Independent DSP Benchmark Results for the Latest Processors

Berkeley Design Technology, Inc.
+1 (510) 665-1600
info@BDTI.com

www.BDTI.com

1

Outline

- Motivation for benchmarking
- DSP benchmarking approaches–pros and cons
- DSP benchmarks: what's available
- Implementing DSP on general-purpose processors
- Benchmark results
- Conclusions

2

DSP Benchmarking Approaches

Candidate approaches:

- Simplified metrics
  - e.g., MIPS (Millions of Instructions Per Second)
- Complete DSP applications
  - e.g., v.90 modem
- DSP algorithm "kernels"
  - e.g., FIR filter, FFT

What's Wrong with MIPS?

MIPS and MFLOPS (Millions of Floating-Point Operations Per Second) are frequently used as shorthand for processor speed. But are they really meaningful?

Two instructions from different processors:

DSP16210

```
A0=A0+P0+P1 P0=Xh*Yh P1=Xl*Yl Y=*R0++ X=*PT0++
```

TMS320C6201

```
ADD   .L1 A0,A3,A0
```


What's Wrong with MMACS?

MMACS (Millions of Multiply-Accumulates per Second) is somewhat more relevant than MIPS, but doesn't answer some key questions:

- Do MACs include associated data moves and pointer updates?
- What other functions can be performed in parallel with MACs?
- What is the performance on DSP software that isn't MAC-intensive?
- If the processor supports multiple parallel MACs, can the algorithms of interest use them?

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 in assembly language usually required in real-world applications
  - Costly, time-consuming to implement
  - Evaluates programmer as much as processor
- Measures *system*, not just processor

What's an Algorithm Kernel?

```
Initialization
Decision-making operations
```

Lots of instructions, but not much processing time

```
start loop
    Instruction 1
    ⋮
    Instruction n
end loop
```

Most of the processing time is spent in sections like this; this is a "kernel." (May contain nested loops.)

```
Decision-making operations
```

 7

Why Use Algorithm Kernels?

- They provide good predictions of processors' performance in real DSP applications
  - Used in conjunction with *application profiling*
- Relevant
  - Can choose a range of kernels from common DSP applications; focus on time-intensive portions
- Practical to specify and implement
  - Small enough to allow assembly implementations; feasible to benchmark many different processors
- Relatively simple to optimize

 8

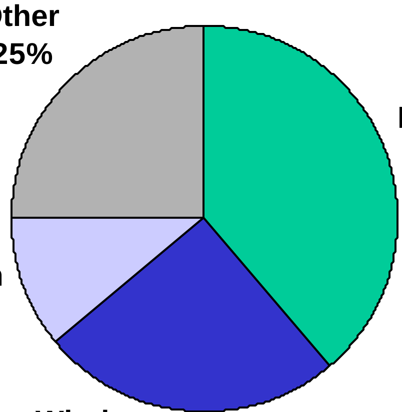
Quick Comparison of Performance: Which is "Right"?

Metric	TI TMS320C6202	TI TMS320C549	Ratio
MHz	250	120	~2 : 1
MIPS	2000	120	~17 : 1
MMACS	500	120	~4 : 1
# of Pins	384	144	~2.7 : 1

Sample Benchmark Results

Benchmark	Speed ratio 'C6202 (250 MHz) : 'C549 (120 MHz)
IIR Filter	2.2 : 1
256-point FFT	8.7 : 1
Viterbi decoder	3.7 : 1

Example Application Profile
AC-3 Audio Decoder



 11

DSP Benchmark Landscape

- **Vendor benchmarks**
  - Many processor vendors provide DSP algorithm kernel benchmark results for their own processors
  - Benchmarks generally not standardized across vendors
  - Results 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.), including some DSP functions
  - Vendors implement benchmarks, EEMBC verifies results
  - Benchmarks implemented in standard C and (optionally) optimized (possibly using assembly code)
  - Results publicly available

 12

DSP Benchmark Landscape

- **BDTI**
  - Uses its own, proprietary set of DSP algorithm kernel benchmarks
  - Implements and/or verifies benchmarks in-house
  - Benchmarks implemented in optimized assembly following specification document
  - Summary results freely available on web site
  - Benchmark results are one input to detailed processor analyses
  - Complete results and analyses offered in published reports
- The rest of this talk focuses on BDTI's benchmark efforts
  - Explains how we have approached the problem of benchmarking for DSP
  - Presents some of our results

