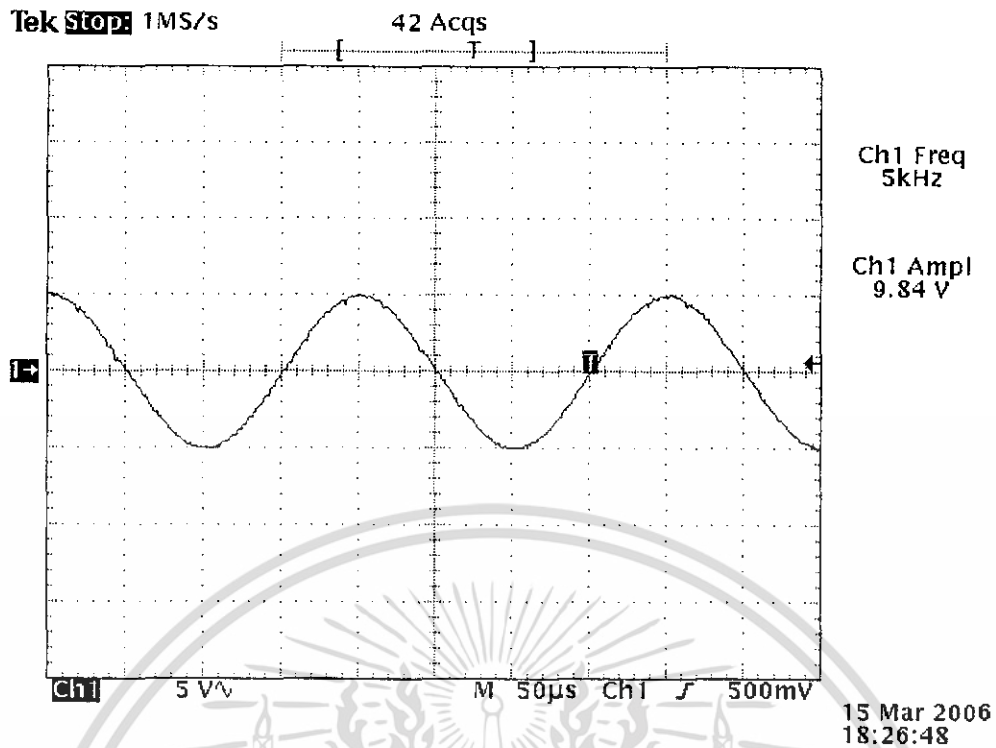


จ) รูปแสดงสัญญาณชานน์ที่ความถี่ 3 kHz จากตารางเปิดดู

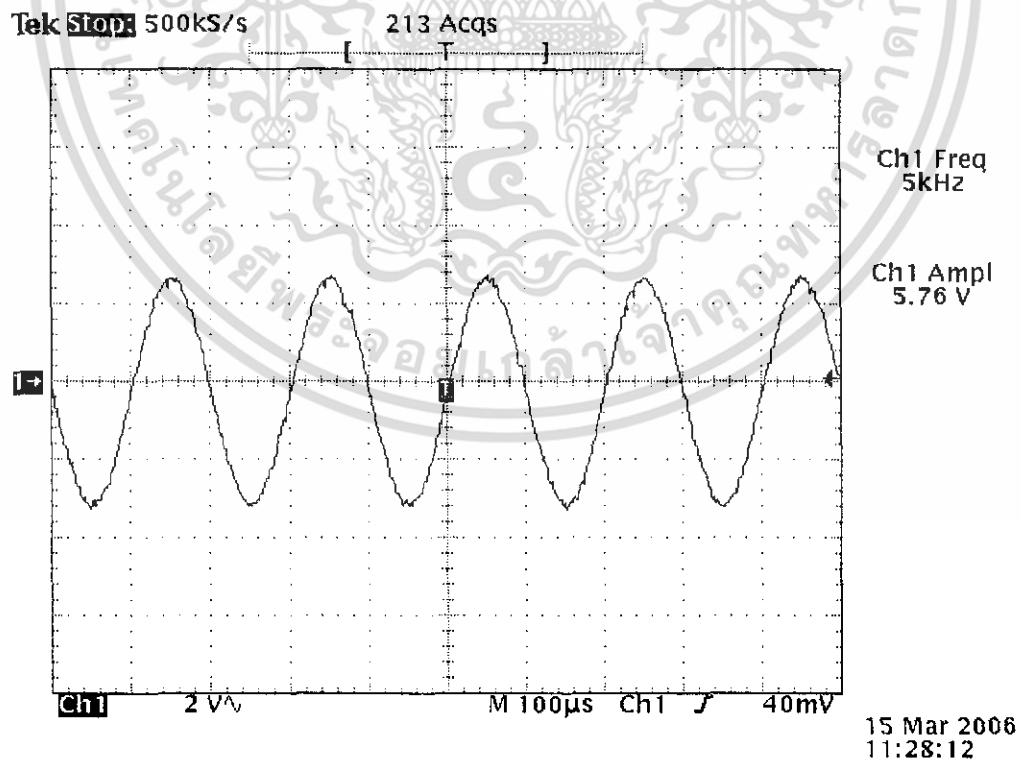


ท) รูปแสดงสัญญาณชานน์ที่ความถี่ 3 kHz จากโพลีโนเมียล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

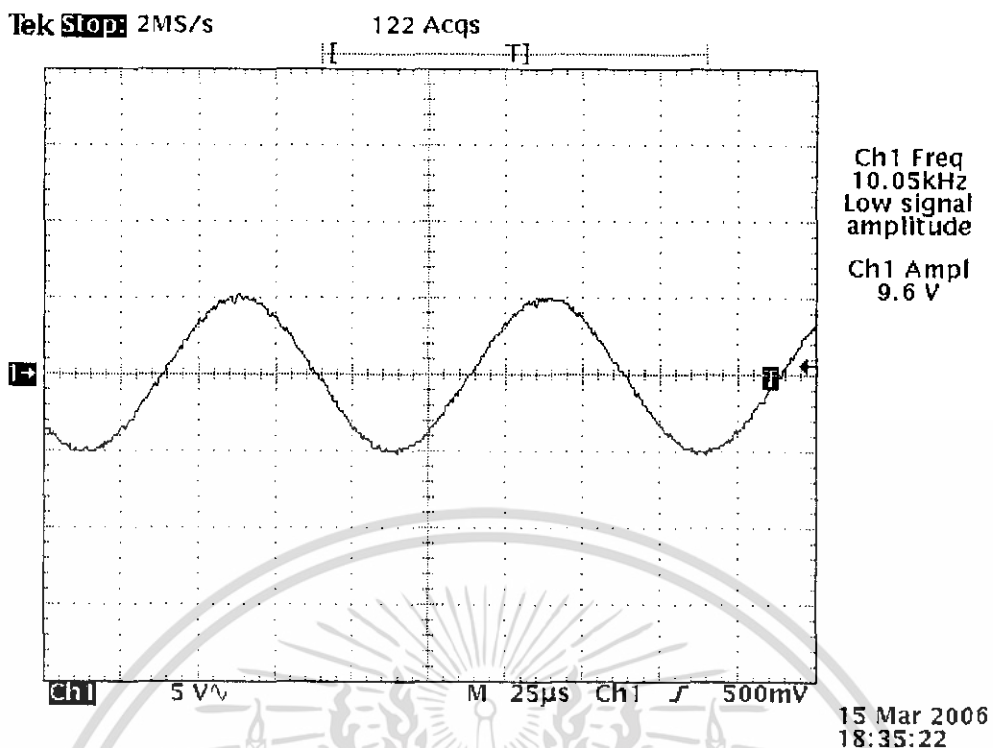


ต) รูปแสดงสัญญาณขาอินพุตที่ความถี่ 5 kHz จากตารางเปิดดู

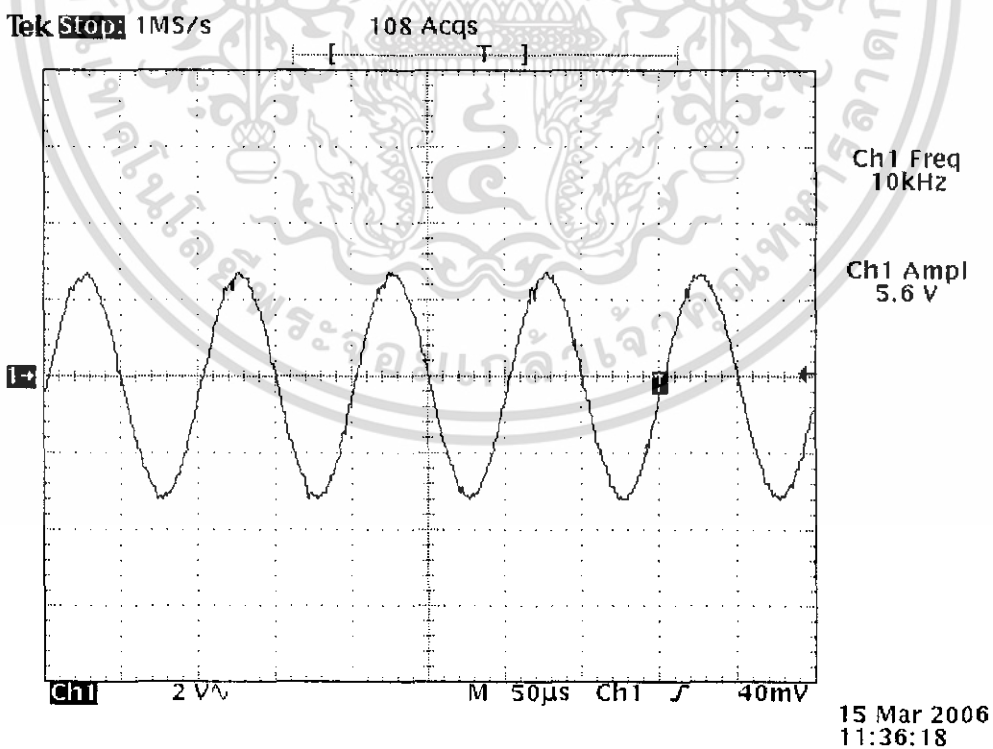


ณ) รูปแสดงสัญญาณขาอินพุตที่ความถี่ 5 kHz จากโพลีโนเมียล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

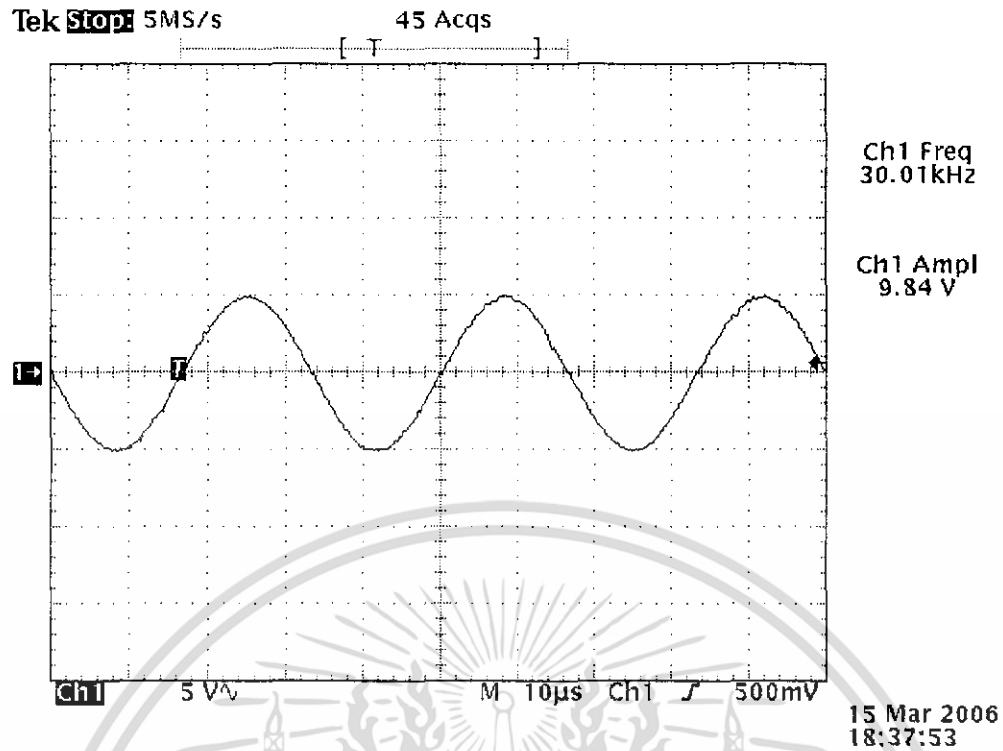


ด) รูปแสดงสัญญาณขาอินพุตที่ความถี่ 10 kHz จากตารางเปิดดู

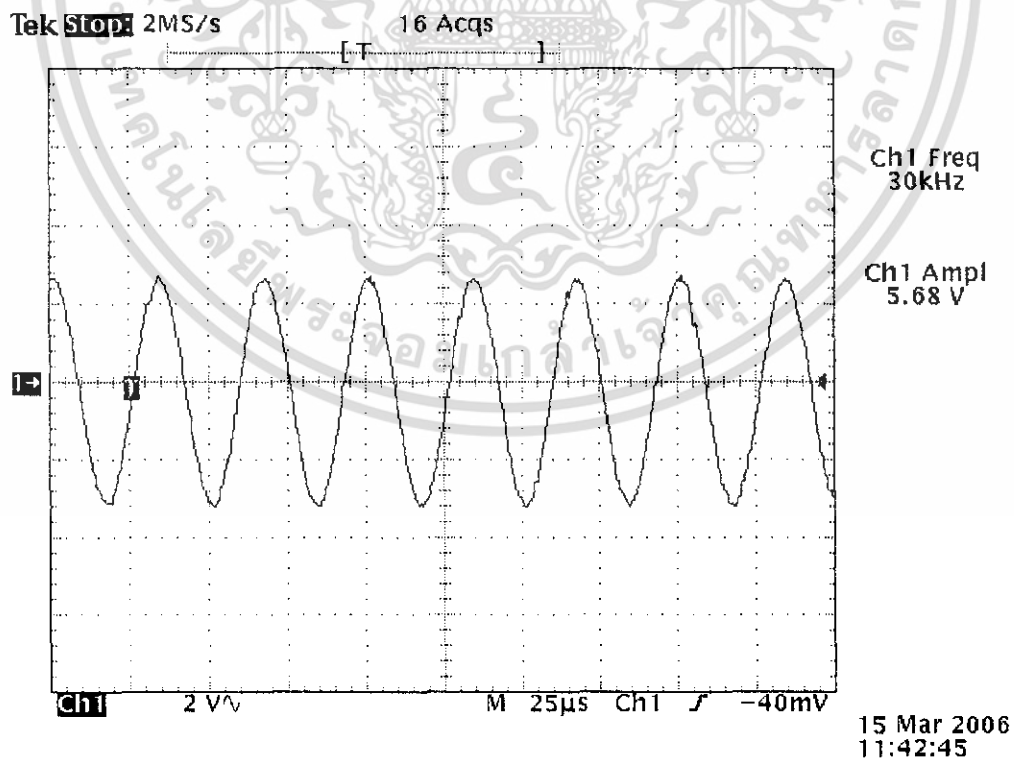


ด) รูปแสดงสัญญาณขาอินพุตที่ความถี่ 10 kHz จากโพลีโนเมียล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

