


# Microprocessors vs. DSPs: Fundamentals and Distinctions

*Insight, Analysis, and Advice on Signal Processing Technology*




## Microprocessors vs. DSPs: Fundamentals and Distinctions (DSP-502)

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. 2

© 2005 Berkeley Design Technology, Inc.

# Microprocessors vs. DSPs: Fundamentals and Distinctions

**BDTi**

## 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

© 2005 Berkeley Design Technology, Inc. 3

**BDTi**


## Example Processors

The diagram is a 2D plot with two axes. The vertical axis is labeled 'DSPs' at the top and 'GPPs' at the bottom. The horizontal axis is labeled 'Low-end' on the left and 'High-performance' on the right. Processors are plotted as follows:

- Low-end, GPPs:** ARM7, ARM9, ARM9E
- Low-end, DSPs:** 'C24x, 'C28x, 58000/E
- Mid-range, GPPs:** ARM10, ARM11
- Mid-range, DSPs:** 'C55x
- High-performance, GPPs:** PowerPC (G4), P4
- High-performance, DSPs:** Blackfin, 'C62x, 'C64x

© 2005 Berkeley Design Technology, Inc. 4

© 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

© 2005 Berkeley Design Technology, Inc. 5



## 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

© 2005 Berkeley Design Technology, Inc. 6

**BDTi**

## 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

© 2005 Berkeley Design Technology, Inc. 7

**BDTi**

## Key Processor Attributes

© 2005 Berkeley Design Technology, Inc. 8



## Comparing DSPs and GPPs


**Instruction Set**

<p><u>Low-end DSP</u></p> <p>Specialized, complex instructions</p> <p>Multiple operations per instruction</p> <p>Poor orthogonality</p>	<p><u>Low-end GPP</u></p> <p>General-purpose instructions</p> <p>Typically only one operation per instruction</p> <p>Good orthogonality</p>
---	---

<pre>mac x0,y0,a x:(r0)+,x0 y:(r4)+,y0</pre>	<pre>mpy r2,r3,r4 add r4,r5,r5 mov (r0),r2 mov (r1),r3 inc r0 inc r1</pre>
--	--

