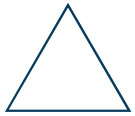
用微基准测试套件来评估SW26010众核处理器

报告：王圣翔　策划：牛世权 陈泽华

指导老师：龚春叶、甘新标、杨博



目录
CONTENTS

1

PART ONE

绪论

引言 / 背景

1.1 引言

神威·太湖之光超级计算机搭载了国产处理器申威26010。然而，与传统商用处理器相比，申威26010公开的微结构信息非常有限，这就导致了研究者很难对太湖之光进行综合性能优化。

1.2 背景

申威SW26010，是一个异构的处理器，内部包含了4个核心组。每个核心组分为两个部分：一个管理处理单元MPE和一个计算处理单元簇CPE Cluster，这个簇包含了64个CPE核心。

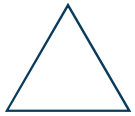
申威SW26010，是一个64位、支持SIMD单指令多数据流的乱序架构RISC处理器。其主处理器和从处理器的主频都是1.45GHz，支持256位的向量。

2

PART TWO

CPE流水线

指令时延 / 指令发射顺序



2.1 指令时延

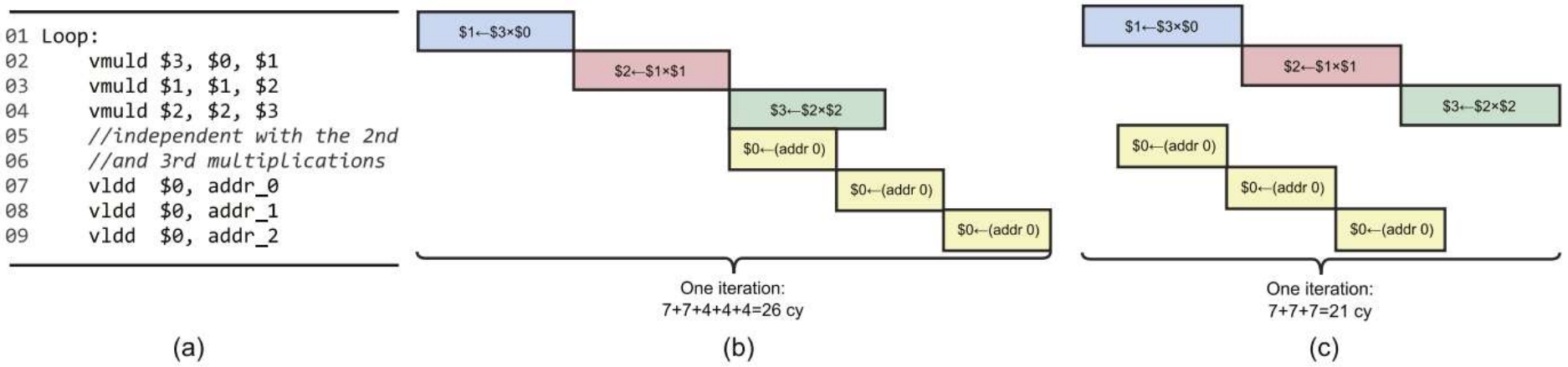
Category	Instructions	Absolute latency (*cy*)	Pipeline latency (*cy*)
Arithmetic	muld, vmuls, vaddd, vadds, vsubd, vsubs, vmad, vmas	7	1
	vsqrtd/vsqrts	32/18	28/14
	vdivd/vdivs	34/19	30/15
SPM Access	vldd, vlds	4	1
Permutation	vinsw, vinsf, vextw, vextf, vshff, vshfw	1	1
Synchronization	sync, synr	14	14

测量了4种指令的绝对时延和流水线时延。

测试方法：执行一系列写后读（RAW）相关的相同指令来测量绝对时延。执行一系列没有写后读（RAW）相关的相同指令来测量流水线时延。

结果：结果如表所示，普通算数运算指令的绝对时延为7个周期，流水线时延为1个周期。

2.2 指令发射顺序



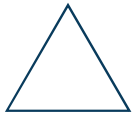
测试方法：作者开发了微基准测试程序来检测流水线P0和P1之间的指令发射顺序。3个相关的vldd指令（4个周期）跟在3个相关的vmuld指令（7个周期）之后，且只与第一个vmuld指令有读后写相关。

如果是按序发射，第一个vldd指令可以和第三个vmuld指令在同周期发射。6条指令的总时延是$7 \times 2 + 4 \times 3 = 26$ 周期。如果是乱序发射，第一个vldd能在第一个vmuld执行1个周期后发射。6条指令的总时延是$7 \times 3 = 21$ 周期。