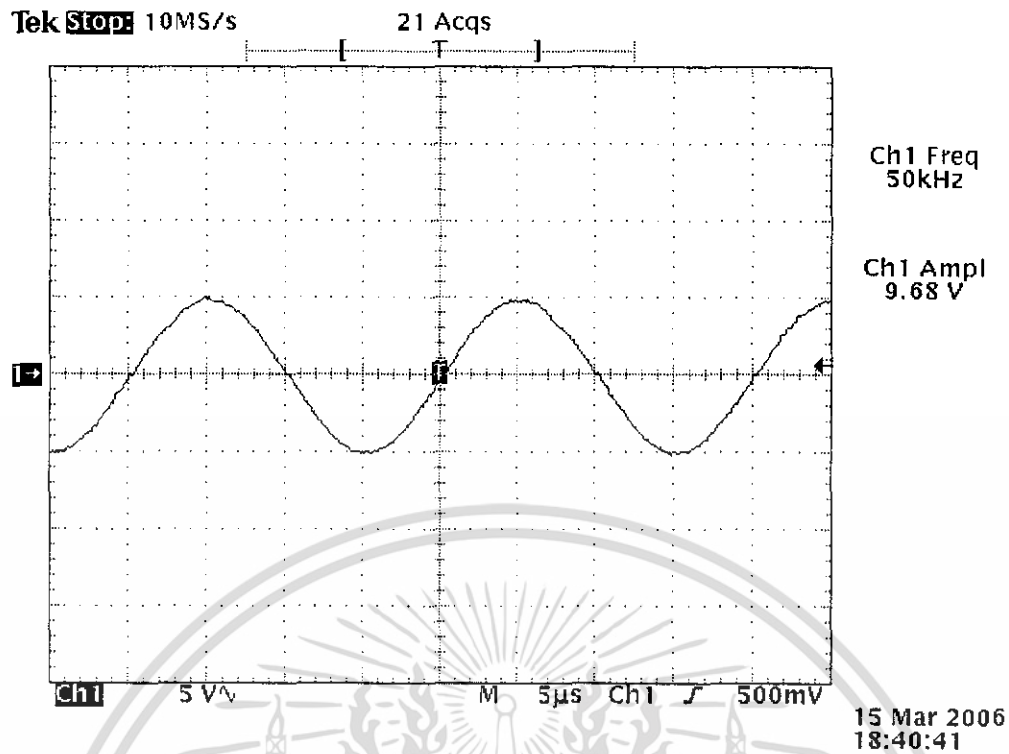


ต) รูปแสดงสัญญาณไซน์ที่มีความถี่ 30 kHz จากตารางเปิดดู

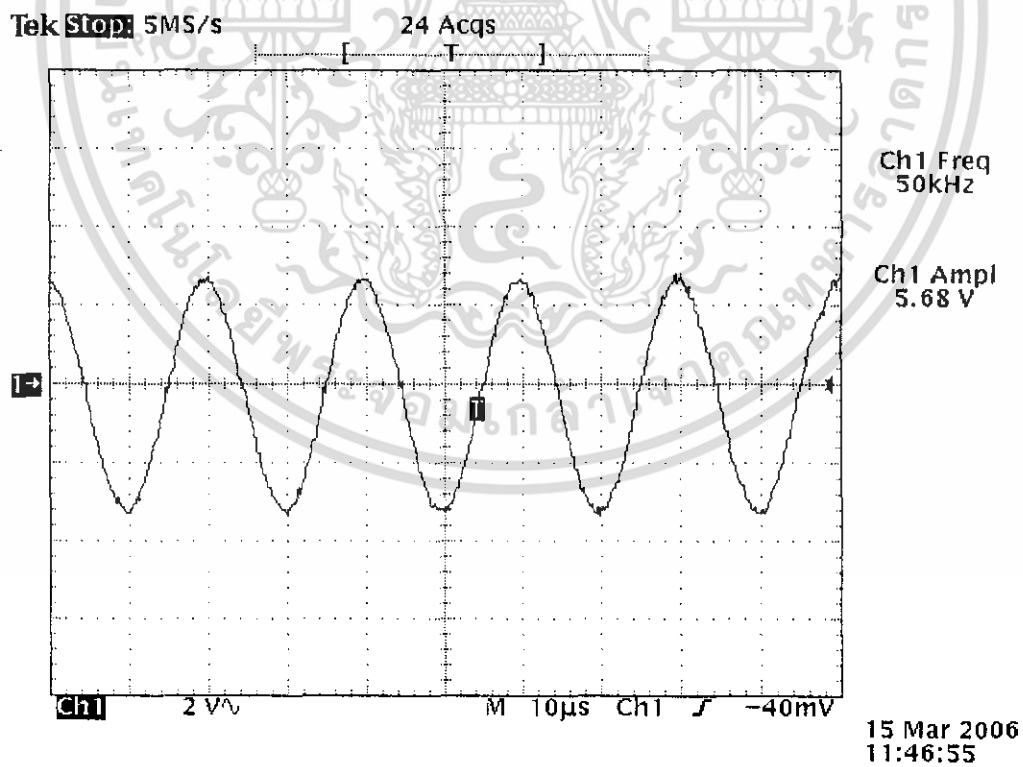


ท) รูปแสดงสัญญาณไซน์ที่มีความถี่ 30 kHz จากโพลีโนเมียล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่นิยมนำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

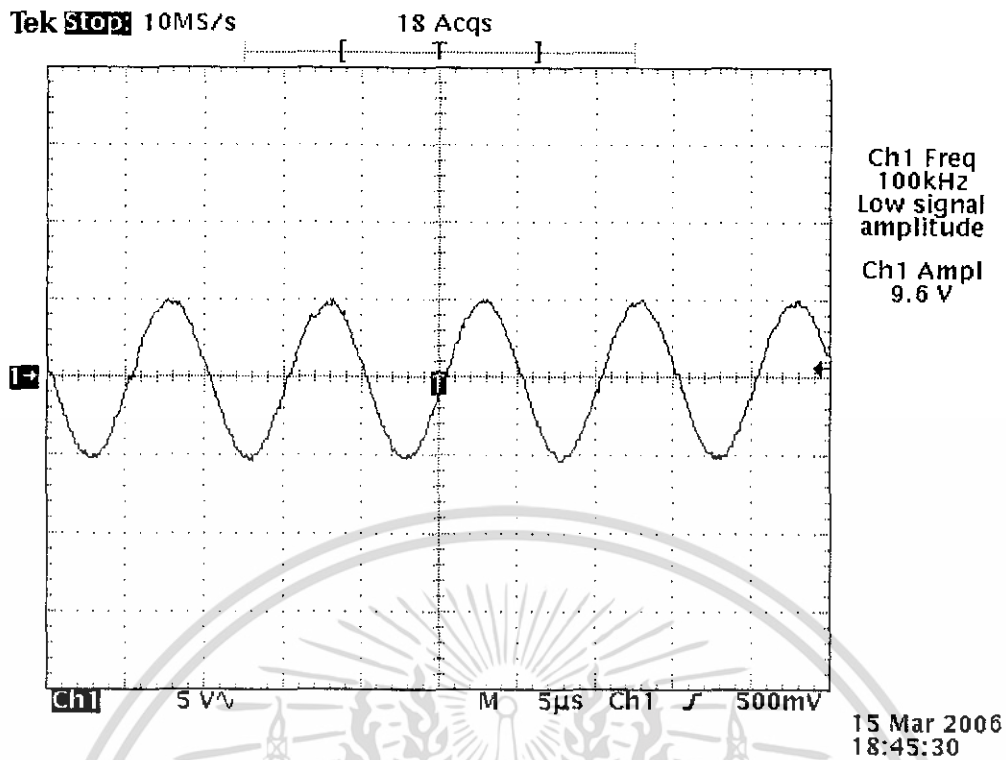


ข) รูปแสดงสัญญาณความถี่ 50 kHz จากตารางเปิดดู

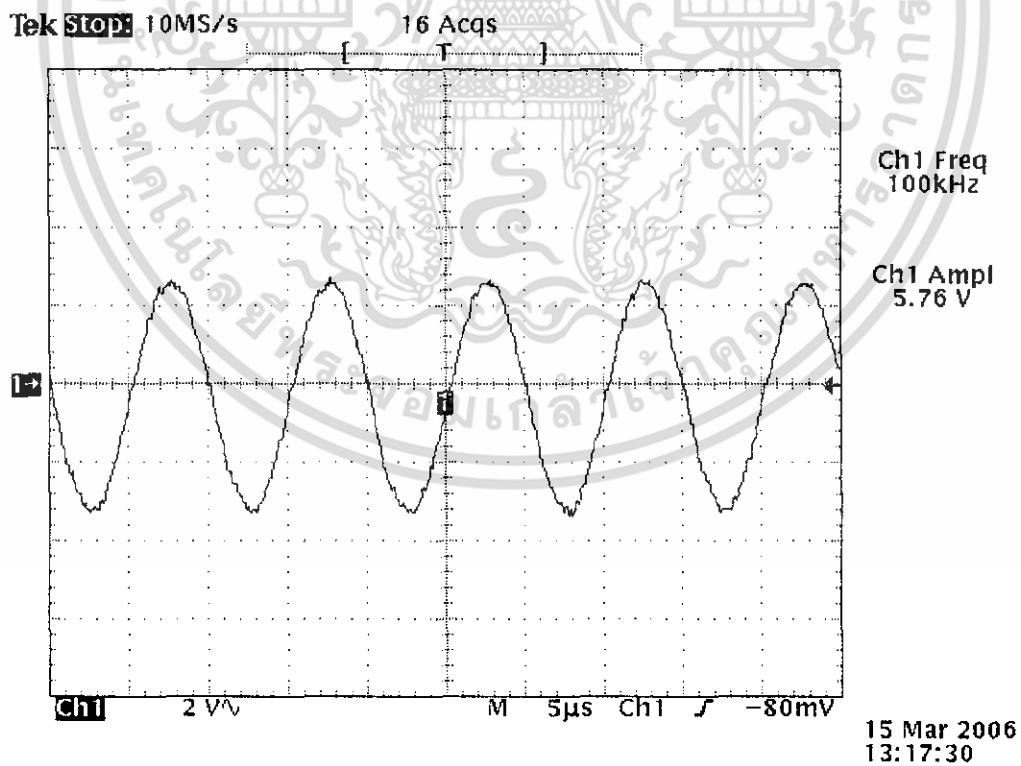


ง) รูปแสดงสัญญาณความถี่ 50 kHz จากโพลีโนเมียด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น เมื่อนำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

