การศึกษาการออกแบบระดับสูงสำหรับระบบที่มีตัวประมวลผลหลายชนิด

The Study of High-Level System Design for Heterogeneous Architecture

เศรษฐกาล โปร่งนุช
สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะเทคโนโลยีอุตสาหกรรม มหาวิทยาลัยราชภัฏสวนสุนันทา

และ **ธีรยศ เวียงทอง**
ภาควิชาวิศวกรรมอิเล็กทอนิกส์ มหาวิทยาลัยเทคโนโลยีมหานคร

Manuscript received September 4, 2014
Revised October 2, 2014

บทคัดย่อ

บทความนี้นำเสนอเรื่องราวเกี่ยวกับการออกแบบระดับสูงสำหรับระบบที่มีตัวประมวลผลหลายชนิดที่สามารถช่วยในการประมวลผลข้อมูลได้เร็วขึ้นเนื่องจากสามารถทำหลาย ๆ งานได้พร้อมกัน ระบบที่ซับซ้อนนี้อาจประกอบด้วยตัวประมวลผลแบบต่าง ๆ ทั้ง CPUs GPUs และ FPGAs หรือ DSPs ถูกต่อเชื่อมกันเป็นโครงสร้างแบบต่างๆ ซึ่งวิธีการพัฒนาออกแบบจำเป็นจะต้องใช้ภาษาระดับสูงสำหรับการอธิบายการทำงานของส่วนต่างๆ ในระบบ เพื่อความสะดวกและรวดเร็วในการออกแบบ ซึ่งจะเห็นว่าภาษา OpenCL กำลังเป็นที่นิยมเพิ่มมากขึ้นเรื่อย ๆ เนื่องจากสามารถนำมาใช้กับระบบที่มีสถาปัตยกรรมที่มีตัวประมวลผลหลายชนิดได้หลากหลาย โดยมีตัวอย่างการนำไปใช้งานกับพาราเลลล่าบอร์ดสำหรับแอปพลิเคชันข้อมูลที่มีการคำนวณที่ซับซ้อนซึ่งจำเป็นต้องใช้การประมวลผลแบบขนานเพื่อให้ได้ผลลัพธ์ที่เร็วขึ้น

คำสำคัญ : *การออกแบบระดับสูง; ระบบที่มีตัวประมวลผลหลายชนิด; OpenCL;*

ABSTRACT

This paper presents the study of high-level system design for heterogeneous architecture which is increasingly popular because it can accelerate computational intensive applications implemented on different types of processors. The system may consist of CPUs, GPUs, FPGAs or DSPs connected by buses and sharing memories. High level design languages are required for developing in such complex system since designers can take advantages of a short design time also quick design assessments. The most promising OpenCL design language which is highlighted in this paper can benefit more varieties of implementing in Heterogeneous System Architectures (HSA). The Parallella development board is introduced. It will be used as a heterogeneous platform for accelerating computations in data dominated applications.

Keywords : *high-level system design; heterogeneous architecture; OpenCL;*

1. บทนำ

ในปัจจุบันข้อมูลสารสนเทศมีปริมาณมากมายมหาศาล ทำให้จำเป็นต้องใช้เทคโนโลยีที่มาจัดการและประมวลผลข้อมูลตามความต้องการในการใช้งานต่าง ๆ ได้ เช่น เวลาที่ใช้ในการประมวลผล รองรับการประมวลผลแบบเวลาจริง รองรับการทำงานแบบขนาน ความน่าเชื่อถือได้ พลังงานที่ใช้ ตลอดจนมีราคาหรือต้นทุนต่ำเป็นต้น ซึ่งการที่จะออกแบบระบบที่รองรับการประมวลผลข้อมูลตามความต้องการต่าง ๆ ได้นั้น จะต้องอาศัยความรู้ความเข้าใจในการทำงานที่จะต้องมีการผสมผสานกันของตัวประมวลผลที่มีมากมายหลายแบบทั้งในส่วนของซอฟต์แวร์ และส่วนของฮาร์ดแวร์ มีการนำมาใช้งานอย่างเหมาะสม สามารถรองรับการทำงานแบบขนาน และมีการจัดการทำงานภายในระบบเพื่อให้สามารถทำงานร่วมกันได้ โดยเฉพาะตอบสนองความต้องการในการใช้งานกับแอปพลิเคชันที่ต้องการการคำนวณที่มีประสิทธิภาพสูง เช่น การประมวลผลภาพวิดีโอความละเอียดสูง (HD Video Processing) การเข้ารหัสถอดรหัส (Codec) การแปลงข้อมูล (Transformation) ในแอปพลิเคชันต่าง ๆ เป็นต้น

บทความนี้จะนำเสนอ ข้อมูลต่าง ๆ เกี่ยวกับการออกแบบระดับสูงสำหรับระบบที่มีตัวประมวลผลหลายชนิด หรือที่เรียกว่า

HSA (Heterogeneous System Architectures) จากงานวิจัยต่างๆ รวมทั้งระบบที่มีอยู่ในท้องตลาด มีการเปรียบเทียบคุณลักษณะความสามารถในด้านต่าง ๆ และแนวทางในการพัฒนาเพื่อใช้งาน โดยในหัวข้อถัดไปจะกล่าวถึงการทบทวนวรรณกรรมและทฤษฎีที่เกี่ยวข้อง ตัวอย่างการออกแบบระบบที่มีตัวประมวลผลหลายชนิด และบทสรุปตามลำดับ

