Microprocessors vs. DSPs: Fundamentals and Distinctions

Kenton Williston
Berkeley Design Technology, Inc.

info@BDTI.com
http://www.BDTI.com

© 2005 Berkeley Design Technology, Inc.

Workshop Outline

• Definitions
• DSP Algorithms Shape DSPs
• Comparing DSPs and GPPs
• Comparing Performance
• When to Use Which
• Conclusions

© 2005 Berkeley Design Technology, Inc.
Definitions

Microprocessors—General-Purpose Processors (GPPs)
- CPUs for PCs and workstations
  - E.g., Intel Pentium III
- 32-bit GPPs for embedded applications
  - E.g., ARM ARM7

Digital Signal Processors (DSPs)
- Microprocessors specialized for signal processing applications

Low-end DSPs and GPPs
- Architectures targeting extremely cost sensitive markets, often older architectures

High Performance DSPs and GPPs
- Architectures that use advanced techniques to improve parallelism, performance
- Usually have higher clock rates

Example Processors

© 2005 Berkeley Design Technology, Inc.
DSP Algorithms Shape DSPs

How Signal Processing is Different From Other Tasks

• Very computationally demanding
• Requires attention to numeric fidelity
• High memory bandwidth requirements
• Streaming data—and lots of it
• Predictable data access patterns
• Execution-time locality
• Math-centric
• Real-time constraints
• Standards: algorithms, interfaces

DSP Algorithms Shape DSPs

Computational demands ➔ Multiple parallel execution units, hardware acceleration of common DSP functions
Numeric fidelity ➔ Accumulator registers, guard bits, saturation hardware
High memory bandwidth ➔ Harvard architecture, support for parallel moves
Predictable data access patterns ➔ Specialized addressing modes, e.g., modulo, bit-reversed
DSP Algorithms Shape DSPs

- Execution-time locality: Hardware looping, streamlined interrupt handling
- Math-centricity: Single-cycle multiplier(s) or MAC unit(s), MAC instruction
- Streaming data: Data memory usually SRAM, not cache; DMA
- Real-time constraints: Few dynamic features, on-chip SRAM instead of cache
- Standards: 16-bit data types; rounding, saturation modes

Key Processor Attributes

- Program Control
- Data Path
- Address Generation
- Memory System
- Peripherals
- Development Infrastructure
## Comparing DSPs and GPPs

### Instruction Set

#### Low-end DSP
- Specialized, complex instructions
- Multiple operations per instruction
- Poor orthogonality

#### Low-end GPP
- General-purpose instructions
- Typically only one operation per instruction
- Good orthogonality

```
mac x0,y0,a x:(r0)+,x0 y:(r4)+,y0
mpy r2,r3,r4
add r4,r5,r5
mov (r0),r2
mov (r1),r3
inc r0
inc r1
```

#### High-Performance DSP
- Simple to moderately-complex instructions
- Moderate to excellent orthogonality

#### High-Performance GPP
- Baseline:
  - Simple instructions
  - Moderate to excellent orthogonality
- With SIMD extensions:
  - Moderately complex instructions
  - Moderate to excellent orthogonality
**Multi-Issue Approaches**

*VLIW vs. Superscalar*

<table>
<thead>
<tr>
<th>Memory</th>
<th>Instruction scheduling, dispatch</th>
</tr>
</thead>
<tbody>
<tr>
<td>INS 1</td>
<td></td>
</tr>
<tr>
<td>INS 2</td>
<td></td>
</tr>
<tr>
<td>INS 3</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>INS n</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Execution Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALU</td>
</tr>
<tr>
<td>MAC</td>
</tr>
<tr>
<td>BMU</td>
</tr>
<tr>
<td>INS 1</td>
</tr>
<tr>
<td>INS 2</td>
</tr>
<tr>
<td>INS 3</td>
</tr>
<tr>
<td>INS 4</td>
</tr>
<tr>
<td>INS 5</td>
</tr>
<tr>
<td>INS 6</td>
</tr>
</tbody>
</table>

---

**Comparing DSPs and GPPs**

**Architecture Type**

**Low-end DSP and GPP**
- Single-issue
  - DSP
    - Compound instructions perform multiple operations, e.g., multiply + load + modify address register
    - Examples: 'C54x, 'C24x, 'C28x
  - GPP
    - RISC instructions perform single operation, e.g., add, load, or store
    - Examples: ARM7, ARM9

