A 98 GMACs/W 32-Core Vector Processor in 65 nm CMOS

Xun HE\(^{(a)}\), Xin JIN\(^{(1)}\), Minghui WANG\(^{(1)}\), Daijiang ZHOU\(^{(1)}\), Nonmembers, and Satoshi GOTO\(^{(1)}\), Fellow

SUMMARY This paper presents a high-performance dual-issue 32-core SIMD platform for image and video processing. The SIMD cores support 8/16 bits SIMD MAC instructions, and vertical vector access. Eight cores with a 4-ports L2 cache are connected by CIB bus as a cluster. Four clusters are connected by mesh network. This hierarchical network can provide more than 192 GB/s low latency inter-core BW in average. The 4-ports L2 cache architecture is also designed to provide 192 GB/s L2 cache BW. To reduce coherence operation in large-scale SMP, an application specified protocol is proposed. Compared with MOESI, 67.8% of L1 cache energy can be saved in 32 cores case. The whole system including 32 vector cores, 256 KB L2 cache, 64-bit DDRII PHY and two PLL units, occupy 25 mm\(^2\) in 65 nm CMOS. It can achieve a peak performance of 375 GMACs and 98 GMACs/W at 1.2 V. 

key words: SIMD, cache coherence, NoC, GMACs, multicore processor

1. Introduction

Continuous improvements in image and video processing require high computational power to deal with the increasing complexity of algorithms and higher definition video. Meanwhile, progress in VLSI technology allows the integration of more transistors on a single chip. As multimedia applications have extensive data-level parallelism, massively-parallel single-instruction-multiple-data (MP-SIMD) machines [1], [2] are widely used to exploit as much data-level parallelism (DLP) as possible. The Stream processor [2] contains two cores for main application threads, and a DPU for vector processing. In the DPU, sixteen SIMD lanes are combined to deliver performance of 512 8-bit GOPS or 128 16-bit GMACs with a power efficiency of 12.2 GMACs/W. However, sixteen lanes can only execute one task, and the TLP is limited. Xetal-II [1] is also a SIMD processor with 320 processing elements (PE), and each PE contains one 16bit MAC. It delivers a peak performance of 107 GOPS or 27 GMACS on 16-bit data while dissipating 600 mW. They can achieve high DLP with very good power efficiency as Fig. 1. However, they are insufficient in instruction-level-parallelism (ILP) and task-level parallelism (TLP).

High performance processor architectures are also moving towards integrating more and more cores in one single chip. Multicore processors can deliver high TLP at low power consumption to address peak performance demands, and so they can offer high computational power and design flexibility. The TILE processor [3] is a 64-core SoC targeting at the high-performance demands of a wide range of embedded applications such as network and multimedia applications. Each core is a 3-wide VLIW machine with a 64-bit instruction word and connected through a scalable 2D mesh network. It attains 384 GOPS and 144 billion instructions per second (GIPS) with 10.8 W. Another processor applies 167 fine-grain cores to build a many-core computational platform for DSP, embedded, and multimedia applications [4]. At a supply voltage of 1.3 V, the chip achieves 196.8 GIPS (16bit) or 393.6 GOPS (16-bit), while dissipating 10.2 W. Each core contains 16-bit ALU, multipler, and a 40-bit accumulator, which just costs 0.17 mm\(^2\) in 65 nm. This fine-grain many-core processor is excellent in TLP.

Figure 1 shows the comparison of throughput and power. Compared with MP-SIMD, multicore architecture requires more instructions and power for vector processing. MP-SIMD can achieve better power efficiency, but poor GIPS performance. To meet future multimedia system’s requirement, a SIMD based Vector Processor (SVP) is presented in this paper. The target throughput and power budget (Embedded Processor for Multimedia) is also shown in Fig. 1.

But the growing trend of core number introduces challenges for both cache coherence and communication networks. There are two dominant cache-coherence protocols, namely snoop and directory. In directory-based systems, processors must hold a home directory, which cost a lot of overhead for large scale multiprocessors. On the
other hand, all coherence transactions are broadcasted in snoop-based systems, which mean the BW requirement increases rapidly with core number. Another problem is communication networks for multicore processor. It’s believed that future Multicore and Manycore processors will become communication-bound. Network-on-chip (NoC) emerges as the interconnection fabric of choice for manycore case [5].

