Independent DSP Benchmarks: Methodologies, Results, and Analysis

Berkeley Design Technology, Inc.
2107 Dwight Way, Second Floor
Berkeley, California U.S.A.

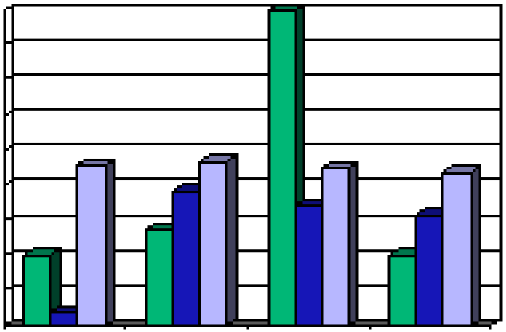
+1 (510) 665-1600
info@BDTI.com
http://www.BDTI.com

Outline

- Motivation for benchmarking
- DSP benchmarking approaches--pros and cons
- DSP benchmarks: what's available
- Benchmark performance of example processors
- The BDTImark: what is it?
- Factors influencing benchmark results
- DSP benchmarking for general-purpose processors
- Conclusions

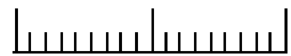
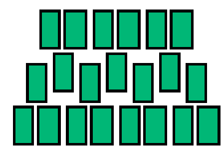
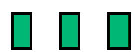
Motivation for Benchmarking

- Need quick and accurate comparisons of processors' DSP performance
- As architectures diversify, it becomes more difficult to compare performance
- There is a need for accurate comparisons of processors' DSP performance



DSP Benchmarking Approaches

There are a number of DSP benchmarking approaches. The main candidates are:

- Simplified metrics (MIPS, MOPS, etc) 
- Complete DSP applications 
- DSP algorithm "kernels" 

What's Wrong with MIPS?

Why not rely on MIPS, MOPS, MACs/sec, MFLOPS...?

These metrics are simple and easy to measure, but can be misleading. Questions to ponder:

- Just what is an "instruction" or "operation"? (or, when is 100 MIPS faster than 120 MIPS?)
- What's included in a MAC, and what if my application does something besides MACs?

BDTi

Benchmarking Full Applications

Why not just use a full DSP application, like a V.90 modem or AC-3 decoder?

This approach is common in PC systems (e.g., SPEC) but is not appropriate for DSP benchmarking because:

- Applications tend to be ill-defined
- Hand-optimization usually required
  - Costly, time-consuming to implement
  - Evaluates programmer as much as processor
- Measures *system*, not just processor

BDTi

What's an Algorithm Kernel?

- DSP algorithm kernels are the most computationally intensive portions of DSP applications.

- Example algorithm kernels include FFTs, IIR filters, Viterbi decoders, etc.

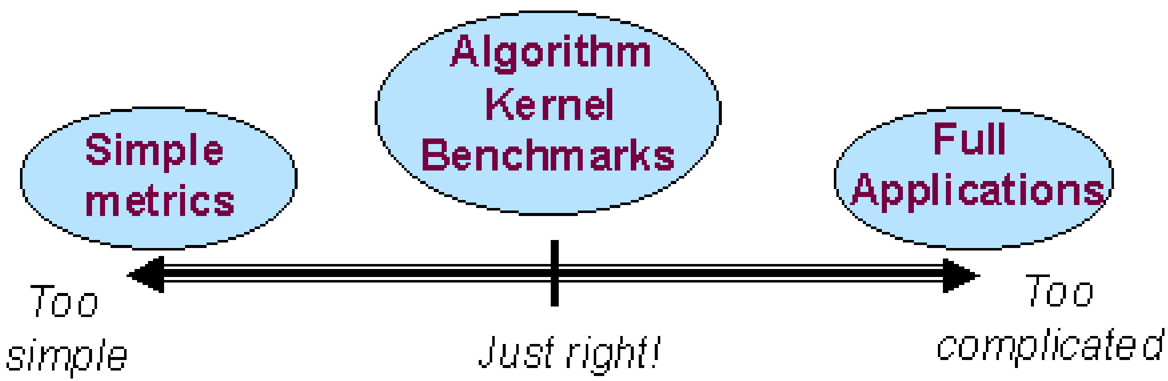
Application-relevant algorithm kernels are strong predictors of overall performance.

BDTi

Why Use Algorithm Kernels?

