การออกแบบระบบ 486 EISA โดยใช้ซอฟต์แวร์ช่วยในการออกแบบ

486 EISA SYSTEM DESIGN USING EDA

โดย

นายณัครวัต ทองประเสริฐ

นายพีรเดช บูรณกาญจน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมคอมพิวเตอร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2536

ปริญญานิพนธ์ปีการศึกษา 2536

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนดลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การออกแบบระบบ 486 EISA โดยใช้ซอฟต์แวร์ช่วยในการออกแบบ

ผู้จัดทำ

1. นายณัครวัต ทองประเสริฐ 33100102
2. นายพีรเดช บูรณกาญจน์ 33100263

.....อาจารย์ที่ปรึกษา

(อาจารย์บรรจง ปิยธำรง)

การออกแบบระบบ 486 EISA โดยใช้ซอฟต์แวร์ช่วยในการออกแบบ

486 EISA SYSTEM DESIGN USING EDA

โดย นายณัครวัต ทองประเสริฐ 33100102
นายพีรเดช บูรณกาญจน์ 33100263

อาจารย์ที่ปรึกษา อาจารย์บรรจง ปิยธำรง

บทคัดย่อ

ในอดีตการออกแบบแผ่นวงจรพิมพ์ ทุกขั้นตอนใช้คนเป็นผู้กระทำทั้งสิ้น ซึ่งสามารถกระทำได้เพราะวงจรไม่ซับซ้อนมากนัก แต่ก็ใช้เวลามากและซ้ำซ้อนเนื่องจากไม่สามารถนำส่วนที่เหมือนกันมาใช้ใหม่ได้ ปัจจุบันจึงได้มีการนำคอมพิวเตอร์เข้ามาช่วยในการออกแบบ โดยการใช้ซอฟต์แวร์ประเภท EDA (Electronic Design Automation) เป็นซอฟต์แวร์ประเภท CAD สำหรับการออกแบบทางอิเล็กทรอนิคตั้งแต่ระดับชิปจนถึงแผ่นวงจรพิมพ์ และมีเครื่องมือช่วยในการวิเคราะห์สิ่งที่ออกแบบด้วย ซึ่งช่วยเพิ่มประสิทธิภาพ ความถูกต้องเชื่อถือได้ และช่วยลดเวลาในการออกแบบ

ในการทำปริญญานิพนธ์นี้ได้เลือกระบบ 486 EISA มาใช้เป็นแบบในการทดลองปฏิบัติ เนื่องจากเป็นระบบคอมพิวเตอร์ส่วนบุคคลที่ใช้งานอย่างแพร่หลาย และมีความซับซ้อนพอสมควรเหมาะแก่การศึกษา

ABSTRACT

In the past all procedures of PCB design were made by human. Whereas the circuit was not much complex but it take much of time because of replication. Now we use EDA (Electronic Design Automation) to help us analyze, design and verify electronics design such as IC and PCB. It can reduce time to design and increase productivity and reliability.

In this thesis use 486 EISA System as a case study. Since this system is very popular and has a proper complexity for studying

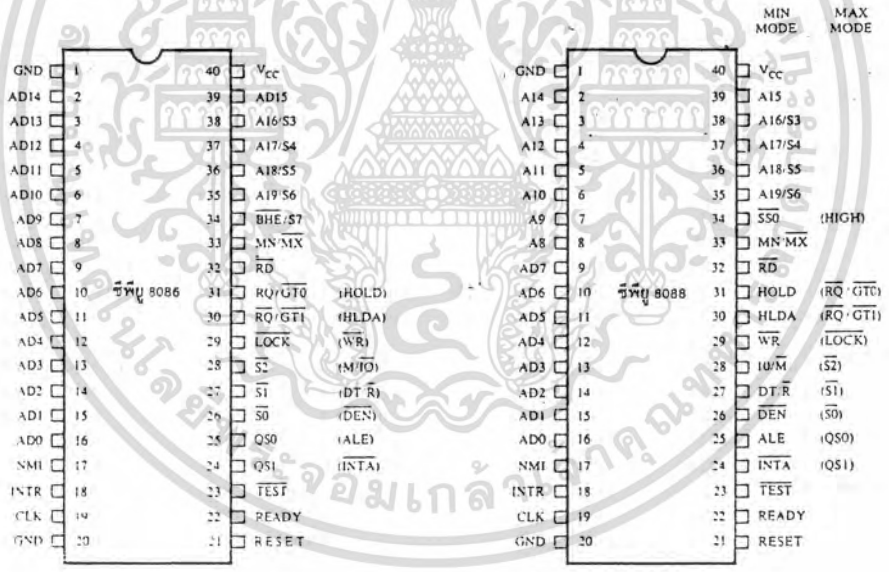
สารบัญ

สารบัญ(ต่อ)

บทที่ 1
สถาปัตยกรรมของไมโครคอมพิวเตอร์พีซี

1.1 โครงสร้างของไมโครคอมพิวเตอร์พีซีเอ็กซ์ที

การออกแบบยึดหลักการของซีพียูขณะนั้น ซึ่งเป็นซีพียู 16 บิต มีการทำโครงสร้างให้มีการขยายต่อไว้อย่างดี ลองพิจารณาโครงสร้างของซีพียู 8088 ที่ บริษัทไอบีเอ็มได้กำหนดไว้ตอนแรกและใช้ในเครื่องไอบีเอ็มพีซีเอ็กซ์ที เมื่อเปรียบเทียบ 8086 ได้จากรูปที่ 1.1



รูปที่ 1.1 โครงสร้างการจัดขาของไอซี 8088/8086

สังเกตว่าขาต่างๆของไอซีทั้งสองตัวนี้มีลักษณะคล้ายกันมาก 8088 มีบัสข้อมูลที่พ่วงมากับแอดเดรสเพียง 8 เส้นในขณะที่ 8086 มี 16 เส้น สัญญาณที่สำคัญมีดังตารางที่ 1.1

ตารางที่ 1.1 หน้าที่ของขาต่างๆ สำหรับซีพียูของระบบ

AD0-AD15 (AD0-AD7)	ขาสัญญาณบัสข้อมูลและแอดเดรส
A16/S3, A17/S4	ตัวแสดงแอดเดรส หรือเซกเมนต์ไอเดนติไฟเออร์
A18/S5	แอดเดรส หรือตัวแสดงสถานะการอีนาเบิลอินเตอร์รัพต์
A19/S6	แอดเดรส/สถานะ
BHE/S7	แสดงไบต์สูง/สถานะ
RD	สัญญาณการอ่าน
READY	สัญญาณต้องการ WAIT
TEST	หยุดรอสำหรับการควบคุมการทดสอบ
INTR	สัญญาณอินเตอร์รัพต์
NMI	นอนมาสเคเบิลอินเตอร์รัพต์
RESET	สัญญาณรีเซต
CLK	สัญญาณนาฬิกาของระบบ
MN/MX	ถ้าเป็น "0" หมายถึง ใช้ระบบแบบโหมด maximum
MN/MX	ถ้าเป็น "1" แสดงว่าเป็นระบบ minimum
S0, S1, S2	แสดงสถานะแมชชีนไซเกิล
RQ/GT0, RQ/GT1	สัญญาณควบคุมลำดับของบัส
QS0, QS1	สัญญาณแสดงคิวคำสั่ง
LOCK	ควบคุมการใช้บัส
M/IO	สัญญาณเลือกหน่วยความจำหรืออินพุตเอาต์พุต
WR	สัญญาณควบคุมการเขียน
ALE	สัญญาณอีนาเบิลการแลตช์แอดเดรส
DT/R	ส่งข้อมูล/รับข้อมูล
DEN	สัญญาณการอีนาเบิลข้อมูล
INTA	สัญญาณตอบอินเตอร์รัพต์
HOLD	สัญญาณการขอ Hold
HLDA	สัญญาณการตอบรับ Hold
สำหรับไอซี 8088มีสัญญาณ SSO ที่ขา 34 มีความหมายถึงการแสดงสถานะ	

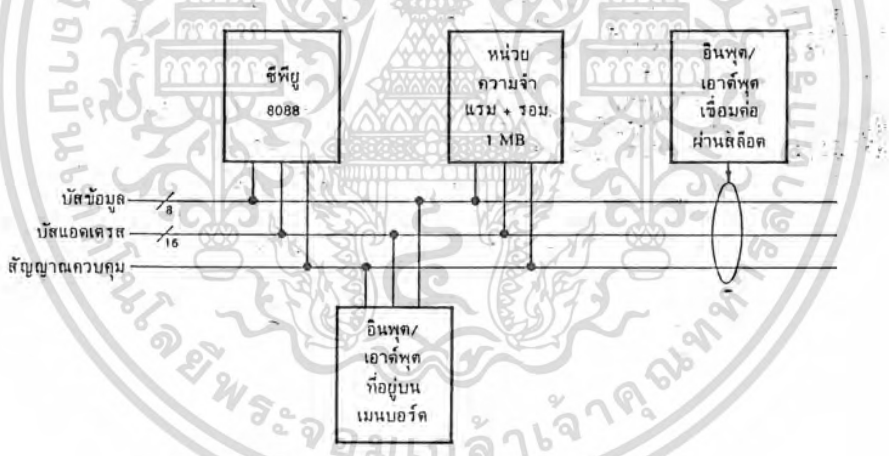
หากพิจารณาจากปรัชญาการออกแบบจะเห็นว่า บริษัทไอบีเอ็มเลือก 8088 ด้วยเหตุผลที่ว่าซีพียูมีบัสข้อมูลเพียง 8 บิตนั้นเหมาะสมกับชิปสนับสนุนขณะนั้น โดยที่เป็นชิปสนับสนุนของกลุ่ม 8080 และ 8085 ซึ่งมีอยู่เป็นจำนวนมากทำให้การใช้ชิปสนับสนุนจึงคล่องตัว และทำให้ต้นทุนในขณะนั้นลดลงได้ โครงสร้างการออกแบบจึงประกอบด้วย

1. การใช้บัสแอดเดรส 20 เส้น จะขยายหน่วยความจำได้ 1 MB

2. วางโครงสร้างอินพุตหรือเอาต์พุตแบบกำหนดพอร์ตตายตัว 8088 ใช้หมายเลขพอร์ต 16 บิต

3. การกำหนดอินเตอร์รัพต์มีการกำหนดลำดับความสำคัญ (priority) ไว้ 8 ระดับการออกแบบฮาร์ดแวร์สนับสนุนอื่นๆจะต้องอาศัยโครงสร้างการอินเตอร์รัพต์

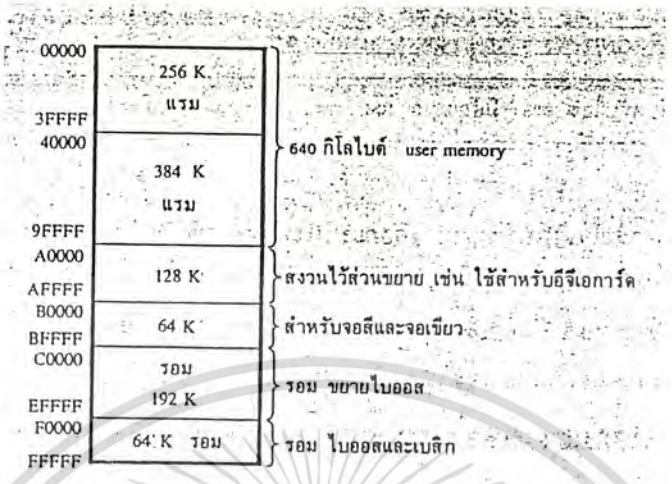
4. สร้างสัญญาณขยายระบบและส่งออกทางสล็อต เพื่อให้ผู้ออกแบบอุปกรณ์เชื่อมต่อทำได้ง่ายด้วยโครงสร้างของซีพียู จึงเขียนเป็นไดอะแกรมให้เข้าใจง่าย ดังรูปที่ 1.2



รูปที่ 1.2 โครงสร้างของไมโครคอมพิวเตอร์ 16 บิต

การใช้บัสแอดเดรส 20 เส้นนี้เองทำให้ซีพียู 8088 มองเห็นหน่วยความจำได้ 1 MB และการจัดหน่วยความจำมีการกำหนดรูปแบบการใช้งานแบบตายตัว โดยแบ่งหน่วยความจำของระบบออกได้ดังรูปที่ 1.3

การใช้หน่วยความจำของเครื่อง 16 บิตในรุ่นแรกที่ใช้ซีพียู 8088 นี้ได้วางโครงสร้างหน่วยความแบบคงที่โดยแรมจะมีได้ในระบบสูงสุด 640 KB ที่เหลือเป็นส่วนที่ได้วางไว้สำหรับให้ระบบใช้งาน เช่น ใช้ในหน่วยแสดงผล ใช้เป็นที่เก็บโปรแกรมเบสิก หรือไบออส ดังนั้นการออกแบบจะต้องวางโคงสร้างของหน่วยความจำให้ตรงกับที่กำหนดไว้ให้ด้วย



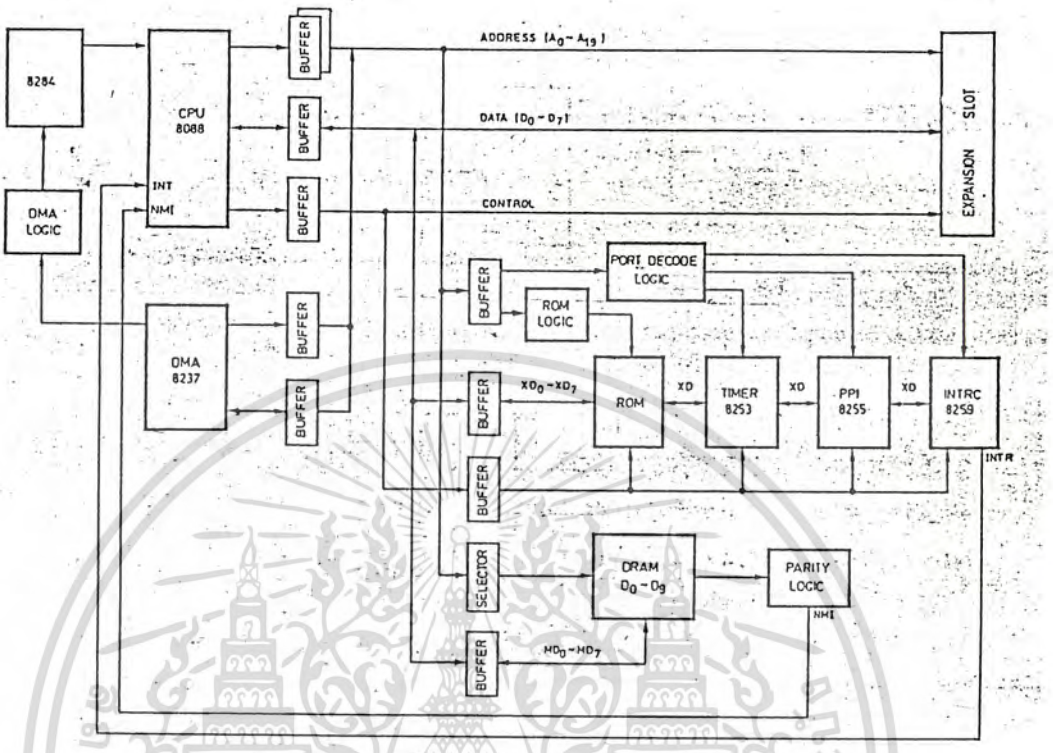
รูปที่ 1.3 การจัดหน่วยความจำ

1.2 โครงสร้างเมนบอร์ดของไมโครคอมพิวเตอร์ 16 บิต

เมนบอร์ดของเครื่องไมโครคอมพิวเตอร์ 16 บิตที่ใช้ซีพียู 8088 ทำงานจะประกอบด้วยชิปหลัก 3 ตัวเป็นการทำงานร่วมกันคือ 8088, 8284 และ 8288 โดยการออกแบบนี้ได้กำหนดสัญญาณนาฬิกาไว้ที่ความถี่ 4.77 MHz โดยเมนบอร์ดในที่นี้จะใช้ไดนามิกแรมขนาด 256 Kx1 มาประกอบเป็น 256 KB บนเมนบอร์ดยังมีรอมที่ทำหน้าที่เป็นไบออสของระบบ และเป็นรอมภาษาเบสิก หรือจะใช้รอมเป็นอะไรก็ได้ นอกจากนี้บนเมนบอร์ดยังมีอุปกรณ์เชื่อมต่อกับอินพุตเอาต์พุตหลายตัวเพื่อทำหน้าที่พิเศษอื่นๆ เช่น ควบคุมและจัดลำดับการเรียกใช้อินเตอร์รัพต์ ตัวกำหนดเวลาหรือไทเมอร์ (timer) ตัวเชื่อมโยงกับพอร์ตแบบขนาน โครงสร้างการจัดเมนบอร์ดเป็นดังรูปที่ 1.4

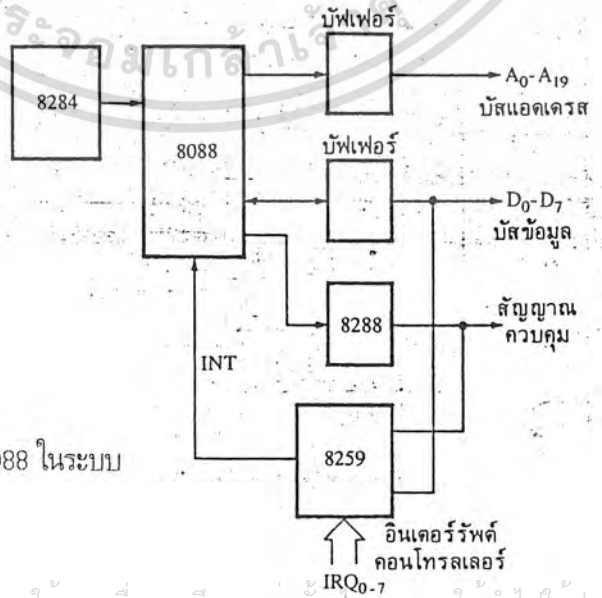
โครงสร้างของเมนบอร์ดอาศัย 8284 เป็นตัวกำเนิดสัญญาณนาฬิกา และควบคุมการรีเซตซีพียู 8288 ทำหน้าที่ควบคุมบัส และสร้างสัญญาณบอกจังหวะการทำงานของซีพียู เช่นการเขียนการอ่านหน่วยความจำ หรือการติดต่อกับอินพุตเอาต์พุต

สิ่งที่สำคัญบนเมนบอร์ดมีไอซี 8259 ซึ่งทำหน้าที่ควบคุมการอินเตอร์รัพต์โดยบนเมนบอร์ดของรุ่นนี้ใช้ 8259 เพียงตัวเดียวจะทำการจัดลำดับการอินเตอร์รัพต์ได้ 8 ระดับ การจัดลำดับนี้เป็นประโยชน์ต่อระบบมาก เพราะการเข้าหาระบบทางซอฟต์แวร์ของอินพุตเอาต์พุตหลายอย่างของคอมพิวเตอร์นี้จะผ่านทางอินเตอร์รัพต์ หากสังเกตบนเมนบอร์ดจะพบไอซีขนาดใหญ่อีกหลายตัว เช่น 8237 ทำหน้าที่ควบคุมการดีเอ็มเอ 8253 ทำหน้าที่เป็นไทเมอร์ กำหนดช่วงเวลาการรีเฟรชหน่วยความจำ กำหนดเวลาของระบบ เป็นตัวสร้างเสียงออกทางลำโพง



รูปที่ 1.4 บล็อกไดอะแกรมของวงจรเมนบอร์ด

หากจะพิจารณาเฉพาะซีพียูกับอุปกรณ์ประกอบนั้น จะเห็นว่าซีพียู 8088 ทำงานร่วมกับชิปสนับสนุน 8284 และ 8288 โดยรับการอินเตอร์รัพต์ผ่านทาง 8259 โดยมีการกำหนดลำดับการอินเตอร์รัพต์ไว้ 8 ระดับ โครงสร้างการต่อเป็นดังรูปที่ 1.5



รูปที่ 1.5 ซีพียู 8088 ในระบบ

1.2.1 การกำหนดสัญญาณควบคุมในซีพียู 8088

การควบคุมระบบของซีพียู 8088 อาศัยการกำหนดสัญญาณใน IO/$\overline{M}$ DT/$\overline{R}$ และ $\overline{SSO}$ ประกอบรวมกันดังตารางที่ 1.2

IO/$\overline{M}$	DT/$\overline{R}$	$\overline{SSO}$	ลักษณะสัญญาณ
0	0	0	การแอกเซส CS
0	0	1	อ่านหน่วยความจำ
0	1	0	เขียนหน่วยความจำ
0	1	1	NOP
1	0	0	ตอบสนองอินเตอร์รัพต์
1	0	1	อ่าน I/O
1	1	0	เขียน I/O
1	1	1	Halt

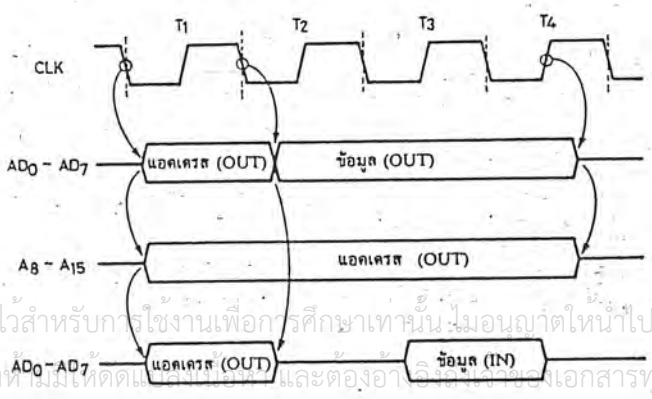
ตารางที่ 1.2 การกำหนดสัญญาณควบคุมในซีพียู 5088

การแยกสัญญาณสถานะทั้งหมด คือแยกสัญญาณมาใช้ควบคุมระบบจะกระทำโดยชิปสนับสนุน ซึ่งบริษทอินเทลได้ออกแบบไว้ก่อนแล้ว

1.2.2 สัญญาณกำหนดบัสสำหรับหน่วยความจำและ I/O

ซีพียู 8088 รับสัญญาณนาฬิกาเข้าทางขา 19 ของไอซี ลักษณะของสัญญาณทางบัสแอดเดรสและข้อมูล มัลติเพลกซ์กันอยู่ การกำหนดแอดเดรสและข้อมูลจึงต้องมีจังหวะที่พอเหมาะ โครงสร้างไดอะแกรมเวลาของ 8088 ในส่วนของบัสทั้งสองเป็นดังรูปที่ 1.6

รูปที่ 1.6 ตารางเวลาการทำงานของซีพียู 8088



การแบ่งเวลานี้เองจึงทำให้การทำงานต้องมีการแยกสัญญาณแอดเดรสและข้อมูลออกจากกัน เพื่อให้เป็นระบบบัสที่สมบูรณ์ทั้งแอดเดรสและข้อมูลที่จะนำไปเชื่อมต่อกับระบบภายนอกได้ง่ายขึ้น

1.2.3 ไอซีกำเนิดสัญญาณนาฬิกา 8284

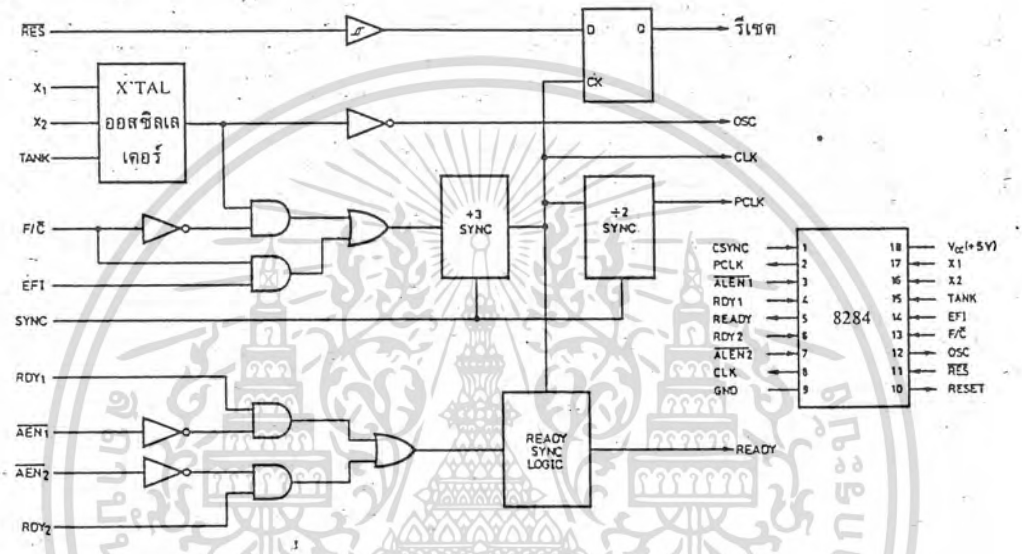
ไอซี 8284 นี้อินเทลได้ออกแบบมาให้ใช้เฉพาะเจาะจงกับ 8088/8086 ลักษณะสัญญาณนาฬิกาที่สร้างโดย 8284 มาจากคริสตอล โดยปกติสัญญาณจากคริสตอลจะได้รับการกระตุ้นภายในชิปให้เกิดการออสซิลเลต และมีการหารสัญญาณด้วย 3 เพื่อเป็นสัญญาณนาฬิกาให้กับระบบไมโครโปรเซสเซอร์ จะเห็นว่าสัญญาณนาฬิกาของระบบไมโครโปรเซสเซอร์มีความถี่เพียง 1 ใน 3 ของคริสตอล ดังนั้นถ้าให้ความถี่ของระบบเป็น 4.77 MHz คริสตอลที่ใช้จะมีค่าเป็น 4.77x3 = 13.31 MHz โครงสร้างของชิปและการจัดขาแสดงได้ดังรูปที่ 1.7 ความหมายของขาต่างๆเป็นดังตารางที่ 1.3

ตารางที่ 1.3 หน้าที่ของแต่ละขาของไอซี 8284

RESET	สัญญาณส่งไรีเซตซีพียู 8086/8088
RES	สัญญาณอินพุตรีเซต
RDY 1, RDY 2	สัญญาณอินพุตสำหรับ wait state
AEN 1, AEN 2	แอดเดรสอีนาเบิล สำหรับ RDY 1 แและ RDY 2
READY	สัญญาณควบคุมส่งต่อไปยัง 8086/8088
X1, X2	ต่อคริสตอล
TANK	ส่วนต่อสำหรับปรับค่าคริสตอล
EFI	สัญญาณนาฬิกาอินพุตทางหนึ่ง
F/C	เลือกสัญญาณนาฬิกาอินพุตมาจากทางใด
CLK	สัญญาณนาฬิกาที่ส่งไปยัง 8086
PCLK	สัญญาณนาฬิการะดับ TTL สำหรับให้อุปกรณ์ประกอบอื่น
OSC	สัญญาณนาฬิกาความถี่เท่าคริสตอล
CSYNC	สัญญาณซิงโครนัส
Vcc, GND	สัญญาณนาฬิกา และกราวนด์

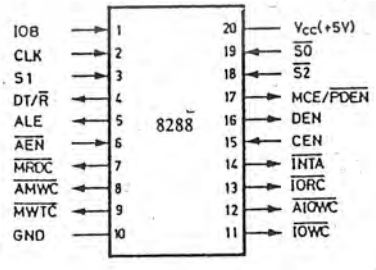
1.2.4 ไอซีควบคุมบัส 8288

สืบเนื่องจากการรวมสัญญาณการแสดงสถานะต่างๆไว้ด้วยกัน อินเทลจึงจำเป็นต้องออกแบบไอซีควบคุมบัสขึ้นมาเฉพาะเพื่อประกอบกับซีพียู 8088/8086 เพื่อให้เกิดสัญญาณควบคุมวงรอบการทำงานที่



รูปที่ 1.7 โครงสร้างภายในและการจัดขาของไอซี 8284

สมบูรณ์และง่ายต่อการใช้งาน สัญญาณควบคุมที่ออกจาก 8088/8086 ในส่วนของ S0, S1, S2 จะได้รับการแปลค่าออกไป โครงสร้างการจัดขาของไอซีและหน้าที่ของแต่ละขาแสดงดังรูปที่ 1.8 และตารางที่ 1.4



รูปที่ 1.8 โครงสร้างการจัดขาไอซี 8288

ตารางที่ 1.4 หน้าที่ของแต่ละขาของไอซี 8288

S0, S1, S2	เป็นการรับสัญญาณแสดงสถานะจากซีพียู
CLK	รับสัญญาณนาฬิกา
AEN	ควบคุมลำดับบัส
CEN	อีนาเบิลคำสั่ง
IOB	โหมดควบคุม
MRDC	สโตรบการอ่านหน่วยความจำ
MWTC	สโตรบการเขียนหน่วยความจำ
AMWC	สโตรบการเขียนหน่วยความมจำแบบ early
IORC	สโตรบการอ่านอินพุตเอาต์พุต
IOWC	สโตรบการเขียนอินพุตเอาต์พุต
AIOWC	สโตรบการเขียนอินพุตเอาต์พุตแบบ early
INTA	การตอบสนองอินเตอร์รัพต์
MCE/PDEN	cascade/peripheral data enable
ALE	อีนาเบิลการแลตช์แอดเดรส
DT/R	ควบคุมทิศทางข้อมูล
DEN	อีนาเบิล บัฟเฟอร์ข้อมูล
Vcc GND	ไฟบวก กราวนด์

1.2.5 ควบคุมการจัดลำดับการอินเตอร์รัพต์

การใช้อินเตอร์รัพต์เป็นเรื่องที่สำคัญมากสำหรับวงจรไมโครคอมพิวเตอร์เพราะเป็นการเพิ่มประสิทธิภาพการทำงานให้กับระบบได้มาก การจัดวางงานในระบบซอฟต์แวร์จะสัมพันธ์กับการจัดวางลำดับของการอินเตอร์รัพต์ การจัดลำดับของการอินเตอร์รัพต์ตามหน้าที่การงานเป็นดังนี้

IRQ0 มีลำดับสูงสุด ใช้สำหรับจัดการเกี่ยวกับนาฬิกา

IRQ1 ใช้กับคีย์บอร์ด

IRQ2 สำรองไว้

IRQ3 ใช้กับพอร์ตสื่อสาร 1

IRQ4 ใช้กับพอร์ตสื่อสาร 2

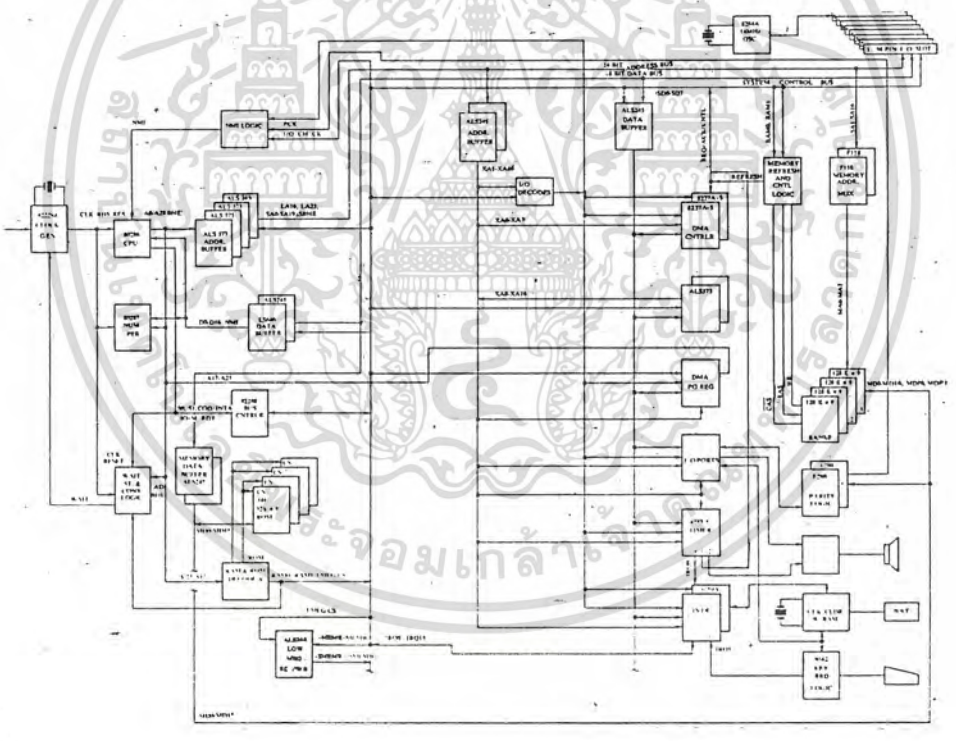
IRQ 5 ใช้กับฮาร์ดดิสก์

IRQ6 ใช้กับฟลอปปี้ดิสก์

IRQ7 ใช้กับเครื่องพิมพ์

1.3 เครื่องไอบีเอ็มเอที

เครื่องไอบีเอ็มเอทีใช้ซีพียู 80286 ทำงานที่ความถี่ของสัญญาณนาฬิกา 6 MHz มีโครงสร้างบัสเป็น 16 บิตเต็ม และมีส่วนขยายของระบบทางฮาร์ดแวร์ที่เพิ่มเติมจากไอบีเอ็มเอ็กซ์ทีหลายส่วน โครงสร้างแสดงดังรูปที่ 1.9



รูปที่ 1.9 บล็อกไดอะแกรมของเมนบอร์ดเอที

ระบบไมโครคอมพิวเตอร์พีซีเอทีนี้ เป็นการปรับปรุงเทคโนโลยีจากเดิม โดยยังคงความเข้ากันได้กับระบบเอ็กซ์ทีเดิม โครงสร้างของระบบดังรูปที่ 1.9 ประกอบด้วย

ไมโครโปรเซสเซอร์เบอร์ 80286 ซึ่งเป็นไมโครโปรเซสเซอร์ที่มีระบบบัสข้อมูล 16 เส้น และแอดเดรส 24 เส้น การอ้างแอดเดรสทำได้ถึง 16 MB นอกจากนี้ยังทำงานร่วมกับโปรเซสเซอร์คณิตศาสตร์เบอร์ 80287 ได้อีกด้วย

วงจรสนับสนุน ประกอบด้วย การจัดการดีเอ็มเอ ได้เพิ่มขยายช่องการทำดีเอ็มเอไปเป็น 7 ช่อง เพิ่มการจัดการอินเตอร์รัพต์จากเดิม 8 ระดับไปเป็น 16 ระดับ มีการจัดการระบบสัญญาณนาฬิกาที่โปรแกรมได้

ต่อเชื่อมรอม สำหรับเก็บโปรแกรมแบบถาวร เช่น ไบออสขนาด 64 KB และเพิ่มขยายต่อเป็น 128 KB

ต่อหน่วยความจำแรม ระบบพีซีเอทีนี้ต่อกับหน่วยความจำแรมบนเมนบอร์ด 512 KB หรือต่อขยายเป็น 1 MB หรือมากกว่านั้น

วงจรควบคุมลำโพง สำหรับเปิด/ปิดเสียง ต่อกับลำโพงขนาดเล็ก

ซีมอสแรม เก็บข้อมูลระบบ ในพีซีเอ็กซ์ทีใช้สวิตช์สำหรับติดตั้งระบบ แต่ในพีซีเอทีใช้หน่วยความจำซีมอสขนาด 64 ไบต์เป็นตัวเก็บข้อมูลโดยมีแบตเตอรี่สำรองข้อมูลไว้

มีชิปนาฬิกาทำงานตลอดเวลา นาฬิกานี้จะทำงานแม้ปิดเครื่อง เวลาของระบบจึงไม่ต้องตั้งใหม่

มีวงจรเชื่อมกับแบตเตอรี่สำรอง ใช้สำหรับจ่ายให้กับซีมอส และนาฬิกา

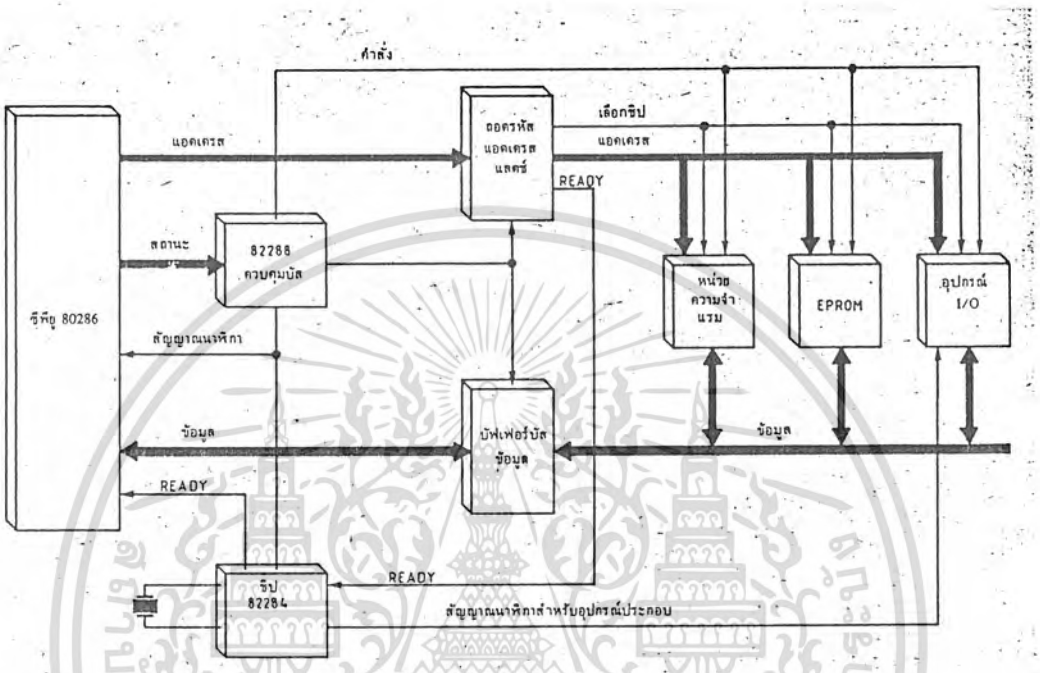
ต่อสล็อตได้ 8 สล็อต ซึ่งเป็นการเพิ่มสล็อตเป็นแบบเอทีเอง โดยเพิ่มซ็อกเกตขนาด 36 ขา ต่อจากของเดิม 62 ขา ในระบบเอ็กซ์ทีออกมาอีก 6 สล็อต

โดยปกติ 80286 จะทำงานร่วมกับชิปกำเนิดสัญญาณนาฬิกา 82284 และชิปควบคุมบัส 82288 ประกอบกันเข้าเป็นระบบเพื่อสร้างบัสเชื่อมโยงกับอุปกรณ์หน่วยความจำหรืออุปกรณ์จำพวกเพอริเฟอรัล แสดงได้ดังรูปที่ 1.10 ซึ่งเป็นโครงสร้างที่ระบบพีซีเอทีจะต้องใช้ในการเชื่อมต่อกับสิ่งต่างๆ และโครงสร้างระบบบัสดังรูป 1.11

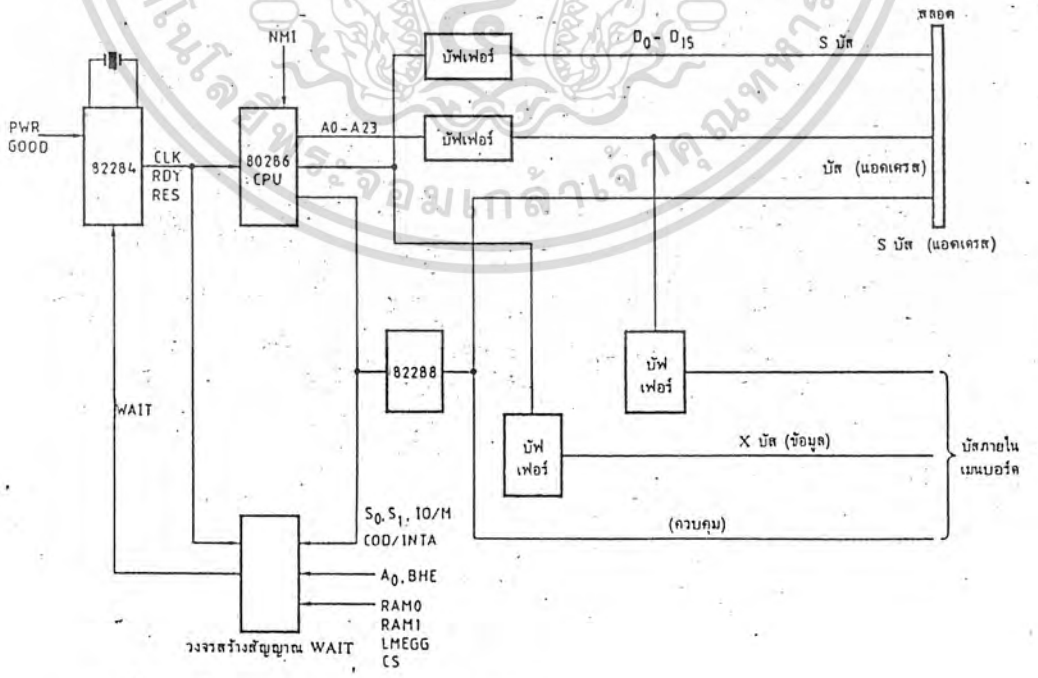
1.3.1 การจัดโครงสร้างแอดเดรส

เนื่องจาก 80286 มีสายแอดเดรสเพิ่มขึ้นมาเป็น 24 เส้น ทำให้ต่อหน่วยความจำจริงได้ถึง 16 MB ไอบีเอ็มเอทีได้จัดโครงสร้างหน่วยความจำของระบบไว้ดังตารางรูปที่ 1.12

จะเห็นว่ามีแอดเดรสบางส่วนที่ซ้ำกัน ทั้งนี้เพราะเมื่อระบบที่ออกแบบมาใช้หน่วยความจำเต็มที่ เช่น โอเอส2 จะมีการย้ายไบออสไปไว้ในแอดเดรสสุดท้ายในส่วน FE0000-FFFFFF ได้



รูปที่ 1.10 ระบบไมโครคอมพิวเตอร์ที่มีซีพียู 80286



รูปที่ 1.11 โครงสร้างของระบบบัสของพีซีเอที

แอดเดรส	หน่วยความจำ	ฟังก์ชัน
000000-07FFFF	หน่วยความจำระบบ 512 กิโลไบต์	หน่วยความจำระบบ
080000-09FFFF	128 กิโลไบต์	หน่วยความจำขยายเพิ่มอีก จนครบ 640 กิโลไบต์
0A0000-0BFFFF	128 กิโลไบต์ วิดีโอ	สงวนไว้สำหรับการแสดงผล
0C0000-0DFFFF	128 กิโลไบต์ หน่วยความจำ ROM สำหรับขยายระบบ I/O	สงวนไว้สำหรับรอมบนบอร์ดอะแดปเตอร์ I/O
0E0000-0EFFFF	สงวนไว้ 64 กิโลไบต์ สำหรับระบบ	ที่ซ้ำกับแอดเดรส FE0000
0F0000-0FFFFF	สงวนไว้ 64 กิโลไบต์ ไบออสรอม	ที่ซ้ำกับแอดเดรส FF0000
100000-FDFFFF	15 เมกะไบต์	ส่วนขยายหน่วยความจำ
FE0000-FFFFFF	128 กิโลไบต์	ซ้ำกับแอดเดรส FE0000

รูปที่ 1.12 ตารางโครงสร้างหน่วยความจำของระบบ

ในไอบีเอ็มเอ็กซ์ที การต่อหน่วยความจำต่อเข้ากับบัสแบบปกติ และใช้สัญญาณรีเฟรชที่มาจากตัวกำเนิดสัญญาณนาฬิกา 8253 ที่กระตุ้นต่อไปยัง 8237 เพื่อขอทำดีเอ็มเอในช่อง 0 เพื่อใช้ในการรีเฟรช แต่สำหรับไอบีเอ็มเอทีเปลี่ยนเป็นการสร้างสัญญาณรีเฟรชโดยตรง โดยนำสัญญาณนาฬิกากำหนดการรีเฟรชมาจากทางช่อง 1 ของ 8254 ซึ่งกำหนดช่วงเวลาประมาณ 15 ไมโครวินาที

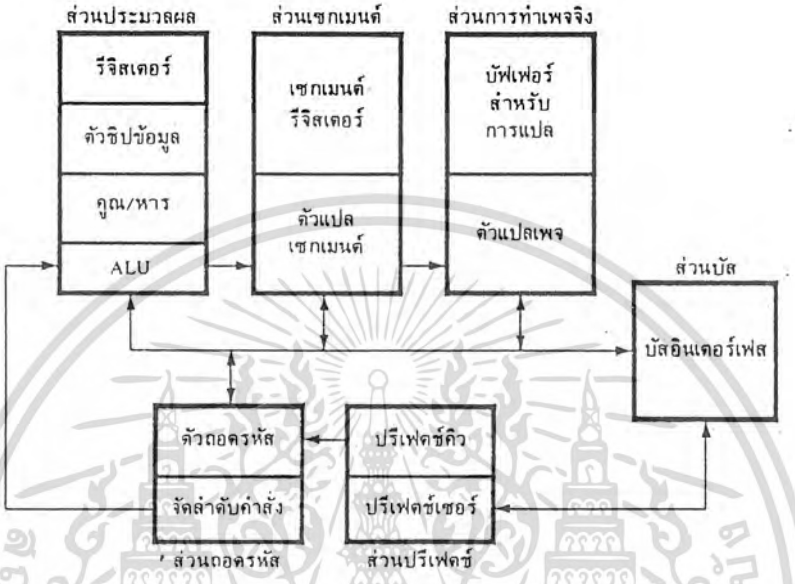
1.4 ซีพียู 80386

80386 เป็นซีพียูขนาด 32 บิต สามารถต่อหน่วยความจำ (Physical memory) ได้สูงสุด 4 GB ผลิตด้วยเทคโนโลยี CHMOS II ประกอบด้วยทรานซิสเตอร์ภายใน 275,000 ตัวมีทั้งหมด 32 ขา มีขาแอดเดรส และขาดาต้า แยกกันต่างหาก สำหรับตัวที่วิ่งที่สัญญาณนาฬิกา 20 เมกะเฮิรตซ์สามารถปฏิบัติคำสั่งได้มากกว่า 5 ล้านคำสั่งต่อวินาที ซึ่งความเร็วขนาดนี้เทียบเท่าเครื่องมินิคอมพิวเตอร์ได้อย่างสบาย ในแต่ละงาน (task) บนเครื่อง 80386 นั้นสามารถใช้เซกเมนต์ได้ 16,381 เซกเมนต์ แต่ละเซกเมนต์ใช้หน่วยความจำได้สูงสุด 4GB ซึ่งทำให้ 80386 อ้างหน่วยความจำได้สูงสุดถึง 64 TB

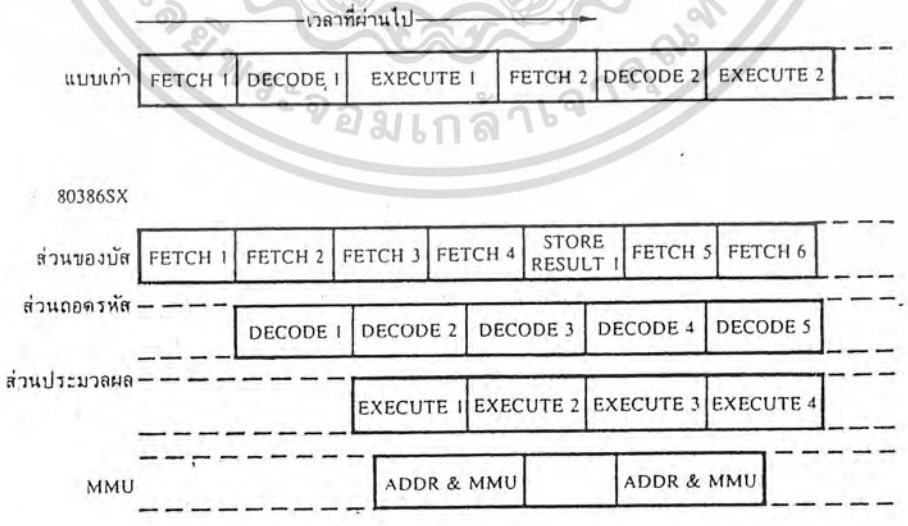
1.4.1 สถาปัตยกรรมภายใน และการทำงานของ 80386

สถาปัตยกรรมภายในของ 80386 นั้น ประกอบด้วยส่วนการทำงาน 6 ส่วน (ดังรูปที่ 1.13) ที่ทำงานอย่างขนาน ซึ่งการทำงานขนานกันนี้เราเรียกว่า ไปป์ไลน์ (pipe line) คือ แต่ละส่วนทำงานของตัวเองเสร็จก็เริ่มทำงานในช่วงต่อไป โดยไม่ต้องรอส่วนอื่นทำงานจนเสร็จเหมือนกับระบบเก่า ซึ่งความคิดนี้เห็นได้ชัด

