เครื่องโปรแกรม GAL

GAL PROGRAMMER

โดย

นางสาวจิตราภรณ์ บุญลักษณานุสรณ์

นางสาวศิริพร ก้อนทอง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2539

เครื่องโปรแกรม GAL

GAL PROGRAMMER

โดย

นางสาวจิตราภรณ์ บุญลักษณานุสรณ์

นางสาวศิริพร ก้อนทอง

อาจารย์ที่ปรึกษา

ดร.กิติพล ชิตสกุล

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2539

ปริญญานิพนธ์ ปีการศึกษา 2539

ภาควิชา อิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เครื่องโปรแกรม GAL

ผู้จัดทำ

1. นางสาวจิตราภรณ์ บุญลักษณานุสรณ์
2. นางสาวศิริพร ก้อนทอง

ดร.กิตติพล ชิตสกุล อาจารย์ที่ปรึกษา

(.....)

ชื่อโครงงานภาษาไทย เครื่องโปรแกรม GAL

ชื่อโครงงานภาษาอังกฤษ GAL PROGRAMMER

ผู้จัดทำ

1. นางสาวจิตราภรณ์ บุญลักษณานุสรณ์ 36014077
2. นางสาวศิริพร ก้อนทอง 36014437

โครงงานนี้ได้รับการตรวจสอบแล้ว พร้อมที่จะทำการสอบได้

.....

(.....)

อาจารย์ที่ปรึกษา

เครื่องโปรแกรม GAL

นางสาวจิตราภรณ์ บุญลักษณานุสรณ์
นางสาวศิริพร ก้อนทอง
ดร.กิติพล ชิตสกุล อาจารย์ที่ปรึกษา
ปีการศึกษา 2539

บทคัดย่อ

ปัจจุบันอุปกรณ์ประเภท PLD (Programmable Logic Device) เริ่มมีบทบาทในการออกแบบวงจรอิเล็กทรอนิกส์มากขึ้น เนื่องจากเหมาะกับการออกแบบสร้างวงจรดิจิตอลที่ต้องใช้ลอจิกเกตจำนวนมากๆซึ่งจะทำให้วงจรมีขนาดเล็กลงและมีราคาถูก ซึ่ง GAL® (Generic Array Logic) เป็นอุปกรณ์ PLD ชนิดหนึ่งที่สามารถทำการลบและโปรแกรมใหม่ได้ด้วยไฟฟ้า ดังนั้น GAL® จึงมีความยืดหยุ่นในการใช้งานและปรับแก้ไข

ในการทำโครงงานชิ้นนี้เพื่อศึกษาถึงโครงสร้างภายใน และวิธีการลบและโปรแกรมใหม่ รวมทั้งสร้างเครื่องโปรแกรมต้นแบบสำหรับ GAL® เบอร์ 16V8 และ 20V8 การศึกษานี้จะช่วยให้ขอบเขตของการนำ GAL® ไปใช้งานได้กว้างขวางยิ่งขึ้น

GAL PROGRAMMER

Miss Jittraporn Bunlaksananusorn
Miss Siriporn Gontong
Dr. Kitipol Chidsakul Advisor
1996

Abstract

Nowadays , Programmable Logic Device (PLD) becomes a great role for digital logic circuit design and construction because it can reduce the size of the circuit and also the investment. GAL® (Generic Array Logic) is a generation of the PLD which can erase and reprogram many times.So GAL® provides flexibility in modification of circuit functions.

The main aim of this project is to study the structure erase programming of GAL® and also to construct a prototype of GAL® programmer for the 16V8 and 20V8.This study will encourage using the GAL® in any applications.

สารบัญ

สารบัญภาพ

สารบัญตาราง

บทที่ 1
บทนำ

การออกแบบและสร้างวงจรดิจิตอลลอจิก หากเป็นวงจรขนาดเล็ก การเลือกใช้ลอจิกเกตแบบเป็นตัวๆ ย่อมสามารถกระทำได้ง่าย แต่ในกรณีวงจรขนาดใหญ่และซับซ้อนการออกแบบดังกล่าวหากต้องทำการพิจารณาใหม่เนื่องจากเกิดความผิดพลาดหรือไม่เหมาะสม ทำให้สิ้นเปลืองค่าใช้จ่ายเพราะต้องใช้อุปกรณ์จำนวนมาก อีกทั้งยังเปลืองเนื้อที่ชิ้นงาน การออกแบบลายวงจรก็มักจะเกิดความยุ่งยากอีกด้วย ดังนั้นการนำเอาอุปกรณ์ลอจิกที่สามารถโปรแกรมได้ (Programmable Logic Device : PLD)มาใช้งานแทน จะลดจำนวนตัวอุปกรณ์ที่ใช้เป็นจำนวนมาก ทำให้ประหยัดเนื้อที่และช่วยลดข้อยุ่งยากในการออกแบบลายวงจร ข้อดีอีกประการหนึ่งของการใช้งานอุปกรณ์ลอจิกที่สามารถโปรแกรมได้นี้ คือ สามารถป้องกันการลอกเลียนแบบได้เป็นอย่างดี

อุปกรณ์ลอจิกที่โปรแกรมได้มีอยู่หลายชนิด ซึ่งมีโครงสร้างโดยทั่วๆไปคล้ายคลึงกัน แต่ที่จะกล่าวถึงในปริญญานิพนธ์ฉบับนี้จะเป็นเจเนอริคอะเรย์ลอจิก (GAL®) ซึ่งสามารถทำการล้างและโปรแกรมใหม่ได้ด้วยไฟฟ้า เราจะกล่าวถึงโครงสร้างโดยทั่วไปและการสร้างเครื่องโปรแกรม GALในบทถัดไป ซึ่งสามารถใช้โปรแกรมวงจรลอจิกได้ตามที่ต้องการพร้อมกับผลการทดสอบโดยการโปรแกรมลอจิกฟังก์ชันง่ายๆ

เครื่องโปรแกรมGALนี้ จะช่วยในการศึกษาการออกแบบอุปกรณ์ลอจิกที่โปรแกรมได้ และสามารถนำไปโปรแกรมGALซึ่งเป็นอุปกรณ์ประเภทโปรแกรมได้ชนิดหนึ่ง เพื่อสามารถนำไปใช้งานในการออกแบบวงจรดิจิตอลที่ไม่ยุ่งยากไปจนถึงระบบที่มีความซับซ้อนได้อย่างมีประสิทธิภาพ

บทที่ 2

PROGRAMMABLE LOGIC DEVICE (PLD)

Programmable Logic Device (PLD) เป็นอุปกรณ์ที่ใช้ในการออกแบบวงจรดิจิตอลแทนลอจิกเกต จุดเริ่มต้นของ PLD เริ่มขึ้นด้วย PROM ก่อน แล้วจึงมีบริษัท Monolithic Memory ซึ่งเป็นผู้คิดค้น PAL(Programmable Array Logic) อีกบริษัทหนึ่งคือ Signetics Corporation ซึ่งเป็นผู้คิดค้น FPLA (Field Programmable Logic Array) หรือเรียกสั้นๆว่า PLA ลอจิกไดอะแกรมของ PROM,PLA และ PAL แสดงได้ดังรูปที่ 2.1

ฟังก์ชันหลักๆของ PAL , PLA, PLS และ PROM เหมือนกัน แต่โครงสร้างภายในเท่านั้นที่แตกต่างกัน โดยพิจารณาที่ PROM ก่อน โครงสร้างของPROMเป็นการจัดเรียงลอจิกเกตในลักษณะแมทริกซ์ โดยที่อินพุทของแอนด์เกตจะถูกป้อนด้วยลอจิกคงที่ตามสัญญาณอินพุท ส่วนอินพุทของออร์เกตจะถูกโปรแกรมได้

PAL มีลักษณะโครงสร้างภายในตรงข้ามกับ PROM คือ อินพุทของแอนด์เกตเป็นแบบโปรแกรมได้ ส่วนอินพุทของออร์เกตจะคงที่

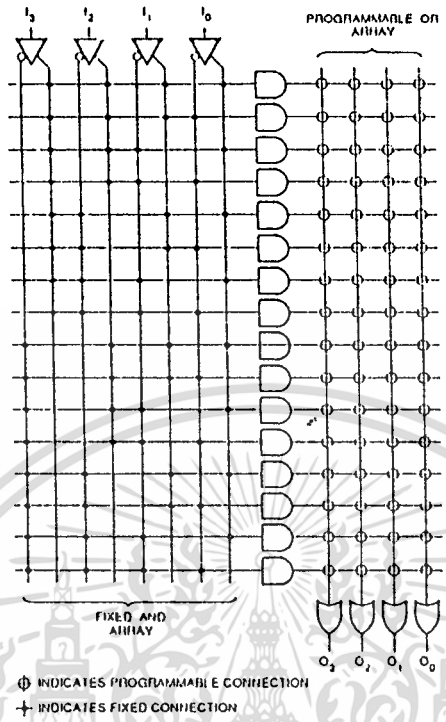
PLAจะมีโครงสร้างที่พิเศษกว่า PROMและ PALคือ อินพุทของแอนด์และออร์เกตสามารถโปรแกรมได้ โครงสร้างภายในของ PLAจะยุ่งยากซับซ้อนและมีความสามารถสูงกว่า อีกทั้งยังช่วยให้การคำนวณลอจิกเพื่อกำหนดหน้าที่ทำได้หลายรูปแบบมากกว่าเนื่องจากสามารถโปรแกรมได้ 2ทาง แต่มีราคาค่อนข้างสูงและเครื่องมือที่ใช้โปรแกรมก็ต้องพิเศษมากกว่า

PLS (Programmable Logic Sequence) มีโครงสร้างซับซ้อนกว่า PLA คือ นอกจากจะโปรแกรมได้ทั้งออร์เกตและแอนด์เกตแล้ว ยังมีฟลิปฟลอบซึ่งทำหน้าที่เป็นรีจิสเตอร์ทั้งทางอินพุทและเอาท์พุทอีกด้วย ดังนั้นจึงสามารถนำมาประยุกต์ใช้งานได้กับวงจรดิจิตอลที่ต้องการให้ทำงานตามลำดับเวลาที่มีสัญญาณนาฬิกามาเกี่ยวข้องด้วย

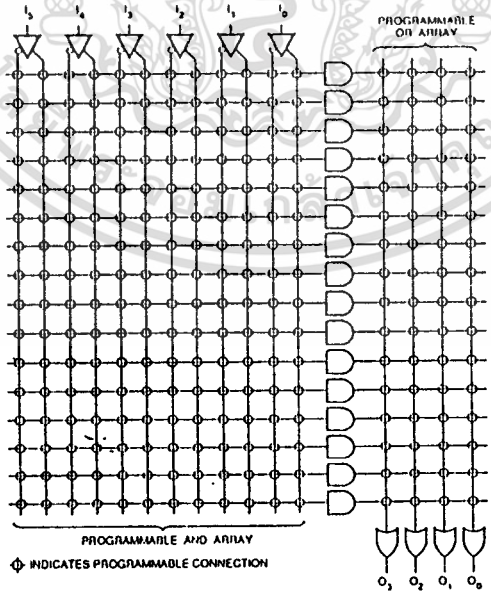
ต่อมาบริษัท International CMOS Technology ได้พัฒนา PAL ให้เป็นอุปกรณ์ programmable electrically erasable logic (PEEL) เพื่อช่วยเพิ่มประสิทธิภาพการทำงานให้สูงขึ้นโดยมี I/O microcell ที่ทำงานได้หลายฟังก์ชัน ดังแสดงดังรูปที่ 2.2

Generic Array Logic (GAL®) ถูกผลิตโดยบริษัท Lattice Semiconductor Corp. ซึ่งผู้ใช้สามารถกำหนดลักษณะการทำงานของเอาท์พุทมาโครเซลล์ (output macrocell) ได้โดยแต่ละแบบสามารถทำการควบคุมแยกกันได้

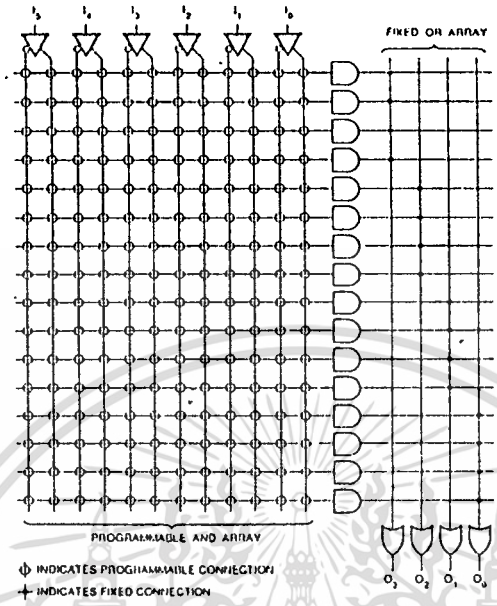
GAL16V8 และ GAL20V8 สามารถใช้แทนอุปกรณ์ไบโพลาร์ PAL ชนิด 20 ขา และ 24 ขาได้ ลักษณะและการออกแบบ GAL ทั้งสองนี้จะกล่าวโดยละเอียดในบทถัดไป



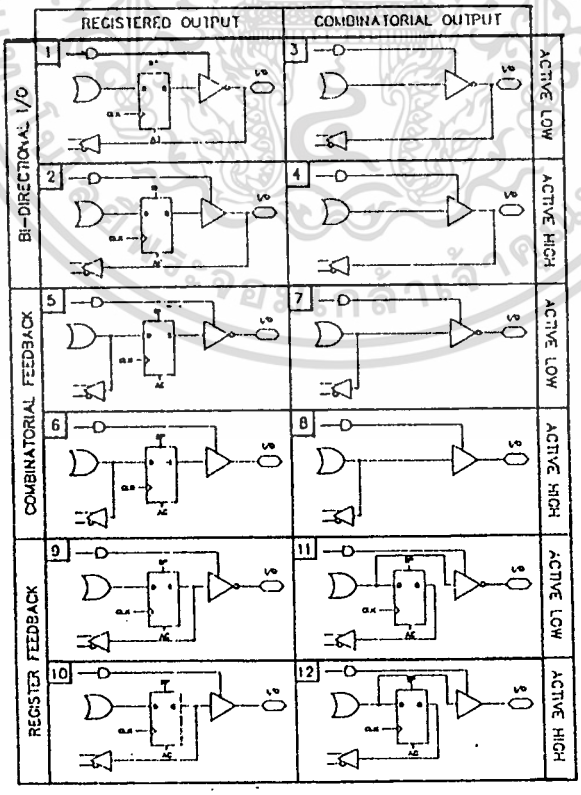
รูปที่ 2.1.1 ลอจิกไดอะแกรมของ PROM



รูปที่ 2.1.2 ลอจิกไดอะแกรมของ PLA



รูปที่ 2.1.3 ลอจิกไดอะแกรมของ PAL

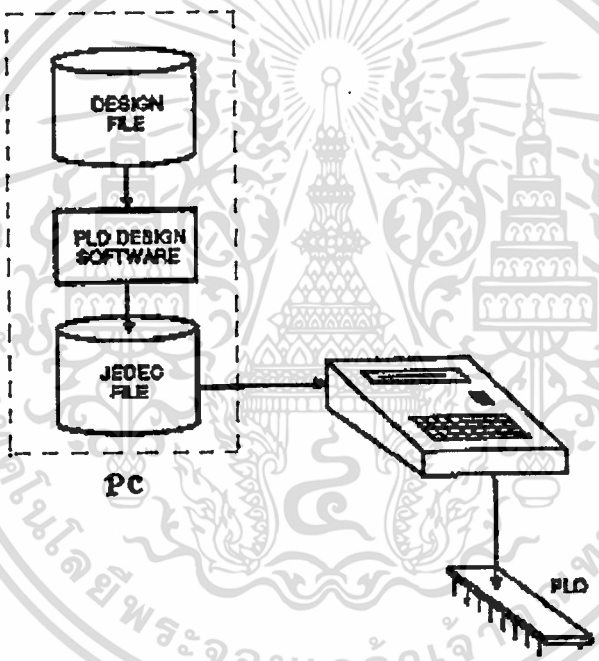


รูปที่ 2.2 ลักษณะการทำงานทั้ง 12 แบบของ PEEL I/O macrocell

2.1 ขั้นตอนการออกแบบ PLD (Programmable Logic Device)

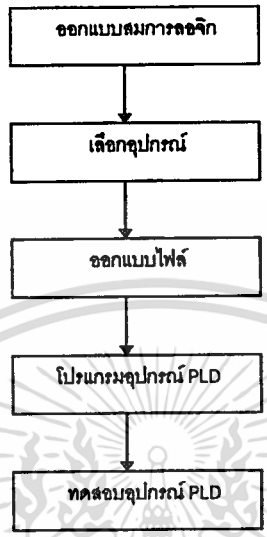
PLD เป็นอุปกรณ์ที่ใช้ในการสร้างวงจรดิจิตอลได้อย่างกว้างขวางตั้งแต่ในงานลอจิกซึ่งมีลอจิกฟังก์ชันตั้งแต่ง่ายๆไปจนถึงระบบที่ซับซ้อน ข้อดีของอุปกรณ์ประเภทนี้คือ มีราคาไม่แพง ใช้ง่าย และง่ายต่อการออกแบบ ซึ่งขั้นตอนการออกแบบจะกล่าวโดยละเอียดต่อไป

อุปกรณ์ PLD ประกอบด้วยโครงสร้างของแอนด์อะเรย์และออร์อะเรย์ โดยอะเรย์ที่สามารถโปรแกรมได้เหล่านี้(programmable array) จะเชื่อมต่อกัน ซึ่งจะถูกโปรแกรมโดยผู้ใช้อีกครั้ง การออกแบบวงจรโดยใช้ PLD จะเกี่ยวข้องกับการออกแบบฟังก์ชันของวงจรโดยซอฟท์แวร์ และ เครื่องโปรแกรมอุปกรณ์ ดังแสดงในรูปที่ 2.3



รูปที่ 2.3 แสดงขั้นตอนการโปรแกรม PLD และการใช้ซอฟท์แวร์และเครื่องโปรแกรม

การออกแบบฟังก์ชันเป็นการกำหนดว่าฟิวส์ไหนจะถูกโปรแกรมตามสมการผลรวมของลอจิกที่ต้องการ ขั้นตอนการออกแบบเริ่มจากการสร้างไฟล์ตามฟังก์ชันที่ต้องการ ซึ่งอยู่ในรูปของสมการผลรวม และสามารถนำไปเขียนอยู่ในรูปไทม์มิ่งไดอะแกรม(timing diagram) หรือตารางค่าความจริงได้ โดยไฟล์ที่ได้จะถูกแปลงเป็น JEDEC file หลังจากนั้นทำการจำลองการทำงาน (simulate)เพื่อให้ฟังก์ชันที่ได้ถูกต้องตามมาตรฐาน JEDEC ไฟล์ที่ได้จะถูกดาวน์โหลด (download)ไปยังเครื่องโปรแกรมเพื่อทำการโปรแกรมอุปกรณ์ต่อไป โดยเครื่องโปรแกรมจะถูกต่อกับเครื่องพีซี ขั้นตอนในการออกแบบ PLD แสดงได้ดังรูปที่ 2.4



รูปที่ 2.4 ขั้นตอนการสร้างลอจิกฟังก์ชันโดยใช้ PLD

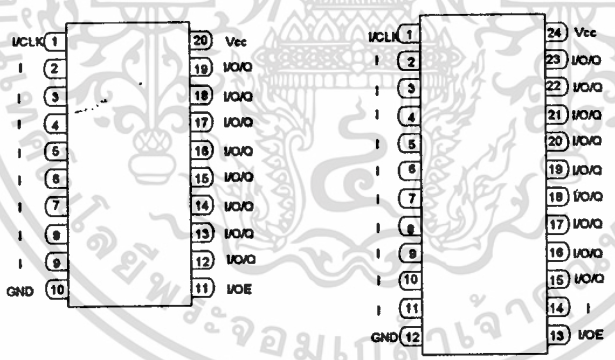
2.2 การทดสอบ PLD

การทดสอบ PLD สำหรับในงานอุตสาหกรรมซึ่งเป็นการค้า สามารถกระทำโดยใช้เครื่องมือโปรแกรมอุปกรณ์หรือใช้เครื่องมือทดสอบอื่น ๆ หลังจากทำการทดสอบเรียบร้อยแล้ว เพื่อป้องกันการลอกเลียนแบบ อาจจะทำการโปรแกรมซีเคียวริตี้บิต (security bit)

บทที่ 3

Generic Array Logic (GAL)

จากอุปกรณ์ไอซีตระกูล PAL (Programmable Array Logic)ซึ่งเป็นอุปกรณ์ที่ทางโรงงานผู้ผลิตได้บรรจุวงจรเกตพื้นฐานแบบต่างๆ เช่น แอนด์เกต ออร์เกต และอินเวอร์เตอร์รวมไว้ในตัวเดียวกัน ผู้ใช้สามารถออกแบบเชื่อมโยงวงจรเกตเหล่านี้เข้าด้วยกันได้เพื่อให้วงจรดิจิตอลลอจิกมีฟังก์ชันการทำงานตามความต้องการ อุปกรณ์ที่สามารถโปรแกรมการทำงานได้นี้จะมีชื่อเรียกรวมกันว่า PLD (Programmable Logic Device) ซึ่งมีอยู่หลายชนิดด้วยกัน และ PAL ก็เป็นอุปกรณ์ชนิดหนึ่งในจำพวก PLD จากไอซี PAL ที่สามารถเลือกชนิดการทำงานของเอาท์พุทแบบต่างๆกันได้ เช่น ทำงานที่ลอจิก "H" (active high) หรือ "L" (active low)หรือเป็นแบบรีจิสเตอร์ที่สามารถออกแบบวงจรซีเควนเชียล (sequential) ได้ ทำให้มีการพัฒนามาเป็น Generic Array Logic (GAL®) โดยที่ GAL 1 ตัวสามารถแทน PAL แบบต่างๆได้มากมาย ซึ่งในที่นี้จะกล่าวถึง GAL16V8 และ GAL20V8 ของบริษัท Lattice semiconductor เท่านั้น ลักษณะการจัดขาของ GAL16V8 และ GAL20V8 แสดงได้ดังรูปที่3.1