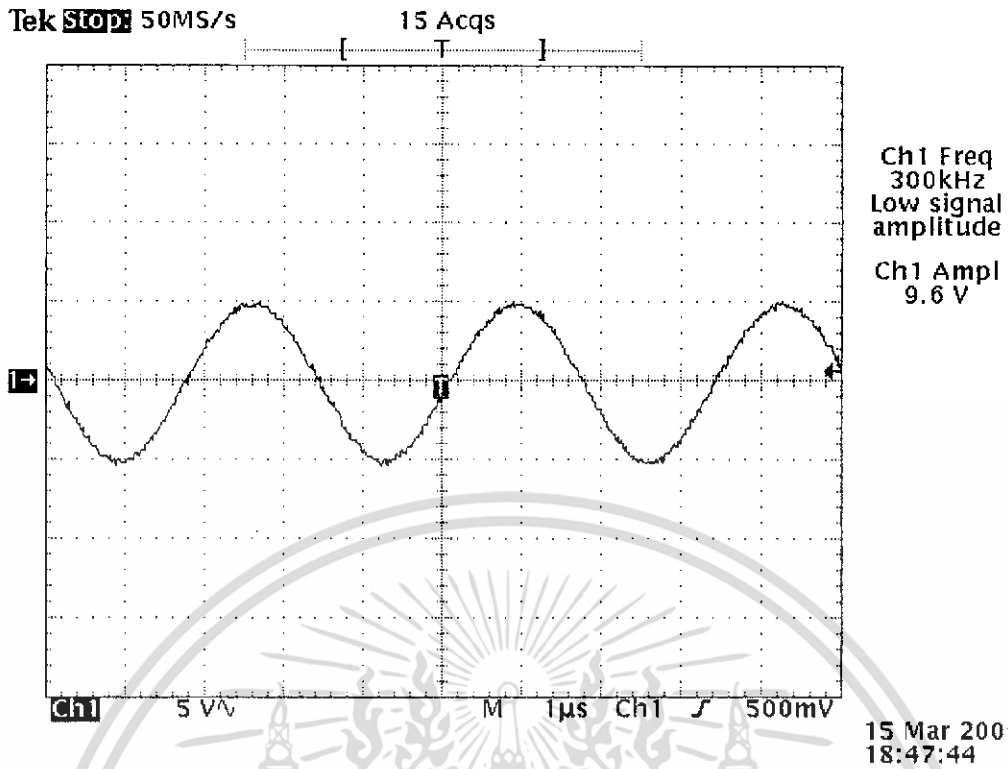


บ) รูปแสดงสัญญาณขาอินพุตที่ความถี่ 100 kHz จากตารางเปิดดู

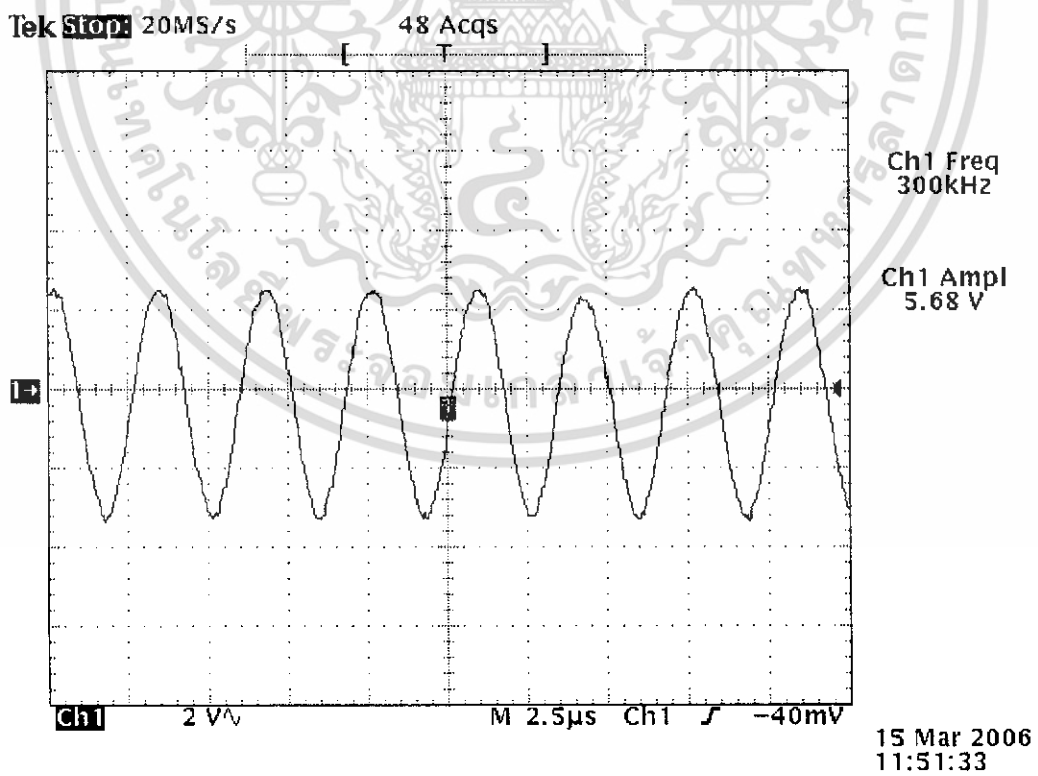


ป) รูปแสดงสัญญาณขาอินพุตที่ความถี่ 100 kHz จากโพลีโนเมียล

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้เพื่อการใช้งานเพื่อการศึกษาระดับบัณฑิตศึกษาเท่านั้น เมื่อผู้ยูสเซอร์เห็นแจ้งไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

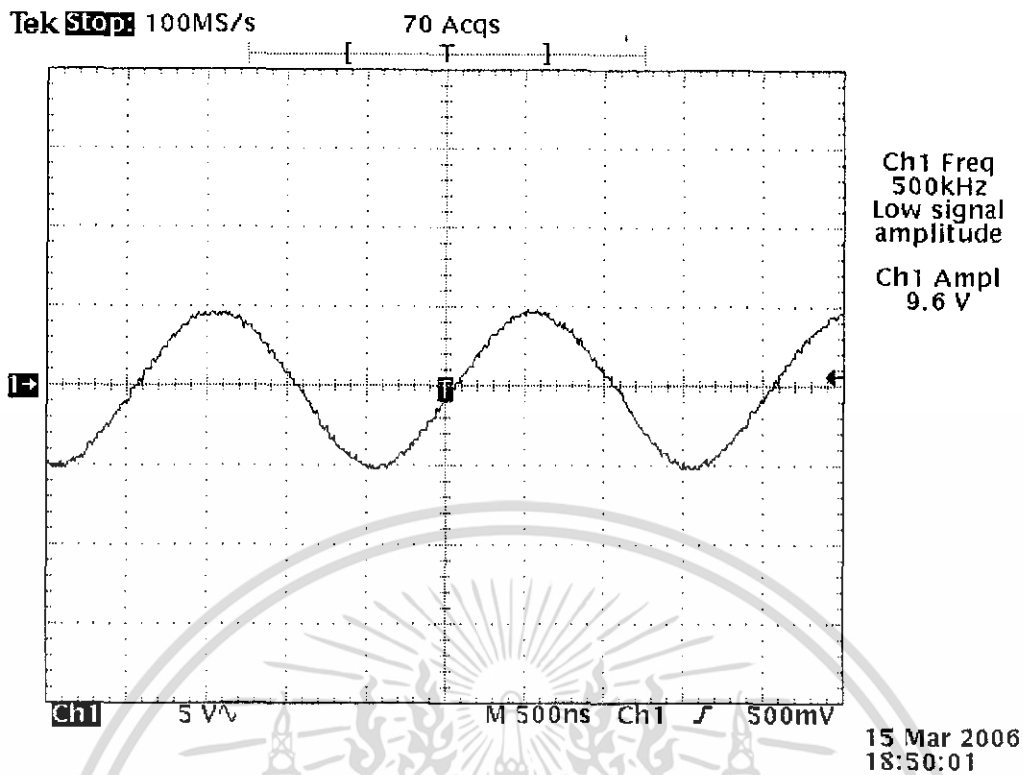


ผ) รูปแสดงสัญญาณขาอินพุตที่ความถี่ 300 kHz จากตารางเปิดดู

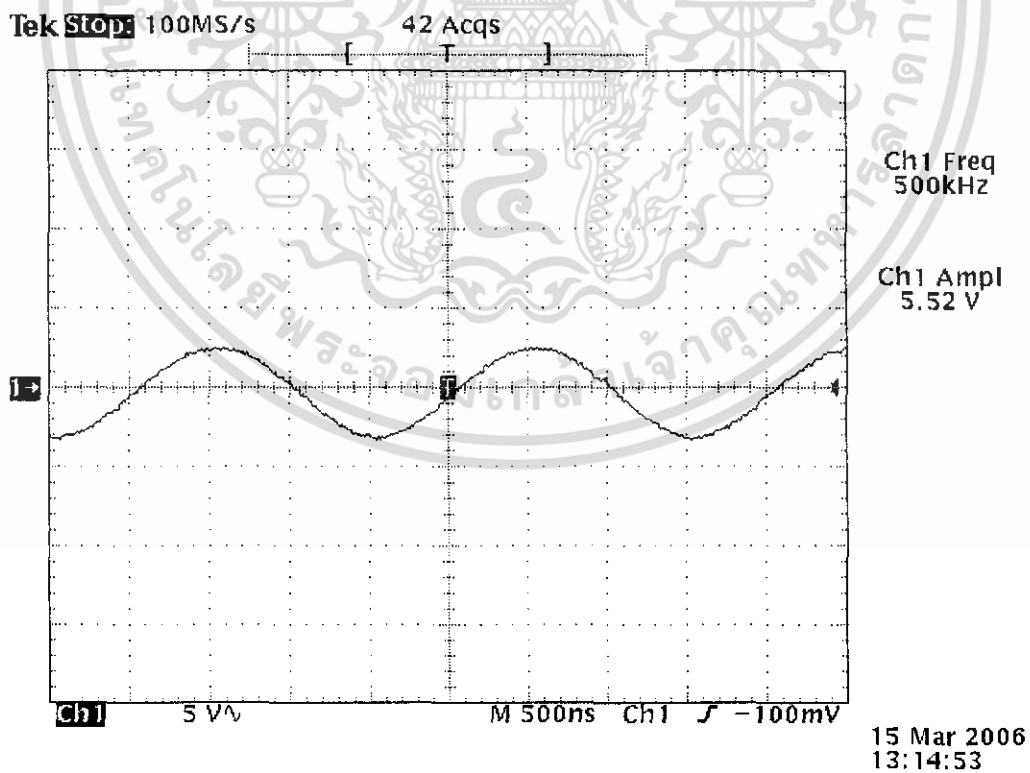


ผ) รูปแสดงสัญญาณขาอินพุตที่ความถี่ 300 kHz จากโพลีโนเมียล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

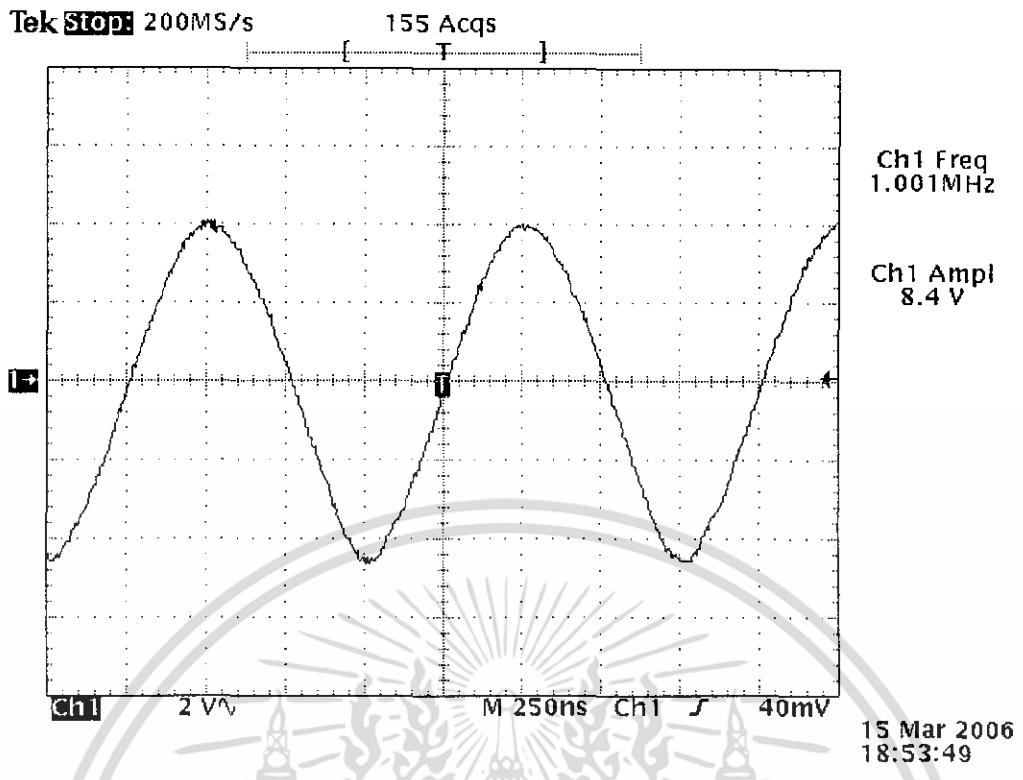


พ) รูปแสดงสัญญาณไซน์ที่มีความถี่ 500 kHz จากตารางเปิดดู

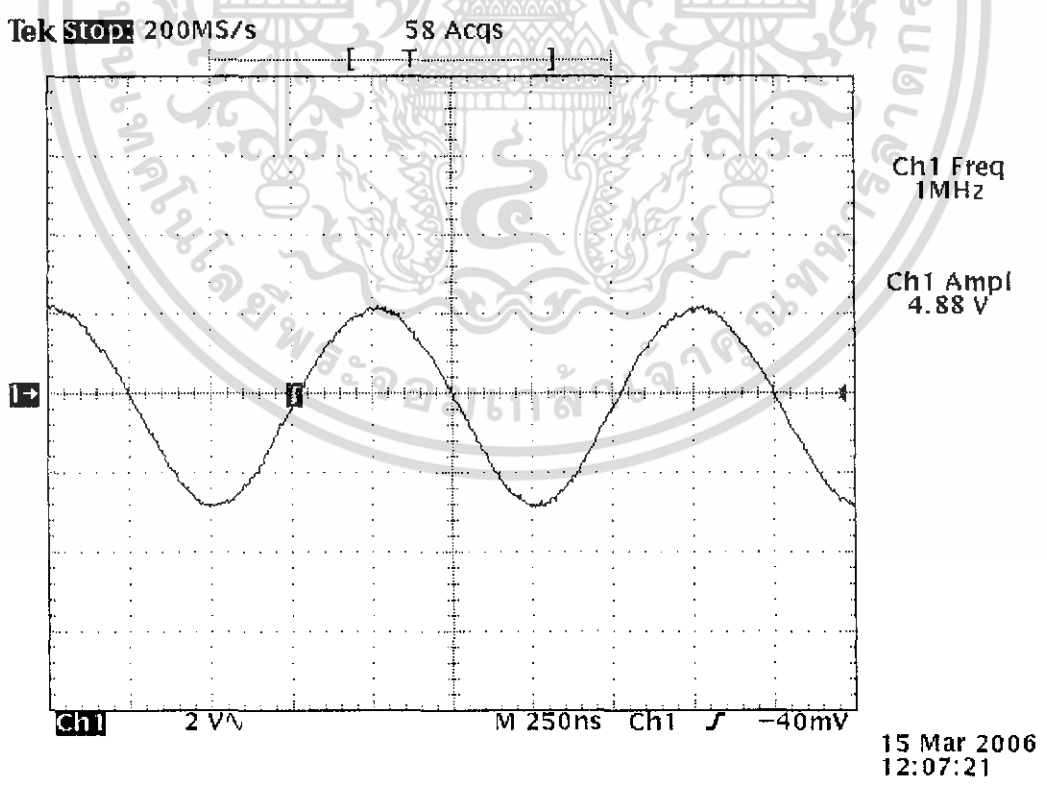


พ) รูปแสดงสัญญาณไซน์ที่มีความถี่ 500 kHz จากโพลีโนเมียล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

