สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

ตัวประมวลผลฟาสท์ฟูเรียร์ทรานส์ฟอร์ม

Fast Fourier Transform Processor

โดย

นายชัชวาล ธรรมราช

นายชูชาติ มูลคำ

นายศุภศิลป์ คำสร้อย

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

เลขหมู่.....
เลขทะเบียน 46462
วัน, เดือน, ปี - 2 เม.ย. 2546

.b.....
.i.....

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตัวประมวลผลฟาสท์ฟูเรียร์ทรานส์ฟอร์ม

Fast Fourier Transform Processor

โดย

นายชัชวาล ธรรมราช

นายชูชาติ มูลคำ

นายศุภศิลป์ คำสร้อย

อาจารย์ที่ปรึกษา

อาจารย์ อัครพล ตรีรัตน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปริญญานิพนธ์ปีการศึกษา 2544

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง **ตัวประมวลผลฟาสท์ฟูเรียร์ทรานส์ฟอร์ม**

Fast Fourier Transform Processor

ผู้จัดทำ

1. **นาย ชัชวาล ธรรมราช 42015010**
2. **นาย ชูชาติ มูลคำ 42015014**
3. **นาย ศุภศิลป์ คำสร้อย 42015036**

..... อาจารย์ที่ปรึกษา

(อาจารย์ อัครพล ตรีรัตน์)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตัวประมวลผลฟาสฟูเรียร์ทรานส์ฟอร์ม

Fast Fourier Transform Processor

โดย นายชัชวาล ธรรมราช 42015010
นายชูชาติ มูลคำ 42015014
นายศุภศิลป์ คำสร้อย 42015036

อาจารย์ที่ปรึกษา อาจารย์ อัครพล ตรีรัตน์

บทคัดย่อ

โครงงานนี้เป็นการศึกษาตัวประมวลผลฟาสฟูเรียร์ทรานส์ฟอร์ม (FFT Processor) ซึ่งเป็นการคำนวณ DFT ประเภทหนึ่ง ในการคำนวณ DFT ที่มีลำดับความยาวเท่ากับ N จะมีอัตราของการคำนวณเท่ากับ N^2 ด้วยหลักการของ FFT อัตราการคำนวณจะมีค่าเพียง $N\log_2 N$ เท่านั้น

โครงงานนี้จะทำการออกแบบ ตัวประมวลผลฟาสฟูเรียร์ทรานส์ฟอร์ม (FFT Processor) โดยใช้ FPGA (Field Programmable Gate Array) โดยที่ส่วนของวงจรคูณจำนวนเชิงซ้อน (Complex Multiplier) จะใช้หลักการของเลขคณิตกระจาย (Distribute Arithmetic) ในการออกแบบ ตัวประมวลผลฟาสฟูเรียร์ทรานส์ฟอร์ม (FFT Processor) นี้จะเป็นไปตามหลักการของ FFT radix –2 decimation - in – frequency และการคำนวณข้อมูลพื้นฐานภายใน ROM โดยตัวประมวลผลฟาสฟูเรียร์ทรานส์ฟอร์มนี้จะประกอบไปด้วย 4 ส่วนใหญ่ๆคือ ส่วนของอินพุท ส่วนประมวลผล ส่วนเอาท์พุท และ ส่วนควบคุม

Abstract

This project describes a FFT processor which is a class of DFT computation. The straightforward computation of the DFT of a sequence length N requires a computation rate that grow as N^2 but, with FFT algorithm, their computation rate grows only as $N\log_2 N$.

In this paper, we propose the 8-point FFT processor designed with Field Programmable Gate Array (FPGA), which employs Distributed Arithmetic (DA) in complex multiplier circuit. A processor performs radix-2 decimation-in-frequency basic computations on data in ROM. This FFT processor comprises 4 main parts that are input part, processor part, output part and control part.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

	หน้า
บทที่ 1 บทนำ	1
1.1 ความเป็นมาของหัวข้อปริญญานิพนธ์	1
1.2 วัตถุประสงค์ของปริญญานิพนธ์	2
1.3 ขอบเขตของปริญญานิพนธ์	2
1.4 เนื้อหาของปริญญานิพนธ์	
บทที่ 2 ทฤษฎีและหลักการ	3
2.1 ทฤษฎีและหลักการแปลงฟาสท์ฟูเรียร์	3
2.1.1 หลักการเบื้องต้นของ การแปลงฟาสท์ฟูเรียร์	3
2.1.2 ขั้นตอนวิธีการแปลงฟาสท์ฟูเรียร์ชนิดลดทอนทางเวลา	7
2.1.3 การลดทอนทางเวลาแบบอื่นๆ	12
2.1.4 การแปลงฟาสท์ฟูเรียร์ ชนิดลดทอนทางความถี่	13
2.1.5 การแปลงฟาสท์ฟูเรียร์สำหรับ N ใดๆ	15
2.2 หลักการเบื้องต้นของโครงสร้างเลขคณิตกระจาย	19
2.2.1 ระบบตัวเลข	19
2.2.2 ทฤษฎีเลขคณิตกระจาย	21
2.3 ทฤษฎีของวงจรแปลงอะนาล็อกเป็นดิจิตอล	27
2.3.1 วงจรเปรียบเทียบขนาน หรือ แบบแฟลช	27
2.3.2 วงจรเอทูดีที่ใช้การอินทีเกรต	28
2.3.3 วงจรเปลี่ยนสัญญาณเอทูดีที่ใช้วงจรนับและวงจรดีทูเอประกอบกัน	29
2.3.4 วงจรเปลี่ยนสัญญาณเอทูดีแบบใช้การประมาณค่า	30
2.3.5 การสุ่มและการคงค่า	30
2.4 พอร์ทเครื่องพิมพ์แบบขนาน	31
2.4.1 สายสัญญาณและหัวต่อ	31
2.4.2 สัญญาณจากพอร์ทเครื่องพิมพ์	33
2.4.3 การนำสัญญาณจากพอร์ทเครื่องพิมพ์มาใช้งาน	34
2.4.4 การใช้พอร์ทเครื่องพิมพ์เป็นพอร์ทสองทิศทาง	37
2.4.5 การใช้พอร์ทเครื่องพิมพ์เป็นพอร์ท อินพุท/เอาท์พุท มากกว่าหนึ่งพอร์ท	38
2.5 ทฤษฎี ภาษาวีเอชดีแอล	41
2.5.1 ประวัติความเป็นมาของภาษา วีเอชดีแอล	41
2.5.2 ส่วนต่างๆ ของแบบของภาษา วีเอชดีแอล	50

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.6 ประวัติความเป็นมาของภาษาซี 61
2.6.1 ประวัติความเป็นมาของภาษาซี 61
2.6.2 การประมวลของภาษาซี 61
2.6.3 ส่วนประกอบของภาษาซี 62
2.6.4 การเขียนโฟลว์ชาร์ต 64
2.7 เอ๋ฟพีจีเอ (Field Programmable Gate Array) 65
2.7.1 ลอจิกอีลีเมนท์ (Logic Element) 65
2.7.2 ลอจิกอาเรย์บล็อก (Logic Array Block) 68
2.7.3 แอมเบดเดดอาเรย์บล็อก (Embedded Array Block) 69
2.7.4 อินพุท เอาท์อีลีเมนท์ (Input Out Element) 70
บทที่ 3 การคำนวณและการสร้าง 71
3.1 ส่วนอินพุท 73
3.1.1 วงจรแปลงสัญญาณอะนาล็อกเป็นสัญญาณดิจิตอล 73
3.1.2 วงจรดีมัลติเพล็กซ์ 73
3.1.3 วงจรบัฟเฟอร์ 74
3.2 ส่วนตัวประมวลผล 75
3.2.1 แนวคิดในการออกแบบ 75
3.2.2 การออกแบบวงจรบวก 79
3.2.3 การออกแบบวงจรลบ 80
3.2.4 วงจรกลับค่าตัวเลข 81
3.2.5 ตัวคูณจำนวนเชิงซ้อน 81
3.3 ส่วนของเอาท์พุท 87
3.3.1 วงจรบัฟเฟอร์ 87
3.3.2 วงจรมัลติเพล็กซ์ 87
3.3.3 ส่วนของการแสดงผล 88
3.4 ส่วนกำเนิดสัญญาณควบคุม 91
บทที่ 4 การทดลองและผลการทดลอง 94
4.1 ส่วนอินพุท 94
4.1.1 วงจรดีมัลติเพล็กซ์ 94
4.1.2 วงจรบัฟเฟอร์ 95
4.2 ส่วนประมวลผล 96
4.2.1 วงจรบวก 96
4.2.2 วงจรลบ 96
4.2.3 วงจรทูคอมพลีเมนท์ 97

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2.4 วงจรคูณจำนวนเชิงซ้อน 98
4.3 ส่วนกำเนิดสัญญาณควบคุม 100
4.4 ส่วนของภาคเอาท์พุท 101
4.4.1 วงจรบัฟเฟอร์ 101
4.4.2 วงจรมัลติเพล็กซ์ 102
บทที่ 5 บทวิจารณ์และบทสรุป 105
หนังสืออ้างอิง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูปภาพ

หน้า

รูปที่ 2.1 กราฟการไหลแสดงถึงวิธีการคำนวณตามสมการ (2.1.9) 6

รูปที่ 2.2 หน่วยผีเสื้อของการคำนวณตามขั้นตอนวิธีลดทอนทางเวลา 8

รูปที่ 2.3 (a) และ (b) แสดงขั้นตอนวิธีแบบ DIT สำหรับ DFT แบบ 8 จุด 10

รูปที่ 2.4 (a) กราฟการไหลสัญญาณแสดงการคำนวณตามรูป 2.3 11
(b) แสดงการสลับตำแหน่งของลำดับ x(n) ด้วยการผันกลับบิต

รูปที่ 2.5 ภาพรวมแสดงขั้นตอนวิธีการคำนวณ DFT ขนาด N จุด แบบลดทอนทางเวลา 12

รูปที่ 2.6 แสดงลำดับขั้นตอนวิธีของ FFT ชนิดการลดทอนทางความถี่ 14

รูปที่ 2.7 แสดงการคำนวณของหน่วยผีเสื้อ ของขั้นตอนวิธีชนิดลดทอนทางความถี่ 15

รูปที่ 2.8 ภาพรวมแสดงขั้นตอนวิธีการคำนวณ DFT ในกรณี N มีค่าใดๆ 18

รูปที่ 2.9 แสดงการจัดรูปแบบจำนวนโดยตรงที่ประกอบด้วยบิทจำนวนเต็มและบิทเศษส่วน 19

รูปที่ 2.10 แสดงการจัดรูปแบบจำนวนโดยตรงที่มีแต่บิทเศษส่วน 20

รูปที่ 2.11 แสดงการจัดรูปแบบจำนวนอิงดรรชนี 21

รูปที่ 2.12 แสดงการคูณแบบเลขส่วนเติมเต็มสองโดยใช้บูทอัลกอริธึม 25

รูปที่ 2.13 แสดงการต่อวงจรเปรียบเทียบขนาน 27

รูปที่ 2.14 แสดงการลักษณะของหัวต่อพอร์ทขนาน 31

รูปที่ 2.15 แสดงสัญญาณเอาท์พุทแบบ 8 บิต ใช้จาก Port Base Address 34

รูปที่ 2.16 แสดงสัญญาณเอาท์พุทแบบ 4 บิต ใช้จาก Port Base +1 Address 35

รูปที่ 2.17 แสดงสัญญาณเอาท์พุทแบบ 4 บิต ใช้จาก Port Base +2 Address 35

รูปที่ 2.18 แสดงรูปการต่อใช้งานของ D0-D3 36

รูปที่ 2.19 แสดงใช้งานของมัลติเพล็กซ์ 36

รูปที่ 2.20 แสดงการใช้พอร์ทเครื่องพิมพ์เป็นพอร์ทแบบ 2 ทิศทาง 37

รูปที่ 2.21 แสดงการใช้งานของพอร์ทขนาน 8 บิต โดยใช้ ไอซี เบอร์ 74LS138 38

รูปที่ 2.22 แสดงการใช้งานของพอร์ทขนาน 16 บิต โดยใช้ไอซี เบอร์ 74LS374 39

รูปที่ 2.23 แสดงการอ่านค่า 8 บิต โดยใช้ไอซี เบอร์ 74LS257 / 74LS157 40

รูปที่ 2.24 VHDL Statement สำหรับ Multiply Accumulate Algorithm 42

รูปที่ 2.25 โครงสร้างของ Multiply-Accumulate Unit 42

รูปที่ 2.26 ขั้นตอนของ Top-Down Design 43

รูปที่ 2.27 Dataflow Description ของ Multiply Accumulate Function 44

รูปที่ 2.28 Gate-level Representation ของ Multiply-Accumulate Function 45

รูปที่ 2.29 Operator ที่กำหนดไว้ใน ภาษา VHDL 49

รูปที่ 2.30 โครงสร้างอย่างง่ายๆของ Entity Design Unit 51

รูปที่ 2.31 รูปแบบของ 2:1 multiplexer , (a) VHDL entity design , (b) มุมมองของ Interface 51

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 2.32 รูปแบบ 2:1 multiplexer ที่ประกอบด้วยข้อมูลเกี่ยวกับเวลา
(a) VHDL entity design , (b) มุมมองของ interface 52
รูปที่ 2.33 Entity design unit ที่ไม่มีการกำหนดช่องติดต่อกับภายนอก 52
รูปที่ 2.34 แสดงให้เห็นโครงสร้างอย่างง่ายๆของ architecture design unit 53
รูปที่ 2.35 Architecture design unit ของ 2:1 mux ตาม Boolean expression 53
รูปที่ 2.36 (a) Architecture description ของ 2:1 multiplexer (structural description)
(b) : Architecture description ของ 2: 1 mux (structural description) 54
รูปที่ 2.37 Architecture description ของ 2:1 mux (behavioral description) 55
รูปที่ 2.38 โครงสร้างของ package declaration 56
รูปที่ 2.39 ตัวอย่างของ Package declaration 57
รูปที่ 2.40 โครงสร้างของ Package body 57
รูปที่ 2.41 ตัวอย่างการเขียน Package 58
รูปที่ 2.42 โครงสร้างของ Configuration 58
รูปที่ 2.43 VHDL model ของ AND-gate2 input 59
รูปที่ 2.44 Configuration ของรูปแบบ (Model) and2 59
รูปที่ 2.45 (a) และ (b) ตัวอย่างของ Configuration 60
รูปที่ 2.46 โฟลว์ชาร์ตการประมวลผลของภาษาซี 62
รูปที่ 2.47 แสดงสัญลักษณ์พื้นฐานการเขียนโฟลว์ชาร์ต 64
รูปที่ 2.48 แสดงโครงสร้างของเอฟพีจีเอ ตระกูล FLEX 10 k 65
รูปที่ 2.49 แสดงโครงสร้างภายในของ LE 66
รูปที่ 2.50 การใช้งาน LUT เป็นโครงข่ายของลอจิก 66
รูปที่ 2.51 แสดงโครงข่ายของการเชื่อมต่อ 67
รูปที่ 2.52 แสดงโครงสร้างภายในของ LAB 68
รูปที่ 2.53 แสดงโครงสร้างภายในของ EAB 69
รูปที่ 2.54 แสดงโครงสร้างภายในของ IOE 70
รูปที่ 3.1 บล็อกไดอะแกรมของตัวประมวลผลฟาสท์ฟูเรียร์ทรานส์ฟอร์ม 72
รูปที่ 3.2 วงจรแปลงสัญญาณอะนาล็อกเป็นสัญญาณดิจิตอล 73
รูปที่ 3.3 บล็อกไดอะแกรมของวงจร ดีมัลติเพล็กซ์ 73
รูปที่ 3.4 บล็อกไดอะแกรมของวงจรบัฟเฟอร์ 74
รูปที่ 3.5 แผนภูมิผีเสื้อ เมื่อทำการแทนค่าTwiddle factor ลงไปในการแปลงฟาสท์ฟูเรียร์ 8 จุด 78
รูปที่ 3.7 บล็อกไดอะแกรมของวงจรบวก 79
รูปที่ 3.8 บล็อกไดอะแกรมของวงจรลบ 80
รูปที่ 3.9 บล็อกไดอะแกรมของวงจรคูณจำนวนเฉพาะตามหลักการเลขคณิตกระจาย 81
รูปที่ 3.10 บล็อกไดอะแกรมของวงจรข้อมูลเข้าแบบขนานข้อมูลออกแบบอนุกรม 82

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 3.11 บล็อกไดอะแกรมของวงจรบวกหรือวงจรลบ 84
รูปที่ 3.12 บล็อกไดอะแกรมของ แอ็คคูมูเลเตอร์ 84
รูปที่ 3.13 บล็อกไดอะแกรมของบัพเฟอร์ 85
รูปที่ 3.14 บล็อกไดอะแกรมของสัญญาณควบคุม 85
รูปที่ 3.15 แสดงสัญญาณที่ใช้ออกแบบส่วนควบคุม 86
รูปที่ 3.16 แสดงสเตทไดอะแกรมในการกำเนิดสัญญาณต่างๆ 86
รูปที่ 3.17 บล็อกไดอะแกรมของวงจรบัฟเฟอร์ 87
รูปที่ 3.18 บล็อกไดอะแกรมของวงจรมัลติเพล็กซ์ 87
รูปที่ 3.19 โฟลว์ชาร์ตการทำงานของส่วนการแสดงผล 89
รูปที่ 3.20 โฟลว์ชาร์ตการทำงานของส่วนพล็อตกราฟ 90
รูปที่ 3.21 โฟลว์ชาร์ตการทำงานของส่วนการแปลงเลขฐานสิบเป็นเลขฐานสอง 91
รูปที่ 3.22 บล็อกไดอะแกรมของส่วนควบคุม 91
รูปที่ 3.23 แสดง Timing ของส่วนควบคุม 92
รูปที่ 3.24 สเตทไดอะแกรมของ ฟาสท์ฟูเรียร์ทรานส์ฟอร์ม 93
รูปที่ 4.1 แสดงสัญลักษณ์ของวงจรดีมัลติเพล็กซ์ ที่ได้จากการ คอมไพล์ 95
รูปที่ 4.2 แสดงการจำลองการทำงานของวงจรดีมัลติเพล็กซ์ 95
รูปที่ 4.3 แสดงสัญลักษณ์ของวงจรบัฟเฟอร์ที่ได้จากการ คอมไพล์ 96
รูปที่ 4.4 แสดงการจำลองการทำงานของวงจรบัฟเฟอร์ 96
รูปที่ 4.5 แสดงสัญลักษณ์ของวงจรบวก ที่ได้จากการ คอมไพล์ 97
รูปที่ 4.6 แสดงการจำลองการทำงานของวงจรบวก 97
รูปที่ 4.7 แสดงสัญลักษณ์ของวงจรลบที่ได้จากการ คอมไพล์ 97
รูปที่ 4.8 แสดงการจำลองการทำงานของวงจรลบ 98
รูปที่ 4.9 แสดงสัญลักษณ์ของวงจรทูคอมพลีเมนท์ ที่ได้จากการ คอมไพล์ 98
รูปที่ 4.10 แสดงการจำลองการทำงานของวงจรทูคอมพลีเมนท์ 98
รูปที่ 4.11 แสดงสัญลักษณ์ของวงจรคูณจำนวนเชิงซ้อน ที่ได้จากการ คอมไพล์ 99
รูปที่ 4.12 แสดงการจำลองการทำงานของวงจรคูณจำนวนเชิงซ้อน 99
รูปที่ 4.13 แสดงสัญลักษณ์ของส่วนตัวประมวลผลฟาสท์ฟูเรียร์ 100
รูปที่ 4.14 แสดงการจำลองการทำงานของส่วนตัวประมวลผลฟาสท์ฟูเรียร์ 100
รูปที่ 4.15 แสดงการคำนวณการแปลงฟาสท์ฟูเรียร์โดยใช้โปรแกรม MATLAB 101
รูปที่ 4.16 แสดงสัญลักษณ์ของภาคกำเนิดสัญญาณควบคุม 101
รูปที่ 4.17 แสดงการจำลองการทำงานวงจรมัลติเพล็กซ์ 102
รูปที่ 4.18 แสดงสัญลักษณ์ของวงจรบัฟเฟอร์ ที่ได้จากการ คอมไพล์ 102
รูปที่ 4.19 แสดงสัญลักษณ์ของวงจรมัลติเพล็กซ์ ที่ได้จากการคอมไพล์ 103
รูปที่ 4.20 แสดงการจำลองการทำงานของวงจรมัลติเพล็กซ์ 103

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 4.21 แสดงการจำลองการทำงานของตัวประมวลผลฟาสท์ฟูเรียร์ทั้งหมด 104
รูปที่ 4.22 แสดงรูปของสัญญาณอินพุทเมื่อป้อนไฟตรง 1V 104
รูปที่ 4.23 แสดงขนาดของสแปคตรัม เมื่อป้อนไฟตรง 1 V 105
รูปที่ 4.24 แสดงเฟสของสเปคตรัม เมื่อป้อนไฟตรง 1V 105
รูปที่ 4.25 แสดงรูปของสัญญาณอินพุทเมื่อป้อนสัญญาณไซน์ 1 kHz 106
รูปที่ 4.26 แสดงขนาดของสแปคตรัม เมื่อป้อนสัญญาณไซน์ 1 kHz 106
รูปที่ 4.27 แสดงเฟสของสเปคตรัม เมื่อป้อนสัญญาณไซน์ 1 kHz 107
รูปที่ 4.28 แสดงรูปของสัญญาณอินพุทเมื่อป้อนสัญญาณไซน์ 2 kHz 107
รูปที่ 4.29 แสดงขนาดของสแปคตรัม เมื่อป้อนสัญญาณไซน์ 2 kHz 108
รูปที่ 4.30 แสดงเฟสของสเปคตรัม เมือ ป้อนสัญญาณไซน์ 2 kHz 108
รูปที่ 4.31 แสดงรูปของสัญญาณอินพุทเมื่อป้อนสัญญาณไซน์ 3kHz 109
รูปที่ 4.32 แสดงขนาดของสแปคตรัม เมื่อป้อนสัญญาณไซน์ 3kHz 109
รูปที่ 4.33 แสดงเฟสของสเปคตรัม เมื่อป้อนสัญญาณไซน์ 3kHz 110
รูปที่ 4.34 แสดงรูปของสัญญาณอินพุทเมื่อป้อนสัญญาณไซน์ 4 kHz 110
รูปที่ 4.35 แสดงขนาดของสแปคตรัม เมื่อป้อนสัญญาณไซน์ 4 kHz 111
รูปที่ 4.36 แสดงเฟสของสเปคตรัม เมื่อป้อนสัญญาณไซน์ 4 kHz 111
รูปที่ 4.37 แสดงการคำนวณ FFT โดยใช้โปรแกรม MATLAB เมื่อป้อนค่าเท่ากับ ค่า Sampling อินพุท ไซน์ความถี่ 1 kHz 112
รูปที่ 4.38 แสดงการคำนวณ ขนาดและเฟสโดยใช้โปรแกรม MATLAB เมื่อป้อนค่าเท่ากับ ค่าอินพุท ไซน์ความถี่ 1 kHz 112
รูปที่ 4.39 กราฟแสดงขนาดของสเปคตรัมของสัญญาณไซน์ 1 kHz ของการแปลง FFT 8 จุดโดย ใช้โปรแกรม MATLAB 113
รูปที่ 4.40 กราฟแสดงเฟสของสเปคตรัมของสัญญาณไซน์ 1 kHz ของการแปลง FFT 8 จุดโดย ใช้โปรแกรม MATLAB 113
รูปที่ 4.41 แสดงรูปของบอร์ด FPGA ที่ใช้ในการทดลอง 114
รูปที่ 4.42 แสดงรูปของบอร์ด FPGA ที่ใช้ในการทดลอง เมื่อต่อกับ ADC 0820 114

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2.4 วงจรคูณจำนวนเชิงซ้อน 99
4.3 ส่วนกำเนิดสัญญาณควบคุม 101
4.4 ส่วนของภาคเอาท์พุท 102
4.4.1 วงจรบัฟเฟอร์ 102
4.4.2 วงจรมัลติเพล็กซ์ 103
4.5 ตัวประมวลผลฟาสท์ฟูเรียร์ทรานส์ฟอร์มทั้งหมด 104
4.6 การแสดงผลผ่านจอภาพคอมพิวเตอร์ 104
4.7 รูปของบอร์ด FPGA 114
บทที่ 5 บทวิจารณ์และบทสรุป 115
หนังสืออ้างอิง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง

หน้า

ตารางที่ 2.1 แสดงขั้นตอนการคูณเลขส่วนเติมเต็มสอง 24
ตารางที่ 2.2 ค่าผลคูณย่อยที่เก็บไว้ในตารางเปิดดูที่กำหนดโดยข้อมูลอินพุท $X_{ij}, i = 0,1,...,N$ 26
ตารางที่ 2.3 แสดงการเชื่อมโยงระหว่างหัวต่อ ดีบี 25 32
ตารางที่ 2.4 แสดงค่าแอดเดรสของการอินพอร์ทและเอาท์พุท 33
ตารางที่ 2.5 แสดงรายละเอียดของข้อมูลแต่ละบิท BASE ADDRESS 33
ตารางที่ 2.6 แสดงรายละเอียดของข้อมูลแต่ละบิท BASE+1 ADDRESS 33
ตารางที่ 2.7 แสดงรายละเอียดของข้อมูลแต่ละบิท BASE +2 ADDRESS 34
ตารางที่ 2.8 แสดงตัวอย่างการเขียนโปรแกรม 63
ตารางที่ 3.1 แสดงสัญญาณควบคุมการทำงาน 74
ตารางที่ 3.2 แสดงการทำงานของวงจรบัฟเฟอร์ 75
ตารางที่ 3.3 ตารางความจริงของการออกแบบวงจรบวก 79
ตารางที่ 3.4 ตารางจำนวนเชิงซ้อนเมื่อป้อนค่าอินพุทส่วนจริงและส่วนจินตภาพ 81
ตารางที่ 3.5 ตำแหน่งและค่าที่เก็บในรอมแต่ละตัว 83
ตารางที่ 3.6 ตารางความจริงของวงจรมัลติเพล็กซ์ 88
ตารางที่ 3.7 ตารางแสดงค่าประจำแต่ละสเตท 93

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
บทนำ

1.1 ความเป็นมาของหัวข้อปริญญานิพนธ์

อุปกรณ์ทางคอมพิวเตอร์ในปัจจุบันมีราคาถูกและมีประสิทธิภาพสูง อีกทั้งความแม่นยำและความแน่นอนจากการคำนวณโดยใช้คอมพิวเตอร์มีมาก จึงทำให้การวิจัยและพัฒนาของการประมวลสัญญาณเชิงเลข เพื่อนำไปประยุกต์ใช้กับงานในสาขาวิชาต่างๆ โดยเฉพาะทางด้านอิเล็กทรอนิกส์มีอย่างกว้างขวาง ตัวอย่างเช่น การประมวลสัญญาณเสียงเชิงเลข (Digital speech processing) การประมวลสัญญาณชีวภาพ (Digital processing of biomedical signal) เป็นต้น การประมวลสัญญาณดังตัวอย่างที่กล่าวมานี้มีการคำนวณผลการประสาน เป็นสำคัญ โดยทั่วๆไปแล้ว การแปลงฟูเรียร์เต็มหน่วย (DFT) นั้นสามารถนำมาใช้ในการคำนวณการประสานได้ แต่การคำนวณฟูเรียร์เต็มหน่วย (DFT) เองนั้นเมื่อลำดับข้อมูลมีมากก็ใช้เวลาในการคำนวณของคอมพิวเตอร์มาก ตัวอย่างเช่น การคำนวณฟูเรียร์เต็มหน่วย (DFT) สำหรับลำดับสัญญาณเข้ายาว N ลำดับหรือจุดนั้น คอมพิวเตอร์ต้องทำการคำนวณ จำนวนเชิงซ้อน (Complex number) ถึง $N \times N$ ครั้ง และบวกจำนวนเชิงซ้อนอีก $N(N-1)$ ครั้ง ซึ่งคอมพิวเตอร์ทั่วไปแล้วไม่มีคำสั่งภาษาเครื่องที่ใช้ในการคูณตัวเลข จึงต้องเขียนเป็นโปรแกรมย่อยหรือ เพิ่มเติมวงจรคูณโดยเฉพาะเข้าไป ส่วนการบวกตัวเลขของคอมพิวเตอร์นั้นทำได้ง่ายและเร็ว จึงกล่าวได้ว่าขบวนการคูณตัวเลขนั้นใช้เวลาคำนวณมากกว่าการบวกตัวเลขมาก จากผลอันนี้ทำให้เห็นชัดว่า ความเร็วในการคำนวณฟูเรียร์เต็มหน่วย (DFT) จึงขึ้นอยู่กับความเร็วและจำนวนครั้งในการคูณตัวเลขเป็นสำคัญ

ในการคำนวณฟูเรียร์เต็มหน่วย (DFT) ให้เร็วนั้นเราเรียกว่า การแปลงฟาสท์ฟูเรียร์ (FFT) จะทำให้การคำนวณ ฟูเรียร์เต็มหน่วย (DFT) ใช้การคูณจำนวนเชิงซ้อนเพียง $N \log N$ ครั้งเท่านั้น หรือจำนวนครั้งในการคูณตัวเลขลดลงไปถึง $N/(\log_2 N)$ เท่า ผลดีอีกประการหนึ่งก็คือทำให้การสร้างวงจรเฉพาะเพื่อการคำนวณฟูเรียร์เต็มหน่วย (DFT) ทำได้ง่าย และคำนวณได้เร็วขึ้น เป็นผลให้การประมวลสัญญาณเชิงเลขสามารถนำไปประยุกต์ใช้กับระบบเวลาจริงได้ โดยทั่วไปแล้วเราอาจจะแบ่งออกได้เป็นชนิดใหญ่ ได้ 2 ชนิดด้วยกันคือ ชนิดลดทอนทางเวลา (Decimation in time หรือ DIT) กับ ชนิดลดทอนทางความถี่ (Decimation in Frequency หรือ DIF) ทั้งสองชนิดนี้โดยหลักการแล้วมีความคล้ายคลึงกัน

1.2 วัตถุประสงค์ของปริญญานิพนธ์

1.2.1 เพื่อศึกษาหลักการ การแปลงฟาสท์ฟูเรียร์ (FFT)

1.2.2 เพื่อศึกษาโครงสร้างเลขคณิตกระจาย (Distributed Arithmetic)

1.2.3 เพื่อนำโครงสร้างเลขคณิตกระจาย (Distributed Arithmetic) มาประยุกต์ใช้ในการคูณ

1.2.4 เพื่อศึกษา ภาษา VHDL เพื่อใช้ในการ ออกแบบ ตัวประมวลผลฟาสท์ฟูเรียร์

1.2.5 เพื่อศึกษา การใช้งาน การทำงานของ เอฟพีจีเอ (Field Programmable Gate Array)

1.2.6 สร้างตัวประมวลผลฟาสท์ฟูเรียร์บน เอฟพีจีเอ (Field Programmable Gate Array)

1.2.7 เพื่อศึกษาการเขียนโปรแกรมการส่งผ่านข้อมูลผ่านทางพอร์ทขนาน (ภาษา ซี)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.3 ขอบเขตของปริญญานิพนธ์

ปริญญานิพนธ์ฉบับนี้เป็นการออกแบบ และสร้างตัวประมวลผลฟาสท์ฟูเรียร์ โดยออกแบบบนพื้นฐานของ โปรแกรมลอจิกชิป (Programmable Logic Chip) หรือ เอฟพีจีเอ (Field Programmable Gate Array) ซึ่งใช้ ข้อมูล 8 บิท โดยการแปลงฟาสท์ฟูเรียร์ การแปลงแบบฐานสอง (radix 2) ความยาว 8 จุด ใช้การแปลงฟาสท์ฟูเรียร์ แบบ ชนิดลดทอนทางความถี่ (Decimation in Frequency หรือ DIF) การสร้างจะใช้การเขียนโปรแกรมด้วยภาษา VHDL แล้วทำการโปรแกรมลงบนชิป แล้วทำการแสดงผลทางจอภาพของคอมพิวเตอร์ โดยส่งข้อมูลผ่านทาง พอร์ทขนาน เปรียบเทียบกับผลการคำนวณ ฟาสท์ฟูเรียร์ 8 จุด โดยใช้ ฟังก์ชัน ฟาสท์ฟูเรียร์ ในโปรแกรมแมทแล็บ (Matlab)

1.4 เนื้อหาของปริญญานิพนธ์

บทที่ 2 กล่าวถึงทฤษฎี ที่นำมาใช้ในการออกแบบและสร้างตัวประมวลผลฟาสท์ฟูเรียร์ (FFT) ซึ่งประกอบด้วยทฤษฎี พื้นฐานของการแปลงฟาสท์ฟูเรียร์, ทฤษฎีพื้นฐานเลขคณิตกระจาย (Distribute Arithmetic หรือ DA), ทฤษฎีการแปลงสัญญาณอะนาล็อกเป็นสัญญาณดิจิตอล (A/D Converter), ทฤษฎีการใช้งานของพอร์ทขนาน (Parallel Port), ทฤษฎีการใช้งานของ เอฟพีจีเอ (Field Programmable Gate Array), ทฤษฎีการ โปรแกรมภาษา VHDL และ หลักการเขียนภาษา ซี

บทที่ 3 กล่าวถึงแนวความคิดในการออกแบบ โดยใช้หลักการของฟาสท์ฟูเรียร์ และ โฟลว์ชาร์ต (Flowchart) ของโปรแกรม และ บล็อกไดอะแกรม (Block Diagram) โครงสร้างของแต่ละส่วน

บทที่ 4 การทดลองและผลการทดลอง

บทที่ 5 บทวิจารณ์ และสรุปผล

หนังสืออ้างอิง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2

ทฤษฎีและหลักการ

2.1 ทฤษฎีและหลักการของฟาสท์ฟูเรียร์ทรานส์ฟอร์ม (Fast Fourier Transform)

2.1.1 หลักการเบื้องต้นของ ฟาสท์ฟูเรียร์ทรานส์ฟอร์ม

เพื่อความสะดวกในการอธิบายจะขอนำคำนิยามของการแปลงฟูเรียร์แบบ DFT มากล่าวไว้อีกครั้งหนึ่ง การแปลงแบบ DFT สำหรับ ลำดับ x(m) ที่ยาว N จุด สามารถนิยามได้ดังนี้คือ

$$X(k) = \sum_{m=0}^{N-1} x(m).W^{mk} \qquad (2.1.1)$$

โดยที่ดรรชนี k,m = 0,1,…..,N-1 และจำนวนเชิงซ้อน W = exp(-j2π/N) ในทางวิชาอิเล็กทรอนิกส์ ลำดับ x(m) มักจะเกี่ยวข้องกับสัญญาณในโดเมนเวลา ส่วน X(k) เกี่ยวข้องกับสัญญาณในโดเมนความถี่ หรือ เรียกว่า สเปกตรัมของสัญญาณ สำหรับการเขียนสมการ (2.1.1) นั้นความจริงแล้ว การคำนวณหา DFT จะต้องมีพจน์ 1/N ประกอบอยู่ด้วยเสมอ แต่ที่ละไว้ก็เพื่อความสะดวกในการอธิบาย ทั้งนี้เนื่องจากจุดที่สนใจก็คือต้องการหาวิธีการคำนวณ สมการ (2.1.1) ให้เร็วเท่านั้น

สมการ (2.1.1) สามารถเขียนให้อยู่ในรูปของสมการเมตริกซ์ได้คือ

$$\{X\} = \{A\} \cdot \{x\} \qquad (2.1.2a)$$

โดยที่ { X } และ { x } เป็น เวคเตอร์แนวตั้ง (column vector) ที่ประกอบด้วยลำดับ X(k) และ x(m) ตามลำดับจำนวน N ลำดับ และ { A } เป็น เมตริกซ์จตุรัส (square matrix) ขนาด NxN ที่มีสมาชิกเป็นจำนวนเชิงซ้อน W^{mk} ตัวอย่างเช่น พิจารณาสำหรับกรณี N=4 ระบบสมการ (2.1.2) สามารถเขียนแยกออกได้เป็น

$$\begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 \\ W^0 & W^2 & W^4 & W^6 \\ W^0 & W^3 & W^6 & W^9 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} \qquad (2.1.2b)$$

การคำนวณจากสมการ (2.1.2b) นี้โดยตรงนั้น สังเกตเห็นได้ว่า ต้องใช้การคูณจำนวนเชิงซ้อน 4x4 หรือ 16 ครั้ง และ ต้องทำการบวกจำนวนเชิงซ้อน อีก N(N-1) = 4(4-1) = 12 ครั้ง

ที่นี้มาดูหลักการสำคัญประการหนึ่งของการแปลงฟูเรียร์แบบ FFT ที่ลดจำนวนครั้งในการคูณจำนวนเชิงซ้อนได้ก็โดยอาศัย คุณสมบัติของ ภาวะเป็นคาบ (periodicity) ของจำนวนเชิงซ้อน W คือ

$$W^{mk} = W^{[mk \bmod(N)]} \qquad (2.1.3)$$

ซึ่ง [mk mod (N)] หมายถึง ส่วนที่เหลือหลังจากการหารพจน์ mk ด้วย N โดยอาศัยคุณสมบัติภาวะเป็นคาบนี้ทำให้สมการ (2.1.2) อาจเขียนได้เป็น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$\begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W^1 & W^2 & W^3 \\ 1 & W^2 & W^4 & W^6 \\ 1 & W^3 & W^6 & W^9 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} \qquad (2.1.4)$$

จากคุณสมบัติ ภาวะเป็นคาบ นี้เองทำให้เราสามารถแยกตัวประกอบของเมตริกซ์ { A } ออกเป็นเมตริกซ์ย่อยหลายเมตริกซ์คูณกัน และสมาชิกภายในเมตริกซ์ย่อยให้มีค่าเป็นศูนย์มากที่สุด วิธีการแยกตัวประกอบนี้จะไม่กระทำโดยตรงจาก { A } แต่จะมีการสลับตำแหน่งหรือจัดกลุ่มของเมตริกซ์ ด้วยวิธีการของการผันกลับบิต (bit reversed) และ เมตริกซ์หลังจัดการสลับแถวแล้วจะนำมาแยกตัวประกอบอีกครั้งหนึ่ง

ในที่นี้จะขออธิบายวิธีการแยกตัวประกอบด้วยการเขียนแทน ดรรชนี k และ m ของสมการ (2.1.1) ด้วย ตัวเลขฐานสอง (binary number) สำหรับกรณี N = 4 ซึ่งดรรชนี k และ m จะมีค่าได้เพียง 0, 1 , 2 และ 3 เท่านั้น เพราะฉะนั้นสามารถแทนตัวเลข 4 ฐานสิบ ได้ด้วย ตัวเลขฐานสอง สองหลัก คือ

$$k = (k_1, k_0) \quad , \quad m = (m_1, m_0) \qquad (2.1.5)$$

โดยที่ k_1, k_0, m_1 และ m_0 เป็นเลขโดดที่มีค่าได้แค่ 0 หรือ 1 เท่านั้น ดรรชนีในสมการ (2.1.5) นี้สามารถเขียนได้ใหม่เป็น

$$k = 2k_1 + k_0 \quad , \quad m = 2m_1 + m_0 \qquad (2.1.6)$$

เมื่อนำค่าดรรชนี k และ m นี้ไปแทนในสมการ (2.1.1) ทำให้ได้

$$X(k_1, k_0) = \sum_{m_0=0}^{1} \sum_{m_1=0}^{1} x(m_1, m_0) W^{(2m_1+m_0)\cdot(2k_1+k_0)} \qquad (2.1.7)$$

จากคุณสมบัติภาวะเป็นคาบของ W ทำให้ได้ความสัมพันธ์

$$\begin{aligned} W^{(2m_1+m_0)\cdot(2k_1+k_0)} &= W^{(2k_1+k_0)2m_1} \cdot W^{(2k_1+k_0)m_0} \\ &= W^{4m_1k_1} \cdot W^{2m_1k_0} \cdot W^{(2k_1+k_0)m_0} \\ &= W^{2m_1k_0} \cdot W^{(2k_1+k_0)m_0} \end{aligned}$$

โดยที่ $W^{4m1k1} = 1$ เพราะฉะนั้นเขียนสมการ (2.1.7) ได้ใหม่เป็น

$$X(k_1, k_0) = \sum_{m_0=0}^{1} \left\{ \sum_{m_1=0}^{1} x(m_1, m_0) W^{(2m_1+m_0)} \right\} W^{(2k_1+k_0)m_0}$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$= \sum_{m_0=0}^{1} \{x_1(k_0, m_0)\}\ W^{(2k_1+k_0)m_0} \qquad (2.1.8a)$$

โดยการสมมติให้ตัวแปร $x_1(k_0, m_0)$ เป็นการคำนวณระหว่างกลาง ผลจาก (2.1.8a) อาจเขียนเป็นสมการเมตริกซ์ได้คือ

$$\begin{matrix}(k_1, k_0)\\ \end{matrix} \qquad \begin{matrix}\\ (m_1, m_0)\end{matrix}$$

$$\begin{bmatrix} x(0,0) \\ x(1,0) \\ x(0,1) \\ x(1,1) \end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \begin{bmatrix} x(0,0) \\ x(0,1) \\ x(1,0) \\ x(1,1) \end{bmatrix} \qquad (2.1.8\ b)$$

โดยที่ผลการคำนวณระหว่างกลางและผลลัพธ์สามารถหาได้ตามลำดับ

$$\begin{matrix}(k_0, m_0)\\ \end{matrix} \qquad \begin{matrix}\\ (m_1, m_0)\end{matrix}$$

$$\begin{bmatrix} x_1(0,0) \\ x_1(1,0) \\ x_1(0,1) \\ x_1(1,1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \begin{bmatrix} x(0,0) \\ x(0,1) \\ x(1,0) \\ x(1,1) \end{bmatrix} \qquad (2.1.9\ a)$$

และ ค่า DFT ของลำดับสัญญาณเป็น

$$\begin{matrix}(k_1, k_0)\\ \end{matrix} \qquad \begin{matrix}\\ (k_0, m_0)\end{matrix}$$

$$\begin{bmatrix} X(0,0) \\ X(1,0) \\ X(0,1) \\ X(1,1) \end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \begin{bmatrix} x_1(0,0) \\ x_1(0,1) \\ x_1(1,0) \\ x_1(1,1) \end{bmatrix} \qquad (2.1.9\ b)$$

ผลที่ได้จากสมการ (2.1.8 b) ซึ่งเป็นสมการแยกตัวประกอบของ { A } นี้จะเห็นว่าสมาชิกตามแนวนอนของเมตริกซ์ที่ได้จากการแยกตัวประกอบนั้น มีเพียง 2 ตัวเท่านั้นที่มีค่าไม่เป็นศูนย์ และในสองตัวนั้นมีสมาชิกตัวหนึ่งมีค่าเป็นหนึ่งเสมอ ส่วนอีกตัวนั้นก็จะเป็นจำนวนเชิงซ้อน แต่ถ้าพิจารณาการคูณเมตริกซ์ย่อย แต่ละเมตริกซ์ของ (2.1.9) จะเห็นว่าต้องการการคูณจำนวนเชิงซ้อนเพียง N = 4 ครั้ง โดยที่ N อยู่ในรูปของสองยกกำลังใดๆ ($N = 2^r$) และ ถ้าลองเขียนสมการ (2.1.7) สำหรับกรณีนี้ดูจะเห็นว่าสามารถเขียนแยกเป็นสมการย่อยได้ $\log_2 N = \log_2 4 = 2$ สมการคือ (2.1.9 a) และ (2.1.9 b) เพราะฉะนั้นด้วยวิธีการนี้จำนวนครั้งของการคูณจำนวนเชิงซ้อนจึงมีเท่ากับ $N \log_2 N = 8$ ครั้ง และบวกจำนวนเชิงซ้อน 8 ครั้ง ซึ่งจำนวนนี้น้อยกว่ากรณีคำนวณโดยตรงซึ่งใช้การคูณจำนวนเชิงซ้อน 16 ครั้งและบวกจำนวนเชิงซ้อน 12 ครั้ง มาถึงตอนนี้เห็นได้ว่า ในกรณีทั่วไปถ้าหากเรามีกรรมวิธีในการแยกการคำนวณ DFT ให้เป็นเมตริกซ์ย่อยดังเช่นสมการ (2.1.8b) ได้ก็จะทำให้จำนวนครั้งในการคูณจำนวนเชิงซ้อนน้อยลง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

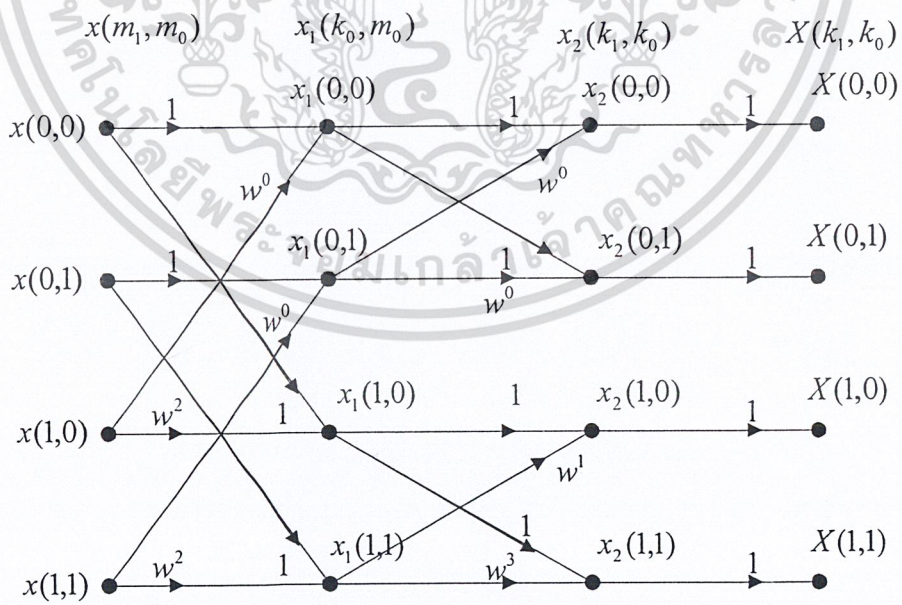
ความเป็นจริงแล้วจำนวนครั้งในการคูณจำนวนเชิงซ้อนอาจลดทอนลงไปได้อีก มาดูการคำนวณสัญญาณระหว่างกลาง $x_1(0,0)$ และ $x_1(0,1)$ ซึ่ง

$$x_1(0,0) = x(0,0) + W^0 \cdot x(1,0) = x(0,0) + W^0 \cdot x(1,0)$$

และ $$x_1(1,0) = x(0,0) + W^2 \cdot x(1,0) = x(0,0) - W^0 \cdot x(1,0) \qquad (2.1.10)$$

ผลจากคุณสมบัติของจำนวนเชิงซ้อน $W^2 = -W^0$ ทำให้การคำนวณ $x_1(0,0)$ และ $x_1(0,1)$ ต้องการการคูณจำนวนเชิงซ้อนเพียงครั้งเดียวเท่านั้น ซึ่งทำได้โดยการคำนวณพจน์ $W^0 . x(1,0)$ ก่อนแล้วนำไปบวกและลบกับพจน์ $x(0,0)$ เพื่อให้ได้ลำดับ $x_1(0,0)$ และ $x_1(1,0)$ ตามลำดับ อย่างไรก็ตามสำหรับกรณีทั่วๆ ไปอาจกล่าวได้ว่า จำนวนครั้งในการคำนวณ DFT ขนาด N จุดโดยใช้ FFT ใช้การคูณจำนวนเชิงซ้อนเพียง $N \log_2 N$ ครั้ง

ลำดับการคำนวณของ FFT ตามสมการ (2.1.8) เป็นผลงานที่เสนอโดย คูลีย์ และ ทูคีร์ ต่อมาก็ได้มีผู้เสนอวิธีการคำนวณ FFT แบบอื่นขึ้นมา ผลงานเหล่านี้หลักการใหญ่แล้วเหมือนกันต่างเพียงวิธีการในรายละเอียดย่อยเท่านั้น ในรูปที่ 2.1 เป็น กราฟการไหลสัญญาณ (Signal flow graph) ที่เขียนแสดงวิธีการคำนวณของ FFT ตามสมการ (2.1.9) โดยที่หัวลูกศรชี้ทิศทางการคำนวณ ส่วนตัวอักษรกำกับเป็นตัวคูณค่าของสัญญาณที่ต้นทางของลูกศรนั้น และที่โหนดเป็นการรวมหรือบวกกันของสัญญาณ ส่วน $x_1(k_0, m_0)$ แทนลำดับการคำนวณระหว่างกลาง และ $X(k_1, k_0)$ เป็นค่า DFT ของลำดับสัญญาณ



รูปที่ 2.1 กราฟการไหลแสดงถึงวิธีการคำนวณตามสมการ (2.1.9)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.1.2 ขั้นตอนวิธีการลดทอนทางเวลา (Decimation - in - time หรือ DIT)

วิธีการที่เสนอโดย คูลีย์ และ ทูคีย์ ซึ่งได้กล่าวมาแล้วนั้น ความเป็นจริงแล้ว เป็นการจัดแบ่งกลุ่มลำดับสัญญาณในโดเมนเวลา x(m) ที่มีขนาด N จุด ออกเป็นสองลำดับสัญญาณที่มีความยาว N/2 จุดเท่ากัน ทั้งสองลำดับนี้ให้ชื่อเรียกว่า ลำดับสัญญาณคู่ และ ลำดับสัญญาณคี่ โดยที่ลำดับสัญญาณคู่เกิดจากการเอาลำดับในตำแหน่งเป็นเลขคู่มาเรียงกัน ที่เหลือก็เป็นลำดับสัญญาณคี่ ดังนั้นถ้าเรานิยามให้ $x_E(m)$ เป็นลำดับคู่ และ ลำดับคี่เป็น $x_0(m)$ ตามลำดับ เพราะฉะนั้น

$$\begin{aligned} x_E(m) &= x(2m) \qquad ;m = 0,1...,(N/2)-1 \\ x_0(m) &= x(2m+1) \qquad ;m = 0,1...,(N/2)-1 \end{aligned} \tag{2.1.11}$$

ด้วยการจัดแบ่งเช่นนี้ ถ้าให้ W_N แทนค่า W ของลำดับยาว N จุด ทำให้การคำนวณการแปลง DFT ของลำดับสัญญาณ x(m) ที่ยาว N จุดเขียนได้เป็น

$$X(k) = \sum_{m=0}^{N-1} x_E(m)(W_N)^{km} + \sum_{m=0}^{N-1} x_0(m)(W_N)^{km}$$

m = เลขคู่ m = เลขคี่

$$= \sum_{m=0}^{N-1} x(2m)(W_N)^{2km} + \sum_{m=0}^{(N/2)-1} x(2m+1)(W_N)^{(2m+1)k}$$

โดยที่เขียนให้พจน์ $(W_N)^2$ เป็น

$$(W_N)^2 = \{\exp(j2\pi/N)\}^2 = \exp(j2\pi/N/2) = W_{N/2}$$

ซึ่ง $W_{N/2}$ หรือค่า W ของลำดับยาว N/2 จุด เพราะฉะนั้น (2.1.12) จัดพจน์ใหม่ได้คือ

$$X(k) = \sum_{m=0}^{(N/2)-1} x_E(m)(W_{N/2})^{km} + (W_N)^k \sum_{m=0}^{(N/2)-1} x_0(m)(W_{N/2})^{km}$$

$$X(k) = X_1(k) + (W_N)^k X_2(k) \tag{2.1.13}$$

โดยที่ $X_1(k)$ และ $X_2(k)$ แทนผลการแปลง DFT ขนาด N/2 จุดของลำดับ $x_E(m)$ และ $x_0(m)$ ตามลำดับ สมการ (2.1.13) แสดงให้เห็นว่าการคำนวณ DFT ขนาด N จุด นั้นสามารถแบ่งการคำนวณย่อยออกเป็นการคำนวณ DFT ขนาด N/2 จุดสองอันดับได้ และข้อสำคัญก็คือ การคูณจำนวนเชิงซ้อนจะลดลงเหลือ $2(N/2)^2 = N^2/2$ ครั้ง ซึ่งจะเห็นว่าลดเวลาการคำนวณลงไปได้ถึง 50 เปอร์เซ็นต์ โดยอาศัยหลักการเดียวกันถ้าเราแบ่งทอนลำดับ $x_E(m)$ และ $x_0(m)$ ออกเป็นลำดับคู่และลำดับคี่ลงไปอีกตามลำดับ จนในที่สุดเหลือเป็นลำดับขนาด 2 จุด หรือกล่าวได้ว่า การคำนวณการแปลง DFT ขนาด N จุด ทำได้โดยการคำนวณการแปลง DFT ขนาด 2 จุด จำนวน N/2 ภาคด้วยกัน ข้อสังเกตที่สำคัญก็คือการซอยเพื่อแบ่งลำดับ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$x(n)$ ออกเป็นทีละครึ่งจนเหลือการคำนวณ DFT ขนาด 2 จุด นี้ สำหรับสัญญาณขนาด N ลำดับ จะทำการแบ่งออกได้ $\log_2 N$ ครั้ง (ดูรูปที่ 2.3 ประกอบ)

