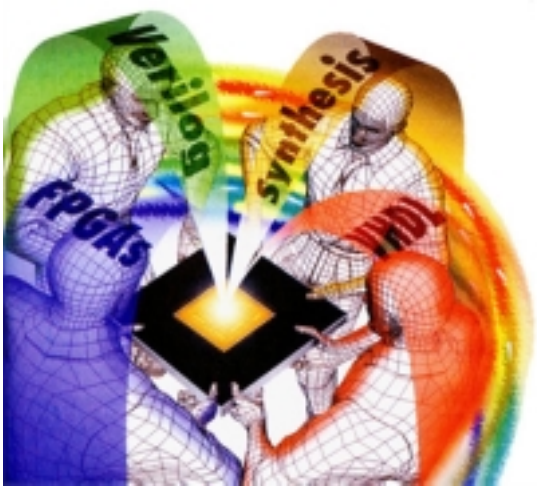
การออกแบบวงจรด้วย FPGA (FPGA Design)

จันทิรา เจือกโว้น[1] และชำนาญ ปัญญาใส[2]



อุปกรณ์ FPGA (Field Programmable Gate Array) เป็นอุปกรณ์ที่ใช้ในการโปรแกรมวงจรที่ได้ออกแบบลงไปเพื่อให้อุปกรณ์ FPGA มีฟังก์ชันการทำงานตามที่ผู้ออกแบบต้องการ ในการทำ FPGA เมื่อเทียบกับการทำ ASICs (Application Specific Integrated Circuits) แล้วนั้นก็มีทั้งข้อดีและข้อเสีย คือ การทำ FPGA จะมีข้อจำกัดในด้านขนาดของวงจรเพราะภายในอุปกรณ์ FPGA จะมีจำนวน gate ให้ใช้จำนวนจำกัด และการทำ FPGA ก็เหมาะกับการทำผลิตภัณฑ์ต้นแบบหรือเพื่อผลิตในปริมาณต่ำ ส่วนข้อดีของการทำ FPGA ก็คือระยะเวลาที่ใช้ในการทำตั้งแต่เขียน code อธิบายฮาร์ดแวร์จนกระทั่ง download นั้นน้อยกว่าการทำ ASIC มาก และการตรวจสอบหรือแก้ไข design ก็ทำได้สะดวก การทำ FPGA ในปัจจุบันมีประสิทธิภาพมากขึ้น และสะดวกขึ้น ทั้งนี้ก็เนื่องจากทางบริษัทผู้ผลิตอุปกรณ์ FPGA ได้เพิ่มความสามารถของอุปกรณ์ FPGA โดยเพิ่มจำนวนองค์ประกอบภายใน หรือปรับปรุงโครงสร้างสถาปัตยกรรมภายใน และยังได้เพิ่มประสิทธิภาพของซอฟต์แวร์ ที่ใช้ทำ PPR (Partitioning, Placement and Routing) สำหรับอุปกรณ์นั้นๆด้วย

FPGA คือ อะไร

FPGA (Field Programmable Gate Array) เป็นอุปกรณ์ที่สามารถโปรแกรมได้ โดยใช้สำหรับโปรแกรมวงจรที่ได้ออกแบบลงไป เพื่อให้อุปกรณ์ FPGA มีฟังก์ชันการทำงานตามแบบที่ต้องการ ซึ่งเป็นวิธีการออกแบบ IC (Integrated Circuit) แบบ Semicustom อีกวิธีหนึ่ง สำหรับตัวอุปกรณ์ FPGA นั้นก็มีโครงสร้างพื้นฐาน เทคโนโลยีที่ใช้สร้าง ตลอดจนเทคนิควิธีการโปรแกรมที่แตกต่างกันสำหรับผู้ผลิตแต่ละราย แสดงดังตารางที่ 1

ตารางที่1 ตัวอย่างอุปกรณ์ FPGA

Product	Gate Count	Architecture	Basic Cell	Programming Method
Actel	2,000-36,000	Gate Array	MUX	Antifuse
Altera	1,000-20,000	Extended PLA	PLA	EPROM
Lucent ORCA	3,500-99,400	Matrix	RAM	SRAM
Xilinx	2,000-62,000	Matrix	RAM block	SRAM

[1] ผู้ช่วยนักวิจัย ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ

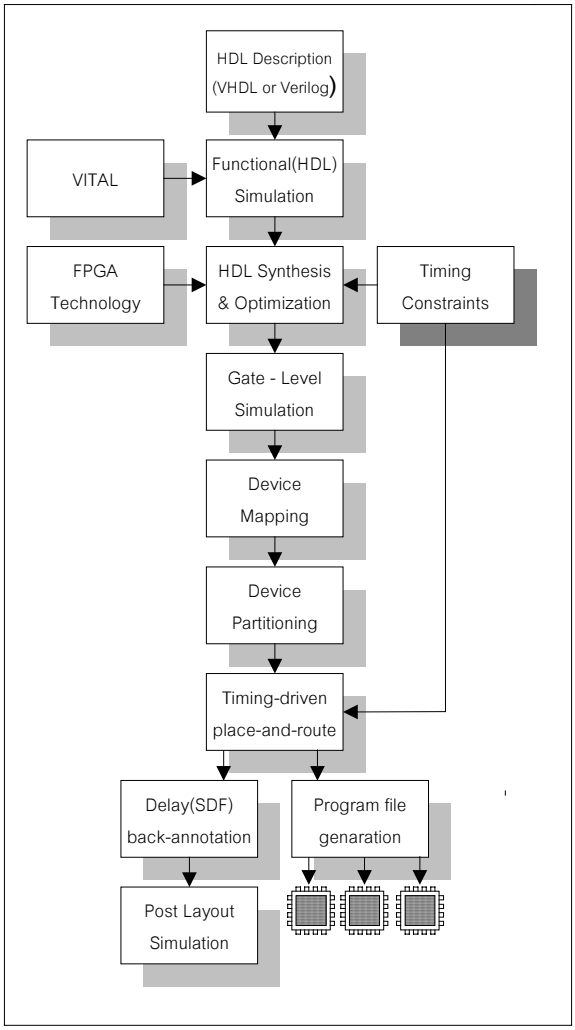
[2] นักวิจัย ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ

นอกจากนั้นอุปกรณ์ FPGA ของแต่ละผู้ผลิตก็มีโครงสร้างและความสามารถที่แตกต่างกันบางส่วนสามารถแสดงดังตารางที่ 2 ส่วนรายชื่อผู้ผลิตอื่นๆสามารถดูได้จากภาคผนวก ก. ซึ่งจะเห็นว่าอุปกรณ์ FPGA มีให้เลือกใช้มากมาย

ตารางที่ 2 แสดงคุณลักษณะอุปกรณ์ FPGA ของ Xilinx

Feature	XC2000	XC3000	XC4000	XCSxxxx
Equivalent gate	600-1,800	2,000-9,000	2,000-62,000	2,000-40,000
Number of CLBs	64-100	64-484	64-2,304	100-784
Input per CLB	4	5	9	9
Flip-flop per CLB	1	2	2	2
Number of IOBs	58-74	64-176	64-384	80-224

ในการใช้งานนั้นอุปกรณ์ FPGA สามารถนำไปประยุกต์ใช้งานได้เช่น ทำ DSP (Digital Signal Processing) หรือ ไมโครคอนโทรลเลอร์ ดังตัวอย่างแสดงในรูปที่ 8 ในการออกแบบวงจรด้วย FPGA ก็จะคล้ายคลึงกับการออกแบบ ASIC ซึ่งมีขั้นตอนต่างๆดังรูปที่ 1



