


Microprocessors vs. DSPs: Fundamentals and Distinctions


Insight, Analysis, and Advice on Signal Processing Technology



Microprocessors vs. DSPs: Fundamentals and Distinctions

Berkeley Design Technology, Inc.
<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 a vertical axis and a horizontal axis. 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, DSPs:** 'C54x, 'C24x, 'C28x, 58000/E
- Low-end, GPPs:** ARM7, ARM9, ARM9E, ARM10
- High-performance, DSPs:** 'C55x, Blackfin, 'C62x, 'C64x
- High-performance, GPPs:** ARM11, PowerPC (G4), P4

© 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

```
graph TD; subgraph DI [Development Infrastructure]; subgraph PS [Processor System]; PC[Program Control]; DP[Data Path]; AG[Address Generation]; MS[Memory System]; PC --> MS; DP <--> MS; MS --> AG; end; MS --- P[Peripherals]; end;
```

© 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

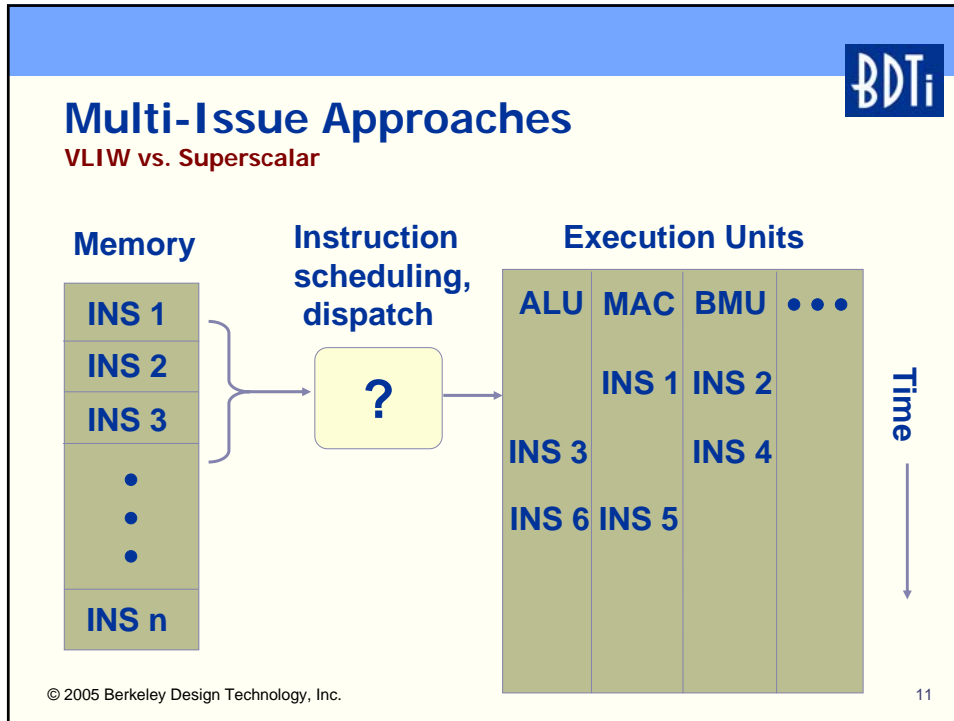


Comparing DSPs and GPPs

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

© 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">• Often provide delay slots to hide branch latencies	Simple pipelines <ul style="list-style-type: none">• No delay slots or branch prediction
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">• May be non-interlocked• May have multi-cycle latencies• May use branch prediction	<ul style="list-style-type: none">• May have very long instruction latencies• Often use branch prediction
May support fast interrupts	May support fast interrupts