2. ทฤษฎีที่เกี่ยวข้อง

การศึกษาการออกแบบระดับสูงสำหรับระบบที่มีตัวประมวลผลหลายชนิดในบทความนี้จะแบ่ง ออกเป็น 2 ส่วน คือ งานวิจัยทางด้านวิธีการออกแบบระบบด้วยภาษาระดับสูง (Design Methodologies) และงานวิจัยทางด้านโครงสร้างระบบต่างๆ ที่มีตัวประมวลผลหลายชนิด (System Architectures) ซึ่งมีรายละเอียดดังต่อไปนี้

2.1 High-Level System Design Methodologies

การออกแบบระบบในระดับต่ำ (Low-level Designs) ทั้งในระดับของทรานซิสเตอร์ หรือในระดับรีจีสเตอร์ จำเป็นต้องอาศัยความชำนาญของผู้ออกแบบ และใช้เวลาค่อนข้างนานกว่าที่จะได้ออกมาเป็นต้นแบบ ปัจจุบันการออกแบบนิยมทำกันในระดับที่สูงขึ้น โดยผู้ออกแบบสามารถใช้ภาษาชั้นสูง เช่น SystemC, OpenCL, OpenMP, VHDL, Verilog ในการบรรยายพฤติกรรมการทำงานทั้งในส่วนของฮาร์ดแวร์ และ/หรือในส่วนของซอฟต์แวร์ โดยมีเครื่องมือในการออกแบบที่เรียกว่า High-Level Synthesis Tools ช่วยในการดำเนินการออกแบบและสร้างระบบที่มีความซับซ้อนขึ้นมาได้อย่างหลากหลาย และรวดเร็ว [1] เช่นระบบที่ออกแบบร่วมกันระหว่างฮาร์ดแวร์กับซอฟต์แวร์ที่ทำงานร่วมกับเทคโนโลยีทางด้านความปลอดภัย (Cyber-Physical System: CPS) ระบบที่ออกแบบร่วมกันระหว่างฮาร์ดแวร์กับซอฟต์แวร์สำหรับนาโนอิเล็กทรอนิกส์ (Codesign for Nanoelectronic System) ระบบที่ออกแบบร่วมกันระหว่างฮาร์ดแวร์กับซอฟต์แวร์ที่สามารถปรับเปลี่ยนการทำงานได้ตามเวลาจริง (Codesign of Runtime Adaptive System) ในส่วนงานวิจัย [2] ได้นำเสนอการใช้งาน Bluespec, Vivado และ SCORE ที่เป็น High-Level Synthesis Tools ผ่านทางโฮสต์ (Host) ที่เป็นคอมพิวเตอร์ส่วนบุคคลกับเอฟพีจีเอที่รองรับการเชื่อมต่อผ่านทาง PCIe อีเทอร์เน็ต (Ethernet) เพื่อจัดการ DRAM และการปรับเปลี่ยนการทำงาน (Reconfiguration) งานวิจัย [3] นำเสนอเรื่องซอฟต์แวร์ที่สามารถโปรแกรมได้บนชิปของ Xilinx ในตระกูล Zynq เพื่อจัดการเสียงรบกวนทางดิจิทัลก่อน นำเสนอการใช้ Vivado มาสร้างระบบเพื่อจัดการเสียงรบกวนทางดิจิทัลก่อน (Digital Pre-Distortion: DPD) ผลลัพธ์ที่ได้คือสามารถลดระยะเวลาการออกแบบระบบดังกล่าวได้ถึง 7 เท่า ส่งผลดีต่อการนำผลิตภัณฑ์ที่ผลิตไปขายสู่ท้องตลาดได้อย่างรวดเร็ว

สำหรับระบบที่มีตัวประมวลผลหลายชนิดนั้นมีภาษาระดับสูงสำหรับช่วยในการออกแบบและจัดการระบบ อาทิเช่น CUDA, OpenCL, OpenMP, OpenMPI เพื่อให้ตัวประมวลผลแต่ละตัวภายในระบบทำงานร่วมกันได้อย่างมีประสิทธิภาพดีที่สุด สามารถรองรับการประมวลผลข้อมูลที่มีปริมาณมหาศาลในอนาคต บทความนี้ให้ความสนใจ OpenCL (Open Computing Language) มากกว่าภาษาระดับสูงแบบอื่น ๆ เนื่องจากถูกออกแบบมาเพื่อใช้ได้กับ HSA ได้หลายแบบ ซึ่งต่างกับภาษา CUDA ที่สนับสนุนการทำงานกับตัวประมวลผลกราฟิกส์ของบริษัท Nvidia เพียงอย่างเดียว ส่วน OpenMP รองรับการทำงานที่ใช้ตัวประมวลผลชนิดเดียว (Homogeneous) เท่านั้น ทางด้านภาษา OpenMPI จะเหมาะสมสำหรับใช้งานด้านการประมวลผลในรูปแบบคลัสเตอร์ (Cluster) นอกจากนี้ ข้อดีอีกอย่างหนึ่งของ OpenCL คือสามารถทำงานร่วมกับ High-Level Synthesis Tools ของบริษัทชั้นนำที่ผลิตอุปกรณ์ที่มีตัวประมวลผลหลายชนิด เช่น Vivado Design Suite ของบริษัท Xilinx, Epiphany SDK ของบริษัท Adapteva และ Altera SDK for OpenCL ของบริษัท Altera