In packet-level-flow-control based router architecture, buffer and channel resources are allocated at the level of packets. Large input buffers are required to store the entire packets. Another method is virtual channels based, which can provide an alternate path for incoming packets to bypass a blocked packet, and hence can improve router performance. However, performance is improved at the expense of extra power consumption and area overhead. In Intel’s 48-core processor [6], memory coherency is maintained through software protocols to eliminate the communication overhead for memory coherent 2D-mesh NoC. Message passing memory (MPMT) is used to optimize data share using these software procedures. All MPMT cache lines are invalidated before read/write operations to shared memory, in order to prevent a core from working on the same data. Each core has a private L2 cache to support message-passing-programming model. Large message-passing buffer (16 KB) is used for every tile, and presents 384 KB on-die shared memory. It costs about 500 mW for each router.

In all, in order to meet the demands of multimedia signal processing applications, embedded SoC must provide a high level arithmetic processing power, a sufficient degree of flexibility, powerful on-chip communication network, and an efficient cache coherence protocol. As SIMD architectures have good power efficiency and DLP, we propose a 32-core SVP chip based on dual-issue 128-bit SIMD architecture. The whole system is presented in Sect. 2. Conventional snoop or directory cache coherence protocol costs too much resource for 32-core Shared-memory Multicore Processor (SMP). Thus Sect. 3 proposes an application specified cache coherence protocol to achieve a power-efficient embedded SMP processor. It’s based on snoop protocol and resolves the high BW cost problem for large scale SMP. Section 4 presents a hierarchical communication network based on ring and mesh topologies, which can achieve a high BW and low latency network. The dual-issue SIMD based core architecture is presented in Sect. 5. Section 6 shows the chip and measured results. Section 7 serves as our conclusion.

2. SVP Architecture Overview

For embedded processor, low power requirement is not as critical as portable processor. But if power consumption exceeds 5 W, we still have to be rather careful about the thermal problem. Thus our power budget is about 2–4 W for this high performance embedded processor. Considering flexibility and power efficiency requirements, SMP based multicore architecture is utilized to achieve scalable performance for computationally-demanding image and video applications, such as FTV and UHD video. However, coherence operation increases rapidly with core number, which causes high power consumption and heavy traffic in communication network. To resolve this problem in power-efficient embedded SMP processor, an application specified cache coherence protocol is proposed, called manual control invalid (MCI) protocol. Based on this coherence protocol, a SMP processor with a hierarchical network is designed for high-performance multimedia applications as in Fig. 2.

It contains 32 dual-issue vector cores, 64-bit DDR2 1066 Mb/s memory and 1 MB on chip memory. Each vector core can achieve a maximal frequency as 750 MHz. Dual-issue vector core contains two 128-bit SIMD pipelines and one 32-bit RISC pipeline. Two SIMD instructions or one SIMD instruction with one RISC instruction can be issued in one cycle. As 8/16-bit wide data is widely used in multimedia applications, this SIMD instructions set focuses on 8/16-bit SIMD operations, including 8-bit MAC instructions. To achieve higher frequency, 32-bit MAC SIMD operations are not supported (but 16 × 32-bit MAC is supported). Generally, the bottlenecks of SIMD system are the data movement and rearranging operations, thus a flexible Vector Memory (VM) is designed with vertical vector access ability.

Then eight vector cores make up a cluster, and the whole chip consists of four clusters. Eight cores are connected by Core Interconnection Bus (CIB) in cluster, and then clusters are connected by 2 × 2 mesh network. For 128 cores, 4 × 4 mesh network is enough. Thus this hierarchical network can offer good flexibility for large-scale SMP. In this network, data transmission is divided into 16-byte packages, and several bits are added as package header. The whole chip can achieve about 192 GB/s inter-core BW (within cluster), and 18 GB/s inter cluster BW. Bandwidth management units (BMU) are used to allocate BW at heavy traffic case. DDR II controller (MIU) is also optimized to improve the bandwidth efficiency by eliminating most of the extra overhead as in [7].

