การออกแบบชุดทดลอง Max+Plus2 ขั้นสูง

Advanced Max+Plus2 Board Design

โดย

นายกล้าศักดิ์ เชื่อนนิล

อาจารย์ที่ปรึกษา

รศ. สมศักดิ์ มิตะถา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

ปริญญานิพนธ์ปีการศึกษา 2543

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การออกแบบชุดทดลอง Max+Plus2 ขั้นสูง

Advanced Max+Plus2 Board Design

ผู้จัดทำ

1. นาย กล้าศักดิ์ เขื่อนนิล รหัสประจำตัว 38013265

_____ อาจารย์ที่ปรึกษา

(รศ. สมศักดิ์ มิตะถา)

การออกแบบชุดทดลอง Max+Plus2 ขั้นสูง

นายกล้าศักดิ์ เขื่อนนิล 38013265
รศ.สมศักดิ์ มิตะถา อาจารย์ที่ปรึกษา
ปีการศึกษา 2543

บทคัดย่อ

การออกแบบวงจรในทางทางด้านดิจิตอลนั้นเดิมทีนั้นไม่สะดวก เพราะหากต้องการเห็นผลลัพธ์การทำงานของวงจรดีจิตอล ต้องต่อลงบนแผงวงจรหรือทำวงจรขึ้นมาทดสอบซึ่งไม่มีประสิทธิภาพการทำงานการออกแบบวงจรในโครงงานนี้มีประสิทธิภาพเร็วขึ้น ทั้งในแง่ของความสามารถจำลองการทำงานวงจรรวมของเกตพื้นฐานทุกอย่าง ไอซีตระกูล74 ทุกตัว อีกทั้งยังสามารถสร้างฟังก์ชันตามที่ออกแบบเองได้ ในรูปแบบของภาษา ในรูปแบบแผนภาพ ในรูปแบบฟังก์ชันสำเร็จรูป ทำให้ได้รับการนำไปใช้อย่างกว้างขวาง เมื่อทำการจำลองการทำงานด้วยโปรแกรมแม็กพลัชทูแล้ว โครงงานนี้มีการออกแบบชุดทดลองเพื่อทดลองต่างๆซึ่งประกอบไปด้วยส่วนของตัวเลขเจ็ดส่วน หลอดแสดงผลแปดตัว สเต็ปมอเตอร์ ดีซีมอเตอร์ สวิตซ์โยก ปุ่มกดโทรศัพท์ จอผลึกเหลว สวิตช์กดติดปล่อยดับ สายดาวน์โหลดข้อมูลระหว่างคอมพิวเตอร์กับชุดทดลอง

Advanced Max+Plus2 Board Design

Klasak keunnin

Assoc. Prof. Somsak mitatha Advisor

ABSTRACT

This paper describes an appoah to the implements of board design on Field Programmable Gate Arrays(FPGA).The software , namely Max Plus II , is used to design and simulate.The structures can contain on a EPM7128SLC84-10 Max Plus II version 9.2. Implementation on FPGA has several advantage compared with other ways such as hight sampling rate , flexibility,and low cost. In modern design practice,the process of digital design has recently taken a major departure form the processes of the recent past as well as from the distant past.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ได้รับความกรุณาจากท่าน อาจารย์ สมศักดิ์ มิตะถา อาจารย์ที่ปรึกษาปริญญานิพนธ์ ท่านได้ให้คำแนะนำชี้แนะวิธีการ และช่วยเหลือด้านอุปกรณ์ ด้านตำราเอกสารเสมอมา

บุคคลที่มีพระคุณที่สุดในชีวิตทำให้ข้าพเจ้าทำโครงงานนี้คือ บิดา มารดา ซึ่งได้ให้โอกาสในการศึกษาในระดับปริญญาจนสำเร็จการศึกษา

ขอขอบคุณคณาจารย์ภาควิชาวิศวกรรมอิเล็กทรอนิกส์มหาวิทยาลัยเทคโนโลยีมหานครสำหรับข้อเสนอแนะ และสำหรับเครื่องมือในการทำโครงงาน

กล้าศักดิ์ เขื่อนนิล

สารบัญ

หน้า

สารบัญภาพประกอบ

หน้า

หน้า

สารบัญตาราง

หน้า

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

เมื่อวิทยาการทางด้านคอมพิวเตอร์และอิเล็กทรอนิกส์เจริญมากขึ้นเรื่อยๆได้มีการแตกขแนงศาสตร์ทางด้านนี้กันมากมายเป็นไปทั้งทางด้านการโปรแกรมและการออกแบบวงจร สองสิ่งที่เป็นของคู่กันในเวลานี้ก็คือวงจรทางด้านดิจิตอล อีกสิ่งหนึ่งคือวงจรทางด้านอนาล็อก ในปัจจุบันพัฒนาการทางด้านดิจิตอลนับว่าได้ก้าวหน้าไปมาก นอกจากพัฒนาการทางด้านการออกแบบโปรแกรมจำลองการทำงานแล้ว การโปรแกรมลงไปยังตัววงจรรวมยังสามารถทำได้ด้วยชุดโปรแกรมเพื่อต่อเชื่อมวงจรรวมกับคอมพิวเตอร์ส่วนบุคคลได้อีกด้วย สมัยอดีตที่ผ่านมาการจะทำการโปรแกรมลงบนวงจรรวมอาจมีจำนวนโลจิกเกตภายในไม่กี่พันตัวแต่สมัยนี้ได้เป็นล้านตัวและยังมีชุดคำสั่งสำหรับประมวลผลพิเศษในการกระทำการประมวลผลสัญญาณดิจิตอลได้ด้วยสิ่งใหม่ๆทางด้านนี้ยังลงพัฒนาต่อไปอย่างไม่หยุดยั้งก็เพราะยังมีความต้องการความเข้าใจในการประยุกต์ใช้งานกันอย่างกว้างขวาง เช่นในองค์การโทรศัพท์แห่งประเทศไทย เดิมทีระบบเครือข่ายโทรศัพท์เป็นการนำเข้า อุปกรณ์ ก็เป็นอุปกรณ์ทางด้านอิเล็กทรอนิกส์เป็นส่วนใหญ่ ต่อมามีการควบคุมผ่านทางคอมพิวเตอร์มีบริการเสริมต่างๆ อย่างมากมาย เช่นระบบแสดงผลเบอร์โทรศัพท์ที่เราคิดต่อด้วย ซึ่ง เดิมก็เป็นระบบ ใช้ไมโครคอนโทรลเล่อร์ แต่ผู้พัฒนาต้องรู้ภาษาเครื่องลึงเช่นภาษาแอสเซมบลี้ ซึ่งต้องใช้ความรู้ทางด้าน HARDWAREพอสมควรถึงจะสามารถเขียนโปรแกรมสั่งงานได้ตามที่มุ่งหมาย ต่อมาเมื่อมีการพัฒนา FPAG จึงมีความง่ายในการพัฒนาเพราะ โครงสร้างภายในเป็น LOGIC GATE เมื่อผู้ใช้ต้องการที่จะทดลองก็สามารถทำการโปรแกรมลงบน CHIP ซึ่ง หากไม่พอใจต้องการแก้ไข ก็เพียงแต่แก้ที่โปรแกรมได้ง่าย นั่นเอง

1.2 วัตถุประสงค์ของงานวิจัย

1.2.1 ออกแบบบอร์ดทดลองและสร้างต้นแบบเพื่อ ใช้ในการศึกษาการทำงานวงจรดิจิตอล โดยกำหนดให้ภายในบอร์ด มีอุปกรณ์อินพุต/เอ้าท์พุตเช่น LED 8 bit,Switch 8 bit,LCD 16 X 1,Keyboard,Stepping Motor,DC Motor,Seven-Segment เป็นต้น

1.2.2 สามารถเขียนโปรแกรมภาษา AHDL ได้ ซึ่งเมื่อเขียนภาษาเสร็จแล้วสามารถจำลองการทำงานได้ด้วยโปรแกรม Maxplus II เพื่อดู Timming Diagram ดูความถี่สูงสุดของวงจร ดูจำนวน Logic Gate ว่าใช้ไปเท่าไหร่ สามารถบอกได้ว่าควรใช้ Chip เบอร์ไหนในการสร้างจริง ๆ

1.2.3 ในด้าน Hardware สามารถทำการต่อวงจร Buffer จากคอมพิวเตอร์ ผ่านทาง Port Printer หรือ Parallel Port เพื่อเป็นการนำข้อมูลที่ออกแบบไว้มา Download ลงบน Chip FPGA ของบริษัท Altera ได้ โดยข้อมูลที่ส่งออกมา อาจจะเป็นในรูปแบบที่เขียนจากภาษา AHDL หรือ เขียนจาก ภาพวาด GDF(Graphic design File) ซึ่งเมื่อส่งข้อมูลออกมาแล้วก็สามารถตรวจสอบการทำงานจริงได้บนบอร์ดทดลองที่ได้ออกแบบไว้แล้ว แต่เป็นไปได้ว่าอาจมีปัญหาบางอย่างซึ่งเกิดจากการปฏิบัติ ดังนั้นตัวผู้ทดลองวิจิต้องสามารถหารสาเหตุให้เจอแล้วก็สามารถที่จะแก้ปัญหาสาเหตุนั้นได้ และจะพยายามหาเทคนิควิธีการมาแก้ไขให้สำเร็จตามที่ได้หาแนวทางมาแก้ปัญหา โครงงานนี้จึงเป็นไปตามความคิดของผู้ใช้อย่างแท้จริงเพราะ FPGA ทำได้หลากหลาย

1.3 ขอบเขตของงานวิจัย

งานวิจัยนี้จะสร้างบอร์ดทดลอง FPGA ของบริษัท ALTERA โดยสามารถเขียนโปรแกรมด้วยภาษา AHDL ในการสั่งงาน HARDWARE ซึ่งมีอุปกรณ์ ต่างๆ เช่น LED,7-Segment,Step motor,Dc-Motor,Lcd,Keyboard,Dip Switch เป็นต้น

1.4 วิธีการดำเนินงาน

งานวิจัยในโครงงานนี้จะเริ่มด้วยการศึกษาทฤษฎีพื้นฐานต่าง ๆ ที่เกี่ยวข้องกับงานวิจัย ซึ่งก็มีเรื่องหลัก ๆ อยู่ 2 เรื่องด้วยกัน คือ ทางด้านทฤฎีและทางด้านปฏิบัติ กล่าวคือในการออกแบบวงจรขับสเตปมอเตอร์จะต้องขับแบบไหนใช้ทฤฎีใดในการสั่งงานในจุดนี้ก็คำนึงถึงขีดความสามารถทั้งในด้านความถี่ที่ควบคุม แนวคิดในการสั่งงาน Chip ในโครงงานนี้จะเป็นการนำ Chip FPGA มาทำหน้าที่แทนการควบคุมโดยไมโครคอนโทรลเล่อร์ เป็นชุดฝึกให้ทดลองเขียนโปรแกรมสั่งงาน ซึ่งจะเป็นอีกทางเลือกหนึ่งในการควบคุม เนื่องจาก Chip FPGA นี้เราสามารถที่จะออกแบบวงจรควบคุมทั้งหมดลงภายในตัวเดียว ซึ่งจะสะดวกกว่าที่เราจะต้องใช้ Chip หรือ Gate จำนวนมากมาต่อรวมกันทำให้เกิดความผิดพลาดได้มากกว่าและทำให้ยุ่งยากในการไล่วงจรและการแก้ไข แต่การใช้ FPGA เรายังสามารถจะจำลองการทำงานกับโปรแกรมก่อนและถ้ามีข้อผิดพลาดยังสามารถแก้ไขได้ทันท่วงทีอีกด้วย

บทที่ 2
ทฤษฎีเบื้องต้นของ CHIP FPGA MAX+PLUS II

จากที่ผ่านมาในการศึกษาวิชาดิจิตอลพื้นฐานเราได้เรียนรู้ถึงการทำงานของเกตต่างๆไปจนถึงการนำเอาเกตเหล่านั้นมาต่อเป็นวงจรดิจิตอลเพื่อไปใช้งาน จะเห็นว่าการที่จะได้วงจรควบคุมดิจิตอลที่ทำงานสลับซับซ้อนจะต้องใช้ IC ดิจิตอลเหล่านั้นมาต่อรวมกันเป็นวงจรใหญ่ ยิ่งวงจรใหญ่มากเท่าไหร่สิ่งที่ไม่พึงปราถนาก็จะตามมา คือราคาของวงจรแพงขึ้น การทำงานที่ช้าลงไปจนถึงความสิ้นเปลืองพลังงานเพิ่มขึ้น

2.1 เทคโนโลยี PLD และ FPGA

Application Specific IC (ASIC)

Field Programmable Logic

- PLD หรือ CPLD (Complex Programmable Logic Device)
- FPGA (Field Programmable Gate Array)
- ชื่อย่อเหล่านี้ ค่อนข้างแล้วแต่ผู้ผลิตที่จะใช้เรียกสินค้าเขา ในที่นี้จะขอใช้คำว่า FPGA แทนตระกูลของชิพที่โปรแกรมได้นี้ทั้งหมด

การออกแบบวงจรสำหรับ ASIC และ FPGA มีส่วนเหมือนกัน

ASIC

- การออบแบบในระดับทรานซิสเตอร์ หรือ Layout มักใช้กับการออกแบบไลบรารี่เซลล์ (Library cells), วงจรเล็ก , และวงจรที่มีลักษณะพิเศษ เช่น RAM
- วงจรโดยทั่ว ๆ ไปมักออกแบบในระดับสูง โดยใช้แผนภาพวงจร ผสมกับ ภาษา(Schematics and Languages) แล้วใช้ซอฟท์แวร์สังเคราะห์ลอจิก (Logic Synthesis Tools) แปลงการออกแบบในระดับสูงนี้ ให้เป็นวงจรระดับล่างที่เป็นการต่อกันของไลบรารี่เซลล์ต่าง ๆ

FPGA

- **ออกแบบในระดับสูง โดยใช้แผนภาพวงจร ผสมกับ ภาษา แล้วใช้ซอฟท์แวร์สังเคราะห์ลอจิก (Logic Synthesis Tools) แปลงเป็นข้อมูลที่สามารถนำไปโปรแกรมลงในชิพได้**

Hardware Description Language (HDL)

เป็นภาษาที่ใช้สำหรับออกแบบฮาร์ดแวร์ หรือวงจรดิจิตอล

ภาษามาตรฐานสากล

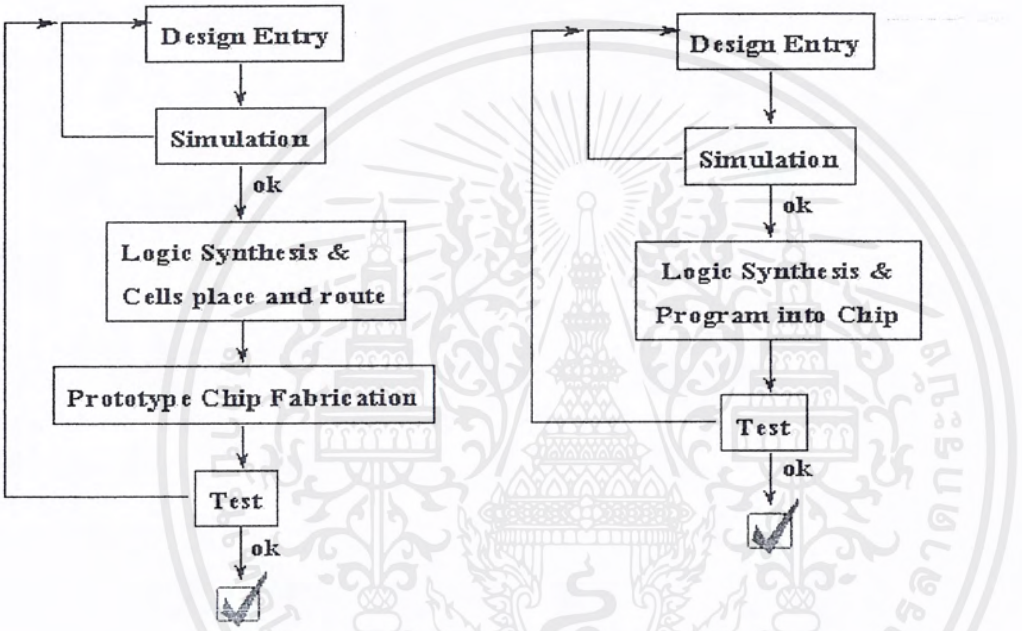
VHDL (VHSIC Hardware Description Language)

Verilog

ภาษาที่ไม่เป็นมาตรฐาน

- AHDL (Altera Hardware Description Language)
- PHDL (Philips Hardware Description Language)
- etc.

2.2 กระบวนการออกแบบ ASIC และ FPGA (ASIC Design Process v.s. FPGA Design Process)



รูปที่ 2.1 แสดงกระบวนการออกแบบ ASIC และ FPGA

2.3 ข้อได้เปรียบของการใช้ FPGA

1. ใช้เวลาการออกแบบเร็วมากเพื่อให้ได้ชิพต้นแบบ
2. ต้นทุนในการออกแบบต่ำกว่า ASIC มาก
3. ความเสี่ยงในการลงทุนน้อย เพราะ ไม่จำเป็นต้องผลิตเป็นจำนวนมาก สามารถผลิตเพิ่มได้เท่าที่มีตลาดต้องการ
4. การเปลี่ยนแปลง หรือแก้ไขวงจรในภายหลัง ทำได้ง่าย และต้นทุนต่ำ
5. **ข้อเสียเปรียบของการใช้ FPGA**

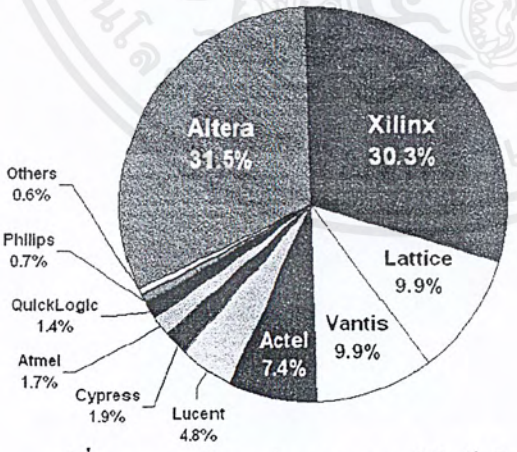
6. ASIC เหมาะกว่าในงานที่มีความต้องการด้านเทคนิคสูงมาก ๆ เช่น ความจุสูงมาก ๆ ความเร็วสูงมาก ๆ หรือกินไฟต่ำมาก ๆ
7. ASIC เหมาะกว่าสำหรับการผลิตในปริมาณมาก ๆ (10000+) เพราะ ที่ปริมาณผลิตมาก ๆ ต้นทุนของ ASIC จะต่ำกว่า FPGA ได้

2.4 ทางเลือก

1. ใช้ FPGA ในงานที่ต้องการผลิตไม่มากนัก
2. ใช้ FPGA ในช่วงเริ่มต้นของสินค้า ซึ่งอาจจะยังไม่แน่ใจในปริมาณตลาด อีกทั้งวงจรอาจยังมีจุดบกพร่องที่ต้องแก้ไข และปรับปรุง
3. การแปลงจาก FPGA เป็น ASIC ทำได้ไม่ยากนัก ถ้าออกแบบด้วยภาษา เช่น VHDL หรือ Verilog
4. FPGA สามารถนำมาใช้ทำชิพต้นแบบของ ASIC ในช่วงที่ออกแบบได้
5. อย่างไรก็ตาม ปัจจุบัน FPGA ใช้เทคโนโลยีสูงขึ้น และมีราคาถูกลงมาก สินค้าหลายอย่างถึงแม้จะผลิตจำนวนมาก ก็ยังสามารถใช้ FPGA ได้ในต้นทุนที่คุ้มค่ากว่า ASIC

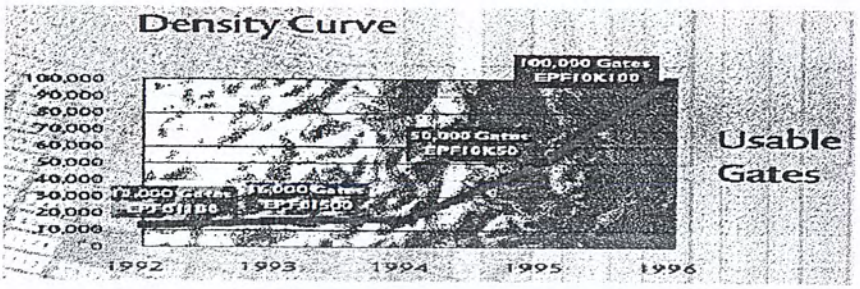
2.5 ผู้ผลิต FPGA

ส่วนแบ่งตลาดปี 1998 (ที่มา: EE Times 4-Oct-99)



รูปที่ 2.2 แสดงส่วนแบ่งตลาดของบริษัทผู้ผลิต CHIP FPGA

2.6 ความจุในชิพ FPGAs เติบโตอย่างรวดเร็ว



รูปที่ 2.3 แสดงแนวโน้มของความจุของเกต

ตัวอย่างของปริมาณเกตที่ใช้

-8-bit counter ใช้ 160 gates

-8-bit full adder ใช้ 200 gates

-8-bit multiplier ใช้ 4000 gates

-8-bit 16-tap FIR filter ใช้ 50,000 gates (parallel structure)

เทคโนโลยีของ FPGA หรือ CPLD

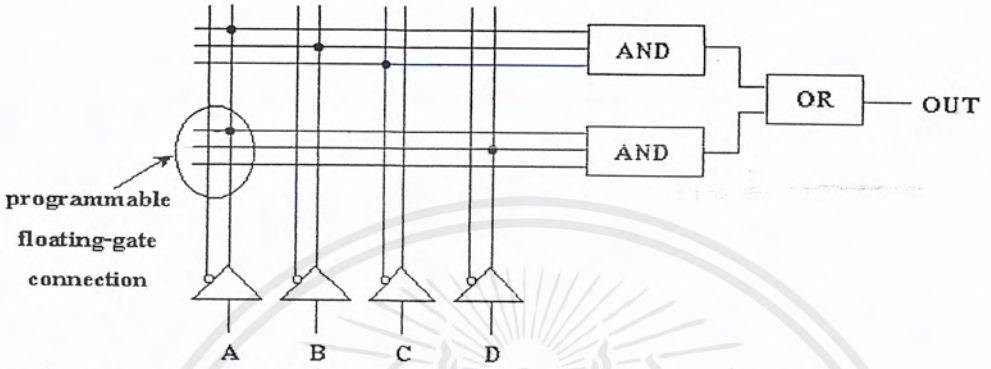
1. Antifuse-Based FPGA (ล้าสมัย)
2. EEPROM-Based FPGA
3. SRAM-Based FPGA
4. Mask-Based FPGA (ใหม่!)
5. MPLD (Mask-Programmed Logic Devices)

EEPROM-Based FPGA

1. มักเรียก FPGA ประเภทนี้ว่า PLD
2. มีความจุเกตต่ำ โดยทั่วไปน้อยกว่า 20,000 เกต
3. ใช้เทคโนโลยีเหมือน EEPROM (floating-gate) ในการโปรแกรม
4. สามารถเก็บข้อมูลที่โปรแกรมลงไปได้โดยไม่จำเป็นต้องมีไฟเลี้ยง (Non-volatile configuration)
5. สามารถโปรแกรมซ้ำได้ประมาณ ประมาณ 10000 ครั้ง
6. ใช้ AND-OR plane ในการทำลอจิกฟังก์ชั่น
7. มักมีการจัดสถาปัตยกรรมในรูปแบบอะเรย์ (ARRAY structure)