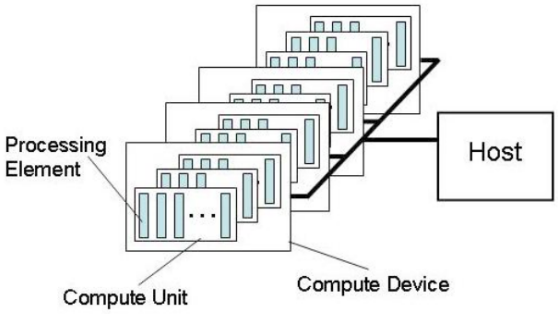
OpenCL [4] เป็นภาษาโปรแกรมที่ใช้แยกงานจากการประมวลผลตามปรกติไปให้หน่วยประมวลผลกราฟิกส์ (Graphics Processing Unit: GPU) และ หน่วยประมวลผลกลาง (Central Processing Unit: CPU) อื่นๆ ประมวลผลร่วมด้วยกัน เพื่อใช้หน่วยประมวลผลกราฟิกส์ในการประมวลผลอื่น ๆ นอกเหนือไปจากการประมวลผลคอมพิวเตอร์กราฟิกส์ตามปรกติ พัฒนาขึ้นโดย Khronos Group ร่วมกับบริษัทอื่น ๆ โดยมีภาษา C99 เป็นพื้นฐาน โดยบริษัทแอปเปิลได้เสนอให้ Khronos Group เป็นตัวกลางเพื่อกำหนด OpenCL เป็นมาตรฐานเพื่อจะได้ใช้งานใน Mac OS 10.6

โมเดลของการนำ OpenCL ไปประยุกต์ใช้งานบนระบบประมวลผลที่มีหลายชนิด สามารถแบ่งออกเป็นสองส่วน [5] คือ โฮสต์ (Host) และอุปกรณ์ประมวลผล (Compute Devices) ดังรูปที่ 1 โดยแต่ละอุปกรณ์ประมวลผลจะประกอบด้วยส่วนย่อย ๆ เรียกว่า Processing Elements ประกอบกันเป็นกลุ่มเรียกว่า Compute Units

สถาปัตยกรรมที่มีตัวประมวลผลหลายชนิดที่ใช้ OpenCL ได้แก่ งานวิจัย [6] ที่นำเสนอการศึกษาความเป็นไปได้ที่จะนำ OpenCL ที่เป็นแอปพลิเคชันโปรแกรมอินเตอร์เฟส (Application Programming Interface: API) มาจัดการทำงานกับระบบที่มีตัวประมวลผลหลายชนิดที่มีทั้ง CPUs GPUs และ FPGAs

ในบทความ [7] ได้นำเสนอการเปรียบเทียบประสิทธิภาพของระบบต่าง ๆ ในการแก้ปัญหาทางวิทยาศาสตร์ที่เรียกว่า Quantum Monte Carlo โดยได้สร้างตัวฮาร์ดแวร์ที่ช่วยคำนวณ (Hardware Accelerator) ที่สามารถทำงานในลักษณะขนานกัน (Parallel

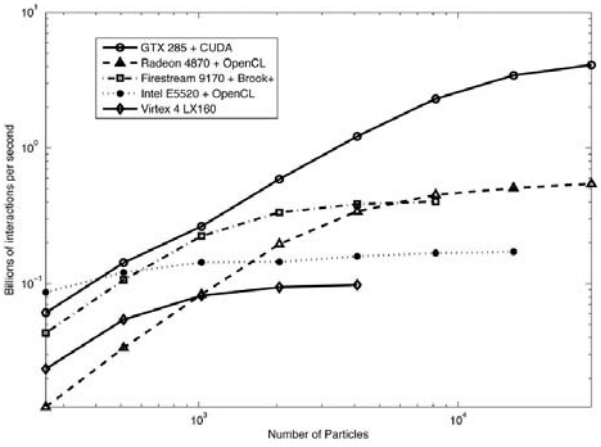
Processing) ด้วยภาษาระดับสูงต่าง ๆ ได้แก่ 1) การใช้งานภาษา CUDA สำหรับการสร้างตัวคำนวนใน GTX285 GPU 2) การใช้งานภาษา Brook+ สำหรับใช้งานใน Firestream9170 - ATI Graphic Accelerator 3) การใช้งานภาษา OpenCL สำหรับการประมวลผลของ Radeon4870 - Graphic Accelerator และใน Intel E5520 - Multicore Processor รวมทั้ง 4) การใช้งาน VHDL ในการสร้างตัวคำนวณใน Virtex 4 LX160 - Xilinx FPGA ซึ่งผลการเปรียบเทียบแสดงได้ในรูปที่ 2 จะเห็นได้ว่า GTX285 มีการประมวลผลได้เร็วที่สุด รวมทั้งพัฒนาได้เร็วกว่าเมื่อเทียบกับการสร้างใน FPGA อย่างไรก็ตาม เมื่อพูดถึงลักษณะของ API (Application Programming Interface) ที่อยู่ใน SDK (Software Development Kit) ในแต่ละตัว ของ Brook+ จะไม่ค่อยซับซ้อน ขณะที่ CUDA และ OpenCL โครงสร้างจะค่อนข้างใกล้เคียงกัน โดยที่ OpenCL จะมีฟังก์ชันให้เลือกใช้งานมากที่สุด เนื่องจากเป็นภาษาที่ถูกออกแบบมาเพื่อใช้งานได้ในหลาย ๆ รูปแบบที่เป็นลักษณะ Multiprocessors ได้มากกว่า CUDA ที่ใช้เฉพาะใน GPU Nvidia ซึ่งทำให้ภาษา OpenCL ค่อนข้างเป็นที่น่าสนใจในการนำมาใช้งานในปัจจุบัน



