ใบรับรองวิทยานิพนธ์

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์

วิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมคอมพิวเตอร์)

ปริญญา

วิศวกรรมคอมพิวเตอร์ | วิศวกรรมคอมพิวเตอร์

สาขา | **ภาควิชา**

เรื่อง การเพิ่มประสิทธิภาพในการทดสอบวงจรด้วยการประมวลผลเชิงขนาน

A Performance Improvement of a Circuit Fault Simulation Using Parallel Processing

นามผู้วิจัย นายสิทธิพร ประภาวัต

ได้พิจารณาเห็นชอบโดย

ประธานกรรมการ .....

(รองศาสตราจารย์ประดนเดช นีละคุปต์, M.Eng.)

กรรมการ .....

(ผู้ช่วยศาสตราจารย์พีรวัฒน์ วัฒนพงศ์, Ph.D.)

กรรมการ .....

(ผู้ช่วยศาสตราจารย์ภูชงค์ อุทโยภาศ, Ph.D.)

หัวหน้าภาควิชา .....

(รองศาสตราจารย์อนันต์ ผลเพิ่ม, Ph.D.)

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์รับรองแล้ว

.....

(รองศาสตราจารย์กัญจนา ธีระกุล, D.Agr.)

คณบดีบัณฑิตวิทยาลัย

วันที่ ..... **เดือน** ..... **พ.ศ.** .....

วิทยานิพนธ์

เรื่อง

การเพิ่มประสิทธิภาพในการทดสอบวงจรด้วยการประมวลผลเชิงขนาน

A Performance Improvement of a Circuit Fault Simulation Using Parallel Processing

โดย

นายสิทธิพร ประภาวัต

เสนอ

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์

เพื่อความสมบูรณ์แห่งปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมคอมพิวเตอร์)

พ.ศ. 2557

สิทธิพร ประภาวัต 2557: การเพิ่มประสิทธิภาพในการทดสอบวงจรด้วยการประมวลผลเชิงขนาน ปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมคอมพิวเตอร์) สาขาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ ประธานกรรมการที่ปรึกษา: รองศาสตราจารย์ประดนเดช นีละคุปต์, M.Eng. 49 หน้า

ในงานวิจัยนี้ได้มีการนำกระบวนการประมวลผลเชิงขนานมาใช้ในการหาจุดบกพร่องของวงจรแบบ Stuck-at Fault สำหรับวงจรมาตรฐาน ISCAS85 และ ISCAS89 ได้มีการปรับปรุงโปรแกรม HOPE ให้สามารถจำลองให้สามารถทดสอบหาจุดบกพร่องสำหรับวงจรทั้ง 2 แบบได้

จากผลการจำลองโดยโปรแกรมที่ได้พัฒนาพบว่า จากการทำประมวลผลแบบขนานโดยใช้หน่วยประมวลผลจำนวน 2 CPU และ 4 CPU ค่า Speed up อยู่ระหว่าง 1.5 - 3 และจะมี Speed up โดยรวมประมาณ 4 สำหรับ 8 และ 16 CPU ตามลำดับ จากการจำลองพบว่าขนาดของปัญหาวงจรที่มีค่าเท่ากับ จำนวนผลคูณของ Test Vector และ Output ของวงจร และขนาดของ Fault List มีผลกระทบต่อความเร็วในการประมวลผล โดยถ้าขนาดของปัญหา มีค่าเกิน 10000 ขึ้นไปจะให้ Speed up ที่ดี และเมื่อขนาดของปัญหาใกล้เคียงกันขนาด ของ Fault List จะมีผลทำให้เวลาที่ใช้แตกต่างกันโดย Fault List ที่มีขนาดใหญ่จะใช้เวลาที่มากกว่าด้วย

_____ _____ ___ / ___ / ___

ลายมือชื่อนิสิต ลายมือชื่อประธานกรรมการ

Sittiporn Prabhavat 2014: A Performance Improvement of a Circuit Fault Simulation Using Parallel Processing. Master of Engineering (Computer Engineering), Major Field: Computer Engineering, Department of Computer Engineering. Thesis Advisor: Associate Professor Pradondet Nilagupta, M.Eng. 49 pages

We use parallel processing technique for finding Stuck-at fault in the ISCAS85 benchmark for combinational circuit and ISCAS89 benchmark for sequential circuit. We modify HOPE Program for simulate both of ISCAS85 and ISCAS89 in fault simulation.

The result of a simulation using 2 and 4 processors gives a speed up between 1.5 - 3 and about 4 using 8 and 16 processors. Circuit Problem Size and Fault List Size are two factors for using Parallel Computing. We find that the number of a test vectors, the number of outputs and the number of a fault list affect the speed up of a computation. Speed up is good when circuit problem size is over 10000. It means that a large circuit problem size has more speed up and efficiency than a small circuit problem size. We compare the two circuits that have a similar circuit problem size. We find that a large number of a fault list spends more time than a circuit which has a less number of a fault list.

_____ _____ ___ / ___ / ___

Student's signature Thesis Advisor's signature

กิตติกรรมประกาศ

ข้าพเจ้าขอขอบพระคุณ รองศาสตราจารย์ ประดนเดช นีละคุปต์ ประธานกรรมการที่ปรึกษาวิทยานิพนธ์ ผู้ช่วยศาสตราจารย์ ดร.พีรวัฒน์ วัฒนพงศ์ กรรมการที่ปรึกษาวิชาเอก และผู้ช่วยศาสตราจารย์ ดร. ภุชงค์ อุทโยภาศ กรรมการที่ปรึกษาวิชารอง และผู้แทนบัณฑิต ผู้ช่วยศาสตราจารย์ ดร. ดุลย์พิเชษฐ์ ฤกษ์ปรีดาพงศ์ จากภาควิชาวิศวกรรมไฟฟ้า ในการให้คำปรึกษาในการเรียนรู้ การค้นคว้าวิจัย ตลอดจนการตรวจแก้ไขวิทยานิพนธ์จนสมบูรณ์

ขอขอบพระคุณ คุณทีปกร ศิริวรรณ สำหรับคำแนะนำในเรื่องการทดสอบวงจรและการสร้าง โปรแกรมทดสอบวงจร และคุณสงกรานต์ ศิริเดชาชัย สำหรับคำแนะนำเกี่ยวกับงานวิจัยด้านการประมวลผลเชิงขนาน ด้วยครับ

สุดท้ายขอกราบขอบพระคุณ คุณพ่อ คุณแม่ พี่ และญาติพี่น้องคนอื่นที่คอยให้กำลังใจและสนับสนุนให้วิทยานิพนธ์ฉบับนี้เสร็จสมบูรณ์ครับ

สิทธิพร ประภาวัต

กรกฎาคม 2557

สารบัญ

หน้า

สารบัญตาราง

สารบัญภาพ

สารบัญภาพ (ต่อ)

การเพิ่มประสิทธิภาพในการทดสอบวงจรด้วยการประมวลผลเชิงขนาน

A Performance Improvement of a Circuit Fault Simulation Using Parallel Processing

คำนำ

ในเรื่องของวงจรในปัจจุบัน เมื่อวิวัฒนาการในเรื่องของ VLSI มีมากขึ้น (Breuer and Friedman,1977). ส่งผลให้ขนาดสเกล ของวงจรมีขนาดใหญ่ขึ้นทำให้เราสามารถใส่องค์ประกอบอันได้แก่ ทรานซิสเตอร์(Transistor) ได้มากขึ้นทำให้เราสามารถออกแบบควบคุมวงจรได้มากขึ้น แต่ก็ทำให้กระบวนการทดสอบวงจรเพื่อหาข้อผิดพลาดนั้นทำได้ยาก และใช้เวลามากขึ้นเช่นเดียวกัน ดังนั้นจึงได้มีแนวคิดที่จะนำการประมวลผลเชิงขนานมาใช้ในการทดสอบวงจรนี้ด้วย การทดสอบวงจรดิจิทัล เป็นงานที่มีความสำคัญมากในการผลิตวงจรดิจิทัล จุดประสงค์เพื่อรับรองความถูกต้องในการทำงานของวงจรดิจิทัล และตรงกับงานที่ได้การออกแบบ จึงต้องมีการวิจัยเพื่อให้ได้เครื่องมือในการสร้างชุดทดสอบวงจร (Circuit Test Vector Generation) และใช้เครื่องมือเพื่อทดสอบวงจรให้มีความครอบคลุมข้อผิดพลาดสูงสุด (Highest Fault Coverage) และมีกรรมวิธีในการสร้างชุดทดสอบวงจรว่า Automatic Test Pattern Generation (ATPG) เราจะใช้กรรมวิธี ATPG ในการสร้างชุดทดสอบและนำไปใช้กับเครื่องมือทดสอบเพื่อหาจุดแตกต่างระหว่างวงจรที่มีการทำงานถูกต้องกับวงจรที่มีความผิดพลาด ความแตกต่างก็คือความผิดพลาดของวงจรนั่นเอง

ความผิดพลาดของวงจรที่ได้รับความสนใจในการหากรรมวิธีสร้าง ATPG นั้น เริ่มแรกจะมุ่งเน้นไปที่ความผิดพลาดเป็นความผิดพลาดในการผลิตที่ทำให้เส้นทางการเดินของวงจรมีค่าทางตรรกะ(Logic) แบบถาวรหรือ Stuck-at Model เป็นหลัก แต่เมื่อวิวัฒนาการในเรื่องของ VLSI มีมากขึ้น ทำให้ความเร็วของวงจรมีผลต่อการทำงานของวงจรด้วย เราจะพบว่ามีการทำงานผิดพลาดของวงจรในช่วงระยะเวลาหนึ่งเกิดขึ้น สาเหตุเกิดจากการควบคุมดีเลย์ (Delay) ของวงจรที่ไม่เหมาะสม และเป็นความผิดพลาดจาก Delay ในวงจร (Delay Fault Model) แต่การหาจุดบกพร่อง (Fault) ที่เป็นแบบ Delay Fault Model ยังซับซ้อนเพราะมีการสร้างแบบจำลองเวลาและขนาดของปัญหายังมีขนาดใหญ่มาก ดังนั้นวิทยานิพนธ์นี้จึงมุ่งไปที่การเกิด Stuck-at model ที่เกิดขึ้นทั้ง Combinational Circuit และ Sequential Circuit

งานวิจัยหลายชิ้นที่ผ่านมาได้ศึกษาในเรื่องของการหากระบวนการทำ fault simulation ด้วยวิธีต่างๆเช่น CRIS(Saab *et al.*, 1992) , FSIM (Lee and Ha,1991) และ HOPE(Lee and Ha,1992) ที่จะทำให้ค่า ความสามารถในการการตรวจจับข้อผิดพลาดของวงจร (Fault Coverage) ของ Tools ที่นำมา ทดสอบวงจรมีค่าดีที่สุด สามารถทดสอบ Fault ที่อาจจะเกิดขึ้นได้ในวงจรได้มากที่สุด นอกจากนั้นแนวทางวิจัยอีกด้านคือการสร้างชุดทดสอบ (Test Vector) ได้เร็วที่สุด และใช้จำนวน Test Vector น้อยที่สุดเช่น HPGAST(Siriwan , 2001.) ,DIGATE(Hsiao *et al.*, 1996) และGATEST(Rudnick *et al.*, 1994); (Rudnick *et al*., 1997) เป็นต้น

สำหรับงานวิจัยเชิงคณิตศาสตร์ และวิทยาศาสตร์ในปัจจุบัน เช่นงานวิจัยทางด้านอวกาศตัวอย่างเช่นการควบคุมการเคลื่อนที่ของยาน Path Finder ,ด้าน bioinformatics ตัวอย่างเช่นการสังเคราะห์โปรตีน งานวิจัย DNA เป็นต้น ซึ่งเป็นงานที่ต้องใช้คอมพิวเตอร์ในการคำนวณและประมวลผลที่มีความเร็วสูง สิ่งที่พบก็คือค่าใช้จ่ายที่สูงในการซื้อคอมพิวเตอร์สมรรถนะสูงเพื่อการประมวลผล และส่งผลไปถึงการ บำรุงรักษา ที่ต้องใช้ค่าใช้จ่ายที่สูงตามด้วยทำให้ต้องมีการประยุกต์ใช้งานคอมพิวเตอร์ที่มีสมรรถนะไม่สูงมากนัก จำนวนมากกว่า 1 เครื่องมาช่วยกันในการประมวลผลแบบขนาน ผลที่ได้คือประสิทธิภาพที่ใกล้เคียงกับเครื่องที่มีสมรรถนะสูงแต่ใช้ค่าใช้จ่ายในการซื้อเครื่องที่น้อยกว่า และค่าใช้จ่ายในการ บำรุงรักษา ด้วย

ในวิทยานิพนธ์นี้จะแสดง การหาจุดผิดพลาดของวงจร Combinational Circuit และ Sequential Circuit โดยนำ Test Vector ที่ได้จาก HPGAST Sequential ATPG และ Combinational ATPG มาทำการทดสอบหา จุดบกพร่องของวงจรว่าเป็น Fault ที่เกิดขึ้นที่จุดใดในวงจร และมีการนำการประมวลผลแบบขนานมาประยุกต์ใช้งานในครั้งนี้ โดยวงจรนั้นเป็นวงจรตามมาตรฐาน ISCAS85 และ ISCAS89

วัตถุประสงค์

1. เพื่อใช้การประมวลผลแบบขนานมาใช้ในการตรวจสอบหาข้อผิดพลาดของวงจร
2. พัฒนากรรมวิธีโดยใช้ทดสอบกับวงจรมาตรฐาน ISCAS85 และ ISCAS89
3. เพื่อประโยชน์ในการตรวจสอบวงจรที่เร็วยิ่งขึ้น ทำให้แก้ไขกระบวนการผลิตได้ทัน

การตรวจเอกสาร

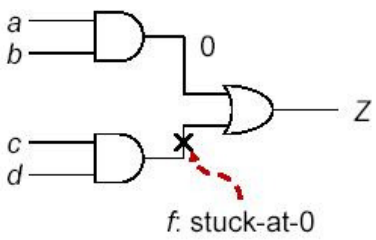
Fault Model

การกำหนดประเภท Fault Model มีประโยชน์เพื่อให้เราสามารถระบุเป้าหมายในการทดสอบวงจร และความเป็นไปได้ในการวิเคราะห์และสามารถวัดประสิทธิภาพได้ในการทดลอง และประเภทของ Fault Model สามารถแยกตามลักษณะการเกิด Fault ได้ 2 ประเภทคือ

