

การจำลองแบบแบ่งเวลาของระบบไม่เชิงเส้น-ดิสเพอร์ซีฟดีเลย์

DISCRETE TIME DOMAIN SIMULATION OF NON-LINEAR
DISPERSIVE-DELAY SYSTEMS

สุเจตน์ จันทรังษ์
SUJATE JANTARANG

อาจารย์ที่ปรึกษา
รศ.ดร.สิทธิชัย โภไคยอุดม
Assc.Prof.Dr.SITTHICHAI POOKAIYAUDOM

วิทยานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรดุษฎีบัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2534
ISBN 974-8155-68-4

บทคัดย่อ		
Abstract		
สารบัญรูป		
บทที่ 1	บทนำการจำลองแบบแบ่งเวลา	1
	1.1 บทนำการจำลองแบบแบ่งเวลา	1
	1.2 เอกสารอ้างอิงบทที่ 1	4
บทที่ 2	การจำลองระบบไม่เชิงเส้นแบบมีการป้อนกลับ	5
	2.1 บทนำการจำลองระบบไม่เชิงเส้นแบบมีการป้อนกลับ	5
	2.2 ออสซิลเลเตอร์แบบวินบริดจ์	7
	2.3 การทดลองออสซิลเลเตอร์แบบวินบริดจ์	12
	2.4 อัลกอริธึมและการจำลองออสซิลเลเตอร์แบบวินบริดจ์	16
	2.5 เอกสารอ้างอิงบทที่ 2	28
บทที่ 3	อินжекชันล็อกกิ้งของออสซิลเลเตอร์	30
	3.1 บทนำอินжекชันล็อกกิ้งของออสซิลเลเตอร์	30
	3.2 อินжекชันล็อกกิ้งของออสซิลเลเตอร์	31
	3.3 การทดลองอินжекชันล็อกกิ้ง	40
	3.4 อัลกอริธึมและการจำลองอินжекชันล็อกกิ้ง	49
	3.5 เอกสารอ้างอิงบทที่ 3	55
บทที่ 4	การจำลองระบบดีเลย์ไลน์ด้วยวิธีแบ่งเวลา	56
	4.1 บทนำการจำลองระบบดีเลย์ไลน์ด้วยวิธีแบ่งเวลา	56
	4.2 ออสซิลเลเตอร์แบบดีเลย์ไลน์	57
	4.3 การทดลองการข้ามโมดของออสซิลเลเตอร์แบบดีเลย์ไลน์	63
	4.4 อัลกอริธึมและการจำลองการข้ามโมดของออสซิลเลเตอร์	70
	4.5 เอกสารอ้างอิงบทที่ 4	79
บทที่ 5	การจำลอง IDT แบบฟังก์ชันเกอมีความกว้าง	81
	5.1 บทนำการจำลอง IDT แบบฟังก์ชันเกอมีความกว้าง	81
	5.2 การจำลองดีเลย์ไลน์และอินเตอร์คิวิตอลทรานสดิวเซอร์	81
	5.3 อัลกอริธึมการจำลอง IDT	88
	5.3.1 อัลกอริธึมสำหรับการจำลอง IDT ตัวส่ง	88
	5.3.2 อัลกอริธึมสำหรับการจำลอง IDT ตัวรับ	92

	หน้า
5.4 การจำลอง IDT แบบมีความกว้าง	97
5.5 เอกสารอ้างอิงบทที่ 5	102
บทที่ 6 โปรแกรมวิเคราะห์เน็ตเวิร์คด้วยสัญลักษณ์	103
6.1 บทนำโปรแกรมวิเคราะห์เน็ตเวิร์คด้วยสัญลักษณ์	103
6.2 การแทนอุปกรณ์อิเล็กทรอนิกส์แบบสัญลักษณ์ในคอมพิวเตอร์	105
6.3 คณิตศาสตร์ของเน็ตเวิร์ค	107
6.4 อัลกอริธึมของโปรแกรมวิเคราะห์เน็ตเวิร์คด้วยสัญลักษณ์	119
6.5 อุปกรณ์และรูปแบบที่ใช้ในโปรแกรมวิเคราะห์เน็ตเวิร์คด้วยสัญลักษณ์	120
6.6 โปรแกรมและการจัดหน่วยความจำ	125
6.6.1 Reduced incident matrix	127
6.6.2 Current source vector matrix	127
6.6.3 Voltage source vector matrix	127
6.6.4 Branch admittance matrix	128
6.7 การคำนวณเมตริกซ์	128
6.8 ตัวอย่าง	135
6.9 เอกสารอ้างอิงบทที่ 6	160
บทที่ 7 สรุป	161
7.1 สรุป	161
กิตติกรรมประกาศ	
ภาคผนวก	
ก. Simulation for Sinusoidal Oscillators with Random Numbers and Initial Impulses.	
ข. Simulations for the Transient Properties of Multi-mode Delay-line Controlled Oscillators.	
ค. Efficient Algorithm for Simulating Interdigital Transducer Delay-Lines	
ง. การออกแบบระบบเปลี่ยนสัญญาณแอนะล็อกเป็นสัญญาณดิจิทัลที่อัตราสูง	
จ. การแก้วงจรแบบสัญลักษณ์ด้วยโปรแกรม	
ฉ. โปรแกรมการจำลองออสซิลเลเตอร์แบบวินบริคจ์ด้วยสภาวะ	

เริ่มต้นต่างๆ

- ช. โปรแกรมการจำลองอินเจคชั่นล๊อกกิ้ง
- ข. โปรแกรมการจำลองการข้ามโมดของออสซิลเลเตอร์แบบ
ดิจิทัล
- ฅ. โปรแกรมการจำลอง IDT แบบมีความกว้างทั้งตัวรับ
และตัวส่ง
- ฉ. โปรแกรม Editor ของโปรแกรมวิเคราะห์เน็ตเวิร์คด้วย
สัญลักษณ์
- ฎ. โปรแกรมวิเคราะห์เน็ตเวิร์คด้วยสัญลักษณ์



บทคัดย่อ

วิทยานิพนธ์ฉบับนี้เสนออัลกอริธึมบางแบบ สำหรับการจำลองแบบแบ่งเวลาของระบบที่ไม่เชิงเส้น ทั้งที่ประกอบด้วยตัวห่วงเวลาและไม่มีตัวห่วงเวลาแบบคิสเพอร์ซีฟ ทั้งนี้ยังได้เสนออัลกอริธึมสำหรับจำลองตัวห่วงเวลาแบบคิสเพอร์ซีฟที่ง่ายและที่ซับซ้อน โดยที่อัลกอริธึมสามารถจำลองตัวห่วงเวลาได้อย่างรวดเร็วและประหยัดหน่วยความจำ และเทคนิคของการจำลองแบบแบ่งเวลา ยังได้นำไปจำลองปรากฏการณ์อันเจกซ์ชันล็อกกิ้งในกรณีที่มีประสิทธิภาพการเชื่อมต้อมีค่าสูง อันเป็นกรณีหนึ่งที่ไม่สามารถวิเคราะห์หาคำตอบได้โดยง่าย รวมทั้งวิทยานิพนธ์ยังได้ศึกษาคุณสมบัติชั่วขณะของออสซิลเลเตอร์ที่ถูกควบคุมด้วยตัวห่วงเวลา ซึ่งโปรแกรมจำลองวงจรที่ใช้กันอยู่ เช่น โปรแกรม SPICE ไม่สามารถจะนำมาประยุกต์เพื่อใช้ในการจำลองได้ง่าย โดยเฉพาะกรณีของตัวห่วงเวลาที่ไม่เป็นอคมคติ นอกจากนี้ในระหว่างการทำวิทยานิพนธ์ยังได้สร้างโปรแกรมสำหรับแก้ปัญหาของวงจรในรูปของสัญลักษณ์ ซึ่งเห็นว่ามีประโยชน์มากเมื่อใช้งานร่วมกับโปรแกรมที่ทำการวิเคราะห์เน็ตเวิร์คเชิงตัวเลขที่มีในปัจจุบัน

Abstract

This thesis presents some new algorithms in the discrete-time simulations of non-linear systems with and without dispersive delay-lines. Very computationally efficient algorithms in the simulation of simple and complex dispersive delay-lines are also presented. The technique is then used to simulate injection locking system with strong mutual injection locking factor, which cannot be easily solved by the usual analytical methods. The transient properties of delay-line controlled oscillators have also been investigated, which using existing circuit simulator, e.g. SPICE, is not easily applicable, especially in the cases of non-ideal dispersive delay-lines. During the course of the thesis investigation, a new program for solving circuit solutions in symbolic forms has also been implemented, where it has proven to be very useful in obtaining various analytical solution of circuits and electronic systems. It is felt that such a program should complement the usefulnesses of several existing numerical network analysis programs.

สารบัญรูป

	หน้า
รูป 2.1 ไคอะแกรมภาพระบบแบบมีการป้อนกลับ	5
รูป 2.2 วงจร RC เน็ตเวิร์คแบบวินบริดจ์	7
รูป 2.3 วงจรขยายแบบการควบคุมอัตราขยายอัตโนมัติ	8
รูป 2.4 วงจรตรวจจับยอดสัญญาณ	10
รูป 2.5 วงจรหาค่าเฉลี่ยของสัญญาณ	11
รูป 2.6 วงจรที่ใช้ทคลองออสซิลเลเตอร์แบบวินบริดจ์	12
รูป 2.7 ผลการทดลองที่ 1 และผลการจำลองที่ 1	13
รูป 2.8 ผลการทดลองที่ 2 และผลการจำลองที่ 2	14
รูป 2.9 ผลการทดลองที่ 3 และผลการจำลองที่ 3	15
รูป 2.10 ไคอะแกรมสถานะของการคำนวณสำหรับการจำลอง เน็ตเวิร์คแบบวินบริดจ์	19
รูป 2.11 ผลการวัดโวลเตจควบคุม $V_c(t)$ ต่ออัตราขยายของ วงจรขยายและค่าผิดพลาดของอัตราขยายเมื่อทำ Polynomial curve fitting	20
รูป 2.12 ไคอะแกรมสถานะการคำนวณของอินทิเกรเตอร์	23
รูป 3.1 บล็อกแผนภาพของวงจรออสซิลเลเตอร์	31
รูป 3.2 แผนภาพเฟเซอร์ของสัญญาณ E_a, E_b และ E_c	32
รูป 3.3 ไคอะแกรมรูปภาพการเกิดปรากฏการณ์ล็อกแบบเชื่อมต่อ ร่วมของออสซิลเลเตอร์ 2 ตัว	36
รูป 3.4 วงจรใช้สำหรับการทดลองเรื่องอินเจกชันล็อกกิ้งและ การล็อกแบบเชื่อมต่อร่วมของออสซิลเลเตอร์	40
รูป 3.5 วงจร Generalized Immittance converter (GIC)	41
รูป 3.6 กราฟแสดงความสัมพันธ์ระหว่างอัตราขยายของวงจรขยายกับ โวลเตจควบคุมจากการทดลองและทำ Polynomial curve fitting	43
รูป 3.7 ผลการทดลองและการจำลองเรื่องอินเจกชันล็อกกิ้ง	45
รูป 3.8 ผลการทดลองและการจำลองเรื่องปรากฏการณ์เชื่อมต่อร่วมใน ออสซิลเลเตอร์	47
รูป 3.9 ความถี่ของสัญญาณจากออสซิลเลเตอร์ขณะเกิดการล็อก	48
รูป 3.10 ไคอะแกรมสถานะการคำนวณแบบเวียนเกิดของวงจรจูนเดี่ยว	50
รูป 4.1 ฟังก์ชันคอสกับไซน์	57

	หน้า
รูป 4.2 สนามไฟฟ้าที่อิเล็กทรอนิกส์	58
รูป 4.3 โคอะแกรมภาพออสซิลเลเตอร์แบบดีเลย์ไลน์	61
รูป 4.4 โคอะแกรมแสดงออสซิลเลเตอร์แบบดีเลย์ไลน์ที่มีการควบคุมการข้ามโมดจากภายนอก	63
รูป 4.5 วงจรการทดลองการข้ามโมดของออสซิลเลเตอร์แบบควบคุมความถี่ด้วยดีเลย์ไลน์	65
รูป 4.6 ผลการทดลองและการจำลองการข้ามโมดของออสซิลเลเตอร์แบบควบคุมความถี่ด้วยดีเลย์ไลน์	69
รูป 4.7 การแบ่งจุดในดีเลย์ไลน์	72
รูป 4.8 โคอะแกรมภาพของดีเลย์ไลน์ออสซิลเลเตอร์ซึ่งมี 1 ฟิงเกอร์ตัวส่งและ 3 ฟิงเกอร์ตัวรับ	74
รูป 4.9 ผลการจำลองหาเวลาการข้ามโมดของออสซิลเลเตอร์กรณีที่มี IDT ตัวรับหลายตัว	77
รูป 4.10 ผลการจำลองหาเวลาการข้ามโมดของออสซิลเลเตอร์จากโมด 16 ไปยังโมด 20 ที่อัตราขยายวนรอบค่าต่างๆกัน	78
รูป 5.1 รูป IDT 1 ส่วน	82
รูป 5.2 IDT ตัวรับช่วงที่ n	86
รูป 5.3 ผลการจำลองที่ 1	97
รูป 5.4 ผลการจำลองที่ 2	98
รูป 5.5 ผลการจำลองที่ 3	99
รูป 5.6 ผลการจำลองที่ 4	100
รูป 5.7 ผลการจำลองที่ 5	101
รูป 6.1 แหล่งจ่ายโวลเตจเปลี่ยนเป็นแหล่งจ่ายกระแส	109
รูป 6.2 การย้ายแหล่งจ่ายโวลเตจ	110
รูป 6.3 Current controlled voltage source (CCVS)	111
รูป 6.4 Voltage controlled current source (VCCS)	111
รูป 6.5 Current controlled current source (CCCS)	112
รูป 6.6 Voltage controlled voltage source (VCVS)	112
รูป 6.7 โฟล์วชาร์ทของโปรแกรม SNAP ส่วน Editor	113
รูป 6.8 โฟล์วชาร์ทของโปรแกรม SNAP ส่วนวิเคราะห์วงจร	116
รูป 6.9 โมเดลของทรานซิสเตอร์	123
รูป 6.10 โมเดลของ MOS ทรานซิสเตอร์	124

	หน้า
รูป 6.11 โมเดลของ FET	124
รูป 6.12 รูปแบบ Record form1	125
รูป 6.13 รูปแบบ Record form2	128
รูป 6.14 วงจรแบ่งโวลเตจ	135
รูป 6.15 วงจรแหล่งจ่ายกระแสและตัวต้านทาน	137
รูป 6.16 วงจร Tank tuned	139
รูป 6.17 วงจรที่ประกอบด้วยแหล่งจ่ายโวลเตจควบคุมด้วยโวลเตจ	141
รูป 6.18 วงจรที่ประกอบด้วยแหล่งจ่ายกระแสควบคุมด้วยโวลเตจ	143
รูป 6.19 วงจรที่ประกอบด้วยแหล่งจ่ายกระแสควบคุมด้วยกระแส	145
รูป 6.20 วงจรที่ประกอบด้วยแหล่งจ่ายโวลเตจควบคุมด้วยกระแส	147
รูป 6.21a วงจรขยายแบบกลับเฟสของสัญญาณ	149
รูป 6.21b วงจรสมมูลย์ของวงจรขยายแบบกลับเฟสของสัญญาณ	149
รูป 6.22a วงจรขยายแบบไม่กลับเฟสของสัญญาณ	151
รูป 6.22b วงจรสมมูลย์ของวงจรขยายแบบไม่กลับเฟสของสัญญาณ	151
รูป 6.23a วงจร GIC ที่ต่ออนุกรมกับตัวต้านทาน	153
รูป 6.23b วงจรเสมือนของวงจร GIC ที่ต่ออนุกรมกับตัวต้านทาน	154

บทที่ 1

บทนำการจำลองแบบแบ่งเวลา

(Introduction to discrete time domain simulation)

1.1 บทนำการจำลองแบบแบ่งเวลา

วิทยานิพนธ์ฉบับนี้ได้้นำการศึกษาระบบแบบแบ่งเวลา (Discrete time system) มาจำลอง (Simulate) ระบบป้อนกลับ (Feedback system) ที่มีส่วนไม่เชิงเส้น (Nonlinear) ประกอบอยู่ด้วย ซึ่งวิธีการการจำลองด้วยวิธีแบ่งเวลานี้จะทำงานบนเครื่องไมโครคอมพิวเตอร์ (Microcomputer) ซึ่งเป็นดิจิทัลคอมพิวเตอร์ (Digital computer) แบบ General purpose ที่มีใช้กันอย่างแพร่หลาย ในวิทยานิพนธ์นี้จะแสดงถึงวิธีการแบ่งวงจรออกเป็นส่วยย่อยเพื่อสร้างแบบจำลองของแต่ละส่วยย่อยแล้วนำเอามาจำลองรวมกัน โดยมีอัลกอริธึมที่ทำให้การคำนวณรวดเร็ว การจำลองระบบด้วยวิธีแบ่งเวลานี้จะสามารถศึกษาระบบที่มีความซับซ้อนซึ่งการศึกษาด้วยวิธีการวิเคราะห์เชิงตัวเลข (Numerical analysis) ทำได้ยาก เช่นระบบที่ประกอบด้วยส่วนที่มีความไม่เชิงเส้น ดังในบทที่ 2 ได้ศึกษาออสซิลเลเตอร์ (Oscillator) ที่มีอัตราขยายของวงจรขยายสัญญาณ (Amplifier) เป็นแบบไม่เชิงเส้น โดยผลจากการศึกษาได้แสดงถึงสัญญาณที่ออกจากออสซิลเลเตอร์เปรียบเทียบกับระหว่างผลการทดลองกับผลการจำลองให้เห็นอย่างชัดเจน ตั้งแต่เริ่มต้นของการออสซิลเลต (Oscillate) คือขณะที่ออสซิลเลเตอร์อยู่ในสภาวะหยุดนิ่ง แล้วมีสัญญาณกระตุ้น (Exciting signal) เกิดขึ้นภายใน ได้แก่ สัญญาณอิมพัลส์ (Impulse signal) อันเป็นสัญญาณขนาดใหญ่ที่เกิดขึ้นชั่วขณะหนึ่ง และสัญญาณรบกวนระดับต่ำ (Noise) ซึ่งเป็นสัญญาณขนาดเล็กที่เกิดขึ้นตลอดเวลา โดยสัญญาณกระตุ้นดังกล่าวจะวิ่งวนอยู่ในออสซิลเลเตอร์เพื่อให้เกิดการเริ่มออสซิลเลต

สำหรับบทที่ 3 ได้นำอัลกอริธึม (Algorithm) ของการจำลองแบบแบ่งเวลาที่ใช้ในการจำลองของบทที่ 2 มาทำการศึกษาปรากฏการณ์อินเจกชันล็อกกิง (Injection locking) ในออสซิลเลเตอร์ ที่ถูกป้อนด้วยสัญญาณที่มีความถี่ใกล้เคียงกับความถี่ทำงานเสรี (Free-running frequency) ของออสซิลเลเตอร์ แล้วออสซิลเลเตอร์จะถูกล็อก คือจะผลิตสัญญาณที่มีความถี่เดียวกับสัญญาณที่ป้อนเข้ามา ซึ่งการศึกษานี้ได้วิเคราะห์หาค่าความถี่สูงสุดและค่าสุดของสัญญาณที่ป้อนเข้าไปยังออสซิลเลเตอร์แล้วทำให้เกิดการล็อก ทั้งกรณีที่เป็นการล็อกทางเดียว และการล็อกแบบเชื่อมต่อกันซึ่งมีออสซิลเลเตอร์ตั้งแต่

สองตัวขึ้นไปส่งสัญญาณไปล็อกซึ่งกันและกัน โดยมีตัวประกอบคุณภาพ (Quality factor) ของรีโซเนเตอร์ (Resonator) ที่เป็นตัวควบคุมความถี่ของการออสซิลเลต และมีสัมประสิทธิ์การเชื่อมต่อ (Coupling coefficient) ที่มีค่าต่าง ๆ กัน นอกจากนั้นในการจำลองการล็อกแบบเชื่อมต่อร่วม ยังแสดงความถี่ของสัญญาณที่ออกจากออสซิลเลเตอร์ในขณะที่เกิดการล็อกด้วย

อย่างไรก็ตามการจำลองในบทที่ 2 และบทที่ 3 นั้น เป็นการจำลองระบบแบบที่ไม่มีอุปกรณ์หน่วงเวลา (Delay) ที่จะทำหน้าที่หน่วงเวลาการเคลื่อนที่ของคลื่นหรือสัญญาณในระบบทางอิเล็กทรอนิกส์ ดังนั้นในบทที่ 4 จึงเสนออัลกอริธึมของการจำลองระบบแบบมีอุปกรณ์หน่วงเวลาที่มีประสิทธิภาพ คือไม่ต้องมีการเคลื่อนย้ายข้อมูลไปตามการเคลื่อนที่ของคลื่นในทางปฏิบัติ และไม่ต้องใช้หน่วยความจำมากเกินไปกว่าที่จำเป็น ทั้งนี้เป็นการใช้อัลกอริธึมดังกล่าวทำการจำลองออสซิลเลเตอร์แบบควบคุมความถี่ด้วยดีเลย์ไลน์ ในขณะที่มีการกระโดดของความถี่จากโหมด (Mode) หนึ่งไปยังอีกโหมดหนึ่ง ซึ่งเป็นปรากฏการณ์ที่เกิดขึ้นในชั่วขณะ

แต่อุปกรณ์หน่วงเวลาแบบ Surface acoustic wave delay line (SAW) ที่นำมาใช้ในการศึกษา มีคลื่นเคลื่อนที่ไปบนผิวของสารพิโซอิเล็กทริก รวมทั้งจะมี IDT (Inter digital transducer) ตัวส่งทำหน้าที่เปลี่ยนพลังงานไฟฟ้าเป็นพลังงานคลื่น และ IDT ตัวรับทำหน้าที่เปลี่ยนพลังงานคลื่นเป็นพลังงานไฟฟ้า โดยที่ IDT นี้มีแบนด์วิดท์ (Bandwidth) ขึ้นกับจำนวนอิเล็กโทรด (Electrode) และอัตราส่วนของความกว้างของโลหะที่อิเล็กโทรดกับช่องว่างระหว่างอิเล็กโทรด ดังนั้นบทที่ 5 จึงนำเสนออัลกอริธึมที่ประหยัดหน่วยความจำและเวลาสำหรับการจำลอง IDT ตัวส่งและ IDT ตัวรับที่ไม่เป็นอุดมคติคือคลื่นจะมีการสะท้อนกลับที่รอยต่อของส่วนที่เป็นโลหะกับช่องว่าง แล้วพลังงานคลื่นส่วนที่เหลือจะเคลื่อนที่ต่อไป โดยทำการศึกษาผลของอัตราส่วนระหว่างความกว้างของโลหะกับช่องว่างระหว่างอิเล็กโทรดที่มีต่อแบนด์วิดท์ของ IDT และศึกษาตัวหน่วงเวลาแบบดิสเพอร์ซีฟ (Dispersive delay line) ซึ่งเป็นตัวหน่วงเวลาที่มีระยะห่างระหว่างอิเล็กโทรดแต่ละคู่ใน IDT ไม่เท่ากัน นอกจากนี้อัลกอริธึมที่เสนอยังสามารถจำลองกรณีที่ความเร็วของคลื่นเปลี่ยนไปซึ่งเกิดจากคลื่นเคลื่อนที่จากบริเวณของตัวกลางหนึ่งไปยังอีกตัวกลางหนึ่งได้ด้วย

อนึ่ง ระหว่างการทำวิทยานิพนธ์ได้พบปัญหาในการหาทรานสเฟอร์ฟังก์ชัน (Transfer function) ของระบบที่ต้องการศึกษา โดยที่บางครั้งทรานสเฟอร์ฟังก์ชันที่หาได้ไม่ถูกต้อง จึงได้พัฒนาโปรแกรมวิเคราะห์วงจรในรูปของสัญลักษณ์ (Symbolic Network Analysis Program, SNAP) ที่

สามารถวิเคราะห์วงจรใน Frequency domain โดยกำหนดให้อุปกรณ์ตัวใดๆ ติดอยู่ในรูปของสัญลักษณ์ และกำหนดค่าเป็นตัวเลขให้กับอุปกรณ์ตัวอื่นได้ ซึ่งในบทที่ 6 ได้แสดงอัลกอริธึมของโปรแกรม SNAP และการจัดการหน่วยความจำ เพื่อเก็บสัญลักษณ์ รวมทั้งการสร้างสมการศูนย์รวม (Node equation) ของวงจร โดยใช้กฎข้อที่ 2 ของเคอร์ชอฟฟ์ (Kirchoff's second law) ซึ่งจะต้องเปลี่ยนแหล่งจ่ายโวลเตจใน Branch ให้เป็นแหล่งจ่ายกระแสก่อนทำการวิเคราะห์

สำหรับการวิเคราะห์วงจรด้วยโปรแกรม SNAP จะให้ผลลัพธ์ซึ่งแสดงสมการโวลเตจที่แต่ละศูนย์รวม (Node) ของวงจรในรูปของสัญลักษณ์ โดยเก็บไว้ในแฟ้มข้อมูล (File) ซึ่งสัญลักษณ์นี้ได้แก่ ตัวต้านทาน (Resistor), ตัวเก็บประจุ (Capacitor), ตัวเหนี่ยวนำ (Inductor), แหล่งจ่ายโวลเตจ (Voltage source) และ แหล่งจ่ายกระแส (Current source) โดยแหล่งจ่ายทั้งสอง สามารถที่จะเป็นแบบควบคุมด้วยโวลเตจที่ตกคร่อม Branch ใด Branch หนึ่ง หรือควบคุมด้วยกระแสที่ไหลผ่าน Branch ใดก็ได้ สำหรับวงจรที่ประกอบด้วยตัวเก็บประจุหรือตัวเหนี่ยวนำ จะได้คำตอบจากการวิเคราะห์ด้วยโปรแกรม SNAP โดยมีตัวแปรความถี่เชิงซ้อน (Complex frequency) คือ S ปรากฏอยู่ ซึ่งสามารถหาผลตอบสนองต่อความถี่ของวงจรได้ด้วยการแทนค่า S ในแฟ้มข้อมูลของผลลัพธ์ นอกจากนั้นการวิเคราะห์วงจรในรูปของสัญลักษณ์ จะช่วยให้สามารถทราบความสัมพันธ์ระหว่างอุปกรณ์บางตัวที่มีต่อคุณลักษณะ (Characteristic) ของวงจร เช่น อุปกรณ์ตัวใดมีผลต่อค่าของ Pole หรือ Zero ซึ่งการวิเคราะห์ด้วยโปรแกรมที่เป็นเชิงตัวเลข เช่น SPICE ไม่สามารถบอกได้ และในตอนท้ายของบทนี้ยังแสดงตัวอย่างของการใช้โปรแกรม SNAP วิเคราะห์วงจรที่ประกอบด้วยอุปกรณ์ทางอิเล็กทรอนิกส์แบบต่างๆด้วย

1.2 เอกสารอ้างอิงบทที่ 1

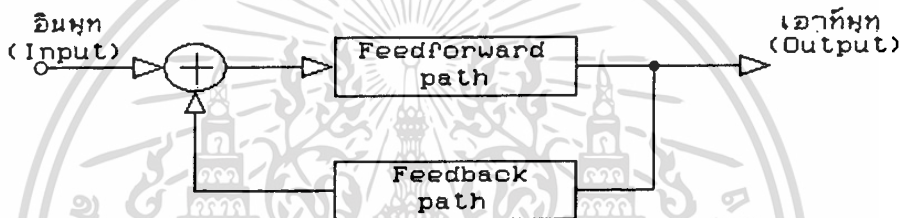
- [1] Jackson A.S. Analog Computation, McGraw-Hill, NewYork (1960).
- [2] Korn G.A., T.M., Electronic Analog and Hybrid Computers, McGraw-Hill, NewYork(1964).
- [3] Scott N.R., Analog and Digital Computer Technology, McGraw-Hill, NewYork(1960).
- [4] Ogata K., Discrete-Time Control Systems, Prentice-Hall International Editions, NewJersey(1987).
- [5] Smith J.M., Mathematical modeling and Digital Simulation for Engineers and Scientists, John Wiley&Sons, NewYork (1977).
- [6] Adler R., "A Study of Locking Phenomena in Oscillators," Proc.IRE, Vol.34, pp.351-357, June 1946.
- [7] Uzunoglu V. and White M.H., "The Synchronous Oscillator : A Synchronization and Tracking Network," IEEE Jour. Solid-state circuit, Vol.SC-20, No.6, pp.1214-1225, Dec. 1985.
- [8] Smith R.W., Gerrard H.M., Collins J.H., Reeder T.M. and Shaw H.J., "Analysis of Interdigital Surface Wave Transducers by use of a Equivalent Circuit Model," IEEE Trans. Microwave Theory Tech., Vol.MTT-17, No.11, pp.856-864, Nov. 1969.
- [9] Pookaiyaudom S. and Kuhanont T., "Automatic Symbolic Polynomial Operations," IEEE Proc., 1977.

บทที่ 2

การจำลองระบบไม่เชิงเส้นแบบมีการป้อนกลับ (Simulation Nonlinear Feedback system)

2.1 บทนำการจำลองระบบไม่เชิงเส้นแบบมีการป้อนกลับ

ในระบบป้อนกลับ จะประกอบด้วยส่วนป้อนไปข้างหน้า (Feedforward path) และส่วนป้อนกลับ (Feedback path) ดังรูป 2.1



รูป 2.1 โค้ดแตรแกรมภาพระบบแบบมีการป้อนกลับ

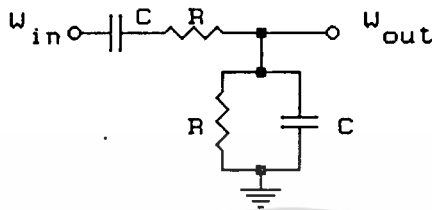
ในการศึกษาระบบทางอิเล็กทรอนิกส์ ส่วนใหญ่จะใช้วิธีศึกษาจากทรานสเฟอร์ฟังก์ชัน (Transfer function) ของระบบที่ต้องการศึกษา แต่ถ้าระบบมีขนาดใหญ่ก็จะหาทรานสเฟอร์ฟังก์ชันได้ยาก รวมทั้งการสร้างสมการแทนส่วนที่ไม่เป็นเชิงเส้น (Nonlinear) ก็สร้างได้ยากหรือไม่ถูกต้อง เพราะต้องใช้วิธีประมาณที่เหมาะสม [1],[2],[3],[4] ดังนั้นวิทยานิพนธ์บทนี้จะเสนอการนำเอาวิธีการจำลองระบบแบบแบ่งเวลา (Discrete time system) มาจำลองระบบป้อนกลับแบบไม่เชิงเส้น โดยทำการศึกษาออสซิลเลเตอร์ซึ่งเป็นระบบป้อนกลับแบบหนึ่งที่ผลิตสัญญาณตามต้องการ ในออสซิลเลเตอร์นั้นสัญญาณเอาต์พุต (Output signal) จะถูกป้อนกลับแบบบวก (Positive feedback) มารวมกับสัญญาณอินพุต (Input signal) ของระบบ ซึ่งเป็นสัญญาณกระตุ้น (Exciting signal) ให้เริ่มเกิดการออสซิลเลต โดยอาจ

เป็นสัญญาณรบกวน (Noise) ระดับต่ำที่มีขนาดเล็กที่เกิดขึ้นในวงจรอยู่ตลอดเวลา หรือเป็นสัญญาณอิมพัลส์ (Impulse signal) ที่มีขนาดใหญ่แต่เกิดในช่วงเวลาสั้น ทั้งนี้ในออสซิลเลเตอร์จะต้องมีส่วนใดส่วนหนึ่งเป็นแบบไม่เชิงเส้น (Nonlinear) เพื่อให้ระบบสามารถเริ่มต้นการออสซิลเลต และรักษาสภาวะการออสซิลเลตในสถานะอยู่ตัว (Steady state)

ออสซิลเลเตอร์ที่ใช้ในการศึกษานี้เป็นแบบควบคุมความถี่การออสซิลเลตด้วยตัวต้านทาน (Resistor) และตัวเก็บประจุ (Capacitor) ที่นำมาต่อกันเป็นเน็ตเวิร์คแบบวินบริดจ์ (Wien bridge network) และในออสซิลเลเตอร์จะมีส่วนป้อนกลับเป็นวงจรขยายแบบควบคุมอัตราขยายอัตโนมัติ (Automatic gain control) ที่มีอัตราขยายเป็นแบบไม่เชิงเส้นโดยขึ้นกับค่าเฉลี่ยและขนาดของสัญญาณที่ออกจากออสซิลเลเตอร์ การศึกษานี้จะทำตั้งแต่เริ่มต้นการออสซิลเลตด้วยสัญญาณกระตุ้นแบบอิมพัลส์และสัญญาณรบกวนระดับต่ำ จนกระทั่งการออสซิลเลตเข้าสู่สภาวะอยู่ตัว โดยที่การจำลองจะแบ่งระบบออกเป็นส่วนย่อยๆ จากนั้นก็สร้างรูปจำลอง (Model) ของแต่ละส่วนย่อย แล้วจะทำการจำลอง โดยให้สัญญาณออกจากส่วนย่อยหนึ่งถูกป้อนไปเป็นสัญญาณเข้าของอีกส่วนย่อยหนึ่ง ตามลำดับของการประกอบกันของส่วนย่อยแต่ละส่วนในระบบ ทำให้การสร้างแบบจำลองระบบทั้งหมดไม่ยุ่งยาก

2.2 ออสซิลเลเตอร์แบบวินบริดจ์

ออสซิลเลเตอร์แบบวินบริดจ์ที่ทำการศึกษานี้ มีส่วนป้อนไปข้างหน้า เป็นเน็ตเวิร์คแบบวินบริดจ์ ดังรูป 2.2



รูป 2.2 วงจร RC เน็ตเวิร์คแบบวินบริดจ์

เน็ตเวิร์คในรูป 2.2 มีทรานส์ฟอร์ฟังก์ชันเป็นแบบเชิงเส้น (Linear) ดังสมการ 2.1 โค้ชทรานส์ฟอร์ฟังก์ชันดังกล่าวได้แสดงคุณสมบัติของเน็ตเวิร์คแบบวินบริดจ์เป็นวงจรผ่านแถบความถี่ (Bandpass filter) แบบหนึ่ง

$$H(S) = \frac{W_{out}}{W_{in}} = \frac{\frac{S}{RC}}{S^2 + \frac{3S}{RC} + \frac{1}{(RC)^2}} \quad \dots (2.1)$$

จากสมการ 2.1 เฟส (Phase) ของสัญญาณเอาต์พุต (W_{out}) เปรียบเทียบกับสัญญาณอินพุต (W_{in}) จะมีค่าเท่ากับ ϕ โดยที่

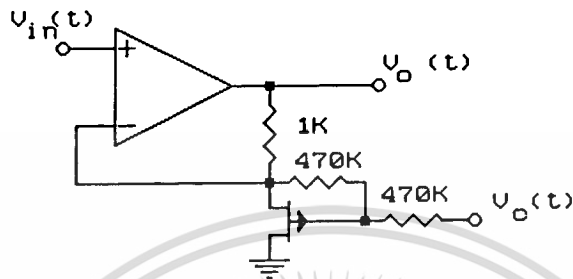
$$\phi = \frac{\pi}{2} - \tan^{-1} \left\{ \frac{3\omega\omega_0}{\omega_0^2 - \omega^2} \right\} \quad \dots (2.2)$$

ให้ ω เป็นความถี่ของสัญญาณอินพุตที่เข้าสู่เน็ตเวิร์คแบบวินบริดจ์

ω_0 เป็นความถี่เรโซแนนท์ (Resonant frequency) ของ เน็ตเวิร์คแบบวินบริดจ์ มีค่าเท่ากับ $1 / (RC)$

ส่วนป้อนกลับของออสซิลเลเตอร์แบบวินบริดจ์ที่ใช้ศึกษา เป็นวงจรขยายแบบควบคุมอัตราขยายอัตโนมัติที่มีคุณสมบัติของอัตราขยายเป็นแบบไม่เป็นเชิงเส้น โดยที่อัตราขยายของวงจรจะขึ้นกับโวลเตจที่มากควบคุม และความไม่เชิงเส้นของส่วนนี้จะมีมากหรือน้อยขึ้นกับชนิดของวงจรขยาย ในที่นี้โวลเตจที่มากควบคุม

อัตราขยายจะนำมาจากสัญญาณที่ออกจากออสซิลเลเตอร์ แล้วผ่านวงจรเปลี่ยนให้เป็นสัญญาณที่เหมาะสมสำหรับการนำไปควบคุมอัตราขยายของวงจรขยาย ดังรูป 2.3



รูป 2.3 วงจรขยายแบบการควบคุมอัตราขยายอัตโนมัติ

จากรูป 2.3 โวลเตจ $V_c(t)$ เป็นโวลเตจสำหรับควบคุมอัตราขยายของวงจรขยาย โดยโวลเตจ $V_c(t)$ จะควบคุมกระแสที่ไหลผ่าน FET ทำให้ความต้านทานระหว่างขาซอร์ส (Source) และขาเดรน (Drain) มีค่าแปรเปลี่ยนตามโวลเตจที่มาควบคุม ดังนั้นอัตราขยายของวงจรขยายจะมีค่าเปลี่ยนไป โดยมีค่าเท่ากับ

$$\frac{V_o}{V_{in}} = 1 + \frac{1}{1000 + R_f}$$

เมื่อ R_f เป็นความต้านทานระหว่างขาซอร์ส และเดรนของ FET ซึ่งขึ้นกับโวลเตจควบคุม $V_c(t)$

วงจรขยายดังรูป 2.3 มีเฟสระหว่างสัญญาณที่อินพุต $V_{in}(t)$ และสัญญาณเอาต์พุต $V_o(t)$ เท่ากับศูนย์องศา ดังนั้นการออสซิลเลตจะเกิดที่ความถี่เรโซแนนท์ของเน็ตเวิร์คแบบวินบรีค ซึ่งมีความถี่ระหว่างสัญญาณเข้า (W_{in}) กับสัญญาณออก (W_{out}) ของเน็ตเวิร์คตามสมการ 2.2 เท่ากับศูนย์องศา ความถี่ของสัญญาณที่ออกจากออสซิลเลเตอร์จึงมีค่า

$$\omega = \frac{1}{Z} = \frac{1}{RC} \quad \dots(2.3)$$

Z เป็นค่าคงตัวเวลา (Time constant) เท่ากับ RC

จากสมการ 2.1 และ 2.3 แสดงให้เห็นว่า ที่จุดเรโซแนนซ์ของ เนทเวิร์คแบบวินบริดจ์จะมีอัตราขยายเท่ากับ $1/3$ ดังนั้นอัตราขยายของวงจร ขยายที่ความถี่เรโซแนนซ์จะต้องมีค่ามากกว่าหรือเท่ากับ 3 เพื่อให้อัตราขยาย วนรอบ (Loop) ในออสซิลเลเตอร์มีค่ามากพอที่จะทำให้เกิดการออสซิลเลต

จากรูป 2.1 ออสซิลเลเตอร์จะเริ่มต้นการออสซิลเลตโดยให้ออสซิลเลเตอร์อยู่ในสถานะหยุดนิ่ง แล้วมีสัญญาณอินพุตที่เป็นสัญญาณกระตุ้นเข้าไปใน วงจร ซึ่งสัญญาณกระตุ้นนี้จะเกิดขึ้นที่ใดในวงจรก็ได้ ในที่นี้ให้สัญญาณกระตุ้น ทั้งหมดเกิดขึ้นที่เดียวเพื่อความสะดวกในการวิเคราะห์วงจร และสัญญาณกระตุ้น นี้จะมีค่ามากหรือน้อยขึ้นกับระบบในความเป็นจริง ดังในบทความ [5], [6] [7] และ [8] สัญญาณกระตุ้นที่จะใช้ในการศึกษาคือ

ก. สัญญาณรบกวนระดับต่ำ (Noise) ซึ่งมีขนาดเล็กและเกิดขึ้น ตลอดเวลาในออสซิลเลเตอร์

ข. สัญญาณอิมพัลส์ (Impulse) มีขนาดใหญ่และเกิดขึ้นชั่วขณะหนึ่ง ได้แก่สัญญาณที่เกิดขึ้นที่ป้อนแรงดันโวลเตจให้กับออสซิลเลเตอร์อย่างทันทีทันใด แล้วบางจุดในวงจรไม่สามารถเปลี่ยนแปลงได้ทันที

สัญญาณกระตุ้นที่เกิดขึ้นจะวิ่งวนในวนรอบ ซึ่งในตอนเริ่มต้นการออสซิลเลตนั้นวงจรขยายต้องมีอัตราขยายมากกว่า 3 เพื่อทำให้สัญญาณกระตุ้นที่วิ่ง วนอยู่มีขนาดสูงขึ้นจนออสซิลเลเตอร์เข้าสู่สถานะอยู่ตัว ที่สถานะอยู่ตัวนี้วงจร ขยายจะมีอัตราขยายเท่ากับ 3 สำหรับในช่วงตั้งแต่เริ่มการออสซิลเลตจน เข้าสู่สถานะอยู่ตัวอาจมีการพุ่งเกิน (Overshoot) ของสัญญาณที่ออกจากออสซิลเลเตอร์เนื่องมาจากวงจรขยายไม่สามารถเปลี่ยนแปลงอัตราขยายได้ทันที ในการควบคุมอัตราขยายของวงจรขยายนั้น จะใช้โวลเตจที่นำมาจากสัญญาณที่ออก จากออสซิลเลเตอร์ไปผ่านวงจรแปลงขนาด (Amplitude converter) ซึ่งมี ได้หลายวิธี แต่ในการศึกษานี้จะทำการศึกษาวงจรแปลงขนาดสัญญาณ 2 แบบ คือ

ก. วงจรแปลงขนาดสัญญาณแบบตรวจจับยอดสัญญาณ (Peak detector) วงจรนี้จะทำการจะตรวจสอบยอดสัญญาณ (Peak) ที่ออกจากออสซิลเลเตอร์ทั้งยอดสัญญาณด้านบวกและยอดสัญญาณด้านลบ จากนั้นจะนำเอาขนาด ของยอดสัญญาณมาควบคุมอัตราขยายของวงจรขยาย โดยอัตราขยายของวงจร

จะคงที่จนกว่าจะตรวจสอบพบข้อผิดพลาดต่อไป

ให้ $V_c(t)$ เป็นสัญญาณที่ออกจากวงจรตรวจสอบข้อผิดพลาด

$A(t)$ เป็นอัตราขยายของวงจรขยายที่เวลา t

t_n^+ เป็นเวลาที่ปรากฏข้อผิดพลาดบวกของไซเคิล (Cycle) ที่ n

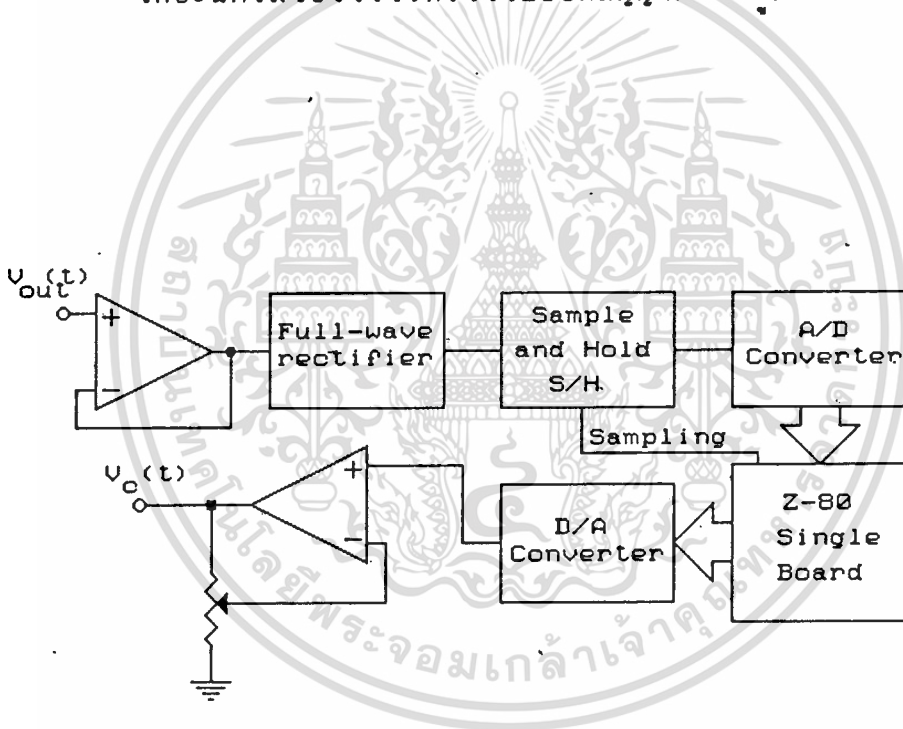
t_n^- เป็นเวลาที่ปรากฏข้อผิดพลาดลบของไซเคิล (Cycle) ที่ n

G เป็นตัวดำเนินการหนึ่ง (Operator) ซึ่งให้ผลลัพธ์เป็นค่าอัตราขยายของวงจรขยายที่ขึ้นกับขนาดของสัญญาณ $V_c(t)$

ดังนั้นจะสามารถแสดงอัตราขยายของวงจรขยายได้ดังสมการ 2.4

$$A(t) = \begin{cases} G[V_c(t_n^+)] & ; t_n^+ < t < t_n^- \\ G[V_c(t_n^-)] & ; t_n^- < t < t_{n+1}^+ \end{cases} \quad \dots (2.4)$$

โคโตะแกรมของวงจรตรวจสอบข้อผิดพลาดมีดังรูป 2.4



รูป 2.4 วงจรตรวจสอบข้อผิดพลาด

จากรูป 2.4 สัญญาณ $V_{out}(t)$ ซึ่งเป็นสัญญาณที่ออกจากออสซิลเลเตอร์ จะถูกส่งผ่านวงจรบัฟเฟอร์ (Buffer) แล้วผ่านวงจรจัดเรียงกระแสแบบเต็มคลื่น (Full-wave rectifier) ไปยังวงจรแปลงสัญญาณแอนะล็อกเป็นข้อมูลดิจิทัล (Analog to digital converter, A/D converter) จากนั้นข้อมูลที่เป็นขนาดของสัญญาณจะถูกอ่านจากวงจร A/D converter เข้าไปยังซิงเกิลบอร์ด Z-80 (Single board Z-80) เพื่อทำการตรวจสอบหาจุดบอดของสัญญาณ เมื่อทราบขนาดบอดของสัญญาณแล้วซิงเกิลบอร์ด Z-80 จะส่งผลการคำนวณหาขนาดของสัญญาณควบคุมอัตราขยายไปยังวงจรแปลงข้อมูลดิจิทัลเป็นสัญญาณแอนะล็อก (Digital to analog converter, D/A converter) เพื่อเปลี่ยนข้อมูลดิจิทัลเป็นสัญญาณแอนะล็อกไปควบคุมอัตราขยายของวงจรขยายต่อไป จะเห็นได้ว่าวิธีการตรวจจับบอดสัญญาณแบบนี้ยากต่อการใช้งานทั่วไป แต่ในที่นี้ใช้เพื่อการศึกษาแบบหนึ่งของวงจรขยายแบบไม่เชิงเส้นในออสซิลเลเตอร์เท่านั้น

ข. วงจรแปลงขนาดสัญญาณแบบหาค่าเฉลี่ยของสัญญาณ (Average detector) เป็นวงจรที่จะอินทิเกรต (Integrate) สัญญาณออกจากออสซิลเลเตอร์ นับตั้งแต่เริ่มต้นการทำงานของวงจรออสซิลเลเตอร์ จนถึงเวลาที่นำเอาสัญญาณนั้นไปควบคุมอัตราขยายของวงจร ซึ่งวงจรหาค่าเฉลี่ยของสัญญาณจะมีดังรูป 2.5



รูป 2.5 วงจรหาค่าเฉลี่ยของสัญญาณ

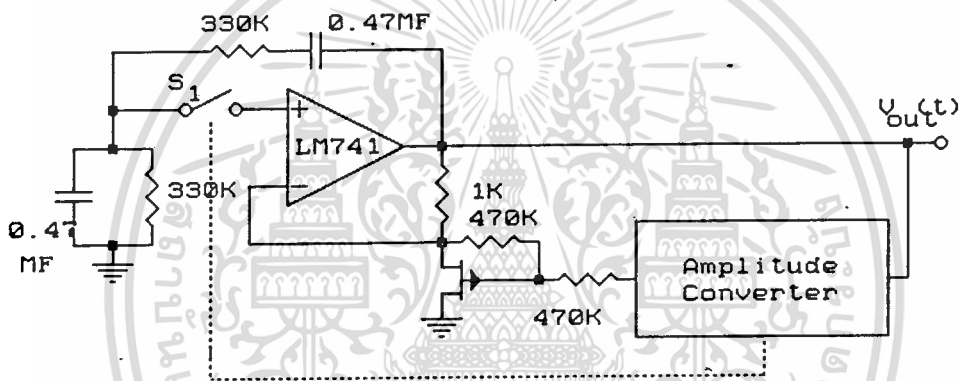
จากรูป 2.5 สัญญาณที่ออกจากออสซิลเลเตอร์ จะผ่านวงจรบัฟเฟอร์ และวงจรจัดเรียงกระแสแบบเต็มคลื่นสัญญาณตามลำดับ ไปยังวงจรอินทิเกรเตอร์ ที่ประกอบด้วยตัวต้านทาน (R) และตัวเก็บประจุ (C) ต่ออนุกรมกันอยู่ สัญญาณที่ออกจากอินทิเกรเตอร์จะถูกนำไปควบคุมอัตราขยายของวงจรขยายในรูป 2.3

2.3 การทดลองออสซิลเลเตอร์แบบวินบริดจ์

มีการทดลอง 3 การทดลองดังนี้

การทดลองที่ 1

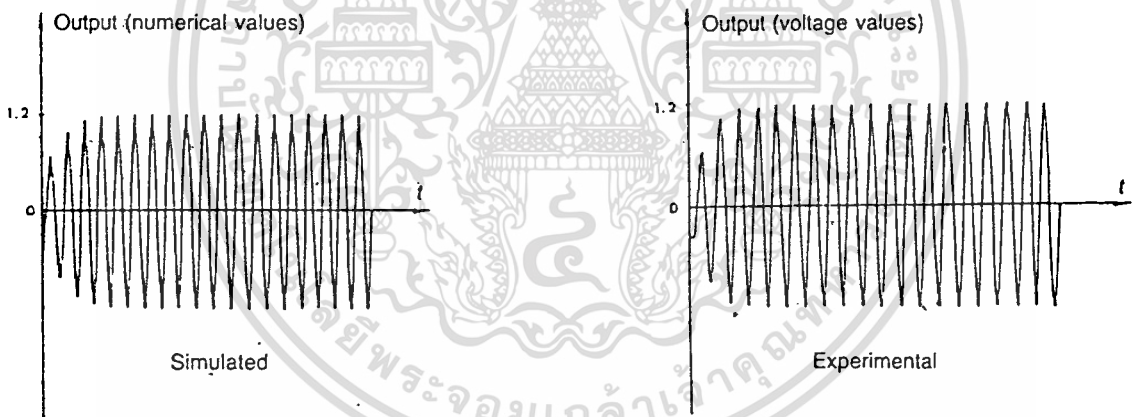
เป็นการทดลองออสซิลเลเตอร์แบบวินบริดจ์ ที่มีการควบคุมอัตราขยาย ด้วยวงจรแปลงขนาดสัญญาณแบบตรวจสอบอคติสัญญาณออกจากออสซิลเลเตอร์ โดยเริ่มต้นการออสซิลเลตด้วยสัญญาณอิมพัลส์ที่เกิดขึ้นในวงจรในขณะที่เริ่มต้นการทำงานของวงจรออสซิลเลเตอร์ วงจรสำหรับทดลองมีดังรูป 2.6



รูป 2.6 วงจรที่ใช้ทดลองออสซิลเลเตอร์แบบวินบริดจ์

จากค่าอุปกรณ์จริงในรูปและสมการ 2.3 จะคำนวณหาความถี่ของการออสซิลเลตได้เท่ากับ 1.0261 เฮิรตซ์ (Hertz) วงจรแปลงขนาดสำหรับการทดลองที่ 1 คือวงจรแปลงขนาดแบบตรวจสอบอคติสัญญาณในรูป 2.4 ซึ่งตอนเริ่มต้นซึ่งเกิลบอร์ด Z-80 ได้ส่งสัญญาณควบคุมให้สวิตช์ (Switch) S_1 เป็ดวงจร (Open circuit) และส่งข้อมูลผ่าน D/A converter เพื่อให้วงจรขยายมีอัตราขยายต่ำสุด จากนั้นซึ่งเกิลบอร์ด Z-80 จะส่งข้อมูลผ่าน D/A

เพื่อควบคุมให้วงจรขยายมีอัตราขยายสูงสุด เสร็จแล้วจะสั่งงานให้สวิทช์ S_1 ปิดวงจร (Close circuit) เป็นการเริ่มต้นการออสซิลเลต ส่วนสัญญาณอิมพัลส์เริ่มต้นจะมาจากโวลเตจที่ตกคล่อมตัวต้านทานและตัวเก็บประจุซึ่งต่อขนานกันอยู่ในรูป 2.6 โวลเตจนี้จะแตกต่างจากโวลเตจที่ขา Non-inverting ของวงจรขยายเชิงปฏิบัติการ (Operational Amplifier) เมื่อสวิทช์ S_1 ปิดวงจรอย่างทันทีทันใดจะทำให้เกิดอิมพัลส์ขึ้นในวงจร การควบคุมอัตราขยายของวงจรขยายอัตโนมัติ ซึ่งเกิลบอร์ด Z-80 จะตรวจสอบสัญญาณ $V_{out}(t)$ ที่ออกจากออสซิลเลเตอร์ผ่านทางส่วน A/D converter ด้วยอัตรา 10,000 ครั้งต่อวินาที เพื่อหาขนาดจุดยอดของสัญญาณ $V_{out}(t)$ ในทุกไซเคิล (Cycle) แล้วนำเอาขนาดของจุดยอดที่วัดได้ไปคำนวณหาอัตราขยายเพื่อส่งออกทาง D/A converter ไปควบคุมอัตราขยายของวงจรขยายสัญญาณ ทำการบันทึกขนาดของสัญญาณ $V_{out}(t)$ นับตั้งแต่เวลาเริ่มต้นที่สวิทช์ S_1 ปิดวงจร จนกระทั่งสัญญาณที่ออกจากออสซิลเลเตอร์อยู่ในสถานะอยู่ตัวโดยใช้เครื่องมือ X-Y recorder ซึ่งผลจากการทดลองมีดังรูป 2.7

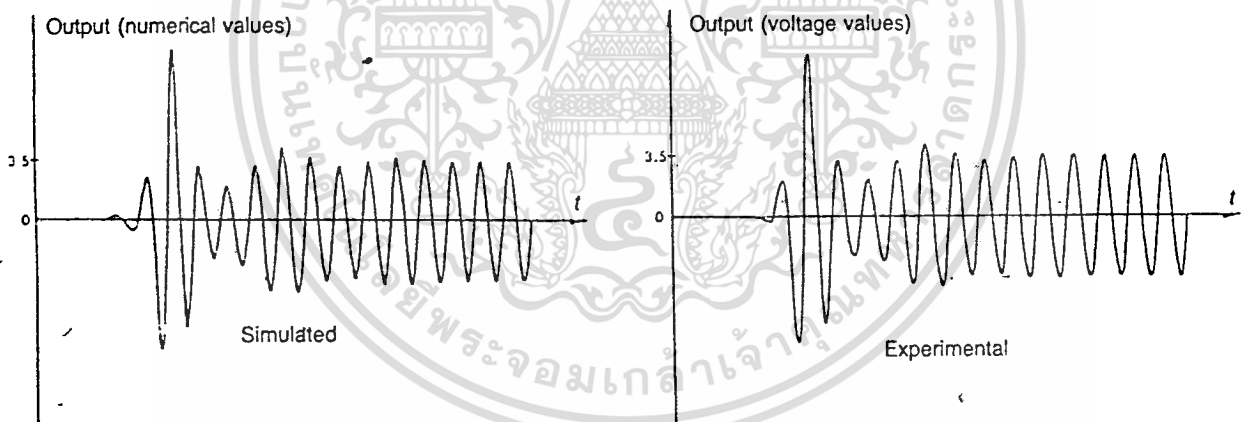


รูป 2.7 ผลการทดลองที่ 1 และผลการจำลองที่ 1

การทดลองที่ 2

เป็นการทดลองออสซิลเลเตอร์แบบวินบริดจ์ ที่มีการควบคุมอัตราขยายของวงจรขยายด้วยวงจรแปลงขนาดสัญญาณแบบหาค่าเฉลี่ยของสัญญาณออกจากออสซิลเลเตอร์ และเริ่มต้นการออสซิลเลตด้วยสัญญาณรบกวนระดับต่ำ

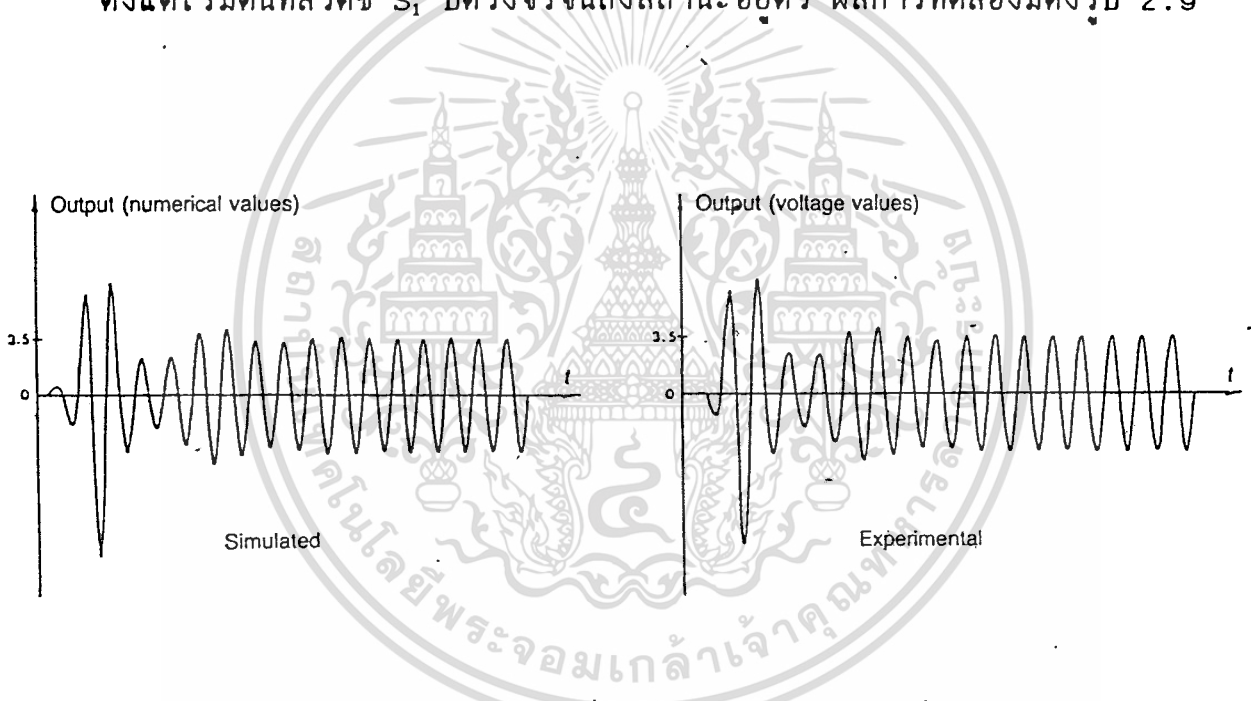
สำหรับการทดลองที่ 2 จากวงจรดังรูป 2.6 การทดลองจะเริ่มจากซิงเกิลบอร์ด Z-80 ส่งสัญญาณควบคุมให้สวิตช์ S_1 ปิดวงจร (Close circuit) และส่งข้อมูลผ่าน D/A converter ให้วงจรขยายมีอัตราขยายต่ำสุดเพื่อไม่ให้เกิดการออสซิลเลตและไม่ให้ระบบเกิดอิมพัลส์ขึ้นเมื่อเริ่มการทดลอง จากนั้นซิงเกิลบอร์ด Z-80 จะเปลี่ยนให้สัญญาณจากวงจรแปลงขนาดในรูป 2.5 ซึ่งมีค่าความต้านทาน (R) เท่ากับ 560 กิโลโอห์มและตัวเก็บประจุ (C) มีขนาด 2.2 ไมโครฟารัด มาควบคุมอัตราขยายของวงจรขยาย ทำการบันทึกขนาดของสัญญาณออกจากออสซิลเลเตอร์ นับตั้งแต่เวลาเริ่มต้นที่สวิตช์ S_1 ปิดวงจร จนกระทั่งสัญญาณออกจากออสซิลเลเตอร์อยู่ในสถานะอยู่ตัวโดยใช้เครื่องมือ X-Y recorder ซึ่งผลจากการทดลองมีดังรูป 2.8



รูป 2.8 ผลการทดลองที่ 2 และผลการจำลองที่ 2

การทดลองที่ 3

การทดลองออสซิลเลเตอร์แบบวินบริดจ์ ที่มีการควบคุมอัตราการขยายของวงจรด้วยวงจรแปลงขนาดสัญญาณแบบหาค่าเฉลี่ยของสัญญาณออกจากออสซิลเลเตอร์ และเริ่มต้นการออสซิลเลตด้วยสัญญาณอิมพัลส์ ซึ่งวงจรสำหรับการทดลองนี้จะเหมือนกับการทดลองที่ 2 โดยการทดลองจะเริ่มต้นจากการให้สวิตช์ S_1 เปิดวงจร (Open circuit) เพื่อหยุดการออสซิลเลต และทำให้มีโวลเตจตกตลอดความต้านทานและตัวเก็บประจุที่ต่อขนานกันอยู่ เมื่อสวิตช์ S_1 ปิดวงจรอย่างทันทีทันใดจะเกิดอิมพัลส์ขึ้นในวงจร เป็นการเริ่มต้นการออสซิลเลต สัญญาณออกจากออสซิลเลเตอร์ส่วนหนึ่งจะถูกป้อนกลับมาเข้าส่วนจัดเรียงกระแสแบบเต็มคลื่น แล้วผ่านวงจรอินทิเกรเตอร์เพื่อเอาสัญญาณที่ได้ไปควบคุมอัตราขยายของวงจรขยาย บันทึกผลการทดลองด้วย X-Y recorder ตั้งแต่เริ่มต้นที่สวิตช์ S_1 ปิดวงจรจนถึงสถานะอยู่ตัว ผลการทดลองมีดังรูป 2.9



รูป 2.9 ผลการทดลองที่ 3 และผลการจำลองที่ 3

2.4 อัลกอริธึมและการจำลองออสซิลเลเตอร์แบบวินบริดจ์

การจำลองระบบป้อนกลับด้วยวิธีแบ่งเวลาบนดิจิทัลคอมพิวเตอร์นั้น จะใช้วิธีการแบ่งระบบที่ต้องการศึกษาออกเป็นส่วยย่อย ซึ่งเป็นวิธีการที่สะดวก สำหรับการจำลองระบบซึ่งมีความซับซ้อน โดยไม่ต้องหาทรานส์เฟอร์ฟังก์ชันรวม ของระบบนั้นทั้งหมด จากนั้นก็สร้างแบบจำลองของการทำงานในแต่ละส่วยเป็น แบบแบ่งเวลา แล้วเขียนโปรแกรมคอมพิวเตอร์เพื่อกำนวณตามแบบจำลองนั้น โดยจะต้องหาคุณสมบัติที่จำเป็นของระบบที่ศึกษาเสียก่อน เช่น ในวงจรขยายที่มี การควบคุมอัตราขยายอัตโนมัติก็ต้องทราบความสัมพันธ์ระหว่างอัตราขยายของ วงจรขยายต่อโวลเตจที่มากควบคุมเสียก่อน

ในบทนี้จะใช้ออสซิลเลเตอร์แบบวินบริดจ์ ซึ่งเป็นระบบป้อนกลับแบบ ไม่เชิงเส้นในการศึกษา โดยจะเสนออัลกอริธึมและการสร้างแบบจำลองแต่ละ ส่วนของออสซิลเลเตอร์ จากนั้นได้ใช้ภาษา TURBO PASCAL สำหรับเขียน โปรแกรมจำลองการทำงานของออสซิลเลเตอร์ การสร้างแบบจำลองของแต่ละ ส่วนในออสซิลเลเตอร์จะมีดังนี้

ก. การสร้างแบบจำลองของเน็ตเวิร์คแบบวินบริดจ์

จากรูปแบบเน็ตเวิร์คแบบวินบริดจ์ในรูป 2.2 และสมการของทราน- สเฟอร์ฟังก์ชันในสมการ 2.1 จะสามารถเขียนสมการนี้ได้ใหม่

$$\begin{aligned} H(S) &= \frac{W_{out}(S)}{W_{in}(S)} \\ &= \frac{S}{Z} \\ &= \frac{S^2 + 3S + 1}{Z^2} \\ &= \frac{1}{Z(A - B)} \left\{ \frac{A}{(S + A)} - \frac{B}{(S + B)} \right\} \quad \dots (2.5) \end{aligned}$$

โดยที่

$$A = \frac{3 + \sqrt{5}}{2Z}$$

$$B = \frac{3 - \sqrt{5}}{2Z}$$

จากเน็ตเวิร์ครูป 2.2 เมื่อต้องการแสดงผลตอบสนองต่อสัญญาณที่

เข้ามายังเนตเวิร์คใน Time domain จะสามารถทำได้โดยการหาคอนโวลูชัน (Convolution) ระหว่างสัญญาณอินพุตกับผลตอบสนองต่ออิมพัลส์ (Impulse response) ของเนตเวิร์ค เมื่อให้สัญญาณอินพุตที่เข้ามายังเนตเวิร์คในช่วงเวลาที่ t น้อยกว่า 0 มีค่าเป็น 0 จะแสดงการคอนโวลูชันได้ดังสมการ 2.6

$$\begin{aligned} W_{out}(t) &= W_{in}(S) * H(S) \\ &= \int_0^t W_{in}(T)h(t-T)dT \end{aligned} \quad \dots(2.6)$$

$W_{in}(t)$ เป็นสัญญาณอินพุตของเนตเวิร์คใน Time domain

$h(t)$ เป็นผลตอบสนองต่ออิมพัลส์ของเนตเวิร์ค โดย $h(t)$ ได้จากการหาอินเวอร์สลาปาสทรานสฟอร์ม (Inverse Laplace Transform) ของ $H(S)$ ซึ่งเป็นทรานสเฟอร์ฟังก์ชันของเนตเวิร์ค

จากสมการ 2.5 จะหาอินเวอร์สลาปาสทรานสฟอร์มของ $H(S)$ ได้ว่า

$$h(t) = \frac{1}{Z(A-B)} (Ae^{-At} - Be^{-Bt}) \quad \dots(2.7)$$

$$\text{ให้ } A' = \frac{A}{Z(A-B)}$$

$$B' = \frac{B}{Z(A-B)}$$

$$h_1(t) = A'e^{-At}$$

$$h_2(t) = B'e^{-Bt}$$

ดังนั้น

$$h(t) = h_1(t) - h_2(t)$$

สมการ 2.7 แสดงการหาคอนโวลูชันแบบเวลาต่อเนื่อง สำหรับระบบแบ่งเวลาจะสามารถเขียนสมการ 2.7 ได้ว่า

$$W_{out}(n\Delta t) = \sum_{k=0}^n W_{in}(k\Delta t)h\{(n-k)\Delta t\}\Delta t \quad \dots(2.8)$$

เมื่อ Δt เป็นคาบเวลาที่ถูกแบ่งสำหรับการคำนวณแต่ละครั้ง

สมการ 2.8 แสดงให้เห็นว่าเมื่อเวลามากขึ้น นั่นคือ $n\Delta t$ มีค่ามากขึ้น จะทำให้เวลาที่ต้องใช้ในการคำนวณเพื่อให้ได้ $W_{out}(n\Delta t)$ จากสมการ 2.8 ก็จะมีมากขึ้นตามลำดับ ถึงแม้ว่าจะมีวิธีการเพิ่มความเร็วในการคำนวณโดยประมาณ

ให้เทอม $h(n\Delta t)$ ในสมการ 2.8 ของเน็ตเวิร์คแบบวินบริดจ์มีค่าเป็นศูนย์ เมื่อ $n\Delta t$ มีค่ามากขึ้น หรือใช้วิธีการประมาณให้อยู่ในรูปของฟังก์ชันอื่น [9] แต่จะทำให้เกิดผิดพลาดขึ้นที่ผลลัพธ์จากการคำนวณเรียกว่า Truncation error ซึ่งมีผลมากในการจำลองระบบป้อนกลับ [10] ซึ่งเมื่อพิจารณาจากสมการ 2.8 จะสามารถเขียนใหม่ได้ดังนี้

$$\begin{aligned} W_{out}(n\Delta t) &= \sum_{k=0}^n W_{in}(k\Delta t)h\{(n-k)\Delta t\}\Delta t \\ &= \sum_{k=0}^n W_{in}(k\Delta t)[h_1\{(n-k)\Delta t\} - h_2\{(n-k)\Delta t\}]\Delta t \\ &= \sum_{k=0}^n W_{in}(k\Delta t)(A'e^{-\Lambda(n-k)\Delta t} - B'e^{-B(n-k)\Delta t})\Delta t \quad \dots(2.9) \end{aligned}$$

$$W_{out}\{(n+1)\Delta t\} = \sum_{k=0}^{n+1} W_{in}(k\Delta t)(A'e^{-\Lambda(n+1-k)\Delta t} - B'e^{-B(n+1-k)\Delta t})\Delta t$$

กำหนดค่าให้

$$W_{out1}(n\Delta t) = \sum_{k=0}^n W_{in}(k\Delta t)(A'e^{-\Lambda(n-k)\Delta t})\Delta t$$

$$W_{out2}(n\Delta t) = \sum_{k=0}^n W_{in}(k\Delta t)(B'e^{-B(n-k)\Delta t})\Delta t$$

ดังนั้นจะเขียนสมการของ $W_{out}\{(n+1)\Delta t\}$ ได้เป็น

$$\begin{aligned} W_{out}\{(n+1)\Delta t\} &= W_{out1}(n\Delta t)e^{-\Lambda\Delta t} - W_{out2}(n\Delta t)e^{-B\Delta t} + \\ &\quad + \Delta t A' W_{in}\{(n+1)\Delta t\} - \Delta t B' W_{in}\{(n+1)\Delta t\} \quad \dots(2.10) \end{aligned}$$

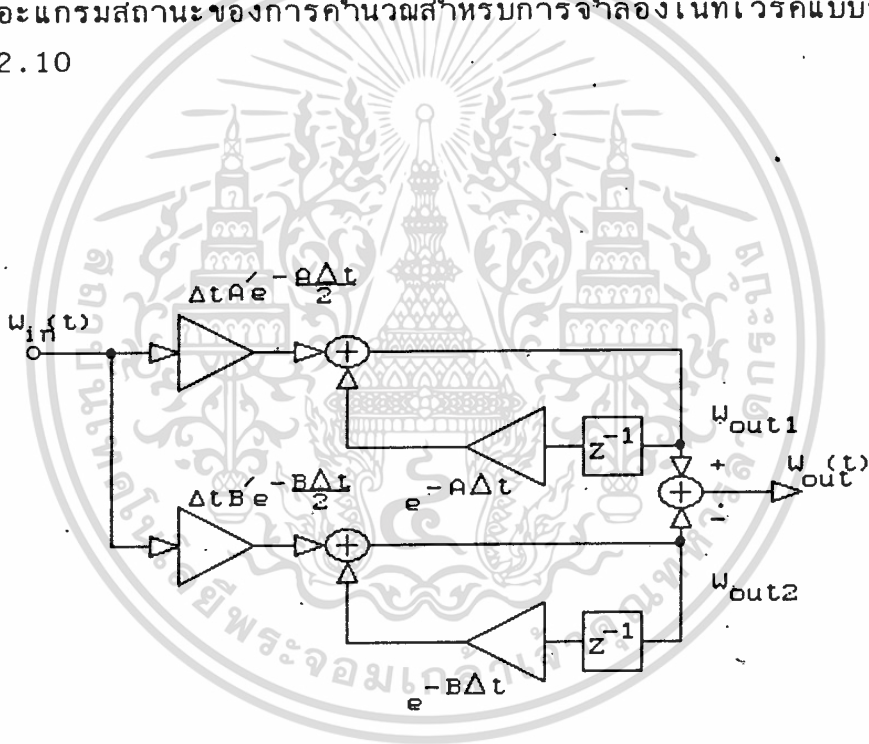
สมการ 2.10 อยู่ในรูปการเวียนเกิด (Recursive formula) คือผลลัพธ์ (W_{out}) ของการคำนวณขั้น (Step) ที่ $n+1$ จะขึ้นกับ W_{out} ที่เป็นผลจากการคำนวณขั้นที่ n และสัญญาณอินพุตในระหว่างเวลา $n\Delta t$ ถึง $(n+1)\Delta t$ ซึ่งทำให้ในแต่ละคาบการจำลอง คือในแต่ละคาบเวลา Δt จะมีจำนวนครั้งของการคำนวณน้อยมากเมื่อเทียบกับการคำนวณในสมการ 2.8

ในการคำนวณตามสมการ 2.10 นั้น เพื่อให้การหาคอนโวลูชันแบบ

แบ่งเวลาได้ค่าผลลัพธ์ใกล้เคียงกับการหาคอนโวลูชันแบบเวลาต่อเนื่อง จึงต้องหาคอนโวลูชันเป็นค่าเฉลี่ยระหว่างวิธี Forward difference integration ซึ่งให้ผลลัพธ์จากการหาคอนโวลูชันมากกว่าคอนโวลูชันแบบเวลาต่อเนื่อง กับวิธี Backward difference integration ซึ่งให้ผลลัพธ์จากการหาคอนโวลูชันน้อยกว่าคอนโวลูชันแบบเวลาต่อเนื่อง การหาคอนโวลูชันโดยที่วิธีหาค่าเฉลี่ยจากวิธี Forward และ Backward difference integration จะทำได้โดยการคูณฟังก์ชันของสัญญาณอินพุตด้วย $e^{-\lambda\Delta t/2}$ และ $e^{-B\Delta t/2}$ ดังนี้

$$W_{out}\{(n+1)\Delta t\} = W_{out1}(n\Delta t)e^{-\lambda\Delta t} - W_{out2}(n\Delta t)e^{-B\Delta t} + \Delta t A' W_{in}\{(n+1)\Delta t\}e^{-(\lambda\Delta t)/2} + \Delta t B' W_{in}\{(n+1)\Delta t\}e^{-(B\Delta t)/2} \quad \dots (2.11)$$

จากสมการแสดงคอนโวลูชันแบบแบ่งเวลาในสมการ 2.11 จะแสดงเป็นโคอะแกรมสถานะของการคำนวณสำหรับการจำลองเนทเวิร์คแบบวินบริจด์ ดังรูป 2.10



รูป 2.10 โคอะแกรมสถานะของการคำนวณสำหรับการจำลองเนทเวิร์คแบบวินบริจด์

จากรูป 2.10 สัญญาณ $W_{in}(t)$ จะแบ่งเป็นสองทางไปยังวงจรขยายที่มีอัตราขยาย $\Delta tA'e^{-\Lambda\Delta t/2}$ และ $\Delta tB'e^{-B\Delta t/2}$ แล้วรวมกับสัญญาณที่ออกจากบล็อก Z^{-1} คือค่าของ W_{out1} และ W_{out2} จะถูกห่วงไว้ 1 คาบ เวลาการจำลอง (Δt) ก่อนจะนำเอามาคำนวณ โดยสัญญาณจากบล็อก Z^{-1} ตัวบนต้องคูณด้วย $e^{-\Lambda\Delta t}$ และสัญญาณจากบล็อก Z^{-1} ตัวล่างต้องคูณด้วย $e^{-B\Delta t}$ ก่อนที่จะเกิดการรวมกับสัญญาณอินพุท

การทดลอง		ค่าผิดพลาดของอัตราขยาย เมื่อทำ Curve fitting เทียบกับการทดลอง(%)
V(t) โวลต์	อัตราขยาย(A)	
0.0	3.10	0.10
1.0	3.00	-0.35
2.0	2.70	0.00
3.0	2.34	-0.20
4.0	1.96	0.32
5.0	1.63	0.59
6.0	1.38	0.02
7.0	1.20	-0.29
8.0	1.08	0.30
9.0	1.025	0.14
10.0	1.00	0.73
11.0	1.00	0.62
12.0	1.00	0.70
13.0	1.00	0.13
14.0	1.00	-0.70
15.0	1.00	0.24

รูป 2.11 ผลการวัดโวลเตจควบคุม $V_c(t)$ ต่ออัตราขยายของวงจรขยายและค่าผิดพลาดของอัตราขยายเมื่อทำ Curve fitting

ข. การสร้างแบบจำลองวงจรขยายแบบควบคุมอัตราขยายอัตโนมัติ
 ในรูป 2.3 ซึ่งอัตราขยายจะขึ้นกับโวลเตจที่ควบคุม ($V_c(t)$) ในการจำลอง

ส่วนนี้จะต้องหาอัตราขยายที่ระดับโวลเตจควบคุมต่างๆเสียก่อน โดยการป้อนโวลเตจ $V_c(t)$ ค่าต่างๆ แล้ววัดอัตราขยายของสัญญาณเอาต์พุต ($V_o(t)$) ต่อสัญญาณอินพุต ($V_c(t)$) ที่ป้อนเข้าไป ผลการวัดดังตารางในรูป 2.1

ในการจำลองถ้าสามารถวัดค่าโวลเตจควบคุมต่ออัตราขยายที่สัมพันธ์กันได้ละเอียด ก็อาจจะเก็บค่านี้ไว้เป็นตารางในโปรแกรมเพื่อใช้อัตราขยายในระหว่างการจำลอง แต่ในการศึกษานี้จะนำเอาค่าจากการวัดในรูป 2.11 ไปทำ Curve fitting ด้วยวิธี Polynomial regression เพื่อหาสัมประสิทธิ์ของสมการโพลีโนเมียล (Polynomial) ที่จะแทนความสัมพันธ์ระหว่างโวลเตจที่ควบคุมกับอัตราขยาย โดยจากการพิจารณาโพลีโนเมียลลำดับ 6 ก็มีความใกล้เคียงเพียงพอสำหรับการแทนความสัมพันธ์ระหว่างโวลเตจที่ควบคุมกับอัตราขยายของวงจรขยาย ดังค่าผิดพลาดที่แสดงในตาราง 2.11 ซึ่งแสดงเปอร์เซ็นต์ความผิดพลาดของอัตราขยายจากการคำนวณด้วยสมการ 2.12 เทียบกับอัตราขยายที่วัดได้ในแต่ละค่าโวลเตจที่ควบคุมอัตราขยาย

$$A = 3.103 + 0.01547V_c(t) - 0.15273V_c^2(t) + 0.025022V_c^3(t) - 1.5182 \times 10^{-3}V_c^4(t) + 3.108 \times 10^{-5}V_c^5(t) + 7.353 \times 10^{-8}V_c^6(t) \quad \dots (2.12)$$

สมการ 2.12 แสดงคุณลักษณะ (Characteristic) ของอัตราขยาย (A) ซึ่งขึ้นกับโวลเตจที่ควบคุม ($V_c(t)$) และความสัมพันธ์นี้จะนำไปใช้ในการให้คอมพิวเตอร์จำลองระบบต่อไป

ค. การสร้างแบบจำลองของวงจรหาค่าเฉลี่ยเพื่อควบคุมอัตราขยายของวงจรขยาย จากวงจรหาค่าเฉลี่ยของสัญญาณดังรูป 2.5 ซึ่งมีตัวต้านทาน (R) ค่า 560 กิโลโอห์มและตัวเก็บประจุ (C) มีขนาดเท่ากับ 2.2 ไมโครฟาร์ด การจำลองส่วนจัดเรียงกระแสแบบเต็มรูปสัญญาณ สามารถทำได้โดยใช้ฟังก์ชัน ABS(X) ในภาษา TURBO PASCAL เพื่อเปลี่ยนจากค่าตัวเลขทั้งจำนวนบวกและจำนวนลบของสัญญาณให้เป็นจำนวนบวกเสมอ

ในการจำลองระบบในส่วนของอินทิเกรเตอร์ซึ่งมีทรานส์เฟอร์ฟังก์ชันดังสมการ 2.13

$$H_{AV}(S) = \frac{1}{1 + SRC} \quad \dots (2.13)$$

เมื่อหาอินเวอร์สลาปาสทรานส์ฟอร์มของสมการ 2.13 จะได้

$$h_{AV}(t) = \frac{1}{RC} e^{-t/(RC)} \quad \dots (2.14)$$

ดังนั้นจากสมการ 2.14 เมื่อหาคอนโวลูชันระหว่างสัญญาณที่ป้อนเข้าไปยังตัวอินทิเกรเตอร์กับผลตอบสนองต่ออิมพัลส์ของอินทิเกรเตอร์ โดยกำหนดให้สัญญาณอินพุตที่เข้ามายังอินทิเกรเตอร์ในช่วงเวลาที่ t น้อยกว่า 0 มีค่าเป็น 0 จะแสดงได้ว่า

$$V_c(t) = \int_0^t |V_{in}(T)| \cdot h_{AV}(t-T) dT \quad \dots (2.15)$$

โดยให้ $|V_{in}(t)|$ เป็นขนาดของสัญญาณที่ออกจากวงจรจัดเรียงกระแสแบบเต็มคลื่น

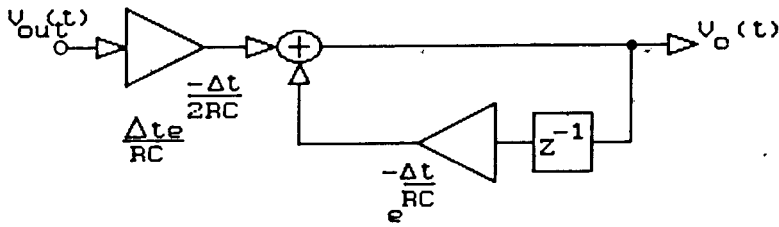
$h_{AV}(t)$ เป็นผลตอบสนองต่ออิมพัลส์ของอินทิเกรเตอร์ ซึ่งหาได้จากอินเวอร์สลาปลาซทรานสฟอร์มของทรานสเฟอร์ฟังก์ชันของอินทิเกรเตอร์ ดังนั้นสมการ 2.15 จะเขียนให้อยู่ในรูปของระบบแบ่งเวลาได้ว่า

$$V_c(n\Delta t) = \sum_{k=0}^n |V_{in}(k\Delta t)| \cdot h_{AV}\{(n-k)\Delta t\} \Delta t \quad \dots (2.16)$$

และสามารถเขียนในรูปของการเวียนเกิด ได้ดังนี้

$$V_c\{(n+1)\Delta t\} = V_c(n\Delta t)e^{-\Delta t/RC} + \frac{\Delta t}{RC} |V_{in}\{(n+1)\Delta t\}| \quad \dots (2.17)$$

นอกจากนี้สามารถหาผลตอบสนองต่อสัญญาณที่ป้อนให้กับตัวอินทิเกรเตอร์ได้ด้วยวิธีการหาคอนโวลูชันแบบแบ่งเวลา และใช้การอินทิเกรตโดยการเฉลี่ยระหว่างวิธี Forward difference integration และ Backward difference integration โดยเขียนได้ดังสมการ 2.18 และแสดงไดอะแกรมสถานะของการคำนวณดังรูป 2.12



รูป 2.12 ไคอะแกรมสถานะของอินทิเกรเตอร์

$$V_o\{(n+1)\Delta t\} = V_o(n\Delta t)e^{-\Delta t/RC} + \frac{\Delta t}{RC} | V_{in}\{(n+1)\Delta t\} | e^{-\Delta t/2RC} \dots (2.18)$$

ง. การสร้างแบบจำลองของการควบคุมอัตราการขยายของวงจร ด้วยขนาดจุดยอดของสัญญาณ การจำลองส่วนตรวจจับจุดยอดของสัญญาณใน คอมพิวเตอร์จะสามารถทำได้ง่าย โดยพิจารณาจากสัญญาณที่ออกจากออสซิลเล-เตอร์ที่เวลา $(n-1)\Delta t$, $n\Delta t$ และ $(n+1)\Delta t$ ซึ่งถ้าสัญญาณที่เวลา $n\Delta t$ เป็น จุดยอดของสัญญาณแล้วจะสามารถแสดงได้ว่า

$$| V_{out}\{(n-1)\Delta t\} | \leq | V_{out}(n\Delta t) | \leq | V_{out}\{(n+1)\Delta t\} |$$

จากนั้นนำเอาขนาดของ $| V_{out}(n\Delta t) |$ ไปคำนวณในส่วนวงจร ขยายแบบควบคุมอัตราขยายอัตโนมัติในข้อ ข.

จ. การสร้างแบบจำลองสัญญาณรบกวนระดับต่ำ สัญญาณรบกวนใน วงจรอิเล็กทรอนิกส์มีได้มากมายหลายคุณลักษณะ และได้มีการศึกษาการสร้าง สัญญาณรบกวนด้วยดิจิทัลคอมพิวเตอร์ไว้หลายวิธี [11], [12] ซึ่งในการวัด สัญญาณรบกวนของวงจรที่ใช้ทดลองที่จุดสัญญาณออกของวงจรขยาย [18] ขณะ ยังไม่เกิดการออสซิลเลต จะมีลักษณะการกระจายของความถี่เป็น Gaussian distribution และมีค่า RMS ประมาณ 3×10^{-4} โวลต์ ซึ่งในการจำลอง สัญญาณรบกวนระดับต่ำนี้จะใช้ฟังก์ชัน Random ของ TURBO PASCAL สำหรับการ สร้างตัวเลขสุ่ม (Random number) โดยให้ตัวเลขสุ่มนี้ผ่านวงจรผ่านแถบ ความถี่เพื่อให้ตัวเลขที่ออกมาจากวงจรผ่านแถบความถี่มีการกระจายของความถี่ ประมาณได้เป็นแบบ Gaussian distribution และมีค่า RMS ประมาณ 3×10^{-4} เหมือนกับสัญญาณรบกวนจริงก่อนที่จะป้อนเข้าสู่ออสซิลเลเตอร์ ซึ่ง

การจำลองวงจรผ่านแถบความถี่จะได้กล่าวต่อไปในบทที่ 3

การสร้างแบบจำลองในข้อ ก. ถึงข้อ จ. จะถูกนำมาเขียนเป็นโปรแกรมคอมพิวเตอร์โดยใช้ภาษา TURBO PASCAL เพื่อจำลองการเกิดของออสซิลเลเตอร์ต่อไป โดยการจำลองที่ 1 จะเป็นการจำลองการเริ่มต้นการออสซิลเลตของออสซิลเลเตอร์ที่มีวงจรขยายเป็นแบบควบคุมด้วยขนาดของยอด (Peak) สัญญาณออกจากออสซิลเลเตอร์และเริ่มต้นการออสซิลเลตด้วยอิมพัลส์ที่เกิดขึ้นในวงจร การจำลองที่ 2 เป็นการจำลองการเริ่มต้นการออสซิลเลตของออสซิลเลเตอร์ที่มีวงจรขยายเป็นแบบควบคุมด้วยค่าเฉลี่ยของสัญญาณออกจากออสซิลเลเตอร์และเริ่มต้นการออสซิลเลตด้วยสัญญาณรบกวนระดับต่ำ , การจำลองที่ 3 เป็นการจำลองการเริ่มต้นการออสซิลเลตของออสซิลเลเตอร์ที่มีวงจรขยายเป็นแบบเคี้ยวกับการจำลองที่ 2 แต่เริ่มต้นการออสซิลเลตด้วยการป้อนตัวเลขแทนอิมพัลส์ให้กับวงจร



การจำลองที่ 1

การจำลองออสซิลเลเตอร์แบบวินบริคส์ ที่มีการควบคุมอัตราขยาย ด้วยวงจรแปลงขนาดสัญญาณแบบตรวจสอบยอดสัญญาณ และเริ่มต้นการออสซิลเลตด้วยสัญญาณอิมพัลส์ซึ่งเกิดขึ้นในวงจร ในการจำลองที่ 1 นี้จะจำลองวงจรในรูป 2.6 การจำลองวงจรดังกล่าวที่เวลา $(n+1)\Delta t$ จะทำเป็นแต่ละขั้นได้ดังนี้

ขั้นที่ 1 : เป็นการจำลองเน็ตเวิร์คแบบวินบริคส์ โดยการคำนวณตามสมการ 2.11 $W_{in}\{(n+1)\Delta t\}$ ในสมการ 2.11 เป็นผลรวมจากผลลัพธ์ของการคำนวณในขั้นที่ 4 ที่เวลา $n\Delta t$ กับสัญญาณอิมพัลส์ ซึ่งจากการทดลองวัดสัญญาณที่เข้าสู่เน็ตเวิร์คแบบวินบริคส์ในตอนเริ่มต้นการออสซิลเลต จะประมาณสัญญาณอิมพัลส์ให้มีขนาดคงที่เท่ากับ 2.0 โวลต์ในช่วงเวลาตั้งแต่เริ่มต้นการจำลองคือเวลา t เท่ากับ 0 จนกระทั่งเวลาเท่ากับ $20\Delta t$ ถ้าเวลามากกว่า $20\Delta t$ จะให้สัญญาณอิมพัลส์ที่จำลองมีค่าเป็น 0 โวลต์ คือ

$$\text{สัญญาณอิมพัลส์} = \begin{cases} 2.0 & ; 0 \leq t \leq 20\Delta t \\ 0.0 & ; t > 20\Delta t \end{cases}$$

ขั้นที่ 2 : เป็นการจำลองส่วนแปลงขนาดสัญญาณ (Amplitude converter) โดยมีวิธีการจำลองดังในข้อ ง. และมี $|V_{in}\{(n+1)\Delta t\}|$ เป็นค่าสัมบูรณ์ (Absolute) ของผลคูณจากการคำนวณในขั้นที่ 1 คือ $W_{out}\{(n+1)\Delta t\}$ กับอัตราขยายของวงจรขยายในระบบที่เวลา $n\Delta t$

ขั้นที่ 3 : เป็นการจำลองวงจรขยายเพื่อหาอัตราขยายของวงจรขยาย โดยใช้ผลจากการคำนวณหาสัญญาณออกจากวงจรแปลงขนาดสัญญาณในขั้นที่ 2 คือ $|V_{out}\{(n+1)\Delta t\}|$ มาแทนค่า $V_o(t)$ ในสมการ 2.12 ซึ่งอัตราขยายที่ได้จะนำไปคำนวณในขั้นที่ 2 ที่เวลา $(n+2)\Delta t$ และใช้สำหรับการคำนวณในขั้นที่ 4

ขั้นที่ 4 : คำนวณหาขนาดของสัญญาณออกจากวงจรขยาย โดยคูณอัตราขยายที่คำนวณได้จากขั้นที่ 3 กับสัญญาณออก (W_{out} จากสมการ 2.11) จากเน็ตเวิร์คแบบวินบริคส์ในขั้นที่ 1 ซึ่งสัญญาณออกที่ได้จะนำไปคำนวณในขั้นที่ 1 ที่เวลา $(n+2)\Delta t$

จำลองซ้ำตั้งแต่ขั้นที่ 1 ถึงขั้นที่ 4 จะได้เอาท์พุทจากการจำลองแสดงเปรียบเทียบกับผลการทดลองดังรูป 2.7 โดยที่มีสเกลของขนาดสัญญาณในแนวตั้งเท่ากัน แต่สเกลของเวลาไม่เท่ากัน

การจำลองที่ 2

การจำลองออสซิลเลเตอร์แบบวินบริดจ์ ที่มีการควบคุมอัตราการขยายของวงจรขยายด้วยวงจรแปลงขนาดสัญญาณแบบหาค่าเฉลี่ย และเริ่มต้นการออสซิลเลตด้วยสัญญาณรบกวนระดับต่ำ มีลำดับขั้นตอนของการจำลองที่เวลา $(n+1)\Delta t$ ดังนี้

ขั้นที่ 1 : ทำการจำลองเนตเวิร์คแบบวินบริดจ์โดยการคำนวณตามสมการ 2.11 ซึ่ง $W\{(n+1)\Delta t\}$ ในสมการ 2.11 เป็นผลรวมจากผลลัพธ์ในการคำนวณขั้นที่ 4 ที่เวลา $n\Delta t$ กับตัวเลขสุ่มแทนสัญญาณรบกวนในระบบ ซึ่งสร้างได้ดังข้อ จ. ในบทที่ 2.4

ขั้นที่ 2 : จำลองส่วนแปลงสัญญาณในรูป 2.12 และข้อ ค. ในบทที่ 2.4 เพื่อหาสัญญาณออกจากวงจรแปลงสัญญาณแบบหาค่าเฉลี่ย โดยการคำนวณจากสมการ 2.18 ซึ่ง $|V_{in}\{(n+1)\Delta t\}|$ ในสมการ 2.18 นั้นเป็นค่าสัมบูรณ์ของผลคูณของ $W\{(n+1)\Delta t\}$ ในการจำลองขั้นที่ 1 กับอัตราขยาย โดยที่ $V_o\{(n+1)\Delta t\}$ ซึ่งเป็นค่าที่ได้จากการคำนวณในขั้นที่ 2 จะถูกนำไปใช้ในขั้นที่ 3

ขั้นที่ 3 : หาอัตราขยายของวงจรขยาย โดย $V_o\{(n+1)\Delta t\}$ ซึ่งเป็นผลการคำนวณจากขั้นที่ 2 จะถูกนำมาแทนค่าในสมการ 2.12 ซึ่งอัตราขยายที่ได้นี้จะนำไปคำนวณในขั้นที่ 2 ที่เวลา $(n+2)\Delta t$ และใช้ในการคำนวณในขั้นที่ 4-ที่เวลา $n\Delta t$

ขั้นที่ 4 : คำนวณหาสัญญาณออก (W_{out}) จากวงจรขยาย โดยคูณอัตราขยายที่คำนวณได้จากขั้นที่ 3 กับสัญญาณที่ออกจากเนตเวิร์คแบบวินบริดจ์ในขั้นที่ 1 และเก็บผลการคูณไว้สำหรับการจำลองในขั้นที่ 1 ของคาบเวลาการจำลองต่อไป

จำลองซ้ำตั้งแต่ขั้นที่ 1 ถึงขั้นที่ 4 จะได้เอาท์พุทจากการจำลองแสดงเปรียบเทียบกับผลการทดลองดังรูป 2.8 โดยที่มีสเกลของขนาดสัญญาณในแนวตั้งเท่ากัน แต่สเกลของเวลาไม่เท่ากัน

การจำลองที่ 3

การจำลองออสซิลเลเตอร์แบบวินบริจด์ ที่มีการควบคุมอัตราขยายของวงจรขยายด้วยวงจรแปลงขนาดสัญญาณหาค่าเฉลี่ย และเริ่มต้นการออสซิลเลตด้วยสัญญาณอิมพัลส์ ซึ่งวิธีการจำลองระบบจะเหมือนกับการจำลองที่ 2 แตกต่างกันเฉพาะในขั้นที่ 1 ซึ่ง $w[(n+1)\Delta t]$ ในสมการ 2.11 นั้น เป็นผลรวมจากผลลัพธ์การคำนวณในขั้นที่ 4 กับอิมพัลส์ซึ่งจะจำลองได้เหมือนการจำลองสัญญาณอิมพัลส์ในการจำลองที่ 1 ผลการจำลองแสดงไว้ในรูป 2.9 เปรียบเทียบกับผลการทดลองที่ 3 โดยที่มีสเกลของขนาดสัญญาณในแนวตั้งเท่ากัน แต่สเกลของเวลาไม่เท่ากัน



2.5 เอกสารอ้างอิงบทที่ 2

- [1] Douglas N.G., "Global Stability Analysis of Automatic Gain Control Circuits," IEEE Trans. Circuit Syst., Vol.CAS-30, pp.78-83, Feb.1983.
- [2] Miller R.K. and Antony N.M., "Analysis of Transient Oscillators in a Nonlinear Feedback System," IEEE Trans. Circuit Syst., Vol.CAS-33, pp.455-458, Apr.1983.
- [3] Kobylarz T.J., "Simulation of Linear and Nonlinear Circuits for Symbolic Analysis and Transient Response," IEEE Trans. Education, Vol.E-12, pp.299-305, Dec.1969.
- [4] Kenneth S.K. and Sangiovanni V.N., "Simulation of Nonlinear Circuits in the Frequency Domain," IEEE Trans. Computer-aided design, Vol.CAD-5, pp.512-535, Oct.1986.
- [5] Letzter S. and Webster N., "Noise in Amplifier," IEEE Spectrum, pp.67-75, Aug.1970.
- [6] Finnegan P.J., "Noise Analysis of feedback amplifiers," Electronic Engineering, pp.612-616, Oct.1967.
- [7] Edson W.A., "Noise in Oscillators," Proc.IRE, pp.1454-1466, Aug.1960.
- [8] Velazque J.R., Stephenson F.W., Riad S.M. and Riad A.A., "Noise Analysis of Analog and Digital Circuits Using Autoregressive Power Spectrum Estimation," IEEE Trans. Circuit Syst., Vol.CAS-30, pp.193-195, Mar.1983.
- [9] Mastascusa E.J., "A Fast Algorithm for Computing The Response of a Linear System to a Large Number of Inputs," Proc.IEEE, pp.338-339, Mar.1964.
- [10] Abu-el-haija A.I. and Peterson A.M., "An Approach to Estimate Round of Errors in Digital Filters," IEEE Trans. Acoustics, Speech and Signal Processing, Vol.ASSP-27, No.2, pp.195-198, Apr.1979.
- [11] Francis N.D., "Generation of Random Numbers on Micros - A simulation Study," Microprocessing and Microprogramming

North-Holland, pp.17-19, 1985.

