การออกแบบเครื่องทดสอบสัญญาณโทรทัศน์โดยใช้วงจรรวมแอฟพีจีเอ

VIDEO PATTERN GENERATOR BY FPGA CIRCUIT

โดย

นาย เรวัติ รัตนโชติ

นาย วิโรจน์ กิ่งเพชรมณี

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตร์บัณฑิต

สาขาวิชาเทคโนโลยีโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

หัวข้อปริญญานิพนธ์	การออกแบบเครื่องทดสอบสัญญาณโทรทัศน์โดยใช้วงจรเอฟพีจีเอ Video Pattern Generator By FPGA Circuit
โดย	นาย เรวัติ รัตนโชติ นาย วิโรจน์ กิ่งเพชรมณี
อาจารย์ที่ปรึกษา	อาจารย์ กฤดากร กล่อมการ
ภาควิชา	เทคนิคอุตสาหกรรม
สาขาวิชา	เทคโนโลยีโทรคมนาคม
ปีการศึกษา	2543

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
อนุมัติให้นับปริญญานิพนธ์ฉบับนี้ เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรอุตสาหกรรมศาสตร์บัณฑิต

คณะกรรมการสอบปริญญานิพนธ์

---------- ประธานกรรมการ
(.....)

----------กรรมการ
(.....)

----------กรรมการ
(.....)

----------กรรมการ
(.....)

----------กรรมการ
(.....)

ลิขสิทธิ์ของคณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

หัวข้อปริญญานิพนธ์	การออกแบบเครื่องทดสอบสัญญาณโทรทัศน์โดยใช้วงจรเอฟพีจีเอ Video Pattern Generator By FPGA Circuit
ชื่อนักศึกษา	นาย เรวัติ รัตนโชติ นาย วิโรจน์ กิ่งเพชรมณี
อาจารย์ที่ปรึกษา	อาจารย์ กฤดากร กล่อมการ
ภาควิชา	เทคนิคอุตสาหกรรม
สาขาวิชา	เทคโนโลยีโทรคมนาคม
ปีการศึกษา	2543

บทคัดย่อ

โครงงานนี้เป็นการสร้างเครื่องกำเนิดสัญญาณภาพทดสอบเครื่องรับโทรทัศน์ ในรูปแบบต่างๆซึ่งสัญญาณที่ส่งออกมาเป็นได้ทั้งสัญญาณวีดีโอและสัญญาณภาพที่ได้สามารถเป็นได้ทั้งภาพสีและขาวดำ ในการออกแบบจะใช้หลักการของ Top –Down Design โดยการแบ่งตัววิดิโอแพทเทอร์เจนเนเรเตอร์ ออกมาเป็นส่วน จากนั้นจึงเขียนแต่ละส่วนด้วยภาษาวีเอชดีแอลและตรวจสอบการทำงานแต่ละส่วนด้วยการใช้ซอฟท์แวร์ซิมูเลชั่น(Simualation) จนแน่ใจว่าแต่ละส่วนที่เขียนขึ้นมีการทำงานที่ถูกต้อง จึงนำแต่ละส่วนที่เขียนขึ้นนี้มารวมเข้าด้วยกันและทำการทดสอบการทำงานจนแน่ใจว่าแต่ละส่วนที่สร้างขึ้นมาแต่ละส่วนสามารถทำงานร่วมกันได้อย่างไม่มีปัญหา จากนั้นจึงนำ Soft code ของวิดิโอแพทเทอร์เจนเนเรเตอร์ได้ เข้าสู่กระบวนการสังเคราะห์(Synthesis) เพื่อแปลงเป็นวีเอชดีแอลโค๊ด(Vhdl Code) ให้เป็นวงจรระดับเกต (Gte-level) และนำวงจรที่ได้จากการสังเคราะห์ไปบันทึกลงในเอฟพีจีเอเพื่อทดสอบการทำงานกับอุปกรณ์ทางฮาร์ดแวร์จริง

Title	Video Pattern Generator By FPGA Circuit
Studte	Mr. REWAT RATTANACHOT
	Mr. WTROT KINGPETMANEE
Advisor	Mr. KITDAKORN KLOMKA
Degree	Bachelor Degree of Industrial Technology
Programme	Telecommunication Technology
Department	Industrial Technology
Academic	2000

ABSTRACT

This project is to make television pattern generator in many shapes. It's signal can be sent in the forms of VDO signal. It can produce both color and white-black picture. In the design process,the researchers use top-down design model to develop the TV pattern generator. The process of design begins writing each part in VHDL language and each part with simulation software to ensure that these parts are working correctly,then combine them together and test the corrective of the cooperation between these part.Then bring the derived soft-core into the synthesis process to convert the VHDL code into gate-level netlist and bring this netlist into the place-and-route process to program this TV pattern generator into the FPGA. When deriving the TV pattern generator in FPGA, bring this FPGA into the circuit board to test the corrective operation of it.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ สำเร็จลุล่วงได้เนื่องจากการสนับสนุนและกำลังใจจากบิดามารดา และด้วยความร่วมมือและความตั้งใจในการทำงานของเพื่อนร่วมงานในกลุ่มและได้รับคำแนะนำจาก อาจารย์ที่ปรึกษา อาจารย์ กฤดากร กล่อมการ ทางด้านเทคนิคและแนวความคิดต่างๆ นอกจากนี้ยังได้รับคำแนะนำในการออกแบบจาก คุณ อุดมพร สุนันทชัยกุล และ คุณ วัชรากร หนูทอง คณะ ผู้จัดทำใคร่ขอขอบคุณมา ณ โอกาสนี้

นาย เรวัติ รัตนโชติ
นาย วิโรจน์ กิ่งเพชรมณี

สารบัญ

สารบัญ(ต่อ)

หน้า

สารบัญตาราง

สารบัญภาพ

สารบัญภาพ (ต่อ)

หน้า

สารบัญภาพ (ต่อ)

หน้า

บทที่ 1
บทนำ

ในอดีตนักออกแบบวงจรดิจิตอลได้ใช้วิธีการออกแบบวงจรด้วยวิธี Bottom-up design โดยทำกันที่ระดับ Gate-level ซึ่งสร้างความยุ่งยากเป็นอย่างมากให้แก่ผู้ออกแบบ ตั้งแต่ปี 90 เป็นต้นมาการออกแบบก็เริ่มทวีความซับซ้อนมากขึ้น ผู้ออกแบบจะต้องออกแบบไอซีที่มีจำนวนเกทมากกว่า 100,000 เกทภายในระยะเวลาอันจำกัดเนื่องจากแรงดันในการที่จะต้องนำผลิตภัณฑ์ออกสู่ท้องตลาดให้ได้ภายในระยะเวลาอันสั้น วิธีการออกแบบที่เคยใช้กันมาเริ่มไม่ใช่ทางออกที่เหมาะสมอีกต่อไป เนื่องจากวิธีการดังกล่าวนั้นยุ่งยาก,เสียเวลา และมักเกิดข้อผิดพลาดขึ้นบ่อย ๆ ปัจจุบัน เทคนิคการออกแบบวงจรก้าวหน้าไปมาก ผู้ออกแบบไม่จำเป็นจะต้องออกแบบในระดับ Gate-level เหมือนดังแต่ก่อน วิธีการออกแบบที่นิยมคือ การเขียนวงจรที่ต้องการด้วยภาษาบรรยายการทำงานของวงจร (Hardware Description Language : HDL) ซึ่งสามารถตรวจสอบความถูกต้องได้ด้วยการใช้ซอฟท์แวร์ Simulation จนแน่ใจว่างานที่ออกแบบนั้นมีการทำงานที่ถูกต้อง จึงนำงานนั้นไปเข้าสู่การบวนการ Synthesis เพื่อแปลง HDL Code ที่เขียนอยู่ในระดับ gate-Level ต่อไป

ด้วยวิธีการออกแบบดังกล่าว ทำให้การออกแบบฮาร์ดแวร์ในปัจจุบันสามารถทำได้อย่างสะดวกรวดเร็ว ส่งผลให้มีผลิตภัณฑ์ด้านฮาร์ดแวร์ใหม่ๆ ออกสู่ท้องตลาดเป็นจำนวนมาก ผู้ออกแบบสามารถสร้างวงจรที่มีฟังก์ชันการทำงานตามที่ต้องการได้ภายในระยะเวลาอันสั้น

ภาษาบรรยายการทำงานของวงจร (HDL) ในปัจจุบันที่ใช้กันอยู่ 2 ภาษาคือ

1.Verilog มีโครงสร้างคล้ายภาษา C มักใช้ในงานสร้างระบบในเชิงพาณิชย์

2.VHDL พัฒนามาจากภาษา ADA ผู้พัฒนาคือ กระทรวงกลาโหมของสหรัฐอเมริกา มีโครงสร้างทางภาษาคล้ายกับภาษา PASCAL ทุกหน่วยงานที่ต้องการออกแบบหรือทำการพัฒนาไมโครโพรเซสเซอร์ให้กับกระทรวงนี้จะต้องใช้ภาษานี้ จึงทำให้ภาษานี้กลายเป็นภาษามาตรฐานในการนำไปออกแบบ

ภาษา VHDL มีลักษณะที่ผสมผสานกันระหว่าง Object Oriented Language และ Concurrent Programming Language โดยมองส่วนต่าง ๆ เป็นโมดูล (Module) หรือออบเจ็ก (Object)และมีสัญญาณเชื่อมต่อระหว่างกัน การส่งสัญญาณต่างๆ ในระบบเกิดขึ้นพร้อมๆ กันได้เช่นเดียวกับกระแสไฟฟ้าที่ไหลไปยังส่วนต่างๆ ของวงจร

ภาษา VHDL สามารถนำมาใช้ในการออกแบบวงจรได้ตั้งแต่วงจร Combination ขนาดเล็กไปจนถึงวงจรรวมขนาดใหญ่ เช่นไมโครโพรเซสเซอร์ได้ ซึ่งหากนักออกแบบวงจรดิจิตอลมี

ความเข้าใจในการเขียน VHDL Code ที่สามารถ Synthesis ได้แล้ว จะทำให้สามารถออกแบบวงจรที่มีความซับซ้อนได้ง่ายขึ้นอันจะส่งผลให้เกิดการพัฒนาทางด้านเทคโนโลยีในการที่จะสร้างอุปกรณ์ฮาร์ดแวร์ใหม่ ๆ ต่อไป

1.1 ความสำคัญและที่มา

การออกแบบวงจรดิจิตอลขนาดใหญ่ที่ระดับ Gate-Level นั้นเป็นงานที่ยุ่งยากมากและซับซ้อนเป็นอย่างมากเป็นผลให้งานที่พัฒนามีความล่าช้า และไม่เสร็จตามกำหนด การนำรูปแบบภาษาวีเอชดีแอล(VHDL) มาช่วยในการออกแบบจะช่วยแก้ปัญหานี้ได้

1.2 วัตถุประสงค์

เนื่องจากภาษาวีเอชดีแอล (VHDL)มีความรวดเร็วในการออกแบบและสามารถเข้าใจได้รวดเร็วเราจึงนำมาออกแบบทีวีแพทเทอร์นเจนเนเรเตอร์ ในโครงงานนี้ได้ทำการประยุกต์เขียนเป็นภาษาวีเอชดีแอล(VHDL)และใช้เทคโนโลยีของ FPGA เข้ามาช่วยในการสังเคราะห์สัญญาณเอ้าท์พุทที่ต้องการออกมา

1.3 ขอบเขตของโครงงาน

ทีวีแพทเทอร์นเจนเนเรเตอร์ที่ผู้จัดทำได้ขึ้น เป็นการประยุกต์จากโครงงานที่ต่อเป็น Hardware และทำการมอง Hardware ของทีวีแพทเทอร์นเจนเนเรเตอร์ ออกเป็นส่วนๆ และแต่ละส่วนเราก็ใช้ภาษาวีเอชดีแอลมาแทนที่ซึ่งก็จะสามารถผลิตสัญญาณกวาดทางแนวนอน (Horizontial signal), สัญญาณกวาดทางแนวตั้ง (Vertical signal) ,สัญญาณภาพรูปจุด,สัญญาณภาพรูปตาราง,สัญญาณภาพรูปแนวตั้ง,สัญญาณภาพรูปแนวนอน,สัญญาณแถบสี(RGB signal)

1.4 ขั้นตอนในการทำโครงงาน

1. ศึกษาการใช้ภาษาวีเอชดีแอล ซึ่งเป็นภาษาที่ใช้ในการอธิบายการทำงาน รวมถึงการใช้ซอฟท์แวร์ทูลส์ในการออกแบบ อันได้แก่โปรแกรม V-system, โปรแกรม Leonardo และโปรแกรม Xilinx Foundation

2.ศึกษาการสร้างสัญญาณทีวีโดยละเอียด

1. ออกแบบโมดูลด้วย ภาษาวีเอชดีแอล(VHDL)โดยแต่ละโมดูลจะแทนส่วนประกอบของทีวีแพทเทอร์เจนเนเรเตอร์

3. นำ soft core ที่ได้ของทีวีแพทเทอร์นเจนเนเรเตอร์ไปเข้าสู่กระบวนการ Synthesis เพื่อแปลง VDHL Code ให้เป็นวงจรในระดับ Gate-level และนำไปเข้าสู่กระบวนการ Place & Route เพื่อบันทึกลงสู่ FPGA ต่อไป
4. ทดสอบ FPGA ที่บันทึกโปรแกรมกับอุปกรณ์ฮาร์ดแวร์ว่าสามารถทำงานได้อย่างถูกต้องหรือไม่

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. เข้าใจขั้นตอนการออกแบบ,การทำงานและโครงสร้างภายในของเครื่องกำเนิดสัญญาณทดสอบภาพโทรทัศน์มากยิ่งขึ้น
2. สามารถใช้ภาษา VHDL ในการออกแบบวงจรดิจิตอลได้เป็นอย่างดี
3 .มีความเชียวชาญ ในการใช้ซอฟท์แวร์ทูลล์ในการออกแบบวงจรดิจิตอล
4. ได้รู้จักขั้นตอนการทำงาน,การวางแผน,การตรวจสอบการทำงานและแนวทางการแก้ไขปัญหาที่เกิดขึ้น

บทที่ 2
หลักการเบื้องต้นของสัญญาณโทรทัศน์

2.1 การสแกน

ในระบบโทรทัศน์จะมีการสแกนอยู่ 2 วิธีด้วยกันคือ การสแกนแบบก้าวหน้า (Progressive Scanning) กับการสแกนแบบสลับเส้น (Interlaced Scanning) การที่จะทำให้การสแกนมีความต่อเนื่องขององค์ประกอบภาพดังที่กล่าวมาแล้ว จะต้องคำนึงถึงหลัก 3 ประการคือ

1.ลำอิเล็กตรอนที่กวาดไปทางแนวนอน (Horizontal Scanning) ในแต่ละครั้ง จะต้องสามารถครอบคลุมองค์ประกอบภาพทั้งหมดของเส้นนั้น ๆ

2. ในแต่ละเส้นของการสแกนลำอิเล็กตรอน ลำแสงจะต้องกวาดด้วยความเร็วสูงไปยังด้านซ้ายเพื่อเริ่มต้นเส้นภาพทางแนวนอนลำดับต่อไป เวลาของการสะบัดกลับเราเรียกว่า "รีเทรซ" (Retrace) หรือฟลายแบ็ค (flyback) ในกรณีดังกล่าวจะต้องไม่มีข้อมูลภาพใด ๆ เพราะว่าทั้งกล้องถ่ายและหลอดภาพจะเกิดการแบล็งค์เอาท์ (Blank out)ในขณะนั้น

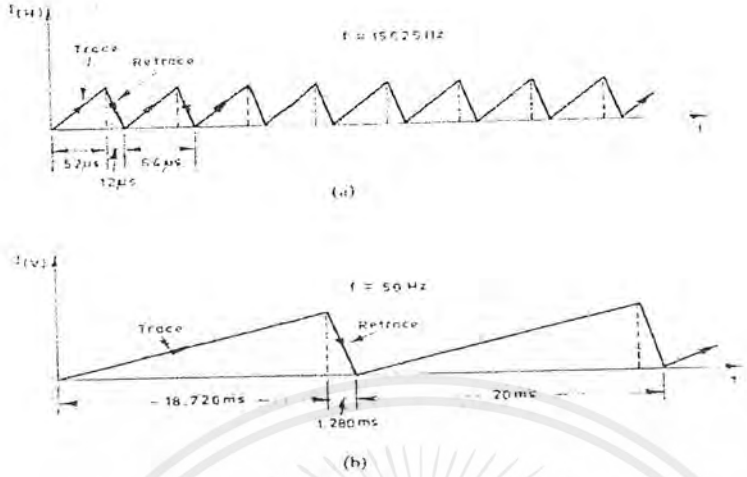
3. ในขณะที่เส้นสแกนสะบัดกลับมาเพื่อเริ่มต้นทางซ้ายใหม่ ตำแหน่งทางแนวตั้งต้องต่ำกว่าเดิมเพื่อทำให้การสแกนเส้นต่อไปไม่ทับกัน ทั้งนี้โดยการควบคุมของสัญญาณทางแนวตั้ง (Vertical Scanning)

2.1.1ระยะการสแกน (Scanning Periods)

รูปสัญญาณของการสแกนทางแนวนอนและการสแกนทางแนวตั้งแสดงดังในรูปที่ 2.4 ดังในรูปที่ 2.1(a) เป็นภาพของการสแกนทางแนวนอนจะเห็นว่ามีคาบเวลาเท่ากับ64ไมโครวินาที (1/15625Hz) ซึ่งจะแบ่งเป็นช่วงเวลาการเทรซ (trace)สัญญาณภาพจะเท่ากับ 52ไมโครวินาที และช่วงการรีเทรด (Retrace)หรือเรียกว่าช่วง ฮอริเซ็นทอร์นแบล็งกิง (Horizontal Blanking)จะเท่ากับ 12 ไมโครวินาที และเมื่อสแกนจบเส้นแล้วก็จะขึ้นเส้นใหม่ทางซ้ายเสมอ

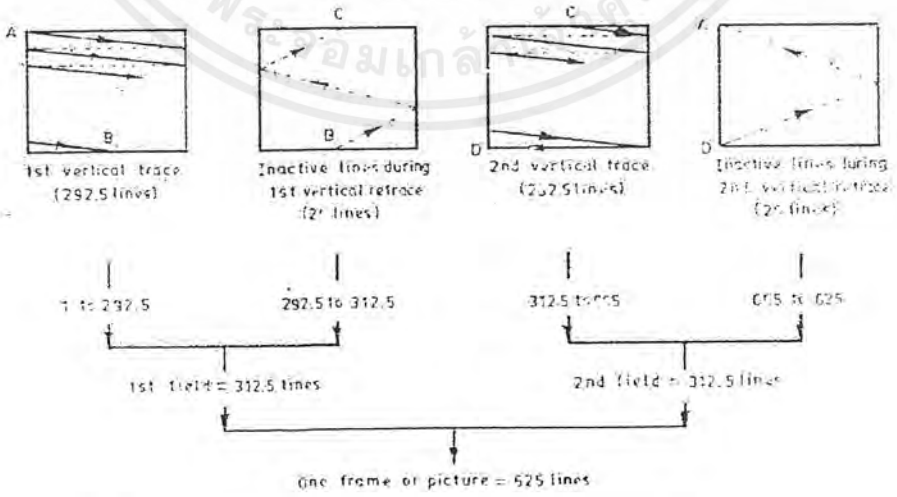
ส่วนรูปที่ 2.1(b) เป็นภาพของการสแกนทางแนวตั้งโดยจะมีคาบเวลาประมาณ 20 msซึ่งจะแบ่งเป็นช่วงเวลาการเทรด (trace)สัญญาณภาพจะเท่ากับ 18.720มิลิวินที และช่วงการรีเทรด (Retrace)หรือเรียกว่าช่วง เวอร์ติคอลแบล็งกิง (Vertical Blanking)จะเท่ากับ 1.280 มิลิวินที

ขั้นตอนการสแกน (Scanning Sequence)



รูปที่ 2.1(a) สัญญาณการสแกนทางแนวนอน
2.1(b) สัญญาณการสแกนทางแนวตั้ง

ตามมาตรฐาน ซีซีไออาร์ (CCIR) ใช้เส้นสแกน 625 เส้นต่อหนึ่งภาพ และใช้ภาพ 25 ภาพต่อวินาทีดังนั้นใน 1 ฟิลด์จะมีเส้นสแกน 312.5 เส้น ภาพนิ่งแต่ละภาพซึ่งเป็นส่วนประกอบขององค์ประกอบภาพจะเกิดขึ้นภายใน 1/25 วินาที ความถี่ที่ใช้เพื่อการหักเหลำอิเล็กตรอนในแนวนอนจึงได้จากจำนวน เส้นภาพ 625 เส้น คูณกับจำนวนภาพในแต่ละเฟรม ดังนั้นเราจึงสามารถหาความถี่ได้จาก 625x25 เท่ากับ 15,625 Hz ความถี่หักเหทางแนวตั้งจึงเท่ากับ 50 Hz ดัง รูปที่ 2.2



รูปที่ 2.2 แสดงลำดับของการสแกนภาพแบบสอดแทรก

จากรูปที่2.2 ได้แสดงวิธีการสแกนแบบสอดแทรกของระบบโทรทัศน์โดยเมื่อเริ่มต้นการสแกน สมมุติว่าการสแกนในกรณีนี้เริ่มจากการสแกนจากเฟรมที่เป็นเส้นสแกนคี่ โดยเริ่มจาก A ซึ่งอยู่ทางขวา นับเป็นเส้นสแกนเส้นที่ 1 แล้วจึงสแกน เส้นที่ 3,5,7,9 และต่อๆ ไป จนกระทั่งได้เส้นสแกน 312.5 เส้นในระบบซีซีไออาร์ ซึ่งก็คือสแกนมาถึงจุด B ดังในรูปภาพที่2.2 ณ.จุดนี้เส้นสแกนจะถูกความถี่หักเหทางแนวตั้งซึ่งเราเรียกว่า เวอร์ติคอล รีเทรซ (Vertical Retrace) หรือสัญญาณฟลายแบ็ค (flyback) ดึงกลับไปยังตำแหน่งในจุด C เพื่อเริ่มต้นการสแกนเส้นคู่ต่อไป

เวลาของการรีเทรซ (Retrace Time) ทั้งการรีเทรซทางเวอร์ติคอลและฮอริซอนทอลเป็นเวลาสั้น ๆ ถึงอย่างไรก็ตามเราก็ไม่ต้องการให้เส้นสแกนที่เป็นช่วงของการสะบัดกลับนี้เข้ามารบกวนการทำให้เกิดสัญญาณภาพ ในส่วนนี้จึงต้องทำการลบเส้นสะบัดกลับ ก่อนที่จะถึงจุดที่ว่านั้นเรามาดูรายละเอียดของการสะบัดกลับในส่วนของการกวาดลำแสงหรือการสแกนในทางแนวนอน เวลาของการรีเทรซจะประมาณ 10-16 เปอร์เซ็นต์ของเวลาทางสแกนทางแนวนอนถ้าระบบ ซีซีไออาร์ เราใช้เวลาในการสแกนเท่ากับ 64ไมโครวินาที ดังนั้นเวลาในการรีเทรซจะประมาณ 6.4 ไมโครวินาทีส่วนของทางด้านความถี่หักเหทางแนวตั้งเราใช้เวลาในการรีเทรซไม่เกิน5-8 เปอร์เซ็นต์ อย่างเช่นเราใช้เวลาในส่วนนี้เท่ากับ 3 เปอร์เซ็นต์จะใช้เวลารีเทรซเท่ากับ 600 ไมโครวินาที นั่นหมายความว่าในช่วงของการรีเทรซทางแนวตั้งกินเวลานานกว่าการสแกนทางแนวนอนประมาณ 8-10 เส้นภาพ

ดังนั้นจากหลักการดังกล่าวเราสามารถสรุปได้ว่าตามความเป็นจริงแล้ว เส้นภาพ 625 เส้นนั้นเรามิอาจจะเห็นได้ครบทุกเส้น อย่างน้อย ๆ ในกรณีที่เกิดเวอร์ติคอลรีเทรซจะกินเวลาของการสแกนทางแนวนอนไปด้วย แต่จะล่วงเลยไปกี่เส้นนั้นแล้วแต่การบังคับการฟลายแบ็ค ซึ่งในเครื่องรับเราเรียกตัวนี้ว่าสัญญาณแบล็งกิ้ง

สัญญาณภาพรวม(Composite Video Signal)

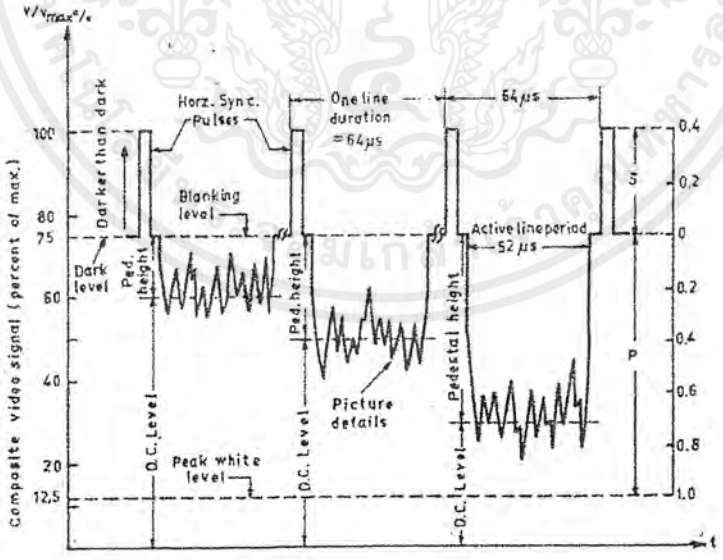
ในโทรทัศน์ขาวดำสัญญาณภาพรวมเกิดจากสัญญาณกล้องถ่ายภาพซึ่งจะมีการเปลี่ยนแปลงตามภาพ แบล็งกิ้ง(blanking)เป็นสัญญาณที่ใช้เพื่อลบเส้นสะบัดกลับและสัญญาณซิงโครไนส์(Synchronizing Pulses)ใช้ในการสแกนให้เครื่องรับและเครื่องส่งทำงานสอดคล้องกัน ซิงก์ทางแนวนอน(horizontal sync)แสดงเป็นคาบเส้นด้วยเหตุนี้ ซิงก์ทางแนวตั้งต้องการหลังจากแต่ละฟิลด์ของการสแกน โดยขนาดของพลัส์ทางแนวตั้งและแนวนอนจะรักษาประสิทธิภาพให้สูงเหมือนกับทางด้านส่งซิงก์พลัส์ต้องต่อเนื่องกัน และส่งพร้อมสัญญาณภาพโดยใช้วิธีการส่งแบบแบ่งเวลาของสัญญาณภาพ

สัญญาณโทรทัศน์สีจะมีการส่งข้อมูลเกี่ยวกับสีของภาพ และซิงก์สี(burst)โดยจะมีการถอดแบบซิงก์สีทางด้านรับซึ่งสัญญาณสี(chroma)และซิงก์สีถูกบรรจุความกว้างในช่องเดียวกัน

ความสำคัญของสัญญาณวีดีโอ

จากรูปที่2.3 แสดงรายละเอียดของเส้นที่แตกต่างกันสามเส้น ซึ่งมีลักษณะที่คล้ายกันแตกต่างกันเพียงระดับความสว่างของสีขาวและสีดำของภาพ โดยในที่นี้จะอธิบายรายละเอียดที่ทำให้สัญญาณภาพมีลักษณะเดียวกับทางด้านส่ง โดยเรียงตามความสว่างที่ สูงสุดเป็นลำดับการอ้างอิงระดับฟีคสีขาวกำหนดที่ 10-12.5 % ของค่าสูงสุดของสัญญาณซึ่งจะมีลักษณะเดียวกับระดับดำประมาณ 72 % ซิงก์พลัสรวมที่ระดับ 75 % เรียกว่าระดับแบล็งกิ้ง ความแตกต่างระหว่างระดับดำและระดับแบล็กกิ้งแสดงเป็นแท่นเชิง ซึ่งแสดงดังรูป ดังนั้นข้อมูลจะอยู่ระหว่าง 10 % จนถึง 75 % ของสัญญาณภาพรวม ซึ่งความสว่างจะสัมพันธ์กับรูปภาพแล้วแต่กรณี โดยลำดับของสัญญาณรบกวนที่น้อยที่สุดจะมีระดับแรงดันต่ำกว่า 10 %

ทางด้านหลอดภาพของเครื่องรับ โดยพื้นฐานของแรงดันวิดิโอทางด้านรับจะมีลักษณะเดียวกันในส่วนของการมอดูเลต 10 %ที่จุดหน้าจอคล้ายคลึงกันโดยการกำหนดคล้ายคลึงกับรูปแบบของระดับดำ นอกจากนี้เครื่องรับโทรทัศน์สามารถทำการปรับความสว่างและความคมชัดของภาพได้ตามความต้องการ



รูปที่ 2.3 สัญญาณแสดงรายละเอียดของของเส้นสแกนทั้งสามเส้นโดยที่ระดับความสว่างเฉลี่ยต่างกัน อัตราส่วนของรูปต่อซิงก์ P/S = 10/4

ส่วนประกอบดีซีของสัญญาณวีดีโอ

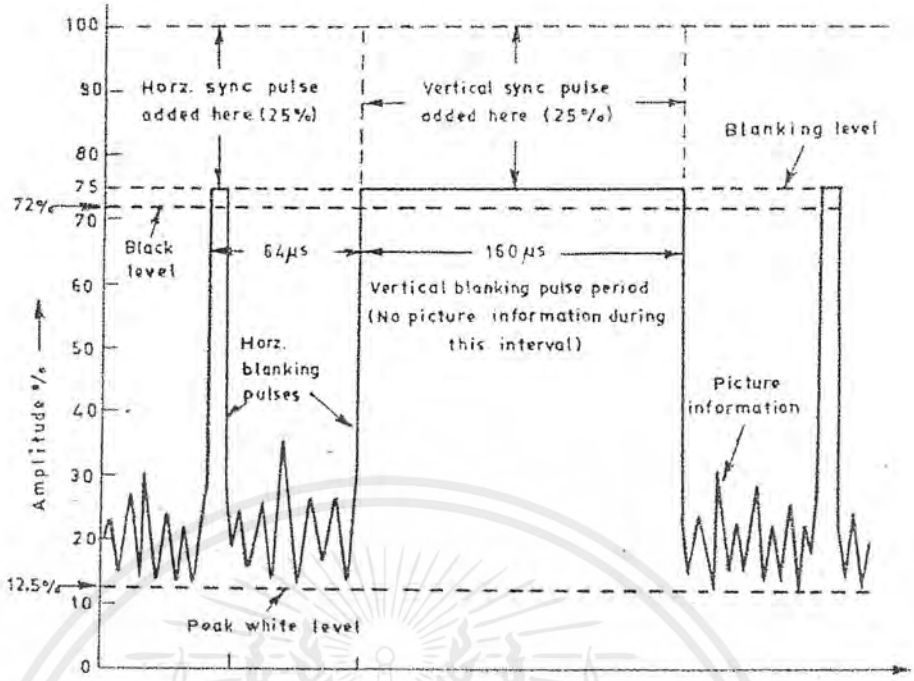
ในการเปลี่ยนแปลงของขนาดอย่างต่อเนื่องของภาพแต่ละส่วน ค่าเฉลี่ยของสัญญาณวิดิโอหรือส่วนประกอบของกระแสตรงมีลักษณะเช่นเดียวกับค่าเฉลี่ยความสว่างของภาพถ้าไม่มีส่วนประกอบของดีซีทางเครื่องรับไม่สามารถปรับค่าความสว่างเช่นเดียวกับสัญญาณเอซีของกล้องถ่ายภาพ สำหรับส่วนประกอบภาพสีเทาบนพื้นสีดำเหมือนกับพื้นสีขาวบนพื้นที่สีเทา ในรูป 2.3 ส่วนประกอบของสัญญาณทั้งสามเส้นมีลักษณะที่คล้ายกัน แตกต่างกันที่ระดับค่าเฉลี่ยความสว่างของภาพมีความสัมพันธ์กับสัญญาณ ส่วนประกอบของดีซีของสัญญาณวีดีโอจะเป็นค่าเฉลี่ยของส่วนประกอบของเฟรมมากกว่าเส้นจากข้อมูลความสว่างของรูปภาพ ดังนั้นรูป 2.3 อธิบายลำดับการเปลี่ยนแปลงค่าเฉลี่ยของความสว่างของภาพสามารถเปลี่ยนแปลงได้อย่างเดียวจากเฟรมไปเฟรม หรือจากเส้นไปเส้น

แท่นเชิง

จากรูป2.3 แท่นเชิงสูงเป็นระยะห่างระหว่างระดับแท่นเชิงกับระดับดีซีเฉลี่ยของสัญญาณวิดิโอ ภาพความสว่างเฉลี่ยวัดต่างจากค่าเฉลี่ยของระดับ แม้ว่าการสูญเสียของสัญญาณที่ผ่านวงจรคาปาซิเตอร์-คัฟเปอร์ ระยะห่างระหว่างแท่นเชิงกับค่าเฉลี่ยกระแสตรง ดังนั้นเหมาะกับการใช้ระดับแท่นเชิงอ้างอิงระดับกับการแสดงค่าเฉลี่ยของความสว่างของภาพ

การกำหนดระดับแท่นเชิง

สัญญาณเอาท์พุตของกล้องทีวีเริ่มจากขนาดที่เล็กมากซึ่งผ่านการหน่วงเวลาหลายครั้งโดยตัวขยายคัฟเปอร์ที่มีเกนการขยายสูงจากการป้อนกลับมันจะควบคุมการขยายโดยที่ค่าซิงก์ และแบล็กกิ้งพลัสเป็นการรวมและยกระดับที่เหมาะสมจากแท่นเชิงเมื่อมีการหาความสูงของแท่นเชิงของค่าเฉลี่ยของความสว่างของภาพจะมีค่าน้อยมากเพื่อทำให้ภาพเหมาะสมซึ่งถ้าแท่นเชิงสูงกว่าย่อมจะส่งผลให้ค่าเฉลี่ยของความสว่างสูงมาก ซึ่งควบคุมได้จากภาพบนจอมอนิเตอร์ของเครื่องรับ เป็นที่รู้กันการใส่ดีซีเท่ากับการรวมเข้ากับเอซีการใส่ดีซีของแท่นเชิงที่สมบูรณ์จะอ้างอิงสีดำและการแสดงแท่นเชิงสูงที่เหมาะสมจะสัมพันธ์กับความสว่างของภาพจำลอง ในการส่งระดับดีซีในการควบคุมการขยายเพื่อควบคุมช่วงการขาดหายซึ่งสามารถทำการคัฟปิ้งได้แม้กระนั้น ส่วนประกอบดีซีสามารถตรวจแก้ได้เมื่อมีความจำเป็นเพราะว่าการกำหนดแท่นเชิงสูงยังคงเหมือนเดิม



รูปที่2.4 พลัส์แบล็งก์กิ้งทางแนวตั้งและแนวนอนเป็นสัญญาณวีดีโอซิงก์พลัส์เป็นการรวมระดับแบล็งก์กิ้งที่ตำแหน่ง 25 %ของขนาดสัญญาณรวม

สัญญาณแบล็กกิ้ง

สัญญาณภาพรวมจะบรรจุแบล็กกิ้งซึ่งใช้เป็นเส้นสะบัดกลับซึ่งจะไม่ปรากฏการรวมขนาดของสัญญาณในการส่งมีระดับสีดำ 75 % ของช่วงเวลาในการสแกนซ้ำรอยเดิมอธิบายในรูป2.4 สัญญาณภาพรวมจะบรรจุพลัส์ทางแนวนอนและแนวตั้งมีลักษณะเดียวกับแบล็งก์ อัตราการทำซ้ำของสัญญาณซิงก์ทางแนวนอนมีค่าเท่ากับความถี่ในการสแกนมีค่าเท่ากับ 15,625 Hz และทำนองเดียวกันความถี่ทางแนวตั้งจะเท่ากับฟิลด์สแกนมีค่าเท่ากับ 50Hz ซึ่งมันอาจมีระดับแบลงก์กิ้งที่แตกต่างกันขึ้นอยู่กับสัญญาณข้อมูลของภาพบางครั้งจะมีลักษณะเช่นเดียวกันส่วนที่มีค่าสูงสุดของรูปซึ่งเกี่ยวข้องกับระดับแบลงก์กิ้งและอาจเกิดการทรอดแทรกโดยของการสแกนซิงโครไนส์ของเครื่องกำเนิดในการออกแบบควรให้ซิงก์พลัส์ความถี่ในการสแกนส่งในระดับ 25 % (75 % ของ 100 %) ของคลื่นพาหะสัญญาณวิดิโอและส่งพร้อมสัญญาณภาพ

ซิงซ์สี(Color Burst)

สัญญาณโครม่า(Chroma)ในระบบ CTV เป็นการมอดูเลตของพาหะเรียกว่า คลื่นพาหะรองของสีและร่วมกับสัญญาณความสว่าง ซิงก์สีมี 8-10 ไซเคริส ของคลื่นพาหะรองของสีและระหว่างแบล็กพล็อชของแต่ละซิงก์ทางแนวนอน

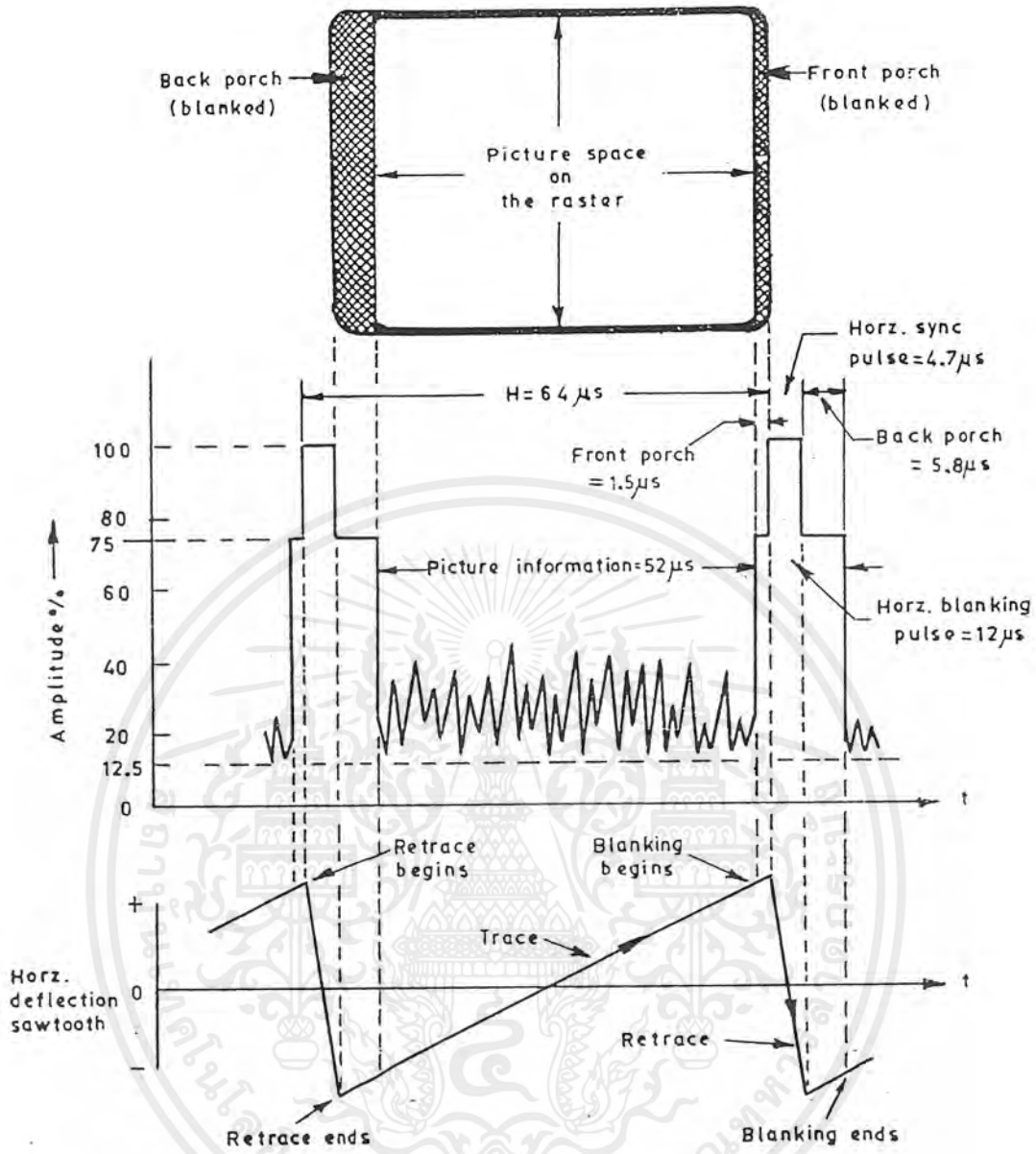
2.2 ส่วนประกอบซิงก์ทางแนวนอน

รายละเอียดของคาบซิงก์ทางแนวนอนและซิงซ์พลัส์อธิบายในรูปที่ 2.6 เส้นสแกนทางแนวนอนจะแทนด้วย H และมีคาบเวลาทั้งหมด 64 μS เป็นคาบเวลาเส้นแบลงก์ 12 μS ระยะระหว่างซิงก์พลัสที่ใส่ พลัส์มีลักษณะเดียวกันอ้างอิงขาขึ้นของซิงซ์พลัส์เป็นการใช้ซิงซก์ของเครื่องกำเนิดในการสแกนทางแนวนอน เหตุผลดูจากรูปที่ 2.6 และรูปอื่นๆต่อไปความแตกต่างของเวลาทั้งหมดเป็นแสดงระหว่างขอบขาขึ้นของซิงก์พลัส์

คาบของเส้นแบล็งก์แบ่งออกเป็นสามส่วนคือ ฟรอนท์ฟอท พลัสซิงซก์ แบล็งก์พอท ซึ่งความแตกต่างของเวลาที่ยอมให้ในแต่ละส่วนอย่างย่อๆซึ่งขอบเขตและผลบนจออธิบายในรูปที่2.6

ตารางที่ 2.1 รายละเอียดของการแสดงทางแนวราบ

Period	Time(μS)
Total line	64
Horz blanking	12±0.3
Horz sync pulse	4.7±0.2
Fornt porch	1.5±0.3
Back porch	5.8±0.3
Visible line time	52



รูปที่ 2.5 ส่วนประกอบรายละเอียดเส้นทางแนวนอนและซิงก์ฟันเลื่อยทางแนวนอนและระยะห่างระหว่างขอ

ฟอนท์ฟอร์ท

คาบจะลด 1.5 μs ระหว่างด้านของรายละเอียดภาพสำหรับเส้นและขอบขาขึ้นของเส้นซิงซ์พลัส์เวลาที่ยอมให้วงจรวีดีโอทางเครื่องรับลดลงอย่างราบรื่นจากระดับแรงดันของภาพคงที่ใดๆของด้านของเส้นรูปของระดับแบล็งก์กิ่งก่อนซิงก์พลัส ดังนั้นวงจรซิงก์ทางด้านรับต้องแยกรายละเอียดของภาพให้เป็นไปตากกฎบังคับเป็นการทำเมื่อทำวงจรวีดีโอเมื่อฟีคขาวปรากฏที่ด้านของเส้น

เส้นซิงก์พลัส์

หลังฟรอนท์ฟอยด์ของแบลงก์กิ้งแนวนอนจะสะบัดกลับก่อนการเริ่มซิงก์พลัส์ ความรวดเร็วจะเป็นการกำหนดแบล็งก์กิ้งเพราะว่าระดับซิงก์จะดำกว่าดำเส้นซิงซ์พลัส์จะทำการแยกทางเครื่องรับเพื่อให้เป็นเส้นเวลาพื้นฐานทางเครื่องรับ ซึ่งจะทำให้ซิงก์พลัส์มีความถูกต้องเช่นเดียวกับทางด้านส่งช่วงเวลาของซิงก์พลัส์เป็น 4.7ไมโครวินาที ส่วนประกอบระหว่างคาบลำแสงของจอมันเป็นการสะบัดกลับและจะปรากฏทางด้านซ้ายสุดของจอภาพ

เบล็กฟอยด์(back fornt)

ระดับแบลงก์กิ้งที่สมบูรณ์จะมีคาบเวลานปีน 5.8ไมโครวินาที สำหรับส่วนของเวลาการสะบัดกลับประกอบด้วยเวลาฟรีมิชของวงจรพื้นฐานทางแนวนอนในทิศทางสวนกระแสในการสแกนเส้นต่อไป ความสัมพันธ์ของเวลาเป็นการกำหนดบาร์ดำขนาดเล็ก(ดูรูป2.6)เป็นรูปแต่ละด้านของจอในระนาบแนวนอน บาร์ดำที่ด้านข้างไม่มีผลกับการผลิตรายละเอียดของรูประหว่างคาบเส้นที่ทำงาน

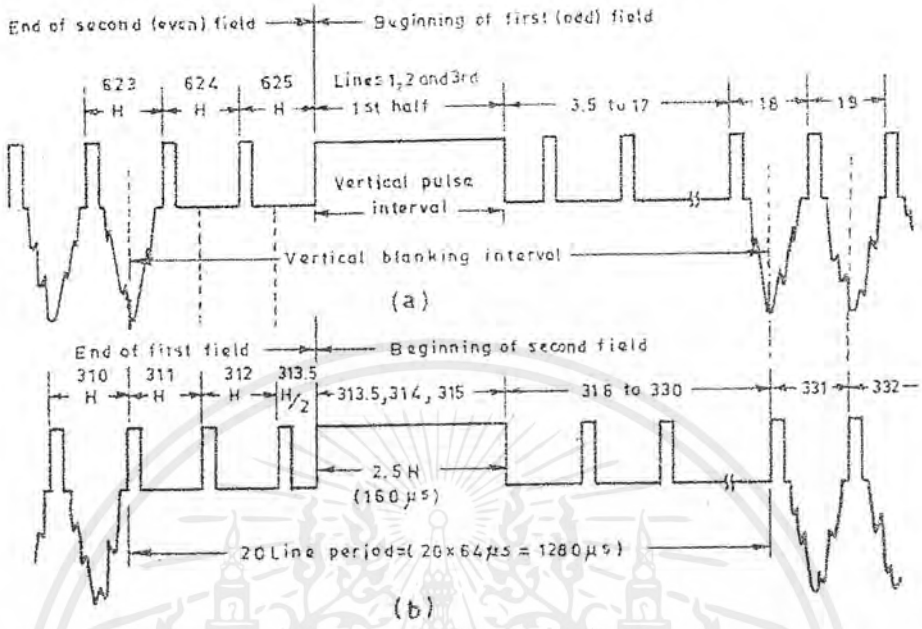
เบล็กฟอยด์ (back fornt)ประกอบด้วยขนาดที่จำเป็นซึ่งจะเท่ากับระดับแบล็งก์กิ้งซึ่งเป็นระดับอ้างอิงและการยอมรับรวมถึงการรักษาระดับดีซีของข้อมูลภาพทางด้านส่ง ที่ด้านรับระดับจะขึ้นอยู่กับรายละเอียดของภาพซึ่งให้ประสิทธิภาพดีที่สุดโดยใช้วงจรเอจีซี(Automatic Gain Control) จะทำหน้าที่ขยายกำลังให้เพิ่มขึ้นโดยการกำหนดซิงก์สีอย่างง่ายในCTV คลื่นพาหะอยู่ระหว่าง เบล็กฟอนด์(back fornt)

2.3รายละเอียดของซิงก์ทางแนวตั้ง

พื้นฐานการรวมซิงก์ของฟิลด์คู่และฟิลด์คี่แสดงในรูป 2.4ซึ่งจะมีความกว้างมากกว่าซิงก์พลัสทางแนวนอนในลำดับการขับฟิลด์ซิงก์พลัส์ที่เหมาะสมของการทริกเกอร์ของออสซิเลเตอร์ทางเครื่องรับในการกวาดฟิลด์ โดยมาตราฐานซิงก์พลัส์ทางแนวตั้งมีคาบเวลา 2.5 ถึง 3 คาบของเส้นทางแนวนอน ถ้าความกว้างน้อยกว่านี้การแยกซิงก์พลัสทางแนวตั้งและแนวนอนเครื่องรับจะทำการแยกลำบาก

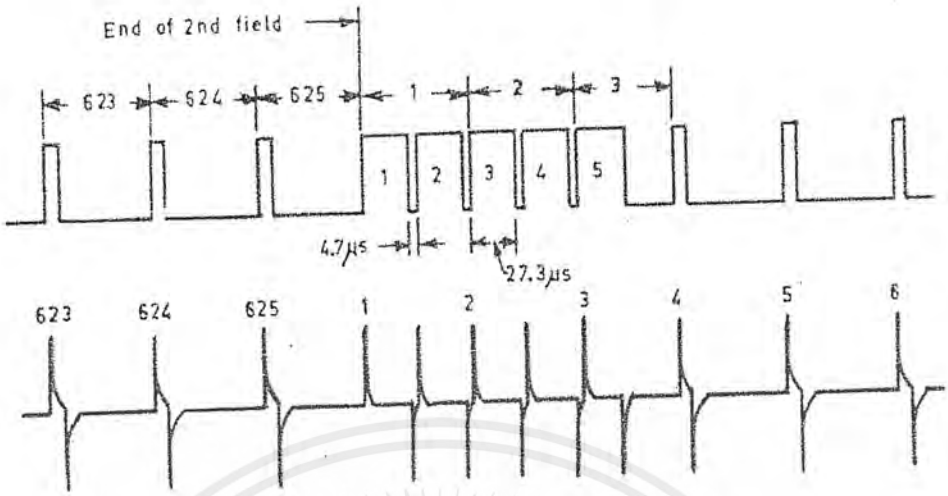
ระบบ 625 เส้น มีคาบเส้น 2.5 (2.5x64=160ไมโครวินาที) โดยหนึ่งเฟรมจะมีการแบ่งเป็นสองฟิลด์คือฟิลด์คู่และฟิลด์คี่ซึ่งจะสแกนฟิลด์คี่ก่อนดังนั้นจะเริ่มสแกนเส้นที่ 1 ของทั้งหมด 313 เส้นและจะไปจบเส้นที่ 315 ซิงก์พลัส์ถัดไปต้องการระยะห่าง 20,มิลลิวินาที จากรูป2.4 เริ่มจากพลัสที่มีเส้นเดียวกันอาจเกิดหลังแต่ละด้านของการดำเนินการทางแนวตั้งของบีมในแต่ละฟิลด์โดยในแต่ละอันมี 1/50ของลำดับที่สอง เส้นตรงของซิงก์พลัส์ทางแนวตั้งหรือที่ด้านจบของครึ่งเส้นและคาบเส้นเต็มผลของความสัมพันธ์ไม่เป็นเส้นตรงของซิงก์ทางแนวราบ อย่างไรก็ตามพิจารณารายละเอียดของขบวนพลัส์ทั้ง 2 ฟิลด์ควรจะมีซิงก์ทางแนวราบอย่างต่อเนื่องมีระยะห่าง

ขบวนพลัสทั้ง 2 ฟิลด์ควรจะมีซิงก์ทางแนวราบอย่างต่อเนื่องมีระยะห่าง 64 μsโดยคาบการสแกนจากเฟรมถึงเฟรมและมีการชิฟ 32 μs ระหว่างซิงก์ทางแนวตั้ง



รูป 2.6 แสดงสัญญาณภาพรวม ซิงก์พลัสทางแนวนอนและแนวตั้งในแต่ละด้านของ
(a)ลำดับที่สอง(คู่) (b)ฟิลด์ลำดับที่หนึ่ง

ข้อมูลของซิงก์ทางแนวราบเกิดจากขบวนพลัสซึ่งมีความแตกต่างกันโดยขบวนพลัสที่มีความถี่สูงผ่านฟิลเตอร์ได้ที่จริงพลัสมีลักษณะเหมือนกันแตกต่างกันที่ขอบขาขึ้นของซิงก์พลัสซึ่งใช้การสแกนทางแนวนอนจากการซิงโครไนส์ออสซิเลเตอร์ซึ่งแสดงในรูป 2.7 ซิงก์พลัสทางแนวราบเหมาะที่จะใช้ระหว่างการดำเนินการและคาบเส้นแบล็งก์ดังนั้นการกวาดทางแนวนอนของออสซิเลเตอร์ทำงานที่ความถี่ 15,625 Hz ควรจะดูแลลำดับของการซิงโครไนส์ระหว่างคาบซิงก์ทางแนวตั้ง แต่ละอันตำแหน่งของฟิลด์คี่เป็นคู่ไม่ดีแสดงในรูป2.4 คาบแบลงก์ทางแนวตั้งของด้านฟิลด์คี่เริ่มจากครึ่งทางโดยเส้นทางในแนวราบเรียงตามลำดับจากรูปจะเห็นว่าขอบขาขึ้นของพลัสซิงก์ทางแนวตั้งที่เข้ามีคาบเวลาของซิงก์ที่ไม่เหมาะสมกับคาบซิงซ์ของออสซิเลเตอร์ทางแนวราบโดยเริ่มจากการตัดสล็อตในพลัสซิงซ์ทางแนวตั้งที่ช่วงระหว่างคร่งเส้นของคาบซิงซ์พลัสทางแนวราบ โดยยกตัวอย่างของฟิลด์คู่และฟิลด์คี่ที่ถูกต้อง เทคนิคในการทำให้ขนาดของสัญญาณวีดีโอกลับไปยังระดับแบล็งก์ 4.7 μs ดังนั้นสล็อตแคบจะมีความกว้าง 4.7 μsรูปซิงก์พลัสของแนวตั้งซึ่งแต่ละด้านมีคาบเวลาท่ากับ 32 μsขาของพลัสจะใช้ทริกเกอร์ทางแนวนอนของเครื่องกำเนิดผลของรูปคลื่นอื่นๆกับเส้นจำนวนของความแตกต่างของเอาท์พุตของคู่ขบวนฟิลด์อธิบายในรูป 2.8ในการใส่พลัสสั้นๆเป็นรอยบากหรือฟันเลื่อยสัญญาณฟิลด์พลัส



รูปที่ 2.7 (a) พลัส์ด้านฟิลด์คี่และมีเอาท์พุทมีลักษณะเหมือนกัน(H.P.F)

(b) พลัส์ ทางด้านฟิลด์คู่ สังเกตุความแตกต่างจากจำนวนเส้นเป็นที่ต้องการอย่างเดียว ของด้านในแต่ละฟิลด์

สังเกตุพัสล์ทางแนวตั้งไม่เรียบโดยซิงก์พลัส์ทางแนวนอนยอมให้ผลของพลัส์ทางแนวตั้งไม่มีการเปลี่ยนแปลง แม้กระนั้นส่วนที่เหลือเกี่ยวกับระดับแรงดันแบล็งก์กิ่งของเวลาทั้งหมดที่ใช้ทำ ความกว้างของพลัส์ยังคงมีความกว้างกว่าพลัสทางแนวนอน ซึ่งทางเครื่องรับสามารถแยกได้ง่ายกลับไปดูรูป 2.8 มันเหมือนกัน ดังนั้นซิงก์พลัสทางแนวนอนยอมให้เอาท์พุตมีปลายแหลมซึ่งเป็นวงจรดิฟเฟอเรอนท์เชียล โดยกำหนดเวลาคอยเปลี่ยนเวลาที่แสดงระหว่างพลัส์ที่ขอบขาขึ้นเพื่อลดการเสื่อมลงพลัสทริกทางด้านลบอาจใช้ไดโอดอย่างเดียวซึ่งกำหนดให้พลัส์บวกผ่านเป็นการล็อกออสซิเลเตอร์ทางแนวนอน

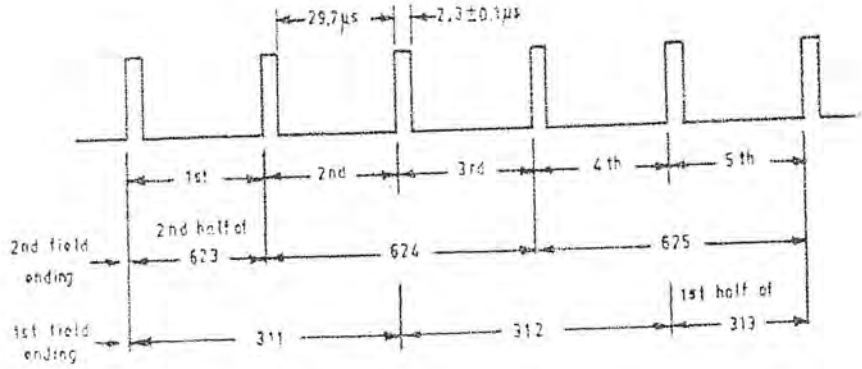
อย่างไรก็ตามพลัส์ตามความเป็นจริงจะมีการเรียงลำดับ ดังนั้นพลัสที่กับจำนวนเส้นของฟิลด์คู่สังเกตุระยะเวลาระหว่าง 64 μs ของขบวนพลัส์มีขอบคล้ายฟันเลื่อยทางแนวตั้ง ความแตกต่างของครึ่งเวลาในการเริ่มแต่ละฟิลด์ ซิงก์พลัสทางแนวตั้งอยู่ระหว่าง 32 μsพลัส์ของออสซิเลเตอร์การกวาดทางแนวราบไม่ต้องการ 64 μs โดยเราไม่สนใจการทริกเกอร์ของออสซิเลเตอร์ ดังนั้นต้องการวงจรล็อกในการกวาดทางแนวราบแม้จะมีการใส่ซิงก์พลัส์ทางแนงตั้งก็ตาม

ในการแยกซิงก์พลัสทางแนวตั้งของขบวนพลัส์โดยผ่านวงจรอินทิเกเตอร์นำพลัส์รวมไปขยายโดยซิงโครไนส์ของการกวาดทางแนวตั้งของออสซิเลเตอร์ของเครื่องรับจุดสำคัญของวงจร

อินทิเกเตอร์คือความถี่ต่ำสามารถผ่าน RC ฟิลเตอร์โดยการเลือกเวลาคงตัวในการผลิตซิงพลัส์ช็อตโฮลที่เอาท์พุตน้อยมากโดยการขยายซิงก์พลัสทางแนวตั้งแม้ว่าการสร้างฟันเลื่อยจะมีความสัมพันธ์กับระดับแรงดันเอาท์พุตที่ผ่านอินทิเกรตคาปาซิเตอร์อย่างไรก็ตามการสร้างระดับแรงดันอ้างอิงขึ้นสำหรับแต่ละฟิลด์โดยเริ่มจากซิงซ์พลัสทางแนวนอนฟิลด์แรกซึ่งมี 625 เส้นเป็นการแยกจากลำดับที่ 1 ของพลัสทางแนวตั้งโดยเดิมหนึ่งเส้นและมีแรงดันผ่านคาปาซิเตอร์ฟิลเตอร์ซึ่งอาจมีเวลาเพียงพอในการสะบัดกลับเป็นศูนย์ ก่อนจะถึงพลัสทางแนวตั้งดังนั้นการสร้างระดับแรงดันเอาท์พุตของฟิลเตอร์เริ่มจากศูนย์ของเรสปอนด์ซิงก์พัลส์ทางแนวตั้งพื้นฐานที่ต่อเนื่องจะสร้างระดับแรงดันที่มากขึ้น เพราะว่าคาปาซิเตอร์มีเวลาชาร์จ์มากกว่าดิสชารจ์อย่างไรก็ตามการเริ่มของฟิลด์ที่สองไม่เหมือนกัน โดยที่ซิงก์พลัส์จะมีลักษณะต่อเนื่องเช่นเดียวกันโดนเริ่มที่เส้น 313 ซึ่งเป็นการแยกซิงซ์พลัสทางแนวตั้งของลำดับที่หนึ่งโดยมีครึ่งเส้นแรงดันที่ผ่านฟิลเตอร์ในแนวตั้งแม้ว่าจะไม่มีเวลาเป็นศูนย์สำหรับการดึงพลัส์ลำดับที่1ของแนวตั้งโดยที่ความสำคัญของการสร้างแรงดันไม่ได้เริ่มจากศูนย์ดังเช่นในฟิลด์ของลำดับที่หนึ่งแรงดันที่เหลือแสดงผลของการขัดแย้งกันของแรงดันพลัส์ทางแนวตั้งพื้นฐานดังนั้นแรงดันที่ผ่านฟิลเตอร์คาปาซิเตอร์บางส่วนในเรื่องฟิลด์ออสซิเลเตอร์ลำดับที่สองทริกเกอร์อาจไม่สำคัญมากดังนั้นการเปรียบเทียบของฟิลด์ลำดับที่หนึ่งโดยความแตกต่างของเวลาการทริกเกอร์อาจจะหยุดสั้นแต่มันไม่เพียงพอที่จะทำให้การสแกนไขว้เขว

อีควอไรสซิ่ง พลัส์

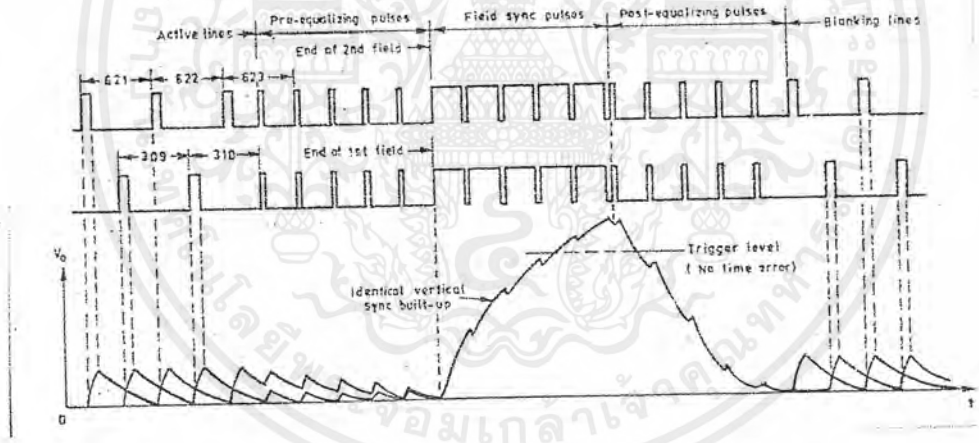
ในเรื่องการเคลื่อนที่ซึ่งแสดงการอธิบายของ ครึ่งเส้นที่ขัดแย้งกันของพลัส์แคบฯ 5 พลัส์การรวมขนาดที่เหมือนกันของซิงก์พลัส์ทางแนวตั้งเป็นที่รู้กัน ฟรีอีควอไรสซิ่งพลัส์และโพสอีควอไรสซิ่งพลัส์การกำหนดแต่ละอันเกิดจากพลัส์ที่มีขนาดแคบ 2.5 คาบเส้นของแต่ละขนาดซิงก์พลัส์ทางแนวตั้งรายละเอียดของฟรีอีควอไรสซิ่งพลัส์และโพสอีควอไรสซิ่งพลัส์กับจำนวนเส้นของแต่ละฟิลด์โดยพิจารณารูป2.9ผลของพลัส์มีการชิฟท์คู่ของเส้นคู่ครึ่งที่ไม่ตรงกัน ฟรีอีควอไรซิ่งมีผลเป็น 2-3 μs ในการดิสชารจ์คาปาซิเตอร์ จุดที่สำคัญของระดับแรงดันเป็นศูนย์ในฟิลด์คู่แม้จะมีเส้นครึ่งไม่ตรงกันการสร้างแรงดันเพิ่มจะรวมไปถึงซิงซ์พลัสทางแนวตั้ง ในรูปที่ 2.10 อธิบายโฟสอีควอไรสซิ่งจำเป็นในการดิจชาร์จ์ที่เร็วของคาปาซิเตอร์ในการยอมรับการทริกเกอร์ของออสซิเลเตอร์ทางแนวตั้งที่เวลาเหมาะสม



รูปที่ 2.8 ฟรีชิงก์อีควอไรสซิ่งและโพสชิงก์อีควอไรสซิ่ง

แรงดันที่ผ่านคาปาซิเตอร์มีความช้าน้อยลง อาจเกี่ยวกับพลัส์ออสซิเลเตอร์ของโพสชิงซ์อีควอไรสซิ่งอาจจะทริกขาขึ้นควรจะก่อนผิดพลาด

ในการใส่ฟรีและโพสอีควอไรสซิ่งพลัส์แคบฯเกิดแรงดันและรูปร่างจะลดลงตามลำดับของฟิลด์(รูปที่ 2.9)และออสซิเลเตอร์ทางแนวตั้งจะทริกตัวอย่างที่ถูกต้องใน 1/50ของลำดับที่สอง



รูปที่ 2.9 การสร้างแรงดันของชิงก์ทางแนวตั้งโดยผ่านวงจรอินทิเกรตติ่งคาปาซิเตอร์

2.4 หน้าที่ของขบวนพลัส์ทางแนวตั้ง

ชิงก์พลัส์ฟันปลาทางแนวตั้งและข้อกำหนดฟรีและโพสอีควอไรสซิ่งพลัส์ความต้องการพื้นฐานที่จำเป็นสำหรับการยอมได้ในการสแกน

(1) ฟิลด์ชิงก์พลัส์ที่เหมาะสมคือการทริกเกอร์ของฟิลด์ออสซิเลเตอร์

(2) ความต่อเนื่องของเส้นของออสซิเลเตอร์ของพลัส์ทริกเกอร์ทางเครื่องรับโดยตรวจแก้กความแตกต่างในตอนแรกและในตอนสุดท้ายของเวลาพื้นฐานฟิลด์ในการกระทำ

(2) ความต่อเนื่องของเส้นของออสซิเลเตอร์ของพลัสทริกเกอร์ทางเครื่องรับโดยตรวจแก้กความแตกต่างในตอนแรกและในตอนสุดท้ายของเวลาพื้นฐานฟิลด์ในการกระทำ

(3) ความเป็นไปได้ในการใส่ซิงก์พลัสทางแนวตั้งที่ด้านของเส้นหลังจากฟิลด์ที่สองและที่ ตัวกลางของเส้นที่ฟิลด์แรกโดยไม่ให้มีความผิดพลาดในการทับกัน

(4) สร้างซิงก์พลัสเพิ่มทางเครื่องรับให้มีรูปร่างและเวลาของฟิลด์คู่และฟิลด์คี่ให้ถูกต้อง

2.5รายละเอียดของลำดับการสแกน

ส่วนประกอบของขบวนพลัสสำหรับฟิลด์รวมเข้ากับอิควอไรส์ซิ่งพลัสแสดงจำนวนเส้นและการออกแบบพลัสตามข้างล่าง

ตารางที่ 2.2 รายละเอียดของลำดับการแสดง

FIRST FIELD(ODD FIELD)

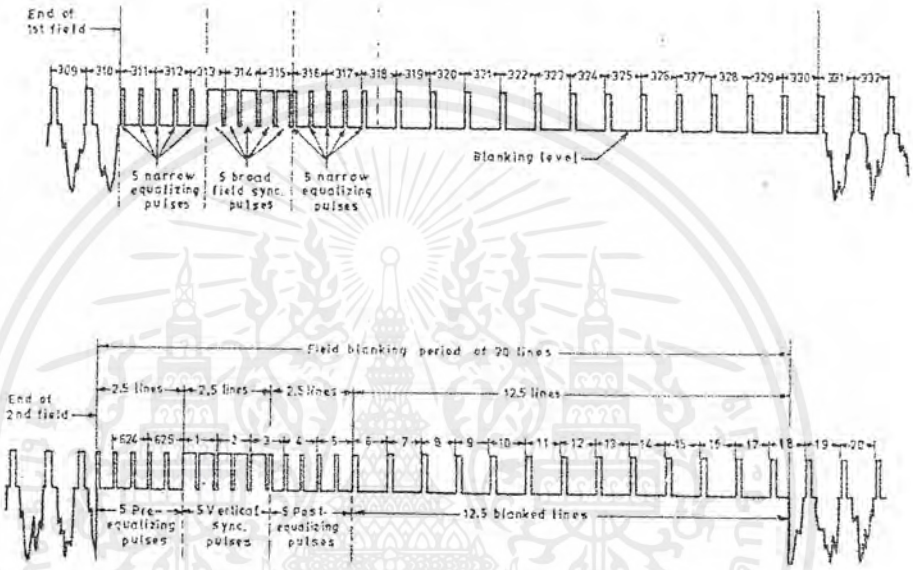
Line number: One to 1st-haft of 313th line(312.5 line)		
1,2 and 3rd 1 st-haft,line	2.5 lines	Vertcal syn pulses
3 rd 2 nd-haft,4and5	2.5 lines	Post-vertical sync equalizing pulses
6 to 7 ,and 18 th 1 st-haft	12.5 lines	Blanking retrace pulses
18 th 2 nd-haft to 310	292.5 lines	picture details
311 ,312 , and 313 th 1 st –haft	2.5 lines	Pre-vertical sysc-equalizing pulses for the 2 nd field
Total number of lines=312.5		

SECOND FIELD(EVEN FIELD)

Line number: 313 th 2 nd-haft to 625(312.5 line)		
313 2 nd-haft,314 and315	2.5 lines	Vertcal syn pulses
316,317 and318 1 st-haft	2.5 lines	Post-vertical sync equalizing pulses
318 th 2 nd-haft-to 330	12.5 lines	Blanking retrace pulses
331 to 1st-haft of 623 rd 310	292.5 lines	picture details

ตารางที่ 2.2 (ต่อ)