รูปที่3.1 การจัดขาและลักษณะตัวถังของ GAL16V8 และ GAL20V8

โดยทั้งสองเบอร์มีคุณสมบัติต่างๆโดยสังเขป ดังนี้

- ใช้เทคโนโลยี E^2CMOS (Electrically Erasable CMOS) ในการผลิต มีข้อดี คือ

- เวลาหน่วง (propagation delay) สูงสุด 15 นาโนวินาที
- ใช้งานความถี่สูงสุด 50 MHz
- เวลาที่ใช้สูงสุดจาก clock ถึงการเปลี่ยนแปลงที่เอาท์พุท 12 นาโนวินาที
- คุณสมบัติทางเอาท์พุทเหมือนกับไอซีทีทีแอล (รับกระแสได้ 24mA)

- โครงสร้างเป็น E^2CELL ทำให้

- สามารถกำหนดลอจิกการทำงานใหม่ได้

- ใช้ความเร็วในการลบและโปรแกรมด้วยไฟฟ้าน้อยกว่า 50 มิลลิวินาที

- เก็บรักษาข้อมูลที่โปรแกรมได้นานถึง 20 ปี

- มีเอาท์พุทลอจิกมาโครเซลล์หรือ OLMC(Output Logic Macrocell) 8 เอาท์พุททำให้

- สะดวกต่อการออกแบบวงจรลอจิกที่ซับซ้อนมากได้

- สามารถโปรแกรมเอาท์พุทให้ทำงานที่ลอจิก "H" หรือ "L" ได้

- GAL16V8 สามารถนำมาใช้งานแทน PAL ขนาด 20 ขาได้หลายชนิด และ GAL20V8 สามารถนำมาใช้งานแทน PAL ขนาด 24 ขาได้เต็มฟังก์ชันการทำงาน

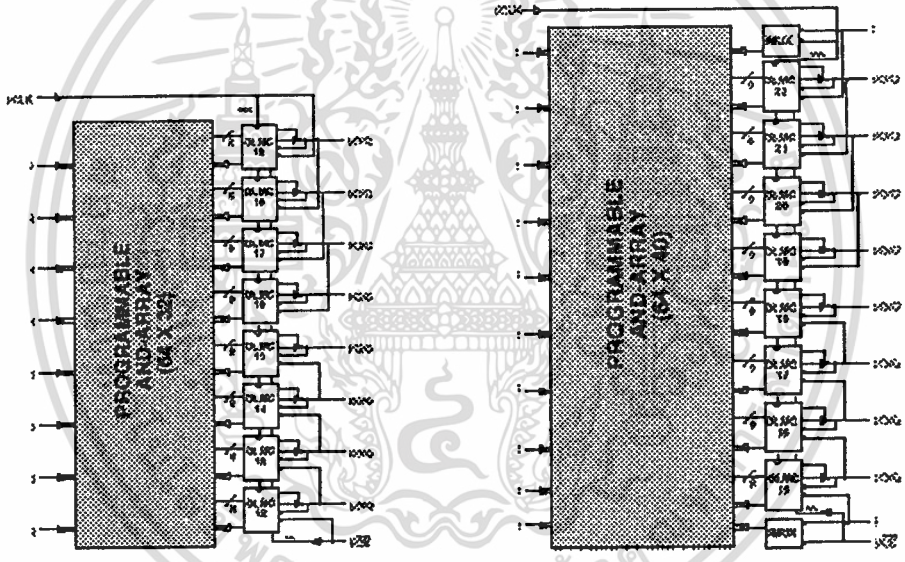
- มีการปรีโหลด(preload)และเพาเวอร์ออนรีเซต (power-on reset) ของทุกรีจิสเตอร์

PAL เบอร์ต่างๆที่สามารถใช้ GAL20V8 แทนได้	PAL เบอร์ต่างๆที่สามารถใช้ GAL16V8 แทนได้
20L8	16L8
20H8	16H8
20R8	16R8
20R6	16R6
20R4	16R4
20P8	16P8
20RP8	16RP8
20RP6	16RP6
20RP4	16RP4
14L8	10L8
16L6	12L6
18L4	14L4
20L2	16L2
14H8	10H8
16H6	12H6
18H4	14H4
20H2	16H2
14P8	10P8
16P6	12P6
18P4	14P4
20P2	16P2

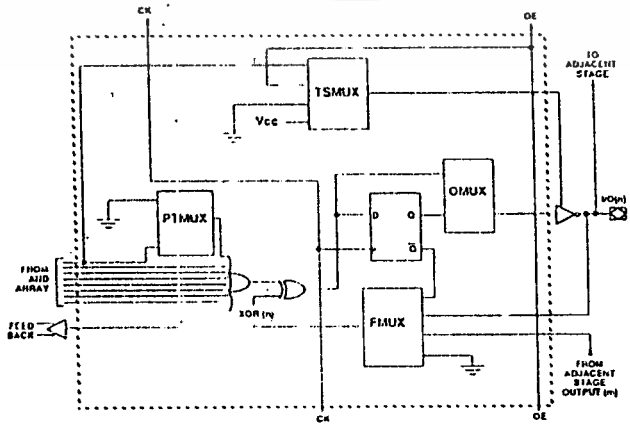
ตารางที่3.1 แสดงไอซี PAL เบอร์ต่างๆที่สามารถใช้ GAL16V8 และ GAL20V8 แทนได้

ความสามารถในการใช้ GAL16V8 และ GAL20V8 แทน PAL เบอร์ต่างๆได้สรุปไว้ในตารางที่3.1 โดยจะเห็นได้ว่า GAL 1 ตัวสามารถแทน PAL ได้ถึง 21 เบอร์ และด้วยเทคโนโลยี E^2CMOS ในการผลิตจึงสามารถลบและเขียนโปรแกรมใหม่ได้ถึง 100 ครั้ง โดยใช้กระแสไฟฟ้าในการลบโปรแกรม นอกจากนี้ GAL ยังกินกระแสน้อยเพียงครึ่งหนึ่งของแบบไบโพลาร์ธรรมดา (50% หรือ 75 mA ในแบบ half power) และยังสามารถเลือกแบบกินกระแสเพียง 1/4 เท่าได้ด้วย (40mA ในรุ่น quarter power)

จากบล็อกไดอะแกรมของ GALในรูปที่3.2 จะมีวงจรที่ชื่อ เอาท์พุทลอจิกมาโครเซลล์ (OLMC) ซึ่ง OLMC นี้จะเป็นหัวใจในการที่ GAL สามารถใช้แทน PAL ได้หลายเบอร์



รูปที่3.2 ลอจิกไดอะแกรมของ GAL16V8 และ GAL20V8



รูปที่3.3 บล็อกไดอะแกรมของเอาท์พุทลอจิกมาโครเซลล์

3.1 เอาท์พุทลอจิกมาโครเซลล์ (OLMC)

บล็อกไดอะแกรมของเอาท์พุทลอจิกมาโครเซลล์แสดงในรูปที่ 3.3 เอาท์พุทของแอนด์เกตในส่วนแอนด์อะเรย์จะป้อนเข้ามายัง OLMCแล้วใช้โปรแกรมการทำงานของ OLMC ได้ 3 โหมด เพื่อให้ได้ OLMC แบบต่างๆตามต้องการ ซึ่งการโปรแกรมเลือกโหมดการทำงานนี้จะถูกทำโดยซอฟท์แวร์หรือฮาร์ดแวร์ที่ใช้ในการออกแบบ GAL เองโดยอัตโนมัติ OLMC มีการทำงานแบ่งเป็น 3 โหมด คือ

1. ซิมเปิลโหมด (Simple mode)
2. คอมเพล็กซ์โหมด (Complex mode)
3. รีจิสเตอร์โหมด (Registered mode)

โดยมีบิต SYN และ ACO เป็นตัวควบคุมการทำงานของแต่ละโหมดของ OLMC แต่ละตัว XOR บิท ของแต่ละ OLMC จะเป็นตัวกำหนดโพลาริตี้ของเอาท์พุท ในขณะที่บิต AC1 เป็นตัวกำหนดลักษณะของ I/O

3.2 COMPILER SUPPORT OLMC

ซอฟท์แวร์คอมไพเลอร์ที่ทำการสนับสนุนการทำงานของ OLMC แต่ละโหมดจะเป็นไปตามชนิดของอุปกรณ์ คอมไพเลอร์ส่วนใหญ่จะมีความสามารถในการเลือกชนิดของอุปกรณ์โดยอัตโนมัติ โดยทั่วไปจะเป็นการใช้รีจิสเตอร์ หรือ เอาท์พุทอีนาเบิ้ล (OE)

การใช้รีจิสเตอร์ บนอุปกรณ์ทำให้ซอฟท์แวร์เลือกการทำงานแบบรีจิสเตอร์โหมด ส่วนการรวมของเอาท์พุททุกตัวกับเอาท์พุทอีนาเบิ้ล ที่ถูกควบคุมโดยโปรดักเทอม(product term) ทำให้ซอฟท์แวร์เลือกการทำงานแบบคอมเพล็กซ์โหมด และซอฟท์แวร์จะทำการเลือกการทำงานแบบซิมเปิลโหมด เมื่อเอาท์พุททุกตัวไม่มีการเชื่อมต่อกับเอาท์พุทอีนาเบิ้ล

ในการใช้ซอฟท์แวร์คอมไพเลอร์ในการกำหนดลักษณะของอุปกรณ์ ผู้ใช้ควรเข้าใจการทำงานของแต่ละโหมด ดังนี้

1. ซิมเปิลโหมด สำหรับ GAL16V8 ขา feedback ของเอาท์พุททุกขาจะถูกต่อผ่านจากขาที่อยู่ใกล้เคียง จึงทำให้ขา 15 และ 16 ไม่มีการป้อนกลับ(feedback) และจะถูกใช้เป็นเอาท์พุท

2. คอมเพล็กซ์โหมด สำหรับ GAL16V8 ขา 1 และขา 11 จะทำหน้าที่เป็นอินพุท และมีการป้อนกลับ จากขา 19 และ 12 เนื่องจากขา 12 และ 19 ถูกใช้ป้อนกลับ ดังนั้นจึงไม่มีการป้อนกลับในโหมดนี้

3. รีจิสเตอร์โหมด สำหรับ GAL16V8 ขา 1 และ ขา 11 จะทำหน้าที่เป็นสัญญาณนาฬิกา และเอาท์พุทอีนาเบิ้ลตามลำดับ ไม่สามารถนำไปใช้งานเป็นอินพุทได้

3.3.1 ซิมเปิลโหมด

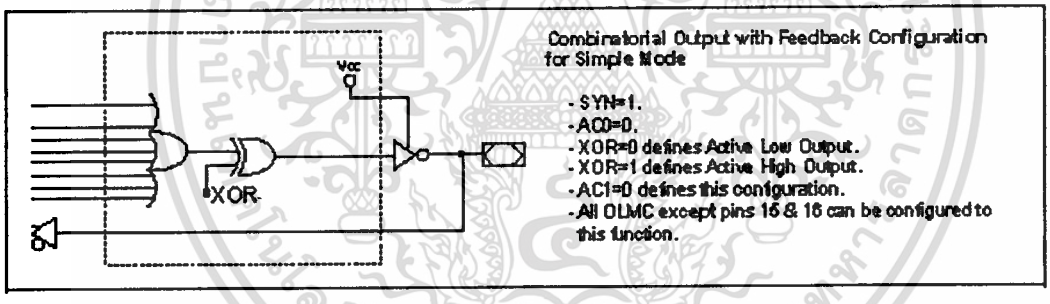
ในโหมดนี้ OLMC จะมีลักษณะการต่อวงจรดังรูปที่ 3.4 ในแต่ละขาของ GALจะทำหน้าที่เป็นอินพุทหรือเอาท์พุทได้เพียงอย่างเดียว

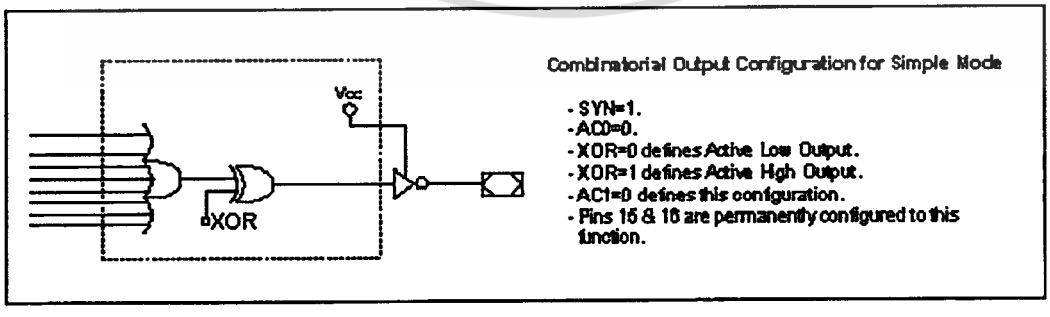
โครงสร้างเอาท์พุทจะเหมือนกับ PAL10L8 18H4 และ 16P6 แต่สามารถโปรแกรมอินพุทและเอาท์พุทให้ทำงานที่ลอจิก "L" หรือ "H" ได้

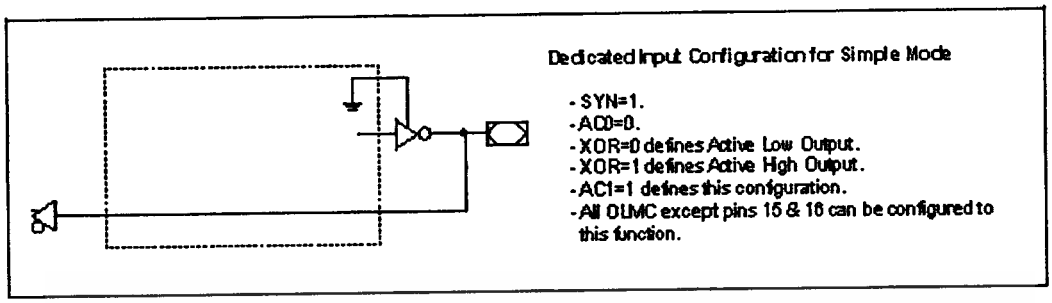
เอาท์พุททั้งหมดจะเป็นของข้อมูลจากโปรดักเทอมจากเอาท์พุททั้ง 6 ของแอนด์เกต มาเข้าออร์เกต โดยแต่ละเอาท์พุทสามารถโปรแกรมให้ทำงานที่ลอจิก "L" หรือ "H" ได้

ในโหมดนี้ขา OE และ CLK (ขา 1 และ 11 ของ GAL16V8, ขา 1 และ 13 ของ GAL20V8) จะไม่ได้ใช้ ทำให้ใช้เป็นขาอินพุทป้อนสัญญาณเข้าแอนด์อะเรย์ได้

แต่ OLMC ตัวที่อยู่ตรงกลางทั้งสองตัว (ขา 15 และ 16 ของ GAL16V8 , ขา 18 และ 19 ของ GAL20V8) จะไม่สามารถใช้เป็นขาอินพุทได้







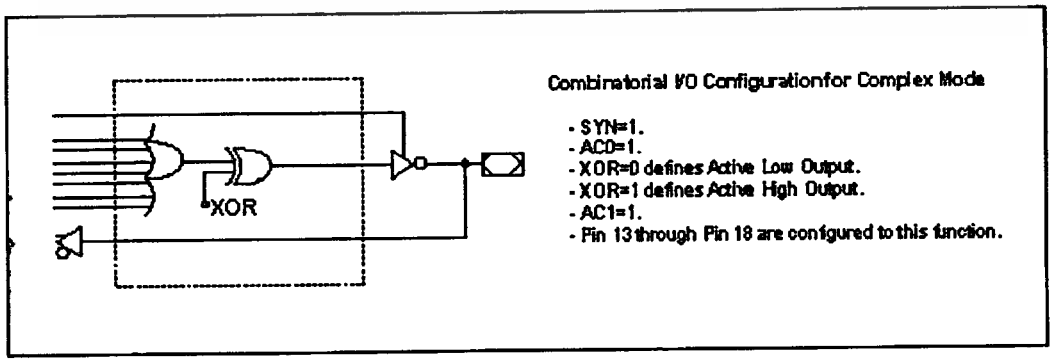
รูปที่3.4 ลักษณะการต่อวงจรภายในของ OLMC ในซิมเปิลโหมด

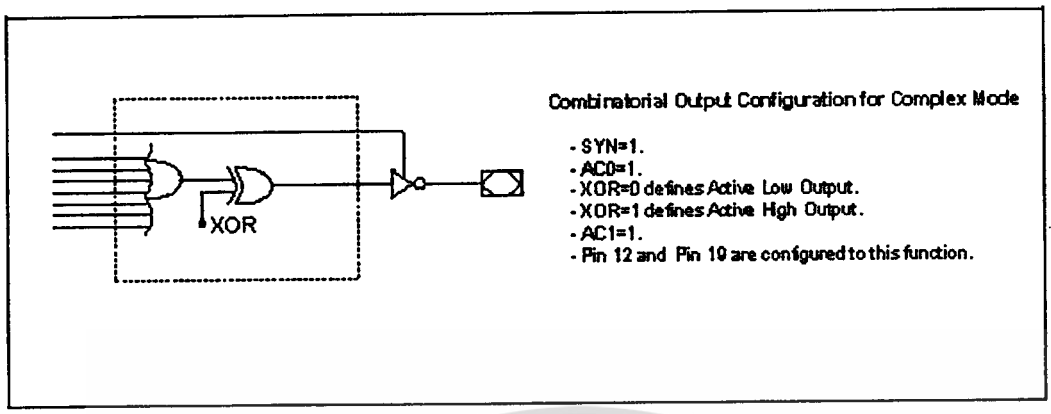
3.3.2 คอมเพล็กซ์โหมด

ในโหมดนี้ OLMC จะมีลักษณะการต่อวงจรดังรูปที่ 3.5 โดยจะต่อเป็นเอาท์พุทอย่างเดียว หรือต่อเป็นอินพุท/เอาท์พุทก็ได้ เหมือนกับ PAL16L8 20L8 และ 16P8

การจัดโครงสร้างในลักษณะนี้จะด้วยการโปรแกรมโพลาริตี้ลงในแต่ละ OLMC ในโหมดนี้จะมีขาที่เป็นอินพุท/เอาท์พุทได้สูงสุด 6 ตัว โดย OLMC อีก 2 ตัว (ขา 12 และ 19 ของ GAL16V8 , ขา 15 และ 22 ของ GAL20V8) จะเป็นเอาท์พุทได้อย่างเดียว สัญญาณอินพุทหรือเอาท์พุทจะสามารถใช้เป็นส่วนหนึ่งของอินพุท/เอาท์พุทได้

เอาท์พุทของ OLMC ทั้งหมดจะได้จากโปรดักเทอม 7 เทอมต่อ 1 เอาท์พุทโดยจะต้องใช้โปรดักเทอม 1 เทอมในการโปรแกรมสัญญาณเอาท์พุทอีนาเบิ้ล ซึ่งจะเหมือนกับซิมเปิลโหมดที่ขา 1 และ 11 ของ GAL16V8 ,ขา 1 และ 13 ของ GAL20V8 สามารถใช้เป็นขาอินพุทป้อนข้อมูลไปยังอะเรย์ได้





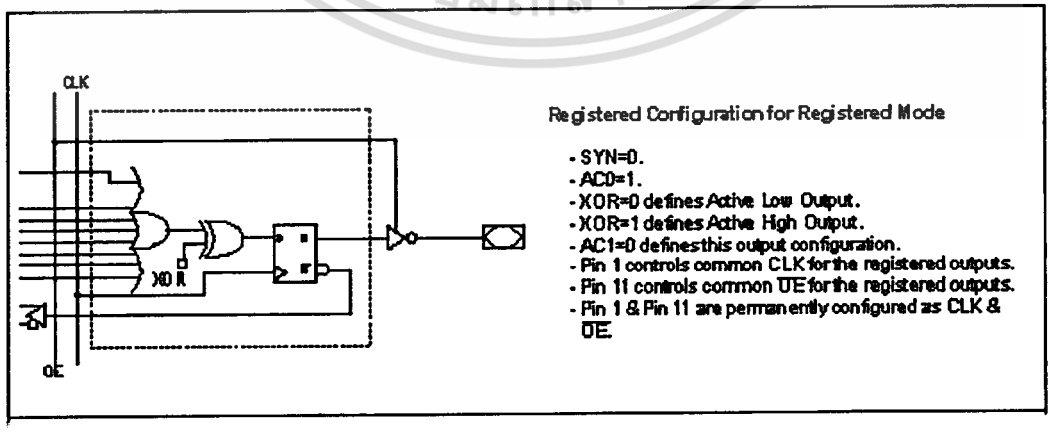
รูปที่3.5 ลักษณะการต่อวงจรภายในของ OLMC ในคอมเพล็กซ์โหมด

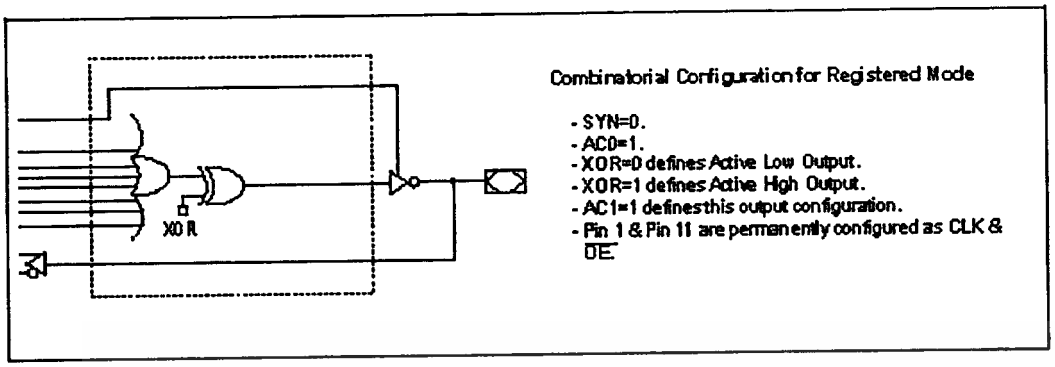
3.3.3 รีจิสเตอร์โหมด

ในโหมดนี้ OLMC จะมีลักษณะการต่อวงจรดังรูปที่ 3.6 เอาท์พุทของ OLMC จะสามารถต่อเป็นรีจิสเตอร์เอาท์พุทหรือเป็นอินพุท/เอาท์พุทฟังก์ชันได้ จากลักษณะของโหมดนี้จะคล้ายกับ PAL16R8 20R8 และ 16RP4