Algorithm kernels are good benchmark candidates because they are:

- Relevant
- Practical to specify and implement
- Relatively simple to optimize



Drawbacks of Algorithm Kernel Benchmarks

- Completeness
  - Limited number of algorithm kernels; may not include all functions relevant to your application
- System design issues mostly ignored
  - e.g., performance degradation if program won't fit in on-chip memory

Other Considerations

- Comparing benchmark results for processors with different data word sizes can be misleading
  - e.g., 24-bit data word provides better accuracy than 16-bit data word
- Comparing fixed-point results to floating-point results can be misleading
  - Floating-point provides better precision...
  - ... but AD and DA converters use fixed-point
  - Meeting bit-exact standards may require extra work on floating-point processors

BDTi

Other Considerations

- Understanding why processors perform as they do is often critical
  - For judging applicability of results
  - For understanding architectural strengths and weaknesses
  - For estimating whole-application performance

DSP Benchmark Landscape

- **Vendor benchmarks**
  - Most DSP processor vendors provide DSP benchmark results for their own processors and selected competitors.
  - Benchmarks are generally not standardized across vendors.
  - Results are not independently verified.
- **EEMBC** (EDN Embedded Microprocessor Benchmark Consortium)
  - Consortium of semiconductor and IP vendors formed in 1998.
