

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



## **Use a Microprocessor, a DSP, or Both? (ESC-348)**

Jeff Bier  
BDTI  
Oakland, California USA  
+1 (510) 451-1800

[info@BDTI.com](mailto:info@BDTI.com)  
<http://www.BDTI.com>

© 2008 Berkeley Design Technology, Inc.



### **Workshop Outline**

Definitions  
DSP algorithms shape DSPs  
Design goals for embedded DSP  
Comparing performance  
When to use which  
Conclusions

© 2008 BDTI

INSIGHT • ANALYSIS • ADVICE  
ON SIGNAL PROCESSING TECHNOLOGY

2

© 2008 BDTI

## Definitions

### Microprocessors—General-Purpose Processors (GPPs)

- 32-bit GPPs for embedded applications
  - E.g., ARM ARM7

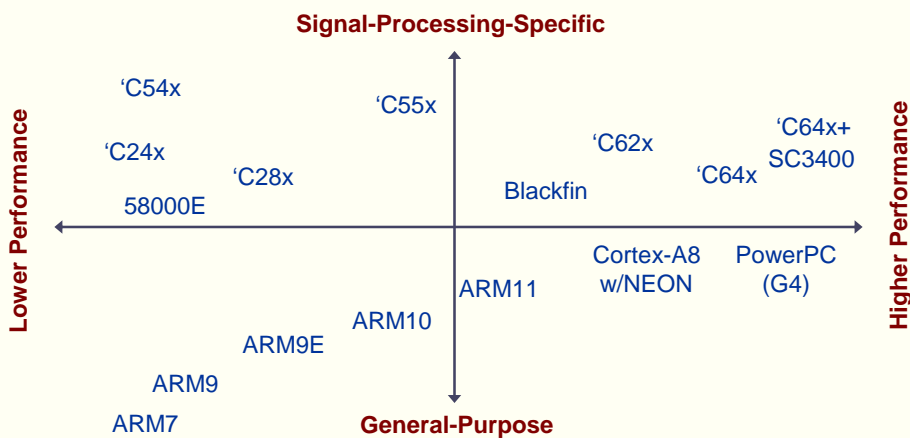
### Digital Signal Processors (DSPs)

- Microprocessors specialized for signal processing applications
  - E.g., Texas Instruments C55x+

### DSP-enhanced GPPs and hybrids

- GPPs with added DSP features, or processors designed with DSP and GPP attributes
  - E.g., MIPS MIPS24KE, Microchip dsPIC, Analog Devices Blackfin

