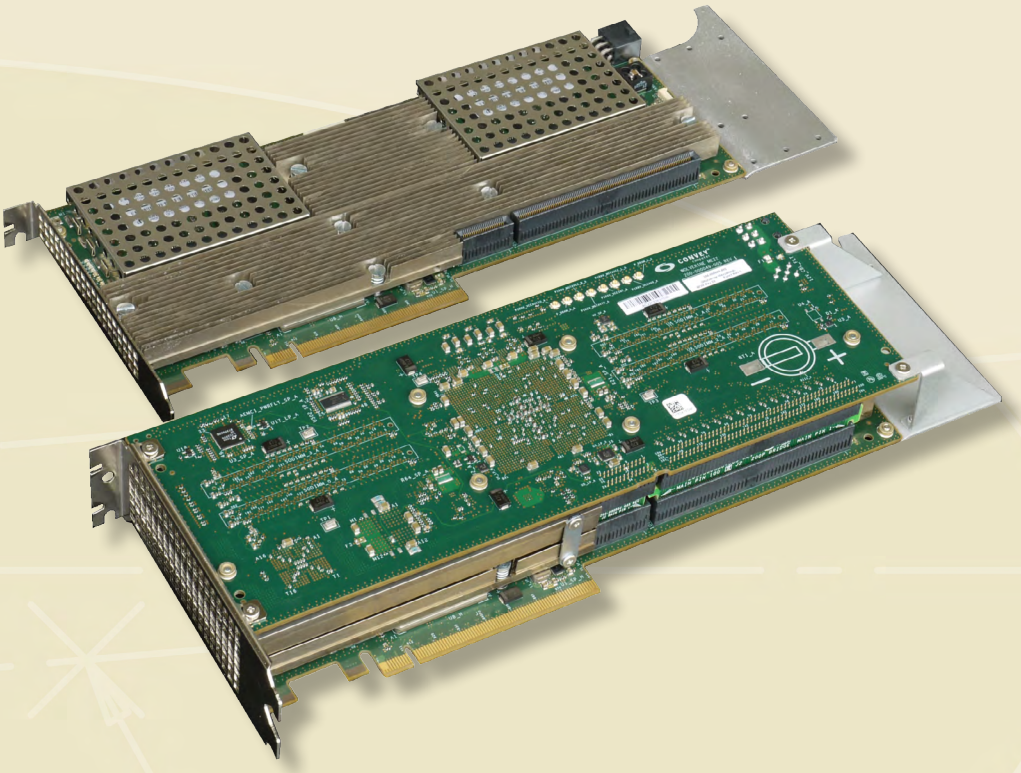
Convey Wolverine® Application Accelerators

Architectural Overview



CONVEY COMPUTER™

Convey Wolverine® Application Accelerators

Architectural Overview

Introduction

Advanced computing architectures are in a continuous race with application performance requirements. As soon as an architecture is developed that meets the needs of current applications, the next generation of "grand challenge" requirements for more performance, more memory and faster I/O leaves it behind. Today that race is driving data center managers with "big data" problems to add ever-increasing numbers of servers to their scalable architectures. But their data centers are running out of power, space, and cooling capacity. In addition, many of today's applications don't scale well; adding more servers leads to a point of diminishing returns. Hybrid (heterogeneous) architectures employing custom logic, FPGAs, and GPGPUs present an alternative that dramatically increase performance per watt, putting architectures back into the race.

The Wolverine accelerators are compact, general-purpose application accelerators based on FPGA technology. They are implemented as PCI Express form factor cards that can be installed in a variety of off-the-shelf servers (Figure 1).