ปัญหาต่อมาก็คือเมื่อแบ่งย่อยลงไปแล้วจะเอาการแปลง DFT ขนาด 2 จุด จำนวน N/2 ภาคนี้มาประกอบกัน หรือรวมตัวกันเพื่อให้เป็นการคำนวณ DFT ขนาด N จุดได้อย่างไร เพราะถ้าหากนำมาประกอบกันอย่างไม่มีหลักเกณฑ์ ผลของ DFT ที่ได้ก็มีค่าผิดพลาดไป ตัวอย่าง เช่น การนำเอา DFT ขนาด N/2 จุดมาต่อกันโดยตรงตามสมการ (2.1.13) ย่อมมีข้อผิดพลาด เพราะว่าทั้ง $X_1(k)$ และ $X_2(k)$ เป็น DFT ขนาด N/2 จุด ที่นิยามเฉพาะสำหรับช่วง $0 \le k \le N/2$ เท่านั้น การนำมาประกอบกันสำหรับลำดับสัญญาณ N ลำดับอย่างถูกต้อง เราต้องทำการนิยามค่าของสมการ (2.1.13) สำหรับค่า $k > N/2$ ด้วยซึ่งทำได้โดยเขียน

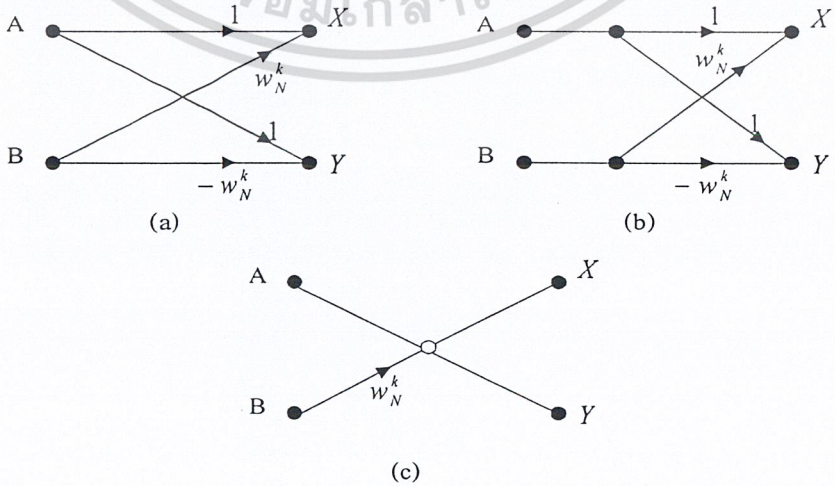
$$\begin{aligned} X(k) &= X_1(k) + (W_N)^k X_2(k) && ;0 \le k \le (N/2)-1 \\ &= X_1(k-N/2) + (W_N)^k X_2(k-N/2) && ;N/2 \le k \le N-1 \end{aligned} \qquad (2.1.14)$$

พจน์ $(W_N)^k$ ในสมการ (2.1.14) นี้มีชื่อเรียกว่า ตัวประกอบหมุน (twiddle factors หรือ phase หรือ rotation factor) พจน์นี้เป็นส่วนสำคัญร่วมกับ DFT ขนาด 2 จุด หรือ DFT ขนาด N/2 จุด ที่จะนำมาประกอบกันเข้าเป็น DFT ขนาด N จุด ได้เหมือนเดิม

ในสมการ (2.1.14) นี้ ถ้าหากเรานำเอาความสัมพันธ์ $(W_N)^{k+N/2} = -(W_N)^k$ แทนลงไปแล้วสมการนี้สามารถเขียนใหม่เป็น

$$\begin{aligned} X(k) &= X_1(k) + (W_N)^k X_2(k) && ;0 \le k \le (N/2)-1 \qquad (2.1.15\text{ a}) \\ &= X_1(k-N/2) - (W_N)^{k-N/2} X_2(k-N/2) && ;N/2 \le k \le N/2-1 \qquad (2.1.15\text{ b}) \end{aligned}$$

สมการที่ได้นี้เป็นหลักการสำคัญสำหรับ FFT แบบนี้ ซึ่งได้กล่าวถึงบ้างแล้วดังเช่นสมการ (2.1.10)



รูปที่ 2.2 หน่วยผีเสื้อของการคำนวณตามขั้นตอนวิธีลดทอนทางเวลา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เป็นต้น สมการ (2.1.15) บอกเราว่าในการคำนวณ DFT ของลำดับคู่หนึ่ง จะประกอบด้วยลำดับ $X(k)$ ใน (2.1.15 a) และลำดับ X(k) ใน (2.1.15b) ซึ่งห่างออกไปจากลำดับ X(k) ใน (2.1.15a) ไป N/2 จุดนั้น สามารถคำนวณได้โดยใช้การคูณจำนวนได้โดยใช้การคูณจำนวนเชิงซ้อนเพียงครั้งเดียวเท่านั้น จากผลนี้เราสามารถนำไปสร้างหน่วยคำนวณที่มีชื่อเรียกว่า หน่วยผีเสื้อ (butterfly unit) โดยหน่วยคำนวณนี้ (ซึ่งอาจสร้างอยู่ในรูปแบบของวงจรหรือโปรแกรม) มีข้อมูลเข้าสองข้อมูล คือ A และ B และให้ข้อมูลออกเป็น X และ Y และ

$$X = A + (W_N)^k \cdot B$$
$$Y = A - (W_N)^k \cdot B \qquad (2.1.16)$$

โดยที่การทำงาน หน่วยผีเสื้อนี้ สามารถเขียนแทนด้วย กราฟสัญญาณไหล (Signal flow graph) ดังรูปที่ 2.2 (a) , 2.2 (b) หรือ 2.2 (c) ในบางครั้ง ปม หรือ บัพ (node) ของลำดับข้อมูล A และ B ตาม (2.1.15) ซึ่งเป็นลำดับที่อยู่ห่างกัน N/2 จุดนี้จะเรียกว่า บัพคู่ หรือ ปมคู่ (dual node)

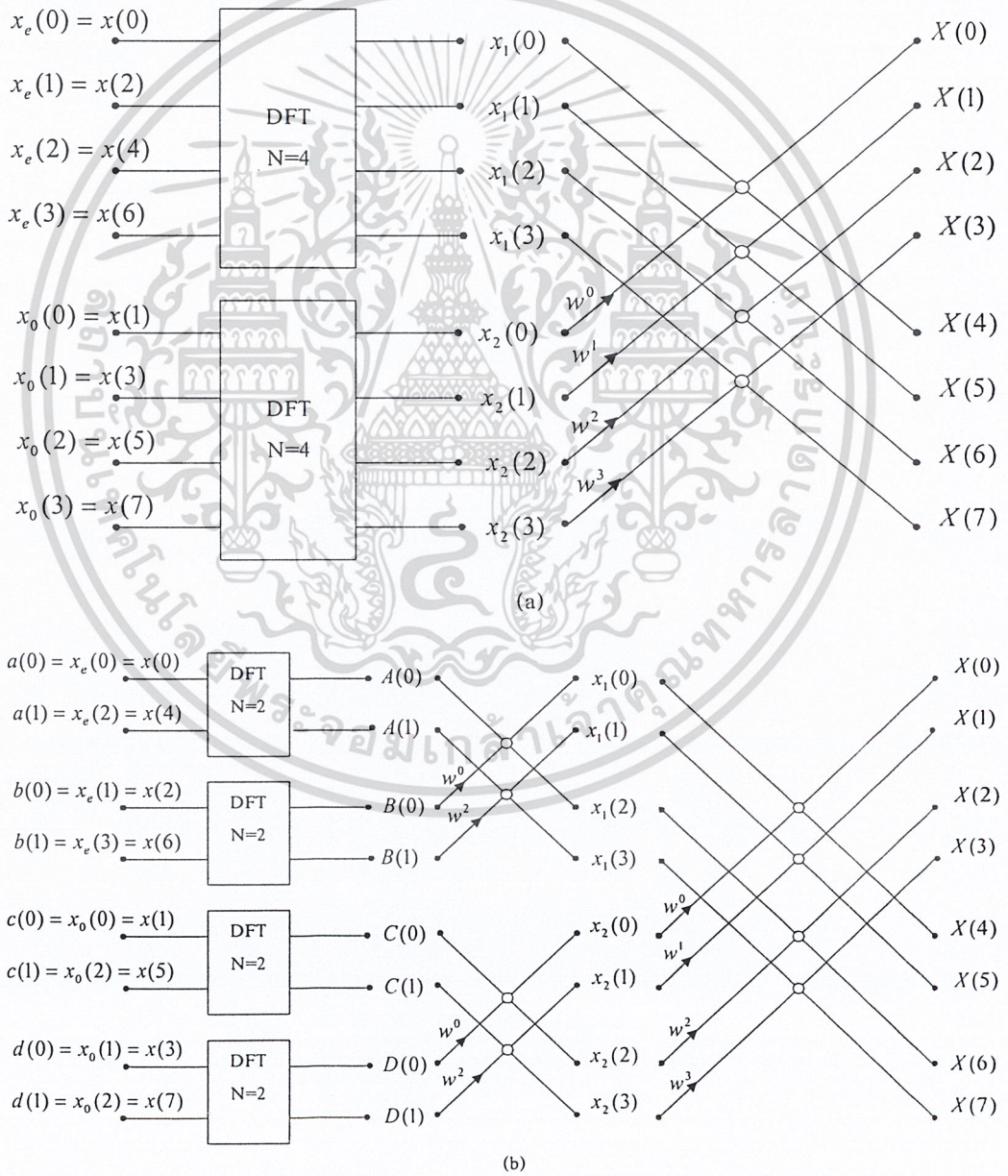
ต่อไปนี้ขอแสดงตัวอย่างการคำนวณ DFT แบบ 8 จุด โดยอาศัยวิธีการอธิบายมาแล้ว ขั้นตอนในการคำนวณได้แสดงอย่างเป็นลำดับไว้ด้วยกราฟการไหลของสัญญาณ ดังรูปที่ 2.3 โดยในเบื้องแรก ลำดับของข้อมูลเข้า $x(n)$ จะถูกแบ่งออกเป็นลำดับคู่ $x_E(m)$ และลำดับคี่ $x_0(m)$ อย่างละสี่ลำดับ ดังรูปที่ 2.3(a) ต่อจากนั้นลำดับ $x_E(m)$ และ $x_0(m)$ ก็ถูกแบ่งย่อยออกไปอีกดังรูปที่ 2.3 (b) ซึ่งในที่สุดการคำนวณ DFT ขนาด 8 จุด ถูกสับย่อยออกเป็นการคำนวณ DFT ขนาด 2 จุด 4 ภาคด้วยกัน ดังปรากฏในรูปที่ 2.4 (a) ซึ่งเป็นผังการไหลของสัญญาณสำหรับการคำนวณ DFT ขนาด 8 จุด อย่างสมบูรณ์ สังเกตเห็นว่าในกรณี N=8 ทำให้ $\log_2 8 = 3$ การคำนวณจึงแบ่งเป็น 3 ภาคย่อย แต่ละภาคประกอบด้วยหน่วยผีเสื้ออยู่ N/2=4 หน่วยและหน่วยผีเสื้อแต่ละหน่วยใช้การคูณเลขเชิงซ้อน 1 ครั้ง เพราะฉะนั้นการคำนวณทั้งหมดมีการคูณเลขเท่ากับ $(8/2)\log_2 8 = 12$ ครั้ง เมื่อเปรียบเทียบกับการคำนวณปกติที่ต้องการคูณกันถึง $8 \times 8 = 16$ ครั้ง

ตามรูปที่ 2.4 (a) นี้มีข้อน่าสังเกตก็คือ ลำดับสัญญาณเข้า $x(n)$ ไม่ได้จัดเรียงอย่างต่อเนื่อง แต่ได้ถูก สลับอันดับ (Scrambled order) อย่างมีหลักเกณฑ์ คือ การสลับอันดับกันนี้จะเป็นไปตามวิธีการที่เรียกว่า การผันกลับบิต (bit-reversed) นั่นคือ ถ้าเราแทนดรรชนี n ของลำดับ $x(n)$ ด้วยเลขฐานสอง โดยที่จำนวนบิตของเลขฐานสองต้องเพียงพอที่จะแทนค่า N ได้เช่น ในกรณี N = 8 ก็ต้องแทนด้วยเลขฐานสอง 3 บิต จากนั้นการจัดลำดับ $x(n)$ ใหม่จะได้จากการผันกลับหลักของเลขไบนารีที่แทนดรรชนี n ดังแสดงในรูปที่ 2.4 (b) คือ $x(001)$ จะถูกแทนที่ด้วย $x(100)$ หรือ $x(110)$ ถูกแทนที่ด้วย $x(011)$ เป็นต้น จากที่กล่าวแล้วว่าปกติดรรชนี n มักเกี่ยวข้องกับเวลาและวิธีการของ FFT แบบนี้เป็นการลดเวลาการคำนวณโดยการ สับ หรือ ตัดทอน ลำดับลำดับในโดเมนเวลา หรือ $x(n)$ ออกเป็นกลุ่มย่อย ที่แต่ละกลุ่มประกอบด้วยลำดับ $x(n)$ เพียงสองลำดับ ซึ่งสองลำดับนี้เป็นปมคู่กัน การจัดกลุ่มนี้คล้ายกับเป็นการสุ่มตัวอย่างลำดับ เดิมอีกครั้งด้วยอัตราการสุ่มตัวอย่างที่ต่ำกว่า และถ้าหากเราถือว่าแต่ละกลุ่มลำดับข้อมูลใหม่ที่จัดขึ้นมารวมเป็นลำดับข้อมูล 1 ลำดับแล้ว ก็เท่ากับว่าเราได้ตัดทอนลำดับในโดเมนเวลาลงไปเป็น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

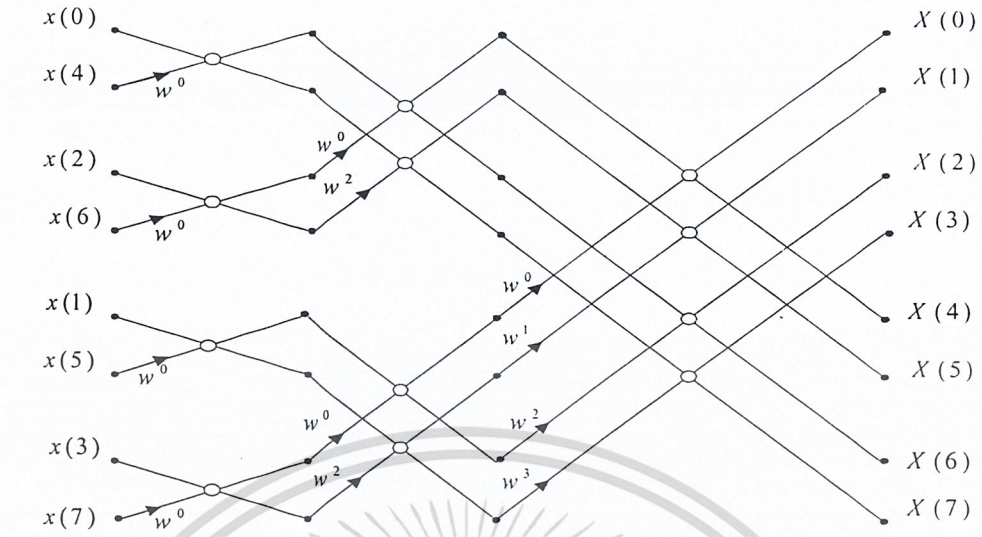
กลุ่มลำดับข้อมูลย่อยหลายลำดับ ดังนั้นวิธีการคำนวณแบบนี้จึงมีชื่อ เรียกว่า การลดทอนทางเวลา ถึงตอนนี้เราอาจเขียนแผนแทนลำดับวิธีการของ DFT ได้ดังรูปที่ 2.5 โดยที่ DFT ขนาด N จุด เดิมถูกแบ่งออกเป็น DFT ขนาด N/2 จุด จำนวน 2 ภาคนำมารวมกันโดยการใช้ ตัวประกอบหมุน และลำดับการนี้จะทำจนกระทั่งผลสุดท้ายเป็นการแปลง DFT ขนาด 2 จุด พึงสังเกตว่าการนำเอา DFT ขนาด N/2 จุด มาประกอบกันขึ้นมีวิธีการหลายแบบขึ้นอยู่กับการจัดเรียงตัวประกอบหมุน ถ้าหากการจัดเรียงตัวประกอบหมุนที่ต่างกันออกไปก็จะได้ FFT ชนิดที่ต่างกันออกไปด้วยซึ่งจะได้แสดงให้เห็นต่อไป

ที่ได้กล่าวไว้ว่า การคำนวณ DFT ขนาด N จุดโดยใช้ FFT นั้นจำนวนครั้งในการคูณจำนวนเชิงซ้อนมีแค่ N $\log_2$ N นั้น ความเป็นจริงค่านี้เป็นเพียงค่าประมาณและใช้กับ FFT โดยทั่วๆ ไป ทั้งนี้ลองย้อยกลับไปดูลำดับการคำนวณตามรูปที่ 2.4 (a) แล้วจะเห็นว่าส่วนแรกซึ่ง

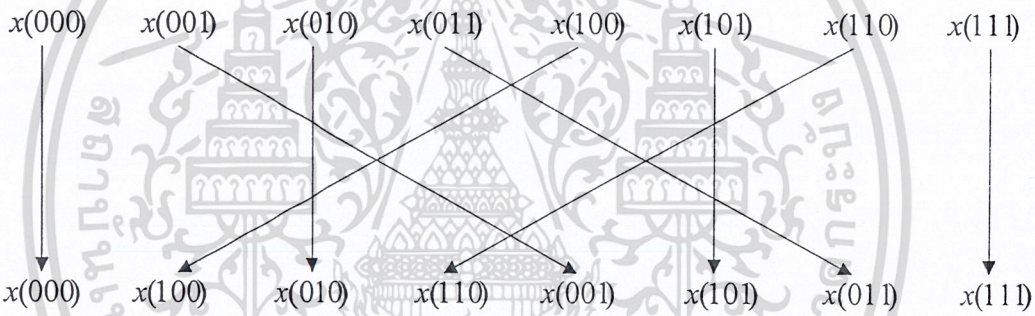


รูปที่ 2.3 (a) และ (b) แสดงขั้นตอนวิธีแบบ DIT สำหรับ DFT แบบ 8 จุด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



(a)



(b)

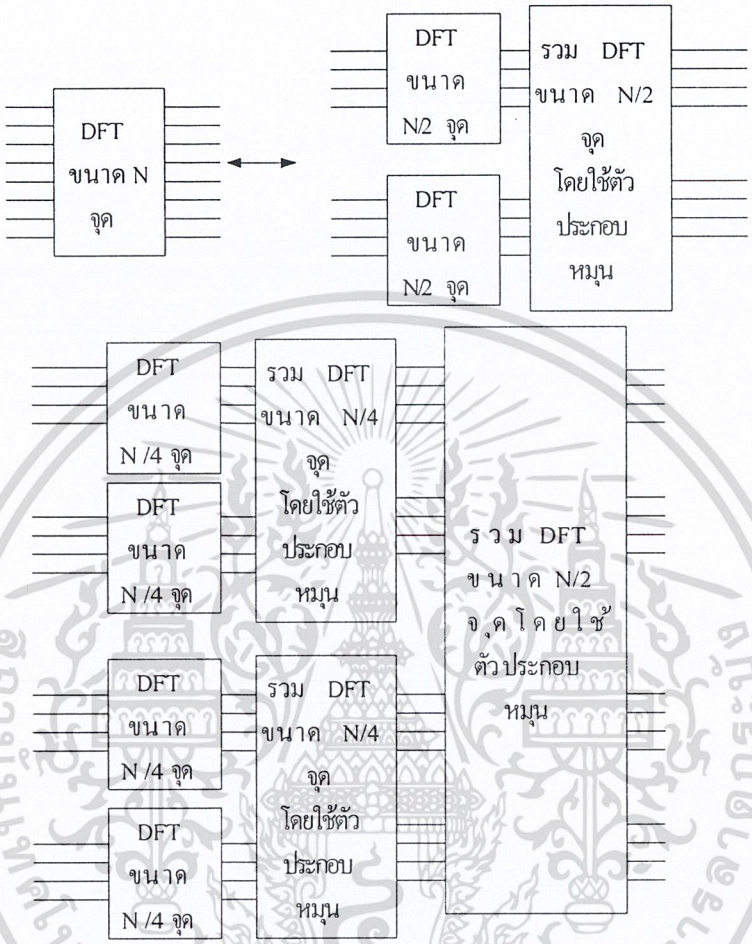
รูปที่ 2.4 (a) กราฟการไหลสัญญาณแสดงการคำนวณตามรูป 2.3
(b) แสดงการสลับตำแหน่งของลำดับ $x(n)$ ด้วยการผันกลับบิต

เป็น DFT ขนาด 2 จุดนั้น การคำนวณมีเฉพาะการบวก และ การลบ จำนวนเชิงซ้อนเท่านั้น ในส่วนที่สองก็เช่นเดียวกัน การคูณเชิงซ้อนมีเฉพาะในส่วนที่สามารถและใช้การคำนวณเชิงซ้อนเพียง 2 ครั้งเท่านั้น แต่ถ้าใช้สูตรการคำนวณ $N \log_2 N$ ก็จะได้ 24 ครั้ง (หรือ 12 ครั้ง เมื่อใช้ความสัมพันธ์ของหน่วยผีเสื้อ) ผลที่ได้แตกต่างกันมาก อย่างไรก็ตามสำหรับกรณี N มีค่าใหญ่มากการที่จะเขียนโปรแกรมหรือสร้างวงจรซึ่งสามารถใช้กับกรณีทั่วๆไปได้ด้วยนั้น ข้อสังเกตเหล่านั้นจำต้องมีโปรแกรมหรือวงจรส่วนหนึ่ง เพื่อมาวิเคราะห์ด้วย ทำให้เกิดความยุ่งยากโดยไม่จำเป็น ดังนั้น จึงสรุปได้ว่า การคูณจำนวนเชิงซ้อนของลำดับ FFT จะมีปริมาณ $N \log_2 N$ ครั้ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.1.3 การลดทอนทางเวลาแบบอื่น

ขั้นตอนวิธีแบบการคำนวณซ้ำที่ มาดูคุณสมบัติบางประการซึ่งเป็นข้อดีของการแปลงฟาสท์ฟูเรียร์ ตามรูปที่ 2.4 (a)



รูปที่ 2.5 ภาพรวมแสดงขั้นตอนวิธีการคำนวณ DFT ขนาด n จุด แบบลดทอนทางเวลา

โดยทั่วไปแล้วอาจกล่าวได้ว่าการคำนวณการแปลง DFT นั้น แท้จริงก็คือ การนำกลุ่มลำดับจำนวนเชิงซ้อนที่มีอยู่ N ลำดับ มาทำการแปลงให้เป็นลำดับจำนวนเชิงซ้อนอีกกลุ่มหนึ่งที่มีอยู่ N ลำดับเช่นกัน โดยการใช้ FFT การแปลงเช่นนี้จะกระทำ $\log_2 N$ ขั้นตอนด้วยกัน ดังนั้นตามรูปที่ 2.4 (a) ซึ่ง N=8 ควรต้องมีหน่วยความจำถึง $\log_2 N$ หรือ 3 แถวลำดับ (array) ด้วยกันสำหรับเก็บข้อมูลที่ต้องใช้ในการคำนวณโดยที่ แถวลำดับแรกไว้เก็บลำดับ $x(n)$ สองแถวลำดับต่อมาไว้เก็บผลการคำนวณระหว่างกลาง $x_1(k)$ และแถวลำดับสุดท้ายสำหรับผลลัพธ์ $x(n)$ ที่นี้มาพิจารณาลำดับการคำนวณที่อธิบายด้วย กราฟการไหล ตามรูปที่ 2.4 (a) นั้น การคำนวณประกอบขึ้นด้วย หน่วยผีเสื้อ ซึ่งลักษณะการคำนวณของหน่วยผีเสื้อนี้ ถ้าหากเรามีหน่วยความจำต่างหากไว้สำหรับเก็บค่าผลคูณของจำนวนเชิงซ้อน $(W_N)^k \cdot B$ ผลลัพธ์ X กับ Y ที่คำนวณสามารถเก็บแทนที่ไว้ในหน่วยความจำที่เก็บลำดับข้อมูลเข้า A และ B ได้ (ดูรูปที่ 2.2 ประกอบ) โดยลักษณะการคำนวณเช่นนี้จะเห็นว่า ตามรูปที่ 2.4(a) ผลลัพธ์ในการคำนวณทาง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ด้านขวามือสามารถบรรจุ แทนที่ในหน่วยความจำทางด้านซ้ายมือได้โดยไม่มีผลต่อการคำนวณส่วนอื่นๆ การคำนวณเหล่านี้จึงเรียกว่า การคำนวณซ้ำที่ (in place computation) ซึ่งข้อดีก็คือ ใช้หน่วยความจำเพียง 1 แถวลำดับ หรือ ต้องการหน่วยความจำสำหรับเก็บจำนวนเชิงซ้อนเพียง N + 1 ค่าเท่านั้น วิธีการนี้จึงเหมาะกับ ลำดับข้อมูลที่ยาวมาก โดยการคำนวณไม่เปลืองเนื้อที่หน่วยความจำ ข้อดีอีกประการหนึ่งของ DFT แบบคำนวณซ้ำที่ ก็คือ ตัวประกอบหมุน (W_N)^K นั้นถูกจัดเรียงอย่างเป็นลำดับคือ จากกำลังน้อยไปสู่กำลังมาก จึงทำให้การเขียนโปรแกรม หรือสร้างวงจรทำได้ง่าย อย่างไรก็ตามสำหรับลำดับการคำนวณรูปที่ 2.4(a) นั้น มีข้อเสียที่ว่า ลำดับ x(n) ต้องมีการสลับที่ตามวิธี ผันกลับบิตเสียก่อนก่อนการคำนวณ นั่นหมายถึงว่าเราต้องมีโปรแกรมหรือ วงจรเพิ่มเติมเพื่อทำการจัดเรียงข้อมูลนี้ใหม่

2.1.4 FFT ชนิดลดทอนทางความถี่ (Decimation – in – Frequency หรือ DIF)

ลำดับการคำนวณ FFT อีกแบบหนึ่งที่นิยมใช้กันอย่างแพร่หลายนั้นมีชื่อเรียกว่า การลดทอนทางความถี่ หลักการสำคัญของ DIF นั้นคล้ายกับ DIT โดยที่ DIF ทำการแบ่งลำดับโดเมนเวลา x(m) ออกเป็นสองส่วนเท่าๆกัน โดยการแบ่งครึ่งทำโดย ถ้าให้ $x_E(m)$ และ $x_O(m)$ แทนลำดับสัญญาณที่ได้จากการแบ่งครึ่งนี้

$$\begin{aligned} x_E(m) &= x(m) \qquad ;m = 0,1,....,(N/2)-1 \\ x_0(m) &= x(m+N/2)\ ;m = 0,1,....,(N/2)-1 \end{aligned} \tag{2.1.17}$$

เพราะฉะนั้นการคำนวณ DFT ขนาด N จุดของ x(n) สามารถเขียนแยกได้เป็น 2 ส่วนคือ

$$\begin{aligned} X(k) &= \sum_{m=0}^{(N/2)-1} x(m)(W_N)^{mk} + \sum_{m=N-2}^{N-1} x(m)(W_N)^{mk} \\ &= \sum_{m=0}^{(N/2)-1} x_E(m)(W_N)^{mk} + \sum_{m=0}^{(N/2)-1} x_0(m)(W_N)^{(m+n/2)k} \\ X(k) &= \sum_{m=0}^{(N/2)-1} x_E(m)(W_N)^{mk} + (W)^{Nk/2} \sum_{m=0}^{(N/2)-1} x_0(m)(W_N)^{mk} \end{aligned} \tag{2.1.18}$$

โปรดสังเกตว่า เครื่องหมายรวมยอด (summation sign) ในสมการ 2.1.18 ซึ่งแสดงถึงผลรวมของการคำนวณใน N/2 จุดนั้น ผลรวมนี้ไม่ใช่ DFT ขนาด N/2 จุดเพราะว่าพจน์ $(W_N)^{mk}$ มีค่า $(W_{N/2})^{mk} \neq (W_N)^{2mk}$

สมการ (2.1.18) นั้นเพราะว่าพจน์ $(W)^{Nk/2} = \exp(-j\pi k) = (-1)^k$ จึงเขียนได้เป็น

$$X(k) = \sum_{m=0}^{(N/2)-1} [x_E(m) + (-1)^k \cdot x_0(m+N/2)](W_N)^{mk} \tag{2.1.19}$$

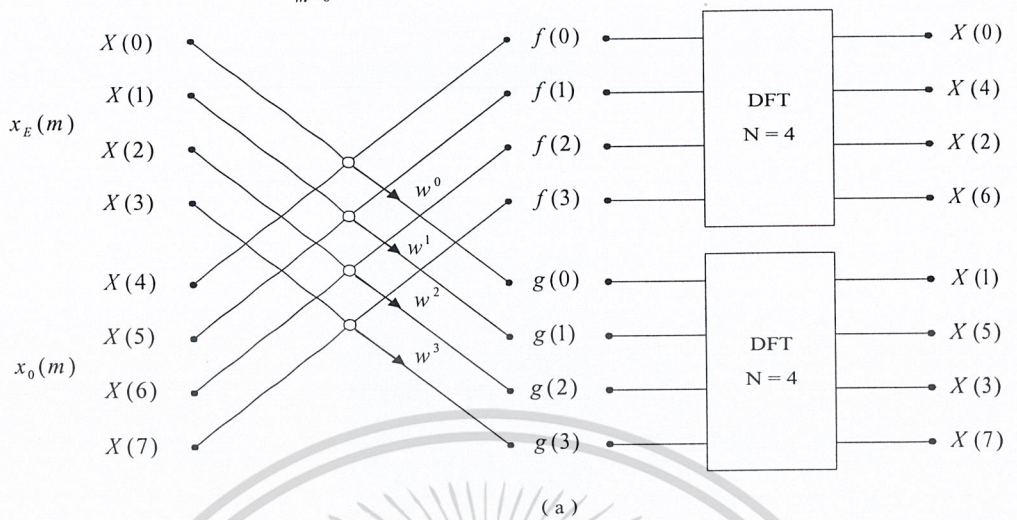
ถ้าเราเขียนแยกดรรชนี k ออกเป็นเลขคู่และคี่ เพราะฉะนั้น X(2k) และ X(2k+1) จะแทน DFT ที่เป็นส่วนของเลขคู่และคี่ตามลำดับ หรือ

$$\begin{aligned} X(2k) &= \sum_{m=0}^{(N/2)-1} [x_E(m) + x_0(m)](W_N)^{2mk} \\ &= \sum_{m=0}^{(N/2)-1} f(m)(W_{N/2})^{mk} \end{aligned} \tag{2.1.20}$$

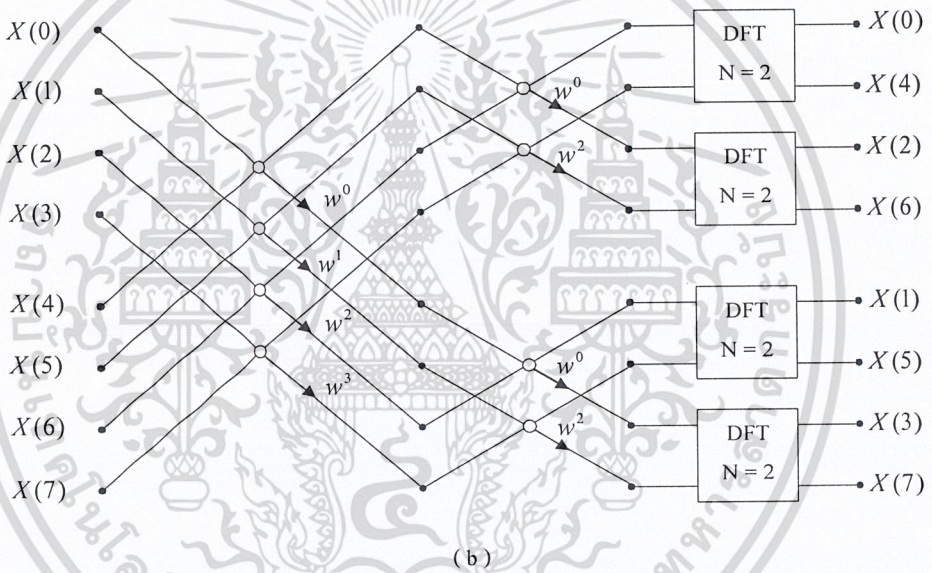
และ $$X(2k+1) = \sum_{m=0}^{(N/2)-1} [x_E(m) - x_0(m)](W_N)^m (W_N)^{2mk}$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

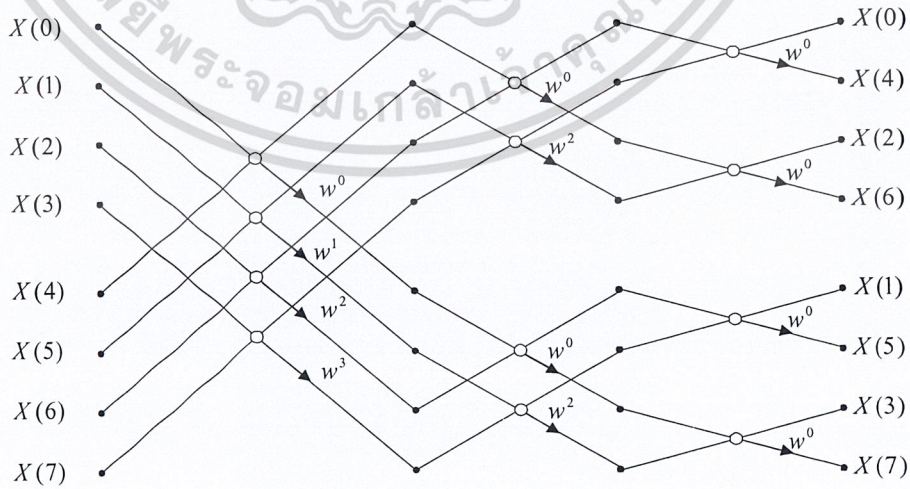
$$= \sum_{m=0}^{(N/2)-1} g(m)(W_{N/2})^{mk} \qquad (2.1.21)$$



(a)



(b)



(c)

รูปที่ 2.6 แสดงลำดับขั้นตอนวิธีของ FFT ชนิดการลดทอนทางความถี่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากผลของ (2.1.20) และ (2.1.21) นี้แสดงว่าการคำนวณ DFT ขนาด N จุดด้วยวิธีนี้จะทำได้โดย ในเบื้องแรกจากลำดับ x(m) เราทำการแบ่งครึ่งออกเป็น 2 ลำดับ ดังสมการ (2.1.17) แล้วนำมาสร้างเป็นลำดับใหม่นี้ชื่อ f(m) และ g(m) ลำดับคู่นี้สามารถสร้างโดยการใช้สมการ

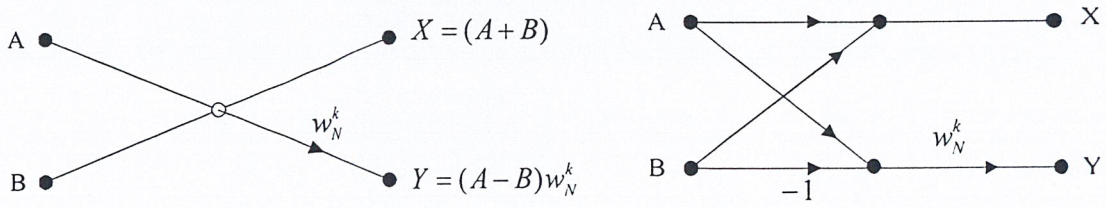
$$f(m) = x_E(m) + x_0(m) \qquad ; m = 0,1,\ldots,(N/2)-1$$
$$g(m) = [\, x_E(m) - x_0(m) \,].\,(W_N)^m \qquad ; \qquad (2.1.22)$$

และจากลำดับความยาว N/2 จุดในสมการ (2.1.22) นี้ เราก็นำไปคำนวณหา DFT ขนาด N/2 จุดโดยใช้ (2.1.20) และ (2.1.21) เพราะฉะนั้นโดยภาพรวมแล้วจะเห็นว่า การคำนวณ DFT ขนาด N จุด ได้ถูกแบ่งแยกออกเป็นการคำนวณ DFT ขนาด N/2 จุดสองภาคด้วยกัน

ดังเช่นวิธีการของ DIT การแบ่งการคำนวณย่อยออกเป็น DFT ขนาด N/2 จุดนี้สามารถแยกย่อยออกไปได้เรื่อยจนในที่สุดเหลือเพียงเป็นการคำนวณขนาด 2 จุด ตัวอย่างที่แสดงการคำนวณโดยใช้กราฟการไหล สำหรับกรณี N = 8 ได้แสดงไว้ในรูปที่ 2.6 โดยในรูปเป็นการแสดงการแบ่งย่อยออกตามลำดับนั้น จนเหลือเป็นการคำนวณ DFT ขนาด 2 จุด และการแบ่งย่อยทำได้ 3 ครั้ง หรือ $\log_2 8$ และจำนวนครั้งในการคูณจำนวนเชิงซ้อนก็จะประมาณ $N \log_2 N$ เช่นเดียวกันกับแบบ DIT เมื่อเปรียบเทียบ DIT และ DIF โดยการใช้รูปที่ 2.3 และ 2.6 แล้วเห็นได้ชัดว่าข้อแตกต่างที่สำคัญระหว่างทั้งสองขั้นตอนวิธีการคำนวณ ก็คือ ประการแรก DIT ข้อมูล x(n) ต้องทำการสลับตำแหน่งกัน แต่ X(k) เรียงตามธรรมชาติ ส่วน DIF จะกลับตรงกันข้าม x(n) จะเรียงตามธรรมชาติ และ X(k) ถูกเรียงสลับแบบผันกลับบิต ประการที่สอง หน่วยผีเสื้อของ DIF ต่างออกไปจากของ DIT เพราะได้เอาลำดับ $x_E(m)$ และ $x_0(m)$ มาบวกและลบกันก่อนแล้วจึงทำการคูณด้วยจำนวนเชิงซ้อน $(W_N)^k$ หน่วยคำนวณผีเสื้อของ DIF ได้แสดงไว้ในรูปที่ 2.7 สำหรับการจัด FFT แบบ DIF ใหม่เพื่อให้ลำดับการคำนวณอื่นๆนั้นสามารถทำได้โดยใช้วิธีการเดียวกับ DIF

2.1.5 การแปลงฟาสท์ฟูเรียร์สำหรับ N ใดๆ

จากที่กล่าวมาเรากำหนดให้ $N = 2^r$ แล้วทำการคำนวณหา DFT จะทำโดยการจัดเรียงลำดับ x(n) ออกเป็นลำดับแบบสองมิติ และทำการคำนวณ DFT แบบสองจุด ในทางปฏิบัติถ้าหาก



รูปที่ 2.7 แสดงการคำนวณของ หน่วยผีเสื้อ ของขั้นตอนวิธีชนิดลดทอนทางความถี่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$N \neq 2^r$ แล้ว เราก็จะเติมค่าลำดับ $x(n) = 0$ เข้าไปจนกระทั่ง N มีค่าเป็นสองยกกำลังใดๆ แล้วจึงทำการคำนวณหา DFT ข้อด้อยที่เห็นชัดคือกรณีที่ N มีค่ามาก r ก็ต้องมีค่ามากตาม ทำให้ในบางครั้งเราอาจจะต้องเติมลำดับ $x(n) = 0$ เข้าไปมาก ซึ่งสิ้นเปลืองหน่วยความจำ และเวลาในการคำนวณก็มากขึ้นโดยไม่จำเป็น นอกจากนี้ DFT ที่ได้ก็จะเป็นการประมาณค่า (interpolate) ของ DFT ของลำดับสัญญาณเดิมที่ยังไม่ได้เติมลำดับค่าศูนย์เข้าไป

วิธีการของ DIT และ DIF ที่ได้อธิบายมา สามารถนำไปใช้กับกรณี N มีค่าเป็นเลขเฉพาะ (Prime) ที่ไม่สามารถแยกออกเป็นผลคูณของตัวเลขจำนวนเต็มได้ หรือ N ใช้กับ N ที่เป็นเลขประกอบ (composite) เพราะฉะนั้นโดยทั่วไป N สามารถเขียนได้เป็น

$$N = r_1 \cdot r_2 \cdot r_3 \cdot \ldots r_n \tag{2.1.23}$$

ถ้ากรณี N เป็น เลขประกอบมาก (highly composite) แล้ว $r_1 = r_2 = r_3 = \ldots = 2$ ก็จะเป็นกรณีของ $N = 2^r$ ที่ได้กล่าวมา ถ้าหากความยาวของลำดับสามารถเขียนอยู่ในรูปของ $N = M \times L$ โดยที่ M เป็นจำนวนลำดับใน แนวตั้ง (column) และ L เป็นจำนวนลำดับใน แนวนอน (row) เราอาจแบ่งลำดับหนึ่งมิติ (one-dimension) x(n) นี้ออกเป็น ลำดับสองมิติ (two-dimension) x(1,n) ได้คือ

$$\begin{array}{lll} x(0) & x(1) & \text{---------- } x(M-1) \\ x(M) & x(M+1) & \text{---------- } x(2M-1) \\ x(2M) & x(2M+1) & \text{---------- } x(3M-1) \\ \vdots & \vdots & \vdots \\ x((L-1)M) & x((L-1)(M+1)) & \text{---------- } x(N-1) \end{array} \tag{2.1.24}$$

หรือดรรชนี m ถูกเขียนแทนด้วย

$$n = Ml + m \tag{2.1.25}$$

เมื่อเราเขียนลำดับ x(n) อยู่ในรูปของลำดับสองมิติแล้ว ในทำนองเดียวกัน DFT ของลำดับนี้ก็ควรจะอยู่ในรูปของลำดับสองมิติด้วย ดังนั้น ถ้าเราให้ r และ s เป็นตัวแปรของลำดับ x(k) ตามแนวตั้งและแนวนอนตามลำดับเราจะได้

$$k = Lr + s \tag{2.1.26}$$

เพราะฉะนั้นจากสมการ (2.1.1) โดยการแทนค่าดรรชนี m และ k โดยใช้สมการ (2.1.25) และ (2.1.26) จะได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

$$x(k) = X(s,r) = \sum_{m=0}^{M-1}\sum_{l=0}^{L-1} x(l,m)W^{(Ml+m)(Lr+s)} \qquad (2.1.27)$$

โดยที่พจน์ $W^{ML\,lr} = W^{Nlr} = 1$ เพราะฉะนั้นสมการ (2.1.27) จึงเขียนรวมได้เป็น

$$X(s,r) = \sum_{m=0}^{M-1} W^{Lmr} \cdot W^{ms} \sum_{l=0}^{L-1} x(l,m)W^{Msl} \qquad (2.1.28)$$

การคำนวณสมการ (2.1.28) ขึ้นอยู่กับว่าเราจะคำนวณเครื่องหมายรวมยอดอันไหนก่อนหรือหลัง และการเลือกคำนวณเครื่องหมายรวมยอดที่ต่างกันก็ได้วิธีการที่แตกต่างกันออกไป ในตอนแรกเรามาดูลำดับการคำนวณแบบ DIT ซึ่งทำได้โดย

1. คำนวณ DFT ขนาด L จุด ของลำดับในแนวตั้งแต่ละแถว นั่นคือ การคำนวณ

$$q(s,m) = \sum_{l=0}^{L-1} x(l,m)W^{Msl} \qquad (2.1.29)$$

และทำให้ (2.1.28) กลายเป็น

$$X(s,r) = \sum_{m=0}^{M-1} W^{Lmr} \cdot W^{ms} \cdot q(s,m) \qquad (2.1.30)$$

2. สร้างลำดับสองมิติขึ้น ด้วยการคูณลำดับ q(s,m) เข้ากับ ตัวประกอบหมุน W^{ms} หรือ

$$h(s,m) = W^{ms} \cdot q(s,m) \qquad (2.1.31)$$

3. สร้าง DFT ขนาด M จุดของลำดับในแต่ละแถวในแนวนอนของลำดับสองมิติ h(s,n)

ในทำนองเดียวกันลำดับการคำนวณแบบ DFT สามารถทำในลักษณะคล้ายคลึงกันด้วยเขียนสมการ (2.1.28) ใหม่เพื่อความสะดวก

$$Z(s,r) = \sum_{l=0}^{L-1} W^{Msl} \sum_{m=0}^{M-1} x(l,m)W^{ms} \cdot W^{Lrm} \qquad (2.1.32)$$

รูปที่ 2.8 แสดงถึงลักษณะการคำนวณ DFT ขนาด 15 จุด ซึ่งแยกย่อยออกเป็น L = 5 และ M = 3 และเป็นการคำนวณแบบ DIT ถ้าพิจารณาลำดับการคำนวณทั้งสองแบบที่ได้กล่าวมาเห็นได้ชัดว่าการคำนวณแบบ DIT และ DIF นั้นคล้ายกันมาก ที่ต่างกันก็คือแบบ DIT ตัวประกอบหมุนจะคูณในลำดับการคำนวณขั้นที่สอง แต่แบบ DIF ตัวประกอบหมุนคูณก่อน

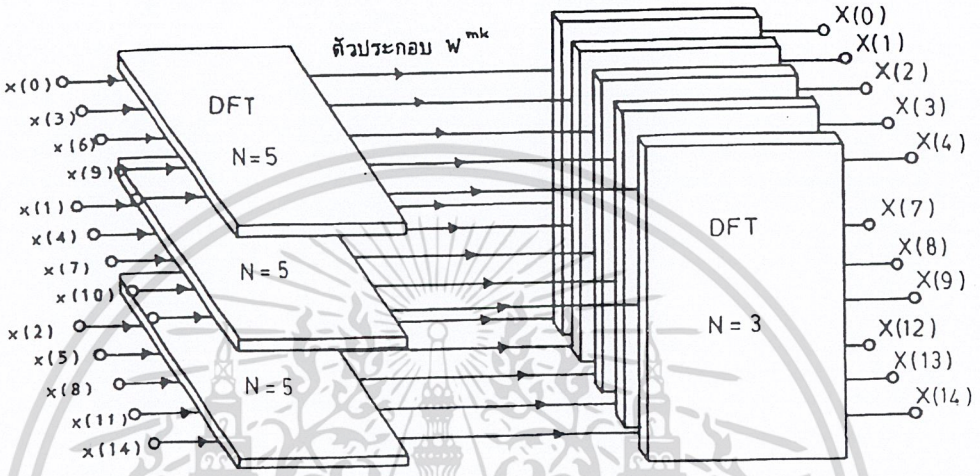
ถ้าหากเราใช้หลักการที่ว่า การคำนวณ DFT ขนาด N จุดต้องมีการคำนวณ (คูณและบวก) จำนวนเชิงซ้อนโดยประมาณ N^2 ครั้ง เพราะฉะนั้นการคำนวณ DFT ตามสมการ (2.1.31) และ (2.1.32) ก็ต้องมีการคำนวณจำนวนเชิงซ้อนเท่ากับ $ML^2 + LM^2$ ครั้งเมื่อรวมการคำนวณเชิงซ้อนที่เกิดจากการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

46462

คูณค่าของตัวประกอบตามสมการ (2.1.31) เข้าด้วยกันจึงสามารถสรุปได้ว่าจำนวนครั้งของการคำนวณจำนวนเชิงซ้อนของวิธีการนี้มีค่าเท่ากับ

$$C_2 = ML^2 + LM^2 + LM = LM(L+M+1) + N(M+L+1) \qquad (2.1.33)$$



รูปที่ 2.8 ภาพรวมแสดงขั้นตอนวิธีการคำนวณ DFT ในกรณี N มีค่าใดๆ

ซึ่งค่านี้จะมีค่าน้อยกว่า N^2 และถ้าหาก M และ L อยู่ในรูปของสองยกกำลังใดๆ แล้วเราสามารถใช้ FFT ที่กล่าวมาในหัวข้อ 2.1 เข้ามาช่วยคำนวณ และ

$$C_2 = N(\log_2 L + \log_2 M + 1) \qquad (2.1.34)$$

ในทำนองเดียวกัน ค่า N อาจถูกแบ่งออกเป็นผลคูณจำนวนเฉพาะสามค่าได้ เช่น N = PML ถ้าหากเราวิเคราะห์เช่นเดียวกันตอนหาค่า C_2 และ C_3 เป็นจำนวนครั้งในการคำนวณจำนวนเชิงซ้อนแล้วเราจะได้ว่า

$$C_3 = P^2 LM + PML^2 + 2PLM \quad N(P+L+M+2) \qquad (2.1.35)$$

เพราะฉะนั้น สำหรับเมื่อ $N = N_1N_2N_3 \ldots\ldots N_J$ แล้วเราสามารถเขียนสมการแสดงถึงจำนวนครั้งโดยประมาณที่ต้องใช้ในการคำนวณเชิงซ้อนโดยวิธีการนี้ได้เป็น

$$C_J = N(\sum_{i=1}^{J} n_1 + J - 1) \qquad (2.1.36)$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2 หลักการเบื้องต้นของโครงสร้างเลขคณิตกระจาย

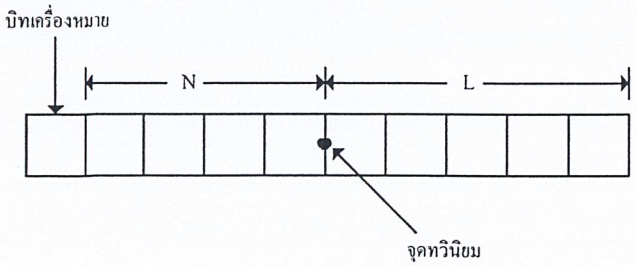
โครงสร้างเลขคณิตกระจาย (Distributed Arithmetic) หรือเรียกอย่างย่อๆว่า "DA" เป็นการจัดรูปแบบทางคณิตศาสตร์ของสมการที่อยู่ในรูปของผลบวกของผลคูณ (Sum of Product) ของเลขฐานสิบให้กระจายออกเป็นบิทหรือในรูปแบบของเลขฐานสอง เพื่อให้การคำนวณทางคณิตศาสตร์สามารถแปลงเป็นวงจรดิจิตอลอิเล็กทรอนิกส์ได้ โดยจะปรากฏอยู่ในงานทางด้านการประมวลผลสัญญาณเชิงเลข หลักการเบื้องต้นของโครงสร้างเลขคณิตกระจายที่นำมาใช้งานด้านการประมวลผลสัญญาณเชิงเลข คือการแปลงฟังก์ชันถ่ายโอน (Transfer Function) ซึ่งเป็นสมการที่อยู่ในรูปผลบวกของผลคูณระหว่างสัญญาณอินพุทกับค่าสัมประสิทธิ์ของวงจรกรอง โดยในการคูณกันระหว่างค่าสัมประสิทธิ์ของวงจรกรองกับสัญญาณอินพุท จะใช้การคูณเลขฐานสองแบบส่วนเติมเต็มสอง(2's complement) และการคูณจะใช้แบบเปิดตาราง (Look-up table) โดยค่าผลบวกของผลคูณระหว่างสัมประสิทธิ์และสัญญาณอินพุทจะถูกเก็บไว้ในหน่วยความจำ EPROM และจะใช้สัญญาณอินพุทเป็นแอดเดรสของ EPROMโดยตรง ทั้งค่าสัมประสิทธิ์ของวงจรกรองและสัญญาณอินพุทจะถูกแทนด้วยเลขส่วนเติมเต็มสอง ดังนั้นโครงสร้างเลขคณิตกระจายจึงมีทฤษฎีพื้นฐานอยู่บนการคูณแบบเลขส่วนเติมเต็มสอง (2's complement Multiplication)