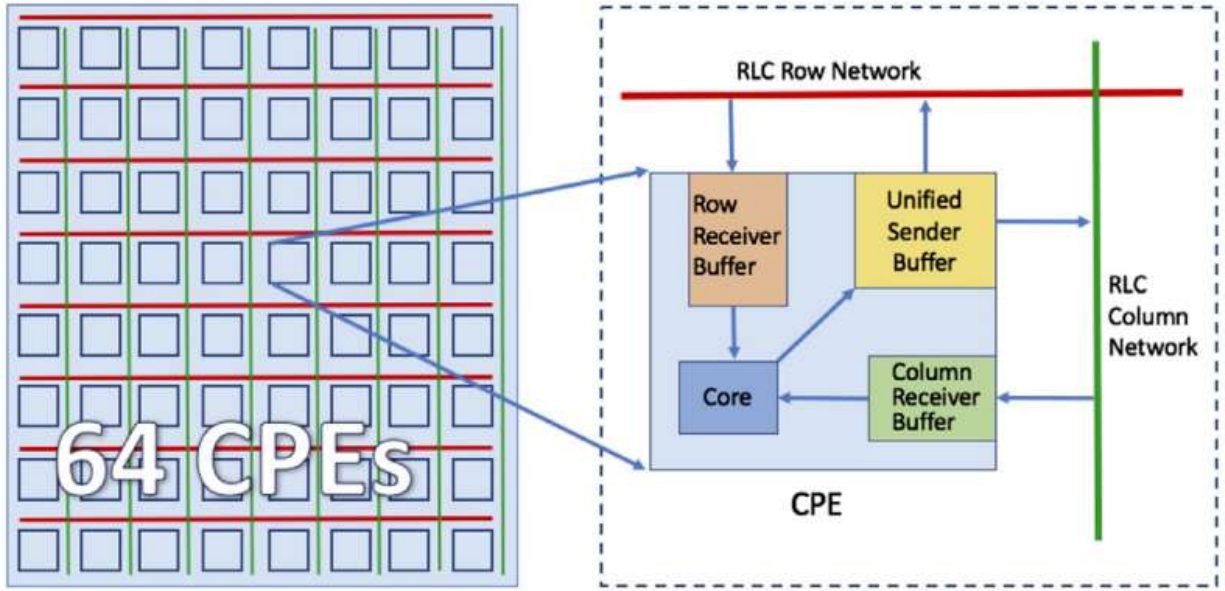
结果：测得的总时延是21个周期。流水线P0和P1之间的指令发射顺序是乱序的。

寄存器级通信

P2P RLC 时延/ Broadcast RLC 时延 / RLC 带宽 / 路由模式



3 寄存器级通信



RLC需要3步，才能在CPE间进行数据交换。
第一步，一个源CPE将寄存器数据发送到统一的发送缓冲中。
第二步，数据通过一个二维片上网络 从源CPE 转移到目的CPE。
第三步，目的CPE将寄存器数据转移到矢量寄存器。

3.1 P2P RLC 时延

RLC仅支持相同行或列之间的通信，要么是点对点模式，要么是广播模式。

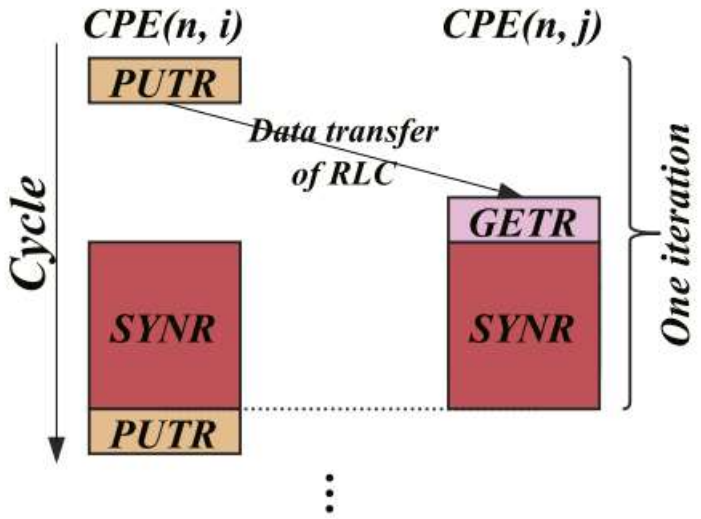
1. 点对点模式中，一个CPE可以向所在相同行或列的其他7个CPE中的一个发送寄存器数据。