© 2005 Berkeley Design Technology, Inc. 15



Comparing DSPs and GPPs

Branch Prediction: Strengths and Weaknesses

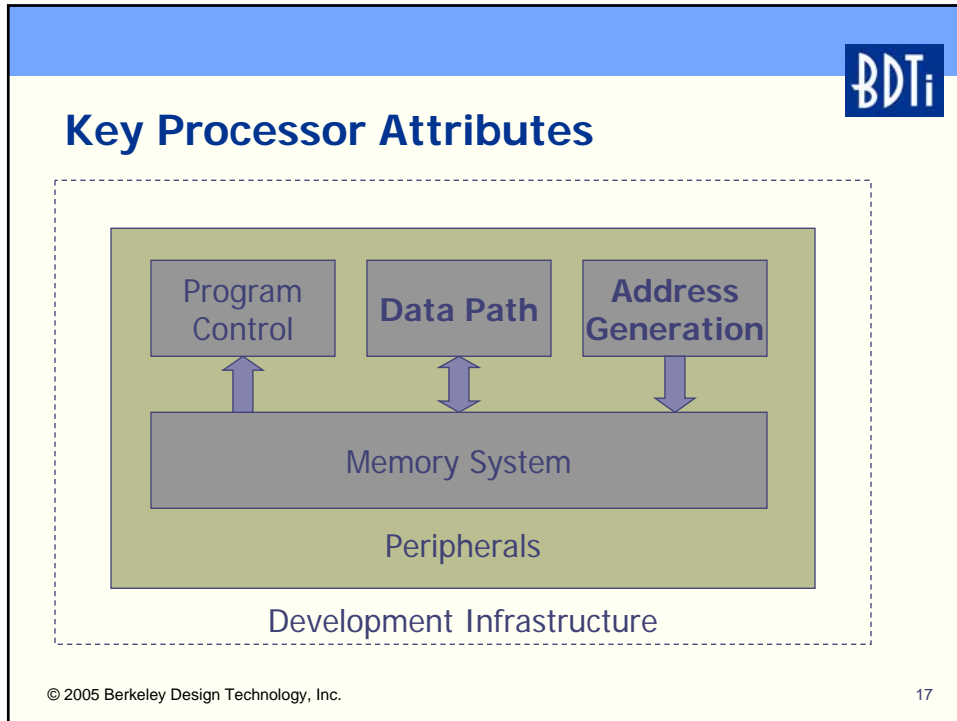
In many applications, branch prediction is very accurate

- 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">• Guard bits, saturation, rounding modes, ...	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

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

© 2005 Berkeley Design Technology, Inc. 19

BDTi

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)

© 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">• E.g., dual add, subtract of 16-bit fixed-point data <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">• E.g., TigerSHARC<ul style="list-style-type: none">• 4 x 32-bit float• 4 x 32-bit integer• 8 x 16-bit integer• 16 x 8-bit integer <p>Extensive SIMD features in high-end GPPs</p> <ul style="list-style-type: none">• E.g., PowerPC 74xx<ul style="list-style-type: none">• 4 x 32-bit float• 4 x 32-bit integer• 8 x 16-bit integer• 16 x 8-bit integer