OLMC ทั้งหมดจะใช้สัญญาณ OE และ CLK ร่วมกัน OLMC แต่ละตัวจะสามารถทำงานเป็นรีจิสเตอร์หรืออินพุท/เอาท์พุทได้ โดยสามารถมีได้สูงสุด 8 ตัว สัญญาณอินพุทหรือเอาท์พุทสามารถนำมาใช้เป็นส่วนหนึ่งของอินพุท/เอาท์พุทได้

ถ้าต่อเป็นรีจิสเตอร์เอาท์พุท เอาท์พุทที่ได้จะเป็นของ 8 โปรดักเทอมต่อ 1 เอาท์พุท ถ้าต่อเป็นอินพุท/เอาท์พุทจะมี 7 โปรดักเทอมต่อ 1 เอาท์พุท





รูปที่3.6 ลักษณะการต่อวงจรภายในของ OLMC ในรีจิสเตอร์โหมด

3.4 อิเล็กทรอนิกส์ซิกเนเจอร์ (Electronic Signature: ES)

อิเล็กทรอนิกส์ซิกเนเจอร์ จะมีอยู่ใน GAL16V8 และ GAL20V8 ทุกตัว ES นี้จะเป็นหน่วยความจำที่สามารถโปรแกรมใหม่ได้ มีขนาด 64 บิต เพื่อให้ผู้ใช้สามารถป้อนข้อมูลของแต่ละคนได้ ES นี้จะไม่เกี่ยวข้องกับสถานะของซีเคียวริตี้เซลล์ (Security Cell)

ภายใน ES จะมีข้อมูลของเช็คซัม(checksum)ไว้ตรวจสอบความถูกต้องของข้อมูล เมื่อเปลี่ยนข้อมูลของ ES จะทำให้ checksum เปลี่ยนค่าได้

3.5 ซีเคียวริตี้เซลล์ (Security Cell)

ซีเคียวริตี้เซลล์ มีไว้เพื่อป้องกันไม่ให้ผู้อื่นมาลอกเลียนแบบข้อมูลภายในตัวGALของเราได้ หลังจากที่โปรแกรมเซลล์นี้แล้ว จะทำให้ไม่สามารถอ่านข้อมูลภายในอะเรย์ว่ามีการต่ออย่างไร เซลล์นี้จะถูกลบได้อย่างเดียวด้วยการลบแบบบัลค์ (Bulk erase cycle) ซึ่งข้อมูลทั้งหมดก็จะถูกลบไปพร้อมกันด้วย

3.6 อินพุทบัฟเฟอร์ (Input buffer)

อินพุทบัฟเฟอร์ของ GAL ทั้งสองเบอร์นี้เพื่อให้ใช้กับระดับแรงดันลอจิกของทีทีแอลได้ โดยจะมีอิมพีแดนซ์สูงทำให้กินกระแสจากวงจรที่นำมาต่อน้อย นอกจากนี้เอาท์พุทยังสามารถนำไปต่อขับอุปกรณ์ลอจิกอื่นๆได้มากกว่าทีทีแอลแบบไบโพลาร์ทั่วไปด้วย

ที่อินพุทแต่ละตัวไม่มีการต่อพูลอัพภายใน ดังนั้นขาอินพุทและขาไอโอแบบสามสถานะที่ไม่ได้ใช้ควรนำไปต่อกับอินพุทอื่นที่ใช้งาน หรือต่อกับไฟเลี้ยงหรือกราวด์เพื่อลดระดับสัญญาณรบกวนและช่วยให้กินกระแส I_{cc} น้อยลง

3.7 เอาท์พุทรีจิสเตอร์ปรีโหลด (Output Register Preload)

ในการทดสอบการทำงานในซีเควนเชียลโหมด จะสามารถตั้งค่าเอาท์พุทของรีจิสเตอร์ได้ว่าต้องการให้เป็น "L" หรือ "H" ได้ เพื่อกำหนดสถานะปัจจุบัน (present state) ได้ แล้วจึงป้อนอินพุทเพื่อทดสอบสถานะถัดไปว่าจะเป็นไปตามค่าที่ต้องการหรือไม่ และยังสามารถทดสอบการใช้งานตามความเป็นจริงคือ หลังจากตั้งค่าเอาท์พุทของรีจิสเตอร์แล้วจะหยุดสัญญาณที่ผิดปกติ (ช่วงเปิดเครื่อง (power up) , ไฟเลี้ยงที่ใช้มีกลิตซ์แฝงมา ฯลฯ) เข้าไปยังรีจิสเตอร์แล้วตรวจสอบดูว่าเอาท์พุทของวงจรที่ทดสอบจะยังสามารถให้เอาท์พุทในสถานะถัดไปที่ถูกต้องได้หรือไม่

3.8 การลบข้อมูลแบบบัลค์โหมด (Bulk erase mode)

ก่อนการโปรแกรมข้อมูลจะต้องมีการลบข้อมูลเก่าทิ้งไปก่อน ซึ่งการลบข้อมูลจะเป็นไปโดยอัตโนมัติจากเครื่องที่ใช้โปรแกรม โดยจะเป็นส่วนหนึ่งของไซเคิลการเขียนข้อมูล ซึ่งใช้เวลาในการลบทั้งหมด 50 มิลลิวินาที

3.9 เพาเวอร์อัพรีเซต (Power-up reset)

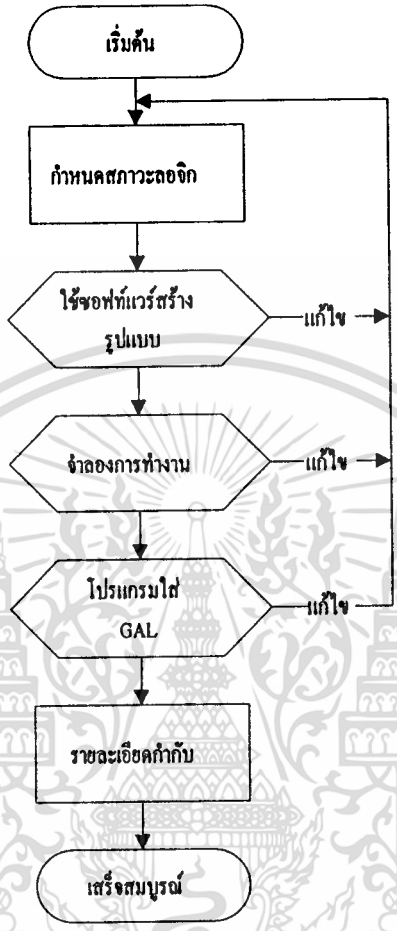
หลังจากที่ป้อนไฟเลี้ยงแล้วจะมีสัญญาณไปรีเซตรีจิสเตอร์ทุกตัว(t_{RESET} สูงสุด 45 μS) เอาท์พุท Q ของรีจิสเตอร์ทุกตัวจะถูกตั้งให้เป็น "L" ส่งผลให้ขาเอาท์พุท (ถ้ามีสัญญาณ OE มากระตุ้นให้ทำงาน) จะมีลอจิกเป็น "H" โดยไม่ขึ้นกับการกำหนดโพลาริตี้ ความสามารถนี้จะทำให้การออกแบบวงจรซีเควนเชียลได้ง่ายขึ้น โดยการกำหนดสถานะช่วงเริ่มต้นของไซเคิลการทำงานให้ปรากฏที่ช่วงเพาเวอร์อัพ

3.10 อุปกรณ์ช่วยออกแบบ

ในอดีตเครื่องมือที่ช่วยในการออกแบบและโปรแกรมอุปกรณ์พวก PLDยังไม่มี (GAL เป็นอุปกรณ์ประเภทหนึ่งของ PLD) ผู้ออกแบบจะต้องออกแบบลดทอนฟังก์ชันเองแล้วจึงนำไปเขียนในแผนที่ตำแหน่งฟิวส์ที่มีจำนวน 1000 ถึง 3000 ตำแหน่ง จากนั้นจึงป้อนเข้าเครื่องโปรแกรมทำการระเบิดฟิวส์ทีละตำแหน่ง เพื่อให้ได้ลอจิกฟังก์ชันตามที่ต้องการ

วิธีการดังกล่าวจะทำให้เกิดความผิดพลาดมาก บางทีก็หลงลืมระเบิดฟิวส์ได้ไม่ครบตำแหน่งหรือเกินตำแหน่งไป ทำให้ PLD ที่โปรแกรมใช้งานไม่ได้หรือเสียเป็นจำนวนมาก

ในปัจจุบันมีซอฟท์แวร์และฮาร์ดแวร์ (ตัวโปรแกรมเมอร์ที่ใช้โปรแกรมข้อมูลในอุปกรณ์) ซึ่งการออกแบบอุปกรณ์ PLD จะมีขั้นตอนดังแสดงในรูปที่ 3.7



รูปที่ 3.7 ขั้นตอนในการออกแบบอุปกรณ์ PLD

3.11 ซอฟท์แวร์

ในปัจจุบันมีซอฟท์แวร์ที่ใช้ออกแบบหลายชนิดด้วยกัน เริ่มจากระดับง่ายที่เรียกว่า แอสเซมเบลอร์(Assembler)ซึ่งไม่มีประสิทธิภาพมากนัก มาจนถึงซอฟท์แวร์ระดับสูงๆที่เรียกว่า คอมไพเลอร์ (Compiler) ที่มีคุณสมบัติพิเศษช่วยให้การออกแบบง่ายขึ้น

ซอฟท์แวร์คอมไพเลอร์ที่เป็นที่ยอดนิยม ชื่อ CUPL® และ ABEL® ของบริษัท Assisted Technology และ Data I/O Corp. โดยจะสามารถใช้ออกแบบอุปกรณ์ PLD ได้ทุกชนิด และยังใช้งานได้ง่าย

3.12 ฮาร์ดแวร์

การโปรแกรมหรือแพตเทอร์นิ่งอุปกรณ์ PLD เป็นขั้นตอนในการนำเอาข้อมูลไปโปรแกรมให้ PLD ทำงานตามฟังก์ชันที่เราต้องการ โดยการป้อนข้อมูลจะต้องป้อนพัลส์ที่มีคาบเวลาและระดับแรงดันที่เหมาะสมกับอุปกรณ์แต่ละตัว ฮาร์ดแวร์ดังกล่าวจะมีจำหน่ายให้เลือกกันหลายแบบ ฮาร์ดแวร์ที่ใช้ควรเลือกแบบที่ใช้งานได้กว้าง และสามารถใช้โปรแกรม PLDได้หลายชนิด

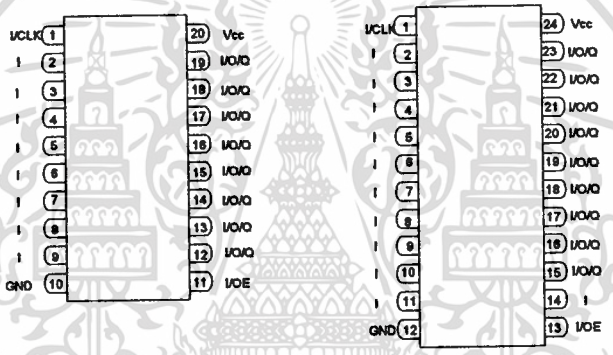
ข้อมูลที่ใช้ในการโปรแกรมจะได้จากอุปกรณ์ที่ผ่านการโปรแกรมแล้วหรือได้จากคอมพิวเตอร์ ข้อมูลที่ใช้กับเครื่องโปรแกรมส่วนใหญ่จะเป็นข้อมูลตามมาตรฐานของ JEDEC

การโปรแกรมข้อมูลของ GAL จะใช้การโปรแกรมแบบขนาน ทำให้เวลาที่ใช้โปรแกรมรวดเร็วมาก(ไม่ถึง 1 วินาที) หลังจากผ่านการโปรแกรมแล้ว ขั้นสุดท้ายของทุกๆเซลล์ของGAL จะถูกทดสอบแบบอนาลอก เพื่อทดสอบให้แน่ใจว่าทุกเซลล์ถูกโปรแกรมแล้ว และสามารถใช้งานได้ตามที่ต้องการ

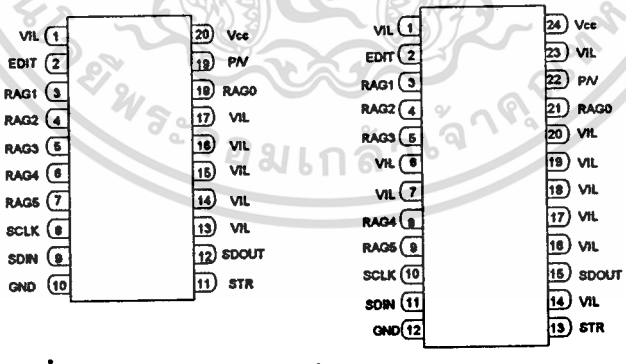
บทที่4
เครื่องโปรแกรม GAL

ในการสร้างเครื่องโปรแกรม GALนั้นจำเป็นต้องทำความเข้าใจถึงลักษณะโครงสร้างและการทำงานเพื่อสร้างฮาร์ดแวร์ และเขียนซอฟท์แวร์ให้สัมพันธ์กับลักษณะและฟังก์ชันของ GAL เบอร์นั้นๆ

การทำงานของ GAL นอกจากโหมดปกติ (normal) ดังตำแหน่งขาของฟังก์ชันต่างๆใน รูปที่ 4.1 แล้ว GALยังมีการทำงานอีกโหมดเรียกว่าโหมด Edit ซึ่งมีการจัดตำแหน่งขาดังรูปที่ 4.2



รูปที่ 4.1 การจัดตำแหน่งขาโหมด Normal ของ GAL

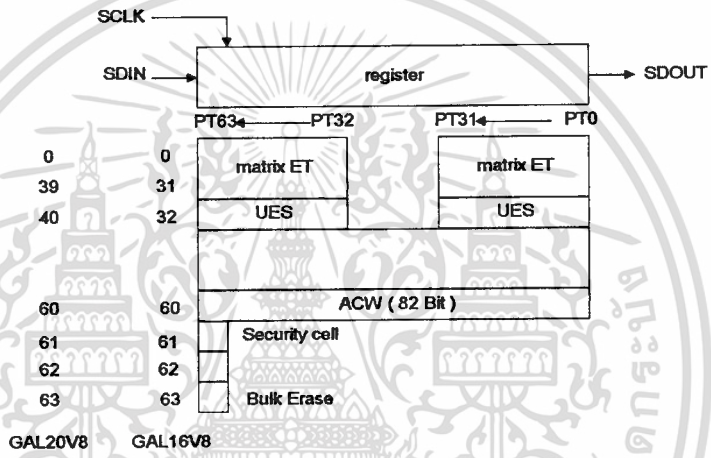


รูปที่ 4.2 การจัดตำแหน่งขาโหมด Edit ของ GAL

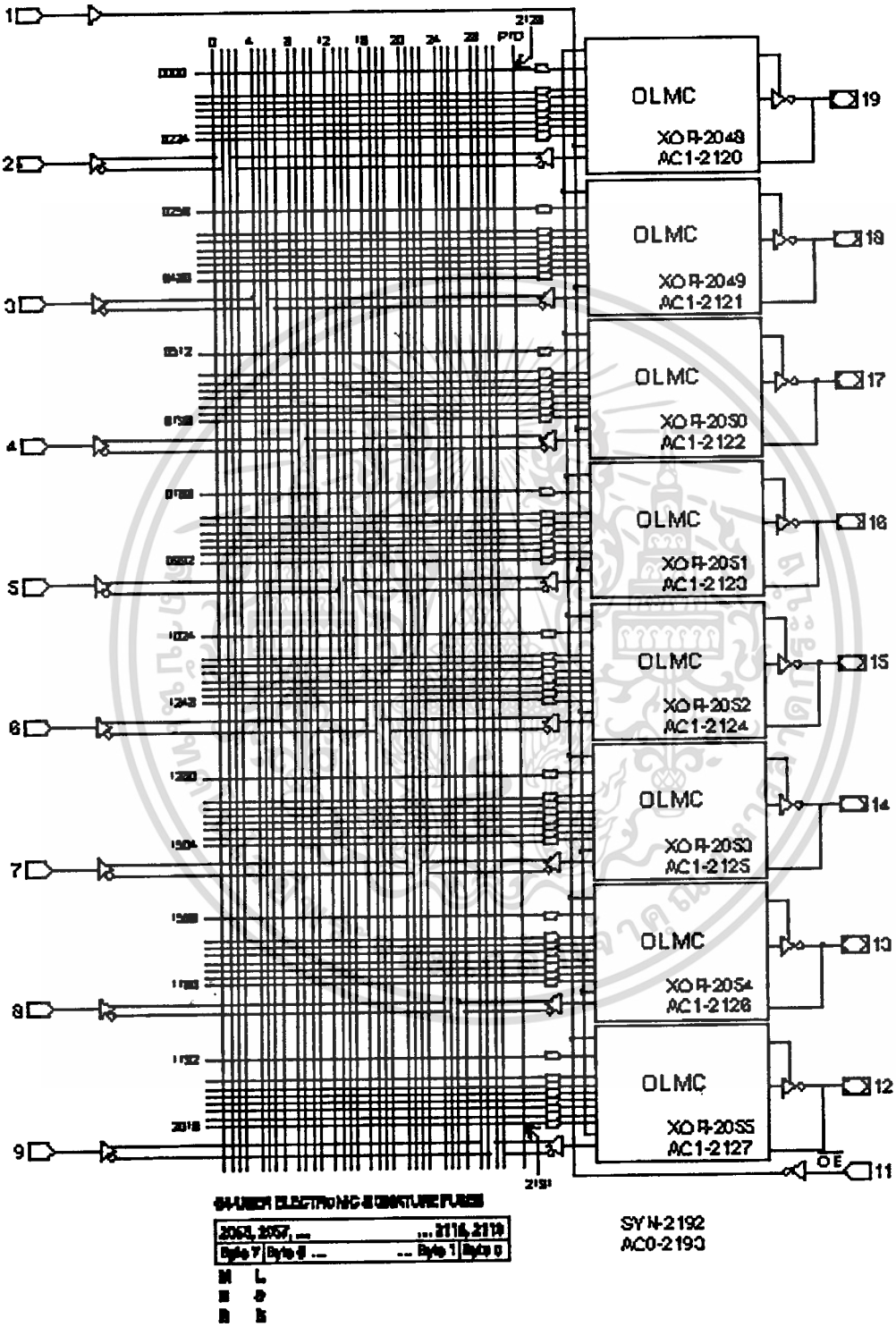
การเข้าสู่โหมด Edit ทำได้โดยการป้อนศักดา 12, 14, 14.5, 15.75, หรือ 16.5 โวลท์เข้าที่ขา EDIT ของ GAL การเปลี่ยนขนาดของศักดาดังกล่าวจะเป็นไปตามขั้นตอนของการโปรแกรม เมื่อทำการป้อนลอจิก1 เข้าที่ขา P/V (*Program/Verify*) จะเป็นการบอกว่าอยู่ในขั้นตอนการโปรแกรม ส่วนลอจิก 0 เป็นการอ่านข้อมูลจาก GAL การถ่ายเทข้อมูลเข้าและออกจาก GALจะเป็น

อนุกรมผ่านทาง ขา SDIN และ SDOUT ตามลำดับ และต้องทำการป้อนสัญญาณพัลส์เข้าที่ขา SCLK ควบคู่ไปด้วย โดยขา /STR จะเป็น '0' เมื่อมีข้อมูลเข้า GAL และขาที่ไม่ได้ใช้งานอย่างเช่น VIL ให้ต่อลงกราวด์ (GND) ผ่านตัวต้านทาน 10 กิโลโอหม์ หรือจ่ายแรงดันทำให้เป็นลอจิก 0 ในรูปที่4.3 แสดงถึงโครงสร้างและตำแหน่งของฟิวส์ใน GAL

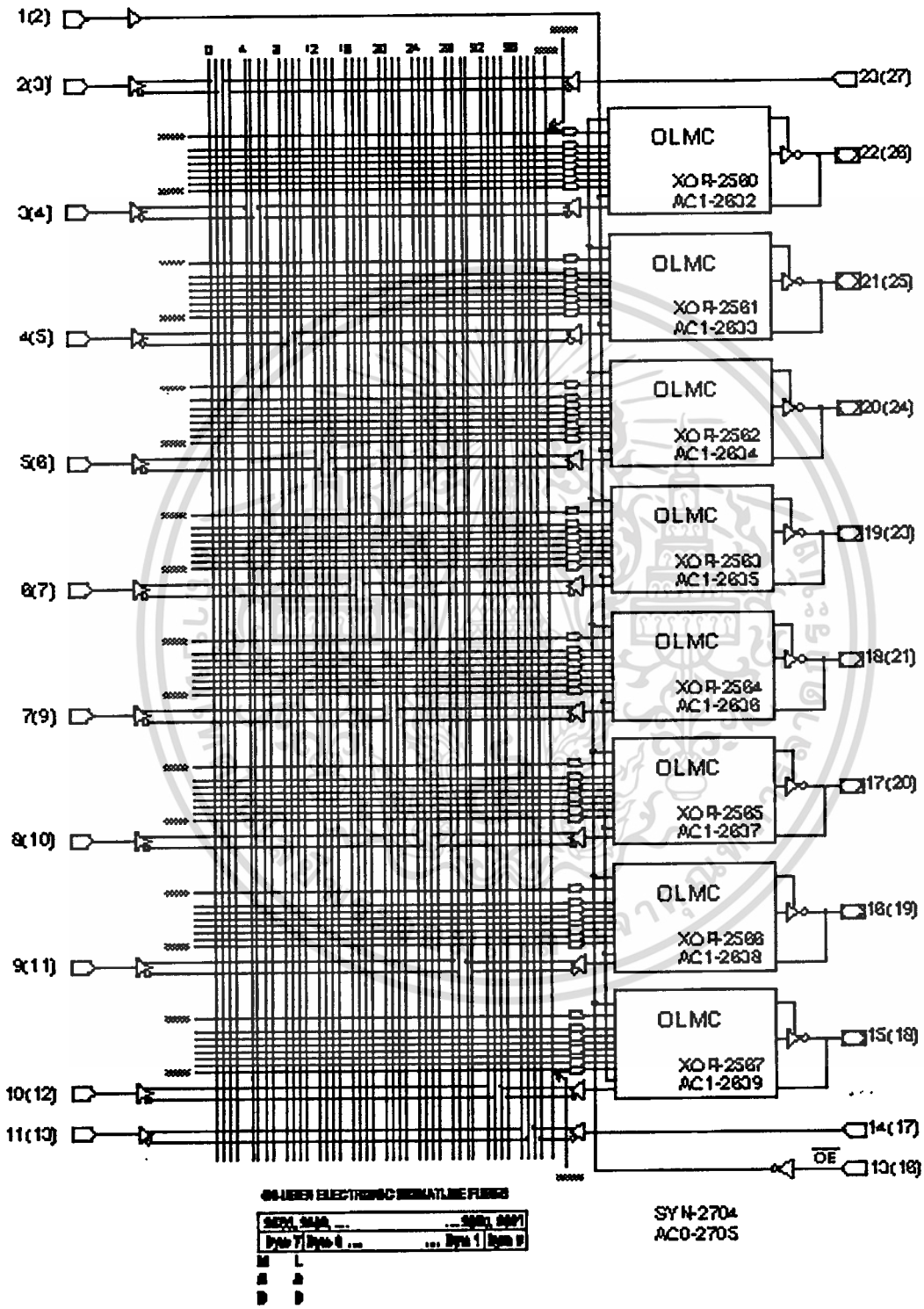
ในการโปรแกรมนั้น ก่อนอื่นจะต้องป้อน แอดเดรสของเส้นเชื่อมต่อที่ระบุในรูปที่ 4.4 และ 4.5 ลงใน RAG0 ถึง RAG 5