  - Uses algorithm kernel benchmarks divided by application area (telecom, automotive, etc.)
  - Vendors implement benchmarks, EEMBC verifies results.
  - Benchmarks implemented in C and optimized assembly.
  - Results publicly available.

DSP Benchmark Landscape

- **BDTI**
  - Independent DSP technology analysis and software development firm that developed proprietary set of DSP algorithm kernel benchmarks in 1994.
  - Implements and/or verifies benchmarks in-house.
  - Benchmarks implemented in optimized assembly following specification.
  - Provides analysis of results; results and analysis available in published reports.
  - Composite speed score ("BDTImark") publicly available.

BDTI Benchmarking Methodology

- Benchmarks are rigorously defined
- All implementations follow the same rules
- Benchmarks are hand-optimized in assembly
- Each benchmark is independently verified for
  - Performance
  - Functionality
  - Optimality
  - Conformance to benchmark specs
- Benchmarks use processor's native data format

BDTi

BDTI Benchmarking Methodology

- Benchmarks are optimized for speed, then memory usage (except control-oriented benchmark, which is the other way around)
- BDTI's benchmarks reveal realistic performance, not necessarily fastest possible performance
- Benchmarks are architecture-independent; can be implemented on any processor (including general-purpose processors)

BDTi

BDTI Benchmark™ Suite

Composed of a wide variety of DSP algorithm kernels.

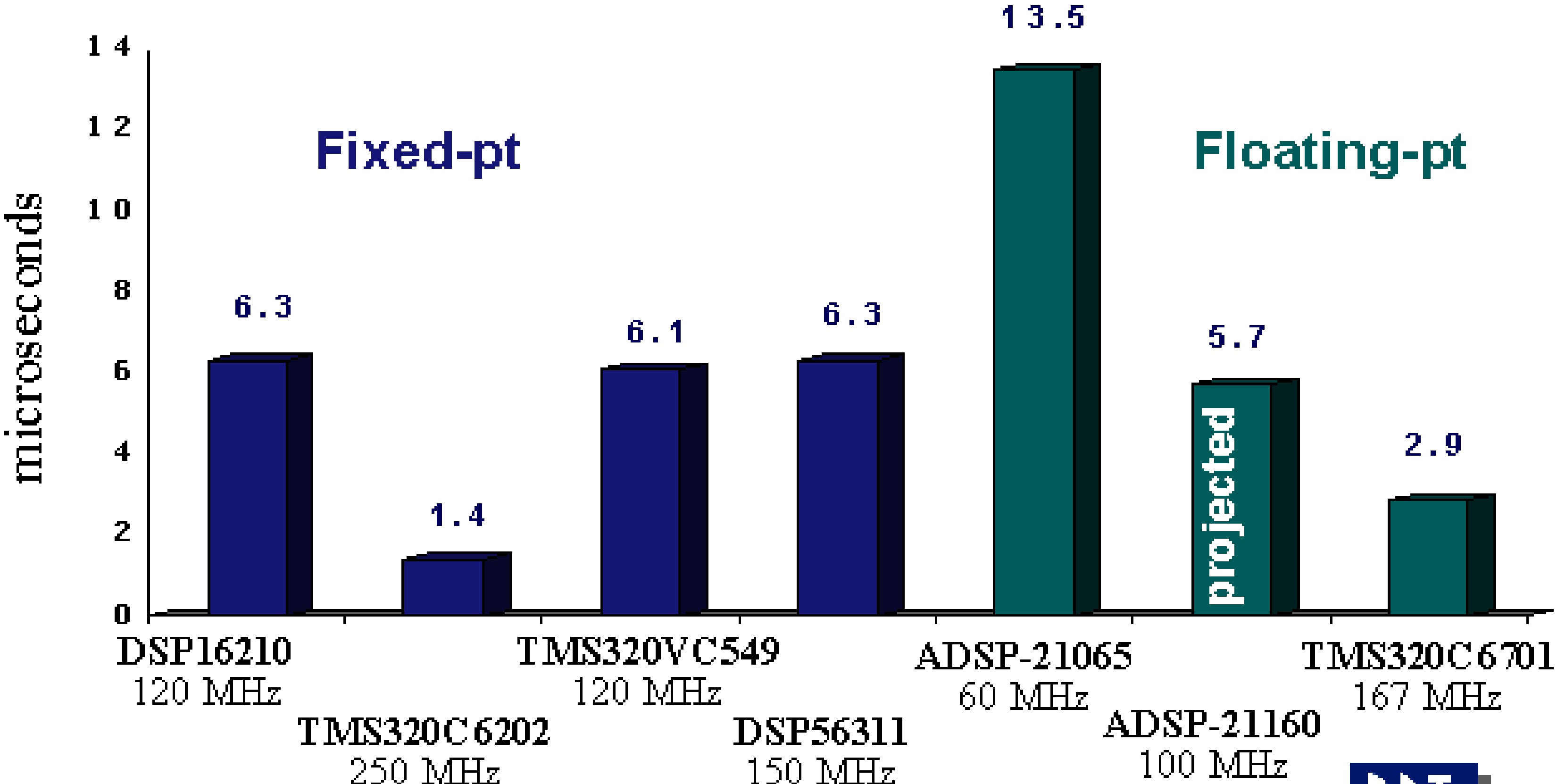
On each benchmark, we measure five quantities:

- Cycle count
- Execution time
- Cost-performance
- Energy Consumption
- Memory use

*All benchmark results in this presentation are taken from BDTI's reports, *Buyer's Guide to DSP Processors 1999 Edition* and *DSP on General-Purpose Processors*

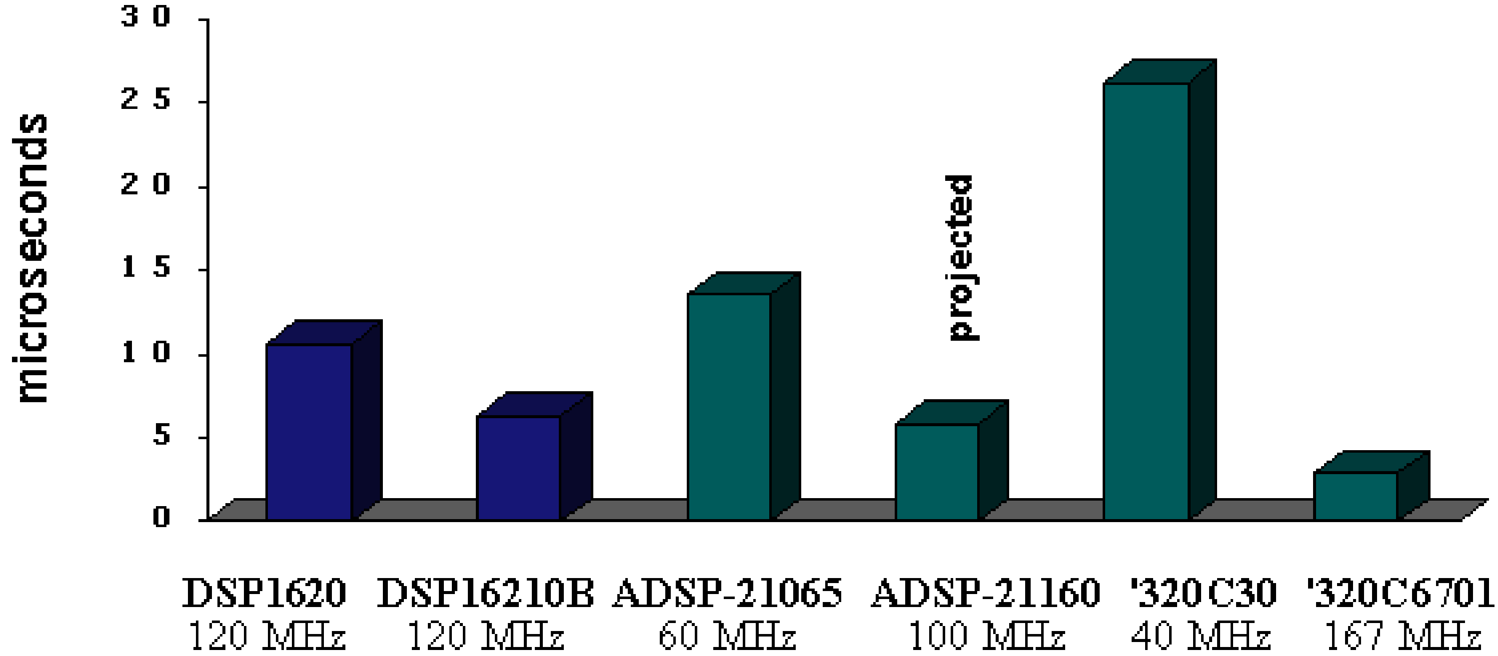
Execution Times

FIR Filter Benchmark

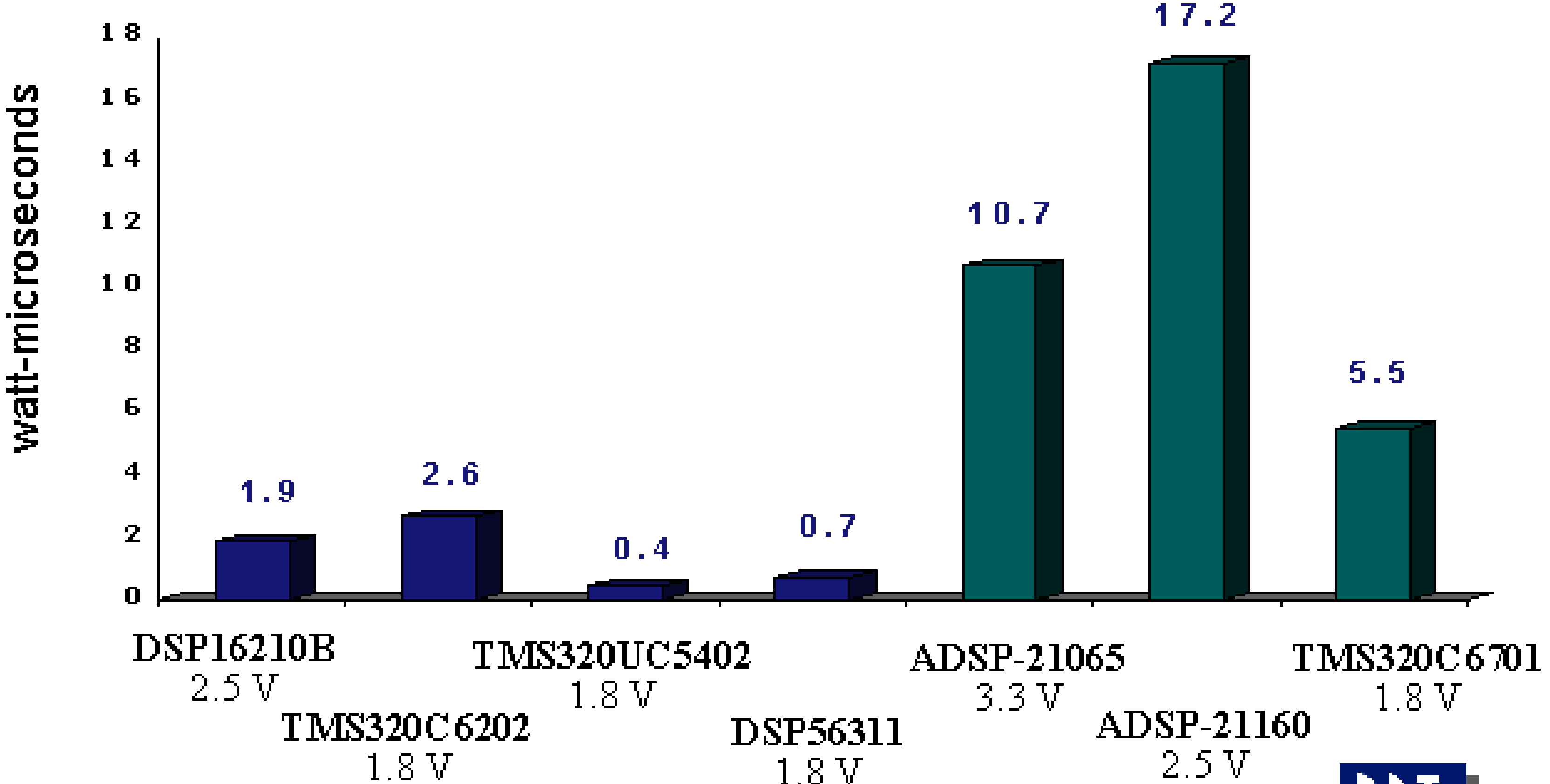


Execution Times
FIR Filter Benchmark

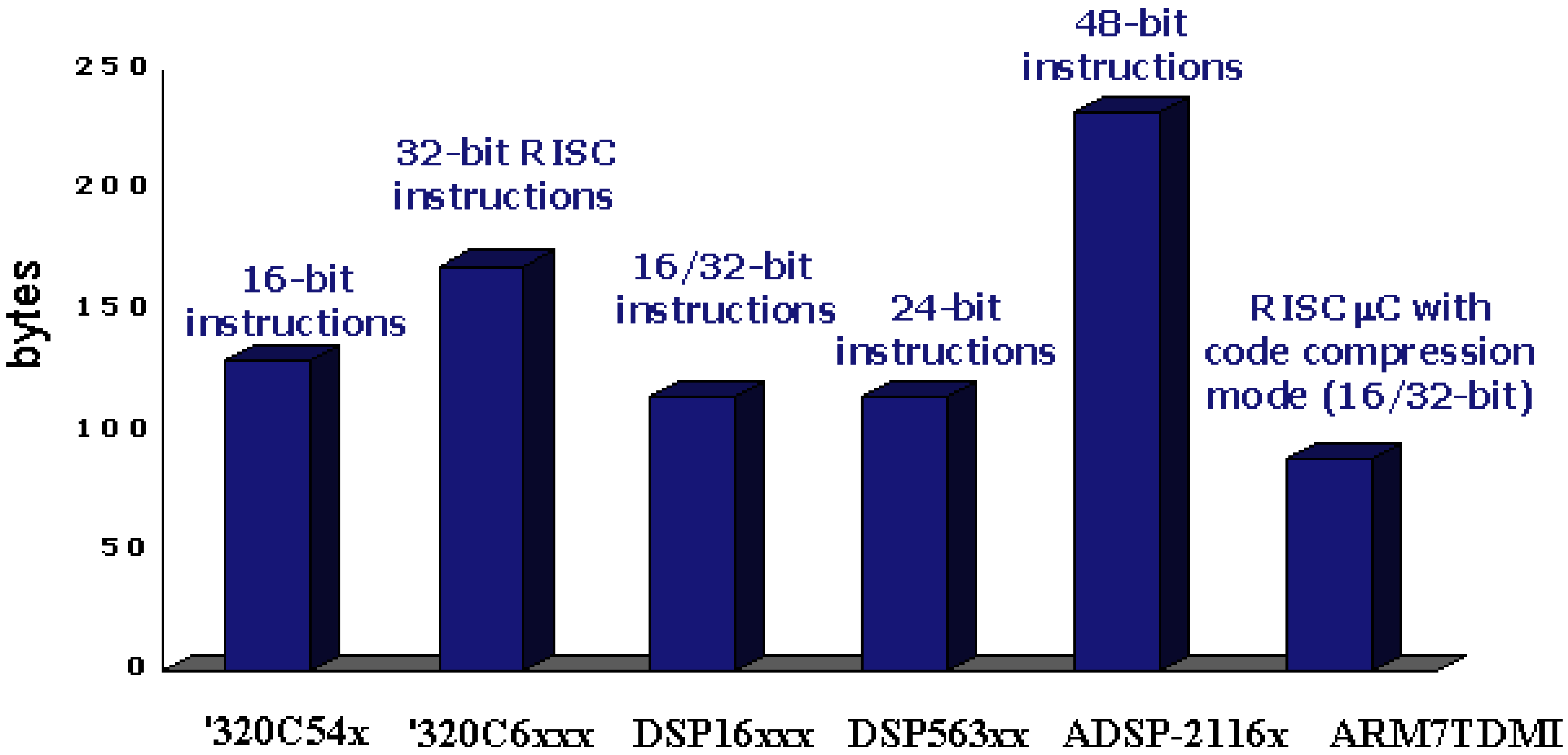
Performance improvements in new generations



