



พื้นฐานของ FPGA และ VHDL

3.1 Field-Programmable Gate Array (FPGA) [7,8,9]

เป็นอุปกรณ์ PLD (Programmable Logic Device) ที่มีความซับซ้อนไปอีกระดับหนึ่ง ซึ่งในความเป็นจริงแล้ว PLD และ FPGA แตกต่างกันอย่างน้อยมาก สำหรับ FPGA แล้วนับว่าเป็นอุปกรณ์ตัวใหม่ที่มีการเจริญเติบโตอย่างรวดเร็วและมีบทบาทที่สำคัญในการเข้ามาแทนที่ระบบอิเล็คทรอนิกส์ที่ใช้ TTL โครงสร้างภายในของ FPGA ประกอบไปด้วยอาร์เรย์ของลอจิกเกตต่างๆ มากมาย ซึ่งในปัจจุบันความจุเกตภายในตัวชิป FPGA ได้เพิ่มขึ้น จากระดับไม่กี่พันเกตจนถึงระดับล้านเกตซึ่งสามารถรองรับวงจรดิจิทัลที่มีความสลับซับซ้อนได้เป็นอย่างดี นอกจากนี้ในด้าน การออกแบบพัฒนาและทดสอบก็ทำได้ง่ายซึ่งในปัจจุบัน การออกแบบวงจรโดยใช้ FPGA กำลังเป็นที่นิยมและมีแนวโน้มที่จะนำมาใช้งานมากขึ้นเรื่อย

3.2 ทำความรู้จักกับ FPGA

ในปัจจุบันมี FPGA อยู่ 4 ชนิดที่วางขายอยู่ในท้องตลาดได้แก่ Symmetrical Array, Row-Based, Hierarchical PLD และ Sea-of-Gates ซึ่งแต่ละชนิดก็มีลักษณะการเชื่อมต่อภายในและการโปรแกรมที่แตกต่างกันไป นอกจากนี้ในการแบ่งประเภทของ FPGA อาจแบ่งได้ตามเทคโนโลยีที่ใช้ในการโปรแกรม ซึ่งมีอยู่ 2 แบบคือ การโปรแกรมโดยการทำให้เกิดการเปลี่ยนแปลงทางกายภาพของตัวชิป และการโปรแกรมโดยการใช้หน่วยความจำ

1. การโปรแกรมโดยทำให้เกิดการเปลี่ยนแปลงทางกายภาพ

- 1.1 Fuse เป็นวิธีการโปรแกรมที่สามารถทำได้เพียงครั้งเดียว ซึ่งหลังจากที่โปรแกรมแล้วจุดเชื่อมต่อจะขาดจากกัน
- 1.2 Anti Fuse เป็นวิธีการโปรแกรมที่คล้ายกับแบบ Fuse แต่ต่างที่หลังจากทำการโปรแกรมแล้วจุดเชื่อมต่อจะเชื่อมถึงกัน

2. การโปรแกรมโดยใช้หน่วยความจำ

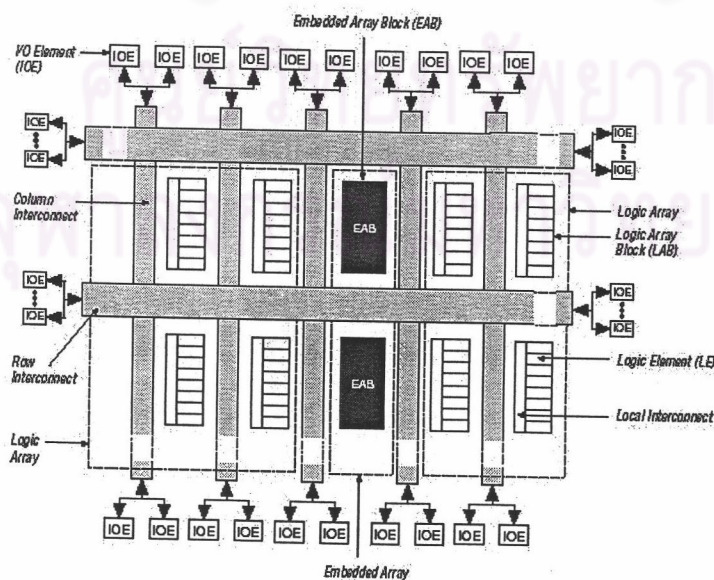
2.1 EEPROM Based FPGA

FPGA ที่ใช้การโปรแกรมแบบนี้มักเรียกว่า CPLD (Complex Programmable Logic Device) ซึ่งเทคโนโลยีที่ใช้จะเหมือนกับ EEPROM ทำให้มีความจุของเกตต่ำ โดยทั่วไปจะน้อยกว่า 20,000 เกต แต่ข้อดีของ EEPROM Based FPGA คือสามารถเก็บข้อมูลที่โปรแกรมลงไปได้โดยไม่จำเป็นต้องมีไฟเลี้ยง และในการโปรแกรมจะใช้ทรานซิสเตอร์ 1 ตัวต่อ 1 บิต การโปรแกรมสามารถทำได้ประมาณ 10,000 ครั้ง

2.2 SRAM Based FPGA

FPGA แบบนี้จะใช้เทคโนโลยีในการโปรแกรมเหมือนกับ SRAM (Static RAM) ทำให้สามารถโปรแกรมซ้ำได้โดยไม่จำกัดจำนวนครั้ง นอกจากนี้ยังมีความจุของเกตในระดับปานกลางถึงสูงมาก (ประมาณ 10,000 - 1,000,000 เกต) ซึ่งข้อดีของ SRAM Based FPGA คือใช้เวลาในการโปรแกรมน้อย (ระดับ nsec) การโปรแกรมทำได้ง่ายเทียบได้กับการเขียน SRAM ทั่วไป และเหมาะสำหรับการออกแบบวงจรที่มีความสลับซับซ้อน ข้อเสียคือไม่สามารถเก็บโปรแกรมในภาวะที่ไม่มีไฟเลี้ยงได้ ดังนั้น FPGA ชนิดนี้จึงมักใช้ควบคู่กับ ROM เพื่อเก็บโปรแกรมและทำการโหลดโปรแกรมลงในตัวชิปในขณะที่ยังไม่ทำงาน

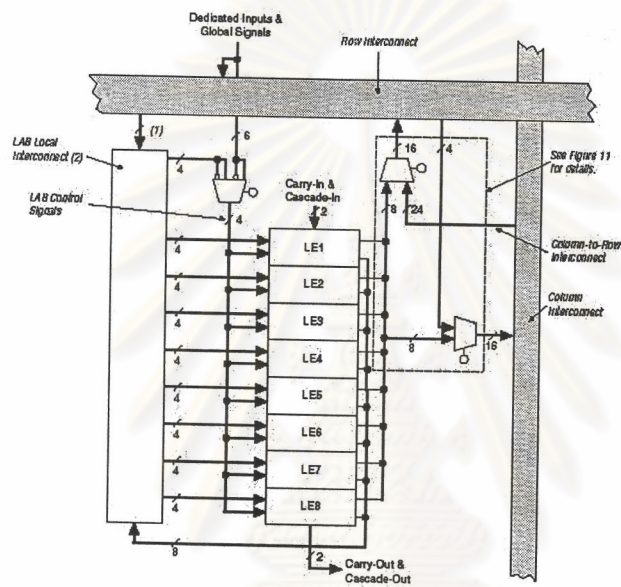
3.3 โครงสร้างภายในของ FPGA [10]