2.7 โครงสร้างของ AND-OR Plane

- ฟังก์ชั่นทางลอจิกใด ๆ สามารถทำได้ด้วย Sum of Products
- i.e. OUT = AB$\underline{C}$ + AD



รูปที่ 2.4 แสดงโครงสร้างแบบ AND-OR PLAN

2.8 ตัวอย่างสถาปัตยกรรมของ MAX7000 (Example: MAX7000 Architecture)

FPGA Chip จัดเป็นอุปกรณ์ประเภท PLD(Programmable Logic Devices) โดยแบ่งออกได้เป็น

1). EEPROM (Electrical Erasable Programmable Memory)

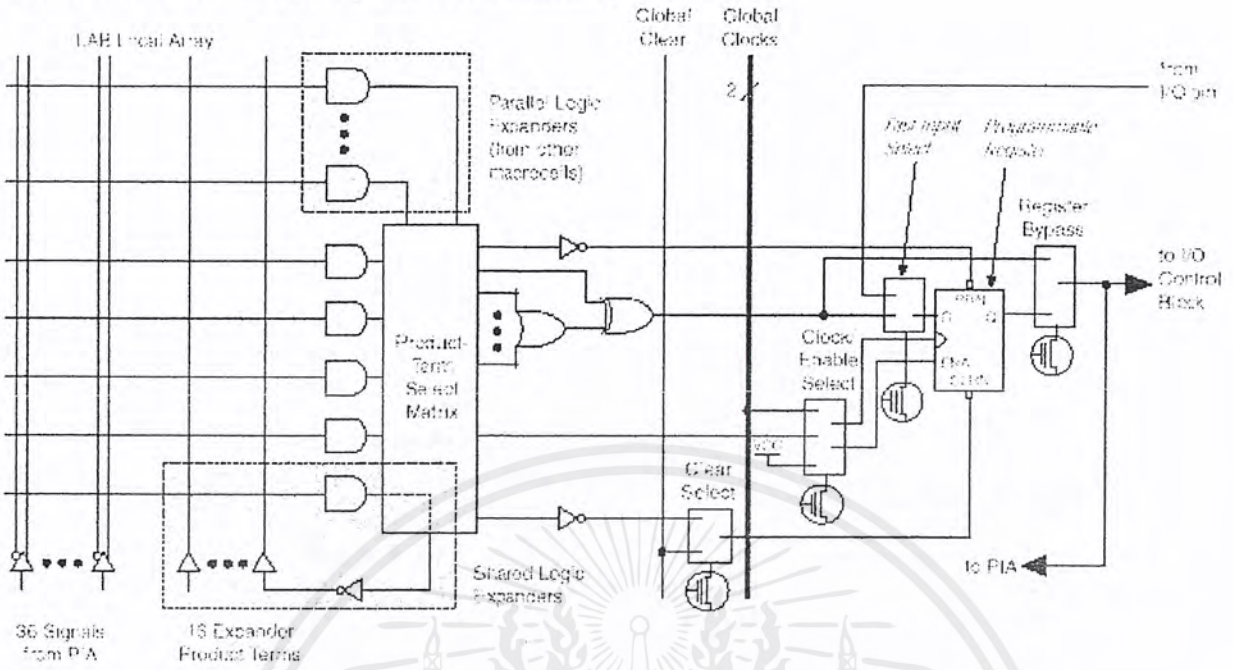
EEPROM จะเป็นโครงสร้างแบบ AND-OR Plan เป็นหน่วยความจำที่สามารถค้างค่าของโปรแกรมได้ โดยไม่ต้องมีแหล่งจ่ายแรงดันไฟฟ้า และสามารถลบค่าภายในโดยใช้แรงดันไฟฟ้า

2).SRAM (Static Random Access Memory)

SRAM เป็นหน่วยความจำที่ค้างค่าของโปรแกรมได้ชั่วขณะ จะทำหน้าที่แบบ Look up Tables การอ่านค่าจากหน่วยความจำภายนอกทุกครั้งที่เริ่มจ่ายแรงดันไฟฟ้าให้กับมันและข้อมูลภายในจะถูกลบไปทันทีที่ไม่มีการจ่ายแรงดันให้กับ SRAM

Altera Max7000 เป็น FPLD (Field Programmable Logic Device) เป็นโครงสร้างแบบ CMOS EEPROM ทั้งหมด เกตที่ใช้จะอยู่ในช่วง 150-5,000 ตัว ใช้เทคโนโลยีการผลิตขนาด 0.5-0.35 um ซึ่งมีพลังงานสูญเสียต่ำ จึงเป็นไปได้ที่นำเอาไปต่อร่วมกับอุปกรณ์รอบข้างโดยอาศัยแหล่งจ่าย พลังงานได้ในชุดเดียวกัน นั่นคือทำให้มีขนาดที่เล็กกระทัด

2.8 สถาปัตยกรรมของ MAX7000 (MAX7000 Architecture)

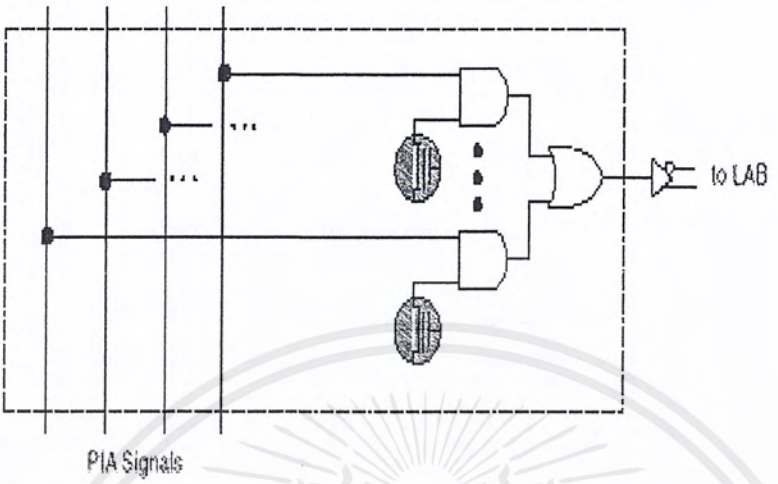


รูปที่ 2.5 แสดงโครงสร้างของสถาปัตยกรรม MAX7000s

โครงสร้างนี้อธิบายตามอุปกรณ์ที่สามารถโปรแกรมได้ โดยใช้สำหรับโปรแกรมวงจรที่ได้ออกแบบลงไป เพื่อให้อุปกรณ์ Fpga มี ฟังก์ชันการทำงานตามแบบที่ต้องการ ซึ่งเป็นวิธีการออกแบบ IC (Integrated Circuit) แบบ Semicustom อีกวิธีหนึ่ง กล่องสี่เหลี่ยมภาค ที่เชื่อมต่ออยู่กับ 16 Expander Product Terms นั้น แปลความหมายได้ก็คือ การนำองค์ประกอบออกเป็นส่วนย่อยๆ มา คูณแล้วรวม อนุกรมกันไป กล่าวคือ เป็น ผลรวมของผลบวกในทางภาษา Digital คือ โครงสร้างของ Product Term Seriect Select Matrix นั้นคือ Fpga ในระดับ เกต(Gate-Level) เหมาะกับโครงสร้างที่มีอยู่ในอุปกรณ์ Fpga นี้เพื่อจะแสดงการหาผลลัพธ์ได้อย่างมีประสิทธภาพ และในการสังเคราะห์สามารถกำหนดข้อบังคับสำหรับโมเดลได้คือ ในเรื่องของเวลาเมื่อมีความจำเป็นต้องใช้ทั้งทรัพยากรโดยรวมของระบบ หรือกำหนดตำแหน่งและหน้าที่ของ I/O ซึ่งข้อบังคับเหล่านี้จะถูกนำไปใช้ในขั้นตอนการ ออปติไมต์เพื่อให้วงจรที่ได้เป็นไปตามข้อกำหนด ส่วนสำคัญของการ ออปติไมต์คือการเทียบ (mapping) Model ให้เข้ากับ เทคโนโลยีที่ใช้เพื่อให้ได้วงจรที่เหมาะสมกับโครงสร้าง โดย I/O ไม่เหมือนกรณี I/O ของวงจร TTL เพราะจะถูกกำหนดได้จาก User ว่าถ้าอยากจะให้เป็น Tristate ตรงนี้เสือนมี Function ซึ่งกั้นระหว่างวงจรทั้งระหว่างขาเข้าและขาออกของกระแสไหลผ่าน เทคโนโลยีที่ใช้ได้เปลี่ยนเป็นหน่วยความจำตารางกล่าวคือ เป็นแผนผังรวมพื้นที่หน่วยความจำ ในตำแหน่ง Cell Model

สำหรับ องค์ประกอบอื่นๆ ภายในตัวอุปกรณ์ FPGA แบ่งออกเป็นส่วนย่อยๆ สำหรับเกณฑ์ในการแบ่งคือให้แต่ละส่วนที่จะแยกออกจากกันมีสัญญาณที่เชื่อมต่อระหว่างกันน้อยที่สุดเท่าที่จะทำได้ เพื่อช่วยลดความหนาแน่นในตอนการทำการเชื่อมต่อสัญญาณ (Routing) ในขั้นตอนนี้จะถูก โปรแกรมข้อมูลโดย MAXPLUS II (S) จะเทียนส่วนประกอบของวงจรเช่น เกท (Gate) ,Filp-Flop ลงในทรัพยากรต่างๆ ที่มีอยู่ภายใน FPGA การแลือกที่ตั้งของแต่ละส่วนของวงจรที่ผ่านการแบ่งวงจร (Partition) แล้วควรจะอยู่ ณ ตำแหน่งไหนในอุปกรณ์ FPGA เพื่อให้ได้ผลลัพธ์ดีที่สุด คือเมื่อวงจรไหนอยู่ใกล้กัน โดยเฉพาะการค้นหาเส้นทาง (Routing) ตามการกำหนดตำแหน่งขา I/O (I/O Pin) ตามตำแหน่งขา I/O จริงๆ แล้วเกณฑ์การตัดสินคือความหน่วง การทำการค้นหาเส้นทาง

2.8.2 Programmable Interconnect



■ Altera เรียก บัสนี้ว่า PIA (Programmable Interconnect Array)

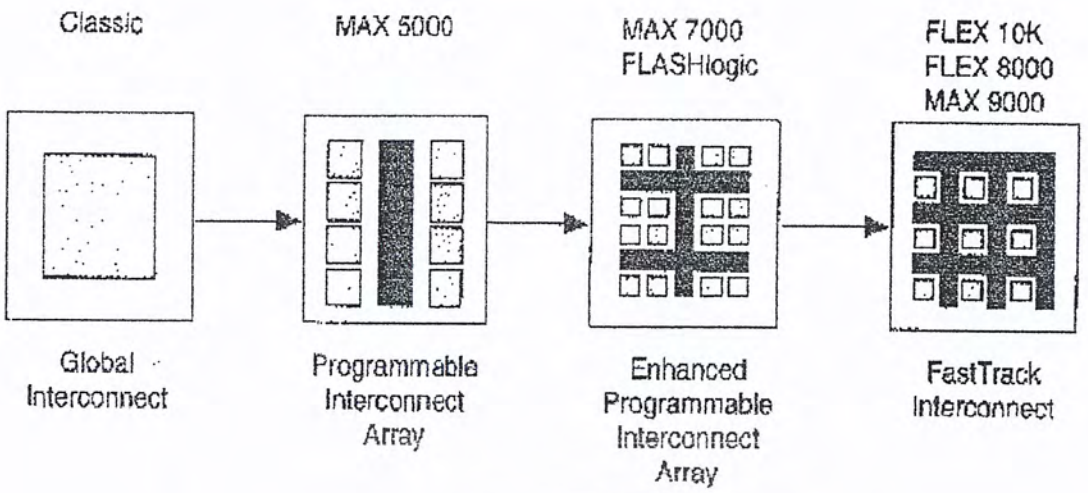
รูปที่ 2.7 แสดงบัสนี้ว่า PIA (Programmable Interconnect Array)

2.8.3 แนวทางในการศึกษาสถาปัตยกรรม

1. เราไม่จำเป็นต้องรู้สถาปัตยกรรมของชิพ ในการโปรแกรม
2. ซอฟท์แวร์สังเคราะห์ลอจิก จะเป็นผู้แปลงจากไฟล์ที่เราออกแบบ (ด้วยภาษา หรือ schematic) ไปเป็นข้อมูลที่จะโปรแกรมลงในชิพ
3. เราสามารถควบคุมเงื่อนไขในการสังเคราะห์ลอจิกได้บ้าง เช่น ต้องการสังเคราะห์ให้มีความเร็วสูงสุด หรือให้กินปริมาณทรัพยากรต่ำที่สุด
4. หลังจากสังเคราะห์วงจรแล้ว สามารถจำลองการทำงานในซอฟท์แวร์ได้ก่อน และวิเคราะห์ timing ได้ด้วย

2.8.4 SRAM-Based PLDs

1. มีความจุเกตปานกลาง หรือสูงมาก (10,000 - 250,000 เกต)
2. ใช้เทคโนโลยีเหมือน SRAM (floating-gate) ในการโปรแกรม
3. ไม่สามารถเก็บโปรแกรมในภาวะที่ไม่มีไฟเลี้ยง (volatile configuration)
4. มักต้องใช้ควบคู่กับ ROM เพื่อเก็บโปรแกรม และโหลดโปรแกรมเข้าในตัวชิพเมื่อเริ่มต้นใช้งาน
5. สามารถโปรแกรมซ้ำได้ไม่จำกัดจำนวนครั้ง
6. ใช้ Look-up Table ในการทำลอจิกฟังก์ชั่น
7. จัดทรัพยากรในโครงสร้างแบบอะเรย์เช่นเดียวกัน



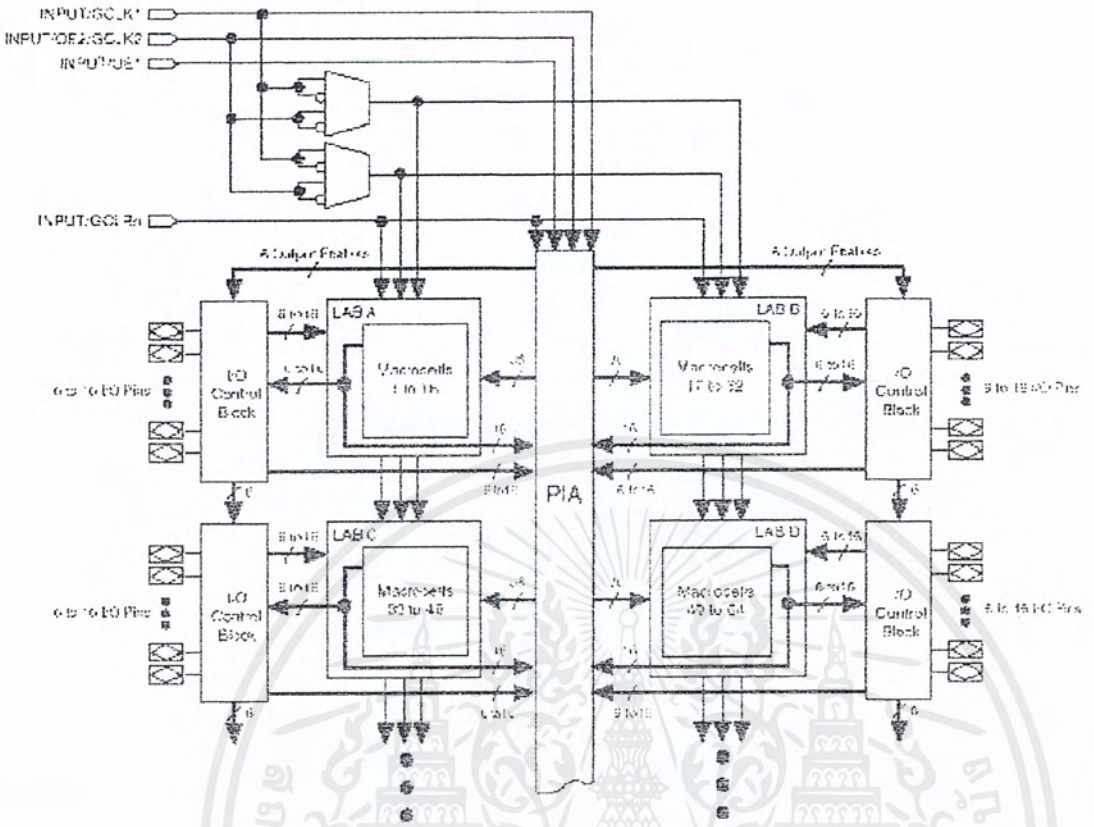
รูปที่ 2.8 แสดงถึงความหนาแน่นใน Chip ที่เพิ่มขึ้นเรื่อยๆ

ในรุ่น Classic เป็นรุ่นที่มีความไม่ซับซ้อนมี เกตภายใน 2000 ถึง 5000 เกต เป็นการใช้เทคโนโลยี หน่วยความจำแบบ Flash สังเกตเห็นว่าพอมาถึงในรุ่นหลังๆ การทำงานมีประสิทธิภาพมากมายทั้งในด้านความเร็ว ด้านการหาเส้นทางในการเดิน เป็นต้น

Table 4. MAX 7000 Device Features

Feature	EPM7032 EPM7064 EPM7096	All MAX 7000E Devices	All MAX 7000S Devices
ISP via JTAG interface			✓
JTAG BST circuitry			✓ (1)
Open-drain output option			✓
Fast input registers		✓	✓
Six global output enables		✓	✓
Two global clocks		✓	✓
Slew-rate control		✓	✓
MultiVolt interface (2)	✓	✓	✓
Programmable register	✓	✓	✓
Parallel expanders	✓	✓	✓
Shared expanders	✓	✓	✓
Power-saving mode	✓	✓	✓
Security bit	✓	✓	✓
PCI-compliant devices available	✓	✓	✓

ตารางที่ 2.1 แสดงคุณลักษณะขององค์ประกอบ MAX 7000 Device



รูปที่ 2.9 แสดงถึง Max7000E และ Max7000s Device Block Diagram

Logic Array Blocks ภายในตัว Chip FPGA MAX7000 มีการเชื่อมต่อสัญญาณกันอย่างมีความน่าเชื่อถือและมีประสิทธิภาพ มีการพัฒนาให้ กลุ่มของ ARRAY ในแต่ละชุดอย่างที่เรียกว่า Logic Array Blocks (LABs) ซึ่ง Logic Array Blocks (LABs) นั้นจะประกอบไปด้วย 16-Macro Cell Array

ในรูปได้แสดงถึงการ Mlutiple กันของ LABs ที่เชื่อมต่อกันโดย Interconnect array (PIA) มี Globlal Bus นั้นจะเตรียมการนำเข้า ขา I/O ทั้งหลายและ Macrocell

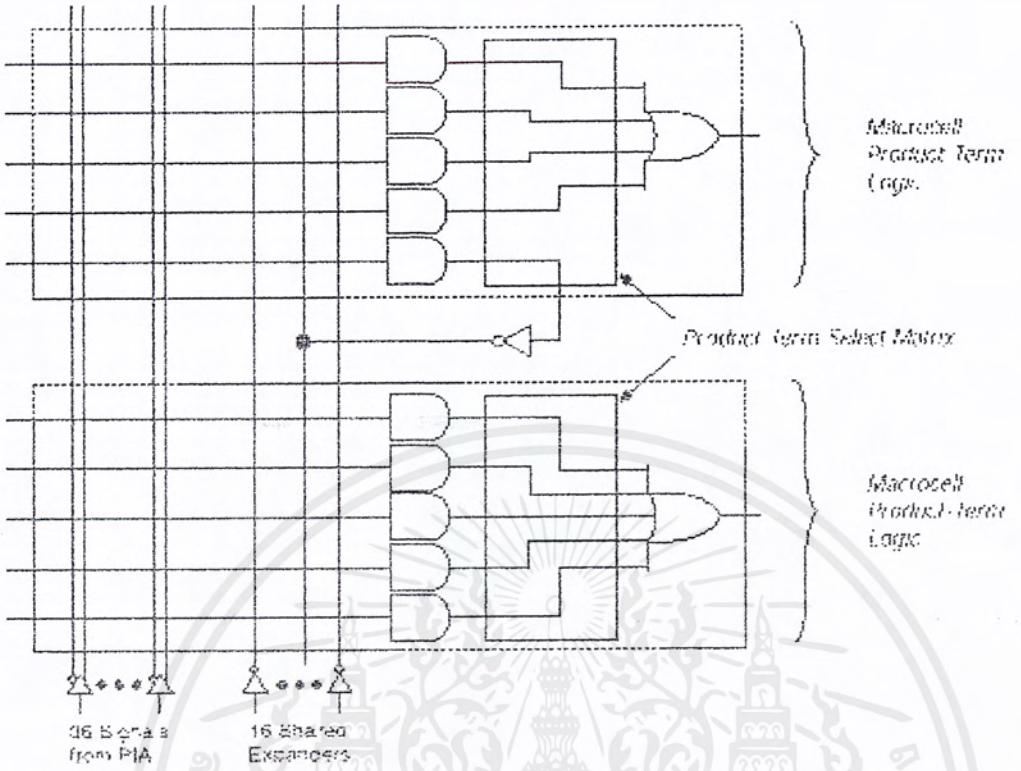
Table 5. MAX 7000 Maximum User I/O Pins Note (1)												
Device	44-Pin PLCC	44-Pin PQFP	44-Pin TQFP	68-Pin PLCC	84-Pin PLCC	100-Pin PQFP	100-Pin TQFP	160-Pin PQFP	160-Pin PGA	192-Pin PGA	208-Pin PQFP	208-Pin RQFP
EPM7032	36	36	36									
EPM7032S	36		36									
EPM7064	36		36	52	68	68						
EPM7064S	36		36		68		68					
EPM7096				52	64	76						
EPM7128E					68	84		100				
EPM7128S					68	84	84 (2)	100				
EPM7160E					64	84		104				
EPM7160S					64		84 (2)	104				
EPM7192E								124	124			
EPM7192S								124				
EPM7256E								132 (2)		164		164
EPM7256S											164 (2)	164

ตารางที่ 2.1 แสดงถึงความแตกต่างของจำนวน INPUT/OUTPUT ของ MAX

จากตาราง EPM ในแต่ละรุ่นถูกระบุไว้ว่าเป็นของ MAX ทั้งหมดโดยยังไม่ได้คิดเหมือนกับเทคโนโลยี โดยใช้ Flex เพราะถึงแม้ว่า Flex จะมีความจุภายใน Chip สูงก็ตามแต่ก็ไม่สามารถเก็บข้อมูลไว้ภายในตัวได้ เมื่อมองแผนการแล้วในการจะเลือกใช้ในประเด็นนี้จึงควรลองกับ MAXPLUS SIMULATE เพื่อหาว่าจะใช้ LOGIC CELL มากไหม ที่สำคัญก็คือว่า จะใช้ขา INPUT/OUTPUT กี่ขา

อย่างไรก็ตามเมื่อมีการปลดตำแหน่งขาออกเมื่อไม่มีการใช้งานคือเป็นการเชื่อมต่อระหว่างขา Input กันภายในจะทำให้เราประหยัดทรัพยากร ต่างๆ ได้มากกว่าการหาเส้นทางเดินเองซึ่ง จำนวนขาที่ไม่แตกต่างหรือแตกต่างกันเหล่านั้น เราจะยังคงเลือก MAX เพราะได้มีพลังไม่มีปัญหาในการ ใช้งาน อย่างไม่มีเวลาที่จะรอ