รูปที่1 แสดงขั้นตอนการออกแบบโดยใช้ FPGA

ออกแบบโดยใช้ภาษาอธิบายฮาร์ดแวร์(Hardware Description Language)

ในการออกแบบวงจรดิจิตอล(digital)นั้น ทำได้ทั้งโดยการวาดวงจร(schematic drawing) หรือใช้ภาษาอธิบายฮาร์ดแวร์ (Hardware Description Language) ในขั้นตอนนี้เป็นขั้นตอนที่ไม่แตกต่างกันระหว่างการออกแบบด้วย FPGA และ ASIC ในกรณีที่ใช้ภาษาอธิบายฮาร์ดแวร์ แต่ในกรณีที่ออกแบบโดยวิธีการวาดวงจรจะต่างกันโดยที่การทำวิธีนี้ผู้ทำการออกแบบต้องคำนึงถึงเทคโนโลยีที่จะใช้ซึ่งแต่ละเทคโนโลยีก็มีความแตกต่างกันไป จะเห็นได้ว่าออกแบบโดยใช้ภาษาอธิบายฮาร์ดแวร์ทำได้สะดวกกว่าเพราะการทำวิธีนี้ผู้ออกแบบไม่ต้องมาคำนึงถึงเทคโนโลยีที่จะใช้ และที่สำคัญการออกแบบโดยวิธีนี้สามารถแก้ไขโมเดล(model) หรือเปลี่ยนเทคโนโลยีได้สะดวกกว่าเพราะไม่ต้องวาดวงจรใหม่ นั่นคือการออกแบบโดยใช้ภาษาอธิบายฮาร์ดแวร์จะทำให้โมเดลที่ได้ไม่ขึ้นกับเทคโนโลยี สำหรับภาษาอธิบายฮาร์ดแวร์ที่ใช้ก็มี VHDL หรือ Verilog ในการเขียนโค้ด(code)นั้นสิ่งที่ต้องคำนึงถึงคือจะต้องเขียนอย่างไรจึงจะสามารถสังเคราะห์เป็นวงจรได้และให้โมเดลที่มีคุณสมบัติถูกต้องตามข้อกำหนด เพราะลักษณะการเขียนโค้ดจะมีผลโดยตรงกับโมเดลที่ได้ เนื่องจากในการสังเคราะห์วงจรนั้น ซอฟต์แวร์สังเคราะห์วงจร(Synthesis Tools) จะสังเคราะห์วงจรตามโค้ด เช่นในการเขียนโค้ด VHDL อธิบายการทำงานของโมเดลวงจรอันเดียวกัน แต่เขียนโค้ดในลักษณะที่ต่างกัน เมื่อสังเคราะห์จะได้วงจรต่างกัน และจากวงจรที่ต่างกัน เมื่อนำไปทำต้นแบบด้วย FPGA หรือการทำ ASIC แล้วจะได้ไอซี ที่มีคุณสมบัติต่างกันทั้งในด้านของขนาดหรือความเร็ว (area and time) ดังนั้นการออกแบบในขั้นตอนนี้ผู้ออกแบบต้องระมัดระวังเป็นพิเศษ ส่วนการที่จะเขียนโค้ดในลักษณะใดเพื่อให้ได้ผลลัพธ์ที่ดีที่สุดนั้นก็ขึ้นอยู่กับประสบการณ์ของผู้ออกแบบ

การสังเคราะห์วงจร (Logic Synthesis)

ในขั้นตอนนี้จะใช้ซอฟต์แวร์สังเคราะห์วงจร (synthesis tools) ทำการสังเคราะห์โค้ด VHDLหรือ Verilog เพื่อให้ได้เป็นวงจรขึ้นมา ซึ่งทำได้โดยใช้ซอฟต์แวร์อีกเช่นกันแต่ต้องตรวจสอบด้วยว่า ซอฟต์แวร์ นั้นๆสนับสนุนเทคโนโลยี FPGA (FPGA Library) ที่ต้องการใช้หรือไม่ ตัวอย่าง FPGA ที่มีการใช้งานแพร่หลายเช่น FPGA ของบริษัท Xilinx และบริษัท Altera ในการทำ FPGA นั้นมีซอฟต์แวร์หลายตัวที่สามารถใช้ได้ เช่น Synopsys, Exemplar ฯลฯ ในขั้นตอนนี้ซอฟต์แวร์สังเคราะห์วงจรจะแปลงโค้ด VHDL และทำออพติไมซ์ (optimization) เพื่อให้ได้วงจรตามเทคโนโลยีที่เลือกใช้ ในการสังเคราะห์วงจรเพื่อทำ FPGA นั้นวงจรระดับเกต (gate-level)ไม่เหมาะสมกับโครงสร้างที่มีอยู่ในอุปกรณ์ FPGA ดังนั้นในการออพติไมซ์นั้นซอฟต์แวร์สังเคราะห์วงจร จะต้องทำการออพติไมซ์ให้ได้เป็นวงจรที่ประกอบด้วยกลุ่มของลอจิก(logic) ที่เหมาะสมกับอุปกรณ์ FPGA นั้นๆ จึงจะทำให้ผลลัพธ์ที่ได้มีประสิทธิภาพ และในขั้นตอนการสังเคราะห์วงจรนี้ผู้ออกแบบสามารถกำหนดข้อบังคับสำหรับโมเดลได้เช่น ข้อบังคับในเรื่องของเวลา(timing constraints) หรือข้อบังคับในเรื่องของพื้นที่(area) หรือกำหนดชนิดและตำแหน่งของ I/O ซึ่งข้อบังคับเหล่านี้จะถูกนำไปใช้ในขั้นตอนออพติไมซ์เพื่อให้วงจรที่ได้เป็นไปตามที่กำหนด ส่วนสำคัญในการออพติไมซ์คือการเทียบ(mapping) โมเดลให้เข้ากับเทคโนโลยีที่ใช้เพื่อให้ได้วงจรที่เหมาะสมกับโครงสร้างสถาปัตยกรรมภายในอุปกรณ์ FPGA ในกรณีของ Xilinx FPGA จะเทียบโดยใช้วิธี LUT(Look Up Table)

เมื่อทำการสังเคราะห์วงจรเสร็จแล้วซอฟต์แวร์สังเคราะห์วงจรก็จะมีการรายงานผลว่าโมเดลที่ออกแบบไปนั้นเป็นอย่างไร เช่น มีความหน่วง(delay)เท่าไร ใช้ทรัพยากรต่างๆใน FPGA อะไรบ้าง เมื่อมาถึงขั้นตอนนี้ผู้ออกแบบก็จะทราบว่าโมเดลเป็นไปตามข้อบังคับหรือไม่ ถ้าไม่ก็ให้ทำการสังเคราะห์ใหม่จนกว่าจะเป็นไปตามที่กำหนด ในขั้นตอนการสังเคราะห์นี้มีเทคนิคต่างๆที่ผู้ออกแบบนำมาใช้เพื่อให้โมเดลเป็นไปตามข้อบังคับที่กำหนด

การแบ่งวงจร (Partitioning)