1. Fault ที่เกิดขึ้นจากความผิดพลาดทางด้านกายภาพ เช่นเกิดการลัดวงจร หรือ การเกิดวงจรเปิดเป็นต้น สำหรับวงจร ดิจิทัลสามารถจำแนกได้ย่อยๆ 3 ประเภทคือ

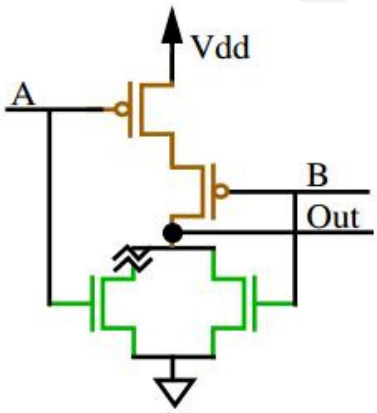
1.1 Stuck –at Fault (Patel, 1998) เป็น Fault ที่เกิดขึ้นจากความผิดพลาดของการต่อสายในวงจร ทำให้ Logic ที่ได้ไม่ตรงกับผลที่เราต้องการ มักจะเกิดขึ้นจากการลัดวงจรของแหล่งจ่ายไฟ (V_{cc}) และ Ground (Gnd) กับ Input ของวงจรและทำให้ แสดงค่า Logic แบบถาวรไม่ว่า สัญญาณขาเข้าจะเป็นอะไรก็ตาม เมื่อพิจารณาข้างในวงจรดิจิทัลแล้วจะพบว่าวงจรดิจิทัลพื้นฐานจะประกอบด้วย Gate ที่ทำหน้าที่ในการกำหนด Logic ในวงจรให้มี Logic หรือ function การทำงานตรงตาม Gate และ Path ที่เป็นเส้นทางการเดินของสัญญาน

การทดสอบวงจรในยุคแรกๆ ที่ความเร็วยังไม่มีผลต่อการออกแบบวงจรดิจิทัลนั้นจะเป็นการทดสอบที่เกิดจากความผิดพลาดของ Logic แบบถาวรอันเกิดจากกระบวนการผลิตหรือเรียกว่า stuck-at fault จากภาพที่ 1 ได้เกิด stuck-at 0 จะทำให้การทำงานของ Z ที่มีการทำงานคือ Z = ab+cd เมื่อ เมื่อกำหนดค่า abcd = 0011 ค่า Z ควรเป็น 1 แต่ผลของการเกิด stuck-at 0 จะทำให้ Z มีค่า 0



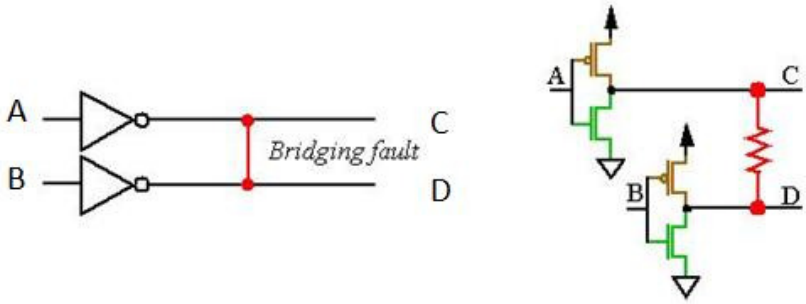
ภาพที่ 1 แสดงการเกิด stuck-at fault ที่ 0 ของวงจร Z = ab+cd

1.2 Stuck –Open และ Stuck-short Fault เป็น Fault ที่เกิดจากเส้นเชื่อมในวงจรขาด (Line Break) เกิด Open Circuit สำหรับ stuck-open (Li *et al*., 2001) และวงจรลัด ที่เป็นแบบ stuck-short ทำให้ผลลัพธ์ของวงจรไม่ตรงกับความต้องการเกิดขึ้นที่ระดับ Transistor ทำให้บางงานวิจัยเรียกว่า Fault ประเภทนี้ว่า Transistor Fault ตัวอย่างแสดงดังภาพที่ 2



ภาพที่ 2 แสดงการเกิด Stuck-Open ที่จุด A

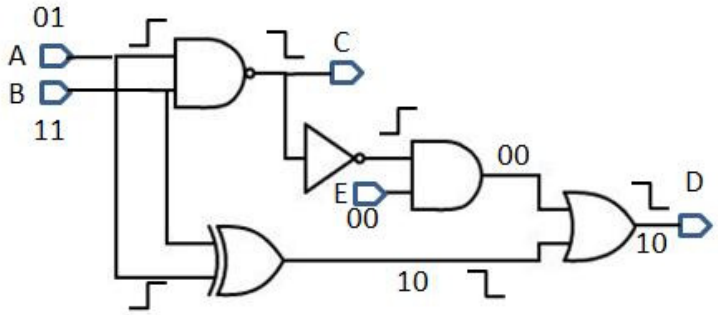
1.3 Bridging Fault (Vijay and Walker, 1999) เป็น Fault ที่เกิดขึ้นจากการลัดวงจรประเภทหนึ่งเป็นการลัดวงจรระหว่างจุด 2 จุดในวงจรทั้งการลัดวงจรภายใน Element ของ Transistor (Source, Gate และ Drain) หรือ ระหว่าง Input กับ Input และ Input กับ Output) ของ องค์ประกอบ ของ Transistor อีกตัวหนึ่ง ซึ่งจะแตกต่างกับ Stuck-at Fault ตรงที่ไม่มีแหล่งจ่ายไฟหรือ Ground มาเกี่ยวข้องดังภาพที่ 3 ที่เกิด Bridging Fault ที่จุด Output C และ D



ภาพที่ 3 แสดง Bridging Fault ที่จุด Output C และ D

2. Delay Fault (Krstic and Cheng, 1998);(Pramanick and Reddy, 1998) เป็น Delay ที่เกิดขึ้นจาก Propagation time เกิดขึ้นจากอุณหภูมิในระหว่างการผลิตและการออกแบบที่มีการองค์ประกอบในวงจรเช่น ความต้านทานไฟฟ้า,ตัวเก็บประจุ และตัวเหนี่ยวนำได้ไม่ดี ลักษณะบางประการของ Delay Fault ก็คือ จะทำให้เกิด Fault ทางกายภาพ ณ ช่วงขณะเวลานึงเกิดขึ้น แต่ไม่ได้เกิดขึ้นถาวรแบบ Stuck-at Fault หากเรามองที่สัญญาณเวลาจะแบ่งได้เป็น Fault แบบ Slow-to-Rise กับ Slow-to-Down ซึ่ง Delay Fault แบ่งได้อีก 3 ชนิด. ก็คือ Transition Delay Fault, Gate Delay Fault และ Path Delay Fault

2.1 Transition Fault Model Transition Fault Model จะพิจารณาที่รูปแบบเชิง Logic ของผลกระทบที่เกิดขึ้นจาก delay การเปลี่ยนแปลงค่าของตรรกะทั้งการเปลี่ยนแปลงขึ้นหรือลงที่ input และ output ของ Logic ใน gate Transition Fault Modelมี อยู่ 2 ประเภทคือ slow-to-rise และ slow-to-fall transition fault ที่เกิดขึ้นแบบ slow-to-rise ตรรกะจะแสดงพฤติกรรมแบบ stuck-at-fault ที่ 0 ส่วน transition fault ที่เกิดขึ้นแบบ slow-to-fall ตรรกะจะแสดงพฤติกรรมแบบ stuck-at-fault ที่ 1 งานวิจัยบางชิ้นจะเรียกรูปแบบนี้ว่า Gross-Delay fault Model ที่จะมีการสังเกตุผลของ Transition ว่ามีผลต่อ Path ที่จะไปยัง Output หรือไม่และมีผลอย่างไรตัวอย่างจากภาพที่ 4 เมื่อกำหนดให้เกิด Slow-to-rise ที่จุด A ค่าจะทำการ Propagate ไปถึง Output ที่จุด D

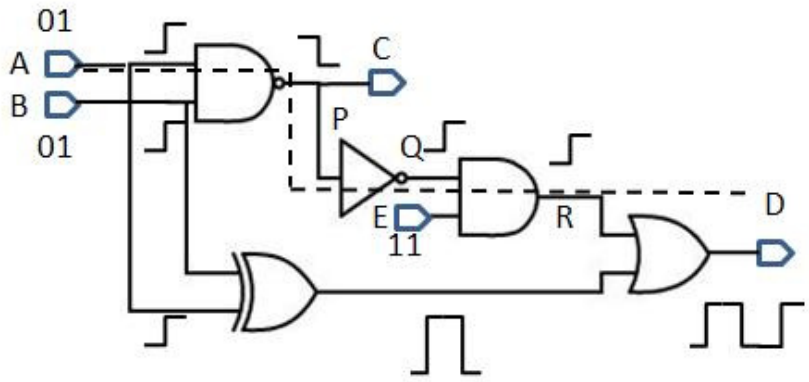


ภาพที่ 4 แสดง Transition fault ที่ Input A ส่งผลต่อ Output ที่ Output D

2.2. Gate delay fault model รูปแบบของ gate delay fault จะมองสถานะการเปลี่ยนแปลงขึ้นหรือลงที่จาก gate input และ Output เวลา จะมีผลต่อจำนวน fault ที่เกิดขึ้นด้วย ดังนั้น gate delay fault ที่ 5 nanosecond จะไม่เท่ากับ gate delay fault ที่ 10 nanosecond Gate Delay Fault จะเหมือน Transition Fault Model ในภาพที่ 4 แต่จะพิจารณาในตัว Gate และ Input

Output ของ gate เท่านั้น มีอยู่ 2 แบบคือ slow-to-rise และ slow-to-fall transition fault ที่เกิดขึ้นแบบ slow-to-rise ตรรกะที่ของสัญญานที่ออกจาก gate output จะแสดงพฤติกรรมแบบ stuck-at-fault ที่ 0 ชั่วคราว ทั้งที่สัญญานที่ถูกต้องควรเป็น 1 ส่วน transition fault ที่เกิดขึ้นแบบ slow-to-fall ตรรกะจะแสดงพฤติกรรมแบบ stuck-at-fault ที่ 1 ชั่วคราว ทั้งที่สัญญานที่ถูกต้องควรเป็น 0

2.3. Path Delay Fault Model Path Delay Fault Model ได้รับความสนใจมากกว่า gate delay fault และ transition fault model ในการวิจัยทั้งเรื่องของการสร้างชุดทดสอบ (test pattern generation) และ fault simulation ในการหา path delay fault Path ใดๆในวงจรที่มี ค่าdelay มากกว่า clock interval ของระบบจะเกิด path delay fault ขึ้น path delay fault ที่เกิดขึ้นจะส่งผลต่อ Path ทั้งหมดของวงจร สำหรับทุก path ที่เกิดขึ้นในวงจรที่เชื่อมต่อระหว่าง input หลัก และ output หลักก็จะมี path delay fault อยู่สองแบบคือ rising path ก็คือ path ที่เกิดขึ้นมีการเปลี่ยนแปลงค่าจาก 0 ไป 1 ตั้งแต่ต้นทางก็คือ input หลักของวงจร, path ก่อนเข้า gate, path เมื่อออกจาก gate และ falling path ก็คือ path ที่เกิดขึ้นมีการเปลี่ยนแปลงค่าจาก 1 ไป 0 ตั้งแต่ต้นทางก็คือ input หลักของวงจร, path ก่อนเข้า gate, path เมื่อออกจาก gate จากภาพที่ 5 เมื่อเราจะมอง Fault เป็น Path ทั้งเส้น เราจะพบว่าค่า Logic ของ D ไม่ได้เกิดขึ้นจาก Path A,P,Q,R และ D อย่างเดียวแต่เกิดจาก Path ที่มีจาก Gate Exclusive-Or ด้วย



ภาพที่ 5 แสดง Path Delay Fault ของ Path A,P,Q,R และ D

การทดสอบวงจร Combinational Logic Circuit

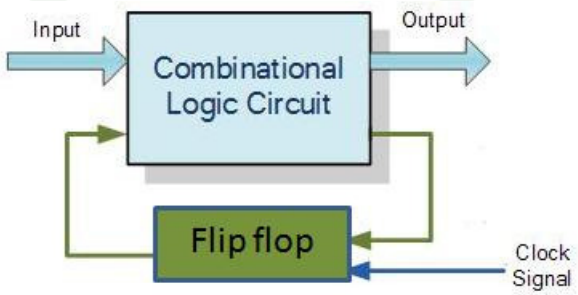
วงจร Combinational Logic Circuit เป็น วงจรที่ประกอบขึ้นด้วยลอจิกเกตต่าง ๆ การสร้างวงจรก็คือ การนำเอาเกตต่าง ๆ มาต่อกันเป็นวงจรเพื่อให้วงจรสามารถทำงานได้ตามที่เราต้องการ การทำงานจะขึ้นอยู่กับคุณสมบัติของเกตและสัญญาณอินพุทที่ป้อนเข้า โดยแสดงออกทางเอาท์พุตของวงจร ดังภาพที่ 6

ภาพที่ 6 แสดงลักษณะวงจร Combinational Circuit

ในการทดสอบวงจรดิจิทัลแบบ Combinational Circuit กรณีที่มี Input k ตัวจะใช้ จำนวน Test Vector จำนวน 2^k ตัว ตัวอย่างจาก วงจรมาตรฐาน ISCAS85 C2670 มี 233 Input จะต้องพิจารณา Test Vector ทั้งหมด 2^{233} ประมาณ $1.38*10^{70}$ test vector ซึ่งจะเป็นขนาดปัญหาที่ใหญ่มาก

การทดสอบวงจร Sequential Circuit

สำหรับวงจรดิจิทัล Sequential circuit จะมีองค์ประกอบ แบ่งได้ 2 ส่วน คือส่วนที่เป็น Combinational Circuit และส่วนที่เป็น Flip flop โดย ความสัมพันธ์ระหว่าง 2 ส่วนนี้ได้แจกแจงไว้ในภาพที่ 7