- [12] Kafadar K., "Gaussian White-Noise Generation for Digital Signal Synthesis," IEEE Trans. Instrumentation and Measurement, Vol.IM-35, NO.4, pp.492-495, Dec.1986.
- [13] Abu-el-haija A.I. and Al-ibrahim M.M., "Improving Performance of Digital Simulation Oscillator by Means of Error Feedback Circuits," IEEE Trans. Circuit Syst., Vol.CAS-33, pp.373-379, Apr.1986.
- [14] Liu E.S.K. and Turner L.E., "Stability, Dynamic Range and Round off Noise in a New Second-Order Recursive Digital Filter," IEEE Trans. Circuit Syst., Vol.CAS-30, pp.815-821, Nov.1983.
- [15] Antonou A., Charalambous C. and Motazediz., "Two Methods for The Reduction of Quantization Effects in Recursive Digital Filters," IEEE Trans. Circuit Syst., Vol.CAS-30, No.30, pp.160-166, Mar.1983.
- [16] Bormar B.N., "Computationally Efficient Low Round off Noise Second-Order State-Space Structures," IEEE Trans. Circuit Syst., Vol.CAS-33, No.1, pp.35-40, Jan.1986.
- [17] Yeung K.S., "An Alternative Approach for Analyzing Feedback Amplifiers," IEEE Trans. Education, Vol.E-25, No.4, pp.132-136, Nov.1982.
- [18] Franco S., "Design with Operational Amplifiers and Analog Integrated Circuit," McGraw-Hill Book Company, pp.594-599, 1988.

บทที่ 3

อินเจกชันล็อกกิ้งของออสซิลเลเตอร์ (Injection locking of oscillator)

3.1 บทนำอินเจกชันล็อกกิ้งของออสซิลเลเตอร์

อินเจกชันล็อกกิ้ง (Injection locking) ของออสซิลเลเตอร์ เป็นปรากฏการณ์อย่างหนึ่ง เกิดขึ้นโดยการที่ออสซิลเลเตอร์ถูกป้อนด้วยสัญญาณที่มีความถี่ใกล้เคียงกับความถี่ทำงานเสรี (Free-running frequency) ของออสซิลเลเตอร์ เหมือนกับเป็นสัญญาณรบกวน ($E_c(t)$) เข้าไปในออสซิลเลเตอร์ดังรูป 3.1 ขณะที่เกิดการล็อก ออสซิลเลเตอร์จะผลิตสัญญาณที่มีความถี่เดียวกับความถี่ของสัญญาณที่ป้อนเข้าไป

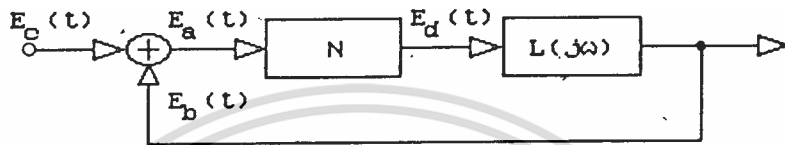
อินเจกชันล็อกกิ้งมีประโยชน์มากในวงจรออสซิลเลเตอร์ เช่น วงจรออสซิลเลเตอร์ควบคุมด้วยผลึก (Crystal-controlled oscillator) จะให้ Long-term stability ดีเมื่อให้ผลึก (Crystal) ทำงานที่ระดับสัญญาณมีขนาดต่ำ แต่วงจรออสซิลเลเตอร์ควบคุมด้วยผลึกจะให้ Short-term phase stability ดี เมื่อให้ผลึกทำงานที่ระดับสัญญาณมีขนาดสูง ดังนั้นเพื่อให้ได้ผลดีที่สุดคือการรวมทั้งสองกรณีเข้าด้วยกัน โดยใช้สัญญาณจากออสซิลเลเตอร์ควบคุมด้วยผลึกแบบระดับสัญญาณต่ำไปล็อกออสซิลเลเตอร์ควบคุมด้วยผลึกแบบระดับสัญญาณสูง จะได้ออสซิลเลเตอร์ซึ่งมีผลดีของทั้งสองแบบรวมกัน

ในการศึกษาการเกิดปรากฏการณ์ของอินเจกชันล็อกกิ้งนี้ Adler [1] และ Paciorek [2] ได้ทำการศึกษาการเกิดอินเจกชันล็อกกิ้งแบบทิศทางเดียว โดยใช้ออสซิลเลเตอร์แบบวงจรจูนเดียว (Single tuned circuit) ซึ่งมีส่วนกำหนดความถี่ทำงานเสรีเป็นวงจรจูนเดียว และผลการศึกษาเป็นที่ยอมรับอย่างกว้างขวาง แม้ว่าต่อมาจะมีการศึกษาเพิ่มเติมเพื่อให้ผลการศึกษาทางทฤษฎีใกล้เคียงกับผลทางปฏิบัติ แต่ก็อาศัยหลักการพื้นฐานของ Adler [3], [4], [5] ซึ่งการศึกษาของบทความ [5] ในปี 1985 ก็ศึกษาการล็อกของออสซิลเลเตอร์แบบทิศทางเดียว โดยทำการศึกษาทั้งกับ Colpitts oscillator แต่ก็ยังมีข้อจำกัดในการวิเคราะห์เช่นเดียวกับของ Adler

ในบทนี้จะใช้วิธีการจำลองระบบแบบแบ่งเวลาของวงจรออสซิลเลเตอร์ในบทที่ 2 ซึ่งให้ผลการจำลองระบบไม่เชิงเส้นตรงกับการทดลอง มาทำการศึกษาปรากฏการณ์ของอินเจกชันล็อกกิ้งในวงจรจูนเดียว เพื่อทำการเปรียบเทียบกับการทดลอง และได้ทำการจำลองการเกิดปรากฏการณ์เชื่อมต่อกัน

(Mutual coupling) ซึ่งเป็นปรากฏการณ์ที่ออสซิลเลเตอร์ตั้งแต่สองตัวขึ้นไปส่งสัญญาณไปล็อกกันและกัน อันเป็นกรณีที่ไม่สามารถวิเคราะห์ทางทฤษฎีได้โดยง่าย

3.2 อินжекชั่นล็อกกิ้งของออสซิลเลเตอร์



รูป 3.1 บล็อกแผนภาพของวงจรออสซิลเลเตอร์

บล็อก N เป็นวงจรขยายแบบไม่เชิงเส้น และบล็อก $L(jw)$ เป็นบล็อกของเน็ตเวิร์คที่ควบคุมความถี่ของการออสซิลเลต ในการพิจารณาจะให้การเลื่อนเฟสทั้งหมดในวงรอบ เกิดจากส่วนของเน็ตเวิร์คเท่านั้น

กำหนดให้

- ω_0 = ความถี่เชิงมุมทำงานเสรีของออสซิลเลเตอร์ เมื่อไม่มีการป้อนสัญญาณ $E_c(t)$ เข้าไปล็อก
- ω_1 = ความถี่เชิงมุมของสัญญาณ $E_c(t)$ ที่ป้อนเข้าไปเพื่อล็อกออสซิลเลเตอร์
- $\Delta\omega_0$ = ผลต่างความถี่เชิงมุมระหว่าง ω_0 กับ ω_1
- ω = ความถี่เชิงมุมของสัญญาณ ($E_b(t)$) ที่ออกจากออสซิลเลเตอร์เมื่อป้อนสัญญาณเข้าไปล็อกออสซิลเลเตอร์
- $\Delta\omega$ = ผลต่างความถี่เชิงมุมระหว่าง ω กับ ω_1
- E_c = สัญญาณที่ป้อนเข้าไปเพื่อล็อกออสซิลเลเตอร์
- E_b = สัญญาณเอาต์พุตของออสซิลเลเตอร์ที่ป้อนกลับมาทางอินพุต
- E_a = ผลรวมทางเฟสเซอร์ (Phasor) ของสัญญาณ E_b กับ E_c
- Q = ตัวประกอบคุณภาพ (Quality factor) ของวงจรจูน

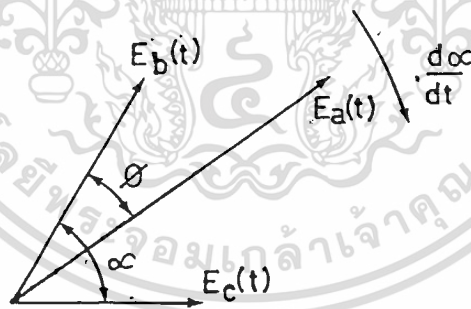
α = ค่าของมุม (Phase) ระหว่างสัญญาณ E_c และ E_b

การวิเคราะห์ปรากฏการณ์อินжекชั่นล็อกกิ้งของออสซิลเลเตอร์ โดยวิธีของ Adler ซึ่งใช้เป็นพื้นฐานในการวิเคราะห์อินжекชั่นล็อกกิ้งในบทความอื่น จะมีสมมติฐานเบื้องต้นคือ

1. ความถี่ของสัญญาณที่ส่งเข้าไปล็อกออสซิลเลเตอร์ จะต้องใกล้เคียงกับความถี่ศูนย์กลาง (Center frequency) ของย่านผ่านความถี่ของวงจรจูน และเมื่อความถี่ของสัญญาณที่ใส่เข้าไปล็อกออสซิลเลเตอร์ยังมีค่าใกล้เคียงกับความถี่ทำงานเสรีของออสซิลเลเตอร์เท่าใด สัญญาณที่ออสซิลเลเตอร์ผลิตออกมา ก็จะมีขนาดสูงขึ้น เสมือนว่าวงจรออสซิลเลเตอร์มีคุณสมบัติเป็นวงจรผ่านแถบความถี่ที่มีความถี่ศูนย์กลางเท่ากับความถี่ทำงานเสรีของออสซิลเลเตอร์

2. $\tau \ll 1/\omega$ เมื่อ τ เป็นค่าคงตัวเวลา (Time constant) ของวงจรควบคุมอัตราขยาย ทั้งนี้เพื่อให้มีเวลาลดทอย (Decay time) พอที่จะเก็บขนาดและเฟสของสัญญาณที่ออกจากออสซิลเลเตอร์ไว้เฉพาะในช่วงเวลา τ โดยจะมีผลต่อสัญญาณในคาบเวลาต่อไปน้อย

3. $E_c/E_b \ll 1$ คือขนาดของสัญญาณที่ใส่เข้าไปเพื่อล็อกออสซิลเลเตอร์ ต้องมีขนาดไม่สูงมาก เพื่อให้สมการที่จะศึกษาต่อไปเป็นจริง



รูป 3.2 แผนภาพเฟเซอร์ของสัญญาณ E_a , E_b และ E_c

จากวงจรจูนเดี่ยวสามารถเขียนแผนภาพเฟสเซอร์ (Phasor diagram) แสดงความสัมพันธ์ระหว่างสัญญาณ E_a , E_b และ E_c ได้ดังรูป 3.2

จากรูป 3.2 ให้ $E_c(t)$ เป็นเฟสเซอร์อ้างอิง (Reference) แล้วเฟสเซอร์ของสัญญาณอื่นจะอ้างอิงกับเฟสเซอร์นี้ ในขณะที่ป้อนสัญญาณ $E_c(t)$ เข้าไปล็อกออสซิลเลเตอร์นั้น เฟสเซอร์ของสัญญาณ $E_b(t)$ ซึ่งเป็นเฟสเซอร์รวมระหว่างเฟสเซอร์ของสัญญาณ $E_b(t)$ กับเฟสเซอร์ของสัญญาณ $E_c(t)$ จะหมุนตามเข็มหรือทวนเข็มนาฬิกาก็ได้ขึ้นกับการเลื่อนเฟส (Phase shifting) ที่เกิดขึ้นในบล็อก $L(j\omega)$ ในรูป 3.2 จะให้เฟสเซอร์ของสัญญาณ $E_b(t)$ หมุนตามเข็มนาฬิกา และความถี่เชิงมุมของสัญญาณ $E_b(t)$ ตามรูปจะให้มีความถี่เท่ากับ $\frac{d\omega}{dt}$

ดังนั้นสัญญาณ $E_b(t)$ จะมีความถี่เชิงมุมเท่ากับ $\omega_1 + \frac{d\omega}{dt}$

ให้ $\Delta\omega$ เป็นผลต่างความถี่เชิงมุมระหว่างสัญญาณ E_b กับสัญญาณ E_c มีค่า

$$\Delta\omega = \frac{d\omega}{dt}$$

ซึ่งในตอนเริ่มต้นป้อนสัญญาณ E_c เข้าไปล็อกออสซิลเลเตอร์นั้น $\Delta\omega$ จะมีค่าเปลี่ยนแปลงตามเวลา จนกว่าจะเกิดการล็อกซึ่งสัญญาณ E_b และ E_c จะมีความถี่เท่ากัน

เฟสเซอร์ E_b จะตาม (Lag) กับเฟสเซอร์ E_c อยู่เท่ากับ θ ซึ่งเป็นผลเนื่องจากการเลื่อนเฟสของบล็อก $L(j\omega)$ ในรูป 3.4 จากแผนภาพเฟสเซอร์จะแสดงความสัมพันธ์ของเฟส (θ) ระหว่างสัญญาณ E_b กับสัญญาณ E_c ได้ดังสมการ 3.1

$$\begin{aligned} \tan(\theta) &= \frac{E_c \sin(-\alpha)}{E_b + E_c \cos(-\alpha)} \\ &= \frac{-E_c \sin(\alpha)}{E_b + E_c \cos(\alpha)} \end{aligned} \quad \dots (3.1)$$

สำหรับวงจรแบบจูนเดี่ยว จะเขียนสมการแสดงค่าเฟสที่มีการเลื่อนได้ว่า

$$\tan(\theta) = \frac{2Q(\omega - \omega_0)}{\omega_0} \quad \dots (3.2)$$

จากสมการ 3.1 และ 3.2 จะสามารถจัดรูปสมการ เพื่อแสดงการเปลี่ยนแปลงของความถี่ของออสซิลเลเตอร์เมื่อเริ่มป้อนสัญญาณ $E_c(t)$ เข้าไปล็อกได้ดังนี้

$$\begin{aligned}
 \frac{-E_c \sin(\alpha)}{E_b + E_c \cos(\alpha)} &= \frac{2Q(\omega - \omega_0)}{\omega} \\
 &= \frac{2Q[(\omega - \omega_1) - (\omega_0 - \omega_1)]}{\omega} \\
 &= \frac{2Q(\Delta\omega - \Delta\omega_0)}{\omega} \\
 &= \frac{2Q(d\alpha - \Delta\omega)}{\omega dt}
 \end{aligned}$$

ซึ่งสามารถหาอัตราการเปลี่ยนแปลงของเฟสระหว่างสัญญาณ E_c และ E_b ต่อหน่วยเวลาได้ว่า

$$\frac{d\alpha}{dt} = \Delta\omega - \frac{(\omega_0 \cdot E_c)}{2Q E_b} \left\{ \frac{\sin(\alpha)}{1 + \frac{E_c \cos(\alpha)}{E_b}} \right\} \quad \dots (3.3)$$

ที่สถานะอยู่ตัวคือเกิดการล็อกขึ้นนั้น สัญญาณ E_c, E_a และ E_b จะมีความถี่เดียวกัน ดังนั้นอัตราการเปลี่ยนแปลงของเฟสระหว่างสัญญาณ E_c และ E_b ต่อหน่วยเวลาจะมีค่าเป็นศูนย์ คือ

$$\frac{d\alpha}{dt} = 0$$

ดังนั้นสมการ 3.3 จะแสดงได้เป็น

$$\Delta\omega = \frac{(\omega_0 \cdot E_c)}{2Q E_b} \left\{ \frac{\sin(\alpha)}{1 + \frac{E_c \cos(\alpha)}{E_b}} \right\} \quad \dots (3.4)$$

จากสมการ 3.4 จะหาช่วงความถี่ของสัญญาณ E_c ที่สามารถล็อก ออสซิลเลเตอร์ซึ่งเรียกว่าย่านการล็อก (Locking range) ได้ดังสมการ 3.5

$$\frac{d(\Delta\omega)}{d\alpha} = \frac{\omega_0 \cdot E_c}{2Q E_b} \left\{ \frac{1 + \frac{E_c \cos(\alpha)}{E_b} \cdot \cos(\alpha) + \frac{E_c \sin^2(\alpha)}{E_b}}{1 + \frac{2E_c \cos(\alpha)}{E_b} + \left[\frac{E_c \cos(\alpha)}{E_b} \right]^2} \right\} \quad \dots (3.5)$$

จากสมการ 3.5 นั้น ย่านการล็อกสูงสุดจะเกิดขึ้นเมื่อ $\Delta\omega$ มีค่าสูงสุด คือ $\frac{d\Delta\omega}{d\alpha}$ มีค่าเป็น 0

ดังนั้นจากสมการ 3.5 เมื่อกำหนดให้ $\frac{d\Delta\omega_0}{d\alpha} = 0$ จะแสดงได้ว่า

$$\cos(\alpha) + \frac{E_c}{E_b} \cos^2(\alpha) + \frac{E_c}{E_b} \sin^2(\alpha) = 0$$

เนื่องจาก $\sin^2(\alpha) + \cos^2(\alpha) = 1$ ดังนั้น

$$\cos(\alpha) = \frac{-E_c}{E_b}$$

$$\sin(\alpha) = [1 - \cos^2(\alpha)]^{1/2} = \left\{ 1 - \left(\frac{E_c}{E_b} \right)^2 \right\}^{1/2}$$

$$\Delta\omega_{0\max} = \frac{(\omega_0 \cdot E_c)}{2Q E_b} \cdot \frac{1}{\left\{ 1 - \left(\frac{E_c}{E_b} \right)^2 \right\}^{1/2}} \quad \dots (3.6)$$

สมการ 3.6 เป็นย่านล็อกสูงสุดของวงจรจูนเดี่ยว ซึ่งแสดงให้เห็นว่าเมื่อตัวประกอบคุณภาพ (Q) มีค่ามากขึ้นย่านล็อกจะมีค่าลดลงดังสมการ 3.6

กำหนดให้ $B = \frac{(\omega_0 \cdot E_c)}{2Q E_b} \cdot \frac{1}{1 + \frac{E_c}{E_b} \cos(\alpha)}$

และ $K = \frac{\Delta\omega_0}{B} = \frac{2Q \cdot E_b \cdot \Delta\omega_0}{E_c \cdot \omega_0}$

ในกรณีที่ไมเกิดการล็อกขึ้นนั้น จากสมการ 3.3 จะเขียนได้ว่า

$$\frac{d\alpha}{dt} = -B[\sin(\alpha) - K]$$

เมื่อการอินทิเกรตทั้ง 2 ข้างเทียบกับ t จะได้

$$\frac{\alpha}{2} = \tan^{-1} \left\{ \frac{1}{K} + \frac{(K^2 - 1)^{1/2} \cdot \tan B(t - t_0) \cdot (K^2 - 1)^{1/2}}{K} \right\} \quad \dots (3.7)$$

ถ้า $|K| > 1$ จะได้ $(K^2 - 1)^{1/2}$ เป็นค่าจริง (Real value)

t_0 เป็นค่าคงที่ของผลลัพธ์จากการอินทิเกรต

เมื่อ t มีค่าเพิ่มขึ้นตั้งแต่ศูนย์ถึงอนันต์ (Infinity) ดังนั้นเทอม

$$\frac{B(t - t_0)(K^2 - 1)^{1/2}}{2} \text{ จะมีค่าตั้งแต่ } \frac{\pi}{2} \text{ ถึง } \frac{3\pi}{2}$$

- กำหนดให้ Q_1 เป็นตัวประกอบคุณภาพของออสซิลเลเตอร์ตัวที่ 1
 Q_2 เป็นตัวประกอบคุณภาพของออสซิลเลเตอร์ตัวที่ 2
 ω_1 เป็นความถี่เชิงมุมเสรีของออสซิลเลเตอร์ตัวที่ 1
 ω_2 เป็นความถี่เชิงมุมเสรีของออสซิลเลเตอร์ตัวที่ 2
 ω เป็นความถี่เชิงมุมของสัญญาณที่ออกจากออสซิลเลเตอร์ทั้งสอง
 ω_1 เป็นเฟสระหว่างสัญญาณ $V_{1c}(t)$ และสัญญาณ $V_{1b}(t)$
 ω_2 เป็นเฟสระหว่างสัญญาณ $V_{2c}(t)$ และสัญญาณ $V_{2b}(t)$

$$\Delta\omega_1 = \omega_1 - \omega$$

$$\Delta\omega_2 = \omega_2 - \omega$$

จากรูป 3.3 และสมการ 3.4 ที่สถานะอยู่ตัว คือเกิดการล็อกระหว่างออสซิลเลเตอร์ทั้งสอง สามารถหาผลต่างของความถี่ขณะเกิดการล็อกกับความถี่เสรีของออสซิลเลเตอร์ทั้งสองได้ว่า

$$\Delta\omega_1 = \frac{(\omega_1 \cdot C_1)}{2Q_1} \cdot \frac{\sin(\omega_1)}{(1 + C_1 \cos(\omega_1))}$$

$$\Delta\omega_2 = \frac{(\omega_2 \cdot C_2)}{2Q_2} \cdot \frac{\sin(\omega_2)}{(1 + C_2 \cos(\omega_2))}$$

เมื่อ C_1 เป็นค่าตัวประกอบเชื่อมต่อกของสัญญาณจากออสซิลเลเตอร์ 2 ไปออสซิลเลเตอร์ 1 และมีค่าเท่ากับ $\frac{V_{1c}(t)}{V_{1b}(t)}$

C_2 เป็นค่าตัวประกอบเชื่อมต่อกของสัญญาณจากออสซิลเลเตอร์ 1 ไปออสซิลเลเตอร์ 2 และมีค่าเท่ากับ $\frac{V_{2c}(t)}{V_{2b}(t)}$

ย่านการล็อกสูงสุดของออสซิลเลเตอร์แต่ละตัวคือ ω_{1max} และ ω_{2max} มีค่าดังนี้

$$\Delta\omega_{1max} = \frac{(\omega_1 \cdot C_1)}{2Q_1} \cdot \frac{1}{(1 - C_1^2)^{1/2}}$$

$$\Delta\omega_{2max} = \frac{(\omega_2 \cdot C_2)}{2Q_2} \cdot \frac{1}{(1 - C_2^2)^{1/2}}$$

เฟสระหว่างสัญญาณออกกับสัญญาณเข้าของวงจรจูนเคียบว จะเลื่อนอยู่ในช่วงจาก $-\pi$ ถึง π ซึ่งในกรณีของการล็อกแบบเชื่อมต่อกพร้อมเมื่ออยู่ในสถานะอยู่ตัวนั้น จะมีความถี่ของสัญญาณที่ออกจากออสซิลเลเตอร์ทั้งสองตัวเท่ากัน และความถี่นี้จะเป็นความถี่ระหว่างความถี่ทำงานเสรีของออสซิลเลเตอร์ทั้งสอง ขณะที่เกิดการล็อก จะทำให้เกิดเฟสระหว่างสัญญาณที่ป้อนเข้าไปกับ

สัญญาณที่ออกมาจากออสซิลเลเตอร์ตัวหนึ่งเป็นแบบนำ (leading) และเฟสที่ออสซิลเลเตอร์อีกตัวจะเป็นแบบตาม (lagging) ดังนั้นย่านล็อกสูงสุดจะต้องพิจารณาใน 2 กรณีคือ

ก. ในกรณีเฟสระหว่างสัญญาณที่ป้อนเข้าไปกับสัญญาณที่ออกมาจากออสซิลเลเตอร์ 1 เป็นแบบตาม และเฟสที่ออสซิลเลเตอร์ 2 เป็นแบบนำ คือ $\phi_1 > 0, \phi_2 < 0$ จะได้ว่า

$$\begin{aligned}\omega_2 &= \omega_1 - \frac{\Delta\omega_{1max}}{2} - \frac{\Delta\omega_{2max}}{2} \\ &= \omega_1 - \frac{(\omega_1 \cdot C_1)}{4Q_1} \cdot \frac{1}{(1 - C_1^2)^{1/2}} - \frac{(\omega_2 \cdot C_2)}{4Q_2} \cdot \frac{1}{(1 - C_2^2)^{1/2}} \quad \dots(3.8)\end{aligned}$$

ข. ในกรณีเฟสระหว่างสัญญาณที่ป้อนเข้าไปกับสัญญาณที่ออกมาจากออสซิลเลเตอร์ 2 เป็นแบบตาม และเฟสที่ออสซิลเลเตอร์ 1 เป็นแบบนำ คือ $\phi_1 < 0, \phi_2 > 0$ จะได้ว่า

$$\begin{aligned}\omega_2 &= \omega_1 + \frac{\Delta\omega_{1max}}{2} + \frac{\Delta\omega_{2max}}{2} \\ &= \omega_1 + \frac{(\omega_1 \cdot C_1)}{4Q_1} \cdot \frac{1}{(1 - C_1^2)^{1/2}} + \frac{(\omega_2 \cdot C_2)}{4Q_2} \cdot \frac{1}{(1 - C_2^2)^{1/2}} \quad \dots(3.9)\end{aligned}$$

ดังนั้นจากสมการ 3.8 และ 3.9 แสดงว่าย่านล็อกอยู่ระหว่าง

$$\omega_2 = \omega_1 \left\{ \frac{1 - \frac{C_1}{2Q_1} \cdot \frac{1}{(1 - C_1^2)^{1/2}}}{1 + \frac{C_2}{2Q_1} \cdot \frac{1}{(1 - C_2^2)^{1/2}}} \right\}$$

ถึง

$$\omega_2 = \omega_1 \left\{ \frac{1 + \frac{C_1}{2Q_1} \cdot \frac{1}{(1 - C_1^2)^{1/2}}}{1 - \frac{C_2}{2Q_1} \cdot \frac{1}{(1 - C_2^2)^{1/2}}} \right\}$$

จากสมการ 3.1 เมื่อให้อัตราส่วนของ E_c ต่อ E_b มีค่าน้อยมาก ค่าของ $\tan(\phi)$ จะมีค่าประมาณเท่ากับ $-\sin(\phi)$

ดังนั้น ที่ออสซิลเลเตอร์ 1 สามารถเขียนสมการ 3.1 ได้เป็น

$$\tan(\phi_1) = \frac{-2Q_1(\omega - \omega_1)}{\omega_1} = -\sin(\phi_1) \quad \dots(3.10)$$

และที่ออสซิลเลเตอร์ 2 สามารถเขียนสมการ 3.1 ได้เป็น

$$\tan(\phi_2) = \frac{-2Q_1(\omega - \omega_2)}{\omega_2} = -\sin(\phi_2) \quad \dots(3.11)$$

ที่สถานะอยู่ตัวของการล็อกแบบเชื่อมต่อยุบรวมระหว่างออสซิลเลเตอร์แบบจูนเดี่ยว 2 ตัว จะมีเฟสระหว่างสัญญาณที่เข้ามอล็อกกับสัญญาณออกจากออสซิลเลเตอร์ขนาดเท่ากัน แต่ทิศทางการเคลื่อนที่ตรงกันข้าม แสดงได้ว่า

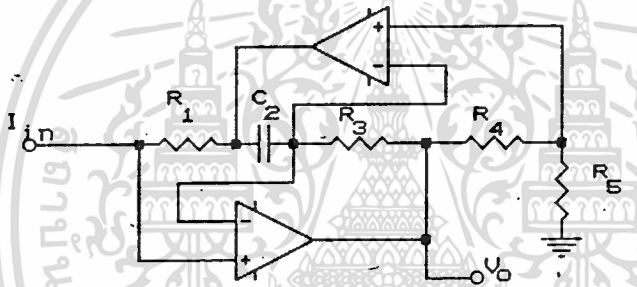
$$\phi_1 = -\phi_2$$

ดังนั้นความถี่ทำงานเสถียรที่เกิดขึ้นจากการล็อกเท่ากับ ω

$$\omega = \frac{\omega_1\omega_2(Q_2 + 1)}{Q_1(Q_2\omega_1 + \omega_2)} \quad \dots(3.12)$$

ในการศึกษาระบบอินเจกชันล็อกกิ้งด้วยทฤษฎีดังกล่าวมาแล้ว จะเห็นว่าข้อจำกัดทางพื้นฐานอยู่ทำให้การออกแบบและทดลองทำได้ยาก แม้จะสามารถนำมาปรับปรุงใช้ศึกษาการล็อกแบบเชื่อมต่อยุบรวมกับระบบที่มีเพียง 2 ออสซิลเลเตอร์ได้ก็ตาม แต่ถ้ามีจำนวนออสซิลเลเตอร์ในระบบมากกว่านี้ก็ศึกษาได้ไม่ถ่วงนัก จึงได้นำวิธีการจำลองแบบแบ่งเวลามาทำการศึกษาระบบอินเจกชันล็อกกิ้งทั้งแบบทิศทางเดียวและแบบเชื่อมต่อยุบรวม ซึ่งจะช่วยให้สามารถบอกถึงคุณสมบัติบางประการ เช่น ยานการล็อกและความถี่ขณะ เกิดการล็อกของออสซิลเลเตอร์ได้

จากรูป 3.4 ออสซิลเลเตอร์ 1 (OSC1) และออสซิลเลเตอร์ 2 (OSC2) เป็นวงจรจูนเดี่ยวนำที่ใช้วงจร Generalized Immittance Converter (GIC) เพื่อแทนตัวเหนี่ยวนำ (Inductor) จริง ทำให้สามารถปรับค่าตัวประกอบคุณภาพและความถี่ได้ตามต้องการ สวิตช์ S_1 และ S_2 ใช้สำหรับเลือกการทดลองอินเจกชันล็อกกิ้งและการล็อกแบบเชื่อมต่อร่วมของออสซิลเลเตอร์ ถ้าให้สวิตช์ S_2 ปิดวงจร (Close circuit) และให้สวิตช์ S_1 เปิดวงจร (Open circuit) จะเป็นการเลือกการทดลองอินเจกชันล็อกกิ้งของออสซิลเลเตอร์ โดย OSC2 จะถูกล็อกด้วยสัญญาณจาก OSC1 ทั้งนี้ในการทดลองสามารถกำหนดค่าของตัวประกอบเชื่อมต่อได้โดยการปรับค่าความต้านทาน R_{cb} สำหรับการทดลองเรื่องการล็อกแบบเชื่อมต่อร่วมจะให้สวิตช์ S_1 และ S_2 ปิดวงจร ซึ่งการเชื่อมต่อจะเกิดโดยผ่านทาง R_{cb} และ R_{ca}



รูป 3.5 วงจร Generalized Immittance converter (GIC)

วงจร GIC ในรูป 3.5 จะตอบสนองต่อสัญญาณเสมือนเป็นตัวเหนี่ยวนำซึ่งมีค่าความเหนี่ยวนำเท่ากับ L

$$L = \frac{C_2 R_1 R_3 R_5}{R_4} \quad \dots (3.13)$$

วงจร GIC ที่ใช้จำลองตัวเหนี่ยวนำ จะถูกต่อขนานกับตัวเก็บประจุ C_1 แล้วทำงานเป็นวงจรเรโซเนเตอร์หรือวงจรผ่านความถี่ ซึ่งเมื่อต่อวงจรขยายป้อนกลับก็จะสามารถทำงานเป็นวงจรออสซิลเลเตอร์ได้ ถ้ามีสภาวะที่

เหมาะสม ในกรณีที่วงจรขยายไม่มีการเลื่อนเฟส ความถี่เรโซแนนซ์ของวงจร ออสซิลเลเตอร์จะมีค่าเท่ากับ ω_0

$$\begin{aligned}\omega_0 &= \frac{1}{(LC_1)^{1/2}} \\ &= \left\{ \frac{R_4}{C_1 C_2 R_1 R_3 R_5} \right\}^{1/2} \quad \dots (3.14)\end{aligned}$$

ให้ตัวต้านทาน R_7 ต่ออนุกรมกับตัวเก็บประจุที่ต่อขนานกับวงจร GIC แล้ว ตัวประกอบคุณภาพของวงจรจะมีค่า

$$Q = \omega_0 C_1 R_7 = R_7 \left\{ \frac{C_1 R_4}{C_2 R_1 R_3 R_5} \right\}^{1/2} \quad \dots (3.15)$$

การปรับค่าความถี่เรโซแนนซ์และค่าตัวประกอบคุณภาพของวงจร จะสามารถปรับได้ โดยการปรับค่าความต้านทานหรือตัวเก็บประจุในสมการ 3.14 และ 3.15 ที่ความถี่เรโซแนนซ์ของเนทเวิร์คแบบจูนเดียวจะมีอัตราขยายของ เนทเวิร์คเท่ากับ 1.0 ดังนั้นเมื่อต้องการให้เกิดการออสซิลเลตจะต้องมีอัตรา ขยายวนรอบ (Loop gain) เท่ากับ 1.0 นั่นคือวงจรขยายจะต้องมีอัตราขยาย เท่ากับ 1.0 ซึ่งจากวงจรในรูป 3.4 ที่เป็นวงจรจริงสำหรับการทดลองจะมี อัตราขยายอยู่ในส่วนของวงจร GIC และวงจรขยายซึ่งถูกควบคุมด้วยเอาต์พุต จากอินทิเกรเตอร์มีอัตราขยายไม่น้อยกว่า 1.0 โดยขณะที่ออสซิลเลเตอร์อยู่ใน สถานะอยู่ตัวนั้น จะมีอัตราขยายวนรอบเท่ากับ 1.0 ดังนั้นจึงต้องใช้ตัวลดทอน สัญญาณ (Attenuator) เพื่อลดขนาดสัญญาณก่อนที่จะเข้าสู่ส่วนของวงจรจูน โดยอัตราการลดทอนสัญญาณมีค่าเท่ากับ 0.1 และอัตราขยายที่ความถี่เรโซแนนซ์ ของวงจรจูนซึ่งใช้วงจร GIC แทนตัวเหนี่ยวนำมีเท่ากับ 2.0 ดังนั้นอัตราขยาย รวมของสองส่วนนี้จึงเท่ากับ 0.2 เพื่อที่จะให้ได้อัตราขยายวนรอบมีค่าเป็น 1.0 จึงต้องให้ส่วนขยายสัญญาณมีอัตราขยายเท่ากับ 5.0 ที่สถานะอยู่ตัว

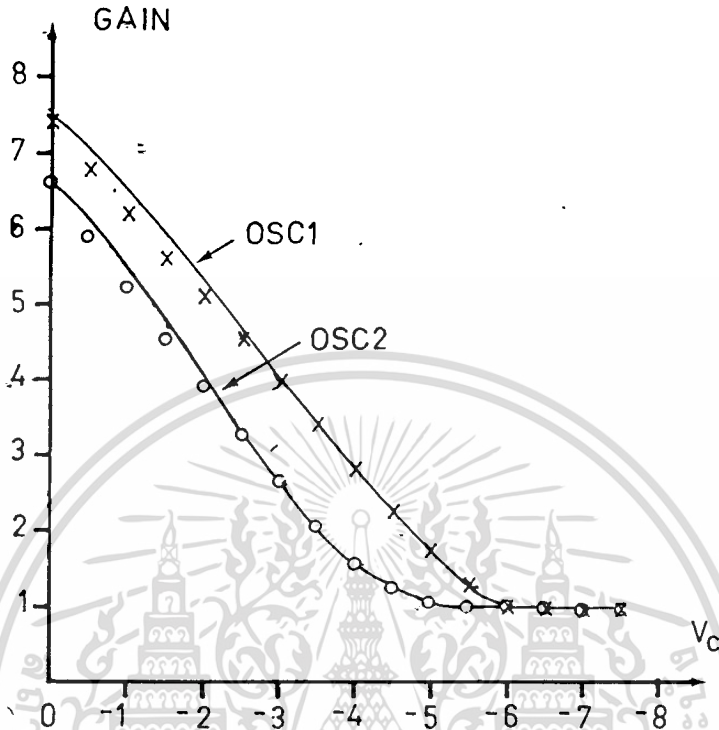
สำหรับส่วนวงจรควบคุมอัตราขยายอัตโนมัติของออสซิลเลเตอร์ OSC1 ในรูป 3.4 จะสามารถหาความสัมพันธ์ของอัตราขยายต่อโวลเตจที่อินพุต ได้ ด้วยวิธีเดียวกับในบทที่ 2.4 ซึ่งความสัมพันธ์ดังกล่าวเขียนในรูปของสมการ โพลีโนเมียลลำดับ 3 (Polynomial order 3) ดังนี้

$$A_1 = 7.35 + 0.87V_c - 0.14V_c^2 - 0.08V_c^3 \quad \dots (3.16)$$

$V_c(t)$ เป็นระดับโวลเตจที่มาควบคุมอัตราขยายของวงจรขยาย ซึ่ง ได้จากการหาโวลเตจเฉลี่ยของสัญญาณเอาต์พุตจากออสซิลเลเตอร์ OSC1 ที่ผ่าน วงจรจัดเรียงกระแสเต็มรูปแบบสัญญาณแล้วผ่านวงจรอินทิเกรเตอร์ ในการจำลอง การทำงานของวงจรนั้น ออสซิลเลเตอร์ OSC2 จะมีสมการโพลีโนเมียลที่แสดง

ความสัมพันธ์ของสัญญาณความคมต่ออัตราขยายคงสมการ 3.17

$$A_2 = 6.7 \pm 1.58V_c + 0.05V_c^2 - 0.01V_c^3 \quad \dots(3.17)$$



รูป 3.6. กราฟแสดงความสัมพันธ์ระหว่างอัตราขยายของวงจรขยายกับโวลเตจควบคุมจากการทดลองและการทำ Polynomial Curve fitting

รูป 3.6 แสดงความสัมพันธ์ของอัตราขยายต่อโวลเตจที่ควบคุมของออสซิลเลเตอร์ทั้งสองตัวทั้งจากการทดลองและการคำนวณจากสมการ 3.16, 3.17 ซึ่งแสดงให้เห็นว่าค่าจากการทดลองและจากการคำนวณตามสมการ 3.16, 3.17 มีค่าใกล้เคียงกัน

จากวงจรการทดลองรูป 3.4 ให้สัญญาณเอาต์พุตจากวงจรออสซิลเลเตอร์ OSC2 ผ่านวงจรจัดเรียงกระแสเต็มคลื่น โดยจะจัดเรียงให้สัญญาณออกจากวงจรจัดเรียงกระแสเต็มรูปแบบสัญญาณเปลี่ยนไป ทำให้มีเฉพาะสัญญาณทางซิกลอป จากนั้นสัญญาณจะไปเข้าวงจรอินทิเกรเตอร์แบบลำดับที่ 1 (First order) ที่มีค่าคงตัวเวลา (Time constant) เท่ากับ τ ซึ่งจากค่าอุปกรณ์ที่ใช้ในการทดลองจริงนั้นจะมีค่าคงตัวเวลาเท่ากับ

$$\tau = RC = 10^4 \cdot 10^{-7} = 10^{-3} \text{ วินาที} \quad \dots(3.18)$$

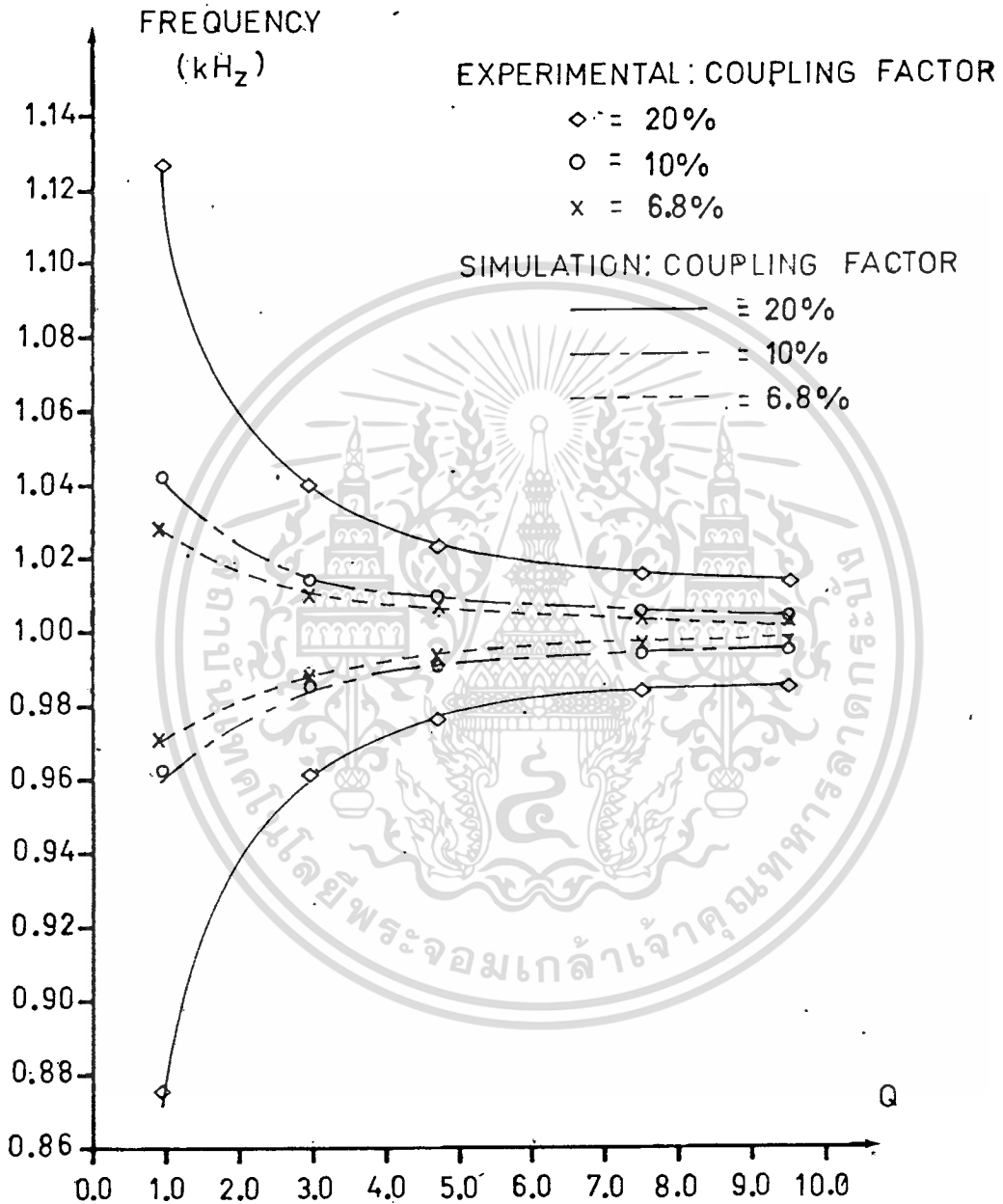
โวลเตจที่ได้จากการเฉลี่ยของสัญญาณเอาต์พุตจากวงจรอินทิเกรเตอร์ จะนำมาใช้ควบคุมอัตราขยายของวงจรขยายเหมือนกับาทดลองในบทที่ 2 จากวงจรการทดลองในรูป 3.4 จะทำการทดลองเพื่อศึกษาปรากฏการณ์ต่าง ๆ ดังนี้

การทดลองที่ 1

การทดลองห่า่านการลือกของออสซิลเลเตอร์ แบบทิสทางเคียว

จากรูปวงจรการทดลอง 3.4 ให้สวิตช์ S_2 ปิดวงจร (Close circuit) และสวิตช์ S_1 เป็ดวงจร (Open circuit) เพื่อกาการทดลองอิน-เจกชั่นลือกกึ่ง การทดลองเริ่มจากให้ OSC1 และ OSC2 เกิดการออสซิลเลต โดยที่ส่วนวงจรจุนของออสซิลเลเตอร์แต่ละตัวมีค่าตัวประกอบคณภาพเท่ากับ Q_1 และ Q_2 ตามลำดับ ซึ่งค่า Q_1 และค่า Q_2 นี้จะมีการปรับเปลี่ยนในระหว่าง การทดลอง ก่อนที่จะให้สวิตช์ S_2 ปิดวงจร ให้รับความถี่ของออสซิลเลเตอร์ OSC2 เพื่อให้สัญญาณเอาต์พุตมีความถี่ 1 กิโลเฮิรตซ์ จากนั้นปรับตัวค่านทาน R_{cb} ในวงจร ให้สัญญาณจากออสซิลเลเตอร์ OSC1 ผ่านไปยังออสซิลเลเตอร์ OSC2 ปรับ R_{cb} ให้มีค่าตัวประกอบเชื่อมต่อเท่ากับ 0.068 ค่าตัวประกอบ เชื่อมต่อเป็นอัตราส่วนของสัญญาณจาก OSC1 ไปยัง OSC2 ต่ออัตราส่วนของ ขนาดสัญญาณเอาต์พุตของ OSC2 ในขณะที่ OSC2 ทำงานเสรี จากนั้นให้สวิตช์ S_2 ปิดวงจร แล้วปรับความถี่การออสซิลเลตของออสซิลเลเตอร์ OSC1 เพื่อหา ความถี่ต่ำสุดและสูงสุดที่ทำให้ออสซิลเลเตอร์ OSC2 ถูกลือกโดยออสซิลเลเตอร์ OSC1 ขณะที่เกิดการลือกสามารถดูได้จากความถี่ของออสซิลเลเตอร์ OSC1 และ OSC2 ซึ่งต้องเป็นความถี่เดียวกัน และขนาดที่จุกบอดของสัญญาณจาก ออสซิลเลเตอร์ OSC2 จะมีขนาดคงที่

ในการทดลองนี้ได้ปรับห่า่านการลือกเมื่อตัวประกอบคณภาพ (Q_2) ของวงจรจุนในออสซิลเลเตอร์ OSC2 มีค่าเป็น 0.93, 2.95, 4.71, 7.5 และ 9.4 ตามลำดับ จากนั้นเปลี่ยนค่าตัวประกอบเชื่อมต่อเป็น 0.068, 0.1 และ 0.2 ทำการทดลองห่า่านลือกที่ตัวประกอบคณภาพ (Q_2) มีค่าดังข้างต้นตาม ลำดับ เพื่อหาความสัมพันธ์ระหว่าง่านลือกกับค่าตัวประกอบคณภาพและตัวประกอบเชื่อมต่อซึ่งได้ผลการทดลองดังรูป 3.7 ที่แสดงให้เห็นว่าแต่ละค่าของ สัมประสิทธิการเชื่อมต่อจะมีเส้นความถี่สูงสุดและต่ำสุดของ่านลือก ในช่วงของความถี่นั้นจึงจะเกิดการลือกที่ออสซิลเลเตอร์ OSC2 ด้วย OSC1



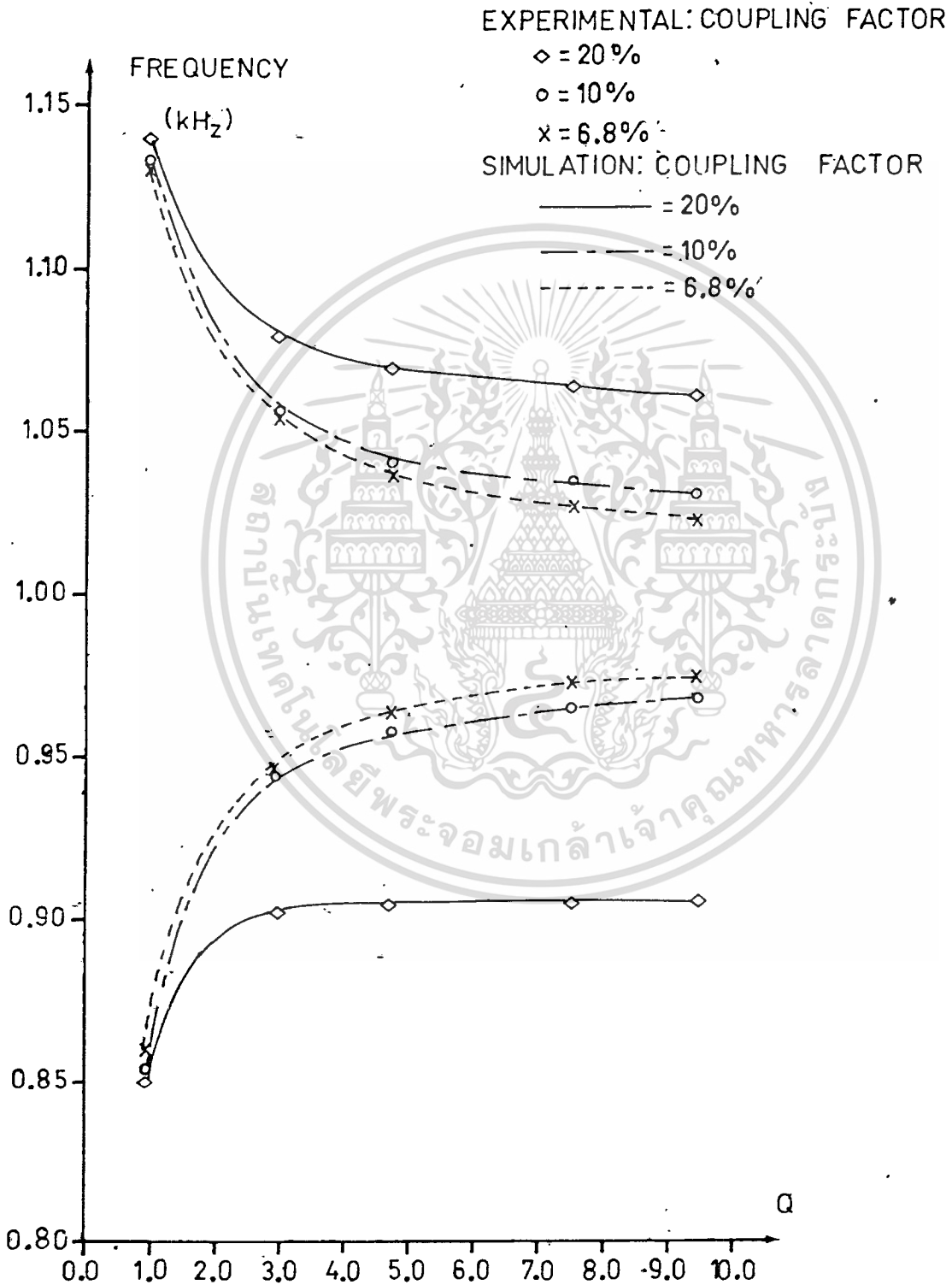
รูป 3.7 ผลการทดลองและการจำลองเรอองอินเจกชันลือกกึ่ง

การทดลองที่ 2

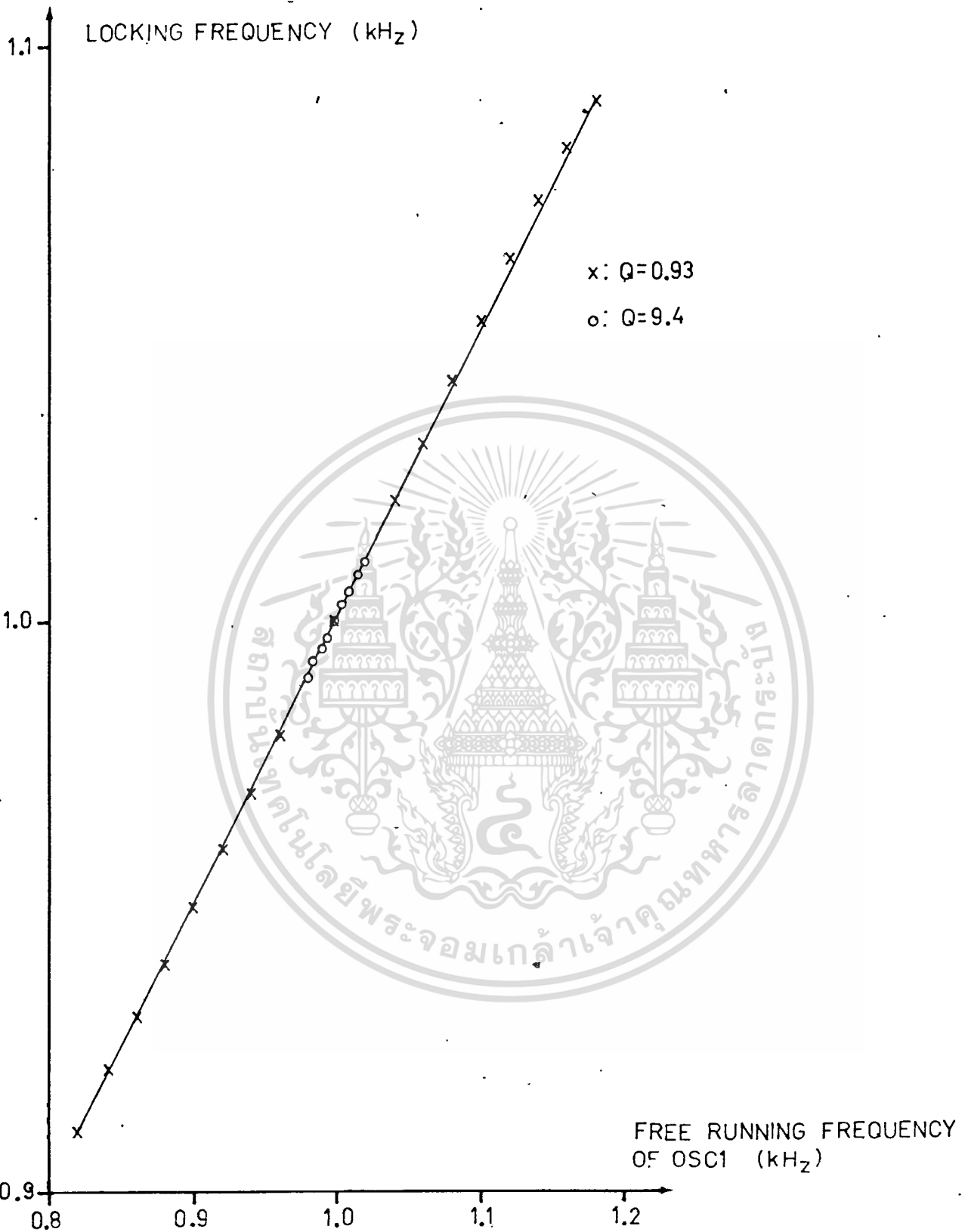
การทดลองการล็อกแบบเชื่อมต่อยวระหว่างออสซิลเลเตอร์ 2 ตัว

การทดลองนี้เป็นการศึกษาการล็อกระหว่างออสซิลเลเตอร์ 2 ตัว โดยที่สัญญาณจากออสซิลเลเตอร์จะล็อกซึ่งกันและกัน ซึ่งในการทดลองดังรูป 3.4 ก่อนที่สวิทช์ S_1 และ S_2 จะปิดวงจร ให้ตั้งออสซิลเลเตอร์ OSC2 มีความถี่ทำงานเสรีคงที่ไว้ที่ 1 กิโลเฮิรตซ์ จากนั้นให้สวิทช์ S_1 และ S_2 ปิดวงจร แล้วปรับความถี่ของออสซิลเลเตอร์ OSC1 เพื่อหาบ้านล็อกและความถี่ขณะที่เกิดการล็อกโดยทำการทดลองที่ออสซิลเลเตอร์ OSC2 มีค่าตัวประกอบคุณภาพของวงจร จนในออสซิลเลเตอร์ OSC2 มีค่าเป็น 0.93, 2.95, 4.71, 7.5 และ 9.4 โดยออสซิลเลเตอร์ OSC1 มีค่าตัวประกอบคุณภาพคงที่เท่ากับ 0.93 และตัวประกอบเชื่อมต้อมีค่าเท่ากับ 0.068, 0.1 และ 0.2 ทั้งจากออสซิลเลเตอร์ OSC2 ไปยังออสซิลเลเตอร์ OSC1 และจากออสซิลเลเตอร์ OSC1 ไปยังออสซิลเลเตอร์ OSC2 ขณะที่เกิดการล็อกนั้นความถี่ออสซิลเลเตอร์ทั้งสองจะเท่ากัน โดยเป็นความถี่ระหว่างความถี่ทำงานเสรีของออสซิลเลเตอร์ทั้งสองและขนาดขดคลื่นของสัญญาณแต่ละไซเคิลของออสซิลเลเตอร์แต่ละตัวจะมีขนาดคงที่ รูป 3.8 เป็นผลการทดลองที่มีค่าสัมประสิทธิ์การเชื่อมต่อจากออสซิลเลเตอร์ตัวหนึ่งไปยังอีกตัวหนึ่งมีค่าเท่ากัน แต่ละค่าของสัมประสิทธิ์การเชื่อมต่อจะมีเส้นความถี่สูงสุดและต่ำสุดของบ้านล็อก ในช่วงความถี่นั้นจึงจะเกิดการล็อก

ในการทดลองที่ 2 จะบันทึกค่าความถี่ของสัญญาณเอาต์พุตจากระบบไว้ด้วย ซึ่งจะเห็นว่าค่าความถี่มีค่าระหว่างความถี่ทำงานเสรีของออสซิลเลเตอร์ทั้งสองตัวดังรูป 3.9



เอกสารนี้เป็นรูป 3.8 ผลการทดลองและการจำลองเรื่องปรากฏการณ์เชื่อมต่อ
ร่วมในออสซิลเลเตอร์



รูป 3.9 ความถี่ของสัญญาณจากออสซิลเลเตอร์ขณะเกิดการล็อก

3.4 อัลกอริทึมและการจำลองอินเจกชันลือกกึ่ง

ก. การจำลองวงจรจูนเคียว

จากวงจรที่ใช้ทดลองในรูป 3.4 จะแสดงทรานสเฟอร์ฟังก์ชันเฉพาะส่วนวงจรจูนเท่ากับ

$$H(S) = \frac{\frac{S\omega_0}{Q}}{S^2 + \frac{S\omega_0}{Q} + \omega_0^2} \quad \dots(3.19)$$

ค่า ω_0 และ Q จะมีค่าตามสมการ 3.14 และ 3.15 ตามลำดับ จากทรานสเฟอร์ฟังก์ชันในสมการ 3.19 ซึ่งอยู่ในรูป S-domain สามารถเปลี่ยนจาก S-domain ไปยัง Z-domain ได้ด้วยวิธีไบลิเนียร์ทรานสฟอร์ม (Bi-linear transform, Tustin method) [6],[7] โดยให้

$$S = \frac{2(1 - Z^{-1})}{\Delta t(1 + Z^{-1})}$$

ซึ่งวิธีไบลิเนียร์ทรานสฟอร์ม [8] จะทำให้ค่าความถี่ใน S-domain และ Z-domain มีความสัมพันธ์ดังสมการ

$$\omega_s = \frac{2}{\Delta t} \tan(\omega_z \Delta t / 2)$$

เมื่อ ω_s เป็นความถี่เชิงมุมใน S-domain

ω_z เป็นความถี่เชิงมุมใน Z-domain

Δt เป็นคาบเวลาในระบบแบ่งเวลา

จากสมการข้างบนจะเห็นว่าความถี่เชิงมุมใน S-domain และ Z-domain สัมพันธ์กันแบบไม่เชิงเส้น โดยความไม่เชิงเส้นนี้จะมากขึ้นเมื่อค่าของ $\omega_z \Delta t / 2$ มีค่ามากขึ้น ดังนั้นถ้าจะให้ความถี่เชิงมุม ω_s และ ω_z มีค่าใกล้เคียงกันก็ต้องให้ $\omega_z \Delta t / 2$ มีค่าน้อยๆ เช่นถ้าต้องการให้ ω_s และ ω_z มีค่าต่างกันไม่เกิน 1 เปอร์เซ็นต์ จะต้องให้ $\omega_z \Delta t / 2$ มีค่าประมาณไม่เกิน 0.307 ดังนั้น

$$\omega_z \Delta t \ll 0.614$$

จะได้ว่า

$$\frac{\Delta t}{T_2} \ll \frac{0.614}{2\pi}$$

โดยที่ T_2 เป็นคาบเวลา 1 ไซเคิลของความถี่เชิงมุม ω_z

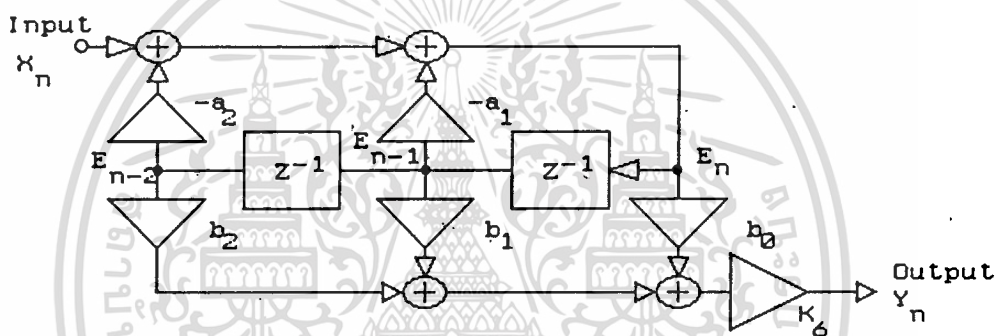
แสดงว่า จะต้องมีความยาวเวลาการจำลอง (Δt) ไม่เกิน $T_z/10$ เพื่อให้ ω_s และ ω_c มีค่าต่างกันไม่เกิน 1 เปอร์เซ็นต์ ซึ่งในการจำลองให้ Δt มีค่าเท่ากับ $T_z/256$ เพื่อจะให้ความสัมพันธ์ระหว่างความถี่ใน S-domain และ Z-domain ประมาณได้เป็นแบบเชิงเส้นตลอดย่านความถี่ที่ใช้ทดลอง

จากสมการ 3.19 จะเขียนใหม่ใน Z-domain ได้ว่า

$$H(Z) = \frac{2\omega_s(1 - Z^{-2})}{Z^{-2}\{4Q - 2\omega_s\Delta t + Q(\Delta t\omega_s)^2\} + Z^{-1}\{2Q(\Delta t\omega_s)^2 - 8Q\} + \{4Q + 2\omega_s\Delta t + Q(\Delta t\omega_s)^2\}}$$

สามารถเขียนเป็นไคอะแกรมสถานะของการคำนวณแบบเวียนเกิดได้

ผังรูป 3.10



$$a_1 = \frac{K_4}{K_3}$$

$$a_2 = \frac{K_5}{K_3}$$

$$b_0 = 1$$

$$b_1 = 0$$

$$b_2 = -1$$

$$K_1 = 2\omega_s\Delta t$$

$$K_2 = (\Delta t\omega_s)^2 Q$$

$$K_3 = 4Q + 2\omega_s\Delta t + Q(\Delta t\omega_s)^2$$

$$K_4 = -8Q + 2Q(\Delta t\omega_s)^2$$

$$K_5 = 4Q - 2\omega_s\Delta t + Q(\Delta t\omega_s)^2$$

$$K_6 = \frac{2\omega_s\Delta t}{K_3}$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น ผังรูป 3.10 มีไคอะแกรมสถานะการคำนวณแบบเวียนเกิดของวงจรจูนได้

จากไดอะแกรมสถานะของการคำนวณซึ่งอยู่ในรูปของการเวียนเกิดนั้น การคำนวณแต่ละครั้งจะขึ้นกับค่าจากสถานะที่ค่าของการแบ่งเวลาที่ผ่านมาของระบบ ซึ่งสามารถเขียนโปรแกรมคอมพิวเตอร์เพื่อจำลองการทำงานของวงจรได้ต่อไป โดยที่เทอม Z^{-1} เป็นการหน่วงเวลาของข้อมูลเท่ากับค่าของการแบ่งเวลาจากไดอะแกรมข้างบน การคำนวณในแต่ละค่าของการแบ่งเวลามีดังนี้

$$E_n = X_n - a_1 E_{n-1} - a_2 E_{n-2} \quad \dots (3.20a)$$

$$Y_n = K_0(b_0 E_n + b_1 E_{n-1} + b_2 E_{n-2}) \quad \dots (3.20b)$$

n ในสมการ 3.20a และ 3.20b เป็นการบอกเวลาเท่ากับ $n\Delta t$

ข. การจำลองตัวอินทิเกรต

การจำลองตัวอินทิเกรตเป็นวิธีเดียวกับที่ใช้ในบทที่ 2.4 โดยต่างกันที่ค่าของตัวแปรในวงจร

ตัวอินทิเกรตหรือวงจรผ่านความถี่ต่ำมีจุดตัดความถี่เท่ากับ ω_c ซึ่งเมื่อทำการ Normalize จะได้ว่า

$$\omega_n = \frac{\omega_c}{RC\omega_c} = \frac{1}{RC}$$

ω_n ซึ่งเป็นความถี่ที่ถูก Normalize ด้วย ω_c จากวงจรที่ทดลองจะมีค่าเท่ากับ 1.59

RC เป็นค่าคงตัวเวลาของอินทิเกรเตอร์ ซึ่งมีค่าดังที่แสดงในสมการ 3.18

ω_c เป็นความถี่ทำงานเสริมของออสซิลเลเตอร์

ดังนั้นการจำลองส่วนอื่นของวงจร เช่น ส่วนควบคุมอัตราขยายอัตโนมัติจะต้องทำการวัดคุณลักษณะจากวงจรจริงของวงจรควบคุมอัตราขยายอัตโนมัติซึ่งมีคุณสมบัติดังสมการ 3.16 และ 3.17

สำหรับการจำลองที่ 1 ถึง 3 เป็นการแสดงอัลกอริธึมของการจำลองหาค่าความถี่ที่เกิดการล็อก ทั้งแบบทิศทางเดียว และ 2 ทิศทาง รวมทั้งความถี่ของสัญญาณจากออสซิลเลเตอร์ขณะที่เกิดการล็อกด้วย โดยมีการจำลองดังนี้

การจำลองที่ 1

การจำลองที่ 1 เป็นการจำลองการทดลองปรากฏการณ์ของการล๊อคแบบทิศทางเดียว จากวงจรการทดลองในรูปที่ 3.4 เมื่อสวิตช์ S_2 และสวิตช์ S_1 เป็ดวงจร (Open circuit) ออสซิลเลเตอร์ OSC2 จะถูกป้อนด้วยสัญญาณจากออสซิลเลเตอร์ OSC1 ซึ่งเป็นออสซิลเลเตอร์ที่ทำงานอย่างเสรี ในการจำลองจะทำการจำลองออสซิลเลเตอร์ OSC1 เหมือนกับเป็นแหล่งกำเนิดสัญญาณไซน์ (Sine wave generator) ซึ่งจะจำลองการสร้างสัญญาณไซน์ออกจากออสซิลเลเตอร์โดยไม่ต้องจำลองการทำงานภายในของออสซิลเลเตอร์ OSC1 การจำลองสัญญาณไซน์นี้จะทำได้โดยเรียกฟังก์ชัน $SIN(X)$ ภายในภาษา TURBO PASCAL การจำลองที่ 1 นี้จะหาอัตราการล๊อคของออสซิลเลเตอร์ OSC2 เมื่อค่าตัวประกอบคุณภาพ (Q) และสัมประสิทธิ์การเชื่อมต้อมีค่าต่าง ๆ กัน อัลกอริทึมของการจำลองในแต่ละคาบเวลาที่แบ่งจะมีดังนี้

- ขั้นที่ 1 : กำหนดค่าตัวประกอบคุณภาพเท่ากับ 0.93 และสัมประสิทธิ์การเชื่อมต่อเท่ากับ 0.068 คำนวณค่าตัวแปรสำหรับการจำลองไดอะแกรมสถานะของวงจรจูนเคียวในรูป 3.10
- ขั้นที่ 2 : คำนวณเพื่อจำลองตามไดอะแกรมสถานะของวงจรจูนเคียวในรูป 3.10 โดยการคำนวณสมการ 3.20a และ 3.20b โดยที่ X_n เป็นสัญญาณรบกวนระดับต่ำซึ่งสร้างเหมือนกับในบทที่ 2.4 สัญญาณรบกวนระดับต่ำนี้จะช่วยให้ออสซิลเลเตอร์เริ่มต้นการออสซิลเลตเท่านั้น เมื่อการออสซิลเลตเข้าสู่สถานะอยู่ตัวจะหยุดการป้อนสัญญาณรบกวนให้กับวงจรจำลอง เพราะสัญญาณนี้มีค่าน้อยมากเมื่อเทียบกับสัญญาณป้อนกลับในออสซิลเลเตอร์ที่สถานะอยู่ตัว
- ขั้นที่ 3 : จากผลของการคำนวณในขั้นที่ 2 คือ Y_n จะนำไปผ่านวงจรขยายซึ่งมีคุณสมบัติของอัตราขยายตามสมการ 3.17 โดยที่อัตราขยายนี้ขึ้นกับระดับสัญญาณจากวงจรแปลงสัญญาณแบบหาค่าเฉลี่ย การคำนวณระดับสัญญาณจากวงจรแปลงสัญญาณนี้จะทำได้ โดยการคำนวณตามสมการ 2.18 ในบทที่ 2.4.ค $V_c(t)$ จากการคำนวณตามสมการ 2.18 จะนำไปหาอัตราขยายของวงจรขยายในสมการ 3.17 อัตราขยายที่ได้จะนำไปใช้ในการคำนวณขั้นที่ 4
- ขั้นที่ 4 : คำนวณขนาดสัญญาณที่จะป้อนกลับไปยังอินพุตรวมกับสัญญาณรบกวนระดับต่ำ เพื่อจำลองวงจรจูนเคียวในข้อที่ 2 โดยการคูณอัตราขยายจากการคำนวณในขั้นที่ 3 กับสัญญาณเอาต์พุตจากเน็ตเวิร์คที่คำนวณได้ในขั้นที่ 2 และอัตราการลดทอนสัญญาณจริงในวงจร เพื่อให้วงจรขยาย

รวมในวงจรมีค่าใกล้เคียง 1 และวงจรควบคุมอัตราขยายสามารถทำงานได้ดังการทดลองที่ 1 ทำการคำนวณในขั้นที่ 2 ถึง 4 จนกระทั่งออสซิลเลเตอร์เข้าสู่สถานะอยู่ตัว และเอาท์พุทที่มีความถี่ Normalize เท่ากับ 1.0 จากนั้นเข้าสู่ขั้นที่ 5

ขั้นที่ 5 : ทำการจำลองขั้นที่ 2 ถึงขั้นที่ 4 โดยเปลี่ยนสัญญาณรบกวนระดับต่ำเป็นสัญญาณชาบนซึ่งสร้างโดยฟังก์ชัน $SIN(X)$ ใน TURBO PASCAL สัญญาณชาบนนี้มีอัตราส่วนต่อขนาดของสัญญาณจากออสซิลเลเตอร์ในขณะที่ทำงานเสรีเท่ากับค่าสัมประสิทธิ์การเชื่อมต่อ ให้ทำการจำลองขั้นที่ 2 ถึงขั้นที่ 4 ว่าสัญญาณนั้นสามารถล็อกให้ออสซิลเลเตอร์ผลิตสัญญาณความถี่เดียวกับสัญญาณชาบนหรือไม่ ถ้าไม่สามารถล็อกได้ขนาดบอดของสัญญาณเอาท์พุทจะมีขนาดไม่คงที่และความถี่ไม่เท่ากับความถี่ของสัญญาณชาบน ให้หาความถี่สูงสุดและต่ำสุดของสัญญาณชาบนที่ยังสามารถล็อกออสซิลเลเตอร์อยู่ได้โดยวิธี Successive approximation

ขั้นที่ 6 : เปลี่ยนค่าตัวประกอบคุณภาพของวงจรจูนใน OSC2 เป็น 2.95, 4.97, 7.5 และ 9.4 ตามลำดับ ทำการจำลองในขั้นที่ 1 ถึงขั้นที่ 5 เพื่อหาว่าการล็อกที่ค่าสัมประสิทธิ์การเชื่อมต่อค่าหนึ่ง แต่ค่าตัวประกอบคุณภาพค่าต่างๆกัน บันทึกบอดการล็อกไว้

ขั้นที่ 7 : เปลี่ยนค่าสัมประสิทธิ์การเชื่อมต่อเป็น 0.1 และ 0.2 ตามลำดับ ทำการจำลองขั้นที่ 1 ถึงขั้นที่ 6 ที่ค่าสัมประสิทธิ์การเชื่อมต่อข้างต้น ผลการจำลองปรากฏดังรูป 3.7

วิธี Successive Approximation ในขั้นที่ 5 จะเริ่มต้นจากการกำหนดว่าย่านการล็อกอยู่ระหว่างความถี่ทำงานเสรี (ω_1) และความถี่สูง (ω_2) พอที่จะไม่ทำให้เกิดการล็อก ทำการจำลองครั้งที่ 1 ตั้งแต่ขั้นที่ 2 ถึง 5 โดยให้สัญญาณชาบนมีความถี่เท่ากับ $(\omega_1 + \omega_2) / 2$ จะเกิดผลลัพธ์ได้ 2 กรณี

1. กรณีไม่เกิดการล็อก

ในการจำลองครั้งต่อไปจะให้ $\omega_2 = (\omega_1 + \omega_2) / 2$

2. กรณีเกิดการล็อก

ในการจำลองครั้งต่อไปจะให้ $\omega_1 = (\omega_1 + \omega_2) / 2$

และครั้งต่อไปจะทำการจำลองข้อที่ 1 ถึงข้อที่ 5 โดยให้สัญญาณชาบนมีความถี่เท่ากับ $(\omega_1 + \omega_2) / 2$ ทำซ้ำกัน 16 ครั้ง ก็จะทราบย่านการล็อก โดยมีความละเอียดของย่านการล็อกจะสูงถึง $(\omega_2 - \omega_1) / 2^{16}$ โดย ω_2 และ ω_1 เป็นความถี่ในตอนเริ่มต้นหาย่านการล็อก

การจำลองที่ 2

การจำลองการล็อกแบบเชื่อมต่อร่วมในวงจรออสซิลเลเตอร์แบบจูนเดี่ยว โดยการจำลองการเกิดปรากฏการณ์เชื่อมต่อร่วมระหว่างออสซิลเลเตอร์ 2 ตัว เป็นการจำลองเพื่อเปรียบเทียบกับกรทดลองที่ 2 ซึ่งอัลกอริทึมของการจำลองในแต่ละคาบการแบ่งเวลามีดังนี้

ขั้นที่ 1 : สร้างตัวแปรของออสซิลเลเตอร์แต่ละตัว ตามโคอะแกรมรูปภาพในรูป 3.10 โดยออสซิลเลเตอร์ OSC2 มีความถี่ทำงานเสรีเท่ากับ 1.0 และความถี่ทำงานเสรีของออสซิลเลเตอร์ OSC1 มีความถี่ต่ำกว่าความถี่ที่จะเกิดการล็อกกับ OSC2 จากนั้นคำนวณตามขั้นที่ 2 ถึงขั้นที่ 4 ของการจำลองที่ 1 จนกระทั่งออสซิลเลเตอร์ทั้งสองเข้าสู่สถานะอยู่ตัว

ขั้นที่ 2 : คำนวณตามขั้นที่ 2 ถึงขั้นที่ 4 ของการจำลองที่ 1 แต่เปลี่ยนสัญญาณรบกวนระดับต่ำเป็นสัญญาณจากออสซิลเลเตอร์อีกตัวหนึ่ง ด้วยสัมประสิทธิ์การเชื่อมต่อ คำนวณซ้ำจนกระทั่งออสซิลเลเตอร์ทั้งสองเข้าสู่สถานะอยู่ตัว บันทึกค่าความถี่ของออสซิลเลเตอร์ทั้งสอง ซึ่งถ้าเกิดการล็อก ความถี่ทั้งสองจะมีค่าเท่ากัน

ขั้นที่ 3 : เปลี่ยนความถี่ทำงานเสรีของออสซิลเลเตอร์ โดยเปลี่ยนตัวแปรในโคอะแกรมสถานะในรูป 3.10 เสียใหม่ แล้วคำนวณในขั้นที่ 1 ถึง 2 ใหม่ เพิ่มความถี่ทำงานเสรีของออสซิลเลเตอร์ OSC1 จนกระทั่งสูงกว่าความถี่สูงสุดที่จะเกิดการล็อกได้

ขั้นที่ 4 : เปลี่ยนค่าตัวประกอบคุณภาพของวงจรจูนในออสซิลเลเตอร์ OSC2 และสัมประสิทธิ์การเชื่อมต่อของออสซิลเลเตอร์ทั้งสองตามการทดลองที่ 2 แล้วทำการคำนวณซ้ำในขั้นที่ 1 ถึงขั้นที่ 3

ผลการจำลองหาบ่านการล็อกจะมีดังรูป 3.8 และความถี่ของออสซิลเลเตอร์ทั้งสองขณะที่เกิดการล็อกจะมีดังรูป 3.9

3.5 เอกสารอ้างอิงบทที่ 3

- [1] Adler R., "A Study of Locking Phenomena in Oscillators," Proc.IRE, Vol.34, pp.351-357, June 1946.
- [2] Paciorek L.J., "Injection Locking of Oscillators," Proc.IEEE, Vol.53, No.11, pp.1723-1727, Nov. 1965.
- [3] Biswas B.N. and Ray S.K., "Descrimination of a Second-order Injection Synchronized Oscillator Against Interfering Tones," IEEE Trans. Circuit Syst., Vol.CAS-21, No.3, pp.402-405, May 1974.
- [4] Jezewski M.T., "An Approach to the Analysis of Injection-Locked Oscillators," IEEE Trans. Circuit Syst., Vol.CAS-21, No.3, pp.395-401, May 1974.
- [5] Uzunoglu V. and White M.H., "The Synchronous Oscillator: A Synchronization and Tracking Network," IEEE Jour. Solid-state Circuit, Vol.SC-20, No.6, pp.1214-1225, Dec. 1985.
- [6] Sage A.P. and Smith S.L., "Real-Time Digital Simulation for System Control," Proc. IEEE, Vol.54, No.12, pp.1802-1812, Dec. 1966.
- [7] Sarkar T.K., Radhakrishna N. and Chen H., "Survey of Various Z-Domain to S-Domain Transformations," IEEE Trans. Instr. Measur., Vol.IM-35, No.4, pp.508-520, Dec. 1986.
- [8] Ogata K., "Discrete Time Control Systems," Prentice-Hall International Edition, 1987.
- [9] Biswas B.N., "Phase Lock Theories and Applications," Oxford & IBH Publishing Co.PVT.Ltd., Bombay, 1985.

บทที่ 4

การจำลองระบบดีเลย์ไลน์ด้วยวิธีแบ่งเวลา

(Discrete time domain simulation of delay line)

4.1 บทนำการจำลองระบบดีเลย์ไลน์ด้วยวิธีแบ่งเวลา

ปรากฏการณ์ของคลื่นบนพื้นผิว (Surface acoustic wave, SAW) เป็นปรากฏการณ์ที่คลื่นเคลื่อนที่ไปบนผิวของสารพิโซอิเล็กทริก (Piezoelectric) ซึ่งได้มีการเสนอครั้งแรกโดย Lord Rayleigh ในปี ค.ศ.1885 แต่ไม่ได้ถูกนำไปใช้งานจนกระทั่งปี ค.ศ.1965 จึงมีการนำเอาคุณสมบัติการเคลื่อนที่ไปบนพื้นผิวดังกล่าวซึ่งเรียกว่า Rayleigh wave ไปใช้เป็นวงจรกรองความถี่ (Filter) และใช้ในการวิเคราะห์สัญญาณแอนะล็อก (Analog signal processing) [1] โดยการส่งสัญญาณไฟฟ้าไปกระตุ้นฟิล์มโลหะ (Metal-film) ซึ่งทำเป็นอินเตอร์ดิจิตอลทรานสดิวเซอร์ (Inter digital transducer, IDT) บนผิวของสารพิโซอิเล็กทริก ปัจจุบันอุปกรณ์ประเภท SAW ได้พัฒนาขึ้น แต่ก็โดยคุณสมบัติพื้นฐานในการเก็บพลังงานหรือข้อมูลไว้ในช่วงเวลาหนึ่ง ซึ่งเป็นการเคลื่อนที่อย่างช้าๆ ไปบนผิวของอุปกรณ์ SAW

ปัจจุบันได้มีการศึกษาผลตอบสนองต่อ SAW ของอินเตอร์ดิจิตอลทรานสดิวเซอร์ โดยการใช่วงจรเสมือน (Equivalent circuit) [2] เช่นการจำลองวงจรเพื่อศึกษาผลตอบสนองชั่วขณะของอินเตอร์ดิจิตอลทรานสดิวเซอร์ [3] หรือการใช่วงจรเสมือนในการวิเคราะห์และออกแบบคิสมิเตอร์ซีฟอนเตอร์ดิจิตอลทรานสดิวเซอร์ [4] แต่ก็มีควมยุ่งยาก

วิทยานิพนธ์นี้จะเสนอการจำลองระบบป้อนกลับแบบไม่เชิงเส้นด้วยวิธีแบ่งเวลา เช่นเกี่ยวกับการจำลองวงจรในบทที่ 2 และบทที่ 3 แต่ในระบบที่ศึกษาจะมีอุปกรณ์หน่วงเวลาประกอบอยู่ด้วย เพื่อศึกษาคุณสมบัติบางประการของระบบซึ่งประกอบด้วยตัวหน่วงเวลา เช่นการกระโดดความถี่ของออสซิลเลเตอร์จากโหมดหนึ่งไปยังอีกโหมดหนึ่งอันเป็นปรากฏการณ์ที่เกิดขึ้นในช่วงเวลาสั้น ซึ่งจะทำให้สามารถใช้วิธีการจำลองระบบดังกล่าวเพื่อการศึกษาคุณสมบัติอื่นๆต่อไป นอกจากนี้ได้เสนออัลกอริธึมที่สามารถช่วยลดเวลาในการคำนวณ ทำให้ได้ผลลัพธ์จากการจำลองรวดเร็วขึ้น

4.2 ออสซิลเลเตอร์แบบคิลีย์ไลน์

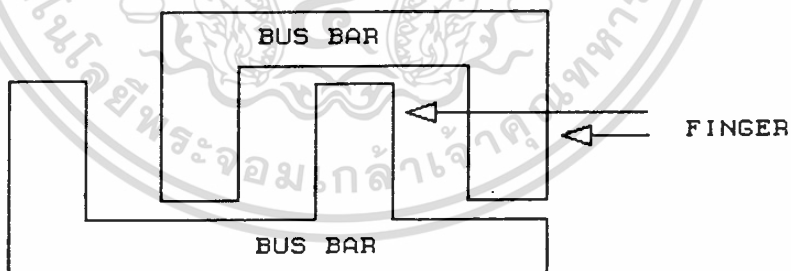
อุปกรณ์ SAW ประกอบด้วย 2 ส่วนพื้นฐานคือ

1. Piezoelectric substrate ได้แก่ สารพวก LiNbO_3 , GaAs, Quartz ผิวจะเรียบ เมื่อมีความเค้นที่ผิวของสารเกิดขึ้นเนื่องจากสนามไฟฟ้า พลังงานของความเค้นนี้จะลดลงเป็นเอกโพเนนเชียล (Exponential) ตามความลึกลงไปใ้ในเนื้อของสารพิโซอิเล็กทริก ความเค้นนี้จะทำให้เกิดคลื่นเคลื่อนที่ไปในเนื้อสารพิโซอิเล็กทริกเรียกว่า Shallow-bulk acoustic wave และความเค้นที่ผิวของสารพิโซอิเล็กทริก จะทำให้เกิดคลื่นเคลื่อนที่ไปบนผิวเรียกว่า Surface wave หรือ Rayleigh wave ซึ่งวิทยานิพนธ์ฉบับนี้จะทำการศึกษาเรื่องของ Rayleigh wave

2. IDT (Inter digital transducer) เป็นโลหะ ที่ใช้วิธี Photo-lithographically สร้างผิวโลหะรูปคล้ายฟันหึ่งลงบนผิวของสารพิโซอิเล็กทริก ซึ่ง IDT นี้สามารถใช้เป็นได้ทั้งตัวส่งและตัวรับ โดยที่ IDT ตัวส่งจะใช้ในการเปลี่ยนสัญญาณไฟฟ้าเป็นพลังงานคลื่น และ IDT ตัวรับจะเปลี่ยนพลังงานคลื่นเป็นพลังงานไฟฟ้า

IDT ประกอบด้วยฟิงเกอร์ (Finger) ต่ออยู่กับบัสบาร์ (Bus bar)

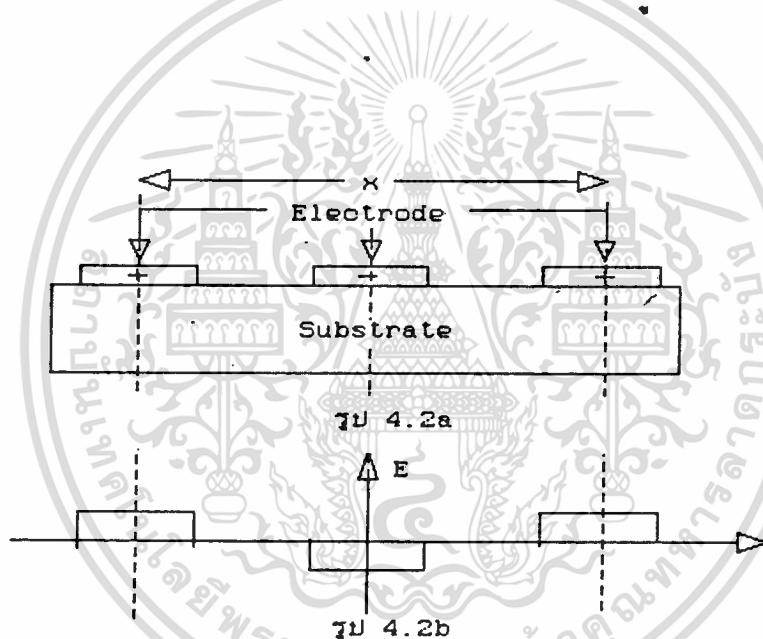
ดังรูป 4.1



รูป 4.1 ฟิงเกอร์ต่ออยู่กับบัสบาร์

จากรูป 4.1 ซึ่งแสดงลักษณะของ IDT นั้น การเหลื่อมกัน (Overlap) ของฟิงเกอร์ที่ต่อกับแต่ละขั้วสับบาร์ อาจมีค่าแตกต่างกันในแต่ละคู่ฟิงเกอร์ (Finger pair) โดยขึ้นกับความต้องการในการออกแบบ ซึ่งค่าสูงสุดของการเหลื่อมกันเรียกว่าอเพอร์เจอร์ (Aperture) และการเหลื่อมกันของแต่ละคู่ฟิงเกอร์ในแต่ละตอน (Section) มีค่าไม่เท่ากัน ทำให้พลังงานที่ส่งและรับในแต่ละตอนไม่เท่ากัน เรียกว่าการถ่วงน้ำหนัก (Weighting) หรือการอะโพไดซ์ (Apodization) ซึ่งเป็นส่วนสำคัญมากในการออกแบบอุปกรณ์ SAW เช่นวงจรกรองความถี่หรือตัวหน่วงสัญญาณ (Delay element) ทั้งนี้เพราะความกว้าง, จำนวนและการเหลื่อมกันของฟิงเกอร์จะมีผลต่อการตอบสนองของสัญญาณอิมพัลส์ (Impulse signal) ที่ป้อนเข้าไปใน IDT ด้วย

ที่ IDT ตัวส่ง เมื่อเราป้อนสัญญาณไฟฟ้าไปที่อิเล็กโทรด (Electrode) จะทำให้เกิดสนามไฟฟ้าระหว่างอิเล็กโทรดดังรูป 4.2 a.