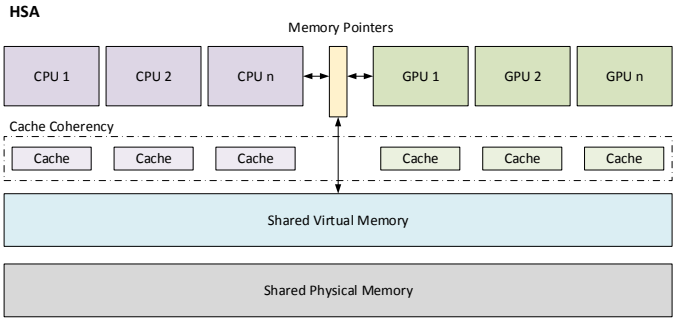
รูปที่ 1 *โครงสร้างระบบที่ใช้กับภาษา OpenCL*

2.2 โครงสร้างที่มีตัวประมวลผลหลายชนิด (Heterogeneous Multiprocessing System)

ระบบที่มีตัวประมวลผลหลายชนิด สามารถสร้างขึ้นมาได้ด้วย CPUs, DSPs, GPUs, ASICs, FPGAs หรือตัวประมวลผลอื่นๆ ที่มีโครงสร้างการเชื่อมต่อ และการจัดการทรัพยากรรูปแบบต่าง ๆ โดยมีวัตถุประสงค์หลัก 4 ข้อ [8] คือ 1) ต้องการให้ระบบมีประสิทธิภาพที่ดีในด้านพลังงานและเวลาในการประมวลผล 2) เพื่อง่ายต่อการโปรแกรมสั่งการตัวประมวลผลแต่ละตัวภายในระบบที่มีตัวประมวลผลหลายชนิด 3) เพิ่มความสะดวกในการเขียนโค้ดในแต่ละระบบปฏิบัติการ และ 4) เพื่อรองรับการใช้งานอย่างแพร่หลายในอุตสาหกรรมแบบต่าง ๆ ทั้งนี้การเลือกใช้งานตัวประมวลผลชนิดต่าง ๆ นั้น จะต้องเลือกมาใช้ให้สอดคล้องกับชนิดของงานที่จะประมวลผล โครงสร้างการเชื่อมต่อกัน ระบบบัส (Bus System) หรืออื่น ๆ เพื่อให้ระบบโดยรวมสามารถทำงานได้ดีที่สุด ทั้งในด้านความเร็ว พลังงานที่ใช้ รวมทั้งราคา



รูปที่ 2 *การเปรียบเทียบประสิทธิภาพของตัวช่วยคำนวณต่าง ๆ*



รูปที่ 3 *สถาปัตยกรรมของหน่วยความจำภายในระบบ HAS*

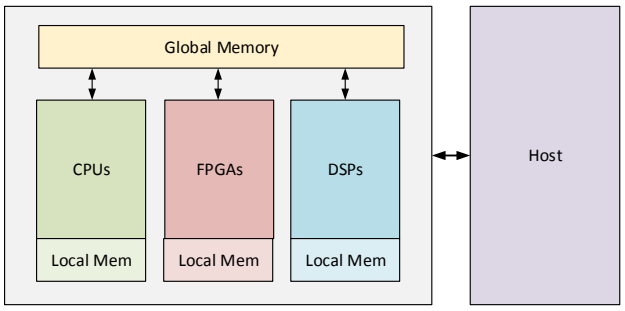
สถาปัตยกรรมหรือโครงสร้างที่มีตัวประมวลผลหลายชนิด (Heterogeneous System Architecture: HSA) ในรูปแบบทั่วไปๆ ดังรูปที่ 3 ได้แก่ CPUs GPUs และตัวประมวลผลอื่นๆ ที่สามารถทำงานร่วมกันได้ ผ่านทางตัวควบคุมที่สามารถกำหนดการดำเนินกิจกรรมต่าง ๆ ในระบบได้ เช่นโปรแกรมที่เขียนโดยใช้ API ด้วยโปรโตคอลการทำงานลักษณะต่าง ๆ ให้สอดคล้องกันทั้งจังหวะและเวลา โดยเฉพาะการบริหารจัดการทรัพยากรที่ใช้งานร่วมกัน การทำ Context Switching ในกรณีที่มีการส่งงานที่ประมวลผลจากตัวประมวลผลตัวใดตัวหนึ่งสำเร็จ ก็จะส่งงานต่อไปยังตัวประมวลผลตัวอื่น ๆ จะต้องทำการบันทึกค่าการจัดลำดับความสำคัญของงานนั้น ๆ จากตัวประมวลผลตัวเดิมไปสู่ตัวประมวลผลตัวใหม่ และจากตัวประมวลผลตัวใหม่กลับมาที่ตัวประมวลผลตัวเดิม