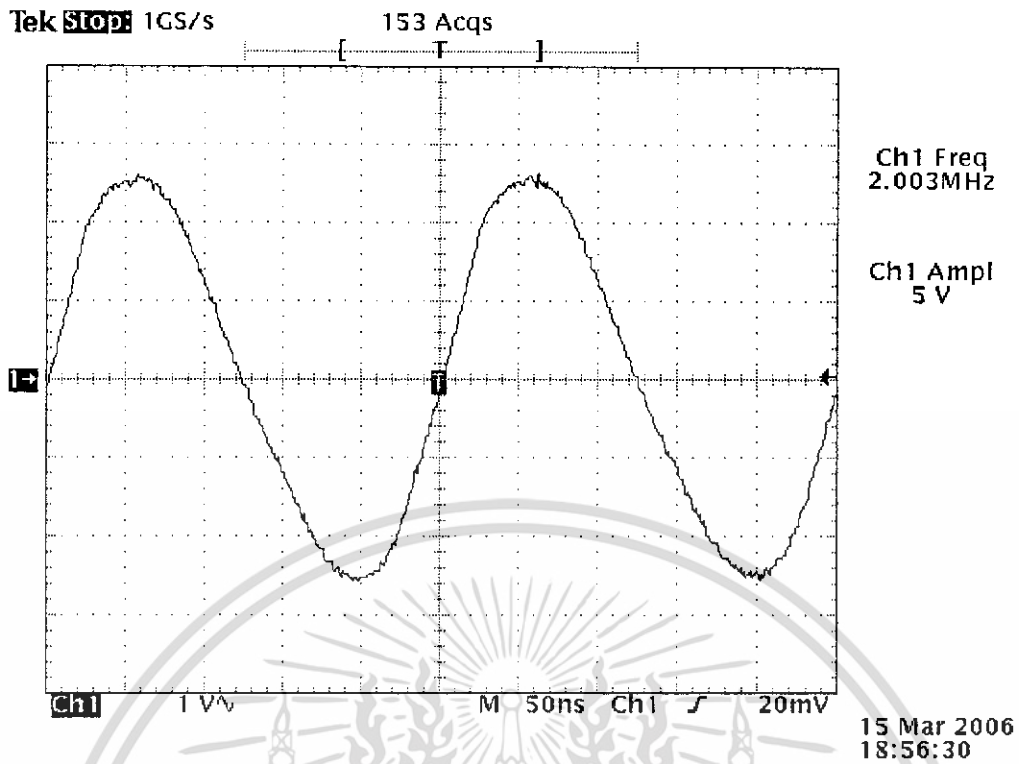


ก) รูปแสดงสัญญาณขาอินพุตที่ความถี่ 1 MHz จากตารางเปิดดู

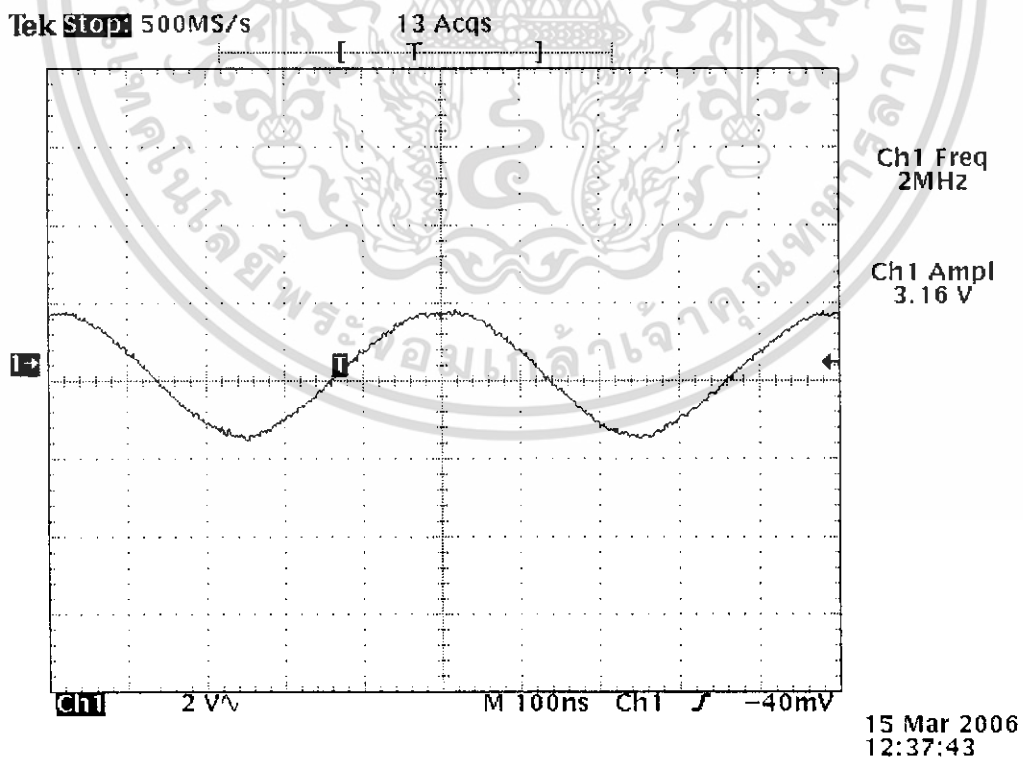


ข) รูปแสดงสัญญาณขาอินพุตที่ความถี่ 1 MHz จากโพลีโนเมียล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับใช้เพื่อการศึกษาเท่านั้น เมื่อผู้ใดเห็นปะโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

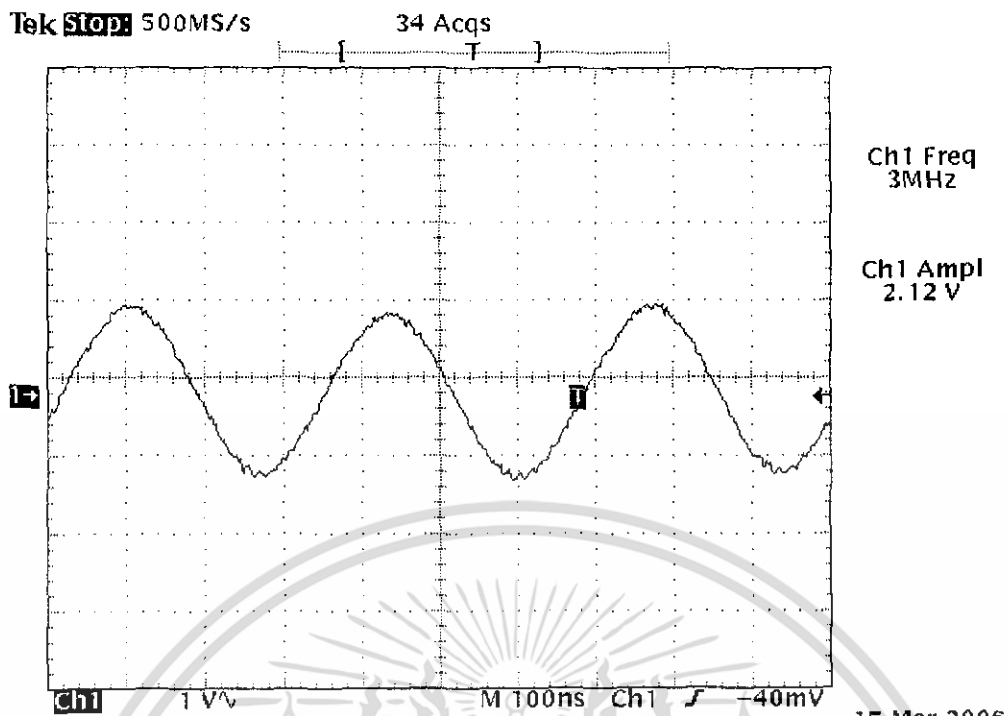


ย) รูปแสดงสัญญาณขาอินพุตที่ความถี่ 2 MHz จากตารางเปิดดู



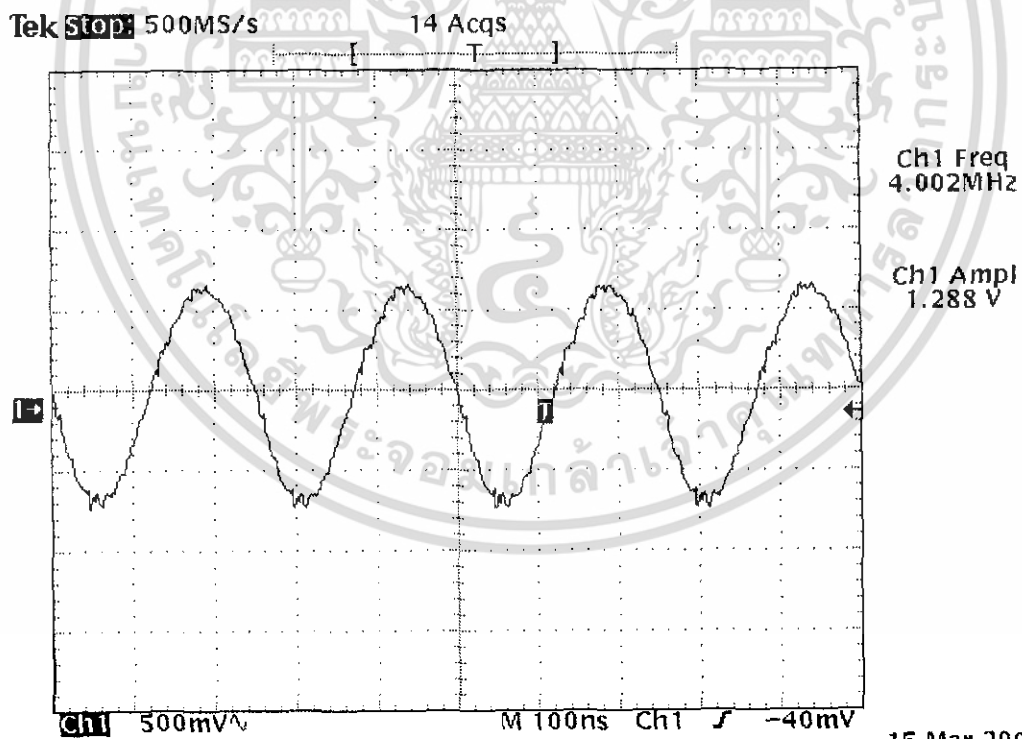
ร) รูปแสดงสัญญาณขาอินพุตที่ความถี่ 2 MHz จากโพลีโนเมียด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



15 Mar 2006
12:44:21

จ) รูปแสดงสัญญาณไซน์ที่มีความถี่ 3 MHz จากโพลีโนเมียด



15 Mar 2006
12:41:23

ค) รูปแสดงสัญญาณไซน์ที่มีความถี่ 4 MHz จากโพลีโนเมียด

รูปที่ 4.48 การทดสอบการกำเนิดสัญญาณไซน์เปรียบเทียบกับสัญญาณไซน์ที่ได้จาก

วิธีการโพลีโนเมียด ที่ความถี่ต่างๆ เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้เพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากผลการทดลองในการกำเนิดสัญญาณ โดยใช้วิธี Direct Digital Frequency Synthesizer :DDS จะใช้หลักการสร้างสองวิธี วิธีแบบแรกวิธีใช้ตารางเปิดดูได้สัญญาณรูปไซน์, สี่เหลี่ยม, สามเหลี่ยม, ฟันเลื่อย และสัญญาณสุ่ม ส่วนวิธีของโพลีโนเมียลจะได้สัญญาณไซน์ ซึ่งวิธีนี้จะสามารถผลิตความถี่ได้สูงกว่าแบบแรกมาก

ความถี่ของสัญญาณนาฬิกาที่ป้อนให้กับเอฟพีซีเอที่สังเคราะห์ความถี่คือ 24 เมกะเฮิร์ตซ์ และเอฟพีซีเอสังเคราะห์ความถี่สามารถกำเนิดสัญญาณไซน์ใช้วิธีตารางเปิดดู ได้ความถี่ตั้งแต่ 0 ถึง 500 กิโลเฮิร์ตซ์ และวิธีของโพลีโนเมียล ได้ความถี่ตั้งแต่ 0 ถึง 4 เมกะเฮิร์ตซ์ ซึ่งความเร็วในการสังเคราะห์สัญญาณเร็วกว่า และจำนวนเกทของเอฟพีซีเอใช้น้อยกว่าการใช้วิธีตารางเปิดดู



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5

บทสรุปและวิจารณ์

ทฤษฎีการออกแบบวงจรสังเคราะห์ความถี่ดิจิทัลโดยตรง(Direct Digital Frequency synthesizer : DDS) นั้นจะอาศัยการเก็บค่าขนาดของสัญญาณไว้ในตารางเปิดดู(Lookup table) แล้วใช้การชี้ตำแหน่งในการกำเนิดสัญญาณออกมา ซึ่งในการออกแบบค่าที่เก็บในตารางเปิดดูนั้นจะเก็บค่าขนาดของสัญญาณเพียง 1 ใน 4 คาบเท่านั้น ของสัญญาณ 1 คาบเท่านั้น แล้วอาศัยการจัดความสัมพันธ์ของอินพุทเพื่อให้ได้เอาท์พุทเป็นสัญญาณหนึ่งคาบ

ในการออกแบบวงจรสังเคราะห์ความถี่ดิจิทัลโดยตรงจะใช้ รีจิสเตอร์ความถี่ขนาด 32 บิต และใช้ค่าควบคุมความถี่ (fcw)ขนาด32 บิต ส่วนการกำเนิดสัญญาณโดยใช้วิธีตารางเปิดดูนั้นได้ลองทำการเก็บค่าแอมพลิจูด 4096 ค่า จำนวน 16 บิต และลองทำการเก็บค่าแอมพลิจูด 1024 ค่า จำนวน 16 บิต จาก การจำลองการทำงาน โดยใช้โปรแกรม ModelSim นั้น Romที่เก็บค่าแอมพลิจูด 4096 ค่า จำนวน 16 บิต จะใช้เวลานานกว่าในการเข้าถึงหน่วยความจำ ส่วนในการสร้างวิธี การประมาณค่าโพลีโนเมียล (The Polynomial Approximation)การทำงานจะได้ Maximum Frequency 1Hz - 4 MHz และจะใช้เวลาน้อยกว่าวิธีตารางเปิดดูในการเข้าถึงหน่วยความจำ ส่วนการกำเนิดสัญญาณโดยใช้วิธีตารางเปิดดูนั้นได้ Maximum Frequency 1Hz -1MHz

สำหรับการประยุกต์ใช้งานวงจรกำเนิดสัญญาณแบบดิจิทัลนั้นสามารถนำสัญญาณมาเข้ารหัสแอฟเอสเค (FSK) รวมทั้งระบบงานต่างๆไปที่ต้องการสัญญาณพื้นฐาน อาทิเช่น สัญญาณชานัน,สัญญาณสามเหลี่ยม,สัญญาณสี่เหลี่ยม หรือสัญญาณฟันเลื่อย เป็นต้น และอาจออกแบบให้ได้สัญญาณเฉพาะ