In single core chips, L2 cache’s miss rate is one of the most important things. However, bandwidth of L2 cache
becomes more important for large-scale SMP chips. In this chip, DMA units are in charge of moving data between VM and L2 cache. Memory access delay becomes less important for DMA access. Another problem is that MCI protocol is just designed for resolving L1 cache’s coherence problem. Adding one more coherence protocol for L2 cache will increase a lot of hardware cost and complexity. Thus L2 cache in this design doesn’t keep duplicated copies of the same cache line. It means conventional private L2 cache isn’t suitable for this chip. High BW unified L2 cache designs such as [8] seems to be satisfied. However, its crossbar and L2 controller with directories cost a lot of hardware resource. It’s not suitable for our low power embedded processor. New low power L2 cache architecture is proposed to reduce power consumption as in Fig. 3. The L2 cache is divided into four units: L2A, L2B, L2C, and L2D. Each unit is a private L2 cache for one cluster. L2 cache line (64 bytes) is four times of L1 cache line. To achieve high BW, L2 cache units are also divided into four data banks (“Ba”∼“Bd”), which can be accessed independently. To save die area and power, only two-way set associative architecture is used for L2 cache. L2 cache’s tag unit consists of two dual-port SRAM units and can offer two access ports. A duplicated tag unit (“Tb”) which contains almost the same data with “Ta” is also applied to double the BW. Figure 3 shows that the tag units “Ta” and “Tb” can produce four different addresses for four banks. Then four different data paths “data/B/C/D” can be accessed. Thus, one L2 cache unit can finish four data requests within one cycle. L2 cache controller assigns the incoming data access, and arbitrates access conflictions. The disadvantage of private L2 cache is that data share between different clusters becomes inefficient and costs more BW. Special data access paths between different L2 cache units are built for sharing common data. Four L2 units don’t keep duplicated cache lines. For example, if cache miss occurs at L2A, these paths can be used to check other L2 cache units one by one as in Fig. 3. In this process, the other L2 cache units provide data for one private L2 cache like L3 cache. Case “A/B” shows two kinds of data access pipelines and their cycle counts. “Mux” denotes the crossbar stage, and “L3” denotes checking other L2 cache units. In this design, it costs 3 cycles to check one L2 cache’s tag unit. If cache miss occurs in all L2 units as case “A”, the whole process costs about 22~28 cycles. In case “B”, cache hits at the second L2 cache, and it just costs 10 cycles. The hit data won’t be copied into other L2 cache unit, it’s directly sent back to cores. However, most of cache misses on private L2 cache can’t be found in other L2 units. It costs too much resource to send requests to other L2 cache units every time. In MCI protocol, memory space is divided as shared and private space. Thus, L2 cache units can also use memory shared information to determine whether to send out data request or not. Case “C” shows that data request is directly sent to MIU for private data, which can reduce a lot of redundant data requests.

3. Application Specified Cache Coherence Protocol

In conventional snoop cache coherence, coherence transaction broadcasts to all cache monitors. When a processor issues a request to its cache, the other cache controllers has to examine the state of its own cache and takes suitable action, which may generate access operation to tag or data unit of cache. In snoop-based systems, all coherence transactions are broadcasted and therefore seen by all processors in the system. As a result, snoop protocols are generally limited to small-scale systems. Results in [9] indicate that more than 70% percentage of all snoop broadcasts miss in other caches. This means that most of the snoop induced tag lookups just waste energy. Thus an application specified cache coherence protocol (MCI) is proposed, which can reduce about 67.8% L1 cache energy for 32 cores SMP processor.

In the proposed MCI protocol, memory space can be dynamically or statically defined as shared or private space. The shared spaces can be shared by all cores, clusters or several cores. Sharing configuration includes three types of information: address ranges (64-bit), shared clusters (4 bits) and cores (8 bits), and share types. Shared clusters and cores are used to denote which cores are sharing this space. For example, “shared cluster: 0001, cores: 00111111” means that cores 0~5 in cluster A share this space. On the other hand, “shared cluster: 1111, cores: 11111111” means this space is shared by all cores. One sharing space can have more than one sharing configuration. Share types include dynamically and statically sharing. Statically shared spaces are always active, and dynamically shared spaces can be dynamically set as inactive for saving power. When write requests occur at shared space, snoopy unit (SU) in Fig. 5 checks every write request and sends invalid message to sharing cores. For read requests, there are no snooping operations. In the proposed architecture, L2 cache units don’t keep duplicated cache lines, and duplicated cache lines only occur in L1 data.
cache at sharing spaces. Programmer must know well about the memory sharing patterns in applications, and configure the memory map correctly.

OpenMP provides a relaxed-consistency parallel programming model, based on shared-memory architecture [10]. Each thread has access to another type of memory that must not be accessed by other threads, called thread private memory. There are two kinds of access to variables used in the associated structured block: shared and private. In MCI, the whole memory map can be configured as shared or private spaces, and shared spaces also have more attributes than OpenMP. Then we propose a new programming model with an additional memory configuration process, and Fig. 4 shows an example of four parallel tasks.

Two additional steps are needed for memory sharing configurations in MCI. Firstly, memory map should be defined according to MCI. Directive “#Program Share” is used to define the sharing memory patterns, including address range, a set of shared cores and shared type. Shared space “Exchange” is defined as a statically shared space by cores 0–3 in cluster A. Three dynamically shared spaces “CA*” are defined to be shared by two cores. Second, the defined memory spaces are assigned to tasks. Directive “#Program Parallel (C0)” is used to assign task “Function A” to core 0, and then two shared spaced “CA0_1” and “Exchange” are assigned to core 0. “Exchange” is used for sharing data with all of cores which are working together. Dynamically shared spaces “CA*” are used to share data between two cores. Undefined space is private.

