การปรับปรุงเจเนติค อัลกอริทึม สำหรับการจัดวางตำแหน่งของแสตนดาร์ดเซล

A MODIFIED GENETIC ALGORITHM FOR STANDARD CELL PLACEMENT

อภิชาต วสุธาพิทักษ์
APICHART VASUTAPITUKS

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้า
บัณฑิตวิทยาลัย
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2545
ISBN 974-324-151-5

การปรับปรุง เจเนติค อัลกอริทึม สำหรับการจัดวางตำแหน่งของแสตนดาร์ดเซล

A MODIFIED GENETIC ALGORITHM FOR STANDARD CELL PLACEMENT

อภิชาต วสุธาพิทักษ์

APICHART VASUTAPITUKS

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2545

ISBN 974-324-151-5

A MODIFIED GENETIC ALGORITHM FOR STANDARD CELL PLACEMENT

APICHART VASUTAPITUKS

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENT FOR THE THE DEGREE OF
MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING
SCHOOL OF GRADUATE STUDIES
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
2002

ISBN 974-324-151-5

หัวข้อวิทยานิพนธ์ การปรับปรุง เจเนติค อัลกอริทึม สำหรับการจัดวางตำแหน่งของแสตนดาร์ดเซล

นักศึกษา นายอภิชาต วสุธาพิทักษ์

รหัสประจำตัว 42061126

ปริญญา วิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชา วิศวกรรมไฟฟ้า

พ.ศ. 2545

อาจารย์ผู้ควบคุมวิทยานิพนธ์ รศ.บรรจง ปิยธำรง

บทคัดย่อ

วิทยานิพนธ์ฉบับนี้ได้นำเสนอ การปรับปรุง เจเนติค อัลกอริทึม ที่เรียกว่าโมดิฟาย เจเนติค อัลกอริทึม (Modified Genetic Algorithm: MGA) สำหรับใช้แก้ปัญหาการจัดวาง (Placement) เลย์เอาต์ของแสตนดาร์ดเซล (Standard Cell) ซึ่งได้พัฒนาเจเนติค โอเปอเรเตอร์ (Genetic Operator) แบบใหม่จาก เจเนติค อัลกอริทึมแบบพื้นฐาน (Simple Genetic Algorithm) ที่เหมาะสมสำหรับการออกแบบวงจรรวมขนาดใหญ่ (VLSI) ระดับกายภาพ (Physical Design) โดยโมดิฟาย เจเนติค อัลกอริทึมเป็นการผสมผสานความรู้และความสามารถของวิธีการฮิวรีสติค (Heuristic Method) เข้ากับวิธีของเจเนติค อัลกอริทึมเข้าด้วยกัน ซึ่งจะได้ ฮิวรีสติค โอเปอเรเตอร์ (Heuristic Operator) แบบใหม่คือกรีดี้ครอสโอเวอร์ (Greedy Crossover) และมูเตชั่นแบบสามโอพีทรีฮิวรีสติค (Mutation Based on 3-Opt Heuristic) ซึ่งโอเปอเรเตอร์ทั้งสองจะช่วยเพิ่มประสิทธิภาพของการแก้ปัญหาการจัดวางมากยิ่งขึ้นและ รวมถึงการใช้วิธีการสร้างประชากรเริ่มต้นที่มีจำนวนน้อยแต่มีคุณภาพสูงเพื่อให้สามารถค้นหาคำตอบที่ดีได้ในระยะเวลาอันรวดเร็ว นอกจากนี้เราได้ประยุกต์หลักการออกแบบผังวงจรแบบลำดับชั้น (Hierarchical Layout Design Method) และ คอสฟังก์ชัน (Cost Function) หรือฟิตเนสฟังก์ชัน (Fitness Function) ที่เหมาะสมสำหรับปัญหาการจัดวางเซลเพื่อให้ได้ประสิทธิภาพมากยิ่งขึ้น เราได้ทดสอบ โมดิฟาย เจเนติค อัลกอริทึม กับวงจร MCNC standard cell benchmark ซึ่งได้ผลการทดสอบดีกว่า เจเนติค อัลกอริทึมแบบพื้นฐานและอัลกอริทึมอื่นๆ

Thesis Title	A Modified Genetic Algorithm for Standard Cell Placement
Student	Mr.Apichart Vasutapituks
Student ID.	42061126
Degree	Master of Engineering
Programme	Electrical Engineering
Year	2002
Thesis Advisor	Assoc.Prof.Bunjong Piyatamrong

ABSTRACT

This thesis presents modified genetic algorithm (MGA), which develops new genetic operator from genetic algorithm for use in standard cell layout placement optimization in VLSI physical design. MGA combines the knowledge of heuristic method and genetic algorithm into a new heuristic operator, greedy crossover and 3-Opt mutation. The new heuristic operator will help enhance the efficiency of the standard cell layout placement. To achieve the efficiency of the solution, the application of the high quality of initial small populations and the hierarchical layout design method with the good cost function is in need. MGA has yielded the overall results reported for set of MCNC standard cell benchmark circuits.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยดี ก็ด้วยคำแนะนำและให้คำปรึกษาความรู้ทางวิชาการที่สำคัญอย่างยิ่งพร้อมทั้งความห่วงใยจากท่าน รศ.บรรจง ปิยธำรง ซึ่งเป็นอาจารย์ผู้ควบคุมวิทยานิพนธ์ ผู้วิจัยมีความรู้สึกซาบซึ้งและสำนึกในความอนุเคราะห์และความช่วยเหลือจากท่านด้วยดีตลอดมา และขอกราบขอบพระคุณท่านเป็นอย่างสูง

ขอกราบขอบพระคุณบิดา มารดาและครอบครัวของผู้วิจัยที่ได้สนับสนุนตัวผู้วิจัยและเป็นกำลังใจมาโดยตลอด จนสำเร็จงานต่างๆมาได้ด้วยดี

ขอกราบขอบพระคุณครูบาอาจารย์ทุกท่านที่อบรมสั่งสอนผู้วิจัย ตั้งแต่ระดับอนุบาล ระดับประถมศึกษา ระดับมัธยมศึกษา ระดับอุดมศึกษา และอาจารย์ทุกท่าน ณ. สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง แห่งนี้ เป็นอย่างสูง

ขอขอบคุณเพื่อนที่เรียนและช่วยเหลือมาด้วยกันตลอด เป็นอย่างดี นายต่อพงศ์ รามศิริ

ขอขอบคุณผู้ที่ได้มีส่วนช่วยเหลือผู้วิจัยทั้งที่ทำงานและทุกท่านที่มิได้ออกนาม ณ.ที่นี้ด้วย

สุดท้ายนี้ ความรู้จากวิทยานิพนธ์นี้จะเป็นประโยชน์ไม่มากก็น้อยต่อผู้ที่สนใจความรู้ในสาขาแขนงนี้ หรือเป็นประโยชน์ต่อการพัฒนาความรู้ต่อไป ผู้วิจัยขอขอบพระคุณต่อผู้มีพระคุณทุกท่านเป็นอย่างสูง

อภิชาต วสุธาพิทักษ์

ธันวาคม 2545

สารบัญ

หน้า

สารบัญ (ต่อ)

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญภาพ

ภาพที่ หน้า

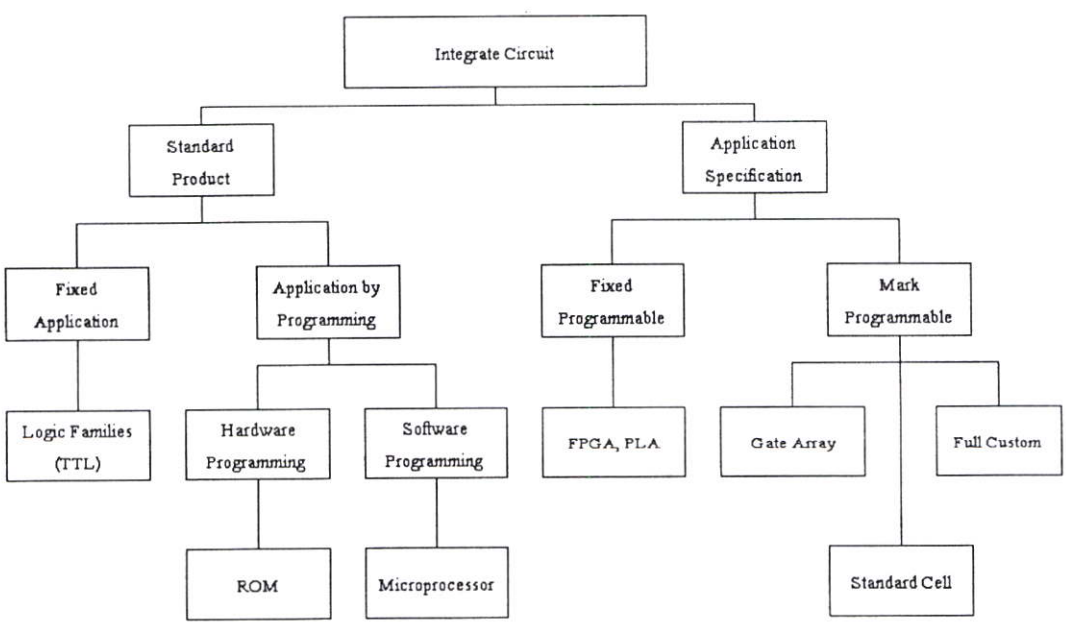
สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

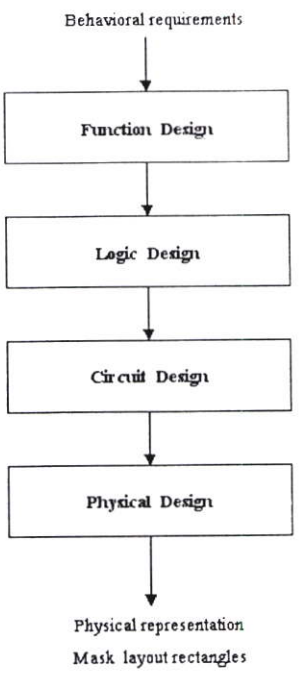
โลกของอิเล็กทรอนิกส์เริ่มต้นเมื่อนักวิทยาศาสตร์ได้ค้นพบการไหลไฟฟ้าในวงจรสามารถควบคุมได้ด้วยหลอดสุญญากาศ จึงมีการประดิษฐ์ หลอดสุญญากาศขึ้นเพื่อใช้กับเครื่องขยายเสียงวิทยุและหลอดไฟที่ใช้ในงานต่างๆ ในปี พ.ศ. 2483 นักวิทยาศาสตร์ได้ค้นพบว่าธาตุเจอร์เมเนียมเมื่อทำให้เป็นผลึกบริสุทธิ์และผ่านขั้นตอนบางอย่าง ก็สามารถทำงานได้อย่างเดียวกับหลอดหลอดสุญญากาศ ซึ่งนี่คือจุดเริ่มต้นของทรานซิสเตอร์ ต่อมาในปี พ.ศ. 2501 นักวิทยาศาสตร์ก็สามารถประดิษฐ์ทรานซิสเตอร์หลายๆตัว รวมกับชิ้นส่วนบางอย่างย่อขนาดให้เล็กลงและบรรจุรวมเป็นวงจรอยู่ในตัวเดียวกันได้ ซึ่งเรียกว่าวงจรรวมหรือไอซี (Integrated Circuit) ปัจจุบันเราได้พัฒนานำซิลิคอนมาทำไอซี แผ่นซิลิคอนเล็กๆ ที่เรียกว่าชิพ (Chip) ซึ่งชิ้นหนึ่งสามารถบรรจุ ทรานซิสเตอร์ได้นับแสนตัว จากจุดนี้ ทำให้ปัจจุบันสามารถสร้างอุปกรณ์อิเล็กทรอนิกส์ที่ใช้ในงานด้านต่างๆ ให้มีขนาดเล็กและกระทัดรัด ราคาถูก สะดวกในการพกพา และสามารถพัฒนาอุปกรณ์อิเล็กทรอนิกส์ต่างๆที่ใช้งานในชีวิตประจำวันได้อีกมากมาย เช่นโทรศัพท์เคลื่อนที่ เครื่องคอมพิวเตอร์แบบส่วนบุคคลและโน๊ตบุคส์



รูปที่ 1.1 ประเภทวงจรรวม

การพัฒนาการสร้างไอซีเริ่มตั้งแต่ไอซีที่สามารถบรรจุทรานซิสเตอร์เตอร์เพียงไม่กี่ตัว ซึ่งเรียกว่า เอสเอสไอ (Small-Scale Integration) เป็น เอ็มเอสไอ (Medium-Scale Integration) แอลเอสไอ (Large-Scale Integration) และ เป็นวีแอลเอสไอ (Very-Large Scale Integration) หรือวงจรรวมขนาดใหญ่ ในปัจจุบัน จากรูป 1.1 โดยทั่วไปวงจรรวมแบ่งเป็นสองประเภทคือ วงจรรวมมาตรฐาน (Standard Product) ที่ใช้ในงานด้านต่างๆ ซึ่งมีจำหน่ายในท้องตลาดทั่วไป วงจรรวมประเภทนี้แต่ละตัวจะมีเบอร์กำกับอยู่ อีกประเภทหนึ่งคือวงจรรวมเฉพาะกิจ (ASIC : Application Specification Integrated Circuit) ที่ใช้ในงานเฉพาะเจาะจงที่ไม่มีขายในท้องตลาด ซึ่งจะต้องออกแบบเอง สำหรับการออกแบบสามารถกระทำได้หลายวิธี เช่น FPGA, PAL, Gate Array, Standard Cell, และ Full Custom ขั้นตอนหลักๆในการผลิตวงจรรวมเฉพาะกิจ แบ่งได้เป็นสองขั้นตอนคือการออกแบบ (Designed) และการเจือสาร (Fabrication)

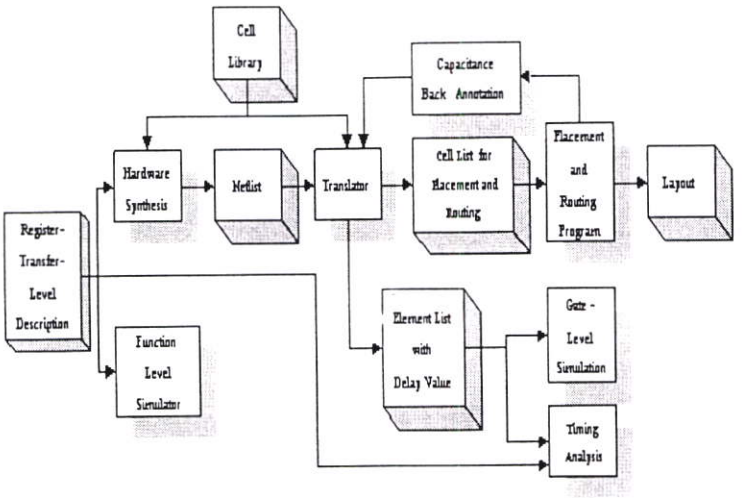
สำหรับการออกแบบวงจรรวมขนาดใหญ่นั้นเนื่องจากมีความซับซ้อนมากจึงได้แบ่งเป็นหลายขั้นตอน ในการออกแบบด้วยกันดังรูปที่ 1.2 คือการออกแบบระดับฟังก์ชัน (Function design) ระดับลอจิก (Logic Design) ระดับวงจร (Circuit Design) และระดับกายภาพ (Physical Design)



รูปที่ 1.2 ขั้นตอนการออกแบบวงจรรวม

ในงานวิจัยนี้จะเน้นการออกแบบการจัดวางตำแหน่ง (Placement) ของวงจรรวมในระดับกายภาพ โดยใช้แสตนดาร์เซล (Standard Cell) ซึ่งในระดับกายภาพจะแบ่งเป็นสองขั้นตอนที่สำคัญคือการจัดวางตำแหน่ง (Placement) และตามด้วยเชื่อมโยงสายสัญญาณ (Routing)

จากรูปที่ 1.3 ไฟล์ที่จะใช้ในงานวิจัยจะเป็นเน็ทริสไฟล์ (Netlist file) ซึ่งจะแสดงการเชื่อมต่อระหว่างเซล เราจะนำเน็ทริสไฟล์ มาทำการแปลงโดยรวมกับไลบรารีย์ (Library) เซลที่ได้ออกแบบไว้แล้วเป็นเป็นไฟล์ที่สามารถมาใช้ในงานการจัดวางตำแหน่งของเซลได้



รูปที่ 1.3 การออกแบบวงจรรวม

1.2 ความมุ่งหมายและวัตถุประสงค์

1. ศึกษาปัญหาการจัดวางตำแหน่งของแสตนดาร์ดเซลในระดับกายภาพ สำหรับการออกแบบวงจรรวมขนาดใหญ่
2. ศึกษาการใช้ เจเนติค อัลกอริทึม สำหรับแก้ปัญหาการจัดวางแสตนดาร์ดเซล
3. ประยุกต์ใช้ โมดิฟาย เจเนติค อัลกอริทึม ที่ผสมผสานความรู้วิธีฮิวรีสติคของการจัดวาง ซึ่งได้ฮิวรีสติค โอเปอเรเตอร์สำหรับแก้ปัญหาการจัดวางตำแหน่งของแสตนดาร์ดเซล
4. ศึกษาปัจจัยของประชากรเริ่มต้นที่มีจำนวนน้อยแต่มีคุณภาพสูงจะส่งผลต่อการแก้ปัญหาอย่างไร โดยจะเปรียบเทียบกับการสร้างประชากรเริ่มต้นแบบสุ่มตัวอย่าง
5. ประยุกต์หลักการออกแบบลำดับชั้น (Hierarchical Design) สำหรับแก้ปัญหาการจัดวางแสตนดาร์ดเซล
6. ทดสอบอัลกอริทึมการในการแก้ปัญหากับวงจร MCNC standard cell benchmark

1.3 สมมติฐานของการศึกษา

การแก้ปัญหาโดยใช้ โมดิฟาย เจเนติค อัลกอริทึมผนวกกับการจัดวางแบบลำดับชั้นจะช่วยเพิ่มประสิทธิภาพในการแก้ปัญหาการจัดวางตำแหน่งของแสตนดาร์ดเซลดีกว่าการใช้ เจเนติค

อัลกอริทึมแบบพื้นฐาน โดย โมดิฟาย เจเนติค โอเปอเรเตอร์ และการสร้างประชากรเริ่มต้นที่มีจำนวนน้อยแต่มีคุณภาพจะเป็นตัวหลักสำคัญในการเพิ่มประสิทธิภาพในการแก้ปัญหาการจัดวางตำแหน่งของแสตนดาร์ดเซล

1.4 ทฤษฎีหรือแนวความคิดที่ใช้ในการวิจัย

1.4.1 คณิตศาสตร์การจัดลำดับและการจัดหมู่

1.4.1.1หลักการนับ (Principles of Counting)

ในการทำงาน ถ้าแบ่งออกเป็น k ขั้นตอน โดยขั้นตอนแรกสามารถเลือกทำได้ m_1 วิธี และในขั้นตอนที่สองเลือกทำได้ m_2 วิธี และในขั้นตอนที่ k สามารถเลือกทำได้ m_k วิธี ดังนั้นจึงมีวิธีทั้งหมดที่สามารถทำงานนั้นได้เสร็จสิ้นเท่ากับ $m_1 x\ m_2 x\ m_3 ,\ldots,xm_k$ วิธี

1.4.1.2 การจัดลำดับ (Permutation)

ถ้ามีสิ่งของอยู่ n สิ่ง ซึ่งแต่ละสิ่งมีลักษณะแตกต่างกัน การจัดลำดับหมายถึงการนำสิ่งของ ทั้ง n สิ่งหรือ r สิ่ง (r n) มาจัดเรียงลำดับโดยถือลำดับเป็นสำคัญจะได้ $\frac{n!}{(n-r)!}$ วิธี

หรือ $P_{n,r} = \frac{n!}{(n-r)!}$ วิธี

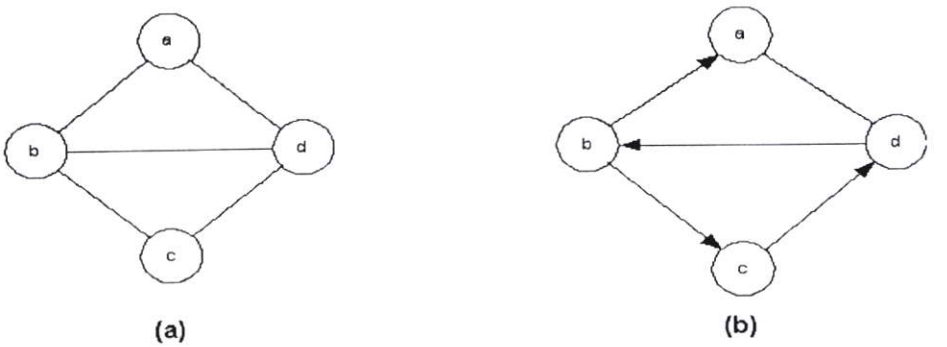
1.4.1.3 การจัดหมู่ (Combination)