Contents

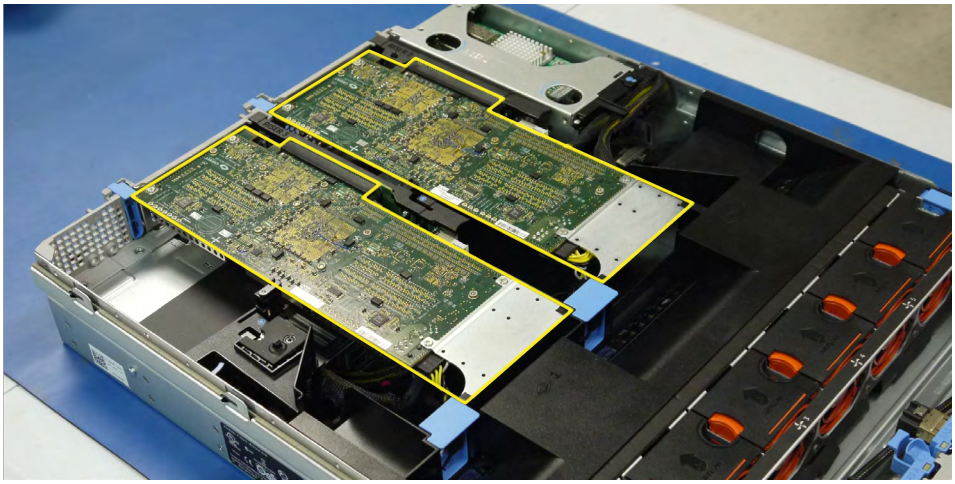


Figure 1. A Dell R720 with two Wolverine accelerators.

Wolverine application accelerators provide a "drop-in" acceleration capability that can dramatically increase the performance of conventional servers.

CONVEY COMPUTER™

Convey Wolverine Architecture

Convey Wolverine application accelerators are PCIe Express form factor coprocessors that provide hardware acceleration of key algorithms. Based on Convey's hybrid-core technology, Wolverine coprocessors can be installed in off-the-shelf x86 servers, and are perfect for accelerating applications in hyperscale, life sciences, big data, security, and other industries requiring accelerated performance.

The coprocessor incorporates the latest high density Xilinx® FPGAs and is targeted at applications requiring very high compute capability and large, high bandwidth memory. FPGAs can provide substantially better performance per watt than conventional servers, and can improve the performance of applications by orders of magnitude over traditional off-the-shelf processors.

Key Benefits of the Convey System Architecture

- *Breaks the current power/ performance wall*
- *Significantly reduces support, power, and facilities costs*
- *Lowers system management costs by using industry-standard, Linux-based system management tools*
- *Reduces application porting and development efforts for high-performance applications*

Hybrid-Core Architecture

Wolverine accelerators support the Convey Hybrid-Core Architecture, which tightly integrates coprocessor resources in a Linux-based x86 server. The accelerators support virtual-to-physical addressing, allowing the coprocessor and its onboard memory to be mapped in a process' virtual address space. The coprocessor has full access to memory residing on the host platform, and the host processors have full access to memory on the coprocessor via load-store operations. Memory is fully protected by the virtual memory subsystem, ensuring that a "rogue" application running on the coprocessor cannot corrupt system memory.

The coprocessor is dynamically reconfigurable, and can be reloaded with application specific instruction sets called personalities. Personalities are a hardware implementation of performance-critical regions of code that need acceleration beyond what can be provided by a series of general-purpose instructions on a classic x86 system. An application executable contains both x86 instructions and references to custom hardware instructions; both of these execute in a common, coherent address space (Figure 2).

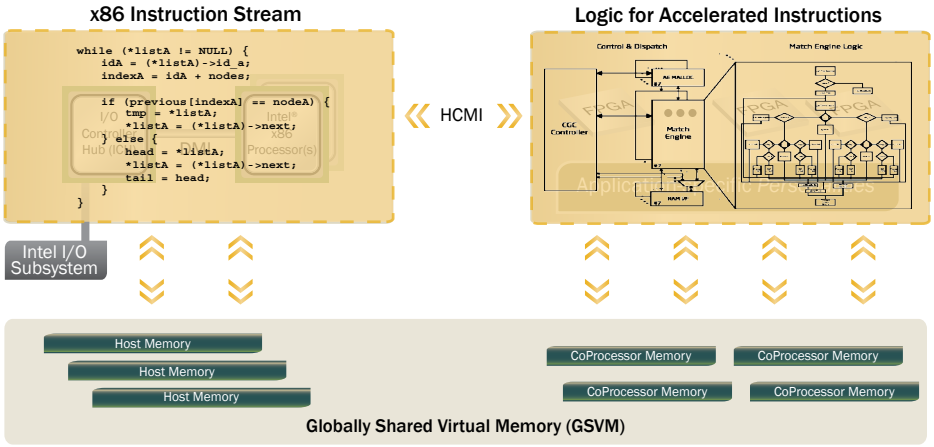


Figure 2. Hybrid-core computing as it appears to an application.