ขั้นตอนนี้เป็นการแบ่งวงจรที่ได้จากการสังเคราะห์ เป็นส่วนย่อยๆสำหรับลงใน CLBs, IOBS หรือองค์ประกอบอื่นๆภายในอุปกรณ์ FPGA สำหรับเกณฑ์ที่ใช้ในการแบ่งคือให้แต่ละส่วนที่จะแยกออกจากกันมีจำนวนสัญญาณที่เชื่อมต่อระหว่างกันน้อยที่สุดเท่าที่จะทำได้ เพื่อช่วยลดความหนาแน่นในตอนทำการเชื่อมต่อสัญญาณ (routing) ในขั้นตอนนี้จะใช้ซอฟต์แวร์ทำ โดยซอฟต์แวร์จะเทียบส่วนประกอบของวงจรเช่น เกท(gate), ฟลิป-ฟลอป(flip-flop) ลงในทรัพยากรต่างๆที่มีอยู่ภายในอุปกรณ์ FPGA (CLBs, IOBs, BUFT และ edge decoder)

หลังจากทำขั้นตอนนี้เสร็จแล้วผู้ออกแบบสามารถที่จะทราบว่าวงจรใช้จำนวนทรัพยากรภายในอุปกรณ์ FPGA ไปเท่าไร ส่วนข้อมูลทางเวลานั้นผู้ออกแบบจะทราบเฉพาะความหน่วงภายในแต่ละส่วนเท่านั้น หรือที่เรียกว่าความหน่วงลอจิก(logic delay) ส่วนซอฟต์แวร์ที่ใช้ในขั้นตอนนี้เช่น M1 ของ Xilinx ซึ่งซอฟต์แวร์ตัวนี้จะรวมเอาซอฟต์แวร์ ย่อยอื่นๆ อีกเพื่อให้การทำ PPR (Partitioning, Placement & Routing) เป็นไปอย่างต่อเนื่อง

การวางอุปกรณ์(Placement)

ขั้นตอนนี้ก็เป็นการเลือกทำเลที่ตั้งของแต่ละส่วนของวงจรที่ผ่านการแบ่งวงจร (partitioning) มาแล้วว่าควรจะอยู่ ณ ตำแหน่งไหนในอุปกรณ์ FPGA เพื่อให้ได้ผลลัพธ์ที่ดีที่สุด เช่นวงจรส่วนไหนควรอยู่ใกล้กันเพื่อจะได้ค้นหาเส้นทาง (route)ได้ง่าย หรือช่วยลดความหน่วง จะเห็นได้ว่าตำแหน่งภายในอุปกรณ์ FPGA นั้นมีความสำคัญเพราะถ้าจัดวางวงจรลงในตำแหน่งที่ไม่เหมาะสมแล้วจะทำให้ความหน่วงเพิ่มขึ้นหรือ Router ทำการค้นหาเส้นทางสัญญาณได้ไม่หมด

การวางอุปกรณ์ที่ดีควรวางส่วนต่างให้อยู่ใกล้กันโดยเฉพาะส่วนที่มีการเชื่อมต่อสัญญาณด้วยกัน นอกจากนั้นการกำหนดตำแหน่งขา I/O(I/O pin) ตามตำแหน่งขา I/O ของ FPGA บนแผ่น PCB ก็จะมีผลโดยตรงเลยคือซอฟต์แวร์ จะวาง I/O ลงในตำแหน่งที่ผู้ออกแบบกำหนด ซึ่งบางครั้งตำแหน่งที่กำหนดไปไม่เหมาะสม ดังนั้นการกำหนดขา I/O ควรกำหนดตำแหน่งให้เหมาะสม หรือไม่ก็ให้ซอฟต์แวร์จัดการเอง ในกรณีที่ใช้ซอฟต์แวร์ M1 ผู้ออกแบบสามารถแก้ไขตำแหน่งใหม่ได้แต่ควรกระทำด้วยความระมัดระวังเป็นอย่างยิ่ง แต่วิธีที่ดีที่สุดคือการให้ซอฟต์แวร์ทำซ้ำหลายๆครั้งเพื่อหาครั้งที่ดีที่สุด สำหรับเกณฑ์ที่ใช้ในการตัดสินคือความหน่วงที่ได้หลังจากทำการค้นหาเส้นทางแล้ว หรือที่เรียกว่า routing delay

การเชื่อมต่อสัญญาณ (Routing)

ในขั้นตอนนี้เป็นการเชื่อมต่อสัญญาณระหว่างองค์ประกอบต่างๆภายในอุปกรณ์ FPGA เช่นระหว่าง CLBs หรือระหว่าง CLBs กับ IOBs ขั้นตอนนี้จะทำต่อเนื่องจากการวางอุปกรณ์ ในกรณีที่ทำการวางอุปกรณ์ไว้ ไม่ดีซอฟต์แวร์ก็จะทำการเชื่อมต่อสัญญาณได้ไม่หมด(เนื่องจากจำนวนทรัพยากร (resource) สำหรับเชื่อมต่อสัญญาณนั้นมีอยู่จำกัด) หรือเกิดความหน่วงเกินค่าที่กำหนดในข้อบังคับ

ผู้ออกแบบสามารถทำขั้นตอนนี้ได้โดยใช้ซอฟต์แวร์เช่น M1 ของบริษัท Xilinx หรือผู้ออกแบบจะทำการเชื่อมต่อสัญญาณด้วยตัวเอง (manual layout) ก็ได้ แต่ทางที่ดีควรใช้ซอฟต์แวร์ทำดีกว่าโดยให้ ทำการค้นหาเส้นทางหลายๆครั้งเพื่อหาครั้งที่ดีที่สุด นอกจากนั้นการกำหนดข้อบังคับทางเวลา(timing constraints)จะช่วยให้ผลที่ได้จากการทำเชื่อมต่อสัญญาณดีขึ้นได้

ความหน่วงด้านเวลา (Delay)

ในการทำ FPGA นั้นความหน่วงที่เกิดขึ้นเป็นความหน่วงที่เกิดจากการวางตำแหน่ง(layout)ของอุปกรณ์ ซึ่งผู้ออกแบบไม่สามารถเข้าไปแก้ไขได้ แต่สามารถทำให้มีความหน่วงน้อยที่สุดได้ สำหรับความหน่วงที่เกิดขึ้นนั้นแยกได้เป็นสองประเภทคือ ความหน่วงลอจิก (Logic delay)เป็นความหน่วงภายในองค์ประกอบของอุปกรณ์ FPGA เอง เช่นความหน่วงภายใน CLBs และความหน่วงที่เกิดจากการเชื่อมต่อสัญญาณ (Routing delay)เป็นความหน่วงที่เกิดจากการเชื่อมต่อสัญญาณระหว่างองค์ประกอบภายในของอุปกรณ์ FPGA โดยปกติแล้วค่าความหน่วงลอจิกไม่ควรเกิน 50% ของค่าความหน่วงที่ยอมรับได้ เพราะว่าความหน่วงที่เกิดจากการเชื่อมต่อสัญญาณมักจะมีค่ามากกว่าค่าความหน่วงลอจิกดังนั้นในการวางอุปกรณ์ และเชื่อมต่อสัญญาณ ผู้ออกแบบควรกำหนดข้อบังคับทางเวลาเพื่อให้ ซอฟต์แวร์ได้ทำงานอย่างมีประสิทธิภาพเพิ่มขึ้น และเพื่อให้ได้ผลลัพธ์ที่ดีขึ้น

