ขั้นตอนการทำ Partial Reconfigurable วงจรฮาร์ดแวร์บนเอฟพีจีเอ
The Proposed Process of Partial Reconfigurable Circuit Design Based on FPGA

ฮาดีย์ หมัดอาด้ำ
Hadee Mad-a-dum

วิทยานิพนธ์นี้สำหรับการศึกษาตามหลักสูตรปริญญา
วิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์
มหาวิทยาลัยสงขลานครินทร์
A Thesis Submitted in Fulfillment of the Requirements for the Degree of
Master of Engineering in Computer Engineering
Prince of Songkla University
2558

ชื่อวิทยานิพนธ์ ขั้นตอนการทำ Partial Reconfigurable วงจรฮาร์ดแวร์บนเอฟพีจีเอ
ผู้เขียน นาย ฮาดีย์ หมัดอาด้ำ
สาขาวิชา วิศวกรรมคอมพิวเตอร์

อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

.....
(ผู้ช่วยศาสตราจารย์ ดร.วรรณรัช สันติอมรทัต)

คณะกรรมการสอบ

.....ประธานกรรมการ
(ดร.ปัญญยศ ไชยกาฬ)

.....กรรมการ
(ดร.ชนันท์กรณ์ จันแดง)

.....กรรมการ
(ผู้ช่วยศาสตราจาร์ย ดร.วรรณรัช สันติอมรทัต)

บัณฑิตวิทยาลัย มหาวิทยาลัยสงขลานครินทร์ อนุมัติให้นับวิทยานิพนธ์ฉบับนี้สำหรับการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

.....
(รองศาสตราจารย์ ดร.ธีระพล ศรีชนะ)
คณบดีบัณฑิตวิทยาลัย

ขอรับรองว่า ผลงานวิจัยนี้มาจากการศึกษาวิจัยของนักศึกษาเอง และได้แสดงความขอบคุณบุคคลที่มีส่วนช่วยเหลือแล้ว

ลงชื่อ.....
(ผู้ช่วยศาสตราจารย์ ดร.วรรณรัช สันติอมรทัต)
อาจารย์ที่ปรึกษาวิทยานิพนธ์

ลงชื่อ.....
(ฮาดีย์ หมัดอาด้ำ)
นักศึกษา

ชื่อวิทยานิพนธ์ ขั้นตอนการทำ Partial Reconfigurable วงจรฮาร์ดแวร์บนเอฟพีจีเอ
ผู้เขียน นาย ฮาดีย์ หมัดอาด้ำ
สาขาวิชา วิศวกรรมคอมพิวเตอร์
ปีการศึกษา 2557

บทคัดย่อ

Partial reconfiguration คือ การปรับเปลี่ยนวงจรภายในเอฟพีจีเอ ในขณะกำลังทำงาน โดยปรับเปลี่ยนวงจรภายในเพียงบางส่วนและไม่ส่งผลกระทบต่อวงจรอื่นที่ไม่มีการปรับเปลี่ยน แต่เนื่องจากกระบวนการทำ Partial reconfiguration มีความซับซ้อนและต้องมีความรู้เกี่ยวกับโครงสร้างของเอฟพีจีเอ ทำให้ความสามารถดังกล่าวไม่ได้ถูกนำมาใช้งานอย่างแพร่หลาย

งานวิจัยจึงได้นำเสนอขั้นตอนและกระบวนการรวมถึงซอฟต์แวร์ในการทำให้เอฟพีจีเอ สามารถปรับเปลี่ยนการทำงานได้แบบ Partial reconfiguration โดยเลือกใช้เอฟพีจีเอตระกูล Spartan-6 เนื่องจากเป็นเอฟพีจีเอขนาดกลาง ราคาถูก และมีโครงสร้างและทรัพยากรที่รองรับการทำ Partial reconfiguration แต่ไม่มีซอฟต์แวร์จากผู้ผลิตเอฟพีจีเอ ที่รองรับการออกแบบ Partial reconfiguration โดยตรง

งานวิจัยได้ทดสอบกระบวนการทำ Partial reconfiguration ของเอฟพีจีเอ โดยใช้วงจรเข้ารหัส (Advance Encription Standard หรือ AES) แบ่งเป็น AES ขนาด 128 บิต และ 192 บิต พร้อมกับคำนวณประสิทธิภาพของวงจรแบบ Partial reconfiguration การสูญเสียพลังงาน และการใช้ทรัพยากรภายในเอฟพีจีเอ ผลการทดลองพบว่า การทำ Partial reconfiguration ใช้ทรัพยากรเอฟพีจีเอ มากกว่าการทำ Full reconfiguration เล็กน้อยเฉพาะในส่วนของวงจรเชื่อมต่อการควบคุม Reconfiguration ทำให้การสูญเสียพลังงานของวงจรทั้งสองแบบมีค่าเท่ากัน นอกจากนั้นได้ออกแบบการโปรแกรมวงจรลงในเอฟพีจีเอ ผ่านระบบเครือข่ายเซนเซอร์ไร้สาย (Wireless Sensor Networks หรือ WSNs) มาตรฐาน ZigBee เพื่อเป็นตัวอย่างการนำไปประยุกต์ใช้กับโนดของระบบเครือข่ายเซนเซอร์ไร้สาย

คำสำคัญ: การปรับเปลี่ยนโครงสร้างเอฟพีจีเอแบบ Partial reconfiguration, เอฟพีจีเอ, เอฟพีจีเอตระกูล Spartan-6

Thesis Title The Proposed Process of Partial Reconfigurable Circuit Design Based on FPGA
Author Mr. Hadee Mad-a-dum
Major Program Computer Engineering
Academic Year 2014

ABSTRACT

Partial reconfiguration is the modification of specific hardware block inside FPGA while other blocks are still running. Since partial reconfiguration process requires a deeply knowledge about the structure of FPGA and is complicated. Therefore partial reconfigurable ability is not widely deployed.

This thesis represents the process of partial reconfiguration design including the related software. FPGA Spartan-6 is chosen because of its compact size and low cost device. Spartan-6 structure supports the partial reconfiguration but the official design tool is not provided.

This thesis has demonstrated the partial reconfiguration process using Advance Encription Standard (AES) 128 and 192 bits. Both of them are presented here to examine the partial reconfiguration performance and overhead. We has evaluated the power consumption and resource usaged. The result of the experiments show that the partial reconfiguration design consumes the resource usaged more than regular one a little bit. Thus the power consumptions of both designs are equally values. Finally the thesis presents the configuration download wirelessly based on ZigBee standard. This shows that the partial reconfigurable design is able to be applied on node in wireless sensor networks.

KEYWORDS: Partial reconfiguration, FPGA, Spartan-6

สารบัญ

สารบัญ (ต่อ)

หน้า

รายการตาราง

หน้า

รายการภาพประกอบ

หน้า

รายการภาพประกอบ (ต่อ)

หน้า

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญ

เครือข่ายเซนเซอร์ไร้สายประกอบด้วยอุปกรณ์ที่เรียกว่า เซนเซอร์โนด (Sensor node) จำนวนมาก โนดจะถูกเชื่อมต่อกันเป็นเครือข่ายไร้สายคลื่นวิทยุที่ย่านความถี่ 2.4 GHz ตามมาตรฐาน IEEE 802.15.4 โดยที่อุปกรณ์เซนเซอร์โนดจะประกอบด้วย หน่วยประมวลผล โมดูลการรับ-ส่งข้อมูลแบบไร้สาย หน่วยความจำ แบตเตอรี่ และเซนเซอร์ เนื่องจากเครือข่ายเซนเซอร์ไร้สาย (Wireless Sensor Networks, WSNs) [1] ถูกนำไปใช้งานกันอย่างแพร่หลายในปัจจุบัน ทำให้เกิดการพัฒนาศักยภาพของระบบเครือข่ายเซนเซอร์ไร้สายอย่างต่อเนื่องเพื่อตอบสนองความต้องการของผู้ใช้งาน ตัวอย่างเช่น การทำให้มีการประมวลผล และตอบสนองด้วยการตัดสินใจของระบบเอง ซึ่งจะใช้ข้อมูลจากสภาวะแวดล้อมที่เกิดขึ้นจริงในขณะนั้นมาช่วยในการตัดสินใจ หรือการพัฒนาคุณสมบัติตัวประมวลผลที่สามารถประมวลผลข้อมูลที่ซับซ้อนได้รวดเร็วและใช้พลังงานต่ำ หรือการพัฒนาการสื่อสารผ่านระบบเครือข่ายที่ออกแบบให้ใช้พลังงานต่ำ หรือการพัฒนาให้คุณสมบัติของเซนเซอร์ที่มีความหลากหลายให้เลือกใช้งาน หรือการพัฒนาให้ขนาดของตัวอุปกรณ์ที่เล็กลง เป็นต้น

อย่างไรก็ตามแนวคิดหลักของการใช้งานเครือข่ายเซนเซอร์ไร้สาย คือ จะต้องสามารถทำงานในสภาพแวดล้อมที่มีผลต่อการเชื่อมต่อเครือข่าย จนอาจทำให้โครงสร้างเครือข่ายมีการเปลี่ยนแปลง หรือรูปแบบของการทำงานของโนดจะต้องสามารถปรับตัว และเปลี่ยนแปลงตัวเองให้เหมาะสมตามข้อมูลในขณะนั้นที่ได้มาจากเซนเซอร์ ดังนั้นทิศทางการพัฒนาส่วนประมวลผลของเซนเซอร์โนดในอนาคต คือ หน่วยประมวลผลจะต้องมีความยืดหยุ่น สามารถปรับเปลี่ยนฟังก์ชันการประมวลผล หรือการใช้งานได้หลายรูปแบบตามสถานการณ์จริงที่ได้ข้อมูลมาจากเซนเซอร์ ดังนั้นเทคโนโลยีเอฟพีจีเอ (Field Programmable Gate Array; FPGA) จึงเป็นทางเลือกที่เหมาะสมที่สุดด้วยคุณสมบัติที่สามารถโปรแกรมให้โครงสร้างของวงจรภายในเอฟพีจีเอเปลี่ยนแปลงได้ นอกจากนี้แล้วโครงสร้างของเอฟพีจีเอเหมาะสำหรับการทำงานแบบขนาน จึงทำให้สามารถประมวลผลได้เร็วกว่าไมโครคอนโทรลเลอร์ทั่วไปโดยที่ไม่ต้องใช้ความถี่ของสัญญาณนาฬิกาที่สูง ทำให้กินกำลังไฟ (Power) ต่ำ ดังนั้นการนำเอฟพีจีเอ มาใช้เป็นเซนเซอร์โนดจะช่วยให้หน่วยประมวลผลสามารถแก้ไข หรือปรับการทำงานของวงจรภายในตัวเองได้สอดคล้องกับสถานการณ์จริงในขณะนั้น

Reconfiguration หรือเรียกว่าความสามารถปรับเปลี่ยนโครงสร้างได้ ซึ่งเป็นคุณสมบัติเด่นของเอฟพีจีเอ ที่สามารถเปลี่ยนโครงสร้างภายในของเอฟพีจีเอในระดับลอจิกเกตใหม่ได้ แบ่งออกเป็น 2 รูปแบบ คือ 1) Full reconfiguration คือ การแก้ไขหรือโปรแกรมวงจรลงบนเอฟพีจีเอแบบปกติ กล่าวคือ เมื่อต้องการโปรแกรมเอฟพีจีเอ จะต้องหยุดการทำงานเดิม และทำการล้างวงจรเดิมโดยการลบไฟล์ที่เก็บวงจรเดิม (.bit) ในหน่วยความจำแบบ RAM บนเอฟพีจีเอ แล้วจึงจะทำการโปรแกรมวงจรใหม่ ซึ่งการทำงานแบบนี้จะต้องมีการกำหนดค่าภายใน

(Configuration) โครงสร้างของเอฟพีจีเอใหม่ทั้งหมดทุกส่วน แม้ว่าจะใช้หรือไม่ใช้งานก็ตาม และอีกวิธี คือ 2) Partial reconfiguration (PR) คือ การแก้ไขหรือโปรแกรมลงบนเอฟพีจีเอเฉพาะบางส่วนโดยที่วงจรส่วนอื่น ๆ จะไม่มีผลกระทบ และยังคงสามารถทำงานได้ [2]

Partial reconfiguration ด้วยคุณสมบัติการปรับเปลี่ยนวงจรฮาร์ดแวร์ได้แบบ Runtime ทำให้มีความยืดหยุ่นในการใช้งาน งานบางประเภทต้องการปรับเปลี่ยนฮาร์ดแวร์ไปตามอินพุตที่รับมา หรือปรับเปลี่ยนฮาร์ดแวร์ให้เข้ากับสภาพแวดล้อม หรือโครงสร้างของการเชื่อมต่อเครือข่ายใหม่เพื่อให้ทำงานได้มีประสิทธิภาพที่สุด เช่น การเลือกประมวลผลอัลกอริทึมการลดสัญญาณรบกวน (Noise filter) ตามสภาพแวดล้อมที่เกิดขึ้น เพื่อใช้ในการลดสัญญาณรบกวนจากเซนเซอร์ เพื่อนำมาประมวลผลการติดตาม (Tracking) ของเซนเซอร์โนด กรณีต่าง ๆ เช่น โนดเคลื่อนที่แบบช้า , เคลื่อนที่แบบเร็ว, โนดเคลื่อนที่ปกติ หรือสภาพแวดล้อมกลางคืน ฝนตก เป็นต้น [3], การออกแบบวิธีการรักษาความมั่นคงของข้อมูลที่สามารถเพิ่มความเร็วของอัลกอริทึมด้วยเอฟพีจีเอ [4] และมีงานวิจัยเกี่ยวกับการสร้างวงจรด้วยเทคนิคการทำ Partial reconfiguration เพื่อประมวลผลอัลกอริทึมที่เกี่ยวข้องกับ Neural network [5] โดยต้องการให้ใช้ทรัพยากรและเวลาในการโปรแกรมน้อยที่สุด มีงานวิจัยที่นำ Partial reconfiguration มาใช้กับการสร้างวงจรที่สามารถประมวลผลอัลกอริทึมการแปลงรหัสวิดีโอมาทำการบีบอัดเพื่อส่ง Stream video ให้เหมาะสมกับแบนด์วิดท์ [6] นอกจากนี้ งานวิจัยเกี่ยวกับ Wireless sensor networks (WSNs) มีการใช้ Partial reconfiguration กับเซนเซอร์โนดที่สามารถโปรแกรมวงจรฮาร์ดแวร์ใหม่ได้ โดยรับวงจรที่จะต้องการโปรแกรมใหม่ผ่านทางเครือข่ายไร้สาย ทำให้เซนเซอร์โนดมีความยืดหยุ่นในการทำงานและเพิ่มประสิทธิภาพการประมวลผลที่สูงขึ้น เช่น ในงานวิจัย [7], [8], และ [9] เป็นต้น

เนื่องจากบริษัทผู้ผลิต คือ Xilinx ได้สร้างอุปกรณ์ช่วยออกแบบที่รองรับกับการออกแบบแบบ Partial reconfiguration คือ PlanAhead [10] ซึ่งมีข้อจำกัดการใช้งาน คือ รองรับการทำ Partial reconfiguration ได้เฉพาะเอฟพีจีเอตระกูล Virtex-4-Virtex-7, Artix-7 families and the Zynq™-7000 All Programmable SoC family เท่านั้น[2] ดังนั้นจะพบว่างานวิจัยส่วนใหญ่ออกแบบโดยใช้อุปกรณ์เอฟพีจีเอตระกูล Virtex ของ Xilinx ซึ่งเป็นเอฟพีจีเอที่มีขนาดพื้นที่ใหญ่ ส่งผลให้มีค่าใช้จ่ายสูง, พลังงานที่ใช้เพิ่มขึ้น, ขนาดไฟล์ของวงจรใหญ่ทำให้ใช้เวลาในการตั้งค่าวงจรนาน และอาจเกินความจำเป็นในบางประเภทงาน ในขณะเดียวกันเอฟพีจีเอตระกูล Spartan ซึ่งมีขนาดกระทัดรัด ราคาถูก ไม่สามารถใช้งานซอฟต์แวร์แวร์จากผู้ผลิตได้ ถึงแม้ว่าโดยโครงสร้างของเอฟพีจีเอตระกูล Spartan จะรองรับการทำงานแบบ Partial reconfiguration ก็ตาม [11] ดังนั้นงานวิจัยนี้จึงทำการศึกษาวิธีการทำ Partial reconfiguration โดยจะนำเสนอขั้นตอนการออกแบบวงจรแบบ Partial reconfiguration ตั้งแต่โปรแกรมช่วยออกแบบ EDA-Tools ที่รองรับ และกระบวนการสร้างทั้งหมด รวมถึงการสร้างวงจรตัวอย่างทดสอบการทำงานโดยออกแบบวงจรเข้ารหัสข้อมูล Advance Encryption Standard (AES) เข้ารหัสข้อมูลขนาด 128 บิต แบ่งเป็นวงจร AES ใช้คีย์ขนาด 128 และ 192 บิต ใช้อุปกรณ์ทดสอบ คือ บอร์ด Xilinx SP601 ซึ่งใช้เอฟพีจี Spartan-6 และรายงานผลทรัพยากรที่ใช้สร้างวงจรในเอฟพีจีเอ, กำลังงานสูญเสีย และเวลาที่ใช้โปรแกรมวงจร เพื่อแสดงให้เห็นว่าเอฟพีจีเอ ตระกูล Spartan-6 สามารถทำ Partial reconfiguration ได้ตามขั้นตอนที่ได้นำเสนอ รวมถึงการทำต้นแบบตัวอย่างโนด

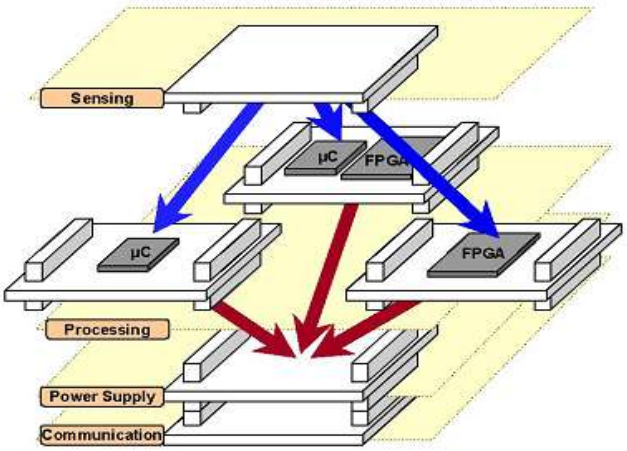
ที่ใช้เอฟพีจีเอในแบบ Partial reconfiguration ทำให้โนดมีความยืดหยุ่น และสามารถรองรับงานวิจัยในอนาคตที่ต้องการเปลี่ยนวงจรการทำงานของโนดได้สอดคล้องกับสภาพแวดล้อมหรือโครงสร้างเครือข่ายที่เปลี่ยนแปลงไปได้

1.2 ตรวจเอกสาร

การตรวจสอบเอกสารที่เกี่ยวข้องจะกล่าวถึง งานวิจัยที่เกี่ยวกับการออกแบบเซนเซอร์โนดให้มีคุณสมบัติที่สามารถโปรแกรมฮาร์ดแวร์ตัวเองได้ และงานวิจัยที่เกี่ยวกับการออกแบบโครงสร้างเอฟพีจีเอและเครื่องมือที่ใช้งาน เพื่อให้เอฟพีจีเอสามารถโปรแกรมแบบ Partial reconfiguration

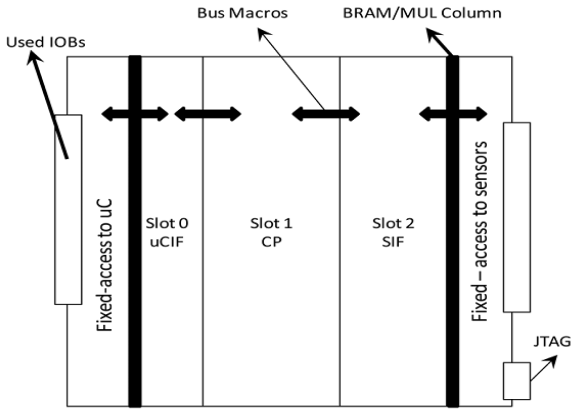
1.2.1 เซนเซอร์โนดที่สามารถโปรแกรมฮาร์ดแวร์ตัวเองได้

มีงานวิจัยพัฒนาเซนเซอร์โนดชื่อว่า Modular node [12] หรือ Cookie node เป็นการออกแบบให้โนดประกอบด้วยเซนเซอร์หลายตัวโนดเดียว เพื่อรองรับการใช้งานที่หลากหลาย โดยตัวเซนเซอร์โนดแบ่งเป็นชั้นการทำงาน (Layer) คือ ส่วนประมวลผล (Processing), แหล่งจ่ายพลังงาน (Power), ตัวตรวจวัด (Sensing) และ ภาคการสื่อสาร (Communication) ดังภาพประกอบ 1-1 โดยใช้งานเอฟพีจีเอ Xilinx Spartan XC3S200 ร่วมกับไมโครคอนโทรลเลอร์ ADUC812 ซึ่งเป็นไมโครคอนเลอร์ตระกูล 8052



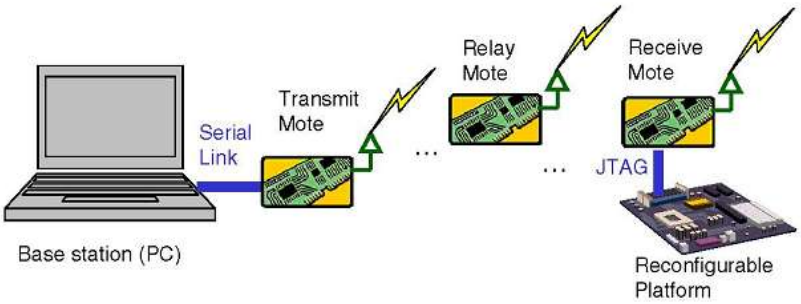
ภาพประกอบ 1-1 โครงสร้างของ Modular Node

Modular node มีโครงสร้างภายในเอฟพีจีเอดังภาพประกอบ 1-2 ในงานวิจัยนี้ [9] แบ่งพื้นที่ใช้งานออกเป็นสองส่วน คือ 1) Fixed area คือ พื้นที่ที่จะกำหนดการทำงานแน่นอน ไม่มีการเปลี่ยนแปลง โดยอยู่ที่ด้านขอบซ้ายและขวาของเอฟพีจีเอ ฝั่งซ้ายใช้สำหรับเชื่อมต่อกับไมโครคอนโทรลเลอร์ ฝั่งขวาเชื่อมต่อกับเซนเซอร์ 2) Reconfigurable area คือ พื้นที่สำหรับวางวงจรที่เป็นอัลกอริทึมของการทำงาน โดยพื้นที่นี้สามารถปรับเปลี่ยนวงจรอัลกอริทึมได้ตามการใช้งานจริง



ภาพประกอบ 1-2 การออกแบบโครงสร้างเอฟพีจีเอของ Modular Node

นอกจากนั้นยังมีงานวิจัย Runtime reconfiguration hardware [13] ที่ได้พัฒนาเซนเซอร์โนดโดยใช้เอฟพีจีเอ ให้สามารถโปรแกรมวงจรได้ผ่านทางเครือข่ายเซนเซอร์ไร้สาย โดยใช้ไมโครคอนโทรลเลอร์ (MSP430) เชื่อมต่อกับเอฟพีจีเอ เพื่อรับส่งข้อมูล และโปรแกรมวงจรบนเอฟพีจีเอผ่านทางโพรโทคอลการเชื่อมต่อแบบ JTAG โดยที่เอฟพีจีเอมีหน้าที่หลัก คือ ประมวลผลข้อมูลตามคำสั่ง และอ่านค่าจากเซนเซอร์ เป็นต้น ภาพรวมของระบบแสดงดังภาพประกอบ 1-3



ภาพประกอบ 1-3 ระบบเครือข่ายเซนเซอร์ไร้สายที่เซนเซอร์โนดโปรแกรมตัวเองได้

1.2.2 การปรับเปลี่ยนโครงสร้างเอฟพีจีเอและคุณสมบัติเอฟพีจีเอแบบ Partial Reconfiguration

ผู้วิจัยเลือกใช้เอฟพีจีเอจากผู้ผลิต Xilinx เป็นหลักเนื่องจากเป็นเอฟพีจีเอที่ถูกใช้อย่างแพร่หลาย เอฟพีจีเอในปัจจุบันมีหลายขนาด ตั้งแต่ขนาดเล็กสำหรับงานทดสอบหรือวงจรที่ไม่ซับซ้อนมาก ไปจนถึงขนาดใหญ่ที่สามารถเขียนวงจรขนาดใหญ่ได้โดยเปรียบเทียบได้ดังตารางที่ 1-1

ตารางที่ 1-1 เปรียบเทียบคุณสมบัติเอฟพีจีเอปัจจุบัน

คุณสมบัติ	**ตระกูลเอฟพีจีเอ**				
	Spartan-3	**Spartan-6**	**Virtex-4**	**Virtex-5**	**Virtex-6**
จำนวน Logic cell	1,728-74,880	3,840-147,443	13,824-200,448	30,720-331,776	74,496-566,784
Quiescent (W)	0.05-0.18	0.01-0.09	0.16-1.28	0.28-2.77	0.75-5.26
รองรับการใช้งาน Partial reconfiguration	✘	✓	✓	✓	✓
ราคาในท้องตลาด	> 11$	> 13$	> 147$	> 507$	> 463 $

จากตารางที่ 1-1 เป็นการเปรียบเทียบคุณสมบัติของเอฟพีจีเอปัจจุบันตั้งแต่ Spartan-3 จนถึง Virtex-6 พบว่าเอฟพีจีเอมีขนาดใหญ่ขึ้น การใช้กำลังไฟและราคาก็จะสูงขึ้นเช่นกัน สำหรับในงานวิจัยต้องการนำมาใช้กับระบบเครือข่ายเซนเซอร์ไร้สาย ดังนั้นจำเป็นต้องให้ความสำคัญกับเรื่องพลังงาน ซึ่งจะเห็นว่า Spartan-3 ใช้พลังงานต่ำสุด แต่ไม่รองรับการปรับเปลี่ยนโครงสร้างแบบ Partial reconfiguration ทำให้ไม่สามารถเลือก Spartan-3 ได้

ดังนั้นพบว่า Spartan-6 เป็นตัวเลือกที่เหมาะสม เพราะใช้พลังงานต่ำกว่ารุ่นอื่น ๆ ที่รองรับการทำ Partial reconfiguration นอกจากนี้ในเรื่องของราคา Spartan-6 มีราคาอยู่ที่ $13 ในขณะที่รุ่นถัดไป คือ virtex-4 มีราคา $145 ซึ่งสูงกว่าสิบเท่าโดยที่มีความสามารถในการทำ Partial reconfiguration ได้เหมือนกัน

1.2.3 การออกแบบเอฟพีจีเอเพื่อโปรแกรมแบบ Partial Reconfiguration

ผู้ผลิตเอฟพีจีเอ Xilinx ได้นำเสนอแนวคิด วิธีการ ซอฟต์แวร์และขั้นตอนการออกแบบวงจรแบบ Partial reconfiguration [2] พัฒนาบนเอฟพีจีเอ Virtex6 และทดสอบ โดยใช้วงจรแสดงผลบนแอลอีดี (LED) กะพริบในความเร็วที่แตกต่างกัน 3 ระดับ จากวงจรแบบ Partial reconfiguration 3 ชุดเพื่อให้เห็นการทำงานแบบ Partial reconfiguration

ส่วนงานวิจัย [14] นำเสนอการทำ Partial reconfiguration บนเอฟพีจีเอที่สามารถแบ่งพื้นที่สำหรับโปรแกรมวงจรซ้ำแบบเป็น 5 ส่วน โดยที่โมดูลวงจรแบบ Partial module 1 วงจรสามารถเลือกโปรแกรมวงจรลงในพื้นที่เอฟพีจีเอได้ทั้ง 5 ส่วน ซึ่งโดยทั่วไปนั้นการออกแบบโมดูลวงจร 1 วงจรสามารถโปรแกรมวงจรได้ในพื้นที่ ที่ได้กำหนดไว้เท่านั้น ดังนั้นทำให้สามารถออกแบบการวางตำแหน่งวงจรในเอฟพีจีเอได้อย่างอิสระ ซึ่งทั้งคุณสมบัติการแบ่งพื้นที่โปรแกรมซ้ำที่มากกว่า 1 พื้นที่ และการใช้โมดูลวงจรเดิม เพื่อโปรแกรมลงในพื้นที่เอฟพีจีเอได้อย่างอิสระยังไม่สามารถทำได้โดยใช้ซอฟต์แวร์ของผู้ผลิตเอฟพีจีเอ Xilinx ได้

งานวิจัย [15] Fast startup tool flow พัฒนาบนเอฟพีจีเอ Spartan-6 โดยนำเสนอกระบวนการเพื่อสร้างไฟล์วงจรทั้งแบบ Full bit file และ Partial bit file สำหรับทำ Partial reconfiguration โดยอ้างอิงจากขั้นตอนการทำงานของ Partial reconfiguration จากซอฟต์แวร์ PlanAhead ของ Xilinx พร้อมกับเสนอเทคนิคที่ทำให้การเริ่มทำงานของโปรแกรมเร็วขึ้น โดยทำให้โปรแกรมเริ่มการทำงานจาก Partial bit file ก่อน ซึ่งโดยปกติแล้วเอฟพีจีเอจะเริ่มการโปรแกรมวงจรจาก Full bit file ก่อน

จากที่กล่าวมานั้นงานวิจัยส่วนใหญ่จะใช้ Tools สำหรับการพัฒนาแตกต่างกันแต่จะอยู่บนพื้นฐานขั้นตอนการออกแบบเดียวกัน ตารางที่ 1-2 แสดงให้เห็นซอฟต์แวร์สำหรับพัฒนาเอฟพีจีแบบ Partial reconfiguration ที่ใช้ในแต่ละงานวิจัย ซึ่งมีคุณสมบัติประกอบไปด้วย

1. การแสดงผล ซอฟต์แวร์ส่วนใหญ่มีการแสดงผลเป็นแบบ Graphic User Interface (GUI) ให้เห็นโครงสร้างของเอฟพีจีเอ เพื่อความสะดวกกับผู้ใช้งานในการเลือกหรือกำหนดพื้นที่ภายในเอฟพีจีเอ มีเพียง Dream tools ที่ใช้รูปแบบ Script คำสั่งซึ่งทำให้การใช้งานยุ่งยาก

2. การเชื่อมต่อระหว่างพื้นที่ ส่วนใหญ่ใช้วิธี Bus macro ซึ่งเป็นการกำหนดให้ LUTs 1 คู่ทำการเชื่อมต่อเข้าด้วยกันเป็นบัสรับส่งข้อมูลขนาด 4 บิต นอกจากนั้นมีการใช้รูปแบบ Proxy ซึ่งเหมือนกับ Bus macro แต่จะกำหนดขนาด Bus ได้เพียง 1 บิต ต่อ 1 LUTs และยังมีการกำหนดการเชื่อมต่อแบบ Core unifier ที่ซอฟต์แวร์ Jbits ออกแบบให้ผู้ใช้สามารถกำหนดขนาด Bus macro ได้ตามต้องการโดยมีกราฟฟิคแสดงผล

3. ตระกูลเอฟพีจีเอที่รองรับพบว่าซอฟต์แวร์ทั้งหมดรองรับการใช้งานกับเอฟพีจีเอ Virtex แต่จะแตกต่างกันไป เช่น PlanAhead รองรับการใช้งาน Virtex-4–Virtex-7 และซอฟต์แวร์ GoAhead รองรับเวอร์ชัน Virtex-4–Virtex-7 รวมไปถึง Spartan-6

คุณสมบัติการทำ Partial reconfiguration แบ่งเป็นคุณสมบัติการสร้างพื้นที่โปรแกรมซ้ำได้หลายรูปแบบ เช่น แบบพื้นที่เดียว (Single style), แบบหลายพื้นที่ (Multiple style) และแบบ Module relocation คือ สามารถวางวงจรในตำแหน่งพื้นที่ไหนก็ได้ ที่ได้ออกแบบเป็น Reconfiguratble part

ตารางที่ 1-2 Tools สำหรับโปรแกรมแบบ Partial Reconfiguration

ซอฟต์แวร์ที่รองรับ Partial Reconfiguration	คุณสมบัติโปรแกรม				
	การแสดงผล	การเชื่อมต่อระหว่างพื้นที่	ตระกูล FPGA ที่รองรับ	Multi island	Module relocation
PlanAhead Xilinx [10]	GUI floorplan	Proxy logic	Virtex-4-Virtex-7	✘	✘
Dream Tools [16]	Script	Virtual border	Virtex-5	✓	✓
Jbits [17]	GUI floorplan	Core unifire	Virtex-2	✘	✓
OpenPR [18]	GUI floorplan	Bus macro	Virtex-4, Virtex-5	✘	✓
Recobus builder [19]	GUI floorplan	Bus macro	Virtex-2	✓	✓
GoAhead [20]	GUI floorplan	Bus macro	Virtex-4-Virtex-7, Spartan-6	✓	✓

เมื่อพิจารณาจากคุณสมบัติของซอฟต์แวร์ทั้งหมดแล้ว พบว่า GoAhead คือซอฟต์แวร์ที่เหมาะสมที่สุดในการนำมาใช้พัฒนาในงานวิจัย เนื่องจากมีคุณสมบัติการแสดงผลเป็น GUI และรองรับกับการใช้งานบน Spartan-6 นอกจากนี้ยังสามารถกำหนดคุณสมบัติการสร้างพื้นที่ได้หลายแบบ

1.3 วัตถุประสงค์

1. เพื่อศึกษาวิธีการโปรแกรมเอฟพีจีเอ แบบ Partial reconfiguration
2. เพื่อเพิ่มความสามารถของเซนเซอร์โนดให้สามารถปรับเปลี่ยนวงจร ประมวลผลในระดับฮาร์ดแวร์ได้ และสามารถโปรแกรมวงจรใหม่ผ่านระบบเครือข่ายเซนเซอร์ไร้สาย

1.4 ขอบเขต

1. ออกแบบและพัฒนาการทำ Partial reconfiguration บนเอฟพีจีเอ และใช้ซอฟต์แวร์ของบริษัท Xilinx เป็นหลักในขั้นตอนการออกแบบ
2. การทดสอบการทำงานของระบบใช้วงจรอย่างง่าย เช่น Full adder, AES ขนาดคีย์ 128 และ 192 บิต
3. การประยุกต์ใช้กับเครือข่ายเซนเซอร์ไร้สาย ใช้มาตรฐาน Zigbee รับส่งข้อมูลทั้งหมด

1.5 ขั้นตอนและวิธีการดำเนินงาน

1. ศึกษาค้นคว้าข้อมูลที่เกี่ยวข้องกับวิทยานิพนธ์
   - คุณสมบัติและลักษณะโครงสร้างภายในของเอฟพีจีเอ
   - กระบวนการ และขั้นตอนการทำ Partial reconfiguration
   - เครื่องมือ (Tools) ที่เกี่ยวข้องและรองรับการทำ Partial reconfiguration
   - เทคนิคการทำ Partial reconfiguration บนเอฟพีจีเอ แต่ละรุ่น
   - เซนเซอร์โนดและวิธีการรับส่งข้อมูลผ่านระบบเครือข่ายไร้สาย
2. ออกแบบ พัฒนา การทำ Partial reconfiguration บนเอฟพีจีเอ Spartan-6
   - วิเคราะห์ลักษณะโครงสร้างเอฟพีจีเอ ที่เหมาะสมกับการทำ Partial reconfiguration
   - การออกแบบขั้นตอนการทำ Partial reconfiguration โดยใช้ Tools GoAhead ร่วมกับ Xilinx ISE
   - ออกแบบและพัฒนาวงจรสำหรับทดสอบหลังจากทำให้เอฟพีจีเอสามารถทำงานแบบ Partial reconfiguration
   - พัฒนาวิธีการโปรแกรมวงจรลงเอฟพีจีเอ โดยระบบไร้สาย (Wireless)
3. การทดสอบและปรับปรุงการทำ Partial reconfiguration
   - ทดสอบคุณสมบัติการทำ Partial reconfiguration โดยโปรแกรมวงจรที่ต่างกัน
   - วิเคราะห์หาข้อผิดพลาดและปรับปรุงแก้ไข
   - ทดสอบการโปรแกรมวงจรลงเอฟพีจีเอ ผ่านระบบไร้สาย
   - ทดสอบระบบทั้งหมดหลังจากได้ปรับปรุงแก้ไข
4. จัดทำรายงานฉบับสมบูรณ์

1.6 โครงสร้างของรายงานวิทยานิพนธ์

ในบทที่ 2 จะกล่าวถึงทฤษฎีและหลักการที่เกี่ยวข้องกับงานวิจัย บทที่ 3 จะนำเสนอขั้นตอนและการออกแบบเพื่อทำ Partial reconfiguration บนเอฟพีจีเอ นำเสนอวงจรทดสอบและวิธีการโปแกรมวงจรลงเอฟพีจีเอ บทที่ 4 กล่าวถึงผลการทดสอบ พร้อมกับวิเคราะห์ผลลัพธ์ที่ได้จากการทดสอบ และในบทสุดท้ายจะกล่าวสรุปผลจากงานวิจัยพร้อมกับข้อเสนอแนะของงานวิจัย

บทที่ 2
ทฤษฎีและหลักการ

ในบทนี้กล่าวถึงทฤษฎีและหลักการที่เกี่ยวข้องกับงานวิจัยนี้ ซึ่งประกอบด้วยหัวข้อดังต่อไปนี้

2.1 สถาปัตยกรรมเอฟพีจีเอ Spartan-6

2.2 แนะนำการใช้งานเอฟพีจีเอแบบ Partial Reconfiguration

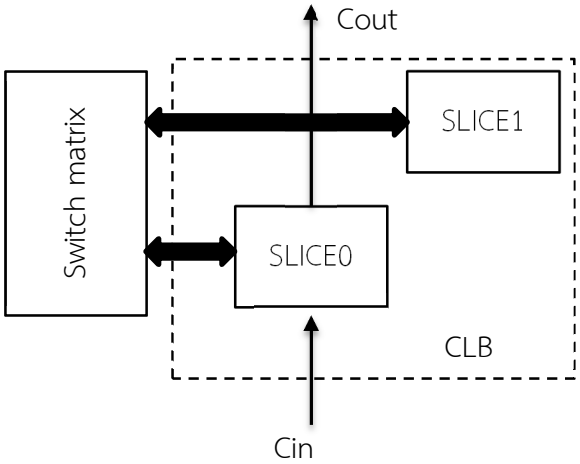
2.3 ระบบและโครงสร้างวงจรสำหรับ Partial Reconfiguration

2.4 กระบวนการทางซอฟต์แวร์สำหรับ Partial Reconfiguration

2.1 สถาปัตยกรรมเอฟพีจีเอ Spartan-6

Field-Programmable Gate Array (FPGA) เป็นอุปกรณ์ที่สามารถถูกโปรแกรมให้เป็นวงจรดิจิทัลได้ โดยที่โครงสร้างภายในเอฟพีจีเอประกอบไปด้วย Configurable Logic Blocks (CLBs) การสร้างวงจร หรือการโปรแกรมวงจรในเอฟพีจีเอนั้นหมายถึง การกำหนดค่าของ CLBs ให้ทำงานตามลอจิก และเชื่อมต่อแต่ละ CLBs ตามการออกแบบวงจรที่กำหนดไว้

Spartan-6 เป็นตระกูลเอฟพีจีเอขนาดกลางที่ผลิตโดย Xilinx มีจำนวน Logic cells ตั้งแต่ 4,000-150,000 หน่วยขึ้นอยู่กับรุ่นของเอฟพีจีเอ ภายใน CLBs 1 ชุด ประกอบไปด้วย 2 SLICE ดังภาพประกอบ 2-1 โดยจะเรียก SLICE ที่อยู่ล่างว่า SLICE0 และที่อยู่ถัดมาว่า SLICE1 ซึ่งระหว่าง 2 SLICE นี้ไม่ได้เชื่อมต่อกันโดยตรง แต่จะเชื่อมต่อกับ Switch matrix ทำหน้าที่เลือกอินพุตและเอาต์พุต ของ CLBs ชุดนั้น



ภาพประกอบ 2-1 โครงสร้าง Configurable Logic Blocks (CLBs)

คุณสมบัติของ SLICE คือ ภายในประกอบไปด้วย Look Up Table (LUT) และหน่วยความจำหรือ Flip-flop ลักษณะของ SLICE แบ่งเป็น 3 ประเภท ดังตารางที่ 2-1 ได้แก่ SLICEX เป็น SLICE มาตรฐาน และมีทั่วไปในเอฟพีจีเอ แบบที่สอง คือ SLICEL มีคุณสมบัติเหมือนกับ SLICEX แต่มีคุณสมบัติเพิ่ม คือ Wide multiplexer และ Carry logic ซึ่งทำให้สามารถเชื่อมต่อระหว่าง SLICE ในลักษณะแนวตั้งได้ และเสริมการทำงานของวงจรบวก หรือวงจรที่มีการส่งต่อสัญญาณระหว่างกัน และแบบที่สาม คือ SLICEM มีคุณสมบัติเหมือนกับ SLICEL แต่จะมีคุณสมบัติเพิ่ม คือ 64 Bits distributed RAM และ 32 Bits shift register

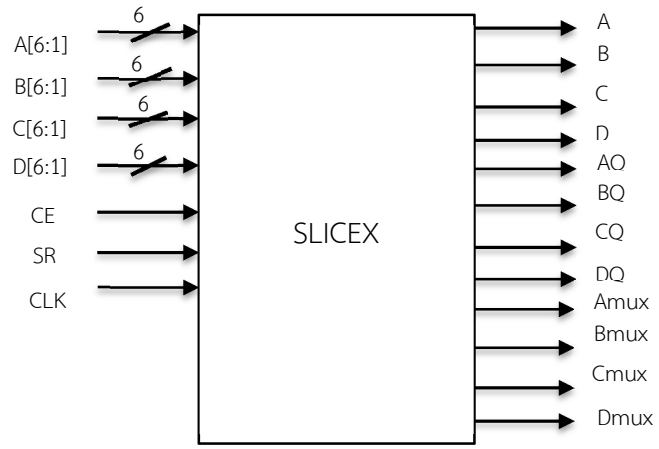
ตารางที่ 2-1 คุณสมบัติพื้นฐานของทรัพยากร SLICE ภายในเอฟพีจีเอ

คุณสมบัติ	**SLICEX**	**SLICEL**	**SLICEM**
6 Input LUTs	●	●	●
8 Flip-flops	●	●	●
Wide Multiplexers		●	●
Carry Logic		●	●
Distributed RAM			●
Shift Register			●

ภายใน CLBs 1 ชุดประกอบไปด้วย 2 SLICE โดยทุก CLBs มี SLICEX หนึ่งตัว และอีกตัวอาจเป็น SLICEL หรือ SLICEM ทำให้ทราบว่า ในเอฟพีจีเอนั้นจะประกอบไปด้วย SLICEX ประมาณ 50% ส่วน SLICEL และ SLICEM จะมีอย่างละ 25% และใน 1 SLICE ประกอบไปด้วย LUTs ที่มี 6 อินพุท จำนวน 4 ชุด และ Flip-flop 8 ชุด ดังนั้น CLBs 1 ชุด ประกอบไปด้วย SLICE 2 ชุด, LUTs 8 ชุด และ Flip-flops 16 ชุด นอกจากนั้นหากใน CLBs มี SLICEM จะทำให้ใน CLBs มีคุณสมบัติ RAM และ Shift register นอกจาก SLICE แล้ว ภายในเอฟพีจีเอมีโมดูลอื่น เช่น DSP48 เป็นวงจรคูณแล้วบวก หรือที่เรียกว่า Multiply Accumulator (MAC) ขนาด 18x18 ใช้สำหรับการประมวลผลสัญญาณ (Signal processing), Block RAM ขนาด 32 Bits และวงจรจัดการสัญญาณ Clock (Digital Clock Management : DCM)

ปัจจุบันขนาดของเอฟพีจีเอใช้หน่วยวัด คือ Logic cells ซึ่งเป็นหน่วยที่ได้จากการคำนวณระหว่าง จำนวน LUTs รวมกับ Flip flops โดยได้เป็นวิธีการคำนวณจำนวน Logic cell คือ Logic cell = 1.6 x LUTs เช่น Spartan-6 LX 16 มี LUTs ทั้งหมด คือ 9,112 จะมี Logic cell = 1.6x9,112 = 14,579 เป็นต้น

SLICE 1 ชุดประกอบไปด้วย LUTs ทั้งหมด 4 ชุด ใช้สัญลักษณ์ A, B, C, D ภาพประกอบ 2-2 คือ บล็อกไดอะแกรมอินพุตและเอาต์พุตของ SLICE 1 ชุด โดยในภาพ คือ SLICEX เนื่องจาก SLICE ทั้ง 3 แบบ มีอินพุตและเอาต์พุตหลักเหมือนกัน แต่จะมีเพิ่มเติมสำหรับ SLICEM คืออินพุตสำหรับการอ่านเขียนข้อมูลลงในหน่วยความจำ Distributed RAM



ภาพประกอบ 2-2 ไดอะแกรมอินพุต-เอาต์พุตของ SLICEX

หน้าที่และคุณสมบัติของอินพุตเอาต์พุตของ SLICEX แสดงดังตารางที่ 2-2

ตารางที่ 2-2 อธิบายสัญญาณอินพุตและเอาต์พุตของ SLICEX

ชื่อสัญญาณ	หน้าที่	คำอธิบาย
A[6:1], B[6:1], C[6:1], D[6:1]	อินพุต	สัญญาณอินพุตของ LUTs A, B, C, D
CE	อินพุต	Clock enable ทำหน้าที่ enable สัญญาณ clock ให้กับ flip-flop ซึ่งแต่ละ flip-flop นั้นใช้สัญญาณเส้นเดียวกัน
SR	อินพุต	Set/ Reset ทำหน้าที่ set และ reset flip-flop โดยที่ flip-flop ทุกตัวใช้สัญญาณเส้นเดียวกัน
CLK	อินพุต	สัญญาณ clock ต่อกับ flip-flop
A, B, C, D	เอาต์พุต	เอาต์พุตจาก LUTs A, B, C, D ตามลำดับ
AQ, BQ, CQ, DQ	เอาต์พุต	เอาต์พุตจาก flip-flop ของ LUTs A, B, C, D ตามลำดับ
Amux, Bmux, Cmux, Dmux	เอาต์พุต	เอาต์พุตจาก Multiplex ซึ่งเลือกระหว่างเอาต์พุตจาก flip-flop กับเอาต์พุตจาก LUTs

2.2 แนะนำการใช้งานเอฟพีจีเอแบบ Partial Reconfiguration

Reconfigurable computing คือ ระบบการประมวลผลที่สามารถแก้ไขปรับเปลี่ยน ขั้นตอนการทำงาน หรือปรับเปลี่ยนวงจร เพื่อให้สามารถประมวลผลให้มีประสิทธิภาพดีขึ้น การทำ Reconfigurable computing ต้องอาศัยอุปกรณ์ประเภท Programmable devices เช่น เอฟพีจีเอ โดยการทำ Reconfiguration บนเอฟพีจีเอ คือ การโปรแกรมโครงสร้างของวงจรดิจิตอลลงในเอฟพีจีเอ การโปรแกรมวงจรลงในเอฟพีจีเอโดยวิธีการทั่วไปเรียกว่า Full reconfiguration และการโปรแกรมโครงสร้างฮาร์ดแวร์โดยวิธีการโปรแกรมเพียงบางส่วนเรียกว่า Partial reconfiguration

การโปรแกรมโครงสร้างวงจรภายในเอฟพีจีเอโดยวิธีทั่วไปนั้นเป็นแบบ Full reconfiguration หมายถึง เมื่อต้องการโปรแกรมวงจรที่ออกแบบ เอฟพีจีเอจะทำการรีเซตวงจรเดิมที่อยู่ภายในทั้งหมด แล้วสร้างวงจรขึ้นมาใหม่ตามที่ได้โปรแกรมลงไป แต่เนื่องจากเอฟพีจีเอบางตระกูลนั้นมีคุณสมบัติที่สามารถโปรแกรมวงจรใหม่ได้โดยไม่จำเป็นต้องรีเซตวงจรก่อนหน้าทั้งหมด และขณะเดียวกันวงจรเหล่านั้นยังคงทำงานได้ตามปกติ เรียกการโปรแกรมแบบนี้ว่า Partial reconfiguration

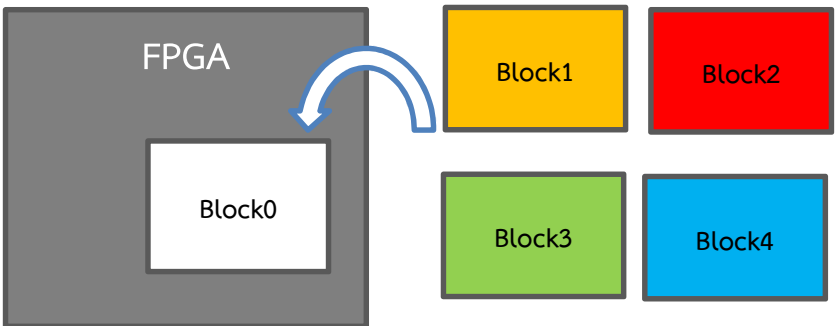
Partial reconfiguration เป็นเทคโนโลยีในการโปรแกรมโครงสร้างของวงจรภายในเอฟพีจีเอ โดยสามารถเลือกโปรแกรมวงจรลงบนพื้นที่ของเอฟพีจีเอเพียงบางพื้นที่ได้ ในขณะที่วงจรพื้นที่ส่วนอื่นของเอฟพีจีเอยังคงทำงานอยู่โดยไม่ได้รับผลกระทบจากการโปรแกรม ทำให้เอฟพีจีเอมีความยืดหยุ่นกับการใช้งานที่ต้องการปรับเปลี่ยนการทำงานบางส่วนโดยที่งานอีกส่วนไม่สามารถหยุดการทำงานได้ ข้อดีของการใช้เอฟพีจีเอแบบ Partial reconfiguration คือ

1. ลดขนาดพื้นที่วงจรเพราะ ไม่จำเป็นต้องโปรแกรมวงจรทั้งหมดลงไปแต่จะใช้วิธีการโปรแกรมวงจรส่วนนั้นเมื่อต้องการใช้งานเท่านั้น เช่น การทำงาน 1 งาน ต้องใช้รูปแบบวงจรแตกต่างกัน 5 แบบ ในเวลาที่ต่างกัน ดังนั้นแทนที่จะต้องโปรแกรมวงจรทั้ง 5 แบบ ก็จะเหลือเพียงแค่ 1 แบบ แล้วใช้วิธีการโปรแกรมวงจรใหม่ซ้ำลงในพื้นที่เดิมให้สามารถทำงานในแบบอื่นตามเวลาที่ต้องการ นั่นคือ จากเดิมที่ต้องใช้พื้นที่ 5 ส่วน จะลดลงเหลือเพียง 1 ส่วนเท่านั้น

2. เพิ่มความยืดหยุ่น (Flexibility) ของระบบฮาร์ดแวร์ในขณะทำงาน เพราะคุณสมบัติการโปรแกรมวงจรซ้ำในขณะทำงานได้ตลอดเวลา ทำให้สามารถปรับเปลี่ยนฮาร์ดแวร์ของระบบได้ในขณะทำงานโดยไม่ต้องมีการรีเซตระบบทั้งหมดใหม่ เช่น ระบบที่มีอุปกรณ์หลายชนิดเชื่อมต่อเข้าด้วยกัน สามารถสร้างฮาร์ดแวร์รูปแบบการสื่อสารหรือโพรโทคอลกับอุปกรณ์แต่ละแบบได้ตามการใช้งานอุปกรณ์ขณะนั้น โดยไม่จำเป็นต้องหยุดการทำงานของระบบเพื่อสร้างฮาร์ดแวร์เชื่อมต่อกับอุปกรณ์นั้นใหม่, การประมวลผลที่ต้องปรับเปลี่ยนอัลกอริทึมของวงจรตามอินพุต หรือตามสภาพแวดล้อมในขณะใช้งาน สามารถสร้างอัลกอริทึมการทำงานได้ตามความเหมาะสมกับสภาพแวดล้อมในขณะนั้น