ถ้ามีสิ่งของอยู่ n สิ่ง ซึ่งแต่ละสิ่งมีลักษณะแตกต่างกัน การจัดหมู่หมายถึงการนำสิ่งของ r สิ่ง (r<n) มาจัดเป็นกลุ่มหรือหมู่ โดยไม่คำนึงถึงลำดับ แต่เน้นเป็นสำคัญจะได้ $\frac{n!}{(n-r)!r!}$

วิธีหรือ $C_{n,r} = \binom{n}{r} = \frac{n!}{(n-r)!r!}$

1.4.2 ทฤษฎีกราฟ (Graph Theory)

กำหนดให้กราฟ G = (V,E) ประกอบด้วยคู่ลำดับของเซ็ต (V,E) โดยที่ V คือเซ็ตของจุด (Vertex) หรือเรียกว่า โหนด (Node) และ E คือเซ็ตของเส้นเชื่อม (Edge) ระหว่างคู่ของจุดในกราฟ ตัวอย่างเช่นกราฟ G = (V, E) ในรูปที่ 1.4 (a) มี V = {a, b, c, d} และ E = { {a,b}, {b,c}, {b,d}, {a,d}, {c,d}} เป็นกราฟที่ไม่มีทิศทาง (Undirected Graph) แทนเส้นเชื่อมของจุดด้วยเซ็ตของคู่จุดลำดับที่จุดไม่มีผลต่อกราฟ มีกราฟอีกประเภทหนึ่งเรียกว่ากราฟมีทิศทาง (Directed Graph) เช่น กราฟในรูปที่ 1.4 (b) จะสังเกตว่ามีลูกศรกำกับทิศทางของเส้นเชื่อมแต่ละเส้นในกราฟ ในกรณีนี้เราจะใช้คู่ลำดับของจุดแทนเส้นเชื่อมมีทิศทางเช่น (b,c)



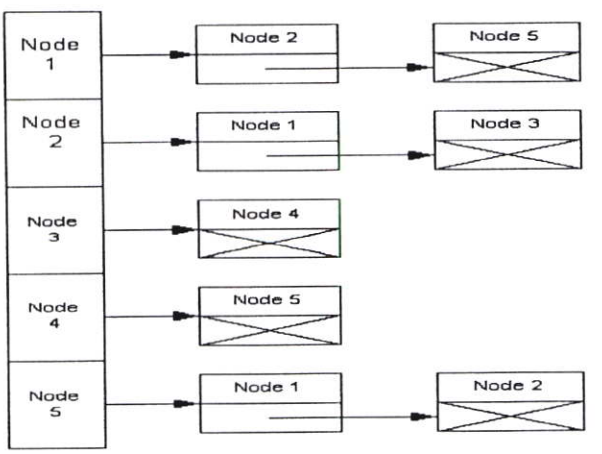
รูปที่ 1.4 กราฟแบบไม่มีทิศทางและแบบมีทิศทาง

1.4.3 ทฤษฎีความซับซ้อน (Complexity Theory)

ปัญหาใดก็ตามที่สามารถแก้ได้ภายในเวลาพหุนามเราเรียกว่าปัญหาประเภทนี้ว่าปัญหาพี (P-Problem : Polynomial Problem) ส่วนปัญหาใดก็ตามที่ไม่สามารถแก้ไขได้ภายในเวลาพหุนาม เราเรียกว่าปัญหาเอ็นพี (NP-Problem: Nondeterminitic Polynomial Problem) ส่วนปัญหาเอ็นพี (NP) ใดๆ ก็ตามที่สามารถแปลงจากปัญหาหนึ่งไปเป็นอีกปัญหาหนึ่งได้ภายในเวลาพหุนาม เราจะเรียกปัญหาประเภทนี้ว่าปัญหาเอ็นพีคือมพลีท (NP-Complete Problem)

1.4.4 โครงสร้างข้อมูลสำหรับกราฟ (Data Structure for Representtation of Graphs)

เราต้องการที่จะสร้างอักอริทึมที่เกี่ยวข้องกับกราฟจำเป็นอย่างยิ่งที่จะต้องมีการแทนข้อมูลของกราฟในระบบคอมพิวเตอร์ซึ่งโครงสร้างข้อมูลสำหรับกราฟมีความจำเป็นโครงสร้างข้อมูลข้อมูลของกราฟแบ่งเป็นสองลักษณะคือ แอดจาเซ็นซีเมกตริกซ์ (Adjacency Matrix) และ แอดจาเซ็นซีลิสท์ (Adjacency List) ซึ่งทั้งสองมีข้อดีและข้อเสียที่แตกต่างกันไป สำหรับในงานวิจัยนี้จะใช้แอดจาเซ็นซีลิสท์ ซึ่งมีความเร็วในการค้นหาข้อมูลมากกว่าแอดจาเซ็นซีเมกตริกซ์ จากรูปที่ 1.5 กราฟจะถูกแทนในลักษณะลิงลิสท์ (Liked List) ตั้งแต่โหนดแรกจนโหนดสุดท้าย แต่ละโหนดจะแสดงการเชื่อมต่อไปยังโหนดต่างๆที่มีเส้นเชื่อม เชื่อมต่อกันอยู่



รูปที่ 1.5 โครงสร้างข้อมูลแบบแอดจาเซ็นซีลิสท์

1.5 ขอบเขตการวิจัยและขั้นตอนการศึกษา

การวิจัยเริ่มจากการศึกษาภาพรวมของการออกแบบวงจรรวมในระดับกายภาพ จากนั้นจะศึกษาขั้นตอนการจัดวางตำแหน่ง โดยจะเน้นการศึกษา การใช้แสตนดาร์เซลสำหรับการจัดวางตำแหน่งในลักษณะการจัดวางแบบแถว (Row-Based Design)

ศึกษารูปแบบของเน็ทริส ไฟล์ (Netlist File) ที่จะนำมาใช้ในการจัดวางตำแหน่งของแสตนดาร์ดเซลดังรูปที่ 1.3 สำหรับเน็ทริสไฟล์ที่เราจะนำมาใช้งานเป็นเน็ทริสไฟล์แบบเยลเดสสคริปชั่น (Yal Description) ซึ่งเป็นวงจร MCNC standard cell benchmark ที่นิยมใช้เป็นวงจรมาตรฐานที่ใช้ในการเปรียบเทียบค่าต่างๆ

ศึกษาเจเนติค อัลกอริทึมแบบพื้นฐาน และประยุกต์นำมาใช้ในการแก้ปัญหาการจัดวางตำแหน่งของแสตนดาร์ดเซล ทำการปรับปรุงเจเนติค อัลกอริทึมแบบพื้นฐานเป็น
โมดิฟาย เจเนติค อัลกอริทึม ซึ่งเหมาะสำหรับแก้ปัญหาการจัดวางตำแหน่งของแสตนดาร์ดเซล ซึ่งปัญหาการจัดวางแสตนดาร์ดเซลของการออกแบบวงจรรวมในระดับกายภาพเป็นปัญหาประเภทเอ็นพีคอมพลีท (NP-Complete) โดยถ้าเมื่อมีจำนวนเซลของแสตนดาร์ดเซลจำนวนมาก ก็จะไม่สามารถหาคำตอบที่ดีที่สุดจาก คำตอบที่เป็นไปได้ภายในระยะเวลาที่เหมาะสม วิธีการแก้ปัญหาดังกล่าวโดยทั่วไปมีสองลักษณะคือ

วิธีแรกคือเริ่มจากการหาคำตอบเริ่มต้นก่อนและค่อยปรับปรุงคำตอบ จนได้คำตอบที่ดีที่สุด ซึ่งถ้านำมาใช้แก้ปัญหาที่เป็นปัญหาประเภทเอ็นพีคอมพลีทจะต้องใช้เวลาในการคำนวณมาก กว่าจะได้คำตอบที่ดีที่สุด

วิธีที่สองคือการหาคำตอบที่พอเหมาะ (Approximate Solution) โดยอัลกอริทึมที่จะใช้หาคำตอบจะต้องมีการประยุกต์ใช้วิธีการต่างๆเช่น วิธีฮิวรีสติค (Heuristic Method), สิมูเลเตท อเนลลิง (Simulated Annealing), นิวรอล เน็ตเวอร์ค (Neural Network), เจเนติค อัลกอริทึม (Genetic Algorithms) และอื่นๆ วิธีนี้จะมีประสิทธิภาพสูงกว่าวิธีแรก

ในงานวิจัยนี้ได้นำ โมดิฟาย เจเนติค อัลกอริทึม หรือเรียกว่า MGA มาประยุกต์ใช้กับการแก้ปัญหาในการจัดวางแสตนดาร์ดเซล โดย MGA เป็นการผสมผสานของวิธี เจเนติค อัลกอริทึมซึ่งมีประสิทธิภาพสูงกับวิธีฮิวรีสติค โดยปกติ เจเนติค อัลกอริทึม มีโอเปอเรเตอร์พื้นฐานที่สำคัญอยู่ 3 โอเปอเรเตอร์ คือ รีโปรดักชัน (Reproduction), ครอสโอเวอร์ (Crossover), และมูเตชั่น (Mutation) [11] สำหรับงานวิจัยนี้โอเปอเรเตอร์รีโปรดักชัน เราจะใช้ ทัวร์นาเมนท์ ซีเล็กชั่น (Tournament Selection) [1] ส่วนโอเปอเรเตอร์ครอสโอเวอร์และและมูเตชั่น เราได้พัฒนาขึ้น ซึ่งได้ ฮิวรีวติค โอเปอเตอร์ (Heuristic Operator) แบบใหม่คือกรีดี้ ครอสโอเวอร์ (Greedy Crossover) และ มูเตชั่นแบบสามโอพีทรี ฮิวรีสติค (Mutation Based on 3-Opt Heuristic) [1] สำหรับใช้ในการแก้ปัญหาการจัดวางเซลที่ยังไม่เคยมีใครใช้มาก่อน โดยปกติประชากรจะถูกสร้างโดยวิธีการสุ่มอย่างเดียวเพื่อนำมาเป็นคำตอบ ซึ่งจะมีจำนวนของคำตอบมากในโซลูชั่นสเปสซ์ (Solution Space) ทำให้การ

ค้นหาคำตอบที่ดีที่สุดใช้เวลานาน ในงานวิจัยเราจะเน้นการสร้างประชากรของคำตอบเริ่มต้นที่มีจำนวนน้อยแต่มีประสิทธิภาพสูง เพื่อนำไปสู่การค้นหาคำตอบที่ดีที่สุดได้ในระยะเวลาอันรวดเร็ว เพื่อเพิ่มประสิทธิภาพในการจัดวางได้ใช้หลักการออกแบบผังวงจรรวมแบบลำดับชั้น (Hierarchical Layout Design Method) [3] และคอสฟังก์ชั่นหรือฟิตเนสฟังก์ชั่นที่เหมาะสม มาประยุกต์กับการจัดวาง การวิจัยจะเน้นในการออกแบบวงจรรวมแสตนดาร์ดเซล โดยอาศัยโมดูลหรือเซลที่มีการออกแบบไว้แล้วในไลบรารีย์

ขั้นตอนสุดท้ายเป็นการสร้างโปรแกรมการจัดวางตำแหน่งของแสตนดาร์เซล ซึ่งใช้ภาษาจาวา (Java) และทดสอบกับวงจร MCNC standard cell benchmark [6] โดยศึกษาตัวแปรต่างๆ ที่จะมีผลต่อการทำงานของ โมดิฟาย เจเนติค อัลกอริทึม เช่น จำนวนของประชากร (Population Size) และจำนวนรุ่น (Generation)

บทที่ 2
การออกแบบวงจรรวมในระดับกายภาพของแสตนดาร์ดเซล

2.1 เจเนติค อัลกอริทึมกับการออกแบบเลย์เอาต์ของวงจรรวมขนาดใหญ่

ปี ค.ศ. 1975 John Holland ได้นำเสนอเจเนติค อัลกอริทึม ซึ่งเลียนแบบการผสมยีนมาใช้ในการแก้ปัญหาเกี่ยวกับการหาค่ามากที่สุดและน้อยที่สุด

ปี ค.ศ. 1986 Jame Cohoon ได้พัฒนาเจเนติค อัลกอริทึมสำหรับการจัดวางเซล เรียกว่า GENIE ซึ่งสามารถจะให้ผลของการจัดวางเซลได้อย่างดี

ปี ค.ศ. 1991 Jame Cohoon ได้พัฒนาเจเนติค อัลกอริทึมแบบกระจาย (Distributed Genetic Algorithms) สำหรับการออกแบบการวางผัง

ปี ค.ศ. 1991 Hulin ได้ค้นพบความแตกต่างของการเข้ารหัสคำตอบ (Coding Schemes) สำหรับปัญหาการจัดกลุ่มเซลของวงจร เพื่อที่จะค้นหาการเข้ารหัสคำตอบ ที่เหมาะสมสำหรับการจัดกลุ่มเซล (Cell Clustering) โดยได้เสนอ เจเนติค อัลกอริทึมของการจัดกลุ่มซึ่งใช้ Bit-Slice Component

ปี ค.ศ. 1991 Khushroo Sahookar และ Pinaki Mazumder ได้เสนอการประยุกต์ใช้เจเนติค อัลกอริทึมสำหรับการจัดวางตำแหน่งของแสตนดร์ดเซล ซึ่งใช้การแปลงโครโมโซม (Chromosome) ของเลย์เอาต์ โดยอัลกอริทึมจะทำงานในกลุ่มของจำนวนประชากรที่ค่าคงที่ การแปลงของโครโมโซมจะใช้โอเปอเรเตอร์ที่แตกต่างคนละชนิดกัน โดยครอสโอเวอร์โอเปอเรเตอร์จะสร้างลักษณะใหม่ของโคโมโซมเดิม ส่วนมูเตชั่นและอินเวอร์ชั่นโอเปอเรเตอร์จะถูกใช้สำหรับการเพิ่มค่าสำหรับการลู่เข้าหาคำตอบที่เหมาะสม และทำการเปรียบเทียบค่ากับการใช้ไซเคิลครอสโอเวอร์ (Cycle Crossover) และแปรเปลี่ยนตัวแปรอื่นๆ เพื่อหาค่าที่เมาะสม เช่น อัตราการเกิดครอสโอเวอร์ (Crossover Rate) อัตราการเกิดมูเตขั่น (Mutation Rate) และอัตราการอินเวอร์ชั่น (Inversion Rate)

ปี ค.ศ. 1993 Abhijit Dey ได้ประยุกต์เจเนติค อัลกอริทึม ที่อยู่ในรูป โพริช เอ็กซ์เพรสชั่น (Polish Expression) เป็นตัวแทนของโครโมโซมใช้สำหรับการจัดกลุ่ม โดยแทนโครโมโซมเป็นแบบไบนารี (Binary Representation)

ปี ค.ศ. 1993 Pinaki Mazumder ได้ประยุกต์เจเนติค อัลกอริทึม เป็นแบบโปรแกรมแบบขนานสำหรับแก้ปัญหาการจัดวางเซล ซึ่งช่วยลดเวลาการหาคำตอบได้

ปี ค.ศ. 1994 Volker Schnecke และ Oliver Vornberger ได้เสนอ เจเนติค อัลกอริทึม สำหรับการออกแบบการจัดวางของทราซิสเตอร์บนวงจรรวมขนาดใหญ่ เพื่อที่จะได้เลย์เอาต์ที่มีพื้นที่ขนาดเล็กที่สุด

ปี ค.ศ. 1996 Maurizio Rebaudengo และ Matteo Sonza Reorda ได้เสนอ เจเนติค อัลกอริทึม สำหรับการวางผัง (Floorplanning) โดยใช้ ฮิวริสติค โอเปอเรเตอร์ (Heuristic Operator) แบบต่างๆสำหรับปรับปรุงการวางผัง

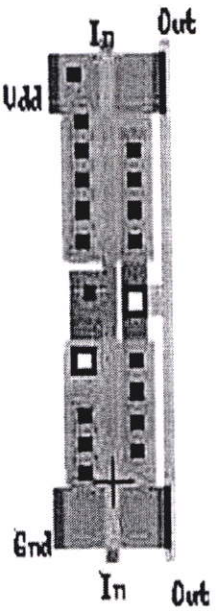
ปี ค.ศ. 1999 Pinaki Mazumder และ Elizabeth M. Rundnick ได้เสนอการใช้ เจเนติค อัลอริทึม ซึ่งผสมการใช้โอเปอเรตอร์แบบต่าๆ สำหรับการจัดวางตำแหน่งและเชื่อมโยงเซลในการออกแบบวงจรรวมขนาดใหญ่

2.2 การออกแบบระดับกายภาพของวงจรรวมแสตนดาร์ดเซล

งานวิจัยการออกแบบจะเริ่มจากการนำเน็ทริสไฟล์มาแปลงและจัดให้อยู่ของเซลริส (Cell list) ที่สามารถจะทำการจัดวางตำแหน่งเซลได้ ซึ่งการจัดวางจะรวมแสตนดาร์ดเซลที่ได้ออกแบบไว้แล้วที่เก็บไว้ในไลบรารีย์มาทำการออกแบบ ดังรูปที่ 1.3 ซึ่งแสตนดาร์ดเซลจะมีความสูงคงที่แต่ความยาวและความกว้างจะแปรเปลี่ยนไปในแต่ละเซล ตำแหน่งของอินพุทและเอาท์พุทจะอยู่ด้านบนและด้านล่างของเซล ส่วนพาวเวอร์ (Power) จะอยู่ส่วนบน และกราวด์ (Ground) จะอยู่ด้านล่างของเซลในระดับกายภาพวงจรรวมแสตนดาร์ดเซลใช้เทคนิคการออกแบบเซมิคัสตอม (Semi-Custom Design) หรือเซลเบส (Cell-Base Design) ที่ถูกออกแบบมาอย่างดีโดยผู้เชี่ยวชาญ ผู้ออกแบบเพียงแค่นำข้อมูลที่เป็นคุณลักษณะเฉพาะของเซลซึ่งจัดเก็บอยู่ในไลบรารีย์ เช่น ขนาดความกว้างและความยาวของเซลตำแหน่งอินพุทและเอาท์พุทหรือข้อมูลอื่นๆ ที่จำเป็นมาใช้ในการจัดวางตำแหน่ง

2.3 ประเภทของเซลที่ใช้ในวงจรรวมแสตนดาร์ดเซล

ฟิคเซล (Fixed-Cell) มีคุณสมบัติที่สำคัญคือจะมีขนาดความสูงคงที่แต่ความยาวเปลี่ยนแปลงไปตามชนิดของเซล มีความเหมาะสมต่อการจัดวางในลักษณะแถวเนื่องจากมีจุดเชื่อมต่อของพาวเวอร์ (Power) และกราวด์ (Ground) อยู่ภายในเซลทั้งด้านซ้ายและด้านขวา มีอินพุทและเอาต์พุทอยู่ด้านบนและด้านล่างของเซล ทำให้เราสามารถเลือกจุดเชื่อมต่อที่ด้านบนหรือด้านล่างของเซลก็ได้ ดังแสดงในรูปที่ 2.1 ซึ่งเป็นวงจรแสตนดาร์เซลของอินเวอร์เตอร์