**High-Performance DSP and GPP**
- Multi-issue
  - DSPs
    - Typically VLIW
    - Up to 8 instructions/cycle
    - Examples: ’C64x, SC140, TigerSHARC
  - GPPs
    - Typically superscalar
    - Up to 4 instructions/cycle
    - Example: PowerPC 74xx
Comparing DSPs and GPPs

Trade-Offs: Superscalar vs. VLIW

Superscalar (high-performance GPPs, mostly)
- Increased hardware complexity
  - Silicon area, power consumption
- Dynamic behavior
  - Complex performance model, timing variability
  - Increased performance with binary compatibility
  - Decreased software complexity (programmer/compiler)

VLIW (high-performance DSPs, mostly)
- Decreased hardware complexity
- No dynamic behavior
- Binary compatibility difficult (downward direction)
- Increased software complexity

Comparing DSPs and GPPs

Program Control

Low-end DSP
- Hardware looping
- Interrupts disabled during certain operations
- Limited or no register shadowing
- Simple pipelines
  - Often provide delay slots to hide branch latencies
- May support fast interrupts

Low-end GPP
- Software looping
- Interrupts rarely disabled
- Register shadowing common
- Simple pipelines
  - No delay slots or branch prediction
- May support fast interrupts

© 2005 Berkeley Design Technology, Inc.
### Comparing DSPs and GPPs

#### Program Control

<table>
<thead>
<tr>
<th>High-end DSP</th>
<th>High-end GPP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usually support hardware looping</td>
<td>Software looping</td>
</tr>
<tr>
<td>Interrupts rarely disabled</td>
<td>Interrupts rarely disabled</td>
</tr>
<tr>
<td>May offer shadow registers</td>
<td>Register shadowing common</td>
</tr>
<tr>
<td>Complicated pipelines in some cases</td>
<td>Moderately to extremely complicated pipelines</td>
</tr>
<tr>
<td>• May be non-interlocked</td>
<td>• May have very long instruction latencies</td>
</tr>
<tr>
<td>• May have multi-cycle latencies</td>
<td>• Often use branch prediction</td>
</tr>
<tr>
<td>• May use branch prediction</td>
<td></td>
</tr>
<tr>
<td>May support fast interrupts</td>
<td>May support fast interrupts</td>
</tr>
</tbody>
</table>

#### Branch Prediction: Strengths and Weaknesses

In many applications, branch prediction is very accurate

- This includes signal processing applications, where most branches are part of for-next loops

Complex branch prediction algorithms introduce timing uncertainty

- It can be difficult to predict whether the prediction will be correct at any given instant
Key Processor Attributes

Comparing DSPs and GPPs

Low-end DSP
Dedicated hardware performs all key arithmetic operations in 1 cycle

Usually 16-bit
Hardware support for managing numeric fidelity
• Guard bits, saturation, rounding modes, ...

Limited bit-manipulation capabilities

Low-end GPP
Multiplies often take >1 cycle
Multi-bit shifts often take >1 cycle

Usually 32-bit, integer only
Saturation, rounding typically take extra cycles

May have superior bit-manipulation capabilities
Comparing DSPs and GPPs

**Data Path**

<table>
<thead>
<tr>
<th>High-Performance DSP</th>
<th>High-Performance GPP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Up to 8 arithmetic units</td>
<td>1-3 arithmetic units</td>
</tr>
<tr>
<td>Some specialized arithmetic units</td>
<td>General-purpose arithmetic units</td>
</tr>
<tr>
<td>• E.g., MAC unit, Viterbi unit</td>
<td>• E.g., integer unit, floating-point unit</td>
</tr>
<tr>
<td>Support multiple data sizes</td>
<td>Support multiple data sizes</td>
</tr>
<tr>
<td>Limited to excellent bit-manipulation capabilities</td>
<td>May have superior bit-manipulation capabilities</td>
</tr>
<tr>
<td>Hardware support for managing numeric fidelity</td>
<td>Saturation, rounding typically take extra cycles</td>
</tr>
</tbody>
</table>

SIMD

**Single Instruction, Multiple Data**

- Performs the same operation simultaneously on multiple sets of operands
- Under the control of a single instruction
- Some SIMD processors support multiple data widths (for example, 32-bit, 16-bit, and 8-bit)
**Comparing DSPs and GPPs**