© 2005 Berkeley Design Technology, Inc. 9

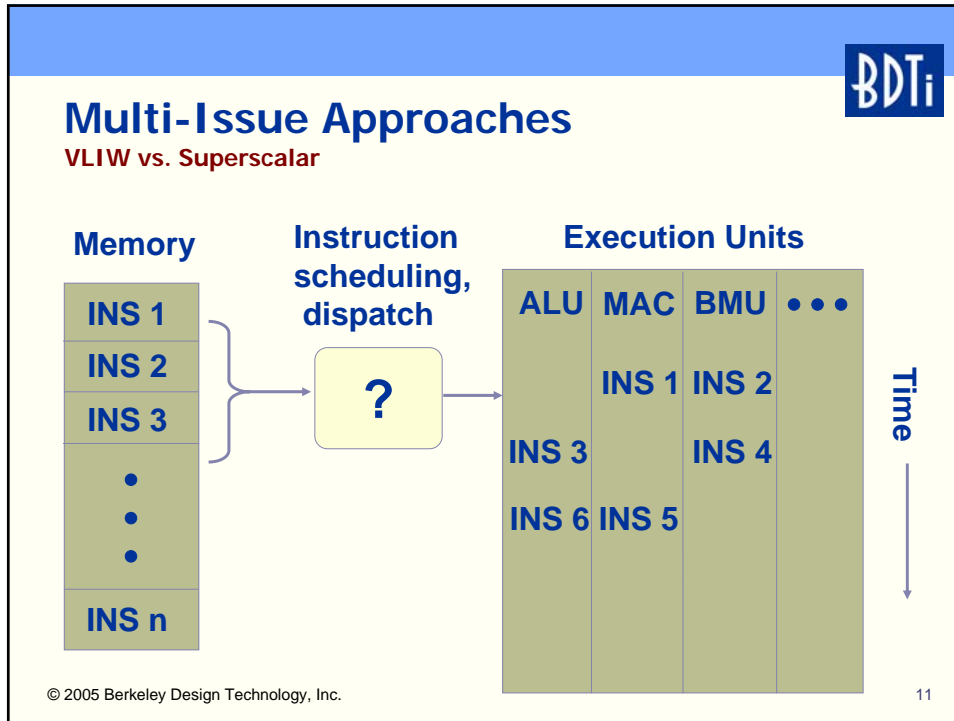


## Comparing DSPs and GPPs

**Instruction Set**

<p><u>High-Performance DSP</u></p> <p>Simple to moderately-complex instructions</p> <p>Moderate to excellent orthogonality</p>	<p><u>High-Performance GPP</u></p> <p><u>Baseline:</u></p> <p>Simple instructions</p> <p>Moderate to excellent orthogonality</p> <p><u>With SIMD extensions:</u></p> <p>Moderately complex instructions</p> <p>Moderate to excellent orthogonality</p>
--	---


© 2005 Berkeley Design Technology, Inc. 10



**Comparing DSPs and GPPs**  
**Architecture Type**

<u>Low-end DSP and GPP</u>	<u>High-Performance DSP and GPP</u>
<p>Single-issue</p> <ul style="list-style-type: none"> <li>DSP                             <ul style="list-style-type: none"> <li>Compound instructions perform multiple operations, e.g., multiply + load + modify address register</li> <li>Examples: 'C54x, 'C24x, 'C28x</li> </ul> </li> <li>GPP                             <ul style="list-style-type: none"> <li>RISC instructions perform single operation, e.g., add, load, or store</li> <li>Examples: ARM7, ARM9</li> </ul> </li> </ul>	<p>Multi-issue</p> <ul style="list-style-type: none"> <li>DSPs                             <ul style="list-style-type: none"> <li>Typically VLIW</li> <li>Up to 8 instructions/cycle</li> <li>Examples: 'C64x, SC140, TigerSHARC</li> </ul> </li> <li>GPPs                             <ul style="list-style-type: none"> <li>Typically superscalar</li> <li>Up to 4 instructions/cycle</li> <li>Example: PowerPC 74xx</li> </ul> </li> </ul>

© 2005 Berkeley Design Technology, Inc. 12



## 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

© 2005 Berkeley Design Technology, Inc. 13




## Comparing DSPs and GPPs

**Program Control**

<u>Low-end DSP</u>	<u>Low-end GPP</u>
Hardware looping	Software looping
Interrupts disabled during certain operations	Interrupts rarely disabled
Limited or no register shadowing	Register shadowing common
Simple pipelines <ul style="list-style-type: none"><li>• Often provide delay slots to hide branch latencies</li></ul>	Simple pipelines <ul style="list-style-type: none"><li>• No delay slots or branch prediction</li></ul>
May support fast interrupts	May support fast interrupts

© 2005 Berkeley Design Technology, Inc. 14




## Comparing DSPs and GPPs

**Program Control**

<u>High-end DSP</u>	<u>High-end GPP</u>
Usually support hardware looping	Software looping
Interrupts rarely disabled	Interrupts rarely disabled
May offer shadow registers	Register shadowing common
Complicated pipelines in some cases	Moderately to extremely complicated pipelines
<ul style="list-style-type: none"><li>• May be non-interlocked</li><li>• May have multi-cycle latencies</li><li>• May use branch prediction</li></ul>	<ul style="list-style-type: none"><li>• May have very long instruction latencies</li><li>• Often use branch prediction</li></ul>
May support fast interrupts	May support fast interrupts