In our chip, each core has 16 sets of sharing space configurations (CF0–15) and 4 active configurations including as address range, a set of shared cores and shared type. The chip has four L2 caches and L2 cache can be accessed by the other clusters, providing a shared L3 cache. When L2 cache miss occurs, L2 cache doesn’t copy the hit cache line from other L2 caches. Only L1 cache may have duplicated data and the MCI protocol is applied to reduce coherence message. Compared with conventional snoopy or directory protocol, no extra hardware resources are used to keep the cache line’s share status, saving a lot of hardware resource for large scale SMP. The overhead of MCI is in software development.

Figure 5 also shows an example about how MCI protocol works on shared data access. At first, Cache line I (CI) in shared space “CA0_1” is required by core 0 and 1, and then CI is maintained by L2A and two L1 units. When core 1 writes new data (CI’) to L1, CI’ is stored in filter cache (FC) unit. FC unit has eight entries with LRU replacement policy and is also used to combine small write operations. When CI’ in FC is replaced and sent back to L2 cache, SU checks the address of this write request. As core 1 writes to a sharing space with core 0 (“CA0_1”), SU of core 1 should send an invalid message to core 0’s L1 cache (to guarantee data coherence in CA0_1). Address of CI’ and targeted core number are packaged in SU and sent out through CIB. Then the old data (CI) in core 0’s L1 caches becomes invalid. Core 1 needs to send out a synchronization message, when core 0 and core 1 need to synchronize the sharing data.

In this example, only one message is sent between two cores, and no broadcasting message is sent to the other 30 cores. Coherence operations are reduced from 31 to 1. To evaluate MCI, We use Simics [11] to simulate a 32 cores SMP processor, with 8 KB L1 data cache per core and 1 MB shared L2 cache. In experiments, only snoopy operations of data cache are measured. We drive the experiments with FFT, Raytrace and Radix of SPLASH2-benchmark. In experiments, four tasks are mapped to four clusters, and each task has eight cores. Snoopy-used L1 cache’s energy is measured for both MOESI [12] and MCI. In MOESI, snoopy operations occupy a lot of L1 data cache resource (more than 50%) as in Fig. 6. In MCI, we define four sharing spaces for four tasks, and 32 private spaces are also used for each core. Sharing data such as “global_memory” in Radix is mapped into one of sharing spaces, and the other local variables are mapped into private space. Then eight cores within cluster work together and broadcasting messages are limited by MCI. Comparing with MOESI, MCI can reduce about 67.8% of L1 cache energy in average. Also better
performance can be achieved by deeply optimizing with dynamic sharing configurations. Compared with conventional snoopy protocol such as MOESI in [12], coherence messages for cache miss at private space can be avoided and broadcasting nature can be constrained by software. Bandwidth cost of snoopy messages is mainly determined by data sharing method, not the number of cores. Large scale SMP processor can achieve good performance in MCI. The software impacts of MCI are the two configuration steps, and the complexity depends on the memory sharing patterns.

4. Communication Network

Both packet-level-flow-control and virtual channel based NoC architectures need a lot of large FIFOs to store data packages in network. In Intel’s 6 × 4 mesh NoC, one router costs about 1.1 mm² and 500 mW in 45 nm. Obviously it can’t be used for this embedded processor. CELL processor [13] applies a ring bus based EIB network. It consists of four 16-byte data rings: two running clockwise and two counterclockwise. Each ring can support three concurrent data transfers, as long as their paths don’t overlap. To initiate a data transfer, each core must send requests to a central bus arbiter, which controls all of the data transfers. This architecture doesn’t need to store data in FIFO, but requires a central bus arbiter which limits scalability.

To take advantage of all the computation power on SVP, communication network must offer sufficient BW for inter-core transfers. As eight cores in one cluster share a L2 cache, high bandwidth and low latency data transfer is very critical within one cluster. One vector core’s maximum data throughput BW is about 12 GB/s. In general, more than two instructions are needed to process one data (such as load, store/ALU instructions). Thus communication network should be able to provide each core more than 6 GB/s BW in average. Another issue is that there are two kinds of data communication: 32-bit short message and large data movement between vector cores. To achieve better efficiency, they should be transmitted through different paths.

