RAMP Gold Hardware and Software Architecture
Rimas Avizienis, Yunsup Lee, Zhangxi Tan, Andrew Waterman, David Patterson, Krste Asanovic

RAMP Gold: A Manycore Emulator
- Leverage RAMP FPGA emulation infrastructure to build prototypes of proposed architectural features
- Fast enough to run real apps
- “tapeout” everyday
- RAMP Gold
  - Single socket tiled manycore target
  - Shared memory, distributed coherent cache
  - Multiple on-chip networks and memory channels
  - Split functional/timing model, both in hardware
  - Host multithreading of both functional and timing models

Host multithreading
- Single hardware multiprocessing with multiple copies of CPU state
  - Fine-grained multithreading, time multiplexing with reasonable FPGA resource consumption
  - Hide emulation latencies
  - Not multithreading target

RAMP Gold on a FPGA
- Single FPGA Implementation
  - Low cost Xilinx ML505 board
  - 64-128 cores, 2GB DDR2, FP, timing model, 100-130 MIPS
  - A 64-core functional model demo

RAMP Gold: Verification Flow
- Frontend Test Server
- FPGA Target: Xilinx ML505, BE3
- C Functional Simulator
- Frontend Core

RAMP Gold: GCC Toolchain
- SPARC cross compiler with newlib
  - Built with (binutils-2.28, gcc-4.3.2/gmp-4.3.2/mpfr-2.3.2, newlib-1.16.0)
  - sparc-elf-gcc, ld, m, objdump, ranlib, strip, ...
- Link newlib statically
  - newlib is a C library intended for use on embedded systems
  - C functions in newlib are narrowed down to 39 system calls
    - exit, close, environ, execute, fork, fstat, getppid, isatty, kill, link, ls, open, read, sbir, stat, times, unlink, wait, write
  - Now we can compile our C, C++ source code (w/ standard C functions) to a SPARC executable
    - sparc-elf-gcc -o hello hello.c -lc - lys -mcps=8

RAMP Gold: C-Gold Functional Simulator
- Baseline functional model to verify our functional model written in system verilog
  - Full 32-bit SPARC v8
  - Includes an IEEE 754 compatible FPU
  - New instruction introduced to support active messages
    - SENDAM
- Written from scratch, no junk in it
  - Very fast, ~50 MIPS on Clovertown
  - Easy to understand
  - Extensible
  - Fully parametrized (Number of target threads, host threads, ...)

RAMP Gold: Frontend Machine
- Multiple backends
  - C-gold functional simulator (link: function calls)
  - Modelsim simulator (link: DPI)
  - Actual H/W (Xilinx ML505, BE3) (link: gigabit ethernet)
- Narrow interface to support a new backend
  - Host/Target interface
    - CPU reset
    - Memory interface
      - [read, write]_[signed, unsigned]_[8,16,32,64]
- Execute system calls received from the backend
  - Signaled by the backend proxy kernel
  - Map a Solaris system call to a Linux system call and execute

RAMP Gold: Proxy Kernel
- How could we support I/Os?
  - The target doesn’t have any peripherals (e.g. disks)
  - It would be a pain to program a system which can’t read or write anything...
  - It would be more pain to make the peripherals work with the actual HW
- A minimal kernel which acts as a proxy for system calls invoked by newlib
  - Proxy kernel sends the arguments and the system call number to the frontend machine
  - The frontend machine does the actual system call and returns the results back to the proxy kernel
  - Finally the PC is moved back to the application and everybody is happy