รูปที่ 3.1 โครงสร้างภายในของ ALTERA FLEX 10K

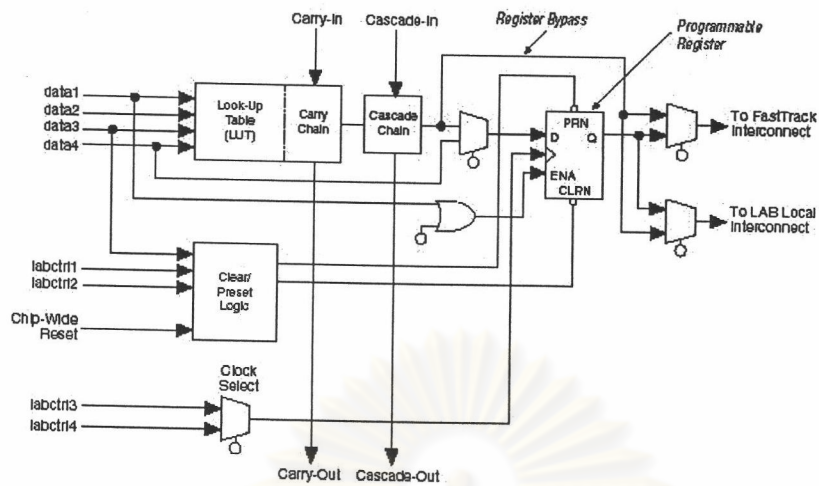
เนื่องจากวิทยานิพนธ์นี้เลือกใช้ FPGA แบบ SRAM based FPGA ของบริษัท ALTERA ในตระกูล FLEX 10K จึงอธิบายโครงสร้างภายในชิปดังต่อไปนี้

รูปที่ 3.1 แสดงโครงสร้างภายในของชิป FLEX 10K ประกอบไปด้วย Logic Array Blocks (LABs) ซึ่งแต่ละ LAB มี 8 Logic Elements ที่อยู่ในลักษณะ Lookup Tables (LUTs) และชิปก็ยังมี Embedded Array Blocks (EABs) ที่เป็นบล็อกของ SRAM ซึ่งสามารถจัดให้เป็นหน่วยความจำตามต้องการ LABs และ EABs สามารถเชื่อมต่อกันได้โดยใช้การเชื่อมต่อสายแบบแถว (Row) และแบบหลัก (Column) และสายเหล่านี้ก็ยังสามารถเชื่อมไปยังขาอินพุต/เอาต์พุตของชิปได้อีกด้วย



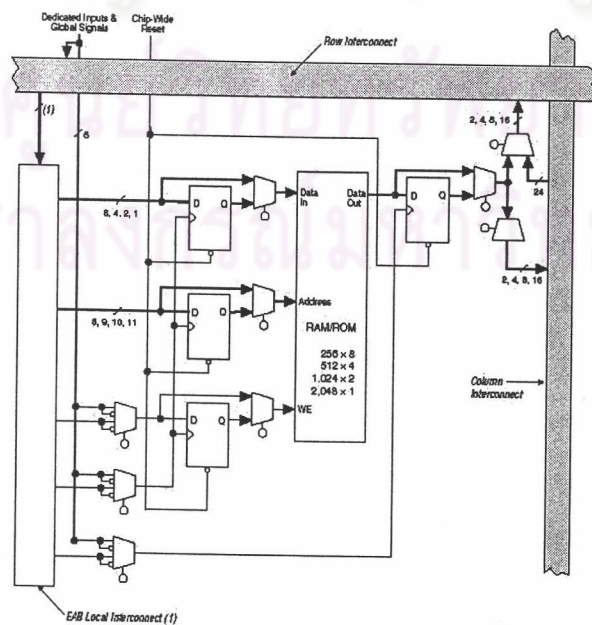
รูปที่ 3.2 โครงสร้างภายใน FLEX 10K logic array block

รูปที่ 3.2 แสดงถึงภายในของ LAB ที่มีอินพุตถูกเชื่อมต่อมาจากสายเชื่อมต่อแบบแถว (Row) ผ่านมาทางสายเชื่อมต่อภายใน ซึ่งสายเชื่อมต่อภายในถูกใช้สำหรับเชื่อมต่ออินพุตเข้ามาที่ LE (Logic Element) และเอาต์พุตของ LE (Logic Element) ก็เชื่อมต่อผ่านสายเชื่อมต่อภายในไปยังสายของแถวหรือสายของหลักเช่นกัน โครงสร้างของ LE (Logic Element) แสดงในรูปที่ 3.3 แต่ละ LE มี 4 อินพุตผ่านเข้าส่วน LUT ออกมายังฟลิปฟล็อป ซึ่งสามารถกำหนดให้ข้ามส่วนนี้ไปได้ สำหรับการกำหนดให้ทำงานเป็นการคำนวณผลบวก 4 อินพุต LUT นี้สามารถกำหนดให้เป็น 2 ตัวของ Three-Input Functions ในฟังก์ชันการบวกพร้อมส่งตัวทดที่อยู่ใน Full-adder



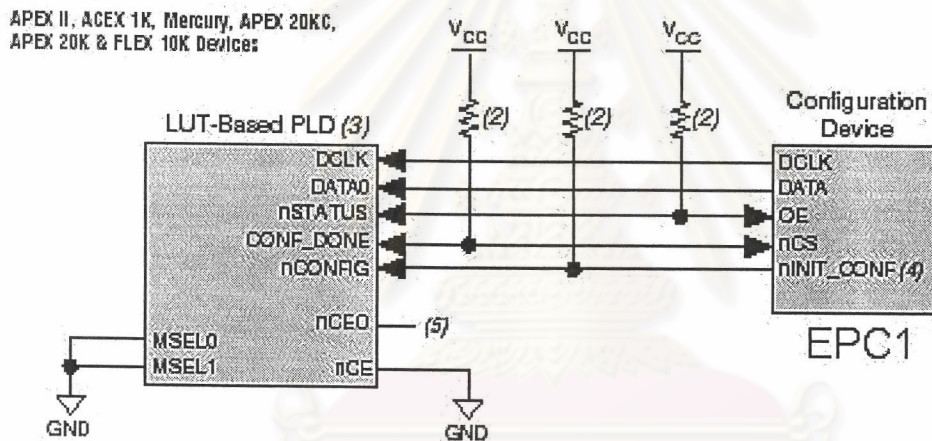
รูปที่ 3.3 โครงสร้างภายในของ FLEX 10K logic element

