การออกแบบส่วนควบคุมที่ใช้สัญญาณนาฬิกาสองเฟสเพื่อเพิ่มประสิทธิภาพของหน่วยประมวลผลแบบแสตก

Design of a two-phase clocked control unit for performance enhancement of a stack processor

อลงกต บุรุษอาชาไนย และ ประภาสจงสถิตย์วัฒนา
ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
จุฬาลงกรณ์มหาวิทยาลัย ถนนพญาไท ปทุมวัน กรุงเทพฯ 10500
Email: g46abl@cp.eng.chula.ac.th, Prabhas.C@chula.ac.th

บทคัดย่อ

บทความนี้นำเสนอการออกแบบส่วนควบคุมสำหรับหน่วยประมวลผลแบบแสตกขนาด 16 บิต เพื่อใช้ในระบบฝังตัว การออกแบบต้องคำนึงถึงการประหยัดทรัพยากร ทางเดินข้อมูลที่เรียบง่าย และคำสั่งที่เพียงพอต่อการใช้งาน เพื่อต้องการให้สังเคราะห์ได้จริงบนวงจรรวมจึงได้ออกแบบหน่วยประมวลผลให้ไม่ซับซ้อน มีการทดลองการทำงานจริง บน เอฟพีจีเอ และวัดผลการทำงานต่างๆออกมาได้อย่างถูกต้อง ผลการสังเคราะห์วงจรรวม มีการใช้ทรัพยากรจำนวน 5413 เกทสมมูล และทำงานได้ที่ความถี่สูงสุด 46 เมกกะเฮริตซ์

1. บทนำ

ในปัจจุบันเทคโนโลยีการผลิตวงจรรวมได้พัฒนาอย่างก้าวกระโดดคือมีการพัฒนารวดเร็วมากตามกฎของมัวร์ (Moore's Law) ทำให้สามารถพัฒนาวงจรรวมให้มีประสิทธิภาพได้มากขึ้นเป็นเท่าตัวภายในเวลาไม่เกิน 18 เดือน จึงทำให้มีพัฒนาวงจรรวมออกมาให้มีคุณสมบัติต่างๆ เช่น พัฒนาให้ประหยัดพลังงาน หรือสามารถเคลื่อนย้ายได้โดยง่าย ระบบฝังตัวเป็นระบบที่ต้องมีสมรรถนะในการคำนวณ ประหยัดพลังงาน เคลื่อนย้ายได้โดยง่าย มีทรัพยากรขนาดเล็กและไม่ซับซ้อน[1] ทุกอย่างของระบบฝังตัวจะต้องทำบนอุปกรณ์ที่มีให้ใช้อย่างประหยัด และข้อดีอีกประการหนึ่งของการออกแบบสถาปัตยกรรมให้ไม่ซับซ้อนนั้นคือ จะสังเคราะห์ลงบนวงจรรวมจริงๆได้เพราะยังมีความต่างกันอยู่พอสมควรระหว่างการสังเคราะห์ลงบนเทคโนโลยีเอฟพีจีเอกับเทคโนโลยีการสังเคราะห์ด้วยวงจรรวมเฉพาะกิจ(ASIC) [2]

บทความนี้จึงได้นำเสนอหน่วยประมวลผลที่คำนึงถึงประสิทธิภาพภายใต้ทรัพยากรและหน่วยความจำที่จำกัดเป็นหลัก มีการพัฒนาหน่วยประมวลผลแบบแสตกเพราะประโยชน์ประการหนึ่งของการเก็บข้อมูลแบบแสตก คือภาษาเครื่องนั้นมีขนาดเล็กและการเปลี่ยนจากภาษาระดับสูงมาเป็นภาษาเครื่องนั้นยังมีความใกล้เคียงกันอีกด้วย

2. การออกแบบหน่วยประมวลผล

2.1 ชุดคำสั่ง

ชุดคำสั่งอยู่ในรูปแบบ ไบต์โค้ด มีคำสั่งทั้งหมด 25 คำสั่งแบ่งกลุ่มตามจำนวนตัวถูกกระทำ(Operand) ได้ 3 แบบคือ 1) ไม่มีตัวถูกกระทำ 2) มีตัวถูกกระทำหนึ่งไบต์ 3) มีตัวถูกกระทำสองไบต์