**SIMD Features**

**Low-end DSP and GPP**

- Very limited SIMD features in low-end DSP
  - E.g., dual add, subtract of 16-bit fixed-point data

- No SIMD support in low-end GPP

**High-Performance DSP and GPP**

- Limited to extensive SIMD features in high-end DSPs
  - E.g., TigerSHARC
    - 4 x 32-bit float
    - 4 x 32-bit integer
    - 8 x 16-bit integer
    - 16 x 8-bit integer

- Extensive SIMD features in high-end GPPs
  - E.g., PowerPC 74xx
    - 4 x 32-bit float
    - 4 x 32-bit integer
    - 8 x 16-bit integer
    - 16 x 8-bit integer

---

**SIMD Challenges**

- Each instruction performs lots of work
  - *Data parallelism*

- Algorithms, data organization must be amenable to data-parallel processing
  - May require programmer creativity, alternative algorithms
  - Data-reorganization penalties can be significant

- Compilers generally don’t use SIMD capabilities

- Most effective on algorithms that process large blocks of data
**SIMD Challenges**

*Example: Viterbi Add-Compare-Select (ACS) Loop*

- **Scalar ACS**
  - Scalar Implementation

- **SIMD ACS**
  - SIMD Implementation
  - Rearrange data

**Comparing DSPs and GPPs**

*Addressing*

**Low-end and High-Performance DSP**
- Dedicated address-generation units
- Specialized addressing modes
  - Autoincrement
  - Modulo (circular)
  - Bit-reversed (for FFT)

**Low-end and High-Performance GPP**
- Often, no separate address-generation units
- General-purpose addressing modes
Key Processor Attributes

Program Control  Data Path  Address Generation

Memory System

Development Infrastructure

Comparing DSPs and GPPs

Memory Architecture

Low-end DSP

Harvard architecture
2-4 memory accesses per cycle
No caches; on-chip SRAM
DMA

Low-end GPP

Von Neumann architecture
Typically 1 access per cycle
Typically use cache(s)

© 2005 Berkeley Design Technology, Inc.
Comparing DSPs and GPPs

Memory Architecture

**High-Performance DSP**
- Harvard architecture
- Per cycle accesses:
  - 1-8 instructions
  - two or more 16- to 64-bit data words
- Sometimes caches, often lockable, configurable as SRAM
- DMA

**High-Performance GPP**
- Harvard architecture
- Per cycle accesses:
  - 1-4 instructions
  - ~two 32- to 64-bit or one 128-bit data word
- Usually use caches

Comparing DSPs and GPPs

Caches: Challenges

Caches work by lowering average access time
- They are effective at doing this in many applications
- But access times vary significantly

Some applications are sensitive to maximum access time (not average)
- E.g., many “hard-real-time” signal processing applications

Signal processing access patterns often predictable
- Thus, DMA may be preferable to a cache
- Some recent caches provide pre-fetching capability
- Some DSP’s caches can be locked or configured as part cache, part SRAM
## Comparing DSPs and GPPs

### Dynamic Features

Dynamic features are used heavily in high-end GPPs to boost performance:
- Superscalar execution
- Caches
- Branch prediction
- Data-dependent instruction execution times

These features are occasionally used in DSPs, too.

These features complicate software development for real-time DSP applications:
- Ensuring real-time behavior
- Optimizing code

---

### Low-end GPPs and DSPs

<table>
<thead>
<tr>
<th>GPPs:</th>
<th>DSPs:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic caches common</td>
<td>Rarely have dynamic features</td>
</tr>
<tr>
<td></td>
<td>Small “loop buffer” instruction cache exception</td>
</tr>
</tbody>
</table>

### High-Performance GPPs and DSPs

<table>
<thead>
<tr>
<th>GPPs: Moderate to extensive use of dynamic features</th>
<th>DSPs: Generally avoid dynamic features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic caches standard</td>
<td>Dynamic cache is most common dynamic feature</td>
</tr>
<tr>
<td>Superscalar execution, branch prediction common</td>
<td>Superscalar execution rare</td>
</tr>
<tr>
<td></td>
<td>Branch prediction sometimes used</td>
</tr>
</tbody>
</table>
**Parallelism**