รูป 4.2 สนามไฟฟ้าที่อิเล็กโทรด

ในรูป 4.2a แสดงถึง IDT ตัวส่ง โดยที่ระยะห่างระหว่างแต่ละตอน ประกอบด้วย 1 คู่ฟิงเกอร์อาจเท่ากันหรือไม่เท่ากันก็ได้ ขึ้นกับการออกแบบ ถ้า ระยะห่างแต่ละตอนเท่ากัน เรียกว่านอนดิสเพอร์ซีฟดีเลย์ไลน์ (Non-dispersive delay line)

ส่วนดิสเพอร์ซีฟดีเลย์ไลน์ (Dispersive delay line) นั้น จะ มีความกว้างของแต่ละตอนไม่เท่ากัน เฟสระหว่างสัญญาณเข้าและออกจากดีเลย์ ไลน์จะขึ้นกับความถี่และความยาวของดีเลย์ไลน์ ดังสมการที่ 4.1

$$\varnothing = \frac{2\pi L}{\lambda} \quad \dots (4.1)$$

กำหนดให้ L เป็นความยาวของดีเลย์ไลน์
 λ เป็นความยาวคลื่น

การหาสมการขอบ (Boundary equation) ของสนามไฟฟ้าในรูป 4.2b จะอยู่ในรูปของสมการ 2 มิติ (Dimension) และขึ้นกับโครงสร้างของ สารพีโซอิเล็กทริกด้วย ทำให้การศึกษาค้นคว้าจะซับซ้อน [5]

การพิจารณาสนามไฟฟ้าที่ผิวของสารพีโซอิเล็กทริกจากรูป 4.2b สามารถพิจารณาด้วยวิธี Electrostatic approximation โดยการหาลา- ปลาซทรานส์ฟอร์ม (Laplace transform) ของศักดาไฟฟ้า (Electric potential) ในสนามที่เกิดขึ้นที่ผิวของสารพีโซอิเล็กทริก ซึ่งกำหนดให้สัมประ- สติการเชื่อมต่อพีโซอิเล็กทริก (Piezoelectric coupling coefficient) มีค่าต่ำ ดังนั้นสนามไฟฟ้ามีค่าคงที่ในบริเวณอิเล็กโทรดและเป็นศูนย์ในบริเวณ ช่องว่างระหว่างอิเล็กโทรด (Interelectrode spacing) IDT จะมีประ- สติภาพสูงสุด เมื่อระยะห่างของทรานสดิวเซอร์ (Transducer) ซึ่งมีค่าเท่า- กับ λ ตามรูป 4.2a เป็นจำนวนเท่าของความยาวคลื่น และถ้าฟิงเกอร์มีอัตรา ส่วนของความกว้างระหว่างโลหะกับช่องว่างเท่ากับ 1.0 IDT จะทำงานได้ดี เมื่อความยาวคลื่นเป็นฮาร์โมนิกเลขคี่ (Odd harmonic) ของความกว้าง IDT

$$\frac{\lambda_0}{\lambda} = \frac{f}{f_0} = 2P + 1 \quad ; P = 0, 1, 2, 3, \dots \quad \dots (4.2)$$

เมื่อ λ_0 เป็นความยาวของ IDT 1 ตอน (Section)

f_0 เป็นความถี่ของคลื่นที่มีความยาวคลื่น λ_0

f เป็นความถี่ของคลื่นที่เข้าสู่ IDT

λ เป็นความยาวคลื่นที่เข้าสู่ IDT

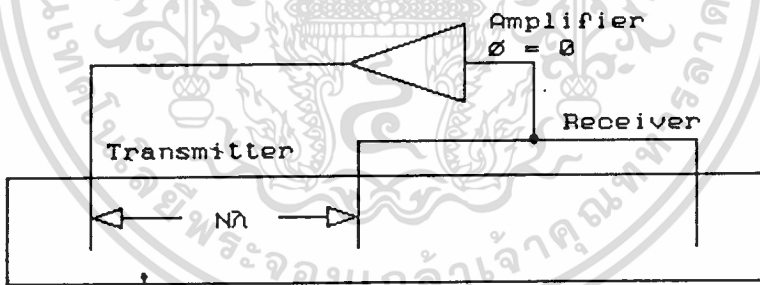
จากวิธีการประมาณ จะให้สนามไฟฟ้ามีค่าคงที่ตามวิธี Electrostatic approximation ทำให้สามารถทนแต่ละตอนได้ด้วยวิธีการประมาณลำดับที่ 1 (First order approximation) ดังรูป 4.2b ในการเคลื่อนที่ของคลื่นออกจาก IDT ตัวส่ง จะมีการเคลื่อนที่ไปทั้ง 2 ทิศทางที่ตรงข้ามกัน ขณะที่คลื่นเคลื่อนที่ไปนั้นจะเกิดการสูญเสียของพลังงานบ้าง [6] ได้แก่ การสูญเสียแพร่กระจาย (Propagation loss) ซึ่งเป็นการสูญเสียขณะที่เกิดการเคลื่อนที่ของคลื่นไปบนผิวของสารพีโซอิเล็กทริก และการสูญเสียที่ทรานสดิวเซอร์ การสูญเสียนั้นมักจะบอกเป็นอัตราส่วนของพลังงานที่ส่งไปที่ทรานสดิวเซอร์ตัวส่ง ต่อพลังงานที่รับมาได้ที่ทรานสดิวเซอร์ตัวรับ ในหน่วยเดซิเบล (Decibel, dB) เรียกว่า Acoustic propagation loss และ Acoustic propagation loss จะเป็นตัวกำหนดย่านความถี่ใช้งานของอุปกรณ์ SAW โดยที่ปัจจุบันอุปกรณ์ SAW มีย่านการใช้งานระหว่าง 10MHz ถึง 11GHz ขณะที่คลื่นเคลื่อนที่ไปนั้น พลังงานของคลื่นจะเคลื่อนที่ในทิศทางที่ทำมุมกับทิศทางการเคลื่อนที่ของคลื่นค่าหนึ่ง เรียกว่า Power-flow angle และ พลังงานของคลื่น 50% จะอยู่ในช่วงของ Beamwidth

เมื่อคลื่นเคลื่อนที่ถึง IDT ตัวรับ พลังงานคลื่นจะทำให้เกิดความเค้นที่ผิวของสารพีโซอิเล็กทริก ทำให้เกิดสนามไฟฟ้าและจะถูกลัดวงจรให้มีศักดาไฟฟ้าที่ทุกจุดบนอิเล็กโทรดของ IDT เท่ากัน [7] มีผลทำให้ความเร็วของคลื่นบนโลหะลดลง เพราะเป็นคุณสมบัติของคลื่นเมื่อมีการเคลื่อนที่จากตัวกลางหนึ่งไปยังอีกตัวกลางหนึ่ง พลังงานของคลื่นบางส่วนจะสะท้อนกลับ ซึ่งเกิดขึ้นใน IDT ตัวส่งด้วย เมื่อคลื่นเคลื่อนที่จากตอนหนึ่งไปยังอีกตอนหนึ่งในทั้ง 2 ทิศทาง คลื่นจะเคลื่อนที่ผ่านอิเล็กโทรดและช่องว่างระหว่างอิเล็กโทรด ขณะที่คลื่นเคลื่อนที่จากตัวกลางหนึ่งไปยังอีกตัวกลางหนึ่งนั้น คลื่นส่วนหนึ่งก็จะสะท้อนกลับในทางตรงกันข้ามและคลื่นส่วนที่เหลือก็จะเดินทางต่อไป สำหรับคลื่นที่เคลื่อนที่ต่อไปจะเสมือนเป็นการเพิ่มค่าคงที่ลอยเลื่อน (Propagation constant) ด้วย คลื่นนี้จะเหนี่ยวนำให้เกิดเป็นประจุในผิวของสารพีโซอิเล็กทริกเป็นรูปร่างเหมือนกับฟิงเกอร์ของ IDT พลังงานรวมของคลื่นบน IDT ตัวรับจึงเป็นค่าผลรวมของสนามไฟฟ้าบนอิเล็กโทรดตัวรับ และจะเปลี่ยนเป็นสัญญาณไฟฟ้าด้วยค่าการเปลี่ยนแปลงค่าหนึ่ง

สำหรับคุณสมบัติการหน่วงสัญญาณคลื่น อันเนื่องจากคลื่นต้องใช้เวลาในการเคลื่อนที่ผ่านผิวของสารพีโซอิเล็กทริก และเวลาในการเคลื่อนที่ของคลื่นผ่านบนผิวของพีโซอิเล็กทริกจะมีค่าขึ้นกับชนิดของสารพีโซอิเล็กทริก จึงมีการนำเอาอุปกรณ์ SAW มาใช้หน่วงสัญญาณความถี่สูงเช่น ในระบบเรดาร์ (RADAR) มี

การใช้อุปกรณ์ SAW สำหรับหน่วงสัญญาณที่ส่งออกไป เพื่อเปรียบเทียบกับสัญญาณที่สะท้อนกลับมา [8] หรือใช้หน่วงสัญญาณที่ส่งออกไปเพื่อจำลองสัญญาณที่สะท้อนกลับมาเป็นการปรับระยะเวลาการทำงานของเรดาร์

นอกจากนี้ยังมีการนำเอา SAW มาใช้ในงานของประมวลผลสัญญาณ (Signal processing) ดังจะเห็นได้จากคุณสมบัติของ Acoustic propagation เป็นตัวกำหนดแบนด์วิธ (Bandwidth) ของการใช้งาน การออกแบบทรานสดิวเซอร์แบบต่างๆ เพื่อให้อุปกรณ์ SAW เป็นวงจรกรองความถี่ [9], [10], [11] ทำให้สามารถสร้างวงจรออสซิลเลเตอร์ความถี่สูงโดยไม่ต้องใช้วงจรคูณความถี่ (Frequency multiplier circuit) ซึ่งเป็นการสร้างออสซิลเลเตอร์ความถี่สูงมาจากออสซิลเลเตอร์ความถี่ต่ำ ออสซิลเลเตอร์แบบคิเลย์ไลน์จะประกอบด้วยคิเลย์ไลน์แบบผลึก (Quartz delay line) ที่มีสัญญาณออกจากคิเลย์ไลน์ถูกป้อนกลับผ่านวงจรขยายให้มีอัตราขยายพอที่จะชดเชยการสูญเสียเมื่อคลื่นเดินทางผ่านไปในสารพิโซอิเล็กทริก สัญญาณไฟฟ้าจากเอาทพุทของวงจรขยายจะกระตุ้นให้เกิดความเค้นที่ให้อิเล็กโทรดตัวส่ง เกิดเป็นคลื่นที่พื้นผิวสารพิโซอิเล็กทริก คลื่นนี้จะเคลื่อนที่ไปใน 2 ทิศทางตรงข้ามกัน ทั้งซ้ายและขวาดังรูป 4.3



รูป 4.3 ใคอะแกรมภาพของออสซิลเลเตอร์แบบคิเลย์ไลน์

เมื่อคลื่นเคลื่อนที่ไปในทิศทางหนึ่งจะต้องใช้เวลาเดินทางไปถึง IDT ตัวรับ แล้วทำให้เกิดประจุนอ์เล็ทโทรคตัวรับเป็นสัญญาณป้อนเข้าสู่วงจรขยาย ถ้าอัตราขยายวนรอบภายในออสซิลเลเตอร์มีขนาดพอจะทำให้เกิดการออสซิลเลต เมื่อให้วงจรขยายมีการเลื่อนเฟสเป็นศูนย์ที่ทุกความถี่ ดังนั้นระยะห่างของแต่ละฟิงเกอร์ต้องทำให้เกิดสภาวะที่การเลื่อนเฟสในดีเลย์ไลน์เท่ากับ $2N\pi$ เมื่อ N เป็นจำนวนเต็มบวก ดังรูป 4.3

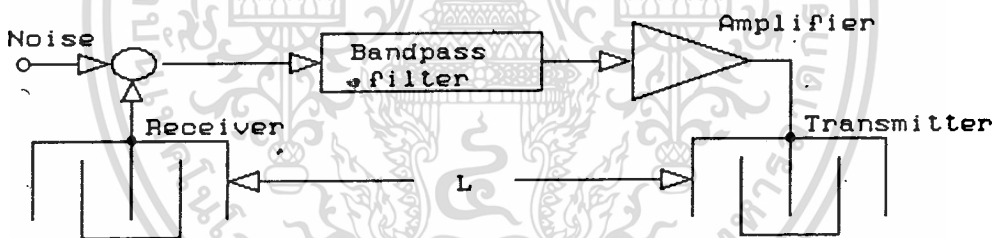
จำนวนไซเคิล (Cycle) ของคลื่นในระหว่างฟิงเกอร์ตัวส่งหรือตัวรับจะเป็นตัวบอกโหมด (Mode) ของออสซิลเลเตอร์ โหมด n หมายถึงที่สถานะอยู่ตัวจะมีคลื่นจำนวน n ไซเคิล ออสซิลเลเตอร์แบบดีเลย์ไลน์สามารถจะควบคุมให้เกิดการกระโดดความถี่จากโหมดหนึ่งไปยังอีกโหมดหนึ่งได้ โดยอาศัยการควบคุมจากภายนอกหรือสัญญาณกระตุ้นระดับสูงในวงจร การข้ามโหมดนี้จะขึ้นกับความกว้างของ IDT และความไม่เชิงเส้นในออสซิลเลเตอร์ด้วย ซึ่งสามารถใช้ดิจิตอลคอมพิวเตอรจ์าลองเพื่อศึกษาปรากฏการณ์ข้ามโหมดนี้ต่อไป โดยเปรียบเทียบกับการทดลองที่ให้ความกว้างของ IDT มีค่าเป็นศูนย์ เพื่อลดผลของแบนด์วิดท์ที่เกิดขึ้นเนื่องจากความกว้างของ IDT หลังจากนั้นจะทำการจำลองผลตอบสนองสัญญาณของ IDT ทั้งตัวรับและตัวส่งที่ความกว้างและจำนวนคู่ฟิงเกอร์ต่างๆกัน ด้วยอัลกอริธึมที่มีความรวดเร็วในการคำนวณและมีความแม่นยำสูง

4.3 การทดลองการข้ามโมดของออสซิลเลเตอร์แบบควบคุมความถี่ด้วยตัวหน่วงเวลา

การทดลองที่ 1

การทดลองปรากฏการณ์การกระโดดข้ามจากโมดหนึ่งไปยังอีกโมดหนึ่งของออสซิลเลเตอร์แบบควบคุมความถี่ด้วยตัวหน่วงเวลา

ตัวหน่วงเวลาที่ใช้ในการศึกษาตอนนี้เป็นแบบนอนคิสเพอร์ซีฟดีเลย์ไลน์ ซึ่งได้มีการนำเอาดีเลย์ไลน์มาใช้ทำเป็นตัวควบคุมความถี่ของออสซิลเลเตอร์ เนื่องจากสามารถทำงานเป็นออสซิลเลเตอร์ที่ความถี่สูงได้ดี และมีความเสถียรของความถี่ (Frequency stability) สูง [12] รวมทั้งสามารถทำงานเป็นออสซิลเลเตอร์ในโมดหนึ่ง แล้วข้ามไปเป็นออสซิลเลเตอร์ในอีกโมดหนึ่งได้ ซึ่งมีประโยชน์มากในการทำงานของ Agile Radars อย่างไรก็ตามการมีความเสถียรที่ดีของดีเลย์ไลน์ ทำให้การกระโดดจากโมดหนึ่งไปยังอีกโมดหนึ่งจะต้องใช้การควบคุมจากภายนอก ดังรูป 4.4



รูป 4.4 ไคอะแกรมแสดงออสซิลเลเตอร์แบบดีเลย์ไลน์ที่มีการควบคุมการข้ามโมดจากภายนอก

จากรูป 4.4 วงจรผ่านแถบความถี่หรือเรโซเนเตอร์จะมีอัตราขยายสูงสุดที่ความถี่เรโซแนนซ์ และที่ความถี่เรโซแนนซ์จะมีการเลื่อนเฟสเท่ากับศูนย์องศา ดังนั้นสภาวะที่จะเกิดการออสซิลเลตได้ดีที่สุดของออสซิลเลเตอร์แบบคิเลย์ไลน์คือ

1. ระยะห่างระหว่างฟิงเกอร์ในแต่ละตอนเท่ากับ

$$(2M + 1)\lambda_c$$

กำหนดให้

λ_c เป็นความยาวคลื่นของความถี่เรโซแนนซ์ของวงจรผ่านแถบ

ความถี่

M เป็นจำนวนเต็มบวก

2. ระยะห่างระหว่างตัวส่งและตัวรับเป็นจำนวนเท่าของความยาว

คลื่น

$$L = n\lambda_c \quad ; n = 1, 2, 3, \dots$$

ในการศึกษาปรากฏการณ์การกระโดดข้ามจากโหมดหนึ่งไปยังอีกโหมดหนึ่งของออสซิลเลเตอร์นี้ จะศึกษาการข้ามโหมดโดยใช้อุปกรณ์ Bucket Brigade Device (BBD) ซึ่ง BBD เป็นอุปกรณ์ที่ทำหน้าที่เป็นตัวหน่วงสัญญาณโดยการส่งประจุไปตามตัวเก็บประจุที่ต่อเนื่องลำดับกันอยู่ภายใน และควบคุมเวลาในการหน่วงด้วยความถี่ของสัญญาณนาฬิกา (Clock signal, F_{ck}) ที่ป้อนให้กับ BBD

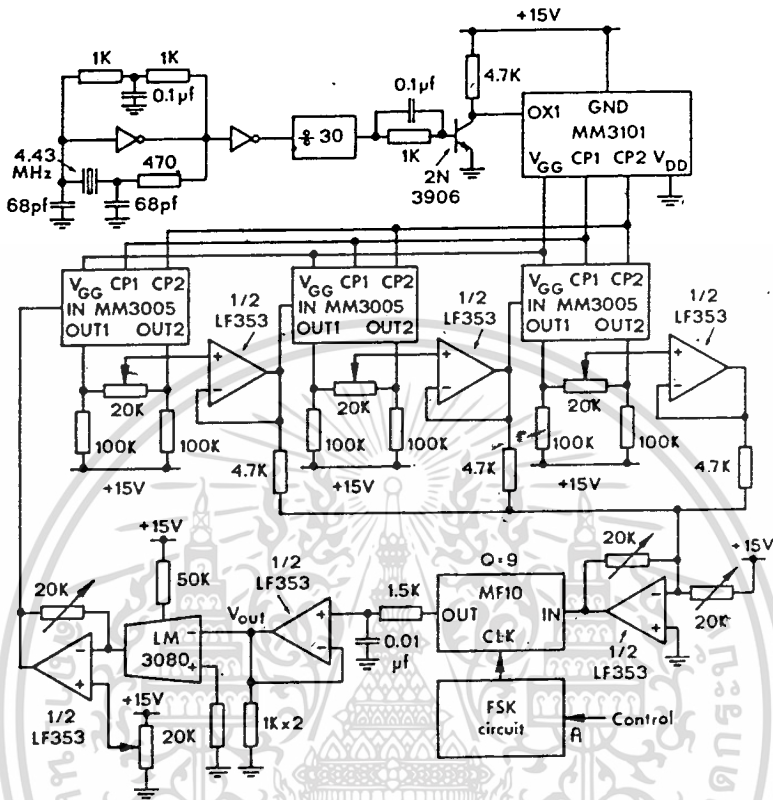
จากวงจรที่ใช้ทำการทดลองดังรูป 4.5 สัญญาณที่ออกจาก BBD ทั้ง 3 ตัวจะถูกนำมารวมกันแล้วทำการศึกษา เหมือนกับว่าในคิเลย์ไลน์มีระยะระหว่างตัวส่งถึงตัวรับตัวที่ 1 (BBD1) และระยะระหว่าง IDT ตัวรับที่ติดกันเท่ากับ M ในการทดลองนี้ถือว่า

1. IDT ตัวส่งและตัวรับเป็นทรานสดิวเซอร์แบบตัวส่งเดี่ยว (Single transmitting transducer) และสัญญาณจากตัวส่งจะเคลื่อนที่ไปในทิศทางเดียวสู่ตัวรับ

2. IDT ทั้งตัวส่งและตัวรับมีความกว้างเกือบเป็นศูนย์ ดังนั้นแบนด์วิดท์ของการใช้งาน IDT จึงคิดให้เป็นอนันต์ (Infinity) ซึ่งในทางปฏิบัติแล้วแบนด์วิดท์ของคิเลย์ไลน์จะมีค่าหนึ่ง [12] แต่ในที่นี้ได้กำหนดให้แบนด์วิดท์มีค่ามากเมื่อเทียบกับความถี่ของสัญญาณจากออสซิลเลเตอร์ในการทดลอง และจะไม่คำนึงถึงเมื่อนำมาเทียบกับค่าตัวประกอบคุณภาพของส่วนวงจรผ่านแถบความถี่ภายนอก

3. คิเลย์ไลน์เป็นแบบนอนออสซิลเลเตอร์คิเลย์ไลน์ และการสูญเสียภายใน

ใน BBD จะแทนได้ด้วยค่าคงที่ค่าหนึ่ง ซึ่งจะนำมาคูณเข้ากับ IDT ตัวรับแต่ละตัว



รูป 4.5 วงจรการทดลองการข้ามโมดของออสซิลเลเตอร์แบบควบคุมความถี่ด้วยคิเลเบิล

4. ไม่มีการสะท้อนของคลื่นเมื่อคลื่นผ่าน IDT
5. วงจรขยายภายนอกไม่ทำให้เกิดการเลือนเฟส และมีคุณสมบัติของอัตราขยายดังสมการ 4.3

$$V_o = A_o \tanh(V_{in}) \quad \dots (4.3)$$

V_o เป็นขนาดของสัญญาณออกจากวงจรขยาย

V_{in} เป็นขนาดของสัญญาณเข้าไปยังวงจรขยาย

A_o เป็นค่าคงที่ของวงจรขยาย

6. สัญญาณรบกวนที่เกิดขึ้นในวงจรจะแทนได้ด้วยแหล่งกำเนิดสัญญาณรบกวน (Noise source) ภายนอก ซึ่งรวมกับสัญญาณที่ป้อนกลับมาเข้าวงจรขยาย

วงจรผ่านแถบความถี่ในรูป 4.5 เป็นวงจรเรโซเนเตอร์ที่ใช้กำหนดย่านผ่านแถบความถี่ของวงจร และเป็นตัวกำหนดโมคของการออสซิลเลตโดยใช้วงจรรวม (Integrated circuit) เบอร์ MF10 ซึ่งเป็นอุปกรณ์ Switched capacitor ที่สามารถเลือกความถี่การออสซิลเลตได้โดยการปรับความถี่ของสัญญาณพิก้าที่ใช้กับวงจรรวม MF10 นี้

$$F_r = \frac{F_{ck10}}{50} \quad \dots (4.4)$$

F_r = ความถี่เรโซแนนซ์ของวงจรผ่านแถบความถี่ MF10

F_{ck10} = ความถี่สัญญาณพิก้าของ MF10

การปรับค่าตัวประกอบคณภาพของวงจรผ่านแถบความถี่ จะสามารถปรับได้ ดังวิธีการใช้งาน MF10 [14]

วงจร FSK (Frequency shift keying) ดังรูป 4.5 ใช้วงจรรวมเบอร์ XR2206 เป็นตัวสร้างสัญญาณพิก้าของ MF10 สัญญาณควบคุมความถี่ของการออสซิลเลตของ XR2206 จะเข้าที่จุด A ความถี่ของสัญญาณที่ออกจากวงจรรวม XR2206 (F_{ck10}) จะมีค่าเท่ากับ 2,884 เฮิรตซ์ เมื่อสัญญาณที่มากับควบคุมมีลอจิก 1 และความถี่ของสัญญาณที่ออกจากวงจร XR2206 (F_{ck10}) จะมีค่าเท่ากับ 3,608 เฮิรตซ์เมื่อสัญญาณที่มากับควบคุมมีลอจิก 0

การทดลองการข้ามโมคของออสซิลเลเตอร์แบบดีเลย์ไลน์โดยใช้ BBD นี้ ความถี่ของการออสซิลเลตจะขึ้นกับ

1. ระยะระหว่างตัวรับสัญญาณที่เอาต์พุตของ BBD แต่ละตัว

เนื่องจากเฟสของวงจรรายนอกเป็นศูนย์ สัญญาณพิก้าที่ใช้กับ BBD ในวงจรรูป 4.5 จะมีความถี่ 73.8333 กิโลเฮิรตซ์ ใน BBD ประกอบด้วยตัวเก็บประจุต่อเรียงตามลำดับ 2 ชุด แต่ละชุดจะมีตัวเก็บประจุ 2049 ตัว และควบคุมด้วยสัญญาณพิก้า 2 ชุด สัญญาณพิก้า 2 ชุดจะมีเฟสตรงข้ามกัน ดังนั้นเวลาของการหน่วง (T) ของคลื่นในการเคลื่อนที่ผ่าน BBD จะเท่ากับ

$$T = \frac{4096}{2 \times F_{ck}} \quad \dots (4.5)$$

ใน BBD 1 ตัว จะมีการหน่วงเวลาของสัญญาณตั้งแต่เข้าถึงออกเท่ากับ 27.738 มิลลิวินาที (millisecond) สัญญาณออกจาก BBD ตัวที่ 1, 2 และ 3 จะถูกบัฟเฟอร์แล้วป้อนเข้าสู่ BBD ตัวที่อยู่ถัดมา โดยสัญญาณที่เกิดจากการรวมกันของ BBD ตัวที่ 1, 2, 3 จะเป็นสัญญาณที่ออกจากดีเลย์ไลน์ ความถี่ของการออสซิลเลตที่เป็นไปได้จะเท่ากับ

$$F_m = \frac{m}{27.738 \times 10^{-3}} H_z ; m = 1, 2, 3, \dots \quad \dots (4.6)$$

เมื่อ m เป็นโมดของการออสซิลเลต

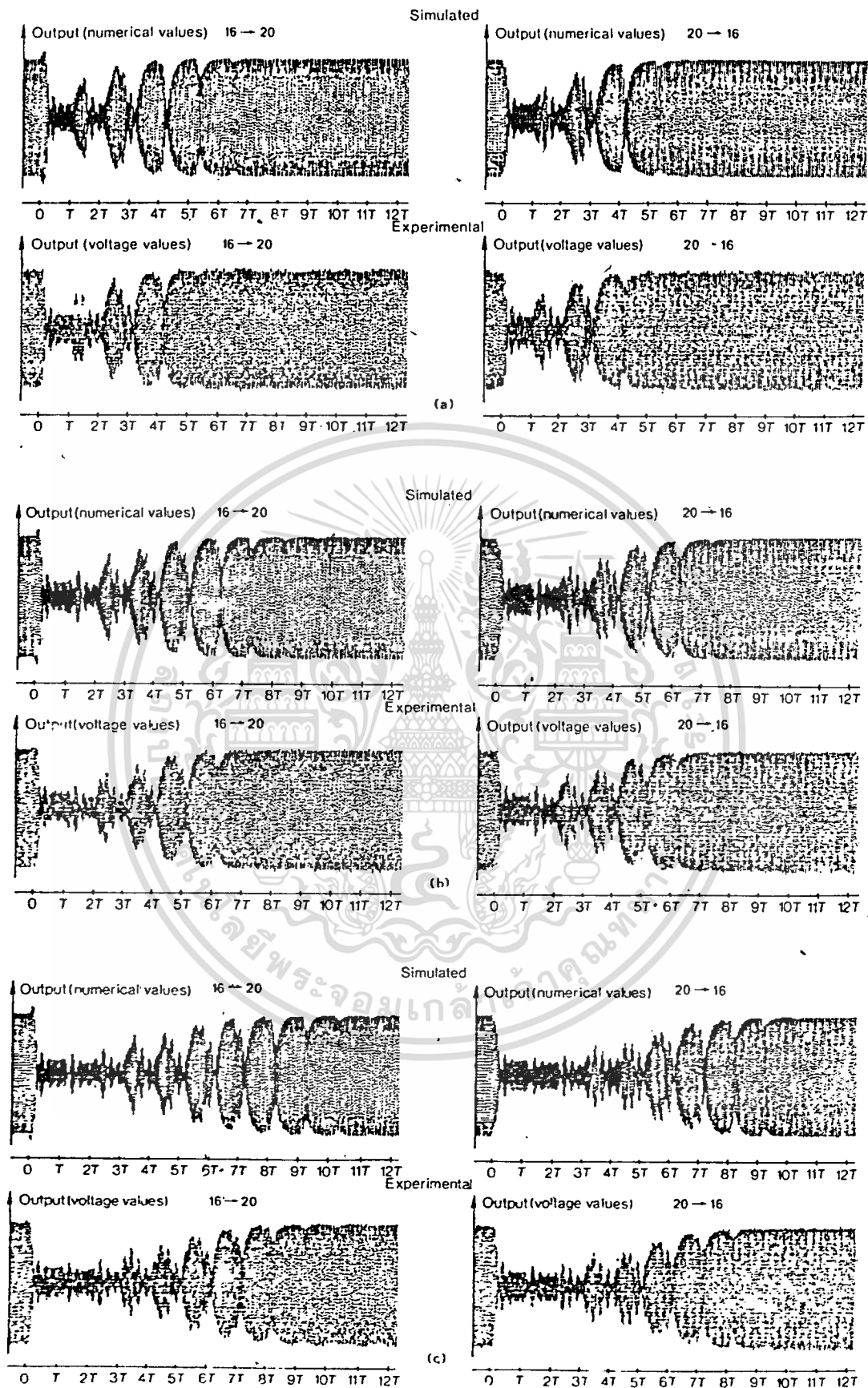
F_m เป็นความถี่ของการออสซิลเลตในโมดที่ m

2. ความถี่เรโซแนนซ์ของวงจรผ่านแถบความถี่ ซึ่งจะเป็นไปตาม

สมการ 4.4

ในการทดลองให้วงจรผ่านแถบความถี่มีค่าตัวประกอบคุณภาพเท่ากับ 9 สัญญาณนาฬิกาซึ่งสร้างจากวงจรออสซิลเลเตอร์แบบคริสตัลในรูป 4.5 จะผลิตความถี่ 4.43 เมกะเฮิรตซ์ แล้วถูกหารลงหารลง 30 เท่าต่อไปยังวงจรรวมเบอร์ MM3101 ซึ่งเป็นตัวปรับระดับโวลเตจของสัญญาณนาฬิกาให้สามารถใช้กับ BBD โดยที่จุดเอาต์พุตของออสซิลเลเตอร์แบบคิเลบ์ไลน์จะต่อเข้ากับบอร์ดเปลี่ยนสัญญาณแอนะล็อกเป็นข้อมูลดิจิทัล (Analog to Digital converter board) ในภาคผนวก ง. ซึ่งมีลักษณะการทำงานแบบมัลติเพล็กซ์แบ่งเวลา (Time division multiplex) ทำให้สามารถบันทึกข้อมูลด้วยอัตราการชักตัวอย่าง (Sampling Rate) สูง บอร์ดเปลี่ยนสัญญาณแอนะล็อกเป็นข้อมูลดิจิทัลนี้ใช้กับเครื่องคอมพิวเตอร์ IBM XT โดยจากบอร์ดเปลี่ยนสัญญาณแอนะล็อกเป็นข้อมูลดิจิทัล จะมีสัญญาณไปควบคุมการทำงานของวงจร FSK ที่จุด A ในรูป 4.5 เพื่อควบคุมออสซิลเลเตอร์ซึ่งผลิตสัญญาณนาฬิกาของ MF10 เป็นการเลือกความถี่เรโซแนนซ์ของวงจรผ่านแถบความถี่โดยใช้วงจรรวม MF10 สัญญาณเอาต์พุตจาก MF10 ซึ่งเป็นอุปกรณ์แบบ Switched capacitor จะมีสัญญาณรบกวนออกมามาก จึงให้สัญญาณที่ออกจาก MF10 ผ่านวงจรผ่านความถี่ต่ำ (Lowpass filter) โดยมีจุดตัดความถี่ (Cut-off frequency) เท่ากับ 10.61 กิโลเฮิรตซ์ เพื่อให้การเลื่อนเฟสที่เกิดจากส่วนของวงจรผ่านความถี่ต่ำมีค่าน้อยมากในขั้นตอนของการเกิดออสซิลเลตของโมดที่ทำการศึกษา สัญญาณที่ออกจากวงจรผ่านความถี่ต่ำจะผ่านบัฟเฟอร์ไปยังวงจรขยายเชิงดำเนินการแบบทรานสคอนดักแตนซ์ (Operational transconductance amplifier, OTA) ซึ่งมีคุณสมบัติความสัมพันธ์ระหว่างเอาต์พุตและอินพุตดังสมการ 4.3 สัญญาณออกจาก OTA ซึ่งอยู่ในรูปของกระแสจะถูกเปลี่ยนเป็นโวลเตจเข้าไปยัง BBD ตัวที่ 1 โดยการทดลองนี้ให้การออสซิลเลตเกิดที่โมด 16 และโมด 20 คือการให้ออสซิลเลเตอร์แบบคิเลบ์ไลน์ผลิตความถี่ 576.8 เฮิรตซ์ และ 721.60 เฮิรตซ์ตามลำดับ ซึ่งในการทดลองการข้ามโมดจะทดลองในกรณีที่มีตัวรับ 1, 2 และ 3 ตัวทำให้มี Q_{eff} ในคิเลบ์ไลน์เท่ากับ 28.3, 42.4 และ 56.9 ตามลำดับ

การทดลองจะเริ่มจากคอมพิวเตอร์ IBM ส่งสัญญาณผ่านบอร์ดเปลี่ยนสัญญาณแอนะล็อกเป็นข้อมูลดิจิทัลไปยังจุด A เพื่อให้วงจร FSK สร้างความถี่สัญญาณนาฬิกาให้มีความถี่เรโซแนนซ์ของ MF10 มีค่าเท่ากับ 576.8 เฮิรตซ์ ออสซิลเลเตอร์แบบคิเลห์ไลน์จะออสซิลเลตที่โมด 16 จากนั้นคอมพิวเตอร์จะส่งสัญญาณไปควบคุมวงจร FSK ให้ผลิตสัญญาณนาฬิกา ทำให้ MF10 มีความถี่เรโซแนนซ์เท่ากับ 721.6 เฮิรตซ์ แล้วเริ่มการบันทึกสัญญาณออกจากออสซิลเลเตอร์ผ่านบอร์ดเปลี่ยนสัญญาณแอนะล็อกเป็นดิจิทัล จนกระทั่งสัญญาณออกจากออสซิลเลเตอร์แบบคิเลห์ไลน์อยู่ที่สภาวะอยู่ตัวโมด 20 และความถี่ของการออสซิลเลตเป็นโมด 20 มีความถี่ 721.60 เฮิรตซ์ ข้อมูลดิจิทัลจะถูกเก็บไว้ในแผ่นแม่เหล็ก (Diskette) เพื่อนำมาทำการแสดงผลต่อไป จากนั้นทดลองการข้ามจากโมด 20 มายังโมด 16 โดยการสั่งให้วงจร FSK เปลี่ยนความถี่เรโซแนนซ์ของ MF10 มาเป็น 576.8 เฮิรตซ์ และบันทึกสัญญาณออกจากออสซิลเลเตอร์แบบคิเลห์ไลน์นับแต่การเปลี่ยนความถี่เรโซแนนซ์ จนกระทั่งมีการออสซิลเลตที่โมด 16 อย่างสมบูรณ์ ผลการทดลองแสดงดังรูป 4.6a, 4.6b และ 4.6c เป็นกรณีที่มีคิเลห์ไลน์ 1, 2 และ 3 ตัว ตามลำดับ



รูป 4.6 ผลการทดลองและการจำลองการข้ามโมดของออสซิลเลเตอร์แบบคิเลบไลน

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.4 อัลกอริทึมและการจำลองการข้ามโมดของออสซิลเลเตอร์แบบคีย์ไลน์

จากวงจรออสซิลเลเตอร์แบบคีย์ไลน์ในรูป 4.5 นั้น สามารถจำลองระบบด้วยคอมพิวเตอร์โดยการแทนคุณสมบัติของวงจรได้ ด้วยการสร้างแบบจำลองในคิจิตอลคอมพิวเตอร์ จากวงจรถัดกล่าวสามารถแบ่งการจำลองระบบเป็นส่วนย่อยๆได้ โดยเฉพาะในส่วนการจำลองคีย์ไลน์จะใช้อัลกอริทึมที่ทำให้ใช้เวลาในการจำลองน้อยที่สุด การสร้างแบบจำลองแต่ละส่วนมีดังนี้

ก. วงจรขยายเชิงค่าเนกการแบบทรานส์คอนคัคแทนซ์

จากวงจรขยาย OTA สามารถแทนได้ในการคำนวณด้วยระบบคิจิตอลคอมพิวเตอร์โดยตรงจากสมการ 4.3 ซึ่งสามารถหาค่าของตัวแปรในสมการได้โดยการทดลองป้อนอินพุตเป็นโวลเตจแล้ววัดขนาดที่เอาต์พุตของ OTA จากนั้นนำค่าที่วัดได้มาทำวิธี Curve fitting ได้ว่า

$$V_o = 1.38 \tanh(1.997 V_{in})$$

V_o เป็นสัญญาณเอาต์พุตจากวงจรขยายเชิงค่าเนกการแบบทรานส์คอนคัคแทนซ์

V_{in} เป็นสัญญาณอินพุตของวงจรขยายเชิงค่าเนกการแบบทรานส์คอนคัคแทนซ์

ข. วงจรผ่านแถบความถี่

MF10 ซึ่งใช้ในการทดลองนี้เป็นผ่านแถบความถี่แบบลำดับที่ 2 ซึ่งมีทรานส์เฟอร์ฟังก์ชัน ดังนี้

$$H(s) = \frac{A_b S}{s^2 + \frac{\omega_0}{Q} s + \omega_0^2} \quad \dots (4.7)$$

A_b เป็นอัตราขยายของ MF10

ω_0 เป็นความถี่เรโซแนนซ์ของวงจรผ่านแถบความถี่ MF10

Q เป็นค่าตัวประกอบคุณภาพของวงจรผ่านแถบความถี่ MF10

โดยใช้วิธีการทรานส์ฟอร์มจาก S-domain ไปเป็น Z-domain เช่นเดียวกับในบทที่ 3.3 จะได้ว่า

$$H(z) = \frac{K_0(1 - Z^{-1})}{1 + K_4K_3Z^{-1} + \frac{K_5Z^{-1}}{K_3}} \quad \dots (4.8)$$

โดยที่

$$\begin{aligned} K_1 &= 2\omega_0Z \\ K_2 &= (\omega_0Z)^2Q \\ K_3 &= 4Q + K_1 + K_2 \\ K_4 &= -8Q + 2K_2 \\ K_5 &= 4Q - K_1 + K_2 \\ K_6 &= \frac{2\omega_0Z}{K_3} \end{aligned}$$

Z เป็นคาบการแบ่งเวลา

ซึ่งเขียนเป็นบล็อกไดอะแกรมแบบเวียนเกิดได้ดังรูป 3.10

การจำลองการทำงานของวงจรผ่านแถบความถี่ จะใช้บล็อกไดอะ-
แกรมดังกล่าวในการสร้างอัลกอริธึมสำหรับการคำนวณ การทรานส์ฟอร์มแบบนี้เป็นแบบการเวียนเกิด ทำให้ผลการคำนวณมีค่าใกล้เคียงความเป็นจริงมากขึ้น โดยไม่มีผลจาก Truncation error แต่ยังมีผลจาก Quantizing error ซึ่งสามารถลดลงได้ โดยการคำนวณด้วยดิจิทัลคอมพิวเตอร์ที่มีความละเอียดสูง การจำลองวงจรผ่านแถบความถี่ข้างต้น มีอัลกอริธึมเช่นเดียวกับการจำลองวงจรผ่านแถบความถี่ในบทที่ 3.3 และการจำลองในวิทยานิพนธ์ฉบับนี้ ใช้ภาษา TURBO PASCAL และมี 8087 Math co-processor ซึ่งมีความละเอียดของตัวเลขในการคำนวณ 64 บิต

ในการจำลองเมื่อมีการเปลี่ยนค่า ω_0 เพื่อบังคับให้การออสซิลเลตมีการข้ามโมดนั้น จะทำการคำนวณค่าตัวแปร K_1 ถึง K_6 ใหม่แล้วทำการคำนวณต่อไป

ค. ดีเลย์ไลน์

การจำลองการทำงานของดีเลย์ไลน์ด้วยดิจิทัลคอมพิวเตอร์นั้น ใช้สมมติฐานในบทที่ 4.2 โดยถือว่าลักษณะการเคลื่อนที่ของคลื่นไม่มีการสูญเสีย และไม่มีการสะท้อนกลับที่ดีเลย์ไลน์ การจำลองการสูญเสียที่เกิดขึ้นเนื่องจากกรณีต่างๆ จะทำการจำลองซึ่งคิดเป็นการสูญเสียรวมของระบบ โดยการคูณสัญญาณเอาท์พุทจาก IDT ด้วยค่าลดทอนสัญญาณ สิ่งที่เกิดขึ้นในดีเลย์ไลน์จึงเป็นเพียงการหน่วงเวลาของสัญญาณระหว่างอินพุทและ เอาท์พุทของดีเลย์ไลน์

พิจารณาดีเลย์ไลน์ซึ่งมีความยาว L แบ่งจำนวนจุดในดีเลย์ไลน์เป็น P จุด ตั้งแต่จุดที่ 1 ถึงจุดที่ P ดังนั้นแต่ละจุดห่างกันเท่ากับ L/P
ให้ $C(n, j)$ เป็นค่าตัวเลขของสัญญาณที่จุด n เมื่อเวลา j
จากรูปของดีเลย์ไลน์ในรูป 4.7



รูป 4.7 การแบ่งจุดในดีเลย์ไลน์

ให้คลื่นเข้าสู่ดีเลย์ไลน์ที่จุด 1 เคลื่อนที่ไปทางขวาออกจากดีเลย์ไลน์ที่จุด P ดังนั้น ที่เวลา j

$$C(k, j) = \begin{cases} I(j) & ; k = 1 \\ C(k-1, j-1) & ; k = 2, 3, \dots, P \end{cases} \quad \dots (4.9)$$

$C(P, j)$ จะเป็นเอาต์พุตจากดีเลย์ไลน์ที่เวลา $j+1$
 $I(j)$ เป็นสัญญาณอินพุตที่เข้าสู่ดีเลย์ไลน์ที่เวลา j

ในการจำลองการเคลื่อนที่ของคลื่นดังสมการ 4.9 นั้น จะใช้หน่วยความจำ P ตำแหน่งในหน่วยความจำ และใช้เวลาในการคำนวณการเคลื่อนย้ายข้อมูล P จำนวนค่าทุกๆ คาบการแบ่งเวลาการจำลอง ซึ่งจะใช้การคำนวณในแต่ละคาบการแบ่งเวลาการจำลองมาก เมื่อพิจารณาจากสมการ 4.9 คลื่นที่เคลื่อนที่เข้าสู่ดีเลย์ไลน์จะใช้เวลา $P\tau$ สำหรับการเคลื่อนที่อยู่ในดีเลย์ไลน์ เมื่อ τ เป็นคาบการแบ่งเวลาการจำลอง

ดังนั้นสามารถเขียนสมการแสดงการเคลื่อนที่ของคลื่น ในระบบแบ่งเวลาได้ว่า

$$C(P+1, j) = I(j-P\tau) \quad \dots (4.10)$$

สมมติให้คิเลย์ไลน์มีความยาวเป็นอนันต์ จาก $-\infty$ ถึง ∞ ให้คลื่นอยู่กับที่และให้ตัวส่งและตัวรับเคลื่อนที่ไปทางซ้าย จะสามารถพิจารณาการเคลื่อนที่ของคลื่น โดยให้ถือเสมือนว่าคลื่นอยู่กับที่และฟังก์เตอร์เคลื่อนที่ไปทางซ้าย ดังนั้นคิเลย์ไลน์ความยาวเป็นอนันต์จะถูกแบ่งออกเป็นจุดย่อยๆ ซึ่งประกอบด้วยจุด $-\infty, \dots, -3, -2, -1, 0, 1, 2, 3, \dots, P, \dots, \infty$

ที่เวลา $j = 0$ สัญญาณออกจากคิเลย์ไลน์จะออกจากจุด P และมีสัญญาณเข้าที่จุด 1

ที่เวลา $j = J$ สัญญาณออกจากคิเลย์ไลน์จะออกจากจุด $P - J$ และสัญญาณเข้าที่จุด $1 - J$ ดังนั้นในการคำนวณจะมีการเคลื่อนย้ายข้อมูลเพียง 2 ค่าเท่านั้น คือการเคลื่อนย้ายข้อมูลเข้าและออกจากคิเลย์ไลน์ในแต่ละคาบเวลาการชักตัวอย่าง แต่โดยอัลกอริธึมนี้ การพิจารณาจากลักษณะที่เกิดขึ้นจริงจะต้องใช้หน่วยความจำจำนวนมาก ดังนั้นจึงต้องใช้วิธีการจัดกฏกับหน่วยความจำเพื่อให้ใช้หน่วยความจำเท่าที่จำเป็น

ให้ $M(k)$ เป็นค่าหน่วยความจำตำแหน่งที่ k ในตอนเริ่มต้นให้หน่วยความจำทุกตำแหน่งมีค่าเป็นศูนย์

$$M(k) = 0.0 \quad ; k = 1, 2, 3, \dots, P$$

ที่เวลา $j = 1$ ตัวเลขแทนอินพุตจะเก็บในหน่วยความจำตำแหน่ง $M(1)$ และตัวเลขแทนเอาต์พุตจะออกจากหน่วยความจำตำแหน่ง $M(P)$

ที่เวลา $j = 2$ ตัวเลขแทนอินพุตจะเก็บในหน่วยความจำตำแหน่ง $M(P)$ และตัวเลขแทนเอาต์พุตจะออกจากหน่วยความจำตำแหน่ง $M(P-1)$

ที่เวลา $j = J$ ตัวเลขแทนอินพุตจะเข้าที่จุด $M(2 - J)$ แต่ถ้า $2 - J$ น้อยกว่า 1 ก็จะมีวนกลับไปยังจุดทางขวาสุดของคิเลย์ไลน์คือจุด P

เช่นที่เวลา $j = 3$ สัญญาณอินพุตจะเข้าที่จุด $P-1$ และ ที่เวลา $j = 4$ สัญญาณอินพุตจะเข้าที่จุด $P-2$ และที่เวลา $j = J$ นี้ตัวเลขแทนเอาต์พุตจะออกจากจุด $M(P-(j-1))$

ด้วยอัลกอริธึมดังกล่าว ทำให้การจำลองการเคลื่อนที่ของคลื่นบนคิเลย์ไลน์จะใช้หน่วยความจำเพียง P ตำแหน่ง และในการคำนวณแต่ละคาบเวลาการจำลอง จะมีการเคลื่อนย้ายข้อมูลเพียง 2 ครั้ง ทำให้ประหยัดเวลาในการคำนวณและเนื้อที่หน่วยความจำ

ง. IDT

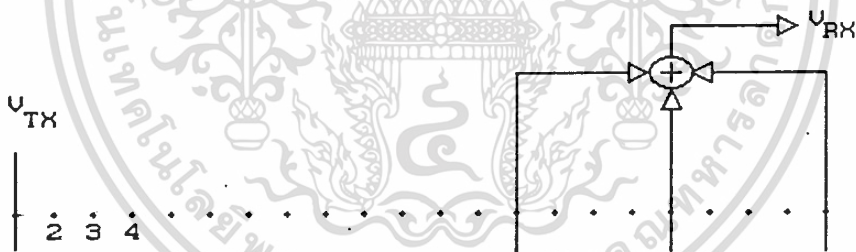
แต่ละ IDT ในการจำลองจะถือว่ามีความกว้างเพียงพอที่จะแทนด้วย 1 จุดของการชักตัวอย่าง ดังนั้นแบนด์วิดท์ของ IDT จะมีค่าเป็นอนันต์ ซึ่งในการทดลองใช้ BBD ที่มีแบนด์วิดท์มากกว่าแบนด์วิดท์ของวงจรผ่านย่านความถี่มาก เพราะฉะนั้นแบนด์วิดท์ของระบบจึงไม่ได้ขึ้นกับแบนด์วิดท์ของ IDT ในการออสซิลเลตโมดที่ M จะมีจำนวน M ไซเคิลของสัญญาณ อยู่ระหว่าง IDT แต่ละตอน กำหนดให้ F_M เป็นความถี่ของการออสซิลเลตในโมดที่ M ดังนั้นแต่ละโมดของการออสซิลเลตที่ติดกันจะมีความถี่ต่างกัน

$$\Delta(F) \Leftarrow F_{M+1} - F_M = F_M - F_{M-1} = \frac{1}{Z}$$

สำหรับค่าตัวประกอบคุณภาพ (Q_{eff}) ของดีเลย์ไลน์ จะเพิ่มได้โดยการเพิ่มจำนวนของทรานสดิวเซอร์ตัวรับ ซึ่งทำให้ Short term stability เพิ่มขึ้นดังสมการ

$$Q_{eff} = \left(\frac{1}{J} \right) \sum_{j=1}^J M_j \quad \dots(4.11)$$

เมื่อ J = จำนวนของทรานสดิวเซอร์ตัวรับ



รูป 4.8 โคอะแกรมภาพของดีเลย์ไลน์ออสซิลเลเตอร์ซึ่งมี 1 ฟิงเกอร์ตัวส่งและ 3 ฟิงเกอร์ตัวรับ

จากวิธีการสร้างแบบจำลองที่ได้กล่าวมาแล้ว จะนำมาจำลองการข้ามโมดของ ออสซิลเลเตอร์แบบดีเลย์ไลน์ เพื่อศึกษาผลในกรณีต่างกันดังนี้

การจำลองที่ 1

การจำลองการข้ามโมคด้วยคอมพิวเตอร์

ในการจำลองการข้ามโมคนี้จะให้คิเลย์ไลน์มีตัวส่ง 1 ตัวและตัวรับ 1, 2 และ 3 ตัวดังรูป 4.8

การจำลองที่ตัวส่งมี 1 ตัวและตัวรับมี 3 ตัว จะแบ่งคิเลย์ไลน์ออกเป็น 6000 จุดและจะใช้หน่วยความจำทั้งหมด 6000 คำแหน่ง คือคำแหน่งที่ 1 ถึง 6000 สำหรับเก็บข้อมูลของแต่ละจุด ตัวรับที่ 1, 2 และ 3 อยู่ห่างจากตัวส่ง 2000, 4000 และ 6000 จุดตามลำดับ การจำลองกรณีตัวรับ 1 ตัวก็จะเอาค่าออกจากคำแหน่ง 2000 กรณีตัวรับ 2 ตัวก็จะเอาค่าเฉลี่ยออกจากคำแหน่ง 2000 และ 4000 กรณีตัวรับ 3 ตัวก็จะเอาค่าออกโดยการเฉลี่ยค่าในหน่วยความจำคำแหน่งที่ 2000, 4000 และ 6000 ตามลำดับ โดยค่าที่ได้จากตัวรับมากกว่า 1 ตัวจะเฉลี่ยด้วยจำนวนตัวรับ ซึ่งจะเขียนได้ว่า

$$V_{rx} = \frac{C(2000,1) + C(4000,1) + C(6000,1)}{3}$$

3

จากนั้นนำเอาค่า V_{rx} ที่ได้ ไปใช้ในการจำลองส่วนวงจรขยายแบบควบคุมอัตราขยายอัตโนมัติและวงจรผ่านแถบความถี่ เอาท์พุทซึ่งเป็นผลที่ได้จากการจำลองวงจรขยายจะนำมาเก็บในคิเลย์ไลน์คำแหน่งที่ 6000 แล้วเลือกคำแหน่งของฟังก์เตอร์ตัวรับไปทางซ้าย 1 คำแหน่ง ก็จะเป็นการจำลองคิเลย์ไลน์ในเวลาถัดไป 1 คาบเวลาการจำลอง โดยมีการจัดการหน่วยความจำดังที่กล่าวมาข้างต้น

ในการจำลองการข้ามโมคนี้ จะทำการจำลองการข้ามโมคระหว่างโมคที่ 16 กับโมคที่ 20 ซึ่งการควบคุมการข้ามโมคจะใช้วงจรผ่านแถบความถี่แบบดิจิทัล (Digital filter) เป็นตัวควบคุม วงจรผ่านแถบความถี่แบบดิจิทัลมีค่าตัวประกอบคุณภาพเท่ากับ 9 ซึ่งน้อยกว่าค่าตัวประกอบคุณภาพของคิเลย์ไลน์ในสมการ 4.11 ค่า Q_{eff} ของคิเลย์ไลน์เฉลี่ยระหว่างโมค 16 กับโมค 20 มีค่าเท่ากับ 28.3, 42.4 และ 56.9 สำหรับกรณีที่มีตัวรับ 1, 2 และ 3 ตัวตามลำดับ การจำลองการข้ามโมคของออสซิลเลเตอร์จะมีอัลกอริธึมดังนี้

ขั้นที่ 1 : จงเนื้อที่หน่วยความจำเท่ากับจำนวนจุดในคิเลย์ไลน์ เพื่อใช้เก็บค่าตัวเลขของสัญญาณ

ขั้นที่ 2 : คำนวณตัวแปรในสมการ 4.8 และรูป 3.10 สำหรับวงจรผ่านความถี่ จะให้มีค่าตัวประกอบคุณภาพเท่ากับ 9 และมีความถี่เรโซแนนซ์เท่ากับความถี่การออสซิลเลตในโมค 16

ขั้นที่ 3 : จำลองวงจรแต่ละส่วนโดยการคำนวณแต่ละส่วนตามลำดับดังนี้

ก. ทำการจำลองส่วนดีเลย์ไลน์ โดยนำผลการคำนวณในข้อ ค. ไปเก็บในดีเลย์ไลน์ แต่ก่อนที่จะเก็บต้องคำนวณหาเอาต์พุต (V_{RX}) จากดีเลย์ไลน์ก่อน ตามวิธีการจำลองดีเลย์ไลน์ที่กล่าวมาแล้ว V_{RX} ที่ได้จะนำไปใช้ในการคำนวณในข้อ ข. และ ค.

ข. ทำการจำลองวงจรผ่านแถบความถี่ โดยกำหนดค่าตัวแปรในไดอะแกรมการเวียนเกิดในรูป 3.10 แล้วคำนวณตามสมการ 3.20a และ 3.20b โดย X_n ในสมการ 3.20a เป็นค่าของ V_{RX} ที่เกิดจากการคำนวณในข้อ ก. การจำลองส่วนนี้เหมือนการจำลองวงจรผ่านแถบความถี่ในบทที่ 3

ค. จำลองวงจรขยายแบบควบคุมอัตราขยายอัตโนมัติ ตามสมการ 4.3 ในวิธีการจำลองบทที่ 4.4 ก. V_{in} ในสมการเป็นเอาต์พุตจากการจำลองวงจรผ่านแถบความถี่ขั้นที่ 3 ข. เอาต์พุตคือ V_o ที่ได้จากวงจรขยายจะนำไปเก็บในดีเลย์ไลน์ตามลำดับขั้นที่ 3 ก.

ขั้นที่ 4 : คำนวณซ้ำในขั้นที่ 3 จนกระทั่งเกิดการออสซิลเลตอยู่ในสถานะอยู่ตัว จากนั้นเริ่มการบันทึกเอาต์พุตของออสซิลเลเตอร์ และเมื่อเกิดการออสซิลเลตอยู่ในสถานะอยู่ตัว จะเข้าสู่การคำนวณในขั้นที่ 5

ขั้นที่ 5 : บังคับให้เกิดการข้ามโมด ด้วยการเปลี่ยนความถี่เรโซแนนซ์ของวงจรผ่านแถบความถี่ โดยการคำนวณตัวแปรสำหรับวงจรผ่านความถี่จากสมการ 4.8 และรูป 3.10 ให้มีความถี่เรโซแนนซ์เท่ากับความถี่การออสซิลเลตในโมด 20. แล้วทำการคำนวณเช่นเดียวกับการจำลองในขั้นที่ 3

ขั้นที่ 6 : เมื่อเกิดการออสซิลเลตจนเข้าสู่สถานะอยู่ตัว ก็ทำการบังคับการออสซิลเลตให้เป็นโมดที่ 16 โดยการบังคับการข้ามโมดเหมือนกับในขั้นที่ 5

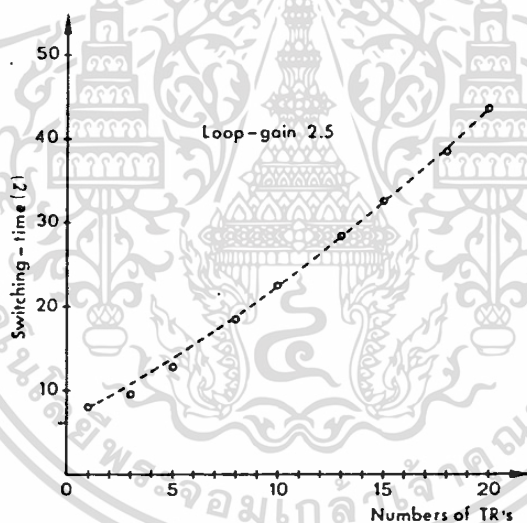
จากนั้นทำการจำลองตั้งแต่ขั้นที่ 1 ถึงขั้นที่ 6 โดยเปลี่ยนตัวรับเป็น 1, 2 และ 3 ตัว

ผลการจำลองแสดงได้ดังรูป 4.6a, 4.6b และ 4.6c ซึ่งเป็นกรณีที่มีดีเลย์ไลน์ 1, 2 และ 3 ตัว ตามลำดับ

การจำลองที่ 2

การจำลองการข้ามโมคที่จำนวนตัวรับต่างกัน

จากการจำลองที่ 1 ซึ่งเปรียบเทียบการทดลองการข้ามโมคของออสซิลเลเตอร์จากโมค 16 ไปยังโมค 20 และจากโมค 20 ไปยังโมค 16 นั้น ให้ผลที่ใกล้เคียงกันมาก จึงใช้อัลกอริธึมในการจำลองที่ 1 เพื่อหาเวลาที่ใช้ในการเปลี่ยนจากโมคหนึ่งไปยังอีกโมคหนึ่ง โดยกำหนดให้อัตราขยายรอบของวงจรมีค่าคงที่เท่ากับ 2.5 และทำการจำลองที่ตัวรับ 1, 3, 5, 8, 10, 14, 16, และ 20 ตัวตามลำดับ ส่วนการบันทึกเวลาที่ใช้ในการเปลี่ยนความถี่การออสซิลเลตจากโมคหนึ่งไปยังอีกโมค จะเริ่มนับจากมีการเปลี่ยนความถี่ถึงกลางของวงจรผ่านแถบความถี่ จนกระทั่งสัญญาณเอาต์พุตจากออสซิลเลเตอร์อยู่ในสถานะอยู่ตัว จึงได้ผลการจำลองแต่ละจุดแล้วทำ Curve Fitting ซึ่งแสดงได้ดังรูป 4.9

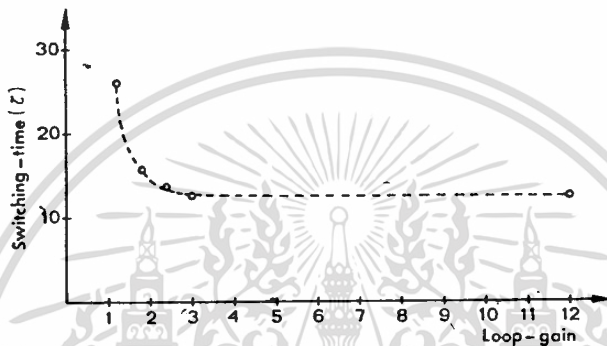


รูป 4.9 ผลการจำลองหาเวลาการข้ามโมคของออสซิลเลเตอร์กรณีที่มี IDI ตัวรับหลายตัว

การจำลองที่ 3

การจำลองหาเวลาในการเปลี่ยนจากโหมดหนึ่งไปยังอีกโหมดหนึ่ง ที่อัตราขยายวนรอบค่าต่างๆกัน

เป็นการจำลองโดยใช้อัลกอริธึมในการจำลองที่ 1 แต่เปลี่ยนอัตราขยายของวงจขยายเป็นค่าต่างๆกัน แล้วให้หาเวลาในการข้ามโหมดของออสซิลเลเตอร์จากโหมด 16 ไปยังโหมด 20 เมื่อได้ผลการจำลองแต่ละจุดแล้วทำ Curve Fitting แสดงดังรูป 4.10



รูป 4.10 ผลการจำลองหาเวลาการข้ามโหมดของออสซิลเลเตอร์จากโหมด 16 ไปยังโหมด 20 ที่อัตราขยายวนรอบค่าต่างๆกัน

4.5 เอกสารอ้างอิงบทที่ 4

- [1] Roberts J.B.G., Darby B.J., Bale R.A. and Eames R., "Fast S.A.W.-Based Discrete Fourier Transformer," Electron. Lett., Vol.14, No.24, pp.783-784, Nov. 1978.
- [2] Smith R.W., Gerrard H.M., Collins J.H., Reeder T.M. and Shaw H.J., "Analysis of Interdigital Surface Wave Transducers by use of a Equivalent Circuit Model," IEEE Trans. Microwave Theory Tech., Vol.MTT-17, No.11, pp.856-864, Nov. 1969.
- [3] Sugiyama K. and Yoshikawa S., "Transient Response of Interdigital Surface Wave Transducers," IEEE Trans. Sonics and Ultrasonics, Vol.SU-20, No.1, pp.34-38, Jan. 1973.
- [4] Smith R.W., Gerrard H.M. and Jones W.R., "Analysis and Design of Dispersive Interdigital Surface-Wave Transducers," IEEE Trans. Microwave Theory Tech., Vol.MTT-20, No.7, pp.458-471, July 1972.
- [5] Tseng C.C., "Frequency Response of an Interdigital Transducer for Excitation of Surface Elastic Waves," IEEE Trans. Electronic devices, Vol.ED-15, No.8, Aug. 1968.
- [6] Mariani E.A., "Terms and Definitions for Surface Acoustic Wave Devices," IEEE Trans. Sonics and Ultrasonics, Vol.SU-32, No.4, pp.476-480, July 1985.
- [7] Adler R., "Simple Theory of Acoustic Amplification," IEEE Trans. Sonics and Ultrasonics, Vol.Su-18, No.3, pp.115-118, July 1971.
- [8] Lewis M.F., "Practical Frequency Source for use in Agile Radars," Electron. Lett., Vol.21, No.22, pp.1017-

1018, Oct. 1985.

- [9] Tancrell R.H. and Holland M.G., "Acoustic Surface Wave Filters," Proc. IEEE, Vol.59, No.3, pp.393-409, March 1971.
- [10] Morgan D.P., "Log-Periodic Transducers for Acoustic Surface Wave," Proc. IEEE, Vol.119, No.1, Jan. 1972.
- [11] Huang F., Paige E.G.S. and Selviah D.R., "High-performance SAW Dispersive delay line Using Reflective Thin Metal Dot Arrays," Electron. Lett., Vol.22, No.12, pp.653-654, June 1986.
- [12] Butler W.J., Barron M.B. and Puckette C.M., "Bucket-Brigade Bandwidth characteristics," Electron. Lett., Vol 8, No.17, pp.449-450, Aug. 1972.
- [13] Crabb J., Lewis M.F. and Maines J.D., "Surface-Acoustic-Wave Oscillators Mode Selection and Frequency Modulation," Electron. Lett., Vol.9, No.10, May 1973.
- [14] National semiconductor corporation, "Linear Databook," 1982.

บทที่ 5

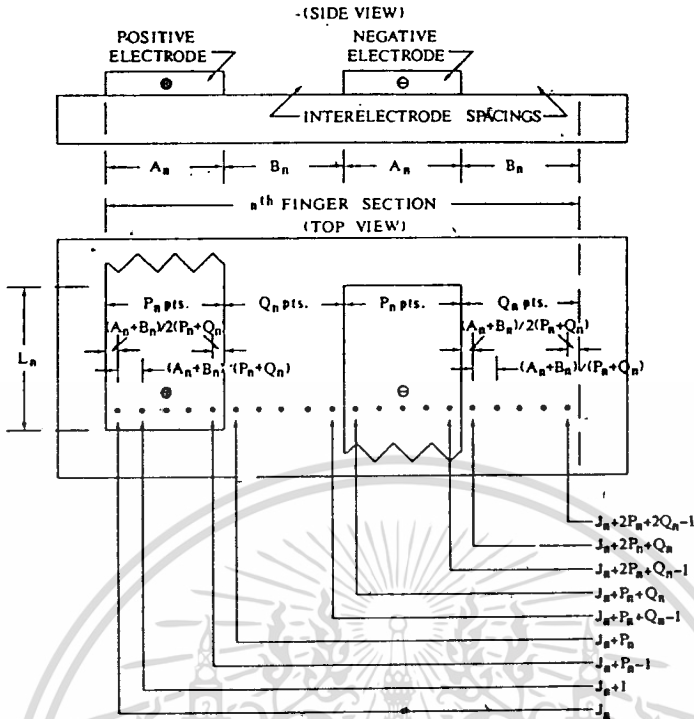
การจำลอง IDT แบบฟิงเกอร์มีความกว้าง (Simulation of IDT with finger-width)

5.1 บทนำการจำลอง IDT แบบฟิงเกอร์มีความกว้าง

จากคุณสมบัติพื้นฐานของ IDT และจากการประมาณสนามไฟฟ้าที่อิเล็กโทรดด้วย Flat field approximation [3] ในบทที่ 4.2 นั้น เมื่อป้อนสัญญาณไปยังอิเล็กโทรด ตัว IDT จะทำให้เกิดความเค้นที่บริเวณใต้อิเล็กโทรดซึ่งเป็นพีโซอิเล็กทริก ทำให้เกิดคลื่นเคลื่อนที่ไปใน 2 ทิศทางที่ตรงข้ามกันจากคุณสมบัติของคลื่นเมื่อเคลื่อนที่สวนทางกันจะเกิดการแทรกสอด (Interference) ที่จุดที่สวนทางกันนั้น ผลรวมของการขจัด (Displacement) ของคลื่นที่จุดนั้นก็จะ เป็นผลรวมของคลื่นทั้งสอง และเมื่อคลื่นเคลื่อนที่ผ่านไบนอิเล็กโทรดของตัวส่ง จะเกิดการเสริมของพลังงานคลื่นที่บริเวณโลหะด้วยพลังงานที่ป้อนเข้าไปยังตัวส่ง ที่บริเวณอิเล็กโทรดของ IDT ตัวรับนั้นจะมีศักดา คงที่เท่ากันตลอดเพลท (Plate) ดังนั้นพลังงานคลื่นที่บริเวณอิเล็กโทรดตัวรับจะทำให้เกิดไฟฟ้าที่เพลทเป็นค่าเฉลี่ยเท่ากันตลอดทั่วทั้งผิวโลหะ

5.2 การจำลองคิเลย์ไลน์และอินเตอร์ดิจิตอลทรานสดิวเซอร์

การจำลองคิเลย์ไลน์และอินเตอร์ดิจิตอลทรานสดิวเซอร์ จะใช้คุณสมบัติพื้นฐานในข้อ 5.1 มาเขียนเป็นอัลกอริทึมสำหรับการจำลองด้วยดิจิตอลคอมพิวเตอร์ โดยทำการแบ่งอุปกรณ์ SAW ออกเป็นส่วนต่างๆ 3 ส่วนคือ IDT ตัวส่งและ IDT ตัวรับ และคิเลย์ไลน์



รูป 5.1 รูป IDT 1 ส่วน

จากรูป 5.1 เป็นช่วงที่ n ของ IDT ในช่วงนี้จะแบ่งออกเป็นจุดย่อย (Discrete point) ของสัญญาณ ที่มีระยะห่างเท่าๆกัน ให้บนอิเล็กโทรดบวกและอิเล็กโทรดลบซึ่งกว้างเท่ากับ A_n แบ่งออกเป็น P_n จุด บนแต่ละอิเล็กโทรด และช่องว่างระหว่างอิเล็กโทรด (Interelectrode spacing) จะแบ่งออกเป็น Q_n จุด ดังนั้นใน 1 ช่วงของ IDT จะแบ่งออกเป็น $2(P_n+Q_n)$ จุด สำหรับบนคิสเพอร์ซีฟติเลย์ไลน์แต่ละช่วง จะแบ่งจำนวนจุดแต่ละช่วงเท่ากัน และ IDT แบบบนคิสเพอร์ซีฟติเลย์ไลน์ N ช่วง สามารถเขียนสมการได้ดังนี้

$$2(P_n + Q_n) = 2(P_1 + Q_1) ; n = 2, 3, 4, \dots, N \quad \dots (5.1)$$

สำหรับคิสเพอร์ซีฟติเลย์ไลน์นั้น ในแต่ละช่วงจะมีความกว้างไม่เท่ากันเนื่องจากความต้องการในการออกแบบไม่เหมือนกัน เช่น แบบ Log-periodic transducer [1] มีความต้องการในเรื่องการกำหนดเวลาหน่วงของสัญญาณขึ้นกับความถี่และค่าตัวประกอบคุณภาพที่ค่อนข้างคงที่ตลอดย่านความถี่การใช้งาน

$$\text{ระยะห่างระหว่างจุดย่อย} = \frac{(A_n + B_n)}{(P_n + Q_n)}$$

จุดที่อยู่ขอบของอิเล็กทรอนิกส์และช่องว่างระหว่างอิเล็กทรอนิกส์จะห่างจากขอบเท่ากับ $\frac{(A_n + B_n)}{2(P_n + Q_n)}$

ในการจำลองด้วยดิจิทัลคอมพิวเตอร์นั้น ที่แต่ละจุดย่อยจะประกอบด้วยหน่วยความจำ 2 ชุด ชุดหนึ่งสำหรับเก็บค่าตัวเลขของคลื่นที่เคลื่อนที่ไปทางขวาและอีกชุดหนึ่งสำหรับเก็บค่าตัวเลขของคลื่นที่เคลื่อนที่ไปทางซ้าย ในที่นี้กำหนดให้ $C_r(k, j)$ เป็นค่าตัวเลขที่จุด k ของคลื่นที่เคลื่อนที่ไปทางขวาเมื่อเวลา j และ $C_l(k, j)$ เป็นค่าตัวเลขที่จุด k ของคลื่นที่เคลื่อนที่ไปทางซ้ายเมื่อเวลา j การจัดรวมของคลื่นที่จุด k เมื่อเวลา j คือ $C_t(k, j)$ จะมีค่า

$$C_t(k, j) = C_r(k, j) + C_l(k, j) \quad \dots (5.2)$$

จากรูป 5.1 ในช่วงที่ n จะแบ่งออกเป็น $2(P_n + Q_n)$ จุด โดยจุดซ้ายสุดของช่วงที่ n คือจุด J_n ซึ่งเป็นขอบซ้ายสุดของอิเล็กทรอนิกส์บวก และจุดทางขวาสุดของช่วงที่ n คือจุด $J_n + 2P_n + 2Q_n - 1$ จะเป็นจุดขวาสุดของช่องว่างระหว่างอิเล็กทรอนิกส์ทางขวาของตอนที่ n

ในกรณีของ IDT ที่ไม่เป็นอุดมคตินั้น เมื่อคลื่นเคลื่อนที่ต่อไปถึงที่บริเวณขอบของอิเล็กทรอนิกส์จะเกิดการเปลี่ยนแปลงของคลื่น คือคลื่นที่เคลื่อนที่จากบริเวณหนึ่งไปยังอีกบริเวณหนึ่ง จะมีพลังงานส่วนหนึ่งผ่านไปได้อีกส่วนจะสะท้อนกลับ

ให้ I_{FM} เป็นอัตราส่วนของคลื่นที่ผ่านเข้าไปในอิเล็กทรอนิกส์ จากบริเวณช่องว่างระหว่างอิเล็กทรอนิกส์ต่ออัตราส่วนของคลื่นที่เข้าหาอิเล็กทรอนิกส์

R_{FM} เป็นอัตราส่วนของคลื่นที่สะท้อนกลับสู่ช่องว่างระหว่างอิเล็กทรอนิกส์ต่อคลื่นที่วิ่งเข้าหาอิเล็กทรอนิกส์

T_{MF} เป็นอัตราส่วนคลื่นเคลื่อนที่จากบริเวณอิเล็กทรอนิกส์ไปยังช่องว่างระหว่างอิเล็กทรอนิกส์ต่อคลื่นที่เคลื่อนที่ที่จะเข้าหาช่องว่างระหว่างอิเล็กทรอนิกส์

R_{MF} เป็นอัตราส่วนคลื่นที่สะท้อนกลับสู่อิเล็กทรอนิกส์ต่อคลื่นที่เคลื่อนที่เข้าหาช่องว่างระหว่างอิเล็กทรอนิกส์

ซึ่งเขียนเป็นสมการแสดงความสัมพันธ์ของคลื่นที่เคลื่อนที่จากตัวกลางหนึ่งไปยังอีกตัวกลางหนึ่ง โดยไม่มีการสูญเสียเป็นพลังงานอย่างอื่นได้ดังนี้

$$T_{FM} + R_{FM} = 1 \quad \dots (5.3a)$$

$$T_{MF} + R_{MF} = 1 \quad \dots (5.3b)$$

ขณะที่คลื่นเคลื่อนที่ไปบนโลหะอิเล็กโทรดตัวส่ง สัญญาณไฟฟ้าที่ป้อนให้กับอิเล็กโทรดตัวส่งจะทำให้เกิดความเค้นที่ผิวของพีโซอิเล็กทริก เป็นการเสริมพลังงานของคลื่นที่กำลังเคลื่อนที่ในทั้งสองทิศทาง

ให้ $S_{in}(j)$ เป็นสัญญาณไฟฟ้าที่ป้อนเข้าไปยังอิเล็กโทรดตัวส่ง และจะถูกเปลี่ยนเป็นพลังงานคลื่นด้วยอัตราส่วนที่เป็นค่าคงที่ค่าหนึ่ง (K_{Tx}) โดยสัญญาณที่เข้าสู่อิเล็กโทรดตัวส่งที่แต่ละคาบเวลา j จะเปลี่ยนเป็นพลังงานคลื่นแสดงได้ดังสมการ 5.4

$$I(j) = K_{Tx} S_{in}(j) \quad \dots (5.4)$$

ให้ $I(j)$ เป็นพลังงานคลื่นบนผิวของอิเล็กโทรดที่คาบเวลา j และโวลเตจ $S_{in}(j)$ ที่มีค่าเท่ากันนี้แต่มีค่าเป็นลบ จะเกิดบนอิเล็กโทรดลบ และทุกจุดบนอิเล็กโทรดจะมีลักษณะของศักดาไฟฟ้าเทียบเท่า ซึ่งโดยวิธีการประมาณด้วย Flat field approximation แล้ว พลังงานดังกล่าวจะแบ่งออกไปยังแต่ละจุดย่อยบนอิเล็กโทรดบวกและลบจุดละเท่าๆกัน ดังนั้นจะแสดงได้ว่า

$$C_r(k, j) = C_r(k, j) \pm \frac{(L_n) I(j)}{R_n} \quad \dots (5.5a)$$

$$C_l(k, j) = C_l(k, j) \pm \frac{(L_n) I(j)}{R_n} \quad \dots (5.5b)$$

$$k = J_n, J_n+1, \dots, J_n+P_n-1, J_n+P_n+Q_n, J_n+P_n+Q_n+1, \dots, J_n+2P_n+Q_n-1$$

ให้ L_n เป็นความยาวของฟิงเกอร์ซึ่งจะใช้เป็นค่า Normalize รวมทั้งเป็นการแทน Apodization ในการใช้งานของ IDT จริง

R_n เป็นจำนวนของจุดย่อยในช่วงที่ n มีค่าเท่ากับ

$$R_n = 2(P_n + Q_n)$$

สมการ 5.5a เป็นสมการของคลื่นที่เคลื่อนที่ไปทางขวาแล้วเพิ่มด้วยพลังงานที่ป้อนให้กับอิเล็กโทรดบวกหรือลบ

สมการ 5.5b เป็นสมการของคลื่นที่เคลื่อนที่ไปทางซ้าย แล้วเพิ่มด้วยพลังงานที่ป้อนให้อิเล็กโทรดบวกหรือลบ

จุด k ในสมการ 5.5a และ 5.5b เป็นจุดบนอิเล็กโทรดบวก หรือลบในช่วงที่ n

ในขณะที่คลื่นเคลื่อนที่ไปในทั้ง 2 ทิศทาง จะสามารถเขียนความสัมพันธ์ของพลังงานที่เคลื่อนที่ไปบนอุปกรณ์ SAW ดังสมการ 5.6 และ 5.7

$$C_r(k+1, j+1) = \begin{cases} T_{FM}C_r(k, j) + R_{MF}C_1(k+1, j) & \dots (5.6a) \\ C_r(k, j) & \dots (5.6b) \\ T_{MF}C_r(k, j) + R_{FM}C_1(k+1, j) & \dots (5.6c) \end{cases}$$

$$C_1(k-1, j+1) = \begin{cases} T_{FM}C_1(k, j) + R_{MF}C_r(k-1, j) & \dots (5.7a) \\ C_1(k, j) & \dots (5.7b) \\ T_{MF}C_1(k, j) + R_{FM}C_r(k-1, j) & \dots (5.7c) \end{cases}$$

จากสมการ 5.6a, 5.6b, 5.6c, 5.7a, 5.7b และ 5.7c แสดงให้เห็นการเคลื่อนที่ของคลื่นใน 3 กรณีคือ

กรณีที่ 1. เป็นการเคลื่อนที่ของคลื่นจากบริเวณช่องว่างระหว่างอิเล็กโทรดเข้าไปยังบริเวณอิเล็กโทรด ดังเห็นได้จากสมการ 5.6a ซึ่งคลื่นที่เคลื่อนที่ไปทางขวาจากช่องว่างระหว่างอิเล็กโทรดเข้าสู่อิเล็กโทรด จะรวมด้วยคลื่นที่เคลื่อนที่จากอิเล็กโทรดไปทางซ้ายเข้าสู่ช่องว่างแล้วเกิดการสะท้อนกลับ และในสมการ 5.7a จะแสดงให้เห็นว่าคลื่นที่เคลื่อนที่ไปทางซ้ายจากบริเวณช่องว่างระหว่างอิเล็กโทรดเข้าสู่อิเล็กโทรดจะรวมด้วยคลื่นที่เคลื่อนที่ไปทางขวาจากอิเล็กโทรดเข้าสู่ช่องว่างระหว่างอิเล็กโทรด แล้วเกิดการสะท้อนกลับ

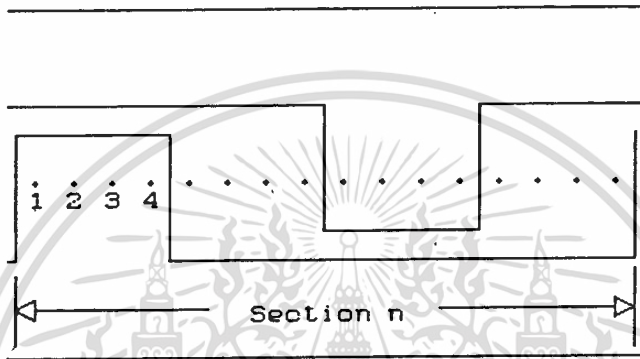
กรณีที่ 2. สมการ 5.6b และ 5.7b เป็นสมการแสดงการเคลื่อนที่ของคลื่นจากจุดหนึ่งไปยังอีกจุดหนึ่งที่อยู่ในบริเวณตัวกลางเดียวกัน

กรณีที่ 3. การเคลื่อนที่ของคลื่นจากบริเวณอิเล็กโทรดไปยังช่องว่างระหว่างอิเล็กโทรด ดังเห็นจากสมการ 5.6c ซึ่งคลื่นที่เคลื่อนที่ไปทางขวาจากอิเล็กโทรดเข้าสู่ช่องว่างระหว่างอิเล็กโทรด จะรวมด้วยคลื่นที่เคลื่อนที่จากช่องว่างระหว่างอิเล็กโทรดไปทางซ้ายเข้าสู่อิเล็กโทรด แล้วเกิดการสะท้อนกลับ และในสมการ 5.7c แสดงให้เห็นว่าคลื่นที่เคลื่อนที่ไปทางซ้ายจากบริเวณอิเล็กโทรดเข้าสู่ช่องว่างระหว่างอิเล็กโทรด จะรวมด้วยคลื่นที่เคลื่อนที่จากช่องว่างระหว่างอิเล็กโทรดไปทางขวา เข้าสู่อิเล็กโทรดแล้วเกิดการสะท้อนกลับ

คลื่นที่เคลื่อนที่จะผ่านออกจากบริเวณของ IDT ช่วงที่ n เมื่อเวลา j คือ $C_r(J_n+2p_n+2q_n-1, j-1)$ และ $C_1(J_n, j-1)$

คลื่น C_r ที่เคลื่อนที่ออกจากช่วงที่ n ทางขวา ก็จะเข้าสู่ช่วงที่ $n+1$ ที่อยู่ถัดไป คลื่นที่ออกทางขวาของช่วงที่ N ซึ่งเป็นช่วงสุดท้ายของ IDT ตัวส่ง เป็นคลื่นที่ออกจาก IDT ตัวส่งทางขวา ส่วนคลื่น C_1 ที่เคลื่อนที่ออกจากช่วงที่ n ทางซ้าย ก็จะเข้าสู่ช่วงที่ $n-1$ ต่อไป และคลื่นที่เคลื่อนที่ออกทางซ้ายของช่วงที่ 1 จะเป็นคลื่นที่เคลื่อนที่ออกจาก IDT ตัวส่งทางด้านซ้าย

สำหรับที่บริเวณ IDT ตัวรับ เมื่อคลื่นจาก IDT ตัวส่งเคลื่อนที่ผ่าน คีเลย์ไลน์ไปจนถึง IDT ตัวรับ แล้วคลื่นเคลื่อนที่ต่อไปบนอิลีกโทรด ความเค้น ที่ผิวของสารพิโซอิเล็กทริกจะก่อให้เกิดประจุบนอิลีกโทรด ทำให้เกิดระดับ สัญญาณขึ้นระหว่างอิลีกโทรดบวกและอิลีกโทรดลบ โดยมีค่าเป็นค่าแตกต่าง ระหว่างค่าเฉลี่ยประจุบนอิลีกโทรดบวกกับอิลีกโทรดลบ



รูป 5.2 IDT ตัวรับช่วงที่ n

พิจารณาจากรูป 5.2 แสดงถึงแต่ละจุดย่อยๆใน IDT ตัวรับช่วงที่ n ให้ $O_{rx}(j)$ เป็นค่าเอาต์พุตจาก IDT ตัวรับที่เวลา j ซึ่งสามารถเขียนเทอม $O_{rx}(j)$ ได้ในรูปความสัมพันธ์กับจำนวนจุดและค่าของคลื่นบนอิลีกโทรดที่เวลา j ได้ดังนี้

$$O_{rx}(j) = K_{rx} \sum_{n=1}^N (L_n) \{F^+(n,j) - F^-(n,j)\} \quad \dots (5.8)$$

N เป็นจำนวนช่วง (Section) ใน IDT ตัวรับ

K_{rx} เป็น Conversion mode factor หรืออัตราส่วนของการ เปลี่ยนจากค่าตัวเลขของคลื่นเป็นค่าตัวเลขของสัญญาณไฟฟ้า

ให้ $F^+(n, j)$ เป็นผลรวมของค่าตัวเลขของคลื่นที่อิเล็กทรอนิกส์บวกของ
ช่วงที่ n ที่เวลา j .
 $F^-(n, j)$ เป็นผลรวมของค่าตัวเลขของคลื่นที่อิเล็กทรอนิกส์ลบของ
ช่วงที่ n ที่เวลา j

$$\begin{aligned} F^+(n, j) &= \sum_{i=0}^{P_n-1} C_T(J_n+i, j) \\ &= \sum_{i=0}^{P_n-1} \{C_r(J_n+i, j) + C_l(J_n+i, j)\} \end{aligned} \quad \dots(5.9)$$

$$\begin{aligned} F^-(n, j) &= \sum_{i=P_n+Q_n}^{2P_n+Q_n-1} C_T(J_n+i, j) \\ &= \sum_{i=P_n+Q_n}^{2P_n+Q_n-1} \{C_r(J_n+i, j) + C_l(J_n+i, j)\} \end{aligned} \quad \dots(5.10)$$

5.3 อัลกอริธึมการจำลอง IDT

5.3.1 อัลกอริธึมสำหรับการจำลอง IDT ตัวส่ง

จากรูป 5.1 เป็นการแสดงแต่ละจุดในช่วงที่ n ของ IDT ซึ่งประกอบด้วยทั้งหมด N ช่วง ทั้งนี้เป็นการจำลองโดยการพิจารณาจากการเคลื่อนที่ของคลื่นที่ต้องใช้การคำนวณและการเคลื่อนย้ายข้อมูลในแต่ละคาบเวลาการจำลอง โดยพิจารณาแต่ละช่วงจากสมการ 5.2, 5.5, 5.6 และ 5.7 ซึ่งใช้จำนวนครั้งการคำนวณคือ 1 คาบการจำลองดังนี้

1. การเคลื่อนย้ายข้อมูลสำหรับ N ช่วง ดังนั้นต้องมีการเคลื่อนย้าย

ข้อมูล $4 \sum_{n=1}^N (P_n + Q_n)$ ครั้งต่อ 1 คาบเวลาการจำลอง

2. การบวก $8N + 4 \sum_{n=1}^N P_n$ ครั้ง

$4 \sum_{n=1}^N P_n$ เป็นจำนวนครั้งการบวกที่ทุกจุดบนอิเล็กทรอนิกส์บวกและลบ

ด้วยสัญญาณที่ป้อนเข้าไปยังอิเล็กทรอนิกส์ตัวส่งสำหรับคลื่นที่เคลื่อนที่ทั้งสองทิศทาง ดังสมการ 5.5a และ 5.5b

$8N$ เป็นจำนวนการบวกสัญญาณที่เคลื่อนที่ในทิศทางหนึ่งกับสัญญาณที่สะท้อนที่ขอบอิเล็กทรอนิกส์

3. การคูณ $1 + 17N$ ครั้ง

โดย N ครั้งเป็นจำนวนครั้งที่ $I(j)$ คูณด้วย L_n/R_n ของแต่ละช่วงของ IDT

1 ครั้ง เป็นการคูณสัมประสิทธิ์การแปลง (Conversion factor, K_{TX}) ด้วยสัญญาณไฟฟ้า $S_{in}(j)$ ที่ป้อนเข้าไปยังอิเล็กทรอนิกส์ดังสมการ 5.4

16N ครั้ง เป็นการคูณเนื่องจากการสะท้อนและการลดทอนของสัญญาณที่ขอบอิเล็กทรอนิกส์

จะเห็นว่าการใช้เวลาในการคำนวณมาก จึงเสนออัลกอริธึมสำหรับการจำลอง IDT ตัวส่ง ซึ่งจะช่วยให้การคำนวณเร็วขึ้นและไม่ต้องใช้หน่วยความจำของคอมพิวเตอร์มากกว่าเดิม

พิจารณาคลื่นซึ่งเคลื่อนที่ไปทางขวา จากรูป 5.1 กำหนดให้ $C_r(k, j)$ ของเวลาที่น้อยกว่า 0 มีค่าเป็น 0 และให้พิจารณาที่ขอบขวาของอิเล็กทรอนิกส์ของช่วงที่ n คือจุด $J_n + P_n - 1$ จะแสดงค่าตัวเลขของคลื่นที่จุดนี้ได้ว่า

$$C_r(J_n + P_n - 1, j) = T_{FM} C_r(J_{n-1} + 2P_{n-1} + 2Q_{n-1} - 1, j - P_n) + \\ P_{n-1} \\ + R_{MF} C_1(J_n, j - P_n) + (L_n) \sum_{R_n, i=0}^{P_{n-1}} I(j-i) \quad \dots (5.11)$$

เมื่อ $C_r(J_{n-1} + 2P_{n-1} + 2Q_{n-1} - 1, j - P_n)$ เป็นค่าตัวเลขคลื่นซึ่งเคลื่อนที่ไปทางขวาที่จุดสุดท้ายของ IDT ช่วงที่ $n-1$ หมายถึงค่าตัวเลขของคลื่นที่ออกมาจากทางขวาของช่วงที่ $n-1$ ที่อยู่ก่อนหน้าช่วงที่ n นั้นเอง ถ้า n มีค่าเป็น 1 ค่าตัวเลขของคลื่นที่จะเข้ามาสู่ IDT ช่วงนี้ อาจเป็นคลื่นที่เกิดจากคลื่นซึ่งเคลื่อนที่ไปทางซ้ายแล้วสะท้อนกลับมาก็ได้ ถ้าคลื่นที่เคลื่อนที่ไปทางซ้ายไม่มีการสะท้อนกลับมาก็จะได้ค่าตัวเลขของคลื่นที่เข้ามาเป็นศูนย์

กำหนดให้

$$S_n(j) = (L_n) \sum_{R_n, i=0}^{P_{n-1}} I(j-i) \quad \dots (5.12)$$

เป็นผลรวมของสัญญาณอินพุตที่เข้าสู่ทรานสดิวเซอร์ตัวส่งในช่วงเวลา $j - (P_n - 1)$ ถึง j สามารถแสดงสมการ 5.11 ใหม่ได้เป็น

$$C_r(J_n + P_n - 1, j) = T_{FM} C_r(J_{n-1} + 2P_{n-1} + 2Q_{n-1}, j - P_n) + \\ + R_{MF} C_1(J_n, j - P_n) + S_n(j) \quad \dots (5.13)$$

ที่ขอบขวาของช่องว่างระหว่างอิเล็กทรอนิกส์ที่ติดกับอิเล็กทรอนิกส์ ค่าตัวเลขของคลื่นที่ไปทางขวาที่จุดนี้ จะเป็นผลรวมของค่าตัวเลขของคลื่นที่เคลื่อนที่ไปทางขวาจากจุด $J_n + P_n - 1$ กับพลังงานคลื่นที่เคลื่อนที่ไปทางซ้ายจากจุด $J_n + P_n$ ที่เวลา Q_n คาบเวลาก่อนหน้าแล้วสะท้อนกลับมา เขียนเป็นสมการได้

$$C_r(J_n + P_n + Q_n - 1, j) = T_{MF} C_r(J_n + P_n - 1, j - Q_n) + \\ + R_{FM} C_1(J_n + P_n, j - Q_n) \quad \dots (5.14)$$

ด้วยวิธีเดียวกันนี้ สามารถพิจารณาที่ขอบขวาของอิเล็กทรอนิกส์คือจุด $J_n + 2P_n + Q_n - 1$ และเขียนเป็นสมการแสดงค่าตัวเลขของคลื่นที่จุดนี้ได้ว่า

$$C_r(J_n + 2P_n + Q_n - 1, j) = T_{FM} C_r(J_n + P_n + Q_n - 1, j - P_n) + \\ + R_{MF} C_1(J_n + P_n + Q_n, j - P_n) - S_n(j) \quad \dots (5.15)$$

ที่ขอบขวาสุดของ IDT ช่วงที่ n แสดงค่าตัวเลขของคลื่นที่จุดนี้ได้
 ดังสมการ 5.16

$$C_r(J_n+2P_n+2Q_n-1, j) = T_{MF}C_r(J_n+2P_n+Q_n-1, j-Q_n) + \\ + R_{FM}C_1(J_n+P_n+Q_n, j-Q_n) \quad \dots (5.16)$$

ดังนั้น

$$S_n(j+1) = (\underline{L}_n) \sum_{R_n, i=0}^{P_n-1} I(j+1-i) \\ = (\underline{L}_n) \{I(j+1) - I(j-P_n)\} + S_n(j) \quad \dots (5.17)$$

ผลจากการคำนวณหาตัวเลขแทนสัญญาณออกจากช่วงที่ n ของคลื่นที่เคลื่อนที่ไปทางขวา จะใช้เป็นคลื่นที่เคลื่อนที่เข้าทางซ้ายสำหรับการคำนวณของช่วงที่ $n+1$ ถัดไป จนถึงช่วงสุดท้ายคือช่วงที่ N ซึ่งตัวเลขแทนสัญญาณจากช่วงที่ N นี้ จะเป็นคลื่นที่ออกจากบริเวณ IDT ตัวส่งทางขวา

ส่วนวิธีการจำลอง IDT ตัวส่งจะเริ่มจากการคำนวณตามสมการ 5.13, 5.14, 5.15 และ 5.16 ตามลำดับ โดยผลจากการคำนวณในสมการ 5.13 คือ $C_r(J_n+P_n-1, j)$ จะถูกห่อหุ้มด้วยเวลา Q_nZ เมื่อ Z เป็นค่าเป็นเวลาการจำลองแต่ละครั้ง แล้วนำมาใช้ในการคำนวณในสมการ 5.14 $C_r(J_n+P_n+Q_n-1, j)$ จะถูกห่อหุ้มด้วยเวลา P_nZ หลังจากนั้นนำมาใช้ในการคำนวณสมการ 5.15

$C_r(J_n+2P_n+Q_n-1, j)$ ที่คำนวณได้จากสมการ 5.15 จะถูกห่อหุ้มด้วยเวลา Q_nZ แล้วนำมาคำนวณในสมการ 5.16 และ $C_r(J_n+2P_n+2Q_n-1, j)$ ซึ่งได้มาจากการคำนวณในสมการ 5.16 ที่เวลา j จะเป็นค่าตัวเลขของคลื่นที่ออกทางขวาของ IDT ช่วงที่ n ที่เวลา $j+1$ ต่อไป

สำหรับคลื่นที่เคลื่อนที่ไปทางซ้ายของ IDT ช่วงที่ n จะเริ่มพิจารณาจากขอบซ้ายของช่องว่างระหว่างอิเล็กโทรดที่ติดกับขอบขวาของอิเล็กโทรดลบ ซึ่งจะเป็นคลื่นที่ออกจากทางซ้ายของ IDT ช่วงที่ $n+1$ ร่วมกับค่าตัวเลขคลื่นที่เคลื่อนที่ไปทางขวา จะเข้าสู่ IDT ช่วงที่ $n+1$ แล้วสะท้อนกลับ โดยมีเวลาหน่วง Q_nZ จึงมาถึงจุด $J_n+2P_n+Q_n$

$$C_1(J_n+2P_n+Q_n, j) = T_{MF}C_1(J_{n+1}, j-Q_n) + \\ + R_{FM}C_r(J_n+2P_n+2Q_n-1, j-Q_n) \quad \dots (5.18)$$

ส่วนที่ขอบซ้ายของอิเล็กทรอนิกส์สามารถแสดงค่าตัวเลขของคลื่นที่เคลื่อนที่ไปทางซ้ายได้ดังนี้

$$C_1(J_n+P_n+Q_n, j) = T_{FM}C_1(J_n+2P_n+Q_n, j-P_n) + R_{MF}C_r(J_n+2P_n+Q_n-1, j-P_n) - S_n(j) \quad \dots(5.19)$$

ที่ขอบซ้ายของช่องว่างระหว่างอิเล็กทรอนิกส์ที่ติดกับอิเล็กทรอนิกส์

คือจุด J_n+P_n

$$C_1(J_n+P_n, j) = T_{MF}C_1(J_n+P_n+Q_n, j-Q_n) + R_{FM}C_r(J_n+P_n+Q_n-1, j-Q_n) \quad \dots(5.20)$$

และที่จุดขอบซ้ายสุดของอิเล็กทรอนิกส์ที่จุด J_n

$$C_1(J_n, j) = T_{FM}C_1(J_n+P_n, j-P_n) + R_{MF}C_r(J_n+P_n-1, j-P_n) + S_n(j) \quad \dots(5.21)$$

สำหรับที่ขอบขวาสุดของช่วงสุดท้าย จากสมการ 5.18 นั้น อาจเป็นค่าตัวเลขของคลื่นที่เคลื่อนที่ออกจาก IDT ตัวส่งทางขวา แล้วสะท้อนกลับเข้ามาก็ได้

การจำลองคลื่นที่เคลื่อนที่ทางซ้าย จะเริ่มการจำลองโดยคำนวณจากสมการ 5.18 ถึง 5.21 ซึ่งผลจากการคำนวณของสมการ 5.18 จะถูกหน่วงด้วยเวลา P_nZ ให้นำมาใช้คำนวณในสมการ 5.19 และผลจากการคำนวณของสมการ 5.19 ก็จะหน่วงด้วยเวลา Q_nZ ให้นำมาใช้คำนวณในสมการ 5.20 และผลจากการคำนวณของสมการ 5.20 ก็จะหน่วงด้วยเวลา P_nZ ให้นำมาใช้คำนวณในสมการ 5.21 และผลลัพธ์ของการคำนวณจากสมการ 5.21 จะเป็นค่าตัวเลขของคลื่นที่เคลื่อนที่ไปทางซ้ายเข้าสู่ช่วงที่ $n-1$ จากนั้นคำนวณการเคลื่อนที่ของคลื่นในช่วงที่ $n-2, n-3$ จนได้ค่าตัวเลขที่ออกจากช่วงที่ 1 ซึ่งอยู่ทางซ้ายสุดของ IDT ในการคำนวณทั้งหมดคือ 1 คาบเวลาการจำลอง จะมีจำนวนครั้งของการคำนวณดังนี้

1. การบวก $16N$ ครั้ง

$2N$ ครั้ง จากการบวกเลขในสมการ 5.17

$14N$ ครั้งจากการบวก, ลบ ในสมการ 5.13, 5.14, 5.15,

5.16, 5.18, 5.19, 5.20 และ 5.21

2. การคูณ $17N+1$ ครั้ง

1 ครั้งจากการคูณในสมการ 5.4

N ครั้งจากการคูณในสมการ 5.17

$16N$ ครั้งจากการคูณในสมการ 5.13, 5.14, 5.15, 5.16

และ 5.18 ถึง 5.21

จะเห็นได้ว่าจำนวนครั้งของการบวกลดลงอย่างมากซึ่งเป็นการช่วยลดเวลาการจำลอง แต่จำนวนครั้งของการคูณยังมีการทำงานเท่าเดิม นอกจากนี้มีข้ออีกคือไม่มีการเคลื่อนย้ายข้อมูล อันเป็นลักษณะของคลื่นที่เคลื่อนที่และหน่วยความจำที่ใช้ก็เท่าเดิม ทั้งนี้เพราะหน่วยความจำแต่ละตำแหน่งจะใช้เก็บค่าตัวเลขของคลื่นในแต่ละจุดและแต่ละทิศทางคงเดิม ส่วนการเคลื่อนที่ของคลื่นก็จะใช้อัลกอริธึมเดียวกับข้อ 4.3 [2] ซึ่งแสดงให้เห็นการจัดการหน่วยความจำและการใช้ตัวชี้เพื่อจะช่วยให้ไม่เปลืองหน่วยความจำของคอมพิวเตอร์

5.3.2 อัลกอริธึมสำหรับการจำลอง IDT ตัวรับ

จากสมการ 5.8, 5.9 และ 5.10 ซึ่งเป็นสมการแสดงการเปลี่ยนจากพลังงานคลื่นเป็นประจุบน IDT ตัวรับ เมื่อพิจารณาเฉพาะการหาค่าตัวเลขเอาต์พุตจาก IDT ตัวรับ, การสะท้อนของคลื่นที่ขอบของอิเล็กโทรด และการเคลื่อนที่ต่อไปของคลื่นจากสมการ 5.6 และ 5.7 จะได้ว่า สำหรับ IDT ตัวรับจำนวน N ช่วงจะต้องใช้จำนวนครั้งการคำนวณต่อ 1 คาบการแบ่งเวลาดังนี้

$$1. \text{ การบวก } 8N + 4 \sum_{n=1}^N P_n$$

$8N$ เป็นจำนวนครั้งการบวกตัวเลขของคลื่นที่เคลื่อนที่ในทิศทางหนึ่งกับตัวเลขของคลื่นที่สะท้อนขอบของอิเล็กโทรด เช่น เกี่ยวกับการจำลอง IDT ตัวส่งในสมการ 5.6a, 5.6c, 5.7a, 5.7c

$$4 \sum_{n=1}^N P_n \text{ เป็นจำนวนการบวกที่ทุกๆจุดบน IDT บวกและลบ}$$

ซึ่งจะต้องนำค่าตัวเลขของแต่ละจุดมารวมกัน

$$2. \text{ การคูณ } 17N + 1 \text{ ครั้ง}$$

N ครั้งเป็นจำนวนครั้งที่คูณเทอมของ $\{F^+(n,j) - F^-(n,j)\}$ ด้วย L_n/R_n ในสมการ 5.8

$16N$ ครั้งเป็นจำนวนครั้งการคูณเนื่องจากคลื่นเกิดการสะท้อนและการลดทอนสัญญาณที่ขอบของอิเล็กโทรดบวกและลบ อันเนื่องมาจากการเคลื่อนที่ของคลื่นจากบริเวณตัวกลางหนึ่งไปยังอีกตัวกลางหนึ่ง

1 ครั้งจากการคูณตัวเลขของคลื่นด้วยสัมประสิทธิ์ตัวแปลงสัญญาณ ดังสมการ 5.8

N

3. การเคลื่อนย้ายข้อมูล 4 $\Sigma (P_n + Q_n)$ ครั้ง เท่ากับจำนวน
n=1

ของจุดทั้งหมดใน N ช่วงของ IDT ตัวรับ

การใช้วิธีการจัดการกับค่าตัวเลขของคลื่นที่จุดต่าง ๆ นั้น จะใช้วิธีคล้ายกับการจำลอง IDT ตัวส่ง โดยเริ่มพิจารณาจากจุดขวาสุดของอิลีกโทรคของรูป 5.1 เมื่อคลื่นเกิดการเคลื่อนที่ไป 1 จุด

$$C_r(J_n + P_n - 1, j) = T_{FM}C_r(J_{n-1} + 2P_{n-1} + 2Q_{n-1} - 1, j - P_n) + R_{MF}C_1(J_n, j - P_n) \dots (5.22)$$

ซึ่งคล้ายกับการจำลองคลื่นที่ IDT ตัวส่งในสมการ 5.13 เพียงแต่ต่างกันตรงที่ไม่มีสัญญาณไฟฟ้าเข้ามาที่สมการ 5.22 และค่าจำกัดความของจุด

$J_n + 2P_{n-1} + 2Q_{n-1} - 1$ จะเหมือนในสมการ 5.13

พิจารณาที่จุดขวาสุดของช่องว่างระหว่างอิลีกโทรค ซึ่งติดกับอิลีกโทรคลบ แสดงได้ว่า

$$C_r(J_n + P_n + Q_n - 1, j) = T_{MF}C_r(J_n + P_n - 1, j - Q_n) + R_{FM}C_1(J_n + P_n, j - Q_n) \dots (5.23)$$

พิจารณาที่ขอบขวาสุดของอิลีกโทรคลบ

$$C_r(J_n + 2P_n + Q_n - 1, j) = T_{FM}C_r(J_n + P_n + Q_n - 1, j - P_n) + R_{MF}C_1(J_n + P_n + Q_n, j - P_n) \dots (5.24)$$

ซึ่งคล้ายกับสมการ 5.15 และขอบขวาสุดของ IDT ตัวรับช่วงที่ n

$$C_r(J_n + 2P_n + 2Q_n - 1, j) = T_{MF}C_r(J_n + 2P_n + Q_n - 1, j - Q_n) + R_{FM}C_1(J_n + 2P_n + Q_n, j - Q_n) \dots (5.25)$$

ซึ่งคล้ายกับสมการ 5.16

ในสมการ 5.22, 5.23, 5.24 และ 5.25 เทอมที่เวลาเป็นลบจะให้เทอมนั้นเป็น 0 และในการคำนวณด้วยคอมพิวเตอร์สำหรับสมการ 5.22 ถึง 5.25 จะเหมือนกับการคำนวณในสมการ 5.13 ถึง 5.16 ซึ่งค่าที่ได้จากการคำนวณในแต่ละสมการ จะนำมาใช้ในการคำนวณของสมการถัดมา โดยมีการหวนเวลาของคลื่นที่เคลื่อนที่ออกจาก IDT ตัวรับทางขวา ซึ่งอาจสะท้อนกลับเข้ามาสู่ IDT ตัวรับได้อีก และสามารถนำมาคำนวณได้ต่อไป โดยเป็นคลื่นที่เคลื่อนที่จากขวาไปซ้าย ซึ่งสามารถเขียนความสัมพันธ์ได้คล้ายกับสมการ 5.18 ถึง 5.21 แสดงได้ดังนี้

ที่ขอบซ้ายของช่องว่างระหว่างอิเล็กโทรดซึ่งติดกับอิเล็กโทรดลบ แสดงการเคลื่อนที่ของคลื่นได้ดังนี้

$$C_1(J_n+2P_n+Q_n, j) = T_{MF}C_1(J_{n+1}, j-Q_n) + R_{FM}C_r(J_n+2P_n+2Q_n-1, j-Q_n) \dots (5.26)$$

ในสมการ 5.26 นี้ จุด J_{n+1} คือจุดที่อยู่นอก IDT ตัวรับทางด้านขวา จุดนี้อาจมีคลื่นที่เคลื่อนที่ไปทางซ้าย อันเนื่องจากการสะท้อนของคลื่นที่ไปทางขวา

ที่ขอบซ้ายของอิเล็กโทรดลบจะแสดงตัวเลขผลลัพธ์การเคลื่อนที่ของคลื่น

$$C_1(J_n+P_n+Q_n, j) = T_{FM}C_1(J+2P_n+Q_n, j-P_n) + R_{MF}C_r(J_n+2P_n+Q_n-1, j-P_n) \dots (5.27)$$

ที่ขอบซ้ายของช่องว่างระหว่างอิเล็กโทรดซึ่งติดกับอิเล็กโทรดบวก

$$C_1(J_n+P_n, j) = T_{MF}C_1(J_n+P_n+Q_n, j-Q_n) + R_{FM}C_r(J_n+P_n+Q_n-1, j-Q_n) \dots (5.28)$$