ในการศึกษาโครงการนี้ ทำให้มีความรู้ในการออกแบบฮาร์ดแวร์ด้วยภาษาวีเอชดีแอลที่มีความยืดหยุ่นในการออกแบบสูง ซึ่งข้อดีของการออกแบบด้วยภาษาวีเอชดีแอล คือวงจรที่ออกแบบสามารถนำไปสังเคราะห์และจัดวางเชื่อมโยงลงชิพ FPGA ได้

หนังสืออ้างอิง

1. D.L.Perry, "VHDL", McGraw-Hill Compony, Inc, volume 3, 1998
2. B. Goldberg, Direct techniques in frequency synthesis, McGraw Hill, 1996, New York.
3. V.F Kroupa, Direct digital frequency synthesizers, IEEE Press, 1999.
4. D.L.Perry, "VHDL", New York: McGraw-Hill, 1995.
5. J.G>Proakis and D.G.Manolakis, "Introduction to Digital Signal Processing", Macmillan publishing, 1988, p372-376
6. มนต์ สัจวรศิลป์, วรรัตน์ ภัทรอมรกุลม, "คู่มือการใช้งาน MATLAB ฉบับสมบูรณ์", ศูนย์การพิมพ์พลซ อินโฟเพลสมกรุงเทพ, 2543



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาคผนวก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.Module Div_2500

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity Div_2500 is
port
(   Clk_in   :in std_logic;
    Clk_out  :out std_logic
);
end Div_2500;
architecture rtl of Div_2500 is
begin
-----
    process(Clk_in)
        variable Clk_temp:std_logic:='0';
        variable count:integer range 0 to 1249;
        begin
            if Clk_in'Event and Clk_in='1'then
            -----
                if count<1249 then
                    count :=count+1;
                    Clk_temp:=Clk_temp;
                Else
                    Count :=0;
                    Clk_temp:=not(Clk_temp);
                end if;
            -----
            Clk_out<=Clk_temp;
            end if;
        end process;
    end rtl;
```

2.Module sereal_com

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity sereal_com is
port(
    BAUD_RATE :in std_logic;
    RX         :in std_logic;
    DATA_RX   :out std_logic_vector(7 downto 0);
    RX_EN      :out std_logic);
end sereal_com ;
-----
architecture rtl of sereal_com is
type State_type_RX is(idel,ReceiveData,Stop);
signal State_RX :State_type_RX:=Idel;
begin
-----
    process(BAUD_RATE,RX)
        variable RX_Data_Count:integer range 0 to 7;
        variable Buffer_RX :std_logic_vector(7 downto 0);
        begin
            -----
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

-----
if BAUD_RATE'Event and BAUD_RATE='1' then
  case State_RX is
-----
    when Idel=>RX_EN<='0';
      if RX='0' then
        RX_Data_Count:=0;
        State_RX    <=ReceiveData;
      Else
        RX_Data_Count:=0;
        State_RX    <=Idel;
      end if;
-----
    when ReceiveData=>RX_EN<='0';
      if RX_Data_Count=7 then
        Buffer_RX(RX_Data_Count):=RX;
        State_RX<=Stop;
      Else
        Buffer_RX(RX_Data_Count):=RX;
        RX_Data_Count:=RX_Data_Count+1;
        State_RX<=ReceiveData;
      end if;
-----
    when Stop=>RX_Data_Count:=0;
      data_rx<=buffer_rx;
      RX_EN<='1';
      State_RX<=Idel;
-----
    when others =>RX_Data_Count:=0;
      RX_EN<='0';
      State_RX<=Idel;
    end case;
-----
  end if;
end process ;
end rtl;

```

3.Module swit_puls

```

LIBRARY IEEE;
use ieee.std_logic_1164.all;
entity Onepulse is
port(Clk          :in std_logic;
      SW          :in std_logic;
      SW_Single_Pulse:out std_logic);
end Onepulse;

```

```

-----
architecture rtl of Onepulse is
signal sw_debounce_delay:std_logic;
begin
  process(Clk)
begin

```

```

-----
if(Clk'Event) and (Clk='1') then
  if(sw='1') and (sw_debounce_delay='0') then
    sw_single_pulse<='1';
  else

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        end if;
        sw_debounce_delay<=sw;
    end if;
-----
    end process;
end rtl;

```

4.Module RANDOM

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity RANDOM is
port (   Clock      : in std_logic;
        RANDOM_out  : out std_logic_vector(11 downto 0)
    );
end RANDOM;

architecture Behavioral of RANDOM is

signal reg : std_logic_vector( 31 downto 0);
signal feedback:std_logic;
begin
    process(Clock,reg,feedback)
    begin
        if( Clock'event and Clock = '1' ) then
            if reg="00000000000000000000000000000000" then
                reg<="00000000000000000000000000000001";
            else
                reg(31 downto 12)<=reg(30 downto 11);
                reg(11)<=reg(10) xor not feedback;
                reg(10)<=reg(9) xor not feedback;
                reg(9) <=reg(8) xor not feedback;
                reg(8) <=reg(7);
                reg(7) <=reg(6);
                reg(6) <=reg(5);
                reg(5) <=reg(4);
                reg(4) <=reg(3);
                reg(3) <=reg(2) xor not feedback;
                reg(2) <=reg(1);
                reg(1) <=reg(0);
                reg(0) <=feedback;
            end if;
        end if;
        feedback<=reg(31);
    end process;

        RANDOM_out<=reg(31 downto 20);
-----

```

```

end Behavioral;

```

5.Module select_input

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity select_input is

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

port(
    input_comm : in std_logic;
    input_mcs  : in std_logic;
    sel       : in std_logic;
    out_put   : out std_logic);

end ;
architecture behave of select_input is
signal k : std_logic;
begin
process(sel,input_comm,input_mcs)
begin
    k<=sel;
    if(k='0')then
        out_put<=input_comm;
    else
        out_put<=input_mcs;
    end if;
end process;
end behave;

```

6.Module Dff

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity Dff is
port(
    en : in std_logic;
    ip : in std_logic_vector(7 downto 0);
    sel_mux : out std_logic_vector(2 downto 0);
    sel : out std_logic_vector(1 downto 0);
    out_fre : out std_logic_vector(31 downto 0)
);
end Dff;
architecture rtl of Dff is
signal buf1,buf2,buf3,buf4: std_logic_vector(7 downto 0);
begin
process(en,ip)
VARIABLE count_rx : integer;
begin
    if en'event and en='1' then
        if count_rx < 4 then
            count_rx:=count_rx+1;
            buf1<=ip;
            buf2<=buf1;
            buf3<=buf2;
            buf4<=buf3;
        else
            count_rx:=0;
            out_fre<=buf1&buf2&buf3&buf4;
            sel_mux(2 downto 0)<=ip(4 downto 2);
            sel(1 downto 0)<=ip(1 downto 0);
        end if;
    end if;
end process;
end rtl;

```

7.Module Dff_com

```

library ieee;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

use ieee.std_logic_1164.all;
entity Dff_com is
port(
    en_1          :in std_logic;
    ip_1          :in std_logic_vector(7 downto 0);
    sel_mux       :out std_logic_vector(2 downto 0);
    out_fre       :out std_logic_vector(31 downto 0));
end Dff_com;
architecture rtl of Dff_com is
--signal buf:std_logic_vector(7 downto 0);
signal data,z5:std_logic_vector(3 downto 0);
signal buf1,buf2,buf3,buf4,buf5,buf6,buf7,buf8: std_logic_vector(3
downto 0);
signal z1,z2,z3,z4:std_logic_vector( 7 downto 0);
begin
p0:process(en_1,ip_1)
begin
--    if en'event and en='1' then
--        case ip_1 is
--            when "00110000" =>data <="0000";--0
--            when "00110001" =>data <="0001";--1
--            when "00110010" =>data <="0010";--2
--            when "00110011" =>data <="0011";--3
--            when "00110100" =>data <="0100";--4
--            when "00110101" =>data <="0101";--5
--            when "00110110" =>data <="0110";--6
--            when "00110111" =>data <="0111";--7
--            when "00111000" =>data <="1000";--8
--            when "00111001" =>data <="1001";--9
--            when "01100001" =>data <="1010";--a
--            when "01100010" =>data <="1011";--b
--            when "01100011" =>data <="1100";--c
--            when "01100100" =>data <="1101";--d
--            when "01100101" =>data <="1110";--e
--            when "01100110" =>data <="1111";--f
--            when others=>data<="0000";
--        end case;
--    end if;
end process p0;
p1:process(en_1,data)
VARIABLE count_rx : integer:=0;
begin
    if en_1'event and en_1='1' then
        if count_rx < 8 then
            count_rx:=count_rx+1;
            buf1<=data;
            buf2<=buf1;
            buf3<=buf2;
            buf4<=buf3;
            buf5<=buf4;
            buf6<=buf5;
            buf7<=buf6;
            buf8<=buf7;
        else
            count_rx:=0;
            z1<=buf7 & buf8;
            z2<=buf5 & buf6;
            z3<=buf3 & buf4;
            z4<=buf1 & buf2;
            z5<=data;
        end if;
    end if;
end process p1;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

out_fre<= z4 & z3 & z2 & z1;
sel_mux<=z5(2 downto 0);
end rtl;

```

8.Module selec_data

```

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity selec_data is
port(
    data      : in std_logic_vector(7 downto 0);
    sel_1     : in std_logic;
    output_mcs : out std_logic_vector(7 downto 0);
    output_com : out std_logic_vector(7 downto 0));
end ;
architecture behave of selec_data is
signal k : std_logic;
begin
process(sel_1,data)
begin
    k<=sel_1;
    if(k='0')then
        output_com<=data;
    else
        output_mcs<=data;
    end if;
end process;
end behave;

```

9.Module data

```

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity data is
port(
    sel_1      : in std_logic;
    mux_com    : in std_logic_vector(2 downto 0);
    data_com   : in std_logic_vector(31 downto 0);
    mux_mcs    : in std_logic_vector(2 downto 0);
    data_mcs   : in std_logic_vector(31 downto 0);
    mux_sel    : out std_logic_vector(2 downto 0);
    data_in    : out std_logic_vector(31 downto 0));
end ;
architecture behave of data is
signal k : std_logic;
begin
process(sel_1,data_com,data_mcs,mux_com,mux_mcs)
begin
    k<=sel_1;
    if(k='0')then
        data_in<=data_com;
        mux_sel<=mux_com;
    else
        data_in<=data_mcs;
        mux_sel<=mux_mcs;
    end if;
end process;
end behave;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

10.Module lut

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.std_logic_unsigned.all;
-----
entity lut is
port (   clk       : in std_logic;
        rst       : in std_logic;
        sel       : in std_logic_vector(1 downto 0);
        freq_data : in std_logic_vector(31 downto 0);
        dout      : out std_logic_vector(11 downto 0)
    );
end lut;
-----
architecture rtl of lut is
-----
signal address      : unsigned( 9 downto 0);
signal dout1,dout2,dout3 : std_logic_vector (15 downto 0);
signal result      : std_logic_vector (15 downto 0);
signal result1     : std_logic_vector (15 downto 0);
signal q1,q2,q3    : std_logic_vector (11 downto 0);
signal accum       : std_logic_vector (31 downto 0);
signal sign        : std_logic;
begin
-----
acc:process(clk,rst,freq_data)
variable count1:std_logic_vector(31 downto 0);
begin
    count1:= freq_data ;
    if rst = '0' then
        accum<="00000000000000000000000000000000";
    elsif clk'event and clk='1' then
        accum<=count1 + accum;
    end if;
end process;
-----
sign <= accum(31);
-----
process(accum)
begin
    if accum(30) = '0' then
        address <= unsigned ( accum (29 downto 20));
    else
        address <= unsigned (not accum (29 downto 20));
    end if;
end process;
-----
out_squre:process( accum)
begin
    if accum(31) = '0' then
        dout2 <= "1111111111111111";
    else
        dout2 <= (others => '0');
    end if;
end process;
-----
lookup_sin:process(address,sign)
subtype slv7 is std_logic_vector(15 downto 0);
type rom1024 is array (0 to 1023 ) of slv7;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
variable sinus_rom : rom1024 :=(
"0000000000000000", "0000000000000000", "0000000000000000", "000000000000
0000", "0000000000000001", "0000000000000001", "0000000000000001", "0000
000000000010", "0000000000000010", "0000000000000011",
"00000000000000100", "00000000000000101", "00000000000000110", "0000000000
00111", "00000000000001000", "00000000000001001", "00000000000001010", "0000
0000000001011", "00000000000001100", "00000000000001110",
"00000000000001111", "00000000000010001", "00000000000010011", "0000000000
10100", "00000000000010110", "00000000000011000", "00000000000011010", "0000
0000000011100", "00000000000011110", "00000000000100000",
"00000000000100011", "00000000000100101", "00000000000100111", "00000000001
01010", "00000000000101101", "00000000000101111", "00000000000110010", "0000
0000000110101", "00000000000111000", "00000000000111011",
"00000000000111110", "0000000001000001", "0000000001000100", "00000000010
00111", "0000000001001011", "0000000001001110", "0000000001010010", "0000
000001010101", "0000000001011001", "0000000001011101",
"0000000001100000", "0000000001100100", "0000000001101000", "00000000011
01100", "0000000001110000", "0000000001110101", "0000000001111001", "0000
000001111101", "00000000010000010", "00000000010000110",
"00000000010001011", "00000000010001111", "000000000100010100", "00000000100
11001", "00000000010011110", "000000000100100011", "00000000010101000", "0000
0000010101101", "00000000010110010", "00000000010110111",
"00000000010111101", "00000000011000010", "00000000011000100", "000000000110
01101", "00000000011010011", "00000000011011001", "00000000011011110", "0000
0000011100100", "00000000011101010", "00000000011110000",
"00000000011110110", "00000000011111101", "0000000100000011", "00000001000
01001", "0000000100010000", "0000000100010110", "0000000100011101", "0000
000100100011", "0000000100101010", "0000000100110001",
"0000000100111000", "0000000100111111", "00000001010000110", "00000001010
01101", "0000000101010100", "0000000101011011", "0000000101100011", "0000
000101101010", "0000000101110010", "0000000101111001",
"0000000110000001", "0000000110001000", "0000000110010000", "00000001100
11000", "0000000110100000", "0000000110101000", "0000000110110000", "0000
000110111000", "0000000111000001", "0000000111001001",
"0000000111010001", "0000000111011010", "0000000111100010", "00000001111
01011", "0000000111110100", "0000000111111101", "00000010000000101", "0000
001000001110", "0000001000010111", "0000001000100000",
"0000001000101010", "0000001000110011", "0000001000111100", "00000010010
00110", "0000001001001111", "0000001001001011001", "0000001001100010", "0000
001001101100", "0000001001110110", "0000001001111111",
"0000001010001001", "0000001010010011", "0000001010011101", "00000010101
01000", "0000001010110010", "0000001010111100", "0000001011000110", "0000
001011010001", "0000001011011011", "0000001011100110",
"0000001011110001", "0000001011111011", "0000001100000110", "00000011000
10001", "0000001100011100", "0000001100100111", "0000001100110010", "0000
001100111110", "0000001101001001", "0000001101010100",
"0000001101100000", "0000001101101011", "0000001101110111", "00000011100
00010", "0000001110001110", "0000001110011010", "0000001110100110", "0000
001110110010", "0000001110111110", "0000001111001010",
"0000001111010110", "0000001111100010", "0000001111101111", "00000011111
11011", "00000100000000111", "00000100000010100", "0000010000100001", "0000
010000101101", "0000010000111010", "0000010001000111",
"0000010001010100", "0000010001100001", "0000010001101110", "00000100011
1011", "0000010010001000", "0000010010010110", "0000010010100011", "0000
01001011000", "0000010010111110", "0000010011001100",
"0000010011011001", "0000010011100111", "0000010011110101", "00000101000
00011", "0000010100010001", "0000010100011111", "0000010100101101", "0000
010100111011", "0000010101001001", "0000010101011000",
"0000010101100110", "0000010101110100", "0000010110000011", "00000101100
10010", "0000010110100000", "0000010110101111", "0000010110111110", "0000
010111001101", "0000010111011100", "0000010111101011",
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