มีการใช้เรจิสเตอร์ทั้งหมด 4 ตัวดังนี้ 1) TOS (Top Of Stack) มีไว้เพื่อเพิ่มความเร็วในการเข้าถึงข้อมูลเพราะโดยเอาข้อมูลตัวบนสุดของแสตกมาเก็บรอไว้เลยจึงไม่ต้องติดต่อผ่านหน่วยความจำ 2) SP (Stack Pointer) เป็นตัวชี้บอกจุดเริ่มต้นของแสตก 3) FP (Frame Pointer) เป็นตัวบอกจุดสิ้นสุดการทำงานของโปรแกรม และ 4) PC (Program Counter) ตัวบอกตำแหน่งของคำสั่ง

คำสั่งทางคณิตศาสตร์จะกระทำกับตัวบนสุดของแสตก 2 ตัว บางคำสั่งจะทำกับตัวบนสุดของแสตกตัวเดียว และคำสั่ง LD และ ST จะทำกับหน่วยความจำส่วนกลาง (Global Memory) ส่วน GET และ PUT จะกระทำกับหน่วยความจำท้องถิ่นภายในแอคติเวชันเรคอร์ด (Activation Record) ซึ่งถูกชี้โดยเรจิสเตอร์ FP เช่นคำสั่ง CALL จะมีการสร้างแอคติเวชันเรคอร์ดใหม่ขึ้นมาและจะมีการส่งผ่านตัวแปร โดยเข้าไปเขียนที่แอคติเวชันเรคอร์ดของตัวที่เรียกส่วนคำสั่ง RET แบบมีการคืนค่าจะส่งผ่านค่าทางเรจิสเตอร์ TOS ด้วย

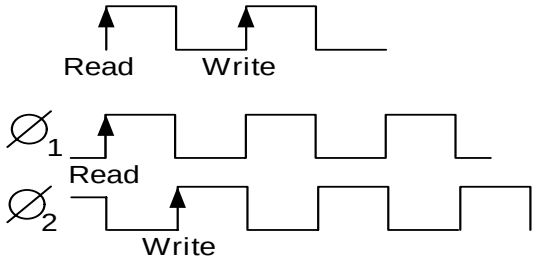
2.2 ระบบทางเดินข้อมูล

หน่วยประมวลผลมีขนาดความกว้างของข้อมูล 16 บิต และมีการใช้เรจิสเตอร์ 4 ตัวตามที่กล่าวมาในหัวข้อ 2.1

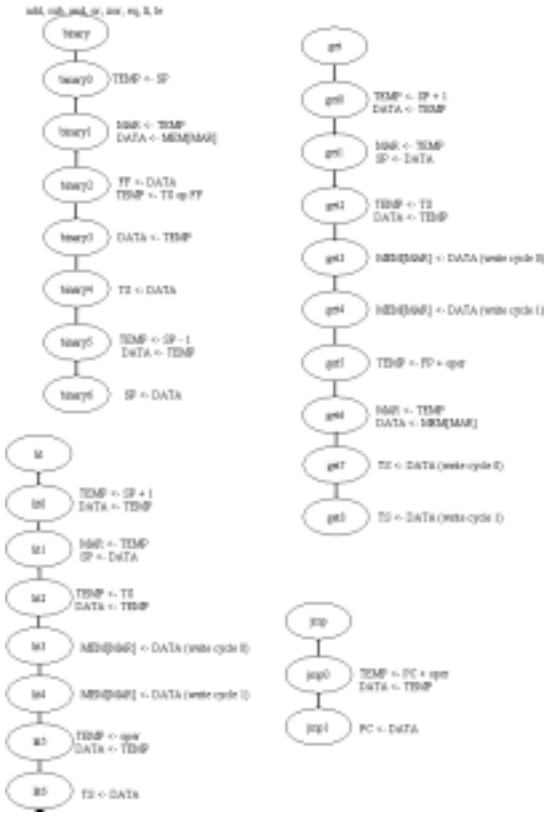
เรจิสเตอร์อื่นๆ ที่ใช้ภายในมี IR (Instruction Register) เก็บคำสั่งปัจจุบัน FF เก็บข้อมูลชั่วคราว MAR (Memory Address Register) เก็บที่อยู่เพื่อติดต่อกับหน่วยความจำ หน่วยเลขและตรรกะ (Arithmetic and Logic Unit, ALU) มีฟังก์ชันพื้นฐานเช่น บวก ลบ เลื่อนบิต เปรียบเทียบ เป็นต้น ข้อมูลจากหน่วยเลขและตรรกะจะส่งกลับไปยังบัสข้อมูล (DBUS) โดยผ่านตัวกัน (BUF) ตัวเลือก (MUXs) จะดึงข้อมูลทีละไบต์ป้อนสู่ IR (ดูรูปที่ 1)