Convey Wolverine application accelerators are PCIe Express form factor coprocessors that provide hardware acceleration of key algorithms.

Personalities provide acceleration via multiple mechanisms, including pipelining, multithreading, and replication. Personalities contain one or more pipelined functions, each of which may replace many instructions on the host processor. These functions are then replicated to execute in parallel, providing a much higher degree of instruction level and data parallelism than is possible on a conventional commodity processor.

For example, Convey's memcached personality accelerates a widely used application that implements a key-value store as a network service. The personality consists of compute units that process incoming requests by parsing the commands and hashing the keys contained within them—thus offloading the most compute intensive portion of the application. Each of these units performs operations that could take tens of thousands of

instructions on an x86 core. The personality contains 16 of these units, processing multiple streams of requests in parallel.

Coprocessor Hardware

Physically the Convey Wolverine Application Accelerator occupies a full-length, double-height PCIe slot. Logically, it is a true coprocessor, and shares the virtual address space of the host through Globally Shared Virtual Memory. GSVM greatly reduces the burden of developers, and removes the programming complexities of treating the PCIe card as an I/O device.

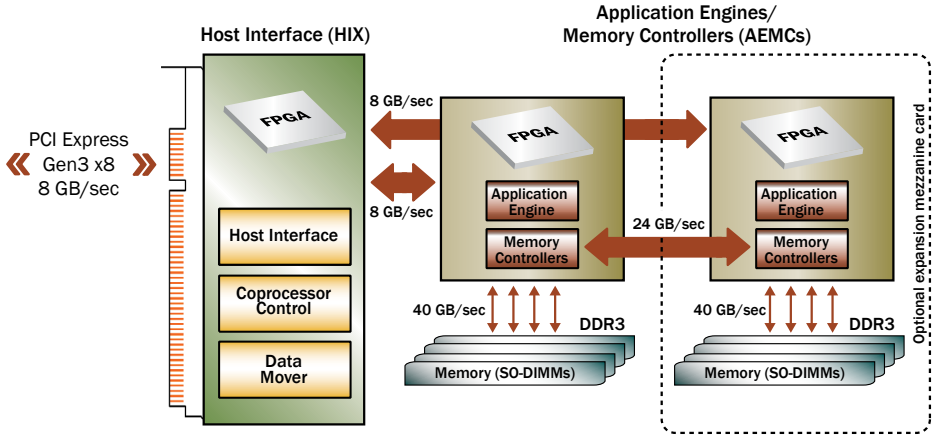


Figure 3. The Wolverine Application Accelerator.

The Wolverine coprocessor is available in multiple variants with FPGAs of different densities, and multiple power and cooling solutions. The coprocessor choice is determined by the logic and memory requirements of the application, and the power requirements of the personalities executed.

Host Interface

The HIX is the host interface and control hub for the coprocessor. It implements the PCIe interface to the host system, handles data movement between the coprocessor and host, and controls coprocessor execution. HIX logic is loaded at system boot time and is considered part of the fixed coprocessor infrastructure.

The Application Engines (AEs) are the heart of the coprocessor. The Wolverine Application Accelerator uses the latest Xilinx® Virtex-7® FPGAs

The HIX implements a PCIe 3.0 x8 electrical connection via a x16 physical slot. At boot time the coprocessor requests Memory Mapped IO High (MMIOH) to map the onboard coprocessor memory into the host's physical address space. This allows host processors to access coprocessor memory directly. Simililarly, the HIX routes requests from processing elements in the AEMC to host memory as required. In addition to ordinary loads and stores, a data mover is incorporated into the HIX to initialize coprocessor memory or perform block copies between the host and coprocessor. The data mover can be used by the operating system or user code, and transfers data at up to 3.7GB/sec.

The HIX receives and processes dispatches from the host and controls execution of the personality in the AEMC. It routes commands and data to the AEMC via 8 high speed serial links running at 8GT/sec (8GB/sec total bandwidth).

