AN OPEN SYSTEMS APPROACH TO REAL-TIME SIGNAL PROCESSING USING AltiVec TECHNOLOGY

Richard Jaenicke
Mercury Computer Systems, Inc., Chelmsford, MA
info@mc.com

Abstract

Providing vector processing in a standard RISC chip, the AltiVec™ technology enhancements to the PowerPC™ architecture hold the promise of the signal processing power of a DSP chip with the programming ease of a RISC chip. This paper explores how this new technology can fit into an open architecture for real-time image and signal processing.

The open systems architecture described herein includes both software and hardware components. The open software architecture addresses how to achieve high performance and productivity for signal processing algorithms while using standard APIs and high-level languages to promote portability and maintainability. The hardware architecture addresses high-performance memory subsystem design and scalable communication bandwidth between multiple processors, also adhering to standards to facilitate technology insertion.

The paper then addresses how to integrate AltiVec technology into the open software and hardware architectures. Finally, an example implementation is provided for a real-time multiprocessor system.

AltiVec Breaks Moore's Law

The motivation for finding a way to include AltiVec-enabled processing into our open architecture strategy is that it fundamentally jumps to a new performance curve that charts the performance increases according to Moore's Law.

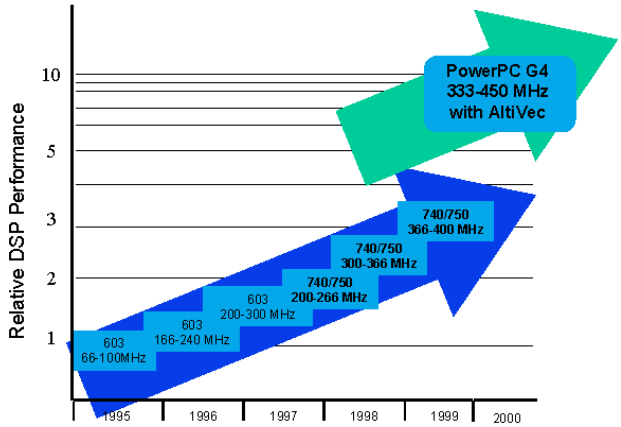


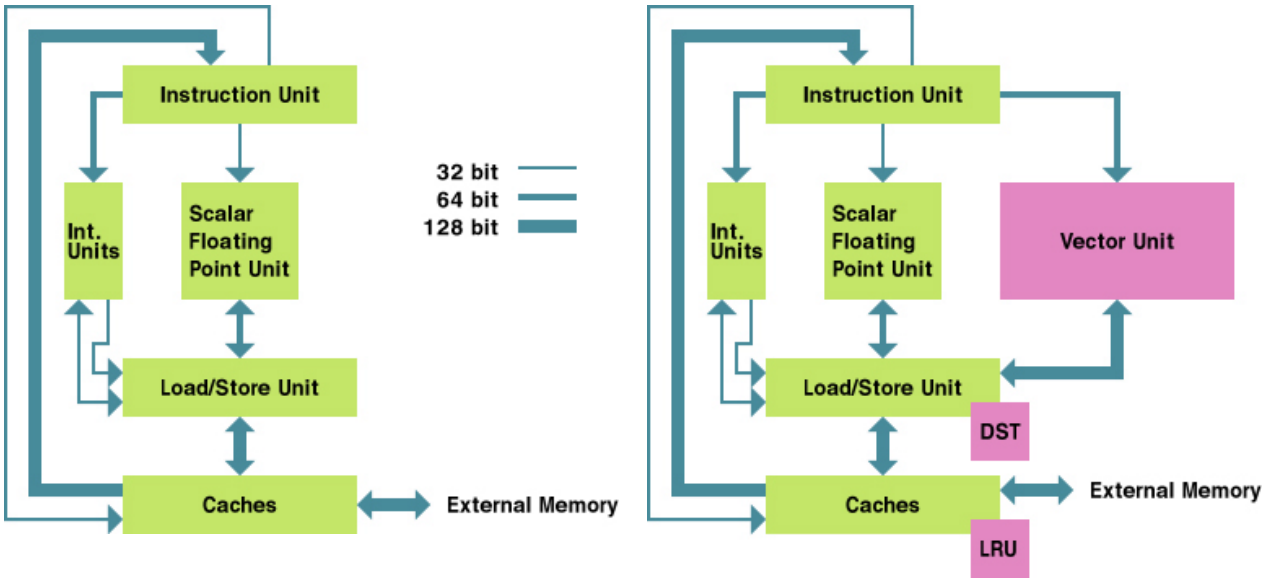
Figure 1: PowerPC performance over time

As Figure 1 shows, the performance of the PowerPC processor has increased at a constant rate over time. The introduction of AltiVec technology has caused a step function in the performance increase. The goal is to ride the new, higher curve instead of the old one.