ที่ขอบซ้ายสุดของอิเล็กโทรด IDT ตัวรับช่วงที่ n จะแสดงได้ว่า

$$C_1(J_n, j) = T_{FM}C_1(J_n+P_n) + R_{MF}C_r(J_n+P_n-1, j-P_n) \dots (5.29)$$

$C_1(1, j)$ จะเป็นตัวเลขของคลื่นออกทางซ้ายของ IDT ตัวรับ ที่จะเคลื่อนที่ไปทางซ้ายผ่านดีเลย์ไลน์เข้าสู่ IDT ตัวส่งในสมการ 5.18 โดย IDT ตัวส่งจะมองคลื่นนี้เป็นคลื่นที่เคลื่อนที่ไปทางซ้ายที่ช่วง N

จะเห็นว่าจนถึงจุดนี้จำนวนครั้งของการคำนวณยังไม่เปลี่ยนแปลง นอกจากไม่มีการเคลื่อนย้ายข้อมูล เมื่อพิจารณาจากการเคลื่อนที่ของคลื่นและสมการ 5.9 จะสามารถเขียนสมการ 5.9 ใหม่ได้ว่า

$$\begin{aligned} F^*(n, j) &= \sum_{i=0}^{P_n-1} \{C_r(J_n+i, j+1) + C_1(J_n+i, j+1)\} \\ F^*(n, j+1) &= \sum_{i=0}^{P_n-1} \{C_r(J_n+i-1, j) + C_1(J_n+i+1, j)\} \\ &= \sum_{i=0}^{P_n-1} [\{C_r(J_n+i, j) + C_1(J_n+i, j)\} + \\ &\quad + \dots + \{C_r(J_n, j+1) - C_r(J_n+P_n-1, j)\} + \\ &\quad + \dots + \{C_1(J_n+P_n-1, j+1) - C_1(J_n, j)\}] \end{aligned}$$

$$F^+(n, j+1) = F^+(n, j) + C_r(J_n, j+1) - C_r(J_n+P_n-1, j) + \\ + C_1(J_n+P_n-1, j+1) - C_1(J_n, j) \quad \dots (5.30)$$

จะเห็นว่า $F^+(n, j+1)$ เขียนอยู่ในรูปของการเวียนเกิดของ $F^+(n, j)$ ซึ่งบวกด้วยค่าตัวเลขคลื่นที่เคลื่อนที่ไปทางขวาเข้าสู่ IDT และคลื่นที่เคลื่อนที่ไปทางซ้ายเข้าสู่ IDT ลบด้วยสัญญาณที่เคลื่อนที่ไปทางขวาและทางซ้ายออกจาก IDT

การเขียนในรูปของการเวียนเกิดนี้ จะช่วยลดเวลาการคำนวณและช่วยลด Quantizing error ที่เกิดขึ้นด้วย จากการพิจารณาการเคลื่อนที่ของคลื่นและสมการ 5.10 สามารถเขียนสมการ 5.10 ใหม่ได้ว่า

$$F^-(n, j) = \sum_{i=P_n+Q_n}^{2P_n+Q_n-1} \{C_r(J_n+i, j+1) + C_1(J_n+i, j+1)\} \\ F^-(n, j+1) = \sum_{i=P_n+Q_n}^{2P_n+Q_n-1} \{C_r(J_n+i-1, j) + C_1(J_n+i+1, j)\} \\ = \sum_{i=P_n+Q_n}^{2P_n+Q_n-1} \{[C_r(J_n+i, j) + C_1(J_n+i, j)] + \\ + \dots + \{C_r(J_n+P_n+Q_n, j+1) - C_r(J_n+2P_n+Q_n-1, j)\} + \\ + \dots + \{C_1(J_n+2P_n+Q_n-1, j+1) - C_1(J_n+P_n+Q_n, j)\}\} \\ = F^-(n, j) + C_r(J_n+P_n+Q_n, j+1) - C_r(J_n+2P_n+Q_n-1, j) + \\ + C_1(J_n+2P_n+Q_n-1, j+1) - C_1(J_n+P_n+Q_n, j) \quad \dots (5.31)$$

ซึ่ง $F^-(n, j+1)$ เขียนอยู่ในรูปการเวียนเกิดของ $F^-(n, j)$ รวมกับค่าตัวเลขของคลื่นที่เคลื่อนที่ไปทางซ้ายและขวาเข้าสู่อิเล็กทรอนิกส์ แล้วลบด้วยค่าตัวเลขของคลื่นที่เคลื่อนที่ไปทางซ้ายและขวาออกจากอิเล็กทรอนิกส์ ค่าที่ได้จากการคำนวณในสมการ 5.22 ถึง 5.29 จะถูกนำมาใช้แทนค่าในสมการ 5.30 และ 5.31 ทำให้การจำลอง IDT ได้รับจากสมการ 5.8 สามารถใช้อัลกอริทึมลดเวลาการคำนวณลง โดยแสดงได้ดังสมการ 5.32

N

$$O_{rx}(j+1) = K_{rx}[O_{rx}(j) + \sum_{n=1}^N [(L_n)\{C_r(J_n, j+1) - C_r(J_n+P_n-1, j)\} + \\ + \{C_1(J_n+P_n-1, j+1) - C_1(J_n, j)\} + \\ - \{C_r(J_n+P_n+Q_n, j+1) - C_r(J_n+2P_n+Q_n-1, j)\} + \\ + \{C_1(J_n+2P_n+Q_n-1, j+1) - C_1(J_n+P_n+Q_n, j)\}] \dots (5.32)$$

ในการคำนวณแต่ละคาบการแบ่งเวลา จะมีจำนวนการคำนวณดังนี้

1. การบวก $16N$ ครั้ง
2. การคูณ $17N+1$ ครั้ง

การจำลอง IDT ด้วยตัวอัลกอริธึมนี้ สามารถลดเวลาการคำนวณ และลดความผิดพลาด

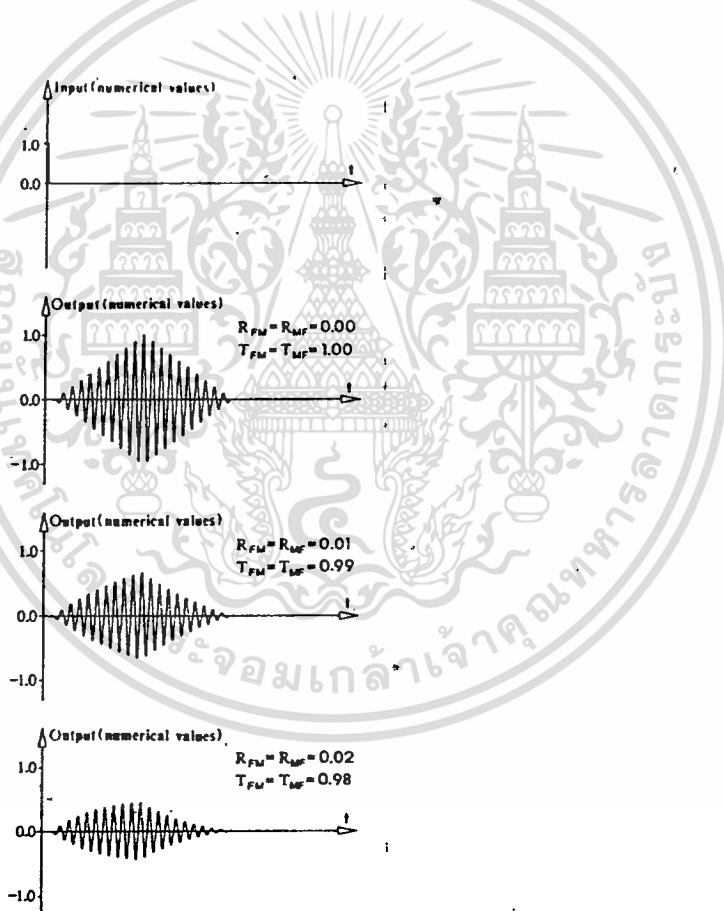
ระหว่าง IDT ตัวส่งและ IDT ตัวรับ จะเป็นคีย์ไลน์ซึ่งสามารถจำลองได้ด้วยอัลกอริธึมในบทที่ 4.3 โดยที่คลื่นที่เคลื่อนที่จากจุด $J_n+2P_n+2Q_n-1$ ของ IDT ตัวส่งไปทางขวาจะเป็นตัวเลขของคลื่นที่ออกทางขวาของ IDT ตัวส่งเข้าไปยังคีย์ไลน์ และคลื่นที่เคลื่อนที่ไปทางขวาออกจากคีย์ไลน์ก็จะเป็นค่าที่เข้าไปยัง IDT ตัวรับ แต่การจำลองคีย์ไลน์จะต้องมีหน่วยความจำ 2 ชุด สำหรับการเคลื่อนที่ของคลื่นใน 2 ทิศทาง ชุดหนึ่งสำหรับคลื่นที่เคลื่อนที่ไปทางขวา อีกชุดสำหรับคลื่นที่เคลื่อนที่ไปทางซ้าย คลื่นที่เคลื่อนที่ไปทางซ้ายออกจาก IDT ตัวรับจะเข้าไปยังหน่วยความจำของคีย์ไลน์ชุดที่ 2 และคลื่นที่ออกจากคีย์ไลน์ชุดที่สองจะเข้าสู่ IDT ตัวส่งที่จุด $J_n+2P_n+2Q_n-1$ ดังในสมการ 5.18 สำหรับคลื่นที่เคลื่อนที่ไปทางขวาออกจาก IDT ตัวส่ง จะเข้าไปยังหน่วยความจำของคีย์ไลน์ชุดที่ 1 และคลื่นที่ออกจากคีย์ไลน์ชุดที่ 1 จะเข้าสู่ IDT ตัวรับที่จุด J_1

จากอัลกอริธึมดังกล่าวเราจะศึกษาคุณสมบัติของ IDT ดังผลการจำลองต่อไปนี้

5.4 การจำลอง IDT แบบมีความกว้าง

การจำลองที่ 1

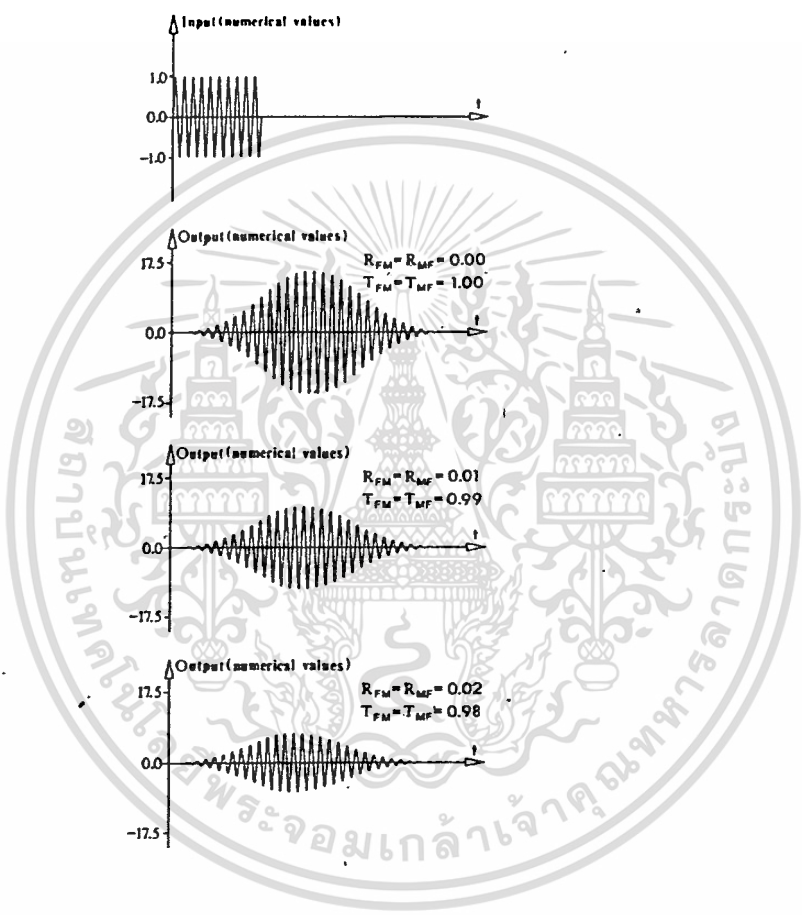
เป็นการจำลอง IDT ตัวส่งและ IDT ตัวรับ โดยการป้อนตัวเลขค่า 1.0 เข้าไปยัง IDT ตัวส่งแบบนอนคิสเพอร์ซีฟตี้เลย์ไลน์และนอนอะโพไดซ์ จำนวน 10 คู่ฟังก์เกอร์หรือ 10 ช่วง ค่าตัวเลขนี้จะป้อนเข้าไป 20 คาบเวลา การจำลอง ค่าตัวเลขของคลื่นที่ออกจาก IDT ตัวส่งจะถูกป้อนเข้าไปยัง IDT ตัวรับที่มีจำนวน 10 คู่ฟังก์เกอร์ ทั้ง IDT ตัวส่งและตัวรับมีอัตราส่วน a/b ดังรูป 5.1 เท่ากับ 1.0 ในแต่ละช่วงของ IDT จะแบ่งออกเป็น 100 จุด ผลการจำลองซึ่งเป็นสัญญาณออกจาก IDT ตัวรับได้ผลการจำลองที่มีผลตอบสนอง ดังรูป 5.3 โดยการจำลองได้ให้อัตราการเคลื่อนที่ผ่านและการสะท้อนมีค่าต่างๆ



รูป 5.3 ผลการจำลองที่ 1

การจำลองที่ 2

เป็นการจำลอง IDT ตัวส่งและ IDT ตัวรับที่มีคุณสมบัติเหมือนกับการจำลองที่ 1 แต่ในการจำลองที่ 2 นี้จะป้อนสัญญาณไซน์ (Sine wave) จำนวน 10 ไซเคิลเข้าไปที่ IDT ตัวส่ง ความยาวคลื่นของสัญญาณแต่ละไซเคิลเท่ากับ 1 ช่วงของ IDT ซึ่งในการศึกษาได้ทำการจำลองกรณีการสะท้อนมีค่าต่าง ๆ ส่วนผลการจำลองซึ่งเป็นสัญญาณออกจาก IDT ตัวรับมีดังรูป 5.4

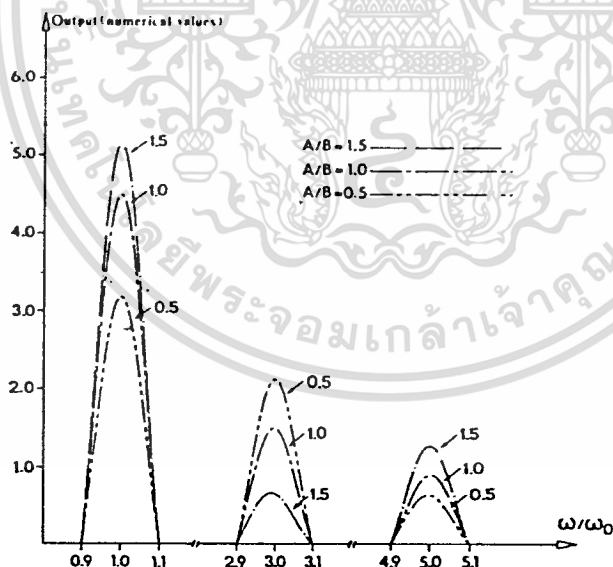


รูป 5.4 ผลการจำลองที่ 2

รูป 5.4 แสดงผลการจำลองที่ 2 จะเห็นว่าแอมพลิจูดของเอาต์พุตและขนาดจะต่างกันขึ้นกับ R_{FM}, R_{MF}, T_{FM} และ T_{MF} ซึ่งตรงกับผลตอบสนองในความเป็นจริงดังในบทความ [3]

การจำลองที่ 3

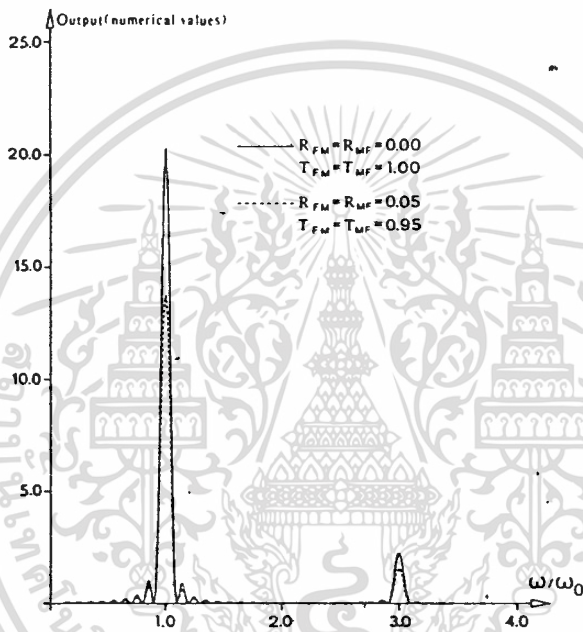
เป็นการจำลองผลของความกว้างของอิเล็กโทรดต่อช่องว่างระหว่างอิเล็กโทรดต่างๆ กัน ซึ่งสามารถใช้ในการจำลองทรานสดิวเซอร์เมื่อความเร็วคลื่นบนผิวอิเล็กโทรดเปลี่ยนไป เสมือนว่าทรานสดิวเซอร์มีความกว้างของอิเล็กโทรดไม่เท่ากับในความเป็นจริง ดังนั้นอัตราส่วนของความกว้างระหว่างอิเล็กโทรดและช่องว่างในการจำลองของฟังก์เจอร์จึงขึ้นกับ $\Delta v/v$ ของความเร็วคลื่นบนอิเล็กโทรด การจำลองหาผลตอบสนองต่อความถี่ของ IDT ตัวรับที่มี 10 จำนวนช่วงจึงสามารถหาผลตอบสนองต่อความถี่ โดยการป้อนค่าตัวเลขของสัญญาณขาเข้าอย่างต่อเนื่องที่ความถี่ต่างๆ กันเข้าไป แล้วนำเอาขนาดยอดของสัญญาณมาพลอต โดยที่อัตราส่วน a/b ของ IDT ตัวรับในรูป 5.1 เท่ากับ 0.5, 1.0 และ 1.5 ตามลำดับ ซึ่งมีค่าการสะท้อนเป็นศูนย์ จะเห็นว่าอัตราส่วนความกว้างของอิเล็กโทรดมีผลต่อการตอบสนองความถี่ดังในบทความ [3] และเมื่อเปรียบเทียบผลการจำลองกับผลของทฤษฎีในบทความดังกล่าวแล้วใกล้เคียงกัน จากผลของความกว้างของฟังก์เจอร์ทำให้ผลตอบสนองความถี่มีขนาดยอดสัญญาณสูงสุดที่ฮาร์โมนิกเป็นเลขคี่เท่านั้น ดังที่ได้กล่าวในบทที่ 4.1 ฮาร์โมนิกที่ 1 คือสัญญาณที่มีความยาวคลื่นเท่ากับความกว้าง 1 ช่วง ผลการจำลองมีดังรูป 5.5



รูป 5.5 ผลการจำลองที่ 3

การจำลองที่ 4

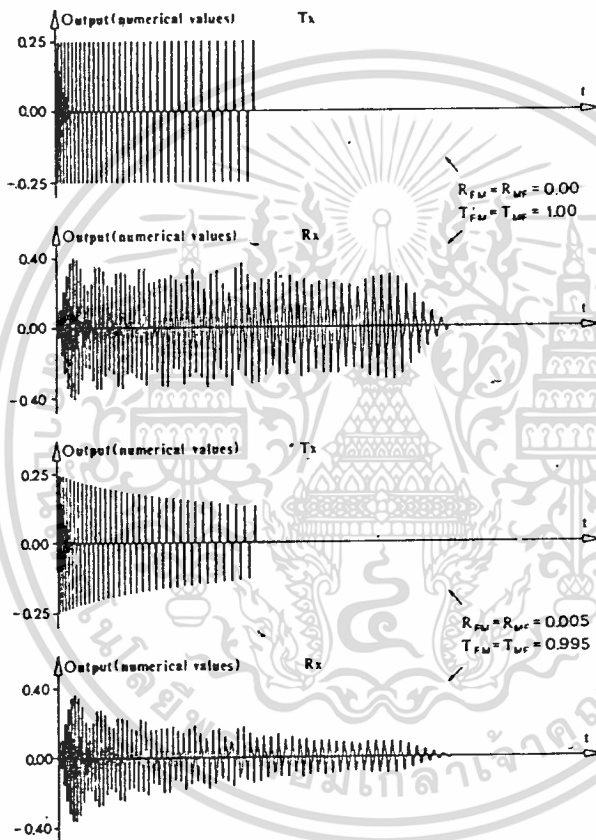
เป็นการจำลองเพื่อศึกษาผลของการสะท้อนคลื่นและจำนวนช่วงที่มีผลต่อการตอบสนองความถี่ โดยมีตัวรับและตัวส่งด้านละ 10 คู่ฟิงเกอร์ IDT และมีอัตราส่วนของ a/b ในรูป 5.1 เท่ากับ 1.0 แต่ในการจำลองจะศึกษากรณีที่อัตราส่วนการสะท้อนมีค่าต่างๆ ผลการจำลองมีดังรูป 5.6 จะเห็นว่า Side lobe ของฮาร์โมนิกที่ 1 กับฮาร์โมนิกที่ 3 มีค่าลดลง นั่นคือค่าตัวประกอบคุณภาพของดีเลย์ไลน์มีค่ามากขึ้น และการสะท้อนทำให้ผลตอบสนองต่อความถี่มีค่านายาคออสัญญาณของฮาร์โมนิกที่ 1 และ 3 ลดลง



รูป 5.6 ผลการจำลองที่ 4

การจำลองที่ 5

เป็นการจำลอง IDTแบบคิสเพอร์ซีฟฟิลด์ไลน์จำนวน 33 คู่ฟังก์เตอร์ โดย IDT ช่วงที่ 1 กว้างเท่ากับ $2\lambda_0$ และช่วงสุดท้ายคือช่วงที่ 33 จะมีความกว้างเท่ากับ λ_0 ระหว่างช่วงที่ 1 ถึงช่วงที่ 33 จะมีความกว้างของแต่ละช่วงลดลงเป็นเชิงเส้น โดยในการจำลองจะให้ IDT ช่วงที่ 1 มี 256 จุดและลดลงช่วงละ 4 จุดจนถึง IDT ช่วงที่ 33 การจำลองจะป้อนตัวเลข 1.0 เข้าไปที่ IDT ตัวส่งเป็นเวลา 20 คาบเวลาการจำลอง โดยถือเสมือนเป็นสัญญาณอิมพัลส์



รูป 5.7 ผลการจำลองที่ 5

รูป 5.7a คลื่นที่ออกจาก IDT ตัวส่ง และเอาท์พุทหนึ่งจะไปเข้าสู่ IDT ตัวรับซึ่งมีลักษณะเหมือนกันแต่ความกว้างในแต่ละช่วงของ IDT มีค่าเพิ่มขึ้นจาก λ_0 ถึง $2\lambda_0$ ได้ผลตอบสนองกับ รูป 5.7b

รูป 5.7c และ 5.7d เป็นการศึกษาผลเมื่อการเคลื่อนที่ของคลื่น เกิดการสะท้อนที่ขอบของอิล็กโทรด

5.5 เอกสารอ้างอิงบทที่ 5

- [1] Morgan D.P., "Log-Periodic Transducers for Acoustic Surface Wave," Proc. IEEE, Vol.119, No.1, Jan. 1972.
- [2] Pookaiyaudom S. and Jantarang S., "Simulation for the Transient Properties of Multi-mode Delay-Line Controlled Oscillators," Int.J. Electronics, Vol.66, No.1, Jan. 1989.
- [3] Tseng C.C., "Frequency Response of an Interdigital Transducer for Excitation of Surface Elastic Waves," IEEE Trans. Electronic Device, Vol.ED-15, No.8, pp.586-594, Aug. 1968.



บทที่ 6

โปรแกรมวิเคราะห์เน็ตเวิร์คด้วยสัญลักษณ์
(Symbolic Network Analysis Program, SNAP)

6.1 บทนำโปรแกรมวิเคราะห์เน็ตเวิร์คด้วยสัญลักษณ์

การวิเคราะห์วงจรไฟฟ้าด้วยคอมพิวเตอร์ในปัจจุบันนั้น นิยมใช้วิธีการวิเคราะห์เชิงตัวเลข (Numerical method) เช่นโปรแกรม SPICE แต่ไม่สามารถบอกถึงคุณสมบัติของวงจรบางอย่างได้โดยตรงเช่น ลำดับความสำคัญของอุปกรณ์ในเน็ตเวิร์ค สำหรับการศึกษาเรื่องการใช้อุปกรณ์ในการวิเคราะห์วงจรไฟฟ้าในรูปของสัญลักษณ์นั้น ได้มีการศึกษามากกว่า 30 ปี [1] โดยเป็นการศึกษาในรูปของการสร้างสมการสถานะสมมูล (Equivalent state equation) แล้วหาผลลัพธ์โดยวิธีการ Interpolation ซึ่งจากสมการสถานะสมมูลจะสามารถวิเคราะห์วงจรได้ ทั้งใน Time domain และ Frequency domain ทั้งนี้ผู้ใช้ต้องเขียนโปรแกรมขึ้นจากการได้ทราบความสัมพันธ์ของโวลเตจและกระแสแต่ละจุดเสียก่อน แล้วจึงเรียกโปรแกรมย่อยที่ผู้เขียนบทความ [1] ได้เขียนไว้ก่อนหน้านี้ ซึ่งการวิเคราะห์วงจรไฟฟ้าในรูปของสัญลักษณ์ด้วยวิธีดังกล่าว สามารถวิเคราะห์ได้ทั้งในวงจรแอนะลอกและดิจิทัล แต่วิธีการหาค่าตอบแบบวิเคราะห์เชิงตัวเลขโดยวิธีกฎของแครมเมอร์ (Cramer's rule) ไม่สามารถใช้กับเน็ตเวิร์คที่มีสัญลักษณ์มากๆได้ เพราะต้องใช้หน่วยความจำจำนวนมาก สำหรับในส่วนของบทความ [2] ได้เสนอวิธีการวิเคราะห์วงจรด้วยสัญลักษณ์ โดยใช้วิธีวิเคราะห์จากการเขียนสมการของโวลเตจที่แต่ละศูนย์รวม (Node) แต่ยังคงเป็นลักษณะที่ต้องใช้การเขียนแจกแจงการต่อของอุปกรณ์ในวงจรที่เรียกว่าแฟ้มข้อมูลเข้า (Input file) ในแบบที่ช่วยคอมพิวเตอร์ จากนั้นโปรแกรมซึ่งเขียนในภาษา Fortran จะเปลี่ยนแฟ้มข้อมูลอินพุตให้เป็นแฟ้มข้อมูลที่ใช้ได้กับโปรแกรม ASnap ซึ่งเขียนจากภาษาลิสป์ ทำให้ไม่สะดวก

วิทยานิพนธ์ฉบับนี้เสนอโปรแกรม SNAP ซึ่งเป็นโปรแกรมที่สามารถวิเคราะห์วงจรอิเล็กทรอนิกส์ใน Frequency domain และค่าของอุปกรณ์ในวงจรจะกำหนดให้เป็นค่าจริงของอุปกรณ์หรือติดอยู่ในรูปของสัญลักษณ์ก็ได้ ผลลัพธ์จากการวิเคราะห์วงจรด้วยโปรแกรม SNAP จะแสดงโวลเตจที่แต่ละศูนย์รวมของวงจร โดยโปรแกรม SNAP เป็นโปรแกรมที่ประกอบด้วย 2 ส่วน ส่วนหนึ่งคือโปรแกรม Editor ที่ใช้ในการสร้างแฟ้มข้อมูลสำหรับเก็บลักษณะการต่อ

ของอุปกรณ์ในเน็ตเวิร์ค, ตรวจสอบรูปแบบ (Format) ของการต่ออุปกรณ์แต่ละตัวรวมทั้งการสร้างแฟ้มข้อมูลไลบรารี (Library file) ของทรานซิสเตอร์, FET และ MOSFET ด้วย แฟ้มข้อมูลที่เก็บลักษณะการต่อของอุปกรณ์ในเน็ตเวิร์คจะสามารถเก็บในแผ่นแม่เหล็กในรูปของแอสกีไฟล์(ASCII file)จึงสามารถที่จะเขียนแฟ้มข้อมูลสำหรับเก็บลักษณะการต่อของอุปกรณ์ด้วยโปรแกรมประมวลผลคำ (Word processor) อื่นได้ สำหรับโปรแกรม SNAP ส่วนที่สองจะทำการวิเคราะห์วงจร ซึ่งสร้างเก็บไว้เป็นแฟ้มข้อมูลในแผ่นแม่เหล็กด้วยโปรแกรม Editor ในส่วนแรก วิธีการวิเคราะห์วงจรจะใช้กฎข้อที่ 2 ของเคอร์ชอฟ (Kirchoff's second law) สำหรับการเขียนสมการกระแสที่แต่ละศูนย์รวม และใช้วิธีการเก็บสัญลักษณ์ของตัวแปรในคอมพิวเตอร์แบบในบทความ [3]



6.2 การแทนอุปกรณอิเล็กทรอนิกส์แบบสัญลักษณ์ในคอมพิวเตอร์

จากบทความ [3] ได้แนะนำการคำนวณแบบสัญลักษณ์ (Symbolic) โดยเสนอการแทนชุดของตัวแปรด้วยชุดของค่ายกกำลัง (Exponent) ของตัวแปรนั้น โดยเรียงตามลำดับตัวแปรที่มีการกำหนดไว้อย่างแน่นอนและนำหน้าด้วยสัมประสิทธิ์ของชุดตัวแปร สมการที่ต้องการเก็บไว้ในรูปสัญลักษณ์อาจประกอบด้วยหลายเทอม และจะสามารถเขียนสมการนั้นได้ว่า

$$Y = \sum_{i=1}^N M_i \quad \dots(6.1)$$

Y = สมการที่ต้องการเก็บไว้ในรูปสัญลักษณ์ประกอบด้วยเทอมทั้งหมด N เทอม

M_i = $(C_i, b_{i1}, b_{i2}, \dots, b_{in})$ คือเทอมที่ i ของสมการ Y ซึ่งประกอบด้วยสัมประสิทธิ์ (C_i) และค่ายกกำลังของตัวแปร (b_{in})

C_i = ค่าสัมประสิทธิ์ของเทอมที่ i ในสมการ Y ค่านี้เป็นเลขจำนวนจริงหรือจำนวนเต็มก็ได้

b_{ik} = ค่ายกกำลังของตัวแปรที่ k ของเทอมที่ i b_{ik} มีค่าเป็นจำนวนเต็ม โดยที่ $k = 1, 2, 3, \dots, n$

การเก็บสมการแบบสัญลักษณ์ด้วยวิธีที่กล่าวมาไว้ในคอมพิวเตอร์ จะทำได้โดยการจองหน่วยความจำสำหรับแต่ละเทอมเพื่อเก็บค่าสัมประสิทธิ์และค่ายกกำลังของแต่ละตัวแปร ซึ่งค่าในหน่วยความจำเป็นตัวเลขจึงทำให้สามารถนำมาบวก, ลบ, คูณ หรือหารได้ โดยการบวกและลบเป็นการนำเอาเทอม 2 เทอมมาบวกหรือลบกัน ถ้าทั้ง 2 เทอมมีจำนวนตัวแปรเท่ากันและค่ายกกำลังของตัวแปรแต่ละตัวในทั้งสองเทอมเท่ากันแบบ 1 ต่อ 1 จะนำเอาสัมประสิทธิ์ของทั้งสองเทอมมาบวกหรือลบกัน เช่นกำหนดให้

$$Y1 = 2.5 V_1 V_2^2 V_3 = (2.5, 1, 2, 1) \quad \dots(6.2a)$$

$$Y2 = 3.0 V_1 V_2^2 V_3 = (3.0, 1, 2, 1) \quad \dots(6.2b)$$

$$Y3 = 4.0 V_1 = (4.0, 1, 0, 0) \quad \dots(6.2c)$$

$$\begin{aligned} \text{ดังนั้น } Y1 + Y2 &= 2.5 V_1 V_2^2 V_3 + 3.0 V_1 V_2^2 V_3 \\ &= (2.5, 1, 2, 1) + (3.0, 1, 2, 1) \\ &= (5.5, 1, 2, 1) \\ &= 5.5 V_1 V_2^2 V_3 \end{aligned}$$

$$\begin{aligned} Y1 - Y2 &= 2.5 V_1 V_2^2 V_3 - 3.0 V_1 V_2^2 V_3 \\ &= (2.5, 1, 2, 1) - (3.0, 1, 2, 1) \\ &= (-0.5, 1, 2, 1) \\ &= -0.5 V_1 V_2^2 V_3 \end{aligned}$$

ถ้าค่าเลขยกกำลังของแต่ละตัวแปรในสองเทอมที่นำมาบวกหรือลบกันมีค่าไม่เหมือนกัน ผลลัพธ์จากการบวกหรือลบก็จะมี 2 เทอมเช่นจากสมการ 6.2a และ 6.2c เมื่อนำมาบวกหรือลบกันจะได้ผลลัพธ์ดังนี้

$$\begin{aligned} Y1 \pm Y3 &= 2.5 V_1 V_2^2 V_3 \pm 4.0 V_1 \\ &= (2.5, 1, 2, 1) \pm (4.0, 1, 0, 0) \end{aligned}$$

ในการคูณของสองเทอมจะนำเอาสัมประสิทธิ์ของแต่ละเทอมมาคูณกัน และนำเอาค่ายกกำลังของตัวแปรตำแหน่งเดียวกันมาบวกกัน สำหรับกรณีที่เป็น การหารก็จะเอาสัมประสิทธิ์ของแต่ละเทอมมาหารกันและนำเอาค่ายกกำลังของ ตัวแปรตำแหน่งเดียวกันมาลบกัน เช่นจากสมการ 6.2a และ 6.2c เมื่อนำมา คูณและหารกันจะได้ผลลัพธ์ดังนี้

$$\begin{aligned} Y1 * Y3 &= 2.5 V_1 V_2^2 V_3 * 4.0 V_1 \\ &= (2.5 * 4.0, 1 + 1, 2 + 0, 1 + 0) \\ &= (10.0, 2, 2, 1) \\ &= 10.0 V_1^2 V_2^2 V_3 \\ Y1 / Y3 &= 2.5 V_1 V_2^2 V_3 / 4.0 V_1 \\ &= (2.5 / 4.0, 1 - 1, 2 - 0, 1 - 0) \\ &= (0.625, 0, 2, 1) \\ &= 0.625 V_2^2 V_3 \end{aligned}$$

จะเห็นว่าผลลัพธ์ก็เหมือนกับการคำนวณด้วยมือมันเอง แต่การแทน สัญลักษณ์ให้อยู่ในรูปของตัวเลขยกกำลังตามตำแหน่งของตัวแปรจะมีประโยชน์ ในการทำงานด้วยคอมพิวเตอร์ เพราะคอมพิวเตอร์สามารถประมวลผลกับตัวเลข เท่านั้น และวิธีการแทนสัญลักษณ์ด้วยวิธีดังกล่าวทำให้สามารถทำงานกับจำนวน เลขที่เป็นรูปของโพลิโนเมียล (Polynomial) ด้วยสัญลักษณ์ในคอมพิวเตอร์ได้ นอกจากนี้ยังสามารถวิเคราะห์ความไม่เชิงเส้นในวงจร

6.3 คณิตศาสตร์ของเน็ตเวิร์ค

จากกฎของเคอร์ชอฟ (Kirchoff's law)

1. ผลรวมความต่างศักย์ไฟฟ้าแต่ละทางเดิน (Path) ในทางเดินครบรอบ (Closed path) จะมีค่าเป็นศูนย์
 2. ผลรวมของกระแสที่ศูนย์รวม (Node) ใดๆ จะมีค่าเป็นศูนย์
- สำหรับระบบซึ่งมีหลายอินพุตและหลายเอาต์พุต สามารถเขียนความสัมพันธ์ได้ในรูปของเมตริกซ์

จากกฎข้อที่ 1 ของเคอร์ชอฟ (Kirchoff's first law) จะแสดงได้ว่า

$$V = A^t e \quad \dots (6.3)$$

เมื่อ V = Branch voltage matrix เป็นเมตริกซ์ที่บอกค่าโวลเตจตกตลอดแต่ละ Branch เมตริกซ์ V มีขนาด $B \times 1$ เมื่อ B เป็นจำนวน Branch ในวงจร แต่ละองค์ประกอบ (Element) ของเมตริกซ์ V จะสามารถเขียนได้เป็น V_{i1} หมายถึงองค์ประกอบในแถว (Row) ที่ i และคอลัมน์ (Column) ที่ 1 ซึ่งเป็นโวลเตจตกตลอด Branch ที่ i โดย i มีค่าได้ตั้งแต่ 1 ถึง B

A^t = ทรานสโพส (Transpose) ของเมตริกซ์ A หรือ Reduced incident matrix เมตริกซ์ A ซึ่งมีขนาด $B \times N$ เมื่อ N เป็นจำนวนศูนย์รวมในเน็ตเวิร์คที่ไม่รวม Datum node เมตริกซ์ A แสดงทิศทางของกระแสในแต่ละ Branch ซึ่งแต่ละองค์ประกอบในเมตริกซ์จะมีข้อกำหนดคือ

$$a_{ik} = \begin{cases} 1 & \text{ถ้า branch } k \text{ มีทิศทางของกระแสออกจากศูนย์รวม } i \\ -1 & \text{ถ้า branch } k \text{ มีทิศทางของกระแสเข้าหาศูนย์รวม } i \\ 0 & \text{ถ้า branch } k \text{ ไม่ต่อกับศูนย์รวม } i \end{cases} \dots (6.4)$$

e = Node to datum matrix เป็นเมตริกซ์แสดงระดับโวลเตจเทียบกับกราวด์ (Ground) ของแต่ละศูนย์รวม เมตริกซ์ e มีขนาด $N \times 1$ แต่ละองค์ประกอบของเมตริกซ์แสดงได้เป็น e_{i1} หมายถึงองค์ประกอบในแถวที่ i และคอลัมน์ที่ 1 ซึ่งเป็นระดับโวลเตจที่ศูนย์รวมที่ i เทียบกับกราวด์ i จะมีค่า 1 ถึง N

จากกฎข้อที่ 2 ของเคอร์ชอฟ (Kirchoff's second law) แสดงได้ว่า

$$AJ = 0 \quad \dots(6.5)$$

J = Branch current vector matrix เป็น Diagonal matrix ขนาด $B \times B$ ซึ่งสามารถบอกขนาดของกระแสในแต่ละ Branch โดยกระแสในแต่ละ Branch นั้นเป็นผลรวมของกระแสจากแหล่งจ่ายกระแส, แหล่งจ่ายโวลเตจ และกระแสที่ไหลผ่านอุปกรณ์ใน Branch นั้นๆ เมตริกซ์ J จะสามารถเขียนได้ดังสมการ 6.6

$$J = GV + J_s - GV_s \quad \dots(6.6)$$

G = Branch conductance matrix เป็น Diagonal matrix ขนาด $B \times B$ เมตริกซ์นี้เป็นเมตริกซ์แสดงขนาดความนำไฟฟ้าในแต่ละ Branch แต่ละองค์ประกอบในเมตริกซ์จะเขียนได้เป็น G_{ij}

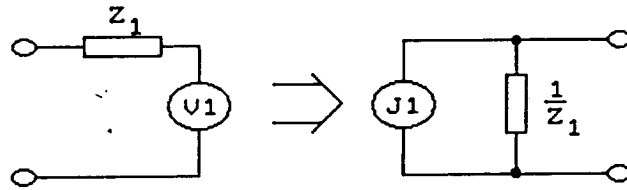
$$G_{ij} = \begin{cases} 0 & ; i \neq j \\ \text{ค่าความนำไฟฟ้าใน Branch } i & ; i = j \end{cases}$$

$i = 1..B, j = 1..B$

J_s = Current source vector matrix มีขนาด $B \times 1$ เป็นเมตริกซ์แสดงแหล่งจ่ายกระแสใน Branch ต่างๆ ทั้งที่เป็นแบบ Dependent และ Independent current source แต่ละองค์ประกอบของเมตริกซ์เขียนได้เป็น J_{s11} หมายถึงแหล่งจ่ายกระแสใน Branch ที่ i

V_s = Voltage source vector matrix มีขนาด $B \times 1$ เป็นเมตริกซ์แสดงแหล่งจ่ายโวลเตจใน Branch-ต่างๆทั้งที่เป็นแบบ Dependent และ Independent current source แต่ละองค์ประกอบของเมตริกซ์เขียนได้เป็น V_{s11} หมายถึงแหล่งจ่ายโวลเตจใน Branch ที่ i

ในการสร้างเมตริกซ์เหล่านี้จะสร้างโดยวิธี Nodal analysis ซึ่งอาศัยกฎข้อที่ 2 ของเคอร์ชอฟ ทั้งนี้แหล่งจ่ายโวลเตจจะถูกเปลี่ยนเป็นแหล่งจ่ายกระแส ด้วยวิธีการทรานส์ฟอร์มดังตัวอย่างรูป 6.1



รูป 6.1 แหล่งจ่ายโวลเตจเปลี่ยนเป็นแหล่งจ่ายกระแส

แหล่งจ่ายโวลเตจ V_1 ซึ่งอนุกรมอยู่กับอิมพีแดนซ์ Z_1 จะเปลี่ยนเป็นแหล่งจ่ายกระแสขนาด (V_1 / Z_1) และต่อขนานอยู่กับแอดมิทแตนซ์ (Admittance) ค่า $\frac{1}{Z_1}$ สำหรับกรณีที่ Branch นั้นมีเฉพาะแหล่งจ่ายโวลเตจ Z_1

จะต้องทำการย้ายแหล่งจ่ายโวลเตจไปยัง Branch ที่มีอิมพีแดนซ์ต่ออยู่กับศูนย์รวมโหนดศูนย์รวมหนึ่งของแหล่งจ่ายโวลเตจดังรูป 6.2

จากสมการ 6.5 และ 6.6 สามารถเขียนสมการซึ่งเปลี่ยนแหล่งจ่ายโวลเตจเป็นแหล่งจ่ายกระแสโดยวิธี Nodal analysis ได้ดังนี้

$$AJ = AGV_s + AJ_s - AGV_s = 0 \quad \dots(6.7)$$

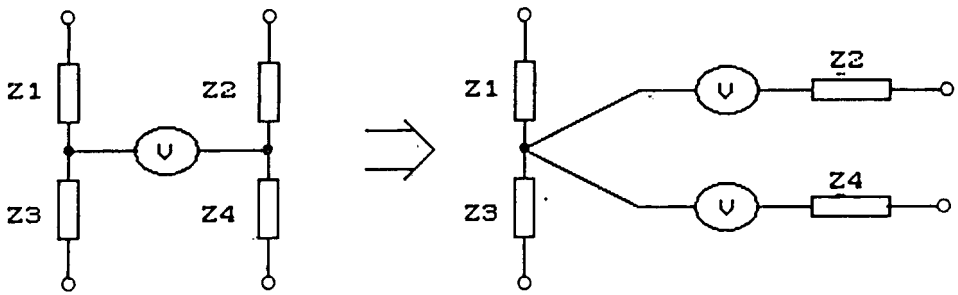
จาก $V = A^t e$ ดังนั้นสมการ 6.7 จะสามารถเขียนใหม่ได้เป็น

$$AJ = AGA^t e + AJ_s - AGV_s = 0 \quad \dots(6.8)$$

$$AGA^t e = AGV_s - AJ_s$$

ซึ่งเมตริกซ์ $AGA^t e$ มีขนาด $N \times 1$

เมตริกซ์ AGV_s และ $-AJ_s$ เป็นเมตริกซ์ ขนาด $N \times 1$



รูป 6.2 แสดงการย้ายแหล่งจ่ายโวลเตจ

เมื่อกำหนดให้เมตริกซ์ $Y_n = AGA^t$ และเมตริกซ์

$$I_s = AGV_s - AJ_s$$

จะสามารถเขียนสมการแสดงความสัมพันธ์ระหว่างโวลเตจที่ศูนย์รวมและแหล่งจ่ายกระแสได้ดังนี้

$$Y_n e = I_s \quad \dots (6.9)$$

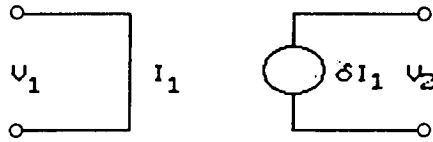
$$e = Y_n^{-1} I_s \quad \dots (6.10)$$

เมตริกซ์ e เป็นเมตริกซ์ที่แสดงโวลเตจของทุกๆ ศูนย์รวมในวงจรและเป็นคำตอบที่ต้องการ

แหล่งจ่ายกระแสและแหล่งจ่ายโวลเตจที่กล่าวมาแล้วเป็นแบบ Independent voltage source และ Independent current source ซึ่งใช้เป็นสัญญาณป้อนให้กับวงจร แต่ยังมีแหล่งจ่ายอีกแบบ คือ Controlled source หรือ Dependent source ซึ่งเป็น Resistive two-port element ประกอบอยู่ใน Branch 2 Branch Branch หนึ่งเรียกว่า Primary branch อาจเป็นแบบวงจรเปิด (Open circuit) หรือ วงจรปิด (Short circuit) ก็ได้ และอีก Branch เรียกว่า Secondary branch เป็นแหล่งจ่ายกระแสหรือแหล่งจ่ายโวลเตจซึ่งควบคุมโดยกระแสหรือโวลเตจใน Primary branch

Controlled source มี 4 ชนิดคือ -

1. Current - controlled voltage source (CCVS)



รูป 6.3 Current controlled voltage source (CCVS)

รูป 6.3 แสดงแหล่งจ่ายโวลเตจทางด้าน Secondary branch ทางขวาจะถูกควบคุมด้วยกระแส i_1 ทางซ้าย ด้วยความสัมพันธ์ดังสมการ 6.11

$$V_2 = \delta I_1 \quad \dots(6.11)$$

δ เรียกว่า Transresistance

2. Voltage - controlled current source (VCCS)



รูป 6.4 Voltage controlled current source (VCCS)

แหล่งจ่ายกระแสทางขวา (I_2) ของรูป 6.4 จะถูกควบคุมด้วยโวลเตจ V_1 ทางด้าน Primary branch ด้วยความสัมพันธ์จากสมการ 6.12

$$I_2 = g_m V_1 \quad \dots(6.12)$$

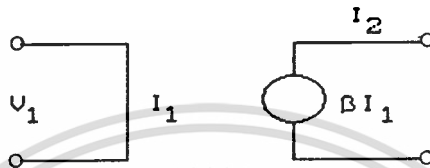
g_m เรียกว่า Transconductance

3. Current - controlled current source (CCCS)

ทางด้าน Secondary branch ของรูป 6.5 จะเป็นแหล่งจ่ายกระแสที่ขึ้นกับกระแสทางด้าน Primary branch โดยมีความสัมพันธ์ดังสมการ 6.13

$$I_2 = \beta I_1 \quad \dots(6.13)$$

β เรียกว่า Current transfer ratio



รูป 6.5 Current controlled current source (CCCS)

4. Voltage - controlled voltage source (VCVS)

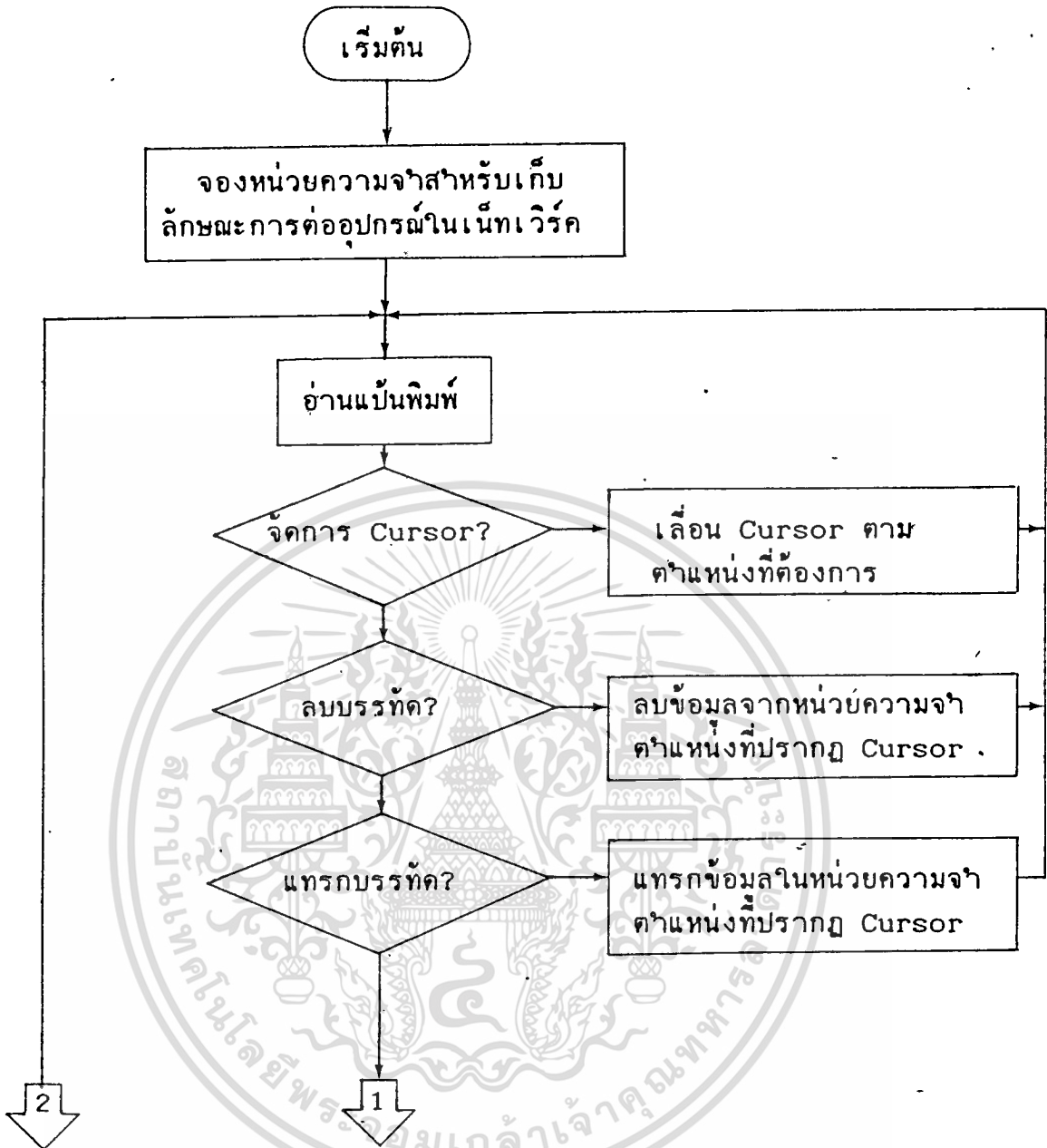
โวลเตจที่เกิดขึ้นทางด้าน Secondary branch (V_2) จะถูกควบคุมด้วยโวลเตจที่ตกคร่อม Primary branch

$$V_2 = \alpha V_1 \quad \dots(6.14)$$

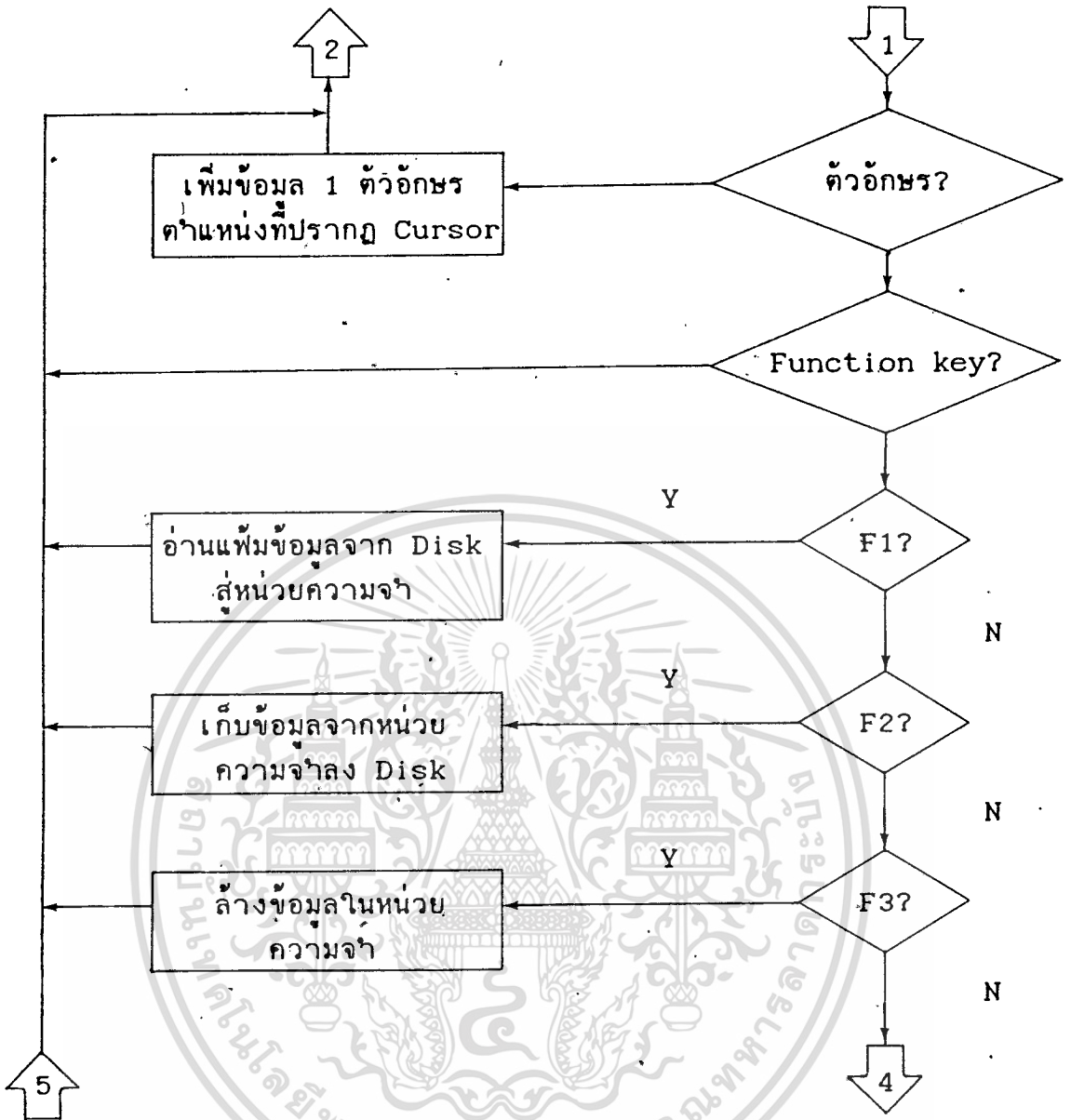


รูป 6.6 Voltage controlled voltage source (VCVS)

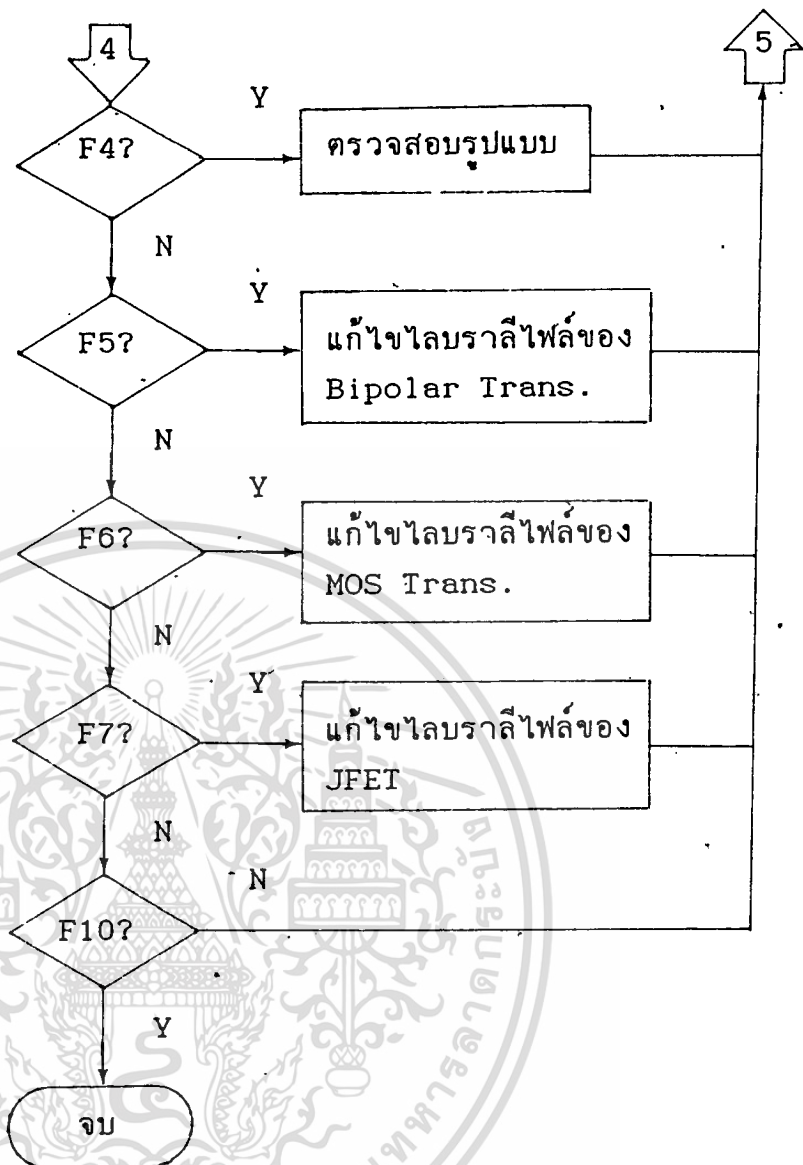
α เรียกว่า Voltage transfer ratio



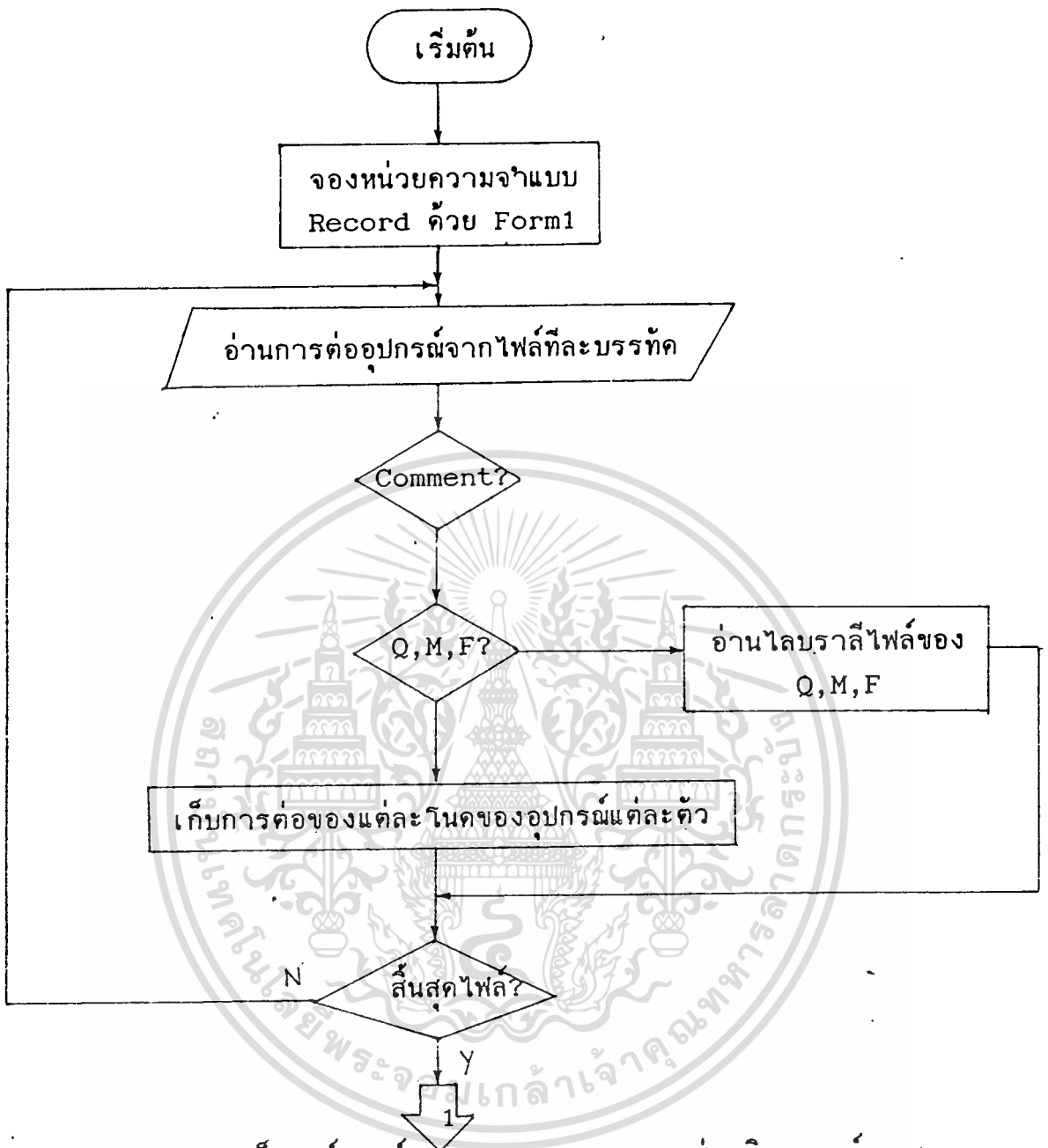
รูป 6.7 เป็นโฟลว์ชาร์ตของโปรแกรม SNAP ส่วน Editor



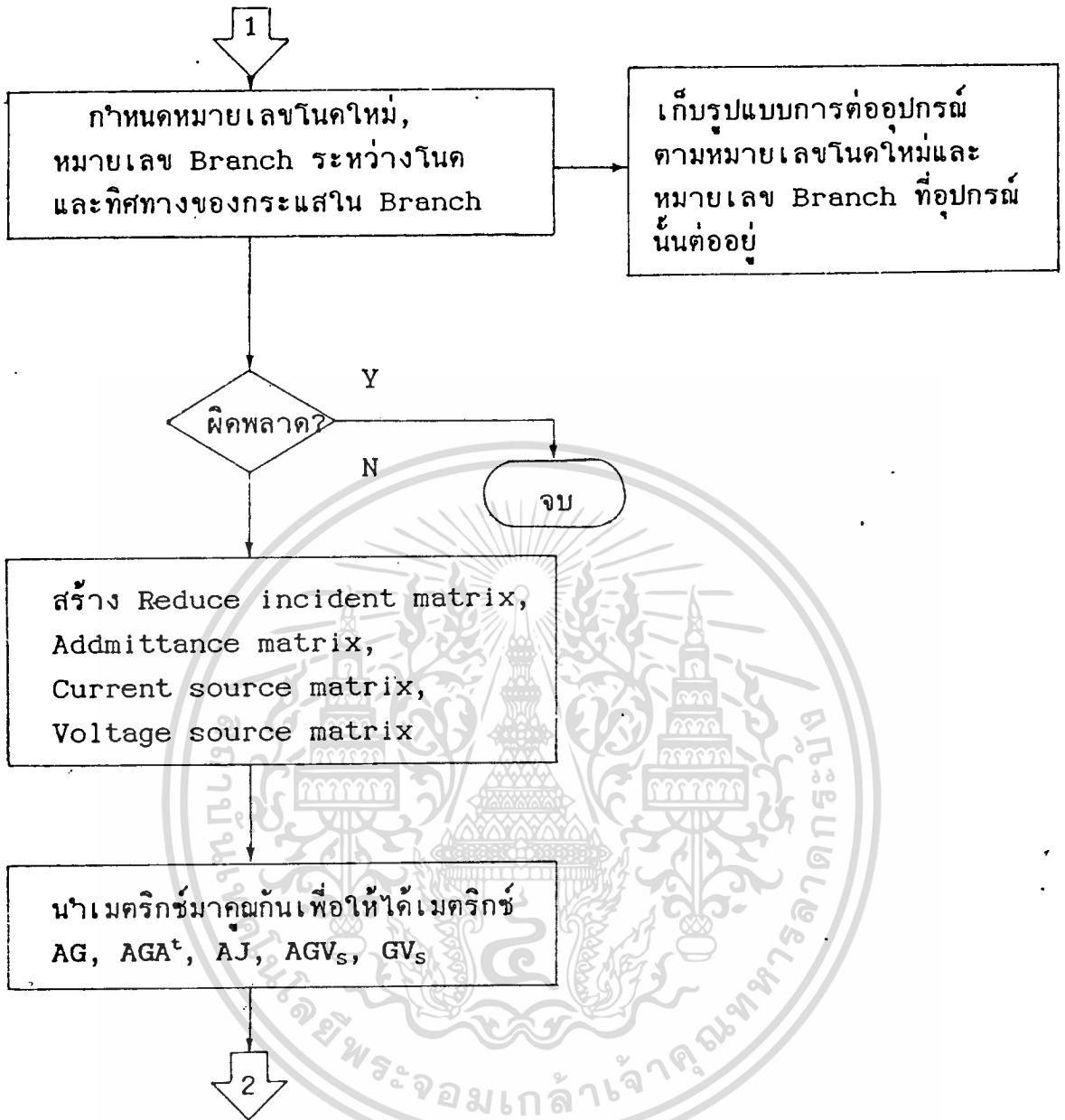
รูป 6.7 เป็นโฟลว์ชาร์ทของโปรแกรม SNAP ส่วน Editor(ต่อ)



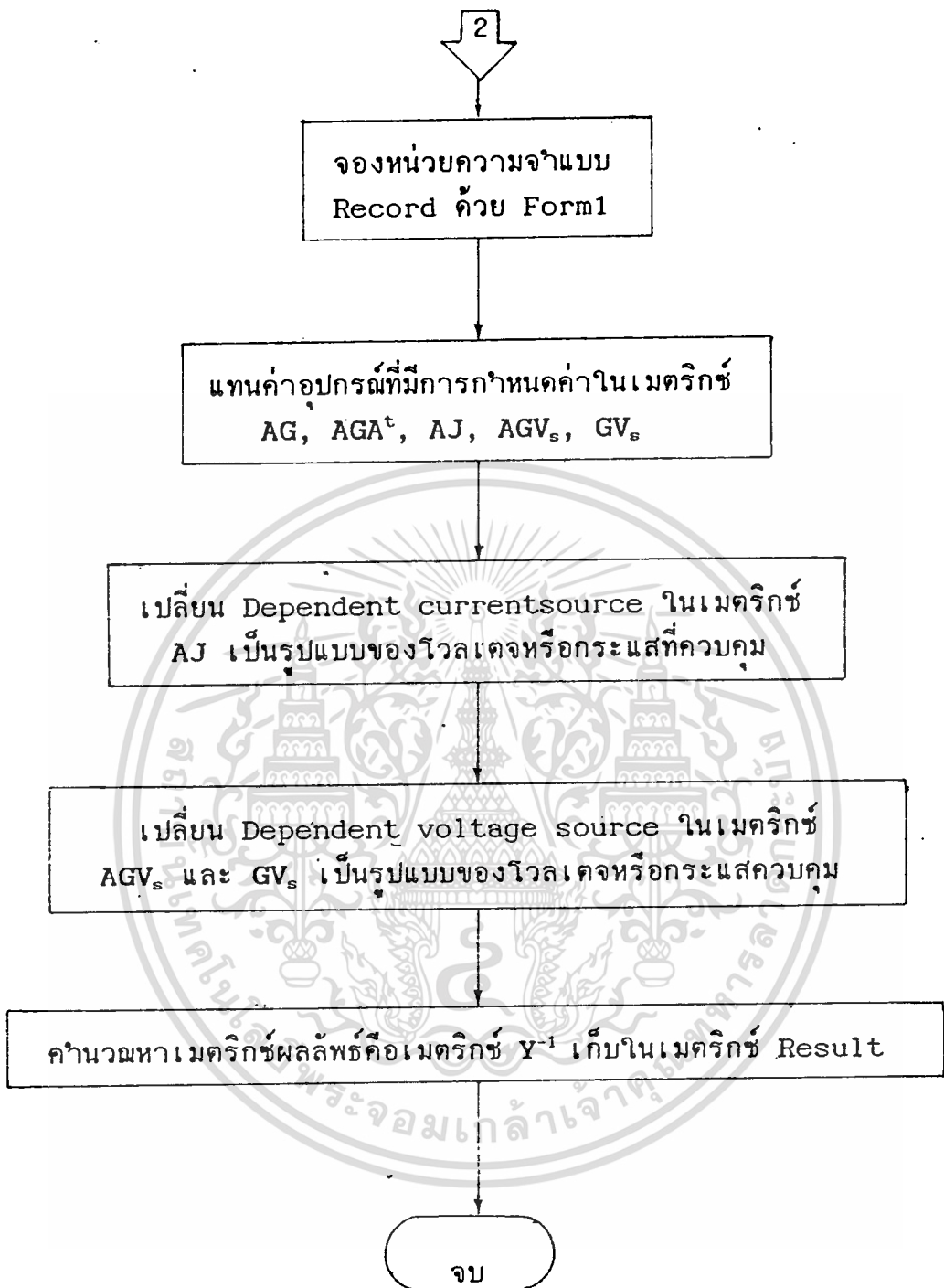
รูป 6.7 เป็นฟลัวร์ชาร์ทของโปรแกรม SNAP ส่วน Editor(ต่อ)



รูป 6.8 เป็นฟลัวร์ชาร์ทของโปรแกรม SNAP ส่วนวิเคราะห์วงจร



รูป 6.8 เป็นโฟลว์ชาร์ทของโปรแกรม SNAP ส่วนวิเคราะห์วงจร(ต่อ)



รูป 6.8 เป็นโฟลว์ชาร์ตของโปรแกรม SNAP ส่วนวิเคราะห์วงจร(ต่อ)

6.4 อัลกอริธึมของโปรแกรมวิเคราะห์เน็ตเวิร์คด้วยสัญลักษณ์

โฟลว์ชาร์ทในรูป 6.7 เป็นโฟลว์ชาร์ท (Flow chart) ของส่วน Editor ซึ่งส่วนนี้จะทำงานในการอ่านข้อมูลจากแป้นพิมพ์ (Keyboard) เข้าไปยังหน่วยความจำ (Memory array) โดยที่การกดแป้นพิมพ์แต่ละครั้ง ก็จะตรวจสอบว่าเป็นการทำงานประเภทใด เช่นแทรกข้อมูล, การลบข้อมูล หรือการป้อนข้อมูลเข้าเป็นต้น รวมทั้งในส่วนนี้สามารถสั่งให้ตรวจสอบรูปแบบ (Format) ของการต่ออุปกรณ์ในแฟ้มข้อมูลได้

จากรูป 6.8 เป็นโฟลว์ชาร์ทของโปรแกรม SNAP โดยเริ่มจากการจองหน่วยความจำของคอมพิวเตอร์เพื่อใช้เก็บรูปสัญลักษณ์ (Symbolic form) ของอุปกรณ์ในเน็ตเวิร์ค จากนั้นจะอ่านชื่อแฟ้มข้อมูลที่เก็บลักษณะของเน็ตเวิร์คที่ต้องการวิเคราะห์ โดยมีรูปแบบของการป้อนลักษณะเน็ตเวิร์คในข้อ 6.5 จากนั้นจะทำการอ่านและตรวจสอบเน็ตเวิร์คที่ละบรรทัดว่าเป็นอุปกรณ์พวก FET (Field effect transistor), ไบโพลารทรานซิสเตอร์ (Bipolar Transistor), ความต้านทาน (Resistor) ฯลฯ กรณีที่เป็นอุปกรณ์พวก FET และ ไบโพลารทรานซิสเตอร์ จะต้องทำการเปิดแฟ้มข้อมูลที่เก็บคุณสมบัติของอุปกรณ์ดังกล่าว จากนั้นจะเก็บทั้งหมดไว้ในแฟ้มข้อมูลชั่วคราว แล้วทำการหาหมายเลขศูนย์รวมและ Branch ทั้งหมดที่ใช้ในวงจร ซึ่งหมายเลขของศูนย์รวมอาจไม่ได้ อยู่ในรูปที่เรียงกันก็ได้ แต่เพื่อความสะดวกโปรแกรมจะทำการจัดเรียงหมายเลขศูนย์รวมเสียใหม่ โดยจะทำให้เป็นเลขเรียงลำดับเริ่มจาก 1, 2, 3 ไปจนถึง N ซึ่งเป็นจำนวนศูนย์รวมของเน็ตเวิร์ค จากนั้นจะทำการตรวจสอบอุปกรณ์ว่ามีอุปกรณ์ซ้ำอยู่หรือไม่ ถ้าหากมีก็จะยกเลิกการทำงาน ถ้าถูกต้องก็จะสร้างแฟ้มข้อมูลของ Reduced incident matrix (A), Current source vector matrix (J_s), Voltage source vector matrix (V_s), Branch admittance matrix (G) แล้วคำนวณหาเมตริกซ์ AG, AGA^t, AJ_s และ AGV_s เมตริกซ์ซึ่งหามาได้ก็จะนำมาใช้คำนวณต่อไปเพื่อหาเมตริกซ์ผลลัพธ์ของ $J_s - GV_s$ ตามสมการ 6.6 จากอุปกรณ์พวก Voltage controlled voltage source (VCVS), Voltage controlled current source (VCCS), Current controlled current source (CCCS) และ Current controlled voltage source (CCVS) นั้น อุปกรณ์เหล่านี้จะเกี่ยวข้องกับ 2 branch ดังนั้นจะต้องมีการเปลี่ยนเมตริกซ์จากผลลัพธ์ของ $(J_s - GV_s)^t$ ในส่วนของแหล่งจ่ายกระแสและแหล่งจ่ายโวลเตจที่ควบคุมด้วย Branch อื่น จากนั้นจะหาผลลัพธ์ซึ่งเป็นโวลเตจที่แต่ละศูนย์รวมโดยการหาอินเวอร์สเมตริกซ์

(Inverse matrix) ของเมทริกซ์ Y ซึ่งมีค่าตามสมการ 6.9 โดยวิธีการทำให้เมทริกซ์ทางด้านซ้ายของสมการ 6.9 เป็น Diagonal matrix จากนั้นจะนำเอา Inverse matrix ที่ได้คูณกับเมทริกซ์ I_s จะได้ผลลัพธ์เป็น Node voltage matrix ซึ่งบอกคุณสมบัติของเน็ตเวิร์คในรูปของโวลเตจที่ศูนย์รวมต่างๆ ดังสมการ 6.10

6.5 อุปกรณ์และรูปแบบที่ใช้ในโปรแกรมวิเคราะห์เน็ตเวิร์คด้วยสัญลักษณ์

โปรแกรม SNAP สามารถวิเคราะห์วงจรซึ่งประกอบด้วยอุปกรณ์ทั้ง Active element และ Passive element โดยการป้อนการต่ออุปกรณ์ให้กับโปรแกรม SNAP ซึ่งเขียนการต่ออุปกรณ์ในเน็ตเวิร์คเป็นแฟ้มข้อมูล และในแฟ้มข้อมูลจะบอกถึงวิธีการต่ออุปกรณ์กับศูนย์รวมต่างๆ มีรูปแบบดังต่อไปนี้

ก. รีซิสเตอร์ (Resistor), อินดักเตอร์ (Inductor) และคาปาซิเตอร์ (Capacitor) มีรูปแบบ

$X_n \quad N_1 \quad N_2 \quad (\text{Value})$

X เป็นชนิดของอุปกรณ์

$R =$ รีซิสเตอร์

$L =$ อินดักเตอร์

$C =$ คาปาซิเตอร์

n เป็นหมายเลขของอุปกรณ์นั้นเป็นเลขจำนวนเต็มมีค่า 0 ถึง 999

N_1, N_2 เป็นหมายเลขศูนย์รวมซึ่งอุปกรณ์นั้นต่ออยู่

Value เป็นค่าของอุปกรณ์นั้น ถ้าไม่มีการป้อนค่านี้ ผลลัพธ์จากการ

วิเคราะห์เน็ตเวิร์คจะอยู่ในรูปของสัญลักษณ์ (Symbolic)

ความต้านทานมีหน่วยเป็นโอห์ม

อินดักเตอร์มีหน่วยเป็นเฮนรี

คาปาซิเตอร์มีหน่วยเป็นฟารัด

เช่นความต้านทาน $R12$ ต่อระหว่างศูนย์รวม 1 กับศูนย์รวม 5 และมีความต้านทาน 1.2 กิโลโอห์มจะเขียนการต่ออุปกรณ์ดังนี้

$R12 \quad 1 \quad 5 \quad 1200$

ข. แหล่งจ่ายโวลเตจ (Voltage source) มีรูปแบบ

$V_n \ N_1 \ N_2$

N_1, N_2 เป็นหมายเลขศูนย์รวมที่แหล่งจ่ายโวลเตจนั้นต่ออยู่ ศูนย์รวม N_1 มีระดับโวลเตจมากกว่าศูนย์รวม N_2 ใน Branch ที่มีแหล่งจ่ายโวลเตจต่ออยู่จะต้องมีอุปกรณ์ตัวใดตัวหนึ่งในข้อ ก. ต่ออนุกรม และต้องป้อนการต่ออุปกรณ์นั้นในบรรทัดต่อจากแหล่งจ่ายโวลเตจ หมายเลขศูนย์รวมของอุปกรณ์ทั้งสองจะเหมือนกัน เช่น ระหว่างศูนย์รวม 0 และศูนย์รวม 1 มีแหล่งจ่ายโวลเตจ V_1 ต่ออยู่กับตัวต้านทาน R_1 ศูนย์รวม 1 มีระดับโวลเตจมากกว่าศูนย์รวม 0 จะเขียนในแฟ้มข้อมูลการต่ออุปกรณ์ได้ว่า

$V1 \ 1 \ 0$

$R1 \ 1 \ 0$

ค. แหล่งจ่ายโวลเตจควบคุมด้วยโวลเตจ (Voltage controlled voltage source) มีรูปแบบที่ขยายมาจากข้อ ข.

$V_n \ N_1 \ N_2 \ V \ N_3 \ N_4$

$V_n \ N_1 \ N_2$ จะมีนิยามเหมือนในข้อ ข.

$N_3 \ N_4$ เป็นหมายเลขศูนย์รวมที่โวลเตจระหว่างศูนย์รวมทั้งสองไปควบคุมแหล่งจ่ายโวลเตจ V_n โดยที่โวลเตจที่ศูนย์รวม N_4 มากกว่าโวลเตจที่ศูนย์รวม N_3 เช่นแหล่งจ่ายโวลเตจ $V1$ อยู่ระหว่างศูนย์รวม 1 และศูนย์รวม 2 ขึ้นกับโวลเตจระหว่างศูนย์รวม 3 กับศูนย์รวม 4 ดังสมการ

$$V_1 - V_2 = \alpha(V_3 - V_4)$$

จะเขียนในแฟ้มข้อมูลการต่ออุปกรณ์ได้ว่า

$V1 \ 1 \ 2 \ V \ 4 \ 3$

ง. แหล่งจ่ายโวลเตจควบคุมด้วยกระแส (Current controlled voltage source) มีรูปแบบที่ขยายมาจากข้อ ข.

$V_n \ N_1 \ N_2 \ J \ N_3 \ N_4$

$V_n \ N_1 \ N_2$ จะมีนิยามเหมือนในข้อ ข.

$N_3 \ N_4$ เป็นหมายเลขศูนย์รวมที่กระแสใน Branch ระหว่างศูนย์รวมนั้นไปควบคุมแหล่งจ่ายโวลเตจ V_n โดยที่ทิศทางของกระแสจะไหลจากศูนย์รวม N_3 ไปยังศูนย์รวม N_4 เช่นแหล่งจ่ายโวลเตจ $V1$ อยู่ระหว่างศูนย์รวม 1 และศูนย์รวม 2 ขึ้นกับกระแสที่ไหลจากศูนย์รวม 3 ไปยังศูนย์รวม 4 เขียนในแฟ้มข้อมูลการต่ออุปกรณ์ได้ว่า

$V1 \ 1 \ 2 \ J \ 3 \ 4$

จ. แหล่งจ่ายกระแส (Current source) มีรูปแบบ

$J_n \ N_1 \ N_2$

มีความหมายถึงแหล่งจ่ายกระแส J_n ไหลจากศูนย์รวม N_1 ไปยังศูนย์รวม N_2 เช่นแหล่งจ่ายกระแส J_1 ไหลจากศูนย์รวม 2 ไปยังศูนย์รวม 3 เขียนในแฟ้มข้อมูลได้ดังนี้

$J_1 \ 2 \ 3$

ฉ. แหล่งจ่ายกระแสควบคุมด้วยโวลเตจ (Voltage controlled current source) มีรูปแบบการต่ออุปกรณ์ที่ขยายมาจากข้อ จ. ดังนี้

$J_n \ N_1 \ N_2 \ V \ N_3 \ N_4$

มีความหมายว่าแหล่งจ่ายกระแสที่ไหลจากศูนย์รวม N_1 ไปยังศูนย์รวม N_2 ขึ้นกับโวลเตจระหว่างศูนย์รวม N_3 และศูนย์รวม N_4 โดยที่โวลเตจที่ศูนย์รวม N_4 มากกว่าโวลเตจที่ศูนย์รวม N_3 เช่นแหล่งจ่ายกระแส J_1 ที่ไหลจากศูนย์รวม 1 ไปยังศูนย์รวม 2 ขึ้นกับโวลเตจระหว่างศูนย์รวม 3 และศูนย์รวม 4 โดยโวลเตจที่ศูนย์รวม 3 มากกว่าโวลเตจที่ศูนย์รวม 4 จะเขียนในแฟ้มข้อมูลได้ว่า

$J_1 \ 1 \ 2 \ V \ 3 \ 4$

ช. แหล่งจ่ายกระแสควบคุมด้วยกระแส (Current controlled current source) มีรูปแบบการต่ออุปกรณ์ที่ขยายมาจากข้อ จ. ดังนี้

$J_n \ N_1 \ N_2 \ J \ N_3 \ N_4$

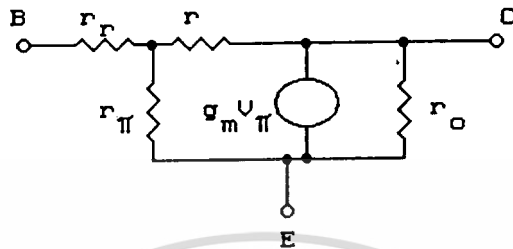
มีความหมายว่าแหล่งจ่ายกระแสที่ไหลจากศูนย์รวม N_1 ไปยังศูนย์รวม N_2 ขึ้นกับกระแสที่ไหลจากศูนย์รวม N_3 ไปยังศูนย์รวม N_4 เช่นแหล่งจ่ายกระแส J_1 ที่ไหลจากศูนย์รวม 1 ไปยังศูนย์รวม 2 ขึ้นกับกระแสที่ไหลจากศูนย์รวม 3 ไปยังศูนย์รวม 4 จะเขียนในแฟ้มข้อมูลได้ว่า

$J_1 \ 1 \ 2 \ J \ 3 \ 4$

ซ. ทรานซิสเตอร์ (Transistor) มีรูปแบบ

Qn Nb Nc Ne name

...(6.15)



รูป 6.9 โมเดลของทรานซิสเตอร์

Q เป็นสัญลักษณ์ของ Bipolar transistor

n เป็นหมายเลขอุปกรณ์นั้นเป็นเลขจำนวนเต็มมีค่า 0 ถึง 999

Nb, Nc, Ne เป็นหมายเลขศูนย์รวมที่ขาเบส (Base), คอลเลคเตอร์ (Collector) และ อิมิตเตอร์ (Emitter) ต่ออยู่ตามลำดับ name เป็นชื่อของโมเดล (Model) ในแฟ้มข้อมูลไลบรารีซึ่งเก็บคุณสมบัติของทรานซิสเตอร์ไว้ โมเดลของทรานซิสเตอร์มีดังรูป 6.9

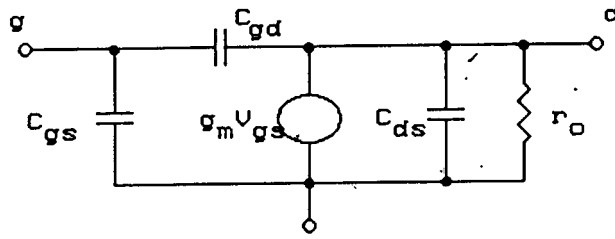
ซ. MOS transistor มีรูปแบบ

Mn Ng Nd Nb Ns name

M เป็นสัญลักษณ์ของ MOS transistor

n เป็นหมายเลขอุปกรณ์นั้นเป็นเลขจำนวนเต็มมีค่า 0 ถึง 999

Ng, Nd, Nb และ Ns เป็นหมายเลขศูนย์รวมที่ขาเกต (Gate), เทรน (Drain), เบส (Base) และซอร์ส (Source) ต่ออยู่ตามลำดับ name เป็นชื่อของโมเดล (Model) ในแฟ้มข้อมูลไลบรารีซึ่งเก็บคุณสมบัติของ MOS ทรานซิสเตอร์ไว้ โมเดลของ MOS ทรานซิสเตอร์มีดังรูป 6.10



รูป 6.10 โมเดลของ MOS ทรานซิสเตอร์

ณ. Field effect transistor มีรูปแบบ

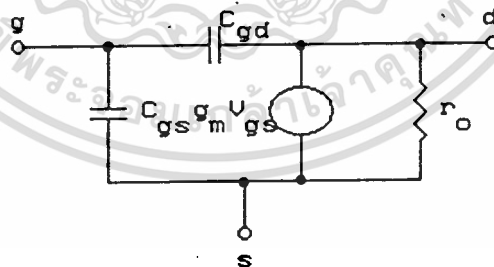
Fn N_g N_d N_s name

F เป็นสัญลักษณ์ของ FET

n เป็นหมายเลขอุปกรณ์เป็นเลขจำนวนเต็มมีค่า 0 ถึง 999

N_g, N_d และ N_s เป็นหมายเลขศูนย์รวมที่ขาเกต (Gate), เทรน (Drain) และซอร์ส (Source) ต่ออยู่ตามลำดับ

name เป็นชื่อของโมเดล (Model) ในแฟ้มข้อมูลไลบรารีซึ่งเก็บคุณสมบัติของ FET ไว้โมเดลของ FET มีดังรูป 6.11



รูป 6.11 โมเดลของ FET

6.6 โปรแกรมและการจัดหน่วยความจำ

โปรแกรม SNAP เขียนขึ้นด้วยภาษาปาสคาล ซึ่งสามารถจัดการกับหน่วยความจำได้ดี ด้วยวิธีเก็บข้อมูลแบบเรคคอร์ด (record) [4] โดยใช้ตัวชี้ตำแหน่งหน่วยความจำ (Memory pointer) และซอร์สโปรแกรม (Source program) ถูกทำการคอมไพล์ (Compile) ด้วย TURBO PASCAL Version 5.0 ซึ่งอุปกรณ์แต่ละตัวจะถูกโปรแกรมเก็บไว้ในหน่วยความจำในคอมพิวเตอร์เป็น Record โดยใช้รูปแบบตัวแปร Record Form1 ดังรูป 6.12 และรูปแบบสมการ 6.16 แต่ในช่วงแรกของโปรแกรมจะยังไม่มีเก็บสัมประสิทธิ์ของเทอมในสมการ ดังนั้นรูปแบบการเก็บข้อมูลเป็น Record ในคอมพิวเตอร์เป็นดังนี้

```
mem_pointer = ^heap_mem;  
heap_mem = record  
    row_no : integer;  
    col_no : integer;  
    equ2 : array[1..component_max] of shortint;  
end;
```

รูป 6.12 รูปแบบ Record form1

$[A_i, B_i, b_{i1}, b_{i2}, \dots, b_{in}] \dots (6.16)$

A_i = หมายเลขศูนย์รวมที่ 1 ที่อุปกรณ์ที่ i ค่อยๆ คอมพิวเตอร์เก็บข้อมูลในตำแหน่งนี้แบบจำนวนเต็ม (Integer) โดยเก็บไว้ในตัวแปรชื่อ row_no ในรูป 6.12

B_i = หมายเลขศูนย์รวมที่ 2 ที่อุปกรณ์ที่ i ค่อยๆ คอมพิวเตอร์เก็บข้อมูลในตำแหน่งนี้แบบจำนวนเต็ม โดยจะเก็บไว้ในตัวแปรชื่อ col_no ในรูป 6.12

b_{ik} = เลขค่ายกกำลังของอุปกรณ์ที่ k ซึ่งอยู่ใน Branch ที่ i ในรูปของค่าความนำ (Conductance) และจะเก็บค่ายกกำลังไว้ในตัวแปรชื่อ equ2[k] ในรูป 6.12

$$k = 1, 2, 3, \dots, K$$

$$b_{ik} = \begin{cases} 0 & ; i \neq k \\ -1 & ; i = k \end{cases}$$

เช่น $[1, 2, 0, 1, 0, 0, 0]$ หมายความว่าอุปกรณ์ที่ 2 ต่อระหว่าง
ศูนย์รวมที่ 1 และศูนย์รวมที่ 2

การเก็บข้อมูลในคอมพิวเตอร์จะทำการเก็บในแบบ Short integer

K = จำนวนอุปกรณ์ทั้งหมดในเน็ตเวิร์ค

การเก็บข้อมูลแบบจำนวนเต็ม ทำให้ตำแหน่งนั้นสามารถเก็บข้อมูลได้
ตั้งแต่ -32768 ถึง 32767 โดยใช้เนื้อที่หน่วยความจำ 2 ไบต์ (Byte) และ
Short integer จะใช้เนื้อที่หน่วยความจำ 1 ไบต์ ทำให้เก็บข้อมูลได้
ระหว่าง -128 ถึง 127 ซึ่งแฟ้มข้อมูลที่เก็บลักษณะของเน็ตเวิร์คเมื่อถูกอ่าน
เข้ามาจะถูกเก็บอยู่ในรูปแบบสมการ 6.16 คือการทำงานดังโพลีชาร์ทรูป 6.8
โดยจะทำการตรวจสอบว่าเป็น FET, MOSFET หรือ ทราวนซิสเตอร์ ซึ่งเป็น
อุปกรณ์ที่ต่อกับศูนย์รวมมากกว่า 2 ศูนย์รวม และอยู่ในรูปแบบที่ต้องการเรียก
แฟ้มข้อมูลไลบรารี ในแฟ้มข้อมูลไลบรารีจะเก็บวงจรเสมือนของอุปกรณ์ข้างต้น
โดยประกอบด้วยอุปกรณ์ซึ่งมี 2 ศูนย์รวม ดังรูป 6.9, 6.10 และ 6.11
อุปกรณ์ในรูปจะถูกกำหนดหมายเลขศูนย์รวมใหม่จากหมายเลขศูนย์รวมเดิมในแฟ้ม
ข้อมูลไลบรารี ให้เป็นไปตามหมายเลขศูนย์รวมในรูปแบบ 6.16 ที่ศูนย์รวมใด
ในแฟ้มข้อมูลไลบรารีไม่ได้ต่อกับศูนย์รวมภายนอกโดยตรง ก็จะถูกกำหนดให้มี
หมายเลขศูนย์รวมใหม่ เมื่ออ่านแฟ้มข้อมูลไลบรารีได้หมด ก็จะทำให้การเก็บใน
Temporary file ไว้ ในรูปแบบซึ่งคอมพิวเตอร์จะนำมาใช้ต่อไปในขั้นตอน
การสร้างเมตริกซ์จากโพลีชาร์ทรูป 6.8 ส่วน Temporary file จะเก็บ
ลักษณะของการต่ออุปกรณ์แต่ละตัวไว้ ในรูปคล้ายกับการเก็บข้อมูลของ Record
ในคอมพิวเตอร์ ดังสมการ 6.16 และแต่ละตำแหน่งจะมีความหมายเปลี่ยนไป
ดังสมการ 6.17

$$[B_1, B_i, b_{i1}, b_{i2}, \dots, b_{iK}] \quad \dots (6.17)$$

B_i = หมายเลข Branch ที่อุปกรณ์ที่ i นั้นต่ออยู่

b_{ik} = ค่ายกกำลังของอุปกรณ์ที่ k ที่อยู่ใน Branch ที่ i

$$k = 1, 2, 3, \dots, K$$

K = จำนวนอุปกรณ์ทั้งหมดในเน็ตเวิร์ค

ค่ายกกำลังของรีซิสเตอร์, อินคักเตอร์จะมีค่าเป็นลบ เนื่องจากค่ายก
กำลังเป็นค่ากำลังของค่าความนำไฟฟ้า อันเป็นส่วนกลับของค่าความต้านทาน

ส่วนค่าบอกกำลังของคาปาซิเตอร์, แหล่งจ่ายกระแส และแหล่งจ่ายโวลเตจ จะมีค่าบอกกำลังเป็นบวก เช่นอุปกรณ์หมายเลข 3 ค่อยู่ใน Branch ที่ 1 เป็นรีซิสเตอร์ จะเขียนในแบบสัญลักษณ์ได้เป็น

$$[1 \ 1 \ 0 \ 0 \ -1 \ 0 \ 0 \ 0]$$

หมายความว่าในเน็ตเวิร์คมีอุปกรณ์ทั้งหมด 6 ตัว และคอมพิวเตอร์จะเก็บหมายเลขแต่ละ Branch ว่าต่อระหว่างศูนย์รวมใดๆไว้ในหน่วยความจำของคอมพิวเตอร์เองจากอุปกรณ์ทั้งหมดที่เก็บไว้ใน Tempolary file โดยจะนำมาสร้างเป็นเมตริกซ์ที่จำเป็นเบื้องต้น ซึ่งสามารถนำไปหาเมตริกซ์อื่นๆต่อไป เมตริกซ์เบื้องต้นเหล่านี้คือ

6.6.1 Reduced incident matrix

จากหมายเลข Branch และหมายเลขศูนย์รวมซึ่งเก็บไว้ในหน่วยความจำของคอมพิวเตอร์จะสร้างเป็น Reduced incident matrix ดังสมการ 6.4 Branch k ซึ่งต่ออยู่ระหว่างศูนย์รวม l และ m ทิศทางของกระแสจะไหลจากศูนย์รวมที่มีหมายเลขน้อยไปยังศูนย์รวมที่มีหมายเลขมากกว่า ดังนั้นจะได้ว่า

$$a_{lm} = -1 ; m > l$$

$$a_{ml} = 1 ; l > m$$

Reduced incident matrix นี้จะถูกเก็บไว้ในหน่วยความจำของคอมพิวเตอร์เพื่อเตรียมที่จะนำไปใช้ในการคำนวณต่อไป เรียกว่าเมตริกซ์ A

6.6.2 Current source vector matrix

Current source vector matrix เป็นเมตริกซ์ขนาด $B \times 1$ การสร้าง Current source vector matrix จะหาได้จากการอ่าน Tempolary file เข้ามา แล้วดูว่าอุปกรณ์ใดเป็นแหล่งจ่ายกระแส ถ้าอุปกรณ์นั้นเป็นแหล่งจ่ายกระแสก็จะทำการเก็บไว้ใน Current source file โดยใช้รูปแบบในสมการ 6.17 ทั้งนี้ Current source อาจเป็น Independent current source หรือ Dependent current source อย่างไรก็ตาม จะมีการจัดการกับ Dependent current source ในการคำนวณต่อไป

6.6.3 Voltage source vector matrix

Voltage source vector matrix เป็นเมตริกซ์ขนาด $B \times 1$ เช่นเดียวกับ Current source matrix เมื่อมีการอ่าน Tempolary file เข้ามาแล้ว จะเก็บทั้ง Dependent voltage source และ Independent voltage source ไว้เพื่อคำนวณต่อไปด้วยรูปแบบ 6.17

6.6.4 Branch admittance matrix

เมื่ออ่าน Tempolary file เข้ามาแล้ว จะทำการสร้าง Branch admittance matrix โดยอุปกรณ์ใน Branch ที่ i จะถูกเก็บไว้ในเมตริกซ์ ที่ตำแหน่ง Row i และ Column j โดยที่นอกจากแนวเส้นทะแยงมุมของ Admittance matrix แล้ว องค์ประกอบนั้นจะมีค่าเป็นศูนย์

6.7 การคำนวณเมตริกซ์

เนื่องจากอุปกรณ์ที่ปรากฏอยู่ในเน็ตเวิร์กนั้น มีบางส่วนไม่จำเป็นที่จะต้องใช้ในการวิเคราะห์ในรูปของสัญลักษณ์ แต่คำนวณอยู่ในรูปค่าที่แท้จริงได้เลย ซึ่งโปรแกรม SNAP สามารถกำหนดค่าที่จริงนี้ได้ตามรูปแบบในข้อ 6.2 เมื่อสร้างเมตริกซ์พื้นฐานในข้อ 6.6 เสร็จแล้ว จะอ่านเมตริกซ์พื้นฐานเหล่านั้นมา เพื่อแทนค่าให้กับอุปกรณ์ที่ต้องการการวิเคราะห์จากค่าจริง ในช่วงนี้เป็นต้นไป การเก็บข้อมูลของคอมพิวเตอร์จะเปลี่ยนรูปแบบของ Record เป็น ดังรูป 6.13 และสมการ 6.18 ซึ่งเพิ่มเติมจากสมการ 6.17 ในส่วนของการเก็บค่าจริง

```
mem_pointer2 = ^heap_mem2;
heap_mem2 = record
    row_no : integer;
    col_no : integer;
    equ3 : real;
    equ2 : array[1..component_max2] of shortint;
end;
```