ในรูปที่ 1.14 เมื่อส่วนติดต่อบัสนำข้อมูลส่วนที่ 1 เข้ามา เสร็จก็เริ่มนำข้อมูลส่วนที่ 2 เข้ามา ขณะที่ส่วน



รูปที่ 1.13 สถาปัตยกรรมภายในของ 80386



รูปที่ 1.14 ส่วนการทำงานของระบบภายใน 80386

ถอดรหัสเริ่มตีความหมายของข้อมูลส่วนที่ 1 และเป็นเช่นนี้เรื่อยไปกับส่วนอื่น ๆ สำหรับส่วนการทำงาน 6 ส่วน ก็ได้แก่

ส่วนติดต่อบัส เป็นตัวเชื่อมต่อระหว่าง 80386 กับอุปกรณ์รอบ ๆ ตัว 80386 เป็นตัวรับส่งข้อมูล และทางผ่านของคำสั่ง (code fetches) เป็นตัวให้สัญญาณ และประมวลผลสัญญาณของบัสไซเคิล ซึ่งมีแอดเดรส ดาต้า การเข้าถึงหน่วยความจำ อินพุต/เอาต์พุต และการใช้โคโปรเซสเซอร์

ส่วนการเฟตช์คำสั่ง จะดึงคำสั่ง (fetch) เข้ามาโดยผ่านทางส่วนติดต่อบัส จะเก็บเป็นคิวขนาด 16 ไบต์ เพื่อรอการประมวลผลโดยส่วนรหัสต่อไป

ส่วนถอดรหัส เป็นตัวแปรคำสั่งที่ได้จากคิวของส่วนการเฟตช์คำสั่งไปเป็นไมโครโค้ด การถอดรหัสคำสั่งนี้ที่ได้จะเก็บไว้ในคิวขนาด 3 คำสั่ง เพื่อรอการประมวลผลของส่วนประมวลผลต่อไป

ส่วนประมวลผล เป็นตัวประมวลคำสั่งประกอบด้วยส่วนย่อยอีก 3 ส่วน คือ

1. ส่วนควบคุม จะมีไมโครโค้ด และฮาร์ดแวร์พเศษที่เป็นแบบขนาน เพื่อความรวดเร็วในการคูณ หาร และการคำนวณแอดเดรสยังผล (EA : effective address)

2. ส่วนของข้อมูล จะมี ALU รีจิสเตอร์ใช้งานทั่วไปขนาด 32 บิต อยู่ 8 ตัว มีตัวชิปข้อมูลขนาด 64 บิต ซึ่งสามารถชิปได้ทีละหลายๆ บิต ใน 1 สัญญาณนาฬิกาและเป็นตัวจัดข้อมูลให้กับส่วนควบคุม

3. ส่วนการทดสอบการป้องกัน เป็นตัวเช็คเซกเมนต์ว่าเกิดการผิดพลาดหรือไม่

ส่วนเซกเมนต์ เป็นตัวแปลแอดเดรสเชิงตรรก ให้เป็นแอดเดรสเชิงเส้นให้กับหน่วยประมวลผล จะมี segment descripter cache เก็บเซกเมนต์ที่กำลังใช้งานอยู่เพื่อความรวดเร็วในการแปล

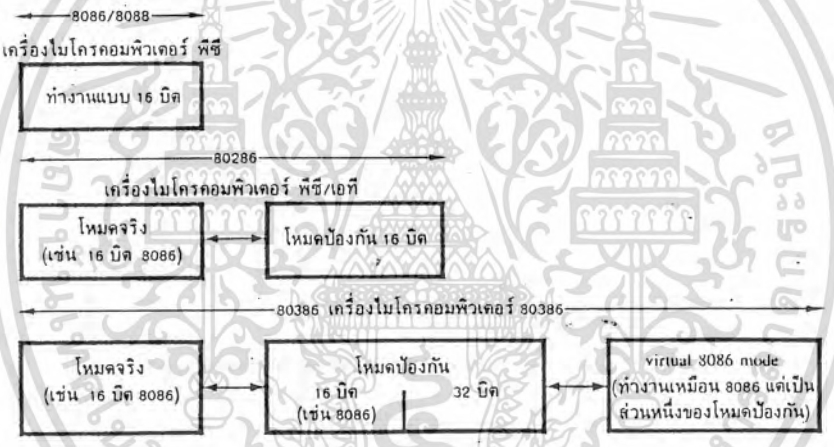
ส่วนเพจจิง เมื่อใช้ 80386 โดยมีเพจจิง (paging) ส่วนนี้ จะแปลแอดเดรสเชิงเส้นที่ได้จากส่วนเซกเมนต์หรือส่วนเฟตช์คำสั่งให้เป็นฟิสิคัลแอดเดรส ถ้าไม่มีการทำเพจจิง ก็จะไม่มีการแปลเกิดขึ้น เช่นเดียวกับเซกเมนต์ ส่วนนี้จะมี page descripter cache เพื่อความรวดเร็วในการแปลฟิสิคัลแอดเดรสที่ได้นี้จะถูกส่งให้ส่วนติดต่อบัสต่อไป

1.4.2 การทำงานของ 80386 จะแบ่งเป็นโหมดการทำงาน 3 โหมด (ดังรูปที่ 1.15) คือ

โหมดจริง สำหรับโหมดนี้ จะเป็นโหมดที่เข้ากันได้กับซอฟต์แวร์ของ 8086 ที่มีอยู่เดิม สามารถอ้างแอดเดรสได้สูงสุด 11 MB และมีขนาดเซกเมนต์ได้แค่ 64 KB แต่สามารถใช้รีจิสเตอร์ขนาด 32 บิต และคำสั่งของ 80386 เกือบทั้งหมดได้

โหมดป้องกัน การทำงานในโหมดนี้จะใช้ความสามารถของ 80386 ได้เต็มความสามารถมีการทำงานแบบเพจจิง สามารถอ้างแอดเดรสได้สูงสุด 64 TB สำหรับโหมดที่ทำงานเป็น 80286 จะอ้างแอดเดรสได้สูงสุด 16 MB สนับสนุนการทำงานแบบมัลติทาสกิง

เวอร์ชวล 8086 โหมด (virtual 8086 mode) เป็นโหมดของ 8086 ที่ถูกสร้างในโหมดป้องกัน ซึ่งจะทำให้เรารู้สึกว่ามี 8086 ได้หลาย ๆ ตัวพร้อมกันโดยใช้ความสามารถของกลไกการป้องกัน (protected machanism) เช่น ขณะที่คนหนึ่งทำงานสเปรดชีต อีกคนใช้เอ็มเอสดอส และขณะเดียวกันอีกคนเล่นยูนิกซ์อยู่ ซึ่งนั้นก็คือ การทำงานแบบมัลติยูสเซอร์



รูปที่ 1.15 การแบ่งโหมดการทำงาน 3 โหมดของ 80386

1.5 ซีพียู 80486

80486 สร้างขึ้นมาด้วยเทคโนโลยี CHMOS IV ขนาด 1 ไมครอน มีทรานซิสเตอร์ 1.2 ล้านตัว มีคุณสมบัติดังนี้

1. มีแคชอยู่บนตัว 80486 เอง ดังรูปที่ 1.16
   - ขนาดของแคช คือ 8 KB ใช้ลักษณะการเข้าถึงแบบ four-way และมีขนาดไลน์ 16 ไบต์ (คำว่า four-way และ line) จะกล่าวภายหลัง
   - การทำแคชนี้เป็นการทำทั้งคำสั่ง (code) และข้อมูล (data)

3. ทำ Burst-mode bus ได้ถึง 106 MB ต่อวินาที ในซีพียูขนาด 33 MHz

4. สนับสนุนการประมวลผลแบบมัลติโปรเซสเซอร์

   - มีแคชในตัว (ทำให้การติดต่อกับอุปกรณ์หรือตัวประมวลผลอื่นได้ดียิ่งขึ้น)

   - มีคำสั่งสนับสนุน (new instructions)

   - มีขาสนับสนุน (new pins)

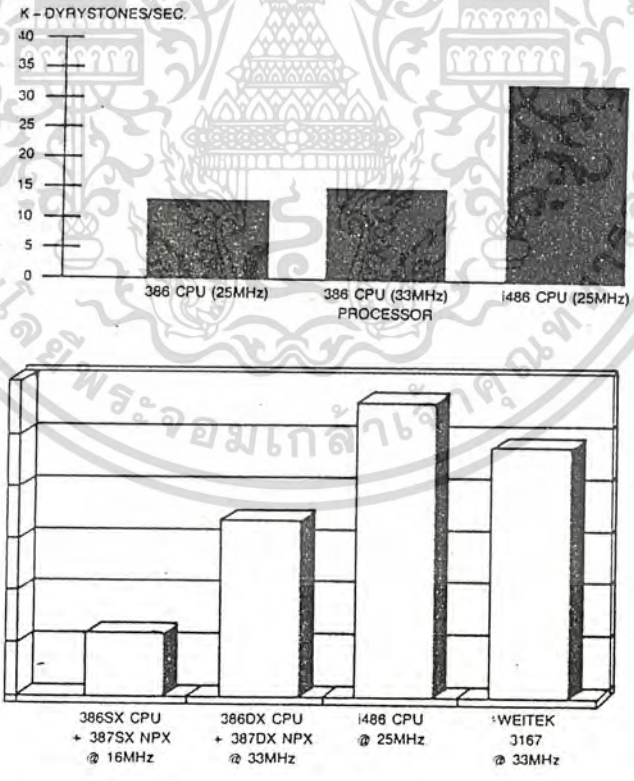
   - สามารถต่อแคชคอนโทรลเลอร์เพิ่มได้ (second level cache controller)

5. ความสามารถ ดังรูปที่ 1.17

   - เร็วกว่า 386DX ประมาณ 2-4 เท่า

   - ประมวลผลได้ 15-2-0 VAX MIPS ที่ 25-33 MHz

   - คำนวณได้ 1.6-2.00 MFLOP FPU



รูปที่ 1.17 กราฟแสดงความสามารถของ 80486

1.5.1 การทำงานในโหมดต่าง ๆ

การทำงาน 80486 มีโหมดการทำงานเช่นเดียวกับ 80386 คือ เรียลโหมด โปรเทคโหมด และเวอร์ชวล 8086 โหมด (จะถือว่ารวมอยู่ในโปรเทคโหมดก็ได้) ดังรูปที่ 1.18 ซึ่งความสามารถก็มีดังนี้

1. เรียลโหมด (Real Mode)

มีการทำงานเป็น 8086 ด้วยความเร็วสูง มีขนาดเซกเมนต์ 64 KB มีการกระทำและการอ้างแอดเดรสเป็น 16 บิต และช่วงการอ้างแอดเดรสสูงสุด 1 MB เช่นเดียวกับ 8086

2. โปรเทคโหมด (Protected Mode)

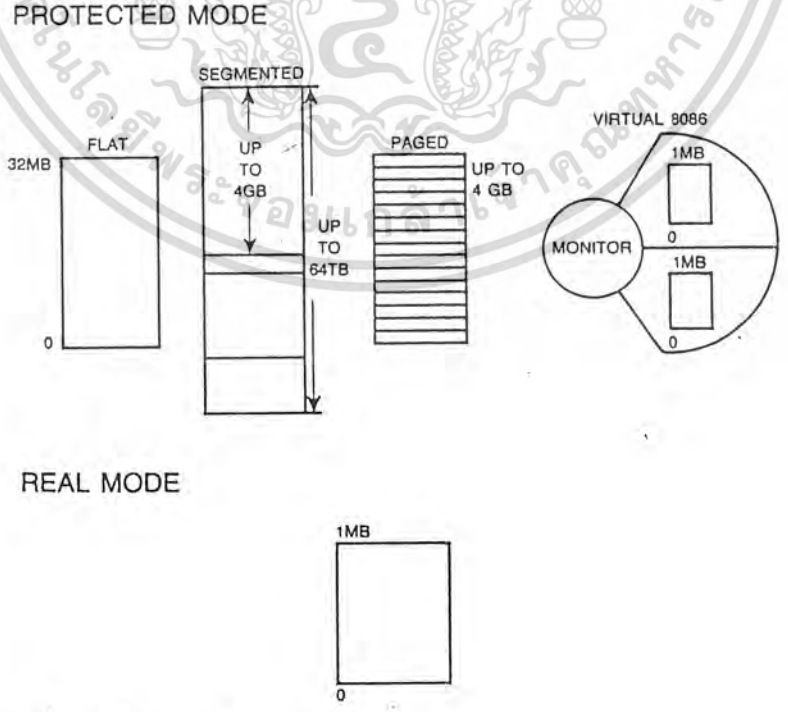
เป็นการใช้ความสามารถของ 80486 เต็มที่ ซึ่งแล้วแต่การกำหนดลักษณะการใช้หน่วยความจำดังนี้

2.1 เซกเมนต์ (segmented) สามารถมีเซกเมนต์ได้ขนาดใหญ่สูงสุด 4 GB และสามารถอ้างลอจิคัลแอดเดรสแบบต่อเนื่อง (เวอร์ชอลแอดเดรส) ได้ใหญ่ที่สุด 64 TB ดังรูปที่ 1.19

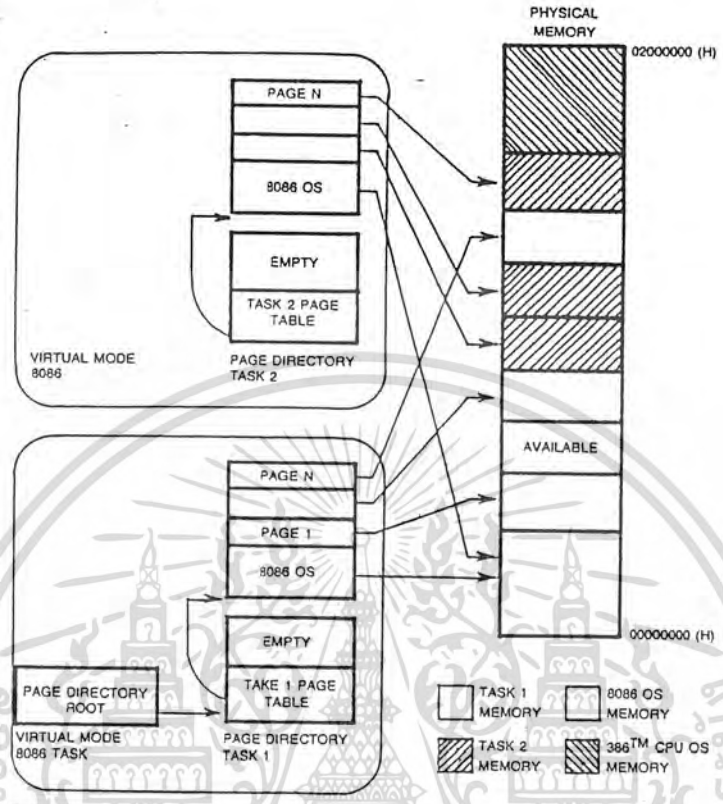
2.2 แฟล็ต (flat) การอ้างแอดเดรสแบบต่อเนื่อง (single linear address space) สามารถทำได้สูงสุด 4 GB

2.3 เพจ (paged) อ้างได้สูงสุด 4 GB

2.4 ไฮบริด (hybrid) เป็นการใช้หน่วยความจำแบบเซกเมนต์และเพจร่วมกันรีจิสเตอร์



รูปที่ 1.18 โหมดการทำงานของ 80486



รูปที่ 1.19 ลักษณะการใช้หน่วยความจำแบบ Virtual Mode

รีจิสเตอร์ของ 80486 เหมือนกับ 80386 ยกเส้นการนำรีจิสเตอร์ของ 80387 เข้ามาร่วมอยู่บนตัวเองและรีจิสเตอร์เพิ่มขึ้นมาใหม่ คือ รีจิสเตอร์สำหรับการทดสอบ (TR3-TR5) และมีการปรับปรุง Translate look aside Buffer (TLB) ให้ดีขึ้น เช่นไม่ต้องหยุดทำเพจจิ้งขณะทำการทดสอบดังรูปที่ 1.20, 1.21 และ 1.22 นอกจากรีจิสเตอร์ที่เพิ่มขึ้นมาแล้ว รีจิสเตอร์บางตัวก็ได้ถูกแก้ไขเพิ่มเติมด้วยดังนี้

รีจิสเตอร์ CRO

เพิ่มบิต

CD - Cache Disable

NW - No Write Through

AM - Alignment Mask

WP - Write Protect

NE - Numerics Exception

ตัดบิตที่ 4 ทิ้ง คือ

ET - Extention Type

รีจิสเตอร์ EFLAGS

เพิ่ม

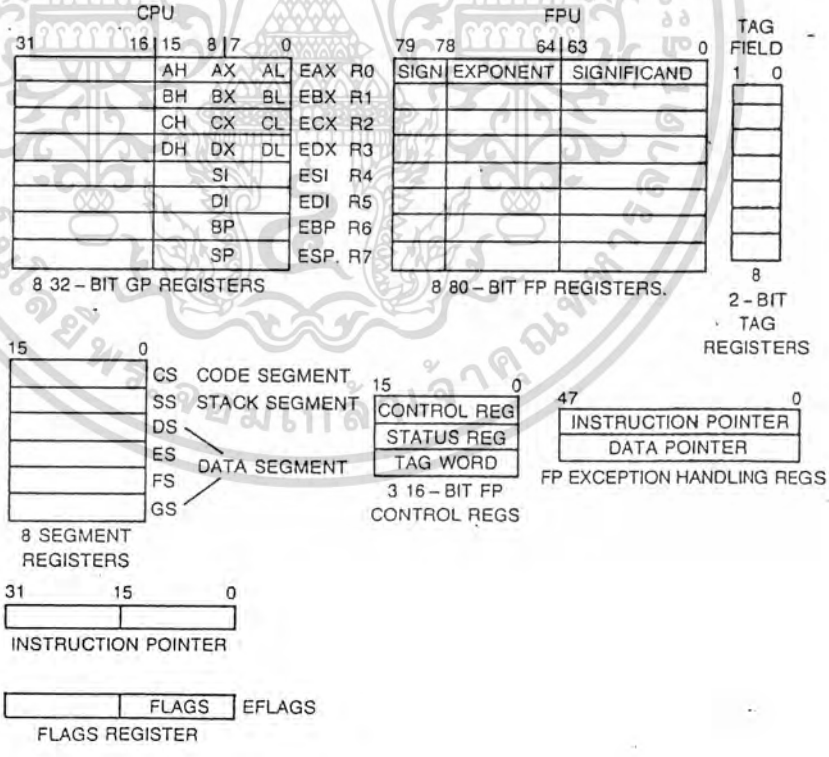
AC - Alignment check

รีจิสเตอร์ CR3

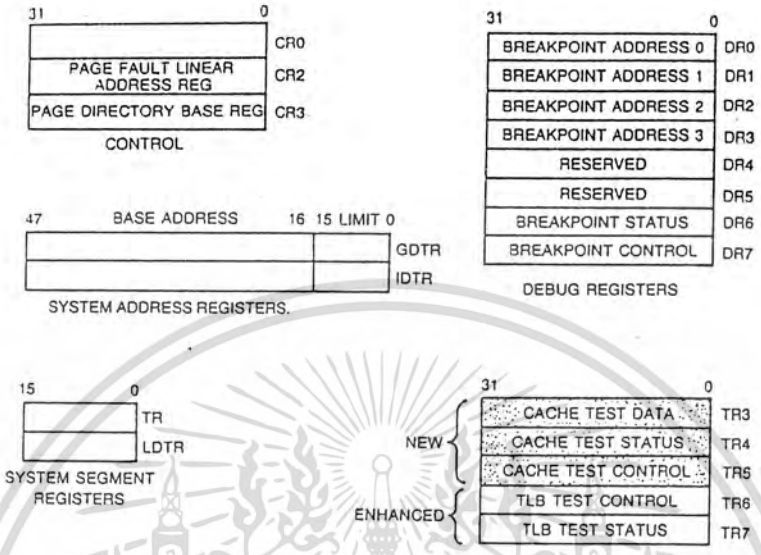
เพิ่ม

PCD - Page Cache Disable

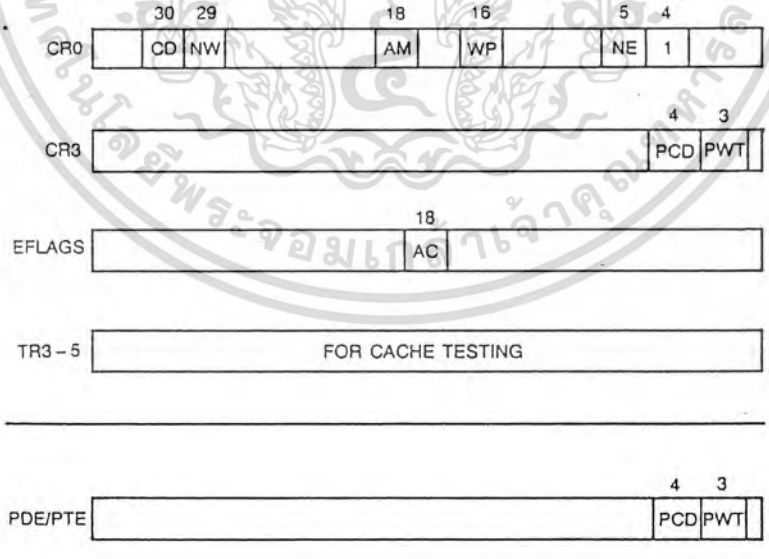
PW - Page Write Through



รูปที่ 1.20 รีจีสเตอร์ใช้งาน (Application level register)



รูปที่ 1.21 รีจีสเตอร์ระบบ (System level register)



รูปที่ 1.22 รีจีสเตอร์ที่ต่างจาก 80386

1.5.2 ชนิดของข้อมูล

80486 สนับสนุนการใช้ข้อมูลของ 80386DX และ 80387DX ดังนี้

1 unsigned ordinal

- ขนาด 8, 16 และ 32 บิต (CPU)

2. signed integer

- ขนาด 8 บิต (CPU) , 16 บิต (CPU,FPU) , 32 บิต (CPU,FPU) , และ 64 บิต (FPU)

3. floating point (มาตรฐาน IEEE 754)

- 32 bit single precision (FPU)
- 64 bit double precision (FPU)
- 80 bit extended precision (FPU)

4. binary coded decimal

- 8 bit packed/unpacked (CPU)
- 80 bit packed (FPU)

5. string

- Byte, WORD, DWORD (1-4 gigabyte)
- bit (สูงสุด 4 gigabyte)

6. ASCII

- byte (CPU)

1.5.3 Big Endian และ Little Endian

ซีพียูตระกูล 80x86 ที่ผ่านมาจะใช้ข้อมูลแบบ Little Endian นั้นคือ ไบต์ล่างจะอยู่ที่ แอเดรสล่าง ๆ และไบต์สูงจะอยู่ที่แอดเดรสสูง เช่น เราเขียนโปรแกรมว่า mov AX,3231h ในหน่วยความจำก็จะเป็นดังตารางที่ 1.5 เช่นเดียวกันกับข้อมูลแบบ DWORD = '4321'ในหน่วยความจำก็จะเป็นดังตารางที่ 1.6

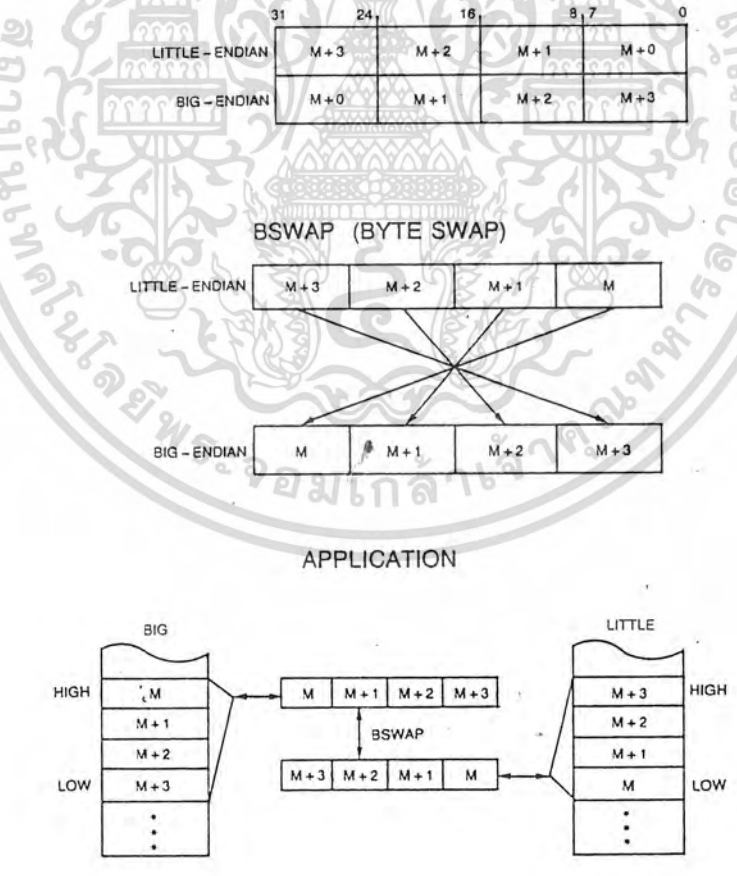
นั่นคือ เรียงไบต์ล่างที่หน่วยความจำล่างไปหาไบต์สูงที่หน่วยความจำตำแหน่งสูง ซึ่งตรงกันข้ามกับ Big Endian ที่เรียงในทางตางกันข้าม สำหรับ 80486 นั้นสนับสนุนทั้ง :Little Endian และ Big Endian โดยมีคำสั่งใหม่ BS-WAP (Byte Swap) เพื่อทำงานร่วมกับตัวประมวลอื่น ๆ ที่ใช้ข้อมูลแบบ Big Endian ได้ นั้นคือสนับสนุนการทำงานแบบมัลติโปรเซสเซอร์ให้สมบูรณ์มากขึ้นนั่นเอง ดังรูปที่ 1.23

ตำแหน่งที่	n	n+1
ข้อมูล	31	32

ตารางที่ 1.5

ตำแหน่งที่	n	n+1	n+2	n+3
ข้อมูล	'1'	'2'	'3'	'4'

ตารางที่ 1.6



รูปที่ 1.23 LITTLE ENDIAN และ BIG ENDIAN

1.5.4 คำสั่ง (Instruction set)

คำสั่งของ 80486 มีความเข้าใจกันได้ (compatible) กับ 80386DX และ 80387DX และได้เพิ่มคำสั่ง ดังนี้

INVD (invalid line in code/data cache)

WBINVD เหมือนกับ INVD ต่างกันที่การนำเสนอ (Write-back and Invalidate data cache)

INVLPG (invalidates a page mapped into the TLB)

XADD (Exchange and Add)

CMPXCHG (Compare and Exchange)

BSWAP ใช้ในการสลับข้อมูลระหว่าง Big-endian กับ Little endian

นอกจากจะมีคำสั่งเพิ่มขึ้นมาแล้ว ยังมีโอเปอรแรนด์เพิ่มขึ้นมาอีก คือ TR3 ,TR4 และ TR5 ตัวอย่างเช่น

mov TR5,EAX

mov EBX,TR3

1.5.5 อินเตอร์รัพต์

กลไกการเกิดอินเตอร์รัพต์ของ 80486 นั้นเหมือนกับ 80386 ดังตารางที่ 1.7 ยกเว้นมีการเพิ่มชนิด (type) ที่ 17 เพื่อใช้ในการทำ alignment checking และอินเตอร์รัพต์ชนิดที่ 16 จะเกิดภายในเท่านั้น และชนิดที่ 9 จะไม่เกิด ทั้งนี้เพราะตัวประมวลผลคณิตศาสตร์ (FPU) ได้รวมเข้ามาไว้ในตัว 80486 เองแล้ว

1.5.6 โครงสร้างภายใน

โครงสร้างภายในประกอบด้วยหน่วยต่าง ๆ คล้ายกับ 80386 ดังรูปที่ 1.24 และ 1.25 ซึ่งจะดูความคล้ายคลึงและแตกต่างได้จากตารางที่ 1.8 โครงสร้างภายในของ 80486 ได้ทำการปรับปรุงให้มีความสามารถเหมือนตัวประมวลแบบ RISC เช่น การ push, POP, load, store, add substract, การกระทำทางลอจิกและการใช้ shift สามารถทำได้ภายใน 1 clock แต่อย่างไรก็ตาม 80486 ก็ไม่ใช้ RISC ทีเดียว โดยดูจากข้อแตกต่างได้ดังตารางที่ 1.9 แล 1.10

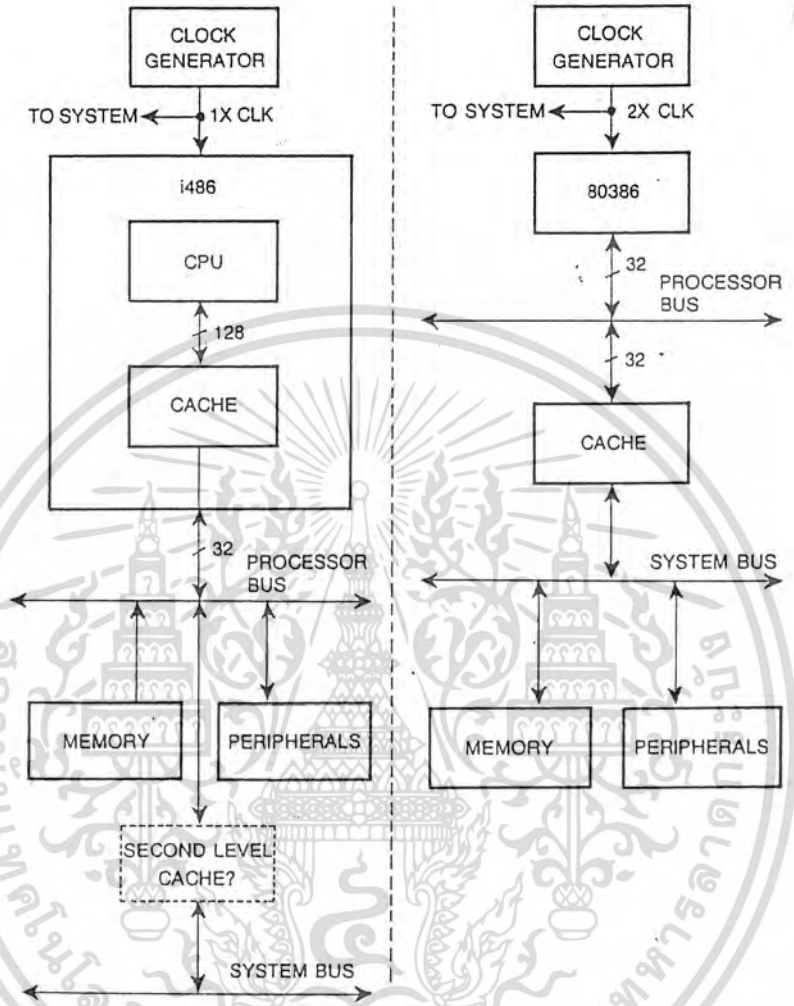
Function	Interrupt Number	Instruction Which Can Cause Exception	Return Address Points to Faulting Instruction	Type
Divide Error	0	DIV, IDIV	YES	FAULT
Debug Exception	1	Any Instruction	YES	TRAP*
NMI Interrupt	2	INT 2 or NMI	NO	NMI
One Byte Interrupt	3	INT	NO	TRAP
Interrupt on Overflow	4	INTO	NO	TRAP
Array Bounds Check	5	BOUND	YES	FAULT
Invalid OP-Code	6	Any Illegal Instruction	YES	FAULT
Device Not Available	7	ESC, WAIT	YES	FAULT
Double Fault	8	Any Instruction That Can Generate an Exception		ABORT
Intel Reserved	9			
Invalid TSS	10	JMP, CALL, IRET, INT	YES	FAULT
Segment Not Present	11	Segment Register Instructions	YES	FAULT
Stack Fault	12	Stack References	YES	FAULT
General Protection Fault	13	Any Memory Reference	YES	FAULT
Page Fault	14	Any Memory Access or Code Fetch	YES	FAULT
Intel Reserved	15			
Floating Point Error	16	Floating Point, WAIT	YES	FAULT
Alignment Check Interrupt	17	Unaligned Memory Access	YES	FAULT
Intel Reserved	18-32			
Two Byte Interrupt	0-255	INT n	NO	TRAP

* Some debug exceptions may report both traps on the previous instruction, and faults on the next instruction.

ตารางที่ 1.7 ค่าอินเตอร์รัพต์ต่างๆ

i486	80386	Comments
Datapath unit	Execution unit	Faster
Segmentation unit	(same)	Faster
Paging unit	(same)	• New caching bits • Faster
Prefetcher	Code prefetch unit	32-byte queue
Instruction Decode	Instruction Decode Unit	• Two-stage decode • Flushed for jmp, call
Bus Interface	Bus Interface Unit	• No bus pipelining • 21 new pins • No 387 interface • Burst mode added • Four write buffers • Eight-bit bus support
Floating Point Unit	(none)	• Faster than 387 DX • 64-bit interunit bus
Cache Unit	(none)	128 bits in one clock for prefetch hit
Control and Protection Test Unit	(similar)	Controls both FPU and Datapath Unit

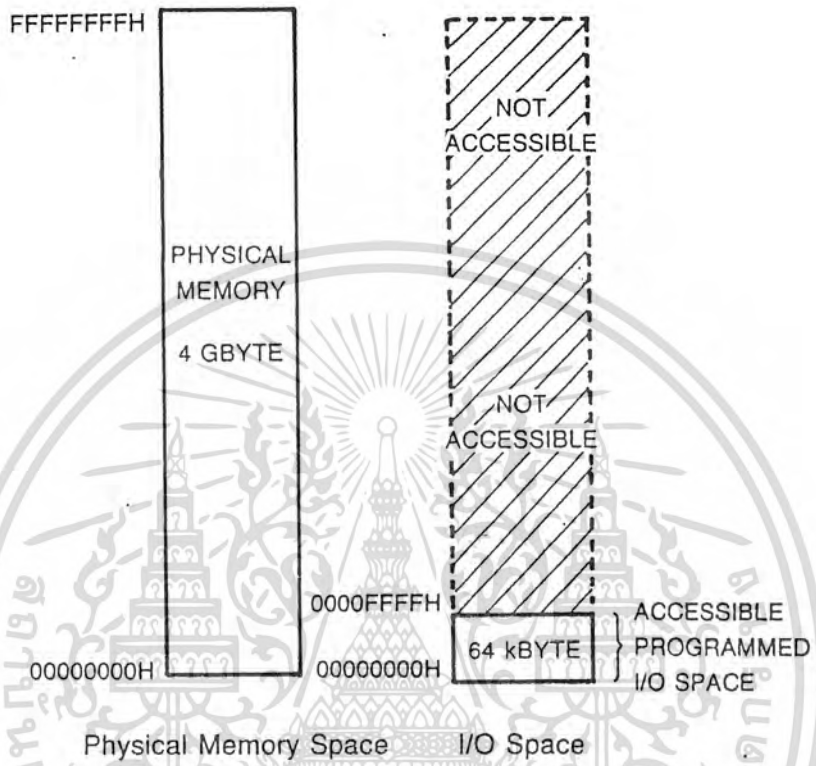
ตารางที่ 1.8 โครงสร้างภายใน 80486 เทียบกับ 80386



รูปที่ 1.24 โครงสร้างของระบบ 80486

RISC	i486™ Microprocessor
Load/store architecture	ALU operations allowed on memory operands
Fixed length instructions	One to fifteen bytes
Hardwired control	Has microcode
Small instruction set (~53)	141 integer instructions 73 FPU instructions
Minimalist hardware	BIST, TLB test register, on-chip cache test registers, breakpoint registers, ICE support

ตารางที่ 1.9 เปรียบเทียบระหว่าง 8048■6 กับ RISC



รูปที่ 1.25 Physical Address และ I/O

Instruction Type	Clock Counts			
	386 DX CPU	i486 CPU	SPARC	88000 (est)
Load	4	1	2	1-3
Store	2	1	3	1
Reg/Reg	2	1	1	1
Jump	9/3	3/1	1/2	2/1
Call	9	3	3	1

ตารางที่ 1.10 จำนวน clock ที่ใช้ระหว่าง 80386 80486 และ RISC บางตัว

1.5.7 แคช (Cache)

นอกจากจุดเด่นที่ 80486 รวมเอา 80387DX เข้าไว้ในตัวแล้ว จุดเด่นอีกอย่างก็คือ 80486 มีแคชอยู่ภายใน 8 KB เราจึงควรศึกษาแคชไปพร้อมกันไปด้วย

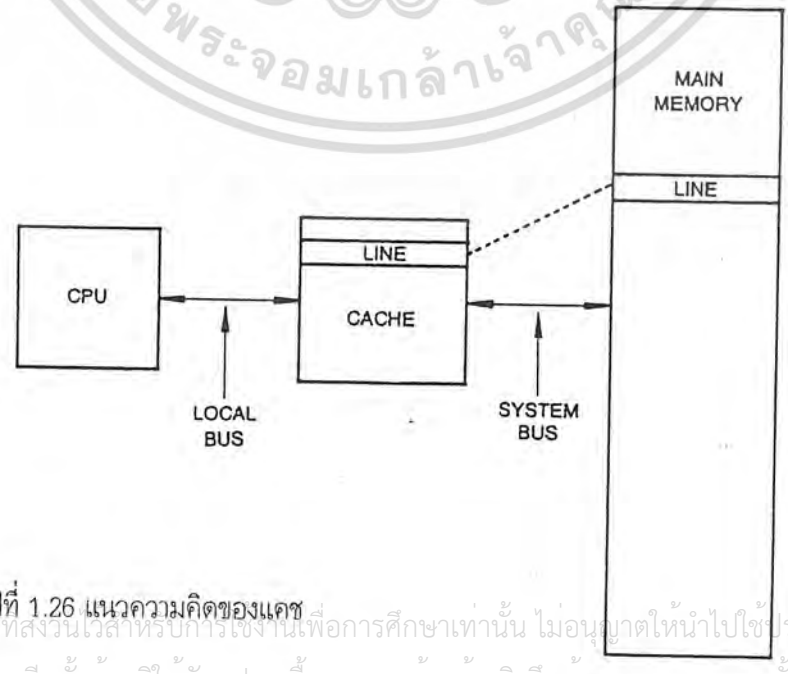
จุดประสงค์และหลักการของแคช

จุดประสงค์ของแคชก็คือ การทำซีพียูมองเห็นหน่วยความจำที่ใช้อยู่ (ช้า) ให้เหมือนกับหน่วยความจำที่เร็ว คำว่า ช้า ก็คือ จะมี 1 หรือหลาย ๆ ช่วงเวลาหยุดรอ (wait states) และเร็วก็คือ ไม่มีช่วงเวลาหยุดรอ (zero wait state) หน่วยความจำที่เราใช้กันอยู่ก็คือ DRAM ซึ่งช้าแต่ก็ต้องใช้เพราะราคาถูกกว่าหน่วยความจำที่มีความรวดเร็ว และราคาแพง นั้นคือ SRAM โครงสร้างของแคชจึงสร้างจาก SRAM การทำงานของแคช ก็คือพยายามดึงข้อมูลที่ถูกใช้บ่อย (frequently) หรือพึ่งใช้ไปมาไว้บนตัวของแคชเอง เมื่อซีพียูต้องการข้อมูลนั้น จึงสามารถนำจากแคช (ซึ่งมีความเร็วสูง) ได้ทันที เช่นเดียวกับการเขียนข้อมูลแคช คอนโทรลเลอร์จะเป็นตัวจัดการให้ดังรูปที่ 1.26 นอกจากจะรู้หลักการแล้ว เราควรรู้จักคำที่ใช้กับระบบแคช บ่อย ๆ ดังนี้ คือ

line จำนวนหน่วยความจำที่น้อยที่สุดที่มีการเคลื่อนย้ายระหว่างแคชกับหน่วยความจำหลักในการทำปรับปรุง (update Cache)

miss ไลน์ที่อ้างถึงไม่ได้อยู่ในแคช

hit ไลน์ที่อ้างถึงอยู่ในแคช



รูปที่ 1.26 แนวความคิดของแคช

การจัดโครงสร้างของแคช

แคช จะมีหน่วยความจำ 2 ประเภทอยู่บนตัวเอง คือ

tag จะเป็นตัวบรรจุข้อมูลของ upper-order address ซึ่งเป็นตัวบอกหน้า (page) ของหน่วยความจำหลักที่จะใช้

date เป็นตัวเก็บข้อมูลที่เหมือนกับหน่วยความจำหลักที่ตำแหน่งตัวชี้ (index) ในหน้านั้นขอให้ดูรูปไดเรคแม็พประกอบ

นอกจากแท็กและดาต้าแล้วยังมีคำอีก 2 คำที่ใช้กับการจัดการแคช นั้น คือ

index เป็นตัวบรรจุ low order address ซึ่งเป็นตัวชี้ของข้อมูลในหน้าที่อ้างอิง

LRU (least-recently used) เป็นตัวบอกว่าไลน์ใดใน n-way cache ที่ถูกปรับปรุงหรือถูกใช้น้อยที่สุด ซึ่งเป็นตัวบอกว่าไลน์นั้นจะถูกเขียนทับเมื่อเกิด read miss

การจัดโครงสร้างของแคชจะเป็นไดเรคแม็พ (Dirct Mapped)

ค่าอินเด็กซ์ที่จะให้ถูกจับคู่ (mapped) บนหน้าเดียวเท่านั้นบนหน่วยความจำหลัก ซึ่งจะทำให้ไม่มี LRU เพราะถ้าไลน์ใด miss แล้วไลน์นั้นจะถูกเขียนทับ การ hit จะเกิดเมื่อค่า upper order addresss ที่ให้จากซีพียูตรงกับค่าเพจที่ตำแหน่งอินเด็กซ์ในหน่วยความจำแท็ก ดังรูปที่ 1.27

Two way set

ค่าอินเด็กซ์ที่ให้จะถูกจับคู่ได้ 2 กรณีในแต่ละทาง (way) ซึ่งเป็นการเพิ่มอัตราการ hit นั้นเอง (การ hit จะเกิดขึ้นเมื่อค่า upper order address ตรงกับค่าเพจในแท็กของ way ใด way หนึ่งในตำแหน่งอินเด็กซ์ที่ให้มา) LRU จะถูกปรับปรุงเป็นไลน์ที่เกิด miss ขึ้น

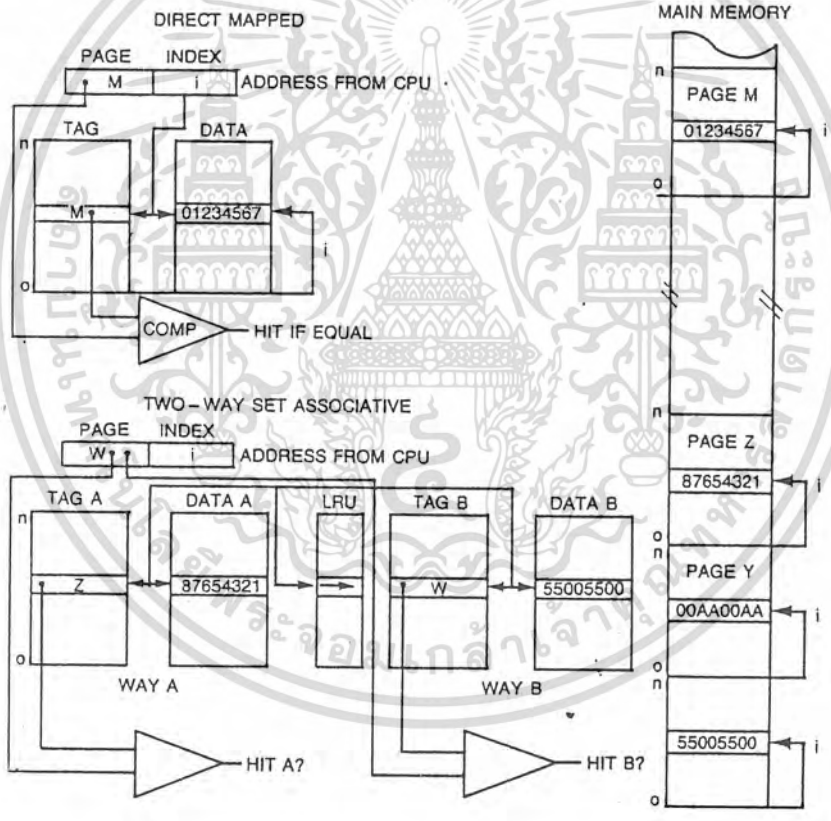
Four way set

เป็นการเพิ่มขยายของ two way set ทั้งนี้เพื่อเพิ่มอัตราการ hit นั้นเอง ใน 80486 ก็มี Four way set แคชอยู่ 2 ที่ คือ Paging Unit (TLB) และ Cache Unit (code/data cache) ดังรูปที่ 1.28

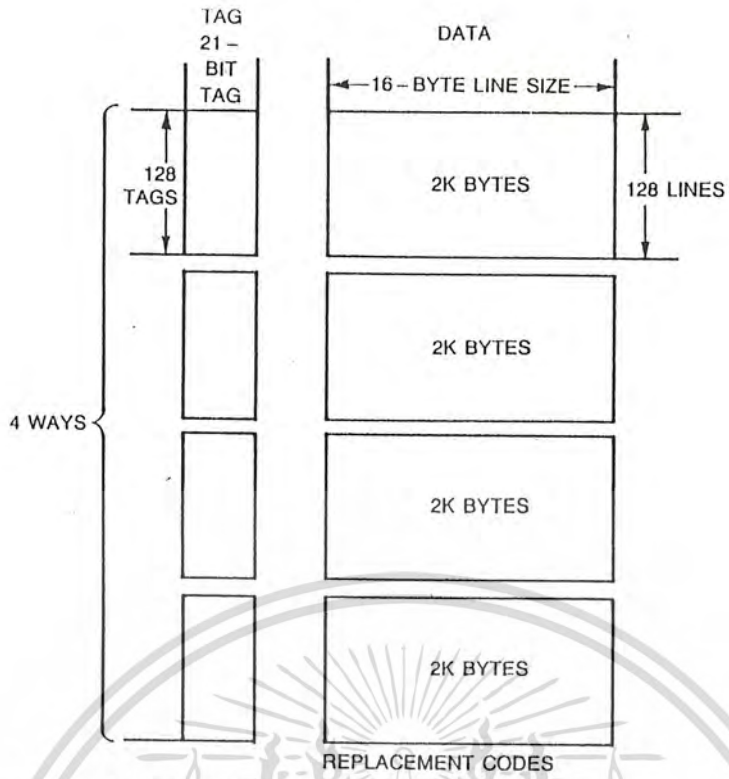
Buffered WRITE SNOPPING

การจะทำให้หน่วยความจำหลักมีค่าสอดคล้องกับแคชขณะที่ซีพียูทำการเขียนข้อมูลกับแคชซึ่งเรียกว่า write-through update policy 'วิธีที่ง่าย ๆ ก็คือ เขียนข้อมูลไปที่หน่วยความจำหลักทุกครั้งเมื่อเขียนไปที่ 'post' หรือ 'buffer' ของแคช ซึ่งข้อมูลที่เขียนไม่ใช่มีเพียงตัวข้อมูล (data) แต่รวมถึงแอดเดรสและค่าควบคุม (address and control information) ด้วย ซึ่งการเขียนไปที่บัฟเฟอร์นั้นแคชคอนโทรลเลอร์

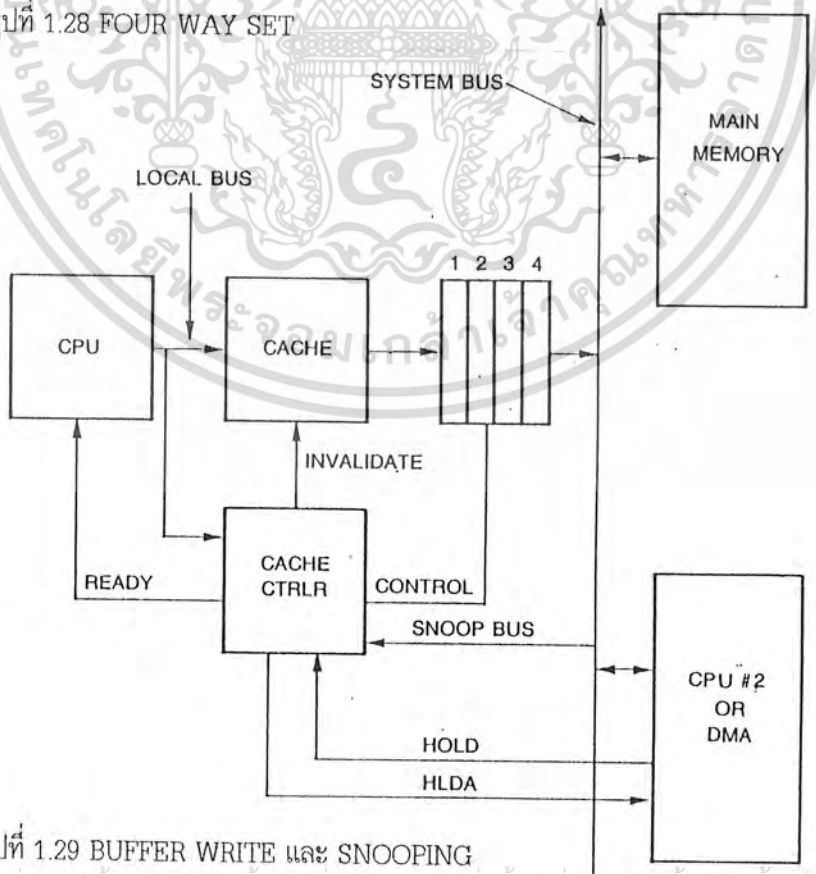
จะเป็นตัวจัดการเขียนไปที่หน่วยความจำหลักได้เอง ทำให้ซีพียูเขียนข้อมูลด้วยความเร็วสูงที่ลอคัลบัส โดยแคชคอนโทรลเลอร์เขียนที่ซิสเต็มบัสให้เอง ดังรูปที่ 1.29