นอกจากตัวประมวลผล แล้วอีกส่วนหนึ่งที่สำคัญใน HSA คือ

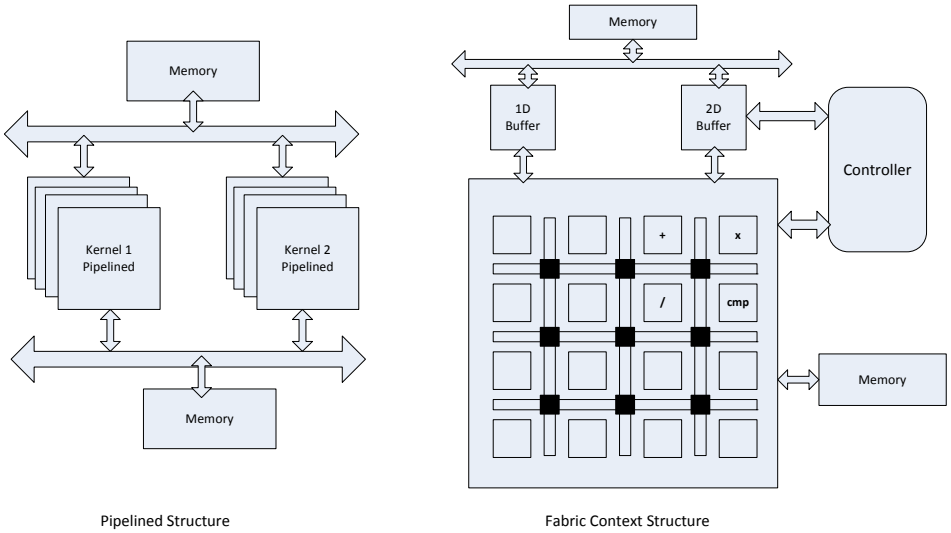
หน่วยความจำ ซึ่งโดยทั่วไปแล้วจะมีหน่วยความจำที่เป็น Local memory ที่จะใช้เฉพาะตัวประมวลผลนั้น ๆ เพื่อให้ง่ายสำหรับการจัดการและทำงานได้อย่างอิสระกับตัวประมวลผลแต่ละตัว ซึ่งอาจรวมถึงหน่วยความจำแคชที่ติดกับหน่วยประมวลผล หน่วยความจำอีกประเภทหนึ่งคือ Shared Memory หรือ Global Memory ที่แต่ละหน่วยประมวลผลสามารถเข้ามาใช้ร่วมกันได้ เพื่อแลกเปลี่ยนข้อมูลระหว่างกัน ซึ่งจำเป็นในการทำงานแบบขนาน

ตัวอย่างที่นำเสนอใน [9] เป็นการพัฒนาตัวสังเคราะห์วงจรสำหรับการสร้างระบบใน FPGA โดยใช้ภาษา OpenCL โดยโครงสร้างได้เปลี่ยนจากแบบ Pipeline Kernels มาเป็นแบบ Fabric Context ดังแสดงในรูปที่ 5 ที่มีตัวกระทำ (Operations) ต่าง ๆ เชื่อมต่อกันไว้ในลักษณะของอาร์เรย์ เพื่อสะดวกในการคอมไพล์และสังเคราะห์วงจรให้เร็วขึ้นกว่าเดิมถึงสี่พันกว่าเท่า โดยวงจรที่ได้จะใช้ทรัพยากรมากกว่าเดิมประมาณ 1.8 เท่าเมื่อเทียบกับรูปแบบเดิม

บทความนี้เสนอแนวคิดที่จะออกแบบระบบที่มีตัวประมวลผลหลายชนิด ซึ่งจะประกอบด้วย หน่วยประมวลผลกลางแบบ Multicores หน่วยประมวลผลสัญญาณดิจิทัล (Digital Signal Processing: DSP) และหน่วยประมวลผลฮาร์ดแวร์ใน FPGAs เพื่อให้รองรับกับการจัดการระบบด้วย OpenCL รวมถึงรองรับการประมวลผลข้อมูลต่าง ๆ เช่น ข้อมูลภาพ หรือวีดีโอ หรือการคำนวนที่มีความซับซ้อนสูง โดยสถาปัตยกรรมที่ออกแบบในเบื้องต้นสามารถแสดงดังรูปที่ 4 โดยจะเน้นไปในด้านการพัฒนาโครงสร้างการเชื่อมต่อ การใช้งานหน่วยความจำ และการจัดการงานต่างๆ ในระบบที่ง่าย และเหมาะสมที่สุด เพื่อให้สามารถประยุกต์ใช้งานได้อย่างกว้างขวางทางด้านการประมวลผลข้อมูลต่าง ๆ โดยพยายามนำเอาส่วนของระบบที่มีอยู่ในท้องตลาดมาพัฒนาเพิ่มเติมโดยให้เกี่ยวข้องในส่วนของการพัฒนาวงจรฮาร์ดแวร์น้อยที่สุด ซึ่งสามารถทำได้ในโครงสร้าง FPGA ที่เป็น Reconfiguration Computing Structure อยู่แล้ว



