VLSI Trends in Microarchitecture
Past, present and future

TAU university
January 24, 2006
Uri Weiser
Agenda

- Microarchitecture
  - VLSI
  - Trends Past and present:
    » Pipeline, superpipeline
    » Out of Order
    » Branch prediction
    » Caches
    » Trace cache
    » Threads and Chip Multiprocessing
  - Future
    » Asymmetric
    » Accelerators
“[In the beginning] we had little idea of what we had started. ...I remember... saying, ‘Okay, we’ve done integrated circuit. What do we do next?’”

Gordon E. Moore
TRENDS IN VLSI

Sources:
Shekhar Borkar
Uri Weiser
Technology trend

Processor

Intel 386™ DX Processor

Intel 486™ DX Processor

Pentium® Processor

Pentium® Pro Processor

Pentium® II Processor

Pentium® III Processor

Pentium® 4 Processor

Process Technology

1.5µ 1.0µ 0.8µ 0.6µ 0.35µ 0.25µ 0.18µ 0.13µ
**Performance History**

1.0u-0.18u, 1989-2001

Frequency increased 61X

1. 18.3X due to process technology
2. Additional 3.3X due to uArch

Performance increased ~100X

1. 14X due to process tech
2. Additional 7X due to uArch & design
Process Technology: Minimum Feature Size

Source: Intel, SIA Technology Roadmap

Feature Size (microns)

Source: Intel, SIA Technology Roadmap
Transistors on a Chip

2X growth in 1.96 years!

Transistors on a chip doubled every two years

Transistors on a chip doubled every two years
Die Size Growth

Die side (mm) = Area^{1/2}

Die size grows? Is it saturated?

Trends
Lead Microprocessors frequency doubles every 2 years
Frequency of Operation

Trends

Years


Frequency MHZ

1000 100 10 1

8080 8085 8086 80286 386DX-16 486DX-25 486DX-33 386DX-32 486DX2-66 486DX-50

Intel

PPC

Other

Alpha

PPro-300

PPro-225

PP-90

PP-66
Trends

Frequency of Operation (cont.)

Intel
PPC
Other

<table>
<thead>
<tr>
<th>Year</th>
<th>Frequency (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1992</td>
<td>486DX2</td>
</tr>
<tr>
<td>1993</td>
<td>PP-90</td>
</tr>
<tr>
<td>1994</td>
<td>PPC601-100</td>
</tr>
<tr>
<td>1995</td>
<td>PP-100</td>
</tr>
<tr>
<td>1996</td>
<td>PP-120</td>
</tr>
<tr>
<td>1997</td>
<td>PP-133</td>
</tr>
<tr>
<td>1998</td>
<td>PPro-300</td>
</tr>
</tbody>
</table>
Brainiacs and Speed demons

Source: ISCA 95, p. 174
Trends of Future Processors

- SPECint/100MHZ
- PII, PIII
- SPECint = 60
- SPECint = 40
- SPECint = 27
- SPECint = 18
- SPECint = 12
- SPECint = 8
- SPECint = 4

- P M
- P IV
- SPECint = 90
Power density continues to get worse

Trends

Watts/cm²

1000

100

10

1

1.5µ 1µ 0.7µ 0.5µ 0.35µ 0.25µ 0.18µ 0.13µ 0.1µ 0.07µ

- i386
- i486
- Pentium® processor
- Pentium Pro® processor
- Pentium II® processor
- Pentium III® processor
- Hot plate
- Nuclear Reactor
- Rocket Nozzle Sun’s Surface
Trends

On Die Cache Memory

Larger % of die area will be memory
Trends

Process trend – the theory (cont)

Performance driven era vs. Power aware era

Processes

"Old" processes

Power ratio

Process generation p vs. p-1

@ same die area

"new" processes

1.4X Frequency
0.75X voltage
0.7X capacity/transistor
1X area
2X transistor

Power increase/generation: 1.1X

1.4X Frequency
0.9X voltage
0.7X capacity/transistor
1X area
2X transistors

Power increase/generation: 1.6X

Performance driven era (no power constrains)

Power aware era (Performance within power envelop)

Processes
Extension of Pollack’s Rule (Micro32, 1999)

Processor roadmap trend – real life (cont)

Processor generation $k$ vs. $k-1$ compacted @ the same process technology

**Perf/power delta ratio**
- 3 : 1
- 1 : >3

**Technology Generation**
- 1.5
- 1
- 0.7
- 0.5
- 0.35
- 0.18

**Growth (X)**
- 4
- 3
- 2
- 1
- 0

**Power**

**Performance**
Microarchitecture
The Generic Processor

Sophisticated organization to “service” instructions

- **Instruction supply**
  - Instruction cache
  - Branch prediction
  - Instruction decoder
  - ...