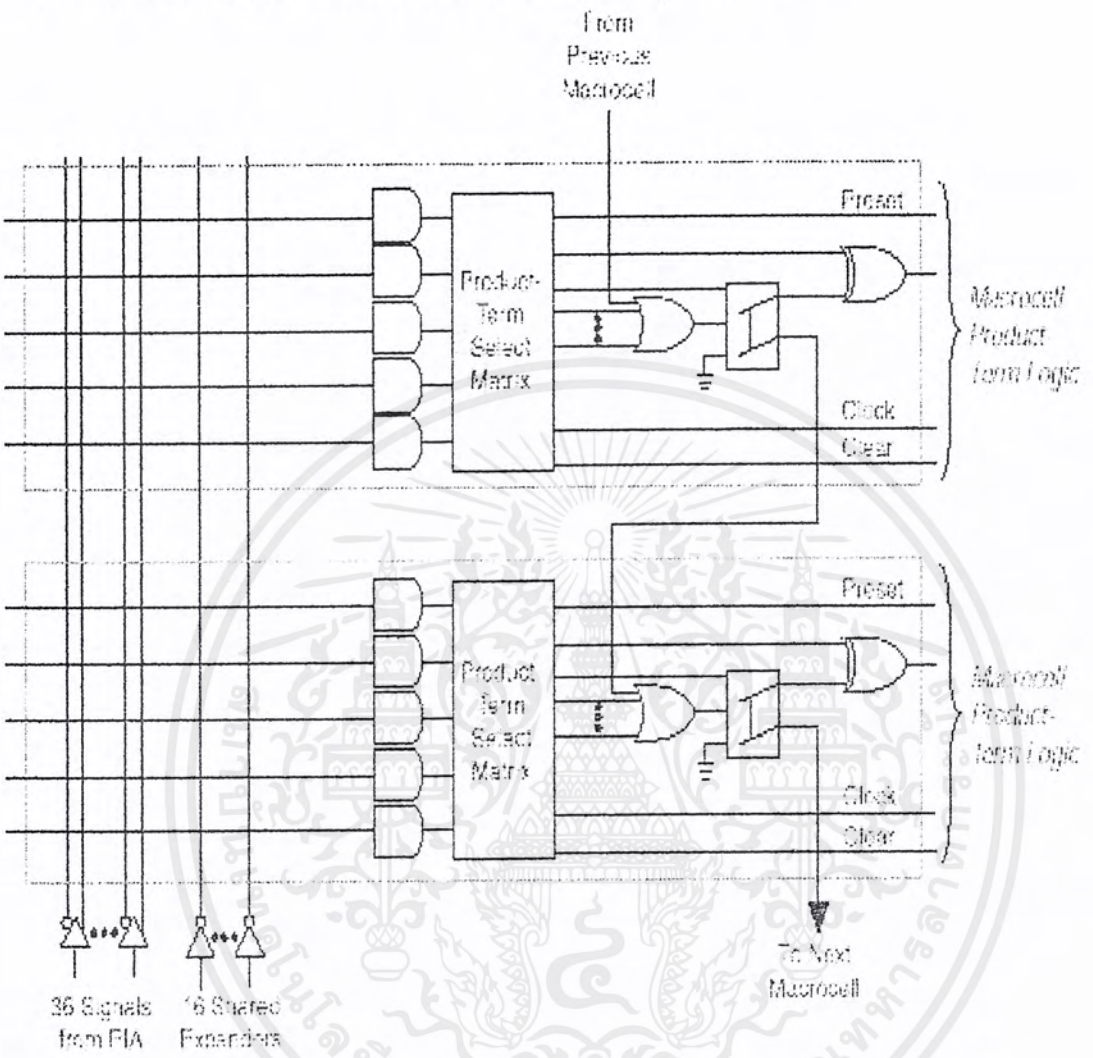
ในกลุ่มของขาเหล่านั้น (All Pin I/O) ได้ดังนี้คือ 44 Pin,68 Pin,84 Pin,100 Pin --240 Pin ในขณะนี้



รูปที่ 2.10แสดงถึงการทำงานแบบ Shareable Expanders

วิธีการ Shareable Expanders คือในแต่ละ Logic Array Blocks จะมี 16 Shareable Expanders ในแต่ละส่วนเหล่านี้สามารถจะมองเห็นคล้ายๆ จะเป็นหนึ่ง Product Terms (จากแต่ละอันของ Macrocell)

โดยวิธีการ inverted outputs และย้อนกลับ Feedback มายัง Logic Array แต่ละ Shareable Expanders สามารถใช้และ Share โดยหลายๆ หรือทั้งหมดของ Macrocell ในหนึ่ง Logic Array Blocks ถึงการทำงานแบบซับซ้อนฝังตัว มีการล่าช้าในการ Delay เล็กๆ (Small Delay)

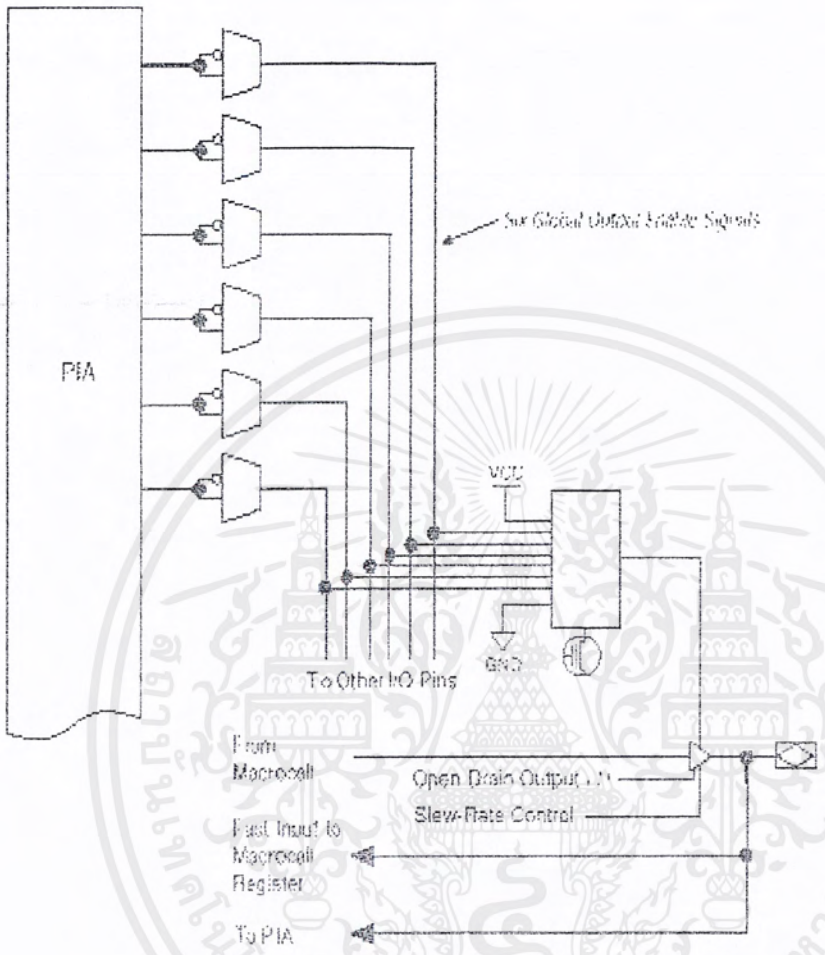


รูปที่ 2.11 แสดงถึงการทำงานแบบ Parallel Expanders

Parallel Expanders ไม่ได้ใช้ Product Term ทั้งนี้สามารถจอง Macrocell เพื่อนบ้าน (neigh boring Macrocell) ที่จะทำให้มีความเร็วและ Complex logic functions

Parallel Expanders อนุญาติให้มีได้ถึง 20 Product Termในทางตรงมีการ OR กันระหว่าง Microcell กับ Logic ด้วย S Product Term ประกอบด้วย Microcell และ 15 Parallel Expanders โดยที่ neigh boring Macrocell ใน Logic Array Blocks มี 2 กลุ่ม ของ 8 Macrocell ในแต่ละ Logic Array Block

ตัวอย่างเช่น Macrocell 1 ถึง 8 และ 9 ถึง 16 Macrocell ในแต่ละ Logic Array Block จาก 2โซ่ที่จะมีการยืม (Borrow parallel Expanders) Macrocell ยืม Parallel Expanders จาก Lower Numbered Macrocell ก็อาจเป็นกรณีเช่น Macro 8 สามารถ ยืม Parallel Expanders จาก macrocell 7,จาก Macrocell 7 และ 6 หรือจาก Macro 7,6 และ 5



รูปที่ 2.12 แสดงถึง I/O Control Block

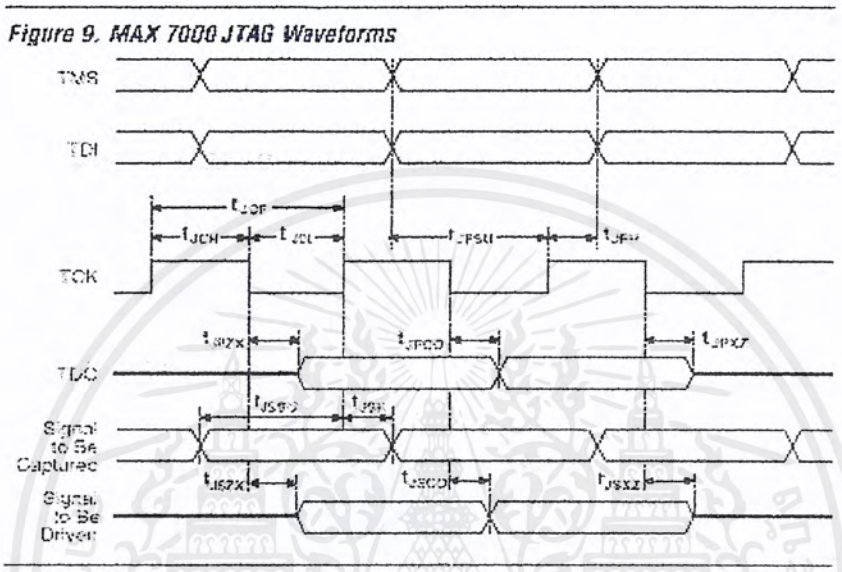
จากรูปที่ 2.10 ได้แสดงให้เห็นถึง I/O Control Block ซึ่ง I/O Control Block อนุญาติให้แต่ละขา ของ I/O มีการทำงานแบบ 2 ทิศทางหรือ Bidirectional Operation

ในการทำงานของ I/O Control Block นั้นเป็นแบบ Tri-State ก็แล้วแต่ว่า คุณ อยากจะต่อเข้า กับ VCC หรือ Ground ก็ได้

จากรูปเมื่อ Tri-State Buffer ขณะที่กำลังต่ออยู่กับ Ground ค่า Output ที่ได้หรือ Tri-State เป็น High impledance และขา I/O สามารถใช้เป็น enable ได้อีกด้วย

2.9 การเชื่อมต่อ JTAC กับ MAX 7000S

การเชื่อมต่อทาง Port Parallel จาก Computer มาสู่ ตัว Chip Max7000s นั้น จะมี ค่า Parameter และเวลาในการ Access Time เหมือนระบบอิเล็กทรอนิกส์ ทั่วไป คือ ต้องให้เวลาแก่ ตัวอุปกรณ์สักช่วงเวลาหนึ่ง มันถึงจะทำงานตามที่เราต้องการได้ ซึ่งทาง บริษัท Altera ก็ได้มีมาตรฐานของตัวเองในการที่จะสามารถทำให้เป็นการ initial system เพื่อให้เกิดสภาวะพร้อมรับคำสั่งได้



รูป 2.13 แสดง Timing diagram ของการเชื่อมต่อในโหมด JTAC MODE

Table 9. JTAG Timing Parameters & Values for MAX 7000S Devices

Symbol	Parameter	Min	Max	Unit
t_{JCP}	TCK clock period	100		ns
t_{JCH}	TCK clock high time	50		ns
t_{JCL}	TCK clock low time	50		ns
t_{JPSU}	JTAG port setup time	20		ns
t_{JPH}	JTAG port hold time	45		ns
t_{JPCO}	JTAG port clock to output		25	ns
t_{JPZX}	JTAG port high impedance to valid output		25	ns
t_{JPXZ}	JTAG port valid output to high impedance		25	ns
t_{JSSU}	Capture register setup time	20		ns
t_{JSH}	Capture register hold time	45		ns
t_{JSCO}	Update register clock to output		25	ns
t_{JSZX}	Update register high impedance to valid output		25	ns
t_{JSXZ}	Update register valid output to high impedance		25	ns

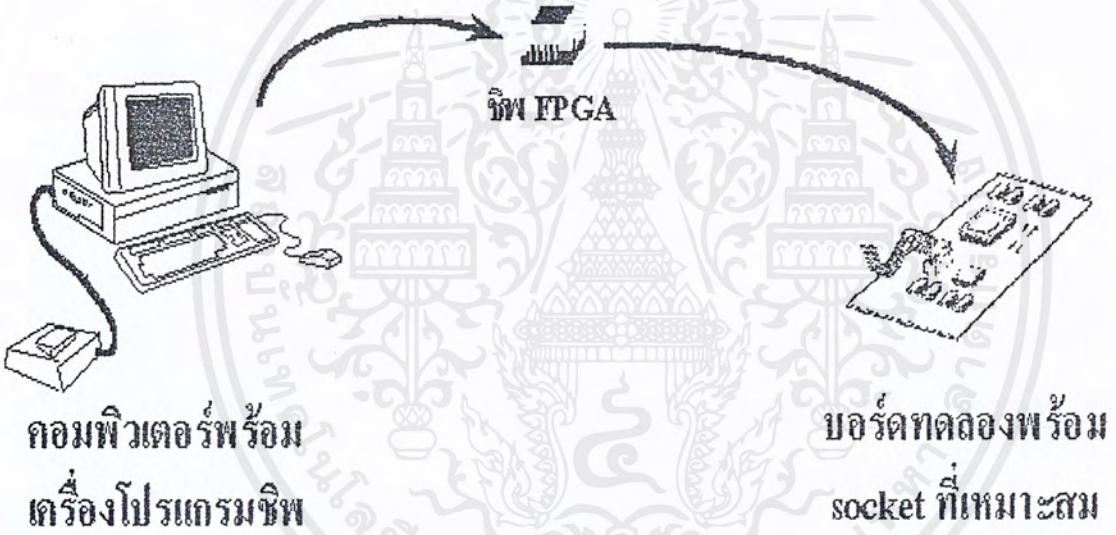
ตารางที่ 2.2 แสดงค่า Parameter ต่างๆ ในการใช้งาน Chip FPGA MAX7000 โดยมีทั้งค่าสูงสุดและค่าต่ำสุด

2.10 การโปรแกรมลงบน Chip Maxplus II

ความสามารถในการโปรแกรมนับเป็นจำนวนครั้งได้ถึง 10,000 ครั้งต่อ Chip MAX7000s โดยในรุ่น Student Version ของ Program ก็ได้แก่การนำ ชื่อ File ที่มีนามสกุลเป็น POF หรือคือ โปรแกรม ภาษาเครื่องผ่านทาง Computer จัดลำดับ จัดการ โดย Operating System Computer จากนั้น เป็นหน้าที่ ของ IC เบอร์ 74LS244 มาเป็นตัว เชื่อมต่อภาคระหว่างกัน เพื่อ ป้องกัน ความเสียหายที่อาจเกิดขึ้นได้นั่นเอง

2.10.1 การโปรแกรมโดยวิธีเก่า

1. กระบวนการเหมือนการโปรแกรม ROM
2. ไม่สะดวก และทำให้การออกแบบใช้เวลานาน
3. รองรับเฉพาะ EEPROM-based FPGA

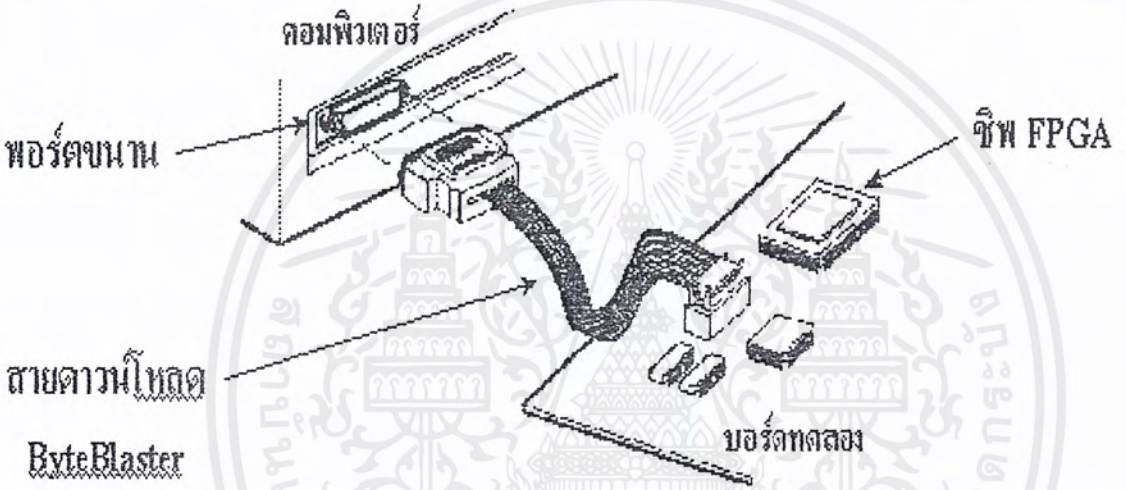


รูปที่ 2.14 แสดงการโปรแกรมลงบนบอร์ดทดลองแบบเก่า

ในการโปรแกรมแบบเก่านั้น มีวิธีการอยู่ว่า เมื่อใดก็ตามที่ต้องการ โปรแกรมลงไปในตัว Chip FPGA นั้นจะต้อง มีเครื่อง โปรแกรม หนึ่ง ตัว มี โปรแกรมพิเศษ เอาไว้ Download ค่าของ ข้อมูลตามรหัสมาตรฐานได้ แต่ไม่สะดวกนักเพราะต้องเปลี่ยนเอา Chip Max7000s ออกมาด้วยทุกครั้ง

2.10.2 การโปรแกรมแบบ In-System Programming

1. ชิพหลาย ๆ ตัวสามารถโปรแกรมได้พร้อมกัน โดยใช้สายดาวน์โหลดเดียวกัน
2. รองรับทั้ง EEPROM-Based และ SRAM-Based FPGA
3. สามารถโปรแกรมชิพได้ ขณะที่มันอยู่ในบอร์ดทดลอง



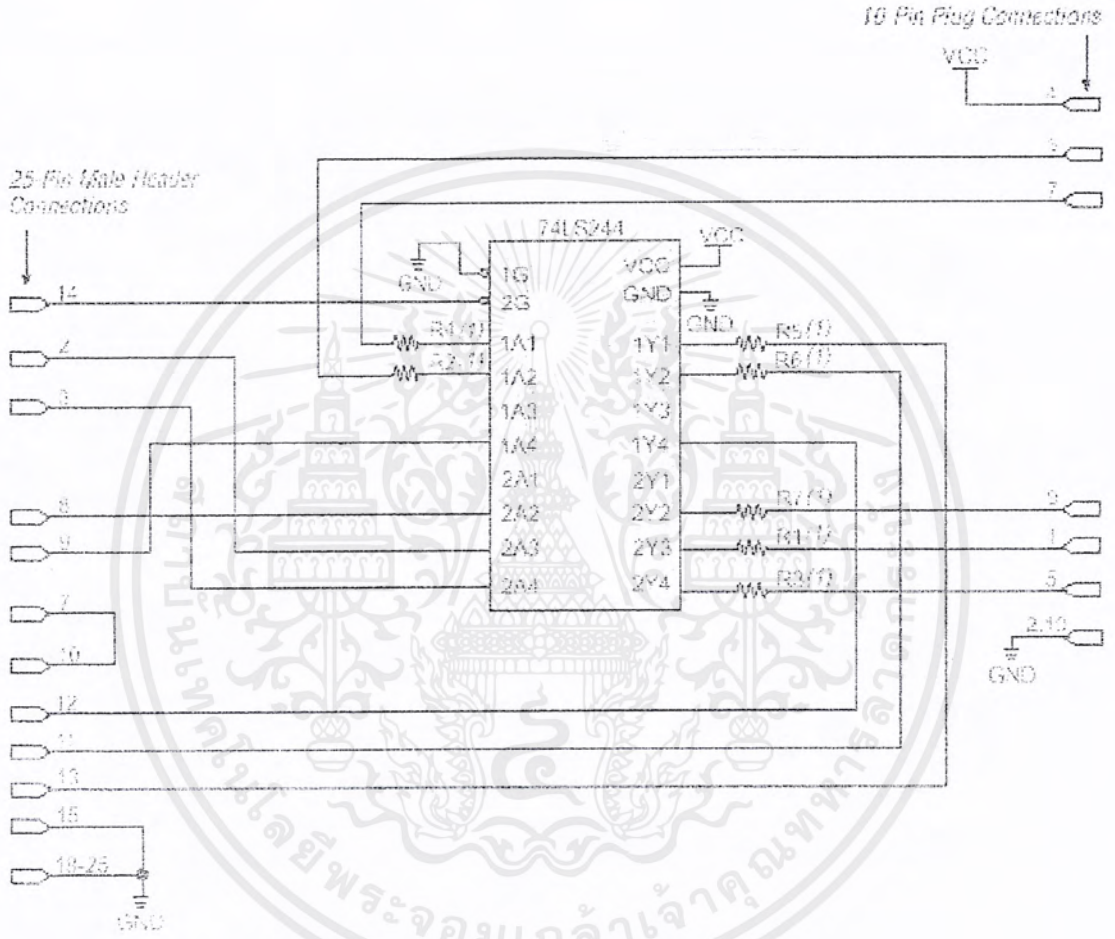
รูปที่ 2.15 แสดงการ โปรแกรม Chip FPGA MAX7000s แบบ Insystem Program

การโปรแกรมแบบใหม่นี้มีความสะดวกสบายกว่าแบบเดิมมาก เพราะไม่ต้องมา ถอดตัวอุปกรณ์เพื่อไปโปรแกรมกับเครื่องโปรแกรม ไม่ต้องมีอุปกรณ์เสริมใดๆ เพียงแต่ต้องต่อกับ Port COM LPT1 หรือ ก็คือ Port Printer ซึ่ง Computer ทุกเครื่องมี Parallel Port DB25 อยู่แล้ว จึงไม่ต้องมีอุปกรณ์เสริมเพิ่มขึ้นมาแต่อย่างใด

มีการใช้เทคนิคนี้กันอย่างแพร่หลายในเวลาต่อมา แต่อย่างไรก็ตามยังคงมีอีกมาตรฐานหนึ่งที่เป็นการ Down Load Data จาก คอมพิวเตอร์มาสู่ Board FPGA MAX7000s โดยการ ใช้สาย แบบ Bit Bluster แทน Byte Bluster ก็มีเหตุผลอยู่ว่าเมื่อ Comport ของ Computer ที่เรียกว่า LPT1 หรือ Palallel Port นั้นถูกใช้งานอยู่กับ Printer หรืออุปกรณ์อื่นใดอีกเช่น FAX/MODEM เป็นต้น

2.10.3 รูปแบบ FPGA MAX7000s ต่อ เข้ากับสายดาวน์โหลด ByteBlaster

เราสามารถทำสาย DownLoad ขึ้นมาได้เองโดย ต้องมีอุปกรณ์คือ IC 74LS244 จำนวน 1 ตัว และทำการเชื่อมต่อกับ DB25 Computer Port LPT1

