VU2 185.324

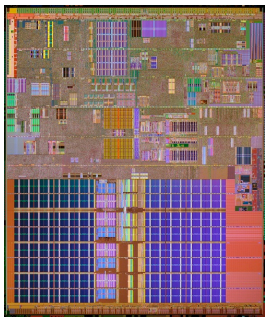
Compilation Techniques for VLIW Architectures

Dietmar Ebner, Florian Brandner
{ebner|brandner}@complang.tuwien.ac.at

http://complang.tuwien.ac.a.t/cd/vliw

In Today's Lecture

- Introduction
- Motivation
- Classification of Embedded Systems
- Application Areas
- Different Forms of Parallelism
- Introduction to VLIW

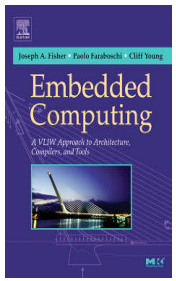


Motivation

"If you round off the fractions, embedded systems consume 100% of the worldwide production of microprocessors."

Jim Turley, Editor, Microprocessor Report

Primary Reference



Embedded Computing

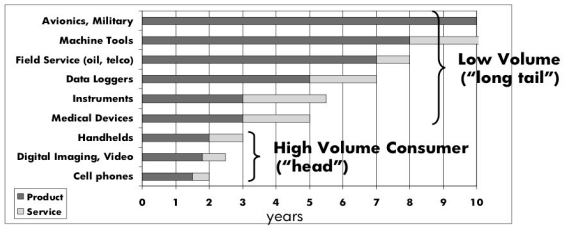
A VLIW Approach to Architecture, Compilers and Tools

J.A. Fisher, P. Faraboschi, C. Young

http://www.vliw.org

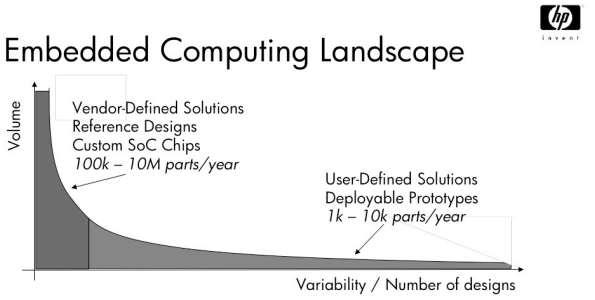
A few copies are available at our library!

Product / Service Lifecycle



Source: P. Faraboschi, HP Research

Market Structure

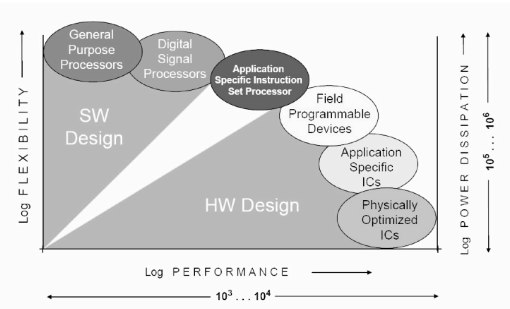


Source: P. Faraboschi, HP Research

Embedded System Landscape

- "Long Tail" (< 10000 units/year)
  - 60% of embedded system developers ship < 1000 units / year
  - Important: NRE costs, flexibility, time-to-market
  - Technologies: FPGAs, SoPC, reconfigurable Architectures
- High Volume Market (> 1M / year)
  - Mainly Consumer Electronics
  - Very few Companies
  - Important: time-to-market, software complexity, **cost**
  - Technologies: MPSoC: general purpose core(s) + accelerators (ASIPs, DSPs, non-programmable-accelerators (NPA))

Flexibility, Power, Performance



Source: T. Noll, RWTH Aachen

Characterization of Embedded Computing

- There is no sharp line between embedded and general purpose computing
- Some different characterizations
  - "everything that is not general purpose"
  - "firmware is *rarely* exchanged"
  - "single purpose"
  - "lack of versatility"
  - "commodities"
- Main purpose is not "computing"

Backward Compatibility

- *Binary Compatibility* has been the only successful approach in the general purpose computing domain
- Important for Embedded Systems:
  - Feature Compatibility

    e.g., floating vs. fixed point hardware, endianess, vector units
  - Source Compatibility
  - Tool Compatibility
  - OS Compatibility

