


Choosing and Using DSPs

Insight, Analysis, and Advice on Signal Processing Technology




Choosing and Using DSPs (DSP-522)

Jeff Bier
Berkeley Design Technology, Inc.
Berkeley, California USA
+1 (510) 665-1600

info@BDTI.com
<http://www.BDTI.com>


© 2006 Berkeley Design Technology, Inc.



Presentation Goals

By the end of this workshop, you should know:

- What to consider when choosing a DSP processor
- Strengths and weaknesses of the latest processors
- How to get the most out of a DSP



© 2006 BDTI

INSIGHT • ANALYSIS • ADVICE
ON SIGNAL PROCESSING TECHNOLOGY

2

© 2006 Berkeley Design Technology, Inc.



Outline

Selecting a DSP

- Selection criteria
- Assessing performance
- Selection methodology

Strengths and weaknesses of current DSPs

Software optimization techniques

Conclusions



Processor Selection Criteria

Performance on relevant tasks

- Speed
- Numeric fidelity
 - Fixed-point vs. floating-point
 - Data word size(s)
- Execution-time predictability
 - Dynamic features confound determinism
- Energy consumption
- Memory bandwidth: on-chip, off-chip
- Memory usage



Assessing Performance

Use results from relevant application modules

- More accurate than kernel benchmark mapping—if available
- Use caution! The data may be misleading or incomplete

Use kernel benchmarks & application profiles

- Useful when application data isn't available
- Use kernel benchmark results to predict application module performance

Use care with either approach

- Hazards include data types, multitasking effects ...



Assessing Performance, continued

Core CPU performance isn't enough


- Data movement can be very costly
 - Must also consider memory sizes and bandwidths
 - I/O bandwidths and overheads

Impact of software partitioning in multi-processor systems

- Must refine software architecture to predict performance

Dynamic features complicate performance prediction

Assessing energy efficiency can be very difficult



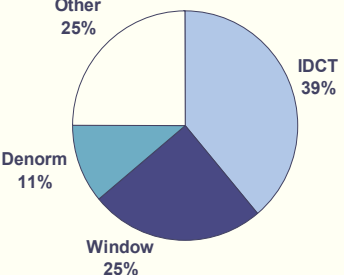
The BDTI DSP Kernel Benchmarks™

Algorithm kernels are the most computationally intensive portions of signal processing applications

Example algorithm kernels include FFTs, IIR filters, and Viterbi decoders


Application-relevant algorithm kernels are strong predictors of overall performance

About 70 architectures already benchmarked



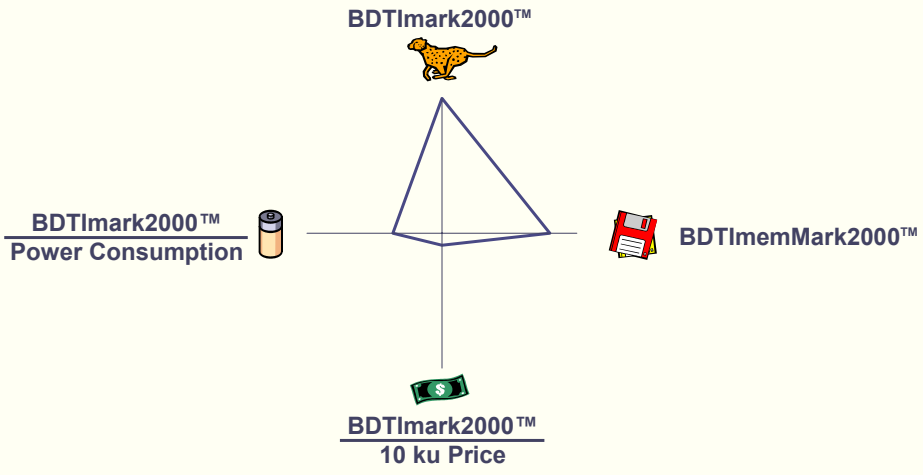
Kernel Type	Percentage
IDCT	39%
Window	25%
Other	25%
Denorm	11%

© 2006 BDTI
INSIGHT • ANALYSIS • ADVICE
ON SIGNAL PROCESSING TECHNOLOGY
7



Benchmark Results

Example: Texas Instruments 'C6455 ('C64x+, 1000 MHz)




BDTImark2000™

BDTImark2000™
Power Consumption

BDTImark2000™
10 ku Price

BDTImemMark2000™

© 2006 BDTI
INSIGHT • ANALYSIS • ADVICE
ON SIGNAL PROCESSING TECHNOLOGY
8



Processor Selection Criteria

Cost


On-chip integration

- Coprocessors
- Memory
- I/O interfaces
- Other peripherals

Packaging options

- Sizes
- Temperature ranges
- Ease of manufacture

© 2006 BDTI INSIGHT • ANALYSIS • ADVICE
ON SIGNAL PROCESSING TECHNOLOGY 9



Availability and Roadmap

Risk

- Availability; reliability of supply
 - Multi-vendor architectures a plus
- What does the errata list look like?