รูปที่ 2.1 ตัวอย่างวงจรแสตนดาร์ดเซล

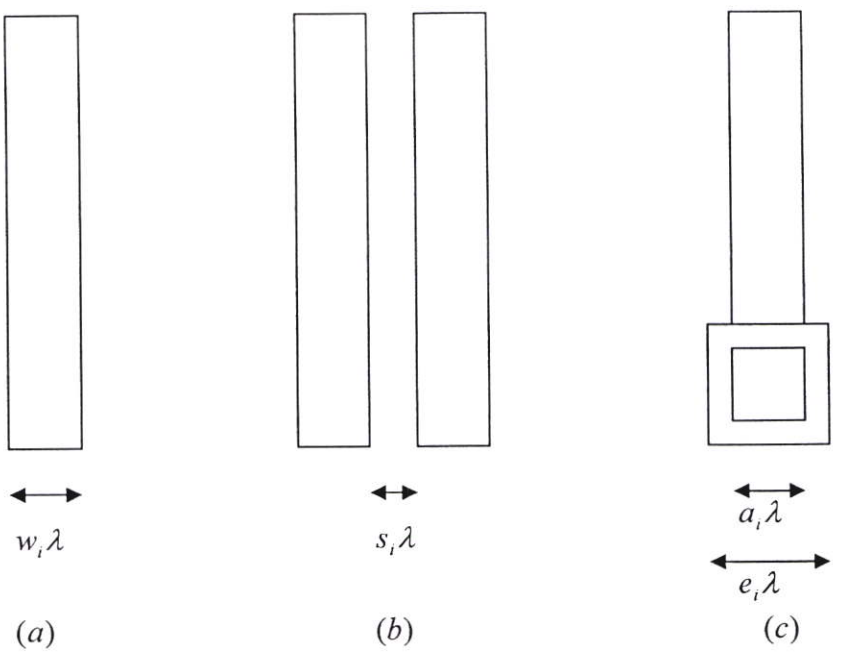
เซลประเภทนี้ไม่สามารถแก้ไขเปลี่ยนแปลงทางกายภาพของเซลได้ จึงเรียกเซลประเภทนี้ว่า Hard Coded Cell ยกตัวอย่างเช่น Gate, Latches, Adders เป็นต้น ส่วนใหญ่เป็นกลุ่มวงจรในระดับ SSI และ MSI ซึ่งถือว่าเป็นเซลพื้นฐาน (Basic Cell) หรือแสตนดาร์ดเซล สำหรับการออกแบบที่ต้องอาศัยความซับซ้อนของวงจรแล้วการใช้ฟิคเซลเพียงอย่างเดียวอาจจะ ไม่สามารถตอบสนองความต้องการบางประการของผู้ออกแบบได้ ดังนั้นอาจจะต้องอาศัยเซลขนาดใหญ่ซึ่งเราเรียกว่า แมคโครเซล (Macro Cell) ซึ่งขนาดของเซลไม่คงที่ ความสูงและความยาวเปลี่ยนแปลงไป เช่น Microprocessor, RAM, และ Rom สำหรับงานวิจัยนี้มิได้ศึกษาการออกแบบการที่ใช้ แมคโครเซลร่วมด้วย

2.4 กฎการออกแบบ (Design Rules)

ในระดับกายภาพของการออกแบบของวงจรรวม ต้องอาศัยกฎเกณฑ์ระหว่างนักออกแบบและโรงงานที่เจือสาร เพื่อให้สามารถผลิตวงจรรวมนั้นไปใช้งานได้จริง ขั้นตอนอยู่มากมาย ตั้งแต่การทำหน้ากาก (Mask) การทำกระบวนการโฟโตริซิส (Photo Resist) การเจือสารเข้าไปในแต่ละชั้นสาร เป็นต้น ในแต่ละขั้นตอนมักจะมีความผิดพลาดเกิดขึ้นเสมอ เช่น หน้ากากแต่ละชุดมีการเหลื่อมเกิดขึ้นที่ขอบของภาพ เมื่อนำวางซ้อนกันขอบของภาพที่ได้แพร่เลยออกจากขอบที่กำหนดไว้ ปัญหาเหล่านี้อาจทำให้วงจรรวมเกิดการลัดวงจร หรือเสียหายจนทำงานไม่ได้ ดังนั้นเพื่อให้

สามารถรับประกันได้ว่าวงจรรวมที่ผลิตขึ้นจะมีคุณสมบัติทางไฟฟ้า เช่นค่าความต้านทาน ค่าความจุไฟฟ้า ถูกต้องตามที่ต้องการจึงต้องมีการกำหนดมาตรฐานของแต่ละส่วนของวงจร ซึ่งรวมเข้าเป็นกฎเกณฑ์การออกแบบ แต่เนื่องจากเทคโนโลยีของขบวนการผลิตวงจรรวมได้พัฒนาไปอย่างรวดเร็วมาก ทำให้สามารถผลิตวงจรรวมที่มีขนาดเล็กลงได้ ดังนั้นขนาดมาตรฐานที่กำหนดไว้ในหน่วยวัดระยะทางจริง จึงต้องเปลี่ยนแปลงไปด้วย สร้างความยุ่งยากให้กับนักออกแบววงจรรวมที่ต้องคอยเปลี่ยนกฎการออกแบบ สำหรับเครื่องมือที่ช่วยออกแบบจะจัดให้อยู่ในรูปของหน่วยวัดระยะทางอิสระ ที่สามารถอ้างอิงถึงหน่วยวัดระยะทางจริงได้ และสามารถลดขนาดลงตามเทคโนโลยีที่เปลี่ยนแปลงไปได้ง่าย โดยไม่มีผลกระทบถึงการออกแบบเลย เช่น กฎการออกแบบของ Mead-Conway ที่ใช้เทคโนโลยีแบบ (NMOS) ซึ่งนิยมใช้กันแพร่หลาย และในภายหลังได้มีผู้ค้นคิดสร้างกฎการออกแบบในทำนองเดียวกันสำหรับเทคโนโลยีอื่นๆ อีกมากมาย เช่น MOSIS CMOS 2.0 เป็นต้น

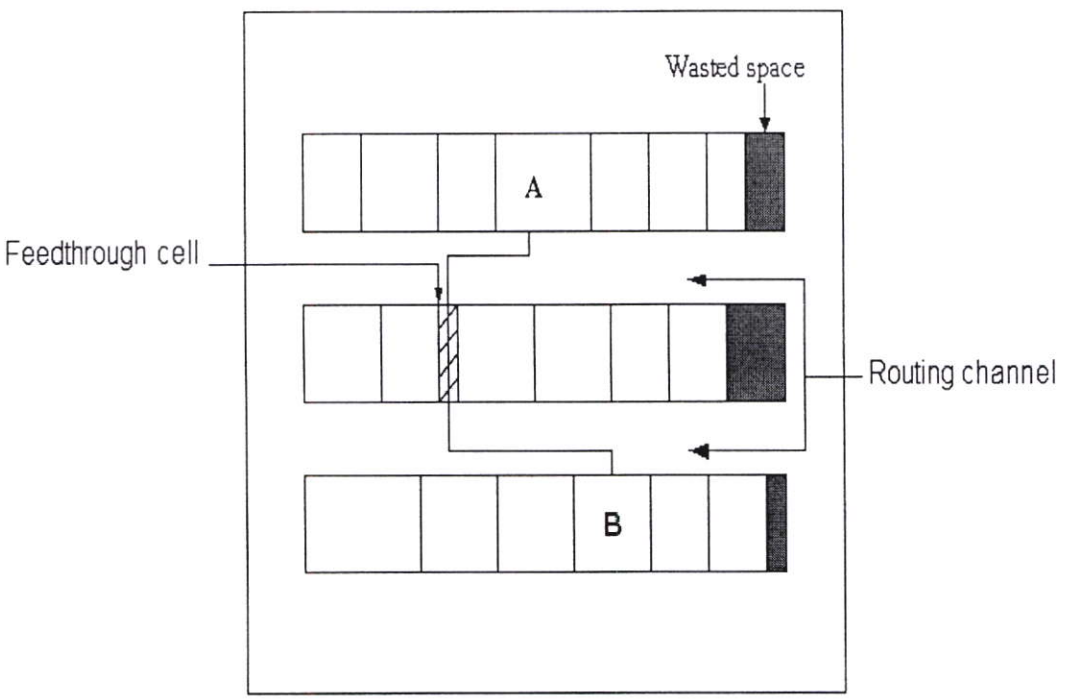
ดังนั้น การออกแบบในระดับกายภาพข้อมูลส่วนหนึ่งต้องอาศัยกฎการออกแบบ คือขนาดความกว้าง (Width) ดังรูป 2.2(a) ระยะห่าง (Separation) ดังรูป 2.2(b) ความยาวของส่วนเกิน (Extension) และความกว้างของคาบเกี่ยว (Overlap) ดังรูป 2.2(c) ที่โดยมีหน่วยวัดระยะทางอิสระตามกฎการออกแบบของ Mead-Conway คือ λ เป็นค่าที่ประมาณจากค่าผิดพลาดทางตำแหน่งที่มากที่สุดที่อาจเกิดขึ้นได้จากการใช้หน้ากากในขั้นตอนการผลิตวงจรรวม สำหรับค่า λ นั้นจะขึ้นกับความสามารถของเทคโนโลยีการผลิตวงจรรวม



รูปที่ 2.2 กฎการออกแบบของเลย์เอาต์

2.5 ลักษณะทางกายภาพวงจรรวมแสตนดาร์ดเซล

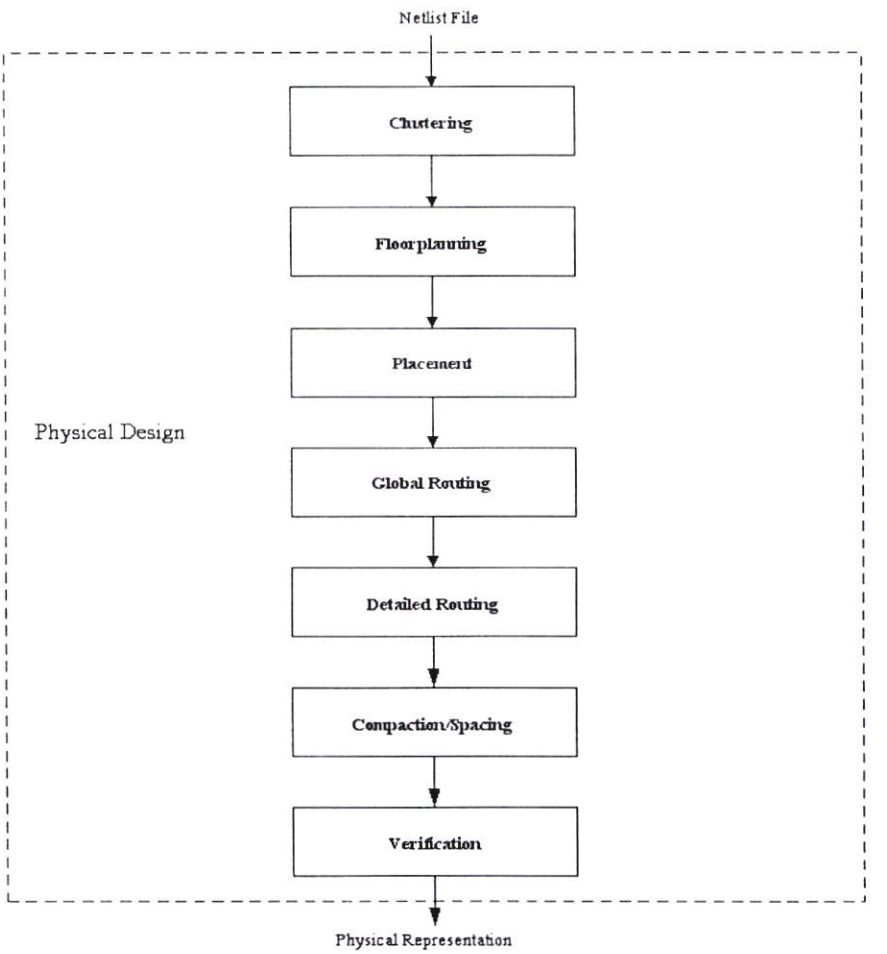
การวางผัง (Floorplan) ทางกายภาพของวงจรรวม แสตนดาร์ดเซลเลย์เอาต์ (Standard Cell Layout) ดังแสดงในรูปที่ 2.3 โดยยึดถือตามรูปแบบการวางผังของแสตนดาร์ดเซลเลเอาต์เท่านั้น เนื่องจากลักษณะทางเรขาคณิตของเซลประเภทนี้ มีขนาดความสูงคงที่ทุกเซล แต่จะมีความกว้างแตกต่างกันขึ้นอยู่กับประเภทของเซลและจุดเชื่อมต่อของอินพุทและเอาต์พุทอยู่ที่ด้านบนและด้านล่างของเซล ดังนั้นโครงสร้างทางกายภาพของวงจรรวมแสตนดาร์ดเซลที่มีความสูงคงที่ จึงใช้วิธีการจัดวางเซลในลักษณะแถวหรือ Row-Base เป็นหลักง่ายต่อการออกแบบ ส่วนพื้นที่ระหว่างแถวของเซลถูกใช้เป็นพื้นที่ในการเชื่อมโยงของสายสัญญาณจุดเชื่อมต่อที่เป็นอินพุทและเอาต์พุทของวงจรจะอยู่ที่ขอบเขตของเลย์เอาต์ทั้งสี่ด้านซึ่งเป็นโครงสร้างพื้นฐานทางกายภาพ วงจรรวม-แสตนดาร์ดเซล ถ้ามีการเชื่อมต่อสายสัญญาณข้ามระหว่างแถว เราอาจจะแทรกฟรีดธรูเซล (Feedthrough Cell) เข้าไปในแถวที่ถูกข้ามเพื่อเป็นทางผ่านของสายสัญญาณ การออกแบบจะให้มีพื้นที่ท้ายแถวที่เหลือว่างเปล่า (Waset Space) น้อยที่สุด เพื่อทำให้พื้นที่ของเลย์เอาต์มีขนาดเล็ก



รูปที่ 2.3 การออกแบบวงจรวมของแสตนดาร์ดเซล

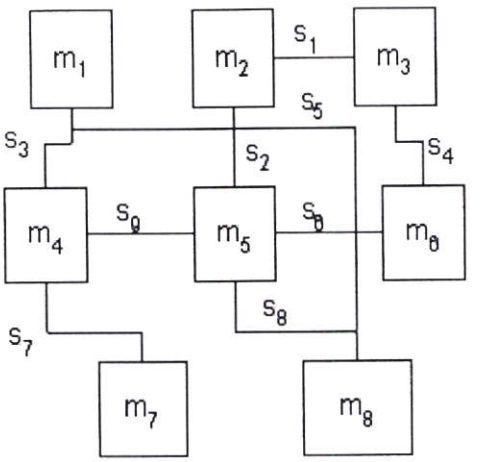
2.6 ขั้นตอนการออกแบบวงจรรวมระดับกายภาพ

การออกแบบวงจรรวมในระดับกายภาพ สำหรับวงจรรวมขนาดใหญ่จะแบ่งขั้นตอนออกเป็นหลายขั้นตอนดังนี้ดงแสดงในรูปที่ 2.4 โดยขั้นตอนแรกจะมีผลต่อขั้นตอนถัดมาเสมอ แต่สำหรับในการวิจัยนี้เป็นการวิจัยที่เน้นการจัดวางตำแหน่งสำหรับแสตนดาร์ดเซล ซึ่งการแก้ปัญหาการจัดวางเป็นการแก้ปัญหาที่อิสระต่อกัน แต่มีความสัมพันธ์กัน นั่นหมายถึงเราต้องทำการจัดวางก่อนแล้วจึงทำการเชื่อมโยงทีหลัง ผลที่ได้จากการจัดวางจึงมีอิทธิพลโดยตรงต่อการเชื่อมโยง สำหรับงานวิจัยนี้เราจะวิจัยเน้นเรื่อการจัดวางตำแหน่งของเซล ซึ่งประกอบด้วยสามขั้นตอนย่อย คือการจัดกลุ่ม (Clustering) การวางผัง (Floorplanning) และการจัดวาง (Placement)



รูปที่ 2.4 ขั้นตอนการออกแบบในระดับกายภาพ

2.7 ปัญหาการออกแบบทางกายภาพ



รูปที่ 2.5 การจัดวางเครือข่ายวงจร

กำหนดปัญหาในการจัดวางของแสตนดาร์ดเซลโดนกำหนดเครือข่ายวงจร (Circuit Network) ดังรูปที่ 2.5 ซึ่งประกอบด้วย $M = \{m_1, m_2, m_3,...,m_n\}$ เป็นเซตของเซล m_i $1 \leq i \leq n$ โดยแต่ละเซลมีความสูงคงที่ และเรียงกันเป็นแถว แต่ความกว้างขึ้นอยู่กับแต่ละเซล $S = \{s_1, s_2, s_3,...,s_h\}$ เป็นสายสัญญาณที่เชื่อมระหว่างเซล $L = \{l_1, l_2, l_3,...,l_p\}$ เป็นเซตของที่ตั้งของตำแหน่งแต่ละเซล โดย $l_k = (x,y)$, $1 \leq k \leq n$ เป็นคู่ลำดับแสดงตำแหน่งบนเลเอาต์ จาก เครือข่ายวงจร แทนด้วย Undirected Graph G(V,E) สำหรับใช้แก้ปัญหา โดย V เป็นเซตของโหนดซึ่งคือเซล $v_i \in V$, $V=\{v_1, v_2, v_3,...,v_n\}$ และ E เป็นเซ็ตของเส้นเชื่อมซึ่งแทนสายสัญญาณ $e_{ij}= (v_i,v_j) \in E$ ค่า c_{ij} เป็นค่า จำนวนเส้นเชื่อม (Weight) ของ e_{ij} โดย $i \neq j$ การจัดวางเป็นการวางเซลลงบนเลย์เอาต์ในลักษณะ 2 มิติ (2-D Placement)

บทที่ 3
เจเนติค อัลกอริทึม

3.1 เจเนติค อัลกอริทึม

เจเนติค อัลกอริทึม เป็นอัลกอริทึมที่ค้นหาคำตอบ (Search Algorithm) ที่มีประสิทธิภาพสูง ได้นำเสนอโดย John Holland ในปี ค.ศ. 1975 ร่วมกับเพื่อนและนักศึกษา ณ. มหาวิทยาลัยของมิชิแกน โดยอัลกอริทึมจะเลียนแบบวิวัฒนาการของสิ่งมีชีวิตในธรรมชาติ โดยในสิ่งมีชีวิต ภายในเซลจะมีส่วนที่เก็บลักษณะเฉพาะตัวของสิ่งมีชีวิตนั้นๆ ซึ่งเรียกว่าโครโมโซม (Chromosome) โดยโครโมโซมจะอยู่เป็นคู่ๆ ซึ่งจะมีอยู่หลายคู่ในแต่ละเซล ภายในโครโมโซมจะมีการเรียงตัวยาวของยีน (Gene) การเรียงตัวของยีนจะมีตำแหน่งเฉพาะสำหรับสิ่งมีชีวิตแต่ละชนิด โดยยีนจะเป็นตัวเก็บข้อมูลต่างๆของสิ่งมีชีวิต ซึ่งจะเป็นตัวถ่ายทอดลักษณะเด่นและด้อยไปสู่ลูกหลาน การผสมของยีนจะทำการ ครอสโอเวอร์ (Crossover) หรือ มูเตชั่น (Mutation) ซึ่งยีนในโคโมโซมของสิ่งมีชีวิตลูกจะมีการสับเปลี่ยนตำแหน่งการเรียงตัวของยีนใหม่ การทำครอสโอเวอร์ คือการตัดบางส่วนของกลุ่มยีนในโครโมโซมของพ่อแม่แล้วผสมกัน ซึ่งจะได้โครโมโซมใหม่สองโครโมโซม ส่วนการทำมูเตชั่น คือการเรียงสับเปลี่ยนตำแหน่งของยีนในโครโมโซม โดยการสุ่มเลือกตำแหน่งการสับเปลี่ยน ทำให้ได้การเรียงของยีนในโครโมโซมใหม่ที่อาจจะไม่เหมือนกับโคโมโซมของพ่อแม่ ดังนั้นประชากรของสิ่งมีชีวิตในรุ่นหลังจะไม่ค่อยได้รับยีนของประชากรรุ่นแรกๆที่ไม่ค่อยแข็งแรง ทำให้ประชากรในรุ่นหลังๆมีความแข็งแรงมากขึ้น

ได้มีการนำวิธีการดังกล่าวมาใช้ในการแก้ปัญหาต่างๆมากมาย เช่น การแก้ปัญหาทางเลือก (Combinatorial Optimization Problem) ซึ่งเป็นปัญหาที่จะต้องใช้เวลานานในการแก้ปัญหาเพื่อที่จะได้คำตอบที่ดีที่สุด ปัญหาหลายๆปัญหาในปัญหาประเภทนี้เป็นกลุ่มของปัญหาประเภทเอ็นพีคอมพลีท (NP-Complete) เช่น Bin packing, Traveling Salesman Problem (TSP), Multiprocessor Scheduling รวมทั้งปัญหาการจัดวางของแสตนดาร์ดเซล

3.1.1 เจเนติคอัลกอริทึมแบบพื้นฐาน (Simple Genetic Algorithm: SGA)