3. ระยะเวลาการโปรแกรมสร้างวงจร (Configuration time) เนื่องจากการทำ Partial reconfiguration สามารถลดพื้นที่ของวงจรได้ดังที่กล่าวถึงในข้อ 1 ส่งผลให้ระยะเวลาที่ใช้โปรแกรมลงเอฟพีจีเอมีระยะเวลาน้อยกว่ามาก

4. ลดการใช้พลังงานได้ โดยปกติการสูญเสียพลังงานของเอฟพีจีเอจะประกอบไปด้วยพลังงานที่ต้องใช้งานแน่นอน (Static power) และพลังงานที่ขึ้นอยู่กับการทำงานนั้น (Dynamic power) ในส่วนของพลังงานที่ใช้งานแน่นอน (Static power) สามารถลดลงได้เนื่องจากเมื่อขนาดพื้นที่วงจรเล็กลงทำให้สามารถเลือกใช้เอฟพีจีเอที่มีขนาดเล็กลงซึ่งใช้พลังงานน้อยกว่า สำหรับพลังงานที่ขึ้นอยู่กับการใช้งาน (Dynamic power) สามารถลดลงได้ เช่น กรณีที่ต้องใช้เอฟพีจีเอทำงานซับซ้อนแต่ไม่ได้ทำงานนั้นตลอดเวลา แทนที่จะต้องโปรแกรมให้เอฟพีจีเอใช้ประสิทธิภาพสูงตลอดเวลาซึ่งใช้พลังงานมาก ก็จะใช้การโปรแกรมใหม่ให้ทำงานประสิทธิภาพสูงเฉพาะเวลาที่ต้องใช้งานเท่านั้น เมื่อไม่ใช้งานก็ให้เอฟพีจีเอทำงานที่ความเร็วต่ำลงซึ่งเป็นการลดการใช้พลังงานลง



ภาพประกอบ 2-3 ลักษณะการทำงานแบบ Partial Reconfiguration

การโปรแกรมแบบ Partial reconfiguration จะมองโครงสร้างของเอฟพีจีเอโดยการแบ่งรูปแบบพื้นที่ภายในเป็น 2 ส่วน คือ 1) Static part คือ พื้นที่ในเอฟพีจีเอที่ถูกโปรแกรมวงจรในครั้งแรกแล้ว เมื่อเริ่มทำงานจะไม่ถูกโปรแกรมซ้ำ และ 2) Reconfigurable part คือ พื้นที่ในเอฟพีจีเอที่สามารถโปรแกรมใหม่ได้ในขณะที่เอฟพีจีเอกำลังทำงาน ซึ่งการโปรแกรมแบบ Partial reconfiguration พื้นที่สองส่วนนี้จะทำงานด้วยกัน กล่าวคือ เมื่อต้องการโปรแกรมวงจรในส่วนของ Reconfigurable part เอฟพีจีเอจะรีเซตวงจรเฉพาะพื้นที่ Reconfigurable part เท่านั้น แล้วโปรแกรมวงจรใหม่ลงไปโดยที่วงจรส่วนของ Static part ไม่ต้องรีเซทการทำงาน

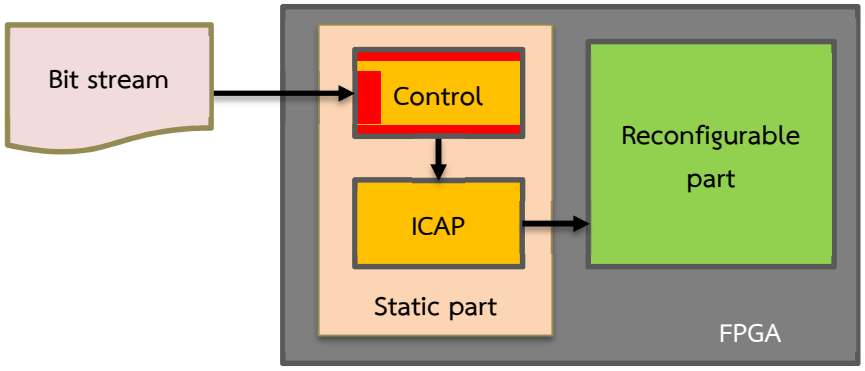
ภาพประกอบ 2-3 ฝั่งซ้ายของภาพ คือ พื้นที่ของเอฟพีจีเอทั้งหมดโดยที่สีเทา คือ พื้นที่ Static part และสีขาว คือ พื้นที่ Reconfigurable part สำหรับฝั่งขวาของภาพ คือ วงจรที่ออกแบบไว้ โดยเป็นวงจรที่แตกต่างกัน 4 แบบ และวงจรเหล่านี้สามารถโปรแกรมแบบ Partial reconfiguration ได้ ลำดับการทำงานของระบบ เริ่มจากโปรแกรมวงจรส่วนของ Static part ลงเอฟพีจีเอแล้วเริ่มการทำงาน จากนั้นผู้ใช้งานออกแบบวงจรที่ต้องการโดยกระบวนการออกแบบจะได้กล่าวในหัวข้อถัดไป เมื่อได้วงจรที่ต้องการแล้วจะโปรแกรมวงจรดังกล่าวลงเอฟพีจีเอ เมื่อต้องการปรับเปลี่ยนเป็นวงจรใหม่สามารถโปรแกรมวงจรใหม่ที่ออกแบบลงในพื้นที่ Reconfigurable part ได้ทันที โดยในขณะที่ทำการโปรแกรมวงจรใหม่ วงจรของพื้นที่ Static part ยังคงทำงานเดิมต่อไปโดยไม่หยุด หรือรีเซทการทำงาน

2.3 ระบบและโครงสร้างวงจรสำหรับ Partial Reconfiguration

กระบวนการในการโปรแกรมแบบ Partial reconfiguration มีความแตกต่างกับการโปรแกรมแบบปกติหรือ Full reconfiguration เนื่องจากการใช้งานแบบ Partial reconfiguration มีการแบ่งพื้นที่ออกเป็น 2 ส่วน ที่ทำงานต่างกัน จึงต้องพิจารณาว่าสามารถใช้งานส่วนใด และไม่สามารถใช้งานส่วนใดได้บ้าง การโปรแกรมวงจรในเอฟพีจีเอใช้ข้อมูลที่เรียกว่า Bit stream file ซึ่งได้มาจากกระบวนการและขั้นตอนการสร้างวงจร โดยที่ลำดับขั้นตอนการโปรแกรมวงจรในเอฟพีจีเอแบบปกติ เริ่มต้นการโปรแกรมเอฟพีจีเอจะเข้าสู่โหมดโปรแกรม (Configuration mode) เพื่อเตรียมทำการรีเซตตัวเองซึ่งจะลบข้อมูลวงจรเดิมที่มีอยู่ก่อนการโปรแกรมวงจรใหม่ จากนั้นเมื่อพร้อมที่จะโปรแกรมแล้ว จะโหลดข้อมูล Bitstream เข้าไป เสร็จแล้วจะเปลี่ยนตัวเองเป็นโหมดการใช้งานปกติ (User mode) เพื่อทำงานตามวงจรที่ได้โปรแกรม และเมื่อต้องการโปรแกรมวงจรลงในเอฟพีจีเอใหม่ ก็จะกลับมาเริ่มในขั้นตอนแรก คือ เข้าสู่โหมดโปรแกรม (Configuration mode) นั่นคือ จะต้องมีการรีเซต และลบข้อมูลวงจรเดิมก่อนเพื่อโปรแกรมวงจรใหม่เข้าไป

แต่ในส่วนของการโปรแกรมแบบ Partial reconfiguration นั้นแตกต่างกัน คือ จะสามารถโปรแกรมวงจรได้ในขณะที่เอฟพีจีเออยู่ในโหมดการใช้งานปกติ (User mode) แล้วเท่านั้น เนื่องจากการโปรแกรมแบบ Partial reconfiguration นั้นเป็นการโปรแกรมวงจรโดยใช้ทรัพยากรภายในเอฟพีจีเอด้วยกันเองซึ่งต่างกับการโปรแกรมแบบปกติที่ใช้อุปกรณ์จากภายนอกเป็นตัวโปรแกรมวงจร การโปรแกรมแบบ Partial reconfiguration ใช้ข้อมูล Bit stream แบบ Partial bit stream ซึ่งได้มาจากกระบวนการขั้นตอนการสร้างวงจรโดยที่มีความแตกต่างกันเล็กน้อยกับ Bit stream แบบปกติทั่วไป ลำดับการโปรแกรมวงจรในเอฟพีจีเอเริ่มต้นเหมือนกับการโปรแกรมแบบปกติ แต่เมื่อต้องการโปรแกรมวงจรลงในเอฟพีจีเอใหม่จะไม่กลับไปสู่โหมดโปรแกรม (Configuration mode) ทำให้สามารถโปรแกรมวงจรใหม่ได้โดยไม่ต้องมีการรีเซทตัวเองหรือลบข้อมูลวงจรเดิมทั้งหมดก่อนการโปรแกรม

ภาพประกอบ 2-4 เป็นตัวอย่างการออกแบบระบบโครงสร้างโมดูลวงจรภายในเอฟพีจีเอ ที่ทำให้เอฟพีจีเอสามารถโปรแกรมแบบ Partial reconfiguration โดยจะเห็นว่าภายในเอฟพีจีเอจะถูกแบ่งพื้นที่ออกเป็น 2 ส่วน คือ พื้นที่ Static part และ Reconfigurable part ในส่วนของ Reconfigurable part จะเป็นพื้นที่ไว้สำหรับวงจรที่ต้องการใช้งานและสามารถโปรแกรมซ้ำได้ พื้นที่อีกส่วน คือ Static part จะเป็นพื้นที่ไว้สำหรับวงจรที่ใช้ควบคุมการโปรแกรมวงจรของพื้นที่ Reconfigurable part รวมทั้งสามารถวางวงจรที่ต้องการให้ทำงานอื่น ๆ โดยที่พื้นที่ส่วนนี้จะถูกโปรแกรมลงเอฟพีจีเอเฉพาะครั้งแรก เมื่อระบบทำงานพื้นที่ส่วนนี้จะไม่ถูกโปรแกรมซ้ำอีก โดยพื้นที่ Static part ประกอบไปด้วยส่วนสำคัญ 2 ส่วน คือ



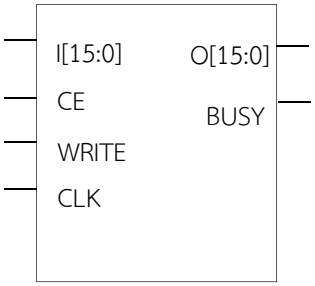
ภาพประกอบ 2-4 การออกแบบโมดูลวงจรเพื่อใช้งานเอฟพีจีเอแบบ Partial Reconfiguration

1. Control หรือส่วนควบคุม ทำหน้าที่จัดการเชื่อมต่อกับอุปกรณ์ภายนอกเพื่อรับข้อมูลไฟล์วงจร (Bitstream file) โดยเป็นไฟล์วงจรของพื้นที่ Reconfigurable part เท่านั้น และทำหน้าที่ควบคุมการดาวโหลวงจรลงในพื้นที่ Reconfigurable part ในการใช้งานจริงนั้นสามารถออกแบบส่วนนี้ได้ 2 รูปแบบ คือ ออกแบบโดยใช้โมดูลโพรเซสเซอร์ร่วมกับโมดูลเฉพาะกล่าวคือ ในเอฟพีจีเอ วิธีการโปรแกรมวงจรโดยใช้ทรัพยากรภายในเอฟพีจีเอ ผู้ผลิตเอฟพีจีเอเองได้ออกแบบให้มีโมดูลโพรเซสเซอร์ และโมดูลเฉพาะที่เป็นลักษณะ Intellectual Property core (IP core) จากผู้ผลิตให้ใช้งาน โดยโมดูลเฉพาะทำหน้าที่ควบคุมการโปรแกรมวงจรลงในพื้นที่ Reconfigurable part โดยต้องควบคุมโมดูลเฉพาะนี้ผ่านทางโมดูลโพรเซสเซอร์อีกชั้นหนึ่ง หรือรูปแบบที่สอง คือ ผู้ใช้งานทำการออกแบบส่วน Control โดยใช้รูปแบบวงจรเชิงลำดับ (State machine) ซึ่งหากใช้วิธีนี้ จะต้องออกแบบการควบคุมขั้นตอนการโปรแกรมวงจรลงในพื้นที่ Reconfigurable part ด้วยตัวผู้ใช้งานเอง โดยที่ทั้งสองวิธีนี้มีข้อดีและข้อเสียแตกต่างกัน ข้อดีของการใช้โมดูลโพรเซสเซอร์ร่วมกับโมดูลเฉพาะ คือ ใช้งานง่ายกว่า และมีความยืดหยุ่นกับการเชื่อมต่อกับอุปกรณ์ภายนอกได้มากกว่า เนื่องจากมี IP core ให้ใช้งานร่วมกันหลายตัว เช่น UART, SPI, Ethernet และ CAN bus เป็นต้น ทำให้การใช้งานไม่ยุ่งยากแต่มีข้อเสีย คือ จะต้องใช้ทรัพยากรจำนวนมากทำให้พื้นที่ที่สามารถใช้งาน เพื่อเป็นวงจรอื่นเหลือน้อยลง สำหรับการออกแบบโดยใช้ State machine มีข้อดีกว่าเรื่องของการใช้ทรัพยากรเพราะ จะออกแบบมาให้ใช้สำหรับเฉพาะงานโปรแกรมวงจรเท่านั้น ทำให้สามารถลดทรัพยากรในส่วนที่ไม่จำเป็นได้ แต่มีข้อจำกัด คือ ไม่มีความยืดหยุ่นในการใช้งานเพราะ ถูกจำกัดมาให้ทำงานในลักษณะเฉพาะเท่านั้น หากต้องการเพิ่มหรือปรับคุณสมบัติอื่นจะต้องแก้ไข และโปรแกรมวงจรใหม่

2. Internal Configuration Access Port (ICAP) คือ โมดูลเฉพาะที่มีอยู่ในเอฟพีจีเอบางตระกูลของผู้ผลิต Xilinx ทำหน้าที่เข้าถึงหน่วยความจำโปรแกรม (Configuration memory) ภายในเอฟพีจีเอซึ่งเป็นหน่วยความจำที่เก็บข้อมูลวงจรไว้ทำให้สามารถใช้ ICAP เพื่อ แก้ไขวงจรภายในเอฟพีจีเอได้ โดยการแก้ไขข้อมูลในหน่วยความจำ การโปรแกรมวงจรในเอฟพีจีเอผ่านทาง ICAP จึงถูกนำมาประยุกต์ใช้กับการใช้งานโปรแกรมวงจรแบบ Partial reconfiguration กล่าวคือ ICAP จะทำหน้าที่เป็นวงจรภายในเอฟพีจีเอที่สามารถโปรแกรมวงจรลงในเอฟพีจีเอด้วยตัวเองได้ ซึ่งเป็นแนวคิดของการโปรแกรมแบบ Partial reconfiguration

โดยคุณสมบัติของ ICAP มีลักษณะเหมือนกับ SelectMAP [11] ซึ่งเป็นรูปแบบการโปรแกรมเอฟพีจีเอรูปแบบหนึ่งที่ใช้กันโดยทั่วไป แต่ ICAP จะมีฟังก์ชันการใช้งานที่น้อยกว่าเนื่องจาก ICAP สามารถใช้โปรแกรมวงจรได้เฉพาะพื้นที่ Reconfigurable part เท่านั้น

ในเอฟพีจีเอแต่ละตระกูลมีคุณสมบัติของ ICAP แตกต่างกัน เนื่องจากความถี่ของสัญญาณนาฬิกาเอฟพีจีเอทำให้ความเร็วที่ ICAP ทำงานได้แตกต่างกัน การโปรแกรมวงจรในเอฟพีจีเอผ่าน ICAP ต้องใช้วงจรควบคุมที่สร้างขึ้นภายในเอฟพีจีเอเท่านั้น ต่างกับโมดูล SelectMAP ที่มี Interface เชื่อมต่อกับอุปกรณ์ภายนอกโดยใช้ Protocol JTAG ทำให้สามารถโปรแกรมเอฟพีจีเอผ่านทางอุปกรณ์ภายนอกได้ ภาพประกอบ 2-5 คือ ไดอะแกรมโมดูล ICAP ประกอบไปด้วยอินพุตและเอาต์พุตทำหน้าที่รับส่งข้อมูล และอีกส่วนหนึ่งทำหน้าที่ควบคุมการรับส่งข้อมูล



ภาพประกอบ 2-5 ไดอะแกรมอินพุต-เอาต์พุตของโมดูล ICAP

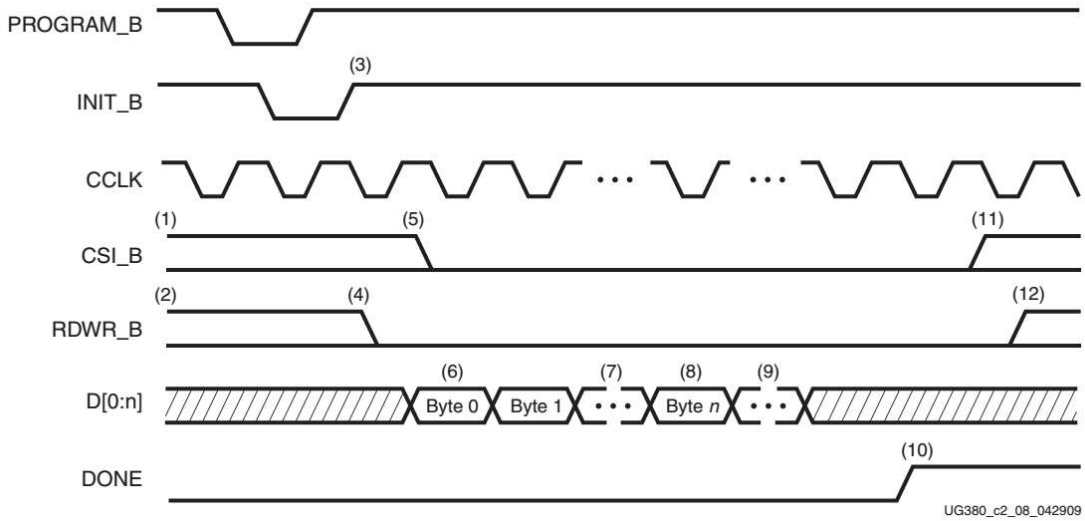
หน้าที่การทำงานของแต่ละสัญญาณได้อธิบายไว้ดังตารางที่ 2-3

ตารางที่ 2-3 หน้าที่การทำงานของอินพุตและเอาต์พุต โมดูล ICAP

สัญญาณ	ประเภท	หน้าที่การทำงาน
CLK	อินพุต	Clock อินพุตสำหรับการทำงานของ ICAP ซึ่งในเอฟพีจีเอ Spartan-6 ใช้ความเร็วไม่เกิน 20MHz
CE	อินพุต	Chip select enable คือ ให้ ICAP ทำงานเมื่อ Active low 0 = Enable, 1 = Disable
WRITE	อินพุต	ควบคุมการอ่าน / เขียนข้อมูล 1 = อ่านข้อมูล, 0 = เขียนข้อมูล
I[15:0]	อินพุต	อินพุตข้อมูลขนาด 16 บิต
O[15:0]	เอาต์พุต	เอาต์พุตข้อมูลขนาด 16 บิต
BUSY	เอาต์พุต	สถานะ การทำงานโดย Active high เมื่ออยู่ในสถานะ Busy

การทำงานของ ICAP มีลักษณะเหมือนกับ SelectMAP มีข้อแตกต่าง คือ ICAP แยกการทำงานของสัญญาณบัสขนาด 16 บิต เป็นบัสอินพุต (I) และบัสเอาต์พุต (O) ในขณะที่ SelectMAP ใช้บัสแบบรับส่งข้อมูลเส้นเดียวกัน (Bidirectional bus)

ดังนั้น Timing diagram การทำงานของ ICAP สามารถอ้างอิงจาก Timing diagram ของ SelectMAP ได้ การเขียนข้อมูล (Wrtite mode) มีลำดับการทำงาน ดังภาพประกอบ 2-6



ภาพประกอบ 2-6 Timing Diagram การทำงานของ SelectMAP **[11]**

การกำหนดชื่อสัญญาณระหว่าง SelectMAP และ ICAP แตกต่างกัน แต่สามารถเปรียบเทียบสัญญาณที่ทำหน้าที่เดียวกันระหว่าง SelectMAP กับ ICAP ได้ คือ

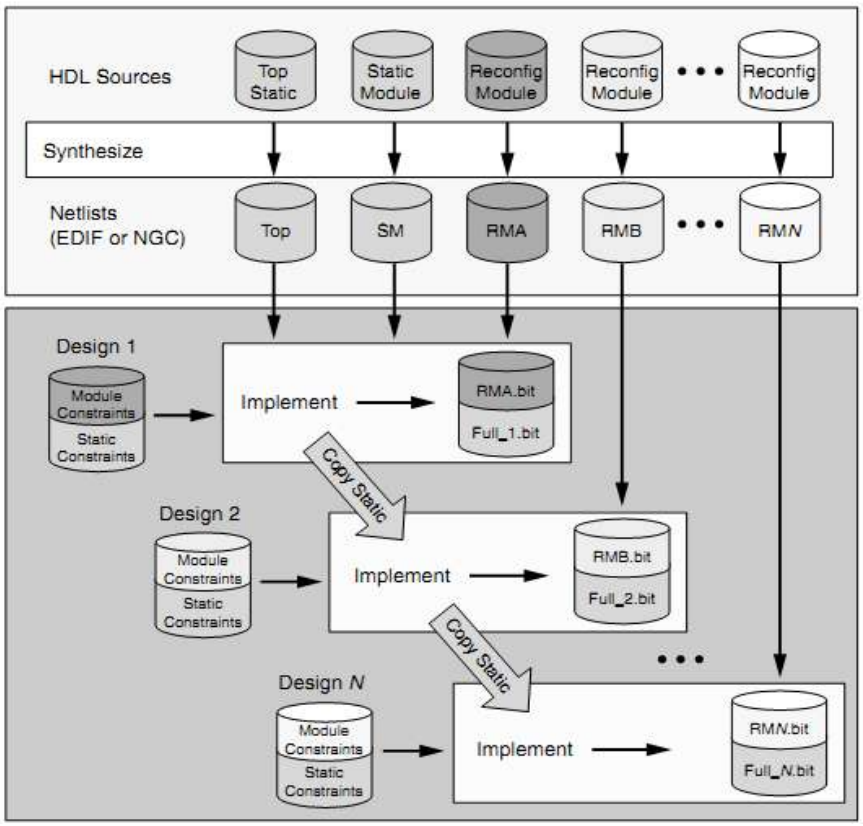
CSI_B	=	CE
RDWR_B	=	WRITE
D [0:n]	=	I [15:0] หรือ O [15:0]
DONE	=	BUSY

การทำงานเริ่มจากสัญญาณนาฬิกาแรก (1) สัญญาณ CSI_B (CE) สามารถกำหนดเป็น High คือ Disable การทำงานของโมดูล หรือกำหนดเป็น Low คือ Enable การทำงานของโมดูลได้ โดยไม่ส่งผลกระทบกับระบบเช่นเดียวกับสัญญาณ RDWR_B ในขั้นตอน (2) จากนั้นขั้นตอน (3) Toggle สัญญาณ INIT_B 1 clock cycle เพื่อเริ่มการทำงานโหมดเขียนข้อมูลหรือโปรแกรมวงจร (Write mode) ขั้นตอนถัดมา (4), (5) กำหนดสัญญาณ CSI_B เป็น Low คือ Enable การทำงานของโมดูล และ RDWR_B = Low เพื่อกำหนดให้บัส D [0:n] ทำหน้าที่เป็นอินพุตเตรียมรับข้อมูล จากนั้นในขั้นตอน (6-9) สัญญาณนาฬิกาถัดมาจะเริ่มส่งข้อมูลมายังบัส D [0:n] โดยส่งข้อมูลทุกขอบขาขึ้นของสัญญาณนาฬิกา เมื่อโหลดข้อมูลทั้งหมดแล้ว สัญญาณ DONE ซึ่งเป็นเอาต์พุตจะเป็น High ในขั้นตอน (10) เพื่อแสดงให้ทราบว่าได้โหลดข้อมูลครบแล้ว จากนั้นในขั้นตอน (11), (12) สามารถกำหนดสัญญาณ CSI_B = High เพื่อ Disable การทำงานของโมดูล และ RDWR_B = High เพื่อกำหนดให้บัส D[0:n] เป็นเอาต์พุต

2.4 กระบวนการทางซอฟต์แวร์สำหรับ Partial Reconfiguration

กระบวนการพัฒนาทางซอฟต์แวร์เพื่อใช้ออกแบบวงจรฮาร์ดแวร์ในเอฟพีจีเอให้มีการทำงานที่สามารถโปรแกรมแบบ Partial reconfiguration นั้นมีความแตกต่างกับการพัฒนาเอฟพีจีเอแบบปกติ (Full reconfiguration) เพียงเล็กน้อย ดังนั้นหลักการโดยทั่วไปจึงเหมือนกัน หัวข้อนี้จึงอธิบายกระบวนการของการออกแบบวงจรฮาร์ดแวร์โดยใช้ซอฟต์แวร์เพื่อโปรแกรมแบบ Partial reconfiguration โดยจะอธิบายในส่วนที่มีการเพิ่มเติมหรือแตกต่างจากขั้นตอนการโปรแกรมแบบ Full reconfiguration การออกแบบใช้ซอฟต์แวร์ Xilinx ISE เป็นซอฟต์แวร์หลัก

ขั้นตอนการออกแบบ (Software flow) แสดงดังภาพประกอบ 2-7 รายละเอียดแต่ละขั้นตอนมีดังนี้



ภาพประกอบ 2-7 ไดอะแกรมขั้นตอนการออกแบบโปรแกรมสำหรับ Partial Reconfiguration **[2]**

(1) HDL source คือ ขั้นตอนเริ่มต้นเป็นการออกแบบวงจรที่ต้องการโดยการเขียน Hardware Description Language (HDL) ด้วยภาษา Verilog หรือ VHDL ในการทำ Partial reconfiguration นั้นจะแบ่งการทำงานตามพื้นที่ คือ Static part และ Reconfigurable part ดังนั้นการออกแบบ HDL จะประกอบไปด้วย 3 ส่วน คือ Top module จะอยู่ใน Static part ทำหน้าที่เป็นโมดูลชั้นบนสุด, Static module คือ วงจรที่ออกแบบให้อยู่ใน Static part โดยสามารถออกแบบ Static module ได้หลายชุดแล้วเชื่อมต่อ Static module ย่อย ๆ ที่ออกแบบไว้เข้าด้วยกันอยู่ภายใน Top static ส่วนสุดท้าย คือ Reconfig module

เป็นวงจรที่ให้ทำงานบนพื้นที่ Reconfigurable part ซึ่งการเขียนโปรแกรมจะมีลักษณะเหมือนกับการเขียนโมดูลวงจรแบบปกติ

(2) Synthesize หรือการสังเคราะห์วงจรเป็นการนำข้อมูล HDL เปลี่ยนเป็นข้อมูลวงจรที่เอฟพีจีเอเข้าใจได้โดยจะได้ไฟล์แบบ Netlist เช่น EDIF, NGC เป็นต้น ในการทำ Synthesize จะแยกกันทำระหว่าง Static part กับ Reconfigurable part โดยสำหรับ Reconfigurable part สามารถแยกการทำ Systhesize แต่ละวงจรได้อย่างอิสระ นั่นคือ 1 โมดูลจะได้ไฟล์ netlist 1 ไฟล์

(3) Constraints คือ การกำหนดข้อจำกัดของวงจร โดยปกติแล้วจะสร้างไฟล์ User Constraints File (UCF) สำหรับข้อกำหนดของวงจร เช่น กำหนดอินพุตเอาต์พุตว่าให้เชื่อมต่อกับส่วนใดของเอฟพีจีเอ เป็นต้น ในการออกแบบ Partial reconfiguration มีการกำหนดข้อจำกัดที่แตกต่างไปจากการออกแบบปกติ โดยจะใส่ข้อจำกัดนี้ไว้ในส่วนการออกแบบ Static part เรียกว่า Static constraints โดยที่ในการออกแบบนั้นใช้ Static constrains แบบเดียวกัน ดังนั้นสามารถคัดลอก UCF file ของ Static part เพื่อใช้สำหรับการออกแบบ Reconfigurable part ที่แตกต่างกันได้ แต่สำหรับการออกแบบ Reconfigurable part จะกำหนดข้อจำกัดของแต่ละโมดูลแยกกัน การกำหนดข้อจำกัดของวงจรในการออกแบบ Partial reconfiguration มีข้อแตกต่างจากการโปรแกรมทั่วไปโดยมีการกำหนดข้อจำกัดเพิ่มเติมดังนี้

I. Area group constraints คือ การกำหนดพื้นที่เพื่อแบ่งระหว่าง Static part กับ Reconfigurable part โดยที่จะจำกัดพื้นที่ Static part ไม่ให้ใช้ทรัพยากรที่กำหนดให้เป็น Reconfigurable part เงื่อนไขและข้อกำหนดของการกำหนดพื้นที่มีดังนี้

- การกำหนดพื้นที่จะกำหนดเป็นพื้นที่สี่เหลี่ยมจากตำแหน่ง SLICE 2 จุด คือ มุมล่างซ้าย (Xmin, Ymin) และมุมบนขวา (Xmax, Ymax) (แนวทแยง) โดยในการกำหนด SLICE ที่เลือกต้องครอบคลุมทั้ง CLBs ไม่สามารถแยกกันได้ กล่าวคือ SLICE มุมล่างซ้าย Xmin จะสามารถเลือกได้เฉพาะ SLICE เลขคู่ เช่น Xmin = X20, Xmin = X2 เป็นต้น เพราะใน CLBs จะกำหนดให้ SLICE ซ้ายมือเป็นเลขคู่และขวามือเป็นเลขคี่ การเลือก SLICE ซ้ายมือเป็นเลขคู่ในมุมซ้ายล่างทำให้พื้นที่ Reconfigurable part ครอบคลุมทั้ง CLBs และเช่นเดียวกันในจุดมุมขวาบน Xmax จะกำหนดเป็น SLICE เลขคี่ ซึ่งเป็น SLICE ด้านขวาใน CLBs ทำให้พื้นที่ Reconfigurable part ครอบคลุมทั้ง CLBs
- มีทรัพยากรบางประเภทที่ต้องอยู่ในพื้นที่ Static part เท่านั้น เช่น Global clock logic, Digital Clock Manager (DCM), Phase Lock Loop (PLL) รวมถึงอินพุตเอาต์พุตจากภายนอก และทรัพยากรที่เกี่ยวข้องกับอินพุตเอาต์พุตภายนอกทั้งหมด

II. Partition pins คือ จุดเชื่อมต่อระหว่าง Static part และ Reconfigurable part เพื่อใช้เป็นจุดอ้างอิงในการเชื่อมต่อระหว่างวงจรในพื้นที่ Static part และ Reconfigurable part นอกจากนี้ Partition pins ยังสามารถทำหน้าที่เป็นอินพุตและเอาต์พุตของพื้นที่ Reconfigurable part เพื่อรับส่งข้อมูลกับพื้นที่ Static part

(4) Implementation ในขั้นตอนนี้ไม่มีความแตกต่างกับการสร้างวงจรในเอฟพีจีเอโดยวิธีทั่วไปที่ใช้ข้อมูลที่ผ่านกระบวนการ Synthesize มาแล้วเพื่อให้ได้ข้อมูล Native Circuit Description (NCD)

(5) Generating bit files เป็นขั้นตอนสุดท้ายของกระบวนการ ทำหน้าที่สร้างไฟล์วงจร (Bitstream) เพื่อใช้โปรแกรมลงเอฟพีจีเอ ขั้นตอนนี้ทั้ง Static part และ Reconfigurable part ใช้คำสั่งแบบเดียวกันแต่มีข้อคำสั่งที่เป็นการกำหนดเงื่อนไขเพิ่มเติมในส่วนของ Reconfigurable part มีคำสั่งดังนี้

I. -g ActiveReconfig: Yes เป็นคำสั่งให้ป้องกันการรีเซตตัวเองของเอฟพีจีเอ โดยปกติแล้วคำสั่งนี้จะไม่ถูกตั้งค่าเป็น Yes เนื่องจากปกติหากโปรแกรมไฟล์วงจรเอฟพีจีเอจะทำการรีเซตตัวเองเพื่อเคลียร์วงจรเดิมทั้งหมดก่อนจะสร้างวงจรใหม่ การที่ต้องเซตคำสั่งนี้ให้เป็น Yes เพื่อให้เมื่อโปรแกรมไฟล์วงจรในพื้นที่ Reconfigurable part แล้วเอฟพีจีเอจะไม่รีเซตตัวเอง

II. *-g Binary: Yes* เป็นคำสั่งในการสร้าง Binary Configuration File (BIN file) เนื่องจากไฟล์ Binary จะบันทึกเฉพาะข้อมูลวงจร (Configuration data) เพียงอย่างเดียว ซึ่งในไฟล์ Bitstream ก็มีข้อมูลวงจรเช่นเดียวกันแต่จะมีส่วนของข้อมูล Header ที่บอกถึงความยาวข้อมูล ซึ่งในการโปรแกรมแบบ Partial reconfiguration ไม่ต้องการส่วนนี้จึงใช้ไฟล์แบบ Binary (BIN)

เมื่อผ่านกระบวนการทั้งหมดแล้ว จะได้ไฟล์ Bitstream ของพื้นที่ Static part และไฟล์ Binary ของพื้นที่ Reconfigurable part ซึ่งในที่นี้จะเรียกว่า Partial bitstream จากนั้นสามารถโปรแกรมไฟล์วงจรลงเอฟพีจีเอเพื่อให้ทำงานแบบ Partial reconfiguration ได้

บทที่ 3
การออกแบบเอฟพีจีเอเพื่อโปรแกรมแบบ Partial Reconfiguration

หัวข้อนี้จะกล่าวถึงกระบวนการและขั้นตอนการออกแบบระบบทั้งหมด มีรายละเอียดดังหัวข้อต่อไปนี้

3.1 การออกแบบวงจรพื้นที่ Static Part

3.2 กระบวนการออกแบบไฟล์วงจรพื้นที่ Static Part

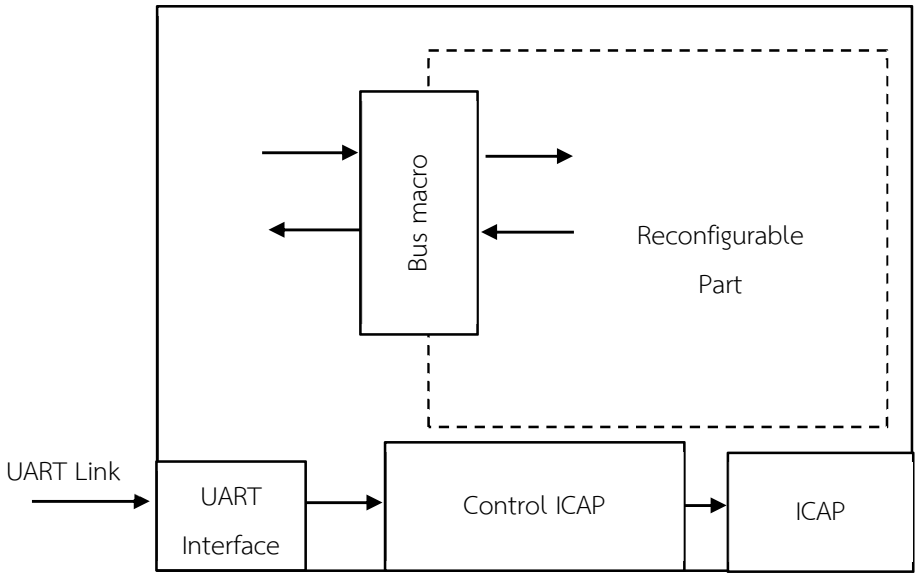
3.3 การออกแบบไมโครโพรเซสเซอร์ในพื้นที่ Static Part

3.4 กระบวนการออกแบบไฟล์วงจรพื้นที่ Reconfigurable Part

3.5 การออกแบบวิธีการ Reconfigure เอฟพีจีเอ

3.1 การออกแบบวงจรพื้นที่ Static Part

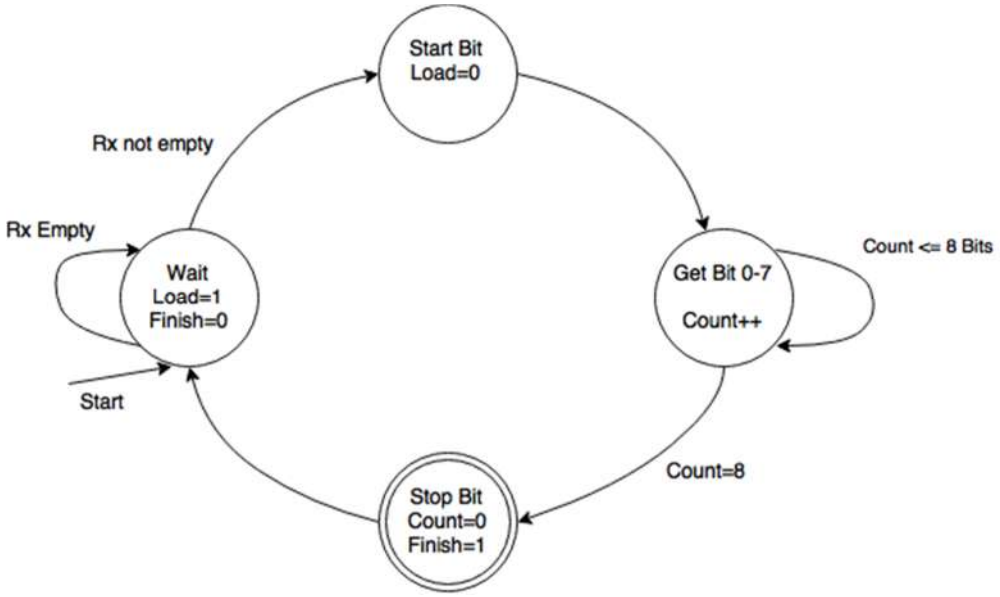
การออกแบบในเอฟพีจีเอเริ่มจากส่วนแรก คือ วางโครงสร้างของพื้นที่ Static part โดยงานวิจัยกำหนดให้ผู้ใช้สามารถเชื่อมต่อกับเอฟพีจีเอ เพื่อโปรแกรมวงจรของพื้นที่ Reconfigurable part ได้ทาง UART interface เนื่องจากเชื่อมต่อกับคอมพิวเตอร์ได้โดยตรง ไม่จำเป็นต้องมีวงจรเพิ่มเติม และเป็น Interface พื้นฐานของอุปกรณ์ฮาร์ดแวร์อื่น ๆ ซึ่งทำให้ไม่ยุ่งยากเมื่อนำไปเชื่อมต่อกับอุปกรณ์อื่น ๆ โครงสร้างโมดูลวงจรในส่วนของ Static part มีลักษณะไดอะแกรมดังภาพประกอบ 3-1 ลักษณะการทำงานของแต่ละส่วนมีดังนี้



ภาพประกอบ 3-1 การออกแบบวงจรสำหรับพื้นที่ Static Part

3.1.1 โมดูลวงจร UART Interface

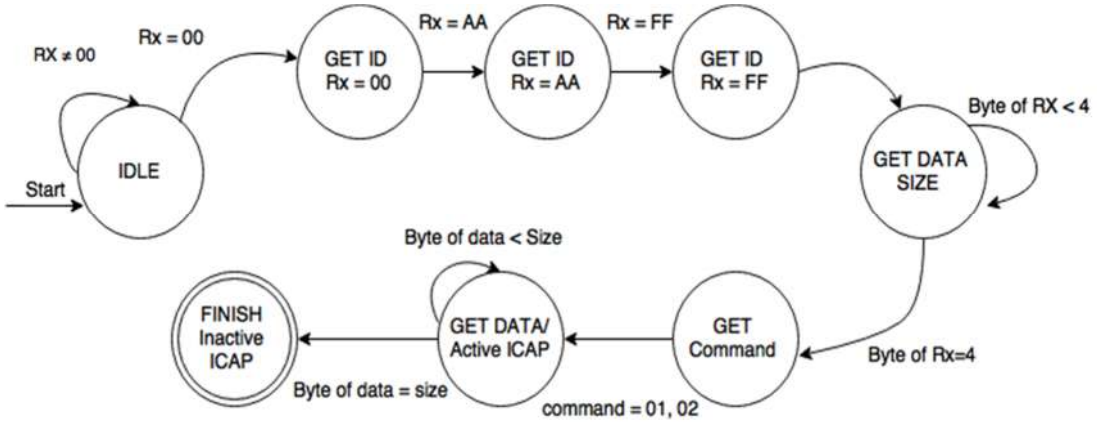
วงจร UART interface ทำหน้าที่ควบคุมการรับส่งข้อมูลระหว่างเอฟพีจีเอกับอุปกรณ์ภายนอกด้วยโพรโทคอล UART รับอินพุตจากสัญญาณ Rx และมีเอาต์พุตต่อกับโมดูล Control ICAP โดยข้อมูลที่รับ คือ ข้อมูล Partial bitstream ภายในเอฟพีจีเอนั้นไม่มี UART interface สำเร็จรูปมาให้ใช้งาน ดังนั้นผู้ใช้จำเป็นต้องออกแบบวงจรด้วยตัวเอง เพื่อทำการดึงค่าข้อมูลมาใช้งานโดย UART รับส่งข้อมูลทีละ 1 บิต ในอัตราความเร็วต่างๆ เช่น 9,600 bit per second (bps), 115,200 bps เหล่านี้เป็นต้น ผู้ใช้จำเป็นต้องรู้ว่าจะเชื่อมต่อกับอุปกรณ์ภายนอกที่อัตราเร็วเท่าใดเนื่องจาก UART ทำงานแบบ Asynchronous ทำให้ไม่มีสัญญาณใด ๆ มากำหนดความเร็วการทำงาน จำเป็นต้องกำหนดให้ฝั่งรับและฝั่งส่งมีการทำงานที่ความเร็วเท่ากัน ภาพประกอบ 3-2 แสดงให้เห็น State machine การทำงานของโมดูล UART interface เริ่มจากสถานะ Wait รอข้อมูลโดยตรวจสอบจาก Buffer Rx ว่ามีข้อมูลเข้ามาหรือไม่ หาก Buffer Rx มีข้อมูลจะส่งไปสถานะถัดไป คือ Start bit เพื่อเริ่มต้นรับค่าข้อมูลขนาด 8 บิต จากนั้นเข้าสู่สถานะ Get bit เพื่ออ่านค่าทีละบิต จำนวน 8 บิต หรือ 1 ไบท์ จากนั้นเข้าสถานะ Stop bit ซึ่งเป็นสถานะสุดท้ายของขั้นตอน นั่นคือ เมื่อจบสถานะ Stop bit ฝั่งรับจะได้ข้อมูลครบ 1 ไบท์ ซึ่งพร้อมที่จะนำข้อมูลไปใช้งานต่อไป จากนั้นจะเข้าสู่สถานะ Wait เพื่อรอข้อมูลใหม่อีกครั้ง



ภาพประกอบ 3-2 State Machine การทำงานของ UART Interface

3.1.2 โมดูลควบคุม ICAP (Control ICAP)

วงจร Control ICAP ทำหน้าที่ควบคุมโมดูล ICAP ในการโปรแกรมวงจรพื้นที่ Reconfigurable part การทำงานของ Control ICAP จะทำงานไปพร้อมกับ UART interface นั่นคือ เมื่อได้รับข้อมูลครบจาก UART แล้วจะทำส่วนของ ControlICAP ทันที ภาพประกอบ 3-3 แสดงให้เห็น State machine การทำงานของ Control ICAP เริ่มกระบวนการจากสถานะ Idle รอข้อมูลจาก UART เมื่อมีข้อมูลเข้ามาจะเข้าสู่สถานะ GET ID 00 ต่อด้วย GET ID AA และ GET ID FF ตามลำดับ ซึ่งเป็นการอ่านค่า Header ของไฟล์ Partial bitstream เพราะในไฟล์ Partial bitstream นั้นมีข้อมูล 3 ไบท์แรกเป็น Header คือ 0x00 AA FF จากนั้นเข้าสู่สถานะ GET SIZE เพื่ออ่านค่าขนาดของไฟล์ Partial bitstream เพราะข้อมูลไบท์ที่ 4-7 ในไฟล์ Partial bitstream จะบอกให้ทราบขนาดของไฟล์นั้น เมื่อทำการอ่านค่าครบ 32 บิต จะเข้าสู่สถานะ GET command เนื่องจากออกแบบให้สามารถโปรแกรมด้วยไฟล์รูปแบบ Binary และ HEX โดยกำหนดให้ command = 01 หากโปรแกรมโดยใช้ไฟล์ Binary และ command = 02 หากโปรแกรมโดยใช้ HEX



ภาพประกอบ 3-3 State Machine การทำงานของ Control ICAP

จากนั้นเข้าสู่สถานะ GET data เพื่อรับข้อมูลไฟล์โปรแกรมโดยเมื่อรับข้อมูลครบ 2 ไบท์ หรือ 16 บิต จะทำการเรียกใช้ ICAP (Activate ICAP) เพื่อให้ ICAP ทำงานโดยเข้าสู่สถานะ GET bit HIGH และ GET bit LOW คือ การนำข้อมูลซึ่งรับมาทีละ 8 บิต มาเรียงต่อกัน เพื่อให้ครบ 16 บิต แล้วส่งต่อไปให้โมดูล ICAP และเซต ICAP CE = 0 คือ คำสั่ง ICAP chip enable โดยทำงานแบบ active low เพื่อสั่งให้ ICAP เขียนข้อมูล Partial bitstream ที่ได้ลงในหน่วยความจำโปรแกรม (Configuration memory) จากนั้นสถานะ GET data ก็จะ active ICAP ทุก ๆ 16 บิต จนข้อมูลครบตามขนาดที่ได้อ่านไว้ก่อนหน้า เมื่อส่งข้อมูลให้ ICAP ครบแล้วเข้าสู่สถานะ Finish เพื่อหยุดการทำงาน ICAP (Deactivate ICAP) และกลับไปยังสถานะ IDLE เพื่อรอรับข้อมูลต่อไป

3.1.3 โมดูลวงจร Internal Configuration Access Port (ICAP)

ICAP คือ โมดูลวงจรทำหน้าที่โปรแกรมพื้นที่ Reconfigurable part อินพุตรับข้อมูลไฟล์โปรแกรม Partial Bitstream สำหรับในงานวิจัยใช้เอฟพีจีเอ Spartan-6 ดังนั้น ICAP ทำงานที่ความถี่สัญญาณนาฬิกา 20 MHz โมดูล ICAP ไม่สามารถทำงานโดยตัวเองได้ ดังนั้นการควบคุมการทำงานทั้งหมดจะถูกกำหนดโดย Control ICAP

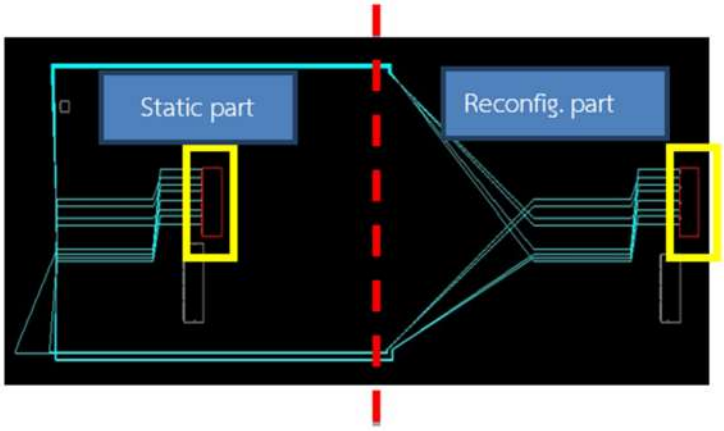
3.1.4 โมดูลวงจรการเชื่อมต่อระหว่างพื้นที่ (Bus Macro)

Bus macro คือ วงจรทำหน้าที่เชื่อมต่อ เพื่อรับส่งข้อมูลระหว่าง Static part และ Reconfigurable part โดยการสร้างจุดเชื่อมต่อ เพื่อเป็นอินพุตเอาต์พุตของพื้นที่นั้น กล่าวคือ พื้นที่ Static part มีอินพุตเอาต์พุตชุดหนึ่ง และพื้นที่ Reconfigurable part ก็จะมีอินพุตเอาต์พุตอีกชุดหนึ่งแล้วเชื่อมต่อระหว่างกันโดยให้เอาต์พุตจาก Static part เชื่อมต่อกับอินพุตของ Reconfigurable part และเอาต์พุตจาก Reconfigurable part เชื่อมต่อกับอินพุตของ Static part เสมือนกับเป็นเส้นทางสื่อสาร (Bus direction) เพื่อรับส่งข้อมูลระหว่างกัน ในการออกแบบบนเอฟพีจีเอใช้ทรัพยากร SLICE เพื่อกำหนดเป็นอินพุตเอาต์พุตพอร์ต โดยต้องระบุตำแหน่ง SLICE ที่ใช้เป็นอินพุตเอาต์พุตทั้งในพื้นที่ Static part และ Reconfigurable part หากไม่กำหนดจะทำให้ตำแหน่งการเชื่อมต่อไม่ถูกต้องเนื่องจาก หากทำการโปรแกรมวงจรลงใน Reconfigurable part ใหม่ นั่นคือ วงจรไม่เหมือนเดิม การวางตำแหน่งจะไม่ใช่ตำแหน่งเดิมส่งผลให้ตำแหน่งของอินพุตเอาต์พุตเปลี่ยนแปลงและไม่สามารถรับส่งข้อมูลได้

การกำหนดตำแหน่งและสร้างการเชื่อมต่อ (Place and route) ในลักษณะที่ต้องกำหนดตำแหน่ง SLICE แบบนี้ไม่สามารถใช้ซอฟต์แวร์ Xilinx ISE ทำกระบวนการ Place and route ได้โดยตรง แต่สามารถทำการแนบโมดูลในลักษณะที่เรียกว่า Macro เข้าไปได้ ดังนั้นจะเรียกโมดูลนี้ว่า Bus macro วิธีการสร้าง Bus macro ทำเมื่อได้กำหนดพื้นที่ Static part และ Reconfigurable part ไว้แล้ว โดยเลือก SLICE 2 ชุดเป็น 1 คู่ ให้ชุดแรกอยู่ใน Static part และอีกชุดอยู่ใน Reconfigurable part และแต่ละ SLICE ให้กำหนดคุณสมบัติตามการใช้งาน คือ เชื่อมต่อเอาต์พุตจาก Static part กับอินพุตของ Reconfigurable part ดังนั้นต้องกำหนด Output pin ใน LUTs ของ SLICE ใน Static part ให้ตรงกับ Input pin ใน LUTs ของ SLICE ใน Reconfigurable part และในทางกลับกัน เมื่อต้องการให้อินพุตจาก Static part เชื่อมต่อกับเอาต์พุตของ Reconfigurable part ต้องกำหนด Input pin ใน LUTs ของ SLICE ใน Static part ให้ตรงกับ Output pin ใน LUTs ของ SLICE ใน Reconfigurable part เพื่อทำการวางเส้นทางการเชื่อมต่อสัญญาณ (Routing) ระหว่างสอง SLICE ดังกล่าว

