บทที่ 3
ผลการศึกษาและทดลองซอฟต์แวร์ช่วยออกแบบ

3.1 บทนำ

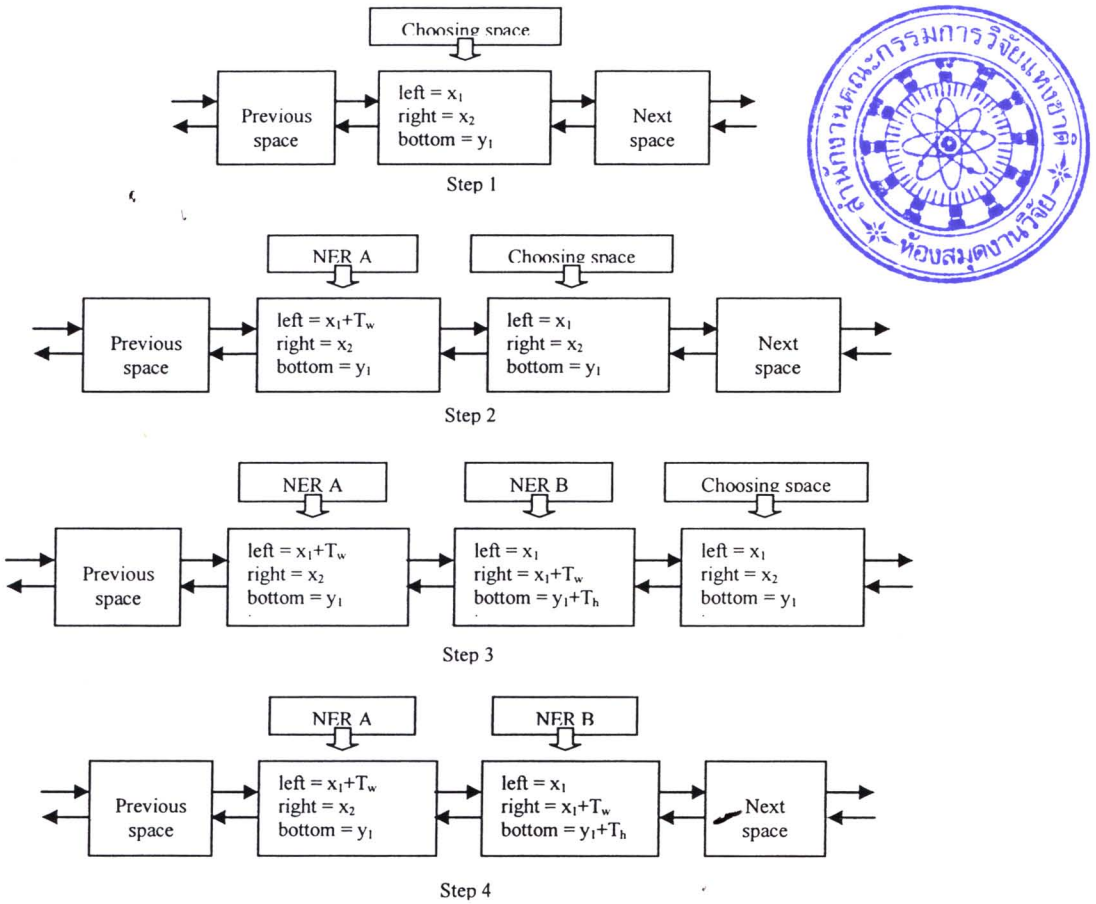
ทฤษฎีการโปรแกรมอุปกรณ์ที่สามารถโปรแกรมได้แบบ run-time หมายถึงให้สามารถโปรแกรมอุปกรณ์ฮาร์ดแวร์อย่างเช่น FPGA ให้สามารถปรับเปลี่ยนวงจรได้ในขณะที่กำลังทำงาน สามารถนำมาประยุกต์เป็นส่วนหนึ่งของการให้โพรเซสเซอร์ร่วมที่สามารถปรับเปลี่ยนโครงสร้างแบบไดนามิก ดังนั้นผู้วิจัยจึงได้ทำการศึกษาแนวคิดของการสร้างซอฟต์แวร์ที่ช่วยออกแบบ ซึ่งหน้าที่หลักของซอฟต์แวร์ที่จะช่วยออกแบบหรือทำให้โพรเซสเซอร์ร่วมสามารถปรับเปลี่ยนโครงสร้างแบบไดนามิกได้นั้นจะต้องประกอบด้วย 1)การพิจารณาการวางหน่วยประมวลผล FU ให้เหมาะสมกับฮาร์ดแวร์ที่สามารถให้งานได้ทั้งหมด นั่นคือการ Placement 2) เนื่องจากในแต่ละ FU สามารถเชื่อมต่อถึงกันผ่านทางระบบบัส ดังนั้นซอฟต์แวร์ช่วยออกแบบจึงต้องคำนึงถึงการเชื่อมโยงแต่ละ FU หรือที่เรียกว่า Routing ในงานวิจัยนี้จะศึกษาเฉพาะในส่วนของอัลกอริทึมของการวางหรือ placement ก่อนในขั้นต้น