รูป 6.13 รูปแบบ Record form2

$$[B_i, B_i, F_i, b_{i1}, b_{i2}, \dots, b_{io}] \quad \dots (6.18)$$

B_i ตำแหน่งแรกจะเก็บในตัวแปร row_no

B_i ตำแหน่งสองจะเก็บในตัวแปร col_no

F_i เก็บในตัวแปร equ3

b_{i1} ถึง b_{io} เก็บในตัวแปร equ2[1] ถึง equ2[o]

F_i เป็นค่าจริงของอุปกรณ์ที่ i ยกกำลังด้วยค่าของ b_{ik} ถ้าอุปกรณ์นั้นจะคงรูปการคำนวณในแบบสัญลักษณ์ ค่าของ F_i จะมีค่าเป็น 1.0 และถ้าอุปกรณ์ที่ i มีการแทนค่าด้วยค่าจริงจะได้ว่า $b_{i1} = 0$

$$F_i = D_i$$

D_i เป็นค่าจริงของอุปกรณ์

จากเมตริกซ์พื้นฐาน ค่า b_{ik} จะมีค่าเป็น 0 หรือ 1 เท่านั้น เนื่องจากการเขียนอุปกรณ์ในรูปของสัญลักษณ์ที่ยังไม่มีการคำนวณอื่นใด ดังนั้นค่าของ F_i ที่เกิดจากการแทนค่าจากเมตริกซ์พื้นฐานจึงมีค่า

$$F = \begin{cases} D_i & ; b_{ik} = 1 \\ 1.0 & ; b_{ik} = 0 \\ 1/D_i & ; b_{ik} = -1 \end{cases}$$

เมื่อมีการแทนค่าอุปกรณ์ด้วยตัวเลขแล้วทำให้ b_i มีค่าเป็นศูนย์ในทุกๆ Record ดังนั้นจึงไม่เก็บค่า b_{ii} อีกต่อไป แต่ได้กำหนดให้ P เป็นจำนวนอุปกรณ์ที่มีการแทนค่า ดังนั้นจะมีอุปกรณ์ที่เหลือติดอยู่ในรูปของสัญลักษณ์อีกเท่ากับ $N - P$ เทอม แต่จะเพิ่มอีกเทอมสำหรับเก็บ S-operator เมื่อมีการแทนค่าอุปกรณ์ซึ่งเป็นอุปกรณ์ที่มีค่าขึ้นกับความถี่ (Frequency dependent) คืออินดักเตอร์และคาปาซิเตอร์ เมื่อมีการแทนค่าให้กับอินดักเตอร์จะให้ S มีค่าเป็น -1 และเมื่อมีการแทนค่าให้กับคาปาซิเตอร์จะให้ $S = 1$ ดังนั้น b จึงเหลือเทอมทั้งหมด

$$N - P + 1 = O \text{ เทอม}$$

ทำให้ประหยัดหน่วยความจำและเวลาที่จะใช้ในการคำนวณ เมื่อมีการแทนค่าเสร็จแล้วจะเก็บไว้ในแฟ้มข้อมูล เรียกว่าเมตริกซ์ G (Branch conductance matrix) เพื่อรอการคำนวณต่อไป

จากเมตริกซ์ A ขนาด $N \times B$ และ เมตริกซ์ G (Branch conductance matrix) - ขนาด $B \times B$ เมื่อนำมาทำการคูณให้ผลลัพธ์เป็นเมตริกซ์ AG ขนาด $N \times B$

$$(AG)_{ij} = \sum_{k=1}^B (A_{ik}G_{kj})$$

เมตริกซ์ AG สามารถนำไปคำนวณต่อเพื่อหาเมตริกซ์ AGA^c ต่อไป
ได้โดยการคูณเมตริกซ์ AG ด้วยเมตริกซ์ A^c

จาก Voltage source vector matrix จะคูณด้วยเมตริกซ์ AG เช่น เดียวกับการหาเมตริกซ์ AGA^c โดยการนำเอาเมตริกซ์ AG ซึ่งมีขนาด $N \times B$ คูณด้วยเมตริกซ์ V_s ขนาด $B \times 1$ ผลลัพธ์ที่ได้จะเป็นเมตริกซ์ AGV_s ที่มีขนาด $N \times 1$ ซึ่งในการคูณเมตริกซ์ AG กับเมตริกซ์ V_s นี้ ทั้งสองเมตริกซ์ เป็นเมตริกซ์ที่มีการเก็บข้อมูลแบบ 6.17 และ 6.18 ตามลำดับ โดยแต่ละ เทอมจะประกอบด้วยหมายเลขแถว, หมายเลขคอลัมน์, สัมประสิทธิ์ และเลขยกกำลังของสัญลักษณ์แต่ละตัวในเน็ตเวิร์ค สำหรับการคูณกันของแต่ละเทอม สัมประสิทธิ์ในเมตริกซ์ จะนำเอาค่าสัมประสิทธิ์ของทั้งสองเทอมมาคูณกัน และ นำเอาเลขยกกำลังของสัญลักษณ์แต่ละตัวที่ตำแหน่งตรงกันมาบวกกัน

$$(AGV_s)_{i1} = \sum_{K=1}^B \sum_{J=1}^B [R_k, C_k, A_{ik} \times F_{kj} \times V_{s(k1)}, b_{AGVS1}, b_{AGVS2}, \dots, b_{AGVS0}] \dots (6.19)$$

$$b_{AGVSn} = b_{AGn} + b_{Vsm}$$

b_{AGVSn} = เลขยกกำลังของสัญลักษณ์ตัวที่ m ของเทอม i ในเมตริกซ์ AGV_s

b_{AGn} = เลขยกกำลังของสัญลักษณ์ตัวที่ m ของเทอม i ในเมตริกซ์ AG

b_{Vsm} = เลขยกกำลังของสัญลักษณ์ตัวที่ m ของเทอม i ในเมตริกซ์ V_s

จากสมการ 6.19 แต่ละองค์ประกอบของเมตริกซ์อาจประกอบด้วย หลายเทอม และบางเทอมอาจมีเลขยกกำลังของสัญลักษณ์แต่ละตัวเท่ากัน หรือเหมือนกัน ทำให้สามารถลดจำนวนเทอมย่อยลงได้เหลือเพียงเทอมเดียว โดยการนำเอาค่า F_{kj} ของแต่ละเทอมที่มีค่ายกกำลังของสัญลักษณ์แต่ละตัวมา รวมกัน

จาก Current source vector matrix จะนำมาคูณกับ Reduced incident matrix A ได้เป็นเมตริกซ์ AJ เมตริกซ์ A มีขนาด $N \times B$ และ เมตริกซ์ J มีขนาด $B \times 1$ ซึ่งจะได้ผลลัพธ์ของการคูณเป็นเมตริกซ์ AJ ขนาด $N \times 1$ การคูณเมตริกซ์ A กับเมตริกซ์ J_s นี้ จะมีวิธีการเหมือนกับการคูณ เมตริกซ์ AG

แหล่งจ่ายโวลเตจจะเปลี่ยนเป็นแหล่งจ่ายกระแสโดยการคูณด้วย Conductance ใน Branch ที่มีแหล่งจ่ายโวลเตจนั้นต่ออยู่ คือการคูณเมตริกซ์ G ด้วยเมตริกซ์ V_s โดยวิธีการคูณจะคล้ายกับการคูณเมตริกซ์ AG ด้วยเมตริกซ์ V_s

จากเมตริกซ์ J_s และ GV_s จะนำมาหาเมตริกซ์ $J_s - GV_s$ ซึ่งเป็นเมตริกซ์ผลลัพธ์ของแหล่งจ่ายกระแสและแหล่งจ่ายโวลเตจ โดยทั้งสองเมตริกซ์นี้จะมีทั้ง Independent และ Dependent source จะต้องทำการเปลี่ยน Dependent source ให้อยู่ในรูปของเมตริกซ์ที่เป็น Independent source

a. Voltage controlled voltage source

Voltage controlled voltage source เป็นแหล่งจ่ายโวลเตจที่มีขนาดขึ้นกับความต่างศักย์ของโวลเตจระหว่าง 2 ศูนย์รวม โดยแหล่งจ่ายโวลเตจจะถูกเปลี่ยนเป็นแหล่งจ่ายกระแสด้วยการคูณเมตริกซ์ A และ G ได้ผลลัพธ์เป็นเมตริกซ์ AGV_s จึงเป็น Current dependent source ซึ่งขึ้นกับความต่างศักย์ของโวลเตจระหว่าง 2 ศูนย์รวม จากความสัมพันธ์ของ Dependent voltage source ในสมการ 6.14 จะได้ว่า

$$V_{ds1} = \alpha(V_{n1} - V_{n2}) \quad \dots(6.20)$$

คือแหล่งจ่ายโวลเตจ V_{ds1} ขึ้นกับความต่างศักย์ระหว่างศูนย์รวม N_1 และศูนย์รวม N_2 ด้วยอัตราส่วนเท่ากับ α ถ้า G ใน Branch นั้นเท่ากับ G_x จะสามารถเปลี่ยนแหล่งจ่ายโวลเตจเป็นแหล่งจ่ายกระแสดังรูป 6.1 และสมการ 6.22

$$J_{ds1} = V_{ds1}G_x = G_x(V_{n1} - V_{n2}) \quad \dots(6.21)$$

จากสมการ 6.9 จึงสามารถเขียนให้มีความสัมพันธ์ของ Dependent source ในสมการใหม่ได้เป็น

$$I_{s1} = Y_n e - J_{ds} \quad \dots(6.22)$$

เมื่อ J_{ds} เป็น Current dependent source และขึ้นกับโวลเตจที่เกิดขึ้นระหว่าง 2 ศูนย์รวม

ในโปรแกรมจะทำการเปลี่ยนแหล่งจ่ายโวลเตจเป็นแหล่งจ่ายกระแสโดยการคูณด้วยค่าการนำใน Branch นั้น แล้วบวกเข้ากับเมตริกซ์ $Y_n e$ การบวกเมตริกซ์จะทำการบวกเมตริกซ์ซึ่งมีรูปแบบ

$$[R_k, C_1, F, b_1, b_2, \dots, b_o] \quad \dots(6.23)$$

และมีหมายเลขแถวและหมายเลขคอลัมน์ตรงกันแล้วในทั้ง 2 เทอม ถ้า b_1 ถึง b_o มีค่าตรงกันจะนำเอาเทอม F ของทั้งสองเทอมมารวมกัน โดย

เก็บผลลัพธ์ไว้และเทอม b_1 ถึง b_o จะมีค่าคงเดิมแต่ถ้า b_1 ถึง b_o ไม่เหมือนกัน ผลลัพธ์จะมี 2 เทอมเกิดขึ้นที่หมายเลขแถวและหมายเลขคอลัมน์ดังกล่าว

b. Current controlled voltage source

Current controlled voltage source เป็นแหล่งจ่ายโวลเตจซึ่งมีค่าขึ้นกับกระแสใน Branch หนึ่ง ดังสมการ 6.11 แหล่งจ่ายโวลเตจดังกล่าวจะเปลี่ยนเป็นแหล่งจ่ายกระแสดังรูป 6.1 ซึ่งให้แหล่งจ่ายโวลเตจ V ขึ้นกับกระแสใน Branch 1 (ระหว่างศูนย์รวม N_1 กับศูนย์รวม N_2) ซึ่งมีค่า i_1 ดังนั้น

$$V_{ds1} = \delta i_1 \quad \dots(6.24)$$

และแหล่งจ่ายโวลเตจเปลี่ยนเป็นแหล่งจ่ายกระแสได้

$$G_x V_{ds1} = G_x \delta i_1 \quad \dots(6.25)$$

โดยที่ G_x เป็น Branch conductance ของ Branch ที่มี V_{ds1} ต่ออยู่ จากสมการ 6.25 ถ้ากระแสที่ไหลระหว่างศูนย์รวม N_1 กับศูนย์รวม N_2 ซึ่งระหว่างศูนย์รวมนั้นมี Branch conductance G_1 ต่ออยู่ จะได้ว่า

$$G_x V_{ds1} = G_1 G_x (V_{n1} - V_{n2}) \quad \dots(6.26)$$

ดังนั้นจากสมการ 6.9 สามารถเขียนให้มีความสัมพันธ์ของ

Dependent voltage source ในสมการได้

$$I_{s2} = Y_n e - J_{dis} \quad \dots(6.27)$$

เมื่อ J_{dis} เป็น Dependent current source ซึ่งทราบสฟอรมจาก Current controlled voltage source โดยสมการ 6.28 การลบกันของเมตริกซ์จะทำให้เหมือนกับการทำงานกับ Voltage controlled voltage source ที่กล่าวมาแล้ว

c. Voltage controlled current source (VCCS)

Voltage controlled current source เป็นแหล่งจ่ายกระแสซึ่งมีค่าขึ้นกับความต่างศักย์ระหว่าง 2 ศูนย์รวม ความสัมพันธ์ระหว่างแหล่งจ่ายกระแสและโวลเตจที่ควบคุมจะเป็นดังสมการ 6.12 โดยมีค่าอัตราส่วนระหว่างค่าของแหล่งจ่ายกระแส และโวลเตจที่ควบคุมมีค่าเท่ากับ g_m ดังนั้นจะเห็นได้จากสมการว่าแหล่งจ่ายกระแสจะติดอยู่ในเทอมของโวลเตจที่ศูนย์รวม

ให้ J_{dvs} เป็น Dependent current source matrix สามารถเขียนสมการ 6.15 ให้มีความสัมพันธ์ของ Dependent current source ได้ว่า

$$I_{s3} = Y_n e - J_{dvs} \quad \dots(6.29)$$

d. Current controlled current source

Current control current source เป็นแหล่งจ่ายกระแส ซึ่งมีค่าขึ้นกับกระแสใน Branch ใด Branch หนึ่ง ดังสมการ 6.13 โดยมี β เป็นอัตราส่วนของแหล่งจ่ายกระแสใน Branch ใด Branch หนึ่ง ซึ่งกระแสใน Branch ที่ไปควบคุมแหล่งจ่ายกระแสหาได้จาก

$$i_c = -(V_{n1} - V_{n2}) G_x$$

เมื่อ V_{n1} และ V_{n2} เป็น Node voltage ซึ่งตกคร่อม Branch ของกระแสควบคุม

G_x เป็น Branch conductance ของ Branch กระแสควบคุม

i_c เป็นกระแสใน Branch ของกระแสควบคุมซึ่งจะเห็นว่าขึ้นกับโวลเตจที่ศูนย์รวม

ให้ J_{dccc} เป็นเมตริกซ์ของ Dependent current control current source แล้วสมการ 6.9 เขียนได้เป็น

$$I_{s4} = Y_n e - J_{dccc} \dots (6.30)$$

จาก 6.22, 6.27, 6.29 และ 6.30 สามารถเขียนเป็นสมการรวมได้

$$Y_n e - J_{ds} - J_{dls} - J_{dvs} - J_{dccc} = I_{sN} \dots (6.31)$$

I_{sN} เป็น Current source vector matrix ที่มีเฉพาะ Independent current source

$$Y_{nd} e = I_{sN} \dots (6.32)$$

แต่ละเทอมของ $Y_{nd} e$ จะหาได้จากการนำเอาเมตริกซ์ทางซ้ายของสมการ 6.31 มาลบกัน ดังนั้นจาก $Y_{nd} e = I_s$ สามารถหาเมตริกซ์ผลลัพธ์ได้

$$Y_{nd}^{-1} Y_{nd} e = Y_{nd}^{-1} I_s$$

$$e = Y_{nd}^{-1} I_s$$

เมตริกซ์ Y_{nd}^{-1} จะหาได้จาก Inverse matrix ของ Y_{nd} โดยการสร้างเมตริกซ์ Identity matrix ให้มีแต่ละสัมประสิทธิ์ของเมตริกซ์มีค่าจริงเป็น 1.0 ถ้าหมายเลขแถวมีค่าเดียวกับหมายเลขคอลัมน์

$$I_{ij} = \begin{cases} [i, j, 1.0, b_1, b_2, \dots, b_o] & ; i = j, b_1 = b_2 = \dots = b_o = 0 \\ [i, j, 0.0, b_1, b_2, \dots, b_o] & ; i \neq j, b_1 = b_2 = \dots = b_o = 0 \end{cases}$$

จากนั้นโปรแกรมจะทำการเปลี่ยนเมตริกซ์ Y_{nd} ให้เป็น Identity matrix โดยการทำให้ Lower triangle เป็นศูนย์ และทำให้ Upper triangle ของเมตริกซ์เป็นศูนย์เช่นกัน โดยการกระทำใดๆต่อแต่ละ Row ของ

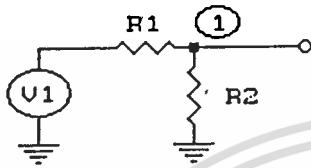
เมตริกซ์ Y_{nd} ก็จะทำเช่นเดียวกันต่อ Row นั้นๆ ของเมตริกซ์ I เมื่อเมตริกซ์ Y_{nd} ให้ผลลัพธ์เป็น Identity matrix คือมีสัมประสิทธิ์ในแนวทแยงมุมเป็น 1 และ Upper, Lower triangle เป็นศูนย์ เมตริกซ์ I ก็คือ Inverse matrix ของเมตริกซ์ Y_{nd} ในตอนแรก และเมื่อคูณเมตริกซ์ Y_{nd} ด้วยเมตริกซ์ J_s จะได้เป็นเมตริกซ์ e คือ node voltage เมตริกซ์ที่แสดงโวลเตจของทุกศูนย์รวมในวงจรตามต้องการ



6.8 ตัวอย่าง

ตัวอย่างที่ 1

ในรูป 6.14 เป็นวงจรแบ่งโวลเตจที่ประกอบด้วยแหล่งจ่ายโวลเตจ และตัวต้านทาน 2 ตัว โดยมีหมายเลขศูนย์รวมกำกับไว้ในรูป



รูป 6.14 วงจรแบ่งโวลเตจ

ซึ่งจะเขียนเป็นแฟ้มข้อมูลการต่ออุปกรณ์ได้ดังนี้

*VOLTAGE DIVIDER

V1 1 0

R1 1 0

R2 1 0

ในการเขียนแฟ้มข้อมูลการต่ออุปกรณ์ของแหล่งจ่ายโวลเตจ จะเขียนอุปกรณ์ที่อนุกรมอยู่กับแหล่งจ่ายโวลเตจไว้ต่อจากบรรทัดของแหล่งจ่ายโวลเตจ ซึ่งจากแฟ้มข้อมูลแสดงว่าแหล่งจ่ายโวลเตจ V1 ต่ออนุกรมกับตัวต้านทาน R1 ที่ต่ออยู่ใน Branch ระหว่างศูนย์รวม 1 กับกราวด์และมี R2 ต่ออยู่ระหว่างศูนย์รวม 1 กับกราวด์ เมื่อคำนวณโดยโปรแกรม SNAP จะได้ผลลัพธ์ในแฟ้มข้อมูล RESULT ดังนี้

```
Voltage at node 1 =  
[( 1.000000E+0000*(V1)*(R2))]  
/  
[( 1.000000E+0000*(R2))  
+( 1.000000E+0000*(R1))  
]
```

ซึ่งแสดงค่าโวลเตจที่ศูนย์รวม 1 และมีความหมายเท่ากับ

$$\frac{V1R2}{R1 + R2}$$

จากรูปดังกล่าวสามารถทำการคำนวณโดยกำหนดค่าให้อุปกรณ์ดังนี้
R1 = 1 กิโลโอห์ม
R2 = 3 กิโลโอห์ม
จะเขียนเป็นแฟ้มข้อมูลการต่ออุปกรณ์ได้ดังนี้

```
*VOLTAGE DIVIDER  
V1 1 0  
R1 1 0 1000  
R2 1 0 3000
```

และได้ผลลัพธ์จากการคำนวณโดยโปรแกรม SNAP เก็บไว้ในแฟ้ม
ข้อมูลชื่อ RESULT ดังนี้

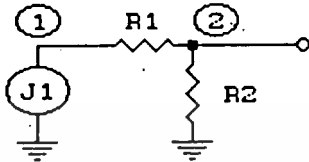
```
Voltage at node 1 =  
[( 7.500000E-0001*(V1))]  
/  
[( 1.000000E+0000)  
]
```

ค่าตอบข้างบนเป็นค่าโวลเตจที่ศูนย์รวม 1 หรือมีความหมายเท่ากับ

0.75 V1

ตัวอย่างที่ 2

เป็นการแสดงตัวอย่างของวงจรที่มีแหล่งจ่ายกระแสดังรูป 6.15 ที่มีแหล่งจ่ายกระแสและตัวต้านทาน 2 ตัว



รูป 6.15 วงจรแหล่งจ่ายกระแสและตัวต้านทาน

ซึ่งเขียนเป็นแฟ้มข้อมูลการต่ออุปกรณ์ได้ดังนี้

*CURRENT SOURCE

J1 0 1

R1 1 2

R2 2 0

จากแฟ้มข้อมูลการต่ออุปกรณ์ มีความหมายว่าแหล่งจ่ายกระแสต่อระหว่างศูนย์รวม 1 กับศูนย์รวม 0 โดยมีทิศทางกระแสจากศูนย์รวม 0 ไปยังศูนย์รวม 1 และมีตัวต้านทาน R1 ต่อระหว่างศูนย์รวม 1 กับศูนย์รวม 2 และความต้านทาน R2 ต่อระหว่างศูนย์รวม 2 กับศูนย์รวม 0 ซึ่งเมื่อคำนวณโดยโปรแกรม SNAP จะได้แฟ้มข้อมูลผลลัพธ์ที่แสดงค่าโวลเตจที่แต่ละศูนย์รวมดังนี้

```
Voltage at node 1 =  
[( 1.000000E+0000*(J1)*(R1)) +  
 ( 1.000000E+0000*(J1)*(R2))]  
/  
[( 1.000000E+0000)  
]  
Voltage at node 2 =  
[( 1.000000E+0000*(J1)*(R2))]  
/  
[( 1.000000E+0000)  
]
```

โวลเตจที่ศูนย์รวม 1 จะมีค่าเท่ากับ $J1(R1 + R2)$

โวลเตจที่ศูนย์รวม 2 จะมีค่าเท่ากับ $J1R2$

ถ้ากำหนดให้แหล่งจ่ายกระแสเป็นแหล่งจ่ายกระแสซึ่งมีสมการ

$J1 = \sin(\omega t)$

จะหาค่าของโวลเตจแต่ละศูนย์รวมได้ว่า

โวลเตจที่ศูนย์รวม 1 เท่ากับ $\sin(\omega t)(R1 + R2)$

โวลเตจที่ศูนย์รวม 2 เท่ากับ $R2 \sin(\omega t)$

ตัวอย่างที่ 3

เป็นตัวอย่างของวงจร Tank tuned ดังรูป 6.16 ประกอบด้วย แหล่งจ่ายโวลเตจ, ตัวต้านทาน, ตัวเหนี่ยวนำและตัวเก็บประจุ จากวงจร เขียนเป็นแฟ้มข้อมูลการต่ออุปกรณ์ได้ว่า

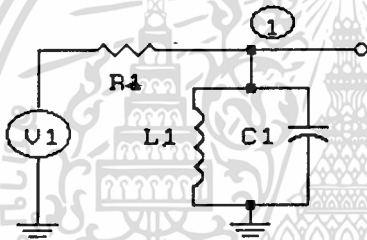
*Tank tuned circuit

V1 1 0

R1 1 0

L1 1 0

C1 1 0



รูป 6.16 วงจร Tank tuned

ใน Branch ของแหล่งจ่ายโวลเตจ จะมีตัวต้านทานต่ออนุกรมอยู่ระหว่างศูนย์รวม 1 กับศูนย์รวม 0 ตัวเหนี่ยวนำและตัวเก็บประจุจะต่ออยู่ระหว่างศูนย์รวม 0 กับศูนย์รวม 1 เช่นกัน เมื่อผ่านการคำนวณโดยโปรแกรม SNAP จะมีค่าของโวลเตจที่ศูนย์รวม 1 ดังนี้

Voltage at node 1 =

$$\frac{[(1.000000E+0000 \cdot (V1) \cdot (L1) \cdot (S))]}{[(1.000000E+0000 \cdot (L1) \cdot (S)) + (1.000000E+0000 \cdot (R1)) + (1.000000E+0000 \cdot (R1) \cdot (L1) \cdot (C1) \cdot (S^2))]}$$

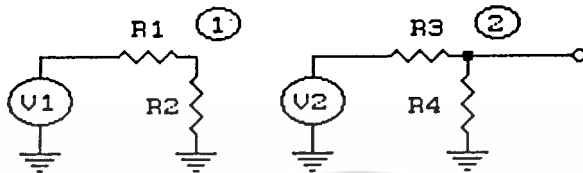
หรือโวลเตจที่ศูนย์รวม 1 ก็คือ

$$\frac{V1SL1}{s^2L1C1R1 + SL1 + R1}$$



ตัวอย่างที่ 4

เป็นตัวอย่างของวงจรที่ประกอบด้วยอุปกรณ์แหล่งจ่ายโวลเตจควบคุมด้วยโวลเตจดังรูป 6.17 แหล่งจ่ายโวลเตจ V2 จะถูกควบคุมด้วยโวลเตจระหว่างศูนย์รวม 1 กับศูนย์รวม 0 หรือโวลเตจที่ตกคร่อม R2



รูป 6.17 วงจรที่ประกอบด้วยแหล่งจ่ายโวลเตจควบคุมด้วยโวลเตจ

ความสัมพันธ์ระหว่างโวลเตจระหว่างศูนย์รวม 1 และศูนย์รวม 0 กับแหล่งจ่ายโวลเตจ V2 สามารถแสดงได้ว่า

$$V_2 = \alpha(V_1 - V_0) = \alpha V_1$$

เพราะโวลเตจที่ศูนย์รวม 0 มีค่าเป็นศูนย์ ใน Branch ระหว่างศูนย์รวม 1 กับศูนย์รวม 0 จึงมีแหล่งจ่ายโวลเตจ V1 ต่ออนุกรมกับตัวต้านทาน R1 และความต้านทาน R2 ในอีก Branch ต่ออยู่ระหว่างศูนย์รวม 1 กับศูนย์รวม 0 และแหล่งจ่ายโวลเตจ V2 ต่ออนุกรมกับตัวต้านทาน R3 ที่ต่ออยู่ระหว่างศูนย์รวม 2 กับศูนย์รวม 0 และความต้านทาน R4 ในอีก Branch ต่ออยู่ระหว่างศูนย์รวม 2 กับศูนย์รวม 0 สามารถจะเขียนเป็นแฟ้มข้อมูลการต่ออุปกรณ์ได้ดังนี้

* Voltage controlled voltage source

V1 1 0

R1 0 1

R2 1 0

V2 2 0 V 0 1

R3 0 2

R4 2 0

ในบรรทัดที่ 5 แสดงการต่อของแหล่งจ่ายโวลเตจควบคุมด้วยโวลเตจในแฟ้มข้อมูล เมื่อคำนวณโดยโปรแกรม SNAP จะได้ผลลัพธ์ซึ่งเป็นค่าโวลเตจที่แต่ละศูนย์รวมดังนี้

Voltage at node 1 =

$$[(1.000000E+0000 \cdot (V1) \cdot (R2))]$$

/

$$[(1.000000E+0000 \cdot (R2))$$

$$+ (1.000000E+0000 \cdot (R1))$$

]

Voltage at node 2 =

$$[(1.000000E+0000 \cdot (V1) \cdot (R2) \cdot (\text{Alpha2}) \cdot (R4))]$$

/

$$[(1.000000E+0000 \cdot (R2) \cdot (R4))$$

$$+ (1.000000E+0000 \cdot (R1) \cdot (R4))$$

$$+ (1.000000E+0000 \cdot (R2) \cdot (R3))$$

$$+ (1.000000E+0000 \cdot (R1) \cdot (R3))$$

]

จะเห็นว่าโวลเตจที่ศูนย์รวม 1 ได้ผลเหมือนวงจรแบ่งโวลเตจในตัวอย่างที่ 1 ส่วนโวลเตจที่ศูนย์รวม 2 จะมีดังนี้

$$\frac{V1(\text{Alpha2})R2R4}{(R1 + R2)(R3 + R4)}$$

เนื่องจากการเก็บผลลัพธ์ไว้ในแฟ้มข้อมูลไม่สามารถเก็บสัญลักษณ์ α ได้ จึงแสดงค่า α เป็น Alpha2 ซึ่งในที่นี้หมายถึง α ที่แหล่งจ่ายโวลเตจ V2

ตัวอย่างที่ 5

รูป 6.18 เป็นวงจรที่ประกอบด้วยแหล่งจ่ายกระแส ซึ่งควบคุมด้วย
โวลเตจที่ตกคร่อม Branch หนึ่ง จากรูป 6.18 เขียนเป็นแฟ้มข้อมูลการต่อ
อุปกรณ์ได้ดังนี้

*Voltage controlled current source

V1 0 1

R1 1 0

R2 1 0

J2 0 2 V 1 0

R3 0 2

R4 0 2



รูป 6.18 วงจรที่ประกอบด้วยแหล่งจ่ายกระแสควบคุมด้วยโวลเตจ

ในบรรทัดที่ 5 แสดงให้เห็นว่า แหล่งจ่ายกระแส J2 มีทิศทางจาก
ศูนย์รวม 0 ไปยังศูนย์รวม 2 จะถูกควบคุมด้วยโวลเตจที่ตกคร่อมระหว่างศูนย์รวม
1 กับศูนย์รวม 0 โดยที่ศักดาที่ศูนย์รวม 1 มีค่ามากกว่าระดับโวลเตจที่ศูนย์รวม 0
ส่วนความสัมพันธ์ระหว่างกระแสและโวลเตจที่ควบคุมจะแสดงได้ดังนี้

$$J2 = g(V_1 - V_0)$$

ซึ่งเท่ากับ gV_1 เมื่อศูนย์รวม 0 เป็น Datum node

ผลจากการคำนวณด้วยโปรแกรม SNAP มีดังนี้

Voltage at node 1 =

$$\frac{[(-1.000000E+0000 \cdot (V1) \cdot (R2))]}{$$

/

$$[(1.000000E+0000 \cdot (R2))$$

$$+(1.000000E+0000 \cdot (R1))$$

]

Voltage at node 2 =

$$\frac{[(-1.000000E+0000 \cdot (V1) \cdot (R2) \cdot (g2) \cdot (R3) \cdot (R4))]}{$$

/

$$[(1.000000E+0000 \cdot (R2) \cdot (R4))$$

$$+(1.000000E+0000 \cdot (R1) \cdot (R4))$$

$$+(1.000000E+0000 \cdot (R2) \cdot (R3))$$

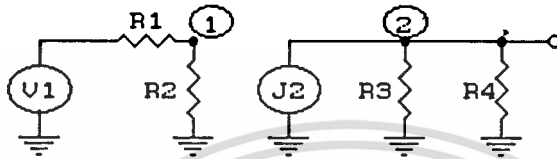
$$+(1.000000E+0000 \cdot (R1) \cdot (R3))$$

]

ค่าโวลเตจที่ศูนย์รวม 1 เป็นค่าจากการคำนวณการแบ่งโวลเตจและ
ค่าโวลเตจที่ศูนย์รวม 2 จะอยู่มีตัวแปร $g2$ อยู่ด้วย ซึ่งหมายถึงอัตราส่วนของ
กระแสที่แหล่งจ่ายกระแส J2 ต่อโวลเตจที่มาควบคุม 2 เป็นหมายเลขของ
แหล่งจ่ายกระแส

ตัวอย่างที่ 6

รูป 6.19 เป็นวงจรที่มีแหล่งจ่ายกระแสแบบควบคุมด้วยกระแสในอีก Branch หนึ่ง ในวงจรจะมีแหล่งจ่ายกระแส J2 ซึ่งควบคุมด้วยขนาดของกระแสที่ไหลผ่านตัวต้านทาน R2 ตัวอย่างหนึ่งของแหล่งจ่ายกระแสแบบควบคุมด้วยกระแสคือวงจรสมมูลภายในของทรานซิสเตอร์รูป 6.9



รูป 6.19 วงจรที่ประกอบด้วยแหล่งจ่ายกระแสควบคุมด้วยกระแส

แฟ้มข้อมูลการต่ออุปกรณ์สามารถเขียนได้ดังนี้

*Current controlled current source

V1 0 1

R1 1 0

R2 1 0

J2 0 2 J 1 0

R3 0 2

R4 0 2

ในบรรทัดที่ 5 กระแสระหว่างศูนย์รวม 1 กับศูนย์รวม 0 จะควบคุมแหล่งจ่ายกระแส J2 เนื่องจากระหว่างศูนย์รวม 1 กับศูนย์รวม 0 ประกอบด้วยอุปกรณ์หลายตัว ดังนั้นเมื่อต้องการให้กระแสที่ไหลผ่านตัวต้านทาน R2 เป็นกระแสควบคุม จึงเขียนการต่อ R2 ไว้ที่บรรทัดที่ 4 ก่อนแหล่งจ่ายกระแส J2 เมื่อคำนวณด้วยโปรแกรม SNAP จะได้ผลลัพธ์แสดงโวลเตจที่ศูนย์รวมต่างๆดังนี้

Voltage at node 1 =

$$\frac{[(-1.000000E+0000 \cdot (V1) \cdot (R2))]}{[(1.000000E+0000 \cdot (R2)) + (1.000000E+0000 \cdot (R1))]}$$

Voltage at node 2 =

$$\frac{[(-1.000000E+0000 \cdot (V1) \cdot (Beta2) \cdot (R3) \cdot (R4))]}{[(1.000000E+0000 \cdot (R2) \cdot (R4)) + (1.000000E+0000 \cdot (R1) \cdot (R4)) + (1.000000E+0000 \cdot (R2) \cdot (R3)) + (1.000000E+0000 \cdot (R1) \cdot (R3))]}$$

โวลเตจที่ศูนย์รวม 2 จะมีเทอม Beta2 ติดอยู่ คืออัตราส่วนของ
กระแสในแหล่งจ่ายกระแส J2 ต่อกระแสที่มากวนคม

ตัวอย่างที่ 7

เป็นตัวอย่างของวงจรที่มีแหล่งจ่ายโวลเตจควบคุมด้วยกระแสต่ออยู่
 ดังรูป 6.20 อัตราส่วนของโวลเตจ V_2 ต่อกระแสที่ควบคุมมีค่าเท่ากับ R_{m2}
 โดบที่ 2 เป็นหมายเลขของแหล่งจ่ายโวลเตจ ซึ่งเมื่อเขียนเป็นแฟ้มข้อมูลการ
 ต่ออุปกรณ์จะสามารถเขียนได้เป็น

*Current controlled voltage source

V1 0 1

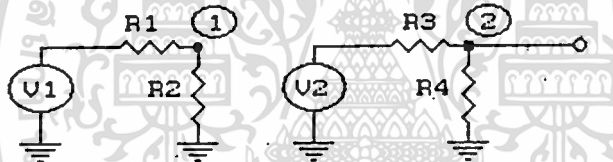
R1 1 0

R2 1 0

V2 0 2 J 1 0

R3 0 2

R4 0 2



รูป 6.20 วงจรที่ประกอบด้วยแหล่งจ่ายโวลเตจควบคุมด้วยกระแส

โวลเตจ V_2 จะถูกควบคุมด้วยกระแสระหว่างศูนย์รวม 0 กับศูนย์รวม
 1 ซึ่งมีอุปกรณ์หลายตัวต่ออยู่ เมื่อต้องการให้กระแสที่มาควบคุมคือกระแสที่ไหล
 ผ่าน R_2 จึงเขียนการต่อของ R_2 ไว้ก่อน และอยู่ติดกับแหล่งจ่ายโวลเตจ V_2

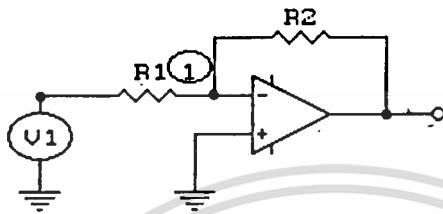
โปรแกรม SNAP จะคำนวณและให้ผลลัพธ์ดังนี้

```
Voltage at node 1 =  
[(-1.000000E+0000*(V1)*(R2))]  
/  
[( 1.000000E+0000*(R2))  
+( 1.000000E+0000*(R1))  
]  
Voltage at node 2 =  
[(-1.000000E+0000*(V1)*(Rm2 )*(R4))]  
/  
[( 1.000000E+0000*(R2)*(R4))  
+( 1.000000E+0000*(R1)*(R4))  
+( 1.000000E+0000*(R2)*(R3))  
+( 1.000000E+0000*(R1)*(R3))  
]
```

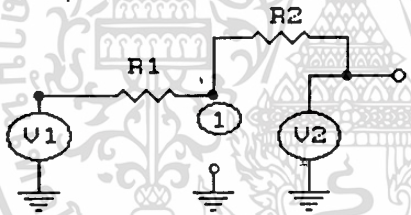
ซึ่งจะเห็นว่าโวลเตจที่ศูนย์รวม 2 มีเทอมของ Rm2 ประกอบอยู่ด้วย

ตัวอย่างที่ 8

ในตัวอย่างนี้จะยกตัวอย่างของวงจรขยายแบบกลับเฟสของสัญญาณ โดยใช้วงจรขยายเชิงค่าเนกการคังรูป 6.21a



รูป 6.21a วงจรขยายแบบกลับเฟสของสัญญาณ



รูป 6.21b วงจรสมมูลย์ของวงจรขยายแบบกลับเฟสของสัญญาณ

จากรูป 6.21a เมื่อเขียนวงจรสมมูลย์จะแสดงได้ดังรูป 6.21b โดยประมาณให้อิมพีแดนซ์ขาเข้าของวงจรขยายเชิงค่าเนกการคังเป็นอนันต์ และเอาท์พุทอิมพีแดนซ์ มีค่าเป็นศูนย์

จากวงจรในรูป 6.21b สามารถเขียนเป็นแฟ้มข้อมูลการต่ออุปกรณ์
ได้ดังนี้

*Inverting amplifier

V1 0 1

R1 0 1

V2 0 1 V 0 1

R2 0 1

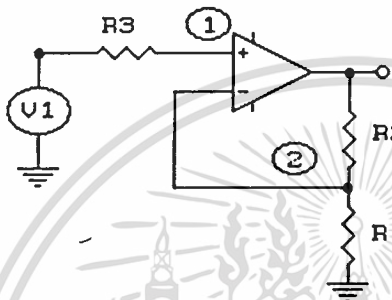
เมื่อคำนวณโดยโปรแกรม SNAP จะได้ผลลัพธ์ซึ่งเป็นโวลเตจที่
ศูนย์รวม 1 ดังนี้

Voltage at node 1 =
[(-1.000000E+0000*(V1)*(R2))]
/
[(-1.000000E+0000*(R2))
+(1.000000E+0000*(R1))
+(1.000000E+0000*(R1)*(Alpha2 -))
]

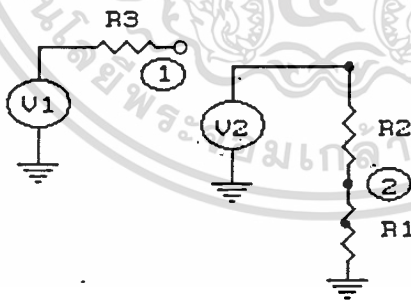
แหล่งจ่ายโวลเตจ V2 มีค่าเท่ากับ αV_1 เมื่อประมาณให้อัตราขยาย
 α มีค่าเป็นอนันต์ จะได้ว่าสัญญาณออกจากวงจรขยายมีค่าอัตราขยายเท่ากับ
 $-\frac{R_2}{R_1}$

ตัวอย่างที่ 9

วงจรในรูป 6.22a เป็นวงจรขยายแบบไม่กลับเฟสของสัญญาณ สามารถเขียนเป็นวงจรสมมูลได้ดังรูป 6.22b โดยที่อินพุตที่มีค่านับเป็นอันดับและเอาต์พุตที่มีค่านับเป็นศูนย์ ตัวต้านทาน R_3 ในวงจรใส่ไปเพื่อให้ Branch นั้นมีแหล่งจ่ายโวลเตจต่ออยู่กับอุปกรณ์อื่น โดยจะไม่มีผลต่อวงจร



รูป 6.22a วงจรขยายแบบไม่กลับเฟสของสัญญาณ



รูป 6.22b วงจรสมมูลของวงจรขยายแบบไม่กลับเฟสของสัญญาณ

เมื่อเขียนเป็นแฟ้มข้อมูลการต่ออุปกรณ์เขียนได้ดังนี้

*Non inverting amplifier

V1 1 0

R3 1 0

V2 2 0 V 1 2

R1 2 0

R2 2 0

และเมื่อกำนวณด้วยโปรแกรม SNAP จะได้ผลของโวลเตจที่ศูนย์รวม
ต่างๆดังนี้

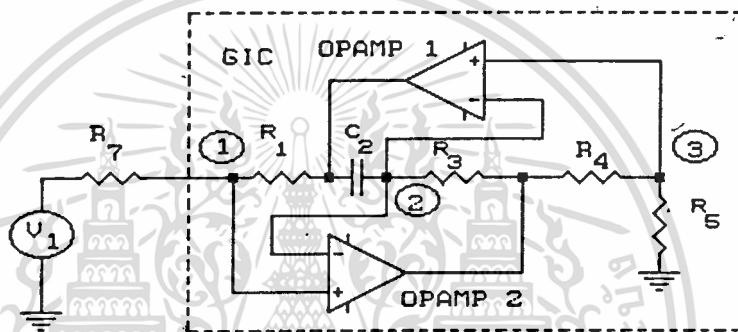
Voltage at node 2 =
[(1.000000E+0000*(V1)*(Alpha2)*(R2))]
/
[(1.000000E+0000*(R2))
+ (1.000000E+0000*(R1))
+ (1.000000E+0000*(Alpha2)*(R2))
]
Voltage at node 1 =
[(1.000000E+0000*(V1))]
/
[(1.000000E+0000)
]

ตัวอย่างที่ 10

รูป 6.23a เป็นวงจร GIC ที่ต่ออนุกรมอยู่กับตัวต้านทาน R_7 ซึ่งในวงจรประกอบด้วยวงจรขยายเชิงค่าเน็การ 2 ตัว และมีการป้อนกลับอยู่ภายในวงจร วงจร GIC ในภาพจะมีคุณสมบัติเป็นเสมือนตัวเหนี่ยวนำที่มีค่าความเหนี่ยวนำในวงจรเท่ากับ L

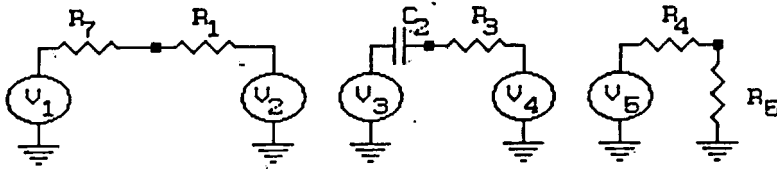
$$L = C_2 R_1 R_3 R_5$$

$$R_4$$



รูป 6.23a วงจร GIC ที่ต่ออนุกรมกับตัวต้านทาน

รูป 6.23a เป็นวงจร GIC ที่ต่ออนุกรมกับตัวต้านทาน R_7 เมื่อต้องการวิเคราะห์ด้วยโปรแกรม SNAP จะสามารถทำได้ โดยการเขียนวงจรเป็นวงจรเสมือนได้ดังรูป 6.23b ด้วยการประมาณให้อินพุตอิมพีแดนซ์ของวงจรขยายเชิงค่าเน็การสองเป็นอนันต์ และเอาท์พุตอิมพีแดนซ์ของวงจรขยายเชิงค่าเน็การสองเป็นศูนย์



รูป 6.23b วงจรเสมือนของวงจร GIC ที่ต่ออนุกรมกับตัวต้านทาน

ในรูป 6.23b นั้น แหล่งจ่ายโวลเตจ V_2 และ V_3 เป็นแหล่งจ่ายโวลเตจเดียวกัน คือเป็นแหล่งจ่ายโวลเตจที่ออกจาก OPAMP 1 และแหล่งจ่ายโวลเตจ V_4 และ V_5 ก็เป็นแหล่งจ่ายโวลเตจเดียวกัน โดยเป็นแหล่งจ่ายโวลเตจที่ออกจาก OPAMP 2 ที่ต้องเขียนแยกกันเป็น 2 แหล่งจ่ายโวลเตจ เพราะเป็นลักษณะการย้ายแหล่งจ่ายโวลเตจ เพื่อการวิเคราะห์วงจรโดยวิธี Nodal analysis

จากวงจรในรูป 6.23a ให้อัตราขยายของ OPAMP 1 และ OPAMP 2 มีค่าเท่ากับ Alpha

กำหนดให้อุปกรณ์ในรูป 6.23a มีค่าดังนี้

$$R_7 = 1.0$$

$$R_1 = 2.0$$

$$C_1 = 3.0$$

$$R_3 = 4.0$$

$$R_4 = 5.0$$

$$R_5 = 6.0$$

ดังนี้

จากค่าอุปกรณ์ดังข้างต้น นำมาเขียนเป็นแฟ้มข้อมูลการต่ออุปกรณ์ได้

*GIC

V1 1 0

R7 1 0 1.0

V2 1 0 V 2 3

R1 1 0 2.0

V3 2 0 V 2 3

C1 2 0 3.0

V4 2 0 V 2 1

R3 2 0 4.0

V5 3 0 V 2 1

R4 3 0 5.0

R5 3 0 6.0

เมื่อทำการวิเคราะห์ด้วยโปรแกรม SNAP จะสามารถทราบโวลเตจที่แต่ละศูนย์รวม โดยคำตอบจะติดอยู่ในรูปของตัวแปร V1, Alpha และ S ดังนี้

Voltage at node 3 =

$$\begin{aligned} & [(-9.000000E-0001*(V1)*(Alpha2)*(S)) + \\ & (-7.500000E-0002*(V1)*(Alpha2)) + \\ & (9.000000E-0001*(V1)*(Alpha2 ^2)*(S))] \\ & / \end{aligned}$$

$$\begin{aligned} & [(2.475000E+0000*(S)) \\ & +(2.062500E-0001) \\ & +(-2.475000E+0000*(Alpha2)*(S)) \\ & +(-2.062500E-0001*(Alpha2)) \\ & +(3.125000E-0002*(Alpha2 ^2)) \\ & +(9.000000E-0001*(Alpha2 ^2)*(S)) \\ &] \end{aligned}$$

Voltage at node 2 =

$$[(-6.187500E-0001*(V1)*(Alpha2)*(S)) +$$

$$\begin{aligned} & (-5.156250E-0002*(V1)*(Alpha2)) + \\ & (9.562500E-0001*(V1)*(Alpha2 ^2)*(S)) + \\ & (5.156250E-0002*(Y1)*(Alpha2 ^2)) + \\ & (-1.718750E-0002*(V1)*(Alpha2 ^3)) + \\ & (-3.375000E-0001*(V1)*(Alpha2 ^3)*(S)) + \\ & (4.050000E+0000*(V1)*(Alpha2 ^2)*(S^2)) + \\ & (-4.050000E+0000*(V1)*(Alpha2 ^3)*(S^2)) + \\ & (1.125000E-0001*(V1)*(Alpha2 ^4)*(S))] \\ & / \\ & [(1.113750E+0001*(S^2)) \\ & +(1.856250E+0000*(S)) \\ & +(-2.227500E+0001*(Alpha2)*(S^2)) \\ & +(-3.712500E+0000*(Alpha2)*(S)) \\ & +(2.643750E+0000*(Alpha2 ^2)*(S)) \\ & +(1.518750E+0001*(Alpha2 ^2)*(S^2)) \\ & +(7.734375E-0002) \\ & +(-1.546875E-0001*(Alpha2)) \\ & +(1.148437E-0001*(Alpha2 ^2)) \\ & +(-7.875000E-0001*(Alpha2 ^3)*(S)) \\ & +(-4.050000E+0000*(Alpha2 ^3)*(S^2)) \\ & +(-3.750000E-0002*(Alpha2 ^3)) \\ & +(3.906250E-0003*(Alpha2 ^4)) \\ & +(1.125000E-0001*(Alpha2 ^4)*(S)) \\ &] \end{aligned}$$

Voltage at node 1 =

$$\begin{aligned} & [(2.756531E+0001*(V1)*(S^3)) + \\ & (6.891328E+0000*(V1)*(S^2)) + \\ & (-8.269594E+0001*(V1)*(Alpha2)*(S^3)) + \\ & (-2.067398E+0001*(V1)*(Alpha2)*(S^2)) + \\ & (2.596430E+0001*(V1)*(Alpha2 ^2)*(S^2)) + \\ & (1.077553E+0002*(V1)*(Alpha2 ^2)*(S^3)) + \\ & (5.742773E-0001*(V1)*(S)) + \\ & (-1.722832E+0000*(V1)*(Alpha2)*(S)) + \end{aligned}$$

$$\begin{aligned}
 & (2.082480E+0000*(V1)*(Alpha2 ^2)*(S)) + \\
 & (-1.747195E+0001*(V1)*(Alpha2 ^3)*(S^2)) + \\
 & (-7.768406E+0001*(V1)*(Alpha2 ^3)*(S^3)) + \\
 & (-1.293574E+0000*(V1)*(Alpha2 ^3)*(S)) + \\
 & (4.431445E-0001*(V1)*(Alpha2 ^4)*(S)) + \\
 & (6.631875E+0000*(V1)*(Alpha2 ^4)*(S^2)) + \\
 & (1.595215E-0002*(V1)) + \\
 & (-4.785645E-0002*(V1)*(Alpha2)) + \\
 & (5.559082E-0002*(V1)*(Alpha2 ^2)) + \\
 & (-3.142090E-0002*(V1)*(Alpha2 ^3)) + \\
 & (8.540039E-0003*(V1)*(Alpha2 ^4)) + \\
 & (3.052688E+0001*(V1)*(Alpha2 ^4)*(S^3)) + \\
 & (-8.349609E-0002*(V1)*(Alpha2 ^5)*(S)) + \\
 & (-1.341563E+0000*(V1)*(Alpha2 ^5)*(S^2)) + \\
 & (-8.056641E-0004*(V1)*(Alpha2 ^5)) + \\
 & (5.273438E-0003*(V1)*(Alpha2 ^6)*(S)) + \\
 & (-5.467500E+0000*(V1)*(Alpha2 ^5)*(S^3)) + \\
 & (1.518750E-0001*(V1)*(Alpha2 ^6)*(S^2))] \\
 & / \\
 & [(4.134797E+0001*(S^3)) \\
 & +(1.033699E+0001*(S^2)) \\
 & +(-1.240439E+0002*(Alpha2)*(S^3)) \\
 & +(-3.101098E+0001*(Alpha2)*(S^2)) \\
 & +(3.821555E+0001*(Alpha2 ^2)*(S^2)) \\
 & +(1.541152E+0002*(Alpha2 ^2)*(S^3)) \\
 & +(8.614160E-0001*(S)) \\
 & +(-2.584248E+0000*(Alpha2)*(S)) \\
 & +(3.158525E+0000*(Alpha2 ^2)*(S)) \\
 & +(-2.474613E+0001*(Alpha2 ^3)*(S^2)) \\
 & +(-1.014905E+0002*(Alpha2 ^3)*(S^3)) \\
 & +(-2.009971E+0000*(Alpha2 ^3)*(S)) \\
 & +(7.111230E-0001*(Alpha2 ^4)*(S)) \\
 & +(8.875195E+0000*(Alpha2 ^4)*(S^2))
 \end{aligned}$$

```

+( 2.392822E-0002)
+(-7.178467E-0002*(Alpha2 ))
+( 8.701172E-0002*(Alpha2 ^2))
+(-5.438232E-0002*(Alpha2 ^3))
+( 1.819336E-0002*(Alpha2 ^4))
+( 3.553875E+0001*(Alpha2 ^4)*(S^3))
+(-1.368457E-0001*(Alpha2 ^5)*(S))
+(-1.670625E+0000*(Alpha2 ^5)*(S^2))
+(-2.966309E-0003*(Alpha2 ^5))
+( 1.831055E-0004*(Alpha2 ^6))
+( 1.054688E-0002*(Alpha2 ^6)*(S))
+(-5.467500E+0000*(Alpha2 ^5)*(S^3))
+( 1.518750E-0001*(Alpha2 ^6)*(S^2))
]

```

เมื่อทำการประมาณค่า Alpha2 ในคำตอบ ให้มีค่าเป็นอนันต์ จะได้คำตอบที่เป็นค่าโวลเตจที่ศูนย์รวมแต่ละศูนย์รวม ดังนี้

```

Voltage at node 3 =
[( 9.000000E-0001*(V1)*(S))]
/
[( 3.125000E-0002)
+( 9.000000E-0001*(S))
]
Voltage at node 2 =
[( 1.125000E-0001*(V1)*(S))]
/
[( 3.906250E-0003)
+( 1.125000E-0001*(S))
]

```

```
Voltage at node 1 =  
[( 1.518750E-0001*(V1)*(S))]  
/  
[( 1.054688E-0002)  
+( 1.518750E-0001*(S))  
]
```

จะเห็นว่าเมื่อทำการประมาณ Alpha2 เป็นอนันต์แล้ว โวลเตจที่ศูนย์รวม 1,2 และ 3 จะมีค่าเท่ากัน เพราะเมื่อประมาณให้วงจรขยายเชิงค่าเน้นการมีอัตราขยายและอินพุทอิมพีแดนซ์เป็นอนันต์ และให้เอาต์พุทอิมพีแดนซ์มีค่าเป็นศูนย์แล้ว โวลเตจที่ขาอินพุททั้งสองของวงจรขยายเชิงค่าเน้นการจะมีค่าเท่ากัน สมการแสดงโวลเตจที่ศูนย์รวม 1 แสดงว่าเป็นจุดที่แสดงคุณสมบัติของวงจรผ่านความถี่สูง



6.8 เอกสารอ้างอิงบทที่ 6

- [1] Kobylarz T.J., "Simulations of Linear and Nonlinear Circuits for Symbolic Analysis and Transient Response, IEEE Trans. Education, Vol.E-12, No.4., pp.299-305, Dec.1969
- [2] Mey G.D., "ASNAP a Symbolic Network Analysis Program," INT.J. Electronics, Vol.67, No.3, pp.385-392, March 1989.
- [3] Pookaiyaudom S. and Kuhanont T., "Automatic Symbolic Polynomial Operations," IEEE Proc., 1977.
- [4] Sedra A.S. and Smith K.C., Microelectronic circuits, Holt-Saunders International Edition, Japan, 1982.
- [5] Borland International, "Turbo Pascal 5.5 reference," 1990.



บทที่ 7
สรุป
(Summary)

7.1 สรุป

การศึกษาระบบด้วยวิธีแบ่งเวลา สามารถใช้กับระบบในแบบต่างๆ เช่น ระบบป้อนกลับที่มีส่วนไม่เชิงเส้นในระบบ หรือระบบที่มีการหน่วงเวลา ทั้งนี้ในการเปลี่ยนจากระบบแบบเวลาต่อเนื่องมาเป็นระบบแบบแบ่งเวลา ต้องอาศัยการทรานส์ฟอร์มเพื่อให้ได้ผลทั้งสองแบบใกล้เคียงกันถึงแม้ว่าการจำลองระบบป้อนกลับด้วยดิจิทัลคอมพิวเตอร์จะเป็นการโปรแกรมแบบลำดับก็ตาม

ในบทที่ 2 ได้เสนอการศึกษาระบบที่มีการป้อนกลับแบบไม่เชิงเส้น โดยใช้วิธีการจำลองระบบแบบแบ่งเวลา ด้วยการใช่วงจรออสซิลเลเตอร์แบบวินบริดจ์เป็นระบบศึกษาตัวอย่าง ทั้งนี้เพราะสามารถเปรียบเทียบระหว่างผลการจำลองกับผลการทดลองของวงจรจริงได้ง่าย โดยได้ทำการศึกษาปรากฏการณ์แบบชั่วขณะ (Transient) ซึ่งยังไม่ค่อยมีการศึกษามาก่อน รวมทั้งยังทำการศึกษาค่าความไม่เชิงเส้นของระบบด้วยการสร้างแบบจำลองจากคุณสมบัติจริงด้วย โดยไม่ต้องใช้วิธีประมาณเช่นในบทความ [2] ของบทที่ 2 จากการเปรียบเทียบผลการจำลองระบบด้วยคอมพิวเตอร์แบบแบ่งเวลากับผลจากการทดลองดังรูป 2.7, 2.8 และ 2.9 จะเห็นว่าการสร้างรูปแบบการจำลองระบบดังข้อ ก, ข, ค และ ง ในหัวข้อที่ 2.4 ให้ผลใกล้เคียงกันมาก ซึ่งการจำลองของบทที่ 2 นั้นแสดงให้เห็นสัญญาณออกจากออสซิลเลเตอร์นับตั้งแต่เวลาที่ออสซิลเลเตอร์เริ่มทำงานคือมีการป้อนแหล่งจ่ายไฟให้กับออสซิลเลเตอร์ ในขณะที่ออสซิลเลเตอร์จะเริ่มต้นการออสซิลเลตด้วยสัญญาณรบกวนระดับต่ำหรือสัญญาณอิมพัลส์ก็ได้ ในการจำลองได้แสดงให้เห็นผลอย่างชัดเจนทั้งกรณีเริ่มต้นการออสซิลเลตด้วยวิธี Hard start และวิธี Soft start ซึ่งมีผลต่อเวลาที่ใช้ในการเข้าสู่สภาวะอยู่ตัวของออสซิลเลเตอร์ โดยวิธี Hard start จะไม่จำเป็นต้องใช้เวลานานนักในการที่สัญญาณอิมพัลส์จะต้องวิ่งวนอยู่ในออสซิลเลเตอร์ให้มีขนาดเพิ่มขึ้น อันเนื่องจากว่าสัญญาณอิมพัลส์เริ่มต้นมีขนาดใหญ่มากกว่าสัญญาณรบกวนระดับต่ำมาก และในวิธี Soft start ต้องใช้เวลาในการที่สัญญาณรบกวนจะต้องวิ่งวนอยู่ในออสซิลเลเตอร์เพื่อให้มีขนาดเพิ่มขึ้น เพราะในตอนเริ่มต้นการออสซิลเลตนั้น สัญญาณรบกวนระดับต่ำมีขนาดเล็กและต้องวิ่งวนอยู่ภายในออสซิลเลเตอร์เป็นเวลานาน สัญญาณรบกวนที่เข้ามาใหม่มีโอกาสดูที่เฟสของสัญญาณนั้นจะหักล้างกับสัญญาณที่วิ่งวนอยู่ ทำให้ต้องใช้เวลาในการเข้าสู่สภาวะอยู่ตัวมาก

ยิ่งขึ้น แต่การเริ่มออสซิลเลตของออสซิลเลเตอร์จะขึ้นกับวงจรควบคุมอัตราขยายอัตโนมัติด้วย โดยจากการศึกษาพบว่าวงจรขยายในวงจรออสซิลเลเตอร์แบบวินบริจที่มีอัตราขยายขึ้นกับค่าสัมบูรณ์ของขดคัลลินั้น จะเข้าสู่สถานะอยู่ตัวเร็วกว่าแบบควบคุมอัตราขยายด้วยค่าเฉลี่ยจากค่าสัมบูรณ์ของสัญญาณออกจากออสซิลเลเตอร์

สำหรับวงจรออสซิลเลเตอร์ที่มีวงจรควบคุมอัตราขยายแบบควบคุมด้วยค่าเฉลี่ยจากค่าสัมบูรณ์ของสัญญาณออกจากออสซิลเลเตอร์นั้น เป็นวงจรที่น่าสนใจมาใช้งานได้ง่าย โดยที่สัญญาณออกจากออสซิลเลเตอร์จะมีการพุ่งเกิน (Overshoot) สูงมากหรือน้อยขึ้นอยู่กับค่าคงตัวเวลา (Time constant) ของส่วนอินทิเกรเตอร์ที่มีระดับโวลเตจไปควบคุมอัตราขยายของวงจรขยาย การจำลองระบบด้วยคอมพิวเตอร์จะช่วยให้สามารถเลือกวงจรควบคุมอัตราขยายที่เหมาะสมหรือการเลือกค่าคงตัวเวลาที่เหมาะสม ทำให้ได้ออสซิลเลเตอร์ที่ผลิตสัญญาณมีลักษณะตามความต้องการได้ เช่นต้องการออสซิลเลเตอร์ที่เข้าสู่สถานะอยู่ตัวเร็วที่สุดโดยที่มีการพุ่งเกินน้อยที่สุด ก็จะเลือกออสซิลเลเตอร์แบบควบคุมอัตราขยายด้วยค่าสัมบูรณ์ของขดคัลลิน แต่ถ้าการใช้งานวงจรมีความยุ่งยากเกินไปก็ใช้แบบควบคุมอัตราขยายด้วยค่าเฉลี่ยจากค่าสัมบูรณ์ของสัญญาณออกจากออสซิลเลเตอร์ โดยเลือกค่าคงตัวเวลาของส่วนอินทิเกรเตอร์ไม่ให้เกิดการพุ่งเกินมากนักด้วยการจำลองวงจรเพื่อคุณสมบัติของวงจรเสียก่อน

การเปรียบเทียบผลการจำลองระบบด้วยคอมพิวเตอร์และการทดลองสามารถยืนยันได้ถึงการจำลองระบบซึ่งมีความไม่เชิงเส้น ทำให้สามารถใช้ในการศึกษาออสซิลเลเตอร์แบบอื่นๆได้ด้วย หรือใช้ในการศึกษาปรากฏการณ์อื่นๆของออสซิลเลเตอร์ เช่น การศึกษาผลของสัญญาณรบกวนแบบต่างๆซึ่งมีแบบวิหัทธ์ของสัญญาณรบกวนต่าง ๆ กัน ก็สามารถจำลองสัญญาณรบกวนดังกล่าวโดยใช้การจำลองวงจรผ่านแถบความถี่ (Bandpass filter) ซึ่งมีช่วงผ่านแถบความถี่ต่าง ๆ กันดังในบทความ [5],[6],[7] ของบทที่ 2 ทำให้ได้สัญญาณรบกวนแบบต่างๆกัน นอกจากนั้นวิธีการจำลองแบบนั้นสามารถสร้างสัญญาณรบกวนแบบต่างๆ โดยมีการศึกษาดังในบทความ [8],[11],[12]

ในกรณีที่สัญญาณรบกวนมีความถี่เดียวและใกล้เคียงกับความถี่ทำงาน-เสริชองออสซิลเลเตอร์ และมีขนาดเพียงพอก็จะเกิดปรากฏการณ์ล็อกกิ้งขึ้นได้อันเป็นปรากฏการณ์ที่พบได้มากและยังมีประโยชน์ด้วย เช่นในการออกแบบออสซิลเลเตอร์ที่มีความเสถียรสูง

ในบทที่ 3 ได้นำเอาอัลกอริธึมการจำลองระบบป้อนกลับแบบไม่เชิงเส้นโดยวิธีแบ่งเวลาในบทที่ 2 มาศึกษาปรากฏการณ์อื่นเจกชั้นล็อกกิ้ง โดย

เสมือนกับการเปลี่ยนสัญญาณรบกวนระดับต่ำหรือสัญญาณอิมพัลส์ที่ป้อนเข้าสู่การจำลองบทที่ 2 ด้วยสัญญาณขาขึ้นที่มีความถี่เดียวเพื่อล็อกออสซิลเลเตอร์นั้น ซึ่งการจำลองนี้มีประโยชน์มากในการออกแบบและการห่าย่านความถี่ของสัญญาณภายนอกที่สามารถล็อกออสซิลเลเตอร์ได้ ในการศึกษาปรากฏการณ์อินเจกชัน-ล็อกกิ้งของออสซิลเลเตอร์ด้วยวิธีแบ่งเวลานั้น สามารถทำได้กับกรณีที่ขนาดของสัญญาณที่ใส่เข้าไปล็อกออสซิลเลเตอร์มีขนาดใหญ่ หรือค่าคงตัวเวลาของวงจรควบคุมอัตราขยายมีค่าต่าง ๆ กัน เมื่อสัญญาณที่ป้อนเข้าไปล็อกออสซิลเลเตอร์มีขนาดใหญ่ จะทำให้อัตราขยายของวงจรขยายมีค่าลดลงและระบบรวมของออสซิลเลเตอร์จะมีอัตราขยายวนรอบน้อยกว่า 1 ทำให้วงจรออสซิลเลเตอร์ทำงานเสมือนเป็นวงจรผ่านแถบความถี่ที่มีความถี่เรโซแนนท์เท่ากับความถี่ทำงานเสรีของออสซิลเลเตอร์ ถ้าความถี่ของสัญญาณที่เข้าไปล็อกออสซิลเลเตอร์แตกต่างจากความถี่ทำงานเสรีของออสซิลเลเตอร์มากเท่าใด ขนาดของสัญญาณที่ออกจากออสซิลเลเตอร์ก็จะมีความถี่ลดลง จนกระทั่งความถี่นั้นไม่สามารถทำให้เกิดการล็อกได้ก็จะปรากฏว่าสัญญาณที่ออกจากออสซิลเลเตอร์มีความถี่บิดเท่ากับผลต่างของความถี่ทำงานเสรีของออสซิลเลเตอร์กับความถี่ของสัญญาณที่ป้อนเข้าไป โดยที่การจำลองทำให้สามารถบอกเฟส, ขนาดของอัตราขยาย, ขนาดของสัญญาณจากออสซิลเลเตอร์ที่แต่ละเวลาได้โดยง่าย

ในกรณีที่การล็อกกันเป็นแบบเชื่อมต่อนั้น ก็สามารถศึกษาได้โดยวิธีการจำลองระบบแบบแบ่งเวลา ซึ่งการล็อกกันแบบเชื่อมต่อนั้นไม่สามารถหาคำตอบโดยวิธีทางคณิตศาสตร์ได้ง่าย ในการจำลองจะมี 2 ระบบต่อเข้าด้วยกันเป็นระบบใหญ่ และมีอัลกอริธึมการจำลองแบบเรียงลำดับช่วยให้สามารถจำลองทั้งสองระบบได้พร้อมกัน โดยจำลองแต่ละระบบเช่นเดียวกับในบทที่ 2 ในกรณีการล็อกแบบเชื่อมต่อนั้นย่านล็อกจะกว้างกว่ากรณีของอินเจกชันล็อกกิ้งทางเดียวดังผลการจำลองและทดลองในรูป 3.7 และ 3.8 และความถี่จากการออสซิลเลตของการล็อกแบบเชื่อมต่อนั้นจะไม่เท่ากับความถี่ทำงานเสรีของออสซิลเลเตอร์ตัวใดตัวหนึ่ง แต่เป็นความถี่ระหว่างความถี่ทำงานเสรีของออสซิลเลเตอร์สองตัวดังรูป 3.9 จะเห็นว่าผลการทดลองกับการจำลองระบบด้วยวิธีแบ่งเวลาสำหรับปรากฏการณ์อินเจกชันล็อกกิ้งและปรากฏการณ์เชื่อมต่อนั้นทั้งรูป 3.7 และ 3.8 มีความใกล้เคียงกันมาก ทำให้สามารถศึกษาหรือออกแบบวงจรอินเจกชันล็อกกิ้งของออสซิลเลเตอร์ได้แม้จะมีค่าสัมประสิทธิ์การเชื่อมต่อสูงซึ่ง Adler ไม่สามารถทำได้ อีกทั้งสมการที่จะใช้อัลกอริธึมนี้สำหรับการจำลองระบบแบบไม่เชิงเส้นที่มีสัญญาณรบกวนแบบต่าง ๆ ได้

บทที่ 4 เป็นการจำลองตัวหน่วงเวลา โดยการจำลองวงจรออสซิล

เลเตอร์แบบควบคุมด้วยดีเลย์ไลน์ ซึ่งลักษณะของดีเลย์ไลน์เป็นสารพีโซอิเล็ก-
ตริกที่มีคลื่นเคลื่อนที่ไปบนผิว ดังนั้นการจำลองในส่วนนี้คือการจำลองการเคลื่อน
ที่ของคลื่นไปบนผิว จึงต้องหาอัลกอริธึมที่มีประสิทธิภาพ ทั้งในการศึกษาตอนแรก
จะทำการจำลองระบบที่มีคุณสมบัติของระบบขึ้นกับส่วนของดีเลย์ไลน์เพียงอย่าง
เดียวโดยศึกษาการกระโดดความถี่ของออสซิลเลเตอร์ หรือการเปลี่ยนโมด
เพราะโมดของการออสซิลเลเตอร์ขึ้นกับความยาวของดีเลย์ไลน์ระหว่างอิเล็ก-
โทรด และในการจำลองนี้ให้ IDT มีความกว้างเพียง 1 จุดของคาบการแบ่ง
เวลา ดังนั้นแบนวิทซ์ของ IDT จึงเป็นอนันต์สำหรับการจำลอง ซึ่งในทางปฏิบัติ
แล้ว BBD ที่ใช้ในการทดลองมีแบนวิทซ์จำกัด แต่แบนวิทซ์ดังกล่าวมีค่ามากกว่า
แบนวิทซ์ของวงจรผ่านความถี่ภายนอกมาก และให้ทิศทางการเคลื่อนที่ของคลื่น
มีเพียงทิศทางเดียว แสดงให้เห็นการเปลี่ยนแปลงของสัญญาณสัญญาณออกจาก
ดีเลย์ไลน์ได้ชัดเจน แสดงว่าอัลกอริธึมดังกล่าวจะช่วยให้การคำนวณใช้เวลา
รวดเร็ว และประหยัดหน่วยความจำที่ใช้สำหรับการจำลองดีเลย์ไลน์ด้วย ซึ่ง
ในการจำลองนี้สามารถสร้างแบบจำลองให้แทนคุณสมบัติของส่วนต่างๆในวงจร
ได้อย่างสะดวก โดยที่ผลจากการจำลองที่ 1 และผลการทดลองที่ 1 ให้ผล
ใกล้เคียงกัน จึงทำการจำลองที่ 2 และการจำลองที่ 3 เพื่อหาเวลาในการ
ข้ามจากโมด 16 ไปยังโมด 20 ของออสซิลเลเตอร์ที่จำนวนตัวรับต่างๆ และ
อัตราขยายรอบค่าต่างๆกัน อัลกอริธึมสำหรับการจำลองดีเลย์ไลน์ที่เสนอยัง
สามารถนำไปใช้ในการจำลองการหน่วงเวลาในวงจรดิจิทัลสวิทช์(Digital
switching) โดยการจำลองการหน่วงเวลาแต่ละจุดในวงจรก็เพียงแต่ใส่อัล-
กอริธึมการจำลองดีเลย์ไลน์เข้าไปที่ช่วงนั้น ให้มีอัตราส่วนของการหน่วงสัมพันธ์
กับที่จุดอื่นในวงจร

ในดีเลย์ไลน์ที่เป็นสารพีโซอิเล็กตริก จะมีตัวเปลี่ยนสัญญาณไฟฟ้าเป็น
พลังงานคลื่นเคลื่อนที่ไปบนผิวของพีโซอิเล็กตริกในทั้งสองทิศทาง และเปลี่ยนจาก
พลังงานคลื่นเป็นสัญญาณไฟฟ้า การจำลองส่วนนี้แสดงให้เห็นได้ในบทที่ 5 ซึ่ง
แสดงการจำลองผลตอบสนองของ IDT ซึ่งมีความกว้างต่างๆ ความกว้างของ
IDT มีผลต่อย่านตอบสนองความถี่ของสัญญาณและอัลกอริธึมที่เสนอ จะมีการคํานวณ
เฉพาะที่ขอบของอิเล็กโทรด ทำให้การจำลองของ IDT ตัวรับเป็นไปได้
อย่างรวดเร็ว และที่ IDT ตัวส่งจะมีอัลกอริธึมในการหาพลังงานคลื่นที่เคลื่อนที่
ออกจาก IDT ตัวส่ง โดยอัลกอริธึมจะหาพลังงานคลื่นจากสัญญาณไฟฟ้าที่เวลา
ต่างๆกัน และมีจำนวนครั้งการคำนวณขึ้นกับจำนวนขอบของอิเล็กโทรดเท่านั้น
ไม่ขึ้นกับจำนวนจุดที่แบ่งบนอิเล็กโทรด นอกจากนี้อัลกอริธึมที่เสนอยังสามารถ
จำลองการเคลื่อนที่ของคลื่นที่มีอัตราการสูญเสียในระหว่างการเดินทางเคลื่อนที่ได้อย่าง

สะดวกและรวดเร็ว อีกทั้งประหยัดหน่วยความจำด้วย สำหรับการสูญเสียที่เกิดขึ้นในระหว่างการเคลื่อนที่ นอกจากจะจำลองโดยการคิดเป็นการสูญเสียรวมทั้งตัวรับหรือตัวส่งแล้ว ยังสามารถจำลองได้โดยการแก้ไขในสมการ 5.3a และ 5.3b ใ้ค่า $T_{FM} + R_{FM} < 1.0$ และ $T_{MF} + R_{MF} < 1.0$ ซึ่งไม่มีผลต่อความเร็วของอัลกอริธึมที่เสนอ

ในการจำลองที่ 1,2 ของบทที่ 5 เป็นการจำลองเพื่อหาผลตอบสนองต่ออิมพัลส์ของนอนคิสเพอร์ซีฟและนอนอะโพไดซ์คิลเบย์ไลน์ ซึ่งเมื่อเปรียบเทียบกับผลในบทความ [3] ของบทที่ 5 จะเห็นว่าได้ผลใกล้เคียงกันในทุกกรณี แม้กระทั่งกรณีที่ม้อัตราการสะท้อนที่ขอบของอิเล็กโทรดค่าต่างๆกัน แสดงว่าอัลกอริธึมที่ใช้มีความถูกต้อง สำหรับอัตราส่วนของขนาดสูงสุดของยอดคลื่นที่ความถี่ต่างๆ จะขึ้นกับอัตราส่วนของความกว้างของฟิงเกอร์ต่อความกว้างของช่องว่างระหว่างฟิงเกอร์ ดังผลการจำลองที่ 3 เทียบกับบทความ [3] การจำลองนี้จะช่วยในการออกแบบวงจรผ่านแถบความถี่ของฟิงเกอร์ได้สะดวก และการจำลองที่ 4 ได้เปรียบเทียบผลของอัตราการสะท้อนค่าต่างๆกัน ที่มีผลต่อการตอบสนองความถี่ของสัญญาณ และการจำลองที่ 5 แสดงให้เห็นถึงอัลกอริธึมดั่งที่ใช้ในการจำลองที่ 1 ถึง 4 ซึ่งจำลอง IDT แบบคิสเพอร์ซีฟ มาใช้จำลอง IDT แบบนอนคิสเพอร์ซีฟ โดยที่เวลาในการจำลองจะเพิ่มขึ้นอันเนื่องมาจากจำนวนของฟิงเกอร์เท่านั้น ทั้งนี้การจำลองดังกล่าวสามารถใช้กับการจำลองวงจรดิจิทัลที่ประกอบด้วยอุปกรณ์ Programmable Array Logic ก็ได้

ในบทที่ 6 เป็นโปรแกรมวิเคราะห์เน็ตเวิร์คด้วยสัญลักษณ์ โปรแกรม SNAP เป็นโปรแกรมที่ช่วยวิเคราะห์ลักษณะของระบบ โดยที่สามารถเลือกอุปกรณ์ใดเป็นตัวแทนก็ได้ ผลการวิเคราะห์นั้นจะอยู่ในรูปของ S-domain ซึ่งจะสามารถนำไปวิเคราะห์ต่อเพื่อหาค่าสมบัติใน Time domain ต่อไปได้ ดังวิธีในตอนต้นซึ่งกล่าวถึงการศึกษาการจำลองระบบแบบ Discrete time domain นอกจากนั้น ในบทนี้ได้แสดงถึงอัลกอริธึมการจัดการกับหน่วยความจำของคอมพิวเตอร์ เพื่อให้สามารถทำงานกับตัวแปรต่างๆในรูปตัวเลขและจัดการกับตัวแปรจำนวนมากได้ด้วยวิธีการแบบ Record อีกทั้งวิธีการจัดการเก็บสัญลักษณ์ของอุปกรณ์ในวงจรที่จะนำมาวิเคราะห์ในแบบ Polynomial โดยโพลีโนเมียลกำลังค่าหนึ่งก็จะจัดการให้เป็นตัวแปรตัวหนึ่ง

ในส่วนของอัลกอริธึมในบทที่ 6 นี้สามารถทำความเข้าใจได้โดยง่าย เพราะมีการจัดการข้อมูลกับแผ่นแม่เหล็กในรูปของไฟล์ต่างๆโดยตลอด และโปรแกรมในภาคผนวกแสดงให้เห็นว่ามีการเรียกส่วนต่างๆแบบ Procedure ซึ่งเป็นข้อดีของ TURBO PASCAL ที่เลือกใช้ เพราะทำให้เพิ่มความสามารถ

ของโปรแกรมในการวิเคราะห์วงจรต่อไปในอนาคต สำหรับการวิเคราะห์วงจรในโปรแกรม SNAP นี้ ใช้วิธี Nodal analysis โดยใช้กฎข้อที่ 2 ของเคอร์ชอฟ เนื่องจากสามารถสร้างเมตริกซ์ได้โดยง่าย แต่อัลกอริธึมดังกล่าวก็สามารถใช้วิเคราะห์โดยใช้กฎข้อที่ 1 ของเคอร์ชอฟก็ได้ ดังนั้นเมื่อต้องการวิเคราะห์ระบบแบบแบ่งเวลาที่มีขนาดใหญ่ ก็ทำได้โดยแบ่งวงจรออกเป็นส่วนย่อยๆ แล้วหาทรานสเฟอร์ฟังก์ชันของส่วนย่อยๆ นั้น โดยใช้โปรแกรม SNAP เพื่อหาทรานสเฟอร์ฟังก์ชัน แล้วสร้างแบบจำลองเช่นในบทที่ 2 สำหรับอุปกรณ์ทั่วไป ทั้งนี้ตัวอย่างการใช้โปรแกรมวิเคราะห์วงจรด้วยสัญลักษณ์ ได้แสดงให้เห็นถึงความสามารถของโปรแกรมในการวิเคราะห์วงจรแบบสัญลักษณ์ ที่มีอุปกรณ์แบบต่างๆ กัน และยังสามารถที่จะกำหนดค่าให้ตัวแปรบางตัวมีค่าเท่ากันได้ด้วย ในกรณีที่ส่วนนั้นมีการห้วงสัญญาณจะใช้อัลกอริธึมในบทที่ 4 และ 5 อัลกอริธึมดังกล่าวนอกจากสามารถใช้จำลองระบบแบบแอนะล็อกแล้ว ยังสามารถใช้จำลองระบบแบบดิจิทัลได้ด้วย



กิตติกรรมประกาศ

ขอขอบพระคุณ ดร.สิทธิชัย โกโคบายอุดม,ดร.พัศศักดิ์ ชีวสุวิทย์,
ดร.วิวัฒน์ กิรานนท์,ดร.วัลลภ สรกาพลธร,ดร.จเร สุรวุฒน์ปัญญา ที่กรุณาให้คำ
ปรึกษาในระหว่างการศึกษาเป็นอย่างสูง



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ก. Simulation for Sinusoidal Oscillators with
Random Numbers and Initial Impulses.



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Simulating Sinusoidal Oscillators with Random Numbers and Initial Impulses

Sittichai Pookaiyaudom
Sujale Jantarang

Abstract

This paper discusses a discrete-time numerical method for simulating complex sinusoidal oscillators. Two modes for the spontaneous starting of oscillations can be simulated. The first method is based on the random electrical noise in the circuit, and the second method is due to the transient, when the power supply is switched on abruptly. These two distinct oscillation starting modes have not been generally discussed in the literature. The amplifier nonlinearities, as well as the automatic amplitude control function, can also be incorporated into the simulation. Very efficient computational algorithms for the output calculation of nonlinearities will be presented. Experimental results have also been obtained to show the accuracy of the simulated results. The paper ends with the proposal that this simulation method can be employed as a powerful tool for both the transient and steady-state analysis and the design of sinusoidal oscillators.

Introduction

The solutions to sinusoidal oscillator problems, from the starting moments to the steady-state oscillations, have always been considered difficult [1]. This is especially so if the amplifier nonlinearities and the automatic amplitude control functions are to be incorporated into the solutions [2]. Consequently, convenient analysis and design tools that cover the complete oscillator characteristics, from initial moments to steady-state oscillations, are not readily available to circuit designers; most electronic textbooks virtually ignore meaningful discussions of such problems.

Furthermore, the spontaneous starting of oscillations, resulting from both the random electrical noises present in the circuits as well as the power supply turn-on transient impulses, cannot be easily demonstrated convincingly in practice. Therefore, most electronic circuit students remain reluctant to accept the explanations for the spontaneous oscillation phenomenon.

In this paper, a computer simulation method will be presented, which can be used as both the transient and steady-state oscillator analysis and the design tool. It will also demonstrate, convincingly, the two distinct mechanisms for the spontaneous oscillation phenomenon. Actual experimental results of a particular example will be given to demonstrate the accuracy of the simulation. Using this method, most aspects of the oscillators can be readily investigated, including amplifier nonlinearities; automatic amplitude control; short-term frequency stability, due to circuit noises; intermediate and long-term frequency stabilities, due to temperature changes and component aging; respectively; dynamic frequency versus loop-gain variations during the transient phase; and so on.

evaluating the convolution summation directly now takes only less than 1 min to completely simulate the response. Simulations can now be performed even on a personal computer without taking an excessively long time. In the next section, specific examples will be given, which should further illustrate the idea of using recursive digital filters in place of the direct evaluation of convolution summation.

We can now consider the case of hard-start oscillators that are initiated by the presence of a strong transient at the starting moment, which is usually due to the abrupt switching on of the power supplies, and which is applicable to the starting of most sinusoidal oscillators in practice. This starting transient can be represented by a short series of high-amplitude samples. If we assume a rectangular transient pulse, then Eq. (3) can be replaced by

$$x_i(t) = \sum_{j=0}^{N_s} N_s \cdot d(t - t_j) \quad (14)$$

where N_s is an integer representing the number of transient samples used in the simulation, and N_s represents the magnitude of the transient. In practice, the integer N_s of less than 10 gives adequate simulation accuracy. The transient magnitude N_s should be given a value representing a realistic amplitude of the starting transient.

Wein-Network Oscillator Simulation

In this section, we will consider an actual oscillator example, using the well-known Wein circuit as the positive feedback circuit. Figure 2a shows the Wein-network sinusoidal oscillator used in the simulation. The voltage gain of the amplifier is controlled by varying the FET resistance in the gain-setting negative-feedback resistor potentiometer. The controlling voltage, V_c , as applied to the gate of a F-channel FET, is the output of the amplitude converter, which, in this paper, will be either an absolute peak-detector circuit or an average value detector. The first method is easy to implement computationally but difficult to realize in the actual test circuit, whereas the reverse is true in the second method. A simple RC low-pass filter, as the averaging element, can be easily implemented in the test circuit; but, computationally, it incurs using the evaluation of a convolution summation, which can be very time-consuming. Fortunately, we can again use the recursive digital filter structure with a single delay element (as explained in the previous section), which will allow efficient computing of the average values. Both the peak-detector method and the average value detector method will be considered here, since they illustrate different aspects of a sinusoidal oscillator.

Figure 2a shows a typical relationship between the amplifier gain and the controlling voltage V_c , which has been experimentally measured by using half of the LF353 operational amplifier and the 2N5462 P-channel FET with the other gain-setting resistor

The actual program has been implemented in Turbo Pascal and can be run efficiently on both 8-bit and 16-bit personal computers. However, all our simulations have been done on IBM PC compatible computers. The average time for each complete simulation is usually less than 1 min.

Simulation Method

We will first consider the simulation of soft-start oscillators that are initiated by the presence of wide-band random electrical noises in the circuit (which will be represented by a small magnitude of random numbers), with a realistic signal-to-noise ratio that can be estimated from each actual test circuit. The discrete time-domain approach, with uniform time intervals of t_{sm} will be used throughout. Therefore, each instant of the discrete time can be written as t_j , where

$$t_j = j \cdot t_{sm} \quad \text{and} \quad j = 0, 1, 2, \dots \quad (1)$$

Figure 1 shows the basic block diagram of a sinusoidal oscillator. We will restrict our simulation in this paper, without loss of too much generality, to cases in which the amplifier is wideband, that is, it will not introduce any phase shift to the signal. The gain magnitude, however, is nonlinear and is dependent upon the output of the amplitude converter unit. Therefore, we can write

$$\text{Gain} = A = A(V_c) \quad (2)$$

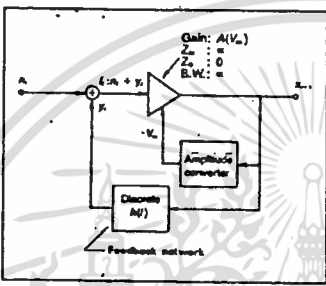


Fig. 1 The basic mathematical block diagram of the simulation setup

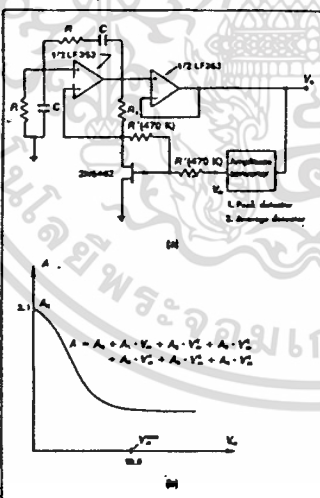


Fig. 2 (a) Wein-network sinusoidal oscillator circuit used in the simulation. (b) Gain versus controlling voltage characteristics of the amplifier.

equaling 1.2 kΩ. It can be readily seen from the figure that the gain decreases monotonically for $V_c < 10$; then it becomes constant for higher V_c . Various gain settings can be employed in the test oscillator by varying the resistor R_c , and this gain-amplitude relationship can be implemented into the simulation by employing the polynomial least-square curve fitting method [4]; thus, the voltage gain expression can be written as

$$A = A_0 + A_1 \cdot V_c + A_2 \cdot V_c^2 + A_3 \cdot V_c^3 + A_4 \cdot V_c^4 + A_5 \cdot V_c^5 + A_6 \cdot V_c^6 \quad (15)$$

where A_0 to A_6 must be calculated after each new setting of the amplifier gain has been measured. A_0 can also be considered the gain of the amplifier when the oscillation amplitude is very small, which must be greater than the attenuation of the feedback network for oscillation to start.

From Fig. 2a, it can be routinely shown that the ratio of the feedback voltage to the output voltage of

where V_m denotes the "amplitude" of the oscillating signal, to be defined more precisely later on. Usually, as V_c increases, $A(V_c)$ decreases to an asymptotic value. In this paper, we will limit our discussions to two types of amplitude conversion functions, representing a peak amplitude detector and an average amplitude detector, respectively. Therefore, V_m equals the absolute peak amplitude of each successive half-cycle in the first case and equals the average amplitude in the second case. These two cases will always be clear when we discuss specific examples in later sections.

The sequence of random numbers of appropriate magnitude as specified previously, can be written as

$$n_i = n(t) = \sum_{j=0}^M n(t_j) \cdot d(t - t_j) \quad (3)$$

where $d(t - t_j)$ denotes the usual impulse function [3].

The feedback network is assumed to be linear time invariant, having $M(t)$ as the impulse response function, which will also be discretized with equal time intervals of t_{sm} . In the case of direct evaluation of the response, $M(t)$ must be truncated when its absolute value remains smaller than an arbitrarily given minimum value, k_{sm} . Therefore, we can write

$$h(t) = \sum_{j=0}^M h(t_j) \cdot d(t - t_j) \quad (4)$$

where $d(t - t_j)$ is again the impulse function (and M exists such that, for all positive integers, j is greater than M) we always have

$$h(t_j) \leq k_{sm} \quad (5)$$

Let the output of the feedback network at instant t , be written as $y_i = y(t_i)$; then we can write the expression for the input to the amplifier as

$$x_i = n_i + y_i \quad (6)$$

Therefore, the amplifier output at the next instant of the discrete time, t_{i+1} , can be written as

$$x_{i+1} = A(V_c) \cdot x_i \quad (7)$$

where for all negative values of i , x_{-1} equals zero. In particular, for $i = -1$, $x_{-1} = x_0 = 0$ and for $i = 0$, $x_{-1} = x_i = A(0) \cdot (n_0 + y_0) = A(0) \cdot n_0$.

The output of the feedback network can be written as the results of the convolution between the previous inputs and the truncated impulse response function of the network, which can be written in discrete time-domain form, as in [3]

$$y(t) = t_{sm} \cdot \sum_{j=0}^M h(t_j) \cdot x(t - t_j) \quad (8)$$

where for all $j \geq i$, $x(t - t_j) = 0$. In particular, when $i = 0$, $y(t) = 0$, as expected from previous discussions. Furthermore, due to the truncation of the impulse response function according to Eq. (4), the

SEPTEMBER 1988

the Wein network can be written in terms of the Laplace transform as follows.

$$H(s) = V_d(s)/V_i(s) = (s/T)/(s^2 + (3/T)s + 1/T^2) \quad (16)$$

where the time constant T equals RC , as usual.

From Eq. (16), it can be shown that the inverse Laplace transform or the time-domain impulse response function of the Wein network [5] can be written as

$$h(t) = h_1(t) + h_2(t) \quad (17)$$

where

$$h_1(t) = (a'/T) \cdot \exp(-a'/T) \quad (18)$$

and

$$h_2(t) = -(b'/T) \cdot \exp(-b'/T) \quad (19)$$

where constants a , b , a' , and b' are equal to 2.6180, 0.3820, 1.1708, and 0.1708, respectively, and T equals RC , as in Eq. (16).

Using similar reasoning, it can be readily shown that the Laplace transforms of the discretized expressions for $h_1(t)$ and $h_2(t)$ can be written as

$$H_1(s) = (a'/T) \cdot (1 - \exp(-a'/T + s) \cdot t_{sm}) \quad (20)$$

$$H_2(s) = -(b'/T) \cdot (1 - \exp(-b'/T + s) \cdot t_{sm}) \quad (21)$$

The above two equations can be realized in practice as the sum of the output of two recursive digital filters with a single delay element in each filter, as shown in Fig. 3. The multiplicative constants a'/T

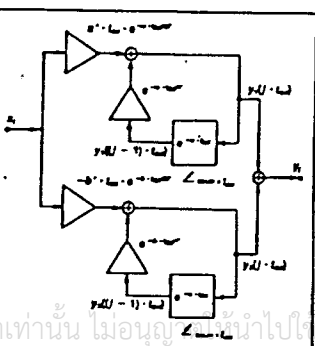


Fig. 3 The recursive digital filter structure to replace convolution integral calculation for the response of the Wein network.

stopped at $j = M$.

The formulation up to now is general for all of linear time-invariant feedback networks. However, the calculation for the discrete convolution summation in Eq. (8) for each discrete output $y(t_i)$ is time-consuming. This is because, for acceptable rates, the truncation limit M is generally quite large (of the order of 1000), and for each new value of convolution summation must be completely recalculated, resulting in almost unacceptable simulation time. For example, in simulating a network oscillator on an IBM PC compatible computer with an 8087 coprocessor, the required computational time is approximately 45 min.

Fortunately, for a limited class of feedback networks, where the impulse response functions can be written as linear summations of exponential functions, recursive digital-filter techniques can be used instead of directly evaluating the convolution summation, resulting in a much shorter calculation time. Many practically useful feedback networks possess such properties. To establish this idea on a discrete basis, we will consider the class of feedback networks in which the impulse response function can be written in the form

$$h(t) = \sum_{j=0}^M h_j(t) \quad (9)$$

where each individual $h_j(t)$ is an exponential function of the form

$$h_j(t) = H_j \cdot \exp(-a_j \cdot t) \quad (10)$$

where t represents the total number of $h_j(t)$ using Eqs. (10) and (11), we have

$$h(t) = \sum_{j=0}^M h_j(t) \cdot d(t - t_j) \quad (11)$$

where $d(t - t_j)$, again, denotes the impulse function. Using Eqs. (10) and (11), we have

$$h(t) = \sum_{j=0}^M H_j \cdot \exp(-a_j \cdot t) \cdot d(t - t_j) \quad (12)$$

It can be further shown that the Laplace transform of the above infinite summation sequence [3] can be written as

$$H(s) = H_0 / (1 - \exp(-a_0 + s) \cdot t_{sm})$$

It is now clear that each $H_j(t)$ can be readily simulated in the time domain as a recursive digital filter with a single delay element [3]. The total response now the linear summation of each $H_j(t)$, which is equivalent to summing the outputs of the recursive digital filters. Most practical feedback network responses consist of a few $H_j(t)$ terms only. There the calculation time is drastically reduced. The first type of oscillator that takes 45 min to simulate

and b'/T come directly from Eqs. (20) and (21) and a multiplicative constant, t_{sm} , in the Laplace transform $H(s)$ is required so that the summed output from the two filters will have same dimension when actually performing the convolution between the input pulse train and the impulse response function, as in Eq. (8), with extending to infinity. The two multiplicative constant terms, $\exp(-a \cdot t_{sm}/2 \cdot T)$ and $\exp(-b \cdot t_{sm}/2 \cdot T)$, are optional, and are used to improve the numerical accuracy without having to use the extremely small time interval t_{sm} . With these two constants, rather than use the values at the extreme end of each time interval, t_{sm} are used for recursive feedback.

Next, the discrete time interval, t_{sm} , must be determined. We know that the steady-state sinusoidal oscillation frequencies in the Wein-network oscillators are given by

$$\omega_0 = 1/T = 2 \cdot \pi \cdot f_0 = 2 \cdot \pi / T_0$$

where ω_0 , f_0 , and T_0 are the angular frequencies—the frequency and the period of the output of a Wein-network oscillator. Therefore, we can let N be number of time intervals, t_{sm} that T_0 can be divided, that is, we can write

$$N = T_0/t_{sm}$$

So, upon substituting Eq. (22) into (23), we have

$$t_{sm}/T = 2 \cdot \pi / N$$

Thus, all the t_{sm}/T ratios in Fig. 3 can be replaced by the term $2 \cdot \pi / N$, which can be thought of as normalized period, to be specified for each particular case of simulation. In most of our simulations of the order of 200–300 has been used.

Experimental Verifications

In this section, we will discuss the experimental verification of the Wein-network oscillator simulation. In order for the oscillation output to be easily recorded on a strip-chart recorder, very low oscillation frequencies have been selected, of the order of 1 Hz. This does not reduce the simulation's general applicability to higher oscillation frequencies, since the simulated outputs are normalized dimensionless quantities.

The general circuit of Fig. 2 was used in our experiments, with two types of amplitude converter unit. The first type of amplitude converter is an absolute peak-detector circuit, which, for the sake of convenience and flexibility, has been implemented in digital form, as shown in Fig. 4. The oscillator output is full-wave rectified, using the standard precision two-operational amplifier circuit [6]. The output is then into a sample-and-hold circuit and is converted into an 8-bit digital signal, which is further fed into a Z-80-based single-board computer. The residue

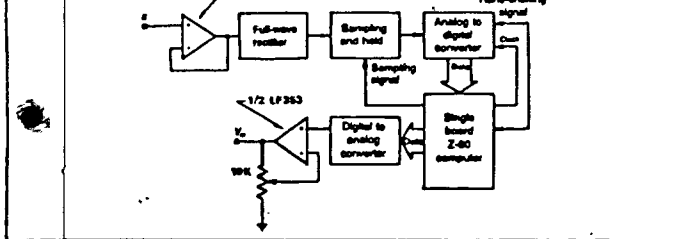


Fig. 4 Absolute peak-amplitude detector with software-controlled soft- and hard-start characteristics

software of this single-board computer will identify each successive peak value, which is then converted back into analog form and is buffered by a gain-adjustable amplifier. This becomes the signal that will control the FET resistance in the gain-setting network, as shown in Fig. 2a. The sampling and the conversion rates are controlled by the single-board computer. Due to the very low frequency of the converted input signal, we did not encounter any speed or other related problems.

The software can also be switched into either the hard-start or the soft-start modes. In the first mode, the controlling signal to the FET circuit is set to zero, giving maximum gain. At the same instant, a pulse of controlled amplitude and duration is injected into the circuit, which are, naturally, not accurately controllable or repeatable. The single-board computer has also been used to control the amplitude and duration of this starting pulse when an 8-bit digital-to-analog converter and a summing operational amplifier have been employed as the interface circuit between the single-board computer and the test circuit. A distinct advantage of using this method is that other shapes of transient pulses can be conveniently injected into the test oscillator.