## Example Processors





## 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

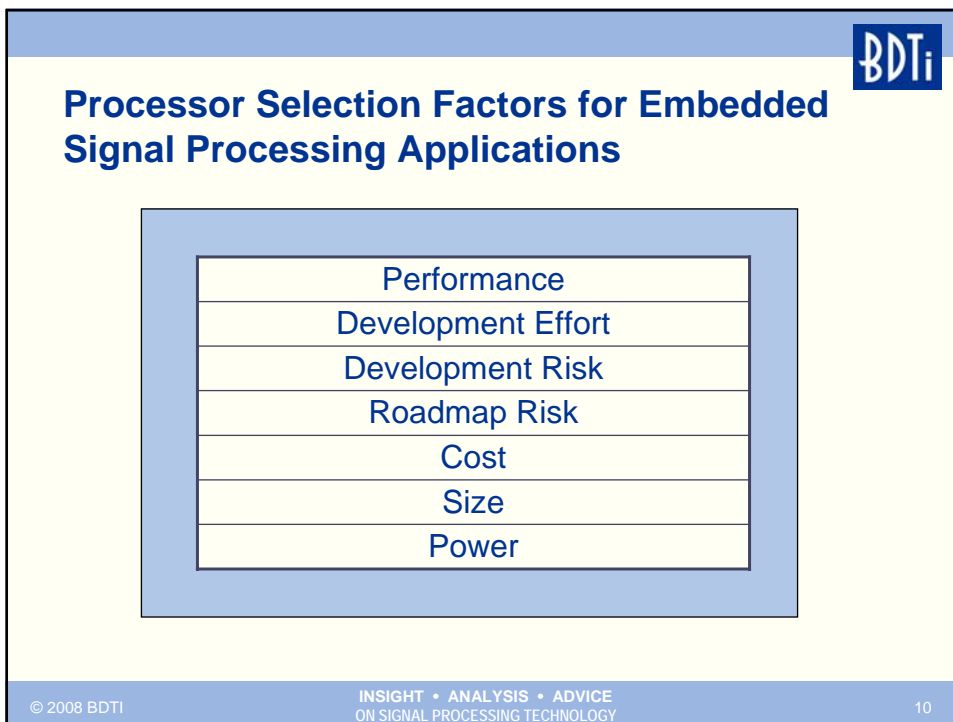
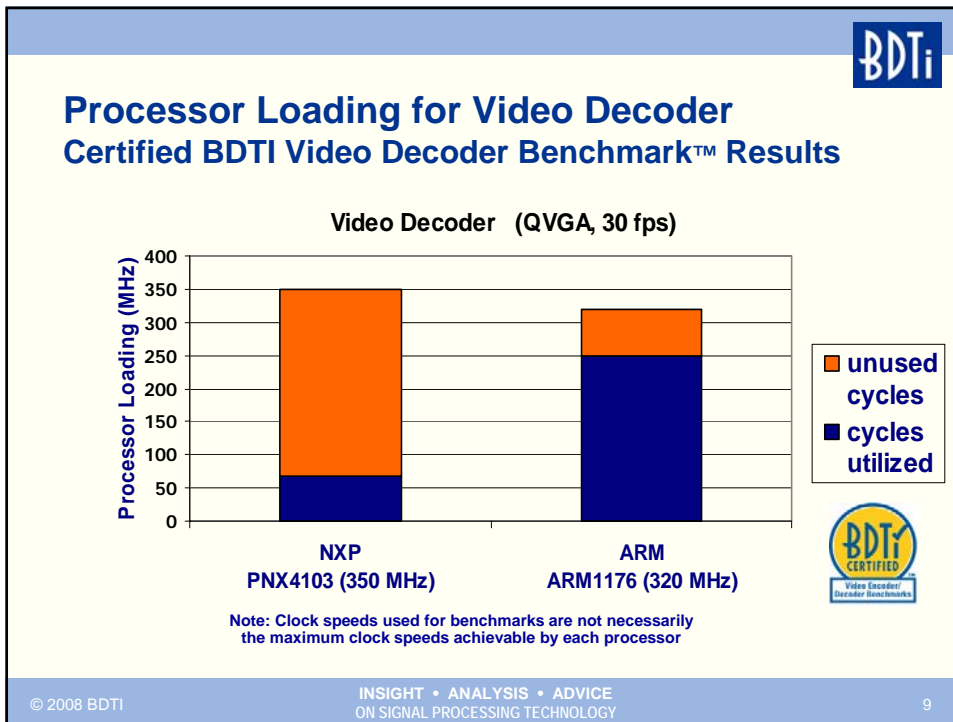
Computational demands	→	Multiple parallel execution units, hardware acceleration of common DSP functions
Numeric fidelity	→	Accumulator registers, guard bits, saturation hardware
High memory bandwidth	→	Harvard architecture, support for parallel moves
Predictable data access patterns	→	Specialized addressing 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

## Example: Video Processing

- Computational demands: high
  - Example: color conversion
    - CIF (352 by 288 pixel) , 15 fps, conversion (without any interpolation) requires over 18 million operations per second
- Numeric fidelity: 8 to 12-bit pixels
- High memory bandwidth
  - E.g., D1 video (720x480), 30 fps
    - (720\*480 pixels) (3 RGB values) (8 bits) (30 frames) = 31.1 Mbytes/second
- Highly parallelizable
- Predictable data access patterns
  - Motion estimation and compensation notable exceptions





## Processor Selection Factors for Embedded Signal Processing Applications

Performance
Development Effort
Development Risk
Roadmap Risk
Cost
Size
Power



## Performance

- Data path
  - Computational resources
  - SIMD
- Memory architecture
  - Harvard vs. Von Neumann
  - Cache vs. SRAM with DMA
- Real-time considerations
  - Non-determinism
  - Dynamic features