รูปที่ 4.3 โครงสร้าง ของ GAL 16V8 และ GAL20V8



รูปที่4.4 ลักษณะโครงสร้างภายในของ GAL16V8



รูปที่ 4.5 ลักษณะโครงสร้างภายในของ GAL20V8

ตำแหน่งบรรทัดที่ 60ที่เป็น ACW จะมีความแตกต่างตามชนิดของ GAL ดังแสดงในรูปที่ 4.6 ถึง 4.9 ซอฟท์แวร์ที่ควบคุมการโปรแกรมจะต้องจัดการให้สอดคล้องกับ ACW ตามระบุในแฟ้มข้อมูล JEDEC เราสามารถล้างฟังก์ชั่น GAL ที่โปรแกรมไว้ก่อนหน้านี้ได้โดยการโปรแกรมเช่นเดียวกันโดยใช้ตำแหน่ง บรรทัดที่ 63 (*bulk erase*)

PT63..PT32	XOR(n) 12..15	SYN	AC1(n) 12..19	AC0	XOR(n) 16..19	PT31..PT0

รูปที่4.6 โครงสร้างของ Control Word ของ GAL16V8

XOR(n) 12..15	SYN	AC1(n) 12..15	PT63..PT0	AC1(n) 16..19	AC0	XOR(n) 16..19

รูปที่4.7 โครงสร้างของ Control Word ของ GAL16V8A และ GAL16V8B

PT63..PT32	XOR(n) 15..18	SYN	AC1(n) 15..22	AC0	XOR(n) 19..22	PT31..PT0

รูปที่4.8 โครงสร้างของ Control Word ของ GAL20V8

XOR(n) 15..18	SYN	AC1(n) 15..18	PT63..PT0	AC1(n) 19..22	AC0	XOR(n) 19..22

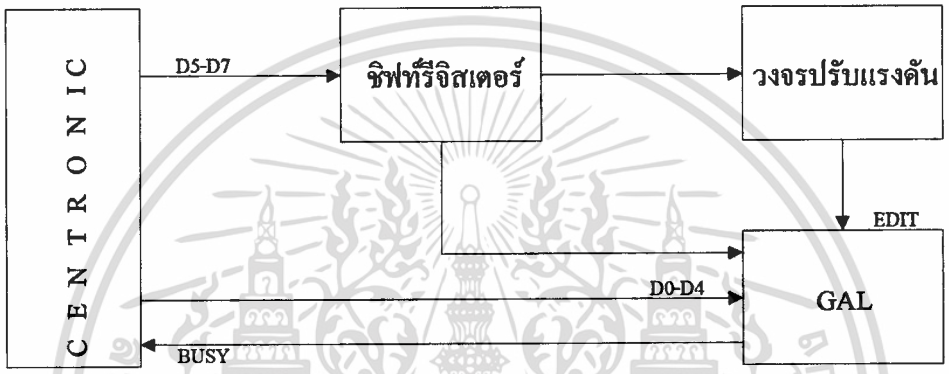
รูปที่4.9 โครงสร้างของ Control Word ของ GAL20V8A และ GAL20V8B

4.1 อุปกรณ์ที่ใช้สำหรับการโปรแกรม

รูปที่4.10 แสดงถึงบล็อกไดอะแกรมของเครื่องโปรแกรม GAL อุปกรณ์ที่ใช้สำหรับการโปรแกรม GALที่สร้างขึ้นนี้ใช้ต่อพ่วงกับพอร์ตขนาน (centronic) โดยข้อมูลบิต D5-D7 จะไปควบคุมชิฟท์รีจิสเตอร์ 2 ตัวพร้อมแลทช์ ด้วยวิธีนี้เมื่อรวมกับเอาท์พุทที่เหลือ 5 เส้นจะทำให้ได้บัสเอาท์พุท ทั้งหมด 16 เส้นเอาท์พุท จากส่วนนี้จะถูกนำไปใช้ในการเลือกใช้ศักดาของวงจรปรับแรงดันในการทำงานในโหมดต่างๆผ่านทางวงจรทรานซิสเตอร์หรืออินเวอร์เตอร์เกตแบบ open collector

ซอฟท์แวร์ที่ใช้ร่วมจะรันบนเครื่องไมโครคอมพิวเตอร์ IBM® Compatible ภายใต้ระบบ

ปฏิบัติการ MS-DOS โดยใช้ภาษาซีในการควบคุมบิตข้อมูลผ่านทางพอร์ตขนาน สายเคเบิลที่ต่อระหว่างเครื่องโปรแกรมและไมโครคอมพิวเตอร์จะใช้แบบ 25 หรือ 26 เส้นก็ได้แต่ไม่ควรยาวเกินกว่าหนึ่งเมตร และใช้คอนเนคเตอร์ แบบ Sub-D 25 ขั้วต่อเข้าทางพอร์ตเครื่องพิมพ์ อีกปลายด้านหนึ่งที่ต่อกับเครื่องโปรแกรมจะเป็นคอนเนคเตอร์แบบ 26 ขั้ว เพื่อความปลอดภัยของอุปกรณ์ ในขณะใช้งานควรต่อคอนเนคเตอร์เข้ากับคอมพิวเตอร์ให้เรียบร้อยก่อน เพื่อว่าวงจรจะได้รับการป้อนไฟพร้อมไปกับคอมพิวเตอร์



รูปที่ 4.10 บล็อกไดอะแกรมของเครื่องโปรแกรม GAL

จากวงจรในรูปที่ 4.11 การสร้างแหล่งจ่ายศักดาสำหรับการโปรแกรม จะใช้ TL497 โดยการจัดวงจรแบบมาตรฐาน วงจรจะจ่ายศักดาที่ขา 6 ได้ตามสมการ:

$$1.2V.\frac{R_5+R_6}{R_6}=1.2V.\frac{27k\Omega+1.1k\Omega}{1.1k\Omega}=30.65V$$

โดยมีศักดาอินพุท 5 โวลท์ ศักดานี้จะจ่ายเป็นไฟเลี้ยงให้กับออปแอมป์และที่ขาอินพุทนอนอินเวอร์ติ้งของออปแอมป์ตัวแรก โดยใช้ซีเนอร์ไดโอดD5 จำกัดแรงดันที่ 22 โวลท์ ศักดานี้จะไปปรากฏที่เอาท์พุทของออปแอมป์ด้วยเช่นเดียวกัน จากนี้ใช้วงจรแบ่งแรงดันซึ่งประกอบขึ้นด้วย R8 R9 และ P1 โดยที่ในสภาวะปกติเมื่อผ่าน R8 แล้วจะได้ศักดาที่ 16.5 โวลท์ ค่าศักดานี้จะไปปรากฏที่เอาท์พุทของออปแอมป์ตัวที่ 2 ค่าศักดานี้จะแม่นยำหรือไม่ขึ้นอยู่กับความเที่ยงตรงของซีเนอร์ไดโอด และการปรับค่า P1 และ R9ซึ่งอยู่ในช่วง 6.8 ถึง8.2 กิโลโอหม์ โดยใช้อินเวอร์เตอร์เกตสองตัวแบบ open collector กับทรานซิสเตอร์ T9 มาต่อเป็นบัฟเฟอร์ให้ R9กับ P1 , R10 กับ P2 และ R11 กับ P3 ส่วนที่ทำหน้าที่ปรับแรงดันให้กับโหมด Edit จะใช้ตัวต้านทานปรับค่าได้ P1 , P2 , P3 และ P4 โดยในโหมดต่างๆจะใช้ระดับแรงดันไฟฟ้าดังต่อไปนี้

- โหมดการโปรแกรมและการลบใช้แรงดันไฟฟ้า 14.5 โวลท์ จะต้องส่งข้อมูลออกจาก IC8 โดย Q5 , Q6 และ Q7 ของ IC8 มีค่าเป็น 1 ,0 และ 0 ตามลำดับ

- โหมดการตรวจสอบบิตข้อมูลจะใช้แรงดันไฟฟ้า 12 โวลท์ จะต้องส่งข้อมูลออกจาก IC8 โดย Q5 , Q6 และ Q7 ของ IC8 มีค่าเป็น 0 ,1 และ 1 ตามลำดับ

ในการทำงานของ GAL20V8 จะต้องส่งค่าข้อมูลเข้า HC595 ดังนี้

IC8 : บิต Q3 และ Q4 มีค่าเป็น '0' และ '1' ตามลำดับเพื่อให้แรงดันไฟฟ้าเข้าที่ขา EDIT ในโหมดการทำงานต่างๆตามที่ต้องการ

: บิต Q2 จะเป็นบิตกำหนด RAG1

: บิต Q1 จะเป็น '0' เพราะขา VIL ไม่ได้นำมาใช้งาน

: บิต Q0 เป็น '1' เพื่อให้ LED สว่าง

IC5 : บิต Q5 , Q0 , Q1 และ Q7 จะเป็นบิตกำหนด RAG0 , RAG2 , RAG3 และ RAG4 ตามลำดับ

: บิต Q2 และ Q3 จะเป็น '0' เพราะขา VIL ไม่ได้นำมาใช้งาน

: บิต Q4 จะเป็นบิต P/V ถ้าเป็น '1' จะเป็นการบอกว่าอยู่ในขั้นตอนการโปรแกรม ส่วนลอจิก 0 เป็นการอ่านข้อมูลจาก GAL

: บิต Q6 จะเป็น '0' เพื่อให้บิต D0 เป็นตัวกำหนดข้อมูลที่เข้าขา SDIN

ในการทำงานของ GAL16V8 จะต้องส่งค่าข้อมูลเข้า HC595 ดังนี้

IC8 : บิต Q2 , Q3 และ Q4 มีค่าเป็น'1', '1' และ '0' ตามลำดับเพื่อให้แรงดันไฟฟ้าเข้าที่ขา EDIT ในโหมดการทำงานต่างๆตามที่ต้องการ

: บิต Q1 จะเป็น '0' เพราะขา 24 ไม่ได้นำมาใช้งาน

: บิต Q0 จะเป็น '1' เพื่อให้ LED สว่าง

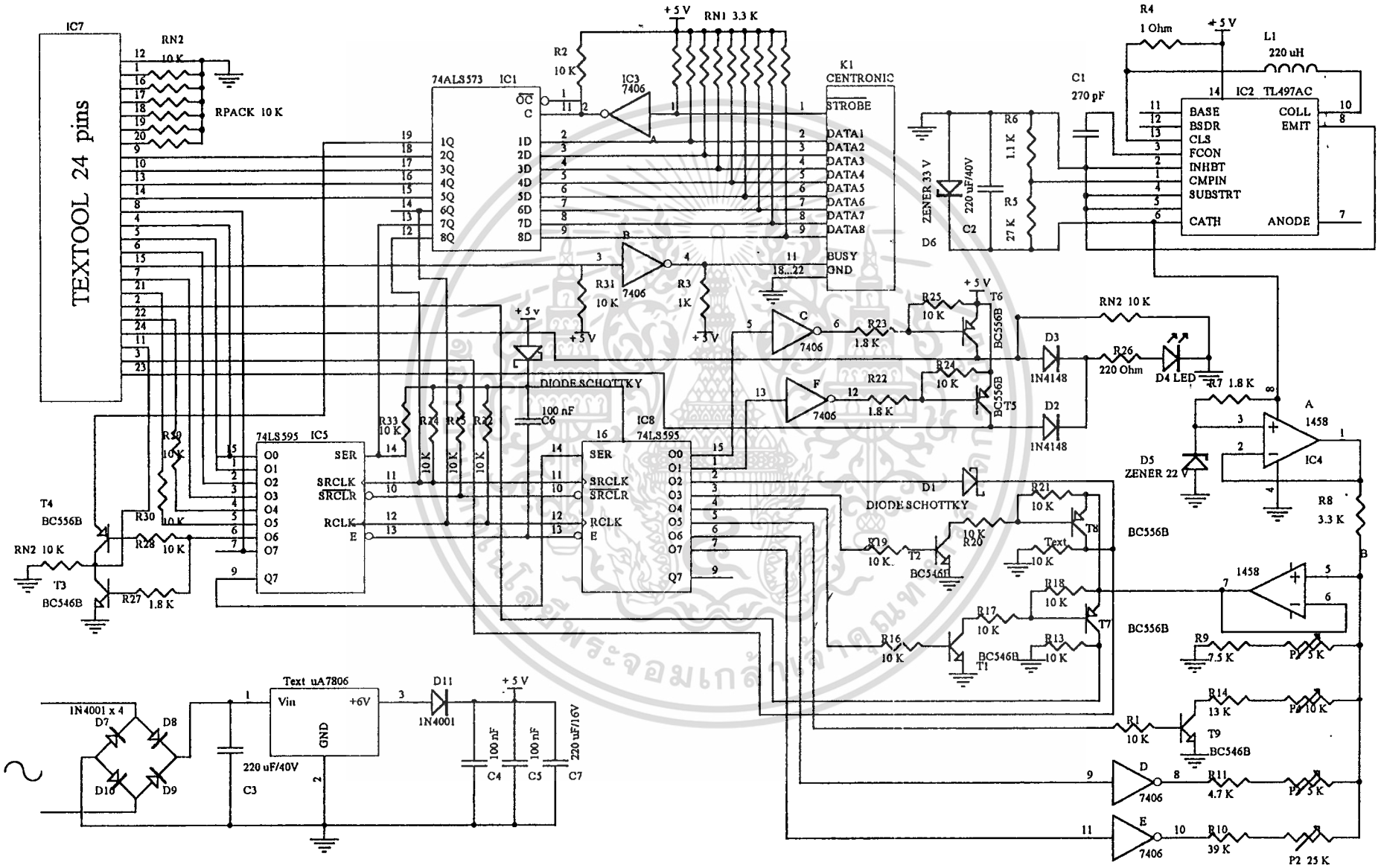
IC5 : บิต Q5 , Q0 , Q1 ,Q2, Q3และ Q7 จะเป็นบิตกำหนด RAG0 , RAG1,RAG2 , RAG3, RAG4 และ RAG5 ตามลำดับ

: บิต Q2 และ Q3 จะเป็น '0' เพราะขา VIL ไม่ได้นำมาใช้งาน

: บิต Q4 จะเป็นบิต P/V ถ้าเป็น '1' จะเป็นการบอกว่าอยู่ในขั้นตอนการโปรแกรม ส่วนลอจิก 0 เป็นการอ่านข้อมูลจาก GAL

: บิต Q6 จะเป็น '1' และบิต D0 เป็น '0' เพื่อให้ขา GND เป็น '0'

เราจะใช้ซอกเก็ต TEXTOOL ชนิดที่ใช้กับ GAL 16V8 , 16V8A และ 16V8B คือ ขา 1ของ GAL จะอยู่ที่ขา 2 ของซอกเก็ต การใช้ซอกเก็ตในลักษณะดังกล่าว จะทำให้ขา SDOUT อยู่ที่ขา 15 ของซอกเก็ตที่ไม่ขึ้นอยู่กับชนิดของ GAL ซึ่งจะส่งผ่านไปยังสาย BUSY ของพอร์ตขนานผ่านอินเวอร์เตอร์เกตแบบ open collector โดยซอกเก็ต TEXTOOL ที่ใช้อาจจะเป็นแถวเล็กหรือแบบใหญ่ที่สามารถวาง IC แบบตัวถังเล็กได้



รูปที่4.11วงจรเครื่องโปรแกรม GAL

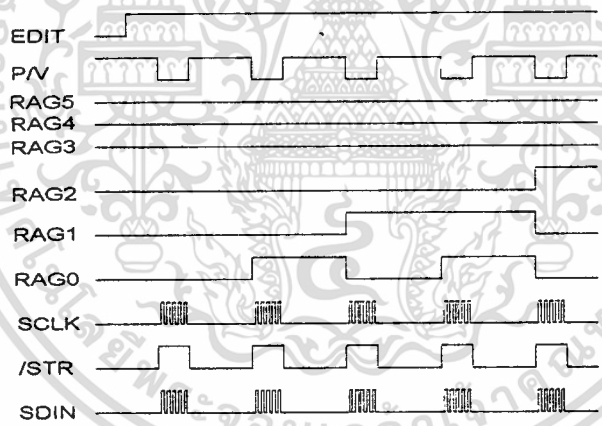
4.2 ขั้นตอนการโปรแกรม

4.2.1 การโปรแกรมเมตริกซ์ ET และ UES

- ป้อนแรงดัน 14.5 โวลท์ (Vpp) เข้าที่ขา EDIT พร้อมทั้งเซตบิต EDIT = '1'
- กำหนดแอดเดรสของเส้นเชื่อมต่อโดยใช้ RAG0 ถึง RAG5
- เซตให้บิต /STR = '1' และบิต P/V = '0'
- ทำการป้อนสัญญาณพัลส์เข้าที่ขา SCLK จำนวน 64 ลูก พร้อมทั้งส่งข้อมูลที่อ่านได้จาก JEDEC ไฟล์เข้าที่ขา SDIN
- รีเซตให้บิต /STR = '0' และบิต P/V = '1'
- เปลี่ยนแอดเดรส และทำตามขั้นตอนข้างต้นจนครบทุกแอดเดรส

4.2.2 การโปรแกรมบิต ACW

ขั้นตอนเหมือนการโปรแกรมเมตริกซ์ ET และ UES แต่จำนวนพัลส์ที่เข้าขา SCLK จะมีจำนวน 82 ลูก

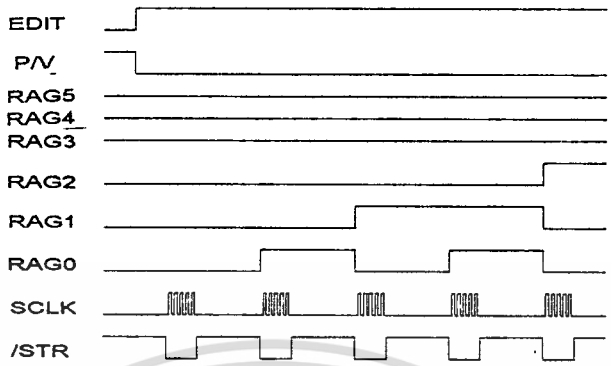


รูปที่ 4.12 programming waveform ของ GAL16V8 และ GAL20V8

4.3 ขั้นตอนการตรวจสอบบิตข้อมูล (Verification)

- ป้อนแรงดัน 12 โวลท์ (Vpp) เข้าที่ขา EDIT พร้อมทั้งเซตบิต EDIT = '1'
- รีเซตให้บิต P/V = '0'
- กำหนดแอดเดรสของเส้นเชื่อมต่อโดยใช้ RAG0 ถึง RAG5
- ป้อนสัญญาณพัลส์เข้าที่ขา SCLK 64 ลูก พร้อมทั้งรีเซตให้บิต /STR = '0'
- ทำการเปลี่ยนแอดเดรสเพื่อตรวจสอบข้อมูลของแอดเดรสถัดไปจนครบทุกแอดเดรส

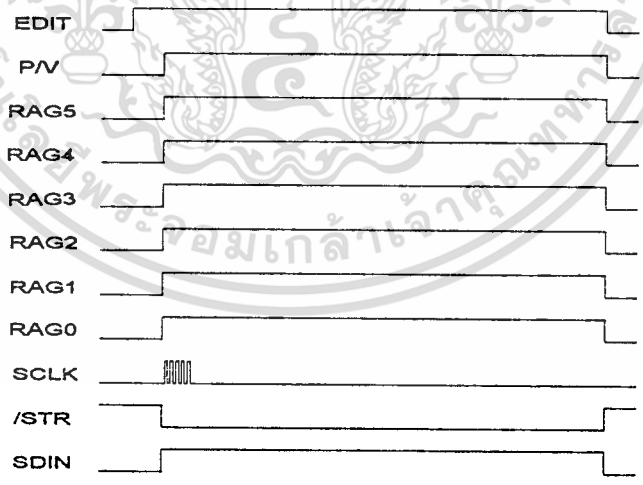
การตรวจสอบบิตข้อมูลของ ACW จะเหมือนกับเมตริกซ์ ET และ UES แต่จำนวนพัลส์ที่เข้าขา SCLK จะมีจำนวน 82 ลูก