รูปที่ 4 *ระบบที่มีตัวประมวลผลหลายชนิดรองรับการจัดการด้วย OpenCL*



รูปที่ 5 *โครงสร้างแบบ Pipelined Kernels และ Fabric Contexts*

3 ตัวอย่างระบบที่มีตัวประมวลผลหลายชนิด PARALLELLA BOARD

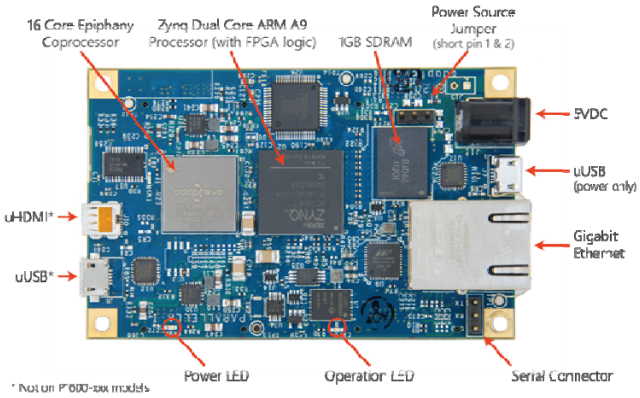
พาราเลลล่าบอร์ด (Parallella Board) [10] เป็นบอร์ดทดลองที่รองรับการประมวลผลขั้นสูง ภายในมีชิปเอฟพีจีเอรุ่น Zynq ของบริษัท Xilinx ซึ่งภายในชิปเอฟพีจีเอที่ถูกออกแบบเป็นระบบภายในชิป (System-On-Chip: SoC) ประกอบด้วยหน่วยประมวลผลที่รองรับการประมวลผลทั่วไปๆ (General Propose Possessor) รุ่น ARM-A9 ที่มีแกนสำหรับประมวลผลจำนวน 2 แกน และ Epiphany Multicore Coprocessor ของบริษัท Adapteva ที่มีแกนสำหรับประมวลผลจำนวน 16 แกน แสดงดังรูปที่ 6, 7 โดยข้อดีของบอร์ดนี้คือ มีขนาดกระทัดรัด ราคาถูก ตัวประมวลผลใช้พลังงานต่ำ มีพอร์ตการเชื่อมต่อแบบ HDMI Ethernet USB และ GPIO 48 พอร์ต รองรับภาษาตามมาตรฐาน ANSI C/C++ และรองรับแอปพลิเคชันโปรแกรมอินเตอร์เฟสแบบ OpenCL มี SDK (Software Development Kits) ที่จำเป็นสำหรับการพัฒนาโปรแกรม

3.1 Zynq FPGA

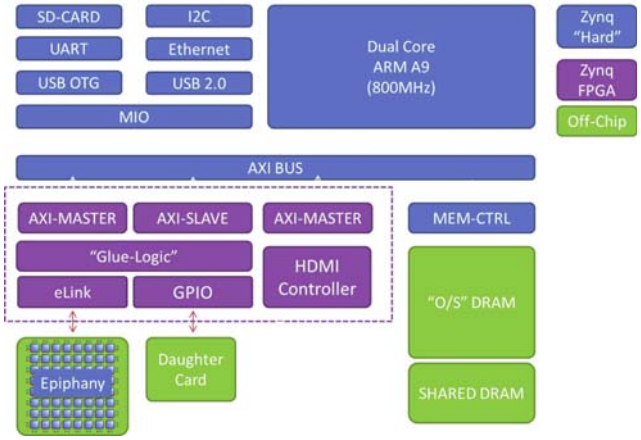
เป็นชิปของบริษัท Xilinx ที่มี ARM Processor อยู่ภายใน โดยเป็นรุ่น ARM Cortex-A9 จำนวน 2 แกนประมวลผล มีหน่วยความจำแคช (Cache) 2 ระดับ ขนาด 32 และ 512 KB ภายในชิปมีหน่วยความจำขนาด 512 KB รองรับการเชื่อมต่อกับหน่วยความจำภายนอกรุ่นล่าสุดคือ DDR3 มีจำนวนลอจิกเซลล์ (Logic Cells) จำนวน 28,000 - 444,000 ลอจิกเซลล์ และหน่วยประมวลผลสัญญาณดิจิทัลจำนวน 80 – 2,000 slices เป็นต้น ซึ่งชิปเอฟพีเอรุ่น Zynq เป็นองค์ประกอบหลักที่สำคัญของพาราเลลาบอร์ด สามารถออกแบบพัฒนาผ่าน Vivado Design Suite [11], [12] ที่พัฒนามาจาก Xilinx AutoESL สามารถแปลภาษาระดับสูง เช่น C/C++, System- C ไปเป็นรูปแบบระดับรีจีสเตอร์ ก่อนสร้างเป็น Netlist ไฟล์ เพื่อไปโปรแกรม (Configuration) ลงในอุปกรณ์

3.2 Epiphany Multicore Coprocessor

เป็น Multicore Coprocessor ที่มีแกนประมวลผลจำนวน 16 แกน โดยมีชุดซอฟต์แวร์สำหรับการพัฒนาเรียกว่า Epiphany SDK หรือ eSDK [13] ที่จะเป็นตัวจัดการการทำงานของแกนประมวลผลต่าง ๆ เช่น การใช้งานทรัพยากรบนตัวประมวลผล การแบ่งปันหน่วยความจำ และการคำนวณต่าง ๆ ที่เกิดขึ้นภายในตัวประมวลผล ซึ่ง Epiphany SDK นี้ประกอบด้วยส่วนหลัก ๆ 4 ส่วน คือ คอมไพเลอร์ภาษา C ที่มีการปรับปรุงเพิ่มเติมจากคอมไพเลอร์ภาษา C ทั่วไป ๆ ให้มีความเหมาะสมต่อฮาร์ดแวร์มากยิ่งขึ้น มีฟังก์ชันสำหรับการจำลอง และแก้ไขการทำงานต่าง ๆ