 13

BDTI Benchmarking Methodology

- Benchmarks are rigorously defined
- Benchmarks are hand-optimized in assembly
- All implementations follow the same rules
  - The most important rule is that only "realistic" optimizations are allowed
- Each benchmark is independently verified for:
  - Performance
  - Functionality
  - Optimality
  - Conformance to benchmark specs

 14

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)
- Benchmarks use processor's native data format

 15

Benchmarking Limitations

- Differences in numeric accuracy are not considered
  - Benchmarks don't reflect differences in accuracy from different data word widths or formats (fixed- vs. floating-point)
- System design issues not represented
  - I/O performance, interrupts, etc.
- Inherent lack of completeness
  - Benchmark suite can't include every relevant algorithm kernel

 16

BDTI Benchmark™ Suite

Composed of a wide variety of DSP algorithm kernels.
On each benchmark, we determine:

- Cycle count
- Memory use

From the cycle counts, we derive:

- Execution time
- Energy consumption
- Cost-performance

Implementing DSP on General-Purpose Processors (GPPs)

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

 19

Drawbacks of Using High-End GPPs for DSP

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

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

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.

 20

Embedded GPPs for DSP

- GPPs designed for embedded applications are starting to address DSP needs
  - For example, 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

Benchmark Results

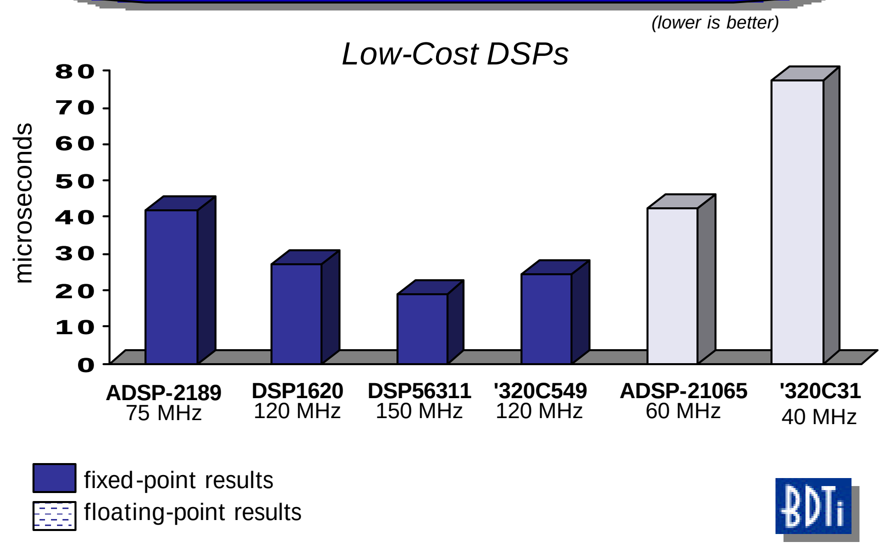
Factors Affecting Speed

- Clock rate
  - Instruction complexity
  - Pipeline depth
- Amount of work performed per cycle
  - Parallel execution units
  - VLIW
  - Superscalar
  - SIMD capabilities
  - Hardware accelerators
  - RISC-like instructions vs complex, compound instructions
  - Memory bandwidth
  - Pipeline