ภาพที่ 7 แสดงลักษณะวงจร Sequential Circuit

โดยทั่วไปในการทดสอบวงจร Sequential Circuit ที่มี Input ในส่วน Combinational k ตัว กับ m Flip Flop ดังภาพที่ 7(Cheng, 1996.) วงจร Sequential Circuit ดังกล่าวจะมีการพิจารณา 2^m State แต่ละ state จะtest vector มี 2^k ตัว ดังนั้นต้องพิจารณา 2^{k+m} ตัว ซึ่งจะเป็นขนาดปัญหาที่ใหญ่มาก ดังนั้นการลดจำนวน test vector จึงเป็นสิ่งที่จำเป็น และต้องนำมาใช้เพื่อลดขนาด test vector ลง ทำให้กระบวนการทำ Fault Simulation ใช้เวลาที่ลดลงได้ จึงได้มีการวิจัยสร้าง Automatic Test Pattern Generation (ATPG) และขอนำเสนอต่อไป

Automatic Test Pattern Generation (ATPG)

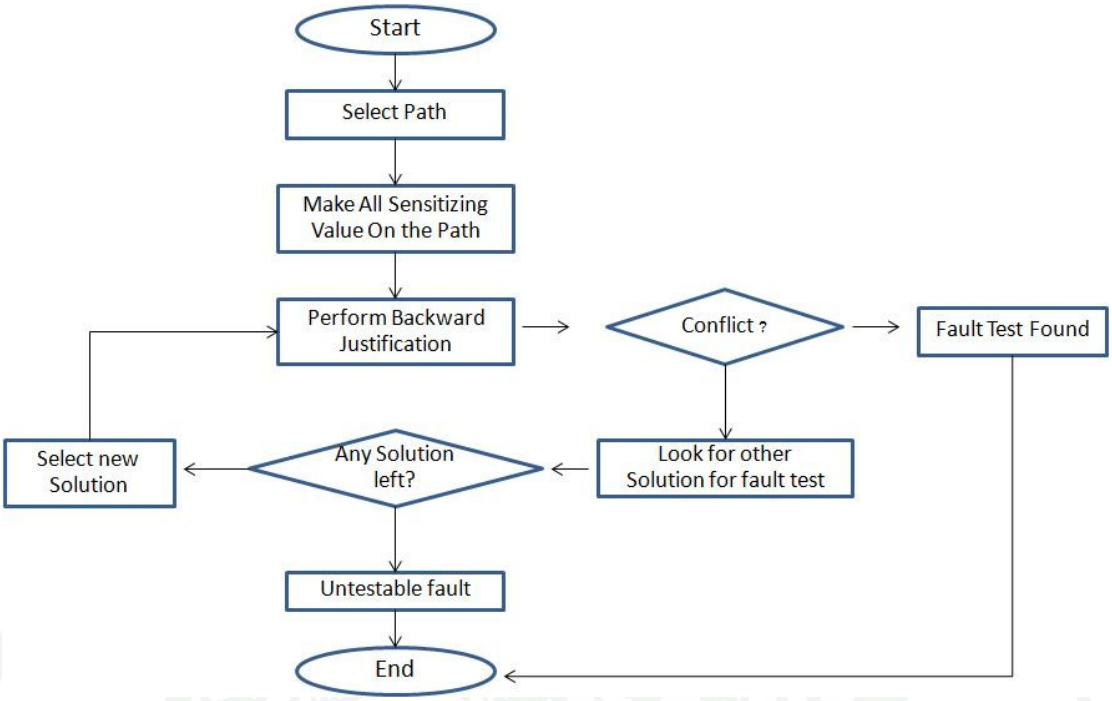
เป็นกระบวนการสำคัญในการออกแบบเพื่อทดสอบวงจร จุดประสงค์ใน ATPG คือการสร้าง Test vector เพื่อนำ Test vector ไปใช้ในกระบวนการ Fault Simulation เนื่องจาก หากไม่มีการสร้าง Test Vector จะต้องใช้เวลาในกระบวนการ Fault Simulation มาก โดย ATPG Tools ที่ดีควรจะตรวจสอบ Fault ได้มากที่สุดและมีขนาดที่ไม่ใหญ่มากเพื่อกระบวนการ Fault Simulation จะได้ใช้เวลาลดลงด้วย

ขั้นตอนสำคัญของ ATPG Tools คือ Path Sensitizing จากภาพที่ 8 โดยมีกระบวนการ 3 แบบคือ

1. Fault Activation เพื่อทำการกำหนดค่า Logic ที่จะเป็นจุดเกิด Fault ตาม ประเภทของ Fault ต่างๆ

2. ทำ Path sensitize คือกำหนดค่าที่จะเกิดของ Logic ตั้งแต่จุดเกิด Fault ไปยัง Output

3. ตรวจดูว่า path ที่ทำการ sensitize มีโอกาสเกิดค่าอื่นที่ไม่เป็นตามกำหนดไว้จาก path sensitize หรือไม่ ถ้าเกิดแสดงว่า path ที่เลือกไม่ถูกต้องต้องกลับไปทำ Path sensitize ใหม่

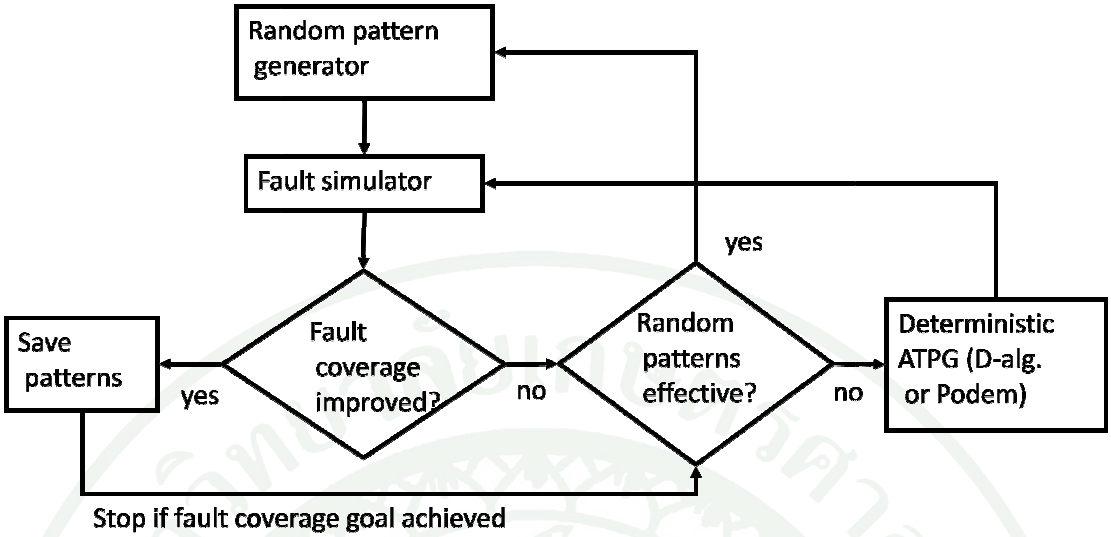


ภาพที่ 8 แสดงกระบวนการ Path Sensitizing

ในกระบวนการทำ ATPG Tools ยังมี กระบวนการลดจำนวน Test vector อยู่ในกระบวนการ ATPG โดยแม้จะลดจำนวน Test Vector ก็ไม่ทำให้ ค่า Fault Coverage ลดลงเรียกว่า Test Compaction. Test Compaction มี 2 แบบคือ แบบ Static Test Compaction และ Dynamic test Compaction (Pomeranz and Reddy ,1996)

Fault Simulation

Fault Simulation เป็นกระบวนการทดสอบวงจร เพื่อจำลองข้อบกพร่องของการทดสอบวงจร ดิจิทัล เป็นกระบวนการที่ใช้แสดงคุณภาพของการตรวจสอบวงจร และ คุณภาพของการผลิต เมื่อมีการผลิตจริง Fault Simulation จะเกิดขึ้นหลังจากการทำ Logic Simulation ที่เป็นกระบวนการออกแบบความถูกต้องของวงจรว่า ลักษณะการทำงานที่แท้จริงของวงจรที่ต้องการเป็นแบบใด กระบวนการ Logic Simulationจะรับ Test Vector เข้ามา และ ผลของ Logic Simulation จะเรียกว่า Fault-Free Circuit หรือ Good Circuit



ภาพที่ 9 แสดงความสัมพันธ์ในการสร้าง Test Vector และ Fault Simulation

กระบวนการที่เกิดขึ้นใน Fault Simulation มีตั้งแต่การรับค่า Test Vector ที่ได้จากกระบวนการ ทำ ATPG Tools ตามภาพที่ 9 เข้าเพื่อผ่านกระบวนการทำ Fault-Free Circuit ที่เกิดขึ้นจาก Logic Simulationจากนั้น เข้าสู่กระบวนการ Fault Simulation โดยดูเงื่อนไขต่างๆ ว่าเกิด Fault ตามแบบ Fault Model แบบใดบ้าง สิ่งที่ชี้วัดคุณภาพของ Fault Simulation คือจำนวน Fault ที่พบจากกระบวนการ Fault Simulation กับ จำนวน Fault ทั้งหมดที่เกิดขึ้นได้ เรียกว่า Fault Coverage

Fault Coverage สำหรับ Stuck-at fault

การสร้าง Fault Simulation สำหรับ Stuck-at Fault นั้น จะเป็นการระบุ fault ที่จะเกิดขึ้นโดยดูลักษณะความแตกต่างของ Output ที่เกิดจากการป้อน Input จาก Test Vector ที่กำหนดให้ ในแต่ละ Test Vector นั้นจะมี ค่า Output จากFault-free circuit จาก Logic Simulation และ Faulty circuit ใน Fault Simulation หาก ค่า Output จากทั้ง สองแบบมีค่าต่างกัน แสดงว่า Fault จะโดนตรวจพบ และFault ดังกล่าวก็เอาออกไปใน List จนกระทั่งจะเหลือแต่ Fault ที่ไม่สามารถตรวจพบได้ ถ้านำไปหักออกจากจำนวน Fault ที่พบทั้งหมดก็จะได้ Fault ที่สามารถตรวจสอบได้และเป็นการแสดงถึงคุณภาพของ Fault Simulation ที่เรียกว่า Fault Coverage โดยวัดได้จาก Fault ที่ตรวจสอบพบได้ในกระบวนการ Fault Simulation กับ Fault ทั้งหมดที่เกิดขึ้นได้ของวงจรทดสอบนั้นๆ Fault Coverage จึงเป็นตัวชี้วัดคุณภาพในการทดสอบวงจรตามสมการที่ 1 (Miczo, 2003) หาก Fault Coverage ที่ดีต้องมี Fault Coverage 100% คือ ตรวจสอบจุดบกพร่องได้ทุกตัว

$$FaultCoverage = \frac{TestableFaultInCircuit}{TestableFaultInCircuit + untestablefaultIncircuit} * 100 \quad (1)$$

TestableFaultInCircuit คือ จำนวน Fault ที่ทดสอบได้ในวงจร

untestablefaultIncircuit คือ จำนวน Fault ที่ทดสอบไม่ได้ได้ในวงจร

งานที่เกี่ยวข้อง

Fault Simulation

ในงานวิจัยด้าน Fault Simulation มีการพัฒนาการทดสอบวงจรในภาพรวมของ Combinational Circuit ว่ามีค่า Fault Coverage ในวงจรอย่างไร ตอนแรกยังไม่มีการพิจารณา Gate Logic Propagation แต่จะมีการทดสอบ Gate Logic Propagation ในภายหลัง และมีการทำงานแบบ Parallel Thread เป็นการหา Fault Coverage จึงได้ Tools ออกมาตัวนึงเรียกว่า FSIM (Lee and Ha ,1991) หลังจากนั้น เริ่มมีการวิจัยด้าน Fault Simulation สำหรับวงจร Sequential Circuit โดยใช้หลักการ Fault Logic Propagation และมีการทำ Fault Ordering แบบ Dynamic ทำให้มีการใช้ Memory ลดลง ด้วยหลักการนี้จึงได้ Tools ออกมาชื่อ PROOF (Rudnick *et al*., 1991.) หลังจากนั้น มี การพัฒนา Fault Simulation โดยเพิ่ม กระบวนการ Fault Injection และ ผสมผสาน Fault Ordering ทั้งแบบ Static และ Dynamic ได้ดีกว่า PROOF และสามารถหา Fault Coverage แบบ Parallel Thread ได้ออกมาเป็น Tool ที่มีชื่อว่า HOPE (Lee and Ha, 1992)แม้ว่างานวิจัยต่อมาจะมีการพัฒนา Fault Simulation ดวยวิธีการต่างๆ โดยวิธีการใช้ Genetic Algorithm (GA) เข้ามาแบบ Tool ที่ชื่อว่า CRIS (Saab *et al*., 1992) แต่ผลลัพธ์ด้าน Fault Coverage ของ CRIS ต่ำกว่า HOPE มาก และเมื่อมีการปรับปรุง Tool ให้มีค่า Fault Coverage เพิ่มขึ้นแต่ก็ยังใช้เวลามาก เมื่อเทียบกับเมื่อใช้งาน HOPE ทีมวิจัย CRIS จึงนำวิธีการของ CRIS ไปใช้ในการพัฒนา ด้าน ATPG แทน เพราะ เดิม CRIS มีส่วนของการสร้าง Test Vector อยู่ด้วย ดังนั้นจากที่ได้ศึกษางานวิจัยที่เกี่ยวข้องด้าน Fault Simulation Tools จึงนำใช้ HOPE มาใช้ในการทดสอบวงจร