AltiVec Technology Overview

AltiVec technology includes a vector execution unit, 32 new vector registers, and many new PowerPC instructions[1]. The vector execution unit executes a single instruction on multiple data (SIMD), and it is in addition to the standard integer and floating-point execution units (Figure 2). The data comes to the vector unit from a 128-bit wide vector register file, which is independent of the integer and floating-point register files. The enhanced

[1] See http://www.altivec.org for a complete list of AltiVec-related papers and articles, including technology overviews.



(a) (b)

Figure 2: PowerPC execution units (a) before AltiVec and (b) with AltiVec

instruction set includes 162 new functions, most of which just extend the scalar instructions into the vector domain. There are, however, some notable additions:

- New instructions for field permutation and formatting
- Load/store options for cache management
- Instructions that control new data prefetch engines

The 128-bit vector can be used as four 32-bit floating-point numbers, four 32-bit integers, eight 32-bit integers, or sixteen 8-bit integers. The latter is particularly useful for operating on pixel data. For image and signal processing, which contain long streams of data, the ability to execute arithmetic instructions on multiple samples simultaneously translates directly into increased performance over competing architectures (Table 1). This paper will concentrate on the floating-point calculations as used in high-end signal processing applications.

Table 1: Competitive AltiVec timings for DSP functions[2]

Function	AltiVec	MMX & SSE	Units
1024 point square (bsqr1)	0.5	1.3	clock cycles per element
1024 point multiply (bMpy2)	0.75	1.33	clock cycles per element
1024 point dot product (DotProd)	0.5	2.21	clock cycles per multiply-add accumulate
256 point complex FFT (FFT)	< 4	6.94	clock cycles per floating-point element
32 tap x 1024 point FIR filter (bfir)	0.33	0.95	clock cycles per multiply-add accumulate
32 tap x 1024 convolution (conv)	11	85	clock cycles per output point

[2] Timings from Apple's developer's web page:
http://developer.apple.com/hardware/altivec/summary.html

Open Systems Software Architecture

An open software architecture is one that publicly defines all interfaces. The best open architectures become recognized standards. A complete AltiVec system must embrace an entire hierarchy of such standards.

The open real-time software architecture described here is based on four main application programming standards: the ANSI C programming language, real-time POSIX for the operating system, VSIPL for the image and signal processing library, and MPI/RT for real-time multiprocessor communication. These APIs are also the main components of other real-time common operating environments, such as the U.S. Navy's tactical advanced signal processor (TASP).[3]

The combination of these four standards can fully exploit AltiVec performance, while enabling application source code portability.

Real-Time POSIX

An open systems architecture can use multiple operating systems, each optimized for a specific task. Users typically dedicate the vast majority of the processors in a signal processing system to computation. These processors should run a specialized operating system that is optimized for computation.

The remaining processors in the system provide data interfaces to networks, disks, user workstations, or just about anything else that can sink or source data streams. Such *I/O* or *host* processors execute operating systems optimized for control and I/O, such as VxWorks or Solaris.

There is something common to all these operating systems – the real-time kernels are based on the POSIX minimum real-time profile[4]. That standard API dictates a lightweight interface for all common operating system functions. Particular attention is paid to signals, threads, timers, and I/O.

A user might develop a parallel program on a Sun workstation, then later deploy that program in a heterogeneous Mercury system leveraging both VxWorks and Mercury's MC/OS runtime environment. The same source code can migrate from development to deployment, provided the programmer builds upon the four standards described within this paper.

The VSIPL API

Most application developers prefer to never write a line of AltiVec-specific source code. Instead, users gain the performance advantages of processor specific enhancements, such as AltiVec, by calling hand-crafted libraries provided by system vendors or third parties.