เจเนติคอัลกอริทึมแบบพื้นฐาน จะมีลำดับในการทำงานดังรูปที่ 3.1 โดยเริ่มต้นจากการสุ่มประชากร (Population) เริ่มต้นแล้วหาค่าแต่ละตัวของประชากร ทำการเลือกประชากร N_p ตัวเพื่อสร้างประชากรรุ่นใหม่ โดยการเลือกประชากรนั้นจะมีหลายวิธีเช่นอาจจะเลือกโดยขึ้นอยู่กับอัตราส่วนของค่า ฟิตเนส (Fitness) จากนั้นสุ่มจับคู่ประชากรที่เลือกเป็นประชากรรุ่นพ่อแม่เพื่อสร้างประชากรรุ่นลูกหลาน โดยทำการทำครอสโอเวอร์ สำหรับการที่จะนำประชากรมาทำการ

ครอสโอเวอร์ นั้นจะขึ้นอยู่กับค่า P_C ซึ่งคือค่า Probability ของการทำครอสโอเวอร์ถ้าประชากรตัวไหนที่ไม่ได้ทำการทำครอสโอเวอร์ประชากรตัวนั้นจะถูกแทนเป็นประชากรรุ่นถัดไปโดยอัตโนมัติ จากนั้นจะนำประชากรไปทำการมูเตชั่น ตามค่า P_M ซึ่งคือค่า Probability ของการการมูเตชั่น จากนั้นประชากรทุกตัวจะเป็นตัวแทนรุ่นต่อไป

1. Initialization

สุ่มประชากรรุ่นแรก

2. Selection

2.1 ใช้การสุ่มเลือกประชากร เช่นอาจใช้การเลือกแบบ Roulette Wheel แล้วแทนที่ลงในประชากรปัจจุบัน

2.2 สุ่มจับคู่ประชากรที่ได้

3. Crossover

พิจารณาอัตราการเกิดครอสโอเวอร์ว่าคู่ของโคโมโซมจากขั้นตอนที่ สองควรเกิดเกิดครอสโอเวอร์หรือไม่ ถ้าควรเกิดการเกิดครอสโอเวอร์ ให้สุ่มตำแหน่งที่จะทำเกิดครอสโอเวอร์แล้วสร้างโครโมโซมคู่ใหม่แทนที่โครโมโซมที่เป็นพ่อแม่ ถ้าไม่ควรเกิดเกิดครอสโอเวอร์ให้คงโครโมโซมคู่เดิมไว้

4. Mutation

สำหรับทุกโครโมโซมให้พิจารณาตามอัตราการเกิดมูเตชั่นว่าควรเกิดมูเตชั่นหรือไม่ ถ้าควรเกิด ให้สุ่มตำแหน่งที่จะเกิดการเปลี่ยนแปลง

5. Reproduction

เป็นการสร้างประชากรรุ่นใหม่ โดยทำตามขั้นตอนที่ 2-4 ซ้ำจนกระทั่งได้ผลตามเงื่อนไข

รูปที่ 3.1 เจเนติค อัลกอริทึมแบบพื้นฐาน

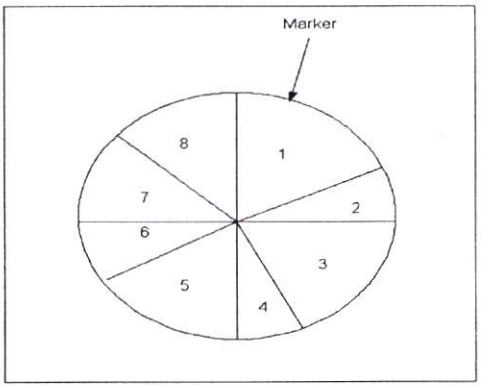
3.1.2 เจเนติค โอเปอเรเตอร์ (Genetic Operator)

เราจะกล่าวถึง เจเนติค โอเปอเรเตอร์ ของ เจเนติคอัลกอริทึมแบบพื้นฐาน เช่น การเลือก (Selection), ครอสโอเวอร์ และมูเตชั่น

3.1.2.1 การเลือก (Selection)

การเลือก จะมีหลายวิธีการเช่น Roulette Wheel Selection, Binary Tournament Selection สำหรับ Roulette Wheel Selection ดังรูปที่ 3.2 วงล้อจะเป็นตัวแทนของประชากรทุกตัว สำหรับ Slot ของวงล้อคือประชากรแต่ละตัว โดยขนาดของ Slot จะเป็นอัตราส่วนกับค่า Fitness

ของประชากรแต่ละตัว กรรมวิธีในการเลือกจะใช้วิธีการหมุนวงล้อ ถ้า Maker ตรงกับ Slot ของประชากรตัวนั้นจะได้รับการคัดเลือก ค่า Probability ของการที่จะถูกเลือกนั้นขึ้นกับอัตราส่วนของค่า Fitness สำหรับ Binary Tournament Selection ประชากร 2 ตัวจะถูกเลือกโดยการสุ่มและค่าของประชากรที่มีค่าดีกว่า 1 ตัว จะถูกเลือก ถ้า Binary Tournament Selection ถูกกระทำโดยไม่ได้แทนที่ของประชากร แล้วประชากร 2 ตัวจะถูกลบออกจากกลุ่มของประชากรทุกๆครั้งที่มีการเลือก โดยจำนวนของประชากรจะมีขนาดคงที่จากรุ่นหนึ่งไปยังอีกรุ่นหนึ่ง ประชากรเริ่มต้น (Original Population) จะถูกเรียกนำกลับคืนมา หลังจากประชากรใหม่ถูกแทนที่ครึ่งหนึ่ง ดังนั้นประชากรตัวที่ดีที่สุดอาจจะถูกเลือกถึง 2 ครั้ง และประชากรตัวที่แย่ที่สุดจะไม่ถูกเลือกเลยก็ได้ สำหรับ Binary Tournament Selection กับการแทนที่ของประชากร ค่าประชากร 2 ตัว จะถูกแทนที่ในประชากรรุ่นเดิมเพื่อสำหรับการเลือกครั้งต่อไป



รูปที่ 3.2 Roulette Wheel Selection

วัตถุประสงค์ของเจเนติค อัลกอริทึม คือการลู่เข้าสู่ค่าที่เป็นคำตอบที่ Optimal โดยการเลือกจะเป็นตัวที่กำหนดอัตราของการลู่เข้าสู่ค่าประชากรมีค่า Optimal การที่มีอัตราการเลือกสูงจะนำไปสู่การที่ประชากรลู่เข้าสู่ค่า Suboptimal

Roulette Wheel Selection จะมีการเน้นการเลือกแบบอัตราการเลือกสูงใน รุ่นต้นๆของประชากร โดยเฉพาะเมื่อประชากรมีค่า Fitness สูงกว่าประชากรตัวอื่นๆ สำหรับ Tournament Selectrion จะเน้นในประชากรรุ่นหลังๆ โดยเมื่อค่า Fitness ของประชากรไม่แตกต่างกันมากนัก ดังนั้น Roulette Wheel Selection จะลู่เข้าสู่ Suboptimal ถ้าประชากรมีค่า Fitness ที่แตกต่างกันมาก

3.1.2.2 ครอสโอเวอร์ (Crossover)

ประชากร 2 ตัวจะถูกเลือกเพื่อนำมาทำการครอสโอเวอร์ การครอสโอเวอร์ จะสร้างประชากรรุ่นลูกหลาน 2 ตัว โดยการครอสโอเวอร์ จะมีหลายแบบเช่น ครอสโอเวอร์ 1 จุด หรือ 2 จุด โครโมโซมจะถูกเลือกตำแหน่งโดยการสุ่มระหว่างความยาวของโครโมโซม และโครโมโซมของพ่อแม่จะถูกตัดที่จุด ตำแหน่งที่สุ่ม จากรูปที่ 3.3 โครโมโซมรุ่นลูกตัวแรก จะเหมือนกับ โครโมโซมพ่อแม่ตัวแรกจนถึงจุดตัดหลังจากจุดตัดนั้นจะเหมือนกับจะเหมือนกับโคโมโซมของลูกตัวที่สอง ใน Uniform Crossover จุดตัดของโครโมโซมจะถูกตัดตามค่า Probability

การครอสโอเวอร์ เป็นกระบวนการแบบสุ่ม ซึ่งผลของกระบวนการจะรวมโครโมโซมพ่อแม่ที่ไม่ดี ซึ่งจะส่งผลไปสู่โครโมโซมลูกหลานที่ไม่ดีด้วย แต่จะถูกกำจัดโดยโอเปอเรเตอร์ของการเลือก ในรุ่นถัดไปจำนวนของครอสโอเวอร์ จะถูกควบคุมโดย Crossover Probability (P_C) ซึ่งกำหนดโดยอัตราส่วนของจำนวนของประชากรรุ่นลูกที่จะถูกสร้างในแต่ละรุ่นต่อจำนวนของประชากรทั้งรุ่น ค่า Crossover Probability สูงๆ จะทำให้หาค่าที่ดีใน Solution space และลดโอกาสที่จะได้รับค่าที่ไม่ดีได้ สำหรับค่า Crossover Probability ต่ำๆ จะทำให้ สามารถใช้ประโยชน์จากประชากรที่มีค่า Fitness สูงๆ

Perent 1:	1 0 1 1 0 1 1 0 1	1 1 1 0 0 1 1 0 0
Parent 2:	0 0 1 1 0 1 1 0 0	1 0 0 1 0 1 0 0 0
Offspring 1:	1 0 1 1 0 1 1 0 1	1 0 0 1 0 1 0 0 0
Offspring 2:	0 0 1 1 0 1 1 0 0	1 1 1 0 0 1 1 0 0

รูปที่ 3.3 การครอสโอเวอร์แบบหนึ่งจุด

นอกจากนี้ยังมีการครอสโอเวอร์หลายแบบเช่น ออร์เดอร์ ครอสโอเวอร์ (Order Crossover), ไซเคิล ครอสโอเวอร์ (Cycle Crossover), พาร์เชียล แมป ครอสโอเวอร์ (Partially-Mapped Crossover) ซึ่งได้มีการประยุกต์กับการแก้ปัญหาหลายๆแบบ ในที่นี้จะยกตัวอย่างออร์เดอร์ ครอสโอเวอร์ดังรูปที่ 3.4 สมมติว่าโครโมโซมแรกเรียงตามลำดับดังนี้ ABCDEFGHI และโครโมโซมตัวที่สองคือ DEBAHGFIC ก่อนที่จะสลับตำแหน่ง เราจะกำหนดตำแหน่งของกลุ่มโครโมโซมในยีนทั้งสองตำแหน่งที่กำหนดจะแบ่งยีนแต่ละอันออกเป็น 2 กลุ่ม โครโมโซมย่อยตำแหน่งที่กำหนดนี้เป็นแบบสุ่ม ไม่มีความแน่นอน จากรูปที่ 3.4 แสดงตัวอย่างการแบ่งยีนทั้งสองออกเป็น 4 กลุ่ม โครโมโซม เส้นที่ลากผ่านยีนทั้งสองเป็นตัวกำหนดตำแหน่งการแบ่งกลุ่ม ยีนอันใหม่ของโครโมโซมลูกตัวแรกได้จากการเอากลุ่มของโครโมโซมทางซ้ายของเส้นตั้งที่แบ่งกลุ่มในยีนอัน

แรกมาใส่ในยีนอันใหม่ กลุ่มโครโมโซมทางซ้ายของเส้นแบ่งกลุ่มในยีนอันแรกคือ ABCDEF จากนั้นเราจะพิจารณาโครโมโซมแต่ละอันจากซ้ายไปขวาในยีนอันที่ 2 เราเอาโครโมโซมที่ยังไม่ได้ปรากฏในยีนอันที่ 3 มาใส่ในยีนอันที่ 3 ตามลำดับทีละตัวจากซ้ายไปขวาโครโมโซมแรกที่ไม่ได้อยู่ในกลุ่ม ABCDEF คือ I เราเอา I มาใส่ต่อจาก F จากนั้นเอา G และ H มาใส่ต่อจาก I เราข้ามโครโมโซม A, C, D ในยีนอันที่ 2 เพราะว่าปรากฏอยู่แล้วในยีนอันที่ 3 เราข้ามไปจนถึงโครโมโซม K และ J หยิบ K และ J ในยีนอันที่ 2 มาใส่ในยีนอันที่ 3 ถัดจาก H ตามลำดับ สำหรับโครโมโซมลูกตัวที่สองก็เกิดจากการสร้างเช่นเดียวกัน

Perent 1:	A B C D E F	G H I J K
Parent 2:	B E I G H F	A C D K J
Offspring 1:	A B C D E F	_ _ _ _ _
Offspring 2:	B E I G H F	_ _ _ _ _
Offspring 1:	A B C D E F	I G H K J
Offspring 2:	B E I G H F	A C D J K

รูปที่ 3.4 ออร์เดอร์ครอสโอเวอร์

3.1.2.3 มูเตชั่น (Mutation)

ประชากรรุ่นลูกที่ถูกสร้างใหม่จะถูกการมูเตชั่นตามค่า Mutation Probability (P_M) การมูเตชั่นจะกระทำโดยการสลับยีน ดังรูปที่ 3.4 ซึ่งจะสร้างการเปลี่ยนแปลงแบบสุ่มหลังจากการครอสโอเวอร์ P_M จะเป็นตัวกำหนดอัตราส่วนของจำนวนของประชากรรุ่นลูกที่จะถูกสร้างใหม่ในแต่ละรุ่นต่อจำนวนของประชากรทั้งรุ่น ถ้า P_M มีค่าน้อยประชากรรุ่นลูกจะมีอัตราการเปลี่ยนค่าต่ำ ถ้า P_M มีค่ามาก ประชากรรุ่นลูกจะมีการเปลี่ยนแปลงค่าสูง แต่ส่งผลต่อประชากรรุ่นพ่อแม่และรุ่นลูกจะมีความแตกต่างกันมาก ซึ่งถ้ามีการปรับค่าที่ดีขึ้นก็จะเหมาะสม แต่ถ้ามีการปรับค่าที่แย่ลงโดยประชากรรุ่นลูกมีค่าแย่แก่รุ่นพ่อแม่

Before Mutation: 1 1 0 1 0 0 0 1 0 0 1 1

After Mutation: 1 1 0 0 0 0 0 1 0 0 1 1

รูปที่ 3.4 มูเตชั่น

3.2 การออกแบบ ไฮบริด เจเนติค โอเปอเรเตอร์ สำหรับปัญหาการจัดวางของแสตนดาร์ดเซล

การพัฒนา โมดิฟาย เจเนติค โอเปอเรเตอร์ จะแบ่งเป็น การเข้ารหัสปัญหา การสร้างประชากรเริ่มต้น, กรีดี้ครอสโอเวอร์, และ มูเตชั่นแบบ สามโอพีทรีฮิวริสติค ดังต่อไปนี้

3.2.1 การเข้ารหัสปัญหา

ในหัวข้อ 2.7 ปัญหาในการจัดวางแสตนดาร์ดเซลหรือ คำตอบของปัญหาจะอยู่ในรูปของ $M = \{m_3, m_5, m_4, m_2, m_8, m_6\ m_1, m_7,\ldots m_n\}$ ซึ่งจะถูกเข้ารหัสเป็นโครโมโซมของเซล ซึ่งเซลจะถูกแทนเป็นตัวเลขจำนวนเต็มตามลำดับของเซลอยู่ในรูปของ {3, 5, 4, 2, 8, 6, 1, 7,...n} ดังรูปที่ 3.6 ในวิทยานิพนธ์จะกล่าวถึง $M = \{m_3, m_5, m_4, m_2, m_8, m_6\ m_1, m_7,\ldots m_n\}$ แทนโครโมโซมของเซล {3, 5, 4, 2, 8, 6, 1, 7,...n} เราจะเพิ่มความรู้ของวิธีการฮิวรีสติคในการจัดวางเซลให้กับโมดิฟาย เจเนติค โอเปอเรเตอร์ โดยจะจัดประชากรให้เซลเรียงกันขึ้นอยู่กับความสัมพันธ์ โดยถ้ามีความสัมพันธ์มากคือเซลมีสายสัญญาณเชื่อมกันจำนวนมาก เซลก็จะเรียงตัวอยู่ในกลุ่มเดียวกัน ดังรูปที่ 3.5 ซึ่งค่า k เป็นกลุ่มของเซล

m_3	m_5	m_4	m_2	m_8	m_6	m_1	m_7	$***m_{(n-1)}$	m_n
กลุ่ม V_1			กลุ่ม V_2		กลุ่ม V_3			$***$กลุ่ม V_k	

รูปที่ 3.5 การเรียงตัวของเซลที่สัมพันธ์กันในกลุ่มเดียวกัน

3	5	4	2	8	6	1	7	***n-1	n

รูปที่ 3.6 การเข้ารหัสเป็นโครโมโซม

3.2.2 การสร้างประชากรเริ่มต้น

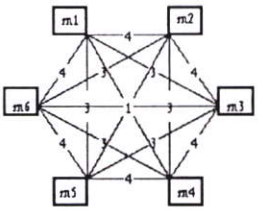
กำหนด k-nearest-neighbour subgraph เป็นเส้นกราฟที่เก็บทุกๆเส้นเชื่อม $e_{ij}= (v_i,v_j)$ แปลงให้อยู่ในรูปของเซลคือ $<m_i,m_j>$ ซึ่งเซล m_i มีสายสัญญาณที่เชื่อมต่อกับเซล m_j เราจะสร้าง k-nearest-neighbour subgraph ให้กับเซล และเริ่มต้นการสร้างประชากร โดยเลือกจากเซลที่เป็น k-nearest-neighbour subgraph เพื่อจัดกลุ่มเซลที่มีจำนวนสายสัญญาณเชื่อมต่อกันมากให้อยู่กลุ่มเดียวกัน คือกลุ่ม V_1, V_2,…,V_K ดังรูปที่ 2 เราทำการแบ่งกลุ่มโดยใช้ Undirected Graph G(V,E) ที่มีเงื่อนไขการแบ่งกลุ่มคือ $\cup_{n=1}^{k} V_n = V$ และ $\forall e_{ij}$ ซึ่ง $[v_i] \neq [v_j]$, โดยหาค่าต่ำสุดของ $C = \sum c_{ij}$ ซึ่ง C คือค่า Cost ของการแบ่งกลุ่ม, $V_n = [v_i]$ เป็น Subset ที่เก็บค่า v_i หรือเซล m_i

3.2.3 กรีดี้ครอสโอเวอร์ (Greedy Crossover)

หลังจากหัวข้อที่ 3.2.2 แล้ว ให้ทำการ สุ่มเลือกประชากรในรุ่น ซึ่งมีค่า Cost ดีที่สุด เพื่อเป็นประชากรรุ่นพ่อแม่ 2 ประชากร ถูกแทนด้วย **P1** และ **P2** ประชากรรุ่นลูกจะถูกสร้างโดยเริ่มต้นจากการสุ่มเช่นสมมติเป็นเซล m_c และตรวจสอบว่ามีเส้นเชื่อมต่อไปยังเซล m_c หรือจากเซล m_c มีสายสัญญาณเชื่อมต่อไปยังเซลอื่นๆหรือไม่ เช่นมีเส้นเชื่อม $<m_c,m_{c(right)}>$ หรือ เส้นเชื่อม $<m_{c(left)},m_c>$ ที่จะมีอยู่ในประชากรทั้ง **P1** และ **P2** ก็จะถูกเลือก ถ้าไม่มี เราจะเปรียบเทียบเส้นเชื่อมทางด้านขวาของเซล m_c ในแต่ละ **P1** และ **P2** โดยเส้นเชื่อมที่มีจำนวนสายสัญญาณที่มากจะถูกเลือก นอกจากมันจะเป็นเส้นเชื่อมแบบ cycle ซึ่งจะไม่ถูกเลือก แต่เราก็จะเลือกเส้นเชื่อมที่มีจำนวนสายสัญญาณที่น้อยกว่า ถ้าเส้นเชื่อมที่มีจำนวนสายสัญญาณที่น้อยกว่าเป็น cycle แล้วเราจะเลือกเส้นเชื่อมถัดไป ประชากรรุ่นลูกตัวที่สอง จะถูกสร้างในลักษณะที่คล้ายกัน แต่จะใช้เส้นเชื่อมทางด้านซ้ายของเซล m_c แทนการใช้เส้นเชื่อมทางด้านขวา