ATPG Tools

ในงานวิจัยด้าน ATPG Tools นั้น เริ่มแรกนั้น สำหรับวงจรที่เป็น Combinational Circuit ได้มีการพัฒนา Tools โดยใช้ หลักการของ Path Sensitize จาก FAN และ PODEM Algorithm โดย Tools ที่สามารถทดสอบวงจรแบบ Combinational circuit คือ SOCRATES (Schulz *et al*., 1987) สำหรับงานวิจัยในการสร้าง Test Vector ของ Sequential Circuit เมื่อพิจารณาคุณสมบัติของ Sequential circuit จะพบว่าผลของ Output ที่ State ที่แล้วจะมีผลต่อ Input ใน state ปัจจุบันด้วย ดังนั้นลักษณะการทำ Simulate จะทำให้ผลของ Fault Coverage ทำได้ดีกว่า และกระบวนการ simulate โดย GA เป็นที่นิยมเป็นอย่างมาก โดยเริ่มแรกกระบวนการสร้าง Test Vector จะแบ่งการทำงานเป็นสองช่วงคือ ช่วงการสร้าง Test Vector ตั้งต้น มา 1 ตัวและช่วงการสร้าง Test vector ตัวถัดไปจาก Test Vector ตั้งต้น จะได้ ATPG Tools คือ GATEST (Rudnick *et al*., 1994); (Rudnick *et al*., 1997) ผลของ GATEST จะได้ Test Vector ที่มีขนาดเล็กแต่ตรวจพบข้อผิดพลาดได้น้อย ต่อมาได้มีอีกทีมวิจัยนึงได้เพิ่มส่วนของ Fault Activation และมีการจำแนกลำดับ Test Vector จาก Flip flop ไปยัง Output ด้วยทำให้ได้ ATPG Tools สำหรับสร้าง Test Vector ชื่อ DIGEST (Hsiao *et al*., 1996) ที่มีการตรวจสอบ Fault ได้ดีกว่า GATEST แต่มีขนาด Test Vector ใหญ่ตาม ขนาดวงจร และต่อมาได้มีการวิจัยด้าน ATPG Tools ที่ใช้หลักการระบุค่า Fitness Function สำหรับวัดผลของ Test Vector ได้ Tools ที่มีชื่อว่า HPGAST (Siriwan, 2001.) HPGAST มีการตรวจหา Fault ในวงจรได้ครอบคลุมมากกว่า GATEST และ DIGESTทำให้มีความน่าเชื่อถือในการนำไปใช้ในกระบวนการ Fault Simulation และ มีขนาด Test Vector ที่ไม่มาก เมื่อเทียบกับ GATEST และ DIGEST ดังนั้นจากที่ได้ศึกษางานวิจัยที่เกี่ยวข้องด้าน ATPG Tools จึงนำผล Test vector ของ HPGAST มาใช้ในการสร้างชุดทดสอบวงจร

การใช้วิธีประมวลผลแบบขนานในงานต่างๆ

การนำวิธีการประมวลผลเชิงขนานหมายถึงการใช้ ไมโครโปรเซสเซอร์อย่างน้อย 2 ตัวมาทำงานในกระบวนการทั้งหมด หลักการก็คือ Server จะแบ่งส่วนปัญหาปัญหาทั้งหมดเป็นส่วนๆ และส่งงานไปยัง ไมโครโปรเซสเซอร์ย่อยๆ และไมโครโปรเซสเซอร์แต่ละตัวจะประมวลผลตามงานที่มอบหมาย และสุดท้าย Server รับผลลัพธ์จาก ไมโครโปรเซสเซอร์ทุกตัว จะทำการสรุปผลลัพธ์ของปัญหาทั้งหมด ตัวอย่างงานที่ใช้ในการประมวลผลแบบขนานก็คือการจำลองการทำงานของชั้นบรรยากาศ วิเคราะห์ผลของการเคลื่อนตัวก้อนเมฆ ลม ความชื้น เพื่อการพยากรณ์อากาศ

(Garrido *et al*., 2007), การวิเคราะห์สมการโครงสร้างอะตอมในทฤษฎีอะตอมของโชร์ดิงเจอร์ในเรื่องกลศาสตร์ควอนตัม (Fijany *et al.*, 1994) และ กรรมวิธีการทำ Matrix Multiplication (Li *et al*., 1998.)

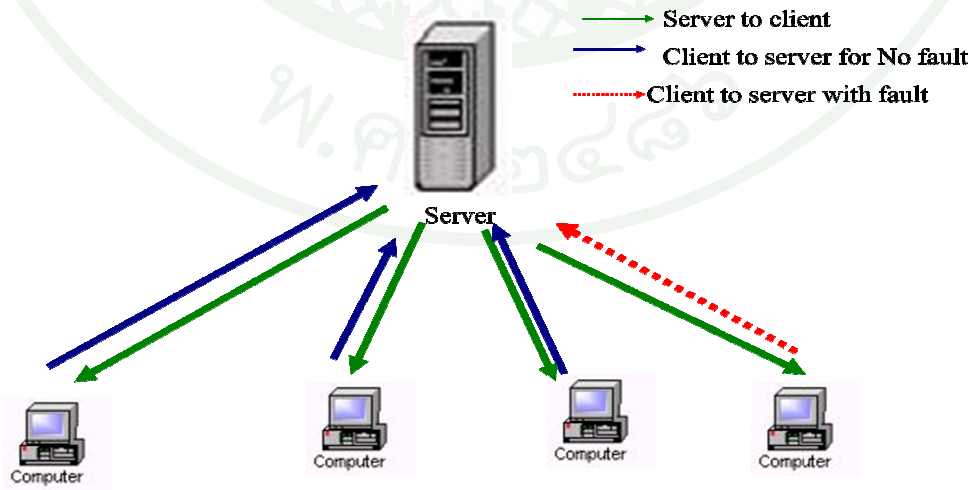
อุปกรณ์และวิธีการ

อุปกรณ์

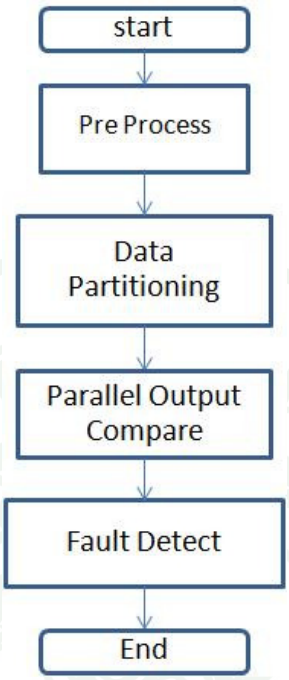
1. ฮาร์ดแวร์ระบบประกอบด้วย
    1.1 PC CPU Pentium III 800 Mhz หน่วยความจำ 256 MBytes
    1.2 PC Cluster PIRUN ระบบคลัสเตอร์คอมพิวเตอร์ จำนวน 1 มี 72 เครื่อง
2. ซอฟต์แวร์ระบบ
    2.1 C Compiler
    2.2 Linux Redhat 6.1
    2.3 MPICH version 1.1.2

วิธีการ

เมื่อมีการนำการประมวลผลแบบขนานเข้ามาใช้ เราสามารถแบ่งการทำงานในส่วนของทาง Server ที่ทำการส่งงานและรับผลฝั่ง Client ตามภาพที่ 10 และกระบวนการที่เกิดขึ้นในภาพที่ 11

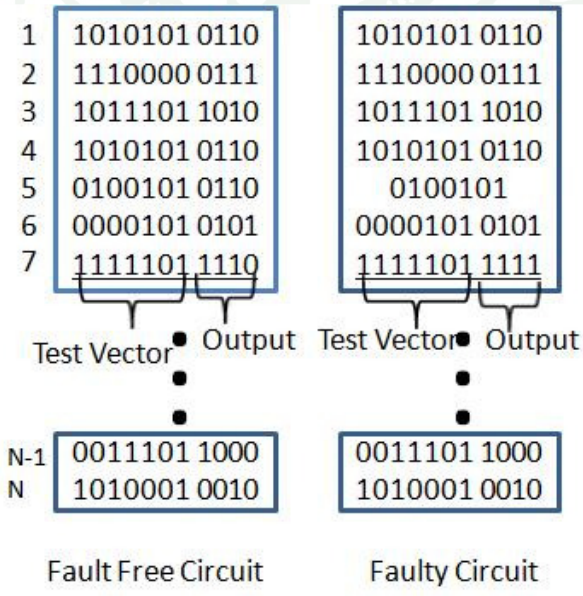


ภาพที่10 แสดงแบบการรับส่งข้อมูลแบบ Client และ Server ในระบบ



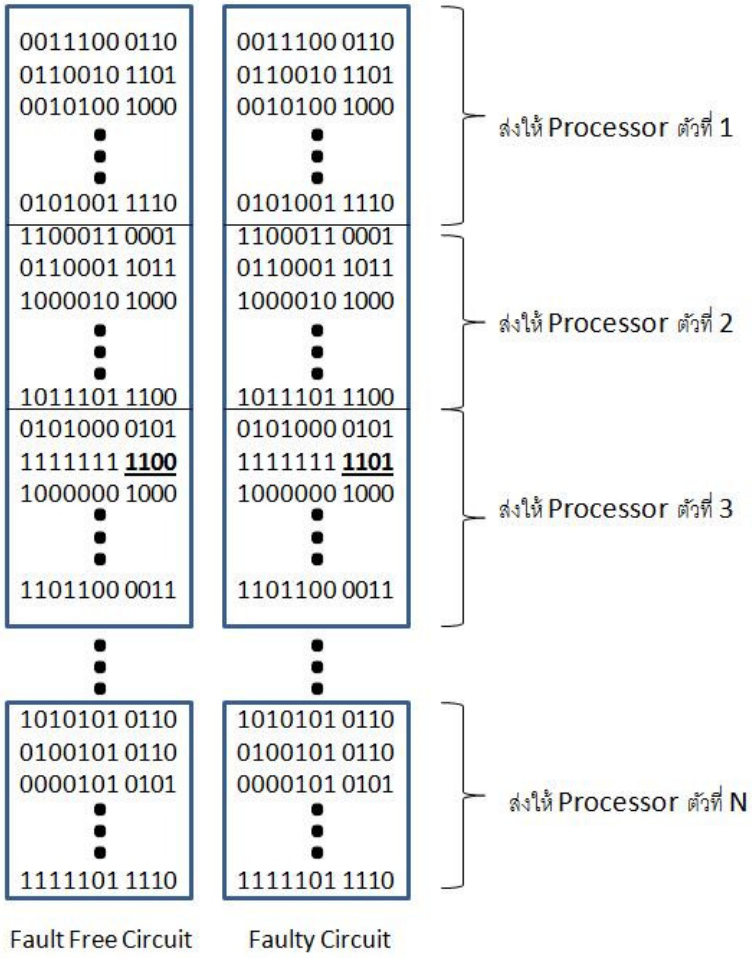
ภาพที่ 11 แสดงกระบวนการที่เกิดขึ้นทั้งระบบ

ในกระบวนการใน Pre Process เป็นกระบวนการเตรียมข้อมูลก่อนที่จะทำการส่งข้อมูลไป ให้แต่ละ Client โดยกระบวนการ Pre Process และผลลัพธ์แสดงไว้ในภาพที่ 12



ภาพที่ 12 แสดงกระบวนการ Pre Process และผลลัพธ์ของกระบวนการ

เมื่อผ่านกระบวนการ Pre Process ขั้นตอนต่อไปคือการแบ่งข้อมูลและกระจายข้อมูลให้แต่ละ Client ดังภาพที่ 13



ภาพที่ 13 แสดงการแบ่งข้อมูลให้แต่ละ Client

ดังนั้นจากจากภาพที่ 11,12 และ 13 การทำงานของฝั่ง Server และ ฝั่ง Client เป็นดังนี้

การทำงานฝั่ง Server

1. ทำกระบวนการ Fault Simulation โดยรับค่า Input ที่เป็น Test Vector
2. ส่งค่า Test Vector, ค่า Output วงจรไปฝั่ง Client
3. ฝั่ง Server ทำงาน เดียวกันพร้อมกันกับแบบ Client
4. รอรับผล Fault ที่อาจเกิดขึ้นจาก Client และนำไปสรุปผล

การทำงานฝั่ง Client

1. รับค่า output ของวงจรมาจากทาง Server
2. ทำการ Compare หาความแตกต่างระหว่าง Fault Free Circuit และ Faulty Circuit
3. หากพบข้อแตกต่างให้เทียบกับ Fault List ว่ามีโอกาสเกิด Fault แบบใดและที่จุดใด
4. ส่งผล Fault ที่อาจเกิดขึ้นกลับไปหา Server

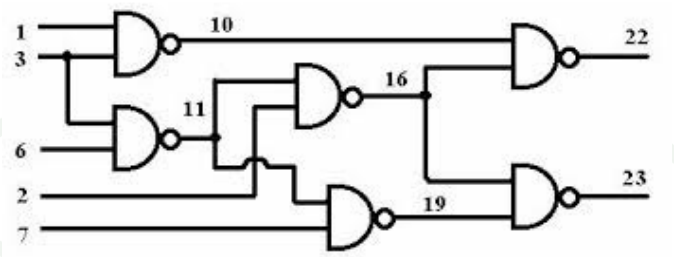
กรรมวิธีในการค้นหา จุดผิดพลาดในวงจรจาก Fault Simulation Tool

เราใช้วงจรมาตรฐานที่ได้มีการกำหนดไว้จากการประชุม International Symposium on Circuits And Systems (ISCAS) ในปี 1985 ได้มีการกำหนดให้ใช้วงจรมาตรฐาน สำหรับ Combinational Circuit แล้วกำหนดชื่อว่า ISCAS85 และในปี 1989 ได้ มีการกำหนดให้ใช้วงจรมาตรฐาน สำหรับ Sequential Circuit แล้วกำหนดชื่อว่า ISCAS89 วงจรมาตรฐานแต่ละตัวจะทำการกำหนดลักษณะ ของ วงจรแต่ละตัวว่าจะมี จำนวน ลักษณะการเชื่อมต่อ Input Output และ Gate สำหรับ Combinational Circuit และ มีจำนวน ลักษณะการเชื่อมต่อ Input Output Gate และ Flipflop สำหรับ Sequential Circuit วงจรมาตรฐาน ISCAS85 Combinational Circuit Benchmark และวงจรมาตรฐาน ISCAS89 Sequential Circuit Benchmark จะมีลักษณะ ดังตารางที่ 1 และตารางที่ 2

ตารางที่ 1 แสดงลักษณะของวงจรมาตรฐาน ISCAS85 Combinational Circuit

Circuit	# of PI	# of PO	# of Gates
c432	36	7	160
c499	41	32	202
c880	60	26	383
c1355	41	32	546
c1908	33	25	880
c2607	233	140	1193
c3540	50	22	1669
c5315	178	123	2307
c6288	32	32	2416
c7552	207	108	3512

จากตาราง #of PI คือจำนวน Primary Input ทั้งหมด และ #of PO คือจำนวน Primary Output ทั้งหมด ในวงจร ตัวอย่าง Schematic และ netlist ของวงจร C17 ที่เป็น Combinational circuit แสดงดังภาพที่ 12 และ 13 ดังนี้