© 2005 Berkeley Design Technology, Inc. 15



## Comparing DSPs and GPPs

**Branch Prediction: Strengths and Weaknesses**

In many applications, branch prediction is very accurate

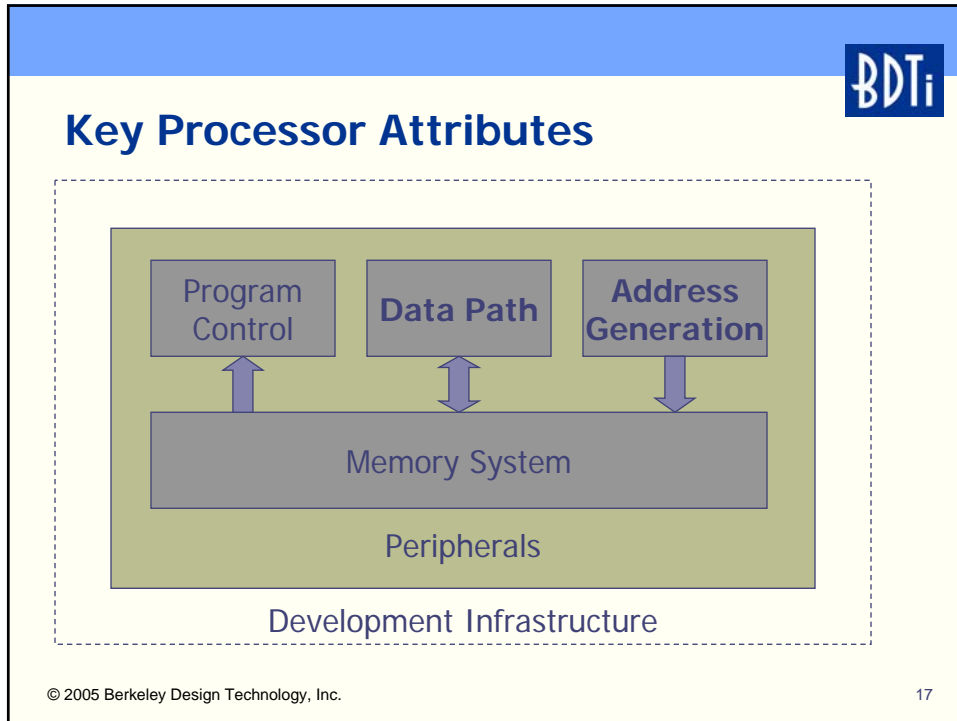
- This includes signal processing applications, where most branches are part of for-next loops

Complex branch prediction algorithms introduce timing uncertainty

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

© 2005 Berkeley Design Technology, Inc. 16





The table, titled "Comparing DSPs and GPPs", compares the "Data Path" characteristics of a "Low-end DSP" and a "Low-end GPP". The BDTi logo is in the top right corner.

<u>Low-end DSP</u>	<u>Low-end GPP</u>
Dedicated hardware performs all key arithmetic operations in 1 cycle	Multiplies often take >1 cycle
Usually 16-bit	Multi-bit shifts often take >1 cycle
Hardware support for managing numeric fidelity <ul style="list-style-type: none"><li>• Guard bits, saturation, rounding modes, ...</li></ul>	Usually 32-bit, integer only
Limited bit-manipulation capabilities	Saturation, rounding typically take extra cycles
	May have superior bit-manipulation capabilities

© 2005 Berkeley Design Technology, Inc. 18

**BDTi**

## Comparing DSPs and GPPs

**Data Path**

<u>High-Performance DSP</u>	<u>High-Performance GPP</u>
Up to 8 arithmetic units	1-3 arithmetic units
Some specialized arithmetic units <ul style="list-style-type: none"><li>• E.g., MAC unit, Viterbi unit</li></ul>	General-purpose arithmetic units <ul style="list-style-type: none"><li>• E.g., integer unit, floating-point unit</li></ul>
Support multiple data sizes	Support multiple data sizes
Limited to excellent bit-manipulation capabilities	May have superior bit-manipulation capabilities
Hardware support for managing numeric fidelity	Saturation, rounding typically take extra cycles

© 2005 Berkeley Design Technology, Inc. 19

**BDTi**

## SIMD

**Single Instruction, Multiple Data**


The diagram illustrates the SIMD (Single Instruction, Multiple Data) architecture. At the top, there are two pairs of 16-bit data inputs. Arrows from these inputs cross and feed into two parallel arithmetic units. Each unit contains symbols for addition (+), subtraction (-), multiplication (x), and division (÷). A dashed line connects the two units, indicating they share a common instruction. Below each unit, an arrow points to a 16-bit data output. This shows that a single instruction is applied to multiple data points simultaneously.