2. 广播模式中，一个CPE能向相同行或列的其他7个CPE进行广播。

公开数据： P2P模式的RLC时延为10个周期。

测试方法：测量了3种类型的P2P模式时延：单向P2P RLC时延、双向P2P RLC时延、SEND/RECEIVE指令时延。

3.1 P2P RLC 时延



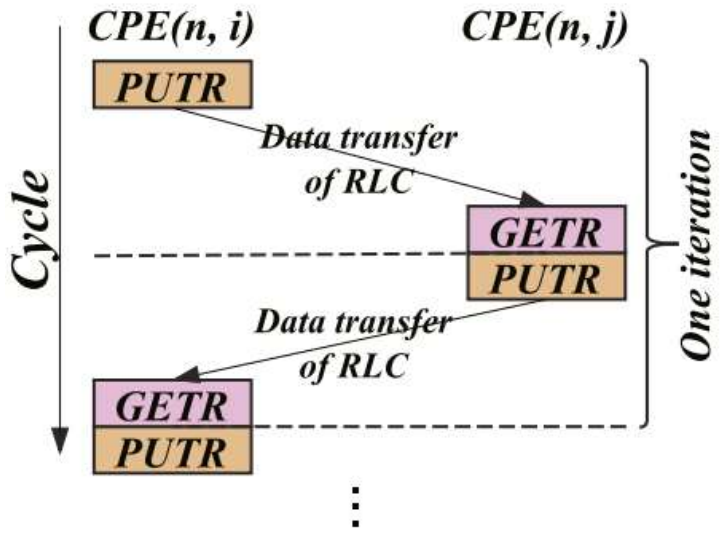
(a)

```
01      //sender CPE(n, i):
02      LOOP:
03      //send data to CPE j
04      putr $data, j
05      //sync in row with CPE j
06      synr $mask
07
08      //receiver CPE(n, j):
09      LOOP:
10      //receive data from CPE i
11      getr $data
12      //sync in row with CPE i
13      sync $mask
```

(b)

首先，测量单向P2P RLC时延。这里使用了同步指令sync/synr，来阻止非同步的SEND指令在连续的RLC过程中被覆盖。因此，计算单向P2P RLC时延时，要减去同步指令时延（14周期）。

3.1 P2P RLC 时延



(a)

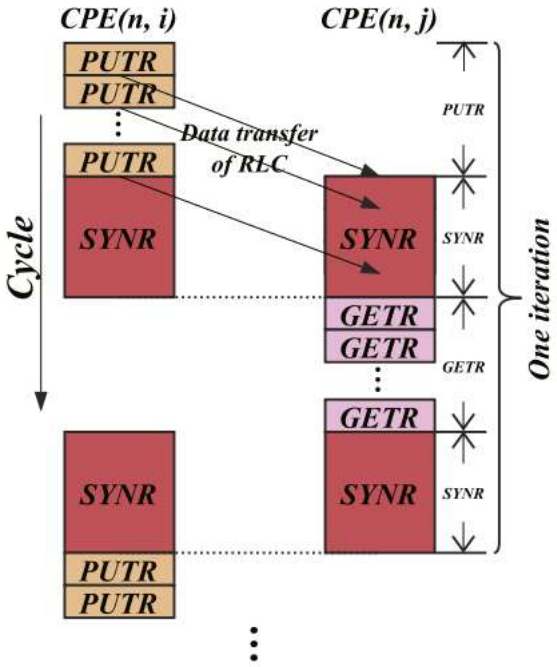
```
01      //CPE(n, i):
02      LOOP:
03      //send data to CPE j
04      putr $data1, j
05      //receive data from CPE j
06      getr $data2
07
08      //CPE(n ,j):
09      LOOP:
10      //receive data from CPE i
11      getr $data1
12      //send data to CPE i
13      putr $data2, i
```

(b)

其次，测量双向P2P RLC时延。

这里，在一对发送和接收CPE之间进行了乒乓测试。没有要求强制同步，因为发送方CPE在收到返回的数据前都是被阻塞的。

3.1 P2P RLC 时延



(a)

```
01    //sender CPE(n, i):
02    LOOP:
03    //send data to CPE j
04    putr $data, j
05    ...
06    putr $data, j
07    //sync in row with CPE j
08    synr $mask
09    //sync again
10    synr $mask
11
12    //receiver CPE(n, j):
13    LOOP:
14    //sync in row with CPE i
15    synr $mask
16    //receive data from CPE i
17    getr $data
18    ...
19    getr $data
20    //sync again
21    synr $mask
```

(b)

最后，为了测量 RECEIVE指令的时延，作者通过将结果写入相同的寄存器来创建写后写相关。此外，作者还用了两对同步指令里隔离SEND和RECEIVE指令的时延。第一对，被用在连续的SEND指令之后和连续的RECEIVE指令之前。第二对，被用在每次循环的结尾。

3.1 P2P RLC 时延

The results of P2P RLC latencies.