รูปที่ 4.13 verification waveform ของ GAL16V8 และ GAL20V8

4.4 ขั้นตอนการลบ

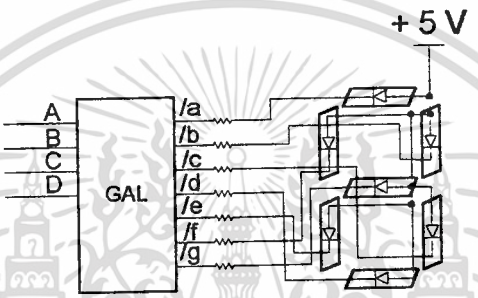
- ป้อนแรงดัน 14.5 โวลท์(Vpp) พร้อมทั้งเซตบิต EDIT = '1'
- เซตให้บิต P/V = '1'
- กำหนดแอดเดรสของเส้นเชื่อมต่อให้เท่ากับ 63 (Bulk erase)
- ป้อนสัญญาณพัลส์เข้าที่ขา SCLK 64 ลูก พร้อมทั้งเซตให้ขา SDIN = '1' และ /STR = '0'



รูปที่ 4.14 bulk erase waveform

บทที่ 5
การประยุกต์ใช้งาน GAL

เมื่อเราได้ทราบถึงขั้นตอนในการใช้เครื่องโปรแกรม GAL ในบทที่ 4 แล้ว ในบทนี้เราจะยกตัวอย่างการออกแบบวงจรลอจิกอย่างง่ายเพื่อเป็นแนวทางในการใช้งาน GAL โดยในตัวอย่างที่จะกล่าวถึงนี้ จะทำการออกแบบวงจรลอจิกที่ใช้แสดงค่าตัวเลขตั้งแต่ 0 - F ออกทาง 7 segment โดยมีวงจรดังรูปที่ 5.1

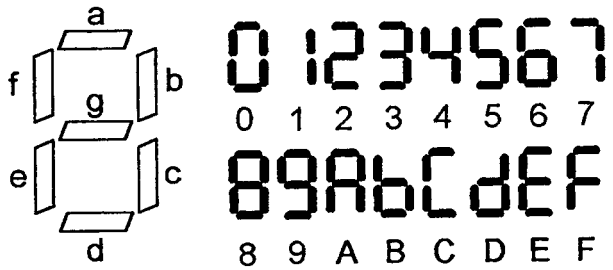


รูปที่ 5.1 วงจรแสดงค่าตั้งแต่ 0 - F ออกทาง 7 segment

ในการออกแบบวงจรลอจิกนั้น ขั้นแรกจะต้องหาสมการแสดงความสัมพันธ์ระหว่างอินพุทและเอาท์พุททั้งหมดให้ได้เสียก่อน หลังจากนั้นทำการลดรูปสมการเพื่อให้ได้สมการบูลลีน ซึ่งกรรมวิธีในการลดรูปนั้นมีหลายวิธีแล้วแต่ว่าผู้ออกแบบจะเลือกใช้วิธีใด ในปัจจุบันมีโปรแกรมที่ใช้ช่วยในการลดรูปสมการยิ่งทำให้สามารถลดรูปสมการได้สะดวกยิ่งขึ้น แต่มักจะมีปัญหาเนื่องจากโปรแกรมเหล่านี้มักจะจำกัดจำนวนอินพุท ในตัวอย่างการทดสอบนี้จะแสดงวิธีการลดรูปสมการโดยใช้แผนผังคนอร์ ซึ่งง่ายต่อการเข้าใจ เมื่อทำการลดรูปจนได้สมการบูลลีนที่ต้องการออกมาแล้ว จะต้องนำสมการบูลลีนที่ได้นั้นไปสร้างไฟล์ข้อมูล แล้วนำไฟล์ข้อมูลนั้นไปคอมไพล์ให้เป็นไฟล์ JEDEC เพื่อทำการโปรแกรมต่อไป ซึ่งโปรแกรมที่ใช้ในการคอมไพล์นั้นก็มีอยู่หลายบริษัทผลิตออกมา แล้วแต่ว่าผู้ออกแบบจะเลือกใช้โปรแกรมใด ส่วนขั้นตอนในการโปรแกรมได้กล่าวไว้อย่างละเอียดแล้วในบทที่ 4 ดังนั้นจึงจะไม่นำมากล่าวซ้ำอีก หลังจากทำการโปรแกรม GAL เรียบร้อยแล้วก็นำมาต่อลงในวงจรดังรูปที่ 5.1 เพื่อทดสอบว่า GAL ที่ได้ออกแบบนั้นใช้งานได้จริง

5.1 สมการแสดงความสัมพันธ์ระหว่างอินพุทและเอาท์พุททั้งหมดของวงจรในรูปที่ 5.1

จากวงจรในรูปที่ 5.1 เราจะสามารถเขียนสมการแสดงความสัมพันธ์ระหว่างอินพุทและเอาท์พุททั้งหมด โดยพิจารณาเหตุการณ์ทั้งหมดที่จะเกิดขึ้นในการแสดงค่าตัวเลขตั้งแต่ 0 - F ออกทาง 7 segment ดังแสดงในรูปที่ 5.2



รูปที่ 5.2 แสดงเหตุการณ์ทั้งหมดที่จะเกิดขึ้นในการแสดงค่าตัวเลขตั้งแต่ 0 - F

จากรูปข้างบน เมื่อนำอินพุทที่มีค่าเป็นตัวเลขมาแปลงให้อยู่ในรูปเลขฐาน 2 ก็จะได้ความสัมพันธ์ใหม่ ดังตารางที่ 5.1

ตัวเลข	อินพุท				เอาท์พุท						
	D	C	B	A	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
A	1	0	1	0	1	1	1	0	1	1	1
B	1	0	1	1	0	0	1	1	1	1	1
C	1	1	0	0	1	0	0	1	1	1	0
D	1	1	0	1	0	1	1	1	1	0	1
E	1	1	1	0	1	0	0	1	1	1	1
F	1	1	1	1	1	0	0	0	1	1	1

ตารางที่ 5.1 แสดงค่าความสัมพันธ์ระหว่างอินพุทในรูปเลขฐาน 2 กับเอาท์พุท

จากตารางที่ 5.1 เราจะได้สมการแสดงความสัมพันธ์ดังต่อไปนี้

$$a = 0+2+3+5+6+7+8+9+A+C+E+F$$

$$b = 0+1+2+3+4+7+8+9+A+D$$

$$c = 0+1+3+4+5+6+7+8+9+A+B+D$$

$$d = 0+2+3+5+6+8+9+B+C+D+E$$

$$e = 0+2+6+8+A+B+C+D+E+F$$

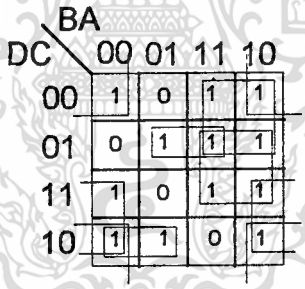
$$f = 0+4+5+6+8+9+A+B+C+E+F$$

$$g = 2+3+4+5+6+8+9+A+B+D+E+F$$

จากสมการความสัมพันธ์เหล่านี้ จะนำไปเขียนแผนผังคนอร์เพื่อทำการลดรูป

5.2 การลดรูปสมการความสัมพันธ์โดยใช้แผนผังคนอร์

จากสมการข้างบนสามารถเขียนแผนผังคนอร์ของเอาท์พุท a ได้ดังนี้

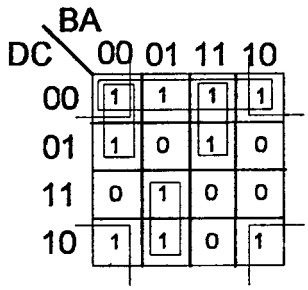


รูปที่ 5.2.1 แผนผังคนอร์ของเอาท์พุท a

เมื่อทำการลดรูปแล้วจะได้สมการบูลลีนดังนี้

a = /A&/C + B&/D + B&C + /A&D + /B&/C&D + A&C&/D

แผนผังคนอร์ของเอาท์พุท b เป็นดังรูปที่ 5.2.2

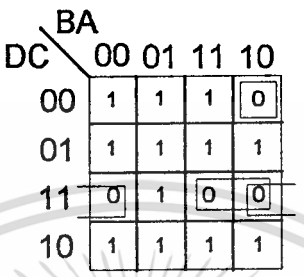


รูปที่ 5.2.2 แผนผังคนอร์ของเอาท์พุท b

เมื่อทำการลดรูปแล้วจะได้สมการบูลลีนดังนี้

b = /A&/C + /C&/D + A&B&/D + A&/B&D + /A&/B&/D

แผนผังคนอร์ของเอาท์พุท c เป็นดังรูปที่ 5.2.3

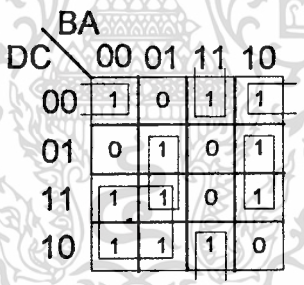


รูปที่ 5.2.3 แผนผังคนอร์ของเอาท์พุท c

เมื่อทำการลดรูปแล้วจะได้สมการบูลลีนดังนี้

/c = /A&B&/C&/D + B&C&D + /A&C&D

แผนผังคนอร์ของเอาท์พุท d เป็นดังรูปที่ 5.2.4

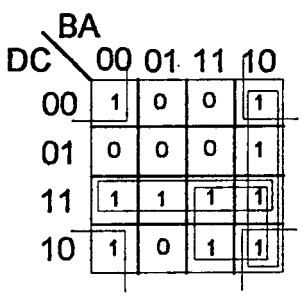


รูปที่ 5.2.4 แผนผังคนอร์ของเอาท์พุท d

เมื่อทำการลดรูปแล้วจะได้สมการบูลลีนดังนี้

d = /B&D + /A&/C&/D + A&B&/C + A&/B&C + /A&B&C

แผนผังคนอร์ของเอาท์พุท e เป็นดังรูปที่ 5.2.5

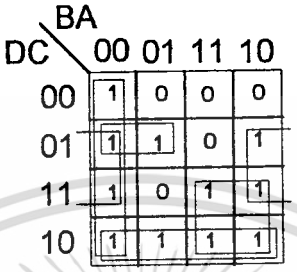


รูปที่ 5.2.5 แผนผังคนอร์ของเอาท์พุท e

เมื่อทำการลดรูปแล้วจะได้สมการบูลลีนดังนี้

e = /A&/C + B&D + C&D + /A&B

แผนผังคนอร์ของเอาท์พุท f เป็นดังรูปที่ 5.2.6

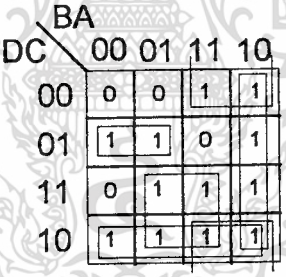


รูปที่ 5.2.6 แผนผังคนอร์ของเอาท์พุท f

เมื่อทำการลดรูปแล้วจะได้สมการบูลลีนดังนี้

f = /A&/B + B&D + /C&D + /A&C + /B&C&/D

แผนผังคนอร์ของเอาท์พุท g เป็นดังรูปที่ 5.2.7



รูปที่ 5.2.7 แผนผังคนอร์ของเอาท์พุท g

เมื่อทำการลดรูปแล้วจะได้สมการบูลลีนดังนี้

g = /A&B + B&/C + A&D + /C&D + /B&C&/D

5.3 สร้างไฟล์ข้อมูลและไฟล์ JEDEC

หลังจากได้สมการบูลลีนของเอาท์พุททั้งหมดแล้ว จะทำสร้างไฟล์ข้อมูลขึ้นมา และนำไฟล์ข้อมูลนั้นไปคอมไพล์ให้เป็นไฟล์ JEDEC ซึ่งรูปแบบของไฟล์ข้อมูลนั้นขึ้นอยู่กับโปรแกรมที่ใช้ ถึงแม้ว่าจะใช้โปรแกรมคอมไพล์ที่ต่างกันแต่ไฟล์ JEDEC ที่ได้จะมีรูปแบบที่คล้ายคลึงกัน เพราะว่าโปรแกรมคอมไพล์ทุกตัวใช้มาตรฐาน JEDEC เดียวกัน ในที่นี้จะใช้โปรแกรมคอมไพล์ GAL - Assembler ของ Ulrich Hack [6]

จากสมการบูลลีนของเอาท์พุททั้งหมดในหัวข้อที่ 5.2 สามารถสร้างไฟล์ข้อมูลที่จะใช้ GAL - Assembler ทำการคอมไพล์ได้ดังนี้

```
01:   Decode binary to 7 segment
02:
03:   *IDENTIFICATION
04:         bin_to_7;
05:
06:   *TYPE
07:         GAL16V8;
08:
09:   *PINS
10:
11:         % input is binary 4 bit %
12:
13:         A = 2,
14:         B = 3,
15:         C = 4,
16:         D = 5,
17:
18:         % output is common anode 7 segment %
19:
20:         /a = 18,
21:         /b = 17,
22:         /c = 16,
23:         /d = 15,
24:         /e = 14,
25:         /f = 13,
26:         /g = 12,
27:
28:   *BOOLEAN - EQUATIONS
29:
30:         a = /A & /C + B & /D + B & C + /A & D + /B & /C & D + A & C & /D;
```

```
31:
32:             b = /A & /C + /C & /D + A & B & /D + A & /B & D + /A & /B & /D;
33:
34:             /c = /A & B & /C & /D + B & C & D + /A & C & D;
35:
36:             d = /B & D + /A & /C & /D + A & B & /C + A & /B & C + /A & B & C;
37:
38:             e = /A & /C + B & D + C & D + /A & B;
39:
40:             f = /A & /B + B & D + /C & D + /A & C + /B & C & /D;
41:
42:             g = /A & B + B & /C + A & D + /C & D + /B & C & /D;
43:
44:     *END
```

หลังจากนั้นนำไฟล์ข้อมูลนี้มาทำการคอมไพล์โดยใช้ GAL - Assembler จะได้ไฟล์ JEDEC ดังนี้

```
01:     <SYX> *
02:
03:     F0 *
04:
05:     N pin 19: not connected *
06:
07:     N pin 18 = '/a' = /function18 *
08:     L0256  1011  1111  1011  1111  1111  1111  1111  1111  *
09:     L0288  1111  0111  1111  1011  1111  1111  1111  1111  *
10:     L0320  1111  0100  0111  1111  1111  1111  1111  1111  *
11:     L0352  1011  1111  1111  0111  1111  1111  1111  1111  *
12:     L0384  111   1011  1011  0111  1111  1111  1111  1111  *
13:     L0416  0111  111   0111  1011  1111  1111  1111  1111  *
14:
```

```
15:  N pin 17 = '/b' = /function17 *
16:  L0512 1011 1111 1011 1111 1111 1111 1111 1111 *
17:  L0544 1111 1111 1011 1011 1111 1111 1111 1111 *
18:  L0576 0111 0111 1111 1011 1111 1111 1111 1111 *
19:  L0608 0111 1011 1111 0111 1111 1111 1111 1111 *
20:  L0640 1011 1011 1111 1011 1111 1111 1111 1111 *
21:
22:  N pin 16 = '/c' = function16 *
23:  L0768 1011 0111 1011 1011 1111 1111 1111 1111 *
24:  L0800 1111 0111 0111 0111 1111 1111 1111 1111 *
25:  L0832 1011 1111 0111 0111 1111 1111 1111 1111 *
26:
27:  N pin 15 = '/d' = /function15 *
28:  L1024 1111 1011 1111 0111 1111 1111 1111 1111 *
29:  L1056 1011 1111 1011 1011 1111 1111 1111 1111 *
30:  L1088 0111 0111 1011 1111 1111 1111 1111 1111 *
31:  L1120 0111 1011 0111 1111 1111 1111 1111 1111 *
32:  L1152 1011 0111 0111 1111 1111 1111 1111 1111 *
33:
34:  N pin 14 = '/e' = /function14 *
35:  L1280 1011 1111 1011 1111 1111 1111 1111 1111 *
36:  L1312 1111 0111 1111 0111 1111 1111 1111 1111 *
37:  L1344 1111 1111 0111 0111 1111 1111 1111 1111 *
38:  L1376 1011 0111 1111 1111 1111 1111 1111 1111 *
39:
40:  N pin 13 = '/f' = /function13 *
41:  L1536 1011 1011 1111 1111 1111 1111 1111 1111 *
42:  L1568 1111 0111 1111 0111 1111 1111 1111 1111 *
43:  L1600 1111 1111 1011 0111 1111 1111 1111 1111 *
44:  L1632 1011 1111 0111 1111 1111 1111 1111 1111 *
```

```
45:    L1664  1111   1011   0111   1011   1111   1111   1111   1111   *
46:
47:    N pin 12 = '/g' = /function12 *
48:    L1792  1011   0111   1111   1111   1111   1111   1111   1111   *
49:    L1824  1111   0111   1011   1111   1111   1111   1111   1111   *
50:    L1856  0111   1111   1111   0111   1111   1111   1111   1111   *
51:    L1888  1111   1111   10111  0111   1111   1111   1111   1111   *
52:    L1920  1111   1011   0111   1011   1111   1111   1111   1111   *
53:
54:    N XOR ( 19..12 ) bits: *
55:    L2048  10010000 *
56:
57:    N user ID: "bin_to_7" *
58:    L2056  01100010011010010110111001011111
59:           01110100011011110101111100110111 *
60:
61:    N AC1 ( 19..12 ) bits: *
62:    L2120  10000000 *
63:
64:    N enable product terms: *
65:    L2128  11111111111111111111111111111111
66:           11111111111111111111111111111111 *
67:
68:    N SYN bit: *
69:    L2192  1 *
70:
71:    N AC0 bit:
72:    L2193  0 *
73:
74:    <ETX>0000
```

เมื่อได้ไฟล์ JEDEC เรียบร้อยแล้ว ก็ทำการโปรแกรม GAL โดยการดาวน์โหลดไฟล์ JEDEC นี้เข้าไปใน GAL และทำการตรวจสอบการโปรแกรมด้วยการ verify แล้วจึงนำ GAL ที่โปรแกรมถูกต้องแล้วนั้นมาต่อลงในวงจรที่ออกแบบไว้ เพื่อใช้งานต่อไป

ถ้าหากเราไม่พอใจ GAL ที่ออกแบบไว้ และต้องการออกแบบใหม่ก็สามารถนำ GAL นั้นมาลบข้อมูลออก แล้วโปรแกรมเข้าไปได้ใหม่

บทที่ 6
บทสรุป

โครงงานนี้เป็นการสร้างเครื่องโปรแกรม GAL ซึ่งทำงานร่วมกับ PC โดยเริ่มจากการศึกษาโครงสร้างและการโปรแกรมอุปกรณ์ชนิดนี้ ตลอดจนพัฒนาฮาร์ดแวร์และซอฟท์แวร์ของเครื่องโปรแกรม ส่วนของฮาร์ดแวร์ได้สร้างเป็นชุดสำเร็จรูปสำหรับการโปรแกรม GAL เบอร์ 16V8 และ 20V8

ส่วนของซอฟท์แวร์ได้เขียนโปรแกรมโดยใช้ภาษาซีควบคุมชิฟท์รีจิสเตอร์ (74HC595) และวงจรทรานซิสเตอร์ผ่านทางพอร์ตขนาน เพื่อไปควบคุมขาต่างๆของ GAL ในโหมดต่างๆให้ทำงานได้อย่างถูกต้อง

เนื่องจากการเป็นการส่งข้อมูลออกทางพอร์ตขนานทำให้เวลาที่ใช้ในการโปรแกรม การตรวจสอบบิตข้อมูล หรือการลบค่อนข้างน้อย อย่างไรก็ตามในการเขียนโปรแกรมจะต้องคำนึงถึงความเร็วในการส่งข้อมูลและ switching time ของฮาร์ดแวร์ให้สัมพันธ์กันด้วย มิฉะนั้นอาจทำให้เกิดความผิดพลาดของข้อมูลได้

การโปรแกรม GAL ในโหมด EDIT จะต้องทำการป้อนแอดเดรสของเส้นเชื่อมต่อโดยการกำหนดลอจิกที่ขา RAG0 ถึง RAG5 ซึ่งสามารถอ้างอิงแอดเดรสได้ 64 แอดเดรส และต้องทำการควบคุมแรงดันที่เข้าขา EDIT ให้มีขนาดคงที่ตลอดเวลาการโปรแกรม และได้ทำการทดสอบเครื่องโปรแกรม โดยการโปรแกรม GAL ให้เป็น 7 Segment decoder ตามขั้นตอนตั้งแต่ ออกแบบวงจรลอจิกจนถึงการโปรแกรมซึ่งให้ผลที่ถูกต้อง

โครงงานชิ้นนี้สามารถใช้เป็นพื้นฐานในการนำไปประยุกต์เพื่อนำไปโปรแกรมอุปกรณ์ PLD ชนิดอื่นต่อไปได้ และเนื่องจากอุปกรณ์ PLD ชนิดนี้ยังไม่เป็นที่นิยมแพร่หลายในเมืองไทยมากนัก จึงทำให้การทำให้ค่อนข้างมีอุปสรรคในการค้นหาข้อมูลและหาซื้ออุปกรณ์ ซึ่งโครงงานชิ้นนี้จะช่วยให้ขอบเขตของการนำ GAL ไปใช้งานได้กว้างขวางยิ่งขึ้น

