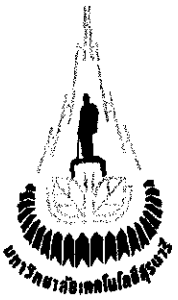
ต้นแบบเครื่องรับส่งข้อมูลด้วยเทคโนโลยี OFDM บนบอร์ด FPGA
Prototype of OFDM Technology on FPGA Board

โดย

นายศักดินนท์	ตรวิชุมณฑ	รหัสนักศึกษา	04704454
นายชัยณพุฒ	จงสงวน	รหัสนักศึกษา	04707301
นายกฤตวัฒ์	ปานทรัพย์	รหัสนักศึกษา	04713401

รายงานนี้เป็นส่วนหนึ่งของวิชา 423496 โครงงานวิศวกรรมโทรคมนาคม
และวิชา 423497 โครงงานพิเศษทางวิศวกรรมโทรคมนาคม
ประจำภาคเรียนที่ 1 และ 2 ปีการศึกษา 2560
หลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมโทรคมนาคม หลักสูตรปรับปรุง พ.ศ.2555
สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี



ต้นแบบเครื่องรับส่งข้อมูลด้วยเทคโนโลยี OFDM บนบอร์ด FPGA

Prototype of OFDM Technology on FPGA Board

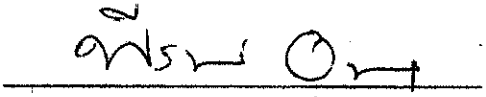
โดย

นายศักรินทร์	เชาว์ขุนทด	รหัสนักศึกษา	B4704454
นายชัยพฤกษ์	พลพวก	รหัสนักศึกษา	B4707301
นายอนุสรณ์	ปราศจาก	รหัสนักศึกษา	B4713401

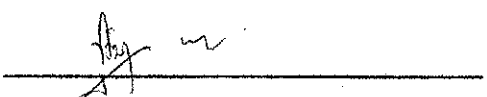
รายงานนี้เป็นส่วนหนึ่งของรายวิชา 427499 โครงงานวิศวกรรมโทรคมนาคม
และวิชา 427494 โครงงานศึกษาวิศวกรรมโทรคมนาคม
ประจำภาคการศึกษาที่ 1 และ 2 ปีการศึกษา 2550
หลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมโทรคมนาคม หลักสูตรปรับปรุง พ.ศ.2545
สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

ต้นแบบเครื่องรับ-ส่งข้อมูลด้วยเทคโนโลยี OFDM บนบอร์ดประมวลผลสัญญาณเชิงดิจิตอล

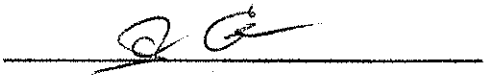
คณะกรรมการสอบโครงงาน



(อาจารย์ ดร. พีระพงษ์ อุฑารสกุล)
อาจารย์ที่ปรึกษาโครงงาน



(ผู้ช่วยศาสตราจารย์ ดร. ชุติมา พรหมมาก)
กรรมการ



(ผู้ช่วยศาสตราจารย์ ร. อ.ดร. ประโยชน์ คำสวัสดิ์)
กรรมการ

มหาวิทยาลัยเทคโนโลยีสุรนารี อนุมัติให้นับรายงานโครงงานฉบับนี้ เป็นส่วนหนึ่งของการศึกษาระดับปริญญาตรี สาขาวิชาวิศวกรรมโทรคมนาคม วิชา 427 494 โครงงานศึกษาวิศวกรรมโทรคมนาคม และวิชา 427 499 โครงงานวิศวกรรมโทรคมนาคม ประจำปีการศึกษา 2550

โครงงาน	ต้นแบบเครื่องรับส่งข้อมูลด้วยเทคโนโลยี OFDMบนบอร์ด FPGA
โดย	นายศักรินทร์ เชาว์ขุนทด
	นายชัยพฤกษ์ พลพวก
	นายอนุสรณ์ ปราศจาก
อาจารย์ที่ปรึกษา	อาจารย์ ดร.พีระพงษ์ อุฑารสกุล
ภาคการศึกษา	1,2/2550

บทคัดย่อ

โครงงานนี้ได้นำเสนอการจำลองการทำต้นแบบเครื่องรับส่งข้อมูลด้วยเทคโนโลยี OFDMบนบอร์ด FPGA โดยเทคโนโลยี OFDM ถูกสร้างขึ้นมาเพื่อใช้งานสำหรับระบบสื่อสารไร้สายแบบเคลื่อนที่แบนด์กว้าง (Broad band) มีอัตราการส่งข้อมูลสูงๆ เนื่องจากการสื่อสารแบบไร้สายมักจะประสบกับปัญหาการจางหายของสัญญาณ (Fading) อันมีสาเหตุมาจากการแพร่กระจายคลื่นสัญญาณเป็นหลายวิถี ซึ่งถ้าเป็นระบบการมอดูเลตและส่งสัญญาณแบบเก่าๆ (การส่งข้อมูลแบบอนุกรม) จะแก้ไขปัญหาเหล่านี้ได้ค่อนข้างยาก โดยเฉพาะเมื่อเป็นการสื่อสารที่มีอัตราการส่งข้อมูลสูงๆ แต่สำหรับ OFDM แล้วปัญหาเหล่านี้สามารถแก้ไขได้ง่ายกว่าและยังสามารถส่งข้อมูลที่มีอัตราสูงๆ ได้อย่างสบาย เนื่องจากในระบบ OFDM ข้อมูลที่เป็นอนุกรมความเร็วสูงจะถูกแปลงให้เป็นข้อมูลแบบขนานความเร็วต่ำเสียก่อน แล้วจึงส่งออกไปยังเครื่องรับพร้อมๆ กัน ซึ่งสามารถลดปัญหาที่ได้กล่าวไปแล้วนั้นลงได้ โดยโครงการนี้จะจำลองการทำงานของเทคโนโลยี OFDM ลงบนบอร์ดประมวลผลสัญญาณเชิงดิจิตอล(FPGA BOARD) แล้วนำผลที่ได้มาทดสอบ ซึ่งการศึกษานี้เป็นเพียงการศึกษาการทำต้นแบบของเทคโนโลยี OFDM เท่านั้น ผู้สนใจสามารถนำไปพัฒนาให้ดีขึ้นเพื่อนำไปประยุกต์ใช้กับระบบที่ใช้งานได้จริง หรือจะพัฒนาตัวโปรแกรมที่เขียนขึ้นให้มีความซับซ้อนขึ้นได้

กิตติกรรมประกาศ

โครงการนี้สำเร็จลุล่วงไปได้ด้วยดีส่งผลให้คณะผู้จัดทำโครงงานได้รับความรู้และประสบการณ์ต่างๆมากมายในระหว่างที่ศึกษาและจัดทำโครงงานนี้ คณะผู้จัดทำได้รับความช่วยเหลือ การให้คำปรึกษาความรู้ในด้านต่างๆ จากบุคคลหลายท่านซึ่งคอยให้คำแนะนำและความช่วยเหลือเป็นอย่างดีเสมอมาได้แก่

อาจารย์ ดร.พีระพงษ์ อุฑารสกุล สาขาวิชาวิศวกรรมโทรคมนาคม อาจารย์ที่ปรึกษาโครงงานที่คอยให้คำปรึกษาและคำแนะนำในด้านต่างๆ ทั้งยังคอยจัดหาทุนให้

สำนักงานกองทุนสนับสนุนการวิจัย ฝ่ายอุตสาหกรรม โครงการโครงงานอุตสาหกรรมสำหรับปริญญาตรี ประจำปี 2550 ที่ให้การสนับสนุนในเรื่องทุนการศึกษาในการทำโครงงานนี้

ห้างหุ้นส่วนจำกัด ภูมินทร์ซอฟต์ ที่ให้การสนับสนุนเรื่องอุปกรณ์อิเล็กทรอนิกส์ บอร์ดอุปกรณ์เชื่อมต่อ และคำแนะนำดีๆในการเขียนโปรแกรม

สุดท้ายนี้คณะผู้จัดทำใคร่ขอขอบพระคุณบุคคลทุกๆท่านที่ได้ให้โอกาส ให้การสนับสนุนและคอยเป็นกำลังใจให้กับคณะผู้จัดทำโครงงานด้วยดีตลอดมา

นายศักรินทร์ เชาว์ขุนทด
นายชัยพฤกษ์ พลพวก
นายอนุสรณ์ ปราศจาก
สาขาวิชาวิศวกรรมโทรคมนาคม
สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

สารบัญ

สารบัญ(ต่อ)

สารบัญรูปภาพ

สารบัญรูปภาพ(ต่อ)

สารบัญรูปภาพ(ต่อ)

เนื้อหา หน้าที่

บทที่ 1
บทนำ

1.1 ความเป็นมา

ในโลกของเทคโนโลยีการสื่อสารแบบไร้สายปัจจุบันนี้ หลายคนอาจจะคุ้นเคยและได้ยินคำว่า 2G (2^{nd} Generation) 3G (3^{rd} Generation) GSM(Global System for Mobile communications) และ CDMA(Code Division Multiple Access) เป็นส่วนใหญ่ แต่เมื่อความต้องการของผู้ใช้งานเริ่มแปรเปลี่ยนจากการสื่อสารที่เน้นเฉพาะเสียงพูด มาเป็นการสื่อสารที่เน้นข้อมูล ทั้งภาพและเสียงมากขึ้น ซึ่งจำเป็นต้องอาศัยการรับส่งข้อมูลด้วยความเร็วสูง ดังนั้น เทคโนโลยี OFDM จะเป็นเทคโนโลยีที่ช่วยเพิ่มความเร็วและปริมาณในการส่งให้ได้ดียิ่งขึ้น โครงงานนี้จึงพัฒนาต้นแบบเครื่องรับส่งข้อมูลด้วยเทคโนโลยี OFDM บนบอร์ดประมวลผลสัญญาณเชิงดิจิตอล

OFDM ย่อมาจาก Orthogonal Frequency Division Multiplex เป็นเทคนิคการมอดูเลชันแบบหลายคลื่นพาห์ (Multiple Carrier Modulation) ซึ่งเป็นรูปแบบของการสื่อสารแบบขนานนั่นเอง (ในอดีตส่วนใหญ่การสื่อสารเป็นแบบอนุกรม) OFDM ถูกสร้างขึ้นมาเพื่อใช้งานสำหรับระบบสื่อสารไร้สายแบบเคลื่อนที่แบนด์กว้าง (Broad band) มีอัตราการส่งข้อมูลสูงๆ เช่น ระบบ LAN แบบไร้สาย (Wireless LAN) ระบบอินเตอร์เน็ตความเร็วสูงแบบไร้สาย (Wireless high speed internet) ระบบกระจายเสียงแบบดิจิตอล (Digital Audio Broadcasting) หรือ DAB และระบบกระจายสัญญาณโทรทัศน์แบบดิจิตอล (Digital Television) หรือ DTVเป็นต้น

เนื่องจากการสื่อสารแบบไร้สายมักจะประสบกับปัญหาการจางหายของสัญญาณ (Fading) อันมีสาเหตุมาจากการแพร่กระจายคลื่นสัญญาณเป็นหลายวิถี (Multipath propagation) นอกจากนั้นคลื่นหลายวิถียังทำให้เกิดการรบกวนแบบแทรกสอดระหว่างสัญลักษณ์ (Inter Symbol Interference) หรือ ISI ของสัญญาณข้อมูลขึ้นที่เครื่องรับอีกด้วย ซึ่งถ้าเป็นระบบการมอดูเลตและส่งสัญญาณแบบเก่าๆ (การส่งข้อมูลแบบอนุกรม) จะแก้ไขปัญหาเหล่านี้ได้ค่อนข้างยาก โดยเฉพาะเมื่อเป็นการสื่อสารที่มีอัตราการส่งข้อมูลสูงๆ แต่สำหรับ OFDM แล้วปัญหาเหล่านี้สามารถแก้ไขได้ง่ายกว่าและยังสามารถส่งข้อมูลที่มีอัตราสูงๆ ได้อย่างสบาย เนื่องจากในระบบ OFDM ข้อมูลที่เป็นอนุกรมความเร็วสูงจะถูกแปลงให้เป็นข้อมูลแบบขนานความเร็วต่ำเสียก่อน แล้วจึงส่งออกไปยังเครื่องรับพร้อมๆ กัน ซึ่งสามารถลดปัญหาเหล่านี้ลงได้

จริงๆ แล้ว OFDM ไม่ใช่สิ่งใหม่อะไรเลยแต่เป็นเทคนิคที่มีการคิดค้น วิจัยและพัฒนามากกว่าห้าสิบปีมาแล้ว คือราวๆ ค.ศ. 1950 เริ่มต้น เทคนิค OFDM ถูกนำไปประยุกต์ใช้ในกิจการสื่อสารทางทหาร ต่อมาได้ถูกพัฒนามาเป็นระบบสื่อสารดิจิตอลความเร็วสูงแบบไร้สายเพื่อใช้งานในเชิงพาณิชย์ เช่น ระบบ HDSL (High bit rate Digital Subcriber Lines) ระบบ ADSL (Asymmetric Digital Subscriber Lines) ระบบ VHDSL (Very High Speed Digital Subcriber Lines) ระบบ DAB (Digital Audio Broadcasting) และระบบ DTV (Digital Television Broadcasting) เป็นต้น

เทคโนโลยีOFDMเป็นเทคโนโลยีที่แบ่งช่องสัญญาณออกเป็นหลายๆช่องย่อยตามความถี่ เช่นเดียวกับที่เราแบ่งช่องสัญญาณในเทคโนโลยี FDMA แต่คุณสมบัติที่เหนือกว่า FDMA คือ คลื่นพาห์

(Carrier) ของช่องสัญญาณใดๆ จะ Orthogonal กับคลื่นพาห์ของช่องสัญญาณอื่นๆ ที่เหลือทั้งหมด ถ้าจะแปลคำว่า Orthogonal จากทางคณิตศาสตร์ให้ง่ายขึ้นก็หมายถึงว่าแถบความถี่สเปกตรัมของแต่ละคลื่นพาห์จะมีค่าก็เฉพาะกับช่องสัญญาณของตัวเองเท่านั้น และจะถูกมองว่า ไม่มีค่าที่ช่องสัญญาณอื่นๆ ทั้งหมดในระบบ โดยจะอาศัยคุณสมบัติพิเศษข้อนี้ทำให้ไม่มีการรบกวนกันระหว่างคลื่นพาห์ ทำให้สเปกตรัมของคลื่นพาห์ สามารถทับซ้อนกันได้ ซึ่งจะนำไปสู่การใช้สเปกตรัมอย่างมีประสิทธิภาพ

จากคุณสมบัติและเทคนิคต่างๆ ที่ได้กล่าวมาข้างต้น ดังนั้น เราจึงทำการศึกษาเทคโนโลยี OFDM และจำลองการทำงานของเทคโนโลยีนี้ลงใน DSP Board โดยจะจำลองการทำงานในรูปแบบของการรับและการส่ง โดยใช้บอร์ดหนึ่งเป็นภาครับ และอีกบอร์ดหนึ่งใช้เป็นภาคส่ง โดยจะใช้คอมพิวเตอร์ในการเขียนคำสั่ง

1.2 วัตถุประสงค์ของโครงงาน

1.เพื่อรวบรวมความรู้ที่ได้จากการศึกษาภาคทฤษฎี มาใช้ในการทำงานได้จริง

2.ศึกษาการทำงานของโปรแกรม Xilinx เพื่อใช้แสดงผลลงบนระบบปฏิบัติการวินโดว์และใช้ในการคำนวณสมการทางคณิตศาสตร์

3.เพื่อศึกษาและจำลองการรับและส่งข้อมูลของเทคโนโลยี OFDM

4.ศึกษาและใช้การเขียนโปรแกรมในภาษา VHDL และการใช้ภาษาVHDL เบื้องต้น

1.3ขอบเขตการทำงาน

1.ศึกษาเทคโนโลยี OFDM เปรียบเทียบเทคโนโลยี OFDM กับเทคโนโลยีอื่นๆ

2.ศึกษาภาษาที่ใช้เขียน รวมถึงโปรแกรมที่ใช้ในการออกแบบ

3.ศึกษาบอร์ดประมวลผลที่ควรจะเลือกใช้

4.ออกแบบและเขียนโปรแกรมที่ใช้คำนวณค่า

5.ใช้บอร์ดแปลงสัญญาณดิจิตอลเป็นอนาล็อก (Digital to Analog Converter) กับบอร์ดอนาล็อกเป็นดิจิตอล(Analog to Digital Converter)ในการส่งค่า โดยจะจำลองการส่งค่าแบบ Modem

1.4ขั้นตอนการทำงาน

1.ศึกษาลักษณะต่างๆของเทคโนโลยี OFDM เบื้องต้น รวมถึงข้อดีข้อเสียของเทคโนโลยีนี้

2.ศึกษาสมการทางคณิตศาสตร์ที่ใช้ในการคำนวณค่าของเทคโนโลยี OFDM เพื่อที่จะนำไปใช้ในการเขียนโปรแกรมต่อไป

3.ศึกษาการเขียนโปรแกรมโดยใช้ภาษา VHDL และทดลองเขียนโปรแกรมโดยใช้ภาษา VHDL อย่างง่าย

4.ออกแบบและทดสอบเขียนโปรแกรมโดยใช้ภาษา VHDLทั้งภาคส่งและภาครับ

5.สร้างโปรแกรมโดยใช้ภาษา VHDL และทดสอบการทำงานของโปรแกรมโดยใช้โปรแกรมที่ชื่อ ModelSim

6.นำโปรแกรมลงบอร์ดประมวลผล FPGA และนำบอร์ด FPGA ไปต่อกับบอร์ดแปลงสัญญาณดิจิตอลเป็นอนาล็อก (Digital to Analog Converter) กับบอร์ดอนาล็อกเป็นดิจิตอล(Analog to Digital Converter)

7.ทดสอบโปรแกรมทั้งหมดรวมกับบอร์ดแปลงสัญญาณดิจิตอลเป็นอนาล็อก (Digital to Analog Converter) กับบอร์ดอนาล็อกเป็นดิจิตอล(Analog to Digital Converter) แล้วทดสอบผลที่ได้

8.แสดงผลที่ได้จากการออกแบบและการเขียนโปรแกรม

9.เขียนรายงาน

10.นำเสนอผลงาน

บทที่2
เทคโนโลยี OFDM

2.1 ความหมาย

OFDM ย่อมาจาก Orthogonal Frequency Division Multiplexing เทคโนโลยีOFDMเป็นเทคโนโลยีที่แบ่งช่องสัญญาณออกเป็นหลาย ๆช่องย่อยตามความถี่เช่นเดียวกับที่เราแบ่งช่องสัญญาณในเทคโนโลยี FDMA แต่คุณสมบัติที่เหนือกว่า FDMA คือ คลื่นพาห์ (Carrier) ของช่องสัญญาณใดๆ จะ Orthogonal กับคลื่นพาห์ของช่องสัญญาณอื่นๆ ที่เหลือทั้งหมด ถ้าจะแปลคำว่า Orthogonal จากทางคณิตศาสตร์ให้ง่ายขึ้นก็หมายถึงว่าแถบความถี่สเปกตรัมของแต่ละคลื่นพาห์จะมีค่าก็เฉพาะกับช่องสัญญาณของตัวเองเท่านั้น และจะถูกมองว่า ไม่มีค่าที่ช่องสัญญาณอื่นๆ ทั้งหมดในระบบ โดยจะอาศัยคุณสมบัติพิเศษข้อนี้ทำให้ไม่มีการรบกวนกันระหว่างคลื่นพาห์ ทำให้สเปกตรัมของคลื่นพาห์ สามารถทับซ้อนกันได้ ซึ่งจะนำไปสู่การใช้สเปกตรัมอย่างมีประสิทธิภาพ

2.2 การนำเทคโนโลยี OFDM ไปใช้งานใน Wimax

ไวแม็กซ์ได้รับการพัฒนาปรับปรุงเพื่อให้กลายเป็นรากฐานทางเทคนิคที่สำคัญของเครือข่ายข้อมูลเฉพาะที่ พกพาไปได้และไร้สาย ไวแม็กซ์คือการนำมาตรฐานใหม่ที่เป็น IEEE 802.16 มาใช้ มาตรฐานดังกล่าวนี้ใช้ Orthogonal Frequency Division Multiplexing (OFDM) ในการทำให้การบริการข้อมูล

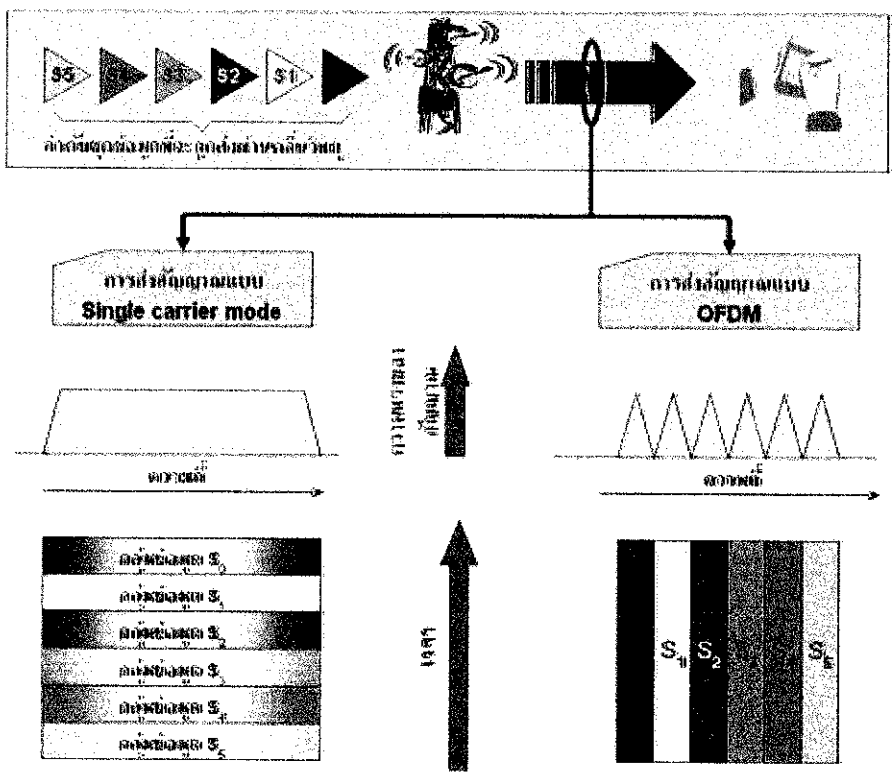
ไร้สายให้มีความสามารถสูงสุด หลักการของเทคโนโลยี OFDM นี้ก็คือการนำคลื่นความถี่วิทยุเล็กๆ (sub-carrier) มาใช้ให้เกิดประโยชน์สูงสุด โดยการนำคลื่นความถี่วิทยุเล็กๆ (ระดับ kHz) มาจัดสรรให้แก่ผู้ใช้ตามข้อกำหนดคลื่นความถี่วิทยุ การที่เราสามารถนำคลื่นความถี่วิทยุที่มีอยู่ทั้งหมดมาใช้ประโยชน์ได้อย่างมีประสิทธิภาพมากยิ่งขึ้นเช่นนี้ทำให้เครือข่าย OFDM มีประโยชน์มหาศาล และมีความเหมาะสมเป็นอย่างยิ่งสำหรับการเชื่อมต่อข้อมูลความเร็วสูงสำหรับผู้ใช้ทั้งที่อยู่ภายในสถานที่และนอกสถานที่ ปัจจุบันเครือข่ายไร้สาย Wireless Wide Area Network หรือ WWAN ที่อิงเทคโนโลยี OFDM ล้วนแต่เป็นมาตรฐานชนิด IEEE 802.16 ทั้งสิ้น

ผู้ให้บริการจะใช้ไวแม็กซ์บนความถี่ทั้งที่ได้รับอนุญาตและไม่ได้รับอนุญาต เทคโนโลยีนี้ช่วยให้เราสามารถสื่อสารแบบไร้สายได้ในระยะทางไกลๆ ด้วยความเร็วสูงสุดถึง 75 เมกะบิตต่อวินาทีในทางทฤษฎี (ความเร็วปกติจะช้ากว่านี้ขึ้นอยู่กับว่าผู้ให้บริการปรับแต่งสถานีฐานสำหรับการใช้เสปคตรัมของคลื่นความถี่วิทยุอย่างไร) แวนไร้สายที่ใช้เทคโนโลยีไวแม็กซ์ครอบคลุมพื้นที่ได้กว้างขวางมากกว่าเครือข่ายไร้สายเฉพาะที่หรือ Wireless Local Area Networks (WLAN) โดยจะสามารถเชื่อมต่อการสื่อสารไร้สายข้ามไปมาระหว่างอาคารต่างๆ แม้ว่าจะอยู่ห่างไกลกันเป็นบริเวณกว้างได้อย่างมีประสิทธิภาพ นอกจากนี้ เรายังสามารถนำไวแม็กซ์มาใช้

กับแอพพลิเคชั่นต่างๆ ได้จำนวนมากรวมถึงการเชื่อมต่อบรอดแบนด์ในระยะใกลเป็นไมล์ๆ แบบฮอตสปอตและระบบเซลลูลาร์ และการเชื่อมต่อด้วยความเร็วสูงสำหรับธุรกิจ

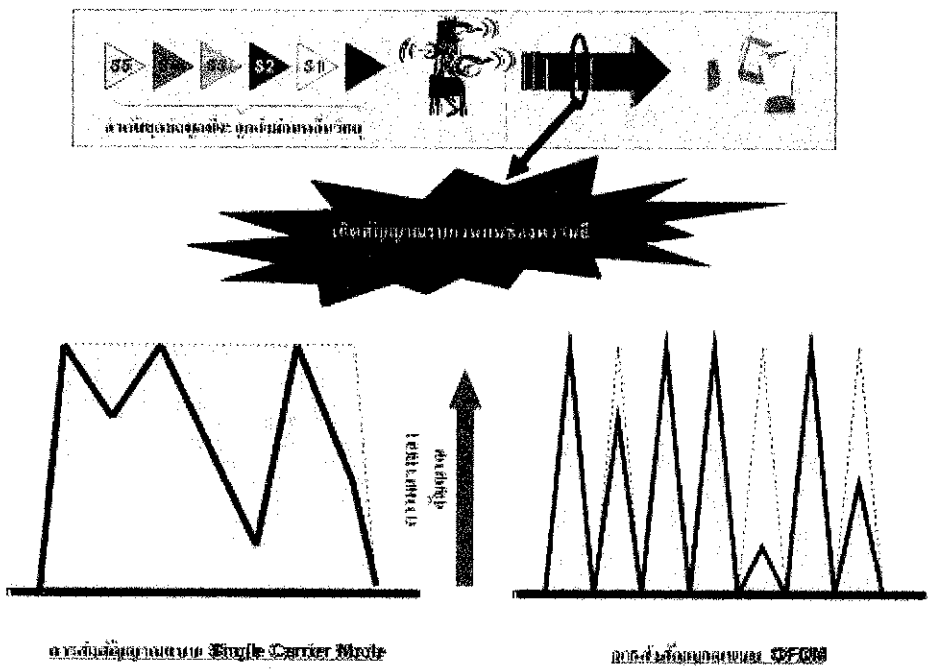
2.3 ข้อแตกต่างระหว่างเทคโนโลยี OFDM กับเทคโนโลยีอื่น ๆ

เทคโนโลยี OFDM แม้จะเป็นหนึ่งในมาตรฐานทางเทคนิคที่แตกแขนงออกมาจากเทคโนโลยี Spread Spectrum ก็ตาม แต่ก็ถือว่ามีการรับส่งข้อมูลแบบ Multiple Carrier Mode ซึ่งหมายถึงการแบ่งย่อยแถบความถี่ออกเป็นแถบย่อย ๆ สำหรับแยกส่งข้อมูลหลาย ๆ ช่อง แตกต่างจากมาตรฐาน Spread Spectrum ทั่วไปที่ใช้แถบความถี่เดียวสำหรับรับส่งข้อมูลช่องเดียว ความแตกต่างและข้อเด่นของเทคโนโลยี OFDM มีแสดงในรูปที่ 7 แนวคิดในการสื่อสารแบบ Spread Spectrum ตั้งอยู่บนพื้นฐานที่ว่าให้นำข้อมูลที่ต้องการส่งทั้งหมด ไม่ว่าจะมาจากที่แหล่งก็ตาม มาทำการมอดูเลตเข้ากับสัญญาณรบกวนเสมือน (Pseudo Noise) แล้วทำการส่งแบบให้กระจายไปทั่วในแถบความถี่สำหรับรับส่ง ตัวอย่างเช่น ในกรณีของเครือข่ายโทรศัพท์เคลื่อนที่ 3G มาตรฐาน W-CDMA ซึ่งมีการกำหนดแถบความถี่กว้าง 5 เมกะเฮิตรซ์สำหรับวงจรความถี่หนึ่งช่อง ข้อมูลของผู้ใช้งานทั้งหมดจะถูกนำมามอดูเลตกับสัญญาณรบกวนเสมือนแล้วกระจายส่งไปตลอดแถบความถี่ 5 เมกะเฮิตรซ์นั้น ๆ คล้ายกับการเป่าผงแป้งหลากสีบนฝ่ามือให้กระจายออกไปในวงกว้าง เครื่องรับปลายทางจะทราบเองว่าต้องแยกรับสัญญาณรบกวนเสมือนรหัสใด โดยจะพิจารณาจับข้อมูลที่ต้องการตลอดแถบความถี่ 5 เมกะเฮิตรซ์นั้น ดังแสดงในส่วนซ้ายมือของรูปที่ 3



รูปที่ 2.1 ความแตกต่างระหว่างการสื่อสารแบบ Spread Spectrum แบบ Single Carrier Mode กับ OFDM

สำหรับการรับส่งข้อมูลด้วยเทคโนโลยี OFDM จะมีข้อแตกต่างออกไป โดยก่อนส่ง จะทำการแบ่งแยกแถบความถี่ออกเป็นแถบความถี่ย่อย ๆ จากนั้นจึงนำข้อมูลที่ต้องการจะส่งมาทำการเรียงลำดับเป็นกลุ่มรหัสข้อมูล (Symbol) โดยเนื้อหาข้อมูลที่อยู่ภายในแต่ละกลุ่มรหัสข้อมูลนั้นไม่จำเป็นว่าจะต้องเป็นข้อมูลของผู้ใช้บริการรายเดียวกัน เปรียบเสมือนการดักน้ำที่ปลายท่อ ซึ่งต้นทางอาจมีการเก็บสะสมน้ำสีต่าง ๆ จากหลากหลายท่อย่อยที่เทรวมกันมา อธิบายให้ง่ายเข้าก็คือเป็นการเพิ่มเงื่อนไขในการทำงานขึ้นจากมาตรฐาน Spread Spectrum แทนที่จะส่งข้อมูลออกไปในแถบความถี่กว้าง ก็ให้นำข้อมูลมาจัดเป็นกลุ่มรหัสข้อมูลเสียก่อนนั่นเอง กลุ่มรหัสข้อมูลแต่ละกลุ่มจะถูกนำไปส่งออกอากาศโดยมีการกำหนดแบ่งแยกแถบความถี่ออกเป็นแถบย่อย ๆ มีจำนวนแถบเท่ากับกลุ่มรหัสข้อมูล ส่วนที่ว่าจะกำหนดให้มีกี่กลุ่มรหัสข้อมูลหรือแถบความถี่ย่อยนั้นก็แล้วแต่ข้อกำหนดของเทคโนโลยีนั้น ๆ จึงคล้ายกับว่ามีการตัดตอนข้อมูลออกเป็นกลุ่มย่อย ๆ แล้วให้แต่ละกลุ่มส่งขนานกันไปในเวลาเดียวกัน เพียงแต่อยู่ในแถบความถี่ย่อย ๆ ผิดกับมาตรฐาน Spread Spectrum ที่หากคิดแบบเดียวกับ OFDM ว่ามีการจัดกลุ่มรหัสข้อมูลขึ้นเหมือนกัน ก็จะเห็นราวกับว่ามีการส่งกลุ่มรหัสข้อมูลเรียงต่อกันไปตามเวลา มาตรฐาน OFDM จึงคล้ายกับเป็นการคิดนอกกรอบออกจากมาตรฐาน Spread Spectrum โดยมีการส่งข้อมูลในแนวขนานแทนที่จะเป็นการส่งต่อเป็นทอด ๆ หรือที่เรียกกันว่าเป็นอนุกรม



รูปที่ 2.2 ข้อดีของการรับส่งข้อมูลแบบ OFDM เปรียบเทียบกับการสื่อสารแบบ Spread Spectrum ชนิด Single Carrier Mode

ข้อดีของการรับส่งข้อมูลแบบ OFDM นั้นสามารถอธิบายให้เห็นได้ชัดเจนด้วยรูปที่ 9 ในกรณีที่เกิดการรบกวนทางความถี่ อันอาจสืบเนื่องมาจากปรากฏการณ์ทางธรรมชาติ เช่น ฟ้าร้อง ฟ้าผ่า หรือถูกรบกวนด้วยคลื่นความถี่วิทยุอื่น ๆ อันมีผลทำให้คุณสมบัติของช่องสื่อสารเกิดเปลี่ยนแปลงไปดังแสดงในด้านซ้ายของรูปที่ 9 การรับส่งข้อมูลแบบ Spread Spectrum Single Carrier Mode ก็จะเริ่มประสบปัญหาทันที สมมติว่า

เกิดการรบกวนในแถบความถี่ดังแสดงในรูปซ้าย ในช่วงเวลาเดียวกับที่มีการส่งกลุ่มรหัสข้อมูล S3 ก็จะมีผลทำให้ข้อมูลเกิดความผิดพลาด อุปกรณ์สื่อสารต้นทางและปลายทางจำเป็นต้องเริ่มทำกระบวนการกู้และแก้ไขข้อมูล (Error Collection) กว่าจะย้อนส่งข้อมูลในกลุ่มรหัส S3 ได้ทั้งหมด ก็ต้องทำให้เกิดความล่าช้าและเกิดภาวะคอขวดต่อรหัสข้อมูลในกลุ่มอื่น ๆ ที่ติดตามมา มองในแง่ของการใช้บริการก็คือ มีปัญหาช่องสื่อสารขัดข้องรับส่งข้อมูลได้ล่าช้าโดยไม่ทราบสาเหตุ

ในกรณีเดียวกัน หากเป็นการรับส่งข้อมูลแบบ OFDM ปัญหาการลดทอนของสัญญาณที่ปรากฏขึ้นจะกลายเป็นเพียงผลกระทบที่มีต่อรหัสข้อมูลเฉพาะกลุ่มเท่านั้น มิได้มีผลกระทบต่อช่องสัญญาณโดยรวม ซึ่งหากเป็นเพียงการทำให้ระดับสัญญาณของแถบความถี่ย่อยบางช่องลดลง ก็อาจไม่มีผลต่อการสื่อสารแต่อย่างใด เนื่องจากวงจรขยายสัญญาณของอุปกรณ์ภาครับอาจทำหน้าที่ปรับระดับความแรงของสัญญาณได้ หรือแม้จะเกิดการรบกวนจนทำให้ข้อมูลในกลุ่มรหัสข้อมูลผิดเพี้ยนไปจนต้องมีการแก้ไขโดยกระบวนการกู้และแก้ไขข้อมูล แต่ก็เป็นเพียงเหตุการณ์ที่เกิดขึ้นกับเฉพาะช่องสื่อสารที่เป็นของกลุ่มรหัสข้อมูลเฉพาะกลุ่มเท่านั้น มิได้ส่งผลกระทบต่อภาพรวมของการสื่อสารข้อมูล ผลที่เกิดขึ้นในทางปฏิบัติแม้จะสามารถสังเกตได้โดยผู้ใช้บริการ แต่แทบจะไม่ทำให้การสื่อสารผ่านเครือข่าย WiMAX เกิดความล่าช้าขึ้นแต่อย่างใด ยิ่งในภาวะปกติที่ไม่มีการถูกรบกวนอย่างรุนแรง ก็ต้องรับว่าเครือข่าย WiMAX ซึ่งใช้เทคโนโลยีการรับส่งข้อมูลแบบ OFDM ยังคงมีภูมิต้านทานต่อสัญญาณรบกวนทั่ว ๆ ไปเหนือกว่าเครือข่าย 3G ที่ใช้เทคโนโลยี Spread Spectrum อยู่มาก ส่งผลเกื้อหนุนให้รองรับการสื่อสารข้อมูลด้วยอัตราเร็วที่สูงกว่ามาก

2.4 การนำเทคโนโลยี OFDM ไปใช้งานจริงในมาตรฐานต่าง ๆ

2.4.1)มาตรฐานIEEE802.11b

มาตรฐานนี้เริ่มต้นใช้งานเมื่อประมาณปี 2542 เป็นมาตรฐานที่คนทั่วโลกนิยมใช้กันมากที่สุด โดยมาตรฐาน IEEE 802.11b สามารถรองรับการรับส่งข้อมูลได้ 11 Mbps และได้ใช้เทคโนโลยีแบบ CCK (Complimentary Code Keying) บวกกับ DSSS (Direct Sequence Spread Spectrum) ผ่านคลื่นความถี่วิทยุ 2.4 GHz เพื่อใช้งานในแบบสาธารณะ ซึ่งเทคโนโลยีแบบ IEEE 802.11b นี้เองเป็นจุดเริ่มต้นที่ก่อให้เกิดเครื่องหมายทางการค้า "WiFi" (Wireless Fidelity) ที่ถูกกำหนดขึ้นจากสมาคม WECA (Wireless Ethernet Compatability Alliance) ดังรูปที่ 1 กับ รูปที่ 2

รูปที่ 2.3 สัญลักษณ์ WiFi

รูปที่ 2.4 สัญลักษณ์ WiFi

2.4.2)มาตรฐาน IEEE 802.11a

มาตรฐาน IEEE 802.11 a ได้ถูกปรับปรุงให้ใช้คลื่นความถี่วิทยุ 5 GHz และปรับปรุงให้สามารถรองรับการรับส่งข้อมูลรวดเร็วมากขึ้นถึง 54 Mbps โดยใช้เทคโนโลยี OFDM (Orthogonal Frequency Division Multiplexing) สำหรับคลื่นความถี่ที่นำมาใช้กับ IEEE 802.11a นั้นจะเป็นคลื่นความถี่สาธารณะของประเทศสหรัฐอเมริกาเนื่องจากจะมีสัญญาณอื่นรบกวนน้อยกว่า 2.4 GHz ที่ใช้ในแบบ IEE 802.11 b แต่การกระจายสัญญาณของมาตรฐาน IEEE 802.11a จะกระจายสัญญาณได้สั้นกว่า.. มาตรฐานแบบ IEE 802.11a นี้จะมีความนิยมใช้กันน้อยในบางประเทศ อันเนื่องมาจากคลื่นความถี่ 5 GHz ได้ถูกจัดสรรเอาไว้ใช้งานประเภทอื่นแล้ว

2.4.3)มาตรฐาน IEEE 802.11g

มาตรฐาน IEEE 802.11 g ถูกสร้างขึ้นมาเพื่อให้สามารถใช้งานร่วมกับอุปกรณ์ที่เป็นมาตรฐานแบบ IEEE 802.11 b ร่วมกันได้ เราอาจจะเห็นอุปกรณ์ชนิด เช่น โน้ตบุค, PDA จะรองรับการใช้งานเครือข่ายไร้สายแบบ 802.11b/802.11g ในด้านการใช้ช่องสัญญาณจะใช้คลื่นความถี่ 2.4 GHz เหมือนกับ IEEE 802.11b แต่จะสามารถรับส่งข้อมูลได้รวดเร็วขึ้นกว่า คือ 54 Mbps สำหรับเทคโนโลยีที่นำมาใช้นั้นจะเป็นแบบ OFDM (Orthogonal Frequency Division Multiplexing)

2.4.4)มาตรฐาน IEEE 802.11e

มาตรฐาน IEEE 802.11e ถูกปรับปรุงให้สามารถรองรับการใช้งานตามหลักการ Qualitiy of Service สำหรับ application เกี่ยวกับมัลติมีเดีย (Multimedia) โดยการปรับปรุง MAC Layer ของ IEEE 802.11 เพิ่มเติม ส่งผลให้สามารถนำไปใช้กับอุปกรณ์ IEEE 802.11 WLAN ทุกเวอร์ชันได้

2.4.5)มาตรฐาน IEEE 802.11i

มาตรฐาน IEEE 802.11i ได้ถูกพัฒนาปรับปรุงให้มีความปลอดภัยทางด้านเครือข่ายมากขึ้น เนื่องจากเครือข่าย Wireless Lan มีช่องโหว่มากมาย เช่น ระบบการเข้ารหัสข้อมูลที่ไม่มีการเปลี่ยนแปลง Key ทำให้มาตรฐาน IEEE 802.11i มีความปลอดภัยมากขึ้นด้วยการแก้ไขปรับปรุง MAC Layer และใช้เทคโนโลยีขั้นสูงในการเข้ารหัสข้อมูลและมีการเปลี่ยนแปลงทุกครั้ง

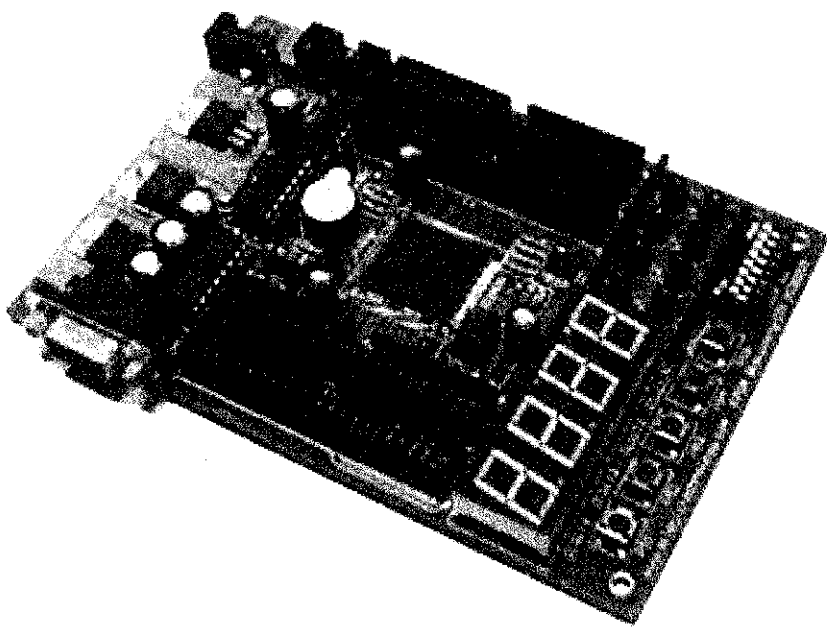
2.4.6)มาตรฐาน IEEE 802.11n

มาตรฐาน IEEE 802.11n เป็นมาตรฐานใหม่ของเครือข่ายไร้สาย Wireless Lan ซึ่งจะเริ่มพิจารณาโครงร่างในเร็ว ๆ นี้ โดยข้อมูลคร่าว ๆ สำหรับมาตรฐานใหม่ของเครือข่ายไร้สายนี้จะมีความเร็วในการรับส่งข้อมูลสูงถึง 320 Mbps และเพิ่มความแรงของสัญญาณให้สามารถส่งได้ไกลขึ้นมากกว่าเดิม

2.5บอร์ดที่ใช้ในการประมวลผล

โดยจะใช้บอร์ดประมวลผลสัญญาณเชิงดิจิตอลรุ่นFPGA Discovery-III XC3S200F4 **บอร์ดทดลอง** และ บอร์ดพัฒนาที่เหมาะสำหรับ LAB ดิจิตอลและออกแบบไอซี เพื่อใช้ออกแบบวงจรดิจิตอลขนาดใหญ่ เช่น ดิจิตอลฟิลเตอร์ ไมโครโพรเซสเซอร์ และ วงจรดิจิตอลที่มีความซับซ้อนและใช้งานที่ความถี่สูงได้เป็นอย่างดี โดยใช้ชิพของ Xilinx เบอร์ XC3S200 ที่มีขนาดความจุวงจร 400,000 เกตและ Platform Flash PROM ที่สามารถโปรแกรมซ้ำได้ถึง 20,000 ครั้งผ่านทางสายดาวน์โหลดได้โดยตรง

- SPARTAN-3 : XC3S400 (400,000 เกต)
- Platform Flash PROM เบอร์ XCF02S
- 7-Segment จำนวน 4 หลัก และ LED 8 ดวง (ใช้ร่วมกับExpansion ports)
- DIP Switch 8 บิต
- Push Button Switch 5 ตัว
- Expansion ports (80 Bits 3.3V I/O)
- I2C Socket และ RS-232C Port (ใช้ร่วมกับExpansion ports)
- Buzzer 1 ตัว(ใช้ร่วมกับExpansion ports)
- 25 MHz Onboard Oscillator (สามารถใช้ DCM ภายในสร้างเป็นความถี่อื่นๆได้ ดังนี้ คือ F_out = (M/D) x F_in โดยที่ M=2-32 , D=1-32)



รูปที่ 2.5 บอร์ดประมวลผลสัญญาณเชิงดิจิตอลรุ่น FPGA Discovery-III XC3S200F4

โดยบอร์ดทดลองอเนกประสงค์รุ่น FPGA Discovery-III XC3S200F4 มีรายละเอียดดังแสดงไปแล้วข้างต้น โดยจะมีความจุวงจรสูง 200,000 – 400,000 เกตและใช้ Platform Flash PROM สำหรับเก็บข้อมูลวงจร สามารถโปรแกรมวงจรลง Platform Flash PROM ผ่านทางสายดาวน์โหลดแบบ JTAG ได้โดยตรง บนบอร์ดมีอุปกรณ์อำนวยความสะดวกที่เพียบพร้อมด้วยอุปกรณ์อินพุตเอาต์พุตอย่างครบครัน จึงเหมาะสำหรับงานออกแบบวงจรดิจิตอล และออกแบบไอซีขั้นสูง งานพัฒนาออกแบบวงจรขนาดใหญ่ โดยจะมีส่วนประกอบอื่นๆดังนี้

◉7-Segment จำนวน 4 หลัก (ใช้ร่วมกับ Expansion ports และสามารถถอดได้)

◉LED จำนวน 8 ดวง (ใช้ร่วมกับ Expansion ports สามารถแยกออกจาก I/O ได้โดยหักเอา RNET3 และ RNET4 ออก)

◉buzzer จำนวน 1 ตัว (ใช้ร่วมกับ Expansion ports)

◉DIP Switch 8 บิต

◉push Botton Switch จำนวน 5 ตัว

◉expansion ports(80 Bits 3.3V.I/O)

◉RS-232C Port 1 Port (ใช้ร่วมกับ Expansion ports)

◉25Mhz Oscillator (สามารถโปรแกรมเป็นความถี่อื่นๆ ได้โดยใช้ Digital Frequency Synthesizer ที่มีอยู่ใน FPGA)

2.6 คุณสมบัติที่สำคัญของชิพ FPGA ที่สำคัญ

◉ความจุวงจร 200,000 เกต

◉18Kb block RAMs จำนวน 16 ชุด (รวม 288KB)

◉18x18 hardware multiplier จำนวน 16 ชุด

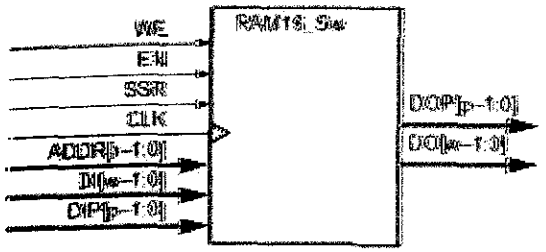
◉digital Clock Manager (DCM) จำนวน 4 ชุด

โดยอุปกรณ์ที่อยู่ภายในชิพมีคุณสมบัติดังนี้

2.6.1)หน่วยความจำ 18 Kb block RAM

18Kb block RAM เป็นหน่วยความจำที่มีความเร็วสูงมาก (โดยประมาณ) 200 Mhz ที่มีอยู่ในชิพสามารถฟอร์มให้เป็น RAM หรือ ROM ที่มีขนาดต่างๆกันได้รวมทั้งทำเป็น FIFO ได้ด้วย โดยรูปที่ 6 แสดงขนาด

RAM แบบ Single port ในรูปที่ 2.7 แสดงตัวอย่างแสดง RAM แบบ Single port ขนาดต่าง ๆ ที่สร้างจาก Block RAM แต่ละชุด



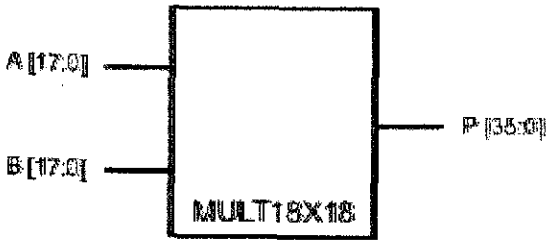
รูปที่ 2.6 แสดงขนาด RAM แบบ Single Port

Organization	Memory Depth	Data Width	Parity Width
512x36	512	32	4
1Kx18	1024	16	2
2Kx9	2048	8	1
4Kx4	4096	4	-
8Kx2	8192	2	-
16Kx1	16384	1	-

รูปที่2.7 RAM แบบ Single Port ขนาดต่างๆที่สร้างจาก Block RAM แต่ละชุด

2.6.2)18x18 Hardware multiplier

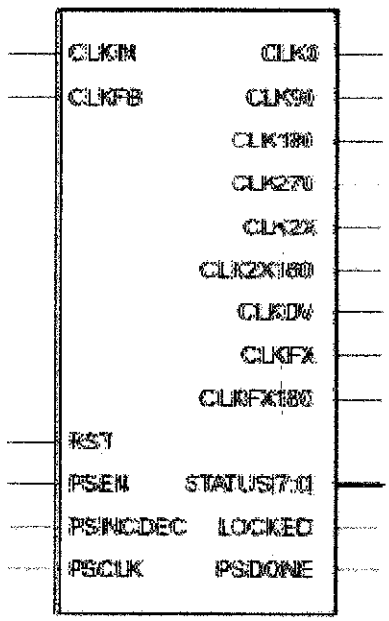
18x18 Hardware multiplier เป็นฮาร์ดแวร์ของวงจรคูณสำเร็จรูปขนาด 18x18 บิตที่มีความเร็วในการทำงานสูง ซึ่งมีสัญลักษณ์ดังรูปที่ 4 สำหรับคนที่ต้องการตัวคูณมาก ๆ อาจจะต้องใช้วิธีมัลติเพล็กซ์เข้าช่วย



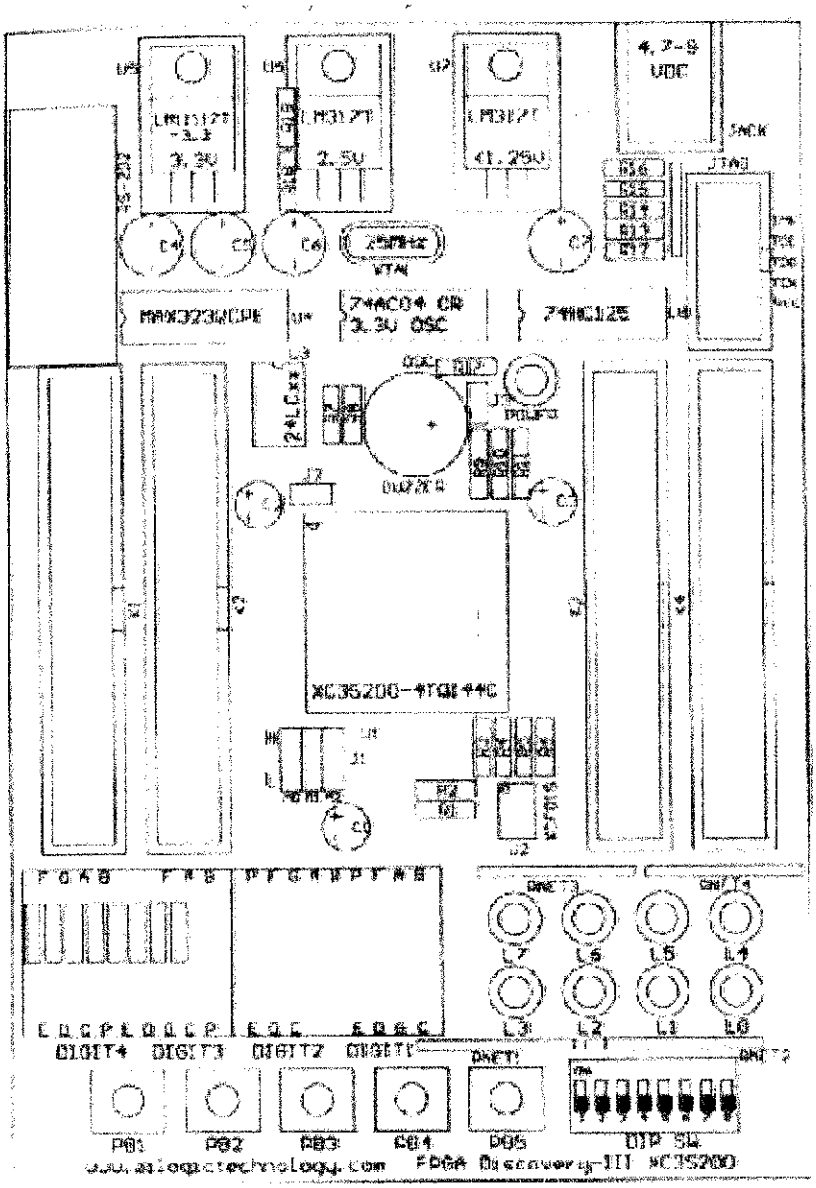
รูปที่ 8 สัญลักษณ์ของ 18x18 hardware multiplier

2.6.3)Digital Clock Manager

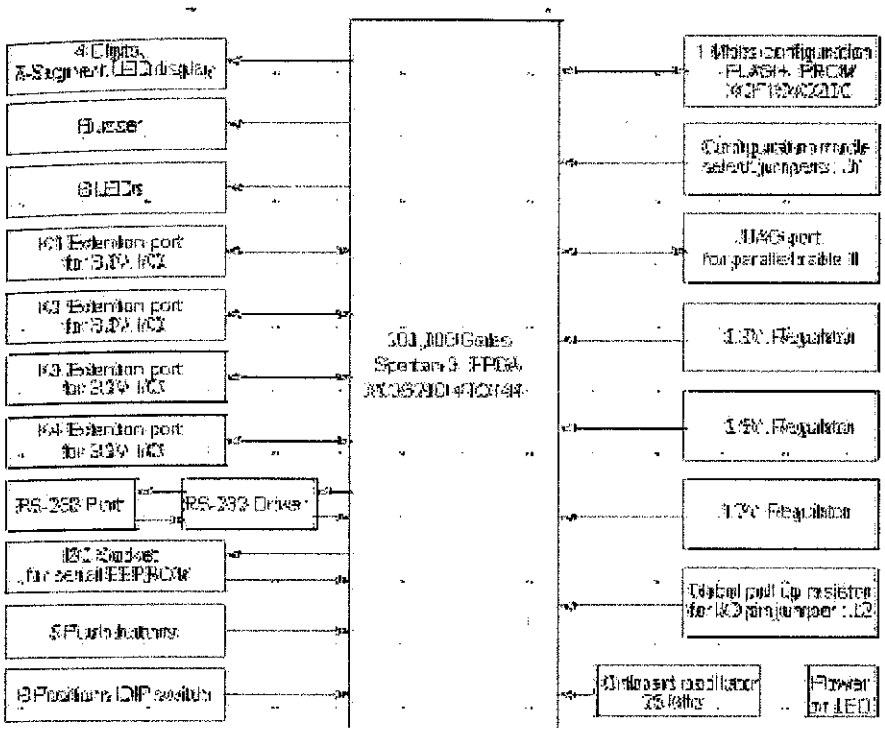
Digital Clock Manager (DCM) เป็นวงจรที่ช่วยจัดการเกี่ยวกับสัญญาณนาฬิกาและเลื่อนเฟส มีจำนวน 4 ชุด จึงทำให้การออกแบบวงจรง่ายขึ้นเนื่องจากสามารถสร้างความถี่ต่าง ๆได้จากออสซิเลเตอร์ภายนอกเพียงชุดเดียวและเอาต์พุตซิงค์โครไนซ์กับสัญญาณของออสซิลเลเตอร์เดิมอีกด้วย DCM มีสัญลักษณ์แสดงดังรูปที่ 8 ซึ่งจะทำหน้าที่ต่อไปนี้



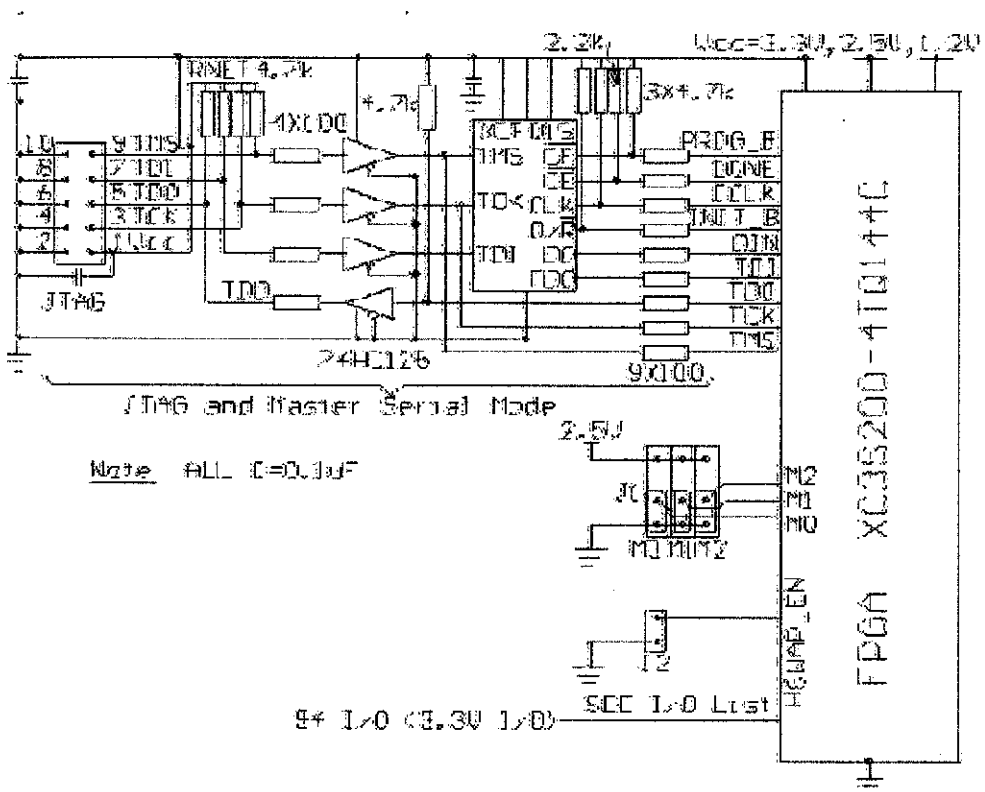
รูปที่2.9 สัญลักษณ์ของวงจร DCM



รูปที่2.10 การจัดวางตำแหน่งการวางอุปกรณ์ด้านบน



รูปที่ 2.11ผังส่วนประกอบของบอร์ด FPGA Discovery-III XC3S200F4



รูปที่ 2.12 การจัดวาง I/O ต่างๆ ของบอร์ดประมวลผล FPGA Discovery-III XC3S200F4

7-Segment	FPGA Pinout	Description
a	p48	a
b	p45	b
c	p42	c
d	p38	d
e	p37	e
f	p43	f
g	p44	g
dp	p39	Decimal Point
DIGIT1	p51	DIGIT1, COMMON CATHODE
DIGIT2	p53	DIGIT2, COMMON CATHODE
DIGIT3	p56	DIGIT3, COMMON CATHODE
DIGIT4	p41	DIGIT4, COMMON CATHODE