- **Execution engine**
  - Instruction scheduler
  - Register files
  - Execution units
  - ...

- **Data supply**
  - Data cache
  - TLB’s
  - ...

- **Goal** - Maximum throughput – balanced design
Parallelism Evolution

Basic configuration

Processor Element

Instruction

Pipeline

Superscalar - In order

Superscalar - Out of Order

VLIW
Pipeline

- Break the work to smaller pieces
- Increased throughput
  - increased # of completed instructions per cycle and reduces cycle time
  - Number of stages varies
- Calls for good balancing among stages

F: Fetch
D: Decode
E: Execute
W: Write Back

1/4 IPC = 4 CPI
1 IPC = 1 CPI

IPC = Instructions Per Cycle
CPI = Cycles Per Instructions

Examples
Intel 486
NS 32532
Pipeline Stalls

- But there are “stalls” in the pipeline
  - “Data Hazards”: Data flow dependency (instructions output/input)
    » Solved by: bypasses, renaming
  - “Control Hazards”: Control flow dependencies
    » Solved by branch prediction
  - “Structural Hazards”: Limited resources
  - Other (Cache misses, long latency instructions, page faults….)
Super Scalar

- Performs more in a single cycle

- Ideally, can multiply the throughput
  - But stall occurs more frequently

2 IPC = 1/2 CPI

Examples
Intel Pentium® Proc.
Alpha 21164
Super Pipeline

- Split to shorter stages - allows higher frequency

Old clk = 0 1 2 3 4 5 6 7 8 9 10 11 12
New clk = 0 1 2 3 4 5 6 7 8 9 10 11 12

F: Fetch
D: Decode
E: Execute
W: Write Back

1 IPC = 1 CPI
33% higher freq!

- Ideally, can (again) multiply the throughput, but
  - Stall penalties do not scale (e.g., control flow stall, cache misses)
  - Clock setup/hold reduces net cycle time - each instruction takes longer!
⇒ In the example above: 2X stages, but performance gain is <33%

Examples:
Intel Pentium® II/III/4
Out Of Order Execution

- **In Order Execution**: instructions are processed in their program order.
  - Limitation to potential Parallelism.
- **OOO**: Instructions are executed based on "data flow" rather than program order

**Before:** src -> dest
(1) load (r10), r21
(2) mov r21, r31 (2 depends on 1)
(3) load a, r11
(4) mov r11, r22 (4 depends on 3)
(5) mov r22, r23 (5 depends on 4)

**After:**
(1) load (r10), r21; (3) load a, r11;
(2) mov r21, r31; (4) mov r11, r22;
(5) mov r22, r23;

- Usually highly superscalar

**Examples:**
- Intel Pentium® II/III/4
- Compaq Alpha 21264

In Order vs. OOO execution.
Assuming:
- Unlimited resources
- 2 cycles load latency
Out Of Order (cont.)

- **Advantages**
  - Help exploit *Instruction Level Parallelism* (ILP)
  - Help cover latencies (e.g., cache miss, divide)
  - Artificially increase the Register file size (i.e. number of registers)
  - Superior/complementary to compiler scheduler
    - Dynamic instruction window
    - Make usage of more registers than the Architecture Registers

- **Complex microarchitecture**
  - Complex scheduler. Involves also
    - Large instruction window
    - Speculative execution
  - Requires reordering back-end mechanism (*retirement*) for:
    - Precise interrupt resolution
    - Misprediction/speculation recovery
    - Memory ordering
Speculation
Branch Prediction

- Goal - ensure enough instruction supply by correct prefetching
- In the past - prefetcher assumed *fall-through*
  - Lose on unconditional branch (e.g., call)
  - Lose on frequently taken branches (e.g., loops)
- Branch prediction
  - Predicts whether a branch is *taken/not taken*
  - Predicts the branch target address
- Misprediction cost varies (higher w/ increased pipeline length)
- Typical Branch prediction rates: ~90%-96%
  - 4%-10% misprediction,
  - 10-25 branches between mispredictions
  - 50-125 instructions between mispredictions
- Misprediction cost increased with
  - Pipeline depth
  - Machine width
  