© 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

BDTi

SIMD Challenges

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

Scalar ACS

Scalar Implementation

Rearrange data

SIMD ACS

SIMD Implementation

© 2005 Berkeley Design Technology, Inc. 23

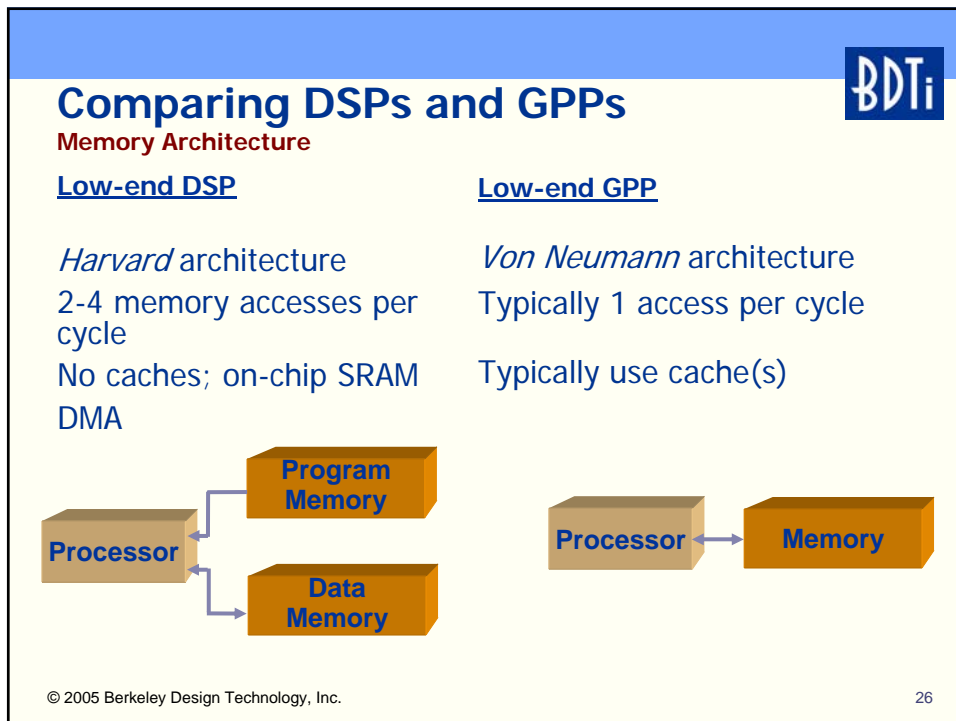
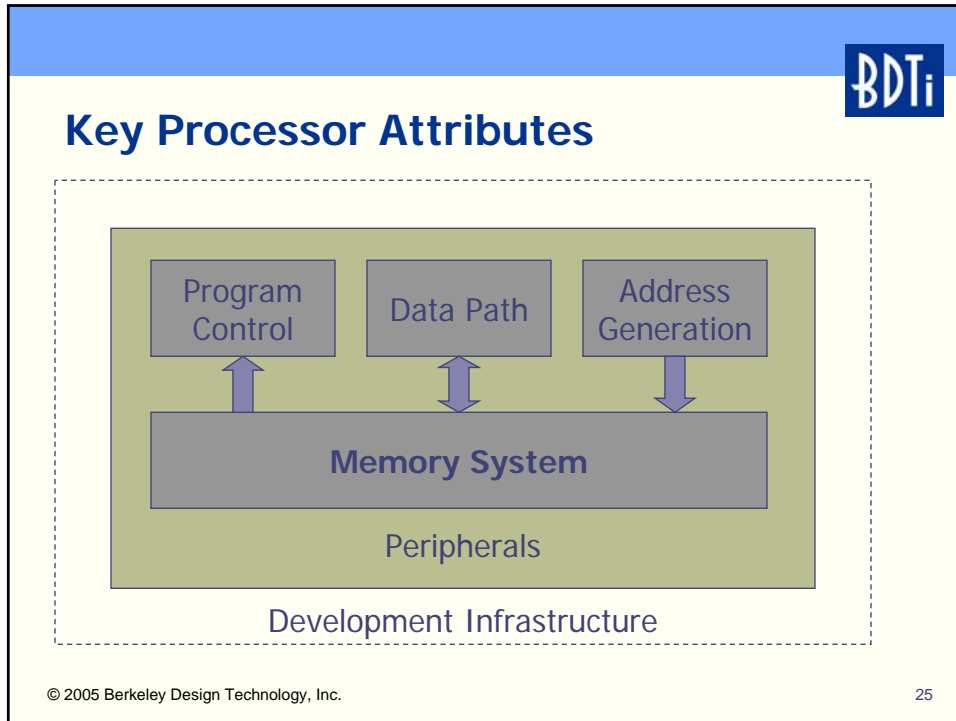
BDTi

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"> • Autoincrement • Modulo (circular) • Bit-reversed (for FFT) 	<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



BDTi

Comparing DSPs and GPPs

Memory Architecture

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

© 2005 Berkeley Design Technology, Inc. 27

BDTi

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

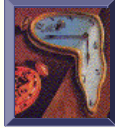


Comparing DSPs and GPPs

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">• Dynamic caches common <p>DSPs:</p> <ul style="list-style-type: none">• Rarely have dynamic features<ul style="list-style-type: none">• Small "loop buffer" instruction cache exception	<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">• Dynamic caches standard• Superscalar execution, branch prediction common <p>DSPs: Generally avoid dynamic features</p> <ul style="list-style-type: none">• Dynamic cache is most common dynamic feature• Superscalar execution rare• Branch prediction sometimes used
---	--