(a) RLC latencies in rows (*cy*)

Sender No.	Receiver No.							
	0	1	2	3	4	5	6	7
0	/	10	10	10	11	11	11	11
1	10	/	10	10	11	11	11	11
2	10	10	/	10	11	11	11	11
3	10	10	10	/	11	11	11	11
4	10	10	10	10	/	11	11	11
5	10	10	10	10	11	/	11	11
6	10	10	10	10	11	11	/	11
7	10	10	10	10	11	11	11	/

(b) RLC latencies in columns (*cy*)

Sender No.	Receiver No.							
	0	1	2	3	4	5	6	7
0	/	10	11	11	10	10	11	11
1	10	/	11	11	10	10	11	11
2	10	10	/	11	10	10	11	11
3	10	10	11	/	10	10	11	11
4	10	10	11	11	/	10	11	11
5	10	10	11	11	10	/	11	11
6	10	10	11	11	10	10	/	11
7	10	10	11	11	10	10	11	/

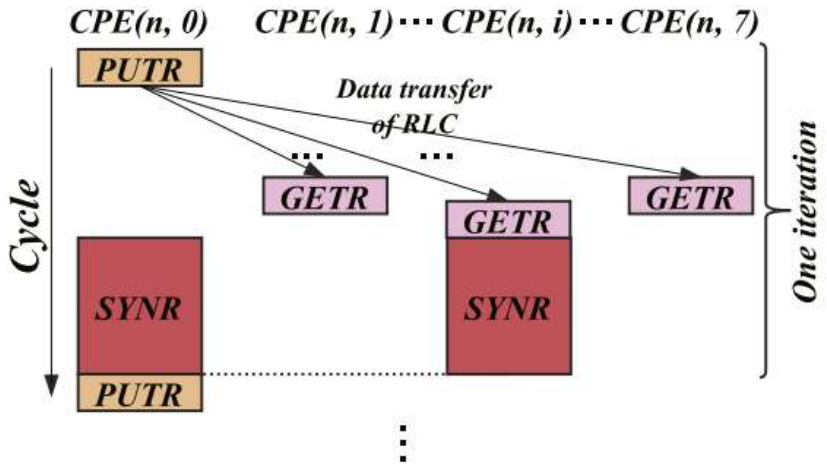
结果：

首先，P2P模式的RLC时延如上表所示，P2P模式的RLC时延在10到11周期间变化。

其次，乒乓测试的结果显示，双向RLC的时延是单向的两倍。

最后，每增加一对SEND和RECEIVE指令对，延时增加2个周期，表明 RECEIVE指令的时延是1个周期，SEND和RECEIVE指令是完全流水化的。

3.2 Broadcast RLC 时延



(a)

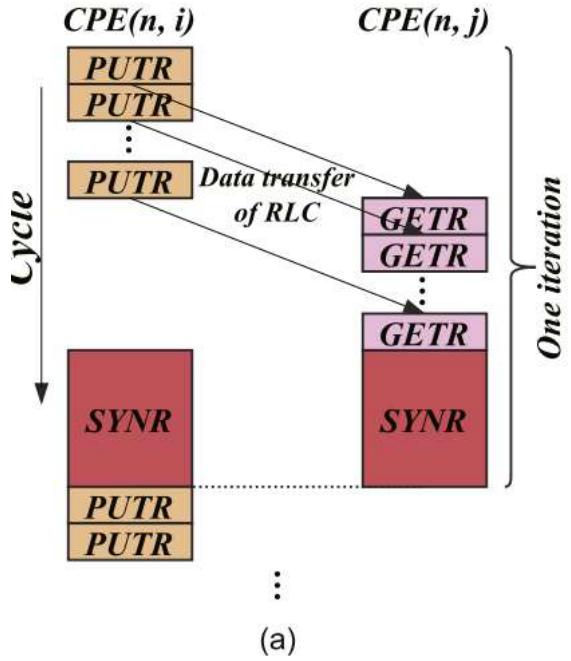
```
01      //CPE(n, i):
02      LOOP:
03      //send data to CPE j
04      putr $data1, j
05      //receive data from CPE j
06      getr $data2
07
08      //CPE(n, j):
09      LOOP:
10      //receive data from CPE i
11      getr $data1
12      //send data to CPE i
13      putr $data2, i
```

(b)