ค่าความหน่วง ที่ได้หลังจากการวางอุปกรณ์ และเชื่อมต่อสัญญาณแล้วเป็นค่าความหน่วงที่ค่อนข้างแน่นอน ซึ่งผู้ออกแบบสามารถทราบได้ว่าโมเดลที่ออกแบบนั้น เป็นไปตามข้อกำหนดหรือไม่

การโปรแกรมอุปกรณ์ FPGA (Configuration)

หลังจากที่โมเดลผ่านขั้นตอนต่างๆจนกระทั่งผ่านการทำ PPR (Partitioning, Placement & Routing แล้วนั้น ถึงตอนนี้ก็สามารถที่จะดาวน์โหลด (download) ลงในอุปกรณ์ FPGA ได้แล้ว ในการดาวน์โหลดนี้ก่อนอื่นต้องแปลงแบบวงจรรวมที่ได้เป็นข้อมูลวงจร(configuration data)ซึ่งอยู่ในรูปของบิตสตรีม(bit-stream) ก่อน แล้วจึงดาวน์โหลดลงไปเพื่อให้อุปกรณ์ FPGA มีฟังก์ชันการทำงานตามโมเดลที่ผู้ออกแบบต้องการ ซึ่งในขั้นตอนนี้จะใช้วิธีที่แตกต่างกันออกไปสำหรับอุปกรณ์ FPGA ของแต่ละบริษัทผู้ผลิตคือ ในกรณีที่เป็นอุปกรณ์ FPGA ชนิดที่ต้องโปรแกรมโดยวิธี SRAM นั้น ในการใช้งานผู้ออกแบบต้องเก็บข้อมูลวงจร(configuration data) ไว้ในหน่วยความจำประเภท EPROM หรือ serial PROM ด้วยเพื่อจะได้ใช้งานสะดวกขึ้น คือในการใช้งานโมเดลครั้งต่อไปไม่ต้องดาวน์โหลดข้อมูลวงจรจากเครื่องคอมพิวเตอร์อีกเพราะมีข้อมูลวงจรเก็บอยู่ในหน่วยความจำอยู่แล้ว แต่ในกรณีที่อุปกรณ์ FPGA เป็นชนิดที่โปรแกรมโดยใช้วิธี EPROM หรือ Antifuse ก็ไม่จำเป็นต้องมีหน่วยความจำไว้สำหรับเก็บข้อมูลวงจร เพราะว่าอุปกรณ์ FPGA ชนิดนี้เมื่อดาวน์โหลดข้อมูลวงจรลงไปแล้ว ข้อมูลที่ดาวน์โหลดลงไปก็ยังคงอยู่ในอุปกรณ์ FPGA และครั้งต่อไปก็ใช้งานโมเดลที่ออกแบบไว้ได้เลย

การจำลองการทำงานของวงจร (Simulation)

ในขั้นตอนนี้เป็นขั้นตอนที่สำคัญอีกขั้นตอนหนึ่ง เพราะเป็นขั้นตอนที่ผู้ออกแบบตรวจสอบฟังก์ชันการทำงานของโมเดลว่าถูกต้องหรือไม่ มีข้อผิดพลาดตรงไหนเพื่อจะได้ทำการแก้ไขให้ถูกต้อง ในขั้นตอนนี้จะมี ซอฟต์แวร์ที่ใช้สำหรับทำการจำลองการทำงานของวงจร ที่ใช้อยู่เช่น ModelSim ของบริษัท Model Technology ในการจำลองการทำงานของวงจร ควรทำทุกครั้งหลังจากที่มีการทำแต่ละขั้นตอนหลักเสร็จแล้ว เพื่อจะได้ทราบว่าข้อผิดพลาดของโมเดลเกิดจากขั้นตอนไหนจะได้แก้ไขข้อผิดพลาดตรงขั้นตอนนั้นๆได้เลย ไม่ต้องมาคอยตรวจหาขั้นตอนที่ทำให้เกิดข้อผิดพลาด นั่นคือการทำการจำลองการทำงานของวงจร ต้องทำทั้งหลังจากเขียนโค้ด, การสังเคราะห์วงจร และการทำ PPR การจำลองการทำงานของวงจรหลังจากที่เขียนโค้ดเสร็จแล้วนั้น ผู้ออกแบบสามารถทราบได้แค่โมเดลทำงานถูกต้องหรือไม่เท่านั้น (functional test) แต่ยังไม่สามารถตรวจสอบการทำงานในเชิงเวลาได้ถูกต้อง ในการจำลองการทำงานของวงจรหลังจากที่สังเคราะห์เป็นวงจรแล้วเพื่อตรวจสอบว่าฟังก์ชันการทำงานยังคงถูกต้องอยู่หรือไม่ และค่าความหน่วงที่เกิดขึ้นเป็นไปตามข้อบังคับหรือไม่ มีข้อผิดพลาดเกิดขึ้นหรือไม่ ถ้ามีก็จะได้แก้ไขให้ถูกต้อง ในการจำลองการทำงานของวงจรหลังจากที่ทำการวางอุปกรณ์ และเชื่อมต่อสัญญาณ (post layout simulation) แล้วก็มีความสำคัญเช่นกันเพราะผลที่ได้จากการจำลองการทำงานของวงจรในตอนนี้จะเป็นผลลัพธ์ของโมเดลเลย ซึ่งผู้ออกแบบนอกจากจะตรวจสอบฟังก์ชันการทำงานแล้วยังต้องตรวจสอบคุณสมบัติอื่นๆเช่น ความหน่วงที่ได้จากการทำ PPR ในรูปแบบค่าความหน่วงมาตรฐาน (Standard Delay Format : SDF) ว่าตรงตามที่กำหนดไว้หรือไม่ หรือตรวจสอบว่าแบบวงจรรวมสามารถใช้งานที่ความถี่สูงสุดเท่าไร นั่นเอง ในการจำลองการทำงานของวงจรควรใช้ ซอฟต์แวร์ตัวเดียวกันตลอดเพื่อจะได้เปรียบเทียบผลที่ได้จากขั้นตอนต่างๆ

ซอฟต์แวร์ FPGA (FPGA Design Tools)

จะเห็นได้ว่าการออกแบบเพื่อทำ FPGA นั้นทำได้สะดวกกว่าการทำ ASIC มากเพราะใช้เวลาน้อยกว่ามากด้วย ส่วนสำคัญที่ใช้ในการทำ FPGA คือ ซอฟต์แวร์ที่ใช้ตั้งแต่เขียนโค้ดอธิบายฮาร์ดแวร์จนกระทั่งดาวน์โหลดลงในอุปกรณ์ FPGA ซึ่งซอฟต์แวร์ที่ใช้ต้องเป็น ซอฟต์แวร์ที่ใช้ทำงานต่อเนื่องกันได้ คือ ซอฟต์แวร์ที่ใช้ทำการสังเคราะห์วงจรกับ ซอฟต์แวร์ ที่ใช้ทำ PPR เช่น Leonardo กับ M1 หรือ Synopsys กับ M1 นั่นคือ M1 ต้องสามารถทำ PPR วงจรที่สังเคราะห์โดยซอฟต์แวร์ Leonardo หรือ Synopsysได้ สำหรับซอฟต์แวร์ที่ใช้ทำการจำลองการทำงานของวงจรนั้น ต้องสามารถใช้งานต่อเนื่องกับซอฟต์แวร์ที่ใช้ทั้งระบบ เพราะโมเดลที่ได้จากการทำขั้นตอนต่างๆ(ด้วยซอฟต์แวร์ต่างๆ)ต้องเอามาจำลองการทำงานได้ และในการจำลองการทำงานของวงจร ควรใช้ ซอฟต์แวร์ตัวเดียวกันตลอดทั้งระบบ เพื่อจะได้เปรียบเทียบผลได้ง่าย ในอดีตซอฟต์แวร์ส่วนใหญ่จะใช้งานอยู่บนคอมพิวเตอร์สมรรถนะสูงอย่างเวิร์คสเตชั่น (Workstation) ในปัจจุบันมีการพัฒนาซอฟต์แวร์ที่ใช้งานบนพีซี (PC)มากขึ้น ซึ่งสามารถลดค่าใช้จ่ายในด้านอุปกรณ์คอมพิวเตอร์ ส่วนรายชื่อตัวอย่างซอฟต์แวร์และบริษัทผู้ผลิตที่ใช้ในการออกแบบแต่ละขั้นตอนแสดงดังตารางที่ 3 ส่วนรายชื่อผู้ผลิตอื่นๆสามารดูได้จากภาคผนวก