623 2 nd –haft,624 and625	2.5 lines	Pre-vertical sysc- equalizing pulses forthe 2 nd field
Total number of lines=312.5		
Total number of lines per Frame=625		



รูปที่2.10 ขบวนพลัส์ซิงซ์ฟิลด์ของระบบTVที่มี625เส้น

ขอบเขตคร่าวๆของจำนวนเส้น

กำลังของซิงก์พลัส์ฟันปลาทางแนวตั้งมีการหักเหของทางเดินfllyback อย่างไรก็ตามโดย fllyback ทั่วไป ไม่ได้มีตั้งแต่เริ่มซิงก์ทางแนวตั้งเพราะว่าซิงก์พลัส์อาจจะสร้างระดับแรงดันที่น้อยที่สุดที่ผ่านคาปาซิเตอร์จนกระทั้งทริกเกอร์เครื่องกำเนิดการสแกนดังนั้นมันจึงเหมือน fllyback ทางแนวตั้งแต่เริ่มกับขอบขาขึ้นของฟันปลาในลำดับที่สี่เวลาทั้งหมดที่ผ่านซิงก์ทางแนวตั้งก่อนการเริ่ม fllyback ทางแนวตั้งเป็น1.5เส้น พลัส์อิควอไรซซิ่งก่อนเริ่มขบวนซิงก์ทางแนวตั้ง

บทที่ 3
ภาษาวีเอชดีแอล (VHDL)

วีเอชดีแอล ย่อมาจากคำว่า VHSIC Hardware Description Language (VHSIC :Very High speed Integrated Circuit) เป็นภาษาโปรแกรมระดับสูง (high level language) ที่ใช้ในการออกแบบฮาร์แวร์ในระบบดิจิตอล ตัวของภาษาสามารถบรรยายพฤติกรรมการทำงานในรูปของลำดับชั้น (hierarchy) ได้ และสามารถที่จะเขียนได้หลายรูปแบบซึ่งจะกล่าวต่อไป จึงทำให้ภาษาวีเอชดีแอล เป็นเครื่องมือที่ใช้ออกแบบตั้งแต่ขั้นตอนบนสุด คือ แนวความคิดที่จะแก้ปัญหา ลงไปทีละขั้นจนถึงขั้นตอนของการสร้างวงจรจริง และตัวภาษาสามารถเปิดโอกาสให้วิศวกร ได้พัฒนาและจำลองการทำงานของรูปแบบฟังก์ชันการทำงานของวงจรอย่างสังเขป โดยที่ยังไม่ต้องไปคำนึงถึงรายละเอียดเกี่ยวกับโครงสร้างวงจรจริง นอกจากนั้นวีเอชดีแอลยังเป็นภาษาที่สนับสนุนลักษณะต่างๆ ของระบบดิจิตอลที่มีความซับซ้อนได้ทั้งหมด จึงเป็นภาษาที่น่าสนใจในการศึกษาและการนำไปใช้งาน

3.1 ประวัติความเป็นมาของภาษาวีเอชดีแอล

วิวัฒนาการของภาษาวีเอชดีแอล นั้นเริ่มต้นประมาณปี ค.ศ. 1981 โดยที่กระทรวงกลาโหมสหรัฐอเมริกา หรือ ดีโอดี (DoD: Department of Defense) มองเห็นว่าอุปกรณ์อิเล็กทรอนิกส์และคอมพิวเตอร์ที่ใช้ในกิจการทางทหาร เป็นอุปกรณ์ที่ได้รับการพัฒนามาเมื่อประมาณ 20 ปีก่อน เพราะเทคโนโลยีในขณะนั้นทำให้การพัฒนาอุปกรณ์อิเล็กทรอนิกส์เป็นไปอย่างล่าช้า ซึ่งเป็นสภาพที่ไม่อาจยอมรับได้ในปัจจุบัน เพราะเทคโนโลยีทางด้านไมโครอิเล็กทรอนิกส์ ได้รับการพัฒนาไปอย่างรวดเร็ว ดังที่จะเห็นได้ว่ามีวงจรดิจิตอลอิเล็กทรอนิกส์หลายวงจร ที่แต่เดิมถูกสร้างขึ้นมาจากเป็นชิ้น ถูกนำมาประกอบกันอยู่บนแผงวงจรไฟฟ้า ที่มีขนาดใหญ่ แต่ในปัจจุบันสามารถที่จะใช้เทคโนโลยีการออกแบบและผลิตวงจรรวมขนาดใหญ่มาก (VLSI : Very Large Scale Integration)รวมอุปกรณ์ต่างๆ เหล่านั้นให้อยู่บนชิ้นอุปกรณ์สารกึ่งตัวนำ ที่มีขนาดประมาณ 1-2 ตร.ซม. ได้ ซึ่งเป็นผลให้ประสิทธิภาพในการทำงานของวงจรสูงขึ้น (ความเร็วในการทำงานของวงจร) ตลอดจนความน่าเชื่อถือในการทำงาน และความคงทนต่อสภาพแวดล้อมสูง ขณะเดียวกันนั้นในวงการทหารได้มีการนำระบบคอมพิวเตอร์และอิเล็กทรอนิกส์ มาใช้ในระบบอาวุธอย่างแพร่หลาย ดังนั้นอุปกรณ์ที่มีใช้อยู่จึงไม่เหมาะสมกับเทคโนโลยีด้านอาวุธของประเทศคู่แข่งขัน การที่จะเปลี่ยนอุปกรณ์ใหม่เป็นสิ่งที่ต้องใช้งบประมาณมาก และก็จะประสบกับปัญหาเช่นเดิมคือ อุปกรณ์ใหม่ได้รับการพัฒนามานานแล้วเช่นกัน เพราะในขณะนั้นขั้นตอนของการออกแบบ การผลิต และการตรวจสอบวงจรต้นแบบ เป็นขบวนการที่ต้องใช้วิศวกร และเวลาสำหรับ

ดำเนินการมาก ฉะนั้นทางดีโอดีจึงตั้งโครงการขึ้นมาเพื่อศึกษา วิธีการที่จะช่วยพัฒนาวงจรอิเล็กทรอนิกส์ โดยเฉพาะอย่างยิ่งวงจรระบบดิจิตอล ให้สามารถนำไปผลิตได้เร็วขึ้น และโครงการดังกล่าวมีชื่อว่า "Very High Speed Integrated Circuits" หรือ วีเอชเอสไอซี (VHSIC) ในระยะแรกนั้นโครงการเป็นความลับทางด้านความมั่นคงของประเทศ และอยู่ในความดูแลควบคุมของ United states International Traffic and Arms Regulations หรือ ไอทีเออาร์ (ITAR) ในปี ค.ศ.1983 ตามคำแนะนำของทางคณะทำงาน ("woods Hole" workshop)ทาง ดีโอดีได้ออกความต้องการมาตรฐานของภาษาที่ใช้สำหรับบรรยายพฤติกรรมของวงจรหรือฮาร์ดแวร์ของระบบสำหรับโครงการวีเอซแอสไอซี ซึ่งมีสาระสำคัญพอสรุปได้ดังนี้

-ต้องเป็นภาษาที่นำไปเขียนรูปแบบระบบดิจิตอล และมีคุณสมบัติที่สามารถจะเข้าใจได้ทั้งคนและเครื่องโดยไม่ต้องมีการแปลหรือเปลี่ยนแปลงอีก

-สามารถนำไปใช้เป็นเอกสารประกอบโครงการได้

-ต้องเป็นภาษาที่เขียนขึ้นสำหรับใช้จำลองการทำงานของวงจร

ฉะนั้นภาษาดังกล่าวนี้จึงจัดเป็นภาษาโปรแกรมระดับสูง เช่นเดียวกับภาษาปาสคาล หรือภาษาซี ซึ่งในทางวิศวกรรมการออกแบบฮาร์แวร์เรียกว่า "Hardware Description Language" หรือ เอชดีแอล (HDL) เริ่มต้นโครงการดีโอดีได้มอบหมายให้บริษัทไอบีเอ็มและบริษัทเท็กซัลอินสตูลเมนท์และบริษัทอินเตอร์เมทริกซ์เป็นผู้ศึกษาและพัฒนา การดำเนินการได้กระทำไปอย่างต่อเนื่อง และได้ผลเป็นที่น่าพอใจ จนกระทั่งปี ค.ศ. 198 ทางไดทีเออาร์ได้ยกเลิกข้อจำกัดในการถ่ายทอดเทคโนโลยีทางทหาร ออกจากโครงการนี้ ดังนั้นภาษาวีเอชดีแอล จึงเริ่มเป็นที่รู้จักกันโดยทั่วไป จนกระทั่งทางไออีอีอี (IEEE) จึงได้รับการภาษานี้เข้ามาศึกษาและประมาณปี ค.ศ. 1987 ได้ยอมรับกำหนดมาตรฐานของภาษา โดยให้ชื่อว่า IEEE 1076-1987 และมีชื่อเรียกว่าวีเอชดีแอลมาตรฐานนี้ก็ได้รับการปรับปรุงจนปัจจุบัน ได้ชื่อว่า IEEE 1076-1993 หรือ วีเอชดีแอล 1993 การที่ทางดีโอดีในขณะนั้น เป็นลูกค้ารายใหญ่ของอุตสาหกรรมอิเล็กทรอนิกส์และคอมพิวเตอร์ จึงมีผู้รับโครงการต่างๆ จากดีโอดี ไปดำเนินการด้านวิจัยและพัฒนามาก เพื่อที่จะให้เป็นมาตรฐานเดียวกันหมด ทางดีโอดีจึงกำหนดว่า ในการส่งโครงการนั้นจะต้องเขียนในรูปของภาษาวีเอชดีแอลเท่านั้น ซึ่งทำให้เกิดข้อดีต่อดีโอดีเองเป็นมาตรฐานเดียวกัน สามารถนำไปจำลองกับเครื่องคอมพิวเตอร์ได้หลาย ๆ ระบบ

3.2 ส่วนประกอบต่าง ของภาษาวีเอชดีแอล

ในการเขียนรูปแบบบรรยายระบบดิจิตอลในมุมมองของการออกแบบลักษณะบนลงล่าง จะต้องทำความเข้าใจในเรื่องของโครงสร้างและส่วนประกอบต่างๆ ของรูปแบบภาษาวีเอชดีแอล เสียก่อน ซึ่งส่วนประกอบที่สำคัญและเป็นพื้นฐานของการเขียนมี 4 หน่วย คือ

- หน่วยการออกแบบเอนทีติ (Entity Design Unit)
  หน่วยการออกแบบสถาปัตยกรรม (Architecture Design Unit)
- หน่วยการออกแบบแพ็กเก็ต (Package Design Unit)
- หน่วยการออกแบบโครงแบบ (Configuration Design Unit)

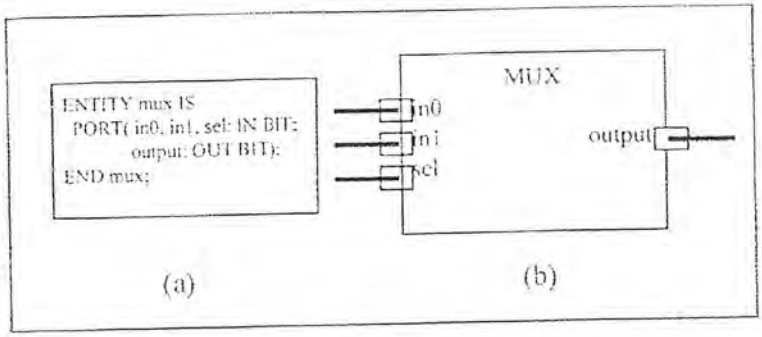
3.2.1หน่วยการออกแบบเอนทิตี

หน่วยการออกแบบนี้เป็นส่วนที่ใช้สำหรับติดต่อระหว่างโลกภายนอกกับรูปแบบที่เขียนขึ้น ที่เรียกว่า หน่วยการออกแบบเอนทิตี ในส่วนนี้ใช้กำหนดจุดเชื่อต่อ ของรูปแบบ กำหนดทิศทางการไหลสัญญาณ และประเภทของค่าที่สามารถกำหนดให้กับสัญญาณตามจุดต่าง ๆ ของข้อมูลที่ไหลผ่านจุดต่อเหล่านั้น รูปที่ 3.1 แสดงให้เห็นโครงสร้างอย่างง่าย ๆ ของ หน่วยการออกแบบเอนทิตี

```
ENTITY component name IS
  input and output ports
  Physical and other parameters
END [component_name];
```

รูปที่ 3.1 แสดงโครงสร้างโดยทั่วไปของหน่วยการออกแบบเอนทิตี

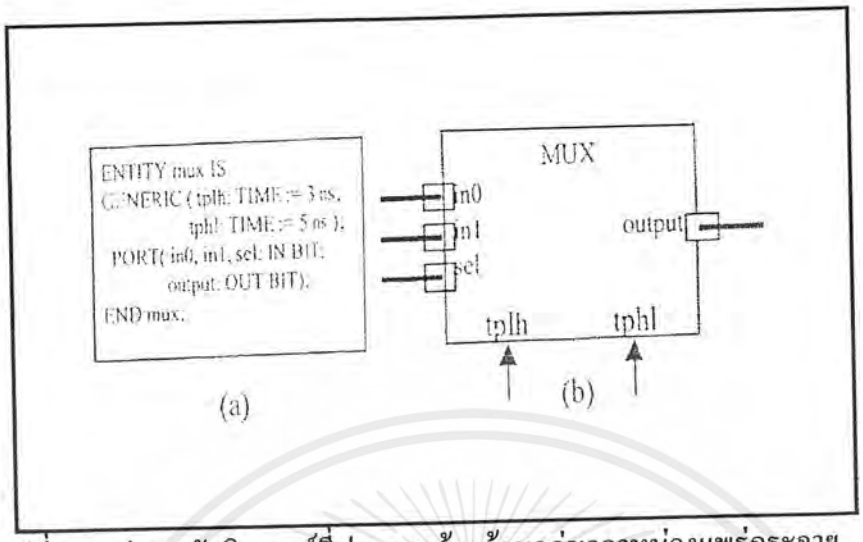
ส่วนนี้จะขึ้นต้นด้วยคำ ENTITY และ IS ระหว่างคำทั้งสองเป็นส่วนสำหรับชื่อของรูปแบบที่ต้องการจะเขียน (component_name) สำหรับการตั้งชื่อนั้นต้องเป็นไปตามกฎเกณฑ์ของภาษาหลังจากนั้นจะตามด้วยส่วนที่ใช้กำหนดช่องทางเข้าและออกของข้อมูล (input-output) รวมทั้งพารามิเตอร์อื่น ๆ ส่วนนี้เรียกว่าส่วนหัว (entity header) และที่สำคัญคือ หน่วยการออกแบบเอนทิตีจะต้องปิดด้วยคำว่า END และเครื่องหมายอัฒภาคเสมอ (;)



รูปที่ 3.2 แสดงรูปแบบของมัลติเพล็กซ์ (a) หน่วยการออกแบบเอนทิตีในรูปของวีเอชดีแอล (b) มุมมองของตัวเชื่อมประสาน (interfacing)

ในรูปที่ 3.2 เป็นหน่วยการออกแบบเอนทิตี ทีบรรยายอุปกรณ์ที่มีชื่อว่ามัลติเพล็กซ์ หรือ MUX ในส่วนหัวของเอนทิตี มีการกำหนดจุดต่อ 4 จุดภายใต้ชุดคำสั่ง PORT โดยที่ 3 จุดแรกเป็นจุดให้ข้อมูลไหลผ่านเข้า ได้แก่ Ino,In1,sel ซึ่งกำหนดด้วยทิศทางการติดต่อกับภายนอกเป็นการไหลเข้าของข้อมูล (IN) ที่แสดงด้วยรูปสี่เหลี่ยมโปร่งในรูปที่ 3.2 ส่วนจุดเอาต์พุตเป็นจุดให้ข้อมูลไหลออก ซึ่งกำหนดด้วยทิศทางการติดต่อกับภายนอกเป็นการไหลออก (OUT) ที่แสดงด้วยรูปสี่เหลี่ยมทึบในรูปที่ 3.2 ส่วนประกอบของข้อมูลที่จะไหล เข้าและออก นั้นเป็นประเภท BIT ที่สามารถมีค่าได้เพียงสองค่าคือ '0' และ '1' เท่านั้น

นอกจากนั้นผู้ออกแบบยังสามารถกำหนดค่าพารามิเตอร์ทางฟิสิกส์ที่เป็นข้อมูลเพิ่มเติมอื่น ๆ ลงในส่วนหัวของเอนทิตีได้อีก เช่น ข้อมูลเกี่ยวกับความเร็วในการทำงานของอุปกรณ์ อันได้แก่ ค่าเวลาหน่วงแพร่กระจาย (Propagation delay time) พารามิเตอร์เหล่านี้ เรียกว่า เจนเนริก (Generic) ที่กำหนดด้วยคำสั่ง GENERIC จากตัวอย่างในรูปที่ 3.3



รูปที่ 3.3 รูปแบบมัลติเพลกซ์ที่ประกอบด้วยข้อมูลค่าเวลาหน่วงแพร่กระจาย
(a) หน่วยการออกแบบเอนทิตีในรูปของวีเอชดีแอล (b) มุมมองของตัวเชื่อม
(b) ประสาน

ในบางกรณีสามารถที่จะใช้ภาษาวีเอชดีแอล สร้างรูปแบบที่ปราศจากช่องทางไหล เข้าและออกของข้อมูล ได้ ซึ่งส่วนใหญ่จะพบในการสร้างรูปแบบ สำหรับตรวจสอบการทำงานของอีกรูปแบบหนึ่ง คือ วีเอชดีแอลสำหรับการทดสอบเปรียบเทียบ (Test bench)1

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

รูปที่ 3.4 หน่วยการออกแบบเอนทิตีที่ไม่มีการกำหนดช่องทางที่ต่อกับภายนอก

3.2.2 หน่วยการออกแบบสถาปัตยกรรม

คือส่วนที่ใช้เขียนบรรยายพฤติกรรมของรูปแบบ ในมุมมองของการจำลองการทำงานพฤติกรรมต่าง ๆ ที่บรรยายในส่วนนี้ขึ้นอยู่กับข้อมูลที่ผ่านเข้าและออก ตรงช่องทางตลอดจนพารามิเตอร์ต่าง ๆ ที่กำหนดใน หน่วยการออกแบบเอนทิติ รูปที่ 3.5 แสดงให้เห็นถึงโครงสร้างอย่างง่าย ๆ ของหน่วยการออกแบบสถาปัตยกรรม

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

รูปที่3.5แสดงโครงสร้างโดยทั่วไปของหน่วยการออกแบบสถาปัตยกรรม

ส่วนของหน่วยการออกแบบสถาปัตยกรรม เริ่มต้นด้วยคำ ARCHITECTURE และตามด้วยชื่อ (identifier) สิ่งที่ต้องกำหนดลงไปได้แก่ สิ่งที่แสดงให้เห็นว่า ARCHITECTURE นั้นใช้บรรยายหน่วยการออกแบบเอนทิตีใด ๆ (OF<entity design unit> IS) ส่วนที่อยู่ระหว่าง ARCGUTCTURE และ BEGIN เป็นพื้นที่ส่วนประกาศหน่วยของสถาปัตยกรรมกำหนด (architecture declarative area) ที่เป็นเพียงส่วนเผื่อเลือก (option) ในบริเวณนี้สามารถใช้เขียนประกาศกำหนดค่าต่าง ๆ ที่จะนำไปใช้ภายสถาปัตยกรรมนั้นได้ อาทิเช่นประเภท (type) ต่างๆ (ตัวอย่าง เช่น bit, bit_vector), สัญญาณ (signal), ตัวคงที่ (constant), โปรแกรมย่อย (ได้แก่ function และ procedure) และอุปกรณ์(component) ส่วนที่ใช้บรรยายความสัมพันธ์ระหว่างข้อมูลที่ไหลเข้า และไหลออกของรูปแบบ(สัญญาณที่กำหนดในชุดคำสั่ง PORT) นั้นจะถูกบรรยายในบริเวณเนื้อที่ระหว่างคำว่า BEGIN กับ END ของหน่วยการออกแบบสถาปัตยกรรม และนอกจากนั้นชุดคำสั่งทุกคำสั่งที่อยู่ภายในบริเวณนี้จะเป็นชุดคำสั่งแบบแข่งขนาน (concurrent statement) เท่านั้น หน่วยการออกแบบสถาปัตยกรรมจะต้องปิดท้ายด้วยคำสั่ง END และชื่อของสถาปัตยกรรมนั้น ๆ ที่เป็นส่วนเผื่อเลือกโดยทั่วไปการเขียนรูปแบบระบบดิจิตอลด้วยภาษาวีเอชดีแอล สามารถเขียนได้ในลักษณะต่างๆ ดังนี้

ประเภทการไหลของข้อมูล (Dataflow description)

ประเภทพฤติกรรม(Behavioral description)

ประเภทโครงสร้าง(Structure description)

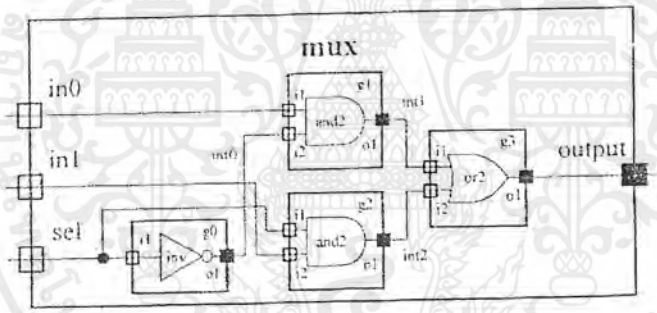
ประเภทผสม(Mixed model description)

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

รูปที่ 3.6 แสดงหน่วยการออกแบบสถาปัตยกรรมของมัลติเพลกซ์ตามฟังก์ชันบูลีน

output =(sel . in0) + (sel.in1)

รูปที่ 3.6 ส่วนที่บรรยายความสัมพันธ์ระหว่างข้อมูลที่ไหลเข้า (in0,in 1) กับข้อมูลที่ไหลออก (output) ประกอบด้วยชุดคำสั่งแบบแข่งขนานเพียงชุดเดียว ซึ่งเขียนเป็นประเภทการไหลของข้อมูลของมัลติเพลกซ์ หรือ ระดับการถ่ายโอนข้อมูลระหว่างเรจิสเตอร์ RTL : Register Transfer Level)



รูปที่ 3.7 แสดงโครงสร้างภายในสถาปัตยกรรมของมัลติเพลกซ์

รูปที่ 3.7 เป็นหน่วยการออกแบบสถาปัตยกรรมของมัลติเพลกซ์ประเภทโครงสร้าง โดยใช้อินเวอร์เตอร์ (inv ที่ตำแหน่ง g0), แอนด์เกต 2 อินพุตจำนวน 2 ตัว (and 2 ที่ตำแหน่ง g1 และ g 2) และออร์เกต 2 อินพุต (or2 ที่ตำแหน่ง g3) มาสร้างตามฟังก์ชันบูลีนของรูปที่ 3.6

```
ARCHITECTURE struc OF mux IS
  COMPONENT inv
  PORT ( i1 : IN BIT ; o1 : OUT BIT );
  COMPONENT and2
  PORT ( i1,i2 : IN BIT ; o3 : OUT BIT );
  COMPONENT or2
  PORT ( i1,i2 : IN BIT ; o1 : OUT BIT );
 END COMPONENT;
 SIGNAL into,int1,int2 : BIT ;
BEGIN
  g0 : inv  PORT MAP ( i1 => sel, o1 => int0);
  g1 : and2  PORT MAP ( i1 => in0, i2 => int0,ol => int1);
  g0 : inv  PORT MAP ( i1 => sel, o1 => int0);
END struc;
```

รูปที่ 3.8 หน่วยการออกแบบสถาปัตยกรรมของมัลติเพลกซ์ประเภทโครงสร้าง

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

รูปที่ 3.9 หน่วยการออกแบบสถาปัตยกรรมของมัลติเพลกซ์ประเภทพฤติกรรม

ไม่ว่าเขียนบรรยายส่วนของสถาปัตยกรรมของมับติเพลกซ์ในลักษณะของ ประเภทพฤติกรรม ประเภทการไหลของข้อมูล ประเภทโครงสร้างหรือประเภทผสมที่นำเอาแต่ละประเภทมา

เขียนไว้ส่วนของสถาปัตยกรรม ก็ตามต่างก็มีพฤติกรรมเดียวกัน และจะให้ผลลัพธ์จากการจำลองการทำงานที่เหมือนกัน ซึ่งนี้ก็เป็นข้อดีของภาษาวีเอชดีแอล

3.2.3 หน่วยการออกแบบแพ็กเก็ต

ข้อมูลต่าง ๆ ตลอดจนโปรแกรมย่อย ที่เป็นประโยชน์ ต่อการเขียนรูปแบบบรรยายระบบดิจิตอล สามารถเก็บไว้ในส่วนของแพ็กแก็ตได้ และข้อมูลเหล่านี้สามารถเรียกไปใช้ได้โดย หน่วยกาออกแบบเอนทิตี หน่วยการออกแบบสถาปัตยกรรม หรือ จากหน่วยการออกแบบแพ็กแก็ตอื่น ๆ นอกจากนั้นสิ่งที่นิยมทำกันมาคือรูปแบบมาตรฐานต่างๆ เช่น อุปกรณ์มาตรฐาน (เช่น ไอซีตระกูล 74 XX เป็นต้น) จะถูกเก็บไว้ในแพ็กแก็ต ที่ทุกคนสามารถเข้าถึง โดยปกติแล้ว แพ็กแก็ตจะแบ่งออกเป็น 2 ส่วนคือ การประกาศแพ็กแก็ต (Package deckaratuib) และส่วนขอบอดี้แพ็กแก็ต (Padkage body) เนื่องจาก แพ็กแก็ตถูกสร้างขึ้นเป็นส่วนแยกต่างหากออกจากรูปแบบที่กำลังเขียนอยู่ ฉะนั้นการที่นำแพ็กแก็ตไปใช้นั้นจะต้องมีการเชื่อมโยงหรืออ้างอิงเสียก่อน ซึ่งในภาษาวีเอชดีแอลสามารถกระทำได้ด้วยชุดคำสั่ง USE

PACKAGE DECLARATION

ส่วนที่มีความสำคัญที่สุดของแพ็กแก็ต (ถ้ามองในแง่ของการนำไปใช้จากภายนอก) ได้แก่ ส่วนการประกาศแพ็กแก็ต เพราะจะเป็นส่วนที่กำหนดชื่อ ของสิ่งที่ประกาศอยู่ภายในแพ็กแก็ตสำหรับนำไปใช้ภายนอกตัวของแพ็กแก็ตเอง สิ่งใด ๆ ถูกประกาศในส่วนของ ส่วนบอดี้แพ็กแก็ตแต่ไม่ถูกประกาศ ในส่วนการประกาศแพ็กแก็ต จะไม่สามารถถูกนำค่า และ พฤติกรรมไปใช้ส่วนนอกได้ ซึ่งสามารถเปรียบเทียบได้กับสิ่งที่ประกาศไว้ในส่วนของการประกาศเอนทิตีคือ จุดเชื่อมต่อ หรือ พอร์ต ที่มีหน้าติดต่อกับโลกภายนอก ฉะนั้นโดยทั่วไปแล้ว แพ็กแก็ต สามารถสร้างขึ้นได้โดยไม่จำเป็นต้องมีส่วนบอดี้ และยังสามารถถูกนำไปใช้จากรูปแบบภายนอกได้เช่น ใช้สำหรับประกาศ ชนิด (type) หรือ สัญญาณ เช่นเดียวกันกับ ส่วนบอดี้แพ็กแก็ต ที่ไม่จำเป็นต้องมี ส่วนขอการประกาศแพ็กแก็ต แต่แพ็กแก็ต นั้นจะไม่สามารถถูกนำไปใช้จากรูปแบบอื่นได้

```
PACKAGE packag_name IS
  Package_dectaraive_part
END  package_name;
```

รูปที่ 3.10 แสดงโครงสร้างโดยทั่วไปของส่วนการประกาศแพ็กแก็ต

PACKAGE BODY

โครงสร้างที่ประกอบด้วยคำสั่งต่าง ๆ ในรูปของคำสั่งลำดับ ที่ใช้บรรยายฟังก์ชันการทำงานของโปรแกรมย่อย (Subprogram) ทั้งหลายที่ชื่อของโปรแกรมย่อมนั้น ๆ ที่ถูกประกาศไปในส่วนของการประกาศแพ็กแก็ต แล้วจะถูกเก็บไว้ในส่วนบอดี้แพ็กแก็ต ทั้งรวมทั้ง แต่ถูกกำหนดค่าในส่วนของบอดี้แพ็กแก็ต) ฉะนั้นส่วนบอดี้แพ็กแก็ต จึงไม่จำเป็นต้องมี ถ้าในส่วนของการประกาศแพ็กแก็ต ไม่มีการประกาศชื่อ ที่เป็นโปรแกรมย่อย หรือค่าคงที่ การเขียนบอดี้แพ็กแก็ตนั้นเป็นไปตามกฎเกณฑ์ที่แสดงในรูปที่ 3.11

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

รูปที่ 3.11 โครงสร้างของบอดี้แพ็กแก็ต

3.2.4 หน่วยการออกแบบโครงแบบ

ดังที่ทราบกันแล้วว่ารูแบบหนึ่งของระบบดิจิตอลไม่ว่าจะเป็นอะไร จะมีหน่วยการออกแบบเอนทีตีได้ เพียงหนึ่งเดียวเท่านั้น แต่ในขณะที่หน่วยการออกแบบเอนทิตี หนึ่งหน่วยนี้อาจจะมีสถาปัตยกรรม ที่เป็นหน่วยรองได้หลายหน่วย ดังนั้นจะต้องมี หน่วยการออกแบบโครงแบบมาเพื่อกำหนดการใช้โครงแบบ (Configuration) ประกอบเอนทีตีกับหน่วยการออกแบบสถาปัตยกรรมหน่วยไหนเข้าด้วยกัน

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

รูปที่ 3.12 โครงสร้างโดยทั่วไปของหน่วยการออกแบบโครงแบบ

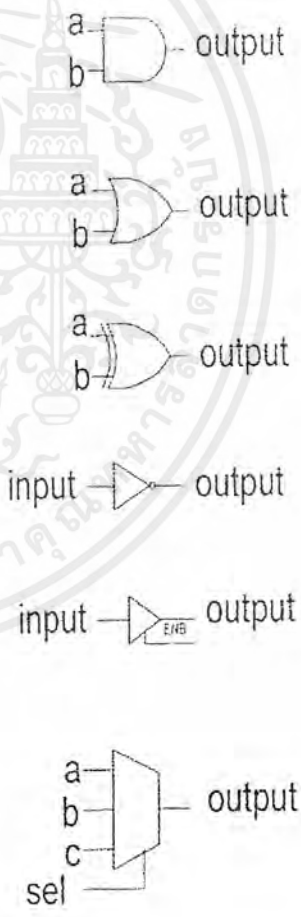
3.3 ภาษาวีเอชดีแอลเพื่อการสังเคราะห์

ภาษาวีเอชดีแอล ซึ่งในบางรูปแบบการเขียนไม่สามารถที่จะนำไปสังเคราะห์ได้ทั้งหมด ดังนั้นถ้าต้องการเขียนเพื่อนำไปสังเคราะห์ ควรหลีกเลี่ยงรูปแบบต่างๆ ที่ไม่สามารถนำไปสังเคราะห์ได้ ดังนั้นในหัวข้อนี้จะแสดงตัวอย่างของการเขียนโมเดลในรูปแบบต่างๆ ที่สามารถนำไปสังเคราะห์ ซึ่งยึดหลักการเขียนตาม ViewSynthesis User's Guide ของโปรแกรม ซึ่งเป็นโปรแกรมที่ใช้ในสังเคราะห์วงจรทั้งหมดในการออกแบบทีวีแพดเทอร์เจนเนเรเตอร์

3.3.1 ตัวอย่างรูปแบบการเขียนเกตพื้นฐาน

```
SIGNAL a, b, c, d, input, output : vlbit_ld(3 DIWBTI 0);
SIGNAL sel  :  vlbit_1d (1 DOWNTO 0);
SIGNAL enb  :  vlbit;
```

```
....

Output <= a AND b;
....
Output <= a OR b;
....

Output <= a XOR b;
....
Output <= a NOT b;
....
Output <=input WHEN enb='1'ELSE"ZZ";
....

WITH sel SELECT
     Output <=  a WHEN "00" ELSE
                b WHEN  '01" ELSE
                c;
```

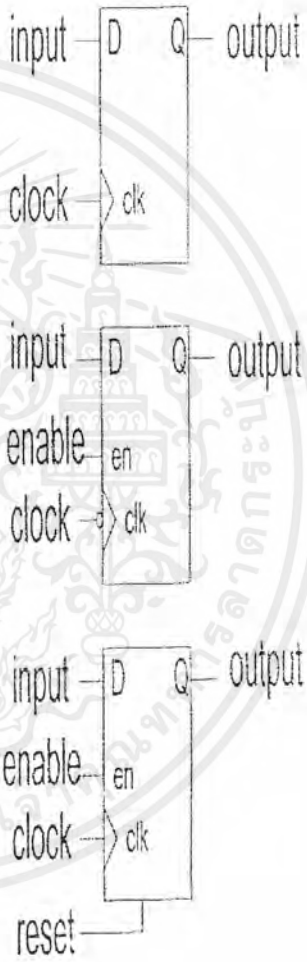


รูปที่ 3.13 ตัวอย่างรูปแบบการเขียนเกตพื้นฐาน

3.3.2ตัวอย่างรูปแบบการเขียนฟลิปฟลอฟพื้นฐาน

```
SIGNAL  input,output : vlbit_ld(3 downto 0);
SIGNAL clock,enable,reset : vlbit;
```

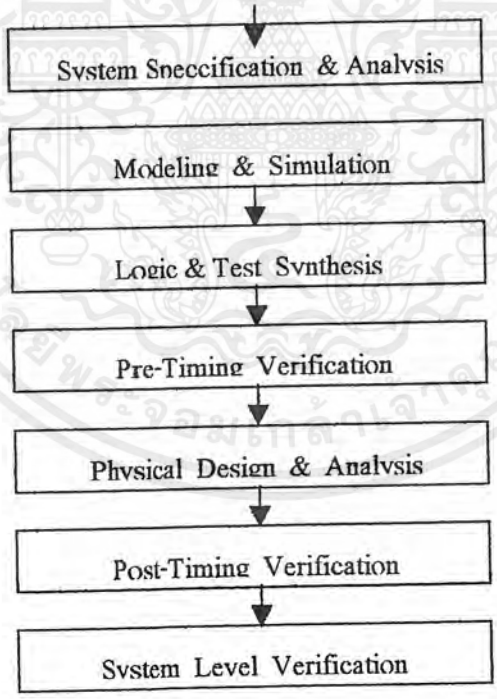
```
....
PROCESS BEGIN
        WAIT UNTIL PRISING(clock);
                Output <= input;
END PROCESS;
....

PROCESS BEGIN
        WAIT UNTIL PFALLING(clock);
                Output <= input;
END PROCESS;
....

PROCESS BEGIN
 WAIT UNTIL PRISING(clock)OR(reset='1');
        IF (reset ='1')THEN output <= "00";
        ELSIF(enble='1')THENoutput<= input;
        END IF;
END PROCESS;
```



รูปที่ 3.14 ตัวอย่างรูปแบบการเขียนเกตพื้นฐาน

3.4 การออกแบบจากบนลงล่าง

ในการพัฒนาวงจรรวมดิจิตอลขนาดใหญ่ที่มีความซับซ้อน เช่น วงจรรวม (ASIC : Application specific Integrated Circuit) วิศวกรหรือผู้ออกแบบมักจะมองการออกแบบให้อยู่ในรูปของของบล็อกไดอะแกรมเสียก่อน ก่อนที่จะวิเคราะห์ให้ลึกถึงรายละเอียดต่อไป ซึ่งภาษาวีเอชดีแอล(VHDL)นั้นอนุญาตให้อธิบายการทำงานของแต่ละบล็อก และวิเคราะห์การทำงาน แก้ไขและปรับปรุงการทำงานจากผลที่วิเคราะห์เพื่อให้ได้การทำงานตามที่ต้องการ และเพิ่มเติมในรายละเอียดที่ละขั้นนี้คือหลักการออกแบบจากบนลงล่าง (TOP-DOWN Design) ถ้าทดลองเปรียบเทียบกับการออกแบบจากล่างขึ้นบน (Bottom-Up Design) จะเห็นได้ว่าการออกแบบจากล่างขึ้นบนจะใช้เวลาการออกแบบมากกว่า 90% เพราะเป็นการวาดวงจรด้วยอุปกรณ์ต่างๆ (Schematic capture) ที่ประกอบกันเข้าเป็นวงจรที่ต้องการออกแบบ จำลองการทำงาน ตรวจสอบความถูกต้อง ซึ่งใช้เวลามาก และถ้าวงจรที่ต้องการออกแบบมีความซับซ้อนก็จะเป็นเรื่องที่ยากมากให้การออกแบบในลักษณะนี้ ดังนั้นการใช้ภาษาวีเอชดีแอล(VHDL)กับหลักการออกแบบจากบนลงล่างจึงเป็นทางออกให้กับวิศวกรออกแบบที่จะสามารถออกแบบและพัฒนาวงจรที่มีซับซ้อนได้มากขึ้น และช่วยลดเวลาและค่าใช้จ่ายในการออกแบบ



รูปที่ 3.15 ขั้นตอนการออกแบบจากบนลงล่าง

จากรูปที่ 3.15 แสดงให้ถึงขั้นตอนของการออกแบบจากบนลงล่าง
ทั้งนี้ในทางปฏิบัติอาจจะมีข้อแตกต่างไปจากนี้บ้างเล็กน้อย ก็เนื่องจากขั้นตอนของการผลิต (Implementation) สามารถกระทำได้หลายๆ เทคโนโลยี เช่น พีแอลดี (PLD : Programmable Logic Device) อันได้แก่ พีแอลดี
(PLD : Programmable Logic Device) ,เอฟพีจีเอ (FPGA : Filed Programmable Gate Array),ซีพีแอลดี (CPLD : Cell Programmable Logic Device) เป็นต้น นอกนั้นยังมี เซมิคัสตัมไอซี (Semicustom IC) ได้แก่ เกตอะเรย์ (Gate array), เซลล์มาตรฐาน (Standard Cell) ขั้นตอนของการออกแบบจากบนลงล่างมีรายละเอียดดังนี้

1.ขั้นตอนการสร้างข้อกำหนดของความต้องการ และวิเคราะห์ระบบ เพื่อหาแนวความคิดและหลักการ (Idea and Concept) ในการแก้ปัญหา

2.ขั้นตอนการเขียนรูปแบบของระบบที่ต้องการออกแบบโดยใช้ภาษาวีเอชดีแอล หรือ ภาษาเอชดีแอลอื่นๆ สำหรับบรรยายพฤติกรรมการทำงาน พร้อมทั้งจำลองการทำงาน เพื่อเปรียบเทียบและตรวจสอบความถูกต้องกับข้อกำหนด

3.หลังจากที่ได้หลักการขั้นต้นพร้อมกับแนวความคิดที่ผ่านการตรวจสอบแล้ว หลักการนี้จะถูกเพิ่มเติมในรายละเอียดลงมาเป็นลำดับขั้นที่สอง จนกระทั่งอยู่ในระดับที่จะนำไปผลิตวงจร หรือสังเคราะห์ ในขั้นตอนนี้เองเทคโนโลยีที่จะมารองรับวงจรออกแบบจะถูกกำหนดขึ้น และระบบช่วยการออกแบบจะสังเคราะห์วงจรที่ได้จากรูปแบบที่เขียนขึ้น ให้อยู่ในรูปของวงจรที่ประกอบด้วยอุปกรณ์อิเล็กทรอนิกส์ หรือวงจรในระดับเกต และการเชื่อมต่อระหว่างกันของอุปกรณ์เหล่านั้นหรือไม่ก็อยู่ในรูปของเน็ทลิสท์ (Netlist) ที่สามารถนำไปผลิตลงบนอุปกรณ์อื่นได้

4. หลังจากการสังเคราะห์วงจรให้อยู่ในรูประดับเกตหรือเน็ทลิสท์แล้ว ข้อมูลที่ได้จากผู้ผลิตอุปกรณ์วงจรนั้น นอกจากจะเป็นข้อมูลสำหรับจำลองการทำงาน ในเรื่องของความถูกต้องของฟังก์ชันแล้ว ยังมีข้อมูลที่เกี่ยวกับเวลาด้วย ซึ่งเป็นความจริงที่ว่า อุปกรณ์ทางอิเล็กทรอนิกส์ทุกชิ้นจะมีเวลาหน่วงของการแพร่กระจาย (Propagation delay time) เสมอ ถึงแม้ว่าจะเป็นเวลาที่น้อยมากในระดับ นาโนวินาที
(10^{-9} นาที) แต่ถ้าภายในวงจรหนึ่งประกอบด้วยเกตของฟังก์ชันต่าง ๆ จำนวน 10,000 เกตขึ้นไป เวลาดังกล่าวนี้จะสะสมกันมากขึ้น จนอาจจะทำให้การทำงานของวงจรรวมทั้งหมดผิดไป หรือไม่สามารถทำงานในย่านความถี่สัญญาณนาฬิกาที่สูงได้

5.ขั้นตอนของการผลิตเป็นวงจรจริง (Technology and device mapping)

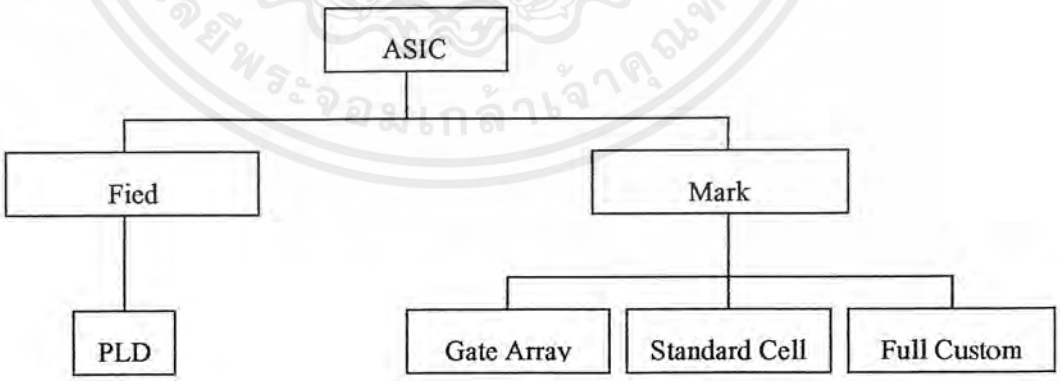
โดยนำข้อมูลที่ได้จากการสังเคราะห์มาผลิต ซึ่งอาจจะอยู่ในรูปของแผงวงจรไฟฟ้า ที่ประกอบด้วยอุปกรณ์หลายๆ ชิ้นหรืออยู่ในรูปของวงจรรวม (ASIC)

6. หลังจากที่ได้วงจรจริงมาแล้ว ยังต้องมีความจำเป็นที่ต้องตรวจสอบการทำงานที่คำนึงถึงเวลาด้วย เพื่อความถูกต้องของวงจรครั้งสุดท้ายก่อนที่จะนำไปรวมเข้ากับอุปกรณ์อื่นๆ ให้เป็นระบบดิจิตอล เพราะในขั้นตอนนี้วงจรที่ออกแบบ จะประกอบด้วยอินพุต และเอาต์พุต แพด (Pad) ซึ่งเป็นจุดต่อสำหรับรับและส่งสัญญาณกับภายนอก

7.หลังจากที่นำวงจรที่ออกแบบรวมเข้ากับอุปกรณ์อื่นๆ ให้เป็นระบบดิจิตอลแล้วนั้น จะต้องทดสอบการทำงานรวมทั้งระบบร่วมกับอุปกรณ์อื่นๆ อีกครั้ง เป็นการควบคุมคุณภาพของผลิตภัณฑ์

เอฟพีจีเอ(FPGA)

ความก้าวหน้าของอุตสาหกรรมอิเลคทรอนิกส์ปัจจุบันทำให้เกิดการพัฒนาความสามารถของอุปกรณ์ต่างๆ มากมายซึ่งทำให้เกิดการลดค่าใช้จ่าย การสิ้นเปลืองพลังงานและขนาด ในขณะเดียวกันก็มีการเพิ่มประสิทธิภาพและระดับความเชื่อถือได้ของวงจรที่สูงขึ้นเห็นได้ชัดจากเทคโนโลยีไมโครโพรเซสเซอร์และหน่วยความจำปัจจุบัน ทุกๆครั้งที่มีการพัฒนาขึ้นทำให้เกิดช่องว่างวงจรรวม และไอซีมาตราฐานมากขึ้น ในการพัฒนาเพิ่มความหนาแน่นและจำนวนฟังก์ชันลอจิกที่เหมาะสม นักออกแบบอุปกรณ์ทางด้านดิจิตอลได้พิจารณาถึงการผลิตวงจรรวม (ASIC: Application Specific Integrated Circuit) ซึ่งวงจรรวม จะแบ่งตามการสร้างออกเป็น 2 กลุ่ม คือ Field programmable และ Mask programmable ดังแสดงในรูปที่ 1



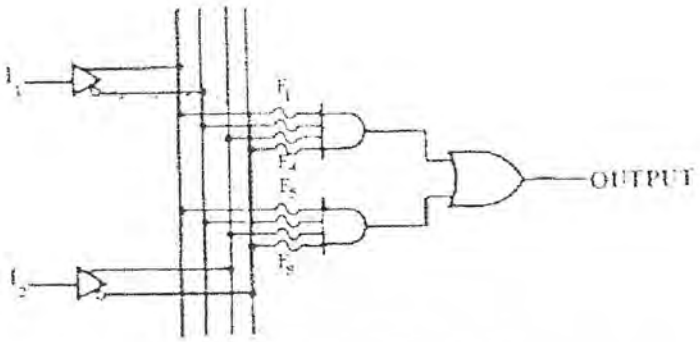
รูปที่ 3.16ผังการแบ่งกลุ่มของวงจรรวม ASIC

3.5.Field Programmable

อุปกรณ์วงจรรวม ASIC แบบ Field Programmable มีอยู่อย่างมากมายหลายชนิด แต่มีลักษณะการสร้างหรือกำหนดการทำงานของวงจรที่เหมือนกัน กล่าวคือ ผู้ใช้สามารถออกแบบและสร้างวงจรที่ต้องการใช้ลงในตัวอุปกรณ์ได้เองไม่ต้องไปโรงงานเพื่อผลิต โดยเฉพาะอย่างยิ่งในปัจจุบันนี้มีเครื่องช่วยการออกแบบ และสร้างวงจรรวมกับไมโครคอมพิวเตอร์ที่มีความสามารถสูงในการพัฒนาตั้งแต่ขั้นตอนการออกแบบ การจำลองการทำงาน จนถึงการจักสร้างวงจรลงในอุปกรณ์ รวมทั้งอุปกรณ์ Field Programmable เหล่านี้สามารถหาซื้อได้ง่ายทำให้การสร้างวงจรอิเล็กทรอนิกส์จนถึงระบบไมโครโปรเซสเซอร์หันมาใช้อุปกรณ์จำพวกนี้ เป็นอุปกรณ์ประกอบในวงจรแทนอุปกรณ์ย่อยๆแยกชิ้น(Discrete component)มาก

3.5.1พีแอลดี (PLD : Programmable Logic Device)

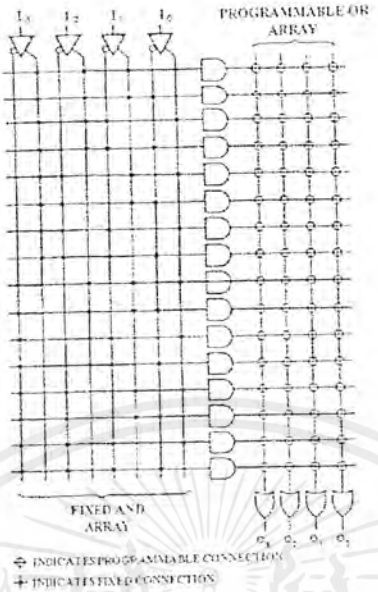
ภายในอุปกรณ์พีแอลดีถูกเตรียมเป็นวงจรพื้นฐานทางลอจิกต่อกันอยู่เป็นกลุ่มมีทั้งวงจรคอมบิเนชั่น(Combination) และซีแควนเชียล(Sequential) ซึ่งมีส่วนประกอบเป็นวงจรภายในเทคโนโลยีที่ใช้สร้างพีแอลดีมีทั้ง ทีที แอล(TTL) อีซีแอว(ECL) และ ซีมอส(CMOS) ตามความเหมาะสมของระบบ อุปกรณ์พีแอลดีทุกชนิดมีหลักการพื้นฐานของวงจรภายในที่เหมือนกันโดยมีวงจรหลักเป็นวงจรคอมบิเนชั่นที่ให้ผลเป็นผลคูณร่วมบวก (Sum of Product) ประกอบไปด้วยชุดแอนด์เกตการโปรแกรมคือ การเลือกว่าจะให้มีการต่ออินพุตภายในของแอนด์เกตกับสัญญาณอินพุตใดบ้างซึ่งมีทั้งสัญญาณภายนอกและสัญญาณป้อนกลับจากเอาต์พุตภายในเอง การติดต่ออินพุต ของออเกตกับเอาต์พุต ของแอนด์เกต วิธีการเลือกหรือการโปรแกรมทางกายภาพ อินพุตต่างๆของอุปกรณ์ทุกตัวจะถูกต่อเข้ากับฟิวส์เข้ากับแหล่งสัญญาณ ซึ่งถ้าไม่ต้องการใช้สัญญาณใดก็จะตัดฟิวส์ทำให้สามารถโปรแกรมได้ครั้งเดียว อุปกรณ์พีแอลดีบางชนิดใช้มอสทรานซิสเตอร์แทนฟิวส์ทำให้สามารถโปรแกรมได้ด้วยกระแสไฟฟ้า และสามารถลบและโปรแกรมได้ใหม่เข้าไปอีก



รูปที่ 3.17 แสดงอุปกรณ์พื้นฐานของอุปกรณ์พีแอลดี ซึ่งอยู่ในรูปผลคูณร่วมบวก

3.5.2 พรอม (PROM :Programmable Read Only Memory)

พรอมคือหน่วยความจำรอม (ROM) ที่โปรแกรมได้ ซึ่งนับว่าเป็นอุปกรณ์พีแอลดี ชนิดหนึ่งซึ่งวงจรภายในของพรอมเสมือนกับประกอบไปด้วยแถวลำดับของแอนด์เกตและออเกต (And Or Array) ผลเอาต์พุตที่ขาคาต้าเอาต์พุตสามารถแสดงอยู่ในสมการฟังก์ชันผลคูณร่วมบวก(Sum of product) ของสัญญาณอินพุตที่ขาแอดเดรส รูปที่3.18แสดงถึงการต่อเป็นแถวลำดับของแอนด์เกตและออเกตของพรอมขนาด 16x4 บิต วงจรทางด้านซ้ายมือสุดเป็นแอนด์เกตที่ให้ผลเป็นผลคูณ (Product) ของกรณีอินพุตเป็น 0000 แอนด์เกตที่อยู่ถัดลงมาเป็นผลคูณของกรณีที่ อินพุตเป็น 0001 ,00101,....จนถึงตัวล่างสุดเป็นผลคูณในกรณีที่อินพุตเป็น 1111 ที่เอาต์พุตแต่ละบิตของหน่วยความจำ สามารถเลือกได้ว่าจะเป็น 1 ในกรณีที่อินพุตจากแอดเดรส เป็นอย่างไรบ้าง เหมือนกันเป็น นำเอาต์พุตจากวงจรที่ต้องการให้เอาต์พุตแต่ละบิตเป็น 1 ไปออรกันจึงเปรียบเหมือนกับว่าในพรอมมีจำนวนแอนด์เกตเท่ากับจำนวนตำแหน่งความจำและมีจำนวนออเกตเท่ากับจำนวนบิตของสัญญาณข้อมูลออก(Data output) อินพุตของออเกตทุกตัวสามารถต่อเข้ากับแอนด์เกตตัวใดก็ได้ทุกตัว ซึ่งอาจเรียกได้ว่าเป็นพีแอลดีแบบ fixed And /programmable OR



รูปที่ 3.18 แสดงลักษณะของพรอมเมื่อเปรียบเทียบเป็นวงจรในรูปผลคูณร่วมบวก

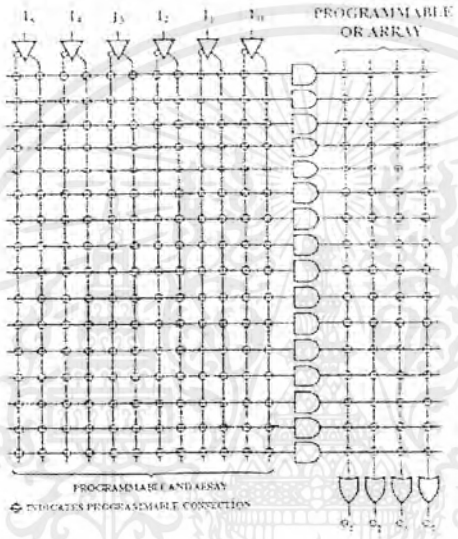
3.5.3 พีเอแอล (PAL : Programmable Array Logic)

ในช่วงกลางปี ค.ศ. 1970 บริษัท เอ็มเอ็มไอ (MMI: Monolithic memory) ในประเทศสหรัฐอเมริกา ได้พัฒนาพีเอแอล เป็นพีแอลดีชนิดใหม่ โดยใช้เทคโนโลยีแบบแอลเอสไอ(LSI :Large Scale Integration) สามารถโปรแกรมเลือกวงจรภายใน โดยใช้ฟิวส์ที่เชื่อมต่ออยู่ระหว่างสัญญาณอินพุตภายนอกและการป้อนกลับจากวงจรภายในกับแอนด์เกตที่ต่อเป็นฟังก์ชันผลคูณ(Product) อยู่ในตัววงจรรวม

3.5.4พีแอลเอ (PLA : Programmable Logic Array)

อุปกรณ์ที่สามารถโปรแกรมได้แบบพีแอลเอเกิดขึ้นเมื่อปี ค.ศ. 1975 โดยบริษัทซิกเนทริกส์ (Signetics) สหรัฐอเมริกา ซึ่งเป็นผู้ผลิตวงจรรวมรายใหญ่รายหนึ่ง ผลิตและนำเสนออุปกรณ์โดยใช้ชื่อว่า เอฟพีแอลเอ (FPLA : Field Programmable Logic Array) สามารถโปรแกรมการต่อลอจิกทั้งทางด้านแอนด์เกตและออเกตได้ และยังเลือกเอาต์พุตเป็น active high หรือ active low

โดยผ่านเอ็กคูสีบออเกต ให้ทำหน้าที่เป็นนอนอินเวอร์เตอร์หรือเป็นอินเวอร์เตอร์แล้วแต่ภายในของพีแอลเอต่อมาปี ค.ศ.1979 บริษัทซิกเนทริกส์ ได้สร้างเอฟพีแอลเอใหม่ที่มีเรจิสเตอร์ต่ออยู่ภายในวงจร เพิ่มขึ้นรวมทั้งสามารถเลือกสัญญาณอินพุตที่มาจากการป้อนกลับของเรจิสเตอร์ได้ด้วยทำให่สามารถใช้อุปกรณ์พีแอลเอใหม่นี้สร้างวงจร State machine ได้ อุปกรณ์ใหม่ที่มีเรจิสเตอร์อยู่ด้วยนี้ถูกตั้งชื่อใหม่เป็น แอฟพีแอลเอส(FPLS : Field Programmable Logic Sequencer) มีทั้งที่เป็นทีทีแอลและซีมอส



รูปที่ 3.19 แสดงวงจรพื้นฐานภายในของพีแอลเอ

3.5.5 แอลซีเอ (LCA : Logic Cell Array)

อุปกรณ์ชนิดนี้ถูกสร้างขึ้นเมื่อประมาณปี ค.ศ. 1986 โดยบริษัทไซริ้ง(XILINX Inc) ซึ่งเป็นบริษัทรวมกับบริษัทเอ็มเอ็มไอ (MMI) สร้างเป็นอาเรย์ที่ประกอบด้วยเกตจำนวน 1,200-1,800 เกต มีลักษณะสถาปัตยกรรมที่ใกล้เคียงกับเกตอาเรย์ (Gate array)โดยโปรเซสแบบซีมอส 1.6 ไมครอนขึ้นโลหะคู่ (CMOS 1.6 micron double-layer metal) สามารถโปรแกรมและลบได้โดยใช้กระแสไฟฟ้า (Static RAM based) ภายในจัดเรียงเป็นแบบเมตริกซ์ของลอจิกเซลล์ อุปกรณ์แอลซีเอตัวแรกของบริษัทไซริ้งคือ แอสซีเอเบอร์ 2.6 ประกอบด้วยเซลล์เรียงตัวกันเป็นเมตริกซ์มีจำนวน 64 เซลล์ แต่ละเซลล์เรียกว่า ซีแอลบี (CLB :Configurable Logic Block) แต่ในปัจจุบันได้พัฒนามาอยู่ในรูปของ เอฟพีจีเอ (FPGA : Field Programmable Gate Array) ซึ่งมีประสิทธิภาพความจุของเกตสูงมากขึ้น โดยสร้างออกมาเป็นอนุกรม(Series) ต่างๆ เช่น ตระกูล XC3000 และตระกูล)XC4000 เป็นต้น

3.5.6 อีพีแอลดี (EPLD :Erasable Programmable Logic Device)

อีพีแอลดีเป็นอุปกรณ์ที่สามารถโปรแกรมได้และสามารถลบและกระทำการโปรแกรมใหม่ได้ เพื่อใช้ทำวงจรต้นแบบตัวอย่างได้แก่ พีแอลดี ในอนุกรมอีพี (EP series) ของบริษัทอัลเทอร่า (Altera Inc) ประเทศสหรัฐอเมริกาซึ่งเป็นบริษัทที่ผลิตอีพีแอลดี เป็นรายแรกโดยเริ่มเมื่อปี ค.ศ . 1984 เป็น พีแอลดี ที่ใช้โปรเซสเหมือนกับซีมอสอีพรอม (CMOS EPROM) คือใช้มอสทรานซิสเตอร์เชื่อมต่อกับสัญญาณอินพุตกับจุดที่ต้องการแทนการใช้ฟิวส์ดังเดิม ทำให้สามารถโปรแกรมการต่อวงจรภายในอุปกรณ์ด้วยการจ่ายไฟฟ้าตามขนาดที่กำหนดเข้าไปยังตัวอุปกรณ์ และลบได้โดยใช้แสงอุลตร้าไวโอเลทจ่ายผ่านช่องหน้าต่างกระจกเข้าไปตกกระทบในตัวชิปของอุปกรณ์

3.6 Mask programmable

การใช้งานวงจรรวม ASIC ในเชิงพาณิชย์ จำเป็นต้องใช้วงจรรวม ASIC แบบ Mark Programmable เนื่องจากต้นทุนที่ต่อหนึ่งตัวต่ำกว่าวงจรแบบ Field Programmable ASIC ในกรณีที่การผลิตสูงนับพันนับหมื่นตัวขึ้นไป ตัวอย่างเช่น อีพีแอลดี ตัวหนึ่งอาจสูงถึงหนึ่งพันบาท ในขณะที่ผลิตวงจรรวมที่มีคุณสมบัติเหมือนกันทุกประการโดยใช้ Mask programmable แล้ว ราคาตัวหนึ่งจะลดลงเหลือไม่ถึงหนึ่งร้อยบาท การใช้วงจรรวมแบบ Mask programmable จึงมีบทบาทในการผลิตสินค้าอิเล็กทรานิกส์ในเชิงพาณิชย์ในปัจจุบัน

วงจรรวมประเภทนี้ หลังจากผู้ใช้ออกแบบวงจรและตรวจสอบการทำงานจนเป็นที่น่าพอใจแล้ว ต้องส่งให้ผู้ผลิตทำการเจือสาร ไม่สามารถโปรแกรมได้ด้วยตนเองเหมือนกับวงจรรวมแบบ Field Programmable ช่วงเวลาการผลิตออกใช้งานจึงใช้เวลานับเดือนและมีค่าใช้จ่ายเบื้องต้นในการเจือสารสูง วงจรรวมแบบ Mask programmable ASIC ในปัจจุบันได้แก่ เกตอาเรย์ เซลล์มาตรฐาน และ ฟูลัสตัม (full custom)

3.6.1 เกตอาเรย์ (Gate Array)

วงจรนี้ประกอบด้วยแถวลำดับของวงจรเกต ซึ่งอาจเป็นวงจรเกตประเภทเดียวกันหรือต่างชนิดกันก็ได้ กับขั้วต่อสายไฟ (Pad) สำหรับต่อวงจรภายนอก ผู้ใช้มีหน้าที่ออกแบบการเชื่อมโยงทางไฟฟ้าระหว่างวงจรเกตแต่ละตัวและขั้วต่อสายไฟเพื่อให้ทำหน้าที่ตามต้องการ แล้วจึงส่งผลการออกแบบนั้นไปยังโรงงานผู้ผลิตวงจรเกตอาเรย์นั้นไปทำการเจือสารต่อไป โดยทั่วไปแล้วปราฏในการใช้วงจรรวมเกตอาเรย์สร้างวงจรที่ต้องการใช้งานทั่วไปจะมีเพียง 25-30 ของพื้นที่ซิลิคอนที่ใช้สำหรับวงจรเกต ส่วนพื้นที่ที่เหลือจะใช้ในการเชื่อมโยงวงจรเกตเข้าด้วยกันเองและเข้ากับขั้วต่อสายไฟ พื้นที่ซิลิคอนจะใช้ประโยชน์สูงถึง 75 สำหรับวงจรเชิงเลขที่มีลักษณะสม่ำเสมอ เช่น หน่วยความจำ เป็นต้น

3.6.1.1 เซลล์มาตราฐาน (Standard Cell)

ปัญหาการใช้พื้นที่ซิลิคอนอย่างมากมายสำหรับการเชื่อมโยงวงจรของวงจรรวมเกตอาเรย์ทำให้เกิดขีดจำกัดในความซับซ้อนของวงจรที่ใช้ ประกอบกับผู้ใช้จำนวนมากต้องการรวมวงจรรวมมาตราฐาน อาทิ วงจรตระกูล 7400 วงจรตระกูล 4000 จำนวนหลายตัวเข้าเป็นวงจรรวม ASIC ทำให้เกิดวงจรรวม ASIC แบบเซลล์มาตราฐานขึ้น วงจรรวมเซลล์มาตราฐานนี้ ผู้ใช้เป็นผู้เลือกกลุ่มวงจรทำหน้าที่ต่างๆ เช่น เกตฟลิปฟลอป ตัวนับ ตัวเลื่อน หน่วยความจำ หรือ กระทั้งไมโครโพรเซสเซอร์จากแฟ้มข้อมูล (Library)ของผู้ผลิตซึ่งอาจเป็นแฟ้มข้อมูลทางคอมพิวเตอร์ เหมาะสมกับสำหรับวงจรเชิงเลขที่มีความซับซ้อน การผลิตวงจรรวม เซลล์มาตราฐานจะมีต้นทุนสูงกว่าวงจรรวมเกตอาเรย์และใช้เวลาในการออกแบบ และเจือสานยาวกว่าสองถึงสามเท่าตัว วงจรรวมเซลล์มาตราฐานจึงเหมาะสมกับการใช้งานทางด้านพาณิชย์ที่มีปริมาณการผลิตนับหมื่นตัวขึ้นไปบทบาทวงจรรวมเซลล์มาตราฐานจะมีเพิ่มมากขึ้นในอนาคต เมื่อต้นทุนการเจือสารน้อยลง

3.6.1.2ฟูลัสตัม (Full Custom)

วงจรฟูลัสตัม นี้ผู้ใช้เป็นผู้ออกแบบเองทั้งหมด ตั้งแต่ระดับวงจรเชิงเลขจนกระทั้งถึงระดับทางกายภาพ แล้วจึงส่งข้อมูลการออกแบบไปยังผู้ผลิตเจือสารในรูปแฟ้มข้อมูลมาตราฐาน เช่น GDS II , CIF การใช้ฟูลัสตัมASIC นี้เท่าที่แพร่หลายในปัจจุบันจะเป็นไปเพื่อการศึกษาวิจัยและเพื่อการผลิตจำนวนน้อย ส่วนใหญ่จะ เป็นการเจือสารในลักษณะที่ค่าใช้จ่ายร่วมกัน กล่าวคือ รวบรวมการออกแบบหลายวงจรลงบนแผ่นซิลิคอนเดียวกันเพื่อประหยัดค่าเจือสาร

รูปแบบการเจือสารวงจรรวมฟูลัสตัมที่ประกอบด้วยการออกแบบหลายวงจรรวมอยู่บนแผ่นซิลิคอนเดียวกัน มีกระทำ 3 รูปแบบดังนี้

1.Multi-project wafer (MPW) การเจือสารวงจรรวมฟูลัสตัมแบบนี้ แผ่นเวเฟอร์(Wafer) จะแบ่งเป็นส่วนๆที่เรียกว่า ไค (Dic) และไคแต่ละชุดจะเจือจางวงจรต่างกัน ผู้ผลิตที่ให้การบริการแบบนี้ได้แก่ โมซิล (MOSIS) และออบิต(ORBIT)ในสหรัฐอเมริกา

2. Multi-project chip (MPC) เป็นการเจือสารการออกแบบหลายวงจรลงบนไคชุดเดียวกันโดยที่ไคทุกชุดบนเวเฟอร์เดียวกันจะมีลักษณะเหมือนกัน การจัดวางวงจรที่ออกแบบลงบนไค เน้นการใช้พื้นที่ซิลิคอนให้เป็นประโยชน์สูงสุดเป็นสำคัญ ตัวอย่างของผู้ผลิตที่ให้บริการแบบนี้ ได้แก่ อาวา (AWA) ในออสเตรเลียและอีแอสทู (ES II)ในสหราชอาณาจักร

3. Multi-project reticle (MPR) เป็นการเจือสารในลักษณะผสมผสานระหว่างฟูลัสตัมกับเซลล์มาตราฐานกล่าวคือจะแบ่งออกเป็นส่วนๆอย่างสม่ำเสมอแต่ละส่วนบรรจุการออกแบบแต่

ละวงจร โดยไดชุดหนึ่งจะบรรจุวงจรที่ออกแบบได้ประมาณ 8 วงจร และไดทุกชุดในแผ่นเวเฟอร์จะมีลักษณะเหมือนกัน ใจปัจจุบันมีเพียงอาวาที่ใช้บริการการเจือสารลักษณะนี้ โดยจำกัดเฉพาะประเภทเซลล์มาตราฐานเท่านั้น

3.7 เอฟพีจีเอ (FPGA :Field Programmable Gate Array)

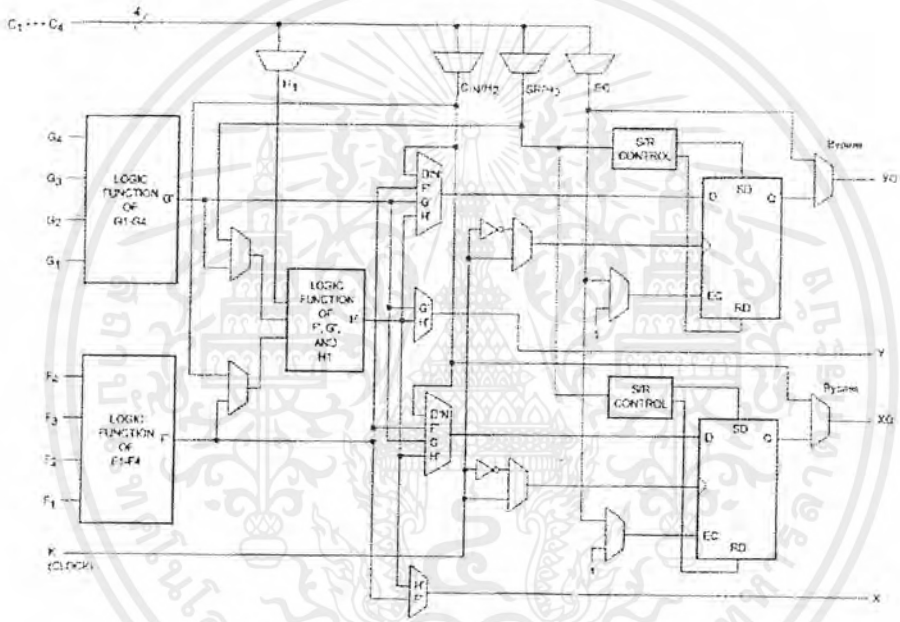
เป็นอุปกรณ์ที่ถูกพัฒนาต่อจากอุปกรณ์แอลซีเอของบริษัทไซริงซ์ (XILINX Inc) โดยมีประสิทธิภาพการทำงานและมีปริมาณความหนาแน่นของเกตสูง สามารถจะกำหนดฟังก์ชันการทำงานได้ความต้องการของผู้ใช้โดยผ่านการโปรแกรมเอฟพีจีเอได้รวบรวมข้อดีทั้งหมดของการทำคัสตัมวีแอสเอสไอ (Custom VLSI) มารวมไว้ทั้งหมดได้แก่ การออกแบบการผลิต , ลดเวลาที่จะส่งตัวผลิตภัณฑ์ออกตลาด ซึ่งเป็นประโยชน์ต่อการผลิตวงจรเป็นอย่างมาก นักออกแบบเพียงกำหนดฟังก์ชันการทำงานของวงจร ดังนั้นการออกแบบวงจรโดยใช้เอฟพีจีเอ สามารถออกแบบและทดสอบภายในเวลาเพียง 2-3 วัน เท่านั้น ตรงกันข้ามกับการออกแบบโดยใช้เกตอาเรย์ ซึ่งใช้เวลาหลายอาทิตย์ การเปลี่ยนแปลงแก้ไขแบบก็เช่นเดียวกัน จากประโยชน์ของเอฟพีจีเอ ดังกล่าวมา ทำให้เกิดการประหยัดค่าใช้จ่ายเป็นอย่างมาก เพราะได้ความเสี่ยงในการที่จะต้องแก้ไขตัววงจร การเลื่อนเวลาการออกผลิตภัณฑ์ ลดค่าเอ็นอาร์อี (NRE : Nonrecurring Engineering Cost) ลงไปด้วย