ในรูปที่ 3.7 เป็นตัวอย่างกรีดี้ครอสโอเวอร์ โดยประชากรรุ่นพ่อแม่ 2 ประชากรถูกแทนด้วย **P1** และ **P2** ประชากรรุ่นลูกจะถูกสร้าง โดยเริ่มต้นเราสุ่มได้จากเซล m_6 ดังนั้นเส้นเชื่อมด้านขวาคือ $<m_6,m_1>$ ในประชากร **P1** และ $<m_6,m_5,>$ ในประชากร **P2** จำนวนสายสัญญาณระหว่างเซลแสดงว่า $<m_6,m_1>$ และ $<m_6,m_5>$ มีจำนวนสายสัญญาณที่เท่ากัน ถ้าเป็นเช่นนั้นเส้นเชื่อมใน P1 จะถูกเลือกก่อนสำหรับแปลงไปเป็นประชากรรุ่นลูกตัวที่ 1 และเส้นเชื่อมใน **P2** จะถูกนำไปสร้างเป็นประชากรรุ่นลูกตัวที่ 2 อย่างไรก็ตามในตัวอย่างเส้นเชื่อม $<m_6,m_5>$ คือเส้นเชื่อมปกติในประชากรรุ่นพ่อแม่ทั้งสองประชากร คือทั้ง **P1** และ **P2** ดังนั้น $<m_6,m_5>$ จะถูกเลือก ถัดจากเซล m_5จะมีเส้นเชื่อมด้านขวา 2 เส้นเชื่อม คือ $<m_5,m_6>$ และ $<m_5,m_4>$ เวลานี้เราเลือก $<m_5,m_4>$ เพราะว่าเส้นเชื่อม $<m_5,m_6>$ จะสร้าง cycle จากเซล m_4 เส้นเชื่อมคือ $<m_4,m_2>$และ $<m_4,m_1>$ จำนวนสายสัญญาณที่มากกว่าที่ถูกเลือกคือ $<m_4,m_2>$ จากนั้นจากเซล m_2 ซึ่งมีเส้นเชื่อมคือ $<m_2, m_5>$ และ $<m_2,m_3>$ จำนวนสายสัญญาณที่มากกว่าที่ถูกเลือกคือ $<m_2,m_3>$ ในที่สุดเรามีเฉพาะ

เซล m_1 ที่เหลืออยู่ สำหรับประชากรรุ่นลูกตัวที่สองจะใช้เส้นเชื่อมทางด้านซ้าย จะเห็นว่าเส้นเชื่อมที่ดีจะถูกถ่ายทอดจากพ่อแม่ไปสู่รุ่นลูก



จำนวนสายสัญญาณระหว่างเซล

m_1	0					
m_2	4	0				
m_3	3	4	0			
m_4	1	3	4	0		
m_5	3	1	3	4	0	
m_6	4	3	1	3	4	0
เซล	m_1	m_2	m_3	m_4	m_5	m_6

Parent 1 (P1) : (m_1 m_3 m_4 m_2 m_5 m_6)

Parent 2 (P2) : (m_1 m_2 m_3 m_6 m_5 m_4)

Offspring 1 (ถ้าสุ่มที่เซล 6)

(m_6 m_5 m_4 m_2 m_3 m_1) → (m_1 m_3 m_2 m_4 m_5 m_6)

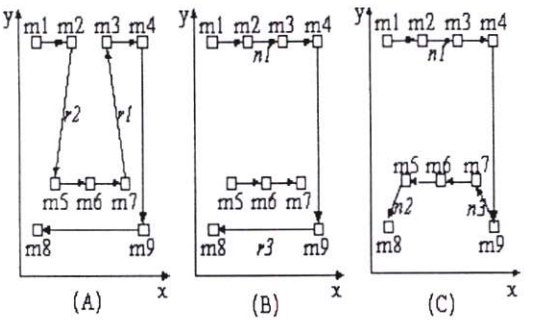
Offspring 2 (ถ้าสุ่มที่เซล 3)

(m_3 m_2 m_1 m_6 m_5 m_4) → (m_1 m_6 m_5 m_4 m_3 m_2)

m_6	<m_6,m_1> **P1**				
	<m_6,m_5> P2	<m_5,m_6> **P1**	→ cycle		
		<m_5,m_4> P2	<m_4,m_2> **P1**	<m_2,m_5> **P1**	
			<m_4,m_1> **P2**	<m_2,m_3> **P2**	m_1

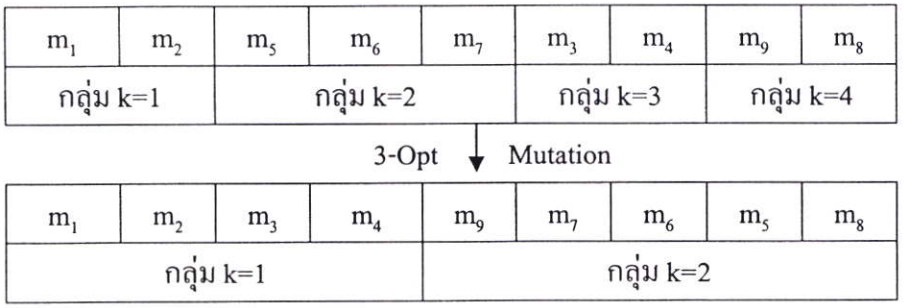
รูปที่ 3.7 ตัวอย่าง กรีดี้ครอสโอเวอร์

3.2.3 มูเตชันแบบสามโอพีทรีฮิวรีสติค (Mutations Based on 3-Opt Heuristic)



รูปที่ 3.8 การมูเตชั่นแบบสามโอพีทรีฮิวรีสติค

การมูเตชั่นเป็นการป้องกันการเข้าสู่ Local Optimal เร็วเกินไป จากรูปที่ 3.8(A) เป็นการเลือกประชากรของ คำตอบ สองตัวจากขั้นตอน การเลือกและการสร้างประชากรโดยถ้าประชากรทั้งสองที่เลือกมีคำตอบเหมือนกัน จะถูกนำมาทำการมูเตชั่น แทน กรีดี้ครอสโอเวอร์โดยจะเริ่มจากการจัดวางเซลตามแกน x และ แกน y โดยพิจารณาถึง k-nearest neighbour subgraph ซึ่งเปรียบเสมือนเป็นการวางเลเอาต์ จากรูปที่ 3.8(A) เป็นการเรียงตัวของคำตอบที่ไม่ถูกต้อง ส่วนในรูปที่ 3.8(B) เป็นการ 3-Opt Move โดยจะลบเส้นเชื่อม 3 เส้นเชื่อม และ เพิ่มเส้นเชื่อมอีก 3 เส้นเชื่อม จากรูปที่ 3.8(C) เส้นเชื่อมที่จะถูกลบ 3 เส้น คือ r_1, r_2, r_3, และเพิ่มเส้นเชื่อมใหม่คือ n_1, n_2, และ n_3 จากรูปที่ 3.8(C) และ รูปที่ 3.9 แสดงให้เห็นว่าการเรียงตัวของคำตอบหลังการมูเตชั่น ทำให้คำตอบถูกต้องมากยิ่งขึ้นตามความเป็นจริงและส่งผลให้การแบ่งกลุ่มถูกต้องด้วย

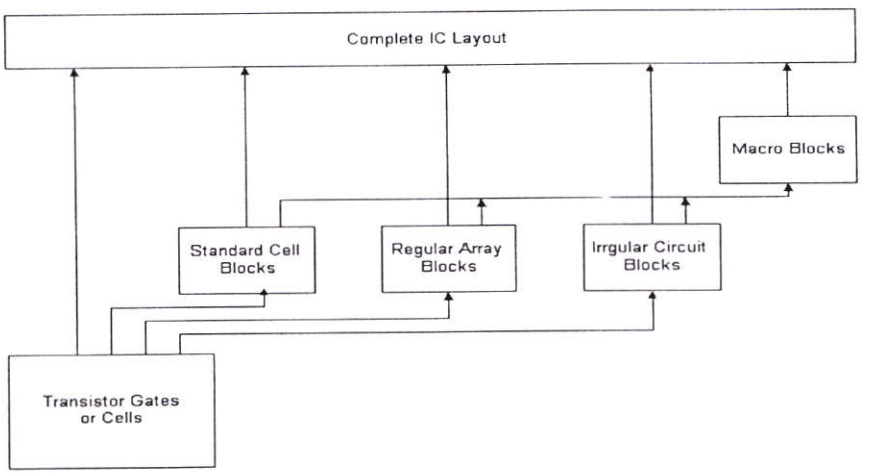


รูปที่ 3.9 ประชากรก่อนและหลังการมูเตชั่นแบบสามโอพีทรีฮิวรีสติค

บทที่ 4
การจัดวางแสตนดาร์ดเซล

4.1 การจัดวางแบบลำดับชั้น (Hierarchical Placement)

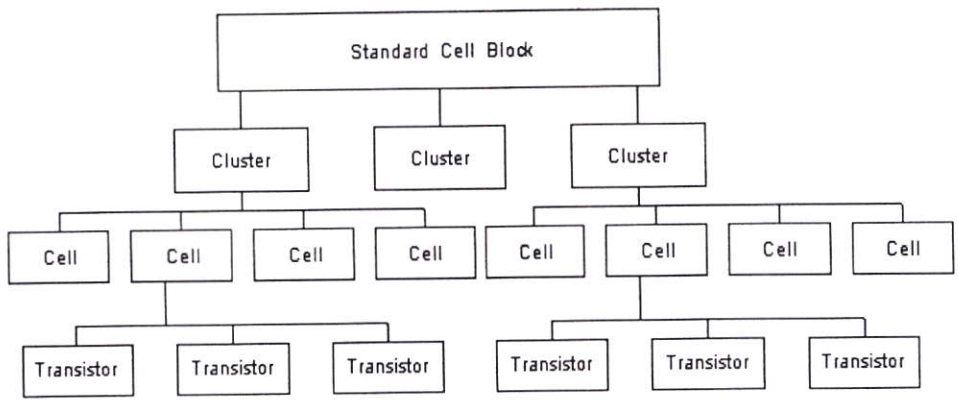
การออกแบบวงจรรวมแบบลำดับชั้น (Hierachical Designs) เริ่มจากการทำจัดวาง ในระดับฟังก์ชันบล็อก ก่อนเพื่อเป็นการหาขนาดของพื้นที่โดยรวมของแต่ละบล็อก จากนั้นจึงจะจัดวางในระดับ Macro Block และบล็อกชนิดต่างๆตามลำดับดังรูปที่ 4.1 โดยหลักการพื้นฐานการออกแบบลำดับชั้นคือการแบ่งแล้วเอาชนะ (Devided and Conquer)



รูปที่ 4.1 การออกแบบวงจรรวมแบบลำดับชั้น

4.2 การจัดวางแบบลำดับชั้น (Hierarchical Placement)

สำหรับงานวิจัยนี้จะเน้นการออกแบบแสตนดาร์ดเซลเพียงอย่างเดียว ดังนั้นปัญหาจะอยู่ในรูปของ Standard Cell Block หรือกลุ่มของเซล ดังรูปที่ 4.2 การที่จะจัดวางกลุ่มของเซลที่มีจำนวนมากเพื่อให้ได้พื้นที่เลเอาต์น้อยที่สุดนั้นเราแบ่งเป็น 3 ขั้นตอนโดยขั้นตอนแรก คือการจัดกลุ่มเซลให้เป็นกลุ่มย่อยหรือเรียกว่า Cluster โดยคำนึงถึงความเหมาะสมสำหรับการออกแบบเลเอาต์ที่เป็นแถว (Row-Based) ขั้นตอนที่สองคือการนำเซลที่ถูกจัดกลุ่มแล้วมาทำการวางผัง โดยพิจารณาถึงเนื้อที่ของเลเอาต์ ส่วนในขั้นตอนสุดท้ายเป็นการวางเซลภายในกลุ่มโดยไม่มีการซ้อนทับระหว่างเซลเพื่อกำหนดตำแหน่งของเซลภายในแถว



รูปที่ 4.2 การจัดวางแบบลำดับชั้น

4.2.1 การจัดกลุ่มเซล (Cell Clustering)

การจัดกลุ่มเป็นการแบ่งกลุ่ม (Partitioning) เซล โดยเราจะใช้หลักการพื้นฐานของคัทไลน์ (Cut Line) แสดงในรูปที่ 4.3 โดยพิจารณาถึง

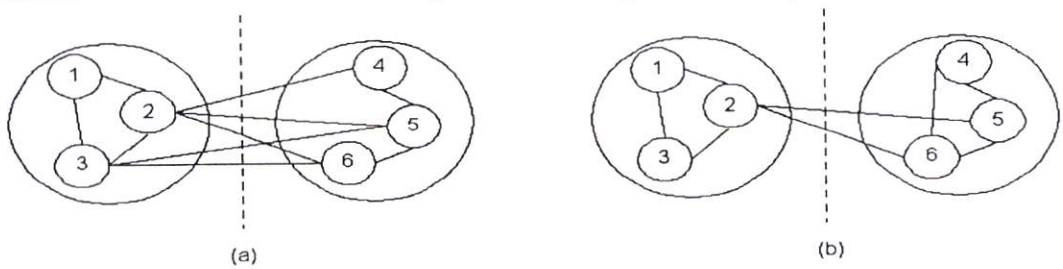
1. ความสัมพันธ์ โดยจัดเซลที่มีความสัมพันธ์กันให้อยู่ภายในกลุ่มเดียวกัน

2. เราได้ใช้การแบ่งกลุ่มแบบ Two-Way Partitioning มาประยุกต์กับการแบ่งกลุ่มในลักษณะ Multiway Partitioning เพื่อให้จำนวนกลุ่มมีมากกว่า 2 กลุ่มขึ้นไป โดยทั่วไปการแบ่งกลุ่มมีอยู่ 2วิธี

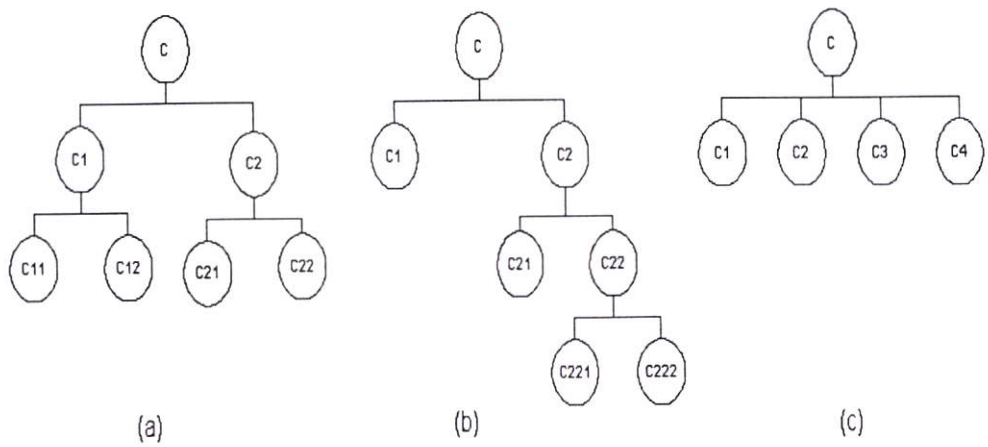
วิธีแรกแสดงในรูปที่ 4.4(a) แบ่งกลุ่มของเซลออกเป็น 2 กลุ่มแต่ละกลุ่มถูกแบ่งออกไปอีกจนได้จำนวนกลุ่มตามต้องการ

วิธีที่สองคือแบ่งกลุ่มของเซลออกเป็น 2 กลุ่มแล้วเลือกแบ่งกลุ่มที่มีขนาดใหญ่กว่าจนได้จำนวนกลุ่มตามที่ต้องการ ดังแสดงในรูปที่ 4.4(b) ข้อเสียก็คือถ้าจำนวนเซลมากขึ้นจำเป็นต้องใช้เวลาในการแบ่งมากขึ้น

เราได้ใช้วิธีที่เหมาะสมสำหรับการจัดวางแสตนดาร์ดเซลโดยการแบ่งกลุ่มออกเท่ากับจำนวนที่กำหนดตั้งแต่เริ่มต้นแล้วใช้ MGA แก้ปัญหาในการแบ่งกลุ่ม ดังแสดงในรูปที่ 4.4(c)



รูปที่ 4.3 การแบ่งกลุ่มโดยใช้คัทไลน์



รูปที่ 4.4 การแบ่งกลุ่มในลักษณะ Multiway Partition

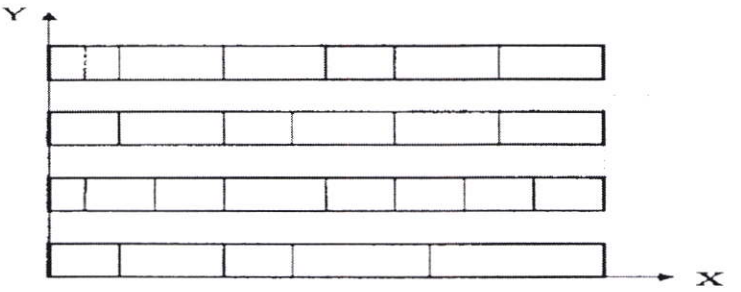
จาก Undirected Graph G(V,E) เราแบ่งกลุ่มเซลทั้งหมดออกเป็น R กลุ่ม โดยใช้ Cost function

$$G = \alpha \sum_{j=1, j\neq i+1}^{R} CF_{ij} + \beta \sum_{i=1}^{N} CPL_i \qquad (1)$$

เมื่อ α,β เป็นค่าคงที่, *CF*=Cut Feed, *CPL*=Critical Path Length

4.2.2 การวางผังเซล (Cell Floorplanning)

เป็นการนำเซลที่ถูกแบ่งกลุ่มแล้วมาจัดวางในลักษณะแถวดังรูปที่ 4.5 โดยเซลที่ถูกแบ่งให้อยู่ในกลุ่มเดียวกันจะจัดให้อยู่ในแถวเดียวกัน และปรับความกว้างของแถวให้ใกล้เคียงกัน สำหรับเซลที่ไม่มีฟีดธรูพิน (Feed through pin) จะต้องแทรกฟีดธรู (Feed through) เซลเข้าไปในแถว แต่จำเป็นต้องทราบทางเดินของสายสัญญาณเสียก่อน โดยใช้มินิมัมสแพนนิงทรี (Minimum Spanning Tree) [5] จึงจะทราบจำนวนฟีดธรูเซลโดยประมาณ และสามารถควบคุมพื้นที่ท้ายแถวได้อย่างมีประสิทธิภาพ ดังนั้นขั้นตอนนี้จึงเป็นการจัดวางผสมกับการทำ Global Route แบบหยาบ เพื่อให้ความกว้างของเลเอาต์มีค่าใกล้เคียงกับ Aspect Ratio



รูปที่ 4.5 การวางผังเซลในแนวแกน x และแกน y

กำหนดให้ R เป็นเซ็ตของกลุ่ม $r_i \in R$, $R=\{r_1, r_2, r_3,...,r_n\}$ และ M_{ri} เป็นเซ็ตของเซลในแถวใดๆ $m_i \in M_{ri}$, $M_{ri}=\{m_1, m_2, m_3,...,m_n\}$ ให้ W เป็นเซ็ตของความกว้างของแถว $w_i \in W_{ri}$, $W_{ri}=\{w_1, w_2, w_3,...,w_n\}$ วางเซลลงบนแถวแล้วให้มีเนื้อที่ว่างเปล่าน้อยที่สุด กำหนด Cost function

$$D = \sum_{i=1}^{r} S_i \qquad (2)$$

โดย D เป็นเนื้อที่ว่างเปล่า, เมื่อ $s_i = |w_i - \overline{W}|$

$W_i = \sum_{i=1}^{n} w_i$ เป็นความยาวรวมของเซลที่อยู่แถวที่ i

$\overline{W} = \sum_{i=1}^{r} W_i$ เป็นความกว้างที่ได้จากการประมาณ

4.2.3 การวางเซล (Cell Final Placement)

เป็นการหาตำแหน่งที่ตั้งของเซลบนเลย์เอาต์ เพื่อให้การเชื่อมสายสัญญาณสั้นที่สุด สำหรับแสตนดาร์ดเซล i ใดๆ จะมีความกว้าง b_i และความสูง c_i ซึ่งจะมีพื้นที่ $a_i = b_i.c_i$ การจัดวางเซลจะถูกกำหนดตำแหน่ง $(x_1,y_i),\ldots,(x_n,y_n)$ ซึ่ง (x_i,y_i) จะเป็นจุดศูนย์กลางเซล i การประมาณความยาวของสายสัญญาณโดยใช้ Rectilinear (Manhattan) Distance [7] $D_{ij} = |x_i - x_j| + |y_i - y_j|$, D_{ij} คือระยะระหว่างจุดกึ่งกลางของเซล i และ j ที่เชื่อมสายสัญญาณ, N_{ij} คือจำนวนสายสัญญาณระหว่างเซล i และ j, ω_{ij} คือ weight ของสายสัญญาณ N_{ij} นอกจากนี้จะมีสายสัญญาณจากเซล i ไปยังพิน (pin) อินพุทและเอาท์พุทถูกแทนด้วย $D_{i,out}$, $D_{i,out} = \min(x_i, y_i, h - x_i, h - y_i)$ โดย ขนาดพื้นที่พิน=hxh, $h = \sqrt{\sum_{i=1}^{n} a_i}$ ดังนั้น Cost function ของผลรวมความยาวของสายสัญญาณ คือ

$$L = \sum_{i=1}^{n} \sum_{j=1}^{i-1} \omega_{ij} N_{ij} D_{ij} + \sum_{i=1}^{n} \omega_{i,out} N_{i,out} D_{i,out} \quad (3)$$