ภาพที่ 14 Schematic Diagramวงจรทดสอบ C17

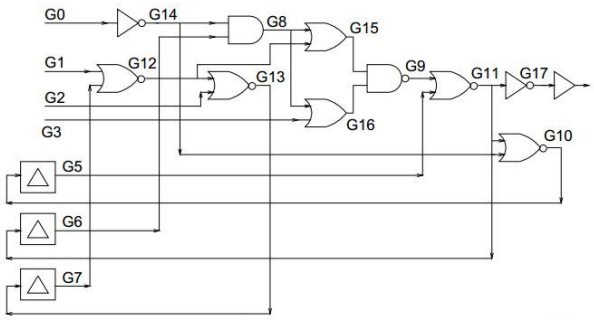
```
# c17
# 5 inputs
# 2 outputs
# 0 inverter
# 6 gates ( 6 NANDs )
INPUT(1)
INPUT(2)
INPUT(3)
INPUT(6)
INPUT(7)
OUTPUT(22)
OUTPUT(23)
10 = NAND(1, 3)
11 = NAND(3, 6)
16 = NAND(2, 11)
19 = NAND(11, 7)
22 = NAND(10, 16)
23 = NAND(16, 19)
```

ภาพที่ 15 Netlist ไฟล์ ของ C17

ตารางที่ 2 แสดงลักษณะของวงจรมาตรฐาน ISCAS89 Sequential Circuit

Circuit	# of PI	# of PO	# of Gates	# of Flip Flop
s298	3	6	119	14
s344	9	11	160	15
s349	9	11	161	15
s382	3	6	158	21
s386	7	7	159	6
s400	3	6	162	21
s444	3	6	181	21
s526	3	6	193	21
s641	35	24	379	19
s713	35	23	393	19
s820	18	19	289	5
s832	18	19	287	5
s1196	14	14	529	18
s1238	14	14	508	18
s1423	17	5	657	74
s1488	8	19	653	6
s1494	8	19	647	6
s5328	35	49	2779	179

สำหรับตัวอย่าง Schematic และ netlist ของวงจร S27 ที่เป็น Sequential circuit แสดงดังภาพที่ 16 และ 17 ดังนี้



ภาพที่ 16 Schematic Diagram วงจร S27

```
# s27
# 4 inputs
# 1 outputs
# 3 D-type flipflops
# 2 inverters
# 8 gates (1 ANDs + 1 NANDs + 2 ORs + 4 NORs)
INPUT(G0)
INPUT(G1)
INPUT(G2)
INPUT(G3)
OUTPUT(G17)
G5 = DFF(G10)
G6 = DFF(G11)
G7 = DFF(G13)
G14 = NOT(G0)
G17 = NOT(G11)
G8 = AND(G14, G6)
G15 = OR(G12, G8)
G16 = OR(G3, G8)
G9 = NAND(G16, G15)
G10 = NOR(G14, G11)
G11 = NOR(G5, G9)
G12 = NOR(G1, G7)
G13 = NOR(G2, G12)
```

ภาพที่ 17 Netlist ไฟล์ ของ S27

เมื่อผ่านจากกระบวนการ ATPG จะได้ ค่าไฟล์ Test Vector ของวงจร ISCAS85 C17 ที่ได้จากกระบวนการ ATPG จะได้เป็นดังนี้

ตารางที่ 3 แสดงค่า Test Vector ของวงจร C17

ลำดับ	Test Vector
1	10100
2	01001
3	11111
4	01010
5	00011
6	10000
7	00111

สำหรับวงจรอื่นๆ ผลจากการใช้ ATPG Tool เข้ามาเพื่อลดจำนวน test vector ลง จะได้จำนวน Vector สำหรับ Combinational Circuit ดังตารางที่ 4 และ Sequential circuit ดังตารางที่ 5

ตารางที่ 4 แสดงจำนวน Test vector ของ ISCAS85

Circuit	# of PI	# of PO	# of Test Vector
c432	36	7	48
c499	41	32	54
c880	60	26	61
c1355	41	32	84
c1908	33	25	117
c2607	233	140	107
c3540	50	22	150
c5315	178	123	122
c6288	32	32	32
c7552	207	108	223

ตารางที่ 5 แสดงจำนวน Test vector ของ ISCAS89

Circuit	# of PI	# of PO	# of Flip Flop	# of Test Vector
s298	3	6	14	246
s344	9	11	15	175
s349	9	11	15	145
s382	3	6	21	629
s386	7	7	6	303
s400	3	6	21	480
s444	3	6	21	647
s526	3	6	21	284
s641	35	24	19	573
s713	35	23	19	559
s820	18	19	5	735
s832	18	19	5	776
s1196	14	14	18	894
s1238	14	14	18	910
s1423	17	5	74	839
s1488	8	19	6	885
s1494	8	19	6	903
s5328	35	49	179	815

จำนวน Test vector ของวงจรจะขึ้นกับ ATPG หน้าที่ของ ATPG คือการสร้าง Test Vector ที่มาจากค่า Input และจะทำงานเสร็จสิ้นเมื่อ Fault Coverage มีค่า 100 % หรือระบุ Fault ทุกตัวในวงจรได้ว่า สามารถทดสอบได้หรือไม่ ดังนั้น ATPG ที่ดีเมื่อทำงานเสร็จจะได้ชุด Test Vector ที่มีขนาดเล็กสุดซึ่งบางโปรแกรมจะใช้เวลานาน และหากใช้เทคนิค Test Compaction ก่อนเสร็จสิ้นการทำงาน จะลดขนาด Test Vector ลงไปอีก โดยจำนวน Input มีผลต่อขนาด Test Vector ทั้งใน ATPG และเทคนิค Test Compaction

ในกระบวนการ หา Test Vector โดย ATPG จะมีส่วนการหา fault Coverage ทำให้เราเราสามารถทราบถึง จำนวน Fault ทั้งหมดที่เกิดได้ในวงจร สัมพันธ์กับค่า Test Vector ตามตารางที่ 6 ดังนี้

ตารางที่ 6 แสดง ผล Fault ของ วงจร C17 ตาม Test Vector

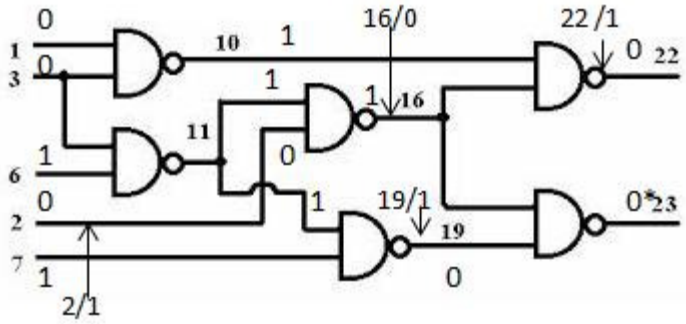
ลำดับ	Test vector	Fault	คำอธิบาย Fault
1	00xxx	22 /1	เกิด stuck-at 1 ที่จุด 22
2	101xx	10 /1	เกิด stuck-at 1 ที่จุด 10
3	1x1xx	22 /0	เกิด stuck-at 0 ที่จุด 22
4	010xx	16->22 /1	เกิด stuck-at 1 ที่เส้น 16ไปขาเข้า gateที่มี output22
5	100xx	3->10 /1	เกิด stuck-at 1 ที่เส้น 3ไปขาเข้าgate ที่มี output 10
6	001xx	1 /1	เกิด stuck-at 1 ที่จุด 1
7	101xx	3 /0	เกิด stuck-at 0 ที่จุด 3
8	100xx	3 /1	เกิด stuck-at 1 ที่จุด 3
9	010xx	16 /1	เกิด stuck-at 1 ที่จุด 16
10	00xxx	16 /0	เกิด stuck-at 0 ที่จุด 16
11	x111x	11->16 /1	เกิด stuck-at 1 ที่เส้น 11ไปขาเข้าgate ที่มี output 16
12	000xx	2 /1	เกิด stuck-at 1 ที่จุด 2
13	x10xx	11 /0	เกิด stuck-at 0 ที่จุด 11
14	x101x	3->11 /1	เกิด stuck-at 1 ที่เส้น 3ไปขาเข้าgate ที่มี output 11
15	0111x	11 /1	เกิด stuck-at 1 ที่จุด 11
16	0110x	6 /1	เกิด stuck-at 1 ที่จุด 6
17	x0xx0	23 /1	เกิด stuck-at 1 ที่จุด 23
18	x00x1	19 /1	เกิด stuck-at 1 ที่จุด 19
19	x10xx	23 /0	เกิด stuck-at 0 ที่จุด 23
20	x10x0	16->23 /1	เกิด stuck-at 1 ที่เส้น 16ไปขาเข้าgate ที่มี output 23
21	xx111	11->19 /1	เกิด stuck-at 1 ที่เส้น 11ไปขาเข้าgate ที่มี output 19
22	x00x0	7 /1	เกิด stuck-at 1 ที่จุด 7

แต่เมื่อพิจารณา Test vector จากตารางที่ 6 และทำการลดขนาดลงแล้วพบว่า Test Vector 1 ตัวสามารถนำไปตรวจสอบ Fault ได้มากกว่า 1 ตำแหน่ง ตามที่แสดงไว้ในตารางที่ 7 สำหรับวงจร C17 ดังนี้

ตารางที่ 7 แสดงค่า fault ที่อาจพบเจอตาม Test Vector ทั้งหมด สำหรับวงจร C17

ลำดับ	Test Vector	Output Fault Free	ลักษณะ fault ที่ พบ
1	10100	10	10 /1, 3 /0, 23 /1
2	01001	11	16->22 /1,16 /1, 11 /0, 6 /1, 23 /0
3	11111	10	22 /0, 11->16 /1,11 /1, 11->19 /1
4	01010	11	3->11 /1,16->23 /1
5	00011	01	22 /1, 16 /0, 2 /1, 19 /1
6	10000	00	3->10 /1, 3 /1, 7 /1
7	00111	00	1 /1

จากตารางที่ 7 เมื่อป้อน Test Vector ลำดับที่ 5 คือ 00011 หากไม่มี Fault เกิดขึ้นในวงจร C17 ค่า logic จะแสดงค่า 01 เหมือนค่าของ Output ใน Fault Free Circuit แต่หากไม่ได้แสดงค่า 01 ก็จะเกิด Fault เกิดขึ้นตามภาพที่ 18



ภาพที่ 18 แสดง Faulty Circuit และ Fault ที่อาจจะเกิดขึ้นได้ในวงจร

จากภาพที่ 18 เมื่อค่า Output ที่ได้เป็น 00 จะสามารถระบุลักณะ Fault ที่อาจเกิดขึ้นคือ Stuck-at 1 ที่ จุด 2, Stuck-at 0 ที่ จุด 16, Stuck-at 1 ที่ จุด 19 และ Stuck-at 1 ที่ จุด 22

จากที่มีการดัดแปลง Fault Simulation ไปใช้งานจริง จะทำให้ทราบว่า เวลาที่ใช้ในกระบวนการทั้งหมดตั้งแต่ การหาความแตกต่างระหว่าง Output ในแต่ละตัวตลอดจน สามารถระบุได้ว่า Fault สามารถอยู่ในจุดใดของวงจรนั้น จะขึ้นอยู่กับ

1. ขนาดของ จำนวน Input ของวงจร และFlipflop (สำหรับวงจร Sequential) ใน Circuit ที่ทำให้เกิด ชุด Test Vector จำนวนมาก แต่มีแก้ปัญหาโดยการนำ ATPG เข้าสร้าง Test Vector
2. จำนวน Output ของวงจร ถ้า Output มาก ก็ต้องเปรียบเทียบค่ามาก
3. ขนาดของ Fault List ที่ได้มีผลต่อเวลาในการหาเมื่อรับค่า Test Vector แล้วนำค่า Fault ที่น่าจะเกิดขึ้นออกมาที่เกิดขึ้นมา

ดังนั้นจึงสรุปปัจจัยที่มีผลต่อเวลา ในการทำ Fault Simulation ได้ตามตารางแสดงค่า Input Output Test Vector และจำนวน Fault วงจรทั้ง Combinational Circuit และ Sequential Circuit ได้ดังตารางที่ 8 และ ตารางที่ 9 ดังนี้

ตารางที่ 8 แสดงค่า Input Output Test Vector และจำนวน Fault ใน ISCAS85

Circuit	# of PI	# of PO	# of Test Vector	# of Fault
c432	36	7	48	524
c499	41	32	54	758
c880	60	26	61	942
c1355	41	32	84	1574
c1908	33	25	117	1879
c2607	233	140	107	2747
c3540	50	22	150	3428
c5315	178	123	122	5350
c6288	32	32	32	7744
c7552	207	108	223	7550

ตารางที่ 9 แสดงค่า Input Output Test Vector และจำนวน Fault ใน ISCAS89

Circuit	# of PI	# of FlipFlop	# of PI+Flipflop	# of PO	# of Test Vector	# of Fault
s298	3	14	17	6	246	264
s344	9	15	24	11	175	329
s349	9	15	24	11	145	335
s382	3	21	24	6	629	362
s386	7	6	13	7	303	312
s400	3	21	24	6	480	381
s444	3	21	24	6	647	424
s526	3	21	24	6	284	448
s641	35	19	54	24	573	404
s713	35	19	54	23	559	476
s820	18	5	23	19	735	573
s832	18	5	23	19	776	541
s1196	14	18	32	14	894	1186
s1238	14	18	32	14	910	1227
s1423	17	74	91	5	839	1345
s1488	8	6	14	19	885	1418
s1494	8	6	14	19	903	1423
s5328	35	179	214	49	815	3335

ในกระบวนการแบ่งเนื้องานไปให้ Client ตามภาพที่ 13 ขนาดเนื้องานมีการกำหนดตัวแปร P_{Size} (Problem size) ที่มีความสัมพันธ์กับจำนวน Output ของวงจร กับ Test Vector ของแต่ละ Output ตามสมการที่ 2 ดังนี้

$$P_{Size} = (\#\ of\ PO) * (\#\ of\ Test\ Vector) \quad (2)$$

# of PO = จำนวน Output ของวงจร

# of Test Vector = จำนวน Test Vector ของแต่ละ Output ในวงจร

และมีการแสดงขนาดปัญหา P_{Size} ของวงจรไว้ในตารางที่ 10 และตารางที่ 11

ตารางที่ 10 แสดงขนาดปัญหา P_{Size} ของวงจรไว้ในการเปรียบเทียบค่า ใน ISCAS85

Circuit	# of P_{Size}
c432	336
c499	1728
c880	1586
c1355	2688
c1908	2925
c2607	14980
c3540	3300
c5315	15006
c6288	1024
c7552	24084