3.7.1 สถาปัตยกรรมภายในของเอฟพีจีเอ ตระกูล XC4000

สถาปัตยกรรมภายในคล้ายๆ กับเกตอาเรย์โดยทั่วไป ภายในมีลักษณะเป็นเมทริกซ์ของลอจิกบล็อก (Logic block) และล้อมรอบไปด้วยบล็อกการเชื่อมต่อของไอโอ(I/O Interface block) การเชื่อมต่อระหว่างซีแอลบี (CLB :Configuration Logic block) และไอโอบี (IOB : Input Output Block) ทำโดยผ่านช่องที่ว่างพาดผ่านระหว่างแถว(Row) และคอลัมน์(Column) มีการทำงานเหมือนกันไมโครโพรเซสเซอร์ ตัวแองซีเอจะทำงานได้ต้องใช้ Program-driven logic device หน้าที่ของซีแอลบีและไอโอบีแต่ละตัว การเชื่อมต่อภายใน(Interconnection) ถูกกำหนดไว้ในโปรแกรมจะถูกโหลดเข้าสู่คอนฟิกกูเรชัน (Configuration program) หรือเก็บไว้ในอีพรอม ภายในแอลซีเอโปรแกรมจะถูกโหลดเข้าสู่แอลซีเอเมื่อมีการจ่ายไฟ(power-up) โดยคำสั่ง (Command) ซึ่งเป็นส่วนหนึ่งของการเริ่มระบบ(System initialization) ประสิทธภาพของแอลซีเอกำหนดโดยความเร็วของลอจิก ส่วนประกอบของหน่วยความจำและการโปรแกรมเชื่อมต่อต่างๆ ความเร็วของอัตราของระบบสัญญาณนาฬิกา (System clock rate) ถูกกำหนดด้วยทอกเกิลฟลิปฟลอป สำหรับการประยุกต์ใช้โดยทั่วไปจะอยู่ที่ประมาณ 1/3 ถึง ½ ค่าที่สูงสุดของทอกเกิลเกต (Maximum toggle gate)

3.7.1.1ซีแอลบี (CLB : Configuration Logic Block)

ภายในแอลซีเอคือเมทริกซ์ของซีแอลบีแต่และตัวประกอบไปด้วยหน่วยของคอมบิเนชันลอจิกที่สามารถโปรแกรมได้ (Programmable combination logic) และส่วนของเรจิสเตอร์เก็บข้อมูล (Storage register) ส่วนของวงจรคอมบิเนชั่นลอจิกสามารถใช้สร้างวงจรทางด้านฟังก์ชันบูลีนของอินพุต ส่วนเรจิสเตอร์รับค่าจากส่วนคอมบิเนชันหรือโดยตรงจากเอาต์พุตของซีแอลบี สามารถขับวงจรคอมบิเนชันลอจิกโดยตรงผ่านเส้นทางเดินย้อนกลับ (Feedback path)



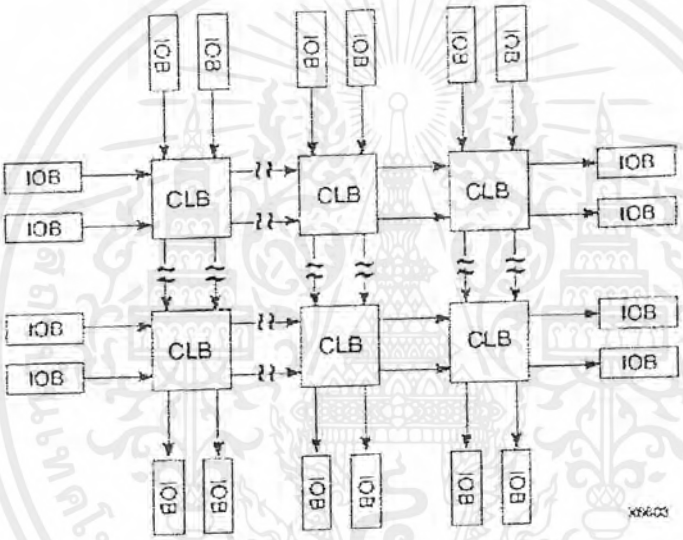
รูปที่ 3.20 แสดงผังวงจรภายในของซีแอลบีของเอฟพีจีเอตระกูล XC4000

3.7.1.2 ไอโอบี (IOB : Input Output Block)

เป็นส่วนที่ติดต่อกับวงจนภายนอกของแอลซีเอสร้างมาจากส่วนของอุปกรณ์อินพุต/เอาต์พุตที่สามารถโปรแกรมได้ (Programmable input/output device) แต่ละตัวสามารถโปรแกรมได้อย่างอิสระโดยจะให้เป็นอินพุต/เอาต์พุตแบบ 3 สถานะหรือไอโอแบบสองทิศทางก็ได้ โดยอินพุตสามารถโปรแกรมให้รู้จักระดับสัญญาณทีทีแอลและซีมอสเทรดโฮลของไอโอบีแต่ละตัวมีฟลิปฟลอปสามารถใช้เป็นบัฟเฟอร์สำหรับอินพุตและเอาต์พุต

3.7.1.3 อินเตอร์คอนเน็ค (Interconnect)

ความยืดหยุ่นของการใช้แอลซีเอมาทำอุปกรณ์ขึ้นอยู่กับการโปรแกรม ทรัพยากรต่างๆที่อยู่ภายในเข้าด้วยกันการที่จะควบคุมการเชื่อมต่อระหว่างจุดสองจุดภายในชิปที่เหมือนกับเกตอาเรย์ทั่วๆไป การเชื่อมต่อภายในแอลซีเอประกอบด้วยเน็ตเวิร์ค 2ทิศทาง คือ ทางแถวและคอลัมน์ ซึ่งอยู่ระหว่าง CLB programmable switch จะทำการเชื่อมต่ออินพุตและเอาต์พุตของไอโอบีที่จุดต่อร่วมระหว่างแถวกับคอลัมน์สามารถสลับสัญญาณจากเส้นทางไปยังส่วนต่างๆ



รูปที่3.21 แสดงเส้นทางเชื่อมต่อระหว่างไอโอบีกับซีแอลบีของเอฟพีจีเอตระกูล XC 4000

3.7.2คุณสมบัติทั่วไปของเอฟพีจีเอตระกูล XC4000

1.เป็นอุปกรณ์รุ่นที่ สามของเอฟพีจีเอ

- มีฟลิปฟลอปเป็นจำนวนมาก
- ในการผลิตฟังก์ชันของการทำงานมีความยืดหยุ่นสูง
- มีจำนวนเกตภายในจำนวน 2,000-10,000 เกต
- เพิ่มความสามารถพิเศษของเรจิสเตอร์และอินพุต/เอาต์พุต
- มีค่าแฟนเอาต์(fan-out) สูง
- มีบัสภายใน 3 สถานะ

- ทำงานกับสัญญาณทีทีแอลและซีมอส

-มีออสซิเลเตอร์แอมปลิฟลายเออร์ภายใน

- มีแรมภายในความเร็วสูง(< 25 Ns)

- ใช้กับงานที่มีความเร็วสูง(ใช้งานได้ที่ความถี่ 70/100/125MHz)

- มี Wide edge decoder

- เส้นทางการเชื่อมต่อ(Interconnect)เป็นแบบลำดับชั้น

- มีการกระจายกำลังงานของสัญญาณต่ำ

2. มีสถาปัตยกรรมภายในที่ยืดหยุ่น

-มีลอจิกบล็อกและไอโอบล็อกที่สามารถโปรแกรมได้

-มีอินเตอร์คอนเน็คและ Wide decoder ที่โปรแกรมได้

3. ทำกระบวนการซับไมครอนชนิดซีมอสได้

-มีลอจิกและอินเตอร์คอนเน็คที่มีความเร็วสูง

-ใช้กำลังงานต่ำ

4. คุณลักษณะทาง System-Oriented

-รองรับมาตราฐาน IEEE 1149.1 ในการทำ boundary-scan logic

-สามารถโปรแกรมค่า output slew rate ได้

-สามารถโปรแกรมให้อินพุตมีลักษณะพูลอัพ(Pull-up) หรือ พูลดาวน์(Pull-down) เรจิสเตอร์ได้

-ให้กระแสอินพุตได้ตั้งแต่ มิลลิแอมป์

5. ทำการโหลดเอาแฟ้มข้อมูลประเภทไบนารี

-ไม่จำกัดจำนวนครั้งในการโปรแกรมซ้ำ

-มีโหมดการโปรแกรมให้เลือก 6 โหมด

6. มีโปรแกรมช่วยพัฒนาได้แก่ XACT Development System ที่ทำงานคอมพิวเตอร์รุ่นต่างๆ เช่น 486/pentiums ,NEC PC ,Apollo ,Sun-4 ,HP700

-สามารถติดต่อกับโปรแกรมอื่นได้ เช่น Viewlogic , Mentor Graphicและ OrCAD เป็นต้น

-มีโปรแกรมการวางและการเชื่อมโยงอุปกรณ์ภายในอัตโนมัติ (Automatic place and routing)ที่ครบสมบูรณ์

-มี interactive Design Editor ที่ใช้สำหรับการทำ optimzation

-มี 288 มาโคร 34 ฮาร์มาโคร และ แรม/รอมคอมพายเลอร์

ตารางที่ 3.1. ตารางแสดงรายละเอียดของอุปกรณ์ภายในเอฟพีจีเอตระกูล XC4000

Device	Max Logic Gate (No Ram)	Max. RAM Bit (No Logic)	Typical Gate Range (logic and RAM)	CLB Matrix	Total Logic Blocks	Number Of Flip-Flop	Max. Decode Input per side	Max User I/O
XC4003E	3,000	3,200	2,000-5,000	10X10	100	360	30	80
XC4005E/L	5,000	6,727	3,000-9,000	14X14	196	616	42	112
XC4006E	6,000	8,192	4,000-12,000	16X16	256	768	48	128
XC4008E	8,000	10,368	6,000-15,000	18X18	324	936	54	144
XC4010E/L	10,000	12,800	7,000-20,000	20X20	400	1,120	60	160
XC4013E/L	13,000	18,432	10,000-30,000	24X24	576	1,536	72	192
XC4020E	20,000	25,088	13,000-40,000	28X28	784	2,016	84	224
XC4025E	25,000	32,768	15,000-45,000	32X32	1,024	5,560	96	256
XC4028EX/XL	28,000	32,768	18,000-50,000	32X32	1,024	2,560	96	256
XC4036EX/XL	36,000	41,472	22,000-65,000	36X36	1,296	3,168	108	288
XC4044EX/XL	44,000	51,200	27,000-80,000	40X40	1,600	3,840	120	320
XC4052XL	52,000	61,952	33,000-100,000	44X44	1,936	4,576	132	352
XC4062XL	62,000	73,728	40,000-130,000	48X48	2,304	5,376	144	384

3.7.4 การโปรแกรมเอฟพีจีเอตระกูล XC4000

คือกระบวนการโหลดข้อมูลในโปรแกรมไปยังแอลซีเอ เพื่อกำหนดหน้าที่ในการทำงานในแต่ละบล็อคภายใน และการเชื่อมต่อซึ่งเอฟพีจีเอตระกูล XC4000 จะต้องใช้ข้อมูลในการโปรแกรมประมาณ 350 บิตต่อซีแอลบี โดยแต่ละบิตจะบอกสถานะของหน่วยความจำสเตติกที่ควบคุมบิตในการควบคุมตารางฟังก์ชันา (function table bit) และมัลติเพล็กซ์อินพุตหรือการเชื่อมต่อกันระหว่างทรานซิสเตอร์

3.7.4.1 โหมดการโปรแกรม

เอฟพีจีเอตระกูล XC4000 มีโหมดการโปรแกรม 6 โหมด ซึ่งแต่ละโหมดจะถูกกำหนดโดยจากบิตโหมด ได้แก่ บิต M0,M1และ M2 โดยมีโหมดการโปรแกรมดังตารางที่3.2

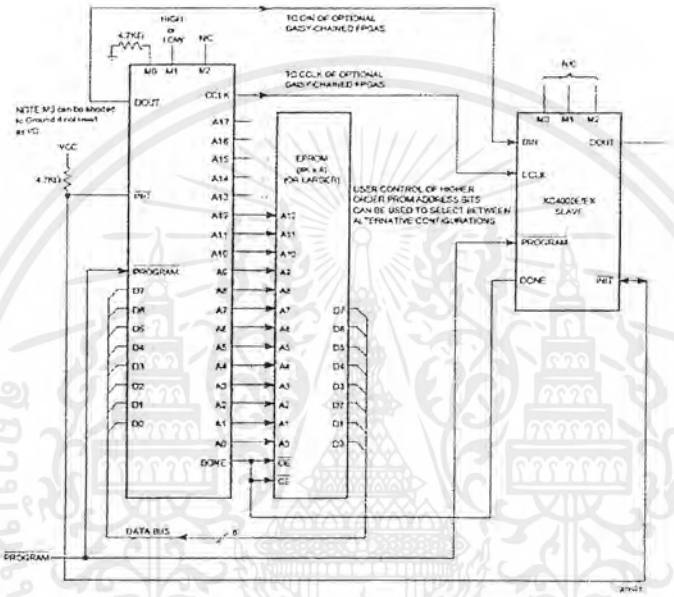
ตารางที่ 3.2 แสดงรูปตารางแบบโหมดต่างๆ ในการโปรแกรมเอฟพีจีเอตระกูล XC 4000

Mode	M2	M1	M0	Clock	Data
Master Serial	0	0	0	Output	Bit-serial
Slave Serial	1	1	1	Input	Bit-serial
Master parallel up	1	0	0	Output	Byte-Wide,00000
Master parallel down	1	1	0	Output	Byte-Wide,3FFFF
Peripheral Synch	0	1	1	Input	Byte-Wide
Peripheral Asynch	1	0	1	Output	Byte-Wide

3.7.4.1.1 โหมดหลัก(Master modes)

ในการทำงานโหมดนี้ตัวแอลซีเอจะถูกโหลดข้อมูลโครงแบบ (configuration data) จากหน่วยความจำภายนอกเข้ามาโดยอัตโนมัติ มีโหมดที่แตกต่างกัน 8 โหมด ช่วงเวลาภายในจ่ายให้ซีคล็อก(CCLK: Configuration clock) เพื่อที่จะเป็นฐานเวลาในการนำข้อมูลที่เข้ามาทางโหมดหลักแบบอนุกรม (serial master mode) และรับข้อมูลโครงแบบเข้ามาทางสัญญาณดีอิน(DIN: Data in) จากแหล่งสัญญาณซิงโครนัส เช่น Xilinx serial configuration PROM , parallel master low and master high mode โดยรับข้อมูลแบบขนานมาจากบิต D0-D7 โดย

หลักแบบขนาน(Master Parallel mode) โดยเริ่มต้นที่แอดเดรส 0000ไบต์ข้อมูล(data byte) จะถูกอ่านเข้ามาแบบขนานทุกๆสัญญาณอาร์คล็อก(RCLK: Read Clock) และส่งเข้าภายในแบบอนุกรมโดยอาศัยสัญญาณนาฬิกาโครงแบบ (configuration clock)



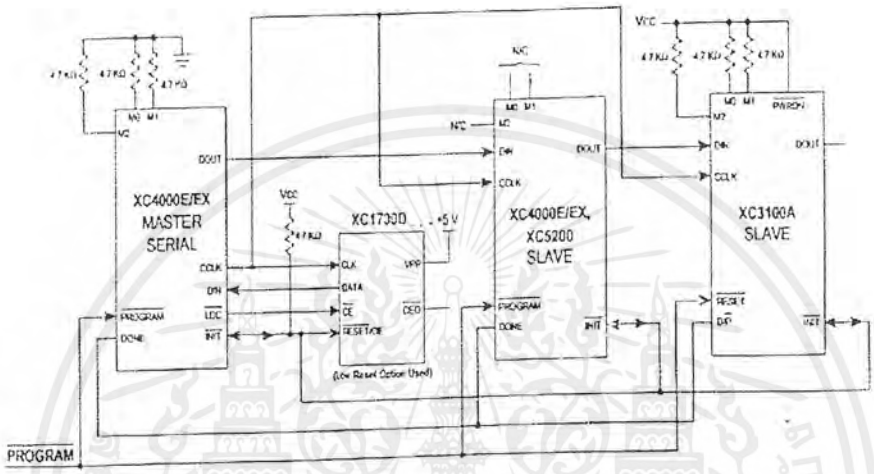
รูปที่ 3.22 แสดงผังวงจรการเชื่อมต่อเอฟพีจีเอในโหมดหลักแบบขนาน

3.7.4.2เพอริเฟอรัลโหมด (Peripheral mode)

โหมดนี้จะรับข้อมูลเป็นไบต์จากบัส สถานะพร้อม(Read)/ กำลังทำงาน(Busy) จะมีอยู่เพื่อให้สัญญาณต่างๆสื่อสารกันได้อย่างมีประสิทธิภาพมากขึ้น
ในอะซิงโครนัสโหมดนั้นจะออสซิเลเตอร์ภายในจะสร้างสัญญาณ CCLK เพื่อรับข้อมูลที่เป็นไบต์ให้อยู่ในรูปแบบที่เหมาะสม และในซิงโครนัสโหมดนั้นสัญญาณนาฬิกาภายนอกจะเป็นตัวควบคุมการรับข้อมูลที่เป็นไบต์

3.7.4.3 โหมดรองแบบอนุกรม(Slave Series modes)

ในโหมดนี้แอลซีเอจะรับข้อมูลโครงแบบที่เป็นอนุกรมในขาขึ้นของสัญญาณ CCLK และหลังจากรับข้อมูลมาแล้วจะส่งข้อมูลเพิ่มเติมออกไปด้วยและแอลซีเอก็จะถูกการซิงโครไนซ์ในขาลงถัดไปของสัญญาณ CCLK



รูปที่ 3.23 แสดงแผนผังการเชื่อมต่อเอฟพีจีเอในโหมดรองแบบอนุกรม

3.7.5 การใช้ความสามารถของแรมในเอฟพีจีเอตระกูล XC4000

แอลซีเอทำงานโดยใช้ตารางการค้นหา(Look-up table) ซึ่งจะทำการเก็บตารางที่ว่านี้ในสแตติกแรม ซึ่งจะถูกเขียนในระหว่างการโปรแกรมโครงแบบลงบนแอลซีเอและจะถูกอ่านในการโอเปอร์-เรชัน ดังนั้นแรมภายในจึงควรถูกรวมไว้ในการออกแบบของผู้ใช้ด้วย

หน้าที่ของแรมในเอฟพีจีเอตระกูล XC 4000 มีหน้าที่คล้ายแรมทั่วๆไป เช่น เอฟไอเอฟโอ (FIFO :First in First out) แอลไอเอฟโอ (LIFO: Last in First out)เรจิสเตอร์ไฟล์แอปพลิเคชันบางอย่างเช่น เรจิสเตอร์เลื่อนข้อมูล (Shift register) แรมในเอฟพีจีเอตระกูล XC 4000 มีความเร็วสูงเหมือนเอสแรม(SRAM) จึงไม่จำเป็นต้องคำนึงถึงเวลาหน่วงของการเชื่อมต่อ (Interconnection delay)

ตารางที่3.3 แสดงตาราง จำนวนแรมในเอฟพีจีเอตระกูล XC 4000

RAM Module	Equivalent Logic	XC4003	XC4005	XC4010
16 x1	4- input Function Generation (F or G)	200	392	800
32x1	Two- input Function Generation One3- input Function Generation(F+G+H)	100	196	400

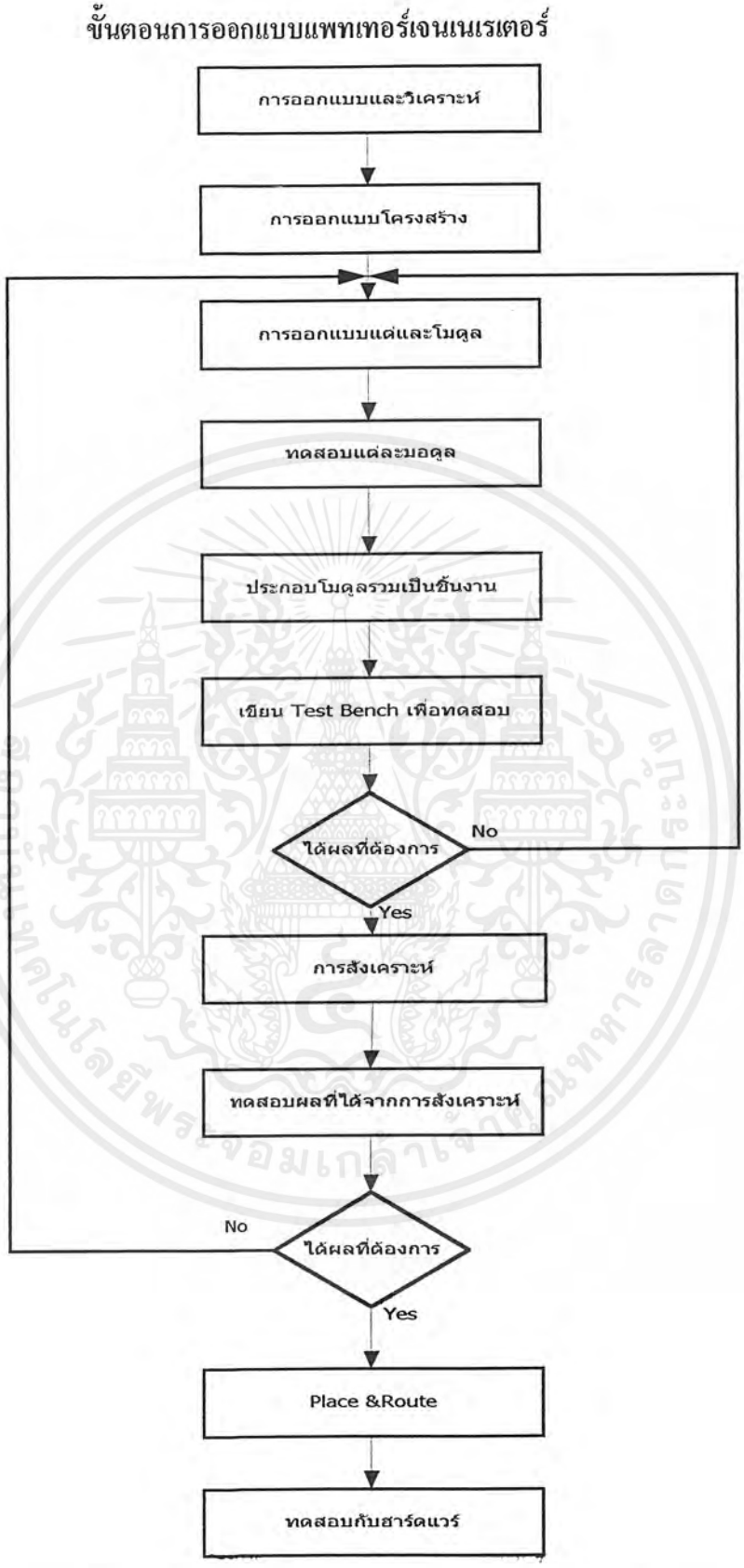
บทที่ 4
หลักการทำงานและการออกแบบวงจร

ในบทนี้จะอธิบายถึงการทำงานต่างๆที่ได้ทำในโครงงานนี้ว่ามีขั้นตอนปฏิบัติงานอย่างไรและรายละเอียดแต่ละคอมโพเนนท์ที่ทำการออกแบบรวมทั้งแนวทางการทดสอบโครงงานนี้ว่าได้ผลอย่างไร เพื่อทดสอบความถูกต้องในการทำงานของแพทเทอร์นเจนเนเรเตอร์ที่ได้ทำการออกแบบ

4.1ขั้นตอนการออกแบบแพทเทอร์เจนเนเรเตอร์

ในการที่เราจะออกแบบชิ้นงานมาได้ เราจะต้องเข้าใจในชิ้นงานที่เราจะออกแบบโดยเราต้องเข้าใจทฤษฎีของตัวงานอย่างละเอียดว่าสัญญาณซิงค์โคไนส์เซชั่นของโทรทัศน์มีช่างเวลาเท่าใดและรูปร่างของสัญญาณเป็นอย่างไร ซึ่งถ้าเราเข้าใจถึงการสร้างสัญญาณแล้วเราจะมองออกว่าเราจะออกแบบด้วยภาษา วีเอชดีแอลลักษณะ

โดยเราจะแบ่งออกเป็นส่วนๆเพื่อให้ง่ายในการออกแบบตามหลักการของออกแบบ แบบ ท็อปดาวท์ดีซายน์ (Top down designe) โดยเราจะต้องรู้โครงสร้างของงานที่เราจะออกแบบแล้วเราจะใช้ภาษาวีเอชดีแอลมาเขียนมาเป็นโมดูล(Moduel)แล้วนำแต่ละโมดูลมาทำการเชื่อมต่อกันเพื่อให้ได้สัญญาณที่เราต้องการที่เราออกแบบ ซึ่งแต่ละโมดูลที่เราทำการออกแบบสามารถตรวจสอบได้โดยการเขียนเทสเบ็น ด้วยการเขียนด้วยภาษาวีเอชดีแอล ซึ่งเป็นลักษณะการทดสอบโดยการจำลอง สัญญาณที่เข้ามาเพื่อที่จะดูลักษณะขอบสัญญาณที่เราทำการออกแบบและการดีเลย์ขอบสัญญาณที่ได้ เพื่อใช้ในการแก้ไขสัญญาณที่เราออกแบบเพื่อให้ได้สัญญาณที่เราต้องการเพื่อให้การทำงานมีความถูกต้องที่สุด แล้วนำวีเอชดีแอลโค๊ด
มาทำการสังเคราะห์(Synthesis) เพื่อทำการแปลงวีเอชดีแอลโค๊ดที่ทำการออกแบบในพฤติกรรม(Behavior) ให้เป็นวีเอชดีแอลในระดับเกตรีเวว(Gate-level) แล้วนำวีเอชดีแอลที่ได้จากการสังเคราะห์นี้ไปทำการทดสอบด้วยเทสเบ็น ตัวเดิมที่เคยทดสอบกับโค๊ดในระดับพฤติกรรมเพื่อดูว่าวงจรที่ได้จากการสังเคราะห์มีฟังก์ชันการทำงานเหมือนที่เราออกแบบหรือไม่ถ้าไม่ได้ก็ทำการแก้ไขโค๊ดในระดับพฤติกรรมใหม่ แล้วทำตามขั้นตอนที่กล่าวมาซ้ำจนกว่าจะได้ตามที่เราออกแบบ เมื่อได้ตามที่เราออกแบบ ขั้นตอนต่อไปคือการเพลส แอน เร้า(Place and rount) แล้วดาวน์โหลดลงสู่ชิพเอฟพีจีเอ แล้วทำการต่อทางฮาร์ดแวร์ขอบบอดที่เราทดสอบเพื่อทำการทดสอบสัญญาณที่ได้



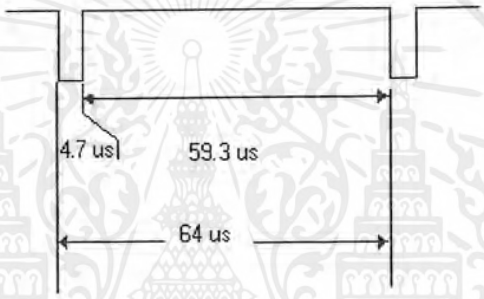
รูปที่ 4.1 Flowchart แสดงขั้นตอนในการออกแบบทีวีแพทเทอร์นเจนเนเรเตอร์ ด้วยภาษา VHDL

4.2 การกำหนดรูปแบบสัญญาณเอ้าท์พุทแพทเทอร์เจนเรเตอร์

ขั้นตอนต่อไปก็กำหนดแนวทางที่จะสร้างสัญญาณเอ้าท์พุทแพทเทอร์เจนเนเรเตอร์มีรายละเอียดดังนี้

1.สัญญาณซิงก์หรือสัญญาณกวาดทางแนวนอน

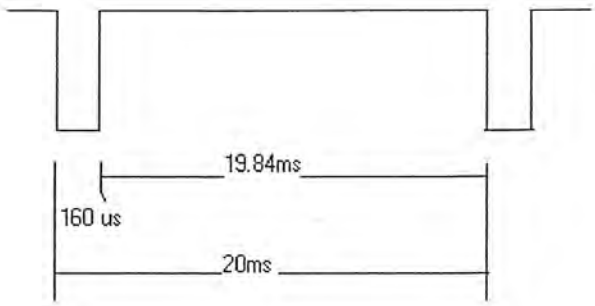
โดยสัญญาณซิงค์โครไนซ์ของระบบโทรทัศน์ระบบ PAL จะมีคาบเวลาเท่ากับ 64 μSและช่วงเวลา Time-off จะเท่ากับ 4.7μS และช่วง Time-on จะเท่ากับ 59.3 μS โดยการสร้างสัญญาณกวาดทางแนวนอนจะกล่าวไว้ในหัวข้อรายละเอียดของการออกแบบ. สัญญาณด้วยภาษาVHDL



รูปที่ 4.2 สัญญาณกวาดทางแนวนอนหรือสัญญาณซิงก์

2.สัญญาณเวอร์ติคอร์น หรือ สัญญาณกวาดทางแนวตั้ง

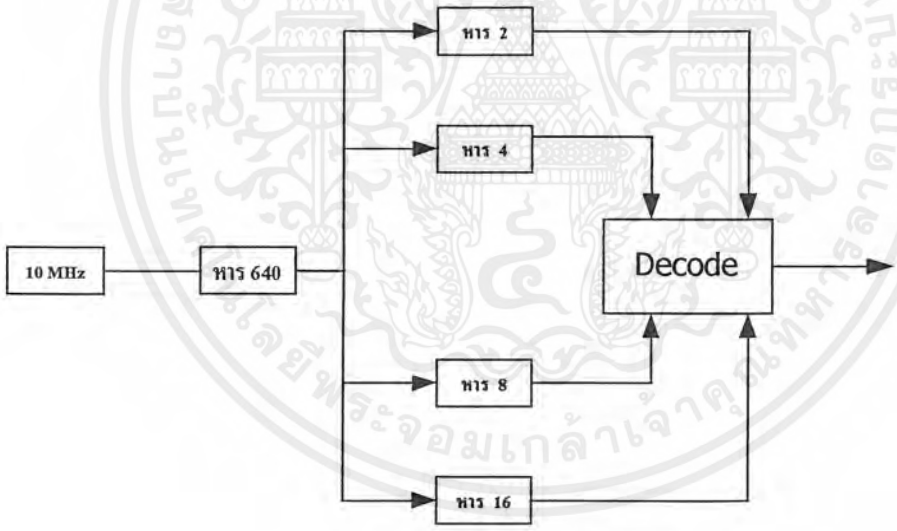
สัญญาณกวาดทางแนวตั้งหรือสัญญาณเวอร์ติคอร์นเป็นสัญญาณที่มีคาบเวลาเท่ากับ 20 ms โดยจะแบ่งเป็นช่วง Time-off เท่ากับ 160 μS และช่วง Time-on เท่ากับ 19.84 msโดยการสร้างสัญญาณกวาดทางแนวตั้งจะขอกล่าวไว้ในหัวข้อรายละเอียดของการออกแบบ.สัญญาณด้วยภาษา VHDL



รูปที่ 4.3 สัญญาณกวาดทางแนวตั้งหรือสัญญาณเวอร์ติเคอร์

3.สัญญาณฮอร์ลิเซ็นทอร์หรือสัญญาณกวาดทางแนวนอน

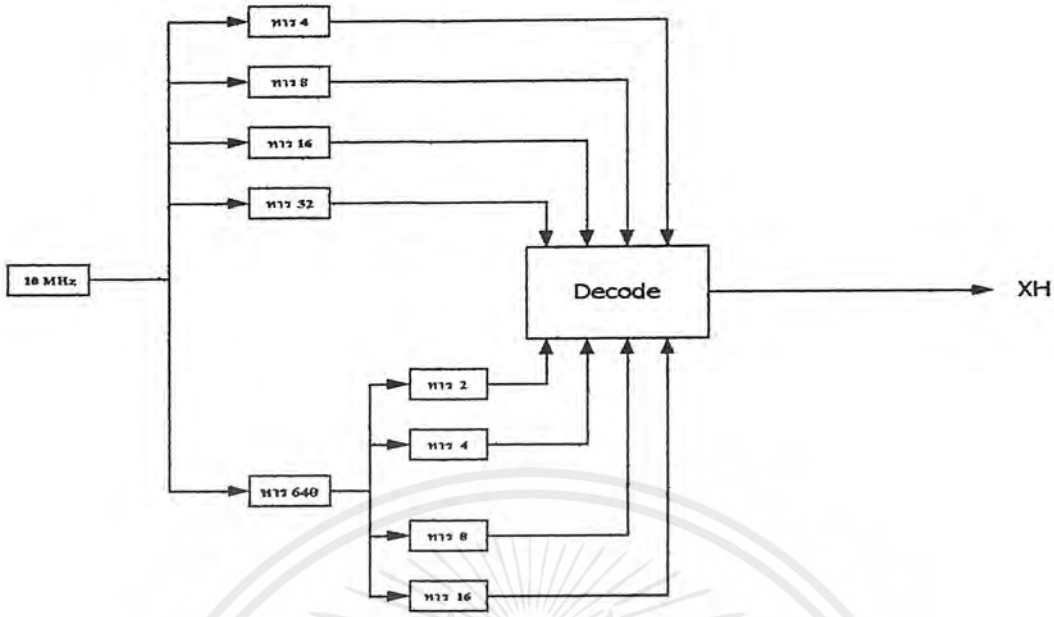
มีลักษณะเป็นพลัซ์กว้าง 4.7 ไมโครวินาทีมีช่วงห่างกัน 64 ไมโครวินาที หรือมีความถี่เป็น 16,250 เฮิรต์ ช่วงระหว่างพลัซ์นี้ปกติจะบรรจุสัญญาณรายละเอียดของภาพ



รูปที่ 4.4 แสดงบล็อคไดอะแกรมของการสร้างสัญญาณกวาดทางแนวนอน

4.สัญญาณรูปตาราง

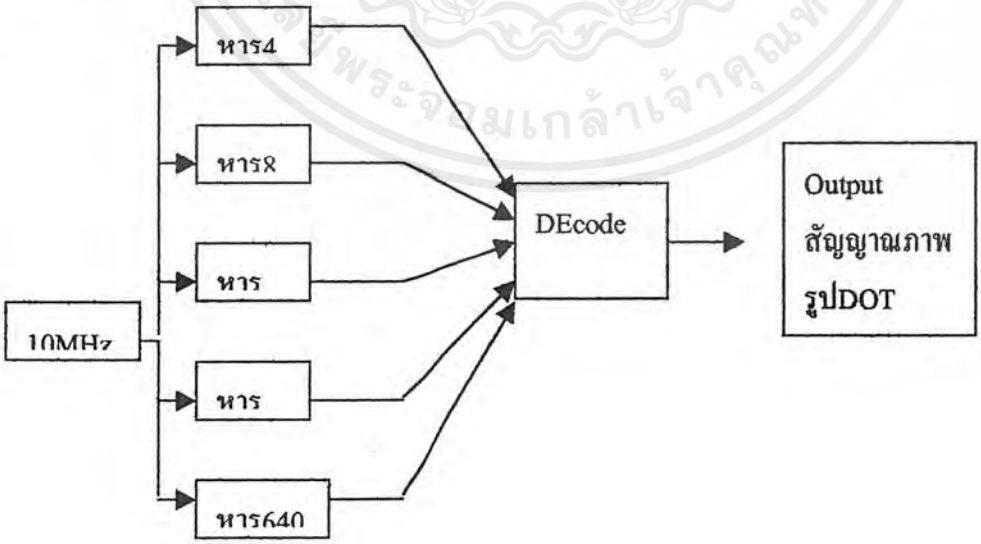
เป็นสัญญาณที่เกิดจากการหารกันของความถี่และนำผลหารที่ได้มาเข้าสู่การดีโค้ด(Decode)โดยผลของการดีโค้ดออกมาก็จะเป็นสัญญาณรูปภาพตาราง สัญญาณรูปภาพจุด



รูปที่ 4.5 แสดงบล็อคไดอะแกรมของการสร้างสัญญาณรูปตาราง

5.สัญญาณรูปภาพจุด

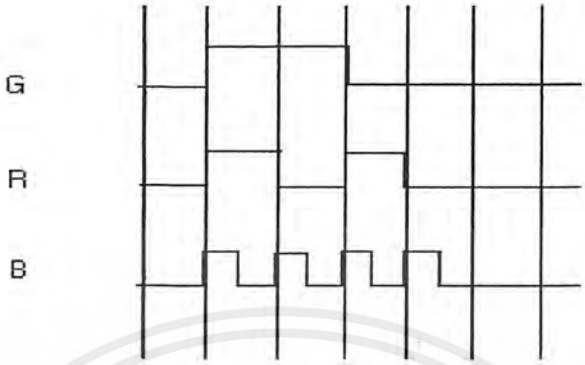
เป็นสัญญาณที่เกิดจากการหารกันของความถี่และนำผลหารที่ได้มาเข้าสู่การดีโค้ด(Decode)โดยผลของการดีโค้ดออกมาก็จะเป็นสัญญาณรูปภาพจุด รูปข้างล่างแสดงบล็อคไดอะแกรมของสัญญาณรูปภาพจุด ส่วนการสร้างสัญญาณรูปภาพจุดจะขอกล่าวไว้หัวข้อ รายละเอียดของการออกแบบสัญญาณโดยภาษาVHDL



รูปที่ 4.6 แสดงบล็อคไดอะแกรมของการสร้างสัญญาณรูปภาพจุด

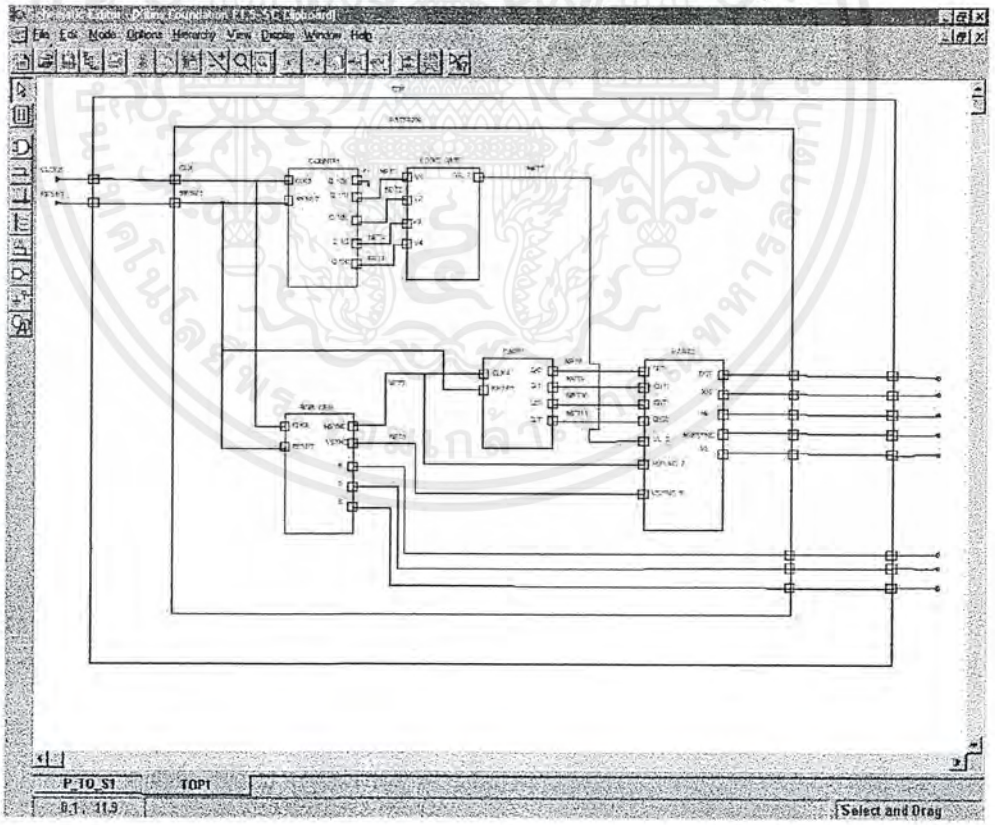
6. สัญญาณภาพ RGB

เป็นสัญญาณทดสอบภาพที่มีลักษณะสัญญาณดังนี้



รูปที่ 4.7 รูปสัญญาณRGB

เมื่อเราได้กำหนดสัญญาณเอาท์พุทได้แล้วเราก็ทำการออกแบบได้วงจรโมดูลรวมดังนี้



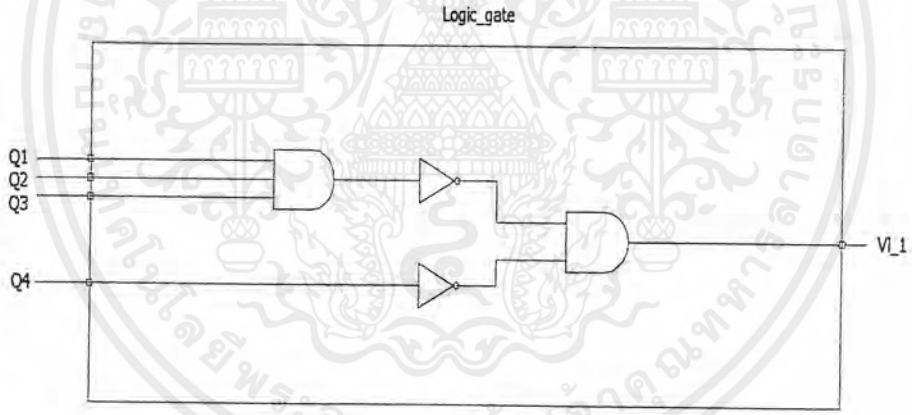
รูปที่4.8 บล็อคโมดูลของทีวีแพตเทอร์เจนเนเรเตอร์ส่วนภาคกำเนิดสัญญาณโทรทัศน์

จากบล็อกไดอะแกรมของบล็อกใหญ่จะประกอบไปด้วยบล็อกเล็กจำนวน 5 บล็อกที่ทำการเชื่อมต่อกันโดยเราจะป้อนสัญญาณคล็อกและสัญญาณรีเซ็ตให้กับวงจรซึ่งจะได้สัญญาณเอาท์พุต 8 สัญญาณคือ สัญญาณทางแนวตั้ง(VL),สัญญาณทางแนวนอน(HL),สัญญาณตาราง(XH), สัญญาณรูปจุด(Dot),สัญญาณR,สัญญาณG,สัญญาณB,สัญญาณMixsync ซึ่งแต่ละบล็อกย่อยสามารถอธิบายดังนี้คือ

1.บล็อก COUNTP1

จะทำหน้าที่ในการหารความถี่ของคล็อกที่ทำการป้อนเข้าไปในวงจรซึ่งโปรแกรมภายในจะมีตัวนับสัญญาณคล็อกโดยที่เมื่อมีสัญญาณรีเซ็ตเข้ามาจะทำให้เอาท์พุตที่ได้เป็นศูนย์ทั้งหมดแต่ถ้ามีการเปลี่ยนแปลงขอบคล็อกและคล็อกเท่ากับหนึ่งแล้วตัวนับจะทำการเพิ่มค่าที่ละหนึ่งซึ่งจากวงจรจะเป็นการหารความถี่ดังนี้คือ หาร2,หาร4,หาร 8,หาร16และหาร 32ตามลำดับ

2.บล็อก logic_gate



รูปที่4.9 บล็อคโมดูล Logic gate

จากบล็อคจะเห็นได้ว่าเราใช้ภาษาวีเอชดีแอลมาทำการออกแบบเกตต่างๆซึ่งเราจะป้อนความถี่ที่ถูกหารความถี่มาจากบล็อกCOUNTP1 มาเป็นอินพุตของ logic_gate แล้วผ่านเกตต่างๆที่เราออกแบบด้วยภาษาวีเอชดีแอลซึ่งเอาท์พุตของสัญญาณที่ได้นั้นจะได้สัญญาณทางแนวตั้ง(VL)

3.บล็อก RGB_GEN

จะทำหน้าที่ผลิตสัญญาณ R,G,B,Hsync,Vsync ซึ่งภายในวงจรจะมีตัวนับเช่นเดียวกับ COUNTP 1 ซึ่งจะทำหน้าที่นับคล็อกที่ป้อนเข้ามาถ้ามีสัญญาณรีเซ็ตเท่ากับศูนย์ก็จะทำให้สัญญาณเอาท์พุตที่ทุกตัวมีค่าเท่ากับศูนย์แต่ถ้ามีการเปลี่ยนแปลงขอบสัญญาณคล็อกและคล็อกเท่ากับหนึ่งแล้วตัวนับคล็อกในโปรแกรมก็จะทำการนับ

สัญญาณ hsync จะอ้างอิงตัวนับสัญญาณที่สมมุติชื่อว่า hcnt ซึ่งตัวนับตัวนี้จะทำการเพิ่มค่าเมื่อตัวมันมีค่าน้อยกว่า 641 มันจะให้ค่าเป็น 1 แต่ถ้าค่าที่นับได้มีค่าเกินกว่านี้มันจะให้ค่าเป็น 0 ซึ่งค่าของ hsync จะเป็น 0 เมื่อตัวนับอยู่ในช่วงของ 0 ถึง 48 ถ้าไม่ได้อยู่ในช่วงนี้มันจะให้เป็น 1

สัญญาณ Vsync จะอ้างอิงตัวนับสัญญาณที่สมมุติชื่อว่า vcnt ซึ่งตัวนับตัวนี้จะทำการเพิ่มค่าเมื่อตัวมันมีค่าน้อยกว่า 313 มันจะให้ค่าเป็น 1 แต่ถ้าค่าที่นับได้มีค่าเกินกว่านี้มันจะให้ค่าเป็น 0 ซึ่งค่าของ vsync จะเป็น 0 เมื่อตัวนับอยู่ในช่วงของ 0 ถึง 3 ถ้าไม่ได้อยู่ในช่วงนี้มันจะให้ค่าเป็น 1

สัญญาณ G จะให้ค่าเป็น 1 โดยการเอา ค่านับhcnt ที่อยู่ในช่วง105 ถึง 366มา ANDกับ vcnt ที่อยู่ในช่วงระหว่างมากกว่าหรือเท่ากับ 20 จนถึง312ถ้าไม่ใช่เงื่อนไขนี้จะเป็น 0

สัญญาณ R จะมีค่าเท่ากับ 1 โดยการเอาเอาค่านับ hcnt ที่อยู่ในช่วง 105จนถึง 235 และช่วง 365 จนถึง 465 มาทำการออร์กัน แล้วนำมาแอนด์กับ vcnt ที่อยู่ในช่วง 20จนถึง 312 ถ้าไม่อยู่ในช่วงนี้สัญญาณที่ได้จะมีค่าเท่ากับ 0

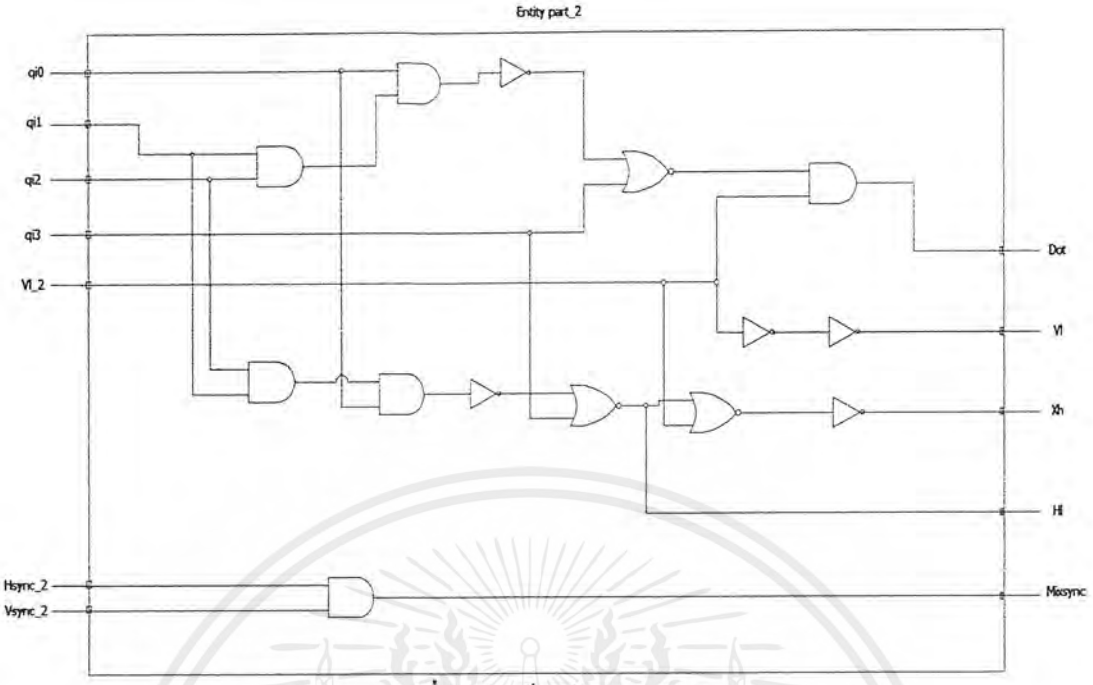
สัญญาณ B จะมีค่าเท่ากับ 1 โดยการเอา hcnt ที่อยู่ในช่วง 105 จนถึง 170 ,ช่วง 235จนถึง 300,ช่วง365 จนถึง 431 ,ช่วง 495 จนถึง 560 นำมาออร์กันและนำมาแอนด์กับ vcnt ที่อยู่ในช่วง 20จนถึง 312 ถ้าไม่ใช่ตามเงื่อนไขนี้จะได้สัญญาณเอาต์พุตเป็น 0

4.บล็อก PART1

จะทำหน้าที่หารความถี่เช่นเดียวกับ COUNTP1 ซึ่งคล็อกที่นำมาป้อนเป็นอินพุตได้มาจากhsync ของบล็อคของ RGB_GEN ซึ่งหลักการจะเหมือนกับ COUNT1 ทุกอย่างต่างกันเพียงคล็อกที่นำมาป้อนให้วงจรเท่านั้น

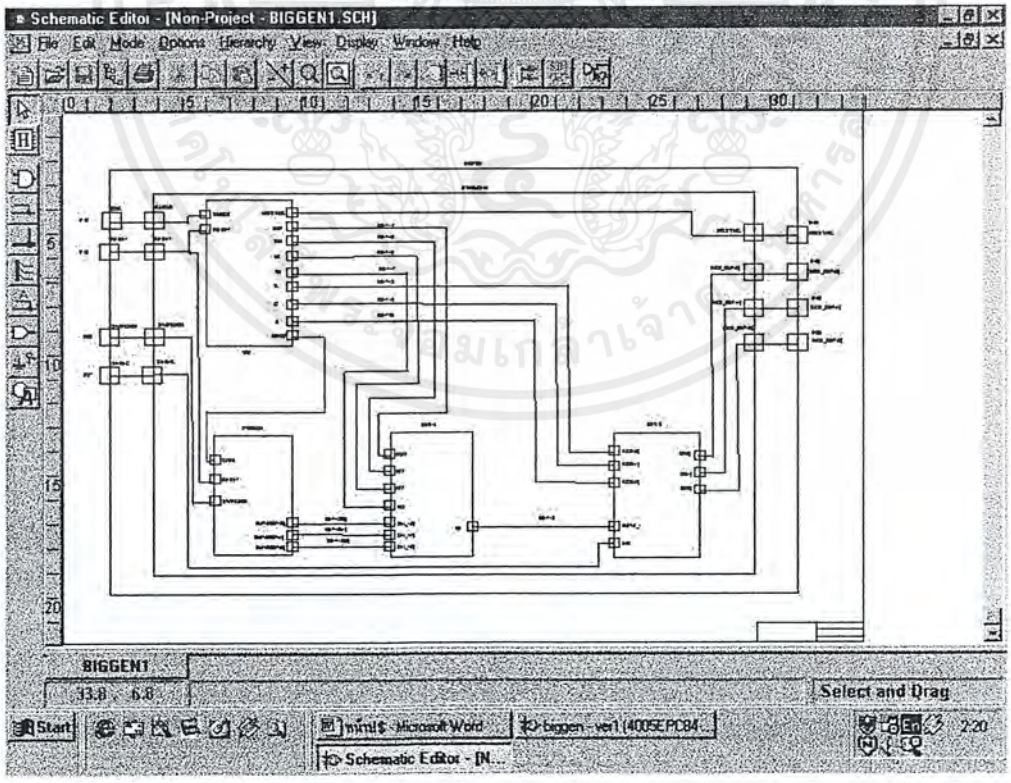
5.บล็อก PART2

จะมีสัญญาณอินพุตที่มาจากส่วนของ PART1 และ ส่วนของ RGB_GEN ซึ่งในส่วนนี้สัญญาณที่ได้ทางเอาท์พุตคือสัญญาณ Dot,XH,HL,Mixsyncและ VL



รูปที่ 4.10 บล็อคโมดูล Part2

จากบล็อคที่แสดงเราจะแทนเกตโดยใช้ภาษาวีเอชดีแอลเขียนแทนเกตต่างๆ และโปรแกรมVHDL ที่ใช้ในการออกแบบจะแสดงไว้ในภาคผนวก (ก)



รูปที่4.11 โมดูลภาคกำเนิดสัญญาณโทรทัศน์ร่วมกับโมดูลสวิตช์เลือกภาพ

รูปที่4.11 โมดูลภาคกำเนิดสัญญาณโทรทัศน์ร่วมกับโมดูลสวิตซ์เลือกภาพ

จากโมดูลภาคกำเนิดสัญญาณโทรทัศน์เมื่อสร้างเสร็จแล้วเราให้ชื่อว่าโมดูล TOPและส่วนภาคเลือกสัญญาณแพทเทอร์เราจะใช้หลักการของการมัลติเพล็กซ์เข้า 4 ออก 1 เพื่อเลือกแพทเทอร์ภาพทดสอบจะให้ชื่อโมดูลว่า MUX4 ส่วนโมดูลSWITCH จะสร้างสัญญาณ 3 บิตจ่ายให้กับ MUX4 ซึ่ง 3บิตนี้จะเลือกได้ 8 แพทเทอร์ (2^3= 8) โดยเมื่อเรากดสวิตซ์ 1 ครั้ง แพทเทอร์ก็จะเปลี่ยนไป โดยเราเซ็ทไว้ที่ 4 แพทเทอร์คือ สัญญาณรูปแนวนอน,สัญญาณรูปแนวตั้ง,สัญญาณรูปจุด,และสัญญาณรูปตาราง โมดูลต่อไปก็จะเป็นโมดูล MUX2 จะเป็นโมดูลมัลติเพล็กซ์เช่นเดียวกันแต่เป็นการเข้า 2 ออก 1 ซึ่งสัญญาณขาเข้าโมดูล MUX2 ก็คือ สัญญาณคัลเลอร์บาร์(RGB signal)และสัญญาณแพทเทอร์ โดยจะมีสวิตซ์ควบคุมเช่นเดียวกัน

4.3 การทดสอบการทำงานของแพทเทอร์เจนเนเรเตอร์โดย การ Simulation

ซอฟท์แวร์ทูลที่ใช้: โดยการใช้ V-System

โปรแกรม V-System เป็นโปรแกรมที่ใช้ในการ Simulate การทำงานของ VHDL Code ว่าการทำงานตรงตามความต้องการหรือไม่ โดยจะแสดงผลให้ผู้ออกแบบดูในรูปของ Wave from เนื่องจากก่อนที่จะนำ VHDL Code ไปทำการซิมมูเลชั่นจะต้องมีตัวคอมไพลตัวโค๊ดเพื่อตรวจสอบความถูกต้องของตัวโค๊ดก่อนดังนั้น โปรแกรม V-System จึงมีประโยชน์อีกประการหนึ่งคือสามารถใช้ในการตรวจสอบว่าโค๊ด VHDL ที่เขียนเสร็จแล้วนั้นมีความถูกต้องตามหลักไวยากรณ์ของภาษา VHDL หรือไม่ โปรแกรม V-System อนุญาตให้ผู้ใช้สามารถสั่ง Force input signal เพื่อกำหนดสัญญาณอินพุตให้กับตัววงจร และยังอนุญาตให้ผู้ใช้สามารถสร้าง Macro file เพื่อเก็บ Test pattern เอาไว้สำหรับจำลองการทำงานในกรณีที่มีอินพุตซ้ำๆกันได้ ในการ Simulate การทำงานนั้นผู้ใช้สามารถเลือกการทดสอบแบบ Single-step นั้นคือผู้ใช้สามารถทดสอบการทำงานของ VHDL-Code ได้ทีละบรรทัดว่าหลังจากการทำคำสั่งในบรรทัดนั้นไปแล้วจะมีผลกระทบอย่างไรต่อ Variable และ Signal ของวงจร

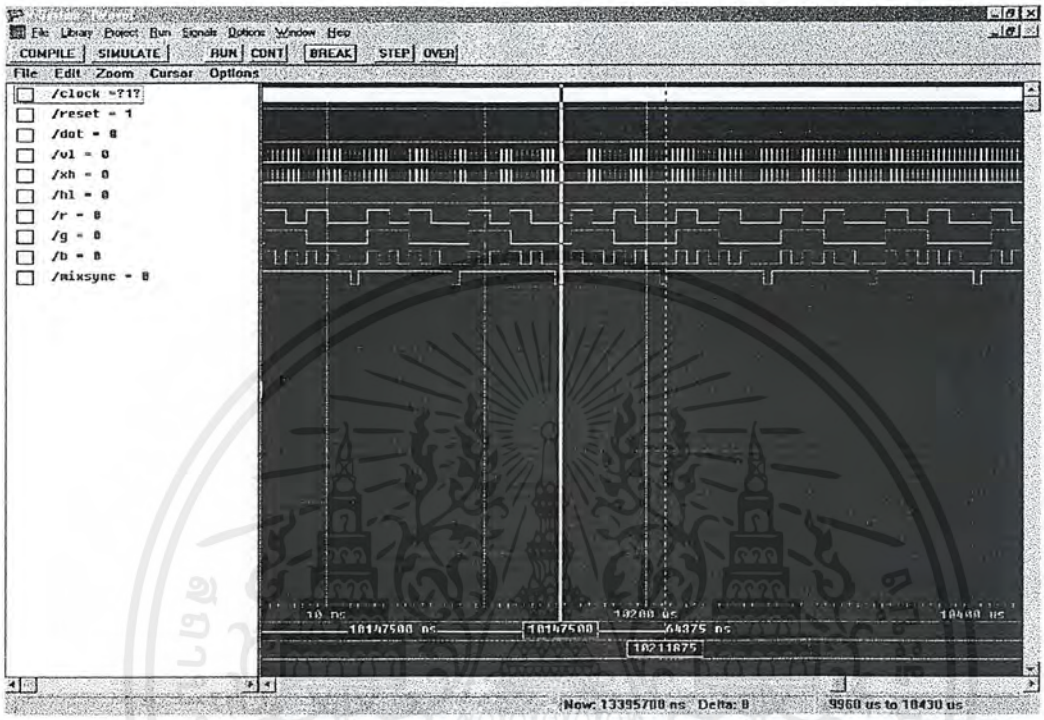
4.3.1การสร้าง Test bench

ในการทดสอบการทำงานของโปรแกรมจะต้องเขียน VHDL Code เพื่อจำลองสัญญาณอินพุตที่ป้อนให้กับวงจรซึ่งจะต้องถูกต้องตามหลักของไวยกรณ์ของ VHDL

4.3.2การทดสอบการทำงาน

ในการทดสอบการทำงานโดยการดูผลที่ได้จากการซิมมูเลชั่นโดยใช้โปรแกรม V-System เพื่อดูว่าโปรแกรมที่เราออกแบบมีการดีเลย์ของสัญญาณเป็นอย่างไรเราสามารถยอมรับได้หรือไม่ ถ้ายอมรับไม่ได้ก็ต้องทำการแก้ไขโปรแกรมที่เราออกแบบไว้

เมื่อทำการซิมมูเลชั่นของโมดูลรวมที่ได้ทำการคอมโพเนนท์เรียบร้อยแล้วจากการใช้โปรแกรม V-System จะได้สัญญาณที่ซิมมูเลชั่นดังนี้



รูปที่ 4.12 สัญญาณเอาท์พุทที่ได้จากการซิมมูเลทโมดูลรวมของทีวีแพทเทอร์นเจนเนเรเตอร์

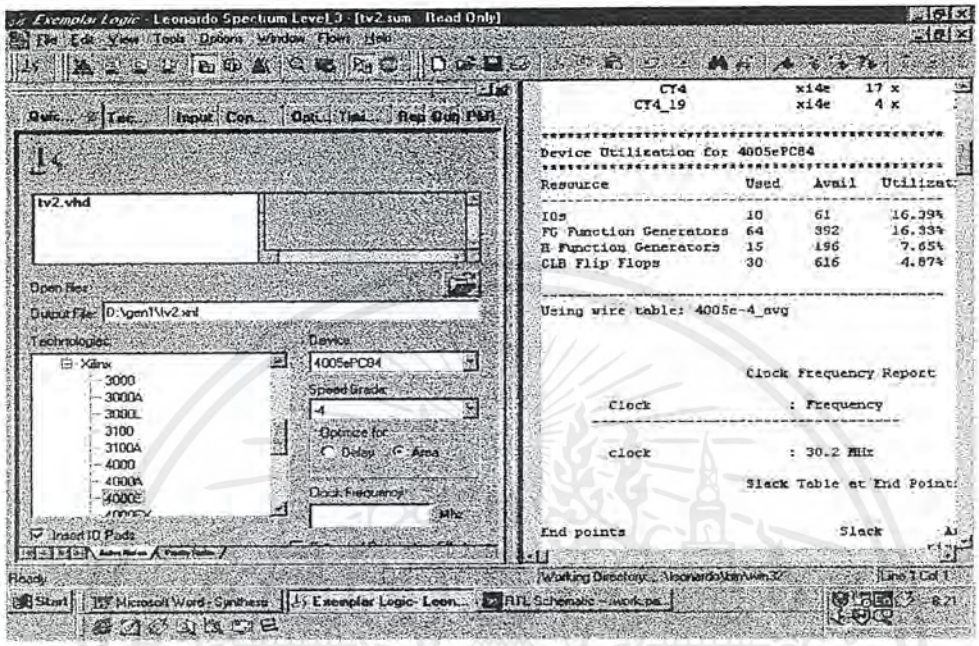
4.4 การ Synthesis และบันทึกโปรแกรมลง FPGA

4.4.1 ซอฟท์แวร์ทูลล์ที่ใช้

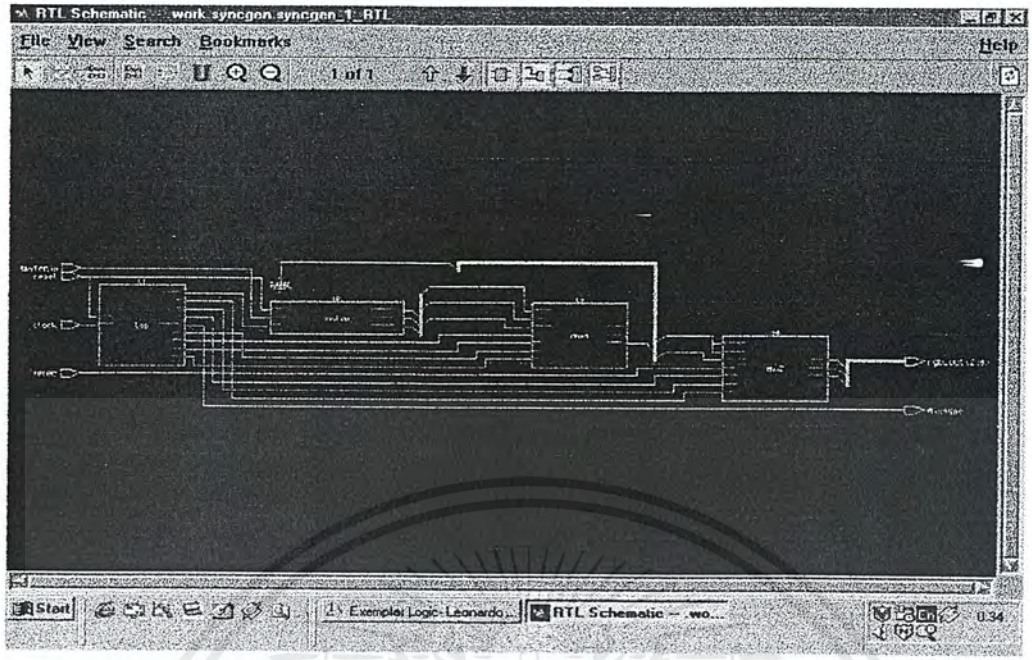
4.4.1.1 โปรแกรม Leonardo Spectrum

เป็นโปรแกรมที่ใช้ในการ Synthesis วงจรเพื่อแปลง VHDL Code ในระดับ Behavior หรือ RTL ให้เป็นวงจรในระดับ Gate-Level ซึ่งในขบวนการ Synthesis นั้นผู้ใช้จะต้องกำหนดว่าจะ Synthesis ลงฮาร์ดแวร์ตัวใด และยังสามารถกำหนดรูปแบบของการ Synthesis ได้ว่าต้องการวงจรแบบมีจำนวนเกทน้อยที่สุดหรือต้องการวงจรแบบมีความเร็วสูงสุด ผู้ใช้สามารถเลือกเอาท์พุทของการ Synthesis ได้หลายแบบเช่น สร้างไพล์นามสกุล .VHD , สร้างไพล์นามสกุล .V, สร้างไพล์นามสกุล .XNF หากเลือกเอาท์พุทเป็นไพล์.VHD แล้วจะได้ VHDL Code ในระดับ Gate-Level ซึ่งผู้ใช้สามารถนำไปซิมูเลทการทำงานได้ว่าไฟล์ที่ได้จากการ Synthesis นี้ยังคงมีการทำงานเหมือนกันกับโค๊ดในระดับ Behavior หรือไม่ หากเลือกเอาท์พุท เป็นไฟล์ .EDF หรือ .XNF ก็จะสามารถนำไฟล์ดังกล่าวส่งไปให้โปรแกรม Xilinx Foundation เพื่อทำขั้นตอนการ Mapping, และ Place and Route ต่อไป

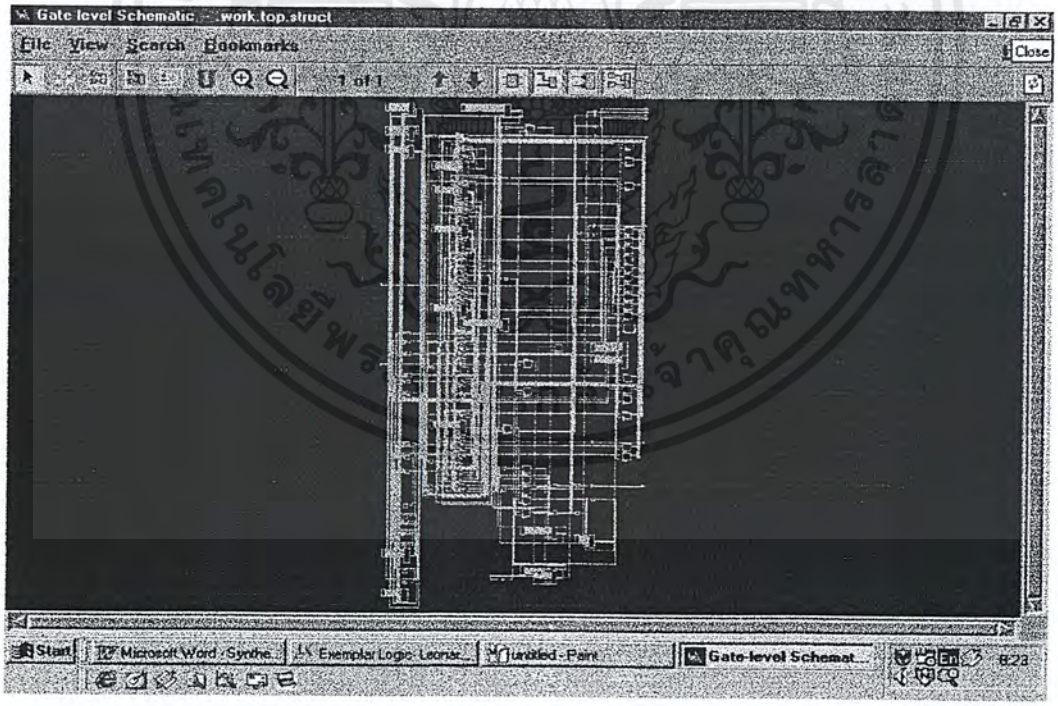
โปรแกรม Leonardo Spectrum อนุญาตให้ผู้ใช้สามารถกำหนดค่า Timing Constraint ของวงจรได้ หากวงจรที่สร้างไม่สามารถบรรจุลงใน ฮาร์ดแวร์ที่กำหนดได้โปรแกรมจะแจ้งให้ทราบและพยายามแมปกับอุปกรณ์ตัวที่มีความจุสูงกว่าซึ่งอยู่ในตระกูลเดียวกันได้



