ILP architectures: trading hardware for software complexity

Presented by Pasi Maliniemi
The Big Picture

Computer architectures

CISC = Complex Instruction Set Computer

RISC = Reduced Instruction Set Computer

ILP = Instruction level parallelism

Superscalar

Multiple instructions per cycle

VLIW = Very Long Instruction Word architecture

TTA = Transport Triggered Architecture

Multiple operations in each instruction
General organization of an ILP architecture

FU = Function Unit
Motivation for ILP

• Increasing VLSI densities; decreasing feature size
• Increasing performance requirements
• New application areas, like
  – multi-media (image, audio, video, 3-D)
  – intelligent search and filtering engines
  – neural, fuzzy, genetic computing
• More functionality
• Use of existing Code (Compatibility)
• Low Power
Pipelined Execution of Instructions

**Purpose:**
- Reduce \#gate\_levels in critical path
- Reduce CPI close to one
- More efficient Hardware

**Problems**
- Hazards: pipeline stalls
  - Structural hazards: add more hardware
  - Control hazards, branch penalties: use branch prediction
  - Data hazards: by passing required
VLIW concept

Instruction Memory

Instruction register

Function units:
- Int FU
- Int FU
- Int FU
- LD/ST
- LD/ST
- FP FU
- FP FU

Int Register File

Floating Point Register File

Data Memory
VLIW characteristics

- Multiple operations per instruction
- Large instructions
- One instruction per cycle issued (at most)
- Compiler is in control
- Only RISC like operation support
  - Short cycle times
  - Easier to compile for
- Flexible: Can implement any FU mixture
- Extensible / Scalable
VLIW drawbacks

• Bypass complexity
• Register file complexity
• Register file design restricts FU flexibility
• Operation encoding format restricts FU flexibility
General organization of a TTA

Instruction memory

Instruction fetch unit

Instruction decode unit

Bypassing network

Register file

FU-1

FU-2

FU-3

FU-4

FU-5

CPU

Data memory
TTA characteristics

Hardware

• Modular: Lego play tool generator
• Very flexible and scalable
  – easy inclusion of Special Function Units (SFUs)
• Low complexity
  – 50% reduction on # register ports
  – reduced bypass complexity (no associative matching)
  – up to 80% reduction in bypass connectivity
  – trivial decoding
  – reduced register pressure
TTA characteristics

Software

• Extra code optimizations

• However: *More difficult to schedule*
Conclusions

• Datapath is the bottleneck on ILP architectures
• Introduction of datapath control to compiler.
• Reduction of complexity -> Result is TTA.
• TTA (Transport Triggered Architecture) has / is
  – extremely flexible building block
  – very scalable
  – semi-automatic design space exploration
  – automatic processor generation
  – low cost, low power