However, in the second mode (the soft-start mode), the amplifier's starting gain is set to unity, with the application of the initial high-level controlling signal set to the FET circuit; the amplifier gain is gradually increased so that oscillations result from the building up of circuit noises rather than the initial transients, resulting in some dead time before the built-up oscillations can be observed. The oscillation amplitude's rate of growth will be dependent upon the gain setting of the amplifier.

The second type of amplitude converter is an average amplitude-detector circuit (in which the oscillation output is also full-wave rectified) that uses the precision two-operational amplifier circuit, as in

the first case; a first-order RC low-pass filter is used as the signal averager. The averaging RC time constant will now have a direct influence on the growth characteristic of the oscillations, as will be clearly shown in the recorded results of the experiment and the simulated results.

Figure 5 shows the comparisons between the actual recorded experimental waveforms and the corresponding simulated results for several circuit parameters, namely, the peak detector with the hard-start condition and the average detector with both soft-start and hard-start conditions, respectively. Large overshoots of the transient oscillation amplitudes, due to excessive initial gain settings, can be clearly seen in Figs. 5b and 5c. With the initial gain reduced to slightly greater than 3, no overshoot is apparent; however, the oscillations take many more cycles to build up to the steady-state magnitude. It should be noted that the apparent discrepancies between the magnitudes of the simulated results and the recorded results of the experiment are due entirely to the gain settings of the plotter and recorder. In actual situations, the numerical values of the simulated waveform magnitude correspond very closely to the measured waveform magnitude in volts units. The only computational parameters that have been arbitrarily adjusted to obtain the best fit between the actual and simulated results are the magnitudes of random numbers and the initial transient pulse, for the corresponding cases of soft start and hard start, respectively. However, the observed circuit-noise magnitude and the initial switching-transient amplitude are still of the same order of magnitude as the adjusted values used in the simulations. It is evident from Fig. 5 that the simulation method gives accurate predictions of sinusoidal oscillator behaviors, at least in the case of Wein-network oscillators.

From the results of Fig. 5, it is seen that the peak-detector method with the hard-start condition is very

other sequences will give rise to the mirror image oscillations can grow and settle down to steady-state amplitude in less than two cycles, with no apparent overshoot.

In the simulated soft-start cases, sometimes we obtain mirror images of the growth waveforms. This is due to the different characteristics of the random

other sequences will give rise to the mirror image oscillations can grow and settle down to steady-state amplitude in less than two cycles, with no apparent overshoot. In the simulated soft-start cases, sometimes we obtain mirror images of the growth waveforms. This is due to the different characteristics of the random

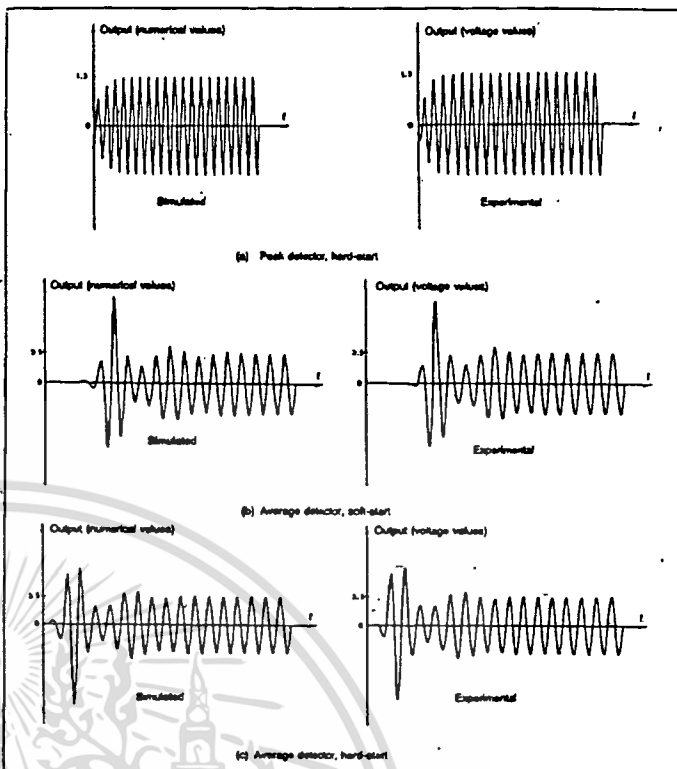


Fig. 5 Comparison between the simulated and the experimentally measured results for various cases of circuit parameters

IEEE CIRCUITS AND DEVICES MAGAZINE

SEPTEMBER 1986

oscillations in real circuits resulting from wide-band noises, as well as switching transients.

Discussions and Conclusions

In this paper, we have shown a computationally efficient method for accurately simulating sinusoidal oscillators. Two distinct modes of oscillation starting have been discussed, namely, hard start and soft start. These two modes were highlighted because of the success in simulating the initial transient and resonant circuit noises, respectively, by using the numerical transient pulse and random numbers as inputs to the simulated oscillator models. Accurate results have been obtained by using realistic magnitudes of the input transient pulse and random numbers. Most oscillator characteristics can be simulated by using this present model.

Although the only comparisons between the simulated and the measured results discussed in this paper are based on the well-known Wein-network oscillators, we have been able to show that similar and accurate simulations also hold for the LC sinusoidal oscillators. Therefore, we are confident that the method discussed in this paper will be applicable to most types of practical sinusoidal oscillators.

It is expected that the present method will also be useful in analyzing other oscillator properties; for example, the predictions of the short-term frequency stability due to circuit noises, the intermediate and the long-term frequency stabilities due to temperature variations and component aging, respectively.

Other solutions of sinusoidal oscillator problems, which have not been easily solvable up to now, can also be expected to benefit from the present method, for example, the solutions to a-coupled oscillator problems, where $n \geq 2$; or the solutions to surface acoustic-wave delay line-controlled oscillator problems with interdigital transducers. These two topics of investigation are being actively pursued by us at the moment.

In all our simulations, the random numbers used have been generated directly from the built-in routine of the Turbo-Pascal compiler. No attempt has been made, as far as study in detail the influences the random-number characteristics had upon simulation accuracy. Numbers simulating other types of noises (for example, shot noise) have not been investigated either. However, these investigations can be carried out as relatively straightforward extensions of the present method.

Finally, simulation offers excellent educational opportunities to students of electronic circuits. The reasons for the phenomenon of spontaneous oscillation starting, cannot be easily demonstrated in practice. Seeing the evidence that sinusoidal oscillations can be started either from random numbers or from a numerical starting impulse will certainly reinforce the understanding of the actual causes of spontaneous

References

- [1] N. Minors, "Nonlinear Oscillations," Van Nostrand Reinhold, 1962.
- [2] St. Stankovic, "Nonlinear Oscillations in Radio Engineering," English translation, Mir Publishers, 1976.
- [3] A. Papoulis, "Circuits and Systems—A Modern Approach," Holt, Rinehart, and Winston, 1981.
- [4] J. Hollman, "Science and Engineering Programs," Oxford University Press, 1981.
- [5] M. E. Spiegel, "Schwarz's Outline of Theory and Problems of Laplace Transforms," McGraw-Hill Book Company, 1965.
- [6] J. G. Grainger, "Applications of Operational Amplifiers," McGraw-Hill Book Company, 1972.



Satchai Pankajaporn

Supak Jantamongkorn

Satchai Pankajaporn was born in Bangkok, Thailand, on November 18, 1948. In 1966-1967, he studied in the Bel Air Senior High School, Bel Air, Maryland, as an American Field Service Exchange Student. From 1968-1972, he was a recipient of the Calverton Post Scholarship from the Australian government. He received the B.E.E. degree in electrical engineering with First Class Honors in 1973 from the University of New South Wales, Kensington, Australia. From 1973-1975, he received the Dean's Scholarship from the Faculty of Engineering, University of New South Wales. He received his Ph.D. in 1975 from that university.

He joined the Faculty of Engineering at King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand, in 1975, where he was the Director of Graduate Studies from 1977-1979, the Head of the Electronics Department from 1979-1981, and the Associate Dean for Administration from 1981-1982. He has been elected and appointed the Dean of Engineering since 1983.

Dr. Pankajaporn is a Consultant to Bender Field Engineering, Columbia, Maryland, and a Board Member of the Communications Authority of Thailand. He is also a member of several government committees and subcommittees.

Supak Jantamongkorn was born on March 21, 1963. He received the B.E.E. degree in Electronics from the King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand. He is presently enrolled in a Ph.D. student in the Faculty of Engineering.

IEEE CIRCUITS AND DEVICES MAGAZINE

ภาคผนวก ข. Simulations for the Transient Properties of
Multi-mode Delay-line Controlled Oscillators.



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Simulations for the transient properties of multi-mode delay-line controlled oscillators

SITTHICHAI POOKKAIYUDOM and SUJATE JANTARANG†

This paper describes a discrete time domain numerical simulation method for multi-mode delay-line controlled oscillators, which can yield solutions to both transient and steady state properties. A new efficient algorithm for numerical simulations of non-dispersive delay-lines is given. Experimental results supporting the validity of the theory are also presented.

1. Introduction

The importance of delay-line controlled oscillators (DLOs) as stable multi-frequency signal sources are well accepted (Davies and Lawrence 1971, Underhill 1972, Crabh, Lewis and Maines 1973). This is particularly so in the cases of DLOs using surface acoustic wave (SAW) delay-lines, operating in the higher frequency range than can be easily obtainable from conventional crystal controlled oscillators (Lewis 1974). Presently, the steady state properties of DLOs seem to be adequately developed, and practical devices are emerging from laboratories into operational systems.

One of the important attractive properties of DLOs is the ability to operate in more than one mode, which can be very useful in frequency agile communication and radar systems (Orant *et al.* 1977, Lewis 1985), and even in consumer product applications, for example, in television tuners (Hartmann 1975). In this mode of operation, the frequencies of oscillations must be able to jump from one mode to another mode upon appropriate application of a command signal. Therefore, it seems that in these applications, it is important to understand the transient characteristics of DLOs, for example, the dynamics of the starting and the ending of various modes of oscillations, and the time required to switch from one mode to another mode.

It is the purpose of this paper to discuss a numerical discrete time domain simulation method, which can be employed to study both the transient and the steady state properties of DLOs, for example, mode distributions, mode switching dynamics, mode switching times and so on. However, only the transient properties will be emphasized in this paper. Experimental results confirming the validity of the theory are also presented, using an electronically variable delay-line oscillator.

2. Theory

2.1. Assumptions

Figure 1 shows the basic block diagram for simulating the multi-mode DLO, where it is seen that the delay-line possesses a single transmitting transducer (TX).

Revision received 14 January 1989; accepted 9 March 1989.

† Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang, Ladkrabang, Bangkok 10520.

ISSN 0020-7179/89 \$3.00 © 1989 Taylor & Francis Ltd.

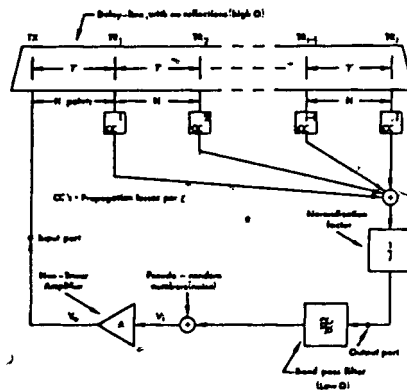


Figure 1. The basic block diagram of the simulation of a multi-mode DLO.

and up to J equally spaced receiving transducers (TR_j , where $j = 1$ to J) having their outputs summed and normalized at the output port, and T is the effective delay time between any two adjacent transducers. Such arrangement closely represents actual multi-mode DLOs (Lewis 1985). The mode switching is done by applying appropriate control signal to the bandpass filter. Restricting our discussion to only first order properties, the following assumptions can be made:

1. Each transducer possesses infinite bandwidth, any bandwidth restrictions in real transducers can be represented by external electrical equivalent circuits.
2. The delay-line is non-dispersive, and the propagation losses can be represented simply as a multiplicative constant at the output of each transducer.
3. No reflections of the propagating signals occur at either end of the delay-line.
4. The external amplifier possesses no phase-shift, but the gain magnitude is non-linear according to the hyperbolic tangent rule, which can be written as $A_0 \tanh(v_0)$, where v_0 and A_0 are the output and input signals respectively, and A_0 can be interpreted as the small signal gain of the amplifier.
5. The electrical noises in the oscillator can be represented by an equivalent noise source at the input of the amplifier which, in the present simulation, will be represented by an infinite sequence of pseudo-random numbers of appropriate magnitude. Such an approach has been shown previously to be adequate in the simulations of conventional RC oscillators (Pookkaiyudom and Jantarang 1986).

Multi-mode delay-line controlled oscillators

111

2.2. Fundamental properties

Using the above assumptions, the following fundamental properties of DLOs are relevant to the present simulation.

$$f_n = n/T \quad (1)$$

$$\text{del}(f_n) = f_{n+1} - f_n = f_n - f_{n-1} = 1/T \quad (2)$$

$$\text{del}(f_n) = f_{n+1} - f_{n-1} = 2/T \quad (3)$$

where f_n is the frequency of oscillation of the n th mode and n is the number of integral wavelengths per delay T between any two adjacent transducers as can be seen from Fig. 1, $\text{del}(f_n)$ is the frequency difference between any two adjacent modes, $\text{del}(f_n)$ is the frequency difference between the adjacent upper and lower modes. Therefore, the effective Q (Q_{eff}) of the delay-line can be increased, for improved short-term frequency stability by increasing the number of TR s, without affecting the mode spacing and the mode frequencies, which are controlled by the delay T (Lewis 1985). It can also be readily shown that for a non-dispersive delay-line having a single TX and J equally-spaced TR s, having n wavelengths per delay T , the Q_{eff} can be written as

$$Q_{\text{eff}} = (J/T) \sum_{n=1}^J n \quad (4)$$

where usually Q_{eff} is much greater than the mode selection bandpass filter Q .

2.3. Delay-line simulation algorithm

A non-dispersive delay-line can be simulated by a series of equally spaced JN discrete points, having the first point at TX and the JN th point at TR_J , where J runs from 1 to J , and JN is the number of TR s. These discrete points will be represented by memory locations containing the sampled values of the signal along the delay-line, and the propagation of the signal can then be simulated by successively moving these sampled numerical values along the chain of memory locations at each given time interval.

If we let $C(k)$ be the contents of the memory location representing the k th point, C_{new} be the latest sampled value of the signal to be fed into the delay-line, then for each given time interval, we can write the simulation process symbolically as:

$$\text{Output} = (1/J) \sum_{j=1}^J C(JN) \quad (5)$$

$$C(k) = C(k-1) \quad (6)$$

where $k = JN, JN-1, JN-2, \dots, 3$ and 2.

$$C(1) = C_{\text{new}} \quad (7)$$

where eqn. (5) is carried out before executing the sequence in eqn. (6), and eqn. (7) is carried out last.

Such simulation method is conceptually straightforward, but it is very inefficient computationally, since JN values must be shifted per time interval. A more efficient equivalent alternative is to let TR s move backward along the delay-line. As each TR

112

S. Pookkaiyudom and S. Jantarang

reaches the first point, it is wrapped around to the JN th point. The point immediately left behind by the last receiving transducer, TR_J , can be thought of as equivalent to the first point of the delay-line, and new values of the sampled input signal can be successively stored into the memory locations representing these points.

Let p denote the sequence of time intervals for each TR to move a complete loop around the delay-line. Therefore, p runs from 0, 1, 2, ..., JN repeatedly. During each sequence of p , we can write the new simulation algorithm symbolically as

$$\text{Output}(p) = (1/J) \sum_{j=1}^J C(JN - p) \quad (8)$$

$$C(JN - p) = C_{\text{new}} \quad (9)$$

where eqn. (8) is carried out before the execution of eqn. (9).

This new simulation algorithm reduces the number of operations in each time interval by JN times, resulting in a drastic reduction in computational time, especially when the number of JN is quite large, say, of the order of several thousands.

2.4. Digital bandpass filter

Referring again to Fig. 1, it is seen that an external band-pass filter is used to select the required mode of oscillation. It is further desired that the frequency stability of the DLO is governed by the Q_{eff} of the delay-line rather than the Q of the bandpass filter. Therefore, it is important to limit the Q of the bandpass filter to a value just enough for reliable mode switching. In the present paper, we will set the bandpass filter Q such that the resulting 3 dB bandwidth equals to $\text{del}(f_n)$ of eqn. (3). Thus, the required bandpass filter Q can be readily shown to be given by

$$Q = J/BW = (J/T)/\text{del}(f_n) = J/2 \quad (10)$$

where BW is the bandwidth of the bandpass filter, and k is now the integer part of the average between the highest and the lowest modes that the DLO is expected to jump.

From eqns. (5) and (10), it is seen that the average ratio of the Q_{eff} of the delay-line to the Q of the bandpass filter is equal to

$$Q_{\text{eff}}/Q = (J/T) \sum_{n=1}^J n \quad (11)$$

where it can be seen that, even when J is equal to unity, the DLO short term frequency stability is more dependent upon the delay-line rather than the mode selection filter.

We further assume a second order mode selection bandpass filter transfer function of the form

$$H(s) = A_0 s / (s^2 + (\omega_0/Q)s + \omega_0^2) \quad (12)$$

where A_0 can be set equal to ω_0/Q to achieve unity passband gain. Equation (12) can be readily transformed into the equivalent digital filter of Fig. 2, using the fol-

The output oscillation voltage values, a_n , can be converted to 8-bit digital data and stored for later analyses by using a specially constructed interface card connected to an IBM compatible personal computer. The maximum sampling rate of the interface card is 8 kHz, which is more than adequate for the range of frequencies our experiments, which are less than 1 kHz. Once the experimental data have been collected, further analyses in both the time and frequency domains can be readily achieved.

2. Experiments

In this section, the results from simulations are compared to the results from experimental measurements. Figures 4(a)-(c) show the time domain mode switching waveforms between the simulated and experimental cases, for delay-lines consisting of one, two and three TRs respectively, and the loop-gain has been set at 2.0 for all three cases. The 16th and the 20th modes have been chosen for switching in the experiments. Therefore, the average mode number in eqn. (10), or L , equals 18, resulting in a mode selection filter Q of 9. Furthermore, from eqn. (11), the average delay-line Q_{av} are 28.3, 42.4 and 56.4, corresponding to the cases of one, two and three delay-lines in Figs. 3, or equivalently one, two and three receiving transducers in Fig. 1, respectively.

Therefore, the short term frequency stabilities are more dependent upon the delay-lines than upon the mode selection filter. As expected, all three cases share the same lower (16th) and upper (20th) mode frequencies, equal 577.30 Hz and 721.60 Hz, respectively. It is clearly seen from Fig. 4 that the simulated results accurately represent the experimental results, for both cases of switching from the lower mode to the upper mode and vice-versa.

Figures 5(a) and (b) show the plots of the switching time from the 16th mode to a 20th mode versus the numbers of TRs and initial loop-gains respectively, where Fig. 5(a) the initial loop-gain has been fixed at 2.5 and in Fig. 5(b) the number of TRs has been fixed at 5. It is seen that the switching time is approximately linearly related to the numbers of TRs, where the slight non-linearity for the cases of low values of TRs is due to the effect of the mode selection filter Q . Furthermore, it is seen that, for the cases when the initial loop-gain are smaller than 2.5, the switching times also increase with decreasing loop-gain values. However, when the initial loop-gains are greater than 2.5, the resulting switching times are relatively independent of the actual loop-gain values.

In practical DLOs, the numbers of TRs can be expected to be in the range of 10-100 (Lewis 1985), where large numbers of TRs are needed to ensure adequate short-term frequency stability. However, as can be seen from Fig. 5(a), the maximum number of TRs is limited by the allowable maximum switching delays associated with each particular system, and Fig. 5(a) can be used as a rough design guideline.

Conclusions

In this paper, a method to simulate the transient characteristics of DLOs has been presented. Although the method is also useful in simulating the steady characteristics of these oscillators, the main attention in this paper is on the study of the transient mode switching properties, especially the time required to switch from one

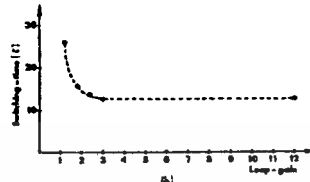
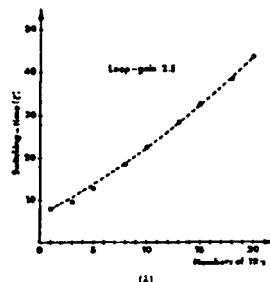


Figure 5. (a) Plot of the switching-time versus the numbers of TRs for a fixed initial loop-gain of 2.5. (b) Plot of the switching-times versus the initial loop-gain for the case of 5 TRs.

mode to the next, which should be important in the design of DLOs as the frequency sources for frequency agile communication and other systems.

Experimental results using electronic BBD delay-lines as the frequency controlling elements showed good agreement with the theory. Thus, the validity of the theory, at least in the cases when the actual experimental conditions follow closely the idealized conditions of §2.1, has been proven.

Other properties of DLOs can also be investigated by the present simulation package, for example, waveform growth and decay times with respect to the initial loop-gains and effective delay-times, and short term frequency stabilities with respect to delay-times and average noise amplitudes, and so on.

REFERENCES

- ALLAN, P. E. 1964, *Switched Capacitor Circuits* (London: Van Nostrand Reinhold Co.).
- CHAM, J., LEWIS, M. F., and MAPLE, J. D. 1973, Surface acoustic wave oscillators: mode selection and frequency modulation, *Electronics Letters*, **9**, 195-196.
- DAVIS, L. W., and LAWRENCE, M. W. 1971, Prospects for surface elastic wave crystal-controlled delay-line oscillators, *Proc. Inst. Radio & Electron. Eng. Aust.*, **1**, 61-62.
- GRANT, P. M., HUMPH, K. M., CHURCH, R. C., and COLLINS, J. H. 1977, Mobile radio frequency synthesizers based on surface acoustic wave oscillators, *Radio Electron. Eng.*, **47**, 208-216.
- HARTMAN, P. 1975, Programmable acoustic surface wave oscillator, *Electronics Letters*, **11**, 119-120.
- LEWIS, M. F. 1974, Oscillators—The next successful surface acoustic wave device? *Ultrasonics*, **11**, 115-121.
- LEWIS, M. F. 1983, Practical frequency source for use in agile radars, *Electronics Letters*, **21**, 1017-1018.
- POEKAIYANDOM, S., and JANTARATONG, S. 1986, Simulations for sinusoidal oscillators with random numbers and initial impulses, *I.E.E.E. Circuits and Devices Magazine*, **5**, 34-40.
- UNDERHILL, M. J. 1972, Delay-stabilized variable oscillator, *Electronics Letters*, **8**, 115-116.

ภาคผนวก ค. Efficient Algorithm for Simulating
Interdigital Transducer Delay-Lines



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$(c) \quad \left\{ 4 \sum_{i=1}^n (P_i + Q_i) \right\}$$

data transfers.

Thus, the resulting simulation speed can be quite slow, especially when the number of discrete points per IDT section is large and there are several sections in each IDT. However, several redundant mathematical steps can be eliminated from the basic algorithm, which follows directly from the following considerations.

We will consider, firstly, the RH propagating SAW, for the n th IDT section. Consider the point $(J_n + P_n - 1)$, which is the right-most point on the positive electrode. We can write the total surface displacement value at this point as

$$\{C_R(J_n + P_n - 1)(j)\} = T_{RH}\{C_L(J_{n-1} + 2P_{n-1} + 2Q_{n-1} - 1)(j - P_n) + \dots$$

$$\dots + R_{RH}\{C_L(J_n)(j - P_n)\} + (L_n/R_n) \sum_{i=0}^{(P_n-1)} I(j - i) \quad (6)$$

where $\{C_R(J_{n-1} + 2P_{n-1} + 2Q_{n-1} - 1)(j - P_n)\}$ is the total RH propagating displacement value of the last discrete point of the $(n-1)$ th section, which is delayed by P_n time intervals, and can be defined to be equal to zero when either $(n-1) < 0$ or $(j - P_n) < 0$; $\{C_L(J_n)(j - P_n)\}$ is the total LH propagating displacement value reflected from the LH edge of the electrode delayed by P_n time intervals, and can also be defined to equal zero when $(j - P_n) < 0$; $I(j - i)$ is the input signal at the time t_{j-i} , and can again be defined to equal zero when $(j - i) < 0$.

Let $S_n(j) = (L_n/R_n) \sum_{i=0}^{(P_n-1)} I(j - i)$, then from (6), we can write:

$$\{C_R(J_n + P_n - 1)(j)\} = T_{RH}\{C_R(J_{n-1} + 2P_{n-1} + 2Q_{n-1})(j - P_n) + \dots$$

$$\dots + R_{RH}\{C_L(J_n)(j - P_n)\} + S_n(j - i) + \dots$$

$$\dots + (L_n/R_n) \{I(j) - I(j - P_n)\} \quad (7)$$

where (7) will be used to evaluate the total RH propagating SAW displacement value at the right-most point of the positive electrode of the n th IDT section.

Next, we consider the discrete point at the right-most edge of the inter-electrode region between the positive and the negative electrodes, where the total RH propagating surface displacement value can be written as

$$\{C_R(J_n + P_n + Q_n - 1)(j)\} = T_{RH}\{C_R(J_n + P_n - 1)(j - Q_n) + \dots$$

$$\dots + R_{RH}\{C_L(J_n + P_n)(j - Q_n)\} \quad (8)$$

where the meaning of each term on the RHS of (8) follows the same pattern as in (7), and all the terms with negative time interval indices can be set to equal zero.

At the right-most point of the negative electrode, we can write the RH propagating total surface displacement values as follows:

$$\{C_R(J_n + 2P_n + Q_n - 1)(j)\} = T_{RH}\{C_R(J_n + P_n + Q_n - 1)(j - P_n) + \dots$$

$$\dots + R_{RH}\{C_L(J_n + P_n + Q_n)(j - P_n)\} - \dots$$

$$- \{S_n(j - 1) + (L_n/R_n) \{I(j) - I(j - P_n)\}\} \quad (9)$$

where again the meaning and the convention of each term on the RHS of (9) follows the same pattern as in (7).

Lastly, we consider the right-most point of the n th IDT section, where we can again write the RH propagating total displacement of the SAW at the point

then be regarded as the normalized value of the sum of all the discrete points in the finger. At the edges of the inter-electrode and the metallic finger regions, part of the propagating SAW is reflected while the rest is transmitted across the interface.

Let K_{RH} be the mechanical wave displacement signal to the electrical signal conversion factor, where K_{RH} should again include the effects of the material electro-mechanical coupling factor and all other losses associated with the receiving IDT.

Let $O_{RH}(j)$ be the output signal of the receiving IDT at time t_j . It can then be shown that

$$O_{RH}(j) = K_{RH} \left[\sum_{i=1}^n \{L_i/R_i \{F^+(n, j) - F^-(n, j)\}\} \right] \quad (15)$$

where $F^+(n, j)$ is the sum of the signals from all the discrete points of the n th section positive electrode, and is given by

$$F^+(n, j) = \sum_{i=0}^{(P_n-1)} \{C_R(J_n + i)(j) + C_L(J_n + i)(j)\} \quad (16)$$

and $F^-(n, j)$ is the sum of the signals from all the discrete points of the corresponding negative electrode, and is given by

$$F^-(n, j) = \sum_{i=0}^{(2P_n+Q_n-1)} \{C_R(J_n + i)(j) + C_L(J_n + i)(j)\} \quad (17)$$

Using (15) together with (15)–(17), the output of the receiving IDT can then be calculated. However, the numbers of mathematical manipulations per clock cycle is equal to

$$8N - 4 \sum_{i=1}^n P_i$$

summations, $17N - 1$ multiplications and

$$4 \sum_{i=1}^n (P_i + Q_i)$$

data transfers, the same as in the case of direct transmitting IDT simulation, resulting in a rather inefficient use of computation time.

The algorithm can again be re-formulated to eliminate several redundant steps, by considering firstly the RH propagating signal, where for the n th section electrode it can be shown that at the right-most points in the positive electrode, the inter-electrode region between the two electrodes, the negative electrode and the right-most inter-electrode region, can be written respectively, as

$$\{C_R(J_n + P_n - 1)(j)\} = T_{RH}\{C_R(J_{n-1} + 2P_{n-1} + 2Q_{n-1} - 1)(j - P_n) + \dots$$

$$\dots + R_{RH}\{C_L(J_n)(j - P_n)\} \quad (18)$$

$$\{C_R(J_n + P_n + Q_n - 1)(j)\} = T_{RH}\{C_R(J_n + P_n - 1)(j - Q_n) + \dots$$

$$\dots + R_{RH}\{C_L(J_n + P_n)(j - Q_n)\} \quad (19)$$

$$\{C_R(J_n + 2P_n + Q_n - 1)(j)\} = T_{RH}\{C_R(J_n + P_n + Q_n - 1)(j - P_n) + \dots$$

$$\dots + R_{RH}\{C_L(J_n + P_n + Q_n)(j - P_n)\} - \dots$$

$$- \{S_n(j - 1) + (L_n/R_n) \{I(j) - I(j - P_n)\}\} \quad (20)$$

$$\{C_R(J_n + 2P_n + Q_n - 1)(j)\} = T_{RH}\{C_R(J_n + 2P_n + Q_n - 1)(j - P_n) + \dots$$

$$\dots + R_{RH}\{C_L(J_n + 2P_n + Q_n)(j - P_n)\} \quad (21)$$

$(J_n + 2P_n + 2Q_n - 1)$ as

$$\{C_R(J_n + 2P_n + 2Q_n - 1)(j)\} = T_{RH}\{C_R(J_n + 2P_n + Q_n - 1)(j - Q_n) + \dots$$

$$\dots + R_{RH}\{C_L(J_n + P_n + Q_n)(j - Q_n)\} \quad (10)$$

where we can again follow the conventions and the meanings of (7) for the RHS terms of (10).

Next, we consider the case of a LH propagating SAW, where, following similar conventions as in (7) to (10), we can write the following equations for the discrete points $(J_n + 2P_n + Q_n)$, $(J_n + P_n + Q_n)$, $(J_n + P_n)$ and (J_n) , respectively, as

$$\{C_L(J_n + 2P_n + Q_n)(j)\} = T_{RH}\{C_L(J_n + 1)(j - Q_n) + \dots$$

$$\dots + R_{RH}\{C_R(J_n + 2P_n + 2Q_n - 1)(j - Q_n)\} \quad (11)$$

$$\{C_L(J_n + P_n + Q_n)(j)\} = T_{RH}\{C_L(J_n + 2P_n + Q_n)(j - P_n) + \dots$$

$$\dots + R_{RH}\{C_R(J_n + 2P_n + Q_n - 1)(j - P_n) - \dots$$

$$- \{S_n(j - 1) + (L_n/R_n) \{I(j) - I(j - P_n)\}\} \quad (12)$$

$$\{C_L(J_n + P_n)(j)\} = T_{RH}\{C_L(J_n + P_n + Q_n)(j - Q_n) + \dots$$

$$\dots + R_{RH}\{C_R(J_n + P_n + Q_n - 1)(j - Q_n)\} \quad (13)$$

$$\{C_L(J_n)(j)\} = T_{RH}\{C_L(J_n + P_n)(j - P_n) + \dots$$

$$\dots + R_{RH}\{C_R(J_n + P_n - 1)(j - P_n) + \dots$$

$$\dots + S_n(j - 1) + (L_n/R_n) \{I(j) - I(j - P_n)\} \quad (14)$$

From the above consideration, it can be seen that in simulating the transmitting IDT responses, we need to evaluate the numerical values of (7) to (14) successively for n equal to 1 to N . The output signal from the transmitting IDT is then the content of the point $(J_N + 2P_N + 2Q_N)$ at the time t_{J_N+1} , which is the content of the discrete point $(J_N + 2P_N + 2Q_N - 1)$ at the time t_j .

Using the reformulated algorithm, it can be seen that the required mathematical steps reduce drastically to only $16N$ summations and $17N + 1$ multiplications per clock cycle. Of the $16N$ summations, $8N$ summations are due to the summing of the incident displacement values to the reflected values at each of the interfaces in the IDT section, while another $4N$ summations are due to the input signals upon the positive and the negative electrodes. Of the $17N + 1$ multiplications, $16N$ multiplications are due to multiplications at the edges of both the positive and the negative electrodes, N multiplications are due to the distribution factor for each section and the remaining single multiplication represents the mechanical-to-electrical conversion process.

The number of mathematical manipulations will be further reduced to only $2N$ summations and $N + 1$ multiplications if the reflection and the transmission coefficients can be approximated to zero and unity, respectively.

5. Receiving IDT simulation algorithm

We can also use Fig. 1 to represent the n th section of a receiving IDT, having N total sections. We shall let the propagating SAW incident upon the receiving IDT at the left-most IDT section, which will be labelled 'section 1', and the right-most IDT section will be labelled section N . Each discrete point in the metallic finger can then be thought of as a data retrieval point and the output signal from each finger can

where we have again let the displacement content terms be zero when the time index terms are negative, and the first term on the RHS of (18) is simply the incident wave upon the IDT for the case when n equals unity.

Similarly, for the case of a LH propagating SAW, we can write the following equations associated with the left-most points in the positive electrode, the inter-electrode region between the two electrodes, the negative electrode and the right-most inter-electrode region, respectively, as

$$\{C_L(J_n)(j)\} = T_{RH}\{C_L(J_n + P_n)(j - P_n) + \dots$$

$$\dots + R_{RH}\{C_R(J_n + P_n - 1)(j - P_n)\} \quad (22)$$

$$\{C_L(J_n + P_n)(j)\} = T_{RH}\{C_L(J_n + P_n + Q_n)(j - Q_n) + \dots$$

$$\dots + T_{RH}\{C_R(J_n + P_n + Q_n - 1)(j - Q_n)\} \quad (23)$$

$$\{C_L(J_n + P_n + Q_n)(j)\} = T_{RH}\{C_L(J_n + 2P_n + Q_n)(j - P_n) + \dots$$

$$\dots + R_{RH}\{C_R(J_n + 2P_n + Q_n - 1)(j - P_n)\} \quad (24)$$

$$\{C_L(J_n + 2P_n + Q_n)(j)\} = T_{RH}\{C_L(J_n + 1)(j - Q_n) + \dots$$

$$\dots + R_{RH}\{C_R(J_n + P_n + Q_n - 1)(j - Q_n)\} \quad (25)$$

where we can let the first term the RHS of (25) be equal to the reflected SAW from the end of the delay-line when n equals N , and the convention concerning the time index terms still holds.

From consideration of the meanings of the quantities $F^+(n, j)$ and $F^-(n, j)$ in (16) and (17), as previously defined, it can be shown that

$$F^+(n, j + 1) = F^+(n, j) + (L_n/R_n) \{C_R(J_n)(j + 1) - \dots$$

$$\dots - \{C_R(J_n + P_n - 1)(j) + C_L(J_n + P_n - 1)(j + 1) - \dots$$

$$\dots - C_L(J_n)(j)\} \quad (26)$$

$$F^-(n, j + 1) = F^-(n, j) + (L_n/R_n) \{C_R(J_n + P_n + Q_n)(j + 1) - \dots$$

$$\dots - \{C_R(J_n + 2P_n + Q_n - 1)(j) + C_L(J_n + 2P_n + Q_n - 1)(j + 1) - \dots$$

$$\dots - C_L(J_n + P_n + Q_n)(j)\} \quad (27)$$

Therefore, using (15), (26) and (27), we can write

$$O_{RH}(j + 1) = K_{RH} \left[O_{RH}(j) + \sum_{i=1}^n \{L_i/R_i \{C_R(J_i)(j + 1) - \dots$$

$$\dots - \{C_R(J_i + P_n - 1)(j) + C_L(J_i + P_n - 1)(j + 1) - \dots$$

$$\dots - C_L(J_i)(j)\} - \dots$$

$$\dots - \{C_R(J_i + P_n + Q_n)(j + 1) - \{C_R(J_i + 2P_n + Q_n - 1)(j) - \dots$$

$$- C_L(J_i + 2P_n + Q_n - 1)(j + 1) - \{C_L(J_i + P_n + Q_n)(j)\}\} \right] \quad (28)$$

The simulation process now consists of evaluating (18) to (25) for $n = 1$ to N and the output from the receiving IDT can then be calculated from (28), where it can be seen that the number of mathematical operations per clock cycle consists of $16N$ summations and $17N + 1$ multiplications, exactly equal to the case of the transmitting IDT.

In cases where the inter-electrode reflections, and the reflections from the ends of the delay-line, can be ignored, the mathematical manipulation steps reduce to $2N$ summations and $N + 1$ multiplications.

6. Delay-line simulation algorithm

The propagation of the numerical values in the region between the transmitting and the receiving IDTs can also be efficiently simulated as a delay-line with an imaginary single-line transmitter and receiver, respectively. Such an algorithm for a one way propagation of SAW has been described previously by the authors (Pookaiyaudom and Jantarang 1989) and will not be repeated here. However, it should be pointed out that, in cases where triple-transit and delay-line end reflections effects need to be accurately simulated, we have to simulate both the forward and the backward propagating SAW, resulting in two data transfers per clock cycle, which is still very efficient computation-wise.

7. $V(r)/r$ effect

The effects of the slowing down of SAW in the metallized electrode regions can be readily incorporated into the algorithm by using the concept of effective finger widths. The finger widths should be increased in proportion to the ratio $V(r)/r$, where r is the SAW velocity in free surface regions and $V(r)$ is the decrease in the SAW velocity in the metallized regions.

8. Simulation examples

The simplified algorithm, with the assumptions that the $V(r)/r$ and the inter-electrode reflection effects are negligible, has been tested against the flat field approximation theory of Tseng (Tseng 1968), and it has been shown that the simulated results using the simplified algorithm agree well with the results calculated from the flat field approximation theory. The following cases have been specifically tested, namely the impulse and the continuous wave passband responses for the fundamental (or the synchronous) and the higher mode frequencies of uniformly distributed IDTs, the responses of IDTs when the ratios of the widths of the metallic fingers to the inter-electrode region widths are non-unity, the responses of IDTs with gaussian and other weighting (or apodization) functions. From these results, we can conclude that the simplified algorithm gives useful results in cases where only the first order properties of SAW delay-line are required.

Better approximations to real devices can be obtained by incorporating $V(r)/r$, as well as inter-electrode reflection effects into the algorithm, at the expense of more computation time. The correctness of this extended algorithm has been checked by simulating the responses of the reflected SAW pulses from several IDT patterns, using only a single numerical input pulse, where for each given IDT pattern, the expected impulse response of the reflected SAW can be readily derived by inspection.

In this section, we show some simulated examples of cases which are difficult to predict using purely analytical theories, especially in cases of dispersive IDT delay-

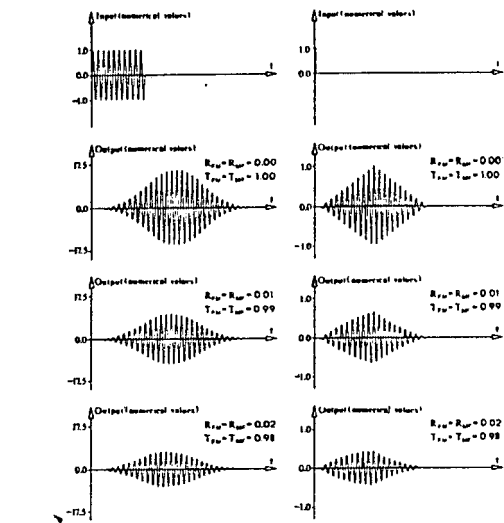


Figure 2. Simulated impulse responses of a delay-line with uniformly distributed 10-finger section transmitting and receiving IDTs.

lines and IDT responses when the inter-electrode reflection coefficients are significant, as they are in some strongly piezoelectric substrate materials.

Figure 2 shows the time domain responses of a delay-line having 10 transmitting and 10 receiving IDT sections, with uniform finger spacings. The LHS of the figure shows cases where the input is a gated sine wave at the synchronous frequency while on the RHS shows cases where the input is a narrow pulse of 20-clock-pulse width. Three cases have been simulated, with reflection coefficients of zero, 1% and 2%, respectively. It is seen that non-zero inter-electrode reflection coefficients result in reductions in the peak amplitudes of the impulse responses and distortions of the impulse response envelopes with respect to the ideal case.

Figure 3 shows the effects of cases where the finger width to inter-electrode spacing ratios are non-unity. It is interesting to note that the relative responses for the three cases of different ratios of the finger widths to the inter-electrode spacing are of the same order for the fundamental frequency and the fifth order frequency responses, while the response of the third order frequency is of the reverse order. Since the inter-electrode reflection coefficient has been assumed to be zero, it is not surprising that the simulated results follow closely the results obtained using the flat field approximation theory of Tseng (Tseng 1968).

Interdigital transducer delay-line simulation algorithm

121

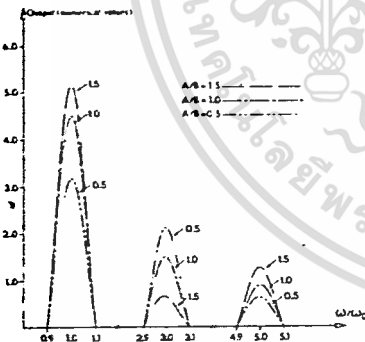


Figure 3. Frequency responses of the fundamental and higher mode frequencies, with IDTs having different ratios of the metallic finger widths to the inter-electrode spacing widths

Figure 4 shows comparisons of the frequency responses of the fundamental mode and the third order mode for cases where the inter-electrode coefficients are equal to zero and 5%, respectively. Again, it is seen that the effect of a non-zero inter-electrode reflection coefficient is to reduce the peak amplitude of the frequency

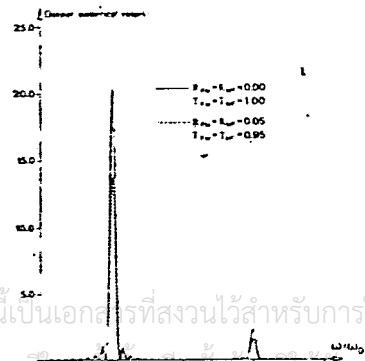


Figure 4. Effect upon the frequency response of non-zero inter-electrode reflection coefficients.

122

S. Pookaiyaudom and S. Jantarang

response. Both the finger spacings in the 10-section transmitting and receiving IDT are uniform.

Figure 5 shows the impulse response of a delay-line with a pair of identical dispersive IDTs, having 33 finger sections each. The ratio of the longest to the shortest finger section lengths is two, and the section length distribution is monotonically linear. The high frequency ends of the two IDTs are placed next to each other. The input signal is an impulse, having twenty clock-pulses duration. The two upper traces show the output from the transmitting and the receiving IDTs, respectively, for the case where the inter-electrode reflection coefficient is zero. The non-uniform output amplitude of the impulse response from the receiving IDT indicates that the frequency domain passband will also be non-constant, and the finger overlapping length weighting function should be employed to smooth out the passband response. The lower two traces show the case where the inter-electrode reflection coefficient is equal to 0.5%; it is seen that the impulse response is greatly distorted with respect to the previous case, and finger overlapping length weighting function will certainly be needed to obtain a useable passband response. These results clearly show the usefulness of the present algorithm in the design of dispersive IDT delay lines, where both the finger section length distributions and the finger overlapping

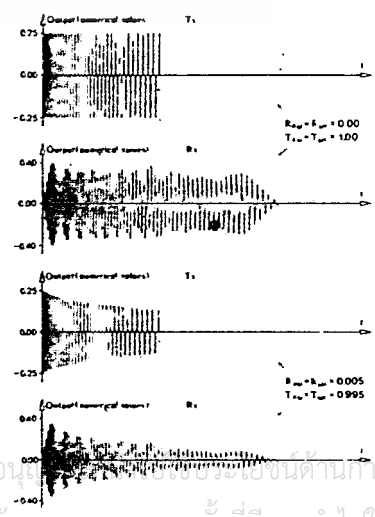


Figure 5. Impulse responses of dispersive delay-lines with 33-finger-section transmitting and receiving IDTs, with monotonical linear frequency distributions, for cases having inter-electrode reflection coefficients of zero and 0.5%, respectively

length functions can be tried out, until an optimum impulse response has been obtained, before the actual fabrication of the device.

9. Conclusions

In this paper, we have proposed a computation-efficient algorithm for simulating the responses of IDT delay-lines. Several properties of the IDTs can be incorporated into the algorithm, namely non-uniform finger distributions (dispersive IDTs), arbitrary finger overlapping length weighting (or apodization) functions, $V(t)/t$ effects of metallized surfaces, inter-electrode reflections and reflections from the ends of the delay-lines.

The proposed algorithm can readily be extended to simulate the time domain properties of SAW resonators. It should prove to be very useful in time domain simulations of electronic systems incorporating dispersive IDTs with arbitrary weighting functions and system non-linearities.

REFERENCES

- POOKAIYAUDOM, S., and JANTARANG, S., 1989, Simulations for the transient properties of multi-mode delay-line controlled oscillators *International Journal of Electronics*, **66**, 941-945.
- SMITH, W. R., GERARD, H. M., COLLINS, J. H., REIDER, T. M., and SHAW, H. J., 1969, Design of surface wave delay lines with interdigital transducers. *I.E.E.E. Transactions on Microwave Theory and Techniques*, **17**, 865-873.
- TSONG, C. C., 1968, Frequency response of an interdigital transducer for excitation of surface elastic waves. *I.E.E.E. Transactions on Electron Devices*, **15**, 586-594.



ภาคผนวก ง. การออกแบบระบบเปลี่ยนสัญญาณาลอกเป็นสัญญาดิจิทัลที่
อัตราสูง



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ผู้พิมพ์ รุ่งพงษ์

บทบาท อานันท์

ภาควิชาอิเล็กทรอนิกส์

ศตวรรษใหม่ไทยที่จะรวมกันแล้ว อีก ๒๖ ปี

บทคัดย่อ:

วงจรเปลี่ยนสัญญาณอนาล็อกเป็นดิจิทัล เป็นอุปกรณ์อย่างหนึ่งที่มีค่าของช่วงความถี่สัญญาณที่แปลงสัญญาณเชิงดิจิทัล เช่น คณิตศาสตร์ ในบทความนี้เราได้นำเอาหลักการของ Time-division multiplex มาใช้ในการออกแบบเพื่อให้ระบบแปลงสัญญาณอนาล็อกเป็นดิจิทัล มีอัตราการสุ่มที่สูงกว่า เวลาการแปลงสัญญาณของตัวแปลงสัญญาณอนาล็อกเป็นดิจิทัล

Abstract:

An approach to the bandwidth improvement of analog-to-digital converters is described. The presented technique offers a sampling rate higher than the conversion time of analog-to-digital converter system, since the input signal is time multiplexing via a parallel of N identical analog-to-digital converters.

1-5-3

ถ้าให้ $T \geq 1/(2.BW)$ (2)

นั่นคือ จะทำการสุ่มสัญญาณที่ต่อเนื่องกัน $2B$ ของสัญญาณ และ การสุ่มสัญญาณในลักษณะนี้ เมื่อทำการสุ่มเป็นไปตามสมการ (2)

ดังที่กล่าวมาแล้วว่า A/D ต้องใช้เวลาในการทำงานในการแปลงสัญญาณ ถ้าให้ T_c เป็นคาบ (period) ของสัญญาณนาฬิกาที่มีกับ A/D และให้ M เป็นจำนวนที่ใช้เกิดต้องใช้ในการแปลงสัญญาณ เพราะฉะนั้นเวลา A/D ต้องใช้ในการแปลงสัญญาณเป็น

$$T_{A/D} = M \cdot T_c \quad (3)$$

และ การแปลงสัญญาณต้องสร้างระบบที่จะทำการสุ่มสัญญาณกลับออกไป นั่นคือ

$$T_s > T_{A/D} \text{ หรือ } T_s > M \cdot T_c \quad (4)$$

นั่นคือว่า $T_{A/D}$ เป็นค่าที่คำนวณค่าหรือสัญญาณที่จะทำการสุ่ม โดยทั่วไป A/D ที่ราคาจะถูกจะใช้เวลาในการทำงานอยู่ประมาณ 640 นาโนวินาที หรือค่า $T_c = 1.5625 \times 10^{-7}$ วินาที และ $M = 64$ เพราะฉะนั้นความถี่สูงสุดของการสุ่มค่าเหล่านี้ 10 กิโลเฮิร์ตซ์ นั่นคือ A/D แบบนี้ใช้ได้กับสัญญาณที่ความถี่สูงสุดได้มากกว่า 5 กิโลเฮิร์ตซ์

การเพิ่มอัตราการสุ่มสัญญาณให้เร็วขึ้นค่า 10^{-7} วินาที

(ก) โดยการเพิ่มจำนวนของสัญญาณที่นำให้สุ่ม นั่นคือการทำให้ $T_{A/D}$ มีค่าลดลง อย่างไรก็ตาม A/D จะมีค่าความถี่สูงสุดได้ตามที่บริษัทผู้ผลิตอุปกรณ์กำหนดไว้

(ข) โดยการให้ A/D แบบขนานที่ค่าความถี่สูง วิธีการนี้ A/D จะมีราคาแพง และต้องใช้งานอยู่ในระดับที่ค่าความถี่สูง ทำให้เกิดความยุ่งยากและอายุการใช้งานที่สั้น

(ค) โดยการเพิ่มจำนวนอุปกรณ์ A/D ในมากขึ้น ซึ่งวิธีการนี้จะใช้ค่าอ้างอิงในเรขาคณิต

การออกแบบระบบ A/D ที่มีอัตราการสุ่มสูง

หลักการเบื้องต้นของวิธีการนี้คือ การสุ่มโดยใช้ time division multiplex โดย A/D หลายตัว จะถูกใช้โดยการสุ่ม T_s เวลาของสัญญาณเป็น M คาบเวลาของหน่วย T_c นั่นคือใช้ D M ตัวในการสุ่มโดยใช้เวลาของสัญญาณ $T_{A/D}$ ทุกตัวมีคุณสมบัติเหมือนกัน โดยสัญญาณ

1. บทนำ

ในการประมวลผลสัญญาณเชิงดิจิทัล (digital signal processing) นั้น อุปกรณ์ที่ใช้ทั่วไปจะประกอบด้วย วงจรเปลี่ยนสัญญาณอนาล็อกเป็นดิจิทัล (Analog to Digital Converter) เสมอ ทั้งนี้เนื่องจากสัญญาณส่วนใหญ่ในระบบมักเป็นสัญญาณแบบอนาล็อก ซึ่งการแปลงสัญญาณจะเปลี่ยนแปลงอย่างต่อเนื่องกับเวลา วงจรเปลี่ยนสัญญาณอนาล็อกเป็นดิจิทัล (ต่อไปนี้เรียกว่า A/D) เป็นอุปกรณ์อย่างหนึ่งที่มีค่าของช่วงความถี่สัญญาณงานของการประมวลผลเชิงดิจิทัล โดยเฉพาะ A/D แบบราคาถูกใช้เวลาใช้ในการเปลี่ยนสัญญาณกับเวลาตาม อย่างไรก็ตามค่าความถี่สัญญาณการนำเอา A/D แบบราคาถูกมาใช้เพื่อให้เกิด ช่วงความถี่สัญญาณงานของวงจรรวมแล้วสูงกว่า ช่วงความถี่สัญญาณงานของตัว A/D แล้วก็ตามค่าความถี่ของสัญญาณจะสูงมาก ทั้งนี้เนื่องจากในกรณีที่ A/D ที่มีช่วงความถี่สัญญาณงานที่มากกว่านั้นค่าความถี่สัญญาณจะสูง และค่าที่ได้อาจมีค่ามาก

2. การสุ่มสัญญาณ

อุปกรณ์ A/D โดยทั่วไปมีการทำงานอยู่หลายแบบ แต่แบบที่นำมาพิจารณานี้ มีการทำงานเป็นแบบ "การประมาณแบบต่อเนื่อง" (Successive approximation) [1] A/D แบบนี้ใช้เวลาในการแปลง (conversion time) จากสัญญาณอนาล็อกเป็นดิจิทัลจะมีความถี่ที่ขึ้นอยู่กับขนาดของสัญญาณที่เข้ามาว่าค่าเท่าใด และเวลาการแปลงจะออกเป็นจำนวนรอบสัญญาณ (cycle) ของสัญญาณนาฬิกา (clock signal) ที่ใช้ [2]

ให้ $x(t)$ เป็นสัญญาณอนาล็อกที่ต้องการแปลงเป็นสัญญาณดิจิทัล $x(n)$ และ

$$x(n) = x(t) \Big|_{t=nT_s} \quad (1)$$

$x(n)$ นี้เรียกว่าเป็น ลำดับสัญญาณดิจิทัล (digital sequence)

T_s เป็นคาบเวลาการสุ่ม (sampling period)

ถ้าให้ช่วงความถี่สัญญาณงาน หรือแบนด์วิดท์ (bandwidth) ของสัญญาณอนาล็อกมีค่าเท่ากับ B และจะนับโดยอาศัย ทฤษฎีการสุ่ม (sampling theory) จะได้

1-5-4

หากให้ค่าของ M เพิ่มขึ้น เพราะฉะนั้นเวลาการสุ่มของระบบรวมของอุปกรณ์ A/D ที่มี Time division multiplex นี้เป็น

$$T = T_s/M$$

หรือเวลาการสุ่มของทั้งระบบลดลง M เท่า นั่นหมายความว่าความถี่ของสัญญาณอนาล็อกที่ค่าความถี่สามารถเพิ่มขึ้นเป็น M เท่าด้วยเช่นกัน

วงจรที่ออกแบบและใช้ในการทดลองได้แสดงไว้ในรูปที่ 1 A/D แต่ละตัวจะมีค่า B/M (sampling and hold) ต่อเนื่องและจะทำงานกับของสัญญาณอนาล็อกที่ทำการสุ่มไว้ที่ค่าที่ (zero order hold) [3] ลักษณะการทำงานของสัญญาณของ A/D การทำงานของระบบสามารถอธิบายได้โดยเขียนเป็นโปรแกรม ดังรูปที่ 2 จะเห็นว่าเวลาในการสุ่มของ A/D ตัวที่ ๑ จะมีการนำเอาสัญญาณเข้าในการเปลี่ยนสัญญาณอนาล็อกเป็นดิจิทัลเป็นจำนวน T_s เวลาการสุ่มระหว่าง A/D ตัวที่ ๑ กับ $M-1$ ตัวนั้น T_s/M ทำให้ความถี่ของระบบมีค่ามากขึ้น

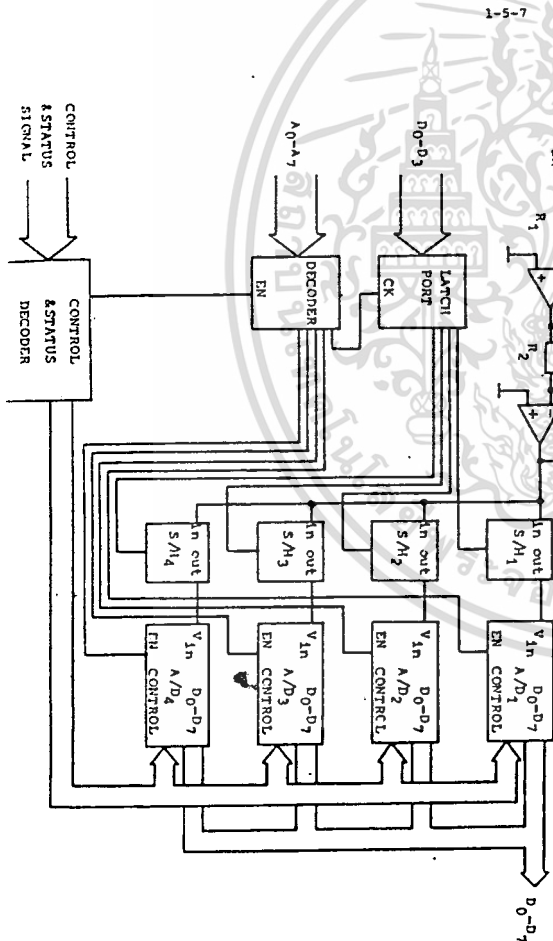
4. การทดลอง

ในการทดลองใช้ A/D แบบ ADC0804 ซึ่งมี conversion time 100 microsecond บิตสัญญาณความถี่ ๘ MHz 4 V $p-p$ ดังรูป 3a เข้าไปยังระบบในวงจรแบบ A/D เพื่อตัวเดียว และทำการสุ่มสัญญาณอนาล็อก เพื่อแปลงเป็นสัญญาณดิจิทัลด้วยเวลาเท่ากับคาบการ conversion time นำเอาค่าที่ได้ไปทำ curve fitting แล้วพล็อตดังรูป 3b จะเห็นว่าเมื่อความถี่สัญญาณมีค่าสูงกว่า bandwidth ของระบบซึ่งถูกกำหนดโดย conversion time จะทำให้เกิดการ Distortion ของสัญญาณที่ค่าความถี่สูง ทำให้ไม่สามารถอธิบายของสัญญาณที่มีความถี่ได้ ความถี่สัญญาณที่เราใช้ในระบบ ซึ่งมี A/D 4 ตัว ทำการสุ่มแต่ละตัว M กับ $M-1$ ด้วยเวลาเท่ากับ 25 microsecond คือ A/D ตัวที่ ๑ จะสุ่มทุกๆ 100 microsecond นำค่าที่ได้ไปทำ Curve fitting แล้วพล็อตดังรูป 3c จะเห็นว่ารูปสัญญาณ หากสุ่มมีความถี่ค่าหนึ่งกับสัญญาณที่เข้ามาแล้ว เพราะสามารถนำเอาไปใช้งานได้อย่างถูกต้อง

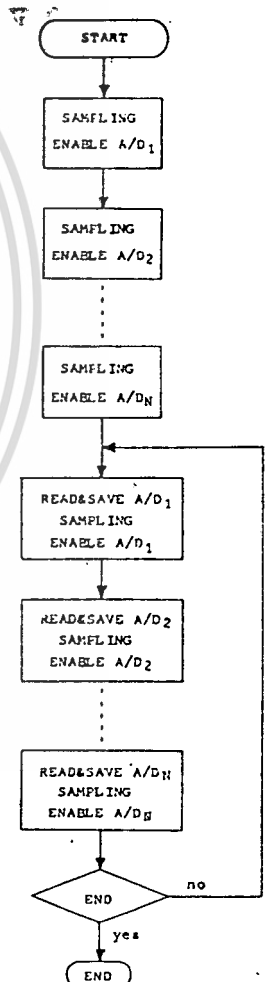
តិកត រង្វៀវ = ការ។

ผู้เรียนควรรอบรู้ ทศ.กร. อื่นด้วย โกลาบอลยุคใหม่ และ กร. ร้องถาม กระทำซ้ำซ้ำ

1. DOUGLAS V. HALL, "Microprocessors and Digital System," McGraw-Hill Company, 1980
2. National Semiconductor Corporation, LINEAR DATA BOOK, 1982, pp. 859
3. Joseph S. ROSKO, "Digital Simulation of Physical Systems," Addison Wesley Publishing Company, 1972



รูปที่ 1 วงจรระบบเบรกไฟฟ้า : จดหมายออกเป็นสัญญาณ
กิจกรรมที่กักการควบคุม



מנהג נביאנו ר' יצחק: ייחוד יר' א' 2 ז' י' י'

ภาคผนวก จ. การแก้วงจรแบบสัญลักษณ์ด้วยโปรแกรม



สัจจน์ จันทวังษ์
ลือชัย ไชโยดม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้า
เจ้าคุณทหาร ลาดกระบัง

บทย่

บทความนี้เป็นการใช้วิธีการโปรแกรมด้วยสัญลักษณ์ในการวิเคราะห์วงจรอิเลคทรอนิกส์ ซึ่งผลลัพธ์ที่ได้อยู่ในรูปของสัญลักษณ์ทำให้เราสามารถวิเคราะห์วงจรได้ทั้งใน Analytic form หรือนำไปหาคะตอบของวงจรในรูปของ Time domain Frequency domain

Abstract

This paper presents a symbolic programming technique for analyzing an electronic circuit. The result is in a symbolic form that we can use to analyse the circuit in analytic form or to find the circuit response in time domain and frequency domain.

3. การกระทำทางคณิตศาสตร์ (Arithmetic operation)

การบวกและลบสำหรับ Symbolic programming จะนำเอาค่าคงที่ของเทอมที่มีค่าคงที่เท่ากันทั้งหมดมาบวกหรือลบกัน เช่น

$$\begin{aligned} Y1 &= (1.0, 1, 1, 0, 0) - (2.3, 1, 0, 0, 1) \\ Y2 &= (1.38, 0, 0, 1, 1) + (0.5, 1, 1, 0, 0) \\ Y1 + Y2 &= (1.5, 1, 1, 0, 0) - (2.3, 1, 0, 0, 1) + (1.38, 0, 0, 1, 1) \\ Y1 - Y2 &= (0.5, 1, 1, 0, 0) - (2.3, 1, 0, 0, 1) - (1.38, 0, 0, 1, 1) \end{aligned}$$

การคูณและหารจะนำเอาค่าคงที่ของตัวแปรในแต่ละเทอมมาคูณหรือลบกันเช่น
ในการคูณหรือหาร คำนวณด้วย เช่น

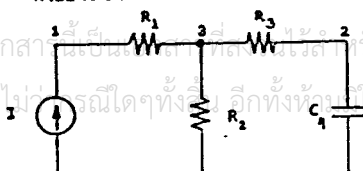
$$\begin{aligned} Y1 &= (0.5, 1, 1, 0, 0) \\ Y2 &= (2.5, 1, 0, 1, 1) \\ Y1 * Y2 &= (1.25, 2, 1, 1, 1) \\ Y1 / Y2 &= (0, 2, 0, 1, -1) \end{aligned}$$

4. การใช้การโปรแกรมด้วยสัญลักษณ์เพื่อหา Transfer function

ใน Network ใดๆ เราสามารถเขียน Nodal admittance matrix ได้ ซึ่งเมทริกซ์ (Matrix) จะให้ความสัมพันธ์ระหว่าง node voltage และ branch current

$$\begin{aligned} Y_n V_n &= A I \\ A &= \text{Incidence matrix} \\ V_n &= \text{Node-voltage vector} \\ I &= \text{Source - current vector} \\ Y_n &= \text{Nodal admittance matrix} \end{aligned}$$

ตัวอย่าง Network



1. บทนำ

ในปัจจุบันมีโปรแกรมวิเคราะห์วงจรอิเลคทรอนิกส์ด้วยคอมพิวเตอร์หลายโปรแกรมเช่น Pspice เป็นการวิเคราะห์โดยวิธีการจำลอง (Simulate) การทำงานของวงจรด้วยคอมพิวเตอร์ แต่สิ่งสำคัญของทฤษฎีวิเคราะห์วงจรคือการหาทรานสเฟอร์ฟังก์ชัน (Transfer function) ของวงจรซึ่งไม่สามารถหาได้ด้วยการจำลองด้วยคอมพิวเตอร์ ซึ่งจากทรานสเฟอร์ฟังก์ชันเราสามารถหาคะตอบเชิงวิเคราะห์ได้ ทั้งในกรณีเฉพาะเรื่อง และในกรณีทั่วไป (Global behavior) แต่ปัญหาในการหาทรานสเฟอร์ฟังก์ชันคือความยุ่งยากซับซ้อนของการคำนวณ โดยเฉพาะในทางปฏิบัติ ซึ่งประกอบด้วยทรานซิสเตอร์หลายตัวเสมอ ทำให้การหาทรานสเฟอร์ฟังก์ชันเมื่อเกิดจะเป็นไปไม่ได้เพราะเสียเวลามากและสิ่งที่ตามมาคือมีความแม่นยำต่ำจนเกินไป จึงได้มีการพัฒนาโปรแกรมคอมพิวเตอร์ซึ่งใช้สำหรับหาทรานสเฟอร์ฟังก์ชันโดยอัตโนมัติการซึ่งเสนอในบทความ[1] ทำให้สามารถหาทรานสเฟอร์ฟังก์ชันได้สะดวกอันจะเป็นประโยชน์ในการวิเคราะห์วงจร ในบทที่ 2 จะแสดงการโปรแกรมด้วยสัญลักษณ์ในบทที่ 3 แสดงการกระทำทางคณิตศาสตร์โดยวิธีโปรแกรมด้วยสัญลักษณ์ ในบทที่ 4 แสดงการหาทรานสเฟอร์ฟังก์ชันโดยทั่วไป ส่วนในบทที่ 5 เป็นการโปรแกรมคอมพิวเตอร์หาทรานสเฟอร์ฟังก์ชันของตัวอย่างในบทที่ 4 และบทที่ 6 เป็นบทสรุป

2. การโปรแกรมด้วยสัญลักษณ์ (Symbolic programming)

เป็นวิธีการแก้ปัญหาโดยวิธีสัญลักษณ์ คือตัวแปรในสมการจะแก้ปัญหามันของสัญลักษณ์ โดยไม่มีการแทนค่าลงในสมการใน Computer เมื่อได้คำตอบแล้วเราสามารถทำการ Normalize ค่าตอบที่ได้ ในแต่ละเทอมของสมการจะเก็บไว้เป็นค่าคงที่และค่าคงที่ของตัวแปร (1)

$$Y = \sum_{i=1}^k (C_i, E_{i1}, E_{i2}, \dots, E_{in})$$

C_i = ค่าคงที่ของเทอมที่ i
 E_{ni} = ค่าสัมประสิทธิ์ของตัวแปรที่ n ในเทอมที่ i
 k = จำนวนเทอมสูงสุดในสมการ

ตัวอย่าง สมการ

$$Y = 2.5ax + 2.0by$$

จะเขียนเป็นรูปของ Symbolic programming technique ได้
 $Y = (2.5, 1, 1, 0, 0) + (2.0, 0, 0, 1, 1)$

สามารถเขียน Nodal admittance matrix ได้ดังนี้

$$Y_n = \begin{bmatrix} 1/R_1 & C & -1/R1 \\ 0 & 1/R_2 - SC & -1/R2 \\ -1/R1 & -1/R2 & 1/R1 + 1/R2 + 1/R3 \end{bmatrix}$$

เมื่อหาค่า A, I, Y_n แล้วเราสามารถหา V_n

จาก Nodal admittance matrix จะทำ Gauss elimination เพื่อเปลี่ยน Y_n ให้เป็น Upper triangular matrix วิธีการทำ Gauss elimination จากสมการ (1)

$$\begin{aligned} y_{11}v_1 + y_{12}v_2 + \dots + y_{1n}v_n &= i_1 \\ y_{21}v_1 + y_{22}v_2 + \dots + y_{2n}v_n &= i_2 \\ &\vdots \\ y_{n1}v_1 + y_{n2}v_2 + \dots + y_{nn}v_n &= i_n \end{aligned} \quad (1)$$

โดยวิธี Gaussian elimination จะหาสมการแถวที่ 1 ของสมการ (1) เมื่อ $y_{11} = 0$ ด้วย y_{11} จะได้

$$\begin{aligned} y_{11} + b_{12}v_2 + b_{13}v_3 + \dots + b_{1n}v_n &= i_1/y_{11} \quad (2) \\ \text{ซึ่ง } b_{ij} &= \frac{y_{ij}}{y_{11}} \quad (j > 1) \end{aligned}$$

จากสมการ (1) เราสามารถกำจัดตัวแปร v_1 จาก (1) โดยเอา (2) คูณด้วย y_{11} ลบออกจากสมการที่ 3 ใน (1) จนถึงสมการที่ n ใน (1) จะได้

$$\begin{aligned} y^{(1)}_{22}v_2 + y^{(1)}_{23}v_3 + \dots + y^{(1)}_{2n}v_n &= i^{(1)}_2 \\ y^{(1)}_{32}v_2 + y^{(1)}_{33}v_3 + \dots + y^{(1)}_{3n}v_n &= i^{(1)}_3 \\ &\vdots \\ y^{(1)}_{n2}v_2 + y^{(1)}_{n3}v_3 + \dots + y^{(1)}_{nn}v_n &= i^{(1)}_n \end{aligned}$$

จากสมการ (1) เราสามารถกำจัดตัวแปร v_1 จาก (1) โดยเอา (2)

ซึ่งคูณด้วย y_{11} ลบออกจากสมการที่ 3 ใน (1) จนถึงสมการที่ n ใน (1) จะได้

$$\begin{aligned} y^{(1)}_{ij} &= y_{ij} - y_{1i}b_{1j} \\ i^{(1)}_i &= i_i - y_{1i}i_1/y_{11} \end{aligned}$$

เอกสารนี้เป็นเอกสารที่จัดทำขึ้นเพื่อการเรียนการสอนเพื่อการศึกษาเท่านั้น ไม่สามารถนำไปใช้เพื่อการค้าโดยไม่ได้รับอนุญาตจากเจ้าของลิขสิทธิ์ หากต้องการนำเอกสารนี้ไปใช้เพื่อการค้า กรุณาติดต่อเจ้าของลิขสิทธิ์เพื่อขออนุญาต

ใน Symbolic programming coefficient y_{ij} เป็นเหมือนประ
 คอบคัยเหมือนอย่างมากกว่า 1 เหนือ เพื่อแก้ปัญหาในการหา Gaussian
 elimination จาก (1) จะกำจัดค่า y_{ij} ออกจากสมการที่ 2 ถึง n ใน (1)
 โดยเอาสมการที่ 2 ใน (1) คูณด้วย y_{11} ลบด้วยสมการที่ 1 ใน (1) ซึ่งคูณด้วย
 y_{21} และ เอาสมการที่ 3 ใน (1) คูณด้วย y_{11} ลบด้วยสมการที่ 1 ใน (1) ซึ่ง
 คูณด้วย y_{31} จนถึงสมการที่ n ใน (1) จะได้ System ที่ประกอบด้วยสมการ (1')
 โดย

$$y_{ij}^{(1)} = y_{ij}y_{11} - y_{1i}y_{1j}$$

ค่าที่ป้อน Lower triangle เป็น 0 เหนือจะเป็น Diagonal matrix
 และเมื่อทำให้เป็น Identity matrix จะสามารถหา Node-voltage vector ได้
 ในหน่วยเดียวกันถ้าเราทราบ A, V_n, V_n เราสามารถหา I ได้

5. โปรแกรมในการแก้สมการด้วย Symbolic programming

โปรแกรมสำหรับแก้สมการด้วย Symbolic programming ได้เขียนขึ้นด้วย
 ภาษา Turbo pascal บนเครื่อง IBM compatible ขึ้นมาความจำ 640 Kbyte
 และ 8087 mathematic coprocessor จากงานวิจัยที่ 1 เขียน Nodal
 admittance matrix และ Source - current vector เข้าไปยัง Computer
 ที่จะมีรูปแบบดังนี้

บรรทัดที่ 1 จะบอก (จำนวน Independent variable) - 1

บรรทัดที่ 2 จะบอก (จำนวน dependent variable) - 1

ตั้งแต่บรรทัดที่ 3 จะบอกค่าในแถวที่ i จาก Column number ตามด้วย Term
 number ของ Column number ข้างต้น

ตัวอย่างการป้อน Input file

สมการที่ 1 ในรูปที่ 1

```
2
0
0 0 (1.0,-1.0,0.0)
1 0 (0.0,0.0,0.0)
2 0 (-1.0,-1.0,0.0)
=
(1.0,0.0,0.0)
```

ผลลัพธ์ในหัวข้อแถวที่ 2

```
2
0
0 0 (0.0,0.0,0.0)
0 0 (0.000000000000E+000,0.0,0.0)
1 0 (1.000000000000E+000,-1,-1,-2.0)
1 1 (2.000000000000E+000,-1,-1,-1.1)
1 2 (1.000000000000E+000,-1,-1,0.2)
1 3 (1.000000000000E+000,-1,0,-2.1)
1 4 (1.000000000000E+000,-1,0,-1.2)
2 0 (0.000000000000E+000,0.0,0.0)
=
(1.000000000000E+000,-1,0,-2.0)
(1.000000000000E+000,-1,0,-1.1)
```

ผลลัพธ์ในหัวข้อแถวที่ 3

```
2
0
0 0 (0.0,0.0,0.0)
0 0 (0.000000000000E+000,0.0,0.0)
1 0 (0.000000000000E+000,0.0,0.0)
2 0 (1.000000000000E+000,-1,-1,-1.0)
2 1 (1.000000000000E+000,-1,-1,0.1)
2 2 (1.000000000000E+000,-1,0,-1.1)
=
(1.000000000000E+000,-1,0,-1.0)
(1.000000000000E+000,-1,0,0.1)
```

6. สรุป

งานที่นำเสนอเพื่อพิสูจน์ให้เห็นว่าอยู่ในรูปของ Laplace transform เราสามารถ
 หาการวิเคราะห์ทางจลนศาสตร์ Analytic และถ้าต้องการวิเคราะห์แบบ Numerical
 เราสามารถทำได้โดยการแทนค่าในทางสเปกตรัมขึ้น โดยไม่ต้องคำนวณใหม่แต่ทำให้
 ประสิทธิภาพ หรือการหา Inverse laplace transform เราสามารถวิเคราะห์
 ทางจลนศาสตร์ Time domain ได้ ถึงแม้ว่าการโปรแกรมด้วยพีซีสักครั้งจะมีประโยชน์มากแต่
 ข้อเสียคือการเก็บค่าคงที่และค่ายกกำลังของตัวแปรแต่ละเทอมและใช้เนื้อที่หน่วย
 ความจำที่น้อยและ 1 หน่วย ความจำที่น้อยถ้า Gauss elimination จะทำให้หน่วย
 ความจำที่น้อยและ 1 หน่วย ความจำที่น้อยถ้า Gauss elimination จะทำให้หน่วย

สมการที่ 2 ในรูปที่ 1

```
2
0
0 0 (0.0,0.0,0.0)
1 0 (1.0,0.0,-1.0)
1 1 (1.0,0.0,0.1)
2 0 (-1.0,0.0,-1.0)
=
(0.0,0.0,0.0)
สมการที่ 3 ในรูปที่ 1
2
0
0 0 (-1.0,-1.0,0.0)
1 0 (-1.0,0.0,-1.0)
2 0 (1.0,-1.0,0.0)
2 1 (1.0,0,-1.0,0)
2 2 (1.0,0.0,-1.0)
=
(0.0,0.0,0.0)
```

เมื่อ Computer ทำงานแล้วได้ผลลัพธ์

ผลลัพธ์ในหัวข้อแถวที่ 1

```
2
0
0 0 (0.0,0.0,0.0)
0 0 (1.000000000000E+000,-2,-1,-1.0)
0 1 (1.000000000000E+000,-2,-1,0.1)
0 2 (1.000000000000E+000,-2,0,-1.1)
1 0 (0.000000000000E+000,0.0,0.0)
2 0 (0.000000000000E+000,0.0,0.0)
=
(1.000000000000E+000,-1,-1,-1.0)
(1.000000000000E+000,-1,-1,0.1)
(1.000000000000E+000,-1,0,-1.1)
(1.000000000000E+000,-2,0,-1.0)
(1.000000000000E+000,-2,0,0.1)
```

เพราะปัจจุบันคอมพิวเตอร์มีหน่วยความจำจำนวนมากและเราสามารถใส่ Hard disk
 เพื่อช่วยเก็บข้อมูลระหว่างการทำงานได้

เอกสารอ้างอิง

- [1] S.pookaiyaudom and T.kuhanont,"Automatic Symbolic
 Polynomial Operations", IEEE Trans.Proc., vol.65, no.5, pp.
 811-812, 1977

การศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ฉ. โปรแกรมการจำลองออสซิลเลเตอร์แบบวินบริคส์ด้วยสภาวะ
เริ่มต้นต่างๆ



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

1 (* 8-AUGUST 1985 *)
2 (* NEW CONVOLUTION *)
3 (* Simulated Oscillator With Automatic Gain Control *)
4 PROGRAM OSCILLATOR;
5 VAR CT : ARRAY[0..10] OF REAL;
6 I,J,GN,GO,TK,NO,OU,LP,LG,PO,PO1,PO2,LQ,LJ,LK : INTEGER;
7 A,B,X,Y,GAIN,VAL1,VAL2,R,CA,RC,T1,T2,LEVEL,SGAIN,DELTA_T,W,USX,US
X,LSY : REAL;
8 JR,A11,Y11,Y21,AGCVAL,COA1,COA2,E_H1,E_H2,CO_H1,CO_H2,TAO : REAL;
9 ST : CHAR;
10 FUNCTION RND : REAL;
11 VAR S,T : REAL;
12 P : INTEGER;
13 BEGIN
14 IF OU < 2 THEN
15 BEGIN
16 RND := -0.9;
17 END
18 ELSE
19 BEGIN
20 S := RANDOM;
21 T := FRAC(10 * S) * 10;
22 P := TRUNC(T);
23 IF ODD(P) THEN S := -S;
24 RND := S * 1E-3;
25 END;
26 END;
27 PROCEDURE READ_COEFFICIENT;
28 BEGIN
29 CT[0] := 3.1031368555;
30 CT[1] := 1.5473467415E-2;
31 CT[2] := -1.5272476131E-1;
32 CT[3] := 2.5022686030E-2;
33 CT[4] := -1.5182273859E-3;
34 CT[5] := 3.108006317E-5;
35 CT[6] := 7.3529695704E-8;
36 END;
37 PROCEDURE GRAPH;
38 CONST SCALE = 5;
39 VAR K : INTEGER;
40 LENGTH : REAL;
41 BEGIN
42 LJ := LJ + 1;
43 IF LJ = LK THEN
44 BEGIN
45 LENGTH := ((Y11 - Y21) * GAIN + 6.0) * SCALE;
46 K := TRUNC(LENGTH);
47 WRITELN('K:', '59 - K');
48 LJ := 0;
49 END;
50 END;
51 PROCEDURE CHECK_PEEK;
52 VAR GT,GJ : REAL;
53 GJ : INTEGER;
54 LABEL NEXT;
55 BEGIN
56 AGCVAL := ABS(X);
57 AGCVAL := A11 * COA2 + AGCVAL * COA1;
58 A11 := AGCVAL;
59 GT := AGCVAL;
60 GI := GT;
61 GAIN := CT[0];
62 FOR GJ := 1 TO 6 DO
63 BEGIN
64 GAIN := GAIN + CT[GJ] * GI;
65 GI := GI * GT;
66 END;
67 IF ((X - VAL2) * (VAL2 - VAL1)) < 0 THEN
68 BEGIN
69 IF (X >= 0.0) THEN
70 BEGIN
71 LP := I;
72 PO1 := TRUNC(X * LSY);
73 PO1 := 100 - PO1;
74 PO2 := TRUNC(I * LSX);
75 DRAW(PO2,100,PO2,PO1,1);
76 END;
77 IF (X < 0.0) THEN
78 BEGIN
79 PO1 := TRUNC(X * LSY);
80 PO1 := 100 - PO1;
81 PO2 := TRUNC(I * LSX);
82 DRAW(PO2,100,PO2,PO1,1);
83 END;
84 END;
85 NEXT: VAL1 := VAL2;
86 VAL2 := X;
87 END;
88 PROCEDURE INT_GRP;
89 BEGIN
90 (* Upper = v , Lower = Amplitude *)
91 HIRES;
92 HIRESCOLOR(YELLOW);
93 DRAW(0,0,0,200,1);
94 DRAW(0,100,600,100,1);
95 LSY := 100 / 13.0;
96 LSX := 600 / NO;
97 END;
98 BEGIN
99 LJ := 0;
100 R := 1;
101 CA := 1;
102 RC := R * CA;
103 W := 1 / RC;
104 NO := 250;
105 DELTA_T := 2 * 3.1415927 * RC / NO;
106 LK := 10;
107 LG := NO;
108 COA1 := 2 * 3.1415927 / NO / 10.32;
109 COA2 := EXP(-(COA1));
110 COA1 := COA1 * EXP(-(COA1) / 2);
111 OU := 1;
112 PO := TRUNC(0.3 * LG);
113 NO := 5000;
114 LP := 0;
115 VAL1 := 0.0;
116 VAL2 := 0.0;
117 A11 := 0;
118 READ_COEFFICIENT;
119 WRITELN;
120 TAO := R * CA;
121 A := (3 + SQRT(5)) / 2.0;
122 B := (3 - SQRT(5)) / 2.0;
123 CO_H1 := A * DELTA_T * EXP(-A * DELTA_T * 0.5 / TAO) / (TAO
T(5));
124 CO_H2 := B * DELTA_T * EXP(-B * DELTA_T * 0.5 / TAO) / (TAO
T(5));
125 E_H1 := EXP(-A * DELTA_T / TAO);
126 E_H2 := EXP(-B * DELTA_T / TAO);
127 Y11 := 0.0;
128 Y21 := 0.0;
129 GAIN := CT[0];
130 FOR I := 0 TO NO DO
131 BEGIN
132 X := RND + (Y11 - Y21) * GAIN;
133 Y11 := CO_H1 * X + Y11 * E_H1;
134 Y21 := -CO_H2 * X + Y21 * E_H2;
135 GRAPH;
136 X := (Y11 - Y21) * GAIN;
137 CHECK_PEEK;
138 OU := OU + 1;
139 END;
140 END.

```

ภาคผนวก ช. โปรแกรมการจำลองอินเจคชั่นล๊อกกิ้ง



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

1 (*p103;
2 [This program is used for simulate INJECTION LOCKING phenomenon
3 of two oscillator that has nearly frequency
4 this program calculate lock frequency]
5 Program Injection_locking;
6
7 (*
8 {SI typedef.sys}
9 {SI graphix.sys}
10 {SI kernel.sys}
11 *)
12 Const pi = 3.1415927; {pi}
13 n = 250; {sampling per period}
14 cop = 0.2; {coupling factor varies}
15 tao1 = 6.28; {Time constant of lowpass1 = k / w0;}
16 loss = 0.1; {loss in system}
17 cop2 = 0.2; {coupling factor2 fixed}
18 tao2 = 6.28;
19 loss2 = 0.1;
20 q = 0.93;
21 w_start = 0.86;
22 w_max = 1.20;
23 qs = 0.93;
24
25 Var r1a,r2a,q2,delta_t,k11,k12,k13,k14,k15,k16,a11,a12,b10,b12 : re
26
27 y10,y11,y12,low_c1,low_c2,low_y,step_i,vout,vgate,gain : real;
28 agc : array[0..6] of real;
29 i,j,k,l,m : integer;
30 count1,count2 : real;
31 k21,k22,k23,k24,k25,k26,a21,a22,b20,b22 : real;
32 y20,y21,y22,low2_c1,low2_c2,low2_y,vout2,vgate2,gain2 : real;
33 agc2 : array[0..6] of real;
34 w0,w1,v_osc1,v_osc2 : real;
35 va_osc1,va_osc2,vb_osc1,vb_osc2 : real;
36 status : char;
37 w_lock,x_cor,t_v1,t_v2 : real;
38 tn : real;
39
40 Function vin1 : real;
41 begin
42   vin1 := cop * vout2;
43 end;
44
45 Function vin2 : real;
46 begin
47   vin2 := cop2 * vout;
48 end;
49
50 Function rnd : real;
51 var s,t : real;
52
53 p : integer;
54 begin
55   s := random;
56   t := frac(10 * s) * 10;
57   p := trunc(t);
58   if odd(p) then s := -s;
59   rnd := s * 1e-3;
60 end;
61
62 Procedure osc_parameter;
63 begin
64   {Oscillator simulation factor}
65   k11 := 2 * w0 * delta_t;
66   k12 := sqrt(delta_t * w0) * q;
67   k13 := 4 * q + k11 + k12;
68   k14 := -8 * q + 2 * k12;
69   k15 := 4 * q - k11 + k12;
70   k16 := 2 * w0 * delta_t / k13;
71   a11 := -k14 / k13;
72   a12 := -k15 / k13;
73   b10 := 1.0;
74   b12 := -1.0;
75
76   k21 := 2 * w1 * delta_t;
77   k22 := sqrt(delta_t * w1) * q2;
78   k23 := 4 * q2 + k21 + k22;
79   k24 := -8 * q2 + 2 * k22;
80   k25 := 4 * q2 - k21 + k22;
81   k26 := 2 * w1 * delta_t / k23;
82   a21 := -k24 / k23;
83   a22 := -k25 / k23;
84   b20 := 1.0;
85   b22 := -1.0;
86 end;
87
88 {Procedure drawframe;
89 begin
90   drawline(0,5,2000,5);
91   drawline(0,-5,2000,-5);
92 end;
93
94 procedure draw_x;
95 begin
96   drawpoint(x_cor,vout);
97   drawpoint(x_cor,vout2);
98   x_cor := x_cor + 1;
99   if x_cor > 2000 then
100     begin
101       x_cor := 0;
102       clearsreen;
103     end;
104   end;
105 }
106
107 (*p103;
108 [This program is used for simulate INJECTION LOCKING phenomenon
109 of two oscillator that has nearly frequency
110 this program calculate lock frequency]
111 Program Injection_locking;
112
113 (*
114 {SI typedef.sys}
115 {SI graphix.sys}
116 {SI kernel.sys}
117 *)
118 Const pi = 3.1415927; {pi}
119 n = 250; {sampling per period}
120 cop = 0.2; {coupling factor varies}
121 tao1 = 6.28; {Time constant of lowpass1 = k / w0;}
122 loss = 0.1; {loss in system}
123 cop2 = 0.2; {coupling factor2 fixed}
124 tao2 = 6.28;
125 loss2 = 0.1;
126 q = 0.93;
127 w_start = 0.86;
128 w_max = 1.20;
129 qs = 0.93;
130
131 Var r1a,r2a,q2,delta_t,k11,k12,k13,k14,k15,k16,a11,a12,b10,b12 : re
132
133 y10,y11,y12,low_c1,low_c2,low_y,step_i,vout,vgate,gain : real;
134 agc : array[0..6] of real;
135 i,j,k,l,m : integer;
136 count1,count2 : real;
137 k21,k22,k23,k24,k25,k26,a21,a22,b20,b22 : real;
138 y20,y21,y22,low2_c1,low2_c2,low2_y,vout2,vgate2,gain2 : real;
139 agc2 : array[0..6] of real;
140 w0,w1,v_osc1,v_osc2 : real;
141 va_osc1,va_osc2,vb_osc1,vb_osc2 : real;
142 status : char;
143 w_lock,x_cor,t_v1,t_v2 : real;
144 tn : real;
145
146 Function vin1 : real;
147 begin
148   vin1 := cop * vout2;
149 end;
150
151 Function vin2 : real;
152 begin
153   vin2 := cop2 * vout;
154 end;
155
156 Function rnd : real;
157 var s,t : real;
158
159 p : integer;
160 begin
161   s := random;
162   t := frac(10 * s) * 10;
163   p := trunc(t);
164   if odd(p) then s := -s;
165   rnd := s * 1e-3;
166 end;
167
168 Procedure osc_parameter;
169 begin
170   {Oscillator simulation factor}
171   k11 := 2 * w0 * delta_t;
172   k12 := sqrt(delta_t * w0) * q;
173   k13 := 4 * q + k11 + k12;
174   k14 := -8 * q + 2 * k12;
175   k15 := 4 * q - k11 + k12;
176   k16 := 2 * w0 * delta_t / k13;
177   a11 := -k14 / k13;
178   a12 := -k15 / k13;
179   b10 := 1.0;
180   b12 := -1.0;
181
182   k21 := 2 * w1 * delta_t;
183   k22 := sqrt(delta_t * w1) * q2;
184   k23 := 4 * q2 + k21 + k22;
185   k24 := -8 * q2 + 2 * k22;
186   k25 := 4 * q2 - k21 + k22;
187   k26 := 2 * w1 * delta_t / k23;
188   a21 := -k24 / k23;
189   a22 := -k25 / k23;
190   b20 := 1.0;
191   b22 := -1.0;
192 end;
193
194 {Procedure drawframe;
195 begin
196   drawline(0,5,2000,5);
197   drawline(0,-5,2000,-5);
198 end;
199
200 procedure draw_x;
201 begin
202   drawpoint(x_cor,vout);
203   drawpoint(x_cor,vout2);
204   x_cor := x_cor + 1;
205   if x_cor > 2000 then
206     begin
207       x_cor := 0;
208       clearsreen;
209     end;
210   end;
211 }
212
213 (*p103;
214 [This program is used for simulate INJECTION LOCKING phenomenon
215 of two oscillator that has nearly frequency
216 this program calculate lock frequency]
217 Program Injection_locking;
218
219 (*
220 {SI typedef.sys}
221 {SI graphix.sys}
222 {SI kernel.sys}
223 *)
224 Const pi = 3.1415927; {pi}
225 n = 250; {sampling per period}
226 cop = 0.2; {coupling factor varies}
227 tao1 = 6.28; {Time constant of lowpass1 = k / w0;}
228 loss = 0.1; {loss in system}
229 cop2 = 0.2; {coupling factor2 fixed}
230 tao2 = 6.28;
231 loss2 = 0.1;
232 q = 0.93;
233 w_start = 0.86;
234 w_max = 1.20;
235 qs = 0.93;
236
237 Var r1a,r2a,q2,delta_t,k11,k12,k13,k14,k15,k16,a11,a12,b10,b12 : re
238
239 y10,y11,y12,low_c1,low_c2,low_y,step_i,vout,vgate,gain : real;
240 agc : array[0..6] of real;
241 i,j,k,l,m : integer;
242 count1,count2 : real;
243 k21,k22,k23,k24,k25,k26,a21,a22,b20,b22 : real;
244 y20,y21,y22,low2_c1,low2_c2,low2_y,vout2,vgate2,gain2 : real;
245 agc2 : array[0..6] of real;
246 w0,w1,v_osc1,v_osc2 : real;
247 va_osc1,va_osc2,vb_osc1,vb_osc2 : real;
248 status : char;
249 w_lock,x_cor,t_v1,t_v2 : real;
250 tn : real;
251
252 Function vin1 : real;
253 begin
254   vin1 := cop * vout2;
255 end;
256
257 Function vin2 : real;
258 begin
259   vin2 := cop2 * vout;
260 end;
261
262 Function rnd : real;
263 var s,t : real;
264
265 p : integer;
266 begin
267   s := random;
268   t := frac(10 * s) * 10;
269   p := trunc(t);
270   if odd(p) then s := -s;
271   rnd := s * 1e-3;
272 end;
273
274 Procedure osc_parameter;
275 begin
276   {Oscillator simulation factor}
277   k11 := 2 * w0 * delta_t;
278   k12 := sqrt(delta_t * w0) * q;
279   k13 := 4 * q + k11 + k12;
280   k14 := -8 * q + 2 * k12;
281   k15 := 4 * q - k11 + k12;
282   k16 := 2 * w0 * delta_t / k13;
283   a11 := -k14 / k13;
284   a12 := -k15 / k13;
285   b10 := 1.0;
286   b12 := -1.0;
287
288   k21 := 2 * w1 * delta_t;
289   k22 := sqrt(delta_t * w1) * q2;
290   k23 := 4 * q2 + k21 + k22;
291   k24 := -8 * q2 + 2 * k22;
292   k25 := 4 * q2 - k21 + k22;
293   k26 := 2 * w1 * delta_t / k23;
294   a21 := -k24 / k23;
295   a22 := -k25 / k23;
296   b20 := 1.0;
297   b22 := -1.0;
298 end;
299
300 {Procedure drawframe;
301 begin
302   drawline(0,5,2000,5);
303   drawline(0,-5,2000,-5);
304 end;
305
306 procedure draw_x;
307 begin
308   drawpoint(x_cor,vout);
309   drawpoint(x_cor,vout2);
310   x_cor := x_cor + 1;
311   if x_cor > 2000 then
312     begin
313       x_cor := 0;
314       clearsreen;
315     end;
316   end;
317 }
318
319 (*p103;
320 [This program is used for simulate INJECTION LOCKING phenomenon
321 of two oscillator that has nearly frequency
322 this program calculate lock frequency]
323 Program Injection_locking;
324
325 (*
326 {SI typedef.sys}
327 {SI graphix.sys}
328 {SI kernel.sys}
329 *)
330 Const pi = 3.1415927; {pi}
331 n = 250; {sampling per period}
332 cop = 0.2; {coupling factor varies}
333 tao1 = 6.28; {Time constant of lowpass1 = k / w0;}
334 loss = 0.1; {loss in system}
335 cop2 = 0.2; {coupling factor2 fixed}
336 tao2 = 6.28;
337 loss2 = 0.1;
338 q = 0.93;
339 w_start = 0.86;
340 w_max = 1.20;
341 qs = 0.93;
342
343 Var r1a,r2a,q2,delta_t,k11,k12,k13,k14,k15,k16,a11,a12,b10,b12 : re
344
345 y10,y11,y12,low_c1,low_c2,low_y,step_i,vout,vgate,gain : real;
346 agc : array[0..6] of real;
347 i,j,k,l,m : integer;
348 count1,count2 : real;
349 k21,k22,k23,k24,k25,k26,a21,a22,b20,b22 : real;
350 y20,y21,y22,low2_c1,low2_c2,low2_y,vout2,vgate2,gain2 : real;
351 agc2 : array[0..6] of real;
352 w0,w1,v_osc1,v_osc2 : real;
353 va_osc1,va_osc2,vb_osc1,vb_osc2 : real;
354 status : char;
355 w_lock,x_cor,t_v1,t_v2 : real;
356 tn : real;
357
358 Function vin1 : real;
359 begin
360   vin1 := cop * vout2;
361 end;
362
363 Function vin2 : real;
364 begin
365   vin2 := cop2 * vout;
366 end;
367
368 Function rnd : real;
369 var s,t : real;
370
371 p : integer;
372 begin
373   s := random;
374   t := frac(10 * s) * 10;
375   p := trunc(t);
376   if odd(p) then s := -s;
377   rnd := s * 1e-3;
378 end;
379
380 Procedure osc_parameter;
381 begin
382   {Oscillator simulation factor}
383   k11 := 2 * w0 * delta_t;
384   k12 := sqrt(delta_t * w0) * q;
385   k13 := 4 * q + k11 + k12;
386   k14 := -8 * q + 2 * k12;
387   k15 := 4 * q - k11 + k12;
388   k16 := 2 * w0 * delta_t / k13;
389   a11 := -k14 / k13;
390   a12 := -k15 / k13;
391   b10 := 1.0;
392   b12 := -1.0;
393
394   k21 := 2 * w1 * delta_t;
395   k22 := sqrt(delta_t * w1) * q2;
396   k23 := 4 * q2 + k21 + k22;
397   k24 := -8 * q2 + 2 * k22;
398   k25 := 4 * q2 - k21 + k22;
399   k26 := 2 * w1 * delta_t / k23;
400   a21 := -k24 / k23;
401   a22 := -k25 / k23;
402   b20 := 1.0;
403   b22 := -1.0;
404 end;
405
406 {Procedure drawframe;
407 begin
408   drawline(0,5,2000,5);
409   drawline(0,-5,2000,-5);
410 end;
411
412 procedure draw_x;
413 begin
414   drawpoint(x_cor,vout);
415   drawpoint(x_cor,vout2);
416   x_cor := x_cor + 1;
417   if x_cor > 2000 then
418     begin
419       x_cor := 0;
420       clearsreen;
421     end;
422   end;
423 }
424
425 (*p103;
426 [This program is used for simulate INJECTION LOCKING phenomenon
427 of two oscillator that has nearly frequency
428 this program calculate lock frequency]
429 Program Injection_locking;
430
431 (*
432 {SI typedef.sys}
433 {SI graphix.sys}
434 {SI kernel.sys}
435 *)
436 Const pi = 3.1415927; {pi}
437 n = 250; {sampling per period}
438 cop = 0.2; {coupling factor varies}
439 tao1 = 6.28; {Time constant of lowpass1 = k / w0;}
440 loss = 0.1; {loss in system}
441 cop2 = 0.2; {coupling factor2 fixed}
442 tao2 = 6.28;
443 loss2 = 0.1;
444 q = 0.93;
445 w_start = 0.86;
446 w_max = 1.20;
447 qs = 0.93;
448
449 Var r1a,r2a,q2,delta_t,k11,k12,k13,k14,k15,k16,a11,a12,b10,b12 : re
450
451 y10,y11,y12,low_c1,low_c2,low_y,step_i,vout,vgate,gain : real;
452 agc : array[0..6] of real;
453 i,j,k,l,m : integer;
454 count1,count2 : real;
455 k21,k22,k23,k24,k25,k26,a21,a22,b20,b22 : real;
456 y20,y21,y22,low2_c1,low2_c2,low2_y,vout2,vgate2,gain2 : real;
457 agc2 : array[0..6] of real;
458 w0,w1,v_osc1,v_osc2 : real;
459 va_osc1,va_osc2,vb_osc1,vb_osc2 : real;
460 status : char;
461 w_lock,x_cor,t_v1,t_v2 : real;
462 tn : real;
463
464 Function vin1 : real;
465 begin
466   vin1 := cop * vout2;
467 end;
468
469 Function vin2 : real;
470 begin
471   vin2 := cop2 * vout;
472 end;
473
474 Function rnd : real;
475 var s,t : real;
476
477 p : integer;
478 begin
479   s := random;
480   t := frac(10 * s) * 10;
481   p := trunc(t);
482   if odd(p) then s := -s;
483   rnd := s * 1e-3;
484 end;
485
486 Procedure osc_parameter;
487 begin
488   {Oscillator simulation factor}
489   k11 := 2 * w0 * delta_t;
490   k12 := sqrt(delta_t * w0) * q;
491   k13 := 4 * q + k11 + k12;
492   k14 := -8 * q + 2 * k12;
493   k15 := 4 * q - k11 + k12;
494   k16 := 2 * w0 * delta_t / k13;
495   a11 := -k14 / k13;
496   a12 := -k15 / k13;
497   b10 := 1.0;
498   b12 := -1.0;
499
500   k21 := 2 * w1 * delta_t;
501   k22 := sqrt(delta_t * w1) * q2;
502   k23 := 4 * q2 + k21 + k22;
503   k24 := -8 * q2 + 2 * k22;
504   k25 := 4 * q2 - k21 + k22;
505   k26 := 2 * w1 * delta_t / k23;
506   a21 := -k24 / k23;
507   a22 := -k25 / k23;
508   b20 := 1.0;
509   b22 := -1.0;
510 end;
511
512 {Procedure drawframe;
513 begin
514   drawline(0,5,2000,5);
515   drawline(0,-5,2000,-5);
516 end;
517
518 procedure draw_x;
519 begin
520   drawpoint(x_cor,vout);
521   drawpoint(x_cor,vout2);
522   x_cor := x_cor + 1;
523   if x_cor > 2000 then
524     begin
525       x_cor := 0;
526       clearsreen;
527     end;
528   end;
529 }
530
531 (*p103;
532 [This program is used for simulate INJECTION LOCKING phenomenon
533 of two oscillator that has nearly frequency
534 this program calculate lock frequency]
535 Program Injection_locking;
536
537 (*
538 {SI typedef.sys}
539 {SI graphix.sys}
540 {SI kernel.sys}
541 *)
542 Const pi = 3.1415927; {pi}
543 n = 250; {sampling per period}
544 cop = 0.2; {coupling factor varies}
545 tao1 = 6.28; {Time constant of lowpass1 = k / w0;}
546 loss = 0.1; {loss in system}
547 cop2 = 0.2; {coupling factor2 fixed}
548 tao2 = 6.28;
549 loss2 = 0.1;
550 q = 0.93;
551 w_start = 0.86;
552 w_max = 1.20;
553 qs = 0.93;
554
555 Var r1a,r2a,q2,delta_t,k11,k12,k13,k14,k15,k16,a11,a12,b10,b12 : re
556
557 y10,y11,y12,low_c1,low_c2,low_y,step_i,vout,vgate,gain : real;
558 agc : array[0..6] of real;
559 i,j,k,l,m : integer;
560 count1,count2 : real;
561 k21,k22,k23,k24,k25,k26,a21,a22,b20,b22 : real;
562 y20,y21,y22,low2_c1,low2_c2,low2_y,vout2,vgate2,gain2 : real;
563 agc2 : array[0..6] of real;
564 w0,w1,v_osc1,v_osc2 : real;
565 va_osc1,va_osc2,vb_osc1,vb_osc2 : real;
566 status : char;
567 w_lock,x_cor,t_v1,t_v2 : real;
568 tn : real;
569
570 Function vin1 : real;
571 begin
572   vin1 := cop * vout2;
573 end;
574
575 Function vin2 : real;
576 begin
577   vin2 := cop2 * vout;
578 end;
579
580 Function rnd : real;
581 var s,t : real;
582
583 p : integer;
584 begin
585   s := random;
586   t := frac(10 * s) * 10;
587   p := trunc(t);
588   if odd(p) then s := -s;
589   rnd := s * 1e-3;
590 end;
591
592 Procedure osc_parameter;
593 begin
594   {Oscillator simulation factor}
595   k11 := 2 * w0 * delta_t;
596   k12 := sqrt(delta_t * w0) * q;
597   k13 := 4 * q + k11 + k12;
598   k14 := -8 * q + 2 * k12;
599   k15 := 4 * q - k11 + k12;
600   k16 := 2 * w0 * delta_t / k13;
601   a11 := -k14 / k13;
602   a12 := -k15 / k13;
603   b10 := 1.0;
604   b12 := -1.0;
605
606   k21 := 2 * w1 * delta_t;
607   k22 := sqrt(delta_t * w1) * q2;
608   k23 := 4 * q2 + k21 + k22;
609   k24 := -8 * q2 + 2 * k22;
610   k25 := 4 * q2 - k21 + k22;
611   k26 := 2 * w1 * delta_t / k23;
612   a21 := -k24 / k23;
613   a22 := -k25 / k23;
614   b20 := 1.0;
615   b22 := -1.0;
616 end;
617
618 {Procedure drawframe;
619 begin
620   drawline(0,5,2000,5);
621   drawline(0,-5,2000,-5);
622 end;
623
624 procedure draw_x;
625 begin
626   drawpoint(x_cor,vout);
627   drawpoint(x_cor,vout2);
628   x_cor := x_cor + 1;
629   if x_cor > 2000 then
630     begin
631       x_cor := 0;
632       clearsreen;
633     end;
634   end;
635 }
636
637 (*p103;
638 [This program is used for simulate INJECTION LOCKING phenomenon
639 of two oscillator that has nearly frequency
640 this program calculate lock frequency]
641 Program Injection_locking;
642
643 (*
644 {SI typedef.sys}
645 {SI graphix.sys}
646 {SI kernel.sys}
647 *)
648 Const pi = 3.1415927; {pi}
649 n = 250; {sampling per period}
650 cop = 0.2; {coupling factor varies}
651 tao1 = 6.28; {Time constant of lowpass1 = k / w0;}
652 loss = 0.1; {loss in system}
653 cop2 = 0.2; {coupling factor2 fixed}
654 tao2 = 6.28;
655 loss2 = 0.1;
656 q = 0.93;
657 w_start = 0.86;
658 w_max = 1.20;
659 qs = 0.93;
660
661 Var r1a,r2a,q2,delta_t,k11,k12,k13,k14,k15,k16,a11,a12,b10,b12 : re
662
663 y10,y11,y12,low_c1,low_c2,low_y,step_i,vout,vgate,gain : real;
664 agc : array[0..6] of real;
665 i,j,k,l,m : integer;
666 count1,count2 : real;
667 k21,k22,k23,k24,k25,k26,a21,a22,b20,b22 : real;
668 y20,y21,y22,low2_c1,low2_c2,low2_y,vout2,vgate2,gain2 : real;
669 agc2 : array[0..6] of real;
670 w0,w1,v_osc1,v_osc2 : real;
671 va_osc1,va_osc2,vb_osc1,vb_osc2 : real;
672 status : char;
673 w_lock,x_cor,t_v1,t_v2 : real;
674 tn : real;
675
676 Function vin1 : real;
677 begin
678   vin1 := cop * vout2;
679 end;
680
677 Function vin2 : real;
678 begin
679   vin2 := cop2 * vout;
680 end;
681
682 Function rnd : real;
683 var s,t : real;
684
685 p : integer;
686 begin
687   s := random;
688   t := frac(10 * s) * 10;
689   p := trunc(t);
690   if odd(p) then s := -s;
691   rnd := s * 1e-3;
692 end;
693
694 Procedure osc_parameter;
695 begin
696   {Oscillator simulation factor}
697   k11 := 2 * w0 * delta_t;
698   k12 := sqrt(delta_t * w0) * q;
699   k13 := 4 * q + k11 + k12;
700   k14 := -8 * q + 2 * k12;
701   k15 := 4 * q - k11 + k12;
702   k16 := 2 * w0 * delta_t / k13;
703   a11 := -k14 / k13;
704   a12 := -k15 / k13;
705   b10 := 1.0;
706   b12 := -1.0;
707
708   k21 := 2 * w1 * delta_t;
709   k22 := sqrt(delta_t * w1) * q2;
710   k23 := 4 * q2 + k21 + k22;
711   k24 := -8 * q2 + 2 * k22;
712   k25 := 4 * q2 - k21 + k22;
713   k26 := 2 * w1 * delta_t / k23;
714   a21 := -k24 / k23;
715   a22 := -k25 / k23;
716   b20 := 1.0;
717   b22 := -1.0;
718 end;
719
720 {Procedure drawframe;
721 begin
722   drawline(0,5,2000,5);
723   drawline(0,-5,2000,-5);
724 end;
725
726 procedure draw_x;
727 begin
728   drawpoint(x_cor,vout);
729   drawpoint(x_cor,vout2);
730   x_cor := x_cor + 1;
731   if x_cor > 2000 then
732     begin
733       x_cor := 0;
734       clearsreen;
735     end;
736   end;
737 }
738
739 (*p103;
740 [This program is used for simulate INJECTION LOCKING phenomenon
741 of two oscillator that has nearly frequency
742 this program calculate lock frequency]
743 Program Injection_locking;
744
745 (*
746 {SI typedef.sys}
747 {SI graphix.sys}
748 {SI kernel.sys}
749 *)
750 Const pi = 3.1415927; {pi}
751 n = 250; {sampling per period}
752 cop = 0.2; {coupling factor varies}
753 tao1 = 6.28; {Time constant of lowpass1 = k / w0;}
754 loss = 0.1; {loss in system}
755 cop2 = 0.2; {coupling factor2 fixed}
756 tao2 = 6.28;
757 loss2 = 0.1;
758 q = 0.93;
759 w_start = 0.86;
760 w_max = 1.20;
761 qs = 0.93;
762
763 Var r1a,r2a,q2,delta_t,k11,k12,k13,k14,k15,k16,a11,a12,b10,b12 : re
764
765 y10,y11,y12,low_c1,low_c2,low_y,step_i,vout,vgate,gain : real;
766 agc : array[0..6] of real;
767 i,j,k,l,m : integer;
768 count1,count2 : real;
769 k21,k22,k23,k24,k25,k26,a21,a22,b20,b22 : real;
770 y20,y21,y22,low2_c1,low2_c2,low2_y,vout2,vgate2,gain2 : real;
771 agc2 : array[0..6] of real;
772 w0,w1,v_osc1,v_osc2 : real;
773 va_osc1,va_osc2,vb_osc1,vb_osc2 : real;
774 status : char;
775 w_lock,x_cor,t_v1,t_v2 : real;
776 tn : real;
777
778 Function vin1 : real;
779 begin
780   vin1 := cop * vout2;
781 end;
782
783 Function vin2 : real;
784 begin
785   vin2 := cop2 * vout;
786 end;
787
788 Function rnd : real;
789 var s,t : real;
790
791 p : integer;
792 begin
793   s := random;
794   t := frac(10 * s) * 10;
795   p := trunc(t);
796   if odd(p) then s := -s;
797   rnd := s * 1e-3;
798 end;
799
800 Procedure osc_parameter;
801 begin
802   {Oscillator simulation factor}
803   k11 := 2 * w0 * delta_t;
804   k12 := sqrt(delta_t * w0) * q;
805   k13 := 4 * q + k11 + k12;
806   k14 := -8 * q + 2 * k12;
807   k15 := 4 * q - k11 + k12;
808   k16 := 2 * w0 * delta_t / k13;
809   a11 := -k14 / k13;
810   a12 := -k15 / k13;
811   b10 := 1.0;
812   b12 := -1.0;
813
814   k21 := 2 * w1 * delta_t;
815   k22 := sqrt(delta_t * w1) * q2;
816   k23 := 4 * q2 + k21 + k22;
817   k24 := -8 * q2 + 2 * k22;
818   k25 := 4 * q2 - k21 + k22;
819   k26 := 2 * w1 * delta_t / k23;
820   a21 := -k24 / k23;
821   a22 := -k25 / k23;
822   b20 := 1.0;
823   b22 := -1.0;
824 end;
825
826 {Procedure drawframe;
827 begin
828   drawline(0,5,2000,5);
829   drawline(0,-5,2000,-5);
830 end;
831
832 procedure draw_x;
833 begin
834   drawpoint(x_cor,vout);
835   drawpoint(x_cor,vout2);
836   x_cor := x_cor + 1;
837   if x_cor > 2000 then
838     begin
839       x_cor := 0;
840       clearsreen;
841     end;
842   end;
843 }
844
845 (*p103;
846 [This program is used for simulate INJECTION LOCKING phenomenon
847 of two oscillator that has nearly frequency
848 this program calculate lock frequency]
849 Program Injection_locking;
850
851 (*
852 {SI typedef.sys}
853 {SI graphix.sys}
854 {SI kernel.sys}
855 *)
856 Const pi = 3.1415927; {pi}
857 n = 250; {sampling per period}
858 cop = 0.2; {coupling factor varies}
859 tao1 = 6.28; {Time constant of lowpass1 = k / w0;}
860 loss = 0.1; {loss in system}
861 cop2 = 0.2; {coupling factor2 fixed}
862 tao2 = 6.28;
863 loss2 = 0.1;
864 q = 0.93;
865 w_start = 0.86;
866 w_max = 1.20;
867 qs = 0.93;
868
869 Var r1a,r2a,q2,delta_t,k11,k12,k13,k14,k15,k16,a11,a12,b10,b12 : re
870
871 y10,y11,y12,low_c1,low_c2,low_y,step_i,vout,vgate,gain : real;
872 agc : array[0..6] of real;
873 i,j,k,l,m : integer;
874 count1,count2 : real;
875 k21,k22,k23,k24,k25,k26,a21,a22,b20,b22 : real;
876 y20,y21,y22,low2_c1,low2_c2,low2_y,vout2,vgate2,gain2 : real;
877 agc2 : array[0..6] of real;
878 w0,w1,v_osc1,v_osc2 : real;
879 va_osc1,va_osc2,vb_osc1,vb_osc2 : real;
880 status : char;
881 w_lock,x_cor,t_v1,t_v2 : real;
882 tn : real;
883
884 Function vin1 : real;
885 begin
886   vin1 := cop * vout2;
887 end;
888
889 Function vin2 : real;
890 begin
891   vin2 := cop2 * vout;
892 end;
893
894 Function rnd : real;
895 var s,t : real;
896
897 p : integer;
898 begin
899   s := random;
900   t := frac(10 * s) * 10;
901   p := trunc(t);
902   if odd(p) then s := -s;
903   rnd := s * 1e-3;
904 end;
905
906 Procedure osc_parameter;
907 begin
908   {Oscillator simulation factor}
909   k11 := 2 * w0 * delta_t;
910   k12 := sqrt(delta_t * w0) * q;
911   k13 := 4 * q + k11 + k12;
912   k14 := -8 * q + 2 * k12;
913   k15 := 4 * q - k11 + k12;
914   k16 := 2 * w0 * delta_t / k13;
915   a11 := -k14 / k13;
916   a12 := -k15 / k13;
917   b10 := 1.0;
918   b12 := -1.0;
919
920   k21 := 2 * w1 * delta_t;
921   k22 := sqrt(delta_t * w1) * q2;
922   k23 := 4 * q2 + k21 + k22;
923   k24 := -8 * q2 + 2 * k22;
924   k25 := 4 * q2 - k21 + k22;
925   k26 := 2 * w1 * delta_t / k23;
926   a21 := -k24 / k23;
927   a22 := -k25 / k23;
928   b20 := 1.0;
929   b22 := -1.0;
930 end;
931
932 {Procedure drawframe;
933 begin
934   drawline(0,5,2000,5);
935   drawline(0,-5,2000,-5);
936 end;
937
938 procedure draw_x;
939 begin
940   drawpoint(x_cor,vout);
941   drawpoint(x_cor,vout2);
942   x_cor := x_cor + 1;
943   if x_cor > 2000 then
944     begin
945       x_cor := 0;
946       clearsreen;
947     end;
948   end;
949 }
950
951 (*p103;
```

```

t_v1 := vgate;
t_v2 := vgate2;
for i := 1 to 6 do
begin
    gain := gain + vgate * agc[i];
    vgate := vgate * t_v1;
    gain2 := gain2 + vgate2 * agc2[i];
    vgate2 := vgate2 * t_v2;
end;
(writeln(vout:20,low_y:25,gain:30));

if abs(vout) > v_osc1 then v_osc1 := abs(vout);
if abs(vout2) > v_osc2 then v_osc2 := abs(vout2);

{multiply vout by gain and loss}
{draw_x;}
vout := vout * loss;
vout2 := vout2 * loss2;
j := j + 1;
if j < 2000 then goto start1;
k := k + 1;
if k <= 18 then
begin
    v_osc1 := 0.0;
    v_osc2 := 0.0;
end;
if k <= 20 then
begin
    {clearscreen;
    drawframe;}
    j := 0;
    goto start1;
end;
(writeln('v_osc1 = ',v_osc1:20);
writeln('v_osc2 = ',v_osc2:20);
writeln('v_osc2 = ',v_osc2:20);)
{find peak value}
v_osc1 := 0.0;
for l := 0 to n do
begin
    y12 := a11 * y10 + a12 * y11 + vin1 + vout;
    y22 := a21 * y20 + a22 * y21 + vin2 + vout2;
    vout := k16 * (b10 * y12 + y11 * b12);
    y11 := y10;
    y10 := y12;
    vout := vout * 2 * gain;

    vout2 := k26 * (b20 * y22 + y21 * b22);
    y21 := y20;
    y20 := y22;
    vout2 := vout2 * 2 * gain2;

    {Full wave rectifier}
    vgate := -abs(vout);
    vgate2 := -abs(vout2);

    {lowpass filter}
    vgate := vgate * low_c1 + low_y * low_c2;
    low_y := vgate;
    vgate := abs(vgate / 2.0295);

    vgate2 := vgate2 * low2_c1 + low2_y * low2_c2;
    low2_y := vgate2;
    vgate2 := abs(vgate2 / 2.995);

    {gain execute}
    gain := agc[0];
    gain2 := agc2[0];
    t_v1 := vgate;
    t_v2 := vgate2;
    for i := 1 to 6 do
    begin
        gain := gain + vgate * agc[i];
        vgate := vgate * t_v1;
        gain2 := gain2 + vgate2 * agc2[i];
        vgate2 := vgate2 * t_v2;
    end;
    if abs(vout) > v_osc1 then v_osc1 := abs(vout);

    va_osc1 := vb_osc1;
    vb_osc1 := vout;
    {multiply vout by gain and loss}
    {draw_x;}
    vout := vout * loss;
    vout2 := vout2 * loss2;
end;
(writeln('peak = ',v_osc1);
writeln('peak = ',v_osc1));
count1 := 0.0;
count2 := 0;
{find Zero crossing}
lock2: y12 := a11 * y10 + a12 * y11 + vin1 + vout;
y22 := a21 * y20 + a22 * y21 + vin2 + vout2;
vout := k16 * (b10 * y12 + y11 * b12);
y11 := y10;
y10 := y12;
vout := vout * 2 * gain;

vout2 := k26 * (b20 * y22 + y21 * b22);