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)

© 2005 Berkeley Design Technology, Inc. 20




## Comparing DSPs and GPPs

**SIMD Features**

<u>Low-end DSP and GPP</u>	<u>High-Performance DSP and GPP</u>
<p>Very limited SIMD features in low-end DSP</p> <ul style="list-style-type: none"><li>• E.g., dual add, subtract of 16-bit fixed-point data</li></ul> <p>No SIMD support in low-end GPP</p>	<p>Limited to extensive SIMD features in high-end DSPs</p> <ul style="list-style-type: none"><li>• E.g., TigerSHARC<ul style="list-style-type: none"><li>• 4 x 32-bit float</li><li>• 4 x 32-bit integer</li><li>• 8 x 16-bit integer</li><li>• 16 x 8-bit integer</li></ul></li></ul> <p>Extensive SIMD features in high-end GPPs</p> <ul style="list-style-type: none"><li>• E.g., PowerPC 74xx<ul style="list-style-type: none"><li>• 4 x 32-bit float</li><li>• 4 x 32-bit integer</li><li>• 8 x 16-bit integer</li><li>• 16 x 8-bit integer</li></ul></li></ul>

© 2005 Berkeley Design Technology, Inc. 21



## 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

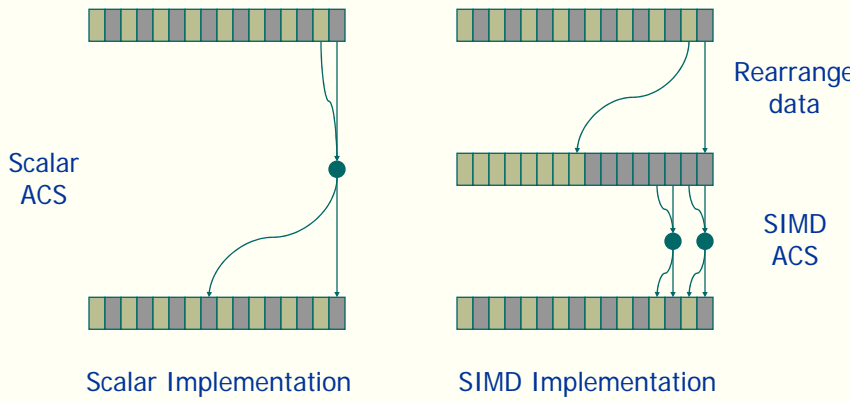
Most effective on algorithms that process large blocks of data