รูปที่ 1.27 DIRECT MAPPED และ TWO WAY SET



รูปที่ 1.28 FOUR WAY SET



รูปที่ 1.29 BUFFER WRITE และ SNOOPING

เมื่อระบบมีการใช้ตัวประมวลผล 2 ตัว หรือหลายตัวขึ้นไปกับระบบแคช จะทำให้เกิดปัญหากับการเขียนข้อมูลลงหน่วยความจำหลัก ถ้าข้อมูลที่เขียนเกิดจากตัวประมวลผลตัวอื่น (ตัวที่ไม่ได้มีแคช) บังเอิญตรงกับส่วนที่ทำการแคชไว้ จะทำให้หน่วยความจำหลักกับส่วนที่อยู่ในแคชไม่สอดคล้องกัน ซึ่งเราพอจะมีวิธีการแก้ได้ดังนี้

1. ทำให้หน่วยความจำที่จะใช้ร่วมกัน ไม่ให้เกิดการแคช (non cacheable) ทำได้โดยฮาร์ดแวร์ ซึ่งมีผลเสียกับความสามารถของระบบ

2. ใช้แคชร่วมกัน นั่นก็คือ ขณะที่ตัวประมวลตัวแรกใช้แคช อยู่ ตัวอื่นต้องหยุดรอก่อน ซึ่งมีผลให้เกิด wait state

3. Snoop โดยแคชคอนโทรลเลอร์จะคอยดูซิสเต็มส์บัส ในระหว่างการเขียนเมื่อตัวเองอยู่ในสภาวะ hold ถ้าตัวประมวลผลตัวอื่นเขียนข้อมูลที่แอดเดรสตรงกับในแคช ตัวควบคุมแคชจะทำการ invalidate line นั้น เพื่อให้เกิดการปรับปรุงแคชเช่นเดียวกับ read miss update

1.5.8 ขาสัญญาณของ 80486

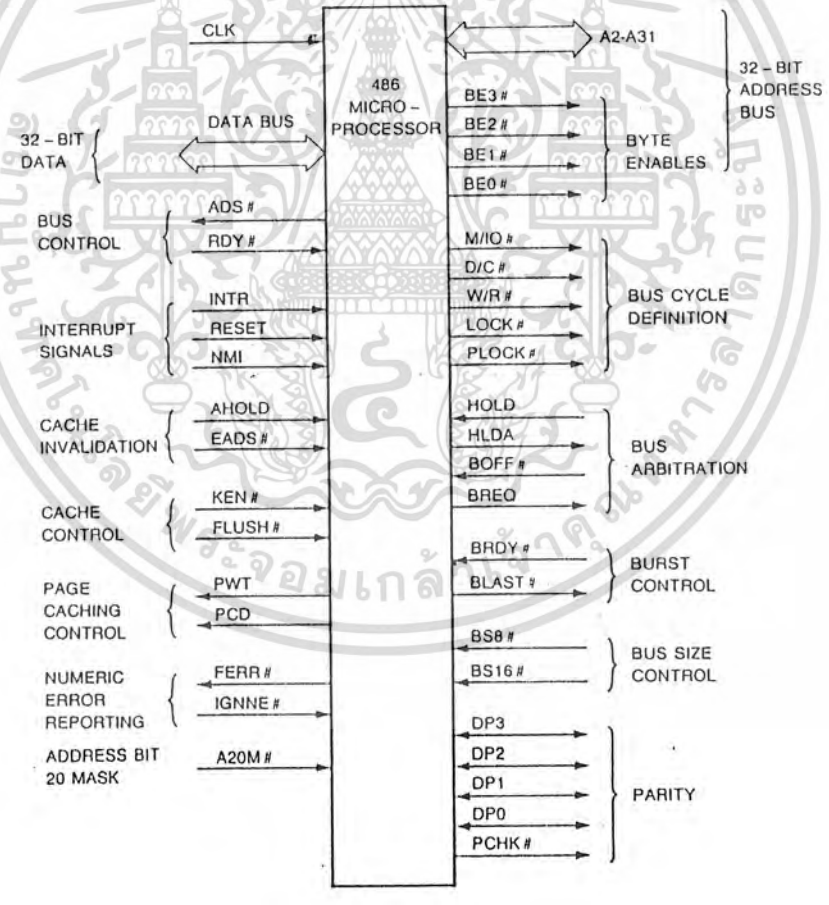
80486 นั้นรวมเอา 80386 และ 80387 เข้าไว้ในตัวเองและแถมยังแคชในตัวอีก ทำให้ 80486 มีขาทั้งหมด 168 ขา ขณะที่ 80386 มีแค่ 132 ขา ดังรูปที่ 1.30 ซึ่งจะได้เห็นข้อแตกต่างได้จากตารางที่ 1.11 แต่กระนั้น 80486 จะไม่มีขาบางขาที่ 80386 มี ทั้งนี้เพราะไม่จำเป็นต้องมีนั่นเอง ดังตารางที่ 1.12

Group	i486	386	Comments
CLK	CLK	CLK2	• TTL compatible • 1X clock
32-bit address bus	A2-A3 BE0#-BE3#	(same)	A4-A31 are bidirectional
Data bus	D0-D31	(same)	No byte-swap logic on-chip
Interrupt	Reset INTR, NMI	(same)	Different way to invoke self-test at reset
Bus arbitration	HOLD, HLDA BOFF#, BREQ#	HOLD, HLDA	New pins provide new flexibility
Bus size control	BS8#, BS16#	BS16#	No byte swap logic on-chip
Bus cycle definition	M/IO#, D/C#, W/R# LOCK#, PLOCK#	M/IO#, D/C#, W/R#, LOCK#	• Different decode for M/IO #, D/C#, W/R# • PLOCK # imacates burst
Parity	PCHK# DP0-DP3	(none)	Even parity generation and check
Cache invalidation	AHOLD, EAD5#	(none)	Used in snooping
Cache control	KEN#, FLUSH#	(none)	• KEN# = cache enable • FLUSH # = 4 clock flush
Page caching control	PWT, PCD	(none)	Used with second-level cache
Numeric error reporting	FERR#, IGNNE #	(none)	DOS-compatibility pins for FPU errors
Address bit, 20 mask	A20M#	(none)	IMB wrap-around support for cache, etc.
Burst control	BRDY#, BLAST#	(none)	• Maintain burst • Signal end of burst

ตารางที่ 1.11 เปรียบเทียบขาของ 80486 และ 80386

Pin Name	Implication
NA #	No pipelined cycles
PEREQ, ERROR #, BUSY #	No 387 support

ตารางที่ 1.12 ขาที่ 80386 มีแต่ 80486 ไม่มี



รูปที่ 1.30 ขาต่างๆของ 80486

1.5.9 ข้อแตกต่างระหว่าง 80486 กับ 80386 + 80387

1. จำนวน clock ใน 1 คำสั่งจะถูกลดลง ทำให้มีประสิทธิภาพในการทำงานมากขึ้น ดังตารางที่ 1.13

Instruction Type	Clock Counts	
	386™ CPU	i486™ CPU
LOAD	4	1
STORE	2	1
REG/REG	2	1
JUMP	9/3	3/1
CALL	9	3
Avg. CPI	4.5	1.8

Instruction Type	Clock Counts	
	387DX	i486™
FLD	14	3
FST	11	3
FADD/FSUB	23-31	10
FMUL	29-57	16
FDIV	88	73

ตารางที่ 1.13 จำนวน clock ที่ใช้ระหว่าง 80486 กับ 80386+80387

2. ระบบบัสมีความเร็วขึ้นและมีข้อแตกต่าง คือ 1X clock, parity support, burst cycles, cacheable cycles, cache invalidate cycles และ 8 bit bus support ดังรูปที่ 1.31

3. มีแคชขนาด 8 KB อยู่ในตัว 804866 เอง ทำให้มีการเพิ่มบิต CD และ NW ในรีจิสเตอร์ CRO และเพิ่มขาขึ้นเพื่อสนับสนุนแคช

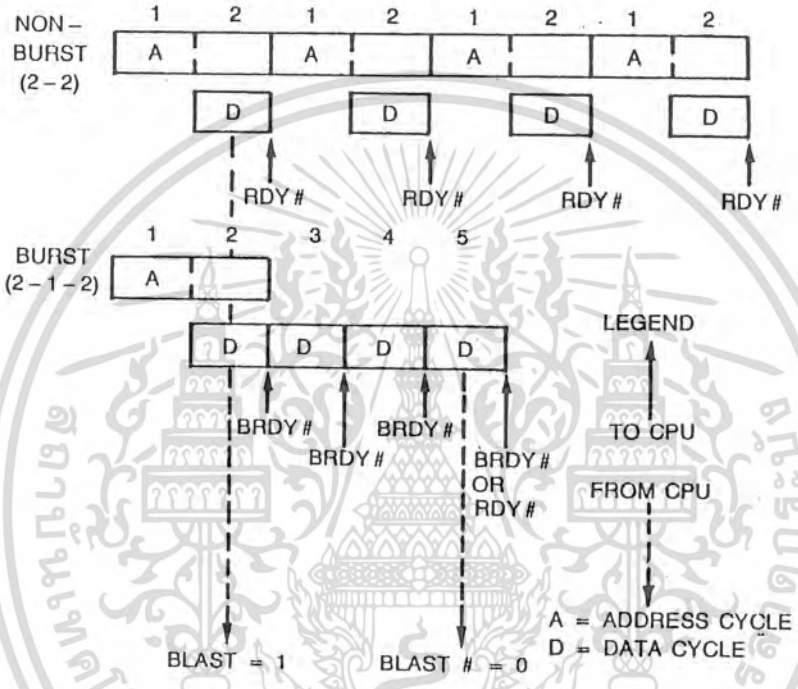
4. มี 80387 อยู่ในตัว ทำให้ไม่ต้องทำ I/O เมื่อใช้คำสั่งเกี่ยวกับโฟลตติ้งพอยน์ อินเตอร์รัพต์ 9 ไม่สามารถเกิดขึ้นได้อีก ขณะที่อินเตอร์รัพต์ 13 เกิดขึ้นแทน

5. สนับสนุนการเกิดการผิดพลาดของโฟลตติ้งพอยน์เพิ่มขึ้น เพื่อความเข้ากันได้กับของเก่า (DOS Compatibility) ซึ่งทำให้ต้องการบิต NE ในรีจิสเตอร์ CRO และขา (pin) เพิ่มขึ้น คือ FERR# และ IGNNE#

6. มีคำสั่งใหม่ 6 คำสั่ง คือ

- BSWAP (Byte SWAP)
- XADD (Exchange a/nd add)

- CMPXCHG (Compare and Exchange)
- INVD (Invalidate Data Cache)
- WBINVD (Write back nad Invalidate data Cache)
- IN_VLPG (Invalidate TLB Entry)



รูปที่ 1.31 BURST และ NON-BURST MODE

7. รีจิสเตอร์ CR3 มีบิตถูกนิยามเพิ่มขึ้น 2 บิต คือ PCD และ PWT

8. มีเพจโปรเทคชันเพิ่มขึ้นซึ่งต้องการบิต WP เพิ่มขึ้นใน CR0

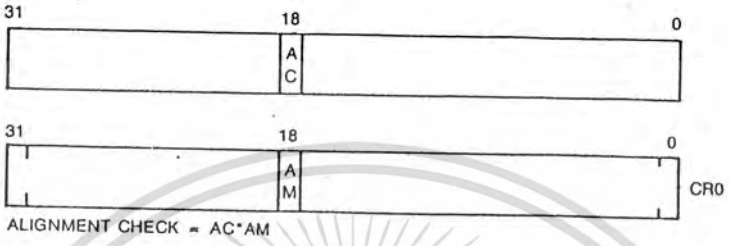
9. มี Alignment Check ทำให้เพิ่มบิต AC ใน EFLAG และบิต AM ในรีจิสเตอร์ CR0 รูปที่ 1.31 และ 1.32

10. เปลี่ยนวิธีการเกี่ยวกับ translation look aside buffer (TLB) โดยใช้วิธีการเช่นเดียวกับ on chip cache

11. มีรีจิสเตอร์ตรวจสอบเพิ่มขึ้นคือ TR3, TR4 และ TR5

12. prefetch queue เพิ่มจาก 16 ไบต์เป็น 32 ไบต์

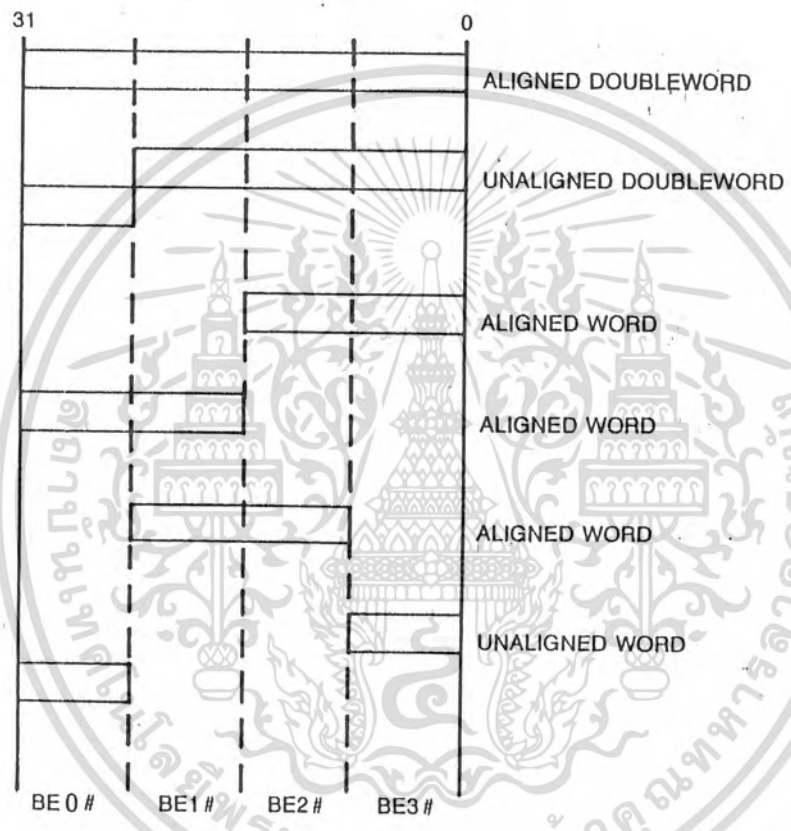
13. หลังจาก reset , ID ใน upper byte ของรีจิสเตอร์ DX มีค่าเท่ากับ 04



DATA TYPE ALIGNMENT REQUIREMENTS

Memory Access	Alignment (Byte Boundary)
Word	2
Dword	4
Single Precision Real	4
Double Precision Real	8
Extended Precision Real	8
Selector	2
48-Bit Segmented Pointer	4
32-Bit Flat Pointer	4
32-Bit Segmented Pointer	2
48-Bit "Pseudo-Descriptor"	4
FSTENV/FLDENV Save Area	4/2 (On Operand Size)
FSAVE/FRSTOR Save Area	4/2 (On Operand Size)
Bit String	4

รูปที่ 1.18 ค่า alignment



รูปที่ 1.33 การ alignment ของข้อมูล

บทที่ 2

สถาปัตยกรรมของระบบ EISA

บัสบนเครื่องคอมพิวเตอร์ดั้งเดิมคือ บัส พีซี และเอ็กซ์ที ที่เป็นที่รู้จักกันเป็นครั้งแรกบนเครื่องไอบีเอ็มพีซีในปี 1981 เมื่อเปรียบเทียบกันแล้วมันเป็นบัสแบบซิงโครนัสขนาด 8 บิตที่ธรรมดามาก ๆ ซึ่งมีการตรวจสอบพาร์ตี้ และมีอินเตอร์รัพต์แบบทริกด้วยขอบ นั่นหมายความว่าแต่ละเส้นของอินเตอร์รัพต์สามารถถูกใช้จากการ์ดอะแดปเตอร์เพียงการ์ดเดียวเท่านั้น (ตารางที่ 2.1) บัสในเครื่องพีซี ดั้งเดิมนั้น ไม่ได้เตรียมไว้สำหรับให้มีการควบคุมบัสจากภายนอก ถ้าไม่ใช้ซีพียูหลักก็ตัวควบคุมดีเอ็มเอทีจะเป็นตัวควบคุมบัสตลอดเวลา

ตารางที่ 2.1 แสดงสัญญาณของบัส พีซี และเอ็กซ์ที

AO-A19	สัญญาณแอดเดรส 20 เส้น จะใช้เพียง 10 เส้นล่างเมื่อเป็นไซเคิลของ I/O
DO-D7	สัญญาณข้อมูล 8 เส้นที่เป็นชนิด 2 ทิศทาง
IRQ2-IRQ7	สัญญาณขอใช้อินเตอร์รัพต์ 6 เส้น
DRQ1-DRO2, DACK1-DACK3	สัญญาณขอใช้ดีเอ็มเอและตอบรับการขอใช้ดีเอ็มเอแชนแนล 0 จะถูกใช้สำหรับการรีเฟรชไดนามิกแรม
IO CH RDY	สัญญาณนี้จะถูกใช้โดยการ์ดหน่วยความจำหรืออุปกรณ์ภายนอก เพื่อกำเนิดสัญญาณ wait state
IOR, IOW, SMEMR, SMEMW	สัญญาณสโตรปสำหรับการเขียน/อ่านหน่วยความจำหรือ I/O
OSC	สัญญาณนาฬิกา 14.31818 MHz ถูกใช้โดยการ์ดแสดงผลบางชนิด ซึ่งจะไม่ซิงโครไนซ์กับสัญญาณของบัส
CLK	สัญญาณนาฬิกาสำหรับบัส (4.77 HMz สำหรับ PC ดั้งเดิม)
AEN, TC	สัญญาณอีนาเบิ้ลสำหรับแอดเดรส สัญญาณนี้จะถูกใช้ระหว่างไซเคิลของดีเอ็มเอ
IO CH CHK	สัญญาณที่กระตุ้นโปรเซสเซอร์ผ่านทางนอนมาร์คอินเตอร์รัพต์ เมื่อมีความผิดพลาดเกิดขึ้น
RESET DRV	สัญญาณแสดงว่าระบบอยู่ระหว่างการรีเซต
Vcc&GND	เพาเวอร์ซัพพลาย

ตารางที่ 2.2 แสดงสัญญาณที่เพิ่มเข้ามาประกอบเป็นบัส ISA 16 บิต

D8-D15	สัญญาณข้อมูล 8 บิตที่เพิ่มขึ้น
SBHE	สัญญาณ byte high enable แสดงว่าจะใช้สัญญาณไบต์บน
IRQ10-IRQ12, IRQ14-IRQ15	สัญญาณขอใช้อิน ค- +@ - เพิ่มขึ้น IRQ13 จะถูกสำรองไว้ใช้สำหรับตัวประมวลผลทางคณิตศาสตร์
DRQO, DACKO, DRQ5-DRQ7, DACK5-DACK7	สัญญาณการขอใช้ดีเอ็มเอที่เพิ่มมากขึ้น
MEMR, MEMW	สัญญาณสโตรปสำหรับการเขียน/อ่านหน่วยความจำ สัญญาณนี้จะแอคตีฟตลอดเวลา ขณะที่ SMEMR SMEMW จะแอคตีฟเฉพาะไซเคิลที่อ้างอยู่ในแอดเดรสของซีพียูเท่านั้น
MASTER	สัญญาณใหม่ที่เพิ่มเข้ามาเพื่อให้การ์ดอะแดปเตอร์สามารถทำตัวเป็นตัวควบคุมบัสหลักได้
MEM CS16, IO CS16	สัญญาณที่การ์ดอะแดปเตอร์ส่งไปบอกเมนบอร์ดว่าสามารถทำการถ่ายข้อมูลแบบ 16 บิตได้

ประมาณในปี 1984 ไอบีเอ็ม ต้องการที่จะขยายบัสจาก 8 บิตไปเป็น 16 บิต โดยที่จะต้องคงความคอมแพตทิเบิลกับของเดิม ดังนั้นสัญญาณต่าง ๆ ที่อยู่ใน slot ขนาด 62 ขาก็ยังคงไว้เช่นเดิม และเพิ่ม slot ใหม่เข้าไปในส่วนด้านล่าง (ตารางที่ 6) แสดงสัญญาณส่วนที่เพิ่มเข้ามา ทำให้เกิดเป็นบัสแบบเอทีี

ไอบีเอ็มได้เพิ่มคุณลักษณะอื่น ๆ ที่จะทำให้บัสเอที คอมแพตทิเบิลกับของเดิม เนื่องจากเครื่องเอที ใช้ซีพียู 80286 ที่ทำงานเร็วกว่าเครื่องซีพีธรรมดาที่ใช้ซีพียู 8088 จึงได้มีการเพิ่มตัวกำเนิดสัญญาณ wait state เพื่อที่จะยืดไซเคิลของบัสให้ยาวออกไป เพื่อให้อุปกรณ์ที่มีความเร็วต่ำสามารถทำงานกับซีพียูที่มีความเร็วสูงได้ ดังนั้นขาของสัญญาณบางขาที่ไม่ถูกใช้ จึงถูกนำมาใช้ เช่น ขา B8 ถูกนำมาทำเป็นสัญญาณ 0WS (zero wait state)

คอนเน็คเตอร์แบบใหม่ซึ่งมีอยู่ 2 แถว ๆ ละ 16 ขานั้น ได้มีการเพิ่มสัญญาณแอดเดรสเข้าไปอีก 4 เส้น (LA20-LA23) รวมทั้งเพิ่ม (LA17-LA19) ซึ่งมีอยู่ใน slot แบบเดิม การเพิ่มสัญญาณทั้ง 3 เข้าไปนั้นซึ่งเป็นการซ้ำซ้อนนั้นเพราะว่าแอดเดรสของไอบีเอ็มพีซีดั้งเดิมนั้นเป็นแบบค้างค่าไว้ (latch) ซึ่งขั้นตอนในการค้างแอเดรสไว้นั้น ทำให้การติดต่อกับอุปกรณ์ภายนอก (peripheral broad) ช้าลง แอดเดรส 3 เส้นที่เพิ่มเข้ามานั้นเป็นแบบไม่ค้างค่าไว้ ดังนั้นบัสเอทีจึงยอมให้การ์ดแบบ 16 บิต ตรวจสอบในช่วงแรกของไซเคิลว่าเป็นการอ้าง

แอดเดรสหรือไม่ เพื่อว่าการ์ดนั้นจะได้ส่งสัญญาณเพื่อบอกว่ามันเป็นการ์ด 16 บิต เพราะฉะนั้นถ้าเป็นไปได้ให้ติดต่อแบบ 16 บิต

สัญญาณพิเศษเหล่านี้เช่น MEM CS16 และ I/O CS16 เป็นตัวสำคัญ ทำให้บัสเอทีคอมแพตทิเบิลกับของเดิม โดยถ้า 80286 พยายามที่จะกระทำการติดต่อแบบ 16 บิต และหนึ่งในสัญญาณทั้งสองสัญญาณนี้ไม่ถูกแทรกเข้าไปในไซเติลของการทำงาน จะมีฮาร์ดแวร์พิเศษตัวหนึ่งทำให้การติดต่อเป็นแบบ 8 บิต 2 ครั้งแทนที่ จะเป็นครั้งเดียว 16 บิต แต่เนื่องจากสัญญาณแอดเดรสที่ไม่ค้างค่าไว้มีเพียง 7 เส้นเท่านั้น คือ LA17-LA23 นั่นหมายความว่า การติดต่อที่ใช้บริเวณไม่มากกว่า 128 กิโลไบต์ขึ้นไป จะไม่ทำให้เกิดกรณีดังที่กล่าวมาแล้ว คือ การทำทีละ 16 บิต ดังนั้นการ์ดหน่วยความจำที่ใช้พื้นที่ไม่เต็ม 128 กิโลไบต์ (คือใช้สัญญาณไม่เกิน LA16) ก็จะไม่ทำให้เกิดสัญญาณที่บอกการ์ดว่าเป็นการกระทำแบบ 16 บิต ในทางปฏิบัติแล้วการ์ดจำพวก EMS และอุปกรณ์ภายนอกที่ถูกมองเป็นหน่วยความจำจะถูกให้ทำการถ่ายข้อมูลทีละ 8 บิตแทนที่จะเป็น 16 บิต

เช่นเดียวกับบัสเอที EISA ถูกสร้างขึ้นมาโดยอิงมาตราฐานเดิม และมีการเพิ่มสัญญาณแอดเดรส สัญญาณข้อมูล และสัญญาณควบคุมให้มากขึ้น แต่อย่างไรก็ตามจะต้องคำนึงถึงความคอมแพตทิเบิ้ลกับบัสแบบเดิมด้วย

แต่เดิมมาแล้ว เอกสารที่เปิดเผยอย่างเป็นทางการของไอบีเอ็มไม่เคยกำหนดช่วงเวลา (timing) ของสัญญาณในแบล็คเพลน (สัญญาณที่อยู่บนเมนบอร์ดและต่ออยู่กับสล้อต)ของเครื่องพีซี และเอทีเลย แต่ใน EISA ซึ่งเป็นส่วนขยายของบัสเอที ไม่อาจจะทำเช่นนั้นได้ มันจะต้องถูกกำหนดอย่างแน่นอน เพราะว่าไม่เพียงแต่ให้เป็นมาตรฐานใหม่เท่านั้น แต่ยังต้องคงความคอมแแพตทิเบิ้ลกับของเดิมด้วย ซึ่งในส่วนนี้ก็ไม่น่าจะมีปัญหาเลย ทั้งนี้เพราะว่าผู้ร่วมโครงการเพื่อทำบัส EISA ล้วนแล้วแต่เป็นบริษัทที่เคยออกแบบเครื่องที่เลียนแบบไอบีเอ็มมาแล้วทั้งสิ้น ซึ่งน่าจะเป็นกลุ่มบุคคลที่ดีที่สุดสำหรับงานนี้โดยเฉพาะ ตารางที่ 2.3 แสดงสัญญาณที่มีขึ้นมาใหม่ในบัส EISA ขาสัญญาณของ EISA จะถูกวางอยู่ระหว่างขาในรูปของบัส ISA

ตารางที่ 2.3 แสดงสัญญาณที่เพิ่มขึ้นประกอบกันเป็นบัส EISA

BE0-BE3	BYTE ENABBLE เป็นสัญญาณที่แสดงว่าไบต์ไหนของข้อมูล 32 บิต ที่เกี่ยวข้องอยู่ในไซเคิล
M//IO	สัญญาณที่แยกความแตกต่างระหว่างไซเคิลของหน่วยความจำกับไซเคิลของ I/O
START	สัญญาณที่แสดงการเริ่มต้นของบัสไซเคิลของ EISA
CMD	สัญญาณที่จัดการช่วงเวลาที่ควบคุมบัสไซเคิลของ EISA
MSBRUST	สัญญาณที่สดงว่าตัวหลักสามารถจะกระทำไซเคิลที่รวดเร็วได้

SLBRUST	สัญญาณที่แสดงว่าตัวรองสามารถจะรับไซเคิลที่รวดเร็วได้
EX32, EX16	สัญญาณที่แสดงว่าบัสรองในการ์ด EISA สามารถรองรับการกระทำแบบ 32 หรือ 16 บิตได้ ถ้าสัญญาณทั้ง2 ไม่เกิดขึ้นในช่วงแรกของไซเคิลก็แสดงว่าบัสจะกลับไปทำงานในโหมดขของ ISA แบบเดิม
EXRDY	สัญญาณที่สดงว่าบัสรองของ EISA พร้อมแล้วที่จะจบไซเคิล
MREQn	ถูกแทรกโดยบัสหลักตัวที่ n เพื่อขอใช้บัส
MAKn	สัญญาณเที่ถูกส่งจากบัสหลักตัวที่ n ว่ามันได้ใช้บัสแล้ว
D16-D31	สัญญาณข้อมูลที่เพิ่มขึ้นมา ทำให้บัส ISA เดิมกลายเป็นบัส EISA ขนาด 32 บิต
LA2-LA16, LA17-LA31	สัญญาณแอดเดรสที่เพิ่มขึ้นมาใหม่ เช่นเดียวกันกับ LA17-LA23 สัญญาณเหล่านี้จะไม่ค้างค่าไว้

ต่อไปจะมาดูสัญญาณของ EISA ที่เพิ่มไปจากบัสเดิมของพีซี ซึ่งจะมาดูคุณลักษณะพิเศษอันอื่นที่มีอยู่ จุดสำคัญของบัส EISA ก็คือไม่ว่าจะเป็นพีซียูหลักหรือบัสมาสเตอร์อื่นสามารถจะแอ็กเซสกับหน่วยความจำหรืออุปกรณ์อื่นๆ ในระบบได้ โดยไม่ต้องคำนึงถึงขนาดความกว้างของบัส

สมมติว่าการ์ดที่เป็นบัสมาสเตอร์ 32 บิต ของ EISA ต้องการจะเขียนข้อมูลลงในหน่วยความจำที่อยู่บนการ์ด 8 บิตแบบ ISA บัสมาสเตอร์จะเริ่มต้นด้วยการขอใช้บัส และเตรียมแอดเดรสและข้อมูล จากนั้นจะแทรกสัญญาณเริ่มต้น (start)เข้าไป และคอยตรวจสอบสัญญาณ ex32,ex16,mem cs16 และ Ows ขณะที่สัญญาณเหล่านี้ตอบกลับมมา ตัวควบคุมบัสก็จะสุ่มเอาสัญญาณเหล่านี้ไว้ด้วย

เมื่อบัสมาสเตอร์และตัวควบคุมบัส เห็นว่าไซเคิลนั้นไม่สามารถทำเป็นบบ 32 บิตได้ ตัวควบคุมบัสก็จะเข้าควบคุมการทำงานแทน โดยตรวจสอบค่าแอดเดรสและข้อมูลจากบัสมาสเตอร์ตัวควบคุมบัสก็จะเริ่มส่งสัญญาณเดียวกันนี้เข้าไปในบัสขณะที่บัสมาสเตอร์กำลังทำอยู่(จะไม่เกิดการชนกัน เพราะว่าทั้งคู่ส่งไปในสายของตัวเอง) ดังนั้นครึ่งไซเคิลต่อมา บัสมาสเตอร์จะหยุดการส่งข้อมูล ตัวควบคุมบัสซึ่งทำหน้าที่อยู่จะทำการถ่ายข้อมูลทีละไบต์ 4 ครั้ง เพื่อทำการส่งข้อมูลให้ครบ 32 บิต

ข้อดีที่ไม่ต้องเข้าไปยุ่งเกี่ยวเลยก็คือ อุปกรณ์ดีเอ็มเอ และบัสมาสเตอร์ไม่จำเป็นต้องมีลอจิกสำหรับการเลื่อนไบต์หรือกำเนิดไซเคิล เพื่อที่จะจัดการกับการถ่ายข้อมูลที่เป็นไปได้หลากหลายตัวอย่างเช่น ชิพที่ทำหน้าที่เป็นบัสมาสเตอร์ 32 บิต Bus Master Interfacr Conntroller (BMIC)สามารถจัดการถ่ายข้อมูลได้เพียงแบบ 32 บิต และอย่างรวดเร็ว 16 บิตเท่านั้นโดยไม่ต้องมีการช่วยเหลือ แต่คอมบิเนชันอื่นๆ ต้องจัดการโดยตัวควบคุมบัสแบบ EISA (EISA BUS CONTROL LER =EBC)

สถานะแปลกใหม่เกิดขึ้นเมื่อมาสเตอร์ขนาด 16 บิต อ้างแอดเดรสเวิล์ดบน (upper word)ของหน่วยความจำขนาด 32 บิต ตัว EBC จัดการสถานะแบบนี้โดยกอปปี้ข้อมูลจาก 2 ไบต์ล่างของบัสข้อมูลขึ้นไปยัง 2 ไบต์บน ดังนั้นข้อมูลก็จะมาถึงในตำแหน่งที่ถูกต้องขบวนการนี้เรียกว่า copy-up ซึ่งทำให้มาสเตอร์ 16 บิตสามารถเอ็กเซสสเลฟ 32 บิต โดยไม่ต้องมีไดรฟ์เวอร์สำหรับสัญญาณข้อมูลทั้ง 32 บิต

2.1 การถ่ายเทข้อมูลความเร็วสูง

จุดเด่นข้อหนึ่งที่ใช้ในการขายของ EISA ก็คือ ความเร็ว ในสภาวะที่ถูกต้องตามปกติ อุปกรณ์แบบ EISA สามารถถ่ายข้อมูลแแบบรวดเร็วได้ถึง 33 เมกะไบต์วินาที บนด์วิดธ์ที่กว้างขนาดนี้สามารถนำไปใช้งานกับระบบแลนความเร็วสูง ดิสค์ไดรฟ์ชนิดความสูง(เช่น IPI หรือ SCSI-2)และจอภาพความละเอียดสูงได้

แต่ไม่ใช่ว่าการติดต่อของบัสจะเป็นไปตามรูปแบบนี้เสมอไป ถ้าเราใช้การ์ด ISA มันก็จะขาดประสิทธิภาพที่โดดเด่นของ EISA ไปเมื่อเปรียบเทียบกับการใช้การ์ดแบบ EISA โดยเฉพาะ

2.2 การใช้อินเทอร์รัพต์ร่วมกัน

ปัญหาอันหนึ่งที่พบอยู่เสมอก็คือเมื่อเราขยายระบบขึ้นไปแล้ว สัญญาณอินเตอร์ไม่เพียงพอก่การใช้งานในระบบบัสแบบ ISA ไม่ยอมให้สัญญาณอินเตอร์รัพต์ร่วมกัน แต่ละสายของสัญญาณ อินเตอร์รัพต์เป็นชนิดทริกด้วยขอบ (edge trigger) และถูกขับด้วยระดับสัญญาณแบบ TTL ที่เป็นแบบ 3 สถานะ

อย่างไรก็ตามในระบบบัสแบบ EISA ได้จัดเตรียมสายสัญญาณอินเตอร์รัพต์ที่เป็นแบบสามารถตรวจสอบการอินเตอร์รัพต์โดยระดับลอจิก ซึ่งทำให้การ์ด EISA หลาย ๆ การ์ดสามารถใช้อินเตอร์รัพต์ร่วมกันได้ แต่อย่าลืมว่าการ์ดแบบ ISA ไม่สามารถใช้อินเตอร์รัพต์ร่วมกันได้ แม้ว่าการ์ดนั้นจะเสียบอยู่บนสล้อตแบบ EISA ก็ตาม

2.3 การใช้บัสจากแต่ละอุปกรณ์

ตัวควบคุมการรีเฟรชชหน่วยความจำชนแแนลดีเอ็มเอ ที่มีสิทธิ์พิเศษสูงและส่วนอื่น ๆ ที่พยายามที่จะใช้บัส จะต้องแข่งขันกันเพื่อครอบครองบัสโดยมีรูปแบบที่หมุนเวียนกันไป 3 ลักษณะ ซึ่งรูปแบบนี้รับประกันได้ว่าไม่มีบัสมาสเตอร์ใดที่จะไม่ได้ใช้บัสถึงแม้ว่าจะเป็นแชนแนลดีเอ็มเอที่มีสิทธิพิเศษต่ำและยังแน่ใจได้ว่าหน่วยความจำจะถูกรีเฟรชได้ถูกต้อง ยิ่งไปกว่านั้นชิพเซตของอินเทลยังมีไทเมอร์ที่เป็น wach dog ที่คอยควบคุมไม่ให้มีการใช้บัสนานเกินไป ถ้าถึงเวลาที่กำหนดล้วผู้ครอบครองบัสขณะนั้นจะถูกตัดออกจากการใช้บัส และจะเกิดนอนมาร์ค อินเตอร์รัพต์ส่งไปให้ซีพียูหลัก

2.4 คอนเน็คเตอร์ของบัสEISA

คอนเน็คเตอร์ของบัส EISA ถูกออกแบบให้คอมแพตทิเบิ้ลกับการ์ดแบบ ISA อย่าง 100 เปอร์เซ็นต์ หน้าสัมผัสบนขอบการ์ดจะถูกจัดเป็น 2 แถว แถวบนจะเป็นส่วนที่เป็นสัญญาณของ ISA ทั้งหมด ส่วนแถวล่างจะเป็นส่วนของสัญญาณ EISA ที่เพิ่มขึ้นมาใหม่

ในคอนเน็คเตอร์แบบ EISA จะมีแง่พลาสติกที่กันไม่ให้การ์ดแบบ ISA เสียบลงไปลึกเกินไปที่จะสัมผัสกับส่วนที่เป็นสัญญาณของ EISA แต่ถ้าเป็นการ์ด EISA เองจะสามารถเลื่อนลงไปได้ง่ายจนเข้าไปในรอยบาก โลหะที่เป็นส่วนหน้าสัมผัสของ EISA ที่มีขนาดเล็กกว่าแบบ ISA และขาของสัญญาณของ EISA จะวางอยู่ระหว่างขาสัญญาณแบบ ISA เดิม

2.5 ชิพเซตของอินเทล

ในปัจจุบันนั้นส่วนใหญ่จะใช้ชิพเซตของบริษัทอินเทล ซึ่งประกอบด้วยชิพ 3 ชิพ คือ Intergrated Syster Peripheral (ISP)และอีก 2 ตัวที่ได้กล่าวถึงไปแล้วคือ BMICและ EBC

EBC เป็นตัวกำเนิดไซเคิลบนบัส EISA และจัดการบัฟเฟอร์ที่เชื่อมต่อกับโลคอลบัสของซีพียู EBC ยังเป็นตัวจัดการสัญญาณรีเซตสำหรับซีพียูและตัวควบคุมหน่วยความจำชนิดแแคช (cache controller) ด้วย EBC เป็นตัวที่ทำงานควบคู่กับ ISP เสมอ (ต้องทำงานร่วมกัน)ISP เป็นตัวจัด Interrupt controller,DMA controller,ounter และtimer ให้กับซีพียู และยังจัดการรีเฟรชหน่วยความจำนอกจากนี้ ISP ยังมีโหมดการทำงานแบบความเร็วต่ำ (slow down)เพื่อลดความเร็วของซีพียู ซึ่งมีความสำคัญในกรณีที่ใช้ซอฟต์แวร์ที่มีการโปรเทค

เพราะว่าชิพเซต 2 ตัวนี้รวมฟังก์ชันที่สำคัญของบัส EISA ไว้มาก ผู้ผลิตที่ใช้ชิพของ อินเทลก็ไม่ชอบที่จะใช้ชิพเซตของบริษัทค่ายที่สามเช่นบริษัทChip& Technologies ในเครื่องเดียวกัน ดังนั้นบริษัทเหล่านี้ก็จะต้องพยายามที่จะพัฒนาชิพเซตของตัวเอง เพื่อเข้ามาแข่งขันในตลาดของเครื่องแบบ EISA นี้

ส่วนประกอบสุดท้ายของชิพเซตของอินเทลก็คือ BMIC มันถูกทำขึ้นโดยตั้งใจว่าจะใช้เป็น bus master interface บน การ์ดที่เรียกว่า Intelligent cards และยังมี local bus interface ซึ่งปรับปรุงโดยเฉพาะเพื่อใช้กับ 80186 โดยซีพียูหลักสามารถติดต่อกับซีพียูบนการ์ดได้โดยผ่านรีจิสเตอร์ที่เรียกว่า MAILBOX และ DOORBELL

การ์ด EISA ทุกการ์ดจะมีหมายเลขประจำเพื่อบอกโดยเฉพาะ ซึ่งจะแอ็กเซสได้ที่แอนเดรสที่กำหนดไว้ล่วงหน้าโดยสล้อตหมายเลขประจำการ์ด EISA เป็นตัวยกความแตกต่างของการ์ดเช่น เดียวกันกับ

หมายเลย ID ของการ์ดไมโครแชนแนล แต่สิ่งหนึ่งที่แตกต่างกันก็คือ หมายเลขการ์ดของ EISA ไม่ได้ถูกจัดการจากส่วนกลาง อย่างเช่นไอบีเอ็ม

โครงสร้างของระบบ EISA

ระบบ EISA ประกอบด้วย 5 ส่วนหลัก คือ

- ระบบย่อยของ ซีพียู
- ส่วนเชื่อมต่อ ISP และ EBC
- ส่วนบัฟเฟอร์ข้อมูลและ แอดเดรส
- ตัวควบคุม DRAM
- X บัสเพอริเฟอรัล

ส่วนต่างๆเหล่านี้เชื่อมต่อกันด้วยบัสเล็ก 4 บัส คือ

- Local CPU bus เป็นบัสความเร็วสูงระหว่าง CPU และหน่วยความจำแคช
- Host differentiation bus คือเส้นทางของข้อมูลและแอดเดรสขนาด 32 บิด ระหว่าง CPU กับหน่วยความจำหลัก หรือเชื่อมต่อกับอุปกรณ์เฉพาะพิเศษอื่นๆ เช่น ถังควบคุมแลน (LAN) 82596
- EISA/ISA bus เป็นส่วนขยายของระบบบัสแบบเดิม (ISA) ให้มีขนาดเป็น 32 บิด
- Peripheral X-bus เป็นบัสของความเร็วต่ำกว่าสำหรับอุปกรณ์ประกอบที่ไม่ต้องการเชื่อมต่อความเร็วสูงของบัส EISA

2.6 ระบบย่อยของซีพียู

ในระบบย่อยนี้ประกอบด้วย ซีพียู หน่วยประมวลผลร่วมทางคณิตศาสตร์ และตัวควบคุมแคชทำงานที่ความถี่ของซีพียู

ตัวควบคุมแคชจัดการกับหน่วยความจำแคชทำให้ซีพียูทำงานได้โดยไม่มีสภาวะรอคอยประมาณ 90% ของเวลาทั้งหมด (hit rate 90%) ซึ่งการจัดการมีได้หลายวิธีเช่น direct map ,two-way set associative

หน่วยประมวลผลรวมทางคณิตศาสตร์ เป็นอุปกรณ์เพิ่มเติมเพื่อเพิ่มประสิทธิภาพในการคำนวน

กิจกรรมทั้งหมดในระบบนี้ ถูกควบคุและกำหนดค่าเริ่มต้นโดยซีพียู และสามารถทำงานไปพร้อม ๆ กับบัสอื่นๆ ในระบบได้ เช่น ซีพียูใช้งานแคช ขณะที่อุปกรณ์ DMA ใช้งาน DRAM

2.7 ส่วนเชื่อมต่อ ISP และ EBC

EBC (EISA Bus Controller) เป็นอุปกรณ์หลักในระบบ EISA ทำหน้าที่แปลงวงรอบการทำงานแบบต่างๆ คือ วงรอบของซีพียู, วงรอบ ISA และวงรอบ EISA ให้ทำงานเข้ากันได้ อุปกรณ์แม่บนแต่ละบัสสามารถติดต่อกับอุปกรณ์ลูกบนบัสใดๆก็ได้ EBC จะทำการแบ่งแยกวงรอบถ้ามีการถ่ายโอนข้อมูลระหว่างบัสที่มีขนาดต่างกัน หรือการจัดเรียงแอดเดรส บัฟเฟอร์ และสัญญาณควบคุมบัสทั้ง ISA และ EISA ถ้าใช้งานที่ความเร็วสูงมากการเชื่อมต่อระหว่าง EBC กับ CPU ต้องมีการใช้ PAL เพื่อจัดจังหวะเวลาใหม่สำหรับสัญญาณที่อ่อนไหวกับความเร็ว

ISP (Integrated System Peripheral) ประกอบด้วยส่วนการทำงานต่างๆ ที่จำเป็นสำหรับระบบ EISA และเข้ากันได้กับระบบ ISA ด้วย ซึ่งรวมถึงดีเอ็มเอ 32 บิต,ตัวควบคุมอินเตอร์รีพลี 2 ถัง, วงจรนับ 5 ชุด, ตัวจัดสรรบัส, ส่วนกำเนิดแอดเดรสสำหรับการรีเฟรช DRAM, และวงจรเบ็ดเตล็ดอื่นๆ ISP ต้องทำงานคู่กับ EBC และยังให้ EBC กำเนิดสัญญาณที่จำเป็นในการทำ DMA

2.8 ส่วนบัฟเฟอร์ข้อมูลและแอดเดรส

ส่วนนี้จัดให้เกิดการเชื่อมต่อระหว่างบัส EISA/ISA กับ host differentiationbus ลอจิกการบัฟเฟอร์ข้อมูลทำการประกอบไบต์ที่จัดเรียงหรือขนาดไม่เท่ากันในการรับส่ง ซึ่งข้อมูลจะถูกเก็บอยู่ชั่วขณะหนึ่งให้มีการทำงานต่างๆ ได้

แอดเดรสบัฟเฟอร์กำเนิดแอดเดรส LA และ SA ที่ถูกต้องเมื่อซีพียูเป็นผู้ควบคุมการแปลงแอดเดรสเกิดขึ้นในทิศทางอื่นของตัวควบคุมในบัส EISA/ISA

2.9 ตัวควบคุม DRAM

การออกแบบตัวควบคุม DRAM ให้ใช้ความสามารถของ EISA ได้เต็มที่นั้นจะมีพื้นฐานมาจากระบบที่ใช้แรม SIMM (single inline memory module) ชนิด 1 Mbit x 9 (หรืออาจเป็น 250 kbit x 9 ก็ได้) ความเร็ว 80 ns ที่มีการจัดแบบ tast page mode

2.10 X บัส เพอริเฟอรัล

บัสเป็นบัสรองซึ่งรับสัญญาณจากบัส ISA เพื่อเชื่อมต่อกับอุปกรณ์ประกอบอื่นๆ (peripherd devices) ที่ไม่จำเป็นต้องใช้การเชื่อมต่อความเร็วสูง อุปกรณ์เหล่านี้ได้แก่ ตัวควบคุมคีย์บอร์ด 8742, ตัวควบคุม

ฟลอบปี้ดิสต์, RTC, SRAM สำหรับการติดตั้ง, รีจัสเตอร์สำหรับติดตั้งของ EISA, BIOS ของระบบ และส่วนติดต่อกับลูกของ ISP นอกจากนี้อาจเป็นอุปกรณ์อื่นๆ เช่น บอร์ดขนาน บอร์ดอนุกรม ตัวควบคุม VGA หรือโมเด็ม

บทที่ 3
ทฤษฎีการออกแบบแผ่นวงจรพิมพ์

3.1 แผ่นวงจรพิมพ์

แผ่นวงจรพิมพ์เป็นอุปกรณ์ที่ใช้ในการสร้างวงจรใช้งานทางอิเลคทรอนิกส์ต่าง ๆ โดยจะมีสารนำไฟฟ้าพิมพ์ หรือฉาบเป็นลายวงจรอยู่บนแผ่นฉนวนเคลือบ (Dielectric Substrate) แผ่นฉนวนเคลือบที่มีสารนำไฟฟ้าเดินเป็นลายวงจรแล้วนี้เรียกว่า แผ่นวงจรพิมพ์การสร้างวงจรทำได้ โดยนำอุปกรณ์อิเลคทรอนิกส์มาประกอบเข้ากับแผ่นวงจรพิมพ์ที่ถูกต้องก็จะได้วงจรตามต้องการแผ่นวงจรพิมพ์แบ่งออกได้ 3 แบบ คือ

3.2.1 Single sided boards คือ แผ่นวงจรพิมพ์ที่มีลวดลายวงจรของสารตัวนำไฟฟ้าอยู่เพียงด้านเดียว อาจมีการเจาะรูเพื่อใช้ยึดหรือเสียบอุปกรณ์อิเลคทรอนิกส์หรือไม่มีก็ได้

3.2.2 Double sided boards คือ แผ่นวงจรพิมพ์ที่เดินลายวงจรเอาไว้ ทั้งสองด้าน ลายวงจรของทั้งสองด้านที่เชื่อมต่อถึงกันนั้นจะเชื่อมต่อกันโดยการเจาะรูแล้วฉาบสารนำไฟฟ้าเชื่อมวงจรทั้งสองด้านผ่านรูที่เจาะนั้น