Figure 2 and Fig. 7 show the CIB, the heart of the SVP’s communication architecture. It consists of two 16-byte data rings and two 4-byte message rings. It supports separate communication paths for 32-bit commands (use 4-byte message rings) and large blocks data movement (use 16-byte data rings). Compared with EIB [13], there is no central arbitrary unit and no bus request operations. Eight bus nodes in a cluster divide CIB into eight segments. Large data is also divided into 16-byte data elements. Data elements are packaged together with target cores and cluster tag as in Fig. 7, and automatically transmitted in CIB. 15 bits are added as package header for automatically routing in the whole communication network. Bus Nodes in CIB are the basic transfer stations, which can accept four packages (two data packages and two message/command packages), send the arrived packages to core and send out the others to next Bus Nodes. Bus node always deals with the received data packages first, and then sends out the new data packages from core at free slots. Figure 7 shows the details of two data links in Bus Node (message links is similar). There is no input FIFO for incoming data from CIB, only a small output FIFO for core. Data in CIB isn’t temporarily stored in FIFO unit. One CIB uses only 12 FIFO units in total. In Intel’s NoC [6], one router needs 5 FIFO units, and eight cores use 40 FIFO units.

With a clock speed of 750 MHz, One Bus Node can support a peak bandwidth of 24 GB/s for inter-core data transfers. However, one data transfer from core 1 to core 4 needs to be propagated three times in CIB. It means that the effective BW for each core depends on the transfer distance. In experiments, we use random traffic pattern to measure the actual BW in a CIB. The destination is also randomly selected. Moreover, we assume that any packet injected in the network is independent, and the arrived packets are consumed immediately. The average packet latency is measured.
under different packet injection rate (16-byte data packets per cycle). Table 1 shows that latency increase rapidly, when injection rate is larger than 70%. When injection rate is less than 50%, latency is less than 8 cycles. It means that each core has 6 GB/s low latency inter-core communication BW in average (192 GB/s for whole chip). This is enough for most of applications. If we just use one 16-byte data ring in CIB, the latency becomes very large when injection rate is larger than 20%. The maximum BW decreases more than 3 times for single ring case, as the average core distance is also increased than dual rings case.

As new packages have lower priority than received packages, new packages have to wait for free slots in CIB. Latency may become very large, when the required path is always occupied. To guarantee an average bandwidth allocation in case of heavy traffic, a Bandwidth Management Unit (BMU) is designed to snatch free slots from others cores. BMU can broadcast a slow down command to the cores which may use the required data link. As in Fig. 7, “output FIFO R” of Node 4 is full of data requests to right side Node (Node 5), then BMU sends a slow down message to Nodes 1–3, which may occupy the required data link. As in Fig. 7, “output FIFO R” of Node 4 is full of data requests to right side Node (Node 5), then BMU sends a slow down message to Nodes 1–3, which may occupy the required data link. It decreases the data issue rate (right side only) of Nodes 1–3 to 25% (within 32 bus cycles). Thus node 4 can get at least 3 GB/s BW to send out packages to Node 5 (within 32 bus cycles). Table 1 also shows that latency can be reduced dramatically for heavy traffic case with BMU.

Mesh network is used to connect clusters. Each router in this 2 × 2 mesh has 3 data ports and 3 message ports as Fig. 8. All input packages are stored in FIFO and then automatically sent to other nodes according to package header. Fixed routes are used in this small mesh network and virtual channel allocation operation [6] can be saved. Compared with [6], the routing latency is reduced by 50%, and large MPB is saved. The power consumption of the whole communication network is only 168 mW (measured at 90% use rate).

### 5. Dual-Issue Vector Core

In video applications, high parallel functions can easily achieve more than eight times speedup in SIMD based processors, but the whole application can’t achieve such an excellent speedup [14]. There are a lot of functions for scalar data in applications, which can’t be optimized by SIMD instructions. It’s a waste to use SIMD register and pipeline to process scalar data. Thus a simple 32-bit RISC pipeline is closely coupled with SIMD pipelines in SVP. To reduce hardware cost, RISC pipeline doesn’t support MMU and branch prediction. It is designed to release vector pipeline from scalar operations, which only costs 23 K gates. To improve the ILP, each core supports dual-issue, including two SIMD instructions or one RISC instruction with SIMD instruction. SIMD based vector core consists of 32-bit scalar pipeline (SP), 128-bit SIMD based vector pipeline (VP), mixed pipeline (MP), 2D-DMA, Snoopy Unit, 16 KB L1 cache and 8 KB Vector Memory (VM) as in Fig. 10. VP supports a lot of common 8/16/32 bits SIMD instructions, except 32×32 bits MAC. SP is the small RISC pipeline without MAC. Vector instructions which needs scalar operand (such as Vector load/store, extract/insert), is executed by MP. As parallelization of MP and VP requires a 5-port register file (RF), Vector RF is divided into two 3-ports RFs as in Fig. 9. Compared with unified 5-port RF, it can reduce about 38% power and 31% area.