Push Button	FPGA Pinout	Description
PB1	p14	Push Button No.1
PB2	p46	Push Button No.2
PB3	p47	Push Button No.3
PB4	p55	Push Button No.4
PB5	p11	Push Button No.5

EEPROM	FPGA Pinout	Description
I2C-SCL	p128	24LCXX
I2C-SDA	p129	24LCXX

RS-232	FPGA Pinout	Description
TX	p181	RCL232SOP
RX	p182	RCL232SOP

LED	FPGA Pinout	Description
L0	p79	L0
L1	p77	L1
L2	p69	L2
L3	p76	L3
L4	p74	L4
L5	p70	L5
L6	p73	L6
L7	p71	L7

Dip SW	FPGA Pinout	Description
1	p52	Dip Switch No.1
2	p54	Dip Switch No.2
3	p55	Dip Switch No.3
4	p56	Dip Switch No.4
5	p59	Dip Switch No.5
6	p60	Dip Switch No.6
7	p61	Dip Switch No.7
8	p64	Dip Switch No.8

Oscillator	FPGA Pinout	Description
OSC	p127	25MHz, OSCILLATOR

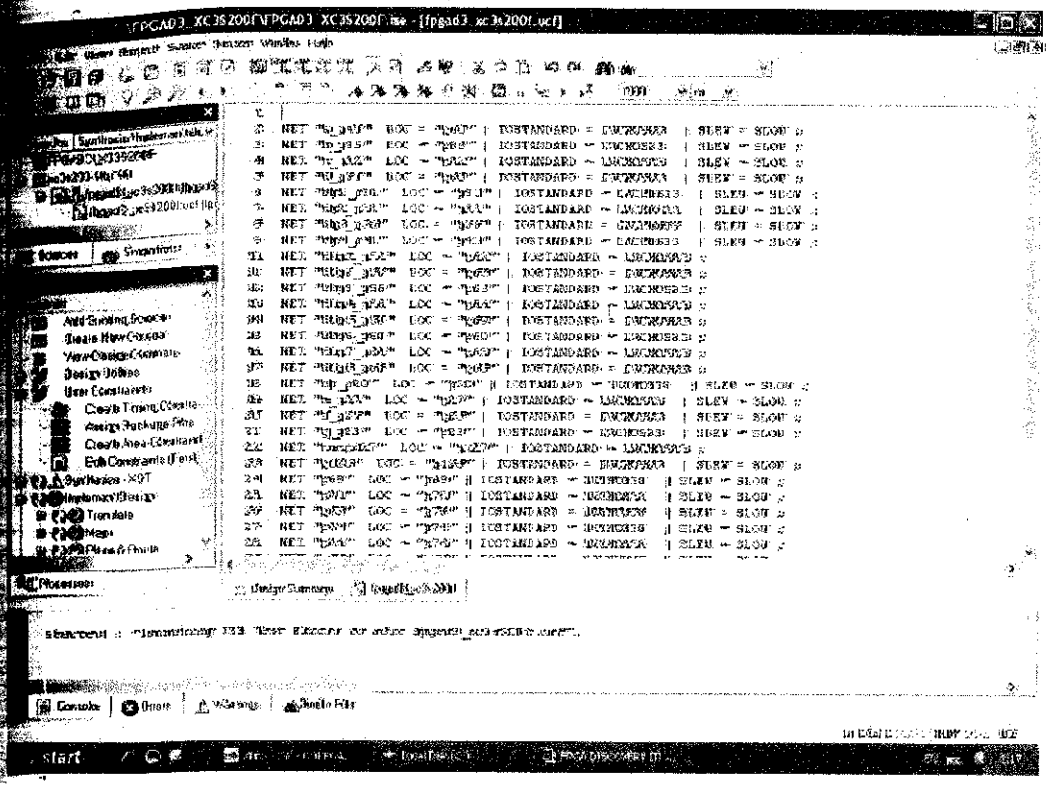
BUZZER	FPGA Pinout	Description
BUZZER	p125	BUZZER

รูปที่ 2.13 รายละเอียดของอุปกรณ์ที่ต่ออยู่กับขา FPGA () I/O list

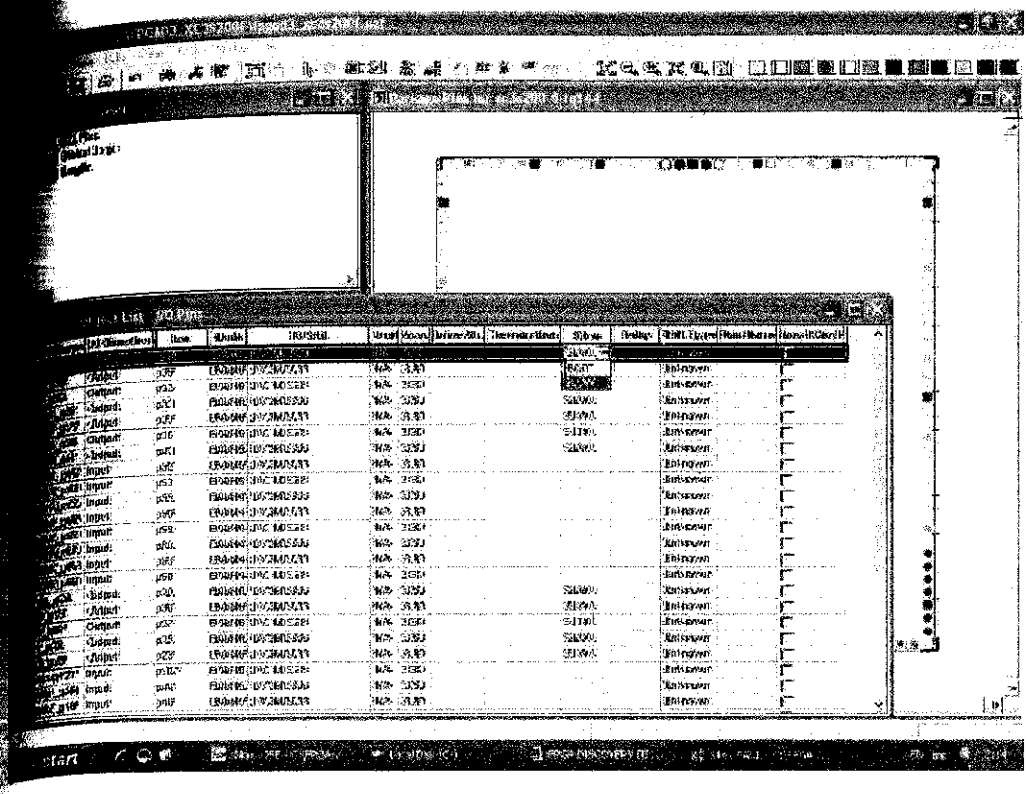
บทที่ 3

การใช้งานโปรแกรมและการโหลดโปรแกรมลงบอร์ด

3.1 การใช้งานโปรแกรม โปรแกรมที่ใช้ ชื่อ Xilinx เป็นโปรแกรมที่ใช้ภาษา VHDL ในการเขียนโปรแกรม สามารถเขียนในหลายรูปแบบเช่น



รูปที่ 3.1 การโปรแกรมเอาต์พุตของ FPGA เป็นแบบ LVTTL และเป็นแบบ Slow slew Rate ใน Edit Constraints(Text)

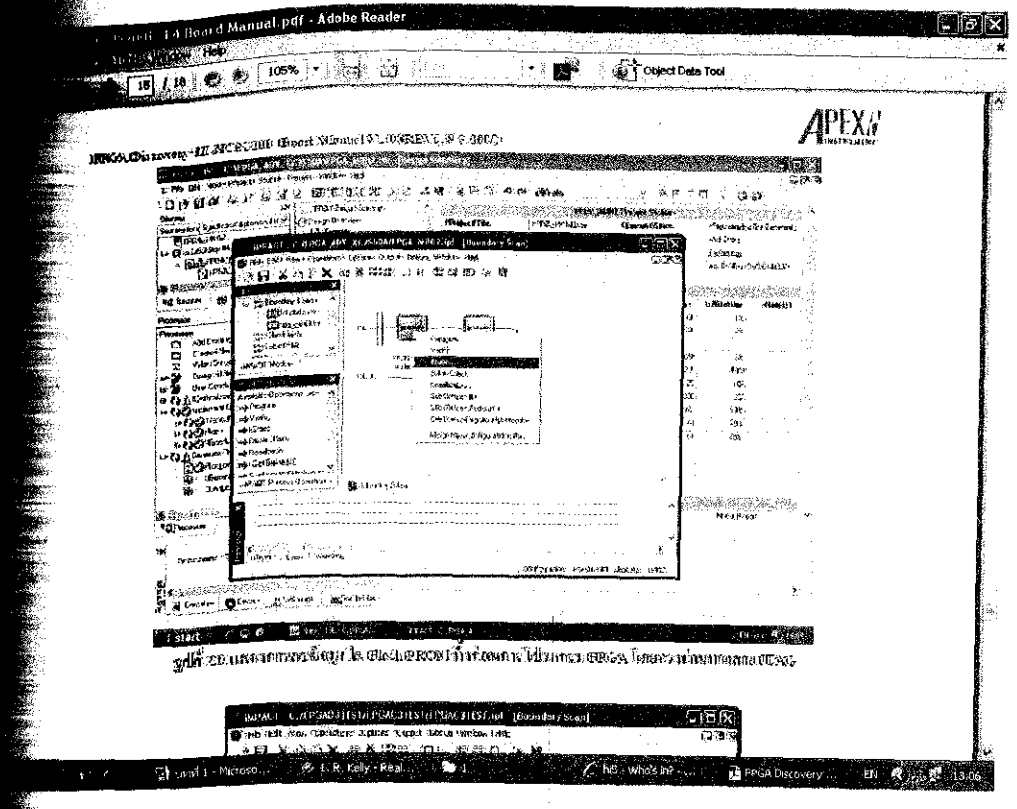


รูปที่ 3.2 การโปรแกรมเอาต์พุตของ FPGA เป็นแบบ LVTTL และเป็นแบบ Slow slew Rate ในหน้าต่าง

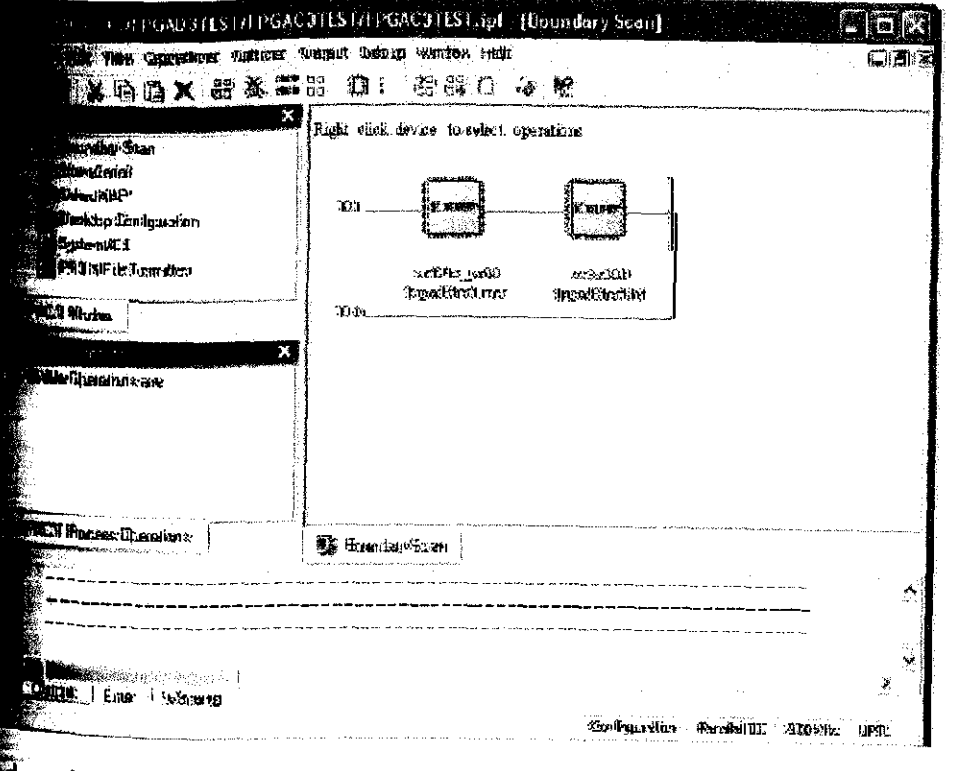
3.2 การโปรแกรมข้อมูลลงบอร์ด FPGA

การโปรแกรม FPGA โดยตรงผ่านทางสาย JTAG นั้นผู้ใช้จะต้องลบข้อมูลใน Flash PROM ทิ้งก่อนเสมอ ตัวอย่างแสดงดังรูป การไม่ลบข้อมูลวงจรออกจาก Flash PROM ก่อนอาจทำให้ข้อมูลที่โปรแกรมลง FPGA ไม่สมบูรณ์ (โดยไม่มีการเตือนว่าเกิดข้อผิดพลาด) เพราะ FPGA จะถูกเซตอยู่ใน Master Serial Mode จึงถูกโปรแกรมจาก Flash PROM อย่างอัตโนมัติเรียบร้อยแล้วทันทีที่เริ่มต้นจ่ายไฟเลี้ยง

การโปรแกรมวงจรลง FPGA นั้นเราจะต้องสร้างไฟล์พร้อมที่จะโปรแกรมลง Platform Flash PROM และ FPGA ก่อน จากนั้นแล้วทำการต่อสาย JTAG และต่อไฟเลี้ยงเข้าบอร์ดแล้วทำการดาวน์โหลดวงจรที่ต้องการลงสู่ Flash PROM และชิพ FPGA ตามลำดับ ซึ่งขั้นตอนการดาวน์โหลดที่จอคอมพิวเตอร์จะปรากฏชิพทั้ง 2 ตัวพร้อมกัน ดังรูป เพราะมีการออกแบบให้ในโหมด JTAG ต่อถึงกันวนแบบลูกโซ่เพื่อสะดวกเมื่อโปรแกรม เราจึงสามารถเลือกที่จะดาวน์โหลดข้อมูลลงชิพทั้งสองตัวหรือตัวใดตัวหนึ่งก็ได้ และเนื่องจาก Platform Flash PROM และ FPGA ต่อกันในโหมด Master serial (M0, M1 , M2=0) อีกด้วย ดังนั้นทุกครั้งที่เริ่มจ่ายไฟให้บอร์ดทดลอง FPGA จะดาวน์โหลดข้อมูลจาก Platform Flash PROM มาที่ FPGA อย่างอัตโนมัติ



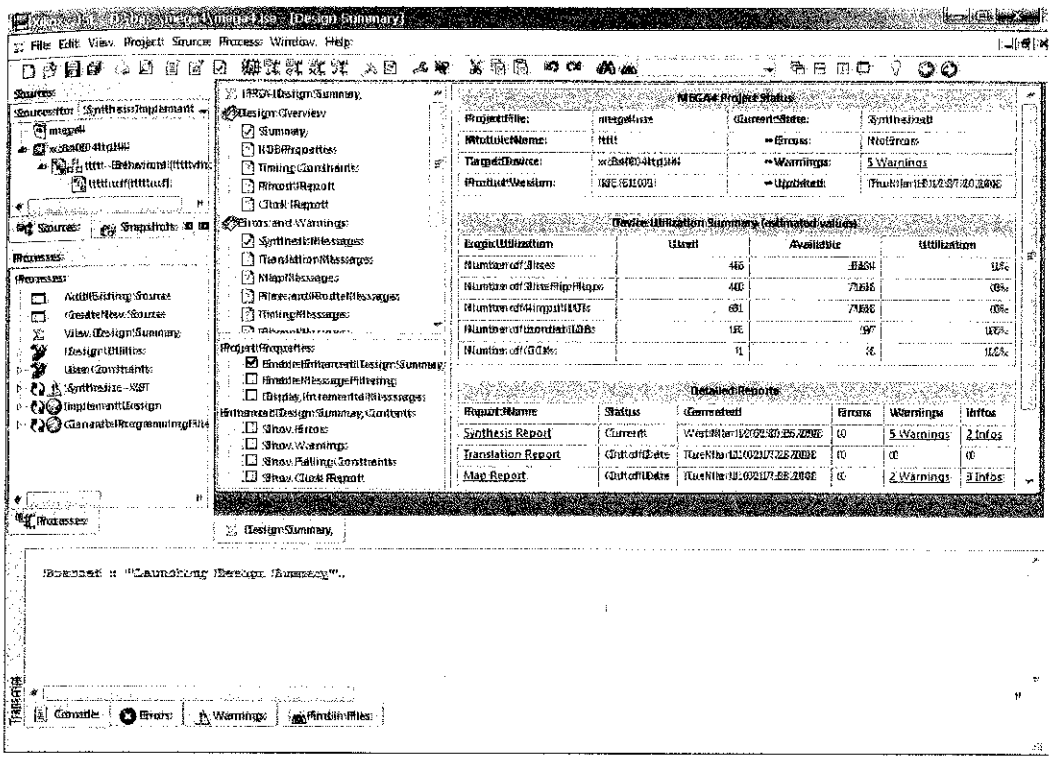
3 แสดงการลบข้อมูลใน FLASH PROM ทิ้งก่อนการโปรแกรม FPGA โดยตรงผ่านทางสาย JTAG



ที่ 3.4 ขั้นตอนการดาวน์โหลดที่จอคอมพิวเตอร์จะปรากฏชิพ Flash PROM และ FPGA พร้อมกัน

3.3 การเปิดโปรแกรมและใช้งานโปรแกรม

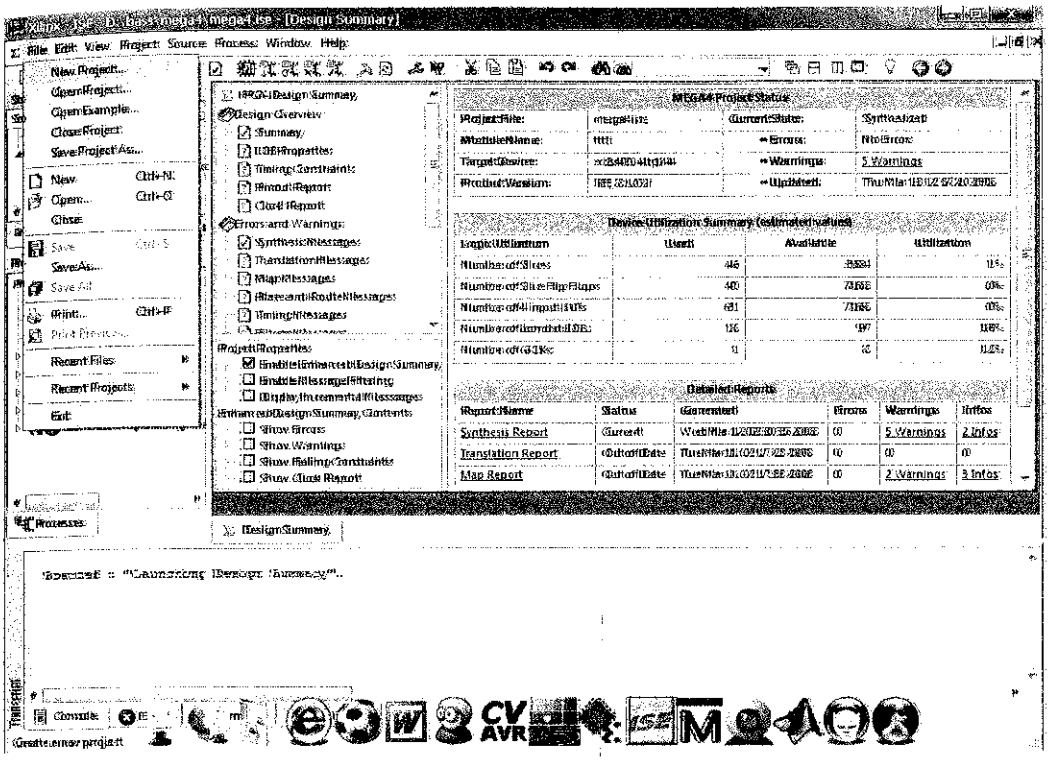
3.3.1)เปิดโปรแกรมขึ้นมา



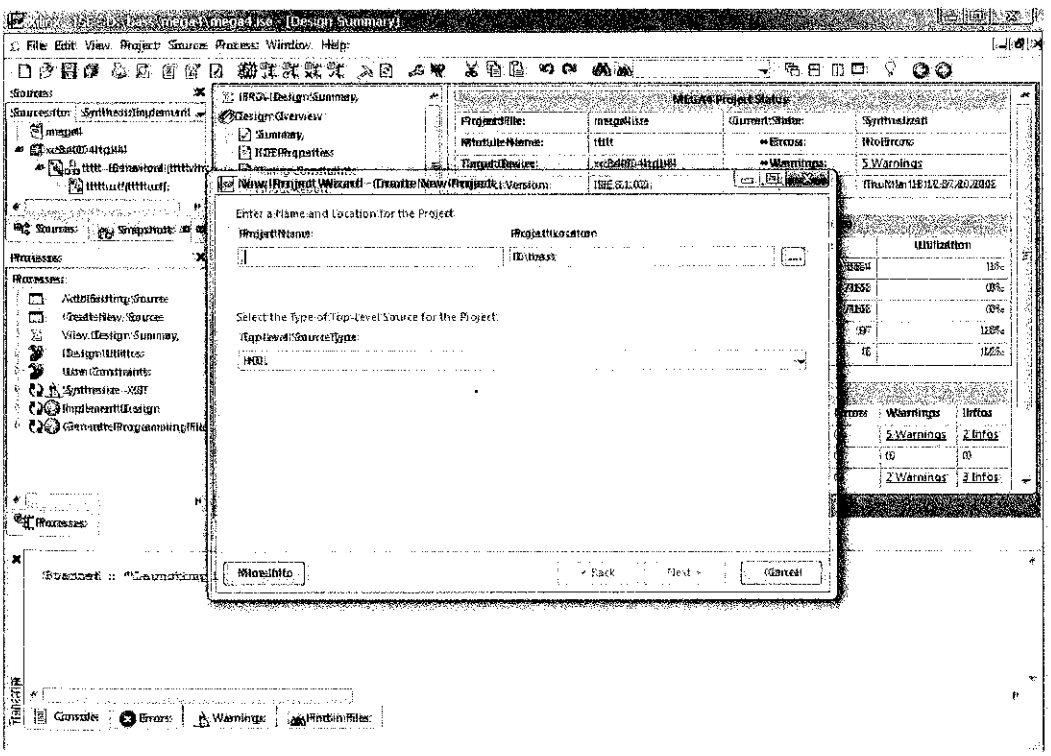
รูปที่ 3.5 หน้าจอที่เปิดโปรแกรมขึ้นมาครั้งแรก

3.3.2)เมื่อเราต้องการเปิดโปรเจคใหม่เราก้อเลือก

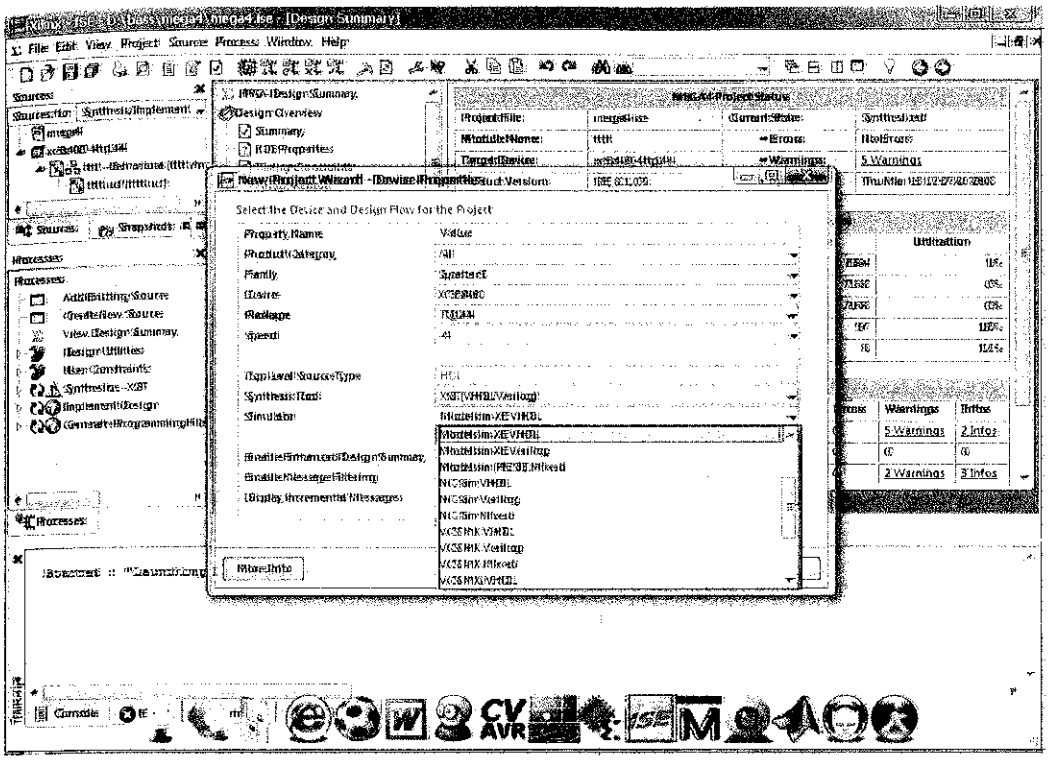
-File – New project



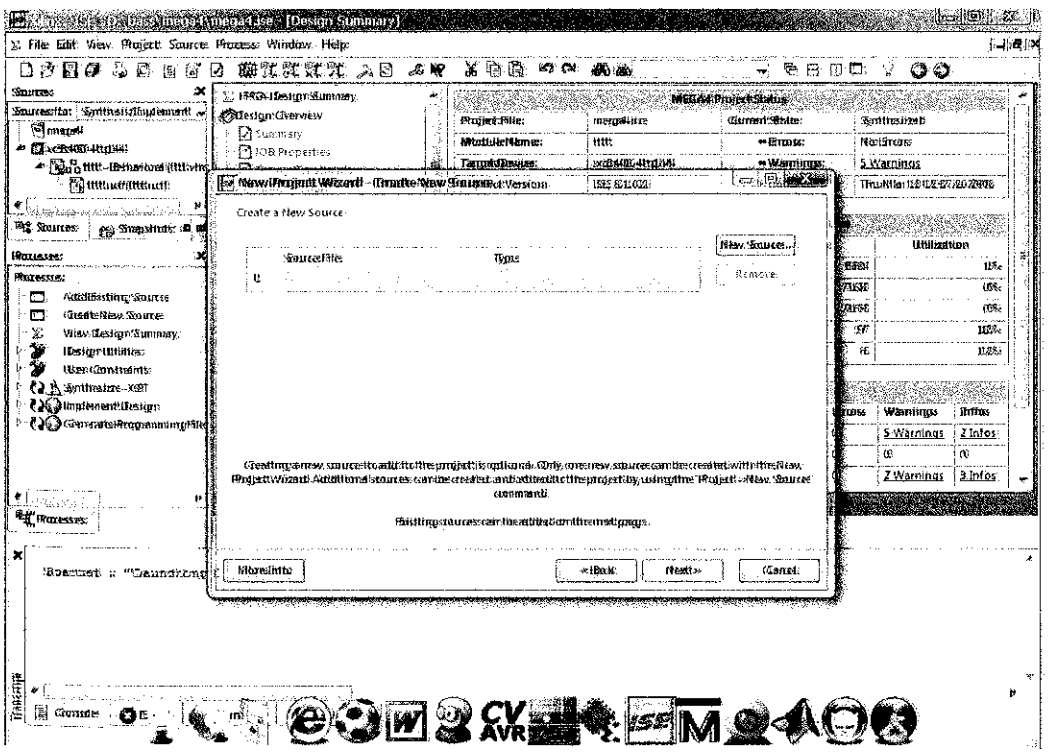
รูปที่ 3.6 เปิดโปรเจคขึ้นมาใหม่เพื่อสร้างงานใหม่



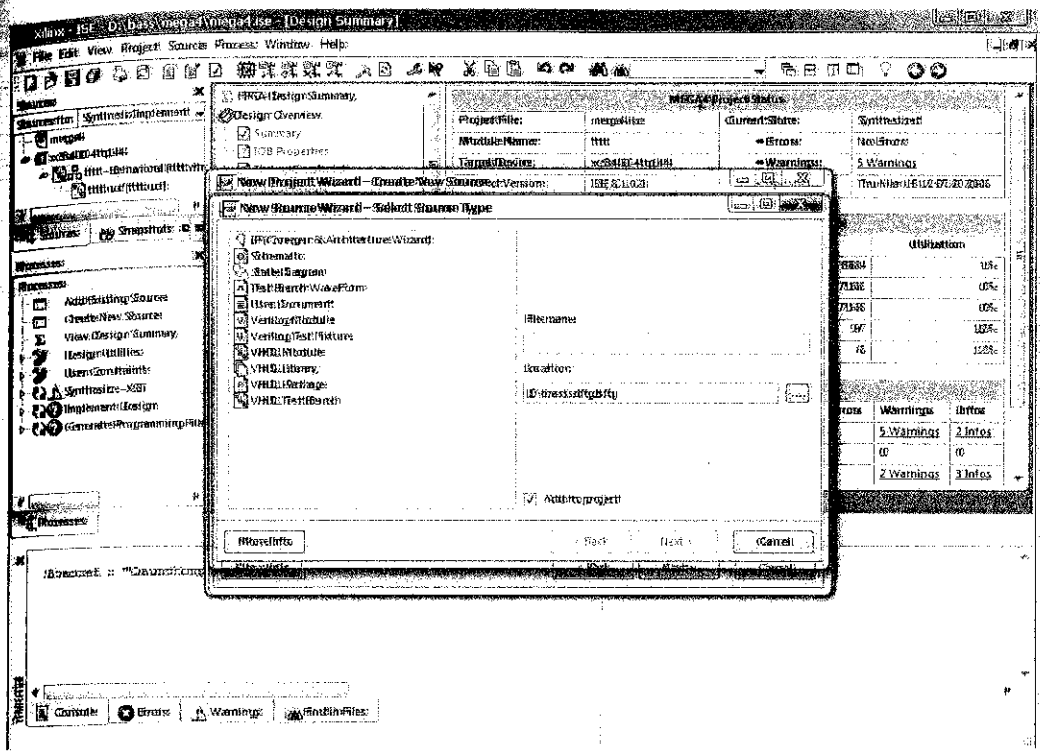
รูปที่3.7 ตั้งชื่อโครงงานและเลือกตำแหน่งจัดเก็บงาน



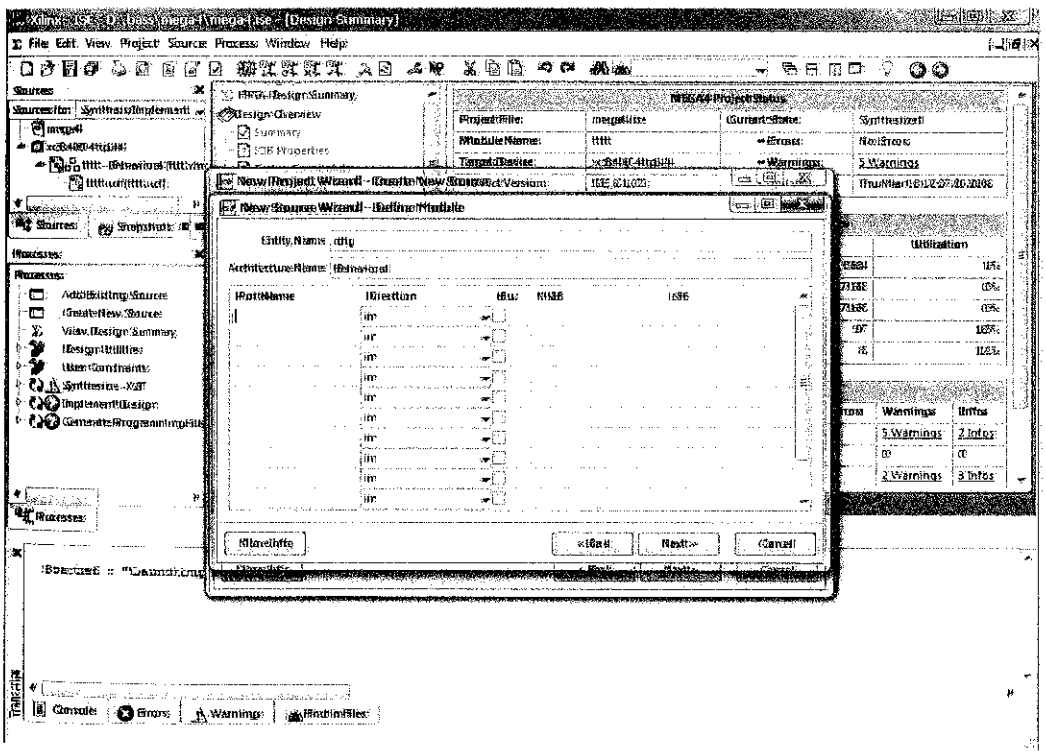
รูปที่ 3.8 เลือกว่าจะจำลองผลการทดสอบโปรแกรมเป็นแบบใด ในรูปเลือกใช้โปรแกรม ModelSim เป็นตัวทดสอบผล



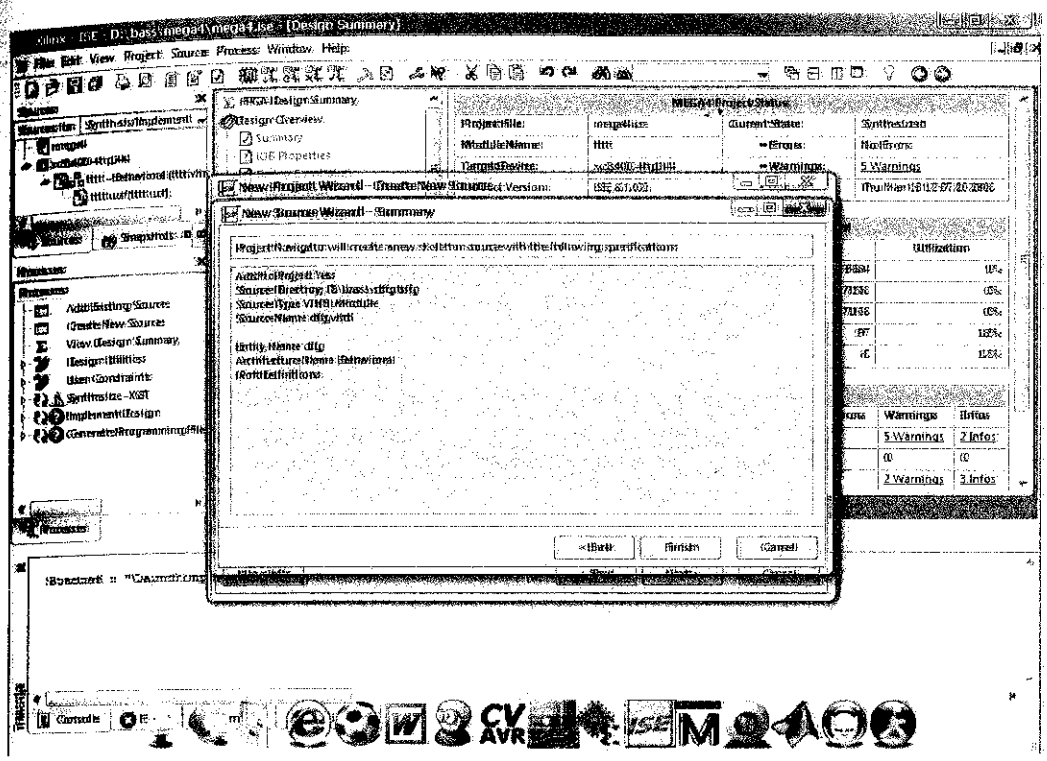
รูปที่ 3.9 เลือก Source ใหม่



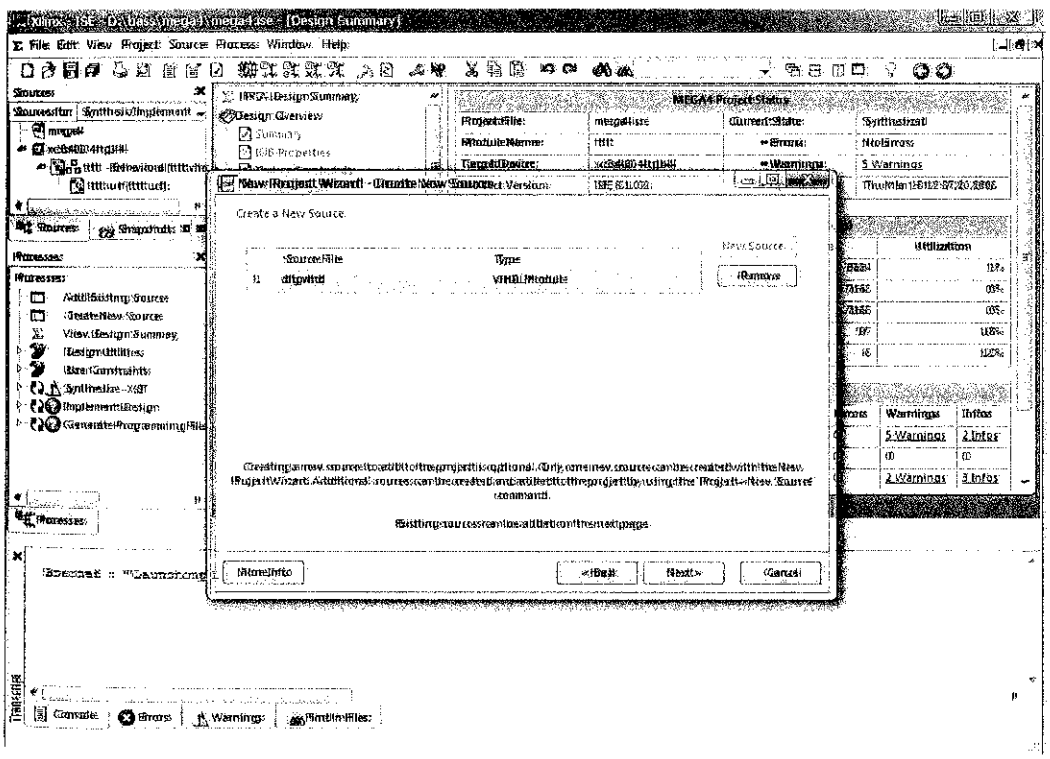
รูปที่ 3.10 เลือกประเภทของไฟล์ที่เราจะสร้างขึ้นมา ว่าจะใช้ภาษาใดเป็นภาษาที่ใช้เขียนโปรแกรม



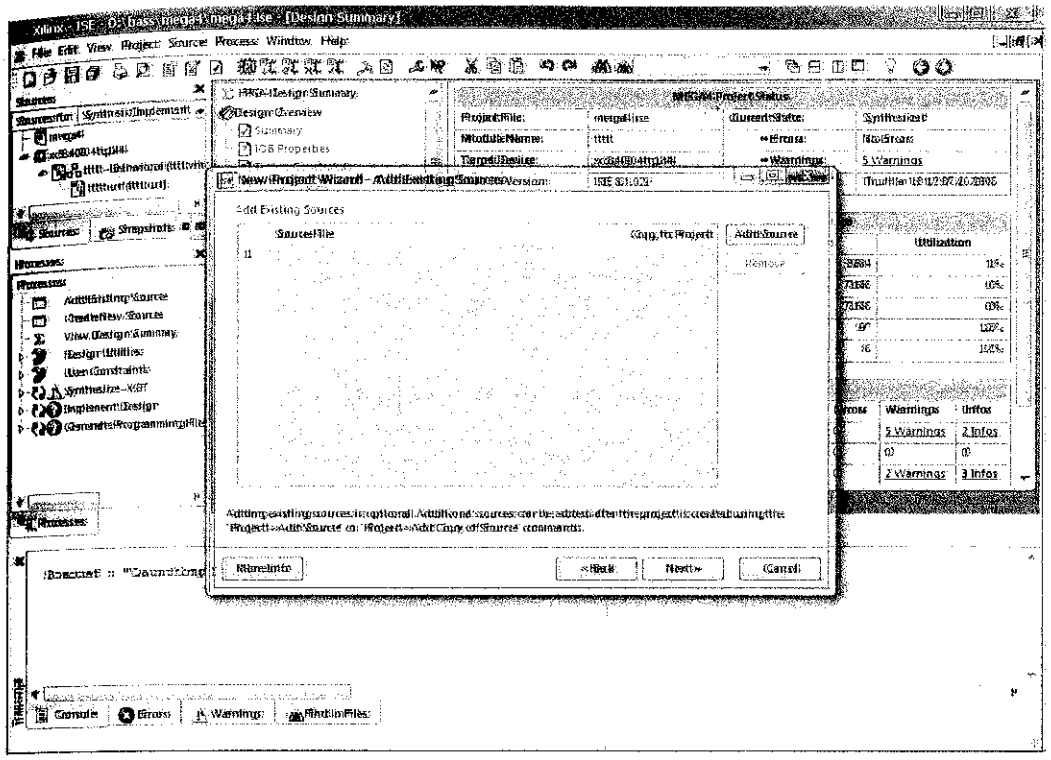
รูปที่ 3.11 เมื่อเลือกไฟล์เสร็จแล้ว จะเป็นการกำหนดพอร์ท หรือตัวแปรที่เราต้องการใช้



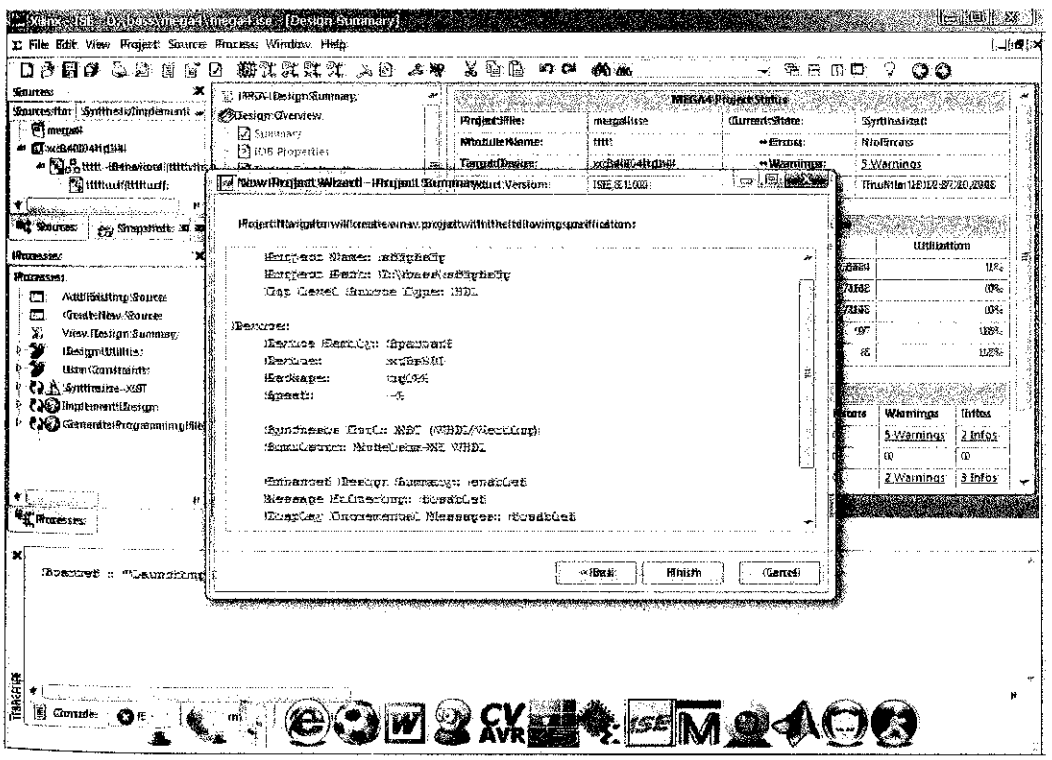
รูปที่ 3.12 เป็นการเสร็จขั้นตอนของการเลือก Source



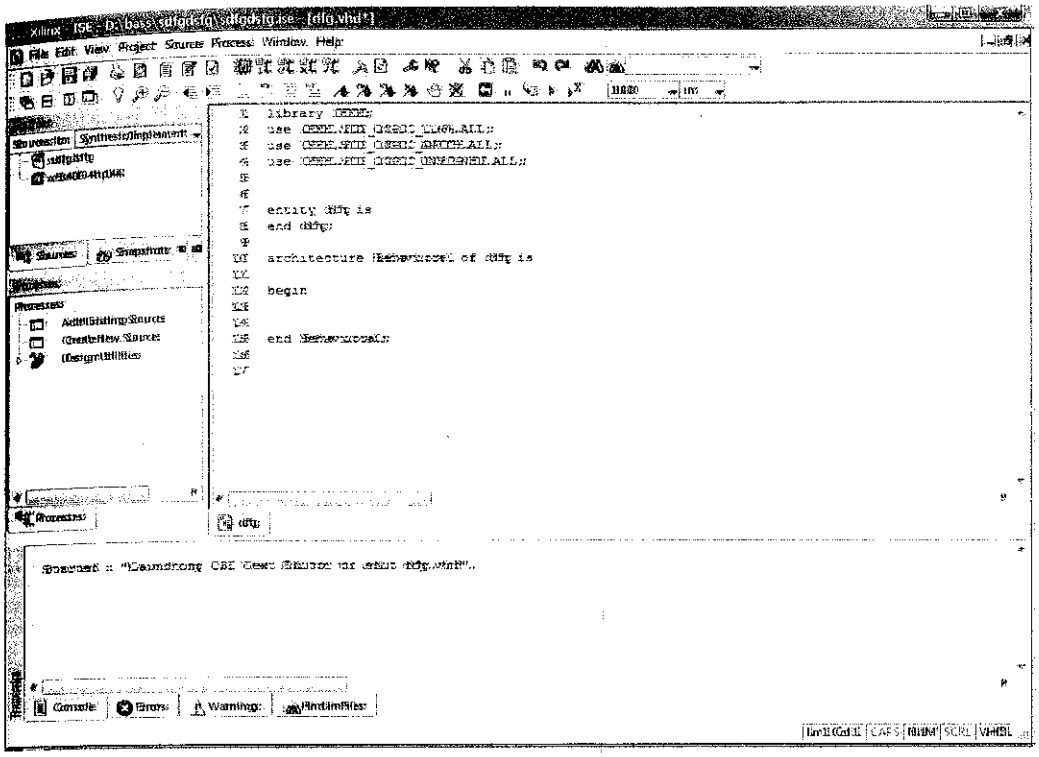
รูปที่ 3.13 กด Next เพื่อเสร็จขั้นตอนการเลือก Source



รูปที่ 3.14 จะเป็นการเลือก source ที่นอกเหนือจากที่เราได้เลือกไว้ในขั้นตอนก่อนหน้านี้



รูปที่ 3.15 จะเป็นการแสดงผลที่เราได้เลือกไว้ว่า เราได้เลือกโปรเจคที่มีรายละเอียดอะไรบ้าง



รูปที่ 3.16 เป็นสภาพโปรแกรมที่พร้อมใช้งาน และจะเริ่มการเขียนโปรแกรมที่หน้าจอนี้

บทที่4
โปรแกรมที่ใช้ในการวิเคราะห์สมการและจำลองผล

4.1 การใช้งานโปรแกรม

หลังจากที่ติดตั้งโปรแกรมครบชุดแล้ว เปิดโปรแกรม WebPACK โดยคลิกที่ไอคอน WebPACK Project Navigator  ใน Xilinx WebPACK 6.1 program group แล้วจะมี Project Navigator ปรากฏขึ้นมา

Project navigator เป็นหน้าจอหลักของโปรแกรม Webpack นี้ จะมีหน้าต่างย่อยที่ใช้บ่อยอยู่ด้วยกันสามหน้าต่างดังนี้

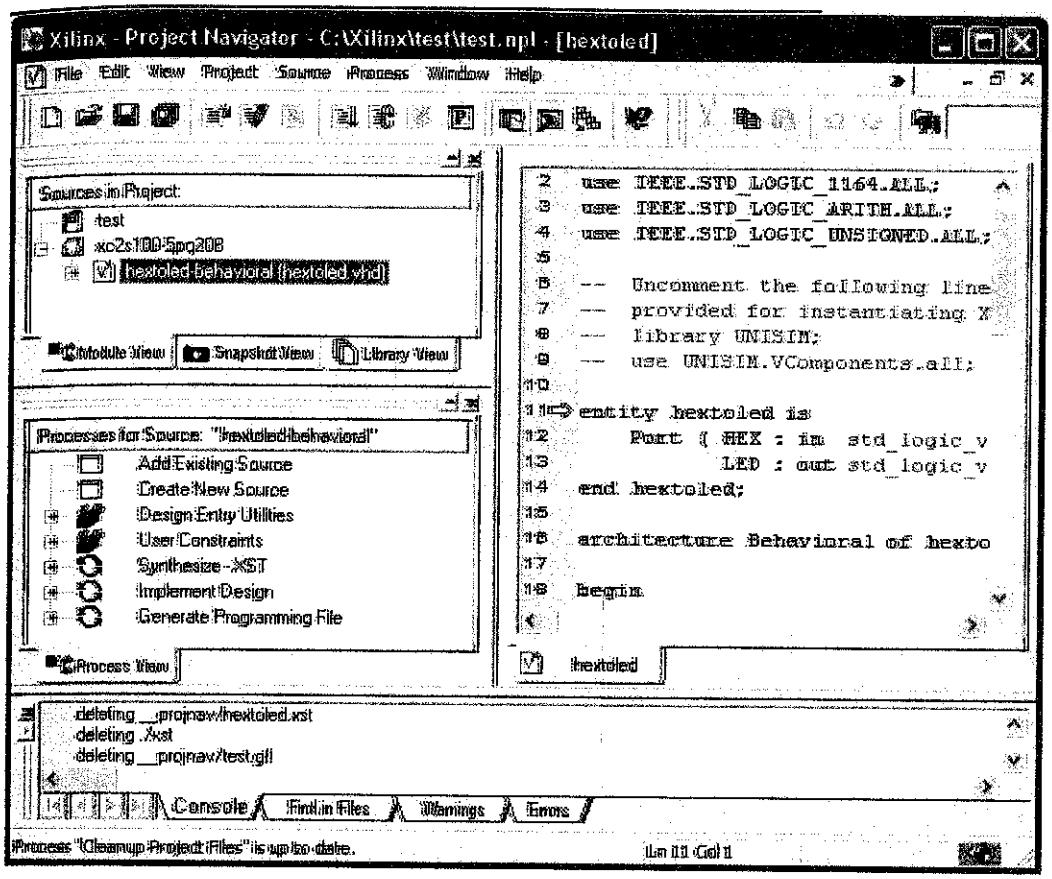
4.1.1 Sources in Project window แสดงตระกูลของ FPGA (Field Programmable Gate Array) และไฟล์ที่มีอยู่ในโปรเจกต์ เป็นหน้าต่างซ้ายบนของรูปที่ 4.1 ไอคอนที่ปรากฏในหน้าต่างนี้จะบ่งบอกชนิดของไฟล์ต่างๆ

a. Project notebook (.npl) บอกถึงชื่อโปรเจกต์ที่เปิดอยู่ในปัจจุบัน
b. Document File (.txt, .wri, .doc etc.) คำอธิบายต่างๆ ที่เกี่ยวข้องกับโปรเจกต์
c. Targeted Device and Synthesis Tool บอกถึงชนิดของ FPGA ที่ใช้ในโปรเจกต์
d. Schematic (.sch) source file ที่เป็นลายวงจร
e. VHDL logic description (.vhd) source file ที่เขียนด้วยภาษา VHDL

4.1.2 Processes for Current Source window แสดงคำสั่งที่สามารถกระทำกับ source file ที่เราเลือกไว้ใน *sources window* ได้ อยู่ด้านซ้ายล่างของรูปที่ 1 ไอคอนที่ปรากฏจะบอกถึงเครื่องมือชนิดต่างๆ

f. Utilities รวมเอา process และ EXE ต่างๆ เอาไว้
g. Report ดูบันทึกผลของการเรียกใช้ process ว่าสำเร็จหรือไม่
h. EXE เรียกโปรแกรมอื่นนอก WebPACK
i. Process เรียกใช้ process ในโปรแกรม WebPACK เอง

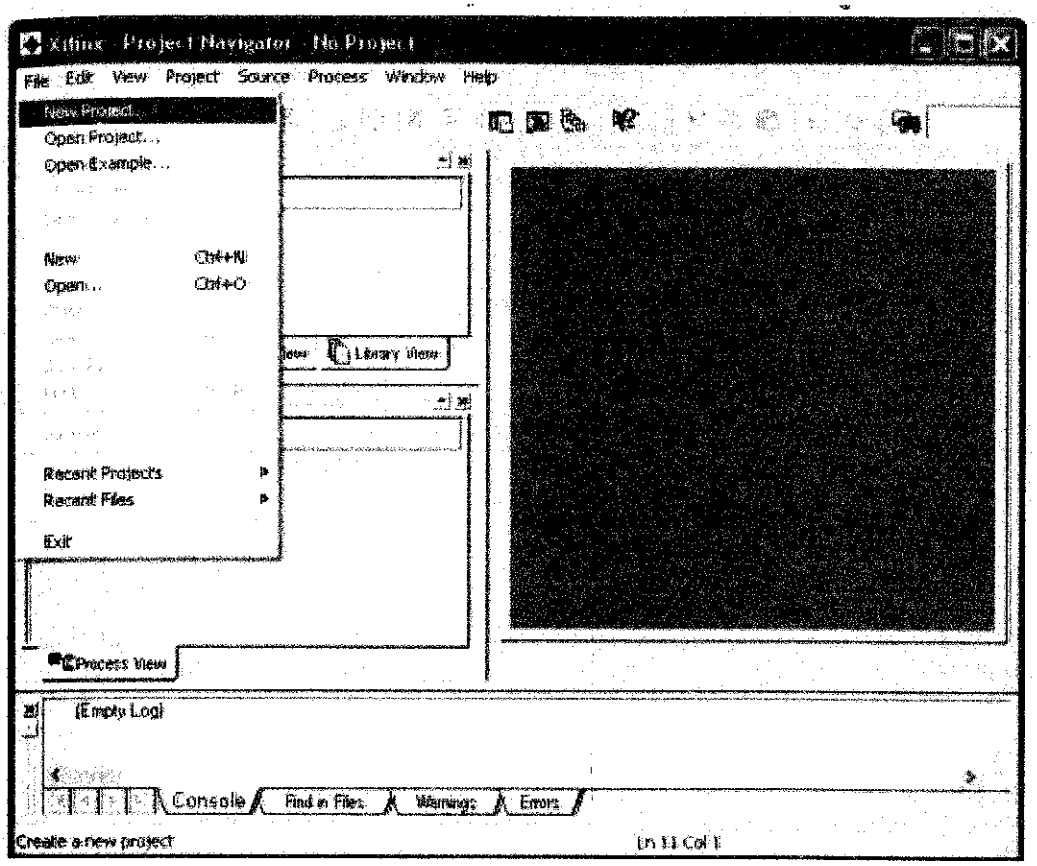
HDL Editor window สำหรับใช้แก้ไข source ที่เป็นข้อความ (ในรูปข้างล่างนี้เป็นของไฟล์ที่ชื่อ my_mod.vhd) อยู่ทางขวาของรูปที่ 4.1



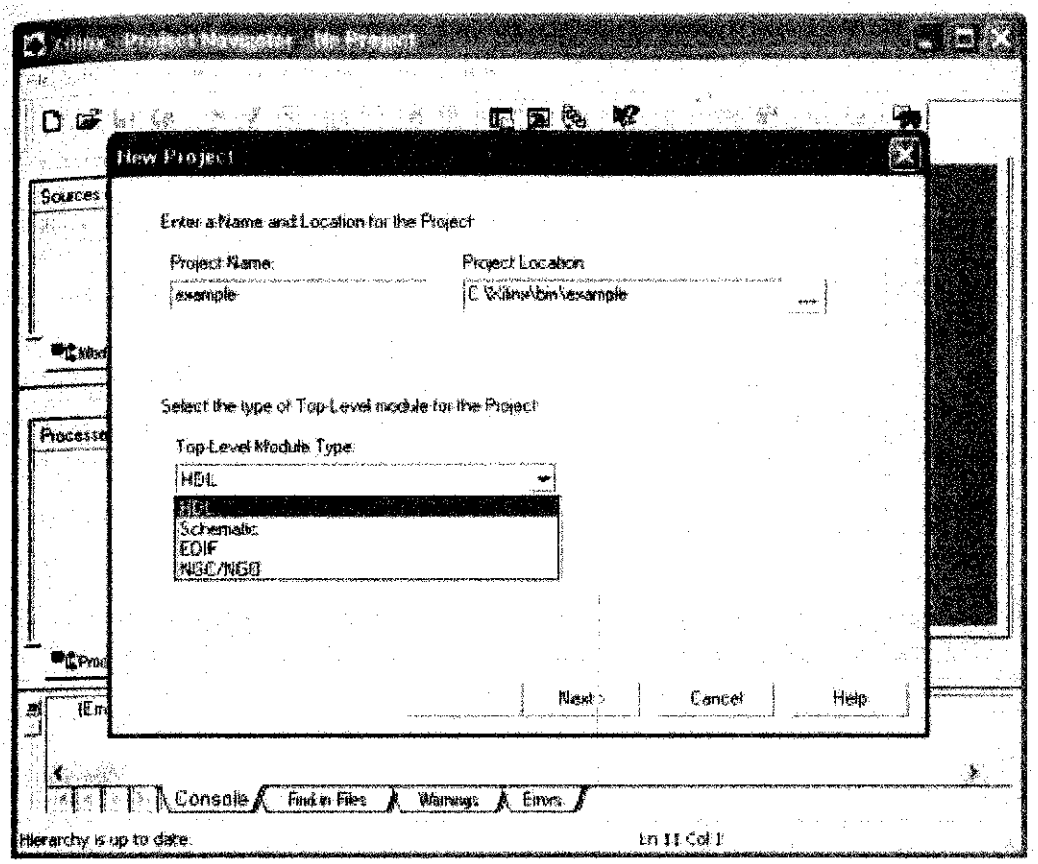
รูปที่ 4.1 Project navigator

4.2.การสร้างโปรเจกต์ใหม่

4.2.1 เลือกเมนู File -> New Project จาก project navigator จะมีไดอะล็อก *New Project* ปรากฏขึ้นมาเพื่อให้ตั้งชื่อไฟล์ของโปรเจกต์ดังรูปที่ 3 ไฟล์ที่สร้างขึ้นนี้มีนามสกุล เป็น *.npl