 23

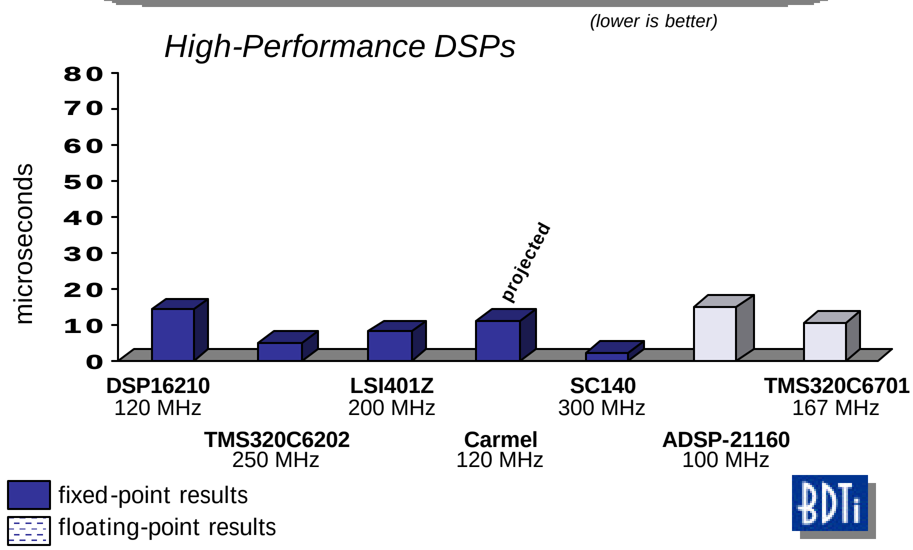
Execution Times

Complex Block FIR Filter Benchmark



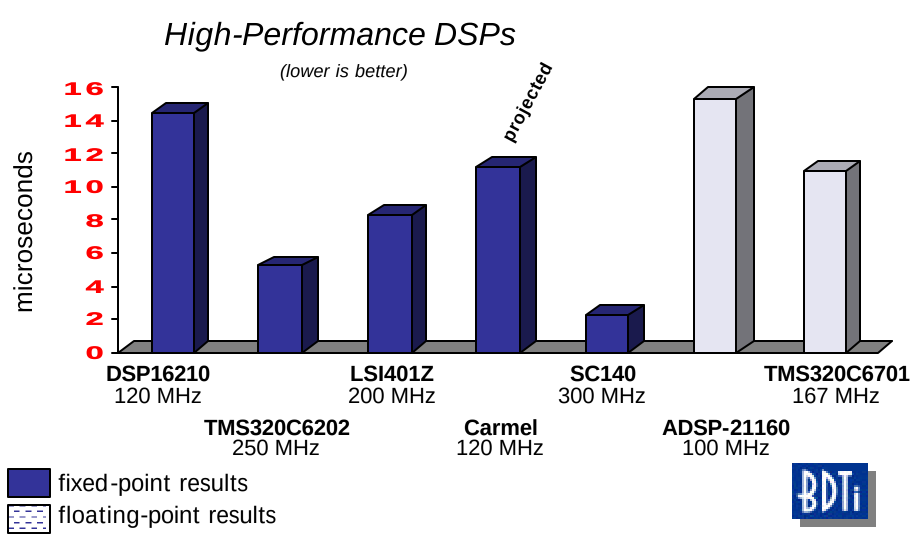
 24

Execution Times
Complex Block FIR Filter Benchmark

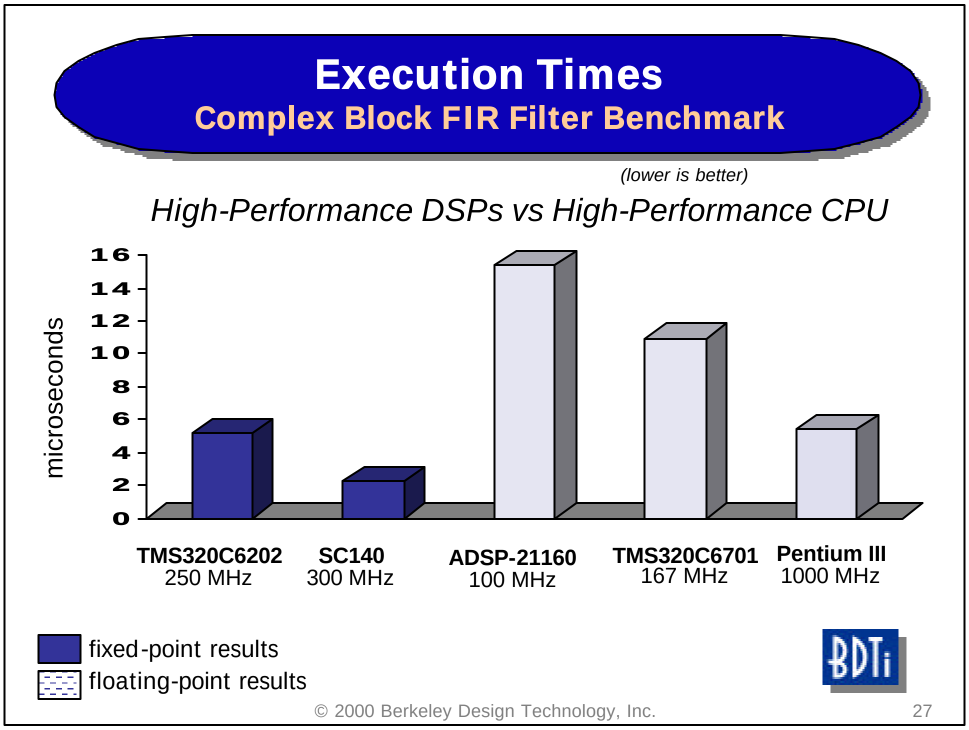


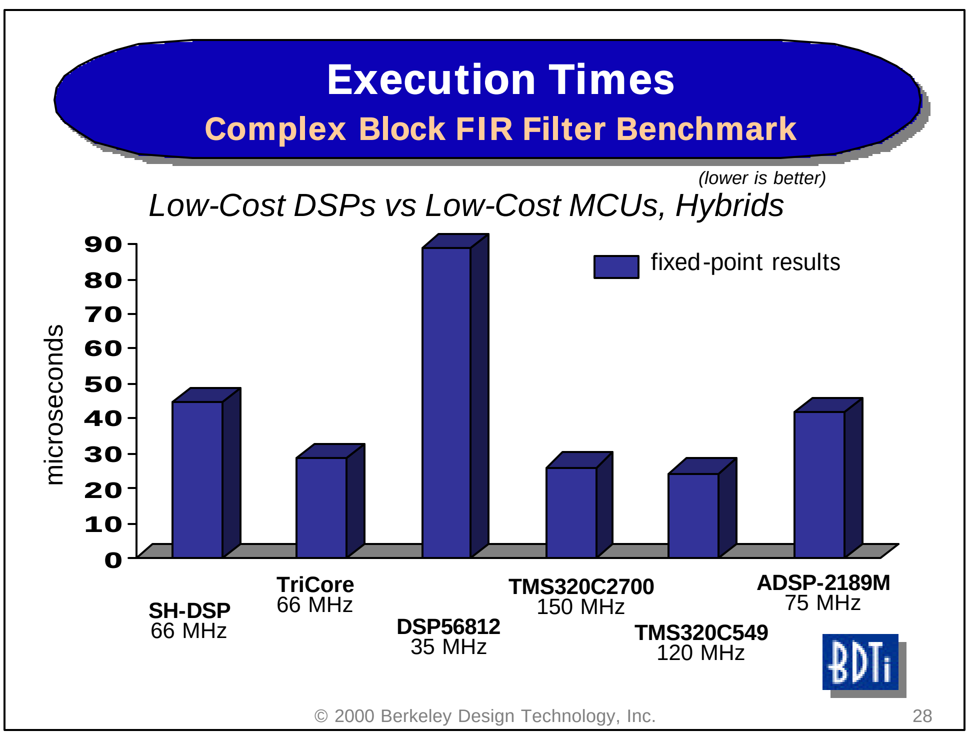
 25

Execution Times
Complex Block FIR Filter Benchmark



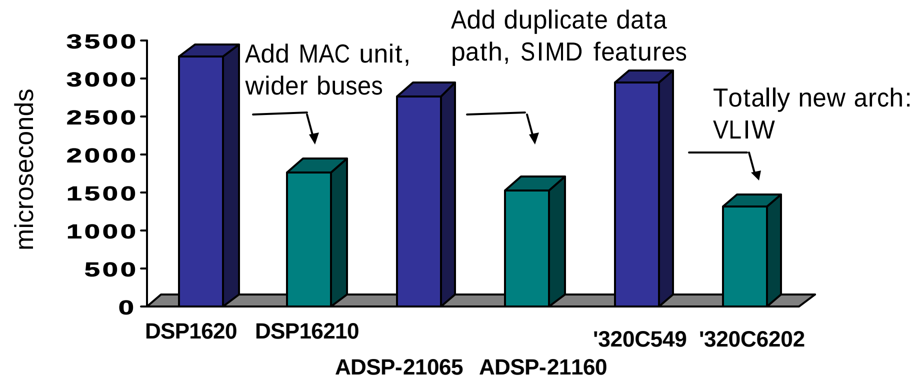
 26


Execution Times
Complex Block FIR Filter Benchmark
(lower is better)
High-Performance DSPs vs High-Performance CPU
microseconds
16
14
12
10
8
6
4
2
0
TMS320C6202
250 MHz
SC140
300 MHz
ADSP-21160
100 MHz
TMS320C6701
167 MHz
Pentium III
1000 MHz
fixed-point results
floating-point results
BDTi
© 2000 Berkeley Design Technology, Inc.
27