© 2005 Berkeley Design Technology, Inc. 22




## SIMD Challenges

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



© 2005 Berkeley Design Technology, Inc. 23

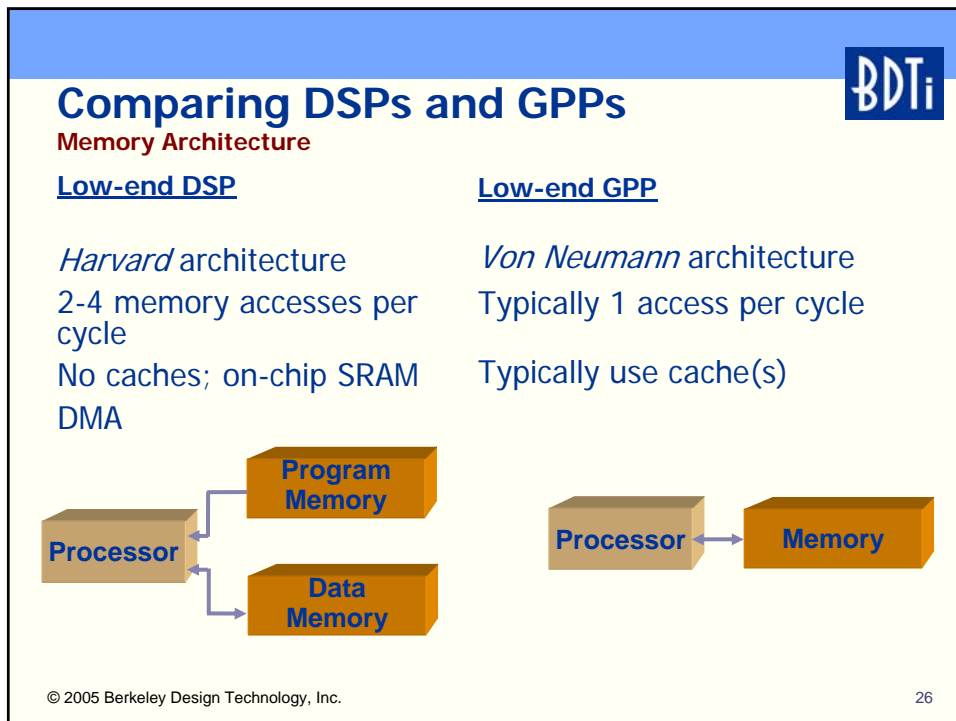
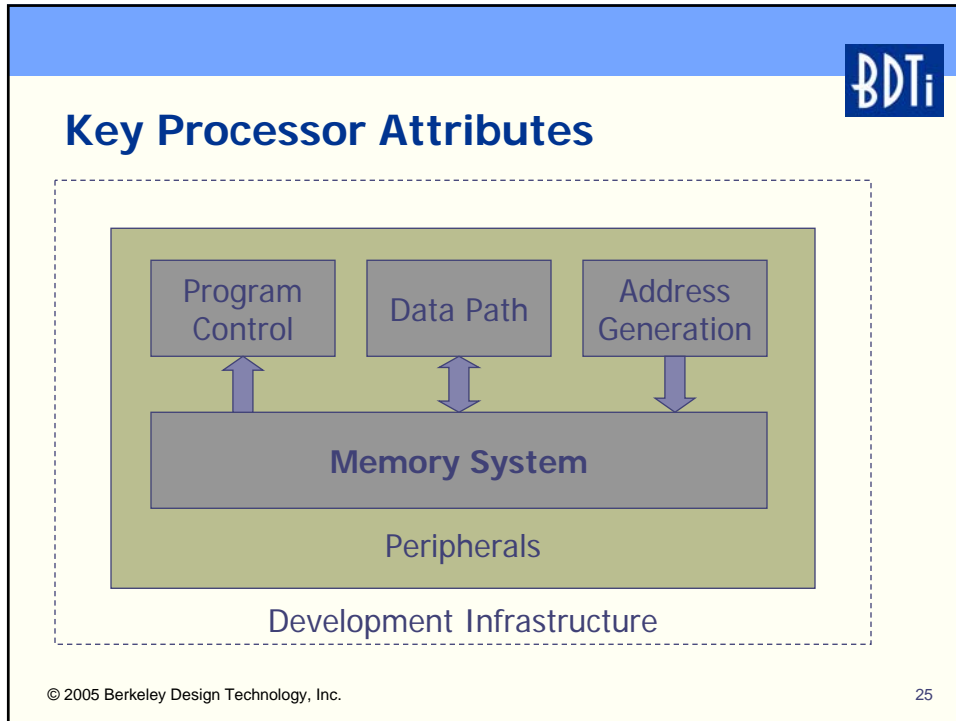



## Comparing DSPs and GPPs

### Addressing

<p><u>Low-end and High-Performance DSP</u></p> <p>Dedicated address-generation units</p> <p>Specialized addressing modes</p> <ul style="list-style-type: none"><li>• Autoincrement</li><li>• Modulo (circular)</li><li>• Bit-reversed (for FFT)</li></ul>	<p><u>Low-end and High-Performance GPP</u></p> <p>Often, no separate address-generation units</p> <p>General-purpose addressing modes</p>
---	---

© 2005 Berkeley Design Technology, Inc. 24

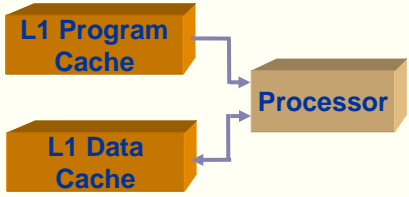
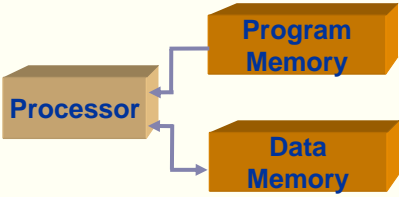