3.2.3 Multilayer boards คือ แผ่นวงจรพิมพ์ที่มีแผ่นฉนวนเคลือบตั้งแต่สองแผ่นขึ้นไปเรียงซ้อน ๆ กันเป็นชั้น ๆ แต่ละชั้นจะมีการเดินลายวงจรเอาไว้แล้วทุกแผ่นและจุดที่เชื่อมต่อระหว่างวงจรของแต่ละชั้นก็ทำการเชื่อมต่อ โดยการเจาะรูทะลุถึงกันในตำแหน่งที่เหมาะสม แล้วฉาบสารตัวนำไฟฟ้าเชื่อมต่อกันระหว่างวงจร รูที่เจาะไว้เหล่านั้น

3.2 วัสดุที่ใช้ทำแผ่นฉนวนเคลือบ

แผ่นฉนวนเคลือบส่วนใหญ่จะผลิตมาจากแผ่นไฟเบอร์กลาส (Fiberglass) ซึ่งฉาบทองแดงด้วย อิพอกซีเรซิน (Epoxy Resin) เรียกว่า แผ่นฉนวนเคลือบแบบอิพอกซีไฟเบอร์กลาส (Epoxy/Fiberglass) นอกจากนี้วัสดุที่นำมาใช้ทำแผ่นฉนวนเคลือบอาจเป็นกลาสกับโพลีไมด์ (Polymide) , เทฟลอน หรือ ไทร์อาซิน เรซิน (Triazine Rasin) หรือเป็นกระดาษเคลือบด้วยฟีนอลิกเรซิน ทั้งหมดนี้จะมีแผ่นทองแดงบาง ๆ ฉาบอยู่ทั้งสองด้านด้วยเรซิน

ปัจจุบันอุตสาหกรรมการผลิตทางด้านอิเลคทรอนิกส์ได้พัฒนาก้าวหน้าไปอย่างรวดเร็ว จากวิวัฒนาการทางด้านอุปกรณ์ที่ใช้สารกึ่งตัวนำเข้ามาแทนได้แก่ ไดโอดทรานซิสเตอร์ ชนิดต่าง ๆ จนกลายเป็นวงจรรวมบรรจุอยู่ในที่เดียวกันและมีขนาดเล็กมาก แต่มีขีดความสามารถทำงานได้มากกว่า และดีกว่าโดยการ

นำเอาวงจรรวมนี้มาประกอบกับอุปกรณ์ขึ้นเป็นวงจรโดยที่ใช้พื้นที่เพียงเล็กน้อยได้อย่างสะดวกรวดเร็ว สามารถทำรวมกันได้โดยใช้แผ่นวงจรพิมพ์ วัสดุที่ใช้ทำแผ่นวงจรพิมพ์ทำจากแผ่นทองแดงบาง ๆ ปะติดอยู่กับสารที่เป็นฉนวนที่ เป็นแผ่นบาง ๆ ฉนวนที่ใช้อาจจะเป็นกระดาษ ฟินอลิก หรือ อีปอกซีกลาส แล้วนำมาผ่านกระบวนการสร้างลายวงจรบนแผ่นซึ่งมีทั้งชนิดหน้าเดียว สองหน้า จนถึงแบบหลายเลเยอร์ หรือหลายชั้น ซึ่งกำลังเป็นที่นิยมใช้กันในวงจรเครื่องคอมพิวเตอร์ในปัจจุบันนี้

3.3 SMT (Surface Mount Technology)

คอมโพเนนท์แแบบ SMT ทุกตัวจะยึดติดกับบอร์ด โดยการปะติดกับทองแดงบนบอร์ดแทนที่จะต้องเจาะรูทะลุบอร์ด แล้วนำขาคอมโพเนนท์เสียบเข้าไปในรูแล้วเชื่อมติดบอร์ดด้วยตะกั่วที่หยอดเข้าไปในรู ซึ่ง SMT มีแนวโน้มที่จะนำมาใช้อย่างแพร่หลายในอนาคต

ในเทคโนโลยีด้านแผ่นวงจรพิมพ์ประกอบด้วยปัจจัยที่ต้องคำนึงถึง คือ ขนาดของบอร์ด ต้นทุนในการผลิตและคุณภาพผลิตภัณฑ์ ซึ่งถ้าใช้ SMT ในการออกแบบและการผลิตแผ่นวงจรพิมพ์จะได้ผลดีกว่าการใช้ THT (Through-hole Technology) ในทุก ๆ ด้าน

3.3.1 ขนาด

จากการเปรียบเทียบการผลิตแแผ่นวงจรพิมพ์ โดยใช้ SMT กับ THT พบว่าการใช้ SMT จะสามารถผลิตที่มีขนาดเล็กกว่าถึง 30-50% ทั้งนี้เป็นผลมาจากที่คอมโพเนนท์แบบ SMT (SMCs) มีขนาดเล็กกว่า ข้อดีของ SMT อีกประการหนึ่งคือ ไม่มีการเจาะรูทะลุบอร์ดทำให้สามารถวางอุปกรณ์ได้ทั้งสองด้านของบอร์ด ข้อได้เปรียบที่ตามมาคือ เวีย (via) ของแผ่นวงจรพิมพ์แบบ SMT จะเล็กกว่าเวียของแผ่นวงจรพิมพ์แบบ THT เส้นผ่าศูนย์กลางของรูที่เจาะบนแผ่นวงจรพิมพ์แบบ SMT จะมีขนาดเล็กดังตารางที่ 3.1 แสดงการเปรียบเทียบขนาดของ SMCs และ THCs (Through-hole Components)

3.3.2 การเชื่อมต่อสัญญาณระหว่างคอมโพเนนท์ (Interconnectivity)

ขนาดของคอมโพเนนท์มีส่วนสำคัญอย่างมากต่อInterconnectivity ซึ่งInterconnectivity เป็นสิ่งสำคัญอย่างมากที่ต้องคำนึงถึงเป็นอย่างมาก เมื่อไอซีบนบอร์ดมีจำนวนมาก ขาไอซีที่เป็น SMCs จะมีขนาดเล็ก และมีอยู่ทั้งสี่ด้านของตัว ไอซี ทำให้ ไอซีขนาดเล็กมีจำนวนขาหนาแน่นมากเป็นการเพิ่ม Interconnectivity ของคอมโพเนนท์ Innterconnectivity สามารถเปรียบเทียบกันได้ โดยแสดงหน่วยในรูปของ Per Unit Area บนแผ่นวงจรพิมพ์ ดังรูปที่ 3.2 แสดง Interconnectivity ของไอซีแบบต่าง ๆ

3.3.3 Performance

ขาของ SMCs สั้นทำให้ลด Parasitic Inductance และ Parasitic Capacitance ซึ่งจะมีค่าสูงในวงจรดิจิตอลและวงจรอนาล็อกที่มีความถี่สูง เหตุผลนี้จึงทำให้วงจรดิจิตอลและวงจรอนาล็อก ที่มีความถี่สูง ๆ นิยมใช้ SMT จะสามารถทำงานได้ที่ความถี่สูงกว่า 3 GHZ ได้ นอกจากนี้ยังเป็นการช่วยลดเวลาหน่วง ซึ่งเวลาหน่วงจะเป็นอุปสรรคสำคัญ ในวงจรที่มีการทำงานแบบซิงโครนัส เช่น ในวงจรดิจิตอล ถ้าสายสัญญาณคล็อก มีความถี่ยิ่งสูงจะยิ่งถูกลดทอนมากทำให้เวลาหน่วงมากจะทำให้สัญญาณคล็อกที่ส่งไปถึงคอมโพเนนท์แต่ละตัวไม่พร้อมกัน เป็นเหตุให้วงจรทำงานผิดพลาด SMT จะช่วยให้วงจรดิจิตอลทำงานได้ที่ความถี่ที่มี คล็อกสูงกว่า 10 MHZ ได้

ข้อดีอื่น ๆ ก็คือสามารถเพิ่มการป้องกันการกระแทก การสั่นสะเทือน และการรบกวนของสนามแม่เหล็กไฟฟ้าของ SMT Assemlies ได้ เพราะแผ่นวงจรพิมพ์และ SMCs มีขนาดเล็กและน้ำหนักเบา

3.4 ขั้นตอนในการผลิต แผ่นวงจรพิมพ์

3.4.1 การออกแบบวงจรที่ต้องการนำมาทำแผ่นวงจรพิมพ์ การออกแบบต้องใช้ความรู้ทางด้านไฟฟ้าในการออกแบบวงจร

3.4.2 สร้างแผนผังของวงจรที่ต้องการทำแผ่นวงจรพิมพ์ โดยในการสร้างแผนผังควรใช้สัญลักษณ์ที่เป็นมาตรฐานตามแบบสากล และง่ายต่อการเข้าใจ

3.4.3 ออกแบบลายวงจรของวงจรที่จะติดอยู่บนแผ่นวงจรพิมพ์ โดยสเกลที่ใช้จะต้องถูกต้องตามความจริง ในปัจจุบันการทำแผนผังของวงจรและการออกแบบลายวงจรสามารถนำคอมพิวเตอร์มาช่วยในการทำงาน

3.4.4 นำลายวงจรที่ออกแบบมาทำอาร์ทเวิร์คบนแผ่นฟิล์ม โดยส่วนที่เป็นลายวงจรจะเป็นสีดำพื้นเป็นสีใส

3.4.5 จะทำการเจาะรูบนแผ่นที่จะทำวงจรพิมพ์ในตำแหน่งที่ต้องทำการเจาะ จากนั้นจะทำการเคลือบทองแดงเข้าในผนังรอบ ๆ รูที่เจาะทุกรู ขั้นตอนนี้เราเรียกว่า Plating Through Hole จึงมีประโยชน์ในการทำการเชื่อมเส้นลายวงจรที่อยู่คนละชั้น ในการทำแผ่นวงจรพิมพ์ที่มากกว่าหนึ่งด้านขึ้นไป

การผลิตในอุตสาหกรรมการเจาะจะใช้ Numerical Control Drilling Machine ซึ่งทำการเจาะแบบอัตโนมัติ และเลือกขนาดของรูที่เจาะได้

3.4.5.1 ทำ Silk screening คือ การ screen ลายวงจรให้ติดไปบนแผ่นวัสดุที่ใช้ทำแผ่นวงจรพิมพ์ เพื่อให้ได้แผ่นวงจรพิมพ์จำนวนมากและเหมือนกัน

3.4.5.2 ทำการกัดทองแดงในส่วนที่ไม่ต้องการออกด้วยสารละลาย เช่น Sulfuric acid hyfrogen peroxide ให้ได้แผ่นวงจรพิมพ์ตามต้องการ

3.4.5.3 นำแผ่นวงจรที่ได้มาตรวจสอบข้อผิดพลาด

3.4.5.4 แก้ไขข้อผิดพลาด

ในการทำงานของวงจรบนแผ่นวงจรพิมพ์ มักพบปัญหาเล็ก ๆ น้อย ๆ ที่เกิดจากแผ่นวงจรพิมพ์อยู่บ่อยครั้ง เช่น มีการทำงานผิดพลาด มีสัญญาณรบกวนเกิดขึ้นเองเป็นบางครั้งบางคราว โดยหาสาเหตุไม่พบ ทำให้เสียเวลา แรงงาน ความคิดและไม่มีประสิทธิภาพเท่าที่ควร ปัญหาที่เกิดจะพบครั้งแรกบนวงจรต้นแบบเท่านั้น เพื่อไม่ให้เกิดปัญหาดังกล่าวนี้ ควรมีการออกแบบแผ่นวงจรพิมพ์อย่างถูกต้อง

3.5 วิธีการวางอุปกรณ์ลงบนแผ่นวงจรพิมพ์

3.5.1 อุปกรณ์ที่ทำงานร่วนกันควรวางให้อยู่รวมกัน เป็นกลุ่มเดียวกันบนแผ่นวงจรพิมพ์ เช่น วงจรหนึ่งประกอบด้วยหน่วยความจำ 8 ตัว ในการวางอุปกรณ์หน่วยความจำทั้ง 8 ตัว ลงบนแผ่นวงจรพิมพ์ควรวางติดเป็นกลุ่มเดียวกัน ดังรูป 3.7

3.5.2 อุปกรณ์ที่มีสารสัญญาณเชื่อมต่อถึงกัน ควรวางอยู่ใกล้กันในการวางอุปกรณ์ลงบนแผ่นวงจรพิมพ์ เพื่อให้การเดินลายเส้นของสัญญาณจะสามารถเดินได้สั้นที่สุด

3.5.3 ในการวางอุปกรณ์ลงบนแผ่นวงจรพิมพ์ ควรวางโดยใช้เนื้อที่น้อยที่สุด เพื่อประหยัดวัตถุดิบในการผลิตและสามารถเดินลายเส้นของวงจรได้สั้นที่สุด

3.5.4 อุปกรณ์ทุกตัวที่วางอยู่บนแผ่นวงจรพิมพ์ ควรวางอยู่ในแนวเดียวกัน เช่น วางอยู่ในแนวดิ่ง หรือในแนวราบ

3.6 การสร้างลายเส้นวงจร

3.6.1 เส้นวงจรแต่ละเส้นนั้นควรจะมีระยะห่างช่วงละเท่า ๆ กัน และเพื่อไม่ให้เกิดความหนาแน่นของลายเส้นวงงจรบางบริเวณมากเกินไป

3.6.2 ในการเดินลายเส้นวงจรที่สำคัญควรเดินให้สั้นที่สุด เพื่อทำให้เกิดการรบกวนและการลดทอนของสัญญาณน้อยที่สุด เช่น สัญญาณคล็อกในวงจรดิจิตอล

3.6.3 ลายเส้นวงจรที่ปีนข้ามกันนั้นต้องน้อยที่สุด เพื่อลดระยะทางของลายเส้นจากการเดินวกวนเสียพื้นที่น้อยลงและลดปัญหาของลายเส้นอื่นด้วย ถ้าใช้แผ่นวงจรพิมพ์ชนิดสองหน้าขึ้นไปใช้การทำ PTH (Plated Through Holes) แทนการโยงสาย

- ในกรณีที่ลายเส้นวงจรต้องหักมุมเลี้ยวเบนนั้น ไม่ควรจะหักเป็นมุมฉากหรือเป็นรูปตัว v เพราะจะมีปัญหาทางด้านการใช้สารละลายเกลือ หรือกรดกัดทองแดงส่วนที่ไม่ต้องการออกไม่หมด และยังจะทำให้เกิดปัญหาหลุดออกจากปลายแผ่นวงจรพิมพ์ได้ง่าย เช่น ในกรณีที่ได้รับความร้อนจากหลายหัวแร้งบัดกรี หรือในกรณีที่มีกระแสไหลผ่านตัว v มาก วิธีที่ดีในการเลี้ยวเบนลายวงจรนั้นควรจะค่อย ๆ โค้งให้ได้ระยะห่างระหว่างลายเส้นเท่า ๆ กัน หรือ จะทำให้เป็นลายเส้นวงจรหักเป็นช่วง ๆ ดังรูป 3.8
- ขนาดของลายเส้นวงจรนั้นควรจะพิจารณาตามปริมาณของ กระแสไฟฟ้าที่ไหลผ่านลายเส้นวงจรที่มีขนาดเล็กมากนั้น ย่อมจะมีค่าความต้านทานมากกว่าลายเส้นวงจรที่มีขนาดใหญ่ และระยะความยาวของลายเส้นวงจรที่เพิ่มขึ้นจะทำให้มีค่าความต้านทานมากขึ้นด้วย ซึ่งจะมีผลทำให้แรงดันของสัญญาณไฟฟ้าขณะที่ถูกโหลดนั้นสูญเสียไปกับลายเส้นวงจร ลายเส้นวงจรที่มีปริมาณกระแสไหลผ่านมากควรมีขนาดใหญ่ ลายเส้นวงจรที่มีปริมาณกระแสไหลผ่านน้อยควรมีขนาดเล็ก เช่น เส้นลายของ Power ควรมีขนาดใหญ่
- ในกรณีแผ่นวงจรพิมพ์สองหน้า แนวของลายวงจรที่อยู่ในแต่ละหน้าควรอยู่ในแนวที่ตั้งฉากกัน เพื่อหลีกเลี่ยงปัญหาการเกิด ค่า C Coupling ในวงจรที่มีความถี่เข้ามาเกี่ยวข้อง
- กราวด์ (Ground) ในวงจรควรเป็นจุดเดียวเพื่อลดปัญหา Stary Inductance ในวงจรดิจิตอลสามารถเดินกราวด์ เป็นแถบที่กว้างและยาวเพื่อลดปัญหานี้

บทที่ 4
ซอฟต์แวร์และทฤษฎีในการออกแบบ

ซอฟต์แวร์ EDA ที่ใช้ในการออกแบบแผ่นวงจรพิมพ์นั้น ได้ใช้ซอฟต์แวร์ Mentor Graphic ซึ่งเป็นซอฟต์แวร์ที่สมบูรณ์พอสมควร โดยจะแบ่งออกเป็นโปรแกรมย่อย ๆ อีกหลายส่วน เพื่อใช้งานในแต่ละส่วนของการออกแบบ

4.1 ขั้นตอนการออกแบบ

สำหรับการออกแบบแผ่นวงจรพิมพ์ของ Mentor Graphic นั้นจะแบ่งขั้นตอนออกเป็น 9 ขั้นตอนหลัก ดังนี้

1. Entry schermatic

   เป็นการออกแบบและป้อนวงจร (schmatic) ที่ต้องการ

2. Create symbol library

   สร้างสัญลักษณ์ (symbol) ที่ต้องการสร้างเพิ่มเติม ตามปกติ Mentor Graphic จะมี library มาตรฐานของสัญลักษณ์มาให้ แต่ในกรณีที่ไม่มีสัญลักษณ์ที่ต้องการก็ต้องสร้างขึ้นมาใหม่เอง เช่น 80486DX , 82357 , 82358 ฯลฯ

3. Creating PCB Parts for the Design

   เป็นการกำหนด ฟิสิคอล geometry ของคอมโพเนนท์ เช่น กำหนด thru-pin , via padstacks , graphics layers รวมถึงการสร้างเมนบอร์ดด้วย

4. Preparing the Packageing Supports Data

   เป็นการกำหนดข้อมูล Mapping file เพื่อเก็บข้อมูลว่า symbol instances ที่บรรจุใน components package และ Catalog file เก็บคอมโพเนนท์ที่ใช้ในการออกแบบ PCB

5. Assigning Symbols to Ccomponentc (Packaging)

   เป็นการกำหนด symbol บน schematic เป็นฟิสิคอลคอมโพเนนท์

6. Placing Components on the Board

   เป็นการวางคอมโพเนนท์ต่าง ๆ ลงบนเมนบอร์ด

7. Routing the Components Pin Connecctionms

เป็นการ route สายต่าง ๆ ของวงจรโดยใช้ซอฟต์แวร์ Layout

8. Generating the Manufacturing Data

สร้างรายงานต่าง ๆ ที่จำเป็นต่อการส่งไปทำ Fabrication

9. Updating the Schematics

ปรับปรุงแก้ไขวงจรเดิม

4.2 รายละเอียดของ Mentor Graphic

เมื่อทราบรายละเอียดของขั้นตอนการออกแบบแล้ว ต่อไปจะเป็นการอธิบายถึงความสามารถและการใช้งานของโปรแกรมแต่ละส่วนของ Mentor Graphic

4.2.1. Design Architecture (DA)

โปรแกรมนี้เป็นส่วนแรกที่ต้องใช้ในการออกแบบ โดยใช้สำหรับวาดสัญลักษณ์ทางไฟฟ้ามาประกอบกันเป็นวงจรตามที่ได้ออกแบบไว้ โดยสามารถเรียกใช้อุปกรณ์ที่อยู่ในไลบรารีมาตรฐานของ DA ซึ่งประกอบด้วยอุปกรณ์มาตรฐานที่ใช้กันอยู่ทั่วไป เช่น ไอซีอนุกรมของ 74LS 74ALS , CMOSs และอุปกรณ์อนาลอกอื่น ๆ ออก แล้วลากเส้นเชื่อมต่อกันขึ้นเป็นวงจรที่ได้ออกแบบไว้

ส่วนประกอบหลักของ DA

ไลบรารีมาตรฐานต่าง ๆ ของอุปกรณ์

พิมพ์วงจรเพื่อใช้ในการอ้างอิงตรวจสอบหรือแก้ไข ซึ่งสามารถกำหนดขนาดได้

การลากสายไฟฟ้า

การตรวจสอบสัญญาณทางไฟฟ้า

การตรวจสอบวงจร ซึ่งเป็นการตรวจสอบว่ามีข้อผิดพลาดเกิดขึ้นหรือไม่ในการป้อนวงจรเข้าเครื่อง สามารถแบ่งวงจรขนาดใหญ่ออกเป็นชีทย่อย ๆ ได้ (Muti sheet)

นอกจากนี้ในการสร้างวงจรนั้น อาจต้องใช้อุปกรณ์พิเศษที่ไม่มีในไลบรารี ก็สามารถสร้างขึ้นมาใช้เองได้ เช่น 80486DX เป็นอุปกรณ์ที่ไม่มีไลบรารีก็สามารถสร้างขึ้นมาใหม่ได้

ส่วนประกอบของอุปกรณ์

อุปกรณ์ที่สร้างขึ้นนั้นจะประกอบด้วย 4 ส่วนใหญ่ ๆ คือ

- รูปร่างของตัวถัง (Body shape) เป็นส่วนที่ปรากฏให้เห็นในวงจร
- ขาอุปกรณ์ (Pin) เป็นจุดสำหรับการเชื่อมต่อทางไฟฟ้า
- จุดกำเนิด (Origin Point) เป็นจุดอ้างอิงสำหรับการวางตัว อุปกรณ์
- คุณสมบัติ (Properties) เป็นข้อความที่ใช้อ้างอิงถึงตัวอุปกรณ์นั้น ๆ

4.2.2 LIBRARIAN

ขั้นตอนนับตั้งแต่นี้ไปจะเป็นการทำในส่วนของบอร์ดสเตชั่นซึ่งบอร์ดสเตชั่นเป็นเซตของโปรแกรมที่ใช้ในการออกแบบแผ่นวงจรพิมพ์และเอาท์พุทจากบอร์ดสเตชั่นก็เป็นอินฟอร์เมชั่นและข้อมูลที่ใช้ในการสร้างแผงวงจรจริง ๆ

เซตของโปรแกรมในส่วนบอร์ดสเตชั่นประกอบด้วย LIBARIAN , PACKAGE , LAYOUT , FABLINK

LIBRARIAN เป็นโปรแกรมแรกในส่วนบอร์ดสเตชั่นโดยสามารถสร้างส่วน parts ที่คุณออกแบบ ซึ่งรวมถึงการสร้าง บอร์ด , จิโอเมตรี (geometry) , แพด และเวีย ต่อไปจะเป็นการกล่าวถึงส่วนประกอบสำคัญที่ต้องศึกษาก่อนใช้โปรแกรม LIBRARIAN

Library Parts Data จะประกอบไปด้วย 3 ส่วนคือ

- Electrical data

  เป็นข้อมูลทางไฟฟ้าที่ใช้ในการออกแบบที่มีผลต่อวงจร เช่น เวลาหน่วงของพินของแนนเกต

- พินและสัญลักษณ์ข้อมูล (Symbol Data)

  พินและสัญลักษณ์ข้อมูลใช้ในการอธิบายว่าฟังก์ชันทางไฟฟ้าอยู่ที่ตำแหน่งใดใน physical package

- Mechanical Data

  ใช้ในการอธิบายขนาดและรูปร่างของ physical package

Parts Data Flow จะกล่าวถึงขั้นตอนในการสร้างพาร์ท

- พินและสัญลักษณ์ข้อมูลจะต้องถูกสร้างเป็นไฟล์แมปปิ้งซึ่งไฟล์เหล่านี้จะประกอบไปด้วยอินฟอร์เมชั่นซึ่งอธิบายความสัมพันธ์ของสัญลักษณ์บน Schematic

- mechanical data จะอยู่ในส่วนไลบรารีพาร์ทซึ่งส่วนนี้จะเก็บในไฟล์จีโอเมตรีซึ่งไฟล์นี้จะอธิบายถึง ขนาด รูปร่าง ตำแหน่งของพินและข้อมูลการวาดส่วนอื่น
- ไฟล์คาตาล็อกจะอธิบายทุกส่วนของหมายเลขพาร์ทในไลบรารี ซึ่งแต่ละหมายเลขพาร์ท จะมีความสัมพันธ์กับรูปแบบจีโอเมตรี และไฟล์แมปปิ้ง
- PACKAGE จะใช้ไฟล์แคตาล็อกในการหาอินฟอร์เมชั่น ซึ่งจะอธิบายว่าสัญลักษณ์บน Schematic กำหนดบน physical package อย่างไร
- ไฟล์จีโอเมตรีและไฟล์แคตาล็อกจะถูกโปรเซสใน PACKAGE ซึ่งมันจะผลิตข้อมูลสำหรับใช้ในโปรแกรม LAYOUT

คุณสมบัติของสัญลักษณ์

- ต้องมีคุณสมบัติแสดงตัวของสัญลักษณ์ตัวอย่างเช่นคุณสมบัติ Ref เท่ากับ U10
- คุณสมบัติเท็กซ์ ซึ่งมีคุณสมบัติดังนี้ visibility , orientation , height , justification
- สำหรับ PCB layout แล้วสัญลักษณ์จะต้องมีพินและคุณสมบัติคอมพ์ (comp)

จีโอเมตรี

- สามารถวาดวงกลม เส้นโค้ง รูปหลายเหลี่ยม หรือเท็กซ์ เพื่อสร้างบอร์ดคอมโพเนนท์ และ แพดสแตก หรือจะวาดฟอร์มแมต โลโก และ จีโอเมตรีอื่น ๆ ได้
- สามารถสร้างเลเยอร์ใช้งานได้ถึง 256 เลเยอร์
- สามารถที่จะสร้าง วิว และแก้ไขได้
- หน่วยที่ใช้มี cm , mm , mils และนิ้ว ซึ่งมีความละเอียดถึง .00005 นิ้ว

แอตตริบิวส์ของจีโอเมตรี

แอตตริบิวต์เป็นลักษณะที่ต้องกำหนด สำหรับใช้ในการ placement และ routing

- แอตตริบิวต์มีความสัมพันธ์กับรูปแบบ geometry
- บางแอตตริบิวต์จำเป็นต้องมี และบางแอตตริบิวต์เป็นออปชั่นให้เลือกใส่เลเยอร์

Mentor Graphics สามารถให้ออกแบบ PCB ให้มีเลเยอร์ถึง 256 เลเยอร์แต่ในทางปฏิบัติก็ไม่ได้ใช้ถึงขนาดนั้น

- เลเยอร์ทั้ง 256 เลเยอร์สามารถตั้งชื่ออื่นได้
- เลเยอร์จะประกอบด้วยลักษณะดังนี้ สี, ชื่อ, หมายเลขชั้น, transparency, fill

pattern และ line style

- บางเลเยอร์ระบบจะเป็นตัวกำหนดการใช้งานหรือสามารถที่จะกำหนดการใช้งานเองก็ได้

พฤติกรรมของเลเยอร์ (Layer Behavior)

พฤติกรรมของเลเยอร์ถูกควบคุมโดยการ display characteristics และ mapping characteristics

- สี, visbility, line style, fill pattern สามารถกำหนดได้ในแต่ละเลเยอร์

คอมโพเนนท์จีโอเมตรี

Component Geometries ถูกสร้างทุก ๆ คอมโพเนนท์

- ประกอบไปด้วย placement outline, pins, silkscreen outline และ reference designator
- มีรูปแบบมาตรฐานของคอมโพเนนท์จีโอเมตรี คือ through pin และ surface mount
- จุดเริ่มต้นของ component geometry จะกำหนดที่ตำแหน่งใด ๆ ก็ได้
- หมายเลขพินสามารถเป็น alpha-numeric , non-sequential ได้

แมปปิ้งไฟล์

mapping files จะอธิบายถึงความสัมพันธ์ระหว่าง logical และ physical pin โดย mapping file จะกำหนด

- ความสัมพันธ์ของพิน
- จำนวนของเกตต่อแพกเกจ
- การสแวปของเกต พิน และ พินเซต
- พินปกติ

แคตาล็อกไฟล์

แคตาล็อกไฟล์จะเก็บรายละเอียดของคอมโพเนนท์พาร์ทดังนี้

- จำนวนเกตต่อแพกเกจ
- ชื่อของคอมโพเนนท์จีโอเมตรี
- คุณสมบัติคอมพ์ของสัญลักษณ์

- ชื่อของแมปปิ้งไฟล์
- คุณสมบัติอื่น ๆ

บอร์ดจีโอเมตรี (Board Geometries)

บอร์ดจีโอเมตรีสร้างได้ใน LIBRARIAN หรือ สร้างใน Package Station (3D และ Design) แต่ไม่สามารถสร้างได้ใน LAYOUT

- บอร์ดจีโอเมตรีจะประกอบไปด้วยกฎการออกแบบในรูปของแอตตริบิวต์ ซึ่งสามารถแก้ไขเปลี่ยนแปลงได้ใน LAYOUT
- บอร์ดจีโอเมตรีประกอบด้วย routing และ placement keepout ด้วยซึ่งอยู่ในรูปของแอตตริบิวต์
- จุดกำเนิดของบอร์ดมีผลต่อการแสดงของกริดต่าง ๆ และ datum point

เมื่อทราบรายละเอียดต่าง ๆ ที่มีความสำคัญในการใช้โปรแกรม LIBRARIAN แล้ว ก็สามารถที่จะสร้าง พิน เวีย คอมโพเนนท์จีโอเมตรี บอร์ดได้ ตามที่ออกแบบไว้ในส่วนของ Schematic

4.2.3 PACKAGE

การทำงานของโปรแกรม PACKAGE จะเป็นการสร้างอินฟอร์เมชั่นเพื่อใช้ในส่วนของโปรแกรม LAYOUT โดยก่อนทำ PACKAGE จะต้องมีส่วนต่าง ๆ ที่สร้างไว้แล้วเพื่อจะนำมาใช้งานดังนี้

- จีโอเมตรีพาร์ทไฟล์สร้างจากส่วน LIBRARIAN ซึ่งประกอบด้วยส่วนแพดสแตก
- คอมโพเนนท์ บอร์ด
- พาร์ทแมปปิ้งไฟล์และแคตาล็อกไฟล์ซึ่งจะประกอบด้วยข้อมูลเกี่ยวกับการทำ Package ของตัวสัญลักษณ์ทางลอจิก
- Package Configuration File จะเป็นส่วนไฟล์ ASCII ซึ่งถูกสร้างโดยอัตโนมัติในส่วนของ LIBRARIAN
- Design File (peb-design.erel) จะประกอบด้วยข้อมูลส่วนเชื่อมต่อ ซึ่งถูกสร้างจากการ Expand PCB

เมื่อมีส่วนต่าง ๆ ที่กล่าวมาแล้ว ก็สามารถทำการ build process เพื่อสร้าง parts fileเพื่อใช้ในโปรแกรม LAYOUT ได้เลย

พาร์ทไฟล์ (Parts File)

พาร์ทไฟล์จะประกอบไปด้วยจีโอเมตรีทั้งหมด ที่ใช้ในการออกแบบ

- พาร์ทไฟล์จะเป็นอินพุทในโปรแกรม LATOUT และ Fabink
- มันจะต้องประกอบไปด้วยเพียงหนึ่งบอร์ดจีโอเมตรี
- จีโอเมตรีอื่น เช่น drill symbol, mechanical part geometry, panel, artwork stackups, drawing borders และ test coupon จะต้องเก็บในพาร์ทไฟล์
- พาร์ทไฟล์จะเก็บใน binary form แต่คุณสามารถที่จะสร้างเป็น ASCIIได้ ตามที่ต้องการ

4.2.4 LAYOUT

ขั้นตอนนี้จะเป็นการนำคอมโพเนนท์ที่ใช้ในวงจรมาวางบนบอร์ดที่ได้สร้างไว้ในตำแหน่งต่างๆ และทำการลากสายสัญญาณตามที่ออกแบบ

การทำงานของโปรแกรม LAYOUT จะแบ่งออกเป็น 2 ส่วนใหญ่ ๆ คือ

- การวางคอมโพเนนท์ (Placement)
- การลากสายสัญญาณ (Rouing)

การวางคอมโพเนนท์

การออกแบบ PCB ที่ดีส่วนใหญ่ก็ขึ้นอยู่กับการวางคอมโพเนนท์ที่ดีด้วย การวางคอมโพเนนท์ที่ดีจะทำให้ง่ายต่อการลากสายสัญญาณ

การวางคอมโพเนนท์ที่ดีควรจะมีลักษณะดังนี้

- ต้องคำนวนว่าขนาดของบอร์ดต้องเล็กที่สุดเท่าที่เป็นไปได้
- ต้องใช้จำนวนเลเยอร์ในการลากสายสัญญาณน้อยที่สุด
- สายสัญญาณที่ลากควรจะต้องสั้นที่สุดเท่าที่จะเป็นไปได้
- ใช้เวียให้น้อยที่สุด
- ผลของมันก็คือจะทำให้ง่ายและเร็วในการลากสายสัญญาณ

เทคนิคการวางคอมโพเนนท์

- อย่างแรกควรวางคอมโพเนนท์ที่อยู่ในตำแหน่งเฉพาะ เช่น connector
- ต่อไปควรวางคอมโพเนนท์ให้อยู่เป็นกลุ่ม ๆ ด้วยกัน

- ต่อไปควรวางคอมโพเนนท์ซึ่งต้องสัมพันธ์กับคอมโพเนนท์ที่วางอยู่ก่อนแล้ว
- ต่อไปวางคอมโพเนนท์ที่เหลือให้หมดยกเว้นดิสครีสคอมโพเนนท์
- สุดท้ายปรับปรุงแก้ไขการวางให้ดีขึ้น

ฮิสโตแกรม

LAYOUT มีเครื่องมือในการคำนวนฮิสโตแกรมของการวางคอมโพเนนท์เพื่อให้เราสามารถวางคอมโพเนนท์ให้ดีที่สุด และสามารถลากสายสัญญาณได้ดี คอมโพเนนท์แต่ละส่วนจะไม่แน่นไปในจุด ๆ เดียว

- ฮิสโตแกรมที่แสดงเป็น 100 เปอร์เซ็นต์ทั้งทางด้านแกนตั้งและแกนนอน
- การวางที่ดีควรจะให้ฮิสโตแกรมแต่ละแกนไม่เกิน 40 เปอร์เซ็นต์

เมื่อเราสร้างอินฟอร์เมชั่นทุกอย่างตามขั้นตอนก่อนหน้านี้ เราจะมีบอร์ดและจีโอเมตรีของคอมโพเนนท์ทุกตัวจากนั้นต้องแมปคอมโพเนนท์ทุกตัวที่ใช้ในการออกแบบลงบนบอร์ดโดยจะต้องใช้โปรแกรม LAYOUT เรียกบอร์ดขึ้นมาก่อนแล้วจึงแมปคอมโพเนนท์ทุกตัวลงข้างบอร์ด

ในการวางคอมโพเนนท์ลงบอร์ดโปรแกรม LAYOUT มีฟีเจอร์ให้เลือก 2 ฟีเจอร์คือให้

- Manual Placement
- Auto Placement

Manual Placement

จะเป็นการวางคอมโพเนนท์ลงบอร์ดโดยให้ผู้ออกแบบเลือกตำแหน่งที่จะวางลงบนบอร์ดเอง ซึ่งโปรแกรม LAYOUT มีฟังก์ชันในการวางคอมโพเนนท์อยู่ เหตุที่ต้องเลือกคอมโพเนนท์ที่ออกแบบมาวางเองเพราะว่าการออกแบบบางครั้งจะต้องมีการวางกลุ่มชนิดของคอมโพเนนท์ให้อยู่เป็นกลุ่มม ๆ เช่น RAM และคอมโพเนนท์บางชนิด ต้องกำหนดที่ค่อนข้างตายตัวสำหรับการวางคอมโพเนนท์ เช่น คอนเนคเตอร์การวางคอมโพเนนท์แบบ Manual Placement นั้นจะใช้ในช่วงแรก ๆ เพื่อจัดตำแหน่งของคอมโพเนนท์ที่สำคัญให้ถูกที่แล้วต่อไปจะเป็นการให้เครื่องทำการวางโดยอัตโนมัติซึ่งเรียกว่า Auto Placement

Auto Placement

เป็นการวางคอมโพเนนท์ลงบนบอร์ดโดยให้เครื่องทำการคำนวนตำแหน่งที่จะวางเอง การวางคอมโพเนนท์แบบ Auto Placement จะใช้ในกรณีที่ทำการวางคอมโพเนนท์ตัวอื่นในตำแหน่งที่ต้องการแล้วเหลือคอมโพเนนท์ที่ไม่จำเป็นต้องกำหนดตำแหน่งแน่นอนบนบอร์ด เช่น ตัวต้านทาน ตัวเก็บประจุ และทรานซิสเตอร์ต่าง ๆ การวางคอมโพเนนท์แบบนี้จะไม่เสียเวลามากนัก

ความสามารถของโปรแกรม LAYOUT ในการวางคอมโพเนนท์แบบ Auto Placement อีกคือ

- กำหนดให้วางเป็นกลุ่มของคอมโพเนนท์ได้ เช่น ให้ Auto Placement เฉพาะ RAM ก่อนได้
- สามารถกำหนด priority ของคอมโพเนนท์ในการวางได้

การจัดปรับตำแหน่งการวางคอมโพเนนท์ (Improving Placement)

เมื่อวางคอมโพเนนท์ทุกตัวลงบนบอร์ดแล้ว ยังสามารถที่จะจัดปรับตำแหน่งของคอมโพเนนท์ได้อีกเพื่อให้การวางมีประสิทธิภาพที่สุดโดยมีอัลกอริธึมในการวางคอมโพเนนท์อยู่ 4 อย่างคือ

- คอมโพเนนท์
- เกต
- พินเซต
- พิน

เราสามารถที่จะเลือกการปรับการวางคอมโพเนนท์ได้ โดยเลือกจากเมนูของโปรแกรม LAYOUT ว่าจะปรับตำแหน่งของคอมโพเนนท์ โดยโปรแกรม LAYOUT จะตรวจสอบ identical component geometry และ สแวป ถ้าผลที่ได้สามารถที่จะลดความยาวของสายสัญญาณได้

สามารถปรับตำแหน่งของเกตโดยโปรแกรม LAYOUT จะตรวจสอบและจะสแวปภายในคอมโพเนนท์ ถ้าผลของมันสามารถที่จะลดความยาวของสายสัญญาณได้ (สแวปเกต)

สามารถปรับตำแหน่งของพินและพินเซตโดยโปรแกรม LAYOUT จะตรวจสอบและจะสแวปพินภายในคอมโพเนนท์ ถ้าผลของมันสามารถที่จะลดความยาวของสายสัญญาณได้

การลากสายสัญญาณ (Routing)

เมื่อวางคอมโพเนนท์ทุกตัวลงบนบอร์ดที่ออกแบบแล้ว ขั้นตอนต่อไปจะเป็นการลากสายสัญญาณ หรือที่เรียกกันว่าการลากสายสัญญาณ

Routing Design Rules

ก่อนที่จะทำการ route จะต้องทำความเข้าใจเกี่ยวกับกฎของการ route เสียก่อน design rules จะทำการตรวจสอบ

- Wire Width: เป็นเซตของความหนาของสายสัญญาณ

- Track Clearance: เป็นเซตของ clearance ที่น้อยที่สุดระหว่างสายสัญญาณกับสายสัญญาณ และเวีย และสายสัญญาณ และ keepout area

- Pad Clearance: เป็นเซตของ clearance ที่น้อยที่สุดระหว่างแพดกับแพด แพดกับเวีย และแพดกับสายสัญญาณ

- Via Name: เซตของ default via

- Allow Tjunctions: ข้อกำหนดพิเศษของ Tjunction ในการ trace

- Allow diagonals: ข้อกำหนดพิเศษของ diagonal route ที่กำหนดไว้

Guide Wires

ในการ route เราสามารถดูสายสัญญาณที่ยังไม่ทำการ route ได้โดยสิ่งที่แสดงจะเป็น Guide Wires ต่อระหว่างพินแต่ละคอมโพเนนท์ที่เชื่อมกันอยู่ เพื่อให้เราสามารถดูเป็นแนวทางในการ route

การ route

ในการ route โปรแกรม LAYOUT มีฟิเจอร์ให้เลือก 2 ฟิเจอร์เหมือนการวางโดยมีดังนี้

- Automatic Routing

- Menual Routing

Automatic Routing

Automatic Routing จะเป็นการลากสายสัญญาณโดยเครื่องจะทำการคำนวนและเลือกเส้นทางเดินของสายสัญญาณที่เหมาะสมมให้

ลักษณะพื้นฐานของการ Automatic Router

- Maze-Runner: แต่ละคอนเนกชั่นในการ ลากสายสัญญาณจาก "source" ไปยัง "target" การลากสายสัญญาณจะลากสายสัญญาณตาม gride point ที่เซตเอาไว้

- Cost-Driven: การลากสายสัญญาณจะตัดสินใจบนพื้นฐานของราคาเป็นหลัก การลากสายสัญญาณ ที่สมบูรณ์ราคาต้องเป็นศูนย์ ซึ่งก็คือการลากสายสัญญาณจาก "source" ไปยัง "target" ต้องไม่มี เวีย bend หรือ intersection

Routing cost Factors:

- Via

- EXcess

- Way
- Diagonal
- Bend
- Site
- Intersection

- Iterative และ Evolutionary: การลากสายสัญญาณ จะใช้ multiple pass ในการแก้ปัญหาการลากสายสัญญาณ
- Rip-Up และ Re-Route: จะใช้ Rip-Up และ Re-route หลังจากแต่ละ pass แล้วยังไม่พอใจ คอนเนกชั่นก็สามารถถูก rip และ reroute
- Squeeze-Through และ Shove-Aside: จะมีอัลกอริธึมพิเศษอยู่ในการ "see and fix"

Manual Routing

manual routing จะเป็นการลากสายสัญญาณโดยกำหนดเส้นทางเดินของสายสัญญาณเอง จะใช้ในกรณีที่ต้องการกำหนด เส้นทางสายสัญญาณเองเช่นสายสัญญาณคล็อก หรืออาจใช้ในกรณีที่ต้องการปรับปรุง สายสัญญาณที่ได้จากการ Automatic Routing แล้วสิ่งที่ได้ยังไม่เป็นที่พอใจ

เมื่อใช้โปรแกรม LAYOUT ทำการลากสายสัญญาณ ไม่ว่าจะเป็นการลากสายสัญญาณโดยการใช้วิธี Automatic Routing หรือ Manual Routing หรือจะใช้ทั้งสองอย่างผสมกัน ผลลัพธ์การลากสายสัญญาณที่สมบูรณ์จะต้องได้เปอร์เซ็นต์ของการลากสายสัญญาณเท่ากับ 100 เปอร์เซ็นต์

4.2.5 Fablink

ส่วนนี้จะเป็นการทำงานในส่วนสุดท้าย โดยจะเป็นการสร้างเอาท์พุทอินฟอร์เมชั่นสำหรับส่งไปทำ fabrication โดยก่อนที่จะทำ Fablink จะต้องมี

- บอร์ดได้ทำการวางคอมโพเนนท์หมดทุกตัว
- บอร์ดได้ทำการลากสายสัญญาณ สายสัญญาณทุกสายสัญญาณ
- บอร์ดจะต้องเก็บไฟล์ใน design directory ดังนี้
- comp_file เก็บ component placement information
- wire_file เก็บ wiring information

เอาท์พุท จาก Fablink

- Artwork Data: เป็นข้อมูลที่ใช้กับ photoplotter โดยปกติจะเป็นฟอร์แมต Gerber ซึ่งเป็นฟอร์แมตมาตรฐาน เมื่อสร้าง artwork data ขึ้นมาแล้วจะใช้ในขบวนการ photolithographicprocess เพื่อนำไปทำ fabricateบอร์ด PCBส่วนใหญ่ที่ทำ manufacture จะใช้ photolithographic process

- Drill Data: ข้อมูลต่อไปสำหรับทำ fabricate คือ drill data ซึ่งข้อมูลส่วนนี้จะนำไปควบคุมเครื่องเจาะในการเจาะรู thru-pin และ via hole

- Milling Data: ข้อมูลส่วนนี้จะนำไปให้ milling machine เพื่อตัดวัสดุ และรูปร่างของบอร์ดตามที่ได้ออกแบบไว้

- Drawing Data: ข้อมูลส่วนนี้จะแสดง fabrication drawing และ assembly drawing โดยการพล็อทใส่กระดาษออกมา

- reports: ข้อมูลส่วนนี้จะรายงานของการออกแบบทั้งหมดโดยจะประกอบไปด้วย

- Bill of Materials: จะเป็นการลิสคอมโพเนนท์ทั้งหมดที่ออกแบบ BOM จะใช้ในการสั่งซื้อคอมโพเนนท์เพื่อมาประกอบบนบอร์ด

- Net Conection List: จะให้ตำแหน่งของพินแต่ละคอมโพเนนท์บนบอร์ด

- Net Lenghts Report: จะแสดง trace length ของแต่ละ net อินฟอร์เมชั่นในรายงานนี้จะถูกใช้ในการ resimulation ของการออกแบบ

- Neutral File: จะประกอบไปด้วย

- ตำแหน่ง, orientation แต่ละคอมโพเนนท์

- ตำแหน่งและขนาดของเพด

เมื่อผลิตเอาท์พุททั้งหมดโดยโปรแกรมFablink ตอนนี้ก็สามารถนำข้อมูลทั้งหมดส่งไปทำ Fabricate เพื่อผลิตแผ่นวงจรพิมพ์ออกมาซึ่งเป็นการสิ้นสุดขั้นตอนการออกแบบ

บทที่ 5
ขั้นตอนการทำงาน

5.1 วงจรที่ใช้ในการออกแบบแผ่นวงจรพิมพ์

ในการออกแบบแผ่นวงจรพิมพ์นี้วงจรที่ใช้ในการออกแบบคือวงจรของเมนบอร์ด 80486DX ที่ใช้สถาปัตยกรรมระบบบัส 32 bit แบบ EISA

5.2 ส่วนการสร้าง Schematics

ในการสร้างส่วน Schematics ของวงจรเมนบอร์ด 80486DX นี้ได้ใช้โปรแกรม ที่มีชื่อว่า DA ของ Mentor Graphics ซึ่งมีวิธีการลำดับขั้นตอนดังนี้ คือ

5.2.1 เรียกใช้โปรแกรมและทำการเซทชีท Schematics ในส่วนของขนาดชีทที่จะใช้เขียนวงจร ขนาดของเท็กซ์ ตัวคอมโพเนนท์ของสกรีนกริด (Screen Grid)

5.2.2 เรียกใช้ตัวคอมโพเนนท์และทำการวางบนชีท ในการเรียกตัวคอมโพเนนท์ขึ้น จะเรียกมาจากส่วนของคอมโพเนนท์ ไลบรารี ซึ่งจะมี ไลบรารี มาตรฐานให้อยู่แล้วหลายชนิด เช่น Als_Lib , Ls_Lib อย่างในส่วนของ Ls_Lib จะบรรจุตัว Ls พาร์ทต่าง ๆ เช่น $74LS15 , $74LS52 เป็นต้น (สำหรับตัว คอมโพเนนท์ ซึ่งจะได้กล่าวไว้ในหัวข้อต่อไป) เมื่อได้ตัวคอมโพเนนท์ที่ต้องการแล้วก็จะทำการวางตัวคอมโพเนนท์นั้นลงบนชีทตามตำแหน่งที่ต้องการ ในการวางนี้อาจจะใช้คำสั่งก็อปปี้เรียกใช้ได้ด้วย

5.2.3 ทำการเชื่อมขาของคอมโพเนนท์ตามวงจรที่ต้องการซึ่งมีตั้งแต่การเชื่อมต่อขาโดยตรงระหว่างขาต่อขา การเชื่อมโดยผ่านรอยต่อ (Junction) การเชื่อมโดยใช้บัสซึ่งในการใช้บัสเชื่อมต่อนั้น จะต้องมีการกำหนดหมายเลขขาของบัสด้วยเพื่อเป็นการอ้างอิงได้อย่างถูกต้อง

5.2.4 การแก้ไขออปเจ็กต์ต่าง ๆ บนชีท ในการแก้ไขนี้มีวิธีต่าง ๆ ที่สามารถเรียกใช้ได้ เช่น การเคลื่อนย้าย ก็อปปี้ หมุน พลิก (Flip) หรือ การลบ ซึ่งขั้นตอนในการแก้ไขออปเจ็กต์นี้มีด้วยกัน 3 ขั้นตอน คือ

1. เลือกออปเจ็กต์ที่จะแก้ไข
2. ทำการแก้ไขด้วยวิธีการต่าง ๆ เช่น เคลื่อนย้าย ลบ เป็นต้น

3. ยกเลิกออปเจ็กต์ที่แก้ไข

สำหรับในการเลือกออปเจ็กต์สามารถทำได้ทั้งแบบเลือกตัวเดียว(Single Oject) และแบบเลือกหลายตัวพร้อมกัน (Multiple Oject)

5.2.5 การเปลี่ยนแปลงแก้ไขชื่อของสายที่เชื่อมต่อในวงจร ซึ่งในส่วนนี้จะใช้ในการ Simulation วงจร และเพื่อให้ง่ายในการอ้างอิงในส่วนของบัส