```

```

y21 := y20;
vout2 := vout2 * 2 * gain2;

{Full wave rectifier}
vgate := -abs(vout);
vgate2 := -abs(vout2);

{lowpass filter}
vgate := vgate * low_c1 + low_y * low_c2;
low_y := vgate;
vgate := abs(vgate / 2.0295);

vgate2 := vgate2 * low2_c1 + low2_y * low2_c2;
low2_y := vgate2;
vgate2 := abs(vgate2 / 2.995);

{gain execute}
gain := agc[0];
gain2 := agc2[0];
t_v1 := vgate;
t_v2 := vgate2;
for i := 1 to 6 do
begin
    gain := gain + vgate * agc[i];
    vgate := vgate * t_v1;
    gain2 := gain2 + vgate2 * agc2[i];
    vgate2 := vgate2 * t_v2;
end;
{multiply vout by gain and loss}
{draw_x;}
va_osc1 := vb_osc1;
vb_osc1 := vout;
vout := vout * loss;
vout2 := vout2 * loss2;
if (va_osc1 < 0) and (vb_osc1 > 0) then goto s
else goto lock2;
start2: status := 'y';
repeat
    y12 := a11 * y10 + a12 * y11 + vin1 + vout;
    y22 := a21 * y20 + a22 * y21 + vin2 + vout;
    vout := k16 * (b10 * y12 + y11 * b12);
    y11 := y10;
    y10 := y12;
    vout := vout * 2 * gain;

    vout2 := k26 * (b20 * y22 + y21 * b22);
    y21 := y20;
    y20 := y22;
    vout2 := vout2 * 2 * gain2;

    {Full wave rectifier}
    vgate := -abs(vout);
    vgate2 := -abs(vout2);

    {lowpass filter}
    vgate := vgate * low_c1 + low_y * low_c2;
    low_y := vgate;
    vgate := abs(vgate / 2.0295);

    vgate2 := vgate2 * low2_c1 + low2_y * low2_c2;
    low2_y := vgate2;
    vgate2 := abs(vgate2 / 2.995);

    {gain execute}
    gain := agc[0];
    gain2 := agc2[0];
    t_v1 := vgate;
    t_v2 := vgate2;
    for i := 1 to 6 do
    begin
        gain := gain + vgate * agc[i];
        vgate := vgate * t_v1;
        gain2 := gain2 + vgate2 * agc2[i];
        vgate2 := vgate2 * t_v2;
    end;
    if ((vb_osc1 - va_osc1) * (vb_osc1 - vout)) > 0 then
    begin
        (writeln('','lock peak check value = ',v
        writeln('','w = ',w1:10,' lock peak che
        value = ',vb_osc1);)
        if (abs(vb_osc1) > 1.003 * v_osc1) or (abs(vb_osc1) < 0.997 * v_osc1)
        then status := 'n';
    end;
    va_osc1 := vb_osc1;
    vb_osc1 := vout;
    count1 := count1 + 1.0;
    if (va_osc1 < 0) and (vb_osc1 > 0) then count2 :=
    until count2 = 100;
    {multiply vout by gain and loss}
    {draw_x;}
    vout := vout * loss;
    vout2 := vout2 * loss2;
    tn := n * 100 / count1;
    until count2 = 100;
    (writeln('lock frequency = ',w1,'w = ':10,tn:20);)
    (writeln('','q2 = ',q2:20,'coupling = ':15,cop);)
    writeln('','q2 = ',q2:20,'coupling = ':15,cop);
    if status = 'n' then writeln('no');
    goto lock;
    final: (writeln('w = ',w1);)
    (readln(i);)
    leavegraphic;
end.

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการ
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

1 of two oscillators that has nearly frequency)
2 Program Injection_locking;
3
4
5 {SI typedef.sys)
6 {SI graphix.sys)
7 {SI kernel.sys)
8
9 Const w0 = 1.0;      {locked frequency)
10 w1 = 1.090; {lock frequency)
11 pi = 3.1415927; {pi)
12 n = 250; {sampling per period) *
13 cop = 0.1; {coupling factor)
14 taol = 6.28; {Time constant of lowpass1 = k / w0;})
15 loss = 0.1; {loss in system)
16
17 Var delta_t,q,k11,k12,k13,k14,k15,k16,a11,a12,b10,b12 : real;
18 y10,y11,y12,low_c1,low_c2,low_y,step_i,vout,vgate,gain : real;
19 agc : array[0..3] of real;
20 i,j : integer;
21 last_y,count1,count2,count3 : real;
22
23 Function vin : real;
24 begin
25   vin := cop * sin(w1 * delta_t * step_i);
26   step_i := step_i + 1;
27 end;
28
29 Function rnd : real;
30 var s,t : real;
31 p : integer;
32 begin
33   s := random;
34   t := frac(10 * s) * 10;
35   p := trunc(t);
36   if odd(p) then s := -s;
37   rnd := s * 1e-3;
38 end;
39
40 {main program)
41 label start1,start2,start3,start4,final,lock;
42 BEGIN
43   InitGraphic;
44   DefineWorld(1,0,-10,2000,10);
45   Definewindow(1,0,0,xmaxglb,ymaxglb);
46   SelectWorld(1);
47   selectwindow(1);
48
49   Delta_t := 2 * pi / w0 / n; {delta time)
50   q := 1.0;
51
52   {Oscillator simulation factor)
53   k11 := 2 * w0 * delta_t;
54   k12 := sqrt(delta_t * w0) * q;
55   k13 := 4 * q + k11 + k12;
56   k14 := -8 * q + 2 * k12;
57   k15 := 4 * q - k11 + k12;
58   k16 := 2 * w0 * delta_t / k13;
59   a11 := -k14 / k13;
60   a12 := -k15 / k13;
61   b10 := 1.0;
62   b12 := -1.0;
63   y10 := 0.0;
64   y11 := 0.0;
65
66   {Lowpass filter parameter)
67   low_c1 := delta_t * exp(-delta_t / taol / 2) / taol;
68   low_c2 := exp(-delta_t / taol);
69   low_y := 0.0;
70
71   {Agc parameter)
72   agc[0] := 7.35;
73   agc[1] := 0.87;
74   agc[2] := -0.14;
75   agc[3] := -0.02;
76
77   step_i := 0;
78   j := 0;
79   vout := 0.0;
80   low_y := 0.0;
81   gain := agc[0];
82
83   start1: {Bandpass filter execute)
84     y12 := a11 * y10 + a12 * y11 + rnd + vout;
85     vout := k16 * (b10 * y12 + y11 * b12);
86     y11 := y10;
87     y10 := y12;
88     vout := vout * 2 * gain;
89
90     {Full wave rectifier)
91     vgate := -abs(vout);
92
93     {lowpass filter)
94     vgate := vgate * low_c1 + low_y * low_c2;
95     low_y := vgate;
96     vgate := vgate / 2.0;
97
98     {gain execute)
99     gain := agc[0];
100
101   start2: {Bandpass filter execute)
102     y12 := a11 * y10 + a12 * y11 + vin + vout;
103     vout := k16 * (b10 * y12 + y11 * b12);
104     y11 := y10;
105     y10 := y12;
106     vout := vout * 2 * gain;
107
108     {Full wave rectifier)
109     vgate := -abs(vout);
110
111     {lowpass filter)
112     vgate := vgate * low_c1 + low_y * low_c2;
113     low_y := vgate;
114     vgate := vgate / 2.0;
115
116     {gain execute)
117     gain := agc[0];
118     for i := 1 to 3 do
119       begin
120         gain := gain + vgate * agc[i];
121         vgate := vgate * low_y;
122       end;
123     end;
124     {multiply vout by gain and loss)
125     vout := vout * loss;
126     if (last_y > 0) and (vout < 0) then goto start3
127     else
128       begin
129         last_y := vout;
130         goto start2;
131       end;
132   end;
133
134   start3: last_y := vout;
135   count1 := 0.0;
136   count2 := 0.0;
137
138   start4: {Bandpass filter execute)
139     y12 := a11 * y10 + a12 * y11 + vin + vout;
140     vout := k16 * (b10 * y12 + y11 * b12);
141     y11 := y10;
142     y10 := y12;
143     vout := vout * 2 * gain;
144
145     {Full wave rectifier)
146     vgate := -abs(vout);
147
148     {lowpass filter)
149     vgate := vgate * low_c1 + low_y * low_c2;
150     low_y := vgate;
151     vgate := vgate / 2.0;
152
153     {gain execute)
154     gain := agc[0];
155     for i := 1 to 3 do
156       begin
157         gain := gain + vgate * agc[i];
158         vgate := vgate * low_y;
159       end;
160     end;
161     {multiply vout by gain and loss)
162     vout := vout * loss;
163     if (last_y > 0) and (vout < 0) then goto start3
164     else
165       begin
166         last_y := vout;
167         goto start2;
168       end;
169   end;
170
171   start5: {Bandpass filter execute)
172     y12 := a11 * y10 + a12 * y11 + vin + vout;
173     vout := k16 * (b10 * y12 + y11 * b12);
174     y11 := y10;
175     y10 := y12;
176     vout := vout * 2 * gain;
177
178     {Full wave rectifier)
179     vgate := -abs(vout);
180
181     {lowpass filter)
182     vgate := vgate * low_c1 + low_y * low_c2;
183     low_y := vgate;
184     vgate := vgate / 2.0;
185
186     {gain execute)
187     gain := agc[0];
188     for i := 1 to 3 do
189       begin
190         gain := gain + vgate * agc[i];
191         vgate := vgate * low_y;
192       end;
193     end;
194     {multiply vout by gain and loss)
195     vout := vout * loss;
196     if (last_y > 0) and (vout < 0) then goto start3
197     else
198       begin
199         last_y := vout;
200         goto start2;
201       end;
202   end;
203
204   start6: {Bandpass filter execute)
205     y12 := a11 * y10 + a12 * y11 + vin + vout;
206     vout := k16 * (b10 * y12 + y11 * b12);
207     y11 := y10;
208     y10 := y12;
209     vout := vout * 2 * gain;
210
211     {Full wave rectifier)
212     vgate := -abs(vout);
213
214     {lowpass filter)
215     vgate := vgate * low_c1 + low_y * low_c2;
216     low_y := vgate;
217     vgate := vgate / 2.0;
218
219     {gain execute)
220     gain := agc[0];
221     for i := 1 to 3 do
222       begin
223         gain := gain + vgate * agc[i];
224         vgate := vgate * low_y;
225       end;
226     end;
227     {multiply vout by gain and loss)
228     vout := vout * loss;
229     if (last_y > 0) and (vout < 0) then goto start3
230     else
231       begin
232         last_y := vout;
233         goto start2;
234       end;
235   end;
236
237   start7: {Bandpass filter execute)
238     y12 := a11 * y10 + a12 * y11 + vin + vout;
239     vout := k16 * (b10 * y12 + y11 * b12);
240     y11 := y10;
241     y10 := y12;
242     vout := vout * 2 * gain;
243
244     {Full wave rectifier)
245     vgate := -abs(vout);
246
247     {lowpass filter)
248     vgate := vgate * low_c1 + low_y * low_c2;
249     low_y := vgate;
250     vgate := vgate / 2.0;
251
252     {gain execute)
253     gain := agc[0];
254     for i := 1 to 3 do
255       begin
256         gain := gain + vgate * agc[i];
257         vgate := vgate * low_y;
258       end;
259     end;
260     {multiply vout by gain and loss)
261     vout := vout * loss;
262     if (last_y > 0) and (vout < 0) then goto start3
263     else
264       begin
265         last_y := vout;
266         goto start2;
267       end;
268   end;
269
270   start8: {Bandpass filter execute)
271     y12 := a11 * y10 + a12 * y11 + vin + vout;
272     vout := k16 * (b10 * y12 + y11 * b12);
273     y11 := y10;
274     y10 := y12;
275     vout := vout * 2 * gain;
276
277     {Full wave rectifier)
278     vgate := -abs(vout);
279
280     {lowpass filter)
281     vgate := vgate * low_c1 + low_y * low_c2;
282     low_y := vgate;
283     vgate := vgate / 2.0;
284
285     {gain execute)
286     gain := agc[0];
287     for i := 1 to 3 do
288       begin
289         gain := gain + vgate * agc[i];
290         vgate := vgate * low_y;
291       end;
292     end;
293     {multiply vout by gain and loss)
294     vout := vout * loss;
295     if (last_y > 0) and (vout < 0) then goto start3
296     else
297       begin
298         last_y := vout;
299         goto start2;
300       end;
301   end;
302
303   start9: {Bandpass filter execute)
304     y12 := a11 * y10 + a12 * y11 + vin + vout;
305     vout := k16 * (b10 * y12 + y11 * b12);
306     y11 := y10;
307     y10 := y12;
308     vout := vout * 2 * gain;
309
310     {Full wave rectifier)
311     vgate := -abs(vout);
312
313     {lowpass filter)
314     vgate := vgate * low_c1 + low_y * low_c2;
315     low_y := vgate;
316     vgate := vgate / 2.0;
317
318     {gain execute)
319     gain := agc[0];
320     for i := 1 to 3 do
321       begin
322         gain := gain + vgate * agc[i];
323         vgate := vgate * low_y;
324       end;
325     end;
326     {multiply vout by gain and loss)
327     vout := vout * loss;
328     if (last_y > 0) and (vout < 0) then goto start3
329     else
330       begin
331         last_y := vout;
332         goto start2;
333       end;
334   end;
335
336   start10: {Bandpass filter execute)
337     y12 := a11 * y10 + a12 * y11 + vin + vout;
338     vout := k16 * (b10 * y12 + y11 * b12);
339     y11 := y10;
340     y10 := y12;
341     vout := vout * 2 * gain;
342
343     {Full wave rectifier)
344     vgate := -abs(vout);
345
346     {lowpass filter)
347     vgate := vgate * low_c1 + low_y * low_c2;
348     low_y := vgate;
349     vgate := vgate / 2.0;
350
351     {gain execute)
352     gain := agc[0];
353     for i := 1 to 3 do
354       begin
355         gain := gain + vgate * agc[i];
356         vgate := vgate * low_y;
357       end;
358     end;
359     {multiply vout by gain and loss)
360     vout := vout * loss;
361     if (last_y > 0) and (vout < 0) then goto start3
362     else
363       begin
364         last_y := vout;
365         goto start2;
366       end;
367   end;
368
369   start11: {Bandpass filter execute)
370     y12 := a11 * y10 + a12 * y11 + vin + vout;
371     vout := k16 * (b10 * y12 + y11 * b12);
372     y11 := y10;
373     y10 := y12;
374     vout := vout * 2 * gain;
375
376     {Full wave rectifier)
377     vgate := -abs(vout);
378
379     {lowpass filter)
380     vgate := vgate * low_c1 + low_y * low_c2;
381     low_y := vgate;
382     vgate := vgate / 2.0;
383
384     {gain execute)
385     gain := agc[0];
386     for i := 1 to 3 do
387       begin
388         gain := gain + vgate * agc[i];
389         vgate := vgate * low_y;
390       end;
391     end;
392     {multiply vout by gain and loss)
393     vout := vout * loss;
394     if (last_y > 0) and (vout < 0) then goto start3
395     else
396       begin
397         last_y := vout;
398         goto start2;
399       end;
400   end;
401
402   start12: {Bandpass filter execute)
403     y12 := a11 * y10 + a12 * y11 + vin + vout;
404     vout := k16 * (b10 * y12 + y11 * b12);
405     y11 := y10;
406     y10 := y12;
407     vout := vout * 2 * gain;
408
409     {Full wave rectifier)
410     vgate := -abs(vout);
411
412     {lowpass filter)
413     vgate := vgate * low_c1 + low_y * low_c2;
414     low_y := vgate;
415     vgate := vgate / 2.0;
416
417     {gain execute)
418     gain := agc[0];
419     for i := 1 to 3 do
420       begin
421         gain := gain + vgate * agc[i];
422         vgate := vgate * low_y;
423       end;
424     end;
425     {multiply vout by gain and loss)
426     vout := vout * loss;
427     if (last_y > 0) and (vout < 0) then goto start3
428     else
429       begin
430         last_y := vout;
431         goto start2;
432       end;
433   end;
434
435   start13: {Bandpass filter execute)
436     y12 := a11 * y10 + a12 * y11 + vin + vout;
437     vout := k16 * (b10 * y12 + y11 * b12);
438     y11 := y10;
439     y10 := y12;
440     vout := vout * 2 * gain;
441
442     {Full wave rectifier)
443     vgate := -abs(vout);
444
445     {lowpass filter)
446     vgate := vgate * low_c1 + low_y * low_c2;
447     low_y := vgate;
448     vgate := vgate / 2.0;
449
450     {gain execute)
451     gain := agc[0];
452     for i := 1 to 3 do
453       begin
454         gain := gain + vgate * agc[i];
455         vgate := vgate * low_y;
456       end;
457     end;
458     {multiply vout by gain and loss)
459     vout := vout * loss;
460     if (last_y > 0) and (vout < 0) then goto start3
461     else
462       begin
463         last_y := vout;
464         goto start2;
465       end;
466   end;
467
468   start14: {Bandpass filter execute)
469     y12 := a11 * y10 + a12 * y11 + vin + vout;
470     vout := k16 * (b10 * y12 + y11 * b12);
471     y11 := y10;
472     y10 := y12;
473     vout := vout * 2 * gain;
474
475     {Full wave rectifier)
476     vgate := -abs(vout);
477
478     {lowpass filter)
479     vgate := vgate * low_c1 + low_y * low_c2;
480     low_y := vgate;
481     vgate := vgate / 2.0;
482
483     {gain execute)
484     gain := agc[0];
485     for i := 1 to 3 do
486       begin
487         gain := gain + vgate * agc[i];
488         vgate := vgate * low_y;
489       end;
490     end;
491     {multiply vout by gain and loss)
492     vout := vout * loss;
493     if (last_y > 0) and (vout < 0) then goto start3
494     else
495       begin
496         last_y := vout;
497         goto start2;
498       end;
499   end;
500
501   start15: {Bandpass filter execute)
502     y12 := a11 * y10 + a12 * y11 + vin + vout;
503     vout := k16 * (b10 * y12 + y11 * b12);
504     y11 := y10;
505     y10 := y12;
506     vout := vout * 2 * gain;
507
508     {Full wave rectifier)
509     vgate := -abs(vout);
510
511     {lowpass filter)
512     vgate := vgate * low_c1 + low_y * low_c2;
513     low_y := vgate;
514     vgate := vgate / 2.0;
515
516     {gain execute)
517     gain := agc[0];
518     for i := 1 to 3 do
519       begin
520         gain := gain + vgate * agc[i];
521         vgate := vgate * low_y;
522       end;
523     end;
524     {multiply vout by gain and loss)
525     vout := vout * loss;
526     if (last_y > 0) and (vout < 0) then goto start3
527     else
528       begin
529         last_y := vout;
530         goto start2;
531       end;
532   end;
533
534   start16: {Bandpass filter execute)
535     y12 := a11 * y10 + a12 * y11 + vin + vout;
536     vout := k16 * (b10 * y12 + y11 * b12);
537     y11 := y10;
538     y10 := y12;
539     vout := vout * 2 * gain;
540
541     {Full wave rectifier)
542     vgate := -abs(vout);
543
544     {lowpass filter)
545     vgate := vgate * low_c1 + low_y * low_c2;
546     low_y := vgate;
547     vgate := vgate / 2.0;
548
549     {gain execute)
550     gain := agc[0];
551     for i := 1 to 3 do
552       begin
553         gain := gain + vgate * agc[i];
554         vgate := vgate * low_y;
555       end;
556     end;
557     {multiply vout by gain and loss)
558     vout := vout * loss;
559     if (last_y > 0) and (vout < 0) then goto start3
560     else
561       begin
562         last_y := vout;
563         goto start2;
564       end;
565   end;
566
567   start17: {Bandpass filter execute)
568     y12 := a11 * y10 + a12 * y11 + vin + vout;
569     vout := k16 * (b10 * y12 + y11 * b12);
570     y11 := y10;
571     y10 := y12;
572     vout := vout * 2 * gain;
573
574     {Full wave rectifier)
575     vgate := -abs(vout);
576
577     {lowpass filter)
578     vgate := vgate * low_c1 + low_y * low_c2;
579     low_y := vgate;
580     vgate := vgate / 2.0;
581
582     {gain execute)
583     gain := agc[0];
584     for i := 1 to 3 do
585       begin
586         gain := gain + vgate * agc[i];
587         vgate := vgate * low_y;
588       end;
589     end;
590     {multiply vout by gain and loss)
591     vout := vout * loss;
592     if (last_y > 0) and (vout < 0) then goto start3
593     else
594       begin
595         last_y := vout;
596         goto start2;
597       end;
598   end;
599
600   start18: {Bandpass filter execute)
601     y12 := a11 * y10 + a12 * y11 + vin + vout;
602     vout := k16 * (b10 * y12 + y11 * b12);
603     y11 := y10;
604     y10 := y12;
605     vout := vout * 2 * gain;
606
607     {Full wave rectifier)
608     vgate := -abs(vout);
609
610     {lowpass filter)
611     vgate := vgate * low_c1 + low_y * low_c2;
612     low_y := vgate;
613     vgate := vgate / 2.0;
614
615     {gain execute)
616     gain := agc[0];
617     for i := 1 to 3 do
618       begin
619         gain := gain + vgate * agc[i];
620         vgate := vgate * low_y;
621       end;
622     end;
623     {multiply vout by gain and loss)
624     vout := vout * loss;
625     if (last_y > 0) and (vout < 0) then goto start3
626     else
627       begin
628         last_y := vout;
629         goto start2;
630       end;
631   end;
632
633   start19: {Bandpass filter execute)
634     y12 := a11 * y10 + a12 * y11 + vin + vout;
635     vout := k16 * (b10 * y12 + y11 * b12);
636     y11 := y10;
637     y10 := y12;
638     vout := vout * 2 * gain;
639
640     {Full wave rectifier)
641     vgate := -abs(vout);
642
643     {lowpass filter)
644     vgate := vgate * low_c1 + low_y * low_c2;
645     low_y := vgate;
646     vgate := vgate / 2.0;
647
648     {gain execute)
649     gain := agc[0];
650     for i := 1 to 3 do
651       begin
652         gain := gain + vgate * agc[i];
653         vgate := vgate * low_y;
654       end;
655     end;
656     {multiply vout by gain and loss)
657     vout := vout * loss;
658     if (last_y > 0) and (vout < 0) then goto start3
659     else
660       begin
661         last_y := vout;
662         goto start2;
663       end;
664   end;
665
666   start20: {Bandpass filter execute)
667     y12 := a11 * y10 + a12 * y11 + vin + vout;
668     vout := k16 * (b10 * y12 + y11 * b12);
669     y11 := y10;
670     y10 := y12;
671     vout := vout * 2 * gain;
672
673     {Full wave rectifier)
674     vgate := -abs(vout);
675
676     {lowpass filter)
677     vgate := vgate * low_c1 + low_y * low_c2;
678     low_y := vgate;
679     vgate := vgate / 2.0;
680
681     {gain execute)
682     gain := agc[0];
683     for i := 1 to 3 do
684       begin
685         gain := gain + vgate * agc[i];
686         vgate := vgate * low_y;
687       end;
688     end;
689     {multiply vout by gain and loss)
690     vout := vout * loss;
691     if (last_y > 0) and (vout < 0) then goto start3
692     else
693       begin
694         last_y := vout;
695         goto start2;
696       end;
697   end;
698
699   start21: {Bandpass filter execute)
700     y12 := a11 * y10 + a12 * y11 + vin + vout;
701     vout := k16 * (b10 * y12 + y11 * b12);
702     y11 := y10;
703     y10 := y12;
704     vout := vout * 2 * gain;
705
706     {Full wave rectifier)
707     vgate := -abs(vout);
708
709     {lowpass filter)
710     vgate := vgate * low_c1 + low_y * low_c2;
711     low_y := vgate;
712     vgate := vgate / 2.0;
713
714     {gain execute)
715     gain := agc[0];
716     for i := 1 to 3 do
717       begin
718         gain := gain + vgate * agc[i];
719         vgate := vgate * low_y;
720       end;
721     end;
722     {multiply vout by gain and loss)
723     vout := vout * loss;
724     if (last_y > 0) and (vout < 0) then goto start3
725     else
726       begin
727         last_y := vout;
728         goto start2;
729       end;
730   end;
731
732   start22: {Bandpass filter execute)
733     y12 := a11 * y10 + a12 * y11 + vin + vout;
734     vout := k16 * (b10 * y12 + y11 * b12);
735     y11 := y10;
736     y10 := y12;
737     vout := vout * 2 * gain;
738
739     {Full wave rectifier)
740     vgate := -abs(vout);
741
742     {lowpass filter)
743     vgate := vgate * low_c1 + low_y * low_c2;
744     low_y := vgate;
745     vgate := vgate / 2.0;
746
747     {gain execute)
748     gain := agc[0];
749     for i := 1 to 3 do
750       begin
751         gain := gain + vgate * agc[i];
752         vgate := vgate * low_y;
753       end;
754     end;
755     {multiply vout by gain and loss)
756     vout := vout * loss;
757     if (last_y > 0) and (vout < 0) then goto start3
758     else
759       begin
760         last_y := vout;
761         goto start2;
762       end;
763   end;
764
765   start23: {Bandpass filter execute)
766     y12 := a11 * y10 + a12 * y11 + vin + vout;
767     vout := k16 * (b10 * y12 + y11 * b12);
768     y11 := y10;
769     y10 := y12;
770     vout := vout * 2 * gain;
771
772     {Full wave rectifier)
773     vgate := -abs(vout);
774
775     {lowpass filter)
776     vgate := vgate * low_c1 + low_y * low_c2;
777     low_y := vgate;
778     vgate := vgate / 2.0;
779
780     {gain execute)
781     gain := agc[0];
782     for i := 1 to 3 do
783       begin
784         gain := gain + vgate * agc[i];
785         vgate := vgate * low_y;
786       end;
787     end;
788     {multiply vout by gain and loss)
789     vout := vout * loss;
790     if (last_y > 0) and (vout < 0) then goto start3
791     else
792       begin
793         last_y := vout;
794         goto start2;
795       end;
796   end;
797
798   start24: {Bandpass filter execute)
799     y12 := a11 * y10 + a12 * y11 + vin + vout;
800     vout := k16 * (b10 * y12 + y11 * b12);
801     y11 := y10;
802     y10 := y12;
803     vout := vout * 2 * gain;
804
805     {Full wave rectifier)
806     vgate := -abs(vout);
807
808     {lowpass filter)
809     vgate := vgate * low_c1 + low_y * low_c2;
810     low_y := vgate;
811     vgate := vgate / 2.0;
812
813     {gain execute)
814     gain := agc[0];
815     for i := 1 to 3 do
816       begin
817         gain := gain + vgate * agc[i];
818         vgate := vgate * low_y;
819       end;
820     end;
821     {multiply vout by gain and loss)
822     vout := vout * loss;
823     if (last_y > 0) and (vout < 0) then goto start3
824     else
825       begin
826         last_y := vout;
827         goto start2;
828       end;
829   end;
830
831   start25: {Bandpass filter execute)
832     y12 := a11 * y10 + a12 * y11 + vin + vout;
833     vout := k16 * (b10 * y12 + y11 * b12);
834     y11 := y10;
835     y10 := y12;
836     vout := vout * 2 * gain;
837
838     {Full wave rectifier)
839     vgate := -abs(vout);
840
841     {lowpass filter)
842     vgate := vgate * low_c1 + low_y * low_c2;
843     low_y := vgate;
844     vgate := vgate / 2.0;
845
846     {gain execute)
847     gain := agc[0];
848     for i := 1 to 3 do
849       begin
850         gain := gain + vgate * agc[i];
851         vgate := vgate * low_y;
852       end;
853     end;
854     {multiply vout by gain and loss)
855     vout := vout * loss;
856     if (last_y > 0) and (vout < 0) then goto start3
857     else
858       begin
859         last_y := vout;
860         goto start2;
861       end;
862   end;
863
864   start26: {Bandpass filter execute)
865     y12 := a11 * y10 + a12 * y11 + vin + vout;
866     vout := k16 * (b10 * y12 + y11 * b12);
867     y11 := y10;
868     y10 := y12;
869     vout := vout * 2 * gain;
870
871     {Full wave rectifier)
872     vgate := -abs(vout);
873
874     {lowpass filter)
875     vgate := vgate * low_c1 + low_y * low_c2;
876     low_y := vgate;
877     vgate := vgate / 2.0;
878
879     {gain execute)
880     gain := agc[0];
881     for i := 1 to 3 do
882       begin
883         gain := gain + vgate * agc[i];
884         vgate := vgate * low_y;
885       end;
886     end;
887     {multiply vout by gain and loss)
888     vout := vout * loss;
889     if (last_y > 0) and (vout < 0) then goto start3
890     else
891       begin
892         last_y := vout;
893         goto start2;
894       end;
895   end;
896
897   start27: {Bandpass filter execute)
898     y12 := a11 * y10 + a12 * y11 + vin + vout;
899     vout := k16 * (b10 * y12 + y11 * b12);
900     y11 := y10;
901     y10 := y12;
902     vout := vout * 2 * gain;
903
904     {Full wave rectifier)
905     vgate := -abs(vout);
906
907     {lowpass filter)
908     vgate := vgate * low_c1 + low_y * low_c2;
909     low_y := vgate;
910     vgate := vgate / 2.0;
911
912     {gain execute)
913     gain := agc[0];
914     for i := 1 to 3 do
915       begin
916         gain := gain + vgate * agc[i];
917         vgate := vgate * low_y;
918       end;
919     end;
920     {multiply vout by gain and loss)
921     vout := vout * loss;
922     if (last_y > 0) and (vout < 0) then goto start3
923     else
924       begin
925         last_y := vout;
926         goto start2;
927       end;
928   end;
929
930   start28: {Bandpass filter execute)
931     y12 := a11 * y10 + a12 * y11 + vin + vout;
932     vout := k16 * (b10 * y12 + y11 * b12);
933     y11 := y10;
934     y10 := y12;
935     vout := vout * 2 * gain;
936
937     {Full wave rectifier)
938     vgate := -abs(vout);
939
940     {lowpass filter)
941     vgate := vgate * low_c1 + low_y * low_c2;
942     low_y := vgate;
943     vgate := vgate / 2.0;
944
945     {gain execute)
946     gain := agc[0];
947     for i := 1 to 3 do
948       begin
949         gain := gain + vgate * agc[i];
950         vgate := vgate * low_y;
951       end;
952     end;
953     {multiply vout by gain and loss)
954     vout := vout * loss;
955     if (last_y > 0) and (vout < 0) then goto start3
956     else
957       begin
958         last_y := vout;
959         goto start2;
960       end;
961   end;
962
963   start29: {Bandpass filter execute)
964     y12 := a11 * y10 + a12 * y11 + vin + vout;
965     vout := k16 * (b10 * y12 + y11 * b12);
966     y11 := y10;
967     y10 := y12;
968     vout := vout * 2 * gain;
969
970     {Full wave rectifier)
971     vgate := -abs(vout);
972
973     {lowpass filter)
974     vgate := vgate * low_c1 + low_y * low_c2;
975     low_y := vgate;
976     vgate := vgate / 2.0;
977
978     {gain execute)
979     gain := agc[0];
980     for i := 1 to 3 do
981       begin
982         gain := gain + vgate * agc[i];
983         vgate := vgate * low_y;
984       end;
985     end;
986     {multiply vout by gain and loss)
987     vout := vout * loss;
988     if (last_y > 0) and (vout < 0) then goto start3
989     else
990       begin
991         last_y := vout;
992         goto start2;
993       end;
994   end;
995
996   start30: {Bandpass filter execute)
997     y12 := a11 * y10 + a12 * y11 + vin + vout;
998     vout := k16 * (b10 * y12 + y11 * b12);
999     y11 := y10;
1000    y10 := y12;
1001    vout := vout * 2 * gain;
1002
1003    {Full wave rectifier)
1004    vgate := -abs(vout);
1005
1006    {lowpass filter)
1007    vgate := vgate * low_c1 + low_y * low_c2;
1008    low_y := vgate;
1009    vgate := vgate / 2.0;
1010
1011    {gain execute)
1012    gain := agc[0];
1013    for i := 1 to 
```

```

202 {Full wave rectifier}
203 vgate := -abs(vout);
204
205 {lowpass filter}
206 vgate := vgate * low_c1 + low_y * low_c2;
207 low_y := vgate;
208 vgate := vgate / 2.0;
209
210 {gain execute}
211 gain := agc(0);
212 for i := 1 to 3 do
213   begin
214     gain := gain + vgate * agc[i];
215     vgate := vgate * low_y;
216   end;
217 {writeln(vout:20,low_y:25,gain:30);}
218
219 {multiply vout by gain and loss}
220 {writeln(gain:20,vout:30);}
221 vout := vout * loss;
222 count1 := count1 + 1;
223 if (last_y > 0) and (vout < 0) then
224   begin
225     count2 := count2 + 1;
226     writeln(count2);
227     if count2 >= 100 then goto final;
228   end;
229 last_y := vout;
230 goto start4;
231 final: writeln('number of point = ',count1);
232        writeln('average = ',count1 / 100);
233 end.

```



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ช. โปรแกรมการจำลองการข้ามโมดของออสซิลเลเตอร์แบบดีเลย์ไลน์



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

1 (.p164)
2 {this program is used for simulate
3 the dispersive delayline which fingerwidth 1/4 lambda}
4 program delayline;
5 uses crt,gkernel,gdriver,printer,dos,graph,turbo3;
6 Const Lambda = 64;
7     long = 6000;
8     no_rx = 10;
9     no_tx = 10;
10    for_array = 39;
11 type
12     mempo2 = ^heap2;
13     heap2 = record
14         data : real;
15     end;
16 var sig_input : real;
17     input      : array[0..lambda] of real;
18     line_forw,line_refl : array[1..long] of mempo2;
19     elect_rec : array[1..no_rx] of real;
20     tpoint_forw,tpoint_back,rpoint_forw,rpoint_back : integer;
21     input_pointer,lambda_by4,lambda_by2 : integer;
22     i,j,k,loop : integer;
23     max_output,delta_t,output : real;
24     step,limit,point_1,point_2,count : integer;
25     key : char;
26     name : string;
27     filename : text;
28     pic,div_r,w,g1_iae,g2_iae,g3_pae,g4_pae : real;
29     lambda_step : array[1..for_array] of integer;
30     lambda_temp,lambda_temp2 : integer;
31     long_limit : integer;
32 var tsin,tsout : integer;
33     t1,t2,t3,t4 : real;
34 var rsin,rsout : integer;
35     r1,r2,r3,r4 : real;
36 procedure gen_reflect;
37 begin
38     case loop of
39         1 : begin
40             g1_iae := 0.0; {reflect ratio spacing to electrode}
41             g2_iae := 0.0; {reflect ratio electrode to spacing}
42             g3_pae := 1.0; {passing ratio spacing to electrode}
43             g4_pae := 1.0; {passing ratio electrode to spacing}
44         end;
45         2 : begin
46             g1_iae := 0.01; {reflect ratio spacing to electrode}
47             g2_iae := 0.01; {reflect ratio electrode to spacing}
48             g3_pae := 0.99; {passing ratio spacing to electrode}
49             g4_pae := 0.99; {passing ratio electrode to spacing}
50         end;
51         3 : begin
52             g1_iae := 0.05; {reflect ratio spacing to electrode}
53             g2_iae := 0.05; {reflect ratio electrode to spacing}
54             g3_pae := 0.95; {passing ratio spacing to electrode}
55             g4_pae := 0.95; {passing ratio electrode to spacing}
56         end;
57     end;
58 end;
59 procedure receive;
60 procedure rexecute(electrode : integer);
61 begin
62     r1 := line_forw[rsin]^data;
63     r2 := line_refl[rsout]^data;
64     r3 := line_forw[point_1]^data;
65     r4 := line_refl[point_2]^data;
66     elect_rec[i] := elect_rec[i] - electrode * (r3 + r2);
67     line_forw[rsin]^data := r1 * g3_pae + r2 * g2_iae;
68     line_forw[point_1]^data := r3 * g4_pae + r4 * g1_iae;
69     line_refl[rsout]^data := r2 * g4_pae + r1 * g1_iae;
70     line_refl[point_2]^data := r4 * g3_pae + r3 * g2_iae;
71     elect_rec[i] := elect_rec[i] + electrode * (line_forw[rsin]^data
+ line_refl[point_2]^data);
72     rsin := rsin + lambda_step[i] * 2;
73     if rsin > long then rsin := rsin - long;
74     rsout := rsout + lambda_step[i] * 2;
75     if rsout > long then rsout := rsout - long;
76     point_1 := rsin + lambda_step[i];
77     if point_1 > long then point_1 := point_1 - long;
78     point_2 := rsout + lambda_step[i];
79     if point_2 > long then point_2 := point_2 - long;
80 end;
81 begin
82     rsin := rpoint_forw;
83     rsout := rpoint_back;
84     point_1 := rsin + lambda_by4;
85     if point_1 > long then point_1 := point_1 - long;
86     point_2 := rsout + lambda_by4;
87     if point_2 > long then point_2 := point_2 - long;
88     output := 0.0;
89     for i := 1 to no_rx do
90         begin
91             rexecute(1);
92             rexecute(-1);
93         end;
94     end;
95     output := output / lambda_temp / 4;
96     line_forw[point_1]^data := -0.0;
97     drawpoint(count,output + pic);
98     {writeln(filename,output:13); }
99 end;
100 label quit;
101 begin
102     lambda_step[1] := lambda div 4;
103     long_limit := lambda_step[1];
104     for j := 2 to no_rx do
105         begin
106             lambda_step[j] := lambda_step[j-1] + 2;
107             long_limit := long_limit + lambda_step[j];
108             write(lambda_step[j]:10);
109         end;
110     long_limit := long_limit * 4;
111     writeln('total = ',long_limit);
112     for i := 1 to long do
113         begin
114             new(line_forw[i]);
115             new(line_refl[i]);
116             line_forw[i]^data := 0.0; line_refl[i]^data := 0.0;
117         end;
118     write('Enter filename of the data ');
119     readln(name);
120     assign(filename,name);
121     rewrite(filename);
122     InitGraphic;
123     DefineWorld(1,0,-2,long_limit + 13 * 3 * lambda,2);
124     {DefineWorld(1,0,5.0,long,-5.0);}
125     Definewindow(1,0,0,xmaxglb,ymaxglb);
126     SelectWorld(1);
127     selectwindow(1);
128     clearscreen;
129     pic := -1.5;
130     for loop := 1 to 2 do
131         begin
132             gen_reflect;
133             writeln(filename,'Rfe = ',g1_iae);
134             writeln(filename,'Ref = ',g2_iae);
135             writeln(filename,'Tfe = ',g3_pae);
136             writeln(filename,'Tef = ',g4_pae);
137             lambda_temp := lambda;
138             for step := 1 to 500 do
139                 begin
140                     max_output := 0.0;
141                     w := step;
142                     w := w / 200 + 0.01;
143                     for i := 0 to lambda do input[i] := 0.0;
144                     for i := 0 to no_rx do elect_rec[i] := 0.0;
145                     for i := 1 to long do
146                         begin
147                             line_forw[i]^data := 0.0; line_refl[i]^data := 0.0;
148                         end;
149                     lambda_by4 := lambda div 4;
150                     lambda_by2 := lambda div 2;
151                     tpoint_forw := 1;
152                     tpoint_back := 1;
153                     rpoint_forw := (no_tx + 1) * lambda;
154                     rpoint_back := (no_tx + 1) * lambda;
155                     sig_input := 0.0;
156                     input_pointer := 1;
157                     delta_t := 2 * 3.1415927 / lambda;
158                     limit := lambda_temp * 10;
159                     for count := 0 to long_limit + limit + lambda do
160                         begin
161                             if keypressed then goto quit;
162                             {write(filename,count:4);}
163                             if count < limit then line_forw[rpoint_forw]^data
:= sin(w * delta_t * count)
164                             else line_forw[rpoint_forw]^data := 0.0;
165                             {drawpoint(count,line_forw[rpoint_forw]^data/5);}
166                             receive;
167                             if abs(output) > max_output then max_output := abs(c
168                             );
169                             line_refl[rpoint_back]^data := 0.0;
170                             dec(rpoint_forw); if rpoint_forw < 1 then rpoint_forw
:= rpoint_back - long;
171                             inc(rpoint_back); if rpoint_back > long then rpoint_b
:= rpoint_back - long;
172                         end;
173                     writeln(filename,step:5,max_output:13);
174                     drawpoint(step,max_output);
175                     pic := pic + 0.5;
176                     lambda_temp := lambda_temp + lambda div 2;
177                 end;
178             end;
179             quit: {read(kbd,key);}
180                 if upcase(key) = 'Y' then
181                     begin
182                         writeln(lst,'Rfe = ',g1_iae);
183                         writeln(lst,'Ref = ',g2_iae);
184                         writeln(lst,'Tfe = ',g3_pae);
185                         writeln(lst,'Tef = ',g4_pae);
186                         Hardcopy(false,1);
187                     end;
188                 close(filename);
189                 leavegraphic;
190 end.

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

1 Program delayline;
2 {SI typedef.sys}
3 {SI graphix.sys}
4 {SI kernel.sys}
5 const gain1 = 382.97872e-3; {1.8 * 1k / 4.7k}
6   lambda = 145;
7   long = 6959;
8   w0 = 1.0;
9   w1 = 1.25;
10  q = 9.0;
11  no_fing = 3;
12  pl = 3.1415927;
13  w_step = 0.0005;
14  low_w = 18.38;
15 var BBD : array[0..long] of real;
16   finger_gain : array[1..3] of real;
17   delta_t, n1, k1, k2, k3, k4, k5, k6, a1, a2, tp, tmpo, tmpol, x, tmp_w : real;
18   low_c1, low_c2, low_con : real;
19   finger, index, space, i, x_cor : integer;
20 procedure coefficient(w : real);
21 begin
22   n1 := w / q;
23   k1 := 2 * delta_t * w;
24   k2 := q * sqrt(delta_t * w);
25   k3 := 4 * q + k1 + k2;
26   k4 := -8 * q + 2 * k2;
27   k5 := 4 * q - k1 + k2;
28   k6 := 2 * n1 * delta_t * q / k3;
29   a1 := -k4 / k3;
30   a2 := -k5 / k3;
31 end;
32
33 Procedure coefficient_lowpass;
34 begin
35   low_c1 := delta_t * exp(-delta_t * low_w / 2) * low_w;
36   low_c2 := exp(-delta_t * low_w);
37 end;
38
39 function tanh(t : real) : real;
40 var m1, pl : real;
41 begin
42   t := t * 3.0;
43   m1 := exp(-t);
44   pl := exp(t);
45   t := (pl - m1) / (pl + m1);
46   if abs(t) > 0.4 then t := 0.4 * t / abs(t);
47   tanh := t;
48 end;
49
50 function rnd : real;
51 var s1, s2 : real;
52   gp : integer;
53 begin
54   s1 := random;
55   s2 := frac(10 * s1) * 10;
56   gp := trunc(s2);
57   if odd(gp) then s1 := -s1;
58   rnd := s1 * 1e-1;
59 end;
60
61 label start, jump, start2, jump2, start3;
62 {main program}
63 begin
64   InitGraphic;
65   DefineWorld(1, 0, -2.5, 32000, 2.5);
66   Definewindow(1, 0, 0, xmaxglb, ymaxglb);
67   Selectworld(1);
68   selectwindow(1);
69
70   {clear delayline}
71   for finger := 0 to long do BBD[finger] := rnd;
72   {set gain of finger}
73   finger_gain[1] := 2.48;
74   finger_gain[2] := 1.72;
75   finger_gain[3] := 1.32;
76   delta_t := 2 * pi / (w0 * lambda);
77   finger := long;
78   tmpo := 0.0;
79   tmpol := 0.0;
80   space := 16 * lambda;
81   x_cor := 0;
82   coefficient(w0); {generate coefficient}
83   coefficient_lowpass;
84   low_con := 0.0;
85   start: tp := 0.0;
86   for index := 1 to 3 do
87     begin
88       i := finger - (index - 1) * space;
89       if i < 0 then i := i + long + 1;
90       tp := tp + finger_gain[index] * bbd[i];
91     end;
92     tp := tp * gain1;
93     tp := tp + tmpol * a2 + tmpo * a1;
94     x := (tp - tmpol) * k6;
95
96     {lowpass filter}
97     {x := x * low_c1 + low_con * low_c2;
98     low_con := x;}
99     drawpoint(x_cor, x);
100

```

```

101   tmpol := tmpo;
102   tmpo := tp;
103   BBD[finger] := tanh(x);
104   finger := finger - 1;
105   if finger < 0 then finger := long;
106   if keypressed then goto jump;
107   x_cor := x_cor + 1;
108   if x_cor > 32000 then
109     begin
110       clearscreen;
111       x_cor := 0;
112     end;
113   goto start;
114   jump: tmp_w := w0;
115   repeat
116     coefficient(tmp_w); {generate coefficient}
117     tp := 0.0;
118     for index := 1 to 3 do
119       begin
120         i := finger - (index - 1) * space;
121         if i < 0 then i := i + long + 1;
122         tp := tp + finger_gain[index] * bbd[i];
123       end;
124       tp := tp * gain1;
125       tp := tp + tmpol * a2 + tmpo * a1;
126       x := (tp - tmpol) * k6;
127
128       {lowpass filter}
129       {x := x * low_c1 + low_con * low_c2;
130       low_con := x;}
131
132       drawpoint(x_cor, x);
133       tmpol := tmpo;
134       tmpo := tp;
135       BBD[finger] := tanh(x);
136       finger := finger - 1;
137       x_cor := x_cor + 1;
138       if finger < 0 then finger := long;
139       if x_cor > 32000 then
140         begin
141           clearscreen;
142           x_cor := 0;
143         end;
144       tmp_w := tmp_w + w_step;
145       until tmp_w >= w1;
146       coefficient(w1); {generate coefficient}
147       start2: tp := 0.0;
148       for index := 1 to 3 do
149         begin
150           i := finger - (index - 1) * space;
151           if i < 0 then i := i + long + 1;
152           tp := tp + finger_gain[index] * bbd[i];
153         end;
154         tp := tp * gain1;
155         tp := tp + tmpol * a2 + tmpo * a1;
156         x := (tp - tmpol) * k6;
157
158         {lowpass filter}
159         {x := x * low_c1 + low_con * low_c2;
160         low_con := x;}
161
162         drawpoint(x_cor, x);
163         tmpol := tmpo;
164         tmpo := tp;
165         BBD[finger] := tanh(x);
166         finger := finger - 1;
167         x_cor := x_cor + 1;
168         if finger < 0 then finger := long;
169         if x_cor > 32000 then
170           begin
171             readln(i);
172             if i = 2 then hardcopy(false, 1);
173             clearscreen;
174             x_cor := 0;
175             if i = 3 then goto jump2;
176           end;
177         - goto start2;
178       jump2: tmp_w := w1;
179       repeat
180         coefficient(tmp_w); {generate coefficient}
181         tp := 0.0;
182         for index := 1 to 3 do
183           begin
184             i := finger - (index - 1) * space;
185             if i < 0 then i := i + long + 1;
186             tp := tp + finger_gain[index] * bbd[i];
187           end;
188           tp := tp * gain1;
189           tp := tp + tmpol * a2 + tmpo * a1;
190           x := (tp - tmpol) * k6;
191
192           {lowpass filter}
193           {x := x * low_c1 + low_con * low_c2;
194           low_con := x;}
195
196           drawpoint(x_cor, x);
197           tmpol := tmpo;
198           tmpo := tp;
199           BBD[finger] := tanh(x);
200           finger := finger - 1;

```

```

201 x_cor := x_cor + 1;
202 if finger < 0 then finger := long;
203 if x_cor > 32000 then
204     begin
205         clearscreen;
206         x_cor := 0;
207     end;
208 tmp_w := tmp_w - w_step;
209 until tmp_w <= w0;
210 coefficient(w0); (generate coefficient)
211 start3: tp := 0.0;
212 for index := 1 to 3 do
213     begin
214         i := finger - (index - 1) * space;
215         if i < 0 then i := i + long + 1;
216         tp := tp + finger_gain [index] * bbd[i];
217     end;
218 tp := tp * gain1;
219 tp := tp + tmpol * a2 + tmpo * a1;
220 x := (tp - tmpol) * k6;
221
222 (lowpass filter)
223 {x := x * low_c1 + low_con * low_c2;
224 low_con := x;}
225
226 drawpoint(x_cor,x);
227 tmpol := tmpo;
228 tmpo := tp;
229 BBD[finger] := tanh(x);
230 finger := finger - 1;
231 x_cor := x_cor + 1;
232 if finger < 0 then finger := long;
233 if x_cor > 32000 then
234     begin
235         readln(i);
236         if i = 2 then hardcopy(false,1);
237         clearscreen;
238         x_cor := 0;
239     end;
240 goto start3;
241 leavegraphic;
242 end.

```



ภาคผนวก ๗. โปรแกรมการจำลอง IDT แบบมีความกว้างทั้งตัวรับและตัวส่ง



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

1 this program is used for simulate
2 the dispersive delayline which fingerwidth 1/4 lambda;
3
4 program delayline;
5 uses cit,gkernel,gdriver,printer,dos,graph,turbo3;
6 const lambda = 64;
7     long = 6000;
8     no_tx = 33;
9     for_array = 39;
10
11 type
12     mempo2 = ^heap2;
13     heap2 = record
14         data : real;
15     end;
16 var sig_input : real;
17     input : array[0..lambda] of real;
18     line_forw,line_refl : array[1..long] of mempo2;
19     tpoint_forw,tpoint_back,rpoin_forw,rpoin_back : integer;
20     input_pointer,lambda_by4,lambda_by2 : integer;
21     temp_step,i,j,k,loop : integer;
22     delta_t,output : real;
23     step,limit,point_1,point_2,count : integer;
24     key : char;
25     name : string;
26     filename : text;
27     pic,div_r,w,g1_iae,g2_iae,g3_pae,g4_pea : real;
28     lambda_step : array[1..for_array] of integer;
29     lambda_temp,lambda_temp2 : integer;
30
31 var tsin,tsout : integer;
32     t1,t2,t3,t4 : real;
33
34 var rsin,rsout : integer;
35     r1,r2,r3,r4 : real;
36
37 procedure gen_reflect;
38 begin
39     case loop of
40         1 : begin
41             g1_iae := 0.0; {reflect ratio spacing to electrode}
42             g2_iae := 0.0; {reflect ratio electrode to spacing}
43             g3_pae := 1.0; {passing ratio spacing to electrode}
44             g4_pea := 1.0; {passing ratio electrode to spacing}
45         end;
46         2 : begin
47             g1_iae := 0.01; {reflect ratio spacing to electro}
48             g2_iae := 0.01; {reflect ratio electrode to spac}
49             g3_pae := 0.99; {passing ratio spacing to electro}
50             g4_pea := 0.99; {passing ratio electrode to spac}
51         end;
52         3 : begin
53             g1_iae := 0.02; {reflect ratio spacing to electro}
54             g2_iae := 0.02; {reflect ratio electrode to spac}
55             g3_pae := 0.98; {passing ratio spacing to electro}
56             g4_pea := 0.98; {passing ratio electrode to spac}
57         end;
58         4 : begin
59             g1_iae := 0.005; {reflect ratio spacing to electro}
60             g2_iae := 0.005; {reflect ratio electrode to spac}
61             g3_pae := 0.995; {passing ratio spacing to electro}
62             g4_pea := 0.995; {passing ratio electrode to spac}
63         end;
64     end;
65 end;
66
67 procedure transmit;
68 var tsin,tsout : integer;
69     t1,t2,t3,t4 : real;
70
71 procedure execute(multip : integer);
72 var sig_inputtemp : real;
73 begin
74     sig_inputtemp := sig_input * multip;
75     t1 := line_forw[tsin]^data;
76     t2 := line_refl[tsout]^data;
77     t3 := line_forw[point_1]^data;
78     t4 := line_refl[point_2]^data;
79     line_forw[tsin]^data := t1 * g3_pae + (t2 + sig_inputtemp) * g2_iae;
80     line_forw[point_1]^data := (t3 + sig_inputtemp) * g4_pea + t4 * g3_pae;
81     line_refl[tsout]^data := (t2 + sig_inputtemp) * g4_pea + t1 * g2_iae;
82     line_refl[point_2]^data := t4 * g3_pae + (t3 + sig_inputtemp) * g2_iae;
83     tsin := tsin + lambda_step[i] * 2;
84     if tsin > long then tsin := tsin - long;
85     tsout := tsout + lambda_step[i] * 2;
86     if tsout > long then tsout := tsout - long;
87     point_1 := tsin + lambda_step[i];
88     if point_1 > long then point_1 := point_1 - long;
89     point_2 := tsout + lambda_step[i];
90     if point_2 > long then point_2 := point_2 - long;
91 end;
92
93 begin
94     tsin := input_pointer - lambda_by4;
95     if tsout < 1 then tsout := lambda + tsout;
96     sig_input := sig_input + input[tsin] - input[tsout];
97     tsin := tpoint_forw;
98     tsout := tpoint_back - 1;
99     if tsout < 1 then tsout := long;
100    line_refl[tsout]^data := 0.0;
101    tsout := tpoint_back;
102    point_1 := tsin + lambda_by4;
103    if point_1 > long then point_1 := point_1 - long;
104    point_2 := tsout + lambda_by4;
105    if point_2 > long then point_2 := point_2 - long;
106    for i := 1 to no_tx do
107        begin
108            execute(i);
109            execute(-i);
110        end;
111        drawpoint(count,line_forw[tsin]^data * 3 + 1.0);
112        writeln(filename,line_forw[tsin]^data);
113    end;
114    label quit;
115    begin
116        lambda_step[1] := (lambda * 3) div 4;
117        step := lambda_step[1];
118        for j := 2 to no_tx do
119            begin
120                lambda_step[j] := lambda_step[j-1] - 1;
121                step := step + lambda_step[j];
122                write(lambda_step[j]:10);
123            end;
124            writeln('total = ',step * 4);
125            repeat until keypressed;
126            temp_step := step * 4;
127            for i := 1 to long do
128                begin
129                    new(line_forw[i]);
130                    new(line_refl[i]);
131                    line_forw[i]^data := 0.0; line_refl[i]^data := 0.0;
132                end;
133            end;
134            write('Enter filename of the data ');
135            readln(name);
136            assign(filename,name);
137            rewrite(filename);
138            InitGraphic;
139            DefineWorld(1,0,5.0, long,-5.0);
140            DefineWindow(1,0,0,xmaxylb,ymaxylb);
141            SelectWorld(1);
142            selectwindow(1);
143            for loop := 1 to 4 do
144                begin
145                    clearscreen;
146                    gen_reflect;
147                    writeln(filename,'Rfe = ',g1_iae);
148                    writeln(filename,'Ref = ',g2_iae);
149                    writeln(filename,'Tfe = ',g3_pae);
150                    writeln(filename,'Tef = ',g4_pea);
151                    for i := 0 to lambda do input[i] := 0.0;
152                    for i := 1 to long do
153                        begin
154                            line_forw[i]^data := 0.0; line_refl[i]^data := 0.0;
155                        end;
156                    lambda_by4 := lambda div 4;
157                    lambda_by2 := lambda div 2;
158                    tpoint_forw := 1;
159                    tpoint_back := 1;
160                    sig_input := 0.0;
161                    input_pointer := 1;
162                    delta_t := 2 * 3.1415927 / lambda;
163                    count := 0;
164                    limit := lambda * 10;
165                    for count := 0 to temp_step + 2 * lambda do
166                        begin
167                            if keypressed then goto quit;
168                            write(filename,count:4);
169                            if count < 20(limit) then
170                                input[input_pointer] := 1 / lambda;
171                            else input[input_pointer] := 0.0;
172                            drawpoint(count,input[input_pointer] * lambda + 3.0);
173                            transmit;
174                            dec(tpoint_forw); if tpoint_forw < 1 then tpoint_forw := long;
175                            inc(tpoint_back); if tpoint_back > long then tpoint_back := tpoint_back - long;
176                            inc(input_pointer);
177                            if input_pointer > lambda then input_pointer := input_pointer - lambda;
178                        end;
179                    end;
180                    quit: {read(kbd,key);}
181                    if upcase(key) = 'Y' then
182                        begin
183                            writeln(lst,'Rfe = ',g1_iae);
184                            writeln(lst,'Ref = ',g2_iae);
185                            writeln(lst,'Tfe = ',g3_pae);
186                            writeln(lst,'Tef = ',g4_pea);
187                            Hardcopy(false,1);
188                        end;
189                    close(filename);
190                    leavegraphic;
191                end;
192            end;
193        end.

```

```

1 (this program is used for simulate
2 the dispersive delayline which fingerwidth 1/4 lambda)
3 program delayline;
4 uses crt,gkernel,gdriver,printer,dos,graph,turbo3;
5 Const Lambda = 64;
6     long = 6000;
7     no_rx = 33;
8     no_tx = 33;
9     for_array = 39;
10
11 type
12     mempo2 = ^heap2;
13     heap2 = record
14         data : real;
15     end;
16 var sig_input : real;
17     input      : array[0..lambda] of real;
18     line_forw,line_refl : array[1..long] of mempo2;
19     elect_rec : array[1..no_rx] of real;
20     tpoint_forw,tpoint_back,rpoin_forw,rpoin_back : integer;
21     input_pointer,lambda_by4,lambda_by2 : integer;
22     temp_step,i,j,k,loop : integer;
23     delta_t,output : real;
24     step,limit,point_1,point_2,count : integer;
25     key : char;
26     name : string;
27     filenameout,filenamein : text;
28     pic,div_r,w,g1_iae,g2_iae,g3_pae,g4_pae : real;
29     lambda_step : array[1..for_array] of integer;
30     lambda_temp,lambda_temp2 : integer;
31 var tsin,tsout : integer;
32     t1,t2,t3,t4 : real;
33 var rsin,rsout : integer;
34     r1,r2,r3,r4 : real;
35 procedure gen_reflect;
36 begin
37     case loop of
38         1 : begin
39             g1_iae := 0.0; {reflect ratio spacing to electrode}
40             g2_iae := 0.0; {reflect ratio electrode to spacing}
41             g3_pae := 1.0; {passing ratio spacing to electrode}
42             g4_pae := 1.0; {passing ratio electrode to spacing}
43         end;
44         2 : begin
45             g1_iae := 0.01; {reflect ratio spacing to electrode}
46             g2_iae := 0.01; {reflect ratio electrode to spacing}
47             g3_pae := 0.99; {passing ratio spacing to electrode}
48             g4_pae := 0.99; {passing ratio electrode to spacing}
49         end;
50         3 : begin
51             g1_iae := 0.02; {reflect ratio spacing to electrode}
52             g2_iae := 0.02; {reflect ratio electrode to spacing}
53             g3_pae := 0.98; {passing ratio spacing to electrode}
54             g4_pae := 0.98; {passing ratio electrode to spacing}
55         end;
56         4 : begin
57             g1_iae := 0.005; {reflect ratio spacing to electrode}
58             g2_iae := 0.005; {reflect ratio electrode to spacing}
59             g3_pae := 0.995; {passing ratio spacing to electrode}
60             g4_pae := 0.995; {passing ratio electrode to spacing}
61         end;
62     end;
63 end;
64 procedure receive;
65 procedure rexecute(electrode : integer);
66 begin
67     r1 := line_forw[rsin]^data;
68     r2 := line_refl[rsout]^data;
69     r3 := line_forw[point_1]^data;
70     r4 := line_refl[point_2]^data;
71     elect_rec[i] := elect_rec[i] - electrode * (r3 + r2);
72     line_forw[rsin]^data := r1 * g3_pae + r2 * g2_iae;
73     line_forw[point_1]^data := r3 * g4_pae + r4 * g1_iae;
74     line_refl[rsout]^data := r2 * g4_pae + r1 * g1_iae;
75     line_refl[point_2]^data := r4 * g3_pae + r3 * g2_iae;
76     elect_rec[i] := elect_rec[i] + electrode * (line_forw[rsin]^data
+ line_refl[point_2]^data);
77     rsin := rsin + lambda_step[i] * 2;
78     if rsin > long then rsin := rsin - long;
79     rsout := rsout + lambda_step[i] * 2;
80     if rsout > long then rsout := rsout - long;
81     point_1 := rsin + lambda_step[i];
82     if point_1 > long then point_1 := point_1 - long;
83     point_2 := rsout + lambda_step[i];
84     if point_2 > long then point_2 := point_2 - long;
85 end;
86 begin
87     rsin := rpoin_forw;

```

```

88     rsout := rpoin_back;
89     point_1 := rsin + lambda_by4;
90     if point_1 > long then point_1 := point_1 - long;
91     point_2 := rsout + lambda_by4;
92     if point_2 > long then point_2 := point_2 - long;
93     output := 0.0;
94     for i := 1 to no_rx do
95     begin
96         rexecute(i);
97         rexecute(-i);
98         output := output + elect_rec[i] / lambda_step[i] /
99         {output := output + elect_rec[i] / lambda;}}
100     end;
101     line_forw[point_1]^data := 0.0;
102     drawpoint(count,output + pic);
103     writeln(filenameout,output:13);
104 end;
105 label quit;
106 begin
107     lambda_step[1] := lambda div 4;
108     step := lambda_step[1];
109     for j := 2 to no_rx do
110     begin
111         lambda_step[j] := lambda_step[j-1] * 1;
112         step := step + lambda_step[j];
113         write(lambda_step[j]:10);
114     end;
115     writeln('total = ',step * 4);
116     temp_step := step * 4;
117     for i := 1 to long do
118     begin
119         new(line_forw[i]);
120         new(line_refl[i]);
121         line_forw[i]^data := 0.0; line_refl[i]^data :=
122         end;
123     write('Enter filenameout of the data ');
124     readln(name);
125     assign(filenameout,name);
126     rewrite(filenameout);
127     write('Enter filename in of the data ');
128     readln(name);
129     assign(filenamein,name);
130     reset(filenamein);
131     readln(filenamein,name);
132     readln(filenamein,name);
133     readln(filenamein,name);
134     readln(filenamein,name);
135     InitGraphic;
136     DefineWorld(1,0,-2,temp_step * 2,2);
137     {DefineWorld(1,0,5.0,long,-5.0);}
138     DefineWindow(1,0,0,xmaxglb,ymaxglb);
139     SelectWorld(1);
140     selectwindow(1);
141     clearscreen;
142     for loop := 1 to 1 do
143     begin
144         gen_reflect;
145         pic := -1.0;
146         writeln(filenameout,'Rfe = ',g1_iae);
147         writeln(filenameout,'Ref = ',g2_iae);
148         writeln(filenameout,'Tfe = ',g3_pae);
149         writeln(filenameout,'Tef = ',g4_pae);
150         lambda_temp := lambda;
151         for step := 1 to i do
152         begin
153             w := lambda;
154             w := w / lambda_temp;
155             limit := round(lambda_temp * 10);
156             lambda_temp2 := limit div 2;
157             div_r := lambda_temp2 / sqrt(abs(ln(0.01)));
158             for i := 0 to lambda do input[i] := 0.0;
159             for i := 0 to no_rx do elect_rec[i] := 0.0;
160             for i := 1 to long do
161             begin
162                 line_forw[i]^data := 0.0; line_refl[i]^data :=
163             end;
164             lambda_by4 := lambda div 4;
165             lambda_by2 := lambda div 2;
166             tpoint_forw := 1;
167             tpoint_back := 1;
168             rpoin_forw := (no_tx + 1) * lambda;
169             rpoin_back := (no_tx + 1) * lambda;
170             sig_input := 0.0;
171             input_pointer := 1;
172             delta_t := 2 * 3.1415927 / lambda;
173             limit := lambda_temp * 10;
174             for count := 0 to temp_step * 2 do
175             begin
176                 if keypressed then goto quit;
177                 write(filenameout,count:4);
178                 read inputfile;
179                 if not(eof(filenamein)) then
180                 begin
181                     readln(filenamein,name); name := copy(name,5,
182                     val(name,line_forw[rpoin_forw]^data,k);
183                 end
184                 else line_forw[rpoin_forw]^data := 0.0;
185                 {drawpoint(count,line_forw[rpoin_forw]^data/5);}
186                 receive;
187                 line_refl[rpoin_back]^data := 0.0;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้คัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

ong;
189      inc(rpoint_back); if rpoint_back > long then rpoint_back
= rpoint_back - long;
190      end;
191      pic := pic + 0.5;
192      lambda_temp := lambda_temp + lambda div 2;
193      end;
194      end;
195      quit: {read(kbd,key);}
196      if upcase(key) = 'Y' then
197      begin
198          Hardcopy(false,1);
199      end;
200      close(filenameout);
201      close(filenamein);
202      leavegraphic;
203  end.

```



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

Program Show Picture;
uses crt, gkernel, gdriver, printer, dos, graph;

const tx_fing = 1; {number of finger}
      rx_fing = 5; {number of finger}
      delay = 1; {delay 10 lambda}
      lambda_0 = 250;
      no_inputcycle = 7;
      tx_step = 0;
      rx_step = 0;
      w_start = 0.7;

var grdriver, grmode, lambda_d4, delay_long : integer;
    delay_line : array[0..6000] of real;
    input_tx, output_rx, delta_t, w_in, bbya : real;
    tx_start, rx_start, delay_start, i, j, k, l, m, lim_input, lambda_width :
integer;
    tx_distance, tx_width, rx_distance, rx_width : array[1..20] of integ
real;
    tx_plus, tx_minus, rx_plus, rx_minus, tx_apodi, rx_apodi : array[1..20
real;
    out_total, abs_max : real;
    min_chk, min_chk2, loop_count : integer;
    file_name : text;
procedure get_input;
begin
    input_tx := sin(w_in * i * delta_t);
    i := i + 1;
end;
label loop, final, start;
begin
    assign(file_name, 'figbb.dat');
    rewrite(file_name);
    InitGraphic;
    DefineWorld(1, 0, 10.0, 300, 0.0);
    Definewindow(1, 0, 0, xmaxglb, ymaxglb);
    SelectWorld(1);
    twindow(1);
    screen;
    bbya := 0.4;
    for i := 1 to 5 do
        begin
            abs_max := i / 0.02;
            l := round(abs_max);
            drawline(1, 0.5, l, 0.0);
        end;
    drawline(0, 0.0, 300, 0.0);
    drawline(0, 30.0, 0, 0.0);
loop:   delta_t := 2 * 3.1415926 / lambda_0;
    {check memory enough or not}
    delay_long := 0;
    w_in := w_start;
    lambda_width := lambda_0 div 4;
    for i := 1 to tx_fing do
        begin
            delay_long := delay_long + lambda_0 + (i - 1) * tx_ste
            tx_distance[i] := lambda_0 + (i - 1) * tx_step;
            tx_width[i] := tx_distance[i] div 4;
        end;
    delay_start := delay_long + 1;
    delay_long := delay_long + lambda_0 * delay;
    tx_start := 1;
    rx_start := delay_long + 1;
    for i := 1 to rx_fing do
        begin
            delay_long := delay_long + lambda_0 + (i - 1) * rx_ste
            rx_distance[i] := lambda_0 + (i - 1) * rx_step;
            rx_width[i] := round(rx_distance[i] * bbya / 2);
        end;
    if (delay_long > 6000) or (tx_fing > 20) or (rx_fing > 20) then
        begin
            writeln('Array of delay line not enough ');
            repeat until keypressed;
            goto final;
        end;
    {clear delay_line}
    for i := 1 to delay_long do delay_line[i] := 0.0;
    for i := 1 to 20 do
        begin
            tx_plus[i] := 0.0;
            tx_minus[i] := 0.0;
            rx_plus[i] := 0.0;
            rx_minus[i] := 0.0;
            tx_apodi[i] := 1.0;
            rx_apodi[i] := 1.0;
        end;
    i := 0;
    loop_count := round(w_in / 0.02);
    lim_input := no_inputcycle * lambda_0;
    abs_max := 0.0;
    min_chk := round((lambda_0 / w_start) * (rx_fing + 1));
    min_chk2 := round((lambda_0 / w_start) * (rx_fing + 3));
    start: get_input; {return in input_tx}
    k := rx_start;
    delay_line[k] := input_tx;
    out_total := 0.0;
    output_rx := 0.0;
    for j := 1 to rx_fing do
        begin
            rx_plus[j] := rx_plus[j] + delay_line[k];
            k := k + rx_width[j];
            if k > delay_long then k := k - delay_long;
            rx_minus[j] := rx_minus[j] + delay_line[k];
            k := k + rx_width[j];
            if k > delay_long then k := k - delay_long;
            rx_minus[j] := rx_minus[j] - delay_line[k];
            output_rx := output_rx - rx_minus[j] * rx_ap
        end;
    {output rx := out_total;}
    output_rx := output_rx / rx_width[1];
    {drawpoint(1 div 16, output_rx);}
    if (abs(output_rx) > abs_max) and (i > 2000) then ab
    := abs(output_rx);
    if (abs(output_rx) > abs_max) and (i > min_chk) then
        x := abs(output_rx);
        k := k - 1;
        if k < 1 then k := k + delay_long;
        delay_line[k] := 0.0;
        tx_start := tx_start - 1;
        rx_start := rx_start - 1;
        delay_start := delay_start - 1;
        if tx_start < 1 then tx_start := tx_start + delay_long
        if rx_start < 1 then rx_start := rx_start + delay_long
        if delay_start < 1 then delay_start := delay_start +
        l := l + 1;
        if l > min_chk2 then
            begin
                writeln(file_name, bbya, ' ', w_in, ' ', abs_max);
                drawpoint(loop_count, abs_max);
                drawpoint(loop_count, 0.0);
                loop_count := loop_count + 1;
                abs_max := 0.0;
                {clear delay_line}
                for i := 1 to delay_long do delay_line[i] := 0
                for i := 1 to 20 do
                    begin
                        tx_plus[i] := 0.0;
                        tx_minus[i] := 0.0;
                        rx_plus[i] := 0.0;
                        rx_minus[i] := 0.0;
                        tx_apodi[i] := 1.0;
                        rx_apodi[i] := 1.0;
                    end;
                i := 0;
                l := 0;
                w_in := w_in + 0.02;
                if (w_in > 1.8) and (w_in < 2.1) then
                    begin
                        w_in := 2.2;
                        loop_count := round(w_in / 0.02);
                    end;
                if (w_in > 3.8) and (w_in < 4.1) then
                    begin
                        w_in := 4.2;
                        loop_count := round(w_in / 0.02);
                    end;
                min_chk := round((lambda_0 / w_start) * (rx_fi
                min_chk2 := round((lambda_0 / w_start) * (rx_fi
            end;
            if w_in < 5.5 then goto start;
            bbya := bbya + 0.1;
            if bbya < 0.65 then goto loop;
            close(file_name);
            hardcopy(false, 1);
            LeaveGraphic;
        end.

```

```

1  [.pl64]
2  program delayline;
3  uses crt,gkernel,gdriver,printer,dos,graph,turbo3;
4  const lambda = 128;
5  long = 4000;
6  no_tx = 10;
7  no_rx = 10;
8  g1_iae = 0.01; {reflect ratio spacing to electrode}
9  g2_iae = 0.01; {reflect ratio electrode.to spacing}
10 g3_pae = 0.99; {passing ratio spacing to electrode}
11 g4_pae = 0.99; {passing ratio electrode to spacing}
12 (* g1_iae = 0.0; {reflect ratio spacing to electrode}
13 g2_iae = 0.0; {reflect ratio electrode to spacing}
14 g3_pae = 1.0; {passing ratio spacing to electrode}
15 g4_pae = 1.0; {passing ratio electrode to spacing} *)
16 var sig_input : real;
17 input : array[0..lambda] of real;
18 line_forw,line_refl : array[1..long] of real;
19 elect_rec : array[1..no_rx] of real;
20 tpoint_forw,tpoint_back,rpoin_forw,rpoint_back : integer;
21 input_pointer,lambda_by4,lambda_by2 : integer;
22 i,j,k : integer;
23 delta_t,output : real;
24 limit,point_1,point_2,count : integer;
25 key : char;
26 procedure transmit;
27 var tsin,tsout : integer;
28 t1,t2,t3,t4 : real;
29 procedure execute(multip : integer);
30 var sig_inputtemp : real;
31 begin
32   sig_inputtemp := sig_input * multip;
33   t1 := line_forw[tsin];
34   t2 := line_refl[tsout];
35   t3 := line_forw[point_1];
36   t4 := line_refl[point_2];
37   line_forw[tsin] := t1 * g3_pae + (t2 + sig_inputtemp) * g2_iae;
38   line_forw[point_1] := (t3 + sig_inputtemp) * g4_pae + t4 * g1_iae;
39   line_refl[tsout] := (t2 + sig_inputtemp) * g4_pae + t1 * g1_iae;
40   line_refl[point_2] := t4 * g3_pae + (t3 + sig_inputtemp) * g2_iae;
41   tsin := tsin + lambda_by2;
42   if tsin > long then tsin := tsin - long;
43   tsout := tsout + lambda_by2;
44   if tsout > long then tsout := tsout - long;
45   point_1 := tsin + lambda_by4;
46   if point_1 > long then point_1 := point_1 - long;
47   point_2 := tsout + lambda_by4;
48   if point_2 > long then point_2 := point_2 - long;
49 end;
50 begin
51   tsin := input_pointer;
52   tsout := input_pointer - lambda_by4;
53   if tsout < 1 then tsout := lambda + tsout;
54   sig_input := sig_input + input[tsin] - input[tsout];
55   tsin := tpoint_forw;
56   tsout := tpoint_back - 1;
57   if tsout < 1 then tsout := long;
58   line_refl[tsout] := 0.0;
59   tsout := tpoint_back;
60   point_1 := tsin + lambda_by4;
61   if point_1 > long then point_1 := point_1 - long;
62   point_2 := tsout + lambda_by4;
63   if point_2 > long then point_2 := point_2 - long;
64   for i := 1 to no_tx do
65     begin
66       execute(1);
67       execute(-1);
68     end;
69   drawpoint(count,line_forw[tsin] + 3.0);
70 end;
71 procedure receive;
72 var rsin,rsout : integer;
73 r1,r2,r3,r4 : real;
74 procedure rexecute(electrode : integer);
75 begin
76   r1 := line_forw[rsin];
77   r2 := line_refl[rsout];
78   r3 := line_forw[point_1];
79   r4 := line_refl[point_2];
80   elect_rec[i] := elect_rec[i] - electrode * (r3 + r2);
81   line_forw[rsin] := r1 * g3_pae + r2 * g2_iae;
82   line_forw[point_1] := r3 * g4_pae + r4 * g1_iae;
83   line_refl[rsout] := r2 * g4_pae + r1 * g1_iae;
84   line_refl[point_2] := r4 * g3_pae + r3 * g2_iae;
85   if (i = 1) and (electrode = 1) then
86     begin
87       drawpoint(count,line_forw[rpoint_forw] * 1 + 7.0);
88       {drawpoint(count,line_refl[rpoint_back] * 100 + 9.0);}
89       {drawpoint(count,line_forw[rsin] * g1_iae * 100 + 9.0);}
90       {drawpoint(count,line_refl[rsout] * g4_pae * 100 + 11.0);}
91     end;
92   elect_rec[i] := elect_rec[i] + electrode * (line_forw[rsin] + li
93   refl[point_2]);
94   rsin := rsin + lambda_by2;
95   if rsin > long then rsin := rsin - long;
96   rsout := rsout + lambda_by2;
97   if rsout > long then rsout := rsout - long;
98   point_1 := rsin + lambda_by4;
99   if point_1 > long then point_1 := point_1 - long;
100  point_2 := rsout + lambda_by4;
101  if point_2 > long then point_2 := point_2 - long;
102  end;
103  begin
104    rsin := rpoint_forw;
105    rsout := rpoint_back;
106    point_1 := rsin + lambda_by4;
107    if point_1 > long then point_1 := point_1 - long;
108    point_2 := rsout + lambda_by4;
109    if point_2 > long then point_2 := point_2 - long;
110    output := 0.0;
111    for i := 1 to no_rx do
112      begin
113        rexecute(1);
114        rexecute(-1);
115        output := output + elect_rec[i];
116      end;
117      output := output / lambda;
118      line_forw[point_1] := 0.0;
119      drawpoint(count,output - 10.0);
120    end;
121    label start,quit;
122    begin
123      for i := 1 to long do
124        begin
125          line_forw[i] := 0.0; line_refl[i] := 0.0;
126        end;
127        for i := 0 to lambda do input[i] := 0.0;
128        for i := 1 to 1 do input[i] := 1.0;
129        for i := 1 to no_rx do elect_rec[i] := 0.0;
130        lambda_by4 := lambda div 4;
131        lambda_by2 := lambda div 2;
132        tpoint_forw := 1;
133        tpoint_back := 1;
134        rpoint_forw := (no_tx + 1) * lambda;
135        rpoint_back := (no_tx + 1) * lambda;
136        input_pointer := 1;
137        sig_input := 0.0;
138        delta_t := 2 * 3.1415927 / lambda;
139        count := 0;
140        InitGraphic;
141        DefineWorld(1,0,20.0,long,-20.0);
142        DefineWindow(1,0,0,xmaxglb,ymaxglb);
143        SelectWorld(1);
144        selectwindow(1);
145        clearscreen;
146        limit := lambda * 10;
147        start: if keypressed then goto quit;
148        if count < limit then
149          input[input_pointer] := sin(count * delta_t) /
150          else input[input_pointer] := 0.0;
151          if count > 20 then
152            input[input_pointer] := 0.0;
153            drawpoint(count,input[input_pointer] + 15.0);
154            if count < limit then line_forw[rpoint_forw] := sin
155            * delta_t)
156            else line_forw[rpoint_forw] := 0.0;
157            transmit;
158            receive;
159            dec(tpoint_forw); if tpoint_forw < 1 then tpoint_forw
160            := tpoint_forw + long;
161            inc(tpoint_back); if tpoint_back > long then tpoint_b
162            := tpoint_back - long;
163            dec(rpoin_forw); if rpoin_forw < 1 then rpoin_forw
164            := rpoin_forw + long;
165            inc(rpoint_back); if rpoint_back > long then rpoint_b
166            := rpoint_back - long;
167            inc(count);
168            inc(input_pointer);
169            if input_pointer > lambda then input_pointer := input
170            := lambda;
171            goto start;
172            quit: read(key);
173            if upcase(key) = 'Y' then
174              begin
175                writeln(lst,'Rfe = ',g1_iae);
176                writeln(lst,'Ref = ',g2_iae);
177                writeln(lst,'Tfe = ',g3_pae);
178                writeln(lst,'Tef = ',g4_pae);
179                Hardcopy(false,1);
180              end;
181            leavegraphic;
182          end.

