THE WORLD’S FIRST HYBRID-CORE COMPUTER.
CONVEY HYBRID-CORE COMPUTING
Hybrid-core Computing

Performance of application-specific hardware

Convey HC-1

Programmability and deployment ease of an x86 server

Heterogenous solutions
- can be much more efficient
- still hard to program

Multicore solutions
- don’t always scale well
- parallel programming is hard

Application Performance/Power efficiency

High

Low

Difficult

Ease of Deployment

Easy

1/22/2010
Hybrid-Core Computing

Applications

Convey Compilers

x86-64 ISA

Custom ISA

Shared Virtual Memory

Application-Specific Personalities

• Extend the x86 instruction set
• Implement key operations in hardware

Cache-coherent, shared memory

• Both ISAs address common memory

Life Sciences

CAE

Custom

Financial

Oil & Gas

*ISA: Instruction Set Architecture
HC-1 Hardware

- Intel Xeon
- PCI I/O
- 8 GB/s
- FPGA
- 80 GB/s
- Memory
- Cache Coherent, Shared Virtual Memory
Using Personalities

- Personalities are reloadable instruction sets
- Compiler generates x86 and coprocessor instructions from ANSI standard C/C++ & Fortran
- Executable can run on x86 nodes or Convey Hybrid-Core nodes
HC-1 Architecture

“Commodity” Intel Server

- Intel 5138 Dual Core Processor
- Intel 5400 MCH
- Intel IO Subsystem
- Memory

Intel x86-64 Server
x86-64 Linux

Convey FPGA-based coprocessor

- Application Engine Hub
- Application Engines
- Memory
- FPGA based
- Shared cache-coherent memory

Direct Data Port
HC-1 Hardware

- **2U enclosure:**
  - Top half of 2U platform contains the coprocessor
  - Bottom half contains Intel motherboard
HC-1 Physical Layout

- Direct Data Port
- Memory DIMMs
- Memory Controller
- AEs
- Memory DIMMs

Dimensions:
- Width: 17.78" (11.15"

Date: 1/22/2010
Inside the Coprocessor

Host interface and memory controllers implemented by coprocessor infrastructure

Implemented with Xilinx V5LX110 FPGAs

Implemented with Xilinx V5LX155 FPGAs

16 DDR2 memory channels

Standard or Scatter-Gather DIMMs

80GB/sec throughput
Coprocesor Block Diagram

scalar instructions executed by IAS, AE instructions broadcast to AE0, AE1, AE2, AE3

MCs translate virtual addresses, maintain coherence with host

datamover for high bandwidth block moves

AE-MC links support 8192 TIDs, 76.8GB/s
Convey SG-DIMMs optimized for 64-bit memory access

- High bandwidth for non-unity strides, scatter/gather
- 3131 interleave for high bandwidth for all strides except multiples of 31

- measured with strid3.c

```c
for( i=0; i<n*incx; i+=incx ) {
   yy[i] += t*xx[i];
}
```

Strided 64-bit Memory Accesses

Nehalem (single core)

HC-1 SG-DIMM 3131
Direct Data Port

- Direct data paths to I/O gates on FPGAs
- Permits custom H/W interface

For more information see “Virtex-5 FPGA User Guide UG190 (v5.2)” November 5, 2009
Convey Software Architecture

- Convey Enhanced Linux Kernel
  - Key routines (BLAS, FFTs) optimized for Convey personalities
  - Objects can be linked with other gcc/glibc compatible objects
  - If coprocessor shared library not present, native x86 code is executed

- Convey Math Library (CML)

- Personality Support System
  - Convey Compilers
  - GCC & compatible compilers

- Intel x86-64 ISA
- Coprocessor ISA

Kernel is Linux Standards-based (LSB) compliant, legacy apps run "as-is".

Personalities define instruction set.
Convey Compilers

- Program in ANSI standard C/C++ and Fortran
- Unified compiler generates x86 & coprocessor instructions
- Seamless debugging environment for Intel & coprocessor code
- Executable can run on x86_64 nodes or on Convey Hybrid-Core nodes
Coprocessor Regions

• The compiler dispatches blocks of code called “coprocessor regions.”

• Switches, directives and pragmas can be used to control what code is compiled for the coprocessor
  – loop or code segment within a routine or a whole routine
  – can also use underlying library interface (“copcall”)

• Coprocessor regions cannot call x86 routines
Memory Placement

- Convey systems have a Non-Uniform Memory Access (NUMA) architecture
  - Host and coprocessor can access all of memory
  - but it’s much more efficient if they access local memory
- OS manages host and coprocessor memory as separate memory pools
- Compiler flags and directives can specify placement
  - #pragma cny coproc_mem (x,y) places static arrays or common blocks in coproc memory
  - cny_cp_malloc() alternative to malloc()
- Memory can also be migrated or copied dynamically
  - “#pragma cny migrate_coproc (X,n)” specifies that array X should be moved to coprocessor memory before execution of the next line of code
  - cny_cp_memcopy() copies buffers using the datamover
Convey Runtime Environment

If `dlopen` of shared library fails, Intel x86-64 code executed

**executable**
- Intel® x86-64 code
- Coprocessor code

**gdb**
Debugging on HW & simulator

**Convey Shared Libraries**

**Convey x86 Simulator**
- shared library executes on x86

**coprocessor hardware**

**SPAT**
- performance simulator

**personalities are demand loaded by OS at runtime**

Custom
- FAP
- SP
A personality is...