© 2005 Berkeley Design Technology, Inc. 30



Parallelism

Key implications of differences

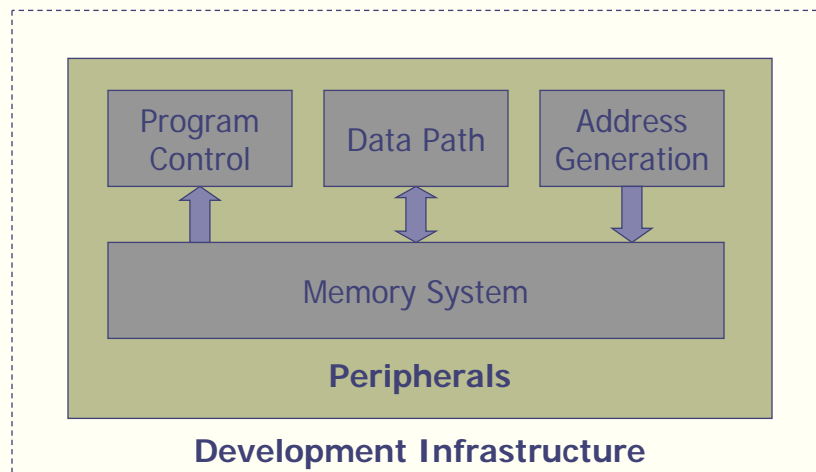
- Cycle efficiency
 - DSPs have advantage on signal processing tasks
 - But may require special software development strategies—like assembly level programming—to realize full advantage
- Memory use efficiency
 - Multi-operation instructions give DSPs advantage on signal processing tasks
 - But GPPs 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">• E.g., video coprocessors, USB ports, ...	<p><u>High-Performance GPPs and DSPs</u></p> <p>Moderate to extensive on-chip integration</p> <ul style="list-style-type: none">• PC CPUs offer very little on-chip integration <p>Often oriented towards communications infrastructure</p> <ul style="list-style-type: none">• E.g., Viterbi decoding coprocessors, UTOPIA ports, ...
---	--

© 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">• I.e., one architecture, one vendor <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">• I.e., one architecture, several (to many) vendors <p>Often binary compatibility between successive generations</p> <p>Often available as licensable core</p> <ul style="list-style-type: none">• E.g., ARM, MIPS
---	---