ตารางที่ 11 แสดงขนาดปัญหา P_{Size} ของวงจรไว้ในการเปรียบเทียบค่าใน ISCAS89

Circuit	# of P_{Size}
s298	1476
s344	1925
s349	1595
s382	3774
s386	2121
s400	2880
s444	3882
s526	1704
s641	13752
s713	12857
s820	13965

ตารางที่ 11 (ต่อ)

Circuit	# of P_{Size}
s832	14744
s1196	12516
s1238	12740
s1423	4195
s1488	16815
s1494	17157
s5328	39935

Parallel Performance Analysis

เราสามารถคำนวณเวลา, Speed Up และ Efficiency ของกระบวนการตรวจสอบวงจรแบบขนานโดย ขนาด Output ทั้งหมดที่นำไปเปรียบเทียบค่า เท่ากับผลคูณของจำนวน Output (PO) ของวงจรกับจำนวน Test Vector ที่ใช้ทดสอบ Output แต่ละตัวในวงจรเนื่องจากข้อมูลทั้งหมดเป็นข้อมูลที่มีอิสระต่อกันจึงสามารถกระจายข้อมูลและงานไปยัง client ใน cluster ที่มี PC รวมกันอยู่ P ตัว กรณีที่ฝั่ง server ทำตัวเป็น Client ด้วย ก็จะมีการส่งค่าไปตัว Client อื่นๆอีก P-1 ครั้งและจำนวนข้อมูลที่ส่งค่ามีทั้งหมด P_{Size} /P ตัว เวลาที่ใช้ในระบบทั้งหมดจะแสดงดังสมการที่ 3

$$T_{total} = T_1 + T_2 + T_3 \qquad (3)$$

T_{total} = เวลาทั้งหมดของระบบ

T_1 = เวลาที่ใช้ก่อนการประมวลผลแบบขนาน

T_2 = เวลาที่ใช้ระหว่างการประมวลผลแบบขนาน

T_3 = เวลาที่ใช้หลังกระบวนการประมวลผลแบบขนาน

ส่วนเวลาที่ใช้ระหว่างการประมวลผลแบบขนานจะแสดงดังสมการที่ 4

$$T_2 = 1/P*(T_{comp} + T_{Comm} + T_{Idle}) \quad (4)$$

T_{comp} = เวลาที่ใช้ในการประมวลผลของ Client แต่ละตัว

T_{comm} = เวลาที่ใช้เชื่อมต่อระหว่าง Client กับ Server

T_{idle} = เวลาในช่วงสถานะที่รอการเชื่อมต่อระหว่าง Client กับ Server ด้วยเทคโนโลยีปัจจุบัน ค่า T_{idle} มีค่าน้อยมากเราสามารถละเลยได้

จากสมการที่ 4ค่า T_{comp} คือเวลาที่ใช้ในการทำงาน 1 ครั้งคูณกับจำนวนงานทั้งหมดที่เกิดขึ้นดังที่ได้แสดงในสมการที่ 5

$$T_{comp} = T_C*(P_{Size}) \quad (5)$$

T_C = เวลาที่ใช้ในการทำงานอันได้แก่การเทียบค่า และหา fault ที่เกิดในวงจร

P_{Size} = ขนาดของปัญหาหรือจำนวนข้อมูลที่เป็นผลคูณของ จำนวน Output ทั้งหมดในวงจรกับจำนวน Test Vector ของแต่ละ Output ในวงจร

และ T_{comm} ที่เป็นเวลาการเชื่อมต่อระหว่าง Client กับ Server สามารถแสดงได้ในสมการที่ 6

$$T_{comm} = T_{St} + T_W * (\frac{P_{Size}}{P}) \quad (6)$$

T_{St} = เวลาที่ใช้ในการเริ่มต้นส่งค่าในเครือข่ายระหว่าง server and client

T_w = เวลาที่ใช้ในการส่งค่าระหว่าง Server กับ Client

จากสมการที่ 6 แสดงค่า T_{comm} ระหว่าง server and 1 Client และ ขนาดปัญหา ด้วยจำนวน Client P-1 ตัวกับ 2 จำนวนชุด Output สำหรับการเปรียบเทียบค่ากัน , ค่า T_{comm} โดยรวมสามารถคำนวณโดยแทนค่าสมการลงในสมการที่ 7 ได้

$$T_{comm} = 2*(P-1)*(T_{St} + T_W * (\frac{P_{Size}}{P})) \quad (7)$$

ดังนั้นเวลาจากกระบวนการประมวลผลแบบขนาน (T_2) ที่ได้แจกแจงในสมการที่ 4 เราสามารถแทนค่า T_{comp} ด้วยสมการที่ 5 และแทน T_{comm} ด้วยสมการที่ 7 และละเลย T_{Idle} เราจะได้ T_2 ดังสมการข้างล่าง

$$T_2 = \frac{1}{P} * [T_C * P_{Size} + 2(P-1) * (T_{St} + T_W * (\frac{P_{Size}}{P}))] \qquad (8)$$

ดังนั้นเราสามารถแสดงค่าของเวลาในระบบทั้งหมด $T_{total} = T_1 + T_2 + T_3$ และเวลาที่วัดระหว่างการประมวลผลแบบขนาน T_2 ที่ได้แจกแจงในสมการที่ 8 เมื่อ P_{Size} เป็นขนาดปัญหาและ P คือจำนวน คอมพิวเตอร์ทั้งหมดใน Cluster

หลังจากที่เราได้เวลาที่ใช้ในกระบวนการประมวลผลแบบขนาน เราสามารถคำนวน Efficiency กับ Speedup ด้วยข้อมูลที่เป็นอิสระต่อกัน Client สามารถดำเนินการประมวลผลแบบขนานได้อย่างอิสระ ค่า Efficiency และสามารถแสดงได้ในสมการที่ 9 และ 10

$$E = \frac{T_S}{P * T_P} \qquad (9)$$

$$S = P*E \qquad (10)$$

E = ค่า Efficiency

S = ค่า Speed Up

P = จำนวน Processor

T_S = เวลาประมวลแบบ Sequential หมายถึงใช้ Client เพียง 1 ตัว

T_P = เวลาที่วัดระหว่างการประมวลผลแบบขนาน

เราสามารถคำนวณหาค่า Efficiency ที่แสดงในสมการที่ 9 แทนค่า T_P หรือค่า T_2 จากสมการที่ 8 เราจะได้ค่า Efficiency ตามสมการที่ 11

$$E = \frac{T_S}{[T_C * P_{Size} + 2(P-1) * (T_{St} + T_W * (\frac{P_{Size}}{P}))]} \quad (11)$$

สำหรับค่า Speed up เราสามารถหาได้จาก สมการ Speed up นั่นคือสมการ 10 และแทนค่า Efficiency จากสมการ 11 จะได้ค่า Speedup เป็นดังสมการที่ 12

$$Speedup = P * \frac{T_S}{[T_C * P_{Size} + 2(P-1) * (T_{St} + T_W * (\frac{P_{Size}}{P}))]} \quad (12)$$

ผลและวิจารณ์

ได้ทำการปรับปรุงโปรแกรม HOPE ให้เพื่อใช้ทดสอบ แบบ Combinational Circuit ได้ และปรับปรุงให้โปรแกรมมีส่วนการวิเคราะห์หาจุดผิดพลาดมีค่า Output แตกต่างกัน จนสามารถ ทราบ fault ที่น่าจะเกิดขึ้นได้ของวงจรทดสอบ มาตรฐาน ISCAS85 สำหรับ Combinational Circuit และ วงจรทดสอบ มาตรฐาน ISCAS89 สำหรับ Sequential Circuit ส่วนกระบวนการสร้าง Test Vector ได้ใช้ HPGAST ในการสร้าง Test vector สำหรับ Sequential Circuit แต่สำหรับ Combinational Circuit ได้พัฒนา ATPG Tools ที่มี Fault Coverage 98-100 % (เมื่อทดสอบกับ ISCAS85) เพื่อสร้าง Test vector สำหรับการทดสอบและ การทดลองแบบ Sequential ทำบนเครื่อง PC 800 MHz Pentium III ที่ หน่วยความจำ 256 MBส่วนการพัฒนาส่วนการประมวลผลเชิงขนาน มีการใช้ MPI Library ในส่วนการรับส่งข้อมูล และประมวลผล ทดลองบนเครื่อง Pirun Beowulf PC-Cluster ที่ 1,2,4,8,16 Computing Node

ในการทดลอง ก่อนที่จะทำการส่งค่า output ทั้งสองแบบไปนั้นทางฝั่ง Fault Circuit จะมีการเปลี่ยนค่าให้ไม่เหมือนกันกับค่าของ Fault-Free Circuit 1 ตัวแบบสุ่ม จากนั้นก็ทำการส่งค่าข้อมูลทั้งสองแบบที่ผ่านการแบ่งงานแล้วไปยังฝั่ง Client ๆใน Cluster ตัว Client จะตรวจสอบว่ามีโอกาสเกิด Fault ที่อาจจะเกิดขึ้นกลับมาแล้วทำการสรุป

ผล

สำหรับ Combinational Circuit เมื่อผ่านการ Run แบบ Sequential จะแสดงได้ดังตารางที่ 12 ข้างล่างนี้

ตารางที่ 12 เวลาที่ใช้ในการหาข้อผิดพลาดวงจรแบบ Combinational ใน ISCAS85

Circuit	Time(sec)
c17	0.87
c432	3.63
c499	8.47
c880	9.67
c1355	11.43

ตารางที่ 12 (ต่อ)

Circuit	Time(sec)
c1908	12.15
c2607	19.81
c3540	13.23
c5315	20.15
c6288	7.88
c7552	24.34

สำหรับ Sequential Circuit เมื่อผ่านการ Run แบบ Sequentialจะได้ผลดังตาราง 13 ข้างล่างนี้

ตารางที่ 13 เวลาที่ใช้ในการหาข้อผิดพลาดวงจรแบบ Sequential ใน ISCAS89

Circuit	Time (sec)
s298	7.67
s344	9.31
s349	8.82
s382	10.43
s386	9.45
s400	9.75
s444	10.23
s526	8.65
s641	17.02
s713	15.32
s820	15.25
s832	18.77
s1196	15.82
s1238	16.17

ตารางที่ 13 (ต่อ)

Circuit	Time (sec)
s1423	10.55
s1488	20.87
s1494	21.17
s5328	35.15

สำหรับ Combinational Circuit เมื่อผ่านการ Run แบบ Parallel ตามจำนวน PC ที่ใช้งาน 1,2,4,8,16 ตัว จะได้ผลตารางที่ 14 และเมื่อทำการวิเคราะห์หา Speed up จะได้ดังตารางที่ 15 ดังนี้

ตารางที่ 14 เวลาที่ใช้ในการหาข้อผิดพลาดวงจรที่ 1,2,4,8,16 Processor ใน ISCAS85

	Time (sec)				
	# of Processor				
Circuit	1	2	4	8	16
c17	0.87	0.84	0.83	0.83	0.83
c432	3.63	2.77	2.52	2.17	2.05
c499	8.47	5.96	3.96	3.44	3.33
c880	9.67	6.86	4.58	3.96	3.85
c1355	11.43	8.05	4.74	4.22	4.02
c1908	12.15	8.44	4.98	4.52	4.35
c2607	19.81	12.78	6.95	6.52	6.13
c3540	13.23	9.59	6.55	5.49	5.33
c5315	20.15	12.07	6.65	5.91	5.52
c6288	7.88	5.79	4.02	3.4	3.32
c7552	27.34	16.77	9.36	7.75	6.97

ตารางที่ 15 Speed up ที่ใช้ในการหาข้อผิดพลาดวงจรที่ 1,2,4,8,16 Processor ใน ISCAS85

	Speed up			
	# of Processor			
Circuit	2	4	8	16
c17	1.04	1.05	1.05	1.05
c432	1.31	1.44	1.67	1.77
c499	1.42	2.14	2.46	2.54
c880	1.41	2.11	2.44	2.51
c1355	1.42	2.41	2.71	2.84
c1908	1.44	2.44	2.69	2.79
c2607	1.55	2.85	3.04	3.23
c3540	1.38	2.02	2.41	2.48
c5315	1.67	3.03	3.41	3.65
c6288	1.36	1.96	2.32	2.37
c7552	1.63	2.92	3.53	3.92

สำหรับ Sequential Circuit เมื่อผ่านการ Run แบบ Parallel ตามจำนวน PC ที่ใช้งาน 1,2,4,8,16 ตัว ผลที่ได้จะเป็นดังตารางที่ 16 ข้างล่างนี้

ตารางที่ 16 เวลาที่ใช้ในการหาข้อผิดพลาดวงจรที่ 1,2,4,8,16 Processor ใน ISCAS89

	Time (sec)				
	# of Processor				
Circuit	1	2	4	8	16
s298	7.67	6.34	3.53	3.18	3.14
s344	9.31	6.8	3.93	3.51	3.47
s349	8.82	7.11	3.92	3.38	3.3
s382	10.43	7.56	4.44	3.91	3.85
s386	9.45	6.9	4.07	3.65	3.61

ตารางที่ 16 (ต่อ)

	Time (sec)				
	# of Processor				
Circuit	1	2	4	8	16
s400	9.75	7.22	4.2	3.65	3.6
s444	10.23	7.75	4.43	3.92	3.89
s526	8.65	6.71	3.81	3.28	3.23
s641	17.02	10.57	6.17	5.83	5.73
s713	15.32	9.82	5.51	5.02	4.76
s820	15.25	9.13	5.43	5.12	4.89
s832	18.77	11.59	6.93	6.22	6.04
s1196	15.82	10.21	5.82	5.17	5.04
s1238	16.17	9.57	5.63	5.2	5.04
s1423	10.55	6.9	3.89	3.49	3.44
s1488	20.87	12.42	6.69	6.27	6.12
s1494	21.17	13.07	6.94	6.21	5.93
s5328	35.15	20.8	10.95	9.87	8.55

เมื่อทำการวิเคราะห์หา Speed up จะได้ดังตารางที่ 17 ข้างล่างดังนี้

ตารางที่ 17 Speed up ที่ใช้ในการหาข้อผิดพลาดวงจรที่ 1,2,4,8,16 Processor ใน ISCAS89

	Speed up			
	# of Processor			
Circuit	2	4	8	16
s298	1.21	2.17	2.41	2.44
s344	1.37	2.37	2.65	2.68
s349	1.24	2.25	2.61	2.67
s382	1.38	2.35	2.67	2.71