The Vector, Signal, and Image Processing Library[5] (VSIPL) is a standard API sponsored by the Defense Advanced Research Projects Agency (DARPA). The VSIPL standard was optimized for real-time signal processing applications hosted on low-power chips such as the PowerPC processor or traditional DSP chips. The VSIPL standard is sufficiently general for typical engineering workstation implementations as well. The entire API consists of more than 1000 functions, including multidimensional image processing. The standard also includes profiles that define subsets of the standard for streamlined implementations. The most commonly used profile is the Core Lite profile[6], which includes the 125 functions most commonly needed for one-dimensional signal processing.

The VSIPL API is object-based and takes deliberate steps to hide the library intricacies from the user. Two abstract data types, *blocks* and *views*, form the base for the API. A *block* is a section of memory that is contiguous,

[3] See also http://www.tasp.org.
[4] IEEE Std 1003.13-1998, 19 March 1998.
[5] See also http://www.vsipl.org.
[6] VSIPL Forum, "VSIPL Core Lite Profile, Version 1.0," 1999.

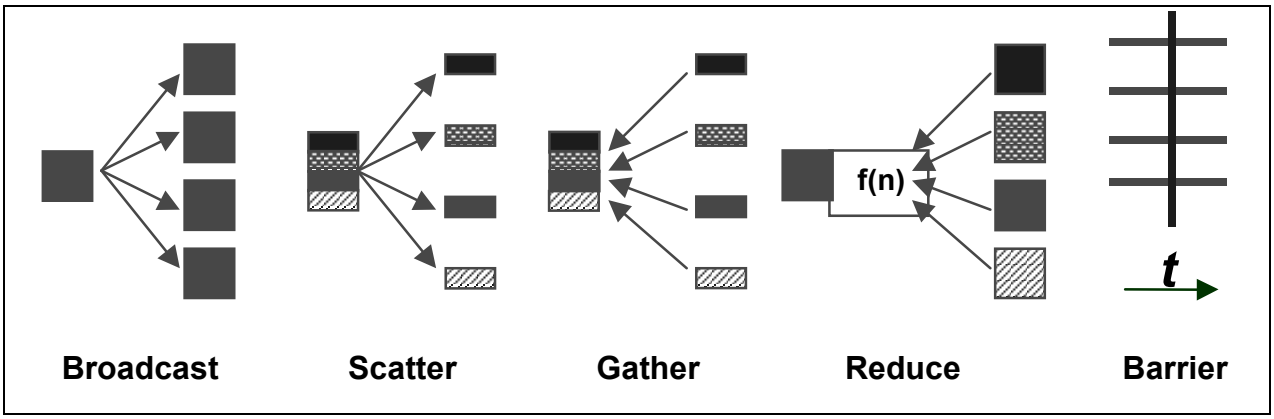


Figure 3: Collective communication operations in MPI/RT

or at least appears so to the user. How the applications see that memory is defined by a *view* into the block. Views are described by an offset, stride, and length, all relative to the beginning of a particular block. Different views, such as a vector and matrix, could share the same data by pointing to the same block.

To discourage users from circumventing the VSIPL API, two distinct types of memory are defined: private and public. Private memory is allocated by VSIPL and can be accessed only by VSIPL functions. Memory allocated outside of VSIPL is public, and must be admitted to VSIPL before general VSIPL functions can access it. Once admitted, that memory is owned by VSIPL until it is released.

Of particular interest for supporting AltiVec instructions is that VSIPL supports both split and interleaved formats for complex floating-point data. Optimum performance from AltiVec functions is generally achieved using the split format, where the real and imaginary values are stored in separate arrays.

The MPI/RT API

The threads or processes that make up every parallel program must exchange data and synchronize events. MPI/RT[7] is an API for communicating among multiple processors in a real-time system. MPI/RT has its basis in the Message Passing Interface (MPI) standard, but the types of extensions needed to enable real-time performance made it neither a proper subset nor superset of MPI.

To effectively coordinate the construction of communication channels for real-time multiprocessor communication, MPI/RT defines a two-stage setup process before beginning the real-time communication. Called *deferred early binding*[8], this two-stage process first defines the resource object needed on all the participating processing nodes and then goes to a *commit* stage for the rest of the setup. The definition stage is performed independently on each processor, while the commit stage is optimized through interaction across the processors.