## Comparing DSPs and GPPs

### Data Path

#### Low-end DSP

Dedicated hardware performs all key arithmetic operations in 1 cycle

Usually 16-bit, fractional, integer

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

#### High-performance DSP

Up to 8 arithmetic units

Some specialized arithmetic units

- E.g., MAC unit, Viterbi unit

Support multiple data sizes

Limited to excellent bit-manipulation capabilities

Hardware support for managing fixed-point numeric fidelity

#### High-performance GPP

1-3 arithmetic units

General-purpose arithmetic units

- E.g., integer unit, floating-point unit

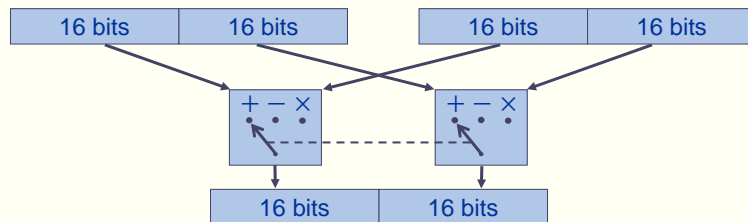
Support multiple data sizes

May have superior bit-manipulation capabilities

Saturation, rounding typically take extra cycles

## 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 & GPP

DSPs: very limited SIMD features

- E.g., dual add, subtract of 16-bit fixed-point data

GPPs: No SIMD support

#### High-performance DSP & GPP

DSPs: limited to extensive SIMD features

- E.g., TigerSHARC
  - 4 x 32-bit float
  - 4 x 32-bit integer
  - 8 x 16-bit integer
  - 16 x 8-bit integer

GPPs: extensive SIMD features

- E.g., PowerPC 74xx
  - 4 x 32-bit float
  - 4 x 32-bit integer
  - 8 x 16-bit integer
  - 16 x 8-bit integer





## Comparing DSPs and GPPs SIMD Features

### DSP-enhanced GPP

Moderate to extensive SIMD features

- E.g., ARM1136J-S
  - 1 x 32-bit integer
  - 2 x 16-bit integer
  - 4 x 8-bit integer



## Memory Structure

- Harvard vs. Von Neumann
  - Harvard – separate memories for data and instructions
  - Von Neumann – single memory for data and instructions
- Bandwidth between processor and on-chip memory
- Size of on-chip memory
  - Larger memory is better for performance, but hurts cost and increases power
  - Fetching data from external memory consumes cycles and power
- Memory control
  - Caches
  - SRAM with DMA



## Comparing DSPs and GPPs

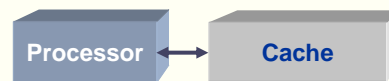
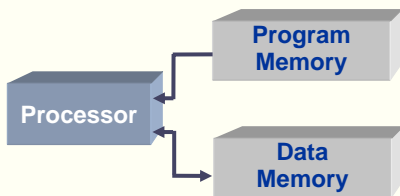
### Memory Architecture

#### 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)



© 2008 BDTI

INSIGHT • ANALYSIS • ADVICE  
 ON SIGNAL PROCESSING TECHNOLOGY

19



## Comparing DSPs and GPPs

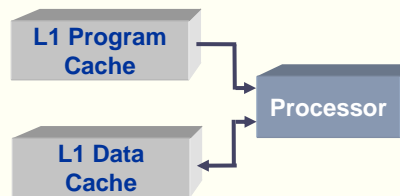
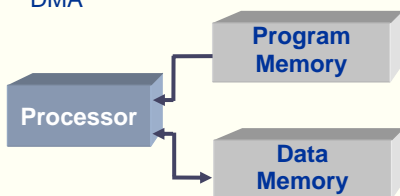
### Memory Architecture

#### 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  
 Use caches



© 2008 BDTI

INSIGHT • ANALYSIS • ADVICE  
 ON SIGNAL PROCESSING TECHNOLOGY

20

© 2008 BDTI

## Real-Time Considerations

- Performance
  - Can the processor handle the load?
- Non-determinism
  - Non-determinism causes load variance
  - Complicates optimization and debugging
  - Caused by:
    - Dynamic processor features
    - Data-dependent algorithm behavior
    - Multi-tasking

## Comparing DSPs and GPPs

### Dynamic Features

Dynamic features are common 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