Execution Times
Complex Block FIR Filter Benchmark
(lower is better)
Low-Cost DSPs vs Low-Cost MCUs, Hybrids
microseconds
90
80
70
60
50
40
30
20
10
0
fixed-point results
SH-DSP
66 MHz
TriCore
66 MHz
DSP56812
35 MHz
TMS320C2700
150 MHz
TMS320C549
120 MHz
ADSP-2189M
75 MHz
BDTi
© 2000 Berkeley Design Technology, Inc.
28

Cycle Counts
Complex Block FIR Filter Benchmark

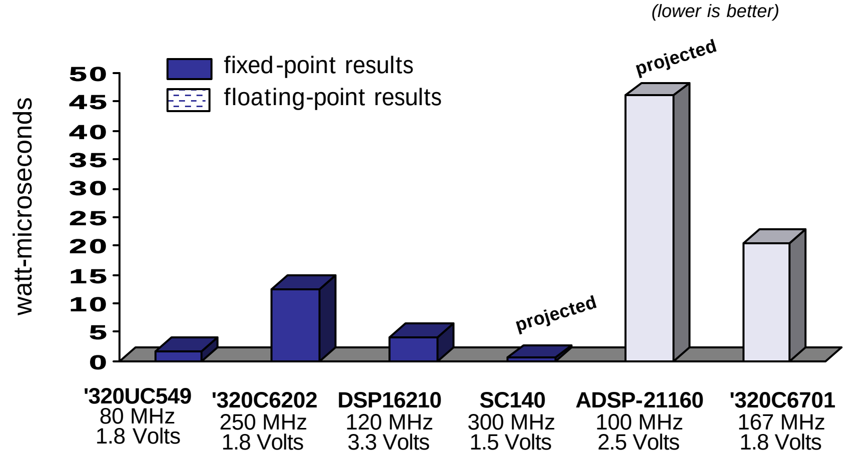
Performance Improvements in New Generations



 29

Energy Consumption
Complex Block FIR Filter Benchmark

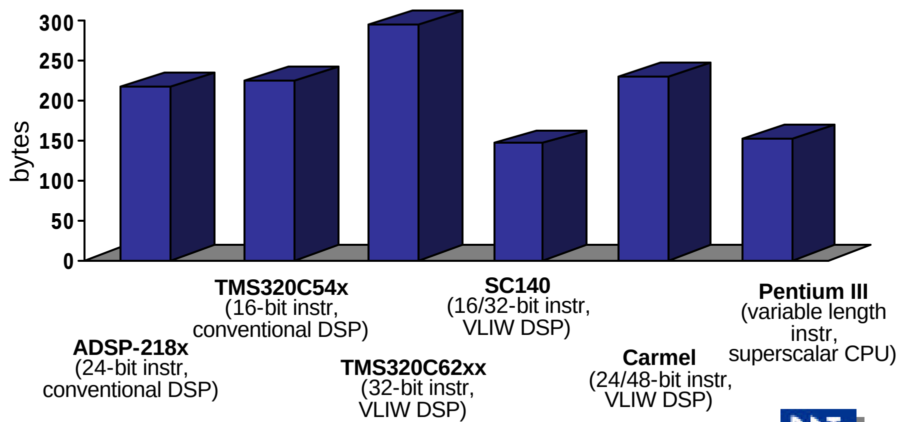
(lower is better)



 30

Memory Use
Control Benchmark

(lower is better)



What About Compilers?