ภาคผนวก

```
/*////////////////////////////////////////////////////  P16V8A  ////////////////////////////////////////////////////*/
#include <stdio.h>
#include <dos.h>
#include <conio.h>
#include <stdlib.h>
#define DEL 150    /* delay 50 ms */
typedef struct
{
 unsigned  char bb[64][34];   /* this array is Matrix ET & UES*/
 unsigned  char cc[82];       /* this array is ACW*/
}zzz;

zzz  aaa;

zzz  openfile(char *);
char address(int rag);
unsigned char check_sdin(int d,int row);
void prog_ACW(void);

unsigned char  data;
int   a,i,j;
char RAG[7],OUT[17];

int main(int argc,char *argv[])
{
 int row,l,m,n,o,p,q,r,s,t,u,i;
 char rag0,rag1,rag2,rag3,rag4,rag5;
 int clk,d;
 char *filename;

 if(argc > 1)
   filename = argv[1];
 else printf("There are many parameter.");
 openfile(filename);
 for(i=0;i<33;i++)
 {
```

```
    for(j=0;j<64;j++)
      printf("%d",aaa.bb[j][i]);
    printf("\n");
  }
  for(j=0;j<82;j++)
    printf("%d",aaa.cc[j]);
  printf("\n");

  data = 0x10; /* initial data  str='1' */

/* ////////////////////////////////////////          PROGRAM  CYCLE          //////////////////////////////////////// */
  for(row=0;row<=32;row++)     /*add 0...32 -->ET & UES programming*/
  {
    l = row/32; m = row%32;   rag5 = address(l);
    n = m/16;   o = m%16;     rag4 = address(n);
    p = o/8;    q = o%8;      rag3 = address(p);
    r = q/4;    s = q%4;      rag2 = address(r);
    t = s/2;                  rag1 = address(t);
    u = s%2;                  rag0 = address(u);

    RAG[5]=rag5;
    RAG[4]=rag4;
    RAG[3]=rag3;
    RAG[2]=rag2;
    RAG[1]=rag1;
    RAG[0]=rag0;

/* out 595_2 */
    OUT[0] = '0';   /*Qh*/
    OUT[1] = '0';   /*bit Qh-Qf is used for setting voltage in programming*/
    OUT[2] = '1';   /*14.5 V*/
    OUT[3] = '0';
    OUT[4] = '1';   /*bit edit*/
    OUT[5] = '1';   /*Qc */
    OUT[6] = '1';   /*Vcc*/
    OUT[7] = '0';   /*Qa*/
```

```
/* out 595_1 */
   OUT[8] = RAG[5];    /* Qh=RAG[5]*/
   OUT[9] = '1';       /* GND*/
   OUT[10] = RAG[0];   /* Qf=RAG[0]*/
   OUT[11] = '0';      /* bit P/V*/
   OUT[12] = RAG[4];   /* Qd*/
   OUT[13] = RAG[3];   /* Qc*/
   OUT[14] = RAG[2];   /* Qb*/
   OUT[15] = RAG[1];   /* Qa*/

   for(clk=0;clk<16;clk++)
   {
    switch(OUT[clk])    /*check serial data input(D6)*/
    {
         case'1':data|=0x40;break;
         case'0':data&=0x16;break;
    }
    for(a=0;a<15;a++)
         outportb(0x378,data);        /*send data to printer port*/
    for(a=0;a<6;a++)
         outportb(0x378,0x80|data); /*set shift_clk = '1'*/
   }
   data |= 0x10;          /* str = '1'*/
   for(a=0;a<15;a++)
    outportb(0x378,data);        /*shift_clk = '0'*/
   for(a=0;a<6;a++)
    outportb(0x378,0x20|data); /*latch_clk = '1'*/
   for(a=0;a<15;a++)
    outportb(0x378,data);        /*latch_clk = '0' str = '1' */

   for(d=0;d<64;d++)
   {
    data = check_sdin(d,row);
    for(a=0;a<15;a++)
         outportb(0x378,data);
    for(a=0;a<6;a++)
```

```
            outportb(0x378,0x02|data); /* sclk = '1' */
    }

    OUT[11] = '1';    /*set p/v = '1' --> program*/
    for(clk=0;clk<16;clk++)
    {
      switch(OUT[clk])
      {
            case'1':data|=0x40;break;
            case'0':data&=0x10;break;
      }
      for(a=0;a<15;a++)
            outportb(0x378,data);
      for(a=0;a<6;a++)
            outportb(0x378,0x80|data);  /* set shift_clk = '1'*/
    }
    for(a=0;a<15;a++)
      outportb(0x378,data);
    for(a=0;a<6;a++)              /*send latch clk */
      outportb(0x378,0x20|data);
    data &= 0x40;
    for(a=0;a<250;a++)
      outportb(0x378,data);       /*str = '0'*/
    delay(DEL);
  }
  prog_ACW();
  printf("already");
  getch();
  return 0;
}

/*////////////////////////////////////////////  open file  ////////////////////////////////////////////////////////*/
zzz  openfile(char *name)
{
  FILE *fp;
  char ch,c2;
```

```
int  i,j,m,k,row,col;


clrscr();
if((fp=fopen(name,"r")) == NULL)
{
  printf("Can't open file\n");
  exit(1);
} /* this if check file to open that is exist */


row = 0;
do
{
  ch = getc(fp);
  if(ch=='\n') ch = getc(fp);
  if(ch=='L')
  {
   i = 1000;
   m = 0;
   do        /*find address by change char to be decimal*/
   {
          ch = getc(fp);
          switch(ch)
          {
            case '0': k = 0; break;
            case '1': k = i*1; break;
            case '2': k = i*2; break;
            case '3': k = i*3; break;
            case '4': k = i*4; break;
            case '5': k = i*5; break;
            case '6': k = i*6; break;
            case '7': k = i*7; break;
            case '8': k = i*8; break;
            case '9': k = i*9; break;
            case ' ': k = 0; break;
          }
          m = m+k;
```

```
        i = i/10;
    }while(ch != ' ');

    if((m<2048)||(m==2056)) /* Matrix ET and UES*/
    {
        if(m==2056)          /* UES address 32*/
        {
         row = 0;
         col = 32;
         while(ch!='*')
         {
           ch = getc(fp);
           switch(ch)
           {
             case '0': aaa.bb[row][col] = 0; row++; break;
             case '1': aaa.bb[row][col] = 1; row++; break;
             case ' ': break;
             case '\n': break;
           }
         }
        }
        else                    /* Matrix ET address 0 -> 39*/
        {
         if(m!=(row*32))     /* insert 0 in line that don't have*/
         {                   /* address in jedec file       */
           do
           {
             for(col=0;col<32;col++)
                   aaa.bb[row][col] = 0;
             row++;
           }while(m!=(row*32));
         }
         if(m==(row*32))     /* fill data from jedec file into array*/
         {
           row= m/32;
           col = 0;
```

```
            while(ch!='*')
            {
              ch = getc(fp);
              switch(ch)
              {
                    case ' ': break;
                    case '0': aaa.bb[row][col] = 0; col++; break;
                    case '1': aaa.bb[row][col] = 1; col++; break;
                    case '*': break;
              }
            }
            row++;
          }
        }
}
if(m==2048)          /* XOR bit*/
{
      for(j=0;j<=3;j++)
      {
        ch = getc(fp);
        if(ch=='0') aaa.cc[j] = 0;
        else aaa.cc[j] = 1;
      }
      for(j=78;j<=81;j++)
      {
        ch = getc(fp);
        if(ch=='0') aaa.cc[j] = 0;
        else aaa.cc[j] = 1;
      }
}
if(m==2120)          /* AC1 bit*/
{
      for(j=5;j<=8;j++)
      {
        ch = getc(fp);
        if(ch=='0') aaa.cc[j] = 0;
```

```
        else aaa.cc[j] = 1;
      }
  for(j=73;j<=76;j++)
      {
        ch = getc(fp);
        if(ch=='0') aaa.cc[j] = 0;
        else aaa.cc[j] = 1;
      }
}
if(m==2128)                /* PT0..PT63*/
{
      for(j=9;j<=40;j++)    /* this loop is for PT0..PT31*/
      {
        ch = getc(fp);
        if(ch=='0') aaa.cc[j] = 0;
        else aaa.cc[j] = 1;
      }
      for(j=0;j<=6;j++)      /* this loop is for get new line of PT*/
        ch = getc(fp);
      for(j=41;j<=72;j++)   /* this loop is for PT32..PT63*/
      {
        ch = getc(fp);
        if(ch=='0') aaa.cc[j] = 0;
        else aaa.cc[j] = 1;
      }
}
if(m==2192)           /* SYN bit*/
{
        ch = getc(fp);
        if(ch=='0') aaa.cc[77] = 0;
        else aaa.cc[77] = 1;
}
if(m==2193)           /* AC0 bit*/
{
      ch = getc(fp);
      if(ch=='0') aaa.cc[4] = 0;
```

```
            else aaa.cc[4] = 1;
      }


    }
    else while((ch!=EOF)&&(ch!='*'))
            ch = getc(fp);
  }while((ch!=EOF)&&(ch!='\n'));
  fclose(fp);
  return aaa;
}

/* ////////////////////////////////////        check address RAG[]      ////////////////////////////////////////// */
char  address(int rag)
{
  char v;

  switch(rag)
  {
    case 0: v = '0'; break;
    case 1: v = '1'; break;
  }
  return v;
}

/* ////////////////////////////    check data from JEDEC to set D0 for send data(SDIN)    ////////////////// */
unsigned char check_sdin(int d,int row)
{
  switch(aaa.bb[d][row])
  {
    case 1: data |= 0x14; break;
    case 0: data &= 0x50; break;
  }
  return data;
}
```

```
/* ////////////////////////////////////////////////////  PROG_ACW  ////////////////////////////////////////////////////// */
void  prog_ACW(void)
{
 int  clk,d;
 char  OUT[17];    /* ACW address = 60 = 111100*/

/* out 595_2 */
 OUT[0] = '0';     /*Qh*/
 OUT[1] = '0';     /*bit Qh-Qf is used for setting voltage in programming*/
 OUT[2] = '1';     /*14.5V*/
 OUT[3] = '0';
 OUT[4] = '1';     /*bit edit*/
 OUT[5] = '1';     /*Qc*/
 OUT[6] = '1';     /*Vcc*/
 OUT[7] = '0';
/* out 595_1 */
 OUT[8] = '1';     /* Qh=RAG[5]*/
 OUT[9] = '1';     /* Qg*/
 OUT[10] = '0';    /* Qf=RAG[0]*/
 OUT[11] = '0';    /* bit P/V*/
 OUT[12] = '1';    /* Qd=RAG[4]*/
 OUT[13] = '1';    /* Qc=RAG[3]*/
 OUT[14] = '1';    /* Qb=RAG[2]*/
 OUT[15] = '0';    /* Qa=RAG[1]*/

 for(clk=0;clk<16;clk++)
 {
  switch(OUT[clk])              /*check serial data input(D6)*/
  {
   case'1':data|=0x40;break;
   case'0':data&=0x16;break;
  }
  for(a=0;a<15;a++)
   outportb(0x378,data);        /*send data to printer port*/
  for(a=0;a<6;a++)
   outportb(0x378,0x80|data); /*set shift_clk = '1'*/
```

```
}
data |= 0x10;              /* str = '1'*/
for(a=0;a<15;a++)
  outportb(0x378,data); /* shift_clk = '0'*/
for(a=0;a<6;a++)
  outportb(0x378,0x20|data);  /* latch_clk = '1' */
for(a=0;a<15;a++)
  outportb(0x378,data);        /* latch_clk = '0' str = '1' */

for(d=0;d<82;d++)
{
  switch(aaa.cc[d])
  {
    case 1:data|=0x14;break;
    case 0:data&=0x50;break;
  }
  for(a=0;a<15;a++)
    outportb(0x378,data);
  for(a=0;a<6;a++)
    outportb(0x378,0x02|data);  /* sclk = '1' */
}

OUT[11] = '1';   /* set p/v = '1' --> program */
for(clk=0;clk<16;clk++)
{
  switch(OUT[clk])
  {
    case'1':data|=0x40;break;
    case'0':data&=0x10;break;
  }
  for(a=0;a<15;a++)
    outportb(0x378,data);
  for(a=0;a<6;a++)
    outportb(0x378,0x80|data);  /* set shift_clk = '1'*/
}
for(a=0;a<15;a++)
```

```
    outportb(0x378,data);
  for(a=0;a<6;a++)              /* send latch clk */
    outportb(0x378,0x20|data);
  data &= 0x40;
  for(a=0;a<250;a++)
    outportb(0x378,data);       /* str = '0'*/
  delay(DEL);
}
```

```
/*////////////////////////////////////////////  V16V8  ////////////////////////////////////////////////////////////*/
#include <dos.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>

#define ESC   27
#define UP    72
#define DOWN  80
#define LEFT  75
#define RIGHT 77

unsigned char address(int);
unsigned char verify(unsigned char,unsigned char,unsigned char,
                     unsigned char,unsigned char,unsigned char);
unsigned char get_data(unsigned char);
void          line_ET(int);
void          line_XOR(void);
void          line_UES(void);
void          line_AC1(void);
void          line_PT(void);
void          line_SYN(void);
void          line_AC0(void);

struct gal16
{
 char bb[33][64];
 char cc[82];
}zzz;

void main()
{
 unsigned char h,rag5,rag4.rag3,rag2,rag1,rag0;
 int           row,col,k,l,m,n,o,t,u,j,x,y,line;
 char          ch;
```

```
clrscr();
for(row=0;row<=32;row++)
{
  rag5 = row/32;          l = row%32;
  rag4 = l/16;            m = l%16;
  rag3 = m/8;             n = m%8;
  rag2 = n/4;             o = n%4;
  rag1 = o/2;
  rag0 = o%2;

  h = verify(rag5,rag4,rag3,rag2,rag1,rag0);
  delay(2);    /*delay before send SCLK to get data*/
  for(col=0;col<=63;col++)
    zzz.bb[row][col] = get_data(h);
}

h = verify(1,1,1,1,0,0);
delay(2);     /*delay before send SCLK to get data*/
for(k=0;k<=81;k++)
  zzz.cc[k] = get_data(h);

/*////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////                          This is display part.                                    //////
//////////////////////////////////////////////////////////////////////////////////////////////////////////////////// */
clrscr();
window(1,1,80,25);
for(line=0;line<=24;line++)
{
  printf("\n");
  line_ET(line);
}
line--;
ch = getch();
while(ch!=ESC)
{
  x = wherex();
```

```
y = wherey();
switch(ch)
{
  case UP:    y--; line--; break;
  case DOWN:  y++; line++; break;
  case LEFT:  x--; break;
  case RIGHT: x++; break;
  case ESC:   break;
}
if((y>=1)&&(y<=25))
  gotoxy(x,y);
else          /* out of window */
{
  if(y<1)
  {
       gotoxy(1,1);
       insline();
       y = 1;
  }
  else
  {
       printf("\n");
       y = 25;
  }
  if((line>=0)&&(line<=63))
  {
       line_ET(line);
       gotoxy(x,y);
  }
  if((line>=64)&&(line<=69))
  {
       switch(line)
       {
         case 64: line_XOR(); gotoxy(x,y); break;
         case 65: line_UES(); gotoxy(x,y); break;
         case 66: line_AC1(); gotoxy(x,y); break;
```

```
            case 67: line_PT();    gotoxy(x,y); break;
            case 68: line_SYN(); gotoxy(x,y); break;
            case 69: line_AC0(); gotoxy(x,y); break;
          }
          gotoxy(x,y);
    }
    if(line<0)
    {
          while(ch!=DOWN)
          {
            if(ch==UP)
            {
              sound(200);
              delay(100);
              nosound();
            }
            gotoxy(x,1);
            line = 0;
            ch = getch();
          }
    }
    if((line>69)&&(line<75))
    {
          printf("\n");
          gotoxy(x,y);
    }
    if(line==75)
    {
          while(ch!=UP)
          {
            if(ch==DOWN)
            {
              sound(200);
              delay(100);
              nosound();
            }
```

```
            gotoxy(x,25);
            line = 80;
            ch = getch();
           }
     }
    }
    ch = getch();
  } /*////          end display part          /// */
}
/* ////////////////////////////////////////////  end main function  ////////////////////////////////////////////// */

/*                          this function send SER to SDIN of GAL                          */
unsigned char verify(unsigned char a,unsigned char b,unsigned char c,
                     unsigned char d,unsigned char e,unsigned char f)
{
 unsigned char z,data,h1,f1,d1,c1,b1,a1,aa[16],sck;
 int g,loop;

 h1 = 64*a;     /*this bit is RAG5*/
 f1 = 64*f;     /*this bit is RAG0*/
 d1 = 64*b;     /*this bit is RAG4*/
 c1 = 64*c;     /*this bit is RAG3*/
 b1 = 64*d;     /*this bit is RAG2*/
 a1 = 64*e;     /*this bit is RAG1*/
 data = 16;     /* /STR is high */
 aa[0] = 64|data;  /*////////////////////////////////////////////////////////////// */
 aa[1] = 64|data;  /*///  these 3 outputs adjust input edit voltage  /// */
 aa[2] = data;     /*//////////////////////////////////////////////////////////////////*/
 aa[3] = data;     /*this output is VIL*/
 aa[4] = 64|data;  /*this output is EDIT*/
 aa[5] = 64|data;  /*this output is 0 or 1 (don't care)*/
 aa[6] = 64|data;  /*VCC is always equal 1 in all cycles*/
 aa[7] = data;     /*this output is not use*/
 aa[8] = h1|data;  /*this output is RAG5*/
 aa[9] = 64|data;  /*this output set 1 for to be 0 when add to D0*/
                   /*(it connect gnd)*/
```

```
  aa[10] = f1|data;   /*this output is RAG0*/
  aa[11] = data;      /*P/V is always equal 0 in verificant cycle*/
  aa[12] = d1|data;  /*this output is RAG4*/
  aa[13] = c1|data;  /*this output is RAG3*/
  aa[14] = b1|data;  /*this output is RAG2*/
  aa[15] = a1|data;  /*this output is RAG1*/

  for(g=0;g<=15;g++)
   {
     for(loop=0;loop<=14;loop++)
            outportb(0x378,aa[g]);      /*send data to parallel port (sck is low)*/
     for(loop=0;loop<=5;loop++)
            outportb(0x378,128|aa[g]); /*send Shift clk to 595 (sck is high)*/
   }
  for(loop=0;loop<=14;loop++)
   outportb(0x378,aa[15]);
  for(loop=0;loop<=5;loop++)
   outportb(0x378,32|aa[15]); /*force lck to high*/
  for(loop=0;loop<=14;loop++)
   outportb(0x378,aa[15]);    /*force lck to low*/

  /* send /str of GAL to low after send data 60 usec */
  for(loop=0;loop<=5;loop++)
   outportb(0x378,239&aa[15]);
  for(loop=0;loop<=14;loop++)
   outportb(0x378,data);
  return data;    /* /str is high*/
}

/*                          this function get data from SDOUT of GAL                    */
unsigned char get_data(unsigned char z)
{
 unsigned char  x,y;

 outportb(0x378,2|z);
 y = inportb(0x379);
```

```
  delay(1);
  outportb(0x378,z);
  delay(1);
  x = 128&y;          /*get bit BUSY from port 0x379*/
  if(x==0)  x = '0';  /*if bit BUSY is 0 return 0 to array*/
  else x = '1';       /*   "   "     1 return 1 to array*/
  return x;           /*return char to array*/
}

/*                          this function force edit,vcc to low and p/v to high          */
void line_ET(int et_addr)
{
  int i;
  printf("%4d ",et_addr*32);
  for(i=0;i<=31;i++)
    printf("%c",zzz.bb[i][et_addr]);
}

void line_XOR(void)
{
  int i;
  printf("2048 ");
  for(i=32;i<=35;i++)
    printf("%c",zzz.cc[i]);
  for(i=46;i<=49;i++)
    printf("%c",zzz.cc[i]);
}

void line_UES(void)
{
  int i;

  printf("2056 ");
  for(i=0;i<=63;i++)
    printf("%c",zzz.bb[32][i]);
}
```

```
void line_AC1(void)
{
  int i;

  printf("2120 ");
  for(i=37;i<=44;i++)
    printf("%c",zzz.cc[i]);
}

void line_PT(void)
{
  int i;

  printf("2128 ");
  for(i=0;i<=31;i++)
    printf("%c",zzz.cc[i]);
  for(i=50;i<=81;i++)
    printf("%c",zzz.cc[i]);

}

void line_SYN(void)
{
  printf("2192 %c",zzz.cc[45]);
}

void line_AC0(void)
{
  printf("2193 %c",zzz.cc[36]);
}
```

```
/*////////////////////////////////////////////////////////////    E16v8A    ////////////////////////////////////////////////////////////*/
#include <stdio.h>
#include <dos.h>
#include <conio.h>
#include <stdlib.h>

#define DEL 350

unsigned char  data;
int   a,i,j,x,y,z;
char RAG[7],OUT[17];

main()
{
 int clk,d;

 clrscr();
 data = 0x14; /*initial data /str,  SDIN = '1'*/
/* ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////// */
/*                                                    ERASE  CYCLE                                                                          */
/* ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////// */
/* out 595_2 */
OUT[0] = '0';    /*Qh*/
OUT[1] = '0';    /*bit Qh-Qf is used for setting voltage in programming*/
OUT[2] = '1';    /*14.5 V */
OUT[3] = '0';
OUT[4] = '1';    /*bit edit*/
OUT[5] = '1';    /*Qc*/
OUT[6] = '1';    /*Vcc*/
OUT[7] = '0';    /*Qa*/
/* out 595_1 */
OUT[8] = '1';   /* Qh=RAG[5]*/
OUT[9] = '1';   /* GND*/
OUT[10] = '0';  /* Qf=RAG[0]*/
OUT[11] = '0';  /* bit P/V*/
OUT[12] = '1';  /* Qd=RAG[4]*/
```

```
OUT[13] = '1';  /* Qc=RAG[3]*/
OUT[14] = '0';  /* Qb=RAG[2]*/
OUT[15] = '1';  /* Qa=RAG[1]*/

        for(clk=0;clk<16;clk++)      /*send shift_clk 16 pulse*/
          {
                switch(OUT[clk])    /*check serial data input(D6)*/
                {
                case'1':data|=0x40;break;
                case'0':data&=0x16;break;
                }
          for(a=0;a<15;a++)
            outportb(0x378,data);              /* send data to printer port*/
          for(a=0;a<6;a++)
                outportb(0x378,0x80|data); /* set shift_clk = '1'*/
        }
      data|=0x10;
        for(a=0;a<15;a++)
          outportb(0x378,data);
        for(a=0;a<6;a++)
                outportb(0x378,0x20|data); /*latch_clk = '1'*/
        for(a=0;a<15;a++)
          outportb(0x378,data);              /* latch_clk = '0'  str = '1'*/

      /* STR = '0'*/
      data&=0x44;
      for(a=0;a<15;a++)
        outportb(0x378,data);

      /* STR = '1'*/
      data|=0x10;
      /* send sclk 64 pulse */
        for(d=0;d<64;d++)     /*1data -->shift_clk 16 -->latch_clk 1*/
        {
          for(a=0;a<15;a++)
          outportb(0x378,data);/*send data to printer port*/
```

```
        for(a=0;a<6;a++)
        outportb(0x378,0x02|data);/*set sclk = '1'*/
      }
 for(a=0;a<20;a++)
  outportb(0x378,data);/* sclk = '0' */
  delay(10);

    /* P/V = '1'  address 63 */
OUT[10] = '1';  /* Qf=RAG[0]*/
OUT[11] = '1';  /* p/v = '1'*/
OUT[14] = '1';  /* Qb=RAG[2]*/
    for(clk=0;clk<16;clk++)
    {
          switch(OUT[clk]) /*check serial data input(D6)*/
          {
          case'1':data|=0x40;break;
          case'0':data&=0x16;break;
          }
     for(a=0;a<15;a++)
          outportb(0x378,data);
    for(a=0;a<6;a++)
          outportb(0x378,0x80|data);/* set shift_clk = '1'*/
    }
for(a=0;a<15;a++)
  outport(0x378,data);        /* shift clk = '0' */
for(a=0;a<6;a++)
  outportb(0x378,0x20|data);/* latch_clk = '1' */
for(a=0;a<15;a++)
  outportb(0x378,data);       /* latch_clk = '0' */