## Comparing DSPs and GPPs

### Dynamic Features

#### Low-end GPPs and DSPs

##### GPPs:

- Dynamic caches common

##### DSPs:

- Rarely have dynamic features

#### High-performance GPPs and DSPs

##### GPPs: Moderate to extensive use of dynamic features

- Dynamic caches standard
- Superscalar execution, branch prediction common

##### DSPs: Mostly avoid dynamic features

- Cache is most common dynamic feature
- Superscalar execution rare
- Branch prediction sometimes used



## 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

- E.g., many “hard-real-time” signal processing applications

Signal processing access patterns are often predictable

- Thus, DMA may be preferable to a cache
- Some caches provide pre-fetching capability
- Some DSPs’ caches can be locked or configured as part cache, part SRAM



## 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

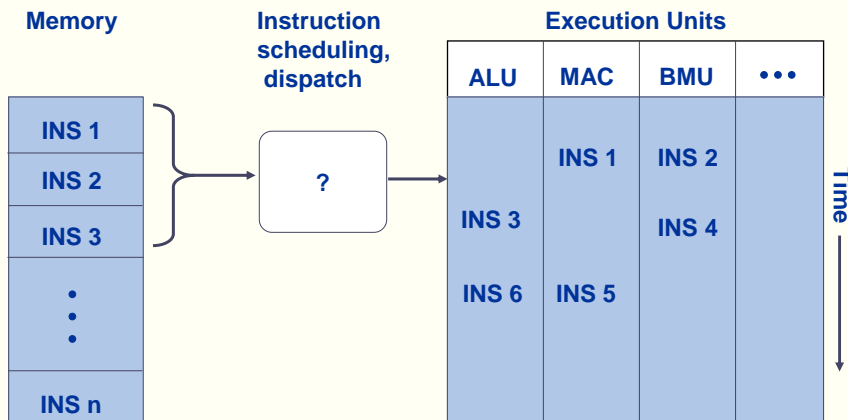
Complex branch prediction algorithms introduce timing uncertainty

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



## Multi-Issue Approaches

### VLIW vs. Superscalar





## Comparing DSPs and GPPs

### Trade-offs: Superscalar vs. VLIW

Superscalar (high-performance GPPs, mostly)

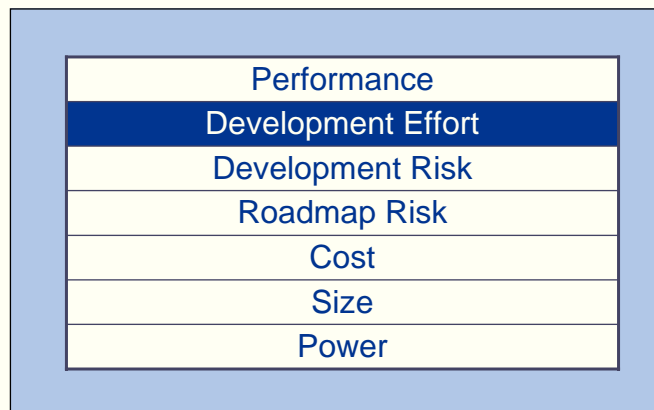
- Increased hardware complexity
  - Silicon area, power consumption
- Dynamic behavior
  - Complex performance model, timing variability
- Increased performance with binary compatibility
- Decreased software complexity (programmer/compiler)

VLIW (high-performance DSPs, mostly)

- Decreased hardware complexity
- No dynamic behavior
- Binary compatibility difficult (downward direction)
- Increased software complexity



## Processor Selection Factors for Embedded Signal Processing Applications





## Development Effort

### Compiler friendliness

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

### Development support

- DSPs have more 3<sup>rd</sup> party DSP-oriented IP, DSP-oriented tools
- GPPs have better non-DSP-oriented support