รูปที่ 4.13 แสดงหน้าจอของโปรแกรม Leonardo Spectrum



รูปที่4.14 แสดงภาพของ Synthesis ออกมาเป็นโมดูล



รูปที่ 4.15 แสดงถึงวงจรระดับ Gate-Level ของทีวีแพทเทอร์นเจนเนเรเตอร์

โปรแกรม Xilinx Foundation

ใช้ในการทำ Place and Rounte เพื่อMAP วงจรที่ Synthesis แล้วลงบนชิฟ FPGA ซอฟท์แวร์ตัวนี้จะประกอบไปด้วยโปรแกรมการใช้งานหลายโมดูลด้วยกัน อันได้แก่ Design Manager ,Flow Engine, Timing Analyzer ,Prom File Formatter, Hardware Debugger, EPIC Design Editor และ JTAG Programe

โปรแกรม Design Manager

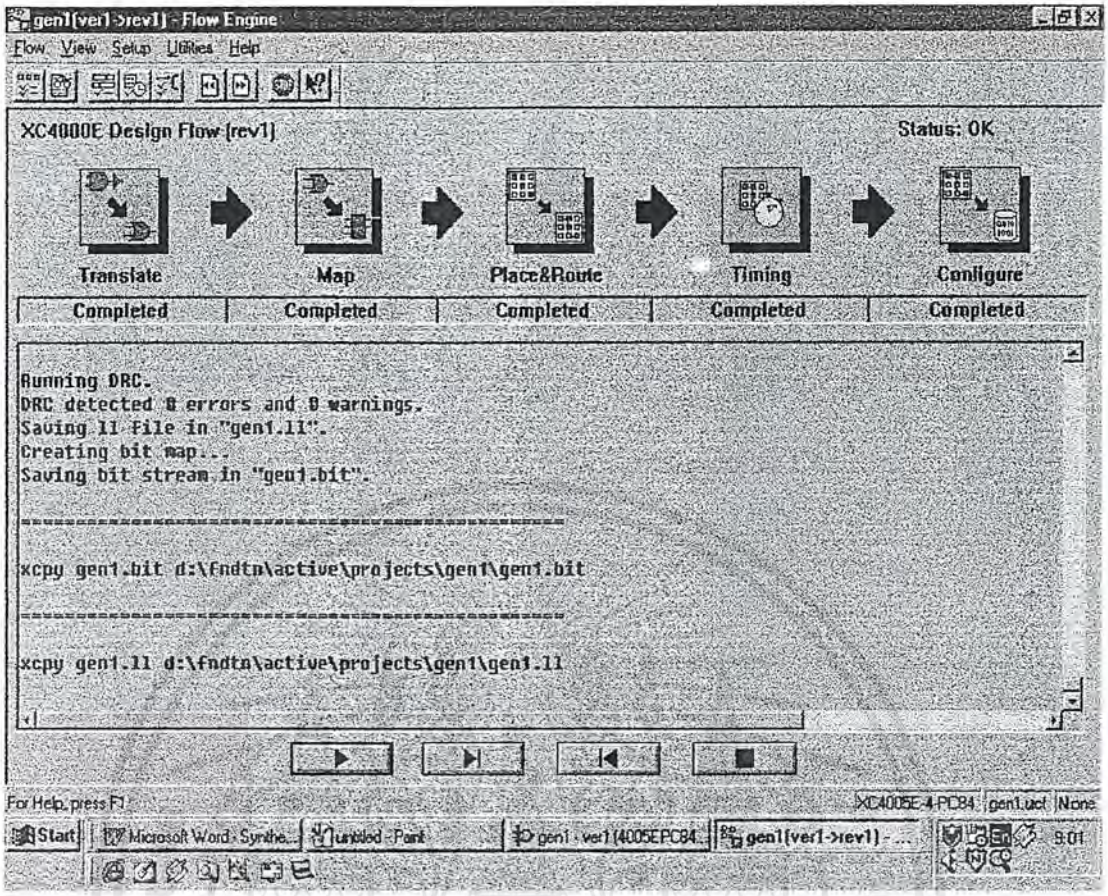
ในการจัดการกับตัว Design -ของผู้ใช้อันได้แก่

--อนุญาตให้สามารถสร้างตัว Design ได้หลายๆ เวอร์ชั่นเพื่อให้ง่ายต่อการบริหารการจักการกับตัวDesign และยังอนุญาตให้ในแต่และเวอร์ชั่นนั้นสามารถสร้างเป็นหลาย Revision ได้

--ใช้ในการเรียกโปรแกรมอื่น ๆ อันได้แก่ Flow Engine, Timing Analyzer ,Prom File Formatter, Hardware Debugger, EPIC Design Editor และ JTAG Programe

โปรแกรม Flow Engine

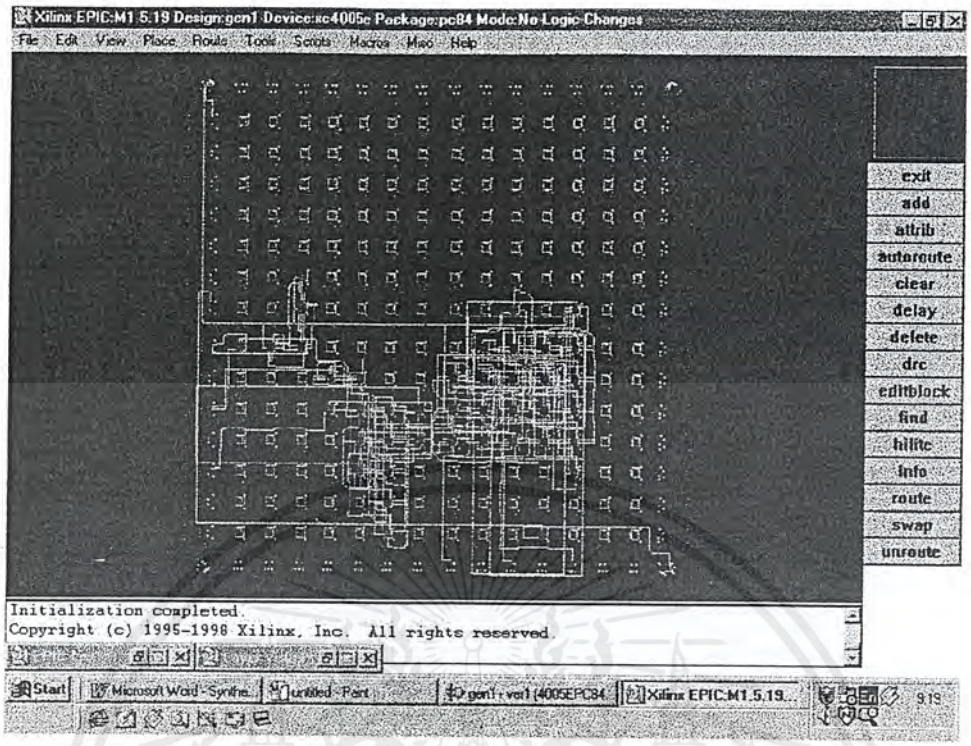
ใช้ในการทำกระบวนการ Place and RouteTimingAnalyzer และสร้างไฟล์นามสกุล .BIT ซึ่งจะ นำไปบันทึกลงใน FPGAทาง XChecker



รูปที่ 4.16 แสดงหน้าจอของโปรแกรม Xilinx Foundation

การใช้ EPIC Design Editor เพื่อพิจารณาดูวงจรภายในอุปกรณ์ FPGA

หลังจากที่ได้ทำขั้นตอนต่าง ๆ ภายใน Flow Engine แล้วเราสามารถเรียกโปรแกรมย่อยอีกตัวหนึ่งซึ่งก็คือ EPIC Design Editor เพื่อใช้ดู Layout ของวงจรภายใน FPGA โดยจะแสดงให้เห็นลอจิกบล็อกต่างและเส้นทางของสัญญาณที่เชื่อมต่อกันภายในวงจรดังรูป

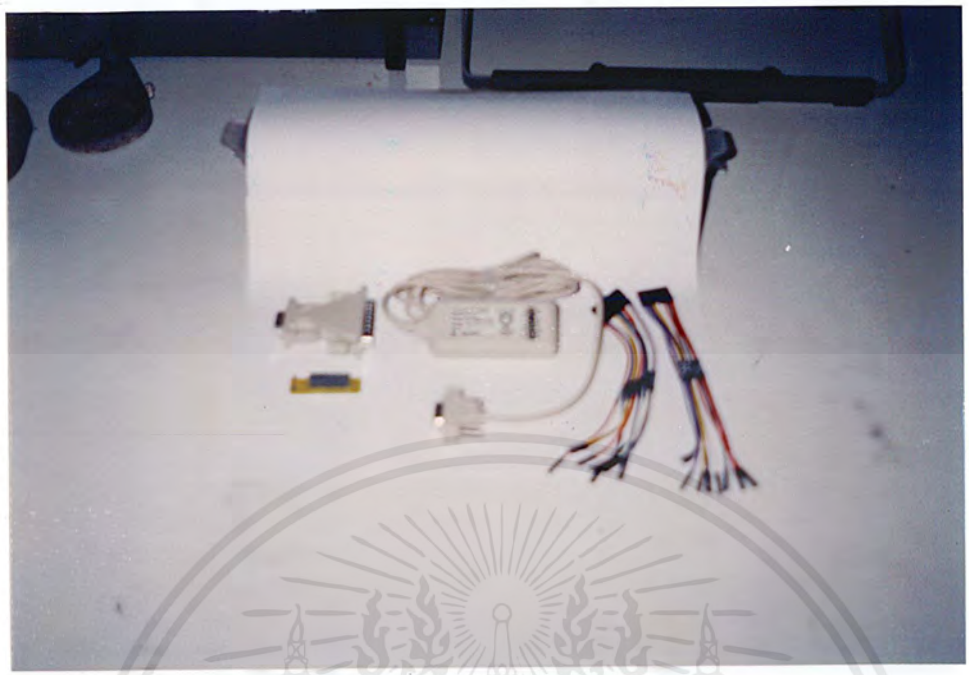


รูปที่ 4.17 EPIC Design Editor

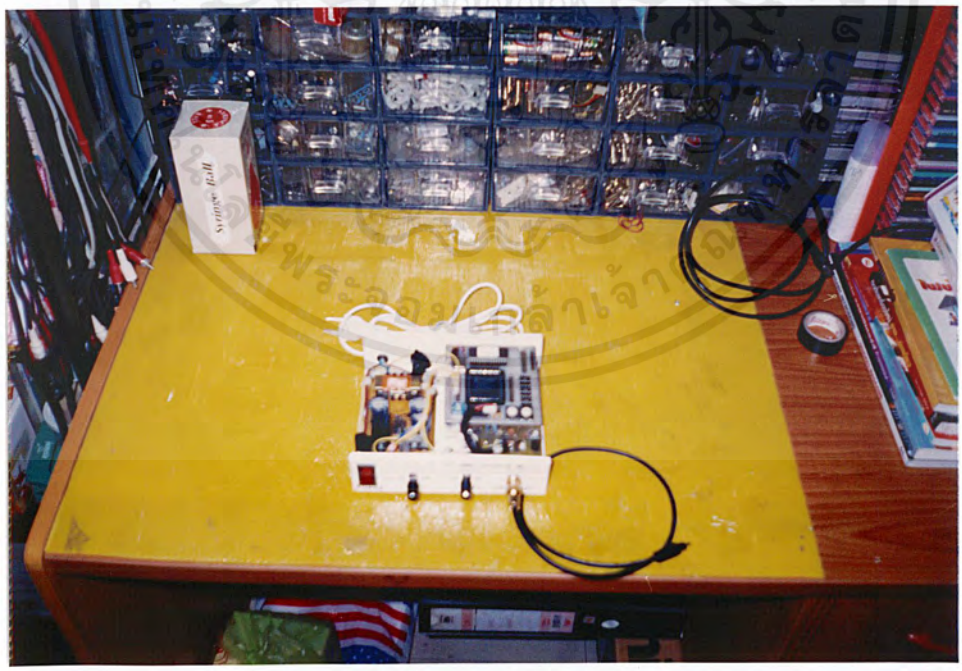
การทำการ Place and Route และการบันทึกโปรแกรมลง FPGA

เมื่อทดสอบด้วยการซิมูเลชั่นจนแน่ใจแล้วว่างานที่ Synthesis ออกมานั้นสามารถทำงานได้อย่างถูกต้องแล้วจึงนำงานที่ได้เข้าสู่ขั้นตอน Place and Route เพื่อทำการแมป(Map)งานนั้นเข้ากับ FPGA ซึ่งในการทำ Place and Route ก่อนที่จะสร้างไฟล์.BIT ควรสร้างไฟล์ .VHD ออกมาก่อนเพื่อนำไปทดสอบการทำงานกับโปรแกรม V-System ว่าวงจรที่ได้จากการทำ Place and Route นี้สามารถทำงานได้ตรงตามที่ต้องการหรือไม่ เมื่อแน่ใจว่าถูกต้องก็สั่งให้โปรแกรม Xilinx สร้างไฟล์นามสกุล.BIT เพื่อที่จะนำไปบันทึกลงใน FPGA

ในการบันทึกลง FPGAนั้นกระทำได้ 2ทางคือ 1.ทางสายเคเบิ้ล DOWNLOAD Xcheckerของ Xilinx โดยเปิดโปรแกรมส่วนของ Hardware Debugger



รูปที่ 4.18 สายดาวน์โหลดข้อมูลลงชิฟ FPGA

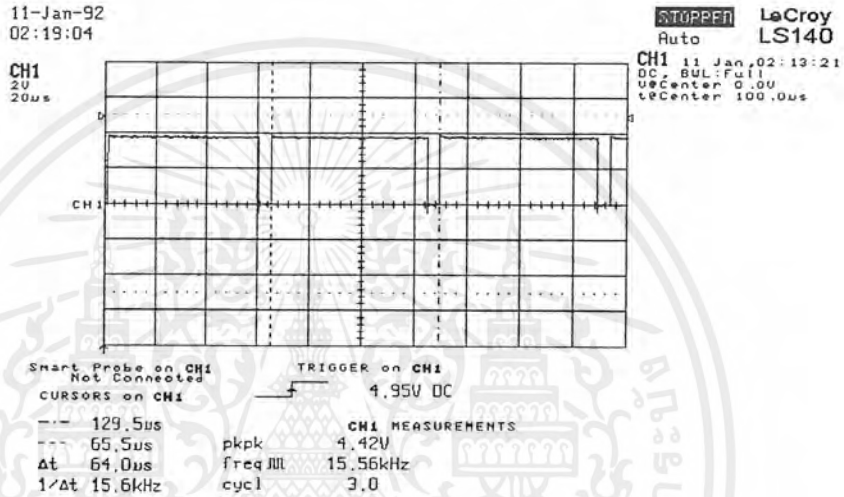


รูปที่ 4.19 เครื่องทดสอบวิดิโอแพทเทอร์เจนเนเรเตอร์

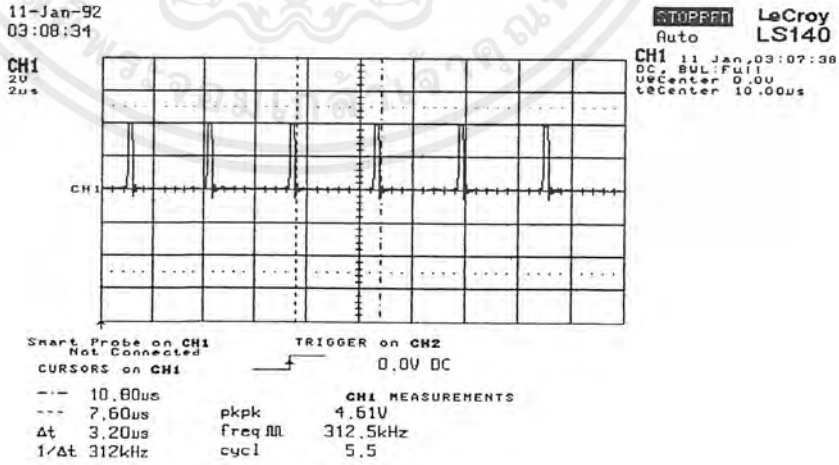
บทที่ 5
ผลการทดลอง สรุป และวิจารณ์

ผลการทดลอง

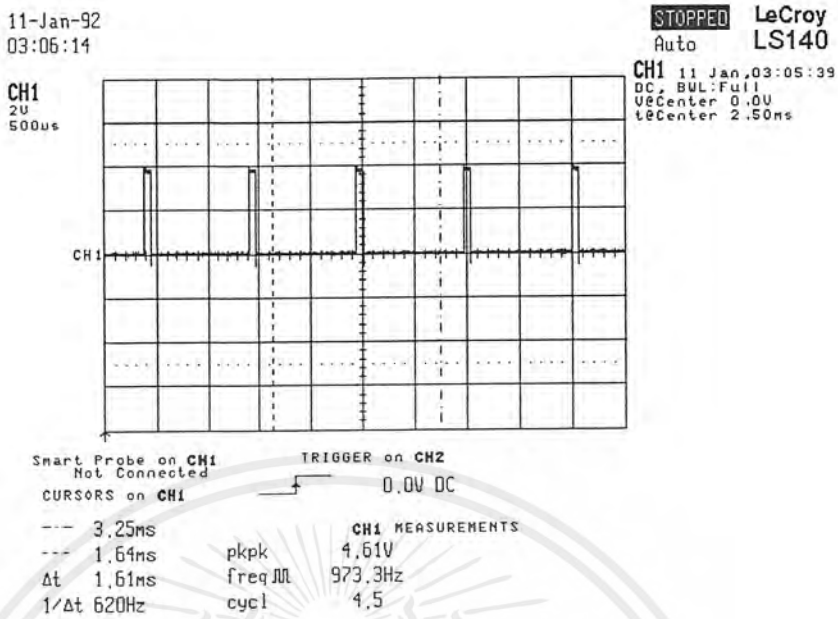
จากการที่ บันทึกโปรแกรมลงบน FPGA เรียบร้อยแล้วและได้ทำการวัดสัญญาณทีวีที่ได้ทำการออกแบบไว้ได้ดังนี้



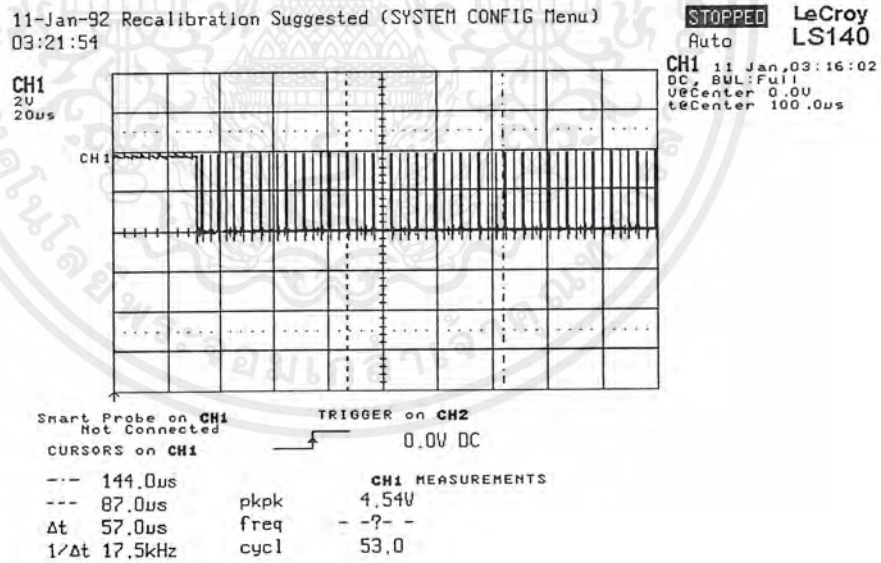
รูปที่5.1 สัญญาณซิงค์ทางแนวนอน 64µs



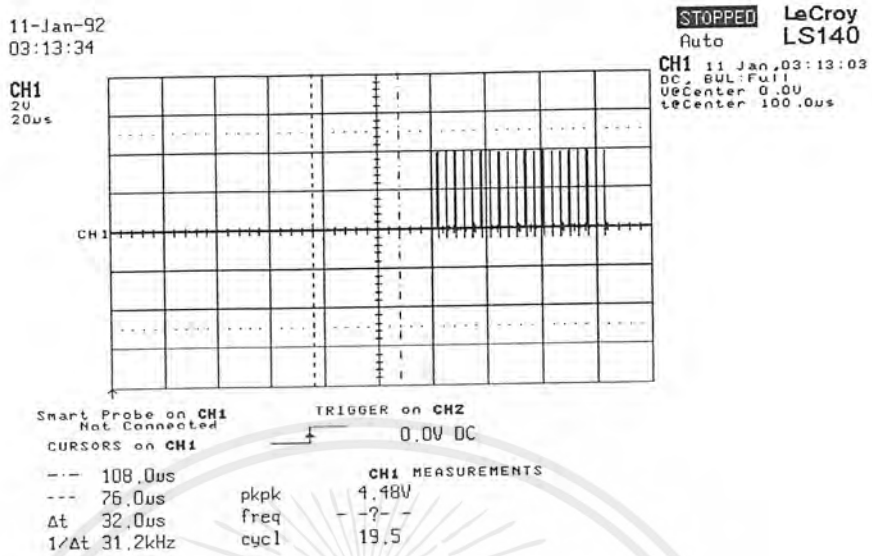
รูปที่ 5.2 สัญญาณรูปภาพแนวตั้ง



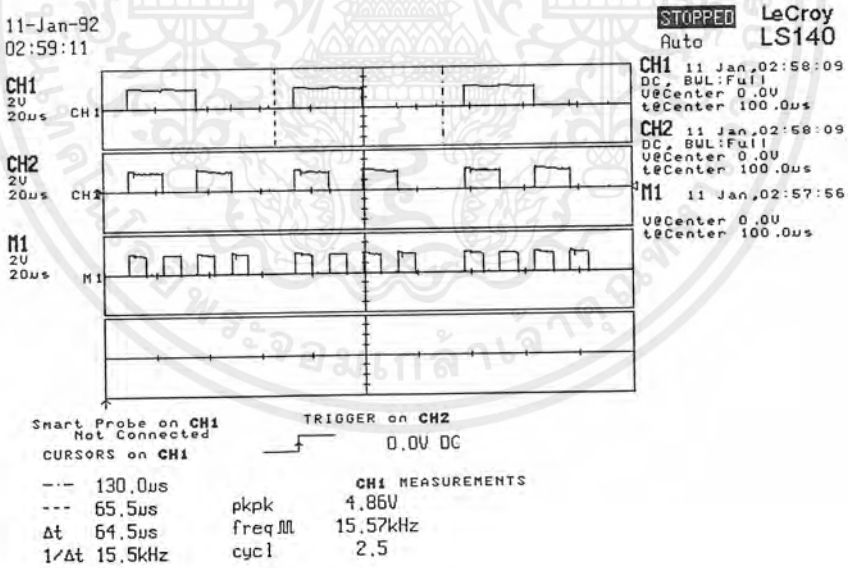
รูปที่5.3 สัญญาณรูปภาพแนวนอน



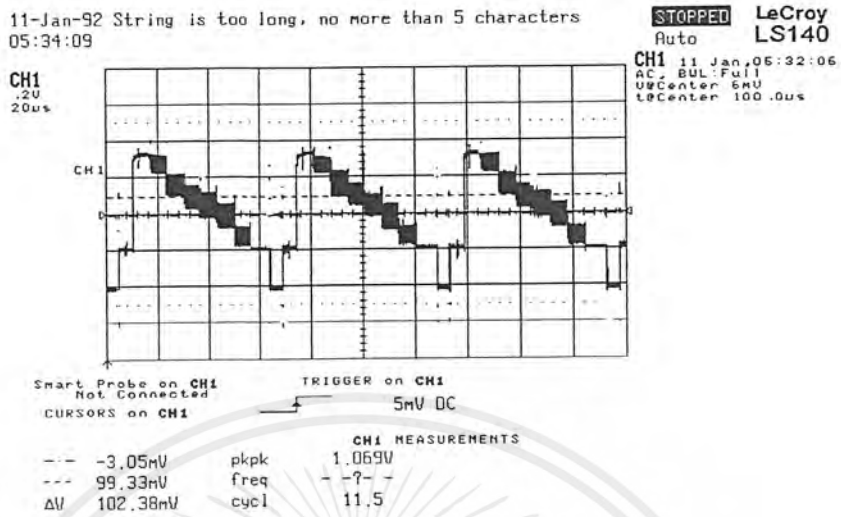
รูปที่ 5.4 สัญญาณรูปภาพตาราง



รูปที่5.5 สัญญาณรูปภาพจุด



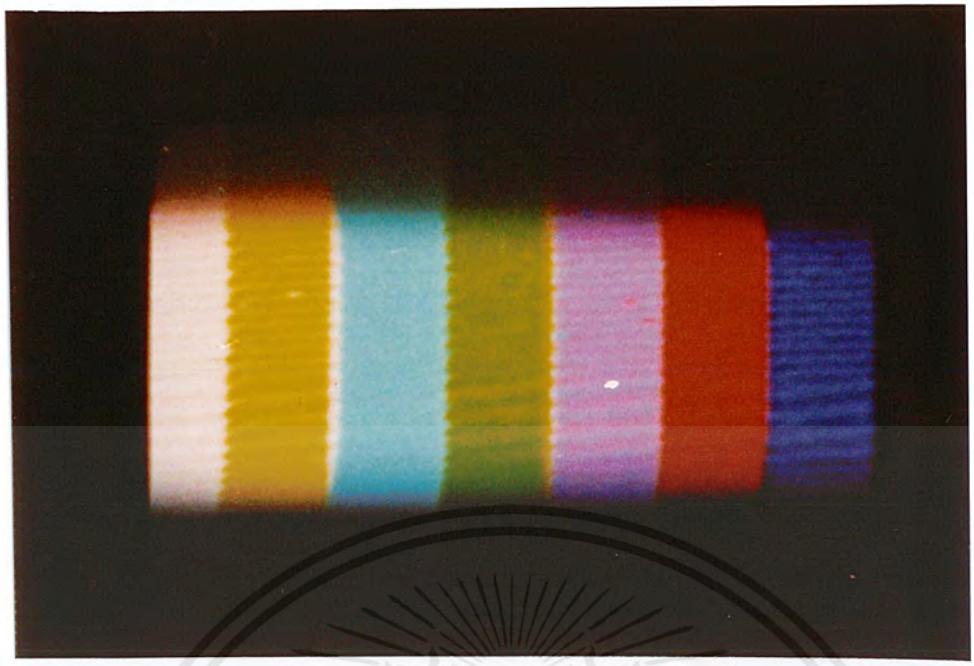
รูปที่ 5.6 สัญญาณ RGB



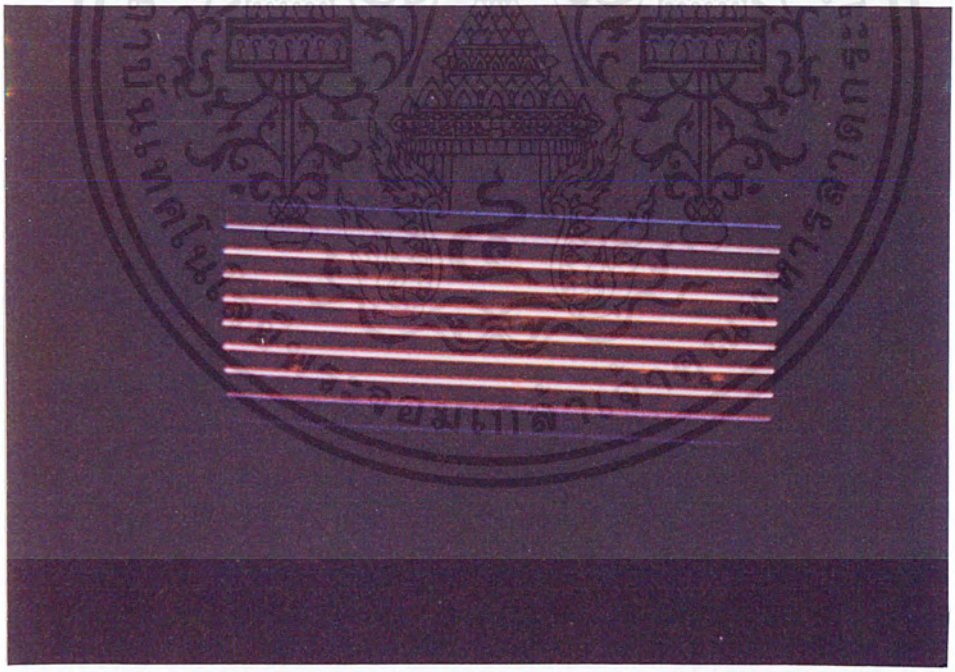
รูปที่ 5.7 สัญญาณ คอมโพสิตคัลเลอร์บาร์(Composite Color Bar)

การทดสอบการทำงานของวิดีโอแพทเทอร์เจนเนเรเตอร์

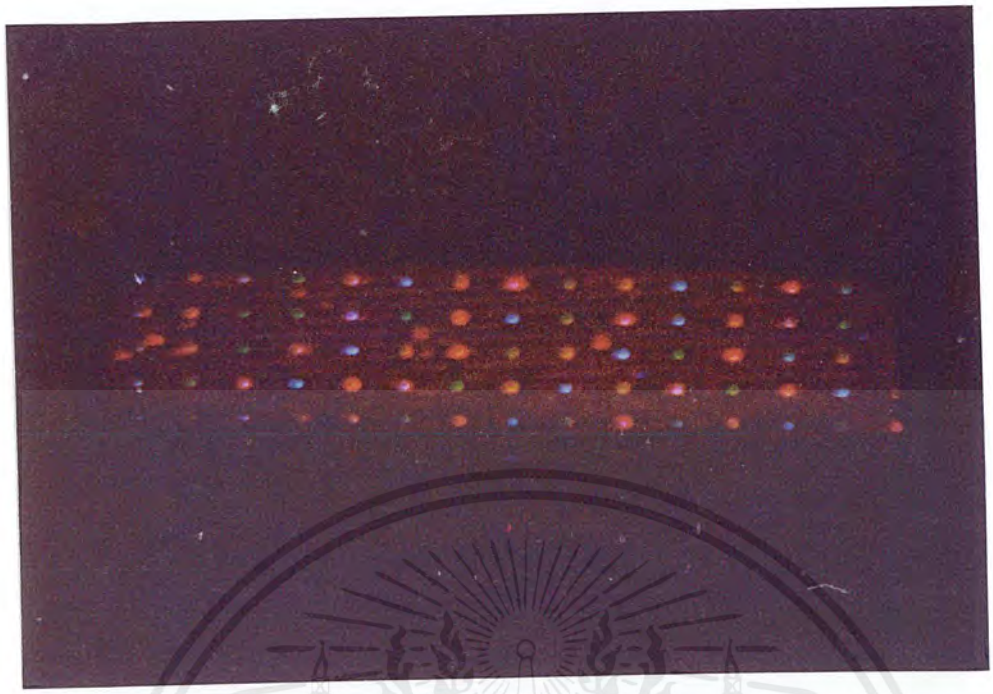
การทดสอบการทำงานของวิดีโอแพทเทอร์เจนเนเรเตอร์เราทำการทดสอบ 2 ครั้ง โดยครั้งแรกเราจะทำการทดสอบโดยการเขียน test bench ซึ่งเป็นการเขียนการทดสอบโปรแกรมเบื้องต้นซึ่ง โปรแกรมที่เราใช้จะเป็นโปรแกรม V-system หรือ Modelsim ซึ่งเมื่อทำการทดสอบเรียบร้อยแล้วต่อไปเราก็จะทำตามลำดับขั้นของโปรแกรม xilinx Fundamental 1.5 ผลสุดท้ายที่ได้จะใช้ในการการดาวน์โหลดโปรแกมที่เราเขียนด้วยภาษาวีเอชดีแอลซึ่งมีนามสกุลจุดบิตโดยในการดาวน์โหลดเราจะใช้สาย X checker ในการดาวน์โหลดโปรแกรมเมื่อทำการดาวน์โหลดเรียบร้อยแล้วทำการวัดสัญญาณที่ได้จะตรงกับที่สัญญาณที่เราได้ทำการออกแบบหรือไม่ซึ่งสัญญาณที่ได้จะมีการดีเลย์ของสัญญาณเราจะต้องทำการแก้ไขการดีเลย์โดยเราสามารถกำหนดการดีเลย์ที่ขาของชิพขอบเอฟพีจีเอหรือทำการแก้โปรแกรมใหม่เพื่อให้ได้สัญญาณที่เราต้องการ



รูปที่5.8 เป็นการทดสอบโดยใช้เครื่องรับสัญญาณโทรทัศน์ซึ่งเป็นคัลเลอร์บาร์



รูปที่ 5.9 เป็นการทดสอบโดยใช้เครื่องรับสัญญาณโทรทัศน์ซึ่งเป็นรูปทางแนวนอน



รูปที่ 5.10 เป็นการทดสอบโดยใช้เครื่องรับสัญญาณโทรทัศน์ซึ่งเป็นรูปจุด

สรุปและวิจารณ์

สรุปโดยรวมทั้งหมดในการออกแบบ

จากการออกแบบตั้งแต่การเรียนรู้ถึงสัญญาณลอจิกพื้นฐานของวิดิโอแพทเทอร์เจนเนเรเตอร์ โดยการศึกษาถึงเทคนิคการสร้างสัญญาณการสแกนทางแนวนอนที่คาบเวลา 64 μs และสัญญาณการสแกนทางแนวตั้งที่คาบเวลา 20 mS ตลอดการเรียนรู้แพทเทอร์แบบต่างๆเช่น สัญญาณคัลเลอร์บาร์ สัญญาณเส้นทางแนวนอน,สัญญาณเส้นทางแนวตั้ง,สัญญาณรูปจุด,สัญญาณรูปตารางซึ่งสัญญาณทังหมด จะใช้ตัวกำเนิดความถี่ตัวเดียวกันคือความถี่ที่ 10 MHz แล้วผ่านวงจรหารจากนั้นผ่านวงจรดีโคด แล้วจะได้สัญญาณที่ถูกต้องมา

เมื่อรู้ถึงหลักการเบื้องต้นจากวงจรลอจิกที่ใช้เราก็จะทำการเปลี่ยนเป็นการใช้ภาษาวีเอชดีแอลซึ่งเราจะต้องมองวงจรทางลอจิกออกมาเป็นบล็อคแล้วแบ่งออกเป็นส่วนย่อยๆแล้วทำการเปลี่ยนแต่ละบล็อกแทนด้วยภาษาวีเอชดีแอลซึ่งจะแทนแต่ละบล็อกด้วยโมดูลแล้วนำแต่ละบล็อกที่ได้มาเชื่อมต่อกันซึ่งการทำงานจะมีหลักการเดียวกับทางด้านลอจิกแต่การเขียนด้วยภาษาวีเอชดีแอลจะมีความง่ายในการออกแบบและใช้เวลาในการทดสอบโปรแกรมไม่นานซึ่งหากเป็นทางด้านลอจิกเราต้องมานั้งต่อวงจรเป็นการเสียเวลายิ่งถ้าเป็นวงจรขนาดใหญ่ต้องเสียเวลาในการต่อนาน

5.2ผลการออกแบบและพัฒนา

5.2.1Device utilization summary ของ FPGA

Number of External IOBs	7 out of 11 %
Flops:	0
Latches :	0
Number of Global Buffer IOB	1 out of 8 12%
Flops:	0
Latches :	0
Number of CLBs	50 out of 196 25%
Total CLB Flops	39 out of 392 9%
4 input LUTS:	88 out of 392 22 %
3 input LUTS:	0 out of 392 22 %
Number of PRI-CLKs	1 out of 4 25%
Number of SEC-CLKs	2 out of 4 50%
Number of STARTUOs	1 out of 1 100%

5.2.2 Timming summary ของ FPGA

The Average Connection Delay for this designed is: 2.543 ns

The Average Connection Delay on critical nets is: 0.000 ns

The Maximun Pin Delay is 13.219 ns

The Average Connection Delay on the Worst Nets is 8.926 ns

Desigen statistics:

Minumum period :31.5 uS

Maximum frequency :31.7 MHz

5.3 ข้อแนะนำ

หากต้องการพัฒนาโครงงานนี้สามารถ ทำการออกแบบเพื่อเติมในส่วนของโปรแกรมเพื่อให้มีหลายรูปแบบของสัญญาณทดสอบการหาข้อมูลส่วนมากจะหาจากหนังสืออิเลคทรอนิกส์การใช้งานและการต่อวงจรต้องระวังอย่าให้ชิพขอบเอฟพีจีเอร้อน

ซึ่งอาจทำให้เกิดความเสียหายแก่ตัวชิพได้ ส่วนการยืมชิพที่ใช้เพื่อทดลองสามารถทำการขอยืมและสามารถหาข้อมูลได้ที่เนคเทค(ลาดกระบัง)

บรรณานุกรม

[1] สมศักดิ์ เตชะเศรษฐ์ธณะ (อ.เกี๋ย) : "ทฤษฎีและปฏิบัติโทรทัศน์สีระบบ PAL", บริษัท ซีเอ็ดยูเคชั่นจำกัด

[2] ดร.ธวัช เมฆสวรรค์ และ นาย โยชิคะชี ซาวามูระ : " Textbook Of Color Television Engineering ", บริษัท ซีเอ็ดยูเคชั่นจำกัด

[3] Stefan Sjoholm and Lennart Lindth : " VHDL for Designers " ,Prentice Hall Europe , 1997

[4] R.R. Gulati : "Modern Television Praetile " ,Wiley Eastren Limited, 1991

[5] Frank A. Scarpino : "VHDL AND AHDL DIGITAL SYSTEM IMPLEMENTATION" ,Prentice Hall PTR ,1998

ภาคผนวก

ภาคผนวก ก

โปรแกรม VHDL ที่ใช้ในการสร้างวงจร

```
library ieee;
use ieee.std_logic_1164.All;
use ieee.std_logic_unsigned.all;
entity countP1 is
      port (    clk2  : in std_logic;
                reset  : in std_logic;
                 Q_1  : out std_logic_vector(5 downto
0)
               );
      end countP1;
      architecture behav of countP1 is
         signal count_1:std_logic_vector(5 downto 0);
      begin
         process(clk2,reset)
            begin
             if reset='0' then
            count_1<= (others=>'0');
             elsif (clk2'event and clk2='1') then
            count_1<=count_1 +"00001";
              end if;
          end process;
            Q_1<=count_1 ;
      end behav;


library ieee;
use ieee.std_logic_1164.All;
use ieee.std_logic_unsigned.all;
entity logic_gate is
      port(V1    : in std_logic;
           V2    : in std_logic;
           V3    : in std_logic;
           V4    : in std_logic;
                VL_1   : out std_logic
            );
 end logic_gate;
      architecture  dataflow of logic_gate is
      begin
           VL_1<=(not(not((V1 and V2)and V3)))and (n
ot V4);
      end dataflow ;


      library ieee;
```

```
use ieee.std_logic_1164.All;
use ieee.std_logic_unsigned.all;
entity RGB_gen is
      port  (reset  : in std_logic;
               clk3  : in std_logic;
             hsync  : buffer std_logic ;
             vsync  : out std_logic ;
                 R   : out std_logic;
                 G   : out std_logic;
                 B   : out std_logic
             );
      end RGB_gen;

architecture RGB_1 of RGB_GEN is
signal hcnt : std_logic_vector (9 downto 0);
signal Vcnt : std_logic_vector (8 downto 0);

begin
A:process(CLk3,reset)
 begin
    if reset='0' then
      hcnt<=(others=>'0');
        elsif (clk3'event and clk3='1')then
          if hcnt < 641 then
            hcnt <= hcnt+"0000000001";
          else
            hcnt <=(others=>'0');
          end if;
    end if;
end process A ;

process(hsync,reset)
begin
if reset ='0' then
      vcnt <= (others=>'0');
   elsif (hsync'event and hsync='1')then
      if vcnt <313 then
        vcnt <=vcnt+"000000001";
      else
        vcnt<=(others=>'0');
      end if;
end if;
end process ;

C:process(CLK3,reset)
```

```
begin
  if reset='0' then
    hsync <= '1';
      elsif (clk3'event and clk3 ='1') then
     if (hcnt>=0 and hcnt<48 )then
       hsync <='0';
     else
       hsync <='1';
     end if;
      end if;
      end process c;

      D:process(hsync,reset)
 begin
     if reset='0' then
      vsync<='0';
      elsif (hsync'event and hsync='1')then
            if (vcnt>=0 and vcnt<3 )then
           vsync <= '0';
           else
           vsync <= '1';
            end if;
          end if;
      end process D;
           G<='1' when ( hcnt>=105 and hcnt
<366)
                    and(vcnt>=20
and vcnt<313)
                 else '0';
           R<='1' when ( (hcnt>=105 and hcn
t<236 ) or
              (hcnt >=365 and hcnt<496 ) )
and ( vcnt>=20 and vcnt<313 )
                        else '0';
           B<='1' when ( (hcnt>=105 and hcn
t<171 ) or ( hcnt>=235 and hcnt<301 ) or
             ( hcnt>=365 and hcnt<431 ) or
( hcnt>=495 and hcnt<561) )
              and (vcnt>=20 and vcnt<313)
                   else '0';
  end RGB_1;


library ieee;
use ieee.std_logic_1164.all;
```

```
use ieee.std_logic_unsigned.all;
 entity part1 is
         port (  clk4: in std_logic;
          reset: in std_logic;
             Q   : out std_logic_vector (3 downto 0)
          );
  end part1;

       architecture part_1 of part1 is
            signal count: std_logic_vector(3 downto 0
);
       begin
          process(clk4,reset)
           begin
           if reset='0' then
              count<=(others=>'0');
            elsif (clk4'event and clk4='1') then
             count<= count+"0001";
           end if;
          end process;
            Q<=count;
          end part_1;

library ieee;
use ieee.std_logic_1164.All;
use ieee.std_logic_unsigned.all;
 entity part2 is
   port (qi0: in std_logic;
          qi1: in std_logic;
          qi2: in std_logic;
          qi3: in std_logic;
   Hsync_2: in std_logic;
    Vsync_2: in std_logic;
        VL_2: in std_logic;
          dot: out std_logic;
           VL: out std_logic;
          XH: out std_logic;
          HL: out std_logic;
   mixsync: out std_logic
          );
  end part2;
       architecture flow of part2 is
          begin
      dot         <=((not(qi0 and(qi1 and qi2)))nor qi3
)and VL_2 ;
```

```
      XH       <=(not(((not((qi1 and qi2)and qi0))nor
 Qi3)nor VL_2));
      VL       <=not(not VL_2);
      HL       <=(not(qi0 and(qi1 and qi2)))nor qi3;
      mixsync <= (hsync_2 and vsync_2);
          end flow;

library ieee;
use ieee.std_logic_1164.All;
use ieee.std_logic_unsigned.all;
entity pattren is
     port(      clk:in std_logic;
           RESET:in std_logic;
            dot   :out std_logic;
            VL    :out std_logic;
            XH    :out std_logic;
      HL    :out std_logic;
     mixsync     :out std_logic;
           r       :out std_logic;
           g       :out std_logic;
           b       :out std_logic;
      div32       :out std_logic
          );
 end pattren;

     architecture structural of pattren is

     component countp1
          port(CLK2        : in std_logic;
               reset       : in std_logic;
                Q_1        : out std_logic_ve
ctor(5 downto 0)
               );
     end component;

     component logic_gate
          port(  v1    :in std_logic;
                 v2    :in std_logic;
                 v3    :in std_logic;
                 v4    :in std_logic;
                 VL_1 :out std_logic
                );
     end component;

     component rgb_gen
```

```
        port (CLK3    : in std_logic;
              reset   : in std_logic;
              Hsync   : buffer std_logic;
              Vsync   : out std_logic;
              R       :out std_logic;
              G       :out std_logic;
              B       :out std_logic
                  );
    end component;

    component part1
        port(clk4 : in std_logic;
             reset: in std_logic;
                 Q: out std_logic_vector(3 do
wnto 0)
              );
        end component;

    component part2
        port(qi0  :in std_logic;
        qi1  :in std_logic;
        qi2  :in std_logic;
        qi3   :in std_logic;
         VL_2  :in std_logic;
         Hsync_2 :in std_logic;
         Vsync_2 :in std_logic;
        dot: out std_logic;
        XH: out std_logic;
        VL : out std_logic;
        HL: out std_logic;
        mixsync: out std_logic
        );
    end component;

        signal net1: std_logic;
        signal net2: std_logic;
        signal net3: std_logic;
        signal net4: std_logic;
        signal net5: std_logic;
        signal net6: std_logic;
        signal net7: std_logic;
        signal net8: std_logic;
        signal net9: std_logic;
        signal net10: std_logic;
        signal net11: std_logic;
```

```
begin
c1:countp1 port map (clk2=>clk
            ,reset=>reset
            ,q_1(0)=>open
            ,q_1(1)=>net1
            ,q_1(2)=>net2
            ,q_1(3)=>net3
            ,q_1(4)=>net4
            ,q_1(5)=>div32
            );

c2:logic_gate port map(v1=>net1
            ,v2=>net2
            ,v3=>net3
            ,v4=>net4
            ,VL_1=>net7
            );

c3:rgb_gen port map(clk3=>clk
            ,reset=>reset
            ,Hsync=>net5
            ,Vsync=>net6
            ,R=>R
            ,B=>B
            ,G=>G
            );

c4:part1 port map(clk4=>net5
            ,reset=>reset
            ,Q(0)=>net8
            ,Q(1)=>net9
            ,Q(2)=>net10
            ,Q(3)=>net11
            );

c5:part2 port map(qi0=>net8
            ,qi1=>net9
            ,qi2=>net10
            ,qi3=>net11
            ,VL_2=>net7
            ,Hsync_2=>net5
            ,Vsync_2=>net6
            ,dot=>dot
            ,VL=>VL
```

```
                ,XH=>XH
                ,HL=>HL
                ,mixsync=>mixsync
                );

    end structural;

library ieee;
use ieee.std_logic_1164.All;
use ieee.std_logic_unsigned.all;
entity TOP is
   port(clock:in std_logic;
        Reset:in std_logic;
           dot:out std_logic;
            VL:out std_logic;
           XH:out std_logic;
           HL:out std_logic;
    mixsync:out std_logic;
             r:out std_logic;
            g:out std_logic;
            b:out std_logic;
       div32:out std_logic
        );
  end TOP;
       architecture struct of TOP is
      component pattren
           port(clk:in std_logic;
               reset:in std_logic;
                 dot:out std_logic;
                 VL:out std_logic;
                     XH:out std_logic;
                    HL:out std_logic;
                    mixsync:out std_logic;
           r:out std_logic;
           g :out std_logic;
           b :out std_logic;
           div32:out std_logic
           );
      end component;
      begin

      k2:pattren port map (
                           clk=>clock
                    ,reset=>reset
                    ,dot=>dot
```

```
                    ,VL=>VL
                  ,div32=>div32
                  ,HL=>HL
                  ,XH=>XH
                   ,mixsync=>mixsync
                  ,r=>r
                  ,g=>g
                   ,b=>b
                  );
      end struct;

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity switch is
        port (clk5: in std_logic;
         switchin : in std_logic;
        reset : in std_logic;
          dataout  : out std_logic_vector (2 downto 0
)
         );
           end switch;

      Architecture switch_1 of switch is
      type state_type is (s0,s1,s2,s3,s4);
           signal state :state_type;


      begin
      process(clk5,reset)
      begin
      if reset='0'then
              state <= S0 ;
        dataout<=(others=>'0');
      elsif clk5'event and clk5='1' then
         case state is
      when S0=> if switchin='0' then
               state<= S1;
      end if;
              dataout<="000";
      when S1=> if switchin='1' then
          state<= S2;
      end if;
          dataout<="010";
      when S2=> if switchin='0' then
```

```
            state<= S3;
        end if;
            dataout<="100";
        when S3=> if switchin='1' then
            state<= S4;
        end if;
            dataout<="110";
        when S4=> if switchin='0' then
            state<= S0;
        end if;
            dataout<="111";

          end case;

        end if;
            end process;

            end switch_1;

        library ieee;
        use ieee.std_logic_1164.all;
        use ieee.std_logic_unsigned.all;
        Entity mux4 is
        port (      dot2     : in std_logic;
            hl2      : in std_logic;
            vl2      : in std_logic;
            xh2      : in std_logic;
            sel_1    : in std_logic_vector(2 downto 0)
;
            q1       : out std_logic_vector(2 downto
 0)
           );
        end mux4;
        Architecture mux4_1 of mux4 is
        begin
        process(dot2,hl2,vl2,xh2,sel_1)
        begin
        case sel_1 is
            when"000"     => q1 <= dot2&dot2&dot2;
            when"010"     => q1 <= hl2 &hl2&hl2;
            when"100"     => q1 <= vl2 &vl2&vl2;
            when"110"     => q1 <= xh2 &xh2&xh2;
            when others    => q1 <= "000";
        end case;
        end process;
```

```
    end mux4_1;

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity mux2 is
    port ( RGB    : in std_logic_vector(2 downto 0);
           selec  : in std_logic;
           data_1 : in std_logic_vector(2 downto 0
);
           q2     :out std_logic_vector(2 downt
o 0)
         );
    end mux2;
    architecture mux2_1 of mux2 is
      begin
        q2<= rgb when selec='1' else data_1;
    end mux2_1;

    library ieee;
    use ieee.std_logic_1164.All;
    use ieee.std_logic_unsigned.all;
    entity syncgen is
        port (clock   : in std_logic;
              reset   : in std_logic;
            switchin  : in std_logic;
           selec      : in std_logic;
           mixsync    : out std_logic;
           RGB_out    : out std_logic_vector(2 down
to 0)
           );
    end syncgen ;
    architecture syncgen_1 of syncgen is
    component top
        port(clock  :in std_logic;
             Reset  : in std_logic;
                dot  :out std_logic;
                 VL  :out std_logic;
                 XH :out std_logic;
                 HL  :out std_logic;
            mixsync :out std_logic;
                 r    :out std_logic;
                 g    :out std_logic;
                 b    :out std_logic;
               div32:out std_logic
```

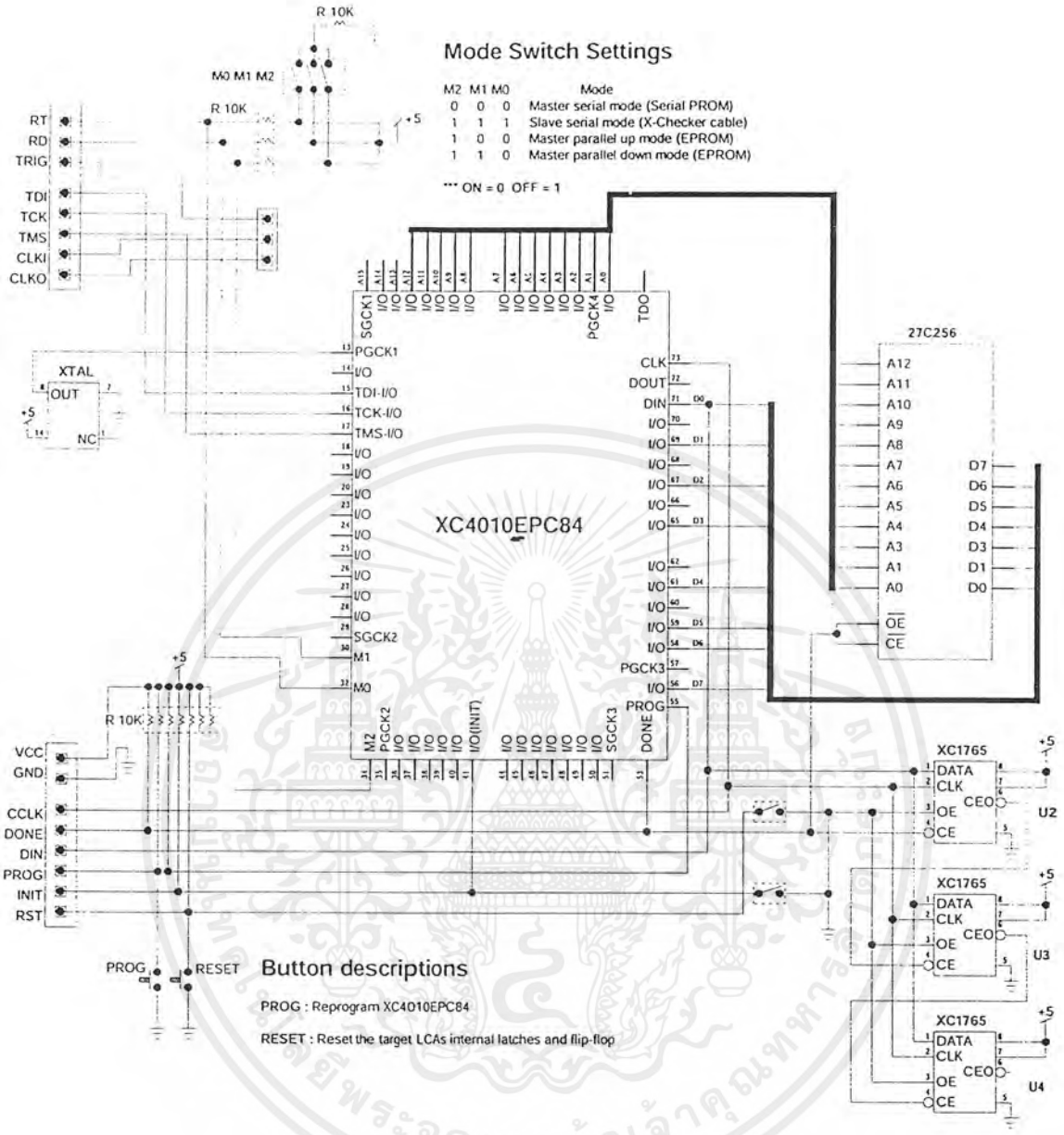
```
                    );
    end component;

    component switch
        port (clk5  : in std_logic;
        switchin   : in std_logic;
            reset  : in std_logic;
              dataout   : out std_logic_vecto
r (2 downto 0)
        );
    end component;

    component mux4
        port (dot2        : in std_logic;
          hl2     : in std_logic;
          vl2     : in std_logic;
          xh2         : in std_logic;
          sel_1       : in std_logic_vector(2 down
to 0);
          q1          : out std_logic_vector(2 d
ownto 0)
          );
    end component;

    component mux2
        port (RGB       : in std_logic_vector(2
downto 0);
          selec      : in std_logic;
          data_1     : in std_logic_vector(2 downto
 0);
          q2         :out std_logic_vector(2 dow
nto 0)
        );
    end component;
    signal net14 : std_logic;
    signal net15 : std_logic;
    signal net16 : std_logic;
    signal net17 : std_logic;
    signal net18 : std_logic;
    signal net19 : std_logic;
    signal net20 : std_logic;
    signal net21 : std_logic;
    signal net12 : std_logic_vector(2 downto 0);
    signal net13 : std_logic_vector(2 downto 0);
    begin
```

```
    t1: top port map (clock=>clock
                ,reset=>reset
                ,mixsync=>mixsync
                ,div32=>net21
                ,dot=>net14
                ,xh=>net15
                ,hl=>net16
                ,vl=>net17
                ,R=>net18
                ,G=>net19
                ,B=>net20
                 );
    t2: switch port map ( clk5=>net21
                ,reset=>reset
                ,switchin=>switchin
                ,dataout(0)=>net12(0)
                ,dataout(1)=>net12(1)
                ,dataout(2)=>net12(2)
                );
    t3: mux4 port map ( dot2=>net14
                ,xh2=>net15
                ,hl2=>net16
                ,vl2=>net17
                ,sel_1(0)=>net12(0)
                ,sel_1(1)=>net12(1)
                ,sel_1(2)=>net12(2)
                ,q1(0)=>net13(0)
                ,q1(1)=>net13(1)
                ,q1(2)=>net13(2)
                );
    t4: mux2 port map  ( RGB(0)=>net18
                ,RGB(1)=>net19
                ,RGB(2)=>net20
                ,data_1(0)=>net13(0)
                ,data_1(1)=>net13(1)
                ,data_1(2)=>net13(2)
                ,selec=>selec
                ,q2(0)=>RGB_out(0)
                ,q2(1)=>RGB_out(1)
                ,q2(2)=>RGB_out(2)
                );
    end syncgen_1;

library ieee;
use ieee.std_logic_1164.All;
```

```
use ieee.std_logic_unsigned.all;
entity bigtop is
        port(osc,reset: in std_logic;
        switchin : in std_logic;
       mixsync : out std_logic;
          selec :in std_logic;
      RGB_out : out std_logic_vector(2 downto 0)
       );
end bigtop;
architecture bigtop_1 of bigtop is
component  syncgen
port (clock,reset : in std_logic;
       switchin: in std_logic;
       selec: in std_logic;
       mixsync : out std_logic;
       RGB_out: out std_logic_vector(2 downto 0)
       );
end component;
begin
a1: syncgen port map (clock=>osc
             ,reset=>reset
             ,switchin=>switchin
             ,selec=>selec
             ,mixsync=>mixsync
             ,RGB_out=>RGB_out
             );
end bigtop_1;
```

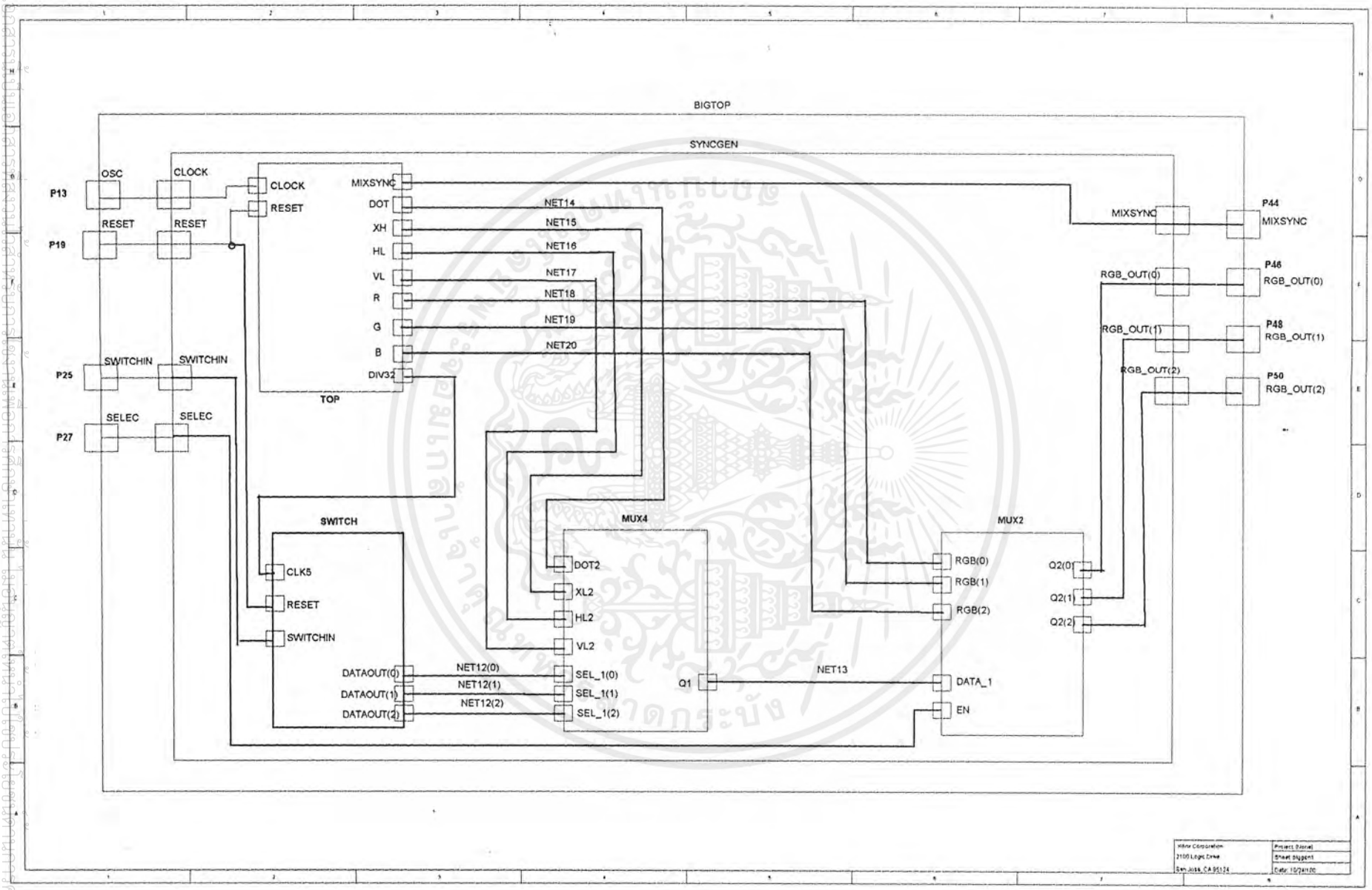
ภาคผนวก ข

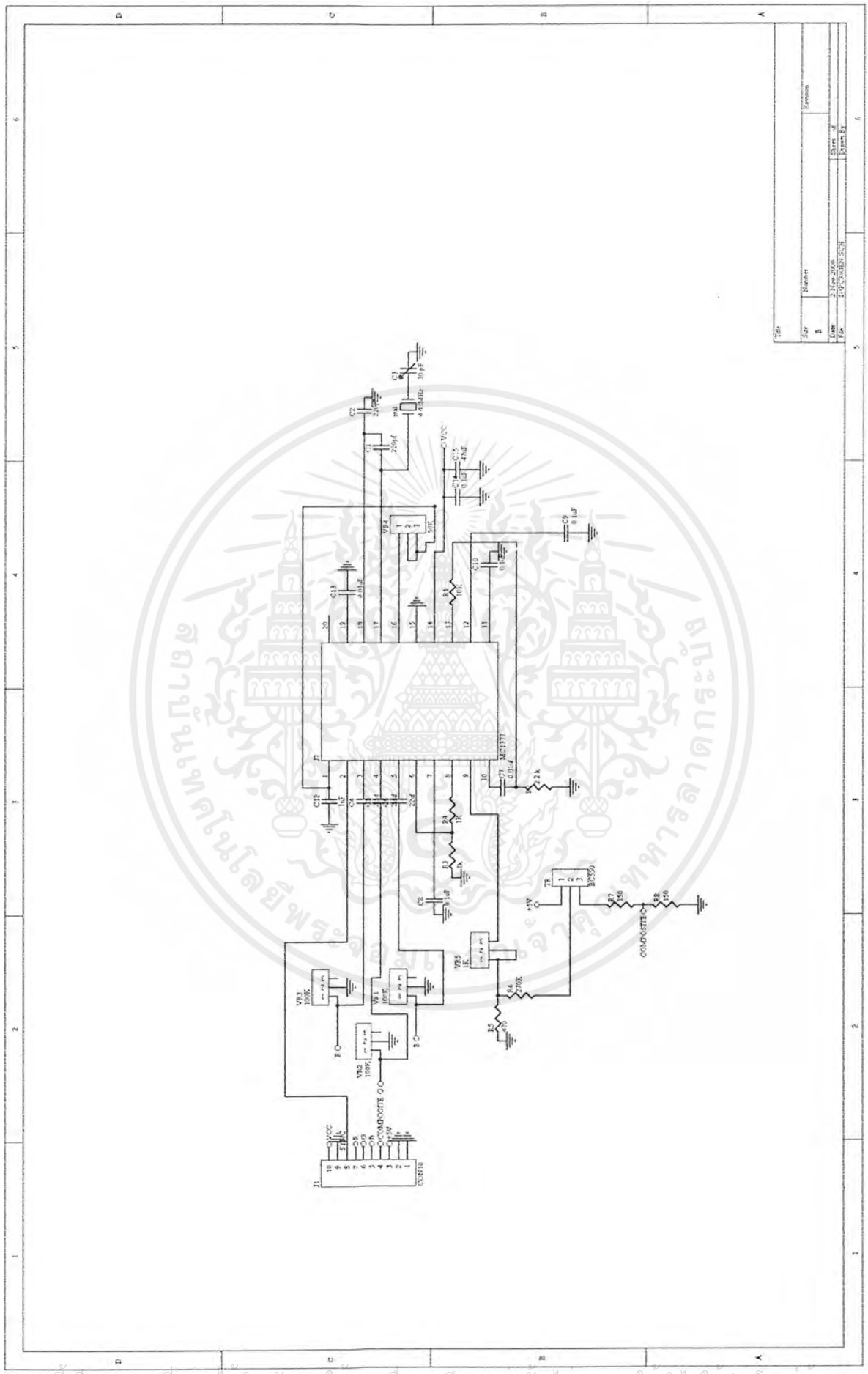
Schematic of Xilinx84 Demo Borad



ภาคผนวก ค

Schematic of TV Pattern Genarator


BIGTOP
SYNCGEN
TOP
SWITCH
MUX4
MUX2
OSC
CLOCK
RESET
SWITCHIN
SELEC
P13
P19
P25
P27
MIXSYNC
DOT
XH
HL
VL
R
G
B
DIV32
NET14
NET15
NET16
NET17
NET18
NET19
NET20
P44
P46
P48
P50
RGB_OUT(0)
RGB_OUT(1)
RGB_OUT(2)
CLK5
DOT2
XL2
HL2
VL2
SEL_1(0)
SEL_1(1)
SEL_1(2)
DATAOUT(0)
DATAOUT(1)
DATAOUT(2)
NET12(0)
NET12(1)
NET12(2)
Q1
NET13
RGB(0)
RGB(1)
RGB(2)
DATA_1
EN
Q2(0)
Q2(1)
Q2(2)



MOTOROLA

MC1377

Advance Information

COLOR TELEVISION RGB to PAL/NTSC ENCODER

COLOR TELEVISION
RGB to PAL/NTSC ENCODER

SILICON MONOLITHIC
INTEGRATED CIRCUIT

. . . an integrated circuit used to generate a composite TV signal from baseband red, blue, green and sync inputs. The MC1377 has color subcarrier oscillator, voltage controlled 90° phase shifter, two DSB suppressed carrier chroma modulators, RGB input matrices and blanking level clamps. It can be operated with very few external parts, but has the pinouts for a fully implemented, top quality composite signal. It is ideal for encoding signals from color cameras and graphics generators.

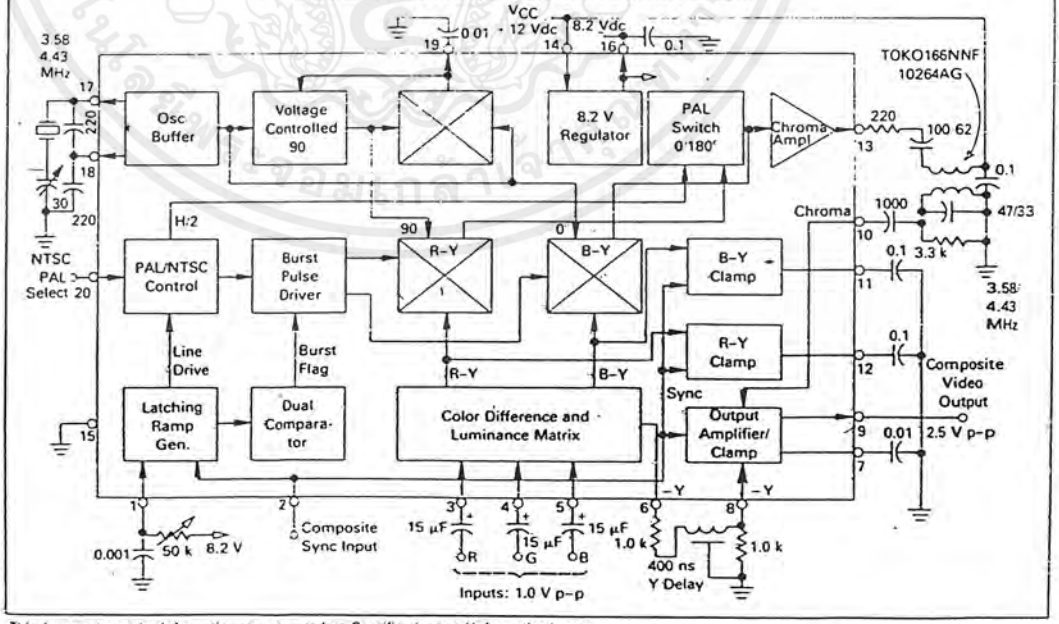
- Reference Oscillator Self-Contained Or Externally Driven
- Nominal 90° ±3.0° Axes Are Optionally Trimmable
- Simple PAL/NTSC Switch
- Luminance And Chroma Channels Can Accept Delay Line/Bandpass Elements Or Direct Connection
- Provides dc Reference To Permit Direct Drive To RF Modulator

P SUFFIX
PLASTIC PACKAGE
CASE 738-03

FN SUFFIX
PLASTIC PACKAGE
CASE 775-02
PLCC-20

ORDERING INFORMATION		
Device	Temperature Range	Package
MC1377P MC1377FN	0–70°C	Plastic DIP PLCC-20

FIGURE 1 — BLOCK DIAGRAM AND APPLICATION CIRCUIT



This document contains information on a new product. Specifications and information herein are subject to change without notice.

MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Supply Voltage	V_{CC}	15	Vdc
8.2 Vdc Regulator Output Current	I_{REG}	- 10	mAdc
Operating Temperature	T_{AMB}	0 to - 70	°C
Storage Temperature	T_{stg}	−65 to - 150	°C
Junction Temperature	$T_{J(max)}$	150	°C
Power Dissipation, package Derate above 25°C	P_D	1.25 10	W mW/°C

RECOMMENDED OPERATING CONDITIONS

Supply Voltage	12 ± 1.2	Vdc
Sync Tip Level Sync, Blanking Level	−0.5 to + 1.0 + 1.7 to + 6.2	Vdc
Red, Green, Blue Inputs (Saturated)	1.0	V_{p-p}

ELECTRICAL CHARACTERISTICS (V_{CC} = 12 Vdc, T_A = 25°C, Circuit Of Figure 1 Unless Otherwise Noted.)

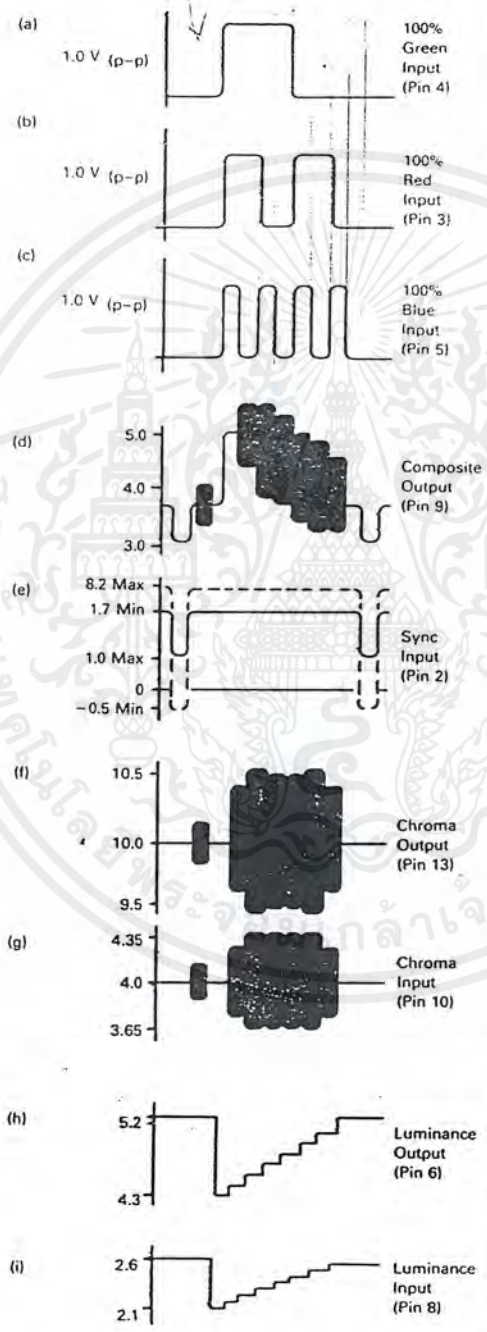
Characteristic	Pin No.	Min	Typ	Max	Unit
Supply Current	14	20	32	40	mAdc
Oscillator Amplitude	18	—	0.5	—	$V_{(p-p)}$
External Subcarrier Input (Oscillator Components Removed)	17	—	0.25	—	V_{RMS}
Subcarrier Input: Resistance Capacitance	17	— —	5.0 2.0	— —	kΩ pF
Modulation Angle (R-Y) to (B-Y)	—	85	90	95	Degrees
(R-Y) Angle Adjustment	19	—	0.25	—	Deg/μA
R, G, B Input For 100% Color Saturation	3, 4, 5	—	1.0	—	$V_{(p-p)}$
R, G, B Input: Resistance Capacitance	3, 4, 5	— —	10 2.0	—	kΩ pF
Sync Threshold (See Figure 2e)	2	—	1.7	—	V
Sync Input Resistance (Input > 1.7 V)	2	—	10	—	kΩ
Chroma Output Level At 100% Saturation	13	—	1.0	—	$V_{(p-p)}$
Chroma Output Resistance	13	—	50	—	Ω
Chroma Input Level For 100% Saturation	10	—	0.7	—	$V_{(p-p)}$
Chroma Input: Resistance Capacitance	10	— —	10 2.0	— —	kΩ pF
Composite Output, 100% Saturation (See Figure 2d): Sync	9	—	0.6	—	$V_{(p-p)}$
Luminance		—	1.4	—	
Chroma		—	1.7	—	
Burst		—	0.6	—	
Output Impedance (See Note 1)	9	—	50	—	Ω
Luminance Bandwidth (3 dB), Less Delay Line	9	—	8.0	—	MHz
Subcarrier Leakage In Output	9	—	20	—	$mV_{(p-p)}$

Note 1: Output Impedance can be reduced to less than 10Ω by using a 150Ω output load from Pin 9 to ground. Power supply current will increase to about 60 mA.

See Application Note AN932 for further information.

9

FIGURE 2 – SIGNAL VOLTAGES
(CIRCUIT VALUES OF FIGURE 1)



APPLICATION NOTES

R.G.B. Inputs should be set up to be 1.0 V p–p for fully saturated levels. This is not arbitrary, since sync and burst levels are internally fixed. The large (15 µF) input capacitors of Figure 1 are needed for the 50/60 Hz vertical component.

Subcarrier Oscillator. The internal common-collector Colpitts can be free run or it can easily be pulled in by a lightly coupled signal from a "master" into Pin 17. Also, it can be disabled entirely and a 0.25 V_{RMS} signal driven into Pin 17.

Modulator Phase Angles are quite accurately established internally. Taking (B–Y) as 0°, burst is at 180°, and the angle of (R–Y) is 90° ±3.0°. The (R–Y) angle can be "tweaked." For example, 470 kΩ from Pin 19 to ground will increase the (R–Y) to (B–Y) angle about 3.0°. Pulling Pin 19 up will decrease the angle.

Composite Output is dc referenced and can be direct coupled to an RF modulator as shown in Figure 3. In this case, the 8.2 V regulator output of the MC1377 is divided down to 5.8 V to provide the zero carrier reference to Pin 1 of the MC1374.

Burst Generation is provided by a sync triggered ramp on Pin 1 and two internal level sensors. Since the early part of this ramp is used, it is quite accurate. Fixed R–C values are feasible, as shown in Figure 3.

Sync Input can be varied over a wide latitude but neverless must be applied correctly. The typical ac coupled sync signal has very little positive value and will require a pull-up resistor to 8.2 Vdc at the input. The sync input is a 10 kΩ/10 kΩ divider in the base of a common emitter stage. For PAL operation, the correctly serrated vertical sync interval must be used, in order to continuously trigger the PAL flip-flop. "Block" vertical sync can be used for NTSC.

(R–Y)(B–Y)(–Y) signals are generated to NTSC values (±5.0%) in the input matrices. They are dc clamped at black level by a sync driven clamp. Burst amplitude is internally fixed to correspond to sync level, allowing for 3.0 dB loss in the chroma bandpass filter. If the filter is not used, as shown in Figure 3, a resistor divider should be inserted between Pin 13 and Pin 10 to provide the proper chroma level. When the chroma bandpass is not used, the (–Y) delay line should also be removed, but the 1.0 k/1.0 k divider from Pin 6 to Pin 8 should be retained.

FIGURE 3 — COUPLING THE MC1377 TO THE MC1374 RF MODULATOR

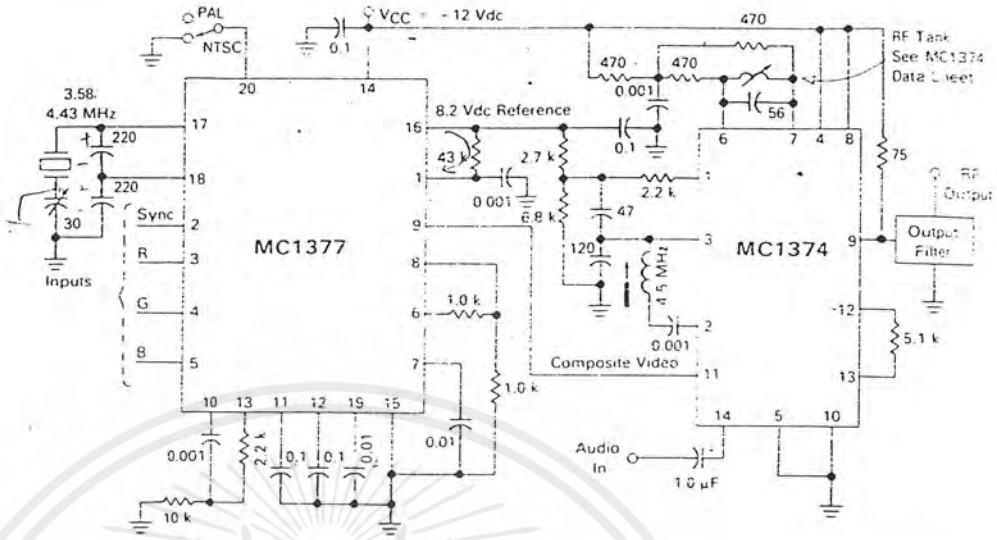


FIGURE 4 — VECTORSCOPE DISPLAY OF 100% SATURATED NTSC COLOR BARS

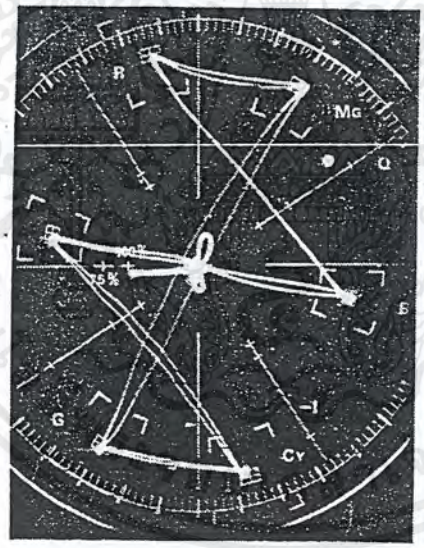
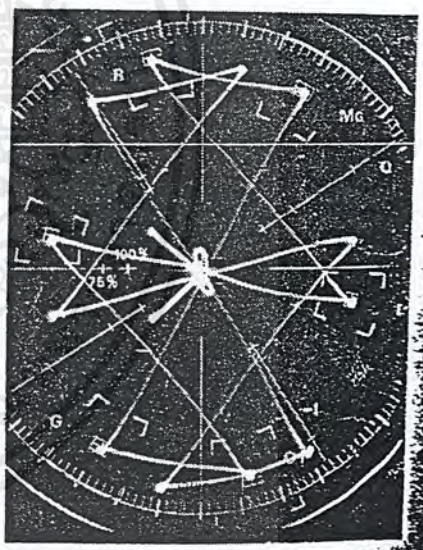


FIGURE 5 — 100% SATURATED PAL COLOR BARS ON NTSC VECTORSCOPE



9

Table 16: Pin Descriptions

Pin Name	I/O During Config.	I/O After Config.	Pin Description
Permanently Dedicated Pins			
VCC	I	I	Eight or more (depending on package) connections to the nominal +5 V supply voltage (+3.3 V for low-voltage devices). All must be connected, and each must be decoupled with a 0.01 - 0.1 µF capacitor to Ground.
GND	I	I	Eight or more (depending on package type) connections to Ground. All must be connected.
CCLK	I or O	I	During configuration, Configuration Clock (CCLK) is an output in Master modes or Asynchronous Peripheral mode, but is an input in Slave mode and Synchronous Peripheral mode. After configuration, CCLK has a weak pull-up resistor and can be selected as the Readback Clock. There is no CCLK High or Low time restriction on XC4000 Series devices, except during Readback. See "Violating the Maximum High and Low Time Specification for the Readback Clock" on page 56 for an explanation of this exception.
DONE	I/O	O	DONE is a bidirectional signal with an optional internal pull-up resistor. As an output, it indicates the completion of the configuration process. As an input, a Low level on DONE can be configured to delay the global logic initialization and the enabling of outputs. The optional pull-up resistor is selected as an option in the XACT*step* program that creates the configuration bitstream. The resistor is included by default.
$\overline{\text{PROGRAM}}$	I	I	$\overline{\text{PROGRAM}}$ is an active Low input that forces the FPGA to clear its configuration memory. It is used to initiate a configuration cycle. When $\overline{\text{PROGRAM}}$ goes High, the FPGA finishes the current clear cycle and executes another complete clear cycle, before it goes into a WAIT state and releases $\overline{\text{INIT}}$. The $\overline{\text{PROGRAM}}$ pin has a permanent weak pull-up, so it need not be externally pulled up to Vcc.
User I/O Pins That Can Have Special Functions			
RDY/$\overline{\text{BUSY}}$	O	I/O	During Peripheral mode configuration, this pin indicates when it is appropriate to write another byte of data into the FPGA. The same status is also available on D7 in Asynchronous Peripheral mode, if a read operation is performed when the device is selected. After configuration, RDY/$\overline{\text{BUSY}}$ is a user-programmable I/O pin. RDY/$\overline{\text{BUSY}}$ is pulled High with a high-impedance pull-up prior to $\overline{\text{INIT}}$ going High.
$\overline{\text{RCLK}}$	O	I/O	During Master Parallel configuration, each change on the A0-A17 outputs (A0 - A21 for XC4000X) is preceded by a rising edge on $\overline{\text{RCLK}}$, a redundant output signal. $\overline{\text{RCLK}}$ is useful for clocked PROMs. It is rarely used during configuration. After configuration, $\overline{\text{RCLK}}$ is a user-programmable I/O pin.
M0, M1, M2	I	I (M0), O (M1), I (M2)	As Mode inputs, these pins are sampled after $\overline{\text{INIT}}$ goes High to determine the configuration mode to be used. After configuration, M0 and M2 can be used as inputs, and M1 can be used as a 3-state output. These three pins have no associated input or output registers. During configuration, these pins have weak pull-up resistors. For the most popular configuration mode, Slave Serial, the mode pins can thus be left unconnected. The three mode inputs can be individually configured with or without weak pull-up or pull-down resistors. A pull-down resistor value of 4.7 kΩ is recommended. These pins can only be used as inputs or outputs when called out by special schematic definitions. To use these pins, place the library components MD0, MD1, and MD2 instead of the usual pad symbols. Input or output buffers must still be used.
TDO	O	O	If boundary scan is used, this pin is the Test Data Output. If boundary scan is not used, this pin is a 3-state output without a register, after configuration is completed. This pin can be user output only when called out by special schematic definitions. To use this pin, place the library component TDO instead of the usual pad symbol. An output buffer must still be used.

Table 16: Pin Descriptions (Continued)

Pin Name	I/O During Config.	I/O After Config.	Pin Description
TDI, TCK, TMS	I	I/O or I (JTAG)	If boundary scan is used, these pins are Test Data In, Test Clock, and Test Mode Select inputs respectively. They come directly from the pads, bypassing the IOBs. These pins can also be used as inputs to the CLB logic after configuration is completed. If the BSCAN symbol is not placed in the design, all boundary scan functions are inhibited once configuration is completed, and these pins become user-programmable I/O. In this case, they must be called out by special schematic definitions. To use these pins, place the library components TDI, TCK, and TMS instead of the usual pad symbols. Input or output buffers must still be used.
HDC	O	I/O	High During Configuration (HDC) is driven High until the I/O go active. It is available as a control output indicating that configuration is not yet completed. After configuration, HDC is a user-programmable I/O pin.
$\overline{\text{LDC}}$	O	I/O	Low During Configuration ($\overline{\text{LDC}}$) is driven Low until the I/O go active. It is available as a control output indicating that configuration is not yet completed. After configuration, $\overline{\text{LDC}}$ is a user-programmable I/O pin.
$\overline{\text{INIT}}$	I/O	I/O	Before and during configuration, $\overline{\text{INIT}}$ is a bidirectional signal. A 1 kΩ - 10 kΩ external pull-up resistor is recommended. As an active-Low open-drain output, $\overline{\text{INIT}}$ is held Low during the power stabilization and internal clearing of the configuration memory. As an active-Low input, it can be used to hold the FPGA in the internal WAIT state before the start of configuration. Master mode devices stay in a WAIT state an additional 30 to 300 µs after $\overline{\text{INIT}}$ has gone High. During configuration, a Low on this output indicates that a configuration data error has occurred. After the I/O go active, $\overline{\text{INIT}}$ is a user-programmable I/O pin.
PGCK1 - PGCK4 (XC4000E only)	Weak Pull-up	I or I/O	Four Primary Global inputs each drive a dedicated internal global net with short delay and minimal skew. If not used to drive a global buffer, any of these pins is a user-programmable I/O. The PGCK1-PGCK4 pins drive the four Primary Global Buffers. Any input pad symbol connected directly to the input of a BUFGP symbol is automatically placed on one of these pins.
SGCK1 - SGCK4 (XC4000E only)	Weak Pull-up	I or I/O	Four Secondary Global inputs each drive a dedicated internal global net with short delay and minimal skew. These internal global nets can also be driven from internal logic. If not used to drive a global net, any of these pins is a user-programmable I/O pin. The SGCK1-SGCK4 pins provide the shortest path to the four Secondary Global Buffers. Any input pad symbol connected directly to the input of a BUFGS symbol is automatically placed on one of these pins.
GCK1 - GCK8 (XC4000X only)	Weak Pull-up	I or I/O	Eight inputs can each drive a Global Low-Skew buffer. In addition, each can drive a Global Early buffer. Each pair of global buffers can also be driven from internal logic, but must share an input signal. If not used to drive a global buffer, any of these pins is a user-programmable I/O. Any input pad symbol connected directly to the input of a BUFGLS or BUFGE symbol is automatically placed on one of these pins.
FCLK1 - FCLK4 (XC4000XLA and XC4000XV only)	Weak Pull-up	I or I/O	Four inputs can each drive a Fast Clock (FCLK) buffer which can deliver a clock signal to any IOB clock input in the octant of the die served by the Fast Clock buffer. Two Fast Clock buffers serve the two IOB octants on the left side of the die and the other two Fast Clock buffers serve the two IOB octants on the right side of the die. On each side of the die, one Fast Clock buffer serves the upper octant and the other serves the lower octant. If not used to drive a Fast Clock buffer, any of these pins is a user-programmable I/O.

Table 16: Pin Descriptions (Continued)

Pin Name	I/O During Config.	I/O After Config.	Pin Description
$\overline{CS0}$, CS1, $\overline{WS}$, $\overline{RS}$	I	I/O	These four inputs are used in Asynchronous Peripheral mode. The chip is selected when $\overline{CS0}$ is Low and CS1 is High. While the chip is selected, a Low on Write Strobe ($\overline{WS}$) loads the data present on the D0 - D7 inputs into the internal data buffer. A Low on Read Strobe ($\overline{RS}$) changes D7 into a status output — High if Ready, Low if Busy — and drives D0 - D6 High. In Express mode, CS1 is used as a serial-enable signal for daisy-chaining. $\overline{WS}$ and $\overline{RS}$ should be mutually exclusive, but if both are Low simultaneously, the Write Strobe overrides. After configuration, these are user-programmable I/O pins.
A0 - A17	O	I/O	During Master Parallel configuration, these 18 output pins address the configuration EPROM. After configuration, they are user-programmable I/O pins.
A18 - A21 (XC4003XL to XC4085XL)	O	I/O	During Master Parallel configuration with an XC4000X master, these 4 output pins add 4 more bits to address the configuration EPROM. After configuration, they are user-programmable I/O pins. (See Master Parallel Configuration section for additional details.)
D0 - D7	I	I/O	During Master Parallel and Peripheral configuration, these eight input pins receive configuration data. After configuration, they are user-programmable I/O pins.
DIN	I	I/O	During Slave Serial or Master Serial configuration, DIN is the serial configuration data input receiving data on the rising edge of CCLK. During Parallel configuration, DIN is the D0 input. After configuration, DIN is a user-programmable I/O pin.
DOUT	O	I/O	During configuration in any mode but Express mode, DOUT is the serial configuration data output that can drive the DIN of daisy-chained slave FPGAs. DOUT data changes on the falling edge of CCLK, one-and-a-half CCLK periods after it was received at the DIN input. In Express modefor XC4000E and XC4000X only, DOUT is the status output that can drive the CS1 of daisy-chained FPGAs, to enable and disable downstream devices. After configuration, DOUT is a user-programmable I/O pin.
Unrestricted User-Programmable I/O Pins			
I/O	Weak Pull-up	I/O	These pins can be configured to be input and/or output after configuration is completed. Before configuration is completed, these pins have an internal high-value pull-up resistor (25 kΩ - 100 kΩ) that defines the logic level as High.

Boundary Scan

The 'bed of nails' has been the traditional method of testing electronic assemblies. This approach has become less appropriate, due to closer pin spacing and more sophisticated assembly methods like surface-mount technology and multi-layer boards. The IEEE Boundary Scan Standard 1149.1 was developed to facilitate board-level testing of electronic assemblies. Design and test engineers can imbed a standard test logic structure in their device to achieve high fault coverage for I/O and internal logic. This structure is easily implemented with a four-pin interface on any boundary scan-compatible IC. IEEE 1149.1-compatible devices may be serial daisy-chained together, connected in parallel, or a combination of the two.

The XC4000 Series implements IEEE 1149.1-compatible BYPASS, PRELOAD/SAMPLE and EXTEST boundary scan instructions. When the boundary scan configuration option is selected, three normal user I/O pins become dedicated inputs for these functions. Another user output pin becomes the dedicated boundary scan output. The details of how to enable this circuitry are covered later in this section.

By exercising these input signals, the user can serially load commands and data into these devices to control the driving of their outputs and to examine their inputs. This method is an improvement over bed-of-nails testing. It avoids the need to over-drive device outputs, and it reduces the user interface to four pins. An optional fifth pin, a reset for the control logic, is described in the standard but is not implemented in Xilinx devices.

The dedicated on-chip logic implementing the IEEE 1149.1 functions includes a 16-state machine, an instruction register and a number of data registers. The functional details can be found in the IEEE 1149.1 specification and are also discussed in the Xilinx application note XAPP 017: "*Boundary Scan in XC4000 Devices*."

Figure 40 on page 43 shows a simplified block diagram of the XC4000E Input/Output Block with boundary scan implemented. XC4000X boundary scan logic is identical.

XC4005E/XL Device Pnout Tables

The following table may contain pinout information for unsupported device/package combinations. Please see the availability charts elsewhere in the XC4000 Series data sheet for availability information.

XC4005E/XL Pad Name	PC 84	PQ 100	VQ 100††	TQ 144	PG 156†	PQ 160	PQ 208	Bndry Scan
VCC	P2	P92	P89	P128	H3	P142	P183	-
I/O (A8)	P3	P93	P90	P129	H1	P143	P184	44
I/O (A9)	P4	P94	P91	P130	G1	P144	P185	47
I/O (A19) ††	-	P95	P92	P131	G2	P145	P186	50
I/O (A18) ††	-	P96	P93	P132	G3	P146	P187	53
I/O (A10)	P5	P97	P94	P133	F1	P147	P190	56
I/O (A11)	P6	P98	P95	P134	F2	P148	P191	59
I/O	-	-	-	P135	E1	P149	P192	62
I/O	-	-	-	P136	E2	P150	P193	65
GND	-	-	-	P137	F3	P151	P194	-
I/O (A12)	P7	P99	P96	P138	E3	P154	P199	68
I/O (A13)	P8	P100	P97	P139	C1	P155	P200	71
I/O	-	-	-	P140	C2	P156	P201	74
I/O	-	-	-	P141	D3	P157	P202	77
I/O (A14)	P9	P1	P98	P142	B1	P158	P203	80
I/O, SGCK1 †, GCK8 †† (A15)	P10	P2	P99	P143	B2	P159	P204	83
VCC	P11	P3	P100	P144	C3	P160	P205	-
GND	P12	P4	P1	P1	C4	P1	P2	-
I/O, PGCK1†, GCK1†† (A16)	P13	P5	P2	P2	B3	P2	P4	86
I/O (A17)	P14	P6	P3	P3	A1	P3	P5	89
I/O	-	-	-	P4	A2	P4	P6	92
I/O	-	-	-	P5	C5	P5	P7	95
I/O, TDI	P15	P7	P4	P6	B4	P6	P8	98
I/O, TCK	P16	P8	P5	P7	A3	P7	P9	101
GND	-	-	-	P8	C6	P10	P14	-
I/O	-	-	-	P9	B5	P11	P15	104
I/O	-	-	-	P10	B6	P12	P16	107
I/O, TMS	P17	P9	P6	P11	A5	P13	P17	110
I/O	P18	P10	P7	P12	C7	P14	P18	113
I/O	-	-	-	P13	B7	P15	P21	116
I/O	-	P11	P8	P14	A6	P16	P22	119
I/O	P19	P12	P9	P15	A7	P17	P23	122
I/O	P20	P13	P10	P16	A8	P18	P24	125
GND	P21	P14	P11	P17	C8	P19	P25	-
VCC	P22	P15	P12	P18	B8	P20	P26	-
I/O	P23	P16	P13	P19	C9	P21	P27	128
I/O	P24	P17	P14	P20	B9	P22	P28	131
I/O	-	P18	P15	P21	A9	P23	P29	134
I/O	-	-	-	P22	B10	P24	P30	137
I/O	P25	P19	P16	P23	C10	P25	P33	140
I/O	P26	P20	P17	P24	A10	P26	P34	143
I/O	-	-	-	P25	A11	P27	P35	146
I/O	-	-	-	P26	B11	P28	P36	149
GND	-	-	-	P27	C11	P29	P37	-
I/O	P27	P21	P18	P28	B12	P32	P42	152
I/O	-	P22	P19	P29	A13	P33	P43	155
I/O	-	-	-	P30	A14	P34	P44	158
I/O	-	-	-	P31	C12	P35	P45	161
I/O	P28	P23	P20	P32	B13	P36	P46	164
I/O, SGCK2 †, GCK2 ††	P29	P24	P21	P33	B14	P37	P47	167
O (M1)	P30	P25	P22	P34	A15	P38	P48	170
GND	P31	P26	P23	P35	C13	P39	P49	-
I (M0)	P32	P27	P24	P36	A16	P40	P50	173
VCC	P33	P28	P25	P37	C14	P41	P55	-
I (M2)	P34	P29	P26	P38	B15	P42	P56	174
I/O, PGCK2 †, GCK3 ††	P35	P30	P27	P39	B16	P43	P57	175
I/O (HDC)	P36	P31	P28	P40	D14	P44	P58	178
I/O	-	-	-	P41	C15	P45	P59	181
I/O	-	-	-	P42	D15	P46	P60	184
I/O	-	P32	P29	P43	E14	P47	P61	187
I/O (LDC)	P37	P33	P30	P44	C16	P48	P62	190

XC4005E/XL Pad Name	PC 84	PQ 100	VQ 100††	TQ 144	PG 156†	PQ 160	PQ 208	Bndry Scan
GND	-	-	-	P45	F14	P51	P67	-
I/O	-	-	-	P46	F15	P52	P68	193
I/O	-	-	-	P47	E16	P53	P69	196
I/O	P38	P34	P31	P48	F16	P54	P70	199
I/O	P39	P35	P32	P49	G14	P55	P71	202
I/O	-	P36	P33	P50	G15	P56	P74	205
I/O	-	P37	P34	P51	G16	P57	P75	208
I/O	P40	P38	P35	P52	H16	P58	P76	211
I/O (INIT)	P41	P39	P36	P53	H15	P59	P77	214
VCC	P42	P40	P37	P54	H14	P60	P78	-
GND	P43	P41	P38	P55	J14	P61	P79	-
I/O	P44	P42	P39	P56	J15	P62	P80	217
I/O	P45	P43	P40	P57	J16	P63	P81	220
I/O	-	P44	P41	P58	K16	P64	P82	223
I/O	-	P45	P42	P59	K15	P65	P83	226
I/O	P46	P46	P43	P60	K14	P66	P86	229
I/O	P47	P47	P44	P61	L16	P67	P87	232
I/O	-	-	-	P62	M16	P68	P88	235
I/O	-	-	-	P63	L15	P69	P89	238
GND	-	-	-	P64	L14	P70	P90	-
I/O	P48	P48	P45	P65	P16	P73	P95	241
I/O	P49	P49	P46	P66	M14	P74	P96	244
I/O	-	-	-	P67	N15	P75	P97	247
I/O	-	-	-	P68	P15	P76	P98	250
I/O	P50	P50	P47	P69	N14	P77	P99	253
I/O, SGCK3 †, GCK4 ††	P51	P51	P48	P70	R16	P78	P100	256
GND	P52	P52	P49	P71	P14	P79	P101	-
DONE	P53	P53	P50	P72	R15	P80	P103	-
VCC	P54	P54	P51	P73	P13	P81	P106	-
PROGRAM	P55	P55	P52	P74	R14	P82	P108	-
I/O (D7)	P56	P56	P53	P75	T16	P83	P109	259
I/O, PGCK3†, GCK5††	P57	P57	P54	P76	T15	P84	P110	262
I/O	-	-	-	P77	R13	P85	P111	265
I/O	-	-	-	P78	P12	P86	P112	268
I/O (D6)	P58	P58	P55	P79	T14	P87	P113	271
I/O	-	P59	P56	P80	T13	P88	P114	274
GND	-	-	-	P81	P11	P91	P119	-
I/O	-	-	-	P82	R11	P92	P120	277
I/O	-	-	-	P83	T11	P93	P121	280
I/O (D5)	P59	P60	P57	P84	T10	P94	P122	283
I/O (CS0)	P60	P61	P58	P85	P10	P95	P123	286
I/O	-	P62	P59	P86	R10	P96	P126	289
I/O	-	P63	P60	P87	T9	P97	P127	292
I/O (D4)	P61	P64	P61	P88	R9	P98	P128	295
I/O	P62	P65	P62	P89	P9	P99	P129	298
VCC	P63	P66	P63	P90	R8	P100	P130	-
GND	P64	P67	P64	P91	P8	P101	P131	-
I/O (D3)	P65	P68	P65	P92	T8	P102	P132	301
I/O (RS)	P66	P69	P66	P93	T7	P103	P133	304
I/O	-	P70	P67	P94	T6	P104	P134	307
I/O	-	-	-	P95	R7	P105	P135	310
I/O (D2)	P67	P71	P68	P96	P7	P106	P138	313
I/O	P68	P72	P69	P97	T5	P107	P139	316
I/O	-	-	-	P98	R6	P108	P140	319
I/O	-	-	-	P99	T4	P109	P141	322
GND	-	-	-	P100	P6	P110	P142	-
I/O (D1)	P69	P73	P70	P101	T3	P113	P147	325
I/O (RCLK, RDY/BUSY)	P70	P74	P71	P102	P5	P114	P148	328
I/O	-	-	-	P103	R4	P115	P149	331
I/O	-	-	-	P104	R3	P116	P150	334
I/O (D0, DIN)	P71	P75	P72	P105	P4	P117	P151	337

XC4005E/XL Pad Name	PC 84	PQ 100	VQ 100††	TQ 144	PG 156†	PQ 160	PQ 208	Bndry Scan
I/O, SGCK4 †, GCK6 †† (DOUT)	P72	P76	P73	P106	T2	P118	P152	340
CCLK	P73	P77	P74	P107	R2	P119	P153	-
VCC	P74	P78	P75	P108	P3	P120	P154	-
O, TDO	P75	P79	P76	P109	T1	P121	P159	0
GND	P76	P80	P77	P110	N3	P122	P160	-
I/O (A0, WS)	P77	P81	P78	P111	R1	P123	P161	2
I/O, PGCK4 †, GCK7 †† (A1)	P78	P82	P79	P112	P2	P124	P162	5
I/O	-	-	-	P113	N2	P125	P163	8
I/O	-	-	-	P114	M3	P126	P164	11
I/O (CS1, A2)	P79	P83	P80	P115	P1	P127	P165	14
I/O (A3)	P80	P84	P81	P116	N1	P128	P166	17
GND	-	-	-	P118	L3	P131	P171	-
I/O	-	-	-	P119	L2	P132	P172	20
I/O	-	-	-	P120	L1	P133	P173	23
I/O (A4)	P81	P85	P82	P121	K3	P134	P174	26
I/O (A5)	P82	P86	P83	P122	K2	P135	P175	29
I/O (A21) ††	-	P87	P84	P123	K1	P137	P178	32
I/O (A20) ††	-	P88	P85	P124	J1	P138	P179	35
I/O (A6)	P83	P89	P86	P125	J2	P139	P180	38
I/O (A7)	P84	P90	P87	P126	J3	P140	P181	41
GND	P1	P91	P88	P127	H2	P141	P182	-

6/10/97

† = E only
†† = XL only

Additional XC4005E/XL Package Pins

TQ144

Not Connected Pins					
P117	-	-	-	-	-

5/5/97

PG156

Not Connected Pins					
A4	A12	D1	D2	D16	E15
M1	M2	M15	N16	R5	R12
T12	-	-	-	-	-

5/5/97

PQ160

Not Connected Pins					
P8	P9	P30	P31	P49	P50
P71	P72	P89	P90	P111	P112
P129	P130	P136	P152	P153	-

6/16/97

PQ208

Not Connected Pins					
P1	P3	P10	P11	P12	P13
P19	P20	P31	P32	P38	P39
P40	P41	P51	P52	P53	P54
P63	P64	P65	P66	P72	P73
P84	P85	P91	P92	P93	P94
P102	P104	P105	P107	P115	P116
P117	P118	P124	P125	P136	P137
P143	P144	P145	P146	P155	P156
P157	P158	P167	P168	P169	P170
P176	P177	P188	P189	P195	P196
P197	P198	P206	P207	P208	-

6/5/97

XC4000E and XC4000X Series Features

Note: Information in this data sheet covers the XC4000E, XC4000EX, and XC4000XL families. A separate data sheet covers the XC4000XLA and XC4000XV families. Electrical Specifications and package/pin information are covered in separate sections for each family to make the information easier to access, review, and print. For access to these sections, see the Xilinx WEBLINX web site at

http://www.xilinx.com/partinfo/databook.htm#xc4000.

- System featured Field-Programmable Gate Arrays
  - Select-RAM™ memory: on-chip ultra-fast RAM with
    - synchronous write option
    - dual-port RAM option
  - Fully PCI compliant (speed grades -2 and faster)
  - Abundant flip-flops
  - Flexible function generators
  - Dedicated high-speed carry logic
  - Wide edge decoders on each edge
  - Hierarchy of interconnect lines
  - Internal 3-state bus capability
  - Eight global low-skew clock or signal distribution networks
- System Performance beyond 80 MHz
- Flexible Array Architecture
- Low Power Segmented Routing Architecture
- Systems-Oriented Features
  - IEEE 1149.1-compatible boundary scan logic support
  - Individually programmable output slew rate
  - Programmable input pull-up or pull-down resistors
  - 12 mA sink current per XC4000E output
- Configured by Loading Binary File
  - Unlimited re-programmability
- Read Back Capability
  - Program verification
  - Internal node observability
- Backward Compatible with XC4000 Devices
- Development System runs on most common computer platforms
  - Interfaces to popular design environments
  - Fully automatic mapping, placement and routing
  - Interactive design editor for design optimization

Low-Voltage Versions Available

- Low-Voltage Devices Function at 3.0 - 3.6 Volts
- XC4000XL: High Performance Low-Voltage Versions of XC4000EX devices

Additional XC4000X Series Features

- Highest Performance — 3.3 V XC4000XL
- Highest Capacity — Over 180,000 Usable Gates
- 5 V tolerant I/Os on XC4000XL
- 0.35 µm SRAM process for XC4000XL
- Additional Routing Over XC4000E
  - almost twice the routing capacity for high-density designs
- Buffered Interconnect for Maximum Speed Blocks
- Improved VersaRing™ I/O Interconnect for Better Fixed Pinout Flexibility
- 12 mA Sink Current Per XC4000X Output
- Flexible New High-Speed Clock Network
  - Eight additional Early Buffers for shorter clock delays
  - Virtually unlimited number of clock signals
- Optional Multiplexer or 2-input Function Generator on Device Outputs
- Four Additional Address Bits in Master Parallel Configuration Mode
- XC4000XV Family offers the highest density with 0.25 µm 2.5 V technology

Introduction

XC4000 Series high-performance, high-capacity Field Programmable Gate Arrays (FPGAs) provide the benefits of custom CMOS VLSI, while avoiding the initial cost, long development cycle, and inherent risk of a conventional masked gate array.

The result of thirteen years of FPGA design experience and feedback from thousands of customers, these FPGAs combine architectural versatility, on-chip Select-RAM memory with edge-triggered and dual-port modes, increased speed, abundant routing resources, and new, sophisticated software to achieve fully automated implementation of complex, high-density, high-performance designs.

The XC4000E and XC4000X Series currently have 20 members, as shown in Table 1.

Table 1: XC4000E and XG4000X Series Field Programmable Gate Arrays

Device	Logic Cells	Max Logic Gates (No RAM)	Max. RAM Bits (No Logic)	Typical Gate Range (Logic and RAM)*	CLB Matrix	Total CLBs	Number of Flip-Flops	Max. User I/O
XC4002XL	152	1,600	2,048	1,000 - 3,000	8 x 8	64	256	64
XC4003E	238	3,000	3,200	2,000 - 5,000	10 x 10	100	360	80
XC4005E/XL	466	5,000	6,272	3,000 - 9,000	14 x 14	196	616	112
XC4006E	608	6,000	8,192	4,000 - 12,000	16 x 16	256	768	128
XC4008E	770	8,000	10,368	6,000 - 15,000	18 x 18	324	936	144
XC4010E/XL	950	10,000	12,800	7,000 - 20,000	20 x 20	400	1,120	160
XC4013E/XL	1368	13,000	18,432	10,000 - 30,000	24 x 24	576	1,536	192
XC4020E/XL	1862	20,000	25,088	13,000 - 40,000	28 x 28	784	2,016	224
XC4025E	2432	25,000	32,768	15,000 - 45,000	32 x 32	1,024	2,560	256
XC4028EX/XL	2432	28,000	32,768	18,000 - 50,000	32 x 32	1,024	2,560	256
XC4036EX/XL	3078	36,000	41,472	22,000 - 65,000	36 x 36	1,296	3,168	288
XC4044XL	3800	44,000	51,200	27,000 - 80,000	40 x 40	1,600	3,840	320
XC4052XL	4598	52,000	61,952	33,000 - 100,000	44 x 44	1,936	4,576	352
XC4062XL	5472	62,000	73,728	40,000 - 130,000	48 x 48	2,304	5,376	384
XC4085XL	7448	85,000	100,352	55,000 - 180,000	56 x 56	3,136	7,168	448

* Max values of Typical Gate Range include 20-30% of CLBs used as RAM.

Note: *All functionality in low-voltage families is the same as in the corresponding 5-Volt family, except where numerical references are made to timing or power.*

Description

XC4000 Series devices are implemented with a regular, flexible, programmable architecture of Configurable Logic Blocks (CLBs), interconnected by a powerful hierarchy of versatile routing resources, and surrounded by a perimeter of programmable Input/Output Blocks (IOBs). They have generous routing resources to accommodate the most complex interconnect patterns.

The devices are customized by loading configuration data into internal memory cells. The FPGA can either actively read its configuration data from an external serial or byte-parallel PROM (master modes), or the configuration data can be written into the FPGA from an external device (slave and peripheral modes).

XC4000 Series FPGAs are supported by powerful and sophisticated software, covering every aspect of design from schematic or behavioral entry, floor planning, simulation, automatic block placement and routing of interconnects, to the creation, downloading, and readback of the configuration bit stream.

Because Xilinx FPGAs can be reprogrammed an unlimited number of times, they can be used in innovative designs where hardware is changed dynamically, or where hardware must be adapted to different user applications. FPGAs are ideal for shortening design and development cycles, and also offer a cost-effective solution for production rates well beyond 5,000 systems per month. For lowest high-volume unit cost, a design can first be implemented in the XC4000E or XC4000X, then migrated to one of Xilinx' compatible HardWire mask-programmed devices.

Taking Advantage of Re-configuration

FPGA devices can be re-configured to change logic function while resident in the system. This capability gives the system designer a new degree of freedom not available with any other type of logic.

Hardware can be changed as easily as software. Design updates or modifications are easy, and can be made to products already in the field. An FPGA can even be re-configured dynamically to perform different functions at different times.

Re-configurable logic can be used to implement system self-diagnostics, create systems capable of being re-configured for different environments or operations, or implement multi-purpose hardware for a given application. As an added benefit, using re-configurable FPGA devices simplifies hardware design and debugging and shortens product time-to-market.

XC4000E and XC4000X Series Compared to the XC4000

For readers already familiar with the XC4000 family of Xilinx Field Programmable Gate Arrays, the major new features in the XC4000 Series devices are listed in this section. The biggest advantages of XC4000E and XC4000X devices are significantly increased system speed, greater capacity, and new architectural features, particularly Select-RAM memory. The XC4000X devices also offer many new routing features, including special high-speed clock buffers that can be used to capture input data with minimal delay.

Any XC4000E device is pinout- and bitstream-compatible with the corresponding XC4000 device. An existing XC4000 bitstream can be used to program an XC4000E device. However, since the XC4000E includes many new features, an XC4000E bitstream cannot be loaded into an XC4000 device.

XC4000X Series devices are not bitstream-compatible with equivalent array size devices in the XC4000 or XC4000E families. However, equivalent array size devices, such as the XC4025, XC4025E, XC4028EX, and XC4028XL, are pinout-compatible.

Improvements in XC4000E and XC4000X

Increased System Speed

XC4000E and XC4000X devices can run at synchronous system clock rates of up to 80 MHz, and internal performance can exceed 150 MHz. This increase in performance over the previous families stems from improvements in both device processing and system architecture. XC4000 Series devices use a sub-micron multi-layer metal process. In addition, many architectural improvements have been made, as described below.

The XC4000XL family is a high performance 3.3V family based on 0.35µ SRAM technology and supports system speeds to 80 MHz.

PCI Compliance

XC4000 Series -2 and faster speed grades are fully PCI compliant. XC4000E and XC4000X devices can be used to implement a one-chip PCI solution.

Carry Logic

The speed of the carry logic chain has increased dramatically. Some parameters, such as the delay on the carry chain through a single CLB (TBYP), have improved by as much as 50% from XC4000 values. See "Fast Carry Logic" on page 18 for more information.

Select-RAM Memory: Edge-Triggered, Synchronous RAM Modes

The RAM in any CLB can be configured for synchronous, edge-triggered, write operation. The read operation is not affected by this change to an edge-triggered write.

Dual-Port RAM

A separate option converts the 16x2 RAM in any CLB into a 16x1 dual-port RAM with simultaneous Read/Write.

The function generators in each CLB can be configured as either level-sensitive (asynchronous) single-port RAM, edge-triggered (synchronous) single-port RAM, edge-triggered (synchronous) dual-port RAM, or as combinatorial logic.

Configurable RAM Content

The RAM content can now be loaded at configuration time, so that the RAM starts up with user-defined data.

H Function Generator

In current XC4000 Series devices, the H function generator is more versatile than in the original XC4000. Its inputs can come not only from the F and G function generators but also from up to three of the four control input lines. The H function generator can thus be totally or partially independent of the other two function generators, increasing the maximum capacity of the device.

IOB Clock Enable

The two flip-flops in each IOB have a common clock enable input, which through configuration can be activated individually for the input or output flip-flop or both. This clock enable operates exactly like the EC pin on the XC4000 CLB. This new feature makes the IOBs more versatile, and avoids the need for clock gating.

Output Drivers

The output pull-up structure defaults to a TTL-like totem-pole. This driver is an n-channel pull-up transistor, pulling to a voltage one transistor threshold below Vcc, just like the XC4000 family outputs. Alternatively, XC4000 Series devices can be globally configured with CMOS outputs, with p-channel pull-up transistors pulling to Vcc. Also, the configurable pull-up resistor in the XC4000 Series is a p-channel transistor that pulls to Vcc, whereas in the original XC4000 family it is an n-channel transistor that pulls to a voltage one transistor threshold below Vcc.

Input Thresholds

The input thresholds of 5V devices can be globally configured for either TTL (1.2 V threshold) or CMOS (2.5 V threshold), just like XC2000 and XC3000 inputs. The two global adjustments of input threshold and output level are independent of each other. The XC4000XL family has an input threshold of 1.6V, compatible with both 3.3V CMOS and TTL levels.

Global Signal Access to Logic

There is additional access from global clocks to the F and G function generator inputs.

Configuration Pin Pull-Up Resistors

During configuration, these pins have weak pull-up resistors. For the most popular configuration mode, Slave Serial, the mode pins can thus be left unconnected. The three mode inputs can be individually configured with or without weak pull-up or pull-down resistors. A pull-down resistor value of 4.7 kΩ is recommended.

The three mode inputs can be individually configured with or without weak pull-up or pull-down resistors after configuration.

The $\overline{\text{PROGRAM}}$ input pin has a permanent weak pull-up.

Soft Start-up

Like the XC3000A, XC4000 Series devices have "Soft Start-up." When the configuration process is finished and the device starts up, the first activation of the outputs is automatically slew-rate limited. This feature avoids potential ground bounce when all outputs are turned on simultaneously. Immediately after start-up, the slew rate of the individual outputs is, as in the XC4000 family, determined by the individual configuration option.

XC4000 and XC4000A Compatibility

Existing XC4000 bitstreams can be used to configure an XC4000E device. XC4000A bitstreams must be recompiled for use with the XC4000E due to improved routing resources, although the devices are pin-for-pin compatible.

Additional Improvements in XC4000X Only

Increased Routing

New interconnect in the XC4000X includes twenty-two additional vertical lines in each column of CLBs and twelve new horizontal lines in each row of CLBs. The twelve "Quad Lines" in each CLB row and column include optional repowering buffers for maximum speed. Additional high-performance routing near the IOBs enhances pin flexibility.

Faster Input and Output

A fast, dedicated early clock sourced by global clock buffers is available for the IOBs. To ensure synchronization with the regular global clocks, a Fast Capture latch driven by the early clock is available. The input data can be initially loaded into the Fast Capture latch with the early clock, then transferred to the input flip-flop or latch with the low-skew global clock. A programmable delay on the input can be used to avoid hold-time requirements. See "IOB Input Signals" on page 20 for more information.

Latch Capability in CLBs

Storage elements in the XC4000X CLB can be configured as either flip-flops or latches. This capability makes the FPGA highly synthesis-compatible.

IOB Output MUX From Output Clock

A multiplexer in the IOB allows the output clock to select either the output data or the IOB clock enable as the output to the pad. Thus, two different data signals can share a single output pad, effectively doubling the number of device outputs without requiring a larger, more expensive package. This multiplexer can also be configured as an AND-gate to implement a very fast pin-to-pin path. See "IOB Output Signals" on page 23 for more information.

Additional Address Bits

Larger devices require more bits of configuration data. A daisy chain of several large XC4000X devices may require a PROM that cannot be addressed by the eighteen address bits supported in the XC4000E. The XC4000X Series therefore extends the addressing in Master Parallel configuration mode to 22 bits.

Detailed Functional Description

XC4000 Series devices achieve high speed through advanced semiconductor technology and improved architecture. The XC4000E and XC4000X support system clock rates of up to 80 MHz and internal performance in excess of 150 MHz. Compared to older Xilinx FPGA families, XC4000 Series devices are more powerful. They offer on-chip edge-triggered and dual-port RAM, clock enables on I/O flip-flops, and wide-input decoders. They are more versatile in many applications, especially those involving RAM. Design cycles are faster due to a combination of increased routing resources and more sophisticated software.

Basic Building Blocks

Xilinx user-programmable gate arrays include two major configurable elements: configurable logic blocks (CLBs) and input/output blocks (IOBs).

- CLBs provide the functional elements for constructing the user's logic.
- IOBs provide the interface between the package pins and internal signal lines.

Three other types of circuits are also available:

- 3-State buffers (TBUFs) driving horizontal longlines are associated with each CLB.
- Wide edge decoders are available around the periphery of each device.
- An on-chip oscillator is provided.

Programmable interconnect resources provide routing paths to connect the inputs and outputs of these configurable elements to the appropriate networks.

The functionality of each circuit block is customized during configuration by programming internal static memory cells. The values stored in these memory cells determine the logic functions and interconnections implemented in the FPGA. Each of these available circuits is described in this section.

Configurable Logic Blocks (CLBs)

Configurable Logic Blocks implement most of the logic in an FPGA. The principal CLB elements are shown in Figure 1. Two 4-input function generators (F and G) offer unrestricted versatility. Most combinatorial logic functions need four or fewer inputs. However, a third function generator (H) is provided. The H function generator has three inputs. Either zero, one, or two of these inputs can be the outputs of F and G; the other input(s) are from outside the CLB. The CLB can, therefore, implement certain functions of up to nine variables, like parity check or expandable-identity comparison of two sets of four inputs.

Each CLB contains two storage elements that can be used to store the function generator outputs. However, the storage elements and function generators can also be used independently. These storage elements can be configured as flip-flops in both XC4000E and XC4000X devices; in the XC4000X they can optionally be configured as latches. DIN can be used as a direct input to either of the two storage elements. H1 can drive the other through the H function generator. Function generator outputs can also drive two outputs independent of the storage element outputs. This versatility increases logic capacity and simplifies routing.

Thirteen CLB inputs and four CLB outputs provide access to the function generators and storage elements. These inputs and outputs connect to the programmable interconnect resources outside the block.

Function Generators

Four independent inputs are provided to each of two function generators (F1 - F4 and G1 - G4). These function generators, with outputs labeled F' and G', are each capable of implementing any arbitrarily defined Boolean function of four inputs. The function generators are implemented as memory look-up tables. The propagation delay is therefore independent of the function implemented.

A third function generator, labeled H', can implement any Boolean function of its three inputs. Two of these inputs can optionally be the F' and G' functional generator outputs. Alternatively, one or both of these inputs can come from outside the CLB (H2, H0). The third input must come from outside the block (H1).

Signals from the function generators can exit the CLB on two outputs. F' or H' can be connected to the X output. G' or H' can be connected to the Y output.

A CLB can be used to implement any of the following functions:

- any function of up to four variables, plus any second function of up to four unrelated variables, plus any third function of up to three unrelated variables[1]
- any single function of five variables
- any function of four variables together with some functions of six variables
- some functions of up to nine variables.

Implementing wide functions in a single block reduces both the number of blocks required and the delay in the signal path, achieving both increased capacity and speed.

The versatility of the CLB function generators significantly improves system speed. In addition, the design-software tools can deal with each function generator independently. This flexibility improves cell usage.

1. When three separate functions are generated, one of the function outputs must be captured in a flip-flop internal to the CLB. Only two unregistered function generator outputs are available from the CLB.

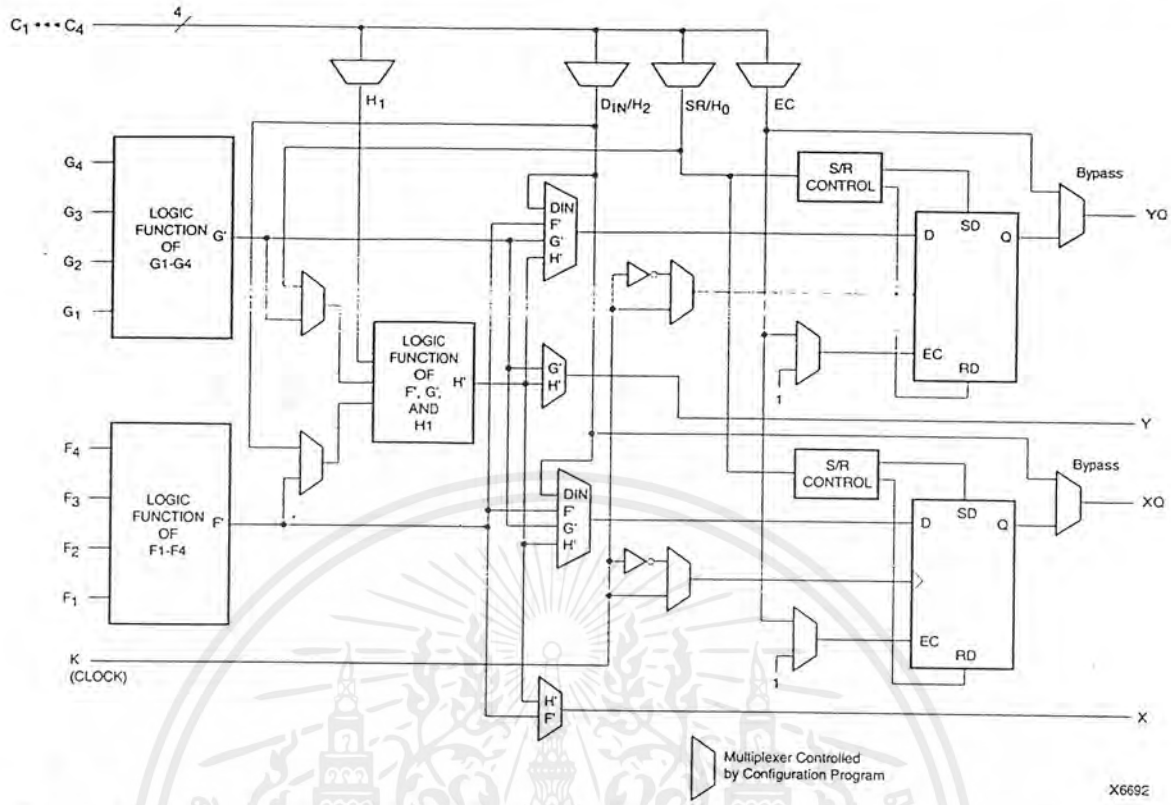


Figure 1: Simplified Block Diagram of XC4000 Series CLB (RAM and Carry Logic functions not shown)

Flip-Flops

The CLB can pass the combinatorial output(s) to the interconnect network, but can also store the combinatorial results or other incoming data in one or two flip-flops, and connect their outputs to the interconnect network as well.

The two edge-triggered D-type flip-flops have common clock (K) and clock enable (EC) inputs. Either or both clock inputs can also be permanently enabled. Storage element functionality is described in Table 2.

Latches (XC4000X only)

The CLB storage elements can also be configured as latches. The two latches have common clock (K) and clock enable (EC) inputs. Storage element functionality is described in Table 2.

Clock Input

Each flip-flop can be triggered on either the rising or falling clock edge. The clock pin is shared by both storage elements. However, the clock is individually invertible for each storage element. Any inverter placed on the clock input is automatically absorbed into the CLB.

Clock Enable

The clock enable signal (EC) is active High. The EC pin is shared by both storage elements. If left unconnected for either, the clock enable for that storage element defaults to the active state. EC is not invertible within the CLB.

Table 2: CLB Storage Element Functionality (active rising edge is shown)

Mode	K	EC	SR	D	Q
Power-Up or GSR	X	X	X	X	SR
Flip-Flop	X	X	1	X	SR
	⎍	1*	0*	D	D
	0	X	0*	X	Q
Latch	1	1*	0*	X	Q
	0	1*	0*	D	D
Both	X	0	0*	X	Q

Legend:

X	Don't care
⎍	Rising edge
SR	Set or Reset value. Reset is default.
0*	Input is Low or unconnected (default value)
1*	Input is High or unconnected (default value)

Set/Reset

An asynchronous storage element input (SR) can be configured as either set or reset. This configuration option determines the state in which each flip-flop becomes operational after configuration. It also determines the effect of a Global Set/Reset pulse during normal operation, and the effect of a pulse on the SR pin of the CLB. All three set/reset functions for any single flip-flop are controlled by the same configuration data bit.

The set/reset state can be independently specified for each flip-flop. This input can also be independently disabled for either flip-flop.

The set/reset state is specified by using the INIT attribute, or by placing the appropriate set or reset flip-flop library symbol.

SR is active High. It is not invertible within the CLB.

Global Set/Reset

A separate Global Set/Reset line (not shown in Figure 1) sets or clears each storage element during power-up, re-configuration, or when a dedicated Reset net is driven active. This global net (GSR) does not compete with other routing resources; it uses a dedicated distribution network.

Each flip-flop is configured as either globally set or reset in the same way that the local set/reset (SR) is specified. Therefore, if a flip-flop is set by SR, it is also set by GSR. Similarly, a reset flip-flop is reset by both SR and GSR.

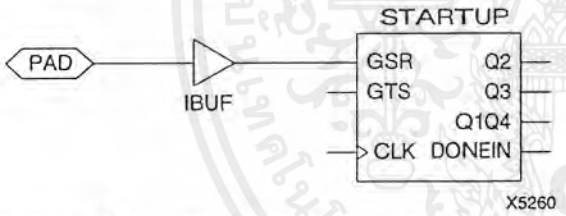


Figure 2: Schematic Symbols for Global Set/Reset

GSR can be driven from any user-programmable pin as a global reset input. To use this global net, place an input pad and input buffer in the schematic or HDL code, driving the GSR pin of the STARTUP symbol. (See Figure 2.) A specific pin location can be assigned to this input using a LOC attribute or property, just as with any other user-programmable pad. An inverter can optionally be inserted after the input buffer to invert the sense of the Global Set/Reset signal.

Alternatively, GSR can be driven from any internal node.

Data Inputs and Outputs

The source of a storage element data input is programmable. It is driven by any of the functions F', G', and H', or by the Direct In (DIN) block input. The flip-flops or latches drive the XQ and YQ CLB outputs.

Two fast feed-through paths are available, as shown in Figure 1. A two-to-one multiplexer on each of the XQ and YQ outputs selects between a storage element output and any of the control inputs. This bypass is sometimes used by the automated router to repower internal signals.

Control Signals

Multiplexers in the CLB map the four control inputs (C1 - C4 in Figure 1) into the four internal control signals (H1, DIN/H2, SR/H0, and EC). Any of these inputs can drive any of the four internal control signals.

When the logic function is enabled, the four inputs are:

- EC — Enable Clock
- SR/H0 — Asynchronous Set/Reset or H function generator Input 0
- DIN/H2 — Direct In or H function generator Input 2
- H1 — H function generator Input 1.

When the memory function is enabled, the four inputs are:

- EC — Enable Clock
- WE — Write Enable
- D0 — Data Input to F and/or G function generator
- D1 — Data input to G function generator (16x1 and 16x2 modes) or 5th Address bit (32x1 mode).

Using FPGA Flip-Flops and Latches

The abundance of flip-flops in the XC4000 Series invites pipelined designs. This is a powerful way of increasing performance by breaking the function into smaller subfunctions and executing them in parallel, passing on the results through pipeline flip-flops. This method should be seriously considered wherever throughput is more important than latency.

To include a CLB flip-flop, place the appropriate library symbol. For example, FDCE is a D-type flip-flop with clock enable and asynchronous clear. The corresponding latch symbol (for the XC4000X only) is called LDCE.

In XC4000 Series devices, the flip flops can be used as registers or shift registers without blocking the function generators from performing a different, perhaps unrelated task. This ability increases the functional capacity of the devices.

The CLB setup time is specified between the function generator inputs and the clock input K. Therefore, the specified CLB flip-flop setup time includes the delay through the function generator.

Using Function Generators as RAM

Optional modes for each CLB make the memory look-up tables in the F' and G' function generators usable as an array of Read/Write memory cells. Available modes are level-sensitive (similar to the XC4000/A/H families), edge-triggered, and dual-port edge-triggered. Depending on the selected mode, a single CLB can be configured as either a 16x2, 32x1, or 16x1 bit array.

Supported CLB memory configurations and timing modes for single- and dual-port modes are shown in Table 3.

XC4000 Series devices are the first programmable logic devices with edge-triggered (synchronous) and dual-port RAM accessible to the user. Edge-triggered RAM simplifies system timing. Dual-port RAM doubles the effective throughput of FIFO applications. These features can be individually programmed in any XC4000 Series CLB.

Advantages of On-Chip and Edge-Triggered RAM

The on-chip RAM is extremely fast. The read access time is the same as the logic delay. The write access time is slightly slower. Both access times are much faster than any off-chip solution, because they avoid I/O delays.

Edge-triggered RAM, also called synchronous RAM, is a feature never before available in a Field Programmable Gate Array. The simplicity of designing with edge-triggered RAM, and the markedly higher achievable performance, add up to a significant improvement over existing devices with on-chip RAM.

Three application notes are available from Xilinx that discuss edge-triggered RAM: "*XC4000E Edge-Triggered and Dual-Port RAM Capability*," "*Implementing FIFOs in XC4000E RAM*," and "*Synchronous and Asynchronous FIFO Designs*." All three application notes apply to both XC4000E and XC4000X RAM.

Table 3: Supported RAM Modes

	16 x 1	16 x 2	32 x 1	Edge-Triggered Timing	Level-Sensitive Timing
Single-Port	√	√	√	√	√
Dual-Port	√			√	

RAM Configuration Options

The function generators in any CLB can be configured as RAM arrays in the following sizes:

- Two 16x1 RAMs: two data inputs and two data outputs with identical or, if preferred, different addressing for each RAM
- One 32x1 RAM: one data input and one data output.

One F or G function generator can be configured as a 16x1 RAM while the other function generators are used to implement any function of up to 5 inputs.

Additionally, the XC4000 Series RAM may have either of two timing modes:

- Edge-Triggered (Synchronous): data written by the designated edge of the CLB clock. WE acts as a true clock enable.
- Level-Sensitive (Asynchronous): an external WE signal acts as the write strobe.

The selected timing mode applies to both function generators within a CLB when both are configured as RAM.

The number of read ports is also programmable:

- Single Port: each function generator has a common read and write port
- Dual Port: both function generators are configured together as a single 16x1 dual-port RAM with one write port and two read ports. Simultaneous read and write operations to the same or different addresses are supported.

RAM configuration options are selected by placing the appropriate library symbol.

Choosing a RAM Configuration Mode

The appropriate choice of RAM mode for a given design should be based on timing and resource requirements, desired functionality, and the simplicity of the design process. Recommended usage is shown in Table 4.

The difference between level-sensitive, edge-triggered, and dual-port RAM is only in the write operation. Read operation and timing is identical for all modes of operation.

Table 4: RAM Mode Selection

	Level-Sensitive	Edge-Triggered	Dual-Port Edge-Triggered
Use for New Designs?	No	Yes	Yes
Size (16x1, Registered)	1/2 CLB	1/2 CLB	1 CLB
Simultaneous Read/Write	No	No	Yes
Relative Performance	X	2X	2X (4X effective)

RAM Inputs and Outputs

The F1-F4 and G1-G4 inputs to the function generators act as address lines, selecting a particular memory cell in each look-up table.

The functionality of the CLB control signals changes when the function generators are configured as RAM. The DIN/H2, H1, and SR/H0 lines become the two data inputs (D0, D1) and the Write Enable (WE) input for the 16x2 memory. When the 32x1 configuration is selected, D1 acts as the fifth address bit and D0 is the data input.

The contents of the memory cell(s) being addressed are available at the F' and G' function-generator outputs. They can exit the CLB through its X and Y outputs, or can be captured in the CLB flip-flop(s).

Configuring the CLB function generators as Read/Write memory does not affect the functionality of the other por-

tions of the CLB, with the exception of the redefinition of the control signals. In 16x2 and 16x1 modes, the H' function generator can be used to implement Boolean functions of F', G', and D1, and the D flip-flops can latch the F', G', H', or D0 signals.

Single-Port Edge-Triggered Mode

Edge-triggered (synchronous) RAM simplifies timing requirements. XC4000 Series edge-triggered RAM timing operates like writing to a data register. Data and address are presented. The register is enabled for writing by a logic High on the write enable input, WE. Then a rising or falling clock edge loads the data into the register, as shown in Figure 3.

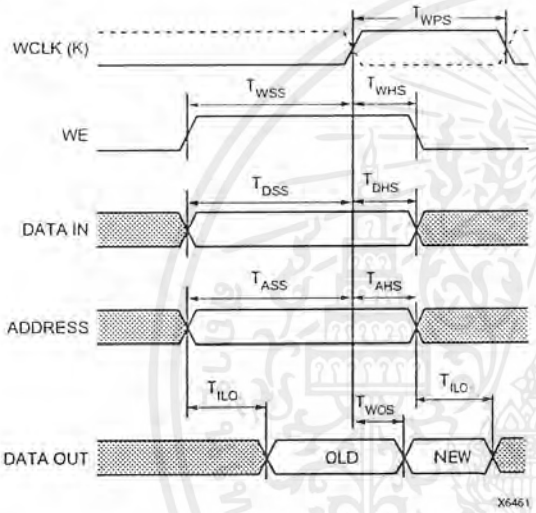


Figure 3: Edge-Triggered RAM Write Timing

Complex timing relationships between address, data, and write enable signals are not required, and the external write enable pulse becomes a simple clock enable. The active edge of WCLK latches the address, input data, and WE signals. An internal write pulse is generated that performs the write. See Figure 4 and Figure 5 for block diagrams of a CLB configured as 16x2 and 32x1 edge-triggered, single-port RAM.

The relationships between CLB pins and RAM inputs and outputs for single-port, edge-triggered mode are shown in Table 5.

The Write Clock input (WCLK) can be configured as active on either the rising edge (default) or the falling edge. It uses the same CLB pin (K) used to clock the CLB flip-flops, but it can be independently inverted. Consequently, the RAM output can optionally be registered within the same CLB either by the same clock edge as the RAM, or by the opposite edge of this clock. The sense of WCLK applies to both function generators in the CLB when both are configured as RAM.

The WE pin is active-High and is not invertible within the CLB.

Note: The pulse following the active edge of WCLK (T_{WPS} in Figure 3) must be less than one millisecond wide. For most applications, this requirement is not overly restrictive; however, it must not be forgotten. Stopping WCLK at this point in the write cycle could result in excessive current and even damage to the larger devices if many CLBs are configured as edge-triggered RAM.

Table 5: Single-Port Edge-Triggered RAM Signals

RAM Signal	CLB Pin	Function
D	D0 or D1 (16x2, 16x1), D0 (32x1)	Data In
A[3:0]	F1-F4 or G1-G4	Address
A[4]	D1 (32x1)	Address
WE	WE	Write Enable
WCLK	K	Clock
SPO (Data Out)	F' or G'	Single Port Out (Data Out)

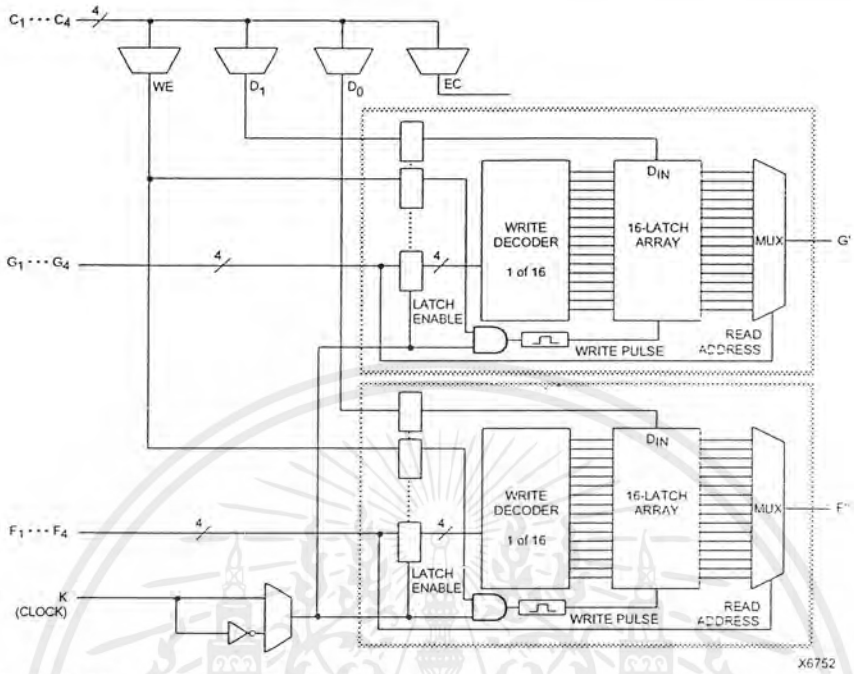


Figure 4: 16x2 (or 16x1) Edge-Triggered Single-Port RAM

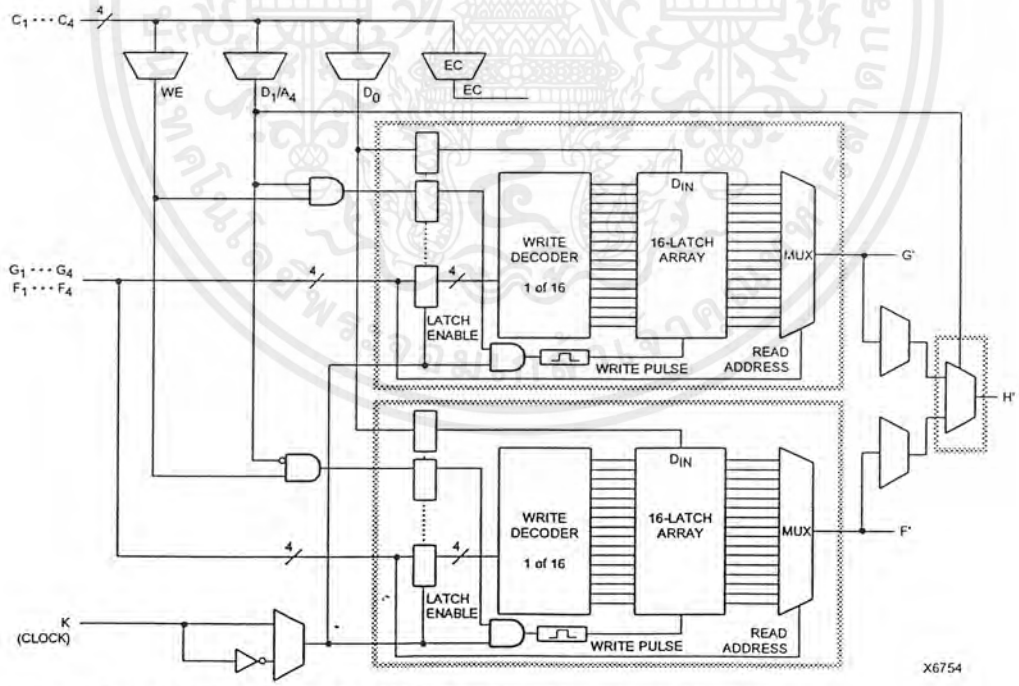


Figure 5: 32x1 Edge-Triggered Single-Port RAM (F and G addresses are identical)

Fast Carry Logic

Each CLB F and G function generator contains dedicated arithmetic logic for the fast generation of carry and borrow signals. This extra output is passed on to the function generator in the adjacent CLB. The carry chain is independent of normal routing resources.