Roadmap

- Vendor commitment to evolving the chip, e.g., improved integration, reduced cost
- Roadmap for next-generation architectures
- Compatibility of future parts
- What is your confidence that the vendor will execute on its roadmap?

© 2006 BDTI INSIGHT • ANALYSIS • ADVICE
ON SIGNAL PROCESSING TECHNOLOGY 10



Development Considerations

Programming model complexity

Developer familiarity

Compatibility

Tools (vendor, 3rd party)

- Accurate cycle-count and memory profiling
- Visibility into cache, pipeline

Debug/development benefit from tools with:

- Peripheral and multi-processor simulation
- Non-intrusive, real-time debug



Development Considerations

Language support

- Quality of C compiler; availability of C++ compiler
- Support for assembly language optimization

Software availability

- Signal processing components
- Device drivers and other general-purpose software
- Operating systems

Hardware/software reference designs



Processor Selection Methodology

Use a hierarchical approach to make the problem manageable:

- Determine selection criteria
- Prioritize or assign weights to selection criteria
- Use critical criteria to eliminate obviously unsuitable choices
 - Begin with classes of processors
- Evaluate and rank candidates
- Weigh trade-offs among non-critical criteria
- Iterate as necessary
 - Refine criteria and analysis of candidates



Outline

Selecting a DSP

- Selection criteria
- Assessing performance
- Selection methodology

Strengths and weaknesses of new DSPs

Software optimization techniques

Conclusions



Texas Instruments TMS320C64x

The 'C62x Gets Serious Enhancements

8-issue 16-bit fixed-point architecture

- Up to four 16-bit MACs per cycle
- Special instructions and co-processors for communications and multimedia
- Compatible with 'C62x, 'C67x

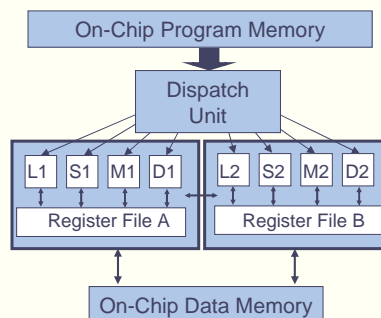
11-stage pipeline with multi-cycle latencies

Two-level cache memory system

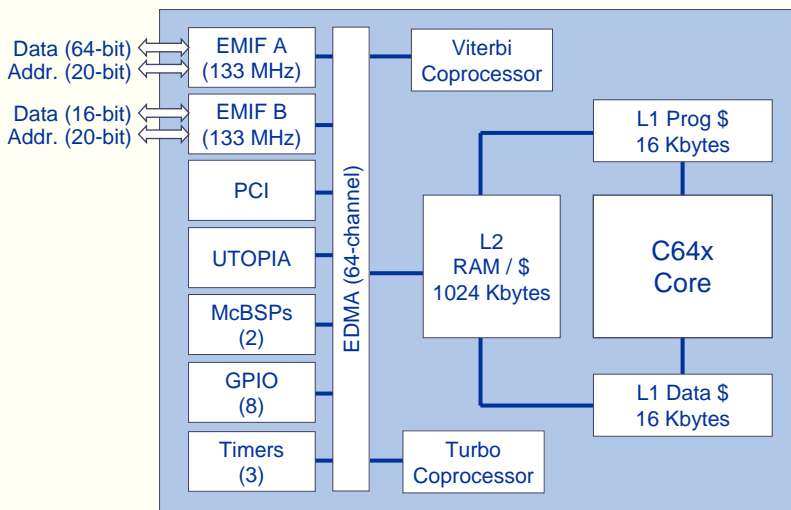
32-bit instruction set


Shipping at 1 GHz, \$189 (10 ku)