## 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,x5
mov (r0),r2
mov (r1),r3
inc r0
inc r1
```



## Comparing DSPs and GPPs

### Instruction Set

#### 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



## Comparing DSPs and GPPs

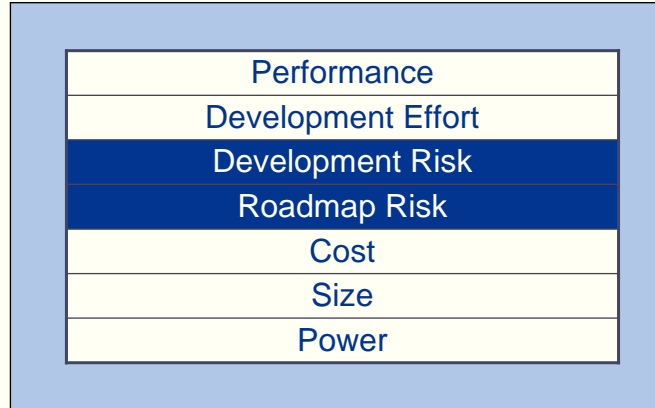
### Development Support

	DSPs	GPPs
<b>Tools in general</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>	Limited but growing 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





## Processor Selection Factors for Embedded Signal Processing Applications



## Comparing DSPs and GPPs Compatibility and Availability

### Low-end DSP

Mostly proprietary architectures

- I.e., one architecture, one vendor

Varying compatibility between successive generations

Rarely available as licensable core

### Low-end GPP

Many shared architectures

- I.e., one architecture, several (to many) vendors

Often binary compatibility between successive generations

Often available as licensable core

- E.g., ARM, MIPS



## Comparing DSPs and GPPs

### Compatibility and Availability

#### High-performance DSP

Mostly proprietary architectures

Sometimes binary compatibility between successive generations

- E.g., 'C6xxx

Rarely available as licensable core

- E.g. CEVA-X

#### High-performance GPP

Mostly shared architectures

- PowerPC, MIPS, ARM, x86

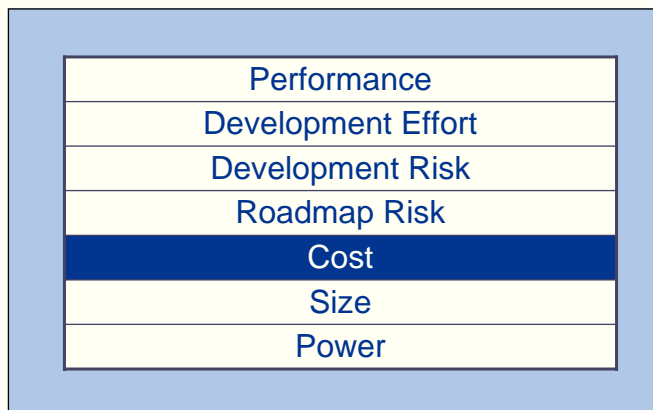
Usually binary compatibility between successive generations

Sometimes available as licensable core

- E.g., ARM, MIPS



## Processor Selection Factors for Embedded Signal Processing Applications



## Cost

- Hardware cost
  - System cost
    - Chip cost
    - On-chip integration
      - Fewer components may lower component costs
  - SoC cost
    - Die size
      - Dynamic features use silicon (e.g., superscalar vs. VLIW)
    - Royalties