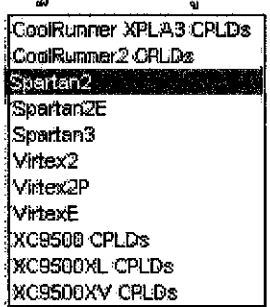


รูปที่ 4.2 เมนู New Project

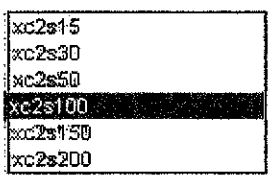


รูปที่ 4.3 ตั้งชื่อ project

4.2.1.1 เลือกตระกูลของ FPGA ที่ใช้ในการออกแบบโดยคลิกในช่อง *Value* ในบรรทัด *Device Family* แล้วจะมีรายการปรากฏขึ้นให้เลือก ดังรูปข้างล่าง ให้เลือก **Spartan2**



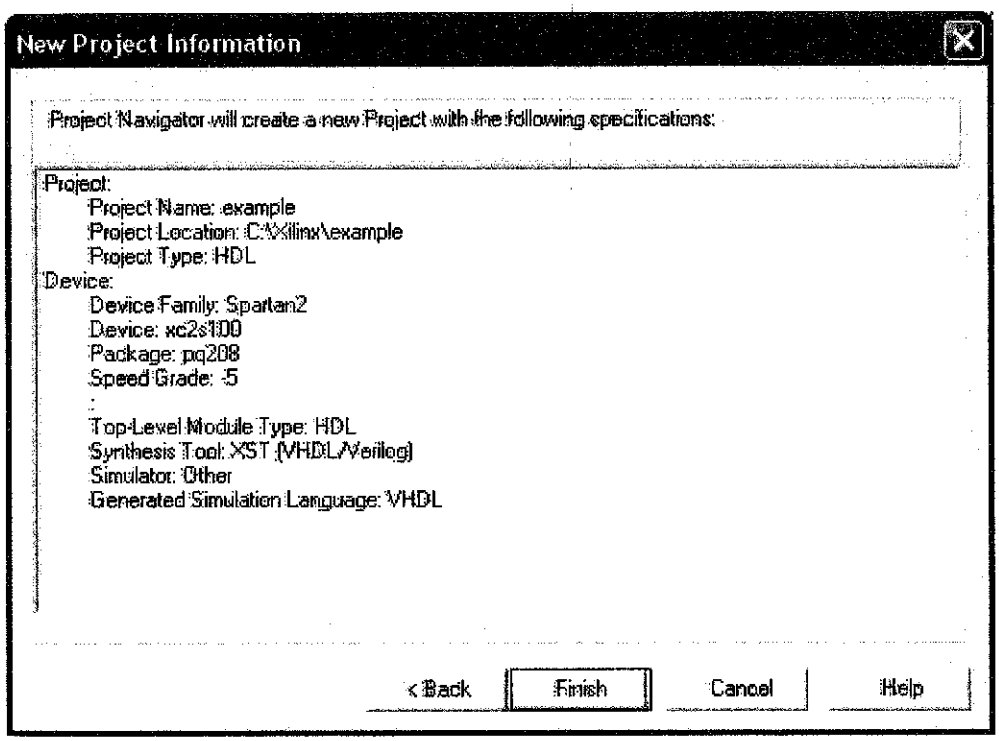
4.2.2 เลือกชนิดและ package ของ FPGA ที่จะใช้ โดยคลิกในช่อง *Value* บรรทัด *Device* จะมีรายการให้เลือกคล้ายในข้อ 2 หรือสามารถที่จะเลือกเป็นแบบ AUTO เพื่อให้โปรแกรมตัดสินใจเลือกชนิดและขนาดที่เหมาะสมที่สุด ให้เลือก **XC2S100** สำหรับ Speed Grade ให้เลือก **-5** และ Package ให้เลือกเป็น **PQ208**



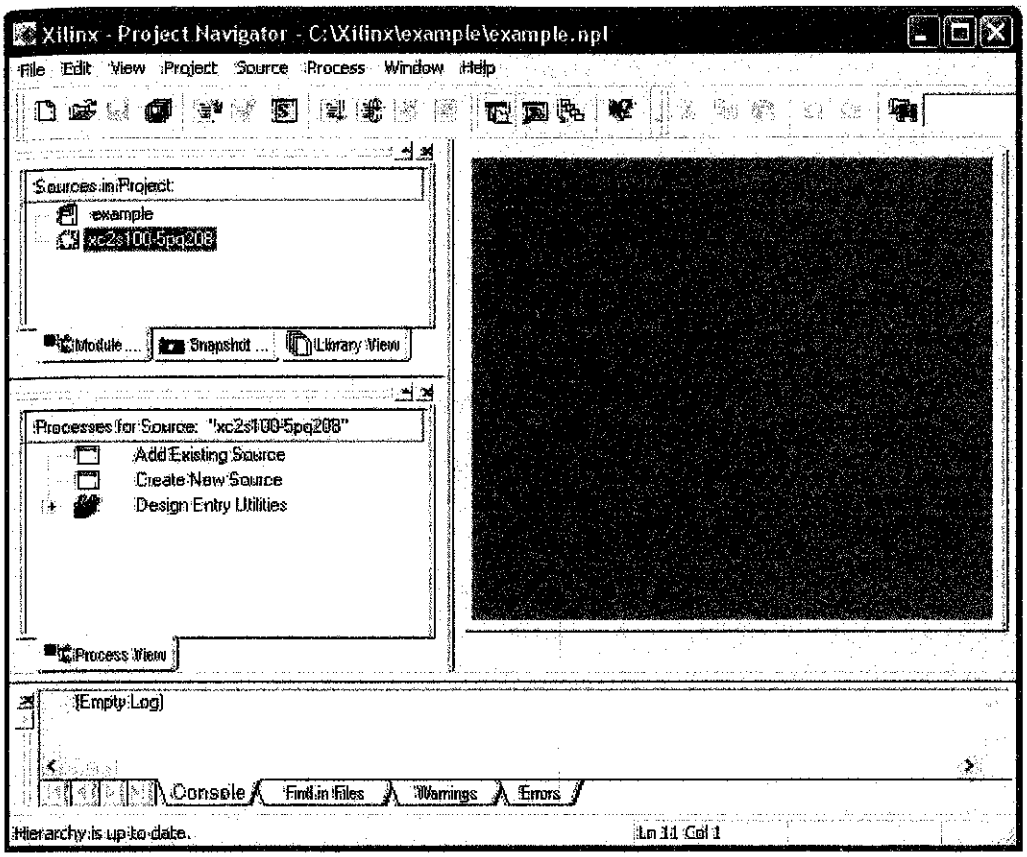
4.2.3 เลือกเครื่องมือที่จะใช้ในการสังเคราะห์วงจร ในช่อง *Synthesis Tool* ในที่นี้คือ XST (Xilinx Synthesis Tool) และเลือกภาษาที่ใช้ ได้แก่ VHDL, Verilog หรือ ABEL ให้เลือกเป็น **XST VHDL**

4.2.4 เมื่อกด Next จะปรากฏหน้าต่างสำหรับการสร้างและเพิ่ม Source File ใหม่ลงในโปรเจกต์ ให้กด Next ไปก่อน 2 ครั้ง (ดูรายละเอียดการสร้างและเพิ่ม Source File ในหัวข้อ Add of Create New Source) หน้า New Project Information จะปรากฏขึ้นมา ซึ่งเป็นการแจ้งรายละเอียดของโปรเจกต์ที่เราเพิ่งสร้างขึ้น

4.2.5 เมื่อกด Next จะปรากฏหน้าต่างสำหรับการสร้างและเพิ่ม Source File ใหม่ลงในโปรเจกต์ ให้กด Next ไปก่อน 2 ครั้ง (ดูรายละเอียดการสร้างและเพิ่ม Source File ในหัวข้อ Add of Create New Source) หน้า New Project Information จะปรากฏขึ้นมา ซึ่งเป็นการแจ้งรายละเอียดของโปรเจกต์ที่เราเพิ่งสร้างขึ้น



4.2.6 เมื่อกด Finish จะกลับมาที่ Project Navigator ซึ่งตอนนี้จะมีโปรเจกต์ใหม่ที่เราเพิ่งสร้างขึ้นปรากฏอยู่ในหน้าต่าง *Sources*

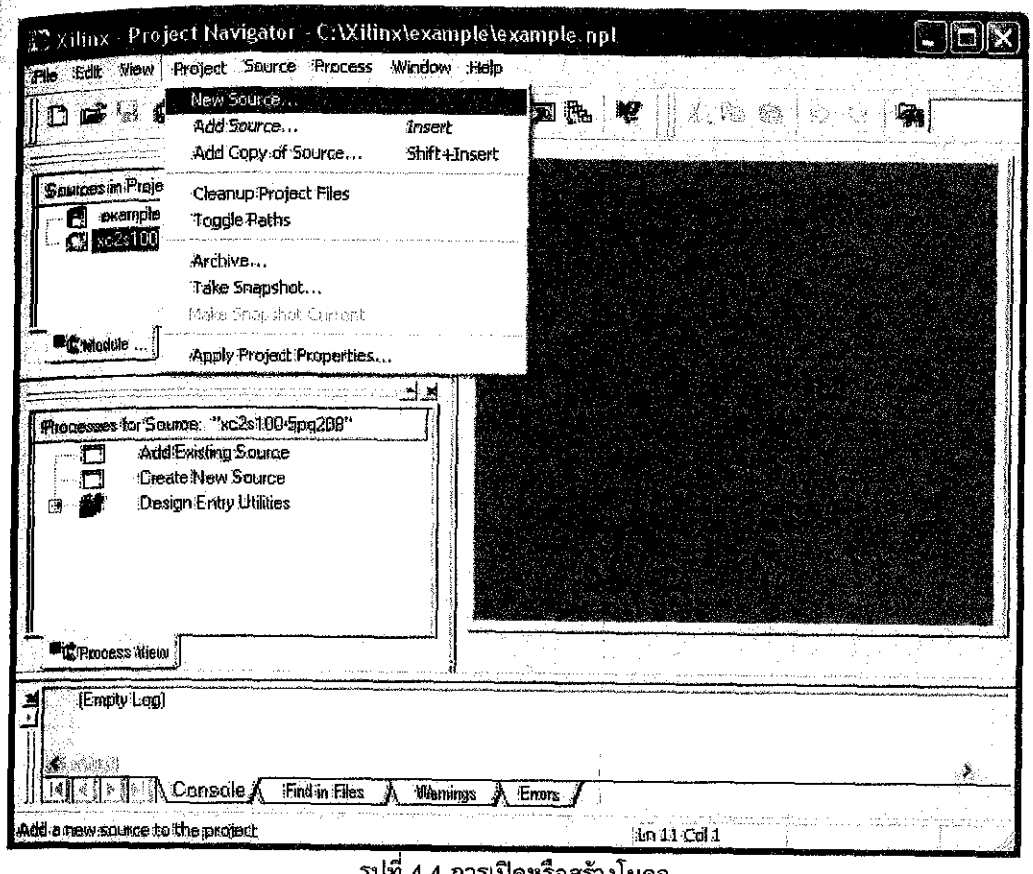


รูปที่ 4.3 โปรเจกต์ใหม่ที่เพิ่งสร้างขึ้นมา

4.2.7 ถ้าต้องการแก้ไขชื่อให้กับโปรเจกต์ ให้คลิกขวาที่ชื่อโปรเจกต์ในหน้าต่าง *Sources* แล้วเลือก *Properties* จะสามารถแก้ไขชื่อโปรเจกต์ในบรรทัดที่เขียนว่า *Project Title* ได้ จากนั้นให้กด OK เพื่อกลับมายัง project navigator

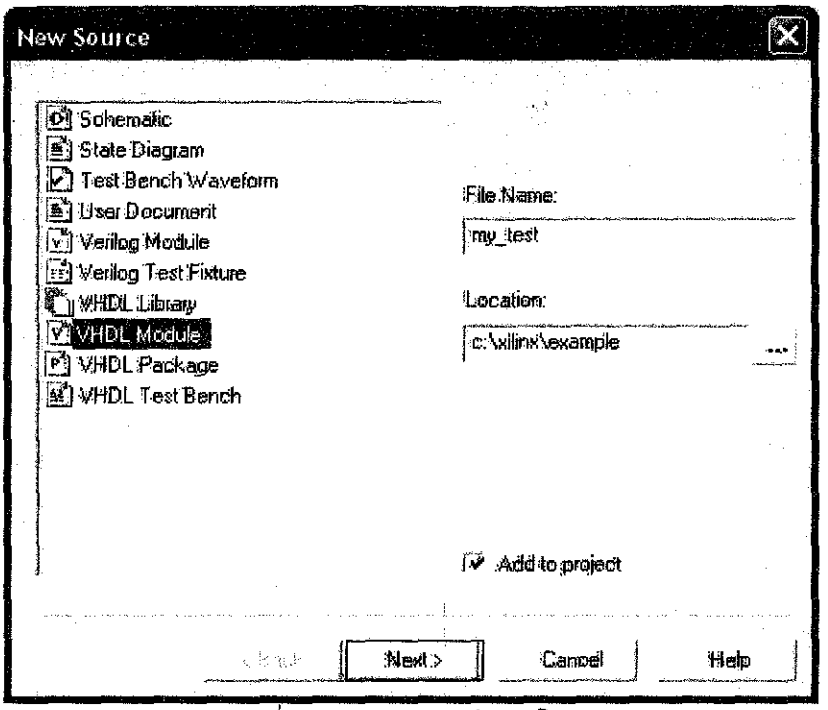
4.3 การเปิดหรือสร้าง source file ใหม่ source ที่เพิ่มเข้าไปในโปรเจกต์นั้นจะเรียกอีกอย่างหนึ่งว่า Module เป็นองค์ประกอบหนึ่งของโปรเจกต์ที่เราออกแบบนั่นเอง

4.3.1 เลือก Project -> Add Source... หรือ Project -> New Source... จากเมนู ไฟล์ที่เลือกได้ เช่น *.vhd และ *.sch เป็นต้น



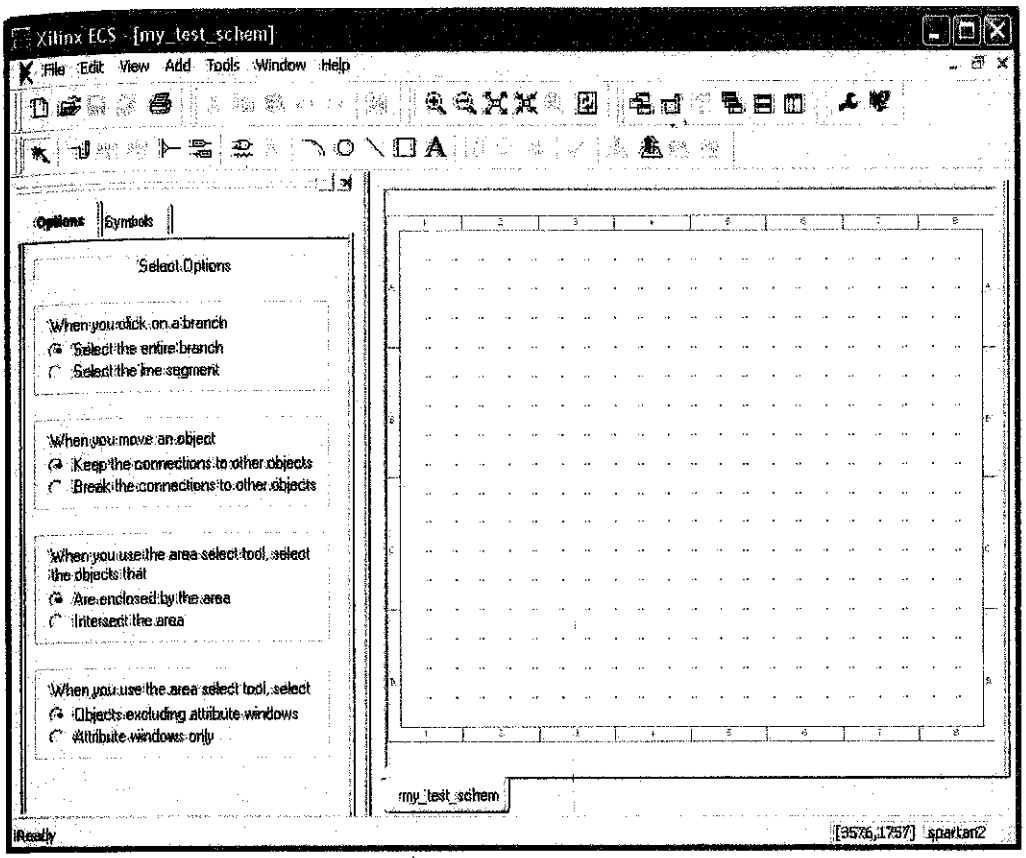
รูปที่ 4.4 การเปิดหรือสร้างโมดูล

4.3.2 กรณีที่สร้างไฟล์ใหม่ (New source) จะปรากฏไดอะล็อก *New source type* ขึ้นมาให้เลือกว่าจะสร้าง source file ชนิดใด โดยชนิดที่เลือกได้นี้จะสัมพันธ์กับ synthesis tool ที่เลือกไว้ในตอนสร้างโปรเจกต์ด้วย อย่างเช่นในรูปที่ 6 เป็นผลที่ได้จากการเลือก synthesis tool เป็น XST VHDL ซึ่ง source file ชนิดที่สำคัญสองตัวได้แก่ **VHDL Module** และ **Schematic**



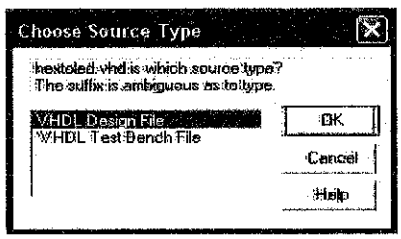
รูปที่ 4.5 กำหนดคุณสมบัติของโมดูล

4.3.3 เลือก **Schematic** เพื่อสร้างวงจรใหม่ขึ้นมา คำว่า schematic นั้นหมายถึงภาพวาดของวงจรนั่นเอง จากนั้นให้ใส่ชื่อ Schematic ในช่อง *File Name* แล้วกด Next> แล้วก็ Finish จะปรากฏโปรแกรม Schematic Editor ขึ้นมาดังรูปที่ 4.6 ซึ่งโปรแกรมนี้ใช้สำหรับสร้างวงจรหรือเรียกว่าโมดูลสำหรับโปรเจกต์ที่เรากำลังออกแบบอยู่ วิธีใช้โปรแกรมนี้ให้ศึกษาเพิ่มเติมเอง



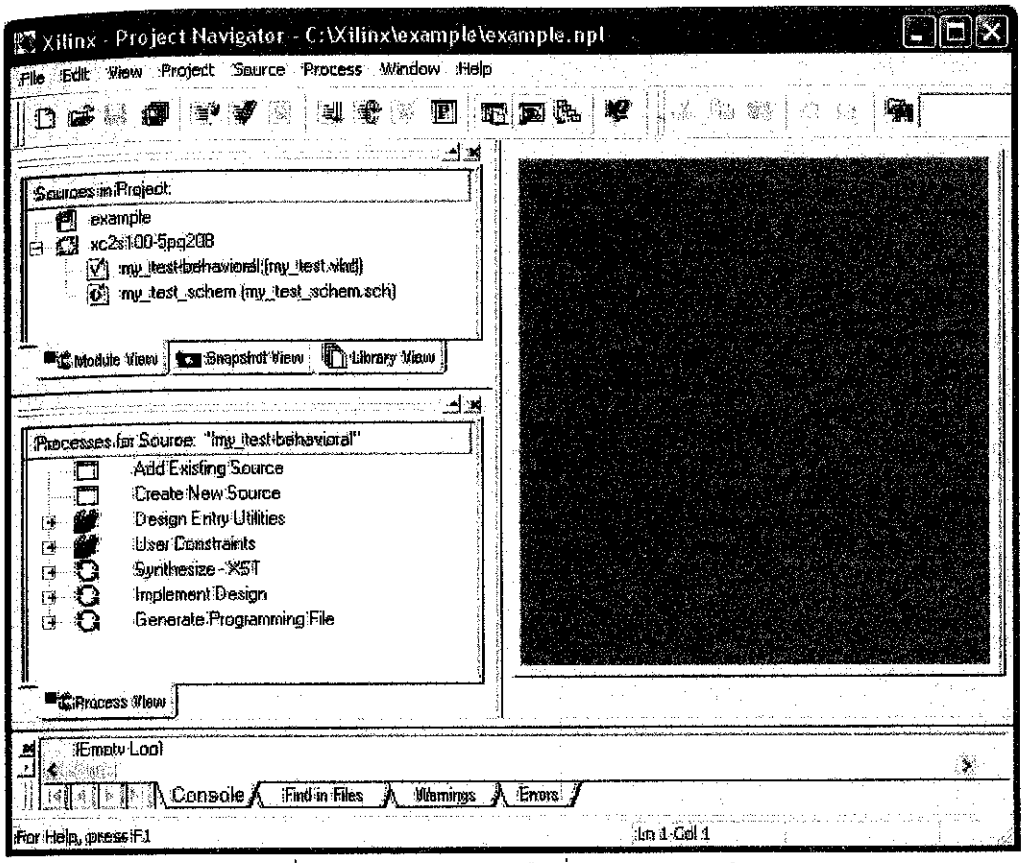
รูปที่ 4.6 Schematic Editor

4.3.4 กรณีของไฟล์ตัวอย่าง "bin27seg.vhd" นั้นเป็น VHDL Module ที่ทำงานเหมือนวงจร binary to 7 segments ถ้าต้องการนำโมดูลนี้มาใส่ ก็ต้องทำการ Add Source ให้กับโปรเจกต์ แล้วเลือกไฟล์ที่ต้องการ จากนั้นปรากฏไดอะลอกถามว่าไฟล์นามสกุล .vhd ที่เลือกนั้นเป็นไฟล์ชนิดใด ดังรูปที่ 8 ให้เลือก **VHDL Design File**



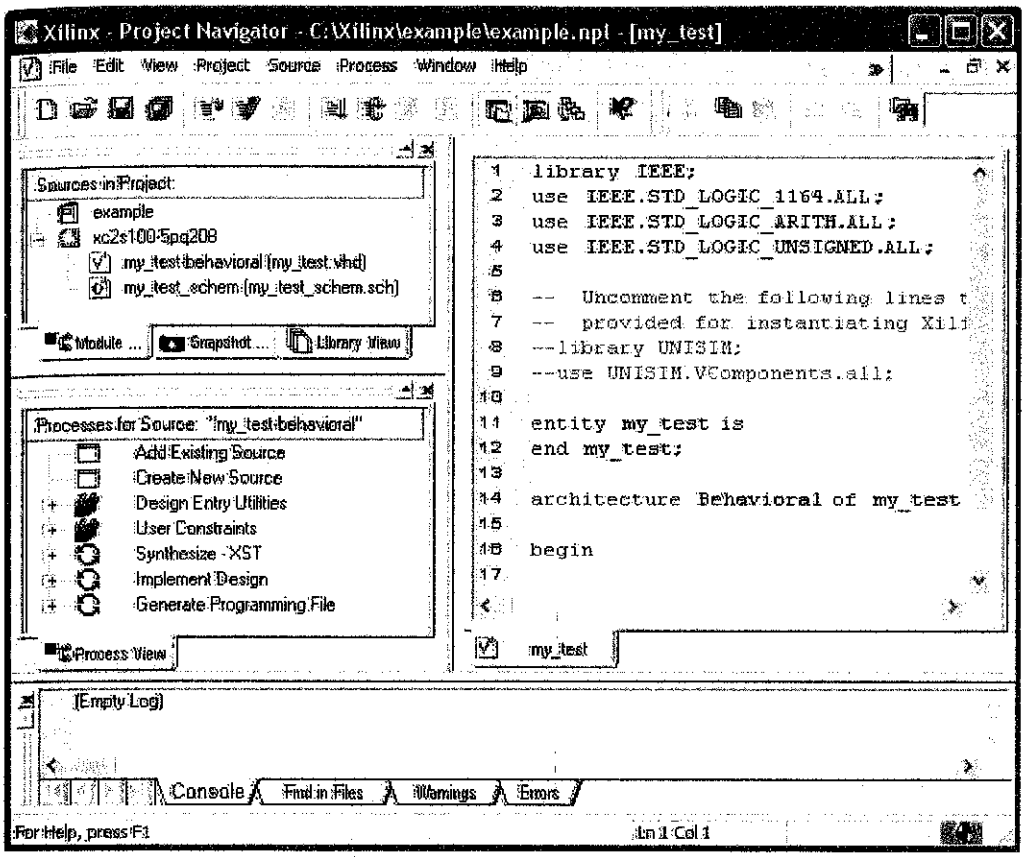
รูปที่ 4.7 เลือกชนิดของไฟล์ .vhd

หมายเหตุ : การ Add Source ชนิดที่เป็น schematic นั้นคล้ายกับของ VHDL แต่ไม่ต้องเลือกชนิดของไฟล์ เมื่อทำการสร้างหรือเพิ่ม source ใหม่แล้ว จะมีชื่อโมดูลและไฟล์ของโมดูลนั้นปรากฏขึ้นใต้ชื่อของ FPGA ในหน้าต่าง *Sources* ดังรูปที่ 9



รูปที่ 4.8 หลังจากทำการสร้าง หรือเพิ่ม source files แล้ว

4.3.5 สามารถเพิ่มรายละเอียด หรือทำการแก้ไขรายละเอียดของแต่ละโมดูลได้ โดยการเลือกที่ชื่อโมดูลนั้นในหน้าต่าง *Sources* แล้วดับเบิลคลิกเพื่อเปิดไฟล์ขึ้นมาทำการแก้ไข ถ้าเป็น source ที่เขียนด้วย VHDL ก็จะมีลักษณะเป็นข้อความ ซึ่งจะปรากฏขึ้นที่หน้าต่าง *documents* ทางขวามือ ดังรูปที่ 10 ส่วนกรณีที่เป็น schematic โปรแกรม Schematic Editor ก็จะเปิด schematic ของโมดูลนั้นขึ้นมาให้



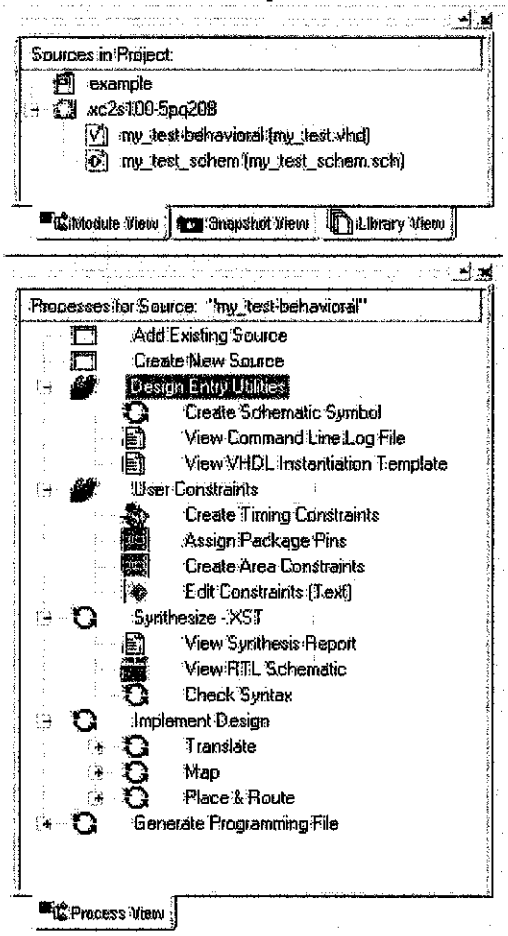
รูปที่ 4.9 การแก้ไข source file ที่เป็น text

4.3.6 เราสามารถสร้าง หรือเพิ่มไฟล์โดยการทำอย่างขั้นตอนที่ 2,3 หรือ 4 ไปจนกว่าจะได้ไฟล์ที่ต้องการครบ

4.3.7 โมดูลต่างๆ จะมีการจัดระดับชั้น เนื่องจากเวลาออกแบบโปรเจกต์ มักจะมีไฟล์หนึ่งที่รวมเอาทุกส่วนของโปรเจกต์เข้าไว้ด้วยกัน เรียกว่า top level design ลักษณะของไฟล์นี้จะมีโมดูลอื่นๆที่เราสร้างขึ้นมาอยู่ภายใน โมดูลนั้นก็จะอยู่ภายใต้ top level อีกทีหนึ่ง ดังเช่นในรูปที่ 10 หน้าต่าง *Sources* จะเห็นว่าโมดูลชื่อ display นั้นมีระดับชั้นสูงสุด เป็น top level รองลงไปก็จะมีโมดูลอย่างเช่น bin27seg, mod200 และ mod60 เป็นต้น ประกอบอยู่ ในการดำเนินการอะไรกับโมดูลนั้น เราสามารถสั่งให้โปรแกรมทำงานกับโมดูลย่อยๆก็ได้ หรือทำงานกับทุกโมดูลย่อย โดยการเลือกทำงานกับโมดูลที่เป็น upper level ของโมดูลย่อยเหล่านั้นก็ได้

4.4 การสังเคราะห์วงจร หรือ Synthesize เมื่อสร้างโมดูลขึ้นมาแล้ว ไม่ว่าจะอยู่ในรูปของ schematic หรือภาษา VHDL ก็ยังไม่อยู่ในรูปแบบที่ FPGA จะสามารถเข้าใจได้ ก่อนที่จะแปลงให้อยู่ในรูปแบบที่ FPGA เข้าใจได้นั้น จะต้องทำอะไรบางอย่างกับ sources ที่เราสร้างขึ้นเสียก่อนโดยการสังเคราะห์ให้ไปอยู่ในรูปของ netlist ซึ่งเป็นการกำหนดการเชื่อมต่อระหว่างโมดูลย่อยๆภายใน source นั้นว่ามีการเชื่อมต่อกันอย่างไร สำหรับโปรแกรม WebPACK จะใช้ XST (Xilinx Synthesis Technology) เป็นตัวสังเคราะห์วงจร ซึ่งจะให้ผลลัพธ์ออกมาเป็น EDIF netlists

4.4.1 เมื่อการออกแบบวงจรเสร็จเรียบร้อยแล้ว สามารถจะสั่งให้ WebPACK ทำการ synthesize ได้โดยเลือก source file ที่ต้องการจะสังเคราะห์ แล้วดับเบิลคลิกที่ **Synthesize** ในหน้าต่าง *Processes* แสดงในรูปที่ 4.10 รอจนกว่าจะเสร็จ (มีเครื่องหมาย ✔ อยู่หน้าคำว่า Synthesize)

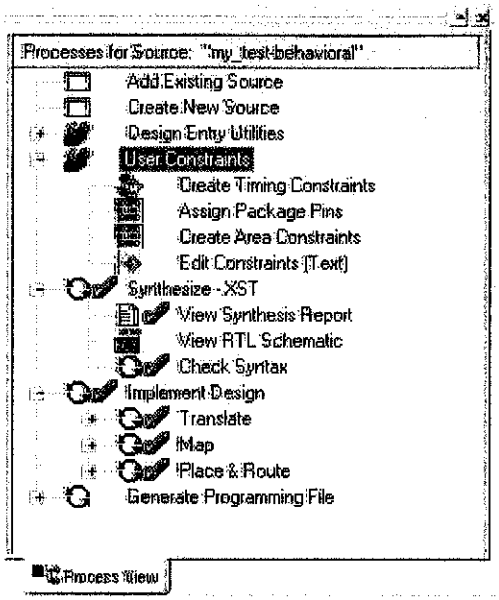


รูปที่ 4.10 Synthesize-Process

4.4.2 หลังจากการสังเคราะห์แล้ว สามารถดูรายงานผลการสังเคราะห์ได้โดยดับเบิลคลิกที่ **View Synthesis Report** ในหน้าต่าง *Processes*

4.5 การ Implement แบบที่เราสร้างไว้ เมื่อได้ netlist จากการ synthesis มาแล้ว ก็สามารถที่จะทำการจัดวงจรที่เราได้ออกแบบไว้ให้อยู่ในรูปแบบที่จัดเตรียมไว้ใน FPGA (Fitting a design) หรือก็คือการจัดวางวงจรที่เราออกแบบไว้ลงไปในชิป FPGA ชนิดที่เราได้เลือกไว้นั่นเอง

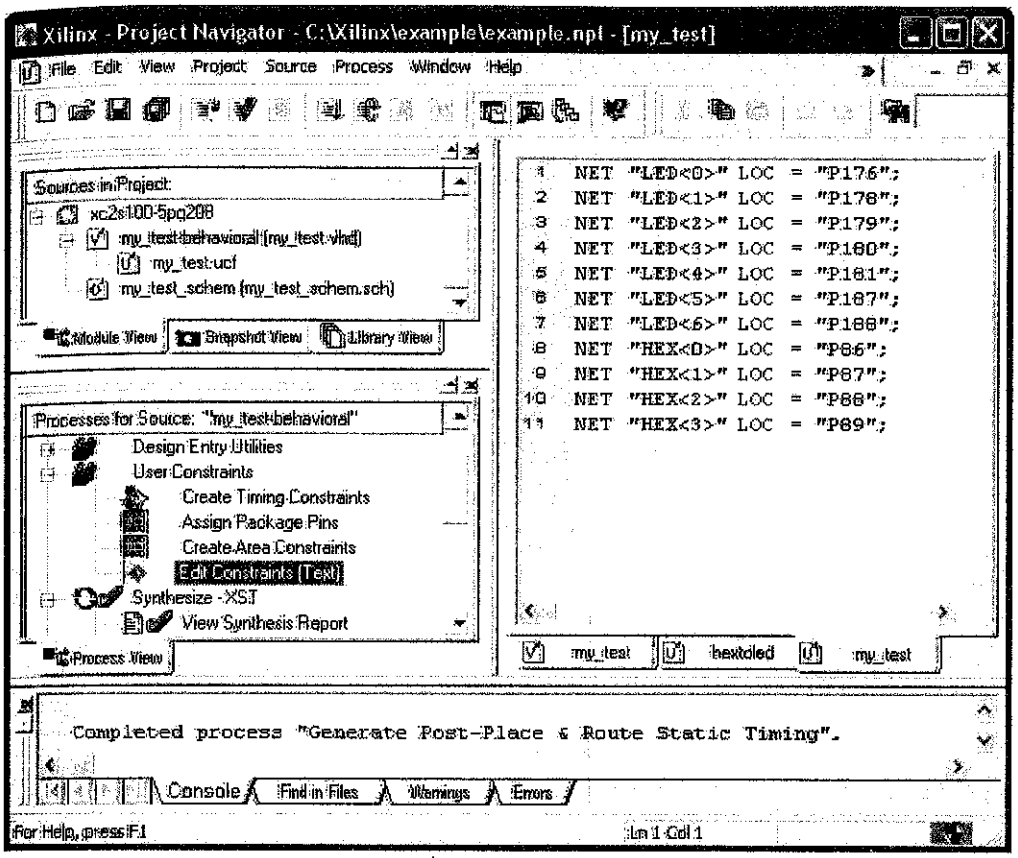
4.5.1 ในหน้าต่าง *sources* เลือกที่โมดูลระดับบนสุด จากนั้นดับเบิลคลิกที่ **Implement Design** ในหน้าต่าง *Processes* ดังรูปที่ 12 เพื่อเริ่มการ implement



รูปที่ 4.11 Implement Design

4.5.2 ในขั้นตอนนี้ เราจะต้องกำหนดให้กับโปรแกรมรู้ก่อนว่าสัญญาณแต่ละสัญญาณในวงจรที่เราออกแบบนั้นคือขาไหนของชิป FPGA ซึ่งการกำหนดทำได้โดยไปที่หน้าต่าง *Processes* และดับเบิลคลิกที่ **Design Entry Utilities -> User Constraints -> Edit Implementation Constraints File** เพื่อเปิดไฟล์นามสกุล .ucf ซึ่งจะทำหน้าที่ระบุ mapping ระหว่างสัญญาณในรูปวงจรกับขาของชิปจริง แล้ว Notepad ก็จะเปิดขึ้นมา ดังในรูปที่ 12 ในรูปนี้เป็นตัวอย่างที่มีการกำหนดขาไว้เรียบร้อยแล้ว ในกรณีที่สร้างโปรเจกต์ใหม่ ให้นิสิตคัดลอกจากไฟล์ *.ucf ที่อยู่ในตัวอย่างไปแปะใน ucf file ของโปรเจกต์ที่สร้างขึ้นมาเอง และทำการแก้ชื่อขาให้ตรงกับที่ต้องการ

หมายเหตุ ถ้ามีการแก้ไข ucf file แล้ว จะต้องทำการ implement ใหม่อีกครั้ง เพื่อให้โปรแกรมรับรู้การเปลี่ยนแปลงของขาสัญญาณที่เกิดขึ้น



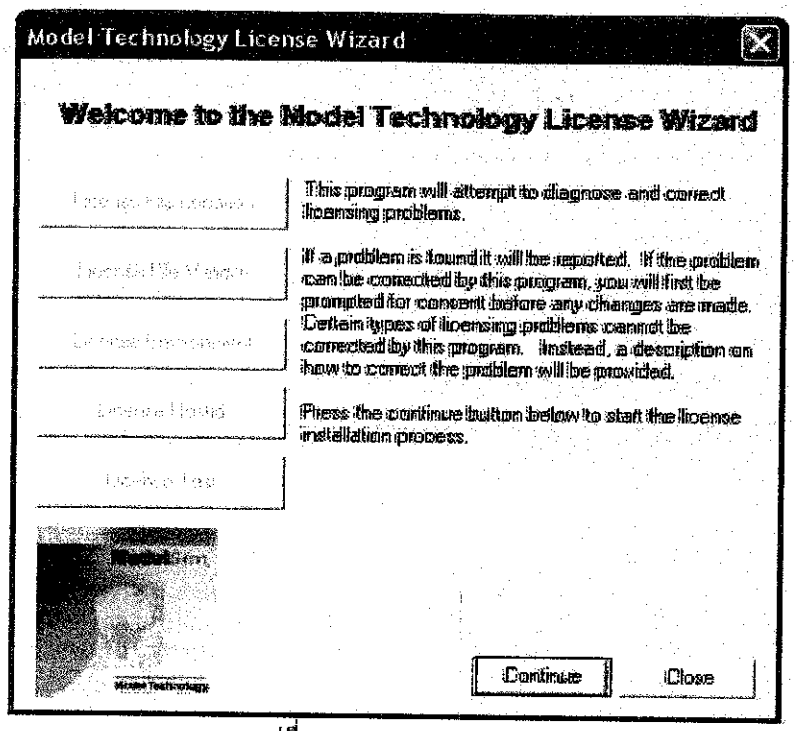
รูปที่ 4.12 Edit UCF file

4.6 การจำลองการทำงาน หรือ Simulation เป็นสิ่งที่จำเป็นต้องทำก่อนการทดลองกับวงจรจริง เพราะการจำลองการทำงานจะเป็นการตรวจสอบสิ่งที่เราออกแบบไว้ว่าทำงานตรงตามที่ออกแบบไว้หรือไม่ เมื่อจำลองการทำงานแล้วพบว่าทำงานถูกต้องตามที่ออกแบบไว้ทุกประการจึงจะนำวงจรที่ออกแบบไว้ไปทดสอบกับชิป CPLD จริงๆ

หมายเหตุ หลังจากลงโปรแกรม ModelSim XE ซึ่งใช้เป็นโปรแกรม Simulator ให้กับชุดโปรแกรม WebPACK แล้วจะยังไม่สามารถใช้งานได้ทันที ต้องทำการของ license file ก่อน ซึ่ง license file ของคอมพิวเตอร์แต่ละเครื่องไม่เหมือนกัน ดังนั้นคอมพิวเตอร์แต่ละเครื่องจะต้องทำการขอ license file เฉพาะของเครื่องนั้นๆ ซึ่งไม่เสียค่าใช้จ่ายแต่อย่างใด ต้องเป็นเครื่องที่ต่ออินเตอร์เน็ตได้ มีขั้นตอนในการขอดังนี้

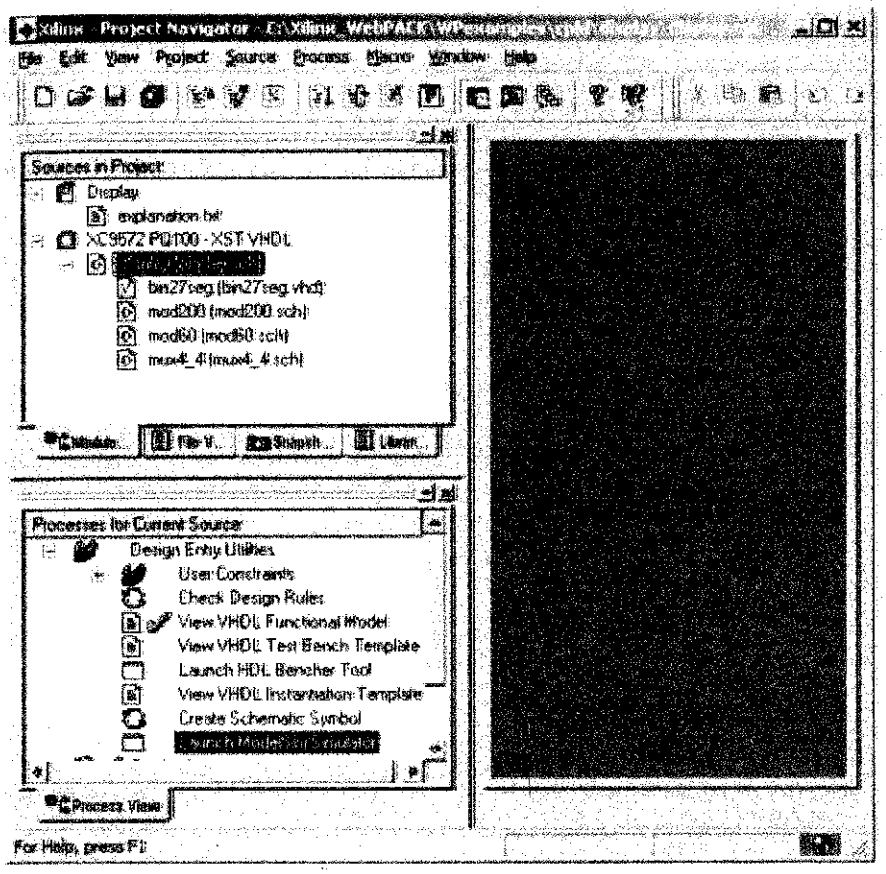
4.6.1 หลังจากลงโปรแกรมแล้ว ให้ไปที่ **Start -> Programs -> ModelSim XE 5.3d -> Submit License Request** จะมี web browser เปิดขึ้นมา ให้กรอกข้อมูลลงไปในช่องว่างจนครบ แล้วกด **Submit** ทางบริษัท ModelSim จะส่ง license file กลับมาให้ทางอีเมล์ ซึ่งใช้เวลาน้อยมาก

4.6.2 ดาวน์โหลด license file เรียบร้อยแล้ว ให้ไปที่ **Start -> Programs -> ModelSim XE 5.3d -> Licensing Wizard** จะปรากฏหน้าจอดังรูปที่ 14 ให้กด Continue แล้วโปรแกรมจะถามถึง license file ให้กด **Browse** แล้วเลือกไดเรกทอรีที่เราดาวน์โหลดไฟล์มาเก็บไว้ แล้วกด **OK** โปรแกรมก็จะตรวจสอบ license file ถ้าถูกต้องก็จะไม่มีปัญหาอะไร และสามารถใช้โปรแกรม ModelSim ได้



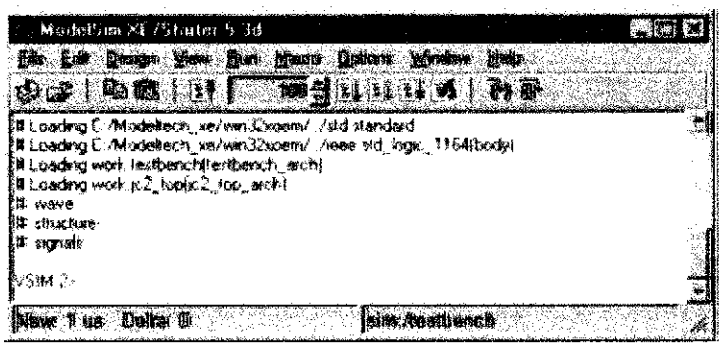
รูปที่ 4.13 Licensing Wizard

4.6.3 เรียกโปรแกรม ModelSim จาก WebPACK โดยเลือกที่โมดูลระดับบนสุด จากนั้นไปที่ **Design Entry Utilities -> Launch ModelSim Simulator** ในหน้าต่าง *Processes* ดับเบิลคลิกเพื่อเรียกโปรแกรม

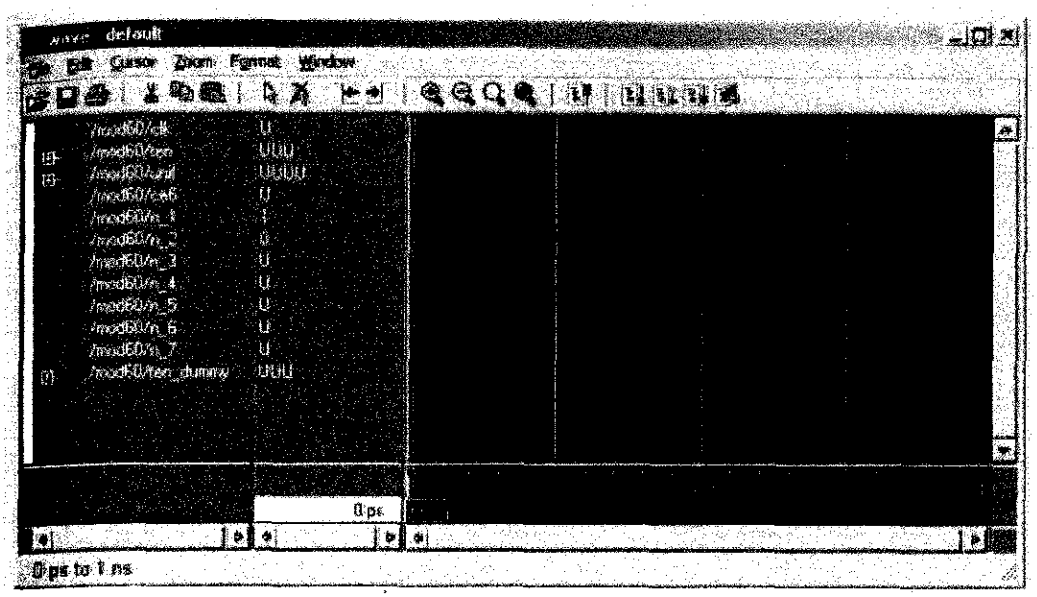


รูปที่ 4.14 Launch ModelSim Simulator

4.6.4 หลังจากเรียกโปรแกรมแล้ว จะปรากฏหน้าต่างขึ้นมาสามหน้าต่างคือ หน้าต่างสำหรับพิมพ์ command line ดังรูปที่ 16, หน้าต่าง *Waveform* ดังรูปที่ 17 และหน้าต่าง *Signals* (ไม่ได้แสดงไว้) สามารถดู help ของโปรแกรมโดยไปที่เมนู **Help -> ModelSim XE User's Manual**



รูปที่4.15 ModelSim command window



รูปที่ 4.16 Display waveform window

4.6.5 กำหนดคาบเวลาให้กับสัญญาณที่ต้องการจะทดสอบ เช่น สัญญาณ clock ที่ป้อนให้กับวงจร ได้โดยการใช้คำสั่ง

force –freeze *item_name value time [,value time] –r period*

โดย *item_name* หมายถึงชื่อสัญญาณ ที่มีการระบุโมดูลย่อยเอาไว้ด้วย คล้ายกับการระบุ path บนดอส เช่น /mod60/clk หมายถึงสัญญาณ clk ในโมดูล mod60 เป็นต้น, *value* หมายถึงระดับ logic ของสัญญาณ ณ เวลาที่กำหนด มีเพียงสองค่าคือ 0 และ 1, *time* (ps)หมายถึงเวลาที่สัญญาณเริ่มมีค่าเท่ากับ value (สามารถทำให้สัญญาณมีการเปลี่ยนแปลงไปมาได้โดยการกำหนด value, time หลายๆชุด) และ period ก็คือการกำหนดว่าจะให้ทำการเปลี่ยนค่าตามรูปแบบที่กำหนดไว้คาบละกี่ ps เราสามารถกำหนดหน่วยของเวลาให้เป็น fs, ps, ns, us, ms และ sec ตัวอย่างเช่น **force –freeze /mod60/clk 0 0, 1 2 us –r 4 us** เป็นการกำหนดสัญญาณ square wave ที่มี duty cycle 50% และคาบ 4 ไมโครวินาที ให้กับสัญญาณที่ชื่อ clk ในโมดูล mod60

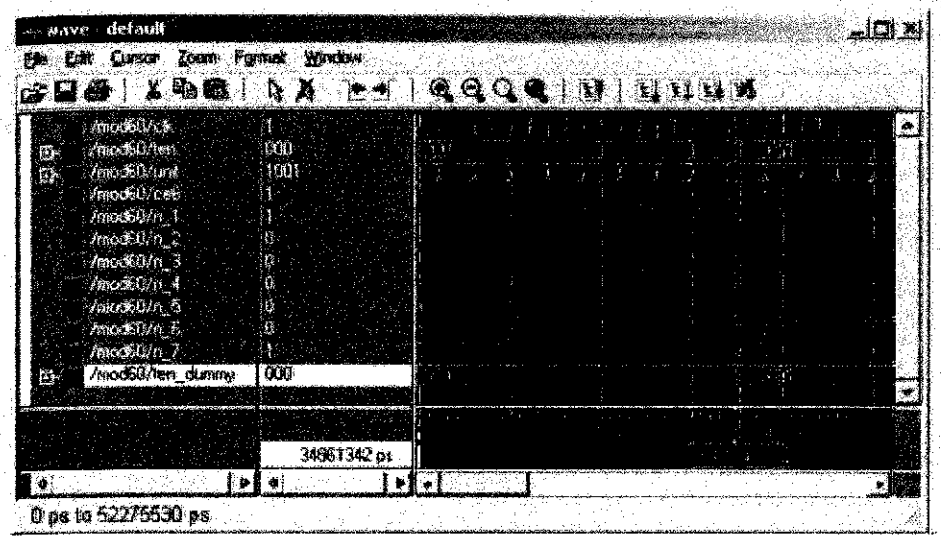
4.6.6 เมื่อกำหนดสัญญาณแล้ว สามารถสั่งให้โปรแกรมเริ่มจำลองการทำงานได้โดยใช้คำสั่ง

run *time time_unit*

โดย *time* คือระยะเวลาที่ต้องการจะจำลองการทำงาน และ *time_unit* คือหน่วยเวลาที่ต้องการ

หมายเหตุ คำสั่ง run จะเป็นการจำลองผลการทำงานต่อจากครั้งก่อน ถ้าหากมีการเปลี่ยนแปลงสัญญาณที่ป้อนให้ หรือต้องการดูผลใหม่ตั้งแต่ต้น ให้ใช้คำสั่ง Restart เพื่อรีเซ็ตการจำลองการทำงานใหม่

4.6.7ผลการจำลองการทำงานจะปรากฏในหน้าต่าง *Wave* ดังเช่นในรูปที่ 18 และสามารถที่จะย่อ หรือขยายภาพสัญญาณ หรือใช้เมาส์คลิกที่ช่วงเวลาที่ต้องการทราบค่า เพื่อทำการอ่านค่าสัญญาณที่ต้องการได้

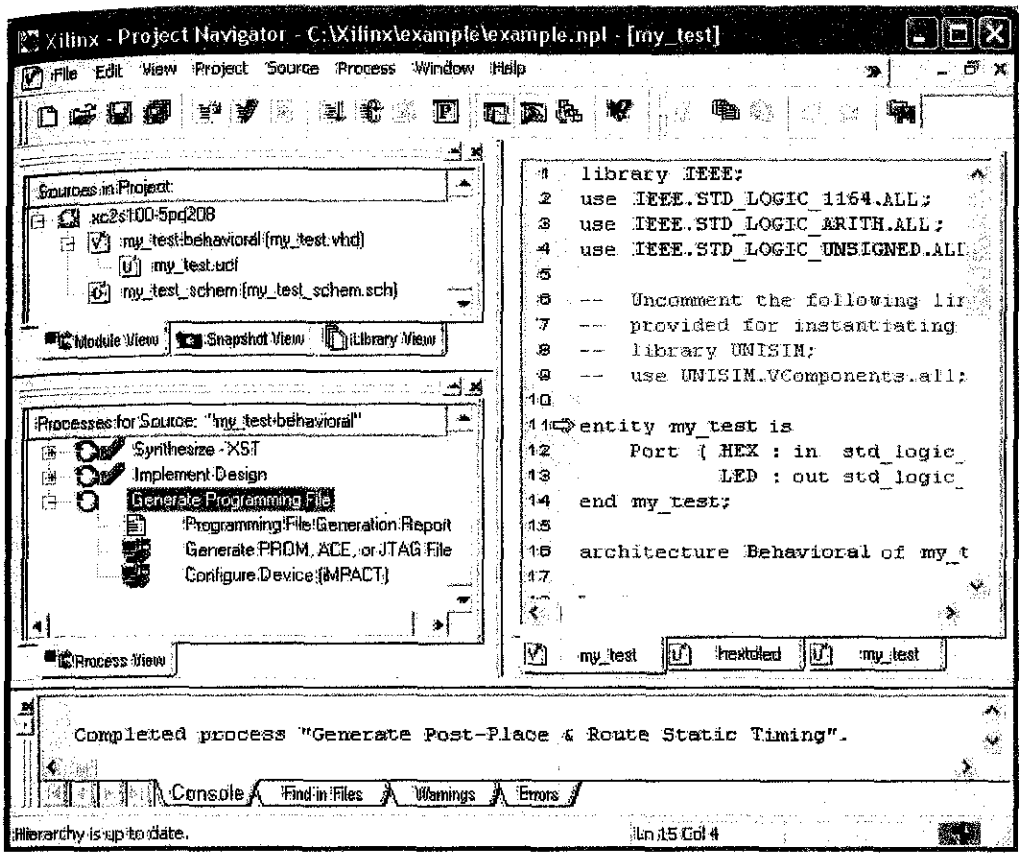


รูปที่ 4.17 ผลการจำลองการทำงาน

หมายเหตุ สำหรับการจำลองการทำงานที่ต้องการช่วงเวลายาวนานมากๆ เช่นการจำลองการทำงานของวงจรนาฬิกา อาจแบ่งการจำลองการทำงานออกเป็นสองส่วน คือส่วนที่ความถี่สูง เช่นสัญญาณนาฬิกาและวงจรหารความถี่สูง ส่วนหนึ่ง อีกส่วนหนึ่งเป็นส่วนที่ความถี่ต่ำ เช่นสัญญาณหนึ่งวินาที โดยการเปลี่ยนค่า item_name ในคำสั่ง force ให้เป็นสัญญาณที่เราต้องการจะป้อนให้วงจรเพื่อทดสอบ

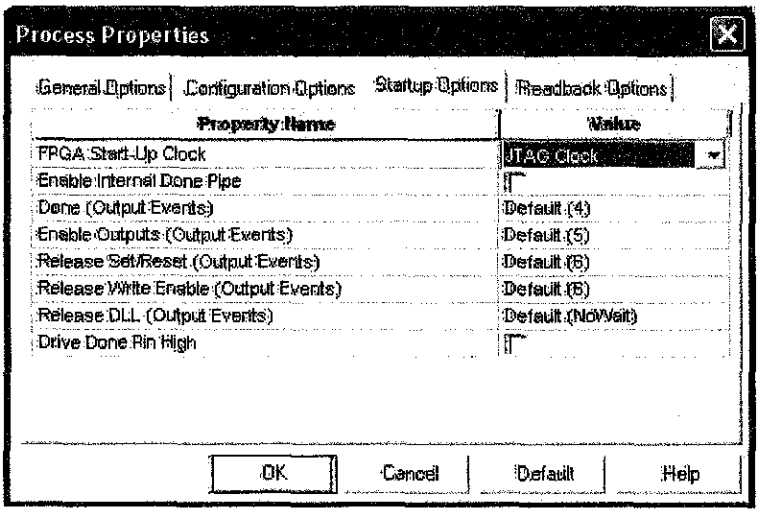
4.7 Device Programming – ดาวน์โหลดวงจรที่ออกแบบไว้ลงสู่ชิป FPGA เป็นขั้นตอนสุดท้ายในการออกแบบ

4.7.1 ต่อสายเคเบิล Xilinx JTAG Parallel Download เข้ากับเครื่องคอมพิวเตอร์ และสายเคเบิลอีกข้างต่อเข้ากับ JTAG บนบอร์ดทดลองที่จะดาวน์โหลด และจ่ายไฟให้กับบอร์ดทดลองให้เรียบร้อย



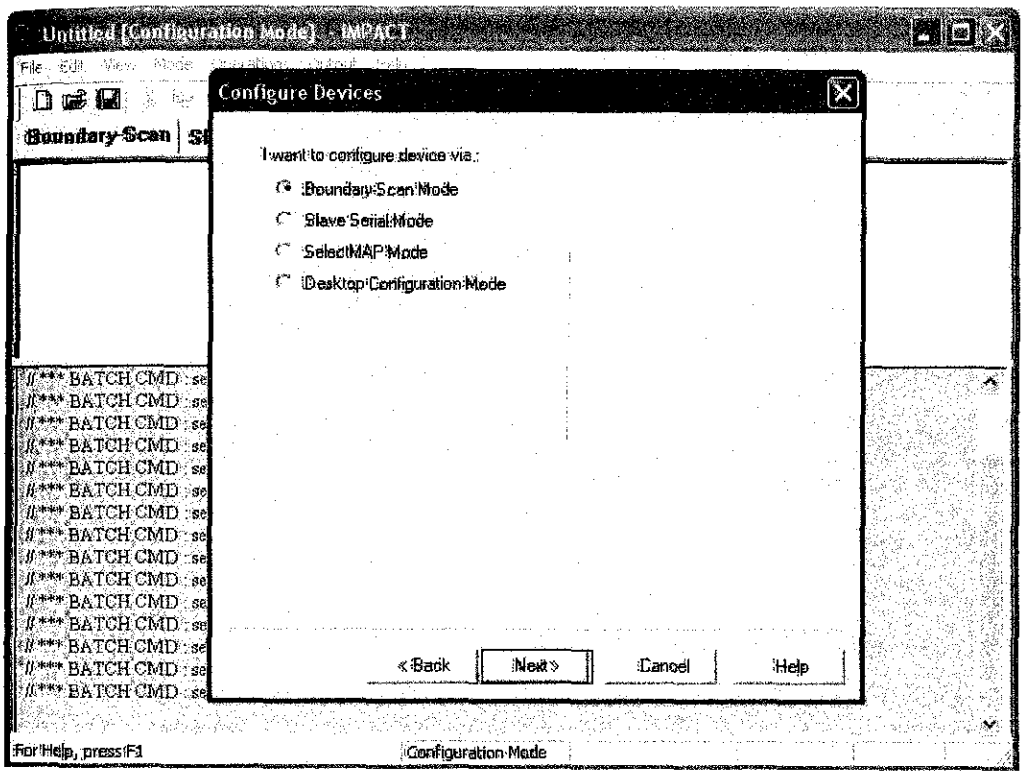
รูปที่ 418 Create Programming File

4.7.1 คลิกขวาที่ **Generate Programming File** แล้วเลือก **Properties..** เพื่อเข้าไปแก้ไขวิธีการดาวน์โหลด เมื่อเลือกแล้วจะมีหน้าต่าง *Process Properties* ปรากฏขึ้นมา ดังรูปที่ 20 ไปที่แท็บ **Startup Option** จากนั้นให้แก้ Property ที่ชื่อ Start-Up Clock จากที่ตั้งไว้เป็น **CCLK** ให้กลายเป็น **JTAG Clock** แล้วกด OK