"0001011101011001", "000101110110110", "0001011110010011", "00010111101
10000", "000101111001110", "0001011111101011", "0001100000001000", "0001
100000100110", "0001100001000011", "0001100001100000",
"0001100001111110", "0001100010011100", "0001100010111001", "00011000110
10111", "000110001110101", "0001100100010011", "0001100100110000", "0001
100101001110", "000110010101101", "0001100110001011",
"0001100110101001", "0001100111000111", "0001100111100101", "00011010000
00100", "0001101000100010", "0001101001000000", "0001101001011111", "0001
101001111110", "0001101010011100", "0001101010111011",
"0001101011011010", "0001101011111001", "0001101100010111", "00011011001
10110", "0001101101010101", "0001101101110101", "0001101110010100", "0001
101110110011", "0001101111010010", "0001101111110001",
"0001110000010001", "0001110000110000", "0001110001010000", "00011100011
01111", "0001110010001111", "0001110010101111", "0001110011001110", "0001
110011101110", "0001110100001110", "0001110100101110",
"0001110101001110", "0001110101101110", "0001110110001110", "00011101101
01110", "0001110111001110", "0001110111101111", "0001111000001111", "0001
111000101111", "0001111001010000", "0001111001110000",
"0001111010010001", "0001111010110010", "0001111011010010", "00011110111
10011", "0001111100010100", "0001111100110101", "0001111101010110", "0001
111101110111", "0001111100110000", "0001111110111001",
"0001111111011010", "000111111111011", "0010000000011100", "00100000001
11110", "0010000001011111", "0010000010000000", "0010000010100010", "0010
000011000100", "0010000011100101", "0010000010000011",
"0010000100101001", "0010000101001010", "0010000101101100", "00100001100
01110", "0010000110110000", "0010000111010010", "0010000111110100", "0010
001000010110", "0010001000111000", "0010001001011011",
"0010001001111101", "0010001010011111", "0010001011000010", "00100010111
00100", "0010001100000111", "0010001100101001", "0010001101001100", "0010
001101101111", "0010001110010001", "0010001110110100",
"0010001111010111", "001000111111010", "0010010000011101", "00100100010
00000", "0010010001100011", "0010010010000110", "0010010010101001", "0010
010011001100", "0010010011110000", "0010010100010011",
"0010010100110111", "0010010101010101", "0010010101111110", "00100101101
00001", "0010010111000101", "0010010111101000", "0010011000001100", "0010
011000110000", "0010011001010100", "0010011001111000",
"0010011010011100", "0010011011000000", "0010011011100100", "00100111000
01000", "0010011100101100", "0010011101010000", "0010011101110100", "0010
011110011001", "0010011110111101", "0010011111100010",
"0010100000000110", "0010100000101011", "0010100001001111", "00101000011
10100", "0010100010011001", "0010100010111101", "0010100011100010", "0010
100100000111", "0010100100101100", "0010100101010001",
"0010100101110110", "0010100110011001", "0010100111000000", "00101001111
00101", "0010101000001010", "0010101000110000", "0010101001010101", "0010
101001111010", "0010101010100000", "0010101011000101",
"0010101011101011", "0010101100010000", "0010101100110110", "00101011010
11100", "0010101110000001", "0010101110100111", "0010101111001101", "0010
10111110011", "0010110000011001", "0010110000111111",
"0010110001100101", "0010110010001011", "0010110010110001", "00101100110
10111", "0010110011111101", "0010110100100100", "0010110101001010", "0010
110101110000", "0010110110010111", "00101101101111101",
"0010110111100100", "0010111000001011", "0010111000110001", "00101110010
11000", "0010111001111111", "0010111010100101", "0010111011001100", "0010
111011110011", "0010111100011010", "0010111101000001",
"0010111101101000", "0010111110001111", "0010111110110110", "00101111110
11101", "0011000000000101", "0011000000101100", "0011000001010011", "0011
000001111011", "0011000010100010", "0011000011001001",
"0011000011110001", "0011000100011000", "0011000101000000", "00110001011
01000", "0011000110001111", "0011000110110111", "0011000111011111", "0011
001000000111", "0011001000101111", "0011001001010111",

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

"0101010011100001", "0101010100010000", "0101010100111111", "01010101011
01111", "0101010110011110", "0101010111001110", "0101010111111101", "0101
011000101101", "0101011001011100", "0101011010001100",
"0101011010111011", "0101011011101011", "0101011100011011", "01010111010
01010", "0101011101111010", "0101011110101010", "0101011111011001", "0101
100000001001", "01011000000111001", "0101100001101001",
"0101100010011000", "0101100011001000", "0101100011111000", "01011001001
01000", "0101100101011000", "0101100110001000", "0101100110111000", "0101
100111101000", "0101101000011000", "0101101001001000",
"0101101001111000", "0101101010101000", "0101101011011000", "01011011000
01000", "0101101100111000", "0101101101101000", "0101101110011001", "0101
101111001001", "0101101111111001", "0101110000101001",
"0101110001011001", "0101110010001010", "0101110010111010", "01011100111
01010", "0101110100011011", "01011101010001011", "0101110101111100", "0101
110110101100", "0101110111011100", "0101111000001101",
"0101111000111101", "0101111001101110", "0101111010011110", "01011110110
01111", "010111101111111", "0101111100110000", "0101111101100001", "0101
111110010001", "0101111111000010", "010111111110010",
"0110000000100011", "0110000001010100", "0110000010000101", "01100000101
10101", "0110000011100110", "0110000100010111", "0110000101001000", "0110
000101111000", "0110000110101001", "011000011011010",
"0110001000001011", "0110001000111100", "0110001001101101", "01100010100
11110", "0110001011001111", "0110001011111111", "0110001100110000", "0110
001101100001", "0110001110010010", "0110001111000011", "0110001111000011",
"011000111110100", "0110010000100110", "0110010001010111", "01100100100
01000", "0110010010111001", "0110010011101010", "0110010100011011", "0110
01010001100", "0110010101111101", "0110010110101111",
"0110010111100000", "0110011000010001", "0110011001000010", "01100110011
10011", "01100110101000101", "0110011011010110", "0110011100000111", "0110
011100111001", "0110011101101010", "0110011110011011",
"0110011111001101", "0110011111111110", "0110100000101111", "01101000011
00001", "0110100010010010", "0110100011000100", "0110100011110101", "0110
100100100110", "0110100101011000", "0110100110001001",
"0110100110111011", "0110100111101100", "0110101000011110", "01101010010
01111", "0110101010000001", "0110101010110011", "0110101011100100", "0110
101100010110", "0110101101000111", "0110101101111001",
"0110101110101011", "0110101111011100", "0110110000001110", "01101100001
1111", "0110110001110001", "0110110010100011", "0110110011010101", "0110
110100000110", "0110110100111000", "0110110101101010",
"0110110110011011", "0110110111001101", "0110110111111111", "01101110001
10001", "0110111001100010", "0110111010010100", "0110111011000110", "0110
111011111000", "0110111100101010", "0110111101011100",
"0110111110001101", "0110111110111111", "0110111111110001", "01110000001
00011", "0111000001010101", "0111000010000111", "0111000010111001", "0111
000011101011", "0111000100011100", "0111000101001110",
"0111000110000000", "0111000110110010", "0111000111100100", "01110010000
10110", "0111001001001000", "0111001001111010", "0111001010101100", "0111
001011011110", "0111001100010000", "0111001101000010",
"0111001101101000", "0111001110100110", "0111001111011000", "01110100000
01010", "0111010000111100", "0111010001101110", "0111010010100000", "0111
010011010011", "0111010100000101", "0111010100110111",
"0111010101101001", "0111010110011011", "0111010111001101", "01110101111
1111", "0111011000110001", "0111011001100011", "0111011010010101", "0111
011011001000", "0111011011111010", "0111011100101100",
"0111011101011110", "0111011110010000", "0111011111000010", "01110111111
10100", "0111100000100111", "0111100001011001", "0111100010001011", "0111
100010111101", "0111100011101111", "0111100100100010",
"0111100101010100", "0111100110000110", "0111100110111000", "01111001111
01010", "0111101000011101", "0111101001001111", "0111101010000001", "0111
101010110011", "0111101011100101", "0111101100011000",

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้


```

"0111000010000000", "0111000010100000", "0111000011000000", "01110000111
00000", "0111000100000000", "0111000100100000", "0111000101000000", "0111
000101100000", "0111000110000000", "0111000110100000",
"0111000111000000", "0111000111100000", "0111001000000000", "01110010001
00000", "0111001001000000", "0111001001100000", "0111001010000000", "0111
001010100000", "0111001011000000", "0111001011100000",
"0111001100000000", "0111001100100000", "0111001101000000", "01110011011
00000", "0111001110000000", "0111001110100000", "0111001111000000", "0111
00111100000", "0111010000000000", "0111010000100000",
"0111010001000000", "0111010001100000", "0111010010000000", "01110100101
00000", "0111010011000000", "0111010011100000", "0111010100000000", "0111
010100100000", "0111010101000000", "0111010101100000",
"0111010110000000", "0111010110100000", "0111010111000000", "01110101111
00000", "0111011000000000", "0111011000100000", "0111011001000000", "0111
011001100000", "0111011010000000", "0111011010100000",
"0111011011000000", "0111011011100000", "0111011100000000", "01110111001
00000", "0111011101000000", "0111011101100000", "0111011110000000", "0111
011101100000", "0111011110000000", "0111011111000000",
"0111100000000000", "0111100000100000", "0111100001000000", "01111000011
00000", "0111100010000000", "0111100010100000", "0111100011000000", "0111
10001100000", "0111100100000000", "0111100100100000",
"0111100101000000", "0111100101100000", "0111100110000000", "01111001101
00000", "0111100111000000", "0111100111100000", "0111101000000000", "0111
101000100000", "0111101001000000", "0111101001100000",
"0111101010000000", "0111101010100000", "0111101011000000", "01111010111
00000", "0111101100000000", "0111101100100000", "0111101101000000", "0111
101101100000", "0111101101100000", "0111101110000000", "0111101110100000",
"0111101111000000", "0111101111000000", "0111110000000000", "01111100001
00000", "0111110001000000", "0111110001100000", "0111110010000000", "0111
110010100000", "0111110011000000", "0111110011100000",
"0111110100000000", "0111110100100000", "0111110101000000", "01111101011
00000", "0111110110000000", "0111110110100000", "0111110111000000", "0111
11011100000", "0111111000000000", "0111111000100000",
"0111111001000000", "0111111001100000", "0111111010000000", "01111110101
00000", "0111111011000000", "0111111011100000", "0111111100000000", "0111
111100100000", "0111111101000000", "0111111101100000",
"0111111110000000", "0111111110100000", "0111111111000000", "01111111111
00000");
begin
    if sign = '0' then
        result1 <= sawtooth_rom(to_integer(address));
    else
        result1 <= std_logic_vector (-
signed(sawtooth_rom(to_integer(address))));
    end if;
end process;
-----
out_sawtooth:process(clk,rst)
begin
    if rst ='1' then
        dout3 <= (others => '0');
    elsif rising_edge(clk) then
        dout3 <= result1;
        dout3(15) <= result1(15);
        dout3(14 downto 0) <= not result1 ( 14 downto 0);
    end if;
end process;
-----
q1<= dout1(15 downto 4 );
q2<= dout2(15 downto 4 );
q3<= dout3(15 downto 4 );
-----

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ทางการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        dout<= q1 when sel ="00" else
              q2 when sel ="01" else
              q3 when sel ="10" else
              accum(31 downto 20);
end rtl;

```

11.Module Polynomial

11.1 Module Polynomial

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Polynomial is
port(
    resetn          :in std_logic;
    clock           :in std_logic;
    w               :in std_logic_vector(31
downto 0);
    --phase_2msb_out :out std_logic_vector(1 downto 0);
    sin_out         :out std_logic_vector(11 downto
0)
);
end Polynomial;

architecture Behavioral of Polynomial is

component acm32
port (
    reset          :in std_logic;
    clk            :in std_logic;
    w              :in std_logic_vector(31 downto 0);
    phase_out      :out std_logic_vector(31 downto 0)
);
end component;

component MUL10MODULE
Port (
    resetn        : in std_logic;
    clock         : in std_logic;
    x             : in std_logic_vector(9 downto 0);
    y             : in std_logic_vector(9 downto 0);
    p             : out std_logic_vector(19 downto 0)
);
end component;

component a1
Port (
    resetn        : in std_logic;
    clock         : in std_logic;
    x             : in std_logic_vector(19 downto 0);
    y             : out std_logic_vector(28 downto 0)
);
end component;

component a2
Port (
    resetn        : in std_logic;
    clock         : in std_logic;
    x             : in std_logic_vector(19 downto 0);
    y             : out std_logic_vector(5 downto 0));