รูปที่ 6 พาราเลลล่าบอร์ด

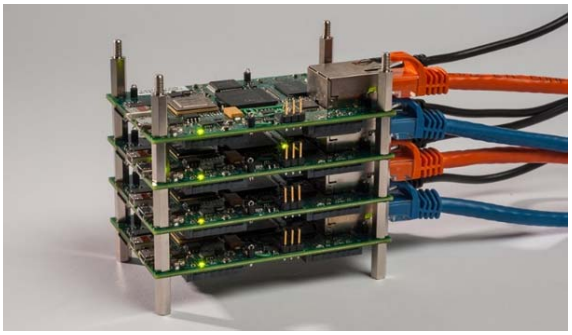


รูปที่ 7 *Parallella High Level Architecture*

3.3 การนำพาราเลลล่าบอร์ดไปประยุกต์ใช้งาน

Beowulf Cluster [14] เป็นการนำบอร์ด FPGA รุ่น Zynq ของบริษัท Xilinx มาเชื่อมต่อกับ Epiphany Multicore Coprocessor ในรูปแบบที่ยังเป็นอุปกรณ์ต่อเพิ่ม เป็นตัวต้นแบบของพาราเลลาบอร์ด ซึ่งใช้จำนวนตัวประมวลผลทั้งหมด 144 ตัว ดังรูปที่ 8 บอร์ด FPGA รุ่น Zynq ได้ติดตั้งระบบปฏิบัติการอูบันทูลินุกซ์ (Ubuntu Linux Operating System) อยู่ภายใน และใช้ตัวจัดการการประมวลผลของแต่ละตัวประมวลผลด้วย OpenMPI ซึ่งเป็นแอปพลิเคชันโปรแกรมอินเตอร์เฟส และเชื่อมต่อแต่ละบอร์ดผ่านอีเทอร์เน็ต ส่วนงานวิจัย [15] ได้นำ OpenCL เข้ามาใช้ ทำให้ตัวประมวลผลหลายชนิดที่อยู่ภายในบอร์ดสามารถจัดสรรการทำงานได้ง่าย ในแต่ละตัวประมวลผลแต่ละชนิด ทำให้สามารถพัฒนาในรูปแบบแอปพลิเคชันเพื่อจัดการการทำงานในส่วนต่าง ๆ ให้สอดคล้องกัน

รูปที่ 8 *Parallella Beowulf Cluster*



รูปที่ 9 Building a Parallella Mini-Cluster

พาราเลลล่าบอร์ดสามารถนำมาทำเป็นคลัสเตอร์ [16] ดังรูปที่ 9 โดยใช้ A Highly Scalable Resource Manager: SLUR ที่เป็นโอเพนซอร์ส นำมาเป็นตัวจัดการถูกออกแบบให้มีลักษณะการทำงานเช่นเดียวกับลินุกซ์คลัสเตอร์ สามารถรองรับจำนวนโหนดได้ถึง 65,536 โหนด และตัวประมวลผลสูงสุดถึง 100,000 ตัว รองรับการทำงานได้ถึง 120,000 งานต่อชั่วโมง สามารถประหยัดพลังงานได้เป็นอย่างดี

จากตัวอย่างการนำพาราเลลล่าบอร์ดไปประยุกต์ใช้งานที่ได้กล่าวในข้างต้น ยังมีการนำพาราเลลาบอร์ดไปประยุกต์ใช้งานอื่นๆ อีกมากมาย [17] เช่น ระบบปฏิบัติการแอนดรอยด์ (Android Operating System) ปัญญาประดิษฐ์ (Artificial Intelligence) การประมวลผลภาพและวีดีโอ LLVM Compiler หุ่นยนต์ การคำนวณทางวิทยาศาสตร์ และ Berkeley Open Infrastructure for Network Computing (BOINC) เป็นต้น

4 บทสรุป

จากงานวิจัยต่าง ๆ ที่ผ่านมาแสดงให้เห็นถึงแนวทางการพัฒนาระบบที่มีตัวประมวลผลหลายชนิด (HAS) มาใช้งานด้วยภาษาระดับสูงต่าง ๆ รวมทั้งการนำพาราเลลล่าบอร์ดที่เป็นลักษณะของ HSAไปประยุกต์ใช้งานต่าง ๆ มากมาย ทำให้ผู้วิจัยเกิดแนวคิดในการนำไปพัฒนาเพื่อที่จะให้สามารถรองรับการประมวลผลข้อมูลที่มีปริมาณมากมายต่าง ๆ สามารถจัดการการทำงานได้อย่างเหมาะสมและมีประสิทธิภาพ โดยหัวข้อที่น่าสนใจคือ คือเรื่องของการจัดการซอฟต์แวร์ เพื่อรองรับการจัดสรรการทำงานและการแบ่งงานให้เหมาะสมกับตัวประมวลผลแต่ละชนิด สามารถติดต่อสื่อสารระหว่างโฮสต์กับชิป Multicore Coprocessor รวมทั้งตัว FPGA ที่อยู่ในระบบ นอกจากนี้ยังมีเรื่องของการออกแบบสถาปัตยกรรมของระบบและพัฒนาอัลกอริทึมเพื่อทำการในการแบ่งงาน การเรียงลำดับการทำงานย่อยต่าง ๆ รวมทั้งการแบ่งหน่วยความจำภายในและภายนอกให้เหมาะสมกับตัวประมวลผลแต่ละชนิด ก็เป็นส่วนที่จะต่อยอดงานวิจัยได้ต่อไป เพื่อรองรับการประมวลผลข้อมูลภาพ ข้อมูลวีดีโอ หรือการแก้ปัญหาอื่น ๆ ที่ต้องการการคำนวนที่ซับซ้อน โดยจะต้องเป็นระบบที่ใช้งานได้ง่ายและราคาไม่แพง