4.3 ปัจจัยที่มีผลต่อการจัดวางแสตนดาร์ดเซล

การวัดประสิทธิภาพการจัดวางตั้งแต่ในอดีตที่ผ่านมาคือการความยาวโดยประมาณของสายสัญญาณที่ใช้ในการเชื่อมโยง (Total Wire Length) ซึ่งการใช้ความยาวของสายเพียงอย่างเดียวทำให้เลเอาต์ที่ได้จากการออกแบบมักเกิดปัญหาในเรื่องของประสิทธิภาพทางไฟฟ้า ดังนั้นในงานวิจัย จึงพยายามคำนึงถึงปัจจัยที่มีผลต่อเลเอาต์เพื่อเป็นการกำหนดฟังก์ชันเป้าหมายที่เหมาะสมต่อการออกแบบแสดนดาร์ดเซลขึ้นมาใช้งานดังต่อไปนี้คือ

4.3.1 การใช้เนื้อที่ของเลย์เอาต์

การใช้เนื้อที่ของเลเอาต์มีประสิทธิภาพเพียงใด ขึ้นอยู่กับปัจจัยที่สำคัญดังต่อไปนี้คือ การจัดวางตำแหน่งของเซล, จำนวนฟีดธรูเซล (Feedthrough Cell) และเนื้อที่ว่างเปล่า (Waste Space)

ปัจจัยแรกคือการจัดวางเซล เนื่องจากการเชื่อมโยงของแสตนดาร์ดเซลเป็นแบบ Channel Route เกิดจากการเชื่อมต่ออินพุทและเอาต์พุทของเซลอย่างน้อย 2 เซลหรือมากกว่า สายสัญญาณที่เชื่อมต่อถูกแบ่งออกเป็น เวอทิคัลเซ็กเมนท์ (Vertical Segment) และฮอริซซอลทัลเซ็กเมนท์ (Horizontal Segment) โดยที่ความยาวของฮอริซซอลทัลเซ็กเมนท์ ขึ้นอยู่กับความยาวของสายสัญญาณโดยตรง ถ้าสามารถลดความยาวของฮอริซซอลทัลเมนท์ลงได้มากเท่าไรจำนวนแทรค (Track) ที่จะใช้ในการเชื่อมโยงจะลดลงตามไปด้วย ซึ่งจะส่งผลดีต่อการเชื่อมโยงและการใช้เนื้อที่ของเลเอาต์ในทางบวกมากกว่าทางลบ

ปัจจัยที่สองคือฟีดธรูเซล ในกรณีการเชื่อมโยงระหว่างเซลที่มีแถวของเซลขวางอยู่ เราไม่สามารถเชื่อมโยงสายสัญญาณทับลงบนเซล ถ้าการเชื่อมโยงเป็นการเชื่อมโยงแบบสองเลเยอร์จำเป็นต้องใช้ฟีดธรูเซลเพื่อเป็นช่องทางผ่านของสายสัญญาณซึ่งสามารถทำได้ 2 วิธีดังนี้

(1) ใช้ฟีดธรูพิน (Feedthrough Pin) ซึ่งเป็น Electrical Equivalent Termtnal ที่อยู่ภายในเซล ในกรณีนี้เซลที่ออกแบบไม่ต้องมีฟีดธรูพินมาด้วย

(2) ใช้ฟีดธรูเซลเพิ่มเข้าไปในแถวซึ่งจะมีผลกระทบต่อเนื้อที่และ Aspect Ratio โดยตรง การลดจำนวนของฟีดธรูเซล จึงเป็นปัจจัยหนึ่งที่ส่งผลดีต่อเนื้อที่ของเลเอาต์เช่นกัน การกำหนดฟังก์ชันเป้าหมาที่ควบคุมฟีดธรูเซลสำหรับการออกแบบแสตนดาร์ดเซลที่ไม่มี Electrical Equivalent Termtnal จึงจำเป็นอย่างยิ่ง

ปัจจัยที่สามคือ พื้นที่ว่างเปล่าท้ายแถวเกิดขึ้นเนื่องจากแต่ละแถวมีจำนวนเซลไม่เท่ากัน แต่ละเซลมีขนาดความกว้างของเซลไม่เท่ากัน จึงทำให้เนื้อที่ว่างเปล่าที่ท้ายแถวไม่เท่ากัน การควบคุมพื้นที่ว่างเปล่าให้เหลืออยู่เท่าๆกัน จะทำให้การใช้เนื้อที่ของเลเอาต์เป็นไปอย่างมีประสิทธิภาพ

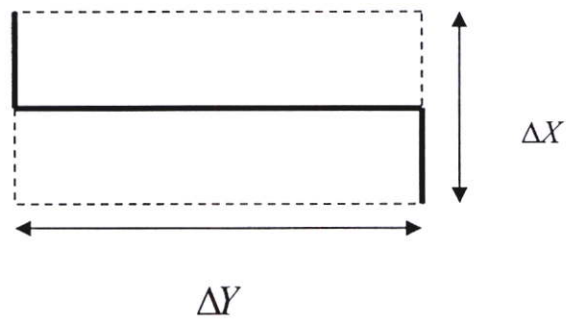
4.3.2 ประสิทธิภาพทางไฟฟ้า

ประสิทธิภาพทางไฟฟ้าของวงจรขึ้นอยู่กับระยะทางการเชื่อมโยงของสายสัญญาณ สำหรับในวงจรรวมเมื่อป้อนกระแสไฟฟ้าเข้าสู่อินพุท กระแสไฟฟ้าไหลผ่านเซลและสายสัญญาณที่ต่อกันเป็นวงจร เพื่อไปออกที่เอาต์พุต ต้องใช้เวลาช่วงหนึ่ง เรียกว่า Propagation Delay เวลาที่ใช้มากที่สุดจะต้องไม่เกินเวลาที่กำหนด ในการออกแบบเลเอาต์เพื่อแก้ปัญหาคลิทิคัลพาท (Critical Path) ที่เกิดจาก Propagation Delay นั้น เงื่อนไขของความล่าช้าเกิดจาก สาเหตุหลายประการด้วยกัน เช่น Intrinsic Delay ของ Device ,Drive Factor, Temperature,Voltag และTrack Length ซึ่งมีผลสืบเนื่องมาจากการจัดวางและการเชื่อมโยง จึงจำเป็นที่ต้องนำเอาเงื่อนไขความยาวของสายสัญญาณมาเป็นส่วนหนึ่งของฟังก์ชันเป้าหมาย

การควบคุมปัญหาคลิทิคัลพาธสามารถทำได้หลายวิธี วิธีที่ง่ายก็คือการใช้ความยาวของสายสัญญาณทุกเส้นที่อยู่ภายใต้คลิทิคัลพาธ เป็นตัวควบคุม โดยกำหนดให้ L(p) เป็น Critical Path Length

$$L(P) = \sum_{\forall n \in p} Half\_perimeter_n \qquad (4)$$

Half Perimeter ของสายสัญญ เณใด ๆ หาได้จาก $\Delta x + \Delta Y$ ของเส้นรอบเซลทั้งหมดที่เป็นสายสัญญาณนั้น ดังแสดงในรูปที่ 4.6



รูปที่ 4.6 Half Perimeter

4.3.3 การเชื่อมโยงที่สมบูรณ์

การเชื่อมโยงของแสตนดาร์ดเซลเป็นการเชื่อมโยงภายในแชนเน็ล พื้นที่ในการเชื่อมโยงอยู่ระหว่างแถวของเซล ขนาดของพื้นที่ระหว่างแถวสามารถปรับเปลี่ยนได้ โดยการขยับแถวของเซล ต่างจากการเชื่อมโยงของ Gate Array ที่มีขนาดของพื้นที่คงที่ ทำให้เกิดปัญหา Routing Congestion ทำให้การเชื่อมโยงไม่สมบูรณ์ร้อยเปอร์เซ็นต์ การเกิด Routing Congestion สาเหตุหนึ่งสืบเนื่องมาจากไม่มีเนื้อที่เพียงพอต่อการเชื่อมโยงสำหรับวงจรรวมแสตนดาร์ดเซล ขึ้นอยู่กับ

การจัดวางเป็นหลัก ซึ่งมีโอกาสเกิด Routing Congestion ได้เช่นกัน วิธีการแก้ปัญหา Routing Congestion สามารถทำได้โดย

(1) การเชื่อมต่อของสายสัญญาณ ควรใช้จำนวนแชนเน็ลและจำนวนแทร็คให้น้อยที่สุด

(2) กระจายฮอริซซอลทัลเซ็กเมนท์ ไปยังแชนเน็ลใกล้เคียงที่อยู่ติดกันที่มีจำนวนความหนาแน่นของแทร็ค (Track Density) น้อยกว่าซึ่งจะกล่าวโดยละเอียดในเรื่อง Global Routing

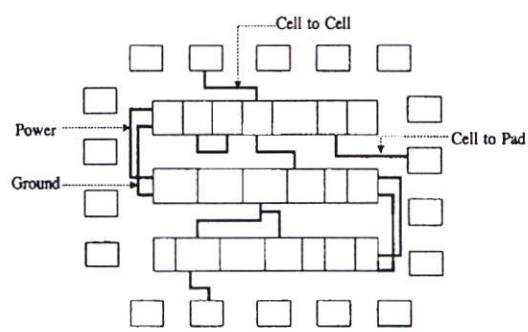
4.3.4 Aspect Ratio

ทุกแถวหลังจากจัดวางต้องมีความกว้าง (Width) ไม่เกิน Core Area ที่กำหนดเนื่องจากขณะที่มีการจัดวางเซลล์มีการสลับตำแหน่งของเซลล์ส่งผลกระทบต่อความกว้างของแถวการควบคุมความกว้างของแถวให้อยู่ภายใต้เงื่อนไข ทำได้โดยกำหนดให้ S เป็นความแตกต่างความกว้างของแถว กับความกว้างที่กำหนด

การควบคุมความสูง (Height) ขึ้นอยู่กับจำนวนแถวและพื้นที่ในการเชื่อมโยงก่อนทำการจัดวางและการเชื่อมโยงได้มีการคำนวณพื้นที่โดยประมาณ (Area Estimate) มาก่อนซึ่งเป็นการกำหนด Aspect Ratio ในระดับหนึ่งแล้ว

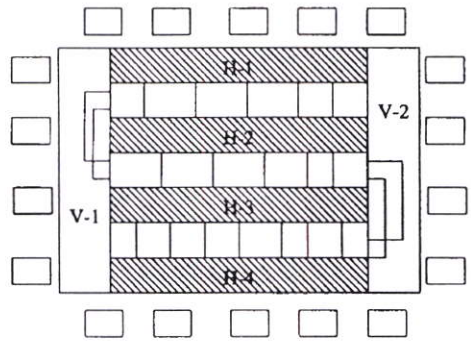
4.3.5 การคำนวณพื้นที่โดยประมาณ (Area Estimate)

การคำนวณหาพื้นที่โดยประมาณของ Core Area ก่อนทำการจัดวางและการเชื่อมโยงจริงเพื่อตรวจสอบจำนวนแถวที่กำหนด สามารถให้เลย์เอาต์ใกล้เคียงตาม Aspect Ratio ที่ต้องการหรือไม่ ก่อนดำเนินการจัดวางและเชื่อมโยง โดยกำหนดให้พื้นที่เลย์เอาต์ถูกแบ่งออกเป็นสองส่วนคือ Core Area กับ Pad Area ดังแสดงในรูปที่ 4.7 การที่จะประมาณพื้นที่ จึงจำเป็นกำหนดช่องทางเดินของสายสัญญาณ (Global Routing)



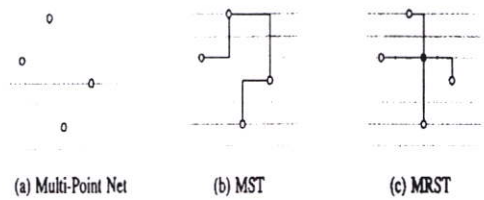
รูปที่ 4.7 พื้นที่ของเลย์เอาต์

นอกจากนี้ยังสามารถหาความยาวของสายสัญญาณ เพื่อจะใช้ในการปรับปรุงการจัดวางตำแหน่งต่อไป การเชื่อมสายสัญญาณจะกระทำในลักษณะสองมิติ โดยพื้นที่แบ่งเป็น ฮอรซซอนทัลแชนเน็ล (Horizontal Channel) ซึ่งประกอบด้วย H-1,H-2, H-3, H-4 และ เวอทิคัลแชนเน็ล (Vertical Channel) ซึ่งประกอบด้วย V-1, V-2, V-3 ดังรูปที่ 4.8



รูปที่ 4.8 แชนเนลที่ใช้เชื่อมสายสัญญาณ

การหาเส้นทางคร่าวๆ ขอสายสัญญาณกระทำได้ใช้วิธีสายสัญญาณที่มีจุดเชื่อมมากกว่า 2 จุด (Multi-Point Net) ดังแสดงในรูปที่ ต้องแบ่งเป็น Two-Point Segment โดยใช้มินิมัมแพนนิงทรี (MST: Minimum Spanning Tree) หรือมินิมัมเรดติลิเนียสแพนนิงทรี (MRST: Minimum Rectilinear Steiner Tree) ดังแสดงในรูปที่ 4.9



รูปที่ 4. 9 Multi-Point Net, MST, และ MRST

ความแตกต่างของ MST และ MRST อยู่ที่จำนวนจุดของการเชื่อมต่อและความยาวของสายสัญญาณ โดย MRST จะสั้นกว่าหรือเท่ากับ MST เสมอ ส่วน MRST จะมีจำนวนจุดที่มากกว่า MST ก็เพราะมีจำนวนจุดเพิ่มเข้ามาเรียกว่าสไทเนอร์พอยท์ (Steiner Point) ในงานวิจัยนี้จะใช้ทั้งคู่ MRST จะใช้เมื่อเป็นคลิทิคัลพาธ ส่วนสายสัญญาณอื่นๆใช้ MST

4.4 สรุปอัลกอริทึม MGA สำหรับการจัดวางของแสตนดาร์ดเซล

การใช้ MGA สำหรับการจัดวางเซลแบ่งเป็น 2 ขั้นตอนหลักคือ ขั้นตอนแรก จะใช้ MGA สำหรับหาคำตอบที่ดีที่สุดของการจัดกลุ่มก่อน ขั้นตอนที่สองจะนำคำตอบจาก ขั้นตอนแรกไปใช้ สำหรับวางผังและวางเซลและหาคำตอบที่ดีโดยใช้ MGA เราสรุป MGA สำหรับการจัดวางดังนี้

1. อ่านไฟล์วงจรทดสอบ (Netlist File) นำข้อมูลที่ได้ ไปเข้ารหัสปัญหาและสร้างประชากรเริ่มต้นโดยการใช้วิธี k-nearest-neighbour และสร้างกลุ่มให้กับประชากร ในตอน 3.2.1 และ 3.2.2 (ข้อนี้จะใช้เฉพาะขั้นตอนแรกเท่านั้น)

2. จับคู่ประชากรทั้งหมดและตรวจสอบว่าทั้งคู่มีคำตอบเหมือนกันหรือไม่ ถ้าไม่เหมือนทำการ Greedy Crossover ถ้าเหมือนกันทั้งคู่จะนำประชากรมา 1 ตัวทำการ 3-Opt Mutation จากนั้นนำประชากรหาค่า Cost ของขั้นตอนแรกตามสมการ 1 (หลังจากเสร็จขั้นตอนแรกส่วนนี้จะเป็นการหาค่า Cost ของขั้นตอนที่สองตามสมการ 2, 3)

3. สร้างประชากรรุ่นใหม่ เราใช้ Tournament Selection โดยจะเลือกสุ่มประชากรมา 2 ประชากร และจะเลือกประชากรที่มีค่า Cost ดีที่สุดเพื่อนำไปสร้างประชากรรุ่นใหม่ ถ้าในประชากรรุ่นใหม่มีค่า Cost น้อยกว่า ประชากรรุ่นปัจจุบัน เราจะเลือกประชากรรุ่นปัจจุบัน ที่มีค่า Cost ดีที่สุด ซึ่งจะถูกนำไปเป็นตัวแทนประชากรรุ่นใหม่

4. สร้างประชากรรุ่นถัดไปโดยกลับไปทำข้อ 2 แต่ถ้าคำตอบที่ดีที่สุด ไม่ได้ถูกปรับปรุงมากกว่า N รุ่นให้หยุดคำนวณ

บทที่ 5

ผลการทดสอบกับวงจร MCNC benchmark circuit

5.1 การทดลองและผลการทดลองกับวงจร MCNC benchmark circuit

ระบบ MGA เขียนโดยใช้ภาษา Java ทำงานบน ไมโครคอมพิวเตอร์ CPU Pentium 4 1.8 GHz ทดสอบกับวงจร MCNC benchmark [6] ดังตารางที่ 5.1 กำหนดสร้างประชากร 500 รุ่น โดยมีขนาดของประชากร 10 และรันโปรแกรมคำนวณ 30 ครั้ง และหาค่าเฉลี่ย ตารางที่ 5.2, 5.3 แสดงผลการจัดกลุ่มโดยเปรียบเทียบกับผลที่ใช้อัลกอริทึม MGA, Genetic Algorithm (GA) [8] และ F-M Algorithm [8] ที่ได้รายงานผลไว้ใน [8] ซึ่ง HGA สามารถให้ค่า Cut Size ที่ดีกว่าอัลกอริทึมอื่นในหลายกรณี

ตารางที่ 5.1 รายละเอียดวงจรทดสอบ

Benchmark	**#Cell**	**#Net**	**#I/O**	**Pin/Net**	**Pin/Cell**
Fract	125	147	24	2.79	3.63
Struct	1888	1920	64	2.82	2.86

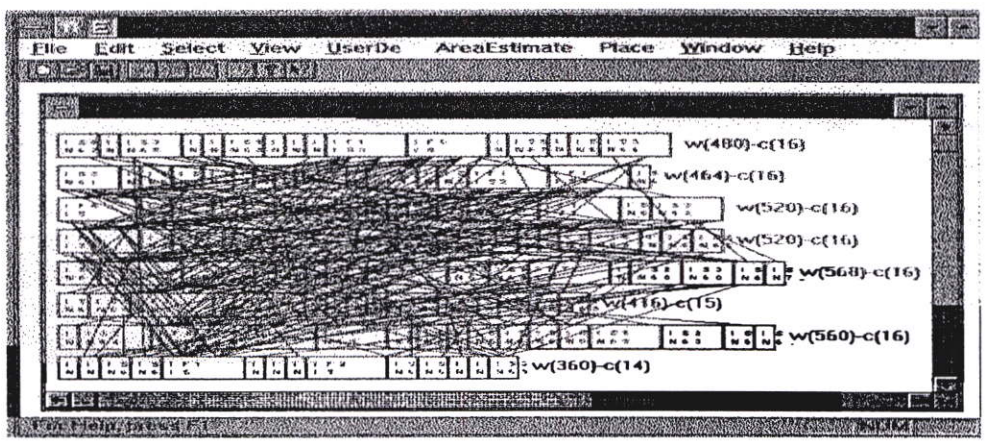
ตารางที่ 5.2 การจัดกลุ่มเซล Multiway Partitioning วงจร Fract

Partitions	**Cut/Balance weight**	**F-M Algorithm [8]**		**Genetic Algorithm (GA) [8]**		**Modified Genetic Algorithm (MGA)**	
		Min Cut	Ave Cut	Min Cut	Ave Cut	Min Cut	Ave Cut
4	100/15	36	49.9	24	41.0	26	45.3
8	100/20	80	101.3	69	97.7	64	92.4
16	100/30	172	206.4	160	197.8	152	190.3
32	100/75	305	409.5	328	383.8	312	374.7