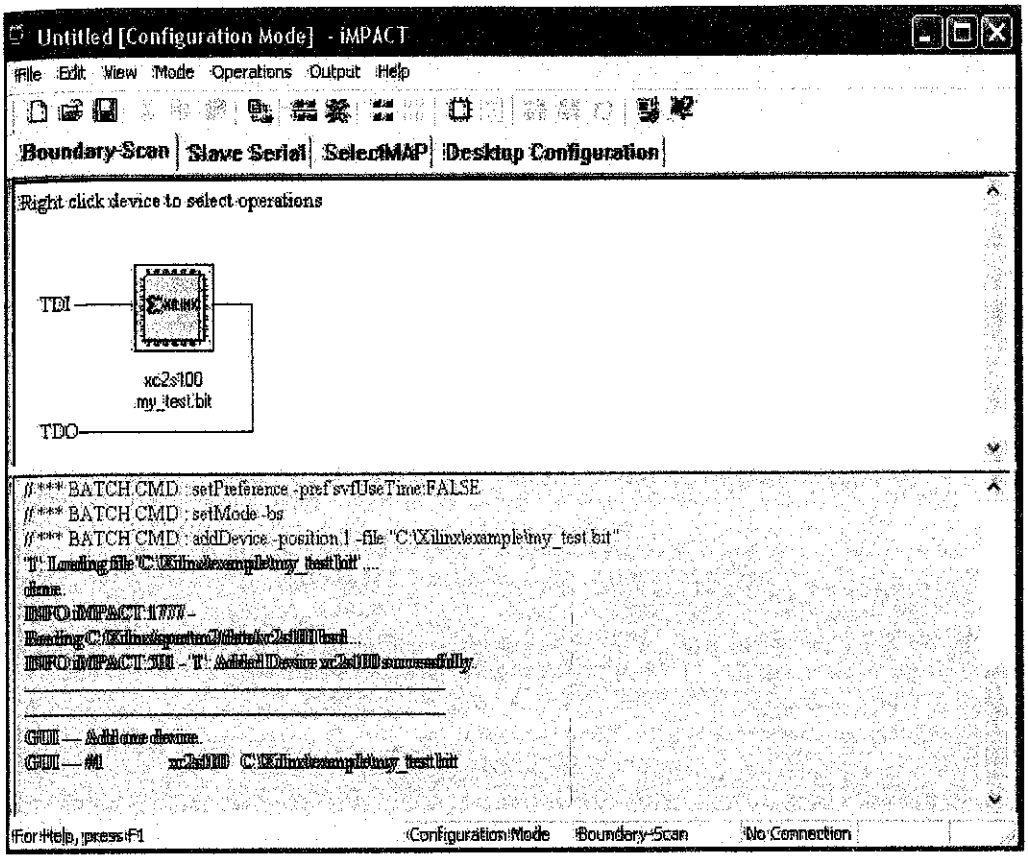
รูปที่ 4.19 Process Properties

4.7.2 เลือก **Configure Device (iMPACT)** ในหน้าต่าง *Processes* ของโปรแกรม WebPACK แล้ว กด **run** จะขึ้นหน้าต่าง **Configure Device** ขึ้นมา ให้กด **Next** จะปรากฏหน้าต่าง **Boundary-Scan Mode Selection** ขึ้นมา ให้คลิก **Finish** รอสักครู่แล้วโปรแกรมจะแจ้งว่าตรวจพบ Device เชื่อมต่ออยู่กับ Boundary-Scan Chain เมื่อกด **ok** แล้ว จะขึ้นหน้าต่างหนึ่งให้มา เพื่อให้เรา **Assign Configuration File** ในหน้าต่างจะมีอยู่ไฟล์หนึ่งที่มีชื่อเดียวกับ **Source Code** ของเราแต่เป็นนามสกุล **.bit** ให้เลือกไฟล์นั้นแล้วกด **ok**



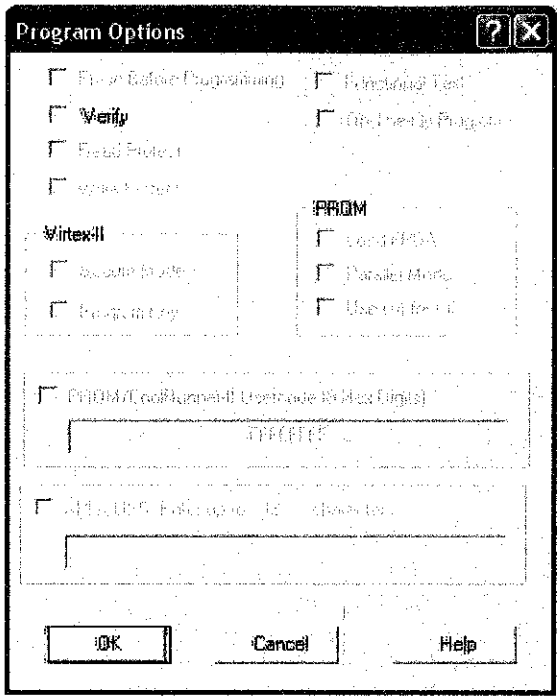
รูปที่ 4.18 Configure Devices

4.7.3 จากนั้นจะได้โปรแกรมหน้าตาดังรูปที่ 21 รายละเอียดของชิป และไฟล์ที่จะใช้ในการดาวน์โหลด จะปรากฏขึ้นมา (สำหรับกรณีที่ทำการ Create Programming File มาก่อน)



รูปที่ 4.19 JTAG Programmer

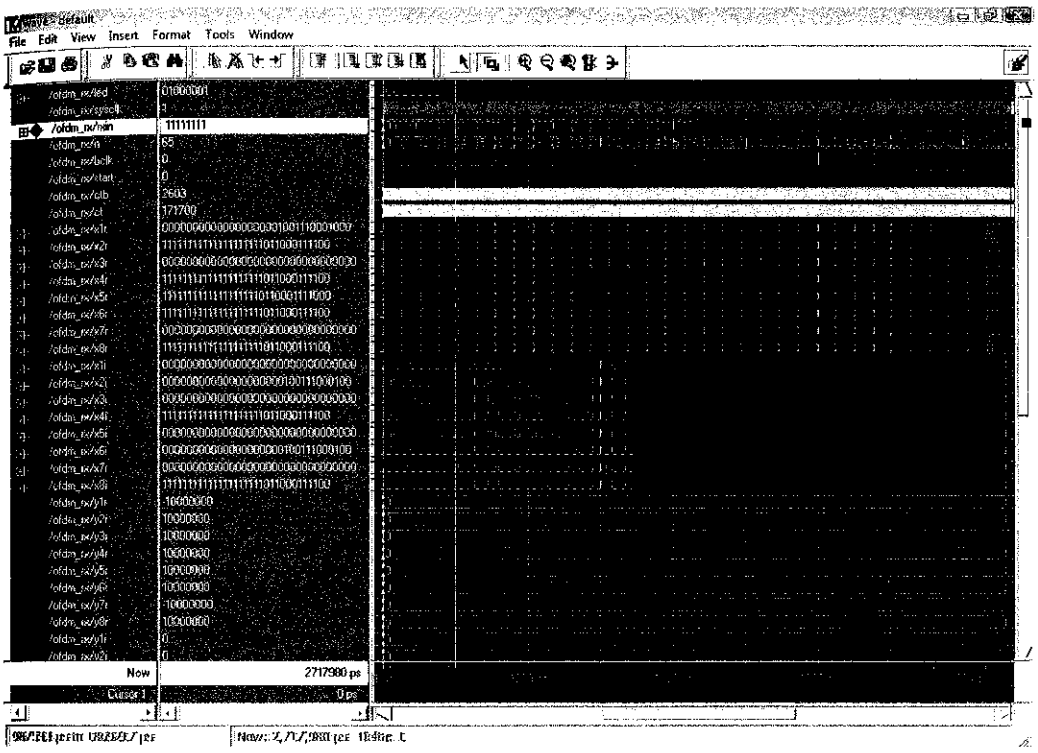
4.7.4 เริ่มการดาวน์โหลดไฟล์โดยคลิกขวาที่รูปชิปแล้วเลือก **Program** จะมีหน้าจอใหม่ปรากฏขึ้นมาคล้ายรูปที่ 23 ให้กด **OK** ก็จะเป็นการเริ่มการดาวน์โหลด เมื่อเสร็จแล้วจะมีหน้าต่างปรากฏขึ้นมาแจ้งให้ทราบ เป็นอันเสร็จสิ้นการออกแบบ



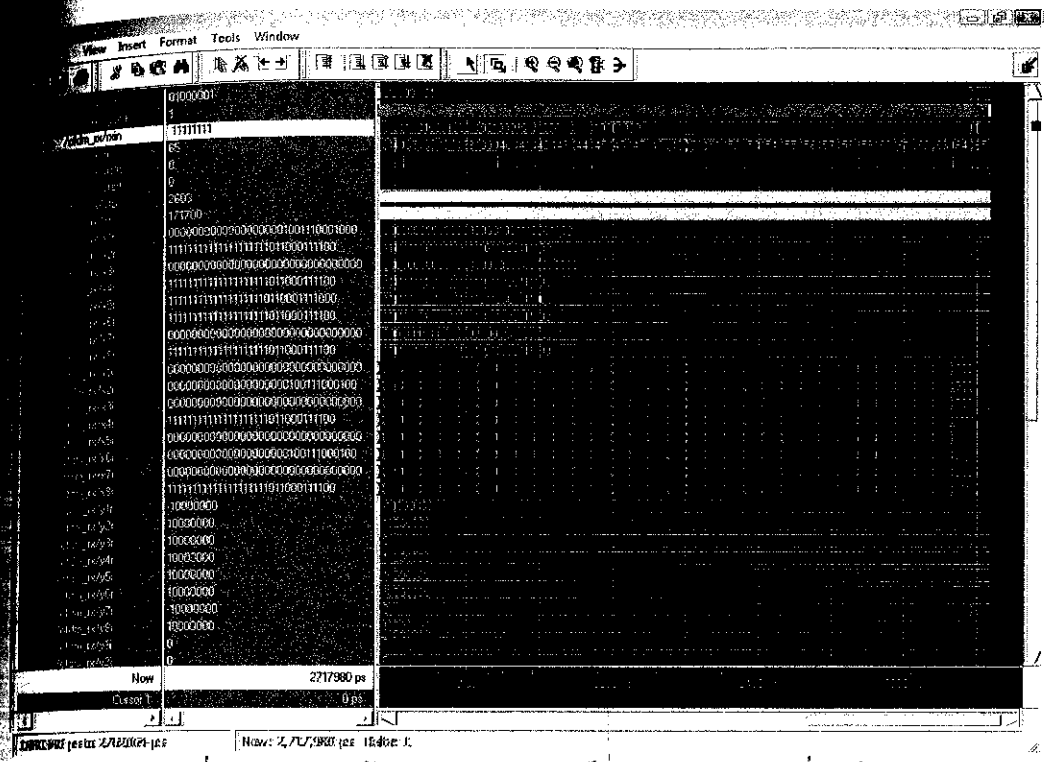
รูปที่ 4.20 Program Options

บทที่ 5
ผลการจำลองโปรแกรมและวิเคราะห์สรุป

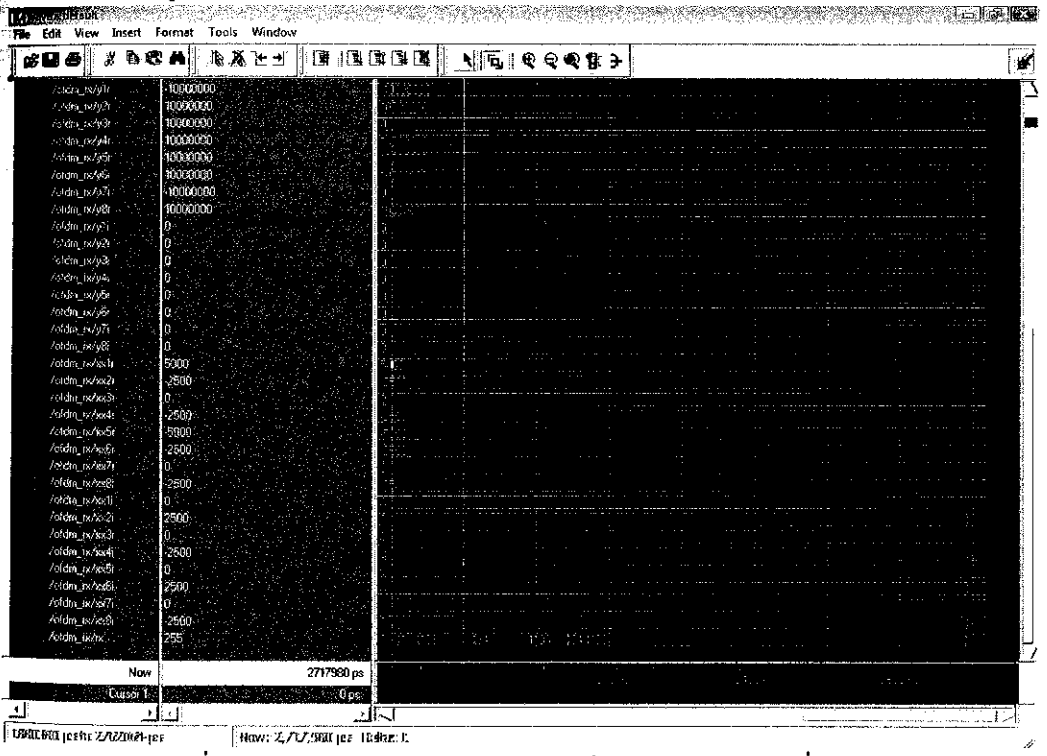
5.1 ภาครับ



รูปที่ 5.1 การจำลองสัญญาณขาออกจากบอร์ดประมวลผล FPGA ที่ภาครับ

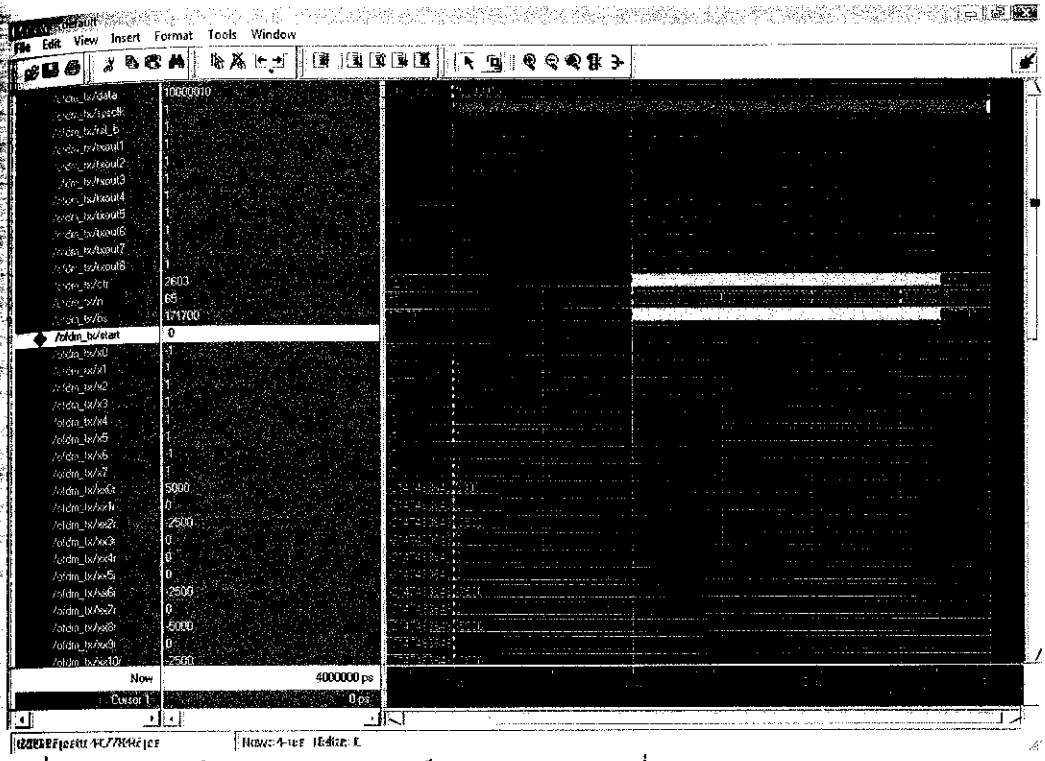


รูปที่ 5.2 การจำลองสัญญาณขาออกจากบอร์ดประมวลผล FPGA ที่ภาครับ

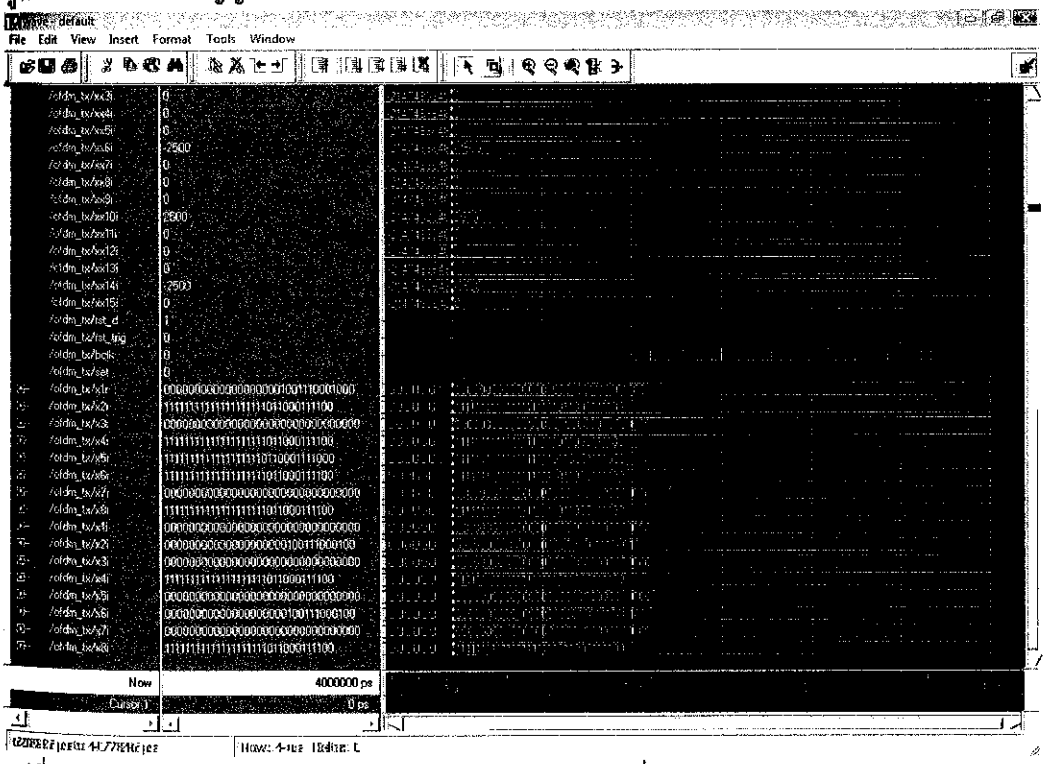


รูปที่ 5.3 การจำลองสัญญาณขาออกจากบอร์ดประมวลผล FPGA ที่ภาครับ

5.2 ภาคส่ง



รูปที่ 5.4 การจำลองสัญญาณขาออกจากบอร์ดประมวลผล FPGA ที่ภาคส่ง



รูปที่ 5.5 การจำลองสัญญาณขาออกจากบอร์ดประมวลผล FPGA ที่ภาคส่ง

5.3 บทสรุปของ OFDM

เนื่องจากการสื่อสารแบบไร้สายมักจะประสบกับปัญหาการจางหายของสัญญาณ (Fading) อันมีสาเหตุมาจากการแพร่กระจายคลื่นสัญญาณเป็นหลายวิถี (Multipath propagation) นอกจากนั้นคลื่นหลายวิถียังทำให้เกิดการรบกวนแบบแทรกสอดระหว่างสัญลักษณ์ (Inter Symbol Interference) หรือ ISI ของสัญญาณข้อมูลขึ้นที่เครื่องรับอีกด้วย ซึ่งถ้าเป็นระบบการมอดูเลตและส่งสัญญาณแบบเก่าๆ (การส่งข้อมูลแบบอนุกรม) จะแก้ไขปัญหาเหล่านี้ได้ค่อนข้างยาก โดยเฉพาะเมื่อเป็นการสื่อสารที่มีอัตราการส่งข้อมูลสูงๆ แต่สำหรับ OFDM แล้วปัญหาเหล่านี้สามารถแก้ไขได้ง่ายกว่าและยังสามารถส่งข้อมูลที่มีอัตราสูงๆ ได้อย่างสบาย

เนื่องจากในระบบ OFDM ข้อมูลที่เป็นอนุกรมความเร็วสูงจะถูกแปลงให้เป็นข้อมูลแบบขนานความเร็วต่ำเสียก่อน แล้วจึงส่งออกไปยังเครื่องรับพร้อมๆ กัน ซึ่งสามารถลดปัญหาเหล่านี้ลงได้ แล้วยังสามารถใช้งานแถบความถี่ในระบบที่เคยใช้สัญญาณพาหะเดี่ยวได้อย่างเต็มประสิทธิภาพ (spectral efficiency) ประมาณสองเท่าของการใช้สัญญาณพาหะเดี่ยวและสามารถป้องกันผลกระทบจากการเคลื่อนที่ของสัญญาณหลายเส้นทาง (immunity to muti-path) และมีความไวต่ำต่อการเลื่อนหายไปของความถี่ที่เลือก (less sensitivity to frequency selective fading)

5.4 ขีดจำกัดของโครงงาน

เนื่องด้วยช่วงเวลาที่ใช้ในการทำโครงงานนี้มีจำนวนจำกัด ซึ่งถ้าเทียบกับปริมาณงานแล้วถือว่าน้อยมาก เพราะโครงการนี้เป็นการเริ่มใหม่หมดสำหรับผู้วิจัยเอง เพราะทุกโปรแกรม ทุกๆงาน เป็นสิ่งใหม่ๆ ต้องการเรียนรู้ และบอร์ดประมวลผลเป็นบอร์ดใหม่ ที่ยังไม่เคยใช้ ยังไม่ทราบถึงขีดจำกัด ข้อดีและข้อด้อยต่างๆของบอร์ดประมวลผลนี้

5.5 ข้อเสนอแนะ

เนื่องจากการทำต้นแบบเครื่องรับส่งข้อมูลด้วยเทคโนโลยี OFDMบนบอร์ดประมวลผลสัญญาณเชิงดิจิตอลนี้ เป็นการทำเครื่องต้นแบบ ในตัวโปรแกรมเองอาจจะยังไม่สมบูรณ์ถ้าผู้ที่สนใจต้องการนำไปพัฒนาต่อหรือนำไปใช้งานจริง ควรจะพัฒนาโปรแกรมให้กระชับหรือสามารถใช้งานได้ดีกว่านี้ เพราะทางกลุ่มเขียนขึ้นมายังเป็นภาษาที่ง่ายๆไม่ซับซ้อน ดังนั้น ถ้านำไปใช้ในงานที่ซับซ้อนกว่านี้ควรปรับปรุงให้มีประสิทธิภาพมากกว่านี้

5.6 กิตติกรรมประกาศ

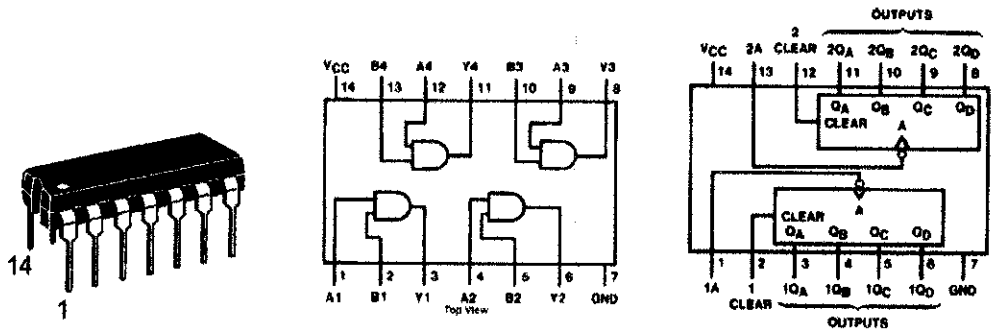
โครงการนี้ได้รับทุนอุดหนุนจาก สำนักงานกองทุนสนับสนุนการวิจัย ฝ่ายอุตสาหกรรม โครงการโครงงานอุตสาหกรรมสำหรับปริญญาตรี ประจำปี 2550 และขอขอบคุณ ห้างหุ้นส่วนจำกัด ภูมินทร์ซอฟต์ ที่ให้การสนับสนุนการทำโครงการนี้

ภาคผนวก ก

การออกแบบวงจรดิจิตอลด้วย FPGA และ CPLD

1.1 ไอซีมาตรฐาน

การออกแบบวงจรดิจิตอลขนาดเล็กโดยปกติจะนิยมใช้ชิพหรือไอซีมาตรฐาน เช่น ไอซี CMOS (อ่านว่าซีมอส) ตระกูล 4000 และ 74HC00 หรือไอซี TTL (อ่านว่าทีทีแอล) ตระกูล 74L500 เป็นต้น ตัวอย่างแพกเกจไอซีแสดงดังรูปที่ 1.1a) ไอซี TTL เช่น เบอร์ 74L508 (แอนด์เกต 2 อินพุต 4 ตัว) แสดงดังรูปที่ 1.1b) และ CMOS เช่น เบอร์ 74HC393 (วงจรนับไบนารีหรือฐานสอง 4 บิต 2 ตัว) แสดงดังรูปที่ 1.1c) จะเห็นได้ว่าไอซีสำเร็จรูปเหล่านี้จะมีฟังก์ชันทำงานทางลอจิกแบบตายตัวและเป็นวงจรขนาดเล็กอยู่ภายในเพียงไม่กี่ตัว จึงไม่เหมาะกับงานออกแบบวงจรขนาดใหญ่หรือความถี่สูงเนื่องจากเกิดเวลาล่าช้า (Delay) ขึ้นภายในตัวไอซีและสายสัญญาณ โดยที่ความเร็วของสัญญาณต่างๆในลายเส้นทองแดงของ PCB (ชนิด FR4) หรือสายสัญญาณนั้นจะมีความเร็ว (ค่าคงที่) ประมาณครึ่งหนึ่งของความเร็วแสงหรือ 15-18 เซนติเมตรต่อนาโนวินาที จากข้อจำกัดดังกล่าวทำให้การออกแบบแผงวงจรขนาดใหญ่ที่ใช้ความถี่สูงหลายสิบเมกะเฮิร์ตซ์มีความยุ่งยากมากและอาจทำไม่ได้



a) b) c)

รูปที่ 1.1 ตัวอย่างไอซีตระกูล TTL และ CMOS

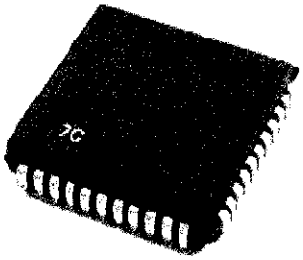
ต่อมาได้มีคิดค้นไอซีหรือชิพดิจิตอลลอเนกประสงค์ที่สามารถโปรแกรมให้มีฟังก์ชันการทำงานตามที่ต้องการได้ โดยที่ภายในชิพจะบรรจุวงจรลอจิกพื้นฐานที่มีฟังก์ชันทำงานแบบไม่ตายตัวไว้เป็นจำนวนมากเรียกว่า Programmable Logic Device = PLD (อ่านว่าพีแอลดี) เป็นผลสำเร็จประมาณปี 1970 ปัจจุบันได้มีการออกแบบวงจรขนาดใหญ่โดยใช้ชิพดิจิตอลลอเนกประสงค์แทนการออกแบบด้วยไอซีมาตรฐานตระกูล TTL หรือ CMOS กันมากขึ้น บ่อยครั้งเราจะพบชิพ FPGA (Field Programmable Gate Array = เอฟพีจีเอ) และ/หรือ CPLD (Complex Programmable Logic Device = ซีพีแอลดี) เป็นส่วนประกอบที่ติดตั้งอยู่บนแผงวงจร เช่น ทางด้านสื่อสาร การแพทย์ การทหาร ระบบเครือข่าย หรือ เครื่องมือวัดต่างๆ เป็นต้น

การที่เราออกแบบวงจรย่อยส่วนต่างๆรวมไว้ใน FPGA และ/หรือ CPLD เพียงตัวเดียวหรือเพียงไม่กี่ตัวได้นั้นจะทำให้แผงวงจรมีขนาดลดลงอย่างมาก ทำให้ได้วงจรที่ทำงานเร็วขึ้นเพราะสายสัญญาณต่างๆสั้นลงและสายสัญญาณส่วนใหญ่จะอยู่ภายในชิพทำให้ได้ผลิตภัณฑ์ที่มีขนาดเล็กกะทัดรัด

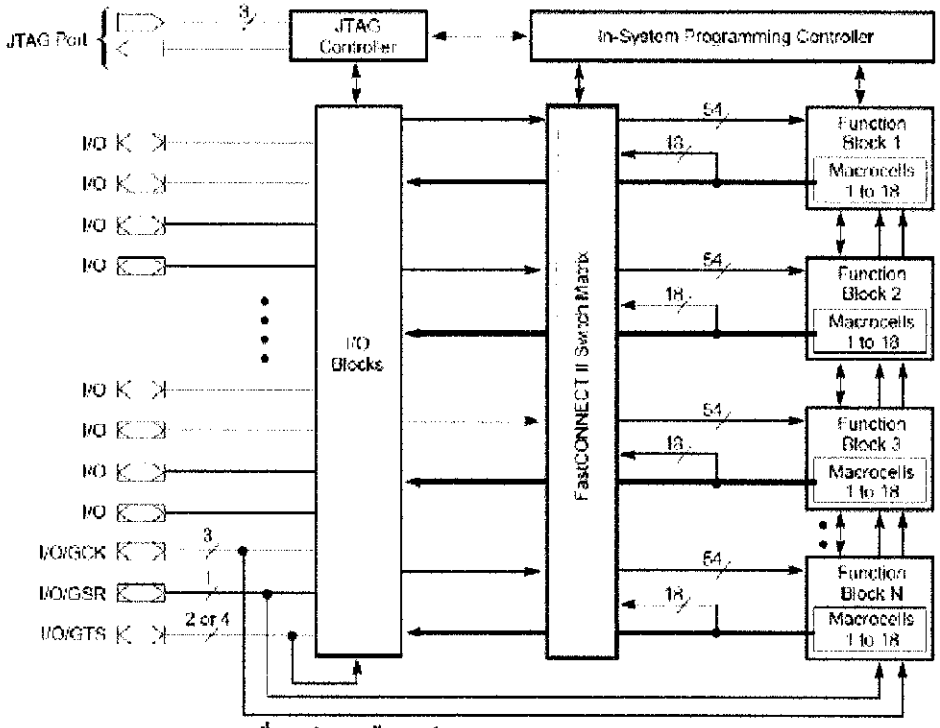
หนังสือเล่มนี้จะอธิบายการออกแบบสร้างวงจรดิจิตอลโดยใช้ FPGA และ CPLD ของบริษัท Xilinx เป็นหลัก ซึ่งผู้ใช้งานเริ่มต้นอาจไม่มีความจำเป็นต้องศึกษารายละเอียดโครงสร้างปลีกย่อยที่อยู่ภายในชิพมากนัก การออกแบบวงจรในลักษณะนี้มีความจำเป็นต้องใช้คอมพิวเตอร์และซอฟต์แวร์ทูล (Software Tool) ช่วยในการออกแบบ

1.2 CPLD

CPLD เป็นไอซีประเภท PLD ซึ่งเป็นชิพดิจิตอลลอเนกประสงค์ชนิดหนึ่งที่สามารถโปรแกรมให้มีฟังก์ชันทำงานตามที่ต้องการได้ ตัวอย่าง CPLD เบอร์ XC9536XL ที่มีความจุวงจร 800 เกตและมี 34 อินพุตเอาต์พุต (I/O) แสดงดังรูปที่ 1.2 โครงสร้างภายในชิพ CPLD ตระกูล XC9500XL แสดงดังรูปที่ 1.3

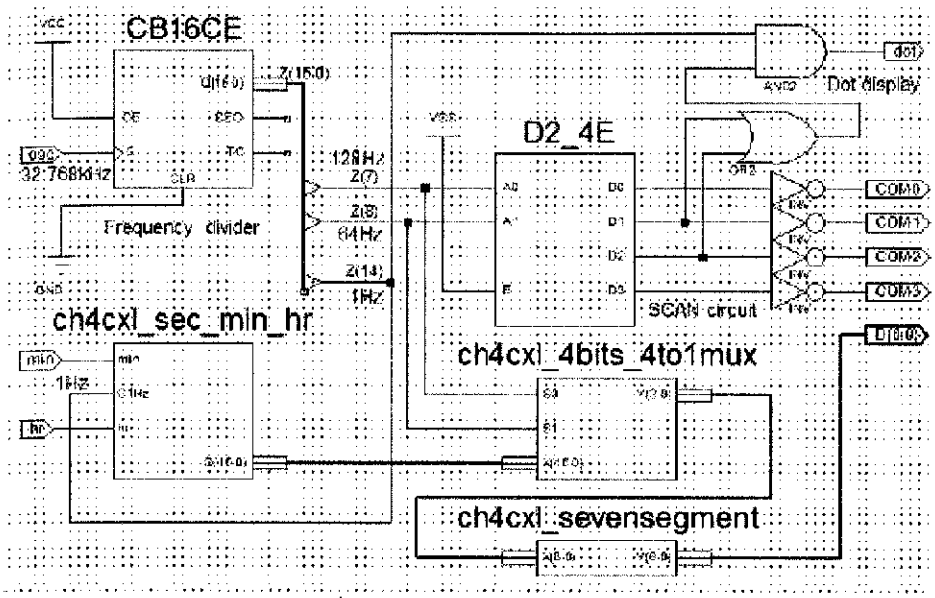


รูปที่ 1.2 ตัวอย่าง CPLD เบอร์ XC9536XL ที่มีความจุวงจร 800 เกต



รูปที่ 1.3 โครงสร้างภายในของ CPLD ตระกูล XC9500XL

โครงสร้าง CPLD ประกอบส่วนหลักๆ คือ Function Block (FB) และ I/O Block (IOB) ที่สามารถเชื่อมต่อถึงกันด้วย Switch matrix ภายใน Function Block จะประกอบไปด้วยวงจรลอจิกพื้นฐานต่างๆที่สามารถโปรแกรมเป็นวงจรดิจิตอลได้ตามต้องการ ส่วน I/O Block จะทำหน้าเป็นบัฟเฟอร์ที่อินพุตหรือเอาต์พุตของตัวชิพ วงจร In-system programming ใช้สำหรับโปรแกรมชิพผ่านทางพอร์ต JTAG (อ่านว่าเจแท็ก)โดยมี JTAG Controller เป็นตัวควบคุม ตัวอย่างวงจรนาฬิกาดิจิตอลดังรูปที่ 1.4 นั้นอาจสร้างได้ด้วยไอซีมาตรฐานตระกูล 7400 ไม่น้อยกว่า 10 ตัว แต่เมื่อสร้างวงจรนี้ด้วย CPLD เบอร์ XC9572XL จะกินความจุวงจรสูงสุดเพียง 70% เท่านั้นซึ่งมีรายละเอียดแสดงดังรูปที่ 1.5 แม้ว่า CPLD จะสามารถสร้างวงจรดิจิตอลทั่วๆไปได้มากมาย แต่ต้องถือว่า CPLD นั้นมีความจุวงจรต่ำมากเมื่อเทียบกับ FPGA ซึ่งจะอธิบายในหัวข้อต่อไป โดยทั่วๆไป CPLD จะมีความจุวงจรไม่เกิน 10,000 เกต ข้อจำกัดของ CPLD คือใช้เวลาในการโปรแกรมค่อนข้างนานเนื่องจากมีโครงสร้างตัวเก็บข้อมูลภายในจำพวก EEPROM ซึ่งเขียนข้อมูลลงไปได้ช้าและราคาแพง (ราคาต่อเกตแพงกว่า FPGA) แต่เมื่อโปรแกรมวงจรไว้ใน CPLD แล้วข้อมูลวงจรนั้นก็จะคงอยู่แม้ว่าจะไม่มีไฟเลี้ยงตัวชิพแล้วก็ตาม การโปรแกรมสามารถทำซ้ำได้หลายๆ ครั้ง



รูปที่ 1.4 ตัวอย่างวงจรนาฬิกาดิจิตอล

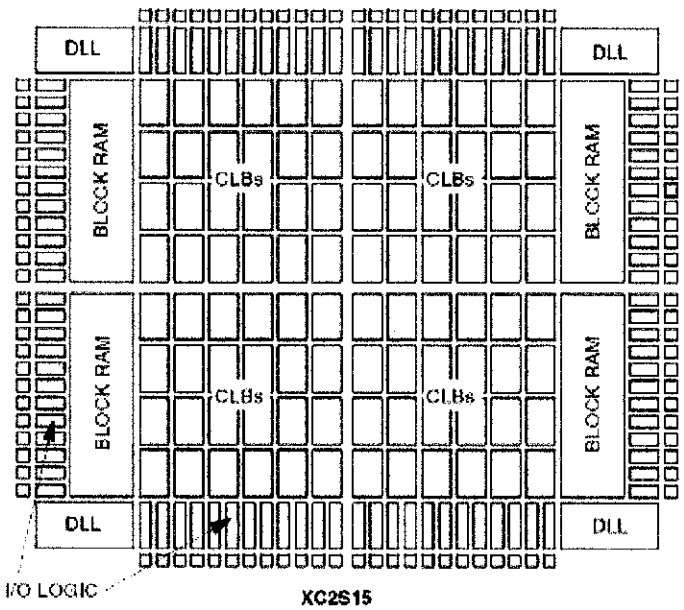
RESOURCES SUMMARY

Macrocells Used	Pterms Used	Registers Used	Pins Used	Function Block Inputs Used
50/72 (70%)	185/360 (52%)	37/72 (52%)	15/34 (45%)	98/216 (46%)

รูปที่ 1.5 แสดงรายการอุปกรณ์ที่ต้องใช้ใน CPLD

1.2 FPGA

FPGA เป็นชิพประเภท PLD เช่นเดียวกัน จึงเป็นชิพอเนกประสงค์ที่สามารถโปรแกรมให้เป็นวงจรดิจิตอลได้ตามที่ต้องการ แต่จะมีโครงสร้างภายในแตกต่างจาก CPLD อย่างสิ้นเชิงและซับซ้อนกว่ามาก ตัวอย่างโครงสร้างภายในของ FPGA ตระกูล Spartan-II เบอร์ XC2S15 แสดงดังรูปที่ 1.6

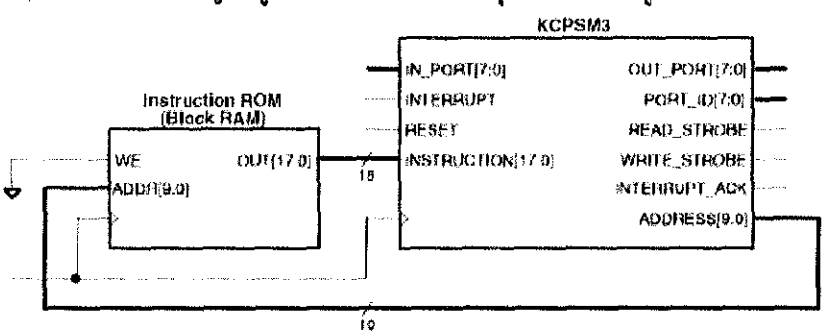


รูปที่ 1.6 โครงสร้างของ FPGA Spartan-II เบอร์ XC2S15

ซึ่งโครงสร้างภายในของ FPGA ประกอบ 2 ส่วนหลักๆ คือ Configurable Logic Block (CLB) ต่างๆ และ I/O LOGIC เพื่อใช้โปรแกรมให้เป็นวงจรดิจิตอลตามที่ต้องการ การเชื่อมต่อถึงกันภายในชิพมีหลากหลายลำดับชั้น นอกจากนี้ยังมีอุปกรณ์ภายในที่ทำหน้าที่เฉพาะงาน เช่น Delay-Locked Loop (DLL) และหน่วยความจำ (Block RAM) เพื่ออำนวยความสะดวกในการออกแบบและเพิ่มสมรรถนะให้วงจร FPGA มีความจุของเกตค่อนข้างสูงตั้งแต่ระดับ 10,000 เกตถึงประมาณในหลักสิบล้านเกตขึ้นไปขึ้นอยู่กับเทคโนโลยีที่ใช้ในการผลิต การโปรแกรม FPGA สามารถทำได้โดยการโหลดข้อมูลวงจร (Configuration data) ลงไปเก็บที่เซลหน่วยความจำแบบ RAM (เป็นคนละส่วน Block RAM) ที่อยู่ภายใน FPGA ดังนั้น FPGA จึงไม่มีข้อจำกัดในการโปรแกรมซ้ำและสามารถโปรแกรมข้อมูลวงจรลงไปได้เร็ว แต่ RAM มีข้อเสียที่สำคัญคือข้อมูลวงจรจะสูญหายหากไม่มีไฟเลี้ยง จึงต้องใช้หน่วยความจำภายนอกชิพที่สามารถเก็บข้อมูลวงจรอยู่ได้แม้ว่าจะไม่มีไฟเลี้ยงตัวชิพ เช่น Serial PROM เพื่อใช้เก็บข้อมูลและจะทำการโหลดข้อมูลจาก Serial PROM ลงชิพ FPGA อย่างอัตโนมัติทุกครั้งที่มีการจ่ายไฟเลี้ยง

1.3 ข้อเปรียบเทียบระหว่าง FPGA และ CPLD กับไมโครคอนโทรลเลอร์

นักออกแบบวงจรดิจิตอลส่วนใหญ่จะคุ้นเคยกับการใช้งานไมโครคอนโทรลเลอร์กันดีอยู่แล้ว เช่น ตระกูล MCS-51 และตระกูล PIC เป็นต้น ซึ่งมีการออกแบบสถาปัตยกรรมและชุดคำสั่งไว้ภายในเรียบร้อยแล้ว จึงสามารถนำชุดคำสั่งต่างๆมาโปรแกรมให้ไมโครคอนโทรลเลอร์ทำงานได้ตามที่ต้องการ ซึ่งต่างจาก FPGA และ CPLD ที่จะนำไปใช้กับงานด้านออกแบบฮาร์ดแวร์ดิจิตอลเป็นหลักโดยเฉพาะอย่างยิ่งงานที่เป็นวงจรขนาดใหญ่หรือความถี่สูงๆ ซึ่งไมโครคอนโทรลเลอร์เองก็สามารถสร้างโดยใช้ FPGA ได้เช่นกัน ปัจจุบันได้มีการออกแบบไมโครคอนโทรลเลอร์แบบฝังตัว (Embedded microcontroller) และวงจรดิจิตอลส่วนต่างๆรวมไว้ใน FPGA ทำให้ได้วงจรที่ทำงานได้เร็วและมีความยืดหยุ่นสูง ตัวอย่างการออกแบบไมโครคอนโทรลเลอร์แบบฝังตัวไว้ใน FPGA ตระกูล Spartan-3 เบอร์ XC3S200 แสดงดังรูปที่ 1.7 โดยจะกินพื้นที่วงจรใน FPGA ประมาณ 4-5% และสามารถ RUN ที่ความถี่สูงมากถึง 87 MHz ซึ่งไมโครคอนโทรลเลอร์ที่สร้างจาก FPGA จะมีความเร็วสูงกว่าไมโครคอนโทรลเลอร์ทั่วไปที่มีขายในท้องตลาดประมาณ 2-20 เท่าและสร้างไว้ใน FPGA ได้หลายๆตัว การออกแบบวงจรดิจิตอลจำนวนมากไว้ใน FPGA ทำให้แผงวงจรมีขนาดเล็กลงอย่างมาก วงจรจะทำงานได้เร็วขึ้นและมีความเชื่อถือได้สูงเพราะสายสัญญาณต่างๆจะสั้นลงและส่วนใหญ่จะอยู่ภายในชิพ FPGA โดยไม่มีจุดเชื่อมวงจรต่ออยู่ภายนอก



รูปที่ 1.7 ตัวอย่างการออกแบบไมโครคอนโทรลเลอร์ตระกูลพิโคเบลซฝังตัวใน FPGA ตระกูล Spartan-3

1.4 กระบวนการออกแบบวงจร

กระบวนการออกแบบวงจรโดยทั่วไปมักจะมีขั้นตอนหรือการทำงานเป็นขั้นตอนดังนี้

- System requirement
- System design
- System implementation
- Testing and debugging
- Documentation

กระบวนการทำ System requirement คือ การหาความต้องการว่าเราต้องการสร้างวงจรอะไร วงจรทำงานอย่างไรและมีอินพุตเอาต์พุตอะไรบ้าง กระบวนการทำ System design คือ การออกแบบและหาวิธีแก้ไขปัญหาจากขั้นตอนแรก จากนั้นจึงเป็นการทำ System implementation เพื่อสร้างวงจรตามที่ได้ออกแบบไว้ แล้วจึงทำการเพื่อทดสอบ (Testing) และหากมีปัญหาก็ทำการแก้ไข (Debugging) จากนั้นจึงทำ Documentation เพื่อสร้างเอกสารอธิบายการทำงานของวงจรที่ออกแบบทั้งหมด

การออกแบบวงจรดิจิตอลด้วย FPGA หรือ CPLD นั้นจะเริ่มจากขั้นตอน System Implementation จนถึง Testing and debugging ซึ่งจำเป็นต้องใช้ซอฟต์แวร์ทูล (Software tool) ช่วยในการออกแบบ ซอฟต์แวร์ทูลของบริษัท Xilinx เวอร์ชันล่าสุดคือ ISE 9.1i หนังสือเล่มนี้จะใช้ ISE 8.1i เป็นหลักเนื่องจากการใช้งานต่างๆเกือบเหมือนกับ ISE 9.1i ทุกประการ แต่ ISE 9.1i จะรองรับ FPGA ตระกูลล่าสุดได้แก่ Vertex-5 และ Spartan-3A เป็นต้น รวมทั้งคุณสมบัติใหม่ๆบางประการที่เราแทบจะไม่ได้ใช้และที่สำคัญคือ ISE9.1i จะใช้หน่วยความจำมากกว่า ISE8.1i ประมาณ 2 เท่า ISE 8.1i มี 2 ตัวด้วยกันคือ ISE WebPACK 8.1i (ไอเอสอีเวบแพค 8.1i) หรือ WebPACK 8.1i (เป็น Limited version และอนุญาตให้ดาวน์โหลดฟรีทางอินเตอร์เนต) และ ISE Foundation 8.1i หรือ Foundation 8.1i (ต้องซื้อ) โดยที่ ISE 8.1i สามารถ RUN บนระบบปฏิบัติการต่างๆได้ดังตารางที่ 1.1 และตารางที่ 1.2 โดยตระกูลชิพที่ใช้ได้กับซอฟต์แวร์ทูลนี้สรุปดังตารางที่ 1.3 ขนาด RAM อย่างน้อยที่สุดของคอมพิวเตอร์ที่แนะนำให้ใช้กับ WebPACK 8.1i และ Foundation 8.1i สรุปได้ดังตารางที่ 1.4 คือต้องไม่น้อยกว่า 256 MB. ส่วนในกรณีที่ใช้ WebPACK 9.1i และ Foundation 9.1i นั้นต้องใช้ RAM ไม่น้อยกว่า 512 MB.

ตารางที่ 1.1 ระบบปฏิบัติการที่ใช้กับ ISE Foundation 8.1i

ISE Foundation™
Microsoft Windows 2000 / XP
Sun Solaris 2.8 or 2.9
Red Hat Enterprise Linux 3 (32 & 64 bit)

ตารางที่ 1.2 ระบบปฏิบัติการที่ใช้กับ ISE WebPACK 8.1i

ISE WebPACK™
Microsoft Windows 2000 / XP
Red Hat Enterprise Linux 3 (32 bit)

ตารางที่ 1.3 ตระกูลของชิพ FPGA และ CPLD ที่ซอฟต์แวร์ทูลรองรับ

Family	ISE *WebPack* (MS Windows & Linux)	ISE *Foundation* (MS Windows, Linux, Linux64, & Solaris)
Virtex devices	Up to V600	All
Virtex-E devices	Up to V600E	All
Virtex-II devices	Up to 2V500	All
Virtex-II Pro devices	2VP2, 2VP4, 2VP7	All
Virtex-II Pro X devices	No	All

ตารางที่ 1.3 ตระกูลของชิพ FPGA และ CPLD ที่ซอฟต์แวร์ทูลรองรับ (ต่อ)

Family	ISE *WebPack* (MS Windows & Linux)	ISE *Foundation* (MS Windows, Linux, Linux64, & Solaris)
Virtex-4 devices	4VLX15, 4VLX25, 4VSX25, 4VFX12	All
QPro Virtex Hi-Rel devices	All	All
QPro Virtex® Hi-Rel devices	All	All
QPro Virtex2 Hi-Rel devices	All	All
QPro Virtex Rad-Hard devices	All	All
QPro Virtex2 Rad-Hard devices	All	All
Spartan-II devices	All	All
Spartan-IIE devices	All	All
Automotive Spartan-IIE devices	All	All
Spartan-3 devices	All	All
Automotive Spartan-3 devices	All	All
Spartan-3E devices	Up to 3S500E	All
XC9500/XL/XV devices	All	All
Automotive 9500XL devices	All	All
CoolRunner-II devices	All	All
Automotive CoolRunner-II devices	All	All
Device Programming Support	All	All

ตารางที่ 1.4 ขนาด RAM อย่างน้อยที่สุดของคอมพิวเตอร์ที่แนะนำให้ใช้กับซอฟต์แวร์ทูล

Xilinx Device	RAM
• XC9500™/XL/XV • Automotive 9500XL • CoolRunner™/ CoolRunner™II • Automotive CoolRunner™ II • Spartan™-II XC2S15 through XC2S200 • Spartan™-IIE XC2S50E through XC2S600E • Automotive Spartan™-IIE AC2S50E through AC2S300E • Spartan™-3 XC3S50 through XC3S1000 • Automotive Spartan™-3 A3S50 through A3S1000 • Spartan™-3E XC3S100E through XC3S1200E • Virtex™ XCV50 through XCV800 • Virtex™-E XCV50E through XCV600E • Virtex™-II XC2V40 through XC2V1000 • Virtex™-II PRO XC2VP2 through XC2VP7 • Virtex™-4 ♦ XC4VLX15 through XC4VLX25 ♦ XC4VFX12 through XC4VFX20 ♦ XC4VSX25	256-512 Megabytes
• Spartan™-3 XC3S1500 through XC3S5000 • Virtex™ XCV1000 • Virtex™-E XCV1000E through XCV3200E • Virtex™-II XC2V1500 through XC2V6000 • Virtex™-II PRO XC2VP20 through XC2VP70 • Virtex™-II PRO X XC2VPX20 through XC2VPX70 • Virtex™-4 ♦ XC4VLX40 through XC4VLX100 ♦ XC4VFX40 through XC4VFX100 ♦ XC4VSX35 through XC4VSX55	1-2 Gigabyte

แม้ว่า WebPACK 8.1i จะเป็น Limited version ของ Foundation 8.1i ตามที่สรุปในตารางที่ 1.3 และตารางที่ 1.5 แต่ก็สามารถระดับเดียวกับ ISE BaseX ในเวอร์ชั่นก่อนๆซึ่งถือว่าเพียงพอสำหรับการออกแบบวงจรขนาดใหญ่ทั่วๆไป โดยที่สามารถที่เพิ่มเข้ามาคือ ISE Simulator, CORE Generator และ FPGA editor

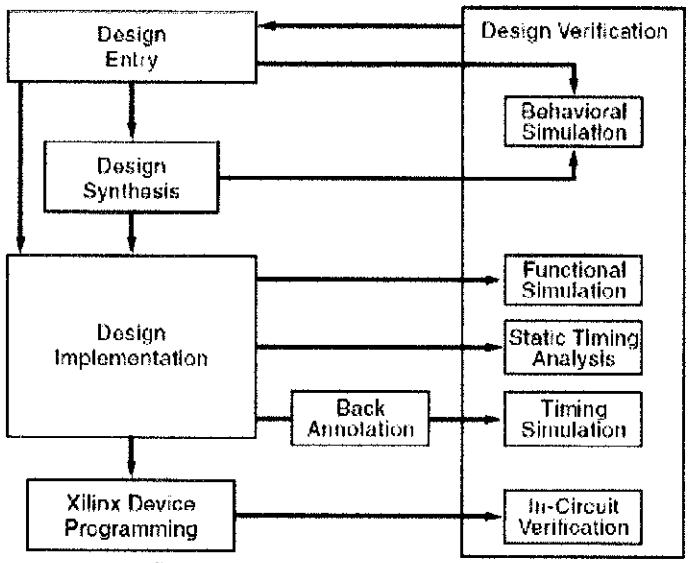
ตารางที่ 1.5 คุณสมบัติและความสามารถที่ซอฟต์แวร์ทูลรองรับ

Feature	ISE *WebPACK* (MS Windows & Linux)	ISE *Foundation* (MS WINDOWS, Linux, Linux64, & Solaris)
Schematic Editor	Yes	Yes
State Diagram Entry	MS Windows Only	MS Windows Only
XST Synthesis	Yes	Yes
Synplify/Synplify Pro Integration	Yes	Yes
Leonardo Spectrum Integration	Yes	Yes
Precision Integration	Yes	Yes
Waveform Editor	Yes	MS Windows & Linux Only
ISE Simulator	Limited version included	MS Windows & Linux Only Limited version included Unlimited version available as option
ModelSim Xilinx Edition	MS Windows only Starter included Full MXE available as option	MS Windows only Starter included Full MXE available as option
ModelSim Integration	Yes	Yes
CORE Generator	Yes	Yes
Floorplanner	Yes	Yes
FPGA Editor	Yes	Yes
Device Programming Support	Yes	Yes

การจำลองการทำงานของวงจรนั้นสามารถเลือกใช้ซอฟต์แวร์ทูลได้ 2 ตัวคือ ISE Simulatorและ ModelSim แต่หนังสือเล่มนี้จะเน้นเฉพาะ ISE Simulator เท่านั้น ซึ่งซอฟต์แวร์ทูลนี้ติดตั้งมาพร้อมกับ ISE WebPACK 8.1i-9.1i และ ISE Foundation 8.1i-9.1i อยู่แล้ว

1.8 ขั้นตอนการออกแบบวงจรดิจิตอลด้วย FPGA หรือ CPLD

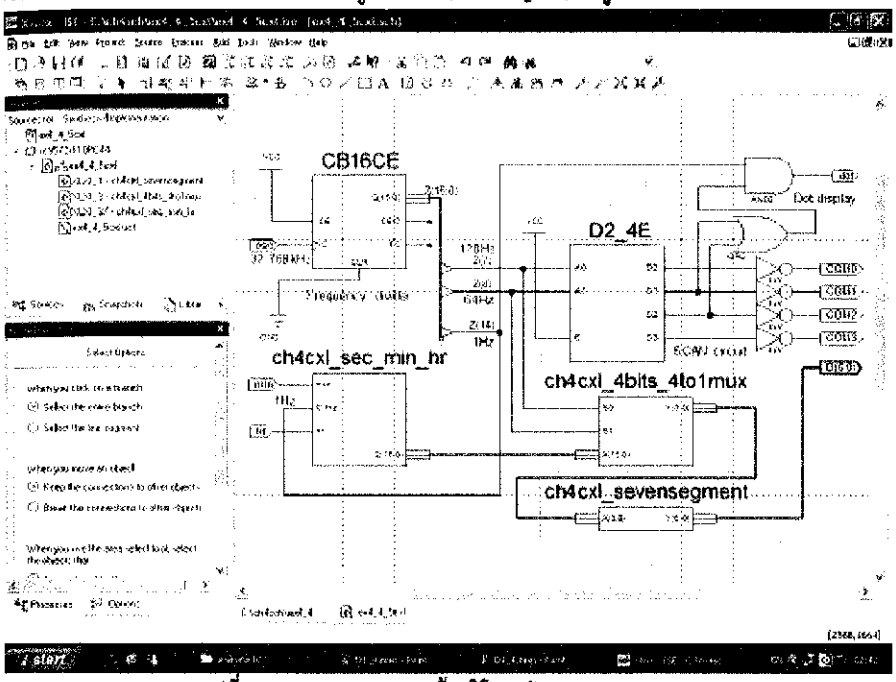
ขั้นตอนการออกแบบวงจรดิจิตอลด้วย FPGA หรือ CPLD มีการทำงานหลายขั้นตอนแสดงรายละเอียดดังรูปที่ 1.8



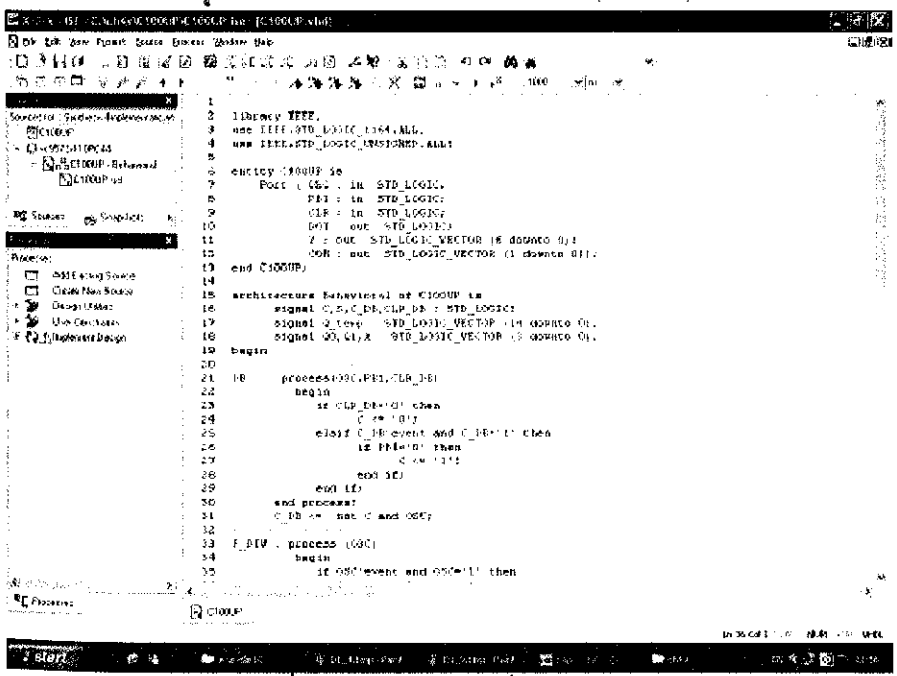
รูปที่ 1.8 ขั้นตอนการออกแบบวงจรดิจิตอลด้วย FPGA หรือ CPLD