Available in ASSPs for communications and multimedia



TMS320C6416 Integration

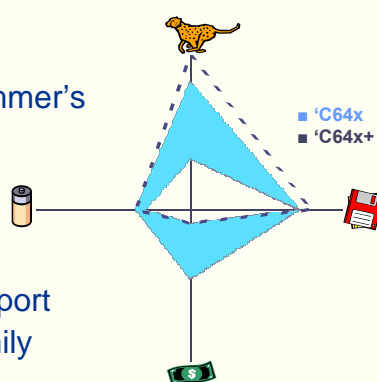





Texas Instruments TMS320C64x

Strengths and Weaknesses

- ↑ Large family spans wide range of cost, performance and on-chip integration
- ↑ Most parts are very fast
- ↓ Complex programming model
 - ↓ 'C6xxx is assembly programmer's worst nightmare
 - ↓ Caches reduce execution-time predictability
- ↑ Good integration
- ↑ Good tools and third-party support
 - ↑ Compatible with 'C6xxx family



© 2006 BDTi
INSIGHT • ANALYSIS • ADVICE
ON SIGNAL PROCESSING TECHNOLOGY
17

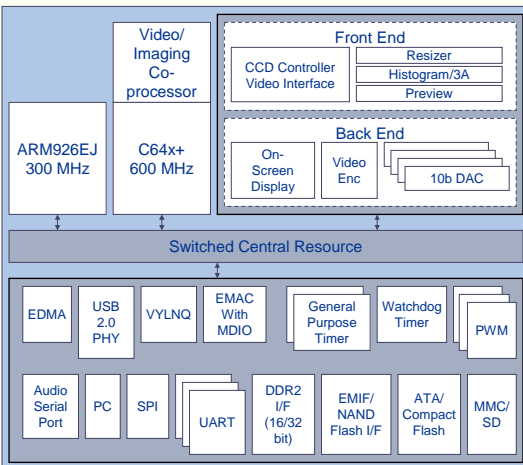


'C64x+: The Next Generation

Modest set of upgrades:

- Up to eight 16-bit MACs per cycle
- New instructions for security and communications apps
- Loop buffers improve throughput, code size
- 16/32-bit instruction set

20% faster than 'C64x
15% smaller memory footprint than 'C64x
Sampling at 1 GHz, \$259 (10 ku)
Available in ASSPs for communications and multimedia



© 2006 BDTi
INSIGHT • ANALYSIS • ADVICE
ON SIGNAL PROCESSING TECHNOLOGY
18



Analog Devices ADSP-BF53x

3-issue VLIW, 16-bit fixed-point architecture

- Up to two 16-bit MACs per cycle
- MMU and mode-dependent instructions

10-stage pipeline with single-cycle latencies

16/32-bit instruction set

Memory system configurable as cache

Speed/voltage scaling: 100 MHz/0.8 V – 750 MHz/1.4 V

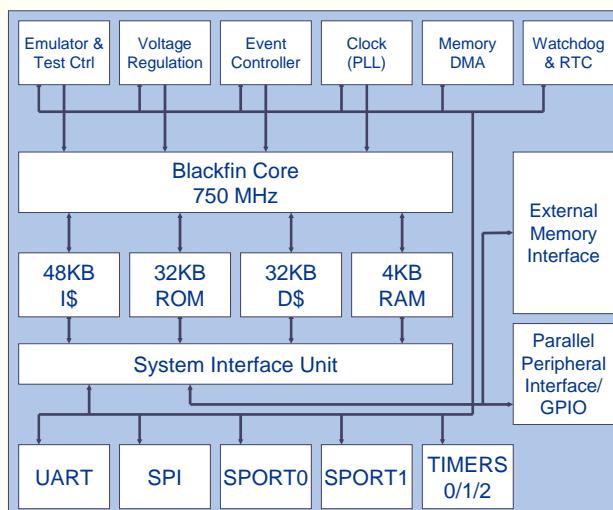
Shipping at 750 MHz, \$32 (10 ku)

- Dual-core 'BF561 shipping at 600 MHz, \$26 (10 ku)

Used in ASSPs for communications and multimedia



ADSP-BF533 On-Chip Integration



BDTi

Analog Devices ADSP-BF53x

Strengths and Weaknesses

- ↑ Excellent memory-, cost-, and energy-efficiency
 - ↑ Unusual speed/energy flexibility
- Modest speed for single-core parts
- ↑ Easy to program; good compiler target
- ↑ OS-friendly hardware features
- Sophisticated but complex memory system
- ↑ Good integration
- ↑ Good tools
- ↓ Not compatible; no legacy code base
 - ↓ Less 3rd-party support than TI DSPs

© 2006 BDTI INSIGHT • ANALYSIS • ADVICE ON SIGNAL PROCESSING TECHNOLOGY 21

BDTi

Outline

- Selecting a DSP
 - Selection criteria
 - Assessing performance
 - Selection methodology
- Strengths and weaknesses of new DSPs
- Software optimization techniques**
- Conclusions