For most multimedia applications, 8-bit data is used for denoting one pixel. 8-bit SIMD instructions (including 8-bit MAC) can achieve about 2 times better performance than 16-bit SIMD instructions for some functions. However, 8-bit SIMD instructions require writing back 256-bit vector to vector register file (V RF). 256-bit V RF costs too much, thus most of processors [2], [4] don’t support 8-bit MAC instructions and use 16-bit MAC instead. In this architecture, the temporal results of 8-bit SIMD operations are stored in 256-bit “acc” registers and only saturated 128-bit result can be written to V RF. Then 8-bit MAC instructions are supported with small hardware cost. Figure 10 shows that it’s very useful in video processing.

To improve the data access ability for vertical vector, one four banks VM is designed for vertical, scattered and regular vector types. VM consists of four addressable banks, which also enable non-alignment access (word address) as in Fig. 9. 128-bit vector data is always stored in VM. Figure 9 shows an example of vertical vector access. Vertical Load/store instructions include register numbers, address of VM (16 bits) and Stride value (8 bits). Stride value (“S”) is used to denote data array’s width. To enable vertical access, each element of vertical vector must be putted into different banks in VM.

A 4 × 4 Data array ((0, 0) ~ (3, 3)) is stored in VM as 5 × 4 in Fig. 9. 2D-DMA unit supports such kind of data movement from external memory to VM. The fifth column is useless data, which denotes as (x,x) in Fig. 9. Then ad-

<table>
<thead>
<tr>
<th>Injection Rate* (%)</th>
<th>1%</th>
<th>30%</th>
<th>40%</th>
<th>50%</th>
<th>70%</th>
<th>75%</th>
<th>80%</th>
<th>85%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dual Rings</td>
<td>3.2</td>
<td>5.1</td>
<td>6.5</td>
<td>7.5</td>
<td>18</td>
<td>40</td>
<td>140</td>
<td>490</td>
</tr>
<tr>
<td>Dual Rings+BMU</td>
<td>3.2</td>
<td>4.8</td>
<td>5.7</td>
<td>6.8</td>
<td>14</td>
<td>27</td>
<td>78</td>
<td>218</td>
</tr>
<tr>
<td>Single Ring</td>
<td>4.8</td>
<td>5.35</td>
<td>6.4</td>
<td>8.7</td>
<td>13</td>
<td>27</td>
<td>121</td>
<td>448</td>
</tr>
<tr>
<td>Single Ring+BMU</td>
<td>4.8</td>
<td>5.32</td>
<td>6.7</td>
<td>8.7</td>
<td>11</td>
<td>21</td>
<td>87</td>
<td>210</td>
</tr>
</tbody>
</table>

*: BW = Inject Rate x 16x0.75 GB/s

**Table 1** BW and latency (cycles) for single/dual rings CIB.
address generation unit (AG) can generate four bank addresses for B0~4. To access the first column, AG outputs 0/1/2/3 for “addr0/1/2/3” with S=1, and data (0,0)~(3,0) is selected as in Fig. 9. To access the second row (1,0)~(1,3), AG outputs 2/1/1/1 for “addr0/1/2/3” with S=0. This architecture just can access 32 bit vertical vector. All of the features and its benefits are shown in Fig. 10. Before optimization with feature a/b/c, this design is poorer than CELL. After optimized with the features one by one, it can achieve about 2.3 times speedup. As CELL’s SIMD core doesn’t support 8-bits MAC or non-alignment access, the proposed work can achieve about 1.4 times better performance than CELL. However, this design doesn’t support 32-bit MAC, and target applications are limited to multimedia applications.

2D-DMA unit manages data transfer between main memory and VM, including core to core transfer through CIB. To improve small data transfer between VM and external memory, DMA unit always sends data request to L2 cache (not main memory). Applications which rely heavily on small size or gather accesses to main memory can take advantage of this feature. Compared with CELL’s DMA [13], the BW performance for small size DMA (< 512 bytes) is about 2 times better.