2.2.1 ระบบตัวเลข

สำหรับระบบเชิงเลข ตัวเลขต่างๆจะถูกแทนด้วยเลขฐานสอง ซึ่งโดยทั่วไปมีรูปแบบที่นิยมใช้กันอยู่ 2 รูปแบบ คือ รูปแบบจำนวนโดยตรง (Fixed point format) และ รูปแบบจำนวนอิงดรรชนี (Floating point format) ซึ่งรูปแบบจำนวนโดยตรงจะมีวงจรฮาร์ดแวร์ที่ใช้ในการคำนวณที่ง่ายกว่า แต่ให้ค่าจากการคูณค่อนข้างจำกัด ส่วนรูปแบบจำนวนอิงดรรชนีจะสามารถแทนค่าของสัญญาณ คือให้ย่านพลวัตร (Dynamic range) ได้มากกว่า แต่ต้องใช้วงจรฮาร์ดแวร์ที่สลับซับซ้อน แพงกว่า และให้ความเร็วในการประมวลผลที่ลดลง

2.2.1.1 รูปแบบจำนวนโดยตรง

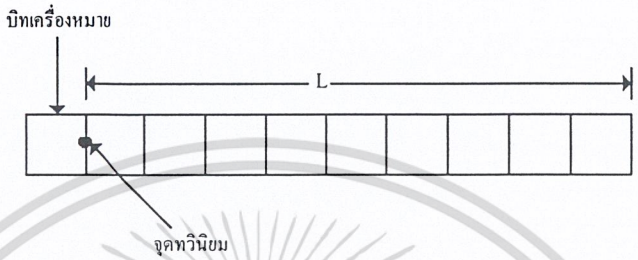
รูปแบบจำนวนโดยตรงปกติจะประกอบไปด้วย 3 ส่วน คือ บิทเครื่องหมาย (Sign bit) 1 บิท บิทจำนวนเต็ม (Integer bit) N บิท และบิทเศษส่วน (Fractional bit) L บิท โดยจะมีจุดทวินิยม (Binary point) อยู่ระหว่างบิทจำนวนเต็มและบิทเครื่องหมาย ดังแสดงในรูปที่ 2.9



รูปที่ 2.9 แสดงการจัดรูปแบบจำนวนโดยตรงที่ประกอบด้วยบิทจำนวนเต็มและบิทเศษส่วน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จำนวนบิท N เป็นตัวกำหนดย่านพลวัตรที่ต้องการ โดยถ้าเลือกให้มีจำนวนน้อยอาจทำให้เกิดการล้น (Overflow) จากการคำนวณได้ แต่ถ้าเลือกให้มีจำนวนมากความเที่ยงตรงก็จะน้อยลง ซึ่งในการสร้างวงจร โดยการแทนด้วยรูปแบบจำนวนโดยตรงนั้น นิยมที่จะทำมาตราส่วน (Scaling) เพื่อให้ขนาดของสัญญาณมีค่าอยู่ระหว่าง $-1 \le X < 1$ คือมีบิทเครื่องหมาย 1 บิท และบิทเศษส่วน L บิท ดังแสดงในรูปที่ 2.10



รูปที่ 2.10 แสดงการจัดรูปแบบจำนวนโดยตรงที่มีแต่บิทเศษส่วน

โดยทั่วไปเลขฐานสองแบบจำนวนโดยตรงแบ่งออกได้เป็น 3 รูปแบบด้วยกัน คือ (1) แบบขนาดและเครื่องหมาย (Sign magnitude) (2) แบบส่วนเติมเต็มหนึ่ง (1's complement) (3) แบบส่วนเติมเต็มสอง (2's complement) ใน 3 รูปแบบนี้ตัวเลขแบบส่วนเติมเต็มสองเป็นที่นิยมใช้กันมากในระบบการประมวลผลสัญญาณเชิงเลข ทั้งนี้เนื่องมาจาก

1. มีการแทนค่าเลขศูนย์ได้เพียงค่าเดียว
2. การสร้างวงจรฮาร์ดแวร์สำหรับการบวก ลบ และคูณ ของเลขส่วนเติมเต็มสองทำได้ง่าย โดยในการคูณสามารถใช้หลักการเลื่อนและบวก (Shift and add) หรือที่เรียกว่าบูทอัลกอริธึม (Booth's algorithm)ได้
3. ในระหว่างผลการบวกย่อย (Partial sum) ของการบวกเลขส่วนเติมเต็มสอง สามหรือสี่จำนวน ถึงแม้อาจจะเกิดการล้น (ตัวทดจากผลการบวกล้นข้ามไปทับบิทเครื่องหมาย) แต่ผลลัพธ์สุดท้ายมักให้ค่าถูกต้องเสมอ ถ้าผลบวกอยู่ในช่วง –1 ถึง $1-2^{-L}$ ดังตัวอย่าง

7/8	0.111	
+ 4/8	0.100	
11/8	1.011	ผลบวกย่อยที่ผิดเนื่องจากเกิดการล้น
- 6/8	1.010	
5/8	0.101	ผลบวกที่ถูกต้อง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

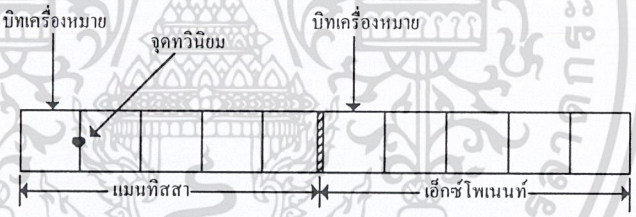
2.2.1.2 รูปแบบจำนวนอิงดรรชนี

รูปแบบจำนวนโดยตรงมีข้อเสียที่สำคัญ 2 ประการ คือ (1) ย่านพลวัตรของตัวเลขมีค่าน้อย เช่น การแทนด้วยเลขส่วนเติมเต็มสอง ค่าที่น้อยที่สุดคือ –1 และค่าที่มากที่สุด คือ $1-2^{-L}$ (2) เปอร์เซ็นต์ความผิดพลาดที่เกิดจากการตัด (Truncation) หรือการปัด (Rounding) จะเพิ่มมากขึ้นเมื่อขนาดของตัวเลขมีค่าลดลง ตัวอย่างเช่น ถ้าจำนวน 0.11011010 และ 0.000110101 ถูกตัดให้จำนวนบิทเศษส่วนเหลือเพียง 4 บิท เปอร์เซ็นต์ความผิดพลาดจะเป็น 4.59 % และ 39.6 % ตามลำดับ โดยข้อเสียนี้สามารถแก้ไขได้โดยการใช้รูปแบบจำนวนอิงดรรชนี ซึ่งตัวเลข X แสดงได้โดย

$$X = M \times 2^{e} \tag{2.2.1}$$

โดย e เป็นจำนวนเต็ม และ $\frac{1}{2} \le |M| < 1$

M และ e เรียกว่า แมนทิสสา (Mantissa) และ เอ็กซ์โพเนนท์ (Exponent) ตามลำดับ ตัวอย่างเช่น จำนวน 0.00110101 และ 01001.11 สามารถแทนได้โดย 0.110101×2^{-2} และ 0.100111×2^{4} ตามลำดับ ส่วนจำนวนที่มีค่าเป็นลบก็ทำในลักษณะเดียวกัน รูปแบบจำนวนอิงดรรชนีสามารถแสดงได้ดังรูปที่ 2.11 โดยแบ่งเป็น 2 ส่วน คือ ส่วนหนึ่งสำหรับแมนทิสสา และอีกส่วนสำหรับเอ็กซ์โพเนนท์



รูปที่ 2.11 แสดงการจัดรูปแบบจำนวนอิงดรรชนี

ข้อดีของการใช้จำนวนอิงดรรชนี คือแทนค่าของสัญญาณได้ละเอียดกว่า และแม่นยำกว่าแบบจำนวนโดยตรง แต่การบวก ลบ หรือคูณจะยุ่งยากกว่ามาก วงจรจึงซับซ้อนและแพงกว่าแบบจำนวนโดยตรงมาก นอกจากนี้ความเร็วในการประมวลผลยังช้ากว่าด้วย ดังนั้นสำหรับการประมวลผลแบบเวลาจริง (Real time) จึงนิยมใช้ระบบตัวเลขแบบจำนวนโดยตรง

2.2.2 ทฤษฎีเลขคณิตกระจาย

จากที่ได้กล่าวมาแล้วว่า โครงสร้างเลขคณิตกระจายมีพื้นฐานอยู่บนการคูณแบบเลขส่วนเติมเต็มสอง ดังนั้นในหัวข้อนี้จะได้อธิบายถึงหลักการของการคูณเลขส่วนเติมเต็มสอง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ให้เลขส่วนเติมเต็มสองของ X ซึ่งแทนด้วย $\overline{X}$ และนิยามโดย

$$\overline{X} = \begin{cases} X & \text{ถ้า } X \geq 0 \\ 2-|X| & \text{ถ้า } X < 0 \end{cases} \tag{2.2.2}$$

โดย X เป็นเลขที่เป็นเศษส่วน (Fractional number)

ในระบบเลขส่วนเติมเต็มสองจะใช้บิทที่มีนัยสำคัญสูงสุด (MSB) เป็นบิทแสดงเครื่องหมาย โดยถ้าเป็นบวกแทนด้วย "0" และถ้าเป็นลบแทนด้วย "1" ถ้าให้ X แทนด้วยเลขฐานสองขนาด L+1 บิท ดังนั้นรูปแบบของเลขส่วนเติมเต็มสองจะเขียนได้ดังนี้

$$\overline{X} = X_0.X_1X_2...X_L \tag{2.2.3}$$

ค่าของ $\overline{X}$ ในรูปของเลขฐานสิบสามารถหาได้ดังนี้

$$X = -X_0 + \sum_{i=1}^{L} X_i\, 2^{-i} \tag{2.2.4}$$

จากนั้นพิจารณาผลคูณต่อไปนี้

$$Y = X\, m \tag{2.2.5}$$

ให้ $\overline{Y}$, $\overline{X}$ และ $\overline{m}$ เป็นเลขส่วนเติมเต็มสองของ Y, X และ m ตามลำดับ จากนั้นพิจารณาจากสมการที่ (2.2.4) และ สมการที่ (2.2.5) จะได้

$$\begin{aligned} Y &= -Y_0 + \sum_{i=1}^{L} Y_i\, 2^{-i} \\ &= -X_0 m + \sum_{i=1}^{L} X_i\, m\, 2^{-i} \end{aligned} \tag{2.2.6}$$

ดังนั้น

$$\begin{aligned} \overline{Y} &= \text{ส่วนเติมเต็มสองของ} \left[-X_0 m + 2^{-1} X_1 m + 2^{-2} X_2 m + 2^{-3} X_3 m + ... + 2^{-L} X_L m \right] \\ &= \text{ส่วนเติมเต็มสองของ} \left[-X_0 m + 2^{-1}(X_1 m + ... + 2^{-1}(X_{L-1} m + 2^{-1}(X_L m))) \right] \end{aligned} \tag{2.2.7}$$

ต่อไปพิจารณาส่วนเติมเต็มสองของ $2^{-1}U$ โดย

$$\overline{U} = U_0.U_1U_2...U_M$$

สำหรับ $U \geq 0$ (หรือ $U_0 = 0$)

$$\text{ส่วนเติมเต็มสองของ } (2^{-1}U) = 2^{-1}\overline{U}$$

และสำหรับ $U < 0$ (หรือ $U_0 = 1$)

$$\text{ส่วนเติมเต็มสองของ } (2^{-1}U) = 2 - \left| 2^{-1}U \right| = 1 + 2^{-1}\left(2 - |U|\right) = 1 + 2^{-1}\overline{U}$$

ดังนั้นสรุปได้ว่า

$$\text{ส่วนเติมเต็มสองของ } (2^{-1}U) = \begin{cases} 2^{-1}\overline{U} & \text{ถ้า } U_0 = 0 \\ 1 + 2^{-1}\overline{U} & \text{ถ้า } U_0 = 1 \end{cases} \tag{2.8}$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สมการที่ (2.2.8) นี้แสดงให้เห็นได้ว่า ส่วนเติมเต็มสองของ ($2^{-1}U$) เป็นการเลื่อนข้อมูลของ $\overline{U}$ ไปทางขวา 1 บิท

$$\therefore \text{ ส่วนเติมเต็มสองของ } (2^{-1}U) = 2_2^{-1}\overline{U} \tag{2.2.9}$$

โดย $2_2^{-1}\overline{U}$ แสดงถึงการเลื่อนข้อมูลของ $\overline{U}$ ไปทางขวา 1 บิท แบบเลขส่วนเติมเต็มสอง ซึ่งสัญลักษณ์ 2_2^{-1} (ซึ่งโดยทั่วไปนิยมเขียนเป็น 2^{-1}) เป็นการแสดงว่าในกรณีที่ $\overline{U}$ เป็นเลขบวก ซึ่งบิทเครื่องหมายจะเป็นเลขศูนย์ โดยหลังจากเลื่อนข้อมูลไปทางขวา 1 บิทแล้ว บิทที่มาแทนบิทเครื่องหมายก็ยังคงเป็นเลขศูนย์ ส่วนในกรณีที่ $\overline{U}$ เป็นเลขลบ หลังจากเลื่อนข้อมูลไปทางขวา 1 บิทแล้ว บิทที่มาแทนบิทเครื่องหมายจะเป็นเลขหนึ่ง (จาก $1+2^{-1}\overline{U}$) ซึ่งในการสร้างเพื่อใช้งานจริงจำเป็นจะต้องมีวงจรที่ใช้ในการตรวจสอบบิทเครื่องหมาย (Sign digit) ทุกครั้งที่มีการเลื่อนข้อมูล จากนั้นพิจารณาสมการที่ (2.2.7) และสมการที่ (2.2.8) จะได้ว่า

$$\begin{aligned}\overline{Y} &= -X_0\overline{m} + 2^{-1}X_1\overline{m} + 2^{-2}X_2\overline{m} + 2^{-3}X_3\overline{m} + \ldots + 2^{-L}X_L\overline{m} \\ &= -X_0\overline{m} + 2^{-1}(X_1\overline{m} + \ldots + 2^{-1}(X_{L-1}\overline{m} + 2^{-1}(X_L\overline{m})))\end{aligned} \tag{2.2.10}$$

ซึ่งจากสมการที่ (2.2.10) จะเห็นได้ว่าผลคูณจากสมการที่ (2.2.5) สามารถหาได้โดยการใช้หลักการเลื่อนและบวก (Shift and add) หรือ บูทอัลกอริธึมนั่นเอง โดยผลลัพธ์ที่ได้จากการคูณแบบเลขส่วนเติมเต็มสอง สามารถหาได้ตามขั้นตอน ดังนี้

1. เคลียร์ค่าข้อมูลในแอคคิวมูเลเตอร์รีจิสเตอร์
2. บวก $X_L\overline{m}$ กับค่าที่อยู่ในแอคคิวมูเลเตอร์รีจิสเตอร์
3. เลื่อนค่าที่อยู่ในแอคคิวมูเลเตอร์รีจิสเตอร์ไปทางขวา 1 บิท
4. ทำซ้ำข้อ 2 และ 3 สำหรับค่า $X_{L-1},\ldots,X_1$
5. ลบค่า $X_0\overline{m}$ ออกจากค่าที่อยู่ในแอคคิวมูเลเตอร์รีจิสเตอร์

ตัวอย่างการทำงานตามอัลกอริธึมนี้

$Y = Xm = 0.8125(-0.390625)$ โดยสมมุติให้ใช้แอคคิวมูเลเตอร์รีจิสเตอร์ขนาด 12 บิท

$m = -0.390625$

$\overline{m} = 2 - |m|$ $\because$ m เป็นเลขลบ

$= 2 - 0.390625$

$= 1.609375$

$\therefore$ $\overline{m} = 1.100111$

$X = 0.8125 = \overline{X}$ $\because$ X เป็นเลขบวก

$\therefore$ $\overline{X} = 0.1101 = X_0.X_1X_2X_3X_4$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โดยมีขั้นตอนการทำงาน ดังตารางต่อไปนี้

ตารางที่ 2.1 แสดงขั้นตอนการคูณเลขส่วนเติมเต็มสอง

การดำเนินการ	ข้อมูลในแอคคิวมูเลเตอร์รีจิสเตอร์
เคลียร์ ACC	0.000 0000 0000
ACC+$X_4\overline{m}$	1.100 1110 0000
เลื่อนข้อมูลใน ACC ไปทางขวา 1 บิท	1.110 0111 0000
ACC+$X_3\overline{m}$	1.110 0111 0000
เลื่อนข้อมูลใน ACC ไปทางขวา 1 บิท	1.111 0011 1000
ACC+$X_2\overline{m}$	1.100 0001 1000
เลื่อนข้อมูลใน ACC ไปทางขวา 1 บิท	1.110 0000 1100
ACC+$X_1\overline{m}$	1.010 1110 1100
เลื่อนข้อมูลใน ACC ไปทางขวา 1 บิท	1.101 0111 0110
ACC-$X_0\overline{m}$	1.101 0111 0110

$$\therefore \quad \overline{Y} = 1.101 \quad 0111 \quad 0110 \quad = Y_0.Y_1Y_2...Y_{11}$$

จะได้

$$\begin{aligned} Y &= -Y_0 + \sum_{i=1}^{11} Y_i 2^{-i} \\ &= -1 + \left[2^{-1} + 2^{-3} + 2^{-5} + 2^{-6} + 2^{-7} + 2^{-9} + 2^{-10} \right] \\ &= -0.3173828125 \end{aligned}$$

จากอัลกอริธึมดังกล่าว สามารถออกแบบการทำงานและสร้างวงจร แสดงได้ดังรูปที่ 2.12

ที่ผ่านมาเป็นหลักการคูณแบบเลขส่วนเติมเต็มสอง ส่วนทฤษฎีเลขคณิตกระจายก็อาศัยหลักการดังกล่าวมาใช้ โดยทำการกระจายสมการที่อยู่ในรูปผลบวกของผลคูณให้แตกออกมาอยู่ในระดับบิท (Bit level) ซึ่งนำเสนอโดย A.Peled และ B.Liu ดังนี้

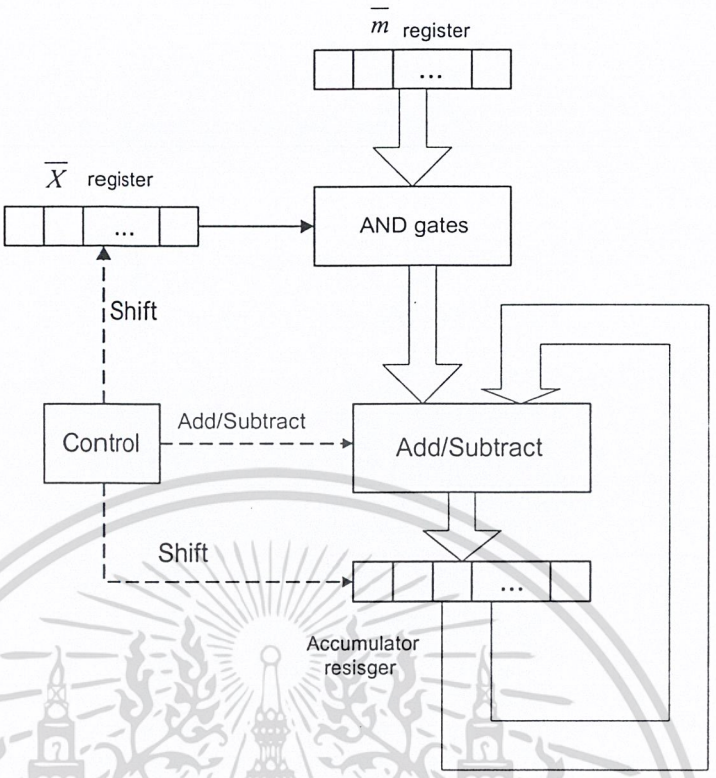
พิจารณาผลบวกของผลคูณต่อไปนี้

$$Y = \sum_{i=0}^{N} m_i X_i \tag{2.2.11}$$

โดย m_i เป็นค่าสัมประสิทธิ์ซึ่งที่ค่าคงที่

X_i เป็นข้อมูลอินพุท

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.12 แสดงการคูณแบบเลขส่วนเติมเต็มสองโดยใช้บูทอัลกอริธึม

ถ้า X_i แต่ละค่าเป็นเลขส่วนเติมเต็มสอง โดย $|X_i| < 1$ สามารถแสดง X_i แต่ละค่าได้ดังนี้

$$X_i = -X_{i0} + \sum_{j=1}^{L} X_{ij} 2^{-j} \qquad (2.2.12)$$

โดย X_{ij} = บิทต่างๆของข้อมูล X_i มีค่าเป็น 0 หรือ 1

X_{i0} = บิทแสดงเครื่องหมาย

X_{iL} = บิทที่มีนัยสำคัญต่ำสุด (LSB)

$L+1$ = จำนวนบิทที่แทนข้อมูลอินพุท

แทนค่า X_i ในสมการที่ (2.2.12) ลงในสมการที่ (2.2.11) จะได้

$$Y = \sum_{i=0}^{N} m_i \left[-X_{i0} + \sum_{j=1}^{L} X_{ij} 2^{-j} \right] \qquad (2.2.13)$$

จัดเทอมของผลบวกใหม่จะได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$Y = -X_{i0}\sum_{i=0}^{N} m_i + \sum_{j=1}^{L} X_{ij}2^{-j}\sum_{i=0}^{N} m_i$$

$$= -\sum_{i=0}^{N} X_{i0}m_i + \sum_{i=0}^{N}\sum_{j=1}^{L} X_{ij}m_i 2^{-j} \qquad (2.2.14)$$

จากนั้นทำการกระจายออกให้เป็นระดับบิท ได้ดังนี้

$$\begin{aligned} Y = &-(X_{00}m_0 + X_{10}m_1 + X_{20}m_2 + ... + X_{N0}m_N) \\ &+2^{-1}(X_{01}m_0 + X_{11}m_1 + X_{21}m_2 + ... + X_{N1}m_N) \\ &+2^{-2}(X_{02}m_0 + X_{12}m_1 + X_{22}m_2 + ... + X_{N2}m_N) \\ &+ ... + 2^{-L}(X_{0L}m_0 + X_{1L}m_1 + X_{2L}m_2 + ... + X_{NL}m_N) \end{aligned} \qquad (2.2.15)$$

สมการที่ (2.2.15) นี้ ถูกกระจายออกให้อยู่ในรูปผลบวกของผลคูณระหว่างสัมประสิทธิ์กับข้อมูลอินพุทในระดับบิท ซึ่งเป็นนิยามของการคำนวณแบบเลขคณิตกระจาย และเมื่อเปรียบเทียบสมการที่ (2.2.15) กับ สมการที่ (2.2.10) จะเห็นว่าการคำนวณหาค่า Y ก็ใช้บูทอัลกอริธึมนั่นเอง เพียงแต่เรานำค่าผลคูณย่อย (Partial product) ที่คำนวณไว้ล่วงหน้าแล้วสำหรับแต่ละค่าที่สอดคล้องกับแต่ละบิทของข้อมูลอินพุทไปเก็บไว้ในตารางเปิดดู ซึ่งเป็นหน่วยความจำ EPROM และใช้ข้อมูลอินพุทเป็นแอดเดรสของหน่วยความจำ เพื่อนำค่าในตารางเปิดดูมาผ่านขั้นตอนการคำนวณตามบูทอัลกอริธึม ซึ่งค่าในตารางเปิดดูสามารถแสดงได้ดังนี้

ตารางที่ 2.2 ค่าผลคูณย่อยที่เก็บไว้ในตารางเปิดดูที่กำหนดโดยข้อมูลอินพุท X_{ij}, $i = 0, 1, ..., N$

Bit pattern ของข้อมูลอินพุท				ผลคูณย่อยที่เก็บไว้ในตารางเปิดดู
X_{Nj}	.....	X_{2j}	X_{1j} X_{0j}	
0	.....	0	0 0	0
0	.....	0	0 1	m_0
0	.....	0	1 0	m_1
0	.....	0	1 1	$m_1 + m_0$
0	.....	1	0 0	m_2
0	.....	1	0 1	$m_2 + m_0$
0	.....	1	1 0	$m_2 + m_1$
0	.....	1	1 1	$m_2 + m_1 + m_0$
⋮				⋮
1	.....	1	1 1	$m_N + m_{N-1} + ... + m_2 + m_1 + m_0$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

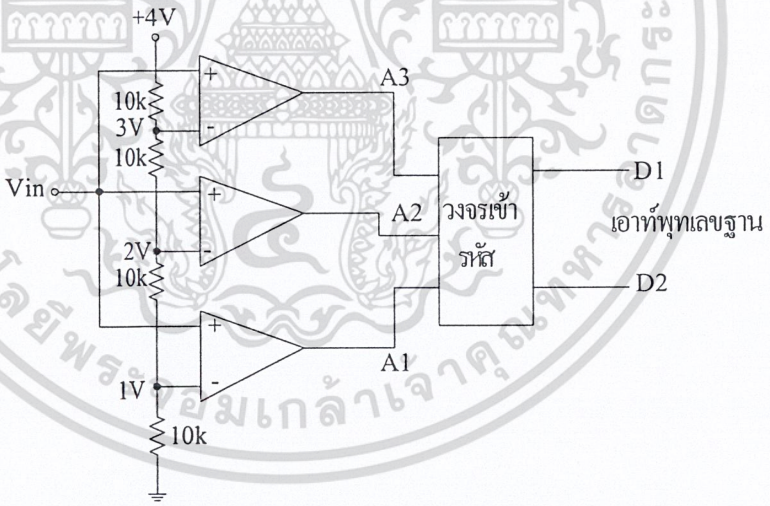
2.3 วงจรแปลงอะนาลอกเป็นดิจิตอล(Analog to Digital Converter: A/D)

วงจรเปลี่ยนสัญญาณอะนาลอกเป็นดิจิตอล ที่ใช่อยู่ทั่วๆไปมีหลายแบบ คือ

2.3.1. แบบใช้วงจรเปรียบเทียบขนาน หรือแบบแฟลซ (Parallel Comparator Simultaneous "Flash" A/D Converter) วงจรเอทูดีแบบนี้ใช้หลักการง่ายๆ อีกทั้งยังเป็นวิธีที่รวดเร็วที่สุด คือ ใช้วงจรเปรียบเทียบที่ต่อขนานกันดังรูปที่ 2.13. ประกอบด้วยออปแอมป์ที่ต่อเป็นวงจรเปรียบเทียบ และตัวต้านทานต่อไว้เพื่อแบ่งแรงดันที่ขาอินพุตแบบกลับ(inverting) ให้มีขนาดต่างๆกัน จากหลักการของวงจรเปรียบเทียบทั่วไปเมื่อแรงดันที่ขาอินพุตแบบไม่กลับ(non-inverting) มีค่าสูงกว่าที่ขาอินพุตแบบกลับเอาท์พุตจะได้แรงดันค่าต่างๆมีผลต่อเอาท์พุต ของวงจรเปรียบเทียบแต่ละตัวอย่างไร ซึ่งเอาต์พุตที่ได้จากวงจรเปรียบเทียบนี้จะนำไปเข้ารหัสให้เป็นเลขฐานสองต่อไป

จำนวนของวงจรเปรียบเทียบที่ต้องใช้ในวงจรขึ้นอยู่กับขนาดของสัญญาณอะนาลอกที่อินพุตจากวงจรรูปที่ 2.13 ถ้าแรงดันอินพุตมีค่า 1 โวลท์ ไม่เพียงพอที่จะทำให้วงจรเปรียบเทียบแปลงให้ค่าเอาท์พุตเป็น "High" ที่แรงดันระหว่าง 1 ถึง 2 โวลท์ วงจรเปรียบเทียบที่มีระดับแรงดันเทรชโฮลด์(threshold)ต่ำสุดให้เอาท์พุตเป็น "High"

แรงดัน 2-3 โวลท์ วงจรเปรียบเทียบทั้ง A1 และ A2 ให้เอาท์พุตเป็น "High" ถ้าแรงดันอินพุตมากกว่า 3 โวลท์ วงจรเปรียบเทียบก็จะเป็น "High" ทั้งหมด



รูปที่ 2.13 แสดงการต่อวงจร Parallel Comparator A/D Converter

เมื่อต้องการวงจรที่มีความละเอียดสูงขึ้นจำเป็นต้องใช้วงจรเปรียบเทียบเพิ่มขึ้น โดยจำนวนวงจรเปรียบเทียบหาได้จาก 2^N-1 เมื่อ N แทนจำนวนบิตหรือความละเอียดที่ต้องการ จะเห็นได้ว่าที่ความละเอียด 8 บิตต้องใช้วงจรเปรียบเทียบมากถึง 255 ตัวซึ่งเป็นข้อเสียของ วงจรดีทูเอแบบนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ข้อเสียอีกประการหนึ่งก็คือ เอาท์พุตที่ได้ไม่เป็นเลขฐานสอง ต้องมีวงจรเพิ่มเติมไปทำงานเข้ารหัส " แฟลช " (Flash Type A/D Converter) วงจรเอทูดีชนิดนี้ ใช้เวลาในการแปลงได้เร็วในระดับนาโนวินาทีทีเดียว

2.3.2 วงจรเอทูดีที่ใช้การอินทีเกรต วงจรเปลี่ยนสัญญาณเอทูดีที่ใช้ เทคนิคการอินทีเกรตสัญญาณมี 4 แบบ คือ

- แบบสโลบเดี่ยวหรือแบบแรมป์ (Single Ramp หรือ Single Slope A/D Converter)

เมื่อเริ่มทำการเปลี่ยนสัญญาณ สัญญาณแรมป์และวงจรนับจะถูกรีเซตเป็น "0 " แรงดันอะนาลอกถูกป้อนไปยังวงจรเปรียบเทียบทางขาอินพุตแบบไม่กลับ เมื่อแรงดันอินพุตที่ขานี้เป็นบวกมากกว่าที่ขาอินพุทแบบกลับ วงจรเปรียบเทียบก็จะให้เอาท์พุตเป็นระดับ "High" ทำให้แอนด์เกตปล่อยสัญญาณนาฬิกาผ่านไปยังวงจรนับได้ และทำให้เริ่มเกิดสัญญาณแรมป์

สัญญาณแรมป์มีแรงดันเป็นบวกขึ้นเรื่อยๆ จนมากกว่าระดับแรงดันอินพุตและเอาท์พุทของ วงจรเปรียบเทียบก็ตกลงมาเป็นระดับ "Low" ปิดแอนด์เกต ไม่มีสัญญาณผ่านไปให้วงจรนับ

วงจรนับจะหยุดนับและเก็บค่าไว้ที่วงจรแลทช์ จากนั้นจึงทำการรีเซต วงจรนับและวงจรกำเนิดสัญญาณแรมป์

วงจรแบบนี้เป็นหลักการเบื้องต้นของดิจิตอลโวลต์มิเตอร์ซึ่งถ้าใช้วงจรนับเลขฐานสองแทนแบบ BCD เอาต์พุตก็จะอ่านค่าได้เลขฐานสองโดยตรง

วงจรลักษณะนี้ มักนำไปใช้งานในการเปลี่ยนเวลาเป็นขนาดของสัญญาณ (Time to Amplitude Conversion) หรืออาจใช้ในดิจิตอลโวลท์มิเตอร์ แต่ไม่ใช้กับงานที่ต้องการความถูกต้องสูง เนื่องจากการเปลี่ยนแปลงในแหล่งกำเนิดสัญญาณแรมป์ขึ้นกับอุณหภูมิ และผลตอบสนองต่อสัญญาณอินพุต ทำให้ไม่มีความคงที่ ดังนั้นจึงมีการปรับปรุงให้ดีขึ้นกลายเป็นแบบสโลบคู่ (Dual Slope)

- แบบสโลบคู่ (Dual – Slope A/D Converters)

ส่วนแรก คือ วงจรกำเนิดสัญญาณแรมป์ หรือวงจรอินทีเกรเตอร์นั่นเอง ที่อินพุตแบบกลับของออปแอมป์มีสภาพเป็นกราวด์เทียม (Virtual ground) ถ้ามีแรงดันอินพุต 2 โวลท์ จะได้กระแสไหลผ่านตัวต้านทาน 10 กิโลโอห์ม เท่ากับ 0.2 มิลลิแอมป์ไปยังจุดร่วม (summing point) เนื่องจากค่าความต้านทานอินพุตของออปแอมป์นั้นสูงมาก กระแสที่ไหลจึงเกิดขึ้นผ่านตัวเก็บประจุ ได้เอาต์พุตออกมาเป็นแรมป์ทางลบ

วงจรเปรียบเทียบก็จะได้แรงดันลบจากวงจรอินทีเกรเตอร์ แล้วให้เอาต์พุตเป็นบวกทำการเปิดสัญญาณแอนด์เกตให้สัญญาณนาฬิกาผ่านเข้าสู่วงจรนับ วงจรนับจะทำการนับไปยังค่าที่กำหนดไว้คงที่แล้วทำการสับสวิตช์ต่อเข้ากันกับแรงดันอ้างอิง

สรุปได้ว่า แรงดันอินพุต ที่ไม่รู้ค่าถูกป้อนเข้าไปในวงจรอินทีเกรเตอร์เมื่อครบช่วงเวลา t_1 วงจรนับจึงถูกรีเซตให้เป็น 0 อินพุตของวงจรอินทีเกรเตอร์ก็จะถูกสวิตช์ต่อกลับมาที่แรงดันอ้างอิง (ที่มีแรงดันคงที่) ให้ความชันของสัญญาณแรมป์คงที่เพิ่มค่าขึ้นไปจนถึงระดับ 0 ช่วงเวลา t_2 นี้เป็นสัดส่วนโดยตรงกับสัญญาณอินพุต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ข้อดีของวงจรเปลี่ยนสัญญาณแบบสโลบคู่นี้คือ ความถูกต้องสูง , ราคาถูก , เสถียรภาพทางด้านอุณหภูมิดี ข้อเสียคือความเร็วต่ำ ในการเปลี่ยนสัญญาณ 1 ครั้งอาจจะใช้เวลาถึง 100 มิลลิวินาที (ในขณะที่แบบแฟลซใช้เวลาประมาณ 30 มิลลิวินาที)

- แบบชาร์จบาลานซ์ (Charge Balance A/D Converters)

วงจรเปลี่ยนสัญญาณเอทูดีแบบชาร์จบาลานซ์ ใช้วงจรสำคัญคล้ายๆแบบสโลบคู่นั่นเอง แต่แทนที่จะให้อินพุตสวิตซ์ ไปมาระหว่างแรงดันที่ไม่รู้ค่ากับแรงดันอ้างอิง ก็ทำการแทรกพัลส์ของกระแสอ้างอิงโดยตรงที่จุดรวม (Summing point) ของวงจรอินทีเกรเตอร์ในช่วงเวลาที่คงที่ โดยที่จำนวนของพัลส์จะเป็นสัดส่วนโดยตรงกับแรงดันอินพุตที่ไม่รู้ค่า

ประโยชน์ของเทคนิคนี้คือ แรงดันตกคร่อมตัวเก็บประจุของวงจรอินทีเกรเตอร์ จะมีค่าใกล้เคียง 0 โวลท์ ดังนั้นจึงไม่เกิดความผิดพลาดจากผลของกระแสรั่วไหล

เอทูดีชนิดนี้จึงมีความถูกต้องสูงกว่าแบบสโลบคู่

- แบบเดลต้า-ซิกม่า (Delta – Sigma A/D Converters)

เมื่อมีแรงดันอินพุตป้อนเข้าไปที่วงจรอินทีเกรเตอร์จะให้เอาต์พุตเข้าไปในวงจรเปรียบเทียบเปรียบเทียบกับแรงดันคงที่ พัลส์ของกระแสที่ได้จะขึ้นอยู่กับเอาต์พุตของวงจรเปรียบเทียบ โดยสวิตช์ที่ทำขึ้นจากเฟสจะควบคุมกระแสให้เข้าไปยังที่จุดรวมหรือลงกราวด์ไป ส่วนวงจรจะนับวงจรพัลส์ด้วยหลักการที่คล้ายกัน

ข้อสรุปของเอทูดีแบบอินทีเกรตสัญญาณ

จุดสำคัญของอินทีเกรติ้งเทคนิคนี้ คืออินพุตที่ให้กับวงจรอินทีเกรเตอร์ต้องเป็นกระแสไอซีคอนเวอร์เตอร์บางตัวอาจมีอินพุตให้สองขา แต่จะมีขาหนึ่งต่อตรงกับจุด Summing point ใช้กับอุปกรณ์ที่เป็นแหล่งจ่ายกระแสโดยตรง

ถ้าให้อินพุตเป็นกระแสก็ไม่ต้องคำนึงถึงแรงดันออฟเซตของวงจรอินทีเกรเตอร์ แต่ถ้าหากใช้กับอินพุตที่เป็นแรงดันต้องปรับออฟเซตของออปแอมป์เสียก่อน

การใช้อินพุตเป็นกระแสทำให้ย่านการใช้งานทางไฟสลับกว้าง

2.3.3 วงจรเปลี่ยนสัญญาณเอทูดีที่ใช้วงจรนับและวงจรดีทูเอประกอบกัน

2.3.3.1 แบบบวงจรนับเดี่ยว (Single Counter)

แท้ที่จริงแล้วสัญญาณแรมป์เชิงเส้น (Linear ramp) อาจประกอบขึ้นด้วยสัญญาณขั้นบันไดเล็กๆ จำนวนมากที่เกิดจากการต่อเอาต์พุทของวงจรนับเข้ากับวงจรแปลงดีทูเอ โดยขนาดของขั้นบันไดแต่ละขั้นขึ้นอยู่กับจำนวนบิตหรือความละเอียดของวงจรดีทูเอนั้นๆ

การกำเนิดสัญญาณแรมป์เดี่ยวด้วยวงจรนับ และวงจรดีทูเอ เมื่อเริ่มแปลงสัญญาณวงจรนับจะถูกรีเซต เอาต์พุทของวงจรดีทูเอ มีระดับ 0 เมื่อแรงดันถูกป้อนเข้าไปยังอินพุทของวงจรเปรียบเทียบ เอาต์พุทก็จะขึ้นสู่ระดับ “High” และเปิดสัญญาณนาฬิกาไปสู่วงจรนับ แต่ละพัลส์ของสัญญาณนาฬิกา ทำให้เกิดการนับและเพิ่มแรงดันขึ้น 1 ขั้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เมื่อเอาท์พุตของดีทูเอมีค่ามากกว่าอินพุต Vin เอาท์พุตของวงจรเปรียบเทียบก็จะกลายเป็น "Low" ทำให้สัญญาณนาฬิกาไม่อาจผ่านไปยังวงจรนับได้ ดังนั้นวงจรควบคุมจะทำการแลทช์เอาต์พุตของวงจรนับและรีเซตวงจรนับให้เริ่มต้นรอบใหม่อีกครั้งหนึ่ง

การทำงานจะคล้ายกับแบบใช้วงจรนับเดี่ยว แต่การนับจะไม่ได้เริ่มจากศูนย์ แต่จะทำการนับขึ้นหรือ นับลงจากค่าล่าสุดไปยังค่าใหม่ แล้วแต่ว่าแรงดันอินพุตในรอบใหม่มีค่าสูงกว่าหรือต่ำกว่าค่าที่แล้ว ข้อดีของดีทูเอแบบแทร็กกิ้ง คือ ทำงานได้เร็วขึ้น

2.3.4 วงจรเปลี่ยนสัญญาณเอทูดีแบบใช้การประมาณค่า (Successive Approximation A/D Converter) วงจรเอทูดีแบบนี้มีข้อได้เปรียบทางด้านความละเอียด เพาะความละเอียด n บิต สามารถกำหนดได้จากสัญญาณนาฬิกา n ลูก ตัวอย่างเช่น วงจรแปลงขนาด 8 บิต ต้องการพลัซ์ของสัญญาณนาฬิกาเพียง 8 ลูก ในขณะที่แบบใช้วงจรนับต้องใช้พลัซ์ถึง 256 ลูก วงจรแปลงเอทูดีชนิดนี้มีความเร็วสูงและความละเอียดสูง จึงเป็นวงจรที่นำมาใช้อย่างแพร่หลาย

2.3.5 การสุ่มและการคงค่า (Sample and Holds)

วงจรเอทูดีต้องการเวลาในการนับในการแปลงสัญญาณหรือ เรียกว่า Conversion Time เพื่อเปลี่ยนสัญญาณอะนาล็อกเป็นสัญญาณดิจิตอลที่เหมาะสม ถ้าสัญญาณอะนาล็อกมีการเปลี่ยนแปลงในระหว่างช่วงเวลาการแปลง เอาท์พุทของวงจรเปลี่ยนสัญญาณอาจเกิดความผิดพลาดจึงต้องมีการป้องกันด้วยวงจรสุ่มและคงค่าสัญญาณ (Sample and Holds) เพื่อใช้จับสัญญาณอะนาล็อกที่จุดเริ่มของการเปลี่ยนสัญญาณแล้วเก็บไว้ในตัวเก็บประจุระหว่างช่วงการแปลง หลังจากทีมีการเปลี่ยนสัญญาณเสร็จสิ้น จึงจับสัญญาณอะนาล็อกค่าใหม่มาเก็บประจะระหว่างช่วงเวลาการแปลง หลังจากที่การแปลงสัญญาณเสร็จสิ้นจึงสัญญาณอะนาล็อกค่าใหม่มาเก็บไว้อีกครั้งเป็นเช่นต่อๆ ไป เราจึงมักได้พบเห็นวงจรสุ่มและคงค่าสัญญาณปรากฏคู่กับวงจรเอทูดีอยู่เสมอ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เมื่อเอาท์พุตของดีทูเอมีค่ามากกว่าอินพุต Vin เอาท์พุตของวงจรเปรียบเทียบก็จะกลายเป็น "Low" ทำให้สัญญาณนาฬิกาไม่อาจผ่านไปยังวงจรนับได้ ดังนั้นวงจรควบคุมจะทำการแลทช์เอาต์พุตของวงจรนับและรีเซตวงจรนับให้เริ่มต้นรอบใหม่อีกครั้งหนึ่ง

การทำงานจะคล้ายกับแบบใช้วงจรนับเดี่ยว แต่การนับจะไม่ได้เริ่มจากศูนย์ แต่จะทำการนับขึ้นหรือ นับลงจากค่าล่าสุดไปยังค่าใหม่ แล้วแต่ว่าแรงดันอินพุตในรอบใหม่มีค่าสูงกว่าหรือต่ำกว่าค่าที่แล้ว ข้อดีของดีทูเอแบบแทร็กกิ้ง คือ ทำงานได้เร็วขึ้น

2.3.4 วงจรเปลี่ยนสัญญาณเอทูดีแบบใช้การประมาณค่า (Successive Approximation A/D Converter) วงจรเอทูดีแบบนี้มีข้อได้เปรียบทางด้านความละเอียด เพราะความละเอียด n บิต สามารถกำหนดได้จากสัญญาณนาฬิกา n ลูก ตัวอย่างเช่น วงจรแปลงขนาด 8 บิต ต้องการพัลซ์ของสัญญาณนาฬิกาเพียง 8 ลูก ในขณะที่แบบใช้วงจรนับต้องใช้พัลซ์ถึง 256 ลูก วงจรแปลงเอทูดีชนิดนี้มีความเร็วสูงและความละเอียดสูง จึงเป็นวงจรที่นำมาใช้อย่างแพร่หลาย

2.3.5 การสุ่มและการคงค่า (Sample and Holds)

วงจรเอทูดีต้องการเวลาในการนับในการแปลงสัญญาณหรือ เรียกว่า Conversion Time เพื่อเปลี่ยนสัญญาณอะนาล็อกเป็นสัญญาณดิจิตอลที่เหมาะสม ถ้าสัญญาณอะนาล็อกมีการเปลี่ยนแปลงในระหว่างช่วงเวลาการแปลง เอาท์พุทของวงจรเปลี่ยนสัญญาณอาจเกิดความผิดพลาดจึงต้องมีการป้องกันด้วยวงจรสุ่มและคงค่าสัญญาณ (Sample and Holds) เพื่อใช้จับสัญญาณอะนาล็อกที่จุดเริ่มของการเปลี่ยนสัญญาณแล้วเก็บไว้ในตัวเก็บประจุระหว่างช่วงการแปลง หลังจากที่มีการเปลี่ยนสัญญาณเสร็จสิ้น จึงจับสัญญาณอะนาล็อกค่าใหม่มาเก็บประจะระหว่างช่วงเวลาการแปลง หลังจากที่การแปลงสัญญาณเสร็จสิ้นจึงสัญญาณอะนาล็อกค่าใหม่มาเก็บไว้อีกครั้งเป็นเช่นต่อๆไป เราจึงมักได้พบเห็นวงจรสุ่มและคงค่าสัญญาณปรากฏคู่กับวงจรเอทูดีอยู่เสมอ

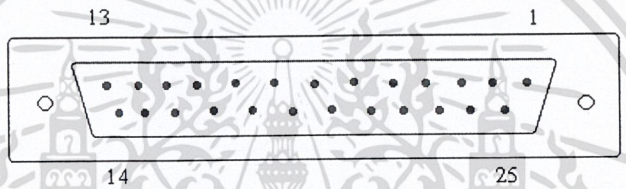
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.4 พอร์ทเครื่องพิมพ์แบบขนาน

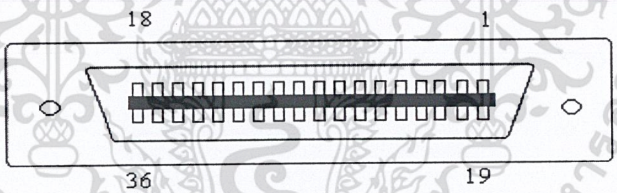
พอร์ทเครื่องพิมพ์แบบขนานบนคอมพิวเตอร์เป็นอุปกรณ์มาตรฐานที่ใช้ต่อกับเครื่องพิมพ์ที่อยู่ภายนอก ในปัจจุบันมีแนวโน้มที่บริษัทผู้ผลิตอุปกรณ์เสริมให้แก่คอมพิวเตอร์ในตระกูลดังกล่าวแล้วจะใช้พอร์ทนี้เชื่อมต่อระหว่างอุปกรณ์ที่ผลิตขึ้นอยู่กับคอมพิวเตอร์เช่น กล้องถ่ายรูปดิจิตอล เครื่องอ่าน CD-ROM เครื่องบันทึกจากแม่เหล็กแบบพกพา และ อุปกรณ์ที่ใช้งานในห้องปฏิบัติการต่างๆ อีกมากมาย

2.4.1 สายสัญญาณและหัวต่อ

สายสัญญาณเชื่อมโยงระหว่างพอร์ทเครื่องพิมพ์แบบขนานจากคอมพิวเตอร์ไปยังอุปกรณ์ภายนอก ด้านพอร์ทเครื่องพิมพ์แบบขนานจะเป็นหัวต่อแบบ D-TYPE 25 และ ด้าน เครื่องพิมพ์หรืออุปกรณ์ภายนอกจะเป็นหัวต่อแบบ Centronics



หัวต่อ D-TYPE



หัวต่อ D-TYPE

รูปที่ 2.14 ลักษณะของหัวต่อพอร์ทขนาน (D-TYPE)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การเชื่อมต่อของสายสัญญาณระหว่างหัวต่อทั้งสองชนิดแสดงไว้ในตารางที่ 2.3

ตารางที่ 2.3 แสดงการเชื่อมโยงระหว่างหัวต่อ D-TYPE 25 และ Centronics

Pin No (D-Type 25)	Pin No (D-Type 25)	SPP Signal	Direction In/Out	Register	Hardware Inverted
1	1	NStrobe	In/Out	Control	Yes
2	2	Data0	Out	Data	
3	3	Data1	Out	Data	
4	4	Data2	Out	Data	
5	5	Data3	Out	Data	
6	6	Data4	Out	Data	
7	7	Data5	Out	Data	
8	8	Data6	Out	Data	
9	9	Data7	Out	Data	
10	10	NAck	In	Status	
11	11	Busy	In	Status	Yes

ตารางที่ 2.3 แสดงการเชื่อมโยงระหว่างหัวต่อ D-TYPE 25 และ Centronics (ต่อ)

Pin No (D-Type 25)	Pin No (D-Type 25)	SPP Signal	Direction In/Out	Register	Hardware Inverted
12	12	Paper-Out /Paper-End	In	Status	
13	13	Select	In	Status	
14	14	nAuto-Linefeed	In/Out	Control	Yes
15	32	Nerror-nFault	In	Status	
16	31	Nlnitaialize	In/Out	Control	
17	36	Nselect – Printer/nSelect-In	In/Out	Control	Yes
18-28	19-30	Ground	Gnd		

หมายเหตุ n หมายถึงสัญญาณที่ส่งเป็นแบบ 1' complement

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.4.2. สัญญาณจาก พอร์ทเครื่องพิมพ์

พอร์ทเครื่องพิมพ์จากเครื่องพิมพ์ IBM PC/XT/AT โดยทั่วไปจะมีพอร์ท คือ LPT1 LPT2 และ LPT3 โดยมีรายละเอียดดังนี้

ตารางที่ 2.4 ตารางแสดงค่าแอดเดรสของการอินพอร์ทและเอาท์พุท

Port เครื่องพิมพ์ (LPIX)	พอร์ทข้อมูล (Base Address)	พอร์ทสถานะ INPUT	พอร์ทควบคุม INPUT/OUTPUT
LPL1	03BCH	03BDH	03BEH
LPL2	0378H	0379H	037AH
LPL3	0278H	0279H	027AH

ตารางที่ 2.5 ตารางแสดงรายละเอียดของข้อมูลแต่ละบิท BASE ADDRESS

	D7	D6	D5	D4	D3	D2	D1	D0
	Data	Data	Data	Data	Data	Data	Data	Data
DB	9	8	7	6	5	4	4	2
Cent	9	8	7	6	5	4	4	2

ตารางที่ 2.6 ตารางแสดงรายละเอียดของข้อมูลแต่ละบิท BASE+1 ADDRESS

	In Busy	In ACK	In Paper End	In Select	In Error
DB	11	10	12	13	15
Cent	11	10	12	13	15

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

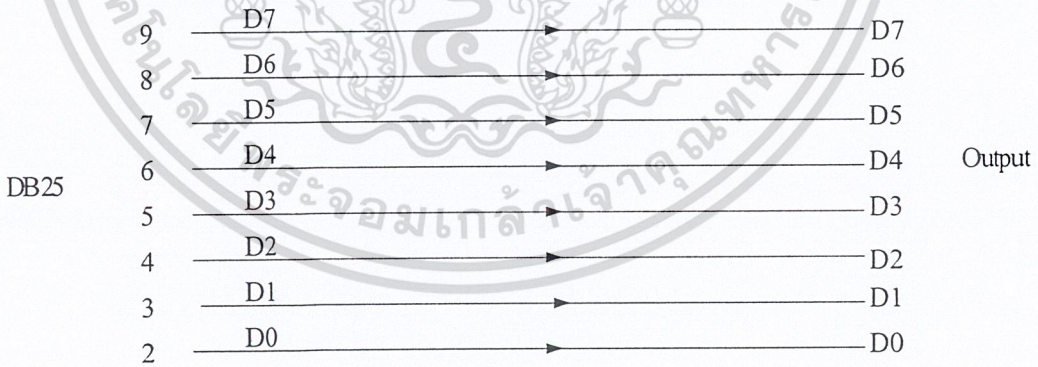
ตารางที่ 2.7 ตารางแสดงรายละเอียดของข้อมูลแต่ละบิท BASE+2 ADDRESS

	Internal Control IRO Enable	In/Out Select Input	In/Out Initialize	In/Out Auto-Feed	In/Out Strobe
DB		17	16	14	1
Cent		36	31	14	1

- DB หมายถึงหัวต่อ DB25, CENT หมายถึงหัวต่อ CENTRONICS
- สัญญาณ SELECT , INPUT, AUTO-FEED และ STROBE เป็นเอาท์พุทแบบ OPEN COLLECTOR
- IRQ ENABLE = บิทควบคุมให้ enable หรือ diable interrupt ของ D6 ที่ BASE H
- ขา Ground สัญญาณ
  - หัวต่อ DB25 อยู่ที่ขา 18-25
  - หัวต่อ Centronics อยู่ที่ขา 16,33 และ 19-30