© 2006 BDTI INSIGHT • ANALYSIS • ADVICE ON SIGNAL PROCESSING TECHNOLOGY 22

BDTi

Optimization Techniques

Optimization process:
→ Profile → Analyze → Optimize

Optimization levels

- Algorithm level
 - Either processor dependent or processor independent
- Software architecture level
- High level language (HLL) level
 - Relies heavily on the compiler
- Hand-coded assembly language level
 - Yields the best performance

© 2006 BDTI INSIGHT • ANALYSIS • ADVICE
ON SIGNAL PROCESSING TECHNOLOGY 23

BDTi

Optimization Techniques

Optimization targets

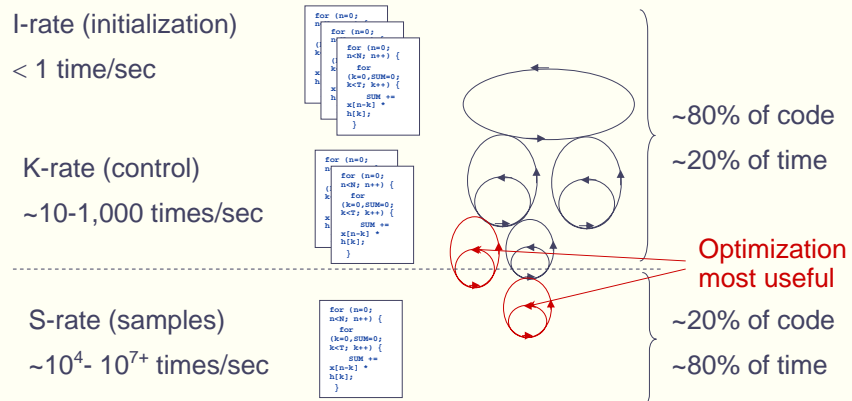
- Execution speed
 - Using more parallelism
 - **Reducing memory accesses**
 - **Avoid cache, or L1, “thrashing”**
- Memory usage
 - May conflict with optimizations for speed
- Energy consumption
 - E.g., minimize off-chip memory accesses

© 2006 BDTI INSIGHT • ANALYSIS • ADVICE
ON SIGNAL PROCESSING TECHNOLOGY 24



Profiling: Find S-rate Operations

Functions can be classified based on invocation rate:



Algorithm and Software Architecture Optimizations

Substitute one algorithm for another

Combine multiple algorithms into a single step

Select appropriate data types

- Resources are wasted when data size is bigger than needed
- CAUTION: Using data size that's too small may cause malfunctions

Memory access can be costly

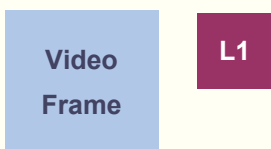
- Organize data flow to minimize cache/L1 thrashing
- Keep data in registers and re-use it
- Use large loads and stores



Memory Access Optimization

Video Processing

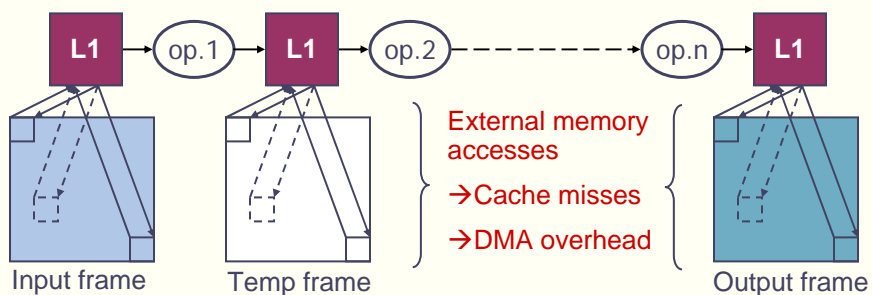
- Video frame much bigger than typical L1 memory (cache or SRAM)
 - L1: typically 10 to 100 KB
 - Frame: typically 100 KB to 1 MB+ for **each** frame



Memory Access Optimization

Default processing sequence

- Operation 1 on entire frame
- Operation 2 on entire frame...



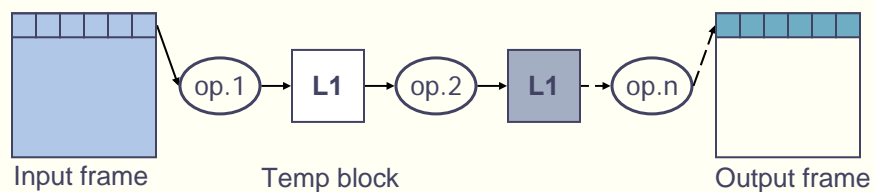


Memory Access Optimization

Observation: Most video algorithms operate on independent blocks of data (8x8, 4x4, lines ...)