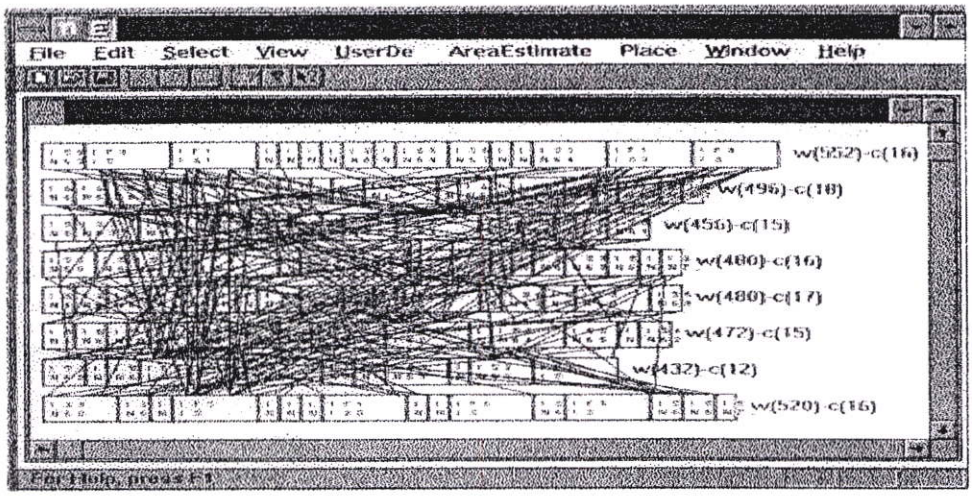
ตารางที่ 5.3 การจัดกลุ่มเซล Multiway Partitioning วงจร Struct

Partitions	Cut/Balance weight	F-M Algorithm [8]		Genetic Algorithm (GA) [8]		Modified Genetic Algorithm (MGA)	
		Min Cut	Ave Cut	Min Cut	Ave Cut	Min Cut	Ave Cut
4	200/10	417	480.9	393	439.6	384	421.2
8	500/10	875	986.5	764	913.4	750	904.7
16	200/10	1605	1854.8	1792	1964.9	1684	1821

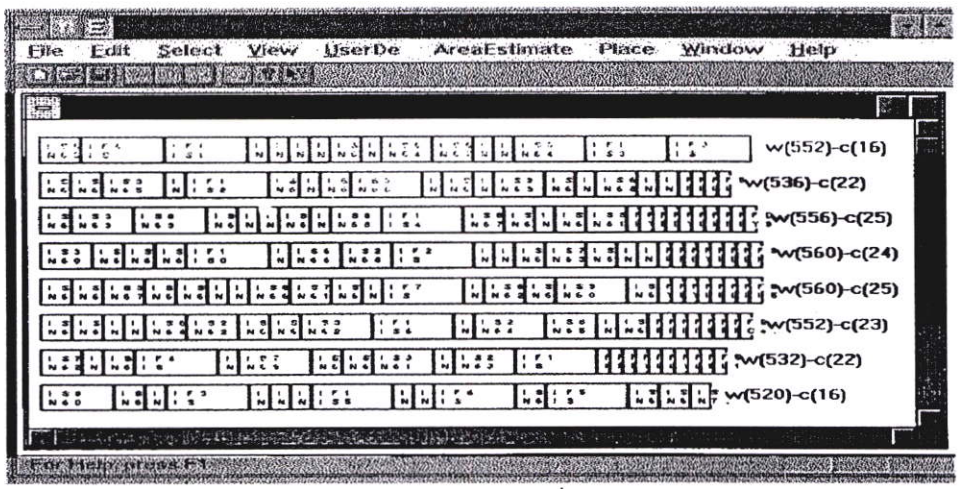
เราได้นำวงจร Fract ที่จัดกลุ่มแล้วโดยมี Partition=8, Min Cut=64, Ave Cut=92.4 ดังแสดงตารางที่ 2 มาทำการวางผังและจัดวางเซลดังแสดงในรูปที่ 5.1, 5.2, และ 5.3 โดยใช้ MGA และ GA แบบ Steady state genetic algorithm [8] (Mutation probability=0.02, ขนาดประชากร=10) และเปรียบเทียบผลกับ SA (Simulated Annealing) ที่ได้รายงานค่าใน [5] ผลปรากฏว่า MGAจะได้ผลของค่าต่างๆดีกว่า GA และ SA ดังตารางที่ 5.4 จึงส่งผลให้พื้นที่เลเอาต์ของวงจร Fract โดยใช้ MGA จะมีค่าน้อยกว่า GA อยู่ 8.14%



รูปที่ 5.1 การจัดกลุ่มเซลพร้อมแสดงคัทฟีดของวงจร Fract



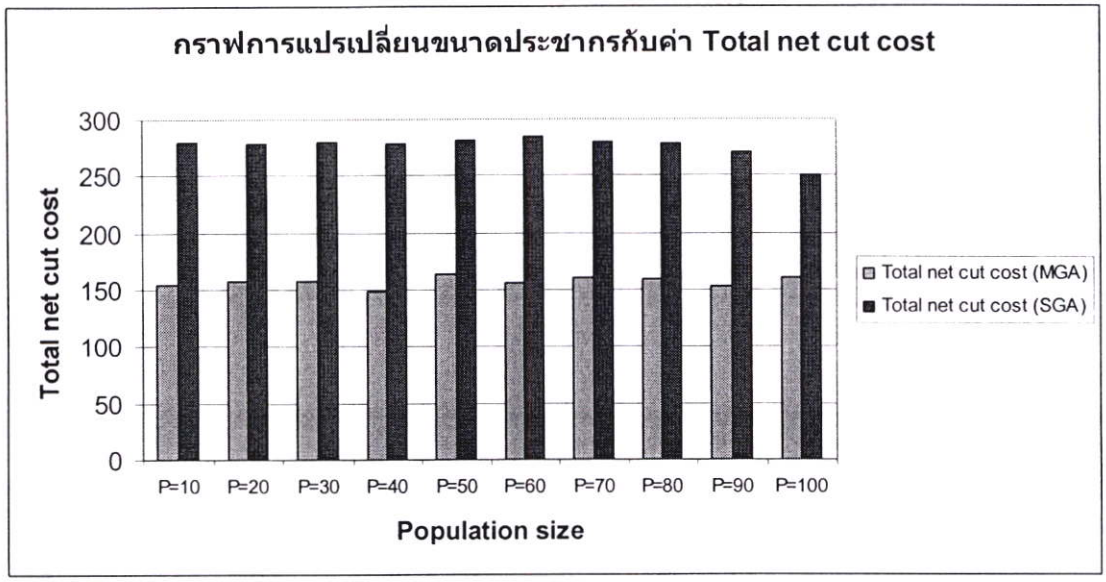
รูปที่ 5.2 การวางผังเซลพร้อมคัทฟีดของวงจร Fract



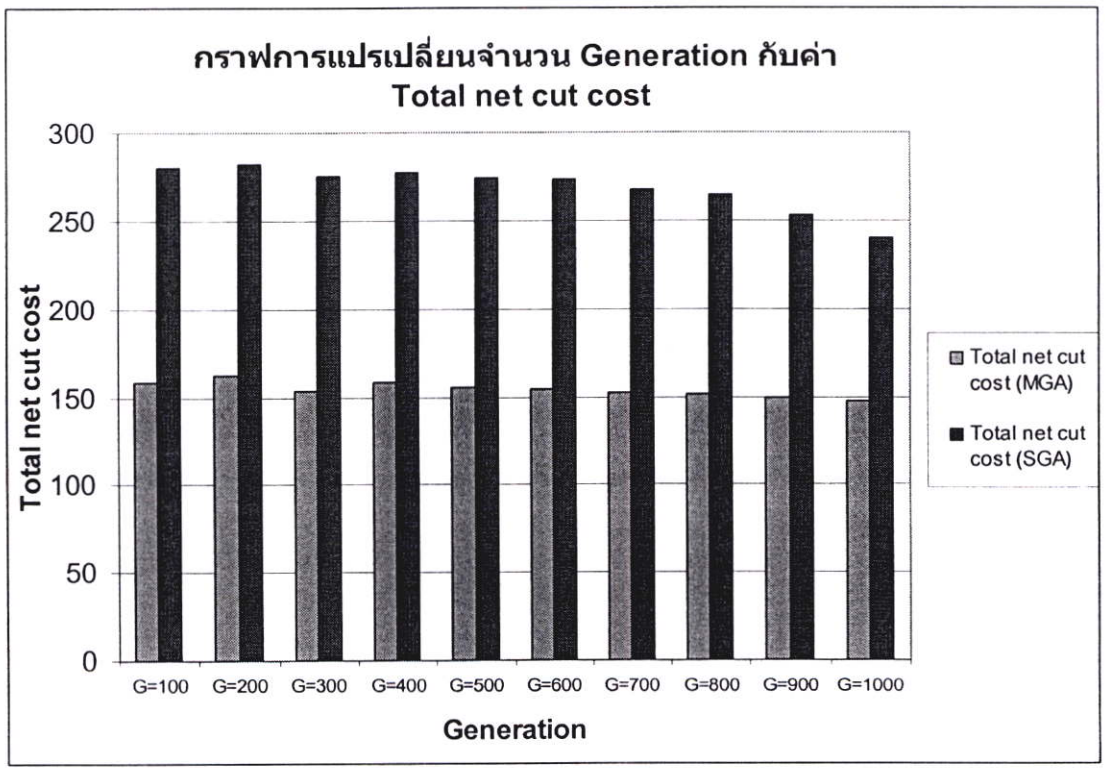
รูปที่ 5.3 การจัดวางเซลของวงจร Fract

ตารางที่ 5.4 ประสิทธิภาพของการจัดวางวงจร Fract โดยใช้อัลกอริทึมต่างๆ

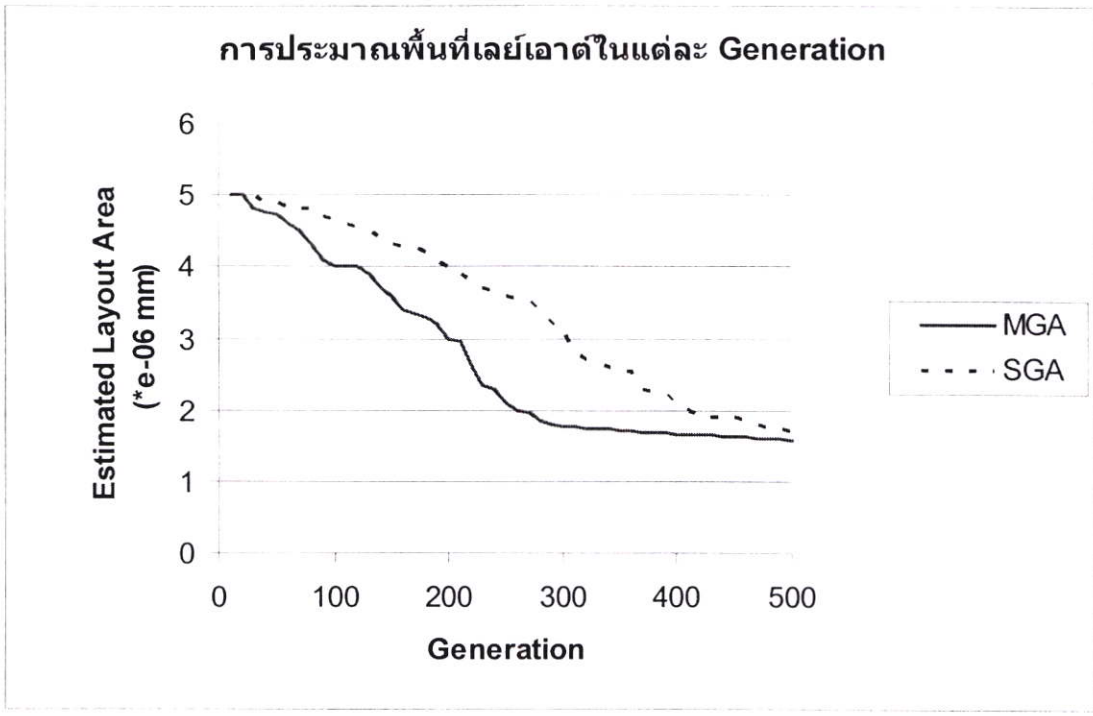
Quantity	SA [5]	GA [8]	MGA
Feed-through Cell (Cell)	51	45	42
Critical Path Length (μm)	335	328	320
Max Row Width (μm)	616	595	580
จำนวน Track	140	134	128
Total Wire length	-	63,014	62,185
Layout Area (μm^2)	-	1.72×10^6	1.58×10^6



รูปที่ 5.4 กราฟการแปรเปลี่ยนค่าขนาดประชากรกับค่า Total net cut cost ของวงจร Fract



รูปที่ 5.5 กราฟการแปรเปลี่ยนจำนวน Generation กับค่า Total net cut cost ของวงจร Fract



รูปที่ 5.6 กราฟประมาณพื้นที่เลเอาต์ของวงจร Fract ในแต่ละ Generation

เรายังได้ทดสอบโดยใช้วงจร Fract ที่มีขนาดกลุ่ม 8 กลุ่มกับ MGA และ SGA (ซึ่งเป็นเจเนติค อัลกอริทึมแบบพื้นฐาน) และเปรียบเทียบค่า Total net cut cost ระหว่างทั้งสองอัลกอริทึม โดยแปรเปลี่ยนค่าขนาดประชากรตั้งแต่ 10 จนถึง 100 โดยจำนวนรุ่นประชากรคงที่เท่ากับ 100 รุ่น รันโปรแกรม 30 ครั้งแล้วหาค่าเฉลี่ย ปรากฏว่า MGA สามารถให้ค่า Total net cut cost ได้ดีกว่า SGA ดังรูปที่ 5.4 โดยการแปรเปลี่ยนขนาดของประชากรมิได้มีผลกระทบต่อค่า Total net cut cost ของ MGA มากนัก แต่จะมีผลกระทบในกรณีของ SGA ซึ่งถ้ายิ่งมีขนาดประชากรมากจะทำให้ค่า Total net cut cost ของ SGA ดีขึ้น แต่จะใช้เวลาในการรันโปรแกรมมากขึ้นตามไปด้วย

รูปที่ 5.5 เราได้แปรเปลี่ยนจำนวนรุ่นของประชากร โดยให้มีขนาดประชากรคงที่เท่ากับ 10 ปรากฏว่า ค่า Total net cut cost ของ MGA ดีกว่า SGA และแนวโน้มค่าของทั้งสองอัลกอริทึมดีขึ้นเมื่อเพิ่มจำนวนรุ่นมากขึ้นด้วย สำหรับ รูปที่ 5.6 เป็นกราฟประมาณพื้นที่เลเอาต์ของวงจร Fract ในแต่ละรุ่นของประชากรเปรียบเทียบระหว่าง MGA กับ SGA

บทที่ 6
สรุปผลการการวิจัยและข้อเสนอนะ

6.1 สรุปผลการวิจัย

เราได้นำเสนอโมดิฟาย เจเนติค อัลกอริทึม สำหรับการจัดวางตำแหน่งของแสตนดาร์ดเซล สำหรับโมดิฟาย เจเนติค อัลกอริทึม มีขั้นตอนหลัก 4 ขั้นตอน คือขั้นตอนการเข้ารหัสปัญหา โดยเราจะแทนปัญหาการจัดวางตำแหน่งของแสตนดาร์ดเซลให้อยู่ในรูปโครโมโซมของเวคเตอร์จำนวนเต็ม (Integer Vector) ขั้นตอนที่สองจะเป็นการสร้างประชากรเริ่มต้น ซึ่งจะทำการสร้าง k-nearest-neighbour subgraph ให้กับเซล และสร้างประชากรจากการเลือกเซลที่เป็น k-nearest-neighbour subgraph เพื่อทำการสร้างกลุ่มของเซลที่มีสายสัญญาณเชื่อมต่อกันให้อยู่กลุ่มเดียวกัน โดยในขั้นตอนนี้จะเป็นการสร้างประชากรที่มีคุณภาพแต่มีจำนวนน้อย เพื่อที่จะสามารถค้นหาคำตอบที่เหมาะสมได้อย่างราดเร็ว ขั้นตอนที่สามกรีดี้ครอสโอเวอร์ ขั้นตอนนี้จะเลือกประชากรโดยวิธี Tournament Selection เพื่อมาทำการกรีดี้ครอสโอเวอร์ การครอสโอเวอร์มีหลายแบบด้วยกัน เช่น การครอสโอเวอร์แบบหนึ่งจุด การครอสโอเวอร์แบบหลายจุด พาร์เชียลแมปครอสโอเวอร์ ออร์เดอร์ครอสโอเวอร์ ไซเคิลครอสโอเวอร์ โดยแต่ละแบบมีการประยุกต์ใช้แก้ปัญหาแตกต่างกันไป กรีดี้ครอสโอเวอร์ เป็นแบบใหม่แบบหนึ่งที่มีการนำเสนอเพื่อใช้แก้ปัญหา TSP เราได้ประยุกต์กรีดี้ครอสโอเวอร์ เพื่อใช้แก้ปัญหาการจัดวางตำแหน่งของแสตนดาร์ดเซล การเลือกประชากรนำมาในขั้นตอนนี้ ถ้าประชากรตัวไหนที่เลือกมาแล้วเหมือนกันจะนำไปทำขั้นตอนที่สี่ต่อไป ขั้นตอนที่สี่เป็นการมูเตชั่นแบบสามโอพีทรีฮิวริสติคจากขั้นตอนที่สามประชากรสองตัวที่เหมือนกันจะถูกนำมาทำการมูเตชั่นแบบสามโอพีทรีฮิวริสติคหนึ่งตัว ซึ่งเป็นการปรับตำแหน่งของยีนใหม่โดยอาศัย k-nearest-neighbour subgraph โดยจัดวางเซลตามแกน x และแกน y และทำการจัดเรียงตัวใหม่ของยีน

นอกจากนี้เราได้ประยุกต์หลักการออกแบบผังวงจรแบบลำดับชั้น มาใช้ในการแก้ปัญหาการจัดวางตำแหน่งของแสตนดาร์ดเซล โดยนำโมดิฟาย เจเนติค อัลกอริทึมมาใช้ในการแก้ปัญหาในการจัดวางตำแหน่งของแสตนดาร์ดเซลซึ่งแบ่งเป็นสามปัญหาย่อยๆคือ ปัญหาการจัดกลุ่มเซล ปัญหาการวางผังเซล และปัญหาการวางเซล ขั้นแรกจะนำโมดิฟาย เจเนติค อัลกอริทึมมาใช้แก้ปัญหาการจัดกลุ่มเซลตามค่าฟิตเนสฟังก์ชั่นหรือคอสฟังก์ชั่นในแต่ละปัญหา เมื่อเราได้คำตอบที่พึงพอใจ จะนำคำตอบไปทำการแก้ปัญหาในขั้นตอนการวางผังเซล และปัญหาการวางเซลเป็นลำดับต่อไป นอกจากนี้เรายังคำนึงถึงประสิทธิภาพทางไฟฟ้า การใช้เนื้อที่ของเลย์เอาต์ และ Aspect Ratio เพื่อให้ได้วงจรที่ออกแบบสามารถใช้งานได้อย่างมีประสิทธิภาพยิ่งขึ้นด้วย

เอกสารอ้างอิง

[1] Yang, R. "Solving Large Travelling Salesman Problems with Small Populations." In Proceeding of GALESIA97, 1997, pp.157-162.

[2] Sun W.-J., Sechen, C. "Efficient and Effective Placement for Very Large Circuits." IEEE Trans. Computer-Aided Design, Vol. 14, 1995, pp.349-359.

[3] Wei, Y.C., Cheng, C.K. "Ratio Cut Partitioning for Hierarchical Designs." IEEE Trans. Computer-Aided Design, Vol. 10, 1991, pp.911-921.

[4] Swartz, W. P. "Automatic Layout of Analog and Digital Mixed Macro/Standard Cell Integrated Circuits." Ph.D. Thesis of Yale University, May 1993, pp. 118-146.

[5] Maisalee, S. "Cell-Based Placement and Routing Optimization." Master thesis, King Mongkut's Institue of Technology Ladkrabang, 1997.

[6] Kozminski, K. "Benchmarks for layout systhesis." in Proc. 28th Design Automation Conf., 1991, pp. 265-270.

[7] David E. Goldberg, Deb, K. "A comparative analysis of selection schemes used in genetic algorithms." in G. Rawlins, ed., Foundations of Genetic Algorithms, San Mateo, CA: Morgan Kaufmann,1991, pp. 69-93.

[8] Pinaki Mazumder, Elizabeth M. Rudnick. **Genetic Algorithms for VLSI Design, Layout & Test Automation**. Prentice Hall PTR, 1999.

[9] Koren, I. and Koren, Z. "Incorporating Yield Enhancement into the Floorplanning Process." IEEE Trans. On Computer, Vol. 49, 2000, pp.532-541.

[10] Rebaudengo, M. and Reorda, M.S. "GALLO: A Genetic Algorithm for Floorplan Area Optimization." IEEE Trans. Computer-Aided Design, Vol. 15, 1996, pp.943-951.