2.4.3 การนำสัญญาณจากพอร์ทเครื่องพิมพ์มาใช้งาน

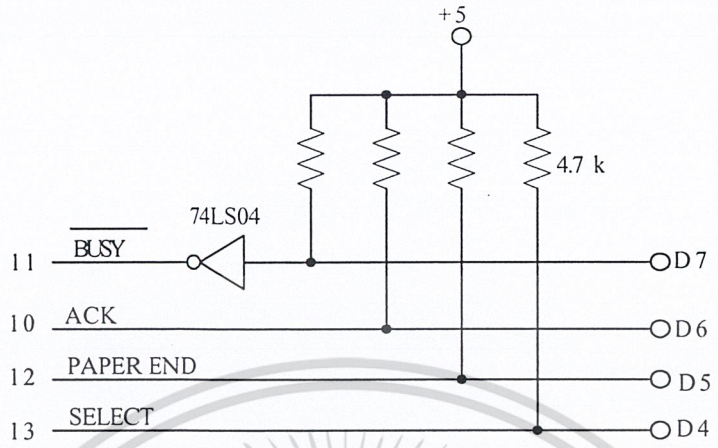
2.4.3.1 เอาท์พุทแบบ 8 บิท นำมาใช้ได้ดังต่อไปนี้



รูปที่ 2.15 สัญญาณเอาท์พุทแบบ 8 บิท ใช้จาก PORT BASE ADDRESS

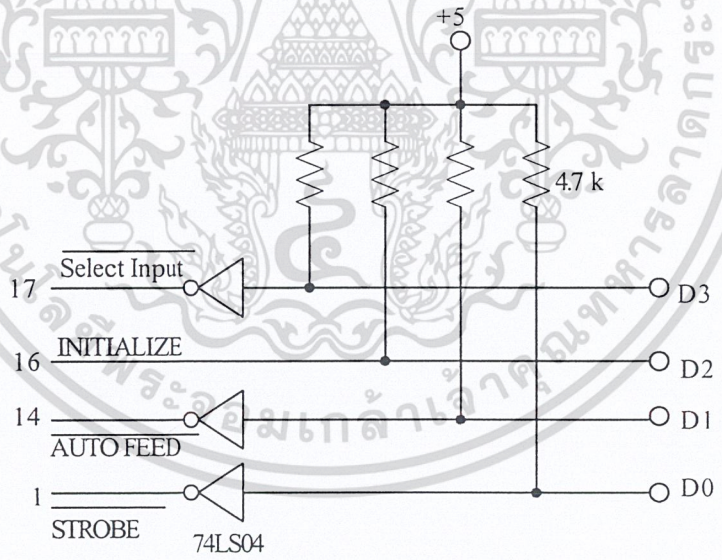
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.4.3.2 สัญญาณอินพุทแบบ 4 บิทบน



รูปที่ 2.16 สัญญาณเอาท์พุทแบบ 4 บิท ใช้จาก PORT BASE ADDRESS+1

2.4.3.3 สัญญาณอินพุทแบบ 4 บิทล่าง



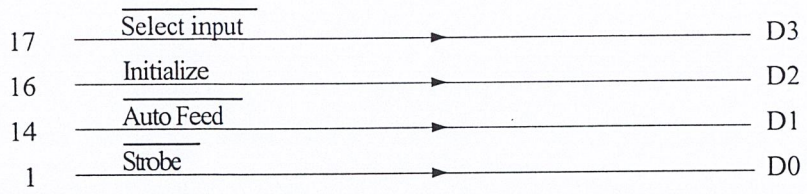
รูปที่ 2.17 การรับสัญญาณอินพุท 4 บิทล่าง จาก PORT BASE ADDRESS+2

อย่างไรก็ตามหากต้องการรับสัญญาณอินพุทแบบ 8 บิท ก็สามารถทำได้โดยการอ่านค่า 4 บิทล่าง จากข้อ 2.4.3.3 และ 4 บิทบนจากข้อ 2.4.3.2 จากนั้นนำมาต่อกันแบบ 8 บิท

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.4.3.4 การส่งสัญญาณเอาท์พุทจาก 4 บิทล่าง

โดยส่งค่าออกจาก PORT BASE ADDRESS+2 ดังนี้

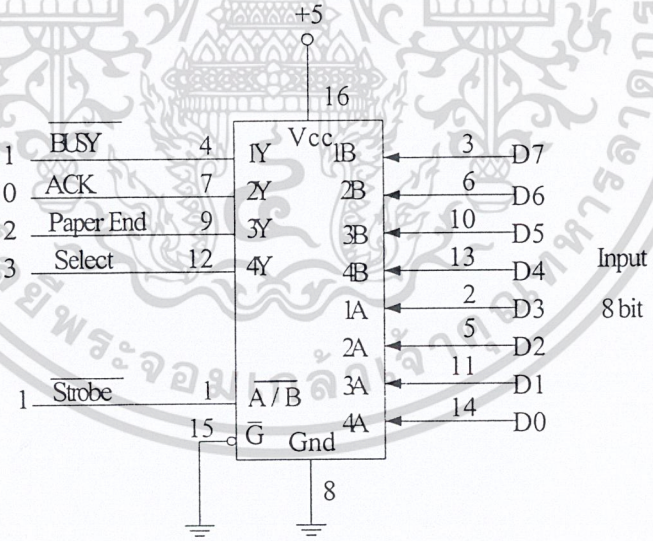


รูปที่ 2.18 รูปการต่อใช้งานของ D0-D3

การนำเอาท์พุทไปใช้งาน ผู้ใช้จะต้องจัดวงจรรองรับด้าน D0-D3 ให้เหมาะกับการนำไปใช้เพราะเอาท์พุทที่ส่งนี้เป็นเอาท์แบบ Open Collector

2.4.3.5 การรับส่งสัญญาณอินพุทแบบ 8 บิท โดยใช้ 74LS157/74LS257 multiplexes

ไอซีดังกล่าวเป็นตัวเลือกสัญญาณแบบ 2 เลือก 1



รูปที่ 2.19 แสดงการใช้งานของ มัลติเพล็กซ์

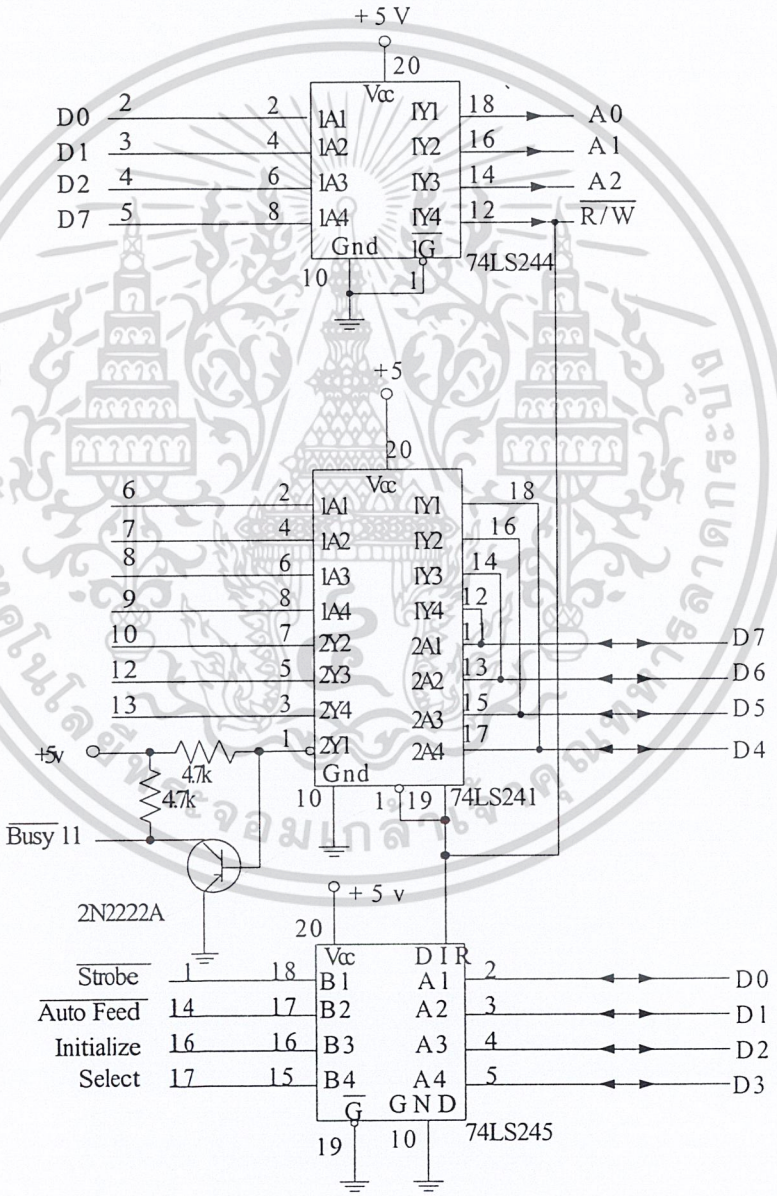
เมื่อคำสั่ง "0" มาที่ขา A/B สัญญาณ 4 บิทล่างจะถูกส่งผ่านไปยังคอมพิวเตอร์ผ่าย PORT BASE ADDRESS +1 โดย DATA จะอยู่ที่ D7, D6, D5 และ D4 ผู้ใช้จะต้องเขียนโปรแกรม Shift Nibble ดังกล่าวแล้วให้กลับมาอยู่ที่ 4 บิทล่าง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากนั้นสั่ง "1" มายังขา A/B สัญญาณ 4 บิทบน จะถูกส่งไปยังคอมพิวเตอร์ในตำแหน่ง 4 บิทบน จะเห็นว่าในการอ่านข้อมูล 8 บิท จะต้องอ่าน 2 ครั้ง ๆ ละ 4 บิท ซึ่งผู้ใช้จะต้องนำข้อทั้งสองครั้งมาต่อกันเป็น 8 บิท

2.4.4. การใช้พอร์ทเครื่องพิมพ์เป็นพอร์ทสองทิศทาง

พอร์ทเครื่องพิมพ์บนเครื่อง PC/XT/AT เป็นช่องทางที่ง่ายที่สุดที่ผู้ใช้จะติดต่อกับอุปกรณ์พอร์ทภายนอกได้ ในงานที่จำเป็นต้องใช้พอร์ทแบบ 2 ทิศทาง



รูปที่ 2.20 แสดงการใช้พอร์ทเครื่องพิมพ์เป็นพอร์ทแบบ 2 ทิศทาง ทรานซิสเตอร์ 2N2222A เป็นทรานซิสเตอร์ชนิดสวิทชิ่ง ทำหน้าที่เป็น Not Gate

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

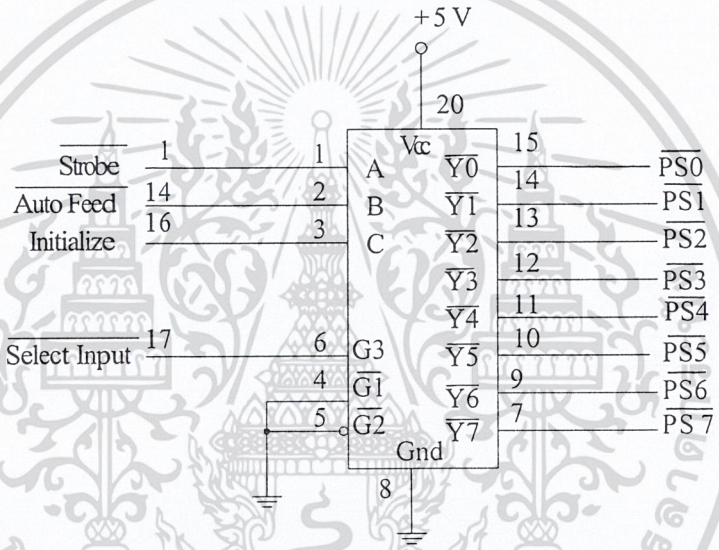
2.4.5. การใช้พอร์ทเครื่องพิมพ์เป็นพอร์ทอินพุท/เอาท์พุท มากกว่าหนึ่งพอร์ท

ในกรณีที่ต้องการใช้เครื่องพิมพ์เป็นพอร์ทแต่ต้องการความสามารถมากกว่าพอร์ทเครื่องพิมพ์จะดำเนินการได้ดังนี้ กำหนดความต้องการ เช่น

- ต้องการพอร์ทเอาท์พุท แบบ 8 บิท 2 พอร์ท (ใช้ 74LS 374)
- ต้องการพอร์ทอินพุท แบบ 8 บิท 2 พอร์ท (ใช้ 74LS 257)

จากคุณลักษณะของสัญญาณจากพอร์ทเครื่องพิมพ์จากข้อ 4 นำมาออกแบบในส่วนต่างๆ ดังนี้

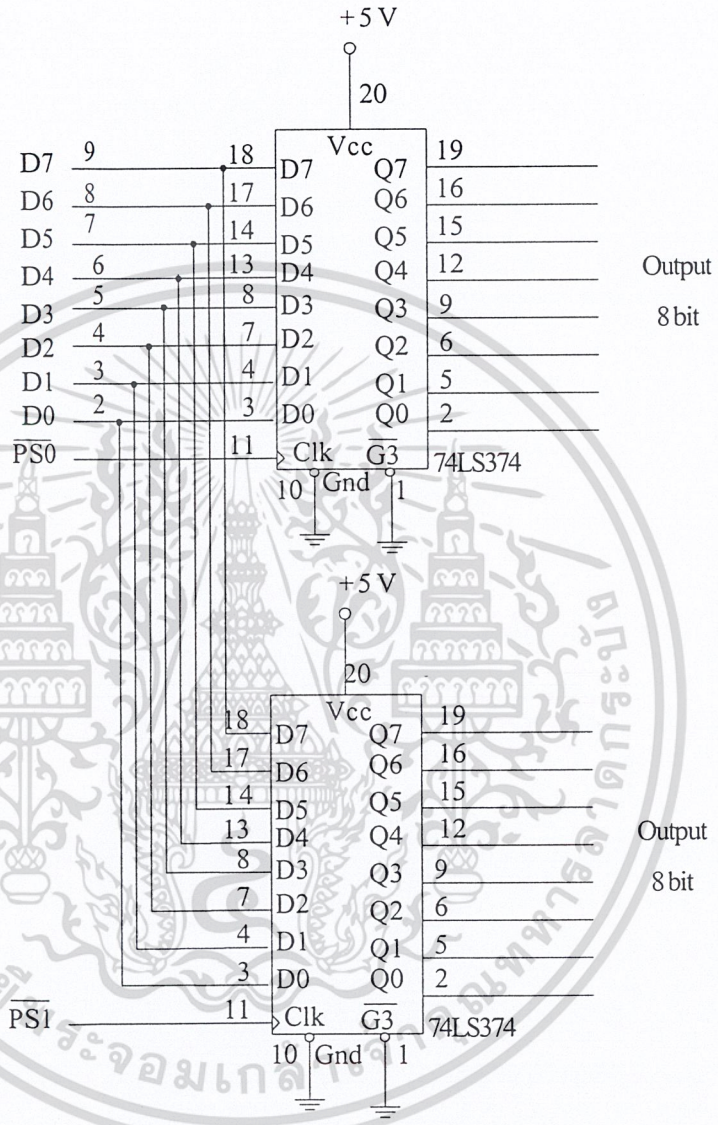
1. ตัว อีนาเบิลพอร์ท ใช้ 74LS138 โดยมีสัญญาณจากแอดเดรส BASE+2 เป็นตัวถอดรหัส PS0-PS7 เป็นสัญญาณเลือก หรือ อีนาเบิลพอร์ทตามการเลือกของสัญญาณจาก PORT BASE+2 ในที่นี้จะเห็นว่าสามารถเลือกพอร์ทได้ถึง 8 พอร์ท



รูปที่ 2.21 แสดงการใช้งานของพอร์ทขนาน 8 บิต โดยใช้ ไอซี เบอร์ 74LS138

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

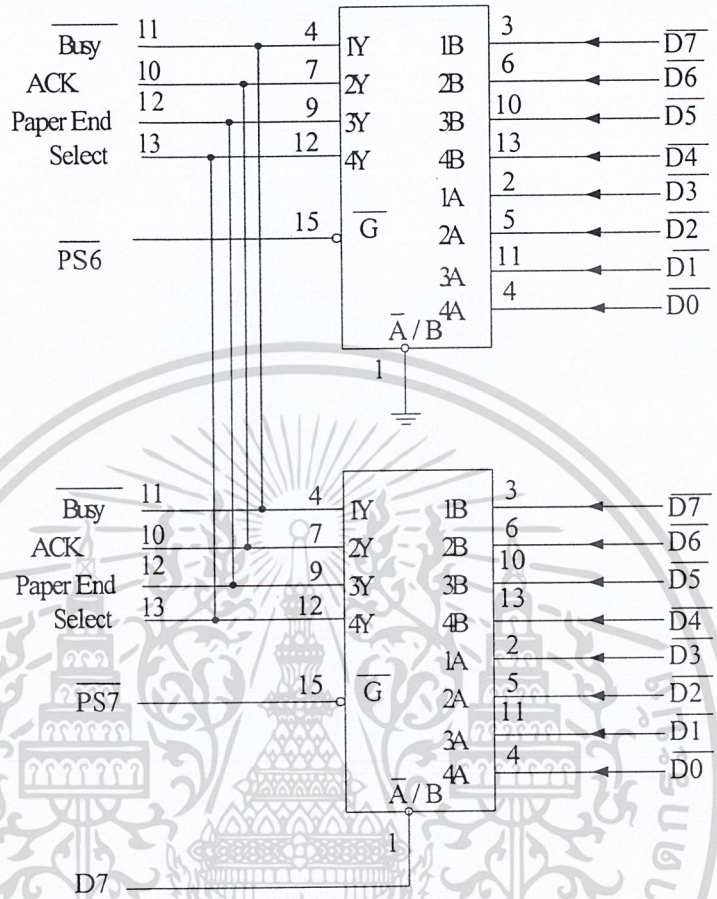
2. ใช้ไอซีแลทซ์เป็นตัวสร้างพอร์ทเอาท์พุท ในที่นี้จะใช้ 74LS374 เป็นพอร์ท 8 บิท โดยมีสัญญาณจากแอดเดรส BASE เป็น Data และใช้สัญญาณอีนาเบิลพอร์ทตามข้อ 1



รูปที่ 2.22 แสดงการใช้งานของพอร์ทขนาน 16 บิต โดยใช้ ไอซี เบอร์ 74LS374

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3. ใช้ 74LS257/74LS157 อ่านค่า 8 บิท โดยอ่านเข้ามาครั้งละ 4 บิท ดังแสดงไว้ในข้อ 4.5



รูปที่ 2.23 การอ่านค่า 8 บิต โดยใช้ไอซี 74LS257 / 74LS157

PS6 และ PS7 ต่อกับเอาท์พุทของ 74LS138 ซึ่งเป็นตัวถอดรหัสในข้อ 1 การอ่านค่าใช้วิธีส่งค่า D7 เลือกชุดของข้อมูล 4 บิทล่าง หรือ 4 บิทบน ถ้าเป็นข้อมูลเป็น 0 เลือก 4 บิทล่าง และถ้าข้อมูลเป็น 1 เลือก 4 บิทบน อ่านค่าเข้าทาง BASE+2 แล้วนำค่าต่อกันเป็น 8 บิท

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.5 ทฤษฎีของ VHDL

2.5.1 ประวัติความเป็นมาของภาษา VHDL

วิวัฒนาการของภาษา VHDL นั้นเริ่มต้นประมาณปี ค.ศ. 1981 โดยที่กระทรวงกลาโหมสหรัฐอเมริกาหรือ Department of Defense (DOD) มองเห็นว่าอุปกรณ์อิเล็กทรอนิกส์และคอมพิวเตอร์ที่ใช้ในกิจการทางทหาร เป็นอุปกรณ์ที่ได้รับการพัฒนามาเมื่อประมาณ 20 ปีก่อนเพราะเทคโนโลยีในขณะนั้นทำให้การพัฒนาอุปกรณ์อิเล็กทรอนิกส์เป็นไปอย่าล่าช้า ซึ่งเป็นสภาพที่ไม่อาจยอมรับได้ในปัจจุบัน เพราะเทคโนโลยีทางด้านไมโครอิเล็กทรอนิกส์ ได้รับการพัฒนาไปอย่างรวดเร็ว ดังที่จะเห็นได้ว่ามีวงจรอิเล็กทรอนิกส์หลายวงจร ที่แต่เดิมถูกสร้างขึ้นมาจากชิ้นส่วนอุปกรณ์อิเล็กทรอนิกส์จำนวนหลายชิ้น ถูกนำประกอบกันอยู่บนแผงวงจรไฟฟ้า ที่มีขนาดใหญ่แต่ในปัจจุบันสามารถที่จะใช้เทคโนโลยีการออกแบบและผลิตวงจรรวมขนาดใหญ่มาก (Very Large Scale Integration หรือ VLSI) รวมอุปกรณ์ต่างๆเหล่านั้นให้อยู่บนชิ้นอุปกรณ์สารกึ่งตัวนำ ที่มีขนาดประมาณ 1-2 ตร.ซม. ได้ ซึ่งเป็นผลให้ประสิทธิภาพในการทำงานของวงจรสูงขึ้น (ความเร็วในการทำงานของวงจร) ตลอดจนความน่าเชื่อถือ และความคงทนต่อสภาพแวดล้อมสูง ทาง DOD จึงตั้งโครงการขึ้นมาเพื่อศึกษา วิธีการที่จะช่วยพัฒนาวงจรอิเล็กทรอนิกส์ โดยเฉพาะอย่างยิ่งวงจรระบบดิจิตอล ให้สามารถนำผลิตได้เร็วขึ้น และโครงการดังกล่าวมีชื่อว่า Very High Speed Integrated Circuits หรือ VHSIC ในระยะแรกโครงการเป็นความลับด้านความมั่นคงของประเทศทางDODได้ออกความต้องการมาตรฐานของภาษาที่ใช้สำหรับบรรยายพฤติกรรมของวงจรหรือ Hardware ของระบบสำหรับโครงการ VHSIC ซึ่งมีสาระสำคัญพอสรุปได้ดังนี้

- ต้องเป็นภาษาที่นำไปเขียนรูปแบบระบบดิจิตอล และมีคุณสมบัติที่สามารถจะเข้าใจได้ทั้งคนและเครื่องโดยไม่ต้องมีการแปลหรือเปลี่ยนแปลงอีก
- สามารถนำไปใช้เป็นเอกสารประกอบโครงการได้ (Project Documentation)
- ต้องเป็นภาษาที่เขียนขึ้นสำหรับใช้จำลองการทำงานของวงจร (Simulation Language)

Top-Down Design

ภาษา VHDL ใช้กำหนดพฤติกรรมฟังก์ชันการทำงานของ Hardware ในระบบดิจิตอลนั้น คือความอ่อนตัวของภาษาที่สามารถจำลองการทำงานจากหลักการของรูปแบบ (Simulate Conceptual designs) แต่ในขณะเดียวกันก็สามารถจำลองการทำงานของ Hardware ที่ให้รายละเอียดเกี่ยวกับเวลาอย่างถูกต้อง (timing based simulation) และจากโครงสร้างของภาษายังสามารถจำลองการทำงานในรูปของลำดับชั้น (hierarchy of simulation levels) ความสามารถดังกล่าวนี้จึงช่วยให้วิศวกรออกแบบ สามารถที่จะเขียนรูปแบบบรรยายจากระดับบนสุดของวงจรที่อยู่ในรูปสังเขป (high level of abstraction) ลงสู่รายละเอียดในระดับล่างของวงจรได้เช่น Gate level เป็นต้น ในช่วงเวลานั้นเองวงการอุสาหกรรมไมโครอิเล็กทรอนิกส์ ตลอดจนสถาบันวิจัยและศึกษา กำลังพัฒนาภาษาที่จะใช้สำหรับการสังเคราะห์วงจรแบบอัตโนมัติ เพื่อลดเวลาในการพัฒนาวงจรลง ภาษา VHDL จึงถูกนำเข้าพิจารณาในโครงการนี้ด้วย โดยเพิ่มขีดความสามาราถของภาษาขึ้นอีกประการหนึ่ง นอกเหนือจากสิ่งที่ทาง DOD กำหนดในครั้งแรกคือเป็นภาษาที่ใช้สำหรับสังเคราะห์วงจร (Synthesis language)

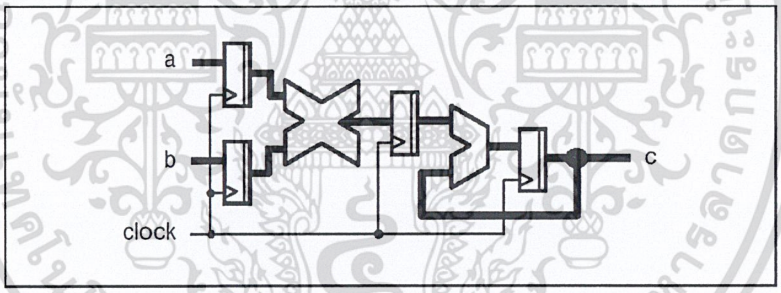
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาษา VHDL เป็นภาษาที่สนับสนุนการเขียนรูปแบบในทุกๆ ลักษณะและวิธีการ ดังเช่นตัวอย่างที่ แสดงในรูปที่ 2.24 คือรูปแบบ (model) ของลำดับขั้นตอนของการคูณและหาผลรวม (multiply accumulate algorithm) ของตัวแปรสองตัว a และ b ส่วนผลลัพธ์คือ c ในลักษณะของเลขฐานสอง (binary number) แบบนี้แสดงในระดับที่สังเขปที่สุดของแนวความคิดที่จะแก้ปัญหา เพื่อหาผลลัพธ์ โดยไม่ได้คำนึงถึงโครงสร้างของวงจรอย่างที่เคยชิน เช่นอุปกรณ์วงจรรวม Arithmetic and Logic Unit (ALU)

```
FOR i IN 1 TO 1024 LOOP
        result := result + a(i) * b(i);
END LOOP;
c <= result;
```

รูปที่ 2.24 VHDL Statement สำหรับ Multiply Accumulate Algorithm

ในขณะที่อีกด้านหนึ่งของมุมมองในปัญหาเดียวกันนี้ ภาษา VHDL ก็สามารถใช้บรรยายลำดับขั้นตอนของการคูณและหาผลรวม (multiply accumulate algorithm) โดยแสดงได้ในรายละเอียดของการสร้างวงจรดังกล่าวจริงๆ ตามที่เห็นได้จากรูปที่ 2.25



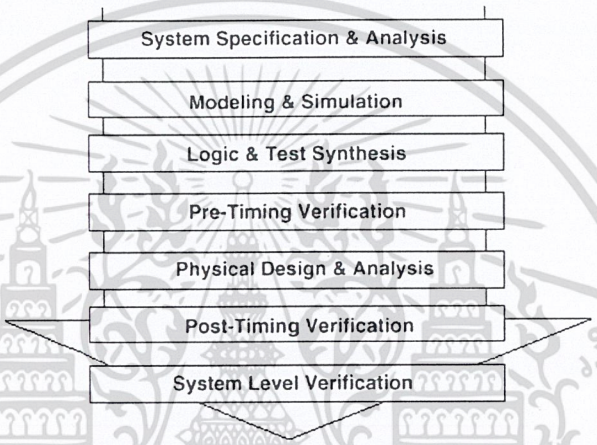
รูปที่ 2.25 โครงสร้างของ Multiply-Accumulate Unit

ฉะนั้นจากความสามารถที่จะเขียนรูปแบบ (Modeling) ได้ในลักษณะต่างๆ นี้เองจึงเปิดโอกาสให้วิศวกรผู้ออกแบบได้พัฒนาและจำลองการทำงานของรูปแบบได้เร็ว ตั้งแต่ในระยะเริ่มต้นของแนวความคิดเกี่ยวกับฟังก์ชั่นการทำงานของวงจรอย่างสังเขป โดยที่ยังไม่ต้องไปคำนึงถึงรายละเอียดเกี่ยวกับโครงสร้างวงจรจริง นอกจากนั้น VHDL ยังเป็นภาษาที่สนับสนุนลักษณะต่างๆของระบบดิจิตอล (digital system) ที่มีความซับซ้อนได้ทั้งหมด

การเริ่มต้นด้วยวิธีการเขียนรูปแบบจากแนวความคิดอย่างสังเขป พร้อมทั้งการจำลองการทำงานรูปแบบที่เขียนขึ้น เพื่อตรวจสอบความถูกต้อง ประกอบกับการกลั่นกรองเพิ่มเติมรายละเอียดลงสู่ระบบดิจิตอลที่สมบูรณ์ในรูปของวงจรไฟฟ้าทีละขั้น นั้นเป็นขบวนการของ Top-Down Design การที่เริ่มต้นด้วยการเขียนรูปแบบ (modeling) ในระดับบน (top-level) ของแนวความคิดอย่างสังเขปนั้น วิศวกรออกแบบสามารถที่จะพัฒนาสภาพแวดล้อมต่างๆ เพื่อการตรวจสอบการทำงานของวงจร (test environment)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ได้ตั้งแต่ในระยะแรกๆ ของการออกแบบ เพื่อใช้สำหรับตรวจสอบความถูกต้องของแนวความคิดกับสิ่งที่ต้องการจริงหรือ specification ของงานดังนั้นจึงเป็นไปได้ยากที่ในระดับล่างลงมา โดยที่ได้เพิ่มรายละเอียดของรูปแบบให้มากขึ้นตามลำดับจะเกิดข้อผิดพลาด หรือเบี่ยงเบนไปจากจุดประสงค์เดิม เพราะในแต่ละขั้นตอนย่อยจะมีการสร้างสภาพแวดล้อมเพื่อการตรวจสอบขึ้นใหม่ โดยอ้างอิงจากระดับที่อยู่สูงกว่าขึ้นไปเสมอ จากรูปที่ 2.26 แสดงให้เห็นขั้นตอนของการออกแบบในลักษณะของ Top-Down Design ทั้งนี้ในทางปฏิบัติอาจจะมีข้อแตกต่างไปจากนี้บ้างเล็กน้อย ก็เนื่องมาจากขั้นตอนของการผลิต (implantation) สามารถกระทำได้ในหลายๆ เทคโนโลยี เช่น Programmable Logic Devices อันได้แก่ PLA , FPGA หรือ CPLD เป็น นอกจากนั้นยังมี Semi-Custom IC (Gate Array , Standard Cell) และ Full Custom IC



รูปที่ 2.26 ขั้นตอนของ Top-Down Design

- Physical Design and Analysis คือ ขั้นตอนของการผลิตเป็นวงจร (technology and device mapping) โดยนำข้อมูลที่ได้จากการสังเคราะห์มาผลิต ซึ่งอาจจะอยู่ในรูปของแผงวงจรไฟฟ้า (Printed Circuit Board: PCB) ที่ประกอบด้วยอุปกรณ์หลายๆชิ้น หรืออยู่ในรูปของวงจรรวมเฉพาะงาน (ASIC)

- Post-Timing Verification หลังจากที่ได้วงจรจริงมาแล้ว ยังต้องมีความจำเป็นที่จะต้องตรวจสอบการทำงานที่คำนึงเวลาด้วย เพื่อความถูกต้องของวงจรครั้งสุดท้ายก่อนที่จะนำไปรวมเข้ากับอุปกรณ์อื่นๆ ให้เป็นระบบดิจิตอลเพราะในขั้นตอนนี้วงจรที่ออกแบบจะประกอบด้วย Input และ output pad ซึ่งเป็นจุดต่อสำหรับรับและส่งสัญญาณกับภายนอก

- System Level Verification หลังจากที่นำวงจรที่ออกแบบรวมเข้าอุปกรณ์อื่นๆ ให้เป็นระบบดิจิตอลแล้วนั้น จะต้องทดสอบการทำงานรวมทั้งระบบร่วมกับอุปกรณ์อื่นๆ อีกครั้ง เป็นการควบคุมคุณภาพของผลิตภัณฑ์

จากความอ่อนตัวของภาษา และความสามารถที่จะเขียนรูปแบบได้หลายลักษณะนี้เอง VHDL จึงเป็นเครื่องมือที่ใช้สำหรับออกแบบตั้งแต่ขั้นตอนบนสุด คือแนวความคิดที่จะแก้ปัญหาลงไปที่ละขั้นจนถึงขั้นตอนของการผลิตวงจรจริง (form idea to implementation) ข้อดีที่เห็นได้ชัดของการนำ VHDL มาใช้ในการออกแบบลักษณะ top-down นี้คือ วิศวกรออกแบบสามารถที่จะสร้างรูปแบบ (model) และจำลองการทำงาน เพื่อตรวจสอบความถูกต้องกับข้อกำหนด (specification) ตั้งแต่เริ่มแรกที่มีแนวความคิดอย่าง

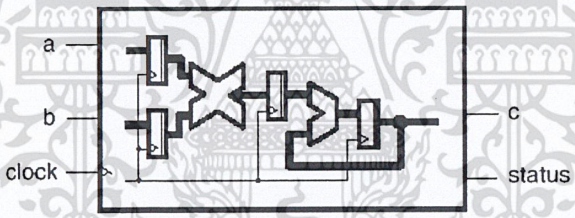
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สังเขป จากการจำลองการทำงานในระยะต้นๆของการออกแบบนั้น หลักการต่างๆ ที่ถูกกำหนดขึ้นใช้ในการแก้ปัญหา (สร้างวงจรให้เป็นตามความต้องการของข้อกำหนด) จะถูกตรวจสอบด้วยทุกครั้ง ก่อนที่จะมีการลงทุนในขั้นตอนสุดท้ายของการออกแบบ หรือการสร้างวงจรนั้นเอง นั้นหมายความว่าข้อผิดพลาดที่อาจจะเกิดขึ้นได้จากหลักการที่กำหนดขึ้น จะถูกตรวจพบและจัดการแก้ไขให้ถูกต้องได้ ก่อนที่จะทำงานในขั้นตอนต่อๆไปของขบวนการออกแบบ

ในตัวอย่างของ Multiply accumulate algorithm สามารถแสดงให้เห็นขบวนการออกแบบในลักษณะของ Top-Down Design ได้ โดยการใช้ multiply accumulate function จุดประสงค์แรกก็คือ การเขียนรูปแบบของฟังก์ชั่นในระดับบนสุดด้วยภาษา VHDL และจำลองการทำงานของรูปแบบที่ได้ เพื่อตรวจสอบความถูกต้อง ซึ่งฟังก์ชันนี้อาจจะเป็น Discrete Fourier Transform (DFT) ก็ได้หลังจากที่ได้ฟังก์ชันที่ต้องการแล้ว (เช่นรูปที่ 2.24 และมีชื่อว่า " multiply_accum") สามารถนำไปใช้ได้ในรูปของ function call คือ

```
c <= multiply_accum(a,b);
```

ถ้าฟังก์ชั่นที่ได้ทำงานเป็นที่พอใจแล้ว ในขั้นตอนต่อไปจะเป็นการเริ่มต้นของการแปลงความคิดไปสู่การสร้างวงจรจริง ขบวนการดังกล่าวเกิดขึ้นโดยการ เพิ่มรายละเอียดให้กับแนวความคิดเดิม รูปที่ 2.22 แสดงให้เห็นวิธีการบรรยายแบบ Dataflow (Dataflow description) ของ multiply accumulate function

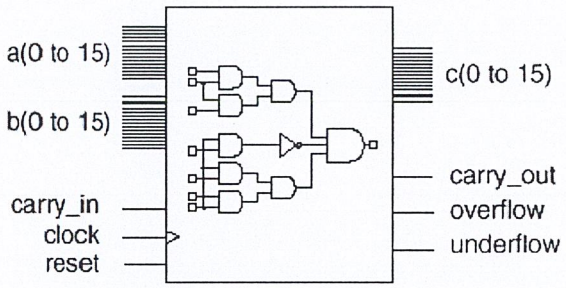


รูปที่ 2.27 Dataflow Description ของ Multiply Accumulate Function

จาก Dataflow format ที่ได้นี้ ขบวนการต่อไปคือการสร้างวงจรให้อยู่ในรูปของอุปกรณ์พื้นฐาน หรือ gate-level implementation ซึ่งอาจใช้วิธีการสังเคราะห์อัตโนมัติ ผลลัพธ์ที่ได้จากการสังเคราะห์จะเป็นการแสดงภาพวงจรด้วย gate ของฟังก์ชันต่างๆหรือในรูปของ netlist และจะเป็นการกำหนดเทคโนโลยีจำเพาะที่จะนำไปผลิตในภายหลัง ในรูปที่ 2.28 แสดงให้เห็นรูปร่างของวงจรที่กำหนดจากการสังเคราะห์ในระดับ gate-level

คุณสมบัติหลักของภาษา VHDL คือความสามารถที่จะใช้บรรยาย hardware ได้ในทุกๆระดับของภาพรวมทั้งระบบ ฉะนั้นวิศวกรออกแบบจึงสามารถใช้เครื่องมือ (ภาษา) เพียงอันเดียวในการบรรยายทั้งระบบ ซึ่งก็เช่นเดียวกันกับเครื่องมือจำลองการทำงาน (simulator)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.28 Gate-level Representation ของ Multiply-Accumulate Function

Terminology and Conventions

การเขียนรูปแบบของระบบดิจิตอลด้วยภาษา VHDL นั้น จะมีศัพท์เทคนิคเฉพาะ ฉะนั้น ในบทนี้จะเป็นการบรรยาย และอธิบายศัพท์บางคำที่ต้องใช้งานบ่อยๆ

ลักษณะของรูปแบบ(model Styles): ลักษณะของการเขียนรูปแบบ (model) ด้วยภาษา VHDL สามารถแบ่งออกได้เป็น

- Behavioral Model: หรือเรียกอีกอย่างหนึ่งว่า **Algorithmic description** เป็นรูปแบบที่บรรยายพฤติกรรมของระบบดิจิตอล ในส่วนที่บรรยายมีโครงสร้างคล้ายกับภาษาชั้นสูง (high level language) ทั่วๆไป เช่น Pascal หรือ C เป็นต้น ในการจำลองการทำงาน (simulation) คำสั่งแต่ละคำสั่ง (statement) จะถูกประเมินผลเป็นไปตามลำดับ (sequential) จากบนลงล่าง ยกเว้นในกรณีของคำสั่ง Loop หรือการเรียกใช้โปรแกรมย่อย รูปแบบลักษณะนี้จะไม่ให้รายละเอียดที่เกี่ยวกับการผลิต หรือโครงสร้างของ hardware แต่ในทางตรงข้ามที่จะให้รายละเอียดที่เกี่ยวกับความสัมพันธ์ระหว่าง input กับ output ที่ดี

- Dataflow Model: เรียกอีกอย่างหนึ่งว่า " Register Transfer Level "(RTL) เป็น รูปแบบที่ถูกเขียนขึ้น เพื่อจุดประสงค์ที่จะใช้เครื่องมือสำหรับสังเคราะห์วงจรอัตโนมัติ รูปแบบลักษณะนี้ส่วนใหญ่จะเป็น procedural constructs และ functional operators

- Structural Model เป็นรูปแบบที่แสดงการเชื่อมต่อกันระหว่างอุปกรณ์ต่างๆ ที่ประกอบกันขึ้นเป็นวงจรหรือระบบดิจิตอล และสามารถเรียกอีกอย่างหนึ่งว่า "netlist representation" เป็นการเขียนที่แสดงให้เห็นโครงสร้างของ Hardware

- Mixed-Level Model จากคุณสมบัติที่อ่อนตัวของภาษา VHDL จึงสามารถที่จะเขียนรูปแบบโดยใช้ลักษณะต่างๆ ที่กล่าวมาข้างต้น บรรยายวงจรหรือระบบดิจิตอลเดียวกัน ฉะนั้นรูปแบบเช่นนี้จึงมีการเขียนแบบผสม

Concurrency: ในภาษา VHDL นั้น ชุดคำสั่ง (statements) แต่ละชุดจะทำงานในเวลาเดียวกันและอิสระต่อกัน ลักษณะเช่นนี้เป็นคุณสมบัติที่เป็นความจริงทางฟิสิกส์ของวงจรอิเล็กทรอนิกส์ ชุดคำสั่งนี้เรียกว่า " **concurrent statement** " และจะทำงานก็ต่อเมื่อมีการเปลี่ยนแปลงค่าของสัญญาณ

Sequential: นอกจากความสามารถที่ชุดคำสั่งจะทำงานแบบ concurrent แล้วบางครั้งการเขียนรูปแบบในลักษณะที่บรรยายพฤติกรรมของวงจร มีความจำเป็นที่จะต้องให้ชุดคำสั่งทำงานเป็นลำดับขั้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เรียงกันจากบน ลงสู่ล่าง อย่างเช่นการเขียนแบบ Behavioral Model เป็นต้น ชุดคำสั่งที่เป็น Sequential นี้จะใช้ในโปรแกรมย่อย (Subprogram) และ process statement ซึ่งจะกล่าวถึงต่อไปในภายหลัง

Driver: สัญญาณต่างๆ (Signal) ใน VHDL นั้นจะถูกควบคุมด้วยตัวขับหรือ "driver" สัญญาณเหล่านี้จะรับค่าใหม่ (ระดับของสัญญาณ) ได้ด้วยตัวขับนี้เอง

Transaction: การเกิด transaction กับ signal นั้นจะเกิดขึ้นเมื่อมีการกำหนดค่าๆ หนึ่งให้กับ signal นั้น ค่าใหม่ที่ signal ได้รับอาจจะมีผลหรือไม่มีผลทำให้เกิดการเปลี่ยนแปลงของระดับสัญญาณ (event) เช่นการเปลี่ยนจากค่า Logic '0' เป็นค่า Logic '1' เป็นต้น

Event: คือการเปลี่ยนระดับค่าของ Signal จากระดับหนึ่งไปสู่ระดับอื่น อย่างเช่นในระบบดิจิตอลมีการเปลี่ยนจาก Logic '0' เป็นค่า Logic '1' หรือในทางตรงกันข้ามถือว่า signal นั้นเกิด "event" ได้นั้นจะต้องเกิด transaction ด้วย แต่ในทางตรงกันข้ามการเกิด transaction ไม่จำเป็นต้องเกิด event ทุกครั้ง

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

Objects: ในภาษา VHDL นั้นคำว่า objects ใช้เขียนเพื่อบ่งบอกถึงองค์ประกอบส่วนหนึ่งของรูปแบบ ซึ่งเปรียบได้เหมือนกับภาชนะที่มีไว้บรรจุค่าต่างๆ สามารถแบ่งออกได้เป็นสามชั้น (class) ด้วยกัน คือ

- **Constant:** ได้ Object ประเภทหนึ่งที่เมื่อกำหนดค่าเริ่มต้นให้แล้ว จะคงค่านั้นไว้ตลอด ไม่สามารถดัดแปลง หรือแก้ไขได้ สามารถประกาศใช้ได้ในส่วนที่เป็นส่วนประกาศต่างๆ ของรูปแบบ (model)

- **Signal:** หมายถึง Object ประเภทหนึ่งที่สามารถกำหนดค่าที่สัมพันธ์กับเวลาให้ได้นั้น หมายความว่า signal สามารถรับค่าได้เพียงค่าเดียวเท่านั้นในขณะเวลาหนึ่ง signal จะรับค่าๆ หนึ่งได้จากตัวขับสัญญาณหรือ driver ซึ่งตัวขับนี้อาจจะเก็บค่าในอนาคตสำหรับ signal ไว้ด้วย signal สามารถประกาศใช้ได้ในส่วนที่เป็นเนื้อที่ของ concurrent body เท่านั้น ดังนั้น signal จึงสามารถถูกนำไปใช้ได้ตลอดโครงสร้างของรูปแบบ(model) หรือที่เรียกว่า global object

- **Variable:** หรือตัวแปรได้แก่ object ที่สามารถกำหนดค่าใดๆ ให้ได้และสามารถที่จะเปลี่ยนแปลงค่าได้ตลอดการจำลองการทำงาน แต่จะเก็บค่าเพียงค่าเดียวเท่านั้นในขณะเวลาหนึ่ง เนื่องจาก Variable สามารถประกาศใช้ได้ในส่วนที่เป็น sequential body เท่านั้นอันได้แก่ส่วนประกาศของ Process, Function หรือ Procedure ดังนั้น Variable จึงสามารถนำไปใช้ได้เฉพาะในขอบเขตที่ถูกประกาศใช้เท่านั้น.

การประกาศใช้ Object (object declaration): การที่จะใช้ Object ชั้นต่างๆ ตามที่กล่าวมาแล้วในการเขียนรูปแบบด้วยภาษา VHDL นั้นจะต้องมีการประกาศใช้ก่อน การประกาศที่ Object สามารถใช้ชุดคำสั่งตามโครงสร้างดังนี้

```
object_class identifier : TYPE [:= initial_value];
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ซึ่ง object_class ได้แก่ Constant , signal หรือ Variable การตั้งชื่อ (identifier) เป็นไปตามกฎของภาษา VHDL ซึ่งจะกล่าวต่อไป Type คือการกำหนดประเภทของ object ที่ประกาศนั้นๆ นอกจากนั้นยังสามารถกำหนดค่าเริ่มต้นของ object ได้ (initial_value) ซึ่งส่วนนี้เป็นเพียง option และการประกาศจะต้องอยู่ในพื้นที่ที่กำหนดให้ของแต่ละส่วนของรูปแบบ (declarative area)

การตั้งชื่อ Object: การตั้งชื่อจะต้องไปตามกฎต่อไปนี้

1. ชื่อ (identifier)ประกอบด้วยตัวหนังสือ (พยัญชนะและตัวเลข) ในภาษาอังกฤษได้แก่
   - พยัญชนะ A-Z,a-z
   - ตัวเลข 0,1,2,3,4,5,6,7,8,9
   - เครื่องหมายขีดเส้นใต้ (Underscore) "_"
2. ชื่อจะต้องขึ้นต้นด้วยพยัญชนะเสมอ
3. ชื่อสามารถประกอบด้วยพยัญชนะตัวเลขและ เครื่องหมายขีดเส้นใต้จำนวนไม่จำกัด
4. การใช้เครื่องหมายขีดเส้นใต้ (_) ทุกครั้ง จะต้องนำหน้าด้วยพยัญชนะหรือตัวเลขและตามหลังด้วยพยัญชนะหรือตัวเลขเช่นกัน
5. พยัญชนะตัวใหญ่หรือตัวเล็กไม่มีความแตกต่างกัน (Case insensitive)
6. ห้ามใช้คำสงวน (Reserved word) ของภาษา VHDL

ค่าเริ่มต้น (Initial value): การประกาศใช้ object ทุกครั้งควรจะต้องกำหนดค่าเริ่มต้นให้ด้วย เพราะค่าที่กำหนดนี้จะถูกนำไปใช้ เมื่อเริ่มต้นจำลองการทำงาน (simulation) ของรูปแบบ

```
SIGNAL example_signal : BIT := '1';
```

ในกรณีที่ไม่มีการกำหนดค่าเริ่มต้น ระบบจำลองการทำงานจะนำค่าที่น้อยที่สุด (อยู่ทางซ้ายมือสุดของกลุ่มค่า) ของแต่ละประเภท (Type) มาเป็นค่าเริ่มต้นแทน

ประเภทข้อมูล (Data type): ได้แก่ type ของ object ที่จะเป็นตัวกำหนดว่าค่าใดบ้างในกลุ่มของค่า (set of type) ของแต่ละ type สามารถที่จะกำหนดให้กับ object ได้นอกจากนั้น type ยังเป็นตัวกำหนดการทำงานในลักษณะต่างๆ (operation) ของ object นั้นๆ type แบ่งออกเป็น 4 ประเภทคือ

1. Scalar ได้แก่ตัวเลข (numeric) ซึ่งในภาษา VHDL มีตัวเลขจำนวนเต็มบวก (Integer) เช่น 1.0,30.,1E2 เป็นต้น
2. Enumeration กลุ่มของค่าประเภทนี้ได้แก่ ตัวหนังสือ หรือชื่อต่างๆ
3. Physical ได้แก่หน่วยทางฟิสิกส์ในระบบ SI
4. File เป็น type ภายนอกสามารถมีค่าได้หลายๆอย่าง

ประเภทย่อยของข้อมูล(subtype): สำหรับ Type ที่กำหนดไว้แล้ว (Predefined type และ user-defined type) สามารถที่จะแบ่งออกเป็นกลุ่มย่อยลงไปได้อีก โดยที่องค์ประกอบของ Type ใหม่นี้จะเป็นส่วนหนึ่งของ type เดิม หรือที่เรียกว่า Subtype

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
TYPE qit IS ('0', '1', 'Z', 'X');
SUBTYPE tit IS qit RANGE '0' TO 'Z';
```

ประเภทของ Object ที่กำหนดไว้แล้ว (Predefined type): ได้แก่ type ที่กำหนดไว้ใน package ชื่อ STANDARD และกำหนดโดย IEEE ว่าจะต้องมีในระบบที่ใช้พัฒนา VHDL ฉะนั้นจึงไม่จำเป็นต้องประกาศใช้ในทุกๆรูปแบบที่เขียนขึ้น type ประเภทนี้ได้แก่

1. Boolean คือ กลุ่มของค่า False และTrue
2. Bit คือ กลุ่มของค่า "0" และ "1"
3. Integer คือ กลุ่มของค่า –214748347 ถึง 214748347
4. Real คือ กลุ่มของค่า –1.0E38 ถึง 1.05E38
5. Character คือกลุ่มของคำ พยัญชนะ 'A'-'Z','a'-'z' อักษรหรือเครื่องหมายพิเศษและตัวอักษรควบคุม
6. Time ได้แก่หน่วยเวลาที่มีพื้นฐานเวลาเป็นวินาที (second ย่อด้วย s หรือ S)
7. Severity Level คือกลุ่มของค่า Note,Warnig,Error,Failure

Operator: เนื่องจาก Type ของ object จะเป็นตัวบ่งบอกถึง operator ที่ object นั้นๆสามารถกระทำได้ ตลอดจน Type ของผลลัพธ์ที่ได้จากการทำงาน ในภาษา VHDL แบ่ง operator ออกได้เป็นประเภทต่างๆตามที่แสดงในรูปที่ 2.29

Delay: หมายถึงช่วงระยะเวลาที่เริ่มต้นของสาเหตุ จนกระทั่งเป็นผลออกมาให้เห็นของปรากฏการณ์หนึ่งๆ ในความเป็นจริงทางธรรมชาติอุปกรณ์ hardware ทุกอย่าง อาทิ เช่น gate ต่างๆ ในระบบดิจิตอลจะมี delay แฝงอยู่เสมอ (ที่เรียกว่า propagation delay time) การที่ภาษา VHDL เป็นภาษาที่ใช้บรรยาย hardware จึงสามารถบรรยายพฤติกรรมของ delay ได้ ซึ่งจำแนก delay ออกเป็น

1. Delay Selection คือการกำหนด delay ที่มีผลต่อ Signal ในรูปแบบลักษณะใด **Inertial delay** ได้แก่ปฏิกิริยาต่อต้านการเปลี่ยนแปลง ระบบที่ประกอบด้วย inertial delay จะแสดงผลของการหน่วยเวลาต่อเมื่อ สัญญาณที่มาบังคับให้เกิดการเปลี่ยนแปลง (จาก driver) มีช่วงระยะเวลานานกว่า inertial delay ของระบบนั้นๆ

Transport Delay จากความหมายของคำว่า transport คือการขนส่งจากจุดหนึ่งไปยังอีกจุดหนึ่ง ฉะนั้น transport Delay จึงแสดงผลของการหน่วงเวลาทุกครั้งเสมอไม่ว่าการเปลี่ยนแปลงของตัวขับ (driver) นั้นจะมีช่วงระยะเวลานานเท่าไร

2. Internal Delay ได้แก่ delay ภายในตัวของระบบ VHDL ทั้งนี้เนื่องมาจากภาษา VHDL เป็นภาษาที่ชุดคำสั่งทั้งหลายทำงานแบบ concurrent ต่อกัน ซึ่งแตกต่างไปจากภาษาโปรแกรมชั้นสูงทั่วๆ ไป แต่การทำงานของระบบจำลองการทำงานด้วยเครื่องคอมพิวเตอร์ (simulator) เครื่องไม่สามารถที่จะทำงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สองคำสั่ง (หรือมากกว่า) ได้กลไกลที่จะทำให้เป็นไปตามหลักการของ concurrent statement ได้นั้นเรียกว่า delta delay (δ) ที่หมายถึง internal delay ของระบบพัฒนา VHDL