Key implications of differences

- **Cycle efficiency**
  - DSPs have advantage on signal processing tasks
  - But may require special software development strategies—like assembly level programming—to realize full advantage

- **Memory use efficiency**
  - Multi-operation instructions give DSPs advantage on signal processing tasks
  - But GPPs often better on non-signal processing tasks—which typically consumes most of the code space

- **Compiler friendliness**
  - GPPs generally have the advantage
  - SIMD difficult for compilers, whether GPP or DSP
    - Often requires assembly programming or use of high level intrinsics—both of which complicate software development

---

**Key Processor Attributes**
### Comparing DSPs and GPPs

#### On-Chip Integration

<table>
<thead>
<tr>
<th>Low-end GPPs and DSPs</th>
<th>High-Performance GPPs and DSPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Typically, wide range of on-chip peripherals and I/O interfaces</td>
<td>Moderate to extensive on-chip integration</td>
</tr>
<tr>
<td>Often oriented towards consumer applications</td>
<td>• PC CPUs offer very little on-chip integration</td>
</tr>
<tr>
<td>• E.g., video coprocessors, USB ports, ...</td>
<td>• E.g., Viterbi decoding coprocessors, UTOPIA ports, ...</td>
</tr>
</tbody>
</table>

#### Compatibility and Availability

<table>
<thead>
<tr>
<th>Low-end DSP</th>
<th>Low-end GPP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mostly proprietary architectures</td>
<td>Many shared architectures</td>
</tr>
<tr>
<td>• I.e., one architecture, one vendor</td>
<td>• I.e., one architecture, several (to many) vendors</td>
</tr>
<tr>
<td>Limited (at best) compatibility between successive generations</td>
<td>Often binary compatibility between successive generations</td>
</tr>
<tr>
<td>Occasionally available as licensable core</td>
<td>Often available as licensable core</td>
</tr>
<tr>
<td>• E.g., ARM, MIPS</td>
<td></td>
</tr>
</tbody>
</table>
Comparing DSPs and GPPs

Compatibility and Availability

**High-Performance DSP**
- Mostly proprietary architectures
  - Exceptions: StarCore, ZSP

**High-Performance GPP**
- Mostly shared architectures
  - PowerPC, MIPS, ARM, x86

Sometimes binary compatibility between successive generations
  - E.g., ‘C6xxx, StarCore, ZSP

Usually binary compatibility between successive generations

Sometimes available as licensable core
  - E.g., StarCore, CEVA-X, ZSP

Sometimes available as licensable core
  - E.g., ARM, MIPS

Development Support

<table>
<thead>
<tr>
<th></th>
<th>DSPs</th>
<th>GPPs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tools</strong></td>
<td>Primitive to moderately sophisticated</td>
<td>Primitive to very sophisticated</td>
</tr>
<tr>
<td><strong>DSP-specific tool support</strong></td>
<td>Good to excellent E.g., cycle-accurate simulators, DSP C extensions</td>
<td>Poor but improving E.g., general lack of cycle-accurate simulators</td>
</tr>
<tr>
<td><strong>3rd-party DSP software support</strong></td>
<td>Poor to excellent</td>
<td>Limited but growing</td>
</tr>
<tr>
<td><strong>Non-DSP 3rd-party software support</strong></td>
<td>Poor Few to moderate RTOS options</td>
<td>Extensive Few to extensive RTOS options</td>
</tr>
<tr>
<td><strong>Links w/ other high-level tools</strong></td>
<td>E.g., MATLAB</td>
<td>E.g., GUI builders</td>
</tr>
</tbody>
</table>

© 2005 Berkeley Design Technology, Inc.
Comparing Performance

When evaluating processors for signal processing, application-specific, product-specific considerations dominate

- Relative performance can vary dramatically depending on the benchmark

Vendor performance claims should be viewed skeptically

- “MIPS” = ...
- Benchmarks are a sharp tool

Performance is more than speed

- Cost/ perf, energy efficiency, memory use...