We also extend the filter cache (FC) proposed by J. Kin et al. [15], in which an extra, small cache (called filter cache) is inserted between the CPU data path and the L1 cache and serves to filter most of the references initiated from the CPU. In this work, FC unit is used to reduce the energy of L1 data cache. It maintains eight entries for recently accessed data cache line (16-byte). Write-back data is also temporally kept in FC to combine small write-back data. When dirty cache lines in FC are displaced (LRU replacement policy), they are written through to L2 cache. Synchronization operation flushes out all of the dirty cache lines to keep the data coherence. In experiments, it can save about 87% power of L1 data cache in MP3 decoder application.

6. Chip Implementation and Performance Evaluation

6.1 Chip Implementation and Measurement

The presented chip is fabricated in SMIC 65 nm CMOS technology. Figure 11 shows the die micrograph of the 32-core dual-issue processor. The die occupies 25 mm², including 32 vector cores, 256 KB L2 cache, 64-bit DDRII’s PHY and two PLL units. Each cluster with CIB and a mesh router occupies 4.4 mm². Extensive clock gating is applied at cluster level, core level, and register level to reduce the power consumption. The power consumption at different supply voltage is measured by the evaluation board. Figure 12 shows measured power and frequency from 0.8 to 1.2 V. The power consumption is measured with 100% utilization rate of 32 cores and about 50% utilization rate of L2 caches at 25°C. The maximum speed of 750 MHz is achieved at 1.2 V with 3.7 W. DDRII PHY is also verified at 533 MHz, 800 MHz and 1066 MHz. The power consumption of different instructions is also measured by forcing all cores to repeat the same instruction. Figure 11 also shows average power consumption (per core) of several instructions. For “NOP” instruction, NOP_CG means turning off clock gating unit of each core, but L2 cache and MIU unit are working. 16-bit SIMD MAC operations cost more power than the others.

The chip can also achieve better power efficiency at lower power supply. It’s not stable at 0.8 V. Then the recommended power condition for better power efficiency is 0.9 V. This chip consumes about 0.5 W while operating at 220 MHz and 0.9 V. There are two PLL units for cores and DDRII PHY unit respectively. To save power consumption, clusters can work at lower frequency. DDRII can also work at lower frequency to save power by configuring the other PLL.
6.2 Performance Evaluation and Comparisons

The performance of MAC instruction is widely used in marketing literature as an indicator of processor’s performance [16], which is also widely offered. Then we measured the GMAC performance of this chip, considering the overhead of setting up data for processing. The whole chip can achieve a peak performance of 375 GMACs, or 750 GOPS of 8-bit data operations. GMACs/W is treated as a general indicator of power efficiency. After resolving the scalability and cache coherence problem, this large-scale SMP processor can achieve 98 GMACs/W at 1.2 V. The chip is compared with two similar high performance processors, which are also designed for multimedia applications. Table 2 shows performance and power efficiency comparison in GMACs/W. For eight bits pixel based data processing, it can achieve 1.9 times higher GMACs performance than the 167 many core chip. For 16 bit GMACs/W, this chip can still achieve about 60% lower power consumption than [4].

The proposed SIMD core architecture is designed for multimedia applications. Max Baron’s report [16] shows that DCT and SAD kernels occupy about 45% of total cycles for MPEG4 codec. SAD becomes dominant in video codec. Thus, we select SAD, DCT and a $3 \times 3$ matrix multiply [16] as our benchmark for evaluating cycle level performance. VCP [17] and several common used DSP processors [16] are compared with proposed design. Table 3 shows the cycle count comparison. For SAD and Matrix Multiply, the proposed work can achieve more than 1.5 times better performance than the other DSPs. As instruction parallelism of DCT $8 \times 8$ is very high, VLIW based DSP can achieve better performance than our design. VCP has three RISC and SIMD pipelines which is three times of our design, thus it’s excellent in both data and instruction parallelism. As most of applications don’t have enough parallel SIMD instructions and sequential parts become the bottleneck. Thus, it just can achieve about 1.5 times better performance than the proposed work in this experiment.

Details of performance and power of an edge detection application (frame: $720 \times 480$) are presented in [17]. We also measure time and power of this application and compare with VCP in Table 4. Application level energy efficiency is defined as the energy cost for processing one frame in uJ. Table 4 shows that VCP (3RISC+3SIMD) is about 4.4 times larger than our design. Our design uses smaller architecture and higher frequency for achieve higher application level performance. Time cost per frame is about 1.4 times better than VCP. Then proposed work can achieve about 2.1 times better energy efficiency than VCP.