Categorization by Type of Processing Engines 1/2

- Computational Micros

  32 or 64bit datapaths, mainframes to high-end portable devices (PDAs), most high-end RISC/CISC engines

  Examples: **x86(_64), PowerPC, SPARC, Alpha, PA-RISC**
- Embedded General Purpose Micros

  usually 32bit datapaths, often scaled-down versions of the former category, wide range of applications (mainly consumer electronics and communications)

  Examples: **ARM, PowerPC, MIPS, 68K, x86**

Categorization by Type of Processing Engines 2/2

- Digital Signal Processors

  focused on efficient arithmetic computations in tight inner loop kernels

  Vendors: **Texas Instruments, Motorola, Agere, Analog Devices, ...**
- Microcontrollers

  industrial electronics, stand-alone operations, often descendents of 8 and 16bit microprocessor

  Examples: **simple processing units, memories, I/O, buses, peripherals, ...**

Categorization by Application Area

- Image Processing and Cosumer Electronics

  Printers, Audio, Video, Cameras, Home Entertainment, ...
- Communications

  Wired and Cellular Networks, Mobiles, Routers, ...
- Automotive

  Entertainment and Safety Systems, Emissions Control, ...

  ≈ 50 Microprocessors / Vehicle

Categorization by Workload

- Controlling

  (strong) real-time components (real-time OS or microkernel), light computational requirements, simple memory systems, tight couping with peripherals and sensors
- Switching / Routing

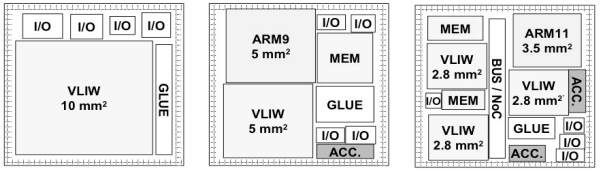
  mostly stream oriented, huge amounts of data, multithreading support, low computational requirements
- Media Processing

  high computational requirements, large memory bandwidth, soft real-time restrictions

The SoC Revolution

- In the last decade, functionally gradually shifted from Hard- to Software
- Recent advanced in VLSI allow for on-chip multi-core systems for high-volume products
- Usually heterogenous ad-hoc designs
  - DSPs, VLIWs, non-programmable accelerators (NPAs)
  - SoC bus or network-on-chip

SoC Evolution



180nm, 250-300 MHz, ('99)

1M gates

Single core + I/O and glue

130nm, 300-400 MHz, ('02)

2.5M gates

Dual core, memory and glue

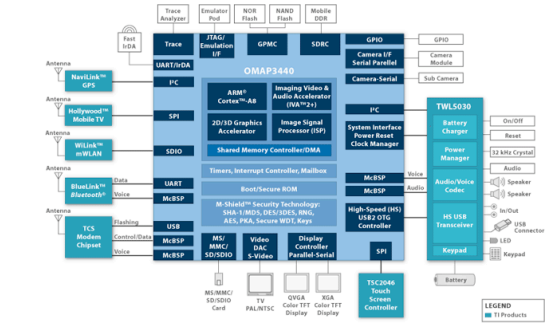
90nm, 400-600 MHz, ('05)

4M gates

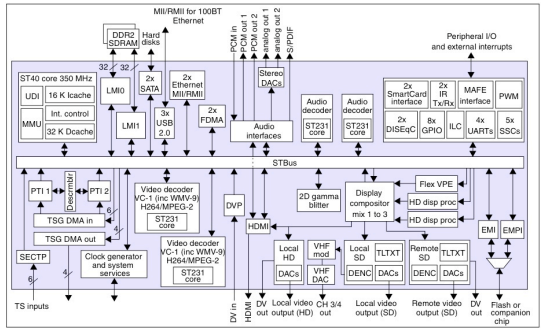
MPSoC, complex interconnect, mix of components