## Comparing DSPs and GPPs

### Memory Architecture

<p><b>High-Performance DSP</b> Harvard architecture Per cycle accesses:</p> <ul style="list-style-type: none"><li>• 1-8 instructions</li><li>• two or more 16- to 64-bit data words</li></ul> <p>Sometimes caches, often lockable, configurable as SRAM DMA</p>	<p><b>High-Performance GPP</b> Harvard architecture Per cycle accesses:</p> <ul style="list-style-type: none"><li>• 1-4 instructions</li><li>• ~two 32- to 64-bit or one 128-bit data word</li></ul> <p>Usually use caches</p>
---	--



© 2005 Berkeley Design Technology, Inc.27



## 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

© 2005 Berkeley Design Technology, Inc.28

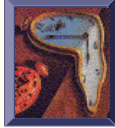


## Comparing DSPs and GPPs

**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

© 2005 Berkeley Design Technology, Inc. 29



## Comparing DSPs and GPPs

**Dynamic Features**

<p><u>Low-end GPPs and DSPs</u></p> <p>GPPs:</p> <ul style="list-style-type: none"><li>• Dynamic caches common</li></ul> <p>DSPs:</p> <ul style="list-style-type: none"><li>• Rarely have dynamic features<ul style="list-style-type: none"><li>• Small "loop buffer" instruction cache exception</li></ul></li></ul>	<p><u>High-Performance GPPs and DSPs</u></p> <p>GPPs: Moderate to extensive use of dynamic features</p> <ul style="list-style-type: none"><li>• Dynamic caches standard</li><li>• Superscalar execution, branch prediction common</li></ul> <p>DSPs: Generally avoid dynamic features</p> <ul style="list-style-type: none"><li>• Dynamic cache is most common dynamic feature</li><li>• Superscalar execution rare</li><li>• Branch prediction sometimes used</li></ul>
---	--

© 2005 Berkeley Design Technology, Inc. 30



## Parallelism

### Key implications of differences

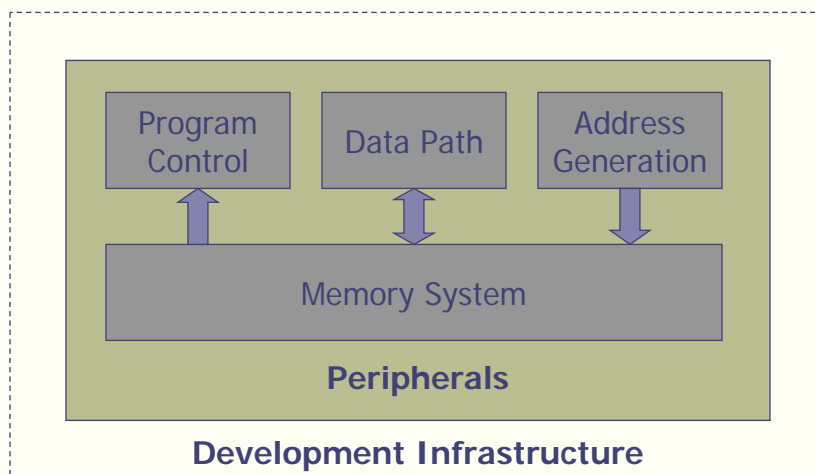
- Cycle efficiency
  - DSPs have advantage on signal processing tasks
    - But may require special software development strategies—like assembly level programming—to realize full advantage
- Memory use efficiency
  - Multi-operation instructions give DSPs advantage on signal processing tasks
  - But 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

© 2005 Berkeley Design Technology, Inc.

31



## Key Processor Attributes




© 2005 Berkeley Design Technology, Inc.

32

© 2005 Berkeley Design Technology, Inc.






## Comparing DSPs and GPPs

**On-Chip Integration**

<p><u>Low-end GPPs and DSPs</u></p> <p>Typically, wide range of on-chip peripherals and I/O interfaces</p> <p>Often oriented towards consumer applications</p> <ul style="list-style-type: none"><li>• E.g., video coprocessors, USB ports, ...</li></ul>	<p><u>High-Performance GPPs and DSPs</u></p> <p>Moderate to extensive on-chip integration</p> <ul style="list-style-type: none"><li>• PC CPUs offer very little on-chip integration</li></ul> <p>Often oriented towards communications infrastructure</p> <ul style="list-style-type: none"><li>• E.g., Viterbi decoding coprocessors, UTOPIA ports, ...</li></ul>
---	--