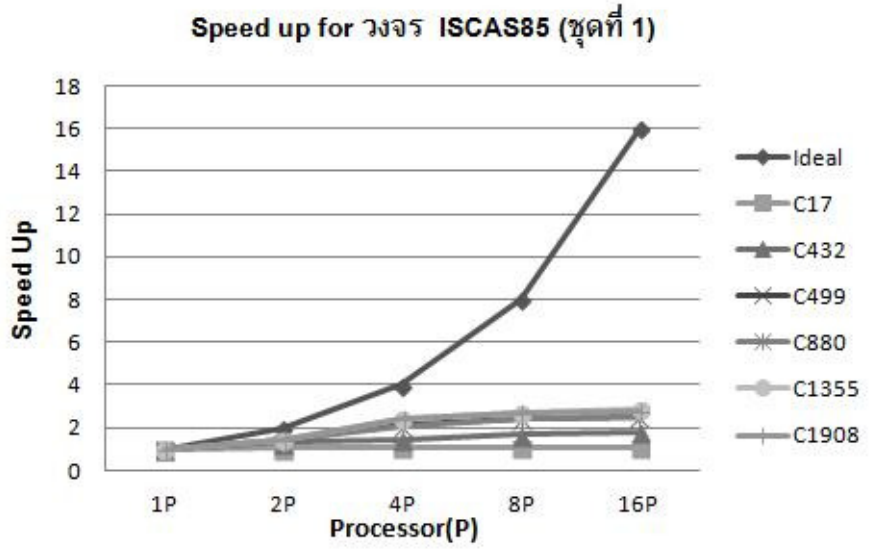
ตารางที่ 17 (ต่อ)

Circuit	Speed up			
	# of Processor			
	2	4	8	16
s386	1.37	2.32	2.59	2.62
s400	1.35	2.32	2.67	2.71
s444	1.32	2.31	2.61	2.63
s526	1.29	2.27	2.64	2.68
s641	1.61	2.76	2.92	2.97
s713	1.56	2.78	3.05	3.22
s820	1.67	2.81	2.98	3.12
s832	1.62	2.71	3.02	3.11
s1196	1.55	2.72	3.06	3.14
s1238	1.69	2.87	3.11	3.21
s1423	1.34	2.36	2.48	2.52
s1488	1.68	3.12	3.33	3.41
s1494	1.62	3.05	3.41	3.57
s5328	1.69	3.21	3.56	4.11

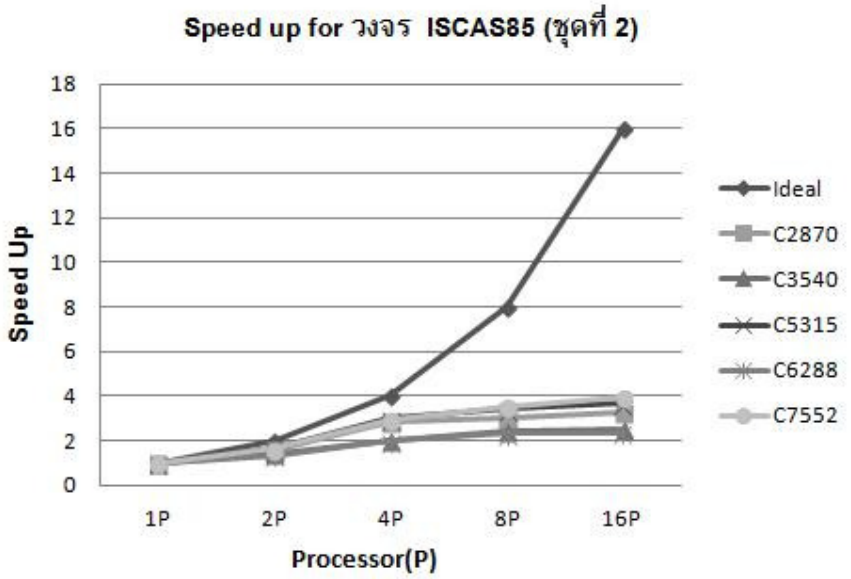
วิจารณ์

จากการทดลองเราจะทราบว่าเมื่อมีการใช้วิธีการประมวลผลแบบขนานนั้นเมื่อใช้งานที่ 2 CPU และ 4 CPU จะทำให้ค่า Speed up อยู่ในช่วง ประมาณ 1.5 ที่ 2 CPU และประมาณ ประมาณ 3 ที่ 4 CPU หรือ Efficiency ประมาณ 75% แต่เมื่อมีการเพิ่ม เป็น 8 CPU และ 16 CPU จะทำให้ Speed up มีค่าประมาณ 3.5 และ 4 ค่า Speed upเพิ่มขึ้นเพียงเล็กน้อย อาจเป็นเพราะว่าเวลาที่แสดงนั้นจะเป็นเวลาที่เกิดขึ้นจริงของตัวโปรแกรมที่เป็นส่วนวิเคราะห์หาส่วนที่ผิดพลาดของวงจรทำให้เวลาส่วนการประมวลผลแบบขนานไม่มีผลต่อเวลาที่ลดลงมากนัก โดยปัจจัยอื่นที่น่าจะมีผลเพิ่มเติมของเวลาที่ใช้ประมวลผลทั้งจึงน่าจะอยู่ที่ขนาด Fault ของแต่ละวงจร ที่ทำให้เกิดเวลาประมวลเพิ่มขึ้นเมื่อมีการนำผลของ Test Vector เข้ามาค้นหาและได้ Fault ที่น่าจะเกิดขึ้นออกมา นอกจากนั้นปัจจัยการเชื่อมต่อเครือข่ายน่าจะส่งผลเวลาที่ได้ในการทดลองครั้งนี้อีกด้วย

ค่า Speedup ที่ดีที่สุดจะมีค่าเท่ากับจำนวน Processor แต่จากการสังเกตผลการทดลองจะเห็นว่าจำนวนข้อมูลหากเป็นการทดสอบวงจรที่มีข้อมูลไม่มากจะทำให้ Speedup น้อยกว่า การทดสอบวงจรที่มีข้อมูลมากกว่า ส่งผลให้เกิด Efficiency น้อยกว่า สำหรับวงจร Combinational Circuit เมื่อทำการเปรียบเทียบค่า Speedup ของ ISCAS85 กับ ISCAS89 กับค่า Ideal Speedup ที่มีค่าเท่ากับจำนวน Processor จะเป็นดังภาพที่ 19 และ ภาพที่ 20

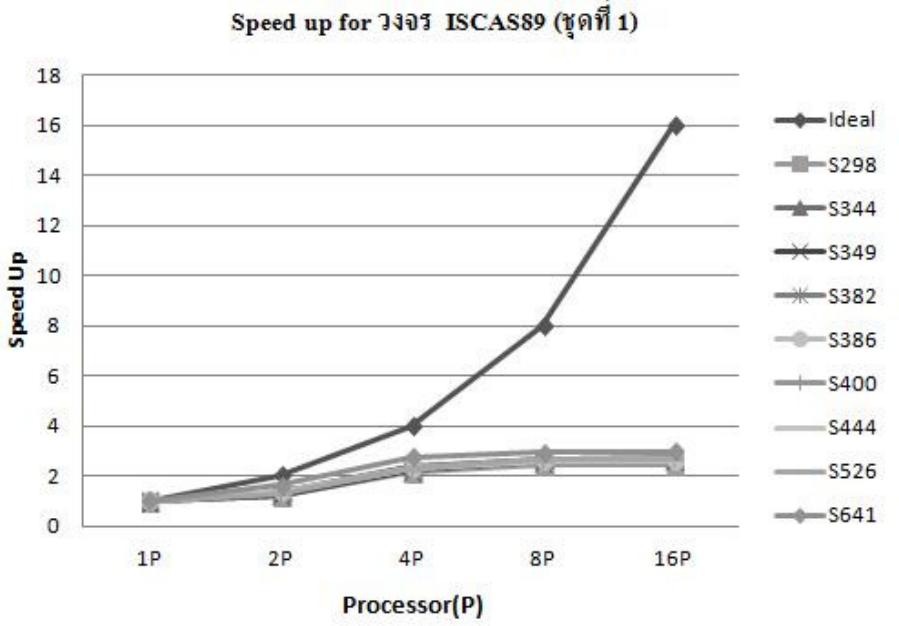


ภาพที่ 19 แสดงค่า Speedup ของวงจร ISCAS85 (ชุดที่ 1)ที่ # of Processors ต่างๆ

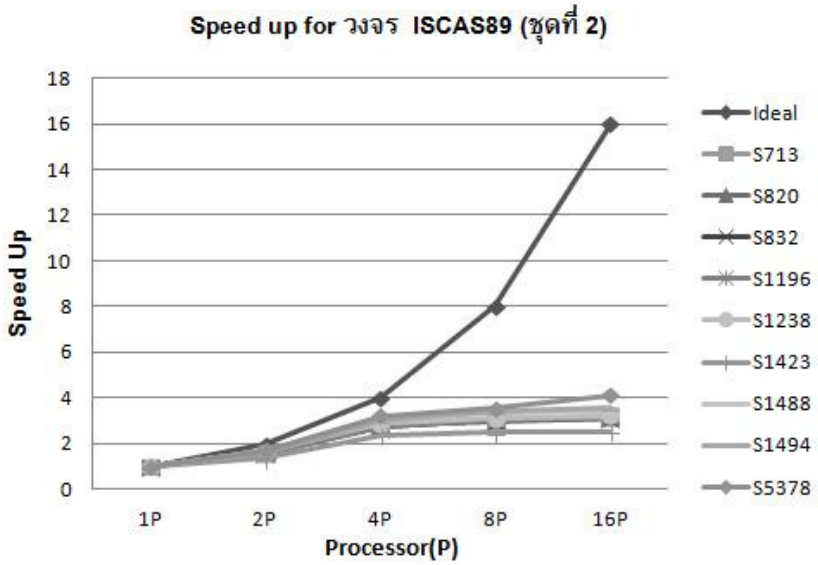


ภาพที่ 20 แสดงค่า Speedup ของวงจร ISCAS85 (ชุดที่ 2)ที่ # of Processors ต่างๆ

สำหรับวงจร Sequential Circuit เมื่อทำการเปรียบเทียบค่า Speedup กับค่า Ideal Speedup จะเป็นดังภาพที่ 21 และ ภาพที่ 22

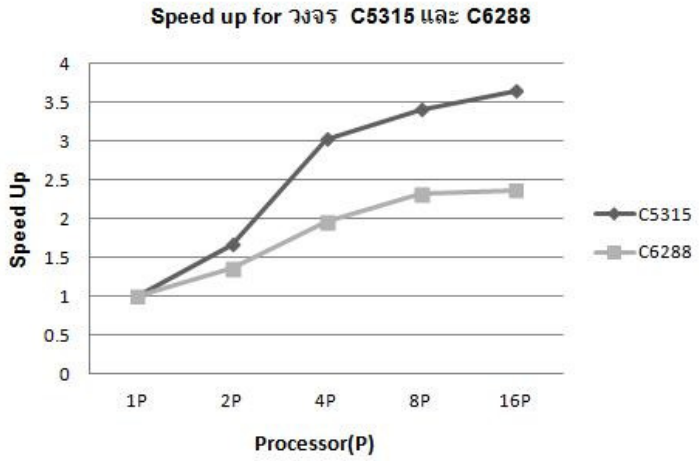


ภาพที่ 21 แสดงค่า Speedup ของวงจร ISCAS89 (ชุดที่ 1)ที่ # of Processors ต่างๆ



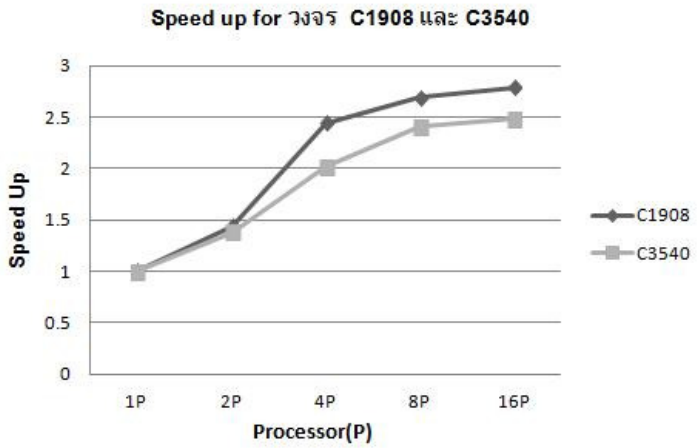
ภาพที่ 22 แสดงค่า Speedup ของวงจร ISCAS89 (ชุดที่ 2)ที่ # of Processors ต่างๆ

จากการทดลอง ปัจจัยที่มีผลต่อ Speed Up คือ ขนาด P_{Size} ของวงจรที่ขึ้นกับจำนวน Output และ Test Vector จะเห็นว่า วงจรที่มีขนาดปัญหา P_{Size} มากกว่า 10000 จะมีค่า Speed Up ที่ 1.5 ขึ้นไปที่ 2 CPU และ 2.7 ขึ้นไปที่ 4 CPU เมื่อเปรียบเทียบค่า Speed up ของวงจร C6288 กับ วงจร C5315 วงจร C6288 มี Gate ที่มากกว่า C5315 แต่วงจร C6288 มีขนาด P_{Size} ที่ 1024 ซึ่งน้อยกว่า C5315 ที่มีขนาด P_{Size} 15006 ทำให้ Speed Up ของ C5315 ดีกว่า C6288 ตามภาพที่ 23



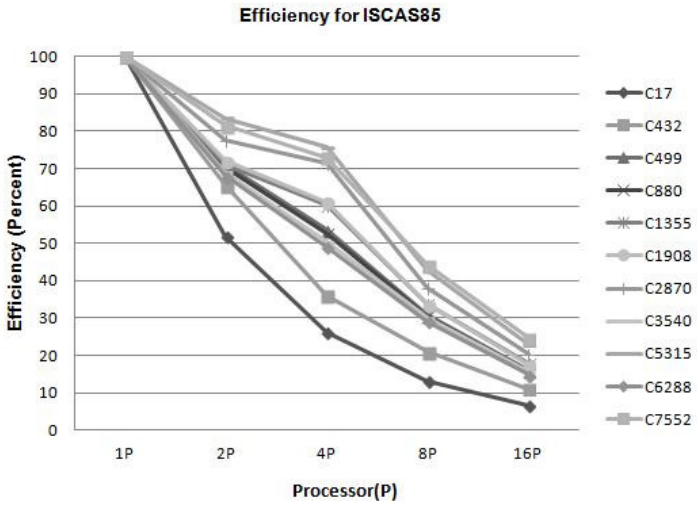
ภาพที่ 23 แสดงความแตกต่างของ Speed Up ที่ขนาด P_{Size} แตกต่างกัน