Application Engine/Memory Controller (AEMC)

The Application Engine is the heart of the coprocessor and is dynamically loaded with the personalities that deliver accelerated application performance. Convey-provided logic blocks implement the dispatch interface to the HIX, Translation Lookaside Buffers for address translation, a memory crossbar, and DDR3 interfaces for the onboard coprocessor memory. These are then incorporated with application specific kernels to implement a loadable personality.

The AEMC is loaded at boot time with a 'default' personality used to initialize the coprocessor and verify operation. When a user runs an application that takes advantage of the coprocessor, the coprocessor device driver will automatically load the personality required for that application. If the next application run requires a different personality, the driver will automatically swap the personalities. This allows the coprocessor to support a unique processing architecture for each application, while retaining the cost benefits of a common hardware platform.

Memory

Wolverine cards can be configured as "memory free" accelerators with no local storage, or with local on-board memory provided by four error-correcting SO-DIMMs. Possible on-board memory configurations are 16GB, 32GB, or 64GB . Each SO-DIMM is connected to the AEMC via its own 1333MHz DDR3 channel, supporting an aggregate bandwidth of 40GB/sec.

Coprocessor memory is mapped in the host's physical address space, supporting access from the host processors via load/store instructions. Memory is mapped into a processes' virtual address space by the device driver and accessed as normal memory.

Mezzanine Interface

Some Wolverine coprocessor configurations include a mezzanine card in addition to the main coprocessor assembly. The mezzanine card has its own set of 8 high speed serial links to the HIX for instructions and data, as well 24 links to the AEMC on the main card for memory access. The mezzanine card can be used to provide an additional AEMC and set of 4 SO-DIMMs, doubling the logic, memory capacity, and bandwidth of the coprocessor. The connection can also support specialized mezzanine cards that provide I/O connections or additional storage.

Application Development

Applications are accelerated on a Convey system by offloading compute or data intensive kernels to the coprocessor. Applications therefore consist of one or more threads executing on the host, and a hardware "personality" that implements the accelerated kernels on the coprocessor. Unlike a conventional processor that executes a series of general-purpose instructions, the hardware logic in a coprocessor personality can be tailored to the particular requirements of the algorithm, achieving much higher levels of parallelism and performance.

Custom personalities allow users to create their own application-specific instructions to accelerate performance-critical applications.

Personality development involves analyzing the algorithms and applications that are to be accelerated, determining which portions of the application are suitable for implementation on the coprocessor FPGAs, and creating the hardware description language (HDL) that will ultimately execute on the coprocessor.

Developing Personalities

In general, hybrid-core computing is designed to increase application performance through parallelism. But there are a variety of ways to express and implement parallelism, and the blank slate presented by an FPGA can be used to implement virtually any of them. Convey's approach is to provide an open architecture that supports multiple development toolchains optimized for different programming models.

Users with hardware design experience, or with existing hardware IP, can implement code using a hardware description language such as VHDL or Verilog. The designer has complete control over the design, and must define what happens in each module on each clock cycle and how each module communicates to other modules.

An alternative is to use a higher-level toolset that translates some definition of the application at a higher level of abstraction into gate-level definitions in an HDL. Third party toolchains are available for the Convey architecture that are based on standard languages such as C, accelerator extensions such as OpenCL, or other high level languages designed to describe highly concurrent algorithms.

Finally, Convey offers the Personality Development Toolset, which is based on a hardware threading model called Hybrid Threading (HT). This toolset allows the programmer to design logic using a threading model similar to software threading, maximizing performance while maintaining developer productivity.

The system infrastructure supporting all of these toolchains is the Personality Development Kit, or PDK. The PDK is analogous to the runtime libraries that provide support functions and access to I/O and storage in a conventional program. On the coprocessor the PDK provides logic libraries that implement the dispatch interface to the HIX, perform virtual-to-physical address translation, and manage on-board coprocessor memory. These components are then linked with the programmer's application specific kernel to create a personality that can be loaded and executed by the coprocessor.

The Convey Hybrid Threading (HT) toolset allows the programmer to design logic using a threading model similar to software threading, maximizing performance while maintaining developer productivity.