© 2005 Berkeley Design Technology, Inc. 33




## Comparing DSPs and GPPs

**Compatibility and Availability**

<p><u>Low-end DSP</u></p> <p>Mostly proprietary architectures</p> <ul style="list-style-type: none"><li>• I.e., one architecture, one vendor</li></ul> <p>Limited (at best) compatibility between successive generations</p> <p>Occasionally available as licensable core</p>	<p><u>Low-end GPP</u></p> <p>Many shared architectures</p> <ul style="list-style-type: none"><li>• I.e., one architecture, several (to many) vendors</li></ul> <p>Often binary compatibility between successive generations</p> <p>Often available as licensable core</p> <ul style="list-style-type: none"><li>• E.g., ARM, MIPS</li></ul>
---	---

© 2005 Berkeley Design Technology, Inc. 34




## Comparing DSPs and GPPs

### Compatibility and Availability

<u>High-Performance DSP</u>	<u>High-Performance GPP</u>
Mostly proprietary architectures <ul style="list-style-type: none"> <li>• Exceptions: StarCore, ZSP</li> </ul>	Mostly shared architectures <ul style="list-style-type: none"> <li>• PowerPC, MIPS, ARM, x86</li> </ul>
Sometimes binary compatibility between successive generations <ul style="list-style-type: none"> <li>• E.g., 'C6xxx, StarCore, ZSP</li> </ul>	Usually binary compatibility between successive generations
Sometimes available as licensable core <ul style="list-style-type: none"> <li>• E.g., StarCore, CEVA-X, ZSP</li> </ul>	Sometimes available as licensable core <ul style="list-style-type: none"> <li>• E.g., ARM, MIPS</li> </ul>

© 2005 Berkeley Design Technology, Inc.
35




## Comparing DSPs and GPPs

### Development Support

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

© 2005 Berkeley Design Technology, Inc.
36



## Comparing Performance

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

- Relative performance can vary dramatically depending on the benchmark

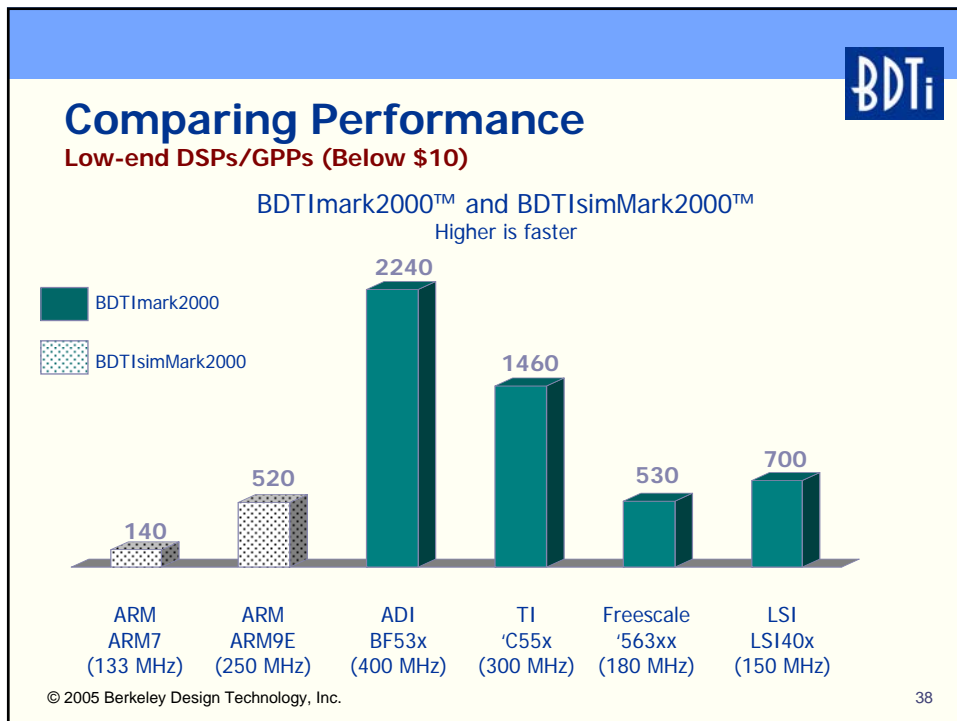
Vendor performance claims should be viewed skeptically