3.2 อัลกอริทึมการ Placement

เมื่อต้องการให้โพรเซสเซอร์ร่วมที่จะต้องปรับเปลี่ยนโครงสร้างแบบไดนามิกได้ทำงานได้ พบว่าฟังก์ชันการทำงานหรือคำสั่งสำหรับการประมวลผลในแต่ละงานประยุกต์จะไม่สามารถทราบได้ล่วงหน้า ดังนั้นลำดับของคำสั่งจะรู้ในขณะที่กำลังทำงานอยู่ตอนนั้น (run time) ซึ่งจะเรียกการจัดวางหน่วย FU ลงบนอุปกรณ์ที่สามารถโปรแกรมได้ในขณะที่กำลังทำงานอยู่นั้นว่า online placement ซึ่งทำให้จะต้องทำการ placement ให้รวดเร็วและมีประสิทธิภาพมากที่สุดเพื่อให้ไม่กระทบต่อการทำงานโปรแกรมนั้นๆที่กำลังดำเนินการอยู่ ซึ่งความรวดเร็วและประสิทธิภาพของการวางนั้นจะเกี่ยวข้องกับการหาที่ว่างที่จะสามารถวางหน่วย FU ซึ่งเรียกว่า finding empty space นอกจากนี้ถ้าที่ว่างบนอุปกรณ์ที่สามารถโปรแกรมได้นั้นไม่เพียงพอหรืออาจจะมีที่ว่างแต่กระจัดกระจายเป็นชิ้นเล็กชิ้นน้อย (fragmentation) ไม่เพียงพอที่จะให้หน่วย FU ทั้งหมดจัดวางลงไปได้ ทำให้ส่งผลต่อการจัดการพื้นที่บนอุปกรณ์ที่สามารถโปรแกรมได้ โดยเฉพาะอย่างยิ่งสำหรับระบบที่สามารถโปรแกรมได้แบบไดนามิกอย่างเช่นโพรเซสเซอร์ที่กำลังออกแบบ

แนวทางของการจัดการกับที่ว่าง (maintaining empty space) บนอุปกรณ์ที่สามารถโปรแกรมได้นั้นสามารถดำเนินการได้ 3 รูปแบบ โดยที่พื้นที่ของการออกแบบเป็นลักษณะของรูปทรงสี่เหลี่ยมพื้นผ้า ฉะนั้นการจะค้นหาพื้นที่ว่างจะสามารถทำได้ดังนี้ 1) การหาพื้นที่ว่างที่ไม่ซ้อนทับกัน 2) หาพื้นที่ว่างที่มากที่สุด 3) ค้นหาจุดที่มารวมกัน ปัญหาของการจัดการพื้นที่ว่างนั้นจัดว่าเป็นปัญหาสำหรับการคำนวณทางคณิตศาสตร์ของ geometry ซึ่งผู้วิจัยได้ทำการเสนอวิธีการค้นหาจัดการกับพื้นที่ว่างที่มีประสิทธิภาพดีขึ้นกว่าเดิมด้วยการใช้ link list ในการเก็บข้อมูลพื้นที่ว่าง