เอกสารอ้างอิง

[1] Teich, J., "Hardware/Software Codesign: The Past, the Present, and Predicting the Future," Proceedings of the IEEE , vol.100, no.Special Centennial Issue, pp.1411,1430, May 13 2012.

[2] Shagrithaya, K.; Kepa, K.; Athanas, P., "Enabling development of OpenCL applications on FPGA platforms," Application-Specific Systems, Architectures and Processors (ASAP), 2013 IEEE 24th International Conference on , vol., no., pp.26,30, 5-7 June 2013.

[3] Ozgul, B.; Langer, J.; Noguera, J.; Visses, K., "Software-programmable digital pre-distortion on the Zynq SoC," Very Large Scale Integration (VLSI-SoC), 2013 IFIP/IEEE 21st International Conference on , vol., no., pp.288,289, 7-9 Oct. 2013.

[4] Wikipedia., “OpenCL,” http://th.wikipedia.org/wiki/OpenCL, May 7, 2014.

[5] Aaftab Munshi., “The OpenCL Specification,” https://www.khronos.org/registry/cl/specs/opencl-1.2.pdf, Nov 14, 2012.

[6] Taneem Ahmed., “OpenCL framework for a CPU, GPU, and FPGA Platform,” https://tspace.library.utoronto.ca/bitstream/1807/30149/3/Ahmed_Taneem_201111_MSc_thesis.pdf, 2011.

[7] Rick Weber, Akila Gothandaraman, Robert J. Hinde, and Gregory D. Peterson “Comparing Hardware Accelerators in Scientific Applications: A Case Study” IEEE Transactions On Parallel And Distributed Systems, Vol. 22, No. 1, January 2011.

[8] AMD., “HSA: A New Architecture for Heterogeneous Computing,” Tirias Research, January 28, 2013.

[9] James Coole, Greg Stitt, “Fast, Flexible High-Level Synthesis From Opencl Using Reconfiguration Contexts” IEEE Micro (Volume:34 , Issue: 1), IEEE Computer Society, Page(s): 42 – 53, Jan.-Feb. 2014.

[10] Adapteva., “Parallella-1.x Reference Manual,” http://www.parallella.org/docs/parallella_manual.pdf, Sep 9 2014.

[11] Xilinx Inc., “UG871-Vivado Design Suite Tutorial: High-Level Synthesis,” http://www.xilinx.com/support/documentation/sw_manuals/xilinx2014_3/ug871-vivado-high-level-synthesis-tutorial.pdf, May 6, 2014.

[12] Xilinx Inc., "Zynq-7000 Silicon Devices," http://www.xilinx.com/products/silicon-devices/soc/zynq-7000/silicon-devices.html, Nov 13, 2014.

[13] Adapteva., "Epiphany sdk reference (REV 5.13.09.10)," http://adapteva.com/docs/epiphany_sdk_ref.pdf, Aug. 2014.

[14] Yaniv Sapir., "Building the World's First Parallella Beowulf Cluster," http://www.adapteva.com/white-papers/building-the-worlds-first-parallella-beowulf-cluster, January 18 2013.

[15] Andreas Olofsson., "OpenCL: Leveling the playing field for processors," http://www.adapteva.com/white-papers/opencl-leveling-the-playing-field-for-processors, December 19, 2012.

[16] Censix., "Mini Parallella Cluster managed with SLURM," http://forums.parallella.org/viewtopic.php?f=32&t=1632, Aug. 13, 2014.

[17] Parallella Community ., "Project," http://forums.parallella.org/viewforum.php?f=6 Nov. 13, 2014.

Sethakarn Prongnuch finished B.Eng. in Computer Engineering in May 2011 from Rajamangala University of Technology Phra Nakhon, Bangkok. In 2013, he received M.Eng. in Electrical Engineering (Computer) from Mahanakorn Unversity of Technology, Bangkok. He is now working as a lecturer in Department of Computer Engineering, Faculty of Industrial Technology, Suan Sunandha Rajabhat University. His interests include high performance embedded system, FPGA, C/C++ Programming, HTML Language, Joomla, Microsoft Office, Microsoft Visual Studio, Moodle, Verilog-HDL/VHDL Programming.

Theerayod Wiangtong received the B.S. degree in electronics engineering from king Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand, in 1993, and the M.S. degree in satellite communication from University of Surrey, UK in 1995, and the Ph.D. degree in digital system design and codesign from Imperial College London in 2004. He is now working at Mahanakorn University of Technology. His research includes Digital IC design, FPGA, Reconfigurable computing platform, Video/Image processing applications, C/C++, Hardware/software co-design and Embedded systems.