โครงสร้างภายในของ EAB แสดงไว้ในรูปที่ 3.4 เป็น SRAM ที่มีขนาด 2048 บิตต่อหน่วย สามารถนำมาประกอบเป็นกลุ่มของหน่วยความจำ เช่น 256×8 , 512×4 , 1024×2 และ 2048×1 บิต โดยตำแหน่งและข้อมูลของกลุ่มหน่วยความจำที่กำหนดขึ้นได้มาจากการเชื่อมต่อกันของสายเชื่อมต่อภายใน สายอินพุตที่ใช้ Enable หน่วยความจำสามารถกำหนดให้เป็นพลิกฟลอปได้ รูปที่ 3.4 แสดงถึงความหลากหลายในการกำหนดรูปแบบการอ้างตำแหน่งและการถึงข้อมูลของหน่วยความจำแบบ SRAM เอาดี้พุตของหน่วยความจำสามารถกำหนดให้เก็บไว้ในพลิกฟลอปได้ และเมื่อต้องการกลุ่มของหน่วยความจำขนาดใหญ่สามารถได้จากการรวมกันของ EABs หลายตัว



สำหรับการกำหนดรูปแบบของ EAB สามารถกำหนดได้ผ่าน LPM library ตัวอย่างเช่น lpm_ram_dq ใช้สำหรับกำหนดรูปแบบ Sram และ lpm_rom ใช้สำหรับ ROM ซึ่งถูกกำหนดผ่านทางภาษา VHDL และข้อมูลเริ่มถูกจะถูเก็บใส่ใน SRAM ก็สามารถกำหนดได้ผ่านไฟล์ที่เรียกว่า memory initialization file จะใช้งานรวมกับโมดูล lpm_ram_dq และ lpm_rom รายละเอียดเพิ่มเติมดูได้จากคู่มือโปรแกรม MAX+plus II

ชิป FLEX 10K มีขนาดให้เลือกตั้งแต่รุ่น 10K10 ถึง 10K250 ที่มีขนาดจำนวนเกตอยู่ระหว่าง 10,000 ถึง 250,000 เกต และชิปก็มีหลายความเร็วให้เลือกขึ้นอยู่กับอักษรที่อยู่ตามหลังเบอร์ชิป และความแตกต่างของการกำหนดความเร็วระหว่าง PALS และ CPLDs ที่กำหนดลำดับความเร็วด้วยค่าความหน่วงเวลาที่แท้จริงในหน่วย นาโนวินาที แต่ FPGA กำหนดไว้โดยการเปรียบกับชิปในตระกูลเดียวกัน เช่น 10K10-1 เร็วกว่า 10K10-2 ส่วนการหน่วงเวลาที่แท้จริงแตกต่างกันเป็นเท่าไร ต้องดูจากคู่มือที่ใช้เขียนโปรแกรม

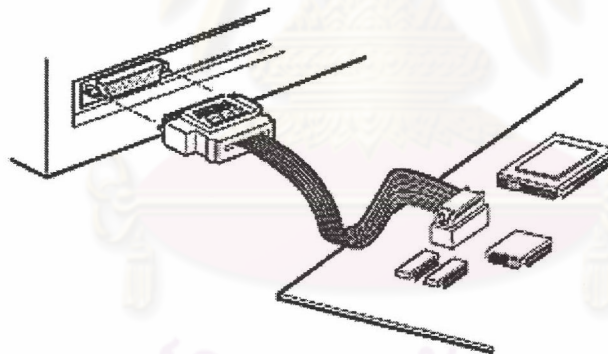


รูปที่ 3.5 แสดงการต่อกันระหว่างชิป FLEX 10K และ EPC1

เนื่องจากชิปตระกูล FLEX 10K เป็น FPGA ชิปแบบ SRAM base ซึ่งไม่สามารถรักษาโปรแกรมไว้ได้ขณะที่ไม่มีไฟเลี้ยง ดังนั้นจึงต้องต่อกับ ROM เพื่อจะต้องทำการโหลดโปรแกรมใหม่ทุกครั้งเมื่อเริ่มจ่ายไฟ สำหรับวิทยานิพนธ์นี้ใช้ชิป FPGA ของ ALTERA EPF10K70RC240-4 ในตระกูล FLEX 10K มีขนาด 70,000 เกต ซึ่งจะใช้ร่วมกับ Serial ROM ของ ALTERA เบอร์ EPC1 ดังแสดงในรูปที่ 3.5

3.4 ปัจจัยที่ทำให้การออกแบบ FPGA ทำได้ง่ายและสะดวกรวดเร็ว

1. ผู้ออกแบบไม่จำเป็นต้องทราบถึงโครงสร้างภายในของตัวชิป เพียงแต่มีความรู้เกี่ยวกับขั้นตอนการออกแบบลอจิกก็เพียงพอแล้ว ต่างกับการใช้ไมโครโพรเซสเซอร์ซึ่งจำเป็นต้องศึกษาโครงสร้างภายในรวมถึงภาษา Assembly ของไมโครโพรเซสเซอร์ตัวนั้นด้วย
2. มีการออกแบบโดยใช้ภาษาในการอธิบายการทำงานของวงจรถือหรือ HDL (Hardware Description Language) เป็นเครื่องมือในการออกแบบ ซึ่งเป็นวิธีการที่มีความยืดหยุ่นสูง ทำได้รวดเร็วและไม่จำเป็นต้องทราบลักษณะของวงจรถือที่ต้องการว่าจะเชื่อมต่อกันอย่างไร เพียงแต่กำหนดลักษณะการทำงาน จากนั้นตัวซอฟต์แวร์จะทำการ Synthesis and Optimize ให้ทั้งหมด นอกจากนี้ภาษาที่ใช้ยังเป็นมาตรฐาน เดียวกันสามารถใช้ได้กับชิปทุกตัวและทุกบริษัท
3. การโปรแกรมสามารถทำได้เองและใช้เวลาไม่นาน เพียงแค่ส่งข้อมูลผ่านสายดาวนโหลดทางพอร์ตของคอมพิวเตอร์ก็สามารถโปรแกรมตัวชิปขณะที่อยู่ในระบบได้ โดยไม่จำเป็นต้องถอดมาโปรแกรมข้างนอก ดังรูปที่ 3.6 และที่สำคัญสามารถโปรแกรมได้หลายครั้ง จึงทำให้ง่ายในการแก้ไขและพัฒนาโดยไม่ต้องเสียค่าใช้จ่ายเพิ่มเติมแต่อย่างใด



รูปที่ 3.6 การโปรแกรมลงในชิป

3.5 การออกแบบโดยใช้ภาษาอธิบายพฤติกรรมของฮาร์ดแวร์

ในการออกแบบวงจรดิจิทัลนั้นสามารถทำได้โดยการวาดวงจร (Schematic) หรือใช้ภาษาอธิบายพฤติกรรม (Hardware Description Language) ของฮาร์ดแวร์ ในกรณีของการออกแบบวงจรด้วย ASIC ชนิด Full Custom ผู้ออกแบบจะต้องเขียนวงจรด้วย Schematic จากนั้นจะนำวงจรที่ออกแบบไว้ไปทำการจำลองการทำงาน (Simulate) ซึ่งหากผลออกมาเป็นที่พอใจก็จะต้อง Layout เป็นชั้นสาร และในการออกแบบ ASIC ชนิดนี้ผู้ออกแบบจำเป็นต้องทราบถึง