To justify the performance of the proposed L2 cache, we compare it with unified L2 cache as in Table 5. They are designed with the same port number, bank architecture (as L2B) and capacity, which can guarantee the same BW. Unified L2 cache can be organized as 256 bytes per line,
which may get better performance for some applications. Hardware cost, average memory access delay (in cycles) and miss rate (MR) are evaluated. Hardware cost only includes the arbitrary and multiplexer units. Overhead of cache coherence for unified L2 cache design is not considered in this experiment. The same applications as Sect. 3 are used. Applications work in parallel within 32 cores. Table 5 shows that unified L2 cache uses a lot of hardware resource for the crossbar and arbitrary. Cache miss rate of proposed work increases about 0.12%, comparing with unified L2. In proposed work, cache miss needs about 3–9 extra cycles as in Fig. 3. In all, the proposed work sacrifices 0.08 cycles memory access time for reducing 63% hardware cost in average. As DMA can hide the memory access delay for vector data, this cost becomes acceptable.

7. Conclusion

A large-scale SMP computational platform that is well-suited for video and image processing has been fabricated in 65 nm CMOS. This chip contains 32 dual-issue cores, which supports 128-bit SIMD instructions (including 8-bit MAC) and vertical vector access. Optimization with these features can achieve 2.3 times speedup in average for several 2D image processing kernels. A filter cache is utilized to reduce L1 data cache access and combine write operations. In large-scale SMP systems, traditional snoop protocols cause a lot of snoopy operations, which occupy a huge BW and L1 cache resources. A new application specified cache coherence protocol is proposed to reduce BW costs and L1 cache’s energy. Compared with MOESI, more than 67% L1 cache energy can be reduced for this 32-core chip. Better performance can also be achieved by deep optimization on memory share patterns. Four CIB networks are used for clusters, and providing 192 GB/s inter-core communication BW in average. Fixed rout mesh network is also applied. This hierarchical network can achieve lower latency and better energy efficiency than mesh. Based on those architectures, this chip can achieve good energy efficiency and DLP as MP-SIMD, and also provide high ILP and TLP as CMPs.

Acknowledgments

This research was supported by “Ambient SoC Global COE Program of Waseda University” of the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References


Table 5 Performance comparisons of three L2 cache designs.

<table>
<thead>
<tr>
<th>applications</th>
<th>Unified [8]</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gate count: 233.6K</td>
<td>Gate count: 87.1K</td>
</tr>
<tr>
<td></td>
<td>MR</td>
<td>delay</td>
</tr>
<tr>
<td>Radix</td>
<td>1.84%</td>
<td>2.35 cycles</td>
</tr>
<tr>
<td>FFT</td>
<td>1.31%</td>
<td>2.21 cycles</td>
</tr>
<tr>
<td>Raytrace</td>
<td>0.41%</td>
<td>2.12 cycles</td>
</tr>
</tbody>
</table>
Xun He received the B.S. degree and M.E. degree in Electronics and Information from Graduate School of Electronic, Information and Engineering of Shanghai Jiao Tong University. He is currently a Ph.D. candidate in Graduate School of Information, Production and Systems, Waseda University, Japan. His research interests include video coding technology, multicore processor.

Xin Jin received the M.S. degree in Communication and Information System and Ph.D. degree at Huazhong University of Science and Technology, Wuhan, China, in 2002 and 2005 respectively. She is a visiting lecturer of Information Technology Research Organization, Waseda University. From 2006 to 2008, she was a Postdoctoral Fellow in the Department of Electronic Engineering at the Chinese University of Hong Kong. Her current research interests include power-rate-distortion theory, power-constrained video processing.

Minghui Wang received the B.E. degree in Department of Electronic Engineering from Tsinghua University in 2007. He is now a Ph.D. candidate in Graduate School of Information, Production and Systems of Waseda University. His research interests are video processing.

Dajiang Zhou received B.E. and M.E. degrees from the Department of Electronic Engineering of Shanghai Jiao Tong University, China, in 2005 and 2008, respectively. In 2010, he received the Ph.D. degree from the Graduate School of Information, Production and Systems of Waseda University, Japan. Currently, he is JSPS PD fellow and working in high performance video decoder chip design at Waseda University.

Satoshi Goto received B.E. and M.E. degrees in Electronics and Communication Engineering from Waseda University in 1968 and 1970, respectively. He received the Doctor of Engineering degree from the same university in 1981. He joined NEC Laboratories in 1970 where he worked on LSI design, multimedia systems and software as GM and VP. Since 2003, he has been a professor at the Graduate School of Information, Production and Systems of Waseda University. Currently, his main interest is in VLSI design methodologies for multimedia and mobile applications. He has published 7 books, 38 journal papers and 67 international conference papers. He served as GC of ICCAD and ASPDAC and was a board member of IEEE CAS society. He is IEEE Life Fellow and Member of Academy Engineering Society of Japan.