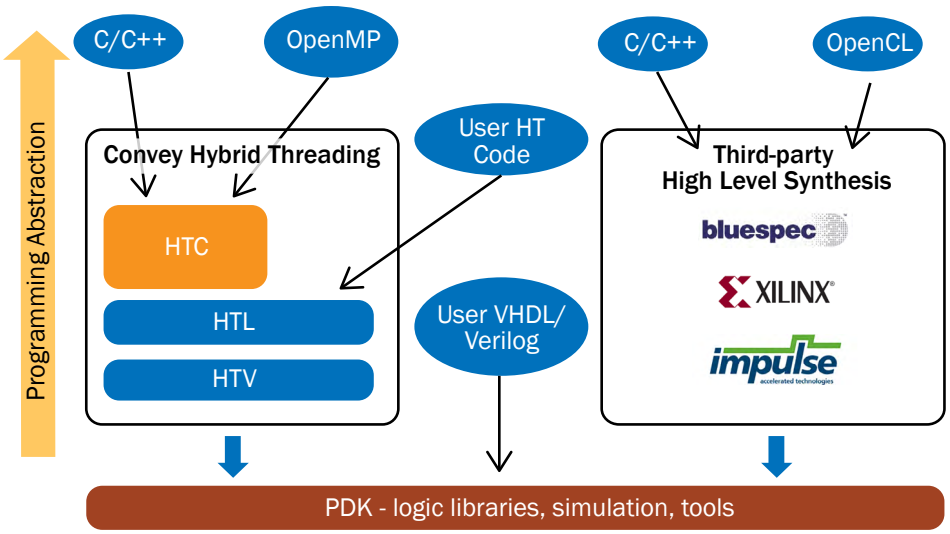


Figure 4. Multiple personality development options.

Personality Development Kit

The PDK includes logic blocks that implement the interfaces between the AEMCs and the other components of the coprocessor (Figure 5), tools to package bitfiles produced by the Xilinx FPGA development tools into a personality, a simulator for debugging, and system and compiler APIs to allow execution of user-defined instructions.

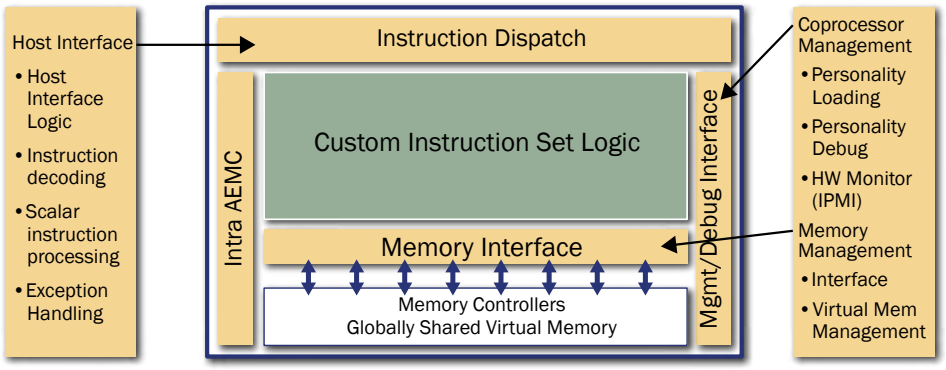


Figure 5. PDK pre-packaged logic blocks reduce development time.

The PDK is the foundation for the other components of the toolset, including Hybrid Threading tools described in the next section.

Hybrid Threading Toolset

The Convey Hybrid Threading Toolset (HT) has taken a different approach to programming the FPGAs, based on a paradigm of parallel hardware threads running on the coprocessor. It is implemented as a combination of higher level programming constructs, based on C++, and a runtime library that supports communication between the host CPU and the processing elements in the FPGA. The toolset manages the details of scheduling threads on multiple hardware units, leaving the programmer free to concentrate on the kernel of his algorithm. Its design:

- Decreases the time it takes to port an application to the Convey architecture. Productivity is commonly cited as the primary roadblock to FPGAs being adopted for mainstream High Performance Computing. The HT tools are designed to decrease the time required to implement accelerated kernels on Convey coprocessors.
- Increases the number of people who can develop custom personalities: While some hardware knowledge in the development team is still required, the HT tools make it easier for a non-expert to be productive on the Convey system.
- Enables maximum kernel performance: The HT tools were designed with a bottom-up approach, starting with the low-level hardware in the FPGA and layering functionality on top to abstract away the details. The programming interface is greatly simplified, but the user still maintains the flexibility required to achieve the maximum performance from the FPGA using features such as pipelining and module replication.
- Focuses on overall application performance: Kernel performance alone doesn't matter to the end user. What matters is how much the application performance is improved. Maximizing application performance means efficiently partitioning the problem to use the best tool for the job (host or coprocessor), and making the best use of each resource (i.e. multithreading on the host processor, overlapping processing and data movement, etc.).