เทคโนโลยีที่ใช้ในการสร้างด้วย หลังจากได้ Layout ที่สมบูรณ์แล้วจึงจะส่งไปเข้ากระบวนการสร้างไอซีหรือ Fabrication เพื่อสร้างเป็นชิปไอซีออกมา แต่ในการออกแบบวงจรด้วย FPGA โดยการใช้ Schematic หรือใช้ภาษาอธิบายการทำงานของวงจรจะทำได้สะดวกกว่า เนื่องจากวิธีการนี้ผู้ออกแบบไม่จำเป็นต้องคำนึงถึงเทคโนโลยีที่จะใช้สร้างไอซีและที่สำคัญการออกแบบโดยวิธีนี้สามารถแก้ไขโมเดล (Model) หรือเปลี่ยนแปลงเทคโนโลยีได้สะดวกกว่า เพราะไม่ต้องวาดวงจรใหม่นั้นคือการออกแบบโดยใช้ภาษาอธิบายฮาร์ดแวร์จะทำให้โมเดลที่ได้ไม่ขึ้นกับเทคโนโลยี สำหรับภาษาที่ใช้สำหรับอธิบายพฤติกรรมของฮาร์ดแวร์ที่ใช้กันก็มี VHDL, AHDL และ Verilog เป็นต้น ส่วนรายละเอียดของขั้นตอนในการออกแบบสามารถอธิบายได้ดังนี้

1. การสังเคราะห์วงจร (Logic Synthesis)

ในขั้นตอนนี้จะใช้ซอฟต์แวร์ในการสังเคราะห์วงจร (Synthesis Tools) ทำการสังเคราะห์พฤติกรรมของวงจรที่ได้จากการออกแบบด้วย Schematic หรือ VHDL ซึ่งต้องทำการตรวจสอบด้วยว่าซอฟต์แวร์นั้นสนับสนุนเทคโนโลยี FPGA (FPGA Library) ที่ต้องการหรือไม่ ตัวอย่างเช่น FPGA ของบริษัท XILINX และบริษัท ALTERA จะมีซอฟต์แวร์หลายตัวที่สามารถใช้ได้ เช่น Max Plus II ในขั้นตอนนี้ซอฟต์แวร์สังเคราะห์วงจรจะทำการแปลงโค้ด VHDL และทำการ Optimize เพื่อให้ได้วงจรตามเทคโนโลยีที่เลือกใช้ในการสังเคราะห์วงจรนั้น วงจรระดับเกต (Gate Level) จะไม่เหมาะสมกับโครงสร้างที่มีอยู่ในอุปกรณ์ FPGA ดังนั้นในการ Optimize ซอฟต์แวร์สังเคราะห์วงจรจะต้องทำการ Optimize ให้ได้เป็นวงจรที่ประกอบด้วยกลุ่มของลอจิกที่เหมาะสมกับอุปกรณ์ FPGA นั้นๆ จึงทำให้ผลที่ได้มีประสิทธิภาพและในขั้นตอนการสังเคราะห์วงจรนี้ ผู้ออกแบบสามารถกำหนดข้อบังคับสำหรับโมเดลแต่ละตัวได้ เช่น ข้อบังคับในเรื่องเวลา (Timing Constraints) หรือข้อบังคับในเรื่องของพื้นที่ (Area) หรือกำหนดชนิดและตำแหน่งของ I/O ซึ่งข้อบังคับเหล่านี้จะถูกนำไปใช้ในขั้นตอน Optimize เพื่อให้วงจรที่ได้เป็นไปตามที่กำหนด ส่วนสำคัญในการ Optimize คือการเทียบ (Mapping) โมเดลให้เข้ากับเทคโนโลยีที่ใช้เพื่อให้ได้วงจรที่เหมาะสมกับโครงสร้างและสถาปัตยกรรมภายในอุปกรณ์ FPGA เมื่อทำการสังเคราะห์วงจรเสร็จแล้ว ซอฟต์แวร์การสังเคราะห์วงจรก็จะมีรายงานผลว่าโมเดลที่ออกแบบไปนั้นเป็นอย่างไร เช่นมีค่าความหน่วง (Delay) เท่าใด ใช้ทรัพยากรใน FPGA อะไรบ้าง เมื่อมาถึงขั้นตอนนี้ ผู้ออกแบบก็จะทราบว่ามีโมเดลเป็นไปตามข้อบังคับหรือไม่ ถ้าไม่ก็สังเคราะห์ใหม่จนกว่าจะได้ตามที่กำหนด

2. การแบ่งวงจร (Partitioning)

ขั้นตอนนี้เป็นกระบวนการแบ่งวงจรที่ได้จากการสังเคราะห์เป็นส่วนย่อยๆ สำหรับลงใน CLB, IOBs หรือองค์ประกอบอื่นๆ ภายในอุปกรณ์ FPGA สำหรับเกณฑ์ที่ใช้ในการแบ่งคือให้แต่ละส่วนที่จะแยกออกจากกันมีจำนวนสัญญาณที่เชื่อมต่อระหว่างกันน้อยที่สุดเท่าที่จะทำได้ เพื่อลดความหนาแน่นในตอนทำการเชื่อมต่อสัญญาณ (routing) ในขั้นตอนนี้จะใช้ซอฟต์แวร์ทำโดยซอฟต์แวร์จะเทียบส่วนประกอบของวงจรเช่น เกต (gate), ฟลิปฟลอป (flip-flop) ลงในทรัพยากรต่างๆ ที่มีอยู่ในอุปกรณ์ FPGA หลังจากทำขั้นตอนนี้เสร็จแล้วผู้ออกแบบสามารถที่จะทราบว่าวงจรใช้จำนวนทรัพยากรภายในอุปกรณ์ FPGA ไปเท่าไร ส่วนข้อมูลทางเวลานั้นผู้ออกแบบจะทราบเฉพาะความหน่วงภายในแต่ละส่วนเท่านั้น หรือที่เรียกว่าความหน่วงลอจิก (logic delay) ส่วนซอฟต์แวร์จะรวมเอาซอฟต์แวร์ย่อยอื่นๆ อีก เพื่อให้การทำ PPR (Partitioning Placement & Routing) เป็นไปอย่างต่อเนื่อง

3. การวางอุปกรณ์ (Placement)

ขั้นตอนนี้เป็นทางเลือกที่ตั้งของแต่ละส่วนของวงจรที่ผ่านการแบ่งวงจร (Partitioning) มาแล้วว่าควรจะอยู่ ณ ตำแหน่งไหนในอุปกรณ์ FPGA เพื่อให้ได้ผลลัพธ์ที่ดีที่สุด เช่นวงจรส่วนไหนควรอยู่ใกล้กันเพื่อจะได้ค้นหาเส้นทางได้ (route) ง่ายหรือช่วยลดความหน่วงจะเห็นได้ว่าตำแหน่งภายในอุปกรณ์ FPGA นั้นมีความสำคัญเพราะถ้าจัดวางวงจรลงในตำแหน่งที่ไม่เหมาะสมแล้วจะทำให้ความหน่วงเพิ่มขึ้นหรือ Router ทำการค้นหาเส้นทางสัญญาณได้ไม่หมด การวางอุปกรณ์ที่ดีควรวางส่วนต่างๆ ให้อยู่ใกล้กันโดยเฉพาะส่วนที่มีการเชื่อมต่อสัญญาณด้วยกันนอกจากนั้นการกำหนดตำแหน่งขา I/O (I/O pin) ตามตำแหน่งขา I/O ของ FPGA บนแผ่น PCB จะมีผลโดยตรงคือซอฟต์แวร์จะวาง I/O ลงในตำแหน่งที่ผู้ออกแบบกำหนด บางครั้งตำแหน่งที่กำหนดไม่เหมาะสม ดังนั้นการกำหนดขา I/O ควรกำหนดตำแหน่งให้เหมาะสมหรือไม่ก็ให้ซอฟต์แวร์จัดการเอง