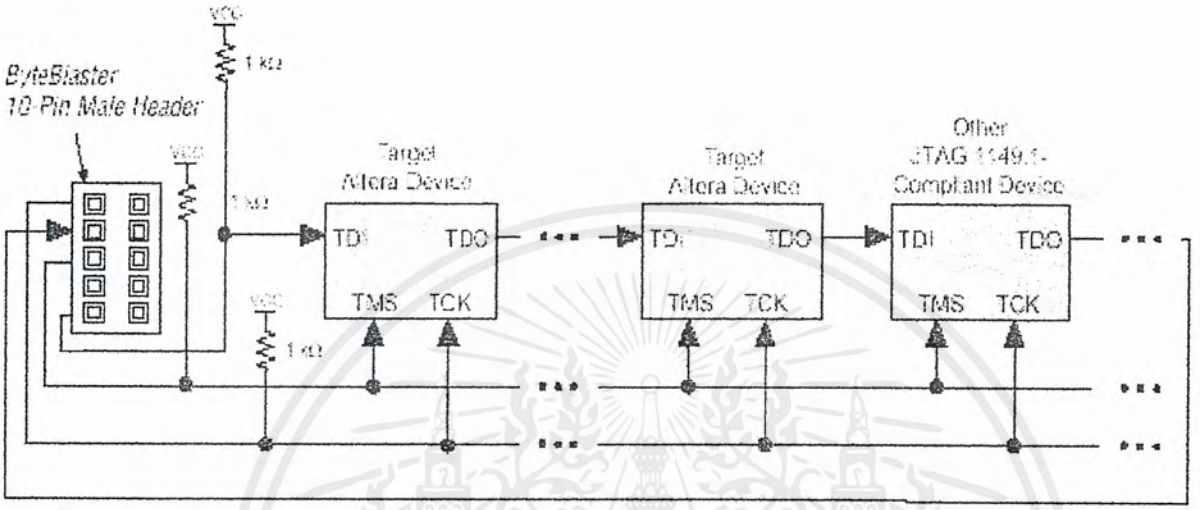


รูปที่ 2.16 แสดงวิธีการต่อสายดาวน์โหลด ByteBlaster

เมื่อจะต่อสายนี้กับ Altera Maxplus II จะต้องแยกออกมาข้างนอกในที่สุด เพราะถ้าเอาไว้ในวงจรอาจจะเปลืองเนื้อที่ค่อนข้างมาก เริ่มต้นจากการที่ตรวจสอบโดยใช้หลอด LED ได้โดยง่ายว่าควรจะนำสาย DownLoad ไปใช้ได้หรือไม่ เนื่องจากหากสาย DownLoad ที่เป็นของบริษัท Altera ได้ทำโดยออกแบบ ดีกว่าที่เป็นอยู่ กล่าวคือมีการ ป้องกันสนามไฟฟ้าด้วยอุปกรณ์ ทางอิเล็กทรอนิกส์

2.10.4 การต่อ Chip FPGA เข้ากับ สาย DownLoad ในโหมด JTAC

ในกรณีที่ผู้ใช้ต้องการที่จะ โปรแกรมลงบนตัว Chip FPGA แบบหลายๆตัวได้พร้อมๆกันของบริษัท Altera นั้น ก็เพื่อประโยชน์ ในการประหยัดเวลา กล่าวคือ อาจจะโปรแกรม ครั้งละ 100 ตัวเป็นต้น



รูปที่ 2.17 ในภาพนี้เป็นการต่อในโหมด JTAC ซึ่งสามารถต่อ Chip MAX7000s ได้ครั้งละ หลายๆตัวพร้อมๆกัน

วิธีการต่อควรจะไม่ลืมว่า การเชื่อมต่อกันนั้นไม่ได้ต่อเองตาม TDI เข้ากับ TDI เพราะว่าเป็นวิธีการที่ผิด ซึ่งเราจะไม่มีการต่อกันแบบ ขนานกันแต่อย่างไรเลย

ดังนั้นจึงต่อสาย TMS เข้าด้วยกัน และมีข้อกำหนดอยู่ว่า TDO ของตัวที่หนึ่งจะถูกส่งไปเป็น TDI ของอีกตัวถัดไปเช่นกันเป็นอย่างนี้ไปเรื่อยๆ จนถึง Chip ตัวสุดท้ายนั่นเอง

ยกตัวอย่างของกรณีมีจุด เชื่อมต่อแบบ JTAC OUT อยู่บน BOARD เราสามารถทำการ DOWN LOAD ข้อมูล ลงบนตัว Chip FPGA MAX7000s ได้แบบต่อเนื่องกันได้โดยการไปกำหนดโหมด ในการโปรแกรมใน MAX_PLUS II ซึ่งผู้ใช้ต้องกำหนดด้วยว่าจะโปรแกรม แบบ MAX7000 หรือว่าเป็น Chip FPGA เป็นแบบ FLEX10K เมื่อกำหนด File ที่ได้ทำการออกแบบ และผ่านขบวนการ TEST จนเกิดความแน่ใจดังนี้

1. มีไฟเลี้ยง +5 Volt
2. มีสาย DownLoad ข้อมูล หรือ Byte Bluster
3. Chip FPGA ต่ออยู่ในวงจรในลักษณะถูกต้อง
4. ทำการต่อสาย Byte Blaster กับ JTAC Chian Mode
5. สำคัญมากอย่าลืม ตั้งโหมด Comport COMPUTER LPT1 ในโหมด ECP เท่านั้น
6. ถ้าเกิดมีปัญหาเช่นเกิด Over Load ให้ทำการปลดแหล่งจ่ายออกก่อน
7. เมื่อ Down Load แล้วไม่ผ่านอาจเป็นไปได้ว่า Chip MAX7000s เสีย

2.10.5 ปัญหาที่เกิดกับ Chip FPGA MAX7000s เบอร์ EPM7128SLC84-10

1.การทำงานไม่แน่นอนเช่นในเรื่องความถี่ที่ไม่เที่ยงตรง

2.ไม่สามารถโปรแกรมลงตัว MAX7000s ได้

3.โปรแกรม ALTERA MAXPLUS II ไม่สามารถติดต่อ CHIP MAX7000S โดยเมื่อเลือก Detec เพื่อเปลี่ยนชื่อ EPM7128SLC84-10 Chip

4.Chip MAX7128 ร้อนมากทำให้อุปกรณ์ประกอบเสียหายได้

บทที่ 3
ภาษา AHDL

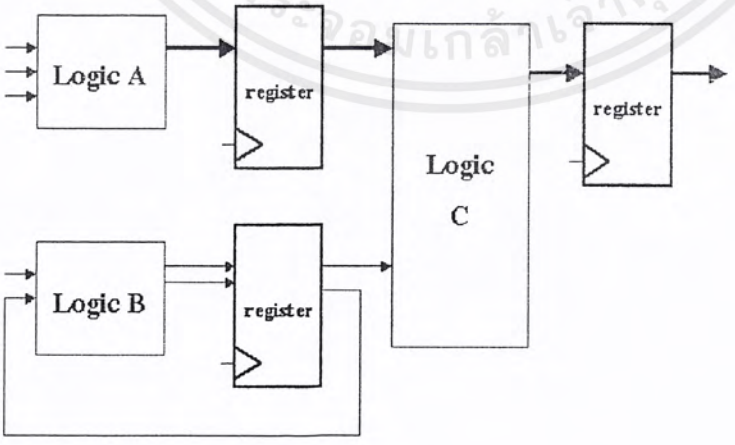
3.1 วงจรคอมบิเนชั่น (Combinational Logic)

3.1.1 Hardware Description Languages (HDL)

1. เป็นภาษาที่สามารถอธิบายการทำงานของฮาร์ดแวร์ (วงจรดิจิตอล) ได้
2. แต่เดิม HDL ถูกใช้สำหรับโมเดลการทำงานของวงจรดิจิตอล
3. ต่อมา ด้วยเทคโนโลยีของการสังเคราะห์ลอจิก (Logic Synthesis) ทำให้ HDL ถูกใช้สำหรับการออกแบบวงจรในไอซีดิจิตอล
4. HDL เป็นมิติใหม่ของการออกแบบวงจรดิจิตอล นอกเหนือจากการใช้แผนภาพวงจร (Schematics) เพราะทำให้ผู้ออกแบบ สามารถออกแบบในระดับพฤติกรรมของวงจรได้มากขึ้น โดยไม่จำเป็นลงลึกไปในรายละเอียดของการต่อกันของเกตต่าง ๆ

3.1.2 ข้อจำกัดของ Hardware Description Languages (HDL)

1. อย่างไรก็ตามเทคโนโลยีของการสังเคาะห์ลอจิกในปัจจุบัน ยังไม่สามารถทำให้เราเขียน HDL ในระดับพฤติกรรมได้ 100%
2. การออกแบบวงจรดิจิตอลด้วย HDL ในปัจจุบัน รองรับรูปแบบการเขียน 2 แบบ คือ
   - 1) แบบ Registry Transfer Level (RTL) เป็นการเขียนที่ต้องแสดงถึง REGISTER (หรือ FLIP FLOP) ที่ใช้ในวงจร และการทำงานของลอจิกต่าง ๆ ที่รายล้อมมัน
   - 2) แบบโครงสร้างการต่อกันของอุปกรณ์ต่าง ๆ (Structure Level) โครงสร้างนี้เทียบเท่ากับการใช้ไฟล์ Netlist หรือแผนภาพ schematic



รูปที่ 3.1 แสดงตัวอย่างลักษณะโครงสร้างแบบ RTL

3.2 HDL ที่เป็นมาตรฐานสากล

1. VHDL (IEEE1076-1987 and IEEE1076-1993) ย่อมาจาก Very high speed IC HDL
2. Verilog HDL (IEEE1364-1995)
3. มีข้อดี คือ ผู้ผลิตซอฟท์แวร์สำหรับออกแบบไอซี และ FPGA หลาย ๆ รายรองรับภาษาที่เป็นมาตรฐาน ทำให้ซอฟท์แวร์ต่างผู้ผลิตสามารถใช้ไฟล์ร่วมกันได้
4. นอกจากนี้ การใช้ภาษามาตรฐานจะทำให้สามารถนำวงจรที่ออกแบบสำหรับ FPGA ไปทำเป็น ASIC ได้ในภายหลังโดยไม่ยากนัก

3.3 ภาษา AHDL

ย่อมาจาก Altera HDL เป็นภาษาที่ใช้สำหรับออกแบบวงจร FPGA ของ Altera เท่านั้น
ข้อดี คือ เป็นภาษาที่ง่าย และมีรายละเอียดไม่มาก
เหมาะสำหรับผู้เริ่มต้นเรียนรู้ HDL ซึ่งจะสามารถนำแนวทางไปศึกษาภาษา HDL อื่นได้ไม่ยากนัก

3.3.1 กุญแจของการออกแบบด้วย HDL สำหรับผู้เริ่มต้น

- "THINK HARDWARE"
- เราคุ้นเคยกับการใช้ภาษาออกแบบซอฟท์แวร์ แต่ระวัง! การใช้ HDL ถึงแม้จะมีคำสั่งหลายคำสั่งคล้าย ๆ กับ--คำสั่งในภาษาสูงทั่วไป เช่น ซี หรือ ปาสคาล แต่กลไกการใช้งานไม่เหมือนกันทีเดียว
- HDL ไม่ใช่เครื่องมือที่แปลงอัลกอริธึมทางซอฟท์แวร์ให้ไปเป็นฮาร์ดแวร์ (ซึ่งเป็นความฝันของนักออกแบบแต่ทว่ามันยังทำไม่ได้ในปัจจุบัน)
- HDL เป็นเครื่องมือสำหรับช่วยอธิบายฮาร์ดแวร์ ผู้เริ่มต้นจึงควรนึกเสมอถึงลักษณะโครงสร้างของฮาร์ดแวร์ที่กำลังออกแบบ

3.3.2 ส่วนประกอบสำคัญของไฟล์ AHDL

1.ส่วนหัว (SUBDESIGN Section)

_ บอกถึงขา INPUT และ OUTPUT ของวงจร

2.ส่วนแสดงตัวแปรที่ใช้ (VARIABLE Section)

_ บอกถึงโหนด หรือ อุปกรณ์สำเร็จรูป (เช่น D filp-flop, Counter, และอื่น ๆ ที่มีใน ไลบรารี่) ที่จะเรียกเข้ามาใช้ในวงจร โดยจะตั้งชื่อให้อุปกรณ์ทุก ๆ ชิ้นที่จะใช้

3.ส่วนลอจิก ซึ่งอธิบายการทำงานของวงจร (Logic Section)

_ เป็นส่วนที่อธิบายการทำงานของวงจร และการต่อกันของอุปกรณ์ต่าง ๆ

ตัวอย่าง decode1.tdf

```
SUBDESIGN decode1
(
    address[15..0]  : INPUT;
    chip_enable     : OUTPUT;
)
BEGIN
    chip_enable = (address[ ] = = H"0370");
END;
```

3.3.3 GROUP คือ BUS

สัญญาณที่เป็น BUS หรือเรียกอีกชื่อว่า GROUP ให้ใช้เครื่องหมายปีกกาสี่เหลี่ยม [ ] ตามหลังชื่อสัญญาณ
ตัวอย่างการสร้าง BUS ในส่วน SUBDESIGN
เช่น address[15..0] : INPUT เป็นการสร้างสัญญาณขาเข้า 16 บิต ชื่อ address15, address14, … address0
การเรียกใช้งาน address15 หมายถึง สัญญาณ 1 บิต (บิตบนสุด)
address[6..0] หมายถึง BUS 7 บิต ประกอบด้วย address6, address5, …, address0
(address6,address4,address2) หมายถึง BUS 3 บิต ประกอบด้วยสัญญาณตามที่ระบุ
address[ ] หมายถึง BUS ที่ประกอบด้วยทุกบิตของ address ในที่นี้ คือ เทียบเท่ากับaddress[15..0]

3.3.4 EQUAL and Comparator Operator

เครื่องหมาย = หมายถึง การจับสัญญาณมาต่อกัน โดยด้านขวาเป็นอินพุตของด้านซ้าย จำนวนบิตของสัญญาณด้านซ้าย และขวาต้องเท่ากัน
ยกเว้นการต่อกับ 1 บิต เช่น
d[2..0] = VCC; ให้ทุกบิตเท่ากับ 1 (VCC กับ 1 ใช้แทนกันได้)
เครื่องหมาย = = หมายถึง การเปรียบเทียบว่าเท่ากันหรือไม่ ซึ่งได้ผลลัพธ์เป็น 1 บิต เช่น

```
chip_enable = (address[ ] = = H"0370");
```

ตัวเปรียบเทียบอื่น ๆ ได้แก่ >, <, <=, >=, !=

ตัวอย่าง boole1.tdf

```
SUBDESIGN boole1
(
    a0, a1, b       : INPUT;
    out1, out2      : OUTPUT;
)
BEGIN
    out1 = a1 & !a0;
    out2 = out1 # b;
END;
```

3.4 Arithmetic and Boolean Operators

- ใช้ได้กับสัญญาณบิตเดียว หรือ group ก็ได้
- \+ บวก - ลบ
- ! NOT
- & AND !& NAND
- # OR !# NOR
- $ XOR !$ XNOR
- เช่น a[3..0] + b[4..1]
- เช่น a[3..0] & b[4..1]
- เช่น a[3..0] & c

ตัวอย่าง boole2.tdf

```
SUBDESIGN boole2
(   a0, a1, b       : INPUT;
    out, out2       : OUTPUT; )
VARIABLE
    a_equals_2      : NODE;
BEGIN
    a_equals_2 = a1 & !a0;
    out = a_equals_2 # b;
```

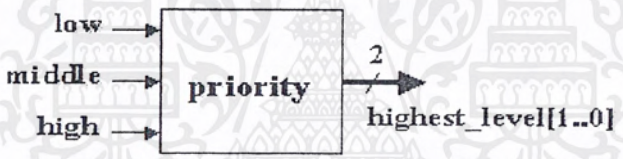
```
    out2 = a_equals_2 $ b;
END;
```

3.5 Combinational Logic

ที่ผ่านมายังอยู่ในส่วนของวงจร combinational ทั้งสิ้น (ยังไม่มีสัญญาณนาฬิกามาเกี่ยวข้อง)
มีคำสั่งพิเศษ 4 คำสั่ง ที่ช่วยให้เราออกแบบวงจร combinational ได้โดยอธิบายพฤติกรรมของมัน คำสั่งเหล่านี้คือ

- IF
- CASE
- TABLE
- FOR

ตัวอย่าง priority.tdf (คำสั่ง IF)



รูปที่ 3.2 แสดงตัวอย่างการออกแบบวงจร Priority

```
SUBDESIGN priority
(
    low, middle, high    : INPUT;
    highest_level[1..0]  : OUTPUT;
)
BEGIN
        IF high THEN
          highest_level[ ] = 3;
        ELSIF middle THEN
          highest_level[ ] = 2;
        ELSIF low THEN
          highest_level[ ] = 1;
```

```
        ELSE
          highest_level[ ] = 0;
        END IF;
END;
```

ตัวอย่าง decoder.tdf (คำสั่ง CASE)

```
SUBDESIGN decoder
(   code[1..0]      : INPUT;
    out[3..0]       : OUTPUT;)
BEGIN
    CASE  code[ ]  IS
          WHEN 0 => out[ ] = B"0001";
          WHEN 1 => out[ ] = B"0010";
          WHEN 2 => out[ ] = B"0100";
          WHEN 3 => out[ ] = B"1000";
    END CASE;
END;
```

ตัวอย่าง decode3.tdf (คำสั่ง TABLE)

```
SUBDESIGN decode3
(   addr[15..0], m/io           : INPUT;
    rom, ram, print, sp[2..1] : OUTPUT;)
BEGIN
  TABLE
    m/io, addr[15..0]                    => rom,  ram,  print,  sp[];
     1,   B"00XXXXXXXXXXXXXX"    =>   1,    0,    0,    B"00";
     1,   B"100XXXXXXXXXXXXX"    =>   0,    1,    0,    B"00";
     0,   B"0000001010101110"     =>   0,    0,    1,    B"00";
     0,   B"0000001011011110"     =>   0,    0,    0,    B"01";
     0,   B"0000001101110000"     =>   0,    0,    0,    B"10";
  END TABLE;
END;
```

3.5.1 ค่า default ของสัญญาณขาออก

สำหรับคำสั่ง IF, CASE, และ TABLE ในกรณีที่เงื่อนไขแจกแจงไม่ครบทุกกรณีของ input จะได้ว่า กรณีที่เหลืออยู่ทั้งหมดจะมีค่า output เป็น 0 หมดเช่น IF a THEN b[1..0] = 2; c = 1; END IF;
กรณีกรณีที่เหลืออยู่ คือ a=0 ซึ่งก็จะให้ทั้ง b[ ] และ c เป็น 0

3.5.1.1 การเปลี่ยนค่า default ของสัญญาณขาออก

ถ้าต้องการค่า default ที่ไม่ใช่ 0 ทำได้ดังนี้

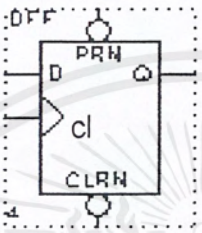
- สำหรับคำสั่ง IF ใช้ ELSE ระบุกรณีอื่น ๆ
- สำหรับคำสั่ง CASE ใช้ WHEN OTHERS ระบุกรณีอื่น ๆ
- สำหรับคำสั่ง TABLE ใช้คำสั่ง DEFAULT ระบุค่า default
- คำสั่ง DEFAULT ยังสามารถใช้ระบุค่า default สำหรับการใช้ IF หรือ CASE ได้ด้วย

ตัวอย่าง default1.tdf (คำสั่ง DEFAULTS)

```
BEGIN
   DEFAULTS
        ascii_code[ ] = B"00111111";
        enable = VCC;
   END DEFAULTS;
   TABLE
        i[3..0]   =>      ascii_code[ ],   enable;
        B"1000" =>        B"01100001",   0;
        B"0100" =>        B"01100010",   0;
        B"0010" =>        B"01100011",   0;
        B"0001" =>        B"01100100",   0;
   END TABLE;
END;
```

3.6 วงจรประเภท Sequential

1. เป็นวงจรที่ทำงานตามจังหวะของ clock
2. ประกอบด้วยวงจร Combinational ผสมกับ Flip Flop
3. กุญแจในการออกแบบวงจร Sequential ใน AHDL คือ การใช้ตัวแปรประเภท Flip Flop
4. ตัวแปรที่สำคัญที่สุด คือ DFF



รูปที่ 3.3 สัญญลักษณ์ของ DFF (D Flip Flop)

เป็นไลบรารี่แบบ Primitive ใน AHDL

ทำหน้าที่เหมือนเป็นหน่วยความจำ 1 บิต หรือ รีจีสเตอร์

สามารถใช้งานเป็น GROUP เพื่อเป็นรีจีสเตอร์แบบหลายบิตได้

การทำงาน

- เมื่อ clk -> q = d
- ณ เวลาอื่น ๆ -> q คงค่าเดิม
- ขา prn เป็น active-low async preset
  (prn=0 -> q=1)
- ขา clrn เป็น active-low async clear
  (clrn=0 -> q=0)

3.6.1 การใช้งาน DFF

ตั้งชื่อในส่วน VARIABLE

เช่น a : DFF; (ตั้ง a เป็น DFF ขนาด 1 บิต)

b[2..0] : DFF; (ตั้ง b เป็น DFF 3 บิต ประกอบด้วย b2,b1,b0)

การเรียกใช้งานในส่วน Logic ใช้รูปแบบ “ชื่อตัวแปร . ชื่อขา” เช่น

- b2.clk หมายถึง สัญญาณขา clk ของ b2

_ b[ ].clk หมายถึง ขา clk ของ b[2..0] b[ ].q หมายถึง ขา q ของ b[2..0]

ขา clk, d, clrn, prn เป็นขา INPUT -> ต้องอยู่ด้านซ้ายของสมการเสมอ

ขา q เป็นขา OUTPUT -> ต้องอยู่ด้านขวาของสมการเสมอ

เช่น

_ b[ ].clk = clock; ถูก

_ b2.d = VCC; ถูก

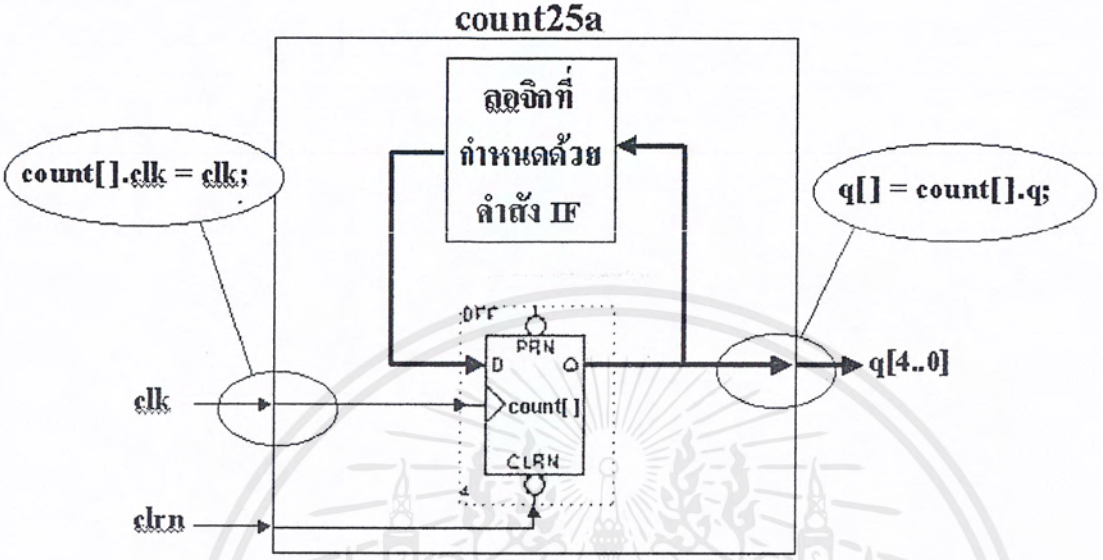
_ b[ ].d = b[ ].q + 1; ถูก

_ out = b[ ].d; ผิด

_ b2.q = VCC ผิด

ตัวอย่าง: วงจร count25a.tdf (ตัวนับ 0-24)

```
SUBDESIGN count25a
(   clk, clrn       : INPUT;
    q[4..0]                 : OUTPUT; )

VARIABLE
    count[4..0]     : DFF;
BEGIN
        count[ ].clk = clk;
        count[ ].clrn = clrn;

        IF  count[ ].q = = 24 THEN
          count[ ].d = 0;
        ELSE
          count[ ].d = count[ ].q + 1;
        END IF;

        q[ ] = count[ ].q;
END;
```