SLICE 1 ชุดประกอบไปด้วย 4 LUTs สามารถกำหนดได้ 4 อินพุตเอาต์พุต ดังนั้น Bus macro 1 ชุด จะได้สัญญาณรับส่งข้อมูลขนาด 4 บิต และจาก 1 CLBs ประกอบไปด้วย 2 SLICE ดังนั้น 1 CLBs จะสามารถสร้าง Bus macro ขนาด 8 บิต แบ่งเป็นอินพุต 4 บิต และเอาต์พุต 4 บิต ภาพประกอบ 3-4 แสดงให้เห็นลักษณะการเชื่อมต่อของสัญญาณระหว่าง CLBs 2 ชุด โดยที่ CLBs ซ้ายมืออยู่ในพื้นที่ Static part และ CLB ขวามืออยู่ในพื้นที่ Reconfigurable part

นอกจากนี้ การเลือกตำแหน่ง CLBs ให้เลือก CLBs บริเวณขอบของพื้นที่ Reconfigurable part เพื่อให้เหลือพื้นที่สำหรับการใช้งานสร้างวงจรได้มากที่สุด

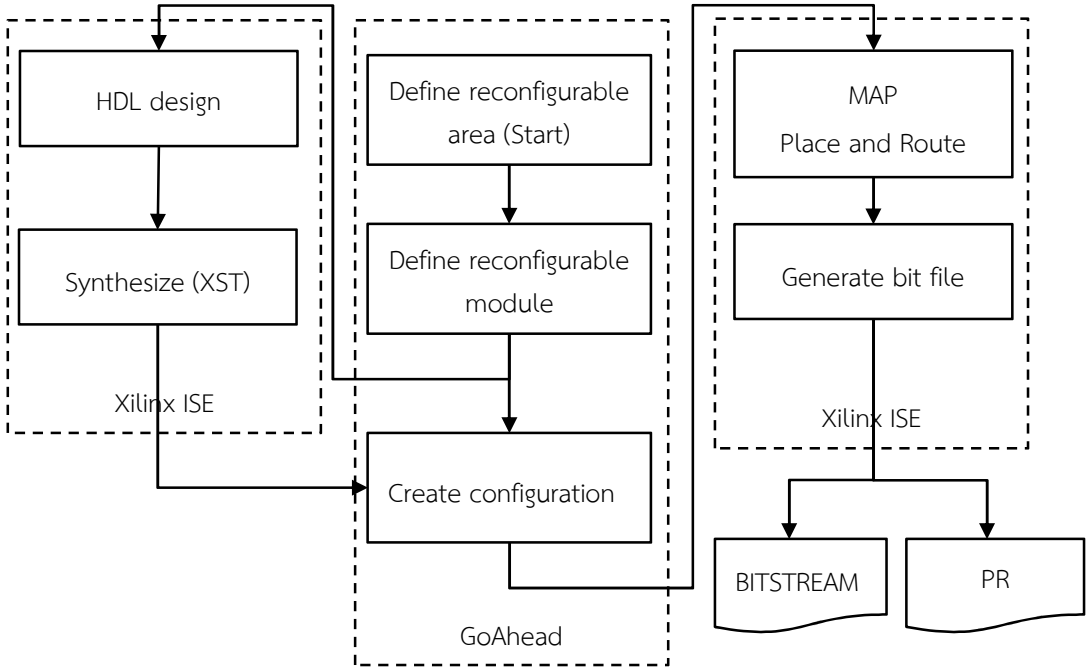


ภาพประกอบ 3-4 Bus Macro ขนาด 4 บิต

โครงสร้างวงจรทั้งหมดที่ออกแบบในพื้นที่ Static part ที่ได้กล่าวมานั้น คือ วงจรที่จำเป็นต้องมี เพื่อให้ใช้โปรแกรม Partial reconfiguration ได้ ซึ่งในการใช้งานจริง ผู้ใช้สามารถเพิ่มเติมวงจรที่ต้องการให้ทำงานในพื้นที่ Static part นอกเหนือจากนี้ได้ โดยวงจรที่เพิ่มเติมนั้นจะไม่มีผลต่อการทำงานของวงจรที่ออกแบบไว้

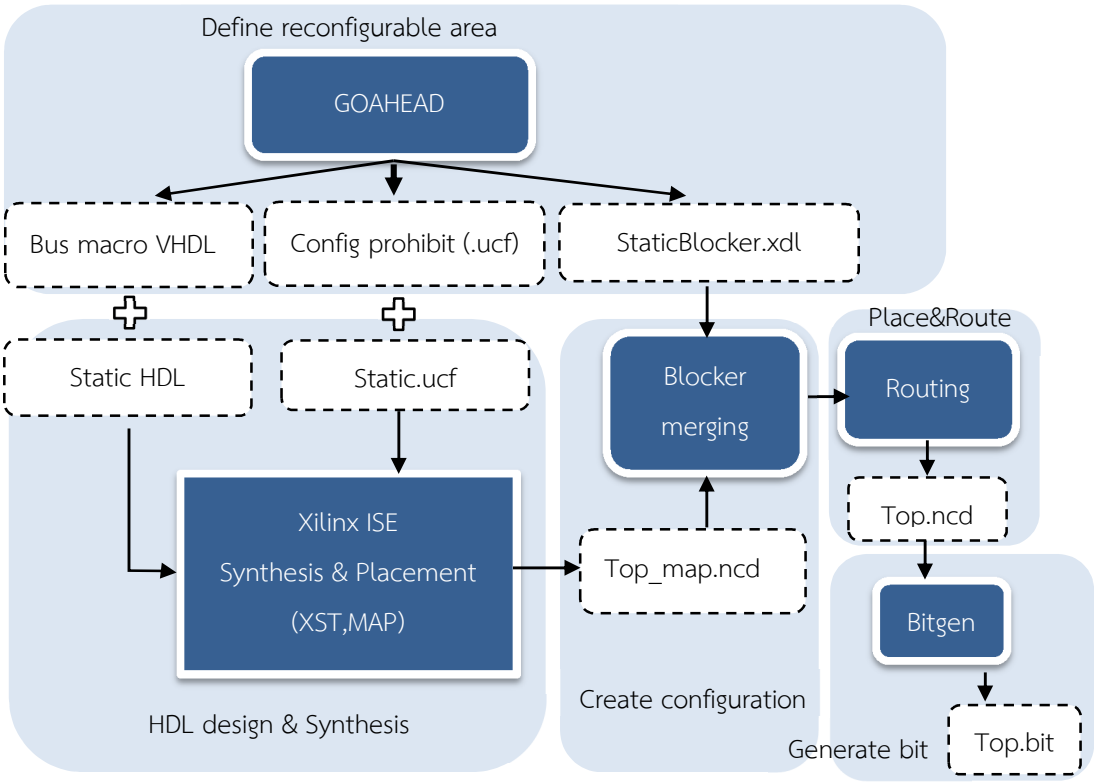
3.2 กระบวนการออกแบบไฟล์วงจรพื้นที่ Static Part

หัวข้อนี้กล่าวถึงกระบวนการและขั้นตอนทางซอฟต์แวร์ในการออกแบบวงจรของพื้นที่ Static part งานวิจัยใช้ซอฟต์แวร์ GoAhead [20] ร่วมกับซอฟต์แวร์ Xilinx ISE ภาพรวมของลำดับขั้นตอนการออกแบบแสดงดังภาพประกอบ 3-5 เริ่มต้นด้วยกระบวนการกำหนดพื้นที่ (Define reconfigurable area) ในเอฟพีจีเอระหว่างพื้นที่ Static part และ Reconfigurable part จากนั้นทำกระบวนการกำหนดประเภทวงจร (Define reconfigurable module) โดยกำหนดตามพื้นที่ที่จะให้วงจรทำงาน แบ่งเป็นวงจรในพื้นที่ Static part และวงจรในพื้นที่ Reconfigurable part ขั้นตอนถัดมา คือ การออกแบบวงจร (HDL design) โดยซอฟต์แวร์ Xilinx จากนั้นผ่านกระบวนการสังเคราะห์วงจร (Synthesize) ข้อมูลที่ได้ถูกนำมาเข้ากระบวนการกำหนดคุณสมบัติวงจรของแต่ละพื้นที่ (Create configuration) เช่น คุณสมบัติการวางตำแหน่งวงจร ทรัพยากรวงจรของพื้นที่ เนื่องจากพื้นที่ Static part และ Reconfigurable part มีรูปแบบและคุณสมบัติการทำงานต่างกัน จากนั้นเข้าสู่กระบวนการวางวงจรและกำหนดเส้นทางวงจรลงเอฟพีจีเอ (MAP, Place and Route) เพื่อนำไปสู่กระบวนการสร้างไฟล์วงจร (Generate bit file) แบ่งเป็นไฟล์ Full bitstream คือ ไฟล์วงจรพื้นที่ Static part และ Partial (PR) bitstream คือ ไฟล์วงจรพื้นที่ Reconfigurable part



ภาพประกอบ 3-5 ขั้นตอนการออกแบบวงจรโดยซอฟต์แวร์ Xilinx, GoAhead

การนำกระบวนการและขั้นตอนออกแบบวงจรมาใช้งานจริงโดยใช้ซอฟต์แวร์ GoAhead และ Xilinx ISE มีกระบวนการดังภาพประกอบ 3-6 โดยนำเสนอขั้นตอนพร้อมกับเปรียบเทียบให้เห็นว่าแต่ละกระบวนการ คือ กระบวนการออกแบบวงจรส่วนใด



ภาพประกอบ 3-6 กระบวนการออกแบบทางซอฟต์แวร์ของพื้นที่ Static Part

กระบวนการทำงานมี 5 ขั้นตอนหลักซิงไซ์ซอฟต์แวร์ Xilinx ISE และ GoAhead รายละเอียดมีดังนี้

(1) GoAhead Process ทำหน้าที่กำหนดพื้นที่ภายในเอฟพีจีเอ โดยให้ส่วนหนึ่งเป็น Reconfigurable part ซอฟต์แวร์ GoAhead สามารถสร้างไฟล์ Config Prohibit.ucf เป็นไฟล์ที่มีข้อมูล SLICE ที่ถูกจำกัดไม่ให้พื้นที่ Static part ใช้งานได้ เพื่อเก็บไว้ใช้งานสำหรับพื้นที่ Reconfigurable part เท่านั้น และซอฟต์แวร์ GoAhead สามารถสร้าง Bus macro VHDL template ที่เป็นโปรแกรมภาษา VHDL อธิบายโครงสร้าง Bus macro และผู้ใช้สามารถกำหนดขนาด Data bus ของ Bus macro ได้ จากนั้นสร้างไฟล์ StaticBlocker.xdl เป็นไฟล์ที่มีข้อมูลระบุว่า Static part สามารถใช้ทรัพยากรส่วนใดได้บ้างในเอฟพีจีเอ เพื่อใช้สำหรับขั้นตอนการกำหนดเส้นทางการเชื่อมต่อสัญญาณ (Routing) ในครั้งสุดท้าย

(2) Xilinx ISE Process Synthesis and Placement คือ กระบวนการที่ทำหน้าที่ออกแบบวงจรที่จะให้ทำงานในพื้นที่ Static part โดยใช้ซอฟต์แวร์ Xilinx ISE เขียนโค้ดโปรแกรมภาษา VHDL เพื่อสร้างโมดูลวงจรตามที่ต้องการพร้อมกับนำข้อมูล Bus macro VHDL template ที่ได้จากซอฟต์แวร์ GoAhead มาเป็นโมดูลวงจรหนึ่งใน Static part เช่นกัน นอกจากนั้น

ต้องนำข้อมูลจาก Config Prohibit.ucf มารวมไว้ใน UCF file ที่ซอฟต์แวร์ Xilinx สร้างไว้ เมื่อเสร็จขั้นตอนการออกแบบโค๊ด VHDL แล้ว ใช้ซอฟต์แวร์ Xilinx ISE ทำกระบวนการสังเคราะห์วงจรฮาร์ดแวร์ และวางตำแหน่งวงจรที่สร้างลงในโครงสร้างของเอฟพีจีเอ ผลลัพธ์ของกระบวนการนี้จะทำให้ได้ไฟล์ Top_map.ncd

(3) Blocker Merging Process คือ กระบวนการของซอฟต์แวร์ GoAhead ทำการประมวลผลสคริปคำสั่ง Merge blocker ลักษณะการทำงานของคำสั่ง คือ การนำข้อมูล NCD file ที่ได้จาก Xilinx ISE กับข้อมูล StaticBlocker.xdl จากซอฟต์แวร์ GoAhead มารวมกัน ทำให้ได้ NCD file ใหม่ซึ่งเป็นข้อมูลที่นำมาใช้ เพื่อทำขั้นตอนการ Routing โดยที่ทรัพยากรภายในเอฟพีจีเอที่ถูกสงวนไว้ (Prohibit) จะถูกป้องกันไม่ให้ใช้งานโดยการเชื่อมต่อทรัพยากรเหล่านี้เข้าด้วยกันแล้วสร้างเส้นสัญญาณ (Netlist) เชื่อมไว้ด้วยกันทั้งหมดเสมือนกับการ Route ทรัพยากรเหล่านี้ไว้ก่อน เพื่อเป็นการป้องกันว่าทรัพยากรเหล่านี้จะไม่ถูกทำการ Route ได้อีก ซึ่งเป็นการเตรียมพร้อมสำหรับกระบวนการ Routing สุดท้าย จากขั้นตอนนี้ทำให้ได้ NCD file ที่มีข้อมูลว่าทรัพยากรที่สงวนไว้ไม่ให้ใช้งานถูกทำการกำหนดเส้นทางการเชื่อมต่อสัญญาณ (Route) ไว้หมดแล้ว นั่นคือ พื้นที่ Reconfigurable part แต่จะเหลือเฉพาะส่วนที่สามารถใช้งานได้ ซึ่งก็คือ Static part

(4) Routing Process คือ การสร้างเส้นทางเชื่อมต่อสัญญาณในลักษณะลากเส้นทางเชื่อมต่ออัตโนมัติ (Auto route) โดยทำการกำหนดเส้นทางการเชื่อมต่อสัญญาณจากข้อมูล NCD file ที่ได้จากขั้นตอน Blocker merging process ผลลัพธ์ที่ได้ คือ NCD file (Top.ncd) ที่โมดูลวงจร ต่าง ๆ ที่ออกแบบไว้ จะถูกทำการกำหนดเส้นทางการเชื่อมต่อของแต่ละสัญญาณได้ครบแล้ว จากนั้นทำการลบสัญญาณ (netlist) ที่ถูกสร้างไว้ก่อนหน้าการทำกระบวนการ Route ในครั้งนี้ ซึ่งก็คือ สัญญาณ (netlist) ที่ได้จากการกำหนดเส้นทางการเชื่อมต่อระหว่างทรัพยากรที่สงวนไว้ (Prohibit) ทั้งหมด เมื่อลบสัญญาณส่วนนี้ไปแล้วเหลือเพียงโมดูลวงจรที่ทำการกำหนดเส้นทางการเชื่อมต่อเฉพาะส่วนที่ออกแบบไว้ใน Static part เท่านั้น

(5) Bitgen Process คือ กระบวนการสุดท้ายก่อนจะได้ไฟล์ซึ่งเป็นรูปแบบไฟล์สำหรับโปรแกรมลงเอฟพีจีเอ กระบวนการ Bitgen ทำหน้าที่สร้างไฟล์วงจร (Bitstream file) โดยรับอินพุตเป็น NCD file แล้วทำการเปลี่ยนเป็น Configuration bitstream file นามสกลุ .bit ซึ่งเป็นรูปแบบไฟล์ที่สามารถโปรแกรมลงเอฟพีจีเอให้ทำงานได้

จะเห็นได้ว่ากระบวนการออกแบบไฟล์วงจรแบบ Partial reconfiguration ให้รองรับเอฟพีจีเอ Spartan-6 ทำได้โดยใช้ซอฟต์แวร์ GoAhead และ Xilinx ISE ทำงานร่วมกัน โดยซอฟต์แวร์ GoAhead ทำกระบวนการกำหนดพื้นที่, กำหนดประเภทและคุณสมบัติของวงจรที่ออกแบบให้รองรับกับการโปรแกรมแบบ Partial reconfiguration ซึ่งกระบวนการเหล่านี้ซอฟต์แวร์เดิมของ Xilinx ไม่รองรับ สำหรับหัวข้อถัดไปจะนำเสนอขั้นตอนการออกแบบให้รองรับกับการนำโพรเซสเซอร์มาใช้งานร่วม เพื่อเพิ่มความสามารถของเอฟพีจีเอ

3.3 การออกแบบไมโครโพรเซสเซอร์ในพื้นที่ Static Part

การออกแบบวงจรในพื้นที่ Static part ประกอบไปด้วย 2 ส่วน คือ วงจรควบคุมการโปรแกรมในพื้นที่ Reconfigurable part และวงจรอื่น ๆ ตามความต้องการผู้ออกแบบ สำหรับงานวิจัยนี้ต้องการเพิ่มความสามารถให้เอฟพีจีเอทำหน้าที่เป็นเซนเซอร์โนดในระบบเครือข่ายเซนเซอร์ไร้สาย ซึ่งโดยปกติแล้วเซนเซอร์โนดประกอบไปด้วยไมโครคอนโทรลเลอร์ทำหน้าที่ควบคุมการทำงาน การรับส่งข้อมูล ดังนั้นจึงเพิ่มการออกแบบวงจรในพื้นที่ Static part ให้ทำหน้าที่เป็นเสมือนไมโครคอนโทรลเลอร์โดยเลือกใช้ไมโครโพรเซสเซอร์เพื่อทำหน้าที่เป็นส่วนประมวลผล ผู้ผลิตเอฟพีจีเอ Xilinx มีฟังก์ชันฮาร์ดแวร์สำเร็จรูปสำหรับทำหน้าที่เป็นไมโครโพรเซสเซอร์ชื่อว่า Microblaze Processor รายละเอียดมีดังนี้

3.3.1 คุณสมบัติไมโครโพรเซสเซอร์ Microblaze

Microblaze processor เป็นไมโครโพรเซสเซอร์ที่ผลิตโดยบริษัท Xilinx ผู้ผลิตเอฟพีจีเอ โดยมีแนวคิดที่จะสร้างไมโครโพรเซสเซอร์ไว้ภายในเอฟพีจีเอ เพื่อให้ผู้ใช้งานสามารถพัฒนาเอฟพีจีเอด้วยการเขียนโปรแกรมภาษาขั้นสูงเช่น C, C++ ได้โดยไม่จำเป็นต้องเขียนด้วยภาษา HDL เพียงอย่างเดียว อย่างไรก็ตาม Microblaze ไม่ได้เป็นฮาร์ดแวร์ภายในเอฟพีจีเออย่างแท้จริง แต่ใช้กระบวนการออกแบบวงจรเพื่อสร้างขึ้นมา การใช้งานจริงผู้ผลิตได้พัฒนาออกมาในรูปแบบ IP core เป็นโมดูลวงจรหนึ่งทำให้ใช้งานได้ง่าย และสามารถกำหนดรายละเอียดของทรัพยากร (Resource), คุณสมบัติของไมโครโพรเซสเซอร์ได้

Microblaze เป็นไมโครโพรเซสเซอร์ขนาด 32 บิต ประมวลผลคำสั่งแบบ RISC ในการใช้งานจริงนั้นผู้ใช้สามารถกำหนดคุณสมบัติของไมโครโพรเซสเซอร์ได้ แต่จะมีบางส่วนที่ผู้ใช้ไม่สามารถกำหนดได้ คือ

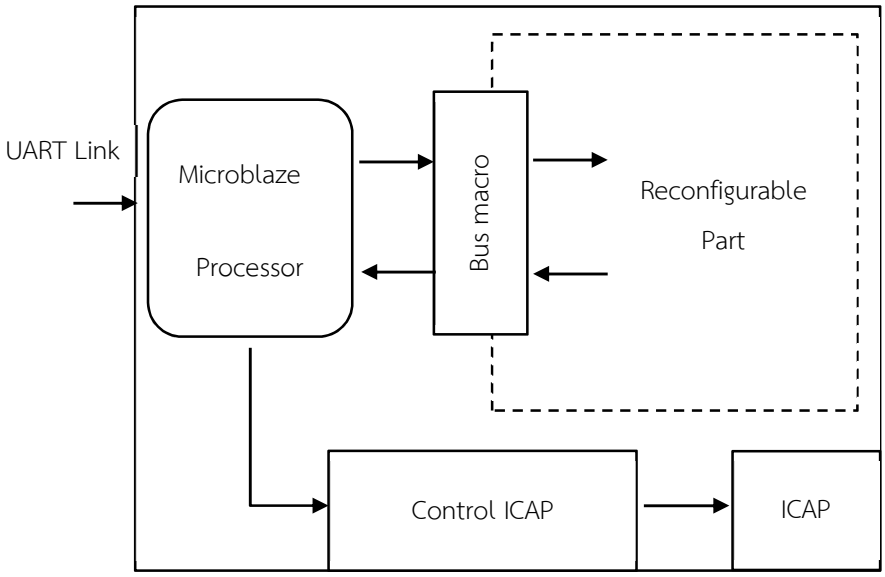
- ขนาด General purpose register 32 บิต
- ขนาด Instruction word 32 บิต ประกอบไปด้วย 3 คำสั่งและ 2 Addressing mode
- ขนาด Address bus 32 บิต
- ประมวลผลแบบ Single issue pipeline
- รองรับการทำงานโดยภาษา C, C++

สำหรับนอกเหนือจากที่กล่าวมานี้ ผู้ผลิตให้ผู้ใช้สามารถกำหนดคุณสมบัติได้ตามความต้องการ โดยในงานวิจัยนี้ได้ทดลองใช้งานกำหนดคุณสมบัติดังนี้

- รันบนความถี่ Clock 66 MHz
- การเชื่อมต่อผ่าน UART interface
- GPIO 1 port ขนาด 32 บิต

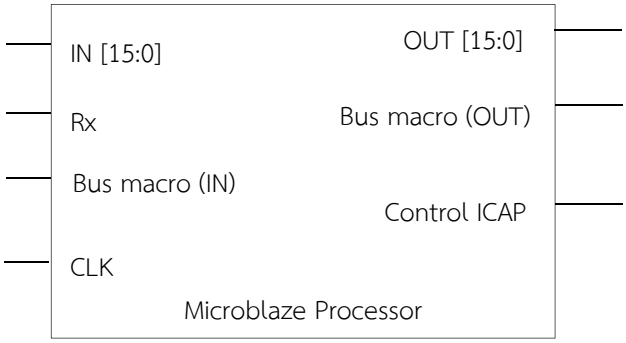
3.3.2 การออกแบบวงจรไมโครโพรเซสเซอร์ในพื้นที่ Static Part

ไมโครโพรเซสเซอร์ Microblaze เสมือนเป็นโมดูลวงจรหนึ่ง โดยงานวิจัยนำ Microblaze มาเป็นวงจรในพื้นที่ Static part มีหน้าที่หลัก คือ ทำหน้าที่ประมวลผลข้อมูลจากอินพุตภายนอกและอินพุตจาก Reconfigurable part, ประมวลผลข้อมูลแล้วส่งออกไปเอาต์พุตภายนอกหรือเอาต์พุตไปยัง Reconfigurable part และทำหน้าที่รับข้อมูลไฟล์วงจรผ่านทาง UART interface เพื่อส่งข้อมูลไฟล์วงจรให้กับโมดูล Control ICAP ทำหน้าที่โปรแกรมวงจรในพื้นที่ Reconfigurable part รูปแบบโครงสร้างแสดงดังภาพประกอบ 3-7



ภาพประกอบ 3-7 การออกแบบวงจรพื้นที่ Static Part ร่วมกับ Microblaze Processor

โครงสร้างไมโครโพรเซสเซอร์ Microblaze ประกอบไปด้วยอินพุตจากสองส่วน คือ UART link (Rx) และจากโมดูล Bus macro โดยที่อินพุตจาก UART คือ ข้อมูลไฟล์วงจร (Partial bitsream) ที่ได้รับจาก UART interface และอีกส่วน คือ อินพุตจาก Bus macro ซึ่งเป็นเอาต์พุตจากวงจรในพื้นที่ Reconfigurable part สำหรับเอาต์พุตของ Microblaze มีสองส่วน คือ เอาต์พุตส่งข้อมูลวงจรไปยัง Control ICAP เพื่อโปรแกรมวงจรและเอาต์พุตต่อกับ Bus macro เพื่อส่งเอาต์พุตให้วงจรในพื้นที่ Reconfigurable part นอกจากนั้นได้เพิ่มอินพุตเอาต์พุตต่อกับอุปกรณ์ภายนอกได้โดยกำหนดให้เป็น GPIO ขนาด 16 บิต ไดอะแกรมอินพุตเอาต์พุตของ Microblaze แสดงดังภาพประกอบ 3-8



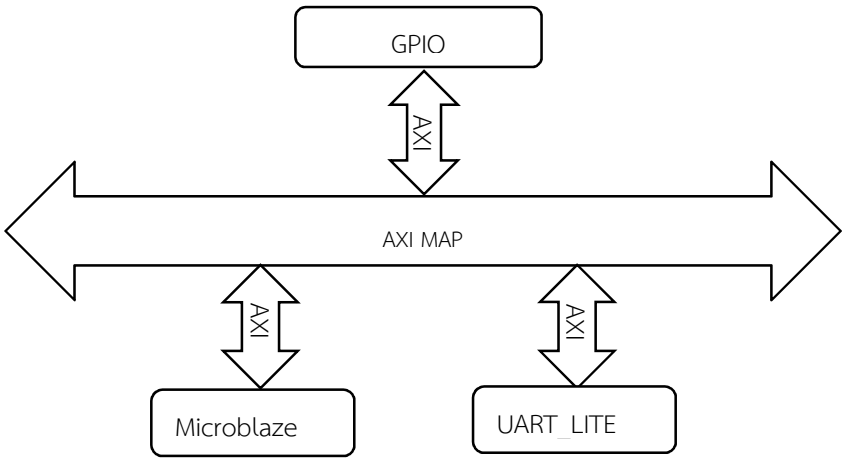
ภาพประกอบ 3-8 ไดอะแกรมอินพุตเอาต์พุตของโมดูล Microblaze

หน้าที่การทำงานของแต่ละสัญญาณได้อธิบายไว้ดังตารางตารางที่ 3-1

ตารางที่ 3-1 หน้าที่การทำงานของอินพุตและเอาต์พุตโมดูล Microblaze

สัญญาณ	ประเภท	หน้าที่การทำงาน
CLK	อินพุต	สัญญาณนาฬิกาความถี่ 66 MHz
Rx	อินพุต	สัญญาณข้อมูลขนาด 1 บิต จาก UART
Bus macro (IN)	อินพุต	อินพุตจาก Bus macro
IN [15:0]	อินพุต	อินพุตข้อมูลขนาด 16 บิต
OUT [15:0]	เอาต์พุต	เอาต์พุตข้อมูลขนาด 16 บิต
Control ICAP	เอาต์พุต	สัญญาณข้อมูลขนาด 1 บิตไปยังโมดูล Control ICAP
Bus macro (OUT)	เอาต์พุต	เอาต์พุตส่งให้กับ Bus macro

ผู้ใช้งานสามารถกำหนดคุณสมบัติของ Microblaze โดยการออกแบบวงจร Peripheral ภายในโพรเซสเซอร์ Microblaze ซึ่งผู้ผลิต Xilinx ได้ออกแบบ Peripheral เหล่านี้ไว้แล้ว เช่น UART controller, Memory controller เป็นต้น ผู้ใช้งานสามารถเลือก Peripheral เหล่านี้มาเชื่อมต่อกันผ่าน Advance Extensible Internal (AXI) interface โดย Peripheral แต่ละตัวมี Address ของตัวเอง เมื่อต้องการเรียกใช้ Peripheral ใด ๆ ต้องทำการกำหนดตำแหน่งอ้างอิง (Address) ของ Peripheral นั้น สำหรับงานวิจัยนี้เรียกใช้ Peripheral 2 ส่วน คือ GPIO สำหรับสร้างเป็นอินพุตเอาต์พุตพอร์ต และ UART LITE สำหรับเชื่อมต่อกับสัญญาณ Rx ใน UART interface ภาพประกอบ 3-9 แสดงให้เห็นโครงสร้างการเชื่อมต่อวงจร Peripheral ต่าง ๆ ที่มีการใช้งานร่วมกับ Microblaze

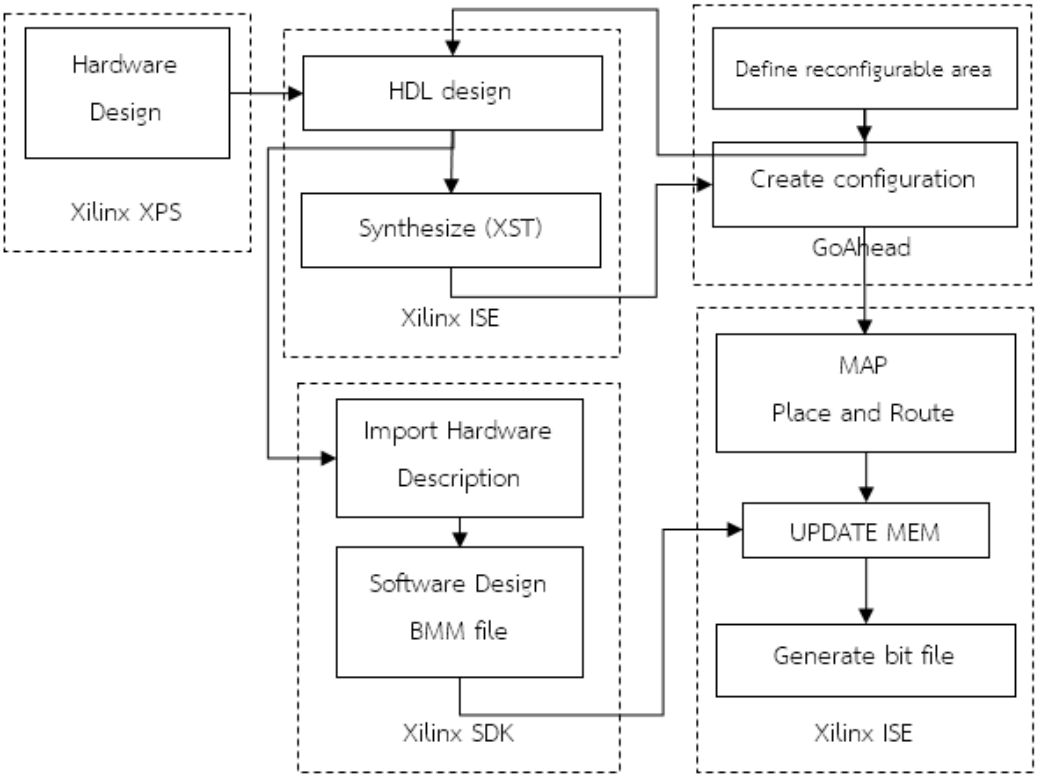


ภาพประกอบ 3-9 โครงสร้างการเชื่อมต่อ Peripheral กับ Microblaze

ผู้วิจัยกำหนดให้ Microblaze ทำหน้าที่รับข้อมูลไฟล์วงจรแล้วส่งต่อให้กับโมดูลที่ทำหน้าที่โปรแกรมวงจร และทำหน้าที่รับส่งข้อมูลระหว่างพื้นที่ Static part กับ Reconfigurable part ดังนั้นคุณสมบัติของ Microblaze ที่ผู้วิจัยออกแบบเป็นคุณสมบัติที่จำเป็นสำหรับการทำหน้าที่เหล่านั้นเท่านั้น การออกแบบเพื่อนำไปใช้งานจริงสามารถปรับเปลี่ยนหน้าที่ หรือเพิ่มคุณสมบัติอื่น ๆ เพื่อให้รองรับกับความต้องการ

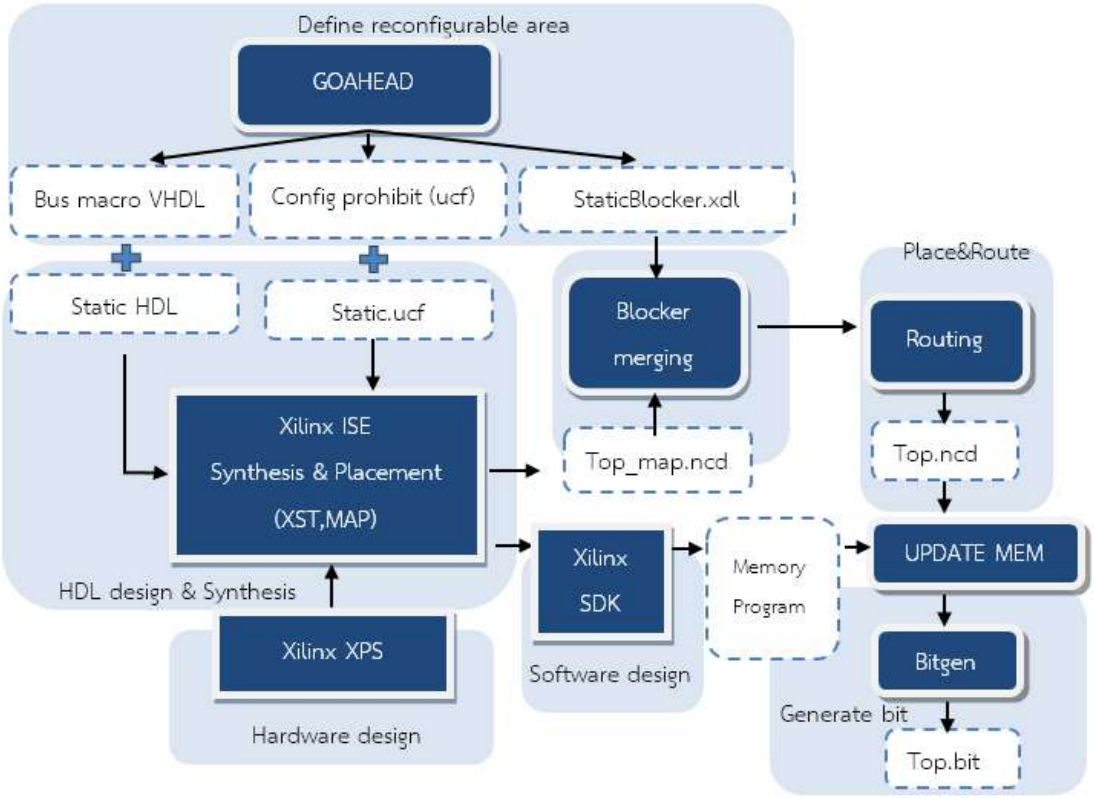
3.3.3 กระบวนการออกแบบไฟล์วงจรพื้นที่ Static Part ที่มีการใช้งานไมโครโพรเซสเซอร์

กระบวนการออกแบบไมโครโพรเซสเซอร์ Microblaze เพื่อใช้งานพื้นที่ Static part เรียกใช้ซอฟต์แวร์ Xilinx XPS, Xilinx ISE, Xilinx SDK และ GoAhead ขั้นตอนทั้งหมดแสดงดังภาพประกอบ 3-10 โดยที่กระบวนการส่วนใหญ่มีรายละเอียดเหมือนกับการออกแบบพื้นที่ Static part ที่ไม่มี Microblaze



ภาพประกอบ 3-10 ขั้นตอนการออกแบบไฟล์วงจรที่มีโพรเซสเซอร์ Microblaze

กระบวนการที่เพิ่มเติม คือ กระบวนการออกแบบโครงสร้าง Microblaze (Hardware design process) เพื่อกำหนดคุณสมบัติของ Microblaze กระบวนการนี้ต้องทำก่อนการออกแบบวงจร (HDL design process) นอกจากนี้กระบวนการออกแบบซอฟต์แวร์ควบคุมการทำงาน Microblaze มีกระบวนการ Import hardware description คือ การนำเข้าข้อมูลคุณสมบัติของ Microblaze สำหรับเขียนซอฟต์แวร์ให้รองรับการทำงานบนพื้นฐานคุณสมบัติของ Microblaze ที่ได้ออกแบบ การเขียนซอฟต์แวร์ใช้ภาษา C มีคอมไพเลอร์อยู่ในซอฟต์แวร์ Xilinx SDK แล้ว ผลลัพธ์ที่ได้ คือ .bmm file จะถูกนำมาเขียนลงในหน่วยความจำของ Microblaze ในกระบวนการ Update mem process ก่อนการสร้างไฟล์วงจรในกระบวนการ Generate bit file การนำกระบวนการและขั้นตอนออกแบบวงจรมาใช้งานจริงมีกระบวนการดังภาพประกอบ 3-11



ภาพประกอบ 3-11 กระบวนการออกแบบไฟล์วงจร Static Part ร่วมกับ Microblaze

การออกแบบมีทั้งหมด 8 ขั้นตอนซึ่งบางส่วนได้อธิบายไว้ในหัวข้อก่อนหน้าแล้ว สำหรับหัวข้อนี้จะอธิบายเพียงขั้นตอนที่เพิ่มเติมซึ่งมี 3 ขั้นตอนดังนี้

(1) Xilinx Platform Studio (XPS) process คือ กระบวนการออกแบบคุณสมบัติของ Microblaze ใช้ซอฟต์แวร์ Xilinx XPS เพื่อกำหนดคุณสมบัติของ Microblaze เช่น ความเร็วของสัญญาณ clock, Peripheral GPIO และ UART, ขนาดของ Program memory เหล่านี้เป็นต้น เมื่อกำหนดทั้งหมดแล้วจะสามารถสร้างโมดูล Microblaze และนำไปใช้งานร่วมกับโมดูลวงจรอื่น ๆ ของวงจรในพื้นที่ Static part เมื่อออกแบบวงจรทั้งหมดเสร็จแล้วทำกระบวนการ Synthesis and placement ได้ผลลัพธ์ คือ ไฟล์ข้อมูล NCD เพื่อนำไปใช้ในกระบวนการถัดไป

(2) Xilinx Software Development Toolkit (Xilinx SDK) process กระบวนการของซอฟต์แวร์ Xilinx ISE ทำการเรียกใช้ซอฟต์แวร์ Xilinx SDK เพื่อนำข้อมูลการออกแบบวงจรจาก Xilinx ISE สร้างเป็นโครงสร้างของ Microblaze (Hardware platform) เพื่อให้ผู้ใช้งานสามารถพัฒนาโปรแกรมบนไมโครโพรเซสเซอร์ Microblaze โดยอ้างอิงการเขียนโปรแกรมบนพื้นฐาน Hardware platform ดังกล่าว ซึ่ง Hardware platform ประกอบไปด้วยข้อมูลที่ระบุว่าไมโครโพรเซสเซอร์มีทรัพยากร (Resource) ใดให้ใช้บ้าง เช่น ขนาดหน่วยความจำ (RAM), อินพุตเอาต์พุต (GPIO), UART เป็นต้น โดยข้อมูลโปรแกรมที่เขียนนั้นจะถูกคอมไพล์และเก็บไว้แบบ Memory program file ในขณะเดียวกันหลังจากออกแบบวงจรทั้งหมดแล้วในซอฟต์แวร์ Xilinx ISE กระบวนการต่อมา คือ Blocker merging และ Routing

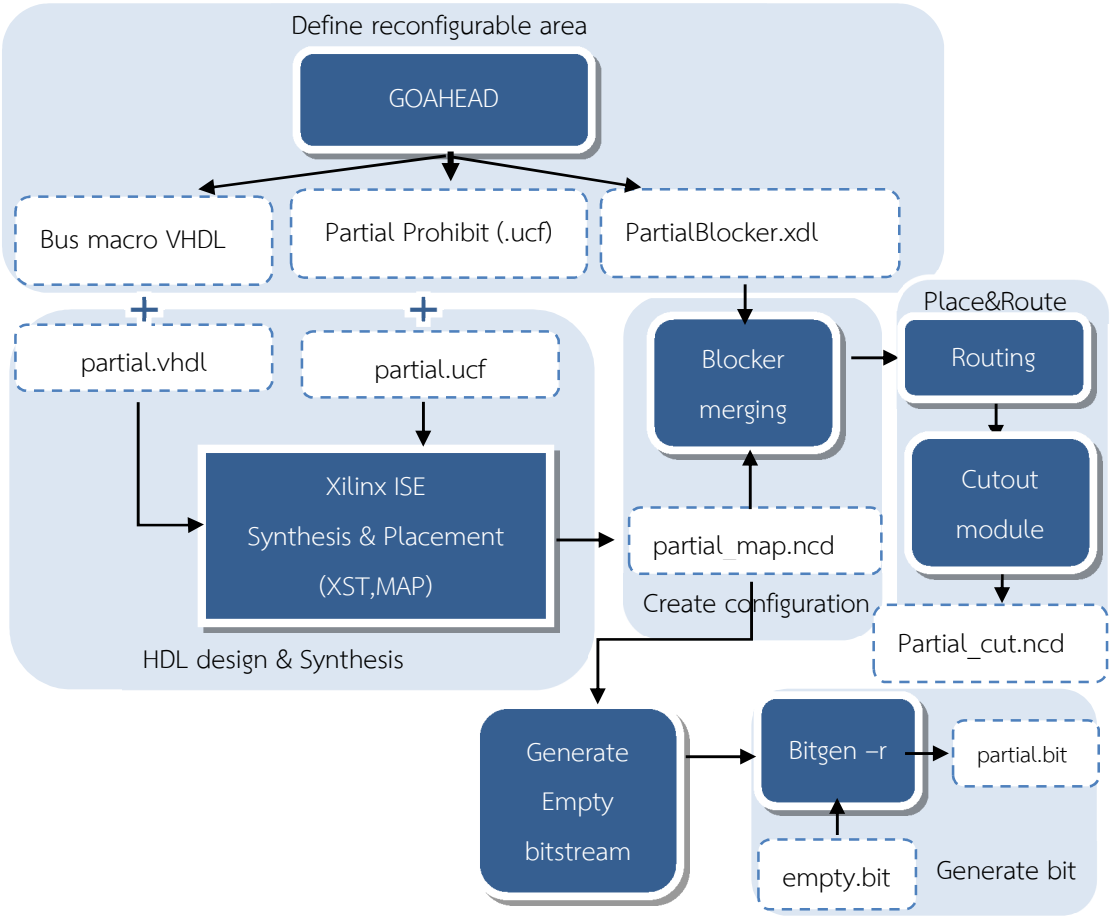
ซึ่งได้อธิบายไว้แล้วในหัวข้อก่อนหน้านี้ เมื่อผ่านกระบวนการ Routing จะทำให้ได้ไฟล์ NCD ซึ่งพร้อมที่จะทำขั้นตอน Bitgen เพื่อสร้างไฟล์โปรแกรม (Bitstream)

(3) Update MEM คือ กระบวนการอัพเดทหน่วยความจำของ Microblaze เนื่องจากมีการใช้งาน Microblaze ซึ่งเป็นไมโครโพรเซสเซอร์ ดังนั้นก่อนการโปรแกรมลงเอฟพีจีเอจริงต้องทำการโปรแกรมในส่วนของ Program memory ของ Microblaze เข้าไปด้วย โดยที่ข้อมูลโปรแกรมบน Microblaze ได้จากซอฟต์แวร์ Xilinx SDK คือ Memory program file โดยไฟล์นี้จะเขียนลงใน Block RAMs ของ Microblaze โดยต้องอ้างอิง Block RAMs จากไฟล์ Top.ncd ซึ่งได้ออกแบบไว้ก่อนหน้านี้ จากนั้นทำขั้นตอน Bitgen เพื่อสร้างไฟล์โปรแกรม (Bitstream) ที่สามารถโปรแกรมลงเอฟพีจีเอ

ดังนั้นการนำโพรเซสเซอร์ Microblaze มาเป็นส่วนหนึ่งของพื้นที่ Static part เพื่อเพิ่มความสามารถของเอฟพีจีเอ โดยออกแบบไมโครโพรเซสเซอร์ให้ทำหน้าที่รับข้อมูลไฟล์วงจรเพื่อโปรแกรม และรับส่งข้อมูลระหว่างพื้นที่ Static part กับ Reconfigurable part พร้อมกับนำเสนอคุณสมบัติพื้นฐานที่ทำให้รองรับหน้าที่เหล่านั้น ซึ่งการนำไปใช้งานจริง ผู้ใช้สามารปรับเปลี่ยนคุณสมบัติของ Microblaze ได้ตามความมาเหมาะสม นอกจากนี้แล้วกระบวนการออกแบบไฟล์วงจร Partial reconfiguration ที่ใช้ซอฟต์แวร์ GoAhead และ Xilinx ISE ไม่รองรับกับกระบวนการออกแบบของ Microblaze ผู้วิจัยได้ปรับปรุงกระบวนการออกแบบไฟล์วงจรแบบ Partial reconfiguration เดิมให้รองรับกับการนำ Microblaze มาใช้งานร่วมในพื้นที่ Static part

3.4 กระบวนการออกแบบไฟล์วงจรพื้นที่ Reconfigurable Part

กระบวนการออกแบบ Reconfigurable part มีขั้นตอนคล้ายกับกระบวนการออกแบบ Static part แต่ต่างกันตรงที่ไฟล์สำหรับโปรแกรม (Bitstream) ซึ่งจะใช้แบบ Partial Bitstream File การออกแบบจะทำควบคู่ไปกับ Static part เพราะต้องใช้พื้นที่ชุดเดียวกัน ภาพประกอบ 3-12 คือ กระบวนการและขั้นตอนการออกแบบ Reconfigurable part ทั้งหมด



ภาพประกอบ 3-12 กระบวนการออกแบบทางซอฟต์แวร์ของพื้นที่ Reconfigurable Part

กระบวนการและขั้นตอนมี 7 กระบวนการ มีรายละเอียดดังนี้

(1) GoAhead Process คือ กระบวนการกำหนดพื้นที่ Reconfigurable part และกำหนดตำแหน่ง Bus macro ซึ่งต้องกำหนดตำแหน่ง SLICE ให้ตรงกับที่กำหนดตำแหน่งไว้ใน Static part จากนั้นซอฟต์แวร์ GoAhead สร้างไฟล์ PartialBlocker.xdl ซึ่งระบุทรัพยากรที่สามารถใช้งานได้ในทำนองเดียวกับการออกแบบ Static part ซอฟต์แวร์ GoAhead สร้าง Bus macro VHDL และ Config_Prohibit.ucf ซึ่งการออกแบบใน Reconfigurable part ทรัพยากรที่ถูกสงวนไว้ (Prohibit) คือ ทรัพยากรในพื้นที่ Static part ทั้งหมด

(2) Xilinx ISE Process Synthesis and Placement คือ กระบวนการออกแบบโมดูลวงจรที่ต้องการให้ทำงานใน Reconfigurable part โดยใช้ซอฟต์แวร์ Xilinx ISE เช่นเดียวกันกับ Static part การออกแบบจะต้องนำ Bus macro VHDL จากซอฟต์แวร์ GoAhead เป็นหนึ่งในโมดูลวงจรทั้งหมดและนำข้อมูลทรัพยากรที่ถูกสงวนไว้ (Prohibit) จากไฟล์ Config_Prohibit.ucf มารวมกับ UCF file ที่ซอฟต์แวร์ Xilinx สร้างขึ้นมาโดยอัตโนมัติ เมื่อออกแบบโมดูลวงจรของระบบทั้งหมดแล้วจะเข้าสู่กระบวนการสังเคราะห์วงจร (Synthesis & placement) ได้ไฟล์ Circuit design (.NCD) ของโมดูลวงจรที่ออกแบบ คือ Partial_map.ncd

(3) Blocker Merging Process ขั้นตอนนี้ทำโดยซอฟต์แวร์ GoAhead ซึ่งใช้คำสั่ง Merge blocker เหมือนกับ Static part คือ การนำข้อมูล NCD file (Partial_map.ncd) ที่ได้จากซอฟต์แวร์ Xiliinx ISE กับข้อมูลจากไฟล์ PartialBlocker.xdl จากซอฟต์แวร์ GoAhead มารวมกันทำให้ได้ข้อมูลทรัพยากรที่จะนำมาทำการกำหนดเส้นทางสัญญาณ (Routing) โดยที่ทรัพยากรที่ถูกสงวนไว้ (Prohibit) ได้ทำการเชื่อมต่อทุกทรัพยากรเข้าด้วยกันแล้วสร้างเส้นสัญญาณ (Netlist) เชื่อมไว้ด้วยกันทั้งหมดก่อนเพื่อป้องกันทรัพยากรเหล่านี้ไม่ให้ถูกทำการกำหนดเส้นทางสัญญาณ (Routing) ในขั้นตอนต่อไป จากกระบวนการนี้จะได้ข้อมูลไฟล์ Circuit design (NCD file) ซึ่งพื้นที่ที่ถูกสงวนไว้ (Prohibit) ไว้ คือ Static part จะถูกทำการกำหนดเส้นทางสัญญาณ (Route) ไว้หมดแล้วเหลือเฉพาะพื้นที่ที่ยังไม่ถูกกำหนดเส้นทางสัญญาณ (Route) คือ Reconfigurable part

(4) Generate Empty Bitstream Process คือ การสร้าง Circuit design (NCD file) ที่ไม่มีข้อมูลวงจรใด ๆ เสมือนกับสร้าง Empty NCD file จากนั้นทำการสร้างเป็นไฟล์ Bitstream ได้เป็น Empty.bit ซึ่งข้อมูลนี้จะถูกนำไปใช้ในกระบวนการอื่นต่อไป

(5) Routing process คือ กระบวนการสร้างเส้นทางเชื่อมต่อสัญญาณวงจรโดยกำหนดให้วางเส้นทางเชื่อมต่อแบบอัตโนมัติ (Auto route) เช่นเดียวกับการทำงานใน Static part โดยใช้ข้อมูลจาก NCD file ที่ได้จากกระบวนการ Blocker merging ผลลัพธ์ที่ได้ คือ วงจรที่ออกแบบไว้ในพื้นที่ Reconfigurable part จะถูกวางเส้นทางสัญญาณ จากนั้นลบสัญญาณ (Netlist) ที่ถูกสร้างไว้ก่อนการทำ Routing ครั้งนี้ซึ่งก็ คือ สัญญาณ (Netlist) เส้นทางเชื่อมต่อทรัพยากรที่สงวนไว้ (Prohibit) ทั้งหมด เมื่อลบสัญญาณส่วนนี้ไปแล้วเหลือเพียงวงจรที่ทำการเชื่อมต่อสัญญาณ (Route) เฉพาะส่วนที่ออกแบบไว้ในพื้นที่ Reconfigurable part เท่านั้น

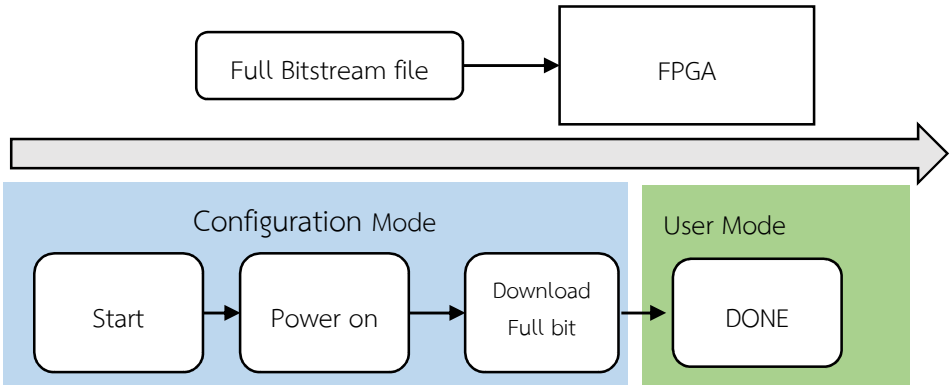
(6) Cut out module process ขั้นตอนนี้ใช้ซอฟต์แวร์ GoAhead ทำงานโดยมีสคริปชื่อคำสั่ง CutoffFromDesign คือ คำสั่งลบข้อมูลทรัพยากร SLICE และการเชื่อมต่อสัญญาณ (Route) ที่อยู่นอกพื้นที่ ที่กำหนดไว้ทั้งหมด เนื่องจากข้อมูล NCD file ที่ได้ถึงแม้จะมีการสงวนทรัพยากร (Prohibit) ไว้แล้วแต่ทรัพยากรบางประเภท เช่น BUFG clock, Bus macro บางส่วนนั้นเป็นการใช้งานสัญญาณ Netlist ร่วมกันระหว่าง Static part และ Reconfigurable part ทำให้เมื่อทำขั้นตอน Routing สัญญาณ Netlist เหล่านี้ยังคงอยู่ ดังนั้นต้องทำการตัดทิ้งในกระบวนการนี้ ผลลัพธ์จากการทำ Cut out module จะได้ NCD file ที่พร้อมสำหรับขั้นตอนการสร้างไฟล์สำหรับโปรแกรมลงเอฟพีจีเอ