测试方法：作者开发的测试程序如图。由于无法保证7个接收CPE能在相同周期收到广播消息，作者分别测量了7个CPE的时延。

结果：广播模式的RLC与P2P模式的RLC有相同的时延。作者推测默认实现的是广播模式的RLC。

3.3 RLC 带宽



(a)

```
01      //sender CPE(n, i):
02      LOOP:
03      //successively send n data
04      putr $data_1, j
05      ...
06      putr $data_n, j
07      //sync in row with CPE j
08      synr $mask
09
10      //receiver CPE(n, j):
11      LOOP:
12      //receive n data
13      getr $data_1
14      ...
15      getr $data_n
16      //sync in row with CPE i
17      synr $mask
```

(b)

测试方法：测试RLC带宽需要2步。第一步，开发测试程序连续发射SEND和 RECEIVE指令对，然后测量从第一条SEND指令到最后一条RECEIVE指令的时延。第二步，使用这个时延来估算带宽。

3.3 RLC 带宽

The latencies and gaps of a single iteration of consecutive RLCs.

	No. of consecutive RLCs									
	1	2	3	4	5	6	7	8	9	10
Latency (*cy*)	10	12	14	17	19	21	24	26	28	31
Gap (*cy*)	1	1	2	1	1	2	1	1	2	1

基于时延和间隙（gap,连续RLC间的最小时间间隔），作者估算了总的RLC带宽。对于每个P2P RLC，一对发送和接收CPE平均1 + g1 + g2/3 = 2.33 个周期发送一个32位的矢量。故P2P RLC带宽为：32/2.33 位/周期 × 1.45 GHz × 32 CPE对 × 4组 = 2549 GB/s。

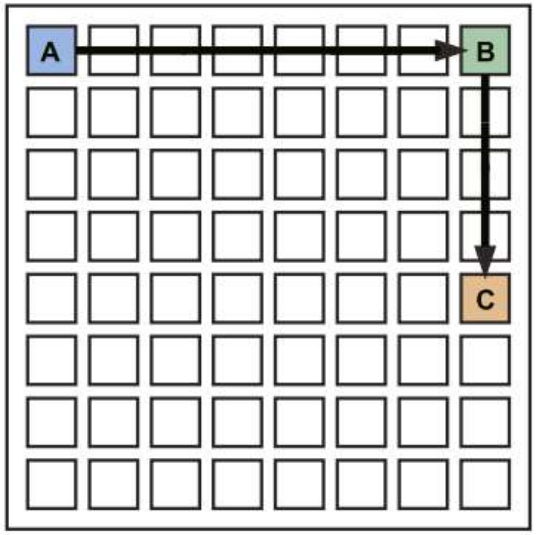
3.4 路由模式

根据不同的路由行为，路由模式被分为两种：静态路由和动态路由。

在静态路由中，源CPE在发送一个RLC消息前就已知目的CPE，目的CPE的ID被包含在RLC消息中。路由CPE可以用这个ID来转发RLC消息。

在动态路由中，目的CPE预先未知，RLC消息中包含的是条件数据。路由CPE需要先检查条件数据，然后再根据检查的结果来转发RLC消息。

3.4.1 静态路由



(a)

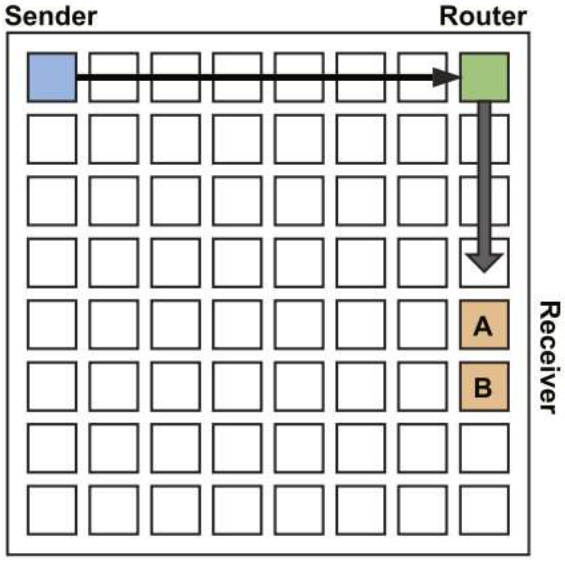
```
01      //Core A:
02      //send data to Core B in row
03      putr $data, 7
04
05      //Core B:
06      //get data from Core A in row
07      getr $data
08      //send data to Core C in column
09      putc $data, 4
10
11      //Core C:
12      //get data from Core B in column
13      getc $data
```

(b)