  » e.g. 3 width x 10 stages = 30 inst flushed!
Target Array + Direction Prediction

- Target and direction are predicted separately
- Tag may be partial

Branch IP

Target Prediction

<table>
<thead>
<tr>
<th>tag</th>
<th>predicted target</th>
</tr>
</thead>
</table>

Direction Prediction

(for conditional branches only)

hit / miss (indicates a branch)  predicted Target Address  predicted direction (taken/not-taken)
Speculative Execution

- Execution of instructions from a predicted (yet unsure) path. Eventually, path may turn wrong.
- Advantages:
  - Ensure instruction supply
  - Allow large scheduling window (for out of order)
- Issues:
  - Misprediction cost
  - Misprediction recovery
**Cache - Motivation & Principle**

- **Memory consumption is growing about 2X every 2 years**
  - Typical size: (Y2000) 64M-128M, (Y2002) 128M-256M

- **CPU speed grows faster than memory and buses**
  - CPU/Bus grew from 1:1 to 6:1, and still growing
    
    | 486 | Pentium | P-II | P-III | P4   |
    |-----|---------|------|-------|------|
    | 25-66MHz | 66-233MHz | 200-450MHz | 0.5-1.33GHz | 1.4-2.4GHz |
    | 33MHz   | 66MHz   | 66-100MHz  | 133-200MHz  | 400MHz    |

  - Memory: DRAM: 60-100ns (“10-16MHz”), Cost: <10$ per 1M
  - SRAM is faster but much more expensive

⇒ Mem**ory becomes the bottleneck for both instructions and data! **Slow or expensive

- **Solution: Cache - A Small, Fast, Close memory**
  - Serves as a buffer between CPU and main memory

- **Contains copy of a portion of the main memory**
  - Small in size
  - Dynamically changed

- **Exploit space and time locality:**
  - Code is fetched sequentially (Space)
  - Code is re-executed (loops, procedures) (Time)
  - Access close or previous data (Space, Time)
The Generic Processor

Speculation – Trace Cache

Instruction Cache

BTB

Instruction fetch/decode

Rename

Scheduler

Trace/decoded cache

Data supply

Execution engine
Fetch bandwidth

Control flow graph
A, B, C are instruction blocks

Dynamic instruction stream

A B C · · · · A B C

time
Trace Cache Concept

• Hold in the “instruction” cache the dynamic stream of the executed instructions

=> Trace cache acts as “branch predictor” + wide instructions supplier
Speculation – Trace Cache

Trace Cache Overview

I Cache
Decoder
Trace Cache
Fill Unit

address

Stream Mode
Build Mode

To Execution Core
Trace cache line

- Tag: identifies starting address of trace
- N instructions (potentially decoded)
- Next address: next fetch address
- path info: branch flags (T, NT), number of branches, trace ends w/ branch?,...
Threads
Scalar Execution

Dependencies reduce throughput/utilization
Superscalar Execution

Time

Generally increases throughput, but decreases utilization
Predication

Generally increases utilization, increases throughput less
(much of the utilization is thrown away)
Threads

CMP – Chip Multi-Processor

Time

Low utilization / higher throughput
Blocked Multithreading

May increase utilization and throughput, but must switch when current thread goes to low utilization/throughput section (e.g. L2 cache miss)
Fine Grained Multithreading

Increases utilization/throughput by reducing impact of dependences
Simultaneous Multithreading

Time

Increases utilization/throughput
Future
Analog Circuit Paradigm

GBWP = Gain Bandwidth Product = constant @ a given technology

e.g. $\text{Gain}_1 \times \text{BW}_1 = \text{Gain}_2 \times \text{BW}_2$
Analog Circuit Paradigm (cont.)

Future

Gain

Frequency

$BW_1$

$BW_n$
“Theory”

- Analog Gain Bandwidth Product (GBWP) is constant for a specific technology, this is also true for other “environments”...

- A computer structure can excel in performance for a specific application set but not at all applications (also true for benchmarks)

- A person can excel in several areas but not at all...

