Hardware / Software co-designed processors

Kyriakos Stavrou

Feb 11th, 2010

What is this presentation about

- CPU design today faces important challenges
  - Power, performance, complexity, validation, cost, …
- This presentation is about a radical approach

Co-designed CPUs
CPU = HW + SW

Agenda

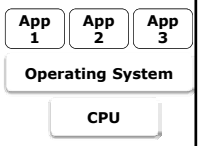
- Overview of the technology
  - What are the HW/SW co-designed processors
- Key Ideas and Advantages
  - From the HW-only CPU to the co-design paradigm
- Research Projects / Market examples
  - Academic Research
  - Products
- Potential and Open Issues
  - A glance to the huge potentials

HW/SW co-designed processors

- Traditionally CPUs have been HW only

- Alternative: co-designed processors
  - CPU = HW + SW

  - The system is **unaware** of the **CPU internals**



HW/SW co-designed processors

- HW/SW co-design targets the major challenges
  - **Power Consumption**
    - Through using **simpler** Hardware
    - Through using **less** Hardware
    - By optimizing code once / using many times
  - **Design Complexity**
    - Co-designed processors have **simpler** HW
    - Simpler HW is much easier to **validate**
  - **Performance**
    - **Synergy** between the HW and SW
    - Exploit **dynamic information**

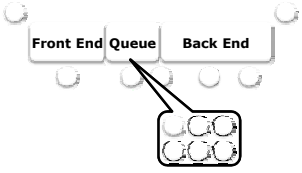
Agenda

- Overview of the technology
  - What are the HW/SW co-designed processors
- Key Ideas and Advantages
  - From the HW-only CPU to the co-design paradigm
- Research Projects / Market examples
  - Academic Research
  - Products
- Potential and Open Issues
  - A glance to the huge improvement potential

The key idea

- How traditional processors work

1. Fetch the next instruction
2. Decode the instruction
3. Store instruction in queue
4. Issue for execution
5. Execute (out-of-order)
6. Reconstruct program order
7. Commit

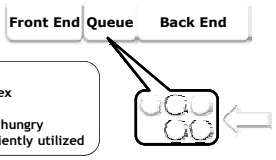


- Instructions in Back-End may need multiple cycles
  - The queue has multiple instructions
  - Key mechanism: Optimize the instructions in the queue

The key idea

- How traditional processors work
  - Fetch the next instruction
  - Decode the instruction
  - Store instruction in queue
  - Apply optimizations
    - Rename instructions
    - Schedule instructions
    - Memory disambiguation
    - Reorder
  - Issue for execution
  - Execute (out-of-order)
  - Reconstruct program order
  - Commit

HW:
- Complex
- Large
- Power hungry
- Inefficiently utilized



These optimizations are:
- Key mechanism of modern CPUs
- Crucial for performance

The key idea

- In a traditional design
  - There must be different HW for different optimizations
    - This HW needs to be designed and validated
  - The cost to optimize the same code is paid multiple times
    - The hardware is very aggressive
    - Consumes a lot of power
  - The hardware tries to optimize always
    - Optimizations do not always pay-off (power / performance)
    - HW exploits limited information (limited #instructions in the queue)
  - Non-efficient resource utilization
    - Area increase
    - Validation cost
    - Power consumption increase
    - Pay the optimization cost multiple times

The key idea

- Golden rule: *"90% of execution time to 10% of the code"*
  i.e. Most execution time is spent in a small portion of the application

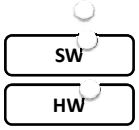
Example: *Matrix Multiply code*

```
read array A;
read array B;

for (i=0 .. n)
  for (j=0 .. n)
    for(k=0 .. n)
      C[i][j]+=A[i][k]*B[k][j]

print array C;
```

These instructions dominate the execution time

The key idea

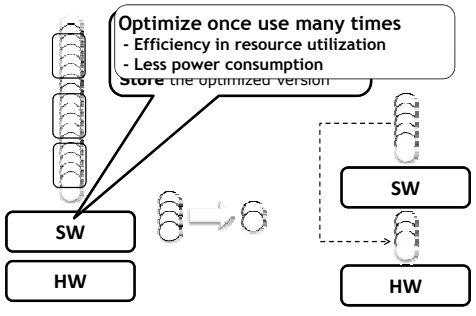
- Co-designed Processors
  - The instruction stream passes through the SW
  - The SW observes the execution (profile)
  - Instructions are sent to the HW for execution
  - By "observing" the instruction stream:
    - The SW "learns" the behavior of the code
    - Identifies hot regions of the code
    - SW optimizes the code once
    - Store the optimized regions
    - HW executes the optimized code many times



The key idea

Optimize once use many times
- Efficiency in resource utilization
- Less power consumption