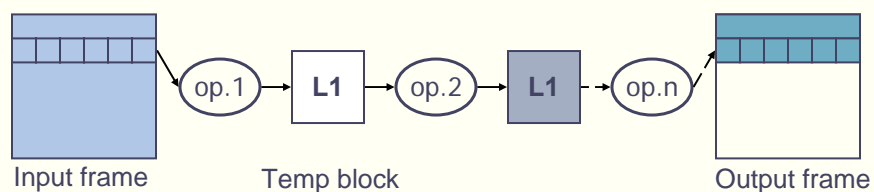
Optimization: process one block at a time through multiple algorithm steps

- Subset stays resident in L1, cutting external memory accesses



Memory Access Optimization

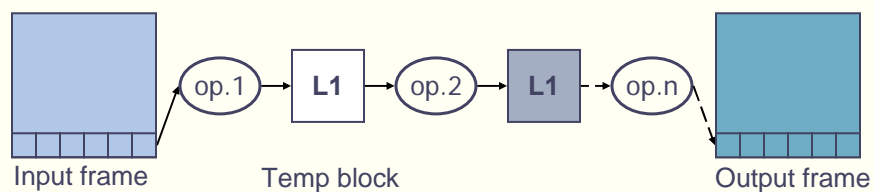
Process second block



Memory Access Optimization

Process last block

External memory accesses have been minimized



High-Level Language Optimizations

Understand compiler behavior

- Compilers struggle with parallelism
 - Compilers rarely take full advantage of VLIW and superscalar architectures
 - SIMD capabilities rarely used
- Compilers rarely use specialized instructions
- Compiled code can be full of subtle inefficiencies
 - Example: How will the compiler handle saturation?

Know the processor architecture

- Different processors require different approaches

Intrinsics are often critical to squeezing performance from the compiler



Assembly-Level Programming Tips

Know *when* to write assembly code

- To improve scheduling
- To improve register allocation
- To use full instruction set
 - SIMD
 - Special instructions
 - Creative use of instructions

Be prepared to make structural changes...

- Code: loop unrolling, software pipelining, etc.
- Data: e.g., to enable SIMD operations



Software Pipelining

Dot Product: Initial Implementation

```

...
LOOP   LDRD    r6, [r0], #8

        LDRD    r10, [r1], #8

        SUBS    r2, r2, #4
        ← 2 stall cycles
        SMLAD   r12, r6, r10, r12
        ← 1 stall cycle
        SMLAD   r12, r7, r11, r12
        BGT     LOOP
...
    
```

Effective throughput: 0.44 MACs per cycle

BDTi

Software Pipelining

Dot Product: Optimized Code

```
...
LDRD    r4, [r0], #8
LDRD    r8, [r1], #8
...
LOOP
LDRD    r6, [r0], #8
SMLAD   r12, r4, r8, r12
LDRD    r10, [r1], #8
SMLAD   r12, r5, r9, r12
SUBS    r2, r2, #8
LDRGTD  r4, [r0], #8
SMLAD   r12, r6, r10, r12
LDRGTD  r8, [r1], #8
SMLAD   r12, r7, r11, r12
BGT     LOOP
...
```

pipelined second iteration

0 stall cycles

Effective throughput: 0.80 MACs per cycle

© 2006 BDTI INSIGHT • ANALYSIS • ADVICE
ON SIGNAL PROCESSING TECHNOLOGY 35

BDTi

Outline

- Selecting a DSP
 - Selection criteria
 - Assessing performance
 - Selection methodology
- Strengths and weaknesses of new DSPs
- Software optimization techniques
- Conclusions**

© 2006 BDTI INSIGHT • ANALYSIS • ADVICE
ON SIGNAL PROCESSING TECHNOLOGY 36



Conclusions

Know what to look for in a processor

- Clearly define the application requirements
- Consider all the processor options
- Be alert for incomplete or misleading information

Know how to use benchmarks

- Today's complex DSPs require a thoughtful benchmarking approach
- Consider all the relevant metrics



Conclusions

Use a hierarchical approach to pick a processor

- Develop a hierarchy of requirements
- Start with critical criteria; iteratively narrow the field
- Expect to make trade-offs

Have a clear software optimization plan

- Start with the highest-return optimizations
- Know how to get the most from the compiler
- Know when and how to write assembly code
- Know when to use third-party software

Choosing and Using DSPs



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




6th Edition

© 2006 BDTI

INSIGHT • ANALYSIS • ADVICE
ON SIGNAL PROCESSING TECHNOLOGY

39