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่นอนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

end component;
-----
-----

signal      phase                :std_logic_vector(31
downto 0);
signal      phase_2msb           :std_logic_vector(1
downto 0);
signal      phase_9downto0       :std_logic_vector(9
downto 0);
signal      phase_decode         :std_logic_vector(9
downto 0);

signal      m2                   :std_logic_vector(19 downto 0);
signal      m4                   :std_logic_vector(19 downto 0);
signal      a1_out               :std_logic_vector(28 downto 0);
signal      a2_out               :std_logic_vector(5  downto 0);
signal      m1_power9            :std_logic_vector(29 downto 0);

signal      s                    :unsigned(9  downto 0);
signal      sum                  :std_logic_vector(9
downto 0);

signal      sin_out_reg          :std_logic_vector(11 downto 0);
signal      wave                 :std_logic_vector(19 downto 0);

signal      msb_d1,msb_d2,msb_d3,msb_d4,msb_d5 :std_logic;
signal      m2_d1                :std_logic_vector(19 downto 0);
signal      m_d1,m_d2,m_d3,m_d4  :std_logic_vector(9  downto 0);
-----
-----

begin
phase_acm32:acm32
port map (
    reset    => resetn,
    clk      => clock,
    w        => w,
    phase_out => phase
);

phase_2msb    <= phase(31 downto 30);
--phase_9downto0 <= phase(29 downto 20);
-----
-----

process(phase_2msb)
begin
    if phase_2msb(0) = '0' then
        phase_9downto0 <= ( phase(29 downto 20));
    else
        phase_9downto0 <= (not phase(29 downto 20));
    end if;
end process;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
-----  
--phase_2msb_out  <= phase_2msb;  
phase_decode      <= phase_9downto0;  
-----
```

```
delay_msb:process( resetn,clock)  
begin  
  if resetn='0' then  
    msb_d1<='0';  
    msb_d2<='0';  
    msb_d3<='0';  
    msb_d4<='0';  
    msb_d5<='0';  
  elsif clock'event and clock ='1' then  
    msb_d1<=phase_2msb(1);  
    msb_d2<=msb_d1;  
    msb_d3<=msb_d2;  
    msb_d4<=msb_d3;  
    msb_d5<=msb_d4;  
  end if;  
end process delay_msb;  
-----
```

```
delay_m:process( resetn,clock)  
begin  
  if resetn='0' then  
    m_d1<=(others=>'0');  
    m_d2<=(others=>'0');  
    m_d3<=(others=>'0');  
    m_d4<=(others=>'0');  
  elsif clock'event and clock ='1' then  
    m_d1<=phase_decode;  
    m_d2<=m_d1;  
    m_d3<=m_d2;  
    m_d4<=m_d3;  
  end if;  
end process delay_m;  
-----
```

```
m_2:MUL10MODULE  
Port map(  
  resetn => resetn,  
  clock  => clock,  
  x      => phase_decode,  
  y      => phase_decode,  
  p      => m2  
);  
-----
```

```
delay_m2:process( resetn,clock)  
begin  
  if resetn='0' then  
    m2_d1<=(others=>'0');  
  elsif clock'event and clock ='1' then  
    m2_d1<=m2;  
  end if;  
end process delay_m2;  
-----
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

m_4:MUL10MODULE
Port map(
    resetn => resetn,
    clock  => clock,
    x      => m2(19 downto 10),
    y      => m2(19 downto 10),
    p      => m4
);

a1_m2:a1
Port map(
    resetn => resetn,
    clock  => clock,
    x      => m2_d1,
    y      => a1_out
);

a2_m4:a2
Port map(
    resetn => resetn,
    clock  => clock,
    x      => m4,
    y      => a2_out
);
-----

m1_power9 <="111011100110101100101000000000";

process( resetn,clock)
begin
    if resetn='0' then
        s<=(others=>'0');
    elsif clock'event and clock = '1' then
        s<=unsigned(m1_power9(29 downto 20))-unsigned(a1_out(28
downto 20))+unsigned(a2_out);
    end if;
end process;

sum <= std_logic_vector(s(9 downto 0));

m_mul_m:MUL10MODULE
Port map(
    resetn => resetn,
    clock  => clock,
    x      => sum,
    y      => m_d4,
    p      => wave
);

wave_out:process( resetn,clock)
begin
    if resetn='0' then
        sin_out_reg<=(others=>'0');
    elsif clock'event and clock = '1' then
        if msb_d5='0' then
            sin_out_reg<='1' & (wave(19 downto 9));
        else
            sin_out_reg<='0' & not (wave(19 downto 9));
        end if;
    end if;
end process wave_out;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        sin_out <= sin_out_reg;
end Behavioral;

```

11.2 Module acm32

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity acm32 is
port(
    reset      :in std_logic;
    clk        :in std_logic;
    w          :in std_logic_vector(31 downto 0);
    phase_out  :out std_logic_vector(31 downto 0)
);
end acm32;
architecture rtl of acm32 is
-----
component adder8
port( in1 :in std_logic_vector(7 downto 0);
      in2 :in std_logic_vector(7 downto 0);
      cin :in std_logic;
      res_out :out std_logic_vector(7 downto 0);
      cout :out std_logic);
end component;
-----
signal w_d1,w_d2,w_d3,w_d4,w_d5,w_d6:std_logic_vector(31 downto 0);
signal to_end : std_logic;

signal acc0_in,acc1_in,acc2_in,acc3_in:std_logic_vector(7 downto 0);
signal acc0_out,acc1_out,acc2_out,acc3_out:std_logic_vector(7 downto 0);
signal c_acc0,c_acc1,c_acc2,c_acc3:std_logic;
signal cl_acc0,cl_acc1,cl_acc2,cl_acc3:std_logic;

signal accp0_in,accp1_in,accp2_in,accp3_in:std_logic_vector(7 downto 0);
signal accp0_out,accp1_out,accp2_out,accp3_out:std_logic_vector(7
downto 0);
signal c_accp0,c_accp1,c_accp2,c_accp3:std_logic;
signal cl_accp0,cl_accp1,cl_accp2,cl_accp3:std_logic;

signal c_d1,c_d2,c_d3:std_logic;

signal pre0_1,pre0_2,pre0_3:std_logic_vector(7 downto 0);
signal pre1_1,pre1_2,pre1_3:std_logic_vector(7 downto 0);
signal pre2_1,pre2_2,pre2_3:std_logic_vector(7 downto 0);
-----
begin
to_end<='0';
-----
delay_w:process(reset,clk)
begin
    if reset='0'then
        w_d1<=(others=>'0');
        w_d2<=(others=>'0');
        w_d3<=(others=>'0');
        w_d4<=(others=>'0');

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        w_d5<=(others=>'0');
        w_d6<=(others=>'0');
    elsif clk'event and clk='1'then
        w_d1<=w;
        w_d2<=w_d1;
        w_d3<=w_d2;
        w_d4<=w_d3;
        w_d5<=w_d4;
        w_d6<=w_d5;
    end if;
end process delay_w;
-----
delay_c:process(reset,clk)
begin
    if reset='0' then
        c_d1<='0';
        c_d2<='0';
        c_d3<='0';
    elsif clk'event and clk='1' then
        c_d1<=c_acc3;
        c_d2<=c_d1;
        c_d3<=c_d2;
    end if;
end process delay_c;
-----
acc0_unit:adder8
port map( in1=>w(7 downto 0),in2=>acc0_out,
        cin=>to_end,res_out=>acc0_in,cout=>c_acc0);
acc1_unit:adder8
port map( in1=>w_d1(15 downto 8),in2=>acc1_out,
        cin=>cl_acc1,res_out=>acc1_in,cout=>c_acc1);
acc2_unit:adder8
port map( in1=>w_d2(23 downto 16),in2=>acc2_out,
        cin=>cl_acc1,res_out=>acc2_in,cout=>c_acc2);
acc3_unit:adder8
port map( in1=>w_d1(31 downto 24),in2=>acc3_out,
        cin=>cl_acc2,res_out=>acc3_in,cout=>c_acc3);
-----
reg_acc0:process(reset,clk)
begin
    if reset ='0'then
        acc0_out<=(others=>'0');
        cl_acc0<='0';
    elsif clk'event and clk='1'then
        acc0_out<=acc0_in;
        cl_acc0<=c_acc0;
    end if;
end process reg_acc0;
-----
reg_acc1:process(reset,clk)
begin
    if reset ='0'then
        acc1_out<=(others=>'0');
        cl_acc1<='0';
    elsif clk'event and clk='1'then
        acc1_out<=acc1_in;
        cl_acc1<=c_acc1;
    end if;
end process reg_acc1;
-----
reg_acc2:process(reset,clk)
begin

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        if reset='0'then
            acc2_out<=(others=>'0');
            cl_acc2<='0';
        elsif clk'event and clk='1'then
            acc2_out<=acc2_in;
            cl_acc2<=c_acc2;
        end if;
    end process reg_acc2;
-----
reg_acc3:process(reset,clk)
begin
    if reset='0'then
        acc3_out<=(others=>'0');
        cl_acc3<='0';
    elsif clk'event and clk='1'then
        acc3_out<=acc3_in;
        cl_acc3<=c_acc3;
    end if;
end process reg_acc3;
-----
accp0_unit:adder8
port map( in1=>w_d3(7 downto 0),in2=>accp0_out,
          cin=>to_end,res_out=>accp0_in,cout=>c_accp0);
accp1_unit:adder8
port map( in1=>w_d4(15 downto 8),in2=>accp1_out,
          cin=>cl_accp0,res_out=>accp1_in,cout=>c_accp1);
accp2_unit:adder8
port map( in1=>w_d5(23 downto 16),in2=>accp2_out,
          cin=>cl_accp1,res_out=>accp2_in,cout=>c_accp2);
accp3_unit:adder8
port map( in1=>w_d6(31 downto 24),in2=>accp3_out,
          cin=>cl_accp2,res_out=>accp3_in,cout=>c_accp3);
-----
reg_accp0:process(reset,clk)
begin
    if reset='0'then
        accp0_out<=(others=>'0');
        cl_accp0<='0';
    elsif clk'event and clk='1'then
        accp0_out<=accp0_in;
        cl_accp0<=c_accp0;
    end if;
end process reg_accp0;
-----
reg_accp1:process(reset,clk)
begin
    if reset='0'then
        accp1_out<=(others=>'0');
        cl_accp1<='0';
    elsif clk'event and clk='1'then
        accp1_out<=accp1_in;
        cl_accp1<=c_accp1;
    end if;
end process reg_accp1;
-----
reg_accp2:process(reset,clk)
begin
    if reset='0'then
        accp2_out<=(others=>'0');
        cl_accp2<='0';
    elsif clk'event and clk='1'then
        accp2_out<=accp2_in;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        cl_accp2<=c_accp2;
    end if;
end process reg_accp2;
-----
reg_accp3:process(reset,clk)
begin
    if reset='0'then
        accp3_out<=(others=>'0');
        cl_accp3<='0';
    elsif clk'event and clk='1'then
        accp3_out<=accp3_in;
        cl_accp3<=c_accp3;
    end if;
end process reg_accp3;
-----

```

```

reg_pre0_1:process(reset,clk)
begin
    if reset='0'then
        pre0_1<=(others=>'0');
    elsif clk'event and clk='1'then
        pre0_1<=accp0_out;
    end if;
end process reg_pre0_1;
-----

```

```

reg_pre0_2:process(reset,clk)
begin
    if reset='0'then
        pre0_2<=(others=>'0');
    elsif clk'event and clk='1'then
        pre0_2<=pre0_1;
    end if;
end process reg_pre0_2;
-----

```

```

reg_pre0_3:process(reset,clk)
begin
    if reset='0'then
        pre0_3<=(others=>'0');
    elsif clk'event and clk='1'then
        pre0_3<=pre0_2;
    end if;
end process reg_pre0_3;
-----

```

```

reg_pre1_1:process(reset,clk)
begin
    if reset='0'then
        pre1_1<=(others=>'0');
    elsif clk'event and clk='1'then
        pre1_1<=accp1_out;
    end if;
end process reg_pre1_1;
-----

```

```

reg_pre1_2:process(reset,clk)
begin
    if reset='0'then
        pre1_2<=(others=>'0');
    elsif clk'event and clk='1'then
        pre1_2<=pre1_1;
    end if;
end process reg_pre1_2;
-----

```

```

reg_pre2_1:process(reset,clk)
begin

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        if reset='0'then
            pre2_1<=(others=>'0');
        elsif clk'event and clk='1'then
            pre2_1<=accp2_out;
        end if;
    end process reg_pre2_1;
-----
phase_out<=accp3_out & pre2_1 & pre1_2 & pre0_3;
end rtl;

```

11.3 Module MUL10MODULE

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity MUL10MODULE is
    Port ( resetn : in std_logic;
          clock  : in std_logic;
          x      : in std_logic_vector(9 downto 0);
          y      : in std_logic_vector(9 downto 0);
          p      : out std_logic_vector(19 downto 0));
end MUL10MODULE;

architecture rtl of MUL10MODULE is
    signal    m : unsigned(19 downto 0);
begin
    process(resetn, clock)
    begin
        if resetn='0' then
            m<=(others=>'0');
        elsif clock'event and clock='1' then
            m<=unsigned(x)*unsigned(y);
        end if;
    end process;

    p<= std_logic_vector(m);
end rtl;

```

11.4 Module A1

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity A1 is
    Port ( resetn : in std_logic;
          clock  : in std_logic;
          x      : in std_logic_vector(19 downto 0);
          y      : out std_logic_vector(28 downto 0)
    );

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

end A1;

architecture rtl of A1 is

    signal    m        :unsigned(28 downto 0);
    signal    k1       :unsigned(27 downto 0);
    signal    k2       :unsigned(26 downto 0);
    signal    k3       :unsigned(22 downto 0);

begin

    k1 <=unsigned(x) & "00000000";
    k2 <=unsigned(x) & "00000000";
    k3 <=unsigned(x) & "000";

process(resetn, clock)

begin

    if resetn = '0' then
        m<=(others=>'0');
    elsif clock'event and clock='1' then
        --m<=('0' and k1 )+k2+k3;
        m<=('0' & k1 )+k2-k3;
    end if;

end process;

y<= std_logic_vector(m);

end rtl;