During the commit stage, the information on the requested objects is exchanged among nodes, those objects are validated, resources are reserved to populate the objects, and finally the global and local resource utilization schedules are optimized and synchronized.

There are three types of persistent objects defined in MPI/RT: buffers, buffer

[7] Real-Time Message Passing Interface Forum. "Real-Time Message Passing Interface (MPI/RT-1.0) Standard." http://www.mpirt.org.

[8] A. Kanevsky, A. Skjellum, and J. Watts, "Standardization of a Communication Middleware for High-Performance Real-Time Systems," RTSS'97, Dec 1997.

iterators, and channels. Buffers hold the messages to be sent or received. Buffer iterators are queues of buffers tasked for a particular communication. Channels are user-defined connections among the nodes, each with an input buffer iterator and an output buffer iterator. Channels can be point-to-point connections or collective operations, including broadcast, reduce, scatter, gather, and a barrier synchronization (see Figure 3). It is important to note that as buffers move through a buffer iterator to a channel, no copying of the buffer takes place.

ANSI C

Most real-time signal processing developers program in C. However, no C compiler to date has been able to automatically generate code for the PowerPC's AltiVec vector unit. Thus we suggest that users gain the benefits of AltiVec through optimized library calls (which we discussed earlier).

Someday this may change. For example, Pacific Sierra Research Corporation, famous for its supercomputer vector code generation tools, is promising AltiVec support.

Another option is the potential for an enhancement to the C language that will make it easier for compiler writes to emit vector code. Such work is going on within ANSI under the label "DSP-C".

Finally, for users who simply cannot wait for portable AltiVec programming solutions, Motorola has defined extensions to the C language unique to AltiVec technology. Use of Motorola's extensions provides high performance and allows portability between PowerPC compilers. However, it is very unlikely that non-PowerPC compiler vendors will ever support these AltiVec extensions.

Open Systems Hardware Architecture

Open software enables the portability of application source code between execution platforms, development workstations, and optimized deployment environments, for example. In contrast, open hardware standards allow users to configure unique deployment environments using hardware modules acquired from multiple sources.

The foundation for the open hardware architecture defined here is the ANSI standard RACEway Interlink[9]. Implemented as a high-speed crossbar-switched network, RACEway can connect hundreds of processors with tens of gigabytes per second bandwidth.

The open hardware architecture is designed for heterogeneous processing, typically using off-the-shelf microprocessors and DSP chips. This enables the new PowerPC G4 processors with AltiVec technology to be mixed with other PowerPC processors as well as with DSP chips, FPGA-based compute nodes, and I/O nodes.

Data can enter and exit the system through a direct connection into the RACEway network or though one of several standard interface options. For example, real-time sensor data passing through an A/D converter board typically enters the system through an ANSI standard interface for 32-bit parallel TTL data called the Front Panel Data Port (FPDP)[10]. When used in conjunction with a RACEway-to-PCI bridge chip, many other off-the-shelf interfaces are compatible. These include ATM, Ethernet, Fibre Channel, SCSI, VME, and many more.

RACEway Interlink

RACEway is a point-to-point communication fabric designed for real-time multiprocessing[11]. Each communication path transfers 32-bits of data on each clock. The original specification for a 40 MHz clock

[9] ANSI/VITA 5-1994 RACEway Interlink. Any ANSI/VITA standard may be ordered from VITA at http://www.vita.com.
[10] ANSI/VITA 17-1998 Font Panel Data Port. See also VITA 33, a proposed standard for a serial version of FPDP.
[11] T. Einstein, "RACEway Interlink - A Real-Time Multicomputing Interconnect Fabric for High-Performance VMEbus Systems," *VMEbus Systems*, February 1996.

provided 160 MB/s peak bandwidth per connection. A recent upgrade to RACEway, called RACE++, uses a 66.66 MHz clock for 267 MB/s peak bandwidth per connection.

Processing nodes, memory nodes, and I/O nodes all interface to the RACEway Interlink through a RACEway interface chip (RIC), which is either an FPGA or an ASIC. Each RIC typically includes a DMA engine to off load the communication processing from a processor. Advanced interfaces also have the ability to link together a number of DMA command blocks into a DMA chain. This further reduces processor intervention.