(7) Bitgen –r Process คือ กระบวนการสร้างไฟล์วงจร (Bitstream file) แต่จะใช้ไฟล์รูปแบบ Partial bitstream file ซึ่งทำได้โดยใช้คุณสมบัติ Differential bitgen [17] หลักการ Differential bitgen คือ การสร้าง Bitstream file (.bit) ที่ได้จากการเปรียบเทียบข้อมูลระหว่างไฟล์วงจร 2 ชุด แล้วเลือกข้อมูลเฉพาะส่วนที่แตกต่างกันระหว่างวงจร 2 ชุดนี้มาเขียนเป็นไฟล์วงจรใหม่ สำหรับงานวิจัยต้องการไฟล์โปรแกรมวงจรแบบ Partial Bitstream คือ ไฟล์ที่ประกอบไปด้วยวงจรฉพาะพื้นที่ Reconfigurable part เท่านั้น และต้องไม่มีวงจรของ Static part รวมถึงคำสั่งที่เป็นการรีเซตเอฟพีจีเอทั้งหมด เพราะโดยปกติแล้วคำสั่ง Bitgen ของซอฟต์แวร์ Xilinx จะสร้างไฟล์วงจร Bitstream ซึ่งนอกจากจะมีข้อมูลวงจรที่จะโปรแกรมแล้วยังมีคำสั่งบางส่วน คือ การรีเซตเอฟพีจีเอด้วยวิธีสร้างไฟล์ Partial bitsteam โดยใช้คุณสมบัติ Differential bitgen จะใช้ไฟล์วงจร Empty.bit กับ Partial_cut.ncd โดยเมื่อนำไฟล์ Empty.bit ซึ่งเป็นไฟล์ที่ไม่มีวงจรกับไฟล์ Partial_cut.ncd เดิมที่ได้จากขั้นตอนก่อนหน้านี้ ผ่านกระบวนการ Bitgen-r ผลลัพธ์ที่ได้จะได้เป็นไฟล์ที่มีข้อมูลวงจรเฉพาะส่วนของพื้นที่ Reconfigurable part เท่านั้น นั่นคือ ไฟล์แบบ Partial Bitsteam ซึ่งเป็นรูปแบบไฟล์ที่สามารถโปรแกรมลงเอฟพีจีเอได้

ดังที่กล่าวมานั้นจะเห็นได้ว่าการออกแบบไฟล์วงจรของพื้นที่ Reconfigurable part ไม่แตกต่างกันจากกระบวนการออกแบบไฟล์วงจรของพื้นที่ Static part แต่มีขั้นตอนเพิ่มเติม คือ การตัดวงจรนอกพื้นที่ Reconfigurable part และขั้นตอนสร้างไฟล์วงจรแบบ Partial bitstream เพื่อให้พื้นที่ Reconfigurable part ต้องสามารถโปรแกรมวงจรซ้ำได้โดยไม่กระทบพื้นที่อื่น และวงจรต้องอยู่ในพื้นที่ที่กำหนดไว้เท่านั้น หัวข้อถัดไปจะนำเสนอวิธีการ Reconfigure พื้นที่ Reconfigurable part โดยนำเสนอการ Reconfigure ผ่านสายสัญญาณและผ่านระบบไร้สาย

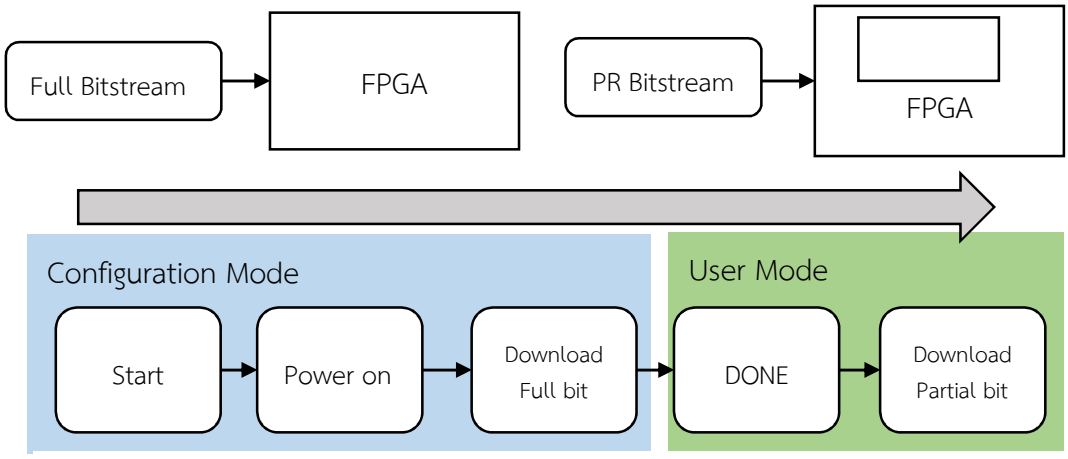
3.5 การออกแบบวิธีการ Reconfigure เอฟพีจีเอ

การโปรแกรมวงจรแบบ Partial reconfiguration มีลำดับการทำงานต่างกับการโปรแกรมแบบ Full reconfiguration ภาพประกอบ 3-13 แสดงลำดับการโปรแกรมวงจรแบบ Full reconfiguration เริ่มต้นเมื่อจ่ายไฟให้กับบอร์ด (Start) -> (Power on) บอร์ดเอฟพีจีเออยู่ในสถานะ Configuration mode จากนั้นดาวน์โหลดไฟล์วงจร (Download full bit) เมื่อดาวน์โหลดเสร็จแล้วบอร์ดจะเข้าสถานะ User mode และเริ่มทำงานตามวงจรที่โปรแกรม



ภาพประกอบ 3-13 ลำดับการโปรแกรมวงจรแบบ Full Reconfiguration

การโปรแกรมวงจรแบบ Partial reconfiguration ต้องดาวน์โหลดวงจร 2 ส่วน คือ Static part และ Reconfigurable part ภาพประกอบ 3-14 แสดงลำดับการโปรแกรมเริ่มต้นด้วยการดาวน์โหลด Static part ก่อนซึ่งมีลำดับการโปรแกรมเหมือนกับกระบวนการของ Full reconfiguration ถัดมาเมื่อเข้าสู่ User mode แล้วสามารถดาวน์โหลดไฟล์วงจร (Download Partial bit) เพื่อโปรแกรมพื้นที่ Reconfigurable part

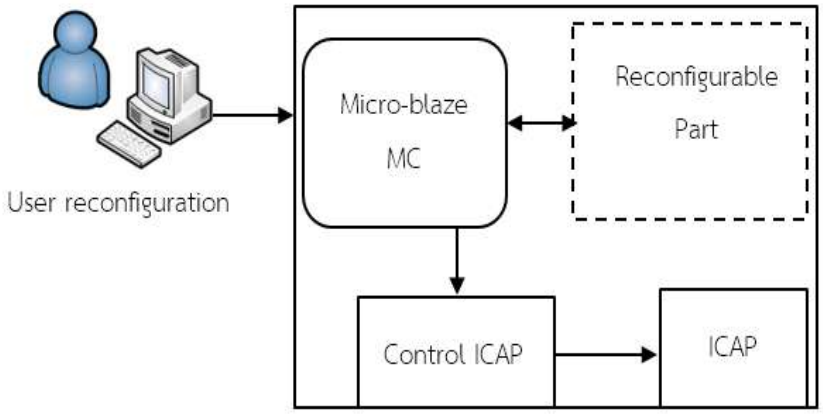


ภาพประกอบ 3-14 ลำดับการโปรแกรมวงจรแบบ Partial Reconfiguration

งานวิจัยนำเสนอรูปแบบการโปรแกรมวงจร 2 แบบ คือ การโปรแกรมวงจรผ่านสายสัญญาณ RS-232 และการโปรแกรมวงจรผ่านระบบไร้สาย (Wireless) รายละเอียดดังนี้

3.5.1 การโปรแกรมวงจรผ่านสายสัญญาณ RS-232

การโปรแกรมวงจรโดยใช้สายสัญญาณตามมาตรฐาน Serial RS-232 คอมพิวเตอร์ฝั่งผู้ใช้มองเห็นเป็นอุปกรณ์เชื่อมต่อกับ COM port ภาพประกอบ 3-15 แสดงรูปแบบการโปรแกรมวงจรผ่านสายสัญญาณโดยที่ฝั่งผู้ใช้ (User reconfiguration) ต่ออุปกรณ์เอฟพีจีเอผ่านทาง COM port device บนคอมพิวเตอร์ การโปรแกรมวงจรใช้วิธีการเหมือนกับการส่งไฟล์ผ่านทาง COM port



ภาพประกอบ 3-15 โครงสร้างรูปแบบการโปรแกรมวงจรผ่านสายสัญญาณ

จุดเด่นของการโปรแกรมผ่านสายสัญญาณ คือ ความถูกต้องของข้อมูลและอัตราเร็วในการดาวน์โหลด แต่มีข้อจำกัดในเรื่องระยะสายสัญญาณจำกัดอยู่ที่ประมาณ 50 ฟุต

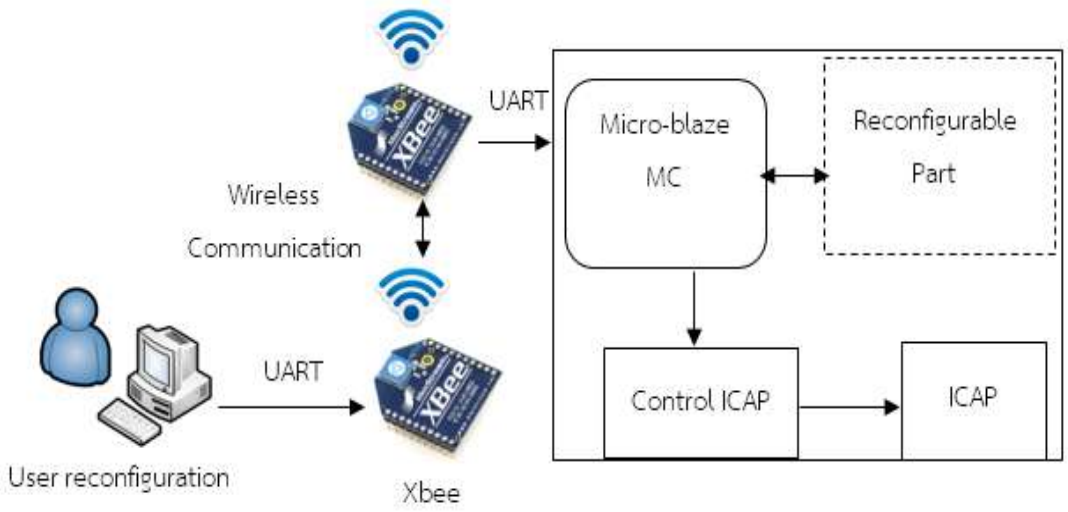
3.5.2 การโปรแกรมวงจรผ่านระบบไร้สาย (Wireless)

เพื่อให้เอฟพีจีเอรองรับการใช้งานเป็นเซนเซอร์โนดในระบบเครือข่ายเซนเซอร์-ไร้สาย (Wireless sensor networks) จึงได้มีการพัฒนาต่อยอดในส่วนของวิธีการโปรแกรมวงจรลงเอฟพีจีเอโดยใช้การสื่อสารตามรูปแบบมาตรฐาน Zigbee protocol ซึ่งเป็นมาตรฐานการรับส่งข้อมูลแบบไร้สาย (Wireless) ที่ใช้พลังงานต่ำ ทำงานที่ความถี่ 2.4 GHz มีอัตราการรับส่งข้อมูลในอากาศประมาณ 250 Kbps ซึ่งเป็นที่นิยมนำมาใช้กับระบบเครือข่ายเซนเซอร์ไร้สาย (Wireless sensor network) สำหรับฮาร์ดแวร์ที่จะนำมาใช้รับส่งข้อมูลแบบ Zigbee protocol นั้นใช้อุปกรณ์ Xbee module

ภาพประกอบ 3-16 คือ ระบบที่ได้ออกแบบใหม่โดยเปลี่ยนจากเชื่อมต่อระหว่างเอฟพีจีเอกับผู้ใช้ (User reconfiguration) โดยผ่านสายสัญญาณมาเป็นการเชื่อมต่อผ่านทาง Xbee module แล้วให้ Xbee module รับส่งข้อมูลระหว่างกันเพื่อใช้ในการส่งข้อมูลจากผู้ใช้ (User reconfiguration) ไปยังเอฟพีจีเอ โดยระบบประกอบไปด้วย Xbee module 2 ชุด กำหนดให้ตัวแรกต่อกับผู้ใช้ (User reconfiguration) ผ่านทาง UART (Serial port) และ Xbee

ตัวถัดมาต่อกับเอฟพีจีเอผ่านทาง UART เมื่อต้องการโปรแกรมวงจรเอฟพีจีจะส่งข้อมูลให้กับ Xbee ตัวที่เชื่อมต่ออยู่กับผู้ใช้งานเพื่อส่งต่อให้กับ Xbee อีกตัวที่ต่ออยู่กับเอฟพีจีเอ

จุดเด่นของระบบนี้ คือ สามารถโปรแกรมเอฟพีจีเอได้จากระยะไกลโดยไม่ต้องมีสายส่งข้อมูลและสามารถโปรแกรมเอฟพีจีเอได้หลายชุดพร้อมกัน



ภาพประกอบ 3-16 โครงสร้างรูปแบบการโปรแกรมวงจรผ่านระบบไร้สาย

จากที่กล่าวมานั้นจะเห็นได้ว่าการโปรแกรมวงจรแบบ Partial reconfiguration มีขั้นตอนต่างกับการโปรแกรมวงจรแบบ Ful reconfiguration เพราะการใช้งานแบบ Partial reconfiguration เอฟพีจีเอต้องอยู่ใน User mode ก่อน จึงจะทำงานได้ โดยที่การโปรแกรมครั้งแรกจะโปรแกรมวงจรพื้นที่ Static part ก่อน งานวิจัยเลือกวิธีการโปรแกรมวงจรพื้นที่ Partial reconfiguration โดยใช้โพรโตคอล UART เพื่อให้สามารถใช้งานได้ง่ายเพราะสามารถเชื่อมต่อกับคอมพิวเตอร์ได้ผ่านทางพอร์ท USB หรือ Serial นอกจากนั้นสามารถเชื่อมต่อกับโมดูล Xbee เพื่อรองรับการนำไปใช้เป็นเซนเซอร์โนดได้ทันที

บทที่ 4
การทดสอบและวิเคราะห์ประสิทธิภาพ

ในบทนี้กล่าวถึงวิธีการทดสอบการทำงานของระบบทั้งหมดโดยงานวิจัยทดสอบกับวงจรบวก (Adder) และวงจรอัลกอริทึม AES encryption รายละเอียดประกอบไปด้วยหัวข้อดังต่อไปนี้

4.1 หน่วยวัดการทำงานของระบบ
4.2 เครื่องมือในการทดสอบระบบ
4.3 ผลทดสอบการออกแบบ Bus Macro
4.4 ผลการทดสอบเอฟพีจีเอของพื้นที่ Static Part
4.5 การทดสอบเอฟพีจีเอของพื้นที่ Reconfigurable Part
4.6 สรุปผลการทดสอบ

4.1 หน่วยวัดการทำงานของระบบ (Measurement Unit Test)

งานวิจัยทดสอบบนเอฟพีจีเอโดยจะรายงานผล 3 ส่วน คือ ทรัพยากรที่ใช้สร้างวงจรในเอฟพีจีเอ (Resource utilization), กำลังงานสูญเสียของวงจร (Power consumption) และเวลาที่ใช้โปรแกรมวงจร (Configuration time)

4.1.1 การใช้ทรัพยากร (Resource Utilization)

ทรัพยากรเอฟพีจีเอที่ใช้รายงานการทดสอบประกอบไปด้วยจำนวน Look Up Tables (LUTs) , Register, Clock resource, จำนวนหน่วยความจำ RAM memory, จำนวนอินพุต เอาต์พุต และโมดูลเฉพาะ เช่น Phase Lock Loops (PLLs) เป็นต้น

4.1.2 กำลังงานสูญเสีย (Power Consumption)

กำลังงานสูญเสียได้จากการคำนวณโดยมีปัจจัยที่เกี่ยวข้อง เช่น ความถี่สัญญาณนาฬิกา, จำนวนทรัพยากรแต่ละประเภทที่ถูกใช้งาน, อินพุตเอาต์พุต เป็นต้น ดังนั้นจึงอ้างอิงชื่อตัวแปรเพื่อใช้รายงานผลกำลังงาน แบ่งตามประเภทของทรัพยากรดังตารางที่ 4-1

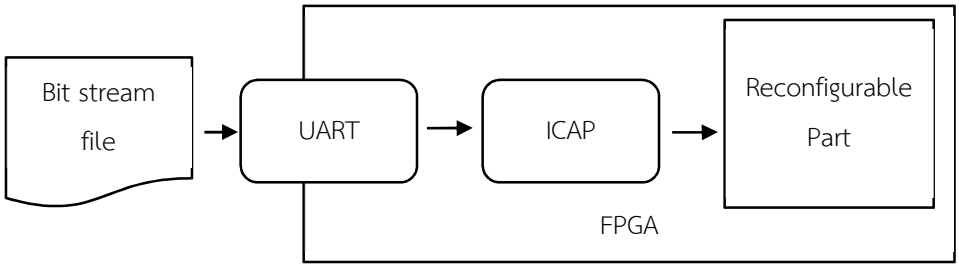
ตารางที่ 4-1 ตัวแปรแหล่งจ่ายแบ่งตามประเภทของทรัพยากร

ชื่อตัวแปร	ประเภททรัพยากร
Vccint	■ Control Logic Blocks (CLBs) ทั้งหมด ■ เส้นสัญญาณ (Signal routing) ทั้งหมด
Vccaux	■ โมดูลจัดการสัญญาณนาฬิกา เช่น PLL,DCM
VccoXX	■ Output buffer ทั้งหมดโดย XX = Voltage IO (1.8, 2.5)

4.1.3 เวลาที่ใช้โปรแกรมวงจร (Configuration Time)

เวลาที่ใช้ในการโปรแกรมไฟล์วงจร (Bitstream) พิจารณาเฉพาะส่วนของการทำ Partial reconfiguration ซึ่งจะเกิดขึ้นหลังจากที่ได้โปรแกรมวงจรส่วนของ Static part วิธีการโปรแกรมในพื้นที่ Reconfigurable part มีรูปแบบการโปรแกรมวงจร 2 รูปแบบ คือ การโปรแกรมวงจรโดยเชื่อมต่อเอฟพีจีเอด้วยสายสัญญาณ RS-232 และ การโปรแกรมวงจรโดยเชื่อมต่อผ่าน Xbee

(1) การโปรแกรมวงจรโดยเชื่อมต่อเอฟพีจีเอด้วยสายสัญญาณ RS-232 มีลำดับการไหลของไฟล์วงจรดังภาพประกอบ 4-1 เริ่มต้น โปรแกรมไฟล์วงจร (Bitstream file) ถูกส่งจากคอมพิวเตอร์ผ่านทาง UART โดยรับส่งข้อมูลแบบ Serial ที่อัตราการรับส่งข้อมูล Baud rate เท่ากับ 115,200 บิต ต่อวินาที (bps) ดังนั้น ใน 1 วินาที สามารถส่งข้อมูลได้ 115,200 บิต ทำให้แบนด์วิดท์ของ UART คือ 115,200 bps (115.2 kbps)



ภาพประกอบ 4-1 ลำดับการโปรแกรมวงจรโดยสายสัญญาณ RS-232

ลำดับถัดมาข้อมูลส่งต่อให้กับโมดูล ICAP เพื่อโปรแกรมลงพื้นที่ Reconfigurable part คุณสมบัติของ ICAP module คือ รับส่งข้อมูลแบบขนานขนาด 16 บิต อัตราการรับส่งข้อมูลขึ้นอยู่กับความถี่สัญญาณนาฬิกาที่ทำงาน ซึ่งในงานวิจัยนี้ใช้ความถี่ 25 MHz ทำให้อัตราการรับส่งข้อมูลของ ICAP อยู่ที่ 25 MHz x 16 = 400 Mbps ตารางที่ 4-2 สรุปรายงานขนาดข้อมูลที่รับส่ง และอัตราการรับส่งข้อมูลของ UART และ ICAP

ตารางที่ 4-2 ขนาดและอัตราการรับส่งข้อมูลแต่ละโมดูล

โมดูลวงจร	ขนาดข้อมูล (บิต)	แบนด์วิดท์
UART module	1	115.2 Kbps
ICAP module (25 MHz)	16	400 Mbps

ดังนั้นสามารถคำนวณเวลาในการโปรแกรมวงจรลงเอฟพีจีเอได้โดยเวลาที่ใช้ทั้งหมดในการโปรแกรมเท่ากับผลรวมของเวลาแต่ละส่วนดังนี้

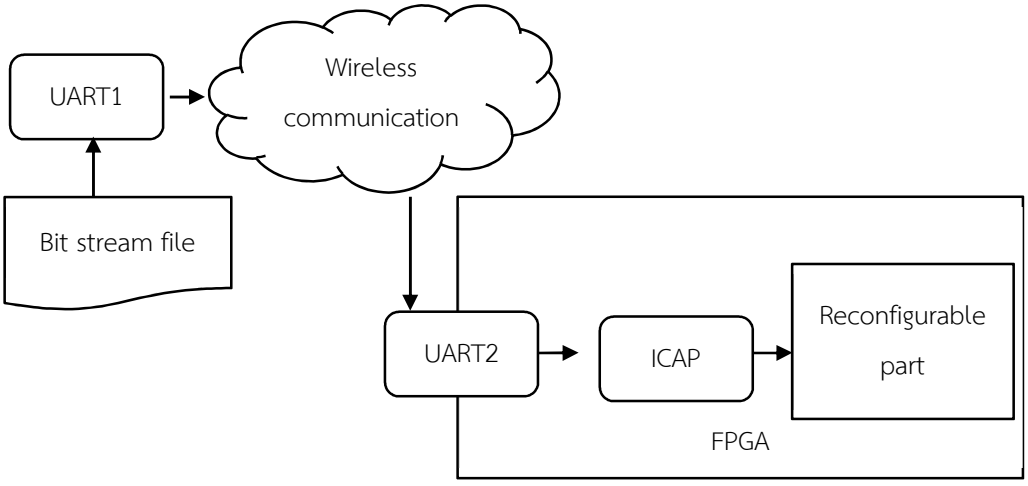
- ระยะเวลาส่งข้อมูลจาก UART ไปยัง ICAP

$$\text{ระยะเวลา}_{uart\_icap}(s) = \frac{\text{ขนาดไฟล์ (bit)}}{\text{อัตราการรับส่งข้อมูล (bps)}} = \frac{\text{ขนาดไฟล์ (bit)}}{115.2\ \text{Kbps}}$$

- ระยะเวลาส่งข้อมูลจาก ICAP แล้วโปรแกรมลง Reconfigurable part

$$\text{ระยะเวลา}_{Icap}(s) = \frac{\text{ขนาดไฟล์ (bit)}}{\text{อัตราการรับส่งข้อมูล (bps)}} = \frac{\text{ขนาดไฟล์ (bit)}}{400\ \text{Mbps}}$$

(2) การโปรแกรมวงจรโดยเชื่อมต่อผ่าน Xbee ลำดับการส่งข้อมูลแสดงดังภาพประกอบ 4-2 เริ่มต้นจาก Bitstream file ส่งข้อมูลจากคอมพิวเตอร์ผ่านทาง UART1 เข้าสู่ระบบรับส่งข้อมูลไร้สาย (Wireless Communication) ในที่นี้ คือ มาตรฐานการส่งข้อมูลแบบ ZigBee protocol จากนั้นข้อมูลถูกส่งต่อให้กับ UART2 รับข้อมูลแล้วส่งต่อไปโมดูล ICAP เพื่อโปรแกรมลงพื้นที่ Reconfigurable part



ภาพประกอบ 4-2 ลำดับการโปรแกรมเอฟพีจีเอแบบไร้สาย

อัตราการรับส่งข้อมูลของแต่ละโมดูลแสดงดังตารางที่ 4-3 UART1 และ UART2 สามารถรับส่งได้ที่อัตรา 9,600 bps สำหรับ Zigbee สามารถรับส่งได้ไม่เกิน 250 Kbps ข้อจำกัด UART1, UART2 ไม่สามารถใช้อัตราการส่งข้อมูลมากกว่า 9,600 bps ได้เพราะจะทำให้ข้อมูลบางส่วนหายไป เนื่องจากข้อจำกัดของอัตราการรับส่งข้อมูลด้วย Zigbee หากใช้งาน UART1, UART2 ที่อัตราเร็วสูงกว่า 9,600 bps ส่งผลให้ข้อมูลบางส่วนสูญหายเพราะเมื่อ UART1

ส่งข้อมูลมาแล้ว Zigbee ไม่สามารถรับและส่งต่อข้อมูลให้ UART2 ได้ทัน ดังนั้นอัตราการรับส่งข้อมูล UART1 และ UART2 ถูกจำกัดให้ใช้งานได้สูงสุด 9,600 bps เท่านั้น

ตารางที่ 4-3 ขนาดและอัตราการรับส่งข้อมูลแต่ละโมดูล

โมดูลวงจร	ขนาดข้อมูล (บิต)	แบนด์วิดท์
UART1 module	1	9,600 bps
Wireless communication	1	< 250 Kbps
UART2 module	1	9,600 bps
ICAP module (25 MHz)	16	400 Mbps

สามารถคำนวณเวลาในการโปรแกรมวงจรได้โดยเวลาที่ใช้ทั้งหมดในการโปรแกรมเท่ากับผลรวมของเวลาจาก 2 ส่วนดังนี้

- ระยะเวลาส่งข้อมูลจาก UART1 ไปยังเครือข่ายไร้สาย และส่งต่อไปยัง UART2

$$\text{ระยะเวลา}_{\text{Uart\_wireless}}(s) = \frac{\text{ขนาดไฟล์ (bit)}}{\text{อัตราการรับส่งข้อมูล (bps)}} = \frac{\text{ขนาดไฟล์ (bit)}}{9{,}600\ \text{bps}}$$

- ระยะเวลาส่งข้อมูลจาก ICAP แล้วโปรแกรมลง Reconfigurable part

$$\text{ระยะเวลา}_{\text{Icap}}(s) = \frac{\text{ขนาดไฟล์ (bit)}}{\text{อัตราการรับส่งข้อมูล (bps)}} = \frac{\text{ขนาดไฟล์ (bit)}}{400\ \text{Mbps}}$$

4.2 เครื่องมือในการทดสอบ (Tools)

เครื่องมือสำหรับการออกแบบและทดสอบแบ่งประเภทเป็นสองส่วน คือ ฮาร์ดแวร์ ได้แก่ เอฟพีจีเอตระกูล Spartan-6, บอร์ด XBee สำหรับการสื่อสารไร้สาย (Wireless) ส่วนซอฟต์แวร์สำหรับออกแบบฮาร์ดแวร์ในเอฟพีจีเอ คือ Xilinx ISE, GoAhead

4.2.1 บอร์ดทดลองเอฟพีจีเอ

การทดสอบการทำงานใช้บอร์ดเอฟพีจีเอ Spartan-6 SP601 [21] ซึ่งเป็นบอร์ดประเภท Evaluation kit โดยใช้เอฟพีจีเอรุ่น SPARTAN-6 LX16 CSG324-2C มีขนาด Logic cells ประมาณ 16,000 Cells และประมาณ 2,300 SLICE คุณสมบัติสำคัญของบอร์ด SP601 ได้แก่

- JTAG USB configuration
- DDR2 memory 128 MB
- Serial (UART) to USB
- 200 MHz oscillator
- 4 LEDs, 4 push buttons, 8 Bits I/O pin

4.2.2 บอร์ด XBee

โมดูลบอร์ด XBee ประกอบไปด้วยส่วนหลัก คือ ไมโครคอนโทรลเลอร์และวงจรที่ใช้ Radio Frequency IC ทำหน้าที่เป็นตัวรับส่งสัญญาณในช่วงความถี่ 2.4 GHz ตามมาตรฐาน Zigbee protocol และมีอินเตอเฟสสำหรับเชื่อมต่อกับอุปกรณ์อื่น ๆ ผ่านทาง UART งานวิจัยนี้ใช้ XBee module รุ่น XBee PRO S28 เป็น XBee series 2 เพิ่มฟังก์ชันการทำ Mesh network โดยรุ่นนี้ไม่สามารถเขียนโปรแกรมเพิ่มได้ สามารถใช้ได้เฉพาะการกำหนด Parameter เกี่ยวกับระบบเครือข่ายเท่านั้น

4.2.3 ซอฟต์แวร์ Xilinx ISE

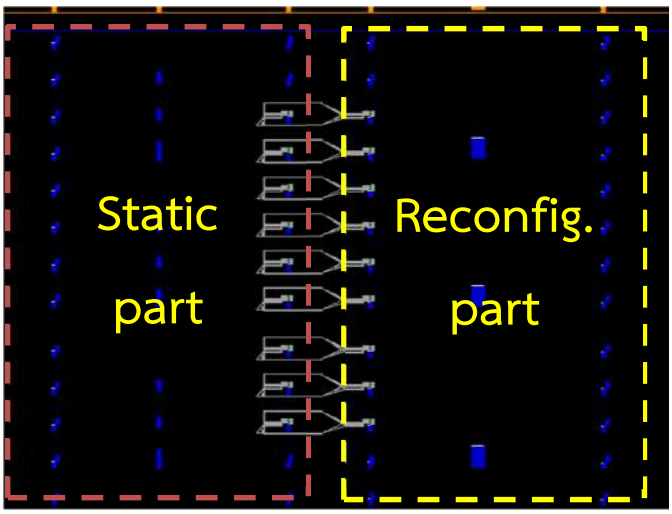
Xilinx ISE คือ ซอฟต์แวร์ของ Xilinx ซึ่งเป็นผู้ผลิตเอฟพีจีเอ เพื่อให้ผู้ใช้งานเอฟพีจีเอ สามารถออกแบบวงจรฮาร์ดแวร์โดยการเขียนซอฟต์แวร์ด้วยภาษา Hardware Description Language (HDL) ได้แก่ VHDL, Verilog ซอฟต์แวร์ Xilinx ISE ทำการเปลี่ยนจากโปรแกรมที่เขียนด้วย HDL เป็นข้อมูลที่สามารถสร้างเป็นวงจรลงในเอฟพีจีเอ นอกจากนี้งานวิจัยได้รายงานผลการทดสอบโดยใช้ซอฟต์แวร์ย่อย คือ Xilinx xpower analyzer เพื่อคำนวณการสูญเสียพลังงานของวงจรที่โปรแกรมลงเอฟพีจีเอ และ Xilinx FPGA editor เพื่อเรียกดูการวางตำแหน่งและการเชื่อมต่อระหว่างวงจรภายในเอฟพีจีเอ

4.2.4 ซอฟต์แวร์ GoAhead

ซอฟต์แวร์ GoAhead เป็นซอฟต์แวร์ที่ออกแบบโดยงานวิจัยของมหาวิทยาลัย University of Oslo (UiO) เพื่อใช้ในกระบวนการโปรแกรมวงจรแบบ Partial reconfiguration โดยซอฟต์แวร์สามารถแสดงผลเป็น Graphic User Interface (GUI) ให้ผู้ใช้เห็นโครงสร้างของเอฟพีจีเอ และสามารถกำหนดคุณสมบัติการทำ Partial reconfiguration ได้

4.3 ผลทดสอบการออกแบบ Bus Macro

การออกแบบส่วนเชื่อมต่อระหว่างพื้นที่ Static และ Reconfigurable part หรือ Bus macro จะวิเคราะห์จากความต้องการใช้งานอินพุตเอาต์พุตของวงจรในพื้นที่ Reconfigurable part เพื่อให้เกิดการใช้งานทรัพยากรได้อย่างเหมาะสม ข้อกำหนดของการเลือก Bus macro คือ ต้องเลือกตำแหน่ง SLICE บริเวณขอบของพื้นที่ Reconfigurable part สำหรับงานวิจัยนี้ต้องการออกแบบให้ Bus macro มีขนาด 36 บิต กล่าวคือ มีอินพุต 36 บิต และเอาต์พุต 36 บิต ดังนั้นต้องกำหนด CLB จำนวน 9 คู่ (CLB 1 คู่ ได้ 4 อินพุต 4 เอาต์พุต) โดยเลือก SLICE ที่อยู่ติดกันในแนวแกนตั้ง (Y axis) และใช้ SLICE 2 ชุด ที่ติดกันในแนวแกนนอน (X axis) เพื่อทำเป็น 1 คู่ เช่น เลือก SLICE ตำแหน่งที่ X11Y13 ก็จะได้คู่กับ SLICE ตำแหน่งที่ X13Y13 และเมื่อนำไปใช้งาน SLICE ตำแหน่งที่ X11Y13 จะอยู่ใน Static part ส่วน SLICE ตำแหน่งที่ X13Y13 จะอยู่ใน Reconfigurable part สำหรับ Bus macro ชุดต่อมาก็จะเลือก SLICE คู่ถัดไปในแนวแกนตั้ง (Y axis) คือ SLICE ตำแหน่งที่ X11Y34 และ SLICE ตำแหน่งที่ X13Y34 จนครบ 9 ชุด CLB ทำให้ได้ Bus macro ขนาด 36 บิต ได้ผลทดสอบการวางตำแหน่งหลังจากออกแบบ Bus macro ดังภาพประกอบ 4-3



ภาพประกอบ 4-3 Bus Macro ขนาด 36 บิต

จากภาพประกอบ 4-3 เป็นผลการวางวงจรของ Bus macro ขนาด 36 บิต ซึ่งพบว่าในพื้นที่ Static part ประกอบไปด้วย CLB จำนวน 9 ชุด มีเส้นการเชื่อมต่อไปยัง CLB 9 ชุด ในพื้นที่ Reconfigurable part โมดูลดังกล่าวจะถูกนำไปใช้งานเป็นส่วนหนึ่งของการออกแบบไฟล์วงจรทั้ง Static part และ Reconfigurable part

หัวข้อต่อไปกล่าวถึงรายงานผลการทดสอบโดยแบ่งตามหัวข้อดังนี้

(1) รายงานผลการทดสอบบนเอฟพีจีเอของพื้นที่ Static Part

(2) รายงานผลการทดสอบบนเอฟพีจีเอของพื้นที่ Reconfigurable Part

4.4 ผลการทดสอบบนเอฟพีจีเอของพื้นที่ Static Part

การทดสอบความถูกต้องกระบวนการออกแบบไฟล์วงจรพื้นที่ Static part ผลลัพธ์ที่ได้ คือ ข้อมูลจากไฟล์วงจร Native Circuit Description (NCD) ซึ่งสามารถพิจารณาลักษณะการวางตำแหน่งของวงจรและเส้นทางเชื่อมต่อวงจรในเอฟพีจีเอ ผลลัพธ์ที่ถูกต้องของกระบวนการออกแบบไฟล์วงจร คือ พื้นที่ในเอฟพีจีเอต้องแบ่งเป็น 2 ส่วนประกอบไปด้วยพื้นที่ Static part และ Reconfigurable part โดยพื้นที่ Static part ประกอบไปด้วยวงจรควบคุมการโปรแกรมพื้นที่ Reconfigurable part ดังที่ได้กล่าวไว้แล้วในบทที่ 3 และพื้นที่ Static part ไม่สามารถสร้างเส้นเชื่อมต่อหรือวางวงจรในตำแหน่งพื้นที่ Reconfigurable part ได้ งานวิจัยทดสอบกระบวนการออกแบบไฟล์วงจรพื้นที่ Static part 2 แบบ คือ

(1) ผลทดสอบกระบวนกการออกแบบไฟล์วงจรพื้นที่ Static Part แบบปกติ

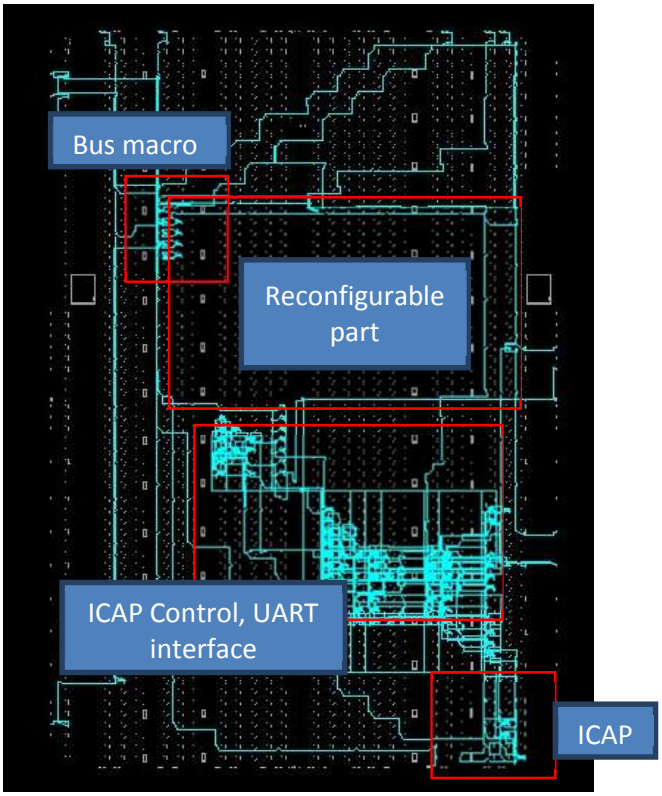
(2) ผลทดสอบกระบวนการออกแบบไฟล์วงจรพื้นที่ Static Part ที่มีการใช้งานไมโครโพรเซสเซอร์

4.4.1 ผลทดสอบกระบวนการออกแบบไฟล์วงจรพื้นที่ Static Part แบบปกติ

ผลการทดสอบความถูกต้องของกระบวนการออกแบบไฟล์วงจรพื้นที่ Static part แบบปกติประกอบไปด้วย ผลการวางตำแหน่งวงจรและเส้นทางเชื่อมต่อ, ผลการใช้ทรัพยากรเอฟพีจีเอ และผลทดสอบค่ากำลังงานสูญเสีย รายละเอียดมีดังนี้

4.4.1.1 ผลการวางตำแหน่งวงจรและเส้นทางเชื่อมต่อในเอฟพีจีเอ

การวางตำแหน่งของวงจรและเส้นทางเชื่อมต่อวงจรในเอฟพีจีเอ มีผลลัพธ์ดังภาพประกอบ 4-4 ซึ่งพบว่าพื้นที่ Static part สามารถวางตำแหน่งวงจรได้ถูกต้อง เนื่องจากผู้วิจัยได้กำหนดให้พื้นที่ Reconfigurable part อยู่บริเวณกลางของพื้นที่ทั้งหมด สังเกตได้ว่าพื้นที่ดังกล่าวจะไม่ถูกลากเส้นหรือวางวงจรใด ๆ และได้กำหนดให้พื้นที่ซ้ายมือติดกับ Reconfigurable part คือ Bus macro เห็นได้ว่า Bus macro อยู่ระหว่างพื้นที่ Static part และ Reconfigurable part



ภาพประกอบ 4-4 โครงสร้างวงจรพื้นที่ Static Part ในเอฟพีจีเอ Spartan-6

พื้นที่ถัดจาก Reconfigurable part คือ โมดูล ICAP Control ซึ่งเชื่อมต่อกับ UART interface และพื้นที่ส่วนสุดท้าย คือ โมดูล ICAP อยู่มุมขวาล่างของเอฟพีจีเอเนื่องจาก ICAP เป็นโมดูลวงจรที่ผู้ผลิต Xilinx กำหนดมาโดยกำหนดให้อยู่ในพื้นที่ดังกล่าว

4.4.1.2 ผลการใช้ทรัพยากร (Resource Utilization)

ผลทดสอบการออกแบบวงจรในพื้นที่ Static part ใช้ทรัพยากรดังตารางที่ 4-4 ประกอบไปด้วยจำนวนการใช้งานและสัดส่วนการใช้งานของทรัพยากรแต่ละประเภท

ตารางที่ 4-4 การใช้ทรัพยากรเอฟพีจีเอ Spartan-6 LX16 ของพื้นที่ Static Part

ทรัพยากรเอฟพีจีเอ	จำนวนที่ใช้งาน	จำนวนทั้งหมด	สัดส่วนการใช้งาน (%)
Clocks	3		
Logic	346	9112	4
Register	260	18224	1
IOs	16	232	7

จากตารางที่ 4-4 เป็นผลการใช้ทรัพยากรเอฟพีจีเอของวงจรพื้นที่ Static part ซึ่งพบว่าสัดส่วนการใช้งานทรัพยากรแต่ละประเภทไม่เกิน 10% โดยเฉพาะทรัพยากร SLICE logic ใช้งานเท่ากับ 4% ดังนั้นการออกแบบวงจรพื้นที่ Static part สูญเสียพื้นที่และทรัพยากรเอฟพีจีเอเพียงเล็กน้อยซึ่งทำให้เหลือพื้นที่สำหรับใช้งานเป็นพื้นที่ Reconfigurable part อย่างเพียงพอที่จะรองรับกับวงจรขนาดใหญ่ได้

4.4.1.3 ผลทดสอบกำลังงานสูญเสีย (Power Consumption)

ผลทดสอบได้ค่ากำลังงานสูญเสีย (Power consumption) ของวงจรในพื้นที่ Static part และรายละเอียดกำลังงานสูญเสียสำหรับแหล่งจ่ายแต่ละประเภทแสดงดังตารางที่ 4-5

ตารางที่ 4-5 กำลังงานสูญเสีย (Power Consumption) ของเอฟพีจีเอ Spartan-6 LX16

แหล่งจ่าย	แรงดัน (V)	กำลังงาน (W)	กระแสไฟ (A)
Vccint	1.2	0.0084	0.007
Vccaux	2.5	0.0075	0.003
Vcco25	2.5	0.005	0.002
Vcco18	1.8	0.002	0.002
ผลรวมกำลังงานสูญเสีย		0.024	

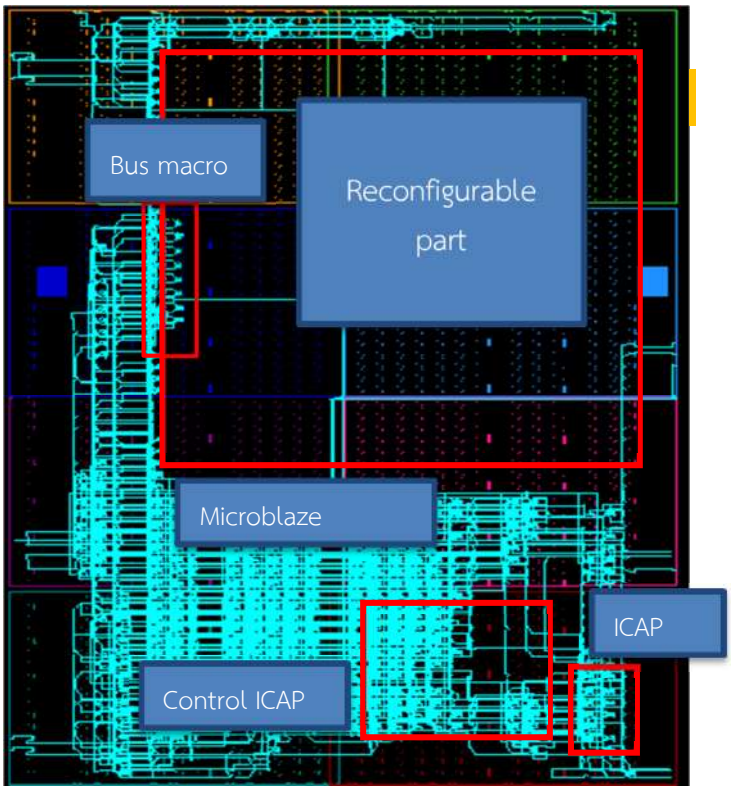
จากตารางที่ 4-5 เป็นผลการทดสอบกำลังงานสูญเสียของวงจรพื้นที่ Static part พบว่ากำลังงานสูญเสียรวมเท่ากับ 24 มิลลิวัตต์ ดังนั้นการออกแบบวงจรพื้นที่ Static part ไม่ได้ใช้กำลังงานสูญเสียมากเกินไปจนต้องคำนึงถึงผลกระทบของกำลังงานสูญเสียทั้งหมด

4.4.2 ผลทดสอบกระบวนการออกแบบไฟล์วงจรพื้นที่ Static Part ที่มีการใช้งานไมโครโพรเซสเซอร์

ผลการทดสอบความถูกต้องของกระบวนการออกแบบไฟล์วงจรพื้นที่ Static part ที่มีไมโครโพรเซสเซอร์ Microblaze ประกอบไปด้วย ผลการวางตำแหน่งวงจรและเส้นทางเชื่อมต่อ, ผลการใช้ทรัพยากรเอฟพีจีเอ และผลทดสอบค่ากำลังงานสูญเสีย รายละเอียดมีดังนี้

4.4.2.1 ผลการวางตำแหน่งวงจรและเส้นทางเชื่อมต่อในเอฟพีจีเอ

การวางตำแหน่งของวงจรและเส้นทางเชื่อมต่อวงจรในเอฟพีจีเอ มีผลลัพธ์ดังภาพประกอบ 4-5 ซึ่งพบว่าพื้นที่ Static part สามารถวางตำแหน่งวงจรได้ถูกต้อง สังเกตได้ว่าพื้นที่ Reconfigurable part ไม่ถูกลากเส้นหรือวางวงจรใด ๆ และได้กำหนดให้พื้นที่ซ้ายมือติดกับ Reconfigurable part คือ Bus macro เห็นได้ว่า Bus macro อยู่ระหว่างพื้นที่ Static part และ Reconfigurable part



ภาพประกอบ 4-5 โครงสร้าง Static Part ที่เพิ่ม Microblaze ในเอฟพีจีเอ Spartan-6

พื้นที่ถัดลงมาจาก Reconfigurable part ถูกกำหนดให้เป็น โมดูล Control ICAP ซึ่งเชื่อมต่อกับโมดูล ICAP บริเวณมุมขวาล่าง และพื้นที่ที่เหลือกำหนดเป็นไมโครโพรเซสเซอร์ (Microblaze) ทั้งหมด

4.4.2.2 ผลการใช้ทรัพยากร (Resource Utilization)

การออกแบบวงจรในพื้นที่ Static part ซึ่งมีไมโครโพรเซสเซอร์ (Microblaze) ได้ผลการใช้งานทรัพยากร (Resource) แสดงดังตารางที่ 4-6

ตารางที่ 4-6 การใช้ทรัพยากรเอฟพีจีเอ Spartan-6 LX16 ของพื้นที่ Static Part กับโพรเซสเซอร์

ทรัพยากรเอฟพีจีเอ	**จำนวนที่ใช้งาน**	**จำนวนทั้งหมด**	**สัดส่วนการใช้งาน (%)**
Clocks	4		
Logic	1,538	9,112	16
Register	1,738	1,8224	9
Memory	158	2,176	7
IOs	16	232	7
PLLs	1	2	50

จากตารางที่ 4-6 เป็นผลการใช้ทรัพยากรในพื้นที่ Static part ที่มีการเรียกใช้ Microblaze พบว่ามีสัดส่วนการใช้งานทรัพยากรโดยเฉพาะ SLICE logic เท่ากับ 16% ดังนั้น การออกแบบให้พื้นที่ Static part มีการใช้งาน Microblaze ทำให้ใช้ทรัพยากรมากขึ้นแต่ยังคงเหลือทรัพยากรกว่า 80% สำหรับการออกแบบเป็นพื้นที่ Reconfigurable part

4.4.2.3 ผลทดสอบกำลังงานสูญเสีย (Power Consumption)

ผลทดสอบกำลังงานสูญเสีย (Power consumption) ของวงจรในพื้นที่ Static part กับไมโครโพรเซสเซอร์ (Microblaze) โดยรายละเอียดกำลังงานสูญเสียของแหล่งจ่ายแต่ละประเภทแสดงดังตารางที่ 4-7 พบว่ากำลังงานสูญเสียรวมเท่ากับ 192 มิลลิวัตต์

ตารางที่ 4-7 กำลังงานสูญเสีย (Power Consumption) ของ Static Part กับ ไมโครโพรเซสเซอร์

แหล่งจ่าย	**แรงดัน (V)**	**กำลังงาน (W)**	**กระแสไฟ (A)**
Vccint	1.2	0.036	0.030
Vccaux	2.5	0.12	0.048
Vcco25	2.5	0.0325	0.013
Vcco18	1.8	0.0036	0.002
ผลรวมกำลังงานสูญเสีย		0.192	

จะเห็นได้ว่ากระบวนการออกแบบไฟล์วงจรที่ได้ออกแบบไว้ทั้งแบบปกติและแบบที่มีไมโครโพรเซสเซอร์ร่วมด้วย ผลการทำงานสามารถทำงานได้ถูกต้อง ผลลัพธ์จากไฟล์วงจรที่ได้ทำให้เห็นการแบ่งพื้นที่ของเอฟพีจีเอซึ่งพบว่าพื้นที่ Reconfigurable part จะได้เป็นพื้นที่โล่งไม่มีวงจรใด ๆ การออกแบบพื้นที่ Static part เมื่อมีการนำไมโครโพรเซสเซอร์มาร่วมด้วย ทำให้เพิ่มความสามารถของพื้นที่ Static part ให้มีความยืดหยุ่นกับการนำไปใช้งานมากขึ้น

อย่างไรก็ตาม การนำไมโครโพรเซสเซอร์มาใช้ร่วมด้วยทำให้มีการใช้ทรัพยากรมากขึ้นส่งผลให้พื้นที่สำหรับ Reconfigurable part ลดลง และมีค่ากำลังงานสูญเสียเพิ่มขึ้นเช่นกัน

4.5 การทดสอบเอฟพีจีเอของพื้นที่ Reconfigurable Part

การทดสอบความถูกต้องกระบวนการออกแบบไฟล์วงจรพื้นที่ Reconfigurable ผลลัพธ์ที่ได้ คือ ข้อมูลจากไฟล์วงจร Native Circuit Description (NCD) ซึ่งสามารถพิจารณาลักษณะการวางตำแหน่งของวงจรและเส้นทางเชื่อมต่อวงจรในเอฟพีจีเอ ผลลัพธ์ที่ถูกต้องของกระบวนการออกแบบไฟล์วงจร คือ วงจรทั้งหมดที่ออกแบบต้องอยู่ในพื้นที่ Reconfigurable part ที่กำหนดไว้เท่านั้น และเมื่อโปรแกรมวงจรลงเอฟพีจีเอแล้วต้องสามารถทำงานได้ถูกต้อง และสามารถโปรแกรมซ้ำได้โดยพื้นที่ Static part ยังคงทำงานอยู่