การดูแลรักษาและจัดการกับพื้นที่ว่างบน FPGA จะใช้ double link list ในตอนเริ่มต้นจะมีเพียงพื้นที่ว่างเพียง 1 ชิ้นที่มีขนาดใหญ่เท่ากับพื้นที่ที่สามารถโปรแกรมได้บน FPGA แบ่งเป็น 3 ขั้นตอนได้แก่ เพิ่มพื้นที่ว่างชิ้นใหม่ ปรับรายการพื้นที่ว่างใหม่และการรวมพื้นที่ว่าง เมื่อมี task ส่วนไหน ทำงานเสร็จสิ้นแล้วออกจาก FPGA พร้อมทั้งคืนพื้นที่ว่างให้ส่วนกลางจะเรียกว่าใช้ โพรเซส *adding new space* ซึ่งจะสร้างพื้นที่ว่างแบบไม่ทับซ้อนโดยที่มุมขอบด้านซ้ายและล่างจะว่างลงบนตำแหน่งของ task ที่เสร็จสิ้นแล้ว สำหรับการรวมพื้นที่ว่างเข้าด้วยกันจะเรียกใช้ AUTO_MERGE ขั้นตอนการจัดการสามารถสรุปได้ดังรูปที่ 11



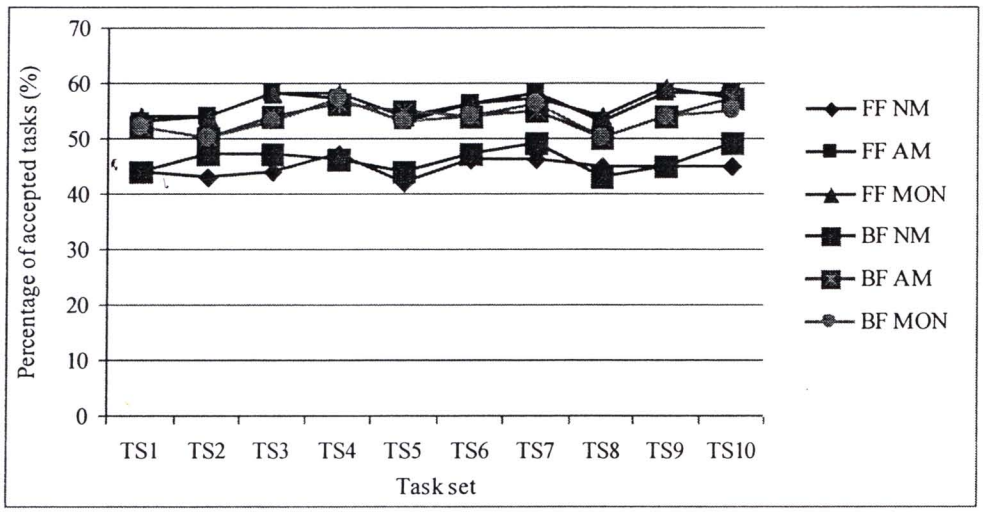
รูปที่ 11 ขั้นตอนของการวาง task

3.3 ผลการทดลองขั้นต้น

อัลกอริทึมที่ทำการทดสอบถูกสร้างขึ้นด้วยภาษาซี ทดสอบบนคอมพิวเตอร์ CPU Pentium IV 3.2 GHz หน่วยความจำ 1 GB ทดลองอัลกอริทึมการเลือกวางแบบ first fit และ best fit ถ้าหากไม่สามารถหาพื้นที่ว่างได้จะทำการ reject task นั้นทิ้งไป กำหนดให้เริ่มต้นจากมีลอจิกบล็อกของ FPGA ที่สามารถโปรแกรมได้ทั้งสิ้น 100 x 100 และชุดทดสอบจำนวน 10 ชุดกำหนดให้แต่ละชุดมี 1,000 task แบบสุ่ม มีขนาดในแต่ละ task อยู่ระหว่าง 2 – 20 หน่วย และมีเวลาในการทำงานอยู่