- Control code, unlike DSP algorithm kernel code, is often compiled rather than hand-optimized in assembly
- Why?
  - There may be too much control code to implement in assembly
  - In control code, speed is less important than memory efficiency

Compilers and DSP Application Development

Typical development process:

1. Compile C implementation of application
2. Identify performance-critical sections
   - Speed
   - Memory usage
3. Optimize critical sections in assembly

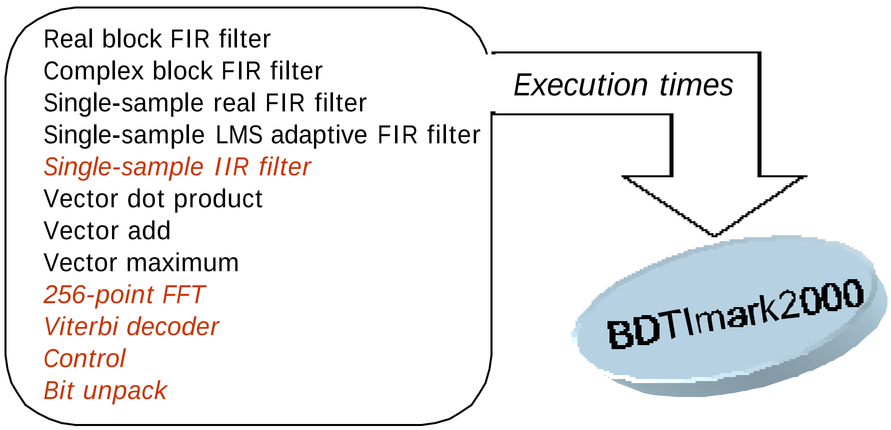
 33

Comparison of Code Density Compiled vs Assembly Code

Control Code Benchmark	**DSP A (Enhanced Conventional)**	**DSP B (VLIW)**
Assembly Code Size (Bytes)	192	296
Compiled Code Size (Bytes)	386	432
Ratio of compiled code to assembly code	2.01 : 1	1.46 : 1

 34

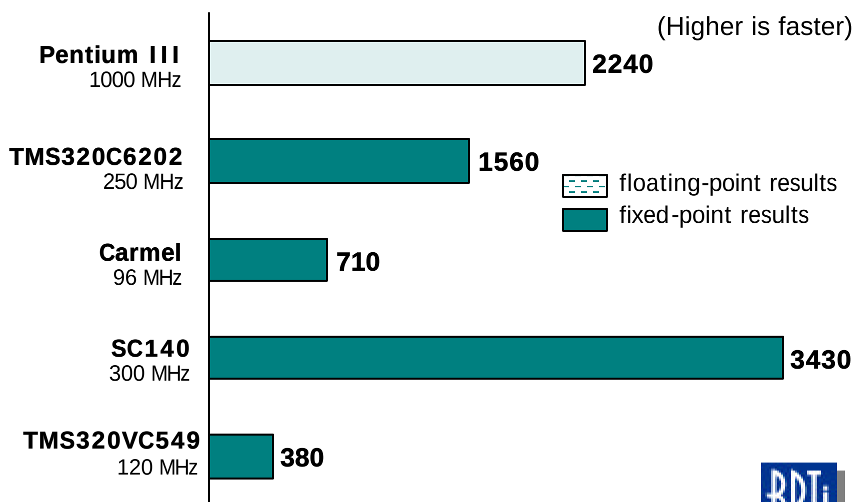
The BDTImark2000™ A Composite DSP Speed Metric



New or updated benchmarks for 2000

 35

Example BDTImark2000 Results (Preliminary)



 36

Comparing Performance with the BDTImark2000

Metric	TI TMS320C6202 (250 MHz)	TI TMS320C549 (120 MHz)	Ratio
BDTImark2000	1560	380	~4 : 1

But remember--your mileage will vary, depending on the benchmark!

 37

Conclusions

- When using benchmark results, it is essential to understand the underlying methodology (and its limitations)
- The "best" processor depends on the application
  - The best processor for a DSP task may not be a "DSP processor"
- Benchmarks must evolve to reflect current uses of processors in DSP applications
- Metrics other than execution speed may be most important
- Benchmarks don't tell the whole story

 38

For More Information...
www.BDTI.com

- White papers on DSP processor architectures and benchmarking
- Article reprints on DSP-oriented processors and apps
    - *Microprocessor Report*
    - *IEEE Spectrum*
    - *IEEE Computer* and others
- *comp.dsp* FAQ
- BDTImark2000 scores (coming soon)