测试方法：如图所示是一个2跳的静态路由。目的CPE的ID包含在RLC消息中。由于源CPE和目的CPE在不同的行和列，作者开发了测试程序通过路由CPE来转发RLC消息。

结果：2条的静态路由时延是单向P2P RLC时延的两倍。n跳的静态路由时延是P2P RLC时延的N倍。

3.4.2 动态路由



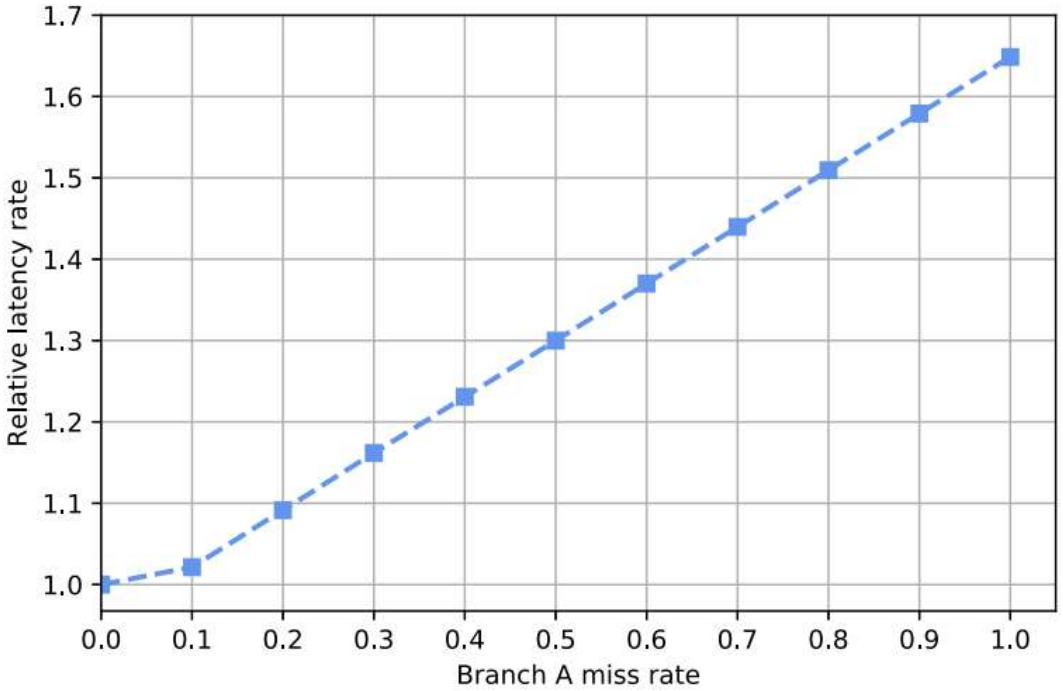
(a)

```
01 //sender:
02     //keep sending data to router
03     putr data, router
04
05 //router:
06     //get data from sender
07     getr data;
08     //choose the receiver based on data
09     if (data satisfy cond A){
10         //send to receiver A
11         putc data, A
12     } else if (cond B){
13         //send to receiver B
14         putc data, B
15     }
16
17 //receiver:
18     //keep receiving data
19     getc data
```

(b)

测试方法：上图说明了一个有2个目的CPE的动态路由。没有目的CPE的ID，只有条件数据被包含在RLC消息中。源CPE，发送RLC消息给相同行的路由CPE。之后，路由CPE检查条件数据，并根据检查结果来决定是将消息发送黑CPE(A)还是CPE(B)。作者通过测试程序实现动态路由模式，并且通过调整条件数据的命中率来测量不同条件下的时。

3.4.2 动态路由



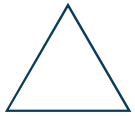
结果：图中显示的是以100%命中默认分支A为基准的相对时延率。可以看出，随着分支A缺失率的增加，相对时延率也增加。这表明，默认分支比另一个分支快。由此，作者推测CPE采用了一种固定硬件的静态分支预测器。

4

PART FOUR

内存

片上暂存内存 / 片外内存



4. 1 片上暂存内存

公开数据：每一个CPE采用片上暂存内存（on-chip SPM）作为它的本地数据内存（LDM)，其时延和峰值带宽分别为4个周期， 47.90 GB/s，持续带宽未知。

测试方法：对于时延，作者采用了BenchIT 基准测试中的指针追踪方法(pointer-chasing)。对于峰值带宽，作者执行了一系列的矢量 load/store指令。对于持续带宽，作者采用了STREAM基准测试。