## Comparing DSPs and GPPs

### On-Chip Integration

#### Low-end GPPs and DSPs

Typically, wide range of on-chip peripherals and I/O interfaces

Often oriented towards consumer applications

- E.g., USB, serial ports, I<sup>2</sup>S, GPIO, ...

#### High-performance GPPs and DSPs

Moderate to extensive on-chip integration

- PC CPUs offer very little on-chip integration

Often oriented towards specific applications

- Viterbi decoding coprocessors, UTOPIA ports, ...



## Processor Selection Factors for Embedded Signal Processing Applications

Performance
Development Effort
Development Risk
Roadmap Risk
Cost
<b>Size</b>
Power



## Processor Selection Factors for Embedded Signal Processing Applications

Performance
Development Effort
Development Risk
Roadmap Risk
Cost
Size
<b>Power</b>

## Power

- Parallelism
  - Parallel computation allows lower clock rate...
  - But may increase leakage current
- Suitability of instruction set
  - A better matched instruction set allows lower clock rate
- Dynamic features
  - Caches may cause data/instruction traffic increase
  - Superscalar – hardware scheduling consumes power
- On-chip integration
  - Memory architecture
    - Fetching data from external source expensive
  - Smart peripherals aid parallelism, may enable more processor sleep time
- Power-management features

## Comparing Performance

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

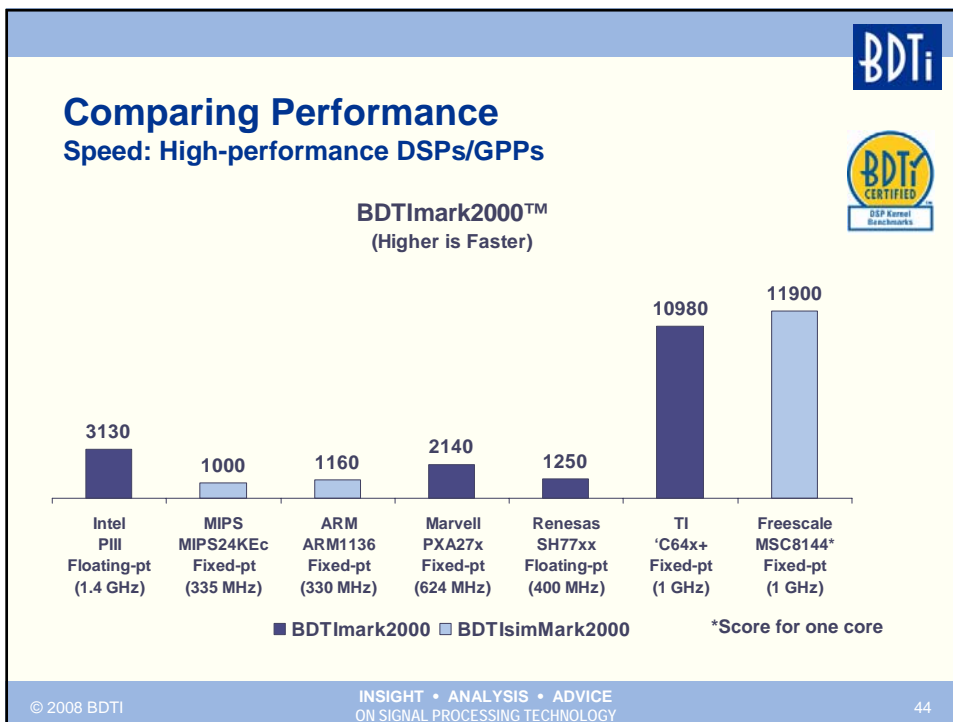
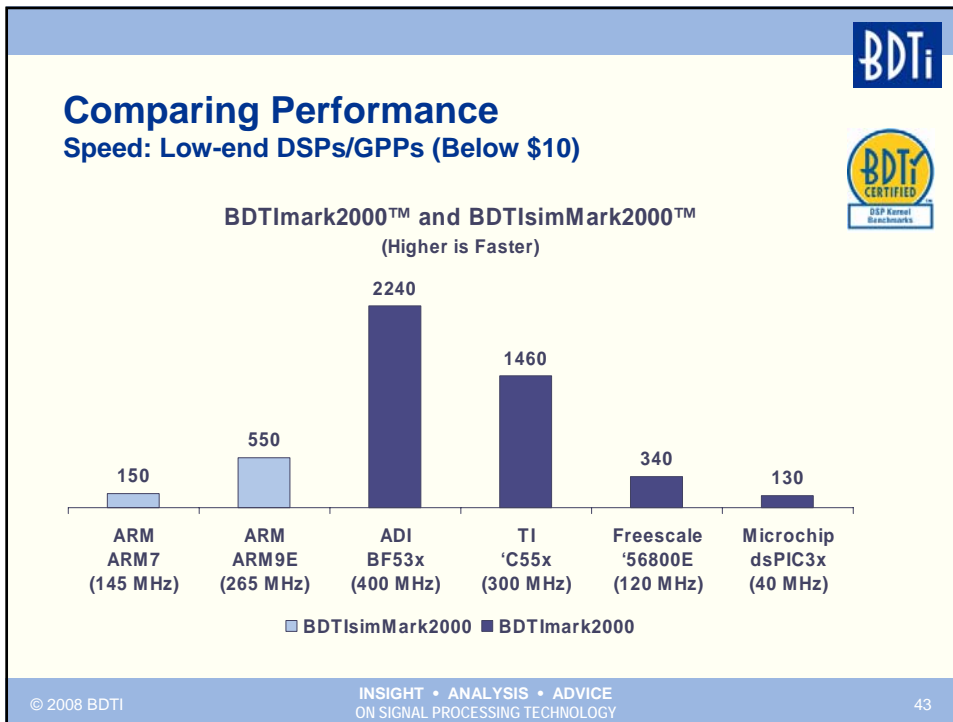
- Relative performance can vary dramatically depending on the benchmark

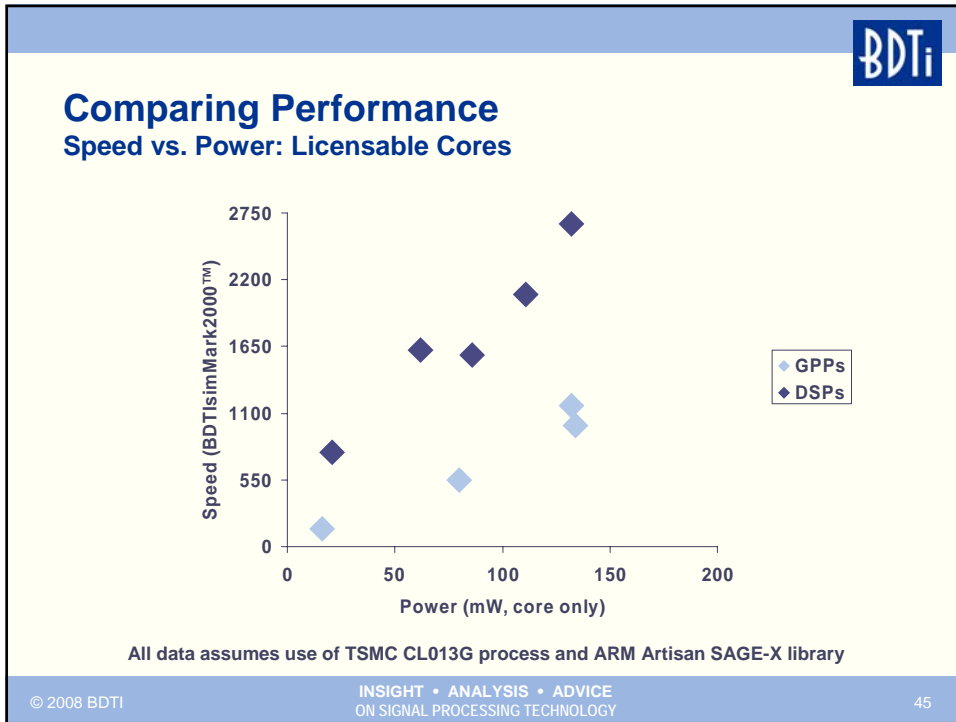
Vendor performance claims should be viewed skeptically

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

Performance is more than speed

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





- 
- ## When Should You Consider a DSP?
