Full-System Architectural Exploration Sandbox

Eriko Nurvitadhi and James C. Hoe
Computer Architecture Lab (CALCM)
Carnegie Mellon University

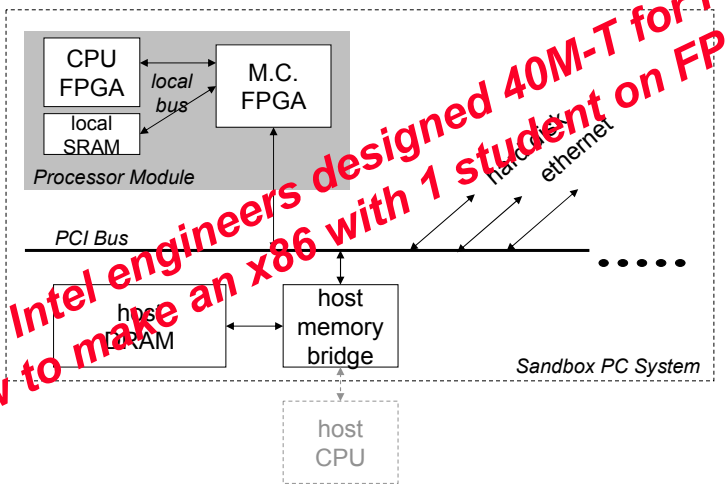
Sandbox Desiderata

- A PC where I can make small but unrestricted changes to the CPU and/or memory architecture
  - add/change instructions
  - add/change mechanisms

  This is not for microarchitecture/ILP studies

- Requirements
  - real operating system, applications, I/O, and networking
  - a few 100 MIPS
  - reasonable one-time investment to build infrastructure
  - reasonable recurring cost to use the Sandbox in a study

Sandbox Platform



500 Intel engineers designed 40M-T for P4.
How to make an x86 with 1 student on FPGA?

We can do it because …

- We don't need a P4
  - detailed microarchitecture and ILP are second-order
  - DRAM and I/O devices are relatively faster

  Pentium or simpler will suffice for IPC=1 at 100MHz
- We don't need "full" x86 compliance
  - use C++ code from Bochs as ISA spec
  - use Bochs simulation for reference behavior

  (http://bochs.sourceforge.net)

  Bochs-compliance is good enough to boot Linux and Windows
- We can use high-level HDL and FPGA
  - emphasis on correctness rather than optimality
  - incremental development

Preliminary Experience

- x86 CISC instruction decoding
  - as little as 1 byte, as many as 16 bytes per instruction
  - yields 947 distinct behaviors (according to Bochs)

    only 558 in pre-MMX x86
  - a major source of complexity in real x86 implementations

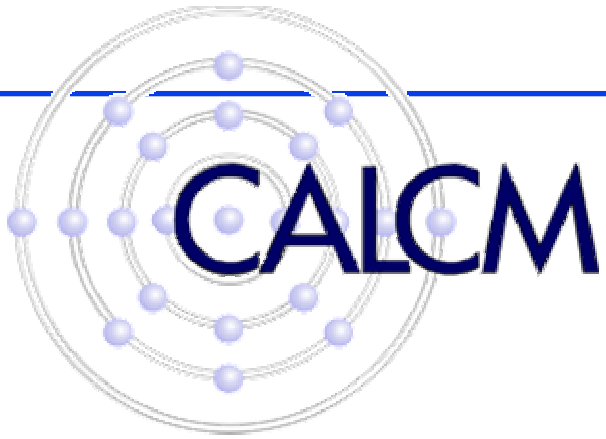
    the real reason there is a trace cache in Intel P4
- Bochs C++ to VHDL
  - naïve, straightforward translation of C++ code to combinational logic in VHDL
  - synthesize to Xilinx XC2V6000-6
    - 5% resource
    - 50MHz

Bochs Example: *BX_CPU_C::ADD_EbGb*

```
BX_CPU_C::ADD_EbGb(bxInstruction_c *i) {
  Bit8u op2, op1, sum;

  op2 = BX_READ_8BIT_REGx(i->nnn(),i->extend8bitL());

  if (i->modC0()) {
    op1 = BX_READ_8BIT_REGx(i->rm(),i->extend8bitL());
    sum = op1 + op2;
    BX_WRITE_8BIT_REGx(i->rm(), i->extend8bitL(), sum);
  } else {
    read_RMW_virtual_byte(i->seg(), RMAddr(i), &op1);
    sum = op1 + op2;
    Write_RMW_virtual_byte(sum);
  }

  SET_FLAGS_OSZAPC_8(op1, op2, sum, BX_INSTR_ADD8);
}
```

To Conclude

- A powerful tool for architectural studies
  - complete control of CPU and memory controller
  - run real operating system and applications
- I am not saying it is easy, but it definitely is doable
  - Bochs as specification and reference behavior
  - incremental development in FPGA
- Why not just use SIMICS, Bochs, etc?
  - 10 MIPS out of the box, but performance degrades very quickly when you tack on new behavior or detail
- Why not use Pentium RTL models?
  - real RTL models are hard to come by
  - real RTL models are impossible to modify



TRUSS

Computer Architecture Lab (CALCM)

Carnegie Mellon University

http://www.ece.cmu.edu/CALCM