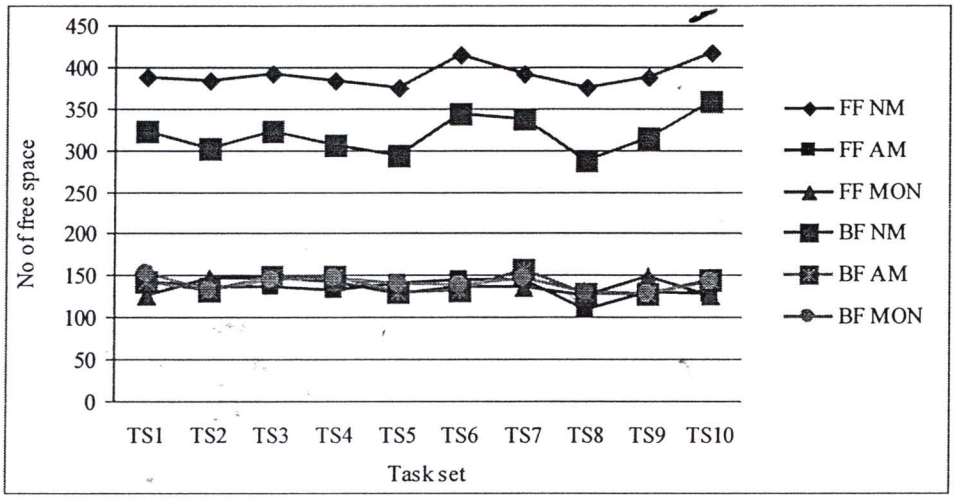
ระหว่าง 50 – 200 ต่อ task การรายงานผลจะเป็นร้อยละของงานที่สามารถหาที่ว่างได้ จำนวนของพื้นที่ว่าง และเวลาโดยเฉลี่ยของการทำงานหน่วยเป็น ns ซึ่งการทดลองจะแบ่งออกเป็น 6 กรณีเช่น

- FF NM: First Fit algorithm with No Merging space
- FF AM: First Fit algorithm with Automatic Merging space
- FF MON: First Fit algorithm with Merging if Only Need
- BF NM: Best Fit algorithm with No Merging space
- BF AM: Best Fit algorithm with Automatic Merging space
- BF MON: Best Fit algorithm with Merging if Only Need

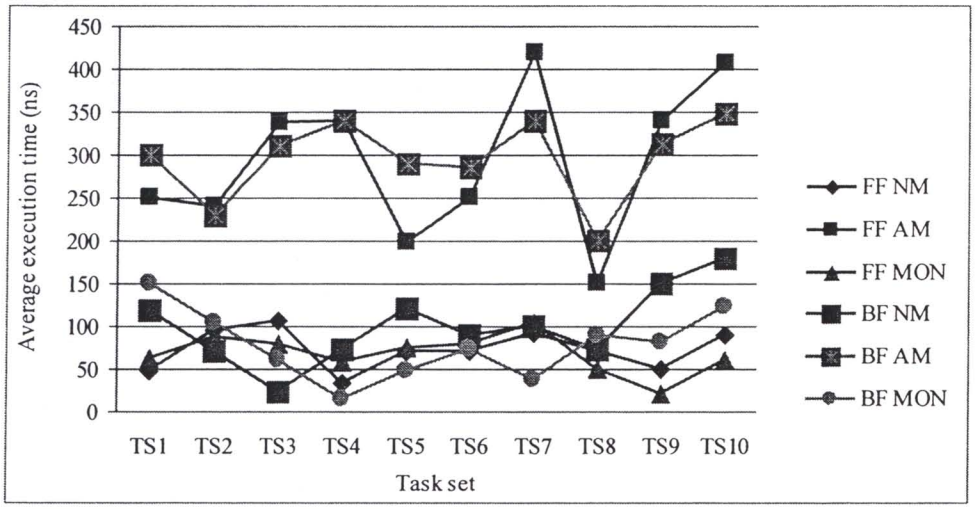


รูปที่ 12 เปอร์เซ็นต์ของงานที่สามารถหาพื้นที่ว่างได้

รูปที่ 12 แสดงร้อยละของงานที่สามารถหาพื้นที่ว่างได้ (accepted task) พบว่าอัลกอริทึมแบบ first fit ให้ผลที่ดีที่สุดในทุกกรณี ในขณะที่ first fit จะให้ผลของจำนวนพื้นที่ว่างเหลือมากกว่าอัลกอริทึมแบบอื่นๆ ดังรูปที่ 13 แต่เวลาเฉลี่ยของอัลกอริทึมแบบ best first จะใช้เวลาน้อยกว่าอัลกอริทึมอื่นๆดังรูปที่ 14