งานวิจัยเริ่มต้นทดสอบการทำงานของพื้นที่ Reconfigurable part โดยออกแบบวงจรอย่างง่าย คือ วงจร AND gate 2 ชุด และวงจร OR gate 2 ชุด โดยรับอินพุตจากปุ่ม Button switch และเอาต์พุตต่อกับหลอด LED ผลการทดสอบ คือ เริ่มต้นทำงานด้วยวงจร AND gate สังเกตผลลัพธ์ที่ LED ซึ่งผลการทำงานถูกต้อง จากนั้นเพื่อทดสอบว่าระบบสามารถโปรแกรมวงจรใหม่ได้โดยไม่จำเป็นต้องหยุดการทำงานทั้งหมด จึงโปรแกรมวงจร OR gate ลงพื้นที่ Reconfigurable part ผลลัพธ์ คือ เอฟพีจีเอเปลี่ยนจากวงจร AND gate เป็นวงจร OR gate อย่างถูกต้อง จากนั้นงานวิจัยทดสอบกระบวนการออกแบบไฟล์วงจรพื้นที่ Reconfigurable part แบ่งเป็นการทดสอบดังนี้

(1) ทดสอบกระบวนการออกแบบไฟล์วงจรพื้นที่ Reconfigurable Part ด้วยวงจรบวกอย่างง่าย

(2) ทดสอบกระบวนการออกแบบไฟล์วงจรพื้นที่ Reconfigurable Part ด้วยวงจร Advanced Encryption Standard (AES)

4.5.1 ผลทดสอบกระบวนการออกแบบไฟล์วงจรพื้นที่ Reconfigurable Part ด้วยวงจรบวก (Adder 8 Bits)

วงจรบวกเลข 8 บิต (Adder 8 bits) ภายในวงจรประกอบไปด้วย Full adder จำนวน 8 ชุด ผลการทดสอบความถูกต้องของกระบวนการออกแบบไฟล์วงจรพื้นที่ Reconfigurable part ประกอบไปด้วย ผลการวางตำแหน่งวงจรและเส้นทางเชื่อมต่อ, ผลการใช้ทรัพยากรเอฟพีจีเอ และผลทดสอบค่ากำลังงานสูญเสีย พร้อมกับเปรียบเทียบผลทดสอบกับการออกแบบไฟล์วงจรแบบปกติ (Full reconfiguration) รายละเอียดมีดังนี้

4.5.1.1 ผลทดสอบการใช้ทรัพยากรของเอฟพีจีเอ (Resource Utilization)

ผลการใช้ทรัพยากรของวงจรบวกที่ใช้กระบวนการออกแบบไฟล์วงจรแบบปกติ Full reconfiguration และกระบวนการออกแบบไฟล์วงจรแบบ Partial reconfiguration แสดงดังตารางที่ 4-8

ตารางที่ 4-8 การใช้ทรัพยากรเอฟพีจีเอของวงจรบวก 8 บิต

ทรัพยากรเอฟพีจีเอ	จำนวนทรัพยากรที่ใช้งาน (%)	
	Full reconfiguration	Partial reconfiguration
Logic LUTs	8 (0.08%)	32 (0.35%)

จากตารางที่ 4-8 เป็นผลทดสอบการใช้ทรัพยากรของวงจรบวก 8 บิต พบว่ากระบวนการออกแบบไฟล์วงจรแบบ Partial reconfiguration ใช้จำนวนทรัพยากรมากกว่ากระบวนการปกติ เนื่องมาจากทรัพยากรบางส่วนของพื้นที่ Reconfigurable part ถูกนำไปใช้เป็น Bus macro สำหรับเชื่อมต่อพื้นที่ Static part และ Reconfigurable part

4.5.1.2 ผลทดสอบกำลังงานสูญเสีย (Power consumption)

กำลังงานสูญเสียของวงจรบวก 8 บิต ที่ใช้กระบวนการออกแบบไฟล์วงจรแบบปกติ Full reconfiguration และกระบวนการออกแบบไฟล์วงจรแบบ Partial reconfiguration แสดงดังตารางที่ 4-9

ตารางที่ 4-9 กำลังงานสูญเสียวงจรบวก 8 บิต

แหล่งจ่าย	กำลังงานสูญเสีย (W)	
	Full reconfiguration	Partial reconfiguration
Vccint (1.2V)	0.0072	0.0072
Vccaux (2.5V)	0.0075	0.0075
ผลรวมกำลังงานสูญเสีย	0.0147	0.0147

จากตารางที่ 4-9 เป็นผลทดสอบกำลังงานสูญเสียของวงจรบวก 8 บิต พบว่ากระบวนการออกแบบไฟล์วงจรแบบ Full reconfiguration และกระบวนการออกแบบ Partial reconfiguration ได้วงจรที่ใช้กำลังงานสูญเสียเท่ากัน คือ 14.7 มิลลิวัตต์ (mW)

จะเห็นได้ว่ากระบวนการออกแบบไฟล์วงจรแบบ Partial reconfiguration พื้นที่ Reconfigurable part มีการใช้ทรัพยากรเอฟพีจีเอมากกว่าการออกแบบไฟล์วงจรแบบ Full reconfiguration เนื่องจากต้องใช้ทรัพยากรบางส่วนเป็นวงจร Bus macro เพื่อรับส่งข้อมูลระหว่างพื้นที่ Static part และ Reconfigurable part อย่างไรก็ตาม ผลทดสอบค่ากำลังงานสูญเสียของวงจรที่ออกแบบโดยทั้ง 2 วิธี พบว่าค่ากำลังงานสูญเสียเท่ากัน ดังนั้นกระบวนการออกแบบไฟล์วงจรแบบ Partial reconfiguration แม้ว่าทรัพยากรที่ใช้เพิ่มขึ้นแต่ทรัพยากรเหล่านั้น ไม่ส่งผลกระทบกับกำลังงานสูญเสีย

4.5.2 ผลการทดสอบกระบวนการออกแบบไฟล์วงจรพื้นที่ Reconfigurable Part ด้วยวงจร Advanced Encryption Standard (AES)

เพื่อทดสอบความถูกต้องของกระบวนการออกแบบไฟล์วงจรแบบ Partial reconfiguration พื้นที่ Reconfigurable part จึงได้ทดสอบการทำงานด้วยวงจรที่มีความซับซ้อนมากขึ้น ผู้วิจัยนำเสนอแนวคิดการออกแบบเซนเซอร์โนดในระบบเครือข่ายเซนเซอร์ไร้สายโดยใช้เซนเซอร์โนดที่สามารถปรับโครงสร้างฮาร์ดแวร์ตัวเองได้

เนื่องจากระบบเครือข่ายเซนเซอร์ไร้สายถูกนำไปใช้อย่างกว้างขวางและในสภาวะแวดล้อมหลายประเภท เช่น เครือข่ายเซนเซอร์เฝ้าดูสภาวะใต้น้ำ, เครือข่ายเซนเซอร์เฝ้าดูสภาวะพื้นที่ป่าขนาดใหญ่ เหล่านี้เป็นต้น ซึ่งเห็นได้ว่าในสภาวะแวดล้อมเหล่านั้น จะเกิดความยุ่งยากหากระบบ มีปัญหา เช่น ระบบเครือข่ายขัดข้อง หรือเซนเซอร์โนดเสียหาย หรือมีความต้องการปรับระบบให้เหมาะสมกับสภาพแวดล้อม ดังนั้นเซนเซอร์โนดที่ปรับเปลี่ยนโครงสร้างให้เหมาะสมกับสภาวะแวดล้อมได้ผ่านระบบเครือข่ายไร้สายจะเป็นแนวทางในการพัฒนาเซนเซอร์โนดในอนาคต

ผู้วิจัยให้ความสนใจเรื่องของความปลอดภัยในการรับส่งข้อมูลของเครือข่ายเซนเซอร์ไร้สายจึงนำแนวคิดเซนเซอร์โนดที่สามารถปรับเปลี่ยนโครงสร้างฮาร์ดแวร์ โดยกำหนดให้มีระบบความปลอดภัยในการรับส่งข้อมูลโดยใช้วงจรเข้ารหัสข้อมูลให้รองรับการใช้งาน AES ขนาดคีย์ 128 และ 192 บิต เอฟพีจีเอสามารถเลือกใช้วงจร AES แต่ละแบบได้ในขณะที่กำลังทำงานทำให้สามารถปรับเปลี่ยนได้ตามสภาวะแวดล้อมขณะนั้นเช่น

(1) เมื่อระบบอยู่ในสภาวะที่ต้องการความปลอดภัยของข้อมูล สามารถเปลี่ยนการเข้ารหัสเป็นแบบ AES 192 บิต เพื่อให้ข้อมูลมีความปลอดภัยกว่า AES 128 บิต

(2) เมื่อระบบอยู่ในสภาวะที่ต้องส่งข้อมูลมากขึ้น การเข้ารหัสข้อมูลที่ซับซ้อนทำให้ใช้เวลารับส่งข้อมูลนานขึ้นรวมถึงอาจส่งผลต่อพลังงานที่สูญเสียมากขึ้น ดังนั้นสามารถเปลี่ยนวงจรเข้ารหัสให้เหมาะสมเช่น จากเดิมใช้วงจร AES 192 บิต เปลี่ยนเป็นวงจร AES 128 บิต เพื่อแก้ปัญหาดังกล่าว

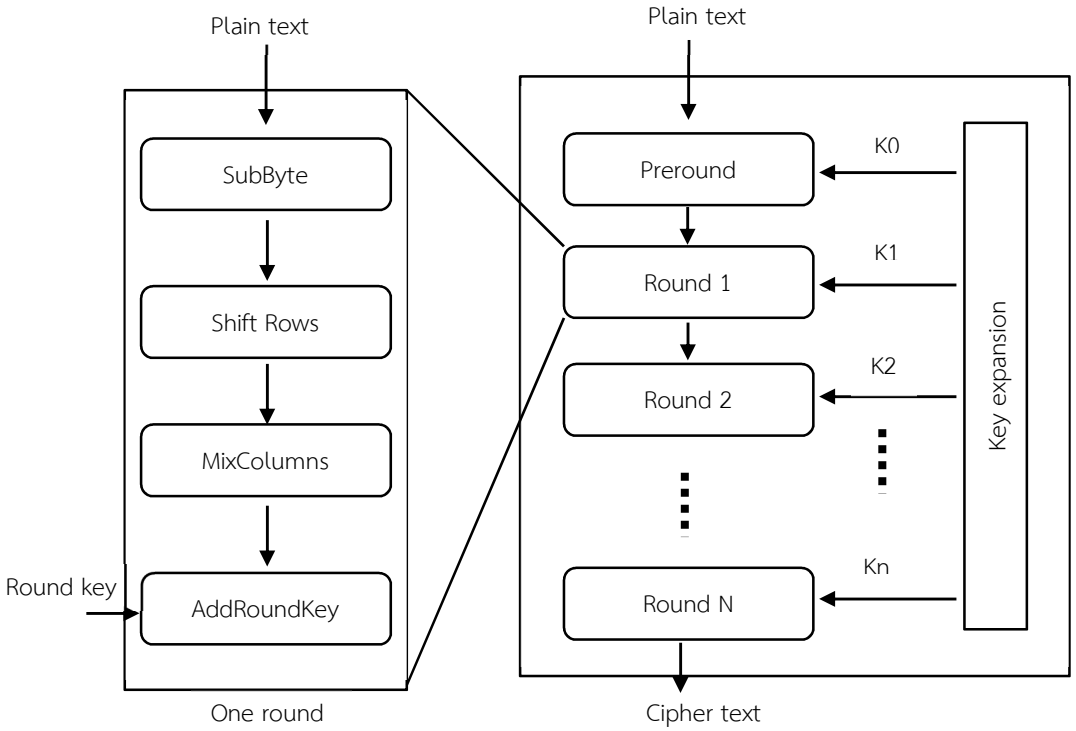
(3) เมื่อระบบต้องประมวลผลข้อมูลที่ซับซ้อนขึ้นทำให้ใช้เวลาทำงานนานขึ้น เพื่อให้ระบบทำงานเร็วขึ้นสามารถลดระยะเวลาในส่วนของการเข้ารหัสข้อมูลจากเดิมใช้วงจร AES 192 บิต เปลี่ยนเป็น AES 128 บิต ที่ใช้เวลาประมวลผลน้อยกว่า และเมื่อระบบไม่ต้องประมวลผลข้อมูลที่ซับซ้อนแล้วก็สามารถเปลี่ยนกับไปใช้ AES 192 บิต เพื่อให้ระบบมีความปลอดภัยสูงขึ้น

จะเห็นได้ว่าด้วยคุณสมบัติ Partial reconfiguration สามารถประยุกต์ใช้กับเครือข่ายเซนเซอร์ไร้สายให้ระบบมีความยืดหยุ่นรองรับกับการใช้งานที่มีประสิทธิภาพสูงสุด งานวิจัยได้ทดสอบการทำงานด้วยวงจรอัลกอริทึม Advanced Encryption Standard (AES) หรือวงจรเข้ารหัสข้อมูลซึ่งใช้คีย์ขนาด 128 บิต และ 192 บิต เพื่อรายงานผลทดสอบการใช้ทรัพยากรเอฟพีจีเอ, วิเคราะห์กำลังงานสูญเสีย และรายงานขนาดของไฟล์โปรแกรมพร้อมกับเปรียบเทียบผลทดสอบกับการออกแบบไฟล์วงจรโดยวิธีปกติ Full reconfiguration

4.5.2.1 Advanced Encryption Standard (AES)

วงจรเข้ารหัสข้อมูลแบบ Advanced Encryption Standard (AES) [22] เป็นระบบการเข้ารหัสที่ถูกกำหนดโดย National institute of standards and technology โดยใช้วิธีเข้ารหัสบนพื้นฐานการเข้ารหัสแบบ Rijindael โดยที่จะเข้ารหัสข้อมูลครั้งละ 128 บิต (16 ไบท์) ด้วยคีย์ (Key) ขนาดต่าง ๆ คือ 128, 192 และ 256 บิต กระบวนการในการเข้ารหัสถูกแบ่งเป็นรอบ แต่ละรอบมีการทำงานที่เหมือนกัน ภาพประกอบ 4-6 แสดงให้เห็นการทำงานของระบบทั้งหมด

การเข้ารหัสจะทำกระบวนการเดิมซ้ำกันเป็นจำนวนรอบ โดยขึ้นอยู่กับขนาดของคีย์ เช่น คีย์ขนาด 128 บิต ทำงาน 10 รอบ (Preround-Round10), คีย์ขนาด 192 บิต ทำงาน 12 รอบ (Preround-Round12) รอบ และคีย์ขนาด 256 บิต ทำงาน 14 รอบ (Preround-Round14) โดยที่เริ่มจากขั้นตอนการเพิ่มจำนวนคีย์ (Key expansion) ซึ่งขนาดของคีย์ที่เพิ่มจะขึ้นอยู่กับจำนวนรอบที่ทำงาน และแต่ละรอบมีอัลกอริทึมการทำงานประกอบด้วย 4 ส่วน คือ



ภาพประกอบ 4-6 กระบวนการทำงานของอัลกอริทึม AES

(1) SubByte คือ ขั้นตอนที่นำข้อมูล Plaintext มาแทนที่ด้วยข้อมูลจากตาราง S-box ซึ่งเป็นตารางแบบอาร์เรย์ที่สร้างขึ้นโดยใช้อัลกอริทึม Multiplicative inverse

(2) ShiftRow คือ ขั้นตอนที่นำข้อมูลมาเรียงลำดับในลักษณะ Matrix 4x4 แล้วทำการเลื่อน (Shift) ข้อมูลแต่ละแถว โดยที่แถวที่ 1 จะไม่ถูกเลื่อน แถวที่ 2, 3, 4 จะเลื่อน (Shift) ขวาไป 1, 2, 3 หลัก ตามลำดับ

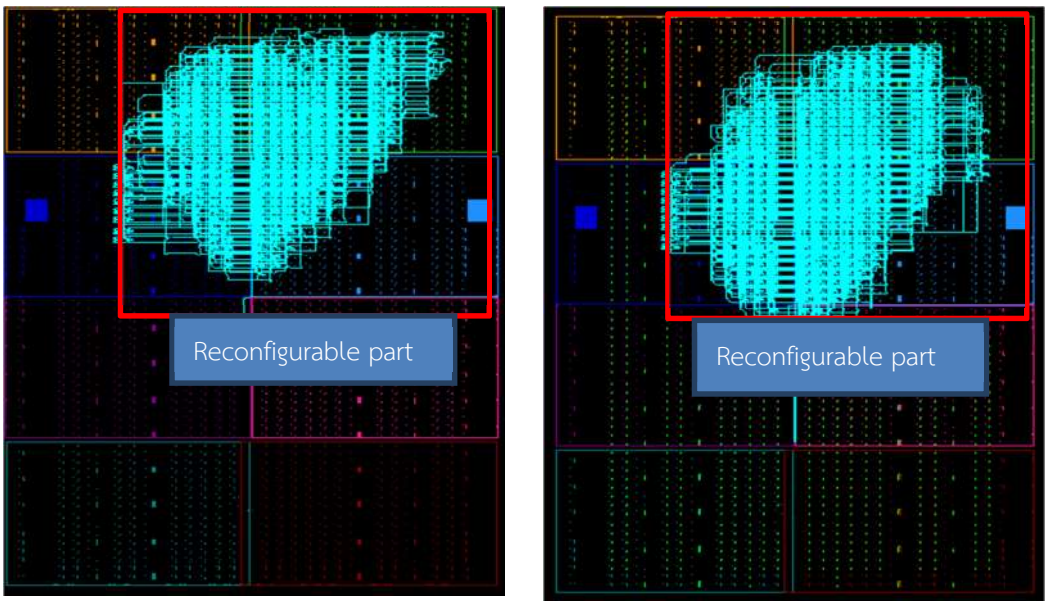
(3) MixColumns คือ การนำข้อมูลทีละคอลัมน์ของ Matrix4x4 ที่ได้มาคูณกับ Matrix ค่าคงที่ ผลลัพธ์ที่ได้จะมีการบวกและการคูณ ในการเข้ารหัสจะกำหนดให้บวก คือ XOR และการคูณ คือ Polynomial term ทำให้ได้ Matrix ชุดใหม่

(4) AddRoundKey ขั้นตอนนี้จะนำข้อมูลแต่ละคอลัมน์ของ Matrix มา XOR กับ Key ที่ตรงกับคอลัมน์นั้น โดยที่ Key นั้นได้มาจากขั้นตอนการทำ Key expansion

กระบวนการเหล่านี้จะทำเป็นจำนวนรอบตามขนาดของคีย์ (Key) โดยเริ่มตั้งแต่ Preround และในรอบสุดท้ายจะทำกระบวนการเดิมแต่ไม่ทำส่วนของ MixColumns เมื่อเสร็จกระบวนการทั้งหมดจะได้ข้อมูล Cipher text ซึ่งเป็นข้อมูลที่ถูกเข้ารหัสแล้ว

4.5.2.2 ผลการวางตำแหน่งวงจรและเส้นทางการเชื่อมต่อ (Place and Route)

ผลการวางตำแหน่งวงจรและเส้นทางการเชื่อมต่อภายในเอฟพีจีเอได้ข้อมูลจากไฟล์ Native circuit design (.ncd) สำหรับวงจร AES 128 บิต และวงจร AES 192 บิต แสดงดังภาพประกอบ 4-7 โดยพื้นที่ในกรอบสี่เหลี่ยม คือ พื้นที่ Reconfigurable part ที่ได้กำหนดไว้ส่วนพื้นที่นอกกรอบ คือ Static part



AES 128 AES 192

ภาพประกอบ 4-7 การวางตำแหน่งและเชื่อมต่อสัญญาณ วงจร AES 128 และ AES 192 บิต

จากภาพประกอบ 4-7 เป็นผลการวางตำแหน่งวงจรและเส้นทางเชื่อมต่อสัญญาณของวงจร AES 128 บิต และ AES 192 บิต พบว่าวงจรถูกวางไว้ในตำแหน่งที่ถูกต้อง คือ พื้นที่ Reconfigurable part ทั้งหมดตามที่ออกแบบและวงจร AES 192 บิต มีขนาดใหญ่กว่าวงจร AES 128 บิต

4.5.2.3 ผลทดสอบการใช้ทรัพยากรของเอฟพีจีเอ (Resource Utilization)

ผลทดสอบการใช้ทรัพยากรภายในเอฟพีจีเอ การออกแบบวงจร AES ขนาดคีย์ 128 บิต ด้วยวิธีการออกแบบไฟล์วงจรแบบ Full reconfiguration และ Partial reconfiguration แสดงดังตารางที่ 4-10

ตารางที่ 4-10 การใช้ทรัพยากรภายในเอฟพีจีเอของวงจร AES 128 บิต

ทรัพยากรเอฟพีจีเอ	จำนวนทรัพยากรที่ใช้งาน (%)	
	Full reconfiguration	Partial reconfiguration
BUFG (clock)	1 (6%)	1 (6%)
Logic LUTs	1,054 (11%)	1,417 (15%)
Register	688 (3%)	822 (4%)
Memory	4 (1%)	4 (1%)

จากตารางที่ 4-10 เป็นผลทดสอบการใช้ทรัพยากรเอฟพีจีเอของวงจร AES 128 บิต พบว่าวิธีการออกแบบไฟล์วงจรแบบ Partial reconfiguration ใช้ทรัพยากรเอฟพีจีเอโดยเฉพาะ SLICE logic มากกว่าแบบ Full reconfiguration สำหรับการออกแบบวงจร AES ขนาดคีย์ 192 บิต ได้ผลทดสอบการใช้ทรัพยากรภายในเอฟพีจีเอดังตารางที่ 4-11

ตารางที่ 4-11 การใช้ทรัพยากรในเอฟพีจีเอ ของวงจร AES 192 บิต

ทรัพยากรเอฟพีจีเอ	จำนวนทรัพยากรที่ใช้งาน (%)	
	Full reconfiguration	Partial reconfiguration
BUFG (clock)	1 (6%)	1 (6%)
Logic LUTs	1,289 (14%)	1,681 (18%)
Register	819 (4%)	941 (5%)
Memory	4 (1%)	4 (1%)

จากตารางที่ 4-11 เป็นผลทดสอบการใช้ทรัพยากรเอฟพีจีเอของวงจร AES 192 บิต พบว่าวิธีการออกแบบไฟล์วงจรแบบ Partial reconfiguration ใช้ทรัพยากรมากกว่าแบบ Full reconfiguration โดยเฉพาะ SLICE logic เช่นเดียวกับ AES 128 บิต ซึ่งเป็นผลมาจากทรัพยากรบางส่วนถูกใช้เป็น Bus macro เชื่อมต่อระหว่างพื้นที่ Static กับ Reconfigurable part

4.5.2.4 ผลทดสอบกำลังงานสูญเสีย (Power Consumption)

ผลการทดสอบกำลังงานสูญเสียของวงจร AES 128 บิต สำหรับการออกแบบไฟล์วงจรแบบ Full reconfiguration และ Partial reconfiguration สามารถแจกแจงรายละเอียดกำลังงานสูญเสียของแต่ละแหล่งจ่ายแสดงดัง ตารางที่ 4-12

ตารางที่ 4-12 กำลังงานสูญเสียของวงจร AES 128 บิต

แหล่งจ่าย	**กำลังงานสูญเสีย (w)**	
	Full reconfiguration	**Partial reconfiguration**
Vccint (1.2 V)	0.0084	0.0084
Vccaux (2.5 V)	0.0075	0.0075
Total	0.015	0.015

และผลทดสอบกำลังงานสูญเสียของวงจร AES 192 บิต สำหรับการออกแบบไฟล์วงจรแบบ Full reconfiguration และ Partial reconfiguration สามารถแจกแจงรายละเอียดกำลังงานสูญเสียของแต่ละแหล่งจ่ายแสดงดังตารางที่ 4-13

ตารางที่ 4-13 กำลังงานสูญเสียของวงจร AES 192 บิต

แหล่งจ่าย	**กำลังงานสูญเสีย (w)**	
	Full reconfiguration	**Partial reconfiguration**
Vccint (1.2 V)	0.0084	0.0084
Vccaux (2.5 V)	0.0075	0.0075
Total	0.015	0.015

จากตารางที่ 4-12 และตารางที่ 4-13 เป็นผลทดสอบกำลังงานสูญเสียของวงจร AES 128 และ AES 192 บิต ตามลำดับ พบว่าการออกแบบไฟล์วงจรแบบ Full reconfiguration และ Partial reconfiguration มีค่ากำลังงานสูญเสียรวมเท่ากัน คือ 15 มิลลิวัตต์ ดังนั้นการโปรแกรม แบบ Partial reconfiguration แม้ว่าจะใช้ทรัพยากรมากกว่า Full reconfiguration แต่ทรัพยากรเหล่านั้นไม่มีผลกระทบกับค่ากำลังงานสูญเสียของวงจร

4.5.2.5 ผลทดสอบขนาดไฟล์วงจร (Configuration File Size)

ผลทดสอบขนาดไฟล์วงจรหรือ Bitstream file ของวงจร AES 128 บิต และ AES 192 บิต ที่ใช้กระบวนการออกแบบไฟล์วงจรแบบ Full reconfiguration และ Partial reconfiguration แสดงดังตารางที่ 4-14

ตารางที่ 4-14 ขนาดไฟล์ Bitstream ของวงจร AES 128 และ AES 192 บิต

ประเภทของไฟล์วงจร	ขนาดไฟล์วงจร (KB)	
	AES 128	AES 192
Full bitstream	453.40	453.40
Static bitstream	453.41	453.41
Partial bitstream	118.8	141.05

จากตารางที่ 4-14 เป็นผลลัพธ์ขนาดไฟล์วงจร (Bitstream file) ของวงจร AES 128 และ AES 192 บิต ที่ออกแบบ Full reconfiguration และ Partial reconfiguration พบว่าไฟล์วงจร Full Bitstream ซึ่งเป็นผลลัพธ์ของการออกแบบ Full reconfiguration และไฟล์วงจร Static Bitstream ซึ่งเป็นผลลัพธ์ของการออกแบบ Partial reconfiguration มีขนาดใกล้เคียงกัน ในขณะที่ไฟล์วงจร Partial Bitstream มีขนาดเล็กกว่าเกือบ 4 เท่า

4.5.2.6 ผลทดสอบเวลาในการโปรแกรมวงจรลงเอฟพีจีเอ (Configuration time)

ผลทดสอบเวลาในการโปรแกรมวงจรด้วยเส้นสัญญาณ RS-232 และโปรแกรมวงจรด้วย Zigbee protocol โดยใช้โมดูล Xbee ผลการทดสอบแสดงดังตารางที่ 4-15

ตารางที่ 4-15 เวลาที่ใช้โปรแกรมวงจร AES 128 และ AES 192 ลงในเอฟพีจีเอ

วิธีการโปรแกรมวงจร	Time (s)	
	AES 128 + ICAP	AES 192 + ICAP
ผ่านสายสัญญาณ RS-232 โดยตรง	8.25 + 0.0023	9.80 + 0.0028
ผ่านโมดูล Xbee (Zigbee protocol)	99.0 + 0.0023	117.54 + 0.0028

จากตารางที่ 4-15 เป็นผลทดสอบเวลาในการโปรแกรมวงจร AES 128 บิต และ AES 192 บิต พบว่าเวลาที่ใช้โปรแกรมผ่านสายสัญญาณใช้เวลาน้อยกว่า 10 วินาที ในขณะที่หากโปรแกรมผ่านโมดูล Xbee ใช้เวลาอย่างน้อย 100 วินาที

ผลการทดสอบพบว่าทั้งวงจรอย่างง่าย คือ วงจรบวก 8 บิต และวงจรที่ซับซ้อนขึ้น คือ AES 128 บิต และ AES 192 บิต มีผลลัพธ์ไปในทิศทางเดียวกัน ผลลัพธ์การวางวงจรสามารถวางในพื้นที่ Reconfigurable part ได้ตามที่กำหนดไว้ ผลทดสอบการใช้ทรัพยากรพบว่าใช้ทรัพยากรมากกว่าการออกแบบวงจรแบบปกติ Full reconfiguration แต่ผลทดสอบค่ากำลังงานสูญเสียพบว่ามีค่ากำลังงานสูญเสียเท่ากับวงจรที่ออกแบบ Full reconfiguration

4.6 สรุปผลการทดสอบ

จากผลการทดสอบพบว่ากระบวนการออกแบบไฟล์วงจรพื้นที่ Static part สามารถทำงานได้ถูกต้อง ผลลัพธ์ของพื้นที่ Static part การวางวงจรและเส้นทางเชื่อมต่อได้แบ่งพื้นที่เอฟพีจีเป็นสองส่วน คือ พื้นที่ Static part ประกอบไปด้วยวงจรควบคุมการโปรแกรม ส่วนที่สอง คือ พื้นที่ Reconfigurable part ซึ่งส่วนนี้จะเป็นพื้นที่โล่งไม่มีวงจรใด เมื่อทดสอบกระบวนการออกแบบไฟล์วงจรพื้นที่ Static part ร่วมกับไมโครโพรเซสเซอร์ สามารถทำงานได้ถูกต้องเช่นกัน ผลลัพธ์ที่ได้คือ ไมโครโพรเซสเซอร์จะอยู่ในพื้นที่ Static part ที่ออกแบบไว้ ผลทดสอบการใช้ทรัพยากรของพื้นที่ Static part พบว่าต้องใช้ทรัพยากรประมาณ 20% ทำให้เหลือทรัพยากรประมาณ 80% ไว้สำหรับใช้งานเป็นพื้นที่ Reconfigurable part และผลทดสอบกำลังงานสูญเสียพบว่า พื้นที่ Static part มีค่ากำลังงานสูญเสียน้อยซึ่งจะไม่ส่งผลกระทบต่อกำลังงานสูญเสียรวมทั้งหมดของเอฟพีจีเอ และมีค่ากำลังงานสูญเสียมากขึ้นเมื่อมีการใช้งานไมโครโพรเซสเซอร์ อย่างไรก็ตามพื้นที่ Static part ที่มีไมโครโพเซสเซอร์ร่วมทำให้เพิ่มความสามารถของพื้นที่ Static part มากยิ่งขึ้นโดยเฉพาะการรองรับการนำไปใช้เป็นเซนเซอร์โนดในระบบเครือข่ายเซนเซอร์ไร้สาย

จากผลการทดสอบกระบวนการออกแบบไฟล์วงจรพื้นที่ Reconfigurable part พบว่าสามารถทำงานได้ถูกต้อง ผลลัพธ์ของพื้นที่ Reconfigurable part ทั้งวงจรอย่างง่าย คือ วงจรบวก 8 บิต และวงจรที่ซับซ้อน คือ AES 128 บิต และ AES 192 บิต สามารถวางวงจรและเส้นทางเชื่อมต่อเฉพาะในพื้นที่ Reconfigurable part ที่กำหนดไว้เท่านั้น ผลทดสอบการใช้งานทรัพยากรของแต่ละวงจรเมื่อเปรียบเทียบกับกระบวนการออกแบบไฟล์วงจรแบบปกติ Full reconfiguration พบว่ามีการใช้ทรัพยากรมากกกว่าเนื่องจากพื้นที่ Reconfigurable part มีทรัพยากรบางส่วนทำหน้าที่เป็น Bus macro สำหรับรับส่งข้อมูลระหว่างพื้นที่ ในขณะที่ผลทดสอบกำลังงานสูญเสียของวงจรบวกและ AES เมื่อเปรียบเทียบกับการออกแบบวงจรแบบ Full reconfiguration พบว่ามีค่ากำลังงานสูญเสียเท่ากัน ทำให้ทราบว่าทรัพยากรที่ใช้มากกว่านั้นไม่มีผลกระทบกับค่ากำลังงานสูญเสีย ผลทดสอบขนาดไฟล์วงจรพบว่า ไฟล์วงจรพื้นที่ Static part มีขนาดใกล้เคียงกับไฟล์วงจรที่ออกแบบโดย Full reconfiguration แต่ขณะเดียวกันไฟล์วงจรแบบ Partial Bitstream มีขนาดเล็กกว่าถึง 4 เท่า เมื่อเทียบกับไฟล์วงจร Full bitstream ของการออกแบบ Full reconfiguration ทำให้เมื่อคำนวณเวลาที่ใช้โปรแกรมวงจรพบว่าการโปรแกรมผ่านสายสัญญาณใช้เวลาน้อยกว่า 10 วินาที และโปรแกรมผ่าน Xbee ใช้เวลาอย่างน้อย 100 วินาที ซึ่งเป็นไปตามที่คาดไว้ เนื่องจากการรับส่งข้อมูลผ่านสายสัญญาณมีอัตราการรับส่งสูงกว่าผ่านทาง Xbee ถึงประมาณ 12 เท่า ทั้งนี้ระยะเวลาโปรแกรมนั้นขึ้นอยู่กับขนาดไฟล์วงจรเช่นกัน

บทที่ 5
สรุปผลการวิจัยและข้อเสนอแนะ

หลังจากที่ได้ศึกษาขั้นตอนการออกแบบวงจรแบบ Partial reconfiguration บนเอฟพีจีเอ ดังรายละเอียดบทที่ 2 ถึง 4 สำหรับในบทนี้จะกล่าวถึงการสรุปผลและข้อเสนอแนะที่ได้จากวิทยานิพนธ์ รวมถึงปัญหาและอุปสรรคที่เกิดขึ้นในการทำวิทยานิพนธ์

5.1 สรุปผลการวิจัย

งานวิจัยในวิทยานิพนธ์นี้ได้ นำเสนอขั้นตอนการโปรแกรมวงจรเอฟพีจีเอแบบ Partial reconfiguration โดยใช้เอฟพีจีเอ Spartan-6 ที่มีขนาดเล็กและใช้กำลังไฟต่ำเพื่อนำไปประยุกต์ใช้เป็นเซนเซอร์โนดในระบบเครือข่ายเซนเซอร์ไร้สาย อย่างไรก็ตามข้อจำกัดของการใช้ Spartan-6 คือ ซอฟต์แวร์ของผู้ผลิต Xilinx ไม่รองรับการออกแบบวงจรแบบ Partial reconfiguration ในเอฟพีจีเอ Spartan-6 จากปัญหาดังกล่าว งานวิจัยนี้จึงนำเสนอกระบวนการออกแบบ Partial reconfiguration โดยใช้ซอฟต์แวร์ GoAhead ร่วมกับซอฟต์แวร์ของผู้ผลิต Xilinx และทดสอบกระบวนการที่ได้ออกแบบด้วยวงจรบวกอย่างง่าย และวงจรเข้ารหัส AES โดยตรวจสอบความถูกต้องของวงจรขณะทำงาน ทั้งยังพิจารณาการวางตำแหน่งวงจรในเอฟพีจีเอพร้อมกับวิเคราะห์การใช้ทรัพยากรและกำลังงานสูญเสีย ผลการทดสอบพบว่า

(1) เมื่อเอฟพีจีเอถูกโปรแกรมวงจรแบบ Partial reconfiguration สามารถทำงานได้อย่างถูกต้อง ทั้งยังสามารถโปรแกรมวงจรในพื้นที่ Reconfigurable ซ้ำได้ โดยผ่านทางพอร์ตอนุกรม RS-232 และระบบเครือข่ายไร้สาย

(2) การวางตำแหน่งวงจรของพื้นที่ Static part และ Reconfigurable สามารถวางตำแหน่งโดยมีเส้นทางวงจรอยู่ในพื้นที่ที่กำหนดไว้ และทำได้อย่างถูกต้อง

(3) การใช้งานทรัพยากรภายในเอฟพีจีเอ พบว่า การออกแบบวงจรด้วยกระบวนการ Partial reconfiguration ใช้ทรัพยากรมากกว่าการออกแบบด้วยกระบวนการแบบปกติ เนื่องจาก ทรัพยากรบางส่วนถูกนำไปใช้เป็น Bus macro เพื่อสื่อสารระหว่างพื้นที่ Static และ Reconfigurable part

(4) ค่ากำลังงานสูญเสียของวงจรพบว่า วงจรที่ออกแบบโดยกระบวนการปกติและวงจรที่ออกแบบโดยกระบวนการ Partial reconfiguration มีค่ากำลังงานสูญเสียเท่ากัน

จากผลการวิจัยพบว่าเอฟพีจีเอสามารถทำงานได้ถูกต้องตามวงจรที่ถูกโปรแกรมด้วยกระบวนการ Partial reconfiguration นั้น มีความสอดคล้องตามทฤษฎีเป็นอย่างดี สามารถนำมาพัฒนาเป็นเซนเซอร์โนดในระบบเครือข่ายเซนเซอร์ไร้สายที่มีคุณสมบัติการปรับเปลี่ยนตัวเองได้ในอนาคต นอกจากนั้นผู้วิจัยได้พัฒนาซอฟต์แวร์สำหรับออกแบบวงจรด้วยกระบวนการ Partial reconfiguration โดยพัฒนาให้ใช้งานง่าย มีกราฟฟิคแสดงผล เพื่อให้บุคคลทั่วไปที่ไม่จำเป็นต้องมีความเข้าใจเชิงลึกเกี่ยวกับโครงสร้างเอฟพีจีเอสามารถนำไปใช้ประโยชน์ได้

5.2 ข้อจำกัดของงานวิจัย

5.2.1 ประเภทหน่วยความจำโปรแกรม (Configuration Memory)

ประเภทหน่วยความจำของเอฟพีจี Spartan-6 มีหน่วยความจำแบบ Static RAM based โดยทั่วไปข้อมูลวงจรโปรแกรมถูกเก็บไว้ใน Static ram เมื่อหยุดจ่ายไฟเอฟพีจีเอจะถูกรีเซททำให้วงจรทั้งหมดหายไป ดังนั้นการนำเอฟพีจีเอมาใช้งานแบบ Partial reconfiguration จำเป็นที่พื้นที่ Static part ต้องจ่ายไฟตลอดเวลา

อย่างไรก็ตาม โดยทั่วไปนั้นมักจะออกแบบให้บอร์ดวงจรที่มีเอฟพีจีเออยู่ภายในจะมีการเชื่อมต่อระหว่างเอฟพีจีเอกับหน่วยความจำประเภท Flash memory ไว้ เพื่อทำหน้าที่เป็น Program memory กล่าวคือ เมื่อมีการจ่ายไฟให้บอร์ดวงจร เอฟพีจีเอถูกเซตให้ทำงานโหมด Configuration mode เช่น JTAG, Master SPI เพื่อให้ Configure ตัวเองโดยข้อมูลวงจรได้จาก Program memory ที่ผู้ใช้บันทึกไว้ในหน่วยความจำ flash memory ดังนั้นทุกครั้งที่เอฟพีจีเอรีเซตตัวเองก็จะ Configure ตัวเองด้วยวงจรที่บันทึกไว้ เสมือนว่าวงจรทั้งหมดยังคงอยู่ในเอฟพีจีเอถึงแม้ว่าจะหยุดจ่ายไฟ

วิธีดังกล่าวนี้นำมาประยุกต์ใช้กับการออกแบบวงจร Partial reconfiguration ได้โดยเมื่อผู้ใช้ออกแบบวงจรในพื้นที่ Static part เสร็จแล้วให้บันทึกลงในหน่วยความจำ Flash memory เมื่อเกิดสถานการณ์ที่ทำให้บอร์ดเอฟพีจีเอหยุดจ่ายไฟหรือรีเซตตัวเอง เอฟพีจีเอจะอ่านข้อมูลวงจรจาก flash memory ซึ่งเป็นวงจรพื้นที่ Static part เพื่อเริ่มต้นทำงานอีกครั้ง

5.2.2 ความถูกต้องของไฟล์วงจร

เอฟพีจีเอสามารถตรวจสอบความถูกต้องของไฟล์วงจรได้ด้วยตัวเอง ซึ่งค่าสถานะความถูกต้องของไฟล์วงจรจะอยู่ในบิตที่ 0 เรียกว่า CRC_ERROR ซึ่งได้จากกระบวนการ CRC โดยอยู่ในส่วนของรีจิสเตอร์ Status ซึ่งเป็นรีจิสเตอร์ขนาด 16 บิต ทำหน้าที่เก็บข้อมูลสถานะของเอฟพีจีเอเมื่อมีการโปรแกรมวงจร โดยที่แต่ละบิตเก็บค่าสถานะแตกต่างกัน สำหรับการอ่านค่ารีจิสเตอร์ Status สามารถใช้วิธี Readback ผ่านทางพอร์ต SelectMAP, JTAG รวมถึง ICAP

อย่างไรก็ตาม การตรวจสอบความถูกต้องของไฟล์วงจร สำหรับการโปรแกรมแบบปกติเริ่มต้นจากสถานะ Configuration mode เมื่อทำการเขียนข้อมูลลงในหน่วยความจำโปรแกรมซึ่งเกิดข้อผิดพลาดของข้อมูล เอฟพีจีเอจะไม่เข้าสู่สถานะ User mode เพื่อเริ่มทำงาน ทำให้ผู้ใช้งานรับรู้ได้ถึงข้อผิดพลาดทันที แต่การโปรแกรมแบบ Partial reconfiguration เอฟพีจีเอต้องอยู่ในสถานะ User mode ก่อนจึงจะเขียนข้อมูลหรือโปรแกรมวงจรได้ ผู้ใช้งานจะรับรู้ถึงข้อผิดพลาดหลังจากวงจรทำงานแล้วผลลัพธ์ไม่เป็นไปตามที่ต้องการ ดังนั้น จะเห็นได้ว่าการตรวจสอบความถูกต้องเพื่อป้องกันวงจรทำงานผิดพลาดโดยวิธี Readback ไม่เหมาะสมกับการโปรแกรมแบบ Partial reconfiguration ได้ซึ่งเป็นข้อจำกัดของงานวิจัยนี้

นอกจากนั้นข้อจำกัดเรื่องความถูกต้องของวงจรอีกประการหนึ่ง คือ เนื่องจากงานวิจัยได้กำหนดให้รับส่งข้อมูลไฟล์วงจรผ่านระบบไร้สายมาตรฐาน Zigbee protocol โดยใช้งานอุปกรณ์ Xbee module ซึ่งอุปกรณ์นี้สามารถกำหนดรูปแบบคำสั่งการทำงานได้ 2 รูปแบบ คือ API mode, AT mode ซึ่งในงานวิจัยนี้เลือกใช้มาตรฐาน AT mode เนื่องจาก

สามารถใช้งานได้ง่าย รูปแบบคำสั่งไม่ซับซ้อน อย่างไรก็ตาม การใช้งาน AT mode มีข้อจำกัดเรื่องของความถูกต้องของข้อมูล คือ ฝั่งส่งทำหน้าที่ส่งข้อมูลโดยไม่มีการตรวจสอบว่าฝั่งรับสามารถรับข้อมูลได้ถูกต้องหรือไม่ ส่งผลให้อาจเกิดกรณีที่ไฟล์วงจรที่เอฟพีจีเอได้รับเกิดข้อผิดพลาด เช่น ข้อมูลไม่ครบ ข้อมูลบางส่วนหายไป เป็นต้น

อย่างไรก็ดี ข้อจำกัดเรื่องการใช้งานแบบ AT mode สามารถปรับปรุงได้โดยเปลี่ยนรูปแบบคำสั่งเป็น API mode ซึ่งมีกระบวนการ Checksum เพื่อตรวจสอบความถูกต้องของข้อมูลที่รับส่ง

5.3 ข้อเสนอแนะ

วิทยานิพนธ์นี้นำเสนอขั้นตอนและกระบวนการออกแบบวงจรบนเอฟพีจีเอ Spartan-6 แบบ Partial reconfiguration ผู้ที่สนใจนำวิทยานิพนธ์ไปใช้งาน สามารถนำไปทดสอบและนำไปใช้ได้จริง นอกจากนั้นผู้ที่สนใจนำวิทยานิพนธ์มาศึกษาต่อเนื่องสามารถปรับปรุงในประเด็นต่าง ๆ ดังนี้

(1) พัฒนาให้เอฟพีจีเอสามารถแบ่งพื้นที่ Reconfigurable part ได้มากกว่า 1 พื้นที่ โดยสามารถเลือกให้โปแกรมวงจรลงในพื้นที่ใดพื้นที่หนึ่งได้ ในขณะเดียวกันพื้นที่อื่น ๆ ยังคงทำงานอยู่ การพัฒนานี้ทำให้เอฟพีจีเอมีความยืดหยุ่นต่อการนำไปใช้งาน

(2) พัฒนาวิธีการโปรแกรมวงจรพื้นที่ Reconfigurable part โดยโพรโตคอลอื่น ๆ เพื่อให้การโปรแกรมวงจรทำได้เร็วขึ้นแทน UART protocol ซึ่งเป็นการสื่อสารแบบอนุกรมที่ใช้ในงานวิจัย เช่น โพรโตคอลที่รับส่งข้อมูลแบบขนาน เป็นต้น

(3) ศึกษากระบวนการตรวจสอบความถูกต้องของไฟล์วงจรที่โปแกรมแบบ Partial reconfiguration เพื่อให้รองรับกับการนำไปใช้งานจริง

บรรณานุกรม

[1] E. Gilbert, K. Baskaran, and E. E. Blessing, “Research issues in wireless sensor network applications: a survey,” *Int. J. Inf. Electron. Eng.*, vol. 2, no. 5, pp. 702–706, 2012.

[2] “Partial Reconfiguration User Guide UG702,” XILINX, v14.5, Apr. 2013.

[3] R. Garcia, A. Gordon-Ross, and A. D. George, “Exploiting Partially Reconfigurable FPGAs for Situation-Based Reconfiguration in Wireless Sensor Networks,” in *17th IEEE Symposium on Field Programmable Custom Computing Machines, 2009. FCCM ’09*, 2009, pp. 243–246.

[4] M. Rahmatian, H. Kooti, I. G. Harris, and E. Bozorgzadeh, “Adaptable intrusion detection using partial runtime reconfiguration,” in *2012 IEEE 30th International Conference on Computer Design (ICCD)*, 2012, pp. 147–152.

[5] J. Echanobe, I. del Campo, R. Finker, and K. Basterretxea, “Dynamic Partial Reconfiguration in Embedded Systems for Intelligent Environments,” in *2012 8th International Conference on Intelligent Environments (IE)*, 2012, pp. 109–113.

[6] P. Leray, A. Nafkha, and C. Moy, “Implementation scenario for teaching partial reconfiguration of FPGA,” in *Proc. 6th International Workshop on Reconfigurable Communication Centric Systems-on-Chip (ReCoSoC), Montpellier, France*, 2011.

[7] Y. Li, Z. Jia, F. Liu, and S. Xie, “Hardware reconfigurable wireless sensor network node with power and area efficiency,” *IET Wirel. Sens. Syst.*, vol. 2, no. 3, pp. 247–252, 2012.

[8] J. Gong, L. Zhao, Q. Hao, F. Hu, and X. Hong, “A reconfigurable hardware platform for cognitive sensor networks towards behavioral biometrics,” in *Sensors, 2012 IEEE*, 2012, pp. 1–4.

[9] Y. Krasteva, J. Portilla, E. de la Torre, and T. Riesgo, “Embedded runtime reconfigurable nodes for wireless sensor networks applications,” *Sens. J. IEEE*, vol. 11, no. 9, pp. 1800–1810, 2011.

[10] “PlanAhead User Guide UG632,” XILINX, v 11.4, Dec. 2009.

[11] “Spartan-6 FPGA Configuration User Guide UG380,” XILINX, v 2.4, Jun. 2012.

[12] J. Portilla, A. De Castro, E. De La Torre, and T. Riesgo, “A Modular Architecture for Nodes in Wireless Sensor Networks,” *J UCS*, vol. 12, no. 3, pp. 328–339, 2006.

[13] J. L. Wilder, V. Uzelac, A. Milenković, and E. Jovanov, “Runtime hardware reconfiguration in wireless sensor networks,” in *System Theory, 2008. SSST 2008. 40th Southeastern Symposium on*, 2008, pp. 154–158.

[14] C. Beckhoff, D. Koch, and J. Torresen, “Design Tools for Implementing Self-Aware and Fault-Tolerant Systems on FPGAs,” *ACM Trans. Reconfigurable Technol. Syst. TRETS*, vol. 7, no. 2, p. 14, 2014.

[15] J. Meyer, J. Noguera, M. Hübner, L. Braun, O. Sander, R. M. Gil, R. Stewart, and J. Becker, “Fast start-up for spartan-6 fpgas using dynamic partial reconfiguration,” in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011*, 2011, pp. 1–6.

[16] A. Otero, E. de la Torre, and T. Riesgo, “Dreams: A tool for the design of dynamically reconfigurable embedded and modular systems,” in *Reconfigurable Computing and FPGAs (ReConFig), 2012 International Conference on*, 2012, pp. 1–8.

[17] S. A. Guccione and D. Levi, “Jbits: A java-based interface to fpga hardware,” *Xilinx Inc San Jose CA*, 1998.

[18] A. A. Sohanghpurwala, P. Athanas, T. Frangieh, and A. Wood, “Openpr: An open-source partial-reconfiguration toolkit for xilinx fpgas,” in *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, 2011, pp. 228–235.

[19] D. Koch, C. Beckhoff, and J. Teich, “Recobus-builder—a novel tool and technique to build statically and dynamically reconfigurable systems for FPGAs,” in *Field Programmable Logic and Applications, 2008. FPL 2008. International Conference on*, 2008, pp. 119–124.

[20] C. Beckhoff, D. Koch, and J. Torresen, “Go ahead: A partial reconfiguration framework,” in *Field-Programmable Custom Computing Machines (FCCM), 2012 IEEE 20th Annual International Symposium on*, 2012, pp. 37–44.

[21] “SP601 Hardware user Guide UG 518,” XILINX, v 1.7, Sep. 2012.

[22] “Advanced Encryption Standard,” National Institute of Standards and Technology (NIST, FIPS PUBS, Nov. 2001.