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้คัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ญ. โปรแกรม Editor ของโปรแกรมวิเคราะห์เน็ตเวิร์คด้วยสัญลักษณ์



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

1 {version 2.0}
2 Program Network_editor;
3 uses turbo3,crt,printer;
4 const pleng = 21;
5 term_max = 1200;
6 var
7   model_val : array[1..10] of real;
8   bran : array[1..2,1..20] of integer;
9   mess : array[0..term_max] of string[50];
10  mess_t1,mess_t2,H_MESS : string[100];
11  esc1,m_t,m_t2,m_t3,m_t4 : char;
12  curx,cury : byte;
13  chk2,cf3,ary_con,ary_index,max_mess,max_var : integer;
14  px,rd1,poin,pol,pol2,pol3,pol4,var_count,ra_max,pntr : integer;
15  instatus,ins2,ins3,chk : boolean;
16  file_name,file1,file2,file3 : text;
17  t_mel,t_me2 : string;
18  cn : array[1..20] of string[2];
19  v_n : array[1..30] of string[6];
20  vx,v_st,v_end : integer;
21  m_val : real;
22 procedure set_com;
23 begin
24   cn[1] := 'R'; cn[2] := 'C'; cn[3] := 'L';
25   cn[4] := '+'; cn[5] := 'Q'; cn[6] := 'J'; cn[7] := 'G';
26   {transistor}
27   v_n[1] := 'R1Rb'; v_n[2] := 'C1Cpi'; v_n[3] := 'R2Rpi'; v_n[4] := 'C2
28   v_n[5] := 'R3Ru'; v_n[6] := 'R4Ro'; v_n[7] := 'R5Rex'; v_n[8] := 'C3C
29   v_n[9] := 'R6Rc'; v_n[10] := 'G1Gm';
30   {mos transistor}
31   v_n[11] := 'C1Cgd'; v_n[12] := 'C2Cgs'; v_n[13] := 'C3Cgb'; v_n[14] :=
32   v_n[15] := 'C5Cdb'; v_n[16] := 'R1Ro'; v_n[17] := 'G1Gm1'; v_n[18] :=
33   v_n[19] := 'C1Cgss'; v_n[20] := 'C2Cgs'; v_n[21] := 'C3Cgd'; v_n[22]
34   v_n[23] := 'R2Rd'; v_n[24] := 'G1Gm';
35 end;
36 procedure Help_menu;
37 begin
38   textbackground(0); clrscr; window(41,1,80,20);
39   textbackground(4); textcolor(1); clrscr;
40   writeln('Help menu :20,memoval); writeln;
41   textcolor(14);
42   writeln('F1 : LOAD FILE F2 : SAVE FILE');
43   writeln('F3 : CLEAR MEMORY F4 : CHECK FORMAT');
44   writeln('F5 : TRANSISTOR F6 : MOS TRANSISTOR');
45   writeln('F7 : JFET F8 : ');
46   writeln('F9 : F10 : EXIT');
47   textcolor(1); writeln('Editor Format :28);
48   textcolor(14); writeln('*<comment> ');
49   writeln('R<Nbr> <nodel> <nodel>');
50   writeln('C<Nbr> <nodel> <nodel>');
51   writeln('L<Nbr> <nodel> <nodel>');
52   writeln('G<Nbr> <nodel> <nodel> <ctrl> <ctrl>');
53   writeln('J<Nbr> <nodel> <nodel> *current sou');
54   writeln('V<Nbr> <nodel> <nodel> *AC voltage sou');
55   writeln('P<Nbr> <nodel> *DC supply');
56   writeln('Q<Nbr> <B> <C> <E> <nodel> transistor');
57   writeln('M<Nbr> <G> <D> <B> <S> <nodel> MOS ');
58   writeln('F<Nbr> <G> <D> <S> <nodel> JFET');
59 end;
60 procedure i_status;
61 begin
62   curx := wherex; cury := wherey;
63   textcolor(4); gotoxy(1,23);
64   if instatus = true then write('Insert':35) else write('':35);
65   textcolor(14); gotoxy(curx,cury);
66 end;
67 procedure Edit;
68 begin
69   window(1,3,40,25); textbackground(0);
70   textcolor(14); clrscr; i_status;
71 end;
72 procedure Help2;
73 begin
74   textbackground(0); clrscr; window(41,1,80,20);
75   textbackground(4); textcolor(1); clrscr;
76   writeln(h_mess:20,' help menu'); writeln;
77   textcolor(14);
78   writeln('Pg up : Previous Pg Dn : Next');
79   writeln('Del : Delete Ins : Insert');
80   writeln('F1 : Save F2 : Quit');
81 end;
82 procedure form3;
83 var fm : char;
84 begin
85   mess_t1 := '';
86   repeat
87     gotoxy(25,1);
88     readln(mess_t1);
89   until mess_t1 <> '';
90   mess[ary_index] := mess_t1;
91   mess_t1 := concat(mess_t1,' ');
92   gotoxy(6,3);
93   cury := wherey;
94   for pol2 := pol3+1 to pol3+ra_max do
95     begin
96       cury := cury + 1;
97       repeat
98         gotoxy(6,cury); fm := readkey;
99         until (fm = 'N') or (fm = 'S') or (fm = 'O');
100        case fm of
101          'N' : begin write('Normal','Value ':16); readln(mess_t2);
102                  dec(vx);
103                  {str(m_val,mess_t2)} mess_t2 := concat(' ',mess
104                  mess[vx] := concat(mess[ary_index],' ',mess_t2);
105                  'S' : write('Short circuit');
106                  'O' : write('Open circuit');
107                end;
108                mess_t1 := concat(mess_t1,fm);
109                end;
110                mess[ary_index] := mess_t1;
111              end;
112              procedure form2;
113              var f2,f3,f4 : integer;
114              mf2 : string[39];
115              begin
116                gotoxy(10,3); cury := wherey; f4 := term_max+1;
117                for pol2 := pol3+1 to pol3+ra_max do
118                  begin
119                    cury := cury + 1;
120                    gotoxy(6,cury);
121                    if copy(mess[ary_index],pol2,1) = 'O' then write('Open circ
122                      else if copy(mess[ary_index],pol2,1) = 'S' then write('SH
123                      else begin write('Normal'); pol4 := pos(' ',mess[ary_in
124                        mess_t1 := copy(mess[ary_index],1,pol4);
125                        repeat dec(f4) until (copy(mess[f4],1,pol4) = mess_t1)
126                        if f4 >= vx then begin
127                          mess_t2 := copy(mess[f4],pol4+1,39);
128                          write(mess_t2:10); end;
129                        end;
130                      end;
131                    end;
132                  end;
133                procedure form1;
134                begin
135                  clrscr;
136                  pol3 := pos(' ',mess[ary_index]);
137                  writeln(ary_index,' TRANSISTOR MODEL ',copy(mess[ary_index],1,
138                  writeln('Normal (S)hort (O)pen circuit':30);
139                  writeln('Rb'); writeln('Cpi'); writeln('Rpi'); writel
140                  writeln('Ru'); writeln('Ro'); writeln('Rex'); writeln('Ccs');
141                  writeln('Rc'); writeln('Gm'); form2;
142                end;
143                procedure form4;
144                begin
145                  clrscr; pol3 := pos(' ',mess[ary_index]);
146                  writeln(ary_index,' MOS TRANSISTOR MODEL ',copy(mess[ary_index],
147                  writeln('Normal (S)hort (O)pen circuit':30);
148                  writeln; writeln('Cgd'); writeln('Cgs'); writeln('Cgb'); writ
149                  writeln('Cdb'); writeln('Ro'); writeln('Gmgs'); writeln('Gmb')
150                  form2;
151                end;
152                procedure form5;
153                begin
154                  clrscr; pol3 := pos(' ',mess[ary_index]);
155                  writeln(ary_index,' JFET ',copy(mess[ary_index],1,pol3));
156                  writeln('Normal (S)hort (O)pen circuit':30);
157                  writeln; writeln('Cgss'); writeln('Cgs'); writeln('Cgd'); writ
158                  writeln('Ro'); writeln('Gm'); form2;
159                end;
160                procedure e_mess;
161                begin
162                  window(1,1,40,2); textbackground(2); textcolor(1); clrscr; writ
163                  itor':23);
164                end;
165                procedure scr_ed;
166                begin
167                  case byte(m_t4) of
168                    81 : begin (pg dn) inc(ary_index);
169                          if ary_index > max_mess then ary_index := max_mess; end;
170                    73 : begin (pg up) dec(ary_index);
171                          if ary_index < 1 then ary_index := 1; end;
172                    83 : begin (del)
173                          pol := pos(' ',mess[ary_index]);
174                          mess_t1 := copy(mess[ary_index],1,pol); pol3 := term_max
175                          for pol2 := term_max down to vx do
176                            if mess_t1 <> copy(mess[pol2],1,pol) then begin
177                              mess[pol3] := mess[pol2]; dec(pol3); end;
178                          vx := pol3+1;
179                          for pol := ary_index to max_mess-1 do mess[pol] := mess
180                        mess[max_mess] := '';
181                        dec(max_mess);
182                        if max_mess < 0 then max_mess := 0;
183                        if ary_index > max_mess then ary_index := max_mess; end;
184                        82 : begin (ins) inc(max_mess); ary_index := max_mess; mess[a
185                        dex] := ' ';
186                        if pntr = 3 then form1 else if pntr = 4 then form4 else

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

86 form3; end;
87 end;
88 if ptr = 3 then form1 else if ptr = 4 then form4 else form5;
89 end;
90 procedure Error_mess(er_mess : integer);
91 begin
92   case er_mess of
93     1 : writeln('File not found');
94     240 : writeln('Write protect');
95     241 : writeln('Filename full');
96   end;
97   writeln('Press any key to continue');
98   repeat until keypressed;
99 end;
100 procedure file_in(s_f : integer);
101 label file_inend;
102 begin
103   repeat
104     case s_f of
105       1 : begin clrscr; writeln('Read file Enter <filename> ');
106             readln(mess_t1);
107             if length(mess_t1) = 0 then goto file_inend;
108           end;
109       2 : mess_t1 := 'symbolic.$$$';
110       3 : mess_t1 := 'trans.lib';
111       4 : mess_t1 := 'mostrans.lib';
112       5 : mess_t1 := 'jfet.lib';
113     end;
114     assign(file_name,mess_t1);
115     (Si-) reset(file_name) (Si+);
116     rd1 := ioresult;
117     if rd1 <> 0 then error_mess(rd1);
118     if (rd1 <> 0) and (s_f > 2) then goto file_inend;
119     until rd1 = 0;
120   file_inend; end;
121 procedure file_out(s_f : integer);
122 var se_int : integer;
123 label file_outend;
124 begin
125   repeat
126     case s_f of
127       1 : begin clrscr; writeln('Save file Enter <filename>'); readln
128             mess_t1;
129             if length(mess_t1) = 0 then goto file_outend; end;
130       2 : mess_t1 := 'symbolic.$$$';
131       4 : mess_t1 := 'trans.lib';
132       5 : mess_t1 := 'mostrans.lib';
133       6 : mess_t1 := 'jfet.lib';
134     end;
135     assign(file_name,mess_t1);
136     (Si-) rewrite(file_name) (Si+);
137     se_int := ioresult;
138     if se_int <> 0 then error_mess(se_int);
139     until se_int = 0;
140   file_outend; end;
141 procedure write_error(we : byte);
142 procedure w_e;
143 begin
144   writeln('ERROR INLINE : ',cf3:5); writeln(mess[cf3]);
145   writeln('Press anykey to continue ');
146   repeat until keypressed;
147   clrscr; writeln; writeln('ERROR MESSAGE':25);
148 end;
149 begin
150   window(41,21,80,25); textbackground(1); textcolor(15);
151   if we = 0 then clrscr;
152   writeln; writeln('ERROR MESSAGE':25); textcolor(4);
153   case we of
154     1 : w_e;
155     2 : begin write('COMPONENT '); w_e; end;
156     3 : begin write('NODE NUMBER '); w_e; end;
157     4 : begin clrscr; writeln;
158           writeln('FORMAT CHECK OK':30);
159           write('Press anykey to continue');
160           repeat until keypressed;
161           write_error(0); end;
162     5 : begin writeln('COMPONENT DUPLICATED'); w_e; end;
163     6 : begin writeln('G NODE ERROR'); w_e; edit; end;
164     7 : begin writeln('FLOAT VOLTAGE ERROR'); w_e; edit; end;
165     8 : begin writeln('Model not found'); w_e; end;
166   end;
167 end;
168 procedure home;
169 var esc2 : integer;
170 begin {home}
171   clrscr; i_status;
172   for esc2 := 1 to pleng do
173     if (esc2 <= max_mess) or (mess[esc2] <> '') then writeln(mess[esc
174   ary_index := 1; gotoxy(1,1);
175 end;
176 procedure load_file(s_f : integer);
177 var esc2 : integer;
178 begin
179   file_in(s_f); ary_index := 1; max_mess := 0;
180   if length(mess_t1) <> 0 then begin
181     for esc2 := 1 to term_max do mess[esc2] := '';
182     repeat
183       readln(file_name,mess[ary_index]);
184       ary_index := ary_index+1; max_mess := max_mess+1;

```

```

204 until eof(file_name);
205 close(file_name); end; end;
206 procedure save_file(s_f : integer);
207 begin
208   file_out(s_f);
209   if length(mess_t1) <> 0 then
210     begin
211       ary_index := 1;
212       for ary_con := 1 to max_mess do writeln(file_name,mess[ary
213     end;
214   if mess[max_mess+1] <> '' then writeln(file_name,mess[max
215   ));
216   close(file_name);
217 end;
218 end;
219 procedure write_e2(we2 : integer);
220 begin
221   write_error(1); window(1,3,40,25); textbackground(0); textcolor
222   i_status;
223   repeat until keypressed;
224   write_error(we2); edit; cf3 := max_mess; end;
225 procedure c_prc;
226 var cp1,cp2,cp3,cp4 : integer;
227 begin
228   save_file(2); file_in(ptr); max_mess := 0; help2;
229   mess[1] := '';
230   vx := term_max+1;
231   if rd1 = 0 then
232     begin
233       repeat
234         readln(file_name,t_me1);
235         if t_me1 <> '' then
236           begin
237             inc(max_mess); t_me2 := t_me1;
238             cp2 := pos(' ',t_me1); mess_t1 := copy(t_me1,1,cp
239             delete(t_me1,1,cp2); cp3 := pos(' ',t_me1); delet
240             cp2 := cp2 + cp3-1; mess[max_mess] := copy(' t_me2,
241             cp3);
242             cp2 := cp2 + cp3-1; mess[max_mess] := copy(' t_me2,
243             cp3);
244             repeat
245               mess_t2 := copy(t_me1,1,1);
246               if (mess_t2 = 'R') or (mess_t2 = 'C') then cp
247             for cp1 := 1 to cp2 do
248               begin
249                 cp3 := pos(' ',t_me1); delete(t_me1,
250               end;
251               cp3 := pos(' ',t_me1);
252               if cp3 = 0 then cp3 := length(t_me1) + 1;
253               mess_t2 := copy(t_me1,1,cp3-1);
254               delete(t_me1,1,cp3);
255               dec(vx); mess[vx] := concat(mess_t1,' ',mess_
256               until (t_me1 = '') or (t_me1 = ' ');
257             end;
258           until eof(file_name);
259           close(file_name);
260         end;
261         ary_index := max_mess;
262       end;
263       procedure check_format;
264       var cf1,cf2 : integer;
265       label cfend,cfst;
266       procedure cv_model(cvlop : integer);
267       begin
268         for po3 := 1 to cvlop + 1 do
269           begin
270             repeat delete(mess_t1,1,1); until copy(mess_t1,1,1) <> ' ';
271             po1 := pos(' ',mess_t1);
272             if po1 = 0 then po1 := length(mess_t1) + 1;
273             mess_t2 := copy(mess_t1,1,po1-1);
274             delete(mess_t1,1,po1-1);
275             val(mess_t2,po1,po2);
276             if (po2 <> 0) and (po3 <= cvlop) then write_error(3);
277           end;
278         repeat delete(mess_t1,1,1); until copy(mess_t1,1,1) <> ' ';
279         po1 := pos(' ',mess_t1);
280         if po1 = 0 then po1 := length(mess_t1) + 1;
281         mess_t2 := copy(mess_t1,1,po1-1);
282         po2 := length(mess_t2);
283       end;
284     begin
285       save_file(2);
286       clrscr;
287       {Check Q M F}
288       if mess[max_mess+1] <> '' then inc(max_mess);
289       var_count := max_mess;
290       for cf3 := 1 to var_count do
291         begin
292           if copy(mess[cf3],1,1) = 'Q' then
293             begin
294               v_st := 1; v_end := 10;
295               file_in(3);
296               mess_t1 := concat(' ',copy(mess[cf3],2,length(mess[cf3]))
297               cv_model(3);
298               cf1 := 0;
299               repeat readln(file_name,t_me1);
300                 if (copy(t_me1,1,po2) = mess_t2) then cf1 := 1;
301               until eof(file_name) or (cf1 = 1);
302               close(file_name);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

if cfl <> 1 then begin write_error(8); goto cfend; end;
end;
if copy(mess[cf3],1,1) = 'M' then
begin
v st := 11; v_end := 18;
file_in(4);
mess_t1 := concat(' ',copy(mess[cf3],2,length(mess[cf3])));
cv_model(4);
cfl := 0;
repeat readln(file_name,t_mel);
if (copy(t_mel,1,pol) = mess_t2) then cfl := 1;
until eof(file_name) or (cfl = 1);
close(file_name);
if cfl <> 1 then begin write_error(8); goto cfend; end;
end;
if copy(mess[cf3],1,1) = 'F' then
begin
v st := 19; v_end := 24;
file_in(5);
mess_t1 := concat(' ',copy(mess[cf3],2,length(mess[cf3])));
cv_model(3);
close(file_name);
end;
end;
{find node of each line}
for cf3 := 1 to var_count do
begin
if copy(mess[cf3],1,1) <> '*' then begin
mess_t1 := concat(mess[cf3], ' ');
cfl := pos(' ',mess_t1);
mess_t2 := copy(mess_t1,2,cfl-2);
if copy(mess[cf3],1,1) <> 'V' then mess[cf3] := copy(mess[cf3]
delete(mess_t1,1,cfl);
val(mess_t2,pol,cfl);
if (cfl <> 0) or ((pol > 999) and (cfl = 0)) then begin write
cfend; end;
cfl := pos(' ',mess_t1);
mess_t2 := copy(mess_t1,1,cfl-1);
delete(mess_t1,1,cfl);
val(mess_t2,pol,cfl);
if (cfl <> 0) or ((pol > 999) and (cfl = 0)) then begin write
cfend; end;
cfl := pos(' ',mess_t1);
mess_t2 := copy(mess_t1,1,cfl-1);
delete(mess_t1,1,cfl);
val(mess_t2,pol,cfl);
if (cfl <> 0) or ((pol > 999) and (cfl = 0)) then begin write
cfend; end;
if (copy(mess[cf3],1,1) = 'P') then
goto cfst;
end;
end;
{check repeat component}
for cf3 := 0 to max_mess-1 do
for pol := cf3+1 to max_mess do
if mess[cf3] = mess[pol] then write_error(2);
{Edit voltage source}
cf3 := 1;
repeat
vx := cf3;
if copy(mess[cf3],1,1) = 'V' then
begin
mess[cf3] := concat(mess[cf3], ' ');
cfl := length(mess[cf3]);
repeat
delete(mess[cf3],cfl,1);
cfl := length(mess[cf3]);
until copy(mess[cf3],cfl,1) <> ' ';
mess_t2 := concat(mess[cf3], 'X');
pol := pos(' ',mess_t2);
mess_t1 := copy(mess_t2,2,pol-2);
val(mess_t1,pol,cfl);
if cfl <> 0 then begin write_error(2); goto cfend; end; {ch
delete(mess_t2,1,pol);
cfl := length(mess_t2);
mess_t1 := copy(mess_t2,1,cfl);
pol := pos(' ',mess_t1);
mess_t1 := copy(mess_t2,1,pol-1);
val(mess_t1,pol,cfl);
if cfl <> 0 then begin write_error(2); goto cfend; end; {no
delete(mess_t2,1,pol);
cfl := length(mess_t2);
mess_t1 := copy(mess_t2,1,cfl);
pol := pos(' ',mess_t1);
mess_t1 := copy(mess_t2,1,pol-1);
val(mess_t1,pol,cfl);
if cfl <> 0 then begin write_error(2); goto cfend; end; {no
end;
inc(cf3);
until cf3 > max_mess;
write_error(4); chk2 := 0;
cfend; end;
procedure ex_lib;
begin
repeat dec(po3);
until copy(mess[po3],1,pol) = mess_t1;

```

```

mess_t2 := copy(mess[po3],pol+1,39);
write(file_name,mess_t2);
end;
472
473 procedure escape;
474 var esc2,esc3 : integer;
475 escchar : char;
476 begin
477 read(kbd,esc1);
478 case byte(esc1) of
479 71 : home;
480 79 : begin {end} clrscr; i_status; ary_index := max_mess;
481
482 esc2 := ary_index - pleng;
483 if esc2 < 1 then esc2 := 1;
484 for esc3 := esc2 to max_mess do writeln(mess[esc3]);
if mess[ary_index] <> ' ' then write(mess[ary_index]);
end;
73 : begin {pg up} ary_index := ary_index - wherey - pleng;
if ary_index < 1 then ary_index := 1;
clrscr; i_status;
for esc2 := ary_index to ary_index+pleng do
if (esc2 < max_mess+1) or (mess[esc2] <> ' ')
n writeln(mess[esc2]);
gotoxy(1,1);
end;
493 81 : begin {pg down} esc2 := pleng - wherey; ary_index :=
y_index + pleng - wherey;
if ary_index > (max_mess + 1 - pleng) then ary_index :=
max_mess+1-pleng;
if ary_index < 1 then ary_index := 1;
clrscr; i_status;
for esc2 := ary_index to ary_index+pleng-1 do
if (esc2 < max_mess+1) or (mess[esc2] <> ' ')
n writeln(mess[esc2]);
gotoxy(1,1);
end;
500
501 82 : begin {ins} instatus := not(instatus); i_status; e
502 83 : begin {del} curx := wherex; cury := where; delete
[ary_index],curx,1);
gotoxy(1,cury); write(' ');
gotoxy(1,cury); write(mess[ary_index]); gotoxy(1,cury);
end;
506
75 : begin {left} curx := wherex; cury := wherey; curx :=
if curx < 1 then curx := 1;
gotoxy(curx,cury);
end;
510
77 : begin {right} curx := wherex + 1; cury := wherey;
if curx > length(mess[ary_index]) then curx :=
gotoxy(curx,cury);
end;
513
72 : begin {up} cury := wherey; cury := cury - 1; ary_index :=
ary_index - 1;
if ary_index < 1 then ary_index := 1;
if cury < 1 then begin clrscr;
for esc2 := ary_index to ary_index + pleng do
if esc2 <= max_mess then writeln(mess[esc2]);
i_status; cury := 1; end;
gotoxy(1,cury);
end;
521
60 : begin {down} cury := wherey; inc(ary_index);
if ary_index > max_mess+1 then ary_index := max_mess+1;
else gotoxy(1,cury+1);
end;
526
59 : begin load_file(1); home; end;
60 : begin save_file(1); home; end;
528
68 : exit;
63 : begin {Transistor} h_mess := 'Transistor'; ra_max :=
c_prc; form1;
repeat repeat read(kbd,m_t4); until byte(m_t4) =
m_t4 := readkey;
532 case byte(m_t4) of
533 81,73,83,82 : scr_ed;
534 59 : begin {save} file_out(4);
for ary_index := 1 to max_mess do
begin write(file_name,mess[ary_index]);
pol := pos(' ',mess[ary_index]);
cn[1] := 'R1'; bran[1,1] := 1; bran[2,1] := 4;
cn[2] := 'C1'; bran[1,2] := 4; bran[2,2] := 6;
cn[3] := 'R2'; bran[1,3] := 4; bran[2,3] := 6;
cn[4] := 'C2'; bran[1,4] := 4; bran[2,4] := 5;
cn[5] := 'R3'; bran[1,5] := 4; bran[2,5] := 5;
cn[6] := 'R4'; bran[1,6] := 5; bran[2,6] := 6;
cn[7] := 'R5'; bran[1,7] := 3; bran[2,7] := 6;
cn[8] := 'C3'; bran[1,8] := 0; bran[2,8] := 5;
cn[9] := 'R6'; bran[1,9] := 2; bran[2,9] := 5;
cn[10] := 'G1'; bran[1,10] := 5; bran[2,10] := 6;
bran[1,11] := 4; bran[2,11] := 6;
for po2 := 1 to 10 do
begin
mess_t1 := copy(mess[ary_index],pol+po2,1);
if mess_t1 = '0' then begin bran[1,po2] := 8; bran[2,po2] := 8; end;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

554 if mess_t1 = 'S' then begin po3 := bran[1,po2]; po4
555 := bran[2,po2];
556 if po3 > po4 then begin po4 := bran[1,po2]; po3
557 := bran[2,po2]; end;
558 for ary_con := 1 to 11 do begin
559 if bran[1,ary_con] = po4 then bran[1,ary_con]
560 := po3;
561 if bran[2,ary_con] = po4 then bran[2,ary_con]
562 := po3;
563 end; end;
564 po3 := term_max+1;
565 po1 := pos(' ',mess[ary_index]);
566 mess_t1 := copy(mess[ary_index],1,pol);
567 for po2 := 1 to ra_max do begin
568 if bran[1,po2] <> bran[2,po2] then begin
569 write(file_name,' ',cn[po2],', ',bran[1,po2],', '
570 if po2 = 10 then write(file_name,' ',bran[1,11]
571 ex_lib;
572 end;
573 end;
574 writeln(file_name,' ');
575 end;
576 close(file_name);
577 end;
578 end;
579 form1;
580 until byte(m_t4) = 60;
581 load_file(2); set_com; help_menu; edit; end;
582 64 : begin (MOS Transistor) h_mess := 'MOS Transistor'; pntr := 4;
583 max := 8;
584 'c_prc; form4;
585 repeat
586 repeat
587 read(kbd,m_t4);
588 until byte(m_t4) = 27;
589 m_t4 := readkey;
590 case byte(m_t4) of
591 81,73,83,82 : scr_ed;
592 59 : begin (save)
593 file_out(5); for ary_index := 1 to max_mess do begin
594 write(file_name,mess[ary_index]);
595 pol := pos(' ',mess[ary_index]);
596 cn[1] := 'C1'; bran[1,1] := 1; bran[2,1] := 2;
597 cn[2] := 'C2'; bran[1,2] := 1; bran[2,2] := 4;
598 cn[3] := 'C3'; bran[1,3] := 1; bran[2,3] := 3;
599 cn[4] := 'C4'; bran[1,4] := 3; bran[2,4] := 4;
600 cn[5] := 'C5'; bran[1,5] := 2; bran[2,5] := 3;
601 cn[6] := 'R1'; bran[1,6] := 2; bran[2,6] := 4;
602 cn[7] := 'G1'; bran[1,7] := 2; bran[2,7] := 4;
603 cn[8] := 'G2'; bran[1,8] := 2; bran[2,8] := 4;
604 bran[1,9] := 1; bran[2,9] := 4;
605 bran[1,10] := 3; bran[2,10] := 4;
606 for po2 := 1 to 8 do begin
607 mess_t1 := copy(mess[ary_index],pol+po2,1);
608 if mess_t1 = 'O' then
609 begin
610 bran[1,po2] := 8;
611 bran[2,po2] := 8;
612 end;
613 if mess_t1 = 'S' then
614 begin
615 po3 := bran[1,po2];
616 po4 := bran[2,po2];
617 if po3 > po4 then
618 begin
619 po4 := bran[1,po2];
620 po3 := bran[2,po2];
621 end;
622 for ary_con := 1 to 10 do
623 begin
624 if bran[1,ary_con] = po4 then bran[1,ary_con]
625 := po3;
626 if bran[2,ary_con] = po4 then bran[2,ary_con]
627 := po3;
628 end;
629 end;
630 po3 := term_max+1;
631 po1 := pos(' ',mess[ary_index]);
632 mess_t1 := copy(mess[ary_index],1,pol);
633 for po2 := 1 to ra_max do
634 begin
635 if bran[1,po2] <> bran[2,po2] then
636 begin
637 write(file_name,' ',cn[po2],', ',bran[1,po2],', '
638 if po2 = 7 then write(file_name,' ',bran[1,9],', '
639 if po2 = 8 then write(file_name,' ',bran[1,10],', '
640 ex_lib;
641 end;
642 end;
643 writeln(file_name,' ');
644 end;
645 close(file_name);
646 end;
647 end;
648 form1;
649 until byte(m_t4) = 60;
650 load_file(2); set_com; help_menu; edit; end;
651 65 : begin (JFET) h_mess := 'JFET'; pntr := 5; ra_max := 6;
652 orms;
653 repeat
654 repeat read(kbd,m_t4); until byte(m_t4) = 27;
655 m_t4 := readkey;
656 case byte(m_t4) of
657 81,73,83,82 : scr_ed;
658 59 : begin (save) file_out(6);
659 for ary_index := 1 to max_mess do
660 begin write(file_name,mess[ary_index]);
661 pol := pos(' ',mess[ary_index]);
662 cn[1] := 'C1'; bran[1,1] := 1; bran[2,1] := 0;
663 cn[2] := 'C2'; bran[1,2] := 1; bran[2,2] := 3;
664 cn[3] := 'C3'; bran[1,3] := 1; bran[2,3] := 4;
665 cn[4] := 'R1'; bran[1,4] := 2; bran[2,4] := 4;
666 cn[5] := 'R2'; bran[1,5] := 3; bran[2,5] := 4;
667 cn[6] := 'G1'; bran[1,6] := 4; bran[2,6] := 3;
668 bran[1,7] := 1; bran[2,7] := 4;
669 for po2 := 1 to 6 do
670 begin
671 mess_t1 := copy(mess[ary_index],pol+po2,1);
672 if mess_t1 = 'O' then begin bran[1,po2] := 8
673 ,po2 := 8; end;
674 if mess_t1 = 'S' then begin po3 := bran[1,po2]
675 := bran[2,po2];
676 if po3 > po4 then begin po4 := bran[1,po2]
677 := bran[2,po2]; end;
678 for ary_con := 1 to 7 do begin
679 if bran[1,ary_con] = po4 then bran[1,ary_con]
680 := po3;
681 if bran[2,ary_con] = po4 then bran[2,ary_con]
682 := po3;
683 end;
684 end;
685 po3 := term_max+1;
686 po1 := pos(' ',mess[ary_index]);
687 mess_t1 := copy(mess[ary_index],1,pol);
688 for po2 := 1 to ra_max do
689 begin
690 if bran[1,po2] <> bran[2,po2] then begin
691 write(file_name,' ',cn[po2],', ',bran[1,po2],', '
692 if po2 = 6 then write(file_name,' ',bran[1,9],', '
693 ex_lib;
694 end;
695 end;
696 writeln(file_name,' ');
697 end;
698 close(file_name);
699 end;
700 end;
701 form5;
702 until byte(m_t4) = 60;
703 load_file(2); set_com; help_menu; edit; end;
704 62 : begin check_format; edit; load_file(2); home; end;
705 61 : begin
706 escchar := ' ';
707 repeat clrscr; textcolor(4); writeln('CLEAR MEMORY':25
708 eln('Are you ok (Y/N) ?');
709 textcolor(14); escchar := readkey; escchar := upcase
710 r);
711 until (escchar = 'Y') or (escchar = 'N');
712 if escchar = 'Y' then
713 begin
714 for ary_index := 1 to term_max do mess[ary_index]
715 := '';
716 max_mess := 0; clrscr;
717 end;
718 end;
719 end;
720 end;
721 begin
722 (generate space)
723 set_com; Help_menu;
724 for ary_index := 1 to term_max do mess[ary_index] := '';
725 ary_index := 1; max_mess := 0; write_error(0); instatus :=
726 e_mess; edit; home; mess_t1 := ' ';
727 mess[0] := ' ';
728 esc1 := '0';
729 repeat
730 read(kbd,m_t);
731 case byte(m_t) of
732 25 : begin
733 if ary_index <= max_mess then begin
734 for ary_con := ary_index to max_mess do
735 mess[ary_con] := mess[ary_con+1];
736 dec(max_mess); ins2 := instatus;
737 instatus := false; i_status; delline;
738 instatus := ins2; i_status; end;
739 if ary_index = (max_mess+1) then begin
740 mess[max_mess+1] := ' '; dec(ary_index);
741 dec(max_mess); ins2 := instatus;
742 cury := wherey;
743 instatus := false; i_status; delline;
744 instatus := ins2; i_status; gotoxy(1,c

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษานี้เท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์อื่นใด

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

731         end;
732     27 : escape;
733     13 : begin
734         cury := wherey;
735         inc(ary_index);
736         if ary_index > max_mess + 1 then begin
737             max_mess := max_mess+1; gotoxy(1,23); wr
738             te('':35);
739             gotoxy(1,cury); writeln(mess[ary_index-1
740             else begin
741                 if (instatus = true) and (wherex = 1
742                 begin
743                     for ary_con := max_mess+1 do
744                         mess[ary_con+1] := mess[
745                         mess[ary_con] := ''; gotoxy(
746                         insline; max_mess := max_mes
747                     end else begin gotoxy(1,cury); wr
748                 end; end;
749             end;
750             8 : begin {del} curx := wherex; cury := wherey; delet
751             (mess[ary_index],curx-1,1);
752                 dec(curx);
753                 if curx < 1 then curx := 1;
754                 gotoxy(1,cury); write('
755             ');
756                 gotoxy(1,cury); write(mess[ary_index]); gotoxy(cu
757             x,cury);
758             end;
759             else begin
760                 if (byte(m_t) > 31) and (byte(m_t) < 128) then
761                     begin curx := wherex; cury := wherey;
762                     mess_t1 := copy(mess[ary_index],1,curx-1);
763                     mess_t2 := copy(mess[ary_index],curx,length(
764                     mess[ary_index])-curx+1);
765                     if instatus = false then
766                         mess_t2 := copy(mess_t2,2,length(mess_t2)
767                     1);
768                     mess[ary_index] := concat(mess_t1,m_t,mess_t
769                     );
770                     gotoxy(1,cury); write('
771                     ');
772                     gotoxy(1,cury); write(mess[ary_index]); goto
773                     y(curx+1,cury);
774                     end;
775                 end;
776             end;
777             until byte(esc1) = 68;
778             clrscr;
779         end.

```

ภาคผนวก ฅ. โปรแกรมวิเคราะห์เน็ตเวิร์คด้วยสัญลักษณ์



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้


```

192 file_in(3);
193 mess_t1 := concat(' ', copy(mess, 2, 100));
194 cv_model(3);
195 repeat readln(file_name, t_mel);
196 until eof(file_name) or (copy(t_mel, 1, po2) = mess_t2);
197 if ana = 'DC' then begin
198   bran[1, 107] := node[2]; node[2] := 900 + node[2];
199   cv_data(3); inc(max_mess); po4 := pos(' ', mess);
200   str(bran[1, 1], t_mel);
201   mess := concat('vbe', copy(mess, 2, po4-2), ' ', t_mel, ' ');
202   str(bran[1, 2], t_mel);
203   mess := concat(mess, t_mel, ' ');
204   end else cv_data(3);
205   close(file_name);
206   goto format_g1;
207 end else
208 if mess_t2 = 'M' then
209   begin
210     v_st := 11; v_end := 18;
211     file_in(4);
212     mess_t1 := concat(' ', copy(mess, 2, length(mess)));
213     cv_model(4);
214     repeat readln(file_name, t_mel);
215     until eof(file_name) or (copy(t_mel, 1, po2) = mess_t2);
216     cv_data(4);
217     close(file_name);
218     goto format_g1;
219   end else
220 if mess_t2 = 'P' then
221   begin
222     v_st := 19; v_end := 24;
223     file_in(5);
224     mess_t1 := concat(' ', copy(mess, 2, length(mess)));
225     cv_model(5);
226     cv_data(5);
227     close(file_name);
228     goto format_g1;
229   end else
230 if mess_t2 <> '' then writeln(file2, mess);
231 format_g1:
232 until eof(file1);
233 close(file1);
234 close(file2);
235 file_in(2); (open file)
236 cf3 := 0;
237 max_mess := 0;
238 (find node of each line)
239 repeat
240   readln(file_name, mess);
241   if copy(mess, 1, 1) <> '' then
242     begin
243       mess_t1 := mess;
244       cf1 := pos(' ', mess_t1);
245       name[cf3+1] := copy(mess, 1, cf1-1);
246       delete(mess_t1, 1, cf1);
247       t_mel := mess_t1;
248       cf1 := pos(' ', mess_t1);
249       mess_t2 := copy(mess_t1, 1, cf1-1);
250       delete(mess_t1, 1, cf1);
251       val(mess_t2, bran[1, cf3+1], cf1);
252       if cf1 <> 0 then begin write_error(1); goto cfend; end;
253       if (copy(mess, 1, 1) = 'P') then
254         begin bran[2, cf3+1] := 0;
255           cf1 := pos(' ', mess_t1);
256           if cf1 = 0 then cf1 := length(mess)+1;
257           delete(mess_t1, 1, cf1);
258           goto cfst; end;
259           cf1 := pos(' ', mess_t1);
260           if cf1 = 0 then cf1 := length(mess)+1;
261           mess_t2 := copy(mess_t1, 1, cf1-1); delete(mess_t1, 1, cf1);
262           val(mess_t2, bran[2, cf3+1], cf1);
263           if cf1 <> 0 then begin write_error(1); goto cfend; end;
264           cfst: name[cf3+1] := concat(name[cf3+1], ' ', mess_t1);
265           inc(cf3);
266           name[cf3+1] := name[cf3];
267           inc(max_mess);
268         end;
269       until eof(file_name);
270       close(file_name);
271       (cf3 = number of branch, max_mess = number of element)
272       node_max := cf3;
273       for cf1 := 1 to node_max do
274         if (copy(name[cf1], 1, 1) = 'P') and (ana = 'AC') then begin
275           po4 := bran[1, cf1];
276           for pol := 1 to node_max do begin
277             if bran[1, pol] = po4 then bran[1, pol] := 0;
278             if bran[2, pol] = po4 then bran[2, pol] := 0; end;
279           end;
280         (change sign )
281         poin := 1;
282         for cf3 := 1 to node_max do
283           begin
284             if bran[1, cf3] <> bran[2, cf3] then
285               begin
286                 cf1 := pos(' ', name[cf3]);
287                 mess_t1 := copy(name[cf3], 1, cf1);
288                 mess_t2 := copy(name[cf3], cf1+1, length(name[cf3]));
289                 if bran[1, cf3] < bran[2, cf3] then
290                   begin
291                     bran[2, cf3] := bran[1, cf3];
292                     bran[1, cf3] := copy(mess, 2, 100);
293                     end else begin
294                       name[cf3] := concat(mess_t1, ' ', mess_t2);
295                       cf1 := bran[1, cf3]; po4 := bran[2, cf3];
296                       bran[1, poin] := po4; bran[2, poin] := cf1;
297                       end;
298                     inc(poin);
299                     end;
300                   end;
301                 (poin-1 := number of branch)
302                 max_var := poin-1;
303                 poin := 1;
304                 (count number of branch that had not overlap)
305                 for po3 := 1 to max_var do
306                   begin
307                     ary_con := 0;
308                     for po2 := 1 to poin-1 do
309                       if (bran[1, po3] = bran[1, po2]) and (bran[2, po3] = bran[2, po2]) then
310                         begin
311                           then ary_con := po2;
312                           if ary_con = 0 then begin
313                             bran[1, poin] := bran[1, po3]; bran[2, poin] := bran[2, po3];
314                             ary_con := poin; inc(poin); end;
315                             node[po3] := ary_con;
316                           end;
317                           (poin-1 := number of branch that not overlap)
318                           node[x] = branch number of that branch)
319                           max_mess := max_var;
320                           (check repeated component)
321                           cf3 := 0;
322                           for pol := 1 to max_mess-1 do
323                             for po2 := pol+1 to max_mess do
324                               if (name[pol] = name[po2]) and (copy(name[pol], 1, 1) = copy(name[po2], 1, 1)) then
325                                 begin
326                                   if cf3 <> 0 then
327                                     begin
328                                       write_error(5);
329                                       chk2 := 1;
330                                       goto cfend;
331                                     end;
332                                     assign(file_name, 'addmit');
333                                     rewrite(file_name);
334                                     for pol := 1 to max_mess do writeln(file_name, node[pol]);
335                                     close(file_name);
336                                     (count all node)
337                                     node_max := 1; pol := 0;
338                                     (find node number that <> 0)
339                                     repeat
340                                       inc(pol); node[1] := bran[1, pol];
341                                       if node[1] = 0 then node[1] := bran[2, pol];
342                                       until (pol >= poin) or (node[1] <> 0);
343                                       (read all node number that not overlap save into node[1])
344                                       node_max := pol;
345                                       for pol := 1 to poin-1 do
346                                         begin
347                                           po3 := 0;
348                                           for po2 := 1 to node_max do if bran[1, pol] = node[po2] then
349                                             if (po3 = 0) and (bran[1, pol] <> 0) then begin
350                                               inc(node_max); node[node_max] := bran[1, pol]; end;
351                                             po3 := 0;
352                                             for po2 := 1 to node_max do if bran[2, pol] = node[po2] then
353                                               if (po3 = 0) and (bran[2, pol] <> 0) then begin
354                                                 inc(node_max); node[node_max] := bran[2, pol]; end;
355                                               po3 := 0;
356                                               (node_max = number of node)
357                                               write_error(4); chk2 := 0;
358                                             end;
359                                             (Bran[1,x], bran[2,x] = branch of branchname node[x])
360                                             name[1..max_mess] = variable name and value keep in form
361                                             sign value)
362                                             procedure gen_matrix;
363                                             var se_int, gm0, gm1, gm2, gm3, gm4 : integer;
364                                             label gend;
365                                             begin
366                                               (save all the name of node)
367                                               for ary_con := 1 to node_max do node2[ary_con] := node[ary_con];
368                                               for ary_con := 1 to node_max do
369                                                 for pol := 1 to poin-1 do
370                                                   inci[ary_con, pol] := 0;
371                                                 for ary_con := 1 to node_max do
372                                                   for pol := 1 to poin-1 do
373                                                     begin
374                                                       if bran[1, pol] = node[ary_con] then po2 := 1
375                                                       else if bran[2, pol] = node[ary_con] then po2 := -1
376                                                       else
377                                                         inci[ary_con, pol] := po2;
378                                                       end;
379               (generate matrix G read branch index)
380               assign(file_name, 'ADDMIT');
381               reset(file_name);
382               pol := 0;
383               repeat inc(pol) readln(file_name, mess_t1); val(mess_t1, node[p
384               2);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้


```

569 begin
570   (open input file1);
571   assign(file3,'AGAT'); rewrite(file3);
572   assign(file2,'ag'); reset(file2);
573   repeat
574     readln(file2,t_mel); convert(term_max);
575     for pol := 1 to node_max do
576       if incl(pol,f_equer_max)^.col_no <> 0 then
577         begin
578           write(file3,f_equer_max^.row_no,':',pol,':1:');
579           po2 := incl(pol,f_equer_max)^.col_no * f_equer_max^.equ2[
580
581           if po2 > 0 then write(file3,':') else write(file3,'-:');
582           for po3 := 2 to max_var+2 do
583             begin
584               po2 := f_equer_max^.equ2[po3];
585               write(file3,po2,':');
586             end;
587             writeln(file3);
588           end;
589           until eof(file2);
590           close(file2);
591           close(file3);
592 end;
593 procedure multiply_agvs;
594 begin
595   (open input file1);
596   assign(file1,'AG'); reset(file1);
597   fir_st := 1;
598   repeat readln(file1,t_mel); convert(fir_st); inc(fir_st);
599   until (eof(file1));
600   fir_end := fir_st - 1;
601   sec_st := fir_st;
602   assign(file2,'VOLSOU'); reset(file2);
603   repeat
604     readln(file2,t_mel); convert(fir_st);
605     inc(fir_st);
606     until eof(file2) or (fir_st = term_max-1);
607     close(file2);
608     sec_end := fir_st - 1;
609     assign(file1,'AGVS');
610     rewrite(file1);
611     (start multiply)
612     for i := 1 to fir_end do
613       for j := sec_st to sec_end do
614         begin
615           if f_equer[i]^col_no = f_equer[j]^row_no then
616             begin
617               (scan for position of V)
618               pol := 1;
619               repeat inc(pol) until
620                 f_equer[j]^equ2[pol] <> 0;
621               inc(pol);
622               if f_equer[i]^equ2[pol] <> 0 then
623                 begin
624                   write(file1,f_equer[i]^row_no,':',f_equer[j]^co
625                   pol := f_equer[i]^equ2[1]*f_equer[j]^equ2[1];
626                   if pol > 0 then write(file1,':') else write(
627                   for k := 2 to max_var + 2 do
628                     write(file1,f_equer[i]^equ2[k]+f_equer[j]^equ2[
629                     writeln(file1);
630                   end;
631                 end;
632               close(file1);
633             end;
634           end;
635         end;
636       end;
637     end;
638   procedure multiply_gvs;
639   begin
640     (open input file1);
641     assign(file1,'admit'); reset(file1);
642     fir_st := 1;
643     repeat readln(file1,t_mel);
644       convert(fir_st);
645       inc(fir_st);
646       until (eof(file1));
647       close(file1);
648       fir_end := fir_st - 1;
649       sec_st := fir_st;
650       assign(file2,'VOLSOU'); reset(file2);
651       repeat
652         readln(file2,t_mel);
653         convert(fir_st);
654         inc(fir_st);
655         until eof(file2) or (fir_st = term_max-1);
656         close(file2);
657         sec_end := fir_st - 1;
658         assign(file1,'GVS');
659         rewrite(file1);
660         (start multiply)
661         for i := 1 to fir_end do
662           for j := sec_st to sec_end do
663             begin
664               if f_equer[i]^col_no = f_equer[j]^row_no then
665                 begin
666                   write(file1,f_equer[i]^row_no,':',f_equer[j]^co
667                   pol := f_equer[i]^equ2[1]*f_equer[j]^equ2[1];
668
669   if pol < 0 then write(file1,':') else
670   for k := 2 to max_var + 2 do
671     write(file1,f_equer[i]^equ2[k]+f_equer[j]^equ2[j]^
672   writeln(file1);
673   end;
674   assign(file2,'cursou'); reset(file2);
675   if not(eof(file2)) then
676     begin
677       repeat readln(file2,t_mel); writeln(file1,t_mel);
678       until (eof(file2));
679       close(file2);
680       end;
681       close(file1);
682     end;
683   procedure analysis;
684   begin
685     check format;
686     if chk2 = 0 then begin
687       gen_matrix;
688       (multiply AG)
689       multiply_A('admit','AG'); (give AG file)
690       multiply_AGAT; (give AGT file)
691       multiply_A('cursou','AJ'); (give AJ file)
692       multiply_agvs; (give AGVS file)
693       multiply_gvs; (give GVS file = js - GVs)
694     end;
695   end;
696   procedure ex_lib;
697   begin
698     repeat dec(po3);
699     until copy(mess[po3],1,pol) = mess_t1;
700     mess_t2 := copy(mess[po3],pol+1,39);
701     write(file_name,mess_t2);
702   end;
703   procedure prd node;
704   var prd : boolean;
705   begin
706     po2 := 0;
707     for pol := 1 to poin-1 do
708       begin
709         prd := false;
710         if bran[1,pol] <> 0 then
711           begin for po3 := 1 to po2 do
712             if bran[1,pol] = node[po3] then prd := true;
713             if prd = false then begin inc(po2); node[po2] := b
714           end;
715         prd := false;
716         if bran[2,pol] <> 0 then
717           begin for po3 := 1 to po2 do
718             if bran[2,pol] = node[po3] then prd := true;
719             if prd = false then begin inc(po2); node[po2] := b
720         end;
721       end;
722     end;
723   procedure read bran;
724   var rb1,rb2,rb3 : integer;
725   begin
726     pol := pos(' ',name[1]);
727     mess := copy(name[1],pol+1,50);
728     pol := pos(' ',mess);
729     val(copy(mess,1,pol-1),po2,po4);
730     delete(mess,1,pol);
731     pol := pos(' ',mess);
732     if pol = 0 then pol := length(mess) + 1;
733     val(copy(mess,1,pol-1),po3,po4);
734     po4 := 1;
735     if po3 < po2 then
736       begin
737         po1 := po2;
738         po2 := po3;
739         po3 := po1;
740         po4 := -1;
741       end;
742     if po2 <> 0 then po2 := node[po2];
743     if po3 <> 0 then po3 := node[po3];
744     if po2 = 0 then
745       begin
746         po2 := po3;
747         po3 := 0;
748       end;
749   end;
750   procedure write_sec;
751   var ws : integer;
752   begin
753     if set_equer[j]^equ3 <> 0 then
754       begin
755         if po2 <> 0 then
756           begin
757             write(file3,set_equer[j]^row_no,':');
758             write(file3,po2,':1:');
759             write(file3,set_equer[j]^equ3 * -1 * po4,':');
760             for ws := 1 to max_var + 1 do
761               write(file3,set_equer[j]^equ2[ws],':');
762             writeln(file3);
763           end;
764         end;
765       end;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

761 if po3 <> 0 then
762   write(file3,set_equ[j]^row_no,':');
763   write(file3,po3,':1:');
764   write(file3,set_equ[j]^equ3 * po4,':');
765   for ws := 1 to max_var + 1 do
766     write(file3,set_equ[j]^equ2[ws],':');
767   writeln(file3);
768 end;
769 m_val := set_equ[j]^equ3;
770 set_equ[j]^equ3 := 0.0;
771 end;
772 end;
773 procedure write_3;
774 var ws : integer;
775 begin
776   if set_equ[j]^equ3 <> 0 then
777     begin
778       write(file2,set_equ[j]^row_no,':',set_equ[j]^col_no,':1:');
779       write(file2,set_equ[j]^equ3,':');
780       for ws := 1 to max_var + 1 do
781         write(file2,set_equ[j]^equ2[ws],':');
782       writeln(file2);
783     end;
784 end;
785 procedure write_fir;
786 var ws : integer;
787 begin
788   if net_equ[j]^equ3 <> 0 then
789     begin
790       write(file3,net_equ[j]^row_no,':');
791       write(file3,net_equ[j]^col_no,':1:');
792       write(file3,net_equ[j]^equ3,':');
793       for ws := 1 to max_var + 1 do
794         write(file3,net_equ[j]^equ2[ws],':');
795       writeln(file3);
796     end;
797 end;
798 procedure write_4;
799 var ws : integer;
800 begin
801   if po2 <> 0 then
802     begin
803       write(file4,set_equ[j]^row_no,':');
804       write(file4,po2,':1:');
805       write(file4,net_equ[k]^equ3,':');
806       for ws := 1 to max_var + 1 do
807         if ws = i then write(file4,net_equ[k]^equ2[ws]*1,':')
808         else write(file4,net_equ[k]^equ2[ws],':');
809       writeln(file4);
810     end;
811   if po3 <> 0 then
812     begin
813       write(file4,set_equ[j]^row_no,':');
814       write(file4,po3,':1:');
815       write(file4,net_equ[k]^equ3 * po4,':');
816       for ws := 1 to max_var + 1 do
817         write(file4,net_equ[k]^equ2[ws],':');
818       writeln(file4);
819     end;
820 end;
821 procedure write_5;
822 var ws : integer;
823 begin
824   if po2 <> 0 then
825     begin
826       write(file4,set_equ[j]^row_no,':');
827       write(file4,po2,':1:');
828       write(file4,net_equ[k]^equ3 * m_val,':');
829       for ws := 1 to max_var + 1 do
830         if ws = i then write(file4,net_equ[k]^equ2[ws]*1,':')
831         else write(file4,net_equ[k]^equ2[ws],':');
832       writeln(file4);
833     end;
834   if po3 <> 0 then
835     begin
836       write(file4,set_equ[j]^row_no,':');
837       write(file4,po3,':1:');
838       write(file4,net_equ[k]^equ3 * po4 * m_val,':');
839       for ws := 1 to max_var + 1 do
840         write(file4,net_equ[k]^equ2[ws],':');
841       writeln(file4);
842     end;
843 end;
844 procedure control_by_current;
845 begin
846   if cur_sou[i] <> 0 then set_equ[j]^equ3 := set_equ[j]^equ3 / c
847   sou[i]
848   else
849     if copy(name[i-1],1,1) = 'C' then set_equ[j]^equ2[i-1] := 1
850     se set equ[j]^equ2[i-1] := -1;
851   write_sec;
852 end;
853 label no_chg,no_chg2,final;
854 begin
855   (generate space)
856   for ary_index := 1 to term_max do begin
857     new(f_equ[ary_index]); f_equ[ary_index]^equ2[1] := 0; end;
858   mess_t1 := ' ';
859   mess := ' ';

```

```

859 if length(mess_t1) <> 0 then
860   begin
861     ana := 'AC';
862     analysis;
863     max_var2 := max_var;
864   end;
865
866 (clear heapmemory)
867 release(heaporg);
868 make_variable; clear_var(3);
869 (Evaluate value)
870 (must change node_max to node_max2)
871 var_count := 1;
872 i := 1;
873 repeat
874   if copy(name[i],1,1) <> '*' then inc(var_count);
875   if (copy(name[i],2,1) = 'G') or (copy(name[i],1,1) = '
876   inc(i);
877 until i > max_var;
878 var_count := poin-1;
879 if var_count > component_max2 then
880   begin
881     writeln('Variable is too much = ',var_count - 1);
882     goto final;
883   end;
884
885 read_add; (give MATSUM and new AGAT)
886 for i := 1 to max_var2 do
887   begin
888     var_val2[i] := var_val[i];
889   end;
890 (rearrange AGVS)
891 read_fir('AGVS');
892 assign(file3,'AGVS');
893 rewrite(file3);
894 for j := 1 to fir_end do write_fir;
895 close(file3);
896
897 (rearrange AJ)
898 read_fir('AJ');
899 assign(file3,'AJ');
900 rewrite(file3);
901 for j := 1 to fir_end do write_fir;
902 close(file3);
903
904 (rearrange GVS)
905 read_fir('GVS');
906 assign(file3,'GVS');
907 rewrite(file3);
908 for j := 1 to fir_end do write_fir;
909 close(file3);
910
911 read_fir4('AGAT');
912 assign(file3,'AGAT');
913 rewrite(file3);
914 for j := 1 to fir_end do write_fir;
915
916 assign(file4,'JMI');
917 rewrite(file4);
918
919 prd_node;
920
921 (change VCVS OK)
922 pol := 0;
923 for i := 1 to max_var do
924   if copy(name[i],1,3) = '^VV' then inc(pol);
925 if pol <> 0 then
926   begin
927     read_sec4('AGVS');
928     for i := 1 to max_var do
929       if copy(name[i],1,3) = '^VV' then
930         begin
931           read_bran; (return pol branch number,po2,
932           po4 = direction of bran]-
933           for j := 1 to sec_end do
934             if set_equ[j]^equ2[i] <> 0 then write_sec
935           end;
936           assign(file2,'AGVS');
937           rewrite(file2);
938           for j := 1 to sec_end do write_3;
939           close(file2);
940         end;
941
942 (change VCVS OK)
943 pol := 0;
944 for i := 1 to max_var do
945   if copy(name[i],1,3) = '^JV' then inc(pol);
946 if pol <> 0 then
947   begin
948     read_sec4('AJ');
949     for i := 1 to max_var do
950       if copy(name[i],1,3) = '^JV' then
951         begin
952           read_bran; (return pol branch number,po2,p
953           po4 = direction of bran)
954           for j := 1 to sec_end do

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้


```

136 write(file2,net_equ[fir_st]^equ3 * set_eq
ec_st]^equ3,',');
137 for k := 1 to max_var + 1 do
138 write(file2,net_equ[fir_st]^equ2[k] +
t_equ[sec_st]^equ2[k],',');
139 writeln(file2);
140 end;
141 read_fir2('INV',i);
142 read_sec3('AGAT',j); {read the correct row}
143 for fir_st := 1 to fir_end do
144 for sec_st := 1 to sec_end do
145 if set_equ[sec_st]^col_no = i then
146 begin
147 write(file2,j,',',net_equ[fir_st]^col_no,
148 write(file2,net_equ[fir_st]^equ3 * set_eq
ec_st]^equ3 * -1,',');
149 for k := 1 to max_var + 1 do
150 write(file2,net_equ[fir_st]^equ2[k] +
u[sec_st]^equ2[k],',');
151 writeln(file2);
152 end;
153 close(file2);
154 {reform temp}
155 read_fir3('TEMP','AGAT');
156 {reform tempinv}
157 read_fir3('TEMPINV','INV');
158 no_chg2: end;
159 {divide on every row}
160 for i := 1 to node_max do
161 begin
162 read_fir2('AGAT',i);
163 if fir_end = 1 then
164 begin
165 assign(file2,'TEMPINV');
166 rewrite(file2);
167 read_sec2('INV',i);
168 for j := 1 to sec_end do
169 begin
170 write(file2,i,',',set_equ[j]^col_no,',':1:');
171 write(file2,set_equ[j]^equ3 / net_equ[i]^
3,',');
172 for k := 1 to max_var + 1 do
173 write(file2,set_equ[j]^equ2[k],',');
174 writeln(file2);
175 end;
176 close(file2);
177 assign(file1,'INV');
178 erase(file1);
179 assign(file2,'TEMPINV');
180 rename(file2,'INV');
181 {change source}
182 assign(file2,'TEMPINV');
183 rewrite(file2);
184 read_sec2('AGAT',i);
185 for j := 1 to sec_end do
186 begin
187 write(file2,i,',',set_equ[j]^col_no,',':1:');
188 write(file2,set_equ[j]^equ3 / net_equ[i]^
3,',');
189 for k := 1 to max_var + 1 do
190 write(file2,set_equ[j]^equ2[k],',');
191 writeln(file2);
192 end;
193 close(file2);
194 assign(file1,'AGAT');
195 erase(file1);
196 assign(file2,'TEMPINV');
197 rename(file2,'AGAT');
198 end;
199 end;

```

{AT this point, INV matrix keep the incorrect answer of the INVER

{to find the complete result of inverse matrix, INV must divide

```

3 find final result AGVS + JMI - AJ
4 for i := 1 to max_var2 do
5 var_val[i] := var_val2[i];
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99

```

```

1 read_fir4('AGVS');
2 read_fir5('JMI');
3 fir_end := fir_st - 1;
4 read_fir5('AJ');
5 for i := fir_end+1 to fir_st-1 do
6 net_equ[i]^equ3 := -1 * net_equ[i]^equ3;
7 fir_end := fir_st - 1;
8 {sorting}
9 for loop1 := 1 to fir_end-1 do
10 for loop3 := loop1+1 to fir_end do
11 begin
12 {check is it equal or not}
13 if (net_equ[loop1]^row_no = net_equ[loop3]^row_no) an
14 (net_equ[loop1]^col_no = net_equ[loop3]^col_no) th
15 begin
16 po4 := 0;
17 for po3 := 1 to max_var + 1 do
18 if net_equ[loop1]^equ2[po3] = net_equ[loop
qu2[po3] then inc(po4);

```

```

1226 if po4 = max_var + 1 then
1227 begin
1228 net_equ[loop1]^equ3 := net_equ[loop1]
+ net_equ[loop3]^equ3;
1229 net_equ[loop3]^equ3 := 0.0;
1230 end;
1231 end;
1232 end;
1233 {keep result into temp2}
1234 assign(file1,'TEMP2');
1235 rewrite(file1);
1236 for loop1 := 1 to fir_end do
1237 begin
1238 if net_equ[loop1]^equ3 <> 0 then
1239 begin
1240 write(file1,net_equ[loop1]^row_no,',');
1241 write(file1,net_equ[loop1]^col_no,',':1:');
1242 write(file1,net_equ[loop1]^equ3,',');
1243 for k := 1 to max_var + 1 do
1244 write(file1,net_equ[loop1]^equ2[k],',');
1245 writeln(file1);
1246 end;
1247 end;
1248 close(file1);
1249
1250 multiply('INV','TEMP2','RESULT');
1251 read_fir3('result','result');
1252 {the result matrix must be divide by agat row by row}
1253 result;
1254 {erase not necessary file}
1255 KILL('ADDMIT'); KILL('ADDMITR'); KILL('CURSOU'); KILL('VOLS
');
1256 KILL('AJ'); KILL('AGVS'); KILL('GVS'); KILL('MATSUM'); KILL
');
1257 KILL('JMI'); KILL('TEMP'); KILL('TEMP2'); KILL('INV'); KILL
bolic.$$$');
1258 final: end.

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

1 procedure clear_var(cr_e : integer);
2 procedure clear1;
3 var c_f : integer;
4 begin
5   for c_f := 1 to term_max2 do net_equ[c_f]^equ3 := 0;
6 end;
7 procedure clear2;
8 var c_f : integer;
9 begin
10  for c_f := 1 to term_max2 do set_equ[c_f]^equ3 := 0;
11 end;
12 begin
13   case cr_e of
14     1 : clear1;
15     2 : clear2;
16     3 : begin clear1; clear2; end;
17   end;
18 end;
19 procedure convert_data(c_d : integer);
20 var c_d1,c_d2 : integer;
21 begin
22   c_d2 := 0;
23   temp_mess := concat(' ',temp_mess);
24   repeat
25     delete(temp_mess,1,1);
26     val(temp_mess,mess_val,c_d1);
27     inc(c_d2);
28   until (c_d1 = 0) or (c_d2 >= c_d);
29 end;
30 procedure convert_data2;
31 begin
32   temp_mess := copy(mess,r_m,30);
33   temp_mess := concat(' ',temp_mess);
34   repeat
35     delete(temp_mess,1,1);
36   until (length(temp_mess) = 0) or (copy(temp_mess,1,1) <> ' ');
37   val(temp_mess,m_val,pol);
38 end;
39 procedure modify_var;
40 begin
41   (modify var_name)
42   po2 := 1;
43   for pol := 1 to max_var do
44     begin
45       if copy(name[pol],1,1) <> '*' then
46         begin
47           name[po2] := name[pol];
48           inc(po2);
49         end;
50     end;
51   assign(file1,'volsou2');
52   reset(file1);
53   repeat
54     readln(file1,c_mess);
55     if copy(c_mess,1,1) = 'J' then
56       begin
57         loop1 := pos(' ',c_mess);
58         for loop2 := 1 to i - 1 do
59           if copy(c_mess,1,loop1-1) = var_label[loop2]^
60             begin
61               delete(c_mess,1,loop1);
62               loop1 := pos(' ',c_mess);
63               var_label[loop2]^mess := concat(copy(
64                 c_mess,1,loop1-1));
65               (* delete(c_mess,1,loop1);
66               loop3 := 1;
67               repeat loop1 := pos(' ',c_mess); if lo
68                 if loop3 <> 1 then var_label[loop2
69                   inc(loop3);
70                   if copy(c_mess,1,1) <> 'R' then
71                     var_label[loop2]^mess := concat(v
72                       var_label[loop2]^mess := concat(v
73                         delete(c_mess,1,loop1);
74                         until length(c_mess) = 0;
75                         var_label[loop2]^mess := concat(var_l
76                           var_label[loop2]^mess := concat(var_label[loop2]
77                             end;
78                             until eof(file1);
79                             close(file1);
80 end;
81 procedure make_variable2;
82 var r_sl : integer;
83 begin
84   for r_sl := 1 to term_max do
85     begin
86       dispose(net_equ[r_sl]);
87       dispose(set_equ[r_sl]);
88     end;
89   (for r_sl := 1 to out_max do
90     begin
91       dispose(out_fir[r_sl]);
92       dispose(out_sec[r_sl]);

```

```

93   end;
94   for r_sl := 1 to 20 do
95     new(var_label[r_sl]);
96   end;
97   procedure read_symbol;
98   var file_name : text;
99   mess : string[10];
100  r_sl,r_s2 : integer;
101  begin
102    make_variable2;
103    assign(file_name,'var_name.lib');
104    reset(file_name);
105    clear_var(2);
106    r_sl := 1;
107    repeat
108      readln(file_name,mess);
109      r_s2 := pos(' ',mess);
110      var_label[r_sl]^mess := copy(mess,1,r_s2-1);
111      inc(r_sl);
112    until eof(file_name);
113    close(file_name);
114    i := r_sl;
115    modify_var;
116  end;
117  procedure execute2;
118  var temp_mess : string[30];
119  begin
120    (read symbolic.lib)
121    read_symbol;
122    assign(file2,'result');
123    rewrite(file2);
124    for loop3 := 1 to node_max do
125      begin
126        str(loop3,mess_t1);
127        writeln(file2);
128        mess_t1 := concat('equ',mess_t1);
129        assign(file1,mess_t1); reset(file1);
130        loop1 := 1;
131        v_st := var_count * 3;
132        write(file2,'{');
133        m_val := 0.0;
134        repeat
135          readln(file1,t_me1);
136          if t_me1 = '=' then
137            begin
138              writeln(file2,' ' * vnode',node2[loop3],
139                m_val := 0.0;
140                loop1 := 1;
141            end else
142            begin
143              if loop1 = 1 then write(file2,'{') else
144                write(file2,copy(t_me1,9 + v_st,30));
145              t_me2 := concat(' ',copy(t_me1,9+v_st,30));
146              repeat delete(t_me2,1,1) until copy(t_me
147                val(t_me2,m_val2,pol);
148                m_val := m_val + m_val2;
149                for loop2 := 1 to var_count - 1 do
150                  begin
151                    if copy(t_me1,loop2*3+6,3) <> '
152                      begin
153                        t_me2 := concat(' ',copy
154                          repeat delete(t_me2,1,1)
155                          write(file2,'(' ,var_label
156                        end;
157                        t_me2 := concat(' ',copy(t_me1,6+v_st,
158                          repeat delete(t_me2,1,1) until copy(t
159                          val(t_me2,po2,pol);
160                          if copy(t_me1,v_st + 6,3) <> ' 0' then
161                            begin
162                              write(file2,'(S^',po2,')');
163                              end;
164                              writeln(file2,')');
165                              inc(loop1);
166                            end;
167                          until eof(file1);
168                          writeln(file2);
169                          m_val := 0.0;
170                          writeln(file2,'-----');
171                          close(file1);
172                        end;
173                        close(file2);
174                      end;
175                    procedure rep_volsou;
176                    begin
177                      assign(file_name,'volsou2');
178                      reset(file_name);
179                      for i := term_max downto term_max - max_var - 2 do
180                        begin
181                          f_equ[i]^col_no := 0;
182                          for pol := 1 to term_max2 do f_equ[i]^equ2[pol] := (

```

เอกสารนี้จัดทำขึ้นเพื่อใช้ในการเรียนการสอนเท่านั้น ไม่ควรนำออกเผยแพร่

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้คัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

end;
for i := 1 to max_var do
  if copy(name[i],1,1) = 'J' then
    begin
      pol := pos(' ',name[i]);
      t_mel := copy(name[i],2,pol-2);
      val(t_mel,pol,pol);
      if pol > 1100 then
        begin
          f_equ[term_max - i + 1].col_no := 1;
          repeat
            readln(file_name,c_mess);
          until (copy(c_mess,1,pol) = copy(name[i],1,p
or (eof(file_name)));
          delete(c_mess,1,pol);
          pol := pos(' ',c_mess);
          delete(c_mess,1,pol);
          repeat
            pol := pos(' ',c_mess);
            t_mel := copy(c_mess,1,pol-1);
            write('*****',i:5);
            for po2 := 1 to max_var do
              begin
                if t_mel = copy(name[po2],1,pol-1) then
                  f_equ[term_max - i + 1].equ2[po2+1] := f_equ[term_max - i + 1].equ2
+ 1;
                write(f_equ[term_max - i + 1].equ2[po2+1]:3);
                end;
                writeln;
                delete(c_mess,1,pol);
                until length(c_mess) = 0;
              end;
            end;
          close(file_name);
          for i := 1 to max_var do
            writeln('hhh',f_equ[term_max - i + 1].col_no);
          end;
        procedure make_variable;
        var m_v : integer;
        begin
          for m_v := 1 to term_max2 do
            begin
              new(net_equ[m_v]);
              net_equ[m_v].equ3 := 0;
            end;
          for m_v := 1 to term_max2 do
            begin
              new(set_equ[m_v]);
              set_equ[m_v].equ3 := 0;
            end;
          end;
        end;
        procedure convert(cvl : integer);
        begin
          pol := pos(':',t_mel); mess_t2 := copy(t_mel,1,pol-1);
          delete(t_mel,1,pol); val(mess_t2,f_equ[cvl].row_no,po2);
          pol := pos(':',t_mel); mess_t2 := copy(t_mel,1,pol-1);
          delete(t_mel,1,pol); val(mess_t2,f_equ[cvl].col_no,po2);
          pol := pos(':',t_mel); delete(t_mel,1,pol);
          if copy(t_mel,1,1) = '-' then po2 := -1 else po2 := 1;
          delete(t_mel,1,2); f_equ[cvl].equ2[1] := po2;
          for pol := 2 to max_var+1 do
            begin po2 := pos(':',t_mel);
              if po2 = 0 then po2 := length(t_mel);
              mess_t2 := copy(t_mel,1,po2-1); delete(t_mel,1,po2);
              val(mess_t2,f_equ[cvl].equ2[pol],po2);
            end;
          f_equ[cvl].equ2[max_var + 2] := 0;
        end;
        procedure convert2(cvl : integer);
        begin
          pol := pos(':',t_mel); mess_t2 := copy(t_mel,1,pol-1);
          delete(t_mel,1,pol); val(mess_t2,net_equ[cvl].row_no,po2);
          pol := pos(':',t_mel); mess_t2 := copy(t_mel,1,pol-1);
          delete(t_mel,1,pol); val(mess_t2,net_equ[cvl].col_no,po2);
          pol := pos(':',t_mel); delete(t_mel,1,pol);
          pol := pos(':',t_mel); mess_t2 := copy(t_mel,1,pol-1);
          delete(t_mel,1,pol); val(mess_t2,net_equ[cvl].equ3,po2);
          for pol := 1 to max_var+1 do
            begin po2 := pos(':',t_mel);
              if po2 = 0 then po2 := length(t_mel);
              mess_t2 := copy(t_mel,1,po2-1); delete(t_mel,1,po2);
              val(mess_t2,net_equ[cvl].equ2[pol],po2);
            end;
          end;
        function power(pv1 : real;
          pv2 : integer) : real;
        begin
          if pv1 <> 0 then power := exp(pv2 * ln(pv1))
          else power := 1;
        end;
        procedure Error_mes(er_mes : integer);
        begin
          case er_mes of
            1 : writeln('File not found');
            240 : writeln('Write protect');
            241 : writeln('Filename full');
          end;
          writeln('Press any key to continue');
          repeat until keypressed;
        end;

```

```

283 procedure file_in(s_f : integer);
284 label file_inend;
285 begin
286   repeat
287     case s_f of
288       1 : begin clrscr; writeln('Read file Enter <filename> ');
289           readln(mess_t1);
290           if length(mess_t1) = 0 then goto file_inend;
291           end;
292       2 : mess_t1 := 'symbolic.$$$';
293       3 : mess_t1 := 'trans.lib';
294       4 : mess_t1 := 'mostrans.lib';
295       5 : mess_t1 := 'jfet.lib';
296     end;
297     assign(file_name,mess_t1);
298     {Si-} reset(file_name) {Si+};
299     rdl := ioread;
300     if rdl <> 0 then error_mes(rdl);
301     if (rdl <> 0) and (s_f > 2) then goto file_inend;
302     until rdl = 0;
303   file_inend: end;
304   procedure file_in2(s_f : integer);
305   label file_inend;
306   begin
307     repeat
308       case s_f of
309         1 : begin clrscr; writeln('Read file Enter <filename> ');
310             readln(mess_t1);
311             if length(mess_t1) = 0 then goto file_inend;
312             end;
313         2 : mess_t1 := 'symbolic.$$$';
314         3 : mess_t1 := 'trans.lib';
315         4 : mess_t1 := 'mostrans.lib';
316         5 : mess_t1 := 'jfet.lib';
317       end;
318       assign(file1,mess_t1);
319       {Si-} reset(file1) {Si+};
320       rdl := ioread;
321       if rdl <> 0 then error_mes(rdl);
322       if (rdl <> 0) and (s_f > 2) then goto file_inend;
323       until rdl = 0;
324     file_inend: end;
325   procedure file_out(s_f : integer);
326   var se_int : integer;
327   label file_outend;
328   begin
329     repeat
330       case s_f of
331         1 : begin clrscr; writeln('Save file Enter <filename>'); re
332             mess_t1);
333             if length(mess_t1) = 0 then goto file_outend; end;
334             2 : mess_t1 := 'symbolic.$$$';
335             3 : mess_t1 := 'symbolic.lib';
336             4 : mess_t1 := 'trans.lib';
337             5 : mess_t1 := 'mostrans.lib';
338             6 : mess_t1 := 'jfet.lib';
339           end;
340           assign(file_name,mess_t1);
341           {Si-} rewrite(file_name) {Si+};
342           se_int := ioread;
343           if se_int <> 0 then error_mes(se_int);
344           until se_int = 0;
345         file_outend: end;
346       procedure file_out2(s_f : integer);
347       var se_int : integer;
348       label file_outend;
349       begin
350         repeat
351           case s_f of
352             1 : begin clrscr; writeln('Save file Enter <filename>'); re
353                 mess_t1);
354                 if length(mess_t1) = 0 then goto file_outend; end;
355                 2 : mess_t1 := 'symbolic.$$$';
356                 3 : mess_t1 := 'symbolic.lib';
357                 4 : mess_t1 := 'trans.lib';
358                 5 : mess_t1 := 'mostrans.lib';
359                 6 : mess_t1 := 'jfet.lib';
360               end;
361               assign(file2,mess_t1);
362               {Si-} rewrite(file2) {Si+};
363               se_int := ioread;
364               if se_int <> 0 then error_mes(se_int);
365               until se_int = 0;
366             file_outend: end;
367             procedure write_error(we : byte);
368             procedure w_e;
369             begin
370               writeln('ERROR INLINE : ',cf3:5); writeln(mess(cf3));
371               writeln('Press anykey to continue ');
372               repeat until keypressed;
373             end;
374           begin
375             writeln; writeln('ERROR MESSAGE':25);
376             case we of
377               1 : w_e;
378               2 : begin write('COMPONENT '); w_e; end;
379               3 : begin write('NODE NUMBER '); w_e; end;
380               4 : begin write;
381                   writeln('FORMAT CHECK OK':30);

```

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้คัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

002      writeln('TOTAL BRANCH = ',poin-1);
003      writeln('TOTAL NODE = ',node_max);
004      write_error(0); end;
005      5 : begin writeln('COMPONENT DUPLICATED'); w_e; end;
006      6 : begin writeln('G NODE ERROR'); w_e; end;
007      7 : begin writeln('FLOAT VOLTAGE ERROR'); w_e; end;
008      end;
009  end;
010  procedure save_file(s_f : integer);
011  begin
012      file_out(s_f);
013      if length(mess_t1) <> 0 then
014          begin
015              ary_index := 1;
016              for ary_con := 1 to max_mess do writeln(file_name,mess[ary_con]
017              );
018          end;
019      end;
020  end;
021  procedure read_and_cv2;
022  begin
023      if not(eof(file1)) then begin
024          repeat
025              readln(file1,t_mel);
026              pol := pos(':',t_mel); mess_t2 := copy(t_mel,1,pol-1);
027              delete(t_mel,1,pol); val(mess_t2,net_equ[fir_st]^row_no,p
028              );
029              pol := pos(':',t_mel); mess_t2 := copy(t_mel,1,pol-1);
030              delete(t_mel,1,pol); val(mess_t2,net_equ[fir_st]^col_no,p
031              );
032              pol := pos(':',t_mel); delete(t_mel,1,pol);
033              if copy(t_mel,1,1) = '-' then po2 := -1 else po2 := 1;
034              delete(t_mel,1,2); net_equ[fir_st]^equ3 := po2;
035              po4 := 0;
036              for pol := 1 to max_var2+1 do
037                  begin po2 := pos(':',t_mel);
038                      if po2 = 0 then po2 := length(t_mel);
039                      mess_t2 := copy(t_mel,1,po2-1); delete(t_mel,1,p
040                      );
041                      val(mess_t2,po3,po2);
042                      if var_val[pol] <> 0 then
043                          begin
044                              net_equ[fir_st]^equ3 := net_equ[fir_st]^equ
045                              ;
046                              end
047                          else
048                              begin
049                                  inc(po4);
050                                  net_equ[fir_st]^equ2[po4] := po3;
051                                  end;
052                              end;
053                          inc(fir_st);
054                          until eof(file1);
055                      end;
056  end;
057  procedure read_and_cv3;
058  begin
059      if not(eof(file1)) then
060          begin
061              repeat
062                  readln(file1,t_mel);
063                  pol := pos(':',t_mel); mess_t2 := copy(t_mel,1,pol-1);
064                  delete(t_mel,1,pol); val(mess_t2,net_equ[fir_st]^row_no,p
065                  );
066                  pol := pos(':',t_mel); mess_t2 := copy(t_mel,1,pol-1);
067                  delete(t_mel,1,pol); val(mess_t2,net_equ[fir_st]^col_no,p
068                  );
069                  pol := pos(':',t_mel); delete(t_mel,1,pol);
070                  if copy(t_mel,1,1) = '-' then po2 := -1 else po2 := 1;
071                  delete(t_mel,1,2); net_equ[fir_st]^equ3 := po2;
072                  po4 := 0;
073                  for pol := 1 to max_var2+1 do
074                      begin po2 := pos(':',t_mel);
075                          if po2 = 0 then po2 := length(t_mel);
076                          mess_t2 := copy(t_mel,1,po2-1); delete(t_mel,1,p
077                          );
078                          val(mess_t2,po3,po2);
079                          if var_val[pol] <> 0 then
080                              begin
081                                  net_equ[fir_st]^equ3 := net_equ[fir_st]^equ
082                                  ;
083                                  end
084                              else
085                                  begin
086                                      inc(po4);
087                                      net_equ[fir_st]^equ2[po4] := po3;
088                                  end;
089                              end;
090                          inc(fir_st);
091                          until eof(file1);
092                      end;
093  end;
094  procedure read_fir(fir_name : string);
095  begin
096      assign(file1,fir_name);
097      reset(file1);
098      fir_st := 1;
099      read_and_cv2;
100      close(file1);
101      fir_end := fir_st - 1;

```

```

473  end;
474  procedure read_sec(sec_name : string);
475  begin
476      assign(file1,sec_name);
477      reset(file1);
478      sec_st := 1;
479      if not(eof(file1)) then begin
480          repeat
481              readln(file1,t_mel);
482              pol := pos(':',t_mel); mess_t2 := copy(t_mel,1,pol-1);
483              delete(t_mel,1,pol); val(mess_t2,set_equ[sec_st]^row
484              );
485              pol := pos(':',t_mel); mess_t2 := copy(t_mel,1,pol-1);
486              delete(t_mel,1,pol); val(mess_t2,set_equ[sec_st]^col
487              );
488              pol := pos(':',t_mel); delete(t_mel,1,pol);
489              if copy(t_mel,1,1) = '-' then po2 := -1 else po2 := 1;
490              delete(t_mel,1,2); set_equ[sec_st]^equ3 := po2;
491              po4 := 0;
492              for pol := 1 to max_var2+1 do
493                  begin po2 := pos(':',t_mel);
494                      if po2 = 0 then po2 := length(t_mel);
495                      mess_t2 := copy(t_mel,1,po2-1); delete(t_mel,1,p
496                      );
497                      val(mess_t2,po3,po2);
498                      if var_val[pol] <> 0 then
499                          begin
500                              set_equ[sec_st]^equ3 := set_equ[sec_st]^
501                              ;
502                              end
503                          else
504                              begin
505                                  inc(po4);
506                                  set_equ[sec_st]^equ2[po4] := po3;
507                                  end;
508                              end;
509                          inc(sec_st);
510                          until eof(file1);
511                      end;
512  end;
513  procedure replace;
514  begin
515      {replace value}
516      po2 := 1;
517      po3 := 0;
518      for i := 1 to max_var do
519          begin
520              pol := pos(':',name[i]);
521              if pol = 0 then pol := pos('-',name[i]);
522              mess := copy(name[i],pol + 2,30);
523              val(mess,m_val,po1);
524              var_val[i] := m_val;
525              if pol <> 0 then
526                  begin
527                      name[po2] := concat('^',copy(name[i],1,1),m
528                      );
529                      if (copy(name[po2],1,3) = '^VJ') or (copy(n
530                      );
531                      if po3 <> 0 then cur_sou[po2] := 0 else
532                      if copy(name[i-1],1,1) = 'C' then cur
533                      else cur_sou[po2] := var_val[po2-1];
534                      inc(po2);
535                      end;
536                      po3 := pol;
537                      var_val[max_var+1] := 0.0;
538                  end;
539  end;
540  procedure read_add;
541  begin
542      replace;
543      read_fir('AGAT');
544      assign(file1,'AGAT');
545      rewrite(file1);
546      for fir_st := 1 to fir_end do
547          begin
548              write(file1,net_equ[fir_st]^row_no,':');
549              write(file1,net_equ[fir_st]^col_no,':');
550              write(file1,net_equ[fir_st]^equ3,':');
551              for i := 1 to po4 do
552                  write(file1,net_equ[fir_st]^equ2[i],':');
553              writeln(file1);
554          end;
555      close(file1);
556      read_fir('Agvs');
557      read_sec('Aj');
558      max_var := po4 - 1;
559      {add keep result}
560      assign(file1,'matsum');
561      rewrite(file1);
562      for fir_st := 1 to fir_end do
563          for sec_st := 1 to sec_end do
564              begin
565                  {check equal}
566                  po2 := 0;
567                  if (net_equ[fir_st]^row_no = set_equ[sec_st]^r
568                  );
569              end;
570          end;
571      end;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

66      (net_equ[fir_st]^col_no = set_equ[sec_st]^col_
then
67      begin
68      for pol := 1 to max_var+1 do
69      begin
70      if net_equ[fir_st]^equ2[pol] = set_equ[sec_st]
71      then
72      begin inc(pol2); end;
73      if pol2 = (max_var+1) then
74      begin
75      net_equ[fir_st]^equ3 := net_equ[fir_st]^equ
set_equ[sec_st]^equ3;
76      set_equ[sec_st]^equ3 := 0;
77      end;
78      end;
79      if net_equ[fir_st]^equ3 <> 0 then
80      begin
81      write(file1,net_equ[fir_st]^row_no,',');
82      write(file1,net_equ[fir_st]^col_no,',1:');
83      write(file1,net_equ[fir_st]^equ3,',');
84      for i := 1 to max_var+1 do
85      write(file1,net_equ[fir_st]^equ2[i],',');
86      writeln(file1);
87      end;
88      end;
89      for sec_st := 1 to sec_end do
90      begin
91      if set_equ[fir_st]^equ3 <> 0 then
92      begin
93      write(file1,set_equ[fir_st]^row_no,',');
94      write(file1,set_equ[fir_st]^col_no,',1:');
95      write(file1,set_equ[fir_st]^equ3 * -1,',');
96      for i := 1 to max_var+1 do
97      write(file1,set_equ[fir_st]^equ2[i],',');
98      writeln(file1);
99      end;
100      end;
101      close(file1);
102      end;
103      procedure read_fir2(fir_name : string;
104      r_check : integer);
105      begin
106      assign(file1,fir_name);
107      reset(file1);
108      fir_st := 1;
109      repeat
110      readln(file1,t_mel);
111      pol := pos(':',t_mel); mess_t2 := copy(t_mel,1,pol-1);
112      delete(t_mel,1,pol); val(mess_t2,net_equ[fir_st]^row_no,p
113
114      pol := pos(':',t_mel); mess_t2 := copy(t_mel,1,pol-1);
115      delete(t_mel,1,pol); val(mess_t2,net_equ[fir_st]^col_no,p
116
117      pol := pos(':',t_mel); delete(t_mel,1,pol);
118      pol := pos(':',t_mel); mess_t2 := copy(t_mel,1,pol-1);
119      delete(t_mel,1,pol); val(mess_t2,net_equ[fir_st]^equ3,pol2
120
121      for pol := 1 to max_var+1 do
122      begin pol2 := pos(':',t_mel);
123      if pol2 = 0 then pol2 := length(t_mel);
124      mess_t2 := copy(t_mel,1,pol2-1); delete(t_mel,1,p
125
126      val(mess_t2,pol3,pol2);
127      net_equ[fir_st]^equ2[pol] := pol3;
128      end;
129      if r_check = net_equ[fir_st]^row_no then inc(fir_st);
130      until eof(file1);
131      close(file1);
132      fir_end := fir_st - 1;
133      end;
134      procedure read_sec2(sec_name : string;
135      r_check : integer);
136      begin
137      assign(file1,sec_name);
138      reset(file1);
139      sec_st := 1;
140      k := 0;
141      repeat
142      readln(file1,t_mel);
143      t_mel2 := t_mel;
144      pol := pos(':',t_mel); mess_t2 := copy(t_mel,1,pol-1);
145      delete(t_mel,1,pol); val(mess_t2,set_equ[sec_st]^row_no,p
146
147      pol := pos(':',t_mel); mess_t2 := copy(t_mel,1,pol-1);
148      delete(t_mel,1,pol); val(mess_t2,set_equ[sec_st]^col_no,p
149
150      pol := pos(':',t_mel); delete(t_mel,1,pol);
151      pol := pos(':',t_mel); mess_t2 := copy(t_mel,1,pol-1);
152      delete(t_mel,1,pol); val(mess_t2,set_equ[sec_st]^equ3,pol2
153
154      for pol := 1 to max_var+1 do
155      begin pol2 := pos(':',t_mel);
156      if pol2 = 0 then pol2 := length(t_mel);
157      mess_t2 := copy(t_mel,1,pol2-1); delete(t_mel,1,p
158
159      val(mess_t2,pol3,pol2);
160      set_equ[sec_st]^equ2[pol] := pol3;
161      end;
162      if r_check = set_equ[sec_st]^row_no then
163      begin if set_equ[sec_st]^col_no = i then inc(k);

```

```

656      inc(sec_st);
657      end else writeln(file2,t_mel2);
658      until eof(file1);
659      close(file1);
660      sec_end := sec_st - 1;
661      end;
662      procedure read_and_cv;
663      begin
664      repeat
665      readln(file1,t_mel);
666      pol := pos(':',t_mel); mess_t2 := copy(t_mel,1,pol-1);
667      delete(t_mel,1,pol); val(mess_t2,net_equ[fir_st]^row
2);
668      pol := pos(':',t_mel); mess_t2 := copy(t_mel,1,pol-1);
669      delete(t_mel,1,pol); val(mess_t2,net_equ[fir_st]^col
2);
670      pol := pos(':',t_mel); delete(t_mel,1,pol);
671      pol := pos(':',t_mel); mess_t2 := copy(t_mel,1,pol-1);
672      delete(t_mel,1,pol); val(mess_t2,net_equ[fir_st]^equ
673
674      for pol := 1 to max_var+1 do
675      begin pol2 := pos(':',t_mel);
676      if pol2 = 0 then pol2 := length(t_mel);
677      mess_t2 := copy(t_mel,1,pol2-1); delete(t_mel,1,p
678
679      val(mess_t2,pol3,pol2);
680      net_equ[fir_st]^equ2[pol] := pol3;
681      end;
682      inc(fir_st);
683      until eof(file1);
684      end;
685      procedure read_fir3(fir_name,sec_name : string);
686      begin
687      assign(file1,fir_name);
688      reset(file1);
689      fir_st := 1;
690      read_and_cv;
691      close(file1);
692      fir_end := fir_st - 1;
693      {sorting}
694      for loop1 := 1 to fir_end-1 do
695      for loop3 := loop1+1 to fir_end do
696      begin
697      {check is it equal or not}
698      if (net_equ[loop1]^row_no = net_equ[loop3]^row_no
699      (net_equ[loop1]^col_no = net_equ[loop3]^col_no
700
701      begin
702      pol4 := 0;
703      for pol3 := 1 to max_var + 1 do
704      if net_equ[loop1]^equ2[pol3] = net_equ
705      ]^equ2[pol3] then inc(pol4);
706      if pol4 = max_var + 1 then
707      begin
708      net_equ[loop1]^equ3 := net_equ[loop1]^
709      net_equ[loop3]^equ3;
710      net_equ[loop3]^equ3 := 0.0;
711      end;
712      end;
713      end;
714      assign(file1,sec_name);
715      rewrite(file1);
716      for loop1 := 1 to fir_end do
717      if net_equ[loop1]^equ3 <> 0 then
718      begin
719      write(file1,net_equ[loop1]^row_no,',');
720      write(file1,net_equ[loop1]^col_no,',1:');
721      write(file1,net_equ[loop1]^equ3,',');
722      for k := 1 to max_var + 1 do
723      write(file1,net_equ[loop1]^equ2[k],',');
724      writeln(file1);
725      end;
726      close(file1);
727      end;
728      procedure read_sec3(sec_name : string;
729      r_check : integer);
730      begin
731      assign(file1,sec_name);
732      reset(file1);
733      sec_st := 1;
734      k := 0;
735      repeat
736      readln(file1,t_mel);
737      t_mel2 := t_mel;
738      pol := pos(':',t_mel); mess_t2 := copy(t_mel,1,pol-1);
739      delete(t_mel,1,pol); val(mess_t2,set_equ[sec_st]^row
2);
740      pol := pos(':',t_mel); mess_t2 := copy(t_mel,1,pol-1);
741      delete(t_mel,1,pol); val(mess_t2,set_equ[sec_st]^col
2);
742      pol := pos(':',t_mel); delete(t_mel,1,pol);
743      pol := pos(':',t_mel); mess_t2 := copy(t_mel,1,pol-1);
744      delete(t_mel,1,pol); val(mess_t2,set_equ[sec_st]^equ3
2);
745      pol := pos(':',t_mel); delete(t_mel,1,pol);
746      pol := pos(':',t_mel); mess_t2 := copy(t_mel,1,pol-1);
747      delete(t_mel,1,pol); val(mess_t2,set_equ[sec_st]^equ3
2);
748      for pol := 1 to max_var+1 do
749      begin pol2 := pos(':',t_mel);
750      if pol2 = 0 then pol2 := length(t_mel);
751      mess_t2 := copy(t_mel,1,pol2-1); delete(t_mel,1,p
752
753      val(mess_t2,pol3,pol2);

```

```

set_equ[sec_st]^equ2[po1] := po3;
end;
if r_check = set_equ[sec_st]^row_no then inc(sec_st);
until eof(file1);
close(file1);
sec_end := sec_st - 1;
end;
procedure read_fir4(fir_name : string);
begin
assign(file1,fir_name);
reset(file1);
fir_st := 1;
repeat
readln(file1,t_mel);
po1 := pos(':',t_mel); mess_t2 := copy(t_mel,1,po1-1);
delete(t_mel,1,po1); val(mess_t2,net_equ[fir_st]^row_no,p
po1 := pos(':',t_mel); mess_t2 := copy(t_mel,1,po1-1);
delete(t_mel,1,po1); val(mess_t2,net_equ[fir_st]^col_no,p

po1 := pos(':',t_mel); delete(t_mel,1,po1);
po1 := pos(':',t_mel); mess_t2 := copy(t_mel,1,po1-1);
delete(t_mel,1,po1); val(mess_t2,net_equ[fir_st]^equ3,po2

for po1 := 1 to max_var+1 do
begin po2 := pos(':',t_mel);
if po2 = 0 then po2 := length(t_mel);
mess_t2 := copy(t_mel,1,po2-1); delete(t_mel,1,p
val(mess_t2,po3,po2);
net_equ[fir_st]^equ2[po1] := po3;
end;
inc(fir_st);
until eof(file1);
close(file1);
fir_end := fir_st - 1;
end;
procedure read_sec4(sec_name : string);
begin
assign(file1,sec_name);
reset(file1);
sec_st := 1;
k := 0;
repeat
readln(file1,t_mel);
t_mel2 := t_mel;
po1 := pos(':',t_mel); mess_t2 := copy(t_mel,1,po1-1);
delete(t_mel,1,po1); val(mess_t2,set_equ[sec_st]^row_no,p
po1 := pos(':',t_mel); mess_t2 := copy(t_mel,1,po1-1);
delete(t_mel,1,po1); val(mess_t2,set_equ[sec_st]^col_no,p
po1 := pos(':',t_mel); delete(t_mel,1,po1);
po1 := pos(':',t_mel); mess_t2 := copy(t_mel,1,po1-1);
delete(t_mel,1,po1); val(mess_t2,set_equ[sec_st]^equ3,po2

for po1 := 1 to max_var+1 do
begin po2 := pos(':',t_mel);
if po2 = 0 then po2 := length(t_mel);
mess_t2 := copy(t_mel,1,po2-1); delete(t_mel,1,p
val(mess_t2,po3,po2);
set_equ[sec_st]^equ2[po1] := po3;
end;
inc(sec_st);
until eof(file1);
close(file1);
sec_end := sec_st - 1;
end;
procedure read_fir5(fir_name : string);
begin
assign(file1,fir_name);
reset(file1);
repeat
readln(file1,t_mel);
po1 := pos(':',t_mel); mess_t2 := copy(t_mel,1,po1-1);
delete(t_mel,1,po1); val(mess_t2,net_equ[fir_st]^row_no,p
po1 := pos(':',t_mel); mess_t2 := copy(t_mel,1,po1-1);
delete(t_mel,1,po1); val(mess_t2,net_equ[fir_st]^col_no,p
po1 := pos(':',t_mel); delete(t_mel,1,po1);
po1 := pos(':',t_mel); mess_t2 := copy(t_mel,1,po1-1);
delete(t_mel,1,po1); val(mess_t2,net_equ[fir_st]^equ3,po2

for po1 := 1 to max_var+1 do
begin po2 := pos(':',t_mel);
if po2 = 0 then po2 := length(t_mel);
mess_t2 := copy(t_mel,1,po2-1); delete(t_mel,1,p
val(mess_t2,po3,po2);
net_equ[fir_st]^equ2[po1] := po3;
end;
inc(fir_st);
until eof(file1);
close(file1);
end;
procedure kill(k_name : string);
begin

```

```

833 assign(file1,k_name); erase(file1);
834 ($I+)
835 end;
836 procedure print_name(p_name : integer);
837 begin
838 po3 := p_name;
839 case po3 of
840 0 : write(file1,'');
841 1 : write(file1,'*',name[po2],',');
842 -1 : write(file1,'*(1/',name[po2],')');
843 else begin
844 if po3 > 1 then write(file1,'*',name[po2],'^',po3,');
845 if po3 < -1 then write(file1,'*(1/',name[po2],'^',ab
846 end;
847 end;
848 if (copy(name[po2],1,1) = 'L') and (po3 <> 0) then j := j
849 if (copy(name[po2],1,1) = 'C') and (po3 <> 0) then j := j
850 end;
851 procedure result;
852 begin
853 max_var := 0;
854 assign(file1,'symbolic.$$$');
855 reset(file1);
856 repeat
857 t_mel := copy(name[max_var+1],1,3);
858 readln(file1,name[max_var+1]);
859 po1 := pos(' ',name[max_var+1]);
860 if t_mel = 'VV' then name[max_var+1] := concat('Alp
y(name[max_var+1],2,po1-1))
861 else
862 if t_mel = 'JV' then name[max_var+1] := concat('g',
me[max_var+1],2,po1-1))
863 else
864 if t_mel = 'VJ' then name[max_var+1] := concat('Rm'
ame[max_var+1],2,po1-1))
865 else
866 if t_mel = 'JJ' then name[max_var+1] := concat('Bet
(name[max_var+1],2,po1-1))
867 else name[max_var+1] := copy(name[max_var+1],1,po1-1
868 inc(max_var);
869 until eof(file1);
870 close(file1);
871 read_fir4('result');
872 read_sec4('agat');
873 assign(file1,concat('\snap\result'));
874 rewrite(file1);
875 fir_st := 0;
876 name[max_var+1] := 'S';
877 for po1 := 1 to fir_end do
878 begin
879 if fir_st <> net_equ[po1]^row_no then
880 begin
881 fir_st := net_equ[po1]^row_no;
882 writeln(file1,'Voltage at node ',node2(fir
= ');
883 write(file1,'{');
884 end;
885 write(file1,'(',net_equ[po1]^equ3:15);
886 j := 0;
887 for po2 := 1 to max_var do print_name((net_equ[po1
2[po2]));
888 po2 := max_var + 1;
889 print_name(net_equ[po1]^equ2[max_var+1]+j);
890 if (po1 < fir_end) and (net_equ[po1+1]^row_no = f
then writeln(file1,') + ')
891 else
892 begin
893 writeln(file1,')});';
894 writeln(file1,')');
895 write(file1,')');
896 ana := '';
897 for po4 := 1 to sec_end do
898 if set_equ[po4]^row_no = fir_st then
899 begin
900 write(file1,ana,'(',set_equ[po
qu3:15);
901 j := 0;
902 for po2 := 1 to max_var do pri
me((set_equ[po4]^equ2[po2]));
903 po2 := max_var + 1;
904 print_name(set_equ[po4]^equ2
905 writeln(file1,') ');
906 ana := '+';
907 end;
908 writeln(file1,')});';
909 end;
910 end;
911 close(file1);
912 end;

```