5.2.6 การวิวงจรซึ่งมีทั้งการ ขยายออก สำหรับไว้ใช้ในการแก้ไขการวิวดูเฉพาะที่ เป็นต้น

5.2.7 ทำการพิมพ์ชีท เพื่อใช้ในการอ้างอิงตรวจสอบ หรือแก้ไขซึ่งสามารถกำหนดสเกล กำหนดชื่อดีไวซ์ ที่ใช้ว่าเป็นเครื่องพิมพ์ , พล็อตเตอร์ หรือ ดีไวซ์ชนิดอื่น ๆ ได้ สามารถหมุน ให้พล็อทได้ รวมทั้งมีวิธีการอื่น ๆ ในการพิมพ์ชีท อีกมากมาย

5.2.8 ทำการ Check วงจร Schematic ในการทำแผ่นวงจรพิมพ์ขั้นตอนของการทำ Schematic นั้นจำเป็นที่จะต้อง Check ทั้งหมดให้ผ่านเสียก่อนจึงจะทำต่อไปได้โดยในการ Check นี้จะมีทั้งที่เป็น Error และ Warning โดยจะต้องทำการแก้ไข จนกระทั่งไม่เกิด Error เลยจึงจะผ่านขั้นตอนนี้ไปได้

ในการ Check ชีทนี้จะมีฟังก์ชันการทำงานดังนี้ เช่น

1. ค่าของ Property or References ผิดปกติ
2. มีขา คอมโพเนนท์ ที่ยังไม่ได้ต่อ
3. มีเส้นที่ต่อลอยอยู่ ไม่ได้เชื่อมกับเส้นอื่น
4. มีชื่อที่แตกต่างกันในสายเส้นเดียวกัน
5. ผิดกฎของการเชื่อมต่อแบบบัส
6. จำนวนขาของสัญลักษณ์ผิด
7. ชื่อของสายที่ต่อผิด
8. สายบัสกับจำนวนขาไม่เท่ากัน
9. Block ใน Diagram ผิดพลาด

ซึ่งเมื่อทำการ Check แล้วไม่ว่าจะเกิด Error หรือ Warning จะมีข้อความ (Messages) ออกมาเตือนหรือบอกให้รู้ว่าเป็น Error หรือ Warning ในเรื่องใดที่ตำแหน่งใด แล้วจึงไปทำการแก้ไขต่อไป

5.2.9 ทำการเซฟ Schematic แล้วออกจากโปรแกรม DA ซึ่งถือว่าเป็นการเสร็จในขั้นตอนของการสร้างวงจร Schematic

สำหรับในการทำSchematic ของวงจรเมนบอร์ด80486DX นี้ เนื่องจากวงจรมีขนาดใหญ่จึงแบ่งวงจรออกเป็นชีทย่อยๆ 20 ชีท ซึ่งในการทำวงจร Schematic จึงต้องใช้การออกแบบโดยวิธีมัลติชีทซึ่งจำเป็นต้องใช้ตัวคอมโพเนนท์ในการเชื่อมต่อข้ามชีทซึ่งมี 2 ตัวด้วยกันคือ $OFFPAG.OUT และ $OFFPAG.IN สำหรับข้อมูลที่ออกจากชีทและเข้าชีท ตามลำดับ

สำหรับสัญลักษณ์คอมโพเนนท์ ที่มีอยู่แล้วสามารถเรียกใช้ได้จากในส่วนของคอมโพเนนท์ไลบรารี ส่วนสัญลักษณ์คอมโพเนนท์ที่ไม่มีอยู่ในไลบรารี จะสามารถสร้างขึ้นได้ ในขั้นตอนดังต่อไปนี้

1. ทำการวาดส่วนสัญลักษณ์ Body ซึ่งสามารถวาดทั้งรูปสี่เหลี่ยม, เส้นตรง และส่วนโค้ง โดยมีเส้นแบบต่าง ๆ ให้เรียกใช้ เช่น เส้นธรรมดา เส้นทึบ เส้นประ เป็นต้น

2. ใส่พิน และ ชื่อพินโดยในการวางพินจะต้องวางบนพินกริดและมีระยะระหว่างพินไม่น้อยกว่าที่ได้เซตเอาไว้ในหัวข้อที่ 5.2.1 รวมทั้งต้องคำนึงถึงระยะของพิน ในส่วนของการสร้าง Schematics ด้วย

3. ส่วนของการเปลี่ยนแปลงแก้ไขซึ่งมีด้วยกัน 3 ขั้นตอน คือ

1. เลือกออปเจ็กต์

2. เปลี่ยนแปลงแก้ไข

3. ยกเลิกการเลือกออปเจ็กต์

4. ทำการ Check ตามกฎของการสร้างสัญลักษณ์โดยจะต้องทำการตรวจสอบให้เรียบร้อยก่อนทำการเซฟลงดิสก์ ถ้าในการ Check มี Error จะต้องทำการแก้ไขให้ถูกต้องทั้งหมดเสียก่อน จึงจะนำไปใช้ในโปรแกรม DA สำหรับการสร้าง ต่อไปได้

สำหรับ Error ที่สามารถจะเกิดในการสร้างสัญลักษณ์ได้มีดังนี้ คือ

- ไม่มีพินในตัวคอมโพเนนท์
- มีชื่อพินซ้ำ
- ค่าของ Property ผิดปกติ

5. ทำการเซฟสัญลักษณ์คอมโพเนนท์แล้วออกจากโปรแกรม SYMED ซึ่งถือว่าเป็นการเสร็จสิ้นในส่วนของการสร้างสัญลักษณ์นั้น ๆ

5.3 ส่วนการสร้าง ไลบรารีพาร์ท

ในส่วนของการสร้างไลบรารีพาร์ท (จีโอเมตรี) ซึ่งจะเรียกใช้โปรแกรมที่มีชื่อว่า LIBRARIAN ของ Mentor Graphics ซึ่งเป็นเครื่องมือที่ไว้ใช้ในการสร้าง และแก้ไขไลบรารีพาร์ท ในการทำงานครั้งนี้ ได้ทำการแบ่งชนิดของไลบรารีพาร์ทออกเป็น 3 พาร์ทใหญ่ ๆ ด้วยกัน คือ

1. คอมโพเนนท์พาร์ท
2. บอร์ดพาร์ท
3. แพดสแตกพาร์ท

สำหรับรายละเอียดขั้นตอนการสร้าง และสิ่งที่ต้องการของแต่ละส่วนนี้ จะเป็นดังนี้ คือ

5.3.1 คอมโพเนนท์พาร์ทในพาร์ท นี้จะเป็นส่วนที่บอกถึงตำแหน่งโครงสร้างของพินบนตัวอุปกรณ์ชนิดของรูเจาะ ขนาดที่แท้จริงของอุปกรณ์จึงทำให้ก่อนที่จะทำการสร้างในส่วนพาร์ท จะต้องรู้ถึงสิ่งต่อไปนี้ คือ

- รายละเอียดของพินบนตัวอุปกรณ์ทุก ๆ พิน
- ขนาดของตัวอุปกรณ์ (Placement Outline)
- สัญลักษณ์บนแผ่นวงจรพิมพ์ (Silkscreen Marking)
- Reference Designator

ซึ่งขั้นตอนในการออกแบบสร้างส่วนของคอมโพเนนท์พาร์ท เมื่อได้ทราบข้อมูลทั้งหมดดังกล่าวข้างต้นแล้ว คือ

1. ทำการวางขา (พิน) ของตัว คอมโพเนนท์ ตามตำแหน่งที่ถูกต้อง
2. ทำการกำหนดชนิดของแพดสแตก (สำหรับวิธีการสร้างแพดสแตกจะกล่าวโดยละเอียดในหัวข้อ 5.3.3)
3. วาดสัญลักษณ์บนแผ่นวงจรพิมพ์โดยเขียนลงบนตำแหน่งของซิลสกรีนเลเยอร์
4. วาดเส้นกำหนดขนาดตัวอุปกรณ์ (Placement Outline)
5. กำหนดเท็กซ์เพื่อเป็นตัวอ้างอิง เช่น $DIP14

6. ในตัวคอมโพเนนท์บางตัวอาจต้องมีการกำหนดพื้นที่ห้ามการ Routing ด้วย เพราะฉะนั้นจะต้องมีการกำหนดขอบเขตนี้ด้วย เรียกว่า Routing Keepout

7. ทำการเซฟพาร์ทเพื่อนำไปใช้งานต่อไป

5.3.2 บอร์ดพาร์ท ในพาร์ทส่วนนี้จะเป็นส่วนที่บอกถึงเนื้อที่ของการวางตัวคอมโพเนนท์ เนื้อที่ของการลากสายสัญญาณ โดยกฎของการ Layout ต่าง ๆ จะถูกกำหนดอยู่ในส่วนของ บอร์ดพาร์ทนี้ด้วย ก่อนที่จะสร้างส่วนนี้ มีความจำเป็นที่จะต้องรู้ถึงสิ่งต่าง ๆ ของบอร์ท ดังต่อไปนี้

- ขนาดของบอร์ด
- ส่วนของการวางอุปกรณ์
- ส่วนของการ Routing
- ตำแหน่งและขนาดรูที่จะเจาะ

สำหรับขั้นตอนในการออกแบบส่วนของบอร์ดพาร์ท มีดังต่อไปนี้คือ

- กำหนดขนาดของบอร์ดทั้งด้านกว่งและด้านยาวขนาดของสายที่จะใช้ลาก กำหนดจำนวนเลเยอร์ที่จะใช้ และทิศทางของการ Route ของแต่ละเลเยอร์
- กำหนดพื้นที่สำหรับการวางตัวคอมโพเนนท์
- กำหนดพื้นที่สำหรับการ Routing และพื้นที่ห้าม Routing
- เจาะรู (Drill-Holes) ตามตำแหน่งและขนาดของรูตามที่กำหนด
- ทำการ เซฟ บอร์ด พาร์ท เพื่อนำไปใช้งานต่อไป

5.3.3 แพดสแตกพาร์ทในพาร์ทส่วนนี้จะแบ่งออกได้เป็น 2 ส่วนใหญ่ ๆ คือ

1. พินแพดสแตกจะบอกถึงขนาดและรูปร่างของที่สำหรับขาของคอมโพเนนท์โดยในส่วนนี้ยังแบ่งได้เป็น Thru-Pin , Surface หรือ Blind แพดสแตก

2. เวียแพดสแตกจะบอกถึงขนาดและรูปร่างของพื้นที่สำหรับการเชื่อมต่อลายเส้นสัญญาณระหว่าง เลเยอร์ ที่แตกต่างกันสิ่งจำเป็นที่จะต้องรู้ก่อนจะสร้างส่วน แพดสแตก คือ

- ชนิดของแพดสแตกที่จะสร้าง
- ขนาดของรูที่จะใช้ (Drill Size)

- พื้นที่เชื่อมสัญญาณในชั้นสัญญาณ
- พื้นที่ป้องกันสัญญาณในชั้นพาวเวอร์

5.4 ส่วนการทำแมปปิ้ง และแคตาล็อกไฟล์

ในส่วนการสร้างแมปปิ้งและแคตาล็อกไฟล์นั้นจะเรียกใช้โปรแกรมที่มีชื่อว่า LIBRARIAN ของ Mentor Graphics ซึ่งเป็นโปรแกรมตัวเดียวกับที่ใช้ในการสร้างตัว ไลบรารีพาร์ท ในหัวข้อนี้จะแยกอธิบายวิธีการของแต่ละตัวได้ดังนี้ คือ

5.4.1 แมปปิ้งไฟล์ จะเป็นตัวที่เชื่อมโยงระหว่างสัญลักษณ์ที่สร้างขึ้นกับลักษณะทางกายภาพของตัวคอมโพเนนท์ โดยจะบอกถึงข้อมูลเกี่ยวกับการสแวปของเกตและพินในตัวคอมโพเนนท์ สำหรับขั้นตอนของการสร้างส่วนแมปปิ้งไฟล์ มีดังนี้คือ

1. ศึกษาข้อมูลตัวคอมโพเนนท์ที่จะทำการแมปจาก Data Book
2. กำหนดตัวสัญลักษณ์ และตัวจีโอเมตรี
3. สร้างส่วนหมายเลขพาร์ท ซึ่งจะประกอบด้วยส่วนที่จำเป็นดังนี้
   - ชื่อ หมายเลขพาร์ท (pn-741s00)
   - ชื่อ แคตาล็อกไฟล์ที่จะเก็บ (Design Catalog)
   - ชื่อ จีโอเมตรี ($dip14)
   - สัญลักษณ์ลอจิก ($741s00)
   - จำนวนสัญลักษณ์ในจีโอเมตรี เช่น ตัว 741s00 มีได้ 4 ตัว เป็นต้น
4. กำหนดขาของสัญลักษณ์ในตัวจีโอเมตรี
5. กำหนดขาพาวเวอร์ เช่น VCC , Ground
6. ทำการ Check หมายเลขพาร์ทให้ถูกต้อง
7. ทำการเก็บข้อมูลในส่วนของแคตาล็อกไฟล์ซึ่งจะได้กล่าวต่อไป โดยในการเก็บนี้จะสร้างส่วนที่เป็นแมปปิ้งไฟล์ขึ้นด้วย

5.4.2 แคตาล็อกไฟล์ ทุก ๆ สัญลักษณ์ในวงจร Schematic จะต้องมีข้อมูลอยู่ในส่วนของแคตาล็อกไฟล์ซึ่งจะประกอบด้วย

1. ชื่อของหมายเลขพาร์ท
2. ชื่อของคอมโพเนนท์
3. ชื่อของจีโอเมตรีที่ใช้
4. แมปปิ้งไฟล์
5. สัญลักษณ์ จำนวนเกต ในตัวของคอมโพเนนท์

สำหรับการสร้างไฟล์นี้จะมาจากการสร้างตอนแรกเท่านั้น หลังจากชั้นข้อมูลในแคตาล็อกไฟล์จะมีเพิ่มขึ้น ได้มาจากในส่วนของการทำแมปปิ้งไฟล์ ซึ่งทุก ๆ ครั้งที่ทำการเซฟในส่วนของการแมป ก็จะเป็นการอัปเดทในตัวที่มีอยู่แล้วในแคตาล็อกไฟล์และเป็นการเพิ่มในตัวที่ยังไม่มีอยู่ในส่วนของแคตาล็อกไฟล์นั้นเอง

5.5 ส่วนการทำ Packaging

ในส่วนของการทำ Packaging นั้นจะเรียกใช้โปรแกรมที่มีชื่อว่า PACKAGE ของ Mentor Graphics โดยก่อนที่จะเริ่มทำส่วนของ PACKAGE ได้จำเป็นที่จะต้องมีส่วนต่าง ๆ ที่สร้างไว้แล้ว เพื่อที่จะนำมาใช้งาน คือ

- จีโอเมตรีพาร์ทไฟล์ สร้างจากส่วน LIBRARAIN ซึ่งประกอบด้วย ส่วนแพดสแตก, คอมโพเนนท์, บอร์ด
- พาร์ทแมปปิ้งไฟล์ และแคตาล็อกไฟล์ ซึ่งจะประกอบด้วยข้อมูลเกี่ยวกับการทำ PACKAGE ของตัวสัญลักษณ์ลอจิก
- PACKAGE Configuration File จะเป็นส่วนำไฟล์ ASCII ซึ่งถูกสร้างโดยอัตโนมัติในส่วนของ LIBRARAIN
- Design File (peb-design.erel) จะประกอบด้วยข้อมูลส่วนเชื่อมต่อ ซึ่งถูกสร้างจาก DA

สำหรับลำดับขั้นตอนการ Packaging มีดังต่อไปนี้ คือ

5.5.1 เรียกใช้โปรแกรม PACKAGE

5.5.2 Load ส่วนของ แคตตาล็อก ที่จำเป็นสำหรับการทำ PACKAGE

5.5.3 ทำการเปลี่ยนแก้ไขข้อมูลที่ต้องการโดยใช้หลักการของการเลือกแก้ไข แล้วยกเลิกการเลือก

5.5.4 ทำการ Packaging อัตโนมัติโดยเรียกใช้คำสั่ง Build

5.5.5 ทำการเซฟในส่วนของ PACKAGE ซึ่งจะทำให้เกิดไฟล์ต่าง ๆ ดังต่อไปนี้คือ

- Component File (Comp_File) เป็นข้อมูลเกี่ยวกับคอมโพเนนท์ทุกตัวที่ใช้
- Nets File (Nets_File) เป็นข้อมูลของชื่อ Net และ พินทั้งหมด
- Instance File (Inst_File) เป็นข้อมูลของสัญลักษณ์ และพินของเกต
- Spares File (Spares_File) เป็นข้อมูลของเกตที่เหลืออยู่ยังไม่ได้ใช้งาน
- Package File (Pkags_File) เป็นจำนวนและชนิดของจีโอเมตรีที่ใช้
- Notes File (Notes_File) เป็นข้อมูลของการแก้ไขใน Nets, พินต่าง ๆ ของ Schematic

5.5.6 ทำการอ่านพาร์ทต่าง ๆ ที่จะใช้ใน PACKAGE แล้วทำการ Check ให้ถูกต้อง

5.5.7 ทำการเซฟพาร์ทสำหรับไว้ใช้ในการ Placement และการ Routing ต่อไป

5.5.8 ออกจากโปรแกรม PACKAGE ซึ่งถือว่าเป็นการเสร็จสิ้นในส่วนของการทำ Packaging

5.6 ส่วนการ Placement

ในส่วนของการวางคอมโพเนนท์จะเรียกใช้โปรแกรมที่มีชื่อว่า LAYOUT ของ Mentor Graphics ซึ่งจะเป็นการวางจีโอเมตรี ของตัวคอมโพเนนท์ลงบนตำแหน่งของบอร์ดเอาท์ไลน์ซึ่งได้กำหนดไว้ในส่วนของการทำบอร์ดพาร์ท โดยสามารถวางได้ทั้งด้านบน ด้านล่าง หรือทั้งสองด้านของบอร์ดได้

สำหรับวิธีในการ Placement นั้นจะมีด้วยกันหลายวิธีและแต่ละวิธีจะมีขั้นตอนการทำงานดังต่อไปนี้

5.6.1 วิธีเลือก Placement เฉพาะตัวที่ต้องการโดยวิธีการจะต้องเลือกตัวคอมโพเนนท์ที่ต้องการแล้วนำมาวางไว้บนบอร์ดตามตำแหน่งที่ต้องการ โดยวิธีเลือกตัวคอมโพเนนท์อาจจะเลือกได้จากค่า Property ของตัวคอมโพเนนท์ ซึ่งได้แก่

- ค่าของ Reference เช่น U1, J1 เป็นต้น
- ตัวคอมโพเนนท์
- ตัวจีโอเมตรี

5.6.2 วิธีเลือก Placement สำหรับสายบัสซึ่งจะทำให้ได้ตัวคอมโพเนนท์ที่มีสายบัสเดียวกัน อ ศ-ในกลุ่มของคอมโพเนนท์เดียวกันซึ่งมีวิธีการทำงานดังต่อไปนี้

1. Set Criteria ซึ่งจะเป็นการกำหนดค่าต่ำสุดของจำนวนคอมโพเนนท์และสายของแต่ละคอมโพเนนท์ สำหรับการพิจารณาของสายบัส

2 สร้างกลุ่มของตัวคอมโพเนนท์ ตามจำนวนที่ได้เซตไว้ในหัวข้อที่ 1 ซึ่งในแต่ละกลุ่มจะมีสายสัญญาณบัสเชื่อมต่อกันเป็นสายเดียวกัน

3. ทำการเลือกกลุ่มของตัวคอมโพเนนท์ที่ต้องการโดยการใส่ชื่อที่ต้องการของกลุ่มของบัส

4. ทำการแมปตัวคอมโพเนนท์ลงบนบอร์ด ซึ่งมีวิธีการแมปหลายวิธีการดังต่อไปนี้คือ

- ออโตเมติกแมปปิ้ง ลงบนพื้นที่ที่กำหนดให้
- ออโตเมติกแมปปิ้ง ภายในบอร์ด
- เลือกแมปตัวคอมโพเนนท์เป็นตัว ๆ ไปภายในกลุ่มบัส

5.6.3 วิธีเลือก Placement โดยอัตโนมัติทั้งหมดให้วางลงบนบอร์ดได้ หรือ อาจจะวางโดยอัตโนมัติเป็นตัว ๆ ไปก็ได้

เมื่อมีการวางตัวคอมโพเนนท์ทั้งหมดแล้วลงบนบอร์ด เรายังสามารถที่จะแก้ไขการ Placement ได้โดยการใช้คำสั่งเคลื่อนย้าย , หมุน เป็นต้น หลังจากวางตัวคอมโพเนนท์ลงบอร์ดเสร็จสิ้นเรียบร้อยแล้วก็จะทำการเซฟซึ่งเป็นขั้นตอนสุดท้ายสำหรับขั้นตอนการ Placement

5.7 ส่วนการ Routing

ในส่วนของการ Routing จะเรียกใช้โปรแกรมที่ชื่อว่า LAYOUT เช่นเดียวกับในส่วนของการ Placement โดยสามารถที่จะทำการ Route ได้ทั้งหมดจากวางตัวคอมโพเนนท์ทั้งหมดลงบนบอร์ดแล้ว หรือจะมีคอมโพเนนท์เพียงบางตัวเท่านั้นก็ได้ สำหรับวิธีการและขั้นตอนของการ Route มีด้วยกันหลายวิธี แต่ก่อนที่จะได้กล่าวโดยละเอียดต่อไป จะกล่าวถึงกฎที่จำเป็นต้องกำหนดไว้ก่อนที่จะทำการ Route สายสัญญาณ เช่น

- ขนาดความกว้างของสายสัญญาณ
- ระยะห่างระหว่างทางเดินของสายสัญญาณ
- ระยะห่างระหว่างแพด
- ขนาดและชนิดของเวีย
- รูปแบบการ Route

วิธีการและขั้นตอนการ Route มีดังนี้

5.7.1 **Interactive Routing** เป็นการ Route สายสัญญาณทีละเส้น ตามแนวทางที่ต้องการมีขั้นตอนดังต่อไปนี้ คือ

- เลื่อนเคอร์เซอร์ไปยังตำแหน่งของสายสัญญาณที่ต้องการ Route
- ทำการเลือกสายสัญญาณขึ้น
- เลื่อนเคอร์เซอร์ไปยังตำแหน่งที่ต้องการ Route สาย (vertex)
- ถ้าต้องการเปลี่ยนเลเยอร์ก็ให้ใส่เวียเข้าไป
- ทำเช่นนี้ไปเรื่อย ๆ จนกระทั่งเชื่อมสัญญาณเส้นนั้นเรียบร้อย

5.7.2 **Automatic Routing** ถึงแม้ว่าจะเป็นการทำงานโดยอัตโนมัติแต่จะต้องมีการควบคุมการทำงานของมัน ซึ่งจะทำในส่วนของการ Set Autorouter โดยสิ่งที่เราจำเป็นต้องควบคุมได้แก่

- สายสัญญาณที่เลือกให้ทำการ Route (อาจจะเลือกทั้งหมดเลยก็ได้)
- เลเยอร์ที่ใช้ในการ Route (จำนวน และชั้นของเลเยอร์)
- พื้นที่สำหรับการ สายสัญญาณ Route (อาจจะทั้งบอร์ดเลยก็ได้)
- จำนวนรอบของการ Route

- กำหนดวิธีการและความสำคัญของการ Route
- ชนิดของรอบที่ทำการ Autorouting ซึ่งมีด้วยกัน 3 ชนิดคือ
  1. Pattern เป็นการ Route ที่จะให้สายสัญญาณแต่ละเส้นลากสายอยู่ภายในเลเยอร์เดียว
  2. Automatic เป็นการ Route โดยวิธีทั่ว ๆ ไป
  3. Manufacturing เป็นการ Route สาย โดยพยายามที่จะทำให้เกิดเวียและ Bends ให้น้อยที่สุดซึ่งจะเรียกว่า Cleanup

5.8 การทำ Manufacturing Output

ในส่วนนี้จะใช้โปรแกรมที่มีชื่อว่า Fablink โดยข้อมูลที่จะเกิดจากส่วนนี้คือ

1. ข้อมูล Photoplot
2. ข้อมูล Drill Machine
3. ข้อมูล Milling Machine
4. Drawings
5. Document และ Reports
6. ไฟล์ ของข้อมูลที่แก้ไข

สำหรับรายละเอียดวิธีการของข้อมูลต่าง ๆ ที่ได้กล่าวมาแล้วนั้นจะขอกล่าวถึงเฉพาะข้อมูลที่อยู่ในขั้นตอนการทำงานเท่านั้น คือ

5.8.1 Photoplot Data จะเป็นส่วนของการทำอาร์ตเวิร์คไฟล์ซึ่งจะใช้ในส่วนของการทำ Manufacturing โดยในแผ่นฟิล์มของอาร์ตเวิร์คจะเป็นเลเยอร์แต่ละชั้นของแผ่นวงจรพิมพ์ โดยขั้นตอนการสร้างข้อมูลทางอาร์ตเวิร์คมีดังต่อไปนี้คือ

1. ทำการเซตข้อกำหนดต่าง ๆ ของอาร์ตเวิร์ค เช่น ขนาดของฟิล์ม ข้อมูลที่จะต้องการเขียน
2. ทำการเซตตารางของ Photoplotter Aperture โดยกำหนดรูปร่างและขนาดของส่วนต่าง ๆ ที่จะวาดลงบนอาร์ตเวิร์คฟิล์ม
3. ทำการ Check อาร์ตเวิร์คสแตกซึ่งเป็นเลเยอร์ที่ใช้ทำลงไปใน อาร์ตเวิร์คไฟล์

4. ทำการสร้างอาร์ตเวิร์ค ซึ่งเราสามารถที่จะกำหนดชนิดและรูปแบบที่แตกต่างกันได้

5. หลังจากสร้างเรียบร้อยแล้ว เราสามารถที่จะวิว และอีดิตได้ ก่อนที่จะทำ Photoplot

5.8.2 Drill Machine Data ซึ่งจะเป็นส่วนที่บอกตำแหน่งและขนาดของรูเจาะต่าง ๆ ในแผ่นวงจรพิมพ์ไม่ว่าจะเป็น Thru-Pin หรือ เวียโฮลโดยขั้นตอนการสร้าง Drill Data มีดังต่อไปนี้

1. ทำการ Check ข้อกำหนดต่าง ๆ ของ Drill Data

2. ทำการเซตตารางของ Drill ซึ่งจะเป็นการกำหนดขนาดและตำแหน่งของ Drill Bit

3. ทำการ Generate Drill เหมือนกับที่ทำในอาร์ตเวิร์ค

4. ทำการวิว และแก้ไข ก่อนทำ Photoplot

5.8.3 Milling Machines Data เป็นข้อมูลสำหรับเอาไว้ใช้ในการตัดบอร์ด ออกเป็นแผ่น ๆ ตามขนาดที่ต้องการ มีวิธีการทำ Mill Data ดังต่อไปนี้

1. กำหนด Path ของ Mill Data ใน Softwore Fablink

2. ทำการ Check ข้อกำหนดต่างๆ ของ Mill Data

3. ทำการเซตตารางของ Drill และ Mill

4. ทำการสร้าง Mill Data

5. ทำการวิวและถ้าต้องการก็สามารถแก้ไขได้ด้วย

5.8.4 Drawing ในการทำส่วนของ Manufacturing Drawing สามารถที่ทำได้หลายวิธีการเช่น

1. ใช้ Drawing Template มาตรฐานตามต้องการ (size A ถึง F)

2. ในส่วนภาพของบอร์ดกับเลเยอร์ที่เลือกสามารถทำเป็นส่วน Drawing ได้โดยอัตโนมัติ

3. ไดเมนชั่นสามารถใส่ได้โดยกำหนดจุดเริ่มต้นและจุดสิ้นสุด

4. Drill Hole จะปรากฏเองโดยอัตโนมัติ

5.8.5 Document & Report ในส่วนของการทำ Manufacturing รายงานที่ได้จาก Fablink มีด้วยกันหลายแบบ เช่น

1. Bill of Material จะเป็นรายละเอียดของส่วนคอมโพเนนท์ที่ใช้ในการออกแบบ

2. Net Connection List จะบอกถึงตำแหน่งของแต่ละคอมโพเนนท์พินบนแผ่นวงจรพิมพ์

โดยจะมีตำแหน่งอ้างอิงอยู่ตำแหน่งหนึ่งบนเซอร์กิตบอร์ด

3. Net Lengths Report เป็นรายละเอียดของความยาวในแต่ละ Net

เอกสารอ้างอิง

ยืน ภู่วรวรรณ . 2533 . เทโนโลยีฮาร์ดแวร์ IBM PC . กรุงเทพมหานคร : บริษัท ซีเอ็ดยูเคชั่น จำกัด .

ถวัลย์ ตั้งลิตานนท์ . 2533 . "MICRO CHANNEL VS EISA สงครามระหว่างบัส" . ไมโครคอมพิวเตอร์ . 5(57) : น. 237-247 .

เอนก ศิรวริยาภิวัฒน์ . 2533 . "ก้าวต่อไปกับ 80486" . ไมโครคอมพิวเตอร์ . 5(61) : น. 269-287 .

Clyd F. Coombs, Jr . 1988 . PRINTED CIRCUITS HANDBOOK (THIRD EDITION) . USA : McGRAW-HILL .

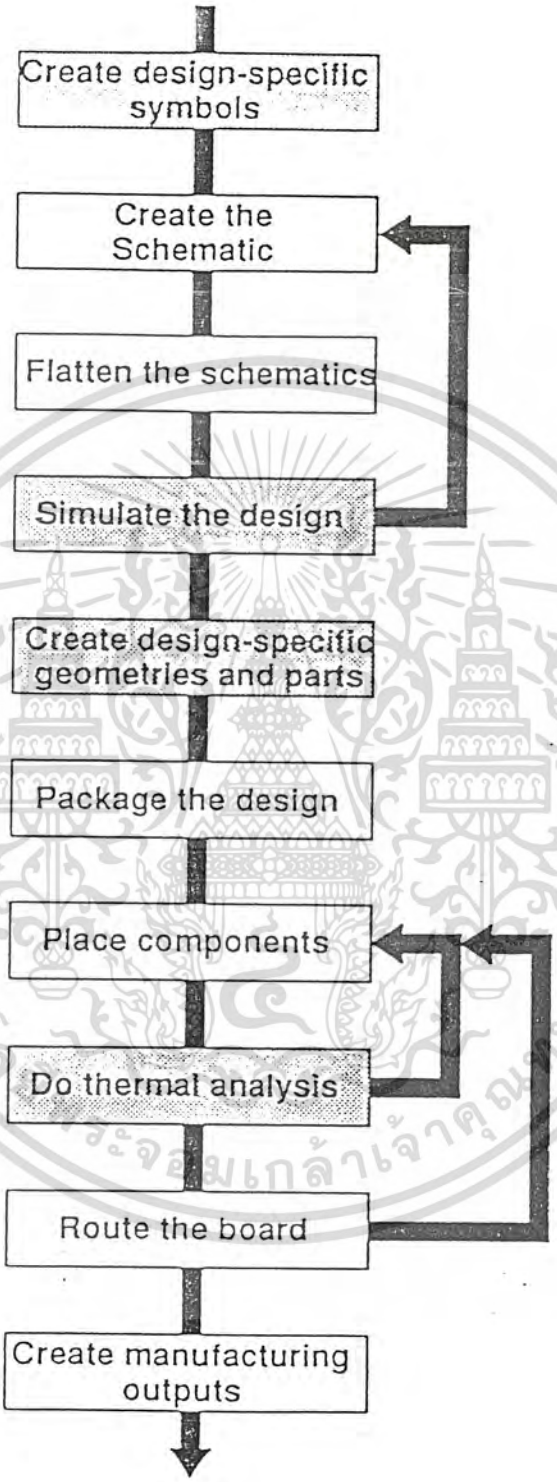
Intel Corporation . 1989 . 82350 EISA CHIP SET . USA : Intel Corporation .

Intel Corporation . 1989 . i486 MICROPROCESSOR . USA : Intel Corporation .

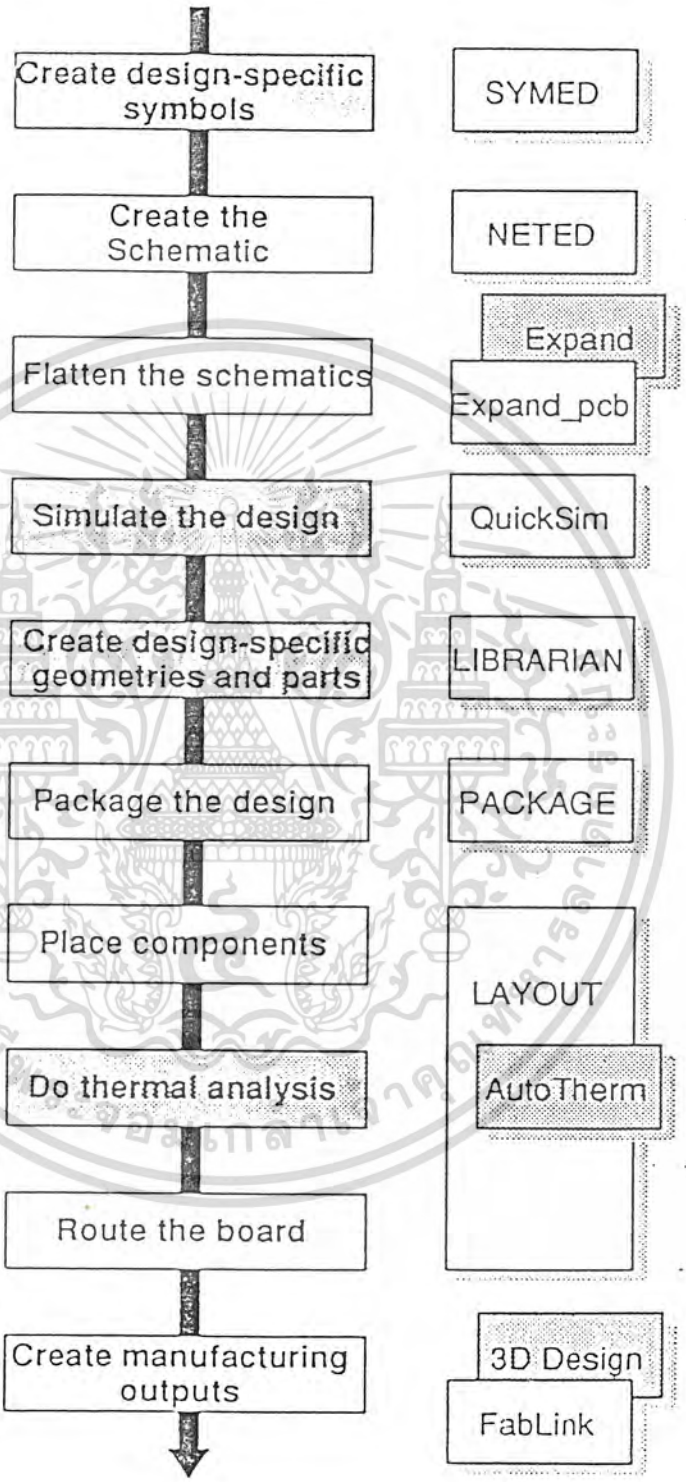
Intel Peripheral Group . 1990 . 82350 EISA CHIP SET DESIGN GUIDE . USA : Intel Corporation .