ภาคผนวก

ภาคผนวก ก
การติดตั้งซอฟต์แวร์ GoAhead

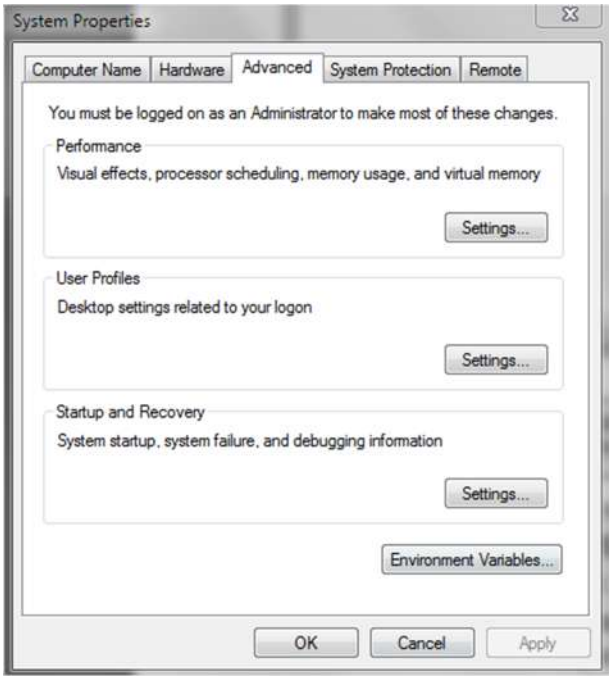
1. การติดตั้งซอฟต์แวร์ GoAhead

ซอฟต์แวร์ GoAhead ออกแบบและจัดทำขึ้นโดยนักวิจัยจาก มหาวิทยาลัย Universtiy of Oslo (UiO) ขั้นตอนการติดตั้งมีดังนี้

1.1. ดาวน์โหลดซอฟต์แวร์ GoAhead จากเว็บไซต์ของทีมวิจัยจาก The Faculty of Mathematics and Natural Sciences เข้าไปที่ลิงค์ "http://www.mn.uio.no/ifi/english/research/projects/cosrecos/goahead/Tool_Download/" จากนั้นคลิก Download โดย .zip ไฟล์ที่ได้มีขนาดประมาณ 70 MB

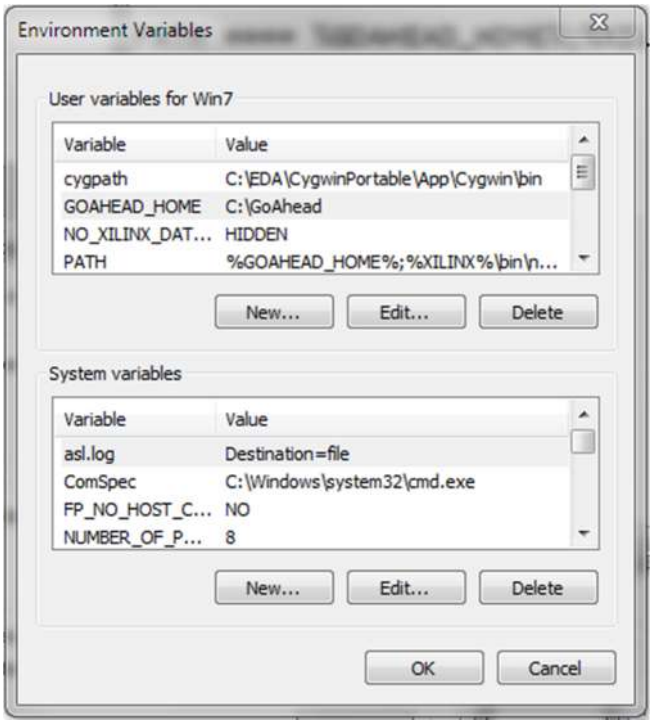
1.2. แตกไฟล์จาก .zip ที่ได้ ซึ่งจะได้โฟลเดอร์ชื่อ GoAhead จากนั้นให้นำโฟลเดอร์ดังกล่าวไปวางไว้ใน C:\

1.3. กำหนดค่า Environment variables ของระบบ โดยเข้าไปที่คลิกขวา My computer > Properties จะปรากฏหน้าต่าง System ให้เลือก Advanced system setting จะได้หน้าต่าง System properties ดังภาพประกอบ ก-1



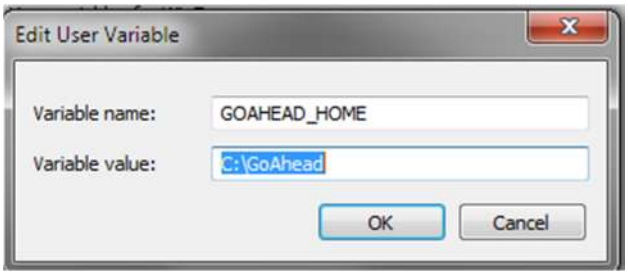
ภาพประกอบ ก-1 หน้าต่าง System Properties

1.4. จากนั้นคลิกเลือก Environment variables จะปรากฏหน้าต่างดังภาพประกอบ ก-2



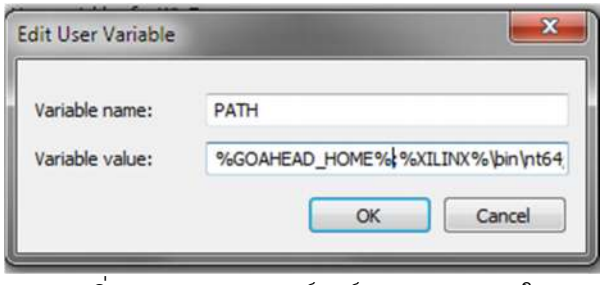
ภาพประกอบ ก-2 หน้าต่าง Environment Variables

1.5. กำหนด Environment variable ของซอฟต์แวร์ GoAhead โดยคลิกปุ่ม New ในช่อง User variables for Win7 โดยกำหนดให้ Variable name: GOAHEAD_HOME และ Variable value: C:\GoAhead ดังภาพประกอบ ก-3



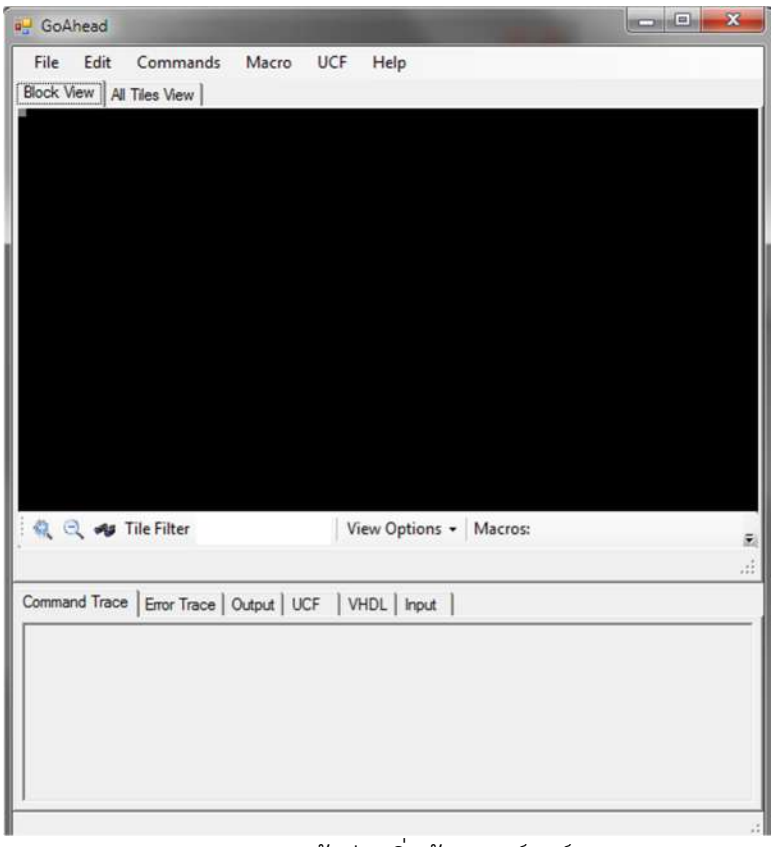
ภาพประกอบ ก-3 การกำหนด Environment Variable ของซอฟต์แวร์ GoAhead

1.6. จากนั้นทำการกำหนด Path variable โดยให้คลิกเลือก PATH ในช่อง Variable จากนั้นคลิกปุ่ม Edit แล้วทำการเพิ่ม Variable ของ GoAhead โดยใส่ %GOAHEAD_HOME%; และ % GOAHEAD_HOME%\CygwinPortable\App\Cygwin\bin; ไว้หน้าสุดดังภาพประกอบ ก-4



ภาพประกอบ ก-4 การเพิ่ม Path ของซอฟต์แวร์ GoAhead ลงใน Window Path เดิม

1.7. เมื่อถึงขั้นตอนนี้ ผู้ใช้สามารถเรียกใช้งานซอฟต์แวร์ GoAhead ได้แล้ว โดยสามารถเรียกใช้งานซอฟต์แวร์ได้ผ่านทาง C:\GoAhead > GoAhead.exe หรืออีกวิธี คือ รันผ่าน Command line โดยเปิด Command promt จากนั้นพิมพ์ GoAhead จะปรากฏหน้าต่าง ดังภาพประกอบ ก-5



ภาพประกอบ ก-5 หน้าต่างเริ่มต้นซอฟต์แวร์ GoAhead

ภาคผนวก ข
การติดตั้งซอฟต์แวร์ Partial Reconfiguration for FPGA (PReFF) Tool

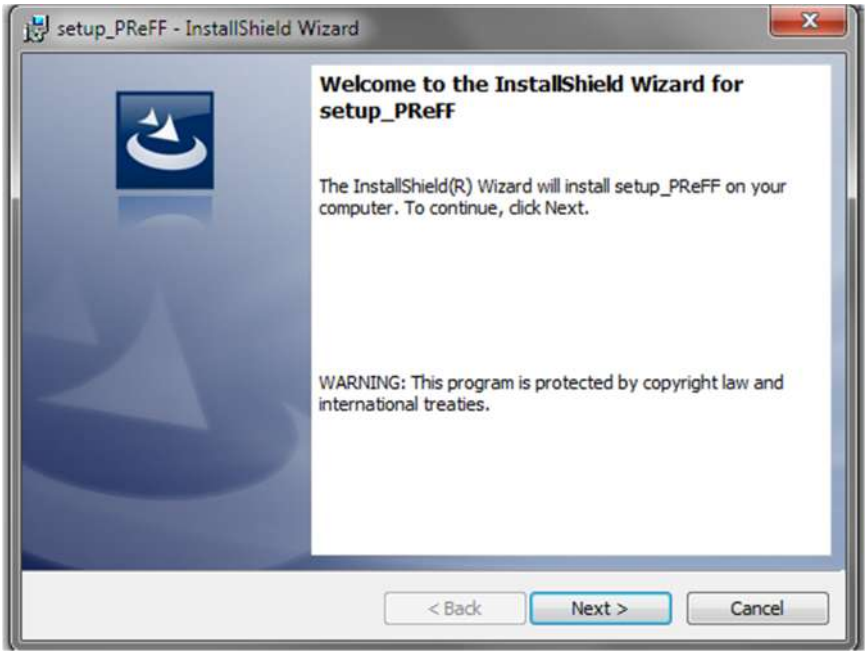
1. การติดตั้งซอฟต์แวร์ Partial Reconfiguration for FPGA (PReFF) Tool

เนื่องจากผู้วิจัยเห็นว่ากระบวนการทำ Partial reconfigurable วงจรฮาร์ดแวร์ในเอฟพีเอนั้นมีความยุ่งยาก และซับซ้อนในการออกแบบ เนื่องจากเป็นการแก้ไขกระบวนการออกแบบเดิม ทำให้ต้องแก้ไข ปรับเปลี่ยนตัวแปร หรือคุณสมบัติของกระบวนการเดิม และใช้ซอฟต์แวร์ GoAhead ซึ่งเป็นซอฟต์แวร์ที่ผลิตมาเพื่อใช้สำหรับผู้ที่ทำวิจัย ซึ่งต้องมีความรู้ ความเข้าใจในกระบวนการออกแบบทั้งหมดเป็นอย่างดี ทำให้เป็นการยากที่บุคคลทั่วไปสามารถใช้ความสามารถของเอฟพีจีเอแบบ Partial reconfiguration ได้ ดังนั้นผู้วิจัยจึงได้ออกแบบและพัฒนาซอฟต์แวร์เพื่อให้บุคคลทั่วไปสามารถออกแบบ Partial reconfigurable วงจรฮาร์ดแวร์ได้โดยไม่ยากนัก

Partial Reconfiguration for FPGA (PReFF) คือ ซอฟต์แวร์ที่ทำหน้าที่แก้ไขตัวแปรและเรียกใช้งานซอฟต์แวร์ที่เกี่ยวข้องกับกระบวนการทำ Partial reconfiguration คือ Xilinx ISE และ GoAhead โดยแสดงผลเป็น Graphic User Interface (GUI) ให้ผู้ใช้งานสามารถใช้ได้สะดวกขึ้น ขั้นตอนการติดตั้งซอฟต์แวร์ PReFF มีรายละเอียดดังนี้

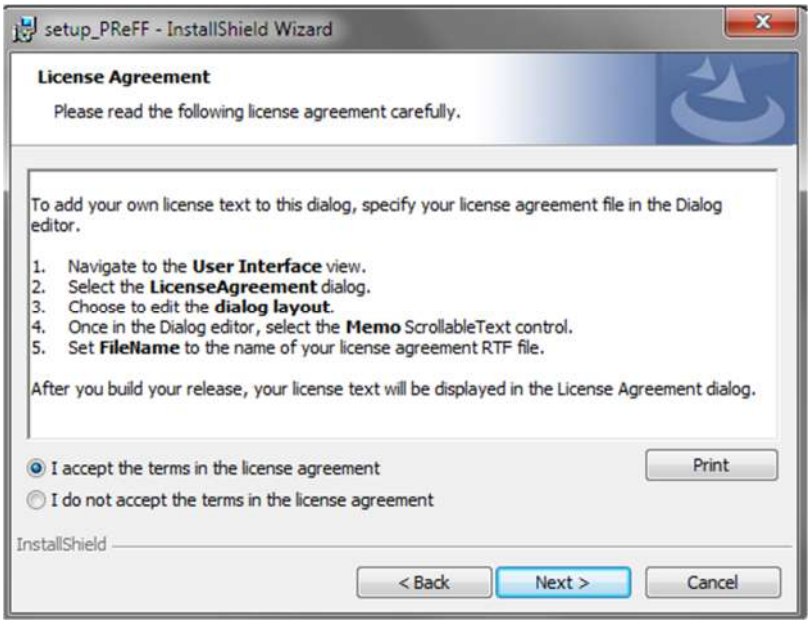
1.1. ผู้วิจัยได้อัพโหลดให้ผู้ที่สนใจสามารถดาวน์โหลดซอฟต์แวร์ได้โดยเข้าไปที่ https://drive.google.com/folderview?id=0BxHZ3RsiATzvMWlVbjBJb2d4Q00&usp=sharing#list

1.2. เมื่อดาวน์โหลดแล้วแยกไฟล์ .zip จะได้โฟลเดอร์ DISK1 ภายในจะมีไฟล์สำหรับติดตั้ง จากนั้นให้ Double click ไฟล์ Setup.exe จะปรากฏหน้าต่าง ภาพประกอบ ข-1 จากนั้นคลิกปุ่ม Next



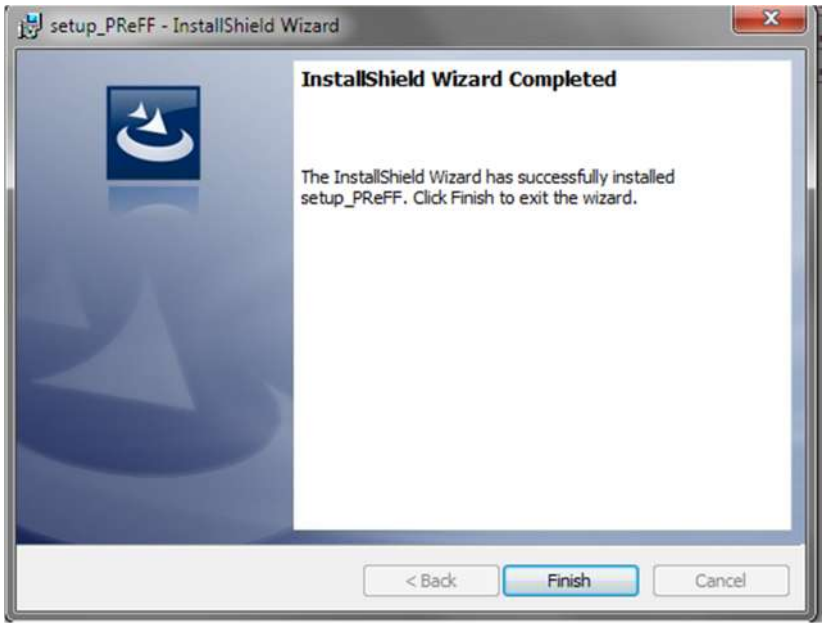
ภาพประกอบ ข-1 หน้าต่าง Setup ซอฟต์แวร์ GoAhead

1.3. จะปรากฏหน้าต่าง license agreement ดังภาพประกอบ ข-2 ให้เลือก I accept the terms in the license agreement จากนั้นคลิก Next



ภาพประกอบ ข-2 หน้าต่าง Setup ซอฟต์แวร์ GoAhead

1.4. จากนั้นคลิก Next จนเสร็จกระบวนการทั้งหมดจะเจอหน้าต่าง InstallShield wizard completed ดังภาพประกอบ ข-3 ให้คลิก Finish

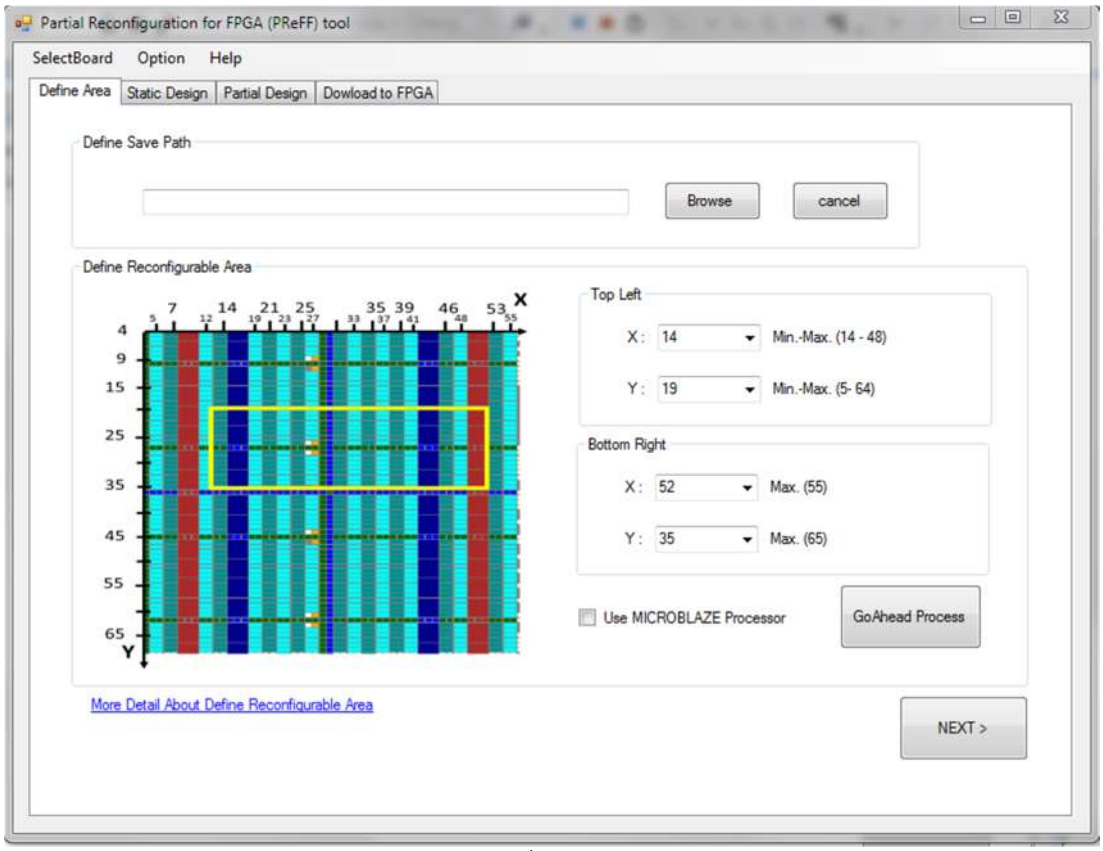


ภาพประกอบ ข-3 หน้าต่าง Setup ซอฟต์แวร์ GoAhead

เมื่อเสร็จกระบวนการติดตั้งจะได้ Shortcut ของซอฟต์แวร์ในหน้า Desktop ดังภาพภาพประกอบ ข-4

ภาพประกอบ ข-4 Shortcut ของซอฟต์แวร์ PReFF

เมื่อผู้ใช้งานคลิก Icon ของซอฟต์แวร์ PReFF จะปรากฏหน้าต่างโปรแกรมแสดงดังภาพประกอบ ข-5



ภาพประกอบ ข-5 หน้าต่างเริ่มต้นของซอฟต์แวร์ PReFF Tool

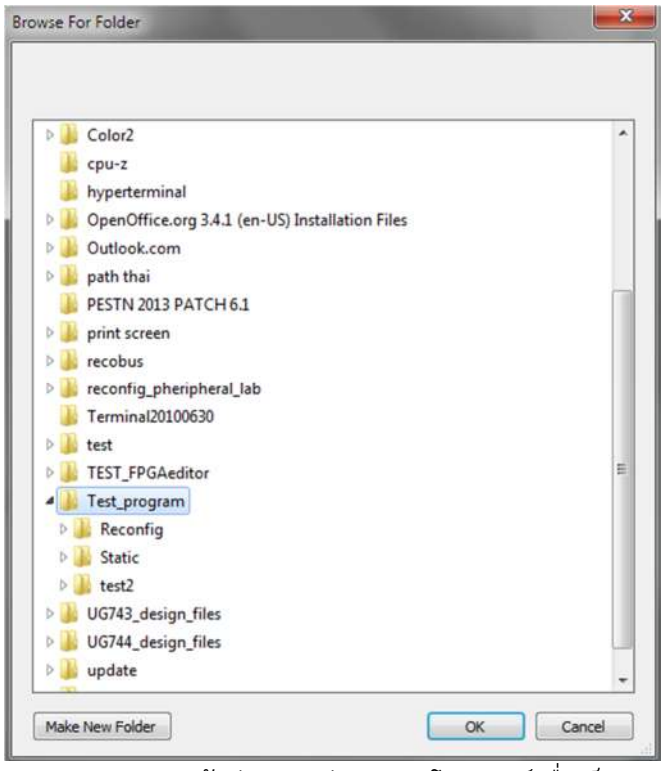
ภาคผนวก ค
การใช้งานซอฟต์แวร์ Partial Reconfiguration for FPGA (PReFF) Tool

การใช้งานซอฟต์แวร์ PReFF ในการ Reconfigurable เอฟพีจีเอจะต้องทำตามลำดับขั้น คือ การกำหนดพื้นที่ Reconfigurable area > การออกแบบพื้นที่ Static design > การออกแบบพื้นที่ Partial design > การ Reconfigure เอฟพีจีเอ (Download to FPGA) รายละเอียดมีดังนี้

1. การกำหนดพื้นที่ Reconfigurable area

เปิดซอฟต์แวร์ PReFF เจอกับหน้าแรก คือ แทบ Define area คือ การกำหนดพื้นที่ Reconfigurable part ของเอฟพีจีเอโดยแทบ Define area ขั้นตอนมีดังนี้

1.1. สร้างโฟลเดอร์สำหรับเก็บข้อมูลผลลัพธ์ที่ได้จากการทำกระบวนการ Partial reconfiguration design โดยคลิกปุ่ม Browse ใน Define save path จะปรากฏหน้าต่าง Browse folder ดังภาพประกอบ ค-1 จากนั้นเลือกโฟลเดอร์ที่ต้องการ โดยข้อกำหนดของการสร้างโฟลเดอร์นั้น คือ ชื่อโฟลเดอร์จะต้องติดกัน เว้นวรรคไม่ได้ จากนั้นเมื่อกำหนดเสร็จแล้วให้กดปุ่ม OK



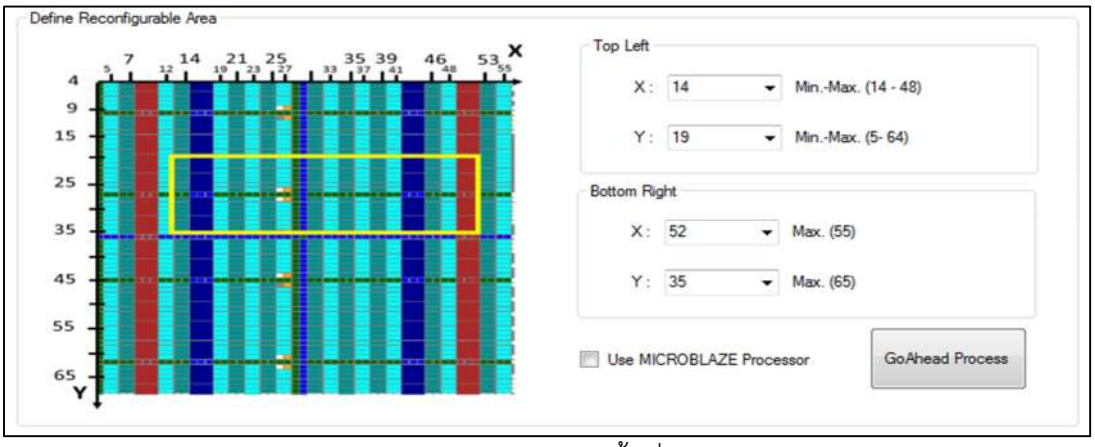
ภาพประกอบ ค-1 หน้าต่างสำหรับกำหนดโฟลเดอร์เพื่อเก็บผลลัพธ์

จากนั้นเมื่อเข้าไปดูในโฟลเดอร์ที่ได้กำหนดไว้ จะเห็นโฟลเดอร์เพิ่มขึ้นมา คือ Static และ Reconfig โฟลเดอร์

1.2. กำหนดตำแหน่งพื้นที่ ในช่อง Define reconfigurable area พื้นที่ที่ได้จะเป็นพื้นที่สี่เหลี่ยม โดยหลักการกำหนดพื้นที่ คือ เลือก SLICE ในเอฟพีจีเอ 2 ชุด โดย SLICE แรกกำหนดให้เป็นตำแหน่งมุมบนซ้าย (Top left) ของพื้นที่ Reconfigurable part และ SLICE อีกชุดกำหนดให้เป็นตำแหน่งมุมล่างขวา (Top right) ของพื้นที่ การเลือกตำแหน่ง SLICE นั้นจะเลือกในลักษณะแถวและคอลัมน์ โดยคอลัมน์หรือแกน X ประกอบไปด้วยตำแหน่งตั้งแต่ 5-55 และในแนวแถว หรือแกน Y ประกอบไปด้วยตำแหน่งตั้งแต่ 5-64 ข้อกำหนดของการเลือกตำแหน่ง SLICE มีดังนี้

- SLICE มุมบนซ้าย (Top left) จะสามารถเลือกได้ตั้งแต่ตำแหน่ง 14-48 ไม่สามารถเลือกตำแหน่งที่ต่ำกว่า 14 ได้ เนื่องจากด้านซ้ายของพื้นที่ Reconfigurable part ถูกกำหนดให้ใช้เป็น Bus macro ดังนั้น จะต้องมี SLICE ที่ติดกับขอบด้านซ้ายของพื้นที่ Reconfigurable part เพื่อทำเป็น Bus macro
- ขนาดของพื้นที่ในแนวนอน ต้องมีขนาดมากกว่า 9 หน่วย เนื่องจากมีการกำหนด SLICE 9 ชุด ทำหน้าที่เป็น Bus macro ดังนั้นการกำหนดตำแหน่ง แกน y ของ SLICE ตำแหน่ง Top left และแกน y ของ SLICE ตำแหน่ง Bottom right ต้องห่างกัน 9 หน่วย ขึ้นไป

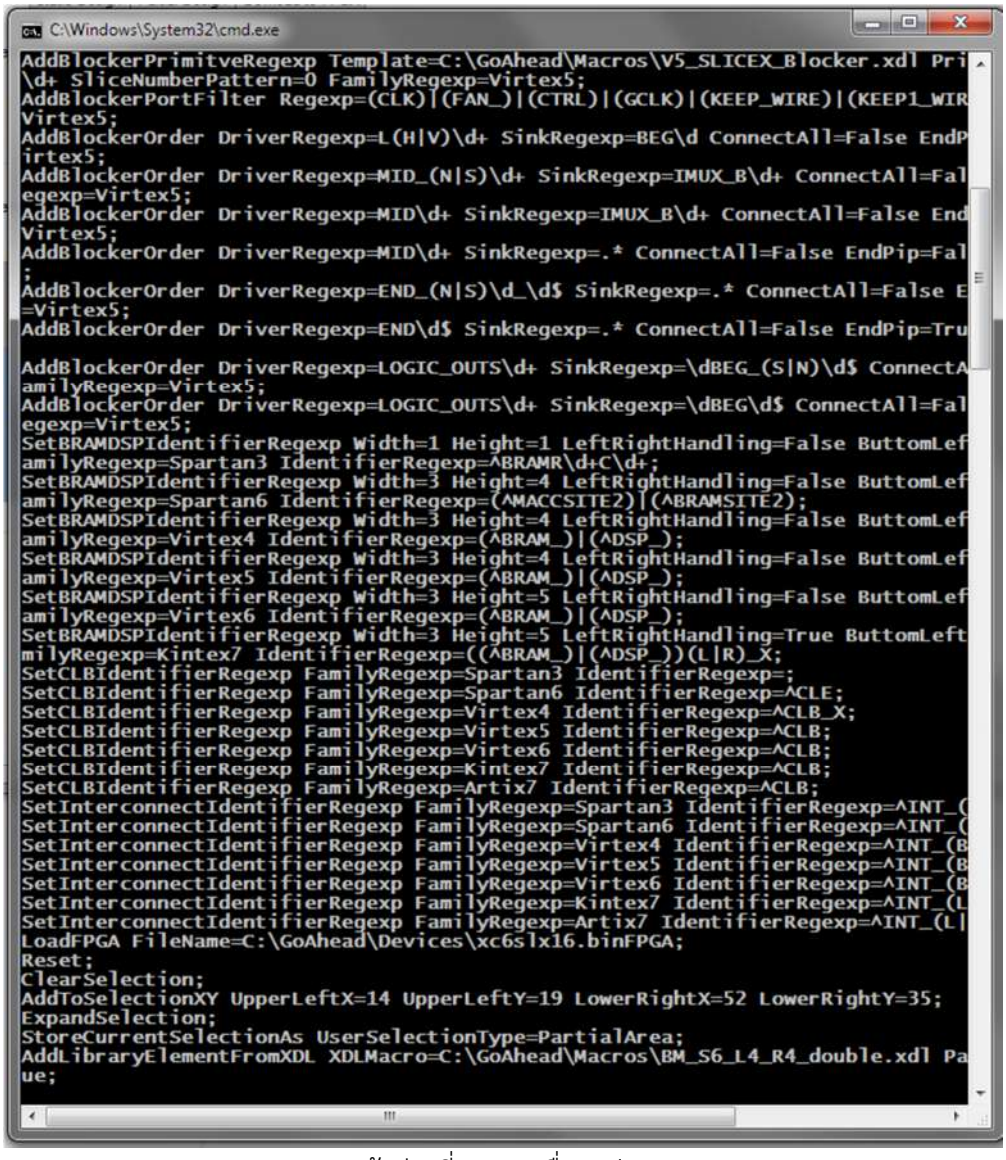
ซอฟต์แวร์นี้ได้กำหนด SLICE ที่สามารถเลือกได้ไว้แล้ว ผู้ใช้ไม่จำเป็นต้องวิเคราะห์ด้วยตัวเอง และเพื่อความสะดวกซอฟต์แวร์ได้แสดงผลกราฟฟิกให้เห็นถึงบริเวณในเอฟพีจีเอที่ผู้ใช้เลือกกำหนดพื้นที่ พร้อมกับกำหนดค่าเริ่มต้นที่เหมาะสมดังภาพประกอบ ค-2



ภาพประกอบ ค-2 การกำหนดตำแหน่งพื้นที่ Reconfigurable Part

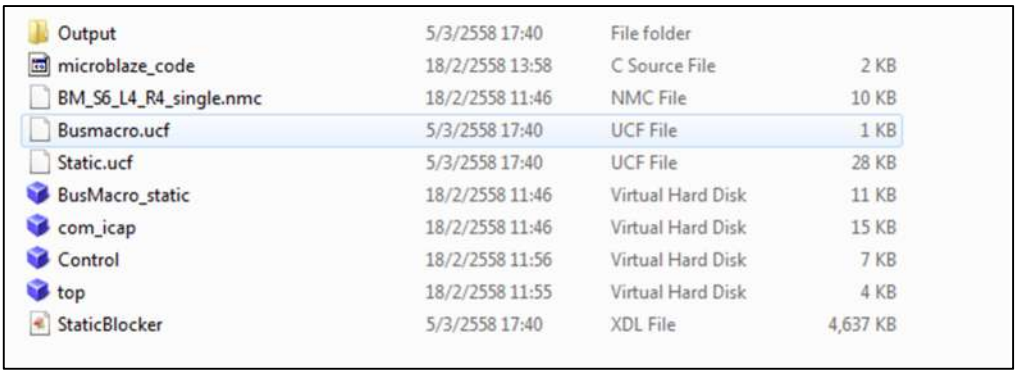
และหากผู้ใช้ต้องการออกแบบพื้นที่ Static ที่ใช้ Microblaze processor ให้คลิกเลือก Use MICROBLAZE Processor

1.3. เมื่อกำหนดพื้นที่เสร็จแล้วให้กดปุ่ม GoAhead process ซึ่งเป็นการเรียกการทำงานของซอฟต์แวร์ GoAhead พร้อมกับคำสั่งต่าง ๆ ในรูปแบบ Command line ซึ่งทำให้ปรากฏหน้าต่าง Command Prompt ดังภาพประกอบ ค-3



ภาพประกอบ ค-3 หน้าต่างที่ปรากฏเมื่อกดปุ่ม GoAhead Process

1.4. หน้าต่าง Command Prompt รันการทำงานเสร็จจะปิดตัวเอง ผลลัพธ์ที่ได้จะอยู่ในโฟลเดอร์ที่ได้เลือกไว้ในขั้นตอนก่อนหน้า โดยในโฟลเดอร์ Static จะได้ข้อมูลดังภาพประกอบ ค-4

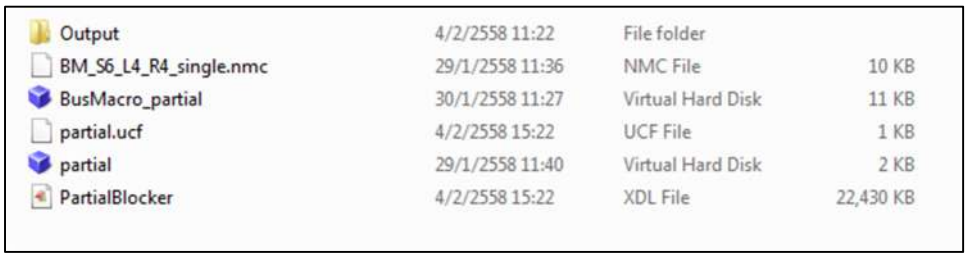


ภาพประกอบ ค-4 ผลลัพธ์ในโฟลเดอร์ Static หลังจากทำงาน GoAhead Process

โดยไฟล์ที่ได้มีดังนี้

- BM_S6_L4_R4_signle.nmc คือ ไฟล์สคริปคำสั่งของ Bus macro
- Busmacro.ucf คือ ไฟล์ที่กำหนดตำแหน่งของ Bus macro ในเอฟพีจีเอ
- Static.ucf คือ ไฟล์ UCF ของพื้นที่ Static part ซึ่งในที่นี้จะกำหนด UCF สำหรับบอร์ด SP601
- BusMacro_static.vhd คือ ไฟล์ VHDL ออกแบบวงจร Bus macro
- Com_icap.vhd คือ ไฟล์ VHDL การออกแบบวงจรควบคุม ICAP
- Control.vhd คือ ไฟล์ VHDL การออกแบบวงจรสำหรับส่งข้อมูลให้กับวงจรควบคุม ICAP
- Top.vhd คือ ไฟล์ VHDL ที่ใช้ออกแบบวงจรในพื้นที่ Static
- StaticBlocker.xdl คือ ไฟล์ XDL ที่ใช้เป็นข้อมูลในกระบวนการของซอฟต์แวร์ GoAhead
- Microblaze_code คือ ไฟล์ภาษา C เขียนโปรแกรมควบคุม Microblaze สำหรับอ่านเขียนโมดูล UART

สำหรับโฟลเดอร์ Reconfig จะได้ข้อมูลดังภาพประกอบ ค-5



ภาพประกอบ ค-5 ผลลัพธ์ในโฟลเดอร์ Reconfig หลังจากทำงาน GoAhead Process

โดยไฟล์ทีได้มีดังนี้

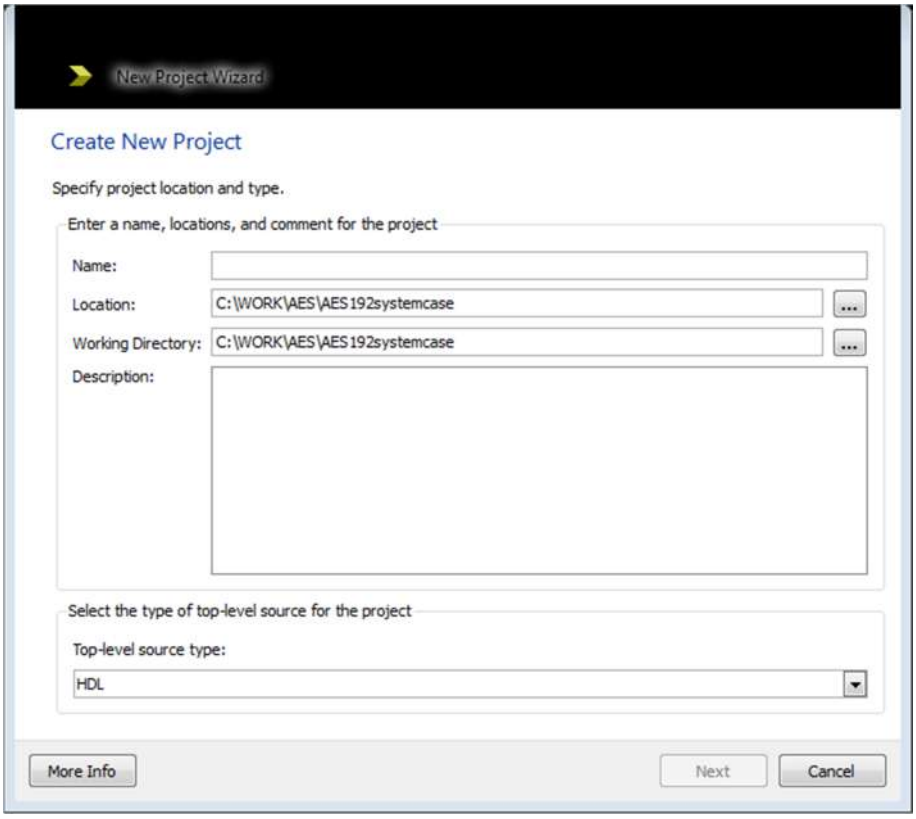
- BM_S6_L4_R4_single.nmc คือ ไฟล์สคริปคำสั่งของ Bus macro
- BusMacro_partial.vhd คือ ไฟล์ VHDL การออกแบบวงจร Bus macro
- Partial.ucf ไฟล์ UCF ของพื้นที่ Reconfig part
- Partial.vhd คือ ไฟล์ VHDL ที่ใช้ออกแบบวงจรในพื้นที่ Reconfig
- PartialBlocker.xdl คือ ไฟล์ XDL ที่ใช้เป็นข้อมูลในกระบวนการของซอฟต์แวร์ GoAhead

หากหลังจากกดปุ่ม GoAhead Process แล้วไม่ได้ผลลัพธ์ไฟล์ดังที่กล่าวข้างต้น ถือว่าเกิดความผิดพลาด ซึ่งส่งผลให้ไม่สามารถทำขั้นตอนต่อไปได้ ให้ตรวจสอบการตั้งค่าต่าง ๆ

2. การออกแบบพื้นที่ Static Part

การออกแบบพื้นที่ Static part ใช้ซอฟต์แวร์ Xilinx ISE และซอฟต์แวร์ PReFF เริ่มต้นจากสร้างโปรเจคโดย Xilinx ISE มีขั้นตอนดังนี้

2.1. เปิดโปรแกรม Xilinx ISE จากนั้นสร้าง Project โดยไปที่ File > New project จะปรากฏหน้าต่างดังภาพประกอบ ค-6 ตั้งชื่อและกำหนดตำแหน่งที่จะบันทึก และกำหนดในช่อง Top-level source type เป็น HDL จากนั้นกด Next

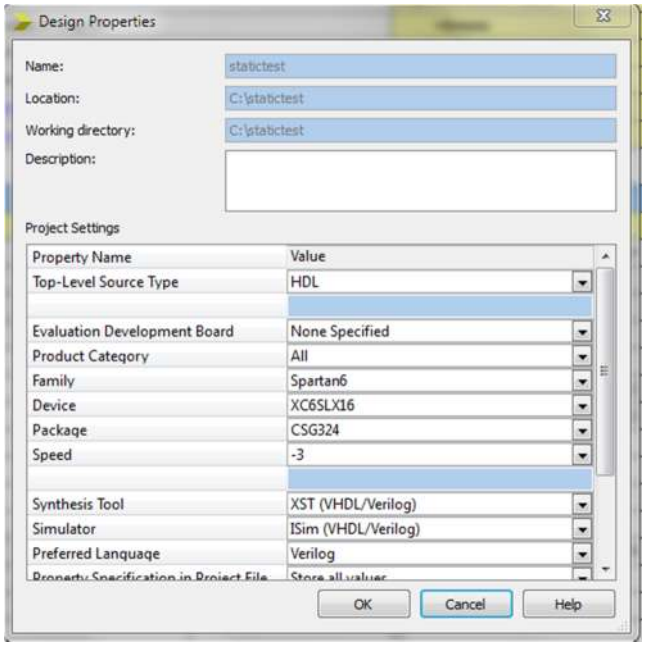


ภาพประกอบ ค-6 โปรแกรม Xilinx ISE

2.2. กำหนดคุณสมบัติของบอร์ดเอฟพีจีเอ ที่จะใช้งาน โดยจะต้องกำหนดรายละเอียดดังนี้

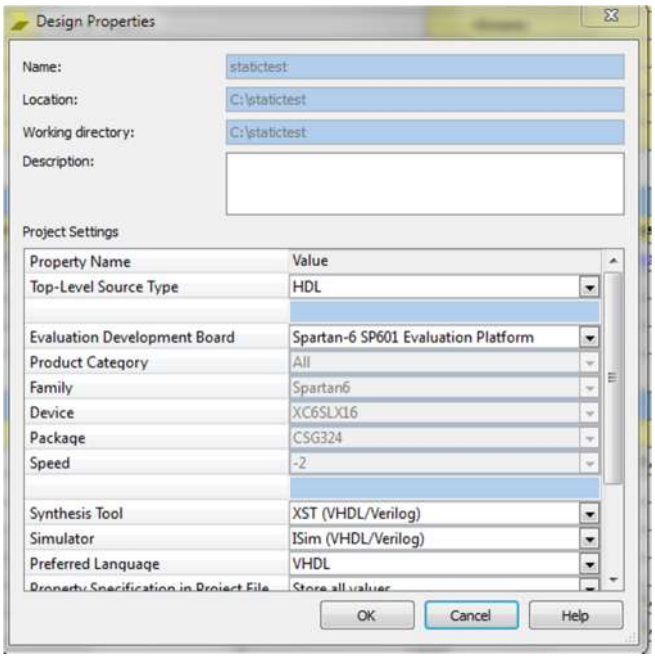
- Family : Spartan-6
- Device : SC6SLX16
- Package : CSG324
- Speed : -2

เมื่อกำหนดเสร็จ จะได้ดังภาพประกอบ ค-7 จากนั้นเลือก Next > Finish จะได้ Project ที่พร้อมใช้งานต่อไป



ภาพประกอบ ค-7 การกำหนดคุณสมบัติของบอร์ดเอฟพีจีเอ

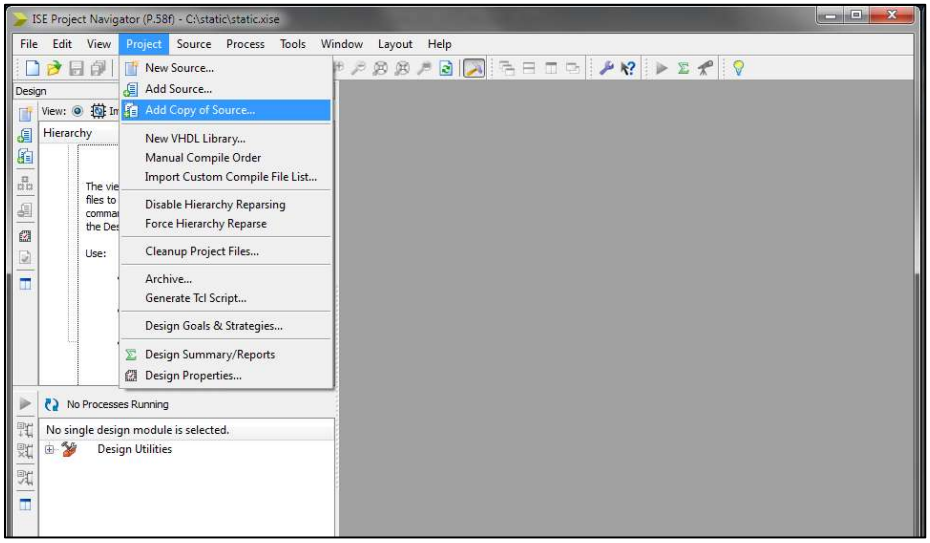
นอกจากนี้สามารถกำหนดจาก Evaluatio development board : Spartan-6 SP601 Evaluation platform ดังภาพประกอบ ค-8



ภาพประกอบ ค-8 กำหนดคุณสมบัติของบอร์ดจาก Platform

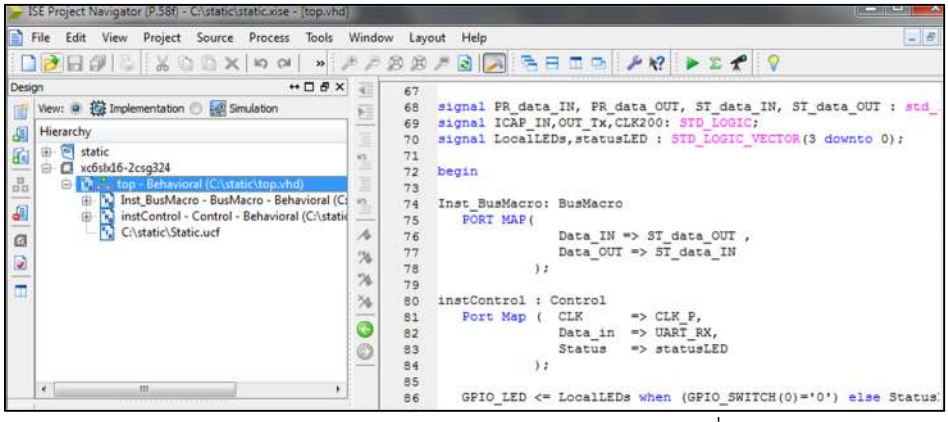
หลังจากนั้นจะได้หน้าต่างโปรเจค ที่ยังไม่มีข้อมูลใด ๆ ให้ทำการ Import file ที่จำเป็นสำหรับพื้นที่ Static part ในขั้นตอนถัดไป

2.3. ในหน้าต่างซอฟต์แวร์ Xilinx ISE ไปที่ Project > Add Copy of Source ดังภาพประกอบ ค-9 จากนั้นให้เลือกไฟล์ที่ได้จากการสร้างในกระบวนการ GoAhead Process ซึ่งจะอยู่ในโฟลเดอร์ Static โดยเลือกไฟล์ Static.ucf, BusMacro_static.vhd, com_icap.vhd, Control.vhd, top.vhd และ BM_S6_L4_R4_single.nmc



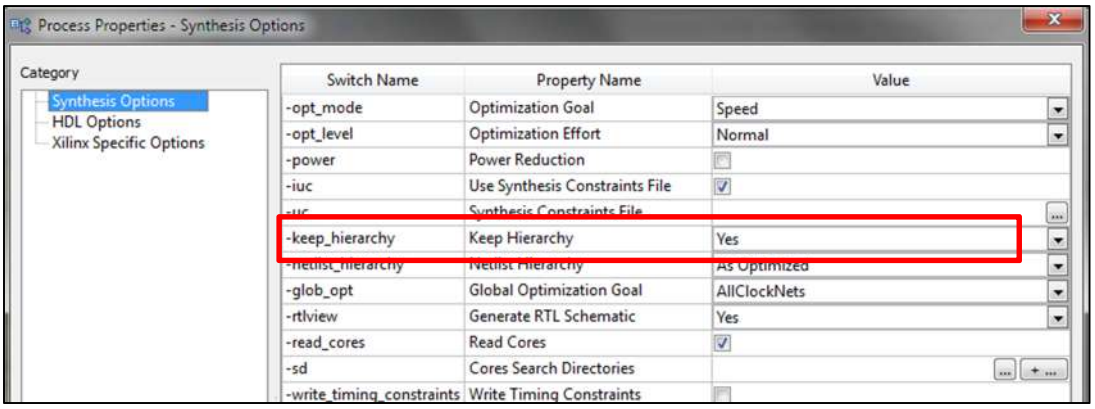
ภาพประกอบ ค-9 การเพิ่มไฟล์ในซอฟต์แวร์ Xilinx ISE

2.4. ผลลัพธ์หลังจากได้ทำการ Import ข้อมูลมาแล้วจะปรากฏดังภาพประกอบ ค-10 ซึ่งจะเห็นได้ว่าโมดูล Top ประกอบไปด้วยโมดูลย่อย 2 ส่วน คือ Bus macro และ Control หากผู้ใช้ต้องการออกแบบเพิ่มเติม ให้เปิดไฟล์โมดูล Top แล้วสามารถเพิ่มหรือแก้ไขวงจรได้ตามต้องการ



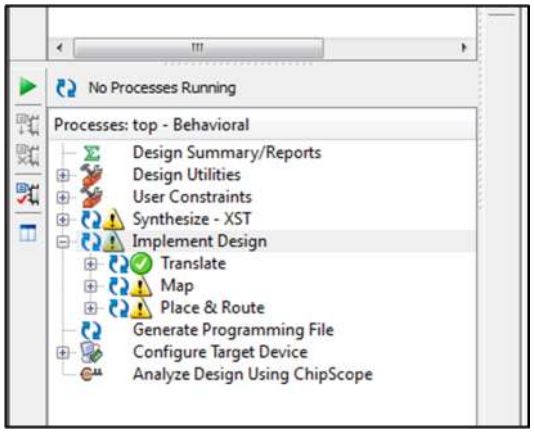
ภาพประกอบ ค-10 ซอฟต์แวร์ Xilinx ISE หลังจากได้เพิ่มข้อมูลไฟล์

2.5. เนื่องจากมีการใช้ Bus macro ดังนั้นจำเป็นต้องตั้งค่าเพิ่มเติม โดยไปที่ Process > Process properties ในช่อง Category ให้เลือก Synthesis option จากนั้นในหัวข้อ Switch Name > -keep_hierarchy ให้เลือก Yes ดังภาพประกอบ ค-11



ภาพประกอบ ค-11 กำหนดคุณสมบัติขั้นตอน Synthesis เพิ่มเติม

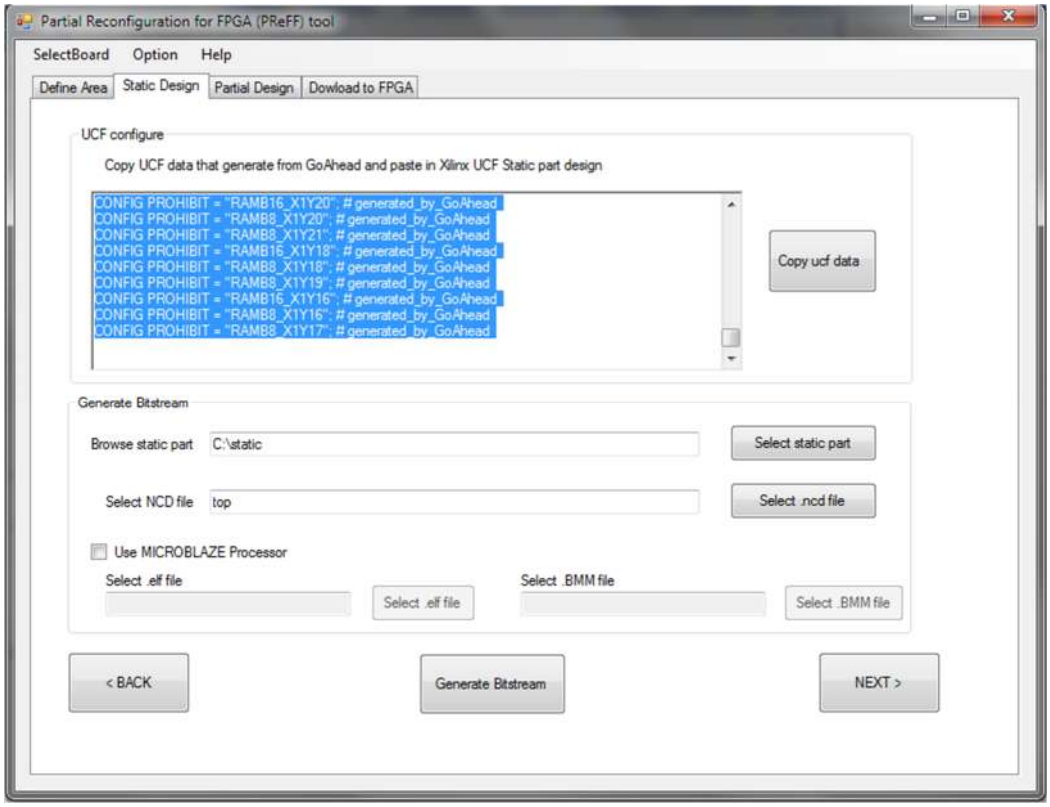
2.6. เมื่อออกแบบเสร็จแล้ว ทำกระบวนการสร้างไฟล์วงจรโดยไปที่ Design tab คลิก Synthesis – XST > Implement Design หากไม่มีข้อผิดพลาดจะได้ดังภาพประกอบ ค-12