Low-end DSPs/ GPPs (Below $10)

<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Model</th>
<th>Frequency (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM</td>
<td>ARM7</td>
<td>133</td>
</tr>
<tr>
<td></td>
<td>ARM8E</td>
<td>250</td>
</tr>
<tr>
<td>ADI</td>
<td>BF53x</td>
<td>400</td>
</tr>
<tr>
<td>TI</td>
<td>'C55x</td>
<td>300</td>
</tr>
<tr>
<td>Freescale</td>
<td>'563xx</td>
<td>180</td>
</tr>
<tr>
<td>LSI</td>
<td>LSI40x</td>
<td>150</td>
</tr>
</tbody>
</table>

BDTI mark2000™ and BDTI simMark2000™

Higher is faster

© 2005 Berkeley Design Technology, Inc.
Comparing Performance

High-Performance DSPs/ GPPs

BDTImark2000™ and BDTIsimMark2000™

Higher is faster

<table>
<thead>
<tr>
<th>Device</th>
<th>BDTImark2000</th>
<th>BDTIsimMark2000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel PIII</td>
<td>3130</td>
<td>9310</td>
</tr>
<tr>
<td>Fixed-pt</td>
<td>(1 GHz)</td>
<td></td>
</tr>
<tr>
<td>Intel PXA27x</td>
<td>2140</td>
<td></td>
</tr>
<tr>
<td>Fixed-pt</td>
<td>(624 MHz)</td>
<td></td>
</tr>
<tr>
<td>Renesas SH775x</td>
<td>750</td>
<td></td>
</tr>
<tr>
<td>Fixed-pt</td>
<td>(240 MHz)</td>
<td></td>
</tr>
<tr>
<td>TI 'C64x</td>
<td>6400</td>
<td></td>
</tr>
<tr>
<td>Fixed-pt</td>
<td>(1 GHz)</td>
<td></td>
</tr>
<tr>
<td>ADI TS201S</td>
<td>4480</td>
<td></td>
</tr>
<tr>
<td>Fixed-pt</td>
<td>(600 MHz)</td>
<td></td>
</tr>
<tr>
<td>Freescale MSC81xx</td>
<td>5610</td>
<td></td>
</tr>
<tr>
<td>Fixed-pt</td>
<td>(500 MHz)</td>
<td></td>
</tr>
</tbody>
</table>

When to Use Which

**DSP**
- Heavy signal processing requirements
- Limited control processing
- The DSP is incumbent
- Software compatibility between generations not required—or can be achieved w/ DSP
- Multi-vendor architecture not desired
- DSP has better integration for application

**GPP**
- Modest signal processing requirements
- Extensive control processing
  - Especially if code density and portability are important
- The GPP is incumbent
- Software compatibility between generations required
- Multi-vendor architecture desired
- GPP has better integration for application
When to Use Which

Challenges in Using GPPs for Signal Processing Tasks

- Not enough DSP horsepower
  - Usually an issue only for very low-end GPPs or very demanding applications
- Limited memory bandwidth
  - Again, mostly an issue for low-end GPPs
- Lack of execution-time predictability
- High cost, power consumption
  - True of PC CPU class GPPs
- Few DSP-oriented development tools
  - E.g., lack of cycle-accurate simulators
- Few DSP-oriented software libraries
- Limited on-chip integration in some cases

Challenges in Using DSPs for Non-Signal-Processing Tasks

- Limited data-type agility
  - Focus on 16-bit fixed-point
- Momentum of popular GPP architectures
- Generally inferior tools (except for DSP-oriented features)
- Inferior third-party support for non-DSP tasks
  - E.g., RTOSs
- Proprietary architectures
Conclusions

Take-Away Points

When either a GPP or DSP is fast enough, other factors become prominent:

- Energy efficiency
- Integration
- Compatibility, availability
  - Multi-vendor architectures
  - Licensable cores
- Tools
  - DSP-oriented
  - General-purpose
- Software

Will DSP-Capable GPPs Render DSPs Obsolete?

No, but they will pose increasingly strong competition

- Why have GPP+DSP if GPP alone is good enough?

Demands of most communications and media-processing applications will continue to favor DSPs

Software infrastructure is key

- DSPs have the advantage for DSP tasks
- GPPs have the advantage for other tasks

For DSPs, the competitive field has become much larger

- Differentiating criteria are changing
For More Information...
www.BDTI.com

Inside [DSP] newsletter and quarterly reports
Benchmark scores for dozens of processors
Pocket Guide to Processors for DSP
  • Basic stats on over 40 processors
Articles, white papers, and presentation slides
  • Processor architectures and performance
  • Signal processing applications
  • Signal processing software optimization
comp.dsp FAQ

© 2005 Berkeley Design Technology, Inc.