2.3 การดึงข้อมูลโดยใช้สัญญาณนาฬิกาสองเฟส

ในการออกแบบหน่วยประมวลผลแต่เดิมใช้สัญญาณนาฬิกาเพียงขอบเดียว คือข้อมูลทุกอย่างจะเปลี่ยนแปลงที่ขอบขาขึ้นของสัญญาณนาฬิกา ดังนั้นในการส่งข้อมูลจากเรจิสเตอร์จากตัวหนึ่งไปยังอีกตัวหนึ่งต้องใช้สัญญาณนาฬิกาสองรอบ รอบแรกปล่อยข้อมูลจากเรจิสเตอร์หนึ่งลงบัส รอบสอง อ่านข้อมูลจากบัสเขียนเข้าสู่เรจิสเตอร์สอง ข้อดีของวิธีนี้คือวงจรง่ายไม่ซับซ้อน แต่มีข้อเสียคือช้า รูปที่ 2 แสดงลำดับการควบคุมของคำสั่ง บางคำสั่งเช่นคำสั่งบวกใช้ 7 รอบ (Binary add) การเพิ่มความเร็วทำได้โดยลดจำนวนรอบของการควบคุม โดยให้การส่งข้อมูลเกิดในรอบเดียวเป็นสองเฟส เฟสแรกอ่านข้อมูลจากเรจิสเตอร์เฟสสองเขียนข้อมูลลงเรจิสเตอร์ การใช้สัญญาณนาฬิกาสองเฟสทำได้โดยให้มีสัญญาณนาฬิกาสองสัญญาณที่ต่างเฟสกัน 180 องศาตามรูป 3



รูปที่ 3 แสดงการเปรียบเทียบระหว่างสัญญาณนาฬิกาแบบเดิมกับสองเฟส



รูปที่ 2 แสดงการการทำงานของคำสั่งบางคำสั่งภายในหน่วยประมวลผล

การทำเช่นนี้จำนวนรอบของการควบคุมจะลดลง ถ้าความถี่สัญญาณนาฬิกาเท่าเดิมการทำงานของหน่วยประมวลผลจะเร็วขึ้นเช่น ตัวอย่างที่ 1

ตัวอย่างที่ 1 แสดงลำดับของการทำงานในการอ่านคำสั่งด้วยสัญญาณเฟสเดียวกับสองเฟส

```
//Single Phase Clock
…
1  PC -> alu (PASS1)
2  Tbus -> mar(shift), MemRead
3  Dbus -> mux8 -> IR, Pc - > alu (INC1)
4  Tbus -> mar(shift),tbus -> dbus
    ,dbus -> PC
```

```
//Double Phase Clock
```

Phase1	Phase2
… 1 PC -> alu (PASS1) 2 MemRead 3 Pc -> alu (INC1)	… Tbus -> mar Dbus -> mux8 -> IR Tbus ->dbus, dbus -> PC

เปรียบเทียบลำดับการควบคุมของการอ่านคำสั่ง (Instruction Fetch) แบบเดิมใช้จำนวน 4 รอบแล้วบวกหนึ่งเข้ากับตัวชี้โปรแกรม ดังต่อไปนี้

1. PC ผ่าน ALU ไป TBUS
2. เก็บค่าจาก TBUS ไป MAR แล้วอ่านคำสั่งจากหน่วยความจำ
3. หลังจากอ่านคำสั่งจากหน่วยความจำแล้วเอาค่าไปเก็บไว้ที่ IR และเพิ่มค่า PC ขึ้นหนึ่ง ผ่านทาง TBUS
4. TBUS แล้วทำผ่าน buf ไป DBUS กลับมาเขียน PC