LRM: คือคู่มืออ้างอิง (Language Reference Manual) ของมาตรฐาน IEEE 1076

Predefined	Operators
Logical Operators :	NOT, AND, OR, NAND, NOR, XOR
Operand TYPE :	BIT, BOOLEAN
Result TYPE :	BIT, BOOLEAN
Relation Operators :	=, /=, <, <=, >, >=
Operand TYPE :	TYPE ใดๆ ก็ได้
Result TYPE :	BOOLEAN
Arithmetic Operators :	+, -, *, /, **, MOD, REM, ABS
Operand TYPE :	INTEGER, REAL, Physical
Result TYPE :	INTEGER, REAL, Physical
Concatenation Operators :	&
Operand TYPE :	Array ของ TYPE ทุกประเภท
Result TYPE :	Array ของ TYPE ทุกประเภท

รูปที่ 2.29 Operator ที่กำหนดไว้ใน ภาษา VHDL

การแสดงกฎเกณฑ์การเขียนคำสั่งในภาษา VHDL จะอยู่ในรูปของ Bachus Naur Format (BNF) ซึ่งแน่นอนที่ว่าไม่สามารถค้นคว้าเพิ่มเติมได้จากหนังสือคู่มือ IEEE Standard 1076 Language Reference Manual (LRM) การแสดงวิธีการเขียนแบบ BNF นั้นจะมีการดัดแปลงไปบ้างเล็กน้อยเพื่อความสะดวกในการศึกษาและมีสิ่งที่ต้องอธิบาย ณ ที่นี้ดังนี้

เครื่องหมายวงเล็บเหลี่ยม [Square brackets]

- สิ่งที่อยู่ในเครื่องหมายนี้เป็น Option สามารถที่จะเขียนหรือไม่เขียนได้

เครื่องหมายวงเล็บปีกกา

- สิ่งที่อยู่ในเครื่องหมายแสดงว่า ในภาษา VHDL อนุญาติให้มีได้ 0,1 หรือหลายๆครั้ง

ตัวพิมพ์ใหญ่ 'Capitalize'

- ในที่นี้ใช้ในความหมายที่แสดงว่า คำที่เขียนด้วยตัวพิมพ์ใหญ่ เป็นคำของ VHDL และคำที่อยู่ใน Standard Package ที่สามารถนำมาใช้ได้ทันทีโดยไม่ต้องมีการประกาศใหม่ ปกติภาษา VHDL เป็นภาษาที่มี case insensitive เช่นการเขียนในลักษณะนี้จะมีความหมายเดียวกันหมด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

entity = Entity = EnTiTy = ENTITY

นอกจากนี้ยังมีจุดประสงค์ เพื่อที่จะแยกความหมายทั่วไปของภาษาประจำวัน กับภาษา VHDL เช่น ฟังก์ชัน (Function) ในภาษาทั่วไปหมายถึงหน้าที่ในการทำงาน เพื่อให้ได้ผลอย่างใดอย่างหนึ่งออกมา แต่ในขณะที่ Function เป็นโปรแกรมย่อยประเภทหนึ่งของภาษา VHDL

ตัวพิมพ์หนาทึบ ' **BOLD** '

มีจุดประสงค์เพื่อต้องการเน้น ไม่ได้เป็นส่วนของกฎเกณฑ์ของภาษาแต่อย่างใดมาดูตัวอย่างของการเขียนในลักษณะ BNF จาก IF-THEN-ELSE statement

IF condition THEN
{sequential-statement(s)}
[{ ELSEIF condition THEN }
{sequential-statement(s)}]
[ELSE
{sequential-statement(s)}]
END IF ;

ตามกฎเกณฑ์การเขียนที่สามารถที่จะเขียน ELSIF ได้หลายครั้ง หรือไม่มีก็ได้ คำว่า ELSIF เป็น option จะเขียนหรือไม่ก็ได้ แต่ถ้าเขียนจะมีเพียงครั้งเดียว ในขอบเขตของ sequential statement ต่างๆลงไปได้หลายๆคำสั่งหรือไม่ก็ได้

นอกจากนั้นยังมีคำศัพท์ที่มีความหมายเฉพาะสำหรับภาษา VHDL ซึ่งจะใช้ทับศัพท์โดยสามารถเข้าใจได้ทันทีโดยไม่ต้องแปลงกลับมาเป็น ภาษาอังกฤษอีก สุดท้ายจะขอยกตัวอย่างบางคำเช่น

Function หรือ ฟังก์ชัน หมายถึงหน้าที่ที่กระทำการใดการหนึ่งแต่ Function หมายถึงโปรแกรมย่อย (subprogram) ชนิดหนึ่งในภาษา VHDL

Process หมายถึงขบวนการใดๆ ในการทำงาน แต่ Process หมายถึง concurrent statement ในภาษา VHDL

Signal หมายถึงสัญญาณต่างๆ แต่ Signal หมายถึงการประกาศใช้ object ที่มีชั้นเป็นประเภท Signal

2.5.2 ส่วนต่างๆของแบบ (Design Unit)

ภาษา VHDL นั้นประกอบด้วยส่วนต่างๆ ที่สำคัญและเป็นพื้นฐานของการเขียนโปรแกรมระบบดิจิตอลที่สำคัญ 4 หน่วยคือ

1. Entity Design Unit
2. Architecture Design Unit

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3. Package Design Unit

4. Configuration Design Unit

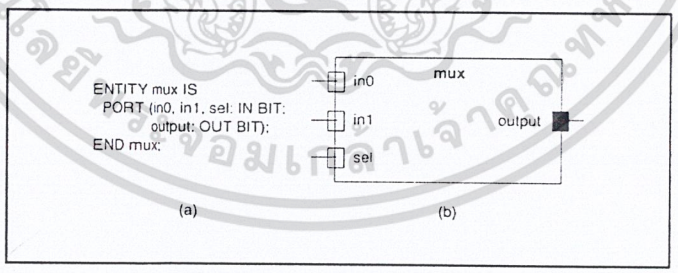
2.5.2.1 Entity Design Unit

หน่วยของแบบ (Design Unit) ส่วนที่ใช้สำหรับติดต่อระหว่างโลกภายนอกกับรูปแบบ(model) ที่จะเขียนขึ้น ส่วนนี้เรียกว่า " entity Design Unit" ในส่วนนี้ใช้กำหนดจุดต่อ (connection point) ของรูปแบบ กำหนดทิศทางการไหลของสัญญาณตามจุดต่างๆ (mode) และประเภทของค่า (type of value) ที่สามารถกำหนดให้กับสัญญาณตามจุดต่างๆ (PORT) ของข้อมูลที่ไหลผ่านจุดต่อเหล่านั้น

```
ENTITY component_name IS
    input and output ports
    physical and other parameters
END [component_name] ;
```

รูปที่ 2.30 โครงสร้างอย่างง่ายๆของ Entity Design Unit

ส่วนนี้จะขึ้นต้นด้วยคำว่า ENTITY และ IS ระหว่างคำทั้งสองเป็นส่วนสำหรับชื่อของรูปแบบที่ต้องการจะเขียน (component_name) สำหรับการตั้งชื่อนั้นจะต้องเป็นไปตามกฎเกณฑ์ของภาษาหลังจากนั้นจะตามด้วยส่วนที่ใช้กำหนดช่องทาง เข้า – ออก ของข้อมูล (input-output) รวมทั้งพารามิเตอร์อื่นๆ ส่วนนี้เรียกว่าส่วนหัว (entity header) และที่สำคัญคือ entity design unit จะต้องปิดท้ายด้วยคำว่า END และเครื่องหมายอัฒภาค หรือ semicolon (;)



รูปที่ 2.31 รูปแบบของ 2:1 multipexer , (a) VHDL entity design , (b) มุมมองของ Interface

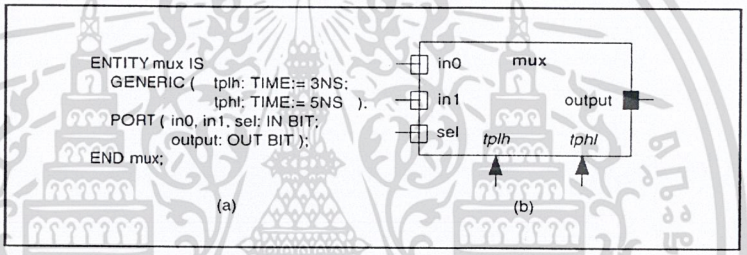
ในรูปที่ 2.31 เป็น Entity design unit ที่บรรยายอุปกรณ์ที่มีชื่อว่า mux ในส่วนหัวของ entity (header) มีการกำหนดจุดต่อ 4 จุด ภายใต้ชุดคำสั่ง PORT โดยที่ 3 จุดแรกเป็นจุดให้ข้อมูลไหลผ่านเข้า (input) ได้แก่ in0, in1 และ sel ซึ่งกำหนดด้วยทิศทางการติดต่อกับภายนอก (mode) เป็นการไหลเข้า (IN) ที่แสดงด้วยรูปสี่เหลี่ยมโปร่งในรูปที่ 2.30 ส่วนจุด output เป็นจุดให้ข้อมูลไหลออก (output) ซึ่งกำหนดด้วยทิศทางการติดต่อกับภายนอกเป็นการไหลออก (out) ที่แสดงด้วยรูปสี่เหลี่ยมทึบรูปที่ 2.31 ส่วน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ประเภท (type) ของข้อมูลที่จะไหล เข้า-ออก นั้น เป็นประเภท bit ที่สามารถมีค่าได้เพียงค่าคือ '0' และ '1' เท่านั้น

นอกจากนั้นผู้ออกแบบยังสามารถกำหนดค่าพารามิเตอร์ทางฟิสิกส์ที่เป็นข้อมูลเพิ่มเติมอื่นๆลงในส่วนหัวของ Entity ได้อีก อาทิเช่น ข้อมูลเกี่ยวกับความเร็วในการทำงานของอุปกรณ์ อันได้แก่ propagation delay time พารามิเตอร์เหล่านี้ เรียกว่า generic ที่กำหนดด้วยคำสั่ง generic จากตัวอย่างในรูปที่ 2.31 สามารถที่สามารถที่จะเพิ่มข้อมูลเกี่ยวกับความเร็วในการทำงานได้ตามที่แสดงในรูปที่ 2.32

เช่นเดียวกับตัวแรกหน่วยของแบบนี้ช่องทางติดต่อกับภายนอก 4 จุด แต่ได้เพิ่มข้อมูลของ Propagation delay time สำหรับนำไปใช้ในการบรรยายพฤติกรรมของอุปกรณ์ ตัวพารามิเตอร์หรือที่เรียกว่า generic นี้มีชื่อว่า tplh และtphl มีประเภทของข้อมูลเป็นเวลา(Time)และจะเป็นค่าตายตัว(default value) สำหรับรูปแบบนี้เสมอ ซึ่งในมี่นี้จะมีค่า 3 nanosecond และ 5 nanosecond ตามลำดับ ค่าตายตัวนี้สามารถที่จะเปลี่ยนแปลงให้มีค่าอื่นได้แล้วแต่กรณี ทั้งนี้ขึ้นอยู่กับอุปกรณ์ที่จะทำการเขียนรูปแบบ



รูปที่ 2.32 รูปแบบ 2:1 multiplexer ที่ประกอบด้วยข้อมูลเกี่ยวกับเวลา
(a) VHDL entity design , (b) มุมมองของ interface

ในบางกรณีสามารถที่จะใช้ภาษา VHDL สร้างรูปแบบที่ปราศจากช่องทางไหล เข้า-ออก ของข้อมูล (input-output) ได้ ซึ่งส่วนใหญ่จะพบในการสร้างรูปแบบ สำหรับตรวจการทำงานของอีกรูปแบบหนึ่ง (VHDL test bench)

```
ENTITY test_bench IS
END test_bench;
```

รูปที่ 2.33 Entity design unit ที่ไม่มีการกำหนดช่องติดต่อกับภายนอก

2.5.2.2 Architecture Design Unit

คือส่วนที่ใช้เขียนบรรยายกำหนดพฤติกรรมของรูปแบบ ในมุมมองของการจำลองการทำงาน (Simulation) พฤติกรรมต่างๆที่บรรยายในส่วนนี้ขึ้นอยู่กับข้อมูลที่ผ่าน เข้า-ออก ตรงช่องทางตลอดจนพารามิเตอร์ต่างๆ (port and generics) ที่กำหนดใน entity design unit

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

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

รูปที่ 2.34 แสดงให้เห็นโครงสร้างอย่างง่ายๆของ architecture design unit

ส่วนของ Architecture design unit นั้นเริ่มต้นด้วยคำว่า Architecture และตามด้วยชื่อ (identifier) สิ่งที่ต้องกำหนดลงไปได้แก่ สิ่งที่แสดงให้เห็นว่า architecture นั้นใช้บรรยาย entity design unit ใด (OF<*entity design unit*> IS) ส่วนที่อยู่ระหว่าง Architecture และ Begin เป็นส่วนหนึ่งประกาศกำหนด (Architecture declarative area) ที่เป็นเพียงส่วนเผื่อเลือก (option) ในบริเวณนี้สามารถใช้เขียนประกาศกำหนดค่าต่างๆที่จะนำไปใช้ภายใน Architecture นั้นได้ อาทิเช่นประเภท (type) ต่างๆ (ตัวอย่างเช่น Bit, Bit_vector), สัญญาณ (Signal) , ตัวคงที่ (Constant), โปรแกรมย่อย (ได้แก่ Function และ Procedure) และอุปกรณ์ (Component) ส่วนที่ใช้บรรยายความสัมพันธ์ระหว่างข้อมูลที่ไหลเข้า และไหลออกของรูปแบบ (สัญญาณที่กำหนดในชุดคำสั่ง Port) นั้นจะถูกบรรยายในบริเวณเนื้อที่ระหว่างคำว่า Begin กับ End ของ Architecture Design Unit และนอกจากนั้นชุดคำสั่งทุกคำสั่งที่อยู่ภายใต้บริเวณนี้จะเป็นชุดคำสั่งแบบแข่งขนาน (concurrent statement) เท่านั้น Architecture Design Unit จะต้องปิดท้ายด้วยคำสั่ง End และชื่อของ architecture (identifier) นั้นๆที่เป็นส่วนหนึ่งเผื่อเลือก

โดยทั่วไปการเขียนรูปแบบระบบดิจิตอลด้วยภาษา VHDL สามารถเขียนได้ในลักษณะต่างๆตามที่กล่าวมาแล้วดังนี้

- Dataflow description
- Behavioral description
- Structure description
- Mixed model description

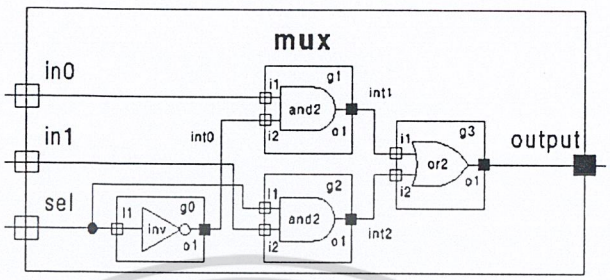
```
ARCHITECTURE data_flow OF mux IS
BEGIN
  output  <=  ((NOT sel) AND in0) OR (sel AND in1);
END data_flow;
```

รูปที่ 2.35 Architecture design unit ของ 2:1 mux ตาม Boolean expression

$$output = (\overline{sel} \cdot in0) + (sel \cdot in1)$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 2.35 (a) ส่วนที่บรรยายความสัมพันธ์ระหว่างข้อมูลที่ไหลเข้า (in0,in1) กับข้อมูลที่ไหลออก (output) ประกอบด้วยชุดคำสั่งแบบแข่งขนานเพียงชุดเดียวซึ่งเป็นการบรรยายพฤติกรรมของ 2:1 mux การบรรยายลักษณะนี้เรียกว่า Dataflow description หรือ register transfer level (RTL)



รูปที่ 2.36 (a) Architecture description ของ 2:1 multiplexer (structural description)

รูปที่ 2.36 เป็น Architecture ของการบรรยาย 2:1 mux ในลักษณะของ structural description โดยใช้ inverter (inv ที่อุปกรณ์ g0) , And-gate 2 input จำนวน 2 gate (and2 ที่อุปกรณ์ g1 และ g2) ,OR-gate 2 input จำนวน 2 gate (or2 ที่อุปกรณ์ g3) มาสร้างตาม Boolean expression ของรูปที่ 2.36

```
ARCHITECTURE struct OF mux IS
    COMPONENT inv
    PORT(   i1: IN BIT; o1: OUT BIT   );
    END COMPONENT;
    COMPONENT and2
    PORT(   i1, i2: IN BIT; o1: OUT BIT   );
    END COMPONENT;
    COMPONENT or2
    PORT(   i1, i2: IN BIT; o1: OUT BIT   );
END COMPONENT;
    SIGNAL int0, int1, int2: BIT;
BEGIN
    g0: inv
    PORT MAP (i1 => sel, o1 => int0);
    g1: and2
    PORT MAP (i1 => in0, i2 => int0, o1 => int1);
    g2: and2
    PORT MAP (i1 => sel, i2 => in1,  o1 => int2);
    g3: or2
    PORT MAP (i1 => int1, i2 => int2, o1 => output);
END struct;
```

รูปที่ 2.36 (b) : Architecture description ของ 2: 1 mux (structural description)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 2.36 (a) เป็นโครงสร้าง (Structure) ของการต่อวงจรในโดยที่มีสัญญาณ in0,in1 และ in2 เป็นสัญญาณภายใน รูปที่ 2.36 (b) เป็น architecture ของ VHDL model ที่เขียนกำหนดการเชื่อมต่อภายในด้วย VHDL netlist

```
ARCHITECTURE behav OF mux IS
BEGIN
   PROCESS (in0, in1, sel)
   BEGIN
      IF (sel = '0') THEN
         output <= in0;
      ELSE
         output <= in1;
      END IF;
   END PROCESS;
END behav;
```

รูปที่ 2.37 Architecture description ของ 2:1 mux (behavioral description)

การบรรยาย 2:1 mux ในลักษณะของ behavioral description ได้แสดงให้เห็นอีกครั้งในรูปที่ 2.37 ซึ่งจะเห็นได้ว่าส่วนที่เป็น Architecture design unit ทั้งหมด ต่างบรรยายพฤติกรรมเดียวกัน และจะให้ผลลัพธ์จากการจำลองการทำงานที่เหมือนกัน

2.5.2.3 Package Design Unit

ข้อมูลต่างๆตลอดจนโปรแกรมย่อย (Subprogram) ที่เป็นประโยชน์ต่อการเขียนรูปแบบบรรยายระบบดิจิตอล สามารถเก็บไว้ในส่วนที่เรียกว่า Package ได้ และข้อมูลเหล่านี้สามารถนำไปใช้ได้โดย entity design unit, architecture design unit หรือจาก package design unit อื่นๆ ด้วยชุดคำสั่ง USE statement นอกจากนั้นสิ่งที่นิยมทำกันมากคือรูปแบบ (model) มาตรฐานต่างๆ อาทิเช่น standard components (model ของ IC ตระกูล 74xx) จะถูกเก็บไว้ใน package ที่ทุกคนสามารถเข้าถึงและนำไปใช้ได้ สิ่งที่สามารถประกาศหรือบรรจุได้ใน package ได้แก่

- Subprogram
- Type
- Constants
- Signals
- Aliases
- Attributes
- Component
- Disconnection Specification

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โดยปกติแล้ว package จะแบ่งเป็นสองส่วนคือ

1. Package declaration
2. Package body

เนื่องจาก package ถูกสร้างเป็นส่วนแยกต่างหากออกจากรูปแบบ (Model) ที่กำลังเขียนอยู่ฉะนั้น การที่นำ package ไปใช้นั้นจะต้องมีเชื่อมโยงหรืออ้างอิงเสียก่อนซึ่งในภาษา VHDL สามารถกระทำได้ด้วยชุดคำสั่ง use statement

Package declaration

ส่วนที่มีความสำคัญที่สุด Package (มองในแง่ของการนำไปใช้จากภายนอก) ได้แก่ Package declaration เพราะจะเป็นส่วนที่กำหนดชื่อ (identifier) ของสิ่งที่ประกาศอยู่ภายใน Package สำหรับนำไปใช้ภายนอกตัวของ Package เอง ถ้าสิ่งใดๆ ถูกประกาศในส่วนของ Package body แต่ไม่ถูกประกาศใน Package declaration จะไม่สามารถนำค่า และพฤติกรรมไปใช้จากส่วนนอกได้ ซึ่งสามารถเปรียบเทียบได้กับสิ่งที่ประกาศไว้ในส่วนของ entity declaration คือ interface ที่มีหน้าที่ติดต่อกับโลกนอก ฉะนั้นโดยทั่วไปแล้ว Package สามารถสร้างขึ้นได้โดยไม่จำเป็นต้องมีส่วน body และยังสามารถถูกนำไปใช้จากรูปแบบ(model) ภายนอกได้เช่นใช้สำหรับประกาศ type หรือ signal (global) เช่นเดียวกันกับ Package body ที่ไม่จำเป็นต้องมี Package declaration แต่ Package นั้นจะไม่สามารถถูกนำไปใช้จากรูปแบบ (model) อื่นได้ การเขียน Package declaration มีกำเกณฑ์ตามที่แสดงในรูปที่ 2.38

```
PACKAGE package_name IS
    package_declarative_part
END package_name ;
```

รูปที่ 2.38 โครงสร้างของ package declaration

คำว่า package และ End package กำหนดขอบเขตของ Package declaration ระหว่างนั้นจะเป็นส่วนที่ใช้ประกาศต่างๆสิ่งที่สามารถประกาศในส่วนนี้ได้แก่

- ส่วนประกาศกำหนดโปรแกรมย่อย (Subprogram declarations)
- Type declaration
- Subtype declaration
- Object declaration (signal, constants)
- Alias declaration
- Attribute specification
- Component specifications
- Disconnection specifications

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับใช้ในการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 2.39 เป็นตัวอย่างของการเขียน Package declaration ที่มีการประกาศ Type, Constant, Component และ Signal

```
-- package declaration
PACKAGE pack_funct IS
        FUNCTION mean (a, b, c : REAL) RETURN REAL;
END pack_funct;
-- package body
PACKAGE BODY pack_funct IS
        FUNCTION mean (a, b, c : REAL) RETURN REAL IS
        BEGIN
            RETURN (a + b + c)/3.0;
        END mean;
END pack_funct;
```

รูปที่ 2.39 ตัวอย่างของ Package declaration

Package body

โครงสร้างที่ประกอบด้วยคำสั่งต่างๆ ในรูปของคำสั่งลำดับ (Sequential statement) ที่ใช้บรรยายฟังก์ชันการทำงานโปรแกรมย่อย (subprogram) ทั้งหลายที่ชื่อของโปรแกรมย่อยนั้นๆที่ถูกประกาศไปในส่วนของ Package declaration แล้ว จะถูกเก็บไว้ใน Package body ทั้งนี้รวมทั้ง deferred constant (อันได้แก่ตัวคงที่ถูกประกาศชื่อก่อนในส่วนของ declaration แต่ถูกกำหนดค่าในส่วนของ body ของ Package) ฉะนั้นส่วน Package body จึงไม่จำเป็นต้องมี ถ้าในส่วน Package declaration ไม่มีการประกาศชื่อ (identifier) ที่มีโปรแกรมย่อย (subprogram) หรือ deferred constant การเขียน Package body นั้นเป็นไปตามกฎเกณฑ์ที่แสดงในรูปที่ 2.40

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

รูปที่ 2.40 โครงสร้างของ Package body

ชื่อของ Package_name ที่ใช้ใน Package body จะต้องเป็นชื่อเดียวกับชื่อที่กำหนดไว้ใน Package declaration ในรูปที่ 2.41 แสดงตัวอย่างของการเขียน Package (declaration และ body ที่สัมพันธ์กัน) โดยนำการกำหนดโปรแกรมย่อย (subprogram) ประเภท Function

ในส่วน ของ Package declaration จะบรรจุส่วนที่เรียกว่า function declaration ในที่นี้เป็นการประกาศชื่อ ของโปรแกรมย่อย (Function) mean และส่วนที่บรรยายการทำงานของโปรแกรมย่อย mean (เรียกว่า function body) จะถูกเก็บไว้ในส่วน Package body แต่สิ่งที่สำคัญอย่างหนึ่งของการเขียน Package คือ ก่อนที่จะนำ Package ไปใช้ (โดยการอ้างอิงจากภายนอก) Package นั้นจะต้องถูกวิเคราะห์เสียก่อนว่าถูกต้อง หรือพูดง่ายๆได้ว่า จะต้องผ่านการ compile ก่อนนั้นเอง

เอกสารนี้สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
-- package declaration
PACKAGE pack_funct IS
        FUNCTION mean (a, b, c : REAL) RETURN REAL;
END pack_funct;
-- package body
PACKAGE BODY pack_funct IS
        FUNCTION mean (a, b, c : REAL) RETURN REAL IS
        BEGIN
            RETURN (a + b + c)/3.0;
        END mean;
END pack_funct;
```

รูปที่ 2.41 ตัวอย่างการเขียน Package

Configuration Design Unit

ดังที่ทราบกันแล้วว่ารูปแบบ หนึ่งของระบบดิจิตอลไม่ว่าจะเป็นอะไร จะมี Entity design unit ได้เพียงหน่วยเดียวเท่านั้น แต่ในขณะที่ Entity design unit หนึ่งหน่วยนี้อาจจะมี architecture ที่เป็นหน่วยรองได้หลายหน่วย

ดังนั้นในการจำลองการทำงานของรูปแบบ (model) นั้น simulator จะนำ architecture อันไหนนั้นจะต้องบอกให้ simulator ทราบและในรูปแบบ VHDL นั้นการบอกหรือการกำหนดคือการใช้ Configuration ประกอบ entity กับ architecture design unit ที่ต้องการเข้าด้วยกัน รูปที่2.42 แสดงกฎเกณฑ์การเขียน Configuration

```
CONFIGURATION identifier OF entity_name IS
        configuration_declarative_part
END;
```

รูปที่ 2.42 โครงสร้างของ Configuration

ในรูปที่ 2.43 เป็นรูปแบบ (model) ง่ายๆของ And-gate2 input ที่มีส่วนรองอันได้แก่ architecture สองแบบคือ Dataflow description และ behavioral description

ก่อนที่จะนำรูปแบบ (model) ไปจำลองการทำงานจะต้องมีการประกอบ Architecture ที่ต้องการเข้ากับ Entity design unit เสียก่อน (configuration) ซึ่งระบบ VHDL ส่วนใหญ่ถ้าไม่กำหนดการประกอบ Architecture เครื่อง simulation จะนำ Architecture หน่วยสุดท้ายที่ผ่านการวิเคราะห์ไปใช้งานจำลองการทำงาน ในรูปแบบที่ 2.44 แสดงการประกอบ Architecture ชื่อ Dataflow เข้ากับ Entity design unit

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
ENTITY and2 IS
        GENERIC (ttl_delay : TIME := 3NS);
        PORT (   in1, in2 : IN BIT;
                 output : OUT BIT   );
END and2;

ARCHITECTURE dataflow OF and2 IS
BEGIN
        output <= in1 AND in2 AFTER ttl_delay;
END dataflow;

ARCHITECTURE behave OF and2 IS
BEGIN
        PROCESS (in1, in2)
        BEGIN
           IF (in1 = '1' AND in2 = '1') THEN
              output <= '1' AFTER ttl_delay;
           ELSE
           output <= '0';
        END PROCESS;
END behave;
```

รูปที่ 2.43 VHDL model ของ AND-gate2 input

```
CONFIGURATION dataflow_and OF and2 IS
        FOR dataflow
        END FOR;
END dataflow_and;
```

รูปที่ 2.44 Configuration ของรูปแบบ (Model) and2

ในตัวอย่างแสดงการประกอบโครงร่าง (Configuration) ชื่อ dataflow_and ให้เป็นตัวกำหนดการเชื่อม entity ชื่อ and2 เข้ากับ architecture ชื่อ Dataflow ซึ่งจะเห็นได้ว่าเป็นการประกอบโครงร่างอย่างง่ายๆ ในรูปที่ 2.45 จะเป็นโครงสร้างที่ซับซ้อนมากขึ้น

ตัวอย่างในรูปที่ 2.45 (a) configuration ชื่อ decode_llcon เชื่อมต่อ entity ชื่อ decode กับ architecture ชื่อ structure และภายใน architecture ชื่อ structure นี้ประกอบด้วยอุปกรณ์ย่อยๆซึ่งแต่ละตัวมีลักษณะการประกอบกับส่วนอื่นด้วยโครงสร้าง For...End และ For...End ที่กำหนด configuration จำเพาะที่อยู่ภายใต้อีกครั้ง (เรียกว่า configuration specification) และ configuration ภายในนี้จะต้องผ่านการวิเคราะห์สำหรับ configuration ปัจจุบัน (วงนอก) ผลลัพธ์ที่ได้จากการ วิเคราะห์จะต้องถูกต้องเก็บไว้ภายใต้ working directory (symbolic name Work) การกำหนดอุปกรณ์จำเพาะนั้น สามารถที่จะ กระทำได้โดยการบอกชื่อ label เช่นในกรณีของ i1 สำหรับ inv หรือใช้ all หรือ others ดังเช่นในกรณีของอุปกรณ์ and3 เป็นต้น ในที่นี้หมายความว่าอุปกรณ์ inv จะใช้ configuration ชื่อ invcon จาก library Work (หรือ working directory ที่มีชื่อตายตัวว่า Work) ที่กำลังทำงานอยู่ จะเห็นได้ว่าเป็นการสร้าง configuration ซ้อน configuration การที่จะใช้งาน configuration บนสุด (decode_llcon) ได้ configuration ล่างสุด (invcon และ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

and3con) จะต้องผ่านการวิเคราะห์ว่าถูกต้องมาแล้ว การสร้างโครงสร้างเช่นนี้เรียกว่า lower-level configuration

รูปที่ 2.45 (b) ใช้ Entity และ architecture design unit เดียวกับรูป (a) สำหรับ inv (กำหนดในส่วน architecture declaration) ถูกกำหนดจำเพาะด้วย label i1 กับอุปกรณ์ที่มี entity ชื่อ inv (สามารถมีชื่อเหมือนหรือต่างกันได้) และใช้ architecture ชื่อ behave ที่ยังคงเหลือในโครงสร้าง (ถ้ามีการใช้อุปกรณ์ inv มากว่าหนึ่งตัว) จะถูกประกอบเข้ากับอุปกรณ์ที่มี entity ชื่อ inv เช่นกันแต่จะใช้ architecture ชื่อ dataflow แทน ทั้งนี้ entity design unit ชื่อ inv และ architecture design unit ชื่อ behave และ dataflow จะต้องถูกวิเคราะห์ก่อน ผลลัพธ์ที่ได้จากการวิเคราะห์จะต้องถูกเก็บไว้ภายใต้ working directory การประกอบลักษณะนี้เรียกว่า entity architecture pair configuration

```
ARCHITECTURE examp_config OF decode IS
    COMPONENT inv PORT (in1: IN BIT; o1: OUT BIT);
    END COMPONENT;
    COMPONENT and3 PORT (in1, in2, in3 : BIT; o1: OUT BIT);
    END COMPONENT;
    :
END examp_config;
```

```
CONFIGURATION decode_llcon OF decode IS
    FOR structural
        FOR i1: inv USE CONFIGURATION WORK.invcon;
        END FOR;
        FOR i2: inv USE CONFIGURATION WORK.invcon;
        END FOR;
        FOR ALL: and3 USE CONFIGURATION WORK.and3con;
        END FOR;
    END FOR;
END decode_llcon;
```

รูปที่ 2.45 (a) ตัวอย่างของ Configuration

```
CONFIGURATION decode_eacon OF decode IS
    FOR structural
        FOR i1: inv USE ENTITY WORK.inv (behave);
        END FOR;
        FOR OTHERS: inv USE ENTITY WORK.inv (dataflow);
        END FOR;
        FOR a1: and3 USE ENTITY WORK.and3 (behave);
        END FOR;
        FOR OTHERS: and3 USE ENTITY WORK.and3 (dataflow);
        END FOR;
    END FOR;
END decode_eacon;
```

รูปที่ 2.35 (b) ตัวอย่างของ Configuration

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.6 ภาษาซี

2.6.1 ประวัติความเป็นมาของภาษาซี

ภาษาซี (C Language) เกิดขึ้นจากการดัดแปลงภาษาบี (B Language) ภาษาซีเป็นภาษาที่คิดค้นโดย เดนนิส ริสชี่ (Dennis Ritchie) ที่ศูนย์วิจัยเบล (Bell Laboratories) ในสหรัฐอเมริการาวปี ค.ศ. 1972 และเป็นภาษาที่ใช้เขียนระบบปฏิบัติการยูนิกส์(Unix Operating System) ซึ่งเป็นระบบปฏิบัติการในระบบเครือข่ายคอมพิวเตอร์ที่แพร่หลายในปัจจุบัน

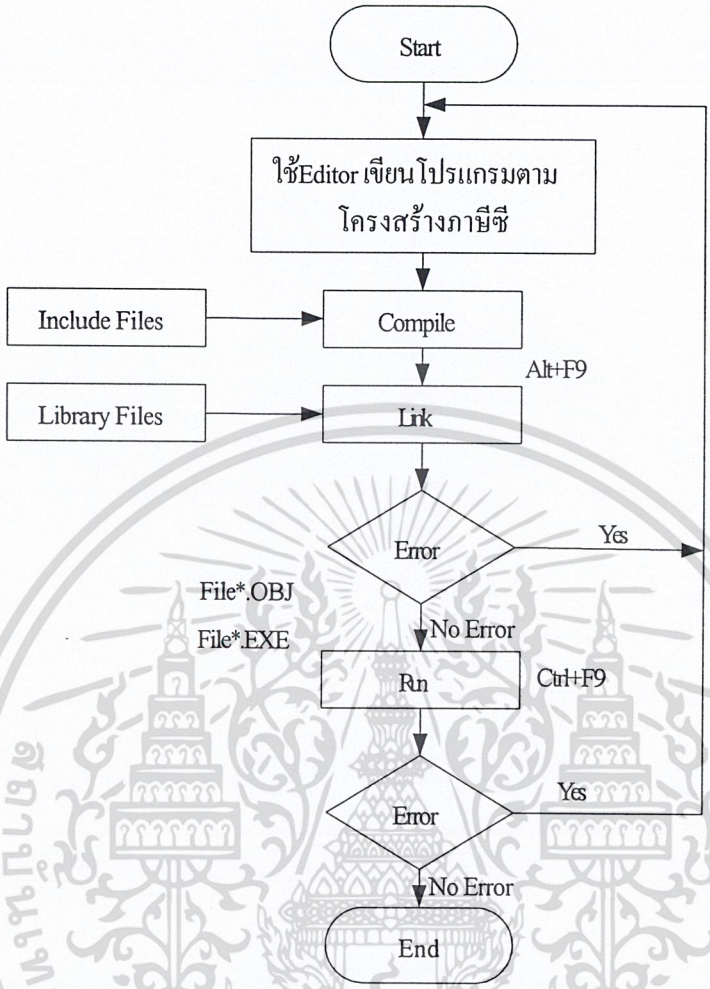
ภาษาซีที่จะกล่าวในที่นี้ คือ เทอร์โบซี (Turbo C) ของบริษัทบอร์แลนด์ (Borland International, Inc. Version 2.0) ภาษาซีถูกจัดให้เป็นภาษาในระดับกลาง เป็นภาษาที่มีโครงสร้าง การเขียนโปรแกรมเป็นแบบลำดับและเป็นภาษาที่ใช้คอมไพเลอร์ (Compiler) เป็นตัวแปลความหมายภาษาซีเป็นภาษาที่สามารถประยุกต์ใช้งานในด้านต่างต่างๆ ได้อย่างกว้างขวาง เช่น โปรแกรมระบบ (Operating System) โปรแกรมประยุกต์ใช้งาน (Application Program) โปรแกรมอรรถประโยชน์ (Utilities Program) โปรแกรมทางด้านงานควบคุม (Control Program)

2.6.2 การประมวลผลของภาษาซี

การทำงานของภาษาซีเริ่มจากใช้ Editor เขียนโปรแกรม (Source Program) ซึ่งต้องเขียนตามลักษณะโครงสร้างของภาษาซี แล้วบันทึก (Save) เป็นไฟล์ที่มีนามสกุลเป็น* .C (File* .C) จากนั้นทำการคอมไฟล์ ถ้ามีข้อผิดพลาด (Error) ให้กลับไปแก้ไขโปรแกรม (Source program) แต่ถ้าไม่มีข้อผิดพลาดจะได้ไฟล์ .BOJ แล้วทำการ Link ไฟล์ * .OBJ เข้ากับไฟล์ Libraries ถ้า ไม่มีข้อผิดพลาดจะได้ไฟล์ * .EXE

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โฟลว์ชาร์ตการประมวลผลของภาษาซี



รูปที่ 2.46 โฟลว์ชาร์ตการประมวลผลของภาษาซี

2.6.3 ส่วนประกอบของภาษาซี

ส่วนประกอบของภาษาซีสามารถแบ่งออกได้เป็น ส่วนใหญ่ ๆ คือ ไฟล์หัวโปรแกรม (Header files) และส่วนของโปรแกรม

2.6.3.1ไฟล์หัวโปรแกรม (Header files) เป็นไฟล์ที่ที่มีส่วนขยายเป็น *.h ใช้เก็บไลบรารี (Library) ของภาษาซีเพื่อใช้รวม (Include) ในการคอมไพล์โปรแกรม เช่น stdio.h เป็นไฟล์ที่เก็บไลบรารีมาตรฐานเกี่ยวกับการรับข้อมูลและการแสดงผล ซึ่งฟังก์ชัน Printf (); ถูกนิยามไว้ใน stdio.h ดังนั้น โปรแกรมใดที่มีการเรียกใช้ฟังก์ชัน Printf (); ควรจะประกาศไฟล์ stdio.h เพื่อใช้รวมในการคอมไฟล์โปรแกรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.6.3.2 ส่วนของตัวโปรแกรม จะเริ่มต้นด้วยฟังก์ชัน Main () ซึ่งเป็นฟังก์ชันหลักของโปรแกรม และมีเครื่องหมายปีกกาเปิด ({) เป็นเครื่องหมายเริ่มต้นการเขียนโปรแกรม เครื่องหมายปีกกาเปิด ({) เป็นเครื่องหมายจบโปรแกรม ภายในฟังก์ชัน main () จะประกอบไปด้วยชุดคำสั่งและฟังก์ชันต่างๆ ซึ่งเกือบทั้งหมดจะเปิดท้ายด้วยเครื่องหมายเซมิคอลอน (;) การเขียนโปรแกรมภาษาซีจะเขียนด้วยตัวอักษรภาษาอังกฤษพิมพ์เล็ก ภาษาซีจะแยกความแตกต่างระหว่างตัวอักษรพิมพ์เล็กและพิมพ์ใหญ่ และส่วนของคำอภิบายจะใช้เครื่องหมาย /* และ */ คั่นตามลำดับซึ่งจะใช้ในการอภิบายโปรแกรมไม่มีผลต่อการคอมไพล์ และในภาษาซีจะใช้คำว่า ฟังก์ชัน แทนคำว่า คำสั่ง ซึ่งทั้ง 2 คำมีความหมายเหมือนกัน

ตารางที่ 2.8 ตารางแสดงตัวอย่างการเขียนโปรแกรม

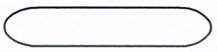
โปรแกรมตัวอย่าง	คำอธิบาย
/* Example Program */	ส่วนของคำอธิบายไม่มีผลต่อการคอมไพล์
# include <stdio.h>	ไฟล์ส่วนหัวโปรแกรม
main()	ฟังก์ชันหลักของโปรแกรม
{	ปีกกาเปิดเริ่มต้นการเขียนโปรแกรม
clrscr();	
Printf("**********\n");	ตัวโปรแกรมภาษาซี
Printf("Example Program\n");	
Printf("**********\n");	
}	ปีกกาปิดจบโปรแกรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

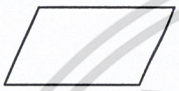
2.6.4 การเขียนโฟลว์ชาร์ต (Flowchart)

การเขียน โฟลว์ชาร์ต หมายถึงการเขียนแผนภาพที่เป็นลำดับเพื่อแสดงขั้นตอนการทำงานของโปรแกรมหรือระบบงานเพื่อให้ง่ายต่อการทำความเข้าใจและการแก้ปัญหา การเขียนโฟลว์ชาร์ตควรเขียนจากบนลงล่างมีจุดเริ่มต้นและสิ้นสุดจุดเดียว

สัญลักษณ์พื้นฐานของโฟลว์ชาร์ต



หมายถึง จุดเริ่มต้นและจุดสิ้นสุด (Start/Stop)



หมายถึง การรับข้อมูล และการแสดงผล (Input/Output)

หมายถึง การคำนวณและการประมวลผล (Process)

หมายถึง ตรวจสอบเงื่อนไขและการเปรียบเทียบ (Compare)

หมายถึง จุดต่อ (Connection)

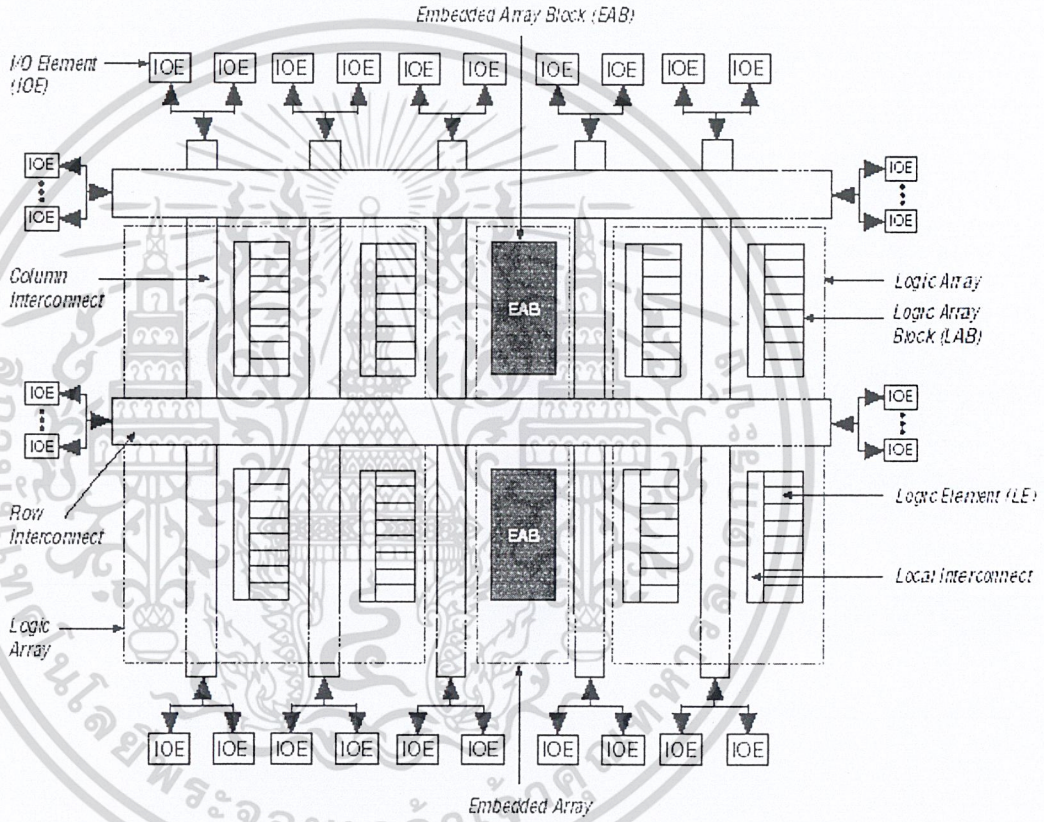
หมายถึง ทิศทางการทำงาน (Direction)

รูปที่ 2.47 สัญลักษณ์พื้นฐานการเขียนโฟลว์ชาร์ต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.7 Field Programmable Gate Array (FPGA)

FPGA ของบริษัท Altera ตระกูล FLEX 10 K เป็นอุปกรณ์ที่มีความหนาแน่นเกทประมาณตั้งแต่ 10,000 – 250,000 เกท โดยการจัดโครงสร้าง (Configuration) จะใช้วิธีโหลดโครงสร้างเข้าไปในSRAM ภายใน ซึ่งหมายความว่าถ้าไม่ได้มีการจ่ายไฟเลี้ยงให้ โครงสร้างที่จัดเอาไว้ก็จะหายไป FPGA ประเภทนี้จะสามารถโปรแกรมซ้ำได้ไม่จำกัดจำนวนครั้ง และการทำงานตามลอจิกฟังก์ชั่นจะใช้วิธีการเปิดตารางดู (Look Up table : LUT) โดยโครงสร้างของ FPGA ตระกูล FLEX 10 K แสดงดังรูปที่ 2.48



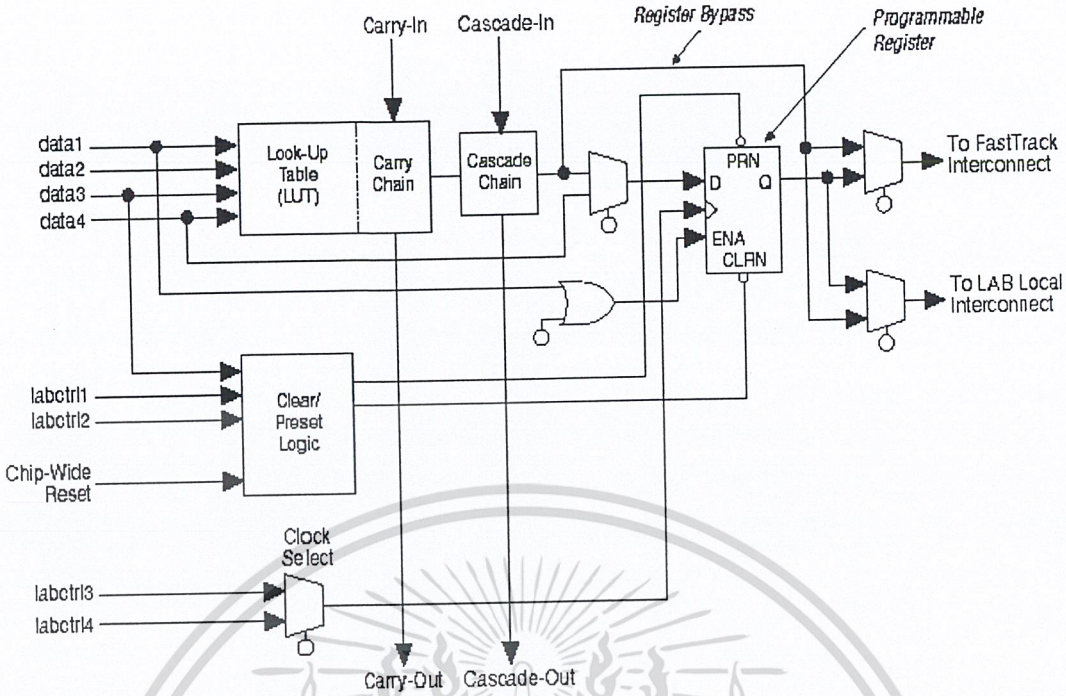
รูปที่ 2.48 แสดงโครงสร้างของ FPGA ตระกูล FLEX 10K

ในโครงสร้างของ FPGA ตระกูล FLEX 10 K สามารถที่จะแบ่งเป็นส่วนต่างๆได้ดังนี้

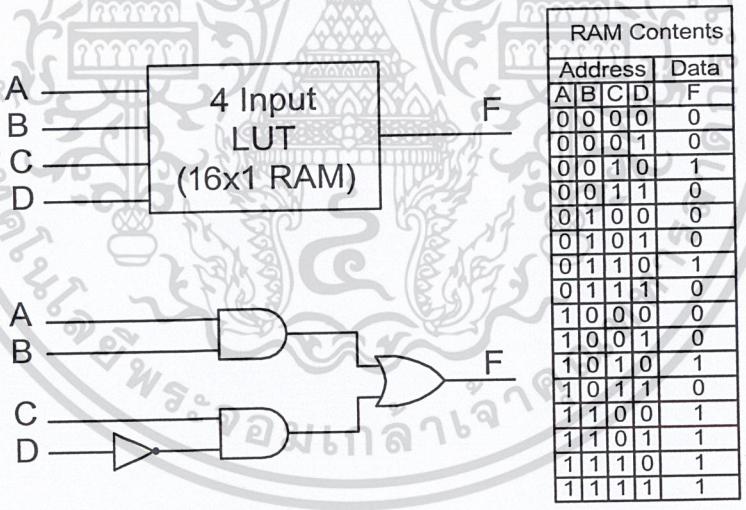
2.7.1 Logic Element (LE)

ในรูปที่ 2.48 แสดงโครงสร้างภายในของ LE โดยการกระทำทางบูลีนของลอจิกเกทจะสร้างด้วยวิธีการ LUT โดย LUT คือ 1x16 SRAM ซึ่ง LUT เพียงตัวเดียวสามารถนำมาทำโครงข่ายของลอจิกเกทที่มี 4 อินพุท และ 1 เอาท์พุท โดยโครงข่ายของลอจิกเกทจะถูกแปลงไปเป็นตารางค่าความจริง (Truth Table) ดังแสดงในรูปที่ 2.49

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.49 แสดงโครงสร้างภายในของ LE

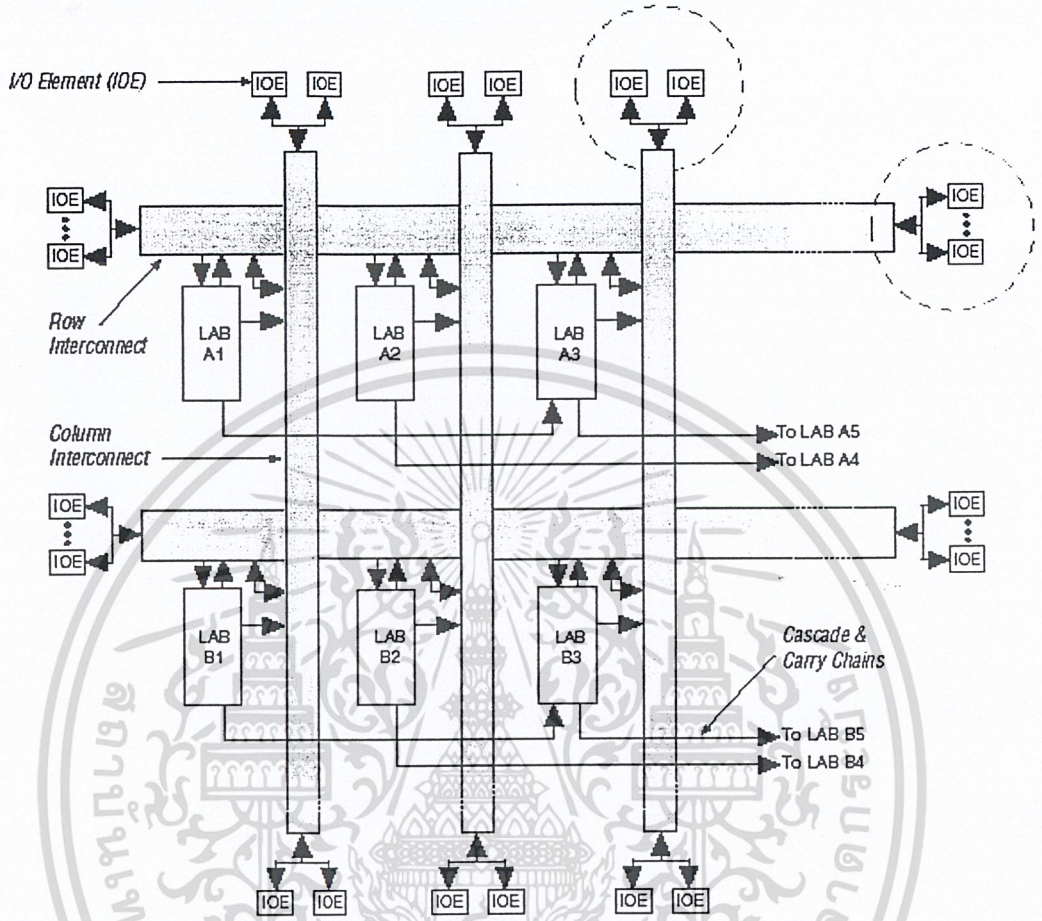


RAM Contents				
Address				Data
A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

รูปที่ 2.50 การใช้งาน LUT เป็นโครงข่ายของลอจิก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ถ้าโครงข่ายของลอจิกเกทมีความซับซ้อนขึ้นจะต้องใช้ LUT ของแต่ละ LE เป็นจำนวนหลายตัว โดยเอาท์พุทของ LUT จะส่งต่อไปยังฟลิปฟลอปและต่อไปยังโครงข่ายการเชื่อมต่อ (Interconnection Network) ดังแสดงในรูปที่ 2.51