Energy Consumption
FIR Filter Benchmark

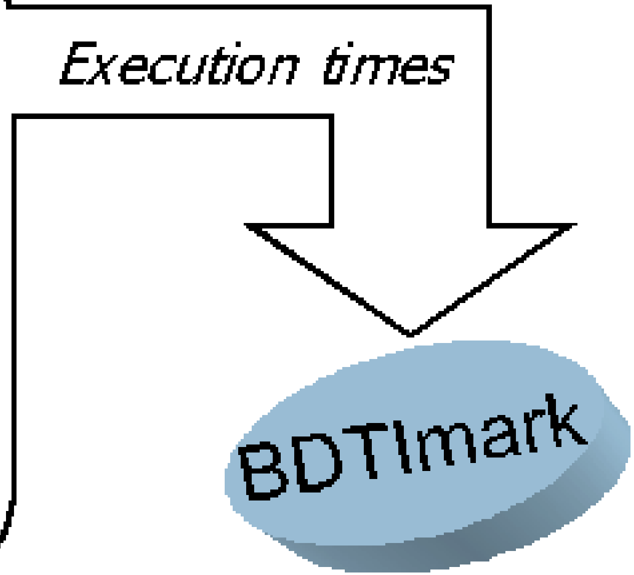


Memory Usage: FSM Benchmark



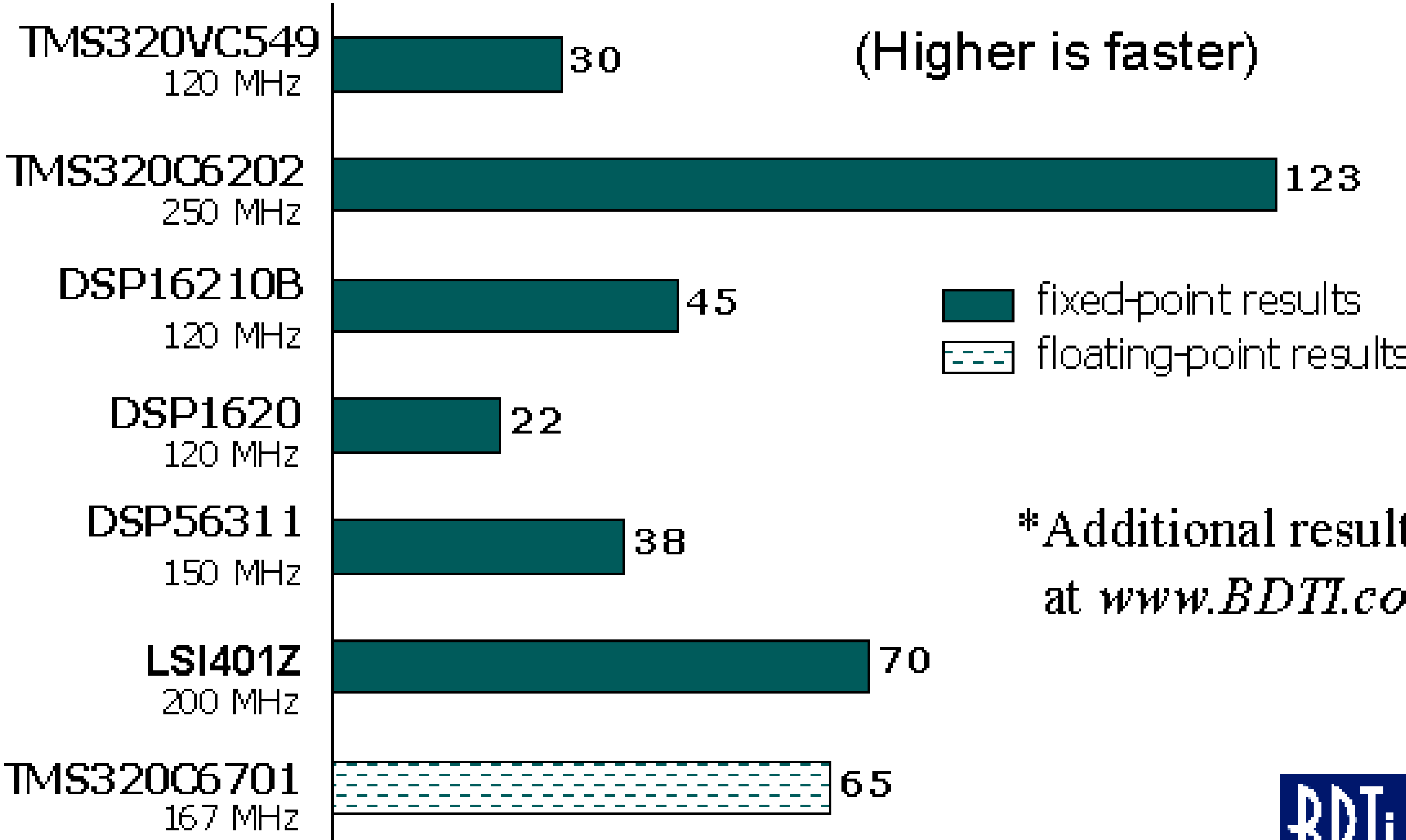
The BDTImark™

- Real block FIR filter
- Complex block FIR filter
- Single-sample real FIR filter
- Single-sample LMS-adaptive FIR filter
- Single-sample IIR filter
- Vector dot product
- Vector add
- Vector maximum
- IS-54 convolutional encoder
- Finite state machine
- 256-point FFT



Note: BDTI is currently updating its benchmark suite.

Example BDTImark™ Results*



*Additional results at *www.BDTI.com*

What Factors Influence Benchmark Results?