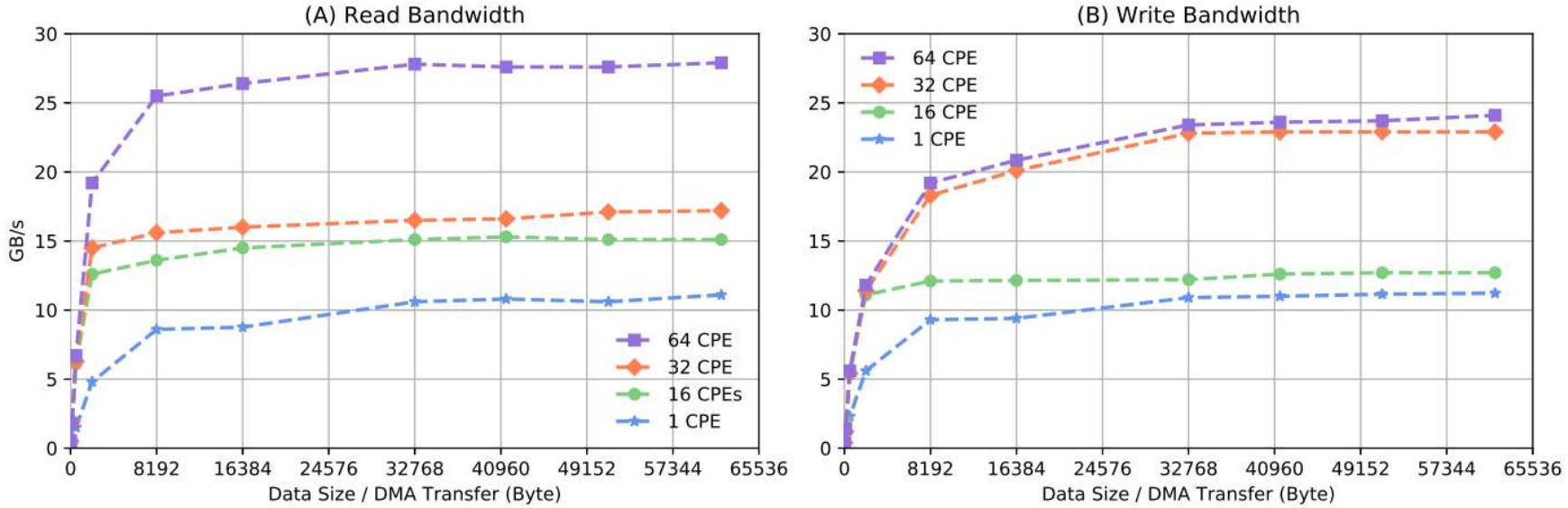
结果：测得的时延是4个周期，峰值带宽是46.4 GB/s。对Copy, Scale, Add 和 Triad 指令的STREAM 测试结果 是43.74 GB/s, 29.36 GB/s, 31.71 GB/s, 和 30.9 GB/s 。

4. 2 片外内存

每个核心组，采用一个8GB的DDR3 片外内存作为主内存。获取主存有两种方法，一种是DMA，另一种是全局load/store指令。前者能获取一大块数据，而后者只能获取一个矢量。

DMA有三种模式： PE模式，BCAST模式，RANK模式。第一种模式中，每一个CPE管理它自己的数据交换。第二种模式中，一个CPE发送一个广播请求给MPE，之后请求的数据将从主存中广播给同一核心组中的64个CPE。第三种模式，数据能被聚居或分散到相同行的CPE中。