ภาคผนวก

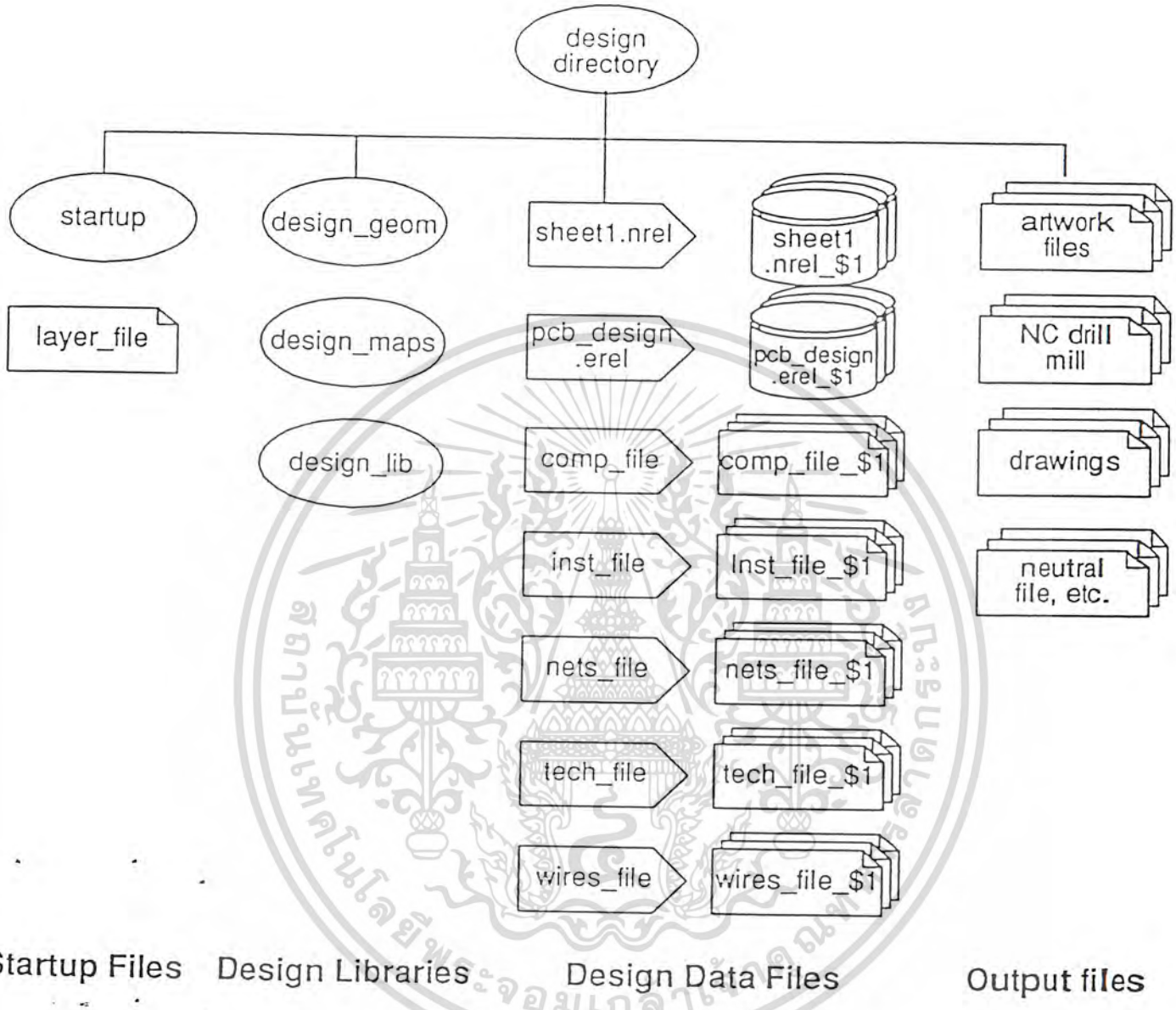
Task Flow



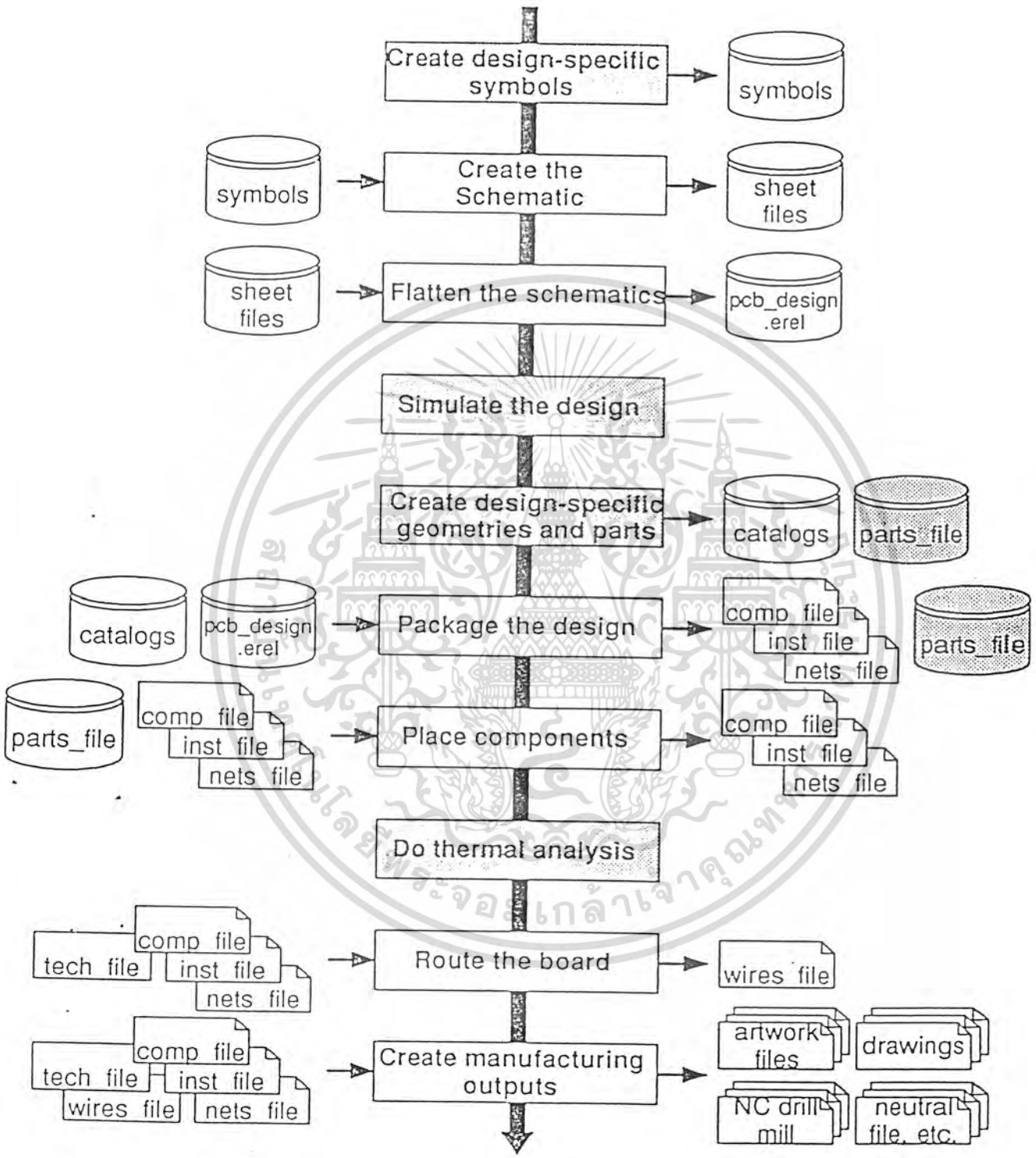
Tool Flow



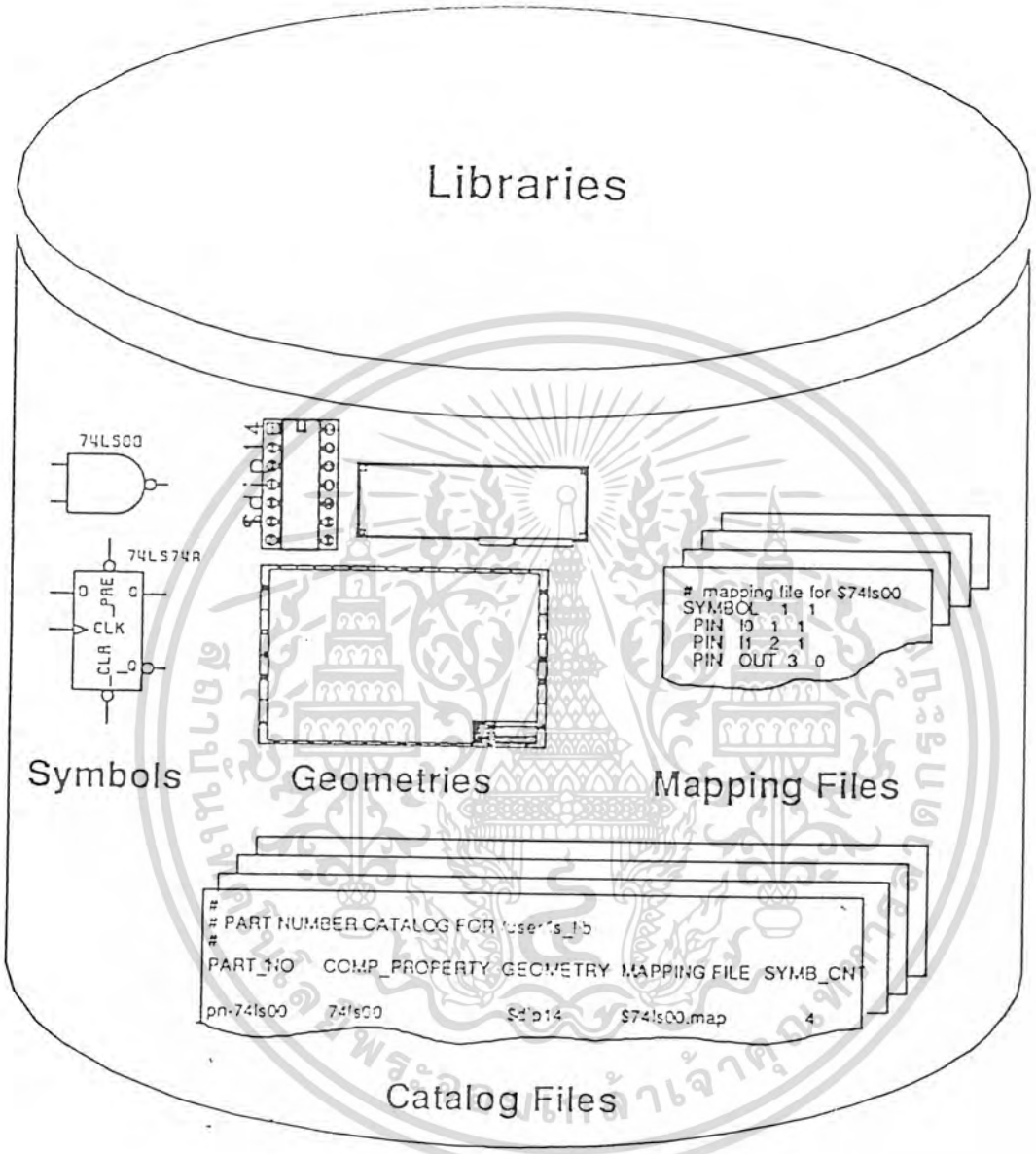
Design Directory



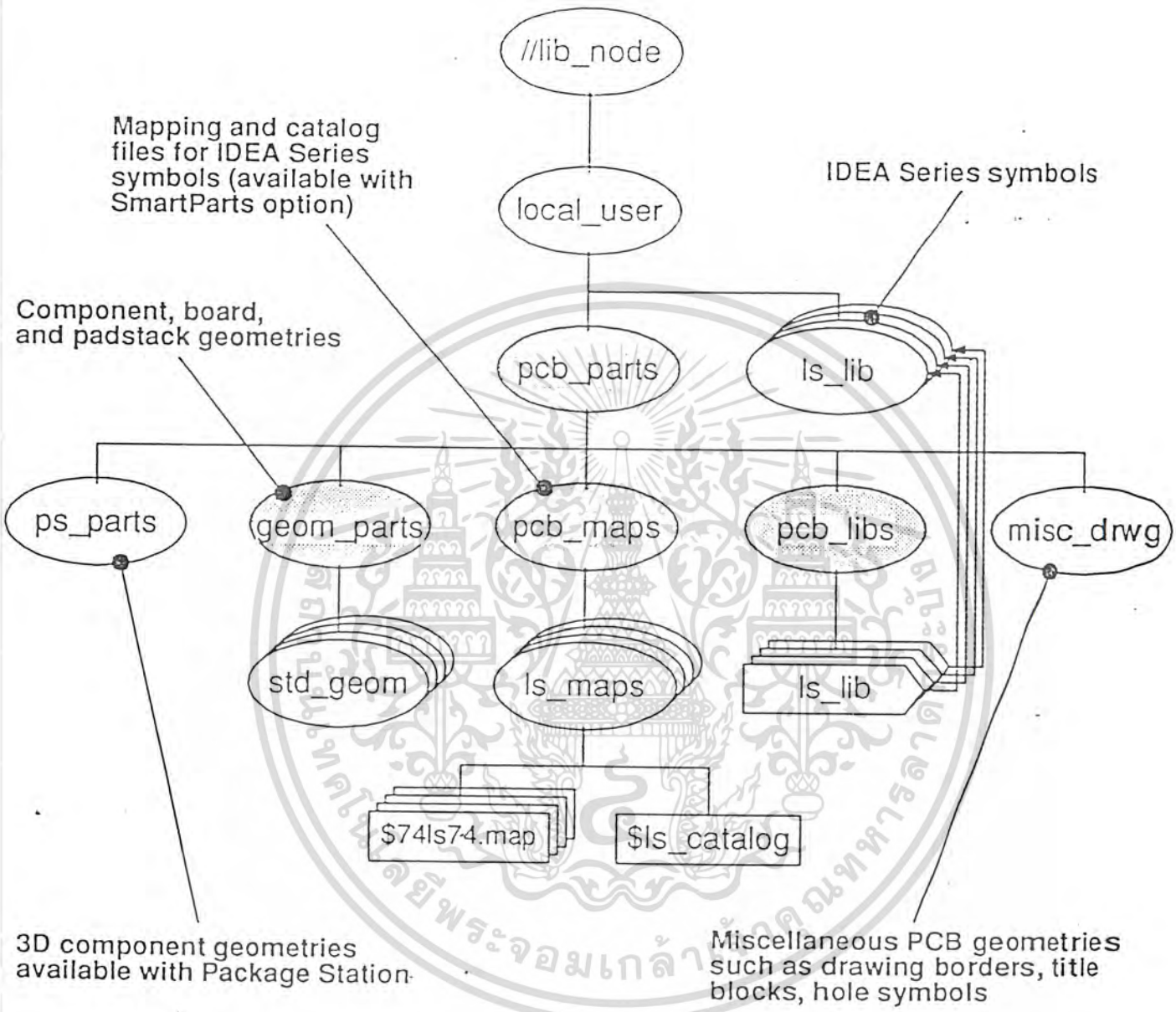
Data Flow

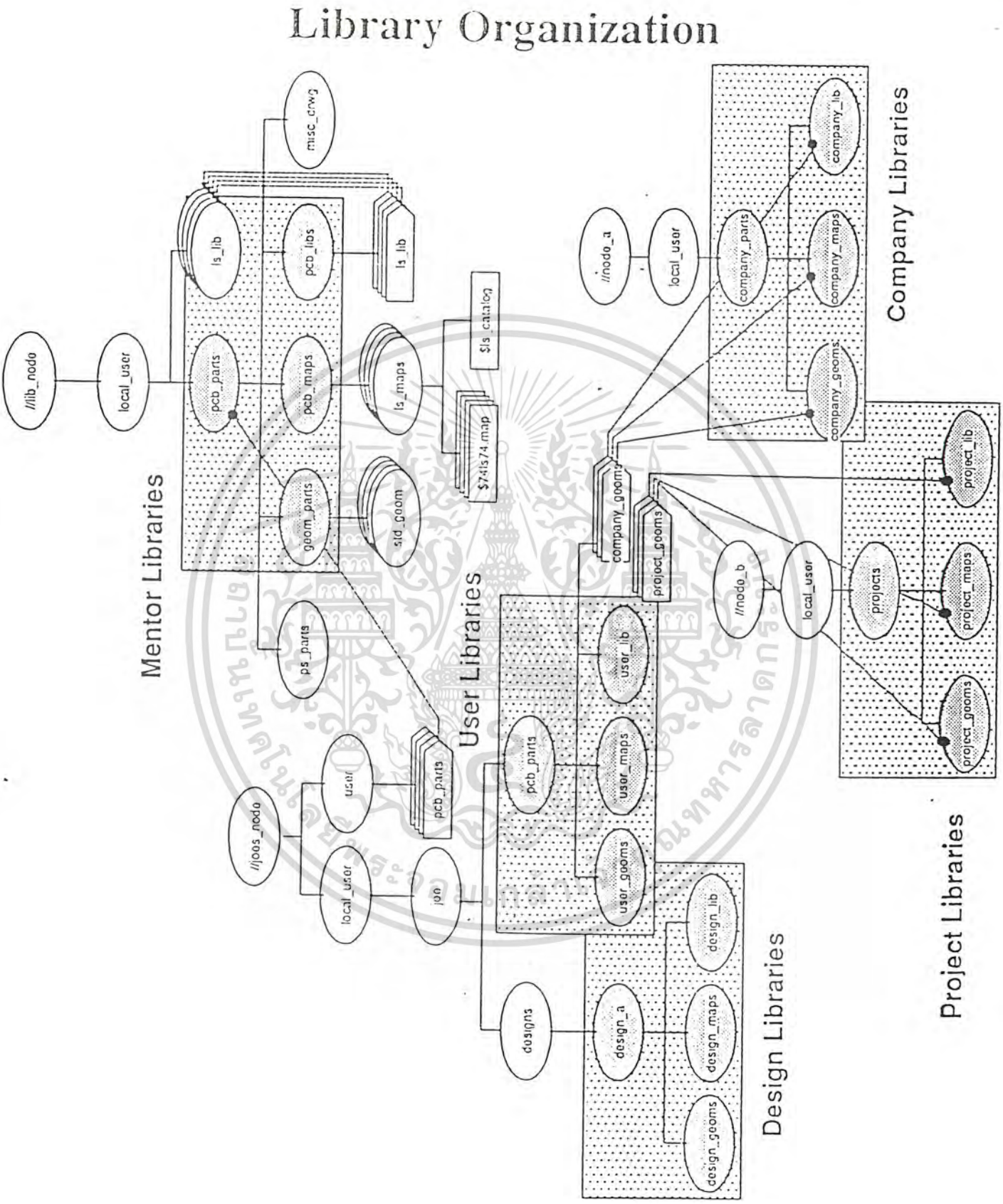


Library Objects

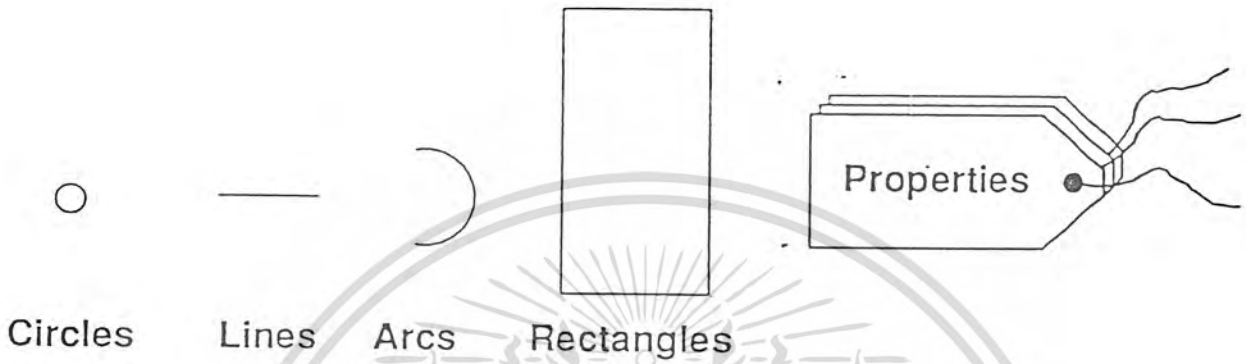


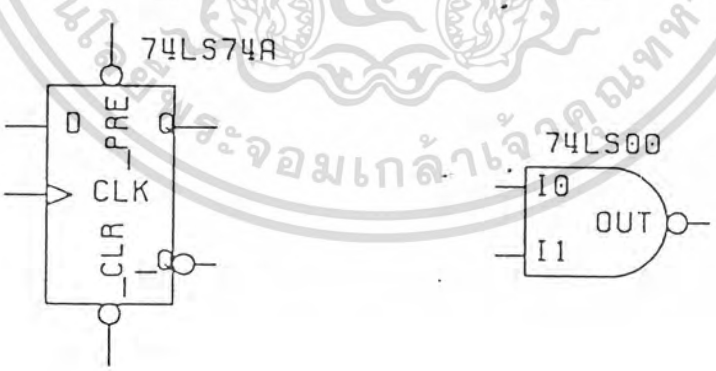
Mentor Libraries



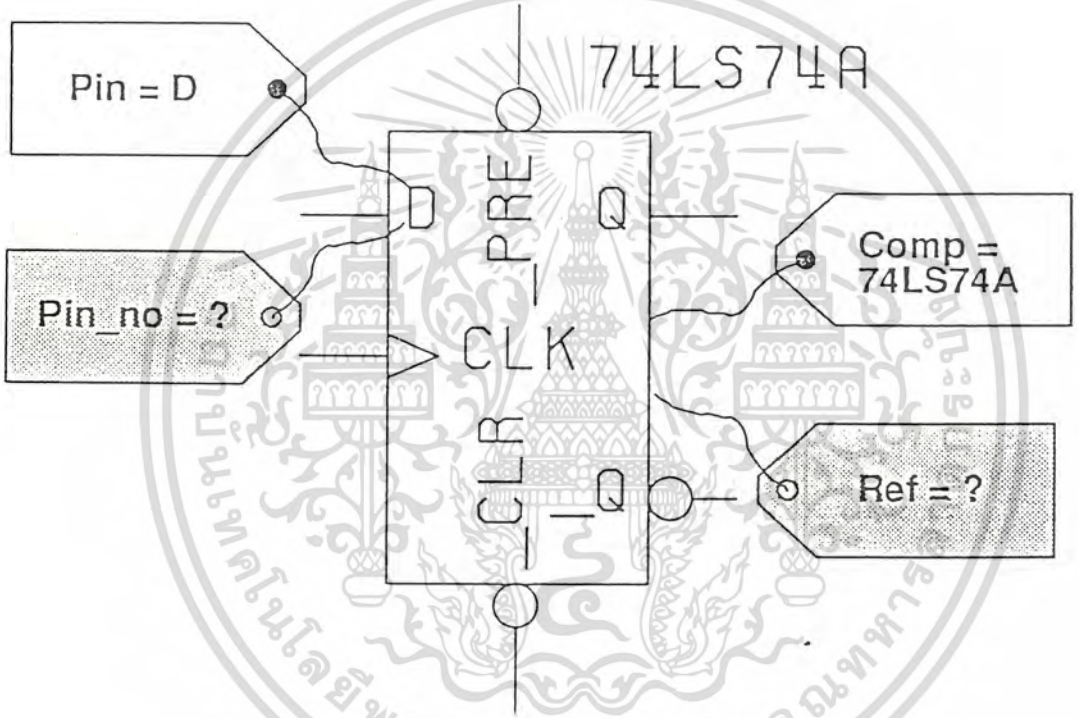


Symbols

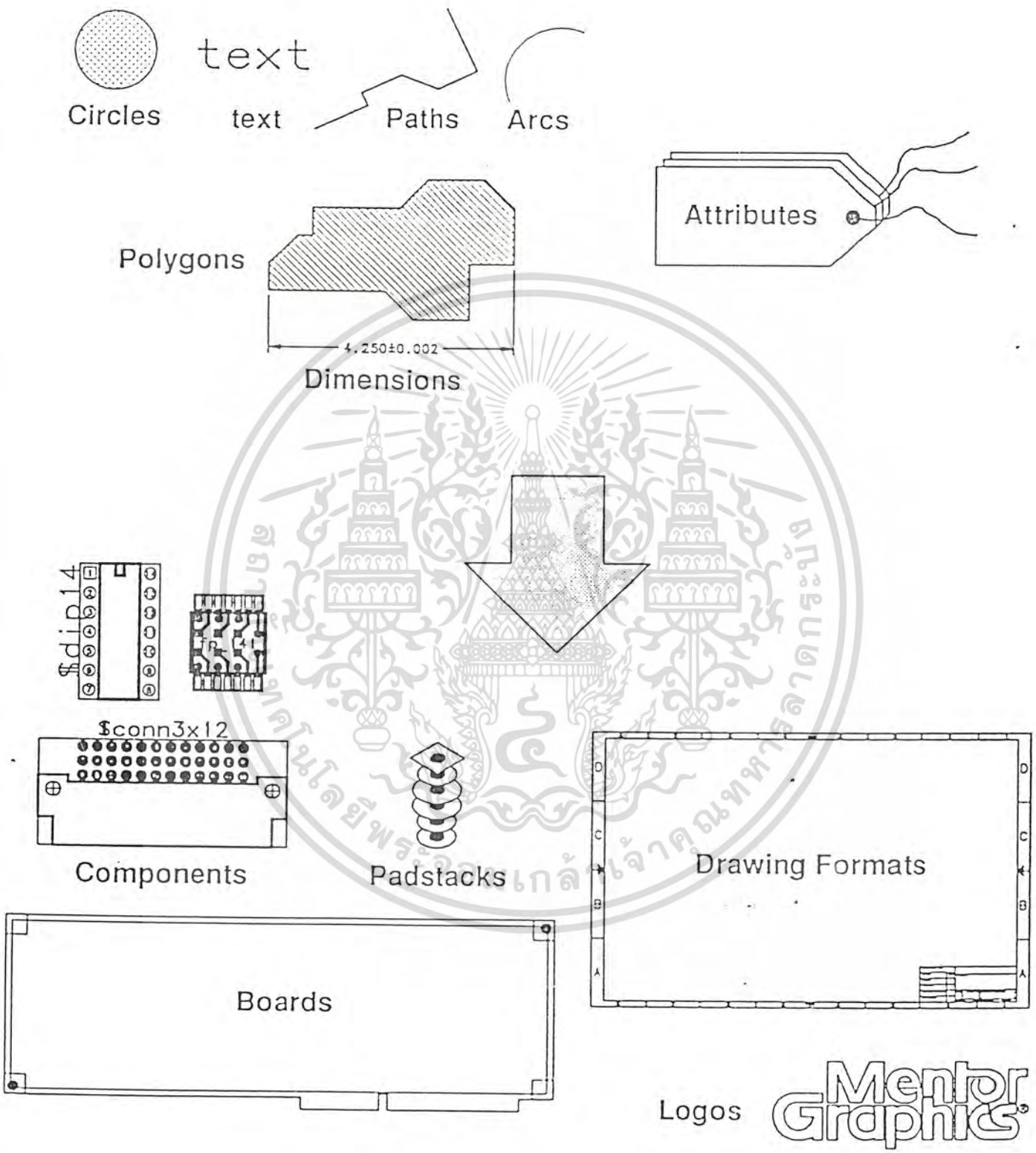




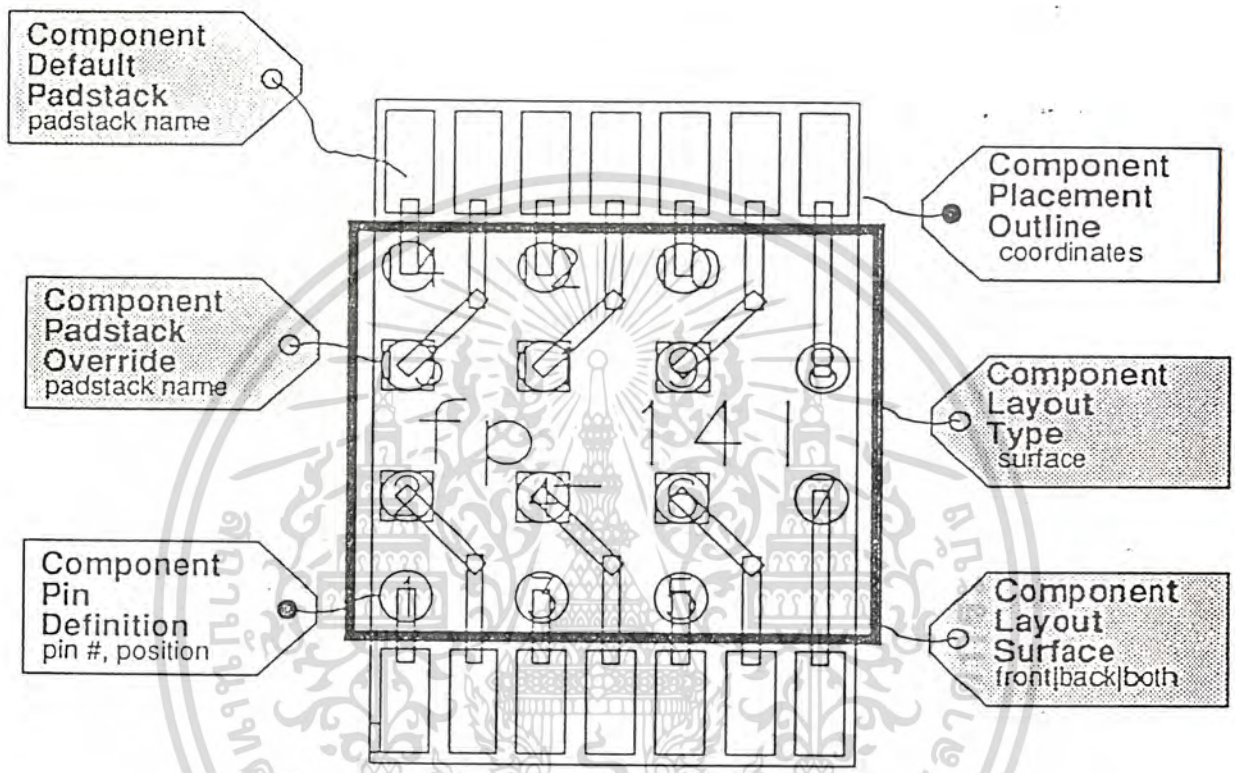
Symbol Properties



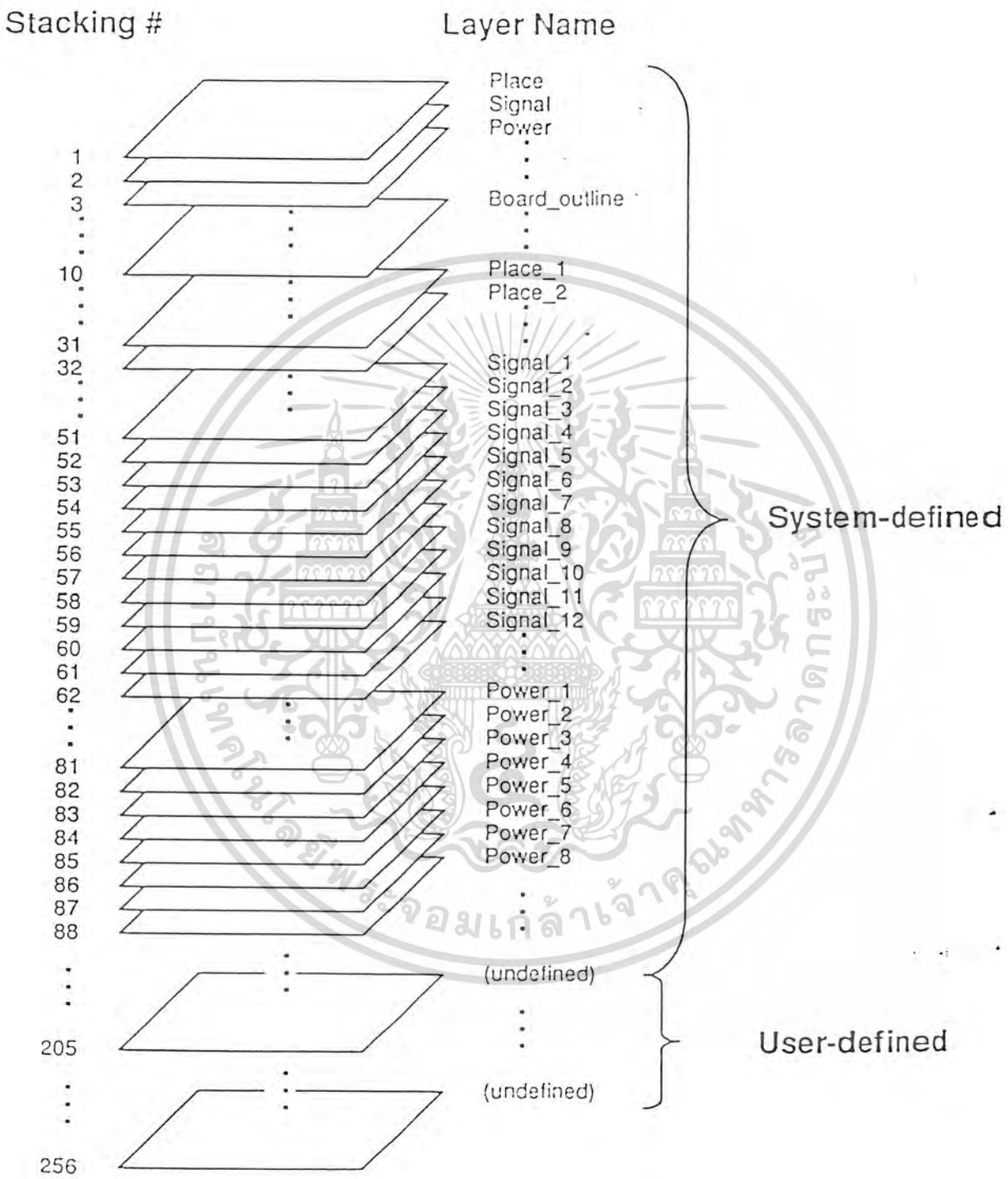
Geometries



Geometry Attributes



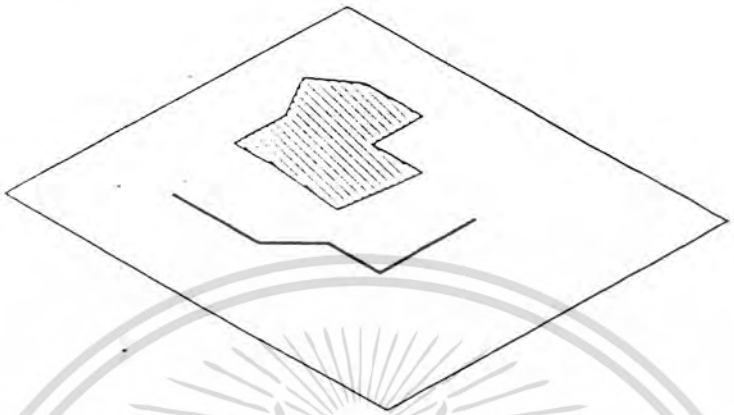
Graphical Layers



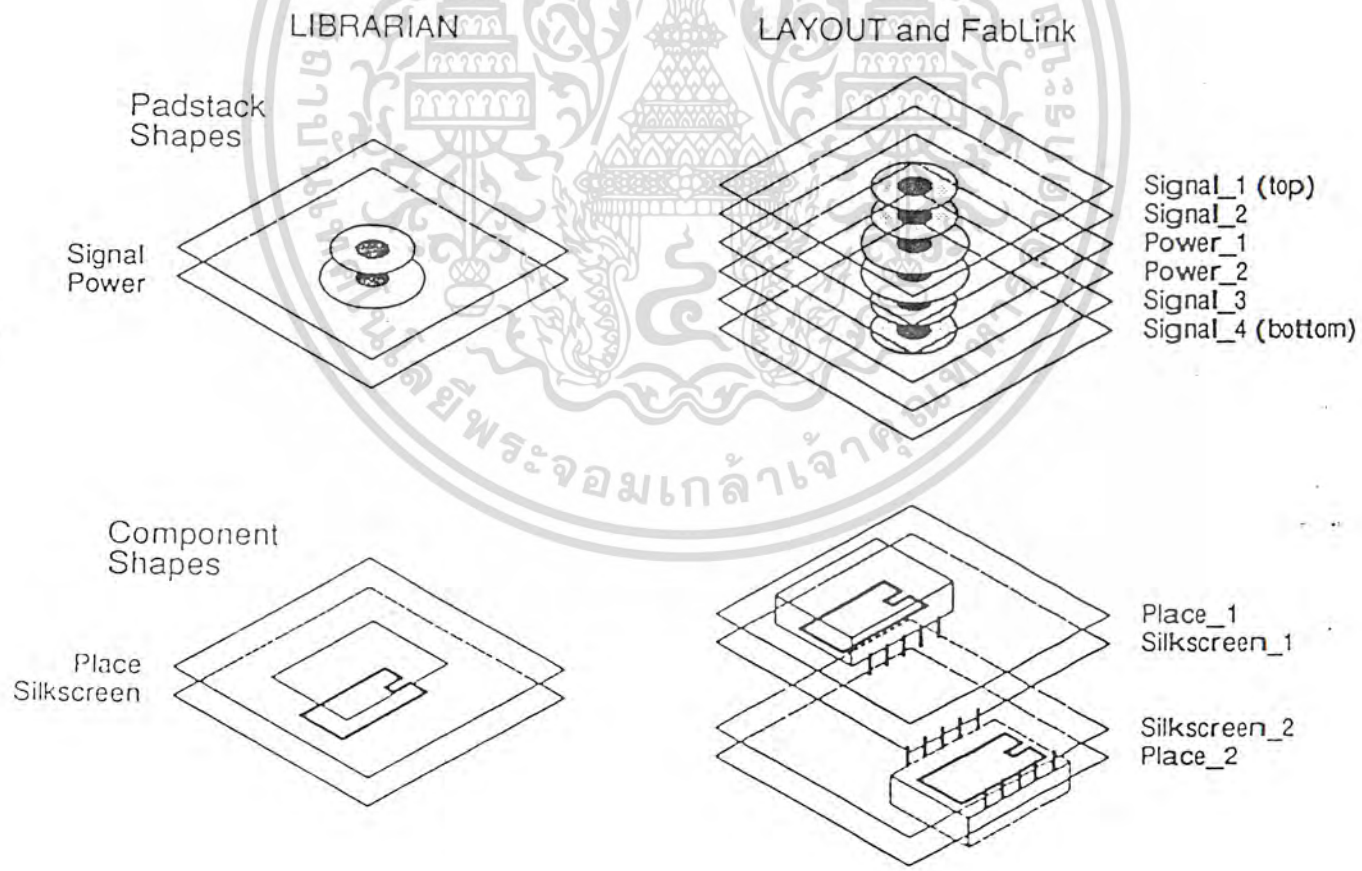
Layer Behavior

Display Characteristics

Color
Fill pattern
Line style
Visibility
Transparency
Selectability

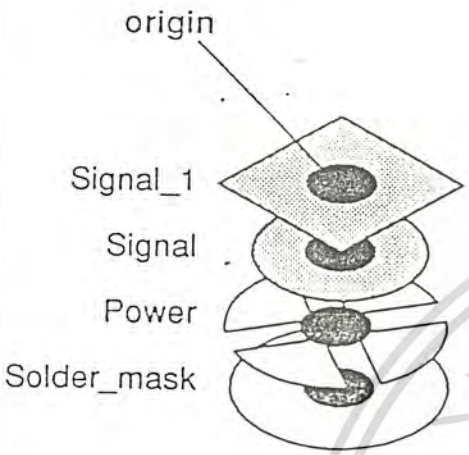


Layer Mapping



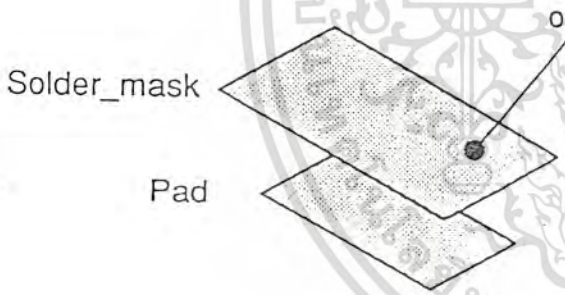
Padstack Geometries

Through Pin Padstack

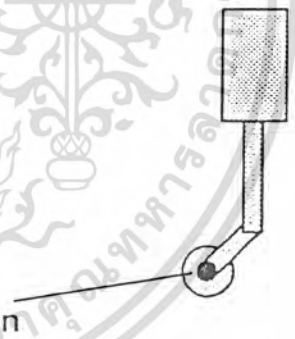



```
create pin pin1_pad
attr 'TERMINAL_THRUHOLE_DEFINITION' ''
circ SIGNAL 0.0 0.0 0.06 0.0
poly SIGNAL_1 -0.03 -0.03 0.03 -0.03 0.03 0.03 -0.03 0.03
circ POWER 0.0 0.0 0.08 0.0
circ SOLDER_MASK_1 0.0 0.0 0.08 0.0
circ SOLDER_MASK_2 0.0 0.0 0.08 0.0
attr 'TERMINAL_DRILL_SIZE' '' -scale 0.028
```

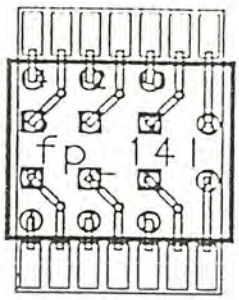
Surface Mount Padstack

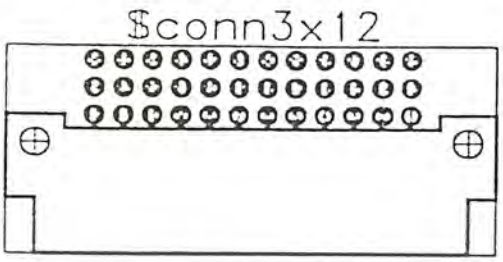


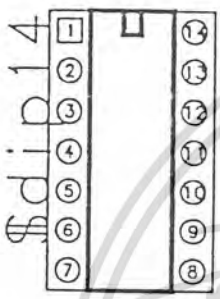
Complex Pad Shape

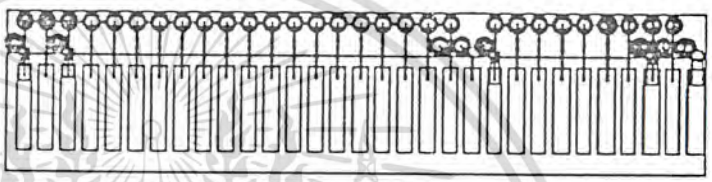


Component Geometries



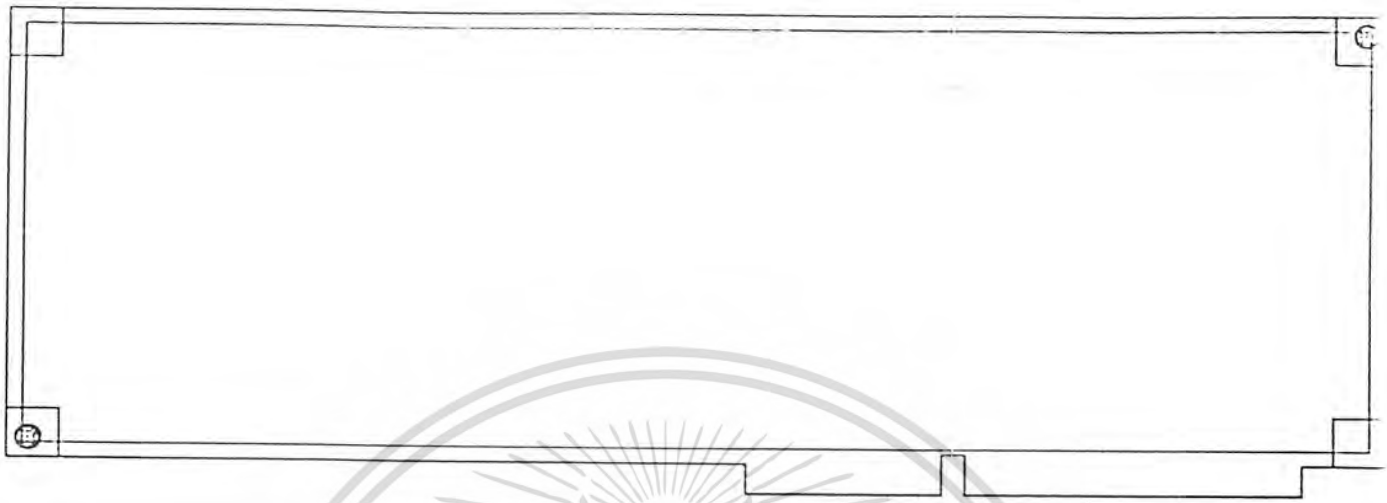






```
create component $dip14
attr 'COMPONENT_PLACEMENT_OUTLINE' "  -scale 0.35 0.05 -0.05 ...
attr 'COMPONENT_PIN_DEFINITION' '14'  -scale 0.3 0.0
attr 'COMPONENT_PIN_DEFINITION' '13'  -scale 0.3 -0.1
attr 'COMPONENT_PIN_DEFINITION' '12'  -scale 0.3 -0.2
attr 'COMPONENT_PIN_DEFINITION' '11'  -scale 0.3 -0.3
attr 'COMPONENT_PIN_DEFINITION' '10'  -scale 0.3 -0.4
attr 'COMPONENT_PIN_DEFINITION' '9'  -scale 0.3 -0.5
attr 'COMPONENT_PIN_DEFINITION' '8'  -scale 0.3 -0.6
attr 'COMPONENT_PIN_DEFINITION' '7'  -scale 0.0 -0.6
attr 'COMPONENT_PIN_DEFINITION' '6'  -scale 0.0 -0.5
attr 'COMPONENT_PIN_DEFINITION' '5'  -scale 0.0 -0.4
attr 'COMPONENT_PIN_DEFINITION' '4'  -scale 0.0 -0.3
attr 'COMPONENT_PIN_DEFINITION' '3'  -scale 0.0 -0.2
attr 'COMPONENT_PIN_DEFINITION' '2'  -scale 0.0 -0.1
attr 'COMPONENT_PIN_DEFINITION' '1'  -scale 0.0 0.0
path SILKSCREEN 0.01 0.05 0.05 0.05 -0.65 0.25 -0.65 0.25 0.05 ...
path SILKSCREEN 0.01 0.17 0.05 0.17 -0.01 0.13 -0.01 0.13 0.05
text SILKSCREEN "^$ref" -0.05 0.05 0.1 BR 90 1.00 0.01 "std"
```

Board Geometries



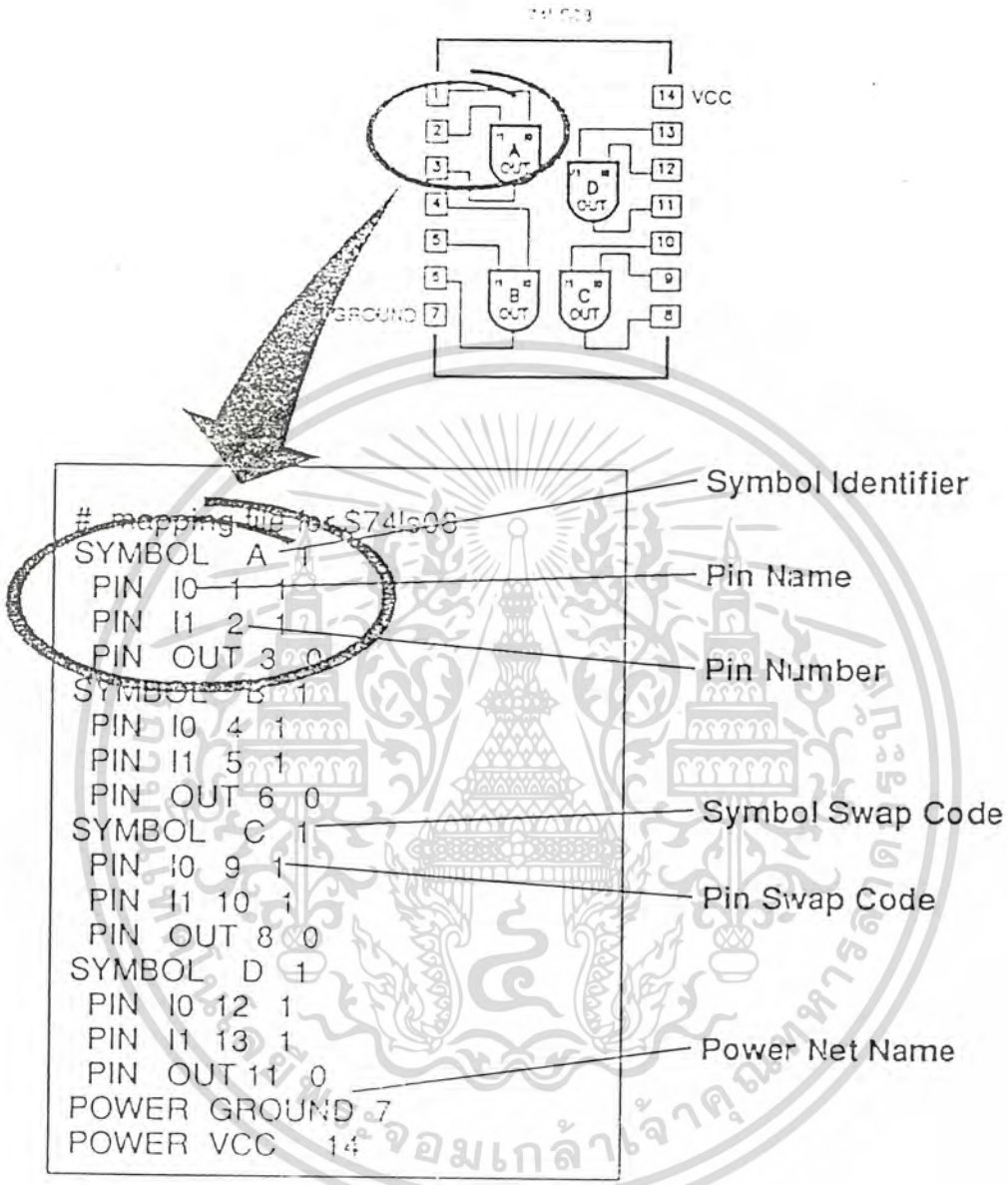
Shapes

```
create board BOARD
attr 'BOARD_DEFINITION_IDENTIFIER' '' 0 0
path BOARD_OUTLINE 0.001 8.975 0.0 7.15 0.0
attr 'BOARD_ROUTING_OUTLINE' '' -scale 0.15 0.45 13.1 0.45 ...
attr 'BOARD_PLACEMENT_OUTLINE' '' -scale 0.15 0.45 13.1 0.45 ...
attr 'BOARD_PLACEMENT_KEEPOUT' '' -scale 12.75 4.8 13.25 4.8 ...
attr 'BOARD_PLACEMENT_KEEPOUT' '' -scale 0.0 4.8 0.5 4.8 ...
attr 'BOARD_PLACEMENT_KEEPOUT' '' -scale 0.0 0.8 0.5 0.8 ...
attr 'BOARD_PLACEMENT_KEEPOUT' '' -scale 12.75 0.8 13.25 ...
circ POWER 13.05 4.6 0.255 0.0
circ POWER 0.2 0.5 0.255 0.0
circ SOLDER_MASK 13.05 4.6 0.2 0.0
circ SOLDER_MASK 0.2 0.5 0.2 0.0
attr 'DRILL_DEFINITION_UNPLATED' '0.125' -scale 0.2 0.5 ...
attr 'DRILL_DEFINITION_UNPLATED' '0.125' -scale 13.05 4.6 ...
```

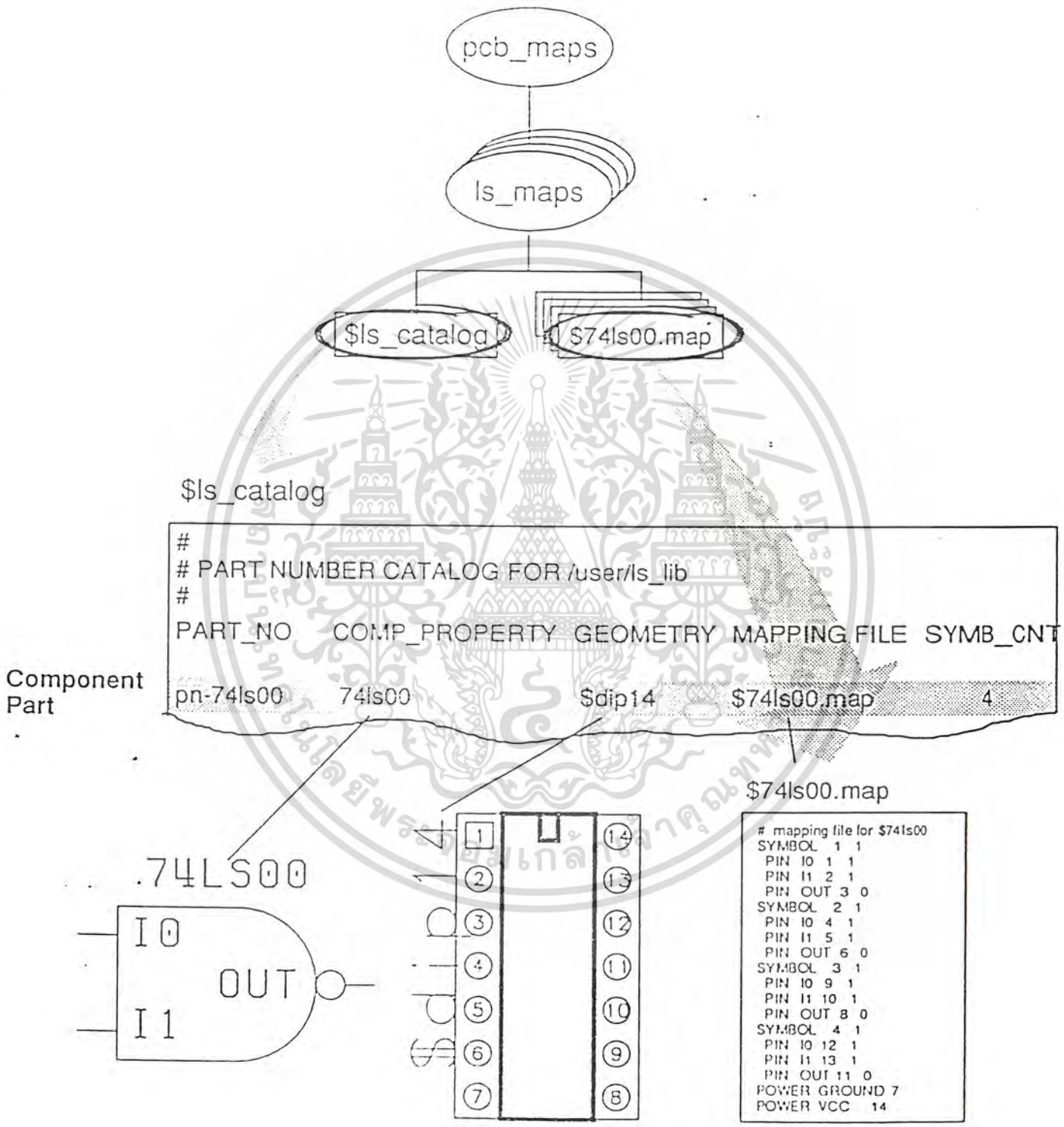
Design Rules

```
attr 'BOARD_PLACEMENT_GRID' '' -scale 0.05
attr 'DEFAULT_TRACE_SIZE' '' -scale 0.01
attr 'DEFAULT_VIA_SIZE' '' -scale 0.05
attr 'DEFAULT_PAD_SIZE' '' -scale 0.05
attr 'BOARD_DEFAULT_PADSTACK' 'PAD_050'
attr 'DEFAULT_ROUTING_CLEARANCE' '' -scale 0.01
attr 'BOARD_PLACEMENT_CLEARANCE' '' -scale 0.0
attr 'POWER_NET_NAMES' 'VCC,GROUND' 0 0
attr 'BOARD_ROUTING_LAYERS' '' 2
attr 'BOARD_ROUTING_DIRECTION' 'HORIZONTAL' 1
attr 'BOARD_ROUTING_DIRECTION' 'VERTICAL' 2
attr 'ORTHOGONAL_PLACEMENT_ONLY' '' 0 0
attr 'TJUNCTIONS_ALLOWED' 'yes' 0 0
attr 'DIAGONAL_ROUTING_ALLOWED' 'yes' 0 0
attr 'DEFAULT_PADSTACK_CLEARANCE' '' -scale 0.01
```