/* STR = '0' */
data&=0x44;
for(a=0;a<250;a++)
  outport(0x378,data);
delay(DEL);
/* STR = '1' & p/v = '0'&  RAG[] = '0' */
```

```
        data&=0x54;
        OUT[8] = '0';  /* RAG[5] */
        OUT[10] = '0'; /* RAG[0]*/
        OUT[11] = '0'; /* p/v = '0' */
        OUT[12] = '0'; /* RAG[4] */
        OUT[13] = '0'; /* RAG[3] */
        OUT[14] = '0'; /* RAG[2] */
        OUT[15] = '0'; /* RAG[1] */

        for(clk=0;clk<16;clk++)
            {
                switch(OUT[clk]) /*check serial data input(D6)*/
                {
                case'1':data|=0x40;break;
                case'0':data&=0x16;break;
                }
          for(a=0;a<15;a++)
              outportb(0x378,data);
          for(a=0;a<6;a++)
                outportb(0x378,0x80|data);/* set shift_clk = '1'*/
          }
        for(a=0;a<15;a++)
          outport(0x378,data);          /* shift clk = '0' */
        for(a=0;a<6;a++)
          outportb(0x378,0x20|data);/* latch_clk = '1' */
        for(a=0;a<15;a++)
          outportb(0x378,data);        /* latch_clk = '0' */

        /* STR = '1' */
        data|=0x10;
        for(a=0;a<250;a++)
          outport(0x378,data&0x50);
        delay(DEL);
    printf("already");
    getch();
    return 0;
}
```

```
////////////////////////////////////////////////  P20V8 ////////////////////////////////////////////////////////////////////
#include <stdio.h>
#include <dos.h>
#include <conio.h>
#include <stdlib.h>
#define  DEL  50
struct gal
{
 unsigned  char bb[64][42];   // this array is Matrix ET & UES
 unsigned  char cc[82];        // this array is ACW
}zzz;
unsigned char  openfile(void);
char address(int rag);
unsigned char check_RAG5(void);
unsigned char check_sdin(int d,int row);
void prog_ACW(void);

unsigned char  data;
int   a,i,j;
char RAG[6],OUT[16];
main()
{
 unsigned
 int row,l,m,n,o,p,q,r,s,t,u,i;
 char rag0,rag1,rag2,rag3,rag4,rag5;
 int clk,d;
 clrscr();
 openfile();
 data = 0x00;  //initial data
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//                                          PROGRAM  CYCLE                                                            //
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
for(row=0;row<=40;row++)     //add 0...40 -->ET & UES programming
{
```

```
l = row/32; m = row%32;   rag5 = address(l);
n = m/16;   o = m%16;     rag4 = address(n);
p = o/8;    q = o%8;      rag3 = address(p);
r = q/4;    s = q%4;      rag2 = address(r);
t = s/2;                  rag1 = address(t);
u = s%2;                  rag0 = address(u);
        RAG[5]=rag5;
        RAG[4]=rag4;
        RAG[3]=rag3;
        RAG[2]=rag2;
        RAG[1]=rag1;
        RAG[0]=rag0;
/* out 595_2 */
OUT[0] = '0';       //Qh
OUT[1] = '0';       //bit Qh-Qf is used for setting voltage in programming
OUT[2] = '1';
OUT[3] = '1';       //bit edit
OUT[4] = '0';
OUT[5] = RAG[1];//Qc = RAG[1]
OUT[6] = '0';
OUT[7] = '1';       //Qa
/* out 595_1 */
OUT[8] = RAG[4];  // Qh=RAG[4]
OUT[9] = '0';
OUT[10] = RAG[0];  // Qf=RAG[0]
OUT[11] = '0';        // bit P/V
OUT[12] = '0';
OUT[13] = '0';
OUT[14] = RAG[3];  // Qb=RAG[3]
OUT[15] = RAG[2];  // Qa=RAG[2]

        for(clk=0;clk<16;clk++)       //send shift_clk 16 pulse
          {
              switch(OUT[clk])   //check serial data input(D6)
```

```
            {
            case'1':data|=0x40;break;
            case'0':data&=0x0b;break;
            }
            outportb(0x378,data);
            outportb(0x378,0x80|data);   //set shift_clk = '1'
            }
data|=0x28;          //send latch clk
outportb(0x378,data);
// check  RAG[5]
            switch(RAG[5])
            {
            case '1': data|=0x40;break;
            case '0': data&=0xed;break;
            }
 for(d=0;d<64;d++)     //1data -->shift_clk 16 -->latch_clk 1
 {
  check_sdin(d,row);  //check bitD0 from sdin -->PT0 first
  for(a=0;a<16;a++)
  outportb(0x378,data);//send data to printer port
  for(a=0;a<6;a++)
  outportb(0x378,0x04|data);//set sclk = '1'
}
// set /str = '0' and p/v = '1'
data &= 0x42;   //set sdin & sclk = '0'
OUT[11] = '1';//set p/v = '1' --> program
for(clk=0;clk<16;clk++)
{
            switch(OUT[clk]) //check serial data input(D6)
            {
            case'1':data|=0x40;break;
            case'0':data&=0x02;break;
            }
            outportb(0x378,data);
```

```
                outportb(0x378,0x80|data);
            }
            data|=0x20;
            outportb(0x378,data);
            delay(DEL);
}
//program  ACW ( 82 bit )
prog_ACW();
printf("already");
getch();
return 0;
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//                                                  open file                                                             //
unsigned char openfile(char  filename)
{
 FILE *fp;
 char ch;
 int  i,j,m,k,row,col;
 clrscr();
 if((fp=fopen("c:filename","r")) == NULL)
 {
  printf("Can't open file\n");
  exit(1);
 }  //this if check file to open that is exist

 row = 0;
 do
 {
  ch = getc(fp);
  if(ch=='\n') ch = getc(fp);
  if(ch=='L')
  {
   i = 1000;
```

```
m = 0;
do        //find address by change char to be decimal
{
      ch = getc(fp);
      switch(ch)
      {
       case '0': k = 0; break;
       case '1': k = i*1; break;
       case '2': k = i*2; break;
       case '3': k = i*3; break;
       case '4': k = i*4; break;
       case '5': k = i*5; break;
       case '6': k = i*6; break;
       case '7': k = i*7; break;
       case '8': k = i*8; break;
       case '9': k = i*9; break;
       case ' ': k = 0; break;
      }
      m = m+k;
      i = i/10;
}while(ch != ' ');

if((m<2560)||(m==2568)) // Matrix ET and UES
{
      if(m==2568)         // UES address 40
      {
       row = 0;
       col = 40;
       while(ch!='*')
       {
        ch = getc(fp);
        switch(ch)
        {
         case '0': zzz.bb[row][col] = 0; row++; break;
```

```
    case '1': zzz.bb[row][col] = 1; row++; break;
    case ' ': break;
    case '\n': break;
   }
  }
 }
else                // Matrix ET address 0 -> 39
{
 if(m!=(row*40))    // insert 0 in line that don't have
 {                  // address in jedec file
  do
  {
   for(col=0;col<40;col++)
         zzz.bb[row][col] = 0;
   row++;
  }while(m!=(row*40));
 }
 if(m==(row*40))    // fill data from jedec file into array
 {
  row= m/40;
  col = 0;
  while(ch!='*')
  {
   ch = getc(fp);
   switch(ch)
   {
         case ' ': break;
         case '0': zzz.bb[row][col] = 0; col++; break;
         case '1': zzz.bb[row][col] = 1; col++; break;
         case '*': break;
   }
  }
  row++;
 }
```

```
        }
    }
    if(m==2560)          // XOR bit
    {
        for(j=0;j<=3;j++)
        {
         ch = getc(fp);
         if(ch=='0') zzz.cc[j] = 0;
         else zzz.cc[j] = 1;
        }
        for(j=78;j<=81;j++)
        {
         ch = getc(fp);
         if(ch=='0') zzz.cc[j] = 0;
         else zzz.cc[j] = 1;
        }
    }
    if(m==2632)          // AC1 bit
    {
        for(j=5;j<=8;j++)
        {
         ch = getc(fp);
         if(ch=='0') zzz.cc[j] = 0;
         else zzz.cc[j] = 1;
        }
        for(j=73;j<=76;j++)
        {
         ch = getc(fp);
         if(ch=='0') zzz.cc[j] = 0;
         else zzz.cc[j] = 1;
        }
    }
    if(m==2640)          // PT0..PT63
    {
```

```
            for(j=9;j<=40;j++)    // this loop is for PT0..PT31
            {
             ch = getc(fp);
             if(ch=='0') zzz.cc[j] = 0;
             else zzz.cc[j] = 1;
            }
            for(j=0;j<=6;j++)     // this loop is for get new line of PT
             ch = getc(fp);
            for(j=41;j<=72;j++)   // this loop is for PT32..PT63
            {
             ch = getc(fp);
             if(ch=='0') zzz.cc[j] = 0;
             else zzz.cc[j] = 1;
            }
        }
        if(m==2704)          // SYN bit
        {
            ch = getc(fp);
            if(ch=='0') zzz.cc[77] = 0;
            else zzz.cc[77] = 1;
        }
        if(m==2705)          // AC0 bit
        {
            ch = getc(fp);
            if(ch=='0') zzz.cc[4] = 0;
            else zzz.cc[4] = 1;
        }

      }
      else while((ch!=EOF)&&(ch!='*'))
             ch = getc(fp);
    }while((ch!=EOF)&&(ch!='\n'));
    fclose(fp);
    return  zzz;
```

```
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//                                          check address RAG[]                                                       //
char address(int rag)
  {
  char v;
        switch(rag)
        {
        case 0: v = '0'; break;
        case 1: v = '1'; break;
        }
        return v;
  }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//              check data from JEDEC to set D0 for send data(SDIN)*/                                                 //
unsigned char check_sdin(int d,int row)
{
              switch(zzz.bb[d][row]) //send PT0 first
              {
              case 1:data|=0x01;break;
              case 0:data&=0x4a;break;
              }
              return data;   //return data to main()
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//                                          PROG_ACW                                                                  //
void  prog_ACW(unsigned char data)
{
int  clk,d;
char  OUT[17];  // ACW address = 60 = 111100
/* out 595_2 */
OUT[0] = '0';    //Qh
OUT[1] = '0';    //bit Qh-Qf is used for setting voltage in programming
OUT[2] = '1';
```

```
OUT[3] = '1';    //bit edit
OUT[4] = '0';
OUT[5] = '0';          //Qc = RAG1
OUT[6] = '0';
OUT[7] = '1';    //Qa
/* out 595_1 */
OUT[8] = '1';  // Qh=RAG[4]
OUT[9] = '0';
OUT[10] = '0';  // Qf=RAG[0]
OUT[11] = '0';  // bit P/V
OUT[12] = '0';
OUT[13] = '0';
OUT[14] = '1';  // Qb=RAG[3]
OUT[15] = '1';  // Qa=RAG[2]
data = 0x00;
// /str='1' & P/V='0'
        for(clk=0;clk<16;clk++)
          {
          //check serial data input(D6)
                switch(OUT[clk])
                {
                 case'1':data|=0x40;break;
                 case'0':data&=0x0b;break;
                }
                outportb(0x378,data);//send data to printer port
                outportb(0x378,0x80|data);//send data to printer port
          }
      data |=0x28;        //send latch clk
      outportb(0x378,data);
      //set RAG[5] = '1'
                data|=0x02;
      for(d=0;d<82;d++)
       {
       //check sdin(D0)
```

```
            switch(zzz.cc[d])
            {
            case 1:data|=0x01;break;
            case 0:data&=0x4a;break;
            }
        for(a=0;a<16;a++)
         outportb(0x378,data);
        for(a=0;a<6;a++)
         outportb(0x378,0x04|data); //set sclk = '1'
         }
        // set /str = '0' and p/v = '1'
        data = 0x42;//set sclk & sdin = '0'
        OUT[11] = '1';//set p/v = '1' --> program
        for(clk=0;clk<16;clk++)
        {
            switch(OUT[clk])    //check serial data input(D6)
            {
            case'1':data|=0x40;break;
            case'0':data&=0x02;break;
            }
            outportb(0x378,data);
            outportb(0x378,0x80|data);
        }
        data|=0x20; // set latch_clk = '1'
        outportb(0x378,data);
        delay(DEL);
}
```

```
////////////////////////////////////////////////////////// V20V8 ////////////////////////////////////////////////////////
#include <dos.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>

#define ESC   27
#define UP    72
#define DOWN  80
#define LEFT  75
#define RIGHT 77

void        reset595(void);
unsigned char address(int);
unsigned char verify(unsigned char,unsigned char,unsigned char,
                     unsigned char,unsigned char,unsigned char);
unsigned char get_data(unsigned char);
void        finish(void);
void        line_ET(int);
void        line_XOR(void);
void        line_UES(void);
void        line_AC1(void);
void        line_PT(void);
void        line_SYN(void);
void        line_AC0(void);

struct gal20
{
 char bb[41][64];
 char cc[82];
}zzz;
void main()
{
 unsigned char h,rag5,rag4,rag3,rag2,rag1,rag0;
 int         row,col,k,l,m,n,o,t,u,j,x,y,line;
 char         ch;
```

```
  clrscr();
  for(row=0;row<=40;row++)
  {
   rag5 = row/32; l = row%32;
   rag4 = l/16;   m = l%16;
   rag3 = m/8;    n = m%8;
   rag2 = n/4;    o = n%4;
   rag1 = o/2;
   rag0 = o%2;
  h = verify(rag5,rag4,rag3,rag2,rag1,rag0);
   delay(2);    /*delay before send SCLK to get data*/
   for(col=0;col<=63;col++)
    zzz.bb[row][col] = get_data(h);
  }
  h = verify(1,1,1,1,0,0);
  delay(2);     /*delay before send SCLK to get data*/
  for(k=0;k<=81;k++)
   zzz.cc[k] = get_data(h);

 /*                    this function send SER to SDIN of GAL                    */
unsigned char verify(unsigned char a,unsigned char b,unsigned char c,
unsigned char d,unsigned char e,unsigned char f)
{
 unsigned char z,data,d7,c2,h1,f1,b1,a1,aa[16],sck;
 int g,loop;
 d7 = 2*a;       /*this bit is RAG5*/
 c2 = 64*e;      /*this bit is RAG1*/
 h1 = 64*b;      /*this bit is RAG4*/
 f1 = 64*f;      /*this bit is RAG0*/
 b1 = 64*c;      /*this bit is RAG3*/
 a1 = 64*d;      /*this bit is RAG2*/
 data = 9|d7;    /*we OR to add RAG5 into data reg*/

 aa[0] = 64|data;  /*////////////////////////////////////////////////// */
 aa[1] = 64|data;  /*////  these 3 outputs adjust input edit voltage  /// */
 aa[2] = 64|data;  /*////////////////////////////////////////////////// */
```

```
  aa[3] = 64|data;   /*EDIT is always equal 1 in all cycles*/
  aa[4] = data;      /*this output set 0 for c2 to be RAG1*/
  aa[5] = c2|data;   /*this output is RAG1*/
  aa[6] = data;      /*this output is VIL*/
  aa[7] = 64|data;   /*VCC is always equal 1 in all cycles*/
  aa[8] = h1|data;   /*this output is RAG4*/
  aa[9] = data;    /*this output set 1 because in verificant cycle we don't*/
                              /*send data to GAL so we don't care SDIN*/
  aa[10] = f1|data;  /*this output is RAG0*/
  aa[11] = data;     /*P/V is always equal 0 in verificant cycle*/
  aa[12] = data;     /*this output is VIL*/
  aa[13] = data;     /*this output is VIL*/
  aa[14] = b1|data;  /*this output is RAG3*/
  aa[15] = a1|data;  /*this output is RAG2*/
  for(g=0;g<=15;g++)
   {
    for(loop=0;loop<=14;loop++)
          outportb(0x378,aa[g]);     /*send data to parallel port (sck is low)*/
    for(loop=0;loop<=5;loop++)
          outportb(0x378,128|aa[g]);  /*send Shift clk to 595 (sck is high)*/
   }
  for(loop=0;loop<=14;loop++)
   outportb(0x378,aa[15]);
  for(loop=0;loop<=5;loop++)
   outportb(0x378,32|aa[15]); /*force lck to high*/
  for(loop=0;loop<=14;loop++)
   outportb(0x378,aa[15]);    /*force lck to low*/
  for(loop=0;loop<=5;loop++)
   outportb(0x378,247&aa[15]);
  for(loop=0;loop<=14;loop++)
   outportb(0x378,data);
  return data;    /* /str is high*/
}

/*                          this function get data from SDOUT of GAL                          */
unsigned char get_data(unsigned char z)
```

```
 {
  unsigned char  x,y;
  outportb(0x378,4|z);
  y = inportb(0x379);
  delay(1);
  outportb(0x378,z);
  delay(1);
  x = 128&y;          /*get bit BUSY from port 0x379*/
  if(x==0)  x = '0';  /*if bit BUSY is 0 return 0 to array*/
  else x = '1';       /*   "   "    1 return 1 to array*/
  return x;           /*return char to array*/
 }

 /*      this function force edit,vcc to low and p/v to high          */
 void line_ET(int et_addr)
 {
  int i;
  printf("%4d ",et_addr*40);
  for(i=0;i<=39;i++)
   printf("%c",zzz.bb[i][et_addr]);
 }

 void line_XOR(void)
 {
  int i;
  printf("2560 ");
  for(i=0;i<=3;i++)
   printf("%c",zzz.cc[i]);
  for(i=78;i<=81;i++)
   printf("%c",zzz.cc[i]);
 }

 void line_UES(void)
 {
  int i;
  printf("2568 ");
```

```
  for(i=0;i<=63;i++)
    printf("%c",zzz.bb[40][i]);
 }

 void line_AC1(void)
 {
  int i;
  printf("2632 ");
  for(i=5;i<=8;i++)
    printf("%c",zzz.cc[i]);
  for(i=73;i<=76;i++)
    printf("%c",zzz.cc[i]);
 }

 void line_PT(void)
 {
  int i;
  printf("2640 ");
  for(i=9;i<=72;i++)
   printf("%c",zzz.cc[i]);
 }

 void line_SYN(void)
 {
  printf("2704 %c",zzz.cc[77]);
 }

 void line_AC0(void)
 {
  printf("2705 %c",zzz.cc[4]);
 }
```

```
/*////////////////////////////////////////////////////////// E20v8A ////////////////////////////////////////////////////////////*/
#include <stdio.h>
#include <dos.h>
#include <conio.h>
#include <stdlib.h>

#define DEL 350

unsigned char  data;
int   a,i,j,x,y,z;
char RAG[7],OUT[17];

main()
{
 int clk,d;

clrscr();

 data = 0x0b;/*initial data /str,  SDIN = '1'*/
/* ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////// */
/*                                                ERASE  CYCLE                                                                */
/* ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////// */
/* out 595_2 */
OUT[0] = '0';    /*Qh*/
OUT[1] = '0';    /*bit Qh-Qf is used for setting voltage in programming*/
OUT[2] = '1';    /*14.5 V */
OUT[3] = '1';
OUT[4] = '0';    /*bit edit*/
OUT[5] = '1';    /*RAG[1]*/
OUT[6] = '0';    /*Vcc*/
OUT[7] = '1';    /*Qa*/
/* out 595_1 */
OUT[8] = '1';   /* Qh=RAG[4]*/
OUT[9] = '0';   /* GND*/
OUT[10] = '0';  /* Qf=RAG[0]*/
OUT[11] = '0';  /* bit P/V*/
```

```
OUT[12] = '0';
OUT[13] = '0';
OUT[14] = '1';  /* Qb=RAG[3]*/
OUT[15] = '0';  /* Qa=RAG[2]*/

        for(clk=0;clk<16;clk++)     /*send shift_clk 16 pulse*/
          {
                switch(OUT[clk])    /*check serial data input(D6)*/
                {
                case'1':data|=0x40;break;
                case'0':data&=0x0b;break;
                }
           for(a=0;a<15;a++)
            outportb(0x378,data);/*send data to printer port*/
           for(a=0;a<6;a++)
                outportb(0x378,0x80|data);/*set shift_clk = '1'*/
      }
    data|=0x08;
       for(a=0;a<15;a++)
         outportb(0x378,data);/*send data to printer port*/
       for(a=0;a<6;a++)
                outportb(0x378,0x20|data);/*latch_clk = '1'*/
        for(a=0;a<15;a++)
         outportb(0x378,data); /* latch_clk = '0'  str = '1'*/

     /* str = '0'*/
    data&=0x43;
    for(a=0;a<15;a++)
      outportb(0x378,data);

    /* STR = '1'*/
    data|=0x08;
    /* send sclk 64 pulse */
       for(d=0;d<64;d++)     /*1data -->shift_clk 16 -->latch_clk 1*/
       {
        for(a=0;a<15;a++)
```

```
        outportb(0x378,data);/*send data to printer port*/
        for(a=0;a<6;a++)
        outportb(0x378,0x02|data);/*set sclk = '1'*/
      }
  for(a=0;a<20;a++)
   outportb(0x378,data);/* sclk = '0' */
   delay(10);

     /* p/v = '1'  address 63 */
  OUT[10] = '1';  /* Qf=RAG[0]*/
  OUT[11] = '1';  /* p/v = '1'*/
  OUT[15] = '1';  /* Qb=RAG[2]*/
     for(clk=0;clk<16;clk++)
     {
           switch(OUT[clk]) /*check serial data input(D6)*/
           {
           case'1':data|=0x40;break;
           case'0':data&=0x0b;break;
           }
      for(a=0;a<15;a++)
           outportb(0x378,data);
      for(a=0;a<6;a++)
           outportb(0x378,0x80|data);/* set shift_clk = '1'*/
     }
  for(a=0;a<15;a++)
   outport(0x378,data); /* shift clk = '0' */
  for(a=0;a<6;a++)
   outportb(0x378,0x20|data);/* latch_clk = '1' */
  for(a=0;a<15;a++)
   outportb(0x378,data);       /* latch_clk = '0'*/

 /* STR = '0' */
 data&=0x43;
 for(a=0;a<250;a++)
   outport(0x378,data);
 delay(DEL);
```

```
        /* STR = '1' & p/v = '0'&  RAG[] = '0' */
        data&=0x49;
        OUT[10] = '0'; /* RAG[0]*/
        OUT[11] = '0'; /* p/v = '0' */
        OUT[8] = '0';  /* RAG[4] */
        OUT[14] = '0';/* RAG[3] */
        OUT[15] = '0';/* RAG[2] */
        OUT[5] = '0';  /* RAG[1] */

        for(clk=0;clk<16;clk++)
            {
                    switch(OUT[clk]) /*check serial data input(D6)*/
                    {
                    case'1':data|=0x40;break;
                    case'0':data&=0x0b;break;
                    }
              for(a=0;a<15;a++)
                    outportb(0x378,data);
              for(a=0;a<6;a++)
                    outportb(0x378,0x80|data);/* set shift_clk = '1'*/
              }
        for(a=0;a<15;a++)
         outport(0x378,data); /* shift clk = '0' */
        for(a=0;a<6;a++)
         outportb(0x378,0x20|data);/* latch_clk = '1' */
        for(a=0;a<15;a++)
         outportb(0x378,data);       /* latch_clk = '0'*/
        /* STR = '1' */
        data|=0x08;
        for(a=0;a<250;a++)
         outport(0x378,data&0x48);
        delay(DEL);
printf("already");
getch();
return 0;
}
```