Factors

- Parallel execution units
- VLIW
- Superscalar
- SIMD capabilities
- Instruction-word size
- Data-word size
- RISC-like instructions vs complex, compound instructions
- Memory bandwidth
- Pipeline
- Hardware accelerators

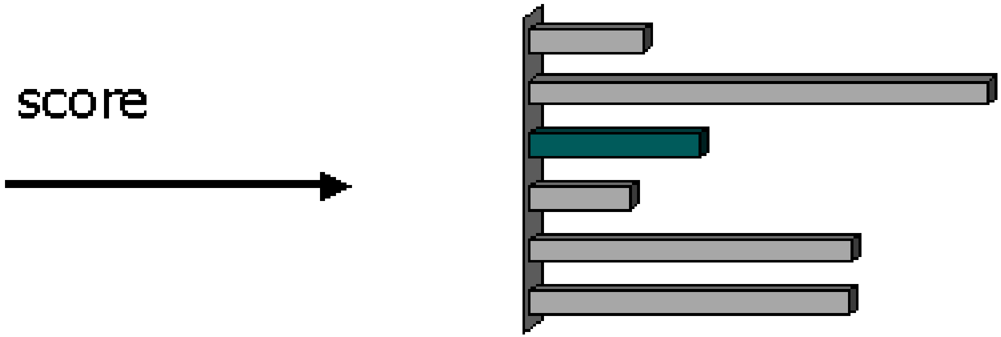
- Clock speed

Case Study: The DSP16xxx

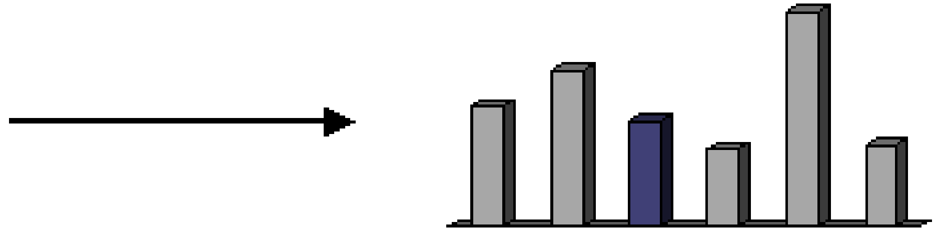
- Traditional DSP architecture, but with major additions
- Dual multipliers, wider memory buses yield 2 MACs/cycle
- Complex instructions, restrictions on parallel operations and register usage
- Simple pipeline

The DSP16210

- Good BDTImark score



- Moderate memory usage



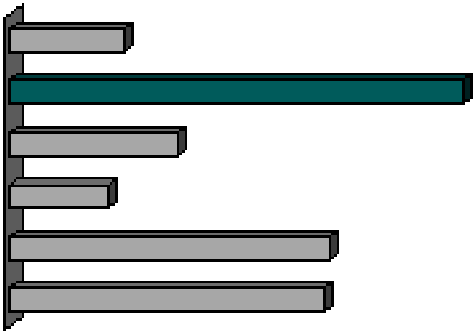
- Moderate energy consumption

Case Study: The TMS320C62xx

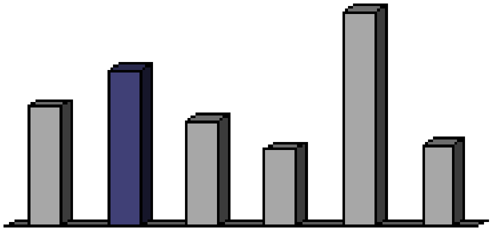
- Radical new VLIW-like architecture
- Simple, RISC-like instructions with few restrictions
- 8 execution units (including 2 multipliers and 4 ALUs) produce 2 MACs/cycle
- Deep, complicated pipeline

The TMS320C6201

- Excellent BDTImark score



- High memory usage



- Moderate energy consumption

GPPs for DSP

High-End GPPs for DSP

Today's high-end general-purpose processors outperform many DSPs *even on DSP applications.*

Why?

- Blazing clock speeds
- Superscalar execution
- Branch prediction, speculative execution
- Integrated DSP-oriented features

Drawbacks of High-End GPPs

Even when their performance is competitive, high-end GPPs don't usually replace DSPs because of:

- Unpredictable execution times
- Poor cost-performance relative to fixed-point DSPs
- High energy consumption
- A lack of DSP-oriented development tools
- Integration difficulties

