193003

ปัญหาคอนเวกซ์ฮัลสองมิติ (2-D Convex Hull Problem) เป็นปัญหาหลักปัญหาหนึ่งทางด้านการคำนวณทางเรขาคณิต ซึ่งมีงานวิจัยก่อนหน้านี้หลายงานวิจัยที่นำเสนอขั้นตอนวิธีสำหรับแก้ปัญหานี้บนระบบประมวลผลแบบขนาน แต่ข้อเสียคือจำเป็นต้องใช้หน่วยประมวลผลเป็นจำนวนมาก บางงานวิจัยใช้จำนวนหน่วยประมวลผลมากถึงกำลังสามของข้อมูลนำเข้า ($N^2 \times N$) ดังนั้นจึงมีแนวความคิดที่จะพัฒนาขั้นตอนวิธีการหาผลลัพธ์ของปัญหานี้บนระบบแบบขนาน โดยใช้จำนวนหน่วยประมวลผลน้อยลง

งานวิจัยนี้จึงได้นำขั้นตอนวิธีการหาคอนเวกซ์ฮัลสองมิติแบบขนานบนระบบเครือข่ายแบบตาข่ายรีคอนฟิกิวเรเบิล (Reconfigurable Mesh) ขนาด $N \times N$ โดยใช้เวลาคงตัว (O(1)) ที่มีผู้พัฒนาไว้แล้วมาพัฒนาต่อ ซึ่งจะใช้วิธีการหาผลลัพธ์แบบเป็นรอบการทำงาน ใช้เทคนิคแบ่งและได้ชัยชนะ (Divide and Conquer) และนำผลลัพธ์จากทุกรอบการทำงานมาผสานกัน (Merge) โดยขั้นตอนวิธีที่เสนอในวิทยานิพนธ์นี้เป็นวิธีทั่วไปที่ใช้จำนวนหน่วยประมวลผลน้อยลงกว่าวิธีเดิม ($N \times \sqrt{N}$) และมีความซับซ้อนด้านเวลาเท่ากับ $O(\sqrt{N})$ และเมื่อใช้จำนวนหน่วยประมวลผลเท่าวิธีเดิม ($N \times N$) ก็จะมีความซับซ้อนด้านเวลาเท่ากับ O(1) ด้วยเช่นกัน

193003

A two dimensional (2-D) Convex Hull problem is one of the most important problems of the computational geometry. Many previous studies presented algorithms for solving this problem on the parallel processing systems of size $N \times N$ processors in O(1) time. However, the weakness of those algorithms is the requisite of a large number of processors -- some algorithms use the third power as many processors as the size of input data ($N^2 \times N$). Therefore, the idea to develop and improve algorithms using less processors for solving this problem will be considerably discussed.

The purpose of this research is to improve the recently studied parallel 2-D Convex Hull algorithm solved in O(1) time on the $N \times N$ reconfigurable mesh. In particular, this algorithm applies the Divide and Conquer technique and merging to merge all outcomes from each cycle. The desired algorithm will use less processors than the previous one ($N \times \sqrt{N}$) and can be performed in $O(\sqrt{N})$ time. In addition, if $N \times N$ processors are used, it can be performed in O(1) time.