รูปที่ 3.4 แผนภาพวงจรสมมูลของ count25a

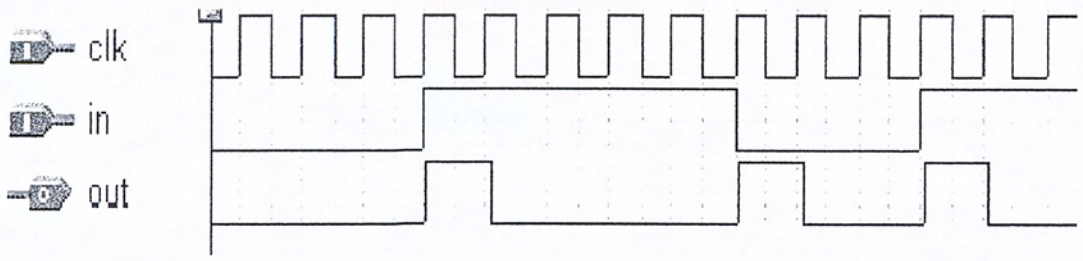
ตัวอย่าง: วงจร ahdlcnt.tdf

```
SUBDESIGN ahdlcnt
(   clk, load, ena, clr, d[15..0]   : INPUT;
    q[15..0]                        : OUTPUT;)
VARIABLE
    count[15..0]    : DFF;
BEGIN
        count[ ].clk = clk;
        count[ ].clrn = !clr;

        IF load THEN
          count[ ].d = d[ ];
        ELSIF ena THEN
          count[ ].d = count[ ].q + 1;
        ELSE
```

```
        count[ ].d = count[ ].q;
    END IF;

    q[ ] = count[ ];
END;
```

ตัวอย่าง: วงจร edgedet.tdf ตรวจจับขอบสัญญาณ

```
SUBDESIGN edgedet
(   in, clk         : INPUT;
    out   : OUTPUT; )

VARIABLE
    in_ff : DFF;
BEGIN
    in_ff.clk = clk;
    in_ff.d = in;

    IF  in  !=  in_ff.q THEN
        out = VCC;
    ELSE
        out = GND;
    END IF;
END;
```



รูปที่ 3.5 แสดง Timeming ตรวจจับขอบสัญญาณ

3.7 การนำวงจรที่สร้างขึ้นไปใช้ในไฟล์อื่น

วงจรที่ใด ๆ ที่ทำเสร็จแล้ว สามารถทำเป็นไลบรารี่ เพื่อนำไปเรียกใช้ได้ในวงจรอื่น

วิธีทำ

_ ถ้าต้องการทำเป็นไลบรารี่ที่เรียกใช้ได้ใน schematic

ให้ ใช้เมนู File | Create Default Symbol สร้างไฟล์ symbol (*.sym)

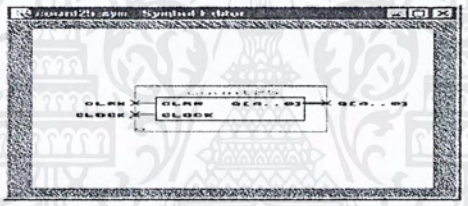
_ ถ้าต้องการทำเป็นไลบรารี่ที่เรียกใช้ได้ใน AHDL

ให้ ใช้เมนู File | Create Default Include File สร้างไฟล์ include (*.inc)

ไฟล์ Symbol และ Include

ภายในไฟล์ Symbol และ Include ที่สร้างขึ้น มีเพียงข้อมูลของพอร์ต INPUT และ OUTPUT ของวงจร และเป็นตัวชี้ไปยังไฟล์ที่เป็นวงจร (tdf หรือ gdf) อีกทีหนึ่ง

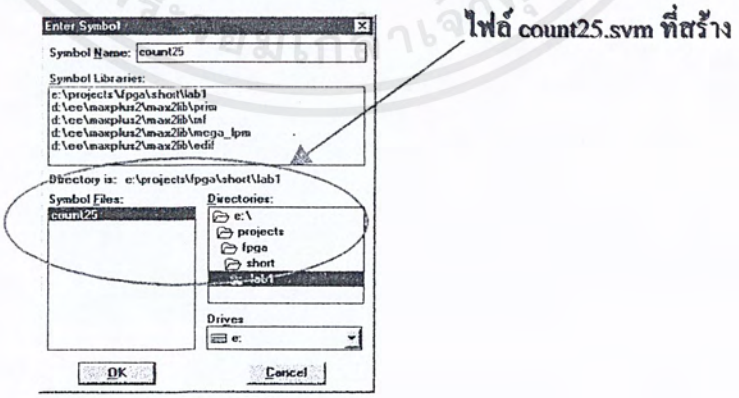
เราสามารถเปิดไฟล์ Symbol ขึ้นมาแก้ตำแหน่งของขาได้



รูปที่ 3.6 แสดงการ **Create Default Symbol**

3.7.1 การเรียกไลบรารี่มาใช้ในไฟล์ Schematic

■ ทำเหมือนการเรียกอุปกรณ์ทั่ว ๆ ไปมาใช้ในวงจร



รูปที่ 3.7 แสดงการเรียกไลบรารี่มาใช้ในไฟล์ Schematic

3.7.2 การเรียกไลบรารี่มาใช้ในไฟล์ AHDL

1. เหมือนวิธีใช้ DFF
2. เพียงเพิ่มคำสั่ง INCLUDE
3. ใช้ "ชื่อตัวแปร . ชื่อขา"
4. ต้องรู้ชื่อขาของอุปกรณ์
   - ในที่นี้ชื่ออุปกรณ์ คือ count25
   - ชื่อตัวแปร คือ counter
   - มีขา INPUT ชื่อ clock, clrn
   - มีขา OUTPUT ชื่อ q[4..0]

```
INCLUDE "count25";
SUBDESIGN test
(       clk, clr     :  INPUT;
        out[4..0]    :  OUTPUT; )

VARIABLE
        counter      :  count25;

BEGIN
        counter.clock = clk;
        counter.clrn = !clr;
        out[ ] = counter.q[ ];
END;
```

3.7.2.1 การเรียกใช้แบบ In-Line Reference

-ไม่ต้องตั้งเป็นตัวแปร

-การเรียกใช้กระทัดรัด แต่ต้องระวัง

ลำดับของ INPUT และ OUTPUT

ในที่นี้ ไฟล์ count25.inc เป็นดังนี้

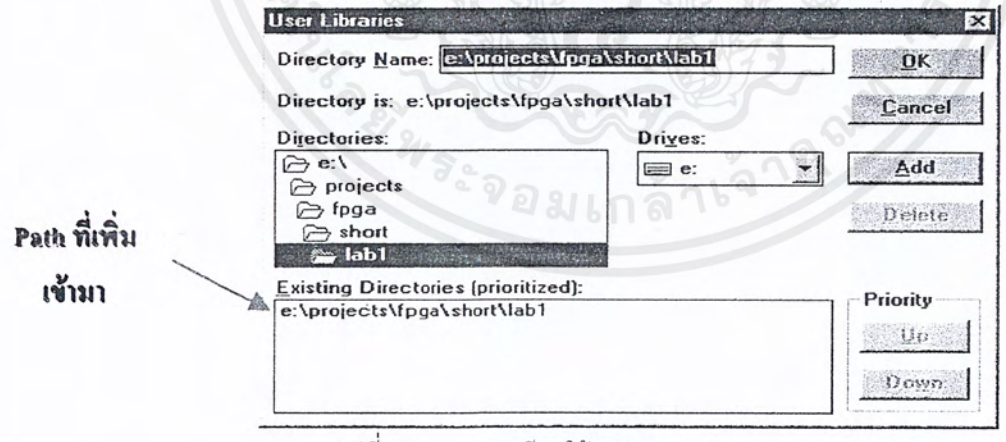
```
FUNCTION count25 (clrn, clock)
   RETURNS (q[4..0]);
```

```
INCLUDE "count25";

SUBDESIGN test2
(       clk, clr     :  INPUT;
        out[4..0]    :  OUTPUT; )

BEGIN
        out[ ] = count25(!clr, clk);
END;
```

- หมายเหตุ อาจใส่คำสั่ง FUNCTION ที่อยู่ใน *.inc นี้เข้าไปโดยตรงใน ไฟล์ที่ออกแบบ แทนการใช้คำสั่ง INCLUDE ก็ได้

User Library Path

- วงจรที่จะเรียกมาใช้ได้ต้องอยู่ในไดเรกทอรีปัจจุบัน (ไดเรกทอรีเดียวกันกับโปรเจคที่ทำงานอยู่) หรือ อยู่ใน User Library Path
- เพิ่ม Library Path ได้โดยใช้เมนู Options | User Libraries



รูปที่ 3.8 แสดงการเรียกใช้ Library Path

3.8 ไลบรารี่ประเภท LPM

LPM = Library of Parameterized Modules เป็นไลบรารี่สำเร็จรูปที่ปรับพารามิเตอร์ได้ตามต้องการ เช่น

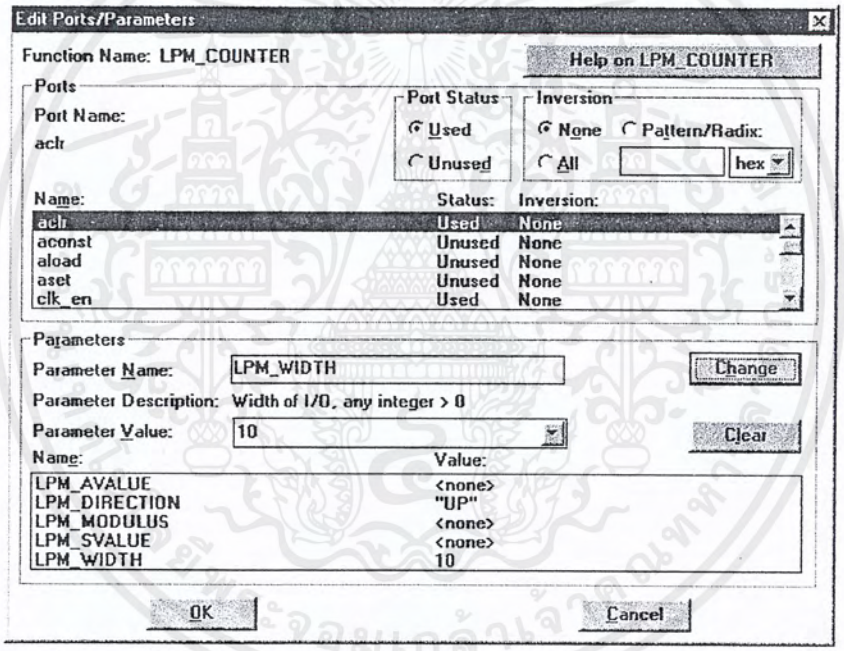
_ lpm_counter เป็นตัวนับที่เลือกได้ว่าเป็นกี่บิต, มีขา up/down หรือไม่, โหลดค่าเข้าได้หรือไม่, ฯลฯ

- lpm_mult เป็นตัวคูณที่เลือกได้ว่าเป็นกี่บิต, คิดเครื่องหมายหรือไม่, ฯลฯ
- lpm_ram_io เป็นหน่วยความจำ (RAM) ที่เลือกขนาดได้
- lpm_shiftreg เป็น Shift Register ที่ปรับจำนวนบิต และทิศทางได้

■ การใช้ LPM มีข้อดี คือ

- สะดวก
- ใช้ทรัพยากรในชิพเท่าที่จำเป็น โดยใส่พารามิเตอร์ให้พอดีกับที่ต้องการ
- ทำให้การสังเคราะห์วงจรบางอย่างทำได้อย่างมีประสิทธิภาพ เช่น การใช้ RAM

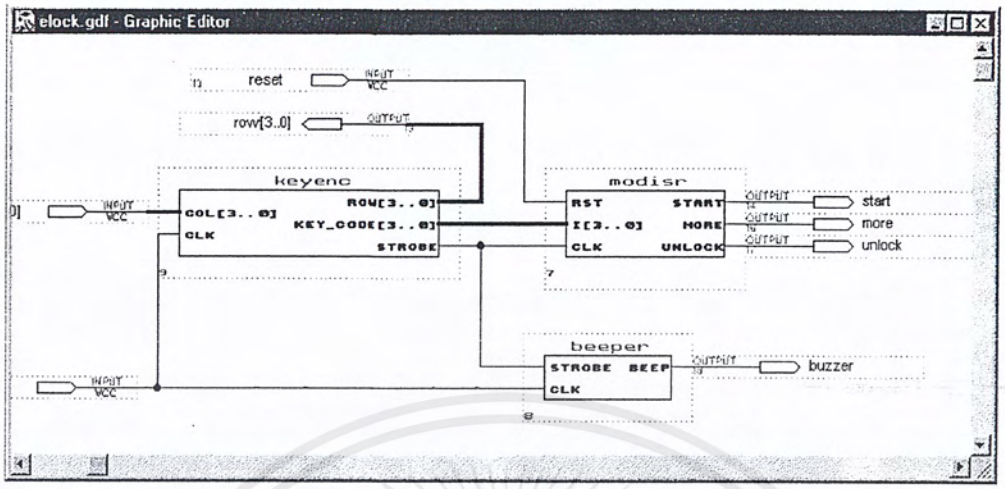
3.9 การเรียกใช้ LPM ใน Schematic



รูปที่ 3.9 การเรียกใช้ LPM ใน Schematic

3.10 การออกแบบอย่างมีโครงสร้าง

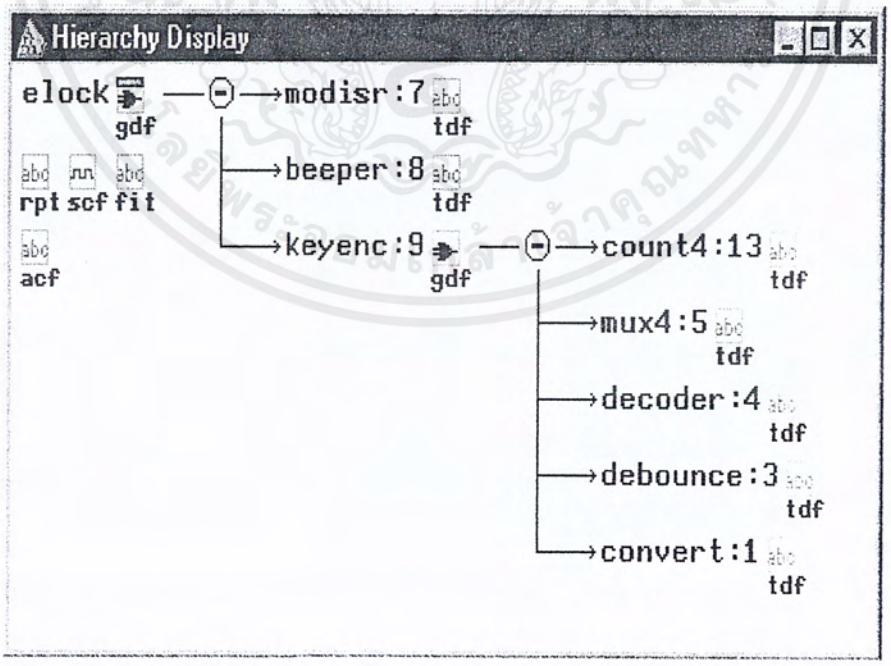
โปรเจคที่มีความซับซ้อน ควรแยกการทำงานออกเป็นวงจรย่อย ๆ แล้วแยกออกแบบวงจรย่อย ๆ เหล่านั้นโดยใช้ Schematic หรือ AHDLจากนั้น สร้างโปรเจกบนสุดเป็น schematic เพื่อนำวงจรต่าง ๆ มาต่อกัน เหมือนกับว่าเรามี การทำแบบแยกส่วนคือทำจากส่วนเล็กไปหาส่วนใหญ่หรือทำจากส่วนย่อยไปหาส่วนรวมซึ่งเทคนิคแบบนี้เราอาจพบเห็นจากการเขียนโปรแกรมในภาษาสูงๆ ทั่วๆไปนั่นเอง การทำอย่างนี้ได้ประโยชน์อยู่ 2 อย่าง คือ ทำการตรวจสอบได้ง่าย อีกประการหนึ่งคือสามารถทำงานเป็นทีมได้



รูปที่ 3.10 แสดงการออกแบบ แบบมีโครงสร้าง

3.10.1 การออกแบบอย่างมีโครงสร้าง Hierarchy

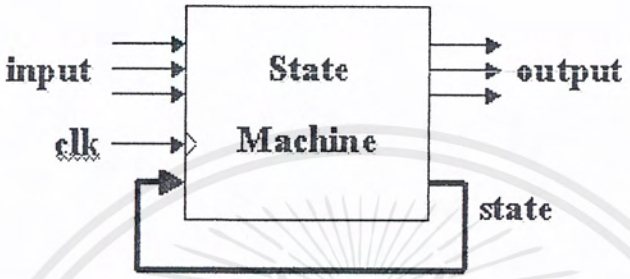
ใช้เมนู MAX+PlusII | Hierarchy Display จะเห็นโครงสร้างของการใช้ไฟล์ต่าง ๆ ในโปรเจค ดังนี้



รูปที่ 3.11 การออกแบบอย่างมีโครงสร้าง Hierarchy

3.11 State Machines

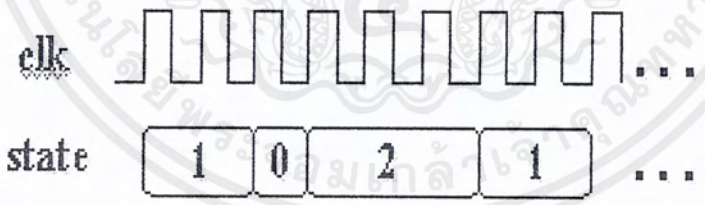
คือ วงจร sequential ซึ่งออกแบบโดยแบ่งสถานะการทำงาน (สเตท) ของ วงจร ออกเป็นหลาย ๆ สถานะ แต่ละสเตทมีลอจิกการทำงานที่ต่างกัน เพื่อกำเนิดค่าเอาต์พุต และค่า สเตทถัดไป มีสัญญาณสเตท ที่กำหนดว่า สเตทปัจจุบันเป็นสเตทไหน



รูปที่ 3.12 แสดงโครงสร้างของ State Machine

3.11.1 ลักษณะการคิด State Machines

สัญญาณเสตทจะถูกเก็บในรีจีสเตอร์ ดังนั้น สเตทจะสามารถเปลี่ยนแปลงได้ที่ขอบของ clock เท่านั้น เช่น



รูปที่ 3.13 แสดง Timing Diagram ของ State Machines

3.11.2 ประเภทของ State Machine

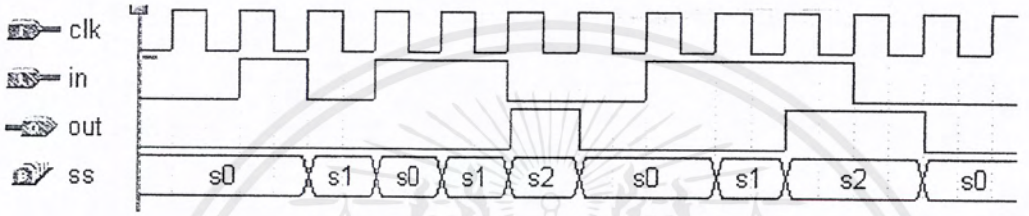
1. Moore Machine มีเอาต์พุตเปลี่ยนแปลงตามจังหวะของ clock เท่านั้น (แต่ละ สเตทมีค่าเอาต์พุตที่กำหนดแน่นอน, เอาต์พุตจะเปลี่ยนก็ต่อเมื่อเสตทเปลี่ยน)

2. Mealy Machine เอาต์พุตไม่จำเป็นต้องเปลี่ยนแปลงตามจังหวะของ clock (ภายใน สเตทหนึ่ง ๆ เอาต์พุต สามารถเปลี่ยนแปลงได้ ถ้าอินพุตเปลี่ยน)

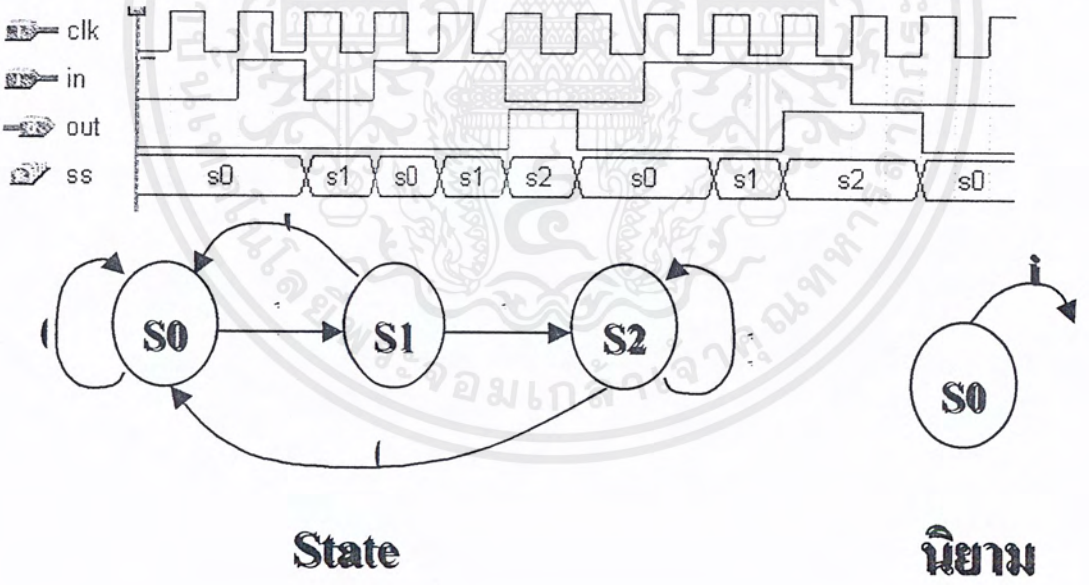
ตัวอย่าง: moore.tdf

■ วงจรตรวจจับลำดับสัญญาณ ถ้า in = 1 ติดกัน 2 จังหวะ จะให้

out = 1



รูปที่ 3.14 แสดง Timing Diagram วงจรตรวจจับลำดับสัญญาณ ถ้า in = 1 ติดกัน 2 จังหวะ จะให้ out = 1

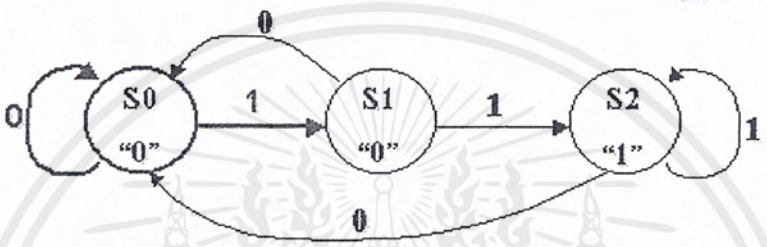


รูปที่ 3.15 แสดง State Diagram ของการออกแบบวงจรตรวจจับสัญญาณ

3.12 ความหมายของ State Diagram

ตัวอย่างการตีความที่สเตท S0

- ที่ S0 วงจรจะให้ out = 0
- ที่ของขาขึ้นของ clk ถ้า in = 1 จะเปลี่ยนสเตทไปเป็น S1
  มิฉะนั้น จะอยู่สเตท S0 เหมือนเดิม