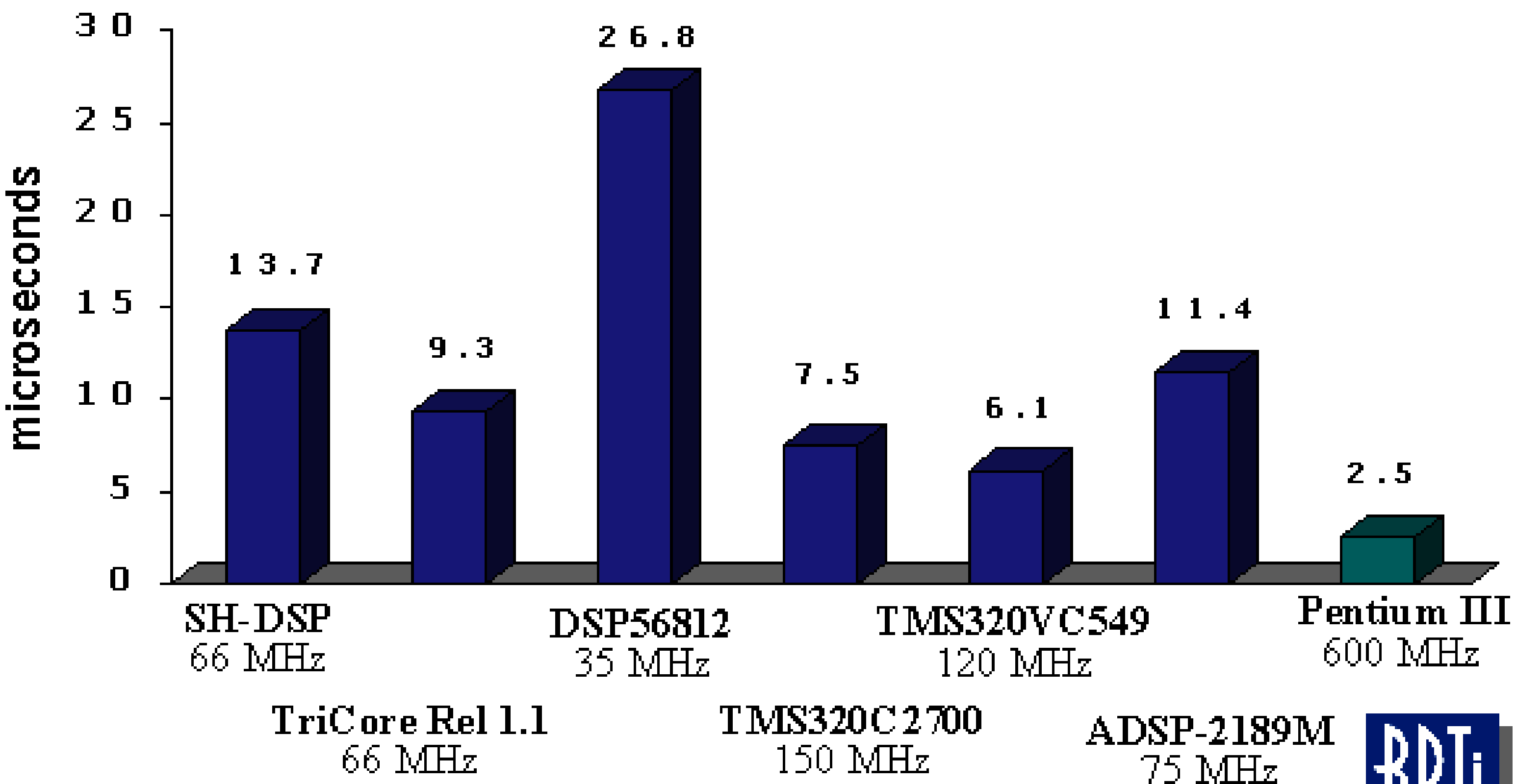
If a high-end GPP is incumbent, it may make sense to use it for DSP work. Otherwise, it's often better to use a DSP.

Embedded GPPs for DSP

- GPPs for embedded applications are starting to address DSP needs
  - Hitachi SH-DSP, ARM9E, Infineon TriCore
- These processors achieve reasonable DSP performance while maintaining relatively low cost and low energy consumption
- Embedded GPPs typically don't have the advanced features that affect execution time predictability, so are easier to use for DSP

Execution Times

FIR Filter Benchmark



Conclusions

- Rigorous benchmark specs are essential
- The "best" processor depends on the application
- The fastest processor for a DSP task may not be a DSP
- Metrics other than execution speed may be most important
- Benchmarks don't tell the whole story

BDTi

Recent Developments

- New Benchmarks
  - New FFT
  - Control - replaces FSM
  - Bit unpacking - replaces convolutional encoder
  - Viterbi decoder

BDTi

Work in Progress

- Work on New Processors

StarCore SC140	(Motorola/Lucent)
TigerSHARC	(Analog Devices)
Teak	(DSP Group)
Palm	(DSP Group)
Carmel	(Infineon - formerly Siemens)
Alpha 21264	(Compaq/Digital)
Pentium III	(Intel)
PowerPC G4	(Motorola)

Check www.BDTI.com

- Slides for this talk will be published on www.BDTI.com
- Check Web site for benchmark results for latest processors (results unavailable for class handouts)

BDTi

For More Information...

Free resources on BDTI's web site,

http://www.bdti.com

- *Evaluating DSP Processor Performance,* a white paper from BDTI
- *DSP Processors Hit the Mainstream* originally printed in IEEE Computer Magazine
- Numerous other BDTI article reprints, slides
- *comp.dsp* FAQ
- BDTImark scores