4. การเชื่อมต่อสัญญาณ (Routing)

ในขั้นตอนนี้เป็นกระบวนการเชื่อมต่อสัญญาณระหว่างองค์ประกอบต่างๆ ภายในอุปกรณ์ FPGA ขั้นตอนนี้จะทำต่อเนื่องจากการวางอุปกรณ์ ในกรณีที่ทำการวางอุปกรณ์ไว้ไม่ดีซอฟต์แวร์ก็จะทำการเชื่อมต่อสัญญาณได้ไม่หมด (เนื่องจากจำนวนทรัพยากรสำหรับเชื่อมต่อสัญญาณนั้นมีอยู่จำกัด) หรือเกิดความหน่วงเกินค่าที่กำหนดในข้อบังคับ ผู้ออกแบบสามารถทำขั้นตอนนี้ได้โดยใช้ซอฟต์แวร์หรือผู้ออกแบบจะทำการเชื่อมต่อสัญญาณด้วยตนเองก็ได้ แต่ควรใช้ซอฟต์แวร์ทำดีกว่า การกำหนดข้อบังคับทางเวลาจะช่วยให้ผลที่ได้จากการเชื่อมต่อสัญญาณดีขึ้นได้

5. ความหน่วงด้านเวลา (Delay)

ในการทำ FPGA นั้นความหน่วงที่เกิดขึ้นเป็นความหน่วงที่เกิดจากการวางตำแหน่ง (layout) ของอุปกรณ์ ซึ่งผู้ออกแบบไม่สามารถแก้ไขได้ แต่สามารถทำให้มีความหน่วงน้อยที่สุดได้ สำหรับความหน่วงที่เกิดขึ้นนั้นแยกได้เป็นสองประเภทคือ

ความหน่วงลอจิก (Logic delay) เป็นความหน่วงภายในองค์ประกอบของอุปกรณ์ FPGA เอง
ความหน่วงที่เกิดจากการเชื่อมต่อสัญญาณ (Routing delay) เป็นความหน่วงที่เกิดจากการเชื่อมต่อสัญญาณระหว่างองค์ประกอบภายในอุปกรณ์ FPGA

โดยปกติแล้วค่าความหน่วงลอจิกไม่ควรเกิน 50% ของค่าความหน่วงที่ยอมรับได้ เพราะความหน่วงที่เกิดจากการเชื่อมต่อสัญญาณมักจะมีค่ามากกว่าค่าความหน่วงลอจิก ดังนั้นในการวางอุปกรณ์ และเชื่อมต่อสัญญาณผู้ออกแบบควรกำหนดข้อบังคับทางเวลาเพื่อให้ซอฟต์แวร์ได้ทำงานอย่างมีประสิทธิภาพเพิ่มขึ้น และเพื่อให้ได้ผลลัพธ์ที่ดีขึ้นค่าความหน่วงที่ได้หลังจากการวางอุปกรณ์ และเชื่อมต่อสัญญาณแล้วจะมีค่าความหน่วงที่ค่อนข้างแน่นอน ซึ่งผู้ออกแบบสามารถทราบได้ว่าโมเดลที่ออกแบบนั้นเป็นไปตามข้อกำหนดหรือไม่

6. การจำลองการทำงานของวงจร (Simulation)

เป็นขั้นตอนที่ผู้ออกแบบตรวจสอบฟังก์ชันการทำงานของโมเดลว่าถูกต้องหรือไม่ มีข้อผิดพลาดตรงไหนเพื่อจะได้ทำการแก้ไขให้ถูกต้อง ในขั้นตอนนี้จะมีซอฟต์แวร์ที่ใช้สำหรับทำการจำลองการทำงานของวงจรที่ใช้อยู่ เช่น Model Sim ของบริษัท Model Technology หรือ Max Plus II ของบริษัท Altera ในการจำลองการทำงานของวงจรควรทำทุกครั้งหลังจากที่มีการทำแต่ละขั้นตอนหลักเสร็จแล้ว เพื่อจะได้ทราบว่าข้อผิดพลาดของโมเดลเกิดขึ้นตอนไหน และทำการแก้ไขข้อผิดพลาดตรงขั้นตอนนี้ๆ ได้เลย ไม่ต้องมาคอยตรวจหาขั้นตอนที่ทำให้เกิดข้อผิดพลาด นั่นคือ การทำการจำลองการทำงานของวงจร ต้องทำทั้งหลังการเขียนโค้ด, การสังเคราะห์วงจร และการทำ PPR การจำลองการทำงานของวงจรหลังจากที่เขียนโค้ดเสร็จแล้วนั้น ผู้ออกแบบสามารถทราบได้แค่โมเดลทำงานถูกต้องหรือไม่เท่านั้น (functional test) ยังไม่สามารถตรวจสอบการทำงานในเชิงเวลาได้ถูกต้อง ในการจำลองการทำงานของวงจรหลังจากที่สังเคราะห์เป็นวงจรแล้ว เพื่อตรวจสอบว่าฟังก์ชันการทำงานยังคงถูกต้องหรือไม่ และค่าความหน่วงที่เกิดขึ้นเป็นไปตามข้อบังคับหรือมีข้อผิดพลาดเกิดขึ้นหรือไม่ ถ้ามีจะแก้ไขให้ถูกต้อง

ในการจำลองการทำงานของวงจรหลังจากที่ทำการวางอุปกรณ์ การเชื่อมต่อสัญญาณ (post layout simulation) แล้วก็มีความสำคัญเช่นกันเพราะผลที่ได้จากการจำลองการทำงานของวงจรในขณะนี้จะเป็นผลลัพธ์ของโมเดลเลย ซึ่งผู้ออกแบบนอกจากจะตรวจสอบฟังก์ชันการทำงานแล้วยังต้องตรวจสอบคุณสมบัติอื่นๆ เช่น ความหน่วงที่ได้จากการทำ PPR ในรูปแบบค่า

แล้วยังต้องตรวจสอบคุณสมบัติอื่นๆ เช่น ความหน่วงที่ได้จากการทำ PPR ในรูปแบบค่าความหน่วงมาตรฐาน (Standard Delay Format : SDF) ว่าตรงตามที่กำหนดหรือไม่ หรือตรวจสอบว่าวงจรรวม สามารถใช้งานที่ความถี่สูงสุดเท่าไรนั่นเอง ในการจำลองการทำงานของวงจรควรใช้ซอฟต์แวร์ตัวเดียวกันตลอดเพื่อจะได้เปรียบเทียบผลที่ได้จากขั้นตอนต่างๆ

7. การโปรแกรมอุปกรณ์ FPGA (Configuration)