รูปที่ 13 จำนวนพื้นที่ว่างที่คงเหลือ



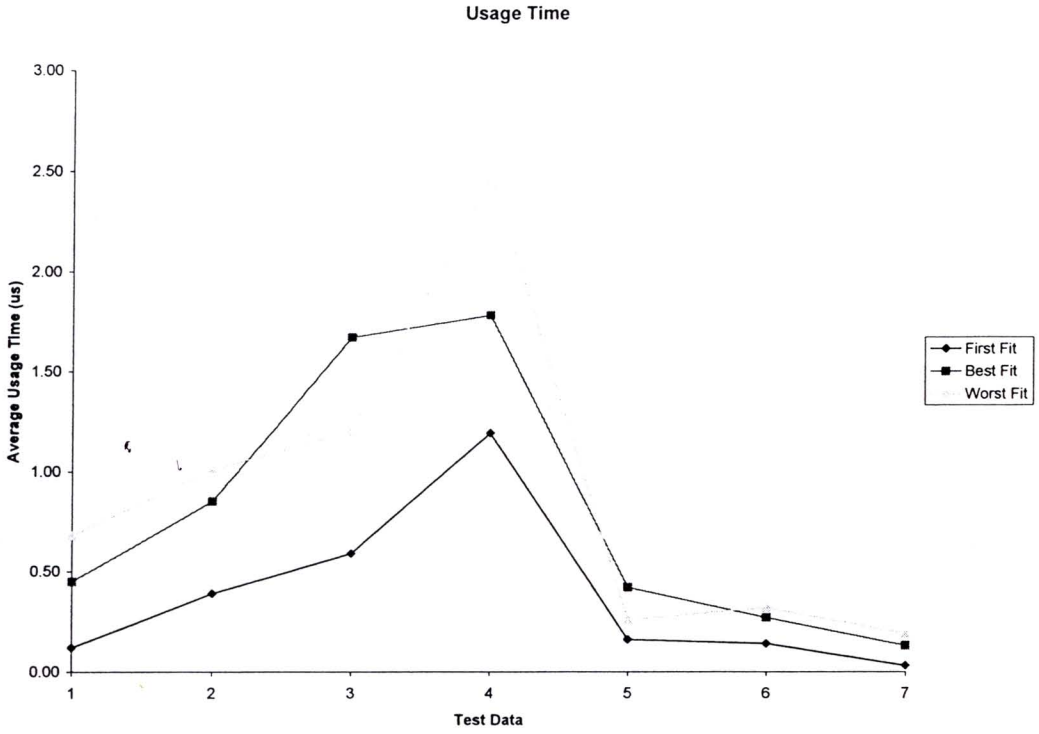
รูปที่ 14 เวลาโดยเฉลี่ยของแต่ละอัลกอริทึม

นอกจากนี้ผู้วิจัยยังได้เพิ่มเติมก็คือในส่วนของการเลือกพื้นที่ว่าง สำหรับวาง task ใหม่ที่เข้ามา โดยทำการเปรียบเทียบระหว่าง 3 algorithm คือ first fit, best fit และ worst fit ซึ่งทั้งสาม algorithm จะรับ input เป็น list ของ maximum empty rectangle และขนาดของ task ที่ต้องการจะวาง โดย first fit algorithm จะเลือกพื้นที่ว่างอันแรกใน list ที่สามารถวาง task ได้ สำหรับ best fit algorithm จะเลือกพื้นที่ว่างที่เมื่อวาง task ใหม่แล้วจะเหลือพื้นที่ที่ไม่ได้ใช้งานน้อยที่สุดและในทางตรงข้าม worst fit algorithm จะเลือกพื้นที่ว่างที่เมื่อวาง task ใหม่แล้วจะเหลือพื้นที่ที่ไม่ได้ใช้งานมากที่สุด ในกรณีที่ขนาดของพื้นที่ที่ไม่ได้ใช้งานมีขนาดเท่ากันใน best fit และ worst fit ก็จะเลือกพื้นที่อันแรกที่หาเจอ

โดยในการทดลองได้จำลองขนาดของ FPGA 1,000x1,000 และทำการสุ่มตัวอย่าง test data ขึ้นมาทั้งหมด 7 ชุด แต่ละชุดมีจำนวน task 10,000 tasks Test data ชุดที่ 1 – 4 มีขนาดของ task อยู่ระหว่าง 25x25 – 250x250 หรือคิดเป็น 1/40 – 1/4 ของพื้นที่บน FPGA โดย task life จะมีขนาดต่างกัน สำหรับ test data ชุดที่ 5-7 มีขนาดของ task อยู่ระหว่าง 25x25 – 500x500 หรือคิดเป็น 1/40 – 1/2 ของพื้นที่บน FPGA