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

© 2005 Berkeley Design Technology, Inc.
35



Comparing DSPs and GPPs

Development Support

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

© 2005 Berkeley Design Technology, Inc.
36

BDTi

Comparing Performance

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

- Relative performance can vary dramatically depending on the benchmark

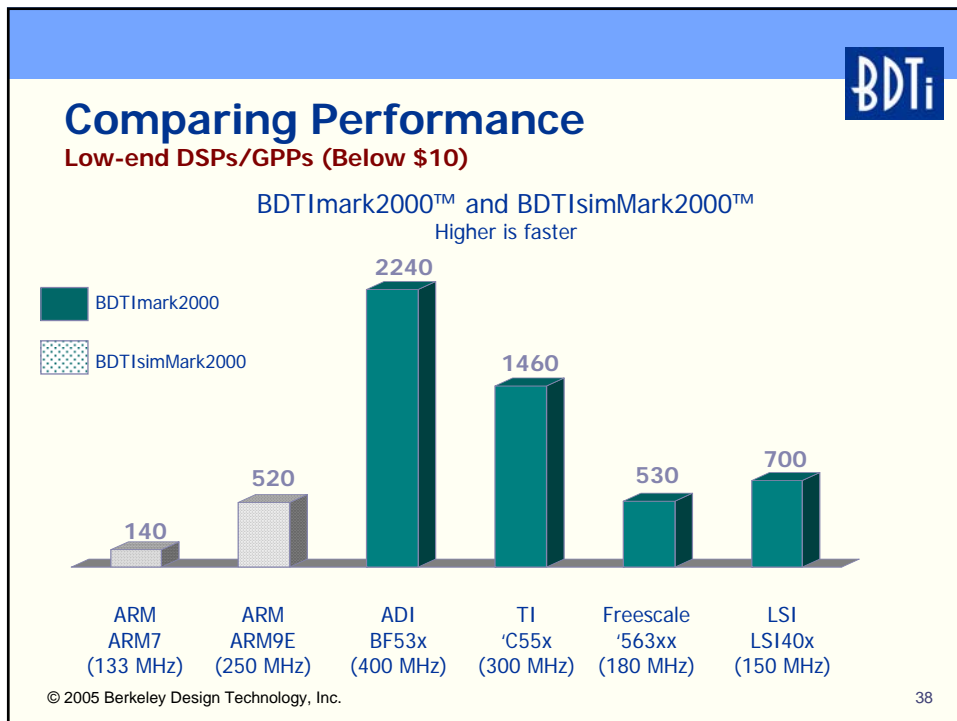
Vendor performance claims should be viewed skeptically

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

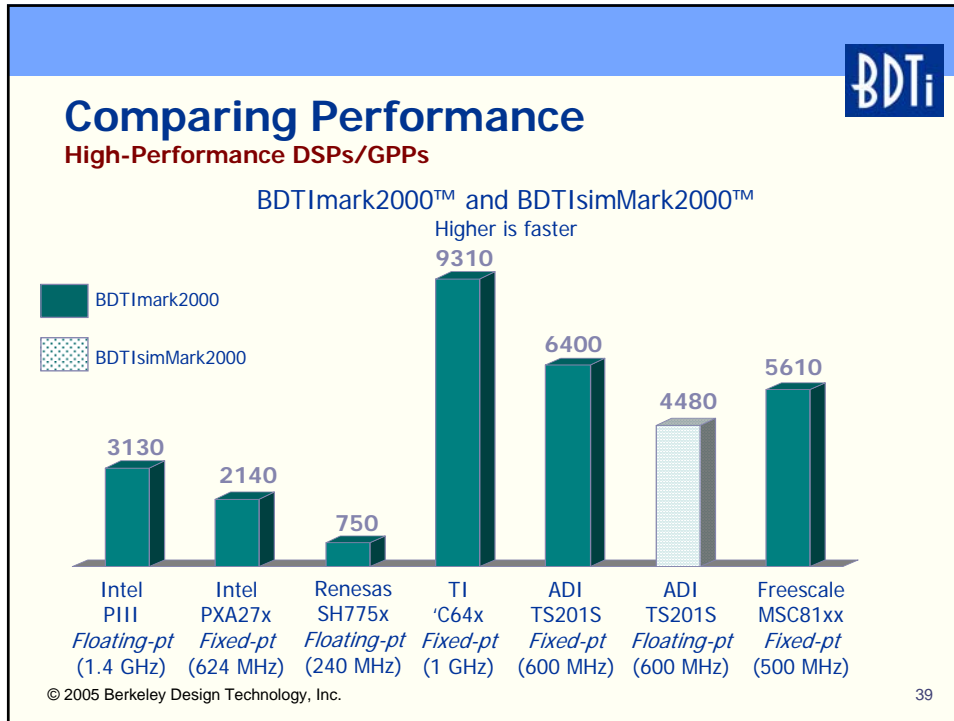
Performance is more than speed

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

© 2005 Berkeley Design Technology, Inc. 37



Microprocessors vs. DSPs: Fundamentals and Distinctions




When to Use Which

<p><u>DSP</u></p> <ul style="list-style-type: none"> • 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 	<p><u>GPP</u></p> <ul style="list-style-type: none"> • Modest signal processing requirements • Extensive control processing <ul style="list-style-type: none"> • 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
---	--

© 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

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



Sixth Edition

45

© 2005 Berkeley Design Technology, Inc.

© 2005 Berkeley Design Technology, Inc.