หลังจากที่โมเดลผ่านขั้นตอนต่างๆ จนกระทั่งผ่านการทำ PPR (Partitioning, Placement & Routing) แล้วนั้นก็สามารถที่จะดาวน์โหลด (download) ลงในอุปกรณ์ FPGA ได้แล้ว ในการดาวน์โหลดนี้ก่อนอื่นต้องแปลงแบบวงจรรวมที่ได้เป็นข้อมูลวงจร (configuration data) ซึ่งอยู่ในรูปของบิตสตรีม (bit stream) ก่อนแล้วจึงดาวน์โหลดลงไปเพื่อให้อุปกรณ์ FPGA มี ฟังก์ชันการทำงานตามโมเดลที่ผู้ออกแบบต้องการ ซึ่งในขั้นตอนนี้จะใช้วิธีที่แตกต่างกันออกไปสำหรับอุปกรณ์ FPGA ของแต่ละบริษัทผู้ผลิตคือ ในกรณีที่เป็นอุปกรณ์ FPGA ชนิดที่ต้องโปรแกรมโดยวิธี SRAM นั้น ในการใช้งานผู้ออกแบบจะต้องเก็บข้อมูลวงจรไว้ในหน่วยความจำประเภท EPROM หรือ serial PROM ด้วยเพื่อจะใช้งานสะดวกขึ้น เนื่องในการใช้งานโมเดลครั้งต่อไปไม่ต้องดาวน์โหลดข้อมูลวงจรจากเครื่องคอมพิวเตอร์อีก เพราะมีข้อมูลวงจรเก็บอยู่ในหน่วยความจำอยู่ แล้ว แต่กรณีที่อุปกรณ์ FPGA เป็นชนิดที่โปรแกรมโดยใช้วิธี EPROM หรือ Anti fuse ก็ไม่จำเป็นต้องมีหน่วยความจำสำหรับเก็บข้อมูลวงจร เพราะว่าอุปกรณ์ FPGA ชนิดนี้เมื่อดาวน์โหลดข้อมูลวงจรลงไป ข้อมูลที่ดาวน์โหลดลงไปก็ยังคงอยู่ในอุปกรณ์ FPGA และครั้งต่อไปก็ใช้งานโมเดลที่ออกแบบไว้ได้เลย

3.6 เครื่องมือสำหรับการออกแบบ FPGA

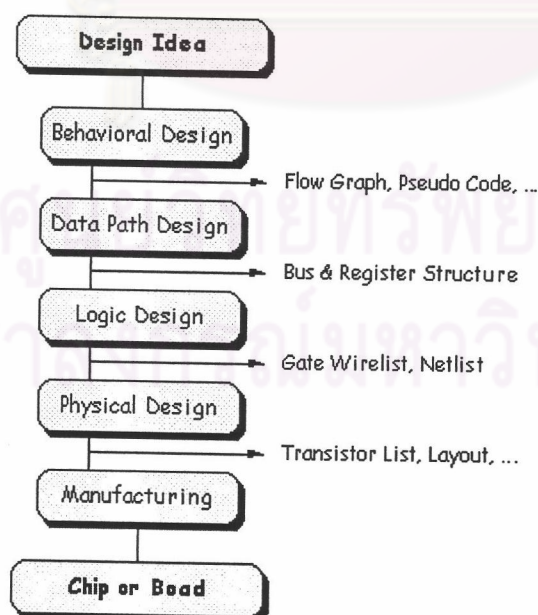
จะเห็นได้ว่าการออกแบบเพื่อทำ FPGA นั้นทำได้สะดวกกว่า ASIC มากเพราะใช้เวลาน้อยกว่า ส่วนสำคัญที่ใช้ในการทำ FPGA คือ ซอฟต์แวร์ที่ใช้ตั้งแต่เขียนโค้ดอธิบายฮาร์ดแวร์จนกระทั่งดาวน์โหลดลงในอุปกรณ์ FPGA ซึ่งซอฟต์แวร์ที่ใช้ต้องเป็นซอฟต์แวร์ที่ทำงานต่อเนื่องกันได้สำหรับซอฟต์แวร์ที่ใช้ทำการจำลองการทำงานของวงจรมันต้องสามารถใช้งานต่อเนื่องกับซอฟต์แวร์ที่ใช้ทั้งระบบเพราะโมเดลที่ได้จากการทำขั้นตอนต่างๆ ต้องเอามาจำลองการทำงานได้ และในการจำลองการทำงานของ วงจรควรใช้ซอฟต์แวร์ตัวเดียวกันตลอดทั้งระบบ เพื่อจะได้เปรียบเทียบผลได้ง่าย

3.7 ภาษา VHDL [7,11]

ความซับซ้อนและขนาดของระบบดิจิทัลในปัจจุบันได้เพิ่มมากขึ้นทุกขณะ ส่งผลให้มีการนำคอมพิวเตอร์เพื่อช่วยในการออกแบบหรือ CAD มาใช้ในขบวนการออกแบบฮาร์ดแวร์เพิ่มขึ้นเช่นกัน อีกทั้งอุปกรณ์และวิธีการออกแบบใหม่ๆ ก็ถูกพัฒนาขึ้นมาเพื่อช่วยอำนวยความสะดวกให้กับนักออกแบบมากขึ้นด้วย สำหรับภาษาบรรยายอุปกรณ์ฮาร์ดแวร์ (HDL : Hardware Description Language) ก็เป็นเครื่องมืออย่างหนึ่งที่ได้รับการพัฒนามาอย่างต่อเนื่อง เพื่อช่วยในการปรับปรุงขบวนการออกแบบระบบดิจิทัลเป็นไปอย่างมีประสิทธิภาพ

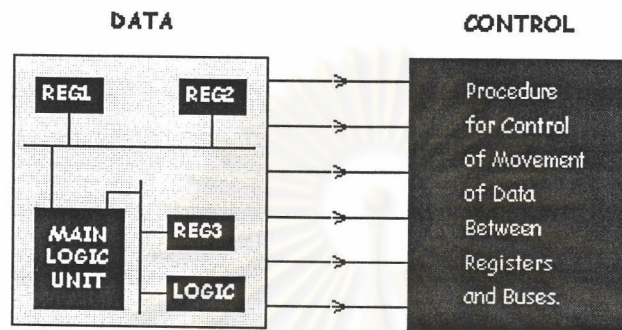
3.8 การออกแบบระบบดิจิทัล

ในการออกแบบระบบดิจิทัล เริ่มตั้งแต่การกำหนดแนวความคิดเบื้องต้นจนกระทั่งได้ออกมาเป็นอุปกรณ์ฮาร์ดแวร์ที่ใช้งานได้จะต้องผ่านขั้นตอนต่างๆ มากมาย และในแต่ละขั้นตอนผู้ออกแบบจะต้องตรวจสอบผลลัพธ์ในแต่ละขั้น ก่อนเข้าสู่กระบวนการออกแบบในขั้นต่อไป รูปที่ 3.7 แสดงขั้นตอนปกติที่ใช้ในการออกแบบระบบดิจิทัลทั่วไป ขั้นแรกผู้ออกแบบจะกำหนดแนวความคิดในการออกแบบแล้วทำการพัฒนาให้สามารถนำมาใช้ได้อย่างสมบูรณ์ ซึ่งภายในขั้นตอนนี้ผู้ออกแบบจำเป็นต้องสร้างรูปแบบระบบในเชิงพฤติกรรมขึ้นมาตรวจสอบซึ่งอาจจะเป็นผังงานแสดงแบบหรือรหัสคำสั่งเทียม (Pseudo code) ก็ได้