MC54/74HC595A

8-Bit Serial-Input/Serial or Parallel-Output Shift Register with Latched 3-State Outputs

High-Performance Silicon-Gate CMOS

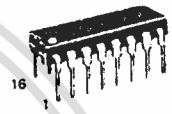
The MC54/74HC595A is identical in pinout to the LS595. The device inputs are compatible with standard CMOS outputs; with pullup resistors, they are compatible with LSTTL outputs.

The HC595A consists of an 8-bit shift register and an 8-bit D-type latch with three-state parallel outputs. The shift register accepts serial data and provides a serial output. The shift register also provides parallel data to the 8-bit latch. The shift register and latch have independent clock inputs. This device also has an asynchronous reset for the shift register.

The HC595A directly interfaces with the Motorola SPI serial data port on CMOS MPUs and MCUs.

- Output Drive Capability: 15 LSTTL Loads
- Outputs Directly Interface to CMOS, NMOS, and TTL
- Operating Voltage Range: 2.0 to 6.0 V
- Low Input Current: 1.0 μA
- High Noise Immunity Characteristic of CMOS Devices
- In Compliance with the Requirements Defined by JEDEC Standard No. 7A
- Chip Complexity: 328 FETs or 82 Equivalent Gates
- Improvements over HC595
  - Improved Propagation Delays
  - 50% Lower Quiescent Power
  - Improved Input Noise and Latchup Immunity

J SUFFIX
CERAMIC
CASE 620-09



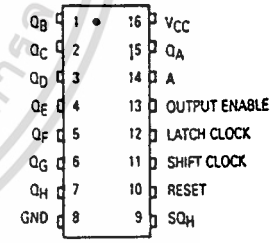
N SUFFIX
PLASTIC
CASE 648-08

D SUFFIX
SOIC
CASE 751B-04

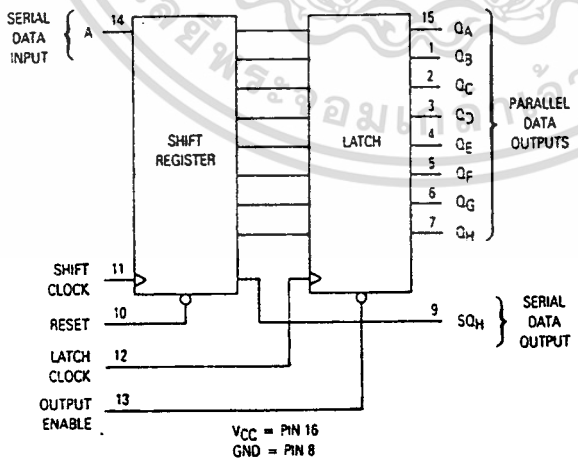
ORDERING INFORMATION

MC74HCXXXAN	Plastic
MC54HCXXXAJ	Ceramic
MC74HCXXXAD	SOIC

PIN ASSIGNMENT

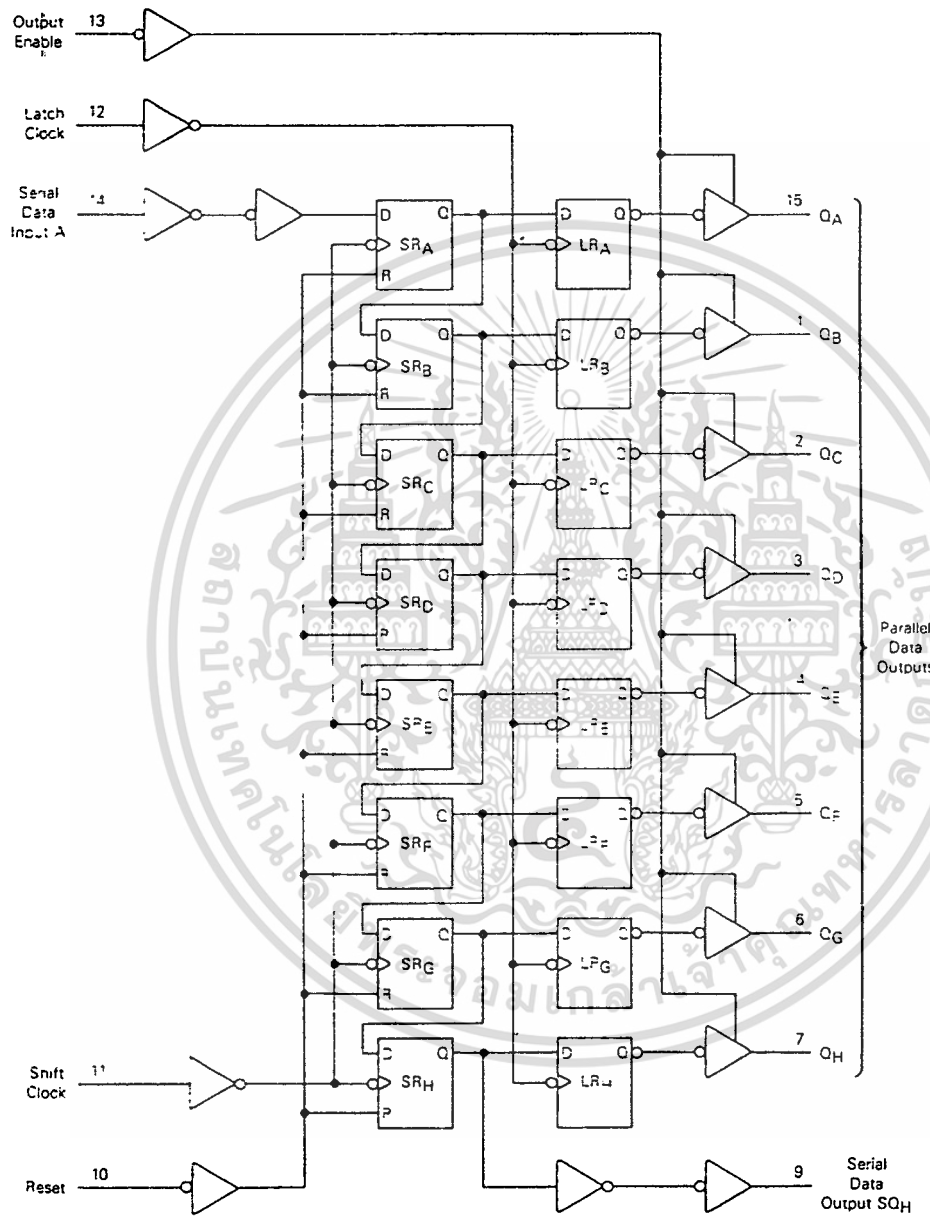


LOGIC DIAGRAM



5

EXPANDED LOGIC DIAGRAM



5

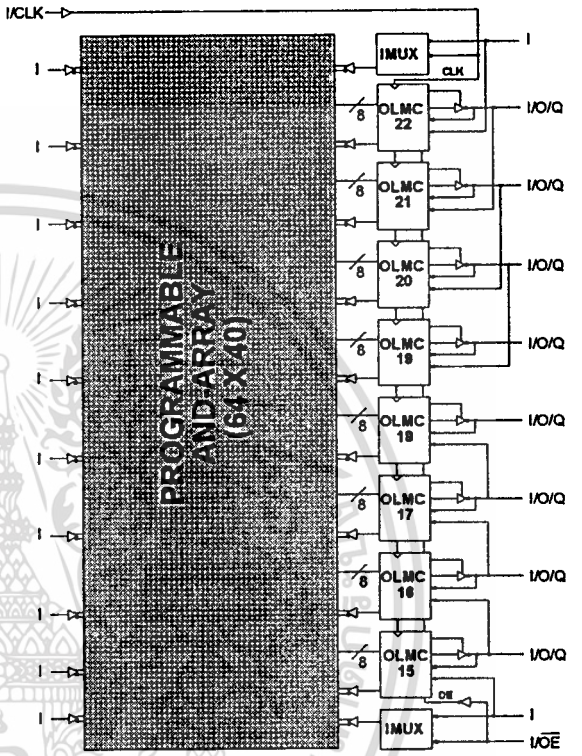
GAL20V8

High Performance E²CMOS PLD
Generic Array Logic™

FEATURES

- **HIGH PERFORMANCE E²CMOS® TECHNOLOGY**
  - 5 ns Maximum Propagation Delay
  - Fmax = 166 MHz
  - 4 ns Maximum from Clock Input to Data Output
  - UltraMOS® Advanced CMOS Technology
- **50% to 75% REDUCTION IN POWER FROM BIPOLAR**
  - 75mA Typ Icc on Low Power Device
  - 45mA Typ Icc on Quarter Power Device
- **ACTIVE PULL-UPS ON ALL PINS**
- **E² CELL TECHNOLOGY**
  - Reconfigurable Logic
  - Reprogrammable Cells
  - 100% Tested/Guaranteed 100% Yields
  - High Speed Electrical Erasure (<100ms)
  - 20 Year Data Retention
- **EIGHT OUTPUT LOGIC MACROCELLS**
  - Maximum Flexibility for Complex Logic Designs
  - Programmable Output Polarity
  - Also Emulates 24-pin PAL® Devices with Full Function/Fuse Map/Parametric Compatibility
- **PRELOAD AND POWER-ON RESET OF ALL REGISTERS**
  - 100% Functional Testability
- **APPLICATIONS INCLUDE:**
  - DMA Control
  - State Machine Control
  - High Speed Graphics Processing
  - Standard Logic Speed Upgrade
- **ELECTRONIC SIGNATURE FOR IDENTIFICATION**

FUNCTIONAL BLOCK DIAGRAM



DESCRIPTION

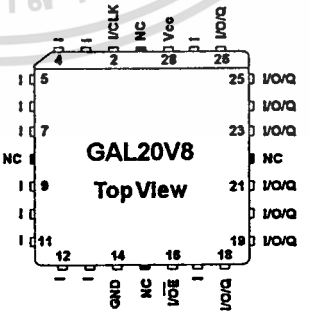
The GAL20V8C, at 5ns maximum propagation delay time, combines a high performance CMOS process with Electrically Erasable (E²) floating gate technology to provide the highest speed performance available in the PLD market. High speed erase times (<100ms) allow the devices to be reprogrammed quickly and efficiently.

The generic architecture provides maximum design flexibility by allowing the Output Logic Macrocell (OLMC) to be configured by the user. An important subset of the many architecture configurations possible with the GAL20V8 are the PAL architectures listed in the table of the macrocell description section. GAL20V8 devices are capable of emulating any of these PAL architectures with full function/fuse map/parametric compatibility.

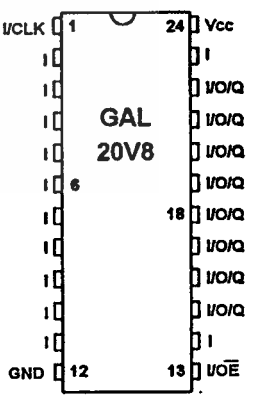
Unique test circuitry and reprogrammable cells allow complete AC, DC, and functional testing during manufacture. As a result, LATTICE is able to guarantee 100% field programmability and functionality of all GAL® products. LATTICE also guarantees 100 erase/rewrite cycles and data retention in excess of 20 years.

PIN CONFIGURATION

PLCC



DIP



LATTICE SEMICONDUCTOR CORP., 5555 Northeast Moore Ct., Hillsboro, Oregon 97124, U.S.A.
Tel. (503) 681-0118; 1-800-FASTGAL; FAX (503) 681-3037

GAL20V8 ORDERING INFORMATION

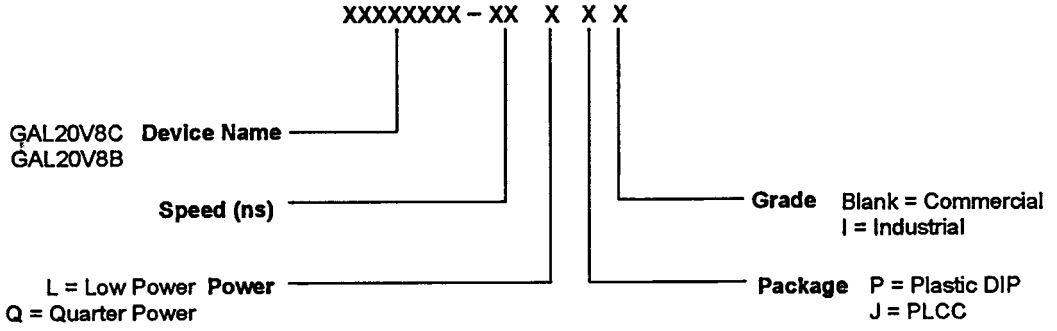
Commercial Grade Specifications

Tpd (ns)	Tsu (ns)	Tco (ns)	Icc (mA)	Ordering #	Package
5	3	4	115	GAL20V8C-5LJ	28-Lead PLCC
7.5	7	5	115	GAL20V8B-7LP	24-Pin Plastic DIP
			115	GAL20V8B-7LJ	28-Lead PLCC
10	10	7	115	GAL20V8B-10LP	24-Pin Plastic DIP
			115	GAL20V8B-10LJ	28-Lead PLCC
15	12	10	55	GAL20V8B-15QP	24-Pin Plastic DIP
			55	GAL20V8B-15QJ	28-Lead PLCC
			90	GAL20V8B-15LP	24-Pin Plastic DIP
			90	GAL20V8B-15LJ	28-Lead PLCC
25	15	12	55	GAL20V8B-25QP	24-Pin Plastic DIP
			55	GAL20V8B-25QJ	28-Lead PLCC
			90	GAL20V8B-25LP	24-Pin Plastic DIP
			90	GAL20V8B-25LJ	28-Lead PLCC

Industrial Grade Specifications

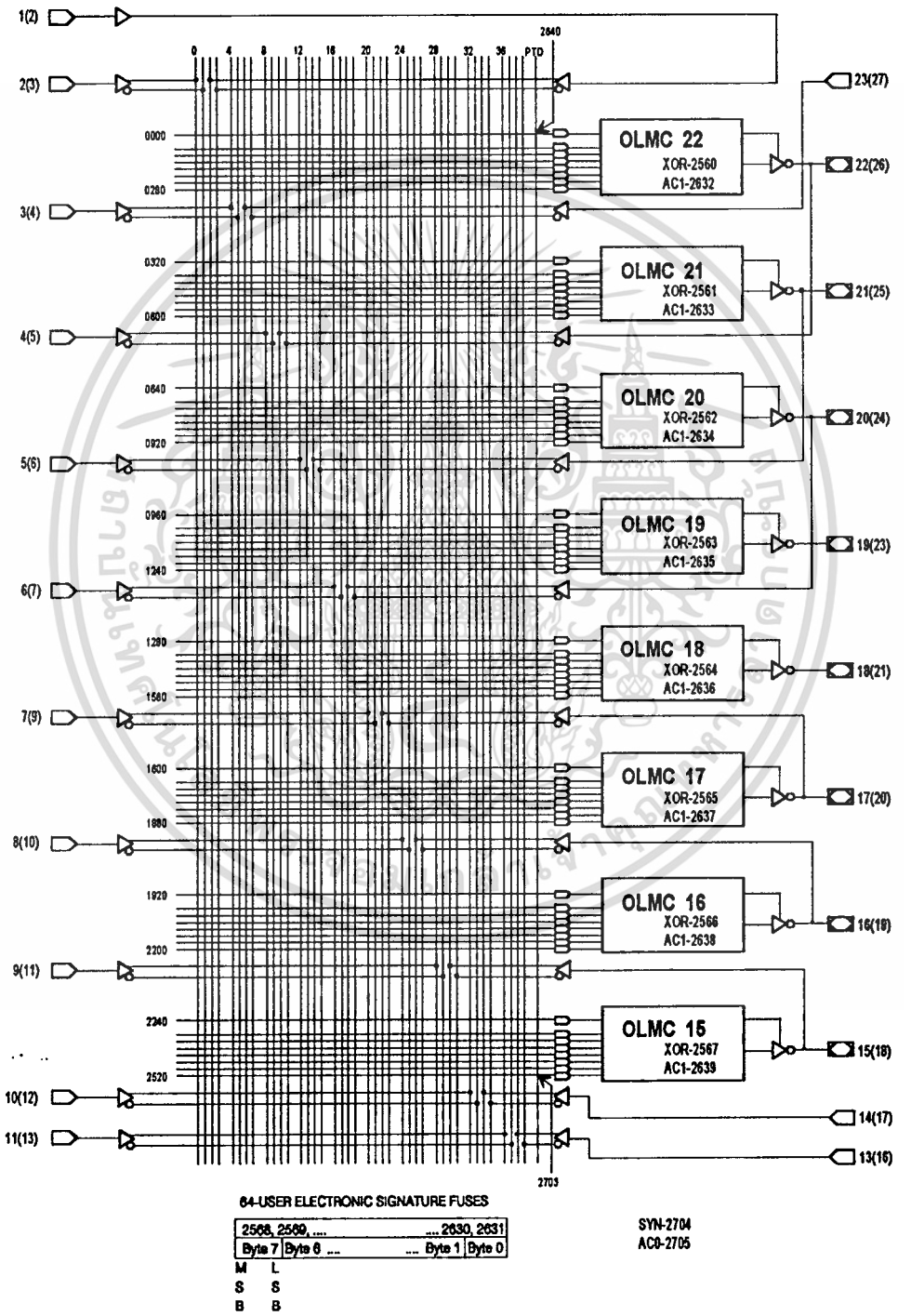
Tpd (ns)	Tsu (ns)	Tco (ns)	Icc (mA)	Ordering #	Package
10	10	7	130	GAL20V8B-10LPI	24-Pin Plastic DIP
			130	GAL20V8B-10LJI	28-Lead PLCC
15	12	10	130	GAL20V8B-15LPI	24-Pin Plastic DIP
			130	GAL20V8B-15LJI	28-Lead PLCC
20	13	11	65	GAL20V8B-20QPI	24-Pin Plastic DIP
			65	GAL20V8B-20QJI	28-Lead PLCC
25	15	12	65	GAL20V8B-25QPI	24-Pin Plastic DIP
			65	GAL20V8B-25QJI	28-Lead PLCC
			130	GAL20V8B-25LPI	24-Pin Plastic DIP
			130	GAL20V8B-25LJI	28-Lead PLCC

PART NUMBER DESCRIPTION



SIMPLE MODE LOGIC DIAGRAM

DIP (PLCC) Package Pinouts



64-USER ELECTRONIC SIGNATURE FUSES

2568, 2569, ...			.... 2630, 2631
Byte 7	Byte 6	.... Byte 1	Byte 0
M S B	L S B		

SYN-2704
AC0-2705

กิตติกรรมประกาศ

โครงงานชิ้นนี้สำเร็จลุล่วงไปได้ด้วยดี ก็ด้วยความช่วยเหลือของ อาจารย์กิติพล ชิตสกุล ที่ให้แนวทางและคำปรึกษาในการทำโครงงาน ตลอดจนอุปกรณ์และเครื่องมือต่าง ๆ และขอขอบคุณสำหรับข้อมูลและคำแนะนำจากอาจารย์วิภา แสงพิสิทธิ์

บรรณานุกรม

1. โอภาส ศิริครรชิตถาวร , " GAL เทคโนโลยีล่าสุดของอุปกรณ์ลอจิกอะเรย์ " , วารสารเซมิคอนดัคเตอร์ , ฉบับที่ 113 , 2535 , หน้า 60 - 70
2. Advanced Micro Devices , " 1996 Databook " , 1994
3. Lattice™ Semiconductor Corporation , " 1996 Databook " , 1996
4. Ronald J Tocci , " Digital System Principle and Application " , Prentiice Hall , 853 p , 1994
5. Motorola , " High - Speed CMOS Logic Data " , 1992
6. U. Hack & Hoffmann , " le manuel des GAL", Publitronic , Pay-Bas , 1994