1.6.1 การออกแบบวงจร

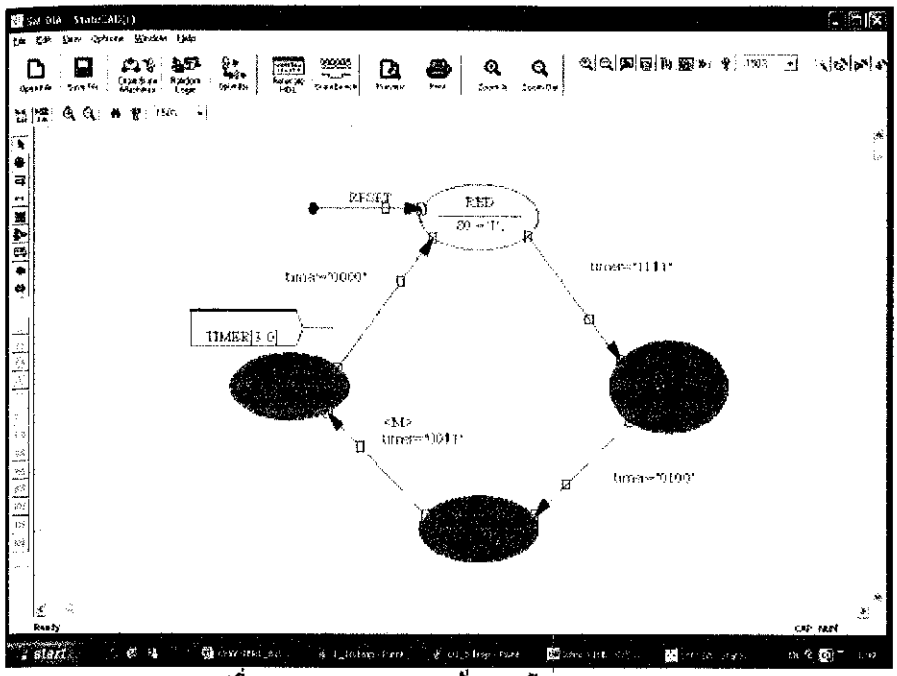
การออกแบบวงจร (Design entry) นั้นสามารถเลือกออกแบบด้วยวิธีการต่างๆได้ เช่น วิธีวาดผังวงจร (Schematic) ดังรูปที่ 1.9 หรือออกแบบด้วยภาษาระดับสูงที่เรียกว่า HDL (Hardware Description Language) เช่น VHDL (หรือ Verilog) ดังรูปที่ 1.10 หรืออาจออกแบบวงจรด้วยวิธีการเขียนแผนภูมิสถานะ (State diagram) ดังรูปที่ 1.11 เพื่อให้ได้วงจรตามที่ต้องการ



รูปที่ 1.9 การออกแบบวงจรด้วยวิธีวาดผังวงจร (Schematic)



รูปที่ 1.10 การออกแบบวงจรด้วยภาษา VHDL



รูปที่ 1.11 การออกแบบวงจรด้วยการเขียน State diagram

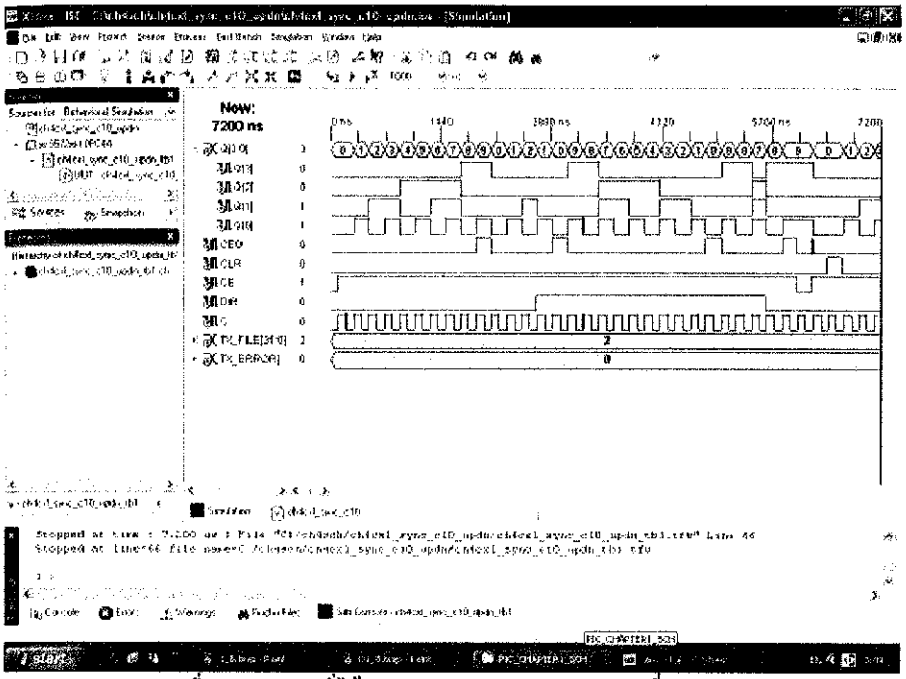
1.5.2 การสังเคราะห์วงจร

การสังเคราะห์วงจร (Design synthesis) คือขั้นตอนการแปลงโค้ดที่ใช้ออกแบบวงจรด้วยภาษาระดับสูงที่เรียกว่าภาษา HDL (Hardware Description Language) ให้เป็นวงจรในระดับเกต (Gate-level) โดยใช้ Xilinx Synthesis Tool หรือ XST โดยวงจรจะถูกเขียนอธิบายอยู่ในรูป Text file ที่เรียกว่า Netlist ที่ระบุรายการอุปกรณ์ (component) ระดับเกตและระบุรายการที่สายสัญญาณ (Net) แต่ละเส้นที่เชื่อมต่อถึงกัน ส่วนการออกแบบวงจรด้วยวิธีวาดผังวงจรซึ่งเป็นระดับเกตอยู่แล้วนั้นจะถูกเขียนอธิบายอยู่ในรูปไฟล์ Netlist เช่นกัน ซึ่งอาจจะอยู่ในรูปไฟล์ Xilinx Netlist Format (XNF) หรือ EDIF (อ่านว่า อิดีฟ) ที่เป็น Industry standard ก็ได้ การสังเคราะห์วงจรนั้นจะต้องมีการระบุว่าเป็น FPGA หรือ CPLD ระบุเบอร์ชิพเพื่อปรับค่าพารามิเตอร์ต่างๆให้กับทูลสังเคราะห์วงจร (Synthesis Tool) ซึ่งสามารถเลือก Optimize โดยเน้นความเร็วหรือพื้นที่ (ประหยัดจำนวนเกต)

1.5.3 การตรวจสอบความถูกต้องของวงจร

การตรวจสอบความถูกต้องของวงจรที่ออกแบบ (Design verification) เป็นการนำโค้ด HDL ของวงจรที่ออกแบบไปตรวจสอบความถูกต้องด้วยโปรแกรมจำลองการทำงาน (Simulation) โดยใช้ซอฟต์แวร์ทูล เช่น ISE Simulator หรือ ModelSim ตัวอย่างสัญญาณที่ได้จากการจำลองการทำงานแสดงดังรูปที่ 1.12 การจำลองการทำงานโดยทั่วไปมี 3 ระดับคือ

- Behavioral simulation เป็นการจำลองเฉพาะพฤติกรรมของวงจรโดยยังไม่คิดถึงโครงสร้างของวงจรภายในเพื่อให้ได้แบบจำลองการทำงานเบื้องต้น ซึ่งโค้ด HDL ที่เขียนนี้อาจจะนำไป Synthesis ไม่ได้แต่เขียนโค้ดได้รวดเร็ว
- Functional simulation เป็นการนำโค้ดในระดับ RTL (Register-transfer-level) ซึ่งเป็นระดับที่สามารถนำไปสังเคราะห์วงจรได้มาทำการจำลองการทำงานเพื่อตรวจสอบทำงานวงจรให้ถูกต้องก่อนนำไปสังเคราะห์วงจร
- Timing simulation เป็นการจำลองการทำงานที่ใกล้เคียงกับ Hardware จริงมากที่สุด เนื่องจากเป็นการนำข้อมูล Timing ที่เกิดขึ้นใน FPGA มาใช้ ทำให้สามารถตัดสินใจเลือกเบอร์ FPGA ได้ถูกต้องก่อนตัดสินใจซื้อ



รูปที่ 1.12 สัญญาณที่ได้จากการจำลองการทำงานของวงจรที่ออกแบบ

โปรแกรมที่ช่วยสร้างชุดข้อมูลที่ใช้ในการจำลองการทำงาน คือ ซอฟต์แวร์ HDL Bencher ที่ผู้ใช้สามารถสั่งป้อนสัญญาณขาของอินพุตต่างๆ ให้เป็นค่าตามที่ต้องการได้ในรูปแบบที่เรียกว่า Test bench waveform (ไฟล์ .tbw) ก่อนจะเรียกใช้งาน ISE Simulator หรือ ModelSim เพื่อแสดงผลลัพธ์ให้ผู้ทดสอบทราบในรูปของ Waveform ซึ่งในหนังสือเล่มนี้เราจะใช้ HDL Bencher ที่มีอยู่ในซอฟต์แวร์ชุด ISE Simulator และติดตั้งมาพร้อมกับ ISE 8.1i-9.1i อยู่แล้วมาใช้ในการจำลองการทำงาน

1.5.4 Design implementation

ในกรณีของ FPGA ขั้นตอนนี้เริ่มจากขั้นตอนการแปล (Translate) โดยนำไฟล์ Design netlist มาทำการออพติไมซ์วงจรเสียก่อน จากนั้นจะนำไปการตรวจสอบว่าสามารถวางหรือบรรจุวงจรลงในชิพเบอร์ที่เรากำหนดได้หรือไม่ ขั้นตอนต่อไปเป็นการแมพ (Map) โดยเลือกอุปกรณ์จากไฟล์ของวงจรเข้าไปวางในอุปกรณ์พื้นฐานต่างๆที่อยู่ภายใน FPGA เบอร์ที่เรากำหนด โดยจะต้องมีการคำนวณหาตำแหน่งที่เหมาะสมเพื่อที่จะนำอาวงจรไปวาง (Place) และเมื่อวางเรียบร้อยแล้วจึงเชื่อมต่อสัญญาณต่างๆเข้าด้วยกัน (Route) ตามลำดับโดยใช้ซอฟต์แวร์ชุด ซึ่งข้อมูลวงจรที่ใช้ในขั้นตอนนี้อาจได้มาจาก Design netlist ที่ได้มาจาก Design Entry ที่เป็น Schematic หรือจากการสังเคราะห์วงจรในกรณีที่เป็น HDL ส่วนกรณีของ CPLD นั้นจะมีความซับซ้อนน้อยกว่ามากจึงเรียกรวมขั้นตอนหลังจากการแปล (Translate) ว่าขั้นตอน Fitting

จากนั้นจะเป็นขั้นตอน Bitstream generation เพื่อสร้างไฟล์ให้เหมาะสำหรับการดาวน์โหลดลง FPGA หรือ CPLD โดยจะได้เป็นไฟล์ที่นามสกุล .bit (ไฟล์ข้อมูลวงจรของ FPGA) หรือไฟล์ที่นามสกุล .jed (ไฟล์ข้อมูลวงจรของ CPLD)

1.5.5 การโปรแกรมข้อมูลวงจรลงชิพ

การโปรแกรมข้อมูลวงจรลงชิพ (Device Programming) นั้นสามารถทำได้โดยนำไฟล์ที่นามสกุล .bit (ไฟล์ข้อมูลวงจรของ FPGA) หรือไฟล์ที่นามสกุล .jed (ไฟล์ข้อมูลวงจรของ CPLD) ที่ได้ในขั้นตอน Implementation หรือขั้นตอน Generate Programming File มาดาวน์โหลดลงชิพโดยใช้เครื่องโปรแกรมหรือใช้ดาวน์โหลดเคเบิลก็ได้

1.6 การติดตั้งซอฟต์แวร์ทูล

1.6.1 ขั้นตอนการขอและดาวน์โหลดซอฟต์แวร์

สำหรับคนที่มีหนังสือเล่มนี้ในท้ายเล่มจะมีแผ่น CD ให้มาด้วยจำนวน 3 แผ่น เนื่องจากซอฟต์แวร์ทูล ISE WebPACK 8.1i เป็นไฟล์ขนาดใหญ่จึงไม่สามารถเก็บลงใน CD แผ่นเดียวได้ โดยที่ CD แผ่นที่ 1 จะมีซอฟต์แวร์ทูล ISE WebPACK 8.1i ส่วนที่ 1 (ประกอบด้วยไฟล์ชื่อ idata, platform และ setup.exe) ส่วน CD แผ่นที่ 2 จะมีไฟล์ซอฟต์แวร์ทูล ISE WebPACK 8.1i ส่วนที่ 2 (ประกอบด้วยไฟล์ชื่อ idata และ Service Pack 2 ชื่อ S_1_02i_win.exe) สำหรับ CD แผ่นที่ 3 ประกอบด้วยตัวอย่างไฟล์การทดลองต่างๆ ไฟล์หนังสือเล่มนี้ (เป็นภาพสี) ข้อมูลเกี่ยวกับบอร์ดทดลองรุ่นต่างๆและข้อมูลของไอซีเบอร์ที่เกี่ยวข้อง

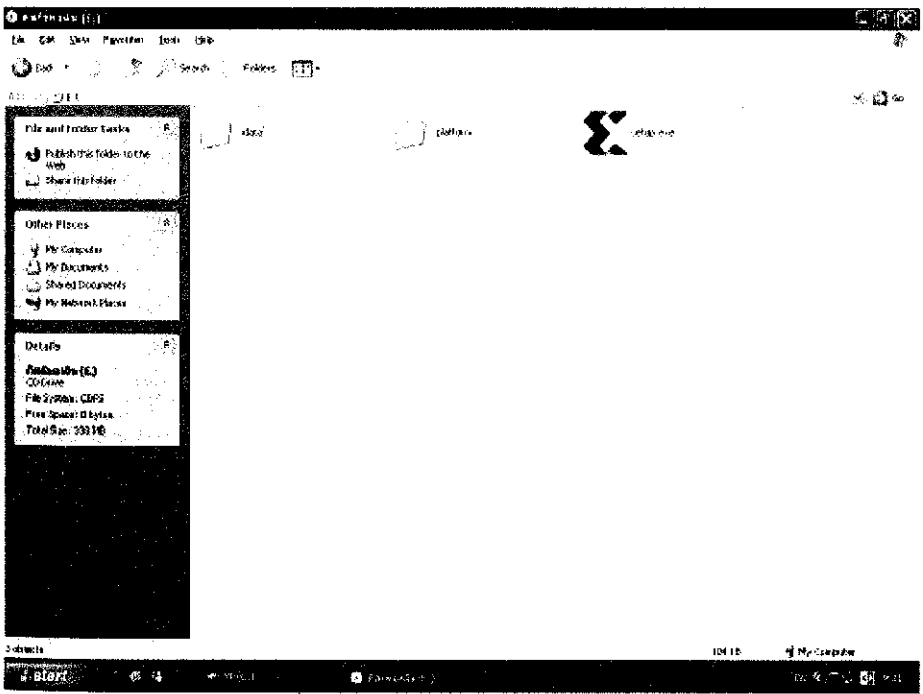
สำหรับคนที่ยังไม่มีซอฟต์แวร์ทูลสามารถดาวน์โหลดได้ฟรีที่ http://www.xilinx.com โดยจะต้องลงทะเบียนกับทาง Xilinx ก่อน จากนั้นทาง Xilinx จะยืนยัน User name และ Pass word โดยส่งมาทางอีเมล์แล้วจึงใช้ User name และ Pass word นี้ในการ Log in เพื่อเข้าไปดาวน์โหลด ISE WebPACK 8.1i โดยเลือกเวอร์ชั่นที่ RUN บน Windows 2000 / Windows XP (ไฟล์ชื่อ WebPACK_81i_SFD.exe)

1.6.2 วิธีการติดตั้ง ISE WebPACK 8.1i

1) ให้ทำการ Uninstall ซอฟต์แวร์เวอร์ชั่นก่อน (ถ้าเคยติดตั้งมาแล้ว) ทิ้ง จากนั้นสร้าง Folder ขึ้นใหม่ในไดรฟ์ C โดยใช้ชื่อ C:\ISE_WebPACK_81i (หรืออาจใช้ชื่ออื่นก็ได้ แต่ต้องไม่ซ้ำกับไฟล์ของ Xilinx)

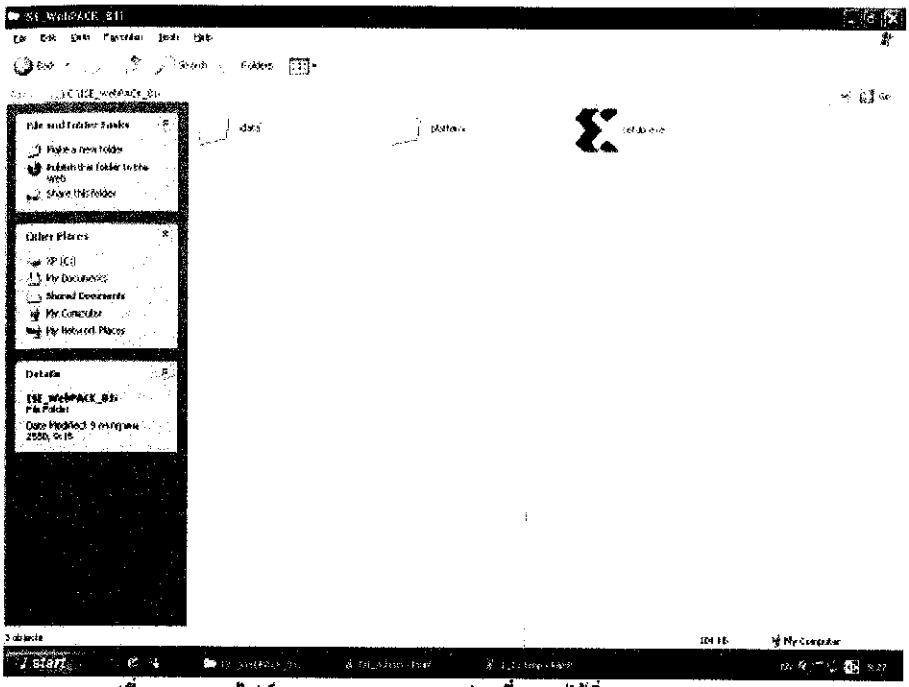
<u>หมายเหตุ</u> สำหรับคนที่ดาวน์โหลดซอฟต์แวร์จาก Xilinx โดยตรงให้ดับเบิลคลิ๊กไฟล์ชื่อ WebPACK_81i_SFD.exe แล้วข้ามไปทำที่ข้อ 6)

2) ให้เปิดไฟล์จาก CD แผ่นที่ 1 ก็จะปรากฏไฟล์ 3 ไฟล์ดังรูปที่ 1.13



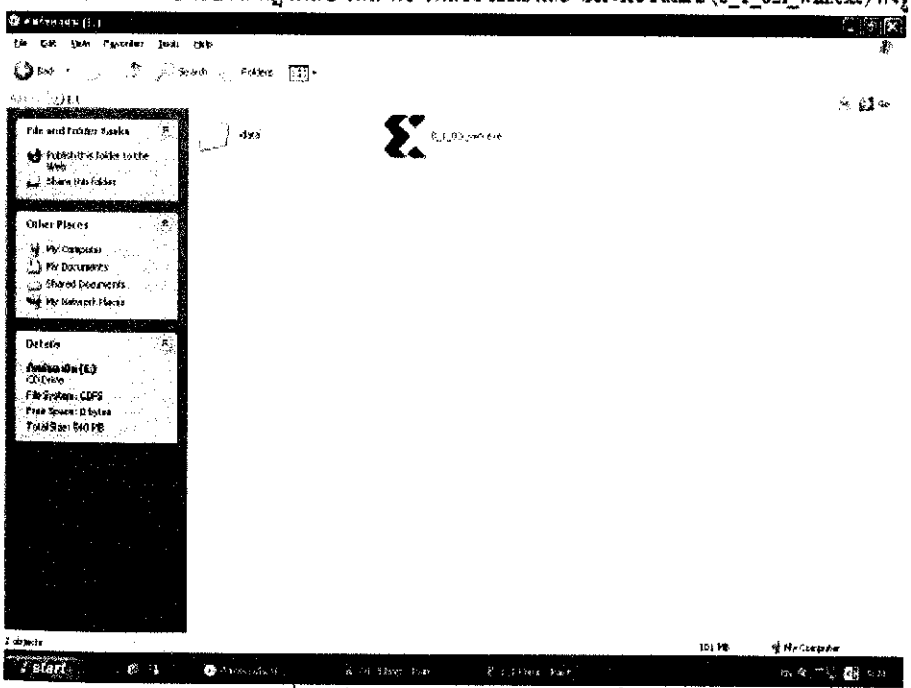
รูปที่ 1.13 ไฟล์ ISE WebPACK 8.1i (ส่วนที่ 1)

3) Copy ไฟล์จาก CD แผ่นที่ 1 มาไว้ที่ C:\ISE_WebPACK_81i จะได้ดังรูปที่ 1.14



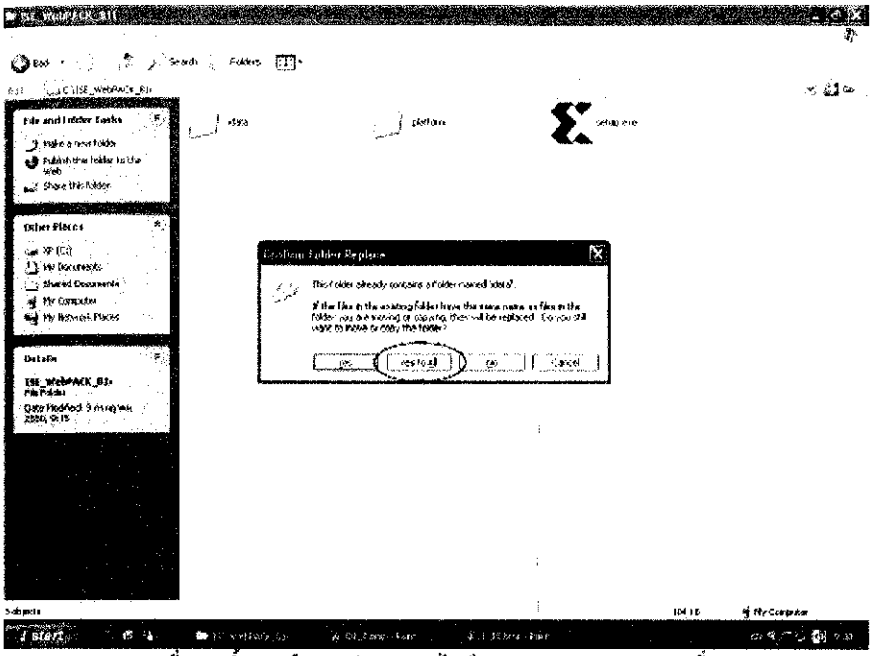
รูปที่ 1.14 Copy ไฟล์ ISE WebPACK 8.1i (ส่วนที่ 1) มาไว้ที่ C:\ISE_WebPACK_81i

4) ให้เปิดไฟล์จาก CD แผ่นที่ 2 จะปรากฏไฟล์ 2 ไฟล์ คือ ไฟล์ชื่อ idata และ Service Pack 2 (8_1_02i_win.exe) ดังรูปที่ 1.15



รูปที่ 1.15 ไฟล์ ISE WebPACK 8.1i (ส่วนที่ 2)

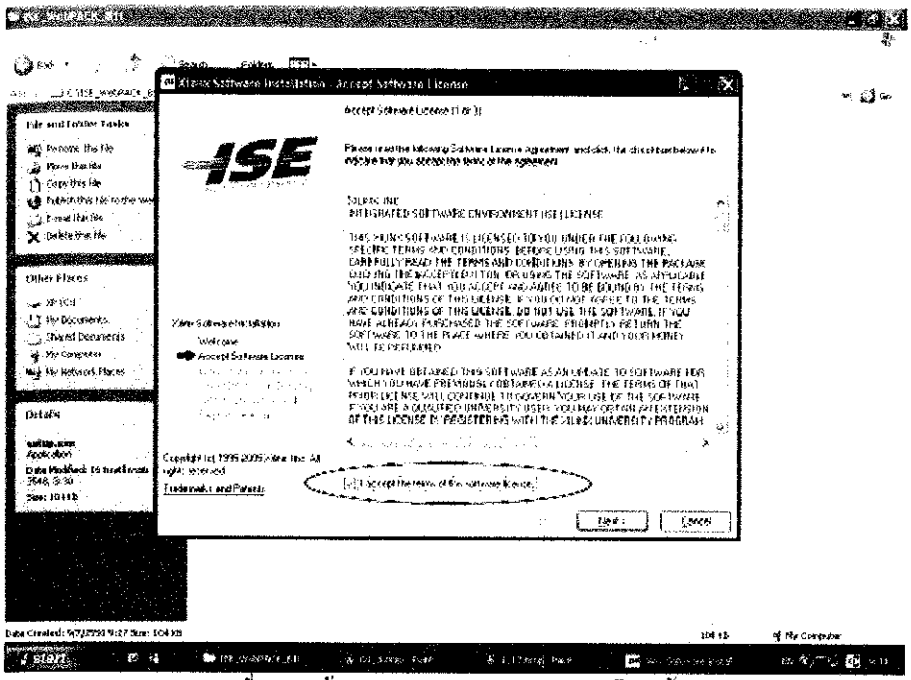
5) Copy ไฟล์จาก CD แผ่นที่ 2 มาไว้ที่ C:\ISE_WebPACK_81i จะได้ดังรูปที่ 1.16 คลิ๊ก Yes to All แล้วจะได้ดังรูปที่ 1.17



รูปที่ 1.16 ขั้นตอนในระหว่าง Copy ไฟล์ ISE WebPACK 8.1i (ส่วนที่ 2)

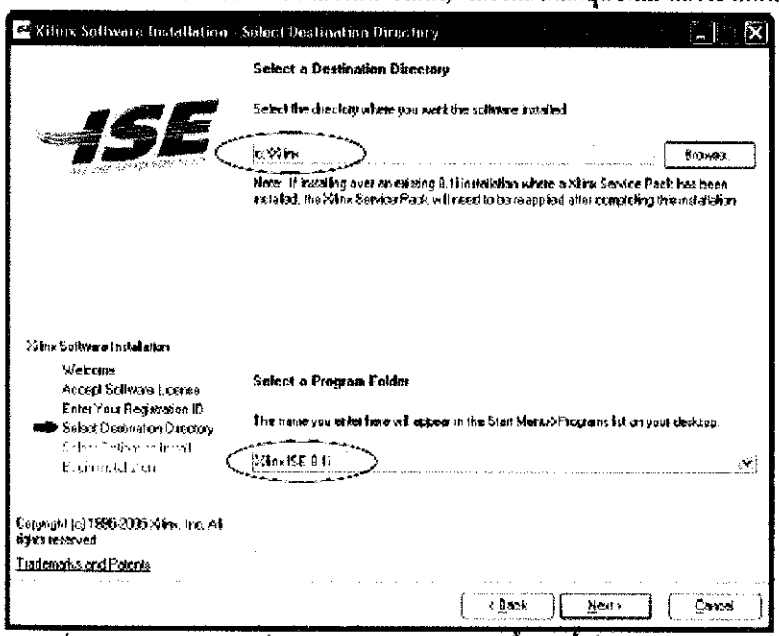
รูปที่ 1.17 แสดงรูปเมื่อรวมไฟล์ ISE WebPACK 8.1i (ส่วนที่ 1 และส่วนที่ 2)

6) ดับเบิลคลิ๊กที่ไฟล์ setup.exe จากนั้นจะได้หน้าต่าง Accept Software License ให้คลิ๊ก "√" ที่หน้า "I accept the term of this software license" ดังรูปที่ 1.18 แล้วคลิ๊กปุ่ม Next แล้วจะได้หน้าต่างถัดไป (ทำซ้ำจนครบ 3 ครั้ง)



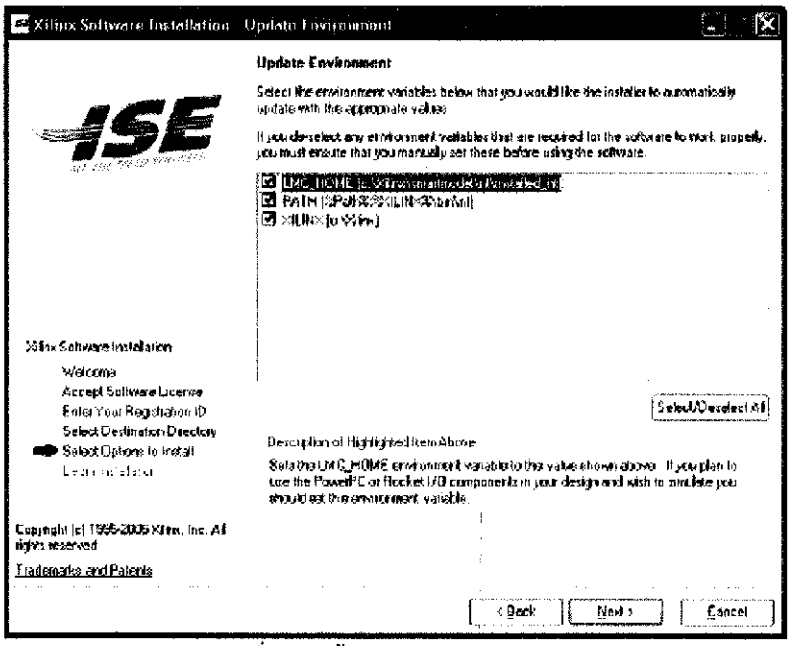
รูปที่ 1.18 หน้าต่าง Accept Software License (มี 3 หน้า)

7) ทำการเลือก Directory ที่จะทำการติดตั้งไฟล์ต่างๆจะได้ดังรูปที่ 1.19 (ในกรณีนี้ติดตั้งที่ C:\Xilinx และชื่อใน Program Folder ใช้ชื่อ Xilinx ISE 8.1i จะเป็นชื่อที่ซอฟต์แวร์ได้กำหนดล่วงหน้าไว้แล้ว) เสร็จแล้วคลิ๊กปุ่ม Next แล้วจะได้หน้าต่างถัดไป



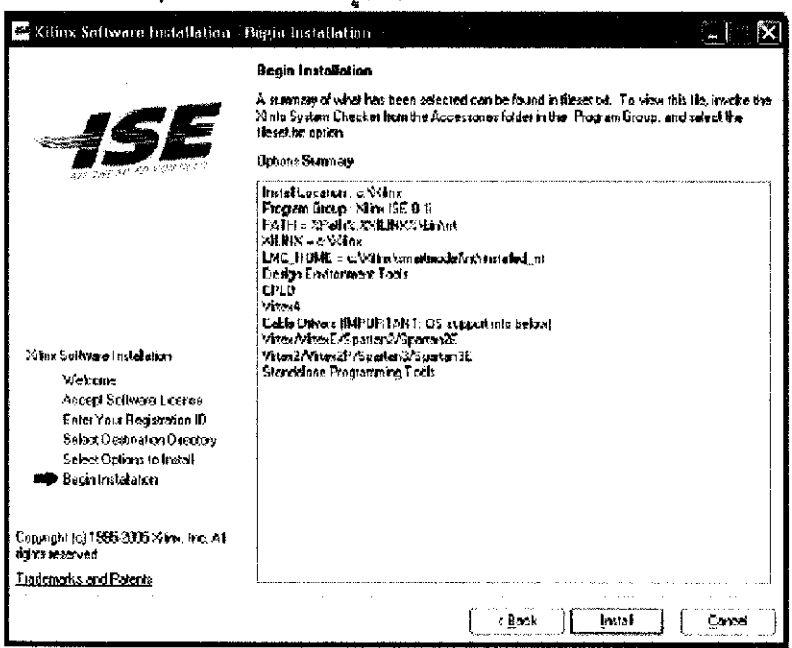
รูปที่ 1.19 เลือก Directory ที่จะเอาซอฟต์แวร์ต่างๆไปติดตั้งและตั้งชื่อ Program Folder

8) โปรแกรมจะ Set และ Update Xilinx variable และ PATH variable ดังรูปที่ 1.20 แล้วคลิ๊กปุ่ม Next

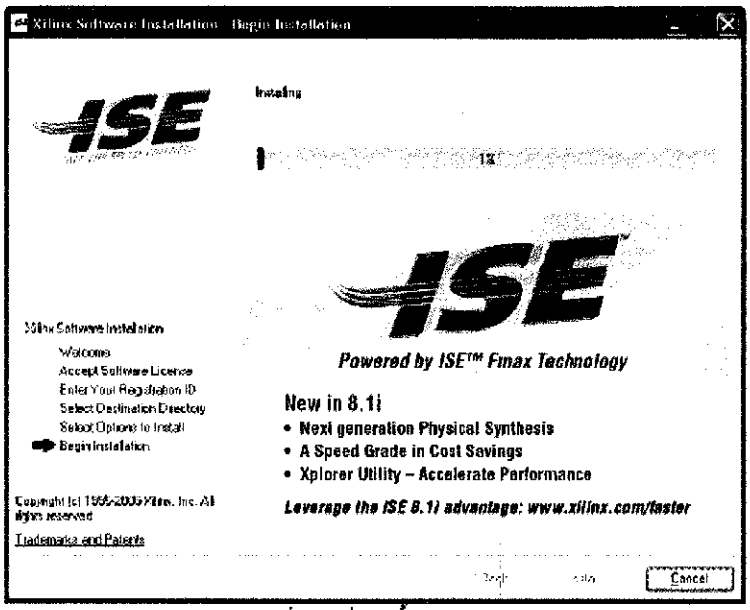


รูปที่ 1.20 หน้าต่าง Update Environment

9)โปรแกรมจะทำการทบทวนค่า Parameter ต่างๆ ที่ใช้ก่อนจะทำการติดตั้งดังรูปที่ 1.21 จากนั้นให้คลิ๊กปุ่ม Install แล้วโปรแกรมจะเริ่มติดตั้งไฟล์ต่างๆลงในคอมพิวเตอร์ ดังรูปที่ 1.22

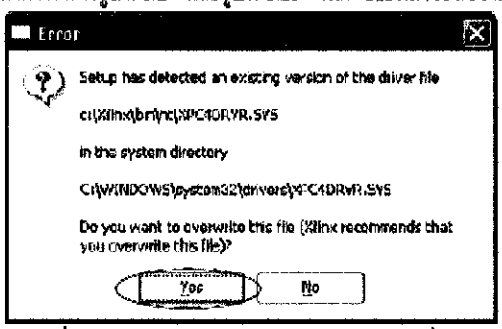


รูปที่ 1.21 หน้าต่าง Begin Installation

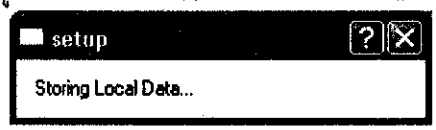


รูปที่ 1.22 เริ่มติดตั้งซอฟต์แวร์

10) ในกรณีที่เครื่องคอมพิวเตอร์เคยติดตั้งซอฟต์แวร์เวอร์ชั่นก่อนหน้านั้นในระหว่างติดตั้งอาจมีหน้าต่าง Error ปรากฏขึ้นดังรูปที่ 1.23 ให้คลิ๊ก Yes แล้วจะได้หน้าต่างติดตั้งรูปที่ 1.24 และรูปที่ 1.25 คลิ๊ก OK แล้วจะถือว่าติดตั้งซอฟต์แวร์ เรียบร้อยแล้ว



รูปที่ 1.23 ยกเลิกไฟล์ Driver ของซอฟต์แวร์เวอร์ชั่นเก่า

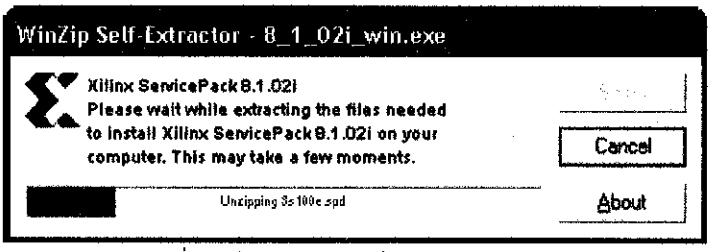


รูปที่ 1.24 หน้าต่าง Setup

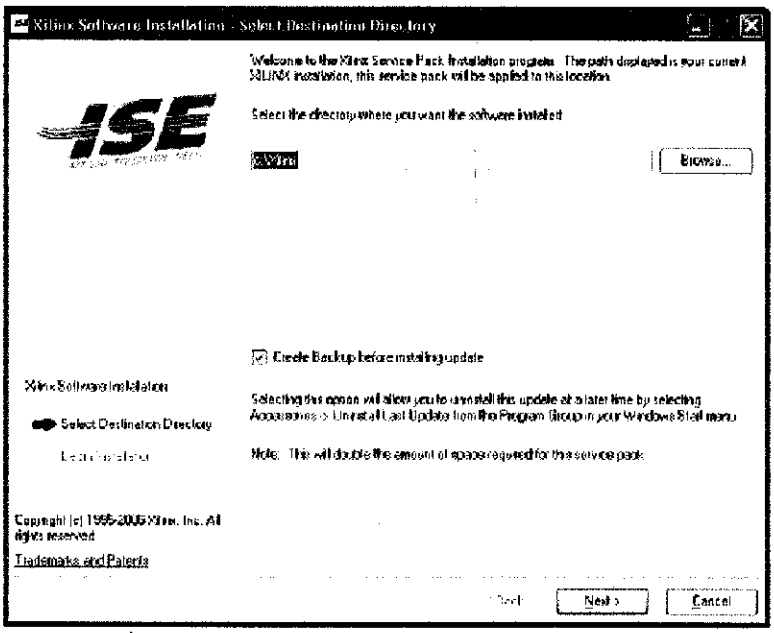
รูปที่ 1.25 หน้าต่าง Xilinx 8.1i Setup

1.6.2 วิธีการติดตั้ง Service Pack 2

1) เข้าไปที่ Folder ที่เราเก็บซอฟต์แวร์ไว้ในรูปที่ 1.17 จากนั้นดับเบิลคลิ๊กที่ไฟล์ 8_1_02i_win.exe จะได้ดังรูปที่ 1.26 แล้วโปรแกรมจะทำการแตกไฟล์ .exe ออกอย่างอัตโนมัติ เมื่อแตกไฟล์เสร็จแล้วจะได้ดังรูปที่ 1.27

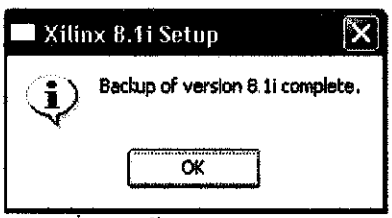


รูปที่ 1.26 โปรแกรมจะแตกไฟล์ 6_3_02I_PC.exe

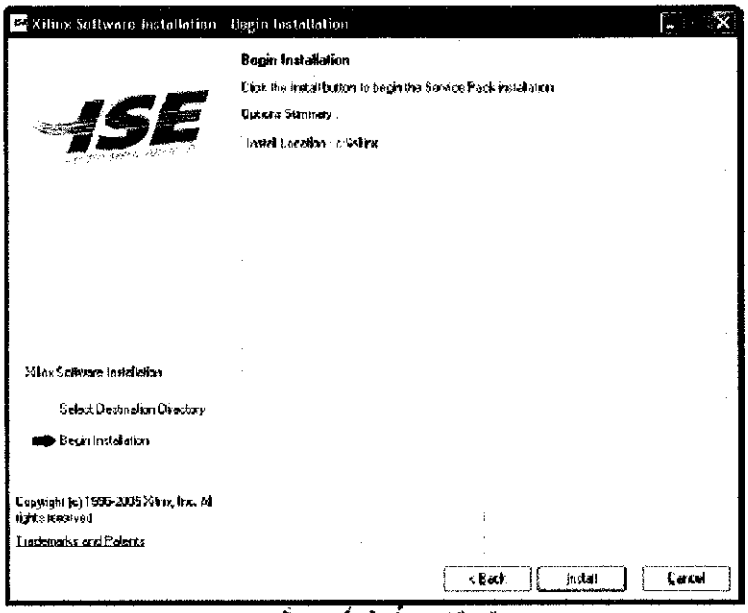


รูปที่ 1.27 เลือก Directory และ Create Backup before installing update

2) คลิ๊ก Next เพื่อเริ่มแบ็คอัพไฟล์ก่อนทำการติดตั้ง เมื่อแล้วเสร็จจะได้หน้าต่างดังรูปที่ 1.28 คลิ๊ก OK จากนั้นจะได้หน้าต่างดังรูปที่ 1.29

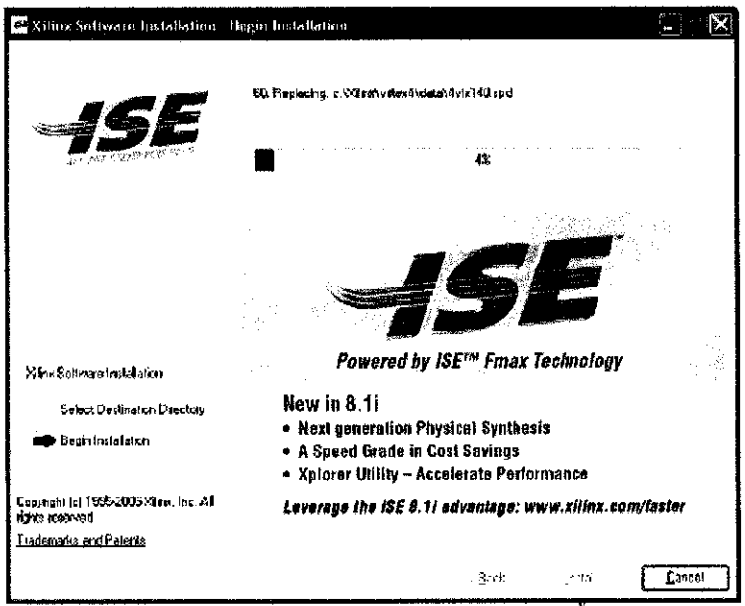


รูปที่ 1.28 หน้าต่าง Xilinx 8.1i Setup

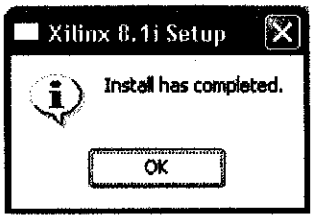


รูปที่ 1.29 เริ่มติดตั้งซอฟต์แวร์

3) คลิ๊ก Install เพื่อเริ่มติดตั้งซอฟต์แวร์ดังรูปที่ 1.30 เมื่อแล้วเสร็จจะได้ดังรูปที่ 1.31 คลิ๊ก OK แล้วถือว่าการติดตั้งซอฟต์แวร์เรียบร้อยแล้ว (ควร Reboot คอมพิวเตอร์ก่อนจะใช้งาน)



รูปที่ 1.30 หน้าต่าง Begin Installation ขณะกำลังติดตั้ง



รูปที่ 1.31 หน้าต่าง Xilinx 8.1i Setup

ไดรเวอร์

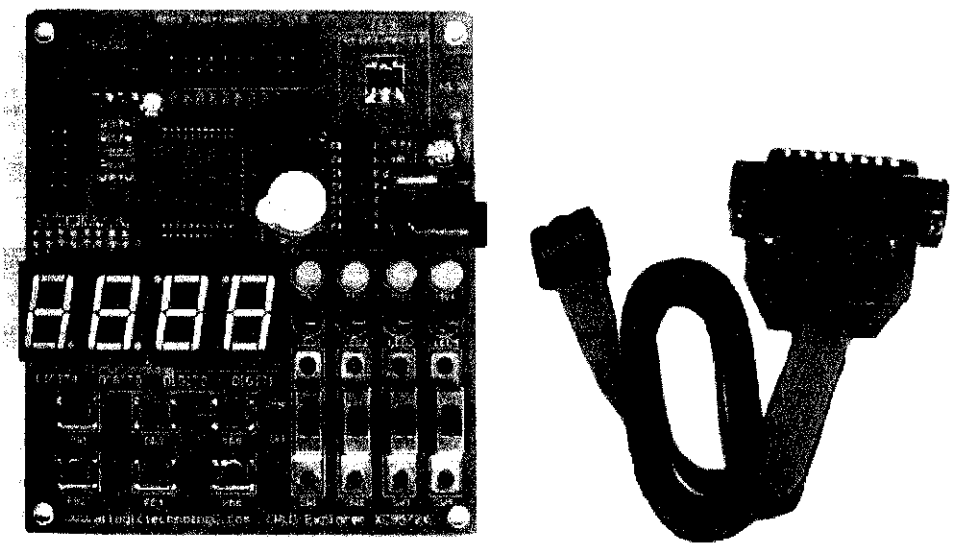
การใช้ซอฟต์แวร์ ISE WebPACK 8.1i-9.1i นั้นมีความจำเป็นต้องติดตั้งซอฟต์แวร์ชื่อ Adobe Reader 7 ขึ้นไปควบคู่กันด้วยเพื่อใช้สำหรับอ่านไฟล์ .pdf ดังนั้นหากยังไม่ได้ติดตั้งก็ให้ผู้ใช้จะติดตั้งซอฟต์แวร์ชื่อ Adobe Reader นี้ด้วย

บอร์ดทดลองที่ใช้ในการทดลองโดยตรง

บอร์ดทดลองที่ใช้ในการทดลองหลักๆ มี 3 บอร์ด คือ บอร์ด CPLD Explorer XC9572XL, FPGA Discovery-III XC3S200F และ FPGA Discovery-III XC3S200F4 ผู้ใช้ควรศึกษารายละเอียดบอร์ดทดลองรุ่นที่จะใช้งาน โดยที่รายละเอียดบอร์ดรุ่นต่างๆสามารถอ่านเพิ่มเติมได้จากคู่มือบอร์ดรุ่นนั้นๆ

1.1 CPLD Explorer XC9572XL

บอร์ดทดลองเอนกประสงค์รุ่น CPLD Explorer XC9572XL เป็นบอร์ดทดลองที่เหมาะสำหรับผู้เริ่มต้นศึกษาการออกแบบวงจรดิจิตอลด้วย FPGA และ CPLD มีรายละเอียดแสดงดังรูปที่ 1.32 คุณสมบัติทั่วๆไปของบอร์ดทดลองเป็นดังนี้



รูปที่ 1.32 CPLD Explorer XC9572XL พร้อมสายดาวน์โหลด

คุณสมบัติทั่วไป

- CPLD เบอร์ XC9572XL (1,600 เกต) แบบ PLCC 44 ขา (PC44) Speed Grade –10
- เซเวนเซกเมนต์ จำนวน 4 หลัก
- LED แสดงผล 2 สถานะ จำนวน 4 ดวง
- Logic monitor ที่เป็น LED แสดงผล 3 สถานะ จำนวน 4 ดวง (ใช้ร่วมกับ K1)
- ออด (Buzzer) จำนวน 1 ตัว
- Slide switch 4 ตัว (ใช้ร่วมกับ Push button switch)

ภาคผนวก ข

ภาษา VHDL

บทนี้จะอธิบายการออกแบบวงจรดิจิตอลด้วยภาษา VHDL ซึ่งเป็นภาษาระดับสูงโดยใช้ซอฟต์แวร์บูล ISE WebPACK 8.1i ของบริษัท Xilinx ที่สามารถดาวน์โหลดได้ฟรีทางอินเตอร์เนต การออกแบบวงจรดิจิตอลด้วยภาษาระดับสูงเหมาะสำหรับการออกแบบวงจรขนาดใหญ่เหมาะเป็นการใช้ความสามารถของซอฟต์แวร์บูลช่วยในการออกแบบ

VHDL ถูกพัฒนาเริ่มต้นโดยกระทรวงกลาโหมสหรัฐอเมริกา ต่อมา IEEE ได้รับภาษานี้เข้ามาปรับปรุงเป็นมาตรฐาน IEEE Std 1076-1987 (VHDL87) และได้ปรับปรุงเป็น IEEE Std 1076-1993 (VHDL93) IEEE Std 1076-2000 และ IEEE Std 1076-2002 ตามลำดับ แต่อย่างไรก็ตามชนิดข้อมูล (Data type) ที่นิยามไว้ใน IEEE Std 1076 นั้นไม่ครอบคลุมลอจิกบางสถานะ เช่น High impedance หรือสถานะอื่นๆที่ใช้จำลองการทำงาน ดังนั้น IEEE จึงได้นิยามชนิดข้อมูล std_logic (และ std_ulogic) ไว้ใน IEEE Std 1164-1993 เพื่อแก้ไขปัญหาดังกล่าว แต่ภายที่ IEEE ไม่ได้นิยามตัวดำเนินการ (Operator) ที่ใช้สำหรับเขียนโค้ด VHDL เพื่อสังเคราะห์วงจรไว้อย่างเพียงพอ ทำให้ผู้ใช้งานและผู้พัฒนาซอฟต์แวร์บูลสังเคราะห์วงจรแต่ละรายต้องสร้าง Package ต่างๆ ขึ้นมาใช้เองซึ่งไม่เป็น Package มาตรฐาน ดังนั้น IEEE จึงได้นิยาม นิยามตัวดำเนินการและชนิดข้อมูลแบบใหม่ไว้ในVHDL Synthesis packages ไว้ใน IEEE Std 1076.3-1997 เพื่อใช้แทน Package ที่ไม่เป็นมาตรฐานเหล่านั้นเพื่อช่วยให้โค้ด VHDL ที่เขียนขึ้นนั้นสามารถเข้ากันได้กับซอฟต์แวร์บูลต่างๆ

2.1 VHDL พื้นฐาน

องค์ประกอบภาษา VHDL (Structural elements) ตาม IEEE 1076 มีหน่วยออกแบบ (Design unit) ได้แก่

- Entity declaration คือส่วน Interface ที่จะบอกว่าอินพุตเอาต์พุตพอร์ต (I/O Port)ของวงจรมีอะไรบ้าง
- Architecture คือส่วนที่เป็น Body ที่บรรยายพฤติกรรมการทำงานของระบบดิจิตอลที่ออกแบบ
- Package เป็นที่เก็บ เช่น ค่าคงที่ โปรแกรมย่อย และ นิยามชนิดข้อมูลต่างๆ เพื่อความสะดวกเขียนโค้ด VHDL
- Configuration ใช้สำหรับเลือก Architecture (ซึ่งอาจมีหลาย Architecture) มาจับคู่กับ Entity

โดยทั่วไปวงจรที่ออกแบบด้วยภาษา VHDL จะประกอบด้วยหน่วยออกแบบอย่างน้อย 2 หน่วย คือ ประกาศใช้เอนติตี้ (Entity declaration) หรือ เอนติตี้ และ อาชิเทคเชอร์บอดี้ (Architecture body) หรือ อาชิเทคเชอร์

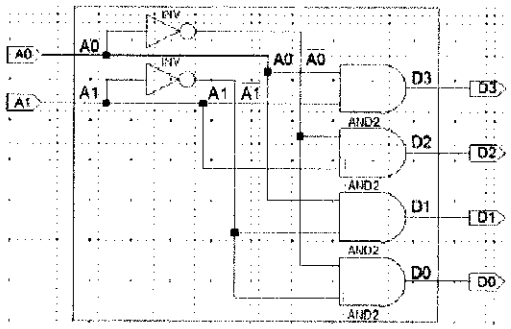
ตัวอย่างที่ 2.1 วงจรถอดรหัสเข้า 2 ออก 4 (2to 4 Decoder) ที่ตารางความจริงดังรูปที่ 2.1a) ซึ่งเราสามารถเขียนสมการบูลีนในรูปฟอร์มสมการ SOP (Sum of product) ได้ดังนี้

$$D0 = \overline{A1} \cdot \overline{A0}$$
$$D1 = \overline{A1} \cdot A0$$
$$D2 = A1 \cdot \overline{A0}$$
$$D3 = A1 \cdot A0$$

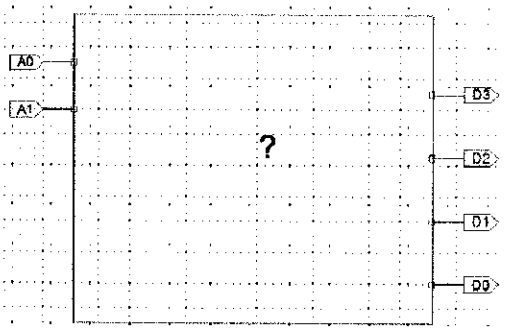
จากสมการบูลีนเราสามารถสร้างวงจรได้ดังรูปที่ 2.1b) ซึ่งเราจะใช้ข้อมูลในรูปที่ 2.1c) และ รูปที่ 2.1d) ไปเขียนโค้ด (Code) VHDL ของวงจร 2 to 4 Decoder ได้ดังรูปที่ 2.2a) และ รูปที่ 2.2b) โดยตัวอย่างนี้เราเขียนคำสงวน (Reserve words) ด้วยตัวพิมพ์เล็ก ซึ่งซอฟต์แวร์บูล ISE WebPACK จะแสดงคำสงวนด้วยตัวอักษรที่เป็นสีต่างๆอย่างชัดเจน

Input		Output			
A1	A0	D3	D2	D1	D0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

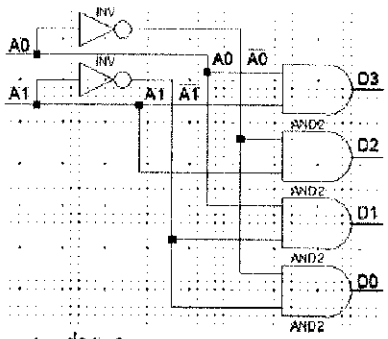
2.1a) ตารางความจริงของวงจร 2to4 Decoder



2.1b) วงจร 2 to4 Decoder



2.1c) ส่วนที่ใช้อธิบาย Entity declaration



2.1d) ส่วนที่ใช้อธิบาย Architecture body

รูปที่ 2.1 รายละเอียดต่างของวงจร 2 to4 Decoder ที่จะนำไปเขียนด้วยภาษา VHDL

ชื่อเอนติตี้ Mode ชนิดข้อมูล bit

```
2  entity DECODER2TO4 is
3      port ( A0 : in  bit;
4             A1 : in  bit;
5             D0 : out  bit;
6             D1 : out  bit;
7             D2 : out  bit;
8             D3 : out  bit);
9  end DECODER2TO4;
10
11 architecture BEHAVIORAL of DECODER2TO4 is
12    begin
13       D0 <= (not A1)and(not A0);
14       D1 <= (not A1)and A0;
15       D2 <= A1 and(not A0);
16       D3 <= A1 and A0;
17 end BEHAVIORAL;
```

จบด้วย ";"

บรรทัดที่ 2-9 เป็นการประกาศใช้เอนติตี้ ซึ่งสามารถจะบอกว่าวงจรมีอินพุตและเอาต์พุตอะไรบ้าง

ชื่อสถาปัตยกรรม ชื่อเอนติตี้

บรรทัดที่ 11-17 เป็นสถาปัตยกรรมบอดี้ ซึ่งสามารถจะบอกว่าวงจรทำงานอย่างไร

Operator

2.2a) รูปแบบ VHDL87

```
2   entity DECODER2TO4 is
3       port ( A0 : in  bit;
4              A1 : in  bit;
5              D0 : out  bit;
6              D1 : out  bit;
7              D2 : out  bit;
8              D3 : out  bit);
9   end entity DECODER2TO4;
10
11  architecture BEHAVIORAL of DECODER2TO4 is
12     begin
13        D0 <= (not A1)and(not A0);
14        D1 <= (not A1)and A0;
15        D2 <= A1 and(not A0);
16        D3 <= A1 and A0;
17  end architecture BEHAVIORAL;
```

บรรทัดที่ 9 อาจจะใช้ end entity แทนคำว่า end

บรรทัดที่ 17 อาจจะใช้ end architecture แทนคำว่า end

2.2b) รูปแบบเพิ่มเติมใน VHDL93 (อาจจะเขียนเพิ่มเติมหรือไม่เขียนเพิ่มเติมก็ได้)

รูปที่ 2.2 โค้ด VHDL ของวงจร 2 to 4 Decoder) เลขบรรทัดมีไว้เพื่อความสะดวกซึ่งไม่ใช่ส่วนของโค้ด)

จากรูปที่ 2.2 จะเห็นว่าโค้ด VHDL นั้นสามารถอ่านเข้าใจง่ายแม้บางคนจะไม่เคยรู้เกี่ยวกับภาษา VHDL มาก่อน วงจร 2 to 4 Decoder นี้มี Entity ชื่อ DECODER2TO4 และ Architecture ชื่อ BEHAVIORAL มี A0 และ A1 เป็นอินพุต (A0 และ A1 มี Mode เป็น Input หรือ in) มี D0-D3 เป็นเอาต์พุต (D0-D3 มี Mode เป็น Output หรือ out)

โดยที่	D0 <= (not A1) and (not A0)	มีความหมายว่า	$D0 = \overline{A1} \cdot \overline{A0}$
	D1 <= (not A1) and A0	มีความหมายว่า	$D1 = \overline{A1} \cdot A0$
	D2 <= A1 and (not A0)	มีความหมายว่า	$D2 = A1 \cdot \overline{A0}$
	D3 <= A1 and A2	มีความหมายว่า	$D3 = A1 \cdot A0$

เราเรียก not และ and ว่าเป็น "ตัวดำเนินการ" หรือ Operator ซึ่งจะอธิบายรายละเอียดในภายหลัง และจากรูปที่ 2.2a) และรูปที่ 2.2b) นั้น A0, A1และ D0-D3 จะมีสถานะลอจิกเป็นได้เฉพาะ '1' หรือ '0' เราจึงใช้ชนิดข้อมูล (Data type หรือ Type) ของอินพุตและเอาต์พุตเป็นชนิดข้อมูล bit

คำสั่ง (Statement) ที่อยู่ใน Architecture คือ ในบรรทัดที่ 13-16 จะทำงานพร้อมกัน (Signal concurrency) ทั้ง 4 คำสั่ง โดยไม่สนใจลำดับก่อนและหลัง เราเรียกว่าคำสั่งคอนเคอร์เรนต์ (Concurrent statement) การเขียนคำสั่งสลับที่กันดังรูปที่ 2.3 จะยังคงให้ผลลัพธ์เหมือนเดิม

```
11  architecture BEHAVIORAL of DECODER2TO4 is
12     begin
13        D3 <= A1 and A0;
14        D2 <= A1 and(not A0);
15        D1 <= (not A1)and A0;
16        D0 <= (not A1)and(not A0);
17  end architecture BEHAVIORAL;
```