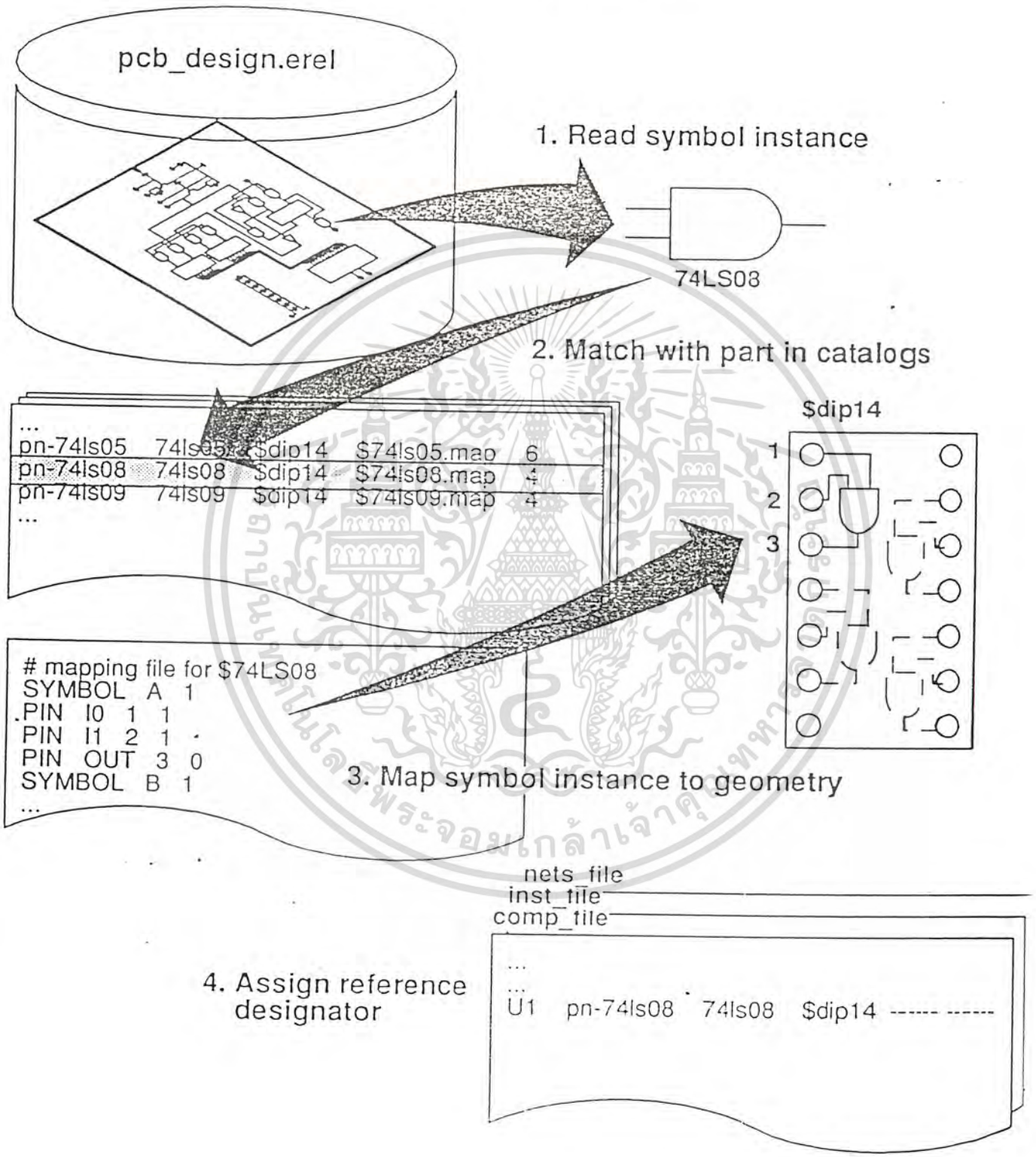
Mapping Files



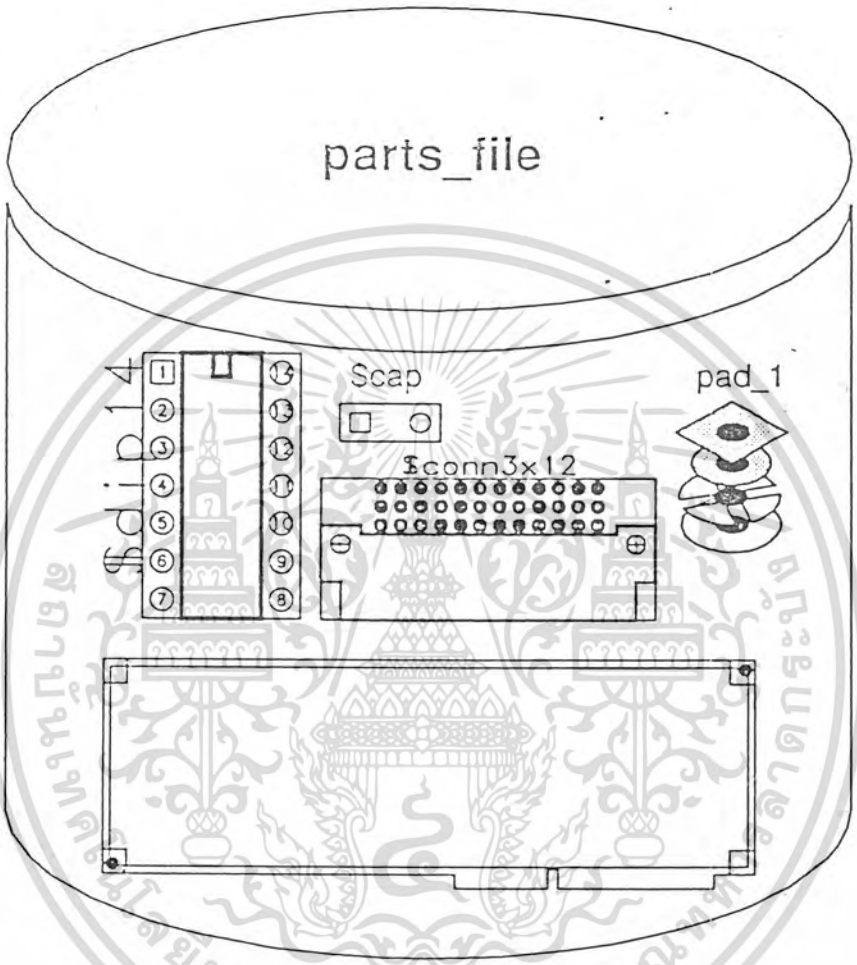
Catalog Files



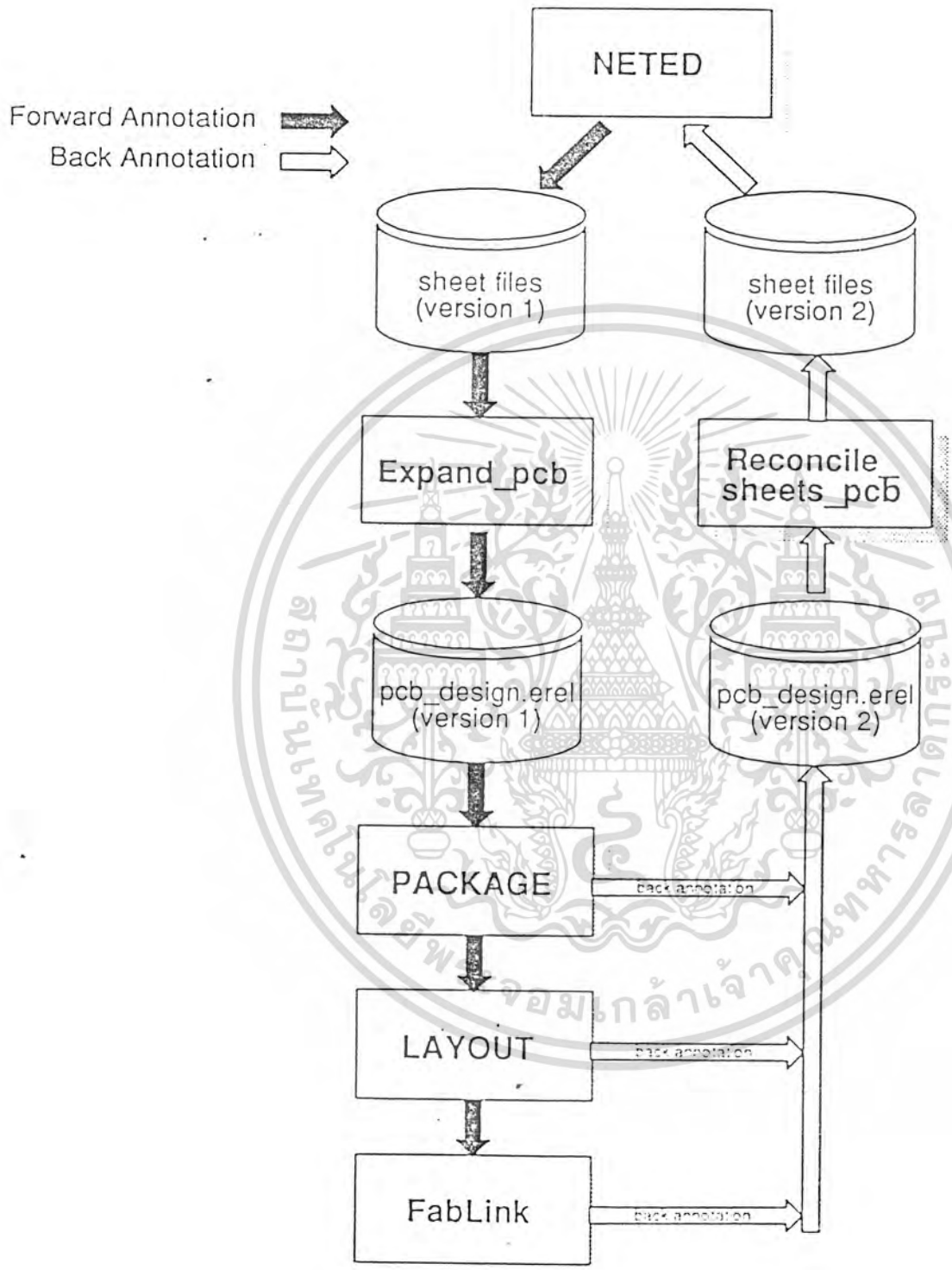
The Build Process



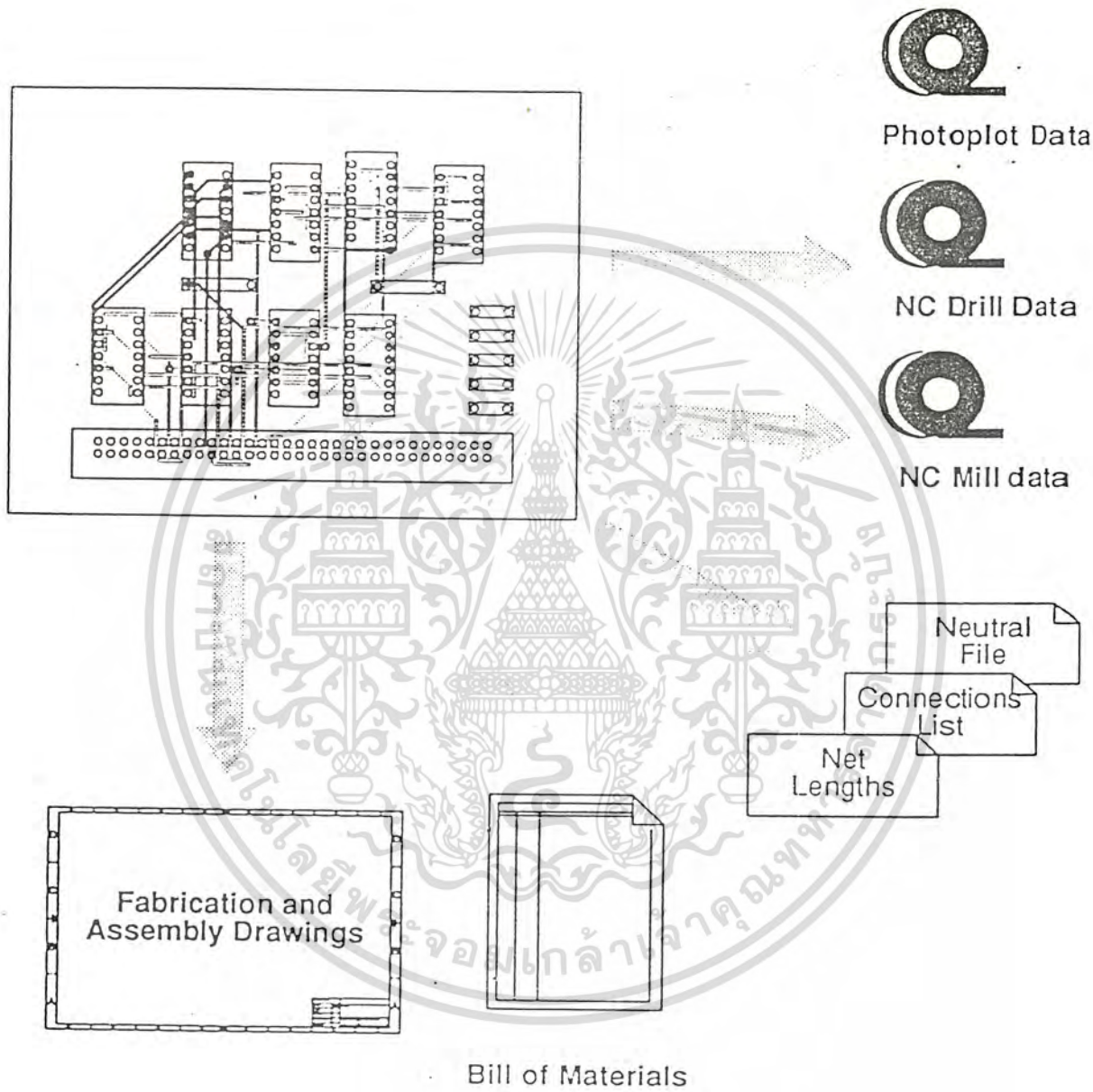
Parts File



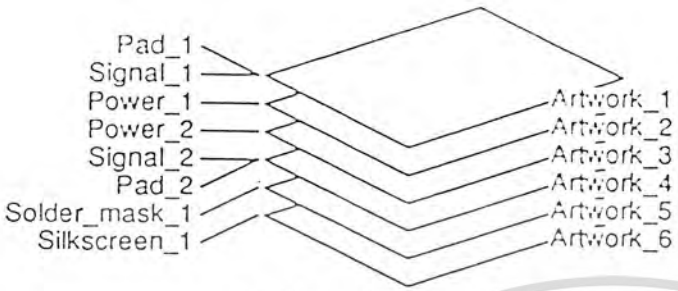
Back Annotation



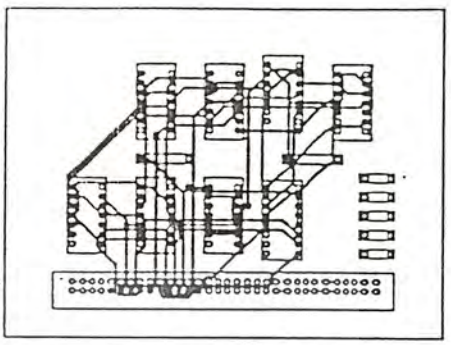
Types of Manufacturing Output



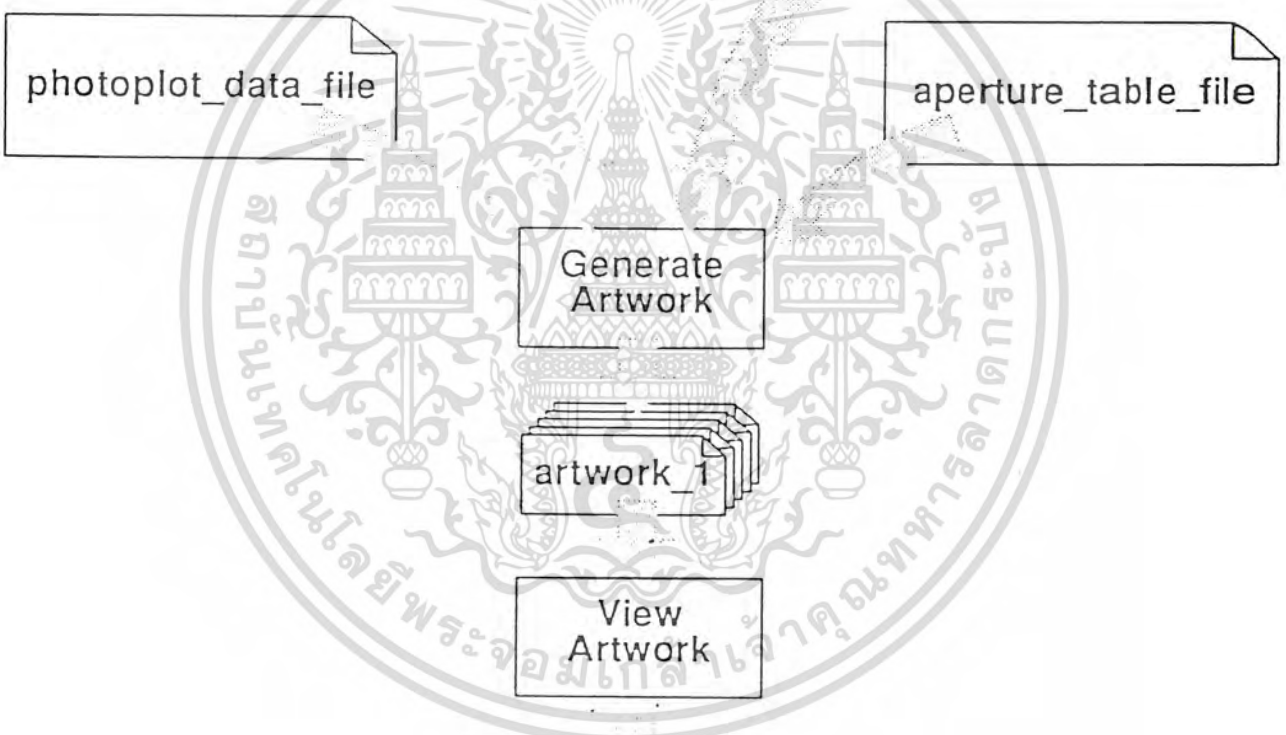
Photoplotter Output

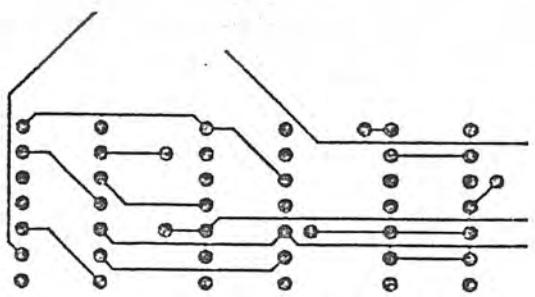


Artwork stackup

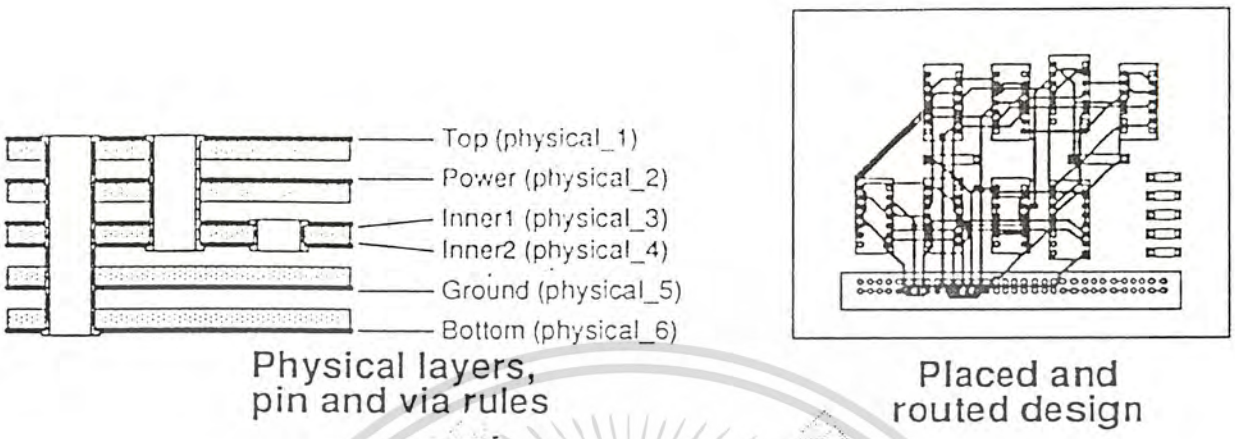


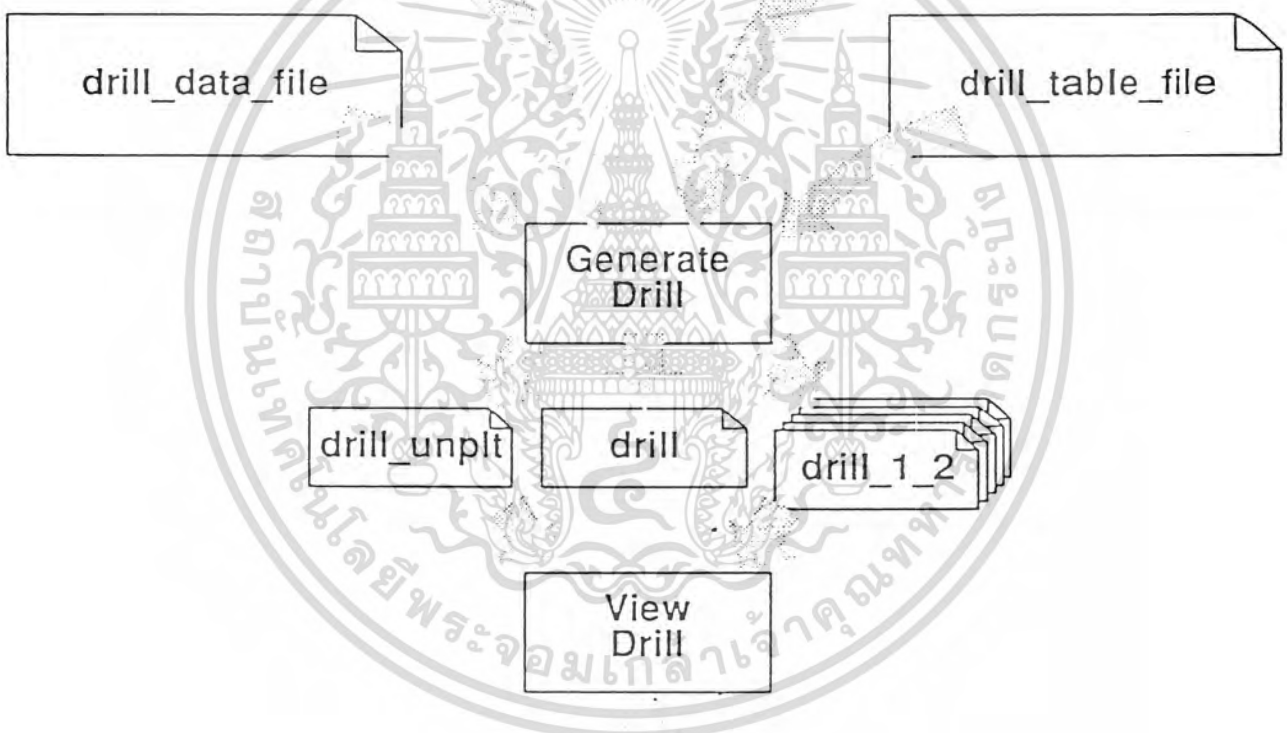
Placed and routed design

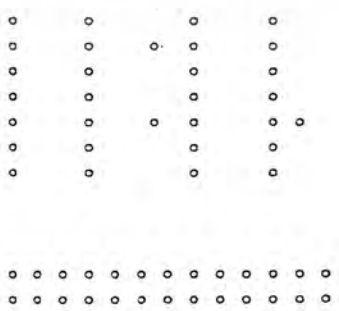




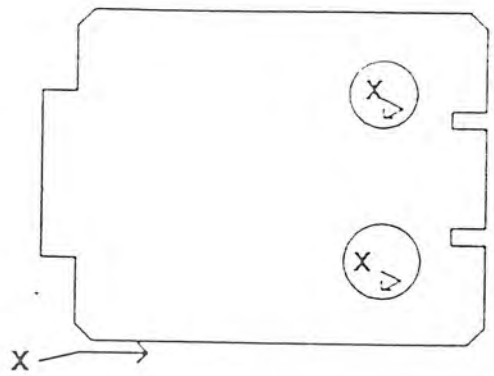
NC Drill Output



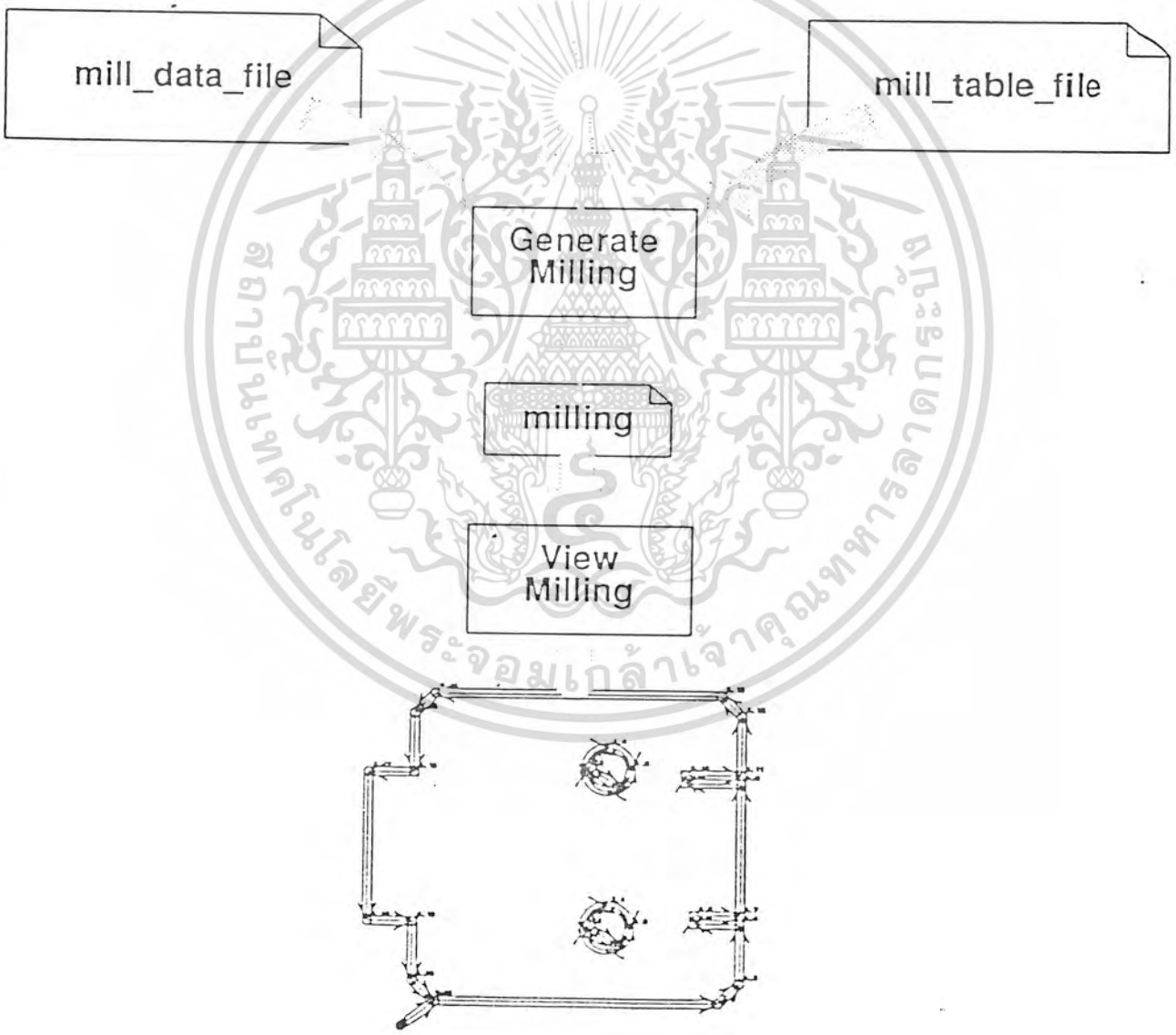




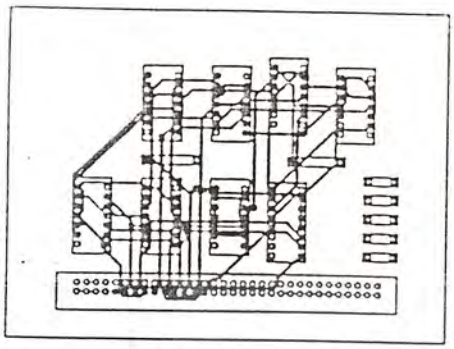
NC Mill Output



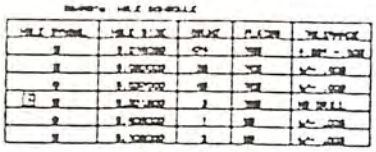
Path specifications



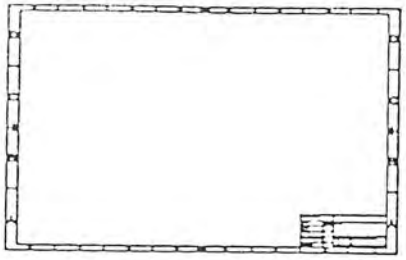
Drawings



Placed and routed design

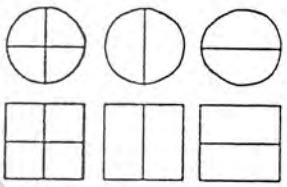


Hole schedule

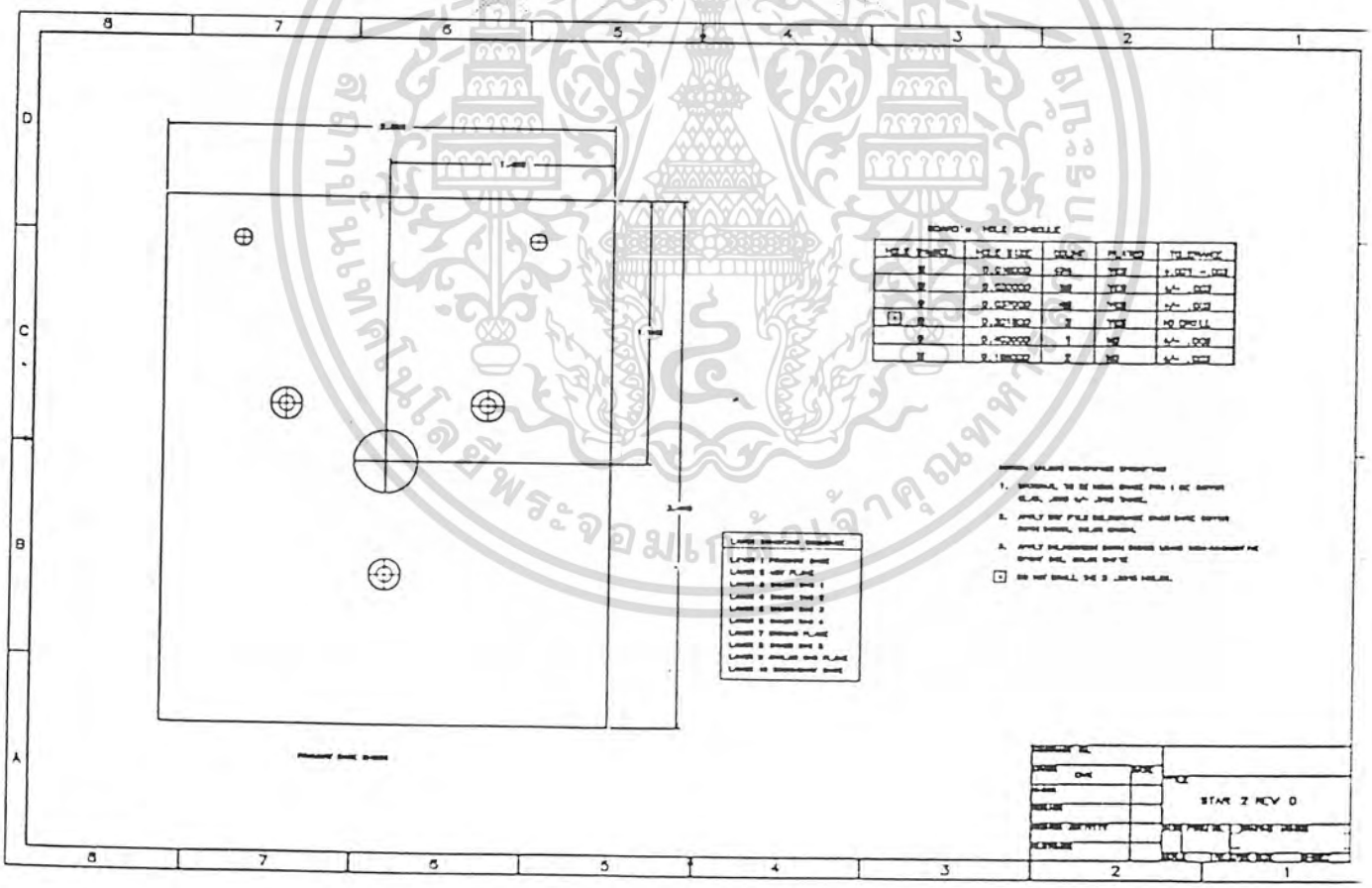


Drawing formats

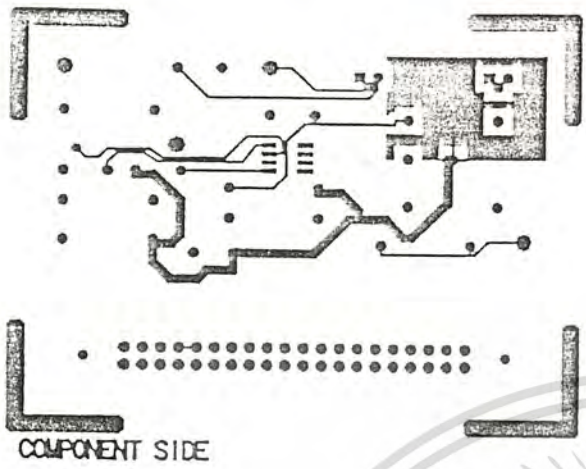
Text and dimensions

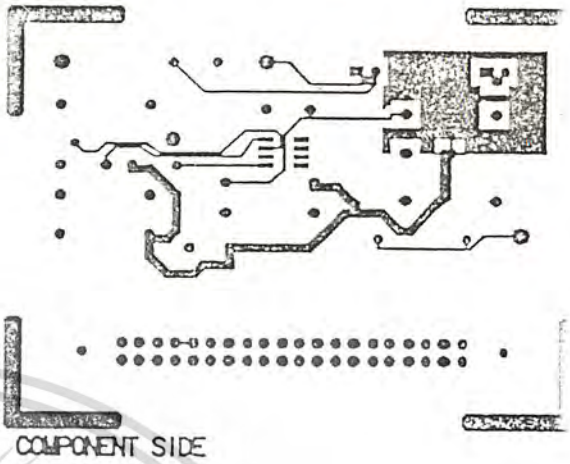


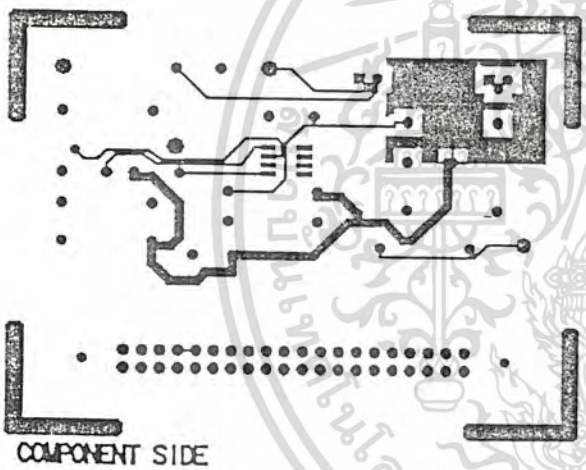
Hole symbols

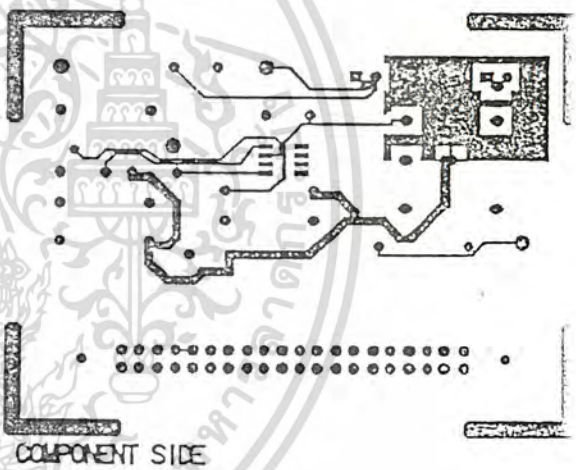


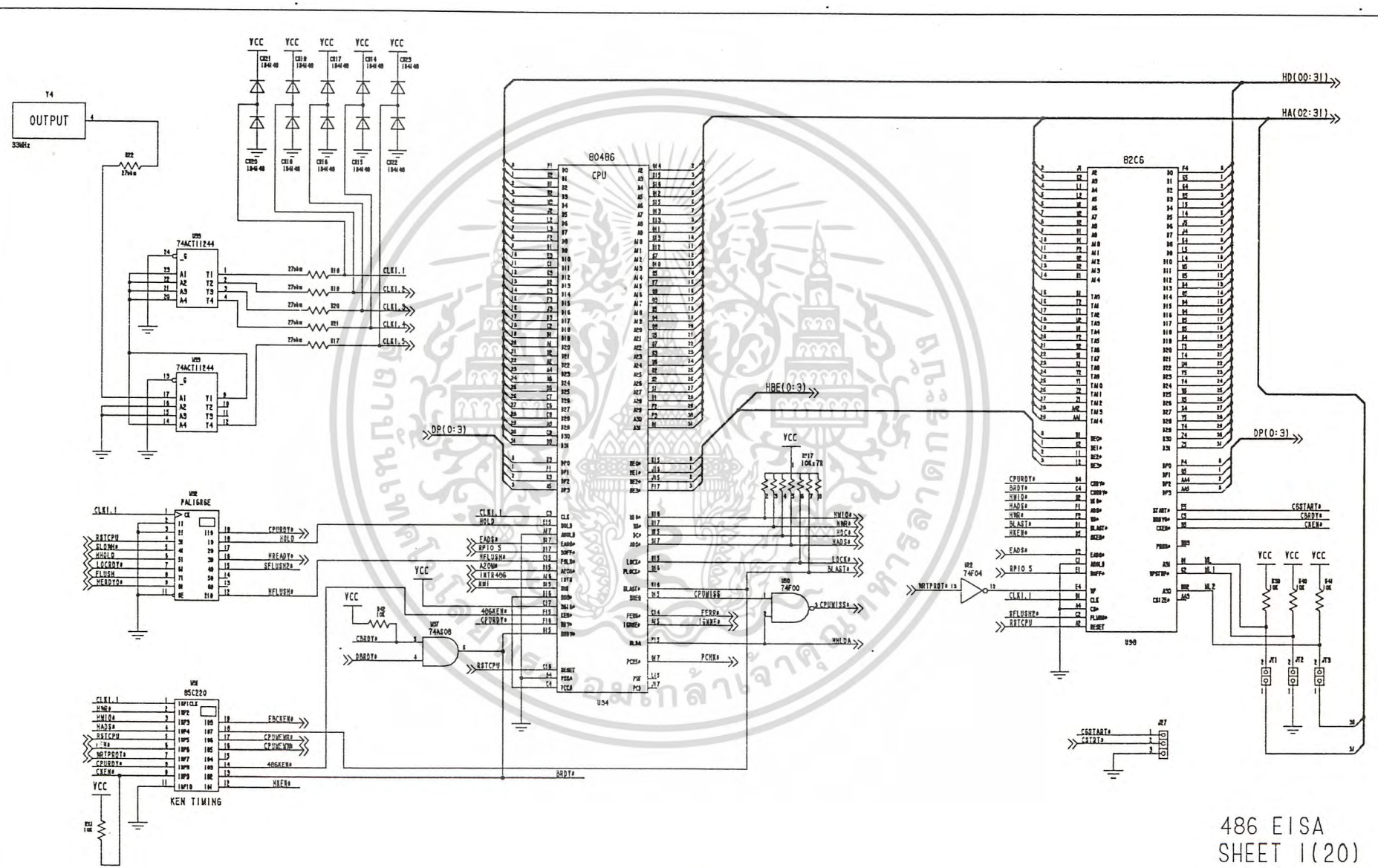
Panelization



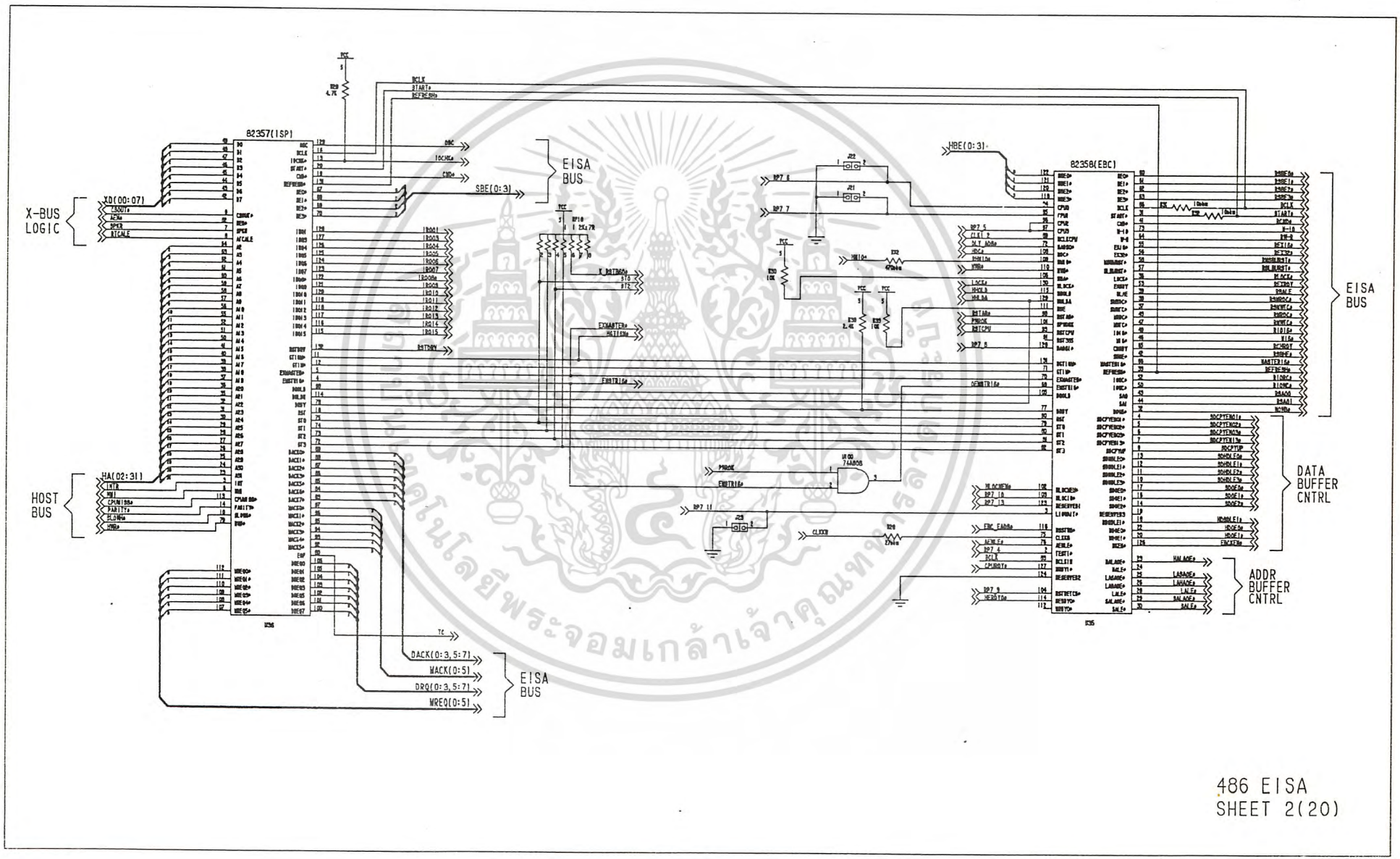




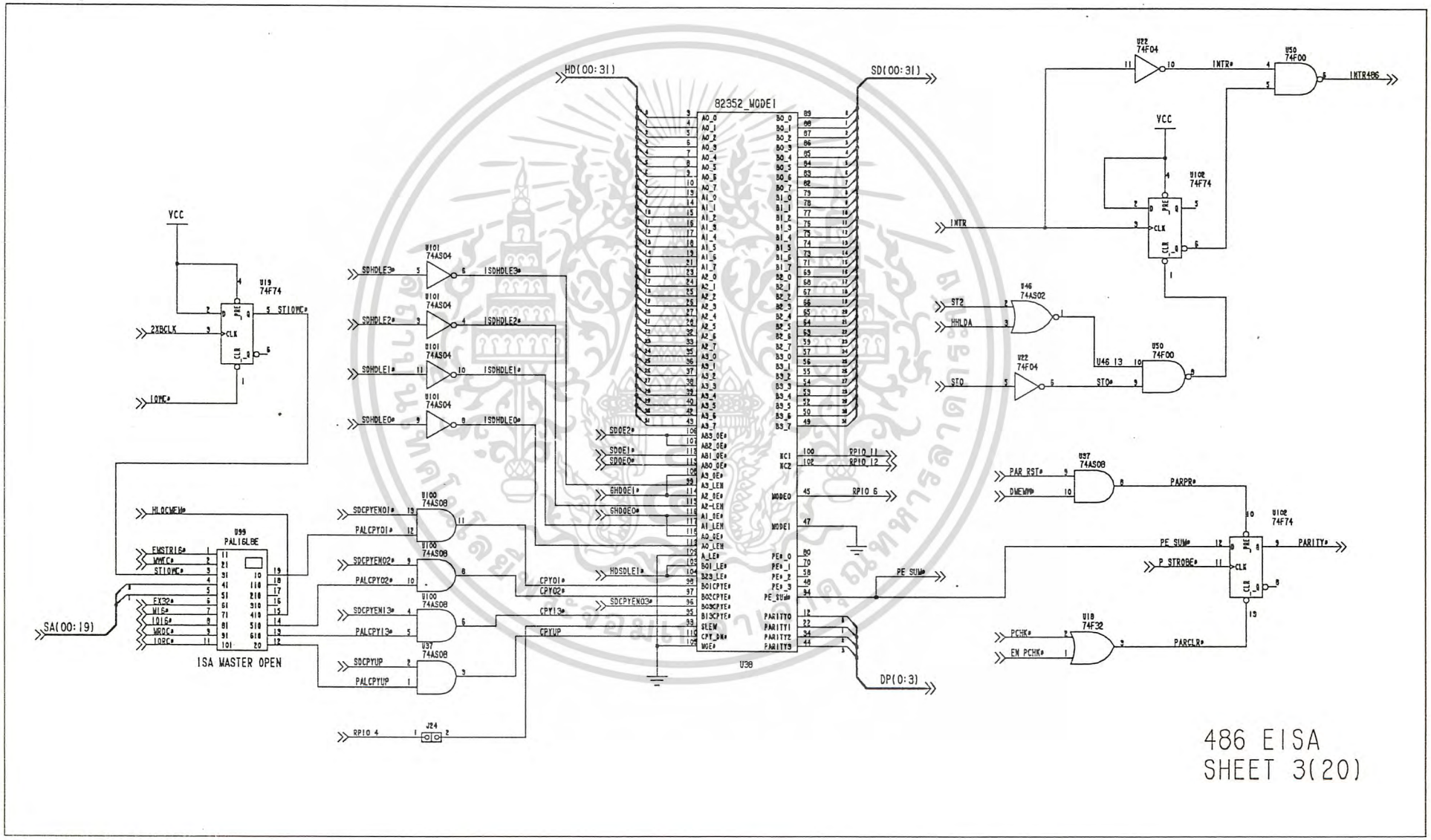




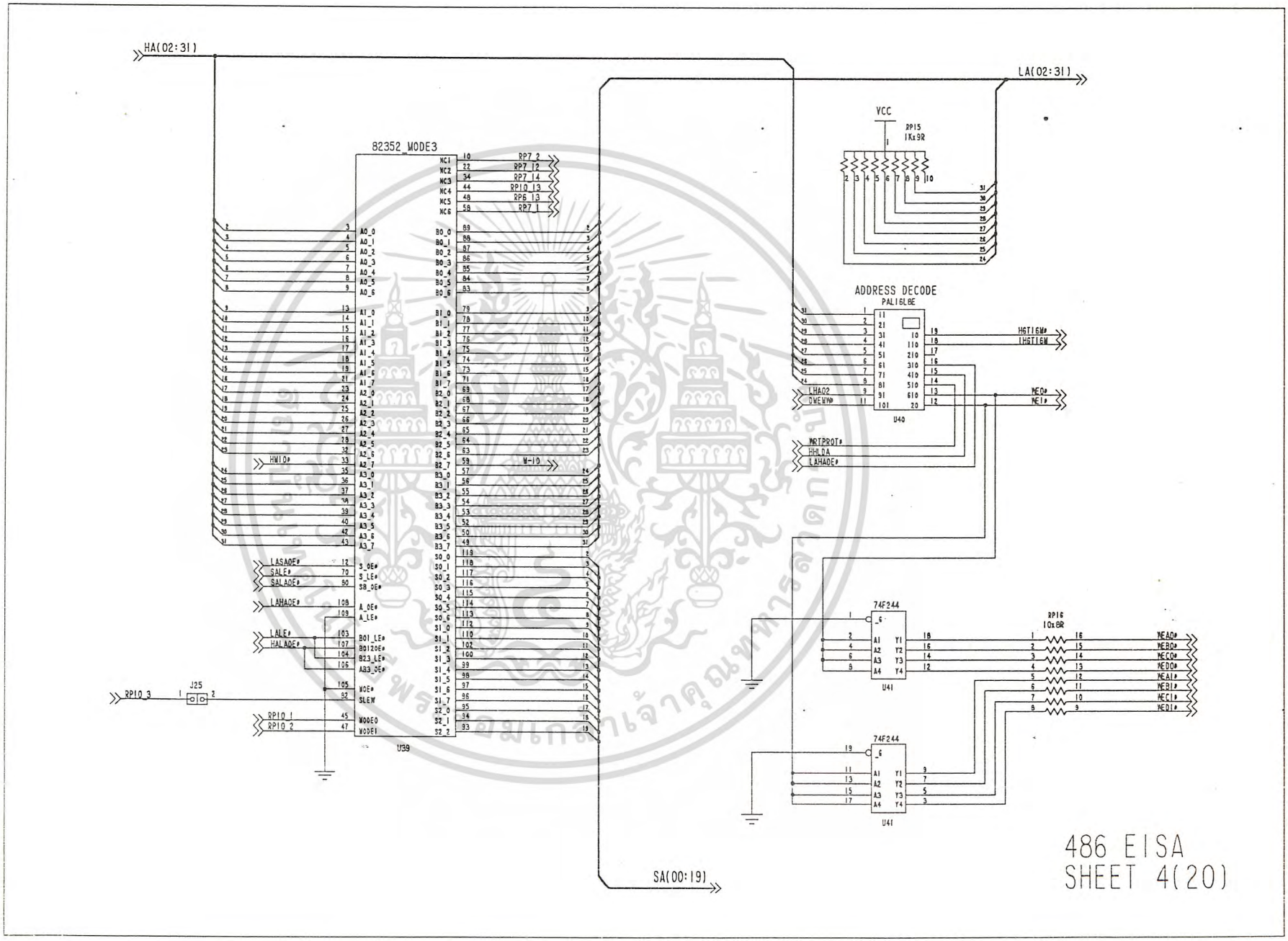
486 EISA
SHEET 1(20)



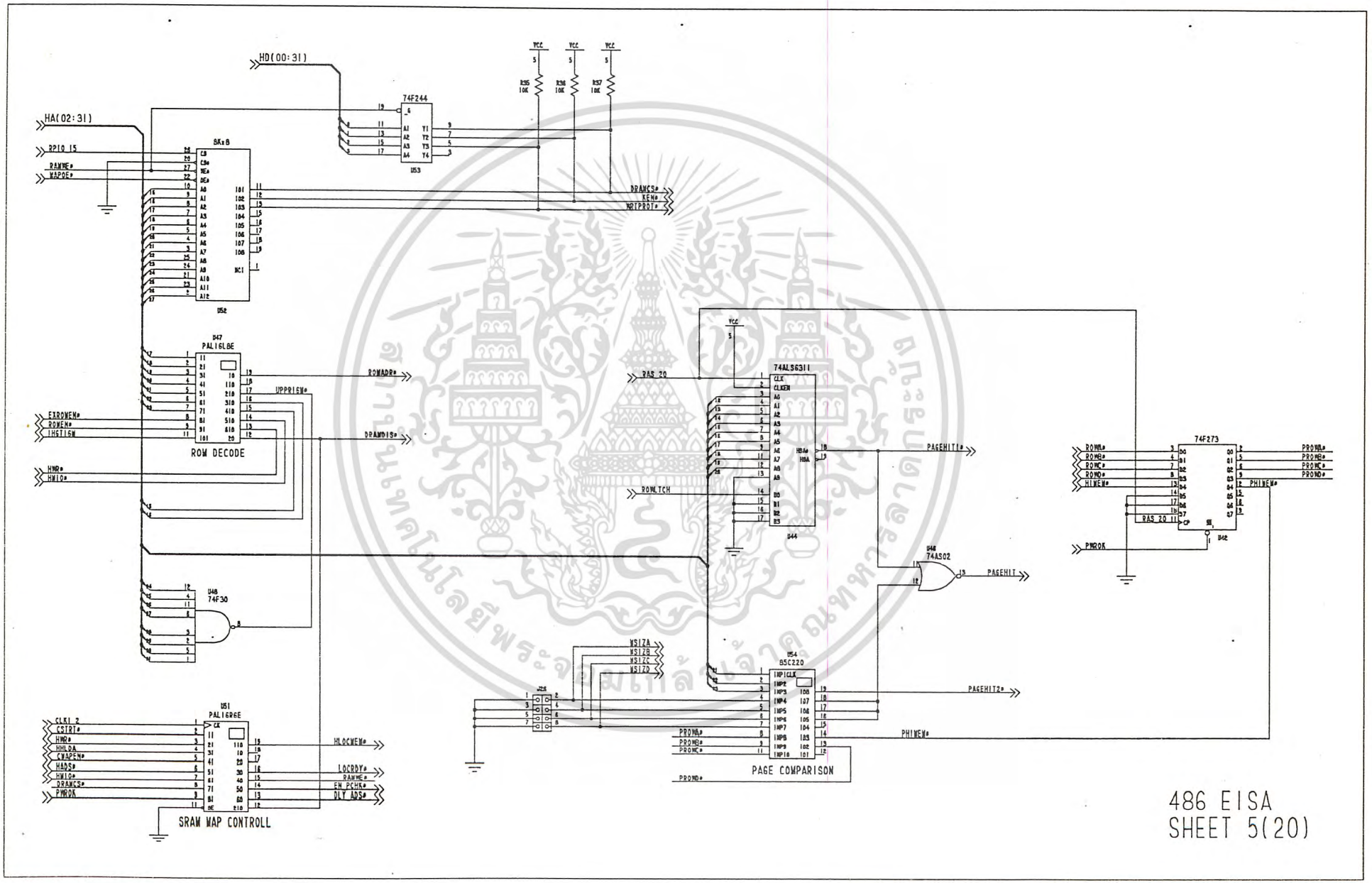
486 EISA
SHEET 2(20)