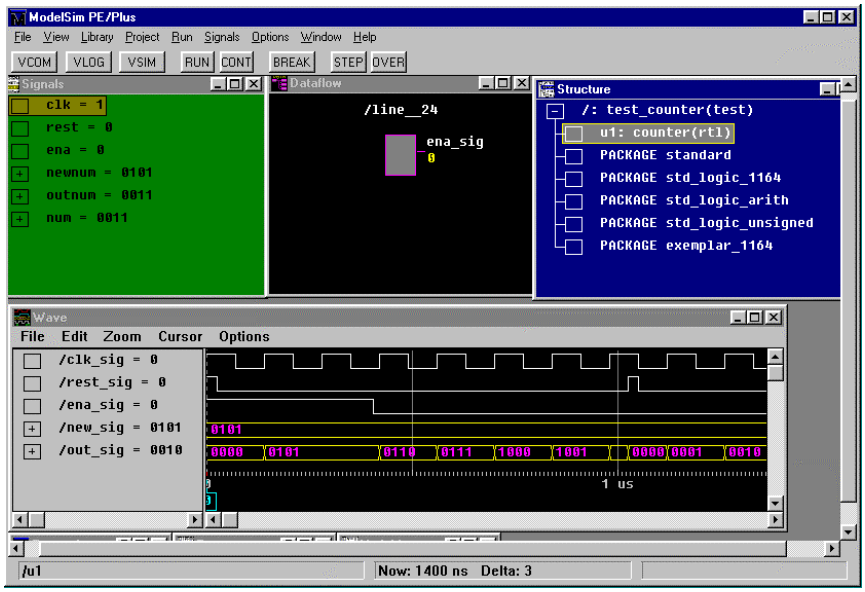
ตารางที่ 3 แสดงตัวอย่างรายชื่อซอฟต์แวร์และผู้ผลิตที่ใช้ในการออกแบบแต่ละขั้นตอน

ขั้นตอนการออกแบบ	ซอฟต์แวร์ที่ใช้	ผู้ผลิต
HDL Simulation	ModelSim/VHDL	Model Technology
HDL Synthesis & Optimization	Leonardo	Exemplar Logic
Partitioning, Placement and Routing	Foundation(M1)	Xilinx

จากที่กล่าวมาเป็นการอธิบายขั้นตอนวิธีการออกแบบวงจรเพื่อทำ FPGA และเพื่อให้เข้าใจในแต่ละขั้นตอนยิ่งขึ้น ก็จะแสดงตัวอย่างโมเดลที่ได้ออกแบบซึ่งเป็นการออกแบบเครื่องนับแบบ 4 บิท (4 bit counter)มีโค้ด VHDL แสดงดังรูปที่ 2 ผลของการจำลองการทำงานของวงจรแสดงดังรูปที่ 3

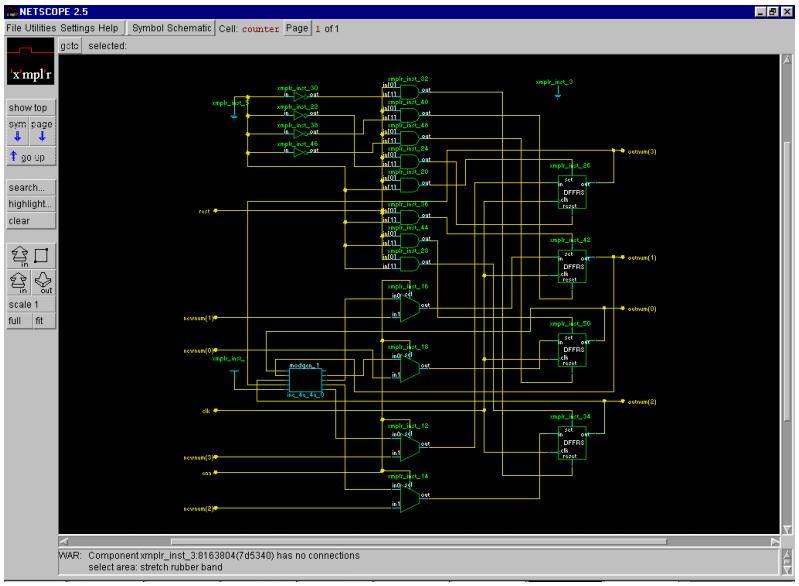
```
library ieee, work;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
use work.all;
library exemplar;
use exemplar.exemplar_1164.all;

entity counter is
port (clk,rest,ena:in std_logic;
      newnum:in std_logic_vector(3 downto 0);
      outnum:out std_logic_vector(3 downto 0));
end counter;

architecture rtl of counter is
signal num :std_logic_vector(3 downto 0);
begin
process(rest,clk)
begin
    if rest ='1' then
       num <= "0000";
    elsif clk'event and clk ='1'then
        if ena = '1'then
           num <= newnum;
        else num <= num + '1';
        end if;
    end if;
end process;
outnum <= num;
end rtl;
```