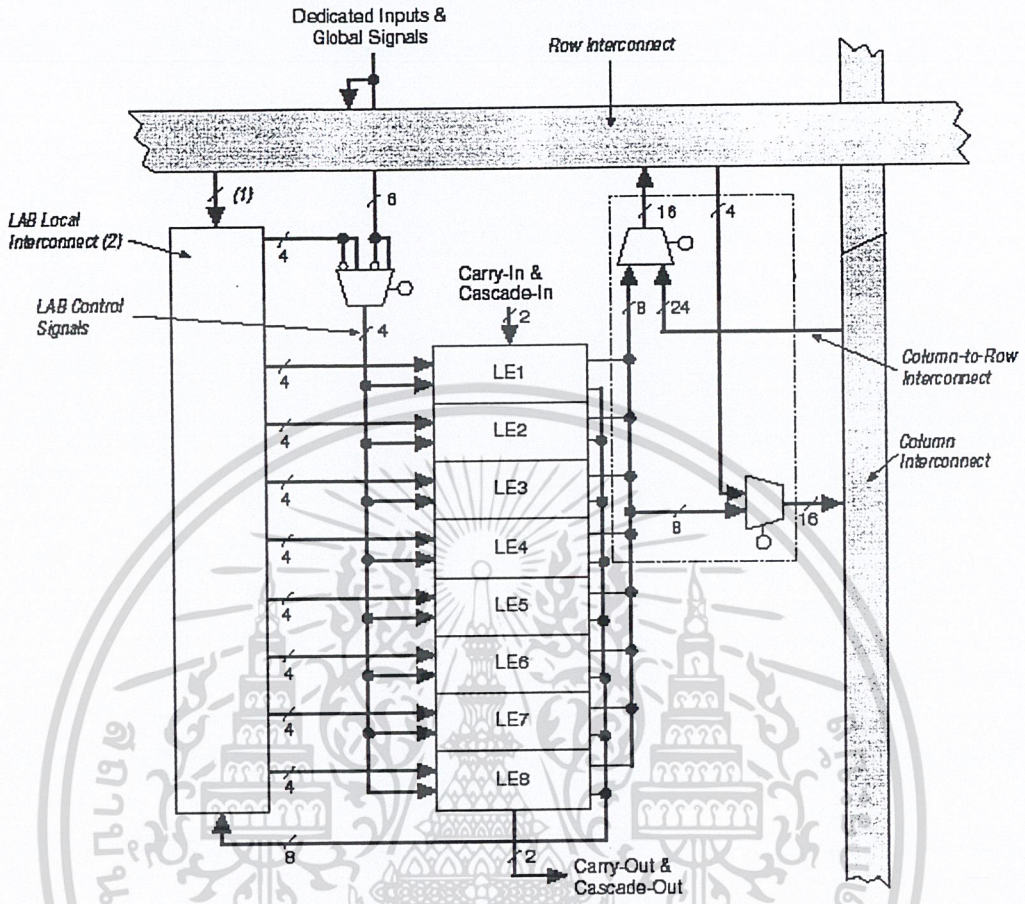


รูปที่ 2.51 แสดงโครงข่ายของการเชื่อมต่อ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.7.2 Logic Array Block (LAB)

LAB 1 ตัว จะประกอบไปด้วย 8 LE ดังแสดงในรูปที่ 2.52

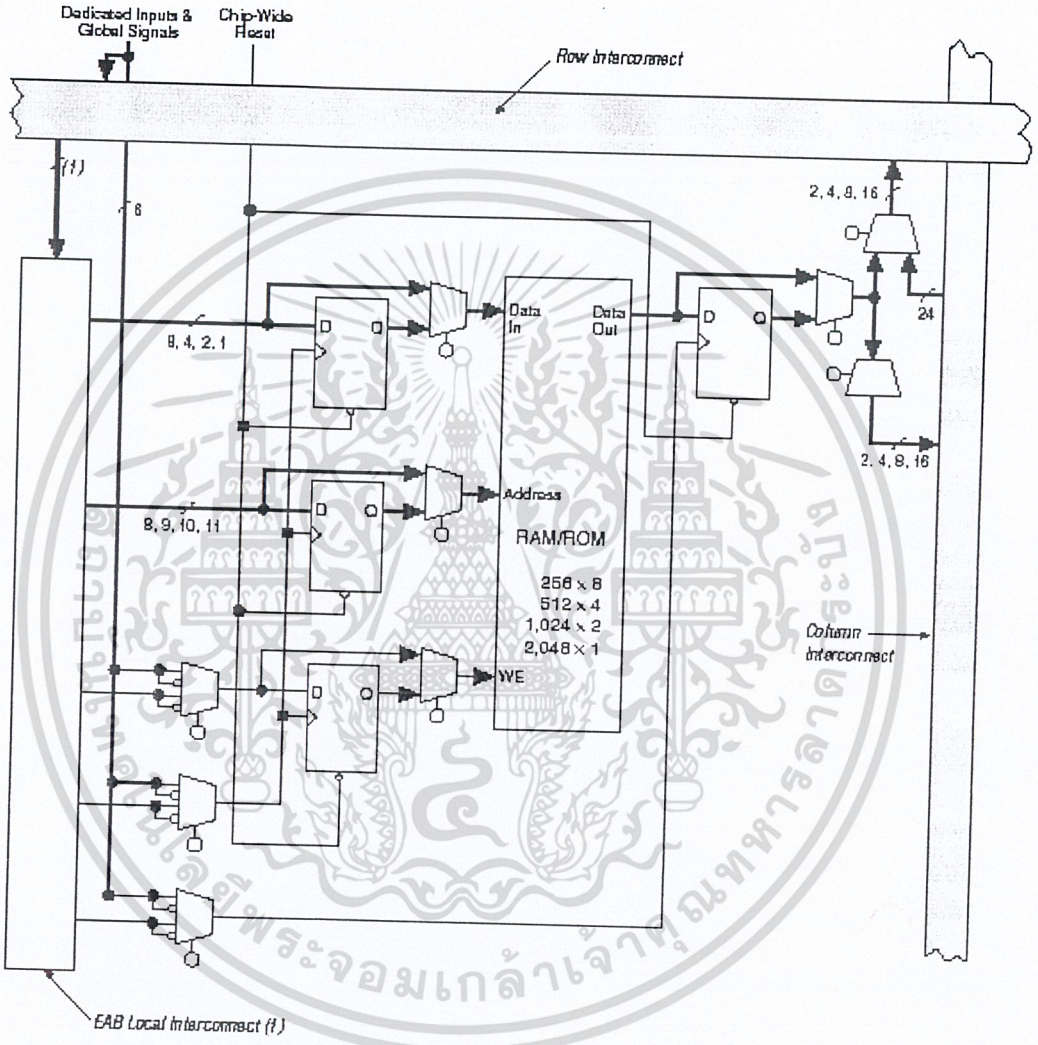


รูปที่ 2.52 แสดงโครงสร้างภายในของ LAB

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.7.3 Embedded Array Block (EAB)

สถาปัตยกรรมโดยทั่วไปของ FLEX 10 K จะมีลักษณะของ LAB ที่มีการจัดเรียงแบบเมตริกซ์ และ EAB ซึ่งมีการเชื่อมต่อผ่านทางแถวและคอลัมน์ โดยในแต่ละแถวจะมี 1 EAB ซึ่ง 1 EAB จะมีขนาด 2048 บิท และสามารถกำหนดความกว้าง (Width) ความลึก (Depth) ของ EAB ได้โดยไม่ส่งผลต่อความเร็ว

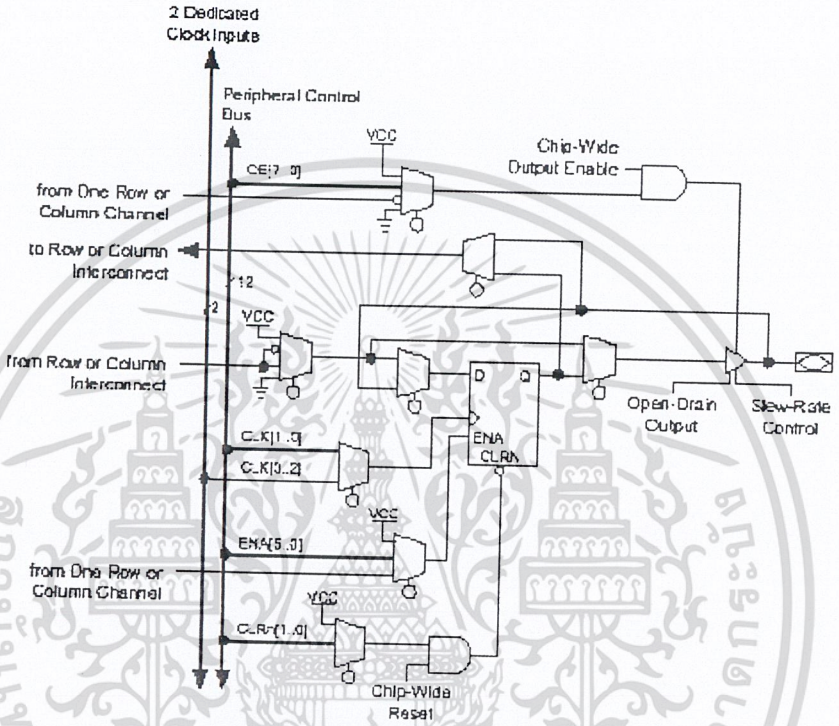


รูปที่ 2.53 แสดงโครงสร้างภายใน EAB

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.7.4 Input Out Element (IOE)

IOE จะถูกต่ออยู่กับขา I/O โดยจะประกอบด้วยส่วนของวงจรที่เป็น Tri State และส่วนที่เป็นฟลิปฟลอป ซึ่งเป็น option ดังแสดงในรูปที่ 2.54



รูปที่ 2.54 แสดงโครงสร้างภายในของ IOE

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3

การคำนวณและการสร้าง

จากทฤษฎีสามารถออกแบบ ตัวประมวลผลฟาสท์ฟูเรียร์ทรานส์ฟอร์ม โดย แบ่งออกเป็น 4 ส่วนใหญ่ๆ ดังนี้

1. ส่วนของอินพุท ซึ่งทำการแปลงสัญญาณอินพุทอะนาล็อก (Analog Signal) ให้อยู่ในรูปของสัญญาณดิจิตอลเพื่อนำไปประมวลผล จะประกอบไปด้วยวงจร ADC (Analog to Digital Converter) วงจรดีมัลติเพล็กซ์ (Demultiplex) เพื่อป้อนข้อมูลให้แก่ ตัวประมวลผลฟาสท์ฟูเรียร์ทรานส์ฟอร์ม

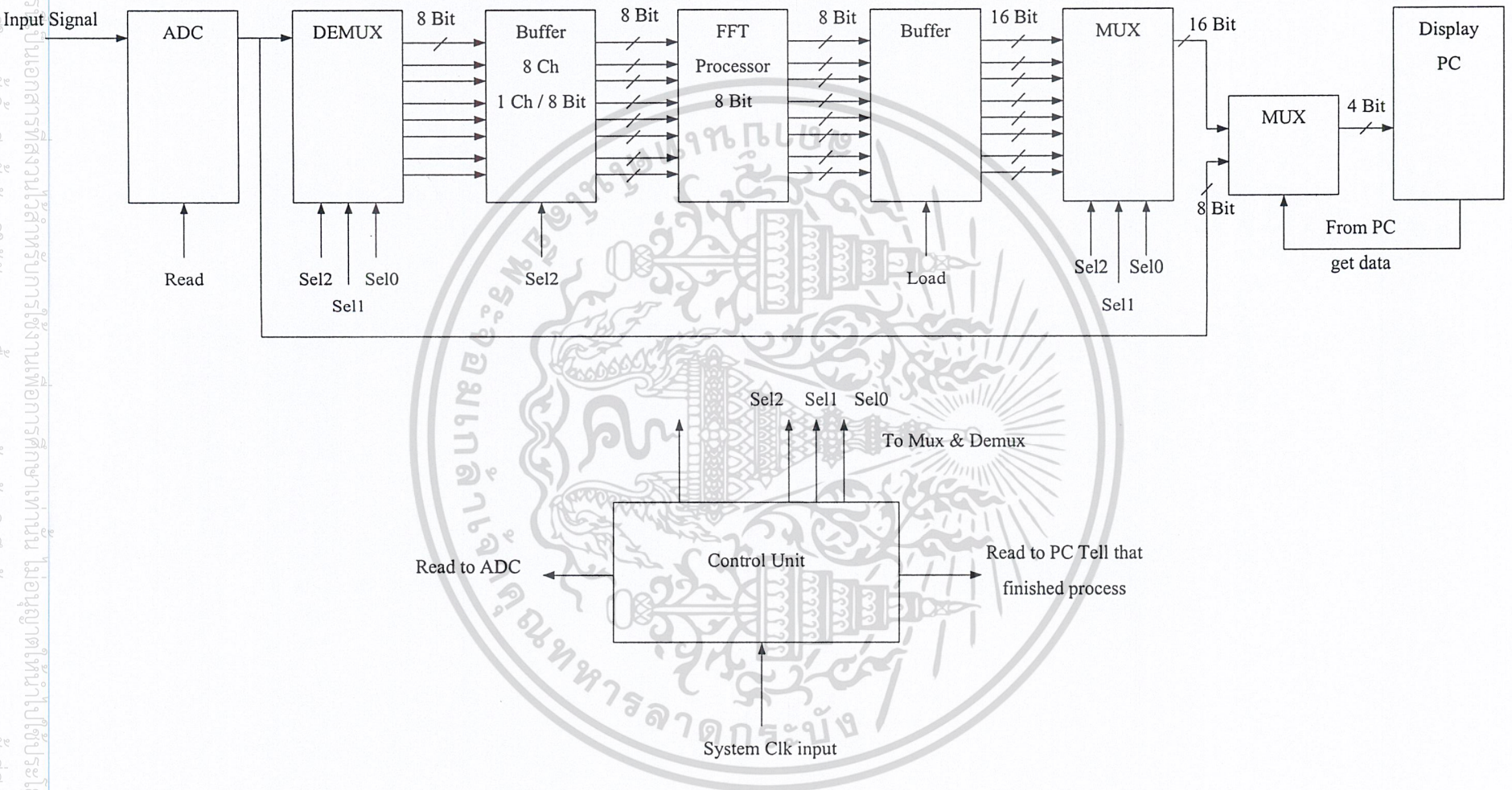
2. ส่วนของตัวประมวลผลฟาสท์ฟูเรียร์ทรานส์ฟอร์ม (FFT Processor) ทำการประมวลผลสัญญาณ ดิจิตอล อินพุท (Digital input) ที่ได้จากส่วนอินพุทให้อยู่ในรูปของ Discrete Spectrum

3. ส่วนของเอาท์พุท จะประกอบไปด้วยวงจร มัลติเพล็กซ์ (Multiplex) ผลที่ได้ออกไปทำการแสดงผลบนจอภาพของคอมพิวเตอร์ผ่านพอร์ทขนาน (Parallel Port)โดยการเขียนโปรแกรมคำนวณค่า Magnitude และ Phase โดยใช้ภาษา C (C Program)

4. ส่วนควบคุมการทำงาน Control Unit เป็นการควบคุมการป้อนข้อมูลอินพุท และ เอาท์พุทไปแสดงผล ซึ่งอยู่ในรูปของตัวกำเนิดสัญญาณควบคุมวงจร ADC, วงจร Multiplex, วงจร Demultiplex ให้การทำงานแบบเวลาจริง (Real Time)

ซึ่งส่วนต่างๆ สามารถเขียน บล็อกไดอะแกรมได้ดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



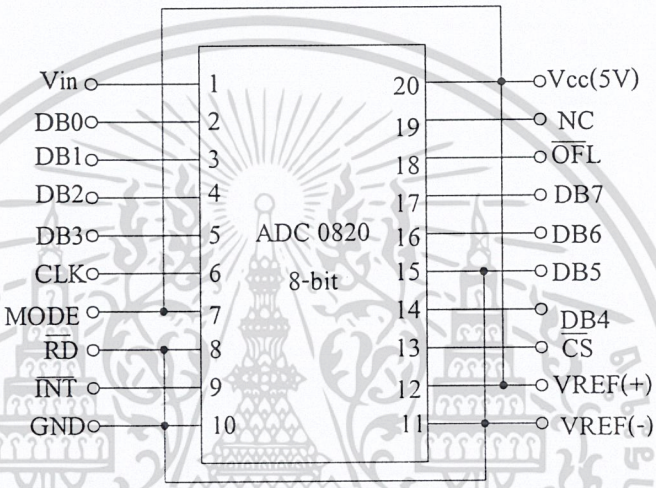
รูปที่ 3.1 บล็อกไดอะแกรมของ ตัวประมวลผลฟาสท์ฟูเรียร์ทรานส์ฟอร์ม

การออกแบบ

3.1 ส่วนอินพุท (Input Part)

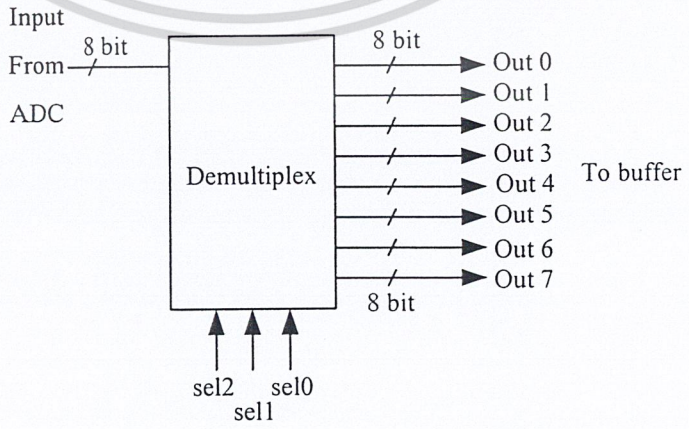
ประกอบไปด้วยส่วนของวงจรแปลงสัญญาณอะนาล็อกเป็นสัญญาณดิจิตอล (A/D Converter) วงจรดีมัลติเพล็กซ์ (Demultiplexer) และวงจรบัฟเฟอร์ (Buffer) ซึ่งมีการออกแบบดังนี้

3.1.1 วงจรแปลงสัญญาณอะนาล็อกเป็นสัญญาณดิจิตอล (A/D Converter) จะใช้ ไอซีเบอร์ ADC 0820 ซึ่งจะทำการแปลงระดับสัญญาณ ขนาด 8 บิต ระดับสัญญาณสามารถแปลงได้ ทั้ง สัญญาณบวก (สูงสุด +5 โวลท์) และสัญญาณลบ (ต่ำสุด -5 โวลท์) ดังวงจรข้างล่างนี้



รูปที่ 3.2 วงจรแปลงสัญญาณอะนาล็อกเป็นสัญญาณดิจิตอล (A/D Converter)

3.1.2 วงจรดีมัลติเพล็กซ์ (Demultiplex) ออกแบบเพื่อทำการเรียงข้อมูลจากวงจรแปลงสัญญาณอะนาล็อกเป็นสัญญาณดิจิตอล (ADC) เพื่อป้อนเข้าสู่ภาคของโปรเซสเซอร์ (FFT Processor) โดยมีการทำงานดังต่อไปนี้



รูปที่ 3.3 บล็อกไดอะแกรมของวงจร ดีมัลติเพล็กซ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

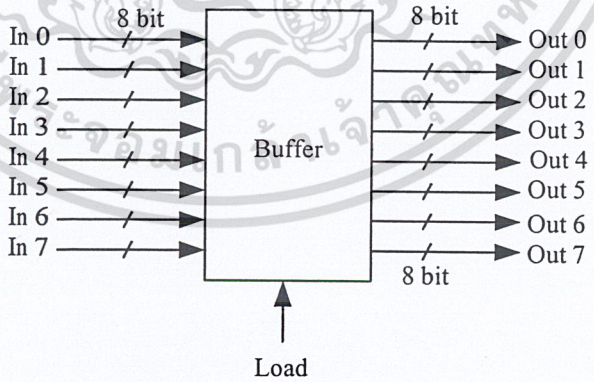
เป็นการรับข้อมูล 8 บิต เข้ามาวางทางด้านซ้ายมือมาวางทางด้านขวามือ 8 จุด จุดละ 8 บิต โดยการควบคุมขา Sel2, sel1, sel0 จากภาคกำเนิดสัญญาณควบคุม มีการทำงานดังต่อไปนี้

ตารางที่ 3.1 ตารางแสดงสัญญาณควบคุมการทำงาน

Sel2	Sel1	Sel0	Out 0	Out 1	Out 2	Out 3	Out 4	Out 5	Out 6	Out 7
0	0	0	Input	Qn	Qn	Qn	Qn	Qn	Qn	Qn
0	0	1	Qn	Input	Qn	Qn	Qn	Qn	Qn	Qn
0	1	0	Qn	Qn	Input	Qn	Qn	Qn	Qn	Qn
0	1	1	Qn	Qn	Qn	Input	Qn	Qn	Qn	Qn
1	0	0	Qn	Qn	Qn	Qn	Input	Qn	Qn	Qn
1	0	1	Qn	Qn	Qn	Qn	Qn	Input	Qn	Qn
1	1	0	Qn	Qn	Qn	Qn	Qn	Qn	Input	Qn
1	1	1	Qn	Qn	Qn	Qn	Qn	Qn	Qn	Input

จากรูปและตารางความจริงนำไปเขียนโปรแกรมทำการจำลองการทำงานและทดสอบต่อไป

3.1.3 วงจรบัฟเฟอร์ (Buffer) เป็นวงจรควบคุมให้ป้อนอินพุท (Input) จากวงจรดีมัลติ - เพล็กซ์ (Demultiplex) ให้กับตัวประมวลผล (processor) พร้อมๆกันทั้ง 8 จุด โดยมีการควบคุมจากสัญญาณโหลด (Load) จากส่วนกำเนิดสัญญาณควบคุม (Control Unit) ตามรูป



รูปที่ 3.4 บล็อกไดอะแกรมของวงจรบัฟเฟอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 3.2 ตารางแสดงการทำงานของวงจรบัพเฟอร์

Load	Output
0	Qn
1	Input

ทำการเขียนโปรแกรมให้มีการทำงานตามนี้ ทำการจำลองการทำงานและทดสอบ

3.2 ส่วนตัวประมวลผล (Processor Part)

3.2.1 แนวคิดในการออกแบบ

การคำนวณและการสร้างจะเริ่มต้นออกแบบที่สมการของการแปลงฟูเรียร์เต็มหน่วย (DFT) แล้วทำการกระจายผลบวก (summation) ในรูปของฐานสอง (radix 2) จะได้สมการการแปลงฟาสท์ฟูเรียร์ แล้วทำการแทนค่าลงไปจะได้ดังนี้

จาก DFT (Discrete Fourier)

$$NF(k) = \sum_{m=0}^{N-1} f(m) w^{km} \quad ;k = 0,1,2...N-1$$

$$F(k) = \frac{1}{N} \sum_{m=0}^{N-1} f(m) w^{km} \quad ;k = 0,1,2...N-1$$

$;\frac{1}{N}$ คิดเป็น Factorในการ Scaling จึงทำการละไว้

$$F(k) = \sum_{m=0}^{N-1} f(m) w^{km}$$

การแปลงฟูเรียร์เต็มหน่วย DFT 8 จุด N=8

$$F(k) = \sum_{m=0}^{7} f(m) w^{km}$$

แปลง FFT โดยทำการกระจายในรูปฐาน 2 (Radix 2)

$$F(k) = \sum_{m_0=0}^{1} \sum_{m_1=0}^{1} \sum_{m_2=0}^{1} f(m_2 m_1 m_0) w^{k(4m_2+2m_1+m_0)}$$

$$F(k) = \sum_{m_0=0}^{1} \left[\sum_{m_1=0}^{1} \left[\sum_{m_2=0}^{1} f(m_2 m_1 m_0) w^{4km_2} \right] w^{2km_1} \right] w^{km_0}$$

ให้ M_2

$$= \sum_{m_2=0}^{1} f(m_2 m_1 m_0) w^{4km_2}$$

กระจาย k ในรูปของเลขฐาน 2 (Radix 2)

$$= \sum_{m_2=0}^{1} f(m_2 m_1 m_0) w^{4m_2(4k_2+2k_1+k_0)}$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จาก $w = e^{-\frac{j2\pi}{8}}$; $w^4 = e^{-\frac{j2\pi(4)}{8}}$

$$= \cos\pi - j\sin\pi = (-1)$$

$$= \sum_{m_2=0}^{1} f(m_2, m_1, m_0)(-1)^{m_2(4k_2+2k_1+k_0)}$$

$$= f(0, m_1, m_0)(-1)^0 + f(1, m_1, m_0)(-1)^{4k_2+2k_1+k_0}$$

$$= f(0, m_1, m_0) + f(1, m_1, m_0)(-1)^{4k_2} \cdot (-1)^{2k} \cdot (-1)^{k_0}$$

$$= f(0, m_1, m_0) + f(1, m_1, m_0)(-1)^{k_0}$$

ให้ $k_0 = 0$

$$= f(0, m_1, m_0) + f(1, m_1, m_0)$$

แทนค่า $m_1 m_0$ ที่เป็นไปได้

$$F_{1,0} = f(000) + f(100) = f_0 + f_4$$

$$F_{1,1} = f(001) + f(101) = f_1 + f_5$$

$$F_{1,2} = f(010) + f(110) = f_2 + f_6$$

$$F_{1,3} = f(011) + f(111) = f_3 + f_7$$

$k_0 = 1$

$$F_{1,4} = f(000) + f(100)(-1) = f_0 - f_4$$

$$F_{1,5} = f(001) + f(101)(-1) = f_1 - f_5$$

$$F_{1,6} = f(010) + f(110)(-1) = f_2 - f_6$$

$$F_{1,7} = f(011) + f(111)(-1) = f_3 - f_7$$

$$M_1 = \sum_{m_1=0}^{1}\left[\sum_{m_2=0}^{1} f(m_2 m_1 m_0) w^{4km_2}\right] w^{2km_1}$$

$$= \sum_{m_2=0}^{1} \tilde{f}(k_0 m_1 m_2) w^{2km_1}$$

$$= f(k_0, m_1, m_0) w^0 + f(k_0, m_1, m_0) w^{2(4k_2+2k_1+k_0)}$$

$$= f(k_0, m_1, m_0) + f(k_0, m_1, m_0)(-j)^{2k_1+k_0}$$

$k_0 = 0, k_1 = 0$

$$F_{2,0} = \tilde{f}(000) + \tilde{f}(010) = \tilde{f}(0) + \tilde{f}(2)$$

$$F_{2,1} = \tilde{f}(001) + \tilde{f}(011) = \tilde{f}(1) + \tilde{f}(3)$$

$k_0 = 1, k_1 = 0$

$$F_{2,2} = \tilde{f}(000) + \tilde{f}(010)(-j)^2 = \tilde{f}(0) - \tilde{f}(2)$$

$$F_{2,3} = \tilde{f}(001) + \tilde{f}(011)(-j)^2 = \tilde{f}(1) - \tilde{f}(3)$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$k_0 = 1, k_1 = 0$

$$F_{2,4} = \tilde{f}(100) + \tilde{f}(110) = \tilde{f}(4) - j\tilde{f}(6)$$

$$F_{2,5} = \tilde{f}(101) + \tilde{f}(111) = \tilde{f}(5) - j\tilde{f}(7)$$

$k_0 = 1, k_1 = 1$

$$F_{2,6} = \tilde{f}(100) + \tilde{f}(110)(-j)^3 = \tilde{f}(4) + j\tilde{f}(6)$$

$$F_{2,7} = \tilde{f}(101) + \tilde{f}(111)(-j)^3 = \tilde{f}(5) + j\tilde{f}(7)$$

$$M_1 = \sum_{m_0=0}^{1} \tilde{\tilde{f}}(k_0 k_1 m_0) w^{km_0}$$

$$= \tilde{\tilde{f}}(k_0, k_1, 0) w^0 + \tilde{\tilde{f}}(k_0, k_1, 1) w^{4k_2 + 2k_1 + k_0}$$

$k_0 = 0, k_1 = 0, k_2 = 0$

$$F_{3,0} = \tilde{\tilde{f}}(000) + \tilde{\tilde{f}}(001) w^0 = \tilde{\tilde{f}}(0) + \tilde{\tilde{f}}(1)$$

$k_0 = 0, k_1 = 0, k_2 = 1$

$$F_{3,1} = \tilde{\tilde{f}}(000) + \tilde{\tilde{f}}(001) w^4 = \tilde{\tilde{f}}(0) - \tilde{\tilde{f}}(1)$$

$k_0 = 0, k_1 = 1, k_2 = 0$

$$F_{3,2} = \tilde{\tilde{f}}(010) + \tilde{\tilde{f}}(011) w^2 = \tilde{\tilde{f}}(2) - j\tilde{\tilde{f}}(3)$$

$k_0 = 0, k_1 = 1, k_2 = 1$

$$F_{3,3} = \tilde{\tilde{f}}(010) + \tilde{\tilde{f}}(011) w^6 = \tilde{\tilde{f}}(2) + \tilde{\tilde{f}}(3)$$

$k_0 = 1, k_1 = 0, k_2 = 0$

$$F_{3,4} = \tilde{\tilde{f}}(010) + \tilde{\tilde{f}}(101) w^1 = \tilde{\tilde{f}}(4) + \left(\frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2} j \right) \tilde{\tilde{f}}(5)$$

$k_0 = 1, k_1 = 0, k_2 = 1$

$$F_{3,5} = \tilde{\tilde{f}}(100) + \tilde{\tilde{f}}(101) w^5 = \tilde{\tilde{f}}(4) + \left(-\frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2} j \right) \tilde{\tilde{f}}(5)$$

$k_0 = 1, k_1 = 1, k_2 = 0$

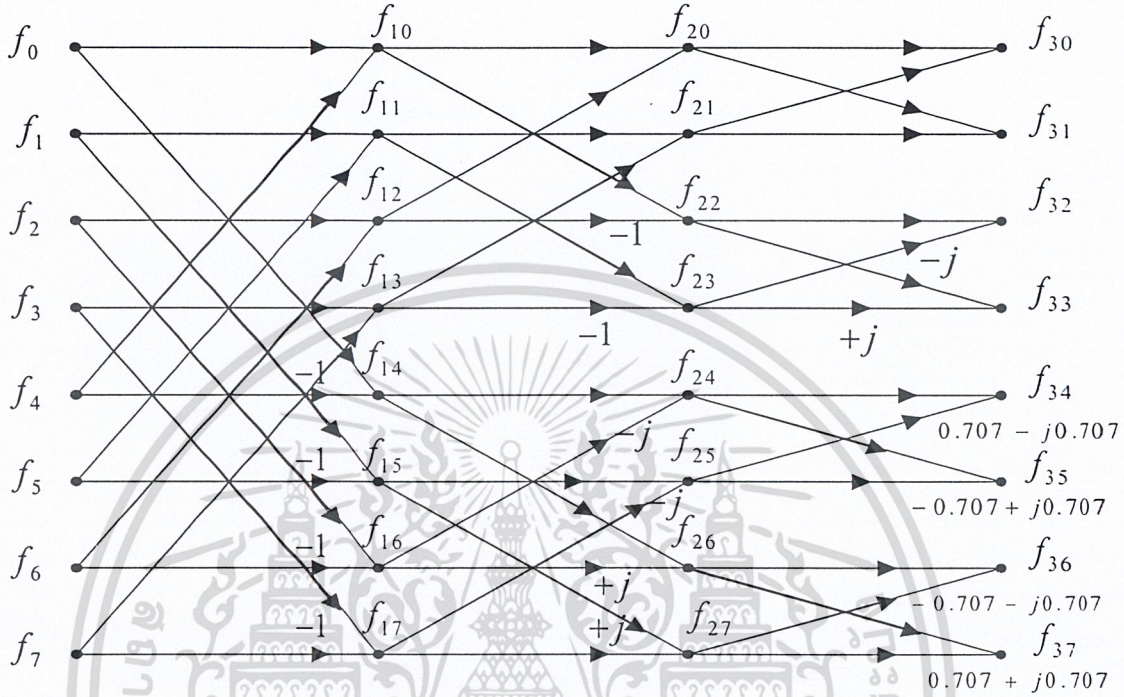
$$F_{3,6} = \tilde{\tilde{f}}(110) + \tilde{\tilde{f}}(111) w^3 = \tilde{\tilde{f}}(6) + \left(-\frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2} j \right) \tilde{\tilde{f}}(7)$$

$k_0 = 1, k_1 = 1, k_2 = 1$

$$F_{3,7} = \tilde{\tilde{f}}(110) + \tilde{\tilde{f}}(111) w^7 = \tilde{\tilde{f}}(6) + \left(\frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2} j \right) \tilde{\tilde{f}}(7)$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากการกระจาย การแปลงฟาสท์ฟูเรียร์ ตามสมการที่ผ่านมาสามารถที่จะเขียนเป็นแผนภูมิผีเสื้อ (Butterfly diagram) ตามรูปที่ 3.5

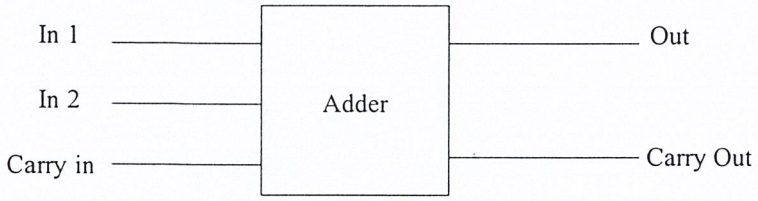


รูปที่ 3.5 แผนภูมิผีเสื้อ เมื่อทำการแทนค่า Twiddle factor ลงไปของการแปลงฟาสท์ฟูเรียร์ 8 จุด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากบล็อกไดอะแกรมนำมาออกแบบแต่ละดังนี้

3.2.2 การออกแบบวงจรบวก(Adder)



รูปที่ 3.7 บล็อกไดอะแกรมของวงจรบวก

ในการบวกข้อมูล 8 bit เราจะได้ Truth table ตามตาราง

ตารางที่ 3.3 ตารางความจริงของการออกแบบวงจรบวก

In1	In2	Carry in	out	Carry out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

ทำการ Map output และ Carry out

Out	In 2 / Carry in			
in1	00	01	11	10
0	0	(1)	0	(1)
1	(1)	0	(1)	0

$$\text{Out} = in1 \cdot \overline{in2} \cdot \overline{Ci} + \overline{in1} \cdot \overline{in2} \cdot Ci + in1 \cdot in2 \cdot Ci + \overline{in1} \cdot in2 \cdot \overline{Ci}$$

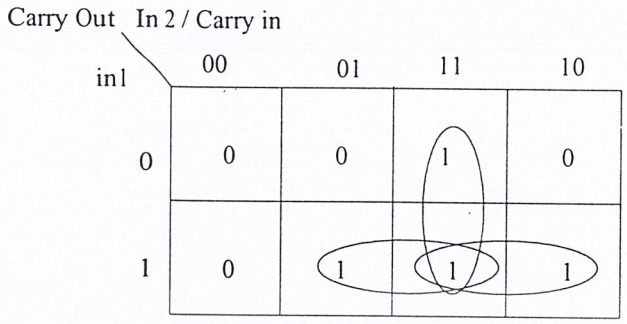
$$= (in1 \cdot \overline{in2} + \overline{in1} \cdot in2) \cdot \overline{Ci} + (\overline{in1} \cdot \overline{in2} + in1 \cdot in2)Ci$$

$$= (in1 \oplus in2)\overline{Ci} + (\overline{in1 \oplus in2})Ci$$

$$\text{Out} = in1 \oplus in2 \oplus Ci$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

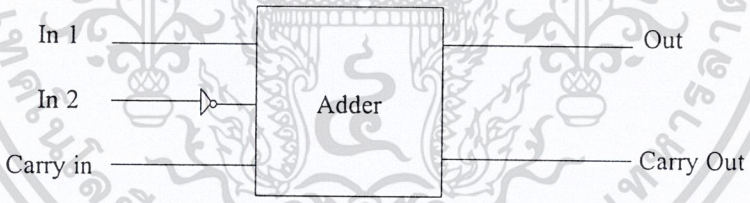
Carry out $= in2 \cdot Cin + in1 \cdot 1Cin + in1 \cdot in2$



ทำการเขียนโปรแกรมโดยทำการวน Loop ให้ครบ 8 bit ทำให้เป็น Adder 8 bit โดยการ Initial carry in ในการบวกบิทแรกให้เป็น " 0 " ในบิทต่อไปทำการต่อ carry out เข้ากับ carry in ทำการวน loop จนครบ 8 bit ให้ได้ output Adder 8 bit

3.2.3 การออกแบบวงจรลบ (Subtractor)

จะเห็นว่าการออกแบบวงจรลบหรือ Subtractor ก็คือการบวกด้วยจำนวนลบนั่นเอง ดังนั้นสามารถปรับวงจรบวก (Adder) ให้เป็นวงจรลบ (Subtractor) ได้โดยการทำการแปลงตัวลบ ให้อยู่ในรูป 2's complement แล้วทำการบวก ในการทำ 2's complement ก็คือการกลับบิท(inverse) ทุกบิทแล้วบวกด้วย 1 สามารถปรับปรุง วงจร Adder ได้ดังนี้



รูปที่ 3.8 บล็อกไดอะแกรมของวงจรลบ

แต่ในการกำหนดค่า(Initial) carry in ในการลบบิทแรกต้องให้เท่ากับ " 1 "

การทำการกลับเครื่องหมาย ก็คือการทำกลับบิท (Inverse)ทุกบิทแล้วทำการบวกด้วย 1 เช่น

1001 1001 = 0110 0110

\+

1

2's complement ⟶ 0110 0111 เป็นต้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2.4 วงจรกลับค่าตัวเลข (2' s complement)

จะเห็นว่าเป็นการหาค่าลบหรือ 0 – (A) ; A คือเลขฐานสอง (binary) ใดๆอาจจะเป็นจำนวนลบหรือจำนวนก็ได้ ก็สามารถประยุกต์ปรับปรุงวงจรรวมโดยให้ตัวตั้งเป็น 0 หมดทุกบิทเมื่อทำการลบด้วยจำนวน (in2) ก็จะได้เลข 2's complement ของข้อมูล in2

in1 = 0000 0000

in2 = xxxx xxxx

Output = 2's complement (in2)

3.2.5 ตัวคูณจำนวนเชิงซ้อน (Complex multiplier)

เป็นการประยุกต์ใช้การคูณโดยหลักการของเลขคณิตกระจาย (Distribute Arithmetic) มาใช้เพื่อเพิ่มความเร็วในการคูณ ให้อินพุท(input)เป็น $(X_r + jX_i)$ ทำการคูณกับค่าของจำนวนเชิงซ้อน (complex number) $(A + jB)$ ซึ่งเป็นค่า twiddle factor ในแผนภูมิผีเสื้อ (Butterfly diagram) ดังนั้นในการคูณจะได้

$$\begin{aligned} y &= (A_r + jA_i)(X_r + jX_i) = y_r + y_i \\ &= A_r X_r + jA_r X_i + jA_i X_r + j^2 A_i X_i \\ &= (A_r X_r - A_i X_i) + j(A_i X_r + A_i X_r) \\ &= y_r + y_i \end{aligned}$$

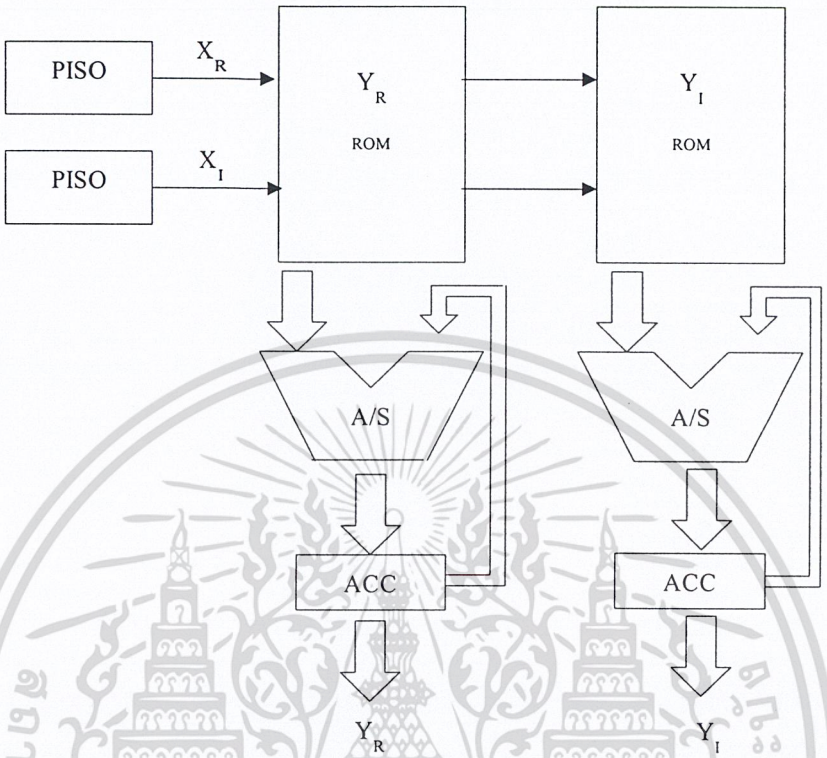
จากสมการสามารถเขียนเป็นตารางเมื่อ อินพุท (Input) ในส่วนจริง X_r และส่วนจินตภาพ X_i มีค่าต่างๆ

ตารางที่ 3.4 ตารางผลคูณจำนวนเชิงซ้อน(Complex multiply) เมื่อป้อนค่า Input X_r, X_i

X_r	X_i	Y_r	Y_I
0	0	0	0
0	1	$-A_I$	A_r
1	1	A_r	A_I
1	0	$A_r - A_i$	$A_r + A_i$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในการออกแบบสามารถประยุกต์การคูณโดยใช้เลขคณิตกระจาย (Distribute Arithmetic) ได้ Hardware ดังนี้

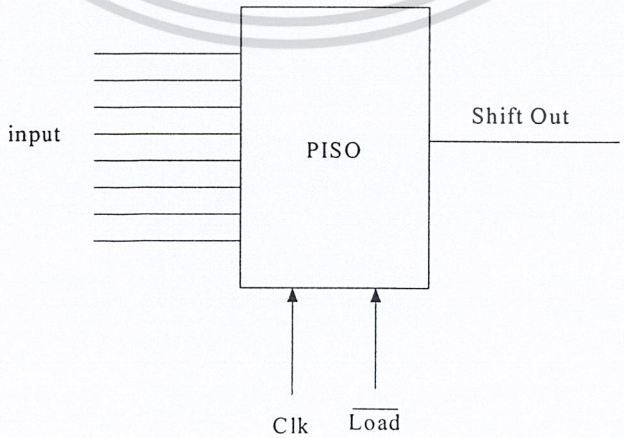


รูปที่ 3.9 Block Diagram ของวงจรคูณจำนวนเฉพาะ (Complex multiplier) ตามหลักการเลขคณิตกระจาย

ทำการเก็บค่าผลลัพธ์ส่วนจริง Y_r และส่วนจินตภาพ Y_i ที่ได้จากตารางที่ 3.4 ในหน่วยความจำส่วนจริง(ROM Y_r) และส่วนจินตภาพ (Y_i ROM) ตามลำดับ

ส่วนประกอบของวงจรคูณจำนวนเชิงซ้อน(Complex Multiplier) โดยหลักการของเลขคณิตกระจาย (Distribute Arithmetic) มีดังต่อไปนี้

3.2.5.1วงจรเลื่อนข้อมูลแบบเข้าขนานออกอนุกรม(Parallel In Serial Out (PISO))



รูปที่ 3.10 บล็อกไดอะแกรมของ วงจร Parallel In Serial Out Shift Register

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การทำงานของวงจร เมื่อมีสัญญาณอินพุท (Input) เข้ามา และสัญญาณที่ขา $\overline{Load}$ Active จะทำการ *Load* ข้อมูลเข้ามาเลขเก็บในชิพรีจีสเตอร์(Shift Register) เมื่อป้อนสัญญาณนาฬิกา (Clock) ก็จะทำให้รีจีสเตอร์ (register) ทำการเลื่อน ข้อมูลออกเอาท์พุท(output) ทีละบิต เริ่มจากบิทที่มีนัยสำคัญต่ำสุด (LSB) ตามสัญญาณนาฬิกา (Clock)

3.2.5.2 หน่วยความจำ (ROM)

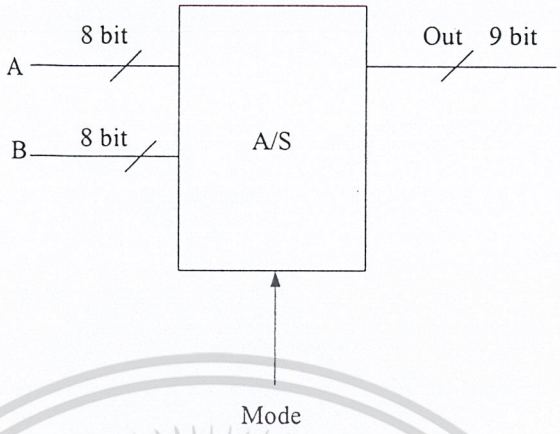
ในการสร้างหน่วยความจำ(ROM) และเก็บค่าลงในหน่วยความจำนั้นสามารถสร้างโดยการเขียนโปรแกรมในรูปของ Package Rom ซึ่งมีรูปแบบเฉพาะ ซึ่งโปรแกรมจะทำการสร้างแอดเดรส (Address generator) และ เก็บข้อมูลตามที่เราเขียนขึ้น ซึ่งสามารถอ่านค่าจากหน่วยความจำที่เขียนขึ้นได้ ในการออกแบบตัวประมวลผลฟาสท์ฟูเรียร์ทรานส์ฟอร์มฟูเรียร์(FFT Processor) 8 จุด จะใช้ตัวคูณจำนวนเชิงซ้อน(Complex multiplier) จำนวน 2 ตัว โดยแต่ละตัวจะมีค่าที่เก็บในหน่วยความจำ(ROM) ต่างกัน ซึ่งค่าสัมประสิทธิ์ที่จะนำไปเก็บ A_r และ A_i ต่างกัน ดังนั้นในตัวคูณจำนวนเชิงซ้อนที่ 1 (Complex multiplier 1) จะมี หน่วยความจำ(ROM1) เก็บค่า Y_r และ Y_i ซึ่งค่าA_r และ A_i จะมีค่า (0.707-j 0.707) แทนค่าลงในตารางผลคูณของ Complex multiply แล้วทำการแปลงให้อยู่ในรูปเลขทศนิยมฐานสอง คิดเครื่องหมาย เก็บไว้ใน ROM1 และ ROM 2 ส่วน Complex multiplier ตัวที่ 2 ใช้ค่าสัมประสิทธิ์ (-0.707 –j 0.707) แทนค่าในตารางผลคูณ Complex multiply แล้วแปลงให้อยู่ในรูปของทศนิยมฐาน คิดเครื่องหมาย แล้วเก็บลง ROM3 และ ROM4 เช่นกัน ดังแสดงในตารางต่อไปนี้

ตารางที่ 3.5 Address และค่าที่เก็บใน ROM แต่ละตัว

		Complex multiplier 1		Complex multiplier 2	
		ROM 1	ROM 2	ROM 3	ROM 4
X_r	X_I	Y_r	Y_I	Y_r	Y_i
0	0	0000 0000	0000 0000	0000 0000	0000 0000
0	1	0010 1101	0010 1101	0010 1101	1101 0011
1	0	0010 1101	1101 0011	1101 0011	1101 0011
1	1	0101 1010	0000 0000	0000 0000	1010 0110

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2.5.3 วงจรบวกหรือวงจรลบ (Adder/Subtractor)

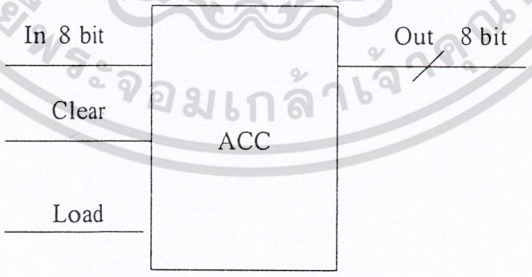


รูปที่ 3.11 บล็อกไดอะแกรมของ วงจรบวกหรือวงจรลบ

ในส่วนของวงจรบวกหรือลบ (Adder & Subtractor) คือ วงจรบวกและวงจรลบที่จะรับค่าที่อ่านจาก ROM เข้ามาทำการบวกหรือลบกับค่าใน Accumulator ซึ่งสามารถเลือกการบวกหรือลบได้จากขา mode ผลลัพธ์ (output) จะมี 9 บิท บิทที่มีนัยสำคัญสูงสุด MSB หรือบิทที่ 9 จะเป็น บิทเช็คเครื่องหมาย เพื่อนำไปเลื่อนแทนที่ในการเลื่อนขวาของข้อมูลในวงจรเก็บค่า (Accumulator)

3.2.5.4 ส่วนเก็บค่าผลการคำนวณ (Accumulator)

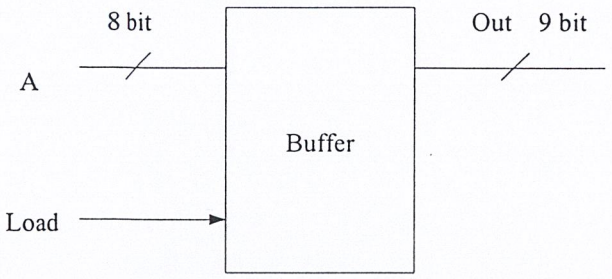
มีหน้าที่ ทำการ load ค่า output ที่ได้จากการบวกหรือลบ (A/S) เข้ามาเก็บแล้วทำการเลื่อนขวา 1 บิท แล้วส่งกลับไปบวกหรือลบกับค่าต่อไปในส่วนของ A/S ตามรูป Block diagram



รูปที่ 3.12 บล็อกไดอะแกรมของ Accumulator

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

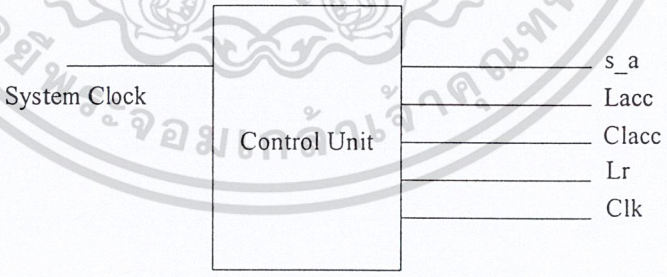
3.2.5.5 บัฟเฟอร์ (Buffer) ทำหน้าที่เก็บผลลัพธ์ของการคูณไว้โดยทำการ load ข้อมูลออกมาเพื่อมีสัญญาณ load



รูปที่ 3.13 บล็อกไดอะแกรมของ Buffer

3.2.5.6 ส่วนกำเนิดสัญญาณควบคุม (Control Unit)