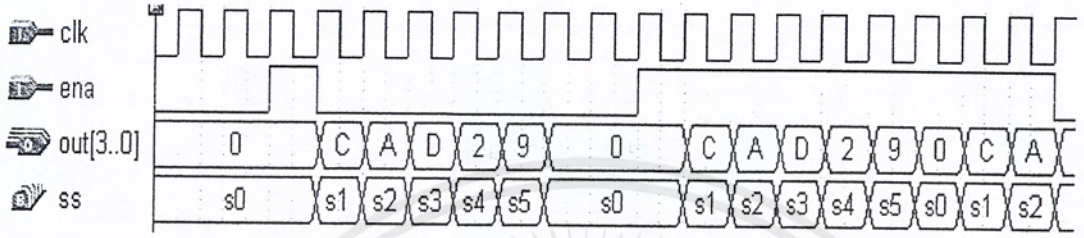
รูปที่ 3.16 แสดงวิธีการตีความหมายของ State Machine

ตัวอย่าง: moore.tdf โดยใช้คำสั่ง TABLE

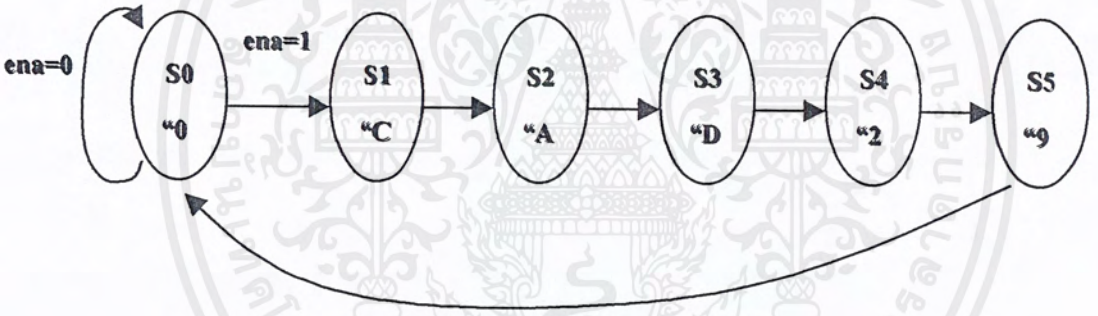
```
SUBDESIGN moore
(   clk, in  : INPUT;
    out    : OUTPUT;)
VARIABLE
        ss : MACHINE OF BITS (out)
            WITH STATES (s0=0, s1=0, s2=1);
BEGIN
        ss.clk = clk;
        TABLE
          ss,       in        =>      ss;
          s0,       1         =>      s1;
          s1,       1         =>      s2;
          s1,       0         =>      s0;
          s2,       0         =>      s0;
        END TABLE;
END;
```

ตัวอย่าง: patgen.tdf

■ วงจรกำเนิดรูปแบบลำดับสัญญาณ เมื่อมี ena กระตุ้น เช่น ต้องการกำเนิดลำดับสัญญาณ 4 บิต มีค่า C, A, D, 2, 9 (ฐาน 16)



รูปที่ 3.17 แสดงการจำลองการทำงานของวงจรกำเนิดรูปแบบลำดับสัญญาณ

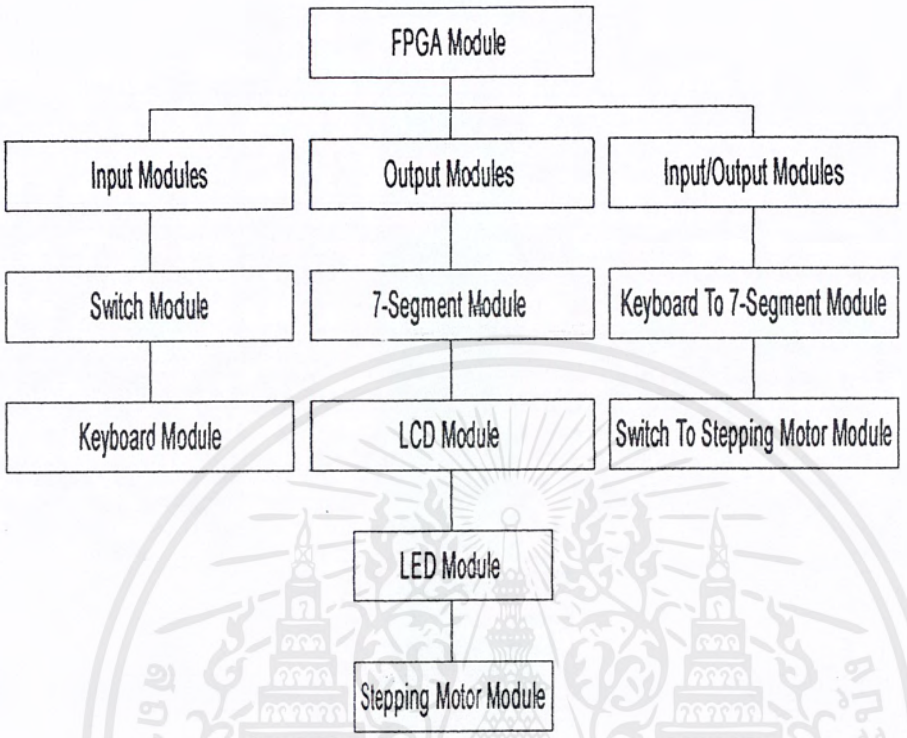


รูปที่ 3.18 แสดง State Diagram ของวงจรกำเนิดรูปแบบลำดับสัญญาณ

```
SUBDESIGN patgen
(   clk, ena        : INPUT;
    out[3..0]       : OUTPUT; )
VARIABLE
   ss : MACHINE WITH STATES (s0, s1, s2, s3, s4, s5);

BEGIN
    ss.clk = clk;
```

```
    TABLE
        ss,     ena     =>      ss,     out[ ];
        s0,     1       =>      s1,     H"0";
        s0,     0       =>      s0,     H"0";
        s1,     x       =>      s2,     H"C";
        s2,     x       =>      s3,     H"A";
        s3,     x       =>      s4,     H"D";
        s4,     x       =>      s5,     H"2";
        s5,     x       =>      s0,     H"9";
    END TABLE;
END;
```

บทที่ 4
การออกแบบฮาร์ดแวร์



รูปที่ 4.1 แสดงถึงโมดูลต่างๆ

จากรูปที่ 4.1 เป็นส่วนต่างๆที่ได้ออกแบบไว้เพื่อแสดงถึงการประยุกต์การใช้งานโมดูลที่ได้ออกแบบสำหรับ แต่ละส่วนได้แยกการออกแบบไว้ดังนี้คือ

- Input Modules
    1. Switch Module
    2. Keyboard Module
- Output Modules
    1. 7-Segment Module
    2. LCD Module
    3. LED Module
    4. Stepping Motor Module
- Input/Output Modules
    1. Keyboard To 7-Segment Module
    2. Switch To Stepping Motor Module

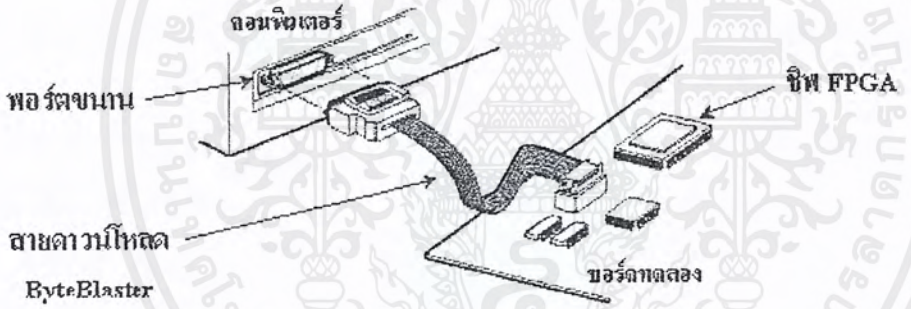
ในการทดลองผู้ทดลองสามารถคิด พลิกแพลงสถาณการ์ณให้เป็นแบบที่ใจ ผู้ทดลองปราถนาได้ดังใจชอบเพราะในการนำไปใช้งานจริงๆอาจมีความหลากหลายและซับซ้อนกว่านี้มากๆ เป็นหลายเท่านัก สิ่งสำคัญที่สุดคือต้องทราบว่า วงจรที่ได้ออกแบบมี Input/output ต่ออยู่กับ Chip FPGA ขาไหนอย่างไร โดยดูได้จากวงจรรวม ในการออกแบบที่แนบมากับปริญญานิพนธ์นี้

4.1 โมดูลต่างๆของบอร์ดทดลอง

4.1.1 เอฟพีจีเอโมดูล

เอฟพีจีเอคมดูล ใช้ Chip Altera เบอร์ MAX 7000 ในลักษณะของ In-System Programming กล่าวคือ

- สามารถโปรแกรมชิพได้ ขณะที่มันอยู่ในบอร์ดทดลอง
- ชิพหลาย ๆ ตัวสามารถโปรแกรมได้พร้อมกัน โดยใช้สายดาวน์โหลดเดียวกัน
- รองรับทั้ง EEPROM-Based และ SRAM-Based FPGA

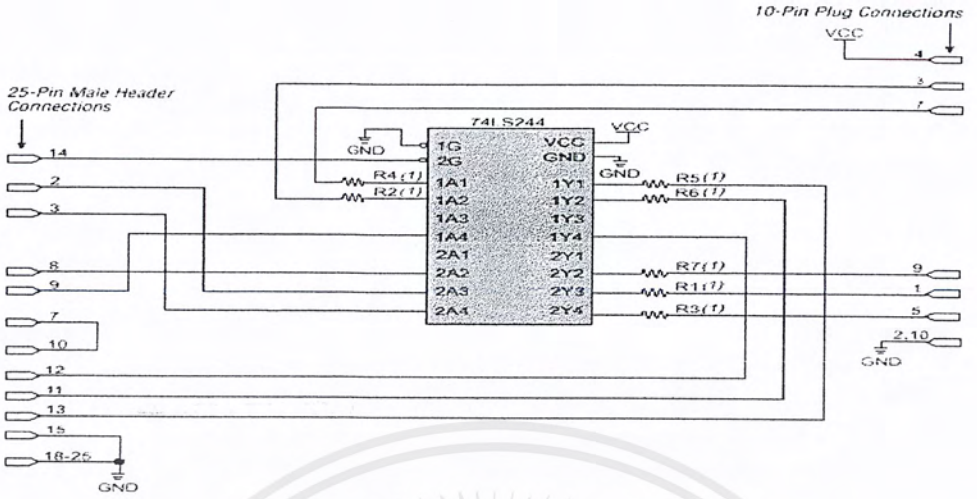


รูปที่ 4.2 แสดงการโปรแกรม FPGA Module ในลักษณะ In System Programming

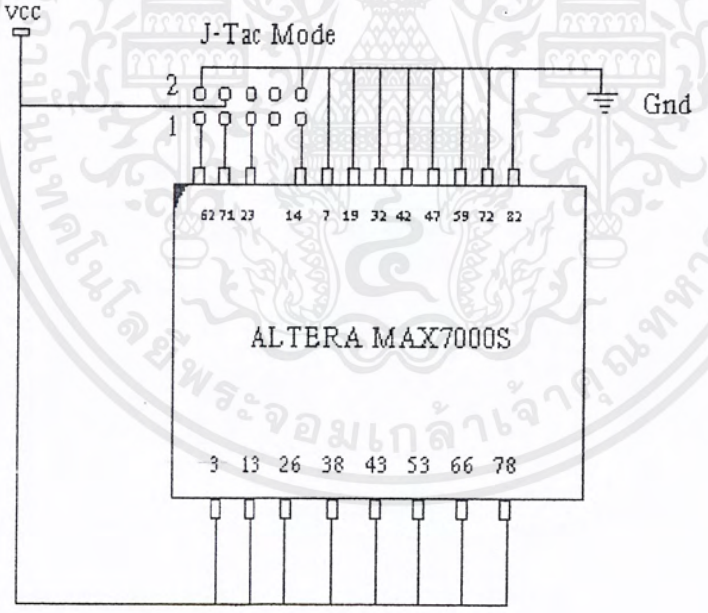
ในการต่อใช้งานลักษณะนี้วิธีการในการทำคือ ต้อง SET เครื่องคอมพิวเตอร์ให้มี Port LPT1 หรือก็คือ Printer port เพื่อที่จะทำการ Down Load ข้อมูลจากคอมพิวเตอร์ มายัง Chip FPGA ซึ่งจำเป็นอย่างยิ่งที่จะต้องต่อ วงจรให้เป็นสองทาง

ทางด้าน Input Jtac ซึ่งได้ออกแบบให้มี หัวต่อ แบบ 10 Pin Connector เพื่อความง่ายในการตรวจสอบ และโดยเฉพาะอย่างยิ่งเพื่อความง่ายในการใช้งานได้สะดวกนั่นเอง

ทางด้าน FPGA จะทำการต่อตรงเข้ามายัง Chip เนื่องจากว่าเป็นข้อกำหนดของทางบริษัท Altera ว่าต้องต่อตามวงจรแบบนี้เท่านั้น ดังนั้นลักษณะการต่อ จึงไม่มีการเปลี่ยนแปลง ดังนั้นความหมายของการต่อแบบนี้จึงเป็นการรองรับ Chip ตัวอื่นๆ ที่มีการ ใช้งานในลักษณะอย่างนี้ได้เหมือนกับว่าเป็นมารตฐานให้เกิดความเข้ากันได้ และความยืดหยุ่นต่อผู้ เขียน Program และผู้พัฒนา เพื่อให้เป็นมาตรฐาน



รูปที่ 4.3 วงจรของสายรับ-ส่ง ข้อมูลเอฟพีจีเอโมดูล

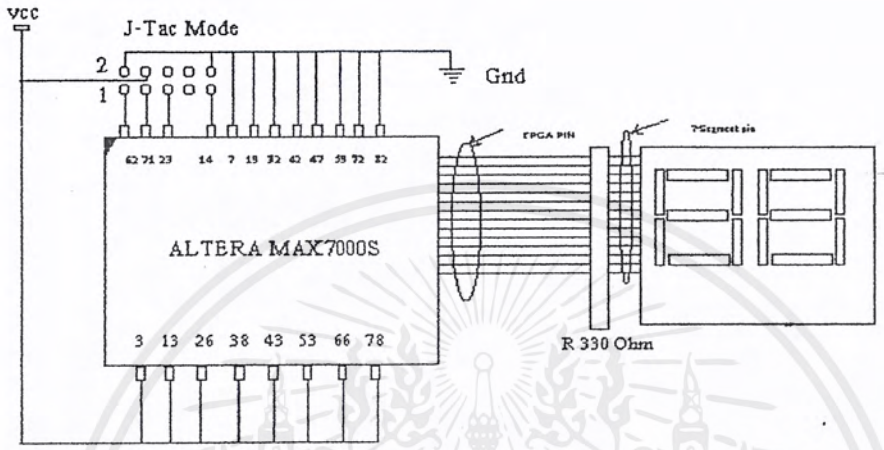


รูปที่ 4.4วงจรของ การจัด Bius ให้กับวงจร FPGA Module

จากรูปที่4.6 นั้นเป็นการป้อนไฟบวก 5 Volt ให้กับขา 3,13,26,,38,43,53,66,78 และทำการป้อน Gnd ให้กับขา 7,19,32,42,47,59,72,82 และต้องทำการต่อ 10 Pin Connector ซึ่งก็คือจุดสำหรับการ นำเอาข้อมูลจากเครื่องคอมพิวเตอร์มาทำการโปรแกรมให้กับ Chip FPGA ALTERA MAX7000S

4.1.2 7-Segment Module

ประ กอบไปด้วย Dual 7-Segment เป็นคู่แฝดติดกันแต่สามารถแยกการใช้งานได้โดยอิสระคือให้นึกเสมือนกับว่ามี LED 7 ตัวต่อ 1 Segment ดังนั้นเวลาใช้งาน Dual 7-Segment จึงใช้งาน 14 ขา และการที่จะให้ 7-Segment แสดงเป็นตัวเลขอะไรออกมาก็ขึ้นอยู่กับการ Control คือการป้อนไฟให้กับ LEDs ต่างๆที่ต่อเป็น ตัวเลข นั่นเอง



รูปที่ 4.5 วงจรของ 7-Segment Module

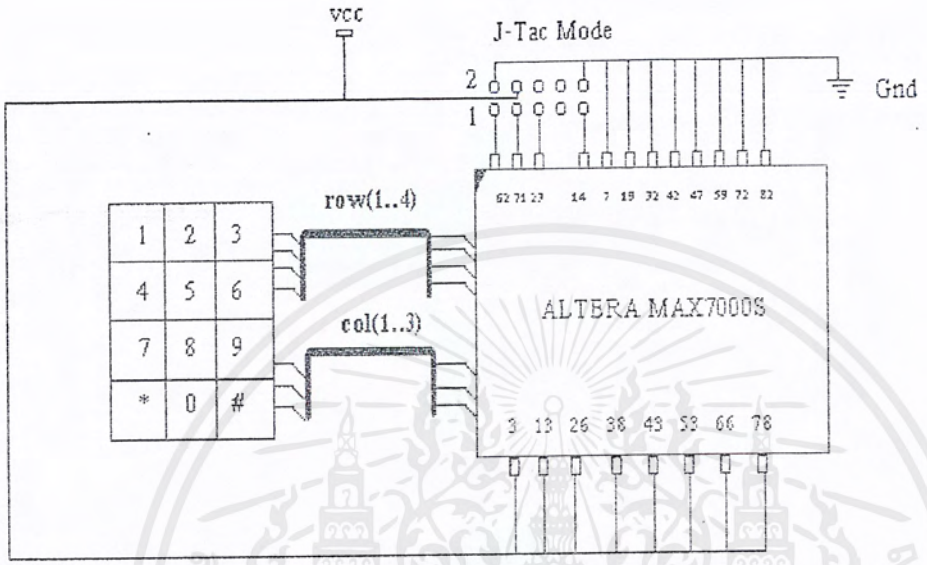
ขาที่ใช้งานกับ FPGA มีดังตารางที่ 5.1 ดังนี้

FPGA	7-Segment
33	A1
34	B1
35	C1
36	D1
37	E1
40	F1
41	G1
44	A2
45	B2
48	C2
49	D2
50	E2
51	F2
52	G2

ตารางที่ 5.1 แสดงขาของ 7-Segment ที่ต่อใช้งานกับ FPGA

4.1.3 Keyboard Module

ในโครงงานนี้ได้เลือกใช้ Keyboard แบบ สามหลัก และ สี่แถว ซึ่งทำให้ได้ขนาด 4X3 คีย์ เมื่อ FPGA รับข้อมูลสามารถนำไปประมวลผล ออกที่ 7-Segment Moule ,Led Module,Stepping motor ได้



รูปที่ 4.6 วงจรของ คีย์บอร์ด โมดูล

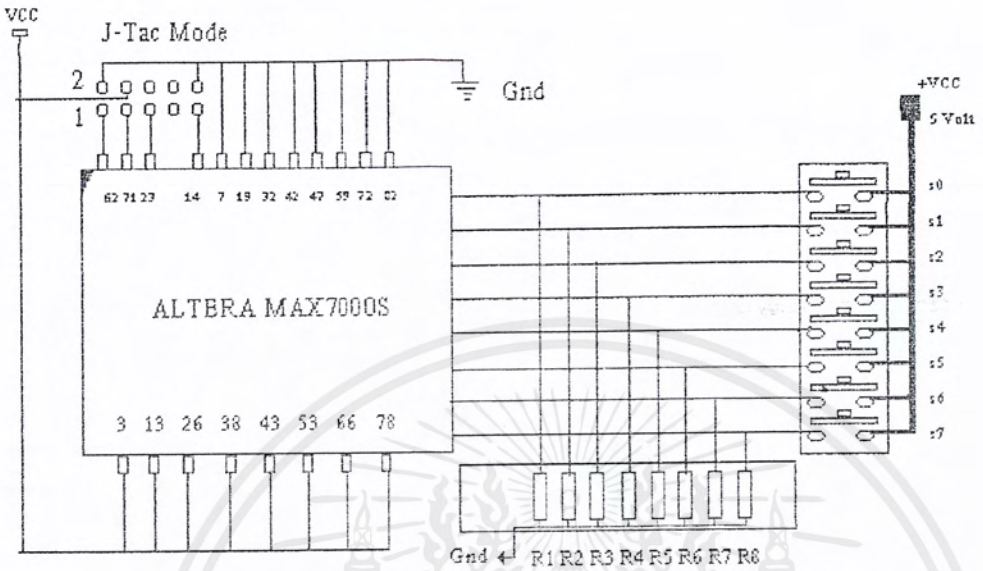
ขาที่ใช้งานกับ FPGA มีดังตารางที่ 4.2 ดังนี้

FPGA	7-Segment
54	C1
55	C2
56	C3
57	R1
58	R2
60	R3
61	R4

ตารางที่ 4.2 แสดงขาของ Keyboard ที่ต่อใช้งานกับ FPGA

4.1.4 Switch Module

ในโครงงานนี้ได้เลือกใช้ Switch แบบ 8 หลัก



รูปที่ 4.7 วงจรของ Switch Module

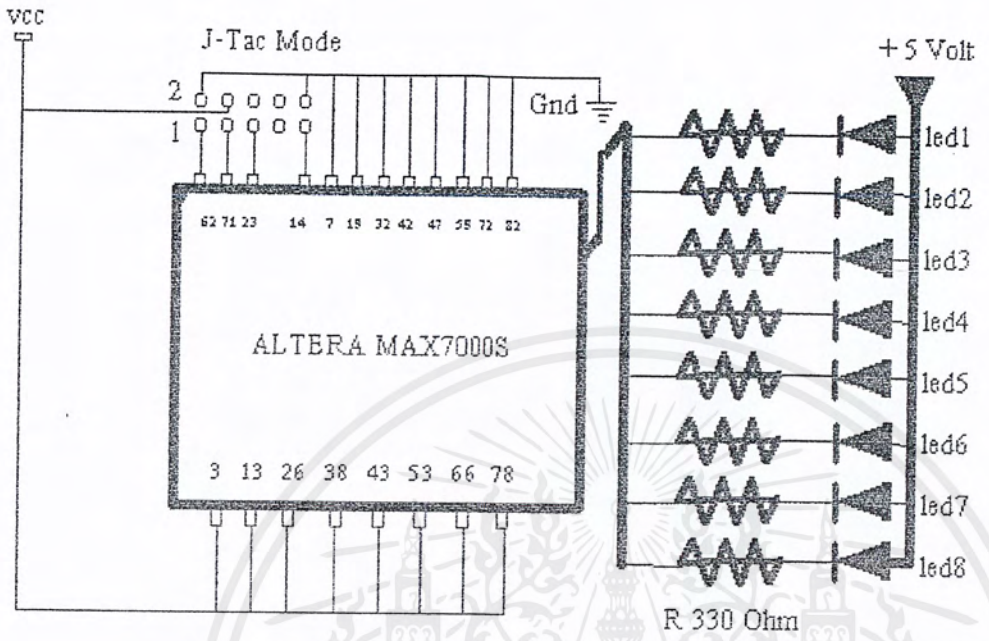
ขาที่ใช้งานกับ FPGA มีดังตารางที่ 4.3 ดังนี้