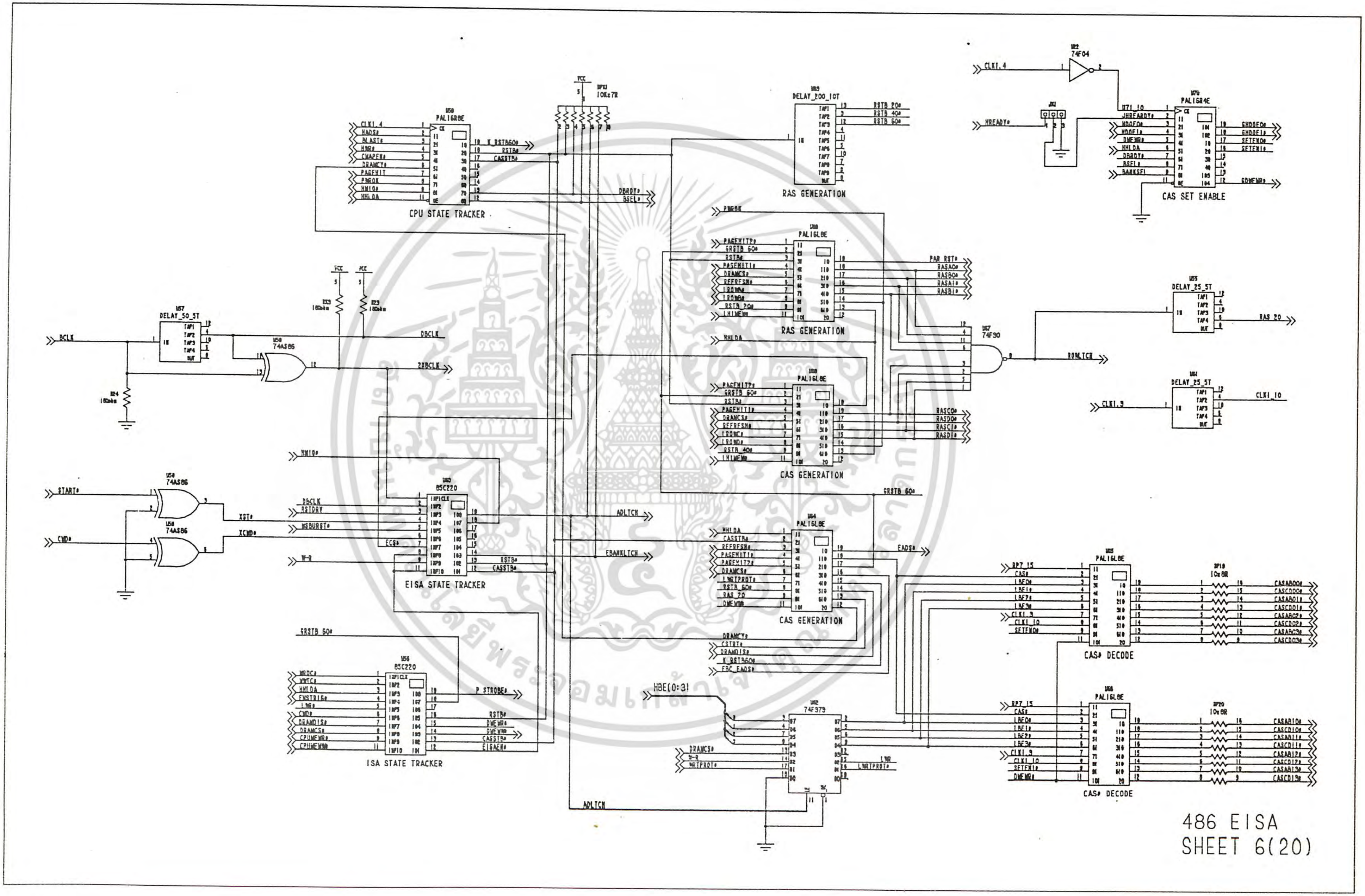
486 EISA
SHEET 3(20)

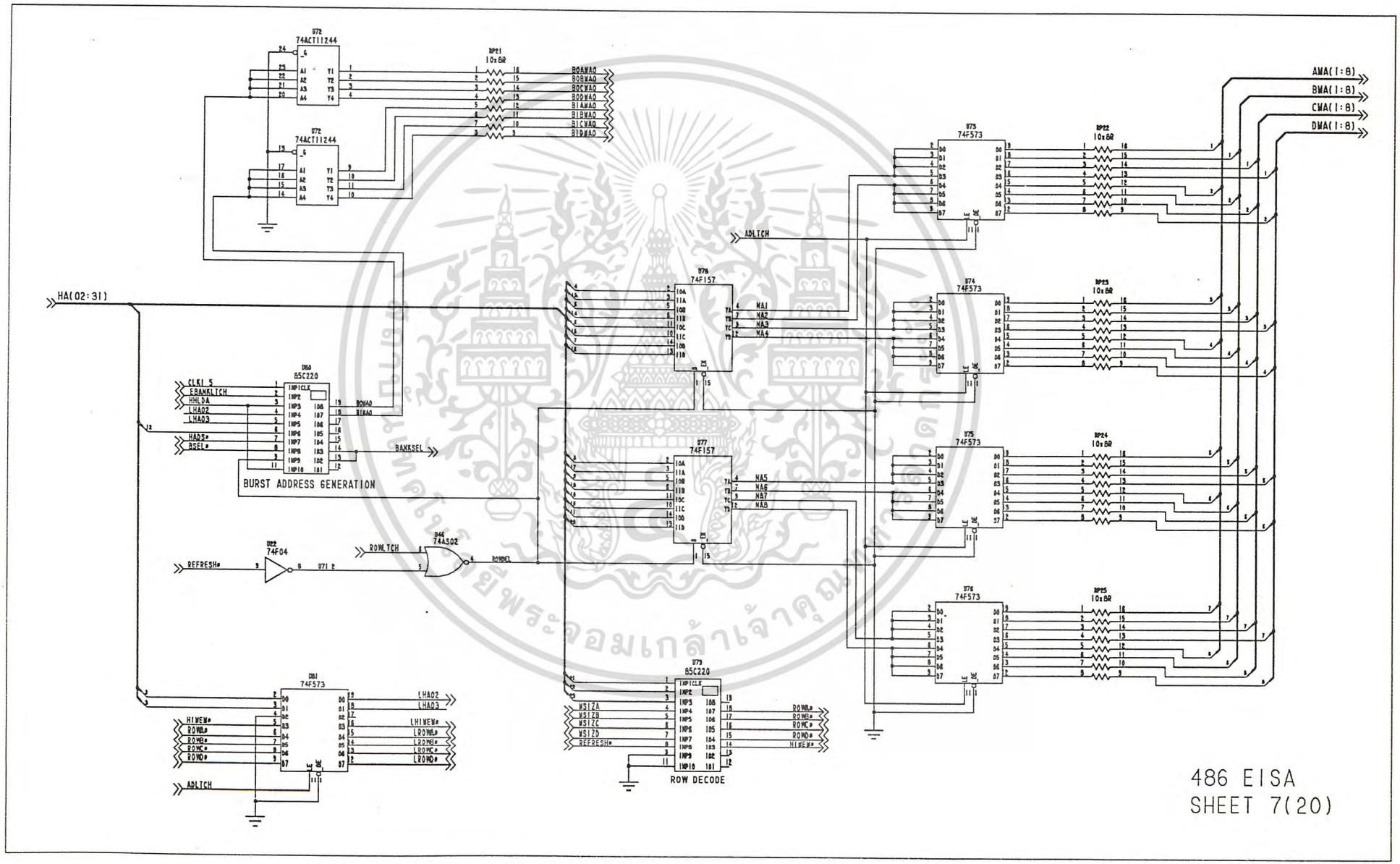


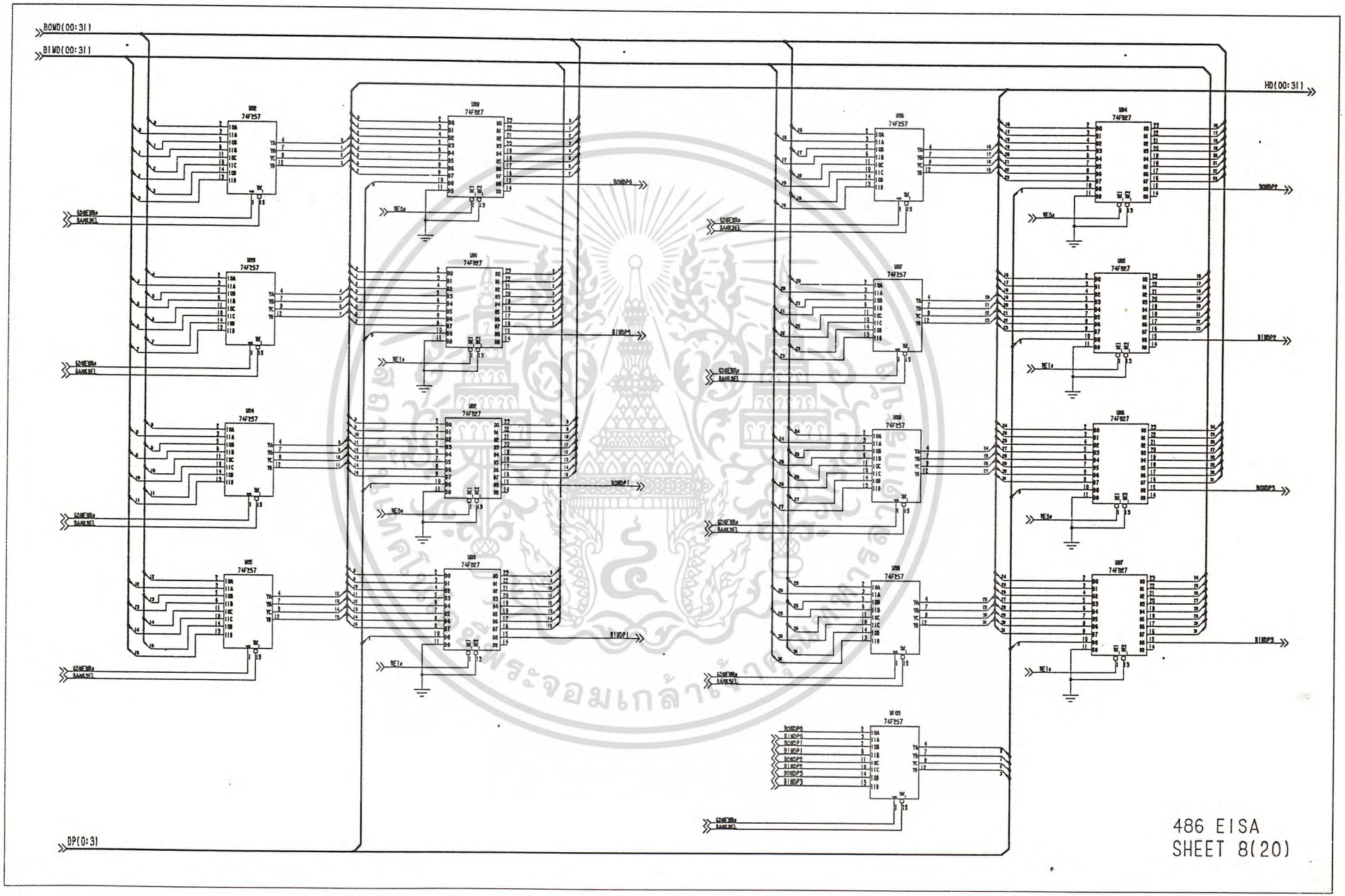
486 EISA
SHEET 4(20)

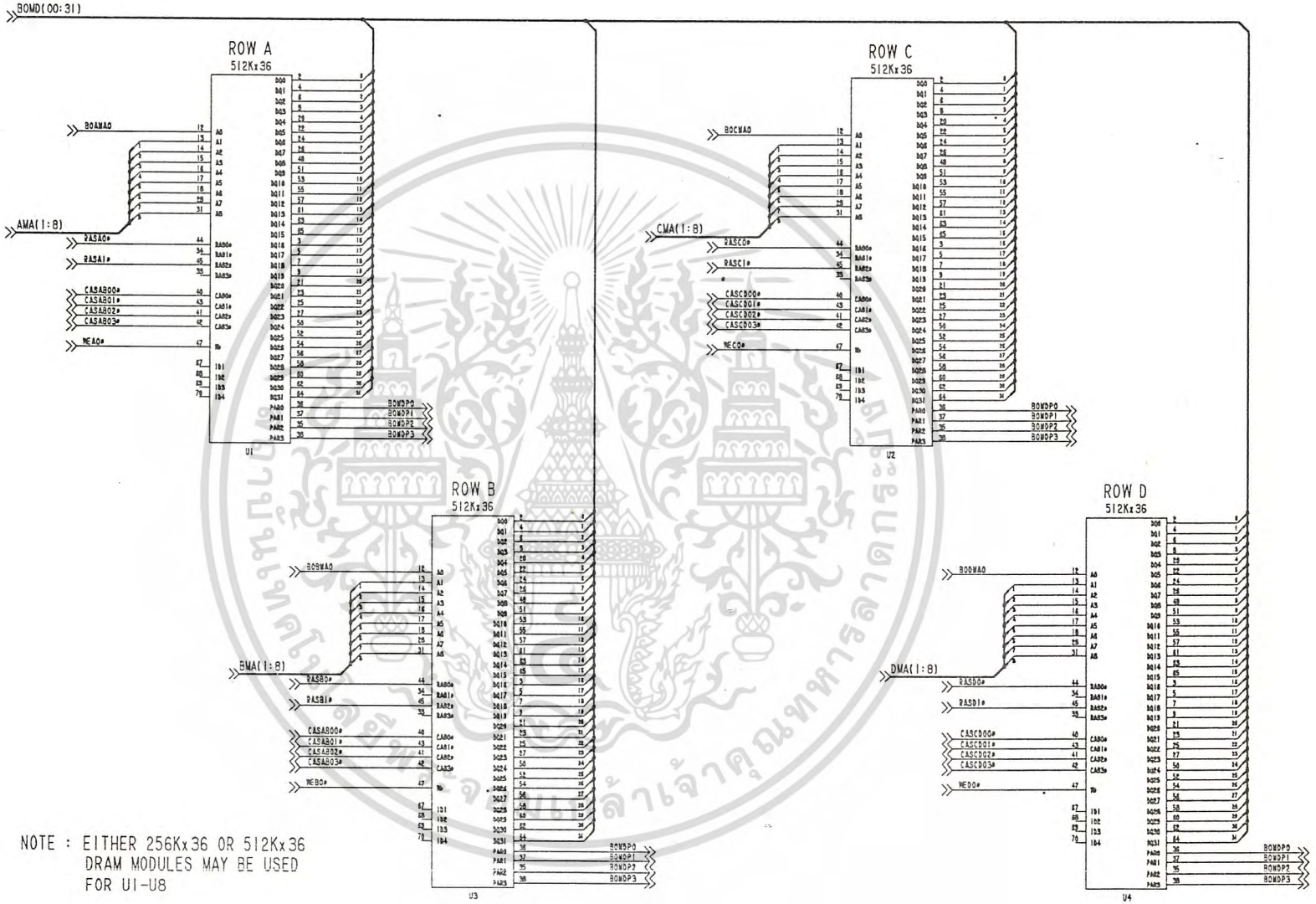


486 EISA
SHEET 5(20)

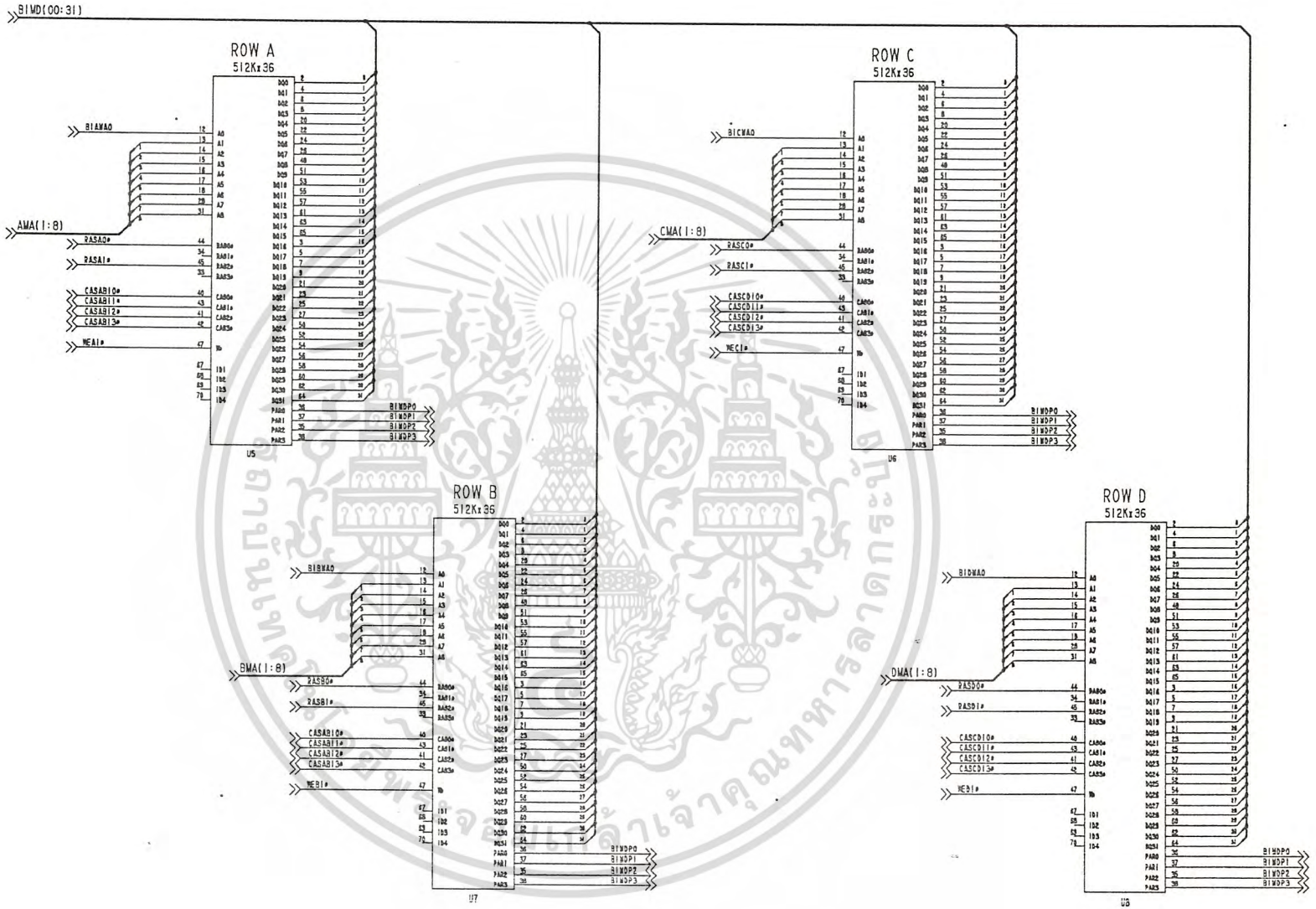

CPU STATE TRACKER
RAS GENERATION
RAS GENERATION
CAS SET ENABLE
CAS GENERATION
CAS GENERATION
EISA STATE TRACKER
ISA STATE TRACKER
CAS# DECODE
CAS# DECODE
486 EISA
SHEET 6(20)


HA(02:31)
AMA(1:8)
BMA(1:8)
CMA(1:8)
DMA(1:8)
ADLTCH
BURST ADDRESS GENERATION
ROW DECODE
REFRESH#
ROWLTCH
ROWSEL
BANKSEL
HIMEM#
ROWL#
ROWH#
ROWC#
ROWD#
LHA02
LHA03
LHIMEM#
LROWL#
LROWH#
LROWC#
LROWD#
74ACT11244
74F573
74F157
85C220
74F04
74AS02
10x8R
486 EISA
SHEET 7(20)

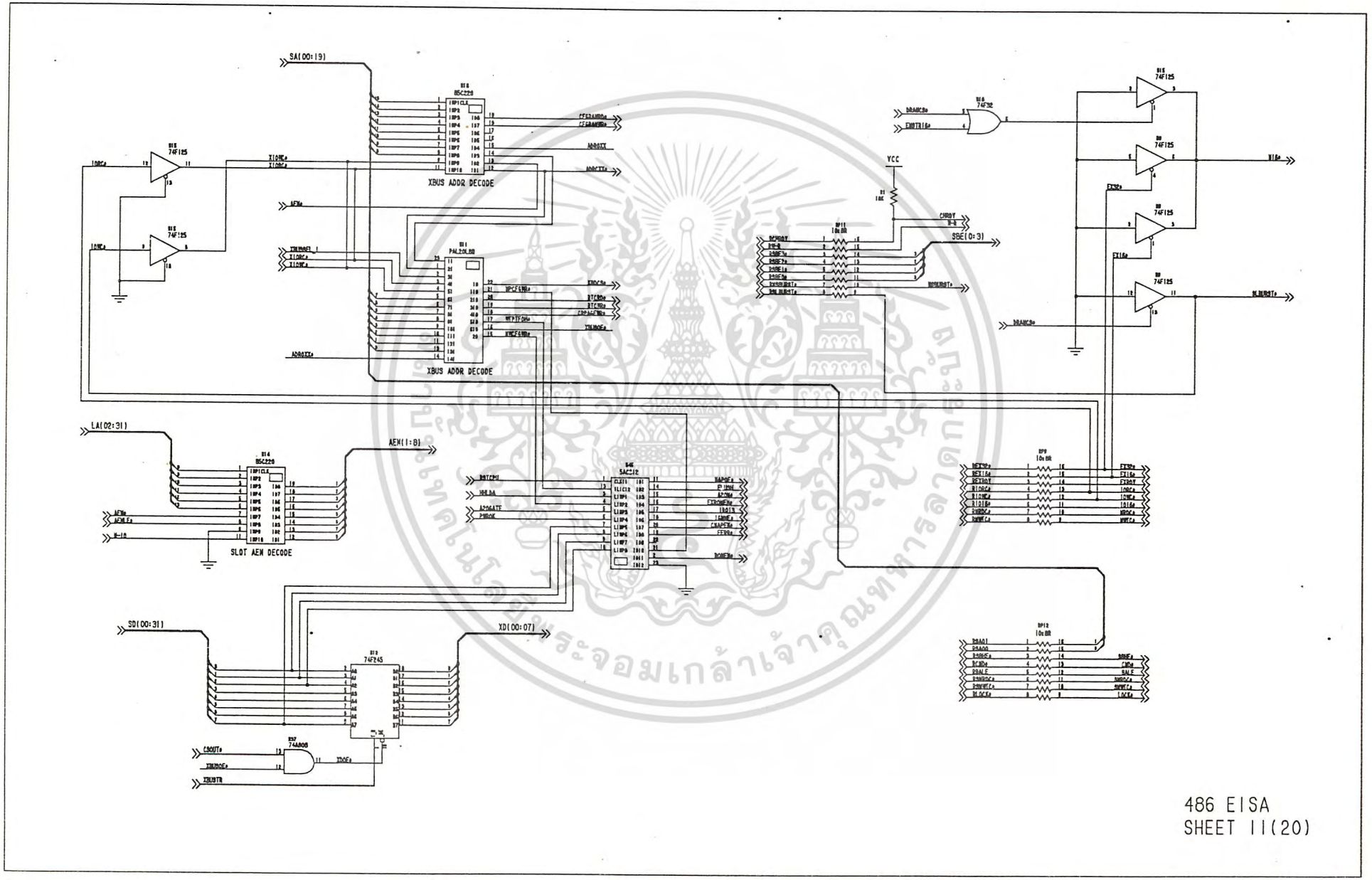

B0WD(00:31)
B1WD(00:31)
HD(00:31)
DP(0:3)
74F257
74F827
486 EISA
SHEET 8(20)

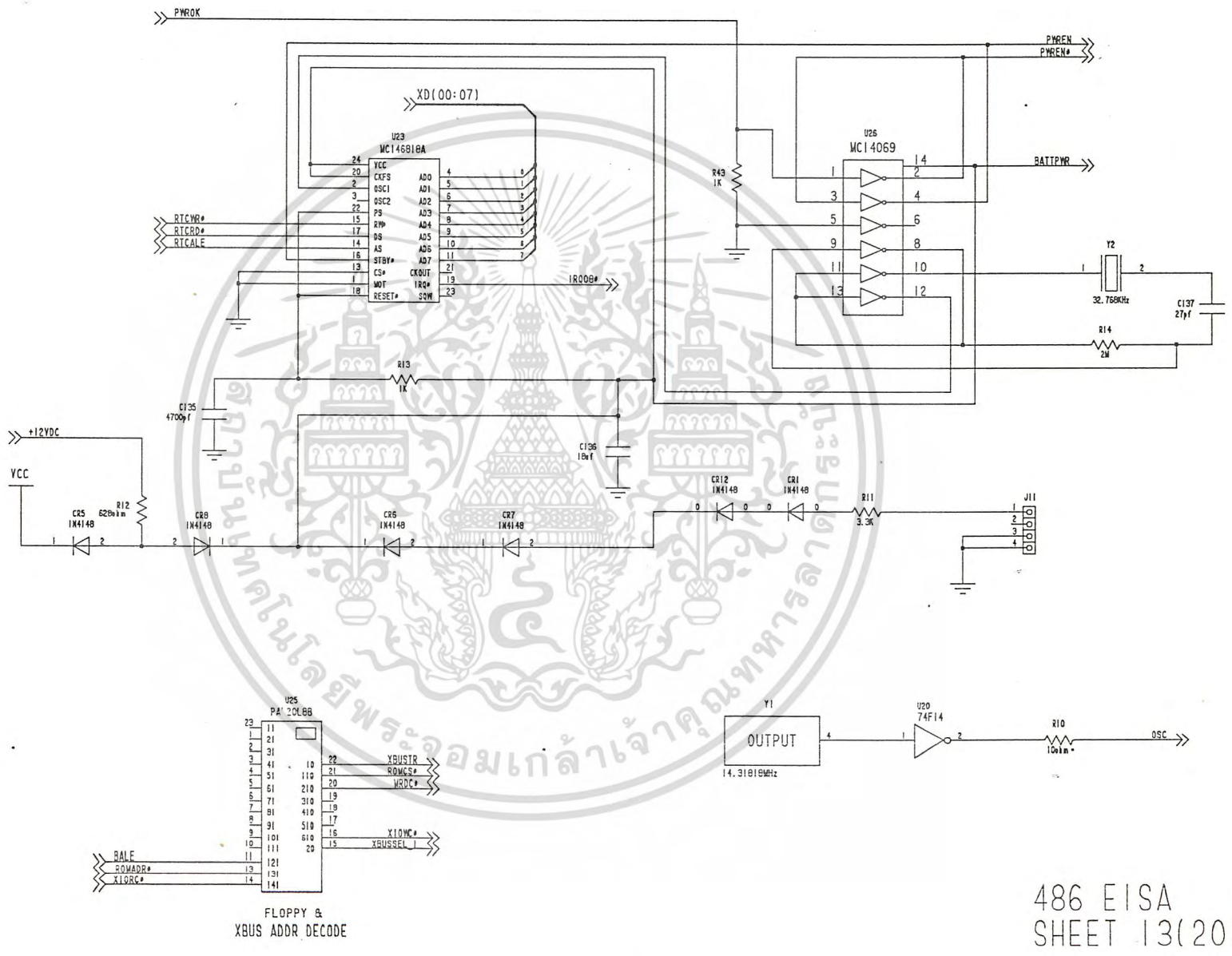


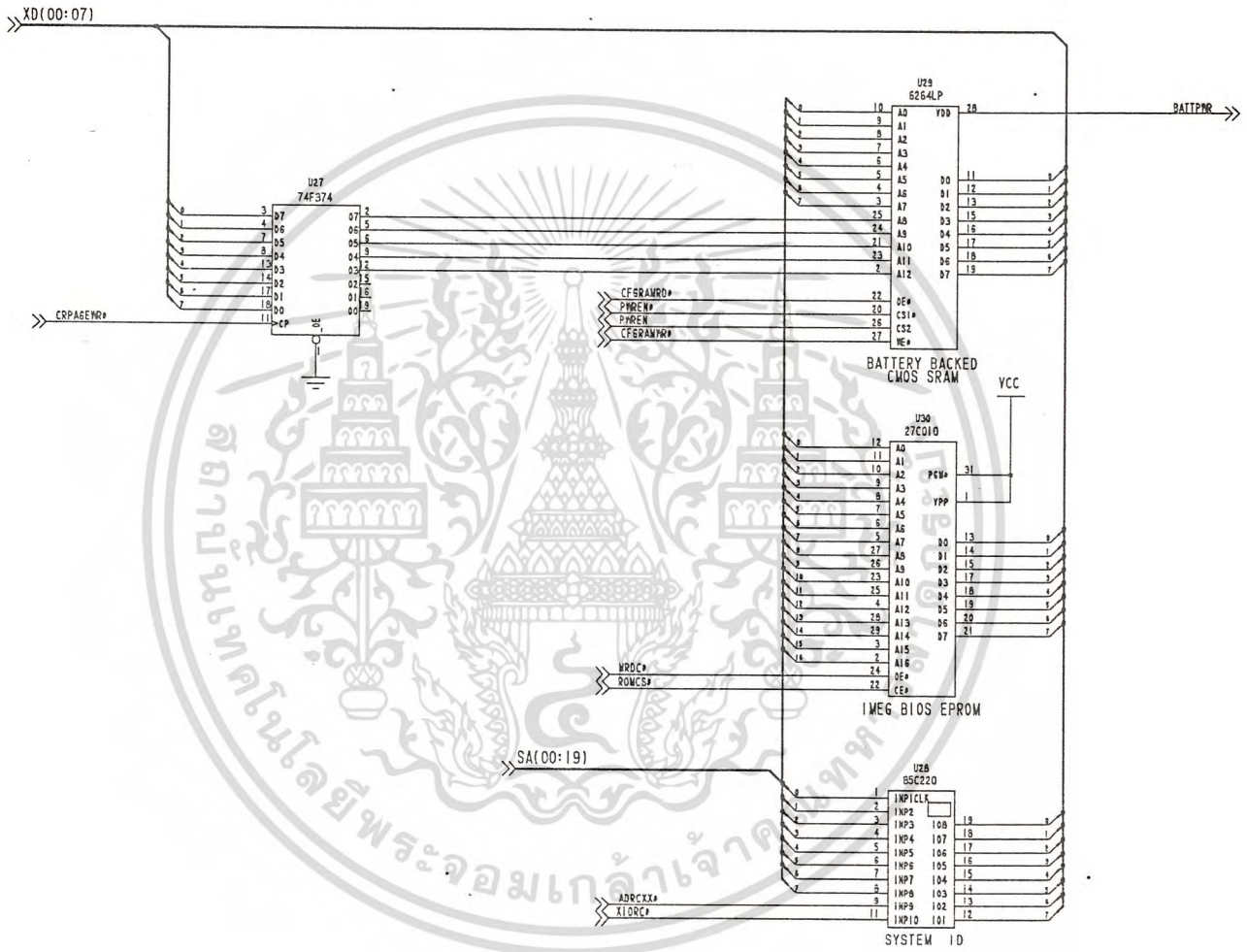
NOTE : EITHER 256Kx36 OR 512Kx36
DRAM MODULES MAY BE USED
FOR U1-U8



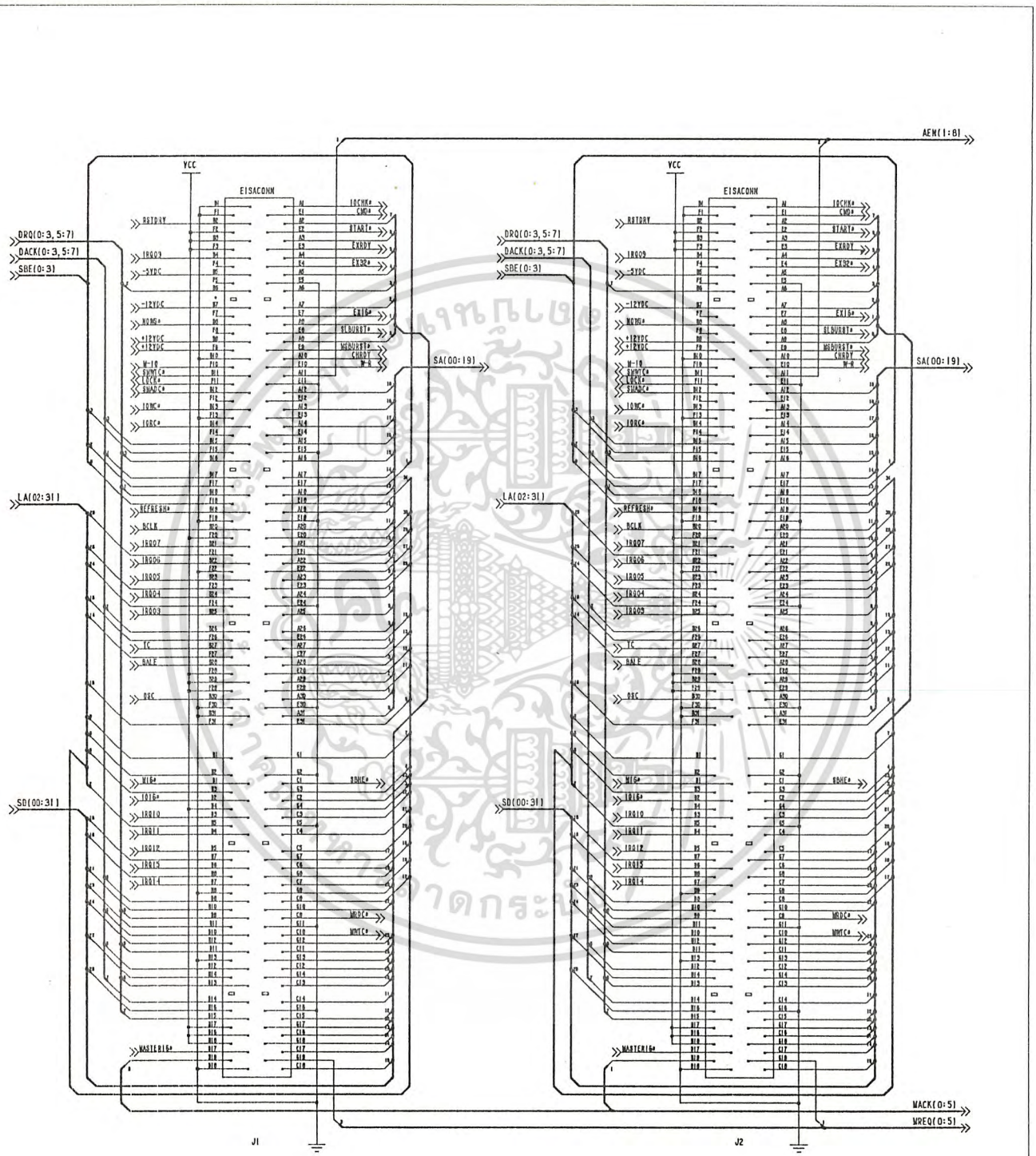
486 EISA
SHEET 10(20)

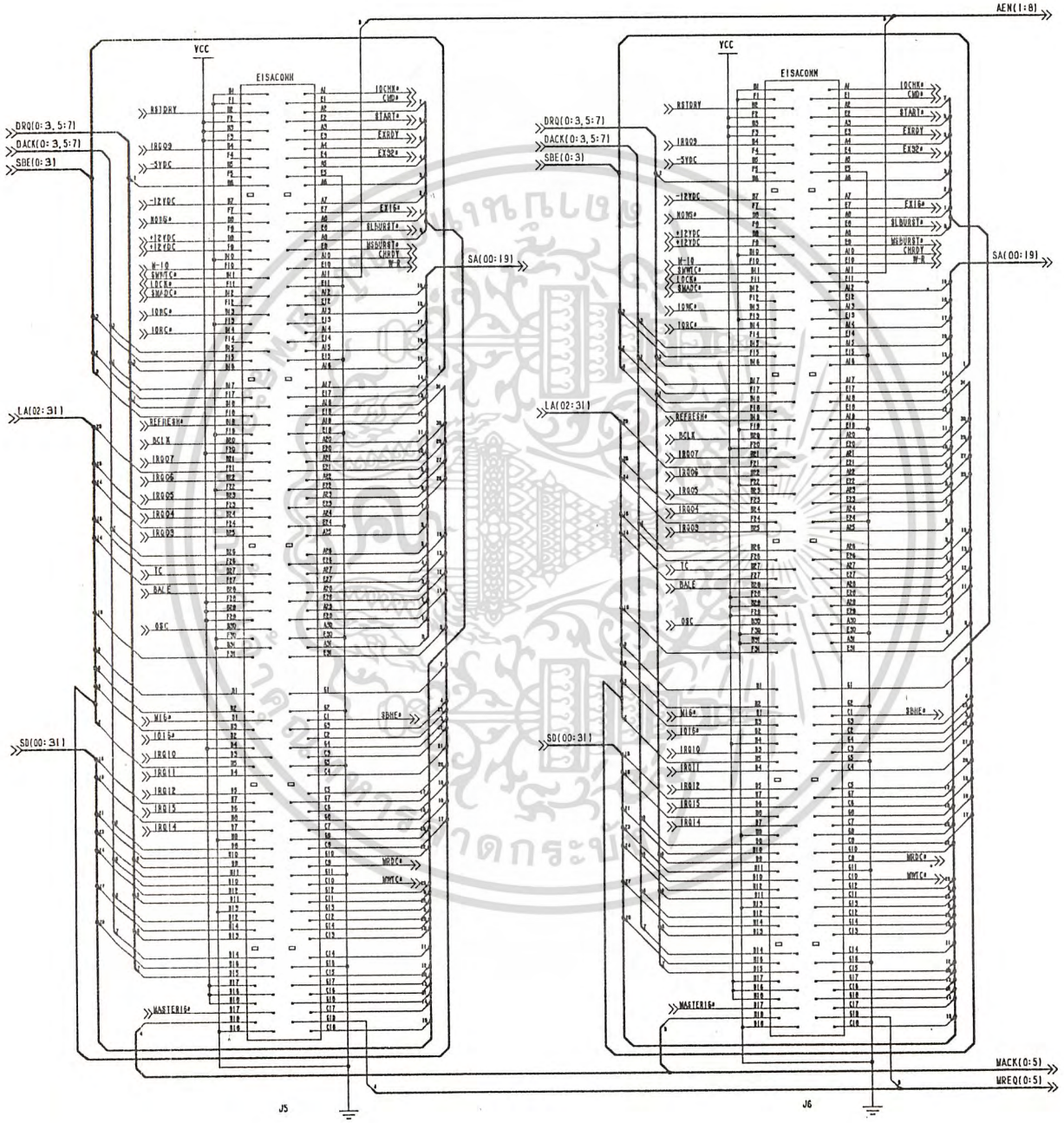

SA(00:19)
XBUS ADDR DECODE
XBUS ADDR DECODE
LA(02:31)
SLOT AEN DECODE
AEN(1:8)
SD(00:31)
XD(00:07)
SBE(0:3)
YCC
486 EISA
SHEET 11(20)

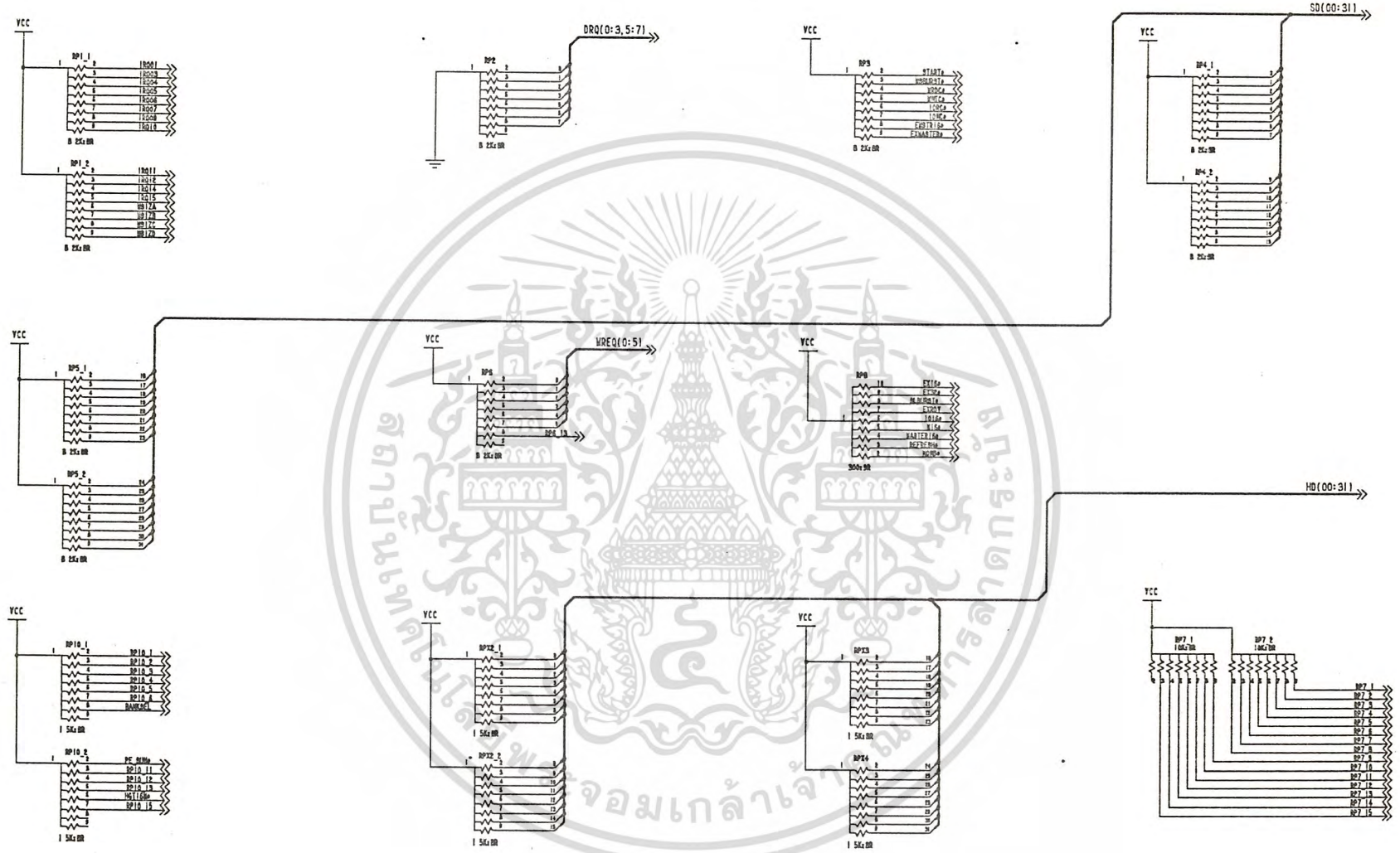

PWROK
PWREN
PWREN#
XD(00:07)
U23
MC146818A
U26
MC14069
BATTPWR
R43
1K
RTCWR#
RTCRD#
RTCALE
IRQ08#
Y2
32.768KHz
C137
27pf
R14
2M
R13
1K
C135
4700pf
C136
18pf
+12VDC
VCC
CR5
IN4148
R12
620ohm
CR8
IN4148
CR6
IN4148
CR7
IN4148
CR12
IN4148
CR1
IN4148
R11
3.3K
J11
U25
PAL20L8B
XBUSTR
ROMCS#
WRDC#
X10WC#
XBUSSEL_L
BALE
ROMADR#
X10RC#
Y1
OUTPUT
14.31819MHz
U20
74F14
R10
10ohm
OSC
FLOPPY &
XBUS ADDR DECODE
486 EISA
SHEET 13(20)



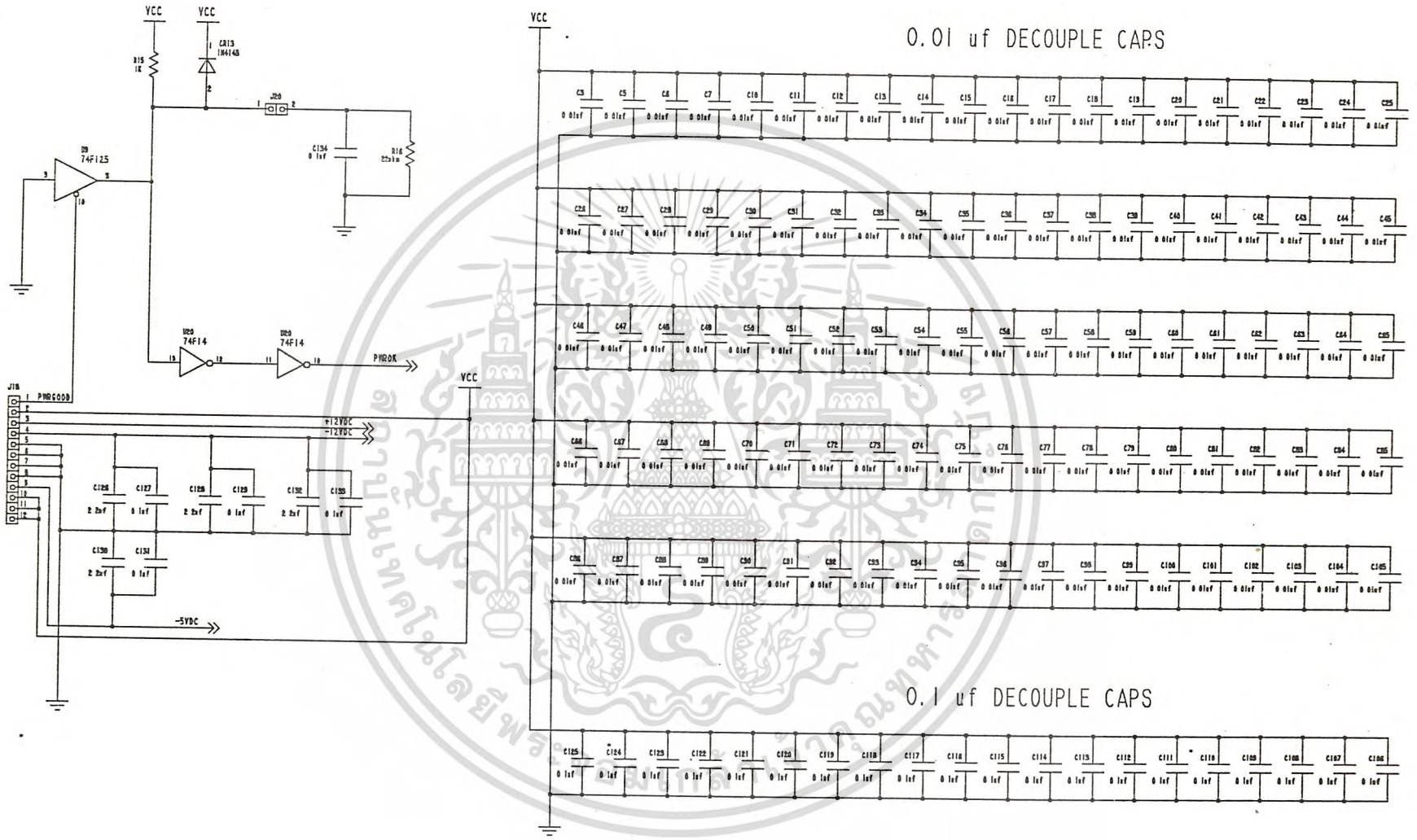
486 EISA
SHEET 14(20)


AEN(1:8)
VCC
EISACONN
DRQ(0:3,5:7)
DACK(0:3,5:7)
SBE(0:3)
SA(00:19)
LA(02:31)
SD(00:31)
J1
J2
WACK(0:5)
WREQ(0:5)


AEN(1:8)
VCC
EISACONN
J5
J6
DRQ(0:3,5:7)
DACK(0:3,5:7)
SBE(0:3)
SA(00:19)
LA(02:31)
SD(00:31)
MACK(0:5)
MREQ(0:5)



486 EISA
SHEET 19(20)



486 EISA
SHEET 20(20)