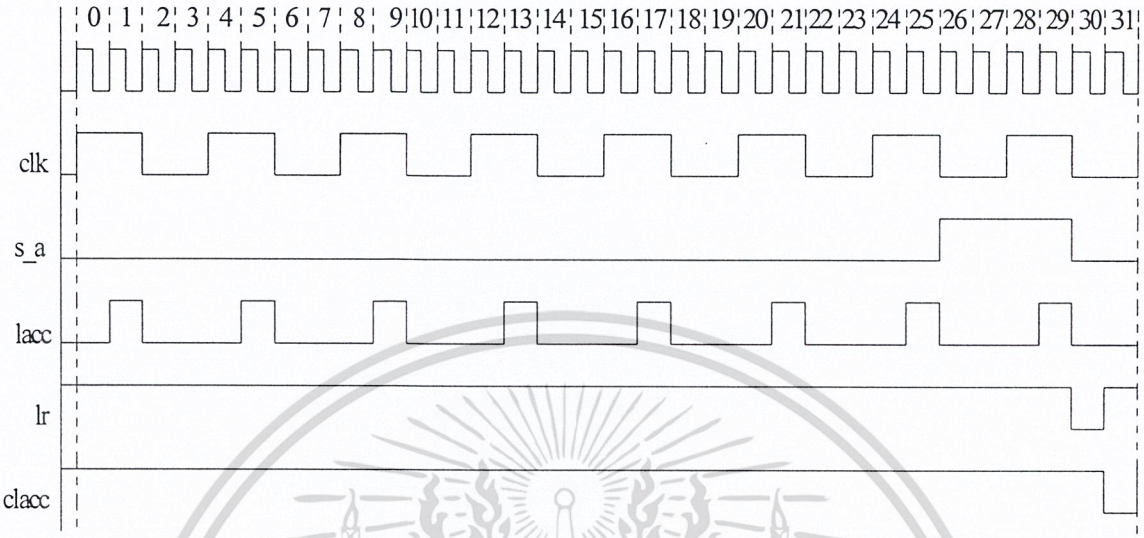
เป็นส่วนที่กำหนดสัญญาณควบคุมส่วนประกอบของวงจรต่างๆ ซึ่งสร้างมาจากสัญญาณนาฬิการะบบ(System clock)สัญญาณที่ออกจากวงจรนี้มีอยู่สัญญาณคือ s_a เป็นสัญญาณควบคุมเลือก mode การบวกหรือลบของวงจรบวกหรือลบ (Adder / Subtractor) , สัญญาณ lacc (load accumulator) ทำการ load สัญญาณเข้าวงจรเก็บค่าการคำนวณ(Accumulator) เพื่อทำการเลื่อนสัญญาณ, clacc (clear accumulator) ทำการ clear accumulator เมื่อ X_i , X_r ค่าใหม่เข้ามา (8 บิท) , สัญญาณ lr คือสัญญาณที่ไปควบคุม 2 วงจร คือ PISO เพื่อควบคุมการ load ข้อมูลเข้า Register และ Buffer เพื่อนำข้อมูลออกเอาท์พุท พร้อมๆกันและสุดท้าย,สัญญาณ clock คือ สัญญาณที่ทำการเลื่อน(shift)ข้อมูลใน Register เพื่อชี้ค่า Address เพื่อทำการอ่านผลคูณในหน่วยความจำ(ROM)



รูปที่ 3.14 บล็อกไดอะแกรมของ สัญญาณควบคุม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในการออกแบบภาค Control Unit จะออกแบบโดยใช้ states – machine โดยการกำหนดการทำงานของแต่ละสัญญาณในช่วงเวลาต่างๆตามไดอะแกรมเวลา(timing diagram)



รูปที่ 3.15 แสดงสัญญาณที่ใช้ออกแบบส่วนควบคุม

A เป็น state ที่ Clk,s_a,Lr,Lacc,Clacc มีค่า 10011

B เป็น state ที่ Clk,s_a,Lr,Lacc,Clacc มีค่า 10111

C เป็น state ที่ Clk,s_a,Lr,Lacc,Clacc มีค่า 00011

D เป็น state ที่ Clk, s_a, Lr, Lacc, Clacc มีค่า 01011

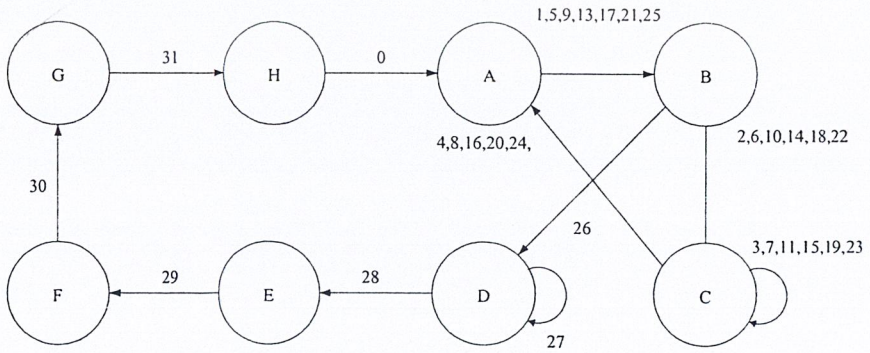
E เป็น state ที่ Clk, s_a, Lr, Lacc, Clacc มีค่า 11011

F เป็น state ที่ Clk, s_a, Lr, Lacc, Clacc มีค่า 11111

G เป็น state ที่ Clk, s_a, Lr, Lacc, Clacc มีค่า 00001

H เป็น state ที่ Clk, s_a, Lr, Lacc, Clacc มีค่า 00010

ได้ State – diagram ดังนี้



รูปที่ 3.16 แสดง State – diagram ในการกำเนิดสัญญาณต่างๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

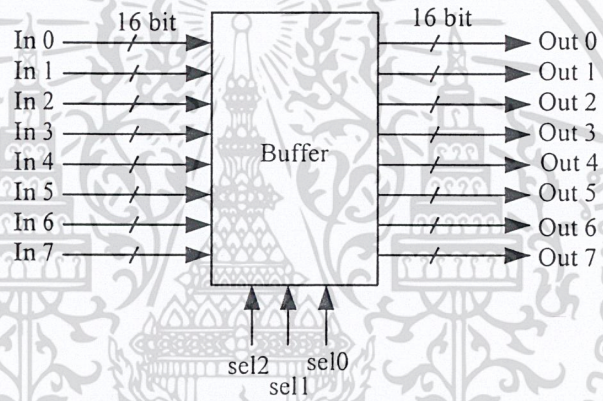
จากไดอะแกรมสภาวะ (State Diagram)นำไปเขียนโปรแกรมกำเนิดสัญญาณไปควบคุมวงจรต่างๆ

เมื่อทำการออกแบบในทุกส่วนเรียบร้อยแล้วทำการรวมทางซอฟแวร์ (Port map) โดยการเรียกใช้อุปกรณ์ที่ทำการเขียนขึ้นมา ให้อยู่ในรูปตัวคูณจำนวนเชิงซ้อน แล้วทำการรวมอุปกรณ์ย่อยต่างๆให้อยู่แบบตามบล็อกไดอะแกรม แล้วทำการทดสอบต่อไป

3.3 ส่วนของเอาท์พุท (Output Part)

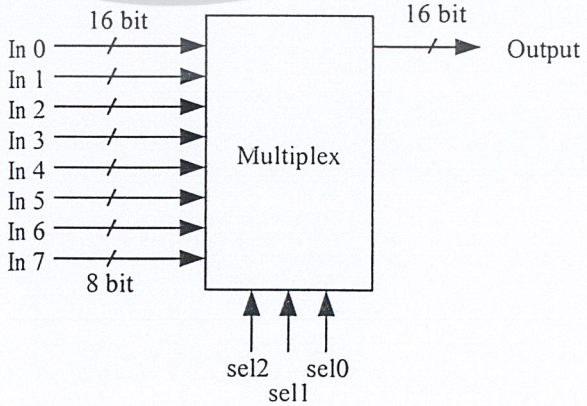
ในส่วนนี้จะประกอบด้วยวงจร บัพเฟอร์(Buffer) และวงจร มัลติเพล็กซ์ (Multiplex) และส่วนของภาคแสดงผลทางจอภาพคอมพิวเตอร์โดยใช้ภาษาซี (C Programming)

3.3.1 วงจรบัพเฟอร์ (Buffer) ก็เหมือนกับการออกแบบวงจร บัพเฟอร์ (Buffer) ในส่วนของอินพุท แต่เป็นข้อมูล 16 บิต เท่านั้น ทำหน้าที่ โหลด ผลลัพธ์ที่ได้จากการคำนวณออกจาก ตัวประมวลผลพร้อมๆ กัน เพื่อทำการ มัลติเพล็กซ์ไปยังส่วนแสดงผลผ่านทางคอมพิวเตอร์



รูปที่ 3.17 บล็อกไดอะแกรมของวงจรบัฟเฟอร์

3.3.2 วงจรมัลติเพล็กซ์ (Multiplex) เป็นการ มัลติเพล็กซ์ ผลลัพธ์ได้ออกไป เพื่อแสดงโดยการมัลติเพล็กซ์ควบคุมจากสัญญาณ sel2, sel1, sel0 จากส่วนควบคุม กำเนิดสัญญาณควบคุม (Control Unit) จะมีลักษณะดังต่อไปนี้



รูปที่ 3.18 บล็อกไดอะแกรมของวงจรมัลติเพล็กซ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สามารถเขียนตารางความจริงได้ดังนี้

ตารางที่ 3.6 ตารางความจริงของวงจรมัลติเพล็กซ์

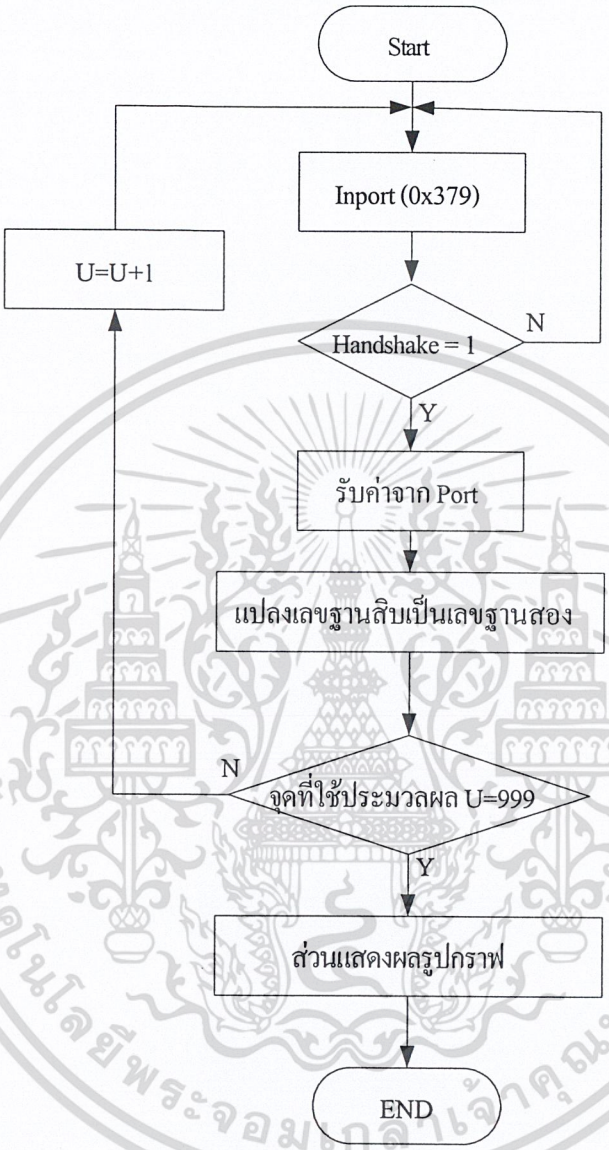
Sel2	Sel1	Sel0	Output
0	0	0	Input 0
0	0	1	Input 1
0	1	0	Input 2
0	1	1	Input 3
1	0	1	Input 4
1	0	1	Input 5
1	1	0	Input 6
1	1	1	Input 7

ทำการเขียนโปรแกรมให้มีการทำงานดังนี้ แล้วจำลองการทำงานและทดสอบ

3.3.3 ส่วนของการแสดงผล (Display) เป็นการนำค่าที่ได้จากการคำนวณไปแสดงผลในรูปกราฟสัญญาณอินพุท (Input Signal) กราฟของขนาด (Magnitude) และกราฟของเฟส (Phase) โดยการเขียนโปรแกรมภาษาซี (C Programming) ส่งค่าข้อมูลจากพอร์ทขนาน (Parallel Port) ดัง โฟลว์ชาร์ตต่อไปนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

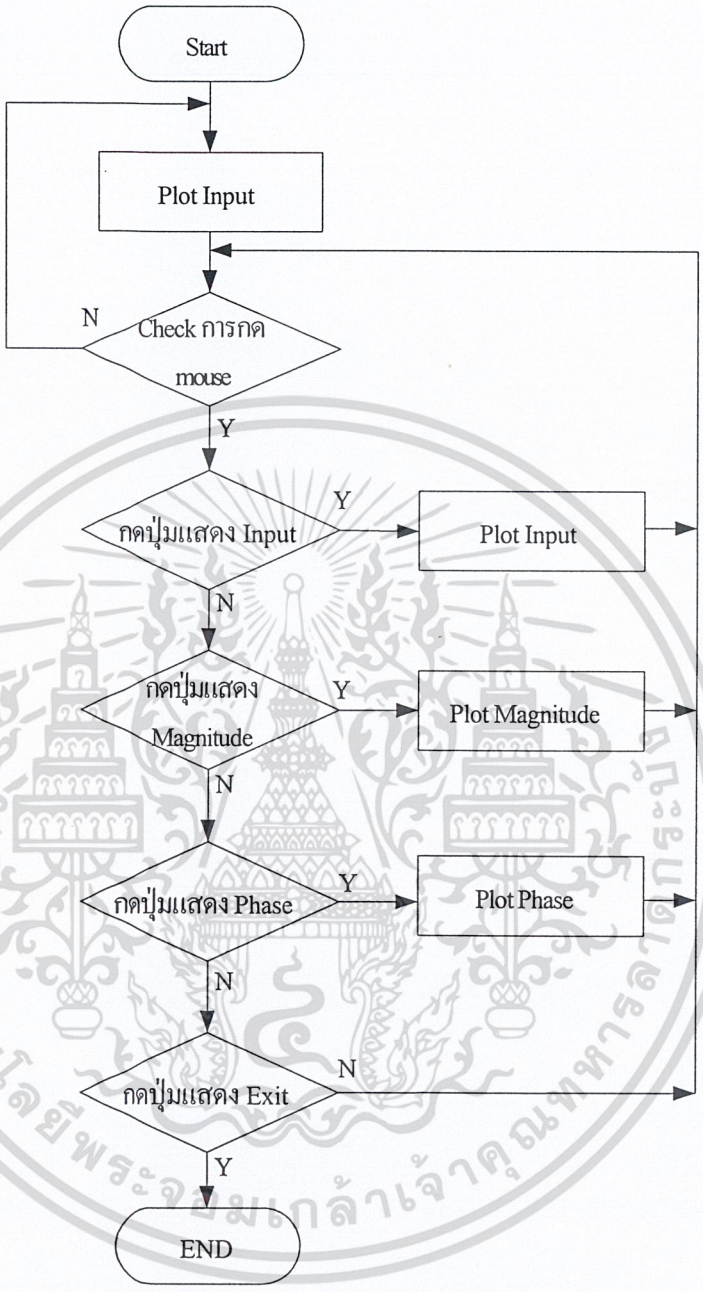
โฟลว์ชาร์ตหลักของส่วนแสดงผล



รูปที่ 3.19 โฟลว์ชาร์ตการทำงานของส่วนการแสดงผล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

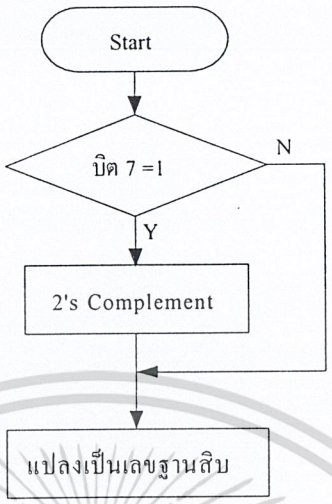
โฟลว์ชาร์ตของส่วนการพล็อตกราฟ



รูปที่ 3.20 โฟลว์ชาร์ตการทำงานของส่วนพล็อตกราฟ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โฟลว์ชาร์ตของส่วนการแปลงเลขฐานสองเป็นเลขฐานสิบ



รูปที่ 3.21 โฟลว์ชาร์ตการทำงานของส่วนการแปลงเลขฐานสองเป็นเลขฐานสิบ

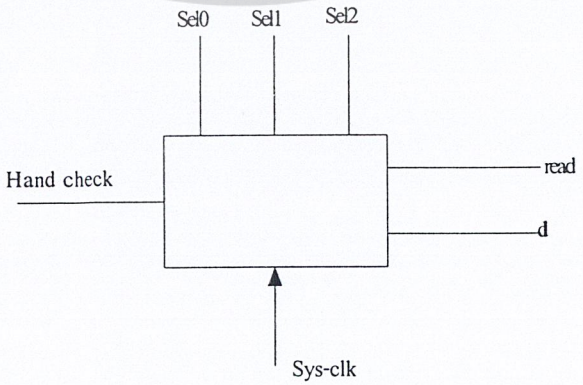
3.4 ส่วนกำเนิดสัญญาณควบคุม (Control Part)

เป็นส่วนที่ออกแบบเพื่อกำเนิดสัญญาณควบคุมวงจรต่างๆ ในส่วนต่างๆ เพื่อให้สามารถทำงานประมวลผลในเวลาจริงได้ (Real Time) ซึ่งสามารถเขียนได้ดังนี้

สัญญาณ Load ควบคุมการ Load สัญญาณ input เข้าส่วนประมวลผลและ Output ออกจากส่วนประมวลผล

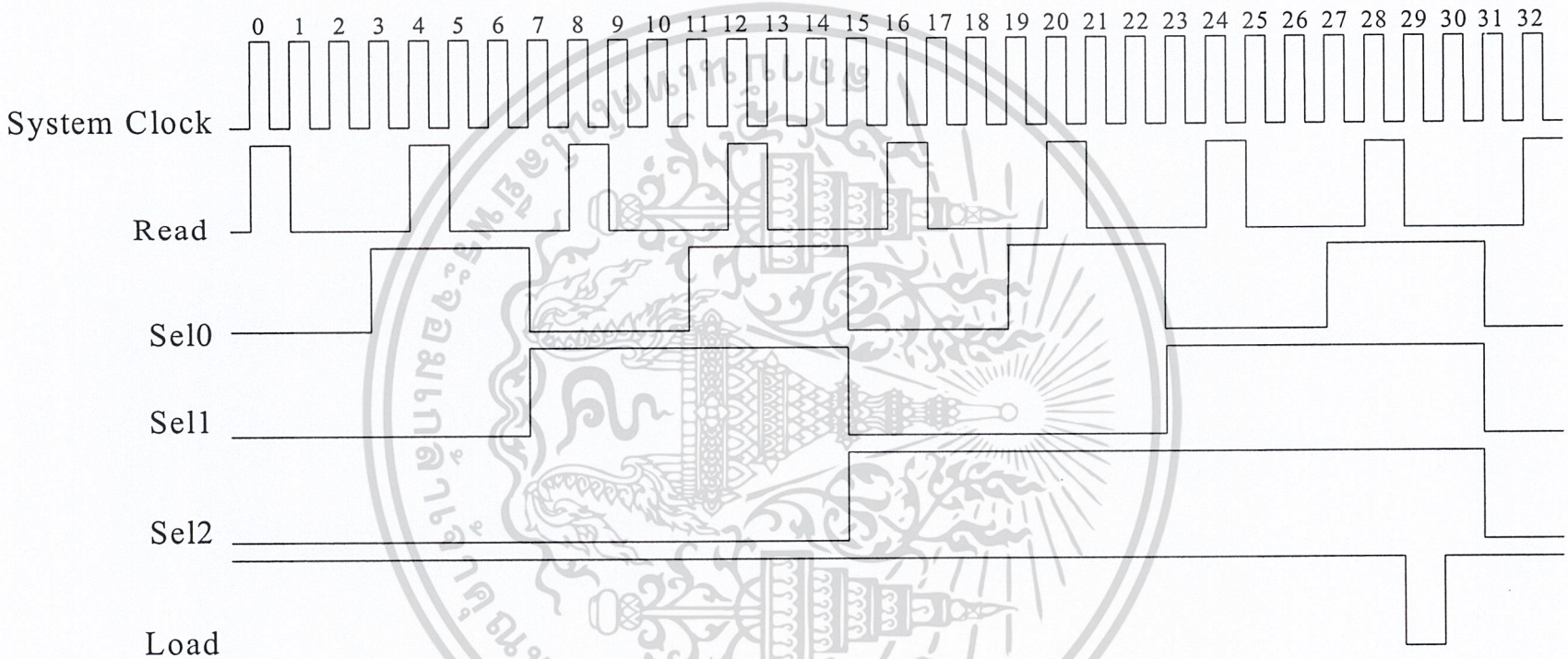
สัญญาณ sel0, sel1,sel2 เลือกคำตอบควบคุมมัลติเพล็กซ์ (Multiplex) และ ดีมัลติเพล็กซ์ (Demultiplex) ข้อมูลเข้าและออกส่วนประมวลผล

สัญญาณ Read ควบคุมการอ่านค่าดิจิตอลของวงจรแปลงสัญญาณอะนาล็อกเป็นสัญญาณดิจิตอล (ADC) Handcheck สัญญาณที่ทำการติดต่อกับคอมพิวเตอร์ เพื่อทำการแสดงผล



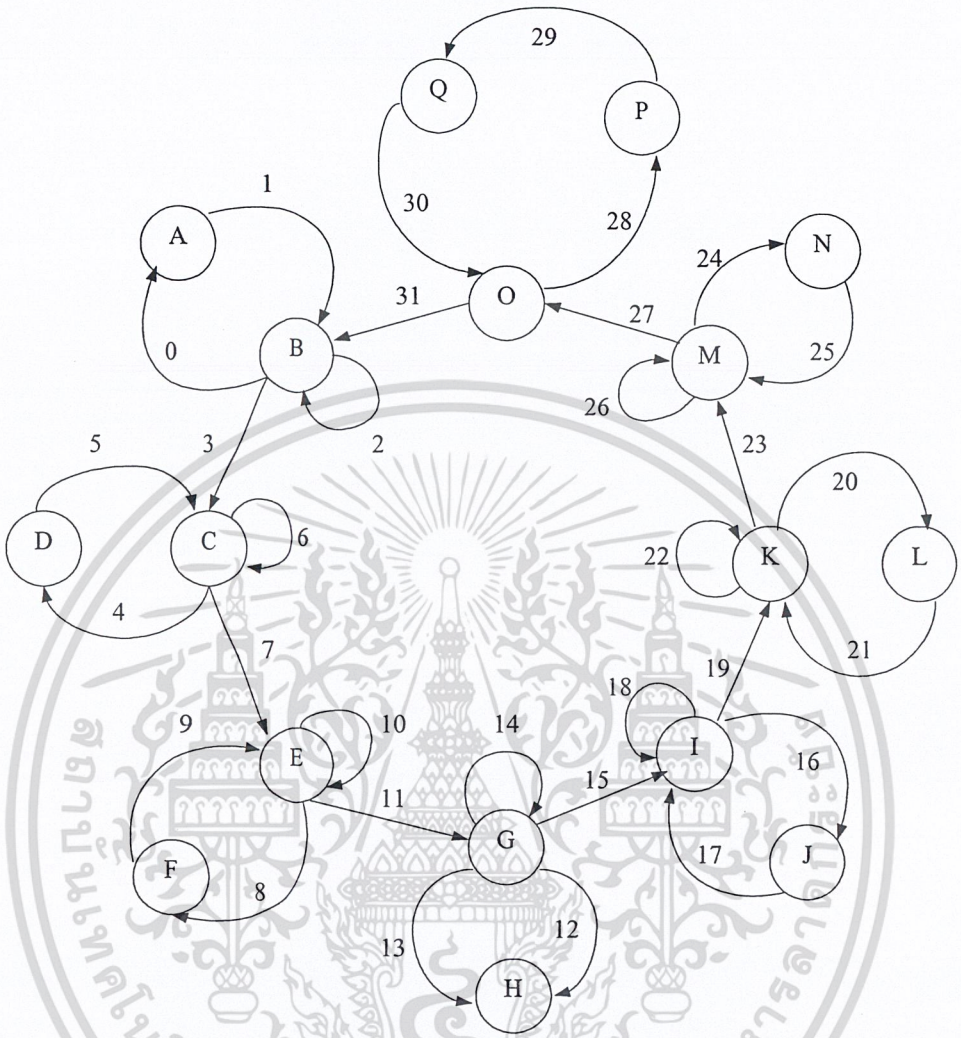
รูปที่ 3.22 บล็อกไดอะแกรมของส่วนควบคุม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.23 แสดง Timing ของส่วนควบคุม

จาก Timing Diagram เขียนเป็น State Diagram ได้ดังนี้



รูปที่ 3.24 State Diagram ของ Fast Fourier Processor

Read

A, D, F, H, J, L, N, P

Sel0

C, D, G, H, K, L, O, P, Q

Sel1

E, F, G, H, M, N, O, P, Q

Sel2

I, J, K, L, M, N, O, P, Q

LD

"0" ที่ Q

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 3.7 ตารางแสดงค่าประจำแต่ละ State

จำนวน Clock	ค่าประจำ State	State	จำนวน Clock	ค่าประจำ State	State
0	10001	A	16	10011	J
1	00001	B	17	00011	I
2	00001	B	18	00011	I
3	01001	C	19	01011	K
4	11001	D	20	11011	L
5	01001	C	21	01011	K
6	01001	C	22	01011	K
7	00101	E	23	00111	M
8	10101	F	24	10111	N
9	00101	E	25	00111	M
10	00101	E	26	00111	M
11	01101	G	27	01111	O
12	1101	H	28	11111	P
13	01101	G	29	01110	Q
14	01101	G	30	01111	O
15	00011	I	31	00001	B

จากสเตทไดอะแกรม สามารถนำไปเขียนโปรแกรม กำเนิดสัญญาณควบคุมต่างๆ แล้วทำการจำลองการทำงานและทดสอบ

ทำการรวมส่วนประกอบต่างๆ ตามบล็อกไดอะแกรม ทางโปรแกรมทำการจำลองการทำงานและทดสอบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4

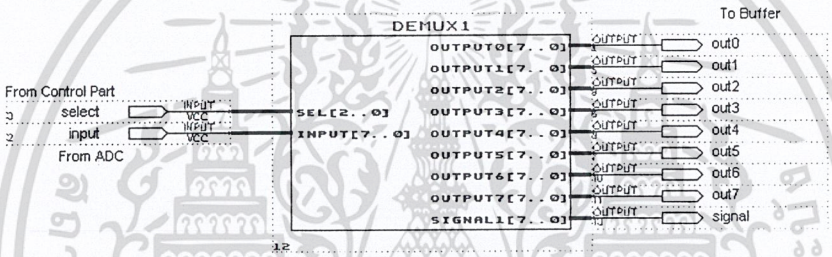
การทดลองและผลการทดลอง

จากการออกแบบส่วนต่างๆของตัวประมวลผลฟาสท์ฟูเรียร์ (Fast Fourier Transform Processor) ทำการเขียนโปรแกรมให้แต่ละส่วนทำงานตามที่ได้ออกแบบโดยใช้ภาษา VHDL ทำการ Compile แล้วทำการจำลองการทำงานของโปรแกรมแต่ละส่วนที่ได้เขียนขึ้นให้ได้ผลตามที่ออกแบบไว้

4.1 ส่วนอินพุท (Input Part)

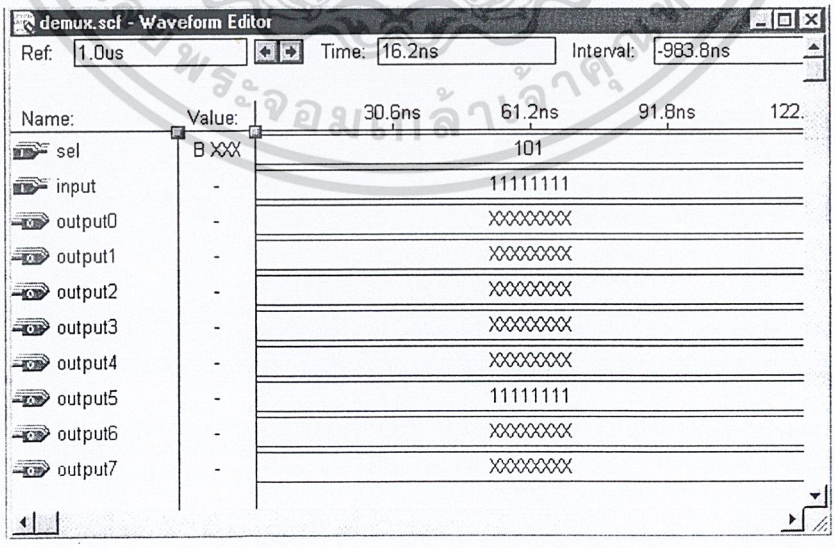
จะประกอบไปด้วยส่วนของวงจร Demultiplex และวงจร Buffer ซึ่งสามารถเขียนโปรแกรมและจำลองการทำงานทาง Software ได้ดังนี้

4.1.1 วงจรดีมัลติเพล็กซ์ (Demultiplex) สามารถเขียนโปรแกรมซึ่งสังเคราะห์ได้ อุปกรณ์มีสัญลักษณ์ (Symbol) ดังนี้



รูปที่ 4.1 แสดงสัญลักษณ์ของวงจรดีมัลติเพล็กซ์ (Demultiplex) ที่ได้จากการ Compile โปรแกรมที่เขียนขึ้น

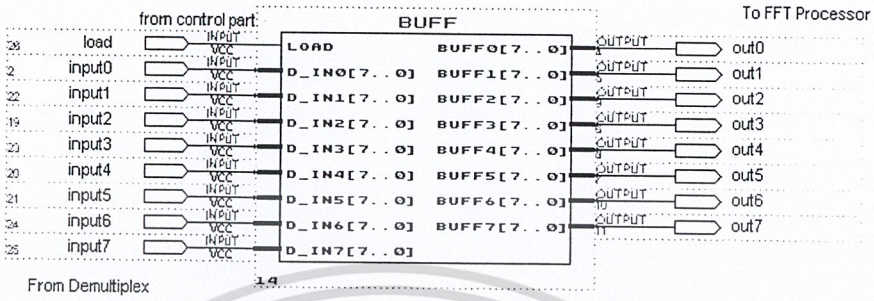
จากโปรแกรมที่เขียนขึ้นสามารถทำการจำลองการทำงาน (Simulation) ได้ดังนี้



รูปที่ 4.2 แสดงการจำลองการทำงานของวงจรดีมัลติเพล็กซ์ (Demultiplex)

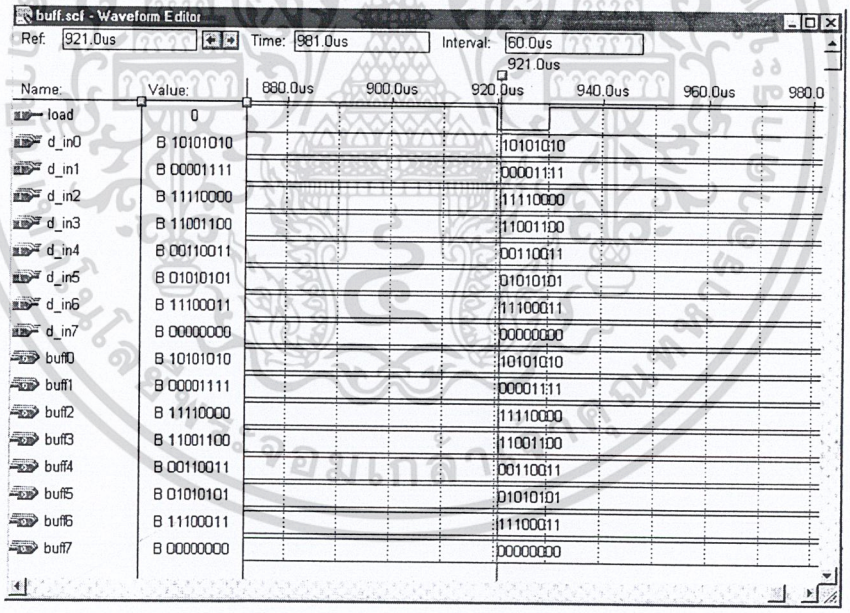
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.1.2 วงจรบัฟเฟอร์ (Buffer) สามารถเขียนโปรแกรมซึ่งสังเคราะห์ได้ อุปกรณ์มีสัญลักษณ์ (Symbol) ดังนี้



รูปที่ 4.3 แสดงสัญลักษณ์ของวงจรบัฟเฟอร์ (Buffer) ที่ได้จากการ Compile โปรแกรมที่เขียนขึ้น

จากโปรแกรมที่เขียนขึ้นสามารถทำการจำลองการทำงาน (Simulation) ได้ดังนี้



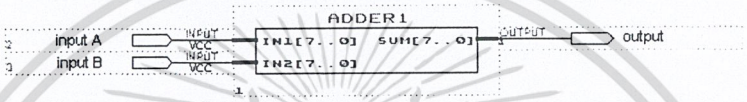
รูปที่ 4.4 แสดงการจำลองการทำงานของวงจร บัฟเฟอร์ (Buffer)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2 ส่วนประมวลผล (Processor Part)

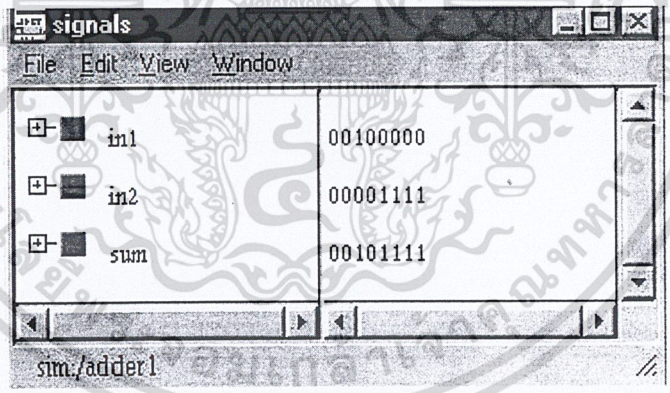
ทำหน้าที่ประมวลผลตัวประมวลผลฟาสท์ฟูเรียร์ (Fast Fourier Transform Processor) ของสัญญาณอินพุทที่เข้ามา จะประกอบไปด้วย วงจรบวก (Adder) วงจรลบ (Subtractor) วงจรทูคอมพลีเมนท์ (Two's complement) วงจรคูณจำนวนเชิงซ้อน (Complex multiplier) ทำการออกแบบ และทำการจำลองการทำงานของส่วนย่อยต่างๆว่าทำงานถูกต้อง แล้วทำการรวมวงจรทางซอฟต์แวร์ ทดสอบส่วนประมวลผลทั้งหมดอีกครั้ง

4.2.1 วงจรบวก (Adder) สามารถเขียนโปรแกรมซึ่งสังเคราะห์ได้ อุปกรณ์มีสัญลักษณ์ (Symbol) ดังนี้



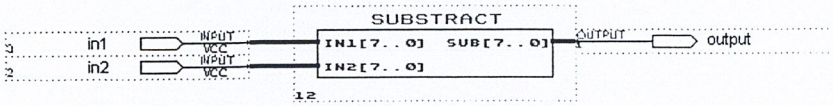
รูปที่ 4.5 แสดงสัญลักษณ์ของวงจรบวก (Adder) ที่ได้จากการ Compile โปรแกรมที่เขียนขึ้น

จากโปรแกรมที่เขียนขึ้นสามารถทำการจำลองการทำงาน (Simulation) ได้ดังนี้



รูปที่ 4.6 แสดงการจำลองการทำงานของวงจรบวก (Adder)

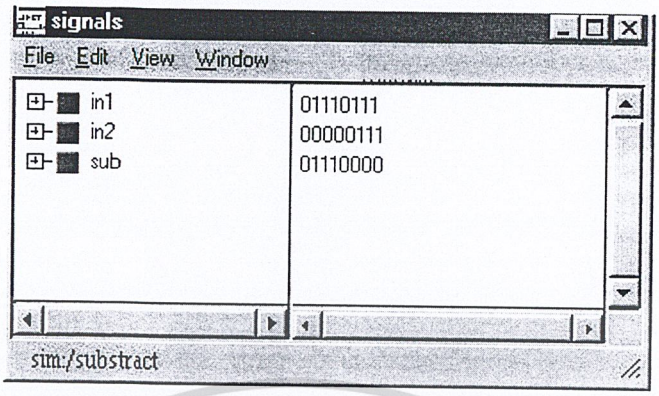
4.2.2 วงจรลบ (Subtractor) สามารถเขียนโปรแกรมซึ่งสังเคราะห์ได้ อุปกรณ์มีสัญลักษณ์ (Symbol) ดังนี้



รูปที่ 4.7 แสดงสัญลักษณ์ของวงจรลบ (Subtractor) ที่ได้จากการ Compile โปรแกรมที่เขียนขึ้น

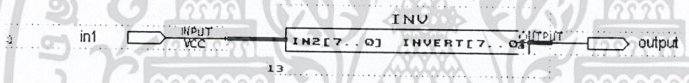
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากโปรแกรมที่เขียนขึ้นสามารถทำการจำลองการทำงาน (Simulation) ได้ดังนี้



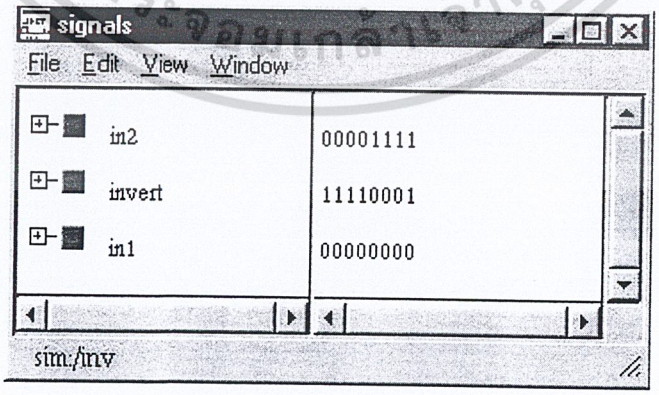
รูปที่ 4.8 แสดงการจำลองการทำงานของวงจรลบ (Subtractor)

4.2.3 วงจรทูคอมพลีเมนท์ (Two 's complement) สามารถเขียนโปรแกรมซึ่งสังเคราะห์ได้อุปกรณ์มีสัญลักษณ์ (Symbol) ดังนี้



รูปที่ 4.9 แสดงสัญลักษณ์ของวงจรทูคอมพลีเมนท์ (Two 's complement) ที่ได้จากการ Compile โปรแกรมที่เขียนขึ้น

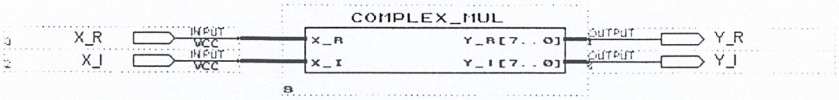
จากโปรแกรมที่เขียนขึ้นสามารถทำการจำลองการทำงาน (Simulation) ได้ดังนี้



รูปที่ 4.10 แสดงการจำลองการทำงานวงจรทูคอมพลีเมนท์ (Two 's complement)

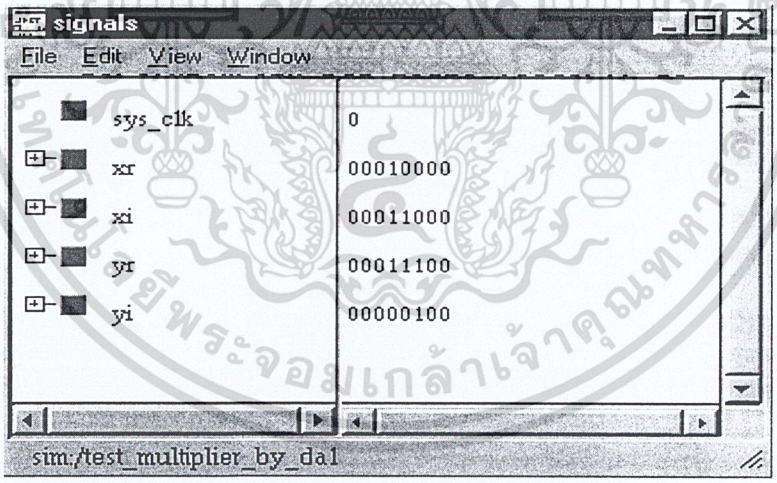
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2.4 วงจรคูณจำนวนเชิงซ้อน (Complex multiplier) ประกอบไปด้วยส่วนย่อยคือ หน่วยความจำ (ROM) วงจรเลื่อนข้อมูลแบบขนานเข้าอนุกรมออก (PISO) วงจรเลือกบวกหรือลบ (Adder or Subtractor) วงจรบัฟเฟอร์ (Buffer) ส่วนกำเนิดสัญญาณควบคุม (Control Unit) ทำจำลองการทำงานของทุกส่วน แล้วนำมาต่อกันทางซอฟต์แวร์ ทดสอบการทำงานทั้งหมดอีกครั้งหนึ่ง สามารถเขียนโปรแกรมซึ่งสังเคราะห์ได้ อุปกรณ์มีสัญลักษณ์ (Symbol) ดังนี้



รูปที่ 4.11 แสดงสัญลักษณ์ของวงจรคูณจำนวนเชิงซ้อน (Complex multiplier) ที่ได้จากการ Compile โปรแกรมที่เขียนขึ้น

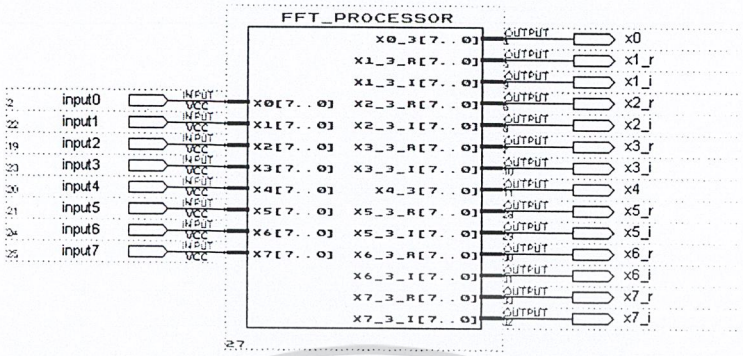
จากโปรแกรมที่เขียนขึ้นสามารถทำการจำลองการทำงาน (Simulation) ได้ดังนี้



รูปที่ 4.12 แสดงการจำลองการทำงานของวงจรคูณจำนวนเชิงซ้อน (Complex multiplier)

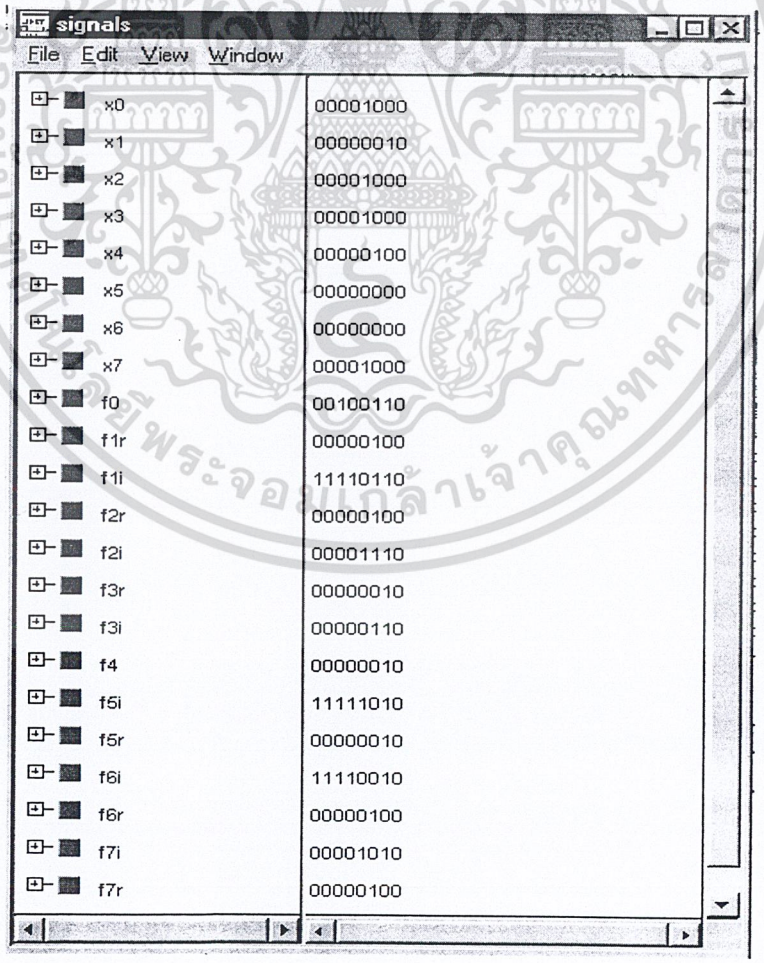
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เมื่อนำอุปกรณ์ที่ทำการออกแบบและเขียนโปรแกรมต่างๆมาประกอบกันให้เป็นส่วนประมวลผลฟาสท์ฟูเรียร์ ทำให้ได้ผลดังนี้



รูปที่ 4.13 แสดงสัญลักษณ์ของวงจร Processor ที่ได้จากการ Compile การรวมโปรแกรมย่อยต่างๆ

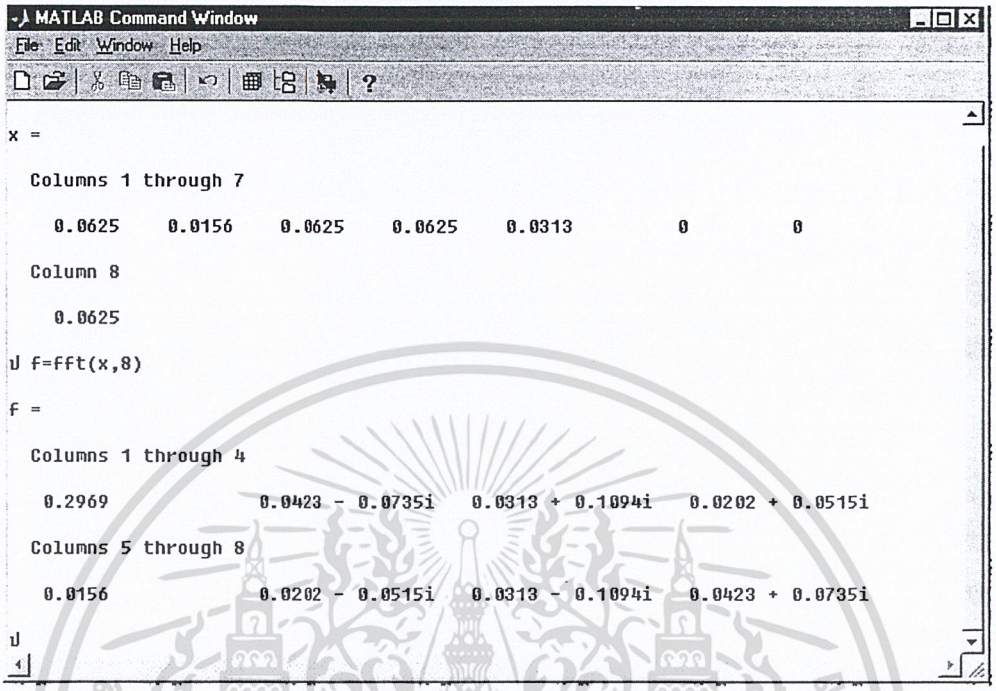
โปรแกรมที่เขียนขึ้นจากการรวมส่วนย่อยต่างๆที่เขียนขึ้นสามารถทำการจำลองการทำงาน (Simulation) ได้ดังนี้



รูปที่ 4.14 แสดงการจำลองการทำงานของส่วนประมวลผลฟาสท์ฟูเรียร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

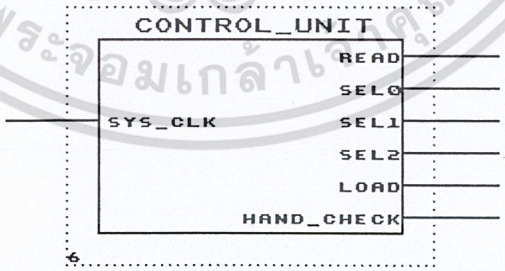
ทำการคำนวณการแปลงฟาสท์ฟูเรียร์โดยใช้โปรแกรม MATLAB โดยป้อนค่าอินพุทเหมือนกับค่าในการทดสอบการจำลองการทำงานของส่วนประมวลผล ได้ผลดังนี้



รูปที่ 4.15 แสดงการคำนวณการแปลงฟาสท์ฟูเรียร์โดยใช้โปรแกรม MATLAB

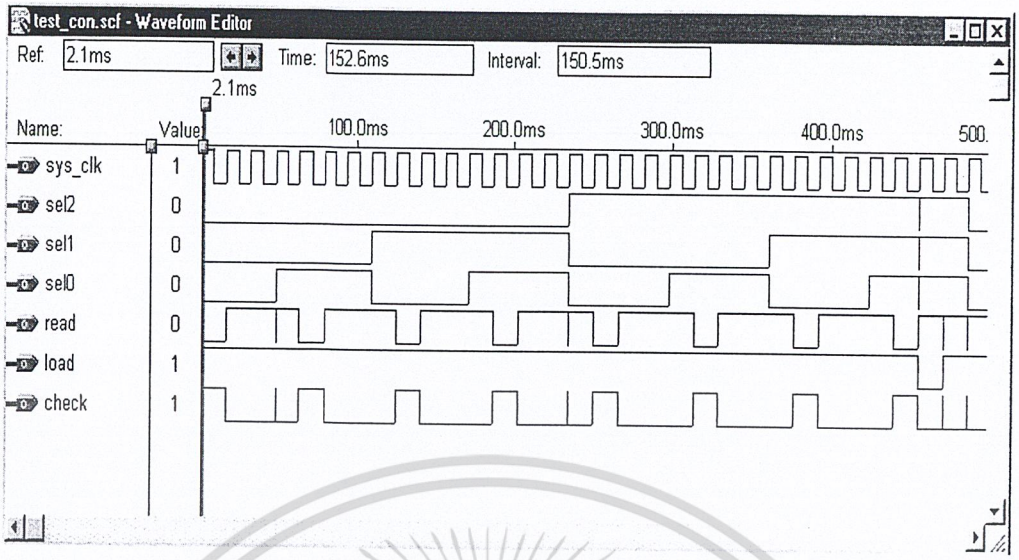
4.3 ส่วนกำเนิดสัญญาณควบคุม (Control Part)

เป็นตัวสร้างสัญญาณควบคุมส่วนต่างๆ จากสัญญาณนาฬิกาภายใน (Internal Clock) ทำการเขียนสามารถเขียนโปรแกรมซึ่งสังเคราะห์ได้ อุปกรณ์มีสัญลักษณ์ (Symbol) ดังนี้



รูปที่ 4.16 แสดงสัญลักษณ์ของภาคกำเนิดสัญญาณควบคุมจากโปรแกรมที่เขียนขึ้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

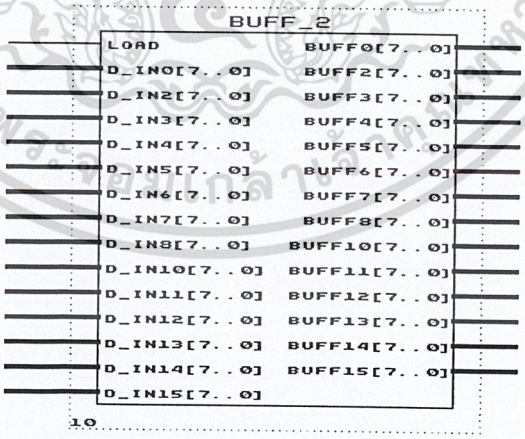


รูปที่ 4.17 แสดงการจำลองการทำงานของวงจร Multiplex

4.4 ส่วนของภาคเอาท์พุท (Output Part)

ประกอบไปด้วยวงจรบัฟเฟอร์ และวงจรมัลติเพล็กซ์ ซึ่งสามารถเขียนโปรแกรมและจำลองการทำงานทาง Software ได้ดังนี้

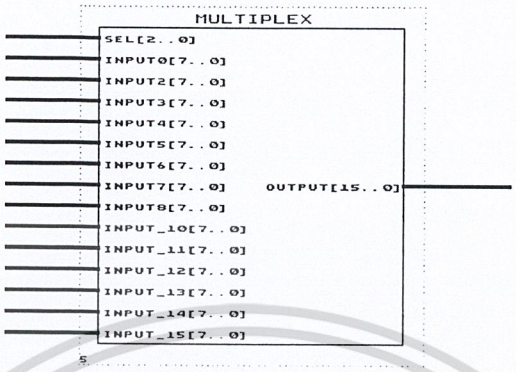
4.4.1 วงจรบัฟเฟอร์ สามารถเขียนโปรแกรมซึ่งสังเคราะห์ได้ อุปกรณ์มีสัญลักษณ์ (Symbol) ดังนี้



รูปที่ 4.18 แสดงสัญลักษณ์ของวงจรบัฟเฟอร์ (Buffer)
ที่ได้จากการ Compile โปรแกรมที่เขียนขึ้น