ในกรณีที่ P_{Size} ใกล้เคียงกันขนาด Fault List จะมีผลต่อเวลา หาก Fault แตกต่างกันมาก เช่น ในวงจร C1908 และ วงจร C3540 เป็นวงจรมี P_{Size} ที่ 3000 แต่ C1908 มีขนาด Fault List ประมาณ 1900 แต่ C3540 มีขนาด Fault List ประมาณ 3400 ทำให้ ค่า Speed Up ของวงจร C3540 มีขนาดน้อยกว่า C1908 ด้วยดังภาพที่ 24

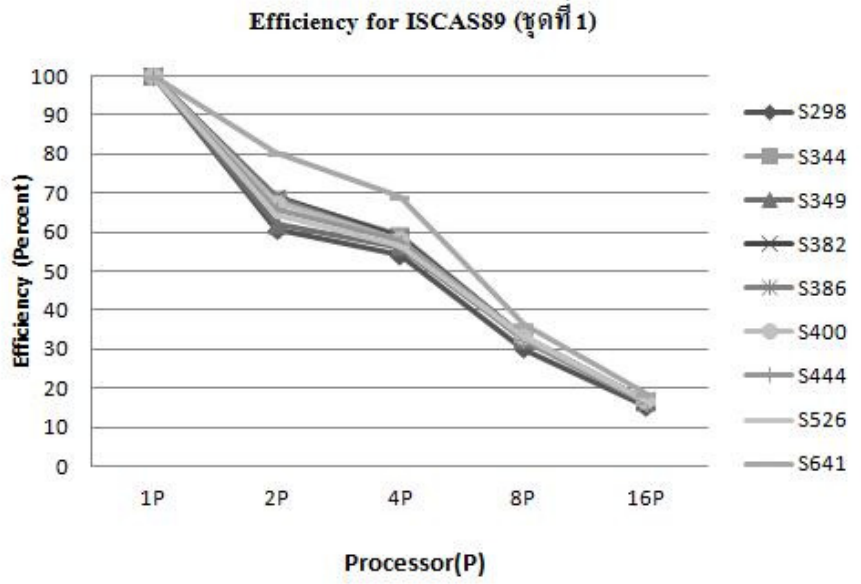


ภาพที่ 24 แสดงความแตกต่างของ Speed Up ที่ P_{Size} ใกล้เคียงกันแต่มีขนาด Fault List ต่างกัน

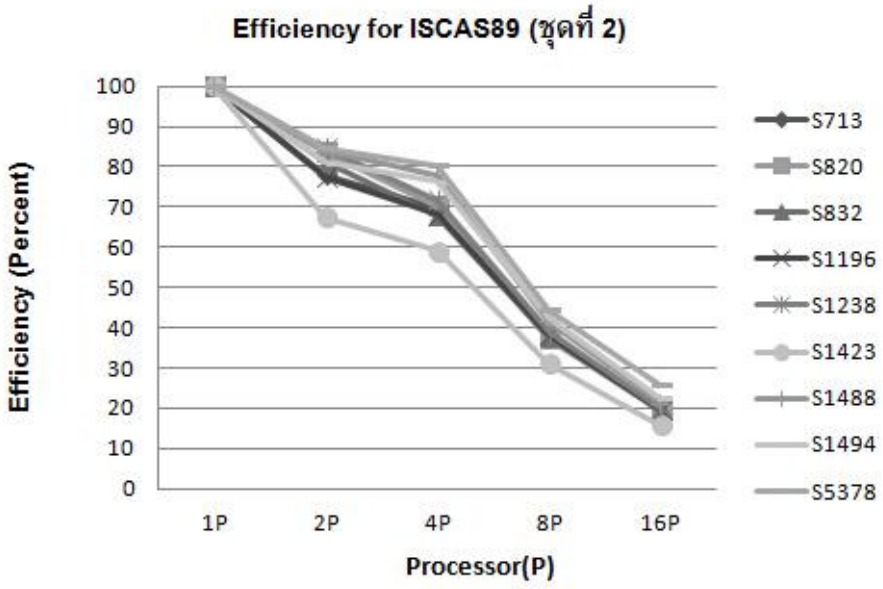
จาก Speed up เมื่อทำการหาค่า Efficiency ก็จะได้ดังรูปจะพบว่าวงจรที่เล็กกว่า จะมี Efficiency ที่ ลดลงมากกว่าวงจรขนาดใหญ่กว่า ดังภาพที่ 25 ภาพที่ 26 และภาพที่ 27



ภาพที่ 25 แสดงค่า Efficiency ของวงจร ISCAS85 ที่ # of Processors ต่างๆ



ภาพที่ 26 แสดงค่า Efficiency ของวงจร ISCAS89(ชุดที่ 1) ที่ # of Processors ต่างๆ



ภาพที่ 27 แสดงค่า Efficiency ของวงจร ISCAS89(ชุดที่ 1) ที่ # of Processors ต่างๆ

สรุปและข้อเสนอแนะ

สรุป

งานวิจัยนี้เป็นการนำ fault simulation tools มาใช้ในการหาจุดผิดพลาดในวงจร ซึ่งได้นำเสนอกระบวนการทดสอบ เพื่อมีจุดประสงค์หลักในการหาข้อผิดพลาดได้เร็วที่สุด ปัจจัยที่มีผลต่อเวลาในการค้นหาก็คือ จำนวน Test Vector ที่มีมากเกินไปจะทำให้หาข้อผิดพลาดได้ช้าลง จึงทำให้มีการนำ ATPG เข้ามาเพื่อลดจำนวน Test Vector ลง ทำให้มีผลให้การค้นหาข้อผิดพลาดทำได้อย่างรวดเร็วขึ้น

จากผลการจำลองโดยโปรแกรมที่ได้พัฒนาพบว่า ค่า Speed up ของการทำประมวลผลแบบขนาน อยู่ในช่วง ระหว่าง 1.5 - 3 ที่จำนวน Processor 2 CPU และ 4 CPU และจะมี Speed up โดยรวมประมาณ 4 สำหรับ 8 และ 16 CPU ตามลำดับ ทำให้วัดประสิทธิภาพ(Efficiency) ได้ที่ 70-80 เปอร์เซนต์ แต่เมื่อเพิ่มจำนวน Processor ขึ้นไป พบว่าเวลาจะลดเพียงน้อยลงซึ่งทำให้ค่า Speed up และ Efficiency ลดลง เนื่องมาจากการประมวลผลเชิงขนานในระดับนึงจะมีผลต่อเวลารวมของระบบน้อยลง เวลาในการทดสอบจะขึ้นกับการเชื่อมต่อเครือข่ายและลักษณะของโปรแกรมมากกว่า จากการจำลองพบว่าขนาดของปัญหา (P_{Size}) ของวงจรและขนาดของ Fault List มีผลกระทบต่อความเร็วในการประมวลผลโดยถ้าขนาดของปัญหา (P_{Size}) ที่มีค่าเท่ากับ จำนวนผลคูณของ Test Vector และ Output ของวงจร มีค่าเกิน 10000 ขึ้นไปจะให้ Speed up ที่ดี จึงสรุปได้ว่า ขนาดของ P_{Size} วงจรหากมีขนาดเล็กจะมีผลต่อ Speed up และ Efficiency น้อยกว่า วงจรที่มี P_{Size} ขนาดใหญ่ และที่ P_{Size} ใกล้เคียงกันขนาด ของ Fault List ที่แตกต่างกันจะมีผลทำให้ Speed up แตกต่างกันโดย Fault List ที่มีขนาดใหญ่จะได้ Speed Up ที่น้อยกว่าด้วย

ข้อเสนอแนะ

เวลาที่ใช้งานในการประมวลผลอาจลดลงได้อีกหากมีกระบวนการสืบค้น Fault ที่อาจจะเกิดขึ้นได้ดีกว่านี้ และ การค้นหาจุดผิดพลาดของวงจรในงานวิจัยนี้ เราสามารถนำไปใช้ ในการปัญหาของการค้นหา Fault ประเภทอื่นๆ เช่น Stuck-Open at transistor หรือ Transition Delay Fault ที่เป็นปัญหาคล้ายคลึงกับ Stuck-at fault ที่มีการ Propagate ค่า Logic เหมือนกัน แต่จะมีขนาดปัญหาที่ใหญ่ขึ้น และมีความคุ้มค่าในการใช้ การประมวลผลแบบมาใช้งานได้ต่อไป

เอกสารและสิ่งอ้างอิง

Breuer, M.A. and A.D. Friedman. 1976. **Diagnosis and Reliable Design of Digital Systems.** Computer Science Press, New York, NY, USA

Cheng, K.-T. 1996. Gate-Level Test Generation for Sequential Circuits. **ACM Trans. Des. Autom. Electron. Syst.** 1 (4): 405-442.

Fijany, A., J. Barhen and N. Toomarian. 1994. Massively Parallel Algorithms for Solution of the Schrodinger Equation, pp. 517-523. *In* **Parallel Processing Symposium,** 1994. **Proceedings., Eighth International.**

Garrido, J., E. Arias, D. Cazorla, F. Cuartero, I. Fernández and C. Gallardo. 2010. Promespar: A High Performance Computing Implementation of the Regional Atmospheric Model Promes, pp. 527-538. *In* S.-I. Ao and L. Gelman, eds. **Electronic Engineering and Computing Technology.** Springer, Netherlands.

Hsiao, M.S., E.M. Rudnick and J.H. Patel. 1996. Automatic Test Generation Using Genetically-Engineered Distinguishing Sequences, pp. 216-223. *In* **Proceedings of the** 14**th IEEE VLSI Test Symposium.** IEEE Computer Society.

Krstic, A. and K.T. Cheng. 1998. **Delay Fault Testing for Vlsi Circuits.** Springer, USA.

Lee, H.K. and D.S. Ha. 1991. An Efficient, Forward Fault Simulation Algorithm Based on the Parallel Pattern Single Fault Propagation, pp. 946-955. *In* **Proceedings of the IEEE International Test Conference on Test: Faster, Better, Sooner.** IEEE Computer Society.

_____, _____. 1992. Hope: An Efficient Parallel Fault Simulator for Synchronous Sequential Circuits, pp. 336-340. *In* **Proceedings of the** 29**th ACM/IEEE Design Automation Conference.** IEEE Computer Society Press, Anaheim, California, USA.

Li, J.C., T. Chao-Wen and E.J. McCluskey. 2001. Testing for Resistive Opens and Stuck Opens, pp. 1049-1058. *In*. **Test Conference,** 2001. **Proceedings. International.**

Li, K., Y. Pan and S.Q. Zheng. 1998. Fast and Processor Efficient Parallel Matrix Multiplication Algorithms on a Linear Array with a Reconfigurable Pipelined Bus System. **IEEE Trans. Parallel Distrib. Syst.** 9 (8): 705-720.

Miczo, A. 2003. **Digital Logic Testing and Simulation.** Wiley, New Jersey.

Patel, J.H. 1998. Stuck-at Fault: A Fault Model for the Next Millennium, p. 1166. *In*. **Test Conference,** 1998. **Proceedings., International.**

Pomeranz, I. and S.M. Reddy. 1996. Dynamic Test Compaction for Synchronous Sequential Circuits Using Static Compaction Techniques, pp. 53-61. *In* **Proceedings of the The Twenty-Sixth Annual International Symposium on Fault-Tolerant Computing (FTCS '**96). IEEE Computer Society.

Pramanick, A.K. and S.M. Reddy. 1988. On the Detection of Delay Faults, pp. 845-856. *In*. **Test Conference,** 1988. **Proceedings. New Frontiers in Testing, International.**

Rudnick, E.M., T.M. Niermann and J.H. Patel. 1991. Methods for Reducing Events in Sequential Circuit Fault Simulation, pp. 546-549. *In* **Computer-Aided Design,** 1991. **ICCAD-**91. **Digest of Technical Papers.,** 1991 **IEEE International Conference on.**

_____, J.H. Patel, G.S. Greenstein and T.M. Niermann. 1994. Sequential Circuit Test Generation in a Genetic Algorithm Framework, pp. 698-704. *In* **Design Automation,** 1994. 31**st Conference on.**

_____, _____, _____ and _____. 1997. A Genetic Algorithm Framework for Test Generation. **Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions On** 16 (9): 1034-1044.

Saab, D.G., Y.G. Saab and J.A. Abraham. 1992. Cris: A Test Cultivation Program for Sequential Vlsi Circuits, pp. 216-219. *In*. **Computer-Aided Design,** 1992. **ICCAD-**92. **Digest of Technical Papers.,** 1992 **IEEE/ACM International Conference on.**

Sar-Dessai, V.R. and D.M.H. Walker. 1999. Resistive Bridge Fault Modeling, Simulation and Test Generation, pp. 596-605. *In*. **Test Conference,** 1999. **Proceedings. International.**

Schulz, M.H., E. Trischler and T.M. Sarfert. 1988. Socrates: A Highly Efficient Automatic Test Pattern Generation System. **Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions On** 7 (1): 126-137.

Siriwan, T. 2001. **High Performance GA-Based for Sequential Circuit Test Generation on PC-Cluster.** M.Eng. Thesis, Kasetsart University.

ประวัติการศึกษา และการทำงาน

ชื่อ –นามสกุล	นายสิทธิพร ประภาวัต
วัน เดือน ปี ที่เกิด	วันที่ 5 สิงหาคม 2520
สถานที่เกิด	กรุงเทพมหานคร
ประวัติการศึกษา	วศ.บ.(วิศวกรรมคอมพิวเตอร์) จุฬาลงกรณ์มหาวิทยาลัย
ตำแหน่งหน้าที่การงานปัจจุบัน	เจ้าหน้าที่ระบบคอมพิวเตอร์
สถานที่ทำงานปัจจุบัน	อุทยานวิทยาศาสตร์ประเทศไทย ถนนพหลโยธิน คลองหนึ่ง คลองหลวง ปทุมธานี