รูปที่ 3.7 ขั้นตอนการออกแบบระบบดิจิทัล

ขั้นตอนต่อไปเป็นการออกแบบระบบเส้นทางของข้อมูล ผู้ออกแบบจะกำหนดส่วนประกอบของรีจิสเตอร์และวงจรถลอจิกที่จำเป็นทั้งหมดเพื่อนำมาประกอบเป็นระบบที่สมบูรณ์ โดยแต่ละองค์ประกอบสามารถเชื่อมต่อกันด้วยบัสหนึ่งหรือสองทิศทาง (Unidirectional or Bidirectional Bus) ส่วนกระบวนการในการควบคุมการเคลื่อนย้ายข้อมูลระหว่างรีจิสเตอร์และวงจรถลอจิกจะขึ้นอยู่กับพฤติกรรมของระบบที่กำหนดไว้ดังรูปที่ 3.8



รูปที่ 3.8 การออกแบบระบบเส้นทางของข้อมูล

ขั้นตอนต่อมาเป็นการออกแบบวงจรถลอจิก ซึ่งจะเกี่ยวข้องกับการนำเกตดิจิตอลพื้นฐานและฟลิปฟลอป (flip-flop) มาประกอบเป็นอุปกรณ์ย่อยต่างๆ เช่น รีจิสเตอร์เก็บข้อมูล บัสวงจรถลอจิก และส่วนควบคุมฮาร์ดแวร์ ซึ่งผลลัพธ์ ที่ได้ในขั้นตอนนี้จะเป็นเครือข่ายของการโยงใยระหว่างเกตและฟลิปฟลอปนั่นเอง การออกแบบในขั้นตอนนี้ถัดไปเป็นการเปลี่ยนเครือข่ายการโยงใยที่ได้จากขั้นตอนนี้แล้วให้เป็นลำดับของทรานซิสเตอร์ (Transistor List) และ Layout ซึ่งขั้นตอนนี้จะเกี่ยวข้องโดยตรงกับการจัดวางทรานซิสเตอร์หรือไลบรารีเซลล์เพื่อ แทนเกตและฟลิปฟลอปต่างๆ และในขั้นตอนนี้สุดท้ายจะเป็นการส่งระบบที่ออกแบบไว้ไปทำการเจือสารที่โรงงานเพื่อผลิตออกมาเป็นวงจรรวมในที่สุด

3.9 ข้อกำหนดของภาษา VHDL

VHDL ย่อมาจากคำว่า VHSIC Hardware Description Language (VHSIC : Very High Speed Integrated Circuit) เป็นภาษาโปรแกรมระดับสูง (High Level Language) ที่ใช้สำหรับการออกแบบฮาร์ดแวร์ในระบบดิจิตอล ตัวของภาษาสามารถบรรยายพฤติกรรมการทำงานในรูปของลำดับชั้น (Hierarchy) และสามารถเขียนได้หลายรูปแบบ ด้วยเหตุผลนี้จึงทำให้ภาษา VHDL เป็นเครื่องมือที่ใช้ออกแบบตั้งแต่ขั้นตอนบนสุดคือ แนวความคิดที่จะแก้ปัญหา ลงไปที่ละขั้นจนถึงขั้นตอนของการสร้างวงจรรวม และตัวภาษาก็เปิดโอกาสให้วิศวกรได้พัฒนาและจำลองการทำงาน

ของรูปแบบฟังก์ชันการทำงานของวงจรรายอย่างสังเขปโดยยังไม่ต้องคำนึงถึงรายละเอียดเกี่ยวกับโครงสร้างวงจรรจริง นอกจากนี้ VHDL ยังเป็นภาษาที่สนับสนุนลักษณะต่างๆ ของระบบดิจิทัลที่มีความซับซ้อนได้ทั้งหมด

กระทรวงกลาโหมสหรัฐอเมริกาหรือ DoD (Department of Defense) ได้ตั้งข้อกำหนดสำหรับภาษา VHDL ในเดือนมกราคมปี ค.ศ.1983 ไว้ดังนี้

1. ลักษณะทั่วไป

DoD ได้กำหนดให้ VHDL เป็นภาษาสำหรับการออกแบบและบรรยายของฮาร์ดแวร์ ซึ่งหมายถึงความสามารถในการอธิบายและออกแบบในระดับสูง การจำลอง (Simulation) การสังเคราะห์ (Synthesis) และการทดสอบ (Testing) นอกจากนี้ VHDL ยังถูกกำหนดไว้สำหรับการบรรยายฮาร์ดแวร์ตั้งแต่ระดับบนซึ่งก็คือระบบจนถึงระดับเกิดอีกด้วย เนื่องจากในการทำงานของระบบดิจิทัลนั้นทุกๆ องค์ประกอบภายในระบบไม่ว่าเล็กหรือใหญ่ จะทำงานไปพร้อมๆ กัน ซึ่งในเรื่องของความพร้อมเพรียงในการทำงานนี้ก็ถือเป็นข้อกำหนดที่สำคัญอย่างหนึ่งของ VHDL ด้วยเช่นกัน (สำหรับในภาษาที่ใช้ในการบรรยายฮาร์ดแวร์นั้น ความพร้อมเพรียงจะหมายถึงทุกๆ คำสั่งองค์ประกอบเกิดหรือวงจรถูกนำมาปฏิบัติทั้งหมดดังนั้นในที่สุดแล้วก็จะดูเหมือนว่าได้มีการปฏิบัติไปพร้อมๆ กัน)

2. สนับสนุนการออกแบบแบบลำดับขั้น

การออกแบบแบบลำดับขั้นเป็นลักษณะที่สำคัญอย่างหนึ่งสำหรับการออกแบบระบบที่มีหลายๆ ระดับ โดยในการออกแบบจะประกอบด้วยส่วนการบรรยายการเชื่อมต่อ และส่วนการบรรยายหน้าที่การทำงาน ซึ่งหน้าที่การทำงานของระบบสามารถกำหนดได้ด้วยตัวเอง หรืออาจถูกกำหนดโดยโครงสร้างที่ประกอบด้วยองค์ประกอบย่อยๆ ลงไปได้เช่นกัน แต่ที่ระดับล่างสุด องค์ประกอบต้องถูกบรรยายหน้าที่การทำงานด้วยตัวมันเอง และไม่สามารถกำหนดการทำงานโดยลักษณะแบบโครงสร้างได้

3. ไลบรารี

VHDL ได้สนับสนุนการมีไลบรารีเพื่อระบบการจัดการที่ดี ผู้ออกแบบสามารถกำหนดลักษณะและการทำงานของอุปกรณ์พื้นฐานไว้ในระบบไลบรารี หรือจะใช้ไลบรารีที่ระบบได้จัดเตรียมไว้แล้วก็ได้ โมเดลและการบรรยายที่ถูกต้องควรจัดเก็บไว้ในไลบรารีหลังจากที่ได้ผ่านการคอมไพล์เรียบร้อยแล้วเพื่อให้ผู้ออกแบบคนอื่นๆ สามารถนำไปใช้ได้ด้วย