รูปที่ 2 แสดงตัวอย่างการออกแบบโดยใช้ ภาษาอธิบายฮาร์ดแวร์ VHDL

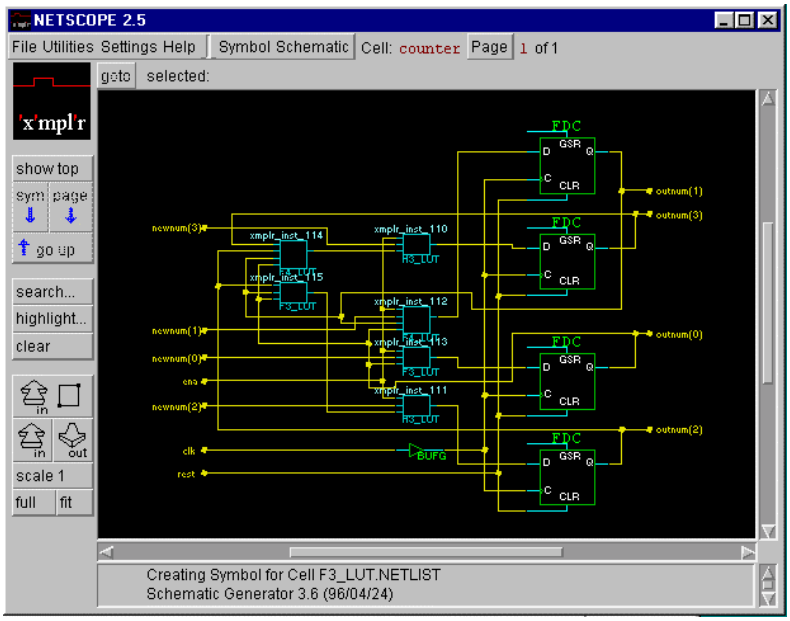


รูปที่ 3 แสดงผลการจำลองการทำงานของวงจรโดยใช้ Model Sim

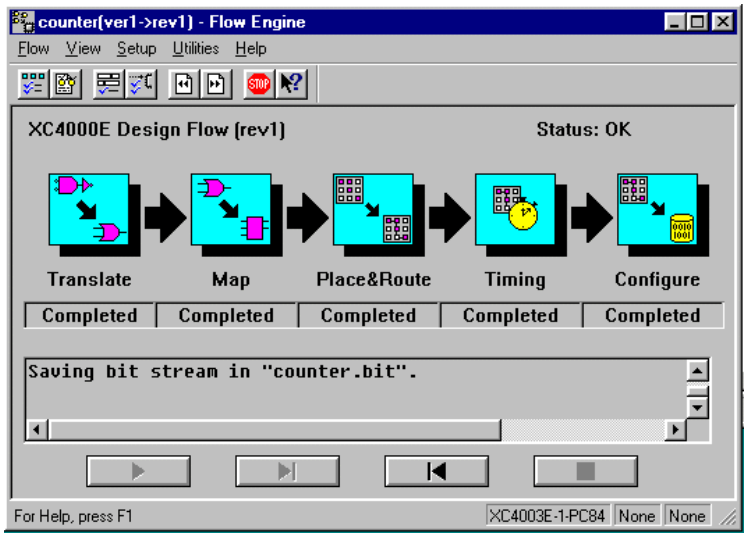
หลังจากนั้นก็นำโค้ด VHDL มาสังเคราะห์ด้วยซอฟต์แวร์ Leonardo โดยใช้เทคโนโลยี FPGA ของบริษัท Xilinx เป็นวงจรได้ดังรูปที่ 4 ซึ่งในรูปนี้ซอฟต์แวร์ (Leonardo)จะสังเคราะห์วงจรตามโค้ด VHDL และรูปที่ 5 ซึ่งแสดงบางส่วนของวงจรที่ได้หลังจากเทียบ(mapping)วงจรในรูปที่ 4 เข้ากับเทคโนโลยีที่เลือกใช้ (XC4003E) ต่อจากนั้นก็ได้มีการทำ PPR (Partitioning, Placement & Routing) โดย ซอฟต์แวร์ M1 ซึ่งแสดงผลที่ได้ดังรูปที่ 7 ในกรณีตัวอย่างนี้มีการใช้ ฟลิปฟล็อปจำนวน 4 ตัวซึ่งอยู่ใน CLB จำนวน 2 CLB จากจำนวน CLB ที่มีทั้งหมด 100 CLB ในรูปที่ 8 แสดงการใช้ FPGA ในผลิตภัณฑ์จำพวกคอมพิวเตอร์



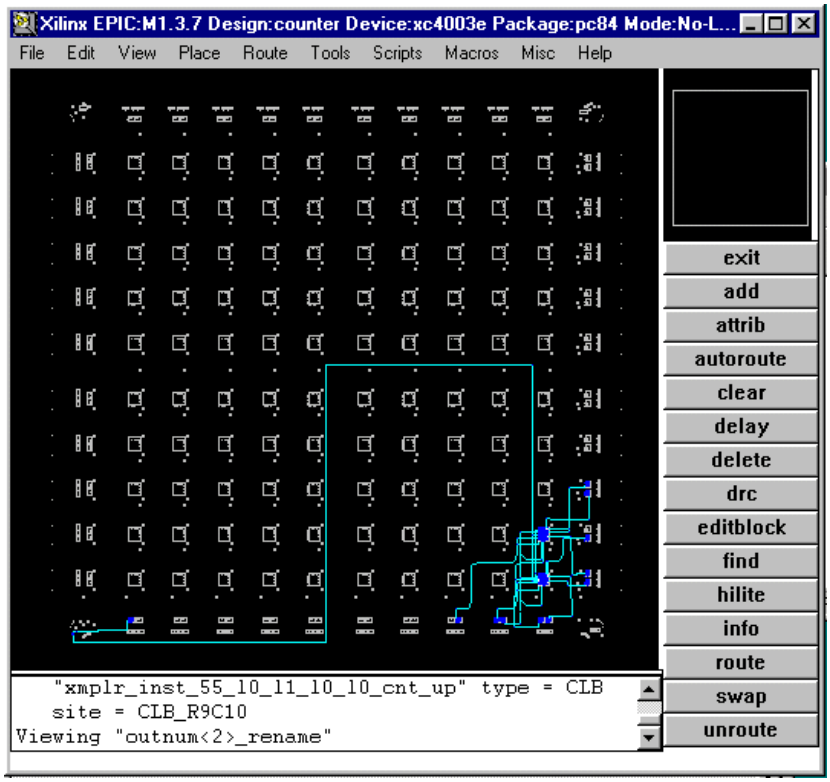
รูปที่ 4 แสดงวงจรที่ได้หลังจากทำการสังเคราะห์ด้วยซอฟต์แวร์ Leonardo
โดยเทคโนโลยี XC4003E ของบริษัท Xilinx ก่อนทำการออพติไมซ์



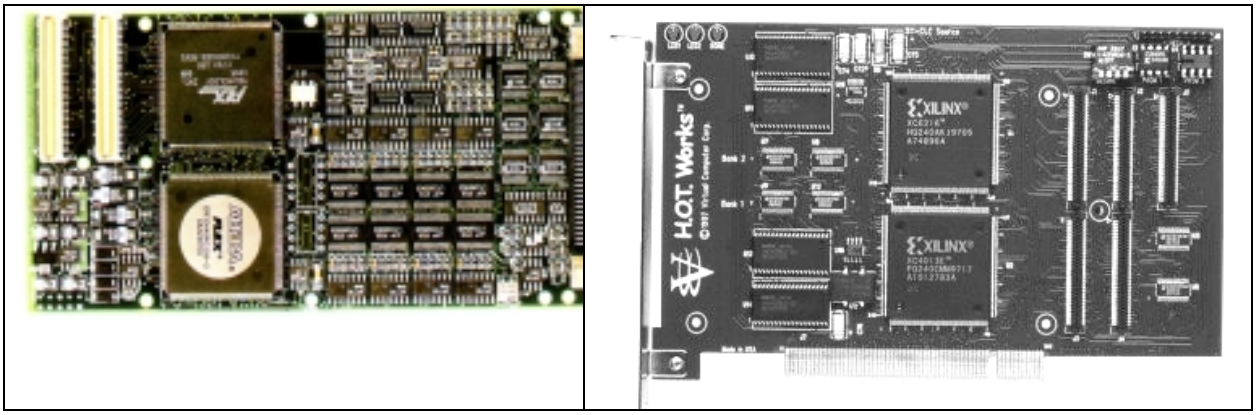
รูปที่ 5 แสดงวงจรหลังจากที่เทียบ(mapping) และทำการออพติไมซ์เข้ากับเทคโนโลยี XC4003E โดยซอฟต์แวร์ Leonardo