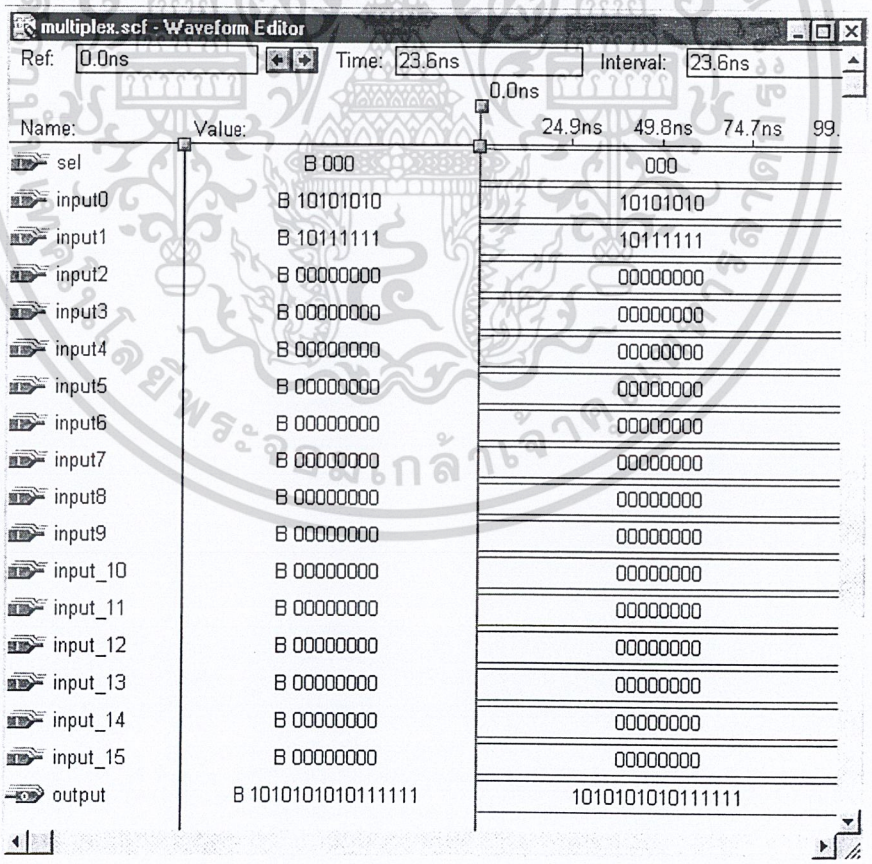
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.4.2 วงจรมัลติเพล็กซ์ สามารถเขียนโปรแกรมซึ่งสังเคราะห์ได้ อุปกรณ์มีสัญลักษณ์ (Symbol) ดังนี้



รูปที่ 4.19 แสดงสัญลักษณ์ของวงจรมัลติเพล็กซ์ (Multiplex) ที่ได้จากการ Compile โปรแกรมที่เขียนขึ้น

จากโปรแกรมที่เขียนขึ้นสามารถทำการจำลองการทำงาน (Simulation) ได้ดังนี้

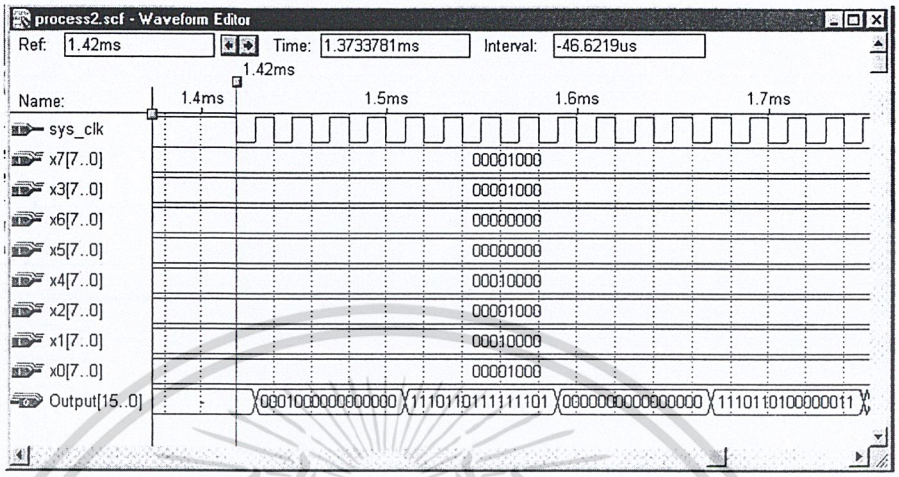


รูปที่ 4.20 แสดงการจำลองการทำงานของวงจรมัลติเพล็กซ์ (Multiplex)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.5 ตัวประมวลผลฟาสท์ฟูเรียร์ทรานส์ฟอร์มทั้งหมด

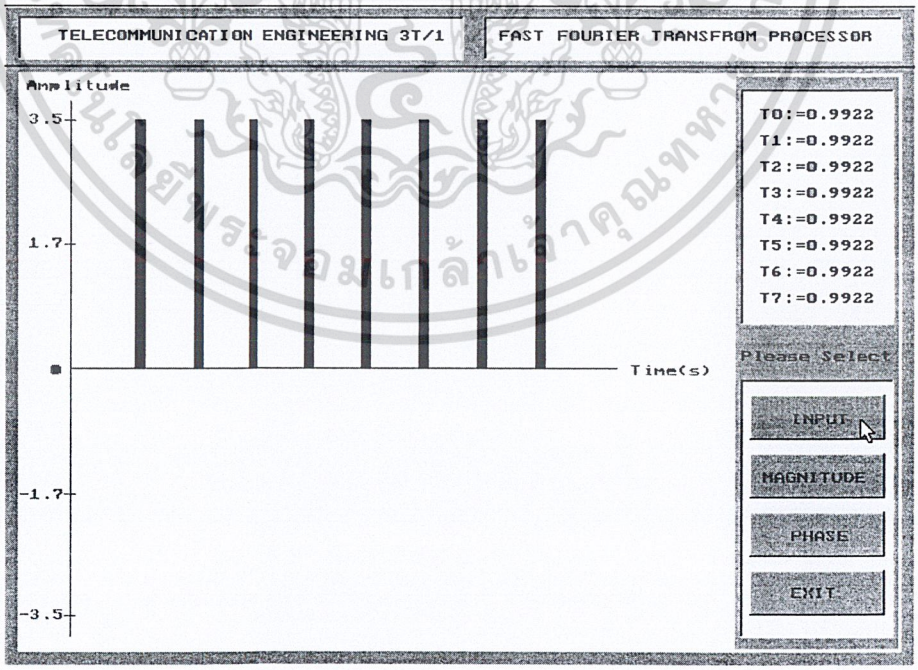
จากการเขียนโปรแกรมทั้งหมดในทุกๆ ส่วน ได้ ตัวประมวลผลฟาสท์ฟูเรียร์ตามที่ออกแบบไว้ ซึ่งทำการจำลองการทำงานได้ดังนี้



รูปที่ 4.21 แสดงการจำลองการทำงานของตัวประมวลผลฟาสท์ฟูเรียร์ทั้งหมด

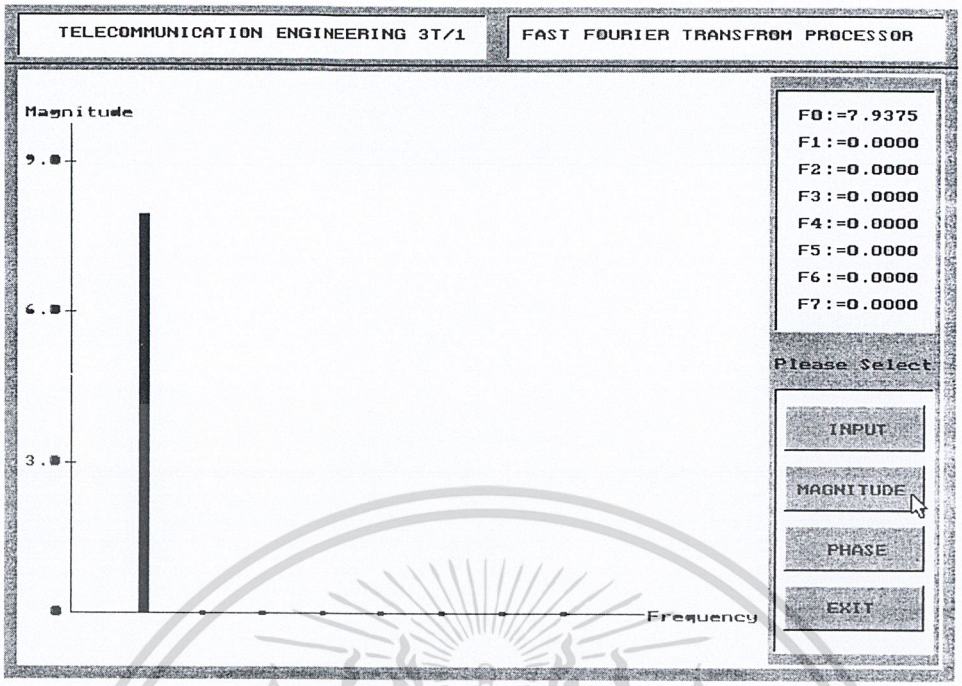
4.6 การแสดงผลผ่านจอคอมพิวเตอร์

ทำการดาวโหลด (Download) โปรแกรมที่ใช้เขียนขึ้นลงบน FPGA แล้วทดสอบโดยการรับค่าจากวงจร ADC เข้าส่วนประมวลผลแล้วแสดงผล แสดงจอภาพได้ดังนี้

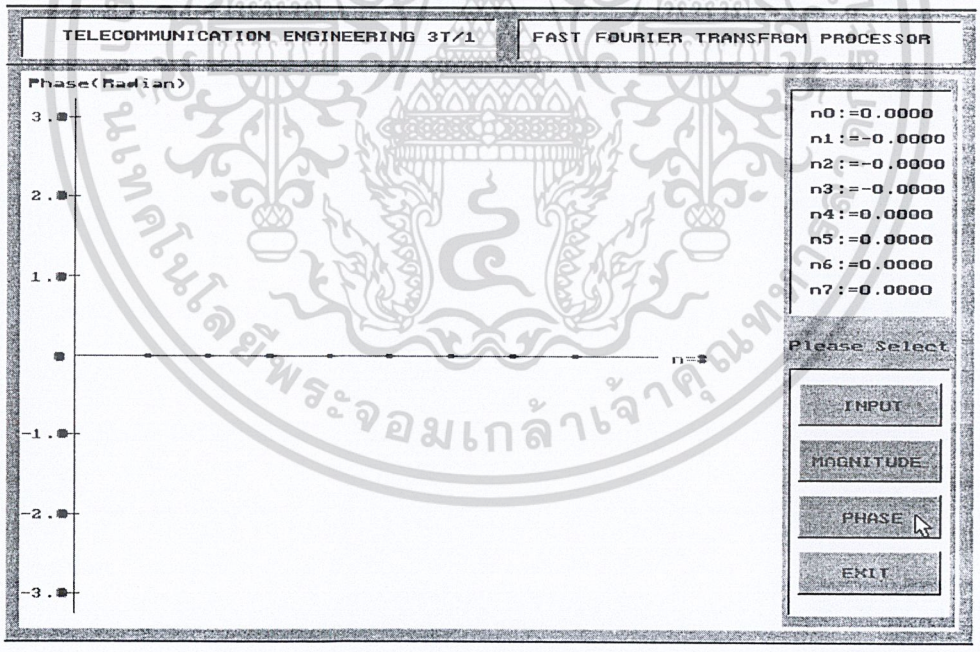


รูปที่ 4.22 แสดงรูปของสัญญาณอินพุทเมื่อป้อนไฟตรง 1V

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

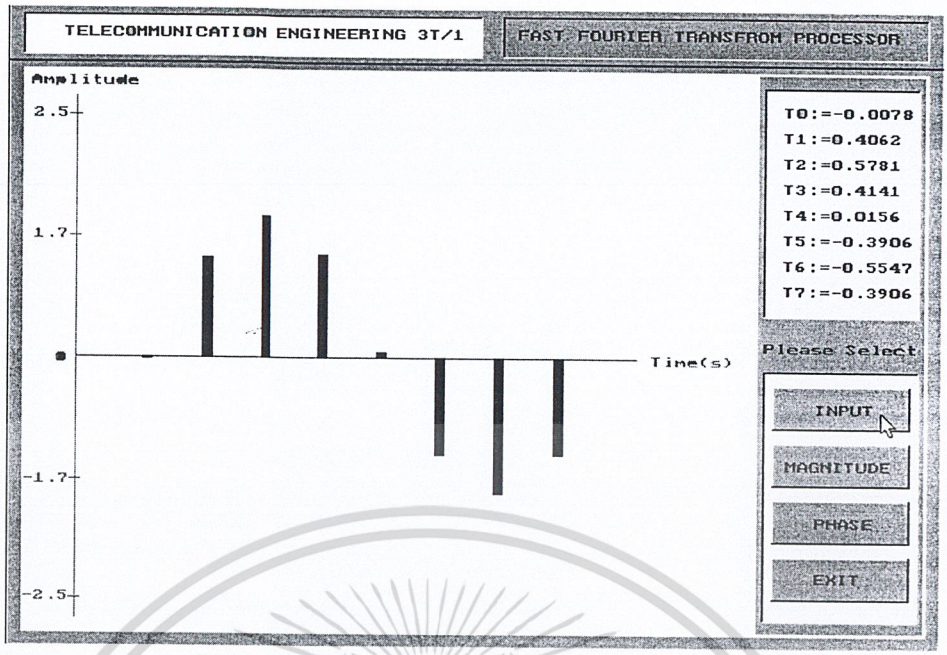


รูปที่ 4.23 แสดงขนาดของสแปคตรัม เมื่อป้อนไฟตรง 1 V

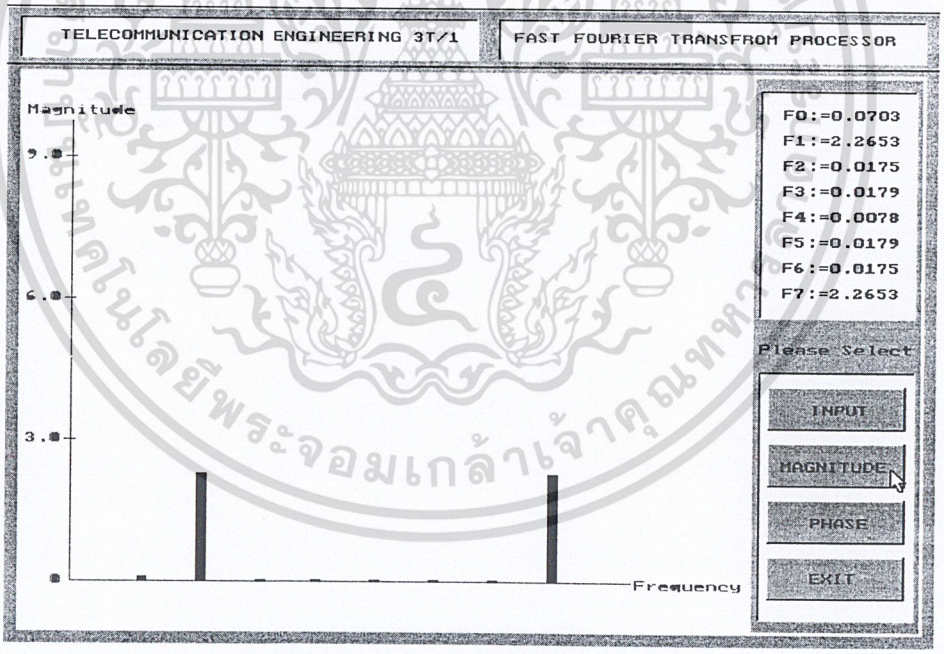


รูปที่ 4.24 แสดงเฟสของสเปคตรัม เมื่อป้อนไฟตรง 1V

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

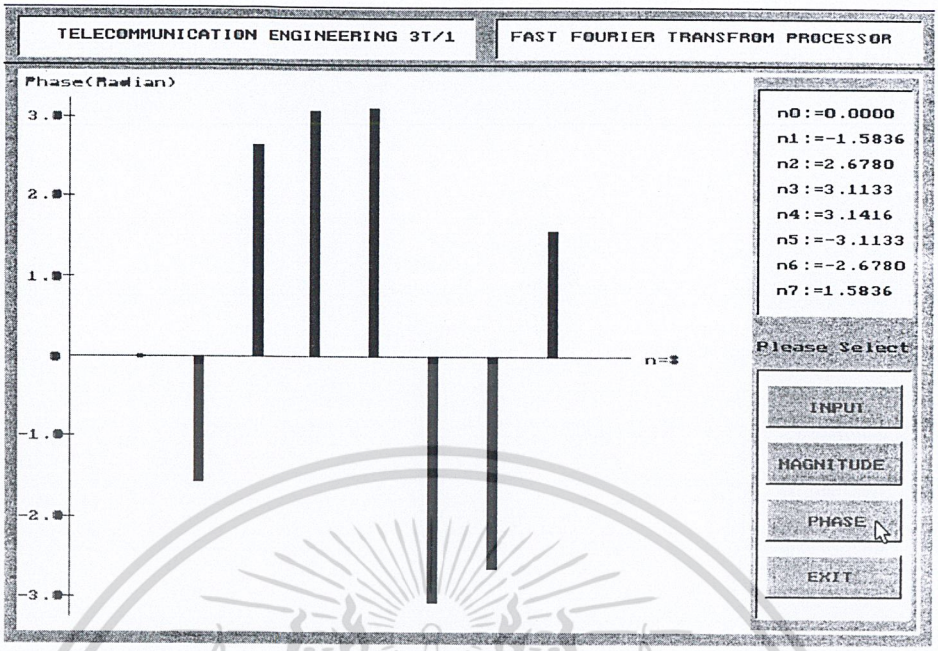


รูปที่ 4.25 แสดงรูปของสัญญาณอินพุทเมื่อป้อนสัญญาณไซน์ 1 kHz

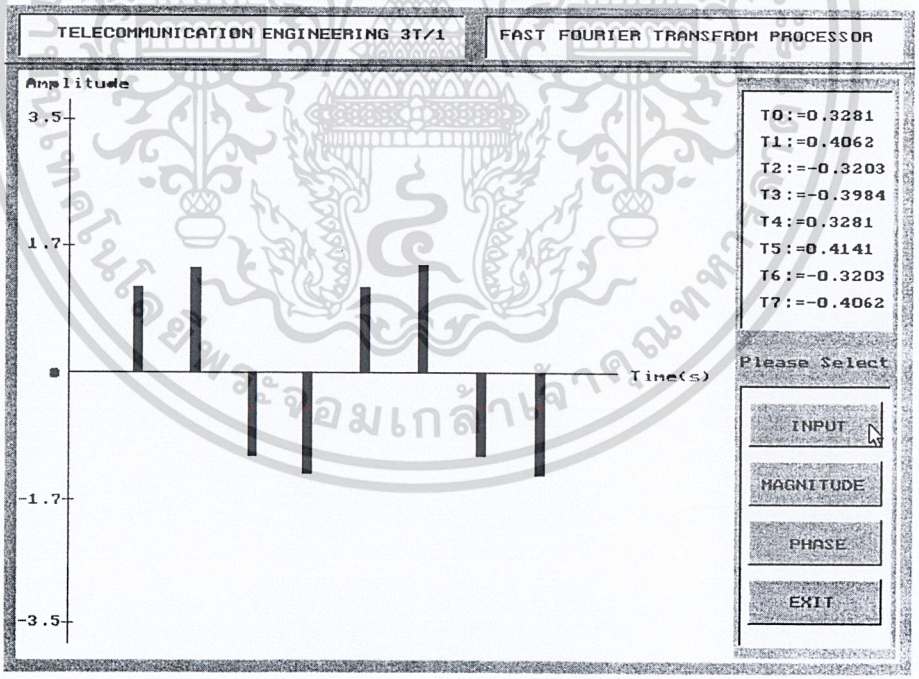


รูปที่ 4.26 แสดงขนาดของสแปคตรัม เมื่อป้อนสัญญาณไซน์ 1 kHz

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

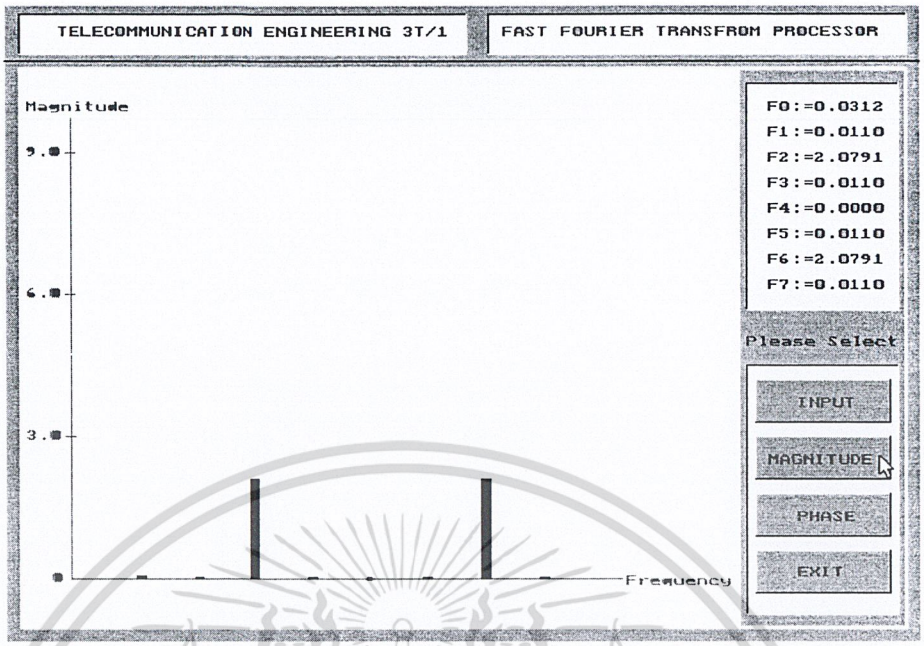


รูปที่ 4.27 แสดงเฟสของสเปคตรัม เมื่อป้อนสัญญาณไซน์ 1 kHz

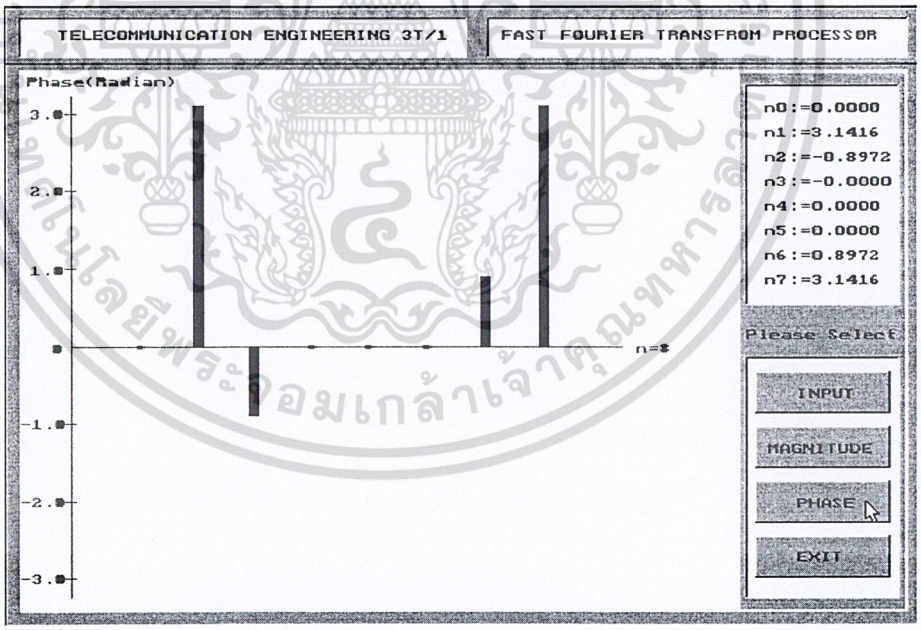


รูปที่ 4.28 แสดงรูปของสัญญาณอินพุทเมื่อป้อนสัญญาณไซน์ 2 kHz

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

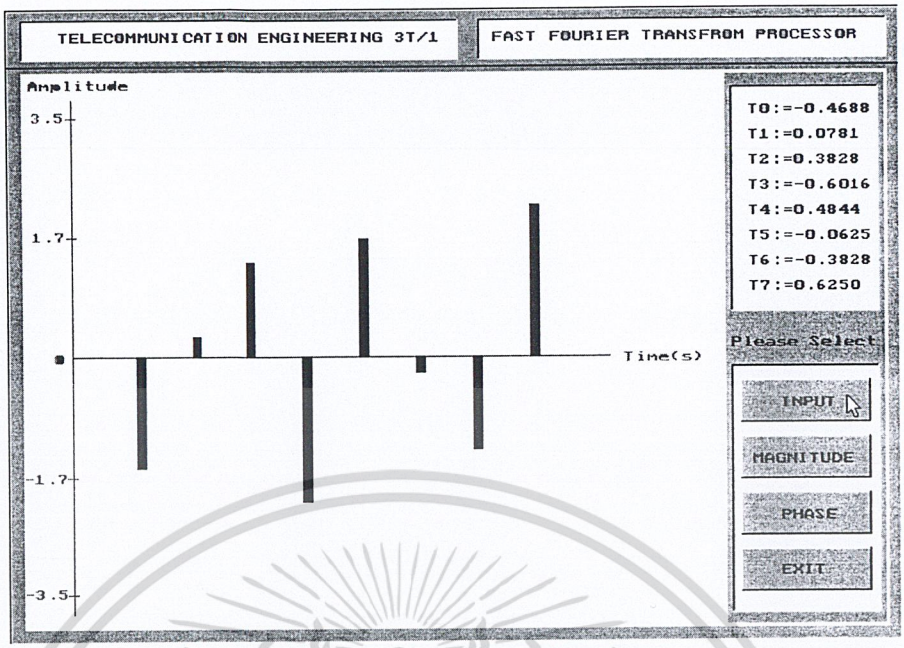


รูปที่ 4.29 แสดงขนาดของสแปคตรัม เมื่อป้อนสัญญาณไซน์ 2 kHz

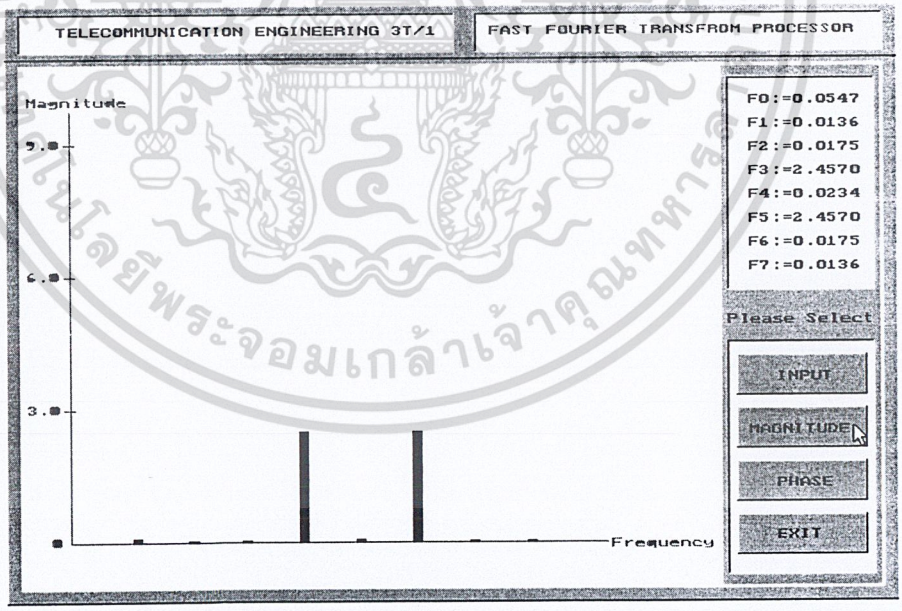


รูปที่ 4.30 แสดงเฟสของสเปคตรัม เมื่อป้อนสัญญาณไซน์ 2 kHz

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

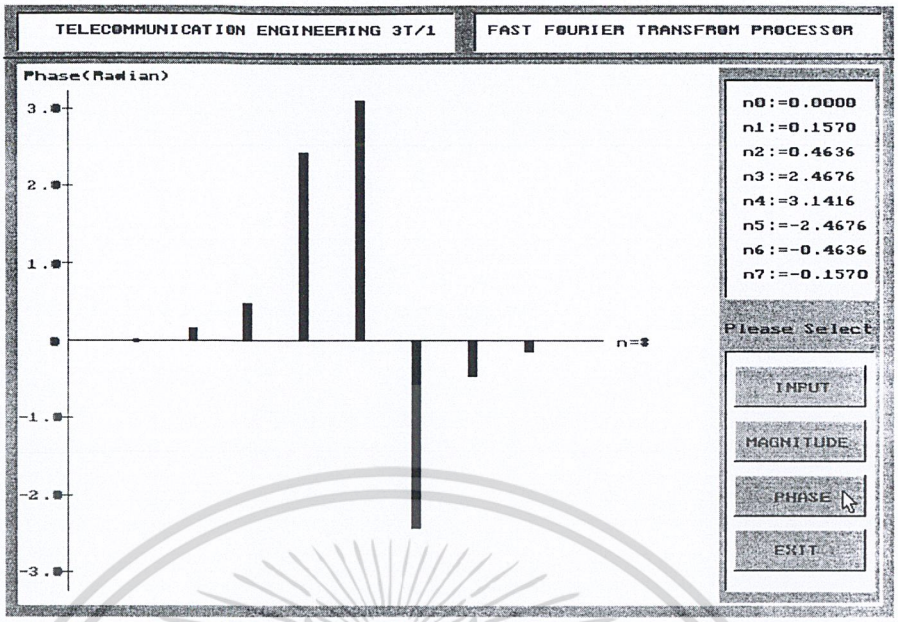


รูปที่ 4.31 แสดงรูปของสัญญาณอินพุทเมื่อป้อนสัญญาณไซน์ 3kHz

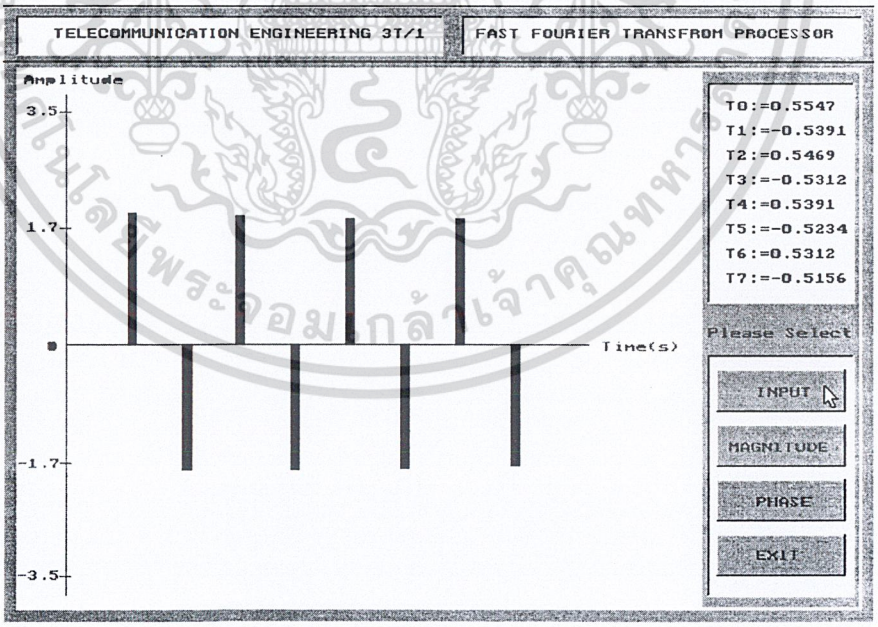


รูปที่ 4.32 แสดงขนาดของสแปคตรัม เมื่อป้อนสัญญาณไซน์ 3kHz

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

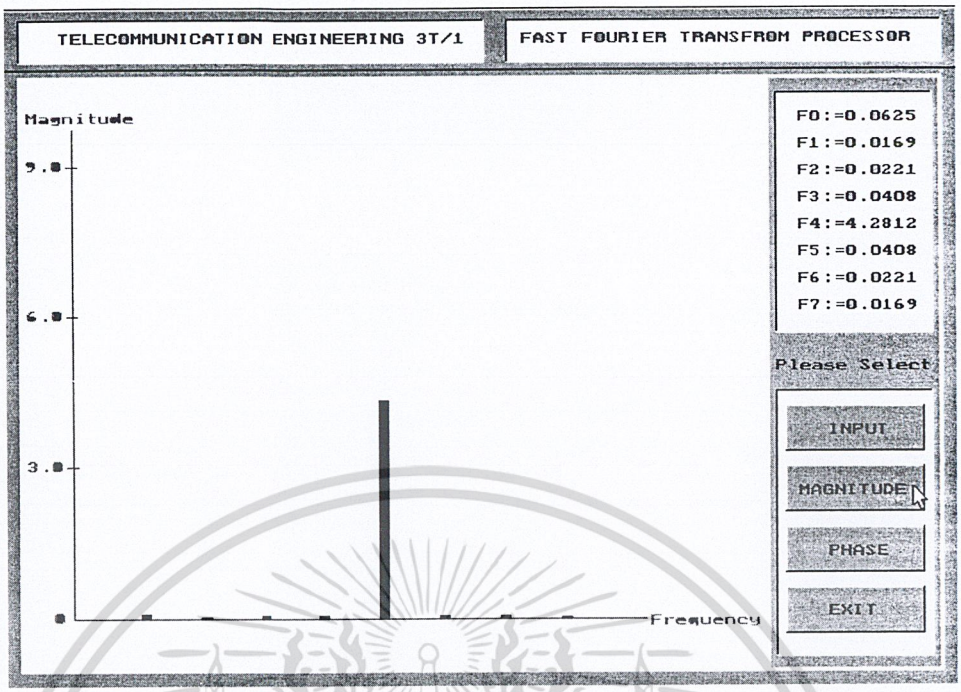


รูปที่ 4.33 แสดงเฟสของสเปคตรัม เมื่อป้อนสัญญาณไซน์ 3kHz

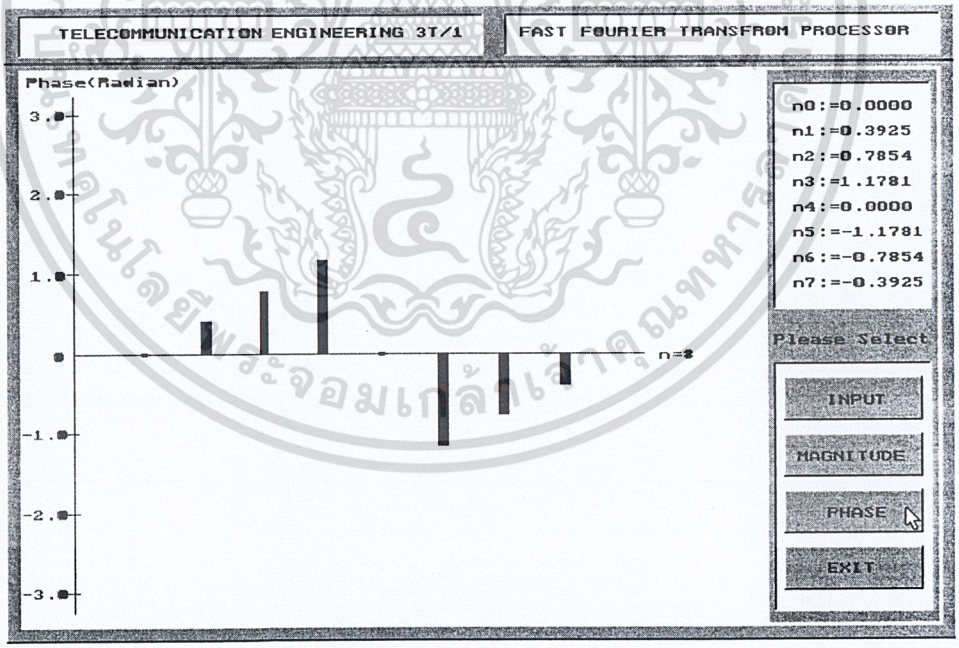


รูปที่ 4.34 แสดงรูปของสัญญาณอินพุทเมื่อป้อนสัญญาณไซน์ 4 kHz

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

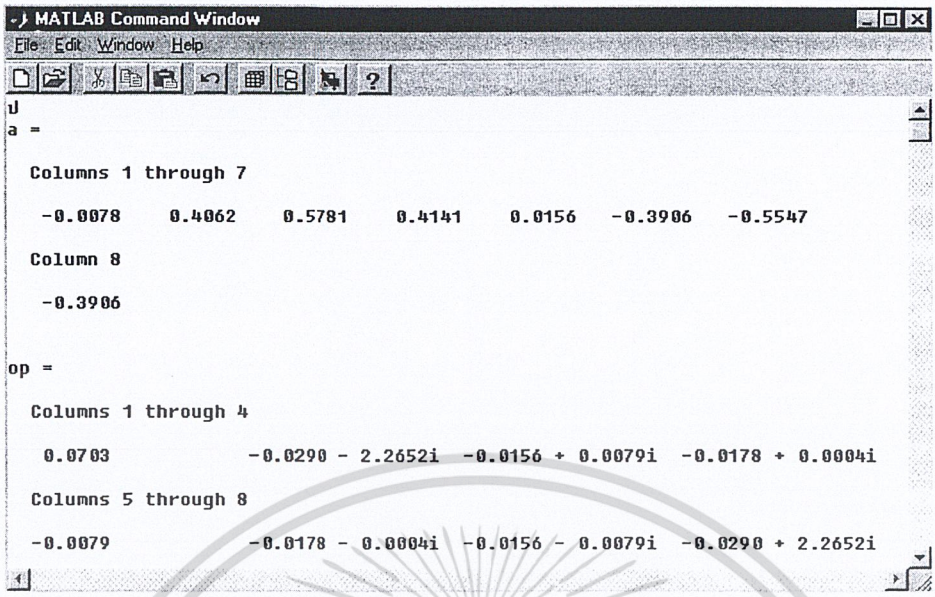


รูปที่ 4.35 แสดงขนาดของสแปคตรัม เมื่อป้อนสัญญาณไซน์ 4 kHz

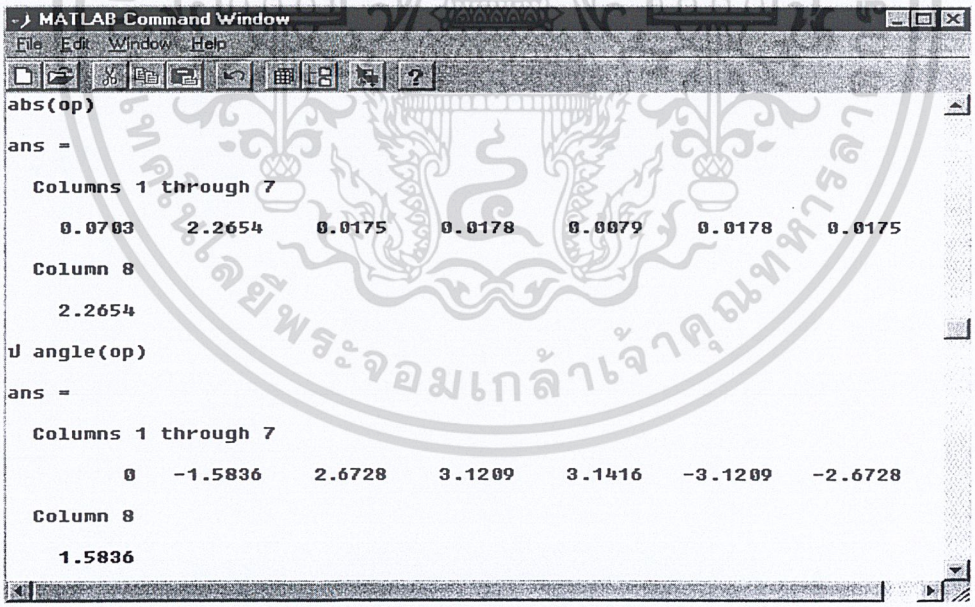


รูปที่ 4.36 แสดงเฟสของสเปคตรัม เมื่อป้อนสัญญาณไซน์ 4 kHz

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

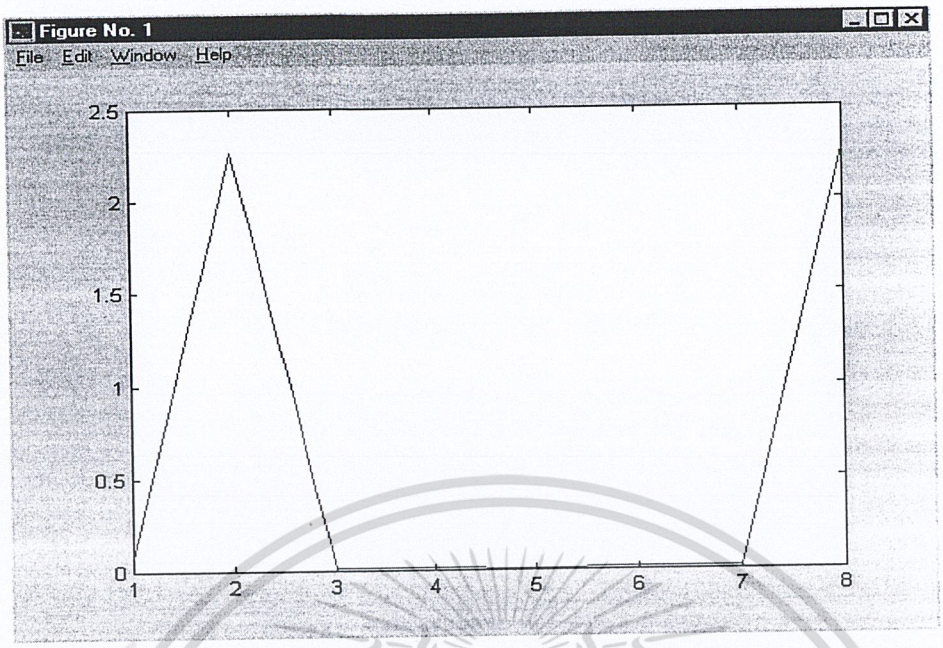


รูปที่ 4.37 แสดงการคำนวณ FFT โดยใช้โปรแกรม MATLAB เมื่อป้อนค่าเท่ากับ ค่า Sampling อินพุท ไซน์ความถี่ 1 kHz

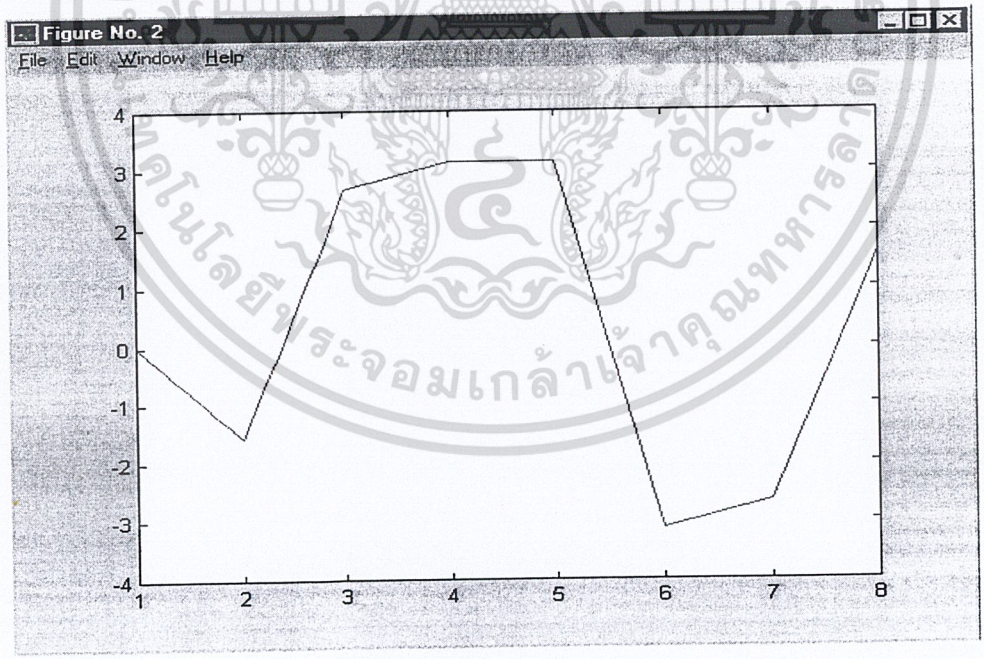


รูปที่ 4.38 แสดงการคำนวณ ขนาดและเฟสโดยใช้โปรแกรม MATLAB เมื่อป้อนค่าเท่ากับ ค่าอินพุท ไซน์ความถี่ 1 kHz

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



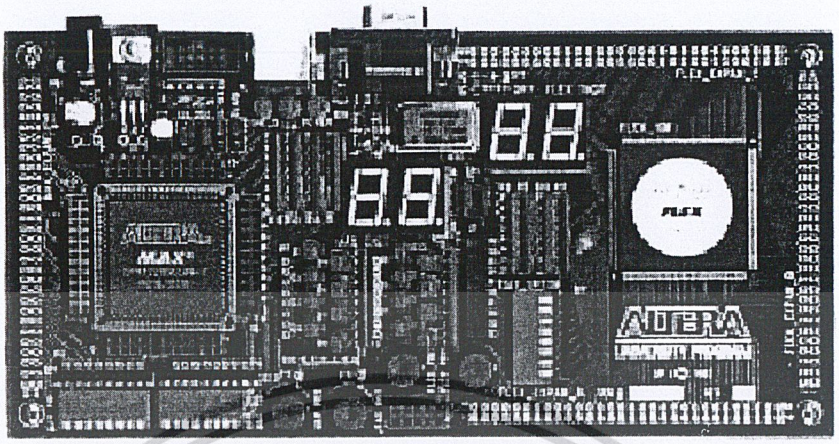
รูปที่ 4.39 กราฟแสดงขนาดของสเปคตรัมของสัญญาณไซน์ 1 kHz ของการแปลง FFT 8 จุดโดยใช้โปรแกรม MATLAB



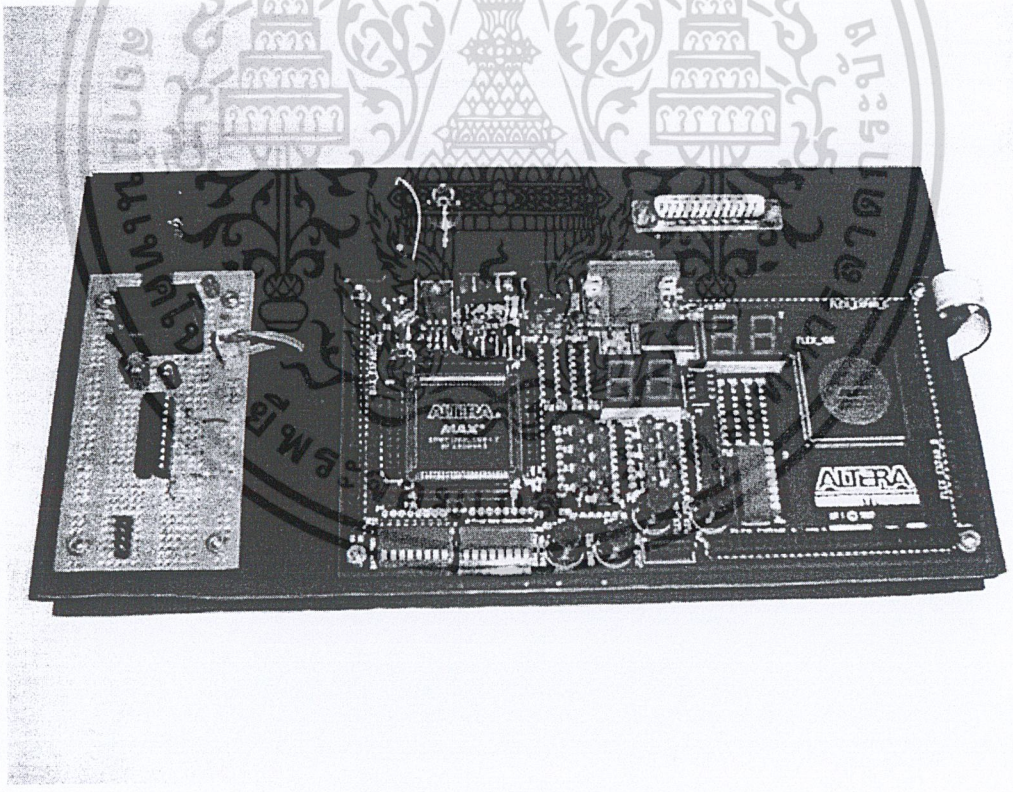
รูปที่ 4.40 กราฟแสดงเฟสของสเปคตรัมของสัญญาณไซน์ 1 kHz ของการแปลง FFT 8 จุดโดยใช้โปรแกรม MATLAB

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.7 รูปของบอร์ด FPGA



รูปที่ 4.41 แสดงรูปของบอร์ด FPGA ที่ใช้ในการทดลอง



รูปที่ 4.42 แสดงรูปของบอร์ด FPGA ที่ใช้ในการทดลอง เมื่อต่อกับ ADC 0820

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5
บทวิจารณ์และบทสรุป

จากการออกแบบ ตัวประมวลผลฟาสท์ฟูเรียร์ (Fast Fourier Transform Processor) จำนวน 8 จุด ข้อมูล Fixed Point 8 bit โดยการเขียนโปรแกรมด้วยภาษา VHDL ทำการ Implement ลงบน FPGA Board แล้วทำการทดสอบโดยการรับค่าขึ้นไปแสดงผลบนจอคอมพิวเตอร์โดยใช้ ภาษาซี(C-Programming) จะเห็นได้ว่า ตัวประมวลผลฟาสท์ฟูเรียร์ สามารถทำงานได้อย่างถูกต้อง โดยเปรียบเทียบจากการคำนวณจากโปรแกรม Matlab และสามารถทำงานแบบเวลาจริงได้ ในช่วงความถี่ของย่านความถี่เสียง โดยการพล็อตค่า Magnitude และ Phase บนจอภาพคอมพิวเตอร์ เมื่อเปรียบเทียบผลการทดลองกับการคำนวณในโปรแกรม Matlab พบว่าได้ค่าที่ตรงกัน รวมถึงมีกราฟขนาด(Magnitude) และกราฟเฟส(Phase) ของ spectrum ที่เหมือนกัน

ปัญหาที่เกิดขึ้นส่วนใหญ่จะเป็นในเรื่องข้อมูลทางด้านฮาร์ดแวร์(Hardware) ของตัวบอร์ด FPGA และซอฟต์แวร์(Software) ที่ใช้ในการคอมไพล์(Compile) , จำลองการทำงาน(Simulation) และสังเคราะห์(Synthesis) ตลอดจนข้อมูลการใช้งานโปรแกรมต่างๆ ซึ่งไม่มีข้อมูลรายละเอียดการใช้งานเลย

แนวทางการพัฒนาจะเห็นได้ว่าจำนวนจุดของฟาสท์ฟูเรียร์ทรานส์ฟอร์มที่ใช้แค่ 8 จุด สามารถบอกสเปกตรัมของความถี่ ได้เพียงคร่าวๆ เท่านั้น ในการแปลงฟาสท์ฟูเรียร์ทรานส์ฟอร์ม จำนวนจุดที่มากกว่าจะสามารถแสดงสเปกตัมของความถี่ที่ละเอียดขึ้นได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หนังสืออ้างอิง

[1] วัลลภ สุระกำพลธร, " การประมวลผลสัญญาณเชิงตัวเลขและการแปลง", ภาควิชาอิเล็กทรอนิกส์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, ปีการศึกษา 2533

[2] ชานนท์ พนาวรรต,สุรพงษ์ วัฒนา,อัครพล ตรีรัตน์, " การออกแบบสร้างวงจรกรองเชิงเลขอันดับ 4 ", ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตร์บัณฑิต สาขาวิศวกรรมโทรคมนาคม สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, ปีการศึกษา 2536

[3] ชาติชาย ดิษฐกุล , " ภาษา VHDL"

[4] S.A White, " Applications of Distributed Arithmetic to Digital Signal Processing : A Tutorial Review", IEEE ASSP. Magazine, Vol.6, NO.3, PP.4-13 July 1989

[5] A. Peled and B. Liu, " Digital Signal Processing Theory, Design and Implementation", John Wiley & Sons, 1979

[6] S. Sjoholm, L. Lindh, " VHDL for Designer", Prentice Hall, 1997

[7] D.F. Elloott, "Handbook of Digital Processing ,Engineering Applications ",Academic Press, INC .1987

[8] R.R. Varghese, C.G. Tharayil," Project Report ,Design, Simulation and Synthesis of an FFT Processor using VHDL",Mahatma Gandhi University, Kottayam, Kerala, 2001

[9] http://www.altera.com

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้