ส่วนลำดับควบคุมแบบสองเฟสใช้ 3 รอบดังนี้

1. PC ผ่าน ALU ไป TBUS แล้ว TBUS ไป MAR
2. อ่านคำสั่งจากหน่วยความจำแล้วเอาไปไว้ที่ IR
3. PC ผ่าน ALU ทำ +1 แล้วทำผ่าน buf กลับมาเขียน PC

จากการทดลองพบว่าความเร็วมิได้เพิ่มเป็นสองเท่า เพราะทางเดินข้อมูลแบบเดิมยังมีบางส่วนที่ทำขนานกันไปได้ เช่นขั้นที่ 3 ของแบบเดิมทำสองอย่างพร้อมกัน คือ ข้อมูลจาก DBUS ไป IR พร้อมกับ PC+1 ไป TBUS อย่างไรก็ตามลำดับการควบคุมในแบบสองเฟสก็ลดลง โดยวัดจากจำนวนเสตทแบบเดิมใช้ 75 เสตท แบบสองเฟสมี 57 เสตท

2.4 ความสัมพันธ์ระหว่างชุดคำสั่งกับภาษาระดับสูง

การออกแบบที่ผ่านมาทั้งหมดนี้มีการคำนึงความสัมพันธ์ระหว่างภาษาระดับสูงกับภาษาเครื่อง เพื่อลดช่องว่างระหว่างภาษา จะได้โค้ดที่มีขนาดเล็ก ตัวอย่างที่ 2 แสดงโปรแกรมในภาษาระดับสูงเพื่อบวกเลขจากหนึ่ง ถึง สิบ ตัวอย่างที่ 3 แสดงโปรแกรมภาษาเครื่องของหน่วยประมวลผลแบบแสตกที่แปลมาจากโปรแกรมในตัวอย่างที่ 2 โดยตรง จะเห็นได้ว่าภาษาทั้งสองมีความ

คล้ายกันอย่างมากซึ่งจะทำให้ช่วยลดช่องว่างในการแปลภาษาจึงทำให้เขียนตัวแปลภาษาได้ง่ายขึ้นโปรแกรมที่เป็นภาษาเครื่องก็อ่านเข้าใจได้ง่ายเช่นกัน

ตัวอย่างที่ 2 แสดงโปรแกรมของภาษาระดับสูง

```
//sum from a to b, s is a local
to sum a b | s =
  s = 0
  while a <= b
    s = s + a
    a = a + 1
  s
to main =
  print sum 1 10
```

ตัวอย่างที่ 3 โปรแกรมภาษาเครื่อง

```
:sum
LIT 0, PUT s,
:loop
GET a, GET b, LE, JF exit,
GET s, GET a, ADD, PUT s,
GET a, LIT 1, ADD, PUT a, JMP loop
:exit
GET s, RETV

:main
LIT 1, LIT 10, CALL sum, SYS print
end
```

3. ผลการทดลอง

ในการวัดผลการทดลองใช้โปรแกรมเพื่อทำการทดลองทั้งหมด 7 โปรแกรม ให้ทดสอบทุกคำสั่ง เพื่อทำการเปรียบเทียบความเร็วการทำงานของหน่วยประมวลผลที่ใช้สัญญาณนาฬิกาเดียวกับหน่วยประมวลผลที่ใช้สัญญาณนาฬิกาสองเฟส การทดลองใช้โปรแกรมจำลองการทำงานที่สามารถวัดจำนวนสัญญาณนาฬิกาได้อย่างถูกต้องตามความจริง ได้ผลตามตารางที่ 2

โปรแกรมทั้งเจ็ดทดสอบการวน การเรียกแบบเวียนเกิด การคำนวณเมตริกซ์ การเรียงลำดับ เป็นต้น ข้อมูลที่เก็บได้คือจำนวนคำสั่งที่ใช้ (I1) จำนวนรอบการควบคุมที่ใช้เพื่อทำงานให้เสร็จของหน่วยควบคุมแบบเดิม (t1) และของหน่วยควบคุมที่ใช้สัญญาณนาฬิกาสองเฟส (t2) จาก t1 และ t2 คำนวณความเร็วที่เพิ่มขึ้น (Speedup) (1- (t2 / t1)) ได้ค่าเฉลี่ย 31 เปอร์เซ็นต์ จากทุกโปรแกรมที่ทดลองตามตารางที่ 2 ถ้าคำนวณจำนวนรอบต่อคำสั่ง (Cycle Per Instruction , CPI) ของทั้งสองแบบจะได้ดังนี้ (CPI1 = t1/I1 CPI2 = t2/I2) CPI1 = 12.58 CPI2 = 8.61 ดังตารางที่ 3