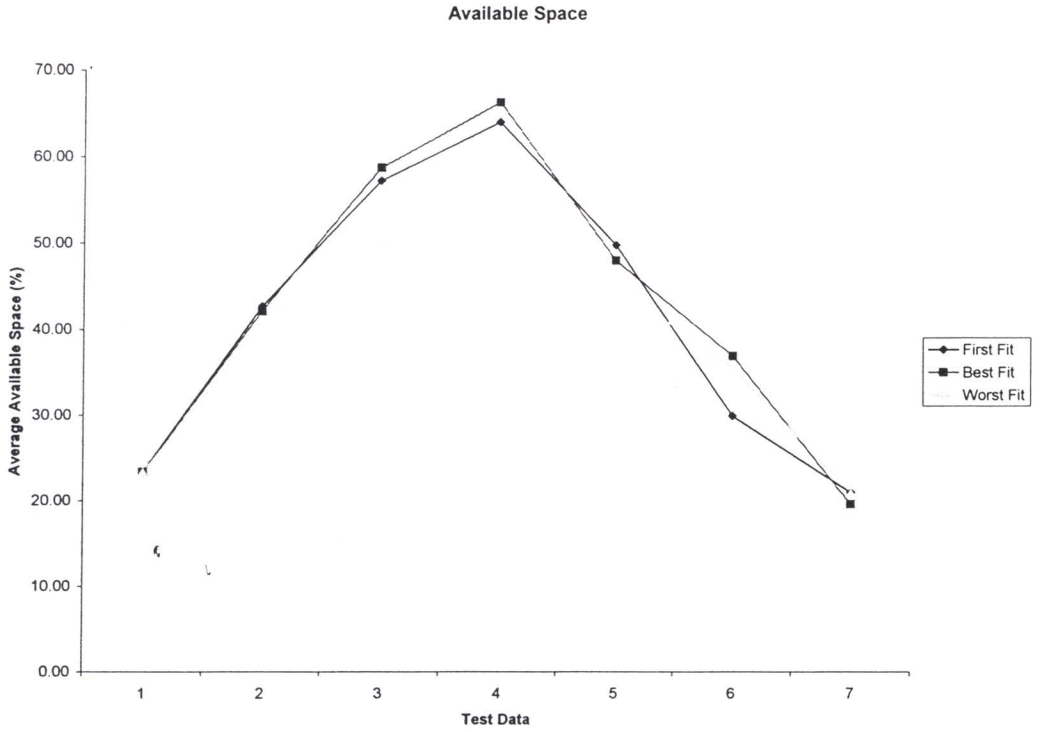
test data	board size	min task life	max task life	min task	max task	total task
1	1000x1000	10	250	25x25	250x250	10000
2	1000x1000	20	500	25x25	250x250	10000
3	1000x1000	40	1000	25x25	250x250	10000
4	1000x1000	80	2000	25x25	250x250	10000
5	1000x1000	10	250	25x25	500x500	10000
6	1000x1000	10	125	25x25	500x500	10000
7	1000x1000	10	50	25x25	500x500	10000

ซึ่งผลการทดลองที่ได้มา พบว่า เวลาเฉลี่ยที่ใช้ในการรัน first fit จะดีที่สุด รองลงมาคือ best fit และ worst fit ดังรูปที่ 15