Dedicated fast carry logic greatly increases the efficiency and performance of adders, subtractors, accumulators, comparators and counters. It also opens the door to many new applications involving arithmetic operation, where the previous generations of FPGAs were not fast enough or too inefficient. High-speed address offset calculations in microprocessor or graphics systems, and high-speed addition in digital signal processing are two typical applications.

The two 4-input function generators can be configured as a 2-bit adder with built-in hidden carry that can be expanded to any length. This dedicated carry circuitry is so fast and efficient that conventional speed-up methods like carry generate/propagate are meaningless even at the 16-bit level, and of marginal benefit at the 32-bit level.

This fast carry logic is one of the more significant features of the XC4000 Series, speeding up arithmetic and counting into the 70 MHz range.

The carry chain in XC4000E devices can run either up or down. At the top and bottom of the columns where there are no CLBs above or below, the carry is propagated to the right. (See Figure 11.) In order to improve speed in the high-capacity XC4000X devices, which can potentially have very long carry chains, the carry chain travels upward only, as shown in Figure 12. Additionally, standard interconnect can be used to route a carry signal in the downward direction.

Figure 13 on page 19 shows an XC4000E CLB with dedicated fast carry logic. The carry logic in the XC4000X is similar, except that COUT exits at the top only, and the signal CINDOWN does not exist. As shown in Figure 13, the carry logic shares operand and control inputs with the function generators. The carry outputs connect to the function generators, where they are combined with the operands to form the sums.

Figure 14 on page 20 shows the details of the carry logic for the XC4000E. This diagram shows the contents of the box labeled "CARRY LOGIC" in Figure 13. The XC4000X carry logic is very similar, but a multiplexer on the pass-through carry chain has been eliminated to reduce delay. Additionally, in the XC4000X the multiplexer on the G4 path has a memory-programmable 0 input, which permits G4 to directly connect to COUT. G4 thus becomes an additional high-speed initialization path for carry-in.

The dedicated carry logic is discussed in detail in Xilinx document XAPP 013: "*Using the Dedicated Carry Logic in XC4000*." This discussion also applies to XC4000E devices, and to XC4000X devices when the minor logic changes are taken into account.

The fast carry logic can be accessed by placing special library symbols, or by using Xilinx Relationally Placed Macros (RPMs) that already include these symbols.

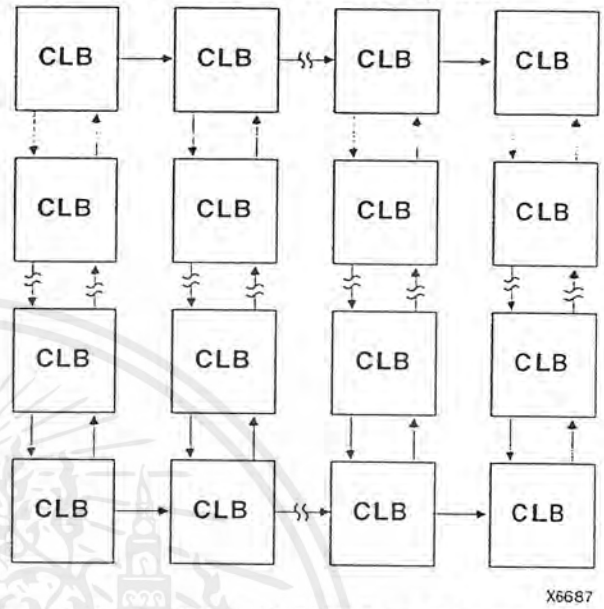


Figure 11: Available XC4000E Carry Propagation Paths

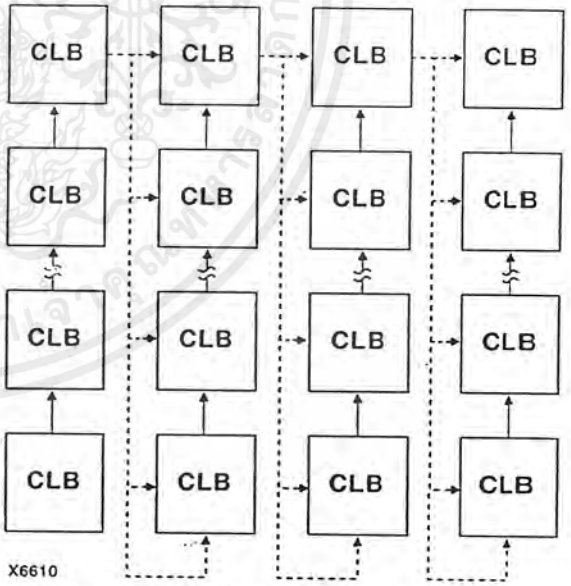


Figure 12: Available XC4000X Carry Propagation Paths (dotted lines use general interconnect)

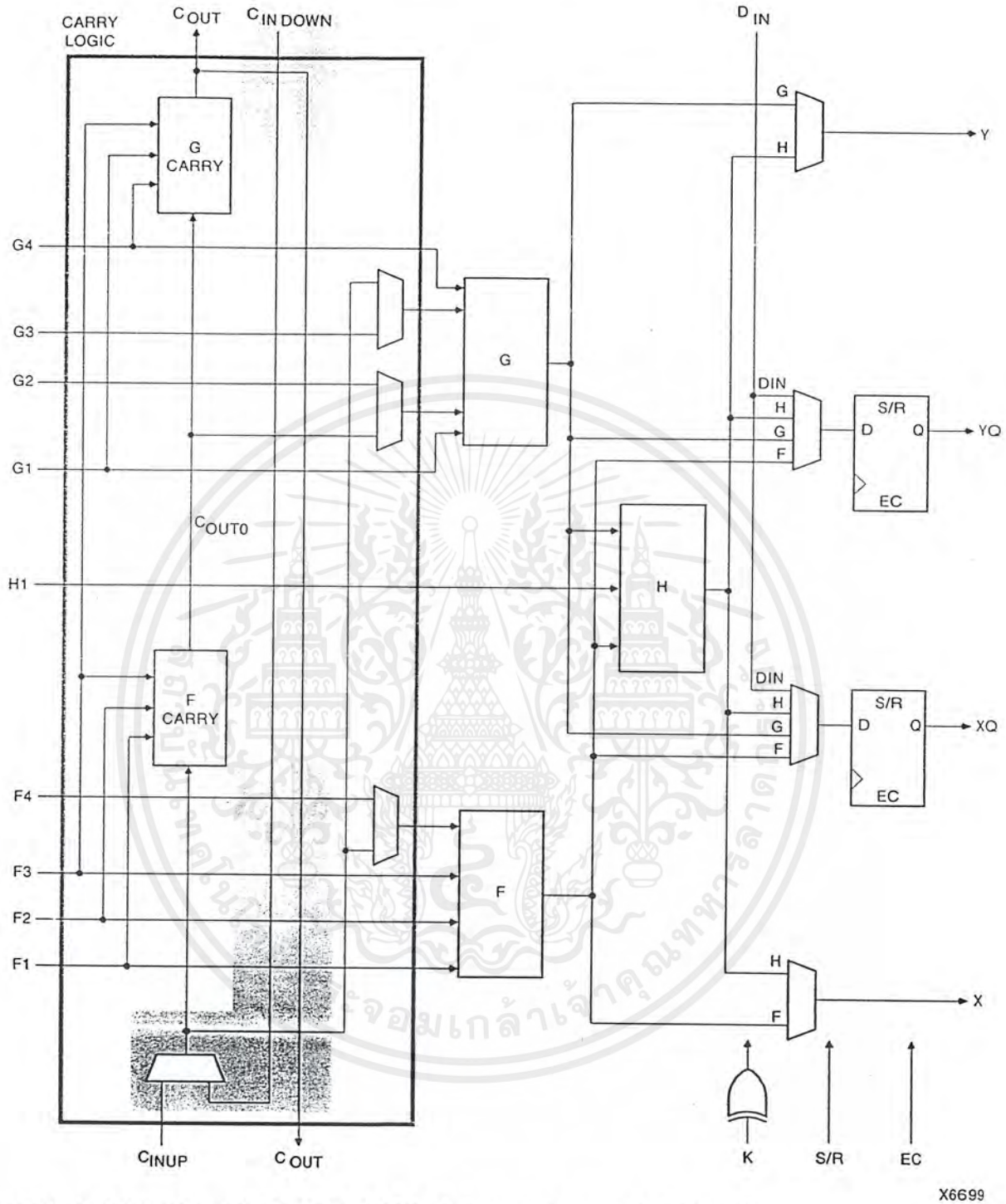


Figure 13: Fast Carry Logic in XC4000E CLB (shaded area not present in XC4000X)

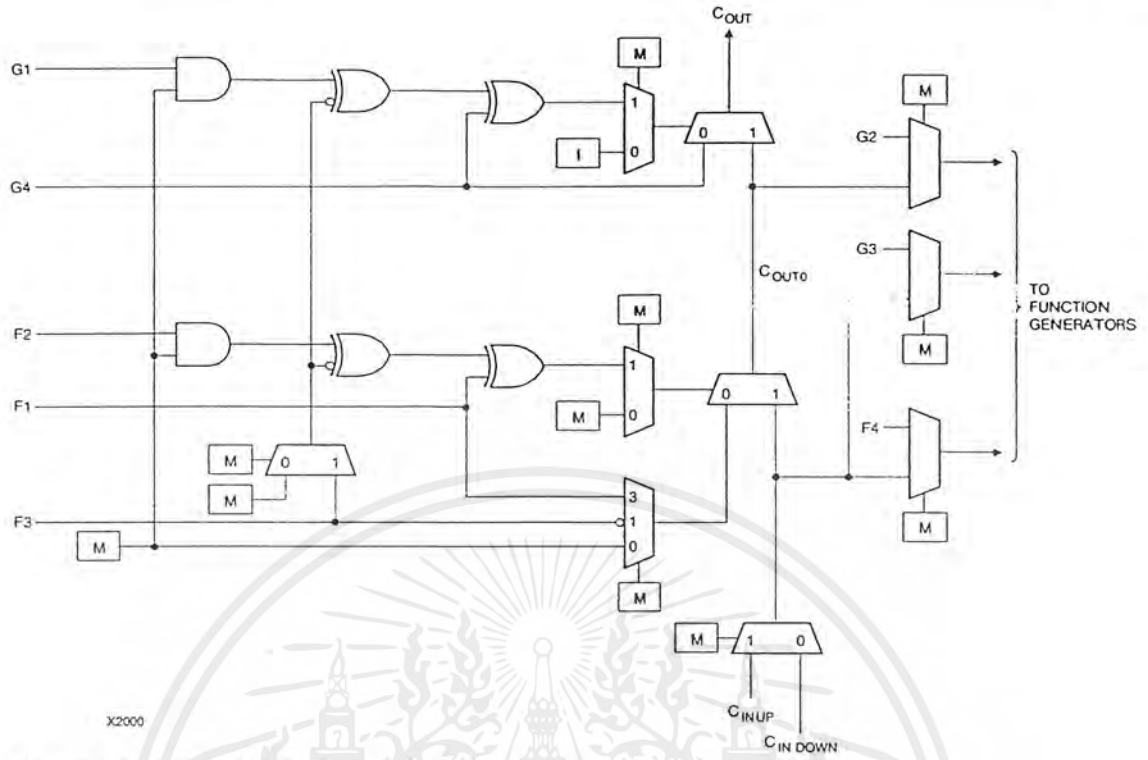


Figure 14: Detail of XC4000E Dedicated Carry Logic

Input/Output Blocks (IOBs)

User-configurable input/output blocks (IOBs) provide the interface between external package pins and the internal logic. Each IOB controls one package pin and can be configured for input, output, or bidirectional signals.

Figure 15 shows a simplified block diagram of the XC4000E IOB. A more complete diagram which includes the boundary scan logic of the XC4000E IOB can be found in Figure 40 on page 43, in the "Boundary Scan" section.

The XC4000X IOB contains some special features not included in the XC4000E IOB. These features are highlighted in a simplified block diagram found in Figure 16, and discussed throughout this section. When XC4000X special features are discussed, they are clearly identified in the text. Any feature not so identified is present in both XC4000E and XC4000X devices.

IOB Input Signals

Two paths, labeled I1 and I2 in Figure 15 and Figure 16, bring input signals into the array. Inputs also connect to an input register that can be programmed as either an edge-triggered flip-flop or a level-sensitive latch.

The choice is made by placing the appropriate library symbol. For example, IFD is the basic input flip-flop (rising edge triggered), and ILD is the basic input latch (transparent-High). Variations with inverted clocks are available, and some combinations of latches and flip-flops can be implemented in a single IOB, as described in the *XACT Libraries Guide*.

The XC4000E inputs can be globally configured for either TTL (1.2V) or 5.0 volt CMOS thresholds, using an option in the bitstream generation software. There is a slight input hysteresis of about 300mV. The XC4000E output levels are also configurable; the two global adjustments of input threshold and output level are independent.

Inputs on the XC4000XL are TTL compatible and 3.3V CMOS compatible. Outputs on the XC4000XL are pulled to the 3.3V positive supply.

The inputs of XC4000 Series 5-Volt devices can be driven by the outputs of any 3.3-Volt device, if the 5-Volt inputs are in TTL mode.

Supported sources for XC4000 Series device inputs are shown in Table 8.

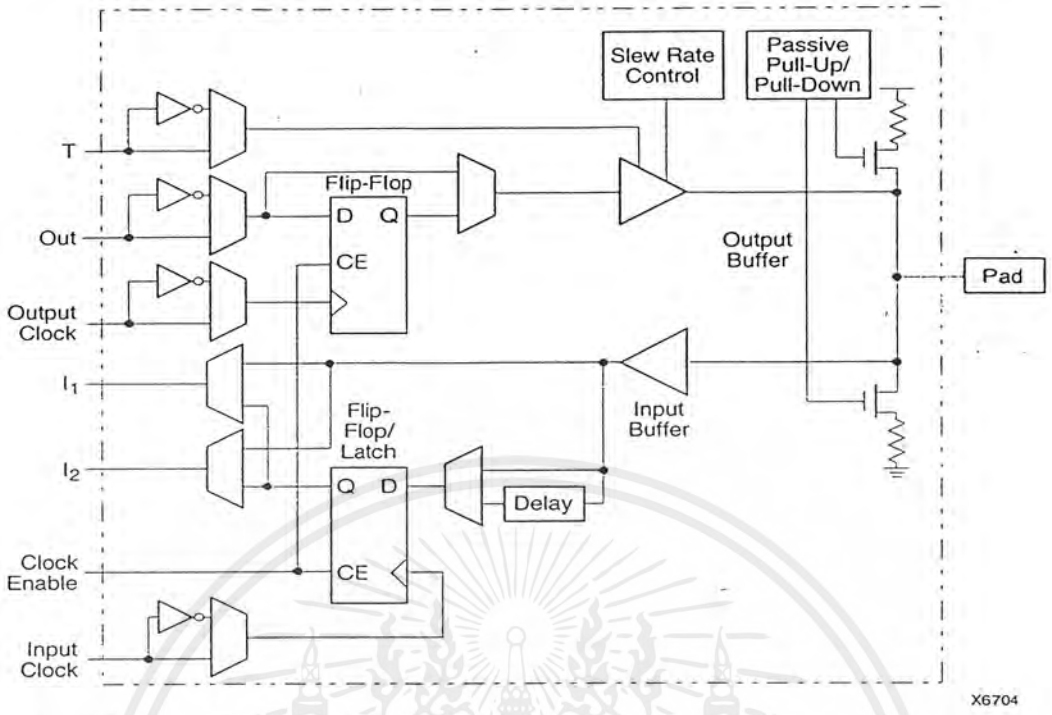


Figure 15: Simplified Block Diagram of XC4000E IOB

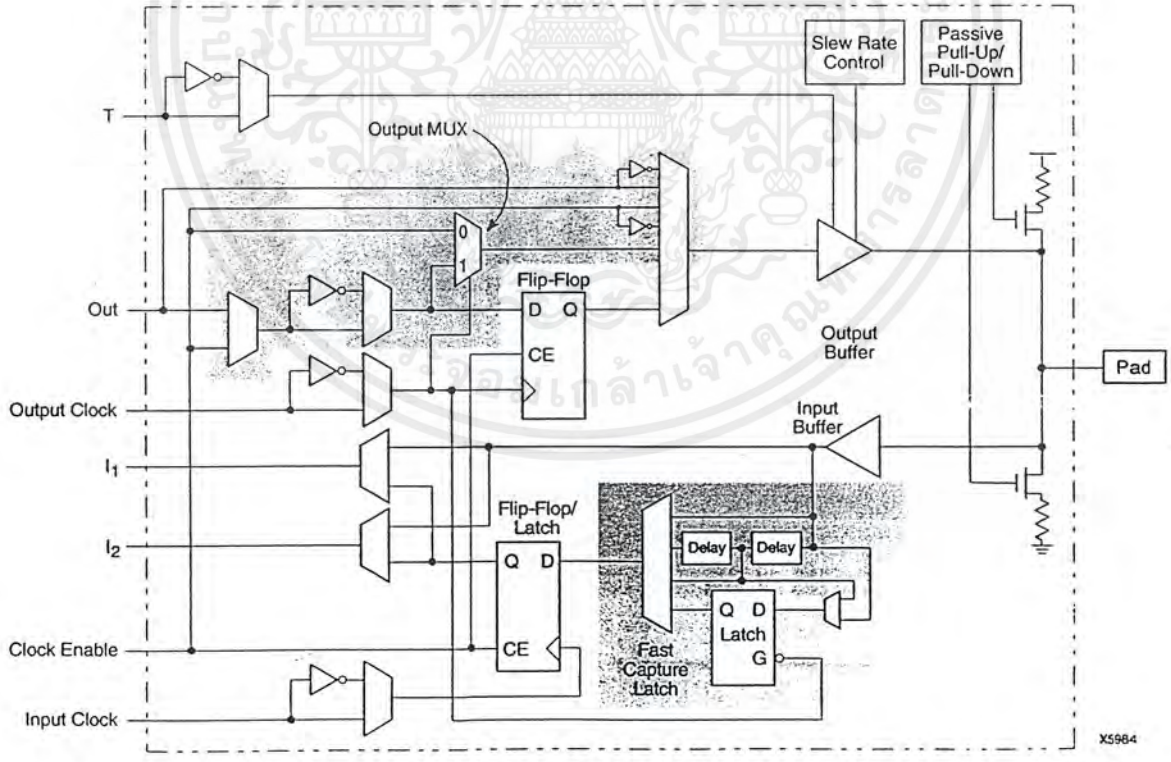


Figure 16: Simplified Block Diagram of XC4000X IOB (shaded areas indicate differences from XC4000E)

Table 8: Supported Sources for XC4000 Series Device Inputs

Source	XC4000E/EX Series Inputs		XC4000XL Series Inputs
	5 V, TTL	5 V, CMOS	3.3 V CMOS
Any device, Vcc = 3.3 V, CMOS outputs	√	Unreli -able Data	√
XC4000 Series, Vcc = 5 V, TTL outputs	√		√
Any device, Vcc = 5 V, TTL outputs (Voh ≤ 3.7 V)	√		√
Any device, Vcc = 5 V, CMOS outputs	√	√	√

XC4000XL 5-Volt Tolerant I/Os

The I/Os on the XC4000XL are fully 5-volt tolerant even though the V_{CC} is 3.3 volts. This allows 5 V signals to directly connect to the XC4000XL inputs without damage, as shown in Table 8. In addition, the 3.3 volt V_{CC} can be applied before or after 5 volt signals are applied to the I/Os. This makes the XC4000XL immune to power supply sequencing problems.

Registered Inputs

The I1 and I2 signals that exit the block can each carry either the direct or registered input signal.

The input and output storage elements in each IOB have a common clock enable input, which, through configuration, can be activated individually for the input or output flip-flop, or both. This clock enable operates exactly like the EC pin on the XC4000 Series CLB. It cannot be inverted within the IOB.

The storage element behavior is shown in Table 9.

Table 9: Input Register Functionality (active rising edge is shown)

Mode	Clock	Clock Enable	D	Q
Power-Up or GSR	X	X	X	SR
Flip-Flop	_/‾	1*	D	D
	0	X	X	Q
Latch	1	1*	X	Q
	0	1*	D	D
Both	X	0	X	Q

Legend:

X	Don't care
_/‾	Rising edge
SR	Set or Reset value. Reset is default.
0*	Input is Low or unconnected (default value)
1*	Input is High or unconnected (default value)

Optional Delay Guarantees Zero Hold Time

The data input to the register can optionally be delayed by several nanoseconds. With the delay enabled, the setup time of the input flip-flop is increased so that normal clock routing does not result in a positive hold-time requirement. A positive hold time requirement can lead to unreliable, temperature- or processing-dependent operation.

The input flip-flop setup time is defined between the data measured at the device I/O pin and the clock input at the IOB (not at the clock pin). Any routing delay from the device clock pin to the clock input of the IOB must, therefore, be subtracted from this setup time to arrive at the real setup time requirement relative to the device pins. A short specified setup time might, therefore, result in a negative setup time at the device pins, i.e., a positive hold-time requirement.

When a delay is inserted on the data line, more clock delay can be tolerated without causing a positive hold-time requirement. Sufficient delay eliminates the possibility of a data hold-time requirement at the external pin. The maximum delay is therefore inserted as the default.

The XC4000E IOB has a one-tap delay element: either the delay is inserted (default), or it is not. The delay guarantees a zero hold time with respect to clocks routed through any of the XC4000E global clock buffers. (See "Global Nets and Buffers (XC4000E only)" on page 35 for a description of the global clock buffers in the XC4000E.) For a shorter input register setup time, with non-zero hold, attach a NODELAY attribute or property to the flip-flop.

The XC4000X IOB has a two-tap delay element, with choices of a full delay, a partial delay, or no delay. The attributes or properties used to select the desired delay are shown in Table 10. The choices are no added attribute, MEDDELAY, and NODELAY. The default setting, with no added attribute, ensures no hold time with respect to any of the XC4000X clock buffers, including the Global Low-Skew buffers. MEDDELAY ensures no hold time with respect to the Global Early buffers. Inputs with NODELAY may have a positive hold time with respect to all clock buffers. For a description of each of these buffers, see "Global Nets and Buffers (XC4000X only)" on page 37.

Table 10: XC4000X IOB Input Delay Element

Value	When to Use
full delay (default, no attribute added)	Zero Hold with respect to Global Low-Skew Buffer, Global Early Buffer
MEDDELAY	Zero Hold with respect to Global Early Buffer
NODELAY	Short Setup, positive Hold time

Additional Input Latch for Fast Capture (XC4000X only)

The XC4000X IOB has an additional optional latch on the input. This latch, as shown in Figure 16, is clocked by the output clock — the clock used for the output flip-flop — rather than the input clock. Therefore, two different clocks can be used to clock the two input storage elements. This additional latch allows the very fast capture of input data, which is then synchronized to the internal clock by the IOB flip-flop or latch.

To use this Fast Capture technique, drive the output clock pin (the Fast Capture latching signal) from the output of one of the Global Early buffers supplied in the XC4000X. The second storage element should be clocked by a Global Low-Skew buffer, to synchronize the incoming data to the internal logic. (See Figure 17.) These special buffers are described in "Global Nets and Buffers (XC4000X only)" on page 37.

The Fast Capture latch (FCL) is designed primarily for use with a Global Early buffer. For Fast Capture, a single clock signal is routed through both a Global Early buffer and a Global Low-Skew buffer. (The two buffers share an input pad.) The Fast Capture latch is clocked by the Global Early buffer, and the standard IOB flip-flop or latch is clocked by the Global Low-Skew buffer. This mode is the safest way to use the Fast Capture latch, because the clock buffers on both storage elements are driven by the same pad. There is no external skew between clock pads to create potential problems.

To place the Fast Capture latch in a design, use one of the special library symbols, ILFFX or ILFLX. ILFFX is a transparent-Low Fast Capture latch followed by an active-High input flip-flop. ILFLX is a transparent-Low Fast Capture latch followed by a transparent-High input latch. Any of the clock inputs can be inverted before driving the library element, and the inverter is absorbed into the IOB. If a single BUFG output is used to drive both clock inputs, the software automatically runs the clock through both a Global Low-Skew buffer and a Global Early buffer, and clocks the Fast Capture latch appropriately.

Figure 16 on page 21 also shows a two-tap delay on the input. By default, if the Fast Capture latch is used, the Xilinx software assumes a Global Early buffer is driving the clock, and selects MEDDELAY to ensure a zero hold time. Select the desired delay based on the discussion in the previous subsection.

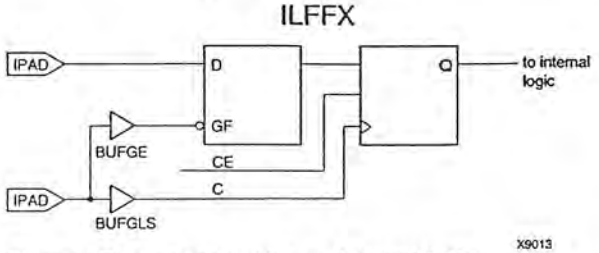


Figure 17: Examples Using XC4000X FCL

IOB Output Signals

Output signals can be optionally inverted within the IOB, and can pass directly to the pad or be stored in an edge-triggered flip-flop. The functionality of this flip-flop is shown in Table 11.

An active-High 3-state signal can be used to place the output buffer in a high-impedance state, implementing 3-state outputs or bidirectional I/O. Under configuration control, the output (OUT) and output 3-state (T) signals can be inverted. The polarity of these signals is independently configured for each IOB.

The 4-mA maximum output current specification of many FPGAs often forces the user to add external buffers, which are especially cumbersome on bidirectional I/O lines. The XC4000E and XC4000EX/XL devices solve many of these problems by providing a guaranteed output sink current of 12 mA. Two adjacent outputs can be interconnected externally to sink up to 24 mA. The XC4000E and XC4000EX/XL FPGAs can thus directly drive buses on a printed circuit board.

By default, the output pull-up structure is configured as a TTL-like totem-pole. The High driver is an n-channel pull-up transistor, pulling to a voltage one transistor threshold below Vcc. Alternatively, the outputs can be globally configured as CMOS drivers, with p-channel pull-up transistors pulling to Vcc. This option, applied using the bitstream generation software, applies to all outputs on the device. It is not individually programmable. In the XC4000XL, all outputs are pulled to the positive supply rail.

Table 11: Output Flip-Flop Functionality (active rising edge is shown)

Mode	Clock	Clock Enable	T	D	Q
Power-Up or GSR	X	X	0*	X	SR
Flip-Flop	X	0	0*	X	Q
	⎍	1*	0*	D	D
	X	X	1	X	Z
	0	X	0*	X	Q

Legend:

X	Don't care
⎍	Rising edge
SR	Set or Reset value. Reset is default.
0*	Input is Low or unconnected (default value)
1*	Input is High or unconnected (default value)
Z	3-state

Any XC4000 Series 5-Volt device with its outputs configured in TTL mode can drive the inputs of any typical 3.3-Volt device. (For a detailed discussion of how to interface between 5 V and 3.3 V devices, see the 3V Products section of *The Programmable Logic Data Book*.)

Supported destinations for XC4000 Series device outputs are shown in Table 12.

An output can be configured as open-drain (open-collector) by placing an OBUFT symbol in a schematic or HDL code, then tying the 3-state pin (T) to the output signal, and the input pin (I) to Ground. (See Figure 18.)

Table 12: Supported Destinations for XC4000 Series Outputs

Destination	XC4000 Series Outputs		
	3.3 V, CMOS	5 V, TTL	5 V, CMOS
Any typical device, Vcc = 3.3 V, CMOS-threshold inputs	√	√	some[1]
Any device, Vcc = 5 V, TTL-threshold inputs	√	√	√
Any device, Vcc = 5 V, CMOS-threshold inputs	Unreliable Data		√

1. Only if destination device has 5-V tolerant inputs

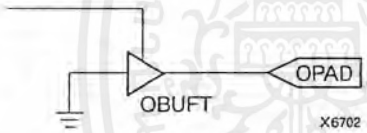


Figure 18: Open-Drain Output

Output Slew Rate

The slew rate of each output buffer is, by default, reduced, to minimize power bus transients when switching non-critical signals. For critical signals, attach a FAST attribute or property to the output buffer or flip-flop.

For XC4000E devices, maximum total capacitive load for simultaneous fast mode switching in the same direction is 200 pF for all package pins between each Power/Ground pin pair. For XC4000X devices, additional internal Power/Ground pin pairs are connected to special Power and Ground planes within the packages, to reduce ground bounce. Therefore, the maximum total capacitive load is 300 pF between each external Power/Ground pin pair. Maximum loading may vary for the low-voltage devices.

For slew-rate limited outputs this total is two times larger for each device type: 400 pF for XC4000E devices and 600 pF for XC4000X devices. This maximum capacitive load should not be exceeded, as it can result in ground bounce of greater than 1.5 V amplitude and more than 5 ns duration. This level of ground bounce may cause undesired transient behavior on an output, or in the internal logic. This restriction is common to all high-speed digital ICs, and is not particular to Xilinx or the XC4000 Series.

XC4000 Series devices have a feature called "Soft Start-up," designed to reduce ground bounce when all outputs are turned on simultaneously at the end of configuration. When the configuration process is finished and the device starts up, the first activation of the outputs is automatically slew-rate limited. Immediately following the initial activation of the I/O, the slew rate of the individual outputs is determined by the individual configuration option for each IOB.

Global Three-State

A separate Global 3-State line (not shown in Figure 15 or Figure 16) forces all FPGA outputs to the high-impedance state, unless boundary scan is enabled and is executing an EXTEST instruction. This global net (GTS) does not compete with other routing resources; it uses a dedicated distribution network.

GTS can be driven from any user-programmable pin as a global 3-state input. To use this global net, place an input pad and input buffer in the schematic or HDL code, driving the GTS pin of the STARTUP symbol. A specific pin location can be assigned to this input using a LOC attribute or property, just as with any other user-programmable pad. An inverter can optionally be inserted after the input buffer to invert the sense of the Global 3-State signal. Using GTS is similar to GSR. See Figure 2 on page 11 for details.

Alternatively, GTS can be driven from any internal node.

Output Multiplexer/2-Input Function Generator (XC4000X only)

As shown in Figure 16 on page 21, the output path in the XC4000X IOB contains an additional multiplexer not available in the XC4000E IOB. The multiplexer can also be configured as a 2-input function generator, implementing a pass-gate, AND-gate, OR-gate, or XOR-gate, with 0, 1, or 2 inverted inputs. The logic used to implement these functions is shown in the upper gray area of Figure 16.

When configured as a multiplexer, this feature allows two output signals to time-share the same output pad; effectively doubling the number of device outputs without requiring a larger, more expensive package.

When the MUX is configured as a 2-input function generator, logic can be implemented within the IOB itself. Combined with a Global Early buffer, this arrangement allows very high-speed gating of a single signal. For example, a wide decoder can be implemented in CLBs, and its output gated with a Read or Write Strobe Driven by a BUFGE buffer, as shown in Figure 19. The critical-path pin-to-pin delay of this circuit is less than 6 nanoseconds.

As shown in Figure 16, the IOB input pins Out, Output Clock, and Clock Enable have different delays and different flexibilities regarding polarity. Additionally, Output Clock sources are more limited than the other inputs. Therefore, the Xilinx software does not move logic into the IOB function generators unless explicitly directed to do so.

The user can specify that the IOB function generator be used, by placing special library symbols beginning with the letter "O." For example, a 2-input AND-gate in the IOB function generator is called OAND2. Use the symbol input pin labelled "F" for the signal on the critical path. This signal is placed on the OK pin — the IOB input with the shortest delay to the function generator. Two examples are shown in Figure 20.

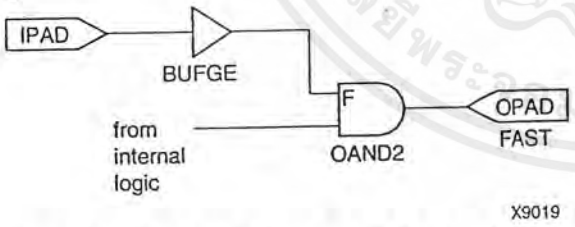


Figure 19: Fast Pin-to-Pin Path in XC4000X

Figure 20: AND & MUX Symbols in XC4000X IOB

Other IOB Options

There are a number of other programmable options in the XC4000 Series IOB.

Pull-up and Pull-down Resistors

Programmable pull-up and pull-down resistors are useful for tying unused pins to Vcc or Ground to minimize power consumption and reduce noise sensitivity. The configurable pull-up resistor is a p-channel transistor that pulls to Vcc. The configurable pull-down resistor is an n-channel transistor that pulls to Ground.

The value of these resistors is 50 kΩ – 100 kΩ. This high value makes them unsuitable as wired-AND pull-up resistors.

The pull-up resistors for most user-programmable IOBs are active during the configuration process. See Table 22 on page 58 for a list of pins with pull-ups active before and during configuration.

After configuration, voltage levels of unused pads, bonded or un-bonded, must be valid logic levels, to reduce noise sensitivity and avoid excess current. Therefore, by default, unused pads are configured with the internal pull-up resistor active. Alternatively, they can be individually configured with the pull-down resistor, or as a driven output, or to be driven by an external source. To activate the internal pull-up, attach the PULLUP library component to the net attached to the pad. To activate the internal pull-down, attach the PULLDOWN library component to the net attached to the pad.

Independent Clocks

Separate clock signals are provided for the input and output flip-flops. The clock can be independently inverted for each flip-flop within the IOB, generating either falling-edge or rising-edge triggered flip-flops. The clock inputs for each IOB are independent, except that in the XC4000X, the Fast Capture latch shares an IOB input with the output clock pin.

Early Clock for IOBs (XC4000X only)

Special early clocks are available for IOBs. These clocks are sourced by the same sources as the Global Low-Skew buffers, but are separately buffered. They have fewer loads and therefore less delay. The early clock can drive either the IOB output clock or the IOB input clock, or both. The early clock allows fast capture of input data, and fast clock-to-output on output data. The Global Early buffers that drive these clocks are described in "Global Nets and Buffers (XC4000X only)" on page 37.

Global Set/Reset

As with the CLB registers, the Global Set/Reset signal (GSR) can be used to set or clear the input and output registers, depending on the value of the INIT attribute or property. The two flip-flops can be individually configured to set

or clear on reset and after configuration. Other than the global GSR net, no user-controlled set/reset signal is available to the I/O flip-flops. The choice of set or clear applies to both the initial state of the flip-flop and the response to the Global Set/Reset pulse. See "Global Set/Reset" on page 11 for a description of how to use GSR.

JTAG Support

Embedded logic attached to the IOBs contains test structures compatible with IEEE Standard 1149.1 for boundary scan testing, permitting easy chip and board-level testing. More information is provided in "Boundary Scan" on page 42.

Three-State Buffers

A pair of 3-state buffers is associated with each CLB in the array. (See Figure 27 on page 30.) These 3-state buffers can be used to drive signals onto the nearest horizontal longlines above and below the CLB. They can therefore be used to implement multiplexed or bidirectional buses on the horizontal longlines, saving logic resources. Programmable pull-up resistors attached to these longlines help to implement a wide wired-AND function.

The buffer enable is an active-High 3-state (i.e. an active-Low enable), as shown in Table 13.

Another 3-state buffer with similar access is located near each I/O block along the right and left edges of the array. (See Figure 33 on page 34.)

The horizontal longlines driven by the 3-state buffers have a weak keeper at each end. This circuit prevents undefined floating levels. However, it is overridden by any driver, even a pull-up resistor.

Special longlines running along the perimeter of the array can be used to wire-AND signals coming from nearby IOBs or from internal longlines. These longlines form the wide edge decoders discussed in "Wide Edge Decoders" on page 27.

Three-State Buffer Modes

The 3-state buffers can be configured in three modes:

- Standard 3-state buffer
- Wired-AND with input on the I pin
- Wired OR-AND

Standard 3-State Buffer

All three pins are used. Place the library element BUFT. Connect the input to the I pin and the output to the O pin. The T pin is an active-High 3-state (i.e. an active-Low enable). Tie the T pin to Ground to implement a standard buffer.

Wired-AND with Input on the I Pin

The buffer can be used as a Wired-AND. Use the WAND1 library symbol, which is essentially an open-drain buffer. WAND4, WAND8, and WAND16 are also available. See the *XACT Libraries Guide* for further information.

The T pin is internally tied to the I pin. Connect the input to the I pin and the output to the O pin. Connect the outputs of all the WAND1s together and attach a PULLUP symbol.

Wired OR-AND

The buffer can be configured as a Wired OR-AND. A High level on either input turns off the output. Use the WOR2AND library symbol, which is essentially an open-drain 2-input OR gate. The two input pins are functionally equivalent. Attach the two inputs to the I0 and I1 pins and tie the output to the O pin. Tie the outputs of all the WOR2ANDs together and attach a PULLUP symbol.

Three-State Buffer Examples

Figure 21 shows how to use the 3-state buffers to implement a wired-AND function. When all the buffer inputs are High, the pull-up resistor(s) provide the High output.

Figure 22 shows how to use the 3-state buffers to implement a multiplexer. The selection is accomplished by the buffer 3-state signal.

Pay particular attention to the polarity of the T pin when using these buffers in a design. Active-High 3-state (T) is identical to an active-Low output enable, as shown in Table 13.

Table 13: Three-State Buffer Functionality

IN	T	OUT
X	1	Z
IN	0	IN

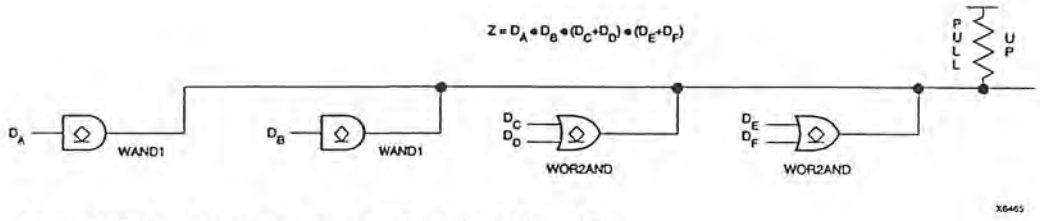


Figure 21: Open-Drain Buffers Implement a Wired-AND Function

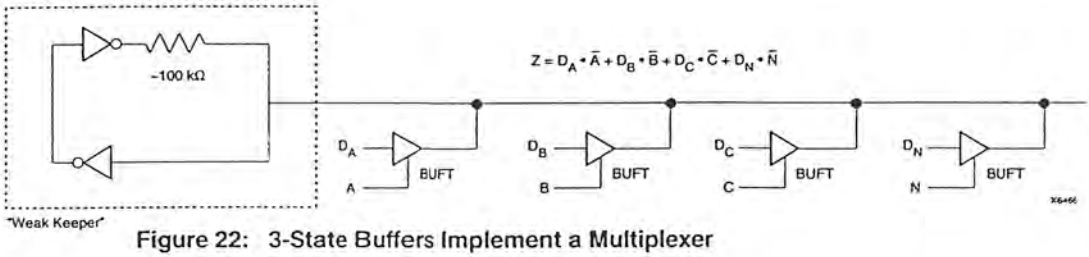


Figure 22: 3-State Buffers Implement a Multiplexer

Wide Edge Decoders

Dedicated decoder circuitry boosts the performance of wide decoding functions. When the address or data field is wider than the function generator inputs, FPGAs need multi-level decoding and are thus slower than PALs. XC4000 Series CLBs have nine inputs. Any decoder of up to nine inputs is, therefore, compact and fast. However, there is also a need for much wider decoders, especially for address decoding in large microprocessor systems.

An XC4000 Series FPGA has four programmable decoders located on each edge of the device. The inputs to each decoder are any of the IOB I1 signals on that edge plus one local interconnect per CLB row or column. Each row or column of CLBs provides up to three variables or their compliments., as shown in Figure 23. Each decoder generates a High output (resistor pull-up) when the AND condition of the selected inputs, or their complements, is true. This is analogous to a product term in typical PAL devices.

Each of these wired-AND gates is capable of accepting up to 42 inputs on the XC4005E and 72 on the XC4013E. There are up to 96 inputs for each decoder on the XC4028X and 132 on the XC4052X. The decoders may also be split in two when a larger number of narrower decoders are required, for a maximum of 32 decoders per device.

The decoder outputs can drive CLB inputs, so they can be combined with other logic to form a PAL-like AND/OR structure. The decoder outputs can also be routed directly to the chip outputs. For fastest speed, the output should be on the same chip edge as the decoder. Very large PALs can be emulated by ORing the decoder outputs in a CLB. This decoding feature covers what has long been considered a weakness of older FPGAs. Users often resorted to external PALs for simple but fast decoding functions. Now, the dedicated decoders in the XC4000 Series device can implement these functions fast and efficiently.

To use the wide edge decoders, place one or more of the WAND library symbols (WAND1, WAND4, WAND8, WAND16). Attach a DECODE attribute or property to each WAND symbol. Tie the outputs together and attach a PULLUP symbol. Location attributes or properties such as L (left edge) or TR (right half of top edge) should also be used to ensure the correct placement of the decoder inputs.

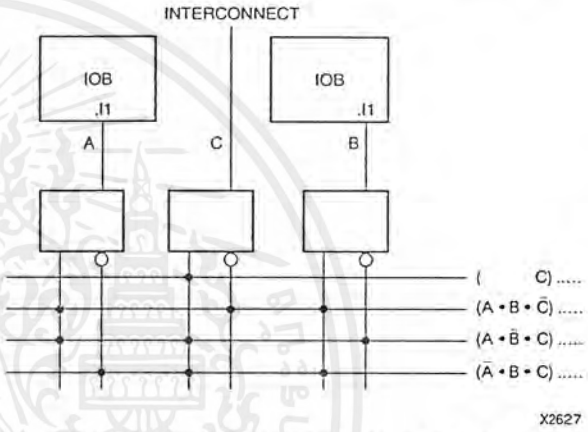


Figure 23: XC4000 Series Edge Decoding Example

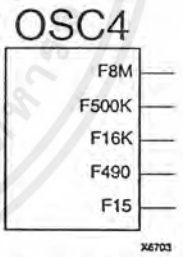


Figure 24: XC4000 Series Oscillator Symbol

On-Chip Oscillator

XC4000 Series devices include an internal oscillator. This oscillator is used to clock the power-on time-out, for configuration memory clearing, and as the source of CCLK in Master configuration modes. The oscillator runs at a nominal 8 MHz frequency that varies with process, Vcc, and temperature. The output frequency falls between 4 and 10 MHz.

The oscillator output is optionally available after configuration. Any two of four resynchronized taps of a built-in divider are also available. These taps are at the fourth, ninth, fourteenth and nineteenth bits of the divider. Therefore, if the primary oscillator output is running at the nominal 8 MHz, the user has access to an 8 MHz clock, plus any two of 500 kHz, 16kHz, 490Hz and 15Hz (up to 10% lower for low-voltage devices). These frequencies can vary by as much as -50% or +25%.

These signals can be accessed by placing the OSC4 library element in a schematic or in HDL code (see Figure 24).

The oscillator is automatically disabled after configuration if the OSC4 symbol is not used in the design.

Programmable Interconnect

All internal connections are composed of metal segments with programmable switching points and switching matrices to implement the desired routing. A structured, hierarchical matrix of routing resources is provided to achieve efficient automated routing.

The XC4000E and XC4000X share a basic interconnect structure. XC4000X devices, however, have additional routing not available in the XC4000E. The extra routing resources allow high utilization in high-capacity devices. All XC4000X-specific routing resources are clearly identified throughout this section. Any resources not identified as XC4000X-specific are present in all XC4000 Series devices.

This section describes the varied routing resources available in XC4000 Series devices. The implementation software automatically assigns the appropriate resources based on the density and timing requirements of the design.

Interconnect Overview

There are several types of interconnect.

- CLB routing is associated with each row and column of the CLB array.
- IOB routing forms a ring (called a VersaRing) around the outside of the CLB array. It connects the I/O with the internal logic blocks.
- Global routing consists of dedicated networks primarily designed to distribute clocks throughout the device with minimum delay and skew. Global routing can also be used for other high-fanout signals.

Five interconnect types are distinguished by the relative length of their segments: single-length lines, double-length lines, quad and octal lines (XC4000X only), and longlines. In the XC4000X, direct connects allow fast data flow between adjacent CLBs, and between IOBs and CLBs.

Extra routing is included in the IOB pad ring. The XC4000X also includes a ring of octal interconnect lines near the IOBs to improve pin-swapping and routing to locked pins.

XC4000E/X devices include two types of global buffers. These global buffers have different properties, and are intended for different purposes. They are discussed in detail later in this section.

CLB Routing Connections

A high-level diagram of the routing resources associated with one CLB is shown in Figure 25. The shaded arrows represent routing present only in XC4000X devices.

Table 14 shows how much routing of each type is available in XC4000E and XC4000X CLB arrays. Clearly, very large designs, or designs with a great deal of interconnect, will route more easily in the XC4000X. Smaller XC4000E designs, typically requiring significantly less interconnect, do not require the additional routing.

Figure 27 on page 30 is a detailed diagram of both the XC4000E and the XC4000X CLB, with associated routing. The shaded square is the programmable switch matrix, present in both the XC4000E and the XC4000X. The L-shaped shaded area is present only in XC4000X devices. As shown in the figure, the XC4000X block is essentially an XC4000E block with additional routing.

CLB inputs and outputs are distributed on all four sides, providing maximum routing flexibility. In general, the entire architecture is symmetrical and regular. It is well suited to established placement and routing algorithms. Inputs, outputs, and function generators can freely swap positions within a CLB to avoid routing congestion during the placement and routing operation.

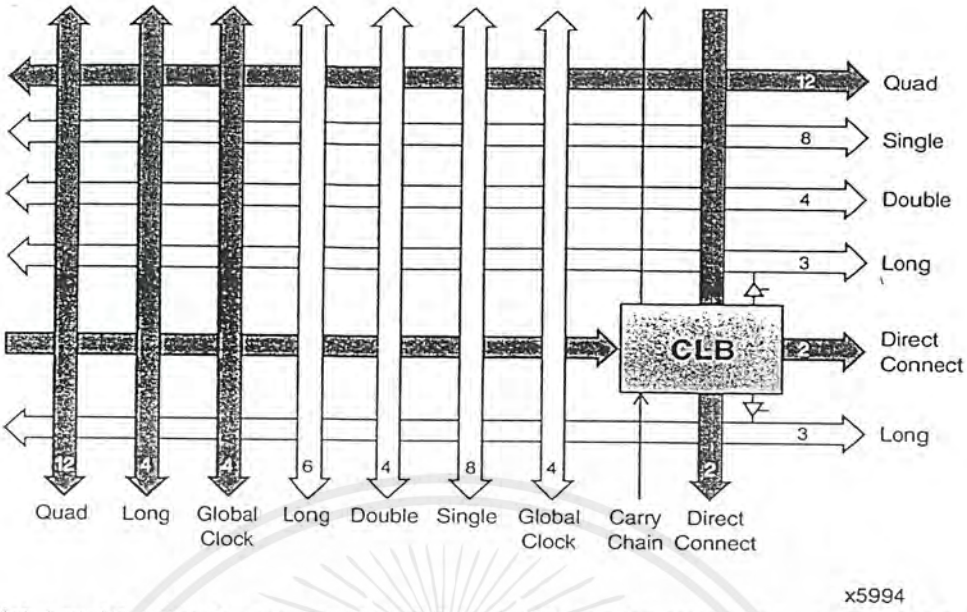


Figure 25: High-Level Routing Diagram of XC4000 Series CLB (shaded arrows indicate XC4000X only)

Table 14: Routing per CLB in XC4000 Series Devices

	XC4000E		XC4000X	
	Vertical	Horizontal	Vertical	Horizontal
Singles	8	8	8	8
Doubles	4	4	4	4
Quads	0	0	12	12
Longlines	6	6	10	6
Direct Connects	0	0	2	2
Globals	4	0	8	0
Carry Logic	2	0	1	0
Total	24	18	45	32

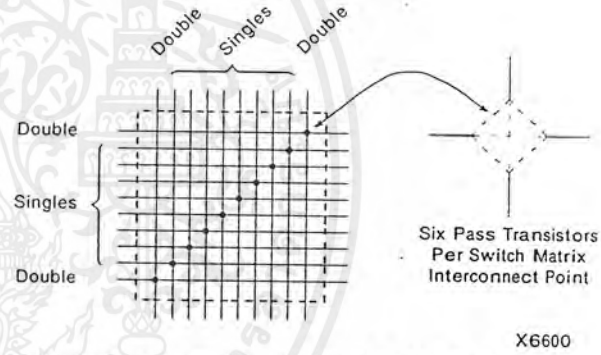


Figure 26: Programmable Switch Matrix (PSM)

Programmable Switch Matrices

The horizontal and vertical single- and double-length lines intersect at a box called a programmable switch matrix (PSM). Each switch matrix consists of programmable pass transistors used to establish connections between the lines (see Figure 26).

For example, a single-length signal entering on the right side of the switch matrix can be routed to a single-length line on the top, left, or bottom sides, or any combination thereof, if multiple branches are required. Similarly, a double-length signal can be routed to a double-length line on any or all of the other three edges of the programmable switch matrix.

Single-Length Lines

Single-length lines provide the greatest interconnect flexibility and offer fast routing between adjacent blocks. There are eight vertical and eight horizontal single-length lines associated with each CLB. These lines connect the switching matrices that are located in every row and a column of CLBs.

Single-length lines are connected by way of the programmable switch matrices, as shown in Figure 28. Routing connectivity is shown in Figure 27.

Single-length lines incur a delay whenever they go through a switching matrix. Therefore, they are not suitable for routing signals for long distances. They are normally used to conduct signals within a localized area and to provide the branching for nets with fanout greater than one.

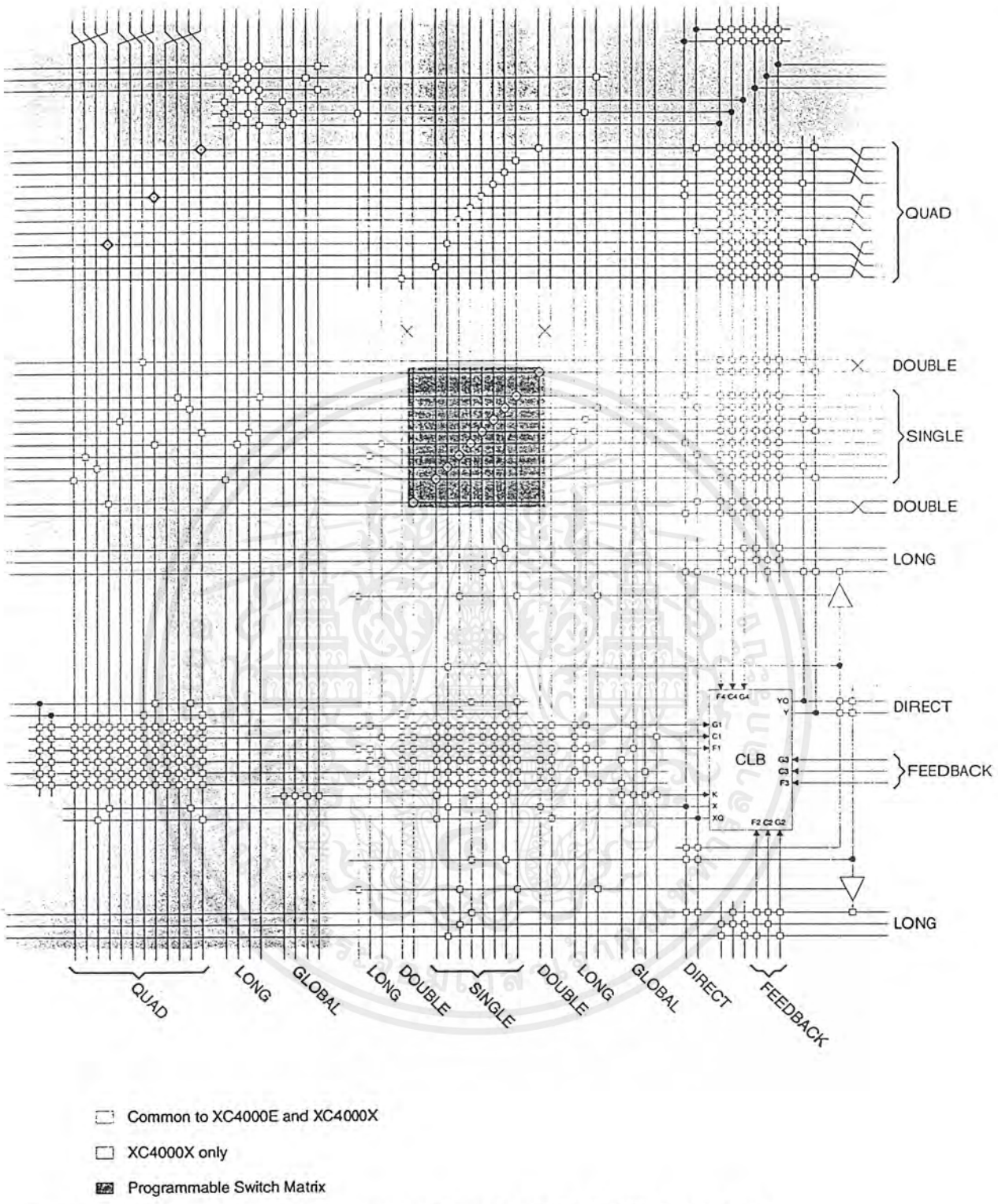


Figure 27: Detail of Programmable Interconnect Associated with XC4000 Series CLB

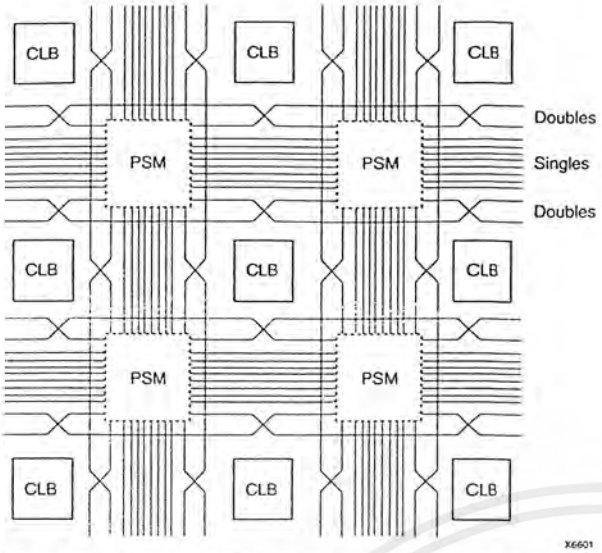


Figure 28: Single- and Double-Length Lines, with Programmable Switch Matrices (PSMs)

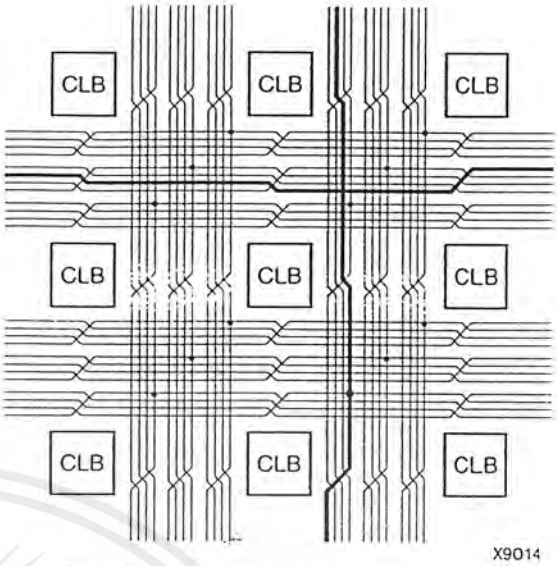


Figure 29: Quad Lines (XC4000X only)

Double-Length Lines

The double-length lines consist of a grid of metal segments, each twice as long as the single-length lines: they run past two CLBs before entering a switch matrix. Double-length lines are grouped in pairs with the switch matrices staggered, so that each line goes through a switch matrix at every other row or column of CLBs (see Figure 28).

There are four vertical and four horizontal double-length lines associated with each CLB. These lines provide faster signal routing over intermediate distances, while retaining routing flexibility. Double-length lines are connected by way of the programmable switch matrices. Routing connectivity is shown in Figure 27.

Quad Lines (XC4000X only)

XC4000X devices also include twelve vertical and twelve horizontal quad lines per CLB row and column. Quad lines are four times as long as the single-length lines. They are interconnected via buffered switch matrices (shown as diamonds in Figure 27 on page 30). Quad lines run past four CLBs before entering a buffered switch matrix. They are grouped in fours, with the buffered switch matrices staggered, so that each line goes through a buffered switch matrix at every fourth CLB location in that row or column. (See Figure 29.)

The buffered switch matrixes have four pins, one on each edge. All of the pins are bidirectional. Any pin can drive any or all of the other pins.

Each buffered switch matrix contains one buffer and six pass transistors. It resembles the programmable switch matrix shown in Figure 26, with the addition of a programmable buffer. There can be up to two independent inputs and up to two independent outputs. Only one of the independent inputs can be buffered.

The place and route software automatically uses the timing requirements of the design to determine whether or not a quad line signal should be buffered. A heavily loaded signal is typically buffered, while a lightly loaded one is not. One scenario is to alternate buffers and pass transistors. This allows both vertical and horizontal quad lines to be buffered at alternating buffered switch matrices.

Due to the buffered switch matrices, quad lines are very fast. They provide the fastest available method of routing heavily loaded signals for long distances across the device.

Longlines

Longlines form a grid of metal interconnect segments that run the entire length or width of the array. Longlines are intended for high fan-out, time-critical signal nets, or nets that are distributed over long distances. In XC4000X devices, quad lines are preferred for critical nets, because the buffered switch matrices make them faster for high fan-out nets.

Two horizontal longlines per CLB can be driven by 3-state or open-drain drivers (TBUFs). They can therefore implement unidirectional or bidirectional buses, wide multiplexers, or wired-AND functions. (See "Three-State Buffers" on page 26 for more details.)

Each horizontal longline driven by TBUFs has either two (XC4000E) or eight (XC4000X) pull-up resistors. To activate these resistors, attach a PULLUP symbol to the long-line net. The software automatically activates the appropriate number of pull-ups. There is also a weak keeper at each end of these two horizontal longlines. This

circuit prevents undefined floating levels. However, it is overridden by any driver, even a pull-up resistor.

Each XC4000E longline has a programmable splitter switch at its center, as does each XC4000X longline driven by TBUFs. This switch can separate the line into two independent routing channels, each running half the width or height of the array.

Each XC4000X longline not driven by TBUFs has a buffered programmable splitter switch at the 1/4, 1/2, and 3/4 points of the array. Due to the buffering, XC4000X longline performance does not deteriorate with the larger array sizes. If the longline is split, the resulting partial longlines are independent.

Routing connectivity of the longlines is shown in Figure 27 on page 30.

Direct Interconnect (XC4000X only)

The XC4000X offers two direct, efficient and fast connections between adjacent CLBs. These nets facilitate a data flow from the left to the right side of the device, or from the top to the bottom, as shown in Figure 30. Signals routed on the direct interconnect exhibit minimum interconnect propagation delay and use no general routing resources.

The direct interconnect is also present between CLBs and adjacent IOBs. Each IOB on the left and top device edges has a direct path to the nearest CLB. Each CLB on the right and bottom edges of the array has a direct path to the nearest two IOBs, since there are two IOBs for each row or column of CLBs.

The place and route software uses direct interconnect whenever possible, to maximize routing resources and minimize interconnect delays.

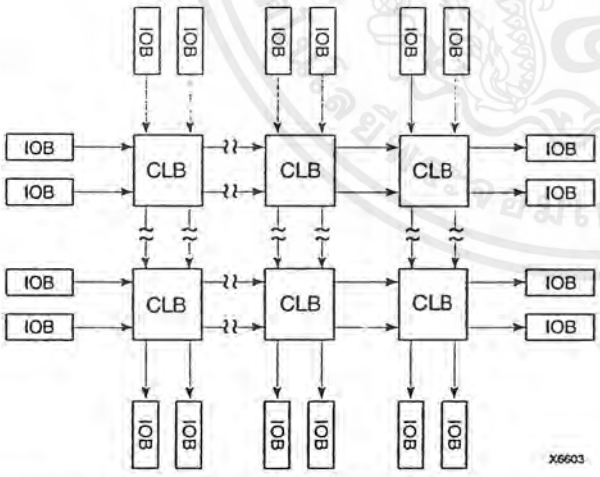


Figure 30: XC4000X Direct Interconnect

I/O Routing

XC4000 Series devices have additional routing around the IOB ring. This routing is called a VersaRing. The VersaRing facilitates pin-swapping and redesign without affecting board layout. Included are eight double-length lines spanning two CLBs (four IOBs), and four longlines. Global lines and Wide Edge Decoder lines are provided. XC4000X devices also include eight octal lines.

A high-level diagram of the VersaRing is shown in Figure 31. The shaded arrows represent routing present only in XC4000X devices.

Figure 33 on page 34 is a detailed diagram of the XC4000E and XC4000X VersaRing. The area shown includes two IOBs. There are two IOBs per CLB row or column, therefore this diagram corresponds to the CLB routing diagram shown in Figure 27 on page 30. The shaded areas represent routing and routing connections present only in XC4000X devices.

Octal I/O Routing (XC4000X only)

Between the XC4000X CLB array and the pad ring, eight interconnect tracks provide for versatility in pin assignment and fixed pinout flexibility. (See Figure 32 on page 33.)

These routing tracks are called octals, because they can be broken every eight CLBs (sixteen IOBs) by a programmable buffer that also functions as a splitter switch. The buffers are staggered, so each line goes through a buffer at every eighth CLB location around the device edge.

The octal lines bend around the corners of the device. The lines cross at the corners in such a way that the segment most recently buffered before the turn has the farthest distance to travel before the next buffer, as shown in Figure 32.

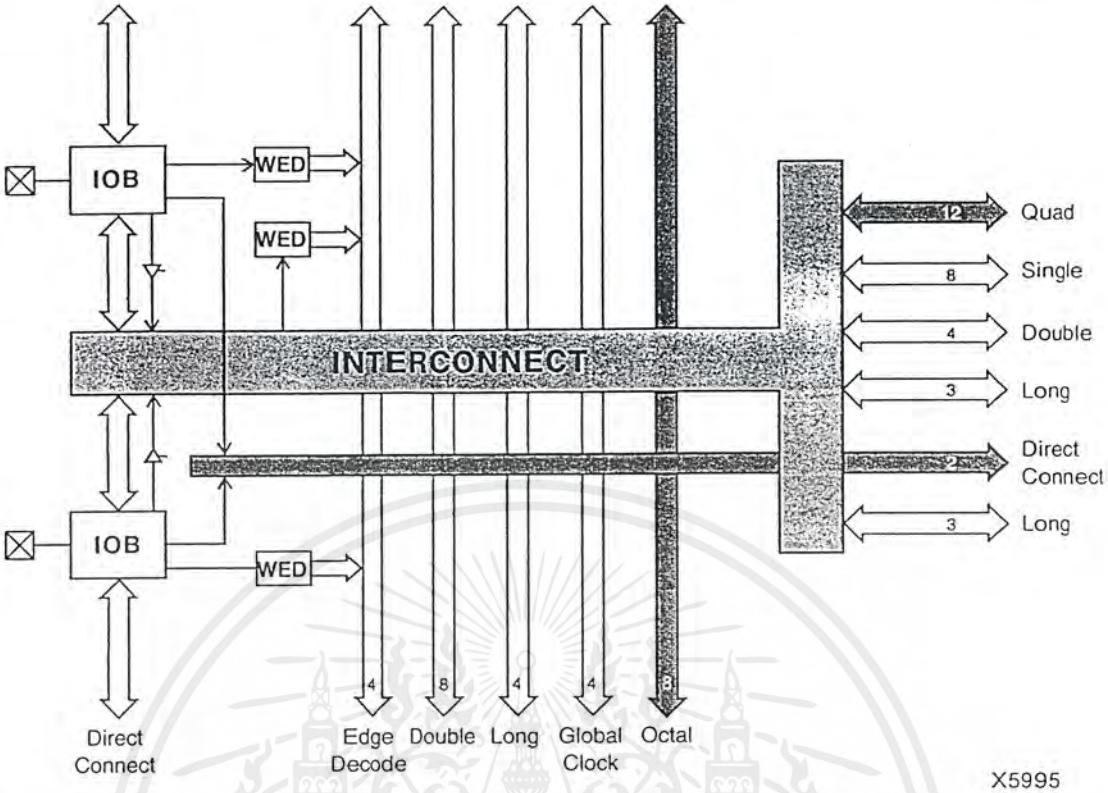


Figure 31: High-Level Routing Diagram of XC4000 Series VersaRing (Left Edge)
WED = Wide Edge Decoder, IOB = I/O Block (shaded arrows indicate XC4000X only)

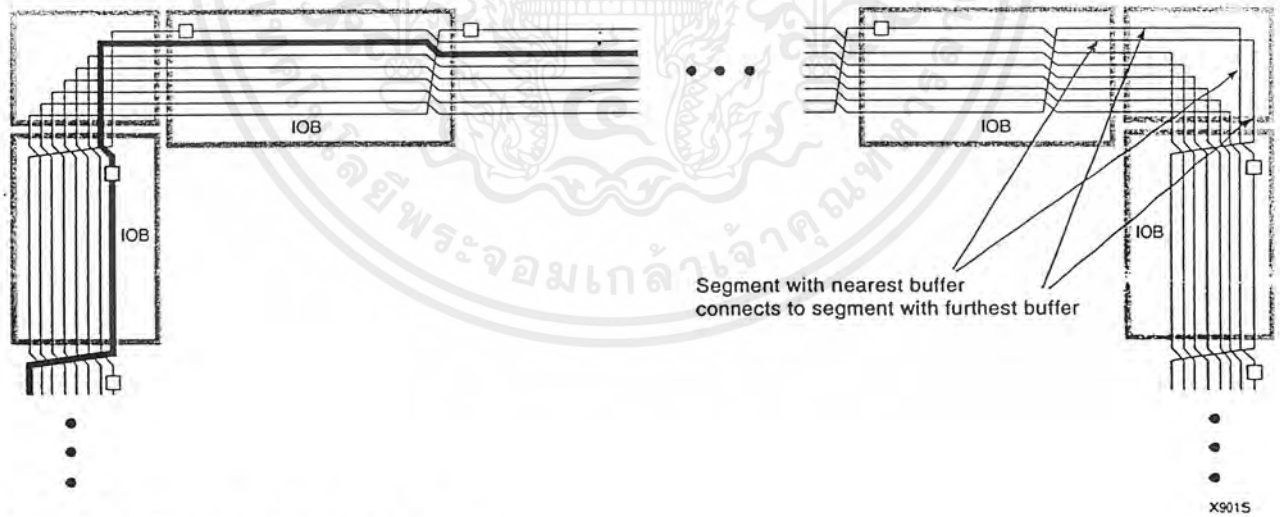
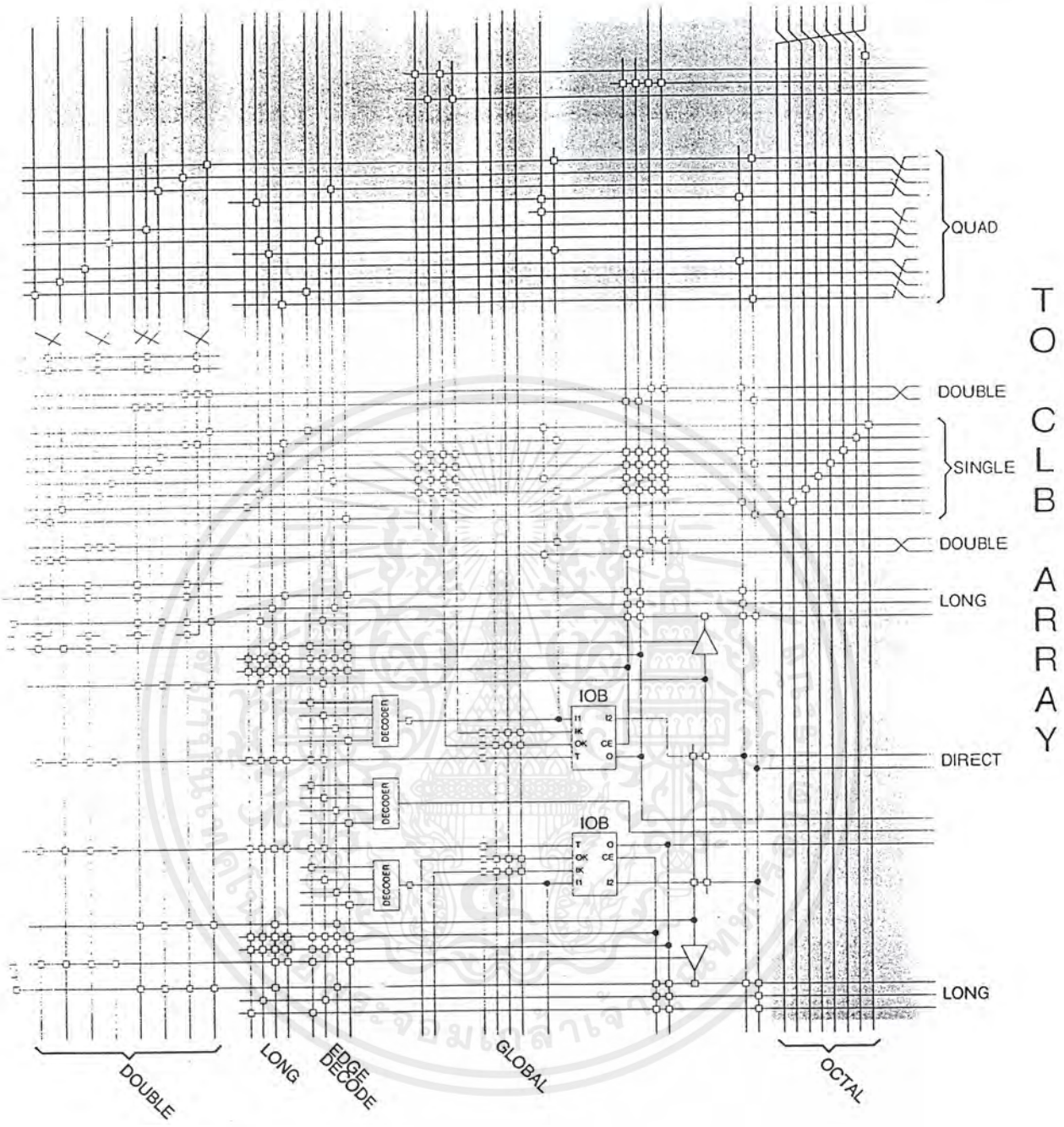


Figure 32: XC4000X Octal I/O Routing



Common to XC4000E and XC4000X

XC4000X only

Figure 33: Detail of Programmable Interconnect Associated with XC4000 Series IOB (Left Edge)

IOB inputs and outputs interface with the octal lines via the single-length interconnect lines. Single-length lines are also used for communication between the octals and double-length lines, quads, and longlines within the CLB array.