FPGA	Switch
12	Sw1
15	Sw2
16	Sw3
17	Sw4
18	Sw5
20	Sw6
21	Sw7
22	Sw8

ตารางที่ 4.3 แสดงขาของ Switch ที่ต่อใช้งานกับ FPGA

4.1.5 LED MODULE

ในโครงงานนี้ได้เลือกใช้ LED แบบ 8 หลัก



รูปที่ 4.8 วงจรของ LED Module

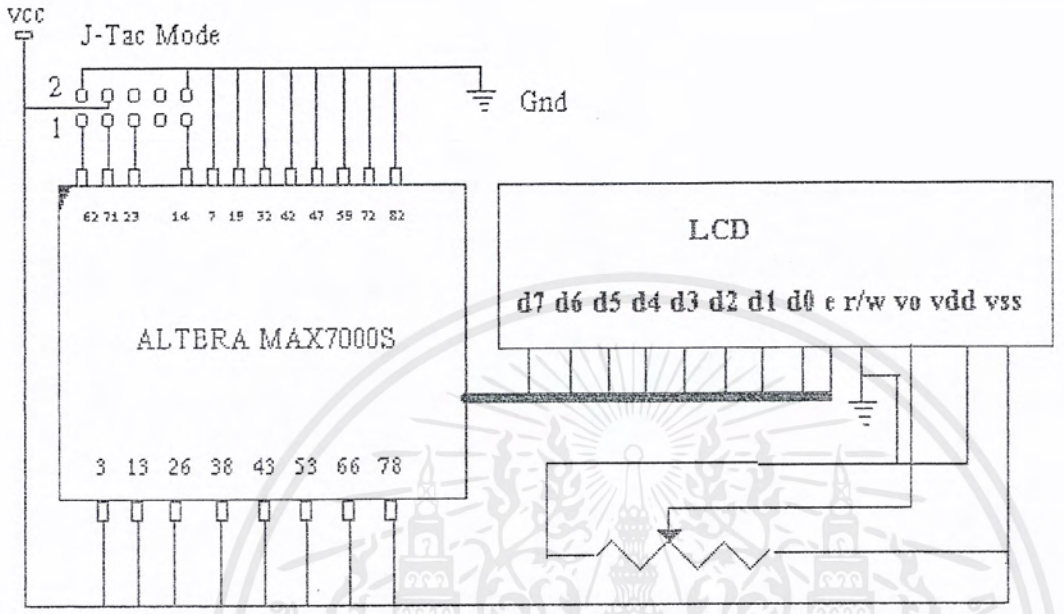
ขาที่ใช้งานกับ FPGA มีดังตารางที่ 4.4 ดังนี้

FPGA	LED
24	LED1
25	LED2
27	LED3
28	LED4
29	LED5
30	LED6
31	LED7
11	LED8

ตารางที่ 4.3 แสดงขาของ Switch ที่ต่อใช้งานกับ FPGA

4.1.6 LCD MODULE

ในโครงงานนี้ได้เลือกใช้ LCD แบบ 16 ตัวอักษร 1 บรรทัด โดยมี ตัวต้านทาน ปรับค่าได้ 250 k โอมห์ เป็นตัวปรับความสว่างของหน้าจอ ผลึกเหลว



รูปที่ 4.9 วงจรของ LED Module

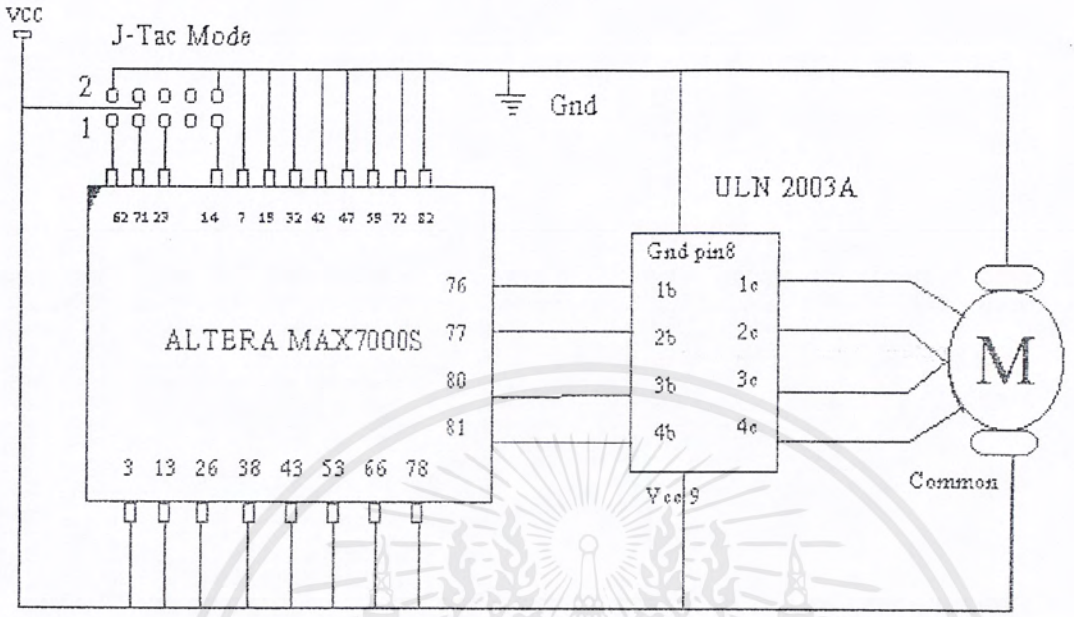
ขาที่ใช้งานกับ FPGA มีดังตารางที่ 4.5 ดังนี้

FPGA	LCD
63	RS
64	E
65	D0
67	D1
68	D2
69	D3
70	D4
73	D5
74	D6
75	D7

ตารางที่ 4.4 แสดงขาของ LCD ที่ต่อใช้งานกับ FPGA

4.1.7 STEPPING MOTOR MODULE

ในโครงงานนี้ได้เลือกใช้ STEPPING MOTOR แบบ 4 เฟส



รูปที่ 4.10 วงจรของ LED Module

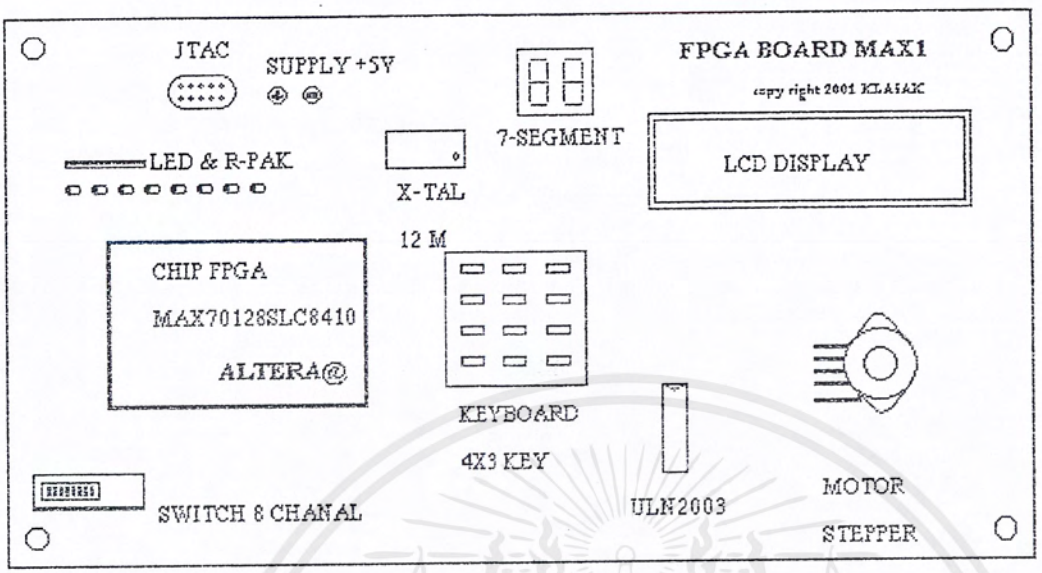
จากรูปที่ 4.10 เป็นวงจรของ STEPPIMG MOTOR ซึ่งต้องใช้ IC ULN2003 มาเป็นตัวขับมอเตอร์

ขาที่ใช้งานกับ FPGA มีดังตารางที่ 4.5 ดังนี้

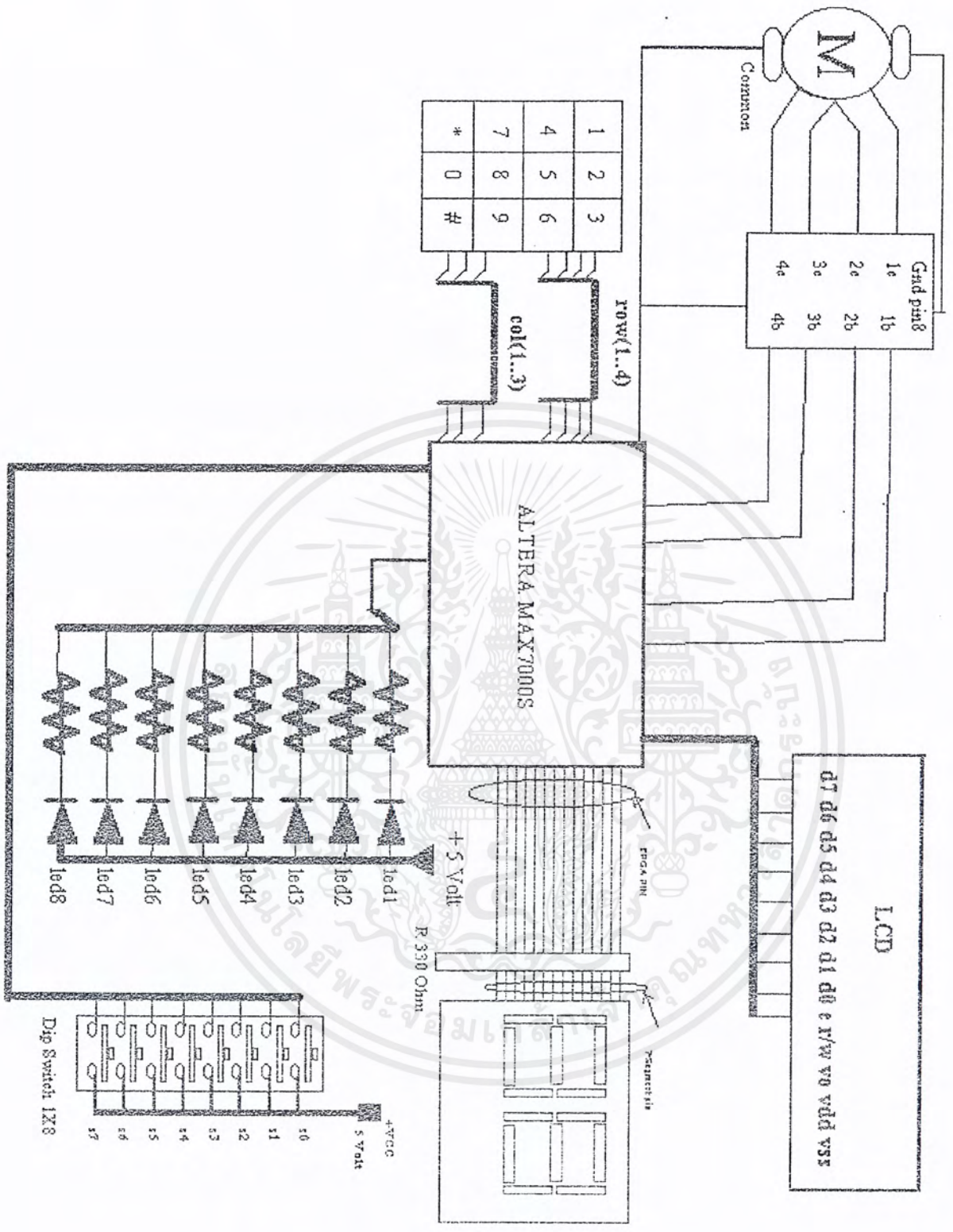
FPGA	STEPPING MOTOR
76	PHASE4
77	PHASE3
80	PHASE2
81	PHASE1

ตารางที่ 4.5 แสดงขาของ STEPPING MOTOR ที่ต่อใช้งานกับ FPGA

4.2 วงจรรวมของบอร์ดทดลอง



รูปที่ 4.11 การจัดวางโมดูลต่างๆ



รูปที่ 4.12 วงจร Schematic รวมของบอร์ดทดลอง

บทที่ 6

การทดลองและผลการทดลอง

6.1 การทดลองที่ 1

โมดูลที่ใช้ในการทดลอง

- FPGA โมดูล

อุปกรณ์ 1.บอร์ดทดลอง FPGA

2.สาย BYTE BLASTER PARALLEL PORT DOWNLOAD CABLE

3.OSCILLOSCOPE

ขั้นตอนการทดลอง

- เขียนโปรแกรมด้วยภาษา AHDL ให้ส่งค่า PLUSE ออกที่ขา 64
- ทดลองให้ออกที่ I/O อื่นๆ

ผลการทดลอง

- มีสัญญาณพัลซ์ที่ขา I/O ที่กำหนด

สรุปผลการทดลอง

- FPGA โมดูลสามารถทำงานได้อย่างถูกต้อง

6.2 หารทดลองที่ 2

โมดูลที่ใช้ในการทดลอง

- FPGA Module
- 7-Segment Module

อุปกรณ์ 1.บอร์ดทดลอง FPGA

2.สาย BYTE BLASTER PARALLEL PORT DOWNLOAD CABLE

ขั้นตอนการทดลอง

- เขียนโปรแกรมด้วยภาษา AHDL ให้สามารถ Control 7-Segment ได้

ผลการทดลอง

- 7-Segment มีการแสดงผลได้ถูกต้องตามที่ได้กำหนดไว้

สรุปผลการทดลอง

- FPGA โมดูลสามารถทำงานแบบ LOOK UP TABLE ได้ในทาง SOFTWARE และ HARDWARE

6.3 หารทดลองที่ 3 โปรแกรมแปลงรหัส BCD เพื่อใช้ในการถอดรหัสและส่งไปยังตำแหน่งต่างๆของ 7-Segment

โมดูลที่ใช้ในการทดลอง

- FPGA Module
- 7-Segment Module
- Dip Switch

อุปกรณ์ 1.บอร์ดทดลอง FPGA

2.สาย BYTE BLASTER PARALLEL PORT DOWNLOAD CABLE

3.Dip Switch

4.7-Segment

ขั้นตอนการทดลอง

- เขียนโปรแกรมด้วยภาษา AHDL ให้สามารถ Control โดยใช้ DipSwitch เพื่อแสดงผลออกทาง 7-Segment ได้

ผลการทดลอง

- 7-Segment มีการแสดงผลได้ถูกต้องตามที่ได้โปรแกรมกำหนดจาก Dip Switch กำหนดไว้

สรุปผลการทดลอง

- FPGA โมดูลสามารถทำงานโดยมีอุปกรณ์ Input / Output ร่วมกันได้

6.4 การทดลองที่ 4 KEYPAD TO 7-SEGMENT

โมดูลที่ใช้ในการทดลอง

- FPGA โมดูล

อุปกรณ์ 1.บอร์ดทดลอง FPGA

2.สาย BYTE BLASTER PARALLEL PORT DOWNLOAD CABLE

3.KEYPAD

4.7-SEGMENT

ขั้นตอนการทดลอง

- เขียนโปรแกรมด้วยภาษา AHDL ให้รับค่า Input จาก Keyboard โดยใช้หลักการ Scan Key
- ให้ส่งข้อมูลไปออกที่ 7-Segment

ผลการทดลอง

- เมื่อมีการกด Key ที่ Keyboard จะทำให้ตัวเลข 7-Segment แสดงผลได้ตามกำหนด

สรุปผลการทดลอง

- FPGA โมดูลสามารถทำงานได้อย่างถูกต้องโดยสามารถออกแบบโปรแกรมที่ควบคุมการทำงานของ Keypad และส่งไปที่ 7-Segment เพื่อแสดงผลตัวเลขที่กด

6.5 การทดลองที่ 5 วงจรนับขึ้นลงและแสดงผลไปยัง 7-Segment

โมดูลที่ใช้ในการทดลอง

- FPGA Module
- 7-Segment Module
- Dip Switch

อุปกรณ์ 1.บอร์ดทดลอง FPGA

2.สาย BYTE BLASTER PARALLEL PORT DOWNLOAD CABLE

3.Dip Switch

4.7-Segment

ขั้นตอนการทดลอง

- เขียนโปรแกรมโดย ให้สามารถ แสดง การนับขึ้นลง และแสดงผลไปยัง 7-Segment โดยเขียนโปรแกรมนับตัวเลขโดยใช้ DipSwitch เพื่อแสดงผลออกทาง 7-Segment ได้

ผลการทดลอง

- 7-Segment มีการแสดงผลได้ถูกต้องตามที่ได้โปรแกรมกำหนดจาก Dip Switch กำหนดไว้

สรุปผลการทดลอง

- FPGA โมดูลสามารถทำงานให้นับขึ้นและนับลงแล้วนำครั้งการนับไปแสดงที่จอ แสดงผลเลข แบบ 7 ส่วนได้ และเป็นการควบคุมด้วย Switch

6.6 การทดลองที่ 6 STEPPING MOTOR CONTROL

โมดูลที่ใช้ในการทดลอง

- FPGA โมดูล
- STEPPING MOTOR MODULE
- SWITCH MODULE

อุปกรณ์ 1.บอร์ดทดลอง FPGA

2.สาย BYTE BLASTER PARALLEL PORT DOWNLOAD CABLE

3.STEPPING MOTOR

4.SWITCH 8 BIT

ขั้นตอนการทดลอง

- เขียนโปรแกรมด้วยภาษา AHDL ให้รับค่า INPUT จากทาง Dip Switch แล้วสั่งงานให้ Stepping Motor หมุน

ผลการทดลอง

- เมื่อมีการกด Key ที่ Dip Switch จะทำให้ตัวเลข Stepping Motor หมุนได้ตามผู้สั่งให้ หมุนแบบไหนก็ได้ตามข้อกำหนดของ Program

สรุปผลการทดลอง

- FPGA โมดูลสามารถทำงานได้อย่างถูกต้องโดยสามารถออกแบบโปรแกรมที่ควบคุมการทำงานของ Input Switch และส่งไปที่ 4-Phase ของ Stepping Motor

6.7 หารทดลองที่ 7 วงจรควบคุม LCD MODULE

โมดูลที่ใช้ในการทดลอง

- FPGA Module
- LCD MODULE
- Dip Switch

อุปกรณ์ 1.บอร์ดทดลอง FPGA

2.สาย BYTE BLASTER PARALLEL PORT DOWNLOAD CABLE

3.Dip Switch

4.LCD

ขั้นตอนการทดลอง

- เขียนโปรแกรมโดย ให้สามารถ แสดง การควบคุม LCD คือให้มีการ ใช้คำสั่งแบบ STATE MACHINE ให้มีการ clear จอภาพ ใช้คำสั่งให้มีการกระพริบของจอภาพ

ผลการทดลอง

- เมื่อใช้คำสั่งแบบ STATE MACHINE มีการแสดงผลได้ถูกต้องตามที่ได้โปรแกรมกำหนดจาก Dip Switch กำหนดไว้

สรุปผลการทดลอง

- FPGA โมดูลสามารถทำงานให้ Output แบบ State Machine คือมีขั้นตอนการคิดแบบ ค่า Output ที่จะได้รับมีผลมาจากค่าในอดีตด้วย

บทที่ 7

สรุปผลและบทวิจารณ์

เมื่อได้ศึกษาการออกแบบ บอร์ดทดลองได้รับรู้ถึงประสิทธภาพของ Chip ทางด้าน ดิจิตอลเป็นอย่างสูง เนื่องจากว่ามีความยืดยื่นในการใช้งาน เมื่อผู้ใช้มี Algorithm ใดๆ ก็สามารถที่จะคิด แบบ digital แล้วแปลงขบวนการคิดนั้นไปสู่ ภาคปฏิบัติทาง HardWare

7.1 ปัญหาและอุปสรรคในการดำเนินงาน

อย่างไรก็ตามไม่มีทางเป็นไปได้ที่จะสามารถเชี่ยวชาญหากไม่ได้ลงมือปกิบัติ เพราะถึงแม้ว่าจะมีทฤษี มากขนาดไหนอาจจะไม่สำเร็จในทางปฏิบัติได้ ปัญหาที่เจอบ่อยๆ ในการทำดครงงานนี้ก็คือ Chip FPGA ความจุยังน้อยอยู่ วิธีแก้คือ เลือกใช้ Chip FPGA ที่มีความจุสูงๆ เป็นผลทำให้มีราคาแพงตามมา

7.2 คำวิจารณ์

เนื่องจากภาษา AHDL นั้นยังไม่มีผู้นิยมมากนักในประเทศไทย มีแต่ผู้ สนใจภาษา VHDL จึงให้ไม่ค่อยมีข้อมูลมากนักต้องหาทาง Internet หรือบริษัทตัวแทนจำหน่ายโดยตรง

7.3 สิ่งที่ต้องพัฒนาต่อไป

- มีความจำเป็น เป็นอย่างยิ่งที่จะมี Chip ที่มีความจุสูงๆ เพราะบอร์ ด FPGA จะสามารถ ดูณสัญญาณได้ สามารถทำสิ่งอื่นๆได้อีกเช่น วงจรทางล้านอิเล็กทรอนิกส์เป็นต้น

ภาคผนวก

โปรแกรม ส่งค่าแสดงผล LEDs แบบ StateMachine (Mealy Model)

```
SUBDESIGN patgen
(       clk, ena : INPUT;
        out[8..1] : OUTPUT; )
VARIABLE
   ss : MACHINE WITH STATES (s0, s1, s2, s3,
s4,s5,s6,s7,s8,s9,s10,s11,s12,s13,s14,s15,s16);

BEGIN
        ss.clk = clk;
        TABLE
                ss,     ena     =>      ss,     out[ ];
                s0,     0       =>      s1,     H"0";
                s0,     1       =>      s1,     H"0";
                s1,     x       =>      s2,     b"10000001";
                s2,     x       =>      s3,     b"01000010";
                s3,     x       =>      s4,     b"00100100";
                s4,     x       =>      s5,     b"00011000";
                s5,     x       =>      s6,     b"01111110";
                s6,     x       =>      s7,     b"10111101";
                s7,     x       =>      s8,     b"11011011";
                s8,     x       =>      s9,     b"11100111";
                s9,     x       =>      s10,    b"00000000";
                s10,    x       =>      s11,    b"11111111";
                s11,    x       =>      s12,    b"00001111";
                s12,    x       =>      s13,    b"11110000";
                s13,    x       =>      s14,    b"11001100";
                s14,    x       =>      s15,    b"00110011";
                s15,    x       =>      s16,    b"10101010";
                s16,    x       =>      s1,     b"01010101";
        END TABLE;
END;
```

โปรแกรม Decode 7-Segment

```
SUBDESIGN dec7seg0
(                   BCD[4..1]:INPUT;
                    a,b,c,d,e,f,g:OUTPUT;
)
BEGIN
DEFAULTS
                    (a,b,c,d,e,f,g)=b"0000000";
END DEFAULTS;
TABLE
BCD[]               => a,  b,  c,  d,  e,  f,  g;
0                           => b"1",b"1",b"1",b"1",b"1",b"1",b"0";
1                           => b"0",b"1",b"1",b"0",b"0",b"0",b"0";
2                           => b"1",b"1",b"0",b"1",b"1",b"0",b"1";
3                           => b"1",b"1",b"1",b"1",b"0",b"0",b"1";
4                           => b"0",b"1",b"1",b"0",b"0",b"1",b"1";
5                           => b"1",b"0",b"1",b"1",b"0",b"1",b"1";
6                           => b"1",b"0",b"1",b"1",b"1",b"1",b"1";
7                           => b"1",b"1",b"1",b"0",b"0",b"0",b"0";
8                           => b"1",b"1",b"1",b"1",b"1",b"1",b"1";
9                           => b"1",b"1",b"1",b"1",b"0",b"1",b"1";
h"a"      => b"1",b"1",b"1",b"0",b"1",b"1",b"1";
h"b"      => b"0",b"0",b"1",b"1",b"1",b"1",b"1";
h"c"      => b"1",b"0",b"0",b"1",b"1",b"1",b"0";
h"d"      => b"0",b"1",b"1",b"1",b"1",b"0",b"1";
h"e"      => b"1",b"0",b"0",b"1",b"1",b"1",b"1";
h"f"     => b"1",b"0",b"0",b"0",b"1",b"1",b"1";
END TABLE;
END;
```