รูปที่ 2.3 ตัวอย่าง Statement ในVHDL ที่วางสลับบรรทัดกัน

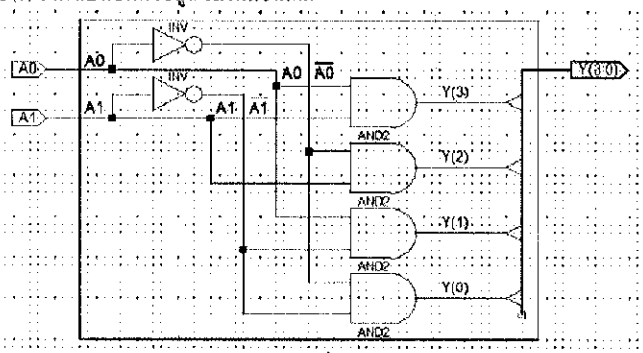
จากวงจร 2 to 4 Decoder ในตัวอย่างที่ 2.1 เราสามารถเขียนประกาศใช้เอนติตี้ให้กระชับขึ้นโดยใช้เครื่องหมาย "." คั่นระหว่างอินพุตหรือเอาต์พุตต่างๆได้ดังรูปที่ 2.4

```
2   entity DECODER2TO4 is
3       port ( A0,A1 : in  bit;
4              D0,D1,D2,D3 : out  bit);
5   end DECODER2TO4;
6
7   architecture BEHAVIORAL of DECODER2TO4 is
8      begin
9         D0 <= (not A1)and(not A0);
10        D1 <= (not A1)and A0;
11        D2 <= A1 and(not A0);
12        D3 <= A1 and A0;
13  end BEHAVIORAL;
```

การเขียนเอ้นพุตหรือเอาต์พุตหลายตัวจะต้องมี "," คั่นระหว่างอินพุตหรือเอาต์พุตทุกตัว

รูปที่ 2.4 โค้ด VHDL ของวงจร 2 to4 Decoder ที่เขียนให้กระชับขึ้น

การเขียนโค้ด VHDL ของวงจร 2 to 4 Decoder โดยให้เอาต์พุตอยู่ในรูปของบัส (Bus) แสดงดังรูปที่ 2.5a) และ รูปที่ 2.5b) โดยรูปที่ 2.5a) เอาต์พุตจะเป็นบัส Y ขนาด 4 บิต โดยที่สมาชิกแต่ละบิตประกอบด้วย Y(3), Y(2), Y(1) และ Y(0) (เป็นคนละตัวกับ Y3, Y2, Y1 และ Y0) ดังนั้นโค้ด VHDL ในรูปที่ 2.5b) Y จะเป็นชนิดข้อมูล bit_vector ซึ่งเป็นชนิดข้อมูลแบบอเรย์ (Array) ของ bit โดยเขียน bit_vector(3 downto 0) แทนความหมายของบัสขนาด 4 บิต แต่สมาชิกของ Y ทุกตัวคือ Y(3), Y(2), Y(1) และ Y(0) ยังคงเป็นชนิดข้อมูล bit เหมือนเดิม



2.5a) ถังวงจร 2 to 4 Decoder ที่มีเอาต์พุตเป็นบัสขนาด 4 บิต

```
2   entity DECODER2TO4 is
3       port ( A0,A1 : in  bit;
4              Y : out  bit_vector(3 downto 0));
5   end DECODER2TO4;
6
7   architecture BEHAVIORAL of DECODER2TO4 is
8      begin
9         Y(0) <= (not A1)and(not A0);
10        Y(1) <= (not A1)and A0;
11        Y(2) <= A1 and(not A0);
12        Y(3) <= A1 and A0;
13  end BEHAVIORAL;
```

2.5b) โค้ด VHDL ของวงจร 2 to 4 Decoder ที่มีเอาต์พุตเป็นบัสขนาด 4 บิต

รูปที่ 2.5 ถังวงจรและ โค้ด VHDL ของวงจร 2 to 4 Decoder ที่มีเอาต์พุตเป็นบัสขนาด 4 บิต

2.2 กฎเกณฑ์ทั่วไป

โค้ด VHDL นั้นสามารถเขียนได้ทั้ง ตัวพิมพ์ใหญ่ ตัวพิมพ์เล็ก หรือเขียนปนกันก็จะมีความหมายเดียวกัน (Case insentitive) การเขียนคำสั่งจะจบด้วย ";" (Semicolon) การกำหนดค่าจะใช้ "<=" (Signal assignment) การเขียนคำอธิบาย (Comment) จะใช้ "--" 2 ตัวติดกันคือ "---" แล้วเขียนตามด้วยคำอธิบาย

2.3 การตั้งชื่อ

การตั้งชื่อ (Identifiers) ตาม VHDL87 อาจประกอบด้วย อักขระ ตัวเลข และ เครื่องหมายขีดเส้นใต้ โดยที่

- ต้องขึ้นต้นด้วยอักขระ (Letter) และ จะต้องเขียนติดกัน โดยไม่เว้นช่องว่าง (White space)
- ห้ามใช้เครื่องหมายขีดเส้นใต้ (Underscore) ติดกันและ ห้ามลงท้ายชื่อด้วยเครื่องหมายขีดเส้นใต้
- ห้ามใช้คำสงวน (Reserved words) ดังตารางในรูปที่ 2.6

abs	but	function	literal	others	return	transport
access	case	generate	loop	out	rol	type
after	component	generic	map	package	ror	unaffected
alias	configuration	group	mod	port	select	units
all	constant	guarded	nand	postponed	severity	until
and	disconnect	if	new	procedure	signal	use
architecture	downto	impure	next	process	shared	variable
array	else	in	nor	pure	sla	wait
assert	elsif	inertial	not	range	sll	when
attribute	end	inout	null	record	sra	while
begin	entity	is	of	register	srl	with
block	exit	label	on	reject	subtype	xnor
body	file	library	open	rem	then	xor
buffer	for	linkage	or	report	to	

รูปที่ 2.6 ตารางแสดงรายการคำสงวนในภาษา VHDL93 (VHDL Reserved words)

ตัวอย่างการตั้งชื่อ ARCHI, aRChi, archi ถูก เป็นชื่อเดียวกัน

HALF_ADDER	ถูก
HALF__ADDER	ผิด เพราะ Underscore ติดกัน 2 ตัว
HALF_ADDER_	ผิด เพราะเว้นช่องว่างและ จบชื่อด้วยเครื่องหมายขีดเส้นใต้
74LS00	ผิด เพราะขึ้นต้นด้วยตัวเลข
open	ผิด เพราะใช้คำสงวน (ดูVHDL Reserved words:(

2.4 ประเภทของพอร์ต

ประเภทของพอร์ต (Port mode หรือ Mode) ที่ใช้ใน Entity declaration ได้แก่

- in คือ อินพุต (Input) เป็นพอร์ตรับสัญญาณจากวงจรภายนอกเข้ามาในวงจรที่ออกแบบ
- out คือ เอาต์พุต (Output) เป็นพอร์ตส่งสัญญาณจากวงจรที่ออกแบบออกไปยังวงจรภายนอก
- inout คือ อินพุต เอาต์พุต (I/O) เป็นพอร์ตรับ-ส่งสัญญาณเข้าหรือออกที่ใช้ติดต่อกับวงจรภายนอก
- buffer คือ เป็นเอาต์พุตพอร์ตที่สามารถอ่านค่ากลับเข้ามาภายในวงจรที่ออกแบบได้

2.5 การเขียนโค้ด VHDL เบื้องต้นโดยใช้ชนิดข้อมูล std_logic และ std_logic_vector

โค้ด VHDL ที่กล่าวมาแล้วจะใช้ชนิดข้อมูล bit และ bit_vector ซึ่งมีสถานะลอจิกเป็น '0' และ '1' เท่านั้น แต่อย่างไรก็ตามวงจรในทางปฏิบัติอาจมี Tri-state buffer รวมอยู่ด้วยซึ่งมีสถานะลอจิกเป็นได้ทั้ง '0','1' และ High-impedance ดังนั้นชนิดข้อมูล bit จะไม่ครอบคลุมทุกสถานะ เราจึงต้องใช้ชนิดข้อมูล (Type) ใหม่ คือ std_logic แทน bit ซึ่งมีได้ 9 สถานะ ดังนี้

'U' มีสถานะ ลอจิก Uninitialized

'X' มีสถานะ ลอจิก Unknown

'0' มีสถานะ ลอจิก Low (ลอจิก'0')

'1' มีสถานะ ลอจิก High (ลอจิก'1')

'Z' มีสถานะ ลอจิก High impedance

'W' มีสถานะ ลอจิก Weak unknown

'L' มีสถานะ ลอจิก Weak low

'H' มีสถานะ ลอจิก Weak high

'-' มีสถานะ ลอจิก Don't care

เนื่องจาก std_logic และ std_logic_vector ไม่เป็นชนิดข้อมูลที่กำหนดล่วงหน้าไว้แล้ว (Predefinded type)ใน IEEE Std 1076 ดังนั้นก่อนใช้ชนิดข้อมูลนี้จึงต้องเรียกใช้ Package ชื่อ std_logic_1164 ที่ถูกคอมไพล์เก็บไว้ใน Library ชื่อ ieee (IEEE Std 1164) โดยต้องเขียนไว้ที่บรรทัดบนสุดของโค้ด (บรรทัดที่ 2 และ 3) เราสามารถใช้ชนิดข้อมูล std_logic และ std_logic_vector แทนชนิดข้อมูล bit และ bit_vector ในรูปที่ 2.5b) ได้ดังรูปที่ 2.7

```
2   library IEEE;
3   use IEEE.STD_LOGIC_1164.ALL;
4
5   entity DECODER2TO4 is
6       port ( A0,A1 : in  STD_LOGIC;
7              Y : out  STD_LOGIC_VECTOR(3 downto 0));
8   end DECODER2TO4;
9
10  architecture BEHAVIORAL of DECODER2TO4 is
11     begin
12        Y(0) <= (not A1)and(not A0);
13        Y(1) <= (not A1)and A0;
14        Y(2) <= A1 and(not A0);
15        Y(3) <= A1 and A0;
16  end BEHAVIORAL;
```

บรรทัดที่ 2 และ 3 เป็นการเรียกใช้ Package ชื่อ std_logic_1164 ที่อยู่ใน library ieee

ชนิดข้อมูล std_logic และ std_logic_vector

รูปที่ 2.7 โค้ด VHDL ของวงจร 2 to 4 Decoder ที่ใช้ชนิดข้อมูล std_logic และ std_logic_vector

2.6 ออบเจ็กต์

ออบเจ็กต์ (Object) ในภาษา VHDL แบ่งออกเป็น 3 ประเภทด้วยกันคือ ค่าคงที่ (Constant) สัญญาณ (Signal) และตัวแปร (Variable) รูปแบบการประกาศใช้ออบเจ็กต์ (Object declaration) เป็นดังนี้

object OBJECT_NAME : TYPE := VALUE ;

1) Constant ใช้กำหนดค่าคงที่ เช่น

constant WIDTH : integer := 16;

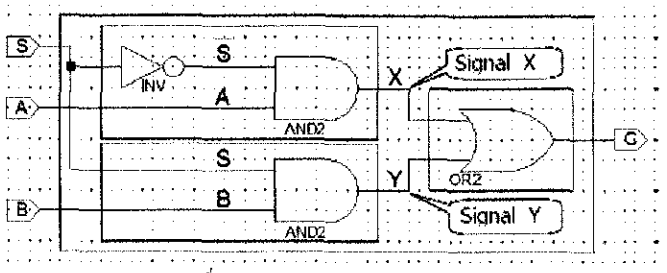
2) Signal เป็นเสมือนสายสัญญาณหรือสายไฟที่ต่ออยู่ภายในวงจร ใช้ส่งผ่านหรือแลกเปลี่ยนข้อมูลระหว่างคำสั่งต่างๆ ในกรณีที่ Signal อยู่ในคำสั่ง Process การอัพเดต (Update) จะทำเมื่อจบ Process ตัวอย่าง Signal เช่น

signal COUNT : std_logic_vector(3 downto 0):= "0000";

3) Variable จะคล้ายกับ Signal แต่ถูกประกาศใช้เฉพาะภายในคำสั่ง Process, Procedure และ Function Variable จะเป็นที่เก็บค่าชั่วคราวและสามารถอัพเดต (Update) ค่าได้ทันที ตัวอย่าง Variable เช่น

variable Y : std_logic;

ตัวอย่างที่ 2.2 วงจรมัลติเพล็กเซอร์ (2 to 1 Multiplexer) มีสัญลักษณ์ของวงจรดังรูปที่ 2.8 มี A และ B เป็นอินพุตและ C เป็นเอาต์พุต โดยมี S เป็นขาควบคุมอินพุต (เมื่อ S = '0' เลือกอินพุต A ไปที่ C และ S = '1' เลือกอินพุต B ไปที่ C



รูปที่ 2.8 ตัววงจร 2 to 1 Multiplexer

ในรูปที่ 2.8 ประกอบด้วยวงจรย่อย 3 วงจรโดยมี X และ Y เป็นสายไฟเชื่อมต่ออยู่ภายใน ซึ่งเขียนสมการบูลีนได้ดังนี้

$X = \overline{S}.A$

$Y = S.B$

$C = X + Y$ หรือ $C = \overline{S}.A + S.B$

เราสามารถเขียนโค้ด VHDL ได้ดังรูปที่ 2.9a) หรือรูปที่ 2.9b) ในภาษา VHDL จะเรียก X หรือ Y นี้ว่า Signal หรือสัญญาณ ซึ่งชนิดข้อมูลของ Signal X และ Signal Y ในที่นี้จะเป็น std_logic โดยจะประกาศใช้ Signal (Signal declaration) ระหว่าง architecture และ begin ดังรูปที่ 2.9a) และจากรูปที่ 2.9a) จะเห็นได้ชัดเจนว่าคำสั่งย่อย 1 คำสั่งแทนวงจรย่อย 1 วงจร ดังนั้น Signal จึงทำหน้าที่ส่งผ่านหรือแลกเปลี่ยนข้อมูลระหว่างคำสั่งต่างๆ

```
2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4
5  entity MUX2TO1 is
6      port ( A,B : in  STD_LOGIC;
7             S : in  STD_LOGIC;
8             C : out  STD_LOGIC);
9  end MUX2TO1;
10
11 architecture BEHAVIORAL of MUX2TO1 is
12    signal X,Y : STD_LOGIC;
13 begin
14    X <= (not S and A);
15    Y <= S and B;
16    C <= X or Y;
17 end BEHAVIORAL;
```

การประกาศใช้ Signal X และ Sinal Y นั้นเราจะประกาศใช้เมื่อชื่อของ Signal นั้นไม่มีชื่อใน port ของ entity

2.9a) โค้ด VHDL วงจร 2 to 1 Multiplexer แบบประกาศใช้ Signal (มองเห็นเป็นวงจรย่อย 3 วงจร)

```
2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4
5  entity MUX2TO1 is
6      port ( A,B : in  STD_LOGIC;
7             S : in  STD_LOGIC;
8             C : out  STD_LOGIC);
9  end MUX2TO1;
10
11 architecture BEHAVIORAL of MUX2TO1 is
12    begin
13
14       C <= (not S and A)or(S and B);
15
16 end BEHAVIORAL;
```

2.9b) โค้ด VHDL วงจร 2 to 1 Multiplexer แบบไม่ประกาศใช้ Signal (มองเห็นเป็นวงจรใหญ่เพียงวงจรเดียว)

รูปที่ 2.9 โค้ด VHDL ของวงจร 2 to 1 Multiplexer ที่เขียนทั้งแบบประกาศใช้และไม่ประกาศใช้ Signal

2.7 ชนิดข้อมูล

ปัญหาในการเขียนโค้ด VHDL ที่เกิดกับผู้เริ่มต้นนั้น คือ ไม่รู้ว่าตัวดำเนินการ (Operator) ใดสามารถใช้กับชนิดข้อมูล (Type) ใดได้บ้าง ซึ่ง Standard package ของ IEEE Std 1076 และ Std_logic_1164 package ของ IEEE Std 1164 จะนิยามชนิดข้อมูลต่างๆและตัวดำเนินการที่เกี่ยวข้องรวมทั้งการนิยามฟังก์ชันต่างๆมาให้

2.7.1 ชนิดข้อมูลตามมาตรฐาน IEEE Std 1076

ซึ่งในเบื้องต้นนี้เราจะใช้ชนิดข้อมูล Scalar types และ Composite types เท่านั้น

2.7.1.1 ชนิดข้อมูล Scalar

ชนิดข้อมูล Scalar (Scalar types) ประกอบด้วย Enumeration types, Integer types, Physical types และ Floating point types โดยที่เราจะเรียก Enumeration types และ Integer types ว่า "Discrete types" และจะเรียก Integer types, Floating point types และ Physical types ว่า "Numeric types" ผู้อ่านควรจดจำชื่อชนิดข้อมูลเหล่านี้ไว้เพื่อจะได้ไม่สับสน

1) ชนิดข้อมูล Enumeration หรือ Enumeration types ที่กำหนดล่วงหน้าไว้แล้ว (Predefined) ตาม IEEE Std 1076 ที่นิยามไว้ใน Package ชื่อ Standard ที่ควรทราบเบื้องต้นได้แก่

- ชนิดข้อมูล character เป็นชนิดข้อมูลที่เป็นตัวอักษรต่างๆ ตาม ISO 8859-1:1987 มีทั้ง 256 ตัว
- ชนิดข้อมูล bit มีค่าเป็นลอจิก '0' และ '1'
- ชนิดข้อมูล boolean มีค่าเป็น เท็จ (หรือ FALSE) และ จริง (หรือ TRUE)

ตัวอย่างการนิยาม Enumeration type โดยผู้ใช้เอง (User defined)หรือนิยามไว้ใน Package ต่างๆ เช่น

```
type COLOR is (BLUE, GREEN, YELLOW, RED);
type BIT is ('0', '1');
```

2) ชนิดข้อมูล integer หรือ integer types ที่กำหนดล่วงหน้าไว้แล้วมีค่าอย่างน้อย -2147483647 ถึง 2147483647 โดยจะกำหนดไว้ล่วงหน้าเป็นเลขฐานสิบ เช่น -5 , -1 , 0 , 8, 100

การนิยาม Integer type โดยผู้ใช้เองหรือนิยามไว้ใน Package ต่างๆจะมีรูปแบบดังตัวอย่าง เช่น

```
type integer is range -2147483647 to 2147483647;
subtype NATURAL is integer range 0 to 2147483647;
subtype POSITIVE is integer range 1 to 2147483647;
type WORD_INDEX is range 31 downto 0;
```

การนิยามเป็นข้อมูลย่อย (Subtype) มีข้อดีกว่าการนิยามเป็น Type ใหม่ คือ สามารถทำ Operation ต่างๆโดยใช้ Operator เดียวกับ Base type ได้ แต่ถ้าเป็นคนละ Type กันจะไม่สามารถใช้ Operator ต่างๆร่วมกันได้

3)ชนิดข้อมูล Physical ที่กำหนดล่วงหน้าไว้แล้วคือชนิดข้อมูล time ซึ่ง Xilinx Synthesis Tool หรือ XST ที่มีอยู่ในซอฟต์แวร์ชุด ISE WebPACK และ ISE Foundation จะละเว้น)Ignored (โดยจะไม่สนใจชนิดข้อมูลนี้

4) ชนิดข้อมูล Floating point (Real) หรือ Real types ที่กำหนดล่วงหน้าไว้แล้วมีค่าอย่างน้อยอยู่ในช่วง -1.0E38 to +1.0E38 ชนิดข้อมูล Real เช่น 1.2, 10.0 (เป็นต้น)ถ้าเขียนเป็น 10 จะกลายเป็น integer)

2.7.1.2 ชนิดข้อมูล Composite

1) ชนิดข้อมูลอะเรย์ (Array) หรือ Array types เป็นชนิดข้อมูลที่นิสมาชิกหลายตัวที่เป็นชนิดข้อมูลเดียวกัน โดย Xilinx Synthesis Tool หรือ XST จะรองรับการใช้ชนิดข้อมูล Multi-dimentional array ได้สูงสุดถึง 3 มิติ

ชนิดข้อมูลอะเรย์ที่กำหนดล่วงหน้าไว้แล้วคือ string และ bit_vector โดยที่ string คือชนิดข้อมูลอะเรย์ 1 มิติ (one-dimensional arrays) ของ character และ bit_vector คือชนิดข้อมูลอะเรย์ 1 มิติของ bit

ตัวอย่างการนิยามชนิดข้อมูลอะเรย์ 1 มิติโดยผู้ใช้เองหรือนิยามไว้ใน Package ต่างๆจะมีรูปแบบเช่น

```
type BIT_VECTOR is array (NATURAL range <>) of BIT;
type STRING is array (POSITIVE range <>) of CHARACTER;
type NIBBLE is array (3 downto 0) of BIT;
subtype BYTE is bit_vector (7 downto 0);
```

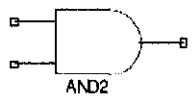
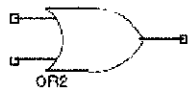
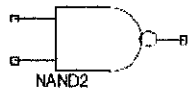
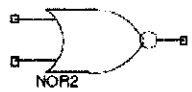
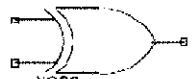
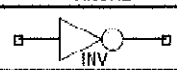
2) ชนิดข้อมูล record หรือ record types จะเหมือนชนิดข้อมูลอะเรย์แต่เป็นของชนิดข้อมูลต่างชนิดกัน

ภาคผนวก ค

ลอจิกเกตและวงจรคอมบิเนชัน (ยังไม่ครบทุกการทดลอง)

3.1 ลอจิกเกต

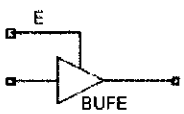
ลอจิกเกตต่างๆและสมการบูลีน (Logic gates and boolean expression) สรุปดังรูปที่ 3.1 สำหรับคนที่มีความรู้พื้นฐานดิจิตอลมาแล้วก็จะเข้าใจตารางลอจิกเกตต่างๆได้เป็นอย่างดี

LOGIC	SYMBOL	BOOLEAN EXPRESSION	INPUT A	INPUT B	OUTPUT Y
AND	 AND2	A . B = Y	0	0	0
			0	1	0
			1	0	0
			1	1	1
OR	 OR2	A + B = Y	0	0	0
			0	1	1
			1	0	1
			1	1	1
NAND	 NAND2	$\overline{A . B} = Y$	0	0	1
			0	1	1
			1	0	1
			1	1	0
NOR	 NOR2	$\overline{A + B} = Y$	0	0	1
			0	1	0
			1	0	0
			1	1	0
XOR	 XOR2	$A \oplus B = Y$	0	0	0
			0	1	1
			1	0	1
			1	1	0
XNOR	 XNOR2	$\overline{A \oplus B} = Y$	0	0	1
			0	1	0
			1	0	0
			1	1	1
INVERTER	 INV	$\overline{A} = Y$	0		1
			1		0

รูปที่ 3.1 ลอจิกเกตต่างๆ

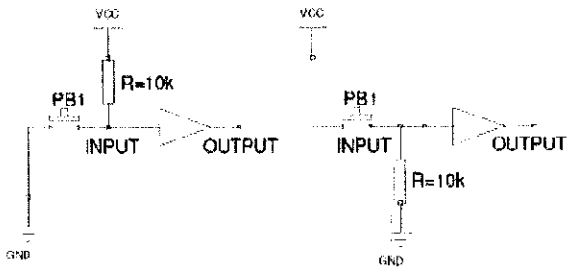
นอกจากลอจิกเกตต่างๆแล้วยังมีบัฟเฟอร์ (Buffer) และไตร-สเตตบัฟเฟอร์ (Tri-state buffer) อีกด้วยแสดงดังรูปที่ 3 และรูปที่ 3.3 ตามลำดับ อินพุตและเอาต์พุตของบัฟเฟอร์จะมีลอจิกเหมือนกัน ยกเว้นไตร-สเตตบัฟเฟอร์ดังตัวอย่างในรูปที่ 3 ถ้าขา E = '0' เอาต์พุตจะเป็น 'Z' หรือ High impedance แต่ถ้าขา E = '1' อินพุตและเอาต์พุตจะมีลอจิกเหมือนกัน

รูปที่ 3.2 บัฟเฟอร์ (Buffer)



รูปที่ 3.3 ไตร-สเตตบัฟเฟอร์ (Tri-state buffer) แบบ Active high (E = '1')

อินพุตของไอซีลอจิกเกตไม่ว่าเป็นตระกูล TTL หรือ CMOS หรือแม้กระทั่ง FPGA และ CPLD ที่รับจากสวิตช์คีย์บอร์ด หรือ หน้าสัมผัสของรีเลย์ ในทางปฏิบัตินั้นเราจะใช้ตัวต้านทานพูลอัพ (Pull up) ต่อเข้ากับแหล่งจ่าย (Vcc) หรือตัวต้านทานพูลดาวน์ (Pull down) ต่อเข้ากับกราวด์ (GND) เพื่อบังคับให้อินพุตมีลอจิกเป็น '1' หรือ '0' ในขณะที่ยังไม่กดสวิตช์ แสดงดังรูปที่ 3.4 การปล่อยอินพุตลอยไว้ (Floating) อาจให้ค่าลอจิกไม่แน่นอนเนื่องจากสัญญาณรบกวนต่างๆอาจทำให้อินพุตเปลี่ยนสถานะได้ อินพุตที่ไม่ได้ใช้งานควรต่อเข้ากับ Vcc หรือ GND โดยตรงเพื่อบังคับให้อินพุตนั้นมีลอจิกเป็น '1' หรือ '0' ยกเว้นขาอินพุตเอาต์พุตของ FPGA หรือ CPLD ที่ไม่ได้กำหนดให้ใช้งานนั้นสามารถปล่อยขาลอยได้ ในทางปฏิบัติตัวต้านทานพูลอัพหรือพูลดาวน์ที่ใช้มีค่าประมาณ 2,200 ถึง 10,000 โอห์ม ซึ่งผู้อ่านควรต้องศึกษารายละเอียดจากดาต้าชีตของผู้ผลิต



a) b)

รูปที่ 3.4 วงจรลอจิกเกตต่างๆ ที่ต่อขาอินพุตเข้ากับตัวต้านทานพูลอัพหรือพูลดาวน์

จากรูปที่ 3.4a) ถ้าไม่กดสวิตช์ อินพุตจะเป็นลอจิก '1' แต่ถ้ากดจะเป็นลอจิก '0' (เรียกว่า Active Low) ส่วนในรูปที่ 3.4b) ถ้าไม่กดสวิตช์ อินพุตจะเป็นลอจิก '0' แต่ถ้ากดจะเป็นลอจิก '1' (เรียกว่า Active high) บอร์ดทดลองทุกรุ่นที่ใช้อ้างอิงในหนังสือเล่มนี้จะใช้สวิตช์ที่เป็นปุ่มกดหรือดิพสวิตช์แบบ Active low ทั้งหมด

การทดลองที่ 3.1.1 ลอจิกเกต

วัตถุประสงค์

1.เพื่อทำความเข้าใจเกี่ยวกับแอนด์เกต ออร์เกต และ อินเวอร์เตอร์

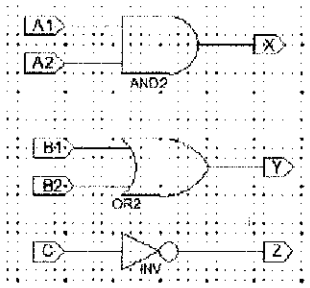
2.เพื่อทดลองใช้ ISE WebPACK 8.1i สร้างลอจิกเกตและโปรแกรมลงชิพ CPLD และ FPGA

อุปกรณ์ทดลอง

บอร์ด CPLD Explorer XC9572XL หรือ CPLD Explorer XC9572 และ FPGA Discovery-III XC3S200F หรือ FPGA Discovery-III XC3S200F4

1.สร้างแอนด์เกต ออร์เกต และ อินเวอร์เตอร์ด้วย CPLD

ให้ออกแบบวงจรสร้างแอนด์เกต ออร์เกต และ อินเวอร์เตอร์ด้วย CPLD ดังรูปที่ L1.1 ตามขั้นตอนที่อธิบายในบทที่ 2



a)

```
 2  library IEEE;
 3  use IEEE.STD_LOGIC_1164.ALL;
 4
 5  entity ex3_1_1vcxl is
 6      Port ( A1,A2 : in  STD_LOGIC;
 7             B1,B2 : in  STD_LOGIC;
 8             C : in  STD_LOGIC;
 9             X : out  STD_LOGIC;
10             Y : out  STD_LOGIC;
11             Z : out  STD_LOGIC);
12  end ex3_1_1vcxl;
13
14  architecture Behavioral of ex3_1_1vcxl is
15     begin
16        X <= A1 and A2;
17        Y <= B1 or B2;
18        Z <= not C;
19  end Behavioral;
```

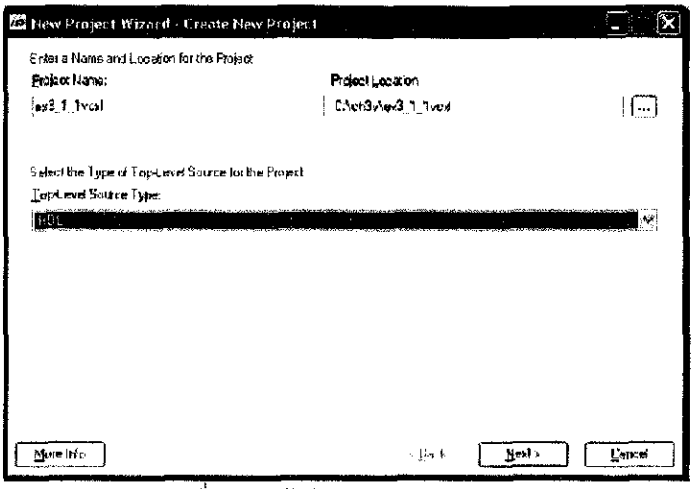
b)

รูปที่ L1.1 วงจรลอจิกเกตต่างๆ

1.1 ขั้นตอนการออกแบบวงจร (Design Entry)

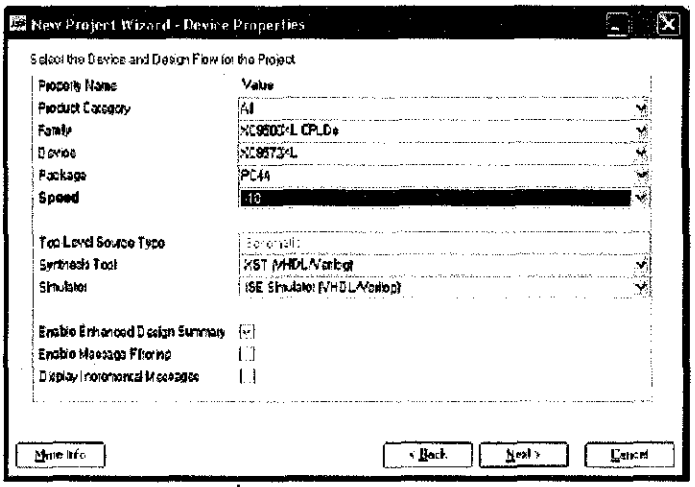
1) ที่หน้าจอคอมพิวเตอร์ ก่อนเข้าโปรแกรม ISE WebPACK ให้สร้าง Folder ชื่อ ch3v ไว้ในไดรฟ์ C ดับเบิลคลิ๊ก แล้วจะได้หน้าต่าง Xilinx-ISE (ถ้ามีหน้าต่าง Tip of the Day ซ้อนขึ้นมาให้คลิ๊ก OK) จากนั้นคลิ๊ก File -> New Project แล้วจะได้หน้าต่าง (หรือ Dialog box) New Project Wizard

2) สร้างโปรเจคต์ไฟล์ไว้ที่ Project Location (หรือ Folder) ชื่อ ch3v และ Project Name ชื่อ ex3_1_1vvcxl ตามลำดับ คลิ๊กที่ Top-Level Module Type เป็น HDL แสดงดังรูปที่ L1.2 คลิ๊ก Next แล้วจะได้หน้าต่างถัดไป

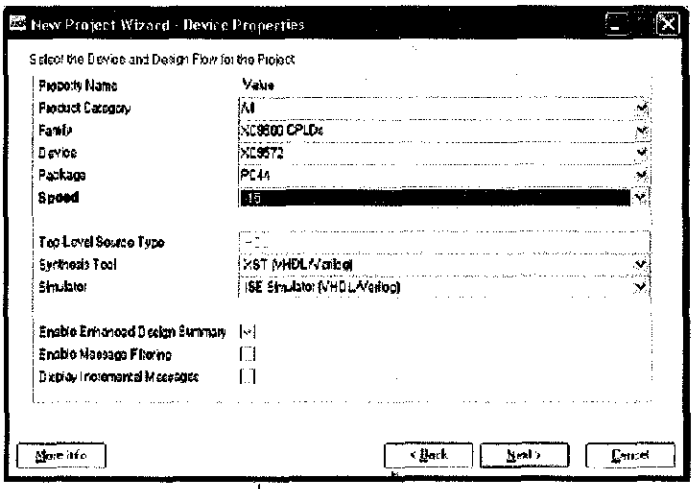


รูปที่ L1.2 หน้าต่าง New Project Wizard

3) ที่หน้าต่าง New Project Wizard คลิ๊กเลือกชิพที่ต้องการดังรูปที่ L1.3 โดยใช้ CPLD ตระกูล XC9500XL เบอร์ XC9572X มี Package แบบ PLCC 44 ขา (Package: PC44) และ Speed Grade: -10 ที่ใช้บนบอร์ด CPLD Explorer XC9572XL (กรณีใ บอร์ด CPLD Explorer XC9572 ให้คลิ๊กเลือก CPLD ตระกูล XC9500 เบอร์ XC9572 Package: PC44 และ Speed Grade: -15



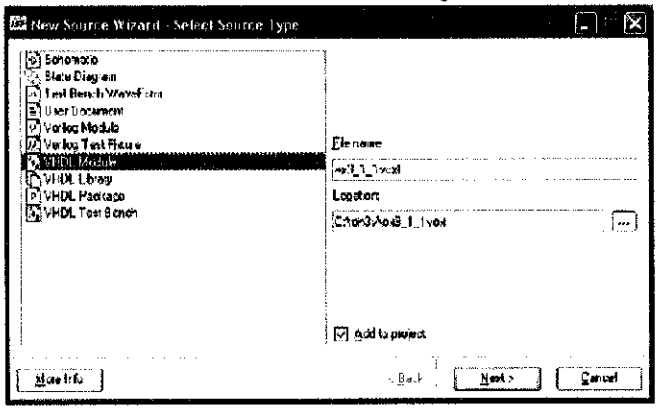
a) CPLD ที่ใช้กับบอร์ด CPLD Explorer XC9572XL



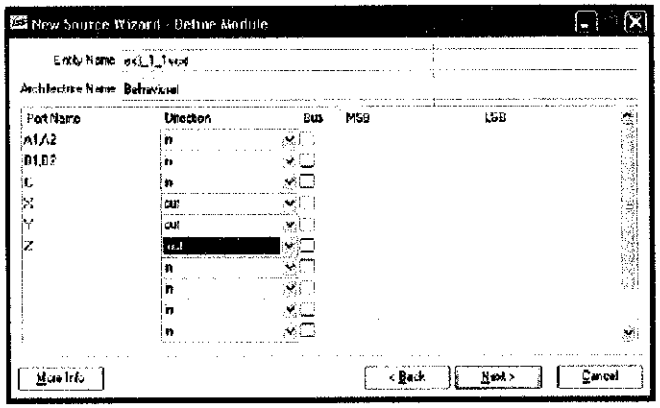
b) CPLD ที่ใช้กับบอร์ด CPLD Explorer XC9572

รูปที่ L1.3 หน้าต่าง New Project Wizard – Device Properties

4) คลิกปุ่ม Next ในรูปที่ L1.3 แล้วจะได้หน้าต่างถัดไป คลิกปุ่ม New Source แล้วจะได้หน้าต่างถัดไป จากนั้นพิมพ์ชื่อ Source File ชื่อ ex3_1_1vvexl ลงในช่อง File Name และคลิกที่ VHDL Module ดังรูปที่ L1.4a) คลิก Next แล้วใส่อินพุต A1,A2, B1,B2 และ C และเอาต์พุต X,Y และ Z ดังรูปที่ L1.4b) แล้วคลิก Next จากนั้นคลิก Finish คลิก Next อีก 2 ครั้งและคลิก Finish แล้วจะได้หน้าต่าง Xilinx-ISE จากนั้นเขียนโค้ด VHDL ดังวงจรตามรูปที่ L1.1 จนแล้วเสร็จดังรูปที่ L1.5

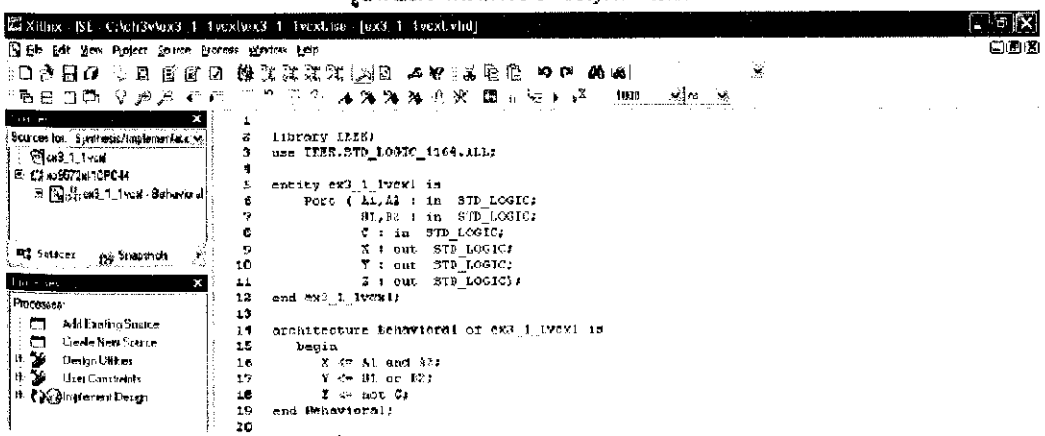


a)



b)

รูปที่ L1.4 หน้าต่าง New Project Wizard

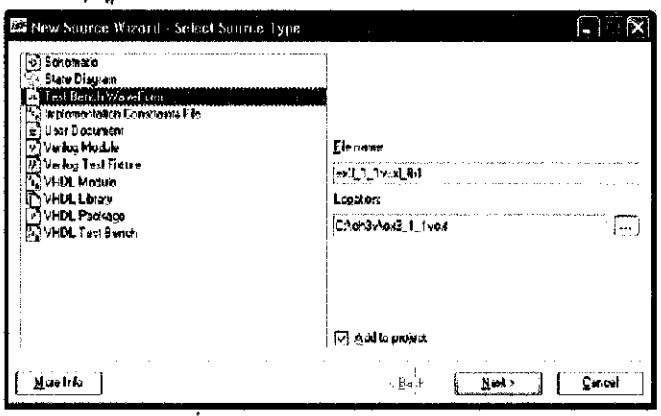


รูปที่ L1.5 หน้าต่าง Xilinx– ISE สำหรับวาดวงจร

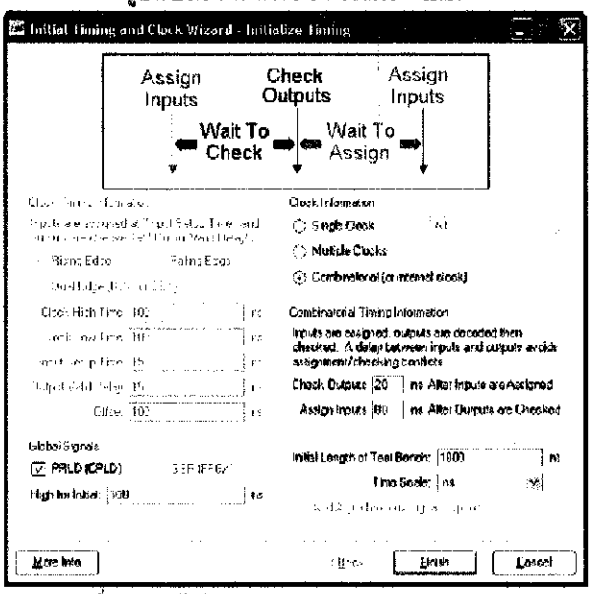
5) การตรวจความถูกต้อง ถ้าไม่มีข้อผิดพลาดก็ให้บันทึกไฟล์ คลิก ✕ (สีดำ) ปิดโปรแกรมเพื่อกลับไปที่หน้าต่าง Xilinx-ISE

1.2 การตรวจสอบวงจรที่ออกแบบ (Design verification) ด้วย Behavioral simulation

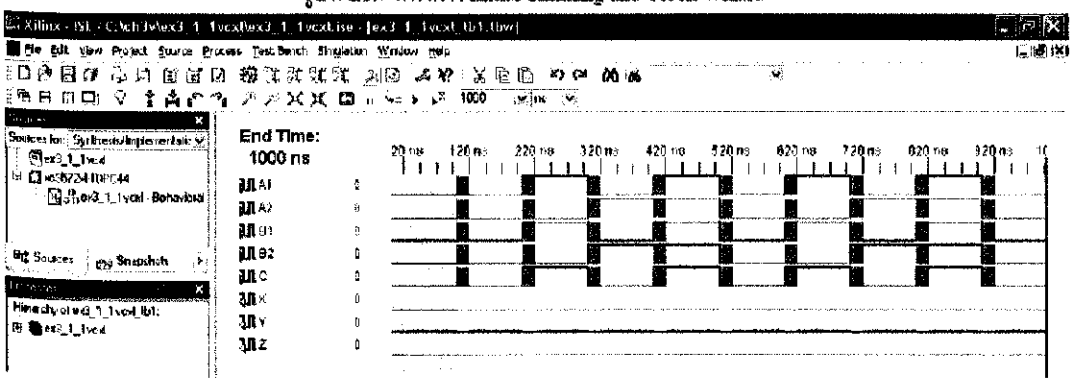
ให้ทำตามขั้นตอนในบทที่ 2 ข้อ 2.18 (ในหนังสือเล่มที่ออกแบบด้วยวิธีวาดสังวงจร) พิมพ์ชื่อ ex3_1_1vvexl_tb1 ดังรูปที่ L1.6 จากนั้นกำหนดค่าเวลาและ Waveform ดังรูปที่ L1.7 และรูปที่ L1.8 เสร็จแล้วให้พิจารณาว่า Behavioral simulation ของวงจรที่ออกแบบเป็นไปตามหลักฐานหรือไม่



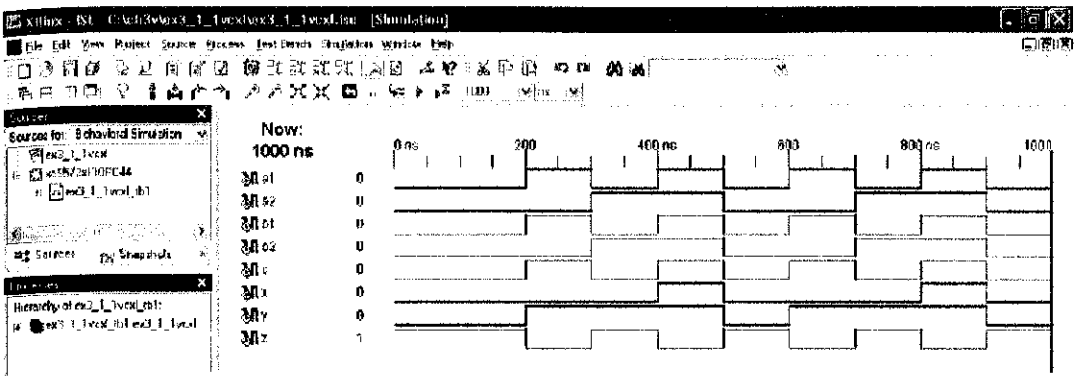
รูปที่ L1.6 หน้าต่าง New Source Wizard



รูปที่ L1.7 หน้าต่าง Initial Timming and Clock Wizard



a)

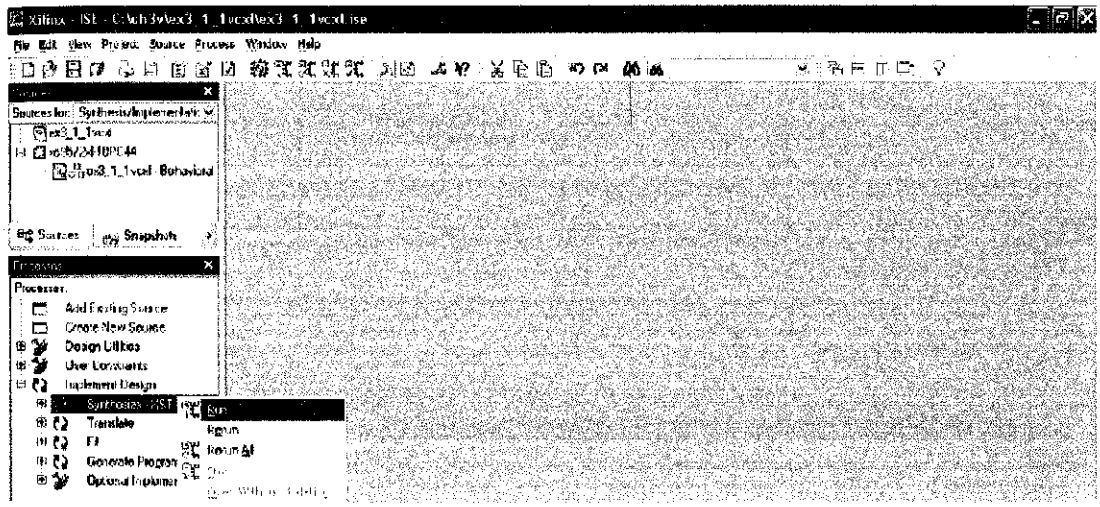


b)

รูปที่ L1.8 หน้าต่างสำหรับกำหนดสัญญาณต่างๆที่ป้อนให้กับอินพุตของวงจรที่เราต้องการทดสอบและผลที่ได้

1.3 การสังเคราะห์วงจร (Design synthesis)

ที่หน้าต่าง Processes คลิก "+" ที่อยู่หน้า Implement Design จนกลายเป็น "-" จากนั้นคลิกขวา Synthesize–XST แล้วคลิก RUN ดังรูปที่ L1.9 เสร็จแล้วถ้าโชว์ ◯ หรือ ⚠ ที่หน้า Synthesize–XST ถือว่าขั้นตอนสังเคราะห์วงจรผ่าน



รูปที่ L1.9 ขั้นตอนการสังเคราะห์วงจร

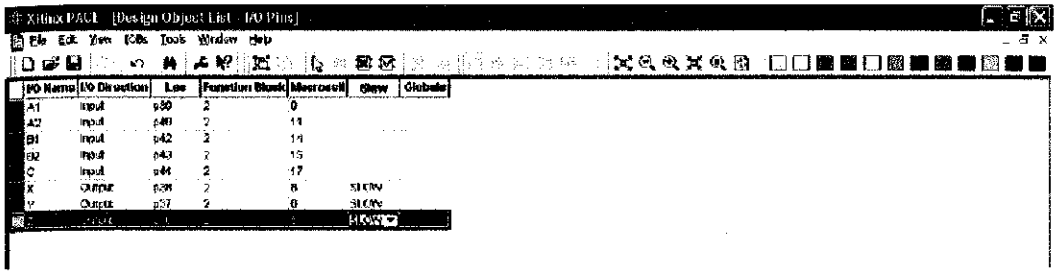
1.4 Design implementation

1) สร้าง Implementation constraints file (UCF) โดยอัตโนมัติ

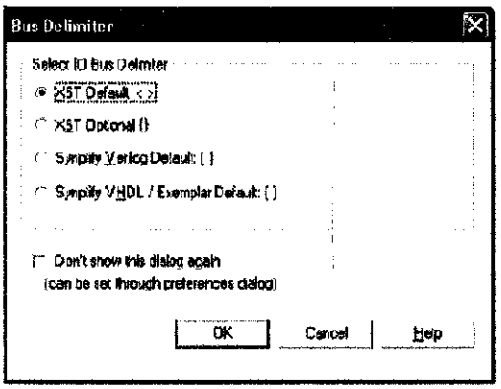
2) ขั้นตอนกำหนดสายสัญญาณต่างๆของวงจรในรูปที่ L1.1 เข้ากับขาของ CPLD จำเป็นต้องกำหนดให้อินพุตและเอาต์พุตของ CPLD สอดคล้องกับอุปกรณ์ที่เตรียมไว้ที่บอร์ดทดลองดังมีรายละเอียดตามตารางที่ 1.6 ในบทที่ 1 ซึ่งจะใช้ปุ่มกด (Push button switch) PB1 ถึง PB5 เป็นอินพุตและ LED1 ถึง LED3 เป็นเอาต์พุต กล่าวคือ

A1 = PB1 = INPUT = p39	X = LED1 = OUTPUT = p38
A2 = PB2 = INPUT = p40	Y = LED2 = OUTPUT = p37
B1 = PB3 = INPUT = p42	Z = LED3 = OUTPUT = P36
B2 = PB4 = INPUT = p43	
C = PB5 = INPUT = p44	

ให้พิมพ์รายละเอียดต่างๆใน Assign Package Pins เมื่อพิมพ์เสร็จแล้วจะได้ดังรูปที่ L1.10a) จากนั้นบันทึกไฟล์โดยคลิ๊ก แล้วได้หน้าต่างซ้อนขึ้นมา ให้คลิ๊ก XST Default และคลิ๊ก OK จากนั้นคลิ๊ก (สีแดง)เพื่อปิดโปรแกรม Assign Package Pins เพื่อกลับไปที่หน้าต่าง Xilinx – ISE อีกครั้ง



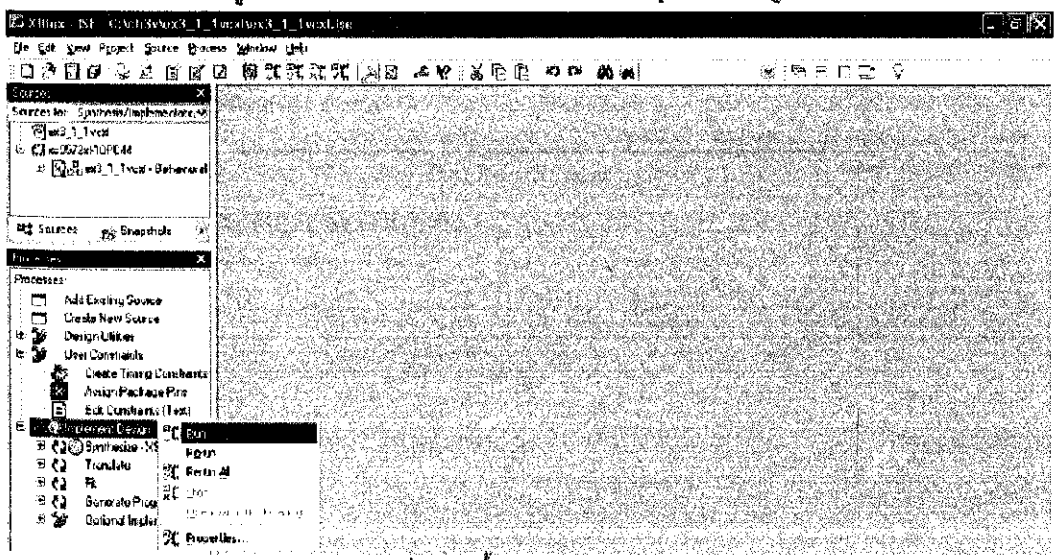
a)



b)

รูปที่ L1.10 การกำหนดขาในหน้าต่าง Edit Constraints (Text)

3) ขั้นตอน Implementation คลิ๊กชื่อ ex3_1_1vvexl ในหน้าต่าง Source และคลิ๊กขวาที่ Implement Design ในหน้าต่าง Processes แล้วคลิ๊ก RUN ดังรูปที่ L1.11 เสร็จแล้วโชว์ หรือ ที่หน้า Implement Design



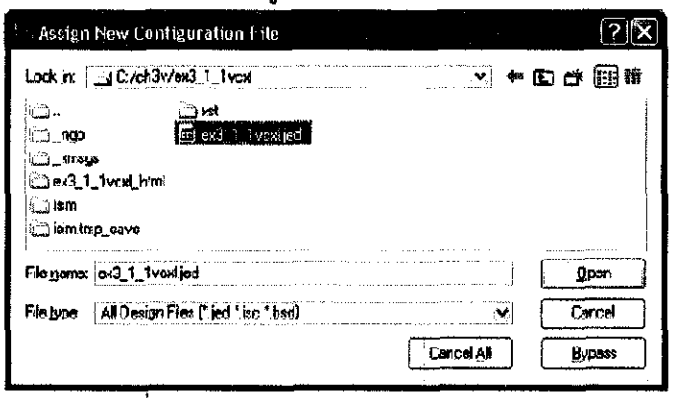
รูปที่ L1.11 ขั้นตอน Implementation

1.5 การโปรแกรมข้อมูลวงจรลงชิพ

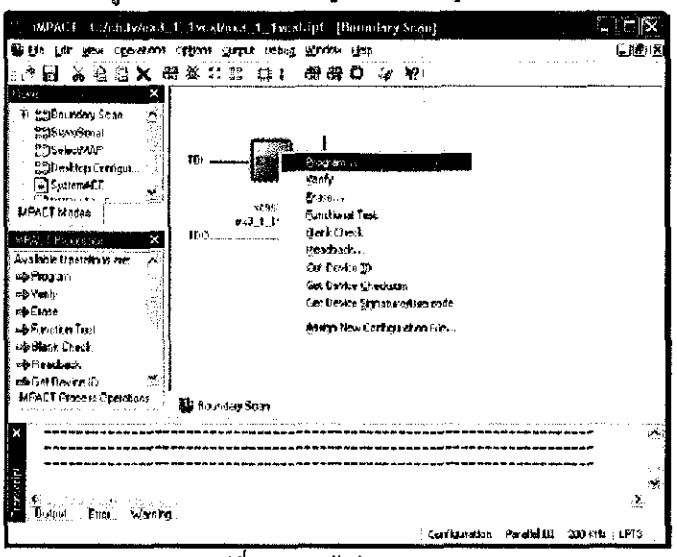
1) ต่อสายดาวน์โหลดเข้าพอร์ตขนานของคอมพิวเตอร์และขั้ว JTAG ที่บอร์ด CPLD Explorer XC9572XL พร้อมทั้งจ่ายไฟเลี้ยง

2) ที่หน้าต่าง Processes คลิ๊ก Configure Device (iMPACT) และคลิ๊ก RUN แล้วจะได้หน้าต่างถัดไป คลิ๊ก Finish แล้วจะได้หน้าต่างถัดไป คลิ๊กไฟล์ชื่อ ex3_1_1vvcxl.jed แล้วคลิ๊ก Open ดังรูปที่ L1.12 แล้วจะได้หน้าต่างถัดไป คลิ๊กขวาที่ตัวชิพจนเป็นสีเขียวแล้วคลิ๊กที่เมนู Program... ดังรูปที่ L1.13 จากนั้นข้อมูลวงจรจะถูกดาวน์โหลดลงชิพ เสร็จแล้วปิดหน้าต่าง iMPACT โดยคลิ๊ก แล้วจะได้หน้าต่าง Exit iMPACT คลิ๊ก No จากนั้นคลิ๊ก เพื่อออกจากโปรแกรม ISE WebPACK

3)หลังจากโปรแกรมแล้วให้ทดลองกด ปุ่ม PB1-PB5 และให้สังเกตดูแอลอีดี LED1-LED3 ว่าให้ลอจิกเอาต์พุตเป็นไปตามทฤษฎีหรือไม่ จากนั้นบันทึกผลการทดลองและต้องพึงระลึกอยู่เสมอว่า Push button switch ทำงานเป็นแบบ Active low



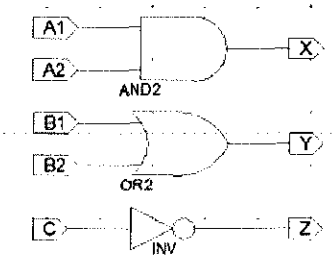
รูปที่ L1.12 หน้าต่าง Assign New Configuration File



รูปที่ L1.13 หน้าต่าง iMPACT

2.สร้างแอนด์เกต ออร์เกต และ อินเวอร์เตอร์ด้วย FPGA

ให้ออกแบบวงจรสร้างแอนด์เกต ออร์เกต และ อินเวอร์เตอร์ด้วย FPGA ดังรูปที่ L2.1 ตามขั้นตอนที่อธิบายในบทที่ 2



a)

```
2   library IEEE;
3   use IEEE.STD_LOGIC_1164.ALL;
4
5   entity ex3_1_1vf is
6       Port ( A1,A2 : in  STD_LOGIC;
7              B1,B2 : in  STD_LOGIC;
8              C : in  STD_LOGIC;
9              X : out  STD_LOGIC;
10             Y : out  STD_LOGIC;
11             Z : out  STD_LOGIC);
12  end ex3_1_1vf;
13
14  architecture Behavioral of ex3_1_1vf is
15     begin
16        X <= A1 and A2;
17        Y <= B1 or B2;
18        Z <= not C;
19  end Behavioral;
```

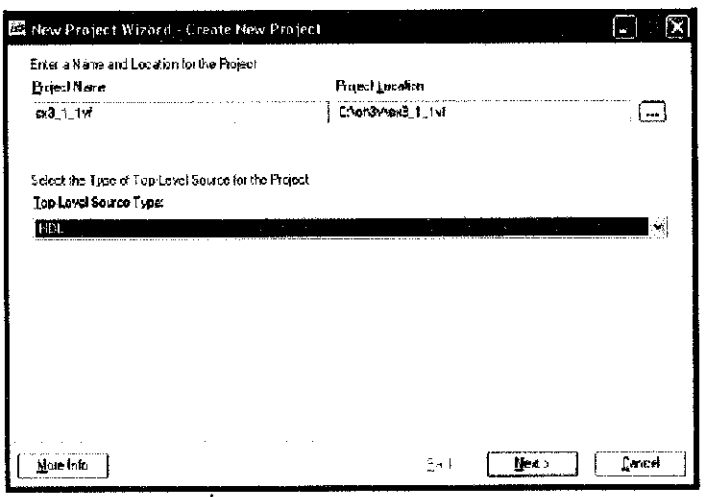
b)

รูปที่ L2.1 วงจรลอจิกเกตต่างๆ

2.1 ขั้นตอนการออกแบบวงจร (Design Entry)

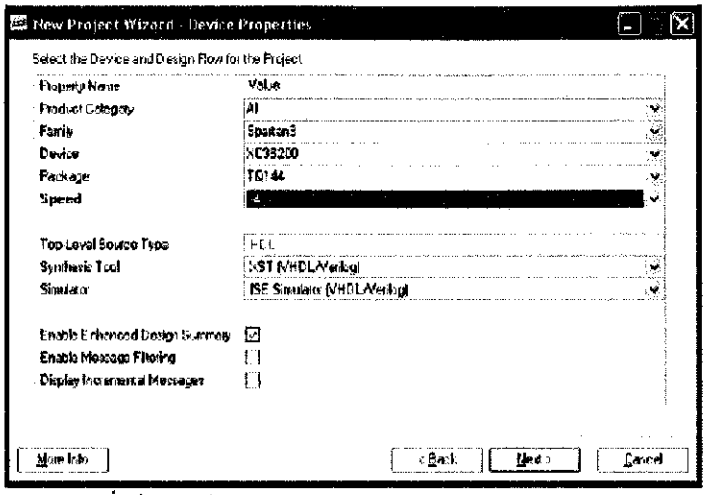
1) ที่หน้าจอคอมพิวเตอร์ ดับเบิลคลิ๊ก Xilinx ISE แล้วจะได้หน้าต่าง Xilinx-ISE (ถ้ามีหน้าต่าง Tip of the Day ซ้อนขึ้นมาให้กด OK) จากนั้นคลิ๊ก File -> New Project แล้วจะได้หน้าต่าง (หรือ Dialog box) New Project Wizard