The communication fabric itself is implemented using a network of crossbar switches. These switches create multiple, simultaneous point-to-point connections, thus providing much greater bandwidth than a shared bus. The current RACE++ crossbar has eight 32-bit ports, enabling four simultaneous connections for an aggregate bandwidth over 1 GB/s from a single crossbar chip. Multiple crossbar chips are connected to form a highly connected fabric (see Figure 4), so that the bisection bandwidth[12] is the same as the aggregate bandwidth.

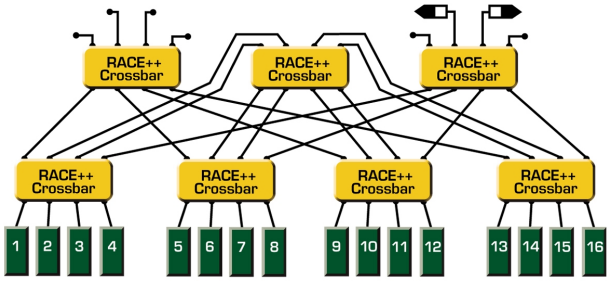


Figure 4: Example RACEway topology

RACEway enables real-time communication by using adaptive routing to avoid congestion in the network and also by providing priority transfers when the congestion cannot be avoided. RACEway networks are generally constructed to provide multiple connections between two crossbars. This enables alternate routes to be chosen at multiple points along the communication path. The choice of path is done automatically.

For times when even the alternate paths are busy, RACEway provides priority transfers. Such transfers will "pull rank" on lower priority communications, ordering them to terminate immediately. The lower priority communications then stop sending data, and terminate their connection after data in the crossbar "pipe" has been flushed. The connection is automatically reestablished after the higher-priority communication is completed.

RACEway Systems

RACEway is most commonly used in VMEbus systems. The RACEway network connecting multiple processors on a VME board is extended to provide communication between boards by sending the signals through the user-defined I/O pins on the VME P2/J2 connector. Backplane overlay cards attached to the standard VME backplane use those signals to connect the networks on each board into a single crossbar-switched network.

A recent extension to the RACEway standard[13] provides for two RACEway connections through a 5-row connector defined by the VME64 standard[14]. This effectively doubles the bandwidth through the point in the system that is usually the communications bottleneck.

Dual RACEway connections are also available for industrial PCI systems. These standard PCI form-factor boards use the PCI edge connectors on the bottom of the board to connect to a standard PCI passive backplane. Additional connectors on the top of the board connect RACEway Interlink backplanes across the top of multiple PCI boards. In this manner, dozens of processors may be connected together in a single PCI system.

[12] The bisection bandwidth is the worst-case bandwidth that occurs when one half of the nodes communicate with the other half. It is often used as the measure of communication performance for a network that is heavily loaded.

[13] ANSI/VITA 5.1-1999 RACEway Interlink.
[14] ANSI/VITA 1-1994 VME64.

AltiVec in the Open Architecture

The remaining task is to incorporate AltiVec technology into the described open systems architecture. Both the software and hardware architectures must be addressed.

AltiVec Impact on the Software Architecture

Implementing the open software architecture with processors that include AltiVec technology requires paying attention to a few implementation details. Modifications are needed in the operating system, the algorithm libraries, and even the communication library.

The main impact on the operating systems is that of the new vector registers. This new, large set of registers needs to be saved and restored every time there is a context switch. Although this is not a difficult enhancement to the OS, it is a crucial one.

A more substantial enhancement is needed if the OS is to support Java mode on PowerPC processors with AltiVec technology. In Java mode, the AltiVec engine will process denormalized results properly. That is, the processor will perform gradual underflow. The cost for this safety net is an extra cycle of latency in floating-point operations. Java mode requires a significant body of new OS code that Motorola does not provide.

However, it is the algorithm libraries, not the OS, where a majority of the changes are needed to support AltiVec. Fortunately, the VSIPL architects had some understanding of optimizing for AltiVec instructions and designed the library API with AltiVec in mind. The VSIPL API includes a provision for splitting a complex data stream into separate real and imaginary data arrays, as required for the implementation on a particular processor. This format conversion is done implicitly during the admission or release of public memory.

More substantial changes are needed when implementing algorithms in the library. At this point, some understanding of the processor architecture is crucial for achieving optimum performance. For example, the most efficient use of the vector execution unit occurs when the loads and stores for the next vector are overlapped with the computation for the current vector. Particular attention must also be paid to the cache-control options to perform vector chaining.