Store the optimized version



The key idea

- Cost of running the SW:
  - Running the SW comes with some cost
    - "Observing" the dynamic code
    - Optimizing the code
    - Store optimized regions
  - The SW cost is amortized
    - The optimized segments are executed many times
    - Using staged optimization helps significantly:
      - Frequent regions : Few optimizations / low SW cost
      - Hot regions : Further optimization / medium SW cost
      - Critical regions : Maximum optimizations / high SW cost

The key idea

- Comparison between HW-only and co-designed CPUs
  - "Amount" of hardware
    - Traditional approach : More hardware
    - Co-designed processors : Less hardware
  - Complexity of hardware
    - Traditional approach : Very complex (e.g. support for ooo)
    - Co-designed processors : Much simpler
  - Power Consumption
    - Traditional approach : High (optimization, HW complexity)
    - Co-designed processors : Significantly lower
  - Performance
    - Traditional approach : High
    - Co-designed processors : In the same order

The key idea

- The benefits
  - Similar performance (often higher performance)
  - Less Power (extended battery life)
  - Smaller Area (lower cost)
  - Easier to design and validate (lower cost / shorter time-to-market)
- How?
  - Use SW instead of HW for optimizing
  - SW is usually easier to debug than HW
  - Keep the optimized code for future use
  - Efficient resource utilization (optimize once, use many times)

Outline

- Overview of the technology
  - What are the HW/SW co-designed processors
- Key Ideas and Advantages
  - From the HW-only CPU to the co-design paradigm
- Research Projects / Market examples
  - Academic Research
  - Products
- Potential and Open Issues
  - A glance to the huge improvement potential

Research projects / Market examples

- Many researchers identify the value of the approach
- A lot of work in academia
  - Parrot
    - ISCA 2004:
      "Power Awareness through Selective Dynamically Optimized Traces"
  - Targets both performance and power
  - The processor has 2 pipelines
    - **Simple** - lower power for "cold" regions
    - **Aggressive** - higher power for "hot" regions
  - Operation
    - **Instructions** initially go through the **"cold pipeline"**
    - **Hot regions** are **identified** and **optimized**
    - **Optimized regions** are **stored** and **reused**

Research projects / Market examples

- Many researchers identify the value of the approach
- A lot of work in academia
  - rePLay
    - IEEE Transactions on computers 2001
      "rePLay: A Hardware Framework for Dynamic Optimization"
  - Mainly targets higher performance
  - It is a HW only solution but follows the same principles
  - Is equipped with an optimization engine
    - Identify hot regions / optimize / store
    - Includes HW mechanisms to enable more optimizations
    - Uses aggressive HW

Research projects / Market examples

- Market Examples
  - Transmeta™ Corporation (1995 - 2009)
  - Transmeta™ built the first co-design processors
    - Crusoe™ - 2000
    - Efficeon™ - 2004
  - VLIW architectures
    - Host ISA : x86
    - Elaborated Software : Code Morphing Software
    - Simple Hardware which provides special support for the optimizer
  - Main target
    - 100% compatibility
    - Similar performance
    - Lower power

Agenda

- Overview of the technology
  - What are the HW/SW co-designed processors
- Key Ideas and Advantages
  - From the HW-only CPU to the co-design paradigm
- Research Projects / Market examples
  - Academic Research
  - Products
- Potential and Open Issues
  - A glance to the huge improvement potential

Open Issues

- Conventional processors have been evolving for many years
- Co-designed processors is a new paradigm
  - A lot of work is needed for full exploitation
  - This is an amazing topic! Compilers + Computer Architecture + …
- Some important questions
  - Exploit the dynamic information
    - Mechanisms to exploit dynamic information
    - Speculation techniques for performance / power
  - Leverage for multiprocessing
    - Software for efficient execution of parallel code
    - Fault tolerance

Open Issues

- Conventional processors have been evolving for many years
- Co-designed processors is a new paradigm
  - A lot of work is needed for full exploitation
  - This is an amazing topic! Compilers + Computer Architecture + …
- Some important questions
  - Segment Specific designs
    - Can we have CPUs optimized for different market segments?
    - One HW and different SW (maximize for performance / power)
  - Are traditional mechanisms good enough?
    - e.g. pre-fetchers, branch predictors
    - How to take advantage of the simpler circuitry

Market examples / Research projects

- Reading:
  - "The Architecture of Virtual Machines"
    IEEE Computer 2005, James E. Smith, Ravi Nair
    Gives an excellent introduction to the whole technology
  - "Power Awareness through Selective Dynamically Optimized Traces"
    ISCA 2004, Rosner et al.
    Easy to understand and follow overview of where the benefits come from
  - "The Technology Behind Crusoe™ Processors"
    White Paper 2000
    A lot of information of the underlying implementation issues

Conclusions

- Radical improvements usually come from radical solutions
- Do not use HW for everything
  - Consumes power, more complex design, higher cost…
  - Non-efficient resource utilization
- Co-designed processors: CPU=SW+HW
  - Efficient resource utilization
  - Less power, less complexity, lower cost, similar (higher) performance
  - *Huge room for technology innovation*
  - *A huge interest in the research community*

Questions

?