รูปที่ 6 แสดงขั้นตอนการทำงานของซอฟต์แวร์ Foundation M1 ซึ่งง่ายต่อการเข้าใจ



รูปที่ 7 แสดงรูปหลังจากที่ทำ PPR (Partitioning,Placement and Routing)



รูปที่ 8 แสดงการนำอุปกรณ์ FPGA ไปใช้งาน

แหล่งข้อมูลเพิ่มเติม

ข้อมูลเพิ่มเติมเกี่ยวกับ FPGA สามารถเข้าไปดูได้ตามเวบไซต์ต่างๆ เช่น http://www.mrc.uidaho.edu/fpga หรือตามวารสารต่างๆ เช่น วารสาร Computer Design (www.computer-design.com) Integrated System Design (www.isd.com) และ EDN (www.ednmag.com) เป็นต้น นอกจากนี้ท่านยังสามารถหาข้อมูลเพิ่มเติมได้จากเวบไซต์ผู้ผลิตอุปกรณ์ FPGA และซอฟต์แวร์ต่างๆ ดังรายชื่อในภาคผนวก

เอกสารอ้างอิง

1 Douglas J. Smith, *HDL basic training : top-down chip design using Verilog and VHDL*, EDN, October 1996

2 Micheal John Sebastian Smith, *Application-Specific Integrated Circuits*, Addison Wesley, 1997

3 1998 Vendor Guide, Supplement to Integrated System Design, January 1998

4 Pak K.Chan and Samiha Mourad, *Digital Design Using Filed Programmable Gate Arrays*, Prentice Hall, 1994

5 Xilinx, *The Programmable Logic Data Book*,1994

6 Steve Wolfe, *Top-down FPGA design from synthesis to silicon*, Electronic Engineering, October 1994

ภาคผนวก

Windows EDA : FPGA & CPLD Design

Company	Product	Description	Windows Platform
Altera www.altera.com	MAX+Plus II	Provides a broad range of features with a graphical user interface. Interfaces to industry-standard EDA tools	Windows 95 & NT
Atmel fpga@atmel.com www.atmel.com	FPGA/DSP Designer 4.0	Design suite for implementing 2,000-to 100,000-gate designs into AT6000 series coprocessor FPGAs	Windows 3.1, 95, & NT
Cypress Semiconductor lis@cypress.com www.cypress.com	Warp2	Compiles VHDL design input and then outputs a JEDEC file for the selected PLD or CPLD. For simulation, provides VHDL and Verilog models	Windows 3.1, 95, & NT
	Warp2sim	All the functions of Warp2 plus timing simulation provided with Viewlogic's Viewsim simulator	Windows 3.1, 95, & NT
	Warp3	All the capabilities of Warp2sim with the addition of schematic capture, mixed-mode design entry, and Speedwave source-level debugging	Windows 3.1, 95, & NT
Lattice Semiconductor isp@latticesemi.com www.latticesemi.com	ispDS+ HDL Synthesis-Optimized Logic Fitter	Interfaces with all leading CAE vender tools. Includes GUI, explore tool, pin editor, static timing analyzer, ispDownload	Windows 3.1, 95, & NT
	ispVHDL Viewlogic System	Includes Viewlogic VHDL synthesis, capture, timing simulator, and ispDS+Fitter, Plus GUI, explore tool, pin editor, static timing analyzer, ispDownload	Windows 3.1, 95, & NT
	ISP Synario System	Includes ABEL-HDL, capture, functional simulator, and isp+Fitter, GUI, explore tool, pin editor, static timing analyzer,ispDownload	Windows 3.1, 95, & NT
	ispHDL Advanced System for Synplicity	Includes Synplify VHDL and Verilog synthesis and ispDS+Fitter, GUI, explore tool, pin editor, static timing analyzer, ispDownload	Windows 3.1, 95, & NT
	ispDS	Includes Lattice-HDL, functional simulator, and ispDS, GUI, interactive logic editor, ispDownload	Windows 3.1, 95, & NT
Exemplar Logic www.exemplar.com	Galileo Extreme	Provides an environment for FPGA and CPLD design. Includes Logic Explorer for bilingual HDL synthesis and architecture-specific area and timing optimization	Windows 95 & NT
Minc info@minc.com www.minc.com	PLDesigner	Design environment for PLD/CPLD design; offers EDIF and HDL entry as well as automatic device selection and partitioning, full placement and routing, and timing-model generation	Windows , 95 & NT
Morphologic info@morphologic.com www.morphologic.com	Evolution	Multichip, multivender tool for implementing DSP algorithms in FPGA-based systems. Includes graphical entry tool, and NT simulator, multichip floorplanner, and rules-based requirements monitor	Windiws 95
Protel Technology salesusa@protel.com www.protel.com	Advanced PLD	Universal programmable logic design for Protel's EDA/Client platform. Frequent library updates available from Protel's Web site	Windows 3.1, 95, & NT
Quicklogic info@quicklogic.com www.quicklogic.com	Quickworks	Integrated FPGA development environment with schematic capture, text entry, VHDL & Verilog synthesis, place & route, timing analysis, simulation, and programming for all Quicklogic FPGAs	Windows 3.1, 95, & NT
	Quicktools	Place & route, timing analysis, third-party CAE tool interfaces, and programming for all Quicklogic FPGA	Windows 3.1, 95, & NT
Synopsys designinfo@synopsys.com www.synopsys.com	FPGA Express	VHDL and Verilog HDL synthesizer for high-density programmable logic design. Architecture-specific synthesis with a grapgical design environment	Windows 95 & NT

Company	Product	Description	Windows Patform
Veribest info@veribest.com www.veribest.com	Veribest FPGA Synthesis	FPGA synthesis powered by Synopsys FPGA Express for VHDL/Verilog synthesis and optimization for FPGAs and CPLDs	Windows NT
	Veribest FPGA Design View	Integrates the FPGA/CPLD design flow for schemat ic, mixed schematic-HDL, and pure HDL flows,including programmable logic vendors's tools	Windows NT
	Veribest PLD	Provides logic synthesis/optimization for PLD design. Includes automatic partitioning and HDL simulation model generation	Windows NT
Viewlogic Systems viewdirect@viewlogic.com www.viewlogic.com	Intelliflow	Design manager. Recognizes the types of source files and technology to automatically configure flows to step the designer through systhesis, simulation, place & route, and analysis. Supports both schematic and HDL (VHDL and Verilog) design methodologies	Windows 95 & NT
	ViewPLD	Allows easy PLD design specification using ABEL with hierarchy support. Optimizes designs with automatic device fitting	Windows 95 & NT
Xilinx fpga@xilinx.com www.xilinx.com	Alliance Series Standard	FPGA/CPLD design solutions leveraging "Open Systems" integration with EDA partners	Windows 95 & NT
	Alliance Series Base, Foundation Series	FPGA/CPLD design solutions leveraging "Open Systems" integration with EDA partners. Provides design entry, simulation, synthesis,and device implementation in a tightly integrated design environment for a broad range of IC densities	Windows 95 & NT