- A reloadable instruction set that augments the x86 ISA
  - Applicable to a class of applications or specific to a particular code
- Each personality is a set of files consisting of:
  - A unique ID
  - The bits loaded into the AE FPGAs
  - Information used by the Convey compilers and tools
    - List of available instructions
    - Performance characteristics for compiler optimization
    - Machine state formatting and modification

```
$ ls /opt/convey/personalities/2.1.1.1.0
ae_fpga.tgz
convey-aesp-2.0.0-10_03_17_62.tgz
lic_info
PersDesc.dat
readme
rest.o
save.o
zero.o
```
How Are Personalities Created?

• **Personalities are a logic design**
  – implemented using a hardware description language (high level tools may be used as well)
  – synthesized with the Xilinx tools
  – interfaced to the coprocessor dispatch, memory, and management infrastructure
  – packaged for demand loading by the Convey OS

• **Convey sells prebuilt personalities for key algorithms and applications**

• **Convey licenses a Personality Development Kit that enables customers to construct their own**
Instruction Based and Algorithmic Personalities

• **Instruction based personalities implement instruction sets**
  – Automatic vectorization generates SIMD instructions from standard languages
  – Different sequences of instructions can implement different algorithms

• **Algorithmic personalities implement complete state machines**
  – implements just one algorithm
  – may be specific to a particular application

```c
parameter (N=100000)
real*4 a(N), b(N), c(N), s
c$cny BEGIN_COPROC
  do i=1,N
    c(i) = a(i) + s*b(i)
  enddo
  c$cny END_COPROC
  end
$cnyf95 -O3 -mcny_vector test.f
VECTOR LOOP in MAIN__0 at 4: L 2 S 1 B 2 U 0 I 0 M 0
```

```c
ret= l_copcall_fmt(sig, solver, a1, size);
```

```c
solver:
  mov %a8, $0, %aeg
  mov %a9, $1, %aeg
  caep00 $0
  rtn
```
Instruction Based Personality Example (FAP)

Vector architecture with optimizations for financial Monte Carlo simulation

- Double precision floating point units
- Functional units for common functions such as log, exp, random number generation
- Supported by the compiler as vector intrinsics

32 Function Pipes across 4 Application Engines
vector elements distributed across function pipes
Hardware implementation of search kernel from InsPect proteomics application

1-of-40 State Machines

- Entire “Best Match” routine implemented as state machine
- Multiple state machines for data parallelism
- Operates on main memory using virtual addresses
Calling the Inspect Personality

cny_get_signature(mckxstr(UCSD_SIGNATURE), &cny_pdk_image, &cny_pdk_image2, &Result);

...
ptr = (char *)cny_cp_malloc(msize);

...
copcall_nowait_stat_fmt(cny_pdk_image, (void *) &pdk_kernel_wrapper, &copcall_hndl, &copcall_status, "AAAAaaaaaA",
max_items_per_pipe_array, num_todo_per_pipe_array, num_done_per_pipe_array, cny_input_per_pipe_array,
GlobalOptions->InitialSTB,
GlobalOptions->InitialTier1,
GlobalOptions->StoreThreshold,
GlobalOptions->StallThreshold,
GlobalOptions->DropLastPosition,
&PeptideMass[(int)'A']);
UCSD InsPect Performance

• **Proteomics application**
  - compares mass spectrometry samples against a database of proteins

• “streaming” personality implements entire search kernel
  - x86 cores place data on queues for processing by the coprocessor
  - multiple independent function pipes process samples

![Relative Performance Graph]

- 3GHz Xeon 1-core
- 3GHz Xeon 4-core
- 3GHz Xeon 8-core
- Convey

1/22/2010
Silicon Vox Speech Recognition Personality
Continuous Automated Speech Recognizer (ASR)

Scoring/Backend Search Pipelines on AEs

- Backend search dominated by bit-level comparisons and random memory accesses
- Multiple pipelines for data parallelism

*First commercial implementation of a hybrid architecture for Speech Recognition*

*Several patents pending for ASR on a hybrid platform*
Custom Personalities

• Personality Development Kit
  – logic libraries implement interfaces to coprocessor infrastructure
  – System simulation environment for debugging
  – Management tools package bit files produced with Xilinx toolset into personalities

• Architected instruction interface
  – transfer data to/from AE
  – control custom AE logic

• Compiler interfaces to generate calls to custom personality instructions
Personality Development Kit (PDK)

- Customer designed logic in Convey infrastructure
- Executes as instructions within an x86-64 address space
- Allows designers to concentrate on Intellectual Property, not housekeeping
CONCLUSION
Energy Efficient, Hybrid Core Computing

- **Higher Performance**
  - 5x to 25x application gains

- **Energy Saving**
  - Up to 90% reduction in data center power usage

- **Easy to program**
  - ANSI standard C, C++, and Fortran

- **Reloadable Personalities**
  - Application specific performance on an x86 base

---

"Convey Computers may be at the forefront of a wave of innovation brought on by developing FPGAs as a viable alternative to CPUs…"

"Convey Computer seeks to use FPGAs to create a hybrid computing platform"

MIS Impact Report, 12/09/08

451 Group

"We have found that one rack of HC-1 servers will replace eight racks of other servers...with correspondingly lowered energy requirements”

Pavel Pevzner - UCSD