```

11.5 Module A2

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity A2 is
    Port ( resetn : in std_logic;
          clock   : in std_logic;
          x       : in std_logic_vector(19 downto 0);
          y       : out std_logic_vector(5 downto 0));
end A2;

```

architecture rtl of A2 is

```

    signal    m        :unsigned(5 downto 0);
    signal    k1       :unsigned(4 downto 0);
    signal    k2       :unsigned(3 downto 0);
    signal    k3       :unsigned(1 downto 0);

begin

    k1 <=unsigned(x(19 downto 15));
    k2 <=unsigned(x(19 downto 16));
    k3 <=unsigned(x(19 downto 18));

process(resetn, clock)

begin

```

```

    if resetn = '0' then

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        m<=(others=>'0');
        elsif clock'event and clock='1' then
            --m<=('0' and k1 )+k2+k3;
            m<=('0' & k1 )+k2-k3;

        end if;

end process;

y<= std_logic_vector(m);

end rtl;

```

11.6 Module add8bit

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity add8bit is
    Port ( din1      : in std_logic_vector(7 downto 0);
          din2      : in std_logic_vector(7 downto 0);
          cin       : in std_logic;
          cout      : out std_logic;
          res_out   : out std_logic_vector(7 downto 0)
        );
end add8bit;

architecture rtl of add8bit is
    -----
    ---signal in carrygenerator---
    -----

    signal p      : std_logic_vector(7 downto 0);
    signal g      : std_logic_vector(7 downto 0);
    signal co     : std_logic_vector(7 downto 0);
    signal c_out  : std_logic;
    signal cry_p1 : std_logic;
    signal cry_g1 : std_logic;
    signal cry_p2 : std_logic;
    signal cry_g2 : std_logic;
    --signal res_out1 : std_logic_vector(3 downto 0);
    --signal res_out2 : std_logic_vector(3 downto 0);

begin
    -----
    -- STAGE ADDER --
    -----

    stage1 : process (din1, din2, cin, p, g)

begin
    -----
        p(0)<=din1(0)or din2(0);
        p(1)<=din1(1)or din2(1);
        p(2)<=din1(2)or din2(2);
        p(3)<=din1(3)or din2(3);
        p(4)<=din1(4)or din2(4);
        p(5)<=din1(5)or din2(5);
        p(6)<=din1(6)or din2(6);
        p(7)<=din1(7)or din2(7);
    -----

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

g(0) <= din1(0) and din2(0);
g(1) <= din1(1) and din2(1);
g(2) <= din1(2) and din2(2);
g(3) <= din1(3) and din2(3);
g(4) <= din1(4) and din2(4);
g(5) <= din1(5) and din2(5);
g(6) <= din1(6) and din2(6);
g(7) <= din1(7) and din2(7);
-----
co(0) <= cin;
co(1) <= g(0) or (cin and p(0));
co(2) <= g(1) or (p(1) and g(0)) or (p(1) and p(0) and cin);
co(3) <= g(2) or (p(2) and g(1)) or (p(2) and p(1) and g(0)) or (p(2) and
p(1) and p(0) and cin);
cry_p1 <= (p(3) and p(2) and p(1) and p(0));
cry_g1 <= g(3) or (p(3) and g(2)) or (p(3) and p(2) and g(1)) or
(p(3) and p(2) and p(1) and g(0));
c_out <= (cry_p1 and cin) or cry_g1;
co(4) <= c_out;
co(5) <= g(4) or (c_out and p(4));
co(6) <= g(5) or (p(5) and g(4)) or (p(5) and p(4) and c_out);
co(7) <= g(6) or (p(6) and g(5)) or (p(6) and p(5) and g(4)) or (p(6) and
p(5) and p(4) and c_out);
-----
end process;

-----
res_out(0) <= din1(0) xor din2(0) xor co(0);
res_out(1) <= din1(1) xor din2(1) xor co(1);
res_out(2) <= din1(2) xor din2(2) xor co(2);
res_out(3) <= din1(3) xor din2(3) xor co(3);
res_out(4) <= din1(4) xor din2(4) xor co(4);
res_out(5) <= din1(5) xor din2(5) xor co(5);
res_out(6) <= din1(6) xor din2(6) xor co(6);
res_out(7) <= din1(7) xor din2(7) xor co(7);
-----
cry_p2 <= (p(7) and p(6) and p(5) and p(4));
cry_g2 <= g(7) or (p(7) and g(6)) or (p(7) and p(6) and g(5)) or (p(7) and
p(6) and p(5) and g(4));
cout <= (cry_p2 and cin) or cry_g2;
-----
end rtl;

```

11.6 Module mux_wave

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY mux_wave IS
    PORT( sel_mux      : in std_logic_vector(2 downto 0);
          dout_osc     : in std_logic_vector(11 downto 0);
          Polynomial_sin,dout_ran : in std_logic_vector(11 downto
0);
          dout        : out std_logic_vector(11 downto 0));
END mux_wave ;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

-----
--
ARCHITECTURE rtl OF mux_wave IS
signal sel :std_logic_vector(2 downto 0);
BEGIN
PROCESS (dout_osc,Polynomial_sin,dout_ran,sel_mux)
BEGIN
sel<=sel_mux;
case sel is
when "000" => dout<=dout_osc;
when "001" => dout<=Polynomial_sin;
when "010" => dout<=Polynomial_sin;
when others => dout<=dout_ran;
end case;

END PROCESS;
END rtl;

```

Program C For mcs-51

```

/*-----*/
// Program : Scan keypad 4x3
// Description : Scan keypad 4x3 for check key push and display on
7-segment
// Filename : l0801.c
// C compiler : RIDE S1 V6.1
/*-----*/
#include<reg51.h> // Header include register of P89C51RD2
#include<string.h>

#define MAX_DIGIT 16
#define KEY_CLEAR 12
#define KEY_ENTER 15
#define KEY_CANCEL 11
#define KEY_MENU_UP 3
#define KEY_MENU_DOWN 7
#define KEY_NONE 0xFF

sbit e = P1^2;
sbit rw = P1^1;
sbit rs = P1^0;
sbit c0 = P2^7; // Bit Column1
sbit c1 = P2^6; // Bit Column2
sbit c2 = P2^5; // Bit Column3
sbit c3 = P2^4; // Bit Column4
sbit r0 = P2^7; // Bit Row1
sbit r1 = P2^6; // Bit Row2
sbit r2 = P2^5; // Bit Row3
sbit r3 = P2^4; // Bit Row4

typedef unsigned char BYTE;
unsigned char lcdbuf[2][MAX_DIGIT];
unsigned char numbuf[MAX_DIGIT];
unsigned char
display[]={'1','2','3','X','4','5','6','X','7','8','9','N','C','0','X',
',','E'};
unsigned char data_buffer[5];
//code unsigned char
display[]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f}; //
Store in Program //memory
/*****/

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่นิยมนำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

/***** Function Delay time *****/
/*****
void delay_db(int time)
{
    do          // do-while loop for delay

    {
        time--;          // Decrease counter
    }while(time>0);    // If time>0 work in block
}
/*****
/
/***** Function Send Command to LCD *****/
void lcd_command(unsigned char com)
{
    rs = 0;
    e = 1;
    P0 = com;
    delay_db(30);
    e = 0;
    delay_db(30);
}
/***** Function Send Data to LCD *****/
void lcd_text(unsigned char text)
{
    rs = 1;
    e = 1;
    P0 = text;
    delay_db(30);
    e = 0;
    delay_db(30);
}
//void lcd_clear()
//{
//    lcd_command(0x01);
//}
/***** Function Set initial parameter LCD *****/
void lcd_init()
{
    delay_db(30);
    lcd_command(0x38);
    lcd_command(0x0c);
    lcd_command(0x06);
}
void display_lcd(BYTE x,char*str)
{
    int i;
    lcd_command(x);
    for(i=0;i<MAX_DIGIT;i++)
        lcd_text(str[i]);
}
void serial_data (unsigned long num)
{
    unsigned long buf;
    data_buffer[3]=num / 0x1000000;
    buf=num % 0x1000000;
    data_buffer[2]=buf / 0x10000;
    buf= buf % 0x10000;
    data_buffer[1]=buf / 0x100;
    data_buffer[0]=buf % 0x100;
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

void start()
{
    int i;
    // PCON=0x00;
    TMOD=0x21;
    TH1=0xfd;
    TL1=0xfd;
    SCOM=0x42;
    TF1=0;
    TR1=1;
    TI = 0;
    for(i=0;i<=4;i++)
    {
        SBUF = data_buffer[i];
        while (~TI);
        TI = 0;
    }
}

unsigned char scankey(void) // Return value key
{
    unsigned char ret = 0xFF; // Initial value ret = 0xFF
    unsigned char col=0x01;
    const unsigned int delay=20;
    int i;
    P2=0xFF;
    for (i=0;i<4;i++,col<<=1){
        P2=~(col&0x0F);
        delay_db(delay);
        if(!r0)ret=i+12;else
        if(!r1)ret=i+8;else
        if(!r2)ret=i+4;else
        if(!r3)ret=i;
        if(ret!=0xFF)break;
    }
    if(ret!=0xFF)while((~(P2))&0xF0)delay_db(delay);
    return ret;
}

void addnum(char*buf, char c){
    unsigned char i;
    bit eof=0;
    for(i=0;i<MAX_DIGIT&&!eof;i++){
        if((buf[i]==0)){
            eof=1;
            buf[i]=c;
            i++;
        }
        if(eof)buf[i]=0;
    }
}

long strtolong(char*txtbuf){
    unsigned char i=0;
    long num=0;
    while((i<16)&&(txtbuf[i]>='0')&&(txtbuf[i]<='9')){
        num=(num*10)+(txtbuf[i]-'0');
        i++;
    }
    return num;
}

void longtostr(char*buf,long num){
    unsigned char i=0;
    unsigned char n=0;
    unsigned char str[10];

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

while(num>0){
    str[n]=(num%10)+'0';
    num/=10;
    n++;
}
buf[n]=0;
for(i=0;i<n;i++){
    buf[i]=str[n-i-1];
}
buf[i]=0;
}
void datatohex(char*buf){
    char i;
    unsigned char asc_temp[]="0123456789ABCDEF";
    for(i=4;i>=0;i--){
        buf[9-((i*2)+1)]=asc_temp[data_buffer[i]/0x10];
        buf[9-(i*2)]=asc_temp[data_buffer[i]%0x10];
    }
    buf[10]='\0';
}
void display_refresh(){
    display_lcd(0x80,lcdbuf[0]);
    lcd_command(0x80);
    display_lcd(0xc0,lcdbuf[1]);
    lcd_command(0x80);
}
void display_settext(unsigned char line,unsigned char *txtbuf){
    unsigned char i;
    bit eof=0;
    for (i=0;i<MAX_DIGIT;i++){
        if (!eof){
            eof=(txtbuf[i]==0);
            if (eof)
                lcdbuf[line][i]=' ';
            else
                lcdbuf[line][i]=txtbuf[i];
        }else
            lcdbuf[line][i]=' ';
    }
}
void main(void)
{
    unsigned char key = 0; // For keep key value
    unsigned long fcw;
    unsigned long fout;
    code unsigned char*menustr[7][3]={{ "LOOKUP
TABLE\0", "SINUSIDOM\0", "SINUSIDOM(Hz)\0"},
                                        {"LOOKUP
TABLE\0", "SQUARE\0", "SQUARE(Hz)\0"},
                                        {"LOOKUP
TABLE\0", "TRIANGULAR\0", "TRIANGULAR(Hz)\0"},
                                        {"LOOKUP
TABLE\0", "RAMP\0", "RAMP(Hz)\0"},
                                        {"POLYNOMIAL", "SINUSIDOM\0", "SINUSIDOM(Hz)\0"},
                                        {"POLYNOMIAL", "RAMP\0", "RAMP(Hz)\0"},
                                        {"VARIATION
AND", "RANDOMIZATION\0", "RANDOM(Hz)\0"}
};
    signed char menu;
    unsigned char max_menu;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับกรใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

bit menu_press=0;

delay_db(20000);
lcd_init();
// display_settext(0,"HELLO\0");
// display_settext(1,"DIRECT DIGITAL\0");
P0 = 0x00;
P2 = 0xff;           // Initial before scankey
max_menu=6;
while(1)           // Infinite loop
{
// menu=0;
display_settext(0,menustr[menu][0]);
display_settext(1,menustr[menu][1]);
display_refresh();

while ((key = scankey())!=KEY_ENTER)// Input value key
from scan keypad & Test key Enter
{
if(key==KEY_MENU_UP)
{
menu--;
menu_press=1;
};
if(key==KEY_MENU_DOWN)
{
menu++;
menu_press=1;
};
if(menu>max_menu)menu=0;
if(menu<0)menu=max_menu;
if(menu_press)
{
display_settext(0,menustr[menu][0]);
display_settext(1,menustr[menu][1]);
display_refresh();
menu_press=0;
}
}
// data_buffer[4]=(menu==5)?0x0C:menu;

display_settext(0,menustr[menu][2]);
display_settext(1,'\0');
display_refresh();

// if (menu==5){
// data_buffer[4]=0x0C;
// numbuf[0]=0;
// }else
// {
data_buffer[4]=menu;
if (menu==6){
data_buffer[4]=0x0c;
}

while((key = scankey())!=KEY_ENTER)
{
if(key==KEY_CLEAR){
numbuf[0]=0;
display_settext(1,0);
display_refresh();
}else
}
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        if((display[key] >='0') && (display[key] <='9')){
// Scankey 3 Columns
            addnum(numbuf, display[key]);
            display_settext(1, numbuf);
            display_refresh();
        }
        if(key==KEY_CANCEL) break;
    }
}
//
}

if(key==KEY_ENTER){
    fout = strtolong(numbuf);
    fcw = (fout*256.0)/24.0;           // 2^8 / 25
    fcw = fcw*(256.0/100.0);         // 2^8 / 100
    fcw = fcw*(256.0/100.0);         // 2^8 / 100
    fcw = fcw*(256.0/100.0);         // 2^8 / 100

    serial_data(fcw);
    start();
    longtostr(numbuf, fcw);           //Look to Dec
// Code_Control
    datatohex(numbuf);               //Look to
    display_settext(0, "SEND");
    display_settext(1, numbuf);
    display_refresh();
    while ((key = scankey()) != KEY_ENTER);
}
numbuf[0]=0;
}
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้