2) สร้างโปรเจคต์ไฟล์ไว้ที่ Project Location (หรือ Folder) ชื่อ ch3v และ Project Name ชื่อ ex3_1_1vf ตามลำดับ คลิ๊กที่ Top Level Module Type เป็น HDL แสดงดังรูปที่ L2.2 คลิ๊ก Next แล้วจะได้หน้าต่างถัดไป

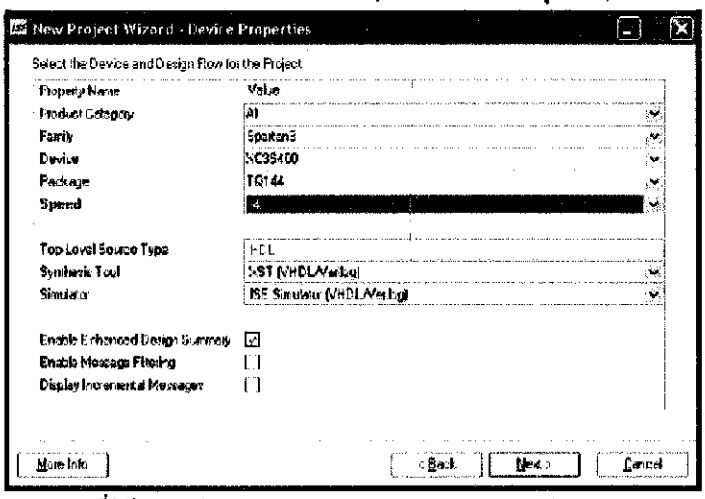


รูปที่ L2.2 หน้าต่าง New Project Wizard

3) ที่หน้าต่าง New Project Wizard–Device Properties เลือกชิพดังรูปที่ L2.3



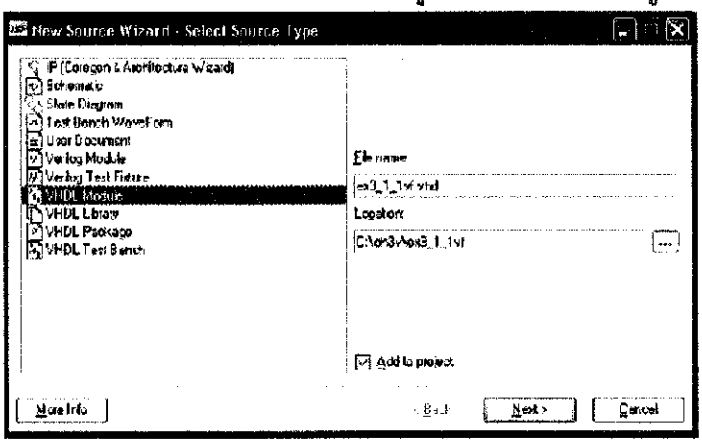
a) FPGA ที่ใช้กับบอร์ด FPGA Discovery-III XC3S200F (รุ่น 200,000 เกต)



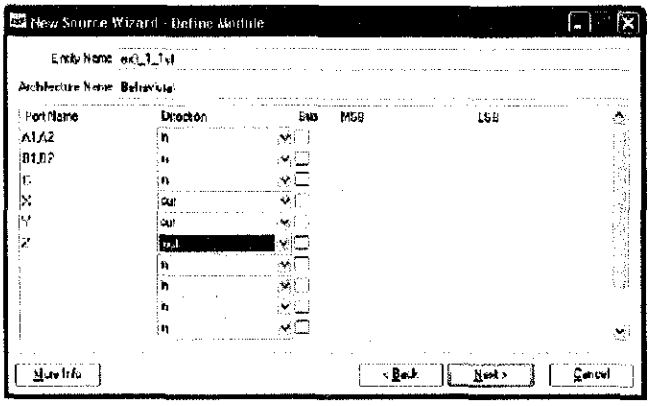
b)FPGA ที่ใช้กับบอร์ด FPGA Discovery-III XC3S200F4 (รุ่น 400,000 เกต)

รูปที่ L2.3 หน้าต่าง New Project Wizard--Device Properties

4) คลิ๊กปุ่ม Next ในรูปที่ L2.3 แล้วจะได้หน้าต่างถัดไป คลิ๊กปุ่ม New Source แล้วจะได้หน้าต่างถัดไป จากนั้นพิมพ์ชื่อ S File ชื่อ ex3_1_1vf ลงในช่อง File Name และคลิ๊กที่ VHDL Module ดังรูปที่ L2.4a) คลิ๊ก Next แล้วแล้วใส่อินพุต A B1,B2 และ C และเอาต์พุต X,Y และ Z ดังรูปที่ L2.4b) แล้วคลิ๊ก Next จากนั้นคลิ๊ก Finish คลิ๊ก Next อีก 2 ครั้งแล Finish แล้วจะได้หน้าต่าง Xilinx-ISE จากนั้นเขียนโค๊ดสังวงจรตามรูปที่ L2.1 จนแล้วเสร็จดังรูปที่ L2.5

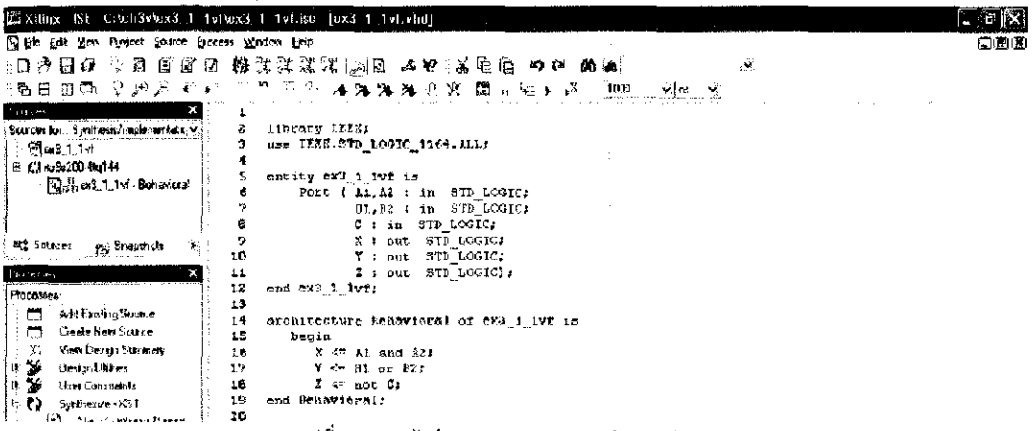


a)



b)

รูปที่ L2.4 หน้าต่าง New Project Wizard – Select Source Type

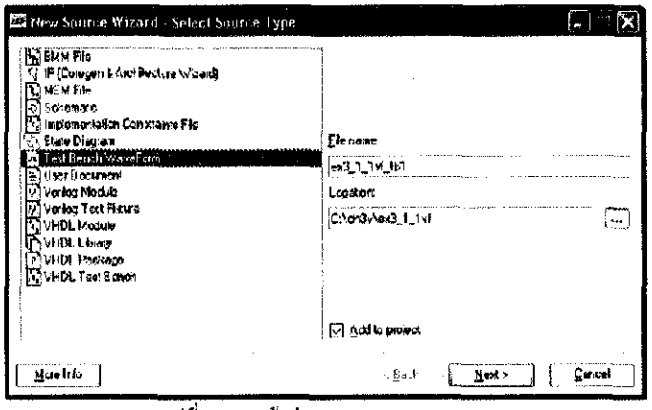


รูปที่ L2.5 หน้าต่าง Xilinx– ISE สำหรับวาดผังวงจร

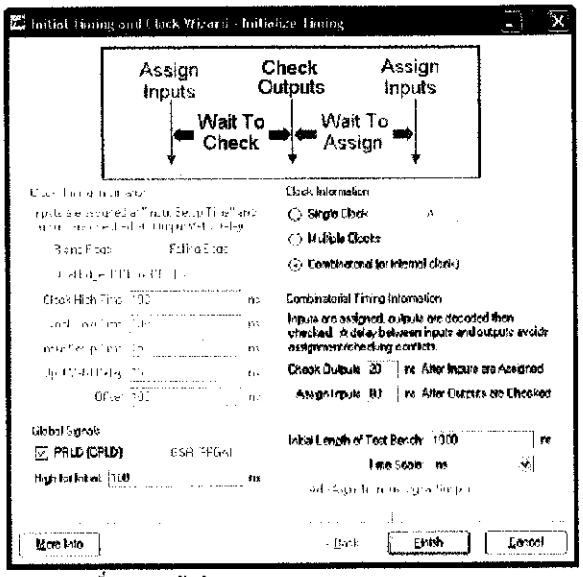
5) การตรวจความถูกต้อง ถ้าไม่มีข้อผิดพลาดก็ให้บันทึกไฟล์ คลิก ✕ (สีดำ) ปิดโปรแกรมเพื่อกลับไปที่หน้าต่าง Xilinx-ISE

2.2 การตรวจสอบวงจรที่ออกแบบ (Design verification) ด้วย Behavioral simulation

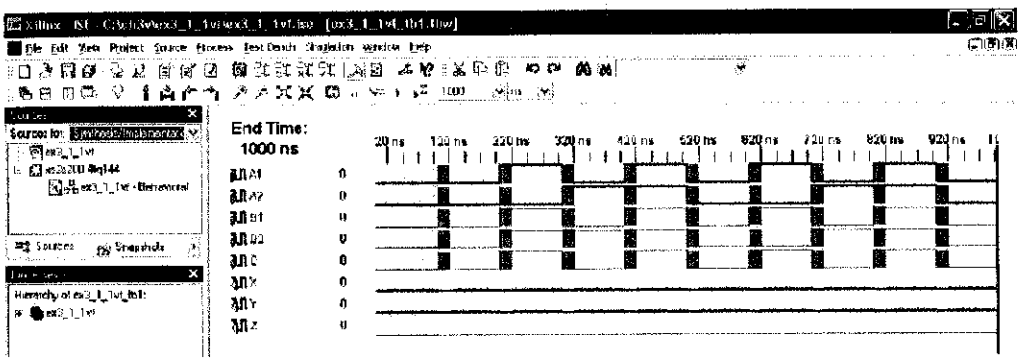
ให้ทำตามขั้นตอนในบทที่ 2 ข้อ 2.18 (ในหนังสือเล่มที่ออกแบบด้วยวิธีวาดผังวงจร) พิมพ์ชื่อ ex3_1_1vf_tb1 ดังรูปที่ L2.6 จากนั้นกำหนดค่าเวลาและ Waveform ดังรูปที่ L2.7 และ รูปที่ L2.8 เสร็จแล้วให้พิจารณาว่า Behavioral simulation ของวงจรที่ออกแบบเป็นไปตามทฤษฎีหรือไม่



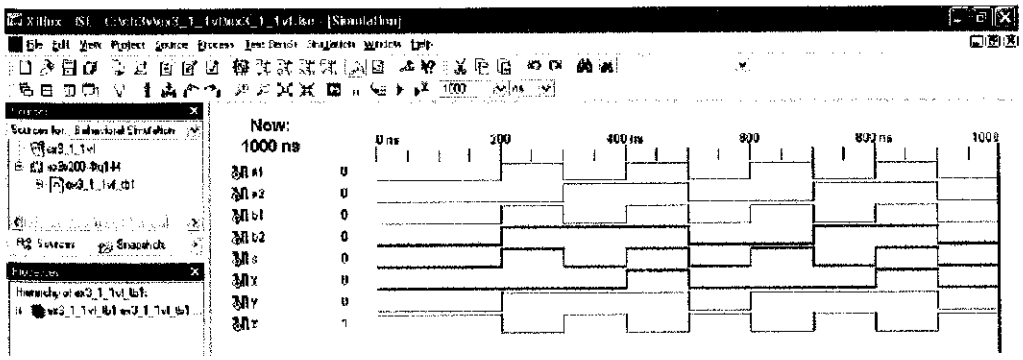
รูปที่ L2.6 หน้าต่าง New Source Wizard



รูปที่ L2.7 หน้าต่าง Initial Timming and Clock Wizard



a)



b)

รูปที่ L2.8 หน้าต่างสำหรับกำหนดสัญญาณต่างๆที่ป้อนให้กับอินพุตของวงจรที่เราต้องการทดสอบและผลที่ได้

วงจรถอดรหัสเป็นวงจรคอมบิเนชั่นเช่นกัน วงจรถอดรหัส เช่น 2 to 4 Decoder และ วงจรถอดรหัส BCD เป็นเซเวนเซกเมนต์ (BCD to 7 segment Decoder) เป็นต้น รูปที่ 3.10 แสดงตารางความจริงของวงจร 2 to4 Decoder ดังนั้นถ้าขา E = '0' จะทำให้เอาต์พุต D0 ถึง D3 = '0' แต่ถ้าขา E = '1' เอาต์พุต D0 ถึง D3 จะเป็นไปตามตารางความจริง

Input			Output			
E	A1	A0	D3	D2	D1	D0
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

รูปที่ 3.10 ตารางความจริงของวงจร 2 to 4 Decoder

จากตารางความจริงเขียนสมการ SOP ได้ดังนี้ (เอาต์พุตเป็นลอจิก '1')

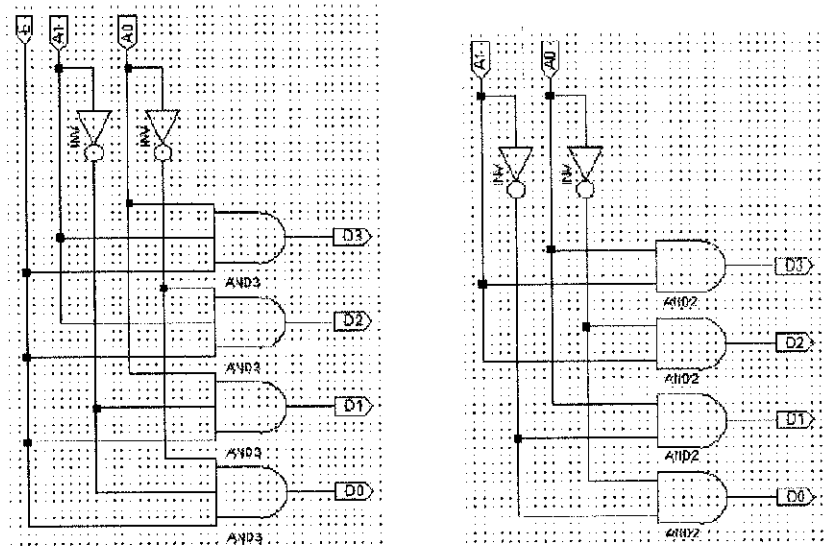
$D0 = E \cdot \overline{A0} \cdot \overline{A1}$

$D1 = E \cdot \overline{A0} \cdot A1$

$D2 = E \cdot A0 \cdot \overline{A1}$

$D3 = E \cdot A0 \cdot A1$

จากสมการนี้จะได้วงจรดังรูปที่ 3.11a) และ ในกรณีที่วงจรไม่มีขา E หรือ E = '1' นั้นวงจร 2 to 4 Decoder จะลดรูปแสดงดังรูปที่ 3.11b)

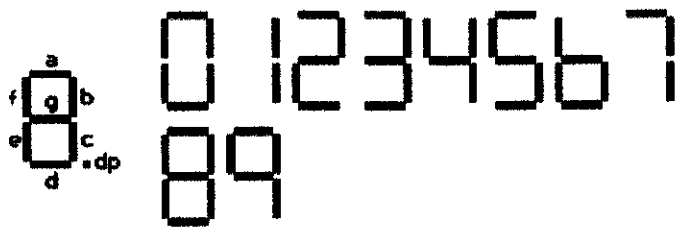


a) b)

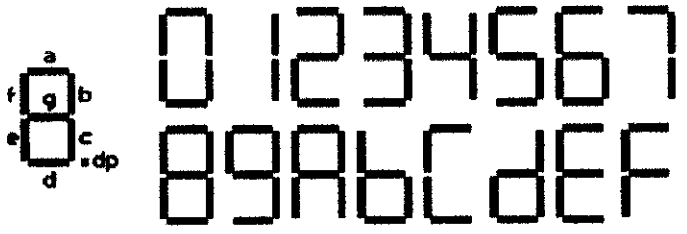
รูปที่ 3.11 วงจร 2 to4 Decoder

วงจรถอดรหัสอีกประเภทหนึ่งที่นิยมใช้กันมาก คือ วงจรถอดรหัสของตัวแสดงผลเซเวนเซกเมนต์ (7-Segment LED display) ได้แก่ วงจรถอดรหัสตัวแสดงผลเซเวนเซกเมนต์ (7-Segment decoder) ที่อยู่ในรูปแบบเลขฐานสิบ (Decimal digit display format) และรูปแบบเลขฐานสิบหก (Hexadecimal digit display format) แสดงดังรูปที่ 3.12a) และรูปที่ 3.12b) ตามลำดับ ซึ่งตัวแสดงผลเซเวนเซกเมนต์ประกอบด้วย LED แสดงผล 7 ส่วนคือ a , b , c , d , e , f , g และจุด (dp = Decimal point)

ความแตกต่างระหว่างรูปแบบเลขฐานสิบและฐานสิบหกจะอยู่ที่เลข 6 และเลข 9 ที่มีส่วนหรือของเซกเมนต์ (Segment) a และ d เพิ่มเข้ามาตามลำดับเพื่อทำให้เลข 6 ไม่ไปซ้ำกับตัว b และรูปแบบเลขฐานสิบหกจะมี A, b, C, d, E, F และ g เพิ่มเข้ามา



a)รูปแบบเลขฐานสิบ



b)รูปแบบเลขฐานสิบหก

รูปที่ 3.12 รูปแบบการแสดงผลบนตัวแสดงผลเซเวนเซกเมนต์

ตัวแสดงผลเซเวนเซกเมนต์มี 2 ชนิด คือ แบบแอโนดร่วม (Common Anode) และแบบแคโถดร่วม (Common cathod ตัวแสดงผลแบบแคโถดร่วมนั้นแต่ละส่วน (Segment) จะติดสว่างได้ก็ต่อเมื่อมีการส่งสัญญาณลอจิก '1' ให้ LED แต่ละส่ว โดยต่อผ่านตัวต้านทานเพื่อจำกัดกระแส) และส่งสัญญาณให้ขาร่วม (Common) เป็นลอจิก '0' หรือต่อลงกราวด์ แต่ถ้าตั แสดงผลเป็นแบบแอโนดร่วมก็จะจะทำงานตรงข้ามกัน

ตารางความจริงของวงจรถอดรหัสตัวแสดงผลเซเวนเซกเมนต์แบบเลขฐานสิบและเลขฐานสิบหกแบบแคโถดร่วมนั้น แสดงดังรูปที่ 3.13 และรูปที่ 3.14 ตามลำดับ

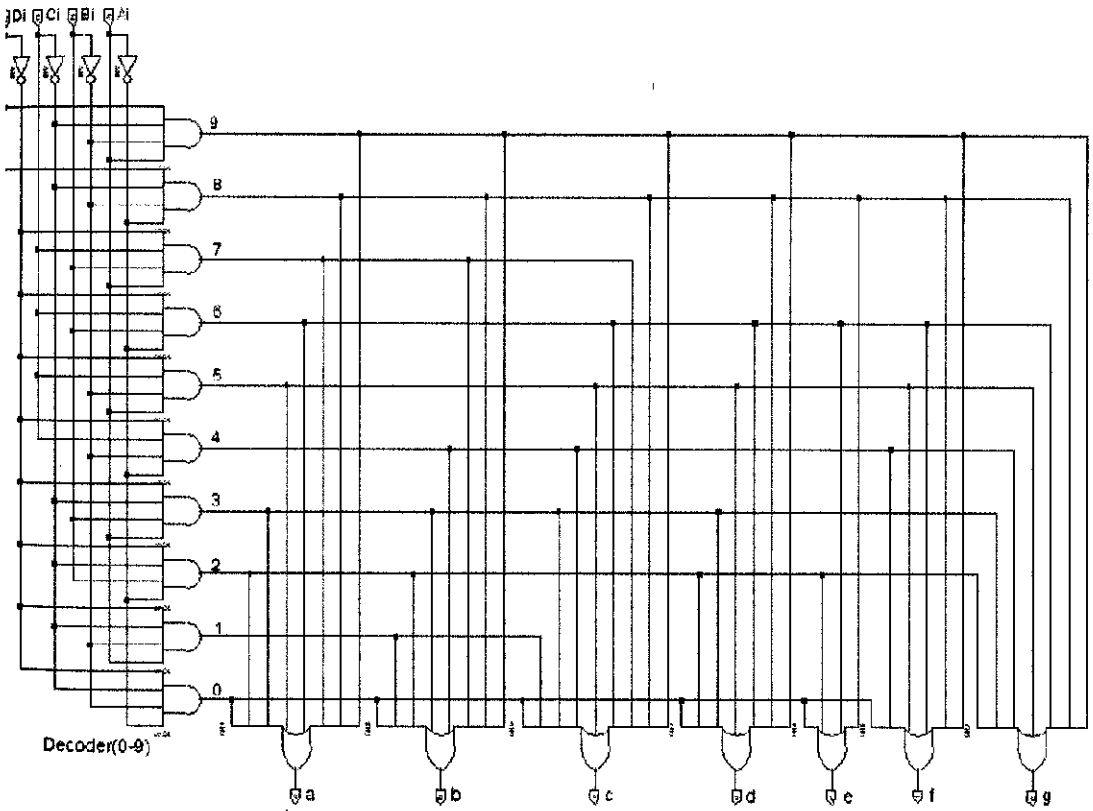
เพื่อให้ตัวเลขทุกตัวที่แสดงผลมีรูปแบบเดียวกัน ดังนั้นวงจรถอดรหัสตัวแสดงผลเซเวนเซกเมนต์ที่ใช้ในหนังสือเล่ นี้จะใช้รูปแบบเลขฐานสิบหกดังรายละเอียดรูปที่ 3.12b) ทั้งหมดแม้ว่าบางวงจรจะใช้ถอดรหัสเลข 0-9 ก็ตาม การออกแบบวงจ ถอดรหัส BCD เป็นเซเวนเซกเมนต์ (BCD to 7-Segment Decoder) นั้นสามารถทำได้โดยนำตารางความจริงรูปที่ 3.14 เฉพา เลข 0-9 ไปเขียนสมการบูลีนโดยนำอาเอาต์พุตเฉพาะช่องที่เป็นลอจิก '1' มา OR กันตามขั้นตอนที่เคยเรียนมาแล้ว ซึ่งหา สังเกตให้ดีจะพบว่าในขั้นตอนการแอนด์จะมีเทอมที่ซ้ำๆกันมาก ดังนั้นเพื่อประหยัดจำนวนเกตจึงควรสร้างวงจรส่วนนี้ทั้งหม เป็นวงจรถอดรหัส BCD เป็นเลขฐาน 10 (BCD to decimal Decoder) คือ เลข 0-9 ก่อน จากนั้นจึงนำอาเอาต์พุตที่ได้ไป OR กันเพื่ สร้างวงจรแต่ละเซกเมนต์ วงจรถอดรหัสแบบแคโถดร่วมที่ออกแบบสมบูรณ์แล้วแสดงดังในรูปที่ 3.15

No.	Input				Output						
	Di	Ci	Bi	Ai	g	f	e	d	c	b	a
0	0	0	0	0	0	1	1	1	1	1	1
1	0	0	0	1	0	0	0	0	1	1	0
2	0	0	1	0	1	0	1	1	0	1	1
3	0	0	1	1	1	0	0	1	1	1	1
4	0	1	0	0	1	1	0	0	1	1	0
5	0	1	0	1	1	1	0	1	1	0	1
6	0	1	1	0	1	1	1	1	1	0	0
7	0	1	1	1	0	0	0	0	1	1	1
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	0	0	1	1	1

รูปที่ 3.13 ตารางความจริงของตัวถอดรหัสเป็น Decimal digit display format แบบคาโถดร่วม

No.	Input				Output						
	Di	Ci	Bi	Ai	g	f	e	d	c	b	a
0	0	0	0	0	0	1	1	1	1	1	1
1	0	0	0	1	0	0	0	0	1	1	0
2	0	0	1	0	1	0	1	1	0	1	1
3	0	0	1	1	1	0	0	1	1	1	1
4	0	1	0	0	1	1	0	0	1	1	0
5	0	1	0	1	1	1	0	1	1	0	1
6	0	1	1	0	1	1	1	1	1	0	1
7	0	1	1	1	0	0	0	0	1	1	1
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	0	1	1	1	1
10 = A	0	0	1	0	1	1	1	0	1	1	1
11 = b	0	0	1	1	1	1	1	1	1	0	0
12 = C	0	1	0	0	0	1	1	1	0	0	1
13 = d	0	1	0	1	1	0	1	1	1	1	0
14 = E	0	1	1	0	1	1	1	1	0	0	1
15 = F	0	1	1	1	1	0	1	0	0	0	1

รูปที่ 3.14 ตารางความจริงของตัวถอดรหัสเป็น Hexadecimal digit display format แบบแคโถดร่วม



รูปที่ 3.15 วงจรถอดรหัส BCD เป็นเซกเมนต์แบบแคโถดร่วม

การทดลองที่ 3.3.1 วงจร 2 to 4 Decoder

วัตถุประสงค์

1.เพื่อทำความเข้าใจเกี่ยวกับหลักการทำงานของวงจรถอดรหัสแบบ 2 to 4 Decoder

2.เพื่อทดลองใช้ ISE WebPACK 8.1i สร้างวงจรเปรียบเทียบและโปรแกรมลงชิพ CPLD หรือ FPGA

อุปกรณ์ทดลอง

บอร์ด CPLD Explorer XC9572XL หรือ CPLD Explorer XC9572 และ FPGA Discovery-III XC3S200F หรือ FPGA Discovery-III XC3S200F4

1. สร้างวงจร 2 to 4 Decoder ขนาด 1 บิตด้วย CPLD

ให้ออกแบบวงจรถอดรหัส 2 to 4 Decoder ด้วย FPGA ดังรูปที่ 3.11a) ตามขั้นตอนที่อธิบายในบทที่ 2 โดยเขียนโค้ดดังรูปที่ L1.1 การทดลองนี้จะใช้ Project Location ชื่อ ch3v แล้วกำหนด Project Name และ Source File ชื่อ ex3_3_1vexl

```
2   library IEEE;
3   use IEEE.STD_LOGIC_1164.ALL;
4
5   entity ex3_3_1vexl is
6       Port ( A : in  STD_LOGIC_VECTOR (1 downto 0);
7              E : in  STD_LOGIC;
8              D : out  STD_LOGIC_VECTOR (3 downto 0));
9   end ex3_3_1vexl;
10
11  architecture Behavioral of ex3_3_1vexl is
12     begin
13  process(A,E)
14     begin
15        if E='0' then
16              D <= "0000";
17        else
18              if    A="00" then D <= "0001";
19              elsif A="01" then D <= "0010";
20              elsif A="10" then D <= "0100";
21              else  D <= "1000";
22              end if;
23        end if;
24  end process;
25  end Behavioral;
```

รูปที่ L1.1 โค้ดระดับอัลกอริธึม

จากนั้นกำหนดขาสัญญาณต่างๆจะใช้ปุ่มกด PB1-PB3 เป็นอินพุตและ LED1-LED4 เป็นเอาต์พุต กล่าวคือ

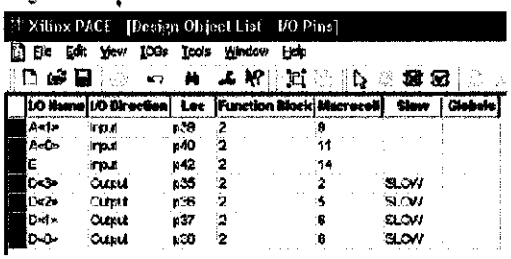
A1 = PB1 = INPUT = p39 D0 = LED1 = OUTPUT = p38

A0 = PB2 = INPUT = p40 D1 = LED2 = OUTPUT = p37

E = PB3 = INPUT = p42 D2 = LED3 = OUTPUT = p36

D3 = LED4 = OUTPUT = p35

โดยพินที่ใน Assign Package Pins สรุปดังนี้



I/O Name	I/O Direction	Loc	Function Block	Macrocell	Slew	Globals
A<1>	Input	p39	2	8		
A<0>	Input	p40	2	11		
E	Input	p42	2	14		
D<3>	Output	p35	2	2	SLOW	
D<2>	Output	p36	2	5	SLOW	
D<1>	Output	p37	2	8	SLOW	
D<0>	Output	p38	2	6	SLOW	

หลังจากโปรแกรมวงจรที่ออกแบบลงชิพ CPLD แล้วให้ทดลองกดปุ่ม PB1–PB3 และให้สังเกตดูผลที่ LED1–LED4 ว่าให้ลอจิกเอาต์พุตติดสว่างเป็นไปตามทฤษฎีหรือไม่ จากนั้นจึงบันทึกผลการทดลอง

2.สร้างวงจร 2 to 4 Decoder ขนาด 1 บิต ด้วย FPGA

ให้ออกแบบวงจรถอดรหัส เช่น 2 to 4 Decoder ด้วย FPGA ดังรูปที่ 3.11a) ตามขั้นตอนที่อธิบายในบทที่ 2 โดยเขียนโค้ดดังรูปที่ L1.1 การทดลองนี้จะใช้ Project Location ชื่อ ch3v แล้วกำหนด Project Name และ Source File ชื่อ ex3_3_1f

```
 2   library IEEE;
 3   use IEEE.STD_LOGIC_1164.ALL;
 4
 5   entity ex3_3_1vf is
 6       Port ( A : in  STD_LOGIC_VECTOR (1 downto 0);
 7              E : in  STD_LOGIC;
 8              D : out  STD_LOGIC_VECTOR (3 downto 0));
 9   end ex3_3_1vf;
10
11   architecture Behavioral of ex3_3_1vf is
12      begin
13   process(A,E)
14      begin
15         if E='0' then
16               D <= "0000";
17         else
18               if    A="00" then D <= "0001";
19               elsif A="01" then D <= "0010";
20               elsif A="10" then D <= "0100";
21               else  D <= "1000";
22               end if;
23         end if;
24   end process;
25   end Behavioral;
```

รูปที่ L2.1 โค้ดวาดับอัลกอริธึม

จากนั้นกำหนดขาสัญญาณต่างๆจะใช้ปุ่มกด PB1-PB3 เป็นอินพุตและ LED L0-L3 เป็นเอาต์พุต กล่าวคือ

A1 = PB1 = INPUT = p44 D0 = L3 = OUTPUT = p76

A0 = PB2 = INPUT = p46 D1 = L2 = OUTPUT = p69

E = PB3 = INPUT = p47 D2 = L1 = OUTPUT = p77

D3 = L0 = OUTPUT = p70

โดยพิมพ์ใน Assign Package Pins สรุปดังนี้

```
NET "A0"  LOC = "p46" | IOSTANDARD = LVCMOS33 ;
NET "A1"  LOC = "p44" | IOSTANDARD = LVCMOS33 ;
NET "D0"  LOC = "p76" | IOSTANDARD = LVCMOS33  | SLEW = SLOW ;
NET "D1"  LOC = "p69" | IOSTANDARD = LVCMOS33  | SLEW = SLOW ;
NET "D2"  LOC = "p77" | IOSTANDARD = LVCMOS33  | SLEW = SLOW ;
NET "D3"  LOC = "p70" | IOSTANDARD = LVCMOS33  | SLEW = SLOW ;
NET "E"  LOC = "p47" | IOSTANDARD = LVCMOS33 ;
```

หลังจากโปรแกรมวงจรที่ออกแบบลง FPGA แล้วให้ทดลองกด ปุ่ม PB1–PB3 และให้สังเกตดูผลที่ LED L0–L3 ว่าให้ลอจิกเอาต์พุตติดสว่างเป็นไปตามทฤษฎีหรือไม่ จากนั้นจึงบันทึกผลการทดลอง

การทดลองที่ 3.3.2 วงจรถอดรหัส BCD เป็นเซเวนเซกเมนต์

วัตถุประสงค์

1.เพื่อทำความเข้าใจเกี่ยวกับวงจรถอดรหัส BCD เป็นเซเวนเซกเมนต์

2.เพื่อทดลองใช้ ISE WebPACK 8.1i สร้างวงจรบวกและ โปรแกรมลงชิพ CPLD และ FPGA

อุปกรณ์ทดลอง

บอร์ด CPLD Explorer XC9572XL หรือ CPLD Explorer XC9572 และ FPGA Discovery-III XC3S200F หรือ FPGA Discovery-III XC3S200F4

1. สร้างวงจรถอดรหัส BCD เป็นเซเวนเซกเมนต์ด้วย CPLD

นำตารางความจริงของวงจรถอดรหัส BCD เป็นเซเวนเซกเมนต์แบบแคโธดร่วมแบบ Hexadecimal digit display format ในรูปที่ 3.14 มาเขียนเป็นโค้ด VHDL ดังรูปที่ L1.1 โดยที่ Y(6)-Y(0) = a-g และ A(3)-A(0) = Di-Ai ซึ่งโค้ดนี้ได้รวมส่วนที่เป็นการต่อขาแคโธดร่วมและ จุด dp ไว้แล้ว โดยจะ ใช้ Project Location ชื่อ ch3v แล้วกำหนด Project Name และ Source File ชื่อ ex3_3_2vcxl

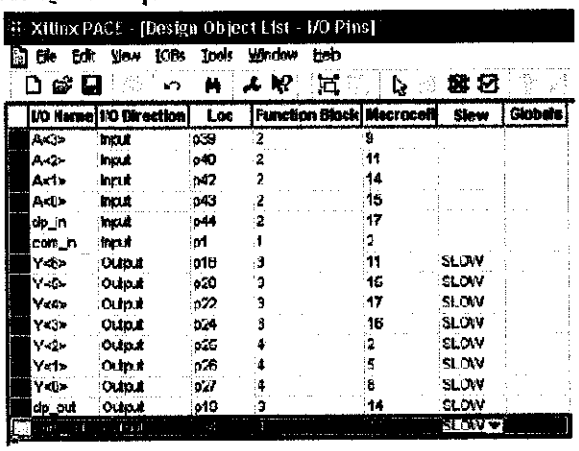
```
 2  library IEEE;
 3  use IEEE.STD_LOGIC_1164.ALL;
 4
 5  entity ex3_3_2vcxl is
 6      Port ( A : in  STD_LOGIC_VECTOR (3 downto 0);
 7             dp_in : in  STD_LOGIC;
 8             com_in : in  STD_LOGIC;
 9             Y : out  STD_LOGIC_VECTOR (6 downto 0);
10             dp_out : out  STD_LOGIC;
11             com_out : out  STD_LOGIC);
12  end ex3_3_2vcxl;
13
14  architecture Behavioral of ex3_3_2vcxl is
15  begin
16  ---------------------- 7 segment decoder ----------------
17        process(A)
18           begin
19              if    A="0000" then Y <= "0111111";
20              elsif A="0001" then Y <= "0000110";
21              elsif A="0010" then Y <= "1011011";
22              elsif A="0011" then Y <= "1001111";
23              elsif A="0100" then Y <= "1100110";
24              elsif A="0101" then Y <= "1101101";
25              elsif A="0110" then Y <= "1111101";
26              elsif A="0111" then Y <= "0000111";
27              elsif A="1000" then Y <= "1111111";
28              elsif A="1001" then Y <= "1101111";
29              elsif A="1010" then Y <= "1110111";
30              elsif A="1011" then Y <= "1111100";
31              elsif A="1100" then Y <= "0111001";
32              elsif A="1101" then Y <= "1011110";
33              elsif A="1110" then Y <= "1111001";
34              else Y <= "1110001";
35              end if;
36        end process;
37  ---------------- Common and Decimal point (dp)----------
38        dp_out  <= dp_in;
39        com_out <= not com_in;
40  end Behavioral;
```

รูปที่ L1.2 วงจรทดสอบวงจรถอดรหัส BCD เป็นเซเวนเซกเมนต์แบบแคโธดร่วมที่เขียนโค้ดระดับอัลกอริธึม

การกำหนดขาสัญญาณต่างๆจะใช้ปุ่มกด PB1 ถึง PB6 (SLIDE SW4) เป็นอินพุต มีตัวแสดงผลเซเวนเซกเมนต์แบบแคโถดร่วมหลักที่ 1 คือ DIGIT1 เป็นเอาต์พุต กล่าวคือ

A3	= PB1 = INPUT = p39	a = a	= OUTPUT = p27
A2	= PB2 = INPUT = p40	b = b	= OUTPUT = p26
A1	= PB3(SLIDE SW1) = INPUT = p42	c = c	= OUTPUT = p25
A0	= PB4(SLIDE SW2) = INPUT = p43	d = d	= OUTPUT = p24
dp_in	= PB5(SLIDE SW3) = INPUT = p44	e = e	= OUTPUT = p22
com_in	= PB6(SLIDE SW4) = INPUT = p1	f = f	= OUTPUT = p20
		g = g	= OUTPUT = p18
		dp_out = dp	= OUTPUT = p19
		com_out = DIGIT1	= OUTPUT = p34

โดยพิมพ์ใน Assign Package Pins สรุปดังนี้


Xilinx PACE - [Design Object List - I/O Pins]

File Edit View IOBs Tools Window Help

I/O Name	I/O Direction	Loc	Function Block	Macrocell	Slew	Globals
A<3>	Input	p39	2	9		
A<2>	Input	p40	2	11		
A<1>	Input	p42	2	14		
A<0>	Input	p43	2	15		
dp_in	Input	p44	2	17		
com_in	Input	p1	1	2		
Y<6>	Output	p18	3	11	SLOW	
Y<5>	Output	p20	3	16	SLOW	
Y<4>	Output	p22	3	17	SLOW	
Y<3>	Output	p24	3	18	SLOW	
Y<2>	Output	p25	4	2	SLOW	
Y<1>	Output	p26	4	5	SLOW	
Y<0>	Output	p27	4	8	SLOW	
dp_out	Output	p19	3	14	SLOW	

หลังจากโปรแกรมวงจรที่ออกแบบลงชิพ CPLD แล้วให้ทดลองกดปุ่ม PB1-PB6 และให้สังเกตดูผลที่ตัวแสดงผลเซเวนเซกเมนต์ว่ามีส่วนติดสว่างโดยให้ลอจิกเอาต์พุตเป็นไปตามหลุมฝุ่นหรือไม่ จากนั้นจึงบันทึกผลการทดลอง

2. สร้างวงจรถอดรหัส BCD เป็นเซเวนเซกเมนต์ด้วย FPGA

นำตารางความจริงของวงจรถอดรหัส BCD เป็นเซเวนเซกเมนต์แบบแคโถดร่วมแบบ Hexadecimal digit display format ในรูปที่ 3.14 มาเขียนเป็นโค้ด VHDL ดังรูปที่ L2.1 โดยที่ Y(6)-Y(0) = a-g และ A(3)-A(0) = Di-Ai ซึ่งโค้ดนี้ได้รวมส่วนที่เป็นการต่อขาแคโถดร่วมและจุด dp ไว้แล้ว โดยจะใช้ Project Location ชื่อ ch3v แล้วกำหนด Project Name และ Source File ชื่อ ex3_3_2vexl

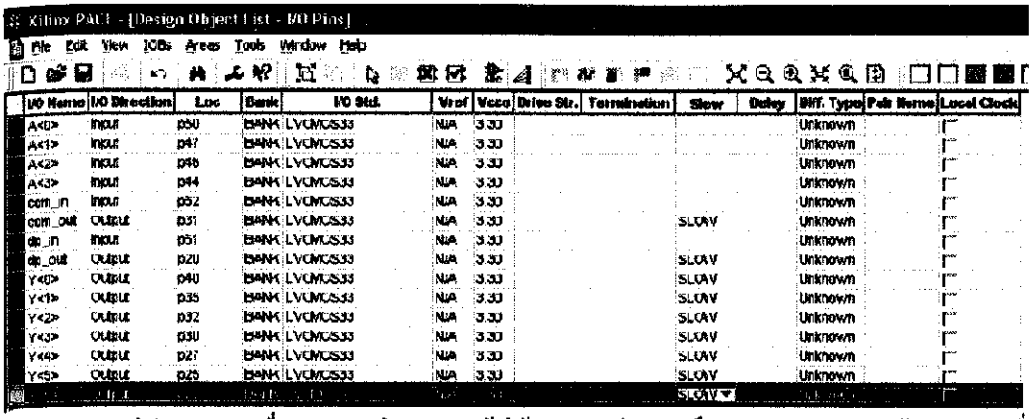
```
 2   library IEEE;
 3   use IEEE.STD_LOGIC_1164.ALL;
 4
 5   entity ex3_3_2vf is
 6       Port ( A : in  STD_LOGIC_VECTOR (3 downto 0);
 7              dp_in : in  STD_LOGIC;
 8              com_in : in  STD_LOGIC;
 9              Y : out  STD_LOGIC_VECTOR (6 downto 0);
10              dp_out : out  STD_LOGIC;
11              com_out : out  STD_LOGIC);
12   end ex3_3_2vf;
13
14   architecture Behavioral of ex3_3_2vf is
15   begin
16   ------------------ 7 segment decoder ---------------
17         process(A)
18            begin
19               if    A="0000" then Y <= "0111111";
20               elsif A="0001" then Y <= "0000110";
21               elsif A="0010" then Y <= "1011011";
22               elsif A="0011" then Y <= "1001111";
23               elsif A="0100" then Y <= "1100110";
24               elsif A="0101" then Y <= "1101101";
25               elsif A="0110" then Y <= "1111101";
26               elsif A="0111" then Y <= "0000111";
27               elsif A="1000" then Y <= "1111111";
28               elsif A="1001" then Y <= "1101111";
29               elsif A="1010" then Y <= "1110111";
30               elsif A="1011" then Y <= "1111100";
31               elsif A="1100" then Y <= "0111001";
32               elsif A="1101" then Y <= "1011110";
33               elsif A="1110" then Y <= "1111001";
34               else Y <= "1110001";
35               end if;
36         end process;
37   ------------- Common and Decimal point (dp)---------
38         dp_out  <= dp_in;
39         com_out <= not com_in;
40   end Behavioral;
```

รูปที่ L2.2 วงจรทดสอบวงจรถอดรหัส BCD เป็นเซเวนเซกเมนต์แบบแคโถดร่วมที่เขียนโค้ดระดับอัลกอริธึม

การกำหนดขาสัญญาณต่างๆจะใช้ปุ่มกด PB1 ถึง PB5 และ Dip SW1 เป็นอินพุต มีตัวแสดงผลเซเวนเซกเมนต์แบบแคโถดร่วมหลักที่ 1 คือ DIGIT1 เป็นเอาต์พุต กล่าวคือ

A3	= PB1	= INPUT = p44		a	= a	= OUTPUT = p40
A2	= PB2	= INPUT = p46		b	= b	= OUTPUT = p35
A1	= PB3	= INPUT = p47		c	= c	= OUTPUT = p32
A0	= PB4	= INPUT = p50		d	= d	= OUTPUT = p30
dp_in	= PB5	= INPUT = p51		e	= e	= OUTPUT = p27
com_in	= Dip SW1	= INPUT = p52		f	= f	= OUTPUT = p25
				g	= g	= OUTPUT = p23
				dp_out	= dp	= OUTPUT = p20
				com_out	= DIGIT1	= OUTPUT = p31

โดยพิมพ์ใน Assign Package Pins สรุปดังนี้



I/O Name	I/O Direction	Loc	Bank	I/O Std.	Vref	Vcco	Drive Str.	Termination	Slew	Delay	Diff. Type	Pair Name	Local Clock
A<0>	INPUT	p50	BANK	LVCMOS33	N/A	3.30					Unknown		
A<1>	INPUT	p47	BANK	LVCMOS33	N/A	3.30					Unknown		
A<2>	INPUT	p46	BANK	LVCMOS33	N/A	3.30					Unknown		
A<3>	INPUT	p44	BANK	LVCMOS33	N/A	3.30					Unknown		
com_in	INPUT	p52	BANK	LVCMOS33	N/A	3.30					Unknown		
com_out	OUTPUT	p31	BANK	LVCMOS33	N/A	3.30			SLOW		Unknown		
dp_in	INPUT	p51	BANK	LVCMOS33	N/A	3.30					Unknown		
dp_out	OUTPUT	p20	BANK	LVCMOS33	N/A	3.30			SLOW		Unknown		
Y<0>	OUTPUT	p40	BANK	LVCMOS33	N/A	3.30			SLOW		Unknown		
Y<1>	OUTPUT	p35	BANK	LVCMOS33	N/A	3.30			SLOW		Unknown		
Y<2>	OUTPUT	p32	BANK	LVCMOS33	N/A	3.30			SLOW		Unknown		
Y<3>	OUTPUT	p30	BANK	LVCMOS33	N/A	3.30			SLOW		Unknown		
Y<4>	OUTPUT	p27	BANK	LVCMOS33	N/A	3.30			SLOW		Unknown		
Y<5>	OUTPUT	p26	BANK	LVCMOS33	N/A	3.30			SLOW		Unknown		
									SLOW				

หลังจากโปรแกรมวงจรที่ออกแบบลงชิพ FPGA แล้วให้ทดลองกดปุ่ม PB1 ถึง PB5 และ Dip SW1 แล้วสังเกตดูผลที่ตัวแสดงผลเซเวนเซกเมนต์ว่ามีส่วนติดสว่างโดยให้ลอจิกเอาต์พุตเป็นไปตามทฤษฎีหรือไม่ จากนั้นบันทึกผลการทดลอง

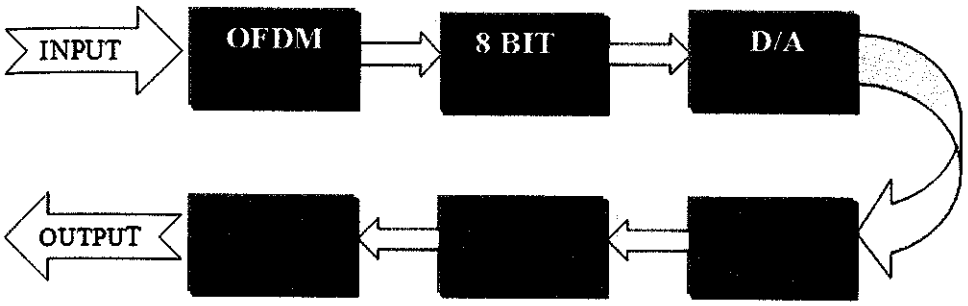
<u>แบบฝึกหัด</u>

1.จงสร้างวงจรเครื่องคิดเลขที่สามารถบวกเลขจำนวนเต็มบวกขนาด 2 บิตโดยแสดงผลทางตัวแสดงผลเซเวนเซกเมนต์

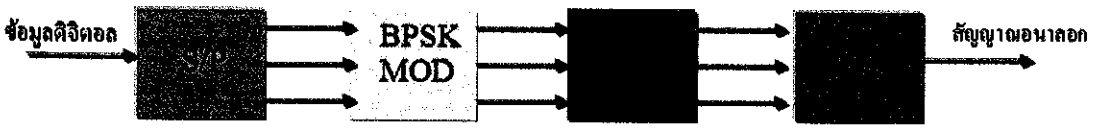
ภาคผนวก ง

โปรแกรมภาคส่งและภาครับ

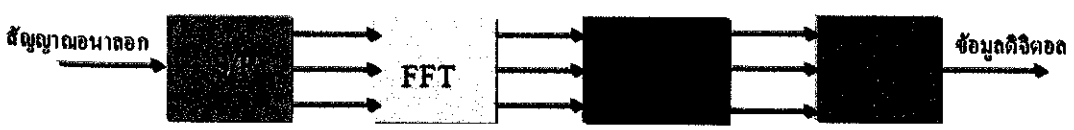
ภาคส่งและภาครับ



ภาคส่ง



ภาครับ



โปรแกรมภาคส่ง

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_ARITH.ALL;
4  use IEEE.STD_LOGIC_UNSIGNED.ALL;
5  use IEEE.NUMERIC_STD.ALL;
6
7  entity ofdm_tx is
8      Port ( DATA : in  STD_LOGIC_VECTOR (7 downto 0);
9             Sysclk, rst_b : in std_logic;
10
11            TxOut1 : out std_logic ;
12            TxOut2 : out std_logic ;
13            TxOut3 : out std_logic ;
14            TxOut4 : out std_logic ;
15            TxOut5 : out std_logic ;
16            TxOut6 : out std_logic ;
17            TxOut7 : out std_logic ;
18            TxOut8 : out std_logic);
19
20 end ofdm_tx;
21
22 architecture Behavioral of ofdm_tx is
23 signal X0,X1,X2,X3,X4,X5,X6,X7 : integer range -1 to 1 ;
24 signal XX0r,XX1r,XX2r,XX3r,XX4r,XX5r,XX6r,XX7r,XX8r,XX9r,XX10r,XX11r,XX12r,XX13r,XX14r,XX15r : integer;
25 signal XX0i,XX1i,XX2i,XX3i,XX4i,XX5i,XX6i,XX7i,XX8i,XX9i,XX10i,XX11i,XX12i,XX13i,XX14i,XX15i : integer;
26
27     signal rst_d,rst_trig,sclk : std_logic;
28     signal set : std_logic := '0';
29     signal Cnr : integer range 0 to 2609 := 2609;
30     signal N : integer range 0 to 65 := 65;
31     signal start : bit;
32     signal bs : integer range 0 to 171700 := 171700;
33 -- baud rate 9600 : set bs at 85800 and Cnr at 2605
34 signal x1r : STD_LOGIC_VECTOR (31 downto 0);
35 signal x2r : STD_LOGIC_VECTOR (31 downto 0);
36 signal x3r : STD_LOGIC_VECTOR (31 downto 0);
37 signal x4r : STD_LOGIC_VECTOR (31 downto 0);
38 signal x5r : STD_LOGIC_VECTOR (31 downto 0);
39 signal x6r,x7r,x8r : STD_LOGIC_VECTOR (31 downto 0);
40 signal x1i,x2i,x3i,x4i,x5i,x6i,x7i,x8i : STD_LOGIC_VECTOR (31 downto 0);
41
42
43 begin
44 process(sysclk,data)
45
46 begin
47
48 if sysclk'event and sysclk='1' then
49 if DATA(7)='0' then X0 <=  1; end if;
50 if DATA(7)='1' then X0 <= -1; end if;
51 if DATA(6)='0' then X1 <=  1; end if;
52 if DATA(6)='1' then X0 <= -1; end if;
53 if DATA(5)='0' then X2 <=  1; end if;
54 if DATA(5)='0' then X0 <= -1; end if;
55 if DATA(4)='0' then X3 <=  1; end if;
56 if DATA(4)='0' then X0 <= -1; end if;
57 if DATA(3)='0' then X4 <=  1; end if;
58 if DATA(3)='0' then X0 <= -1; end if;
59 if DATA(2)='0' then X5 <=  1; end if;
60 if DATA(2)='0' then X0 <= -1; end if;
61 if DATA(1)='0' then X6 <=  1; end if;

62 if DATA(1)='0' then X0 <= -1; end if;
63 if DATA(0)='0' then X7 <=  1; end if;
64 if DATA(7)='0' then X0 <= -1; end if;
65
66
67
68
69 XX0r <= ((X0)*625)+((X1)*625)+((X2)*625)+((X3)*625)+((X4)*625)+((X5)*625)+((X6)*625)
70         +((X7)*625)+((X0)*625)+((X1)*625)+((X2)*625)+((X3)*625)+((X4)*625)+((X5)*625)+((X6)*625)+((X7)*625);
71 XX1r <= (X0*625)+(X1*577)+(X2*441)+(X3*239)+(X5*(-239))+(X6*(-441))+(X7*(-577))
72         +(X0*(-625))+(X1*(-577))+(X2*(-441))+(X3*(-239))+(X5*239)+(X6*441)+(X7*577);
73 XX2r <= (X0*625)+(X1*441)+(X3*(-441))+(X4*(-625))+(X5*(-441))+(X7*441)
74         +(X0*625)+(X1*441)+(X3*(-441))+(X4*(-625))+(X5*(-441))+(X7*441);
75 XX3r <= (X0*625)+(X1*239)+(X2*(-441))+(X3*(-577))+(X5*577)+(X6*441)+(X7*(-239))
76         +(X0*(-625))+(X1*(-239))+(X2*441)+(X3*577)+(X5*(-577))+(X6*(-441))+(X7*239);
77 XX4r <= (X0*625)+(X2*(-625))+(X4*625)+(X6*(-625))+(X0*625)+(X2*(-625))+(X4*625)+(X6*(-625));
78 XX5r <=(X0*625)+(X1*(-239))+(X2*(-441))+(X3*577)+(X5*(-577))+(X6*441)+(X7*239)
79         +(X0*(-625))+(X1*239)+(X2*441)+(X3*(-577))+(X5*577)+(X6*(-441))+(X7*(-239));
80 XX6r <=(X0*625)+(X1*(-441))+(X3*441)+(X4*(-625))+(X5*441)+(X7*(-441))
81         +(X0*625)+(X1*(-441))+(X3*441)+(X4*(-625))+(X5*441)+(X7*(-441));
82 XX7r <=(X0*625)+(X1*(-577))+(X2*441)+(X3*(-239))+(X5*239)+(X6*(-441))+(X7*577)
83         +(X0*(-625))+(X1*577)+(X2*(-441))+(X3*239)+(X5*(-239))+(X6*441)+(X7*(-577));
84 XX8r <=(X0*625)+(X1*(-625))+(X2*625)+(X3*(-625))+(X4*625)+(X5*(-625))+(X6*625)+(X7*(-625))
85         +(X0*625)+(X1*(-625))+(X2*625)+(X3*(-625))+(X4*625)+(X5*(-625))+(X6*625)+(X7*(-625));
86 XX9r <=(X0*625)+(X1*(-577))+(X2*441)+(X3*(-239))+(X5*239)+(X6*(-441))+(X7*577)
87         +(X0*(-625))+(X1*577)+(X2*(-441))+(X3*239)+(X5*(-239))+(X6*441)+(X7*(-577));
88 XX10r<=(X0*625)+(X1*(-441))+(X3*441)+(X4*(-625))+(X5*441)+(X7*(-441))
89         +(X0*625)+(X1*(-441))+(X3*441)+(X4*(-625))+(X5*441)+(X7*(-441));
90 XX11r<=(X0*625)+(X1*(-239))+(X2*(-441))+(X3*577)+(X5*(-577))+(X6*441)+(X7*239)
91         +(X0*(-625))+(X1*239)+(X2*441)+(X3*(-577))+(X5*577)+(X6*(-441))+(X7*(-239));
92 XX12r<=(X0*625)+(X2*(-625))+(X4*625)+(X6*(-625))+(X0*625)+(X2*(-625))+(X4*625)+(X6*(-625));
93 XX13r<=(X0*625)+(X1*239)+(X2*(-441))+(X3*(-577))+(X5*577)+(X6*441)+(X7*(-239))
94         +(X0*(-625))+(X1*(-239))+(X2*441)+(X3*577)+(X5*(-577))+(X6*(-441))+(X7*239);
95 XX14r<=(X0*625)+(X1*441)+(X3*(-441))+(X4*(-625))+(X5*(-441))+(X7*441)
96         +(X0*625)+(X1*441)+(X3*(-441))+(X4*(-625))+(X5*(-441))+(X7*441);
97 XX15r<=(X0*625)+(X1*577)+(X2*441)+(X3*239)+(X5*(-239))+(X6*(-441))+(X7*(-577))
98         +(X0*(-625))+(X1*(-577))+(X2*(-441))+(X3*(-239))+(X5*239)+(X6*441)+(X7*577);
```

การ modulation แบบ BPSK

ส่วน IFFT

```
99
100   XX0i <= (X0*0)+(X1*0)+(X2*0)+(X3*0)+(X4*0)+(X5*0)+(X6*0)+(X7*0)
101           +(X0*0)+(X1*0)+(X2*0)+(X3*0)+(X4*0)+(X5*0)+(X6*0)+(X7*0);
102   XX1i <= (X1*239)+(X2*441)+(X3*577)+(X4*625)+(X5*577)+(X6*441)+(X7*239)
103           +(X1*(-239))+(X2*(-441))+(X3*(-577))+(X4*(-625))+(X5*(-577))+(X6*(-441))+(X7*(-239));
104   XX2i <= (X1*441)+(X2*625)+(X3*441)+(X5*(-441))+(X6*(-625))+(X7*(-441))
105           +(X1*441)+(X2*625)+(X3*441)+(X5*(-441))+(X6*(-625))+(X7*(-441));
106   XX3i <= (X1*577)+(X2*441)+(X3*(-239))+(X4*(-625))+(X5*(-239))+(X6*441)+(X7*577)
107           +(X1*(-577))+(X2*(-441))+(X3*239)+(X4*625)+(X5*239)+(X6*(-441))+(X7*(-577));
108   XX4i <= (X1*625)+(X3*(-625))+(X5*625)+(X7*(-625))+(X1*625)+(X3*(-625))+(X5*625)+(X7*(-625));
109   XX5i <= (X1*577)+(X2*441)+(X3*(-239))+(X4*(-625))+(X5*(-239))+(X6*441)+(X7*577)
110           +(X1*(-577))+(X2*(-441))+(X3*239)+(X4*625)+(X5*239)+(X6*(-441))+(X7*(-577));
111   XX6i <= (X1*441)+(X2*(-625))+(X3*441)+(X5*(-441))+(X6*625)+(X7*(-441))
112           +(X1*441)+(X2*(-625))+(X3*441)+(X5*(-441))+(X6*625)+(X7*(-441));
113   XX7i <= (X1*239)+(X2*(-441))+(X3*577)+(X4*(-625))+(X5*577)+(X6*(-441))+(X7*239)
114           +(X1*(-239))+(X2*441)+(X3*(-577))+(X4*625)+(X5*(-577))+(X6*441)+(X7*(-239));
115   XX8i <= (X0*0)+(X1*0)+(X2*0)+(X3*0)+(X4*0)+(X5*0)+(X6*0)+(X7*0)+(X0*0)
116           +(X1*0)+(X2*0)+(X3*0)+(X4*0)+(X5*0)+(X6*0)+(X7*0);
117   XX9i <= (X1*(-239))+(X2*441)+(X3*(-577))+(X4*625)+(X5*(-577))+(X6*441)+(X7*(-239))
118           +(X1*239)+(X2*(-441))+(X3*577)+(X4*(-625))+(X5*577)+(X6*(-441))+(X7*239);
119   XX10i<= (X1*(-441))+(X2*625)+(X3*(-441))+(X5*441)+(X6*(-625))+(X7*441)
120           +(X1*(-441))+(X2*625)+(X3*(-441))+(X5*441)+(X6*(-625))+(X7*441);
121   XX11i<= (X1*(-577))+(X2*441)+(X3*239)+(X4*(-625))+(X5*239)+(X6*441)+(X7*(-577))
122           +(X1*577)+(X2*(-441))+(X3*(-239))+(X4*625)+(X5*(-239))+(X6*(-441))+(X7*577);
123   XX12i<= (X1*(-625))+(X3*625)+(X5*(-625))+(X7*625)+(X1*(-625))+(X3*625)+(X5*(-625))+(X7*625);
124   XX13i<= (X1*(-577))+(X2*(-441))+(X3*239)+(X4*625)+(X5*239)+(X6*(-441))+(X7*(-577))
125           +(X1*577)+(X2*441)+(X3*(-239))+(X4*(-625))+(X5*(-239))+(X6*441)+(X7*577);
126   XX14i<= (X1*(-441))+(X2*(-625))+(X3*(-441))+(X5*441)+(X6*625)+(X7*441)
127           +(X1*(-441))+(X2*(-625))+(X3*(-441))+(X5*441)+(X6*625)+(X7*441);
128   XX15i<= (X1*(-239))+(X2*(-441))+(X3*(-577))+(X4*(-625))+(X5*(-577))+(X6*(-441))+(X7*(-239))
129           +(X1*239)+(X2*441)+(X3*577)+(X4*625)+(X5*577)+(X6*441)+(X7*239);
130
```

ส่วน IFFT