Segmentation into buffered octals was found to be optimal for distributing signals over long distances around the device.

Global Nets and Buffers

Both the XC4000E and the XC4000X have dedicated global networks. These networks are designed to distribute clocks and other high fanout control signals throughout the devices with minimal skew. The global buffers are described in detail in the following sections. The text descriptions and diagrams are summarized in Table 15. The table shows which CLB and IOB clock pins can be sourced by which global buffers.

In both XC4000E and XC4000X devices, placement of a library symbol called BUFG results in the software choosing the appropriate clock buffer, based on the timing requirements of the design. The detailed information in these sections is included only for reference.

Global Nets and Buffers (XC4000E only)

Four vertical longlines in each CLB column are driven exclusively by special global buffers. These longlines are in addition to the vertical longlines used for standard interconnect. The four global lines can be driven by either of two types of global buffers. The clock pins of every CLB and IOB can also be sourced from local interconnect.

Two different types of clock buffers are available in the XC4000E:

- Primary Global Buffers (BUFGP)
- Secondary Global Buffers (BUFGS)

Four Primary Global buffers offer the shortest delay and negligible skew. Four Secondary Global buffers have slightly longer delay and slightly more skew due to potentially heavier loading, but offer greater flexibility when used to drive non-clock CLB inputs.

The Primary Global buffers must be driven by the semi-dedicated pads. The Secondary Global buffers can be sourced by either semi-dedicated pads or internal nets.

Each CLB column has four dedicated vertical Global lines. Each of these lines can be accessed by one particular Primary Global buffer, or by any of the Secondary Global buffers, as shown in Figure 34. Each corner of the device has one Primary buffer and one Secondary buffer.

IOBs along the left and right edges have four vertical global longlines. Top and bottom IOBs can be clocked from the global lines in the adjacent CLB column.

A global buffer should be specified for all timing-sensitive global signal distribution. To use a global buffer, place a BUFGP (primary buffer), BUFGS (secondary buffer), or BUFG (either primary or secondary buffer) element in a schematic or in HDL code. If desired, attach a LOC attribute or property to direct placement to the designated location. For example, attach a LOC=L attribute or property to a BUFGS symbol to direct that a buffer be placed in one of the two Secondary Global buffers on the left edge of the device, or a LOC=BL to indicate the Secondary Global buffer on the bottom edge of the device, on the left.

Table 15: Clock Pin Access

	XC4000E		XC4000X			Local Interconnect
	BUFGP	BUFGS	BUFGLS	L & R BUFGE	T & B BUFGE	
All CLBs in Quadrant	√	√	√	√	√	√
All CLBs in Device	√	√	√			√
IOBs on Adjacent Vertical Half Edge	√	√	√	√	√	√
IOBs on Adjacent Vertical Full Edge	√	√	√	√		√
IOBs on Adjacent Horizontal Half Edge (Direct)				√		√
IOBs on Adjacent Horizontal Half Edge (through CLB globals)	√	√	√	√	√	√
IOBs on Adjacent Horizontal Full Edge (through CLB globals)	√	√	√			√

L = Left, R = Right, T = Top, B = Bottom

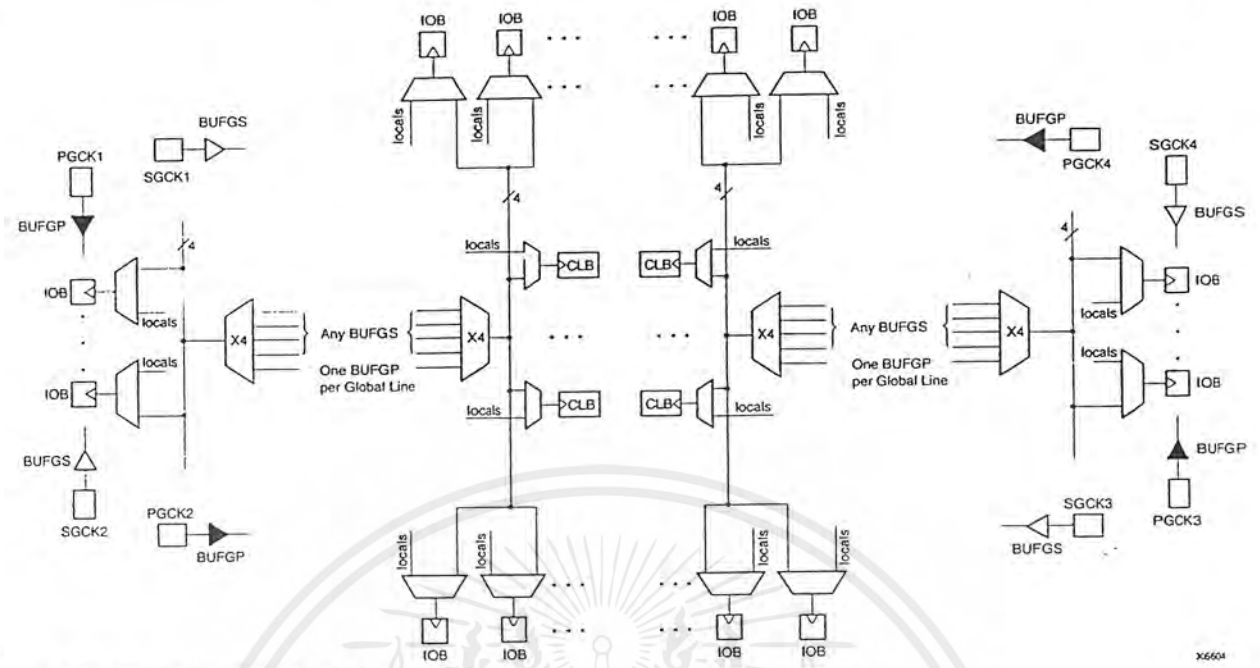


Figure 34: XC4000E Global Net Distribution

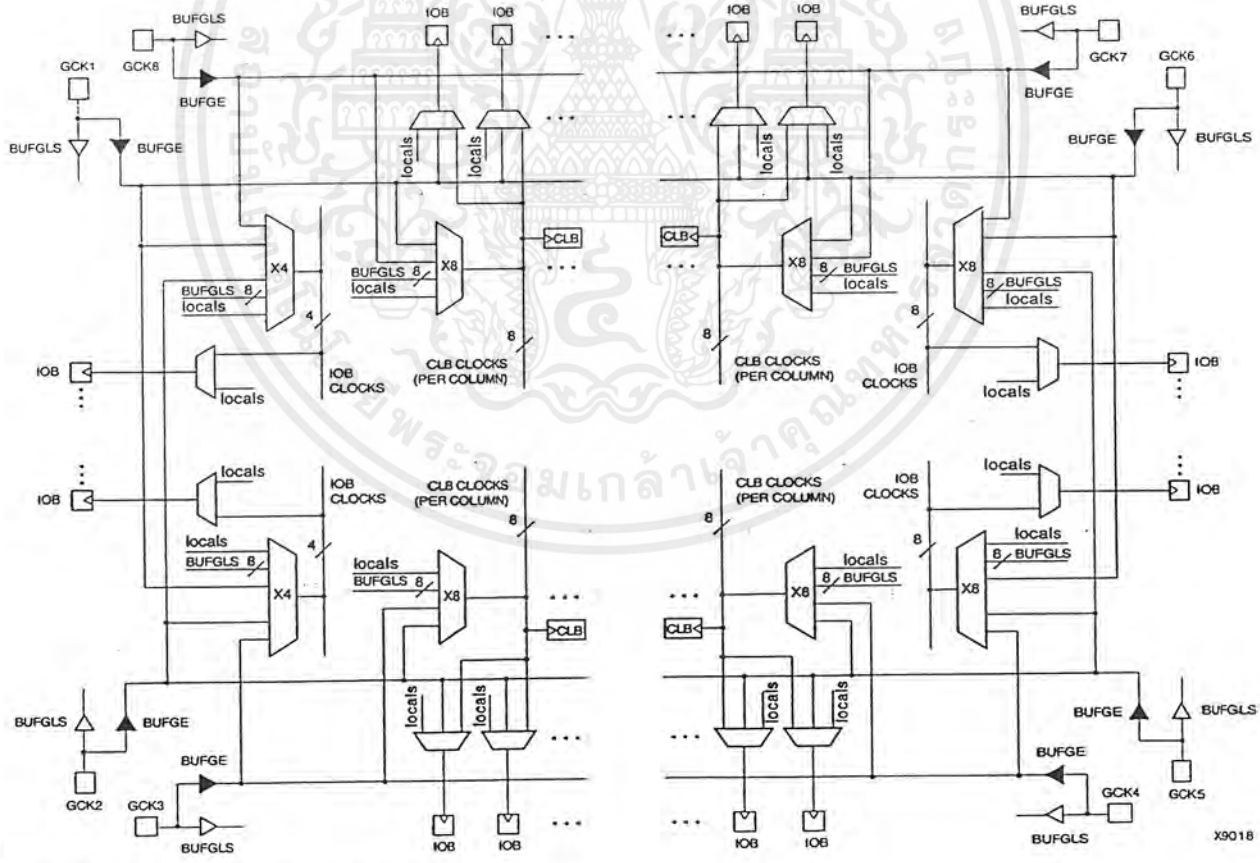


Figure 35: XC4000X Global Net Distribution

Global Nets and Buffers (XC4000X only)

Eight vertical longlines in each CLB column are driven by special global buffers. These longlines are in addition to the vertical longlines used for standard interconnect. The global lines are broken in the center of the array, to allow faster distribution and to minimize skew across the whole array. Each half-column global line has its own buffered multiplexer, as shown in Figure 35. The top and bottom global lines cannot be connected across the center of the device, as this connection might introduce unacceptable skew. The top and bottom halves of the global lines must be separately driven — although they can be driven by the same global buffer.

The eight global lines in each CLB column can be driven by either of two types of global buffers. They can also be driven by internal logic, because they can be accessed by single, double, and quad lines at the top, bottom, half, and quarter points. Consequently, the number of different clocks that can be used simultaneously in an XC4000X device is very large.

There are four global lines feeding the IOBs at the left edge of the device. IOBs along the right edge have eight global lines. There is a single global line along the top and bottom edges with access to the IOBs. All IOB global lines are broken at the center. They cannot be connected across the center of the device, as this connection might introduce unacceptable skew.

IOB global lines can be driven from two types of global buffers, or from local interconnect. Alternatively, top and bottom IOBs can be clocked from the global lines in the adjacent CLB column.

Two different types of clock buffers are available in the XC4000X:

- Global Low-Skew Buffers (BUFGLS)
- Global Early Buffers (BUFGE)

Global Low-Skew Buffers are the standard clock buffers. They should be used for most internal clocking, whenever a large portion of the device must be driven.

Global Early Buffers are designed to provide a faster clock access, but CLB access is limited to one-fourth of the device. They also facilitate a faster I/O interface.

Figure 35 is a conceptual diagram of the global net structure in the XC4000X.

Global Early buffers and Global Low-Skew buffers share a single pad. Therefore, the same IPAD symbol can drive one buffer of each type, in parallel. This configuration is particularly useful when using the Fast Capture latches, as described in "IOB Input Signals" on page 20. Paired Global Early and Global Low-Skew buffers share a common input; they cannot be driven by two different signals.

Choosing an XC4000X Clock Buffer

The clocking structure of the XC4000X provides a large variety of features. However, it can be simple to use, without understanding all the details. The software automatically handles clocks, along with all other routing, when the appropriate clock buffer is placed in the design. In fact, if a buffer symbol called BUFG is placed, rather than a specific type of buffer, the software even chooses the buffer most appropriate for the design. The detailed information in this section is provided for those users who want a finer level of control over their designs.

If fine control is desired, use the following summary and Table 15 on page 35 to choose an appropriate clock buffer.

- The simplest thing to do is to use a Global Low-Skew buffer.
- If a faster clock path is needed, try a BUFG. The software will first try to use a Global Low-Skew Buffer. If timing requirements are not met, a faster buffer will automatically be used.
- If a single quadrant of the chip is sufficient for the clocked logic, and the timing requires a faster clock than the Global Low-Skew buffer, use a Global Early buffer.

Global Low-Skew Buffers

Each corner of the XC4000X device has two Global Low-Skew buffers. Any of the eight Global Low-Skew buffers can drive any of the eight vertical Global lines in a column of CLBs. In addition, any of the buffers can drive any of the four vertical lines accessing the IOBs on the left edge of the device, and any of the eight vertical lines accessing the IOBs on the right edge of the device. (See Figure 36 on page 38.)

IOBs at the top and bottom edges of the device are accessed through the vertical Global lines in the CLB array, as in the XC4000E. Any Global Low-Skew buffer can, therefore, access every IOB and CLB in the device.

The Global Low-Skew buffers can be driven by either semi-dedicated pads or internal logic.

To use a Global Low-Skew buffer, instantiate a BUFGLS element in a schematic or in HDL code. If desired, attach a LOC attribute or property to direct placement to the designated location. For example, attach a LOC=T attribute or property to direct that a BUFGLS be placed in one of the two Global Low-Skew buffers on the top edge of the device, or a LOC=TR to indicate the Global Low-Skew buffer on the top edge of the device, on the right.

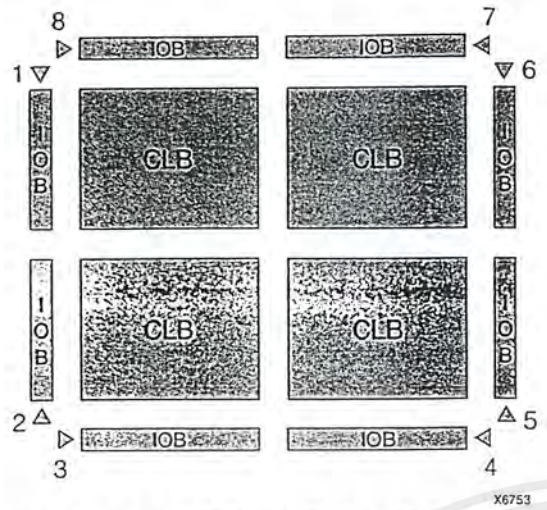


Figure 36: Any BUFGLS (GCK1 - GCK8) Can Drive Any or All Clock Inputs on the Device

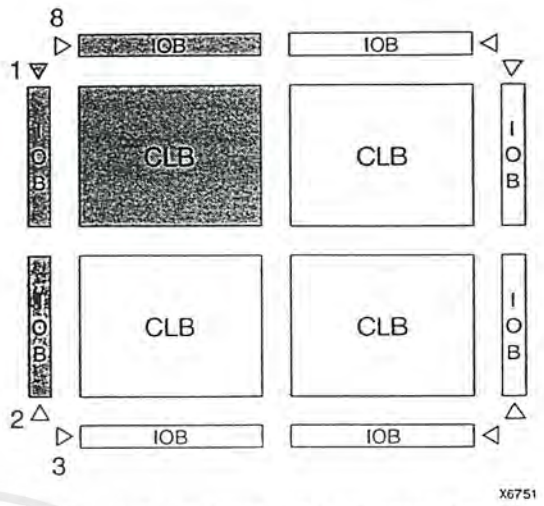


Figure 37: Left and Right BUFGEs Can Drive Any or All Clock Inputs in Same Quadrant or Edge (GCK1 is shown. GCK2, GCK5 and GCK6 are similar.)

Global Early Buffers

Each corner of the XC4000X device has two Global Early buffers. The primary purpose of the Global Early buffers is to provide an earlier clock access than the potentially heavily-loaded Global Low-Skew buffers. A clock source applied to both buffers will result in the Global Early clock edge occurring several nanoseconds earlier than the Global Low-Skew buffer clock edge, due to the lighter loading.

Global Early buffers also facilitate the fast capture of device inputs, using the Fast Capture latches described in "IOB Input Signals" on page 20. For Fast Capture, take a single clock signal, and route it through both a Global Early buffer and a Global Low-Skew buffer. (The two buffers share an input pad.) Use the Global Early buffer to clock the Fast Capture latch, and the Global Low-Skew buffer to clock the normal input flip-flop or latch, as shown in Figure 17 on page 23.

The Global Early buffers can also be used to provide a fast Clock-to-Out on device output pins. However, an early clock in the output flip-flop IOB must be taken into consideration when calculating the internal clock speed for the design.

The Global Early buffers at the left and right edges of the chip have slightly different capabilities than the ones at the top and bottom. Refer to Figure 37, Figure 38, and Figure 35 on page 36 while reading the following explanation.

Each Global Early buffer can access the eight vertical Global lines for all CLBs in the quadrant. Therefore, only one-fourth of the CLB clock pins can be accessed. This restriction is in large part responsible for the faster speed of the buffers, relative to the Global Low-Skew buffers.

The left-side Global Early buffers can each drive two of the four vertical lines accessing the IOBs on the entire left edge of the device. The right-side Global Early buffers can each drive two of the eight vertical lines accessing the IOBs on the entire right edge of the device. (See Figure 37.)

Each left and right Global Early buffer can also drive half of the IOBs along either the top or bottom edge of the device, using a dedicated line that can only be accessed through the Global Early buffers.

The top and bottom Global Early buffers can drive half of the IOBs along either the left or right edge of the device, as shown in Figure 38. They can only access the top and bottom IOBs via the CLB global lines.

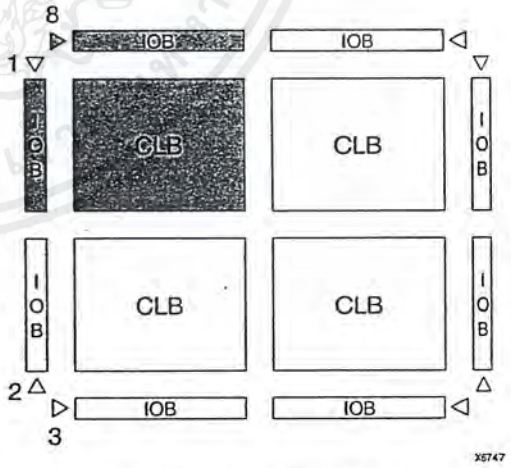


Figure 38: Top and Bottom BUFGEs Can Drive Any or All Clock Inputs in Same Quadrant (GCK8 is shown. GCK3, GCK4 and GCK7 are similar.)

The top and bottom Global Early buffers are about 1 ns slower clock to out than the left and right Global Early buffers.

The Global Early buffers can be driven by either semi-dedicated pads or internal logic. They share pads with the Global Low-Skew buffers, so a single net can drive both global buffers, as described above.

To use a Global Early buffer, place a BUFGE element in a schematic or in HDL code. If desired, attach a LOC attribute or property to direct placement to the designated location. For example, attach a LOC=T attribute or property to direct that a BUFGE be placed in one of the two Global Early buffers on the top edge of the device, or a LOC=TR to indicate the Global Early buffer on the top edge of the device, on the right.

Power Distribution

Power for the FPGA is distributed through a grid to achieve high noise immunity and isolation between logic and I/O. Inside the FPGA, a dedicated Vcc and Ground ring surrounding the logic array provides power to the I/O drivers, as shown in Figure 39. An independent matrix of Vcc and Ground lines supplies the interior logic of the device.

This power distribution grid provides a stable supply and ground for all internal logic, providing the external package power pins are all connected and appropriately de-coupled. Typically, a 0.1 µF capacitor connected between each Vcc pin and the board's Ground plane will provide adequate de-coupling.

Output buffers capable of driving/sinking the specified 12 mA loads under specified worst-case conditions may be capable of driving/sinking up to 10 times as much current under best case conditions.

Noise can be reduced by minimizing external load capacitance and reducing simultaneous output transitions in the same direction. It may also be beneficial to locate heavily loaded output buffers near the Ground pads. The I/O Block output buffers have a slew-rate limited mode (default) which should be used where output rise and fall times are not speed-critical.

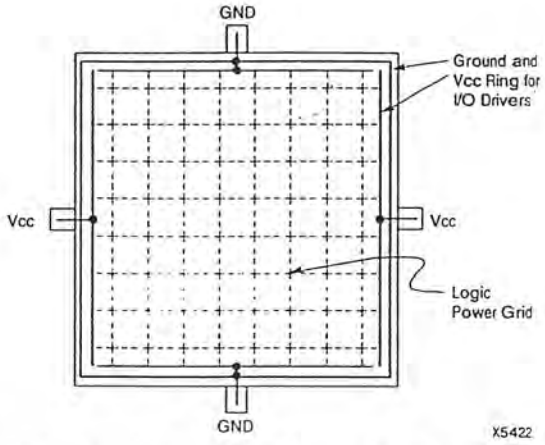


Figure 39: XC4000 Series Power Distribution

Pin Descriptions

There are three types of pins in the XC4000 Series devices:

- Permanently dedicated pins
- User I/O pins that can have special functions
- Unrestricted user-programmable I/O pins.

Before and during configuration, all outputs not used for the configuration process are 3-stated with a 50 kΩ - 100 kΩ pull-up resistor.

After configuration, if an IOB is unused it is configured as an input with a 50 kΩ - 100 kΩ pull-up resistor.

XC4000 Series devices have no dedicated Reset input. Any user I/O can be configured to drive the Global Set/Reset net, GSR. See "Global Set/Reset" on page 11 for more information on GSR.

XC4000 Series devices have no Powerdown control input, as the XC3000 and XC2000 families do. The XC3000/XC2000 Powerdown control also 3-stated all of the device
I/O pins. For XC4000 Series devices, use the global 3-state net, GTS, instead. This net 3-states all outputs, but does not place the device in low-power mode. See "IOB Output Signals" on page 23 for more information on GTS.

Device pins for XC4000 Series devices are described in Table 16. Pin functions during configuration for each of the seven configuration modes are summarized in Table 22 on page 58, in the "Configuration Timing" section.

Figure 41 on page 44 is a diagram of the XC4000 Series boundary scan logic. It includes three bits of Data Register per IOB, the IEEE 1149.1 Test Access Port controller, and the Instruction Register with decodes.

XC4000 Series devices can also be configured through the boundary scan logic. See "Readback" on page 55.

Data Registers

The primary data register is the boundary scan register. For each IOB pin in the FPGA, bonded or not, it includes three bits for In, Out and 3-State Control. Non-IOB pins have appropriate partial bit population for In or Out only. $\overline{\text{PROGRAM}}$, CCLK and DONE are not included in the boundary scan register. Each EXTEST CAPTURE-DR state captures all In, Out, and 3-state pins.

The data register also includes the following non-pin bits: TDO.T, and TDO.O, which are always bits 0 and 1 of the data register, respectively, and BSCANT.UPD, which is always the last bit of the data register. These three boundary scan bits are special-purpose Xilinx test signals.

The other standard data register is the single flip-flop BYPASS register. It synchronizes data being passed through the FPGA to the next downstream boundary scan device.

The FPGA provides two additional data registers that can be specified using the BSCAN macro. The FPGA provides two user pins (BSCAN.SEL1 and BSCAN.SEL2) which are the decodes of two user instructions. For these instructions, two corresponding pins (BSCAN.TDO1 and BSCAN.TDO2) allow user scan data to be shifted out on TDO. The data register clock (BSCAN.DRCK) is available for control of test logic which the user may wish to implement with CLBs. The NAND of TCK and RUN-TEST-IDLE is also provided (BSCAN.IDLE).

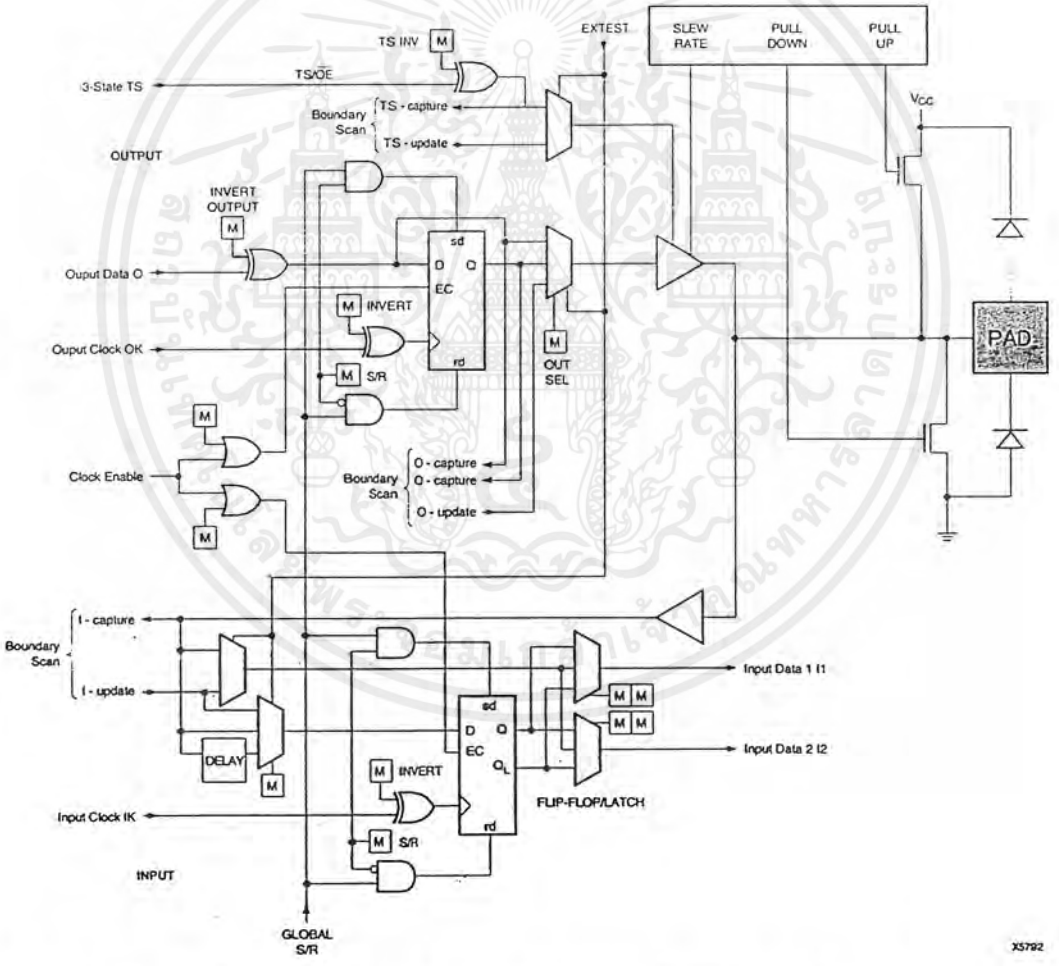


Figure 40: Block Diagram of XC4000E IOB with Boundary Scan (some details not shown). XC4000X Boundary Scan Logic is Identical.

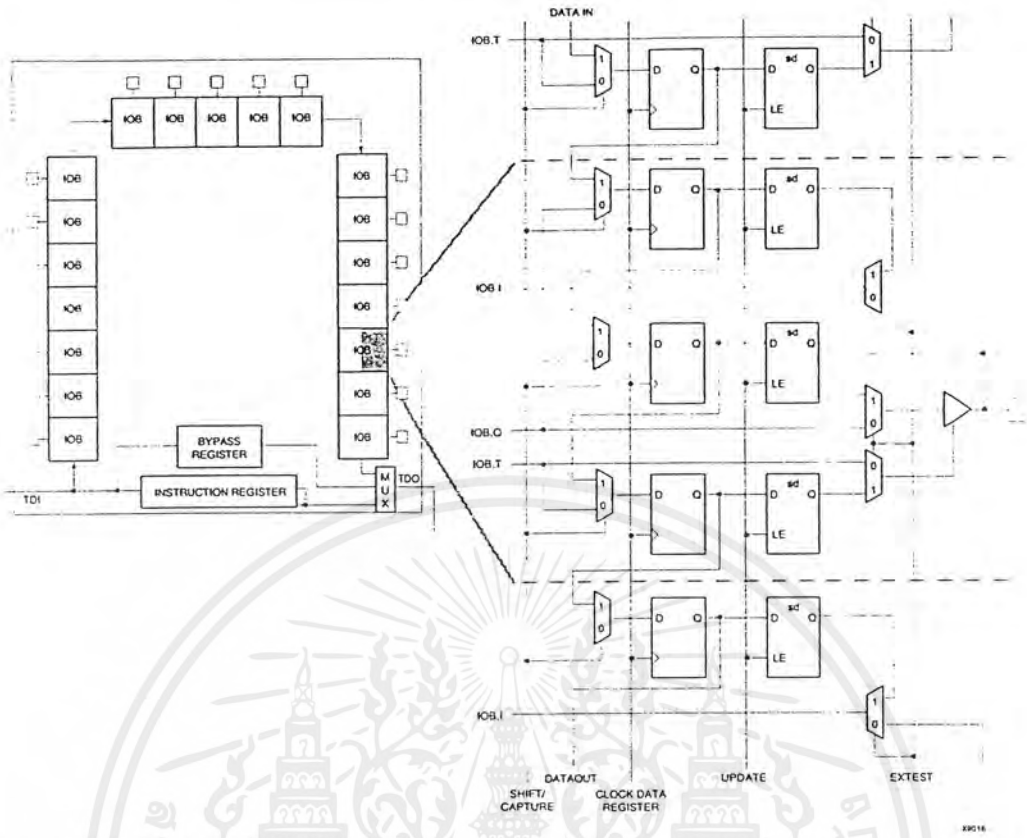


Figure 41: XC4000 Series Boundary Scan Logic

Instruction Set

The XC4000 Series boundary scan instruction set also includes instructions to configure the device and read back the configuration data. The instruction set is coded as shown in Table 17.

Bit Sequence

The bit sequence within each IOB is: In, Out, 3-State. The input-only M0 and M2 mode pins contribute only the In bit to the boundary scan I/O data register, while the output-only M1 pin contributes all three bits.

The first two bits in the I/O data register are TDO.T and TDO.O, which can be used for the capture of internal signals. The final bit is BSCANT.UPD, which can be used to drive an internal net. These locations are primarily used by Xilinx for internal testing.

From a cavity-up view of the chip (as shown in XDE or Epic), starting in the upper right chip corner, the boundary scan data-register bits are ordered as shown in Figure 42. The device-specific pinout tables for the XC4000 Series include the boundary scan locations for each IOB pin.

BSDL (Boundary Scan Description Language) files for XC4000 Series devices are available on the Xilinx FTP site.

Including Boundary Scan in a Schematic

If boundary scan is only to be used during configuration, no special schematic elements need be included in the schematic or HDL code. In this case, the special boundary scan pins TDI, TMS, TCK and TDO can be used for user functions after configuration.

To indicate that boundary scan remain enabled after configuration, place the BSCAN library symbol and connect the TDI, TMS, TCK and TDO pad symbols to the appropriate pins, as shown in Figure 43.

Even if the boundary scan symbol is used in a schematic, the input pins TMS, TCK, and TDI can still be used as inputs to be routed to internal logic. Care must be taken not to force the chip into an undesired boundary scan state by inadvertently applying boundary scan input patterns to these pins. The simplest way to prevent this is to keep TMS High, and then apply whatever signal is desired to TDI and TCK.

Table 17: Boundary Scan Instructions

Instruction I2	I1	I0	Test Selected	TDO Source	I/O Data Source
0	0	0	EXTEST	DR	DR
0	0	1	SAMPLE/PRELOAD	DR	Pin/Logic
0	1	0	USER 1	BSCAN.TDO1	User Logic
0	1	1	USER 2	BSCAN.TDO2	User Logic
1	0	0	READBACK	Readback Data	Pin/Logic
1	0	1	CONFIGURE	DOUT	Disabled
1	1	0	Reserved	—	—
1	1	1	BYPASS	Bypass Register	—

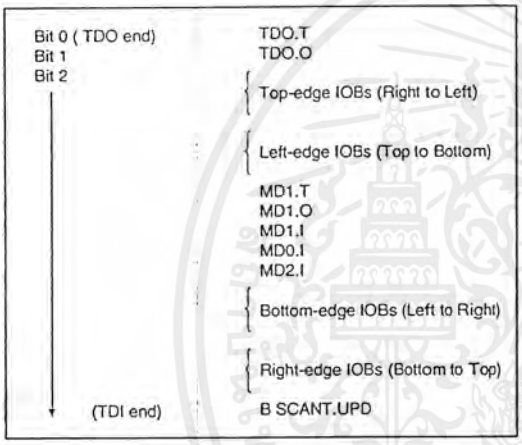


Figure 42: Boundary Scan Bit Sequence

Avoiding Inadvertent Boundary Scan

If TMS or TCK is used as user I/O, care must be taken to ensure that at least one of these pins is held constant during configuration. In some applications, a situation may occur where TMS or TCK is driven during configuration. This may cause the device to go into boundary scan mode and disrupt the configuration process.

To prevent activation of boundary scan during configuration, do either of the following:

- TMS: Tie High to put the Test Access Port controller in a benign RESET state
- TCK: Tie High or Low—don't toggle this clock input.

For more information regarding boundary scan, refer to the Xilinx Application Note XAPP 017.001, "*Boundary Scan in XC4000E Devices*."

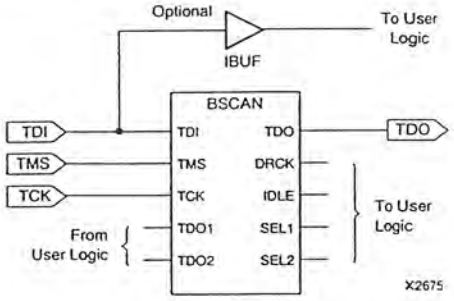


Figure 43: Boundary Scan Schematic Example

Configuration

Configuration is the process of loading design-specific programming data into one or more FPGAs to define the functional operation of the internal blocks and their interconnections. This is somewhat like loading the command registers of a programmable peripheral chip. XC4000 Series devices use several hundred bits of configuration data per CLB and its associated interconnects. Each configuration bit defines the state of a static memory cell that controls either a function look-up table bit, a multiplexer input, or an interconnect pass transistor. The XACT*step* development system translates the design into a netlist file. It automatically partitions, places and routes the logic and generates the configuration data in PROM format.

Special Purpose Pins

Three configuration mode pins (M2, M1, M0) are sampled prior to configuration to determine the configuration mode. After configuration, these pins can be used as auxiliary connections. M2 and M0 can be used as inputs, and M1 can be used as an output. The XACT*step* development system does not use these resources unless they are explicitly specified in the design entry. This is done by placing a special pad symbol called MD2, MD1, or MD0 instead of the input or output pad symbol.

In XC4000 Series devices, the mode pins have weak pull-up resistors during configuration. With all three mode pins High, Slave Serial mode is selected, which is the most popular configuration mode. Therefore, for the most common configuration mode, the mode pins can be left unconnected. (Note, however, that the internal pull-up resistor value can be as high as 100 kΩ) After configuration, these pins can individually have weak pull-up or pull-down resistors, as specified in the design. A pull-down resistor value of 4.7 kΩ is recommended.

These pins are located in the lower left chip corner and are near the readback nets. This location allows convenient routing if compatibility with the XC2000 and XC3000 family conventions of M0/RT, M1/RD is desired.

Configuration Modes

XC4000E devices have six configuration modes. XC4000X devices have the same six modes, plus an additional configuration mode. These modes are selected by a 3-bit input code applied to the M2, M1, and M0 inputs. There are three self-loading Master modes, two Peripheral modes, and a Serial Slave mode, which is used primarily for daisy-chained devices. The coding for mode selection is shown in Table 18.

Table 18: Configuration Modes

Mode	M2	M1	M0	CCLK	Data
Master Serial	0	0	0	output	Bit-Serial
Slave Serial	1	1	1	input	Bit-Serial
Master Parallel Up	1	0	0	output	Byte-Wide, increment from 00000
Master Parallel Down	1	1	0	output	Byte-Wide, decrement from 3FFFF
Peripheral Synchronous*	0	1	1	input	Byte-Wide
Peripheral Asynchronous	1	0	1	output	Byte-Wide
Reserved	0	1	0	—	—
Reserved	0	0	1	—	—

* Can be considered byte-wide Slave Parallel

A detailed description of each configuration mode, with timing information, is included later in this data sheet. During configuration, some of the I/O pins are used temporarily for the configuration process. All pins used during configuration are shown in Table 22 on page 58.

Master Modes

The three Master modes use an internal oscillator to generate a Configuration Clock (CCLK) for driving potential slave devices. They also generate address and timing for external PROM(s) containing the configuration data.

Master Parallel (Up or Down) modes generate the CCLK signal and PROM addresses and receive byte parallel data. The data is internally serialized into the FPGA data-frame format. The up and down selection generates starting addresses at either zero or 3FFFF (3FFFFF when 22 address lines are used), for compatibility with different microprocessor addressing conventions. The Master Serial mode generates CCLK and receives the configuration data in serial form from a Xilinx serial-configuration PROM.

CCLK speed is selectable as either 1 MHz (default) or 8 MHz. Configuration always starts at the default slow frequency, then can switch to the higher frequency during the first frame. Frequency tolerance is -50% to +25%.

Additional Address lines in XC4000 devices

The XC4000X devices have additional address lines (A18-A21) allowing the additional address space required to daisy-chain several large devices.

The extra address lines are programmable in XC4000EX devices. By default these address lines are not activated. In the default mode, the devices are compatible with existing XC4000 and XC4000E products. If desired, the extra address lines can be used by specifying the address lines option in bitgen as 22 (bitgen -g AddressLines:22). The lines (A18-A21) are driven when a master device detects, via the bitstream, that it should be using all 22 address lines. Because these pins will initially be pulled high by internal pull-ups, designers using Master Parallel Up mode should use external pull down resistors on pins A18-A21. If Master Parallel Down mode is used external resistors are not necessary.

All 22 address lines are always active in Master Parallel modes with XC4000XL devices. The additional address lines behave identically to the lower order address lines. If the Address Lines option in bitgen is set to 18, it will be ignored by the XC4000XL device.

The additional address lines (A18-A21) are not available in the PC84 package.

Peripheral Modes

The two Peripheral modes accept byte-wide data from a bus. A RDY/$\overline{\text{BUSY}}$ status is available as a handshake signal. In Asynchronous Peripheral mode, the internal oscillator generates a CCLK burst signal that serializes the byte-wide data. CCLK can also drive slave devices. In the synchronous mode, an externally supplied clock input to CCLK serializes the data.

Slave Serial Mode

In Slave Serial mode, the FPGA receives serial configuration data on the rising edge of CCLK and, after loading its configuration, passes additional data out, resynchronized on the next falling edge of CCLK.

Multiple slave devices with identical configurations can be wired with parallel DIN inputs. In this way, multiple devices can be configured simultaneously.

Serial Daisy Chain

Multiple devices with different configurations can be connected together in a "daisy chain," and a single combined bitstream used to configure the chain of slave devices.

To configure a daisy chain of devices, wire the CCLK pins of all devices in parallel, as shown in Figure 51 on page 60. Connect the DOUT of each device to the DIN of the next. The lead or master FPGA and following slaves each passes resynchronized configuration data coming from a single source. The header data, including the length count,

is passed through and is captured by each FPGA when it recognizes the 0010 preamble. Following the length-count data, each FPGA outputs a High on DOUT until it has received its required number of data frames.

After an FPGA has received its configuration data, it passes on any additional frame start bits and configuration data on DOUT. When the total number of configuration clocks applied after memory initialization equals the value of the 24-bit length count, the FPGAs begin the start-up sequence and become operational together. FPGA I/O are normally released two CCLK cycles after the last configuration bit is received. Figure 47 on page 53 shows the start-up timing for an XC4000 Series device.

The daisy-chained bitstream is not simply a concatenation of the individual bitstreams. The PROM file formatter must be used to combine the bitstreams for a daisy-chained configuration.

Multi-Family Daisy Chain

All Xilinx FPGAs of the XC2000, XC3000, and XC4000 Series use a compatible bitstream format and can, therefore, be connected in a daisy chain in an arbitrary sequence. There is, however, one limitation. The lead device must belong to the highest family in the chain. If the chain contains XC4000 Series devices, the master normally cannot be an XC2000 or XC3000 device.

The reason for this rule is shown in Figure 47 on page 53. Since all devices in the chain store the same length count value and generate or receive one common sequence of CCLK pulses, they all recognize length-count match on the same CCLK edge, as indicated on the left edge of Figure 47. The master device then generates additional CCLK pulses until it reaches its finish point F. The different families generate or require different numbers of additional CCLK pulses until they reach F. Not reaching F means that the device does not really finish its configuration, although DONE may have gone High, the outputs became active, and the internal reset was released. For the XC4000 Series device, not reaching F means that readback cannot be initiated and most boundary scan instructions cannot be used.

The user has some control over the relative timing of these events and can, therefore, make sure that they occur at the proper time and the finish point F is reached. Timing is controlled using options in the bitstream generation software.

XC3000 Master with an XC4000 Series Slave

Some designers want to use an inexpensive lead device in peripheral mode and have the more precious I/O pins of the XC4000 Series devices all available for user I/O. Figure 44 provides a solution for that case.

This solution requires one CLB, one IOB and pin, and an internal oscillator with a frequency of up to 5 MHz as a clock source. The XC3000 master device must be configured with late Internal Reset, which is the default option.

One CLB and one IOB in the lead XC3000-family device are used to generate the additional CCLK pulse required by the XC4000 Series devices. When the lead device removes the internal RESET signal, the 2-bit shift register responds to its clock input and generates an active Low output signal for the duration of the subsequent clock period. An external connection between this output and CCLK thus creates the extra CCLK pulse.

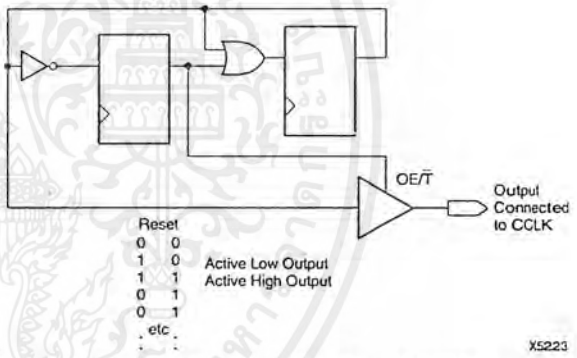


Figure 44: CCLK Generation for XC3000 Master Driving an XC4000 Series Slave

Setting CCLK Frequency

For Master modes, CCLK can be generated in either of two frequencies. In the default slow mode, the frequency ranges from 0.5 MHz to 1.25 MHz for XC4000E and XC4000EX devices and from 0.6 MHz to 1.8 MHz for XC4000XL devices. In fast CCLK mode, the frequency ranges from 4 MHz to 10 MHz for XC4000EX devices and from 5 MHz to 15 MHz for XC4000XL devices. The frequency is selected by an option when running the bitstream generation software. If an XC4000 Series Master is driving an XC3000- or XC2000-family slave, slow CCLK mode must be used. In addition, an XC4000XL device driving a XC4000E or XC4000EX should use slow mode. Slow mode is the default.

Table 19: XC4000 Series Data Stream Formats

Data Type	All Other Modes (D0...)
Fill Byte	11111111b
Preamble Code	0010b
Length Count	COUNT(23:0)
Fill Bits	1111b
Start Field	0b
Data Frame	DATA(n-1:0)
CRC or Constant Field Check	xxxx (CRC) or 0110b
Extend Write Cycle	—
Postamble	01111111b
Start-Up Bytes	xxh

Legend:

Not shaded	Once per bitstream
Light	Once per data frame
Dark	Once per device

Data Stream Format

The data stream ("bitstream") format is identical for all configuration modes.

The data stream formats are shown in Table 19. Bit-serial data is read from left to right, and byte-parallel data is effectively assembled from this serial bitstream, with the first bit in each byte assigned to D0.

The configuration data stream begins with a string of eight ones, a preamble code, followed by a 24-bit length count and a separator field of ones. This header is followed by the actual configuration data in frames. The length and number of frames depends on the device type (see Table 20 and Table 21). Each frame begins with a start field and ends with an error check. A postamble code is required to signal the end of data for a single device. In all cases, additional start-up bytes of data are required to provide four clocks for the startup sequence at the end of configuration. Long daisy chains require additional startup bytes to shift the last data through the chain. All startup bytes are don't-cares; these bytes are not included in bitstreams created by the Xilinx software.

A selection of CRC or non-CRC error checking is allowed by the bitstream generation software. The non-CRC error checking tests for a designated end-of-frame field for each frame. For CRC error checking, the software calculates a running CRC and inserts a unique four-bit partial check at the end of each frame. The 11-bit CRC check of the last frame of an FPGA includes the last seven data bits.

Detection of an error results in the suspension of data loading and the pulling down of the $\overline{\text{INIT}}$ pin. In Master modes, CCLK and address signals continue to operate externally. The user must detect $\overline{\text{INIT}}$ and initialize a new configuration by pulsing the $\overline{\text{PROGRAM}}$ pin Low or cycling Vcc.

Table 20: XC4000E Program Data

Device	XC4003E	XC4005E	XC4006E	XC4008E	XC4010E	XC4013E	XC4020E	XC4025E
Max Logic Gates	3,000	5,000	6,000	8,000	10,000	13,000	20,000	25,000
CLBs (Row x Col.)	100 (10 x 10)	196 (14 x 14)	256 (16 x 16)	324 (18 x 18)	400 (20 x 20)	576 (24 x 24)	784 (28 x 28)	1,024 (32 x 32)
IOBs	80	112	128	144	160	192	224	256
Flip-Flops	360	616	768	936	1,120	1,536	2,016	2,560
Bits per Frame	126	166	186	206	226	266	306	346
Frames	428	572	644	716	788	932	1,076	1,220
Program Data	53,936	94,960	119,792	147,504	178,096	247,920	329,264	422,128
PROM Size (bits)	53,984	95,008	119,840	147,552	178,144	247,968	329,312	422,176

Notes: 1. Bits per Frame = (10 x number of rows) + 7 for the top + 13 for the bottom + 1 + 1 start bit + 4 error check bits
Number of Frames = (36 x number of columns) + 26 for the left edge + 41 for the right edge + 1
Program Data = (Bits per Frame x Number of Frames) + 8 postamble bits
PROM Size = Program Data + 40 (header) + 8
2. The user can add more "one" bits as leading dummy bits in the header, or, if CRC = off, as trailing dummy bits at the end of any frame, following the four error check bits. However, the Length Count value **must** be adjusted for all such extra "one" bits, even for extra leading ones at the beginning of the header.

Table 21: XC4000EX/XL Program Data

Device	XC4002XL	XC4005	XC4010	XC4013	XC4020	XC4028	XC4036	XC4044	XC4052	XC4062	XC4085
Max Logic Gates	2,000	5,000	10,000	13,000	20,000	28,000	36,000	44,000	52,000	62,000	85,000
CLBs (Row x Column)	64 (8 x 8)	196 (14 x 14)	400 (20 x 20)	576 (24 x 24)	784 (28 x 28)	1,024 (32 x 32)	1,296 (36 x 36)	1,600 (40 x 40)	1,936 (44 x 44)	2,304 (48 x 48)	3,136 (56 x 56)
IOBs	64	112	160	192	224	256	288	320	352	384	448
Flip-Flops	256	616	1,120	1,536	2,016	2,560	3,168	3,840	4,576	5,376	7,168
Bits per Frame	133	205	277	325	373	421	469	517	565	613	709
Frames	459	741	1,023	1,211	1,399	1,587	1,775	1,963	2,151	2,339	2,715
Program Data	61,052	151,910	283,376	393,580	521,832	668,124	832,480	1,014,876	1,215,320	1,433,804	1,924,940
PROM Size (bits)	61,104	151,960	283,424	393,632	521,880	668,172	832,528	1,014,924	1,215,368	1,433,852	1,924,992

Notes: 1. Bits per frame = (13 x number of rows) + 9 for the top + 17 for the bottom + 8 + 1 start bit + 4 error check bits.
Frames = (47 x number of columns) + 27 for the left edge + 52 for the right edge + 4.
Program data = (bits per frame x number of frames) + 5 postamble bits.
PROM size = (program data + 40 header bits + 8 start bits) rounded up to the nearest byte.
2. The user can add more "one" bits as leading dummy bits in the header, or, if CRC = off, as trailing dummy bits at the end of any frame, following the four error check bits. However, the Length Count value must be adjusted for all such extra "one" bits, even for extra leading "ones" at the beginning of the header.

Cyclic Redundancy Check (CRC) for Configuration and Readback

The Cyclic Redundancy Check is a method of error detection in data transmission applications. Generally, the transmitting system performs a calculation on the serial bitstream. The result of this calculation is tagged onto the data stream as additional check bits. The receiving system performs an identical calculation on the bitstream and compares the result with the received checksum.

Each data frame of the configuration bitstream has four error bits at the end, as shown in Table 19. If a frame data error is detected during the loading of the FPGA, the configuration process with a potentially corrupted bitstream is terminated. The FPGA pulls the $\overline{INIT}$ pin Low and goes into a Wait state.

During Readback, 11 bits of the 16-bit checksum are added to the end of the Readback data stream. The checksum is computed using the CRC-16 CCITT polynomial, as shown in Figure 45. The checksum consists of the 11 most significant bits of the 16-bit code. A change in the checksum indicates a change in the Readback bitstream. A comparison to a previous checksum is meaningful only if the readback data is independent of the current device state. CLB outputs should not be included (Read Capture option not

used), and if RAM is present, the RAM content must be unchanged.

Statistically, one error out of 2048 might go undetected.

Configuration Sequence

There are four major steps in the XC4000 Series power-up configuration sequence.

- Configuration Memory Clear
- Initialization
- Configuration
- Start-Up

The full process is illustrated in Figure 46.

Configuration Memory Clear

When power is first applied or is reapplied to an FPGA, an internal circuit forces initialization of the configuration logic. When Vcc reaches an operational level, and the circuit passes the write and read test of a sample pair of configuration bits, a time delay is started. This time delay is nominally 16 ms, and up to 10% longer in the low-voltage devices. The delay is four times as long when in Master Modes (M0 Low), to allow ample time for all slaves to reach a stable Vcc. When all $\overline{\text{INIT}}$ pins are tied together, as recommended, the longest delay takes precedence. Therefore, devices with different time delays can easily be mixed and matched in a daisy chain.

This delay is applied only on power-up. It is not applied when re-configuring an FPGA by pulsing the $\overline{\text{PROGRAM}}$ pin

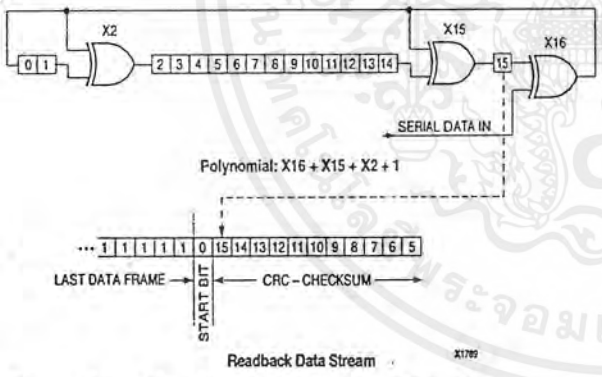


Figure 45: Circuit for Generating CRC-16

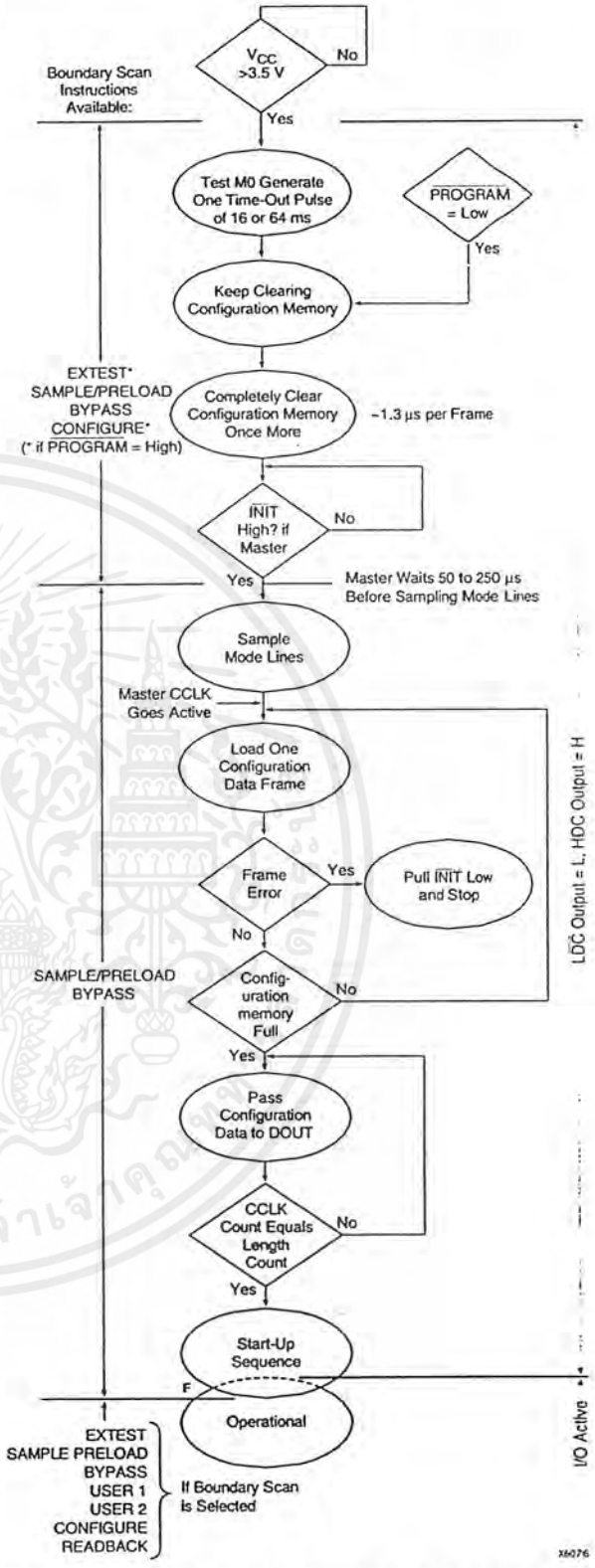


Figure 46: Power-up Configuration Sequence

Low. During this time delay, or as long as the $\overline{\text{PROGRAM}}$ input is asserted, the configuration logic is held in a Configuration Memory Clear state. The configuration-memory frames are consecutively initialized, using the internal oscillator.

At the end of each complete pass through the frame addressing, the power-on time-out delay circuitry and the level of the $\overline{\text{PROGRAM}}$ pin are tested. If neither is asserted, the logic initiates one additional clearing of the configuration frames and then tests the $\overline{\text{INIT}}$ input.

Initialization

During initialization and configuration, user pins HDC, $\overline{\text{LDC}}$, $\overline{\text{INIT}}$ and DONE provide status outputs for the system interface. The outputs $\overline{\text{LDC}}$, $\overline{\text{INIT}}$ and DONE are held Low and HDC is held High starting at the initial application of power.

The open drain $\overline{\text{INIT}}$ pin is released after the final initialization pass through the frame addresses. There is a deliberate delay of 50 to 250 µs (up to 10% longer for low-voltage devices) before a Master-mode device recognizes an inactive $\overline{\text{INIT}}$. Two internal clocks after the $\overline{\text{INIT}}$ pin is recognized as High, the FPGA samples the three mode lines to determine the configuration mode. The appropriate interface lines become active and the configuration preamble and data can be loaded.Configuration

The 0010 preamble code indicates that the following 24 bits represent the length count. The length count is the total number of configuration clocks needed to load the complete configuration data. (Four additional configuration clocks are required to complete the configuration process, as discussed below.) After the preamble and the length count have been passed through to all devices in the daisy chain, DOUT is held High to prevent frame start bits from reaching any daisy-chained devices.

A specific configuration bit, early in the first frame of a master device, controls the configuration-clock rate and can increase it by a factor of eight. Therefore, if a fast configuration clock is selected by the bitstream, the slower clock rate is used until this configuration bit is detected.

Each frame has a start field followed by the frame-configuration data bits and a frame error field. If a frame data error is detected, the FPGA halts loading, and signals the error by pulling the open-drain $\overline{\text{INIT}}$ pin Low. After all configuration frames have been loaded into an FPGA, DOUT again follows the input data so that the remaining data is passed on to the next device.

Delaying Configuration After Power-Up

There are two methods of delaying configuration after power-up: put a logic Low on the $\overline{\text{PROGRAM}}$ input, or pull the bidirectional $\overline{\text{INIT}}$ pin Low, using an open-collector (open-drain) driver. (See Figure 46 on page 50.)

A Low on the $\overline{\text{PROGRAM}}$ input is the more radical approach, and is recommended when the power-supply rise time is excessive or poorly defined. As long as $\overline{\text{PROGRAM}}$ is Low, the FPGA keeps clearing its configuration memory. When $\overline{\text{PROGRAM}}$ goes High, the configuration memory is cleared one more time, followed by the beginning of configuration, provided the $\overline{\text{INIT}}$ input is not externally held Low. Note that a Low on the $\overline{\text{PROGRAM}}$ input automatically forces a Low on the $\overline{\text{INIT}}$ output. The XC4000 Series $\overline{\text{PROGRAM}}$ pin has a permanent weak pull-up.

Using an open-collector or open-drain driver to hold $\overline{\text{INIT}}$ Low before the beginning of configuration causes the FPGA to wait after completing the configuration memory clear operation. When $\overline{\text{INIT}}$ is no longer held Low externally, the device determines its configuration mode by capturing its mode pins, and is ready to start the configuration process. A master device waits up to an additional 250 µs to make sure that any slaves in the optional daisy chain have seen that $\overline{\text{INIT}}$ is High.

Start-Up

Start-up is the transition from the configuration process to the intended user operation. This transition involves a change from one clock source to another, and a change from interfacing parallel or serial configuration data where most outputs are 3-stated, to normal operation with I/O pins active in the user-system. Start-up must make sure that the user-logic 'wakes up' gracefully, that the outputs become active without causing contention with the configuration signals, and that the internal flip-flops are released from the global Reset or Set at the right time.

Figure 47 describes start-up timing for the three Xilinx families in detail. The configuration modes can use any of the four timing sequences.

To access the internal start-up signals, place the STARTUP library symbol.

Start-up Timing

Different FPGA families have different start-up sequences.

The XC2000 family goes through a fixed sequence. DONE goes High and the internal global Reset is de-activated one CCLK period after the I/O become active.

The XC3000A family offers some flexibility. DONE can be programmed to go High one CCLK period before or after the I/O become active. Independent of DONE, the internal global Reset is de-activated one CCLK period before or after the I/O become active.

The XC4000 Series offers additional flexibility. The three events — DONE going High, the internal Set/Reset being de-activated, and the user I/O going active — can all occur in any arbitrary sequence. Each of them can occur one CCLK period before or after, or simultaneous with, any of the others. This relative timing is selected by means of software options in the bitstream generation software.

The default option, and the most practical one, is for DONE to go High first, disconnecting the configuration data source and avoiding any contention when the I/Os become active one clock later. Reset/Set is then released another clock period later to make sure that user-operation starts from stable internal conditions. This is the most common sequence, shown with heavy lines in Figure 47, but the designer can modify it to meet particular requirements.

Normally, the start-up sequence is controlled by the internal device oscillator output (CCLK), which is asynchronous to the system clock.

XC4000 Series offers another start-up clocking option, UCLK_NOSYNC. The three events described above need not be triggered by CCLK. They can, as a configuration option, be triggered by a user clock. This means that the device can wake up in synchronism with the user system.

When the UCLK_SYNC option is enabled, the user can externally hold the open-drain DONE output Low, and thus stall all further progress in the start-up sequence until DONE is released and has gone High. This option can be used to force synchronization of several FPGAs to a common user clock, or to guarantee that all devices are successfully configured before any I/Os go active.

If either of these two options is selected, and no user clock is specified in the design or attached to the device, the chip could reach a point where the configuration of the device is complete and the Done pin is asserted, but the outputs do not become active. The solution is either to recreate the bitstream specifying the start-up clock as CCLK, or to supply the appropriate user clock.

Start-up Sequence

The Start-up sequence begins when the configuration memory is full, and the total number of configuration clocks received since $\overline{\text{INIT}}$ went High equals the loaded value of the length count.

The next rising clock edge sets a flip-flop Q0, shown in Figure 48. Q0 is the leading bit of a 5-bit shift register. The outputs of this register can be programmed to control three events.

- The release of the open-drain DONE output
- The change of configuration-related pins to the user function, activating all IOBs.
- The termination of the global Set/Reset initialization of all CLB and IOB storage elements.

The DONE pin can also be wire-ANDed with DONE pins of other FPGAs or with other external signals, and can then be used as input to bit Q3 of the start-up register. This is called "Start-up Timing Synchronous to Done In" and is selected by either CCLK_SYNC or UCLK_SYNC.

When DONE is not used as an input, the operation is called "Start-up Timing Not Synchronous to DONE In," and is selected by either CCLK_NOSYNC or UCLK_NOSYNC.

As a configuration option, the start-up control register beyond Q0 can be clocked either by subsequent CCLK pulses or from an on-chip user net called STARTUP.CLK. These signals can be accessed by placing the STARTUP library symbol.

Start-up from CCLK

If CCLK is used to drive the start-up, Q0 through Q3 provide the timing. Heavy lines in Figure 47 show the default timing, which is compatible with XC2000 and XC3000 devices using early DONE and late Reset. The thin lines indicate all other possible timing options.

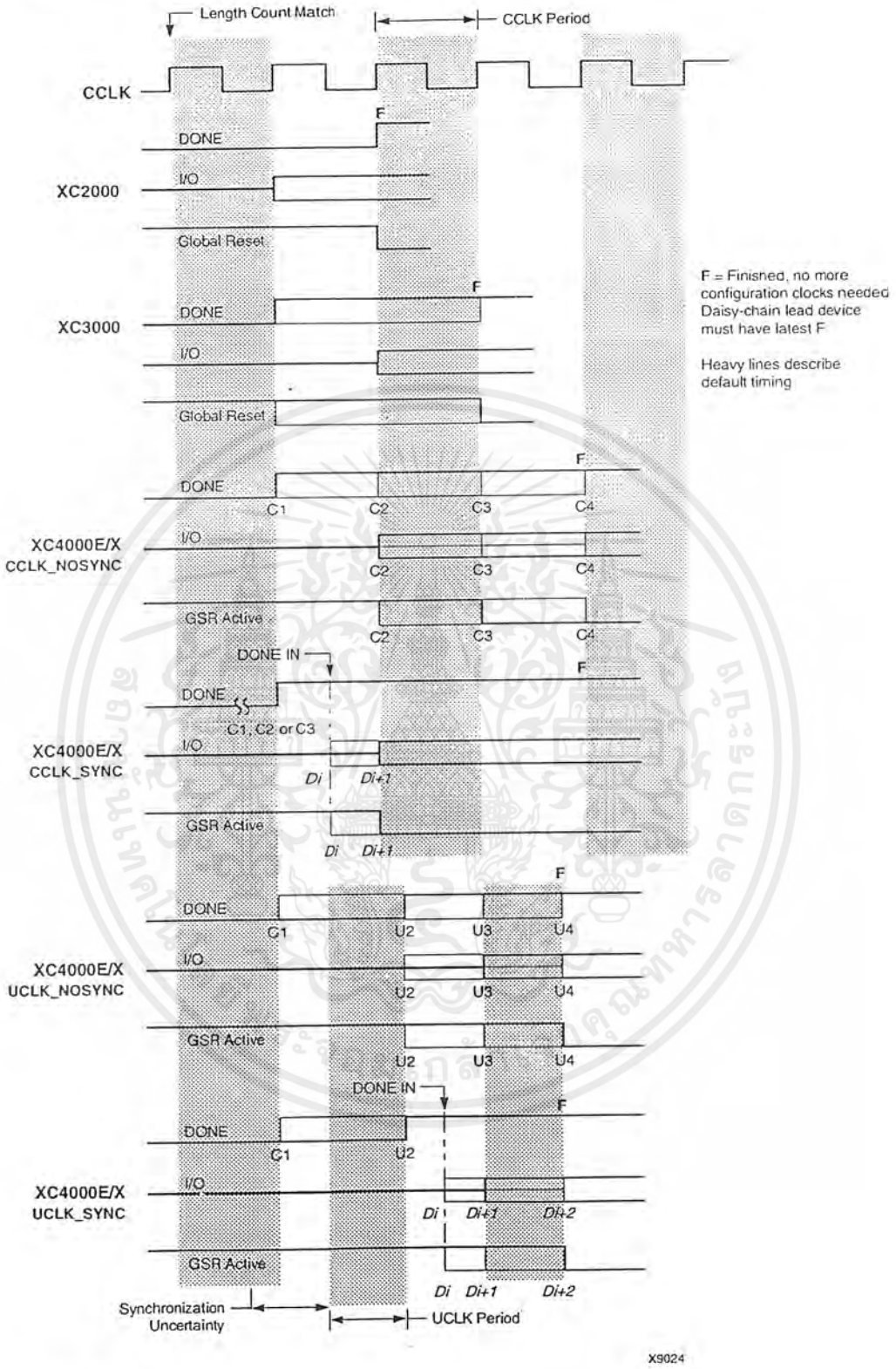


Figure 47: Start-up Timing

Start-up from a User Clock (STARTUP.CLK)

When, instead of CCLK, a user-supplied start-up clock is selected, Q1 is used to bridge the unknown phase relationship between CCLK and the user clock. This arbitration causes an unavoidable one-cycle uncertainty in the timing of the rest of the start-up sequence.

DONE Goes High to Signal End of Configuration

XC4000 Series devices read the expected length count from the bitstream and store it in an internal register. The length count varies according to the number of devices and the composition of the daisy chain. Each device also counts the number of CCLKs during configuration.

Two conditions have to be met in order for the DONE pin to go high:

- the chip's internal memory must be full, and
- the configuration length count must be met, *exactly*.

This is important because the counter that determines when the length count is met begins with the very first CCLK, not the first one after the preamble.

Therefore, if a stray bit is inserted before the preamble, or the data source is not ready at the time of the first CCLK, the internal counter that holds the number of CCLKs will be one ahead of the actual number of data bits read. At the end of configuration, the configuration memory will be full, but the number of bits in the internal counter will not match the expected length count.

As a consequence, a Master mode device will continue to send out CCLKs until the internal counter turns over to zero, and then reaches the correct length count a second time. This will take several seconds [2^{24} * CCLK period] — which is sometimes interpreted as the device not configuring at all.