The Convey Hybrid Threading Toolset (HT) manages the details of scheduling threads on multiple hardware units, leaving the programmer free to concentrate on the kernel of his algorithm.

Hybrid Threading Architecture

At the heart of the HT architecture is the HT hardware thread. As a function is ported to the coprocessor, the programmer defines the number of threads available to execute that particular function. Each thread executes on the same hardware such that on each clock cycle, one thread is executing one instruction (which may be many lines of C++ code).

The HT architecture is designed to provide familiar and easy to use constructs to the programmer, while still maximizing the performance available from the reprogrammable coprocessor hardware. It presents a hardware framework that includes the following main components:

HT threads are hardware threads that process streams of data. As a function is ported to the coprocessor, the programmer defines the number of threads available to execute that particular function. The threads execute within the context of the process executing on the host, and access data on the host or coprocessor using virtual addresses. HT threads are analogous to software threads running on the x86 host, and communicate with each other and with host threads through memory.

Processing Units execute HT threads. Processing units are made up of one or more modules, and represent the instantiation in hardware of a set of functions or a kernel from the application. Processing units are analogous to an x86 core, and can execute multiple HT threads. Unlike a general purpose core, however, processing units are tailored to specific functionality, and contain only the logic required for that function. The HT support infrastructure schedules the threads for execution so that the programmer is free of the details (i.e. spawn a thread if available, otherwise retry). The ability to pause and resume a thread prevents the thread from polling for status and wasting compute cycles.

Modules implement specific functions within a processing unit. They are analogous to a functional unit such as an ALU in an x86 core, except they typically implement the equivalent of an entire software subroutine or function. The HT framework allows the

programmer to control the number of copies of each module instantiated within a processing unit, allowing greater parallelism for heavily used functions.

HT instructions are the basic unit of execution within the HT architecture. They are analogous to x86 instructions, except they are defined by the user in C or C++ and can implement many lines of source code as a single instruction.

The architecture presents a model that should be familiar to programmers—multiple threads executing instructions within a shared address space. The instructions, however, can be much richer than the fixed instructions on an x86. Moreover, the programmer is given control over the parallelism implemented at different levels, allowing fine-grained tuning of the implementation to match a particular application or workload.

Hybrid Threading Toolset

The Hybrid-Threading toolset is made up of two main components: the HT host API, and the HT programming language for programming the coprocessor. The programmer starts by profiling an application to determine which functions in the call graph are suitable to be moved to the coprocessor. In the example described in Figure 6, the main function and function fn5 remain on the host, while functions f1, fn2, fn3 and fn4 are moved to the coprocessor. Each of these functions is effectively programmed as a single thread of execution using the HT programming language, and the HT tools automatically thread each function to achieve desired parallelism.