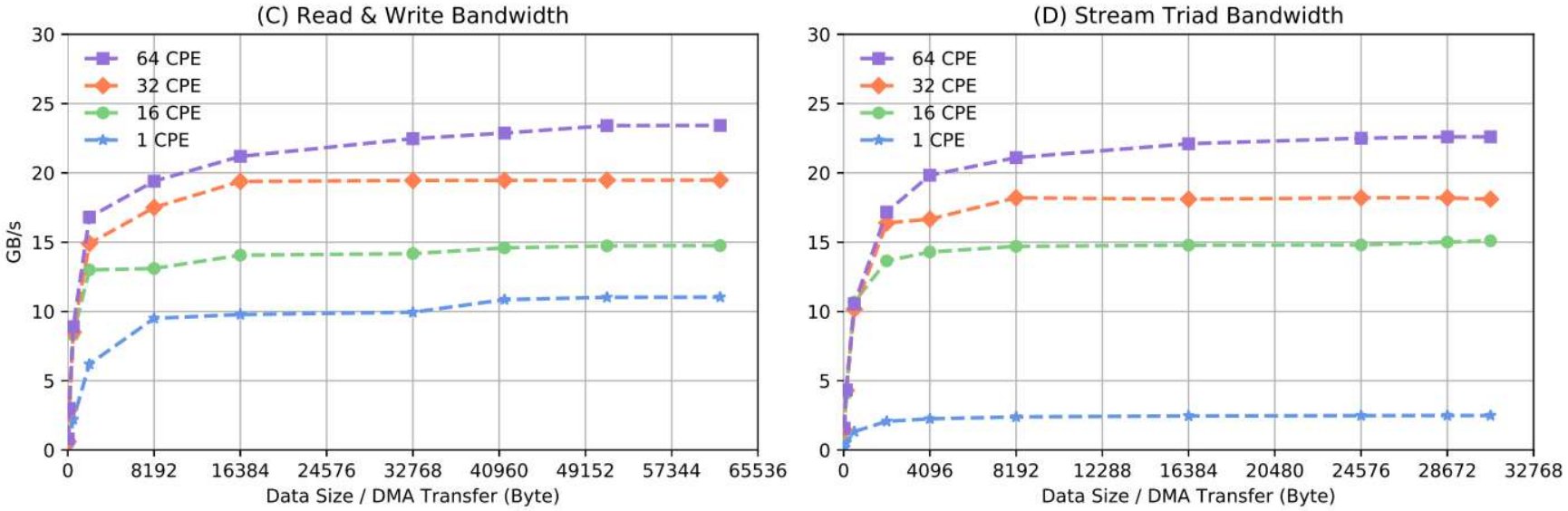
4.2.1 DMA带宽



公开数据：对于16KB大小的数据块，一个CPE和64个CPE，在PE模式下的读内存带宽分别是10.6 GB/s and 30.9 GB/s。

测试方法：作者实现了STREAM基准测试，来测量PE模式和BCAST模式下的带宽，并且测量不同数量CPE和不同数据块大小下的带宽。

4.2.1 DMA带宽



结果： 首要，图中显示了PE模式下的STREAM测试结果，(a)中 Copy的最大带宽是 27.9 GB/s， (b)中Scale的最大带宽是24.1GB/s，(c)中 Add的最大带宽是 23.4 GB/s， (d)中Triad的最大带宽是22.6 GB/s。

其次，BCAST 模式的带宽比PE模式低。比如，测得的Copy的带宽是6.97 GB/s，大概只有PE模式的23%。

4.2.2 全局 load/store指令

公开数据：全局 load/store指令（gld/gst)的时延分别是177个周期和278个周期。

测试方法：作者开发了指针追踪（pointer-chasing benchmark）基准测试程序，来测量gld的时延。

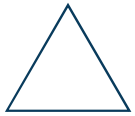
结果：gld指令的时延，与同时访问主存的CPE数量（用n表示）有关。n < 2时时延为194-200周期， 3 < n <= 64时，时延增加到84 x n个周期。因此，作者推测能被同时执行的最大gld指令数为2。

5

PART FIVE

结果总结

性能优化指南



5.1 性能优化指南

基于基准测试结果，作者对SW26010处理器提出了性能优化指南。该指南主要针对3个层级：CPE级、Inter-CPE级、核心组级。

1.CPE级：

这一级的优化目标是开发指令集并行（ILP）。这里有3种方法。
第一，循环应该被展开，指令应该被重组。
第二，浮点指令和数据转移指令应该在双发射流水线中成对出现。
第三，由于缺乏有效的硬件支持，transcendental操作应该由软件程序来代替。

5.1 性能优化指南

2. Inter-CPE级：

这一级的主要编程挑战是重用片上数据。该挑战可用4种方法来应对。

第一，RLC应该与计算重叠，从而隐藏寄存器级通信的成本。

第二，连续的RLC发送指令应该和浮点指令同时发射，以此来隐藏RLC的gaps。

第三，设计有利于RLC的算法，来适应RLC默认的基于行或列的通信。

第四，那些需要访问随机数据的算法，应该采用RLC的动态路由模式。

5.1 性能优化指南

3 .核心组级：

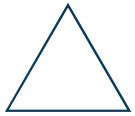
这一级的主要优化目标是最小化内存流量开销。这里有2种方法。
第一，不要使用低效的的全局load/store指令。
第二，主内存和局部数据内存之间的数据交换应该使用DMA。

6

PART SIX

结论

结论



6.1 结论

为了解决SW26010处理器微结构信息公开不足的问题，作者开发了开源的 swCandle 微基准测试套件来评估SW26010处理器的3个主要微结构特征：CPE流水线，寄存器级通信，内存访问方法。

对SW26010处理器微结构的详细研究是对处理器进行性能优化的关键。

THANKS!