- "MIPS" = ...
- Benchmarks are a sharp tool

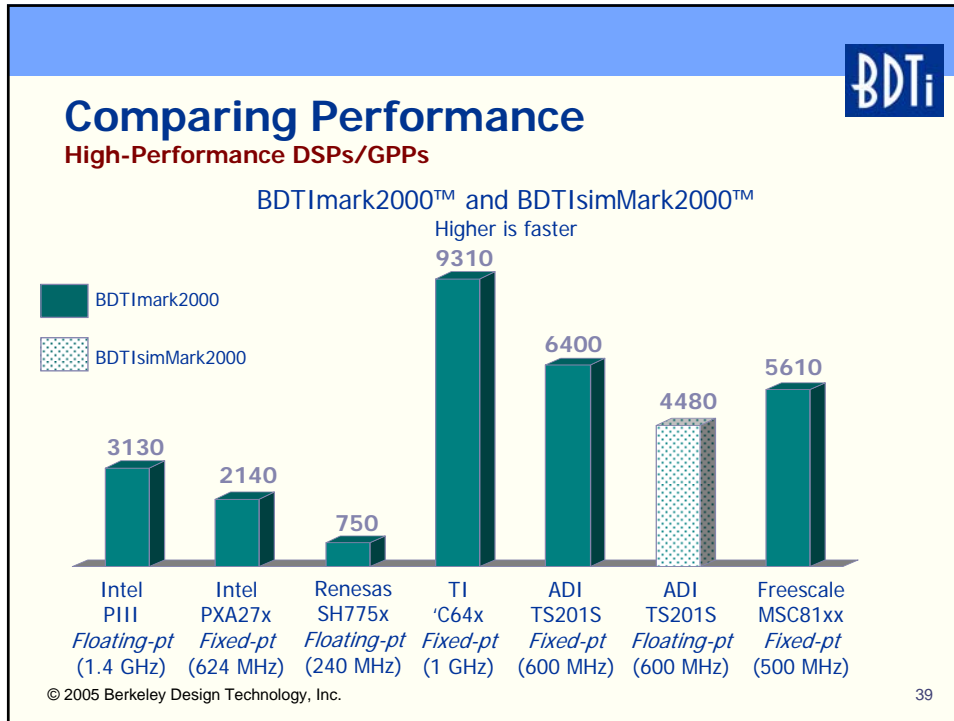
**Performance is more than speed**

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

© 2005 Berkeley Design Technology, Inc. 37



# Microprocessors vs. DSPs: Fundamentals and Distinctions




## When to Use Which

<p><b><u>DSP</u></b></p> <ul style="list-style-type: none"> <li>• Heavy signal processing requirements</li> <li>• Limited control processing</li> <li>• The DSP is incumbent</li> <li>• Software compatibility between generations not required—or can be achieved w/ DSP</li> <li>• Multi-vendor architecture not desired</li> <li>• DSP has better integration for application</li> </ul>	<p><b><u>GPP</u></b></p> <ul style="list-style-type: none"> <li>• Modest signal processing requirements</li> <li>• Extensive control processing <ul style="list-style-type: none"> <li>• Especially if code density and portability are important</li> </ul> </li> <li>• The GPP is incumbent</li> <li>• Software compatibility between generations required</li> <li>• Multi-vendor architecture desired</li> <li>• GPP has better integration for application</li> </ul>
---	--

© 2005 Berkeley Design Technology, Inc. 40

© 2005 Berkeley Design Technology, Inc.




## 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

© 2005 Berkeley Design Technology, Inc. 41




## When to Use Which

**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

© 2005 Berkeley Design Technology, Inc. 42




## 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

© 2005 Berkeley Design Technology, Inc. 43



## Conclusions

**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

© 2005 Berkeley Design Technology, Inc. 44

## Microprocessors vs. DSPs: Fundamentals and Distinctions

**For More Information...**

**[www.BDTI.com](http://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



2004 Edition

45

© 2005 Berkeley Design Technology, Inc.

© 2005 Berkeley Design Technology, Inc.