ภาพประกอบ ค-12 กระบวนการสร้างไฟล์วงจรของซอฟต์แวร์ Xilinx ISE

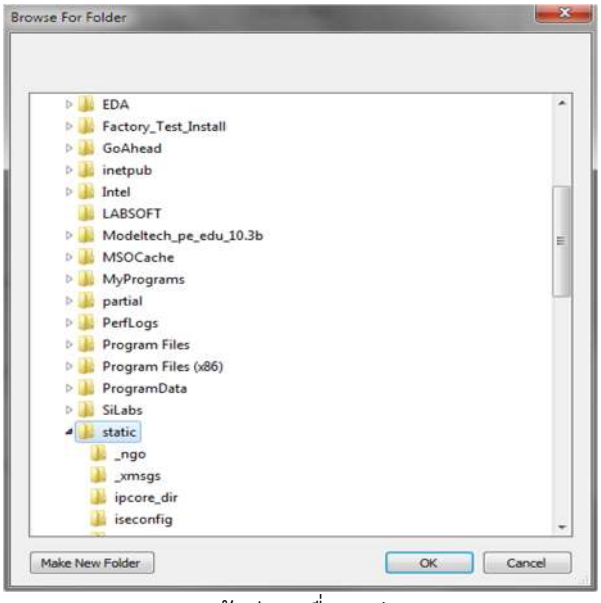
เมื่อเสร็จกระบวนการออกแบบวงจรแล้วให้กลับไปที่ซอฟต์แวร์ PReFF ซึ่งหลังจากที่ออกแบบพื้นที่แล้วให้กดปุ่ม Next เพื่อไปยัง Static design tab

2.7. หน้าต่างใน Static design tab แสดงดังภาพประกอบ ค-13 ประกอบไปด้วย UCF config และ Generate bistream โดยใน UCF config จะแสดงให้เห็น UCF data ของพื้นที่ Static part ผู้ใช้สามารถ Copy UCF ดังกล่าวไปสร้างไฟล์ .ucf ในกระบวนการออกแบบ โดย Xilinx ISE ได้ ซึ่งในที่นี้ไม่จำเป็นต้องใช้ เนื่องจากข้อมูลทั้งหมดถูกเก็บไว้ใน Static.ucf แล้ว



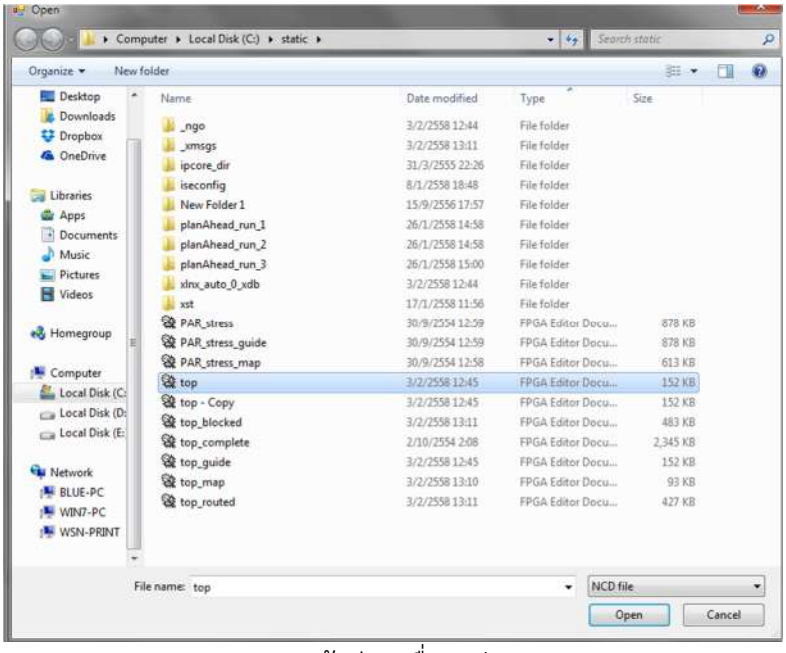
ภาพประกอบ ค-13 หน้าต่าง Static Design

2.8. ขั้นตอนต่อมาเลือก Select static part ซึ่งจะต้องเลือกไปยังโฟลเดอร์ที่ได้ออกแบบในขั้นตอนของ Xilinx ISE โดยคลิก Select static part จะปรากฏหน้าต่างดังภาพประกอบ ค-14 ในที่นี้ได้กำหนดชื่อโฟลเดอร์ที่สร้างโปรเจคไว้ คือ static จากนั้นคลิก OK



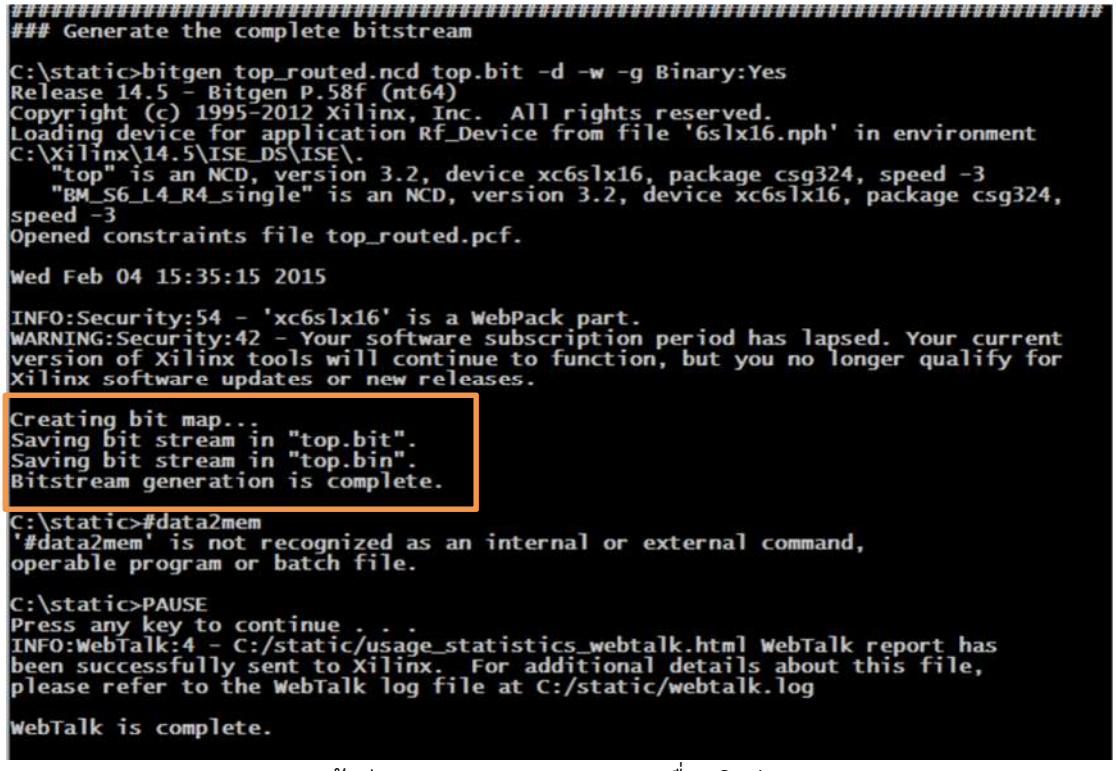
ภาพประกอบ ค-14 หน้าต่าง เมื่อกดปุ่ม Select Static Part

2.9. เลือกไฟล์ .ncd โดยคลิกปุ่ม select .ncd file จะปรากฏหน้าต่างดังภาพประกอบ ค-15 ให้เลือกไฟล์ .ncd ที่ได้จากกระบวนการสร้างโดย Xilinx ISE process ซึ่งชื่อไฟล์จะตรงกับชื่อไฟล์ Top module โดยในที่นี้ตั้งชื่อ คือ top



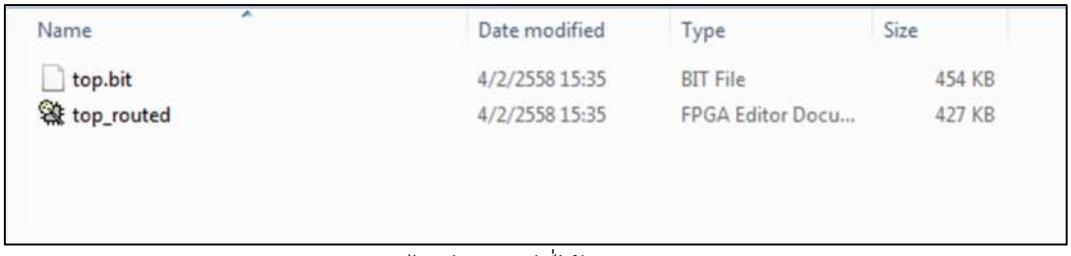
ภาพประกอบ ค-15 หน้าต่าง เมื่อกดปุ่ม Select .ncd File

2.10. เมื่อกำหนดไฟล์ครบแล้วให้คลิกปุ่ม Generate bitstream จะปรากฏหน้าต่าง Command Prompt ขึ้นเพื่อรันคำสั่งในการสร้างไฟล์ Bitstream สำหรับพื้นที่ Static part เมื่อรันเสร็จจะแสดงผลดังภาพประกอบ ค-16 ซึ่งจะแสดงผล Bitstream generation is complete.



ภาพประกอบ ค-16 หน้าต่าง Command Prompt เมื่อคลิกปุ่ม Generate Bitstream

2.11. ผลลัพธ์ที่ได้จะถูกเก็บไว้ในโฟลเดอร์ ~/Static/Output ที่ได้กำหนดไว้ในขั้นตอนการออกแบบพื้นที่ ดังภาพประกอบ ค-17 โดยไฟล์ที่ได้ คือ top.bit ซึ่งเป็นข้อมูล Bitstream ที่สามารถ Reconfigure FPGA ได้ และไฟล์ top_routed.ncd ซึ่งเป็นไฟล์ที่แสดงการวางตำแหน่งและการกำหนดเส้นทางใน FPGA ของวงจร Static part ผู้ใช้สามารถเปิดไฟล์นี้ได้โดยซอฟต์แวร์ FPGA editor



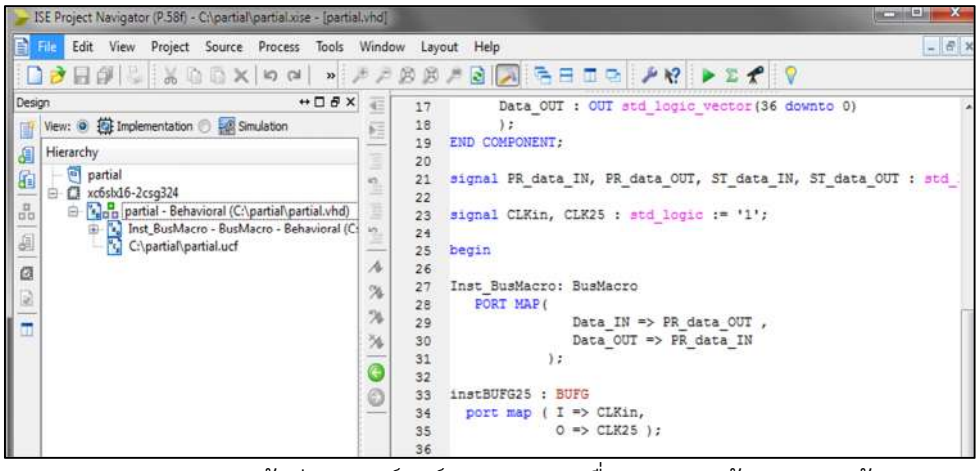
ภาพประกอบ ค-17 ไฟล์ผลลัพธ์ที่ได้หลังจากรัน Static Process

3. การออกแบบพื้นที่ Reconfigurable Part

การออกแบบพื้นที่ Reconfigurable part ใช้ซอฟต์แวร์ Xilinx ISE และซอฟต์แวร์ PReFF เริ่มต้นสร้างโปรเจคใหม่อีกครั้งโดยซอฟต์แวร์ Xilinx ISE จากนั้น Import file ที่จำเป็นสำหรับพื้นที่ Reconfigurable part ดังนี้

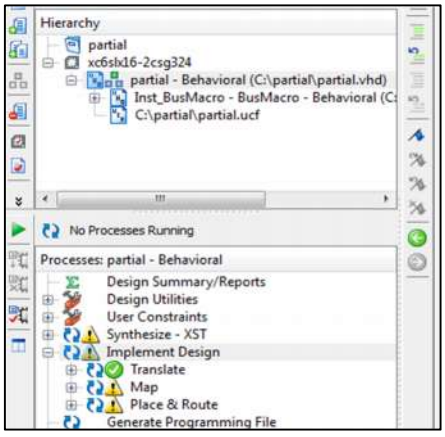
3.1. ในหน้าต่างซอฟต์แวร์ Xilinx ISE ไปที่ Project > Add Copy of Source... จากนั้นให้เลือกไฟล์ที่ได้จาก GoAhead Process ในโฟลเดอร์ Reconfig โดยเลือกไฟล์ partial.ucf, BusMacro_partial.vhd, BM_S6_L4_R4_single.nmc

3.2. ผลลัพธ์หลังจากได้ทำการ Import ข้อมูลมาแล้วจะปรากฏดังภาพประกอบ ค-18 ซึ่งจะเห็นได้ว่าโมดูล partial ซึ่งเป็น Top module ประกอบไปด้วยโมดูลย่อย คือ Bus macro หากผู้ใช้ต้องการออกแบบเพิ่มเติม ให้เปิดไฟล์โมดูล partial แล้วสามารถเพิ่มหรือแก้ไขวงจรได้ตามต้องการ



ภาพประกอบ ค-18 หน้าต่างซอฟต์แวร์ Xilinx ISE เมื่อ Import ข้อมูลวงจรเข้ามา

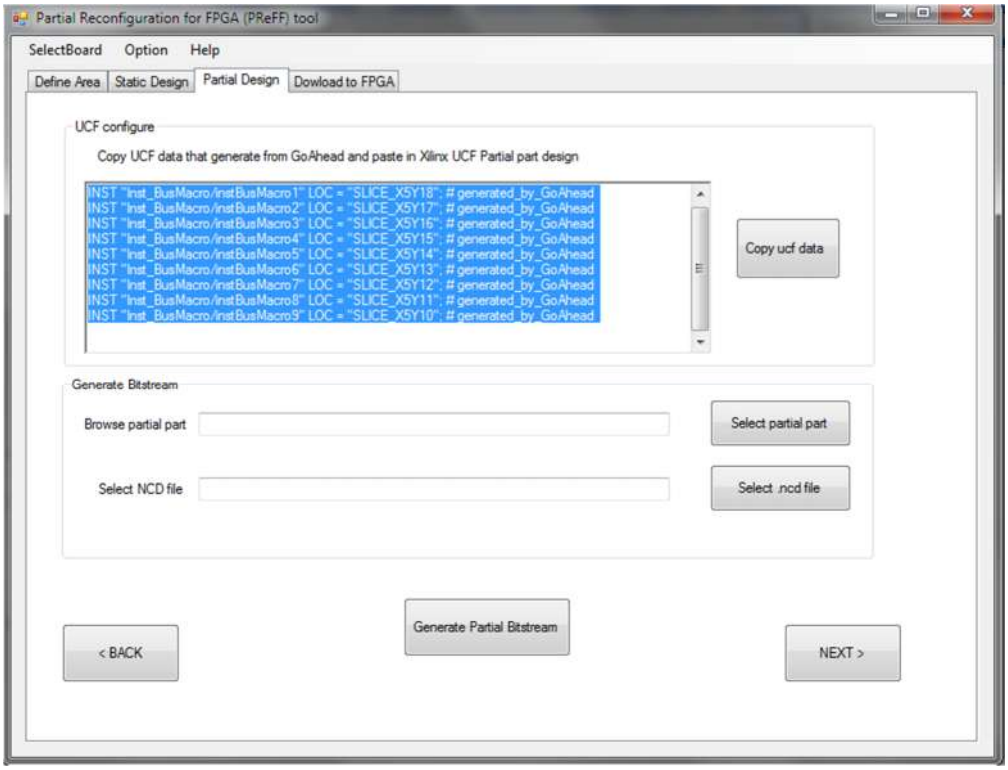
เมื่อออกแบบเสร็จแล้ว ทำกระบวนการสร้างไฟล์วงจรโดยไปที่ Design tab คลิก Synthesis – XST > Implement design หากไม่มีข้อผิดพลาดจะได้ดังภาพประกอบ ค-19



ภาพประกอบ ค-19 กระบวนการสร้างไฟล์วงจรของซอฟต์แวร์ Xilinx ISE

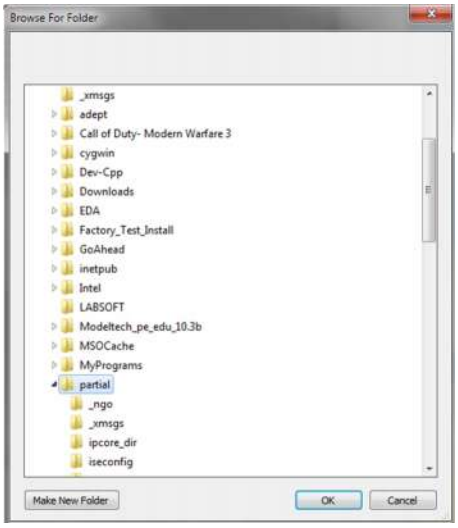
เมื่อเสร็จกระบวนการออกแบบวงจรแล้วให้กลับไปที่ซอฟต์แวร์ PReFF ซึ่งหลังจากที่ออกแบบ Static Design แล้วให้กดปุ่ม Next เพื่อไปยัง Partial design tab

3.3. หน้าต่างใน Partial design tab แสดงดังภาพประกอบ ค-20 ประกอบไปด้วย UCF config และ Generate bistream group โดยใน UCF config group จะแสดงให้เห็น UCF ของพื้นที่ Reconfigurable part ผู้ใช้สามารถ Copy UCF ดังกล่าวไปสร้างไฟล์ .ucf ในกระบวนการออกแบบโดย Xilinx ISE ได้ ซึ่งในที่นี้ไม่จำเป็นต้องใช้ เนื่องจากข้อมูลทั้งหมดถูกเก็บไว้ใน partial.ucf แล้ว



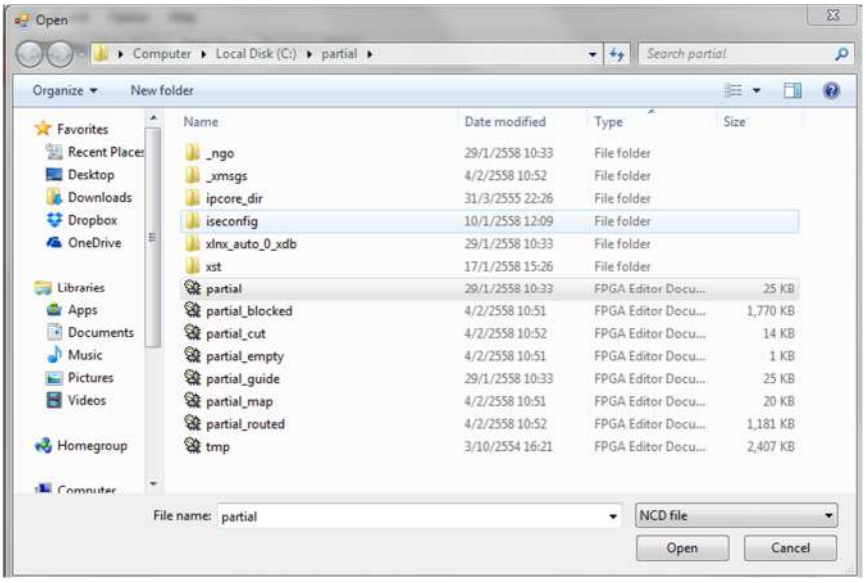
ภาพประกอบ ค-20 หน้าต่าง Partial Design

3.4. ขั้นตอนต่อมา คือ การเลือกโฟลเดอร์ที่ใช้ออกแบบ Reconfigurable part โดยซอฟต์แวร์ Xilinx ISE ในขั้นตอนก่อนหน้า โดยการคลิก Select partial part จะปรากฏหน้าต่างดังภาพประกอบ ค-21 โดยในที่นี้กำหนดให้สร้างไว้ในโฟลเดอร์ Partial



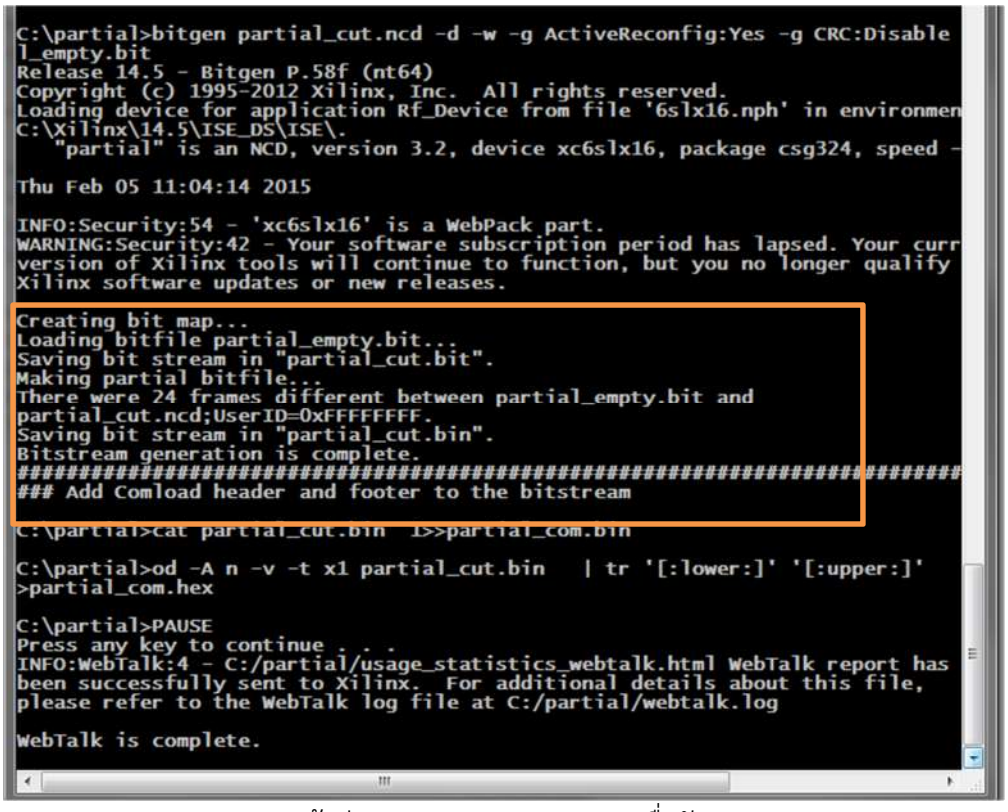
ภาพประกอบ ค-21 หน้าต่างเลือก Partial Design Path

3.5. ขั้นตอนถัดมาคลิก Select .ncd file เพื่อเลือกไฟล์ .ncd ที่ได้จากการกระบวนการสร้างไฟล์วงจรของ Reconfigurable part ดังภาพภาพประกอบ ค-22 โดยจะเลือก .ncd ที่ตรงกับชื่อ Top module ซึ่งในที่นี้กำหนดให้ Top module ชื่อ Partial ดังนั้นจะเลือกไฟล์ partial.ncd



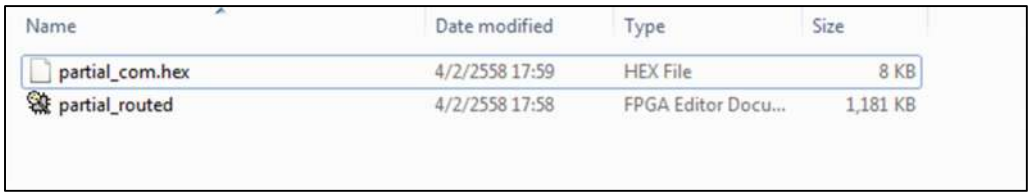
ภาพประกอบ ค-22 หน้าต่าง Select .ncd File

3.6. เมื่อกำหนดครบแล้วคลิก Generate partial bitstream จะปรากฏหน้าต่าง Command Prompt ขึ้นมาเพื่อรันคำสั่งในการสร้างไฟล์ Partial Bitstream ดังภาพประกอบ ค-23 เมื่อเสร็จกระบวนการจะเห็นข้อความ Bitstream generation is complete คือ สามารถสร้างไฟล์วงจรได้สำเร็จ



ภาพประกอบ ค-23 หน้าต่าง Command Prompt เมื่อรัน Partial process

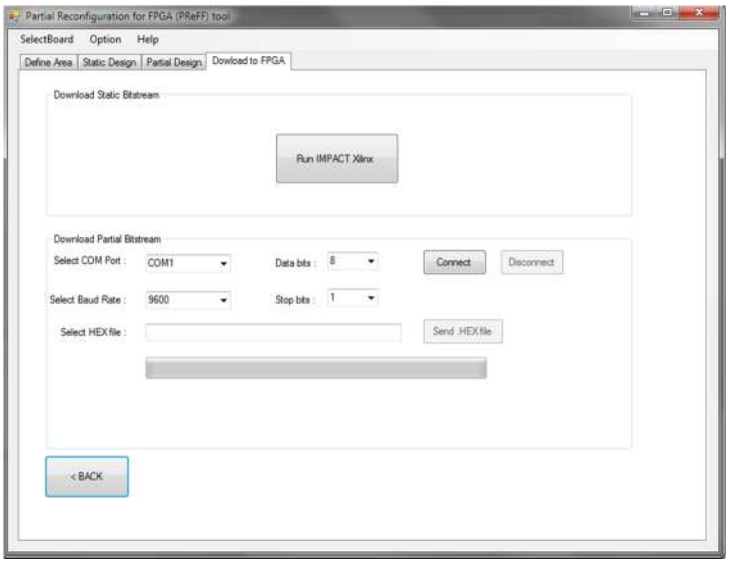
3.7. ผลลัพธ์ที่ได้จะถูกเก็บไว้ในโฟลเดอร์ที่ได้กำหนดไว้ในขั้นตอนการออกแบบพื้นที่ ซึ่งจะอยู่ใน ~/Reconfig/Output ผลลัพธ์ประกอบไปด้วยไฟล์ดังภาพประกอบ ค-24 โดย partial_com.hex คือ .HEX file ที่จะใช้ Reconfigure ในพื้นที่ Reconfigurable part และ partial_routed.ncd คือ ไฟล์แสดงการกำหนดเส้นทางและการวางตำแหน่งของวงจรในพื้นที่ Reconfigurable part



ภาพประกอบ ค-24 ผลลัพธ์จากการรัน Partial Process

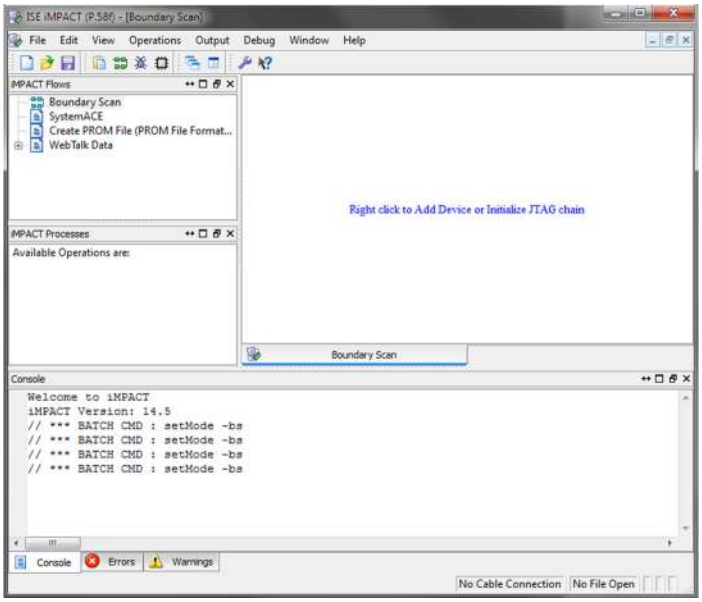
4. การ Reconfigure เอฟพีจีเอ (Download to FPGA)

หลังจากเสร็จขั้นตอนการสร้าง Partial design แล้ว คลิก Next จะเข้าสู่ Download to FPGA tab ดัง ภาพประกอบ ค-25 การ Reconfigure FPGA จะเริ่มจากพื้นที่ Static part ก่อน จากนั้นจึง Reconfigure พื้นที่ Reconfigurable part ได้



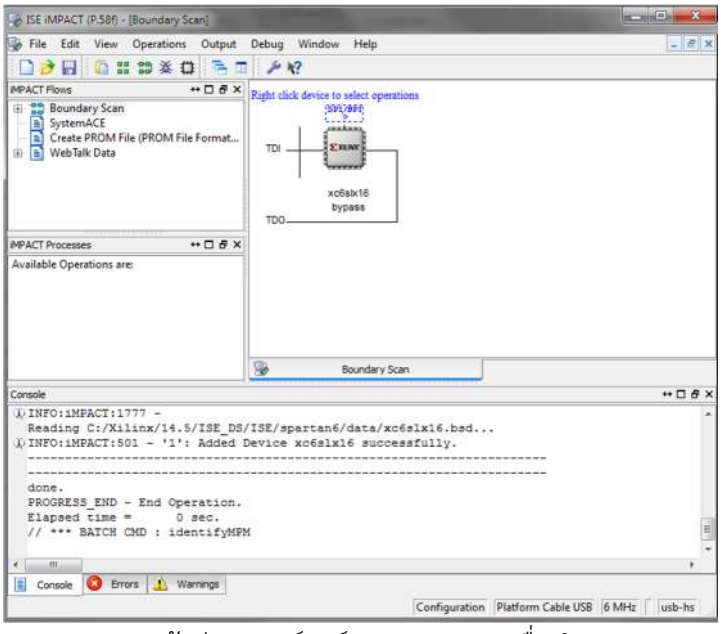
ภาพประกอบ ค-25 หน้าต่าง Download to FPGA

4.1. คลิกปุ่ม Run IMPACT Xilinx เพื่อเปิดซอฟต์แวร์ Xilinx ISE iMPACT ผลลัพธ์จะได้ดังภาพประกอบ ค-26



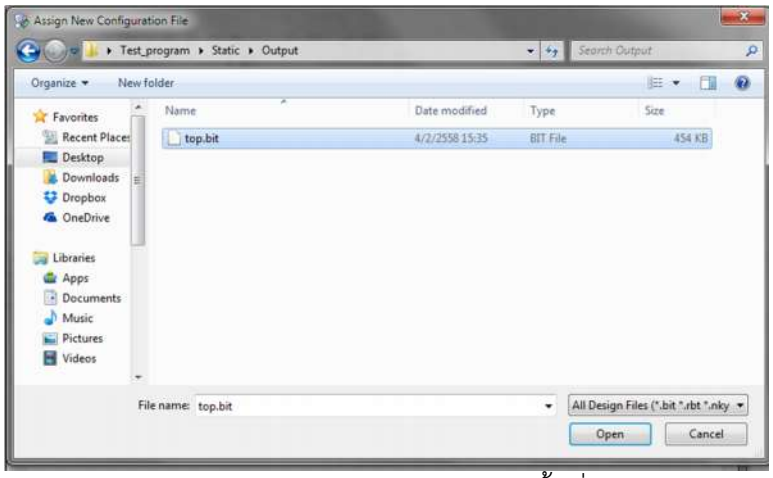
ภาพประกอบ ค-26 หน้าต่างซอฟต์แวร์ Xilinx ISE iMPACT

4.2. จากนั้นใน iMPACT Flows ให้คลิกที่ Boundary scan แล้วไปที่ File > Initialize Chain เพื่อค้นหาเอฟพีจีเอที่กำลังเชื่อมต่อกับคอมพิวเตอร์ หากผู้ใช้ต่อบอร์ด SP601 และจ่ายไฟกับบอร์ดแล้ว ผลลัพธ์จะได้เอฟพีจีเอ Xilinx xc6slx16 ดังภาพประกอบ ค-27



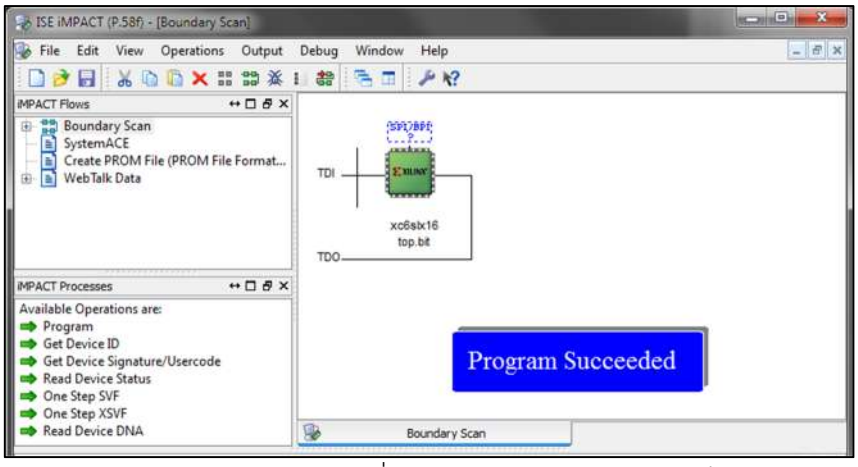
ภาพประกอบ ค-27 หน้าต่างซอฟต์แวร์ ISE iMPACT เมื่อทำการ Initialize Chain

4.3. คลิกขวาในรูปเอฟพีจีเอ จากนั้นเลือก Assign new configuration file... จะปรากฏหน้าต่างให้เลือกไฟล์ .bit ซึ่งเป็นผลลัพธ์ที่อยู่ใน ~/Static/Output โดยในภาพประกอบ ค-28 ใช้ไฟล์ Top.bit จากนั้นกด Open



ภาพประกอบ ค-28 เลือก Bitstream File พื้นที่ Static Part

4.4. ลำดับต่อมาให้คลิกขวาที่รูปเอฟพีจีเอ เลือก Program ซอฟต์แวร์จะทำการ Reconfigure เอฟพีจีเอ หากทำสำเร็จจะปรากฏหน้าต่างดังภาพประกอบ ค-29 โดยขึ้นข้อความ Program Succeedded



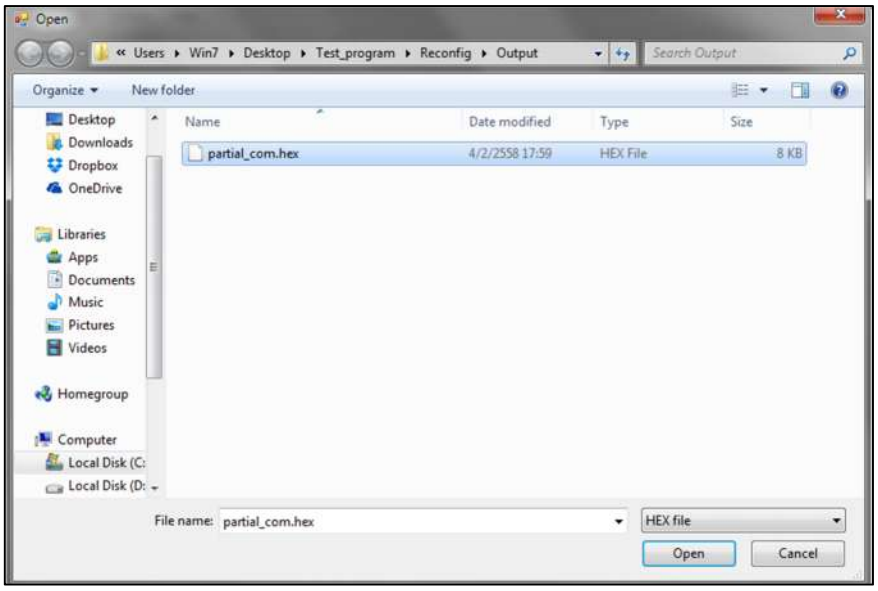
ภาพประกอบ ค-29 เมื่อ Reconfigure เอฟพีจีเอเสร็จ

ในขณะนี้ เอฟพีจีเอสามารถทำงานได้แล้วตามการออกแบบวงจรในพื้นที่ Static part โดยผู้ออกแบบได้กำหนดให้แสดงผล ON LED1-LED4 บนบอร์ด

4.5. Reconfigure พื้นที่ Reconfigurable part จะส่งข้อมูลวงจรผ่านทาง UART COM port โดยผู้ใช้ทำการกำหนดหมายเลข Comport ในช่อง Select COM Port และขณะนี้ยังไม่สามารถปรับเปลี่ยน Baud rate ได้ จึงกำหนดให้ใช้ Baud rate คงที่ คือ 9,600 bps จากนั้นคลิก Connect ผลลัพธ์จะได้ดังภาพประกอบ ค-30

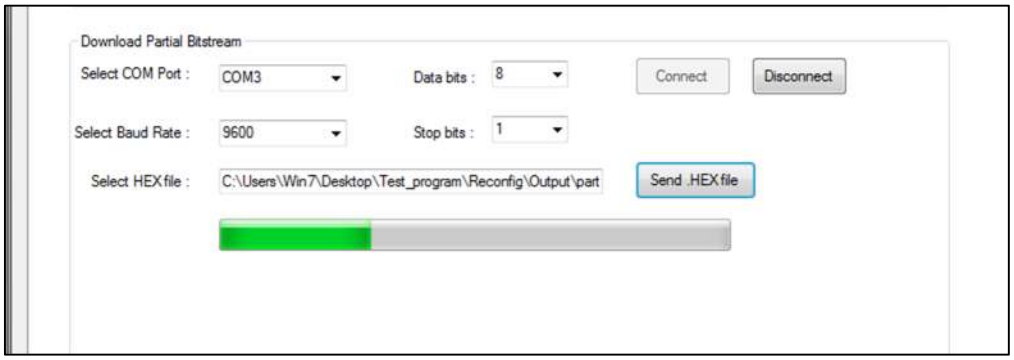
ภาพประกอบ ค-30 การเชื่อมต่อกับ Com Port

4.6. จากนั้นเลือกไฟล์ที่ต้องการ Reconfigure โดยคลิก Send.HEX file จะปรากฏหน้าต่างภาพประกอบ ค-31 ให้เลือกไฟล์ที่อยู่ใน ~/Reconfig/Output ซึ่งเป็นไฟล์นามสกุล .hex จากนั้นคลิก Open



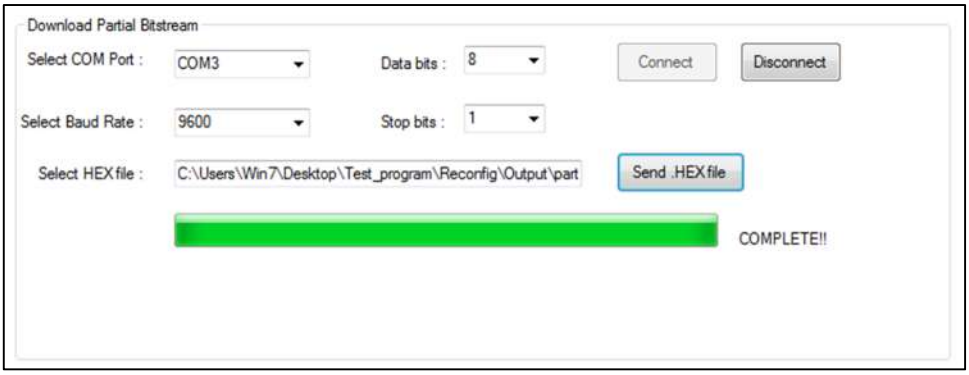
ภาพประกอบ ค-31 หน้าต่างเลือก .HEX file

4.7. เมื่อคลิก Open ซอฟต์แวร์จะเริ่ม Reconfigure เอฟพีจีเอโดยแสดงสถานะ Reconfigure ดังภาพประกอบ ค-32



ภาพประกอบ ค-32 สถานะขณะ Reconfigure วงจรของพื้นที่ Reconfigurable Part

4.8. เมื่อเสร็จกระบวนการ Reconfigure จะแสดงสถานะ COMPLETE ดังภาพประกอบ ค-33

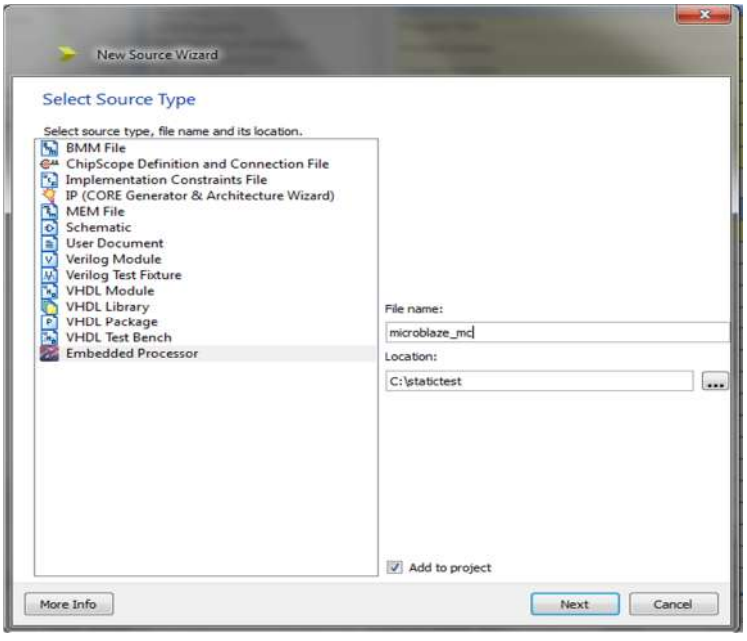


ภาพประกอบ ค-33 สถานะเมื่อ Reconfigure วงจรของพื้นที่ Reconfigurable Part เสร็จสิ้น

5. การใช้งาน Microblaze สำหรับการทำ Partial Reconfiguration

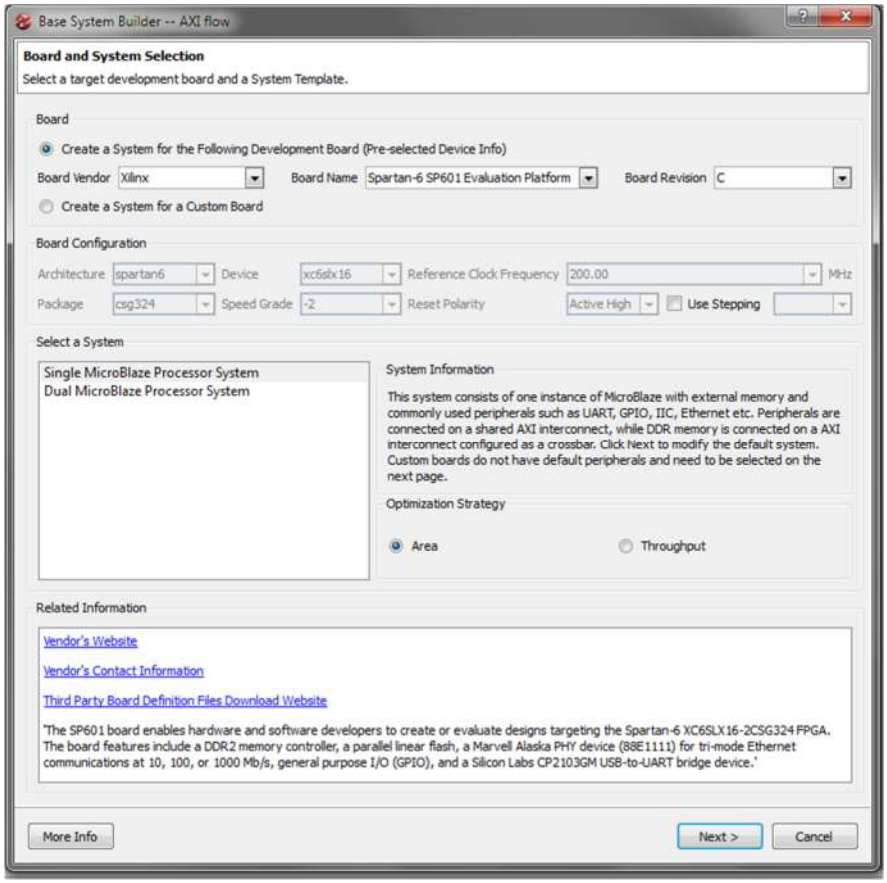
Microblaze จะถูกเพิ่มในพื้นที่ Static part ดังนั้นกระบวนการที่เพิ่มเติมจะทำในกระบวนการออกแบบพื้นที่ Static part เท่านั้น โดยให้สร้างโปรเจค Xilinx ISE ใหม่แล้วทำขั้นตอนดังต่อไปนี้

5.1. ในหน้าต่างซอฟต์แวร์ Xilinx ISE เลือก Project > Add new source ดังภาพประกอบ ค-34 ในช่อง Select source type ให้เลือก Embedded processor พร้อมกับกำหนดชื่อจากนั้นกด Next > Finish



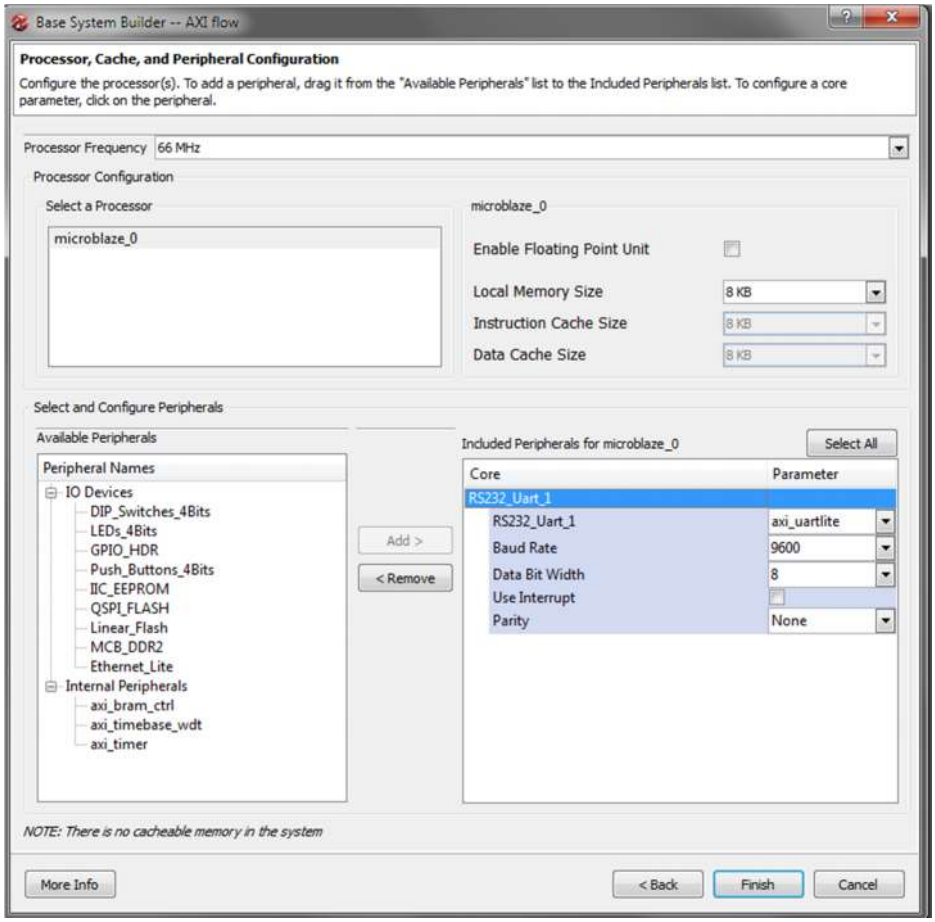
ภาพประกอบ ค-34 หน้าต่าง New Source สำหรับสร้าง Microblaze

5.2. ซอฟต์แวร์จะทำการสร้างโมดูล Microblaze ขึ้นมา โดยจะเรียกซอฟต์แวร์ Xilinx Platform Studio (XPS) เพื่อให้ผู้ใช้งานกำหนดคุณสมบัติของ Microblaze ดังภาพประกอบ ค-35 โดยค่าเริ่มต้นกำหนดคุณสมบัติบอร์ดตามที่กำหนดไว้ใน Xilinx ISE จากนั้นเลือก Single microblaze processor System และ Optimization strategy : Area จากนั้นคลิก Next



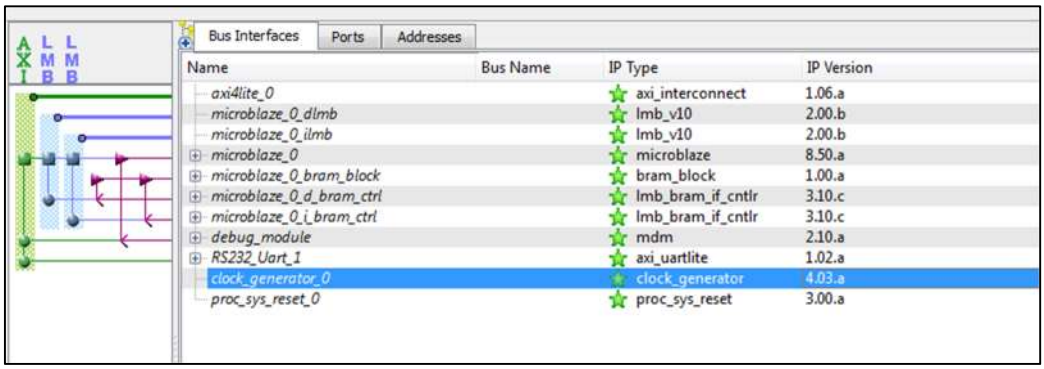
ภาพประกอบ ค-35 หน้าต่าง Base System Builder จากซอฟต์แวร์ Xilinx XPS

5.3. จากนั้นกำหนดคุณสมบัติ Peripheral ที่ต้องการใช้งาน ในที่นี้เลือกใช้เฉพาะ RS-232 UART ดังนั้นในช่อง Include peripherals for Microblaze_0 จะมีเพียง RS23_Uart_1 สำหรับโมดูลอื่น ๆ ให้คลิกที่โมดูลแล้วคลิก Remove ผลลัพธ์จะได้ดังภาพประกอบ ค-36 อย่างไรก็ตาม หากผู้ใช้งานต้องการใช้งานส่วนอื่น ๆ สามารถคลิกเลือกได้เช่นกัน จากนั้นคลิก Finish เพื่อให้ซอฟต์แวร์ทำการ Generate ไฟล์ข้อมูลที่เกี่ยวข้องทั้งหมด