[11] David E. Goldberg. **Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley**, 1989.

[12] Alexandria Virginia. **Understanding Computers the Chipmaker**. The Editors of Time Life Books, Time-Life Books.,1988.

[13] Stanley L. Hurst. **Custom VLSI Microelectronics**. Prentice Hall International (UK) Ltd.,1992.

[14] บวร ปภัสราทร, ประเสริฐ คันธมานนท์ และ สุเมธ อังคะศิริกุล. **เทคโนโลยีการออกแบบวงจรรวม**. 2533.

[15] John. P. Uyemura. **Physical Design of CMOS Integrated Circuits Using L-EDIT™** . PWS Publishing Company, 1995.

[16] ดร.สุชาย ธนวเสถียร และคณะ. **การออกแบบวงจรรวม VLSI Design**. กรุงเทพฯ, ศรีอนันต์การพิมพ์, 2535.

[17] John. P.Huber, Mark W. Rosneck. **Successful ASIC Design the First Time Through**. Van Nostrand Reinhold 1991.

[18] Hicks, P.J. **Semi-Custom IC Design and VLSI**. Peter Peregrinus Ltd., 1983.

[19] Malcolm R. Haskard, **An Introduction to Application Specific Integrated Circuits**. Prentice Hall of Australia Pty Ltd., 1990.

[20] Sadiq M.Sait, Habib Youssef. **VLSI Physical design Automation**. McGraw –Hill Book Company, 1995.

[21] Sara Baase. **Computer Algorithms Introdution to Design and Analysis**. Addison Wesley Publishing Company, 1989.

[22] P.R. Adby, M. A. H. Dempstek. **Introduction to Optimization Methods**. London Chapman and Hall , 1974.

[23] Cheung Y.S., Lee Y.W., and Yeung C.S.K. "Hierarchical Simulated Annealing For Standard Cell Placement." Proc.ISIC – 91,No.2, 1991, pp.17-22.

[24] Sun W.J., Sechen C. "Efficient and Effective placement For Very Large Circuits." IEEE Transactions on Computer – Aided Design of integrated circuits and System, Vol.14. No.3, March 1995, pp.349-359.

[25] Fadi J. Kurdasi, Alice C. Parker ," Techniques For Area Estimation of VLSI Layouts." IEEE Transactions on Computer –Aided Desing ,VOL.8 ,No.1,January 1989.

[26] Bryan T. Preas, Michael J. Lorenzetti. **Physical Design automation of VLSI Systems**. The BenJamin / Cummings Publishing company Inc.,1988.

[27] Wei Y.W., Cheng C.K., "Ratio Cut Partionting for Hierarchical Designs." IEEE Transections on Computer-Aided Desing ,Vol.10,No.7, July 1991, pp.911-921.

[28] Cheng C.K., Wei,Y.C. "An Improved Two-Way Partitioning Algorithm With Stable Performance." IEEE Transections on Computer-Aided Design Vol.10,No.12, December 1991, pp.1502-1511.

[29] Sanchis L.A. "Multiple-Way Network Partitionting." IEEE Transections on Computers Vol.38,No.1, January 1989, pp.62-81.

[30] Micheal Pecht. **Placement and Routing of Electronic Models**, Marcel Dekker, Inc.,1993.

[31] Griffith, J., Robins G., Salowe J.S., and Zhang T. "Closing the Gap: Near-Optimal Stener Trees in Polynomial Time", Department of Computer Science,University of Virginia, Charlottesville, Va 22903-2442.

[32] T. Ohtsuki. **Layout Design and Vertification**. Elsevier Science Publishing Company Inc.,1986.

ภาคผนวก ก. ผลงานวิจัยที่ได้รับการตีพิมพ์

[1] อภิชาต วสุธาพิทักษ์, บรรจง ปียธำรง. "การปรับปรุงประสิทธิภาพการจัดวางเลเอาต์ของแสตนดาร์ดเซลโดยใช้ ไฮบริด เจเนติค อัลกอริทึม." การประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 24, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, 22-23 พฤศจิกายน 2544,หน้า 1350-1355.

ภาคผนวก ก. โปรแกรมการทดลอง MGA_Placement

```
/*
 * MGA_PLACEMENT.java      version 1.00     8 March 2002
 import java.io.*;
import Utilities.*;
import GAutilities.*;
import Chromosomes.*;
import Selections.*;
import Crossovers.*;
import t7.*;
class MGA {

  // give fundamental parameters controlling the GA default values
  private static int       populationSize =  10;
  private static int       greedyXoverPoints;
  private static boolean   doElitism     = true;
  private static double    crossoverRate =    0.7;
  private static double    optmutationRate;

  // these parameters control the running of the program
  private static int       printPerGens   =  2; // print every this often
  private static int       maxGenerations = 100; // quit when reached
  private static String    logFileName    =  "pm30.dat";

  // record the time the program started
  private static final long  startTime     = System.currentTimeMillis();
  private static long age() {
    return System.currentTimeMillis() - startTime;
  }
```

```
public static void main(String[] args) {

  // process command line arguments, overriding GA parameter defaults
  GetOpt go = new GetOpt(args, "Udp:x:c:m:P:G:EF:");
  go.optErr = true;
  String usage = "Usage: -d -p populationSize -x numXoverPoints\n"
    + "       -E -c crossoverRate -m mutationRate\n"
    + "       -P printPerGens -G maxGenerations -F logFileName";
  int ch = -1;
  while ((ch = go.getopt()) != go.optEOF) {
    if     ((char)ch == 'U') {
      System.err.println(usage);  System.exit(0);
    }
    else if ((char)ch == 'd') Debug.flag = true;
    else if ((char)ch == 'p')
      populationSize = go.processArg(go.optArgGet(), populationSize);
    else if ((char)ch == 'x')
      numXoverPoints = go.processArg(go.optArgGet(), numXoverPoints);
    else if ((char)ch == 'c')
      crossoverRate = go.processArg(go.optArgGet(), crossoverRate);
    else if ((char)ch == 'm')
      mutationRate = go.processArg(go.optArgGet(), mutationRate);
    else if ((char)ch == 'E') doElitism = true;
    else if ((char)ch == 'P')
      printPerGens = go.processArg(go.optArgGet(), printPerGens);
    else if ((char)ch == 'G')
      maxGenerations = go.processArg(go.optArgGet(), maxGenerations);
    else if ((char)ch == 'F')
      logFileName = go.optArgGet();
    else { // undefined option
      System.err.println(usage);  System.exit(1);
```

```
    }
  }

  Defaults.logFileName = logFileName;
  Globals.stdout.println
    ("The GA parameters are:\n" +
     "  populationSize     = " + populationSize      + "\n" +
     "  numXoverPoints     = " + numXoverPoints      + "\n" +
     "  crossoverRate      = " + crossoverRate       + "\n" +
     "  mutationRate       = " + mutationRate        + "\n" +
     "  doElitism          = " + doElitism           + "\n" +
     "  printPerGens       = " + printPerGens        + "\n" +
     "  maxGenerations     = " + maxGenerations      + "\n" +
     "  Debug.flag         = " + Debug.flag          + "\n" +
     "  logFileName        = " + logFileName);

  // force loading class MyChromosome and setting static variables
  new MyChromosome();
  Globals.stdout.println("GA: the number of row  = " +
    Chromosome.getRow());
  Globals.stdout.println("GA: chromosome length = " +
    Chromosome.getChromosomeLength());

  sGA theGA = new sGA();
  theGA.mainLoop();
}

private sGA() { super(); } // only GA.main() can create GA

private void printChromosome(String name, Chromosome c) {
  Globals.stdout.println(name +
    ":\n   " + c.toGenotype() +
```

```
//      "\n   " +
      "\n    fitness= " + c.getFitness());    // + c.toPhenotype() +
  }

  private final Chromosome theBest = new MyChromosome();
  private int theBestGeneration; // generation theBest first showed up
  // 0..popSize-1 holds population and theBest holds the most fit
  private Chromosome[] population    = new Chromosome[populationSize];
  private Chromosome[] newPopulation = new Chromosome[populationSize];

  private int generationNum;     // current generation number
  private int numCrossovers, numMutations;  // tabulate for report()

  private Selection theSelection = null;
  private Crossover theCrossover = null;

  private void mainLoop() {
    Globals.stdout.println("GA: mainLoop");
    if (Chromosome.isSolutionFitnessKnown()) {
      Globals.stdout.println("Known solution fitness is " +
        Chromosome.getSolutionFitness());
    }
    initialize();
    findTheBest();
    report("Initial population");
    while (!terminated()) {
      generationNum++;
      try {
        theSelection.select(population, newPopulation, populationSize);
        swapPopulationArrays();
      } catch (FitnessSumZeroException e) {
        // replace current population with a new random one;
```

```
      // other possibilities are to do nothing and hope mutation
      // fixes the problem eventually
      if (Debug.flag) Globals.stdout.println(e + " (randomizing)");
      for (int i = 0; i < populationSize; i++) {
        population[i].initializeChromosomeRandom();
        System.out.println( " population "+i);
      }
    } catch (SelectionException e) {
      if (Debug.flag) Globals.stdout.println(e);
    }
    if (Debug.flag) report("Selection");
    crossover();
    if (Debug.flag) report("Crossover");
    mutate();
    if (Debug.flag) report("Mutation");
    if (doElitism) elitism();
    else justUpdateTheBest();
    if (Debug.flag || (generationNum % printPerGens) == 0) {
      report("Report");
    }
  }
  Globals.stdout.println("Simulation completed in " + generationNum +
    " generations and " + age()/1000.0 + " seconds");
  printChromosome("Best member (generation="+theBestGeneration+")",
    theBest);
  System.exit(0);
}

private void initialize() {

  generationNum = 0; numCrossovers = 0; numMutations = 0;
```

```
theSelection = new ProportionalSelection();

if (numXoverPoints == 0) theCrossover = new UniformCrossover();
else if (numXoverPoints == 1) theCrossover = new OnePointCrossover();
else if (numXoverPoints == 11) theCrossover = new GreedyCrossover();
else theCrossover = new NPointCrossover(numXoverPoints);
for (int j = 0; j < populationSize; j++)
    {

   population[j] = new MyChromosome();
   population[j].initializeChromosomeRandom();
   newPopulation[j] = new MyChromosome();

     }
 if (Debug.flag) {
   for (int j=0; j<populationSize; j++) {
     printChromosome("p" + j, population[j]);
   }
 }
}

private void findTheBest() {  // called on the initial population only
  double currentBestFitness = population[0].getFitness();
  double next = -1;
  int currentBest = 0; // index of the current best individual

  for (int j = 1; j < populationSize; j++) {
    if ((next = population[j].getFitness()) > currentBestFitness ) {
      currentBest = j;
      currentBestFitness = next;
    }
  }
```

```
    // once the best member in the population is found, copy the genes
    population[currentBest].copyChromosome(theBest);

    if (Debug.flag) {
      printChromosome("currentBest (generation="+theBestGeneration+")",
        theBest);
    }
  }

  private boolean terminated() {
    return (theBest.isSolutionFitnessKnown() &&
        theBest.getFitness() == theBest.getSolutionFitness()) ||
        generationNum >= maxGenerations;
  }

  private void swapPopulationArrays() {
    Chromosome[] temp = population;
    population = newPopulation;
    newPopulation = temp;
  }

  /*****************************************************************/
  /* Crossover selection: selects two parents that take part in  */
  /* the crossover.  For each population member, flip a weighted */
  /* coin.  Every two times it comes up < crossoverRate, then    */
  /* crossover those two chromosomes.                            */
  /*****************************************************************/
  private void crossover() {
    int one = -1;    // compiler complains not being initialized
    int first  =  0; // count of the number of members chosen

    for (int mem = 0; mem < populationSize; ++mem) {
```

```
    if (MyRandom.dblRandom() < crossoverRate) {
      ++first;
      if (first % 2 == 0) {
        numCrossovers++;
        if (Debug.flag) {
          Globals.stdout.println("crossing " + one + " and " + mem);
        }
        theCrossover.xOver(population[one], population[mem]);
      } else one = mem;
    }
  }
}

/*****************************************************************/
/* Mutation: Random uniform mutation. A variable selected for */
/* mutation is replaced by a random value between lower and   */
/* upper bounds of this variable                              */
/*****************************************************************/
private void mutate() {
  int chromosomeLength = Chromosome.getChromosomeLength();
   int row = Chromosome.getRow();
  for (int i = 0; i < populationSize; i++)
{
    for (int j = 0; j < row; j++)
  {   for (int k = 0; k < chromosomeLength; k++)
        {
      if (MyRandom.dblRandom() < mutationRate)
      {
        numMutations++;
        population[i].mutateGene(j,k);
        if (Debug.flag) {
          printChromosome("mutation, i=" + i + ", gene=" + j,
```

```
            population[i]);
                    }
            }
        }
    }
  }
}

/*****************************************************************/
/* Report function: Reports progress of the simulation.
/*****************************************************************/
private void report(String title) {
  double best_val;          // best fitness in this population
  double most_fit;          // most fit seen in previous generations
  double avg;               // avg population fitness
  double stddev;            // std. deviation of population fitness
  double sum_square;        // sum of square for std. calc
  double square_sum;        // square of sum for std. calc
  double sum;               // total population fitness
  double fitness;

  sum = 0.0;
  sum_square = 0.0;
  best_val = -1.0;

  for (int i = 0; i < populationSize; i++) {
    fitness = population[i].getFitness();
    sum += fitness;
    sum_square += fitness * fitness;
    if (fitness > best_val) best_val = fitness;
  }
```

```
avg = sum/(double)populationSize;
square_sum = sum * sum/(double)populationSize;
stddev = Math.sqrt((1.0/(double)(populationSize - 1))
  *(sum_square - square_sum));
most_fit = theBest.getFitness();

Globals.stdout.println(title + ": generation=" + generationNum
  + " best value=" + best_val + " avg=" + avg + " stddev=" + stddev);
printChromosome("most fit (previous generation=" + theBestGeneration
  + ")", theBest);
Globals.stdout.println("number of crossovers and mutations: "
  + numCrossovers + " and " + numMutations);

if (Debug.flag) {
  for (int j=0; j<populationSize; j++) {
    printChromosome("p" + j, population[j]);
  }
}
Globals.stdout.flush();
}

/*****************************************************************/
/* Elitist function: The best member of the previous generation */
/* is stored in theBest Chromosome.  If the best member of      */
/* the current generation is worse then the best member of the  */
/* previous generation, the latter one would replace the worst  */
/* member of the current population                             */
/*****************************************************************/
private void elitism() {

  double best, worst;         // best and worst fitness values
  int bestMember, worstMember; // indexes of the best and worst member
```

```
int start; // index used to start the loop

if (populationSize % 2 == 0) {
  best = -1; bestMember = -1;
  worst = Double.MAX_VALUE; worstMember = -1;
  start = 0;
} else {
  best  = population[0].getFitness(); bestMember = 0;
  worst = population[0].getFitness(); worstMember = 0;
  start = 1;
}
for (int i = start; i < populationSize - 1; i+=2) {
  if (population[i].getFitness() > population[i+1].getFitness()) {
    if (population[i].getFitness() > best) {
      best = population[i].getFitness(); bestMember = i;
    }
    if (population[i+1].getFitness() < worst) {
      worst = population[i+1].getFitness(); worstMember = i + 1;
    }
  } else {
    if (population[i].getFitness() < worst) {
      worst = population[i].getFitness(); worstMember = i;
    }
    if (population[i+1].getFitness() > best) {
      best = population[i+1].getFitness(); bestMember = i + 1;
    }
  }
}
// if best individual from the new population is better than
// the best individual from the previous population, then
// copy the best from the new population; else replace the
// worst individual from the current population with the
```

```
    // best one from the previous generation
    if (best > theBest.getFitness()) {
      population[bestMember].copyChromosome(theBest);
      theBestGeneration = generationNum;
    } else {
      theBest.copyChromosome(population[worstMember]);
    }
  if (Debug.flag) {
      printChromosome("elitism, best (index="+bestMember+")",
        population[bestMember]);
      printChromosome("elitism, worst (index="+worstMember+")",
        population[worstMember]);
    }
  }
  private void justUpdateTheBest() {
    double currentBestFitness = population[0].getFitness();
    double next = -1;
    int currentBest = 0; // index of the current best individual
    for (int j = 1; j < populationSize; j++) {
      if ((next = population[j].getFitness()) > currentBestFitness ) {
        currentBest = j;
        currentBestFitness = next; }
    }if (currentBestFitness > theBest.getFitness()) {
      population[currentBest].copyChromosome(theBest);
      theBestGeneration = generationNum;
    }
    if (Debug.flag) {
      printChromosome("theBest (generation="+theBestGeneration+")",
        theBest);
    }
  }
}
```

ภาคผนวก ข. MCNC benchmark circuit

This directory contains the following standard cell benchmarks:

Name	YAL datafile/library	VPNR datafile/library	#cells	#nets	#I/O
SC0		fract.vpnr/db.vpnr	125	147	24
SC1	primary1.yal/sclib.yal	primary1.vpnr/sclib.db	752	904	81
SC2		struct.vpnr/db.vpnr	1888	1920	64
SC3		industry1.vpnr	2271	2594	814
SC3a		industry1a.vpnr	2271	2479	580
SC4	primary2.yal/sclib.yal	primary2.vpnr/sclib.db	2907	3029	10
SC5		biomed.vpnr/db.vpnr	6417	5766	97
SC6		industry2.vpnr	12142	13915	495
SC7		industry3.vpnr	15059	21966	375

Notes:

Please treat ALL benchmarks as 'PRIMARY' - they are all of equal importance

and cover a wide spectrum of circuit sizes.

Plot magnifications will be specified at a later time.

Benchmarks SC0, SC2, SC5, SC6, and SC7 use SCMOS MOSIS rules. The width of

the P/G rails in the cells is 6 microns.

Here are SCMOS rules:

metal_1 wire width : w1 = 3

metal_1 wire-wire spacing : s1 = 3

metal_2 wire width : w2 = 3

metal_2 wire-wire spacing : s2 = 4

via size : svia = 2

via surrounding : vsur = 1

Benchmarks SC1 and SC4 use design rules specified in sclib.yal. See the

beginng of the 'sclib.yal', even if you are using 'sclib.db'. The width

of the P/G rails in the cells is 8 microns.

Benchmark SC3 uses the following design rules:

w1 = s1 = s2 = svia = 2 microns

w2 = 2.8 microns

vsur1 = 1 micron

vsur2 = 0.4 micron

Due to the specifics of the cells used in SC3, only VPNR description is available.

Unless specified otherwise, all signal ports in all cells are on

METAL2 layer, and all power/ground ports are on METAL1.

SC0: This benchmark was used in the Physical Design Workshop 1989. It is also used in the ISCAS'89 Sequential Testability Benchmarks.

SC1: This is the old 'Primary1' benchmark from Physical Design Workshop '87 and Workshop on Placement & Routing 1988.

SC2: This circuit contains a fair amount of repetitive subcircuits.

SC3: Benchmark 'industry1' contains 115 'pass-through' signals that occur only at the pins but do not connect to any cell. If your software cannot handle such nets, please use 'industry1a' which is the same circuit, but with the 'pass-through' I/Os removed.
The cells used in SC3 differ significantly from the other cells.
The signal ports are actually metal1/metal2 vias; metal 2 can route anywhere over the cells, metal1 is forbidden over the cells.
Please route this benchmark with metal2 running verically and metal1 running horizontally (parallel to the cell rows).

SC4: This is the old 'Primary2' benchmark from Physical Design Workshop '87 and Workshop on Placement & Routing 1988.

SC5: This circuit contains a number of buses, hence it has fewer nets than cells.

SC0 and SC5: These benchmarks can be routed with or without the scan chain.
The scan chain is not connected in the benchmark files; your software is free to connect it in any manner that will be suitable. The only cell that goes into the scan chain is 'dsr2s', and the scan_in input is the second terminal of this cell. If your software cannot handle automatic scan chaining, please leave this input unconnected and make a note of it when presenting the benchmark results.

ประวัติผู้เขียน

นายอภิชาต วสุธาพิทักษ์ เกิดเมื่อวันที่ 1 มิถุนายน 2517 สำเร็จมัธยมศึกษาจาก โรงเรียน บดินทรเดชา (สิงห์ สิงหเสนี) สำเร็จวิศวกรรมศาสตรบัณฑิต (ไฟฟ้า) มหาวิทยาลัยสงขลานครินทร์ พ.ศ. 2539 ปัจจุบันทำงานที่ คณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยหัวเฉียวเฉลิมพระเกียรติ