4. ลำดับคำสั่ง

แม้ว่าการปฏิบัติคำสั่งหรือกระบวนการโดยพร้อมเพรียงกันจะเป็นคุณสมบัติที่สำคัญของ VHDL ก็ตาม ตัวภาษาเองก็ยังมี การจัดเตรียมลักษณะการควบคุมแบบลำดับคำสั่งไว้ให้ด้วย เมื่อผู้ออกแบบได้กำหนดหน้าที่และองค์ประกอบที่ทำงานพร้อมกันของระบบไว้เรียบร้อยแล้ว ผู้ออกแบบยังสามารถบรรยายหน้าที่การทำงานซึ่งเป็นรายละเอียดภายในของแต่ละองค์ประกอบได้ในลักษณะเดียวกับการเขียนโปรแกรมที่ประกอบด้วยโครงสร้างแบบ case, if - then - else และ loop ทั่วๆ ไปได้ การบรรยายแบบลำดับคำสั่งทำให้การออกแบบหน้าที่การทำงานของอุปกรณ์กระทำได้สะดวกและง่ายขึ้น อย่างไรก็ตามโครงสร้างทั้งหมดของ VHDL ก็ยังคงเป็นการทำงานแบบพร้อมเพรียงกันเช่นเดิม

5. การกำหนดคุณสมบัติ

นอกจากการกำหนดอินพุตและเอาต์พุตแล้ว เงื่อนไขอื่นๆ ก็มีผลต่อการปฏิบัติหน้าที่ของ อุปกรณ์ฮาร์ดแวร์ด้วยเช่นกัน โดยสิ่งนี้รวมถึงสภาพแวดล้อมและลักษณะทางกายภาพของ อุปกรณ์นั้นๆ ด้วย ซึ่งภาษาสำหรับการออกแบบที่ดีควรให้ผู้ออกแบบกำหนดคุณสมบัติของ อุปกรณ์ที่ใช้ได้ด้วย เช่น สามารถกำหนดขนาด ลักษณะทางกายภาพเวลาไหล และเงื่อนไขทางสภาพแวดล้อมอื่นๆ ซึ่งความสามารถในการกำหนดคุณสมบัตินี้ก็เป็นส่วนหนึ่งที่มีอยู่ใน ภาษา VHDL ด้วยเช่นกัน

6. ชนิดของข้อมูล

VHDL สามารถกำหนดชนิดของข้อมูลไม่เพียงแต่ชนิด BIT และ BOOLEAN เท่านั้น แต่ยังสามารถกำหนดชนิดของข้อมูลเป็นจำนวนเต็ม จำนวนจริง จุดทศนิยม และชนิดลำดับการนับ (Enumerate Type) หรือแม้แต่ชนิดของข้อมูลที่ผู้ออกแบบกำหนดขึ้นมาเองก็ได้

7. โปรแกรมย่อย

ความสามารถในการใช้ฟังก์ชันและโพรซีเจอร์ (Procedure) ก็เป็นข้อกำหนดอีกอย่างหนึ่งใน VHDL ซึ่งผู้ออกแบบสามารถนำโปรแกรมย่อยมาใช้ในการเปลี่ยนแปลงชนิดของข้อมูล การกำหนดหน่วยของลอจิก การกำหนดตัวกระทำหรือหน้าที่อื่นๆ ตามที่ต้องการได้เช่นเดียวกับการเขียนโปรแกรมทั่วไป

8. การควบคุมเวลา

VHDL อนุญาตให้ผู้ออกแบบสามารถกำหนดเวลาในการส่งผ่านข้อมูลหรือสัญญาณได้ตามต้องการ การตรวจสอบ การออกแบบเกตหรือการห้วงเวลาก็สามารถกระทำได้โดยการกำหนดช่วงเวลาที่แน่นอนหรือกำหนดให้มีการรอคอย เหตุการณ์ (Event) นอกจากนี้ก็ยังสามารถกำหนดรูปแบบของสัญญาณนาฬิกาได้อีกด้วย

9. การกำหนดแบบโครงสร้าง

การกำหนดโครงสร้างขององค์ประกอบต่างๆ สามารถกระทำได้ในทุกระดับของการออกแบบ โดยการกำหนดโครงสร้างขององค์ประกอบรวมที่เกิดจากองค์ประกอบย่อยซึ่งแตกต่างกันหรือเหมือนกันก็เป็นข้อกำหนดอย่างหนึ่งของ VHDL เช่นกัน

3.10 การออกแบบ PLC คอนโทรลเลอร์โดย FPGA

เริ่มจากการกำหนดข้อกำหนดของ PLC ที่จะสร้าง กำหนดจำนวนคำสั่งที่จะสร้าง การจัดแบ่งหน่วยความจำภายในแบ่งออกเป็นหน่วยความจำภายในส่วนโปรแกรม หน่วยจำภายในส่วนข้อมูล กำหนดรีจิสเตอร์และการดำเนินการของแต่ละคำสั่งกับรีจิสเตอร์ หลังจากนั้นทำการเขียนเป็นลักษณะภาษาของ VHDL ทำการจำลองการทำงาน และทดสอบการห้วงเวลา ถ้าผลที่ได้เป็นที่น่าพอใจก็จะทำการโหลดโปรแกรมลง FPGA ชิปจริง และทำการทดสอบทุกคำสั่ง หรือทดสอบฟังก์ชันต่างๆ ของ PLC ให้ครบ แล้วนำไปประกอบเป็น PLC ต้นแบบต่อไป

3.11 การนำ FPGA ซิปมาสร้าง CPU และ PLC

ในประเทศไทยก็ได้เริ่มนำมีการศึกษาและวิจัยนำเอา FPGA ซิปเข้ามาพัฒนาเป็นคอนโทรลเลอร์เฉพาะงาน เช่น ทีวีไมโครคอนโทรลเลอร์ที่สามารถถอดรหัสคำบรรยายภาพที่มีชุดคำสั่ง 35 คำสั่ง [12] หรือในการประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 21 พ.ศ. 2541 ได้มีการนำ FPGA ซิปมาสร้างไมโครคอนโทรลเลอร์ทั่วไป [13] ที่มีชุดคำสั่ง 36 คำสั่ง ซึ่งไมโครคอนโทรลเลอร์ที่ได้ก็สามารถทำงานได้ดี แต่ยังไม่มีการนำ FPGA มาพัฒนาเป็น PLC คอนโทรลเลอร์เลย

ในต่างประเทศจากบทความของ IEEE ได้มีการนำเทคโนโลยีของ FPGA โดยใช้ภาษา VHDL เขียนแทนในลักษณะที่ทำงานแบบโปรแกรมขั้นบันได [14] ซึ่งก็สามารถทำการควบคุมแบบลำดับได้และมีความเร็วในการควบคุมสูง แต่ไม่สะดวกในการแก้ไขโปรแกรม ต้องทำการเขียนโปรแกรมภาษา VHDL ใหม่ทุกครั้งที่มีการเปลี่ยนแปลงโปรแกรมขั้นบันได ทำให้ไม่สะดวก ดังนั้นวิทยานิพนธ์นี้จึงเสนอการสร้าง CPU ของ PLC ให้เป็น PLC คอนโทรลเลอร์โดยใช้ FPGA สร้างแบบจำลองการทำงานภายในให้เหมือนกับ PLC