Windows EDA : HDL Simulation

Company	Product	Description	Windows Platforms
Accolade Design Automation sales@acc-eda.com www.acc-eda.com	Peak VHDL Pro	Includes 32-bit direct-compiled VHDL simulator with waveform viewer and source-level debugger, built-in source code editor, hierarchy browser, and VHDL Wizards	Windows 95 & NT
	Peak VHDL Pro+VITAL	Includes all features of Peak VHDL Pro adds support for IEEE107.4 (VITAL) as well as advanced VHDL editing features	Windows 95 & NT
Aldec info@aldec.com www.aldec.com	Active - VHDL	VHDL simulation and design entry (VITAL/IEEE1076-93)	Windows 95 & NT
Avanti info@avanticorp.com www.avanticorp.com	Polaris-Int (interpreted)& Polaris-Com (compiledcode)	Verilog simulator that offers IEEE 1364 compliance, full SDF back annotation, and PLI integration	Windows 95 & NT
Cadence Design Systems cadenceconnect@cadence.com www.cadence.com	Verilog-XL	ASIC sign-off simulator	Windows NT
	Verilog -XL Turbo NT	ASIC sign-off simulator	Windows NT
Mentor Graphics www.mentorg.com	Quick HDL Lite	HDL simulation environment with such features as direct-compiled code architecture, user interface, and debugging	Windows 3.1, 95, & NT
Model Technology sales@model.com www.model.com	Modelsim EE & ModelsimPE	Full-featured Verilog, VHDL, or mixed-HDL simulation, optimized direct compilation, source-level debugging, hazard race detection, toggle testing,stability checks, buds contention, IP protection, & VCD I/O	Sun OS, Solaris, HP-UX, AIX; Windows 3.1, 95, & NT
MyCAD sales@mycad.com www.mycad.com	MyVHDL Station	VHDL simulator tool that contains source level debugger, waveform editor, and displayer. Accepts standard IEEE 1076-1987. Provides many execution and display windows	Windows 95 & NT
Quickturn Design Systems info@quickturn.com www.quickturn.com	Speedsim	Cycle-based logic simulator. Supports multiple clocks, asynchronous loops, transparent latches. Compiles 1 million gates in 10 minutes into 10 Mbyte run time image	Windows NT
Simucad silos@simucad.com www.simucad.com	Silos III	Performs logic and fault simulation for ASIC and FPGA designs using Verilog HDL at the behavioral, gate, and swith level	Windows 95 & NT
Veribest info@veribest.com www.veribest.com	Veribest VHDL Simulator	VHDL development environment that supports VHDL'87 and VHDL'93 for ASIC, FPGA, PLD, PCB, and system simulation	Windows NT
	Veribest Verilog Simulator	Verilog-XL clone simulator for ASIC, FPGA, PLD, PCB, and system simulation. Includes a comprehensive simulation and analysis environment	Windows NT
	Veribest Wave Bench	Stimulus creation environment. Stimulus is created by drawing waveforms graphically or from a text format. Generates VHDL or Verilog	Windows NT
	Veribest Analog	Analog simulation environment for IC, board, and system simulation. Behavioral modeling options, Spice compatability, tight integration with design entry tools	Windows NT
Viewlogic Systems viewdirect@viewlogic.com www.viewlogic.com	Speedwave for Windows	VHDL simulator with VITAL support. Includes hierarchical design browser, source-level debugger, and behavioral, gate-level, and RTL waveform, viewer. Support for Smartmodels and swift models	Windows 95 & NT
	VCS for windows	Verilog simulator. Includes hierarchical design browser, source-level debugger,and weaveform viewer. Complete PLI support	Windows 95 & NT
	Fusion for Windows	Single cosimulation environment for mixed designs containing gates, VHDL, Verilog, and Spice models. Includes a common source-level debugger, user interface, and waveform viewer	Windows 95 & NT

รายชื่อผู้ผลิตอุปกรณ์ FPGA

Company	Product family	Architecture	Maximum circuit density
Actel www.actel.com	MX	Interconnect Architecture Block	30,000 gates + 3 kbits memory
Altera www.altera.com	Flex 10KA/B	LUT-base architecture with embedded memory and Fasttrack Interconnect	250,000 gates + 40 kbits memory
	Flex 10K	LUT-base architecture with embedded memory and Fasttrack Interconnect	100,000 gates + 24 kbits memory
	Flex 6000/8000	LUT-base architecture with Fasttrack Interconnect	16,000-24,000 gates
	Max 9000A	Product term-based architecture with Fasttract Interconnect	12,000 gates
	Max 7000/7000S	Product term-based architecture with programmable interconnect array	5,000 gates
Atmel pld@atmel.com www.atmel.com	V2500B	Macrocells, product terms, interconnect	2,500 gates
	F1500 series	Macrocells, product terms, interconnect	1,500 – 4,000 gates
Chip Express moreinfo@chipx.com www.chipexpress.com	QYH500 LPGA	Gate array	75,000 gates
	CX2000 LPGA	Gate array	160,000 gates +128 kbits memory
Cypress Semiconductor jxl@cypress.com www.cypress.com	Flash370i	Programmable I/O blocks, logic blocks, interconnect	3,000 gates
	Ultra37000	Programmable I/O blocks, logic blocks, interconnect	10,000 gates
	Max340	Programmable I/O blocks, logic blocks, interconnect	3,000 gates
Gatefield gfinfo@gatefield.com www.gatefield.com	GF250F	Fine-grained, gate array-like sea of tiles optimized for synthesis	100,000 gates
	GF260F	Fine-grained, gate array-like sea of tiles with configurable embedded memory blocks, optimized for synthesis	92,000 gates + 23 kbits memory
Lucent Technologies orcafpga@aloft.lucent.com www.lucent.com/micro/fpga	ATT3000 series	Symmetrical cell array	4,500 gates
	OR2CxxA series	Configurable-grain logic array	99,400 gates
	OR2TxxA series	Configurable-grain logic array	99,400 gates
	OR3Cxx series	Configurable-grain logic array	116,000 gates
	OR3Txxx series	Configurable-grain logic array	186,000 gates
Motorola Semiconductor Products Sector sps.motorola.com/fpga	MPA1000	SRAM -based, fine-grain architecture using hierarchical routing	22,000 gates
Philips Semiconductors coolpld@abq.sc.philips.com www.coolpld.com	GAL- type devies	V architecture, combinational, and D register I/O	600 gates
	Programmable logic arrays	Programmable AND and programmable OR array	700 gates
	Sequencers	Programmable AND and programmable OR array, buried and output registers (one part with asynchronous and synchronous clocks)	700 gates
Quicklogic info@quicklogic.com www.quicklogic.com	pASIC 3	Variable-grained logic blocks optimized for Verilog/VHDL synthesis, programmable I/O cell with input registers and JTAG boundary scan	25,000 gates + 5.76 kbits memory
Xilinx publicrelations@xilinx.com www.xilinx.com	Xilinx XC4000E FPGA	Programmable logic blocks, I/O blocks, interconnect, plus user RAM, wide decoders, boundary-scan logic, fast-carry logic, 8 global nets, asynchronous/synchronous, dual-port RAM option	25,000 gates + 32 kbits memory

	Xilinx XC4000XL/XV FPGA	Programmable logic blocks, I/O blocks, interconnect, plus user RAM, wide decoders, boundary-scan logic, fast-carry logic, 8 global nets, asynchronous/synchronous, dual-port RAM option	125,000 gates + 148 kbits RAM
	Xilinx XC6200 Reconfigurable Co-processor FPGA	SRAM-based FPGA, programmable logic blocks, I/O blocks and interconnect. Built-in processor interface. Optimized for structured logic, datapath, reconfigurable coprocessing applications	100,000 gates + 256 kbits memory
	Xilinx XC9500 CPLD	Advanced 36V8 function blocks interconnected by fully populated switch matrix	6,400 gates