- You need maximum performance or efficiency on a signal-processing-heavy workload
  - You have compatible software you want to re-use
  - Your developers are already familiar with it
  - You need limited non-signal-processing software
  - You'll be developing demanding DSP software
  - A DSP offers good off-the-shelf software for your application
  - You don't need a full-featured operating system
  - You need maximum execution-time determinism
  - A DSP offers superior integration
- © 2008 BDTI      INSIGHT • ANALYSIS • ADVICE ON SIGNAL PROCESSING TECHNOLOGY      46



## When Should You Consider a GPP?

- A GPP offers sufficient performance and efficiency on your signal-processing-heavy workload
- You have compatible software you want to re-use
- You want to be able to switch vendors but not ISAs
- Your developers are already familiar with it
- You need extensive non-signal-processing software
- You won't be developing much DSP software
- A GPP offers good off-the-shelf software for your application
- You need a full-featured operating system
- Execution-time determinism is not critical
- A GPP offers superior integration



## Can I Have the Best of Both Worlds?

Maybe.

Options include:

- Two processors
  - One or two chips
  - But: Cost; multiprocessor software development
- DSP-enhanced GPP
  - But: Typically compromise on DSP-oriented tools, software, integration
- Hybrid
  - But: Typically compromise on GPP-oriented tools, software
- Media processors / application processors
  - But: Tend to be focused on multimedia applications





## For More (Free!) Information...

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

[www.INSIDEdsp.com](http://www.INSIDEdsp.com)

*Inside DSP* newsletter

benchmark scores for dozens of processors, FPGAs, video solutions, etc.

*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