  - ……

examples: benchmarks, application in coming foils people…. 
Tuning for Applications

Future

Performance

Apps₁

Appsₙ

“Applications”
Provide Specialized “efficient” MIPS

- Find a way to support the new performance requirements via an efficient “mechanism”
- A tailored solutions (to a specific application set) can provide an “efficient” MIPS via INTEGRATION, how?
Future

The Need
the environment

● These days is the PC's 20th birthday
  – 835 Million PC sold 1981-2001
  – 138 million PCs in year 2001\(^{(IDC)}\), 10X number of cars, 1.5X of television sold annually
  – 2.2 Billion Email a day, 10X of the first class mail
  – 400 million on line users (200 in Sep99)
  – CPU performance improved $\sim 8000 \times$ !!!

● What will be the need for performance in the coming 20 years?

● What will be the technology progress in the coming 20 years? 10 years? 5 years?

Statistics courtesy of Gartner Dataquest, U.S. News & World Report, Jupiter Internet Population Model, and NUA Internet Surveys
Windows XP examples that needs excessive performance:

- Movie Maker Video Indexing
- Video smoothing

Example 1: Movie Maker Video Indexing

320*240, 30fps
4X slower than real Time on Centrino™ @1.6Ghz

1980x1080, 30fps
~100X over Centrino™ @1.6Ghz
Future

Video smoothing

Example 2:
Emulation of:
Video smoothing
Video Enhancement

352*240 pixels
CPU usage:
70% of Centrino™ @1.6Ghz

1980x1080, 30fps
~21X over Centrino™ @1.6Mhz
The Need

Future
The need: Build a Panorama

Performance: >30min P4 3GHz

Simplified capabilities at Microsoft Digital Image Suite 10 ($129.95)
Future
### CPU Usage

- **an example**
  - (measured on IBM X20)

### Streaming vs. General purpose

- **Streaming Processing**
  - 6 low resolution videos-->
  - Continuous need for MIPS

---

#### General purpose usage

**Excel and Outlook --**

**Burst need for MIPS**
Process trend – the theory (cont)
Performance driven era vs. Power aware era

Process generation p vs. p-1
@ same die area

- **Performance driven era (no power constrains)**
  - 1.4X Frequency
  - 0.75X voltage
  - 0.7X capacity/transistor
  - 1X area
  - 2X transistor
  - Leakage <5%
  - Power increase/generation: 1.1X

- **Power aware era (Performance within power envelop)**
  - 1.4X Frequency
  - 0.9X voltage
  - 0.7X capacity/transistor
  - 1X area
  - 2X transistors
  - Leakage 30%
  - Power increase/generation: 1.6X

Future
Processor roadmap trend – real life (cont)

Extension of Pollack’s Rule (Micro32, 1999)

Processor generation $k$ vs. $k-1$ compacted @ the same process technology

<table>
<thead>
<tr>
<th>Technology Generation</th>
<th>Growth (X)</th>
<th>Perf/power delta ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.18</td>
<td>0-0.35</td>
<td>1 : &gt;3</td>
</tr>
<tr>
<td>0.35</td>
<td>0.35-0.5</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>0.5-0.7</td>
<td></td>
</tr>
<tr>
<td>0.7</td>
<td>0.7-1</td>
<td>3 : 1</td>
</tr>
<tr>
<td>1</td>
<td>1-1.5</td>
<td></td>
</tr>
<tr>
<td>1.5</td>
<td>1.5-4</td>
<td></td>
</tr>
</tbody>
</table>

Future
solution 1: CMP (Chip Multi-Processor)

Future

Performance

Power

One processor

Conventional design

2 CMP

CMP*

3 CMP

4 CMP

penalty: MP

1% performance for 3% in power

CMP = Symmetric General Purpose (GP) cores

1% performance for 3% in power
Future

solution 2: ACCMP (Asymmetric Cluster CMP)

- >3% performance for 1% in power
- ~1% performance for 1% in power
- penalty: Specialized MIPS

Future
• What is the ACCMP?
  – On Die Asymmetric Clusters of cores
  – Efficient specialized MIPS clusters with
    >3-4X performance/power over GP cores
  – Compatible ISA?

• Penalties
  – Multi-Processing (tasks or threads)
    Specialized MIPS

ACCMP is a solution that enables to continue (for a while)
Moore’s performance law within the power envelop
ACCMP

Future

Host Cluster
General Purpose
MIPS

Specialized MIPS
A Cluster

Specialized MIPS
B Cluster
Future - Processors

- applications need
- Specialized MIPS
- Detached from the CPU core
- Different engines
- Mixture of Programmable and fixed function
- ?