```
132   x1r <= conv_std_logic_vector(XX0r,32);
133   x2r <= conv_std_logic_vector(XX2r,32);
134   x3r <= conv_std_logic_vector(XX4r,32);
135   x4r <= conv_std_logic_vector(XX6r,32);
136   x5r <= conv_std_logic_vector(XX8r,32);
137   x6r <= conv_std_logic_vector(XX10r,32);
138   x7r <= conv_std_logic_vector(XX12r,32);
139   x8r <= conv_std_logic_vector(XX14r,32);
140
141   x1i <= conv_std_logic_vector(XX0i,32);
142   x2i <= conv_std_logic_vector(XX2i,32);
143   x3i <= conv_std_logic_vector(XX4i,32);
144   x4i <= conv_std_logic_vector(XX6i,32);
145   x5i <= conv_std_logic_vector(XX8i,32);
146   x6i <= conv_std_logic_vector(XX10i,32);
147   x7i <= conv_std_logic_vector(XX12i,32);
148   x8i <= conv_std_logic_vector(XX14i,32);
149
150
151   end if;
152   end process;
153
154
155
156   Checking: process(Sysclk)
157    begin
158    rst_d <= rst_b;
159    rst_trig <= rst_b and (not rst_d);
160    if (rising_edge(Sysclk)) then

161      if rst_trig = '1' then
162         bs <= 0;
163      else
164         if bs /= 171700 then bs <= bs+1;
165         end if;
166      end if;
167    end if;
168    end process;
169
170    Debounce: process(bs)
171    begin
172      case bs is
173      when 0 to 171699 =>
174         start <= '1';
175      when 171700 =>
176         start <= '0';
177      end case;
178    end process;
179
180    GenClock: process(Sysclk)
181    begin
182    if (rising_edge(Sysclk)) then
183      if start = '1' and Ctr = 2603 then
184       Ctr <= 0;
185       Bclk <= '1';
186      elsif start = '1' and Ctr /= 2603 then
187       Ctr <= Ctr+1;
188       Bclk <= '0';
189      elsif start = '0' then
190       Ctr <= 2603;
191      end if;
192    end if;
193    end process;
194
195    Sending: process(Bclk)
```

```
196   begin
197    if Bclk = '1' then
198        if N = 65 then N <= 0;
199        else N <= N+1;
200        end if;
201    end if;
202   end process;
203
204   Display: process(N)
205   begin
206        if (N = 0) then
207
208             TxOut1 <= '0';
209             TxOut2 <= '0';
210             TxOut3 <= '0';
211             TxOut4 <= '0';
212             TxOut5 <= '0';
213             TxOut6 <= '0';
214             TxOut7 <= '0';
215             TxOut8 <= '0';
216         end if;
217        if (N = 1) then
218
219             TxOut1 <= x1r(0);
220             TxOut2 <= x2r(0);
221             TxOut3 <= x3r(0);
222             TxOut4 <= x4r(0);
223             TxOut5 <= x5r(0);
224             TxOut6 <= x6r(0);
225             TxOut7 <= x7r(0);
226             TxOut8 <= x8r(0);
227         end if;
228         if (N = 2) then
229
230             TxOut1 <= x1r(1);
231             TxOut2 <= x2r(1);
232             TxOut3 <= x3r(1);
233             TxOut4 <= x4r(1);
234             TxOut5 <= x5r(1);
235             TxOut6 <= x6r(1);
236             TxOut7 <= x7r(1);
237             TxOut8 <= x8r(1);
238         end if;
239         if (N = 3) then
240
241             TxOut1 <= x1r(2);
242             TxOut2 <= x2r(2);
243             TxOut3 <= x3r(2);
244             TxOut4 <= x4r(2);
245             TxOut5 <= x5r(2);
246             TxOut6 <= x6r(2);
247             TxOut7 <= x7r(2);
248             TxOut8 <= x8r(2);
249         end if;
250         if (N = 4) then
251
252             TxOut1 <= x1r(3);
253             TxOut2 <= x2r(3);
254             TxOut3 <= x3r(3);
255             TxOut4 <= x4r(3);
256             TxOut5 <= x5r(3);
257             TxOut6 <= x6r(3);
258             TxOut7 <= x7r(3);
259             TxOut8 <= x8r(3);
260         end if;
261         if (N = 5) then
262
263             TxOut1 <= x1r(4);
264             TxOut2 <= x2r(4);
265             TxOut3 <= x3r(4);
266             TxOut4 <= x4r(4);
267             TxOut5 <= x5r(4);
268             TxOut6 <= x6r(4);
269             TxOut7 <= x7r(4);
270             TxOut8 <= x8r(4);
271         end if;
272         if (N = 6) then
273
274             TxOut1 <= x1r(5);
275             TxOut2 <= x2r(5);
276             TxOut3 <= x3r(5);
277             TxOut4 <= x4r(5);
278             TxOut5 <= x5r(5);
279             TxOut6 <= x6r(5);
280             TxOut7 <= x7r(5);
281             TxOut8 <= x8r(5);
282         end if;
283         if (N = 7) then
284
285             TxOut1 <= x1r(6);
286             TxOut2 <= x2r(6);
287             TxOut3 <= x3r(6);
288             TxOut4 <= x4r(6);
289             TxOut5 <= x5r(6);
290             TxOut6 <= x6r(6);
291             TxOut7 <= x7r(6);
292             TxOut8 <= x8r(6);
293        end if;
```

```
294         if (N = 8) then
295
296               TxOut1 <= x1r(7);
297               TxOut2 <= x2r(7);
298               TxOut3 <= x3r(7);
299               TxOut4 <= x4r(7);
300               TxOut5 <= x5r(7);
301               TxOut6 <= x6r(7);
302               TxOut7 <= x7r(7);
303               TxOut8 <= x8r(7);
304         end if;
305         if (N = 9) then
306
307               TxOut1 <= x1r(8);
308               TxOut2 <= x2r(8);
309               TxOut3 <= x3r(8);
310               TxOut4 <= x4r(8);
311               TxOut5 <= x5r(8);
312               TxOut6 <= x6r(8);
313               TxOut7 <= x7r(8);
314               TxOut8 <= x8r(8);
315         end if;
316         if (N = 10) then
317
318               TxOut1 <= x1r(9);
319               TxOut2 <= x2r(9);
320               TxOut3 <= x3r(9);
321               TxOut4 <= x4r(9);
322               TxOut5 <= x5r(9);
323               TxOut6 <= x6r(9);
324               TxOut7 <= x7r(9);
325               TxOut8 <= x8r(9);
326         end if;
327         if (N = 11) then
328
329               TxOut1 <= x1r(10);
330               TxOut2 <= x2r(10);
331               TxOut3 <= x3r(10);
332               TxOut4 <= x4r(10);
333               TxOut5 <= x5r(10);
334               TxOut6 <= x6r(10);
335               TxOut7 <= x7r(10);
336               TxOut8 <= x8r(10);
337         end if;
338         if (N = 12) then
339
340               TxOut1 <= x1r(11);
341               TxOut2 <= x2r(11);
342               TxOut3 <= x3r(11);
343               TxOut4 <= x4r(11);
344               TxOut5 <= x5r(11);
345               TxOut6 <= x6r(11);
346               TxOut7 <= x7r(11);
347               TxOut8 <= x8r(11);
348         end if;
349         if (N = 13) then
350
351               TxOut1 <= x1r(12);
352               TxOut2 <= x2r(12);
353               TxOut3 <= x3r(12);
354               TxOut4 <= x4r(12);
355               TxOut5 <= x5r(12);
356               TxOut6 <= x6r(12);
357               TxOut7 <= x7r(12);
358               TxOut8 <= x8r(12);
359         end if;
360         if (N = 14) then
361
362               TxOut1 <= x1r(13);
363               TxOut2 <= x2r(13);
364               TxOut3 <= x3r(13);
365               TxOut4 <= x4r(13);
366               TxOut5 <= x5r(13);
367               TxOut6 <= x6r(13);
368               TxOut7 <= x7r(13);
369               TxOut8 <= x8r(13);
370         end if;
371         if (N = 15) then
372
373               TxOut1 <= x1r(14);
374               TxOut2 <= x2r(14);
375               TxOut3 <= x3r(14);
376               TxOut4 <= x4r(14);
377               TxOut5 <= x5r(14);
378               TxOut6 <= x6r(14);
379               TxOut7 <= x7r(14);
380               TxOut8 <= x8r(14);
381         end if;
382         if (N = 16) then
383
384               TxOut1 <= x1r(15);
385               TxOut2 <= x2r(15);
386               TxOut3 <= x3r(15);
```

```
387             TxOut4 <= x4r(15);
388             TxOut5 <= x5r(15);
389             TxOut6 <= x6r(15);
390             TxOut7 <= x7r(15);
391             TxOut8 <= x8r(15);
392         end if;
393         if (N = 17) then
394
395             TxOut1 <= x1r(16);
396             TxOut2 <= x2r(16);
397             TxOut3 <= x3r(16);
398             TxOut4 <= x4r(16);
399             TxOut5 <= x5r(16);
400             TxOut6 <= x6r(16);
401             TxOut7 <= x7r(16);
402             TxOut8 <= x8r(16);
403         end if;
404         if (N = 18) then
405
406             TxOut1 <= x1r(17);
407             TxOut2 <= x2r(17);
408             TxOut3 <= x3r(17);
409             TxOut4 <= x4r(17);
410             TxOut5 <= x5r(17);
411             TxOut6 <= x6r(17);
412             TxOut7 <= x7r(17);
413             TxOut8 <= x8r(17);
414         end if;
415         if (N = 19) then
416
417             TxOut1 <= x1r(18);
418             TxOut2 <= x2r(18);
419             TxOut3 <= x3r(18);
420             TxOut4 <= x4r(18);
421             TxOut5 <= x5r(18);
422             TxOut6 <= x6r(18);
423             TxOut7 <= x7r(18);
424             TxOut8 <= x8r(18);
425         end if;
426         if (N = 20) then
427
428             TxOut1 <= x1r(19);
429             TxOut2 <= x2r(19);
430             TxOut3 <= x3r(19);
431             TxOut4 <= x4r(19);
432             TxOut5 <= x5r(19);
433             TxOut6 <= x6r(19);
434             TxOut7 <= x7r(19);
435             TxOut8 <= x8r(19);
436         end if;
437         if (N = 21) then
438
439             TxOut1 <= x1r(20);
440             TxOut2 <= x2r(20);
441             TxOut3 <= x3r(20);
442             TxOut4 <= x4r(20);
443             TxOut5 <= x5r(20);
444             TxOut6 <= x6r(20);
445             TxOut7 <= x7r(20);
446             TxOut8 <= x8r(20);
447         end if;
448         if (N = 22) then
449
450             TxOut1 <= x1r(21);
451             TxOut2 <= x2r(21);
452             TxOut3 <= x3r(21);
453             TxOut4 <= x4r(21);
454             TxOut5 <= x5r(21);
455             TxOut6 <= x6r(21);
456             TxOut7 <= x7r(21);
457             TxOut8 <= x8r(21);
458         end if;
459         if (N = 23) then
460
461             TxOut1 <= x1r(22);
462             TxOut2 <= x2r(22);
463             TxOut3 <= x3r(22);
464             TxOut4 <= x4r(22);
465             TxOut5 <= x5r(22);
466             TxOut6 <= x6r(22);
467             TxOut7 <= x7r(22);
468             TxOut8 <= x8r(22);
469         end if;
470         if (N = 24) then
471
472             TxOut1 <= x1r(23);
473             TxOut2 <= x2r(23);
474             TxOut3 <= x3r(23);
475             TxOut4 <= x4r(23);
476             TxOut5 <= x5r(23);
477             TxOut6 <= x6r(23);
478             TxOut7 <= x7r(23);
479             TxOut8 <= x8r(23);
480         end if;
481         if (N = 25) then
482
```

```
483             TxOut1 <= x1r(24);
484             TxOut2 <= x2r(24);
485             TxOut3 <= x3r(24);
486             TxOut4 <= x4r(24);
487             TxOut5 <= x5r(24);
488             TxOut6 <= x6r(24);
489             TxOut7 <= x7r(24);
490             TxOut8 <= x8r(24);
491         end if;
492         if (N = 26) then
493
494             TxOut1 <= x1r(25);
495             TxOut2 <= x2r(25);
496             TxOut3 <= x3r(25);
497             TxOut4 <= x4r(25);
498             TxOut5 <= x5r(25);
499             TxOut6 <= x6r(25);
500             TxOut7 <= x7r(25);
501             TxOut8 <= x8r(25);
502         end if;
503         if (N = 27) then
504
505             TxOut1 <= x1r(26);
506             TxOut2 <= x2r(26);
507             TxOut3 <= x3r(26);
508             TxOut4 <= x4r(26);
509             TxOut5 <= x5r(26);
510             TxOut6 <= x6r(26);
511             TxOut7 <= x7r(26);
512             TxOut8 <= x8r(26);
513         end if;
514         if (N = 28) then
515
516             TxOut1 <= x1r(27);
517             TxOut2 <= x2r(27);
518             TxOut3 <= x3r(27);
519             TxOut4 <= x4r(27);
520             TxOut5 <= x5r(27);
521             TxOut6 <= x6r(27);
522             TxOut7 <= x7r(27);
523             TxOut8 <= x8r(27);
524         end if;
525         if (N = 29) then
526
527             TxOut1 <= x1r(28);
528             TxOut2 <= x2r(28);
529             TxOut3 <= x3r(28);
530             TxOut4 <= x4r(28);
531             TxOut5 <= x5r(28);
532             TxOut6 <= x6r(28);
533             TxOut7 <= x7r(28);
534             TxOut8 <= x8r(28);
535         end if;
536         if (N = 30) then
537
538             TxOut1 <= x1r(29);
539             TxOut2 <= x2r(29);
540             TxOut3 <= x3r(29);
541             TxOut4 <= x4r(29);
542             TxOut5 <= x5r(29);
543             TxOut6 <= x6r(29);
544             TxOut7 <= x7r(29);
545             TxOut8 <= x8r(29);
546         end if;
547         if (N = 31) then
548
549             TxOut1 <= x1r(30);
550             TxOut2 <= x2r(30);
551             TxOut3 <= x3r(30);
552             TxOut4 <= x4r(30);
553             TxOut5 <= x5r(30);
554             TxOut6 <= x6r(30);
555             TxOut7 <= x7r(30);
556             TxOut8 <= x8r(30);
557         end if;
558         if (N = 32) then
559
560             TxOut1 <= x1r(31);
561             TxOut2 <= x2r(31);
562             TxOut3 <= x3r(31);
563             TxOut4 <= x4r(31);
564             TxOut5 <= x5r(31);
565             TxOut6 <= x6r(31);
566             TxOut7 <= x7r(31);
567             TxOut8 <= x8r(31);
568         end if;
569        if (N = 33) then
570
571             TxOut1 <= x1i(0);
572             TxOut2 <= x2i(0);
573             TxOut3 <= x3i(0);
574             TxOut4 <= x4i(0);
575             TxOut5 <= x5i(0);
576             TxOut6 <= x6i(0);
577             TxOut7 <= x7i(0);
578             TxOut8 <= x8i(0);
579         end if;
```

```
580         if (N = 34) then
581
582              TxOut1 <= x11(1);
583              TxOut2 <= x21(1);
584              TxOut3 <= x31(1);
585              TxOut4 <= x41(1);
586              TxOut5 <= x51(1);
587              TxOut6 <= x61(1);
588              TxOut7 <= x71(1);
589              TxOut8 <= x81(1);
590         end if;
591         if (N = 35) then
592
593              TxOut1 <= x11(2);
594              TxOut2 <= x21(2);
595              TxOut3 <= x31(2);
596              TxOut4 <= x41(2);
597              TxOut5 <= x51(2);
598              TxOut6 <= x61(2);
599              TxOut7 <= x71(2);
600              TxOut8 <= x81(2);
601         end if;
602         if (N = 36) then
603
604              TxOut1 <= x11(3);
605              TxOut2 <= x21(3);
606              TxOut3 <= x31(3);
607              TxOut4 <= x41(3);
608              TxOut5 <= x51(3);
609              TxOut6 <= x61(3);
610              TxOut7 <= x71(3);
611              TxOut8 <= x81(3);
612         end if;
613         if (N = 37) then
614
615              TxOut1 <= x11(4);
616              TxOut2 <= x21(4);
617              TxOut3 <= x31(4);
618              TxOut4 <= x41(4);
619              TxOut5 <= x51(4);
620              TxOut6 <= x61(4);
621              TxOut7 <= x71(4);
622              TxOut8 <= x81(4);
623         end if;
624         if (N = 38) then
625
626              TxOut1 <= x11(5);
627              TxOut2 <= x21(5);
628              TxOut3 <= x31(5);
629              TxOut4 <= x41(5);
630              TxOut5 <= x51(5);
631              TxOut6 <= x61(5);
632              TxOut7 <= x71(5);
633              TxOut8 <= x81(5);
634         end if;
635         if (N = 39) then
636
637              TxOut1 <= x11(6);
638              TxOut2 <= x21(6);
639              TxOut3 <= x31(6);
640              TxOut4 <= x41(6);
641              TxOut5 <= x51(6);
642              TxOut6 <= x61(6);
643              TxOut7 <= x71(6);
644              TxOut8 <= x81(6);
645        end if;
646        if (N = 40) then
647
648              TxOut1 <= x11(7);
649              TxOut2 <= x21(7);
650              TxOut3 <= x31(7);
651              TxOut4 <= x41(7);
652              TxOut5 <= x51(7);
653              TxOut6 <= x61(7);
654              TxOut7 <= x71(7);
655              TxOut8 <= x81(7);
656         end if;
657         if (N = 41) then
658
659              TxOut1 <= x11(8);
660              TxOut2 <= x21(8);
661              TxOut3 <= x31(8);
662              TxOut4 <= x41(8);
663              TxOut5 <= x51(8);
664              TxOut6 <= x61(8);
665              TxOut7 <= x71(8);
666              TxOut8 <= x81(8);
667         end if;
668         if (N = 42) then
669
670              TxOut1 <= x11(9);
671              TxOut2 <= x21(9);
672              TxOut3 <= x31(9);
673              TxOut4 <= x41(9);
674              TxOut5 <= x51(9);
675              TxOut6 <= x61(9);
676              TxOut7 <= x71(9);
677              TxOut8 <= x81(9);
678         end if;
```

```
679         if (N = 43) then
680
681              TxOut1 <= x11(10);
682              TxOut2 <= x21(10);
683              TxOut3 <= x31(10);
684              TxOut4 <= x41(10);
685              TxOut5 <= x51(10);
686              TxOut6 <= x61(10);
687              TxOut7 <= x71(10);
688              TxOut8 <= x81(10);
689         end if;
690         if (N = 44) then
691
692              TxOut1 <= x11(11);
693              TxOut2 <= x21(11);
694              TxOut3 <= x31(11);
695              TxOut4 <= x41(11);
696              TxOut5 <= x51(11);
697              TxOut6 <= x61(11);
698              TxOut7 <= x71(11);
699              TxOut8 <= x81(11);
700         end if;
701         if (N = 45) then
702
703              TxOut1 <= x11(12);
704              TxOut2 <= x21(12);
705              TxOut3 <= x31(12);
706              TxOut4 <= x41(12);
707              TxOut5 <= x51(12);
708              TxOut6 <= x61(12);
709              TxOut7 <= x71(12);
710              TxOut8 <= x81(12);
711         end if;
712         if (N = 46) then
713
714              TxOut1 <= x11(13);
715              TxOut2 <= x21(13);
716              TxOut3 <= x31(13);
717              TxOut4 <= x41(13);
718              TxOut5 <= x51(13);
719              TxOut6 <= x61(13);
720              TxOut7 <= x71(13);
721              TxOut8 <= x81(13);
722         end if;
723         if (N = 47) then
724
725              TxOut1 <= x11(14);
726              TxOut2 <= x21(14);
727              TxOut3 <= x31(14);
728              TxOut4 <= x41(14);
729              TxOut5 <= x51(14);
730              TxOut6 <= x61(14);
731              TxOut7 <= x71(14);
732              TxOut8 <= x81(14);
733         end if;
734         if (N = 48) then
735
736              TxOut1 <= x11(15);
737              TxOut2 <= x21(15);
738              TxOut3 <= x31(15);
739              TxOut4 <= x41(15);
740              TxOut5 <= x51(15);
741              TxOut6 <= x61(15);
742              TxOut7 <= x71(15);
743              TxOut8 <= x81(15);
744         end if;
745         if (N = 49) then
746
747              TxOut1 <= x11(16);
748              TxOut2 <= x21(16);
749              TxOut3 <= x31(16);
750              TxOut4 <= x41(16);
751              TxOut5 <= x51(16);
752              TxOut6 <= x61(16);
753              TxOut7 <= x71(16);
754              TxOut8 <= x81(16);
755         end if;
756         if (N = 50) then
757
758              TxOut1 <= x11(17);
759              TxOut2 <= x21(17);
760              TxOut3 <= x31(17);
761              TxOut4 <= x41(17);
762              TxOut5 <= x51(17);
763              TxOut6 <= x61(17);
764              TxOut7 <= x71(17);
765              TxOut8 <= x81(17);
766         end if;
767         if (N = 51) then
768
769              TxOut1 <= x11(18);
770              TxOut2 <= x21(18);
771              TxOut3 <= x31(18);
772              TxOut4 <= x41(18);
773              TxOut5 <= x51(18);
774              TxOut6 <= x61(18);
775              TxOut7 <= x71(18);
776              TxOut8 <= x81(18);
777         end if;
```

```
778         if (N = 52) then
779
780              TxOut1 <= x1i(19);
781              TxOut2 <= x2i(19);
782              TxOut3 <= x3i(19);
783              TxOut4 <= x4i(19);
784              TxOut5 <= x5i(19);
785              TxOut6 <= x6i(19);
786              TxOut7 <= x7i(19);
787              TxOut8 <= x8i(19);
788         end if;
789         if (N = 53) then
790
791              TxOut1 <= x1i(20);
792              TxOut2 <= x2i(20);
793              TxOut3 <= x3i(20);
794              TxOut4 <= x4i(20);
795              TxOut5 <= x5i(20);
796              TxOut6 <= x6i(20);
797              TxOut7 <= x7i(20);
798              TxOut8 <= x8i(20);
799         end if;
800         if (N = 54) then
801
802              TxOut1 <= x1i(21);
803              TxOut2 <= x2i(21);
804              TxOut3 <= x3i(21);
805              TxOut4 <= x4i(21);
806              TxOut5 <= x5i(21);
807              TxOut6 <= x6i(21);
808              TxOut7 <= x7i(21);
809              TxOut8 <= x8i(21);
810         end if;
811         if (N = 55) then
812
813              TxOut1 <= x1i(22);
814              TxOut2 <= x2i(22);
815              TxOut3 <= x3i(22);
816              TxOut4 <= x4i(22);
817              TxOut5 <= x5i(22);
818              TxOut6 <= x6i(22);
819              TxOut7 <= x7i(22);
820              TxOut8 <= x8i(22);
821         end if;
822         if (N = 56) then
823
824              TxOut1 <= x1i(23);
825              TxOut2 <= x2i(23);
826              TxOut3 <= x3i(23);
827              TxOut4 <= x4i(23);
828              TxOut5 <= x5i(23);
829              TxOut6 <= x6i(23);
830              TxOut7 <= x7i(23);
831              TxOut8 <= x8i(23);
832         end if;
833         if (N = 57) then
834
835              TxOut1 <= x1i(24);
836              TxOut2 <= x2i(24);
837              TxOut3 <= x3i(24);
838              TxOut4 <= x4i(24);
839              TxOut5 <= x5i(24);
840              TxOut6 <= x6i(24);
841              TxOut7 <= x7i(24);
842              TxOut8 <= x8i(24);
843         end if;
844         if (N = 58) then
845
846              TxOut1 <= x1i(25);
847              TxOut2 <= x2i(25);
848              TxOut3 <= x3i(25);
849              TxOut4 <= x4i(25);
850              TxOut5 <= x5i(25);
851              TxOut6 <= x6i(25);
852              TxOut7 <= x7i(25);
853              TxOut8 <= x8i(25);
854         end if;
855         if (N = 59) then
856
857              TxOut1 <= x1i(26);
858              TxOut2 <= x2i(26);
859              TxOut3 <= x3i(26);
860              TxOut4 <= x4i(26);
861              TxOut5 <= x5i(26);
862              TxOut6 <= x6i(26);
863              TxOut7 <= x7i(26);
864              TxOut8 <= x8i(26);
865         end if;
866         if (N = 60) then
867
868              TxOut1 <= x1i(27);
869              TxOut2 <= x2i(27);
870              TxOut3 <= x3i(27);
871              TxOut4 <= x4i(27);
872              TxOut5 <= x5i(27);
873              TxOut6 <= x6i(27);
874              TxOut7 <= x7i(27);
875              TxOut8 <= x8i(27);
876         end if;
```

```
877         if (N = 61) then
878
879              TxOut1 <= x11(28);
880              TxOut2 <= x21(28);
881              TxOut3 <= x31(28);
882              TxOut4 <= x41(28);
883              TxOut5 <= x51(28);
884              TxOut6 <= x61(28);
885              TxOut7 <= x71(28);
886              TxOut8 <= x81(28);
887         end if;
888         if (N = 62) then
889
890              TxOut1 <= x11(29);
891              TxOut2 <= x21(29);
892              TxOut3 <= x31(29);
893              TxOut4 <= x41(29);
894              TxOut5 <= x51(29);
895              TxOut6 <= x61(29);
896              TxOut7 <= x71(29);
897              TxOut8 <= x81(29);
898         end if;
899         if (N = 63) then
900
901              TxOut1 <= x11(30);
902              TxOut2 <= x21(30);
903              TxOut3 <= x31(30);
904              TxOut4 <= x41(30);
905              TxOut5 <= x51(30);
906              TxOut6 <= x61(30);
907              TxOut7 <= x71(30);
908              TxOut8 <= x81(30);
909         end if;
910         if (N = 64) then
911
912              TxOut1 <= x11(31);
913              TxOut2 <= x21(31);
914              TxOut3 <= x31(31);
915              TxOut4 <= x41(31);
916              TxOut5 <= x51(31);
917              TxOut6 <= x61(31);
918              TxOut7 <= x71(31);
919              TxOut8 <= x81(31);
920         end if;
921         if (N = 65) then
922
923              TxOut1 <= '1';
924              TxOut2 <= '1';
925              TxOut3 <= '1';
926              TxOut4 <= '1';
927              TxOut5 <= '1';
928              TxOut6 <= '1';
929              TxOut7 <= '1';
930              TxOut8 <= '1';
931         end if;
932
933     end process;
```

โปรแกรมภาครับ

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_ARITH.ALL;
4  use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6
7
8  entity ofdm_rx is
9      Port ( led : out std_logic_vector (7 downto 0);
10            Sysclk : in  STD_LOGIC;
11            RxIn : in  STD_LOGIC_VECTOR (7 downto 0));
12
13 end ofdm_rx;
14
15 architecture Behavioral of ofdm_rx is
16    signal N : integer range 0 to 65 := 65;
17    signal Bclk : std_logic;
18    signal start : std_logic := '0';
19    signal Ctb : integer range 0 to 2603 := 2603;
20    signal Ct : integer range 0 to 171700 := 171700;
21    signal x1r,x2r,x3r,x4r,x5r,x6r,x7r,x8r :  STD_LOGIC_VECTOR (31 downto 0);
22    signal x1i,x2i,x3i,x4i,x5i,x6i,x7i,x8i :  STD_LOGIC_VECTOR (31 downto 0);
23    signal y1r,y2r,y3r,y4r,y5r,y6r,y7r,y8r :integer;
24    signal y1i,y2i,y3i,y4i,y5i,y6i,y7i,y8i :integer  ;
25    signal xx1r,xx2r,xx3r,xx4r,xx5r,xx6r,xx7r,xx8r : integer;
26    signal xx1i,xx2i,xx3i,xx4i,xx5i,xx6i,xx7i,xx8i : integer;
27    signal xx : integer;
28 begin
29
30 GenClock: process(Sysclk)
31  begin
32  if (rising_edge(Sysclk)) then
33    if start = '1' and Ctb = 2603 then
34     Ctb <= 0;
35     Bclk <= '1';
36    elsif start = '1' and Ctb /= 2603 then
37     Ctb <= Ctb+1;
38     Bclk <= '0';
39    elsif start = '0' then
40    Ctb <= 2603;
41    end if;
42  end if;
43  end process;
44
45 Listening: process(Sysclk)
46 begin
47 xx <= conv_integer(RxIn);
48    if (rising_edge(Sysclk)) then
49       if Ct = 171700 and xx = 0 and N = 65 then
50        start <= '1';
51        Ct <= 0;
52       elsif Ct = 171700 and xx = 255 then
53        start <= '0';
54       elsif Ct /= 171700 then
55        Ct <= Ct +1;
56       end if;
57    end if;
58 end process;
59
60 Receiveing: process(Bclk)
61 begin
62    if Bclk = '1' then
63       if N = 65 then N <= 0;
64       else N <= N+1;
65       end if;
66    end if;
67 end process;
68
69 Saving: process(N)
70 begin
71    case N is
72    when 0 =>
73
74    when 1 =>
75       x1r(0) <= RxIn(0);
76       x2r(0) <= RxIn(1);
77       x3r(0) <= RxIn(2);
78       x4r(0) <= RxIn(3);
79       x5r(0) <= RxIn(4);
80       x6r(0) <= RxIn(5);
81       x7r(0) <= RxIn(6);
82       x8r(0) <= RxIn(7);
83
84    when 2 =>
85       x1r(1) <= RxIn(0);
86       x2r(1) <= RxIn(1);
87       x3r(1) <= RxIn(2);
88       x4r(1) <= RxIn(3);
89       x5r(1) <= RxIn(4);
90       x6r(1) <= RxIn(5);
91       x7r(1) <= RxIn(6);
92       x8r(1) <= RxIn(7);
```

```
93      when 3 =>
94         x1r(2) <= RxIn(0);
95         x2r(2) <= RxIn(1);
96         x3r(2) <= RxIn(2);
97         x4r(2) <= RxIn(3);
98         x5r(2) <= RxIn(4);
99         x6r(2) <= RxIn(5);
100        x7r(2) <= RxIn(6);
101        x8r(2) <= RxIn(7);
102     when 4 =>
103        x1r(3) <= RxIn(0);
104        x2r(3) <= RxIn(1);
105        x3r(3) <= RxIn(2);
106        x4r(3) <= RxIn(3);
107        x5r(3) <= RxIn(4);
108        x6r(3) <= RxIn(5);
109        x7r(3) <= RxIn(6);
110        x8r(3) <= RxIn(7);
111     when 5 =>
112        x1r(4) <= RxIn(0);
113        x2r(4) <= RxIn(1);
114        x3r(4) <= RxIn(2);
115        x4r(4) <= RxIn(3);
116        x5r(4) <= RxIn(4);
117        x6r(4) <= RxIn(5);
118        x7r(4) <= RxIn(6);
119        x8r(4) <= RxIn(7);
120     when 6 =>
121        x1r(5) <= RxIn(0);
122        x2r(5) <= RxIn(1);
123        x3r(5) <= RxIn(2);
124        x4r(5) <= RxIn(3);
125        x5r(5) <= RxIn(4);
126        x6r(5) <= RxIn(5);
127        x7r(5) <= RxIn(6);
128        x8r(5) <= RxIn(7);
129     when 7 =>
130        x1r(6) <= RxIn(0);
131        x2r(6) <= RxIn(1);
132        x3r(6) <= RxIn(2);
133        x4r(6) <= RxIn(3);
134        x5r(6) <= RxIn(4);
135        x6r(6) <= RxIn(5);
136        x7r(6) <= RxIn(6);
137        x8r(6) <= RxIn(7);
138     when 8 =>
139        x1r(7) <= RxIn(0);
140        x2r(7) <= RxIn(1);
141        x3r(7) <= RxIn(2);
142        x4r(7) <= RxIn(3);
143        x5r(7) <= RxIn(4);
144        x6r(7) <= RxIn(5);
145        x7r(7) <= RxIn(6);
146        x8r(7) <= RxIn(7);
147     when 9 =>
148        x1r(8) <= RxIn(0);
149        x2r(8) <= RxIn(1);
150        x3r(8) <= RxIn(2);
151        x4r(8) <= RxIn(3);
152        x5r(8) <= RxIn(4);
153        x6r(8) <= RxIn(5);
154        x7r(8) <= RxIn(6);
155        x8r(8) <= RxIn(7);
156     when 10 =>
157        x1r(9) <= RxIn(0);
158        x2r(9) <= RxIn(1);
159        x3r(9) <= RxIn(2);
160        x4r(9) <= RxIn(3);
161        x5r(9) <= RxIn(4);
162        x6r(9) <= RxIn(5);
163        x7r(9) <= RxIn(6);
164        x8r(9) <= RxIn(7);
165     when 11 =>
166        x1r(10) <= RxIn(0);
167        x2r(10) <= RxIn(1);
168        x3r(10) <= RxIn(2);
169        x4r(10) <= RxIn(3);
170        x5r(10) <= RxIn(4);
171        x6r(10) <= RxIn(5);
172        x7r(10) <= RxIn(6);
173        x8r(10) <= RxIn(7);
174     when 12 =>
175        x1r(11) <= RxIn(0);
176        x2r(11) <= RxIn(1);
177        x3r(11) <= RxIn(2);
178        x4r(11) <= RxIn(3);
179        x5r(11) <= RxIn(4);
180        x6r(11) <= RxIn(5);
181        x7r(11) <= RxIn(6);
182        x8r(11) <= RxIn(7);
183     when 13 =>
184        x1r(12) <= RxIn(0);
185        x2r(12) <= RxIn(1);
186        x3r(12) <= RxIn(2);
187        x4r(12) <= RxIn(3);
188        x5r(12) <= RxIn(4);
189        x6r(12) <= RxIn(5);
190        x7r(12) <= RxIn(6);
191        x8r(12) <= RxIn(7);
```

```
192     when 14 =>
193        x1r(13) <= RxIn(0);
194        x2r(13) <= RxIn(1);
195        x3r(13) <= RxIn(2);
196        x4r(13) <= RxIn(3);
197        x5r(13) <= RxIn(4);
198        x6r(13) <= RxIn(5);
199        x7r(13) <= RxIn(6);
200        x8r(13) <= RxIn(7);
201     when 15 =>
202        x1r(14) <= RxIn(0);
203        x2r(14) <= RxIn(1);
204        x3r(14) <= RxIn(2);
205        x4r(14) <= RxIn(3);
206        x5r(14) <= RxIn(4);
207        x6r(14) <= RxIn(5);
208        x7r(14) <= RxIn(6);
209        x8r(14) <= RxIn(7);
210     when 16 =>
211        x1r(15) <= RxIn(0);
212        x2r(15) <= RxIn(1);
213        x3r(15) <= RxIn(2);
214        x4r(15) <= RxIn(3);
215        x5r(15) <= RxIn(4);
216        x6r(15) <= RxIn(5);
217        x7r(15) <= RxIn(6);
218        x8r(15) <= RxIn(7);
219     when 17 =>
220        x1r(16) <= RxIn(0);
221        x2r(16) <= RxIn(1);
222        x3r(16) <= RxIn(2);
223        x4r(16) <= RxIn(3);
224        x5r(16) <= RxIn(4);
225        x6r(16) <= RxIn(5);
226        x7r(16) <= RxIn(6);
227        x8r(16) <= RxIn(7);
228     when 18 =>
229        x1r(17) <= RxIn(0);
230        x2r(17) <= RxIn(1);
231        x3r(17) <= RxIn(2);
232        x4r(17) <= RxIn(3);
233        x5r(17) <= RxIn(4);
234        x6r(17) <= RxIn(5);
235        x7r(17) <= RxIn(6);
236        x8r(17) <= RxIn(7);
237     when 19 =>
238        x1r(18) <= RxIn(0);
239        x2r(18) <= RxIn(1);
240        x3r(18) <= RxIn(2);
241        x4r(18) <= RxIn(3);
242        x5r(18) <= RxIn(4);
243        x6r(18) <= RxIn(5);
244        x7r(18) <= RxIn(6);
245        x8r(18) <= RxIn(7);
246     when 20 =>
247        x1r(19) <= RxIn(0);
248        x2r(19) <= RxIn(1);
249        x3r(19) <= RxIn(2);
250        x4r(19) <= RxIn(3);
251        x5r(19) <= RxIn(4);
252        x6r(19) <= RxIn(5);
253        x7r(19) <= RxIn(6);
254        x8r(19) <= RxIn(7);
255     when 21 =>
256        x1r(20) <= RxIn(0);
257        x2r(20) <= RxIn(1);
258        x3r(20) <= RxIn(2);
259        x4r(20) <= RxIn(3);
260        x5r(20) <= RxIn(4);
261        x6r(20) <= RxIn(5);
262        x7r(20) <= RxIn(6);
263        x8r(20) <= RxIn(7);
264     when 22 =>
265        x1r(21) <= RxIn(0);
266        x2r(21) <= RxIn(1);
267        x3r(21) <= RxIn(2);
268        x4r(21) <= RxIn(3);
269        x5r(21) <= RxIn(4);
270        x6r(21) <= RxIn(5);
271        x7r(21) <= RxIn(6);
272        x8r(21) <= RxIn(7);
273     when 23 =>
274        x1r(22) <= RxIn(0);
275        x2r(22) <= RxIn(1);
276        x3r(22) <= RxIn(2);
277        x4r(22) <= RxIn(3);
278        x5r(22) <= RxIn(4);
279        x6r(22) <= RxIn(5);
280        x7r(22) <= RxIn(6);
281        x8r(22) <= RxIn(7);
282     when 24 =>
283        x1r(23) <= RxIn(0);
284        x2r(23) <= RxIn(1);
285        x3r(23) <= RxIn(2);
286        x4r(23) <= RxIn(3);
287        x5r(23) <= RxIn(4);
288        x6r(23) <= RxIn(5);
289        x7r(23) <= RxIn(6);
290        x8r(23) <= RxIn(7);
```

```
291      when 25 =>
292         x1r(24) <= RxIn(0);
293         x2r(24) <= RxIn(1);
294         x3r(24) <= RxIn(2);
295         x4r(24) <= RxIn(3);
296         x5r(24) <= RxIn(4);
297         x6r(24) <= RxIn(5);
298         x7r(24) <= RxIn(6);
299         x8r(24) <= RxIn(7);
300      when 26 =>
301         x1r(25) <= RxIn(0);
302         x2r(25) <= RxIn(1);
303         x3r(25) <= RxIn(2);
304         x4r(25) <= RxIn(3);
305         x5r(25) <= RxIn(4);
306         x6r(25) <= RxIn(5);
307         x7r(25) <= RxIn(6);
308         x8r(25) <= RxIn(7);
309      when 27 =>
310         x1r(26) <= RxIn(0);
311         x2r(26) <= RxIn(1);
312         x3r(26) <= RxIn(2);
313         x4r(26) <= RxIn(3);
314         x5r(26) <= RxIn(4);
315         x6r(26) <= RxIn(5);
316         x7r(26) <= RxIn(6);
317         x8r(26) <= RxIn(7);
318      when 28 =>
319         x1r(27) <= RxIn(0);
320         x2r(27) <= RxIn(1);
321         x3r(27) <= RxIn(2);
322         x4r(27) <= RxIn(3);
323         x5r(27) <= RxIn(4);
324         x6r(27) <= RxIn(5);
325         x7r(27) <= RxIn(6);
326         x8r(27) <= RxIn(7);
327      when 29 =>
328         x1r(28) <= RxIn(0);
329         x2r(28) <= RxIn(1);
330         x3r(28) <= RxIn(2);
331         x4r(28) <= RxIn(3);
332         x5r(28) <= RxIn(4);
333         x6r(28) <= RxIn(5);
334         x7r(28) <= RxIn(6);
335         x8r(28) <= RxIn(7);
336      when 30 =>
337         x1r(29) <= RxIn(0);
338         x2r(29) <= RxIn(1);
339         x3r(29) <= RxIn(2);
340         x4r(29) <= RxIn(3);
341         x5r(29) <= RxIn(4);
342         x6r(29) <= RxIn(5);
343         x7r(29) <= RxIn(6);
344         x8r(29) <= RxIn(7);
345      when 31 =>
346         x1r(30) <= RxIn(0);
347         x2r(30) <= RxIn(1);
348         x3r(30) <= RxIn(2);
349         x4r(30) <= RxIn(3);
350         x5r(30) <= RxIn(4);
351         x6r(30) <= RxIn(5);
352         x7r(30) <= RxIn(6);
353         x8r(30) <= RxIn(7);
354      when 32 =>
355         x1r(31) <= RxIn(0);
356         x2r(31) <= RxIn(1);
357         x3r(31) <= RxIn(2);
358         x4r(31) <= RxIn(3);
359         x5r(31) <= RxIn(4);
360         x6r(31) <= RxIn(5);
361         x7r(31) <= RxIn(6);
362         x8r(31) <= RxIn(7);
363      when 33 =>
364         x1i(0) <= RxIn(0);
365         x2i(0) <= RxIn(1);
366         x3i(0) <= RxIn(2);
367         x4i(0) <= RxIn(3);
368         x5i(0) <= RxIn(4);
369         x6i(0) <= RxIn(5);
370         x7i(0) <= RxIn(6);
371         x8i(0) <= RxIn(7);
372
373      when 34 =>
374         x1i(1) <= RxIn(0);
375         x2i(1) <= RxIn(1);
376         x3i(1) <= RxIn(2);
377         x4i(1) <= RxIn(3);
378         x5i(1) <= RxIn(4);
379         x6i(1) <= RxIn(5);
380         x7i(1) <= RxIn(6);
381         x8i(1) <= RxIn(7);
382      when 35 =>
383         x1i(2) <= RxIn(0);
384         x2i(2) <= RxIn(1);
385         x3i(2) <= RxIn(2);
386         x4i(2) <= RxIn(3);
387         x5i(2) <= RxIn(4);
388         x6i(2) <= RxIn(5);
389         x7i(2) <= RxIn(6);
390         x8i(2) <= RxIn(7);
```

```
391      when 36 =>
392         x1i(3) <= RxIn(0);
393         x2i(3) <= RxIn(1);
394         x3i(3) <= RxIn(2);
395         x4i(3) <= RxIn(3);
396         x5i(3) <= RxIn(4);
397         x6i(3) <= RxIn(5);
398         x7i(3) <= RxIn(6);
399         x8i(3) <= RxIn(7);
400      when 37 =>
401         x1i(4) <= RxIn(0);
402         x2i(4) <= RxIn(1);
403         x3i(4) <= RxIn(2);
404         x4i(4) <= RxIn(3);
405         x5i(4) <= RxIn(4);
406         x6i(4) <= RxIn(5);
407         x7i(4) <= RxIn(6);
408         x8i(4) <= RxIn(7);
409      when 38 =>
410         x1i(5) <= RxIn(0);
411         x2i(5) <= RxIn(1);
412         x3i(5) <= RxIn(2);
413         x4i(5) <= RxIn(3);
414         x5i(5) <= RxIn(4);
415         x6i(5) <= RxIn(5);
416         x7i(5) <= RxIn(6);
417         x8i(5) <= RxIn(7);
418      when 39 =>
419         x1i(6) <= RxIn(0);
420         x2i(6) <= RxIn(1);
421         x3i(6) <= RxIn(2);
422         x4i(6) <= RxIn(3);
423         x5i(6) <= RxIn(4);
424         x6i(6) <= RxIn(5);
425         x7i(6) <= RxIn(6);
426         x8i(6) <= RxIn(7);
427      when 40 =>
428         x1i(7) <= RxIn(0);
429         x2i(7) <= RxIn(1);
430         x3i(7) <= RxIn(2);
431         x4i(7) <= RxIn(3);
432         x5i(7) <= RxIn(4);
433         x6i(7) <= RxIn(5);
434         x7i(7) <= RxIn(6);
435         x8i(7) <= RxIn(7);
436      when 41 =>
437         x1i(8) <= RxIn(0);
438         x2i(8) <= RxIn(1);
439         x3i(8) <= RxIn(2);
440         x4i(8) <= RxIn(3);
441         x5i(8) <= RxIn(4);
442         x6i(8) <= RxIn(5);
443         x7i(8) <= RxIn(6);
444         x8i(8) <= RxIn(7);
445      when 42 =>
446         x1i(9) <= RxIn(0);
447         x2i(9) <= RxIn(1);
448         x3i(9) <= RxIn(2);
449         x4i(9) <= RxIn(3);
450         x5i(9) <= RxIn(4);
451         x6i(9) <= RxIn(5);
452         x7i(9) <= RxIn(6);
453         x8i(9) <= RxIn(7);
454      when 43 =>
455         x1i(10) <= RxIn(0);
456         x2i(10) <= RxIn(1);
457         x3i(10) <= RxIn(2);
458         x4i(10) <= RxIn(3);
459         x5i(10) <= RxIn(4);
460         x6i(10) <= RxIn(5);
461         x7i(10) <= RxIn(6);
462         x8i(10) <= RxIn(7);
463      when 44 =>
464         x1i(11) <= RxIn(0);
465         x2i(11) <= RxIn(1);
466         x3i(11) <= RxIn(2);
467         x4i(11) <= RxIn(3);
468         x5i(11) <= RxIn(4);
469         x6i(11) <= RxIn(5);
470         x7i(11) <= RxIn(6);
471         x8i(11) <= RxIn(7);
472      when 45 =>
473         x1i(12) <= RxIn(0);
474         x2i(12) <= RxIn(1);
475         x3i(12) <= RxIn(2);
476         x4i(12) <= RxIn(3);
477         x5i(12) <= RxIn(4);
478         x6i(12) <= RxIn(5);
479         x7i(12) <= RxIn(6);
480         x8i(12) <= RxIn(7);
481      when 46 =>
482         x1i(13) <= RxIn(0);
483         x2i(13) <= RxIn(1);
484         x3i(13) <= RxIn(2);
485         x4i(13) <= RxIn(3);
486         x5i(13) <= RxIn(4);
487         x6i(13) <= RxIn(5);
488         x7i(13) <= RxIn(6);
489         x8i(13) <= RxIn(7);
```

```
490      when 47 =>
491         x1i(14) <= RxIn(0);
492         x2i(14) <= RxIn(1);
493         x3i(14) <= RxIn(2);
494         x4i(14) <= RxIn(3);
495         x5i(14) <= RxIn(4);
496         x6i(14) <= RxIn(5);
497         x7i(14) <= RxIn(6);
498         x8i(14) <= RxIn(7);
499      when 48 =>
500         x1i(15) <= RxIn(0);
501         x2i(15) <= RxIn(1);
502         x3i(15) <= RxIn(2);
503         x4i(15) <= RxIn(3);
504         x5i(15) <= RxIn(4);
505         x6i(15) <= RxIn(5);
506         x7i(15) <= RxIn(6);
507         x8i(15) <= RxIn(7);
508      when 49 =>
509         x1i(16) <= RxIn(0);
510         x2i(16) <= RxIn(1);
511         x3i(16) <= RxIn(2);
512         x4i(16) <= RxIn(3);
513         x5i(16) <= RxIn(4);
514         x6i(16) <= RxIn(5);
515         x7i(16) <= RxIn(6);
516         x8i(16) <= RxIn(7);
517      when 50 =>
518         x1i(17) <= RxIn(0);
519         x2i(17) <= RxIn(1);
520         x3i(17) <= RxIn(2);
521         x4i(17) <= RxIn(3);
522         x5i(17) <= RxIn(4);
523         x6i(17) <= RxIn(5);
524         x7i(17) <= RxIn(6);
525         x8i(17) <= RxIn(7);
526      when 51 =>
527         x1i(18) <= RxIn(0);
528         x2i(18) <= RxIn(1);
529         x3i(18) <= RxIn(2);
530         x4i(18) <= RxIn(3);
531         x5i(18) <= RxIn(4);
532         x6i(18) <= RxIn(5);
533         x7i(18) <= RxIn(6);
534         x8i(18) <= RxIn(7);
535      when 52 =>
536         x1i(19) <= RxIn(0);
537         x2i(19) <= RxIn(1);
538         x3i(19) <= RxIn(2);
539         x4i(19) <= RxIn(3);
540         x5i(19) <= RxIn(4);
541         x6i(19) <= RxIn(5);
542         x7i(19) <= RxIn(6);
543         x8i(19) <= RxIn(7);
544      when 53 =>
545         x1i(20) <= RxIn(0);
546         x2i(20) <= RxIn(1);
547         x3i(20) <= RxIn(2);
548         x4i(20) <= RxIn(3);
549         x5i(20) <= RxIn(4);
550         x6i(20) <= RxIn(5);
551         x7i(20) <= RxIn(6);
552         x8i(20) <= RxIn(7);
553      when 54 =>
554         x1i(21) <= RxIn(0);
555         x2i(21) <= RxIn(1);
556         x3i(21) <= RxIn(2);
557         x4i(21) <= RxIn(3);
558         x5i(21) <= RxIn(4);
559         x6i(21) <= RxIn(5);
560         x7i(21) <= RxIn(6);
561         x8i(21) <= RxIn(7);
562      when 55 =>
563         x1i(22) <= RxIn(0);
564         x2i(22) <= RxIn(1);
565         x3i(22) <= RxIn(2);
566         x4i(22) <= RxIn(3);
567         x5i(22) <= RxIn(4);
568         x6i(22) <= RxIn(5);
569         x7i(22) <= RxIn(6);
570         x8i(22) <= RxIn(7);
571      when 56 =>
572         x1i(23) <= RxIn(0);
573         x2i(23) <= RxIn(1);
574         x3i(23) <= RxIn(2);
575         x4i(23) <= RxIn(3);
576         x5i(23) <= RxIn(4);
577         x6i(23) <= RxIn(5);
578         x7i(23) <= RxIn(6);
579         x8i(23) <= RxIn(7);
```

```
580     when 57 =>
581        x1i(24) <= RxIn(0);
582        x2i(24) <= RxIn(1);
583        x3i(24) <= RxIn(2);
584        x4i(24) <= RxIn(3);
585        x5i(24) <= RxIn(4);
586        x6i(24) <= RxIn(5);
587        x7i(24) <= RxIn(6);
588        x8i(24) <= RxIn(7);
589     when 58 =>
590        x1i(25) <= RxIn(0);
591        x2i(25) <= RxIn(1);
592        x3i(25) <= RxIn(2);
593        x4i(25) <= RxIn(3);
594        x5i(25) <= RxIn(4);
595        x6i(25) <= RxIn(5);
596        x7i(25) <= RxIn(6);
597        x8i(25) <= RxIn(7);
598     when 59 =>
599        x1i(26) <= RxIn(0);
600        x2i(26) <= RxIn(1);
601        x3i(26) <= RxIn(2);
602        x4i(26) <= RxIn(3);
603        x5i(26) <= RxIn(4);
604        x6i(26) <= RxIn(5);
605        x7i(26) <= RxIn(6);
606        x8i(26) <= RxIn(7);
607     when 60 =>
608        x1i(27) <= RxIn(0);
609        x2i(27) <= RxIn(1);
610        x3i(27) <= RxIn(2);
611        x4i(27) <= RxIn(3);
612        x5i(27) <= RxIn(4);
613        x6i(27) <= RxIn(5);
614        x7i(27) <= RxIn(6);
615        x8i(27) <= RxIn(7);
616     when 61 =>
617        x1i(28) <= RxIn(0);
618        x2i(28) <= RxIn(1);
619        x3i(28) <= RxIn(2);
620        x4i(28) <= RxIn(3);
621        x5i(28) <= RxIn(4);
622        x6i(28) <= RxIn(5);
623        x7i(28) <= RxIn(6);
624        x8i(28) <= RxIn(7);
625     when 62 =>
626        x1i(29) <= RxIn(0);
627        x2i(29) <= RxIn(1);
628        x3i(29) <= RxIn(2);
629        x4i(29) <= RxIn(3);
630        x5i(29) <= RxIn(4);
631        x6i(29) <= RxIn(5);
632        x7i(29) <= RxIn(6);
633        x8i(29) <= RxIn(7);
634     when 63 =>
635        x1i(30) <= RxIn(0);
636        x2i(30) <= RxIn(1);
637        x3i(30) <= RxIn(2);
638        x4i(30) <= RxIn(3);
639        x5i(30) <= RxIn(4);
640        x6i(30) <= RxIn(5);
641        x7i(30) <= RxIn(6);
642        x8i(30) <= RxIn(7);
643     when 64 =>
644        x1i(31) <= RxIn(0);
645        x2i(31) <= RxIn(1);
646        x3i(31) <= RxIn(2);
647        x4i(31) <= RxIn(3);
648        x5i(31) <= RxIn(4);
649        x6i(31) <= RxIn(5);
650        x7i(31) <= RxIn(6);
651        x8i(31) <= RxIn(7);
652     when 65 =>
653
654
655
656     end case;
657
658  xx1r <= conv_integer(x1r);
659  xx2r <= conv_integer(x2r);
660  xx3r <= conv_integer(x3r);
661  xx4r <= conv_integer(x4r);
662  xx5r <= conv_integer(x5r);
663  xx6r <= conv_integer(x6r);
664  xx7r <= conv_integer(x7r);
665  xx8r <= conv_integer(x8r);
666
667  xx1i <= conv_integer(x1i);
668  xx2i <= conv_integer(x2i);
669  xx3i <= conv_integer(x3i);
670  xx4i <= conv_integer(x4i);
671  xx5i <= conv_integer(x5i);
672  xx6i <= conv_integer(x6i);
673  xx7i <= conv_integer(x7i);
674  xx8i <= conv_integer(x8i);
675
676  end process;
677
```

```
678   process(sysclk)
679   begin
680   if sysclk'event and sysclk='1' then
681
682   y1r <= (xx1r*1000)+(xx2r*1000)+(xx3r*1000)+(xx4r*1000)+(xx5r*1000)+(xx6r*1000)+(xx7r*1000)+(xx8r*1000);
683   y2r <= (xx1r*1000)+(xx2r*707)+(xx3r*0)+(xx4r*(-707))+(xx5r*(-1000))+(xx6r*(-707))+(xx7r*0)+(xx8r*707);
684   y3r <= (xx1r*1000)+(xx2r*0)+(xx3r*(-1000))+(xx4r*0)+(xx5r*1000)+(xx6r*0)+(xx7r*(-1000))+(xx8r*0);
685   y4r <= (xx1r*1000)+(xx2r*(-707))+(xx3r*0)+(xx4r*707)+(xx5r*(-1000))+(xx6r*707)+(xx7r*0)+(xx8r*(-707));
686   y5r <= (xx1r*1000)+(xx2r*(-1000))+(xx3r*1000)+(xx4r*(-1000))+(xx5r*1000)+(xx6r*(-1000))
687          +(xx7r*1000)+(xx8r*(-1000));
688   y6r <= (xx1r*1000)+(xx2r*(-707))+(xx3r*0)+(xx4r*707)+(xx5r*(-1000))+(xx6r*707)+(xx7r*0)+(xx8r*(-707));
689   y7r <= (xx1r*1000)+(xx2r*0)+(xx3r*(-1000))+(xx4r*0)+(xx5r*1000)+(xx6r*0)+(xx7r*(-1000))+(xx8r*0);
690   y8r <= (xx1r*1000)+(xx2r*707)+(xx3r*0)+(xx4r*(-707))+(xx5r*(-1000))+(xx6r*(-707))+(xx7r*0)+(xx8r*707);
691
692   y1i <= (xx1i*1000)+(xx2i*1000)+(xx3i*1000)+(xx4i*1000)+(xx5i*1000)+(xx6i*1000)+(xx7i*1000)+(xx8i*1000);
693   y2i <= (xx1i*0)+(xx2i*(-707))+(xx3i*(-1000))+(xx4i*(-707))+(xx5i*0)+(xx6i*707)+(xx7i*1000)+(xx8i*707);
694   y3i <= (xx1i*0)+(xx2i*(-1000))+(xx3i*0)+(xx4i*1000)+(xx5i*0)+(xx6i*(-1000))+(xx7i*0)+(xx8i*1000);
695   y4i <= (xx1i*0)+(xx2i*(-707))+(xx3i*1000)+(xx4i*(-707))+(xx5i*0)+(xx6i*707)+(xx7i*(-1000))+(xx8i*707);
696   y5i <= 0;
697   y6i <= (xx1i*0)+(xx2i*707)+(xx3i*(-1000))+(xx4i*707)+(xx5i*0)+(xx6i*(-707))+(xx7i*1000)+(xx8i*(-707));
698   y7i <= (xx1i*0)+(xx2i*1000)+(xx3i*0)+(xx4i*(-1000))+(xx5i*0)+(xx6i*1000)+(xx7i*0)+(xx8i*(-1000));
699   y8i <= (xx1i*0)+(xx2i*707)+(xx3i*1000)+(xx4i*707)+(xx5i*0)+(xx6i*(-707))+(xx7i*(-1000))+(xx8i*(-707));
700
701   if y1r = -10000000 then led(0)<='1'; end if;
702   if y1r = 10000000  then led(0)<='0'; end if;
703   if y2r = -10000000 then led(1)<='1'; end if;
704   if y2r = 10000000  then led(1)<='0'; end if;
705   if y3r = -10000000 then led(2)<='1'; end if;
706   if y3r = 10000000  then led(2)<='0'; end if;
707   if y4r = -10000000 then led(3)<='1'; end if;
708   if y4r = 10000000  then led(3)<='0'; end if;
709   if y5r = -10000000 then led(4)<='1'; end if;
710   if y5r = 10000000  then led(4)<='0'; end if;
711   if y6r = -10000000 then led(5)<='1'; end if;
712   if y6r = 10000000  then led(5)<='0'; end if;
713   if y7r = -10000000 then led(6)<='1'; end if;
714   if y7r = 10000000  then led(6)<='0'; end if;
715   if y8r = -10000000 then led(7)<='1'; end if;
716   if y8r = 10000000  then led(7)<='0'; end if;
717
718   end if;
719
720   end process;
721
722   end Behavioral;
723
```

ส่วน FFT

ส่วน demodulation