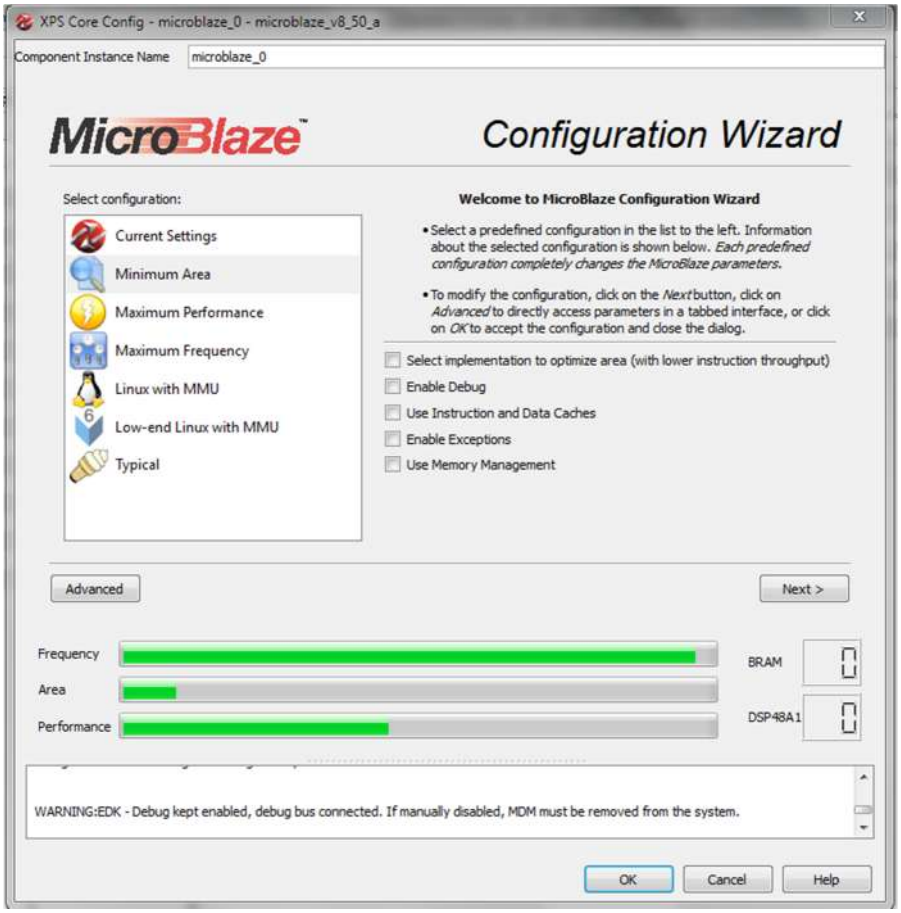
ภาพประกอบ ค-36 กำหนด Peripheral Microblaze

5.4. เมื่อทำการ Generate เสร็จแล้วจะเข้าสู่หน้าต่างภาพประกอบ ค-37 ข้อมูลประกอบไปด้วย Bus interface tab แสดงให้เห็นการเชื่อมต่อระหว่างโมดูลต่าง ๆ ผ่าน Bus interface แบบ AXI และ LMB, Port tab แสดงให้เห็นอินพุตเอาต์พุตของแต่ละโมดูล, Address Tab แสดงตำแหน่งของโมดูลที่เก็บในหน่วยความจำ



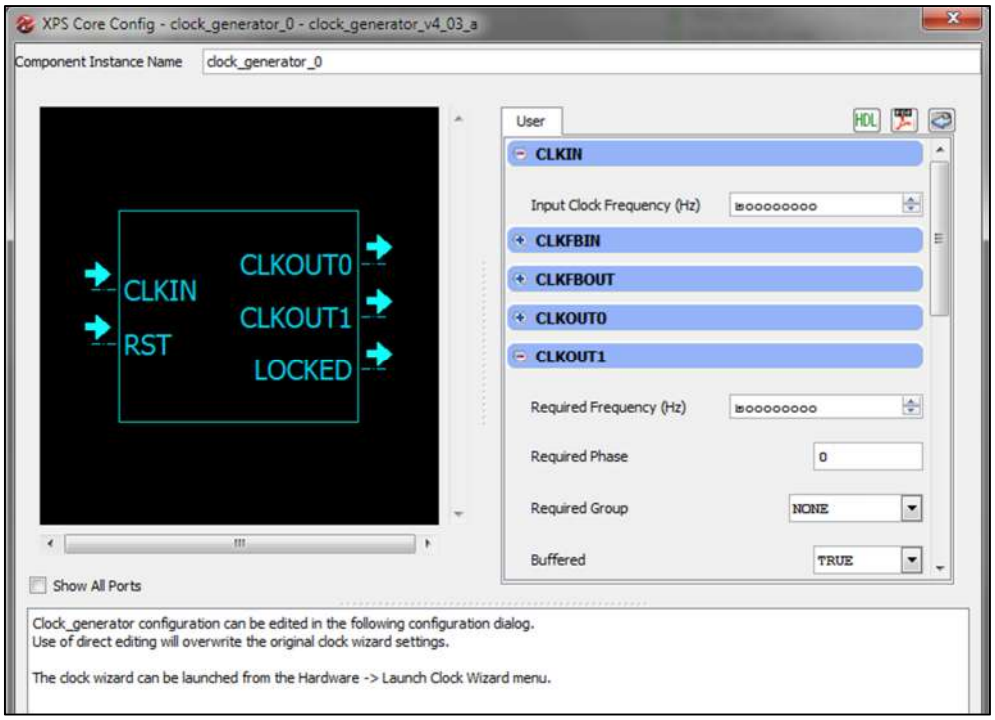
ภาพประกอบ ค-37 กำหนดคุณสมบัติ Microblaze โดยซอฟต์แวร์ Xilinx XPS

5.5. กำหนดคุณสมบัติของ Microblaze โดยพิจารณาในแทบ Bus interface เลือก microblaze_0 คลิกขวาเลือก Configure IP จะปรากฏหน้าต่าง XPS core configure ดังภาพประกอบ ค-38 โดยในช่อง Select configuration ให้เลือก Minimum area เพื่อให้ใช้พื้นที่น้อยที่สุด และให้คลิกไม่เลือก Enable debug อย่างไรก็ตามคุณสมบัติเหล่านี้ ผู้ใช้สามารถเปลี่ยนแปลงได้ จากนั้นกด OK



ภาพประกอบ ค-38 กำหนดคุณสมบัติ Microblaze

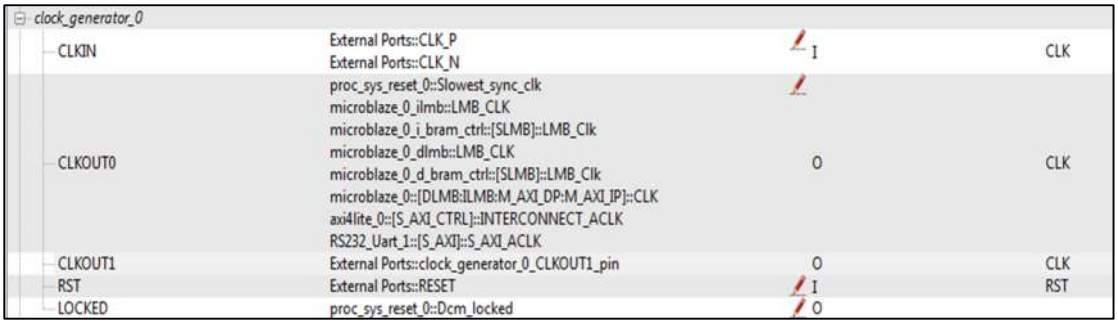
5.6. กลับมายังหน้า Bus interface ให้คลิกเลือก clock_generator_0 คลิกขวาเลือก Configure IP เพื่อกำหนดคุณสมบัติของ Clock generator จะปรากฏหน้าต่างดังภาพประกอบ ค-39



ภาพประกอบ ค-39 หน้าต่างกำหนดคุณสมบัติ Clock Generator

ให้กำหนดเอาต์พุตของ Clockgenerator เพิ่มขึ้นมาโดยกำหนดที่ CLKOUT1 ให้มีความถี่ Required Frequency (Hz) = 200 MHz เสร็จแล้วกด OK

5.7. กำหนดให้ สัญญาณ CLKOUT1 เป็น External output โดยคลิกเลือกแทบ Port จากนั้นเลือก clock_generator_0 จะเห็นว่า CLKOUT1 นั้นยังไม่ได้กำหนดพอร์ตให้คลิกเลือกเป็น External Port : clock_generator_0_CLKOUT1_pin ดังภาพประกอบ ค-40



ภาพประกอบ ค-40 กำหนดให้ CLKOUT1 เป็น External Port

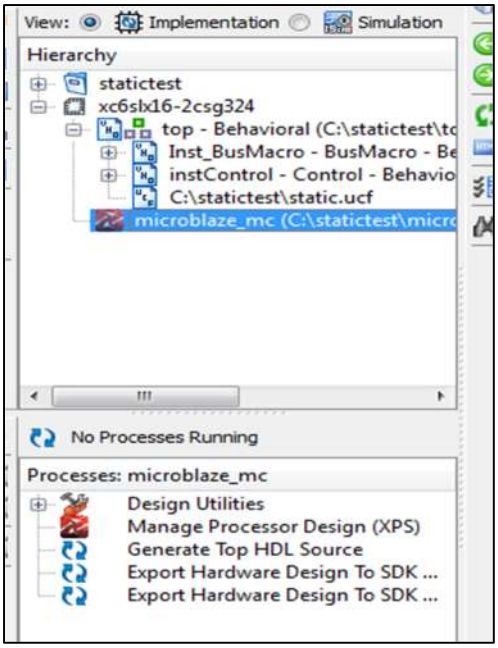
ดังนั้นหากผู้ใช้งานคลิกดูใน External ports จะเห็นว่าโมดูล Microblaze จะมีพอร์ตทั้งหมด 6 ชุด ดังภาพประกอบ ค-41

Name	Connected Port	Direction	Range	Class	Frequency(Hz)
External Ports					
CLK_N	clock_generator_0::CLKIN	I		CLK	200000000
CLK_P	clock_generator_0::CLKIN	I		CLK	200000000
RESET	proc_sys_reset_0::Ext_Reset_In clock_generator_0::RST	I		RST	
RS232_Uart_1_sin	RS232_Uart_1::[uart_0]::RX	I		NONE	
RS232_Uart_1_sout	RS232_Uart_1::[uart_0]::TX	O		NONE	
clock_generator_0_CLKOUT1_pin	clock_generator_0::CLKOUT1	O		CLK	200000000

ภาพประกอบ ค-41 อินพุตเอาต์พุตพอร์ตโมดูล Microblaze

ขณะนี้ถือได้ว่ากำหนดคุณสมบัติของโมดูล Microblaze เสร็จแล้ว ให้ปิดซอฟต์แวร์ XPS แล้วกลับไปยังโปรเจคในซอฟต์แวร์ Xilinx ISE

5.8. เพิ่มโครงสร้าง Template และการเชื่อมต่อสัญญาณ Microblaze โดยเลือกโมดูล Microblaze_mc จะปรากฏดัง ภาพประกอบ ค-42 ในแทบ Processes ให้เลือก Generate top hdl source เพื่อสร้าง Template ของ Microblaze ด้วย VHDL



ภาพประกอบ ค-42 โมดูล Microbaze ใน Xilinx ISE

ผลลัพธ์จะได้ข้อมูลดังภาพประกอบ ค- 43 ซึ่งประกอบไปด้วยโค๊ด Template และโค๊ดการนำโมดูลไปเชื่อมต่อกับสัญญาณใน Top module

```
-- VHDL Instantiation Created from source file microblaze_mc.vhd -- 11:23:19 02/18/2015
--
-- Notes:
-- 1) This instantiation template has been automatically generated using types
-- std_logic and std_logic_vector for the ports of the instantiated module
-- 2) To use this template to instantiate this entity, cut-and-paste and then edit

   COMPONENT microblaze_mc
   PORT(
      RS232_Uart_1_sin : IN std_logic;
      RESET : IN std_logic;
      CLK_P : IN std_logic;
      CLK_N : IN std_logic;
      RS232_Uart_1_sout : OUT std_logic;
      clock_generator_0_CLKOUT1_pin : OUT std_logic
      );
   END COMPONENT;

attribute box_type : string;
attribute box_type of microblaze_mc : component is "user_black_box";


   Inst_microblaze_mc: microblaze_mc PORT MAP(
      RS232_Uart_1_sout => ,
      RS232_Uart_1_sin => ,
      RESET => ,
      CLK_P => ,
      CLK_N => ,
      clock_generator_0_CLKOUT1_pin =>
   );
```

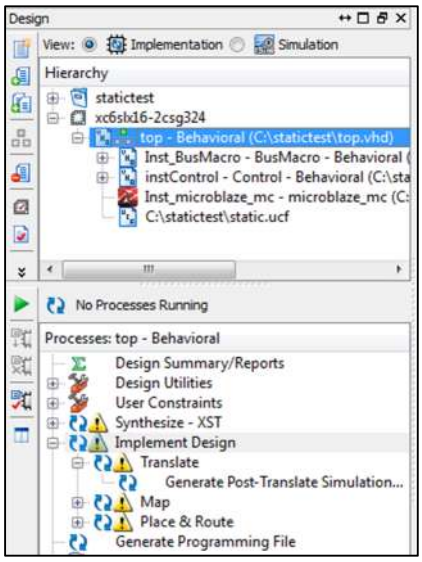
ภาพประกอบ ค- 43 Template โมดูล Microblaze

5.9. นำข้อมูลทั้งสองส่วนใส่ไว้ใน Top module และทำการเชื่อมต่อสัญญาณตามภาพประกอบ ค-44 โดยที่ อินพุตเอาต์พุตของ Microblaze สามารถปรับเปลี่ยนได้ตามเหมาะสม ยกเว้นสัญญาณ clock_generator_0_CLKOUT1_pin ต้องเชื่อมกับ CLK200 เท่านั้น

```
106 ----------------------------------------------------------------------------------
107 -- You can replace your code below this comment
108 ----------------------------------------------------------------------------------
109
110    Inst_microblaze_mc: microblaze_mc PORT MAP(
111       RS232_Uart_1_sout => UART_TX,
112       RS232_Uart_1_sin => UART_RX,
113       RESET => RESET,
114       CLK_P => CLK_P,
115       CLK_N => CLK_N,
116       clock_generator_0_CLKOUT1_pin =>  CLK200
117    );
118
119
120    LocalLEDs <= ST_data_IN(3 downto 0);
121    ST_data_OUT(3 downto 0)<= GPIO_BUTTON;
122    ST_data_OUT(35 downto 4) <= ST_data_IN(35 downto 4);
123
124 ----------------------------------------------------------------------------------
125 -- end of your code
126 ----------------------------------------------------------------------------------
```

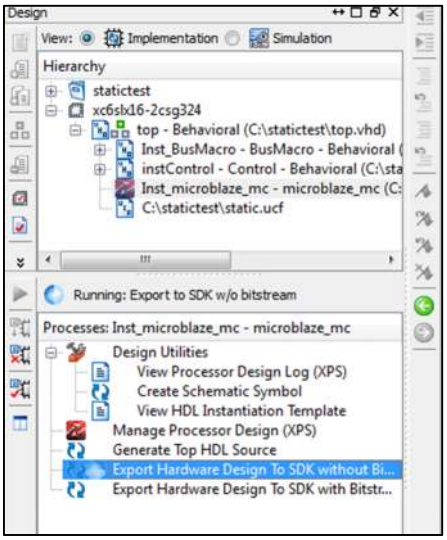
ภาพประกอบ ค-44 การเชื่อมต่อสัญญาณของโมดูล Microblaze

5.10. เมื่อแก้ไขเสร็จแล้ว จะเห็นว่าในแทบ Hierrachy จะมีโมดูล Microblaze เข้ามาอยู่ใน Top แล้ว ขั้นตอนถัดมา คือ ทำกระบวนการสร้างไฟล์วงจรโดยคลิก Implementation design เพื่อเริ่มการ Synthesis > Implementation ดังภาพประกอบ ค-45



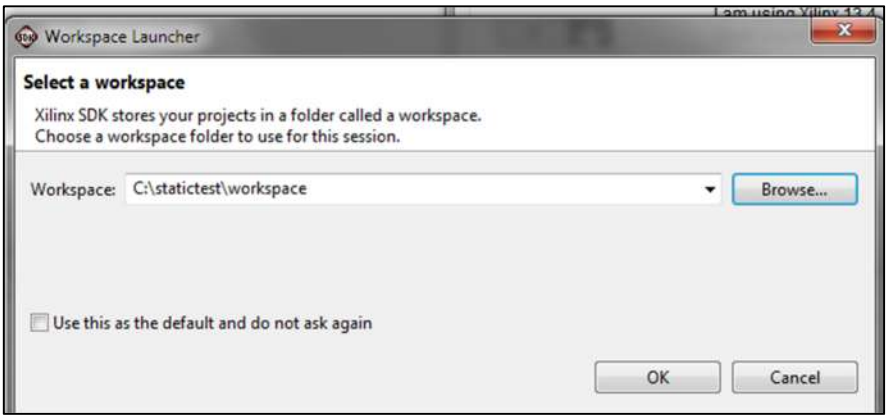
ภาพประกอบ ค-45 ทำกระบวนการสร้างไฟล์วงจร

5.11. จากนั้นคลิกที่โมดูล Microblaze เลือก Export hardware design to sdk without bitstream ดังภาพประกอบ ค-46 เพื่อเรียกใช้ซอฟต์แวร์ Xilinx Software Development Toolkit (SDK) สำหรับเขียนโค๊ดโปรแกรมที่ให้ทำงานบน Microblaze



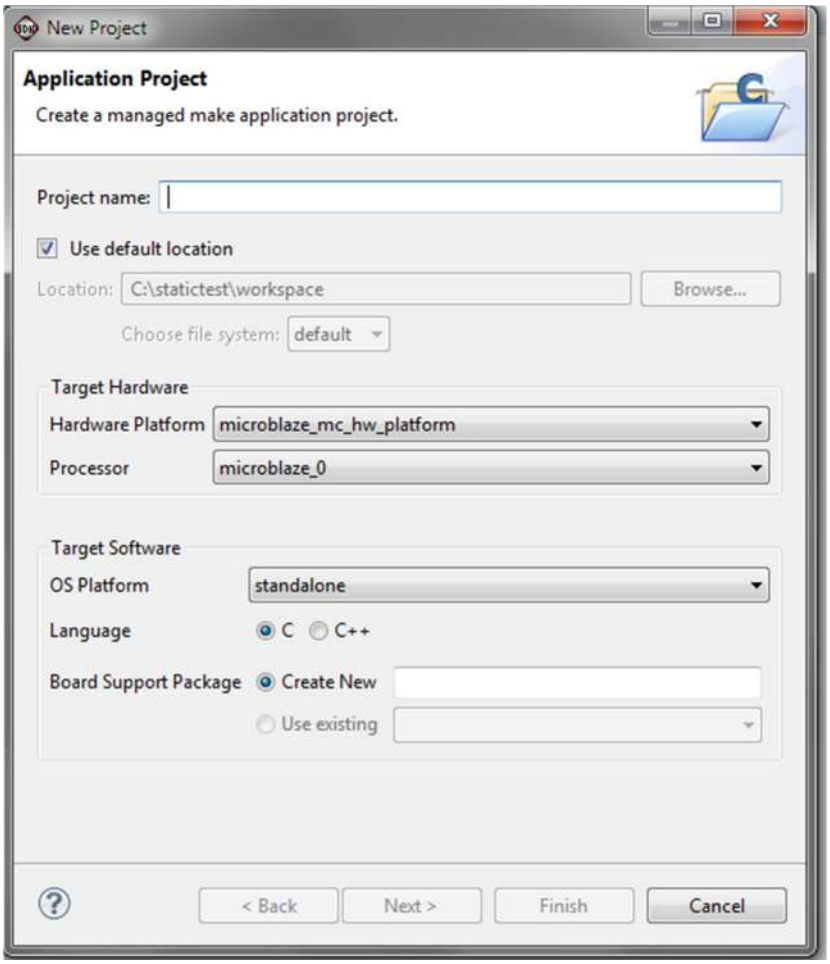
ภาพประกอบ ค-46 เรียกใช้ซอฟต์แวร์ SDK

จะปรากฏหน้าต่างให้กำหนด Workspace ดังภาพประกอบ ค-47 เพื่อเก็บโค๊ดโปรแกรม โดยให้สร้างโฟลเดอร์ใหม่ไว้ที่เดียวกับโฟลเดอร์โปรเจค จากนั้นกด OK



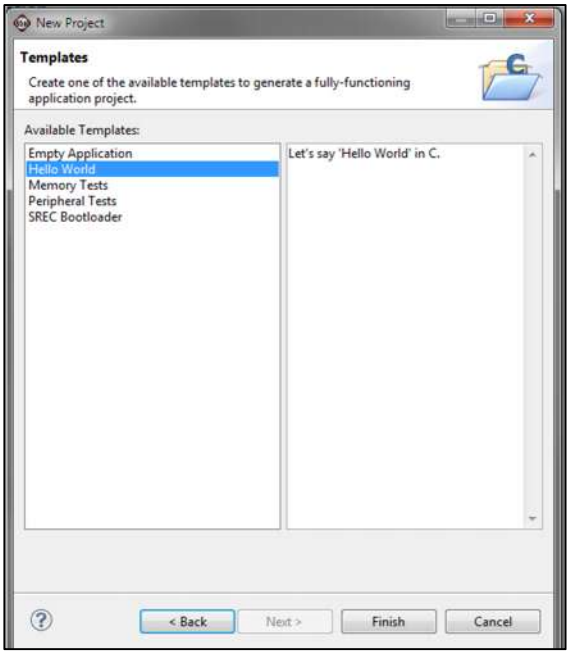
ภาพประกอบ ค-47 หน้าต่างกำหนด Workspace

5.12. ทำการสร้างโปรเจคโดยไปที่ Project > New Project จะปรากฏหน้าต่าง ดังภาพประกอบ ค-48 ให้กำหนดชื่อ Project name ตามที่ต้องการ จากนั้นกด Next



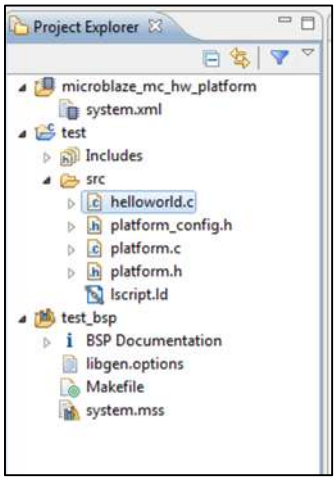
ภาพประกอบ ค-48 สร้างโปรเจคในซอฟต์แวร์ SDK

5.13. หน้าต่าง Template สำหรับเป็นโครงสร้างเริ่มต้นของการสร้างโปรแกรม ดังภาพประกอบ ค-49 ให้เลือก Empty application หรือ Hello world จากนั้นกด Finish



ภาพประกอบ ค-49 กำหนด Template เริ่มต้นของโปรแกรม

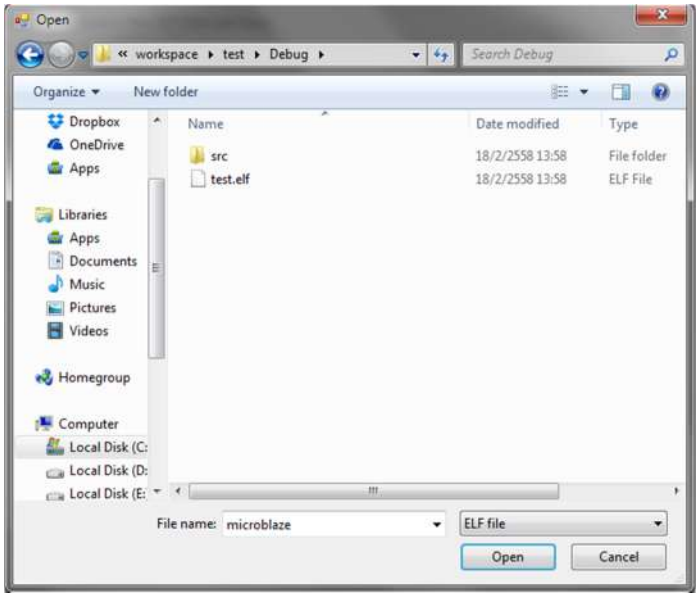
เมื่อดูในแทบ Project explorer จะเห็นว่ามีโฟลเดอร์เพิ่มมาดังภาพประกอบ ค-50 ให้เข้าไปดูใน src folder จะเจอไฟล์ .c, .h โดยในที่นี้เลือกใช้ Template hello world ดังนั้น ฟังก์ชัน main() จะอยู่ในไฟล์ helloworld.c



ภาพประกอบ ค-50 โครงสร้างโปรเจคใน Xilinx SDK

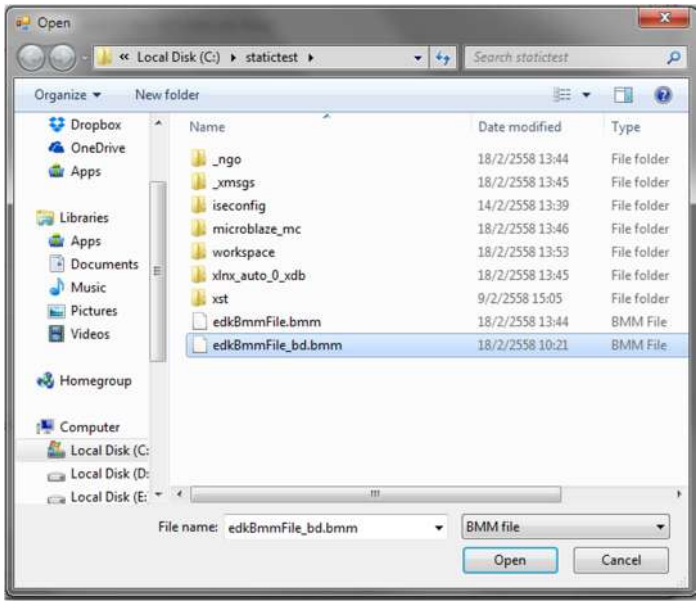
คัดลอกโค๊ดใน Microblaze_code.c มาแทนที่ใน Hellowold.c หรือในไฟล์ที่เก็บฟังชัน main() ทั้งหมด จากนั้นกด save เพื่อทำการคอมไพล์ จากนั้นกดปิดซอฟต์แวร์ SDK แล้วกลับไปยังซอฟต์แวร์ PReFF

5.14. ขั้นตอนของซอฟต์แวร์ PReFF ในกระบวนการ Define area, Partial design และ Download นั้นเหมือนเดิม แต่จะเพิ่มในกระบวนการ Static design โดยเพิ่มไฟล์ .elf ในช่อง select .elf ซึ่งไฟล์นี้จะอยู่ใน workspace/*project name*/Debug/ . ในภาพประกอบ ค-51 ได้ออกแบบโปรเจคชื่อ test ดังนั้นใน Workspace จะมีโฟลเดอร์ชื่อ test และภายในจะมีไฟล์ .elf



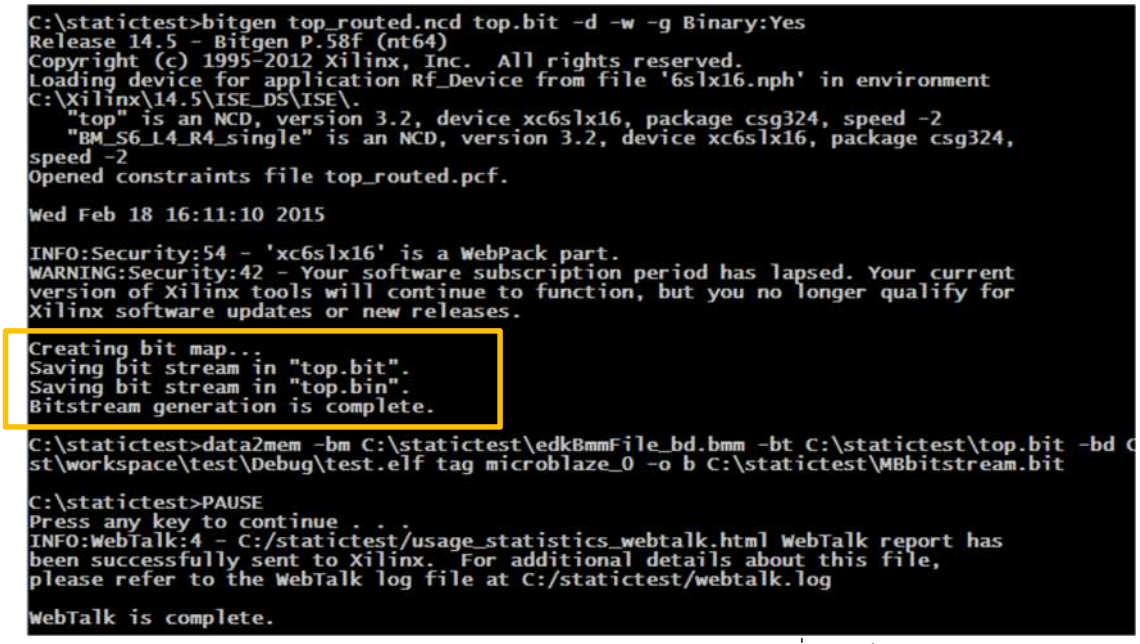
ภาพประกอบ ค-51 การเลือกไฟล์ .elf

5.15. เลือกไฟล์ .bmm โดยไฟล์จะอยู่ในโปรเจค Xilinx ให้เลือกไฟล์ edkBmmFile_bd.bmm ดังภาพประกอบ ค-52



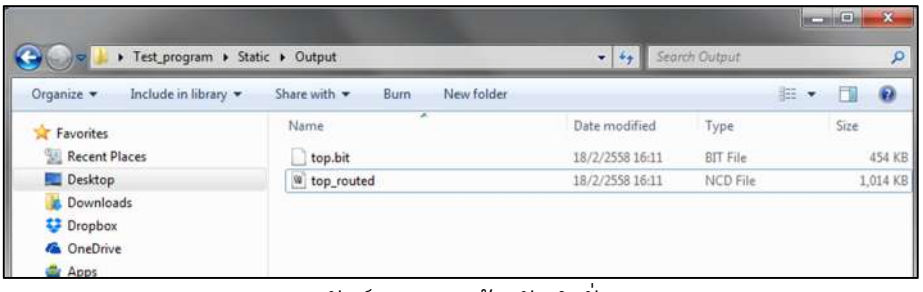
ภาพประกอบ ค-52 การเลือกไฟล์ Bmm

5.16. เมื่อเตรียมข้อมูลครบแล้วคลิก Generate bitstream เพื่อรันคำสคริป คำสั่งในการสร้างไฟล์ Bitstream จะปรากฏหน้าต่าง Command Prompt รันการทำงาน เมื่อเสร็จการทำงานจะปราฏหน้าต่างดังภาพประกอบ ค-53



ภาพประกอบ ค-53 หน้าต่าง Command Prompt สคริปคำสั่งสร้างไฟล์ Bitstream

5.17. ผลลัพธ์จากการรันคำสั่ง Generate bitstream อยู่ในโฟลเดอร์ Define path /Static/Output ประกอบไปด้วยไฟล์ .ngc และ .bit ดังภาพประกอบ ค-54



ภาพประกอบ ค-54 ผลลัพธ์จากการสร้างรันคำสั่ง Generate Bitstream

จากนั้นผู้ใช้งานสามารถโปรแกรมไฟล์วงจรผลลัพธ์นี้ลงเอฟพีจีเอเพื่อให้ทำงานได้ ซึ่งกระบวนการดาวน์โหลดวงจรลงเอฟพีจีเอได้อธิบายไว้แล้วในหัวข้อก่อนหน้านี้

ภาคผนวก ง
ผลงานตีพิมพ์เผยแพร่จากวิทยานิพนธ์

The 29th International Technical Conference on Circuit/Systems Computers and Communications (ITC-CSCC 2014)

July 1-4, 2014 Phuket Graceland Resort and Spa, Phuket, Thailand

The 29th International Technical Conference on Circuit/Systems Computers and Communications (ITC-CSCC), Phuket, Thailand, July 1-4, 2014

Development of Remote Partial Reconfigurable Circuits Implementation on FPGA

Hadee Mayurapong
Department of computer engineering,
Faculty of Engineering, Prince of Songkla University,
Hatyai Songkhla 90112 Thailand
5510120103@email.psu.ac.th

W. Suntiamorntut
Department of computer engineering,
Faculty of Engineering, Prince of Songkla University,
Hatyai Songkhla 90112 Thailand
wannarat@coe.psu.ac.th

***Abstract*—This paper presents the implementation of partial reconfigurable circuits remotely based on Field Programmable Gate Arrays (FPGAs). This feature enables the flexibility and resource utilization. Partial Reconfiguration (PR) has been introduced in a recent day in order to modify the implemented logic in Field Programmable Gate Arrays (FPGAs) while the device is running. By using this, the circuits are able to be changed or modified according to the current functions. However, the developer is required the specific and deep knowledge about the FPGA structure and design tools. Thus the partial reconfiguration feature is not widely used. This paper is therefore introduced and demonstrated the development process of partial reconfiguration. This will make the developer to understand and is able to follow up the design methodology easily. In addition, we will analyze the benefit of using partial reconfiguration feature. The remote partial reconfiguration has also been introduced to apply in wireless sensor networks. This allows us to reconfigure the circuits wirelessly depending on its function at that time.**

Keywords—partial reconfigurable, FPGA, Spartan-6

I. Introduction

Partial Reconfiguration (PR) has been introduced in a recent day in order to modify the implemented logic in Field Programmable Gate Arrays (FPGAs) while the device is still operating. This feature enables the flexibility and resource utilization. By using this, the circuits are able to be changed or modified according to the current functions. However, the developer is required the specific knowledge about the FPGA structure and design tools. Thus the partial reconfiguration feature is not widely used. This paper is therefore explained and demonstrated the design flow of partial reconfigurable implementation. This will make the developer to understand and is able to follow up the design methodology easily. In addition, we will analyze the advantages of using partial reconfiguration feature.

There are some research works applying the reconfigurable device in wireless sensor networks reported in [1-3] in order to increase the flexibility and computational resources. However, it is not a partial reconfigurable technique. Thus the wireless sensor node which is able to partial reconfigure remotely has been introduced. This allows us to reconfigure the circuits suited with each situation or environment of wireless sensor network system leading to achieve the energy efficiency.

Although the partial reconfiguration has been introduced and launched a commercial tool since 3-4 years ago, it is not widely used because of the limitation of design tools. Xilinx announced PlanAhead[4] to support the partial reconfigurable FPGA. However, PlanAhead only supports some FPGA families such as Virtex4 – Virtex7, Artix-7 and Zynq™-700[5]. Those FPGA families are expensive and power hungry. Thus we have reviewed and demonstrated the design flow of the partial reconfiguration based on the low power and cheap FPGA, Spartan-6 LX16. Moreover, the first prototype of remote partial reconfigurable node is introduced to apply in wireless sensor networks.

The rest of this paper is organized as follows: the related works are explained in Section II. The partial reconfiguration design flow and implementation are described in Section III. In Section IV, the experiments are reported. Finally, we conclude in Section V.

II. Related Works

We have explored the partial reconfigurable FPGA based on Spartan Families. The fast startup design flow based on Spartan-6 has been proposed in [6]. Their design flow based on Xilinx-PlanAhead was the process to generate *full bit file* and *partial bit file*. They also improved the speed of design process by starting the process with partial bit file instead of the full bit file. Meanwhile, the research work in [7] presented cPCAP (compressed Parallel Configuration Access Port) module which was the interface between BRAM memory and ICAP (Internal Configuration Access Port) and was used to control the partial reconfigurable area. In [8], the authors proposed the partial reconfigurable tools and demonstrated successfully on Xilinx Spartan-6. Their tool is independent and does not need to connect with Xilinx EDA Tools. It was able to re-implement the logics in the partial reconfigurable area whilst the static reconfigurable area was remained performing. The partial reconfiguration was applied in wireless sensor networks for power management algorithm by choosing the hardware component corresponding to their functions [9]. This research work used the microcontroller and Xilinx Spartan-6 to demonstrate their works. In [10], the partial reconfigurable tool for Xilinx Spartan-6 was proposed.

The concept of this tool was to change the static design to be partial design. For example, two static designs were multiplexed to perform.

The research works explained above are different based on the design flow tools. For instance, PlanAhead is a Xilinx tool. Unfortunately, it does not support Spartan to reconfigure partially. Dream Tool [11] was proposed to deal with the routing method after the static and partial parts were placed. Unfortunately, this tool only supported Xilinx Virtex-5. JBits Tool [12] was developed using JAVA to implement partial reconfiguration. It was easy to use via GUI showing a floorplan. But again, only Xilinx Virtex-5 was supported. OpenPR tool [13] collaborated with Xilinx PlanAhead and supported Virtex-4 and Virtex-5. Recobus-Builder [14] was similar to JBit tool and supported Virtex-2 and Spartan-3. The last tool we have reviewed was GoAhead [15] which was developed from Recobus-Builder by including the communication part between static and partial parts. GoAhead supported Xilinx Virtex-4, Virtex-5, Virtex-6, Virtex-7 and Spartan-6. We are planning to apply low power Xilinx Spartan-6 in our wireless sensor networks works, thus GoAhead tool is chosen to use it in this paper. The detail of Spartan-6 and GoAhead tool will be described in the next section.

III. Spartan-6 Partial Reconfiguration

In this section, we will explain the architecture of Xilinx Spartan-6 and how to reconfigure it partially using GoAhead tool. For partial reconfiguration, the FPGA structure is divided into static part which will perform all the time (not be interrupted by another part) and, partial part or dynamic part which will be reconfigured only within the partial area. As you can notice, there are two parts running on the FPGA. Thus there will be a connector. Bus macro is one type of the famous connector working likes I/O port. Moreover, we need to activate the clock if both static and partial parts share the master clock.

A. Spartan-6 Architecture

Spartan-6 [16] is a FPGA chip of Xilinx Spartan family working as same as the others. The circuits is converted to Bitstream file and then downloaded onto FPGA via JTAG and serial cable. Xilinx provides the design tool beginning with HDL code or schematic to synthesis, place and route before the Bitstream file is generated. FPGA contains a sea of CLBs (Configuration Logic Blocks) whilst a CLB has two columns of SLICE. LUT (Look up Table) and storage element (Flip-Flop) are located in each SLICE. There are three types of SLICE, SLICEX (generic), SLICEL (similar to SLICEX and contains wide multiplexer and carry logic) and SLICEM (as same as SLICEL including 64-bit distributed RAM and 32-bit shift register in each LUT). Normally, one CLB must have SLICEX and either SLICEL or SLICEM. One SLICE contains four LUTs and eight Flip-Flops. Thus one CLB will have eight LUTs and sixteen Flip-Flops together with wide multiplexer plus carry logic or 256-bit distributed RAM plus 128-bit shift register depends on having SLICEL or SLICEM.

Apart from CLBs, FPGA has many special modules such as DSP48, 32-bit Block RAM and DCM (Digital Clock Management). In this paper, Spartan-6 LX16 is used. Thus we have 2278 SLICEs, 32 modules of DSP48 and ICAP (Internal Configuration Port) which is the most important module to reconfigure partially. ICAP only exists in Spartan-6 and Virtex family.

B. GoAhead Tool

GoAhead tool [15,17] has been developed by the researchers at University of Oslo. Its advantage is able to open the floorplan of FPGA on GUI. Thus it is easy to define the area of static and partial part in FPGA as shown in Fig.1. GoAhead can open XDL (Xilinx Design Langauge) which is a file format of Xilinx floorplan.

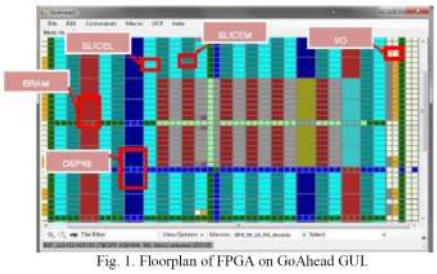


Fig. 1. Floorplan of FPGA on GoAhead GUI.

The process will start from defining the partial area and not allow the static part to be placed in this area. Then we will generate the bitstream file following the flow as shown in Fig.2. After the partial area is defined on GoAhead, the Config Prohibit.ucf is generated. The SLICEs in that area is completely reserved for partial reconfiguration. The next step is to design and generate Bus macro VHDL which will be explained later on. GoAhead will produce the last file, StaticBlocker.xdl. This file contains the area information for static part when the routing process is activated. The regular static design can be designed and implemented by HDL. Then every file is combined using Xilinx ISE and produced Top_map.ncd file. The Top_map.ncd file will merge with StaticBlocker.xdl file and route to connect between static and partial part. Top.bit and Top.ncd files are generated at the end. We will use Top.bit to download onto FPGA.

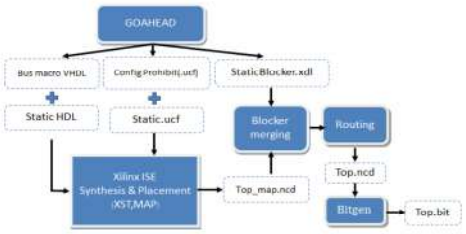


Fig. 2. Static part design flow

≠

System architecture after we combine the static and partial part. The partial communicates to the static part via Bus macro. The UART link is used to interface between FPGA and outside world to download bitstream. However, ICAP is the interface port connecting to partial part. Thus there must be a converter called UART_ICAP linked between UART and ICAP.

C. *Peripherals in Partial Reconfigurable System*

As we have shown the system architecture in the previous section, the necessary peripherals are described here.

a) UART_ICAP is a converter receiving a data from UART and flow data sending it to the ICAP. The state machine for control ICAP to write data which sending from UART also implement in UART_ICAP.

b) ICAP is an another type of SelectMAP using to interface between a regular area in FPGA and partial part. ICAP has two functions, read and write 16-bit data. We can control these functions of ICAP using state machine or HWICAP which is a Xilinx IP Core. However, we require additional microcontroller in case of using HWICAP.

c) Bus macro is an interface port between static and partial part. The SLICEs of both static and partial part have to be declared as the I/O of static and partial part. The output pins of LUT in the SLICE of each static and partial part has to be the same. If we want to use one SLICE, the bus macro will have 4 I/O due to one SLICE contained 4 LUTs. In this paper, we create 16-bit Bus macro. So four SLICEs are used as shown in Fig.3. The pair of four SLICEs are aligned in y-axis.

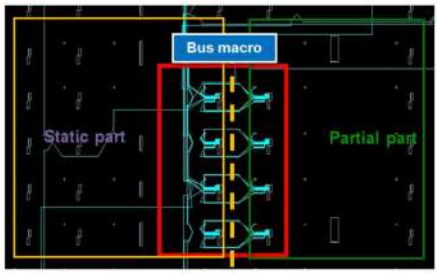


Fig. 3. 16-bit Bus macro communication

Up to this process, the full bit file is ready to download onto FPGA. We have tested the process using Spartan6LX16 FPGA or SP601 development board. The bit file has been downloaded via JTAG. We found that the partial part is leave as a blank area as shown in Fig.4. Whilst the necessary peripherals such as UART_ICAP, ICAP and Bus macro are located in the FPGA. Now the system is ready to support partial reconfiguration. The partial bit file will be downloaded via UART_ICAP.

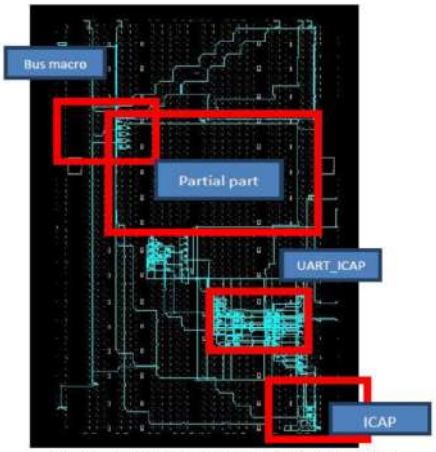


Fig. 4. The system of partial reconfiguration in Floorplan

D. *Partial Part Design*

The partial part design is similar with the static part design. The output files are partial.bit and partial.ncd. Normally, we can design the partial part using VHDL or Verilog. In this paper, we use VHDL to design the circuits in the partial part.

IV. Experiment and Results

We have designed 8-bit full adder to partial part for testing the design flow, UART_ICAP, ICAP and Bus macro and to analysis the design overhead. The Spartan-6LX16 or SP601 development board is chosen in this experiment.

A. *Resource Usage*

In order to analysis the resource overhead, the 8-bit full adder has been designed and configured in both full reconfiguration (original) and partial reconfiguration. We found that partial reconfiguration is able to generate the hardware or resource overhead about 4-time in case of 8-bit full adder implementation as shown in Table1. Whilst the number of the internal signals are not different. The logic overhead came from the necessary peripheral such as Bus macro, UART_ICAP and ICAP.

B. *Configuration File*

The size of bit file in a regular reconfiguration system is 454 Kbytes whilst the partial bit file is very small or depends on the size of circuits in the partial part as shown in TableII. Thus the partial circuits or partial bit files are capable to store on board memory or memory in the microcontroller. The microcontroller will download the partial bit file onto FPGA when it requires that circuits.

TABLE I. RESOURCE USED IN SPARTAN-6LX16

On-CHIP resources	Type of configuration	
	Full reconfiguration	Partial reconfiguration
Logic	8	32
Signal	27	36

TABLE II. SIZE OF BIT STREAM FILES

Configuration design	File size (.bit, KB)
Full reconfiguration bit file (bit file in a normal FPGA)	454
Partial bit file	4

C. Power Consumption

Apart from the resources, we would like to show the power consumption when the partial reconfiguration is applied shown in TableIII. There is not an overhead of peripheral for partial part in term of power consumption.

TABLE III. POWER CONSUMPTION

Sources	Power (W)	
	Full reconfiguration	Partial reconfiguration
Vccint	0.0072	0.0072
Vccaux	0.0075	0.0075
Total	0.0147	0.0147

V. Conclusion & Future Work

This paper demonstrates how to reconfigure the circuits partially on Spartan-6 using GoAhead tool using 8-bit full adder successfully. We measure the time since the device has got the bit file and start to reconfigure via UART (115200 bps) and ICAP. It takes 0.28 s to complete this chain. There is a little hardware overhead (peripheral module). Whilst the power consumption of those peripheral modules will not affect to the power dissipation in FPGA. Thus, the partial reconfiguration is capable to apply in wireless sensor node for energy efficiency. The necessary circuits are in FPGA at the time we want to use. There will not have the wasting modules consumed the power. In future, we have plan to use internal microcontroller (MicroBlaze) on FPGA to control the reconfiguration partially.

ACKNOWLEDGMENT

This work was supported by Centre of Excellence in Wireless Sensor Networks and Faculty of Engineering, Prince of Songkla University. The authors also gratefully acknowledge the software tools donated by Xilinx.

REFERENCES

[1] Yibin and Jia, Zhiping and Liu, Fucai and Xie, Shuai Li, "Hardware reconfigurable wireless sensor network node with power and area efficiency," *IET Wireless Sensor Systems*, vol. 2, pp. 247-252, 2012.

[2] Jiaqi and Zhao, Lei and Hao, Qi and Hu, Fei and Hong, Xiaoyan Gong, "A reconfigurable hardware platform for cognitive sensor networks towards behavioral biometrics," in *Proceedings of IEEE Sensors*, 2012, pp. 1-4.

[3] Yana Esteves and Portilla, Jorge and de la Torre, Eduardo and Riesgo, Teresa Krasteva, "Embedded runtime reconfigurable nodes for wireless sensor networks applications," *Sensors Journal, IEEE*, vol. 11, pp. 1800-1810, 2011.

[4] XILINX. (2009, December 1) PlanAhead User. UG632 (v 11.4).

[5] XILINX. (2013, April 26) Partial Reconfiguration User Guide. UG702 (v14.5).

[6] Joachim and Noguera, Juanjo and Hübner, M and Braun, Lars and Sander, Oliver and Gil, R Mateos and Stewart, Rodney and Becker Meyer, "Fast start-up for spartan-6 fpgas using dynamic partial reconfiguration," in *Design, Automation & Test in Europe Conference & Exhibition (DATE) IEEE*, 2011.

[7] Salih and Yurdakul, Arda and Tukel, Mehmet Bayar, "A self-reconfigurable platform for general purpose image processing systems on low-cost spartan-6 fpgas," in *Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), 2011 6th International Workshop on IEEE*, 2011, pp. 1-9.

[8] Christian Beckhoff, and Jim Torrison. Koch Dirk, "Advanced partial run-time reconfiguration on Spartan-6 FPGAs," in *Field-Programmable Technology (FPT), 2010 International Conference on IEEE*, 2010, pp. 361-364.

[9] Miguel and Camarero, Julio and Valverde, Juan and Portilla, Jorge and de la Torre, Eduardo and Riesgo, Teresa Lombardo, "Power management techniques in an FPGA-based WSN node for high performance applications," in *Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), 2012 7th International Workshop on IEEE*, 2012, pp. 1-8.

[10] Christian and Koch Dirk and Torresen, Jim Beckhoff, "Migrating Static Systems to Partially Reconfigurable Systems on Spartan-6 FPGAs," in *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on IEEE*, 2011, pp. 212-219.

[11] Andr'es and de la Torre, Eduardo and Riesgo, Teresa Otero, "Dreams: A tool for the design of dynamically reconfigurable embedded and modular systems," in *Reconfigurable Computing and FPGAs (ReConFig), 2012 International Conference on IEEE*, 2012, pp. 1-8.

[12] Daniel and Moraes, Fernando and Palma, Jos 'e and Moller, Leandro and Calazans, Ney Mesquita, "Remote and Partial Reconfiguration of FPGAs: tools and trends," in *Parallel and Distributed Processing Symposium, 2003. Proceedings. International IEEE*, 2003, pp. 8.

[13] Ali Asgar and Athanas, Peter and Frangieh, Tannous and Wood, Aaron Sohanghpurwala, "Openpr: An open-source partial-reconfiguration toolkit for xilinx fpgas," in *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on IEEE*, 2011, pp. 228-235.

[14] Koch, D., Beckhoff, C., Haubelt, C., & Teich, J., "ReCoBus-Builder: a Novel Tool for Component-based System Design on FPGAs".

[15] Christian and Koch Dirk and Torresen, Jim Beckhoff, "Go Ahead: A Partial Reconfiguration Framework," in *Field-Programmable Custom Computing Machines (FCCM), 2012 IEEE 20th Annual International Symposium on IEEE*, 2012, pp. 37-44.

[16] Xilinx. (2010, February 23) Spartan-6 FPGA Configurable Logic Block User Guide. UG384 (v1.1).

[17] Jim and Beckhoff, Christian and Ziener, Daniel and Dennl, Christopher and Breuer, Volker and Teich, Jurgen and Feilen, Michael and Stechele, Walter Koch Dirk and Torresen, "Partial reconfiguration on FPGAs in practice—Tools and applications," in *ARCS Workshops (ARCS), 2012 IEEE*, 2012, pp. 1-12.

ประวัติผู้เขียน

ชื่อ สกุล นายฮาดีย์ หมัดอาด้ำ

รหัสประจำตัวนักศึกษา 5510120103

วุฒิการศึกษา

วุฒิ	ชื่อสถาบัน	ปีที่สำเร็จการศึกษา
วิศวกรรมศาสตรบัณฑิต (วิศวกรรมคอมพิวเตอร์)	มหาวิทยาลัยสงขลานครินทร์	2553

ทุนการศึกษา (ที่ได้รับในระหว่างการศึกษา)

1. ทุนศิษย์ก้นกุฏิ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยสงขลานครินทร์ ประจำปีการศึกษา 2555

การตีพิมพ์เผยแพร่ผลงาน

H. Mayurapong and W. Suntiamorntut, “Development of Remote Partial Reconfigurable Circuits Implementation on FPGA,” *In Proceedings of 29th International Technical Conference on Circuit/Systems Computers and Communications (ITC-CSCC 2014),* Phuket, Thailand, 1-4 July 2014, pp. 440-443.