If it is not possible to have the data ready at the time of the first CCLK, the problem can be avoided by increasing the number in the length count by the appropriate value. The *XACT User Guide* includes detailed information about manually altering the length count.

Note that DONE is an open-drain output and does not go High unless an internal pull-up is activated or an external pull-up is attached. The internal pull-up is activated as the default by the bitstream generation software.

Release of User I/O After DONE Goes High

By default, the user I/O are released one CCLK cycle after the DONE pin goes High. If CCLK is not clocked after DONE goes High, the outputs remain in their initial state — 3-stated, with a 50 kΩ - 100 kΩ pull-up. The delay from DONE High to active user I/O is controlled by an option to the bitstream generation software.

Release of Global Set/Reset After DONE Goes High

By default, Global Set/Reset (GSR) is released two CCLK cycles after the DONE pin goes High. If CCLK is not clocked twice after DONE goes High, all flip-flops are held in their initial set or reset state. The delay from DONE High to GSR inactive is controlled by an option to the bitstream generation software.

Configuration Complete After DONE Goes High

Three full CCLK cycles are required after the DONE pin goes High, as shown in Figure 47 on page 53. If CCLK is not clocked three times after DONE goes High, readback cannot be initiated and most boundary scan instructions cannot be used.

Configuration Through the Boundary Scan Pins

XC4000 Series devices can be configured through the boundary scan pins. The basic procedure is as follows:

- Power up the FPGA with $\overline{\text{INIT}}$ held Low (or drive the $\overline{\text{PROGRAM}}$ pin Low for more than 300 ns followed by a High while holding $\overline{\text{INIT}}$ Low). Holding $\overline{\text{INIT}}$ Low allows enough time to issue the CONFIG command to the FPGA. The pin can be used as I/O after configuration if a resistor is used to hold $\overline{\text{INIT}}$ Low.
- Issue the CONFIG command to the TMS input
- Wait for $\overline{\text{INIT}}$ to go High
- Sequence the boundary scan Test Access Port to the SHIFT-DR state
- Toggle TCK to clock data into TDI pin.

The user must account for all TCK clock cycles after INIT goes High, as all of these cycles affect the Length Count compare.

For more detailed information, refer to the Xilinx application note XAPP017, "*Boundary Scan in XC4000 Devices*." This application note also applies to XC4000E and XC4000X devices.

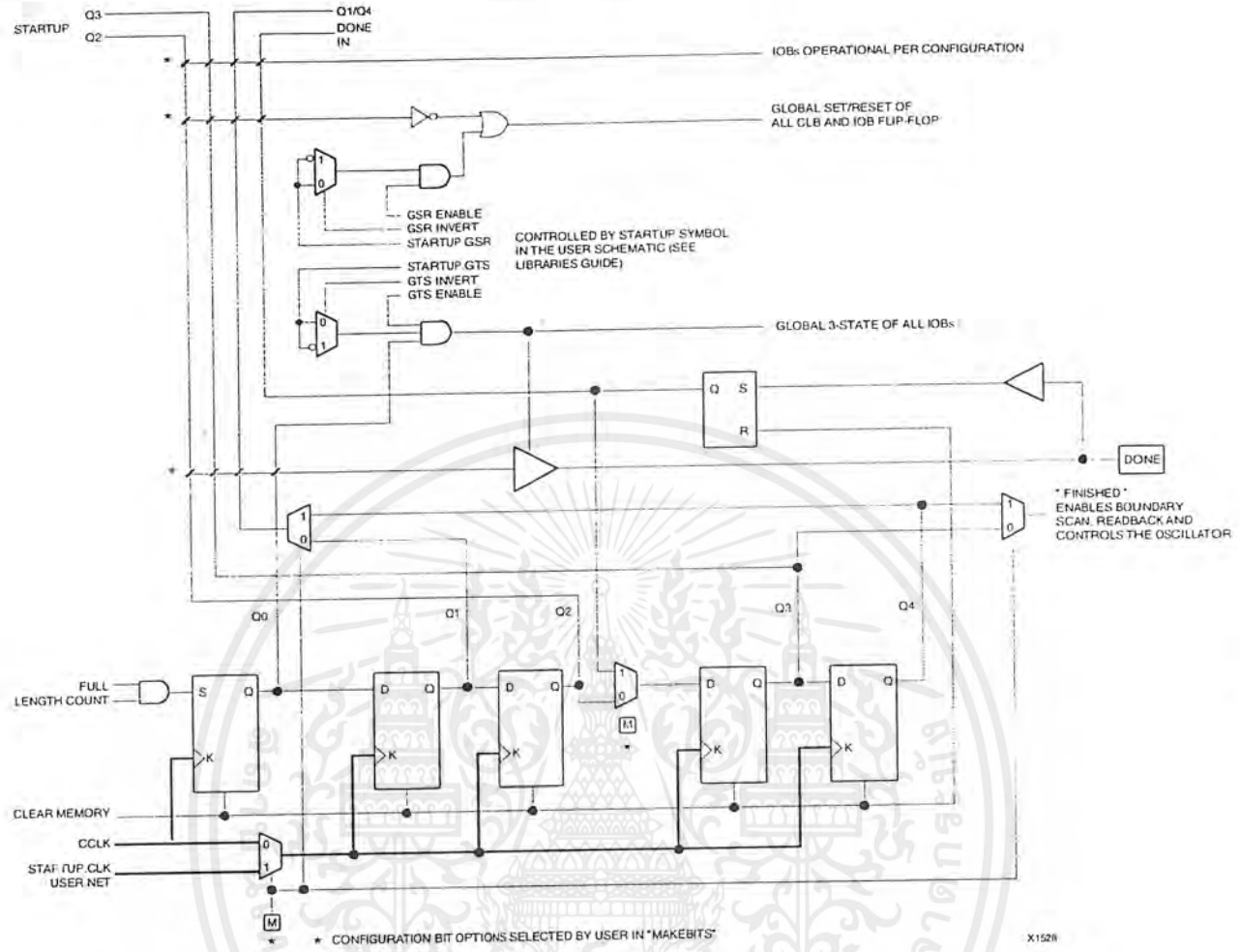


Figure 48: Start-up Logic

Readback

The user can read back the content of configuration memory and the level of certain internal nodes without interfering with the normal operation of the device.

Readback not only reports the downloaded configuration bits, but can also include the present state of the device, represented by the content of all flip-flops and latches in CLBs and IOBs, as well as the content of function generators used as RAMs.

Note that in XC4000 Series devices, configuration data is *not* inverted with respect to configuration as it is in XC2000 and XC3000 families.

XC4000 Series Readback does not use any dedicated pins, but uses four internal nets (RDBK.TRIG, RDBK.DATA, RDBK.RIP and RDBK.CLK) that can be routed to any IOB. To access the internal Readback signals, place the READBACK library symbol and attach the appropriate pad symbols, as shown in Figure 49.

After Readback has been initiated by a High level on RDBK.TRIG after configuration, the RDBK.RIP (Read In Progress) output goes High on the next rising edge of RDBK.CLK. Subsequent rising edges of this clock shift out Readback data on the RDBK.DATA net.

Readback data does not include the preamble, but starts with five dummy bits (all High) followed by the Start bit (Low) of the first frame. The first two data bits of the first frame are always High.

Each frame ends with four error check bits. They are read back as High. The last seven bits of the last frame are also read back as High. An additional Start bit (Low) and an 11-bit Cyclic Redundancy Check (CRC) signature follow, before RDBK.RIP returns Low.

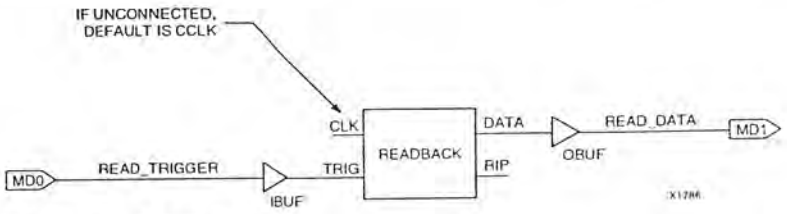


Figure 49: Readback Schematic Example

Readback Options

Readback options are: Read Capture, Read Abort, and Clock Select. They are set with the bitstream generation software.

Read Capture

When the Read Capture option is selected, the readback data stream includes sampled values of CLB and IOB signals. The rising edge of RDBK.TRIG latches the inverted values of the four CLB outputs, the IOB output flip-flops and the input signals I1 and I2. Note that while the bits describing configuration (interconnect, function generators, and RAM content) are *not* inverted, the CLB and IOB output signals *are* inverted.

When the Read Capture option is not selected, the values of the capture bits reflect the configuration data originally written to those memory locations.

If the RAM capability of the CLBs is used, RAM data are available in readback, since they directly overwrite the F and G function-table configuration of the CLB.

RDBK.TRIG is located in the lower-left corner of the device, as shown in Figure 50.

Read Abort

When the Read Abort option is selected, a High-to-Low transition on RDBK.TRIG terminates the readback operation and prepares the logic to accept another trigger.

After an aborted readback, additional clocks (up to one readback clock per configuration frame) may be required to re-initialize the control logic. The status of readback is indicated by the output control net RDBK.RIP. RDBK.RIP is High whenever a readback is in progress.

Clock Select

CCLK is the default clock. However, the user can insert another clock on RDBK.CLK. Readback control and data are clocked on rising edges of RDBK.CLK. If readback must be inhibited for security reasons, the readback control nets are simply not connected.

RDBK.CLK is located in the lower right chip corner, as shown in Figure 50.

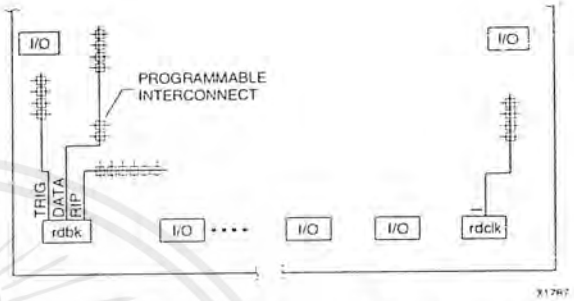


Figure 50: READBACK Symbol in Graphical Editor

Violating the Maximum High and Low Time Specification for the Readback Clock

The readback clock has a maximum High and Low time specification. In some cases, this specification cannot be met. For example, if a processor is controlling readback, an interrupt may force it to stop in the middle of a readback. This necessitates stopping the clock, and thus violating the specification.

The specification is mandatory only on clocking data at the end of a frame prior to the next start bit. The transfer mechanism will load the data to a shift register during the last six clock cycles of the frame, prior to the start bit of the following frame. This loading process is dynamic, and is the source of the maximum High and Low time requirements.

Therefore, the specification only applies to the six clock cycles prior to and including any start bit, including the clocks before the first start bit in the readback data stream. At other times, the frame data is already in the register and the register is not dynamic. Thus, it can be shifted out just like a regular shift register.

The user must precisely calculate the location of the readback data relative to the frame. The system must keep track of the position within a data frame, and disable interrupts before frame boundaries. Frame lengths and data formats are listed in Table 19, Table 20 and Table 21.

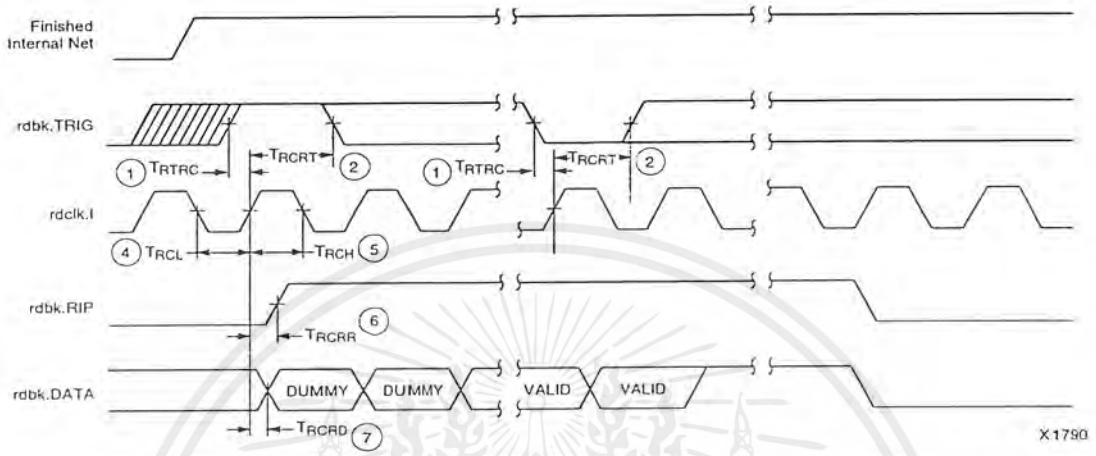
Readback with the XChecker Cable

The XChecker Universal Download/Readback Cable and Logic Probe uses the readback feature for bitstream verification. It can also display selected internal signals on the PC or workstation screen, functioning as a low-cost in-circuit emulator.

XC4000E/EX/XL Program Readback Switching Characteristic Guidelines

Testing of the switching parameters is modeled after testing methods specified by MIL-M-38510/605. All devices are 100% functionally tested. Internal timing parameters are not measured directly. They are derived from benchmark timing patterns that are taken at device introduction, prior to any process improvements.

The following guidelines reflect worst-case values over the recommended operating conditions.



E/EX

	Description	Symbol		Min	Max	Units
rdbk.TRIG	rdbk.TRIG setup to initiate and abort Readback	1	T_{RTRC}	200	-	ns
	rdbk.TRIG hold to initiate and abort Readback	2	T_{RCRT}	50	-	ns
rdclk.1	rdbk.DATA delay	7	T_{RCRD}	-	250	ns
	rdbk.RIP delay	6	T_{RCRR}	-	250	ns
	High time	5	T_{RCH}	250	500	ns
	Low time	4	T_{RCL}	250	500	ns

Note 1: Timing parameters apply to all speed grades.
Note 2: If rdbk.TRIG is High prior to Finished, Finished will trigger the first Readback.

XL

	Description	Symbol		Min	Max	Units
rdbk.TRIG	rdbk.TRIG setup to initiate and abort Readback	1	T_{RTRC}	200	-	ns
	rdbk.TRIG hold to initiate and abort Readback	2	T_{RCRT}	50	-	ns
rdclk.1	rdbk.DATA delay	7	T_{RCRD}	-	250	ns
	rdbk.RIP delay	6	T_{RCRR}	-	250	ns
	High time	5	T_{RCH}	250	500	ns
	Low time	4	T_{RCL}	250	500	ns

Note 1: Timing parameters apply to all speed grades.
Note 2: If rdbk.TRIG is High prior to Finished, Finished will trigger the first Readback.

Table 22: Pin Functions During Configuration

CONFIGURATION MODE <M2:M1:M0>						
SLAVE SERIAL <1:1:1>	MASTER SERIAL <0:0:0>	SYNCH. PERIPHERAL <0:1:1>	ASYNCH. PERIPHERAL <1:0:1>	MASTER PARALLEL DOWN <1:1:0>	MASTER PARALLEL UP <1:0:0>	USER OPERATION
M2(HIGH) (I)	M2(LOW) (I)	M2(LOW) (I)	M2(HIGH) (I)	M2(HIGH) (I)	M2(HIGH) (I)	(I)
M1(HIGH) (I)	M1(LOW) (I)	M1(HIGH) (I)	M1(LOW) (I)	M1(HIGH) (I)	M1(LOW) (I)	(O)
M0(HIGH) (I)	M0(LOW) (I)	M0(HIGH) (I)	M0(HIGH) (I)	M0(LOW) (I)	M0(LOW) (I)	(I)
HDC (HIGH)	HDC (HIGH)	HDC (HIGH)	HDC (HIGH)	HDC (HIGH)	HDC (HIGH)	I/O
$\overline{\text{LDC}}$ (LOW)	$\overline{\text{LDC}}$ (LOW)	$\overline{\text{LDC}}$ (LOW)	$\overline{\text{LDC}}$ (LOW)	$\overline{\text{LDC}}$ (LOW)	$\overline{\text{LDC}}$ (LOW)	I/O
$\overline{\text{INIT}}$	$\overline{\text{INIT}}$	$\overline{\text{INIT}}$	$\overline{\text{INIT}}$	$\overline{\text{INIT}}$	$\overline{\text{INIT}}$	I/O
DONE	DONE	DONE	DONE	DONE	DONE	DONE
$\overline{\text{PROGRAM}}$ (I)	$\overline{\text{PROGRAM}}$ (I)	$\overline{\text{PROGRAM}}$ (I)	$\overline{\text{PROGRAM}}$ (I)	$\overline{\text{PROGRAM}}$ (I)	$\overline{\text{PROGRAM}}$ (I)	$\overline{\text{PROGRAM}}$
CCLK (I)	CCLK (O)	CCLK (I)	CCLK (O)	CCLK (O)	CCLK (O)	CCLK (I)
		RDY/$\overline{\text{BUSY}}$ (O)	RDY/$\overline{\text{BUSY}}$ (O)	$\overline{\text{RCLK}}$ (O)	$\overline{\text{RCLK}}$ (O)	I/O
			$\overline{\text{RS}}$ (I)			I/O
			CS0 (I)			I/O
		DATA 7 (I)	DATA 7 (I)	DATA 7 (I)	DATA 7 (I)	I/O
		DATA 6 (I)	DATA 6 (I)	DATA 6 (I)	DATA 6 (I)	I/O
		DATA 5 (I)	DATA 5 (I)	DATA 5 (I)	DATA 5 (I)	I/O
		DATA 4 (I)	DATA 4 (I)	DATA 4 (I)	DATA 4 (I)	I/O
		DATA 3 (I)	DATA 3 (I)	DATA 3 (I)	DATA 3 (I)	I/O
		DATA 2 (I)	DATA 2 (I)	DATA 2 (I)	DATA 2 (I)	I/O
		DATA 1 (I)	DATA 1 (I)	DATA 1 (I)	DATA 1 (I)	I/O
DIN (I)	DIN (I)	DATA 0 (I)	DATA 0 (I)	DATA 0 (I)	DATA 0 (I)	I/O
DOUT	DOUT	DOUT	DOUT	DOUT	DOUT	SGCK4-GCK5-I/O
TDI	TDI	TDI	TDI	TDI	TDI	TDI-I/O
TCK	TCK	TCK	TCK	TCK	TCK	TCK-I/O
TMS	TMS	TMS	TMS	TMS	TMS	TMS-I/O
TDO	TDO	TDO	TDO	TDO	TDO	TDO-(O)
			$\overline{\text{WS}}$ (I)	A0	A0	I/O
				A1	A1	PGCK4-GCK6-I/O
			CS1	A2	A2	I/O
				A3	A3	I/O
				A4	A4	I/O
				A5	A5	I/O
				A6	A6	I/O
				A7	A7	I/O
				A8	A8	I/O
				A9	A9	I/O
				A10	A10	I/O
				A11	A11	I/O
				A12	A12	I/O
				A13	A13	I/O
				A14	A14	I/O
				A15	A15	SGCK1-GCK7-I/O
				A16	A16	PGCK1-GCK8-I/O
				A17	A17	I/O
				A18*	A18*	I/O
				A19*	A19*	I/O
				A20*	A20*	I/O
				A21*	A21*	I/O
						ALL OTHERS

Table 23: Pin Functions During Configuration

CONFIGURATION MODE <M2:M1:M0>						
SLAVE SERIAL <1:1:1>	MASTER SERIAL <0:0:0>	SYNCH. PERIPHERAL <0:1:1>	ASYNCH. PERIPHERAL <1:0:1>	MASTER PARALLEL DOWN <1:1:0>	MASTER PARALLEL UP <1:0:0>	USER OPERATION
M2(HIGH) (I)	M2(LOW) (I)	M2(LOW) (I)	M2(HIGH) (I)	M2(HIGH) (I)	M2(HIGH) (I)	(I)
M1(HIGH) (I)	M1(LOW) (I)	M1(HIGH) (I)	M1(LOW) (I)	M1(HIGH) (I)	M1(LOW) (I)	(O)
M0(HIGH) (I)	M0(LOW) (I)	M0(HIGH) (I)	M0(HIGH) (I)	M0(LOW) (I)	M0(LOW) (I)	(I)
HDC (HIGH)	HDC (HIGH)	HDC (HIGH)	HDC (HIGH)	HDC (HIGH)	HDC (HIGH)	I/O
$\overline{\text{LDC}}$ (LOW)	$\overline{\text{LDC}}$ (LOW)	$\overline{\text{LDC}}$ (LOW)	$\overline{\text{LDC}}$ (LOW)	$\overline{\text{LDC}}$ (LOW)	$\overline{\text{LDC}}$ (LOW)	I/O
$\overline{\text{INIT}}$	$\overline{\text{INIT}}$	$\overline{\text{INIT}}$	$\overline{\text{INIT}}$	$\overline{\text{INIT}}$	$\overline{\text{INIT}}$	I/O
DONE	DONE	DONE	DONE	DONE	DONE	DONE
$\overline{\text{PROGRAM}}$ (I)	$\overline{\text{PROGRAM}}$ (I)	$\overline{\text{PROGRAM}}$ (I)	$\overline{\text{PROGRAM}}$ (I)	$\overline{\text{PROGRAM}}$ (I)	$\overline{\text{PROGRAM}}$ (I)	$\overline{\text{PROGRAM}}$
CCLK (I)	CCLK (O)	CCLK (I)	CCLK (O)	CCLK (O)	CCLK (O)	CCLK (I)
		RDY/$\overline{\text{BUSY}}$ (O)	RDY/$\overline{\text{BUSY}}$ (O)	$\overline{\text{RCLK}}$ (O)	$\overline{\text{RCLK}}$ (O)	I/O
			$\overline{\text{RS}}$ (I)			I/O
			$\overline{\text{CS0}}$ (I)			I/O
		DATA 7 (I)	DATA 7 (I)	DATA 7 (I)	DATA 7 (I)	I/O
		DATA 6 (I)	DATA 6 (I)	DATA 6 (I)	DATA 6 (I)	I/O
		DATA 5 (I)	DATA 5 (I)	DATA 5 (I)	DATA 5 (I)	I/O
		DATA 4 (I)	DATA 4 (I)	DATA 4 (I)	DATA 4 (I)	I/O
		DATA 3 (I)	DATA 3 (I)	DATA 3 (I)	DATA 3 (I)	I/O
		DATA 2 (I)	DATA 2 (I)	DATA 2 (I)	DATA 2 (I)	I/O
		DATA 1 (I)	DATA 1 (I)	DATA 1 (I)	DATA 1 (I)	I/O
DIN (I)	DIN (I)	DATA 0 (I)	DATA 0 (I)	DATA 0 (I)	DATA 0 (I)	I/O
DOUT	DOUT	DOUT	DOUT	DOUT	DOUT	SGCK4-GCK5-I/O
TDI	TDI	TDI	TDI	TDI	TDI	TDI-I/O
TCK	TCK	TCK	TCK	TCK	TCK	TCK-I/O
TMS	TMS	TMS	TMS	TMS	TMS	TMS-I/O
TDO	TDO	TDO	TDO	TDO	TDO	TDO-(O)
			WS (I)	A0	A0	I/O
				A1	A1	PGCK4-GCK6-I/O
			CS1	A2	A2	I/O
				A3	A3	I/O
				A4	A4	I/O
				A5	A5	I/O
				A6	A6	I/O
				A7	A7	I/O
				A8	A8	I/O
				A9	A9	I/O
				A10	A10	I/O
				A11	A11	I/O
				A12	A12	I/O
				A13	A13	I/O
				A14	A14	I/O
				A15	A15	SGCK1-GCK7-I/O
				A16	A16	PGCK1-GCK8-I/O
				A17	A17	I/O
				A18*	A18*	I/O
				A19*	A19*	I/O
				A20*	A20*	I/O
				A21*	A21*	I/O
						ALL OTHERS

* XC4000X only

Notes 1. A shaded table cell represents a 50 kΩ - 100 kΩ pull-up before and during configuration.
2. (I) represents an input; (O) represents an output.
3. $\overline{\text{INIT}}$ is an open-drain output during configuration.

Configuration Timing

The seven configuration modes are discussed in detail in this section. Timing specifications are included.

Slave Serial Mode

In Slave Serial mode, an external signal drives the CCLK input of the FPGA. The serial configuration bitstream must be available at the DIN input of the lead FPGA a short setup time before each rising CCLK edge.

The lead FPGA then presents the preamble data—and all data that overflows the lead device—on its DOUT pin. There is an internal delay of 0.5 CCLK periods, which means that DOUT changes on the falling CCLK edge, and the next FPGA in the daisy chain accepts data on the subsequent rising CCLK edge.

Figure 51 shows a full master/slave system. An XC4000 Series device in Slave Serial mode should be connected as shown in the third device from the left.

Slave Serial mode is selected by a <111> on the mode pins (M2, M1, M0). Slave Serial is the default mode if the mode pins are left unconnected, as they have weak pull-up resistors during configuration.

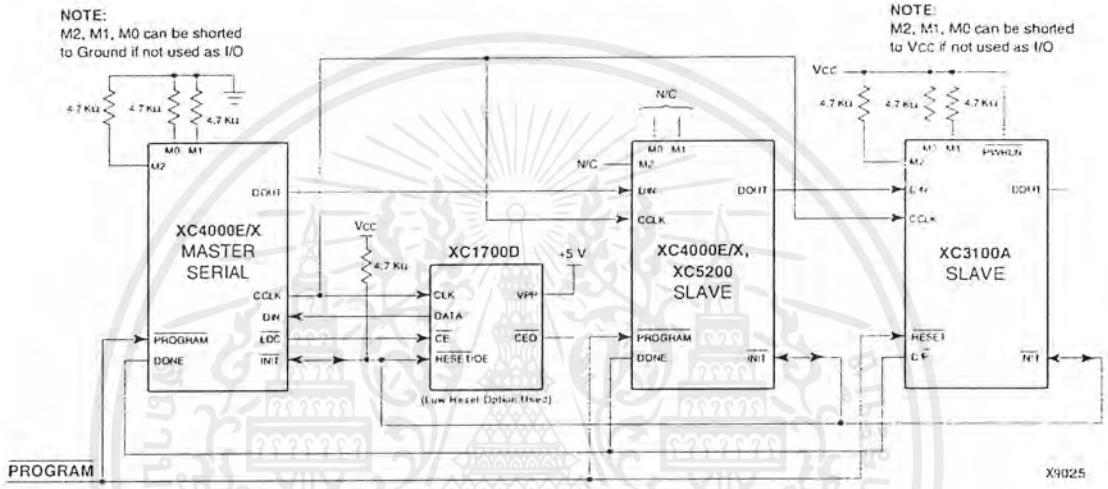
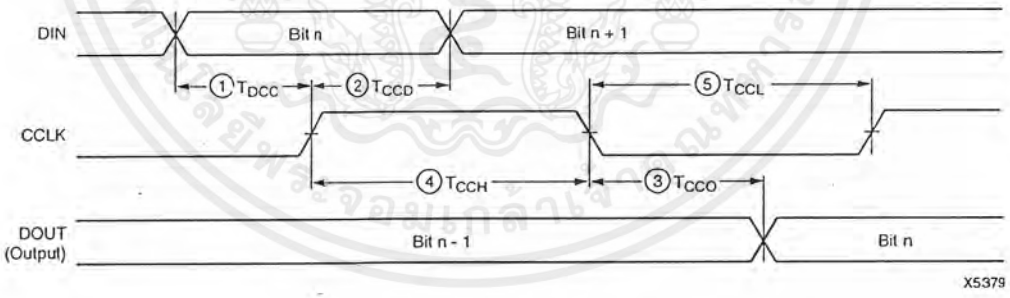


Figure 51: Master/Slave Serial Mode Circuit Diagram



	Description	Symbol		Min	Max	Units
CCLK	DIN setup	1	T_{DCC}	20		ns
	DIN hold	2	T_{CCD}	0		ns
	DIN to DOUT	3	T_{CCO}		30	ns
	High time	4	T_{CCH}	45		ns
	Low time	5	T_{CCL}	45		ns
	Frequency		F_{CC}		10	MHz

Note: Configuration must be delayed until the $\overline{INIT}$ pins of all daisy-chained FPGAs are High.

Figure 52: Slave Serial Mode Programming Switching Characteristics

Master Serial Mode

In Master Serial mode, the CCLK output of the lead FPGA drives a Xilinx Serial PROM that feeds the FPGA DIN input. Each rising edge of the CCLK output increments the Serial PROM internal address counter. The next data bit is put on the SPROM data output, connected to the FPGA DIN pin. The lead FPGA accepts this data on the subsequent rising CCLK edge.

The lead FPGA then presents the preamble data—and all data that overflows the lead device—on its DOUT pin. There is an internal pipeline delay of 1.5 CCLK periods, which means that DOUT changes on the falling CCLK edge, and the next FPGA in the daisy chain accepts data on the subsequent rising CCLK edge.

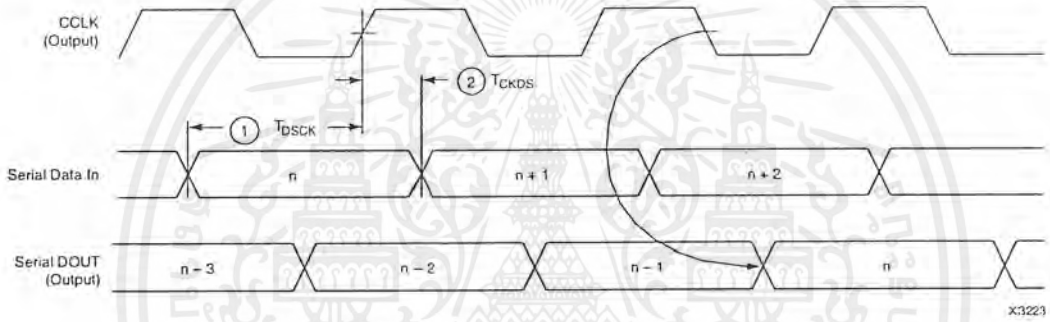
In the bitstream generation software, the user can specify Fast ConfigRate, which, starting several bits into the first frame, increases the CCLK frequency by a factor of eight.

For actual timing values please refer to "Configuration Switching Characteristics" on page 68. Be sure that the serial PROM and slaves are fast enough to support this data rate. XC2000, XC3000/A, and XC3100A devices do not support the Fast ConfigRate option.

The SPROM CE input can be driven from either $\overline{LDC}$ or DONE. Using $\overline{LDC}$ avoids potential contention on the DIN pin, if this pin is configured as user-I/O, but $\overline{LDC}$ is then restricted to be a permanently High user output after configuration. Using DONE can also avoid contention on DIN, provided the early DONE option is invoked.

Figure 51 on page 60 shows a full master/slave system. The leftmost device is in Master Serial mode.

Master Serial mode is selected by a <000> on the mode pins (M2, M1, M0).



	Description	Symbol		Min	Max	Units
CCLK	DIN setup	1	T_{DSCK}	20		ns
	DIN hold	2	T_{CKDS}	0		ns

Notes: 1. At power-up, Vcc must rise from 2.0 V to Vcc min in less than 25 ms, otherwise delay configuration by pulling PROGRAM Low until Vcc is valid.
2. Master Serial mode timing is based on testing in slave mode.

Figure 53: Master Serial Mode Programming Switching Characteristics

Master Parallel Modes

In the two Master Parallel modes, the lead FPGA directly addresses an industry-standard byte-wide EPROM, and accepts eight data bits just before incrementing or decrementing the address outputs.

The eight data bits are serialized in the lead FPGA, which then presents the preamble data—and all data that overflows the lead device—on its DOUT pin. There is an internal delay of 1.5 CCLK periods, after the rising CCLK edge that accepts a byte of data (and also changes the EPROM address) until the falling CCLK edge that makes the LSB (D0) of this byte appear at DOUT. This means that DOUT changes on the falling CCLK edge, and the next FPGA in the daisy chain accepts data on the subsequent rising CCLK edge.

The PROM address pins can be incremented or decremented, depending on the mode pin settings. This option allows the FPGA to share the PROM with a wide variety of microprocessors and micro controllers. Some processors must boot from the bottom of memory (all zeros) while others must boot from the top. The FPGA is flexible and can load its configuration bitstream from either end of the memory.

Master Parallel Up mode is selected by a <100> on the mode pins (M2, M1, M0). The EPROM addresses start at 00000 and increment.

Master Parallel Down mode is selected by a <110> on the mode pins. The EPROM addresses start at 3FFFF and decrement.

Additional Address lines in XC4000 devices

The XC4000X devices have additional address lines (A18-A21) allowing the additional address space required to daisy-chain several large devices.

The extra address lines are programmable in XC4000EX devices. By default these address lines are not activated. In the default mode, the devices are compatible with existing XC4000 and XC4000E products. If desired, the extra address lines can be used by specifying the address lines option in bitgen as 22 (bitgen -g AddressLines:22). The lines (A18-A21) are driven when a master device detects, via the bitstream, that it should be using all 22 address lines. Because these pins will initially be pulled high by internal pull-ups, designers using Master Parallel Up mode should use external pull down resistors on pins A18-A21. If Master Parallel Down mode is used external resistors are not necessary.

All 22 address lines are always active in Master Parallel modes with XC4000XL devices. The additional address lines behave identically to the lower order address lines. If the Address Lines option in bitgen is set to 18, it will be ignored by the XC4000XL device.

The additional address lines (A18-A21) are not available in the PC84 package.

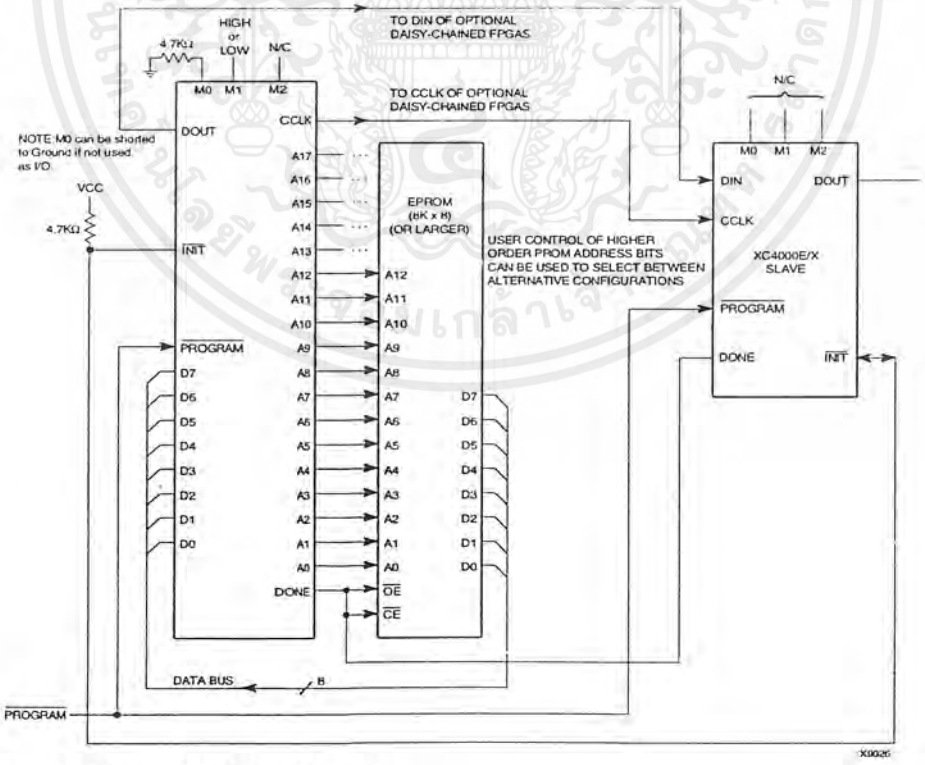
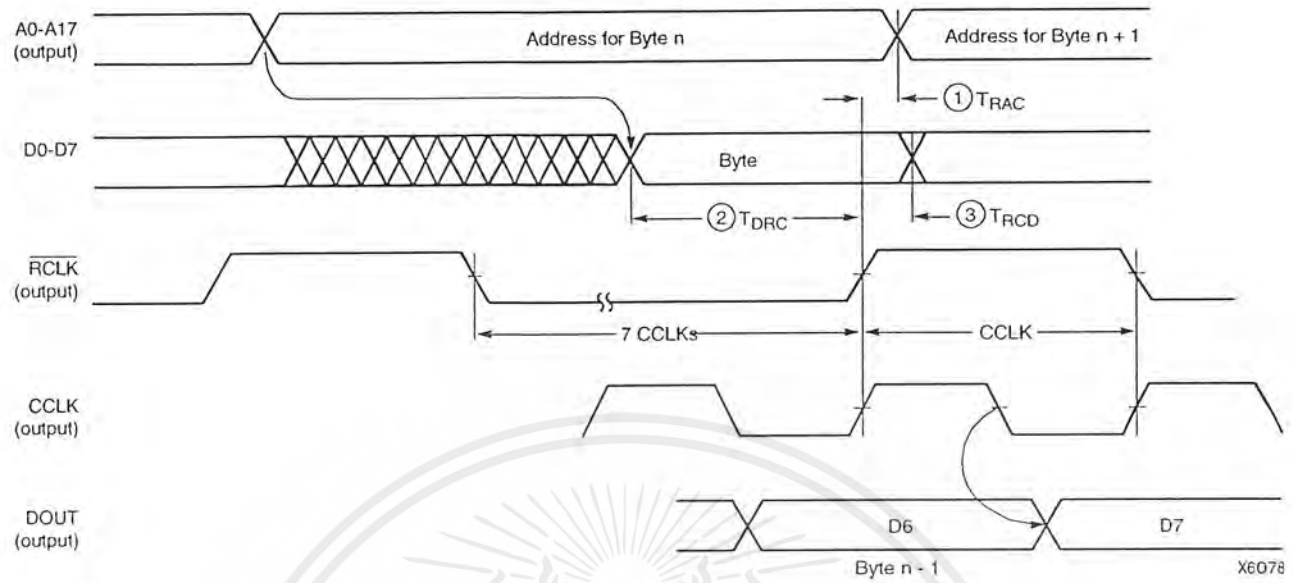


Figure 54: Master Parallel Mode Circuit Diagram



	Description	Symbol		Min	Max	Units
RCLK	Delay to Address valid	1	T_{RAC}	0	200	ns
	Data setup time	2	T_{DRC}	60		ns
	Data hold time	3	T_{RCD}	0		ns

Notes: 1. At power-up, Vcc must rise from 2.0 V to Vcc min in less than 25 ms, otherwise delay configuration by pulling $\overline{\text{PROGRAM}}$ Low until Vcc is valid.
2. The first Data byte is loaded and CCLK starts at the end of the first $\overline{\text{RCLK}}$ active cycle (rising edge).

This timing diagram shows that the EPROM requirements are extremely relaxed. EPROM access time can be longer than 500 ns. EPROM data output has no hold-time requirements.

Figure 55: Master Parallel Mode Programming Switching Characteristics

Synchronous Peripheral Mode

Synchronous Peripheral mode can also be considered Slave Parallel mode. An external signal drives the CCLK input(s) of the FPGA(s). The first byte of parallel configuration data must be available at the Data inputs of the lead FPGA a short setup time before the rising CCLK edge. Subsequent data bytes are clocked in on every eighth consecutive rising CCLK edge.

The same CCLK edge that accepts data, also causes the RDY/$\overline{\text{BUSY}}$ output to go High for one CCLK period. The pin name is a misnomer. In Synchronous Peripheral mode it is really an ACKNOWLEDGE signal. Synchronous operation does not require this response, but it is a meaningful signal for test purposes. Note that RDY/$\overline{\text{BUSY}}$ is pulled High with a high-impedance pullup prior to $\overline{\text{INIT}}$ going High.

The lead FPGA serializes the data and presents the preamble data (and all data that overflows the lead device) on its DOUT pin. There is an internal delay of 1.5 CCLK periods, which means that DOUT changes on the falling CCLK edge, and the next FPGA in the daisy chain accepts data on the subsequent rising CCLK edge.

In order to complete the serial shift operation, 10 additional CCLK rising edges are required after the last data byte has been loaded, plus one more CCLK cycle for each daisy-chained device.

Synchronous Peripheral mode is selected by a <011> on the mode pins (M2, M1, M0).

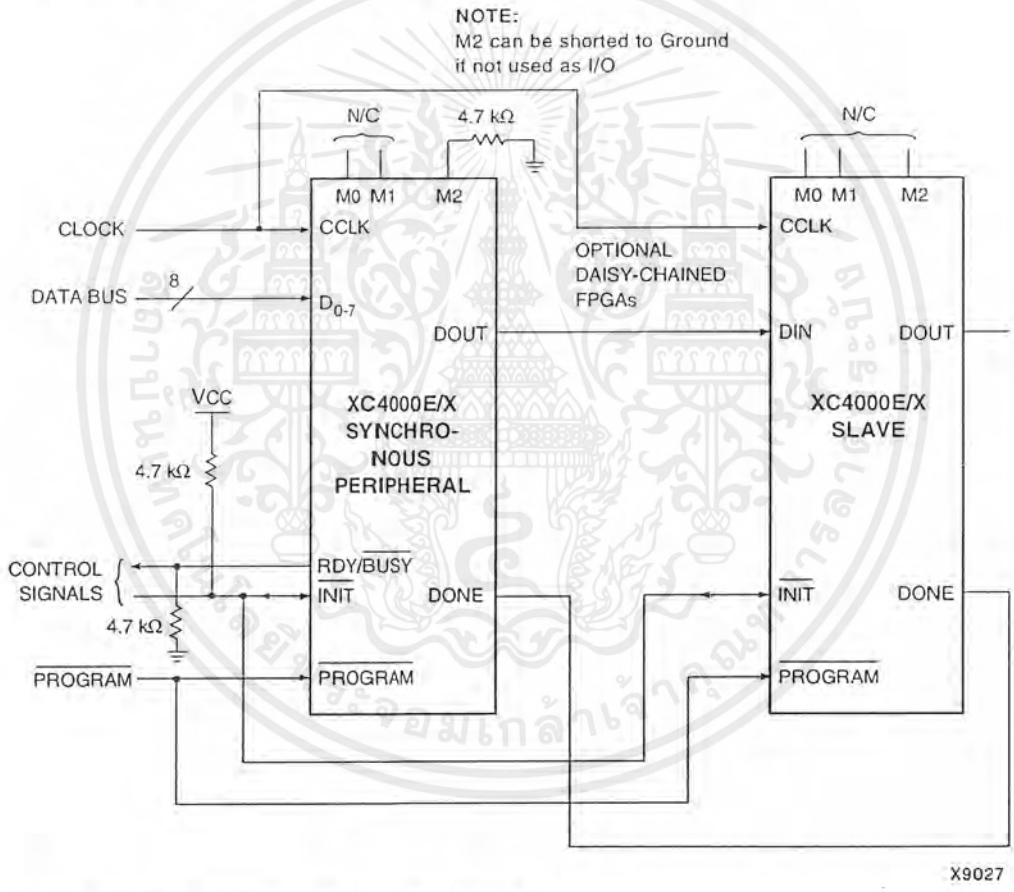
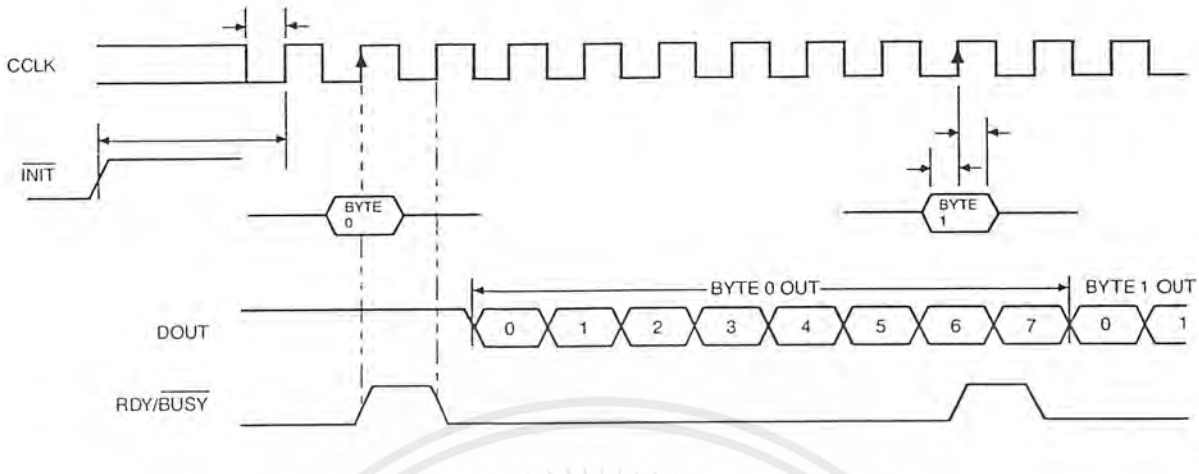


Figure 56: Synchronous Peripheral Mode Circuit Diagram



	Description	Symbol	Min	Max	Units
CCLK	INIT (High) setup time	T_{IC}	5		µs
	D0 - D7 setup time	T_{DC}	60		ns
	D0 - D7 hold time	T_{CD}	0		ns
	CCLK High time	T_{CCH}	50		ns
	CCLK Low time	T_{CCL}	60		ns
	CCLK Frequency	F_{CC}		8	MHz

Notes:
1. Peripheral Synchronous mode can be considered Slave Parallel mode. An external CCLK provides timing, clocking in the first data byte on the second rising edge of CCLK after INIT goes High. Subsequent data bytes are clocked in on every eighth consecutive rising edge of CCLK.
2. The RDY/BUSY line goes High for one CCLK period after data has been clocked in, although synchronous operation does not require such a response.
3. The pin name RDY/BUSY is a misnomer. In Synchronous Peripheral mode this is really an ACKNOWLEDGE signal.
4. Note that data starts to shift out serially on the DOUT pin 0.5 CCLK periods after it was loaded in parallel. Therefore, additional CCLK pulses are clearly required after the last byte has been loaded.

Figure 57: Synchronous Peripheral Mode Programming Switching Characteristics

Asynchronous Peripheral Mode

Write to FPGA

Asynchronous Peripheral mode uses the trailing edge of the logic AND condition of $\overline{WS}$ and $\overline{CS0}$ being Low and $\overline{RS}$ and CS1 being High to accept byte-wide data from a microprocessor bus. In the lead FPGA, this data is loaded into a double-buffered UART-like parallel-to-serial converter and is serially shifted into the internal logic.

The lead FPGA presents the preamble data (and all data that overflows the lead device) on its DOUT pin. The RDY/$\overline{BUSY}$ output from the lead FPGA acts as a handshake signal to the microprocessor. RDY/$\overline{BUSY}$ goes Low when a byte has been received, and goes High again when the byte-wide input buffer has transferred its information into the shift register, and the buffer is ready to receive new data. A new write may be started immediately, as soon as the RDY/$\overline{BUSY}$ output has gone Low, acknowledging receipt of the previous data. Write may not be terminated until RDY/$\overline{BUSY}$ is High again for one CCLK period. Note that RDY/$\overline{BUSY}$ is pulled High with a high-impedance pull-up prior to $\overline{INIT}$ going High.

The length of the $\overline{BUSY}$ signal depends on the activity in the UART. If the shift register was empty when the new byte was received, the $\overline{BUSY}$ signal lasts for only two CCLK periods. If the shift register was still full when the new byte was received, the $\overline{BUSY}$ signal can be as long as nine CCLK periods.

Note that after the last byte has been entered, only seven of its bits are shifted out. CCLK remains High with DOUT equal to bit 6 (the next-to-last bit) of the last byte entered.

The READY/$\overline{BUSY}$ handshake can be ignored if the delay from any one Write to the end of the next Write is guaranteed to be longer than 10 CCLK periods.

Status Read

The logic AND condition of the $\overline{CS0}$, CS1and $\overline{RS}$ inputs puts the device status on the Data bus.

- D7 High indicates Ready
- D7 Low indicates Busy
- D0 through D6 go unconditionally High

It is mandatory that the whole start-up sequence be started and completed by one byte-wide input. Otherwise, the pins used as Write Strobe or Chip Enable might become active outputs and interfere with the final byte transfer. If this transfer does not occur, the start-up sequence is not completed all the way to the finish (point F in Figure 47 on page 53).

In this case, at worst, the internal reset is not released. At best, Readback and Boundary Scan are inhibited. The length-count value, as generated by the XACT*step* software, ensures that these problems never occur.

Although RDY/$\overline{BUSY}$ is brought out as a separate signal, microprocessors can more easily read this information on one of the data lines. For this purpose, D7 represents the RDY/$\overline{BUSY}$ status when $\overline{RS}$ is Low, $\overline{WS}$ is High, and the two chip select lines are both active.

Asynchronous Peripheral mode is selected by a <101> on the mode pins (M2, M1, M0).

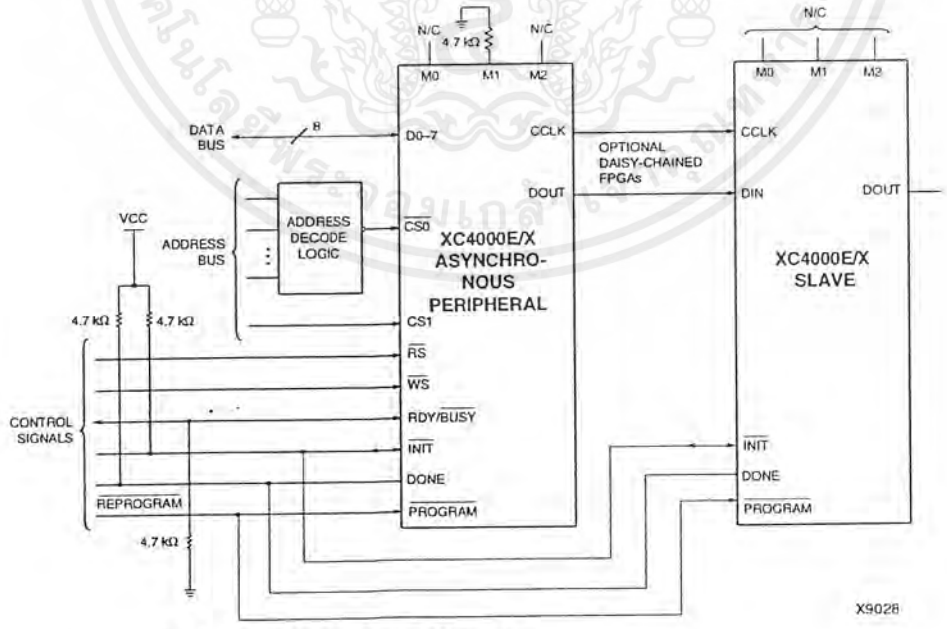
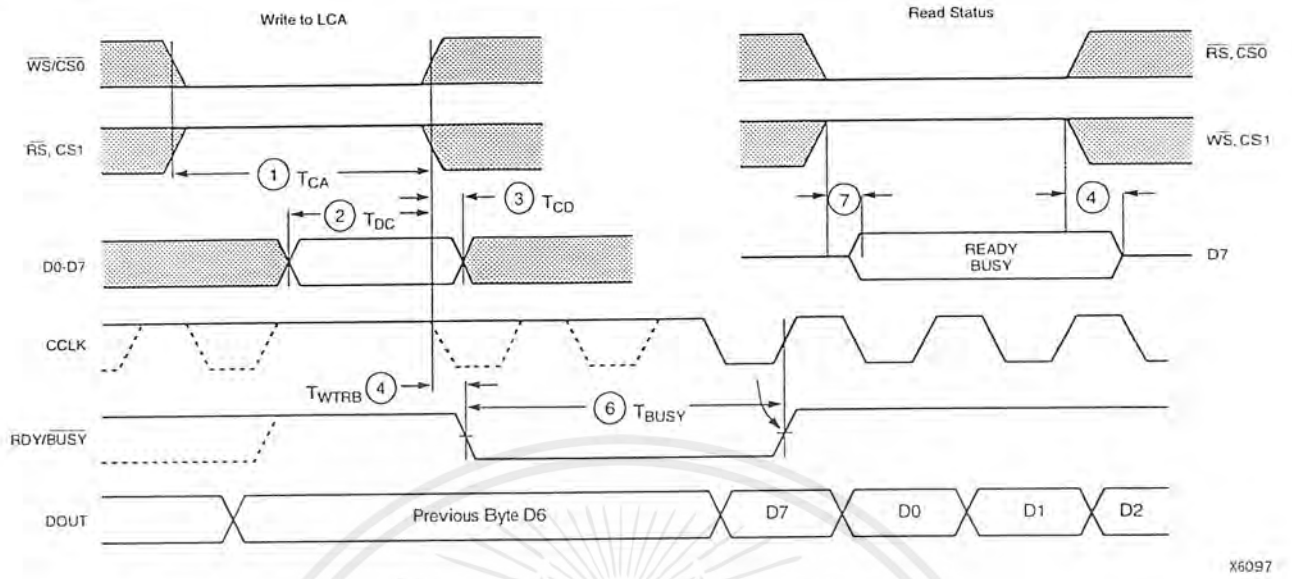


Figure 58: Asynchronous Peripheral Mode Circuit Diagram



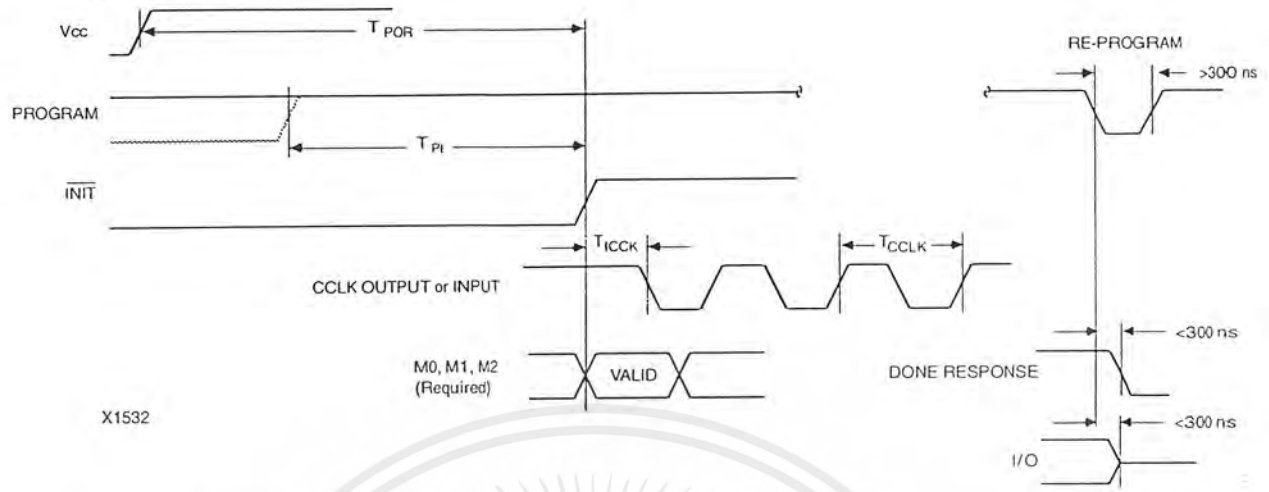
	Description	Symbol		Min	Max	Units
Write	Effective Write time ($\overline{CS0}$, $\overline{WS}$=Low; $\overline{RS}$, CS1=High)	1	T_{CA}	100		ns
	DIN setup time	2	T_{DC}	60		ns
	DIN hold time	3	T_{CD}	0		ns
RDY	RDY/$\overline{BUSY}$ delay after end of Write or Read	4	T_{WTRB}		60	ns
	RDY/$\overline{BUSY}$ active after beginning of Read	7			60	ns
	RDY/$\overline{BUSY}$ Low output (Note 4)	6	T_{BUSY}	2	9	CCLK periods

Notes:
1. Configuration must be delayed until the $\overline{INIT}$ pins of all daisy-chained FPGAs are High.
2. The time from the end of $\overline{WS}$ to CCLK cycle for the new byte of data depends on the completion of previous byte processing and the phase of the internal timing generator for CCLK.
3. CCLK and DOUT timing is tested in slave mode.
4. T_{BUSY} indicates that the double-buffered parallel-to-serial converter is not yet ready to receive new data. The shortest T_{BUSY} occurs when a byte is loaded into an empty parallel-to-serial converter. The longest T_{BUSY} occurs when a new word is loaded into the input register before the second-level buffer has started shifting out data

This timing diagram shows very relaxed requirements. Data need not be held beyond the rising edge of $\overline{WS}$. RDY/$\overline{BUSY}$ will go active within 60 ns after the end of $\overline{WS}$. A new write may be asserted immediately after RDY/$\overline{BUSY}$ goes Low, but write may not be terminated until RDY/$\overline{BUSY}$ has been High for one CCLK period.

Figure 59: Asynchronous Peripheral Mode Programming Switching Characteristics

Configuration Switching Characteristics



Master Modes (XC4000E/EX)

Description		Symbol	Min	Max	Units
Power-On Reset	M0 = High	T_{POR}	10	40	ms
	M0 = Low	T_{POR}	40	130	ms
Program Latency		T_{PI}	30	200	µs per CLB column
CCLK (output) Delay		T_{ICCK}	40	250	µs
CCLK (output) Period, slow		T_{CCLK}	640	2000	ns
CCLK (output) Period, fast		T_{CCLK}	80	250	ns

Master Modes (XC4000XL)

Description		Symbol	Min	Max	Units
Power-On Reset	M0 = High	T_{POR}	10	40	ms
	M0 = Low	T_{POR}	40	130	ms
Program Latency		T_{PI}	30	200	µs per CLB column
CCLK (output) Delay		T_{ICCK}	40	250	µs
CCLK (output) Period, slow		T_{CCLK}	540	1600	ns
CCLK (output) Period, fast		T_{CCLK}	67	200	ns

Slave and Peripheral Modes (All)

Description	Symbol	Min	Max	Units
Power-On Reset	T_{POR}	10	33	ms
Program Latency	T_{PI}	30	200	µs per CLB column
CCLK (input) Delay (required)	T_{ICCK}	4		µs
CCLK (input) Period (required)	T_{CCLK}	100		ns

Product Availability

Table 24, Table 25, and Table 26 show the planned packages and speed grades for XC4000-Series devices. Call your local sales office for the latest availability information, or see the Xilinx WEBLINX at http://www.xilinx.com for the latest revision of the specifications.

Table 24: Component Availability Chart for XC4000XL FPGAs

	PINS	84	100	100	144	144	160	160	176	176	208	208	240	240	256	299	304	352	411	432	475	559	560
	TYPE	Plast. PLCC	Plast. PQFP	Plast. VQFP	Plast. TQFP	High-Perf. TQFP	High-Perf. QFP	Plast. PQFP	Plast. TQFP	High-Perf. TQFP	High-Perf. QFP	Plast. PQFP	High-Perf. QFP	Plast. PQFP	Plast. BGA	Ceram. PGA	High-Perf. QFP	Plast. BGA	Ceram. PGA	Plast. BGA	Ceram. PGA	Ceram. PGA	Plast. BGA
	CODE	PC84	PQ100	VQ100	TQ144	HT144	HQ160	PQ160	TQ176	HT176	HQ208	PQ208	HQ240	PQ240	BG256	PG299	HQ304	BG352	PG411	BG432	PG475	PG559	BG560
XC4002XL	-3	C I	C I	C I																			
	-2	C I	C I	C I																			
	-1	C I	C I	C I																			
	-09C	C	C	C																			
XC4005XL	-3	C I	C I	C I	C I			C I				C I											
	-2	C I	C	C I	C I			C I				C I											
	-1	C I	C I	C I	C I			C I				C I											
	-09C	C	C	C	C			C				C											
XC4010XL	-3	C I	C I		C I			C I	C I			C I			C I								
	-2	C I	C I		C I			C I	C I			C I			C I								
	-1	C I	C I		C I			C I	C I			C I			C I								
	-09C	C	C		C			C	C			C			C								
XC4013XL	-3					C I		C I		C I		C I		C I	C I								
	-2					C I		C I		C I		C I		C I	C I								
	-1					C I		C I		C I		C I		C I	C I								
	-09C					C		C		C		C		C	C								
	-08C					C		C		C		C		C	C								
XC4020XL	-3					C I		C I		C I		C I		C I	C I								
	-2					C I		C I		C I		C I		C I	C I								
	-1					C I		C I		C I		C I		C I	C I								
	-09C					C		C		C		C		C	C								
XC4028XL	-3						C I				C I		C I		C I	C I	C I	C I					
	-2						C I				C I		C I		C I	C I	C I	C I					
	-1						C I				C I		C I		C I	C I	C I	C I					
	-09C						C				C		C		C	C	C	C					
XC4036XL	-3						C I				C I		C I				C I	C I	C I	C I			
	-2						C I				C I		C				C I	C I	C I	C I			
	-1						C I				C I		C I				C I	C I	C I	C I			
	-09C						C				C		C				C	C	C	C			
	-08C						C				C		C				C	C	C	C			
XC4044XL	-3						C I				C I		C I				C I	C I	C I	C I			
	-2						C I				C I		C I				C I	C I	C I	C I			
	-1						C I				C I		C I				C I	C I	C I	C I			
	-09C						C				C		C				C	C	C	C			
XC4052XL	-3												C I				C I		C I	C I			C I
	-2												C I				C I		C I	C I			C I
	-1												C I				C I		C I	C I			C I
	-09C												C				C		C	C			C
XC4062XL	-3												C I				C I			C I	C I		C I
	-2												C I				C I			C I	C I		C I
	-1												C I				C I			C I	C I		C I
	-09C												C				C			C	C		C
	-08C												C				C			C	C		C
XC4085XL	-3																			C I		C I	C I
	-2																			C I		C I	C I
	-1																			C I		C I	C I
	-09C																			C		C	C

1/29/99

C = Commercial T_J = 0° to +85°C

I= Industrial T_J = -40°C to +100°C

Table 25: Component Availability Chart for XC4000E FPGAs

	PINS	84	100	100	120	144	156	160	191	208	208	223	225	240	240	299	304
	TYPE	Plast. PLCC	Plast. PQFP	Plast. VQFP	Ceram. PGA	Plast. TQFP	Ceram. PGA	Plast. PQFP	Ceram. PGA	High-Perf. QFP	Plast. PQFP	Ceram. PGA	Plast. BGA	High-Perf. QFP	Plast. PQFP	Ceram. PGA	High-Perf QF
	CODE	PC84	PQ100	VQ100	PG120	TQ144	PG156	PQ160	PG191	HQ208	PQ208	PG223	BG225	HQ240	PQ240	PG299	HQ304
XC4003E	-4	C I	C I	C I	C I												
	-3	C I	C I	C I	C I												
	-2	C I	C I	C I	C I												
	-1	C	C	C	C												
XC4005E	-4	C I	C I			C I	C I	C I			C I						
	-3	C I	C I			C I	C I	C I			C I						
	-2	C I	C I			C I	C I	C I			C I						
	-1	C	C			C	C	C			C						
XC4006E	-4	C I				C I	C I	C I			C I						
	-3	C I				C I	C I	C I			C I						
	-2	C I				C I	C I	C I			C I						
	-1	C				C	C	C			C						
XC4008E	-4	C I						C I	C I		C I						
	-3	C I						C I	C I		C I						
	-2	C I						C I	C I		C I						
	-1	C						C	C		C						
XC4010E	-4	C I						C I	C I	C I	C I		C I				
	-3	C I						C I	C I	C I	C I		C I				
	-2	C I						C I	C I	C I	C I		C I				
	-1	C						C	C	C	C		C				
XC4013E	-4							C I		C I	C I	C I	C I	C I	C I		
	-3							C I		C I	C I	C I	C I	C I	C I		
	-2							C I		C I	C I	C I	C I	C I	C I		
	-1							C		C	C	C	C	C	C		
XC4020E	-4									C I		C I		C I			
	-3									C I		C I		C I			
	-2									C I		C I		C I			
	-1									C		C		C			
XC4025E	-4											C I		C I		C I	C I
	-3											C I		C I		C I	C I
	-2											C		C		C	C

1/29/99

C = Commercial T_J = 0° to +85°C

I= Industrial T_J = -40°C to +100°C

Table 26: Component Availability Chart for XC4000EX FPGAs

	PINS	208	240	299	304	352	411	432
	TYPE	High-Perf. QFP	High-Perf. QFP	Ceram. PGA	High-Perf. QFP	Plast. BGA	Ceram. PGA	Plast. BGA
	CODE	HQ208	HQ240	PG299	HQ304	BG352	PG411	BG432
XC4028EX	-4	C I	C I	C I	C I	C I		
	-3	C I	C I	C I	C I	C I		
	-2	C	C	C	C	C		
XC4036EX	-4		C I		C I	C I	C I	C I
	-3		C I		C I	C I	C I	C I
	-2		C		C	C	C	C

1/29/99

C = Commercial T_J = 0° to +85°C

I= Industrial T_J = -40°C to +100°C

User I/O Per Package

Table 27, Table 28, and Table 29 show the number of user I/Os available in each package for XC4000-Series devices. Call your local sales office for the latest availability information, or see the Xilinx WEBLINX at http://www.xilinx.com for the latest revision of the specifications.

Table 27: User I/O Chart for XC4000XL FPGAs

Device	Max I/O	Maximum User Accessible I/O by Package Type																				
		PC84	PQ100	VQ100	TQ144	HT144	HQ160	PQ160	TQ176	HT176	HQ208	PQ208	HQ240	PQ240	BG256	PG299	HQ304	BG352	PG411	BG432	PG475	PG559	BG560
XC4002XL	64	61	64	64																			
XC4005XL	112	61	77	77	112			112				112											
XC4010XL	160	61	77		113			129	145			160			160								
XC4013XL	192					113		129		145		160		192	192								
XC4020XL	224					113		129		145		160		192	205								
XC4028XL	256						129				160		193		205	256	256	256					
XC4036XL	288						129				160		193				256	288	288	288			
XC4044XL	320						129				160		193				256	289	320	320			
XC4052XL	352												193				256		352	352			352
XC4062XL	384												193				256			352	384		384
XC4085XL	448																			352		448	448

1/29/99

Table 28: User I/O Chart for XC4000E FPGAs

Device	Max I/O	Maximum User Accessible I/O by Package Type															
		PC84	PQ100	VQ100	PG120	TQ144	PG156	PQ160	PG191	HQ208	PQ208	PG223	BG225	HQ240	PQ240	PG299	HQ304
XC4003E	80	61	77	77	80												
XC4005E	112	61	77			112	112	112			112						
XC4006E	128	61				113	125	128			128						
XC4008E	144	61						129	144		144						
XC4010E	160	61						129	160	160	160		160				
XC4013E	192							129		160	160	192	192	192	192		
XC4020E	224									160		192		193			
XC4025E	256											192		193		256	256

1/29/99

Table 29: User I/O Chart for XC4000EX FPGAs

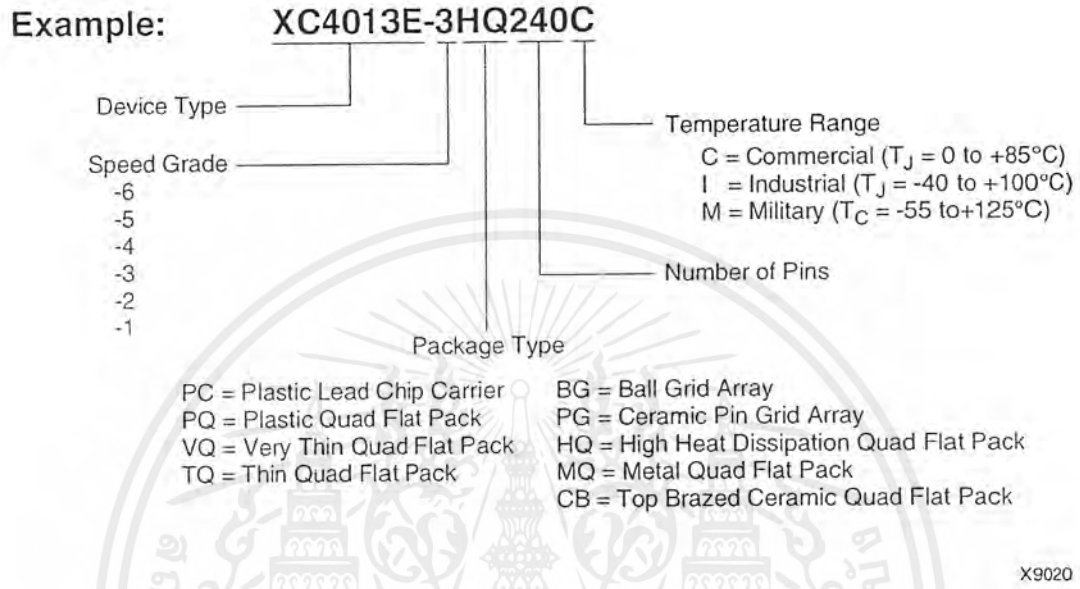
Device	Max I/O	Maximum User Accessible I/O by Package Type						
		HQ208	HQ240	PG299	HQ304	BG352	PG411	BG432
XC4028EX	256	160	193	256	256	256		
XC4036EX	288		193		256	288	288	288

1/29/99

XC4000 Series Electrical Characteristics and Device-Specific Pinout Table

For the latest Electrical Characteristics and package/pinout information for each XC4000 Family, see the Xilinx web site at http://www.xilinx.com/partinfo/databook.htm#xc4000

Ordering Information



Revision Control

Version	Description
3/30/98 (1.5)	Updated XC4000XL timing and added XC4002XL
1/29/99 (1.5)	Updated pin diagrams
5/14/99 (1.6)	Replaced Electrical Specification and pinout pages for E, EX, and XL families with separate updates and added URL link for electrical specifications/pinouts for WebLINX users