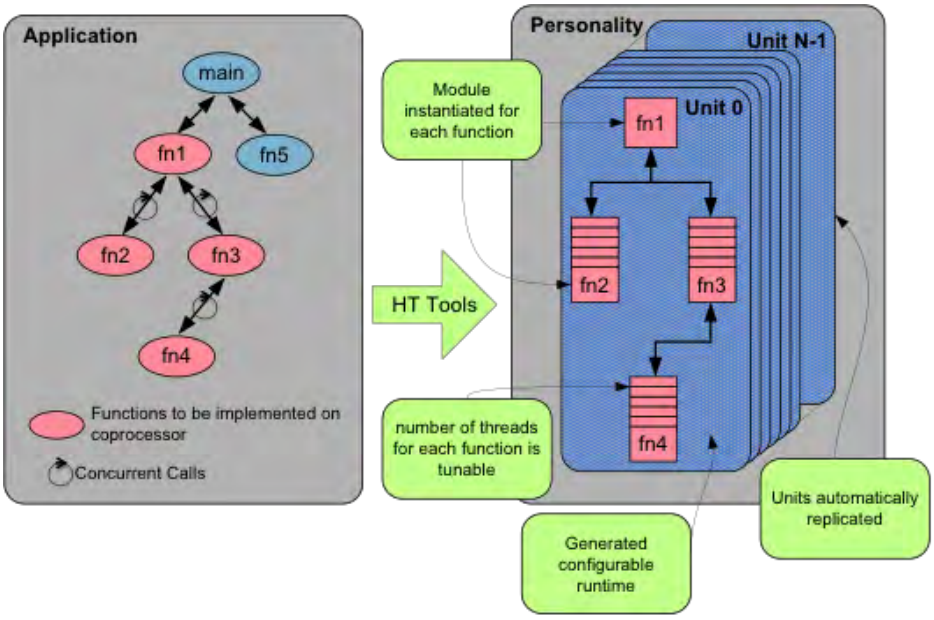


Figure 6. Hybrid Threading call graph example illustrating the HT architecture

HT compute units are programmed using the HT programming language, which is a subset of C++ and includes a configurable runtime that provides commonly used functions. The user writes C++ source for each module, as well as a hybrid threading description file (HTD) which describes how the modules are called and connected, and which provides declarations for module instructions and variables.

Because the programmer explicitly defines the interconnections and capabilities for the module, the HT tools generate only the infrastructure needed by the module, and therefore avoid wasting space on unneeded logic and ultimately increasing performance.

Debugging and Profiling

The majority of debugging of an HT design can be done in software simulation, which enables the programmer to use familiar software debugging methods such as printf and GDB. In addition to these standard debugging tools, HT provides an HtAssert() function.

In simulation, this acts as a normal assert() call, which aborts the program if the expression fails. But when the Verilog is generated for the personality, logic is also generated for the assertion such that it also exists in the actual hardware design. If an assertion fails, a message with the instruction source file (*_src.cpp) and line number is sent to the host and printed to standard error. Finding a problem at the source due to an assertion is significantly easier than trying to trace back from a symptom such as a wrong answer, and because the HtAssert() call is built in to the HT infrastructure, no special debugging tools are required to use it. The user can choose to globally enable or disable asserts as the design moves from the functional verification phase to production use.

HT also provides a profiler to provide instruction and memory tracing, program cycle counts, and other information that is useful in optimizing performance of a design. This helps the programmer to explore architectural changes and identify performance bottlenecks before running on the actual hardware.

SUMMARY

Convey Wolverine application accelerators leverage hybrid-core computing to multiply the performance of x86 servers on compute- and data- intensive applications. The PCIe form factor cards can be installed into standard servers, and the resulting platform looks like a standard UNIX-based node to the rest of the nodes in a cluster.

The hybrid-core accelerated nodes provide the benefits of increased performance within a single node: increased scalability, reduced power, cooling, and floor space, and reduced total cost of ownership. Wolverine is available in multiple configurations to match the appropriate hardware to the application performance requirements.

Convey Wolverine benefits include:

- Ease of integration—Standard PCIe interface for compatibility and high-bandwidth data transfers; familiar Linux environment.
- Ease of development—With Globally Shared Virtual Memory, commodity host x86 processors and the on-board FPGAs share the same virtual address space, simplifying software and hardware development efforts. The Convey Personality Development Toolset facilitates development of applications that leverage the ease of programming industry-standard processors with the performance of FPGAs.
- Maximum Performance—Powerful, state-of-the art Xilinx Virtex-7 FPGAs, with up to 1.9M logic cells and 46 Mbit of block RAM (3.9M logic cells and 93 Mbit of block RAM with mezzanine card), provide logic resources for compute kernels. Four 1333MHz SO-DIMMs (with ECC support) providing up to 64GB of memory and 40 GB/sec of bandwidth (128GB and 80GB/sec with mezzanine card) provide capacity and performance for real applications.

Convey Computer Corporation
1302 E. Collins Boulevard
Richardson, Texas 75081
Phone: 214.666.6024 Fax: 214.576.9848
Toll Free: 866.338.1768
www.conveycomputer.com