When the application developer needs to access the AltiVec execution unit outside of a standard library, the best programming method is to use the AltiVec C extensions. These extensions look like a function call with vectors as arguments. Different load and store instructions are used for different cache control options.

The compiler enhancements required to handle AltiVec C extensions are not a part of the runtime architecture described here, and there are also other changes required to the compile-debug tool chain. Luckily, several compilers exist today that parse AltiVec C extensions.

Compared to systems based on DSP chips, this approach has the benefit of high-level language support for the vector instructions and implementation in several robust compilers available from multiple vendors.

AltiVec Impact on the Hardware Architecture

The open hardware architecture easily accommodates processors with AltiVec technology, as the architecture was designed from the beginning to support heterogeneous processing. A single ASIC can connect the PowerPC processor with AltiVec technology to the RACEway communications fabric and to the local memory for that processor.

The PowerPC processor then can read data from and write data to any other processor, memory, or I/O node in the system. Remote processors can also read data from or write data to the memory of the local node.

Because the interfaces to the multiprocessor boards are all standard interfaces –

RACEway, FPDP, and VME or PCI – AltiVec technology processor boards can be substituted directly in the system for any previous processor board type.

Example Implementation of the Open Systems Architecture

Because of the low power consumption of PowerPC processors, even those with AltiVec technology, it is possible to include four processors on a 6U VME board complete with each processing node containing L2 cache, SDRAM, and a RACEway interface in addition to the processor (see Figure 5). It is also possible to implement a processing node that contains multiple processors attached to a single RACEway interface / DRAM pair.

Each RACEway interface contains an enhanced DMA engine to achieve the highest possible performance from the RACEway interconnect. In addition to being able to transmit blocks of data based on a processor command, the DMA interface is also capable of chaining those commands and providing strided memory accesses. DMA command chaining amortizes the startup overhead for a DMA transfer over multiple DMA data blocks. Strided accesses enable sending a sub-image or sub-array in a single DMA block. Without strided DMA, many 2-D image and signal processing algorithms would overload a system when processing the data on a distributed set of processors.

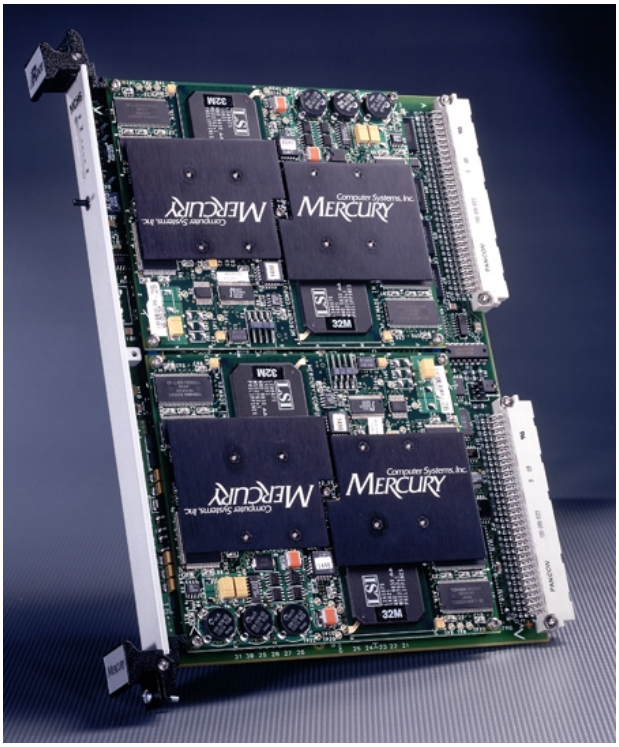


Figure 5: Four AltiVec processors on a 6U VME board. The processors are under the heat sinks.

Figure 6: 6U VME system with 80 processors

A pair of processor nodes is mounted on a processor daughtercard, which has two RACEway connections to the motherboard. The VME motherboard has a RACEway crossbar switch that connects the four processors to each other and provides two links through the backplane to the larger RACEway network.

Up to 20 VME boards can be connected by the RACEway network. 6U VME systems can hold up to 80 processors (see Figure 6) with an aggregate processing performance of 128 GFLOPS. 9U VME systems can hold up to 360 processors with an aggregate processing performance of 576 GFLOPS.