โปรแกรม หารความถี่

```
subdesign 'div'
(
  clk:input;
  q[15..0]:output;
)
variable
  count[15..0]:dff;
begin
  count[].clk=clk;
if count[].q==23767 then
  count[].d=0;
else
count[].d=count[].q+1;
end if;
q[]=count[].q;
end;
```

โปรแกรมสแกน Keyboard ทางด้านแนวนอน

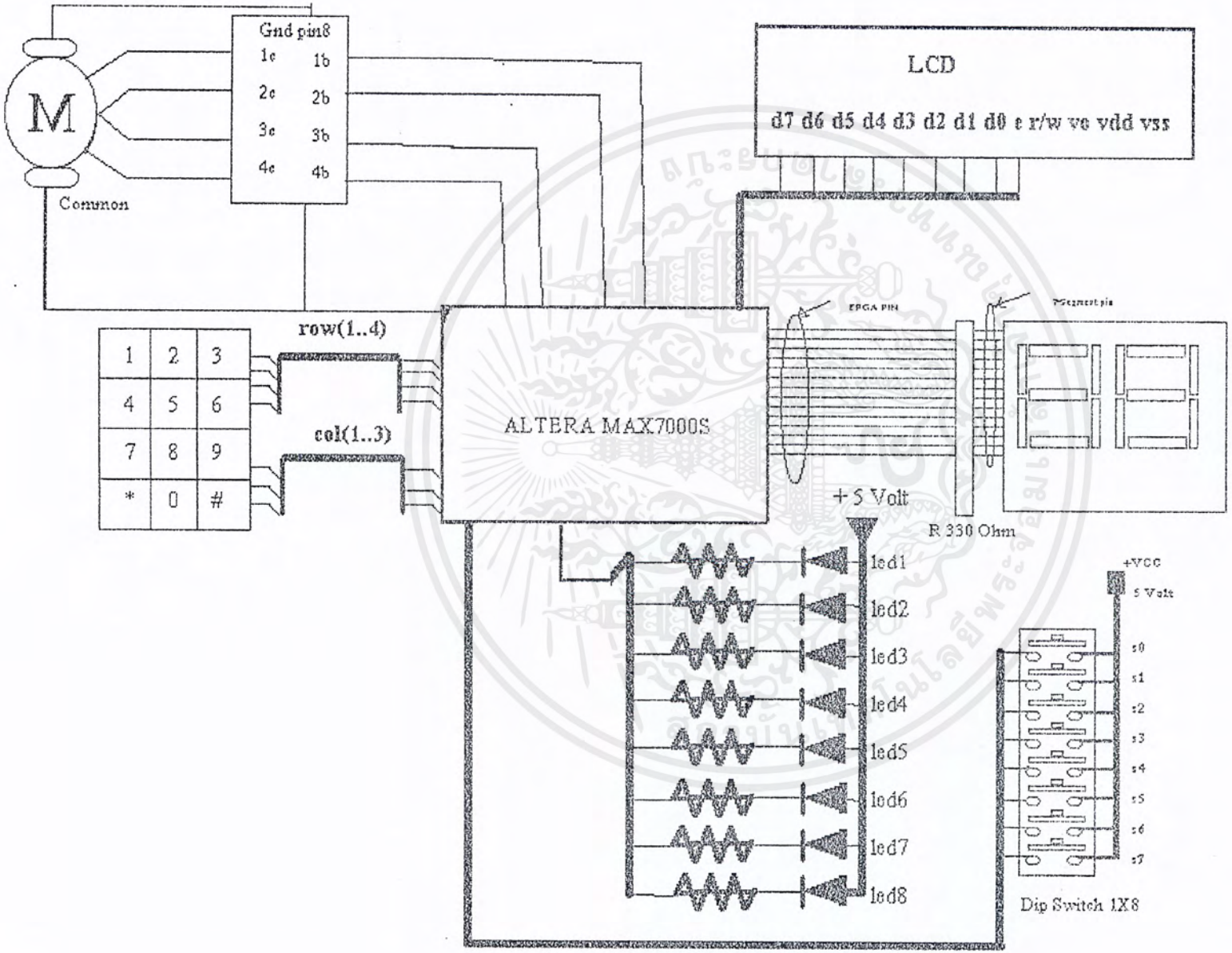
```
Title "scan row";
subdesign scc_row
(
  clock40:input;
  row[4..1]  :output;
)
variable
  row[4..1]  :dff;
begin
  row[].clk=clock40;
row[]=(row[3..1],row[3..1]==0);
end;
```

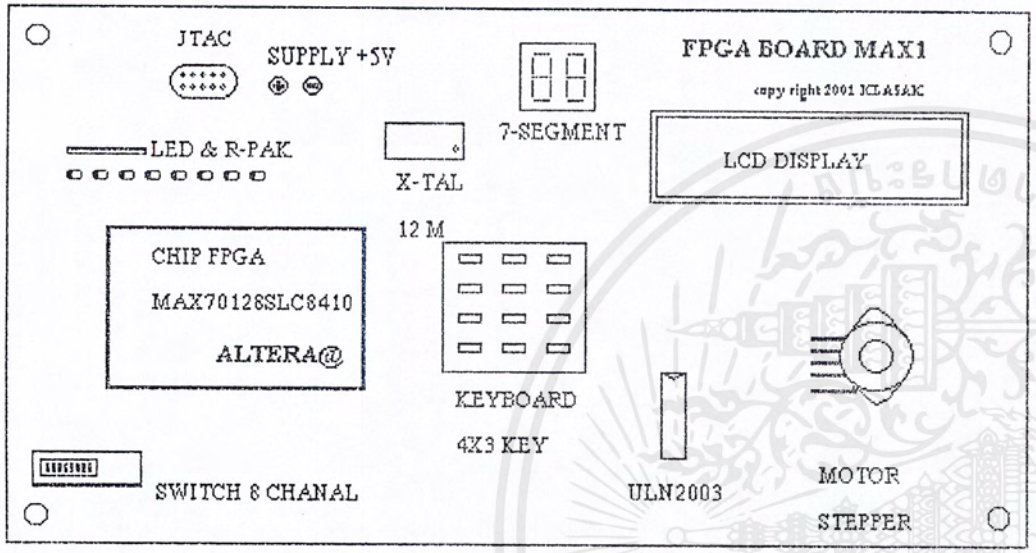
โปรแกรม SCAN KEYBORD โดยการใช้ TABLE

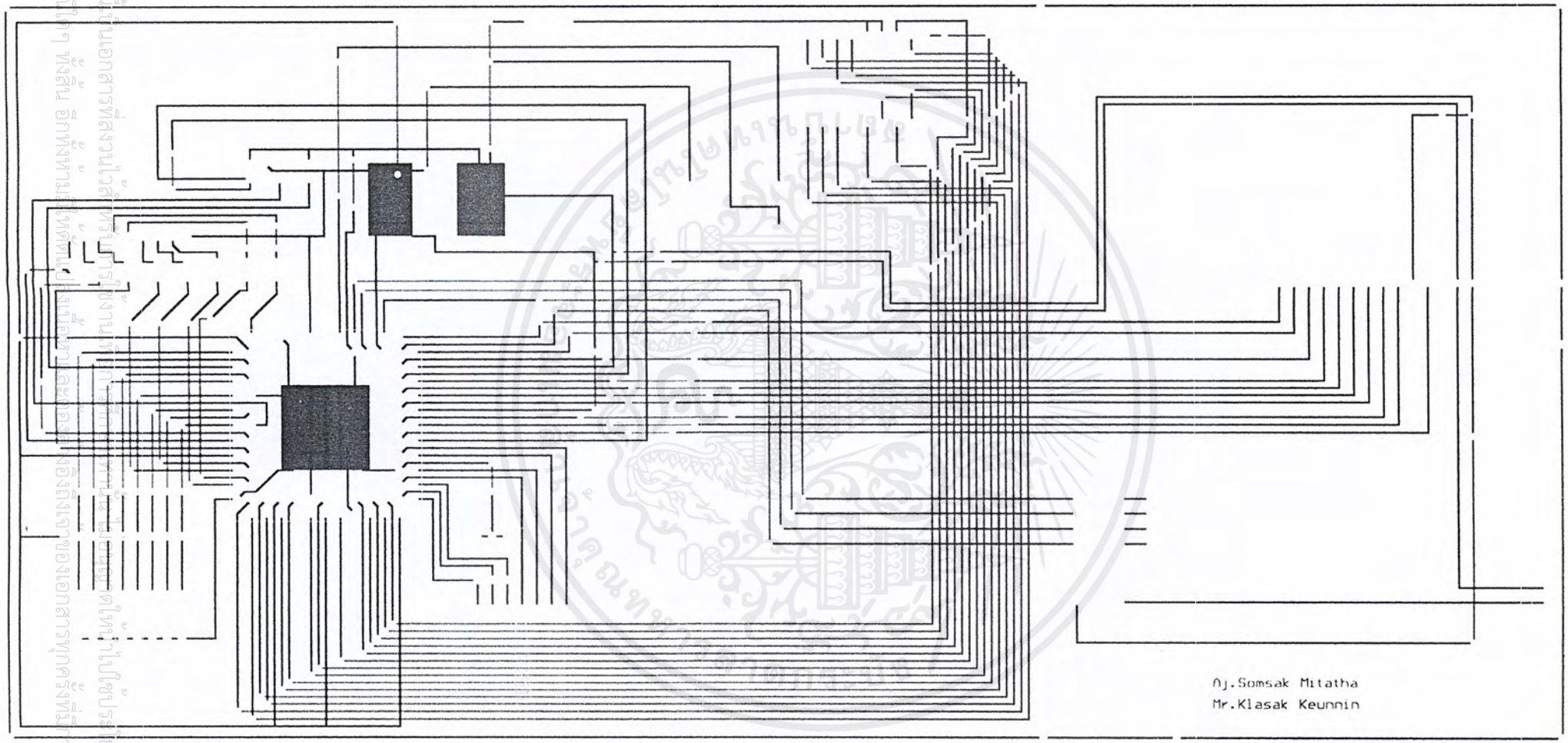
```
subdesign scn  %scan column%
(
  row[4..1],col[3..1],clock40   :input;
  number[3..0],en               :output;
)
variable
   en:node;
number[3..0]:dffe;
begin
defaults
   number[]=gnd;
end defaults;

   en = col1 # col2 #col3;
   number[].clk = clock40;
   number[].ena = en;
    table
    row[],col[] => number[];
    b"0001",b"001"  => h"1";
    b"0001",b"010"  => h"2";
    b"0001",b"100"  => h"3";
    b"0010",b"001"  => h"4";
    b"0010",b"010"  => h"5";
    b"0010",b"100"  => h"6";
    b"0100",b"001"  => h"7";
    b"0100",b"010"  => h"8";
    b"0100",b"100"  => h"9";
    b"1000",b"001"  => h"a";
    b"1000",b"010"  => h"0";
    b"1000",b"100"  => h"c";
end table;

end;
```

โปรแกรม Stepping Motor

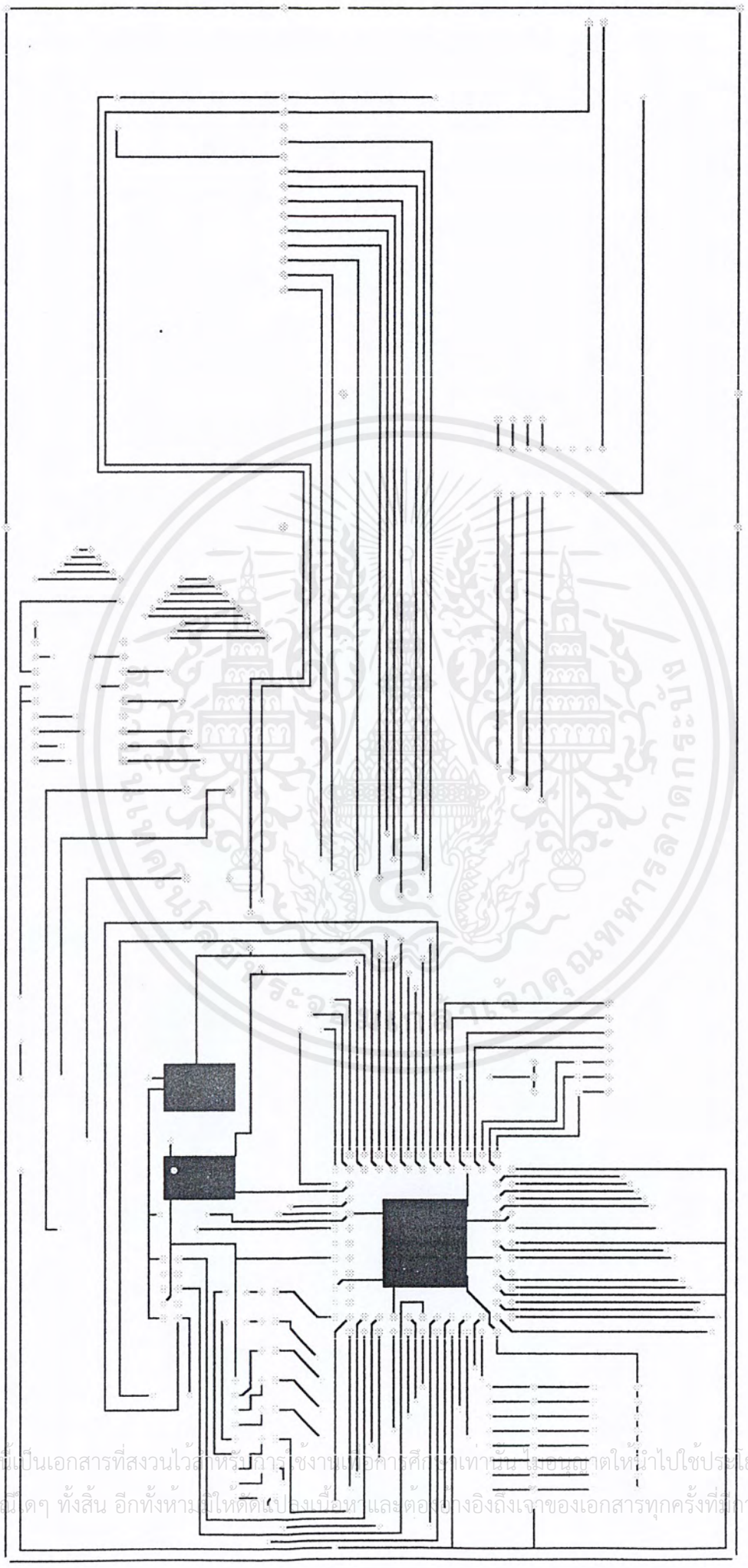
```
SUBDESIGN steppingmotor
(       clk, ena : INPUT;
        out[3..0] : OUTPUT; )
VARIABLE
   ss : MACHINE WITH STATES (s0, s1, s2, s3, s4);

BEGIN
        ss.clk = clk;
        TABLE
                ss,     ena     =>      ss,     out[ ];
                s0,     0       =>      s1,     H"0";
                s0,     1       =>      s1,     H"0";
                s1,     x       =>      s2,     H"8";
                s2,     x       =>      s3,     H"4";
                s3,     x       =>      s4,     H"2";
                s4,     x       =>      s1,     H"1";
        END TABLE;
END;
```

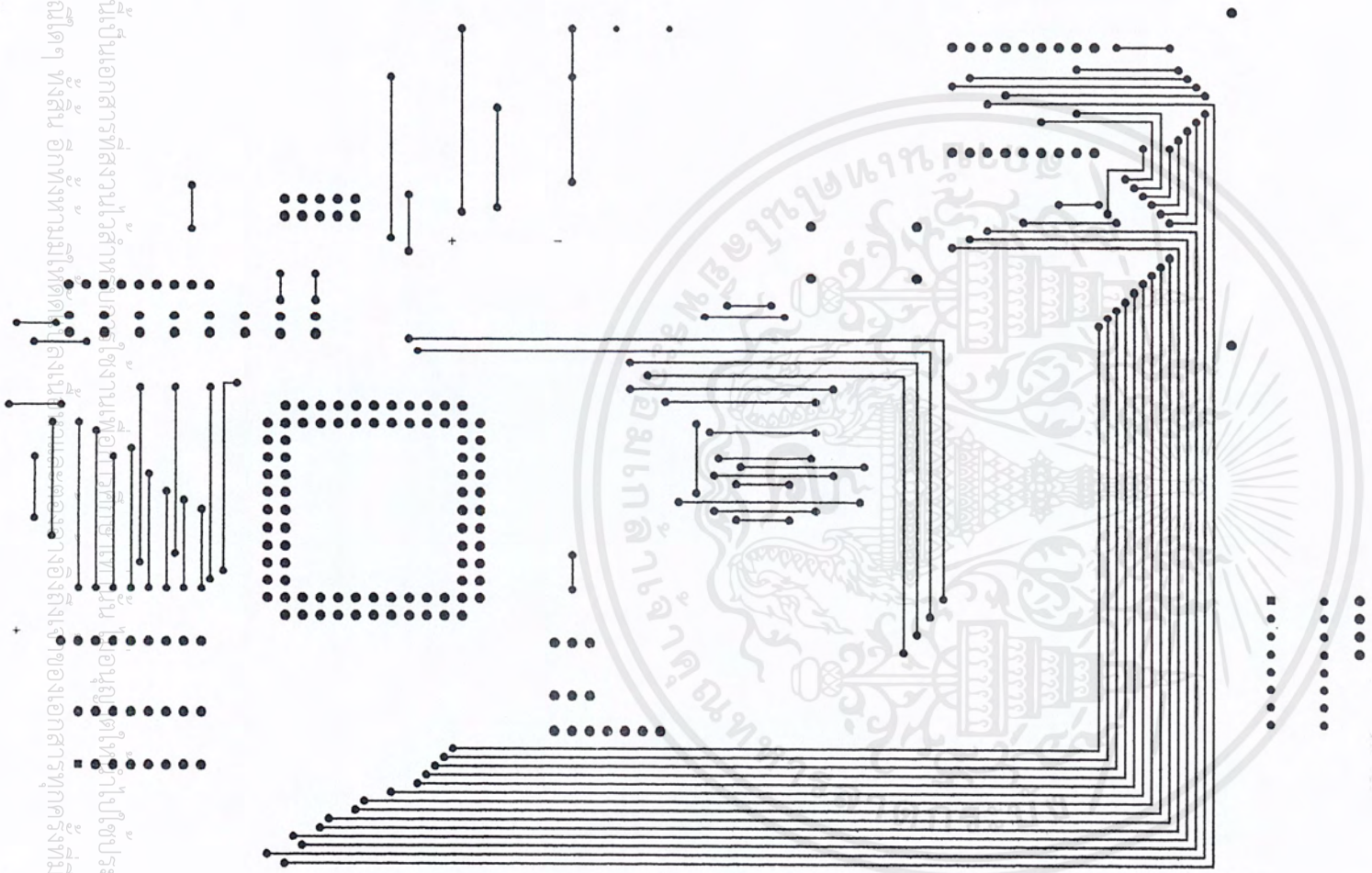





Aj.Somsak Mitatha
Mr.Klasak Keunnin

Aj.Somsak Mitatha
Mr.Klasak Keunnin





Aj.Somsak Mitatha
Mr.Klasak Keunnin

SN74LS240, SN74LS244

Octal Buffer/Line Driver with 3-State Outputs

The SN74LS240 and SN74LS244 are Octal Buffers and Line Drivers designed to be employed as memory address drivers, clock drivers and bus-oriented transmitters/receivers which provide improved PC board density.

- Hysteresis at Inputs to Improve Noise Margins
- 3-State Outputs Drive Bus Lines or Buffer Memory Address Registers
- Input Clamp Diodes Limit High-Speed Termination Effects

ON Semiconductor

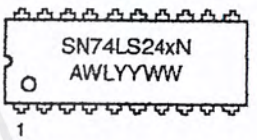
http://onsemi.com

LOW POWER SCHOTTKY

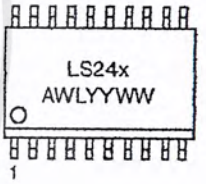
GUARANTEED OPERATING RANGES

Symbol	Parameter	Min	Typ	Max	Unit
V_{CC}	Supply Voltage	4.75	5.0	5.25	V
T_A	Operating Ambient Temperature Range	0	25	70	°C
I_{OH}	Output Current – High			–3.0	mA
				–15	mA
I_{OL}	Output Current – Low			24	mA

MARKING DIAGRAMS



PDIP–20
N SUFFIX
CASE 738



SOIC–20
DW SUFFIX
CASE 751D

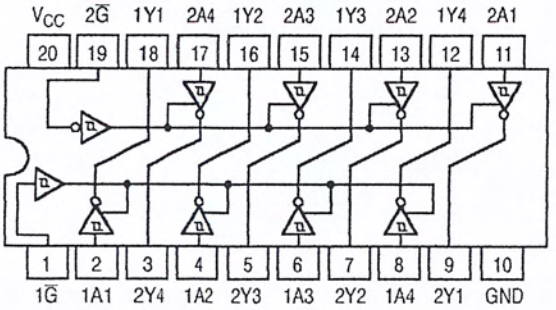
x = 0 or 4
A = Assembly Location
WL = Wafer Lot
YY = Year
WW = Work Week

ORDERING INFORMATION

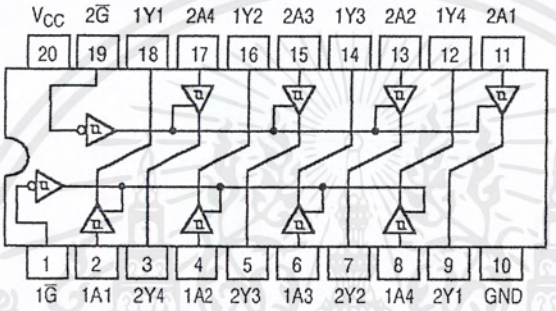
Device	Package	Shipping
SN74LS240N	PDIP–20	1440 Units/Box
SN74LS240DW	SOIC–20	2500/Tape & Reel
SN74LS244N	PDIP–20	1440 Units/Box
SN74LS244DW	SOIC–20	2500/Tape & Reel

LOGIC AND CONNECTION DIAGRAMS DIP (TOP VIEW)

SN74LS240



SN74LS244



TRUTH TABLES

SN74LS240

INPUTS		OUTPUT
1G, 2G	D	
L	L	H
L	H	L
H	X	(Z)

SN74LS244

INPUTS		OUTPUT
1G, 2G	D	
L	L	L
L	H	H
H	X	(Z)

H = HIGH Voltage Level
L = LOW Voltage Level
X = Immaterial
Z = HIGH Impedance

เอกสารอ้างอิง

[1] พรชัย ภววงศ์ศักดิ์,"เอกสารประกอบการอบรมพิเศษ FPGAs", ภาควิชาวิศวกรรมอิเล็กทรอนิกส์ มหาวิทยาลัยเทคโนโลยีมหานคร

[2] ธีรยศ เวียงทอง,"เอกสารประกอบการเรียนการสอน การออกแบบระบบดิจิตอลแนวใหม่", ภาควิชาวิศวกรรมอิเล็กทรอนิกส์ มหาวิทยาลัยเทคโนโลยีมหานคร

[3] Frank A. Scarpino,"VHDL and AHDL Digital System Implementation ,Pretice-Hall,1998