ตารางที่ 2 แสดงการเปรียบเทียบประสิทธิภาพในการประมวลผล

โปรแกรม	I1	t1	t2	Speedup
bubble	11925	147731	100638	0.318
hanoi	2317	29696	20419	0.312
matmul	13886	182235	126338	0.306
perm	5469	69301	47547	0.313
queen	765141	8909675	5963164	0.330
quick	3972	49687	34099	0.313
sieve	17151	221670	151870	0.314
เฉลี่ย				0.315

ตารางที่ 3 แสดงการเปรียบเทียบจำนวนครั้งที่ใช้ในการประมวลผลของโปรแกรมทดลอง

โปรแกรม	CPI1	CPI2
bubble	12.38	8.44
hanoi	12.82	8.81
matmul	13.12	9.1
perm	12.67	8.69
queen	11.65	7.79
quick	12.51	8.58
sieve	12.93	8.86
เฉลี่ย	12.58	8.61

จำนวนทรัพยากรที่ใช้ก็แตกต่างกันดังตารางที่ 4 เปรียบเทียบทรัพยากรในการสังเคราะห์หน่วยประมวลผลทั้งสองแบบและนำหน่วยประมวลผลอีก 2 ตัวที่หามาได้จาก opencores.org [3] มาเปรียบเทียบด้วย คือ C16 เป็นหน่วยประมวลผลแบบ 16 บิตที่ใช้เรจิสเตอร์เป็นหลัก และ T80 ที่เป็นหน่วยประมวลผลขนาด 8 บิตจะเห็นว่าการใช้ทรัพยากรของหน่วยประมวลผลที่นำเสนอนับว่าต่ำเมื่อเทียบกับหน่วยประมวลผลตัวอื่นๆ และสมรรถนะก็ใช้ได้โดยดูจากความถี่สัญญาณนาฬิกา

ตารางที่4 ผลการสังเคราะห์หน่วยประมวลผลต่างๆ

Processor	Frequency	Equivalence gate
Stack Single Phase	45 MHz	6,497
C16	40 MHz	N/A
T80	33 MHz	15,009
Stack Double Phase	46 MHz	5413

4. งานวิจัยที่เกี่ยวข้อง

เครื่องเสมือนจาวา (Java Virtual Machine) เป็นงานวิจัยที่มีชื่อเสียงมากที่ใช้ชุดคำสั่งแบบแสตก [4] ซึ่งจะถูกฝังตัวเข้าไปในเว็บเบราว์เซอร์ และงานวิจัยอีกชิ้นหนึ่งคือ พิโคจาวาชิพ ซึ่งปัจจุบันได้ผลิตชิพออกมาในเชิงพาณิชย์ [5] งานวิจัย [6] ได้มีการออกแบบฮาร์ดแวร์ให้มีความสามารถทางด้านการทำงานของจาวาไบต์โค้ดเพื่อให้ทำงานบนหน่วยคอมพิวเตอร์แบบระบบเรจิสเตอร์ได้อย่างมีประสิทธิภาพ

งานที่กล่าวมาเหล่านี้เน้นที่สมรรถนะเป็นสำคัญต่างจากตัวประมวลผลในบทความนี้ที่คำนึงถึงการประหยัดทรัพยากรเป็นสำคัญ

5. สรุป

ผลการทดลองพบว่า หน่วยประมวลผลที่ใช้สัญญาณนาฬิกาสองเฟสนี้สามารถทำงานได้เร็วกว่าแบบสัญญาณนาฬิกาเดียว เพราะในแต่ละรอบการทำงานของหน่วยประมวลผลที่ใช้สัญญาณนาฬิกาสองเฟสนั้นสามารถทำได้ 2 อย่างในหนึ่งรอบการทำงาน คือทำได้ทั้งสัญญาณขาขึ้นและขาลง ถ้าเทียบกับหน่วยประมวลผลแบบสัญญาณนาฬิกาเดียว และจากผลการทดลองพบได้ว่าหน่วยประมวลผลสามารถเพิ่มประสิทธิภาพได้โดยไม่ขึ้นกับชนิดของคำสั่งเลย เพราะจากการทดลองจากโปรแกรม จะเห็นได้ว่าสามารถลดเวลาในการประมวลผลได้ประมาณ 31 เปอร์เซ็นต์ใกล้เคียงกันทุกโปรแกรมการทดลอง และเปรียบเทียบกับหน่วยประมวลผลที่ได้ของโอเพนคอร์[3] แล้วพบว่าหน่วยประมวลผลแบบแสตกนี้มาขนาดวงจรที่เล็กเป็นที่น่าพอใจ