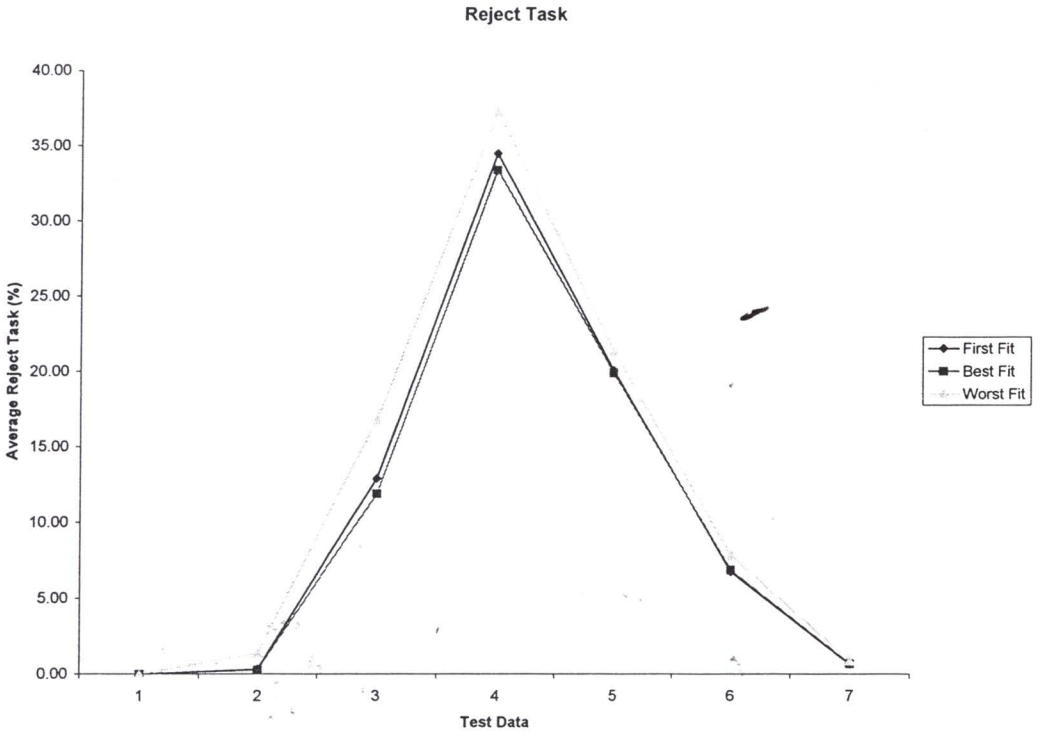


รูปที่ 15 เวลาโดยเฉลี่ยของแต่ละอัลกอริทึม

และเมื่อมาดูพื้นที่ว่างที่เหลืออยู่หลังจากทำการแทรก task ทั้ง 10,000 tasks แล้ว กลับพบว่า worst fit มีแนวโน้มที่ดีที่สุด ที่เป็นเช่นนี้เพราะว่าใน worst fit มีจำนวน task ที่ถูก rejected มากที่สุดนั่นเอง และเมื่อเปรียบเทียบระหว่าง first fit กับ best fit พบว่า มีขนาดพื้นที่ว่างและจำนวน task ที่ถูก rejected ไม่ได้แตกต่างกันมากนัก ดังรูปที่ 16 และ 17



รูปที่ 16 พื้นที่ที่เหลือโดยเฉลี่ยของแต่ละอัลกอริทึม



รูปที่ 17 อัตราการ rejected task ของแต่ละอัลกอริทึม

3.4 เอกสารอ้างอิง

[1] P. Healy and M. Creavin. An Optimal Algorithm for Rectangle Placement. In Technical Report UL-CSIS-97-1. Dept. of Computer Science and Information Systems, University of Limerick, Limerick, Ireland, 1997.

[2] M. Orlowski. A New Algorithm for the Largest Empty Rectangle. In Algorithmica, volume 5, pages 65–73, 1990.

[3] K. Bazargan, R. Kastner, and M. Sarrafzadeh. Fast Template Placement for Reconfigurable Computing Systems. In IEEE Design and Test of computers Special Issue on Reconfigurable Computing, volume 17(1), pages 68–83, Jan.-Mar. 2000.

[4] M. Handa, and R. Vemuri, "An efficient algorithm for finding empty space for online FPGA placement". In Proceedings of the 41st annual conference on design automation, San Diego, CA, June 2004, pp 960-965.

[5] J. Cui, Q. Deng, X. He, and Z. Gu, "An efficient algorithm for online management of 2D area of partially reconfigurable FPGAs". In Proceedings of the conference on Design, automation and test in Europe, Nice, France, April 2007, pp 1-6.

[6] Y. Lu, T. Marconi, G. Gaydadjiev, and K. Bertels, "An efficient algorithm for free resources management on the FPGA". In Proceedings of the conference on Design, automation and test in Europe, Munich, Germany, March 2008, pp 1095-1098.

[7] M. Gericota, G. Alves, M. Silva and J. Ferreira, "Runtime management of logic resources on reconfigurable systems", In Proceedings of Design, Automation and Test in Europe, March, 2003.

[8] M. Handa, and R. Vemuri, "Area fragmentation in reconfigurable operating systems", In Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms, CSREA Press, June, 2004.