Source: P. Faraboschi, HP Research

Example 1: TI OMAP3440



Example 2: STi7200



Different Forms of Parallelism

- **Instruction Level Parallelism (ILP)**
  covered in this course
- Vector Processing
  vector operations are mixed with normal scalar instructions,
  very popular in 70s/80s (scientific and numeric applications)
- Multiprocessing / Multithreading
  separate processes / threads are execute at the same time
- Micro-SIMD
  registers are treated as “vectors” of smaller data types

Operation Dependence

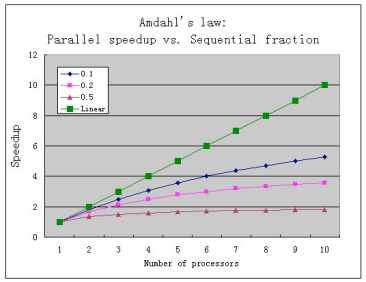
Exploiting ILP requires careful analysis of operation dependencies:

- Control Dependencies
- Data Dependencies
  - Real Dependence (RAW)
  - Anti-Dependence (WAR)
  - Output Dependence (WAW)

Excursion: Amdahl's Law

Used to find the overall speedup (S_{total}) when only a fraction of a program (P_k) is improved by S_k.

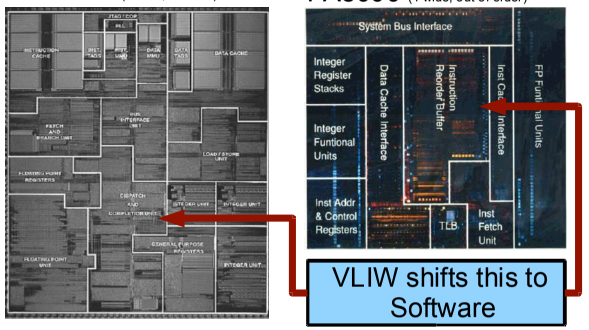
$$S_{total} = \frac{1}{\sum_{k=0}^{n} \left(\frac{P_k}{S_k} \right)}$$



Superscalar vs. VLIW Architectures

PPC 604 (2-wide, in-order)

PA8000 (4-wide, out-of-order)



Design Style Comparison

Superscalar	***VLIW***
• Scheduling is done by hardware	• Scheduling is done by Software
• Sequential stream of scalar operations	• Sequential stream of parallel operations
• Allows for In-order and out-of-order execution	• Only allows for in-order issue
• Number of issued instructions is dynamically determined by a hardware dispatch unit	• Number of issued instructions is statically determined by the compiler
• **Microarchitecture** technique	• **Architecture** technique

VLIW Design Philosophy

- VLIW is a *Design Philosophy* rather than a concrete architectural technique
- "Informal set of guidelines, principles, and common building blocks that distinguish one processor design from another"
- First Implementations in the early 80s
  (Multiflow TRACE, Cydrome)

VLIW Design Philosophy

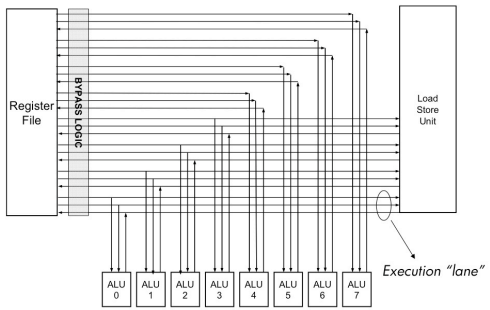
"Expose ILP in the Architecture Design"

- Consequences for several components
  - Instruction Set Architecture (ISA)
  - Microarchitecture / Hardware
  - **Compiler**
- "If you can do it in software, do it in software!"

Typical VLIW Attributes

- Every *instruction* specifies multiple *operations*
- Assignment of operations to execution units is done statically by the compiler
  - Less control logic
  - Shorter Pipelines
- Operation dependencies are provided *explicitly*
- Compiling to a specific implementation is required for correctness

VLIW Datapath Example



Consequences

- Architectures do not scale with number of ALUs and register ports → Clustering
- Compilers cannot always find enough ILP leading to many NOPs → Efficient Encoding
- Binary compatibility cannot be established among various generations in general
- Compiling for a particular implementation is required for correctness

Outlook

- VLIW Design Principles
- ISA Design
- Microarchitectural Implications
- Instruction Set Encoding
- Clustered Architectures
- Instruction Set Extensions