นอกจากนี้จากการสังเคราะห์วงจรพบว่าหน่วยประมวลผลที่ใช้สัญญาณนาฬิกาสองเฟสยังได้ขนาดที่เล็กลงกว่าหน่วยประมวลผลที่ใช้สัญญาณนาฬิกาเดียว วงจรมีขนาดเล็ก คือใช้เพียงแค่ 5413 เกทสมมูลเท่านั้น และยังสามารถทำงานที่ความถี่สูงสุดจากการสังเคราะห์ได้ถึง 46 เมกกะเฮิร์ตอีก ด้วย

6.รายการอ้างอิง

[1] C. Kozyrakis and D. Patterson,"A new direction for computer architecture research", IEEE computer, Nov. 1998, pp.24-32.

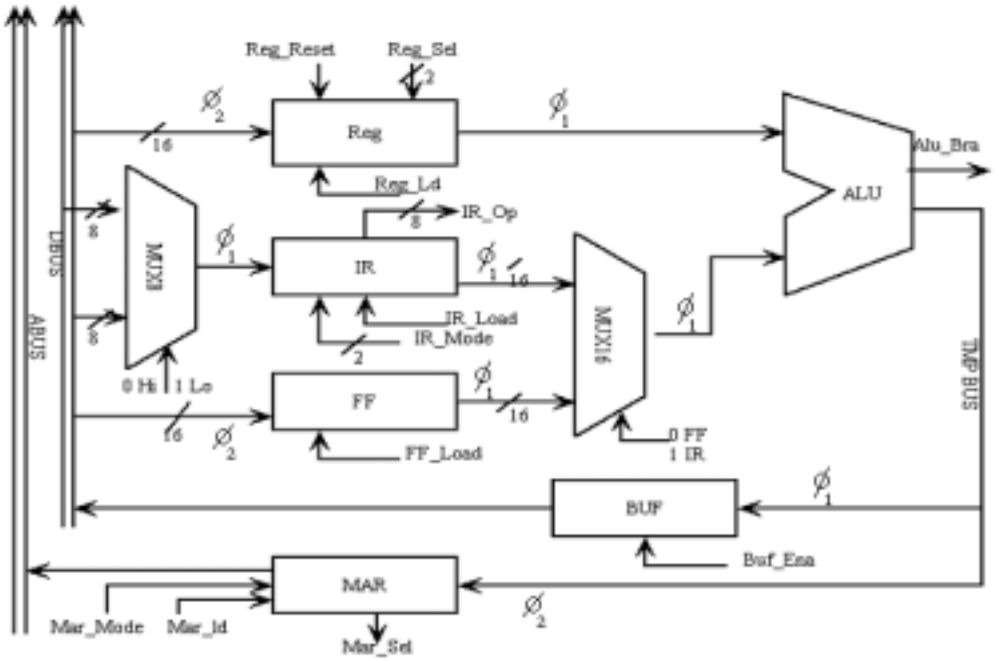
[2] P. Zuchowski, C. Reynolds, R. Grupp, S. Davis, B. Cremen, B. Troxel,"A Hybid ASIC and FPGA Architecture", IEEE computer, Nov 2002, pp 187-194

[3] www.opencores.org

[4] B. Joy (Ed), G. Steele, J. Gosling, G. Bracha, Java(TM) Language Specification (2nd Ed), Addison Wesley Pub., 2000.

[5] H. McGhan and M. O'Conner, "PicoJava : a direct execution engine for Java bytecode", IEEE Computer, Vol.31 No. 10, 1998.

[6] R. Radhakrishnan, R. Bhargava and L. John, "Improving Java Performance Using Hardware Translation", In Proceedings of 15th ACM international Conference on Supercomputing, pages 427-439, 2001.



รูปที่ 1 แสดงส่วนการเชื่อมต่อของทางเดินข้อมูล